RZ/T2L Group

HIPERFACE DSL Safety sample program

Introduction

This application note explains a sample program for acquiring and indicating information including safety data from an encoder in conformance with the HIPERFACE DSL® communications protocol specification by using the encoder Interface of the RZ/T2L.

The features of the program:
• Acquiring angle information, etc. from an encoder (EDM35-2KF0A020A) compliant with the HIPERFACE DSL® communications protocol specification

Target Device

RZ/T2L

HIPERFACE DSL is a registered trademark of SICK AG.

Table of Contents

1. Specifications

Table 1.1 lists the peripheral functions 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 Functions and Applications

Peripheral Module	Application
HIPERFACE DSL controller (HDSL)	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 HDSL 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 HDSL module.
Serial Communication Interface (SCI) UART	Asynchronous communications of the SCI are used for COM port communications by using USB interface.

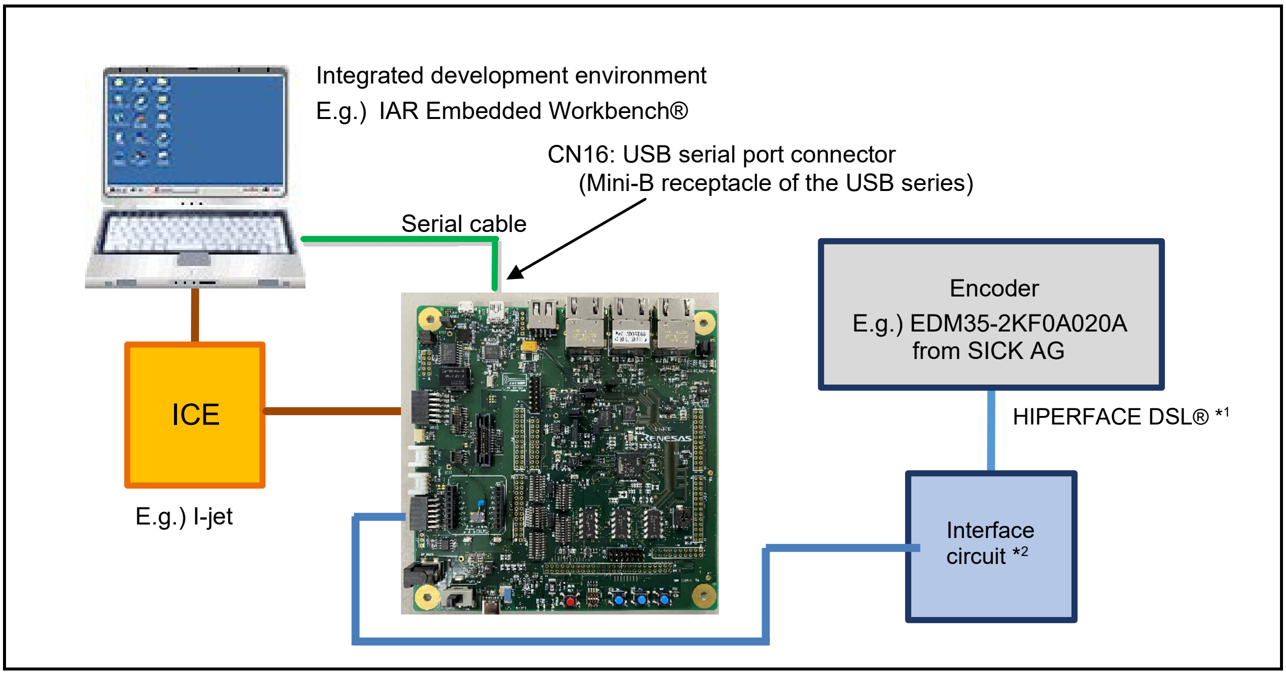
Integrated development environment
E.g.) IAR Embedded Workbench®
CN16: USB serial port connector
(Mini-B receptacle of the USB series)
Serial cable
Encoder
E.g.) EDM35-2KF0A020A
from SICK AG
HIPERFACE DSL® *1
ICE
E.g.) I-jet
Interface
circuit *2

Figure 1.1 Operating Environment

Note 1. Contact the manufacturer of the encoder you are using regarding the length of the cable that can handle transfer.

2. Refer to the HIPERFACE DSL® Implementation Manual for details of the interface circuit. The specification can be obtained by contacting SICK AG.

IAR Embedded Workbench is a registered trademark of IAR Systems.

2. Operating Environment

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

Table 2.1 Operating Environment

Item	Description
Microcomputer	RZ/T2L group
Operating frequency	CPUCLK = 800MHz
Operating voltage	1.1V(Core) / 1.8V(PLL, etc.) / 3.3V(I/O)
Integrated development environment *[1]	IAR Systems Embedded Workbench® for ARM RENESAS e^2 studio
Board	RSK+RZT2L（RTK9RZT2L0C00000BJ)
Devices	None

Note 1. Refer to the release note for the RZ/T2L Group Encoder I/F HIPERFACE DSL Safety sample program 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/T2L Group User’s Manual: Hardware”.

3.1 Pins

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

Table 3.1 Pins Used and Their Functions

Channel	Pin Name	I/O	I/O Port	Description
HFDSL0	ENCIFDI0 (dsl_in0)	Input	P02_2	Data input pin
	ENCIFDO0 (dsl_out0)	Output	P02_3	Data output pin
	ENCIFOE0 (dsl_en0)	Output	P01_7	Drive/receive control pin
HFDSL1	ENCIFDI1 (dsl_in1)	Input	P10_1	Data input pin
	ENCIFDO1 (dsl_out1)	Output	P10_0	Data output pin
	ENCIFOE1 (dsl_en1)	Output	P09_7	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/T2L Group Encoder I/F HIPERFACE DSL Safety 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	8
	R_HFDSL_Control	9
User-defined functions	hfdsl_int_nml_callback	14
	hfdsl_int_err_callback	14
	hfdsl_int_safety_callback	15
	hfdsl_int_mrcv_callback	15
Interrupt handlers	hfdsl_int_isr_ch0	16
	hfdsl_int_isr_ch1	16
	hfdsl_fpr_isr_ch0	16
	hfdsl_fpr_isr_ch1	16
	hfdsl_sp_isr_ch0	17
	hfdsl_sp_isr_ch1	17
	hfdsl_err_isr	17

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 r_hfdsl_rzt2_dat.h
Declaration	int32_t R_HFDSL_Open(const int32_t id, r_hfdsl_info_t* pinfo);
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. R_HFDSL0_ID : Specifies channel 0 R_HFDSL1_ID : Specifies channel 1 Other than those above : Setting is not allowed pinfo 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.
Returned value	R_HFDSL_SUCCESS: Normal termination R_HFDSL_ERR_INVALID_ARG: Abnormal termination (a value for a member variable of the r_hfdsl_info_t structure for a value for id or pinfo has not been specified) R_HFDSL_ERR_ACCESS: Abnormal termination
Note	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 r_hfdsl_rzt2_dat.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. R_HFDSL0_ID : Specifies Channel 0 R_HFDSL1_ID : Specifies Channel 1 Other than above : Setting is not allowed
Return Value	R_HFDSL_SUCCESS: Normal termination R_HFDSL_ERR_INVALID_ARG: Abnormal termination (the value of id was not the value specified for the encoder)
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 r_hfdsl_rzt2_dat.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 higher-order bits and the minor part of the version number is stored in the sixteen lower-order bits.

4.4.4 R_HFDSL_Control

R_HFDSL_Control	
Synopsis	Controlling operation of the encoder.
Header	r_hfdsl_rzt2_if.h r_hfdsl_rzt2_dat.h
Declaration	int32_t R_HFDSL_Control(const int32_t id, const r_hfdsl_cmd_t cmd, void *const pbuf);
Description	This function controls operations of the encoder by using the cmd argument. See Section 4.4.4(1), Protocol Initialization Commands and Section 4.4.4(2), Control Commands for the operation of the control com
Argument	id : Designates the ID code to be used. R_HFDSL0_ID : Specifies channel 0 R_HFDSL1_ID : Specifies channel 1 Other than above : Setting is not allowed cmd : Command For details, see Section 4.4.4(1), Protocol Initialization Commands and Section 4.4.4(2), Control Commands. pbuf : 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 a stipulated value.) See Section 4.4.4(1), Protocol Initialization Commands and Section 4.4.4(2), Control Commands for other returned values.

(1) Protocol Initialization Commands

(a) R_HFDSL_CMD_INIT

R_HFDSL_CMD_INIT	
Synopsis	Protcol initialization
Header	r_hfdsl_rzt2_if.h r_hfdsl_rzt2_dat.h
Declaration	int32_t R_HFDSL_Control (const int32_t id, const r_hfdsl_cmd_t cmd, void *const pbuf);
Description	Call this function after executing function R_HFDSL_Open or after a protocol reset. For how to detect a protocol reset, see Section 4.5.2 hfdsl_int_err_callback.
Argument	id : Specifies the ID to be used. R_HFDSL0_ID : Specifies channel 0 R_HFDSL1_ID : Specifies channel 1 : Setting is not allowed cmd : Specifies R_HFDSL_CMD_INIT pbuf : Specify NULL
Return value	R_HFDSL_SUCCESS: Normal termination R_HFDSL_ERR_INVALID_ARG: Abnormal termination (a value for id or pbuf is invalid) R_HFDSL_ERR_ACCESS: Abnormal termination (R_HFDSL_Open has not been executed.) R_HFDSL_ERR_INIT: Abnormal termination (Link check timed out.)
Note	Calling this API function from within a callback function is prohibited.

(b) R_HFDSL_CMD_ENCID

R_HFDSL_CMD_ENCID	
Synopsis	Check encoder ID
Header	r_hfdsl_rzt2_if.h r_hfdsl_rzt2_dat.h
Declaration	int32_t R_HFDSL_Control (const int32_t id, const r_hfdsl_cmd_t cmd, void *const pbuf);
Description	Call this function after the R_HFDSL_CMD_INIT protocol initialization command. 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_INIT after executing the control command R_HFDSL_CMD_RST.
Argument	id : Specifies the ID to be used. R_HFDSL0_ID : Specifies channel 0 R_HFDSL1_ID : Specifies channel 1 Those than above : Setting is not allowed cmd : Specifies R_HFDSL_CMD_ENCID pbuf : Encoder ID Specify the unit32_t pointer which holds the encoder ID.
Returned value	R_HFDSL_SUCCESS: Normal termination R_HFDSL_ERR_INVALID_ARG: Abnormal termination (a value for id is invalid or pbuf is null) R_HFDSL_ERR_ACCESS: Abnormal termination (R_HFDSL_CMD_INIT has not been executed) R_HFDSL_ERR_INIT: Abnormal termination (the ID of the connected encoder does not match the specified ID value)
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 r_hfdsl_rzt2_dat.h
Declaration	int32_t R_HFDSL_Control(const int32_t id, const r_hfdsl_cmd_t cmd, void *const pbuf);
Description	This function acquires the fast position by reading the fast position registers (POS4~POS0).
Argument	id : Specifies the ID to be used. R_HFDSL0_ID : Specifies channel 0 R_HFDSL1_ID : Specifies channel 1 Other than above : Setting is not allowed cmd : Specifies.R_HFDSL_CMD_POS pbuf : 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 (a value for id is invalid or pbuf is null)

(b) R_HFDSL_CMD_VPOS

R_HFDSL_CMD_VPOS	
Synopsis	Acquiring the safe position
Header	r_hfdsl_rzt2_if.h r_hfdsl_rzt2_dat.h
Declaration	int32_t R_HFDSL_Control (const int32_t id, const r_hfdsl_cmd_t cmd, void *const pbuf);
Description	This function acquires the safe position by reading the safe position registers (VPOS4~VPOS0), and safe position CRC registers (VPOSCRC_H, VPOSCRC_L). If the safe channel 1 interface register access is disabled, this function returns access error.
Argument	id : Specifies the ID to be used. R_HFDSL0_ID : Specifies channel 0 R_HFDSL1_ID : Specifies channel 1 Other than above : Setting is not allowed cmd : Specifies R_HFDSL_CMD_VPOS pbuf : 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 (a value for id is invalid or pbuf is null) R_HFDSL_ERR_ACCESS: Abnormal termination (access to the safe channel 1 interface registers is disabled)

(c) R_HFDSL_CMD_VEL

R_HFDSL_CMD_VEL	
Synopsis	Acquiring the rotational velocity of the motor.
Header	r_hfdsl_rzt2_if.h r_hfdsl_rzt2_dat.h
Declaration	int32_t R_HFDSL_Control (const int32_t id, const r_hfdsl_cmd_t cmd, void *const pbuf);
Description	This function acquires the rotational velocity of the motor by reading the velocity registers (VEL2~VEL0).
Argument	id : Specifies the ID used R_HFDSL0_ID : Specifies channel 0 R_HFDSL1_ID : Specifies channel 1 Other than above : Setting is not allowed cmd : Specifies R_HFDSL_CMD_VEL pbuf : 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 (a value for id is invalid or pbuf is null)

(d) R_HFDSL_CMD_MSG

R_HFDSL_CMD_MSG	
Synopsis	Transmitting messages
Header	r_hfdsl_rzt2_if.h r_hfdsl_rzt2_dat.h
Declaration	int32_t R_HFDSL_Control (const int32_t id, const r_hfdsl_cmd_t cmd, void *const pbuf);
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. R_HFDSL0_ID : Specifies channel 0 R_HFDSL1_ID : Specifies channel 1 Other than above : Setting is not allowed cmd : Specifies R_HFDSL_CMD_MSG pbuf : 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 (a value for id is invalid or pbuf is null) R_HFDSL_ERR_ACCESS: Abnormal termination (the protocol initialization function described in Section 4.4.4(1) Protocol Initialization Functions, has not been executed)
Note	Calling this API function from within a callback function is prohibited. To proceed with a next transmission, execute this function following a call of the hfdsl_init_mrcv_callback function.

(e) R_HFDSL_CMD_RST

R_HFDSL_CMD_RST	
Synopsis	Protocol reset
Header	r_hfdsl_rzt2_if.h r_hfdsl_rzt2_dat.h
Declaration	int32_t R_HFDSL_Control (const int32_t id, const r_hfdsl_cmd_t cmd, void *const pbuf);
Description	This function resets the protocol. After this function is called. An HDSLn_INT interrupt is generated in response to the PRST bit in the EVENT_H being set to 1. To resume communications, call the functions described in Section 4.4.4(1), Protocol Initialization Commands.
Argument	id : Specifies the ID to be used R_HFDSL0_ID : Specifies channel 0 R_HFDSL1_ID : Specifies channel 1 Other than above : Setting is not allowed cmd : R_HFDSL_CMD_RST pbuf : Specify NULL
Return value	R_HFDSL_SUCCESS: Normal termination R_HFDSL_ERR_INVALID_ARG: Abnormal termination (a value for id or pbuf 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 HDSLn_FPR 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 pcb_nml of the argument r_hfdsl_info_t structure of the R_HFDSL_Open function. It is called when an HDSLn_FPR interrupt is generated. This interrupt shows that the fast position registers (POS4~POS0) have been updated. The fast position can be acquired by executing the function R_HFDSL_Control (R_HFDSL_CMD_POS) from within this function. This function is in the context of an interrupt handler. To secure responsiveness to interrupts, make sure that this function is returned immediately. The function name given above is only an example and can be freely set.
Argument	event : Source of the interrupt Holds the value POS_RDY_BIT. The value of this argument is only valid within this function.
Return value	None

4.5.2 hfdsl_int_err_callback

hfdsl_int_err_callback	
Synopsis	Indicating the generation of HDSLn_INT 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 pcb_err of the argument r_hfdsl_info_t structure of the R_HFDSL_Open function. It is called when an HDSLn_INT interrupt is generated in response to the SUM, POS, DTE or PRST bits in the EVENT_H register, or the MIN, ANS or QMLW bits in the EVENT_L register being set to 1. This function is in the context of an interrupt handler. To secure responsiveness to interrupts, make sure that this function is returned immediately. The function name given above is only an example and can be freely set.
Argument	event_err : Source of the HDSLn_INT interrupt Holds the value of the EVENT_H, EVENT_L registers. The value of this argument is only valid within this function.
Return value	None
Note	This function is not called when an HDSLn_INT is generated in response to the FREL bit in the EVENT_L register being set to 1.

4.5.3 hfdsl_int_safety_callback

hfdsl_int_safety_callback		
Synopsis	Indicating the generation of HDSLn_SP interrupt	
Header	r_hfdsl_rzt2_if.h	
Declaration	void hfdsl_int_safety_callback(uint8_t *psafety1);	
Description	This callback function is registered with the member variable pcb_safety of the argument r_hfdsl_info_t structure of the R_HFDSL_Open function. It is called when an HDSLn_SP interrupt is generated. This interrupt shows that the safety position registers have been updated. This function is in the context of an interrupt handler. To secure responsiveness to interrupts, make sure that this function is returned immediately. The function name given above is only an example and can be freely set.	
Argument	psafety1[]	: Safety status, safe position, and CRC If the safe channel 1 interface register access is enabled. the array pointed by psafety1 holds vertical channel data. vertical channel data contains safety status (SAFE_SUM) register data, safe position (VPOS4~VPOS0) register data, and safe position CRC (VPOSCRC_H, VPOSCRC_L) register data. If the safe channel 1 interface register access is disabled, psafety1 holds NULL pointer. The value of this argument is only valid within this function.
Return value	None	

4.5.4 hfdsl_int_mrcv_callback

hfdsl_int_mrcv_callback		
Synopsis	Indicating that the HDSLn_INT interrupt by the FREL bit in the EVENT_L register has occurred.	
Header	r_hfdsl_rzt2_if.h	
Declaration	void hfdsl_int_mrcv_callback(uint8_t* msg_data);	
Description	This callback function is registered with the R_HFDSL_Control (R_HFDSL_CMD_MSG) function. It is called when the HDSLn_INT interrupt by the FREL bit in the EVENT_L register occurs and data storage of the received message is completed. This function is in the context of an interrupt handler. To secure responsiveness to interrupts, make sure that this function is returned immediately. The function name given above is only an example and can be freely set.	
Argument	msg_data[]	: Message address and PC_BUFF register values (long messages) Two bytes of the message address (PC_ADD_H, PC_ADD_L) and the values of the PC_BUF0~PC_BUF7 registers (long messages) are stored. The fifth bit LOFF of the message address PC_ADD_H holds the message reception error flag. The value of this argument remains valid until the next HDSLn_INT interrupt caused by the FREL bit is generated.
Return value	None	

4.6 Interrupt Handler

4.6.1 hfdsl_int_isr_ch0

hfdsl_int_isr_ch0	
Synopsis	Interrupt handler for the HDSL0_INT
Header	-
Declaration	static void hfdsl_int_isr_ch0(void);
Description	An interrupt handler for the HDSL0_INT interrupt. If the source of an interrupt is the FREL bit of the EVENT_L register, function hfdsl_int_mrcv_callback is called as a callback function. If the source of an interrupt is other bits of the EVENT_H register and the EVENT_L register, function hfdsl_int_err_callback is called as a callback function.
Argument	None
Return value	None

4.6.2 hfdsl_int_isr_ch1

hfdsl_int_isr_ch1	
Synopsis	Interrrupt handler for the HDSL1_INT
Header	-
Declaration	static void hfdsl_int_isr_ch1(void);
Description	An interrupt handler for the HDSL1_INT interrupt. If the source of an interrupt is the FREL bit of the EVENT_L register, function hfdsl_int_mrcv_callback is called as a callback function. If the source of an interrupt is other bits of the EVENT_H register and the EVENT_L register, function hfdsl_int_err_callback is called as a callback function.
Argument	None
Return value	None

4.6.3 hfdsl_fpr_isr_ch0

hfdsl_fpr_isr_ch0	
Synopsis	Interrupt handler for the HDSL0_FPR
Header	-
Declaration	static void hfdsl_fpr_isr_ch0(void);
Description	An interrupt handler for the HDSL0_FPR interrupt. If the interrupt is generated, function hfdsl_int_nml_callback is called as a callback function.
Argument	None
Return value	None

4.6.4 hfdsl_fpr_isr_ch1

hfdsl_fpr_isr_ch1	
Synopsis	Interrupt handler for the HDSL1_FPR
Header	-
Declaration	static void hfdsl_fpr_isr_ch1(void);
Description	An interrupt handler for the HDSL1_FPR interrupt. If the interrupt is generated, function hfdsl_int_nml_callback is called as a callback function.
Argument	None
Return value	None

4.6.5 hfdsl_sp_isr_ch0

hfdsl_sp_isr_ch0	
Synopsis	Interrupt handler for the HDSL0_FPR
Header	-
Declaration	static void hfdsl_sp_isr_ch0(void);
Description	An interrupt handler for the HDSL0_SP interrupt. If the interrupt is generated, function hfdsl_int_safety_callback is called as a callback function.
Argument	None
Return value	None

4.6.6 hfdsl_sp_isr_ch1

hfdsl_sp_isr_ch1	
Synopsis	Interrupt handler for the HDSL1_FPR
Header	-
Declaration	static void hfdsl_sp_isr_ch1(void);
Description	An interrupt handler for the HDSL1_SP interrupt. If the interrupt is generated, function hfdsl_int_safety_callback is called as a callback function.
Argument	None
Return value	None

4.6.7 hfdsl_err_isr

hfdsl_err_isr	
Synopsis	Interrupt handler for the PERI_ERR0
Header	-
Declaration	static void hfdsl_err_isr(void);
Description	An interrupt handler for the PERI_ERR0 interrupt. If the interrupt is generated, this function reads error events from PERIERR_STAT3 register and clear interrupt.
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
HDSL0_INT	263	This interrupt is generated when the value of any bit in the ch0 EVENT_L, EVENT_H registers is updated to 1.
HDSL1_INT	265	This interrupt is generated when the value of any bit in the ch1 EVENT_L, EVENT_H registers is updated to 1.
HDSL0_FPR	273	This interrupt is generated when the fast position value of the ch0 is ready to read.
HDSL1_FPR	274	This interrupt is generated when the fast position value of the ch1 is ready to read.
HDSL0_SP	275	This interrupt is generated when the safety position value of the ch0 is ready to read.
HDSL1_SP	276	This interrupt is generated when the safety position value of the ch1 is ready to read.
PERI_ERR0	388	This interrupt is generated when the value of any bit indicating HDLS ch0 or ch1 error in the PERIERR_STAT3 register is updated to 1.

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_SYNC_CTRL	3	Setting of the SYNC_CTRL register
R_HFDSL_ACC_ERR	31	Setting of the ACC_ERR register
R_HFDSL_MASK_H	4Bh	Setting of the MASK_H register *1
R_HFDSL_MASK_L	36h	Setting of the MASK_L register *1

Note 1. To change R_HFDSL_MASK_H and R_HFDSL_MASK_L, change processing of the hfdsl_int_isr_ch0 function, hfdsl_int_isr_ch1 function in accord with the settings in R_HFDSL_MASK_H and R_HFDSL_MASK_L.

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

Table 4.5 Interface Mode Codes for the Safe Interface

Constant Name	Setting	Description
R_HFDSL_INTERNAL_BUS_MODE	0	Internal bus mode
R_HFDSL_SPI_MODE	1	SPI mode

4.9 Fixed-Width Integers

Table 4.6 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 4.6 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 Enumerations

The main structures, unions, and enumerations are listed below.

4.10.1 Structures

(1) r_hfdsl_info_t

Information on initialization of the HFDSL driver

typedef struct

{

uint8_t	safe1_if_mode;	Select the interface mode for safe channel 1 interface. (0: Internal bus mode, 1: SPI mode) *1
uint8_t	safe2_if_mode;	Select the interface mode for safe channel 2 interface. *2
r_hfdsl_int_nml_cb_t	pcb_nml;	Pointer to the callback function to be called when an HDSLn_FPR interrupt is generated. For details, see Section 4.5.1, hfdsl_int_nml_callback. *3, *4
r_hfdsl_int_err_cb_t	pcb_err;	Pointer to the callback function to be called when an HDSLn_INT interrupt is generated. For details, see Section 4.5.2, hfdsl_int_err_callback. *3
r_hfdsl_int_safety_cb_t	pcb_safety;	Pointer to the callback function to be called when an HDSLn_SP interrupt is generated. For details, see Section 4.5.3 hfdsl_int_safety_callback. *3

} r_hfdsl_info_t

Note
1. If the SPI mode is selected for safe channel 1 interface, access to the safe channel 1 interface registers from sample program is disabled. SPI mode is used to access to the safe channel 1 registers by external CPU via SPI interface.
2. Safe channel 2 interface is always SPI mode for the RZ/T2L. Setting value of this parameter is not used by the RZ/T2L.
3. This function is not called if NULL is specified.
4. This function is not called when an HDSLn_INT interrupt is generated in response to the FREL bit in the EVENT_L 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
{
    uint8_t              *pdata;        Pointer to the array which holds message data for
                                        transmission.
                                        Set the pointer to the array which holds message data
                                        for transmission.
    r_hfdsl_msg_cb_t     pcb_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 Enumerations

Not used

4.11 Description of the Sample Program

4.11.1 Outline of Operations

This sample program supports the encoder (EDM35-2KF0A020A 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 the encoder among the resources)
   D) Vertical channel data
2) Runs in SYNC mode.
3) This sample program ends by a protocol reset.

(1) System Block Diagram

Figure 4.1 shows a block diagram of the system.

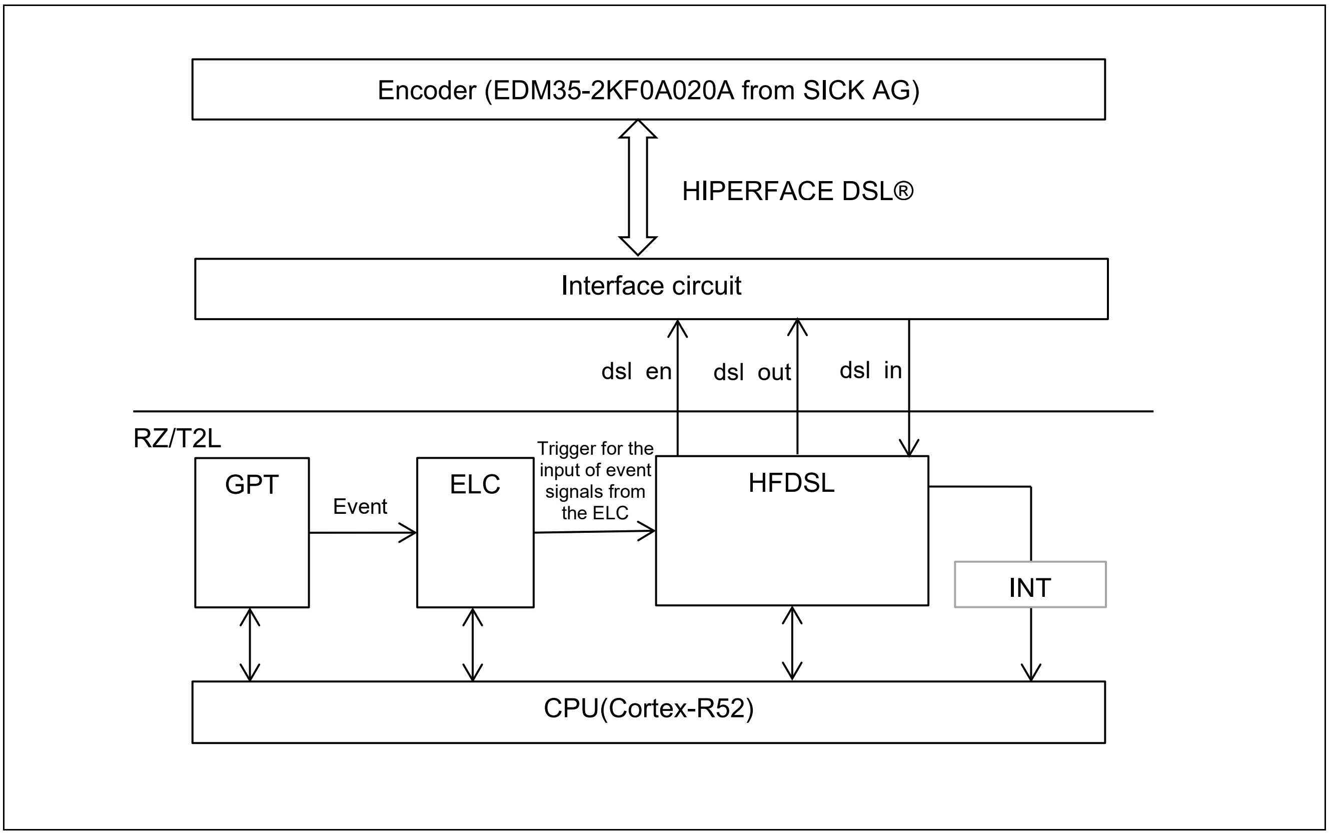

Figure 4.1 System Block Diagram

(2) Software Configuration

Figure 4.2 is a block diagram of the software.

The HFDSL driver has six sections: the opening processing part configured of function R_HFDSL_Open, the closing processing 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 control section of the sample program controls the HFDSL driver, acquires the positional value, and sends messages and the results indication section (callback) indicates the result of data reception.

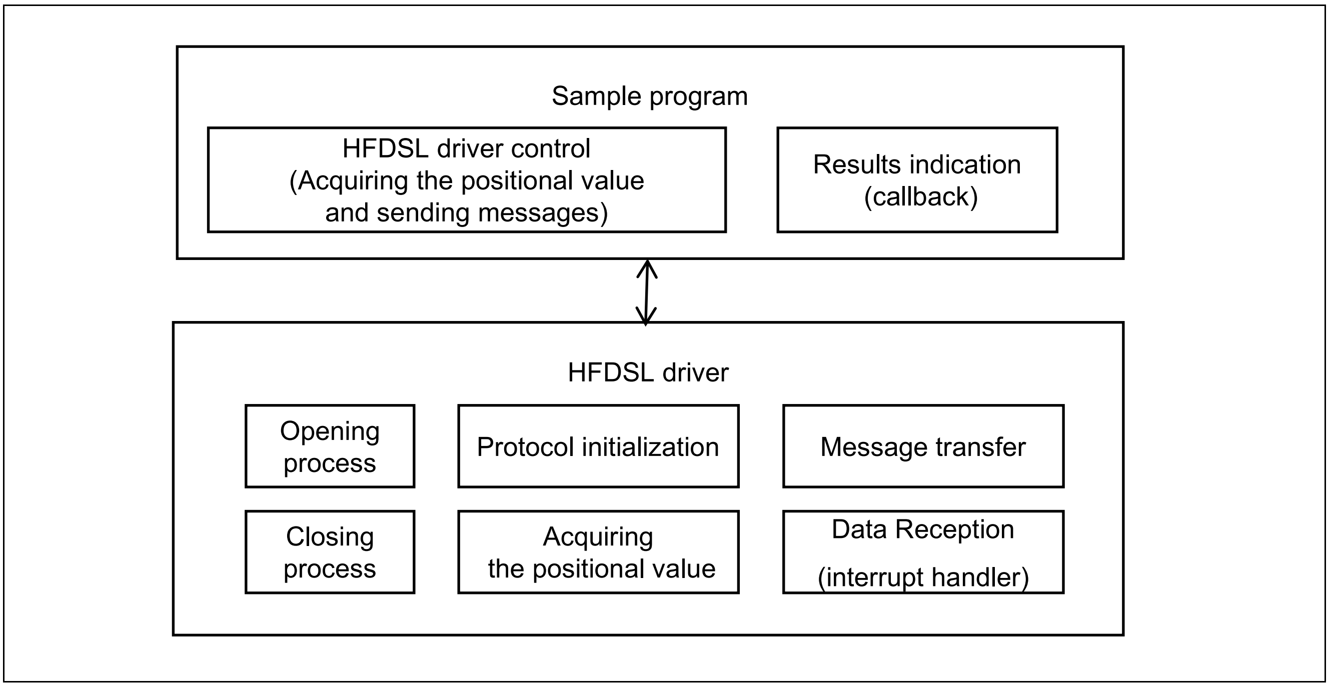

Figure 4.2 Software Configuration

4.11.2 Variables for the Sample Program

Table 4.7 lists the main static variables.

Table 4.7 Main 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)
bool	prst_found	Protocol reset warning detection flag. (true: Protocol reset warning is detected false: Protocol reset warning is not detected)
uint32_t	err_info	Holds the HDSLn_INT 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.
bool	vpos_valid	Holds result of the safe position is valid or not.
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.
uint8_t	lmsg_recv[LMSG_RECV_SIZE]	Holds received data in long messages.
bool	safety1_valid	Holds result of the vertical channel data is valid or not.
uint8_t	safety1[SAFETY_CNT_MAX]	Holds the received data including vertical channel data.

4.11.3 Constants for the Sample Program

Table 4.8 lists the main constants for the sample program.

Table 4.8 Main Constants

Constant Name	Setting	Description
ENC_ID	00000153h	Encoder ID of EDM35-2KF0A020A *1 *2
RES_BIT	0	Position of the least significant bit of the positional information in the POS4~0 registers *1
RES_MASK	000FFFFFh	Masking of the positional information in the POS3~0 registers *1
RES_MASK_H	00000000h	Masking of the positional information in the POS4 register *1
ROT_BIT	20	Position of the least significant bit of the rotational information in the POS4~0 registers *1
ROT_MASK	00000FFFh	Masking of the number of rotations in the POS3~0 registers *1
ROT_MASK_H	00000000h	Masking of the number of rotations in the POS4 register *1
LMSG_RECV_SIZE	10	Maximum size of received data in long messages
SAFETY_CNT_MAX	8	Vertical channel data size
TIMEOUT_UNIT	1000	Setting of timeout unit (1 ms)
TIMEOUT_COUNT	1000	Setting of timeout (1 ms x 1000)
INIT_RETRY_COUNT	10	Retry times of initialization error

Note 1. To run the sample program with an encoder other than an EDM35-2KF0A020A, change the settings to suit the specifications of the connected encoder.

2. Refer to the *HIPERFACE DSL® Implementation Manual* for the details. The manual can be obtained by contacting SICK AG.

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

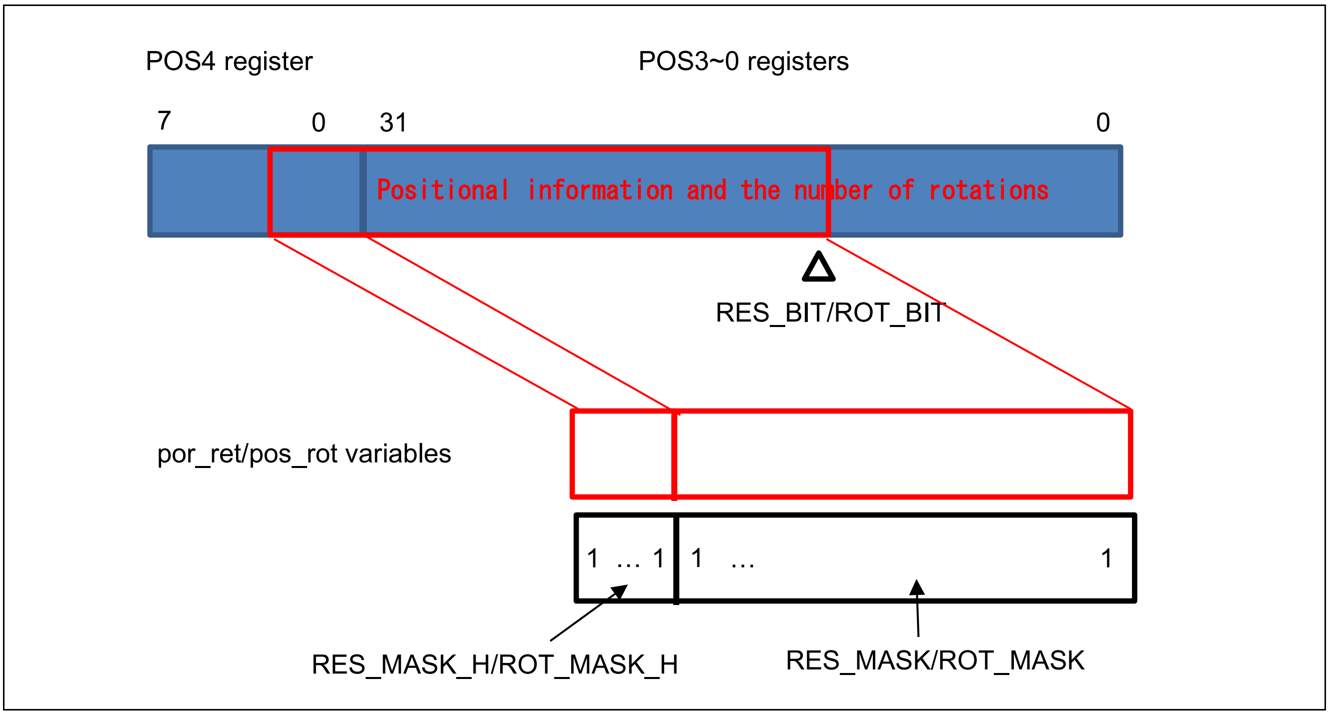

Figure 4.3 Mechanism for Storing the Positional and Rotational Information

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

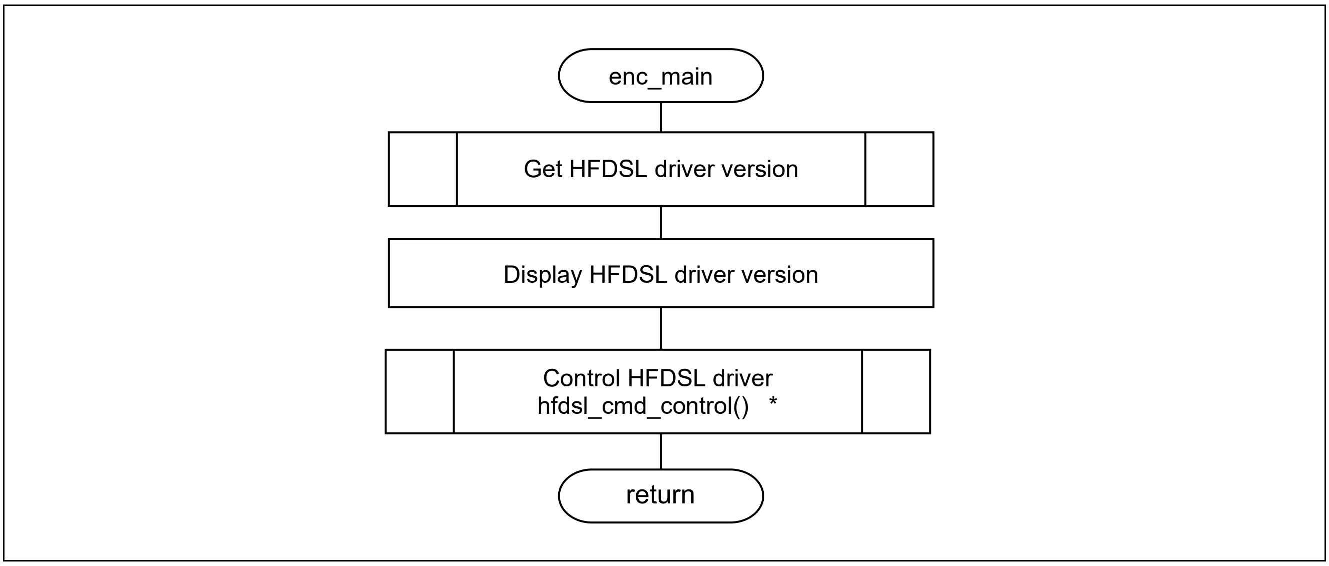

Figure 4.4 Flowchart of the enc_main Function

(2) Flowchart of hfdsl_cmd_control

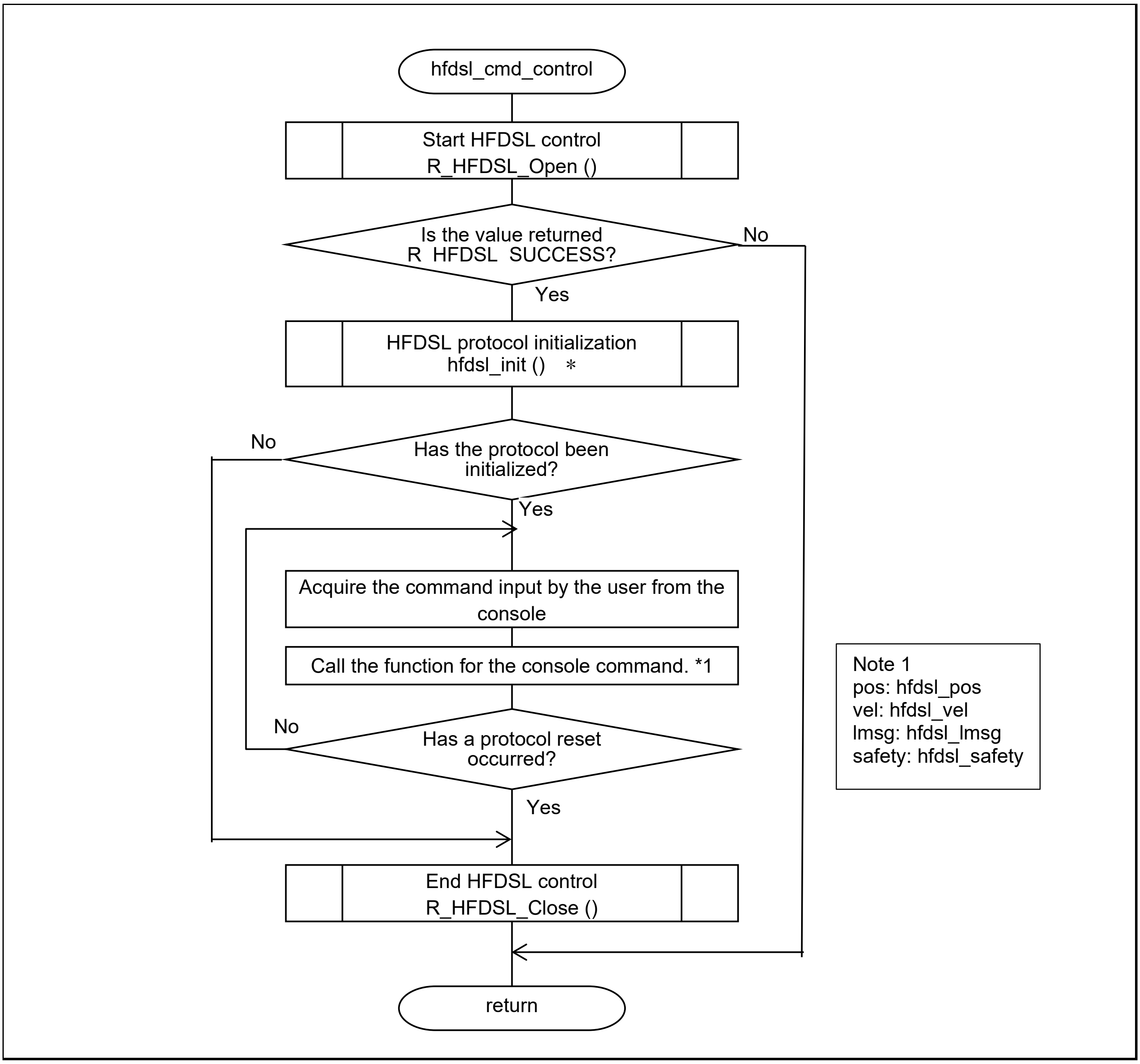

Figure 4.5 Flowchart hfdsl_cmd_control

(3) Flowchart of hfdsl_init

This function initializes the protocol.

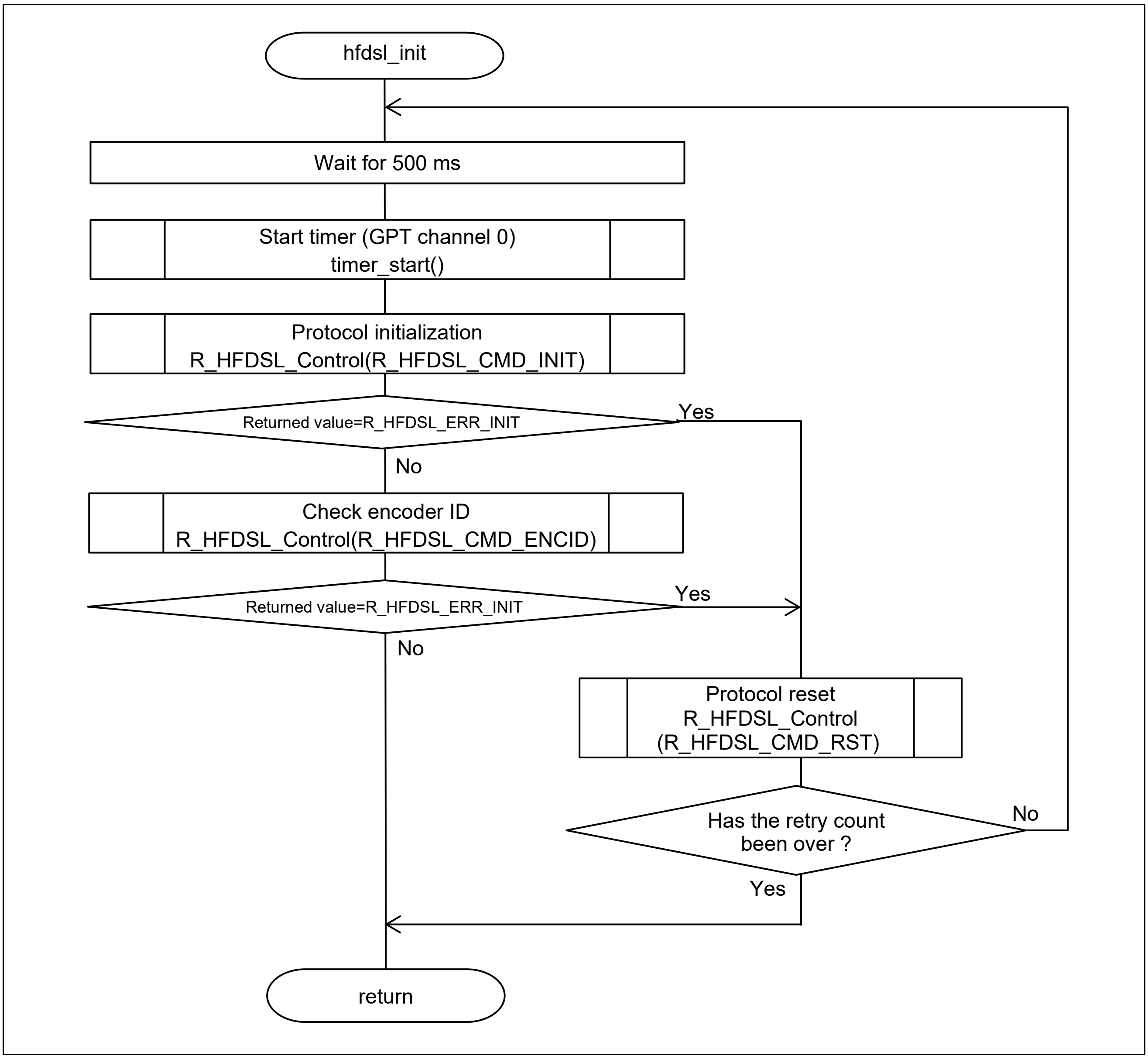

Figure 4.6 Flowchart of the hfdsl_init

(4) Flowchart of hfdsl_pos, hfdsl_vel, hfdsl_safety

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

Table 4.9 Functions Corresponding to the Console Commands “pos”, “vel”, “safety”

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
safety	hfdsl_safety	safety1

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

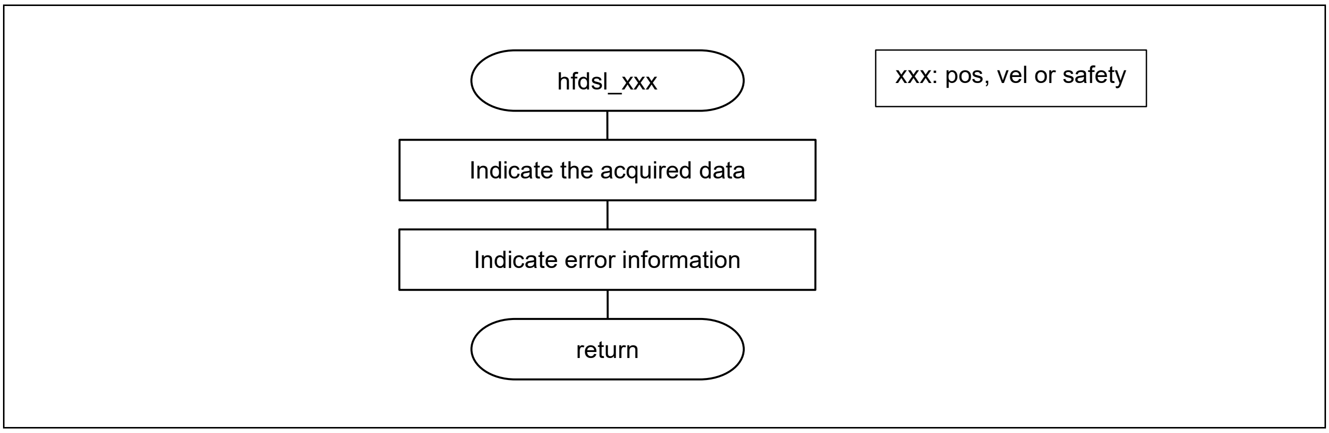

Figure 4.7 Flowchart of the hfdsl_pos, hfdsl_vel, hfdsl_safety

(5) Flowchart of hfdsl_lmsg

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

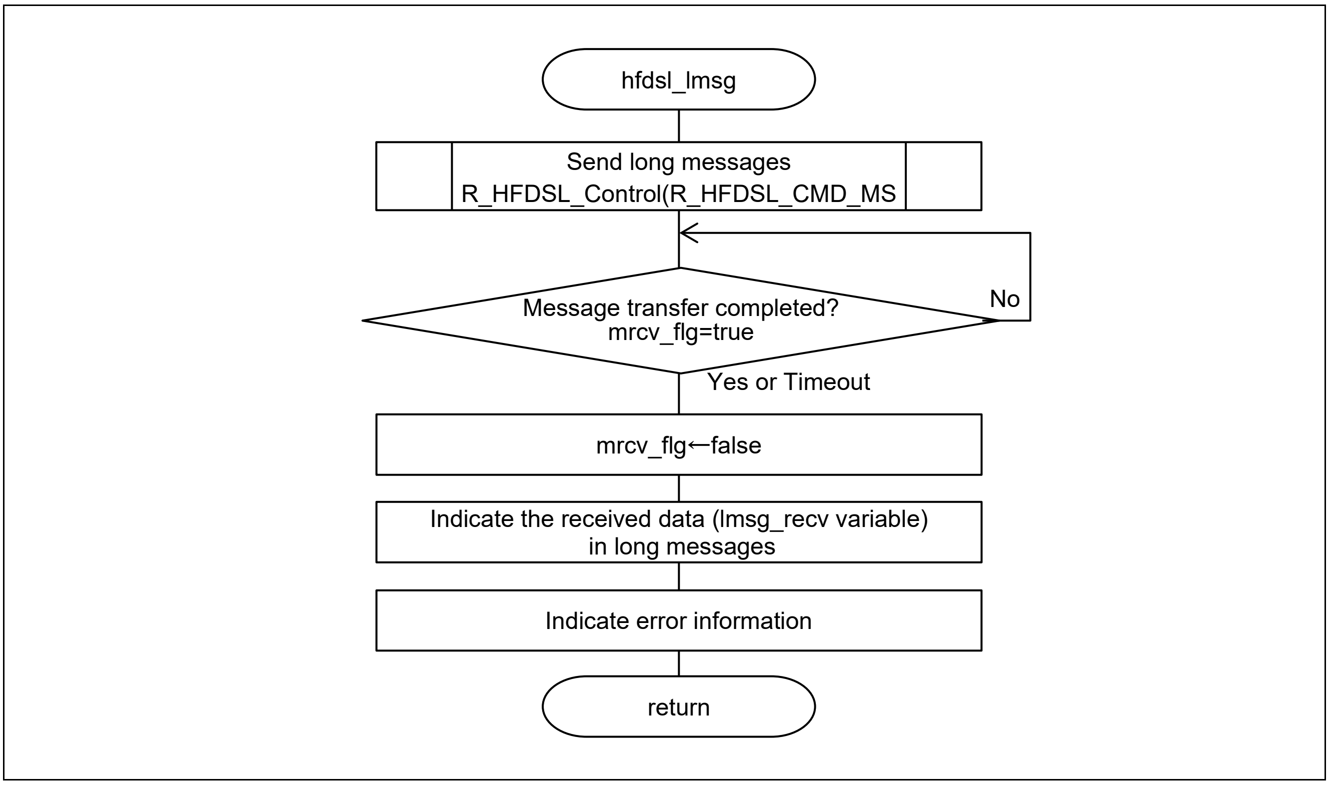
hfdsl_lmsg
Send long messages
R_HFDSL_Control(R_HFDSL_CMD_MS
Message transfer completed?
mrcv_flg=true
No
Yes or Timeout
mrcv_flg←false
Indicate the received data (lmsg_recv variable)
in long messages
Indicate error information
return

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 HDSLn_FPR interrupt.

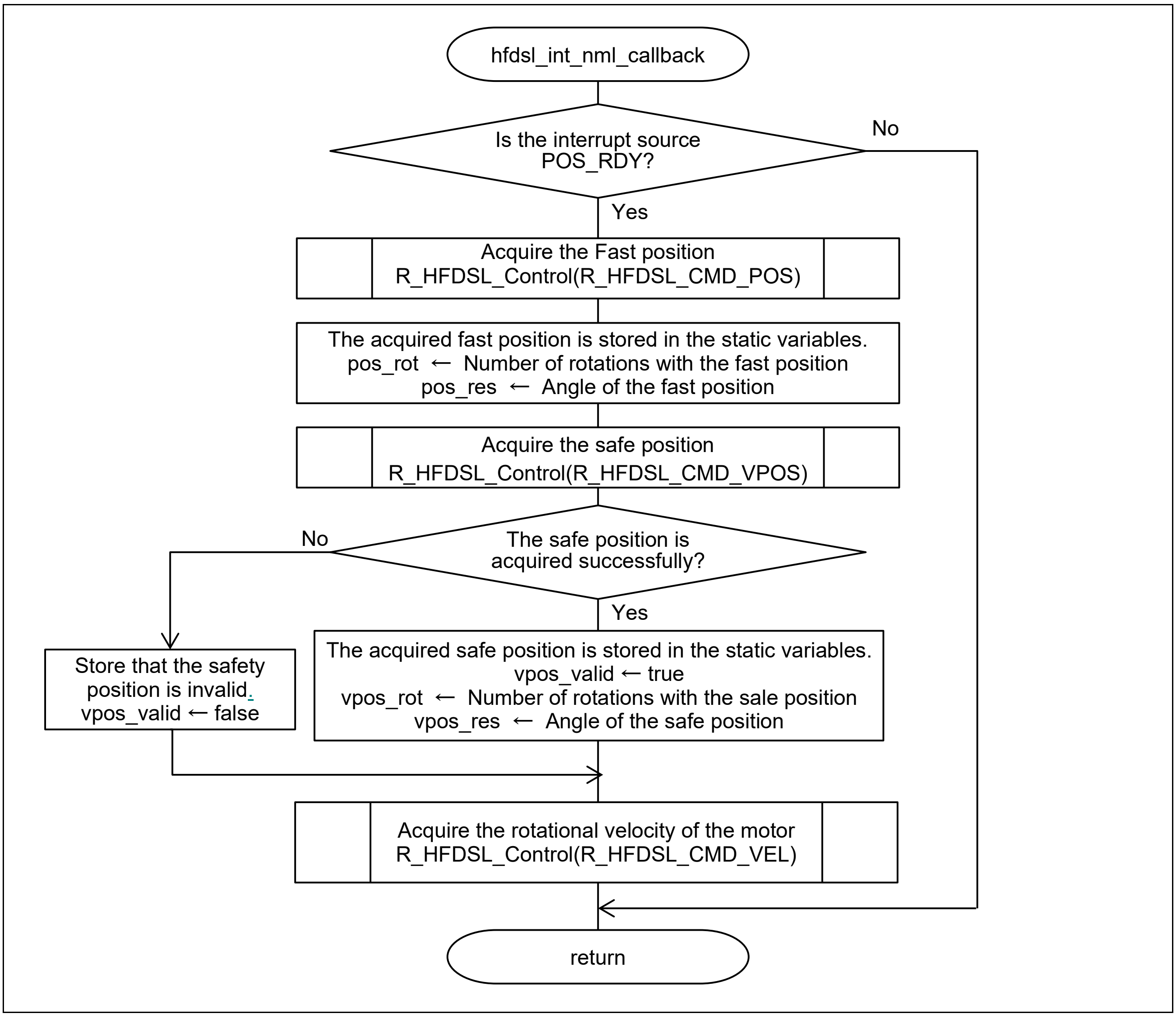
hfdsl_int_nml_callback
Is the interrupt source POS_RDY?
No
Yes
Acquire the Fast position
R_HFDSL_Control(R_HFDSL_CMD_POS)
The acquired fast position is stored in the static variables.
pos_rot ← Number of rotations with the fast position
pos_res ← Angle of the fast position
Acquire the safe position
R_HFDSL_Control(R_HFDSL_CMD_VPOS)
No
The safe position is acquired successfully?
Yes
Store that the safety position is invalid.
vpos_valid ← false
The acquired safe position is stored in the static variables.
vpos_valid ← true
vpos_rot ← Number of rotations with the sale position
vpos_res ← Angle of the safe position
Acquire the rotational velocity of the motor
R_HFDSL_Control(R_HFDSL_CMD_VEL)
return

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 HDSLn_INT interrupt.

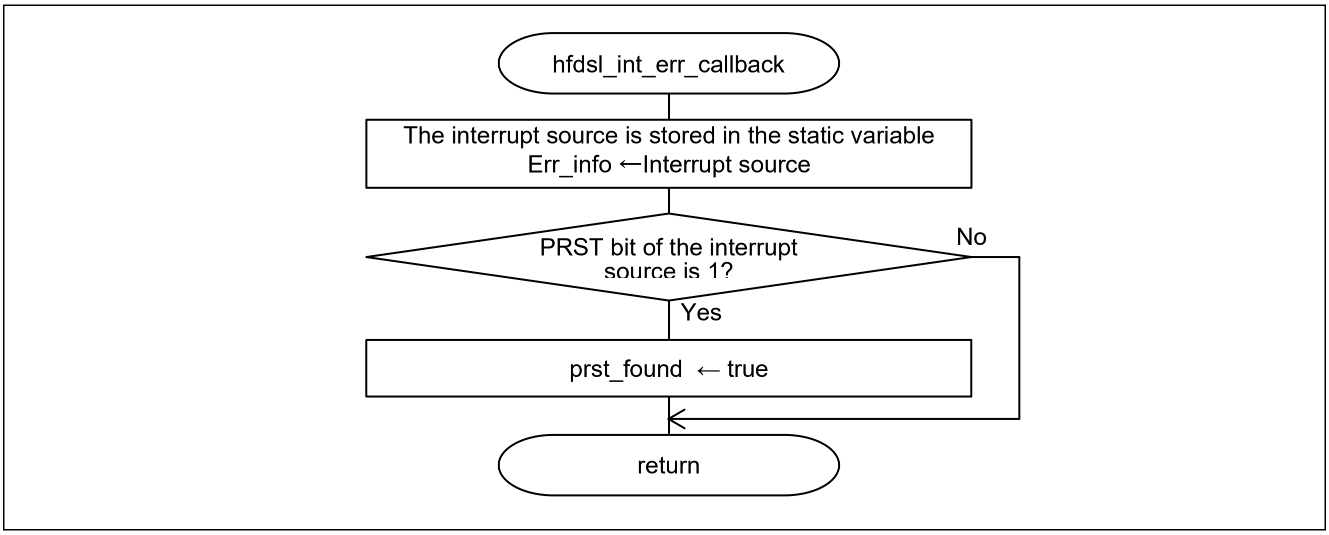
hfdsl_int_err_callback
The interrupt source is stored in the static variable
Err_info ←Interrupt source
PRST bit of the interrupt source is 1?
No
Yes
prst_found ← true
return

Figure 4.10 Flowchart of the hfdsl_int_err_callback

(8) Flowchart of hfdsl_int_safety_callback

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

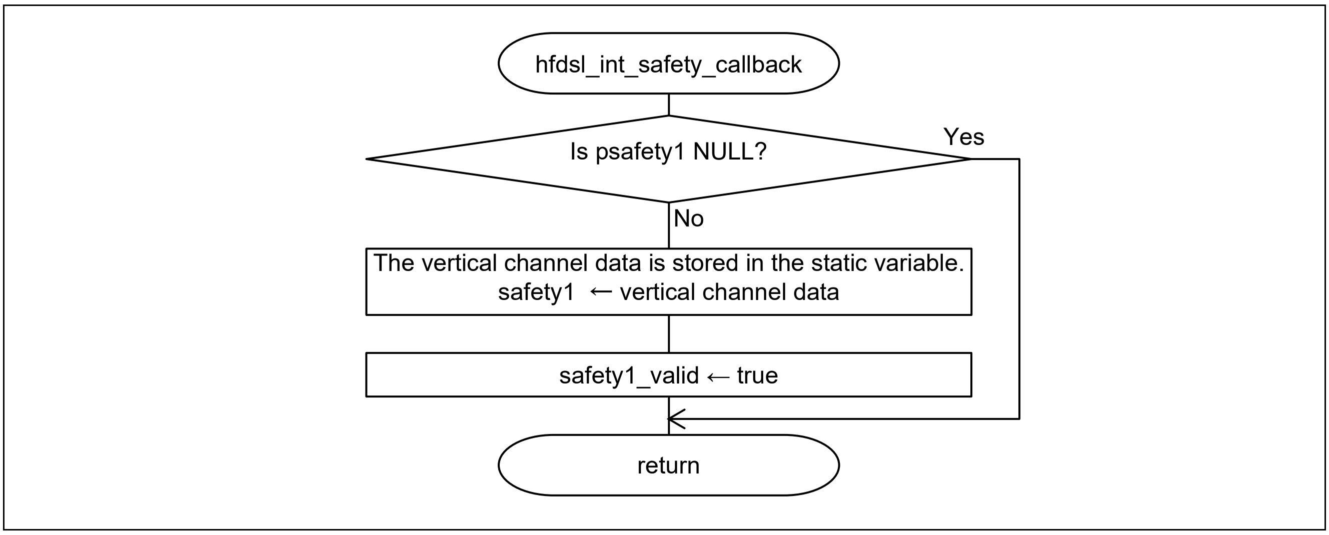
hfdsl_int_safety_callback
Is psafety1 NULL?
Yes
No
The vertical channel data is stored in the static variable.
safety1 ← vertical channel data
safety1_valid ← true
return

Figure 4.11 Flowchart of the hfdsl_int_safety_callback

(9) Flowchart of hfdsl_int_mrcv_callback

This callback function is called when the HDSLn_INT interrupt by the EVENT_L register FREL bit occurs and data storage of the received message is completed.

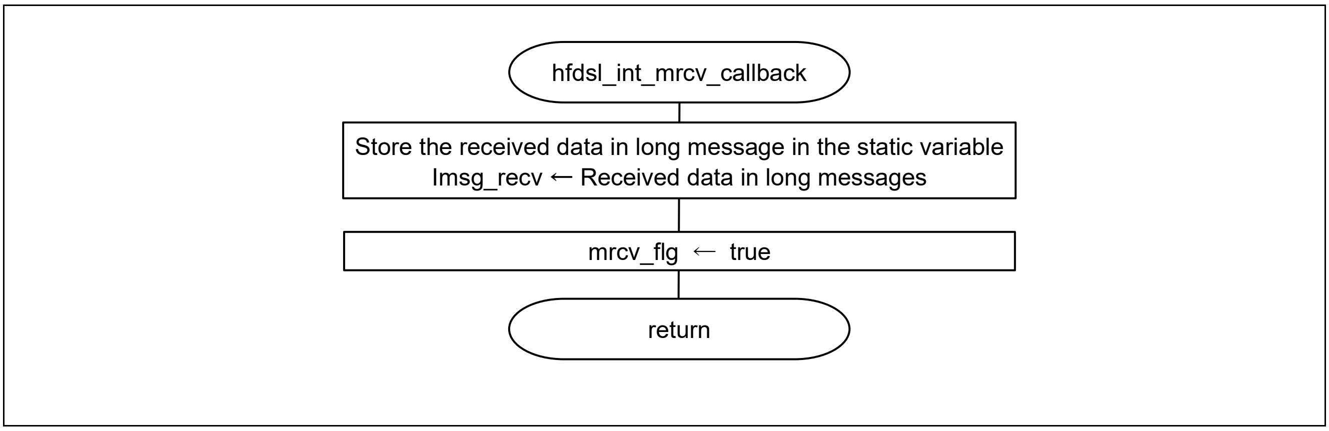
hfdsl_int_mrcv_callback
Store the received data in long message in the static variable
lmsg_recv ← Received data in long messages
mrcv_flg ← true
return

Figure 4.12 Flowchart of the hfdsl_int_mrcv_callback

4.11.5 Operation Sequence

(1) Startup Sequence

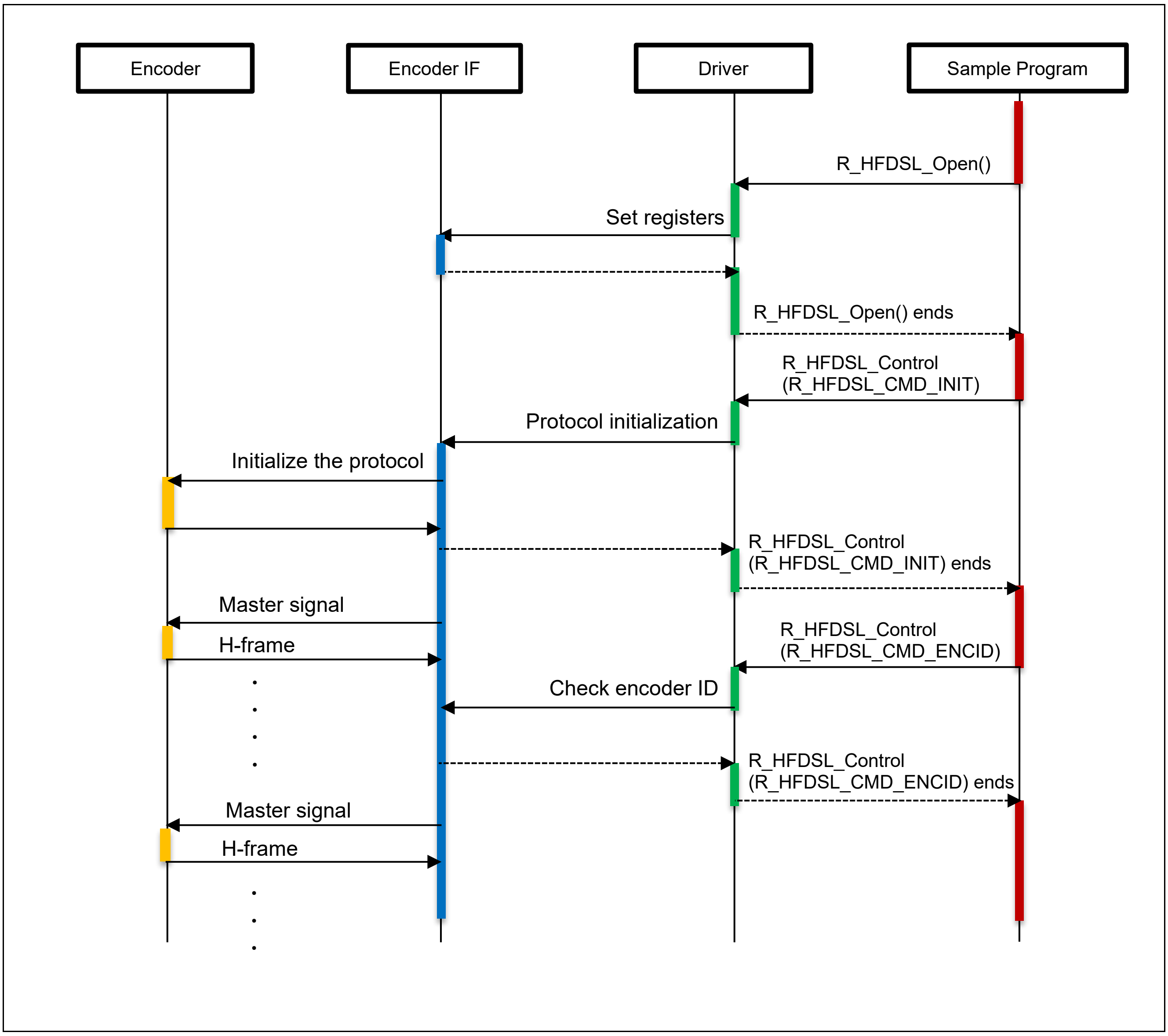

Figure 4.13 Startup Sequence Diagram

(2) Sequence for Acquiring the Fast Position in SYNC Mode

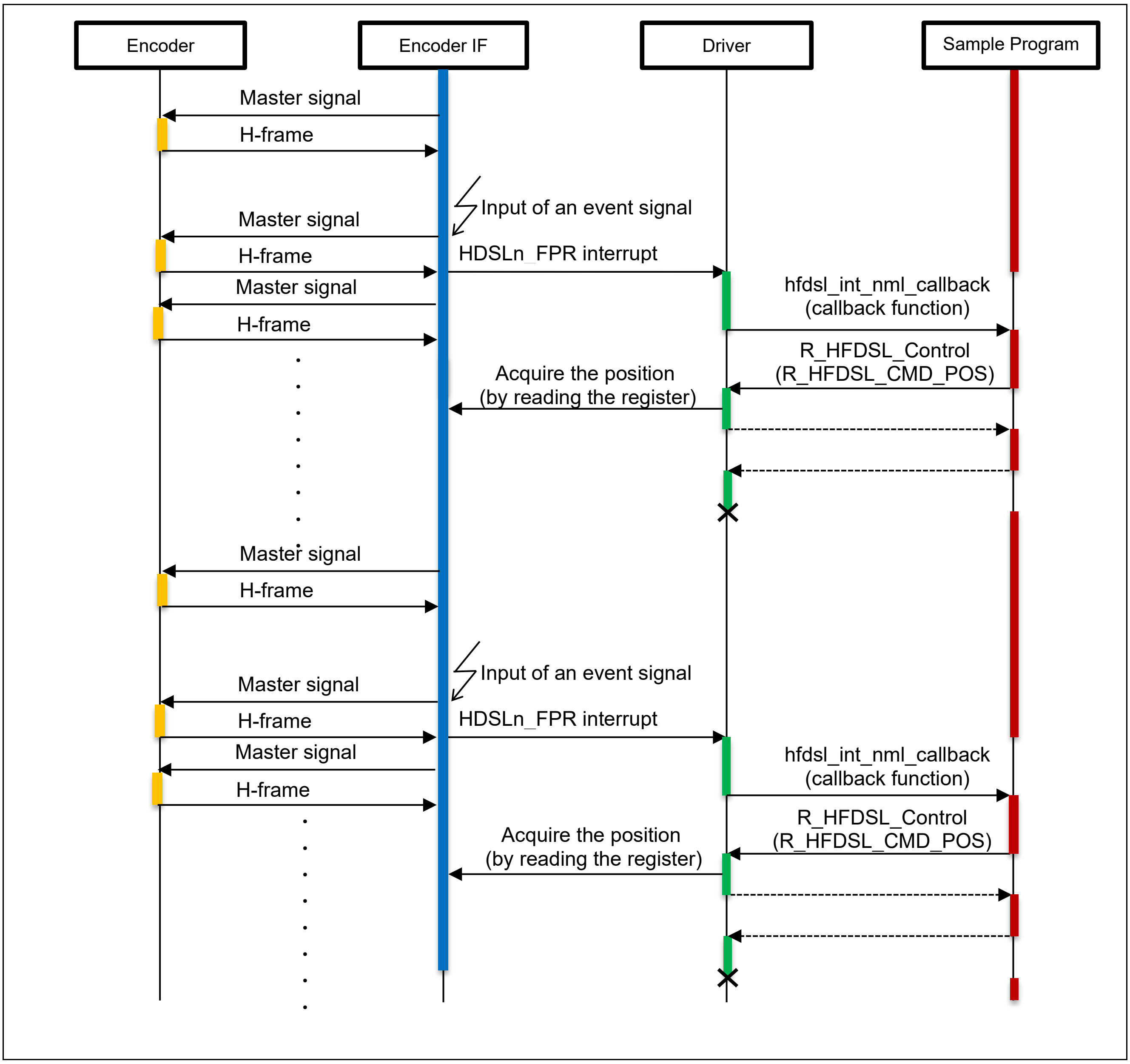

Figure 4.14 Sequence for Acquiring the Fast Position in SYNC Mode

(3) Sequence for Acquiring the Vertical Channel Data

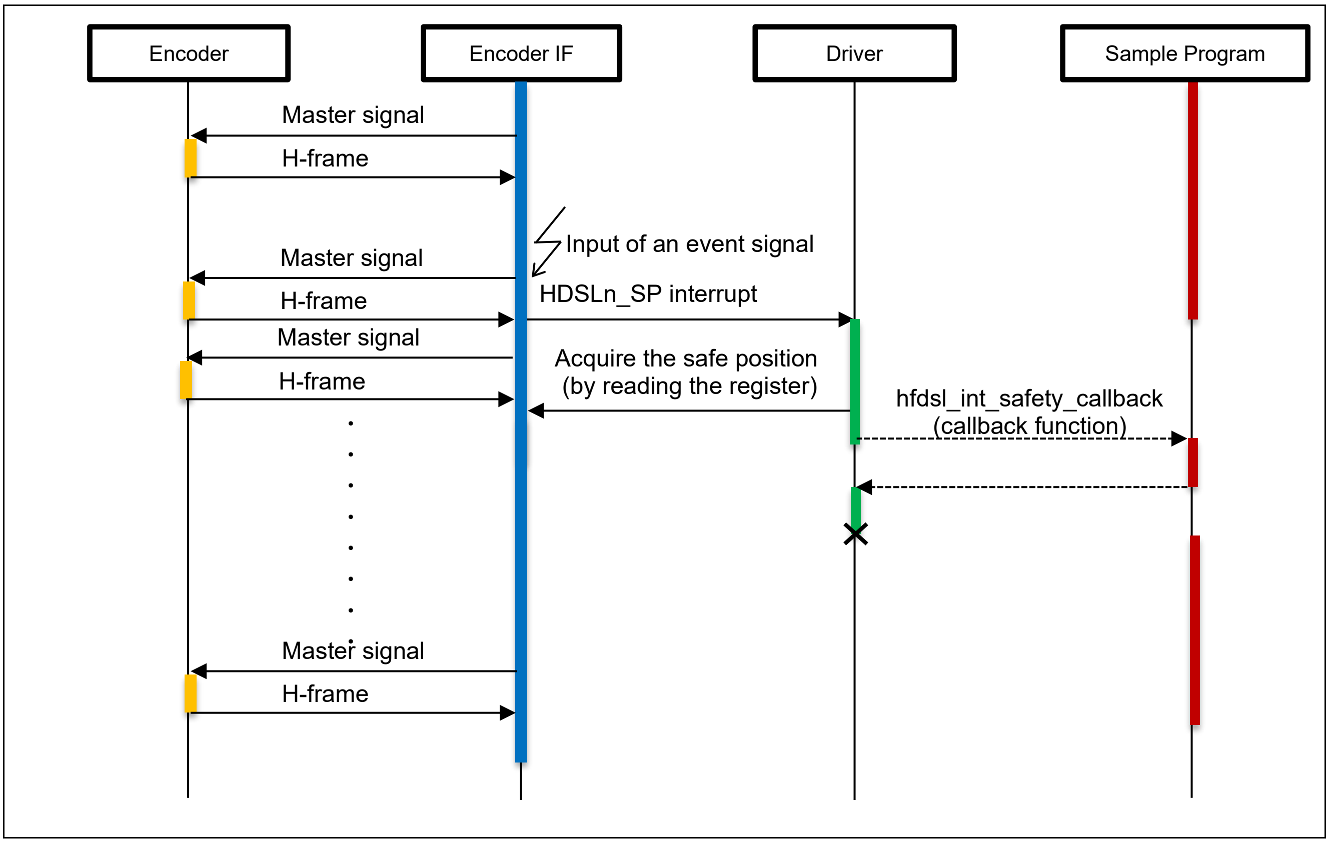

Figure 4.15 Sequence for Acquiring the Vertical Channel Data

(4) Message Transfer Sequence

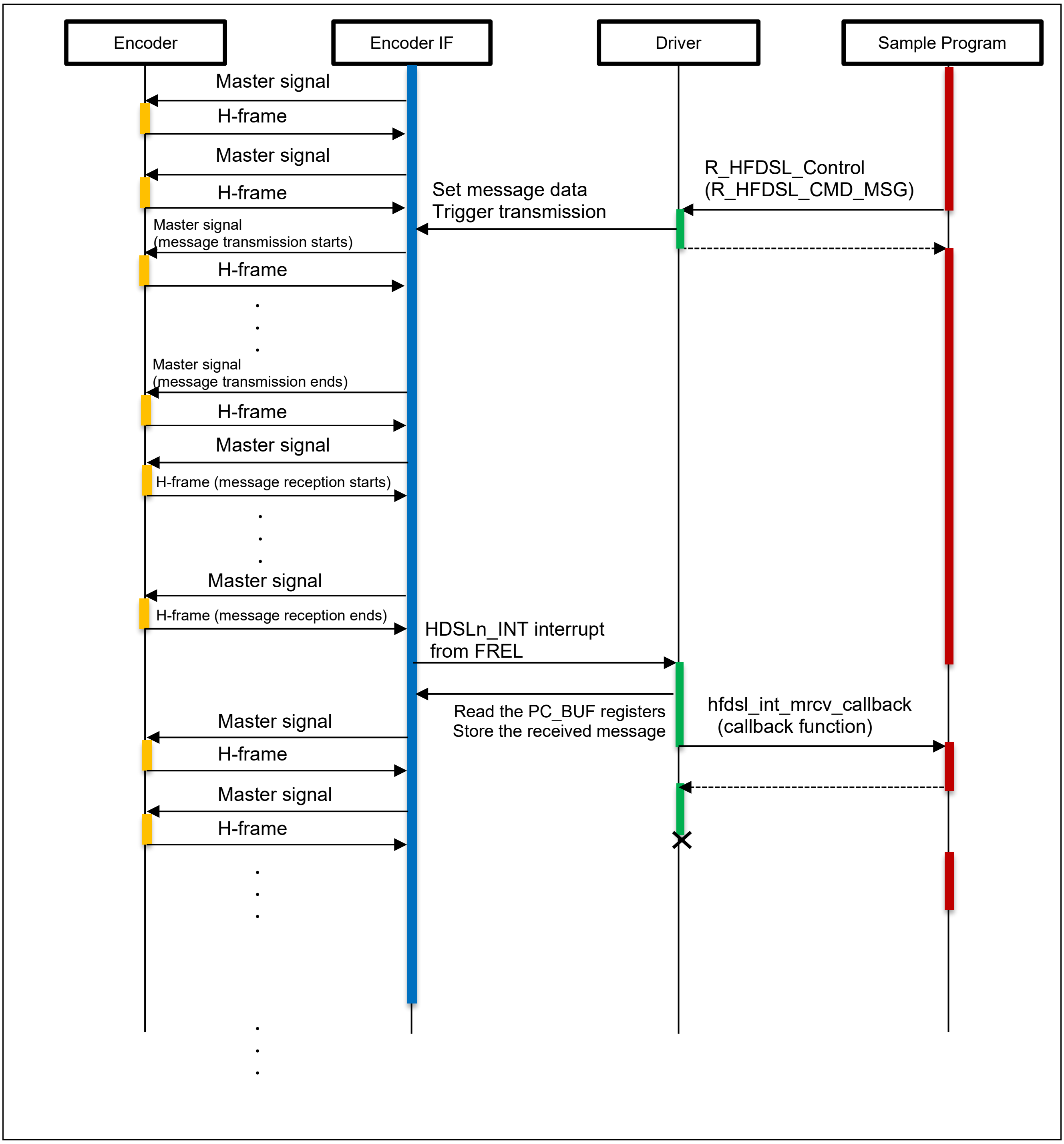

Figure 4.16 Message Transfer Sequence

(5) Stop Sequence

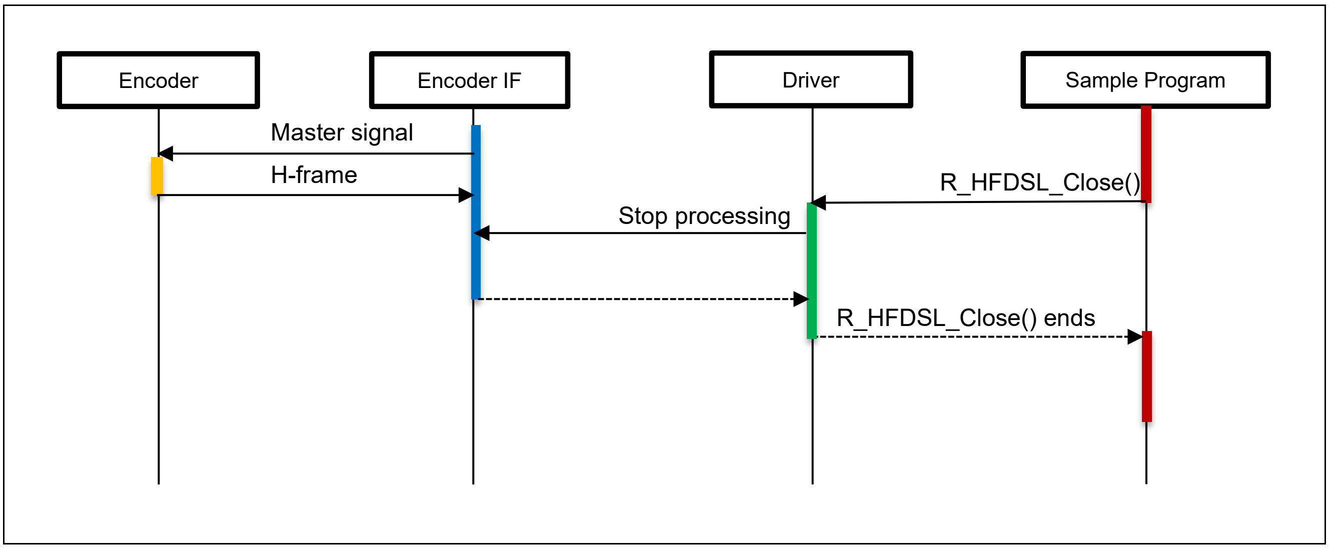

Figure 4.17 Stop Sequence

4.11.6 Console Commands

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

Table 4.10 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.
safety	Displays received data of vertical channel.

(1) run

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

```
HFDSL Safety sample program start
R_HFDSL_GetVersion = 1.0


hfdsl >
```

(2) pos command

Fast position: The result of R_HFDSL_CMD_POS in the hfdsl_int_nml_callback function is displayed.

Safe position: The result of R_HFDSL_CMD_VPOS in the hfdsl_int_nml_callback function is displayed. *

Error information: The result of the hfdsl_int_err_callback function is displayed.

```
hfdsl >pos
Fast position
   Rotations    : 0x000002E1
   Angle        : 0x0002D564
Safe position
   Rotations    : 0x000002E1
   Angle        : 0x0002D564
Error information
   EVENT_ERR    : 0x00000000
```

Note: If the SPI mode is selected for safe channel 1 interface mode by the argument pinfo of the R_HFDSL_Open function, access to the safe channel 1 interface registers is disabled. Values for the safe position are shown as “–".

(3) vel command

Motor rotation speed: The result of R_HFDSL_CMD_VEL in the hfdsl_int_nml_callback function is displayed.

Error information: The result of the hfdsl_int_err_callback function is displayed.

```
hfdsl >vel
Motor rotation speed
   Speed       : 0x00000026
Error information
   EVENT_ERR   : 0x00000000
```

(4) lmsg command

Message address: The message address of the long message is displayed.

Motor rotation speed: The result of the hfdsl_int_mrcv_callback function is displayed.

Error information: The result of the hfdsl_int_err_callback function is displayed.

```
hfdsl >lmsg
Message address
   PC_ADD_H : 0x54
   PC_ADD_L : 0x80
Received data
   PCBUF[0] : 0x00
   PCBUF[1] : 0x02
Error information
   EVENT_ERR   : 0x00000000
```

(5) safety command

SAFETY POSITION 1 data: The result of the hfdsl_int_safety_callback function is displayed. *

“data” are register data, “Rotations” and “Angle” are the values after conversion.

SAFETY POSITION 2 data: Safety position 2 is not accessible from this sample program, Values are displayed as “—“.

```
hfdsl >safety
SAFETY POSITION 1 data
   Rotations   : 0x000002E1
   Angle       : 0x000F055D
   data : 0x05 0x00 0x2E 0x1F 0x05 0x5D 0x79 0x7F
SAFETY POSITION 2 data
   Rotations   : --
   Angle       : --
   data : -- -- -- -- -- -- -- --
```

Note: If the SPI mode is selected for safe channel 1 interface mode by the argument pinfo of the R_HFDSL_Open function, access to the safe channel 1 interface registers is disabled. Values for the SAFETY POSITION 1 are shown as “–", too.

5. Sample Code

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

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	May 31.23	-	First Edition issued.
2.00	May 24.24	5 20	Update description of the board name. Remove description about location of integer type definition.

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

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/.