RZ/T2M Group

HIPERFACE DSL sample program

Introduction

This application note explains a sample program for acquiring and indicating information from an encoder in conformance with the HIPERFACE DSL® communications protocol specification by using the Encoder I/F Configuration Library (“EC-Lib”) of the RZ/T2M.

The features of the program:

- Acquiring angle information, etc. from an encoder (EFM50-0KF0A023A) compliant with the HIPERFACE DSL® communications protocol specification

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
HIPERFACE DSL controller (HFDSL)	Handling transfer to and from an absolute encoder incorporating a facility for handling the HIPERFACE DSL communications protocol
Interrupt controller (ICU)	Controlling interrupts from the HFDSL controller
General PWM timer (GPT) channel 0	Generating event cycles for input to the ELC
Event link controller (ELC)	Makes the link between events output from channel 0 of the GPT and the HFDSL 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.

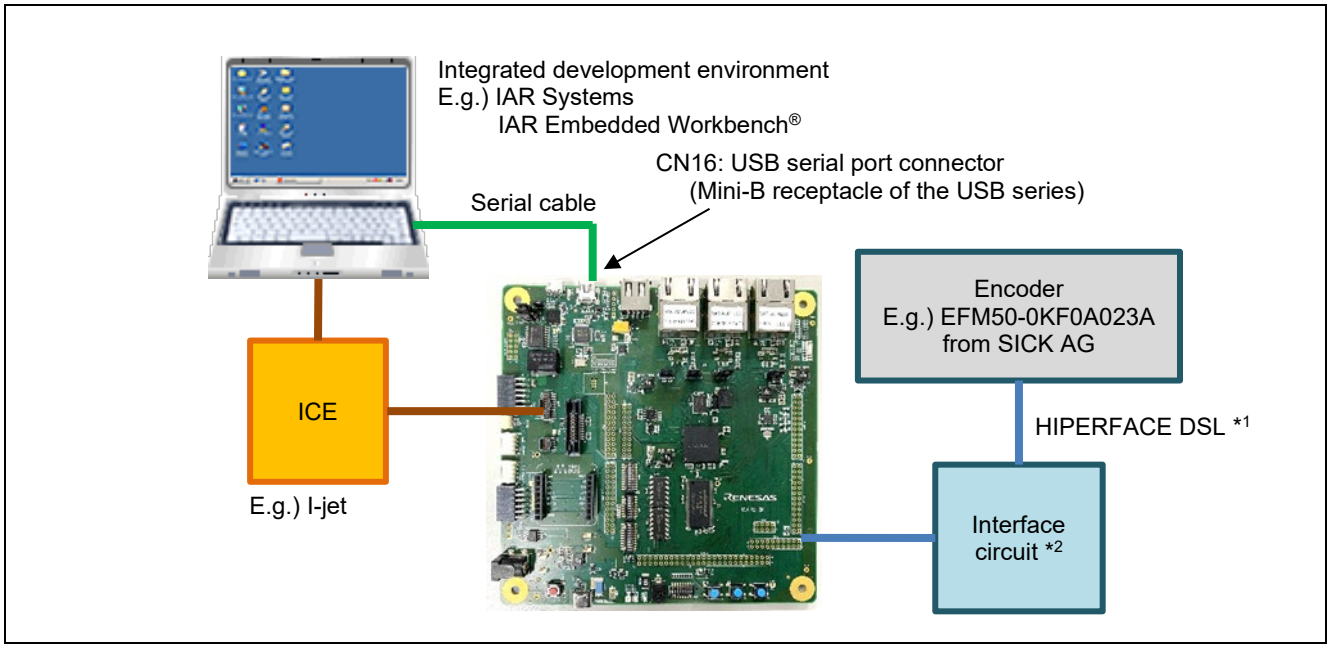

Figure 1.1 Operating Environment

Note: 1. For allowable cable length, refer to the encoder manuals.
2. Refer to the “HIPERFACE DSL® MASTER Integration Manual”.

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 = 800 MHz
Operating voltage	1.1 V (Core) / 1.8 V (PLL, etc.) / 3.3 V (I/O)
Integrated development environment *1	IAR Systems: IAR Embedded Workbench® for Arm® RENESAS: e² studio
Board	RSK+RZT2M (RTK9RZT2M0C00000BE)
Devices (function to be used on the board)	None

Note: 1. Refer to the RZ/T2M Group Encoder I/F HIPERFACE DSL sample program Release Note to check the version number of the 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
HFDSL0	dsl_in0 (ENCIF0)	P01_6	Input	Data input pin
	dsl_out0 (ENCIF2)	P02_0	Output	Data output pin
	dsl_en0 (ENCIF3)	P02_2	Output	Drive/receive control pin
HFDSL1	dsl_in1 (ENCIF5)	P17_3	Input	Data input pin
	dsl_out1 (ENCIF7)	P17_5	Output	Data output pin
	dsl_en1 (ENCIF8)	P03_0	Output	Drive/receive control pin

4. Software

4.1 HFDSL Driver Function

The functions of the HFDSL driver are listed below.

1. Initial settings
2. Acquiring positional data
3. Transmitting and receiving messages

4.2 File Structure

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

4.3 Functions

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

Table 4.1 Functions

Category	Function Name	Page Number
HFDSL driver API functions	R_HFDSL_Open	8
	R_HFDSL_Close	8
	R_HFDSL_GetVersion	9
	R_HFDSL_CheckInitSeq	9
	R_HFDSL_Control	9
User-defined functions	hfdsl_int_nml_callback	16
	hfdsl_int_err_callback	16
	hfdsl_int_raw_callback	17
	hfdsl_int_mrcv_callback	17
	hfdsl_int_init_callback	17
Interrupt handlers	hfdsl_int_nml_isr_ch0	18
	hfdsl_int_nml_isr_ch1	18
	hfdsl_int_err_isr_ch0	18
	hfdsl_int_err_isr_ch1	18
	hfdsl_int_tovr_isr_ch0	19
	hfdsl_int_tovr_isr_ch1	19

4.4 Specifications of API Functions

4.4.1 R_HFDSL_Open

R_HFDSL_Open	
Synopsis	Starts controlling operation of the encoder.
Header	r_hfdsl_rzt2_if.h
Declaration	int32_t R_HFDSL_Open(const int32_t id, r_hfdsl_info_t* p_info);
Description	Call this function before using the HFDSL driver. It initializes the driver. ・Setting the interrupts ・Setting the callback functions
Argument	id : Specifies the ID to be used. (It is defined in r_hfdsl_rzt2_dat.h.) R_HFDSL0_ID : Specifies channel 0. R_HFDSL1_ID : Specifies channel 1. Others : Setting is not allowed. p_info : Holder for the initial settings of the driver Set the pointer to the r_hfdsl_info_t structure which holds the information on the initial settings of the driver.
Return value	R_HFDSL_SUCCESS: Normal termination R_HFDSL_ERR_INVALID_ARG: Abnormal termination (The id or the r_hfdsl_info_t structure member pointed by the p_info is not specified.) R_HFDSL_ERR_ACCESS: Abnormal termination (This function is already executed.)
Note	Before calling this function, be sure to use the EC-Lib to configure and start the multi-protocol encoder interface. Calling this API function from within a callback function is prohibited.

4.4.2 R_HFDSL_Close

R_HFDSL_Close	
Synopsis	Ending control of the encoder
Header	r_hfdsl_rzt2_if.h
Declaration	int32_t R_HFDSL_Close(const int32_t id);
Description	This function stops controlling operation of the encoder on the designated channel.
Argument	id : Specifies the ID to be used. (It is defined in r_hfdsl_rzt2_dat.h.) R_HFDSL0_ID : Specifies Channel 0. R_HFDSL1_ID : Specifies Channel 1. Others : Setting is not allowed.
Return Value	R_HFDSL_SUCCESS: Normal termination R_HFDSL_ERR_INVALID_ARG: Abnormal termination (The id is not specified.)
Note	Before calling this function, be sure to call R_HFDSL_Open. Calling this API function from within a callback function is prohibited.

4.4.3 R_HFDSL_GetVersion

R_HFDSL_GetVersion	
Synopsis	Acquire the version number of the encoder interface driver.
Header	r_hfdsl_rzt2_if.h
Declaration	uint32_t R_HFDSL_GetVersion(void);
Description	This function acquires the version number of the HFDSL driver.
Argument	None
Return value	The 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. Ex.) For ver.1.2, the value returned is 0x00010002.
Note	Before calling this function, be sure to use the EC-Lib to configure and start the multi-protocol encoder interface.

4.4.4 R_HFDSL_CheckInitSeq

R_HFDSL_CheckInitSeq			
Synopsis	Acquire status of the encoder initialization sequence.		
Header	r_hfdsl_rzt2_if.h		
Declaration	uint32_t R_HFDSL_CheckInitSeq(const int32_t id);		
Description	This function acquires status of the encoder initialization sequence.		
Argument	id	: Specifies the ID to be used. (It is defined in r_hfdsl_rzt2_dat.h.)	
		R_HFDSL0_ID	: Specifies channel 0.
		R_HFDSL1_ID	: Specifies channel 1.
		Others	: Setting is not allowed.
Return value	0 to 3: Shows INIT[1:0] bit value of the EVENT register. R_HFDSL_ERR_INVALID_ARG： Abnormal termination (The id is invalid.)		

4.4.5 R_HFDSL_Control

R_HFDSL_Control			
Synopsis	Controlling operation of the encoder.		
Header	r_hfdsl_rzt2_if.h		
Declaration	int32_t R_HFDSL_Control(const int32_t id, const r_hfdsl_cmd_t cmd, void *const p_buf);		
Description	This function controls operations of the encoder by using the cmd argument. See section “4.4.5(1), Protocol Initialization Commands” and section “4.4.5(2), Control Commands” for the operation.		
Argument	id	: Specifies the ID to be used. (It is defined in r_hfdsl_rzt2_dat.h.)	
		R_HFDSL0_ID	: Specifies channel 0.
		R_HFDSL1_ID	: Specifies channel 1.
		Others	: Setting is not allowed.
	cmd	: Command For details, see section “4.4.5(1), Protocol Initialization Commands” and section “4.4.5(2), Control Commands”.	
	p_buf	: Arguments corresponding to each cmd.	
Return value	R_HFDSL_SUCCESS: Normal termination R_HFDSL_ERR_INVALID_ARG: Abnormal termination (The id or cmd is not specified.) See section “4.4.5(1), Protocol Initialization Commands” and section “4.4.5(2), Control Commands” for other returned values.		

(1) Protocol Initialization Commands

(a) R_HFDSL_CMD_INIT1

R_HFDSL_CMD_INIT1	
Synopsis	Protocol initialization1
Header	r_hfdsl_rzt2_if.h
Declaration	int32_t R_HFDSL_Control(const int32_t id, const r_hfdsl_cmd_t cmd, void *const p_buf);
Description	Call this function after executing function R_HFDSL_Open or after a protocol reset. Call the protocol initialization commands R_HFDSL_CMD_INIT2 to R_HFDSL_CMD_INIT6 after calling this function. For how to detect a protocol reset, see section “4.5.2 hfdsl_int_err_callback”.
Argument	id : Specifies the ID to be used. (It is defined in r_hfdsl_rzt2_dat.h.) R_HFDSL0_ID : Specifies channel 0. R_HFDSL1_ID : Specifies channel 1. Others : Setting is not allowed. cmd : Specify R_HFDSL_CMD_INIT1. p_buf : Specify NULL.
Return value	R_HFDSL_SUCCESS: Normal termination R_HFDSL_ERR_INVALID_ARG: Abnormal termination (The id or p_buf is invalid.) R_HFDSL_ERR_ACCESS: Abnormal termination (R_HFDSL_Open has not been executed.)
Note	Calling this API function from within a callback function is prohibited.

(b) R_HFDSL_CMD_INIT2

R_HFDSL_CMD_INIT2	
Synopsis	Protocol initialization 2
Header	r_hfdsl_rzt2_if.h
Declaration	int32_t R_HFDSL_Control(const int32_t id, const r_hfdsl_cmd_t cmd, void *const p_buf);
Description	Execute the protocol initialization command R_HFDSL_CMD_INIT1, and then execute this function after the INIT bits in the EVENT register have been set to 1. Execute the protocol initialization command R_HFDSL_CMD_INIT3 following a call of this function.
Argument	id : Specifies the ID to be used. (It is defined in r_hfdsl_rzt2_dat.h.) R_HFDSL0_ID : Specifies channel 0. R_HFDSL1_ID : Specifies channel 1. Others : Setting is not allowed. cmd : Specify R_HFDSL_CMD_INIT2. p_buf : Specify NULL.
Return value	R_HFDSL_SUCCESS: Normal termination R_HFDSL_ERR_INVALID_ARG: Abnormal termination (The id or p_buf is invalid.) R_HFDSL_ERR_ACCESS: Abnormal termination (R_HFDSL_CMD_INIT1 has not been executed.)
Note	Calling this API function from within a callback function is prohibited.

(c) R_HFDSL_CMD_INIT3

R_HFDSL_CMD_INIT3	
Synopsis	Protocol initialization 3
Header	r_hfdsl_rzt2_if.h
Declaration	int32_t R_HFDSL_Control(const int32_t id, const r_hfdsl_cmd_t cmd, void *const p_buf);
Description	Execute the protocol initialization command R_HFDSL_CMD_INIT2, and then execute this function. Execute the protocol initialization command R_HFDSL_CMD_INIT4 following a call of this function. The settings for the individual registers are the constants listed in “Table 4.3 User-defined Constants for the HFDSL Driver (r_hfdsl_rzt2_config.h)”. When changing operation of the HFDSL controller, change these constants.
Argument	id : Specifies the ID to be used. (It is defined in r_hfdsl_rzt2_dat.h.) R_HFDSL0_ID : Specifies channel 0. R_HFDSL1_ID : Specifies channel 1. Others : Setting is not allowed. cmd : Specify R_HFDSL_CMD_INIT3. p_buf : Specify NULL.
Return value	R_HFDSL_SUCCESS: Normal termination R_HFDSL_ERR_INVALID_ARG: Abnormal termination (Ther id or p_buf is invalid.) R_HFDSL_ERR_ACCESS: Abnormal termination (R_HFDSL_CMD_INIT2 has not been executed.)
Note	Calling this API function from within a callback function is prohibited.

(d) R_HFDSL_CMD_INIT4

R_HFDSL_CMD_INIT4	
Synopsis	Protocol initialization 4
Header	r_hfdsl_rzt2_if.h
Declaration	int32_t R_HFDSL_Control(const int32_t id, const r_hfdsl_cmd_t cmd, void *const p_buf);
Description	Execute the protocol initialization command R_HFDSL_CMD_INIT3, and then execute this function, after waiting for 1 ms following the INIT bits in the EVENT register having been set to 2. Execute the protocol initialization command R_HFDSL_CMD_INIT5 following a call of this function. If the value returned is R_HFDSL_ERR_INIT and the protocol is to be initialized again, start over again from the protocol initialization command R_HFDSL_CMD_INIT1 after executing the control command R_HFDSL_CMD_RST.
Argument	id : Specifies the ID to be used. (It is defined in r_hfdsl_rzt2_dat.h.) R_HFDSL0_ID : Specifies channel 0. R_HFDSL1_ID : Specifies channel 1. Others : Setting is not allowed. cmd : Specify R_HFDSL_CMD_INIT4. p_buf : Specify NULL.
Return value	R_HFDSL_SUCCESS: Normal termination R_HFDSL_ERR_INVALID_ARG: Abnormal termination (The id or p_buf is invalid.) R_HFDSL_ERR_ACCESS: Abnormal termination (R_HFDSL_CMD_INIT3 has not been executed.) R_HFDSL_ERR_INIT: Abnormal termination (EDGES register value is NG.)
Note	Calling this API function from within a callback function is prohibited.

(e) R_HFDSL_CMD_INIT5

R_HFDSL_CMD_INIT5	
Synopsis	protocol initialization 5
Header	r_hfdsl_rzt2_if.h
Declaration	int32_t R_HFDSL_Control(const int32_t id, const r_hfdsl_cmd_t cmd, void *const p_buf);
Description	Call this function after waiting for at least 24 µs following a call of the R_HFDSL_CMD_INIT4 protocol initialization command. After calling this function, execute the protocol initialization command R_HFDSL_CMD_INIT6. If the value returned is R_HFDSL_ERR_INIT and the protocol is to be initialized again, start over again from the protocol initialization command R_HFDSL_CMD_INIT1 after executing the control command R_HFDSL_CMD_RST. The allowable upper limit on delay time due to the length of the cable is defined in R_HFDSL_DELAY_UPP_LIMIT of “Table 4.3, User-defined Constants for the HFDSL Driver (r_hfdsl_rzt2_config.h)”.
Argument	id : Specifies the ID to be used. (It is defined in r_hfdsl_rzt2_dat.h.) R_HFDSL0_ID : Specifies channel 0. R_HFDSL1_ID : Specifies channel 1. Others : Setting is not allowed. cmd : Specify R_HFDSL_CMD_INIT5. p_buf : Specify NULL.
Return value	R_HFDSL_SUCCESS: Normal termination R_HFDSL_ERR_INVALID_ARG: Abnormal termination (The id or p_buf is invalid) R_HFDSL_ERR_ACCESS: Abnormal termination (R_HFDSL_CMD_INIT4 has not been executed.) R_HFDSL_ERR_INIT: Abnormal termination (Transfer may not stable due to the cable delay.)
Note	Calling this API function from within a callback function is prohibited.

(f) R_HFDSL_CMD_INIT6

R_HFDSL_CMD_INIT6	
Synopsis	Protocol initialization 6
Header	r_hfdsl_rzt2_if.h
Declaration	int32_t R_HFDSL_Control(const int32_t id, const r_hfdsl_cmd_t cmd, void *const p_buf);
Description	Call this function after the INIT bits in the EVENT register have been set to 3 following a call of the R_HFDSL_CMD_INIT5 protocol initialization command. The positional value, etc. are acquired after normal return from this function and the INIT_END bit in the EVENT register has been set to 1. If the value returned is R_HFDSL_ERR_INIT and the protocol is to be initialized again, start over again from the protocol initialization command R_HFDSL_CMD_INIT1 after executing the control command R_HFDSL_CMD_RST.
Argument	id : Specifies the ID to be used. (It is defined in r_hfdsl_rzt2_dat.h.) R_HFDSL0_ID : Specifies channel 0. R_HFDSL1_ID : Specifies channel 1. Others : Setting is not allowed. cmd : Specify R_HFDSL_CMD_INIT6. p_buf : Encoder ID Specify the unit32_t pointer carriable which holds the encoder ID.
Return value	R_HFDSL_SUCCESS: Normal termination R_HFDSL_ERR_INVALID_ARG: Abnormal termination (The id is invalid or p_buf is null.) R_HFDSL_ERR_ACCESS: Abnormal termination (R_HFDSL_CMD_INIT5 has not been executed.) R_HFDSL_ERR_INIT: Abnormal termination (The connected encoder ID does not match the specified ID.)
Note	Calling this API function form within a callback function is prohibited.

(2) Control Commands

(a) R_HFDSL_CMD_POS

R_HFDSL_CMD_POS	
Synopsis	Acquiring the fast position
Header	r_hfdsl_rzt2_if.h
Declaration	int32_t R_HFDSL_Control(const int32_t id, const r_hfdsl_cmd_t cmd, void *const p_buf);
Description	This function acquires the fast position by reading the fast position register H (POS_H) and fast position register (POS).
Argument	id : Specifies the ID to be used. (It is defined in r_hfdsl_rzt2_dat.h.) R_HFDSL0_ID : Specifies channel 0. R_HFDSL1_ID : Specifies channel 1. Others : Setting is not allowed. cmd : Specify R_HFDSL_CMD_POS. p_buf : Fast position Specifies the pointer to the r_hfdsl_pos_t structure which holds the fast position value. For details, see section “4.10.1(2) r_hfdsl_pos_t”.
Return value	R_HFDSL_SUCCESS: Normal termination R_HFDSL_ERR_INVALID_ARG: Abnormal termination (The id is invalid or p_buf is null.)

(b) R_HFDSL_CMD_VPOS

R_HFDSL_CMD_VPOS	
Synopsis	Acquiring the safe position
Header	r_hfdsl_rzt2_if.h
Declaration	int32_t R_HFDSL_Control(const int32_t id, const r_hfdsl_cmd_t cmd, void *const p_buf);
Description	This function acquires the safe position by reading the safe position register H (VPOS_H), safe position register (VPOS), and safe position CRC register (VPOSCRC).
Argument	id : Specifies the ID to be used. (It is defined in r_hfdsl_rzt2_dat.h.) R_HFDSL0_ID : Specifies channel 0. R_HFDSL1_ID : Specifies channel 1. Others : Setting is not allowed. cmd : Specify R_HFDSL_CMD_VPOS. p_buf : Safe position Specifies the pointer to the r_hfdsl_vpos_t structure which holds the safe position value. For details, see section “4.10.1(3) r_hfdsl_vpos_t”.
Return value	R_HFDSL_SUCCESS: Normal termination R_HFDSL_ERR_INVALID_ARG: Abnormal termination (The id is invalid or p_buf is null.)

(c) R_HFDSL_CMD_VEL

R_HFDSL_CMD_VEL	
Synopsis	Acquiring the rotational velocity of the motor.
Header	r_hfdsl_rzt2_if.h
Declaration	int32_t R_HFDSL_Control(const int32_t id, const r_hfdsl_cmd_t cmd, void *const p_buf);
Description	This function acquires the rotational velocity of the motor by reading the velocity register (VEL).
Argument	id : Specifies the ID to be used. (It is defined in r_hfdsl_rzt2_dat.h.) R_HFDSL0_ID : Specifies channel 0. R_HFDSL1_ID : Specifies channel 1. Others : Setting is not allowed. cmd : Specify R_HFDSL_CMD_VEL. p_buf : Rotational velocity of the motor Specifies the pointer to uint32_t which holds the rotational velocity of the motor.
Return value	R_HFDSL_SUCCESS: Normal termination R_HFDSL_ERR_INVALID_ARG: Abnormal termination (The id is invalid or p_buf is null.)

(d) R_HFDSL_CMD_MSG

R_HFDSL_CMD_MSG	
Synopsis	Transmitting messages
Header	r_hfdsl_rzt2_if.h
Declaration	int32_t R_HFDSL_Control(const int32_t id, const r_hfdsl_cmd_t cmd, void *const p_buf);
Description	This function transmits messages. The data received is indicated by function hfdsl_int_mrcv_callback. For details of the function, see section “4.5.4 hfdsl_int_mrcv_callback”.
Argument	id : Specifies the ID to be used. (It is defined in r_hfdsl_rzt2_dat.h.) R_HFDSL0_ID : Specifies channel 0. R_HFDSL1_ID : Specifies channel 1. Others : Setting is not allowed. cmd : Specify R_HFDSL_CMD_MSG. p_buf : Message data for transmission Specifies the pointer to the r_hfdsl_send_msg_t structure which holds message data for transmission. For details, see section “4.10.1(4) r_hfdsl_send_msg_t”.
Return value	R_HFDSL_SUCCESS: Normal termination R_HFDSL_ERR_INVALID_ARG: Abnormal termination (The id is invalid or p_buf is null.) R_HFDSL_ERR_ACCESS: Abnormal termination (The protocol initialization function described in section “4.4.5(1) Protocol Initialization Commands”, has not been executed.)
Note	Calling this API function from within a callback function is prohibited. To proceed with the next transmission, execute this function following the hfdsl_int_mrcv_callback function.

(e) R_HFDSL_CMD_RST

R_HFDSL_CMD_RST	
Synopsis	Protocol reset
Header	r_hfdsl_rzt2_if.h
Declaration	int32_t R_HFDSL_Control(const int32_t id, const r_hfdsl_cmd_t cmd, void *const p_buf);
Description	This function resets the protocol. After this function is called. An INT_err interrupt is generated in response to the PRST bit in the EVENT_ERR being set to 1. To resume communications, call the functions described in section “4.4.5(1), Protocol Initialization Commands”.
Argument	id : Specifies the ID to be used. (It is defined in r_hfdsl_rzt2_dat.h.) R_HFDSL0_ID : Specifies channel 0. R_HFDSL1_ID : Specifies channel 1. Others : Setting is not allowed. cmd : Specify R_HFDSL_CMD_RST. p_buf : Specify NULL.
Return value	R_HFDSL_SUCCESS: Normal termination R_HFDSL_ERR_INVALID_ARG: Abnormal termination (The id or p_buf is invalid.)

4.5 Specification of User-defined Functions

4.5.1 hfdsl_int_nml_callback

hfdsl_int_nml_callback	
Synopsis	Indicating the generation of an INT_nml interrupt
Header	r_hfdsl_rzt2_if.h
Declaration	void hfdsl_int_nml_callback(uint8_t event);
Description	This callback function is registered with the member variable p_cb_nml of the argument r_hfdsl_info_t structure of the R_HFDSL_Open function. It is called when an INT_nml interrupt is generated by the MIN, POS_RDY, or POS_UPD bit of the EVENT register. If the setting of the POS_RDY or POS_UPD bit in the EVENT register is 1, the fast position register H (POS_H) or the fast position register (POS) has been updated. In such cases, the fast position can be acquired by executing the function R_HFDSL_Control (R_HFDSL_CMD_POS) from within this function. This function is the context of the interrupt handler. To ensure interrupt responsiveness, return immediately. The function name is an example and can be set freely.
Argument	event : Source of the INT_nml interrupt Holds the value of the EVENT register. The value of this argument is only valid within this function.
Return value	None
Note	This function is not called when an INT_nml interrupt is generated in response to the MRCV, or INIT_END bit in the EVENT register being set to 1.

4.5.2 hfdsl_int_err_callback

hfdsl_int_err_callback	
Synopsis	Indicating the generation of an INT_err interrupt
Header	r_hfdsl_rzt2_if.h
Declaration	void hfdsl_int_err_callback(uint32_t event_err);
Description	This callback function is registered with the member variable p_cb_err of the argument r_hfdsl_info_t structure of the R_HFDSL_Open function. It is called when an INT_err interrupt is generated. This function is the context of the interrupt handler. To ensure interrupt responsiveness, return immediately. The function name is an example and can be set freely.
Argument	event_err : Source of the INT_err interrupt Holds the value of the EVENT_ERR register. The value of this argument is only valid within this function.
Return value	None

4.5.3 hfdsl_int_raw_callback

hfdsl_int_raw_callback		
Synopsis	Indicating the generation of an INT_tovr interrupt and notifying that a value is stored in the FIFO.	
Header	r_hfdsl_rzt2_if.h	
Declaration	void hfdsl_int_raw_callback (uint16_t* raw_data);	
Description	This callback function is registered with the member variable p_cb_raw of the argument r_hfdsl_info_t structure of the R_HFDSL_Open function. It is called when an INT_tovr interrupt is generated and notify that a value is stored in the FIFO. This function is the context of the interrupt handler. To ensure interrupt responsiveness, return immediately. The function name is an example and can be set freely.	
Argument	raw_data[]	: FIFO_DATA register value Holds the FIFO_DATA register value. The value of this argument remains valid until the next INT_tovr interrupt is generated.
Return value	None	

4.5.4 hfdsl_int_mrcv_callback

hfdsl_int_mrcv_callback		
Synopsis	Indicating the generation of an INT_tovr interrupt and notifying the completion of the message reception.	
Header	r_hfdsl_rzt2_if.h	
Declaration	void hfdsl_int_mrcv_callback(uint16_t* msg_data);	
Description	This callback function is registered with the R_HFDSL_Control (R_HFDSL_CMD_MSG) function. It is called when the INT_tovr interrupt occurs and data storage of the received message is completed. This function is the context of the interrupt handler. To ensure interrupt responsiveness, return immediately. The function name is an example and can be set freely.	
Argument	msg_data[]	: FIFO_DATA register value (short and long messages) The FIFO_DATA register (short and long messages) values are stored for the number of received data size. The value of this argument remains valid until the next INT_tovr interrupt is generated.
Return value	None	

4.5.5 hfdsl_int_init_callback

hfdsl_int_init_callback	
Synopsis	Indicating the generation of an INT_nml interrupt by the INT_END bit of the EVENT register.
Header	r_hfdsl_rzt2_if.h
Declaration	void hfdsl_int_init_callback (void);
Description	This callback function is registered with the member variable p_cb_init of the argument r_hfdsl_info_t structure of the R_HFDSL_Open function. It is called when an INT_nml interrupt is generated by the INIT_END bit of the EVENT register. This function is the context of the interrupt handler. To ensure interrupt responsiveness, return immediately. The function name is an example and can be set freely.
Argument	None
Return value	None

4.6 Interrupt Handler

4.6.1 hfdsl_int_nml_isr_ch0

hfdsl_int_nml_isr_ch0	
Synopsis	Interrupt handler for the INT_nml ch0
Header	-
Declaration	static void hfdsl_int_nml_isr_ch0(void);
Description	An interrupt handler for the INT_nml interrupt. If the source of an interrupt is the INT_END bit of the EVENT register, function hfdsl_int_init_callback is called as a callback function. If the source of an interrupt is other bits of the EVENT register, function hfdsl_int_nml_callback is called as a callback function.
Argument	None
Return value	None

4.6.2 hfdsl_int_nml_isr_ch1

hfdsl_int_nml_isr_ch1	
Synopsis	Interrrupt handler for the INT_nml ch1
Header	-
Declaration	static void hfdsl_int_nml_isr_ch1(void);
Description	An interrupt handler for the INT_nml interrupt. If the source of an interrupt is the INT_END bit of the EVENT register, function hfdsl_int_init_callback is called as a callback function. If the source of an interrupt is other bits of the EVENT register, function hfdsl_int_nml_callback is called as a callback function.
Argument	None
Return value	None

4.6.3 hfdsl_int_err_isr_ch0

hfdsl_int_err_isr_ch0	
Synopsis	Interrupt handler for the INT_err ch0
Header	-
Declaration	static void hfdsl_int_err_isr_ch0(void);
Description	An interrupt handler for the INT_err interrupt. If the interrupt is generated, function hfdsl_int_err_callback is called as a callback function.
Argument	None
Return value	None

4.6.4 hfdsl_int_err_isr_ch1

hfdsl_int_err_isr_ch1	
Synopsis	Interrupt handler for the INT_err ch1
Header	-
Declaration	static void hfdsl_int_err_isr_ch1(void);
Description	An interrupt handler for the INT_err interrupt. If the interrupt is generated, function hfdsl_int_err_callback is called as a callback function.
Argument	None
Return value	None

4.6.5 hfdsl_int_tovr_isr_ch0

hfdsl_int_tovr_isr_ch0	
Synopsis	Interrupt handler for the INT_tovr ch0
Header	-
Declaration	static void hfdsl_int_tovr_isr_ch0(void);
Description	An interrupt handler for the INT_tovr interrupt. If message reception is completed, the function hfdsl_int_mrcv_callback is called. If reception of one H-frame is completed, the function hfdsl_int_raw_callback is called.
Argument	None
Return value	None

4.6.6 hfdsl_int_tovr_isr_ch1

hfdsl_int_tovr_isr_ch1	
Synopsis	Interrupt handler for the INT_tovr ch1
Header	-
Declaration	static void hfdsl_int_tovr_isr_ch1(void);
Description	An interrupt handler for the INT_tovr interrupt. If message reception is completed, the function hfdsl_int_mrcv_callback is called. If reception of one H-frame is completed, the function hfdsl_int_raw_callback is called.
Argument	None
Return value	None

4.7 Interrupts

Table 4.2 lists the interrupts for the HFDSL driver.

Table 4.2 Interrupts for the HFDSL Driver

Interrupts	ID	Outline
INT_nml0 (ENCIF_INT0)	372	Generated when any bit in the ch0 EVENT register is updated to 1.
INT_nml1 (ENCIF_INT4)	376	Generated when any bit in the ch1 EVENT register is updated to 1.
INT_err0 (ENCIF_INT1)	373	Generated when any bit in the ch0 EVENT_ERR register is updated to 1.
INT_err1 (ENCIF_INT5)	377	Generated when any bit in the ch1 EVENT_ERR register is updated to 1.
INT_tovr0 (ENCIF_INT2)	374	Generated when the number of receptions in the ch0 FIFO_DATA register exceeds the threshold.
INT_tovr1 (ENCIF_INT6)	378	Generated when the number of receptions in the ch1 FIFO_DATA register exceeds the threshold.

4.8 Constants and Error Codes

The tables below list the constants and error codes. For the definitions, see the respective tables.

Table 4.3 User-defined Constants for the HFDSL Driver (r_hfdsl_rzt2_config.h)

Constant Name	Setting	Description
R_HFDSL_ES_PRDY	0008h	Setting of the ES and PRDY bits in the SYS_CTRL register
R_HFDSL_MAXACC	1023	Setting of the MAXACC register
R_HFDSL_ACC_ERR	31	Setting of the ACC_ERR register
R_HFDSL_MAXDEV	65535	Setting of the MAXDEV register
R_HFDSL_MASK	64h	Setting of the MASK_ERR register *1
R_HFDSL_MASK_ERR	00FF33DFh	Setting of the MASK_ERR register *1
R_HFDSL_RAW_EN	FFh	Setting of the RAW_EN register *2
R_HFDSL_DELAY_UPP_LIMIT	09h	The allowable upper limit on the delay time due to the length of the cable *3
R_HFDSL_STUFF	08h	Setting of the STUFF register
R_HFDSL_EXLEN	00h	Setting of the EXLEN register
R_HFDSL_EXTRA	0003h	Setting of the EXTRA register

Note 1. To change R_HFDSL_MASK and R_HFDSL_MASK_ERR, change processing of the hfdsl_int_nml_isr_ch0 function, hfdsl_int_nml_isr_ch1 function, hfdsl_int_err_isr_ch0 function, hfdsl_int_err_isr_ch1 function in accord with the settings in the R_HFDSL_MASK and R_HFDSL_MASK_ERR.

2. To change R_HFDSL_RAW_EN, change the shub variable given in “Table 4.6, Major Static Variables”, and the hfdsl_shub and hfdsl_int_raw_callback functions in accord with the settings in R_HFDSL_RAW_EN. In this sample program, the array shub holds eleven values for the FIFO_DATA register.
3. The allowable upper limit on delay time due to the length of the cable is set to a value from 900 ns to 1000 ns (corresponding to a length of approximately 100 m). If the allowable upper limit is to be changed, change the setting of this constant.

Table 4.4 Error Codes

Constant Name	Setting	Description
R_HFDSL_SUCCESS	0	Normal termination
R_HFDSL_ERR_INVALID_ARG	-1	Argument error
R_HFDSL_ERR_ACCESS	-2	API execution order error
R_HFDSL_ERR_INIT	-3	Failure in initialization of the HFDSL controller and encoder

4.9 Fixed-width Integers

Table 4.5 lists the fixed-width integers for the sample code. The fixed-width integers used in the sample code are defined in the standard library.

Table 4.5 Fixed-width Integers for the Sample Code

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

4.10 Structures, Unions, and Enumerated Types

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

4.10.1 Structures

(1) r_hfdsl_info_t

Information on initialization of the HFDSL driver

```
typedef struct
{
    r_hfdsl_int_nml_cb_t      p_cb_nml;  Pointer to the callback function to be called when an INT_nml
                                         interrupt is generated.
                                         For details, see section “4.5.1, hfdsl_int_nml_callback”. *1, *2
    r_hfdsl_int_err_cb_t      p_cb_err;  Pointer to the callback function to be called when an INT_err
                                         interrupt is generated.
                                         For details, see section “4.5.2, hfdsl_int_err_callback”.  *1
    r_hfdsl_int_fifo_raw_cb_t p_cb_raw;  Pointer to the callback function to be called when an INT_tovr
                                         interrupt is generated.
                                         For details, see section “4.5.3, hfdsl_int_raw_callback”. *1
    r_hfdsl_int_init_cb_t     p_cb_init  Pointer to the callback function to be called when an INT_nml
                                         interrupt is generated in response to the INIT_END bit in the
                                         EVENT register being set to 1.
                                         For details, see section “4.5.5, hfdsl_int_init_callback”. *1
} r_hfdsl_info_t
```

Note 1. This function is not called if NULL is specified.
2. This function is not called when an INT_nml interrupt is generated in response to the MRCV or INIT_END bit in the EVENT register being set to 1.

(2) r_hfdsl_pos_t

For storing fast position

```
typedef struct
{
    bool      all;   Enables the member variable posh.
                     (true: The member variable posh is enabled,
                      false: The member variable is disabled)
    uint8_t   posh;  Holds bits [39:32] of the fast position.
                     The value is updated when the member variable all is true.
    uint32_t  pos;   Holds bits [31:0] of the fast position.
} r_hfdsl_pos_t
```

(3) r_hfdsl_vpos_t

For storing the safe position

```
typedef struct
{
    uint8_t   vposh;  Holds bits [39:32] of the safe position.
    uint32_t  vpos;   Holds bits [31:0] of the safe position.
    uint16_t  crc;    Holds the CRC of the vertical channel
} r_hfdsl_vpos_t
```

(4) r_hfdsl_send_msg_t

For storing message data for transmission.

```
typedef struct
{
    uint16_t            *p_data;    Pointer to the array which holds message data for transmission.
                                    Set the pointer to the array which holds message data for
                                    transmission. If the size of data for transmission exceeds 14,
                                    R_HFDSL_ERR_INVALID_ARG occurs.
    r_hfdsl_msg_cb_t    p_cb_msg;   Pointer to the callback function to be called when a message is
                                    received.
                                    For details, see section “4.5.4, hfdsl_int_mrcv_callback”.
                                    Be sure to set the address of hfdsl_int_mrcv_callback.
} r_hfdsl_send_msg_t
```

4.10.2 Unions

Not used.

4.10.3 Enumerated Types

Not used.

4.11 Description of the Sample Program

4.11.1 Outline of Operations

This sample program supports the encoder (EFM50-0KF0A023A from SICK AG) compliant with the HIPERFACE DSL communications protocols specification. It handles the following processing.

1) Indicates the following information by using a command input from the console.
   A) Fast and safe positions
   B) Rotational velocity of the motor
   C) Results of transmission and reception of long messages (the type of encoder from resources)
   D) Sensor Hub Channel data
2) Runs in SYNC mode.
3) This sample program ends by detecting a protocol reset.

(1) System Block Diagram

Figure 4.1 shows a block diagram of the system.

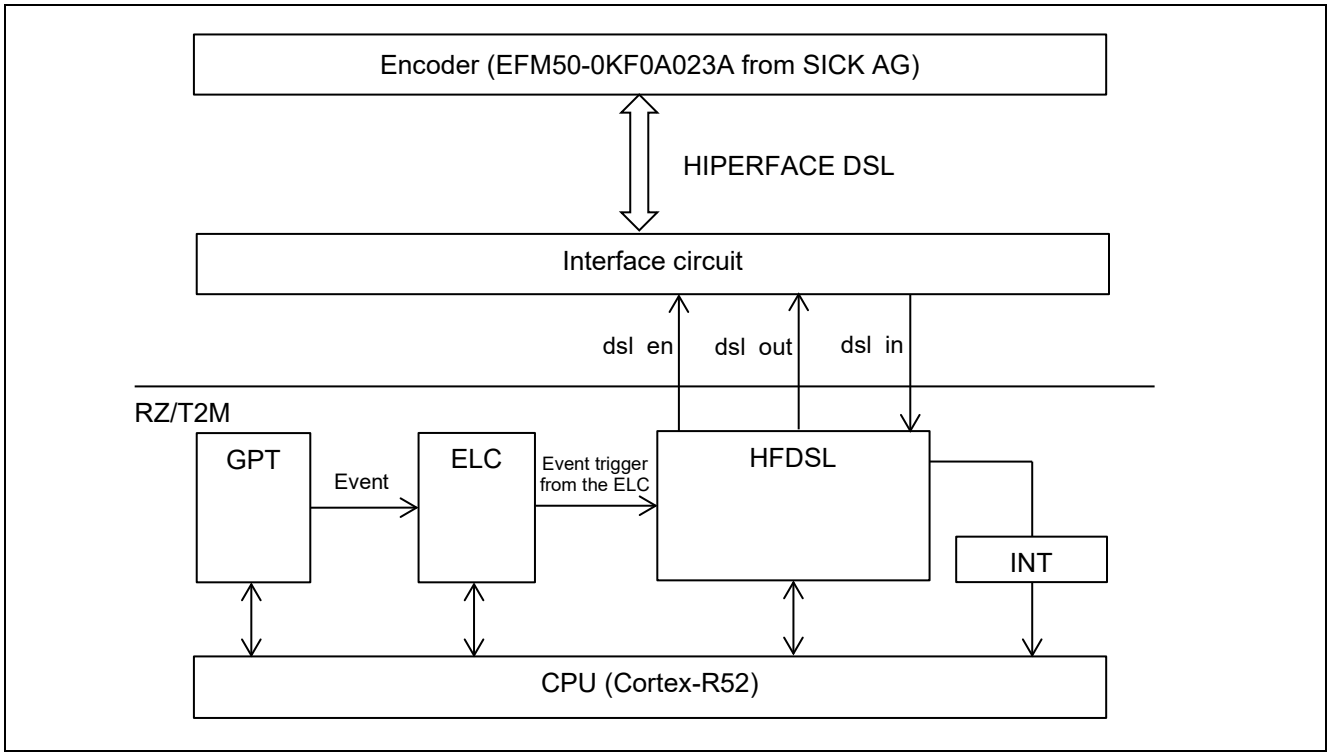

Figure 4.1 System Block Diagram

(2) Software Configuration

Figure 4.2 is a block diagram of the software.

The HFDSL driver has six sections: the opening process part configured of function R_HFDSL_Open, the closing process part configured of function R_HFDSL_Close, the protocol initialization, positional value acquisition, and message transmission parts configured of function R_HFDSL_Control, and the data reception part (interrupt handler) configured of the callback function.

The HFDSL driver controller section of the sample program controls the HFDSL driver, acquires the positional value, and sends messages. The results indication section (callback) displays the results of data reception.

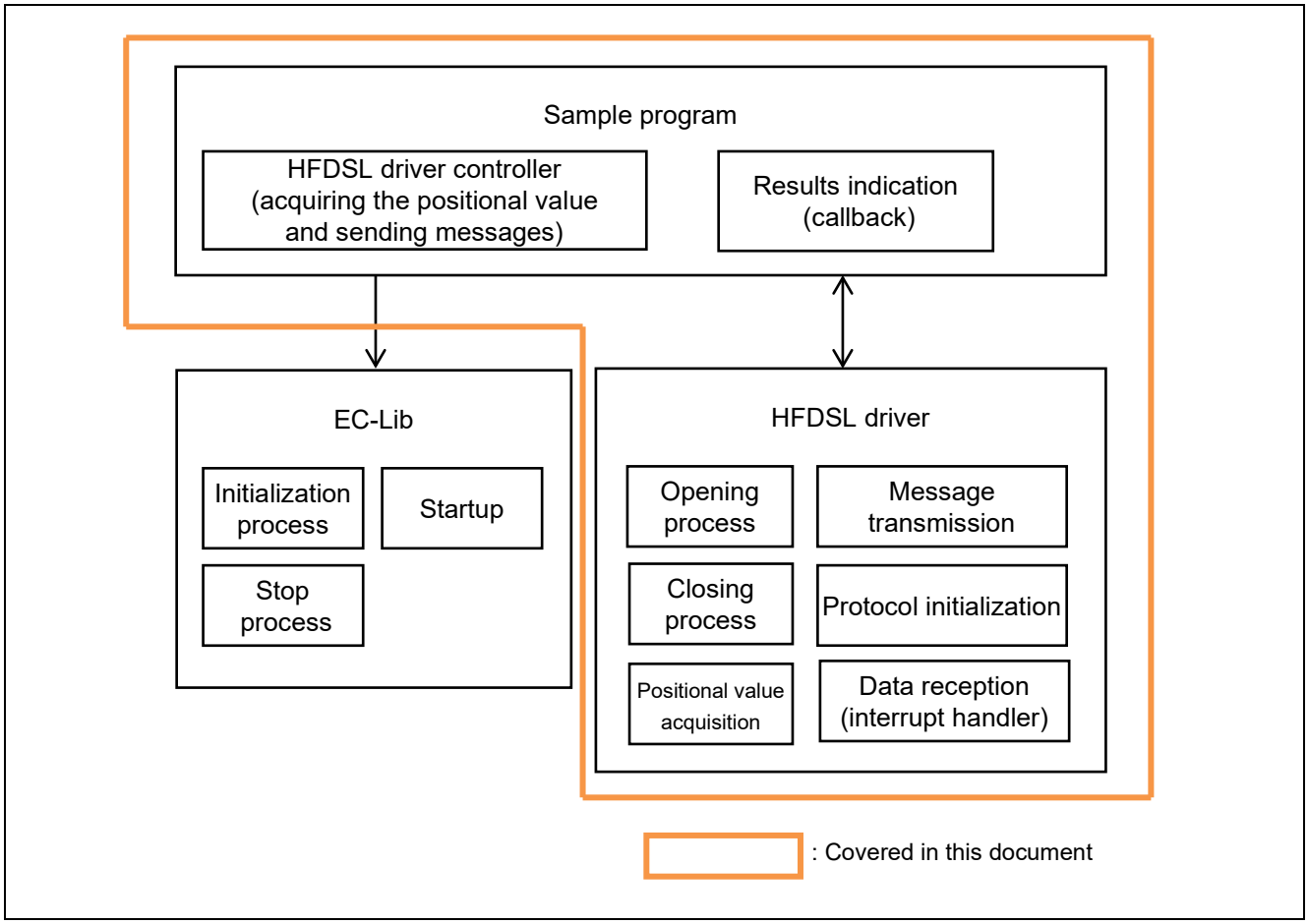

Figure 4.2 Software Configuration

4.11.2 Variables for the Sample Program

Table 4.6 lists the major static variables.

Table 4.6 Major Static Variables

Type	Variable Name	Description
bool	mrcv_flg	Message transfer completed flag (true: Transfer of messages has been completed false: Transfer of messages is in progress)
uint32_t	err_info	Holds the INT_err interrupt source.
uint32_t	pos_rot	Holds the number of rotations with the fast position.
uint32_t	pos_res	Holds the angle of the fast position.
uint32_t	vpos_rot	Holds the number of rotations with the safe positions.
uint32_t	vpos_res	Holds the angle of the safe position.
uint32_t	vel	Holds the rotational velocity of the motor.
uint16_t	lmsg_recv[LMSG_RECV_SIZE]	Holds received data in long messages.
uint16_t	shub[FIFO_DEP_MAX]	Holds the sensor hub data including received data.
bool	init_end_flg	Protocol initialization completion flag (true: Protocol initialization is completed. false: Protocol initialization is not completed.)

4.11.3 Constants for the Sample Program

Table 4.7 lists the major constants for the sample program.

Table 4.7 Major Constants

Constant Name	Setting	Description
ENC_ID	00000183h	Encoder ID of EFM50-0KF0A023A *1 *2
RES_BIT	0	Position of the least significant bit of the positional information in the POS register *1
RES_MASK	007FFFFFh	Masking of the positional information in the POS register *1
RES_MASK_H	00000000h	Masking of the positional information in the POS_H register *1
ROT_BIT	23	Position of the least significant bit of the rotational information in the POS register *1
ROT_MASK	000001FFh	Masking of the number of rotations in the POS register *1
ROT_MASK_H	00000E00h	Masking of the number of rotations in the POS_H register *1
LMSG_RECV_SIZE	16	Maximum size of received data in long messages
FIFO_DEP_MAX	11	Data size of FIFO_DATA register
TIMEOUT_UNIT	1000	Setting of timeout unit (1 ms)
TIMEOUT_COUNT	1000	Setting of timeout (1 ms x 1000)

Note: 1. To run the sample program with an encoder other than an EFM50-0KF0A023A, change the settings to suit the specifications of the connected encoder.
2. Refer to the “HIPERFACE DSL® MASTER Integration Manual” for details.

The figure below shows the mechanism for storing the positional and rotational information.

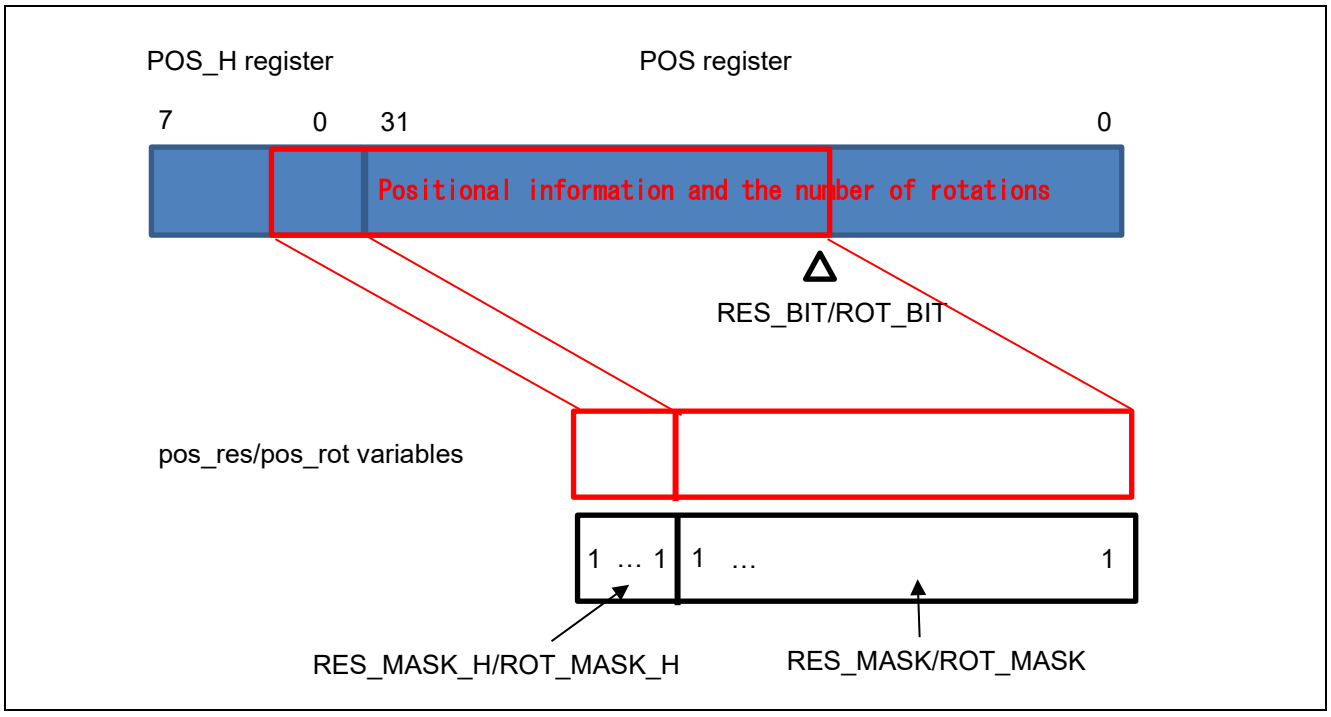

Figure 4.3 Mechanism for Storing the Positional and Rotational Information

4.11.4 Flowchart of Main Processing

The flowchart below shows processing by the main function.

Processing marked with * in the figure is shown separately in a subsequent flowchart.

(1) Flowchart of enc_main

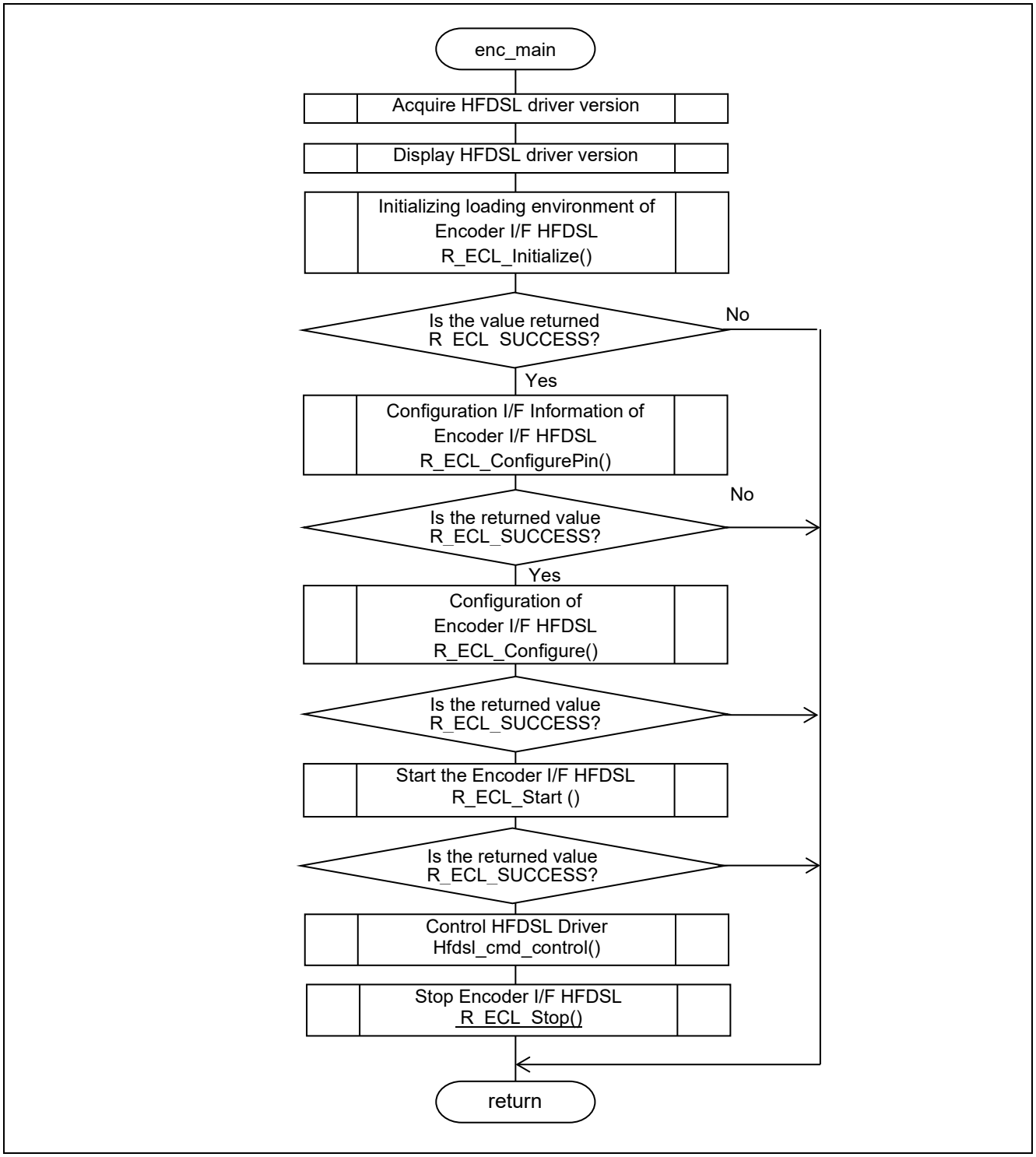

Figure 4.4 Flowchart of the enc_main Function

(2) Flowchart of hfdsl_cmd_control

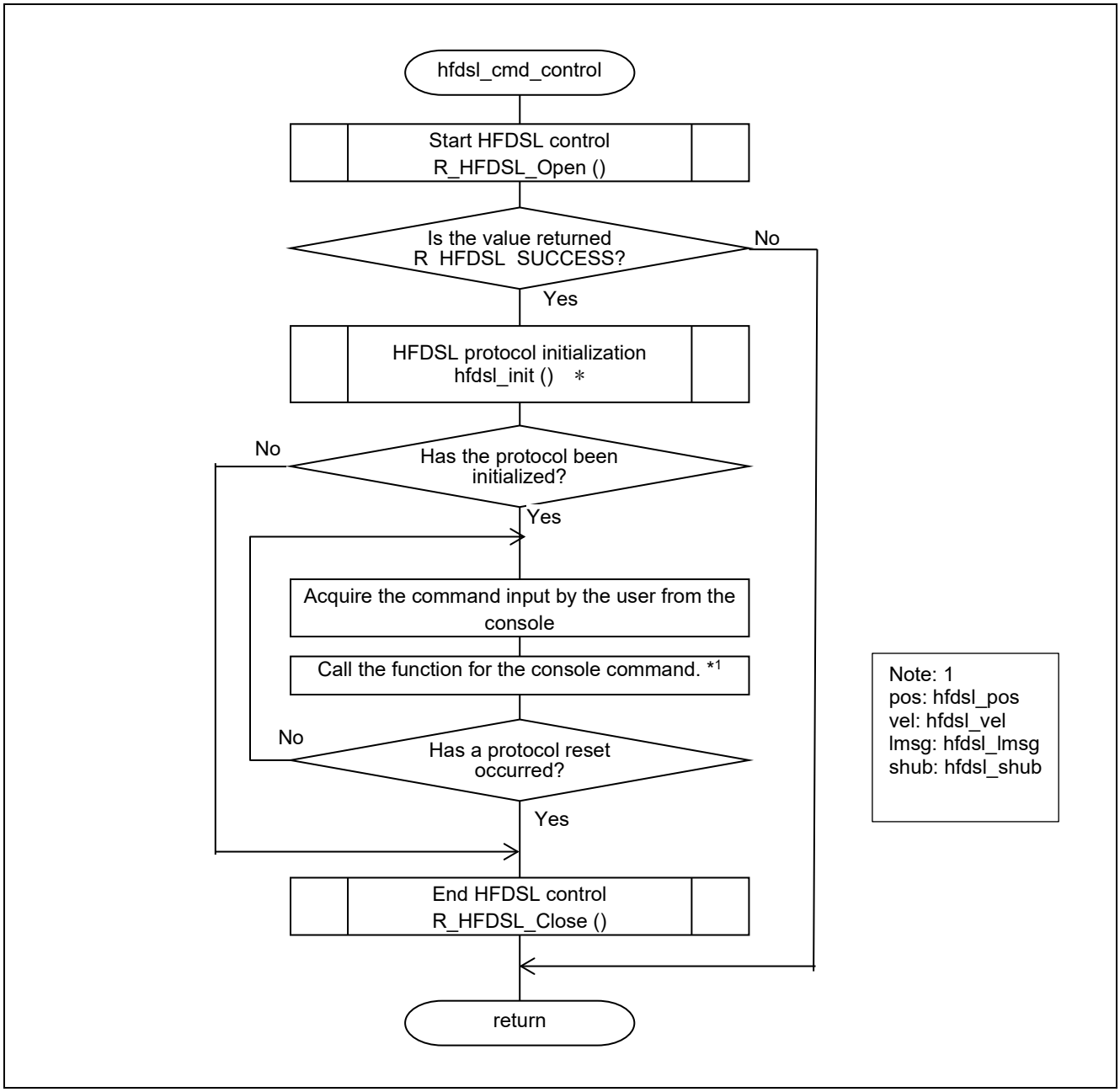

Figure 4.5 Flowchart of the hfdsl_cmd_control

(3) Flowchart of hfdsl_init

This function initializes the protocol.

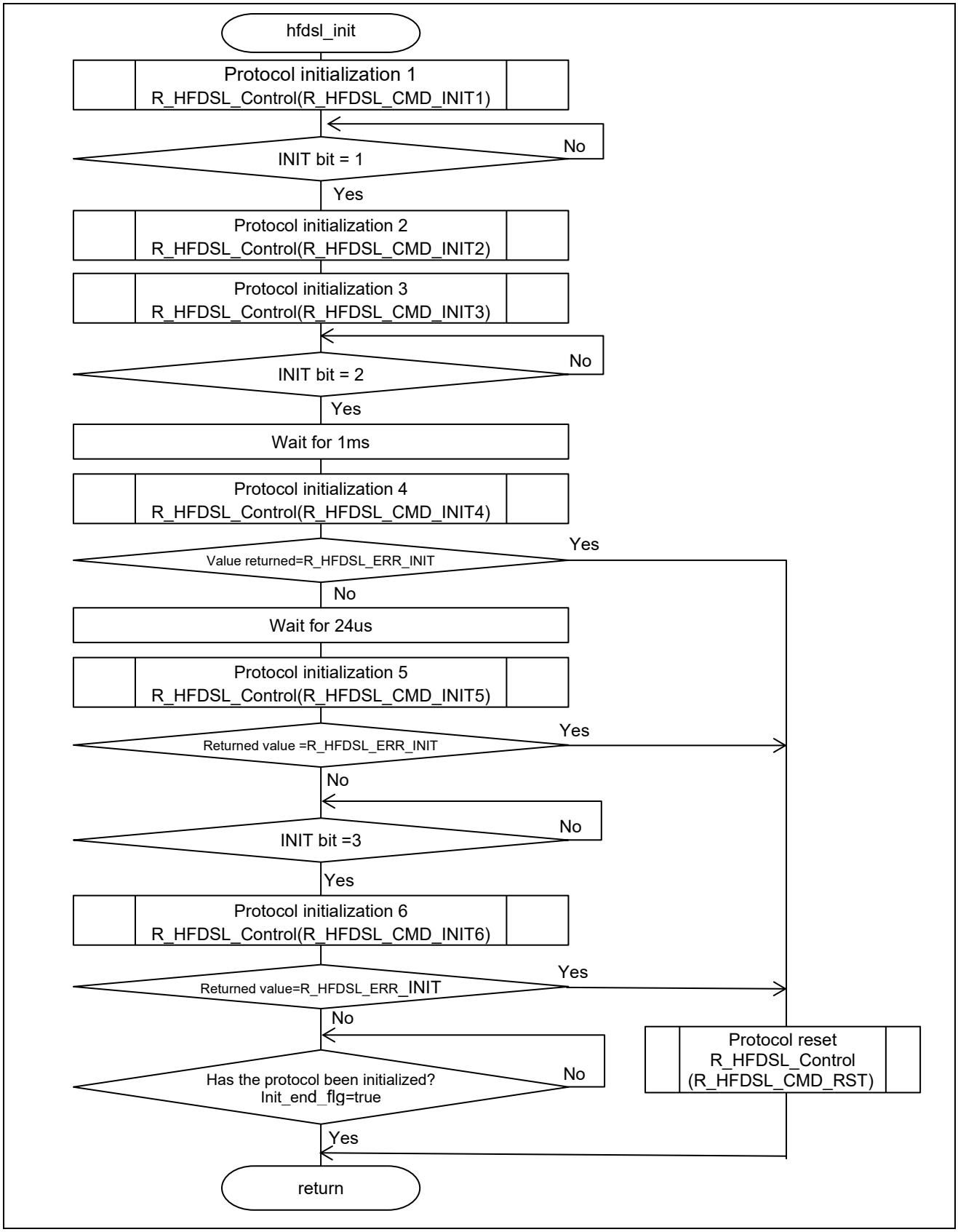

Figure 4.6 Flowchart of the hfdsl_init

(4) Flowchart of hfdsl_pos, hfdsl_vel, hfdsl_shub

These functions are executed in response to input of the console commands “pos”, “vel”, and “shub” and indicate the acquired data. The functions corresponding to the respective console commands and details of the items displayed are below.

Table 4.8 Functions Corresponding to the Console Commands “pos”, “vel”, “shub”

Console Command	Corresponding Function	Items Displayed
pos	hfdsl_pos	pos_rot, pos_res vpos_rot, vpos_res err_info
vel	hfdsl_vel	vel, err_info
shub	hfdsl_shub	shub, err_info

Since the procedures for processing of the hfdsl_pos, hfdsl_vel, and hfdsl_shub functions are similar, they are shown in the same flowchart.

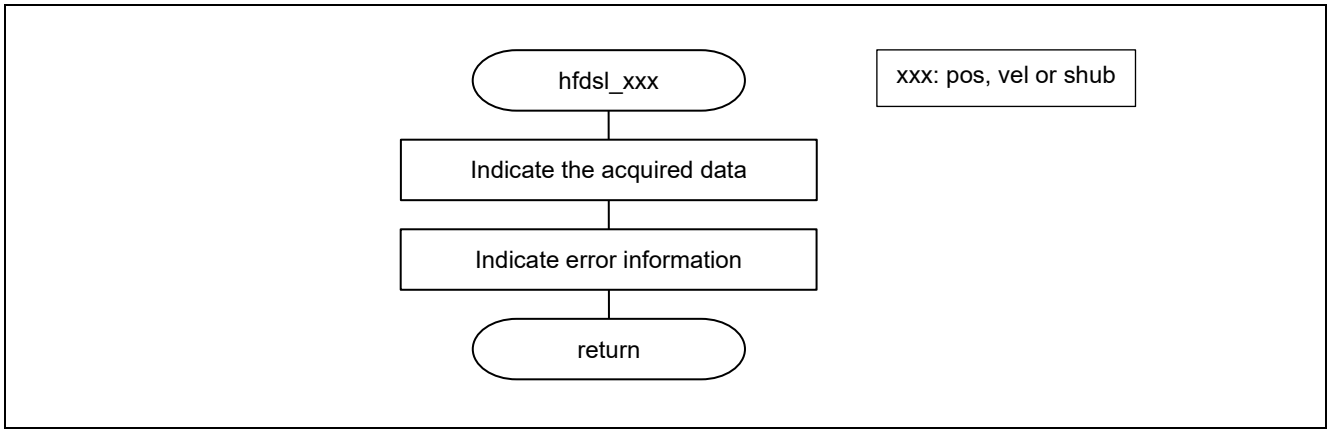

Figure 4.7 Flowchart of the hfdsl_pos, hfdsl_vel, hfdsl_shub

(5) Flowchart of hfdsl_lmsg

This function is executed in response to input of the console command “lmsg”.

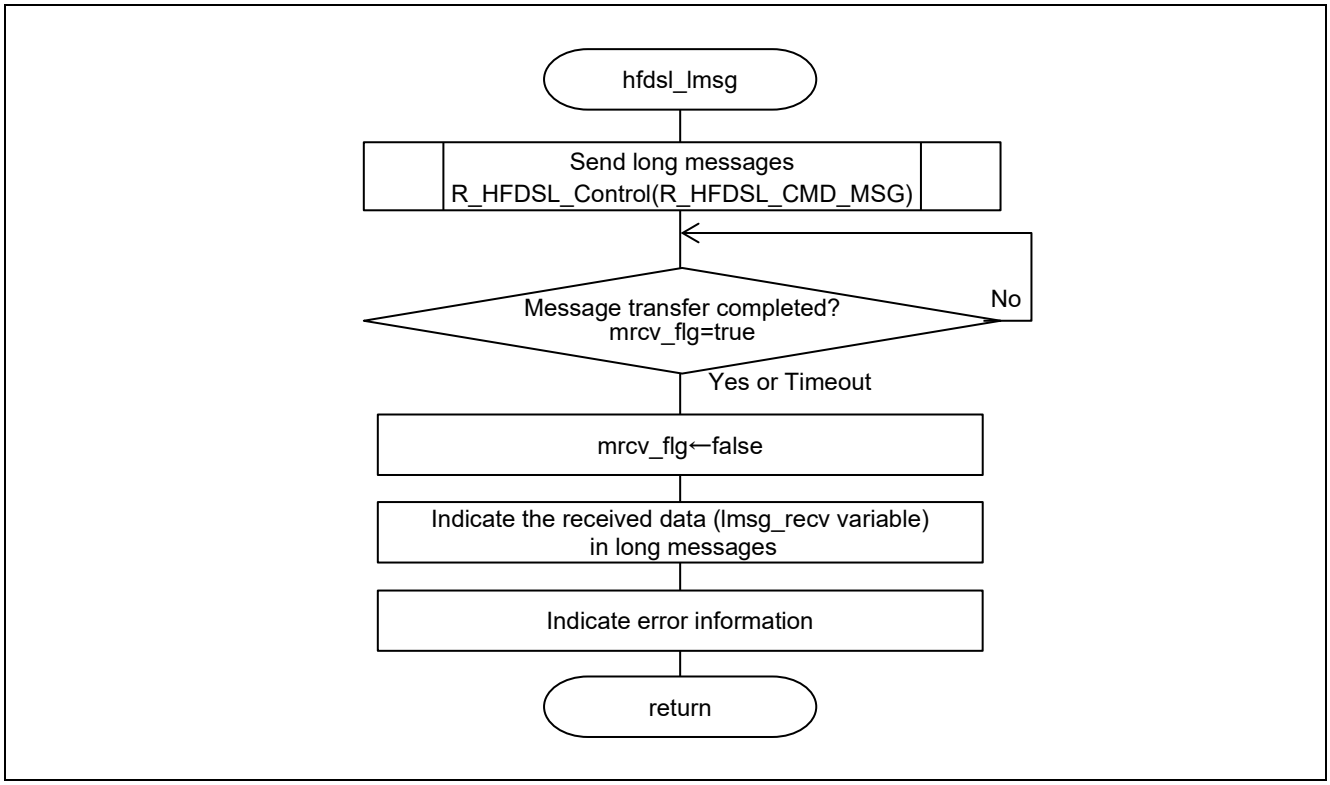

Figure 4.8 Flowchart of the hfdsl_lmsg

(6) Flowchart of hfdsl_int_nml_callback

This callback function is called in response to generation of an INT_nml interrupt.

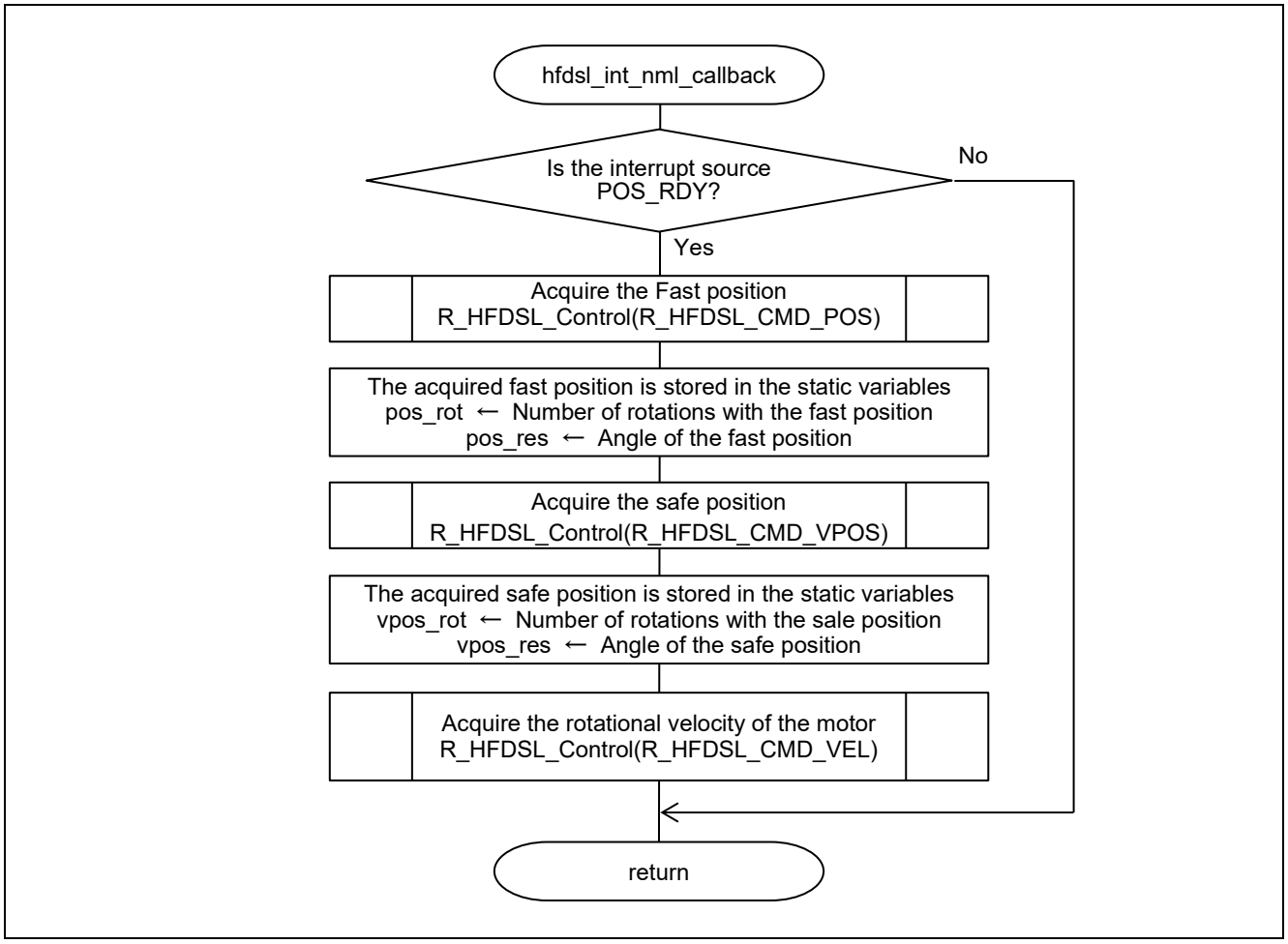

Figure 4.9 Flowchart of the hfdsl_int_nml_callback

(7) Flowchart of hfdsl_int_err_callback

This callback function is called in response to generation of an INT_err interrupt.

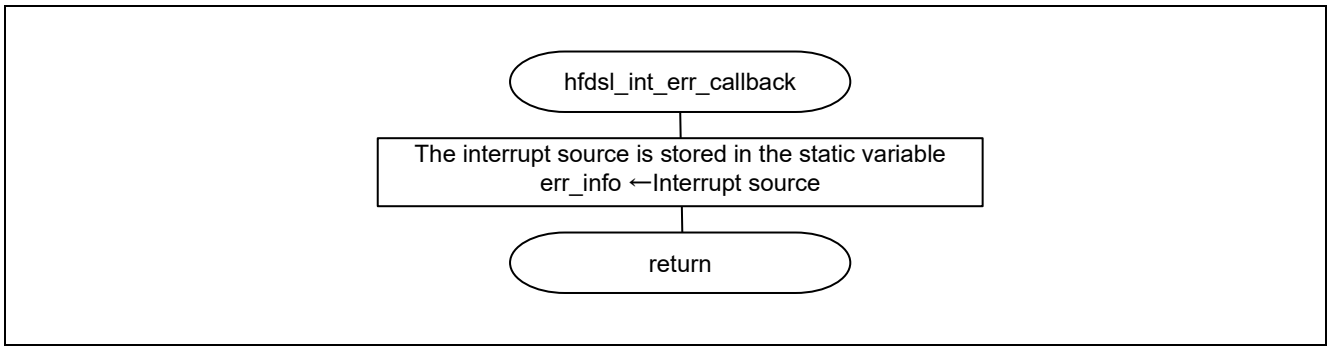

Figure 4.10 Flowchart of the hfdsl_int_err_callback

(8) Flowchart of hfdsl_int_raw_callback

This callback function is called when an INT_tovr interrupt is generated.

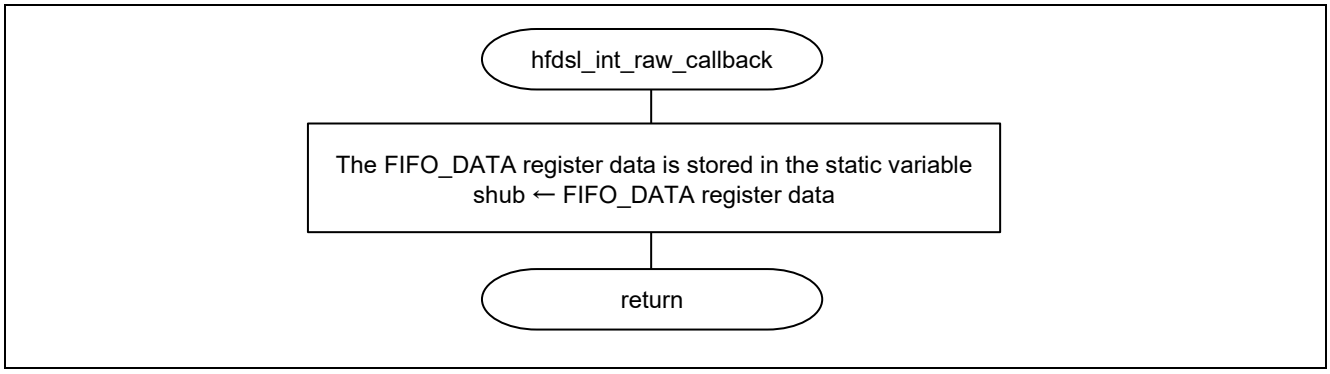

Figure 4.11 Flowchart of the hfdsl_int_raw_callback

(9) Flowchart of hfdsl_int_mrcv_callback

This callback function is called when the INT_tovr interrupt occurs and data storage of the received message is completed.

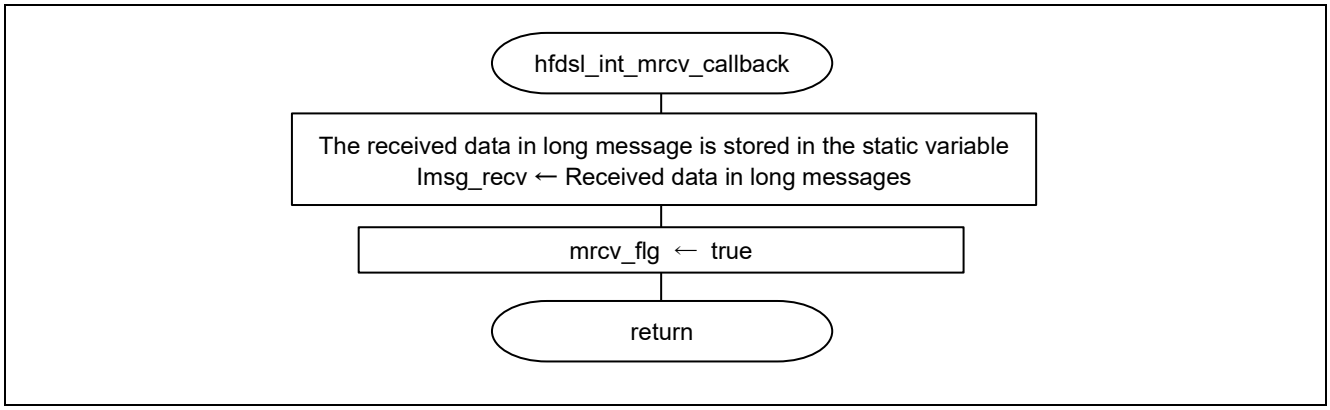

Figure 4.12 Flowchart of the hfdsl_int_mrcv_callback

(10) Flowchart of hfdsl_int_init_callback

This callback function is called when an INT_nml interrupt is generated in response to the INIT_END bit in the EVENT register being set to 1.

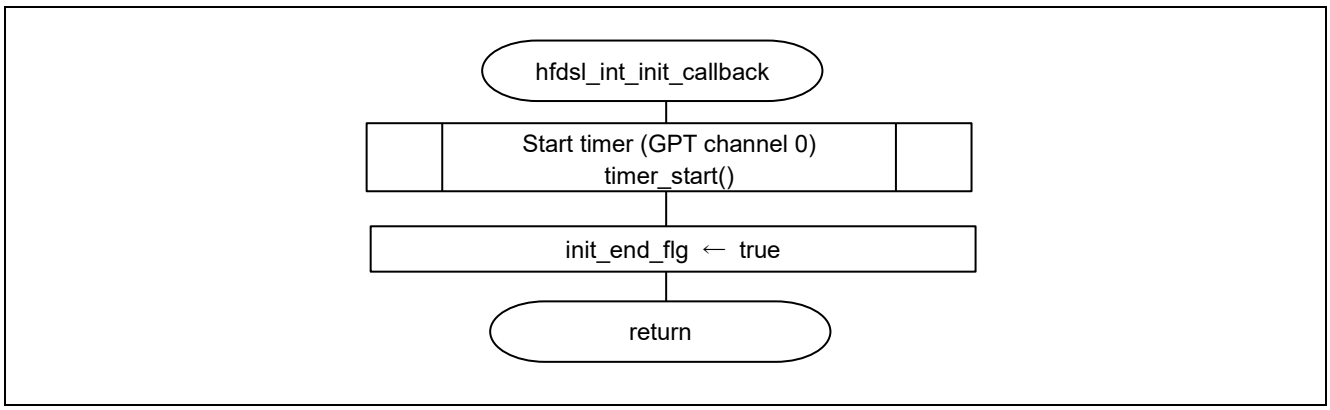

Figure 4.13 Flowchart of the hfdsl_int_init_callback

4.11.5 Operation Sequence

(1) Startup Sequence

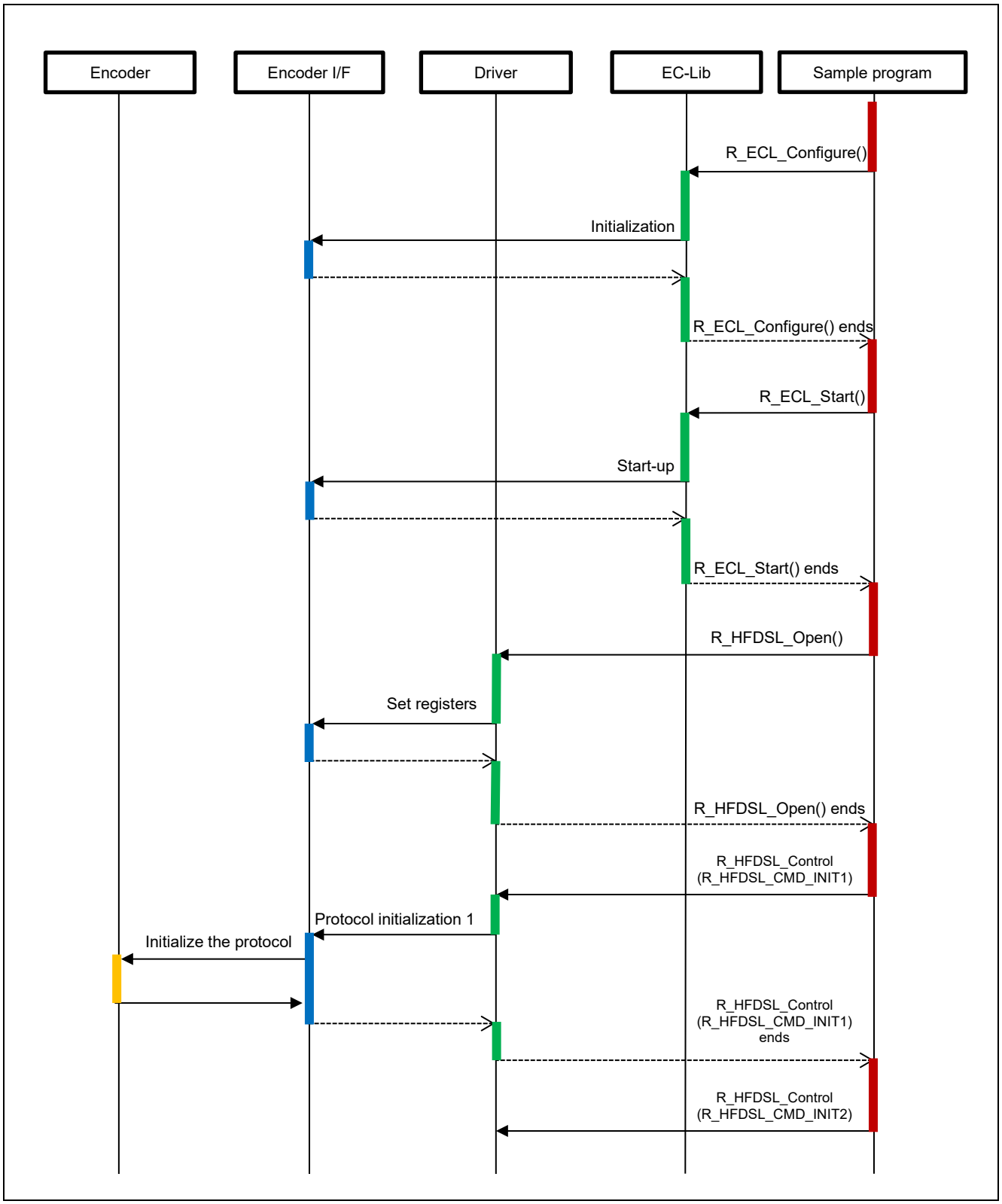

Figure 4.14 Startup Sequence Diagram (1/3)

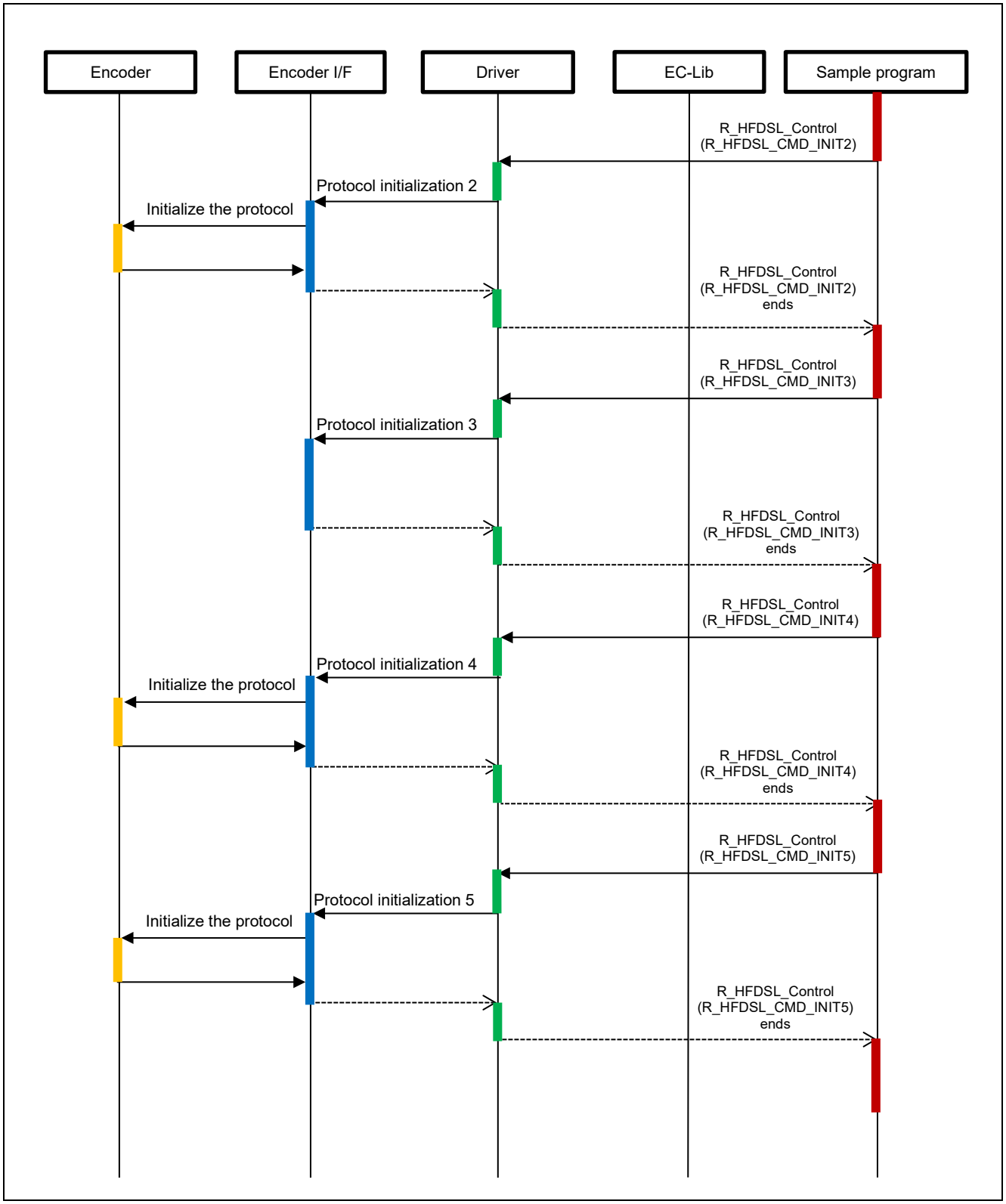

Figure 4.15 Startup Sequence Diagram (2/3)

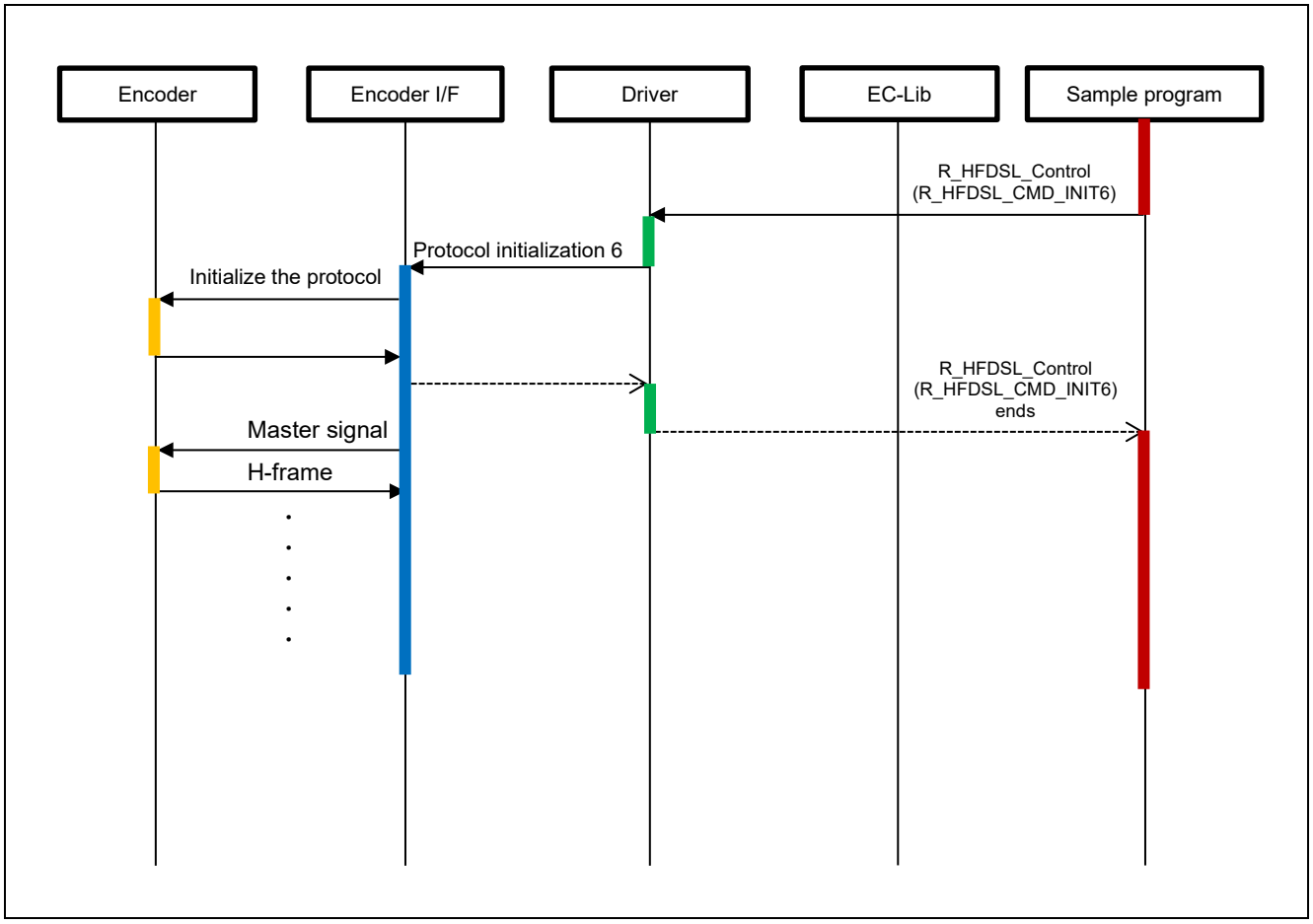

Figure 4.16 Startup Sequence Diagram (3/3)

(2) Fast Position in SYNC Mode Acquisition Sequence

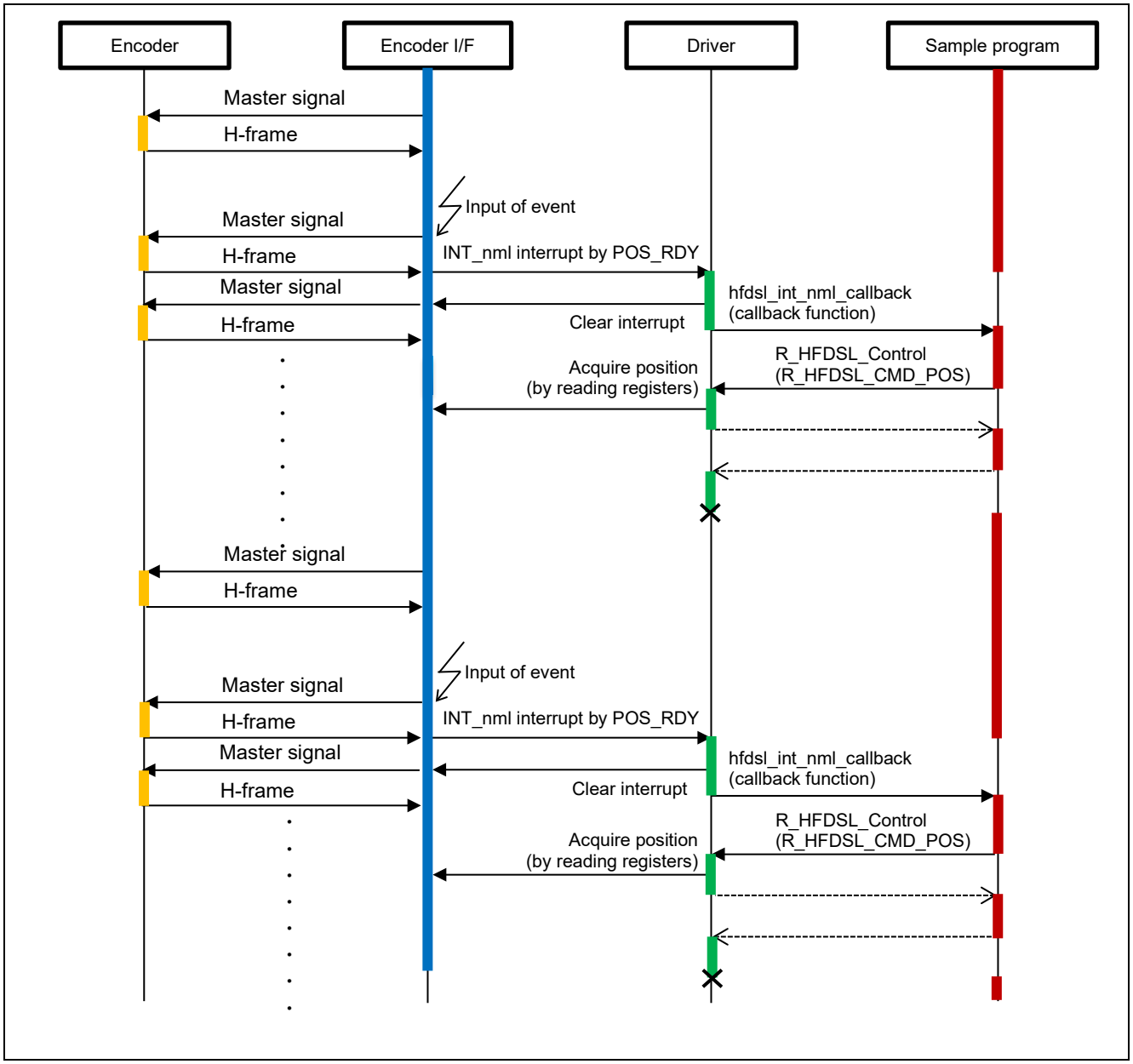

Figure 4.17 Fast Position in SYNC Mode Acquisition Sequence Diagram

(3) Sensor Hub Channel Acquisition Sequence

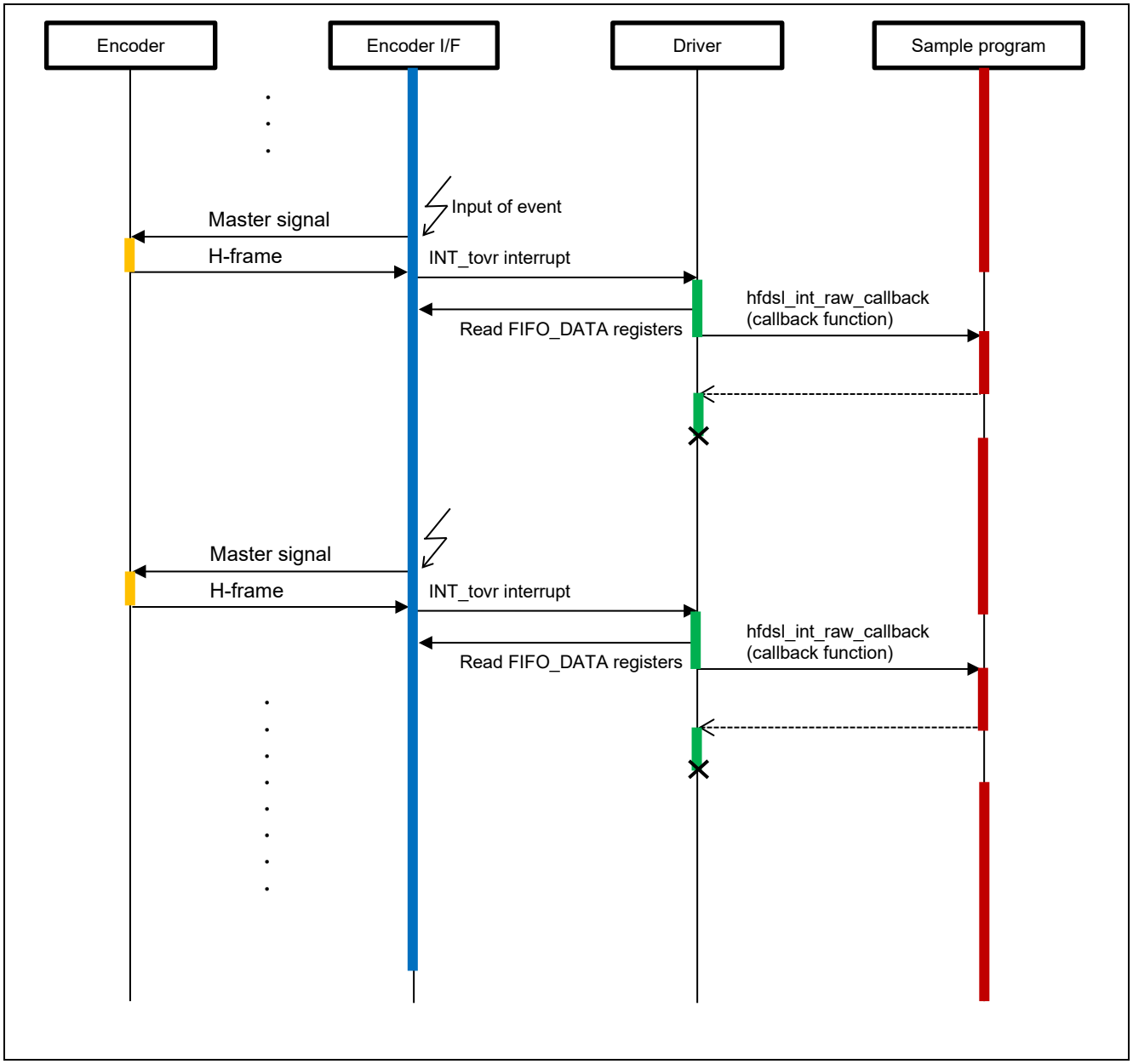

Figure 4.18 Sensor Hub Channel Acquisition Sequence Diagram

(4) Message Transfer Sequence

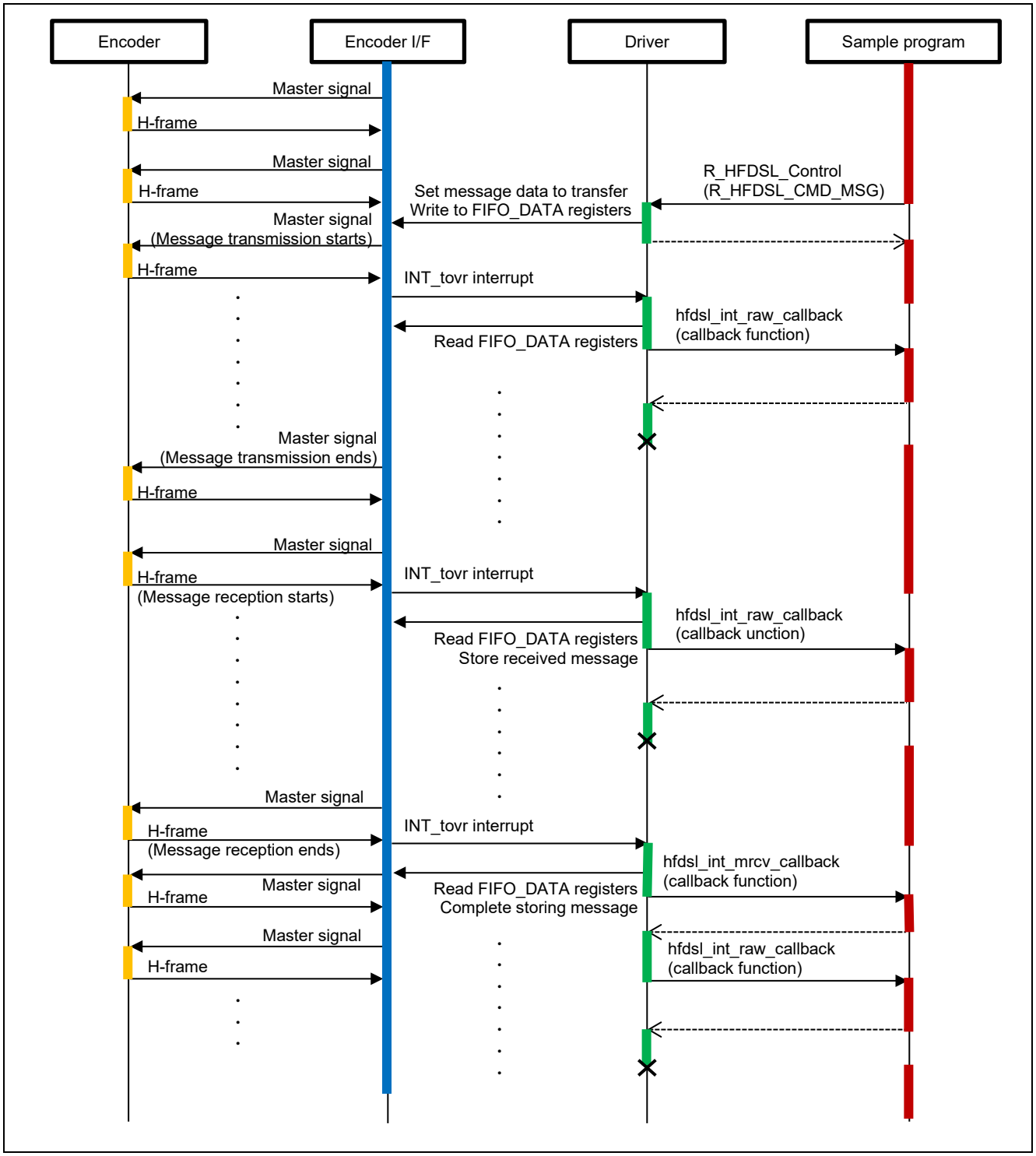

Figure 4.19 Message Transfer Sequence Diagram

(5) Stop Sequence

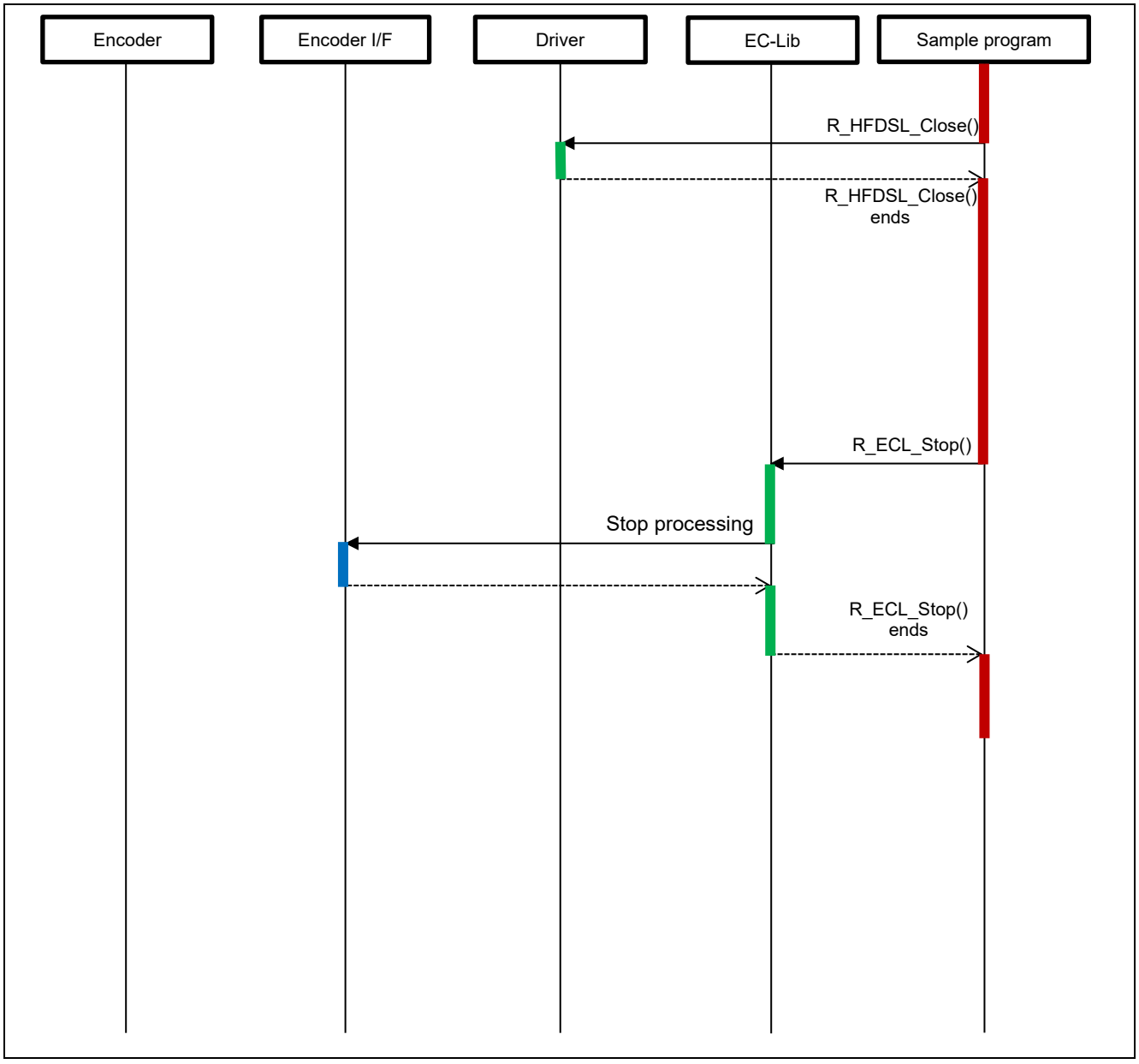

Figure 4.20 Stop Sequence Diagram

4.11.6 Console Commands

The commands available for input from the console are listed below.

Table 4.9 Console Commands

Command	Description
pos	Indicates the fast and safe positions
vel	Indicates the rotational velocity of the motor.
lmsg	Among the encoder resources, acquire the type of the encoder as a long message.
shub	Indicates a series of received data including sensor hub data.

(1) Result of Running

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

```
HFDSL sample program start
R_HFDSL_GetVersion = 4.0


hfdsl >
```

(2) Example of Command Execution

It is an example of executing pos command. Fast position, Safe position and Error information are reported as the response from the encoder.

```
hfdsl >pos
Fast position
    Rotations   : 0x00000997
    Angle       : 0x00028AF4
Safe position
    Rotations   : 0x00000997
    Angle       : 0x00028AF4
Error information
    EVENT_ERR   : 0x00010040
```

5. Sample Code

The sample code 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	5,6,43	Corrected the description of operating voltage Corrected the description of the device used Corrected the list of used pins Deleted memory size description
1.20	Nov.01.22	7 18 19, 20 21 21 23 28 33 35 40 41	Add tovr interrupt description. Change descriptions of hfdsl_int_raw_callback and hfdsl_int_mrcv_callback to called by INT_tovr interrupt. Add INT_tovr interrupts. Change descriptions of callback functions called by the handlers. Remove R_HFDSL_TH_RAW from table 4.3 Correct the number of stored values in the table 4.3 note 2. Change description of the element pcb_raw in the structure r_hfdsl_info_t. Add FIFO_DEP_MAX, TIMEOUT_UNIT, TIMEOUT_COUNT in the table 4.7 main constants. Correct description of condition in the fig. 4.8. Add description of timeout condition. Add 4.11.4(8) flowchart of hfdsl_int_raw_callback. Add 4.11.5(3) sequence of acquiring the sensor hub channel. Change sequence diagram of fig. 4.19 message transfer sequence.
2.00	Jun 07.24	5 22 36 to 42	Update description of the board name. Remove description about location of integer type definition. Update sequence diagrams.
3.00	Oct 17.25	1, 4, 5 4, 27 8 to 17 42	Change description for trademarks. Update notes of HIPERFACE DSL MASTER Integration Manual. Revise to unify description for functions. Add example of command execution.
4.00	Apr 03.26	8 to 17, 22 27	Change prefix of pointer variables to “p_”. Correct variable name of position in the fig. 4.3.

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/.