RZ/T2M Group

BiSS-C sample program

Summary

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

This sample program supports RZ/T2M Encoder I/F Configuration Data (BiSS) Ver.1.0 or later.

The major features of the program are listed below.

- Supports sensor mode, register access, and continuous register access
- Obtain angle and other information from encoders compliant with BiSS-C specifications (Hengstler AD36/1219AF.OCBEF, Wachendorff WDGF 58M)

Target Device

RZ/T2M

Table of Contents

1. Specifications

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

Table 1.1 Peripheral Modules and Applications

Peripheral Module	Application
BiSS-C Encoder Interface (BISS)	Communication with encoders compliant with BiSS-C specifications
Interrupt Controller (ICU)	BiSS-C Encoder Interface interrupt
General PWM Timer (GPT)	Generation of communication cycle for continuous processing and event cycle to be input to ELC
Event Link Controller (ELC)	Link events output by GPT to BiSS-C Encoder Interface
Serial Communication Interface (SCI) UART	Asynchronous communications of the SCI are used for COM port communications by USB interface. It is used for console interface of the sample program.

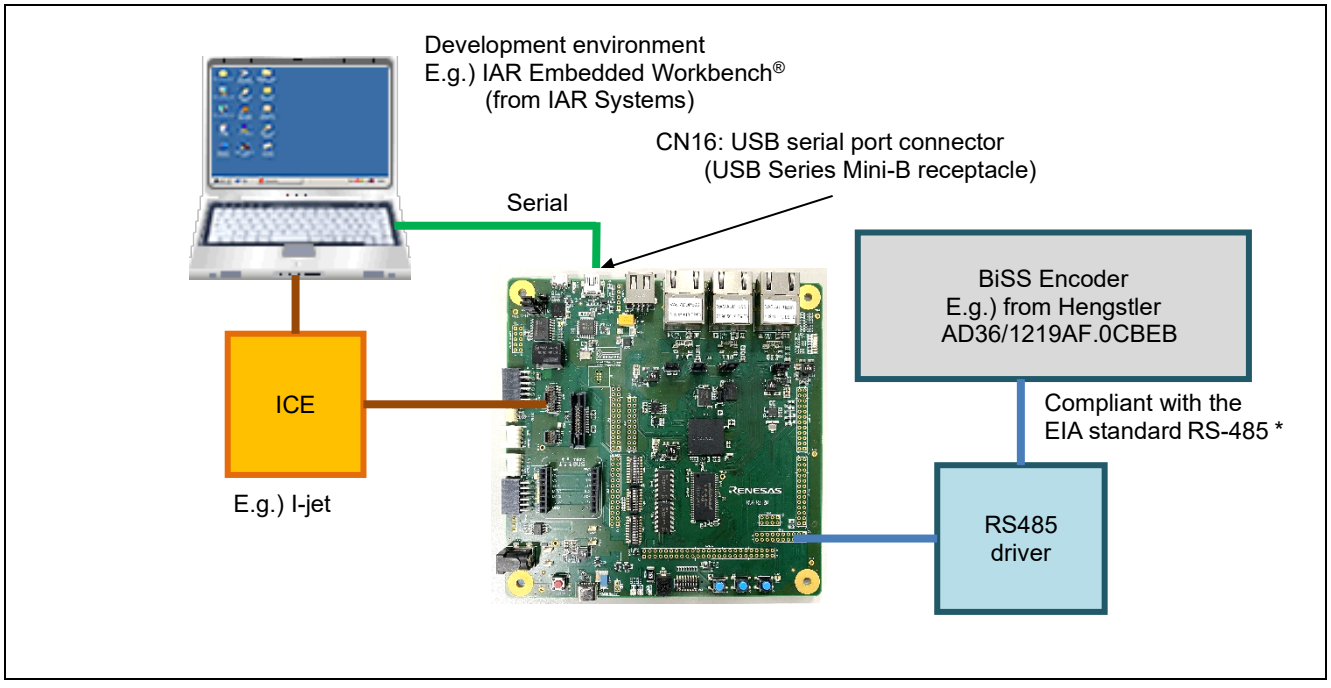


Figure 1.1 Operating Environment

Note: For allowable cable length, refer encoder manuals.

2. Operating Environment

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

Table 2.1 Operating Environment

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

Note: Refer to the RZ/T2M Group Encoder I/F BiSS-C sample program Release Note to check the version number of the integrated development environment.

3. Peripheral Modules

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

3.1 Pins

Table 3.1 lists the pins used and their functions.

Table 3.1 Pins Used and Their Functions

Channel	Port Name (Function Pin Name)	I/O Port	Input/Output	Description
BISS0	SLO0 (ENCIF0)	P01_6	Input	Sensor / Control data
	MA0 (ENCIF4)	P02_3	Output	Clock / Control data
BISS1	SLO1 (ENCIF5)	P17_3	Input	Sensor / Control data
	MA1 (ENCIF9)	P03_3	Output	Clock / Control data

4. Software

4.1 BiSS-C Driver Functions

The functions of the BiSS-C driver are listed below

1. Initializing the BiSS-C Encoder Interface
2. Transmitting requests to the BiSS-C and receiving results
3. Notification of errors in transfer from the BiSS-C

4.2 File Structures

For the file structure, refer to the RZ/T2M Group Encoder I/F BiSS-C sample program Release Note.

4.3 Functions

Table 4.1 lists the functions to be used.

Table 4.1 List of Functions

Category	Function Name	Page number
BiSS driver API functions	R_BISS_Open	7
	R_BISS_Close	7
	R_BISS_GetVersion	8
	R_BISS_Control	8
User-defined functions	callback_get_pos	13
	callback_reg_access	13
	callback_elctimer_result	14
Interrupt-handlers	bissc0_rx_int_isr	14
	bissc1_rx_int_isr	14

4.4 Specifications of API Functions

4.4.1 R_BISS_Open

R_BISS_Open	
Synopsis	Starts control of the BiSS encoder.
Header	r_biss_rzt2_if.h
Declawration	r_biss_err_t R_BISS_Open(const int32_t id, const r_biss_type_t type, r_biss_info_t* p_info);
Description	Starts control of the BiSS encoder designated by the arguments.
Arguments	id : Specifies the ID to be used. (It is defined in r_biss_rzt2_dat.h.) R_BISS0_ID : Specifies channel 0. R_BISS1_ID : Specifies channel 1. Others : Setting is not allowed. type Specifies the encoder type. R_BISS_TYPE_C should be specified. p_info Specifies the encoder information. Specify the address of the structure r_biss_info_t, which contains encoder information. (See "4.10.1(1)(a) r_biss_info_t" for details on the structure.)
Return value	R_BISS_SUCCESS : Normal termination R_BISS_ERR_INVALID_ARG : Abnormal termination (The argument id or type is not specified, or p_info is NULL.) R_BISS_ERR_ACCESS : Abnormal termination (Control has already been initiated.)
Note	Before executing this function, be sure to configure and start Multi-Protocol Encoder I/F using EC-Lib.

4.4.2 R_BISS_Close

R_BISS_Close	
Synopsis	Terminiates control of the BiSS encoder.
Header	r_biss_rzt2_if.h
Declaration	r_biss_err_t R_BISS_Close(const int32_t id);
Description	Terminates control of the BiSS encoder for the specified channel.
Arguments	id : Specifies the ID to be closed. (It is defined in r_biss_rzt2_dat.h.) R_BISS0_ID : Specifies channel 0. R_BISS1_ID : Specifies channel 1. Others : Setting is not allowed.
Return value	R_BISS_SUCCESS : Normal termination R_BISS_ERR_INVALID_ARG : Abnormal termination (The argument id is not specified.) R_BISS_ERR_ACCESS : Abnormal termination (Communication is in progress or ELC is being accepted.)
Note	Cannot Close if the encoder is transmitting/receiving.

4.4.3 R_BISS_GetVersion

R_BISS_GetVersion	
Synopsis	Obtains the version number of BiSS driver.
Header	r_biss_rzt2_if.h
Declaration	uint32_t R_BISS_GetVersion(const r_biss_type_t type);
Description	Obtains the version of the specified BiSS driver.
Arguments	type : Specifies the driver type. R_BISS_TYPE_CMN : Specifies BiSS driver common part R_BISS_TYPE_C : Specifies BiSS-C driver
Return value	The upper 16 bits store the major version, and the lower 16 bits store the minor version. Example: If the return value is 0x00010002, Ver.1.2
Supplement	If a type other than the above is specified, the return value is 0xFFFFFFFFFF.

4.4.4 R_BISS_Control

R_BISS_Control	
Synopsis	Controls BiSS encoders
Header	r_biss_rzt2_if.h
Declaration	r_biss_err_t R_BISS_Control(const int32_t id, const r_biss_cmd_t cmd, void *const p_buf);
Description	Controls BiSS encoders using the argument commands.
Arguments	id : Specify the ID to be used. (It is defined in r_biss_rzt2_dat.h.) R_BISS0_ID : Specifies channel 0. R_BISS1_ID : Specifies channel 1. Others : Setting is not allowed. cmd : Control commands For a list of commands, see "Table 4.5 Major Constants Used in the BiSS-C Driver (r_bissc_rzt2_if.h)". p_buf Store the necessary information for each control command. The information to be stored is described in "4.4.4(1) R_BISS_TYPE_C Control Command" in the operation description of each control command.
Return value	R_BISS_SUCCESS : Normal termination R_BISS_ERR_INVALID_ARG : Abnormal termination (The argument id or cmd is not specified, or a value of cmd is not supported by opening type.) R_BISS_ERR_BUSY : Abnormal termination (Operation error because acquiring position or register access is in progress.) R_BISS_ERR_ACCESS : Abnormal termination (Not opened.)
Note	Execute R_BISS_Open before executing this function.

(1) R_BISS_TYPE_C Control Command

This section describes command operation in BiSS-C mode.

(a) R_BISSC_CMD_POS

R_BISSC_CMD_POS	
Synopsis	Requests position information to BiSS encoder
Header	r_biss_rzt2_if.h
Declaration	r_biss_err_t R_BISS_Control(const int32_t id, const r_biss_cmd_t cmd, void *const p_buf);
Description	Requests BiSS encoder position information.
Arguments	id : Specify the ID to be used. (It is defined in r_biss_rzt2_dat.h.) R_BISS0_ID : Specifies channel 0. R_BISS1_ID : Specifies channel 1. Others : Setting is not allowed. cmd : Specifies R_BISSC_CMD_POS. p_buf : Request information (It is defined in r_bissc_rzt2_if.h.) Specify the pointer to the r_bissc_sensreq_t structure that contains the request information. See "4.10.1(2)(a) r_bissc_sensreq_t" for details.
Return value	R_BISS_SUCCESS : Normal termination R_BISS_ERR_INVALID_ARG : Abnormal termination (p_buf is NULL) R_BISS_ERR_BUSY : Abnormal termination (Operation error because acquiring position or register access is in progress.)

(b) R_BISSC_CMD_REG_READ

R_BISSC_CMD_REG_READ	
Synopsis	Register read request to BiSS encoder
Header	r_biss_rzt2_if.h
Declaration	r_biss_err_t R_BISS_Control(const int32_t id, const r_biss_cmd_t cmd, void *const p_buf);
Description	Reads the register value of the BiSS encoder.
Arguments	id : Specify the ID to be used. (It is defined in r_biss_rzt2_dat.h.) R_BISS0_ID : Specifies channel 0. R_BISS1_ID : Specifies channel 1. Others : Setting is not allowed. cmd : Specifies R_BISSC_CMD_REG_READ. p_buf : Request information (It is defined in r_bissc_rzt2_if.h.) Specify the pointer to the r_bissc_regreq_t structure that contains the Request information. See "4.10.1(2)(b) r_bissc_regreq_t" for details.
Return value	R_BISS_SUCCESS : Normal termination R_BISS_ERR_INVALID_ARG : Abnormal termination (p_buf is NULL.) R_BISS_ERR_BUSY : Abnormal termination (Operation error because acquiring position or register access is in progress.)
Note	If a callback (callback_get_pos) occurs to indicate data reception after this function, the command request R_BISSC_CMD_POS should be executed continuously until a callback (callback_reg_access) occurs to indicate completion of register access. (See "4.11.6(3) Register Access (Read / Write) Sequence".)

(c) R_BISSC_CMD_REG_WRITE

R_BISSC_CMD_REG_WRITE	
Synopsis	Register write request to BiSS encoder
Header	r_biss_rzt2_if.h
Declaration	r_biss_err_t R_BISS_Control(const int32_t id, const r_biss_cmd_t cmd, void *const p_buf);
Description	Writes the specified data to the BiSS encoder register.
Arguments	id : Specify the ID to be used. (It is defined in r_biss_rzt2_dat.h.) R_BISS0_ID : Specifies channel 0. R_BISS1_ID : Specifies channel 1. Others : Setting is not allowed. cmd : Specifies R_BISSC_CMD_REG_WRITE. p_buf : Request information (It is defined in r_bissc_rzt2_if.h.) Specify the pointer to the r_bissc_regreq_t structure that contains the Request information. See "4.10.1(2)(b) r_bissc_regreq_t" for details.
Return value	R_BISS_SUCCESS : Normal termination R_BISS_ERR_INVALID_ARG : Abnormal termination (p_buf is NULL.) R_BISS_ERR_BUSY : Abnormal termination (Operation error because acquiring position or register access is in progress.)
Note	If a callback (callback_get_pos) occurs to indicate data reception after this function, the command request R_BISSC_CMD_POS should be executed continuously until a callback (callback_reg_access) occurs to indicate completion of register access. (See "4.11.6(3) Register Access (Read / Write) Sequence”.)

(d) R_BISS_CMD_REG_SREAD

R_BISS_CMD_REG_SREAD	
Synopsis	Continuous register read request to BiSS encoder
Header	r_biss_rzt2_if.h
Declaration	r_biss_err_t R_BISS_Control(const int32_t id, const r_biss_cmd_t cmd, void *const p_buf);
Description	Reads the BiSS encoder register value continuously.
Arguments	id : Specify the ID to be used. (It is defined in r_biss_rzt2_dat.h.) R_BISS0_ID : Specifies channel 0. R_BISS1_ID : Specifies channel 1. Others : Setting is not allowed. cmd : Specifies R_BISS_CMD_REG_SREAD. p_buf : Request information (It is defined in r_bissc_rzt2_if.h.) Specify the pointer to the r_bissc_regreq_t structure that contains the Request information. See "4.10.1(2)(b) r_bissc_regreq_t" for details.
Return value	R_BISS_SUCCESS : Normal termination R_BISS_ERR_INVALID_ARG : Abnormal termination (p_buf is NULL.) R_BISS_ERR_BUSY : Abnormal termination (Operation error because acquiring position or register access is in progress.)
Note	If a callback (callback_get_pos) occurs to indicate data reception after this function, the command request R_BISSC_CMD_POS should be executed continuously until a callback (callback_reg_access) occurs to indicate completion of register access. (See "4.11.6(4) Continuous register access (Read / Write) Sequence”.) To stop the continuous register read process, execute the stop request of the continuous register access (R_BISSC_CMD_REG_SSTOP).

(e) R_BISSC_CMD_REG_SWRITE

R_BISSC_CMD_REG_SWRITE	
Synopsis	Continuous register write request to BiSS encoder
Header	r_biss_rzt2_if.h
Declaration	r_biss_err_t R_BISS_Control(const int32_t id, const r_biss_cmd_t cmd, void *const p_buf);
Description	Writes the specified data to the BiSS encoder register continuously.
Arguments	id : Specify the ID to be used. (It is defined in r_biss_rzt2_dat.h.) R_BISS0_ID : Specifies channel 0. R_BISS1_ID : Specifies channel 1. Others : Setting is not allowed. cmd : Specifies R_BISSC_CMD_REG_SWRITE. p_buf : Request information (It is defined in r_bissc_rzt2_if.h.) Specify the pointer to the r_bissc_regreq_t structure that contains the Request information. See "4.10.1(2)(b) r_bissc_regreq_t" for details.
Return value	R_BISS_SUCCESS : Normal termination R_BISS_ERR_INVALID_ARG : Abnormal termination (p_buf is NULL.) R_BISS_ERR_BUSY : Abnormal termination (Operation error because acquiring position or register access is in progress.)
Note	If a callback (callback_get_pos) occurs to indicate data reception after this function, the command request R_BISSC_CMD_POS should be executed continuously until a callback (callback_reg_access) occurs to indicate completion of register access. (See "4.11.6(4) Continuous register access (Read / Write) Sequence".) To stop the continuous register write process, execute the stop request of the continuous register access (R_BISSC_CMD_REG_SSTOP).

(f) R_BISSC_CMD_REG_SSTOP

R_BISSC_CMD_REG_SSTOP	
Synopsis	Request to stop continuous register access to BiSS encoder
Header	r_biss_rzt2_if.h
Declaration	r_biss_err_t R_BISS_Control(const int32_t id, const r_biss_cmd_t cmd, void *const p_buf);
Description	Stop continuous register access to BiSS encoder
Arguments	id : Specify the ID to be used. (It is defined in r_biss_rzt2_dat.h.) R_BISS0_ID : Specifies channel 0. R_BISS1_ID : Specifies channel 1. Others : Setting is not allowed. cmd : Specifies R_BISSC_CMD_REG_SSTOP. p_buf : Not used (Specifies NULL.)
Return value	R_BISS_SUCCESS : Normal termination R_BISS_ERR_INVALID_ARG : Abnormal termination (id is invalid.) R_BISS_ERR_ACCESS : Abnormal termination (Not operable because continuous register access is not working.)
Note	This command is valid only during continuous register access (read/write). If a callback (callback_get_pos) occurs to indicate data reception after this function, the command request R_BISSC_CMD_POS should be executed continuously until a callback (callback_reg_access) occurs to indicate completion of register access. When continuous register accesses are stopped, a callback (callback_reg_access) indicating the completion of register accesses occurs and the result of the access information before the termination is set. (See "4.11.6(6) Continuous Register Access (Read / Write) Stop Sequence".)

(g) R_BISSC_CMD_REG_FSTOP

R_BISSC_CMD_REG_FSTOP	
Synopsis	Request to stop register access and continuous register access to BiSS encoder
Header	r_biss_rzt2_if.h
Declaration	r_biss_err_t R_BISS_Control(const int32_t id, const r_biss_cmd_t cmd, void *const p_buf);
Description	Stop register access and continuous register access to BiSS encoder.
Arguments	id : Specify the ID to be used. (It is defined in r_biss_rzt2_dat.h.) R_BISS0_ID : Specifies channel 0. R_BISS1_ID : Specifies channel 1. Others : Setting is not allowed. cmd : Specifies R_BISSC_CMD_REG_FSTOP. p_buf : Not used (Specifies NULL.)
Return value	R_BISS_SUCCESS : Normal termination R_BISS_ERR_INVALID_ARG : Abnormal termination (id is invalid.)
Note	This command forces register access and continuous register access to stop. Use this command when the status of register access completion is not returned from the encoder during execution of register access and continuous register access. Execute the position information request command (R_BISSC_CMD_POS) after executing this command to stop register access. Depending on the timing, a callback (callback_reg_access) indicating the completion of register access may occur after this function is executed.

4.5 User-defined Functions Definition

4.5.1 callback_get_pos

callback_get_pos		
Synopsis	Notification of position information acquisition results from BiSS-C encoder	
Header	-	
Declaration	static void callback_get_pos(void *p_result);	
Description	Callback function registered with the R_BISS_Control() function. This function notifies the result of position information acquisition. This function is the context of an interrupt handler. To ensure the responsiveness of the interrupt, be sure to return promptly. The function name is an example and can be set freely.	
Arguments	p_result	: Pointer to the r_bissc_sensordt_t structure that stores the result of the position information acquisition. See "4.10.1(2)(c) r_bissc_sensordt_t". (It is defined in r_bissc_rzt2_if.h.) This structure can be referenced until the next execution of the R_BISS_Control() function with a command excluding R_BISSC_CMD_REG_SSTOP and R_BISSC_CMD_REG_FSTOP.
Return value	None	

4.5.2 callback_reg_access

callback_reg_access		
Synopsis	Notification of register read/write results from BiSS-C encoder	
Header	-	
Declaration	static void callback_reg_access(void *p_result);	
Description	Callback function registered with the R_BISS_Control() function. This function notifies the result of the encoder's register access request. This function is the context of the interrupt handler. To ensure the responsiveness of the interrupt, return promptly. The function name is an example and can be set freely.	
Arguments	p_result	: Pointer to the r_bissc_regdata_t structure that contains the register access result. See "4.10.1(2)(d) r_bissc_regdata_t". (It is defined in r_bissc_rzt2_if.h.) This structure can be referenced until executing following register access command (R_BISSC_CMD_REG_READ / R_BISSC_CMD_REG_WRITE / R_BISSC_CMD_REG_SREAD / R_BISSC_CMD_REG_SWRITE).
Return value	None	
Note	When an error occurs during register access / continuous register access, this callback notifies the error even during register access. If the result sent/received is R_BISSC_REQ_ERR, the register access process is terminated because an error has occurred. If the result sent/received is R_BISSC_REQ_ERR_WR, an incorrect value may have been written to the register due to the occurrence of an error. Write the correct value to the register again. (See "4.11.6(5) Continuous Register Access (Read / Write) Sequence (Error on the way)".)	

4.5.3 callback_elctimer_result

callback_elctimer_result		
Synopsis	Notification of position information acquisition from BiSS-C encoder (ELC timer mode)	
Header	-	
Declaration	static void callback_elctimer_result(void *p_result);	
Description	Callback function registered with the R_BISS_Control() function to notify the acquisition of position information while operating in ELC timer mode. This function is the context of the interrupt handler. To ensure the responsiveness of the interrupt, return promptly. The function name is an example and can be freely set.	
Arguments	p_result	: Pointer to the r_bissc_sensordt_t structure that stores the result of position information acquisition. See "4.10.1(2)(c) r_bissc_sensordt_t". (It is defined in r_bissc_rzt2_if.h.) This structure can be referenced until the next execution of the R_BISS_Control() function with a command excluding R_BISSC_CMD_REG_SSTOP and R_BISSC_CMD_REG_FSTOP.
Return value	None	

4.6 Interrupt Handlers

4.6.1 bissc0_rx_int_isr

bissc0_rx_int_isr	
Synopsis	BiSS-C data transmission/reception completion interrupt handler for channel 0
Header	-
Declaration	void bissc0_rx_int_isr(void);
Description	Interrupt handler for data send/receive from BiSS-C encoder / register access complete / error reception interrupt. Reads the value of the interrupt register and checks the status.
Arguments	None
Return value	None

4.6.2 bissc1_rx_int_isr

bissc1_rx_int_isr	
Synopsis	BiSS-C data transmission/reception completion interrupt handler for channel 1
Header	-
Declaration	void bissc1_rx_int_isr(void);
Description	Interrupt handler for data send/receive from BiSS-C encoder / register access complete / error reception interrupt. Reads the value of the interrupt register and checks the status.
Arguments	None
Return value	None

4.7 List of Interrupts Used

Table 4.2 shows the interrupts used by the BiSS-C driver.

Table 4.2 Interrupts Used by BiSS-C Driver

Interrupt	ID	Synopsis
ENCIF_INT0	372	Interrupt is generated by the communication status of channel 0
ENCIF_INT4	376	Interrupt is generated by the communication status of channel 1

4.8 List of Constants/Error Code

Table 4.3 lists the major constants used by the BiSS driver (r_biss_rzt2_if.h). Table 4.4 shows the major constants used in the BiSS driver (r_biss_rzt2_dat.h), Table 4.5 shows the major constants used in the BiSS-C driver (r_bissc_rzt2_if.h) and Table 4.6 shows the major constants used in the BiSS-C driver (r_bissc_rzt2_config.h).

Table 4.3 Major Constants Used in the BiSS Driver (r_biss_rzt2_if.h)

Constant	Set value	Contents	
R_BISS_TYPE_CMN	0	Encoder or driver type	Select BiSS common (for R_BISS_GetVersion function use)
R_BISS_TYPE_C	2		Select BiSS-C
R_BISS_SUCCESS	0	BiSS driver error code	Normal termination
R_BISS_ERR_INVALID_ARG	-1		Abnormal arguments
R_BISS_ERR_BUSY	-2		API cannot be executed
R_BISS_ERR_ACCESS	-3		API execution order error

Table 4.4 Major Constants Used in the BiSS Driver (r_biss_rzt2_dat.h)

Constant	Set value	Content
R_BISS0_ID	0x01	BISS channel 0 ID (R_ECL_CH_0)
R_BISS1_ID	0x02	BISS channel 1 ID (R_ECL_CH_1)
R_BISS_FREQ	10000UL	BISS I/F Operating frequency (R_ECL_FREQ_10000KHZ)

Table 4.5 Major Constants Used in the BiSS-C Driver (r_bissc_rzt2_if.h)

Constant	Set value	Content	
R_BISSC_SLAVE_0	0	Slave ID (0 to 7)	
R_BISSC_SLAVE_1	1		
R_BISSC_SLAVE_2	2		
R_BISSC_SLAVE_3	3		
R_BISSC_SLAVE_4	4		
R_BISSC_SLAVE_5	5		
R_BISSC_SLAVE_6	6		
R_BISSC_SLAVE_7	7		
R_BISSC_CMD_POS	0xC0000001	command	Position information acquisition request
R_BISSC_CMD_REG_READ	0xC0000002		Register value read (1 Byte)
R_BISSC_CMD_REG_WRITE	0xC0000003		Register value write (1 Byte)
R_BISSC_CMD_REG_SREAD	0xC0000004		Register value read (consecutive)
R_BISSC_CMD_REG_SWRITE	0xC0000005		Register value write (consecutive)
R_BISSC_CMD_REG_SSTOP	0xC0000006		Stop continuous register read/write
R_BISSC_CMD_REG_FSTOP	0xC0000007		Stop register read/write, continuous register read/write
R_BISSC_CMD_MAX	0xC0000007		Maximum value of command
R_BISSC_SUCCESS	0	Transceiver results	Normal termination of data transmission
R_BISSC_REQ_ERR	-1		Data transceiver error occurred
R_BISSC_REQ_ERR_WR	-2		Data transceiver error occurred and Unauthorized register access occurs
R_BISSC_BR_10MHZ	0x13	Value to be set to the bit rate setting register (BR1)	
R_BISSC_BR_8_33MHZ	0x17		
R_BISSC_BR_4MHZ	0x31		
R_BISSC_BR_2_5MHZ	0x4F		
R_BISSC_BR_1MHZ	0xC7		
R_BISSC_BR_400KHZ	0x17C		
R_BISSC_BR_299_40KHZ	0x1A6		
R_BISSC_BR_200KHZ	0x1F9		
R_BISSC_BR_100KHZ	0x27C		
R_BISSC_BR_80_12KHZ	0x29B		
R_BISSC_BR_NUM	10	Number of the bit rate setting register definitions	
R_BISSC_ELC_DISABLE	0	Value to set to the ELCIN control bit (ELCINE)	
R_BISSC_ELC_ENABLE	1		

Table 4.6 Major Constants Used in the BiSS-C Driver (r_bissc_rzt2_config.h)

Constant	Set value	Content
BISSC_ISR_PRI0	11	BiSS-C I/F transmit/receive interrupt priority for channel 0 (0 to 15)
BISSC_ISR_PRI1	11	BiSS-C I/F transmit/receive interrupt priority for channel 1 (0 to 15)

4.9 List of Fixed Width Integers

Table 4.7 lists the fixed-width integers used in the sample code. The fixed-width integers used in the sample code are defined in the standard library.

Table 4.7 Fixed-width Integers Used in the Sample Code

Symbol	Contents
int8_t	Signed 8-bit integer
int16_t	Signed 16-bit integer
int32_t	Signed 32-bit integer
int64_t	Signed 64-bit integer
uint8_t	Unsigned 8-bit integer
uint16_t	Unsigned 16-bit integer
uint32_t	Unsigned 32-bit integer
uint64_t	Unsigned 64-bit integer

4.10 Structures, Unions, and Enumerated Types

4.10.1 Structures

(1) BiSS common use structures

(a) r_biss_info_t

Initialization information for BiSS control unit

```
typedef struct
{
    uint16_t        clk_freq;       Clock frequency setting to be sent to the encoder (80 kHz to 10 MHz) *1
    uint16_t        clk_re_freq;    Not used
    r_biss_pos_crc_t crc_info;      CRC information (For BiSS-C position information acquisition)
    uint8_t         mtdata_size;    Multi-turn data size setting (0 to 31) *2
    uint8_t         stdata_size;    Single-turn data size setting (0 to 31) *2
    uint8_t         aldata_size;    Alignment data size setting (0 to 31) *2
} r_biss_info_t
```

Note: 1. For details on the setting values, see Bit Rate Setting Register 1 (BR1) in the RZ/T2M Group BiSS Interface Application Note.
2. For setting values, check the specifications of the encoder to be used.

(b) r_biss_pos_crc_t

BiSS-C CRC information when acquiring position information

```
typedef struct
{
    uint8_t         crc_size;       CRC size setting (0 to 16) *1
    uint16_t        crc_polynomial; CRC generator polynomial setting *1
    uint16_t        crc_start_value; CRC initial value *1
} r_biss_pos_crc_t
```

Note: 1. For setting values, check the specifications of the encoder to be used.
The default values for both AD36/1219AF.0CBEB and WDGF 58M are shown below.
CRC size 6 bits
CRC generator polynomial $X^6 + X^1 + 1$
CRC initial value 0
For details on the setting values, see the CRC data size setting bit (CRC1SIZ) in the data size setting register (SIZE), the CRC generator polynomial setting register (CPOLY), and the CRC initial value setting register (CINIT) in the RZ/T2M Group BiSS Interface Application Note.

(2) Dedicated to BiSS-C structures

(a) r_bissc_sensreq_t

Request information when acquiring position information from a BiSS-C compliant encoder

```
typedef struct
{
    uint8_t         slave_id;       Slave ID (0 to 7)
                                    Specifies the Slave ID. If the encoder connection is point to point, it
                                    should be fixed to 0.
    uint32_t        timeout_clk;    Specify watchdog timer setting value
                                    For setting values, see the RZ/T2M Group BiSS Interface
                                    Application Note.
    uint8_t         elc_enable;     Specify enable or disable the ELC event input trigger.
                                    (0: ELC event input trigger disabled, 1: ELC event input trigger
                                    enabled)
    r_biss_isr_cb_t sdresult_cb;    Pointer to callback function to notify the result of position
                                    information acquisition.
                                    See "4.5.1 callback_get_pos" for details. If NULL is specified, no
                                    callback occurs.
} r_bissc_sensreq_t
```

(b) r_bissc_regreq_t

Request information when accessing BiSS-C compliant encoder registers

```
typedef struct
{
    uint8_t         slave_id;       Slave ID (0 to 7)
                                    Specifies the Slave ID. If the encoder connection is point to point, it
                                    should be fixed to 0.
    uint32_t        timeout_clk;    Specify watchdog timer setting value.
                                    For setting values, see the RZ/T2M Group BiSS Interface
                                    Application Note.
    uint8_t         regdtnum;       Register read/write count (1 to 64 Bytes)
                                    Sets the byte for register reads or writes; set to 1 for 1-byte register
                                    reads or writes (R_BISSC_CMD_REG_READ,
                                    R_BISSC_CMD_REG_WRITE).
    uint8_t         regaddress;     Specifies the register address of the encoder. (0 to 127)
                                    For details on register addresses, refer to the used encoder
                                    manual.
    uint8_t         *p_regdata;     Sets the data to be written to the encoder registers. During register
                                    read, the read result is written.
    r_biss_isr_cb_t sdresult_cb;    Pointer to callback function to notify the result of position
                                    information acquisition.
                                    See "4.5.1 callback_get_pos" for details. If NULL is specified, no
                                    callback occurs.
    r_biss_isr_cb_t rdresult_cb;    Pointer to a callback function that will be notified of the register
                                    access result.
                                    See "4.5.2 callback_reg_access" for details. If NULL is specified,
                                    no callback occurs.
} r_bissc_regreq_t
```

(c) r_bissc_sensordt_t

Status of position information reception from BiSS-C encoder

```
typedef struct
{
    r_bissc_req_err_t   result;     Receive result *1
                                    See "4.8 List of Constants/Error Code" for details.
    uint32_t            stdata;     Single turn data *2
    uint32_t            mtdata;     Multi turn data *2
    bool                timeout;    Timeout error *3
    bool                crc1_err;   CRC error (Position information, encoder state) *3
    bool                alarm;      Encoder alarm *3
    bool                warning;    Encoder warning *3
    bool                notready;   Register-not-ready bit (true: busy state) *4
} r_bissc_sensordt_t
```

Note: 1. The receive result is treated as a transmission/reception error if either timeout or crc1_err is true. Alarms and warnings are not treated as data reception errors. Please refer to the manual for each encoder.
2. The result received is set to the value even if an error excluding timeout error occurs.
3. Sets true when an error, alarm, or warning occurs.
4. If register access does not complete while this flag has been set, reset or retry after forced stop command execution (R_BISSC_CMD_REG_FSTOP).

(d) r_bissc_regdata_t

Results of BiSS-C encoder register access

```
typedef struct
{
    r_bissc_req_err_t   result;     Transmission/reception results *1
                                    See "4.8 List of Constants/Error Code" for details.
    uint8_t             idl;        ID-Lock Data
                                    E.g.) One encoder connected:   0b00000001
                                          Two encoders connected:  0b00000011
    uint8_t             combyte;    Number of bytes of register access completed *2
    bool                timeout;    Timeout error *3
    bool                crc1_err;   CRC error (position information, encoder state) *3
    bool                crc2_err;   CRC error (register data) *3
    bool                alarm;      Encoder alarm *3
    bool                warning;    Encoder warning *3
    bool                regw_err;   Register write error *3
    bool                readbit_err;  Read bit data error *3
    bool                writebit_err; Write-bit data error *3
    bool                stopbit_err;  Stop-bit information of the register access
                                      (true: Next register to access is not exist, or address is not
                                    specified by automatic inclement.)
    bool                cds_err;    After ID-Lock data error
                                      (true: After receiving 8-bits of the IDL, CDS value before the
                                    R-bit is invalid in the register access.)
} r_bissc_regdata_t
```

Note: 1. Transmission/reception results are treated as error when any one of following conditions, timeout, crc1_err, crc2_err, regw_err, readbit_err, writebit_err, cds_err is true. Alarm, warning and stop-bit information are not treated as errors. The conditions for the occurrence of alarm and warning depend on the specifications of each encoder. Please refer to the manual for used encoder.
2. If an error excluding timeout error occurs, the number of bytes accessed up to before the error occurs is set.
3. Sets true when an error, alarm or warning occurs.

4.10.2 Unions

Not used.

4.10.3 Enumerator

Not used.

4.11 Description of Sample Program

4.11.1 Operation Outline

This sample program performs the following process.

1) Get position information from BiSS-C encoder by commands entered from the console.
2) Read/write to BiSS-C encoder registers by commands entered from the console
3) Periodically obtain position information from the BiSS-C encoder using the ELC event input trigger function.
4) Display the data received from the BiSS-C encoder on the console

(1) System Block Diagram

Figure 4.1 shows a system block diagram.

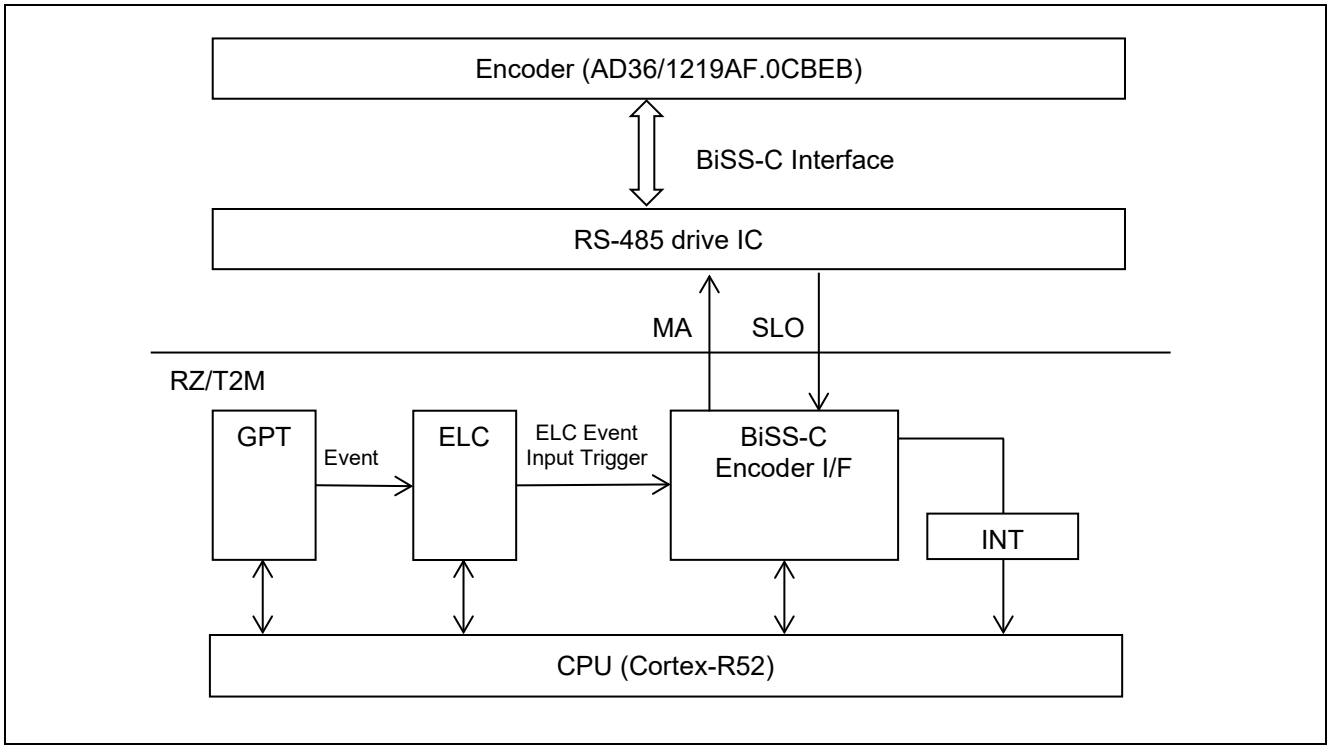


Figure 4.1 System Block Diagram

(2) Software Structure

Figure 4.2 shows software structure.

The BiSS driver has a BiSS driver common part, an opening process part consisting of the R_BISS_Open function, a closing process part consisting of the R_BISS_Close function, a sending requests part consisting of the R_BISS_Control function, and a data reception part (interrupt handler) consisting of callback functions.

The sample program has a BISS-C driver controller part which controls the BiSS-C driver through the BiSS driver common part and sends requests, and a results indication part (callback) for indicating the results of data transmission and reception.

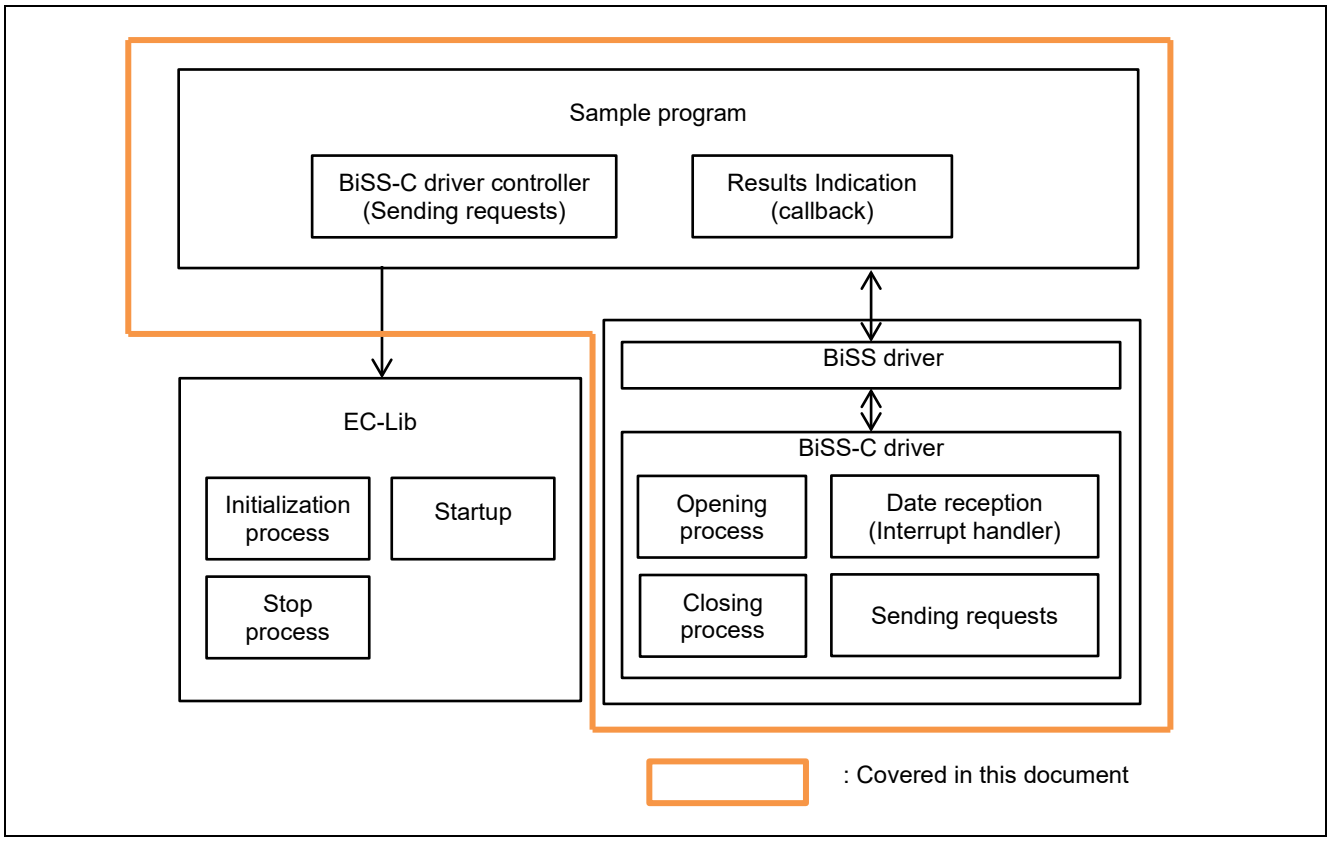


Figure 4.2 Software Structure

4.11.2 Sample Program Functions

Table 4.8 lists the major functions of the sample program.

Table 4.8 Major Functions of the Sample Program

Function Name	Page Number
hal_entry	25
enc_main	25
biss_req_send	25
bissc_get_pos	25
bissc_position	26
bissc_cmd_read	26
bissc_cmd_write	26
bissc_trans_timer	26
bissc_cmd_elctimer	27
bissc_cmd_elcstop	27
bissc_exit	27
reg_acc_req	27
callback_get_pos	28
callback_reg_access	28
callback_elctimer_result	28

4.11.3 Specifications of Sample Program Functions

(1) hal_entry

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

(2) enc_main

enc_main		
Synopsis	Main function of the BiSS sample program	
Header	-	
Declaration	int32_t enc_main(uint8_t ch);	
Description	This is the main function of the BiSS sample program. See “4.11.5(1) Flowchart of enc_main” for details.	
Arguments	ch	Encoder channel number 0: specify channel 0, 1: specify channel 1
Return value	0	: Normal termination
	Others	: Abnormal termination (error code of the encoder interface)

(3) biss_req_send

biss_req_send		
Synopsis	Request operating function to BiSS-C encoder	
Header	-	
Declaration	static int32_t biss_req_send(uint32_t arg_num, char_t *p_arg[]);	
Description	Processes requests to the BiSS-C encoder. See "4.11.5(2) Flowchart of biss_req_send" for details.	
Arguments	arg_num	Number of request information
	*p_arg[]	Request information
Return value	cmd_index	Command number

(4) bissc_get_pos

bissc_get_pos	
Synopsis	Function to get position information from BiSS-C encoder
Header	-
Declaration	static void bissc_get_pos(void);
Description	Obtains position information from the BiSS-C encoder. See "4.11.5(3) Flowchart of bissc_get_pos" for details.
Arguments	None
Return value	None

(5) bissc_position

bissc_position		
Synopsis	Function to acquire and display position information from BiSS-C encoder	
Header	-	
Declaration	static void bissc_position(char_t *p_arg[], const uint32_t arg_num);	
Description	Displays position information obtained from the BiSS-C encoder. The bissc_get_pos() function is called internally to acquire position information. See "4.11.5(4) Flowchart of bissc_position" for details.	
Arguments	*p_arg[]	Request information (Not used.)
	arg_num	Number of request information
Return value	None	

(6) bissc_cmd_read

bissc_cmd_read		
Synopsis	Function to read specified number of bytes of data from BiSS-C encoder registers	
Header	-	
Declaration	static void bissc_cmd_read(char_t *p_arg[], const uint32_t arg_num);	
Description	Reads specified number of bytes of data from BiSS-C encoder registers. See "4.11.5(5) Flowchart of bissc_cmd_read" for details.	
Arguments	*p_arg[]	Request information
	arg_num	Number of request information
Return value	None	

(7) bissc_cmd_write

bissc_cmd_write		
Synopsis	Function to write specified number of bytes of data to BiSS-C encoder registers	
Header	-	
Declaration	static void bissc_cmd_write(char_t *p_arg[], const uint32_t arg_num);	
Description	Writes specified number of bytes of data to BiSS-C encoder registers. See "4.11.5(6) Flowchart of bissc_cmd_write" for details.	
Arguments	*p_arg[]	Request information
	arg_num	Number of Request information
Return value	None	

(8) bissc_trans_timer

bissc_trans_timer	
Synopsis	Function to set up position information acquisition synchronized with ELC events on BiSS-C encoder interface
Header	-
Declaration	static void bissc_trans_timer(void);
Description	Set the BiSS-C encoder interface to acquire position information synchronously with the ELC event input trigger. See "4.11.5(7) Flowchart of bissc_trans_timer" for details.
Arguments	None
Return value	None

(9) bissc_cmd_elctimer

bissc_cmd_elctimer		
Synopsis	Function to read position information from BiSS-C encoder synchronously with ELC events	
Header	-	
Declaration	static void bissc_cmd_elctimer(char_t *p_arg[], const uint32_t arg_num);	
Description	Periodically reads position information from the BiSS-C encoder synchronously with the ELC event input trigger. bissc_trans_timer() function is called internally to set the BiSS-C encoder interface. See "4.11.5(8) Flowchart of bissc_cmd_elctimer" for details.	
Arguments	*p_arg[]	Request information
	arg_num	Number of Request information
Return value	None	

(10) bissc_cmd_elcstop

bissc_cmd_elcstop		
Synopsis	Function to terminate position information readout synchronized with ELC events	
Header	-	
Declaration	static void bissc_cmd_elcstop(char_t *p_arg[], const uint32_t arg_num);	
Description	This function terminates the reading of position information from the BiSS-C encoder synchronized with the ELC event input trigger, and displays the acquired position information. See "4.11.5(9) Flowchart of bissc_cmd_elcstop" for details.	
Arguments	*p_arg[]	Request information (Not used.)
	arg_num	Number of Request information
Return value	None	

(11) bissc_exit

bissc_exit		
Synopsis	BiSS-C Encoder Program Exit Function	
Header	-	
Declaration	static void bissc_exit(char_t *p_arg[], const uint32_t arg_num);	
Description	Exit the BiSS-C encoder program. See "4.11.5(10) Flowchart of bissc_exit" for details.	
Arguments	*p_arg[]	Request information (Not used.)
	arg_num	Number of Request information
Return value	None	

(12) reg_acc_req

reg_acc_req		
Synopsis	BiSS-C encoder register access function	
Header	-	
Declaration	static void reg_acc_req(r_biss_cmd_t cmd);	
Description	Performs register access for BiSS-C encoders. See "4.11.5(11) Flowchart of reg_acc_req" for details.	
Arguments	cmd	Command
Return value	None	

(13) callback_get_pos

callback_get_pos	
Synopsis	Callback function to notify position information reception from BiSS-C encoder
Header	-
Declaration	static void callback_get_pos(void *p_result);
Description	The result is stored in memory and the acquisition completion flag is set. See "4.11.5(12) Flowchart of callback_get_pos" for details.
Arguments	p_result : Pointer to the r_bissc_sensordt_t structure that contains the result of position information acquisition. See "4.5.1 callback_get_pos" for details.
Return value	None

(14) callback_reg_access

callback_reg_access	
Synopsis	Callback function to notify the result of register access to BiSS-C encoder
Header	-
Declaration	static void callback_reg_access(void *p_result);
Description	The result is stored in memory and the access completion flag is set. See "4.11.5(13) Flowchart of callback_reg_access" for details.
Arguments	p_result : Pointer to the r_bissc_regdata_t structure that contains the register access result. See "4.5.2 callback_reg_access" for details.
Return value	None

(15) callback_elctimer_result

callback_elctimer_result	
Synopsis	Callback function to notify the result of receiving position information from BiSS-C encoder by ELC event synchronization.
Header	-
Declaration	static void callback_elctimer_result(void *p_result);
Description	The result of receiving position information is stored in the biss_ti_result variable. See "4.11.5(14) Flowchart of callback_elctimer_result" for details.
Arguments	p_result : Pointer to the r_bissc_sensordt_t structure that contains the result of position information acquisition. See "4.5.3 callback_elctimer_result" for details.
Return value	None

4.11.4 List of Variables in the Sample Program

Table 4.9 lists the major static variables. Table 4.10 lists the major static constant variables.

Table 4.9 Major Static Variables

Type	Variables	Contents	Function used
bool	bissc_sensor_flg	Position information acquisition completion flag Initial value: false	bissc_get_pos, bissc_trans_timer, reg_acc_req, callback_get_pos
bool	bissc_register_flg	Register access completion flag Initial value: false	biss_req_send, reg_acc_req, callback_reg_access
r_bissc_sensreq_t	bissc_sensreq	Request information for acquiring position information	bissc_get_pos, bissc_trans_timer
r_bissc_regreq_t	bissc_regreq	Request information for register access	biss_req_send, bissc_cmd_read, bissc_cmd_write, reg_acc_req
r_bissc_sensordt_t*	p_bissc_result_sns	Position data acquisition results	bissc_position, callback_get_pos, callback_elctimer_result
r_bissc_regdata_t*	p_bissc_result_reg	Register access result	biss_req_send, callback_reg_access
bool	bissc_cmd_flg	Command processing flag Initial value: false	enc_main, biss_req_send, reg_acc_req
bool	bissc_elc_flg	ELC timer mode operating flag Initial value: false	biss_req_send, bissc_cmd_elctimer, bissc_cmd_elcstop
uint8_t	reg_data[64]	Buffer for register read/write	biss_req_send, bissc_cmd_read, bissc_cmd_write
r_bissc_sensordt_t	biss_ti_result[10]	ELC timer mode ring buffer for position information storage	bissc_cmd_elcstop, callback_elctimer_result
uint32_t	biss_ti_count	Ring buffer storage position counter	bissc_cmd_elctimer, bissc_cmd_elcstop, callback_elctimer_result
uint32_t	biss_ti_valid	Number of ring buffer valid data	bissc_cmd_elctimer, bissc_cmd_elcstop, callback_elctimer_result
bool	biss_ti_full	Ring buffer full flag	bissc_cmd_elctimer, bissc_cmd_elcstop, callback_elctimer_result

Table 4.10 Major Static Constant Variables

Type	Variables	Contents	Function used
char_t *[]	p_cmd_tbl	Command string check table Number of elements: CMD_NUM Contents: { “pos”, “read”, “write”, “elctimer”, “elcstop”, “exit” }	biss_req_send
cmd_func_t []	cmd_func_tbl	Command function address table Number of elements: CMD_NUM Contents: { &bissc_position, &bissc_cmd_read, &bissc_cmd_write, &bissc_cmd_elctimer, &bissc_cmd_elcstop, &bissc_exit }	biss_req_send

4.11.5 Flowchart of Major Processes

Flowchart of the major functions are described below.

(1) Flowchart of enc_main

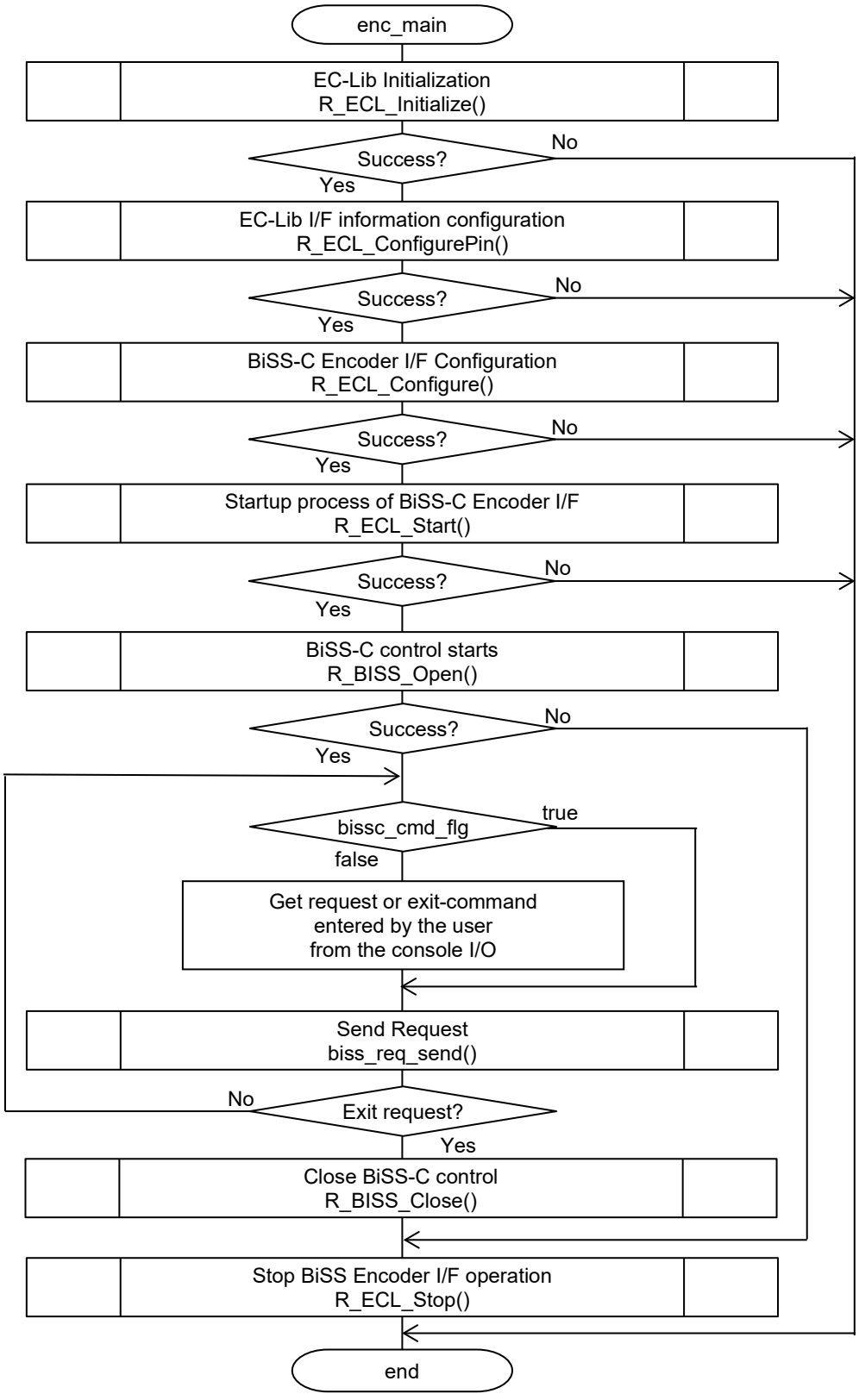
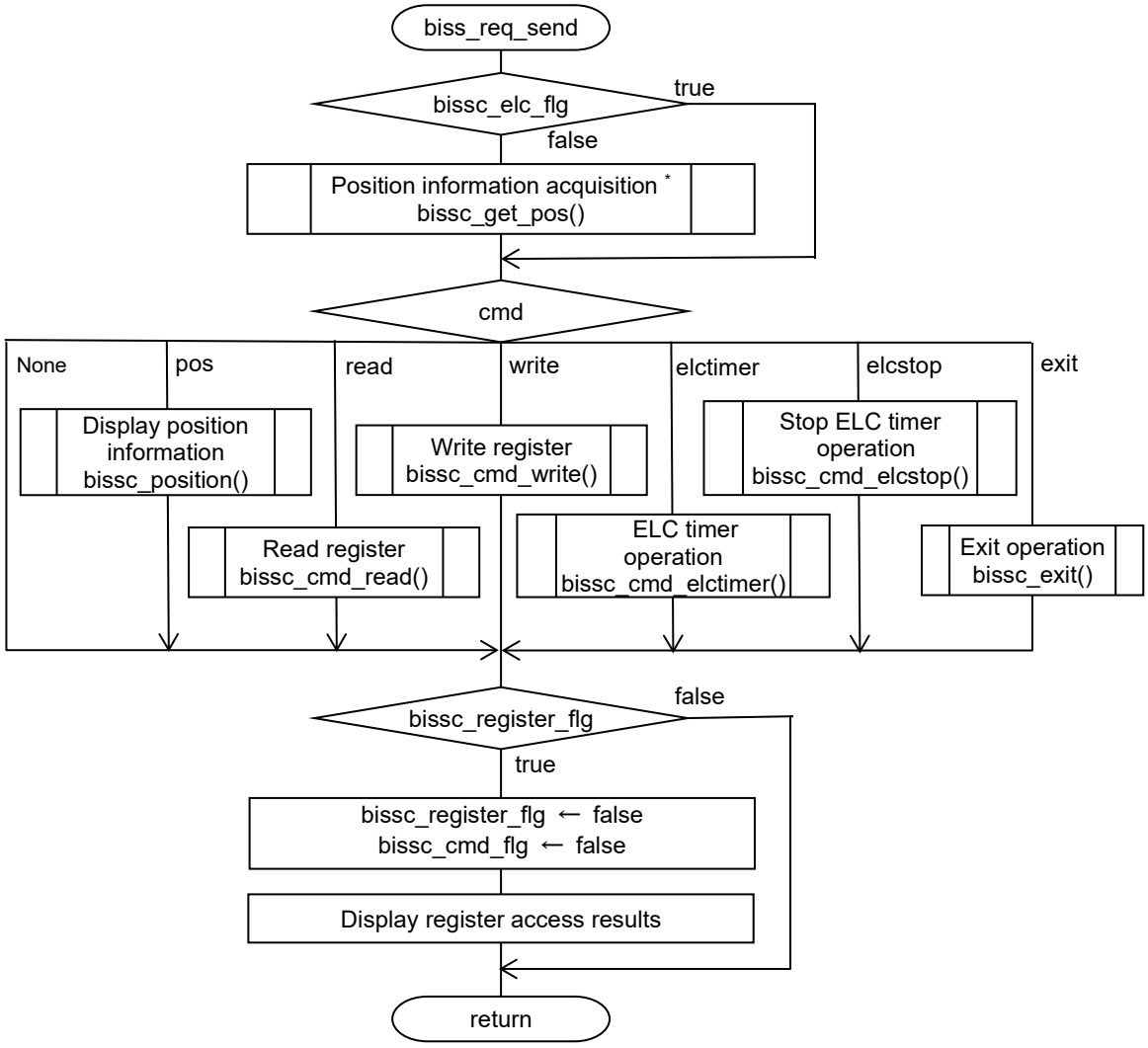


Figure 4.3 Flowchart of enc_main

(2) Flowchart of biss_req_send



Note When executing a continuous register write, depending on the encoder's EEPROM access time, a wait time may be required before acquiring position information.
For the EEPROM access time, check the specifications of the used encoder.

Figure 4.4 Flowchart of biss_req_send

(3) Flowchart of bissc_get_pos

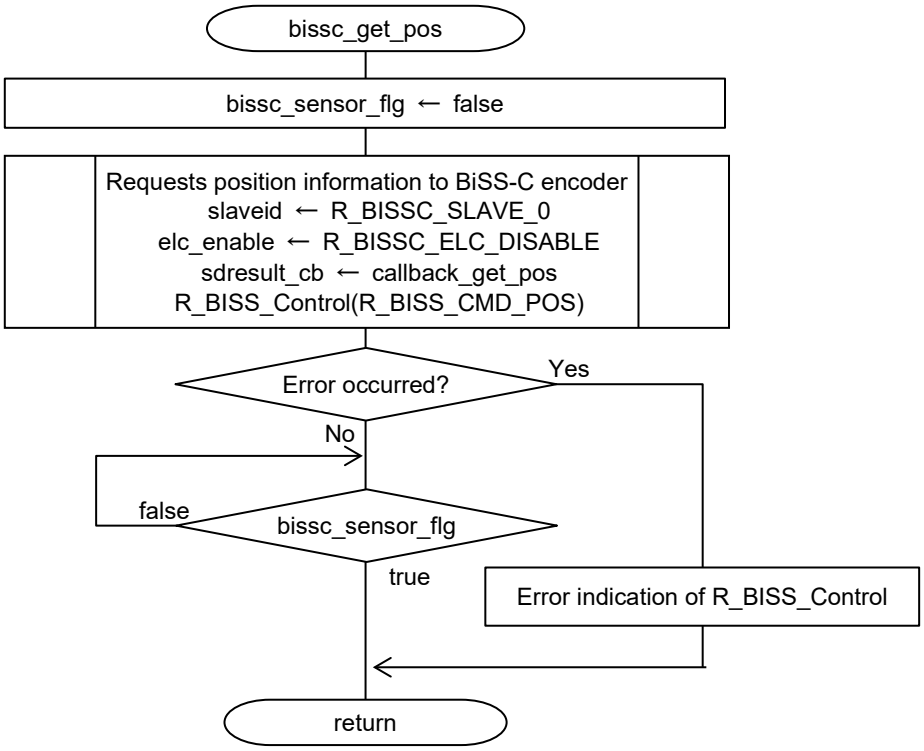


Figure 4.5 Flowchart of bissc_get_pos

(4) Flowchart of bissc_position

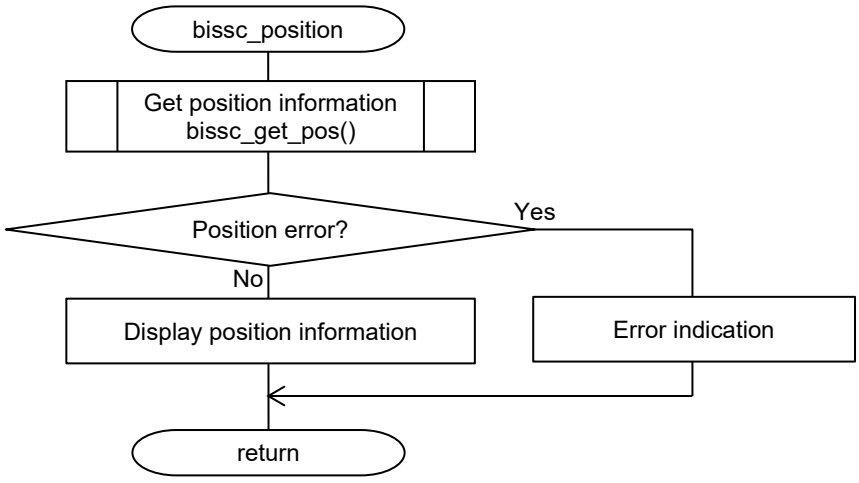


Figure 4.6 Flowchart of bissc_position

(5) Flowchart of bissc_cmd_read

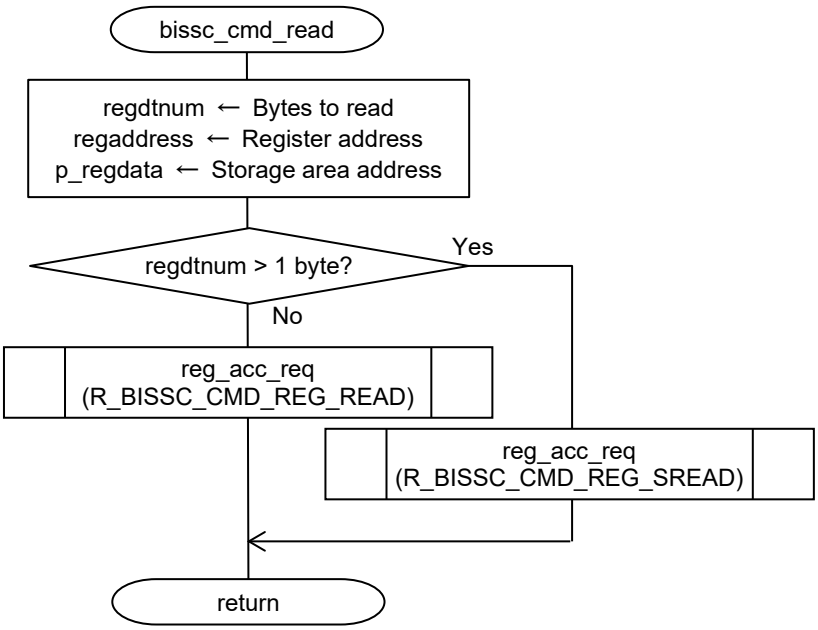


Figure 4.7 Flowchart of bissc_cmd_read

(6) Flowchart of bissc_cmd_write

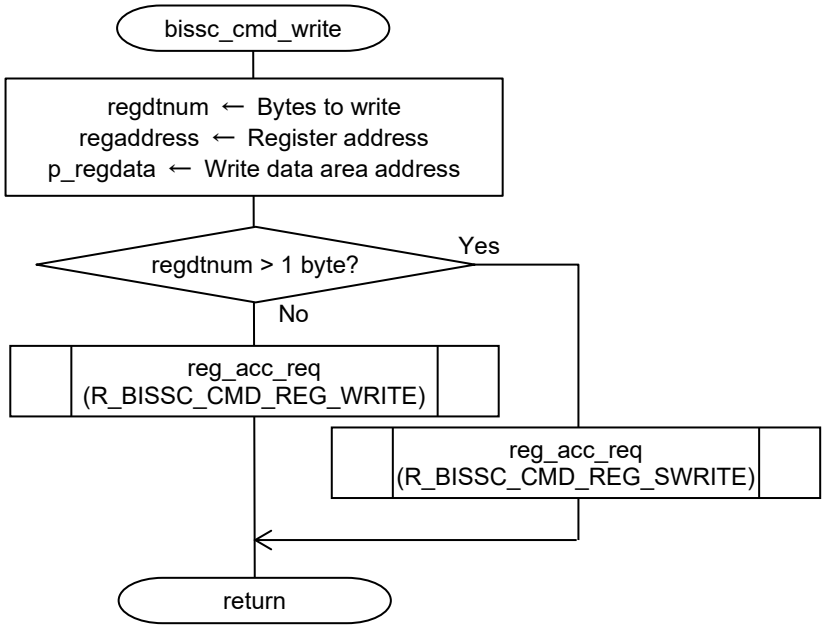


Figure 4.8 Flowchart of bissc_cmd_write

(7) Flowchart of bissc_trans_timer

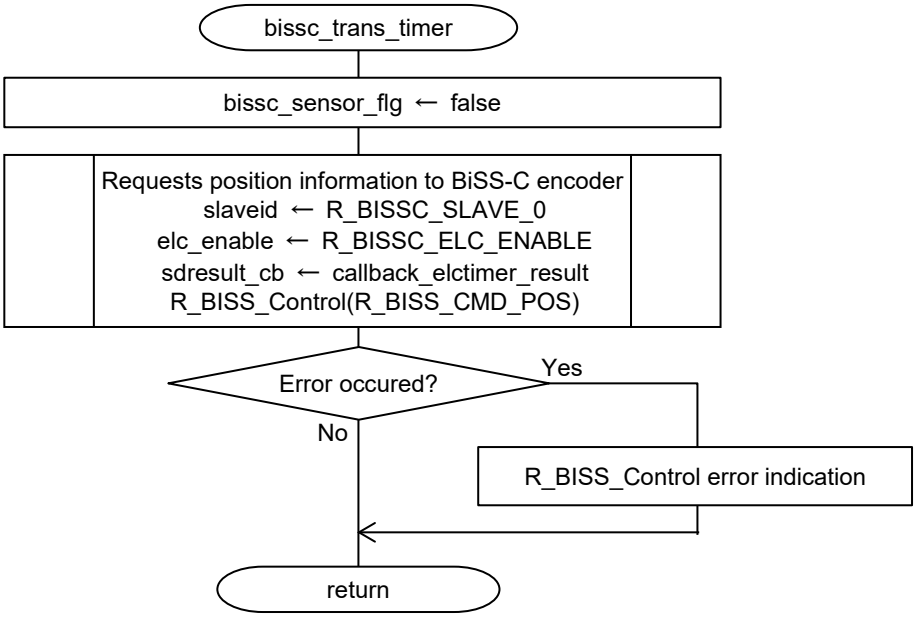


Figure 4.9 Flowchart of bissc_trans_timer

(8) Flowchart of bissc_cmd_elctimer

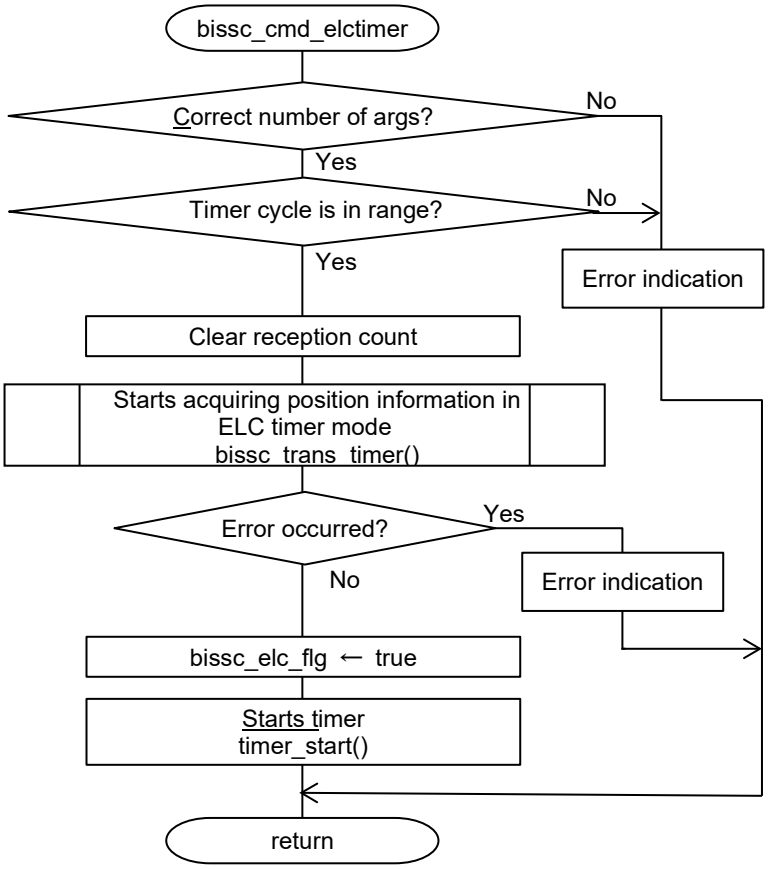


Figure 4.10 Flowchart of bissc_cmd_elctimer

(9) Flowchart of bissc_cmd_elcstop

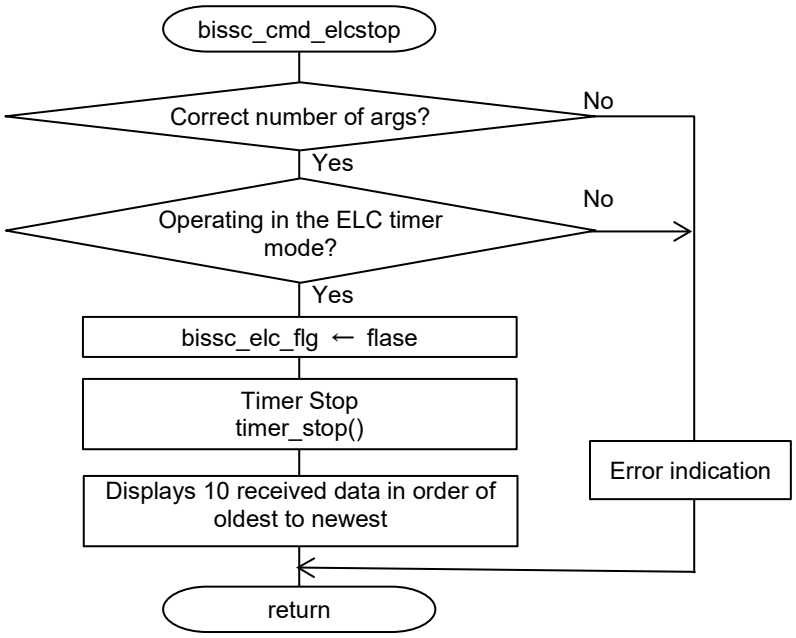


Figure 4.11 Flowchart of bissc_cmd_elcstop

(10) Flowchart of bissc_exit

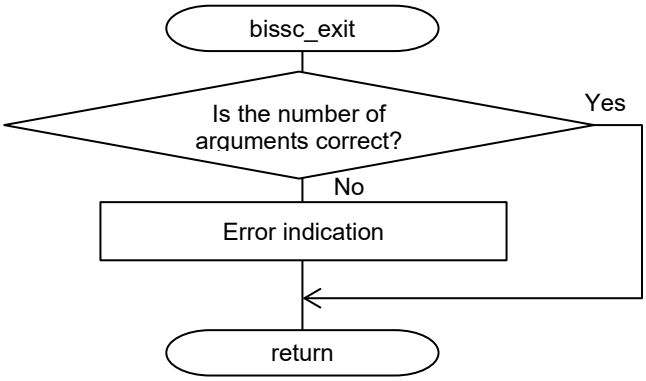


Figure 4.12 Flowchart of bissc_exit

(11) Flowchart of reg_acc_req

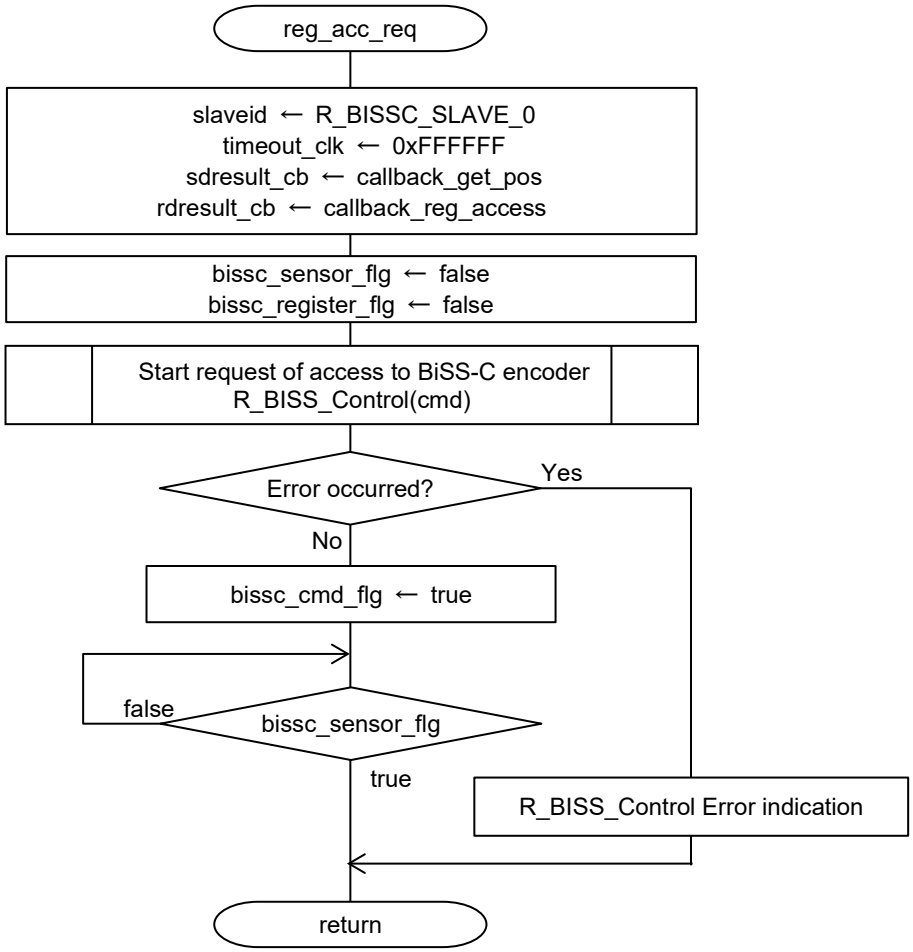


Figure 4.13 Flowchart of reg_acc_req

(12) Flowchart of callback_get_pos

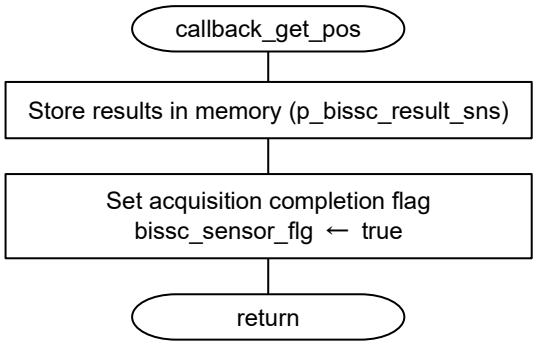


Figure 4.14 Flowchart of callback_get_pos

(13) Flowchart of callback_reg_access

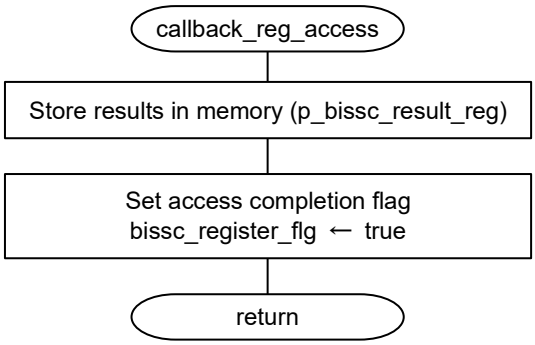


Figure 4.15 Flowchart of callback_reg_access

(14) Flowchart of callback_elctimer_result

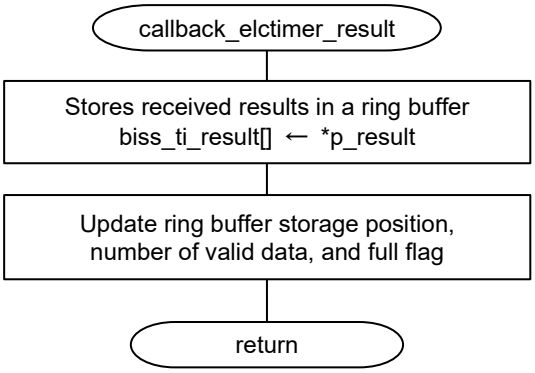


Figure 4.16 Flowchart of callback_elctimer_result

4.11.6 Operation Sequence

(1) Start Sequence

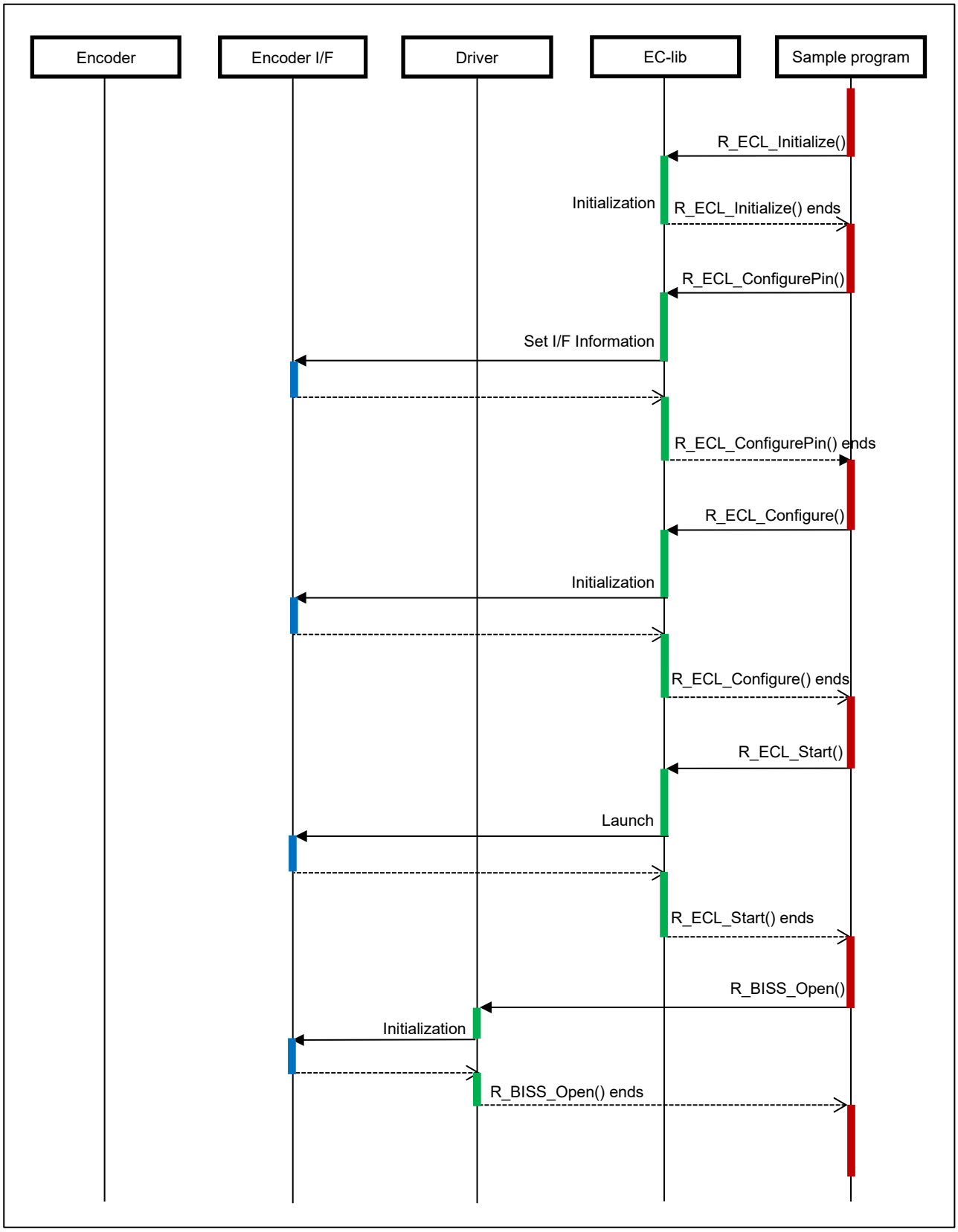


Figure 4.17 Start Sequence Diagram

(2) Sensor Mode (Acquisition of Position Information) Sequence

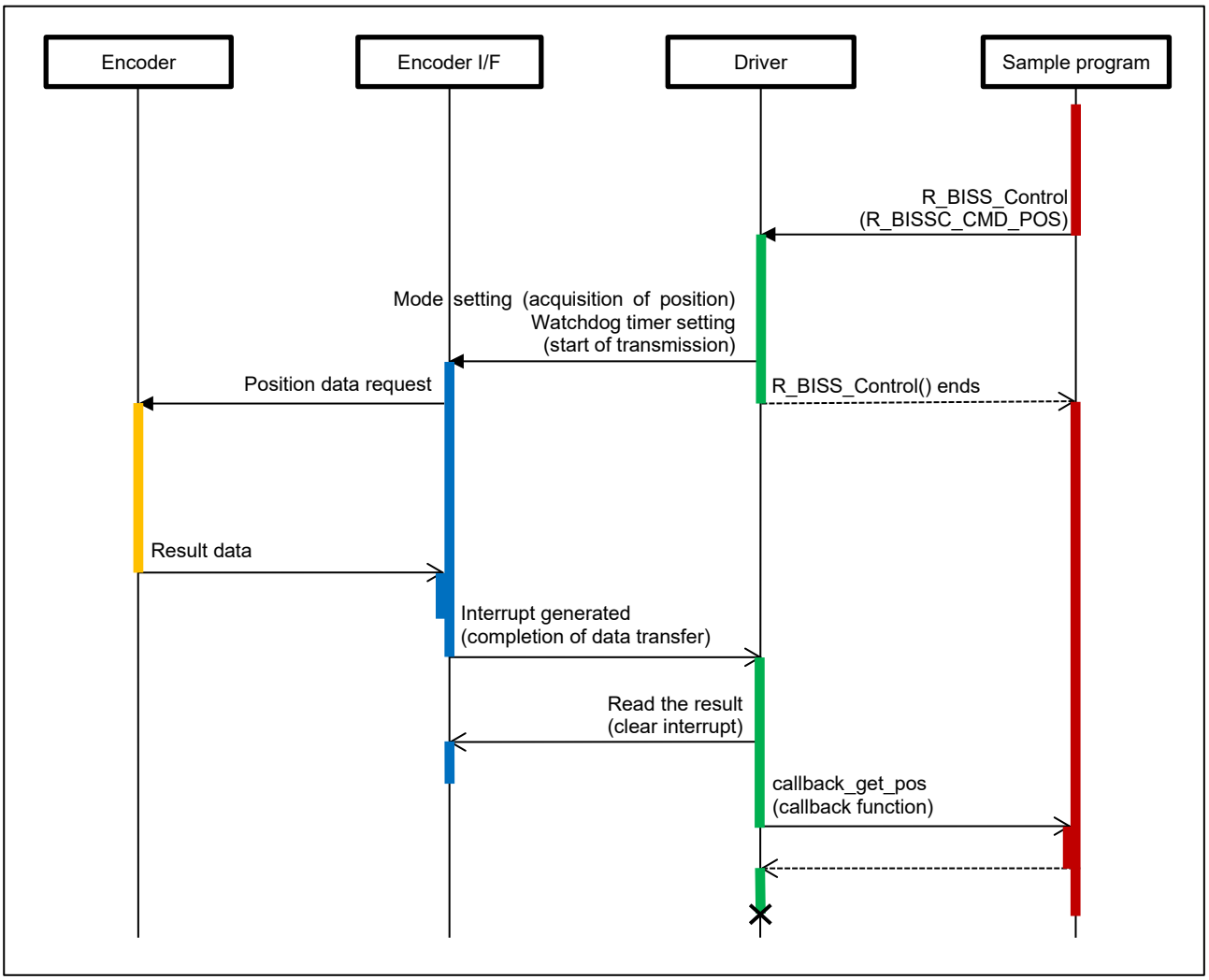


Figure 4.18 Sensor Mode (Acquisition of Position Information) Sequence Diagram

(3) Register Access (Read / Write) Sequence

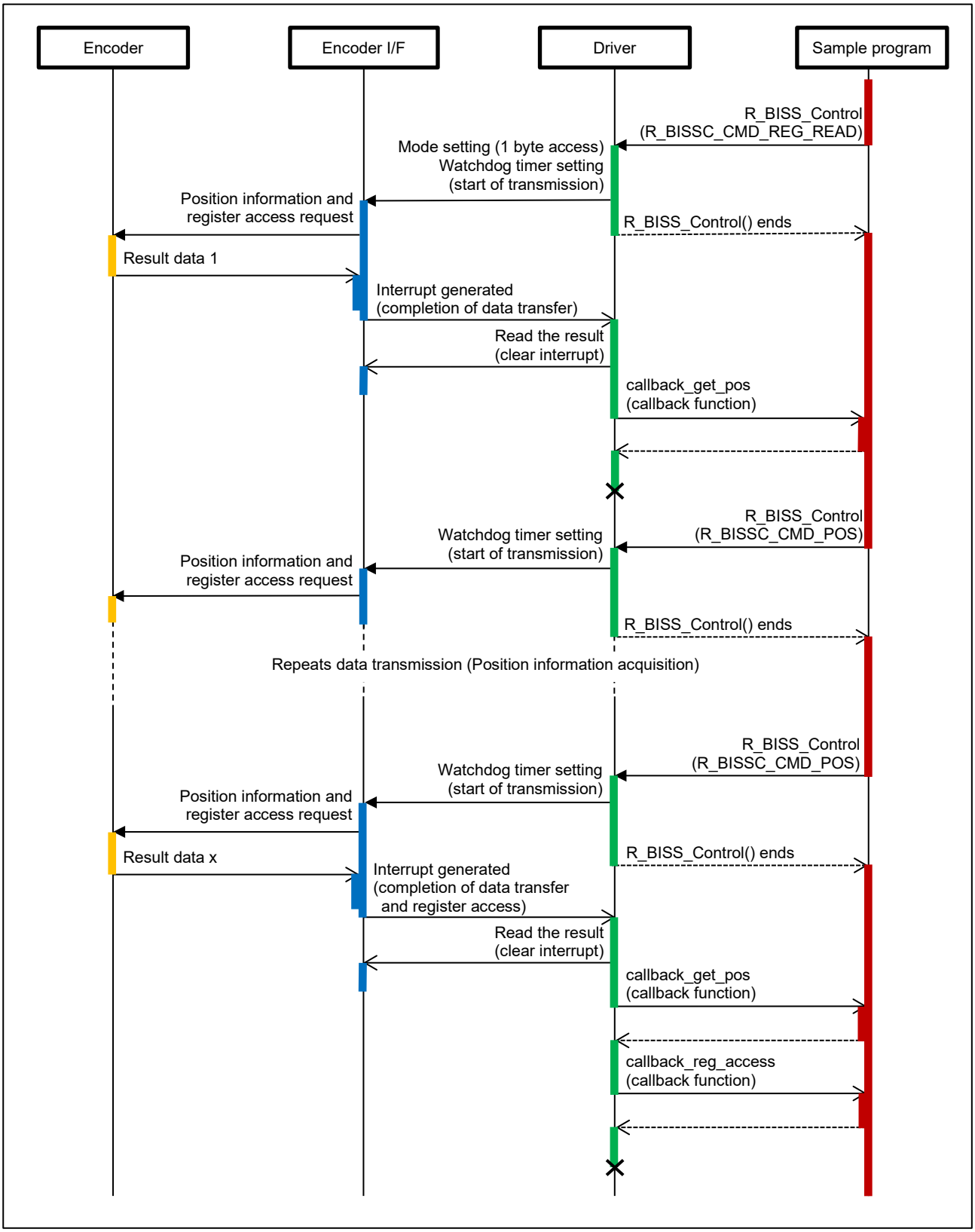


Figure 4.19 Register Access (Read / Write) Sequence Diagram

(4) Continuous register access (Read / Write) Sequence

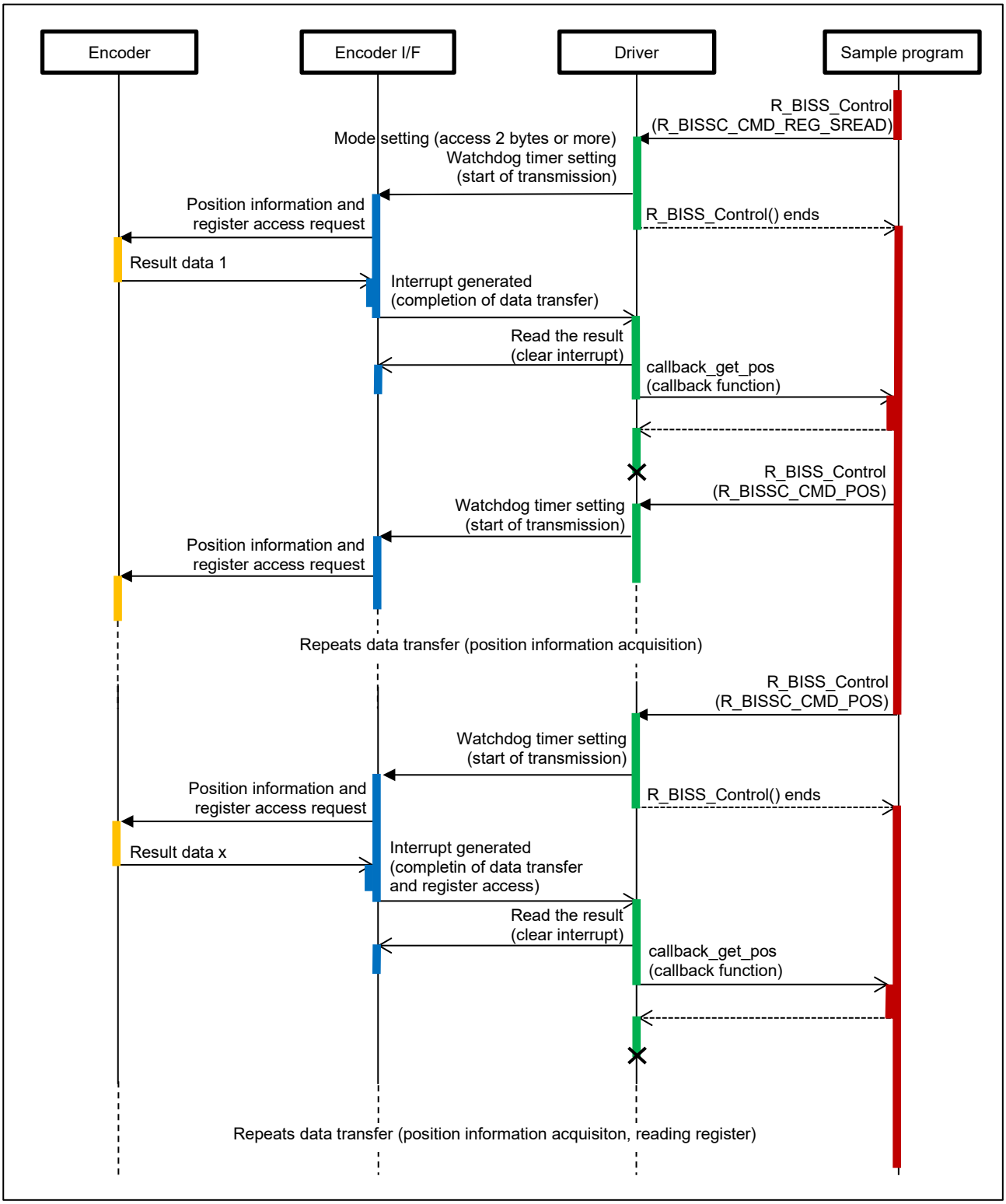


Figure 4.20 Continuous Register Access (Read / Write) Sequence Diagram (1/2)

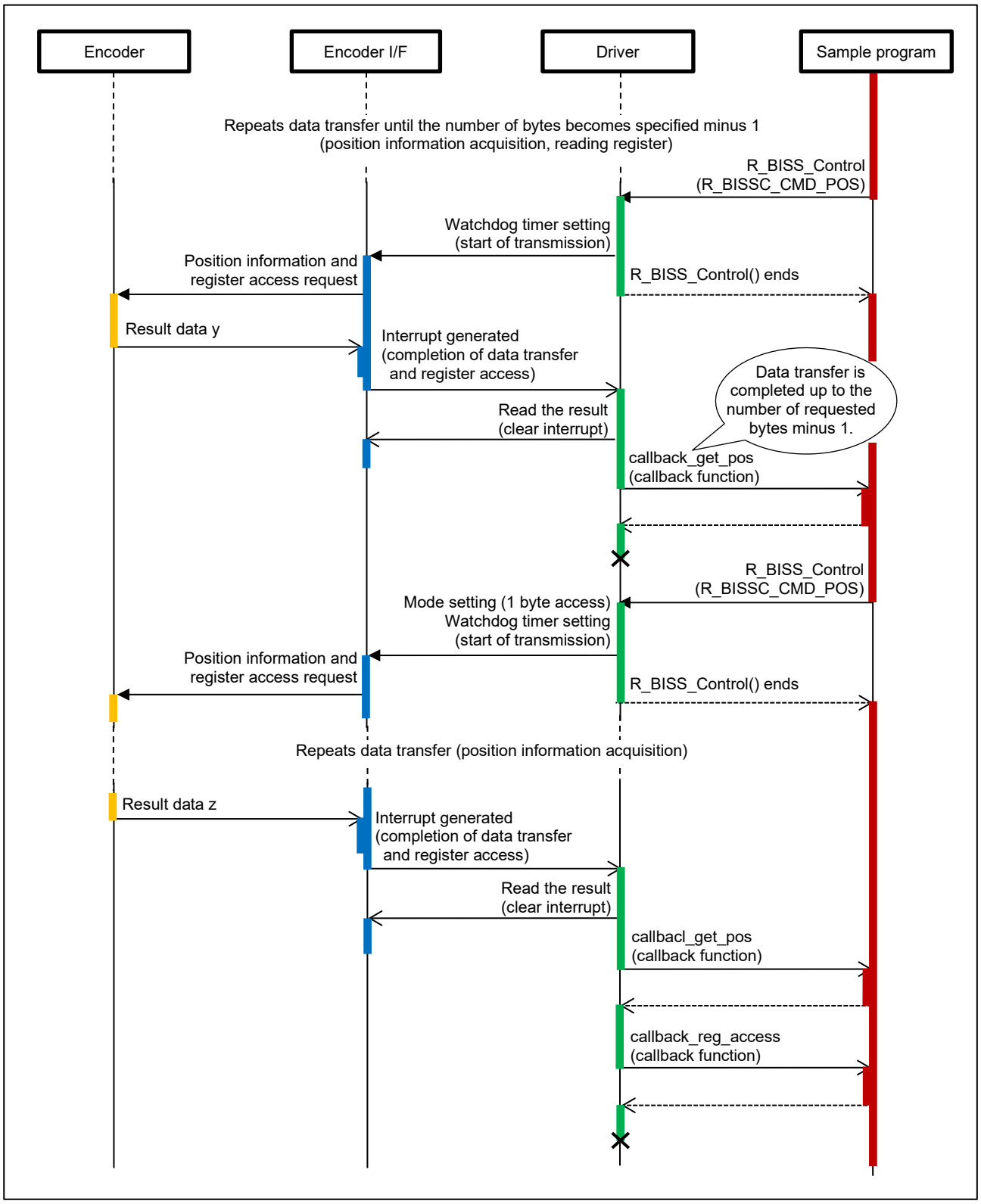


Figure 4.21 Continuous Register Access (Read / Write) Sequence Diagram (2/2)

(5) Continuous Register Access (Read / Write) Sequence (Error on the way)

Figure 4 22 shows a sequence diagram when an error occurs during continuous register access.

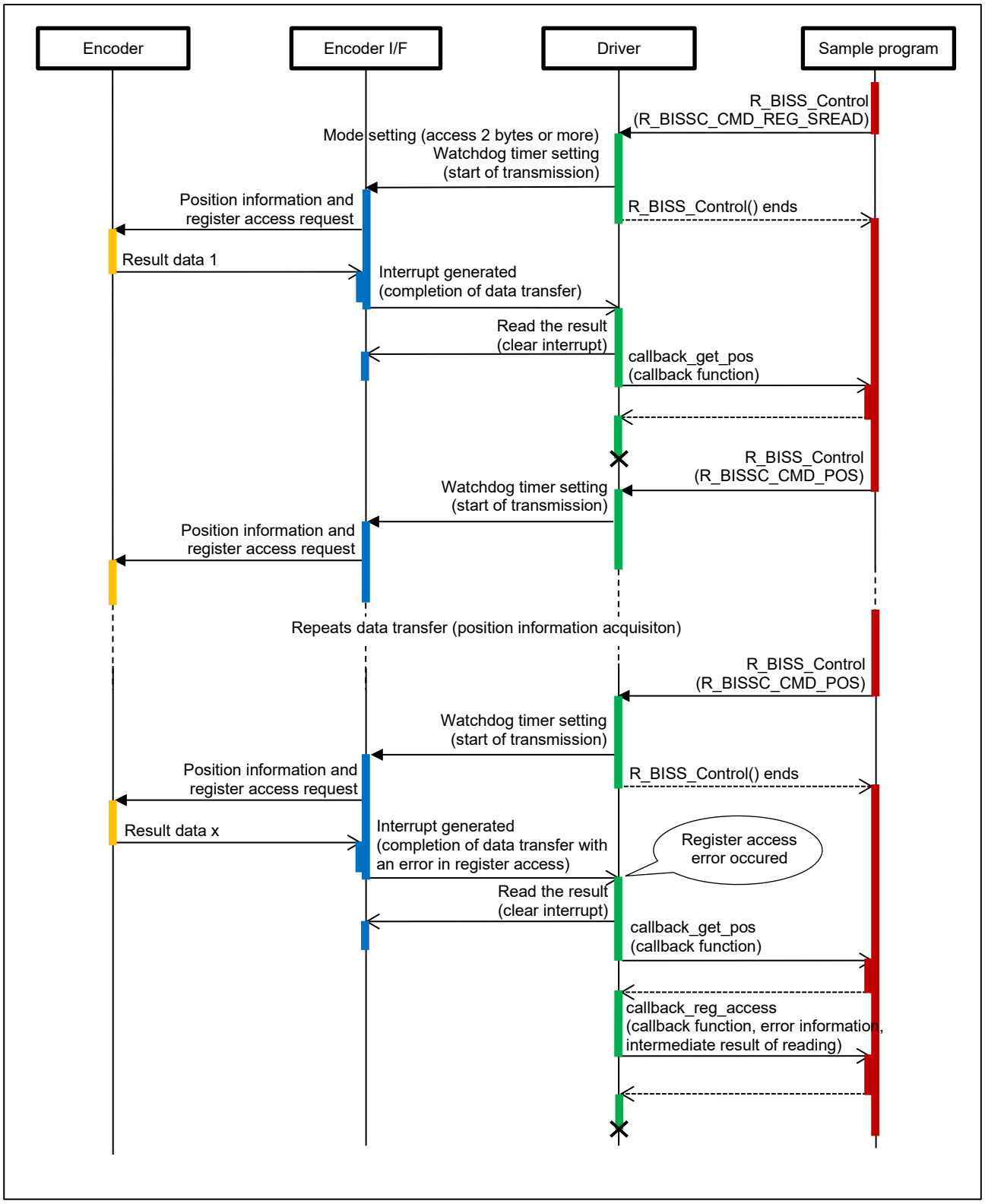


Figure 4.22 Continuous Register Access (Rread / Write) Sequence Diagram (Error on the way)

(6) Continuous Register Access (Read / Write) Stop Sequence

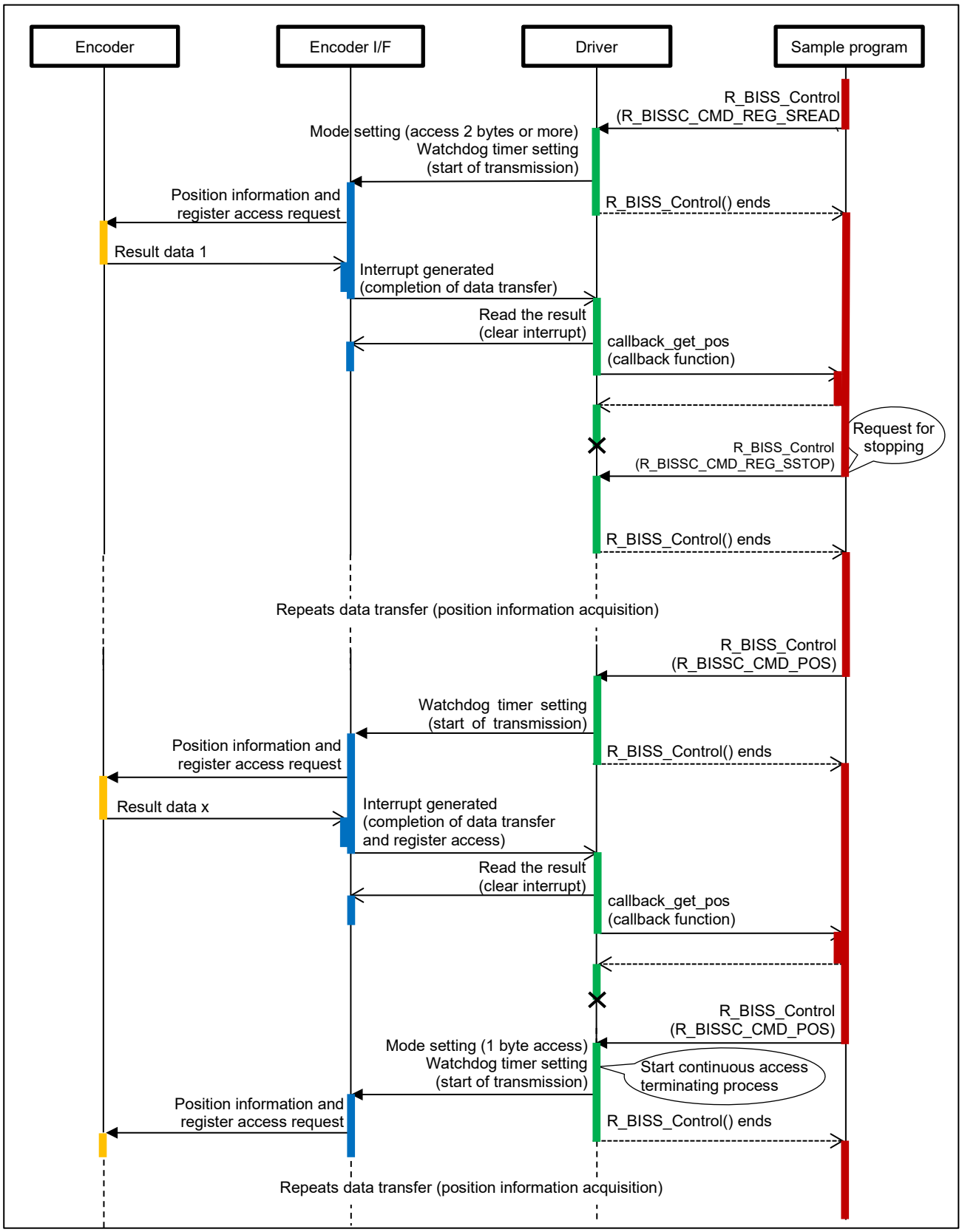


Figure 4.23 Continuous Register Access (Read / Write) Stop Sequence Diagram (1/2)

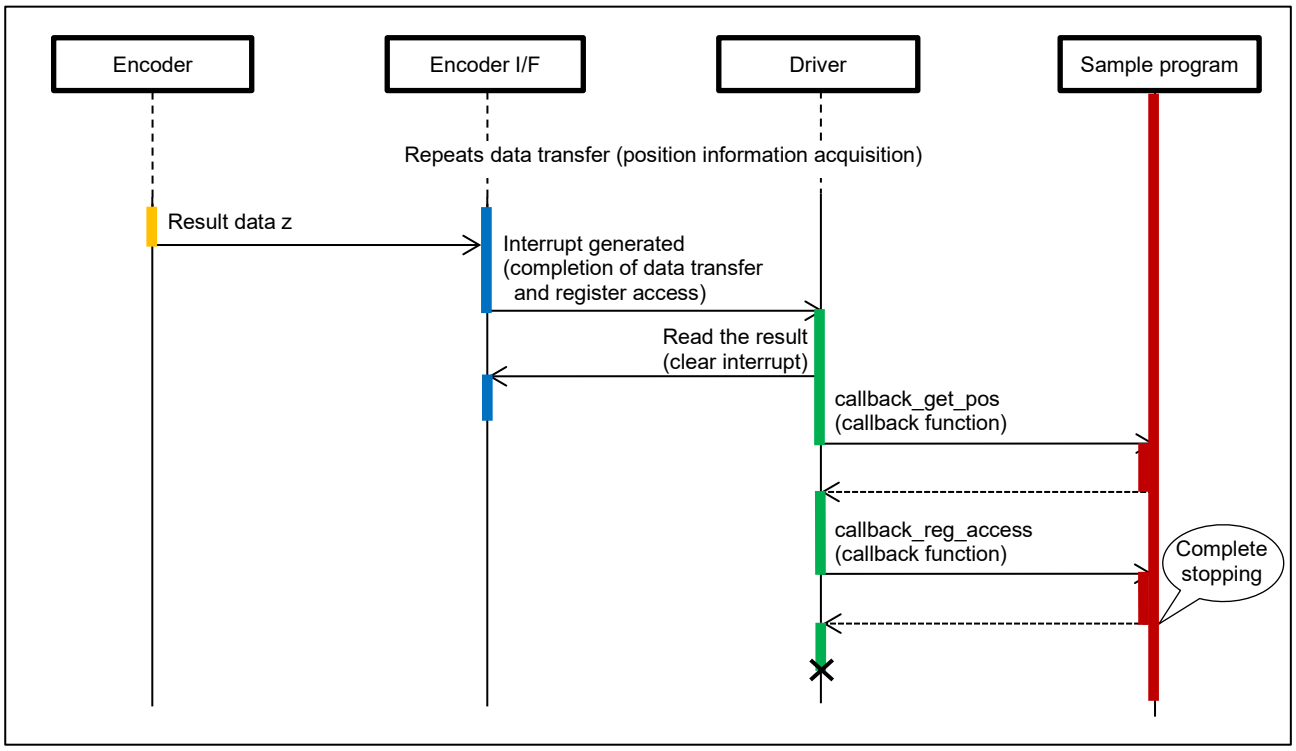


Figure 4.24 Continuous Register Access (Read / Write) Stop Sequence Diagram (2/2)

(7) ELC Timer Mode Acquisition of Position Information Sequence

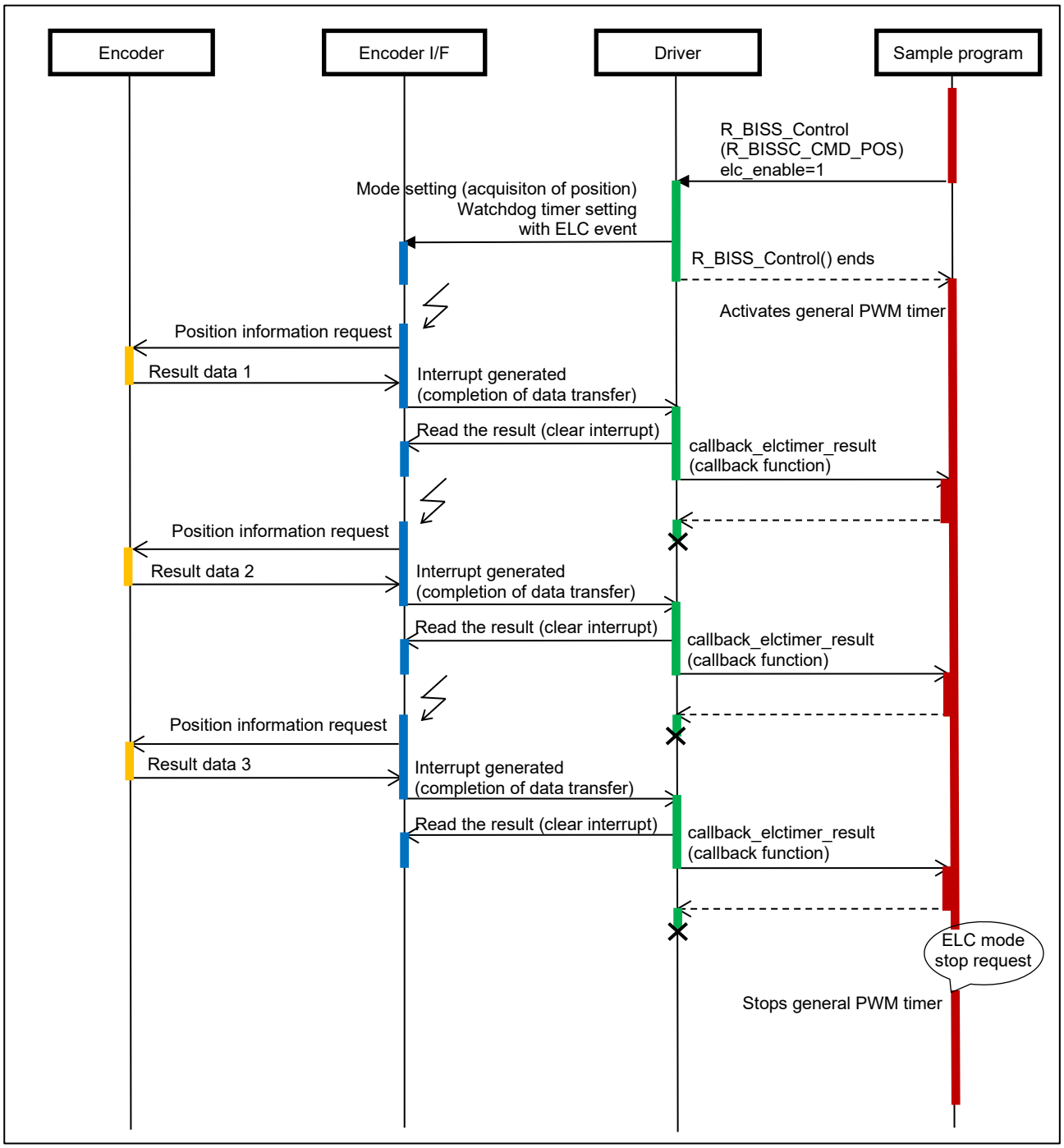


Figure 4.25 ELC Timer Mode Acquisition of Position Information Sequence Diagram

(8) Stop Sequence

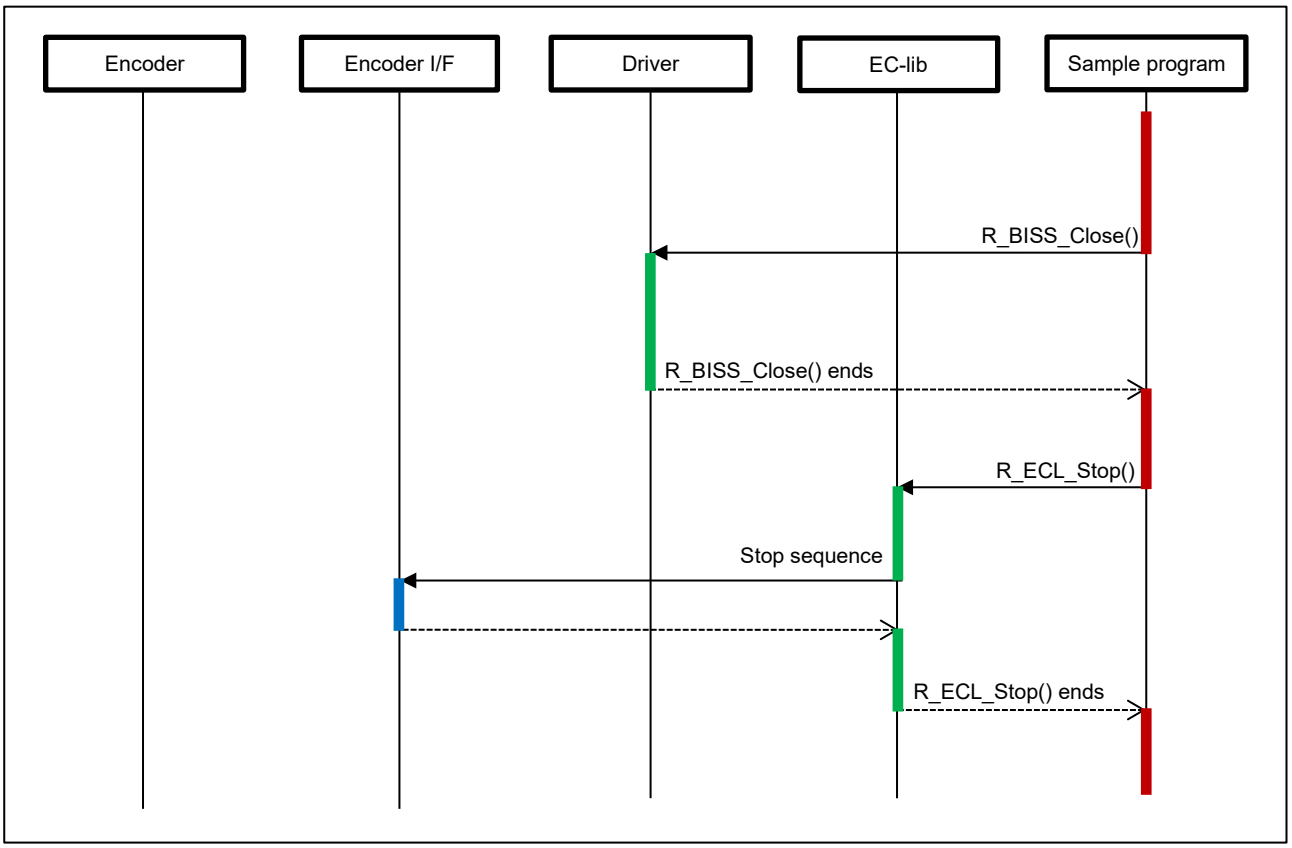


Figure 4.26 Stop Sequence Diagram

4.11.7 Console Commands

This sample program supports the absolute encoder AD36/1219AF.0CBEB from Hengstler. The commands that can be input from the console are as follows.

Table 4.1 Console Command List

Command	Description
pos	Get position information once
read *AA BB*	Reads data from register address “*AA*” for “*BB*” bytes. *
write *AA BB CCCCCCCC*	Write "*CCCCCCCC*" data from register address "*AA*" for "*BB*" bytes.*
elctimer *VAL*	The position information is continuously acquired in a timer cycle as an ELC event input trigger operation. The timer cycle *VAL* is specified in micro seconds by decimal number (maximum :6990 us). To stop continuous acquisition, enter the "elcstop" command.
elcstop	Stops continuous acquisition of position information.
exit	Exit Program

Note “*AA*", "*BB*", and "*CCCCCCCC*" are recognized as hexadecimal numbers.
Example: If *BB* = "24", 0x24 (36 bytes) are requested.

(1) Result of running

After running, it will display the command prompt. Please enter commands after ‘biss>’ appears.

```
BiSS sample program start
biss>
```

(2) Example of command execution

This is an example of executing the 'pos' command. Based on the response from the encoder, positional data and error information are displayed.

```
biss>pos
get position
  result
    multi turn data:      2807
    single turn data:     24917
    alarm err:           0
    warning err:         0
biss>
```

5. Sample code

The sample code can be downloaded from the Renesas Electronics website.

Revision History

Rev.	Date	Description	
		Page	Summary
Rev.1.00	Jun.28.22	-	First edition
Rev.2.00	Jun 20.24	4 17	Update description of the board name. Remove description about location of integer type definition.
Rev.3.00	Oct 10.25	1, 3, 4 4 7 to 12 22 49	Change description for trademarks. Correct board name of the operation environment. Change description of headers for functions. Revise description of operation outline and software structure. Correct description for elctimer command. Add example of command execution.
Rev.4.00	Feb 27.26	13 - 14 16 7 - 38	Revised the header column of “4.5 Specifications of User-defined Function”. Revised BiSS-C I/F transmit/receive interrupt priority for channel 0 and 1 to 11. Change the prefix of pointer variables to “p_”. (ex. pinfo -> p_info)

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/.