RZ/T2M グループ

HIPERFACE DSL サンプルプログラム

要旨

本アプリケーションノートでは、RZ/T2M の Encoder I/F Configuration Library（以下 EC-Lib）を使用して、HIPERFACE DSL® 通信プロトコル仕様に準拠したエンコーダから、情報を取得・表示するサンプルプログラムについて説明します。

プログラムの特徴を以下に示します。

・HIPERFACE DSL® 通信プロトコル仕様に準拠したエンコーダ (EFM50-0KF0A023A) から角度情報等を取得

動作確認デバイス

RZ/T2M

目次

1. 仕様 4

2. 動作環境 5

3. 周辺機能説明 6
3.1 使用端子一覧 6

4. ソフトウェア説明 7
4.1 HFDSL ドライバ機能 7
4.2 ファイル構成 7
4.3 関数一覧 7
4.4 API 関数仕様 8
4.4.1 R_HFDSL_Open 8
4.4.2 R_HFDSL_Close 8
4.4.3 R_HFDSL_GetVersion 9
4.4.4 R_HFDSL_CheckInitSeq 9
4.4.5 R_HFDSL_Control 9
4.5 ユーザー定義関数仕様 16
4.5.1 hfdsl_int_nml_callback 16
4.5.2 hfdsl_int_err_callback 16
4.5.3 hfdsl_int_raw_callback 17
4.5.4 hfdsl_int_mrcv_callback 17
4.5.5 hfdsl_int_init_callback 17
4.6 割り込みハンドラ 18
4.6.1 hfdsl_int_nml_isr_ch0 18
4.6.2 hfdsl_int_nml_isr_ch1 18
4.6.3 hfdsl_int_err_isr_ch0 18
4.6.4 hfdsl_int_err_isr_ch1 18
4.6.5 hfdsl_int_tovr_isr_ch0 19
4.6.6 hfdsl_int_tovr_isr_ch1 19
4.7 使用割り込み一覧 19
4.8 定数/エラーコード一覧 20
4.9 固定幅整数一覧 21
4.10 構造体/共用体/列挙型一覧 22
4.10.1 構造体 22
4.10.2 共用体 23
4.10.3 列挙型 23
4.11 サンプルプログラムの説明 24
4.11.1 動作概要 24
4.11.2 サンプルプログラムの変数一覧 26
4.11.3 サンプルプログラムの定数一覧 27
4.11.4 メイン処理のフローチャート 28
4.11.5 動作シーケンス 35
4.11.6 コンソールコマンド 42

5. サンプルコード 43

改訂記録 44

1. 仕様

表 1-1 に使用する周辺機能と用途を、図 1-1 にサンプルコード実行時の動作環境を示します。

表 1-1　使用する周辺機能と用途

周辺機能	用途
HIPERFACE DSL コントローラ(HFDSL)	HIPERFACE DSL 通信プロトコルによる通信機能を備えたアブソリュートエンコーダとの通信
割り込みコントローラ(ICU)	HFDSL の割り込み制御
汎用 PWM タイマ(GPT) チャネル 0	ELC に入力するイベント周期の生成
イベントリンクコントローラ（ELC）	GPT チャネル 0 が出力するイベントと HFDSL をリンク
シリアル通信インタフェース(SCI) UART	SCI の調歩同期式 I/F を使用し、USB インタフェースによる COM ポート通信に使用 サンプルプログラムのコンソールインタフェース用

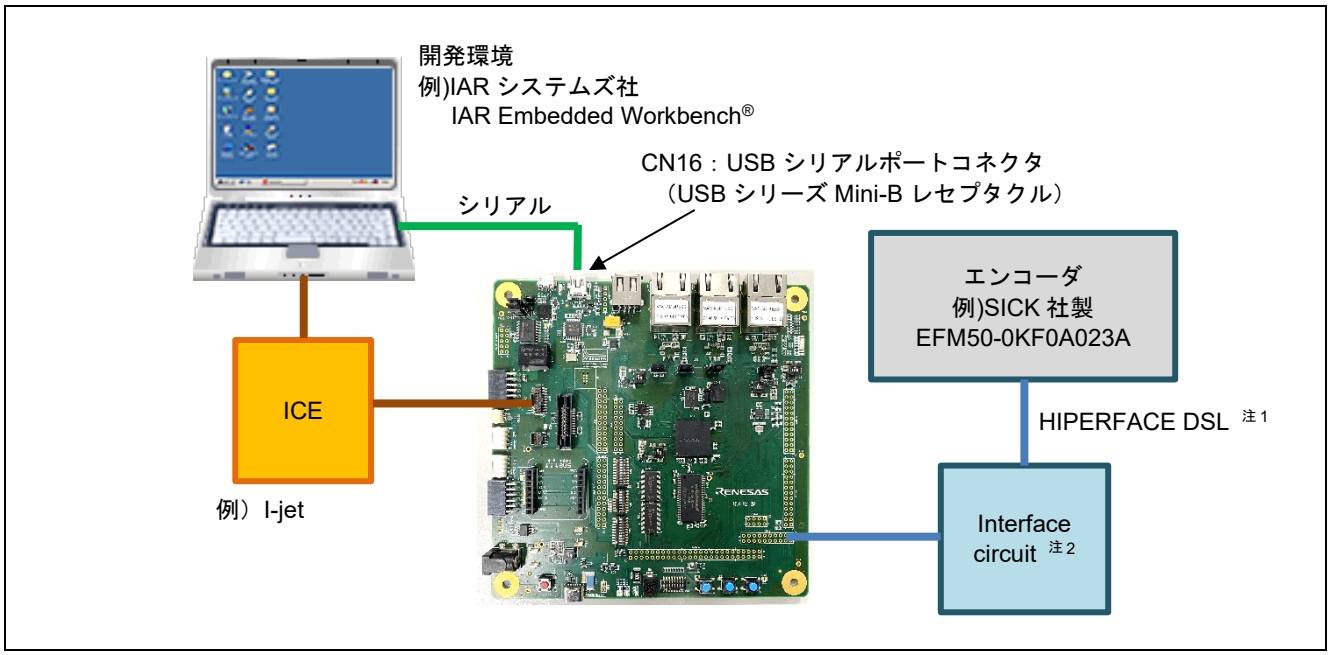

図 1-1　動作環境

【注】 1. 送受信可能なケーブル長は、エンコーダのマニュアルを参照してください。
2. 「HIPERFACE DSL® MASTER Integration Manual」を参照してください。

2. 動作環境

本アプリケーションノートのサンプルコードは、下記の環境を想定しています。

表 2-1　動作環境

項目	内容
使用マイコン	RZ/T2M グループ
動作周波数	CPUCLK = 800MHz
動作電圧	1.1V(Core) / 1.8V(PLL, etc.) / 3.3V(I/O)
統合開発環境 注 1	IAR システムズ製　IAR Embedded Workbench® for Arm® RENESAS 製　e² studio
使用ボード	RSK+RZT2M（RTK9RZT2M0C00000BE）
使用デバイス（ボード上で使用する機能）	なし

【注】 1. 統合開発環境のバージョンは、RZ/T2M グループ Encoder I/F HIPERFACE DSL sample program リリースノートを参照してください。

3. 周辺機能説明

周辺機能、動作モード、レジスタについての基本的な内容は、RZ/T2M グループ・ユーザーズマニュアル ハードウェア編に記載しています。

3.1 使用端子一覧

表 3-1 に本アプリケーションノートのサンプルコードで使用する端子と機能を示します。

表 3-1 使用端子と機能

チャネル	端子名 (機能ピン名)	I/O ポート	入出力	内容
HFDSL0	dsl_in0 (ENCIF0)	P01_6	入力	データ入力端子
	dsl_out0 (ENCIF2)	P02_0	出力	データ出力端子
	dsl_en0 (ENCIF3)	P02_2	出力	ドライブ／レシーブ制御端子
HFDSL1	dsl_in1 (ENCIF5)	P17_3	入力	データ入力端子
	dsl_out1 (ENCIF7)	P17_5	出力	データ出力端子
	dsl_en1 (ENCIF8)	P03_0	出力	ドライブ／レシーブ制御端子

4. ソフトウェア説明

4.1 HFDSL ドライバ機能

HFDSL ドライバの機能は以下です。

1. 初期設定
2. 位置データの取得
3. メッセージの送受信

4.2 ファイル構成

ファイル構成は、RZ/T2M グループ Encoder I/F HIPERFACE DSL sample program リリースノートを参照してください。

4.3 関数一覧

表 4-1 に関数一覧を示します。

表 4-1 関数一覧

カテゴリ	関数名	ページ番号
HFDSL ドライバ API 関数	R_HFDSL_Open	8
	R_HFDSL_Close	8
	R_HFDSL_GetVersion	9
	R_HFDSL_CheckInitSeq	9
	R_HFDSL_Control	9
ユーザー定義関数	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
割り込みハンドラ	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 API 関数仕様

4.4.1 R_HFDSL_Open

R_HFDSL_Open	
概　要	エンコーダ制御の開始
ヘッダ	r_hfdsl_rzt2_if.h
宣　言	int32_t R_HFDSL_Open(const int32_t id, r_hfdsl_info_t* p_info);
説　明	HFDSL ドライバを使用する前に本関数を実行してください。本関数ではドライバの初期化を行います。 ・割り込みの設定 ・コールバック関数の設定
引　数」	id　：使用する ID を指定します。(r_hfdsl_rzt2_dat.h で定義されています。) R_HFDSL0_ID　：チャネル 0 を指定 R_HFDSL1_ID　：チャネル 1 を指定 上記以外　：設定不可 p_info　：ドライバの初期設定情報 ドライバの初期設定情報を格納した r_hfdsl_info_t 構造体のポインタを指定してください。
リターン値	R_HFDSL_SUCCESS：正常終了 R_HFDSL_ERR_INVALID_ARG：異常終了 (id, p_info に指定した r_hfdsl_info_t 構造体のメンバ変数が規定されていない値) R_HFDSL_ERR_ACCESS：異常終了（既に本関数が実行されています）
注　意	本関数実行前に、必ず EC-Lib を用いて Multi-Protocol Encoder I/F のコンフィグレーションと起動を行ってください。 コールバック関数内で、本 API 関数を実行することは禁止します。

4.4.2 R_HFDSL_Close

R_HFDSL_Close	
概　要	エンコーダの制御を終了
ヘッダ	r_hfdsl_rzt2_if.h
宣　言	int32_t R_HFDSL_Close(const int32_t id);
説　明	R_HFDSL_Open で設定した割り込みをディセーブルし、エンコーダの制御を終了します。
引　数	id　：使用する ID を指定します。(r_hfdsl_rzt2_dat.h で定義されています。) R_HFDSL0_ID　：チャネル 0 を指定 R_HFDSL1_ID　：チャネル 1 を指定 上記以外　：設定不可
リターン値	R_HFDSL_SUCCESS：正常終了 R_HFDSL_ERR_INVALID_ARG：異常終了 (id に指定した値が規定されていない値)
注　意	本関数実行前に、必ず R_HFDSL_Open を実行してください。 コールバック関数内で、本 API 関数を実行することは禁止します。

4.4.3 R_HFDSL_GetVersion

R_HFDSL_GetVersion	
概　要	ドライバのバージョンを取得
ヘッダ	r_hfdsl_rzt2_if.h
宣　言	uint32_t R_HFDSL_GetVersion(void);
説　明	ドライバのバージョンを取得します。
引　数	なし
リターン値	上位 16 ビットにメジャーバージョン、下位 16 ビットにマイナーバージョンが格納されます。 例）リターン値が 00010002h の場合、Ver.1.2
注　意	本関数実行前に、必ず EC-Lib を用いて Multi-Protocol Encoder I/F のコンフィグレーションと起動を行ってください。

4.4.4 R_HFDSL_CheckInitSeq

R_HFDSL_CheckInitSeq	
概　要	エンコーダの初期化状態
ヘッダ	r_hfdsl_rzt2_if.h
宣　言	int32_t R_HFDSL_CheckInitSeq (const int32_t id);
説　明	エンコーダの初期化状態を取得します。
引　数	id　：使用する ID を指定します。(r_hfdsl_rzt2_dat.h で定義されています。) R_HFDSL0_ID　：チャネル 0 を指定 R_HFDSL1_ID　：チャネル 1 を指定 上記以外　：設定不可
リターン値	0～3：EVENT レジスタの INIT[1:0]ビットの値を示します。 R_HFDSL_ERR_INVALID_ARG：異常終了（id に設定した値が不正）

4.4.5 R_HFDSL_Control

R_HFDSL_Control	
概　要	エンコーダの制御
ヘッダ	r_hfdsl_rzt2_if.h
宣　言	int32_t R_HFDSL_Control(const int32_t id, const r_hfdsl_cmd_t cmd, void *const p_buf);
説　明	引数 cmd を使ってエンコーダを制御します。 制御コマンドの動作は「4.4.5(1) プロトコル初期化コマンド」と「4.4.5(2) 制御コマンド」を参照してください。
引　数	id　：使用する ID を指定します。(r_hfdsl_rzt2_dat.h で定義されています。) R_HFDSL0_ID　：チャネル 0 を指定 R_HFDSL1_ID　：チャネル 1 を指定 上記以外　：設定不可 cmd　：コマンド 内容は「4.4.5(1) プロトコル初期化コマンド」と「4.4.5(2) 制御コマンド」を参照してください。 p_buf　：各 cmd に対応する引数
リターン値	R_HFDSL_SUCCESS：正常終了 R_HFDSL_ERR_INVALID_ARG：異常終了（id, cmd に設定した値が不正） その他リターン値は「4.4.5(1) プロトコル初期化コマンド」と「4.4.5(2) 制御コマンド」を参照してください。

(1) プロトコル初期化コマンド

(a) R_HFDSL_CMD_INIT1

R_HFDSL_CMD_INIT1	
概　要	プロトコルの初期化 1
ヘッダ	r_hfdsl_rzt2_if.h
宣　言	int32_t R_HFDSL_Control(const int32_t id, const r_hfdsl_cmd_t cmd, void *const p_buf);
説　明	R_HFDSL_Open 関数実行後、またはプロトコルリセットが発生した後に実行してください。本関数に続けてプロトコル初期化コマンド「R_HFDSL_CMD_INIT2～R_HFDSL_CMD_INIT6」を実行してください。プロトコルリセット発生の検出方法については、「4.5.2 hfdsl_int_err_callback」を参照してください。
引　数	id　：使用する ID を指定します。(r_hfdsl_rzt2_dat.h で定義されています。) R_HFDSL0_ID　：チャネル 0 を指定 R_HFDSL1_ID　：チャネル 1 を指定 上記以外　：設定不可 cmd　：R_HFDSL_CMD_INIT1 を指定します。 p_buf　：NULL を設定してください。
リターン値	R_HFDSL_SUCCESS：正常終了 R_HFDSL_ERR_INVALID_ARG：異常終了（id、p_buf が不正値） R_HFDSL_ERR_ACCESS：異常終了（R_HFDSL_Open が実行されていない。）
注　意	コールバック関数内で、本 API 関数を実行することは禁止します。

(b) R_HFDSL_CMD_INIT2

R_HFDSL_CMD_INIT2	
概　要	プロトコルの初期化 2
ヘッダ	r_hfdsl_rzt2_if.h
宣　言	int32_t R_HFDSL_Control(const int32_t id, const r_hfdsl_cmd_t cmd, void *const p_buf);
説　明	プロトコル初期化コマンド「R_HFDSL_CMD_INIT1」を実行し、EVENT レジスタの INIT ビットが 1 になってから、本関数を実行してください。本関数実行後に、プロトコル初期化コマンド「R_HFDSL_CMD_INIT3」を実行してください。
引　数	id　：使用する ID を指定します。(r_hfdsl_rzt2_dat.h で定義されています。) R_HFDSL0_ID　：チャネル 0 を指定 R_HFDSL1_ID　：チャネル 1 を指定 上記以外　：設定不可 cmd　：R_HFDSL_CMD_INIT2 を指定します。 p_buf　：NULL を設定してください。
リターン値	R_HFDSL_SUCCESS：正常終了 R_HFDSL_ERR_INVALID_ARG：異常終了（id、p_buf が不正値） R_HFDSL_ERR_ACCESS：異常終了（R_HFDSL_CMD_INIT1 が実行されていない。）
注　意	コールバック関数内で、本 API 関数を実行することは禁止します。

(c) R_HFDSL_CMD_INIT3

R_HFDSL_CMD_INIT3	
概　要	プロトコルの初期化 3
ヘッダ	r_hfdsl_rzt2_if.h
宣　言	int32_t R_HFDSL_Control(const int32_t id, const r_hfdsl_cmd_t cmd, void *const p_buf);
説　明	プロトコル初期化コマンド「R_HFDSL_CMD_INIT2」を実行してから、本関数を実行してください。本関数実行後に、プロトコル初期化コマンド「R_HFDSL_CMD_INIT4」を実行してください。 各レジスタの設定値は「表 4-3　HFDSL ドライバで使用するユーザー定義の定数 (r_hfdsl_rzt2_config.h)」記載の定数です。HFDSL の動作を変更する場合は、この定数を変更してください。
引　数	id　：使用する ID を指定します。(r_hfdsl_rzt2_dat.h で定義されています。) R_HFDSL0_ID　：チャネル 0 を指定 R_HFDSL1_ID　：チャネル 1 を指定 上記以外　：設定不可 cmd　：R_HFDSL_CMD_INIT3 を指定します。 p_buf　：NULL を設定してください。
リターン値	R_HFDSL_SUCCESS：正常終了 R_HFDSL_ERR_INVALID_ARG：異常終了 (id、p_buf が不正値) R_HFDSL_ERR_ACCESS：異常終了（R_HFDSL_CMD_INIT2 が実行されていない。）
注　意	コールバック関数内で、本 API 関数を実行することは禁止します。

(d) R_HFDSL_CMD_INIT4

R_HFDSL_CMD_INIT4	
概　要	プロトコルの初期化 4
ヘッダ	r_hfdsl_rzt2_if.h
宣　言	int32_t R_HFDSL_Control(const int32_t id, const r_hfdsl_cmd_t cmd, void *const p_buf);
説　明	プロトコル初期化コマンド「R_HFDSL_CMD_INIT3」を実行し、EVENT レジスタの INIT ビットが 2 になった後に、1ms 待機後、本関数を実行してください。本関数実行後に、プロトコル初期化コマンド「R_HFDSL_CMD_INIT5」を実行してください。 リターン値が R_HFDSL_ERR_INIT で再度プロトコルの初期化を行う場合は、制御コマンド「R_HFDSL_CMD_RST」を実行後、プロトコル初期化コマンド「R_HFDSL_CMD_INIT1」からやり直してください。
引　数	id　：使用する ID を指定します。(r_hfdsl_rzt2_dat.h で定義されています。) R_HFDSL0_ID　：チャネル 0 を指定 R_HFDSL1_ID　：チャネル 1 を指定 上記以外　：設定不可 cmd　：R_HFDSL_CMD_INIT4 を指定します。 p_buf　：NULL を設定してください。
リターン値	R_HFDSL_SUCCESS：正常終了 R_HFDSL_ERR_INVALID_ARG：異常終了 (id、p_buf が不正値) R_HFDSL_ERR_ACCESS：異常終了（R_HFDSL_CMD_INIT3 が実行されていない。） R_HFDSL_ERR_INIT：異常終了 (EDGES レジスタ値が NG)
注　意	コールバック関数内で、本 API 関数を実行することは禁止します。

(e) R_HFDSL_CMD_INIT5

R_HFDSL_CMD_INIT5	
概　要	プロトコルの初期化 5
ヘッダ	r_hfdsl_rzt2_if.h
宣　言	int32_t R_HFDSL_Control(const int32_t id, const r_hfdsl_cmd_t cmd, void *const p_buf);
説　明	プロトコル初期化コマンド「R_HFDSL_CMD_INIT4」を実行し、24us 以上待機後に、本関数を実行してください。本関数実行後に、プロトコル初期化コマンド「R_HFDSL_CMD_INIT6」を実行してください。 リターン値が R_HFDSL_ERR_INIT で再度プロトコルの初期化を行う場合は、制御コマンド「R_HFDSL_CMD_RST」を実行後、プロトコル初期化コマンド「R_HFDSL_CMD_INIT1」からやり直してください。 ケーブルによる遅延時間の許容上限値は、「表 4-3　HFDSL ドライバで使用するユーザー定義の定数(r_hfdsl_rzt2_config.h)」の「R_HFDSL_DELAY_UPP_LIMIT」で定義しています。
引　数	id　：使用する ID を指定します。(r_hfdsl_rzt2_dat.h で定義されています。) R_HFDSL0_ID　：チャネル 0 を指定 R_HFDSL1_ID　：チャネル 1 を指定 上記以外　：設定不可 cmd　：R_HFDSL_CMD_INIT5 を指定します。 p_buf　：NULL を設定してください。
リターン値	R_HFDSL_SUCCESS：正常終了 R_HFDSL_ERR_INVALID_ARG：異常終了 (id、p_buf が不正値) R_HFDSL_ERR_ACCESS：異常終了 (R_HFDSL_CMD_INIT4 が実行されていない。) R_HFDSL_ERR_INIT：異常終了 (ケーブルによる遅延時間が大きく、正常に通信ができない場合がある。)
注　意	コールバック関数内で、本 API 関数を実行することは禁止します。

(f)　R_HFDSL_CMD_INIT6

R_HFDSL_CMD_INIT6	
概　要	プロトコルの初期化 6
ヘッダ	r_hfdsl_rzt2_if.h
宣　言	int32_t R_HFDSL_Control(const int32_t id, const r_hfdsl_cmd_t cmd, void *const p_buf);
説　明	プロトコル初期化コマンド「R_HFDSL_CMD_INIT5」を実行し、EVENT レジスタの INIT ビットが 3 になってから、本関数を実行してください。本関数が正常終了し、EVENT レジスタの INIT_END ビットが 1 になってから、位置値等の取得が行われます。 リターン値が R_HFDSL_ERR_INIT で再度プロトコルの初期化を行う場合は、制御コマンド「R_HFDSL_CMD_RST」を実行後、プロトコル初期化コマンド「R_HFDSL_CMD_INIT1」からやり直してください。
引　数	id　：使用する ID を指定します。(r_hfdsl_rzt2_dat.h で定義されています。) R_HFDSL0_ID　：チャネル 0 を指定 R_HFDSL1_ID　：チャネル 1 を指定 上記以外　：設定不可 cmd　：R_HFDSL_CMD_INIT6 を指定します。 p_buf　：エンコーダ ID エンコーダ ID を格納した uint32_t 型のポインタ変数を設定してください。
リターン値	R_HFDSL_SUCCESS：正常終了 R_HFDSL_ERR_INVALID_ARG：異常終了 (id が不正値、p_buf が NULL) R_HFDSL_ERR_ACCESS：異常終了 (R_HFDSL_CMD_INIT5 が実行されていない。) R_HFDSL_ERR_INIT：異常終了 (接続しているエンコーダと指定したエンコーダ ID が違う。)
注　意	コールバック関数内で、本 API 関数を実行することは禁止します。

(2)　制御コマンド

(a)　R_HFDSL_CMD_POS

R_HFDSL_CMD_POS	
概　要	Fast position の取得
ヘッダ	r_hfdsl_rzt2_if.h
宣　言	int32_t R_HFDSL_Control(const int32_t id, const r_hfdsl_cmd_t cmd, void *const p_buf);
説　明	Fast position レジスタ H(POS_H)、Fast position レジスタ(POS)をリードして、Fast position を取得します。
引　数	id　：使用する ID を指定します。(r_hfdsl_rzt2_dat.h で定義されています。) R_HFDSL0_ID　：チャネル 0 を指定 R_HFDSL1_ID　：チャネル 1 を指定 上記以外　：設定不可 cmd　：R_HFDSL_CMD_POS を指定します。 p_buf　：Fast position Fast position を格納する r_hfdsl_pos_t 構造体のポインタを指定します。詳細は「4.10.1(2) r_hfdsl_pos_t」参照してください。
リターン値	R_HFDSL_SUCCESS：正常終了 R_HFDSL_ERR_INVALID_ARG：異常終了（id が不正値、p_buf が NULL）

(b) R_HFDSL_CMD_VPOS

R_HFDSL_CMD_VPOS	
概　要	Safe position の取得
ヘッダ	r_hfdsl_rzt2_if.h
宣　言	int32_t R_HFDSL_Control(const int32_t id, const r_hfdsl_cmd_t cmd, void *const p_buf);
説　明	Safe position レジスタ H（VPOS_H）、Safe position レジスタ(VPOS)、Safe position CRC レジスタ(VPOSCRC)をリードして、Safe position を取得します。
引　数	id　：使用する ID を指定します。(r_hfdsl_rzt2_dat.h で定義されています。) R_HFDSL0_ID　：チャネル 0 を指定 R_HFDSL1_ID　：チャネル 1 を指定 上記以外　：設定不可 cmd　：R_HFDSL_CMD_VPOS を指定します。 p_buf　：Safe position Safe position を格納する r_hfdsl_vpos_t 構造体のポインタを指定します。詳細は「4.10.1(3) r_hfdsl_vpos_t」参照してください。
リターン値	R_HFDSL_SUCCESS：正常終了 R_HFDSL_ERR_INVALID_ARG：異常終了（id が不正値、p_buf が NULL）

(c) R_HFDSL_CMD_VEL

R_HFDSL_CMD_VEL	
概　要	モータの回転速度の取得
ヘッダ	r_hfdsl_rzt2_if.h
宣　言	int32_t R_HFDSL_Control(const int32_t id, const r_hfdsl_cmd_t cmd, void *const p_buf);
説　明	速度レジスタ(VEL)をリードして、モータの回転速度を取得します。
引　数	id　：使用する ID を指定します。(r_hfdsl_rzt2_dat.h で定義されています。) R_HFDSL0_ID　：チャネル 0 を指定 R_HFDSL1_ID　：チャネル 1 を指定 上記以外　：設定不可 cmd　：R_HFDSL_CMD_VEL を指定します。 p_buf　：モータの回転速度 モータの回転速度を格納する uint32_t 型のポインタを指定します。
リターン値	R_HFDSL_SUCCESS：正常終了 R_HFDSL_ERR_INVALID_ARG：異常終了（id が不正値、p_buf が NULL）

(d) R_HFDSL_CMD_MSG

R_HFDSL_CMD_MSG	
概　要	メッセージの送信
ヘッダ	r_hfdsl_rzt2_if.h
宣　言	int32_t R_HFDSL_Control(const int32_t id, const r_hfdsl_cmd_t cmd, void *const p_buf);
説　明	メッセージの送信を行います。受信データは hfdsl_int_mrcv_callback 関数によって通知されます。関数の詳細は「4.5.4 hfdsl_int_mrcv_callback」を参照してください。
引　数	id　：使用する ID を指定します。(r_hfdsl_rzt2_dat.h で定義されています。) R_HFDSL0_ID　：チャネル 0 を指定 R_HFDSL1_ID　：チャネル 1 を指定 上記以外　：設定不可 cmd　：R_HFDSL_CMD_MSG を指定します。 p_buf　：メッセージ送信データ メッセージ送信データを格納する r_hfdsl_send_msg_t 構造体のポインタを指定します。詳細は「4.10.1(4) r_hfdsl_send_msg_t」参照してください。
リターン値	R_HFDSL_SUCCESS：正常終了 R_HFDSL_ERR_INVALID_ARG：異常終了（id が不正値、p_buf が NULL） R_HFDSL_ERR_ACCESS：異常終了 (4.4.5(1) プロトコル初期化コマンドが実行されていない)
注　意	コールバック関数内で、本 API 関数を実行することは禁止します。 次の送信を行う場合は、hfdsl_int_mrcv_callback 関数がコールされてから、本関数を実行して送信を行ってください。

(e) R_HFDSL_CMD_RST

R_HFDSL_CMD_RST	
概　要	プロトコルリセット
ヘッダ	r_hfdsl_rzt2_if.h
宣　言	int32_t R_HFDSL_Control(const int32_t id, const r_hfdsl_cmd_t cmd, void *const p_buf);
説　明	プロトコルをリセットします。 本関数を実行後は、EVENT_ERR レジスタの PRST ビットによる、INT_err 割り込みが発生します。 通信を再開する場合は、「4.4.5(1) プロトコル初期化コマンド」を実行してください。
引　数	id　：使用する ID を指定します。(r_hfdsl_rzt2_dat.h で定義されています。) R_HFDSL0_ID　：チャネル 0 を指定 R_HFDSL1_ID　：チャネル 1 を指定 上記以外　：設定不可 cmd　：R_HFDSL_CMD_RST を指定します。 p_buf　：NULL を設定してください。
リターン値	R_HFDSL_SUCCESS：正常終了 R_HFDSL_ERR_INVALID_ARG：異常終了（id、p_buf が不正値）

4.5　ユーザー定義関数仕様

4.5.1　hfdsl_int_nml_callback

hfdsl_int_nml_callback	
概　要	INT_nml 割り込み発生を通知
ヘッダ	r_hfdsl_rzt2_if.h
宣　言	void hfdsl_int_nml_callback(uint8_t event);
説　明	R_HFDSL_Open 関数の引数 r_hfdsl_info_t 構造体のメンバ変数 p_cb_nml で登録したコールバック関数です。EVENT レジスタの MIN ビット、POS_RDY ビット、POS_UPD ビットによる INT_nml 割り込み発生時にコールされます。 EVENT レジスタの POS_RDY ビット、POS_UPD ビットが 1 の場合には、Fast position レジスタ H(POS_H)、Fast position レジスタ(POS)が更新されています。その場合、本関数内で R_HFDSL_Control(R_HFDSL_CMD_POS)関数を実行することで、Fast position を取得することができます。 本関数は割り込みハンドラのコンテキストとなります。割り込みの応答性を確保するため、速やかにリターンするようにしてください。関数名は例であり、自由に設定できます。
引　数	event　：INT_nml 割り込みの発生要因 EVENT レジスタの値が格納されています。 本関数内のみ有効です。
リターン値	なし
注　意	EVENT レジスタの MRCV ビット、INIT_END ビットによる INT_nml 割り込み発生時はコールされません。

4.5.2　hfdsl_int_err_callback

hfdsl_int_err_callback	
概　要	INT_err 割り込み発生を通知
ヘッダ	r_hfdsl_rzt2_if.h
宣　言	void hfdsl_int_err_callback(uint32_t event_err);
説　明	R_HFDSL_Open 関数の引数 r_hfdsl_info_t 構造体のメンバ変数 p_cb_err で登録したコールバック関数です。INT_err 割り込み発生時にコールされます。 本関数は割り込みハンドラのコンテキストとなります。割り込みの応答性を確保するため、速やかにリターンするようにしてください。関数名は例であり、自由に設定できます。
引　数	event_err　：INT_err 割り込みの発生要因 EVENT_ERR レジスタの値が格納されています。本関数内のみ有効です。
リターン値	なし

4.5.3 hfdsl_int_raw_callback

hfdsl_int_raw_callback	
概　要	INT_tovr 割り込みが発生し、FIFO に値が格納されたことを通知
ヘッダ	r_hfdsl_rzt2_if.h
宣　言	void hfdsl_int_raw_callback(uint16_t* raw_data);
説　明	R_HFDSL_Open 関数の引数 r_hfdsl_info_t 構造体のメンバ変数 p_cb_raw で登録したコールバック関数です。INT_tovr 割り込みが発生し、FIFO に値が格納された時にコールされます。 本関数は割り込みハンドラのコンテキストとなります。割り込みの応答性を確保するため、速やかにリターンするようにしてください。関数名は例であり、自由に設定できます。
引　数	raw_data[]　：FIFO_DATA レジスタ値 FIFO_DATA レジスタ値が格納されています。 次の INT_tovr 割り込み発生まで有効です。
リターン値	なし

4.5.4 hfdsl_int_mrcv_callback

hfdsl_int_mrcv_callback	
概　要	INT_tovr 割り込みが発生し、受信メッセージのデータ格納が完了したことを通知
ヘッダ	r_hfdsl_rzt2_if.h
宣　言	void hfdsl_int_mrcv_callback(uint16_t* msg_data);
説　明	R_HFDSL_Control (R_HFDSL_CMD_MSG)関数で登録したコールバック関数です。 INT_tovr 割り込みが発生し、受信メッセージのデータ格納が完了した時にコールされます。 本関数は割り込みハンドラのコンテキストとなります。割り込みの応答性を確保するため、速やかにリターンするようにしてください。関数名は例であり、自由に設定できます。
引　数	msg_data[]　：FIFO_DATA レジスタ値 Short and long massages の FIFO_DATA レジスタ値が受信データサイズ分格納されています。 次の INT_tovr 割り込み発生まで有効です。
リターン値	なし

4.5.5 hfdsl_int_init_callback

hfdsl_int_init_callback	
概　要	EVENT レジスタの INIT_END ビットによる INT_nml 割り込み発生を通知
ヘッダ	r_hfdsl_rzt2_if.h
宣　言	void hfdsl_int_init_callback(void);
説　明	R_HFDSL_Open 関数の引数 r_hfdsl_info_t 構造体のメンバ変数 p_cb_init で登録したコールバック関数です。EVENT レジスタの INIT_END ビットによる INT_nml 割り込み発生時にコールされます。 本関数は割り込みハンドラのコンテキストとなります。割り込みの応答性を確保するため、速やかにリターンするようにしてください。関数名は例であり、自由に設定できます。
引　数	なし
リターン値	なし

4.6 割り込みハンドラ

4.6.1 hfdsl_int_nml_isr_ch0

hfdsl_int_nml_isr_ch0	
概　要	INT_nml ch0 割り込みの割り込みハンドラ
ヘッダ	-
宣　言	static void hfdsl_int_nml_isr_ch0(void);
説　明	INT_nml 割り込みに対する割り込みハンドラです。 割り込み要因が EVENT レジスタの INIT_END ビットだった場合は、コールバック関数の hfdsl_int_init_callback 関数をコールします。 その他の EVENT レジスタのビットが要因で割り込みが発生した場合は、コールバック関数の hfdsl_int_nml_callback 関数をコールします。
引　数	なし
リターン値	なし

4.6.2 hfdsl_int_nml_isr_ch1

hfdsl_int_nml_isr_ch1	
概　要	INT_nml ch1 割り込みの割り込みハンドラ
ヘッダ	-
宣　言	static void hfdsl_int_nml_isr_ch1(void);
説　明	INT_nml 割り込みに対する割り込みハンドラです。 割り込み要因が EVENT レジスタの INIT_END ビットだった場合は、コールバック関数の hfdsl_int_init_callback 関数をコールします。 その他の EVENT レジスタのビットが要因で割り込みが発生した場合は、コールバック関数の hfdsl_int_nml_callback 関数をコールします。
引　数	なし
リターン値	なし

4.6.3 hfdsl_int_err_isr_ch0

hfdsl_int_err_isr_ch0	
概　要	INT_err ch0 割り込みの割り込みハンドラ
ヘッダ	-
宣　言	static void hfdsl_int_err_isr_ch0(void);
説　明	INT_err 割り込みに対する割り込みハンドラです。 割り込み発生時はコールバック関数の hfdsl_int_err_callback 関数をコールします。
引　数	なし
リターン値	なし

4.6.4 hfdsl_int_err_isr_ch1

hfdsl_int_err_isr_ch1	
概　要	INT_err ch1 割り込みの割り込みハンドラ
ヘッダ	-
宣　言	static void hfdsl_int_err_isr_ch1(void);
説　明	INT_err 割り込みに対する割り込みハンドラです。 割り込み発生時はコールバック関数の hfdsl_int_err_callback 関数をコールします。
引　数	なし
リターン値	なし

4.6.5 hfdsl_int_tovr_isr_ch0

hfdsl_int_tovr_isr_ch0	
概　要	INT_tovr ch0 割り込みの割り込みハンドラ
ヘッダ	-
宣　言	static void hfdsl_int_tovr_isr_ch0(void);
説　明	INT_tovr 割り込みに対する割り込みハンドラです。 メッセージ受信が完了した場合は、コールバック関数の hfdsl_int_mrcv_callback 関数をコールします。 H フレーム 1 回分の受信が完了した場合は、コールバック関数の hfdsl_int_raw_callback 関数をコールします。
引　数	なし
リターン値	なし

4.6.6 hfdsl_int_tovr_isr_ch1

hfdsl_int_tovr_isr_ch1	
概　要	INT_tovr ch1 割り込みの割り込みハンドラ
ヘッダ	-
宣　言	static void hfdsl_int_tovr_isr_ch1(void);
説　明	INT_tovr 割り込みに対する割り込みハンドラです。 メッセージ受信が完了した場合は、コールバック関数の hfdsl_int_mrcv_callback 関数をコールします。 H フレーム 1 回分の受信が完了した場合は、コールバック関数の hfdsl_int_raw_callback 関数をコールします。
引　数	なし
リターン値	なし

4.7 使用割り込み一覧

表 4-2 に HFDSL ドライバで使用する割り込みを示します。

表 4-2 HFDSL ドライバで使用する割り込み

割り込み	ID	概要
INT_nml0 (ENCIF_INT0)	372	ch0 EVENT レジスタの何れかのビットが“1”に更新されると割り込みが発生します。
INT_nml1 (ENCIF_INT4)	376	ch1 EVENT レジスタの何れかのビットが”1”に更新されると割り込みが発生します。
INT_err0 (ENCIF_INT1)	373	ch0 EVENT_ERR レジスタの何れかのビットが“1”に更新されると割り込みが発生します。
INT_err1 (ENCIF_INT5)	377	ch1 EVENT_ERR レジスタの何れかのビットが“1”に更新されると割り込みが発生します。
INT_tovr0 (ENCIF_INT2)	374	ch0 FIFO_DATA レジスタの受信数が閾値以上になると割り込みが発生します。
INT_tovr1 (ENCIF_INT6)	378	ch1 FIFO_DATA レジスタの受信数が閾値以上になると割り込みが発生します。

4.8　定数/エラーコード一覧

定数とエラーコードを示します。各定義については、それぞれの表を参照してください。

表 4-3　HFDSL ドライバで使用するユーザー定義の定数(r_hfdsl_rzt2_config.h)

定数名	設定値	内容
R_HFDSL_ES_PRDY	0008h	SYS_CTRL レジスタの ES ビットと PRDY ビットの設定値
R_HFDSL_MAXACC	1023	MAXACC レジスタの設定値
R_HFDSL_ACC_ERR	31	ACC_ERR レジスタの設定値
R_HFDSL_MAXDEV	65535	MAXDEV レジスタの設定値
R_HFDSL_MASK	64h	MASK レジスタの設定値 注 1
R_HFDSL_MASK_ERR	00FF33DFh	MASK_ERR レジスタの設定値 注 1
R_HFDSL_RAW_EN	FFh	RAW_EN レジスタの設定値 注 2
R_HFDSL_DELAY_UPP_LIMIT	09h	ケーブルによる遅延時間の許容上限値 注 3
R_HFDSL_STUFF	08h	STUFF レジスタの設定値
R_HFDSL_EXLEN	00h	EXLEN レジスタの設定値
R_HFDSL_EXTRA	0003h	EXTRA レジスタの設定値

【注】 1. R_HFDSL_MASK と R_HFDSL_MASK_ERR を変更する場合は、割り込みハンドラ hfdsl_int_nml_isr_ch0 関数、hfdsl_int_nml_isr_ch1 関数と hfdsl_int_err_isr_ch0 関数、hfdsl_int_err_isr_ch1 関数の処理を、R_HFDSL_MASK と R_HFDSL_MASK_ERR の設定値に合わせて変更してください。

2. R_HFDSL_RAW_EN を変更する場合は、「表 4-6　主要な static 型変数」の shub 配列と hfdsl_shub 関数、hfdsl_int_raw_callback 関数を、R_HFDSL_RAW_EN の設定値に合わせて変更してください。本サンプルプログラムでは、FIFO_DATA レジスタから 11 個の値を、shub 配列に保存しています。

3. ケーブルによる遅延時間の許容上限値を 900ns～1000ns(ケーブル長約 100m)に設定しています。ケーブルによる遅延時間の許容上限値を変更する場合は、本定数の設定値を変更してください。

表 4-4　エラーコード

定数名	設定値	内容
R_HFDSL_SUCCESS	0	正常終了
R_HFDSL_ERR_INVALID_ARG	-1	引数異常
R_HFDSL_ERR_ACCESS	-2	API の実行順序エラー
R_HFDSL_ERR_INIT	-3	HFDSL、エンコーダの初期化に失敗

4.9 固定幅整数一覧

表 4-5 にサンプルコードで使用する固定幅整数を示します。サンプルコードで使用する固定幅整数は、標準ライブラリで定義されています。

表 4-5 サンプルコードで使用する固定幅整数

シンボル	内容
int8_t	8 ビット整数、符号あり
int16_t	16 ビット整数、符号あり
int32_t	32 ビット整数、符号あり
int64_t	64 ビット整数、符号あり
uint8_t	8 ビット整数、符号なし
uint16_t	16 ビット整数、符号なし
uint32_t	32 ビット整数、符号なし
uint64_t	64 ビット整数、符号なし

4.10 構造体/共用体/列挙型一覧

主要な構造体／共用体／列挙型の一覧を記載します。

4.10.1 構造体

(1) r_hfdsl_info_t

HFDSL ドライバの初期化情報。

```
typedef struct
{
    r_hfdsl_int_nml_cb_t        p_cb_nml;   INT_nml 割り込み発生時にコールされるコールバック関数のポインタ
                                            詳細は「4.5.1 hfdsl_int_nml_callback」を参照してください。注1、注2
    r_hfdsl_int_err_cb_t        p_cb_err;   INT_err 割り込み発生時にコールされるコールバック関数のポインタ
                                            詳細は「4.5.2 hfdsl_int_err_callback」を参照してください。注1
    r_hfdsl_int_fifo_raw_cb_t   p_cb_raw;   INT_tovr 割り込み発生時にコールされるコールバック関数のポインタ
                                            詳細は「4.5.3 hfdsl_int_raw_callback」を参照してください。注1
    r_hfdsl_int_init_cb_t       p_cb_init   EVENT レジスタの INIT_END ビットによる INT_nml 割り込み発生時にコールされるコールバック関数のポインタ
                                            詳細は「4.5.5 hfdsl_int_init_callback」を参照してください。注1
} r_hfdsl_info_t
```

【注】 1：NULL を指定するとコールされません。
2：EVENT レジスタの MRCV ビット、INIT_END ビットによる INT_nml 割り込み発生時はコールされません。

(2) r_hfdsl_pos_t

Fast position の格納

```
typedef struct
{
    bool        all;    メンバ変数 posh 有効化の設定
                        (true：メンバ変数 posh が有効、
                        false：メンバ変数 posh が無効)
    uint8_t     posh;   Fast position の[39:32]ビットが格納されます。
                        メンバ変数 all が true だった場合に更新されます。
    uint32_t    pos;    Fast position の[31:0]ビットが格納されます。
} r_hfdsl_pos_t
```

(3) r_hfdsl_vpos_t

Safe position の格納

```
typedef struct
{
    uint8_t     vposh;    Safe position の[39:32]ビットが格納されます。
    uint32_t    vpos;     Safe position の[31:0]ビットが格納されます。
    uint16_t    crc;      Vertical channel の CRC が格納されます。
} r_hfdsl_vpos_t
```

(4) r_hfdsl_send_msg_t

メッセージ送信データを格納

```
typedef struct
{
    uint16_t        *p_data;    メッセージ送信データが格納された配列のポインタ
                                メッセージ送信データを格納する配列のポインタを設定してください。送信データサイズが 14 を超えると
                                R_HFDSL_ERR_INVALID_ARG が発生します。
    r_hfdsl_msg_cb_t    p_cb_msg;   メッセージ受信時にコールされるコールバック関数のポインタ
                                詳細は「4.5.4 hfdsl_int_mrcv_callback」を参照してください。
                                必ず hfdsl_int_mrcv_callback のアドレスを設定してください。
} r_hfdsl_send_msg_t
```

4.10.2 共用体

使用しません。

4.10.3 列挙型

使用しません。

4.11 サンプルプログラムの説明

4.11.1 動作概要

本サンプルプログラムは HIPERFACE DSL 通信プロトコル仕様に準拠したエンコーダ (SICK AG 社製 EFM50-0KF0A023A) に対応しています。本サンプルプログラムは以下の処理を行います。

1) コンソールから入力したコマンドで下記の情報を表示
   A) Fast position と Safe position
   B) モータの回転速度
   C) ロングメッセージの送受信結果 (Resources の Type of encoder)
   D) Sensor Hub Channel data
2) SYNC モードで動作
3) プロトコルリセットが発生した場合は、本サンプルプログラムを終了します。

(1) システムブロック図

図 4-1 にシステムブロック図を示します。

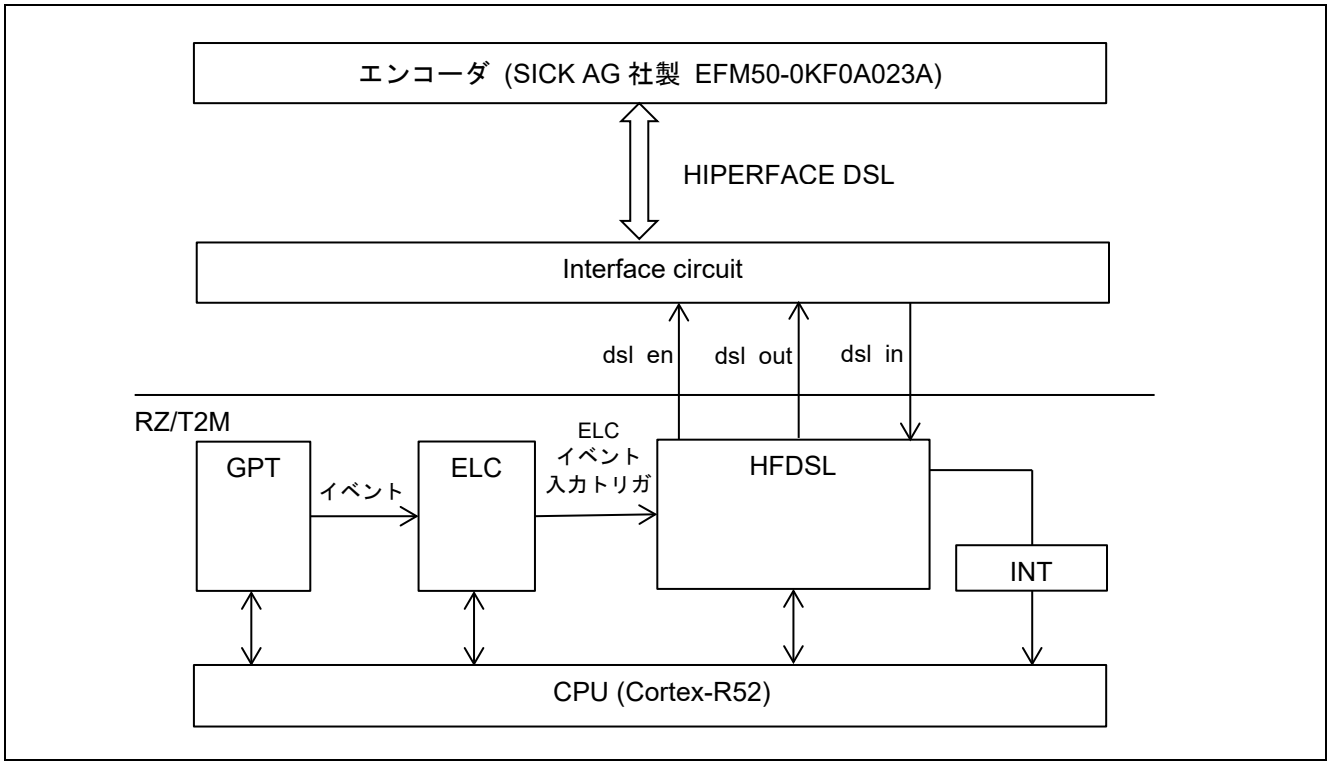

図 4-1 システムブロック図

(2)　ソフトウェア構成図

図 4-2 にソフトウェア構成図を示します。

HFDSL ドライバには、R_HFDSL_Open 関数で構成される開始処理部、R_HFDSL_Close 関数で構成される終了処理部、R_HFDSL_Control 関数で構成されるプロトコル初期化、位置値取得、メッセージ送信部、コールバック関数で構成されるデータ受信部分（割り込みハンドラ）があります。

サンプルプログラムには、HFDSL ドライバを制御し、位置値取得、メッセージの送信を行う HFDSL ドライバ制御部分、データ受信結果の表示を行う結果表示部分（コールバック）があります。

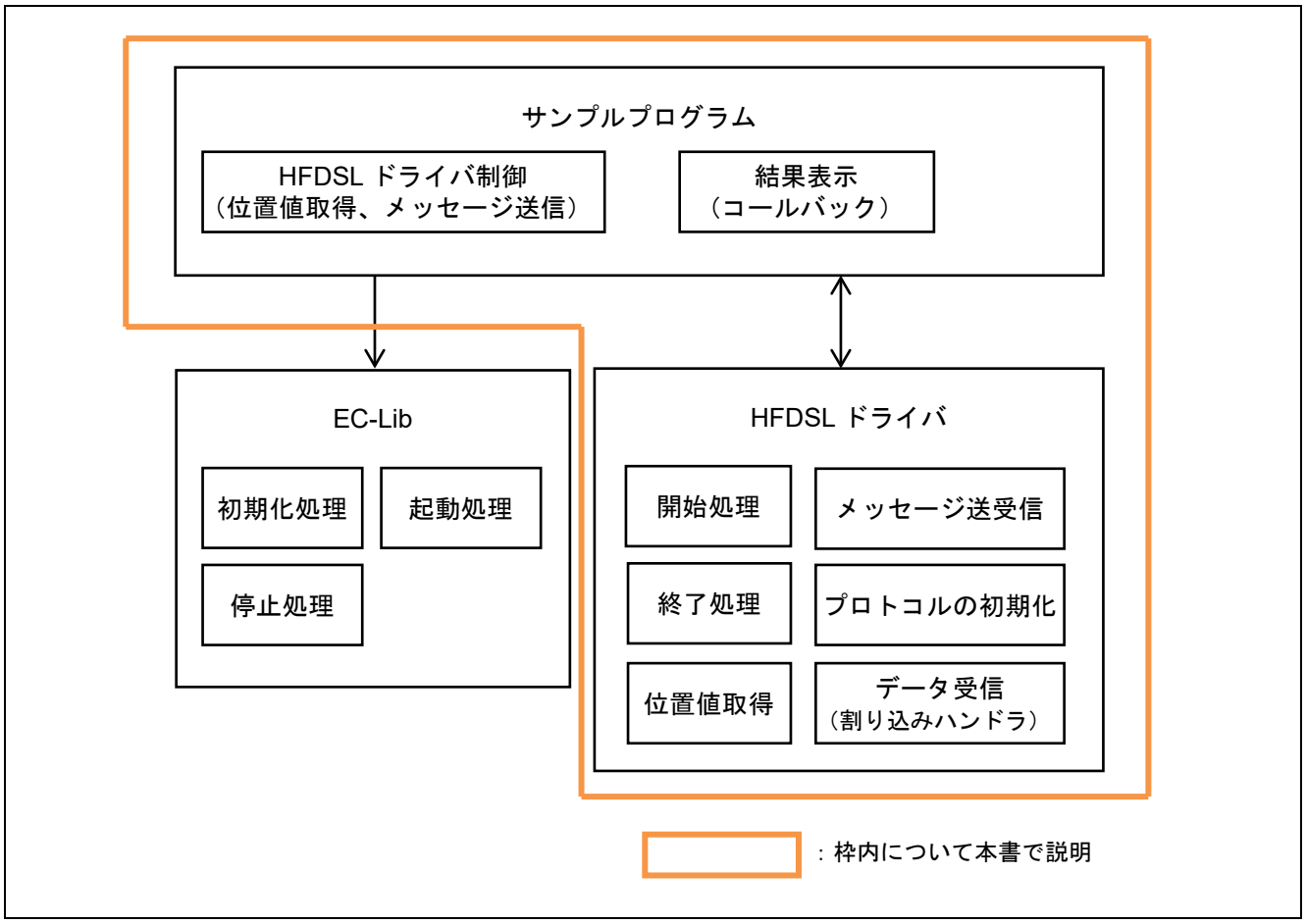

図 4-2　ソフトウェア構成図

4.11.2 サンプルプログラムの変数一覧

表 4-6 に主要な static 型変数を示します。

表 4-6　主要な static 型変数

型	変数名	内容
bool	mrcv_flg	メッセージ送受信完了フラグ (true：メッセージの送受信が完了、 false：メッセージが送受信中)
uint32_t	err_info	INT_err 割り込み発生要因を格納します。
uint32_t	pos_rot	Fast position の回転数を格納します。
uint32_t	pos_res	Fast position の角度を格納します。
uint32_t	vpos_rot	Safe position の回転数を格納します。
uint32_t	vpos_res	Safe position の角度を格納します。
uint32_t	vel	モータの回転速度を格納します。
uint16_t	lmsg_recv[LMSG_RECV_SIZE]	ロングメッセージの受信データを格納します。
uint16_t	shub[FIFO_DEP_MAX]	Sensor Hub データを含む受信データを格納します。
bool	init_end_flg	プロトコル初期化完了フラグ (true：プロトコル初期化完了、 false：プロトコル初期化未完了)

4.11.3 サンプルプログラムの定数一覧

表 4-7 にサンプルプログラムで使用する主要な定数を示します。

表 4-7　主要な定数

定数名	設定値	内容
ENC_ID	00000183h	EFM50-0KF0A023A のエンコーダ ID 注 1 注 2
RES_BIT	0	POS レジスタの位置情報の最下位ビット位置 注 1
RES_MASK	007FFFFFh	POS レジスタの位置情報のマスク 注 1
RES_MASK_H	00000000h	POS_H レジスタの位置情報のマスク 注 1
ROT_BIT	23	POS レジスタの回転数情報の最下位ビット位置 注 1
ROT_MASK	000001FFh	POS レジスタの回転数のマスク 注 1
ROT_MASK_H	00000E00h	POS_H レジスタの回転数のマスク 注 1
LMSG_RECV_SIZE	16	ロングメッセージの最大受信データサイズ
FIFO_DEP_MAX	11	FIFO_DATA レジスタのデータサイズ
TIMEOUT_UNIT	1000	タイムアウト単位設定(1ms)
TIMEOUT_COUNT	1000	タイムアウト設定(1ms x 1000)

【注】 1. 本サンプルプログラムを EFM50-0KF0A023A 以外のエンコーダで動作させる場合は、接続したエンコーダに合わせて、設定値を変更してください。
2. 詳細は「HIPERFACE DSL® MASTER Integration Manual」を参照してください。

下記に位置情報と回転情報を格納する仕組みを示します。

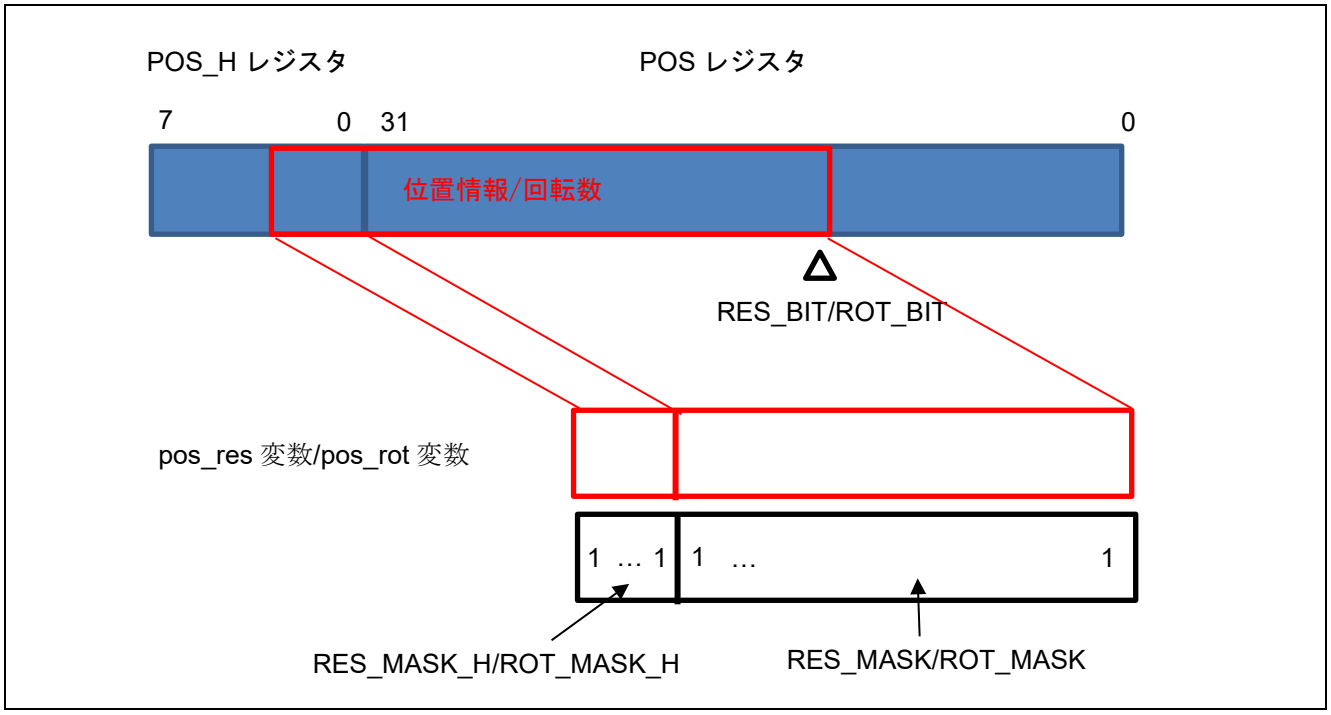

図 4-3　位置情報と回転情報を格納する仕組み

4.11.4 メイン処理のフローチャート

以下に主要な処理を行うものについてフローチャートを記載します。

図中の※がついている処理は別途フローチャートに記載しています。

(1) enc_main フローチャート

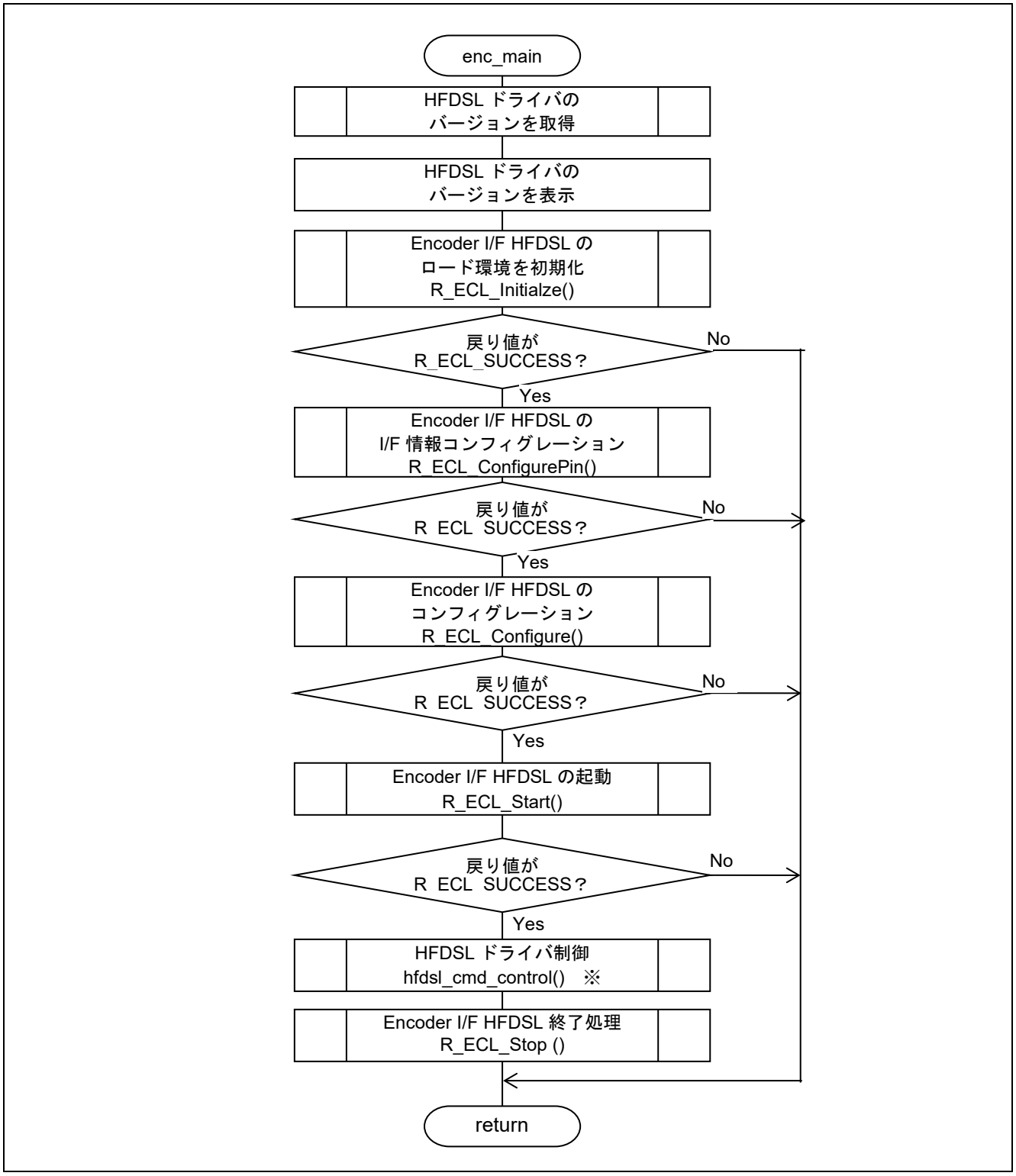

図 4-4　enc_main 関数のフローチャート

(2)　hfdsl_cmd_control フローチャート

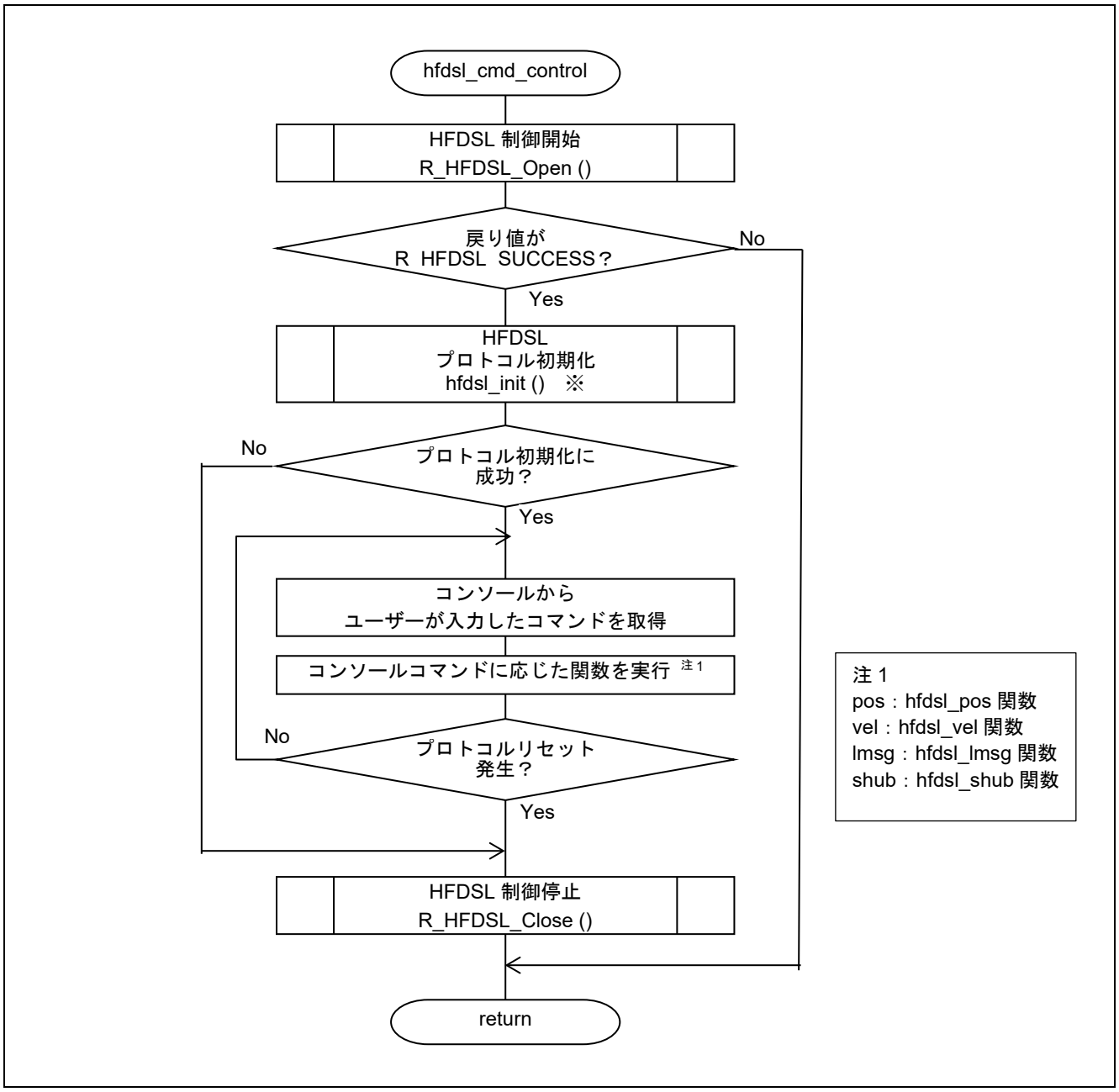

図 4-5　hfdsl_cmd_control 関数のフローチャート

(3)　hfdsl_init フローチャート

プロトコルの初期化を行います。

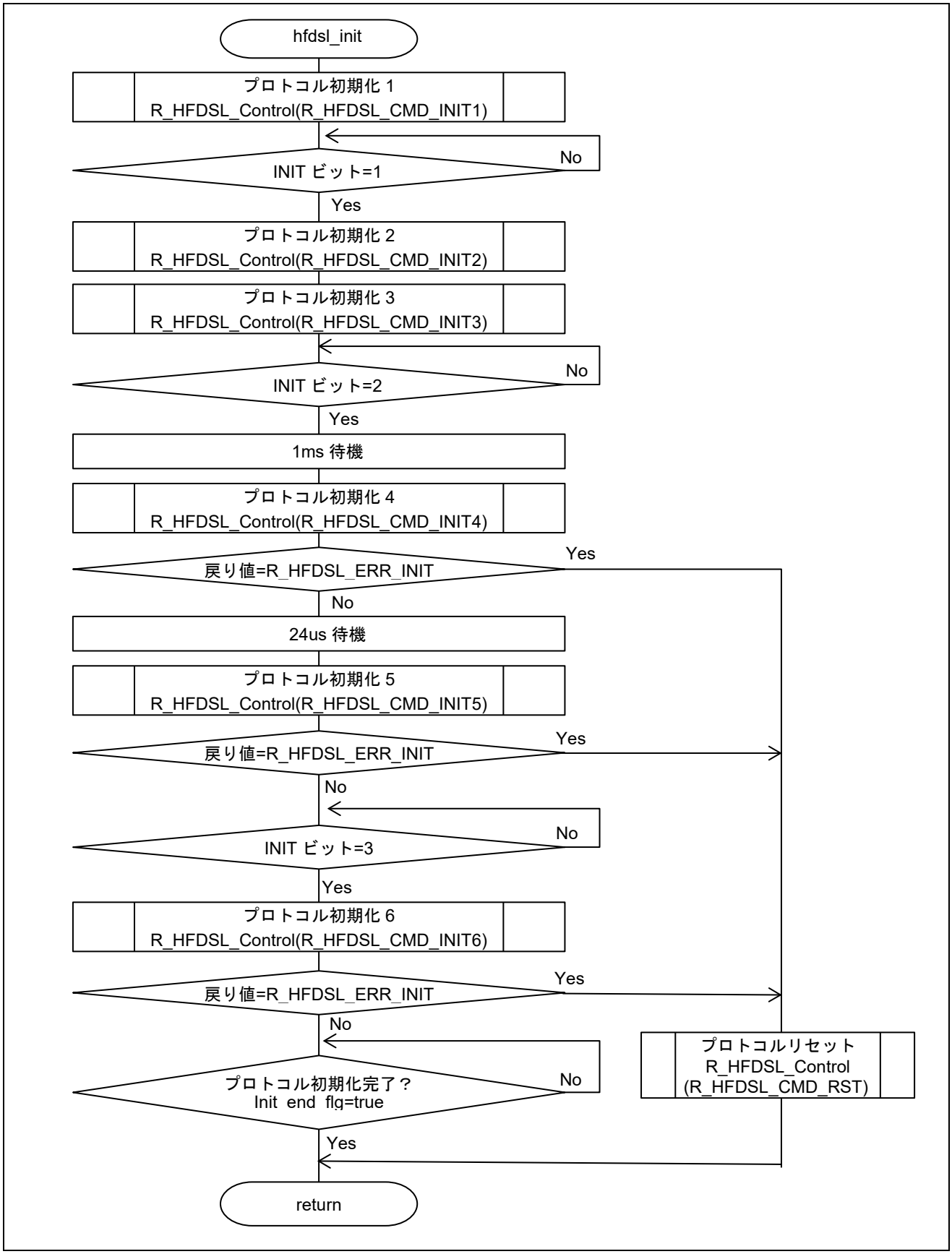

図 4-6　hfdsl_init 関数のフローチャート

(4)　hfdsl_pos、hfdsl_vel、hfdsl_shub フローチャート

コンソールコマンド “pos”, “vel”, “shub” を入力した際にそれぞれ実行され、取得したデータを表示する関数です。各コンソールコマンドと対応する関数と表示内容を以下に示します。

表 4-8　コンソールコマンド “pos”, “vel”, “shub” に対応する関数

コンソールコマンド	対応する関数	表示内容
pos	hfdsl_pos 関数	pos_rot、pos_res vpos_rot、vpos_res err_info
vel	hfdsl_vel 関数	vel、err_info
shub	hfdsl_shub 関数	shub、err_info

hfdsl_pos 関数、hfdsl_vel 関数、hfdsl_shub 関数のフローチャートは同様の処理のため、同一のフローチャートに記載します。

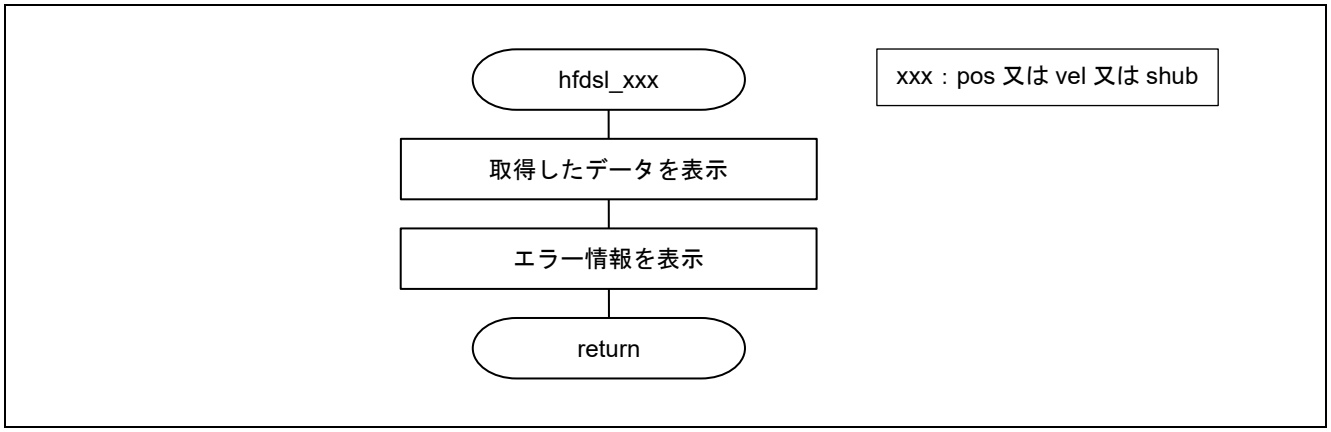

図 4-7　hfdsl_pos 関数、hfdsl_vel 関数、hfdsl_shub 関数のフローチャート

(5)　hfdsl_lmsg フローチャート

コンソールコマンド”lmsg”入力時に実行される関数です。

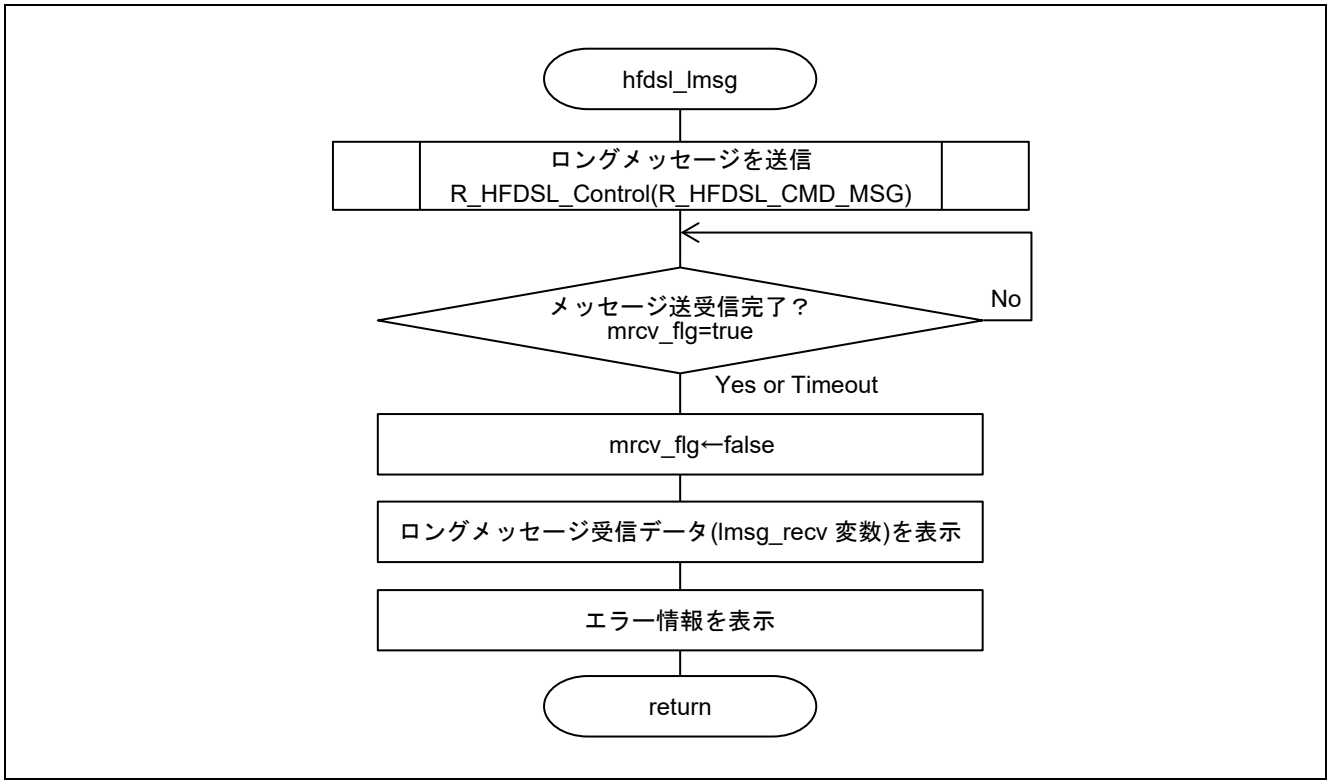

図 4-8　hfdsl_lmsg 関数のフローチャート

(6)　hfdsl_int_nml_callback フローチャート

INT_nml 割り込み発生時にコールされるコールバック関数です。

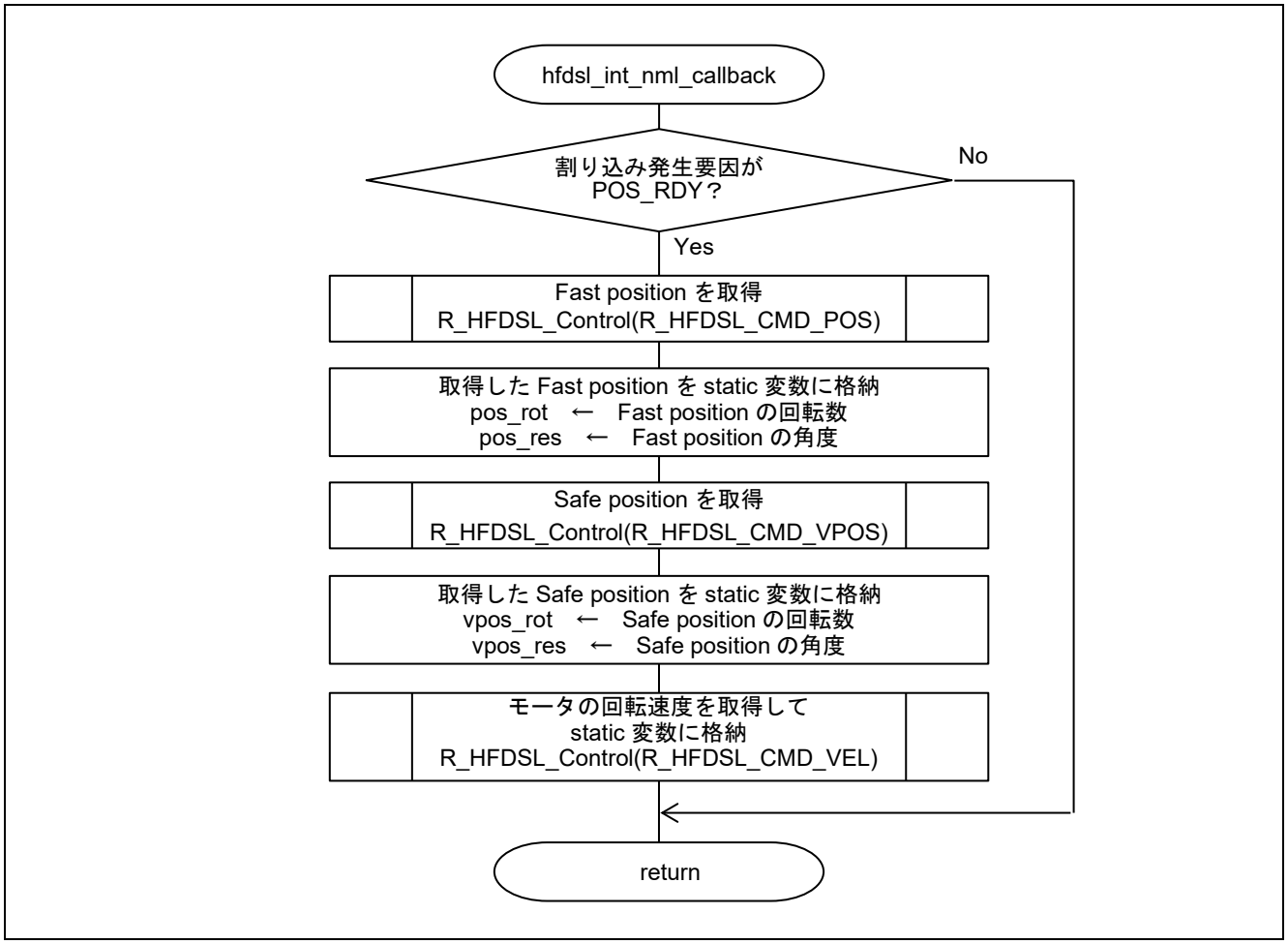

図 4-9　hfdsl_int_nml_callback 関数のフローチャート

(7)　hfdsl_int_err_callback フローチャート

INT_err 割り込み発生時にコールされるコールバック関数です。

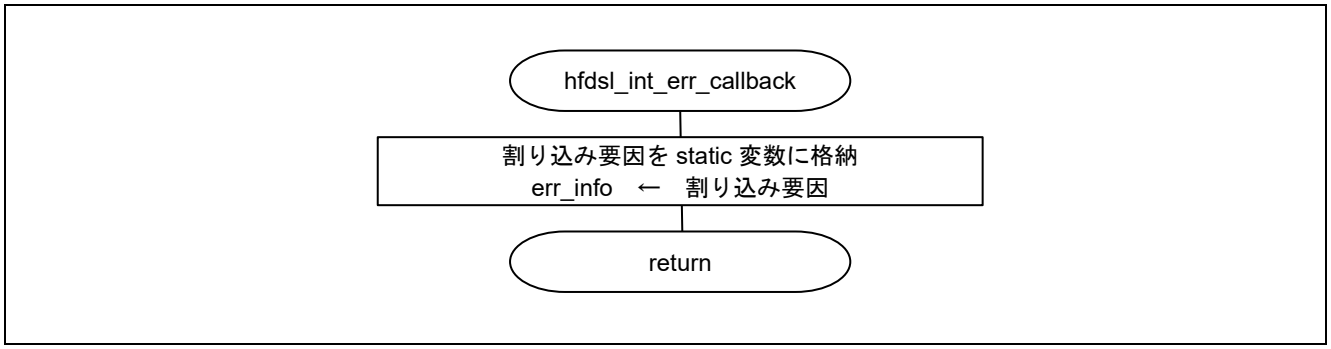

図 4-10　hfdsl_int_err_callback 関数のフローチャート

(8)　hfdsl_int_raw_callback フローチャート

INT_tovr 割り込み発生時にコールされるコールバック関数です。

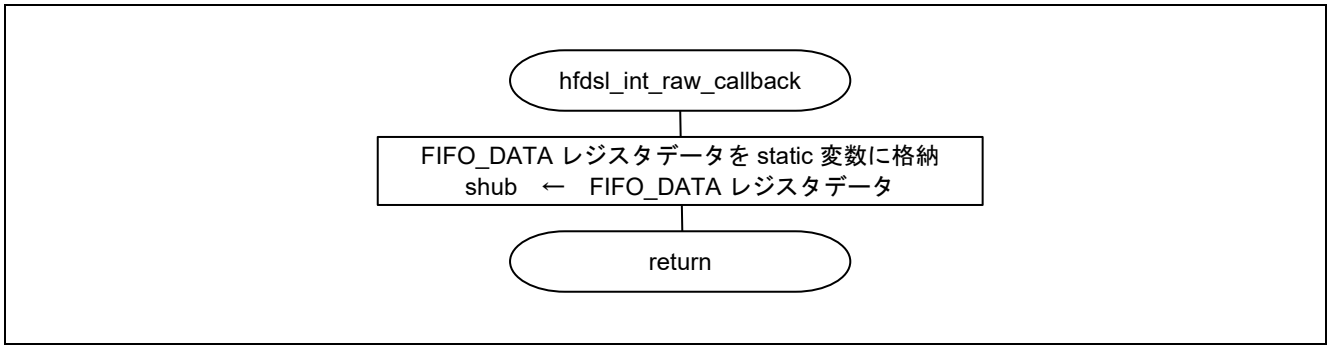

図 4-11　hfdsl_int_raw_callback 関数のフローチャート

(9)　hfdsl_int_mrcv_callback フローチャート

INT_tovr 割り込みが発生し、受信メッセージのデータ格納が完了した時にコールされるコールバック関数です。

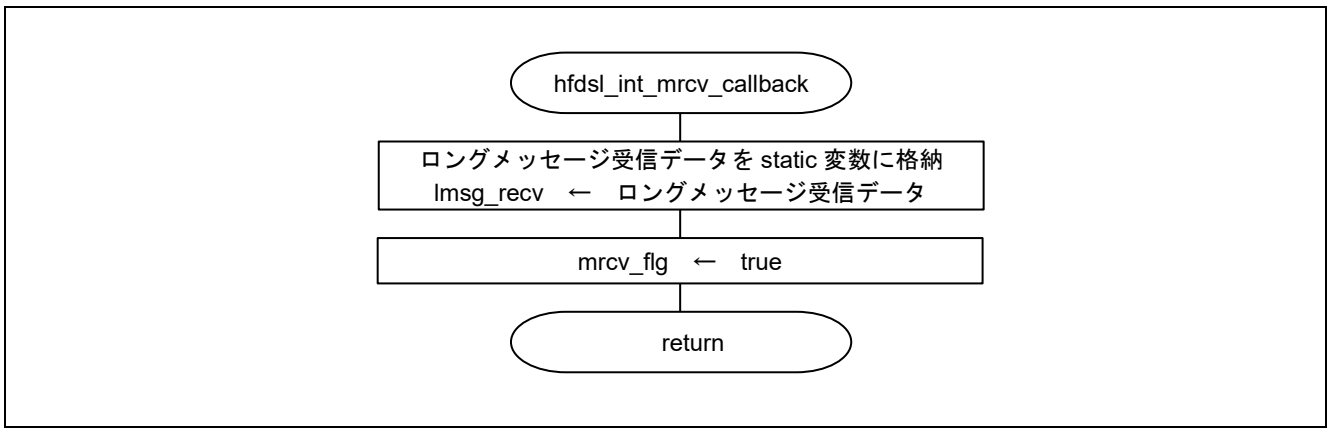

図 4-12　hfdsl_int_mrcv_callback 関数のフローチャート

(10) hfdsl_int_init_callback フローチャート

EVENT レジスタの INIT_END ビットによる、INT_nml 割り込み発生時にコールされるコールバック関数です。

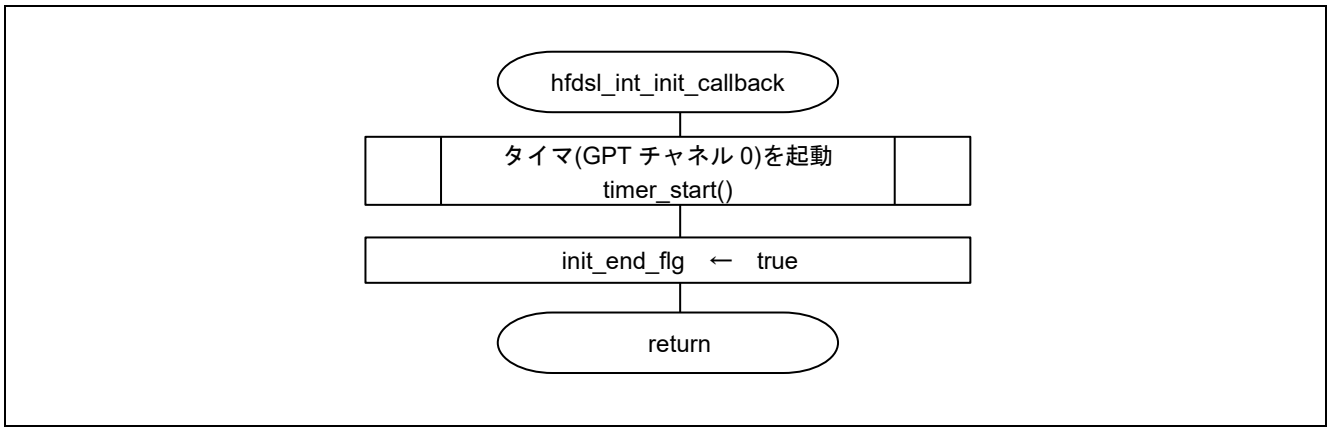

図 4-13　hfdsl_int_init_callback 関数のフローチャート

4.11.5 動作シーケンス

(1) 開始シーケンス

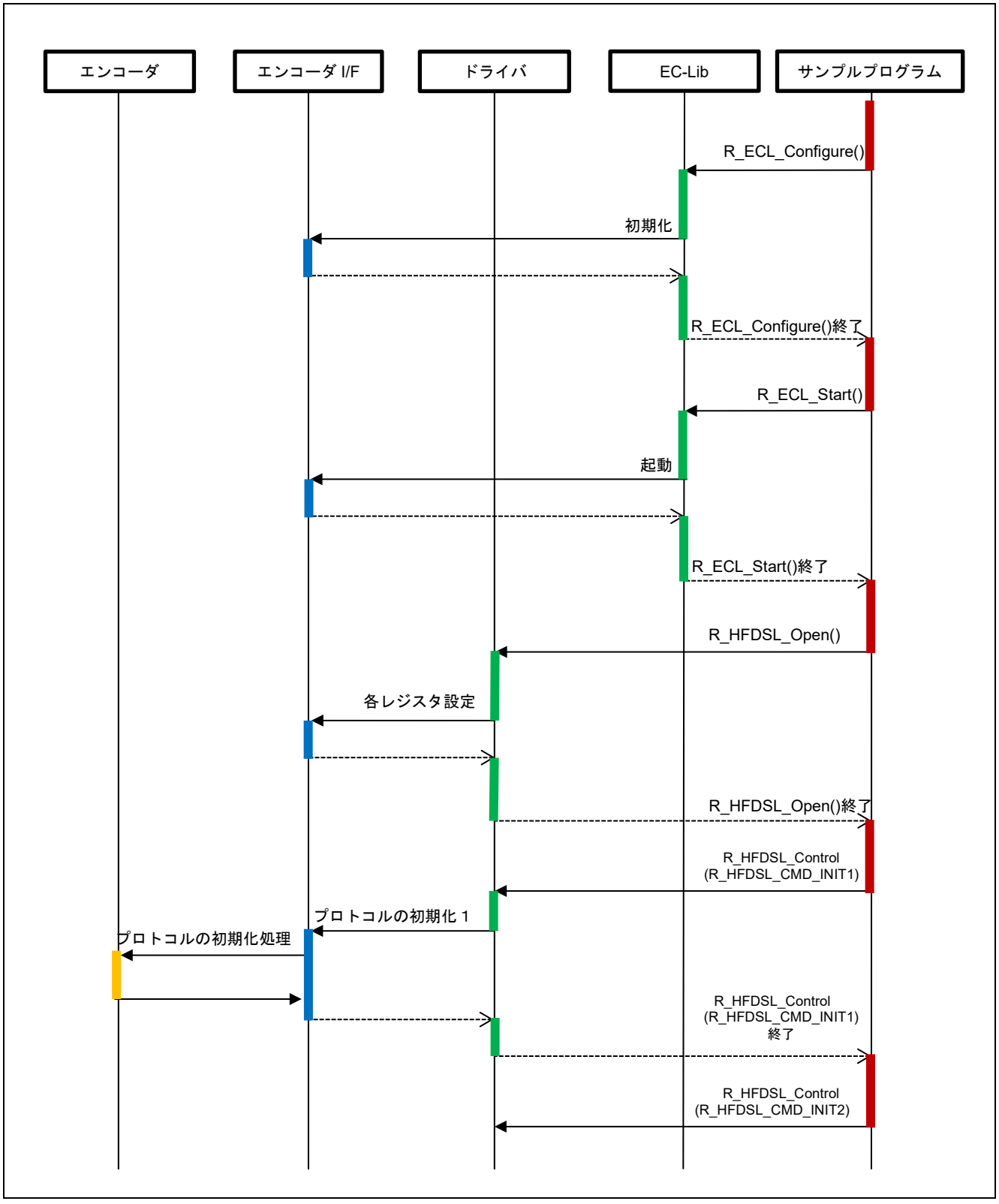

図 4-14　開始シーケンス図(1/3)

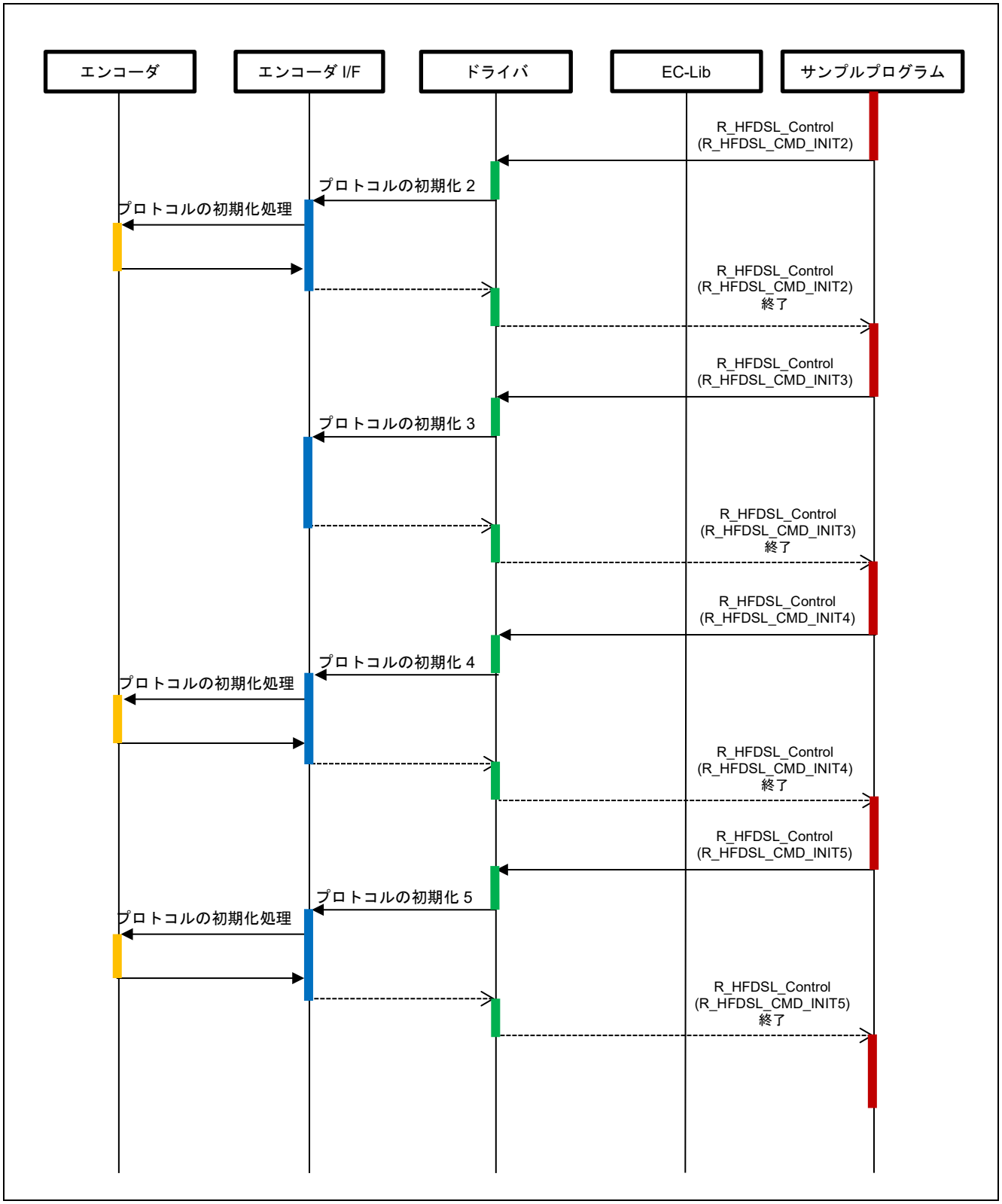

図 4-15　開始シーケンス図(2/3)

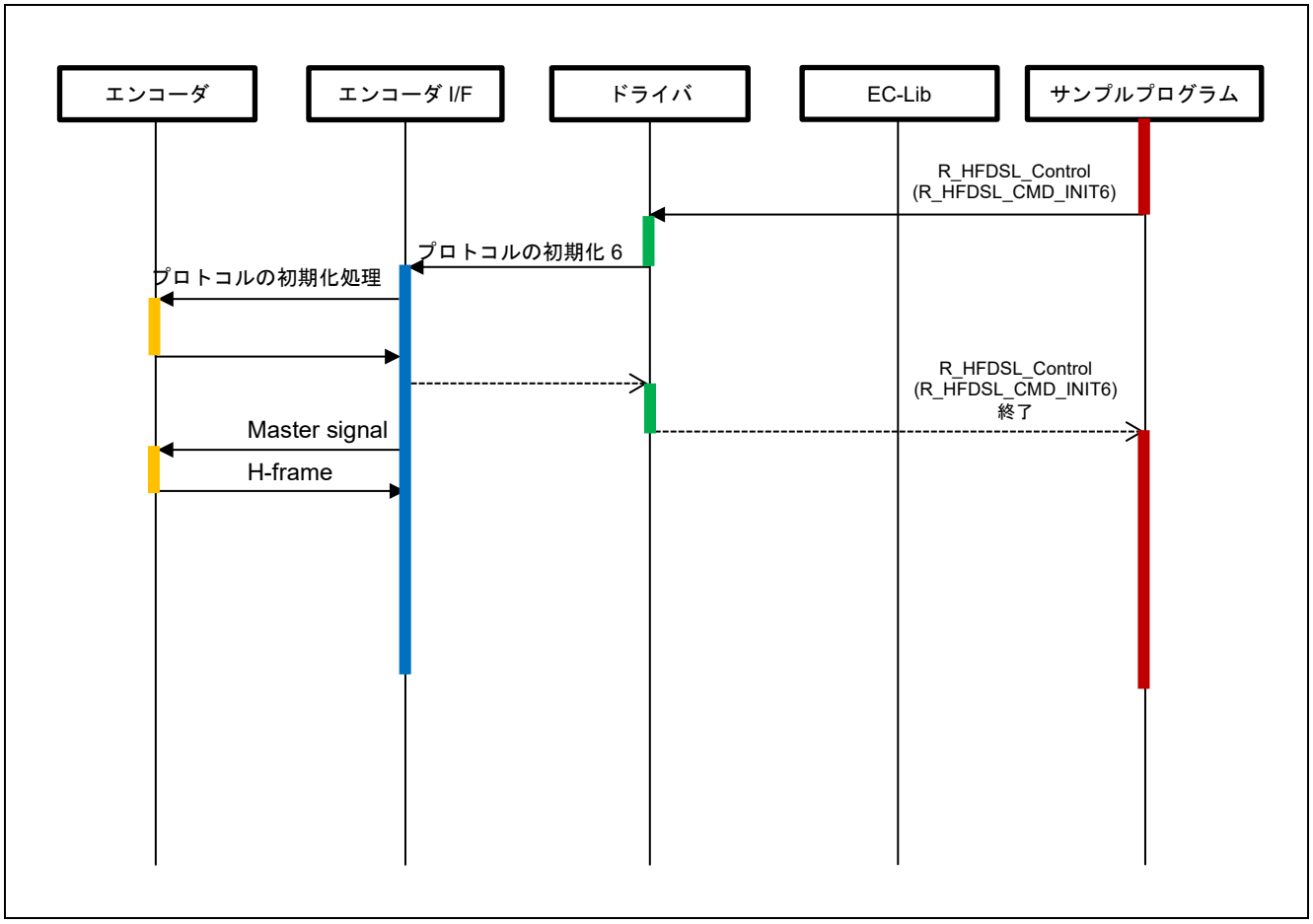

図 4-16 開始シーケンス図(3/3)

(2)　SYNC モードの Fast position 取得シーケンス

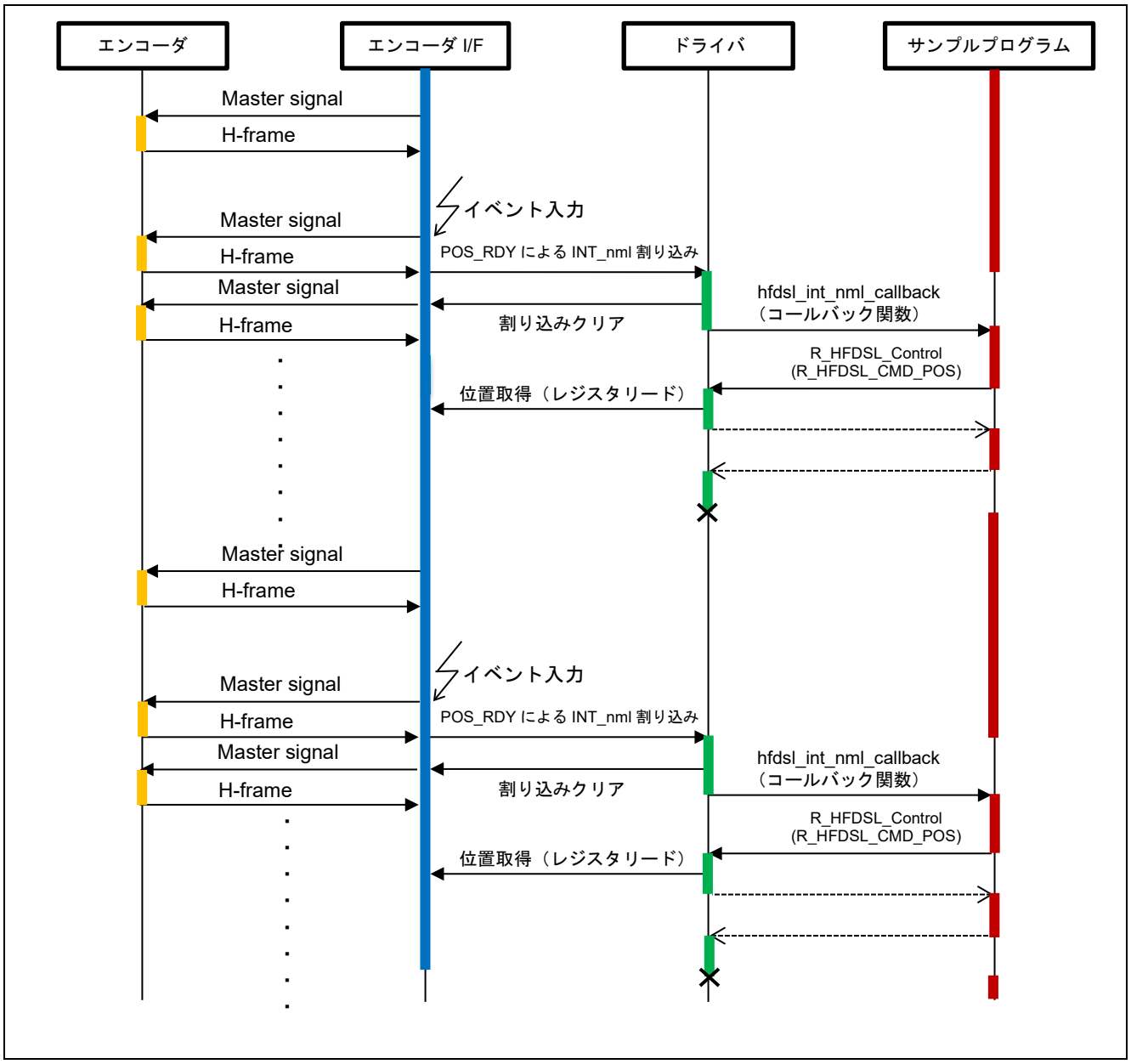

図 4-17　SYNC モードの Fast position 取得シーケンス図

(3)　Sensor Hub channel データ取得シーケンス

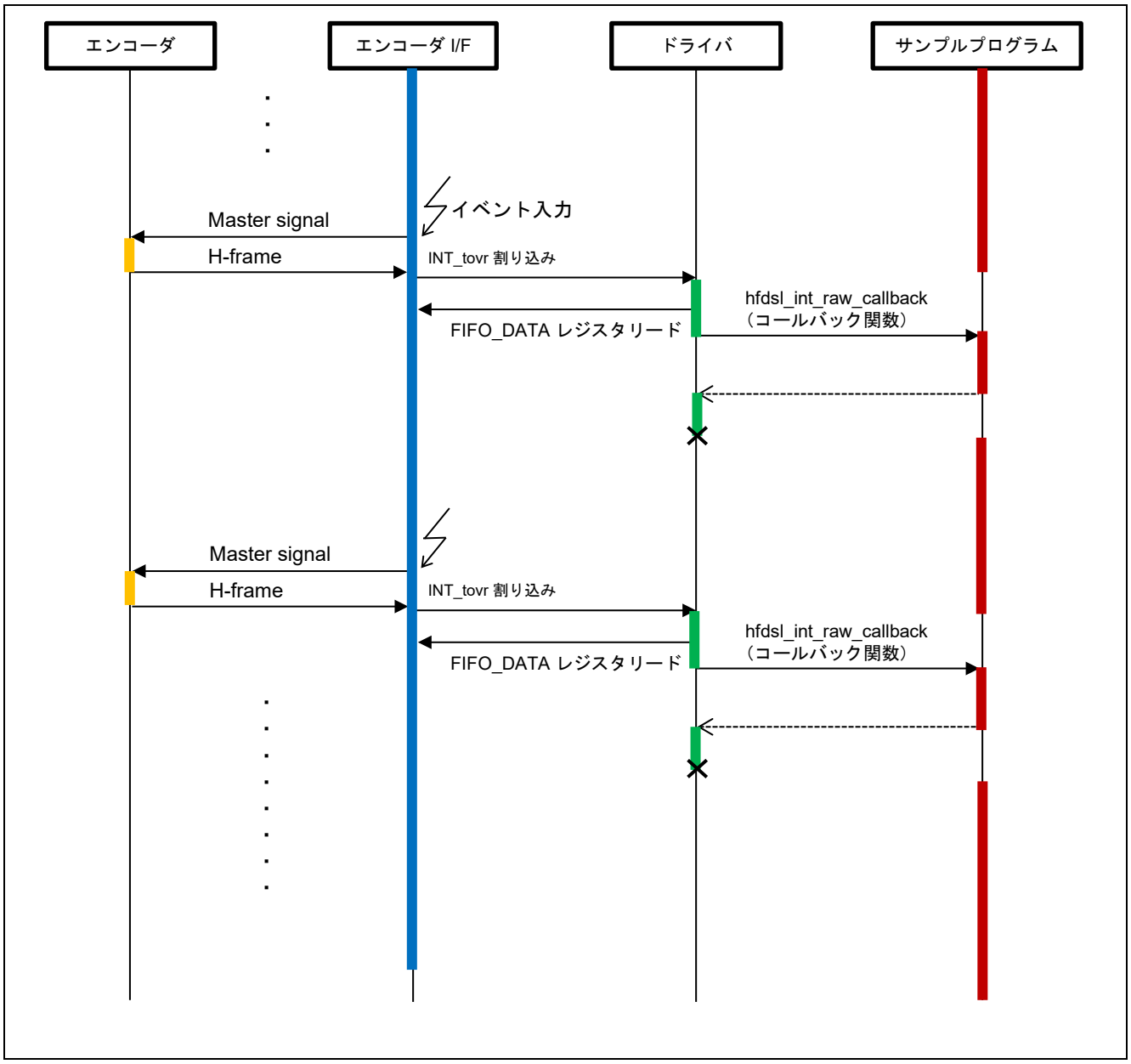

図 4-18　Sensor Hub channel データ取得シーケンス図

(4)　メッセージ送受信シーケンス

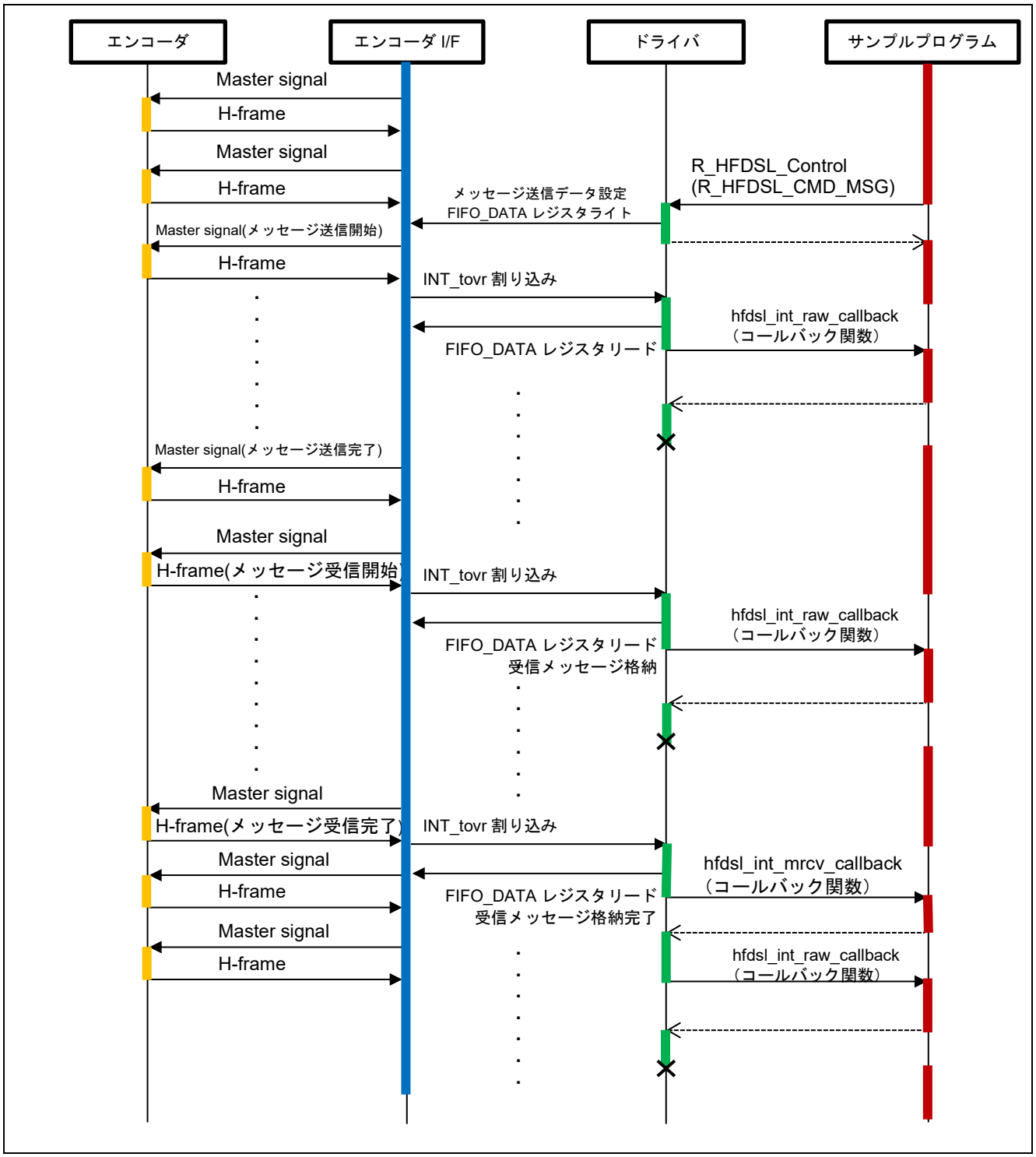

図 4-19　メッセージ送受信シーケンス図

(5)　停止シーケンス

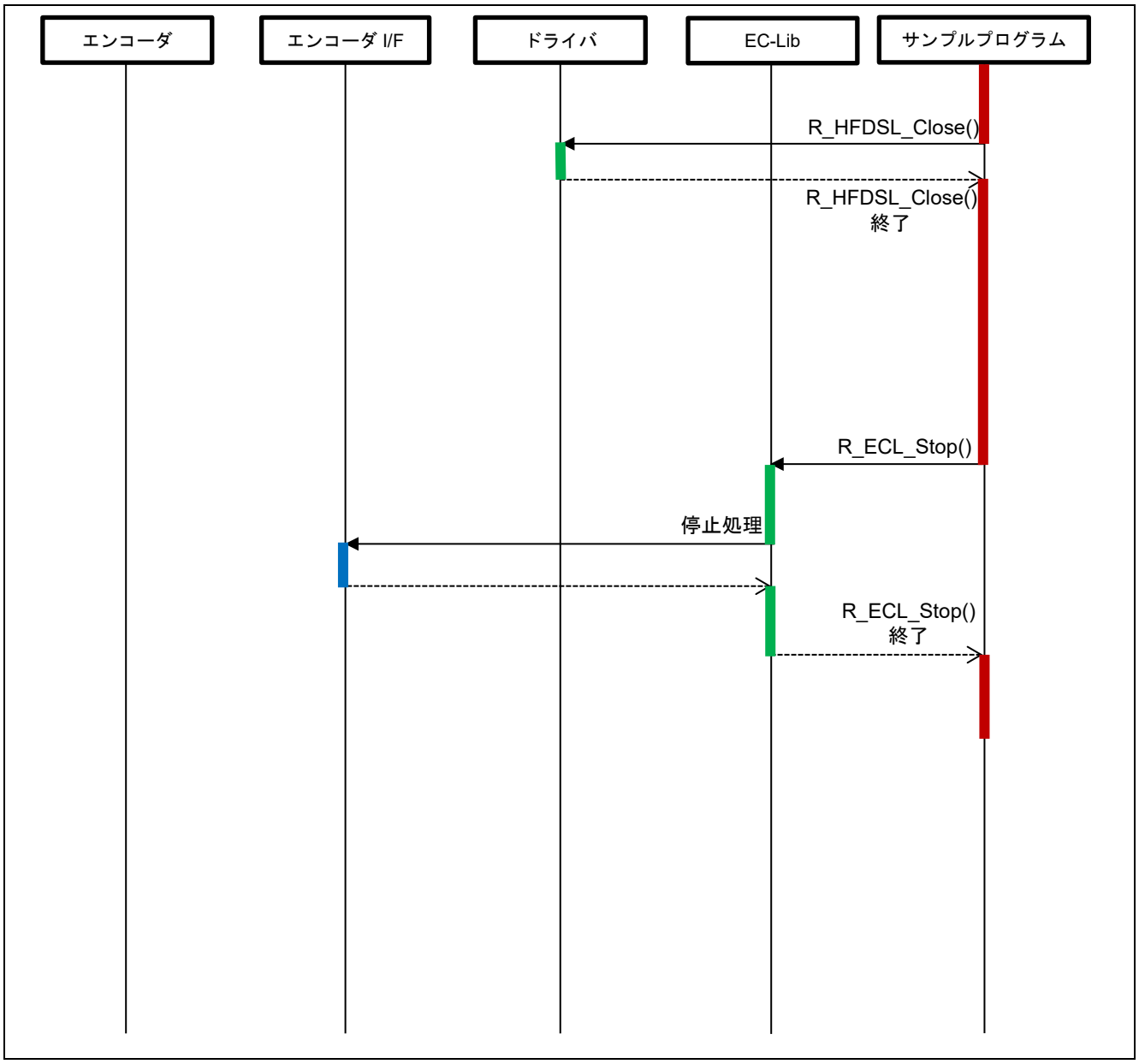

図 4-20　停止シーケンス図

4.11.6 コンソールコマンド

コンソールから入力可能なコマンドは以下となります。

表 4-9 コンソールコマンド一覧

コマンド	内容
pos	Fast position、Safe position を表示します。
vel	モータの回転速度を表示します。
lmsg	エンコーダの Resources の Type of encoder をロングメッセージで取得します。
shub	センサーハブデータを含む一連の受信データを表示します。

(1) サンプルプログラム実行

プログラムを実行すると、バージョンに続いてコマンドプロンプトが表示されます。”hfdsl >”に続けてコマンドを入力してください。

```
HFDSL sample program start
R_HFDSL_GetVersion = 4.0


hfdsl >
```

(2) コマンド実行例

pos コマンドを実行した例です。エンコーダからの応答に基づき、Fast position, Safe position, エラー情報が表示されます。

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

5. サンプルコード

サンプルコードは、ルネサス エレクトロニクスホームページから入手してください。

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Apr.08.22	--	初版発行
1.10	Jun.28.22	5,6,44	動作電圧記載を修正 使用デバイス記載を修正 使用端子一覧記載を修正 メモリサイズ記載削除
1.20	Nov.01.22	7 19 20,21,22 23 25 30 35 37 42 43	tovr 割り込みの記載を追加 hfdsl_int_raw_callback 関数と、hfdsl_int_mrcv_callback 関数が INT_tovr 割り込みで呼び出されるように、説明を変更 INT_tovr 割り込みの記載を追加。割り込みハンドラから呼ばれるコールバック関数の説明を変更 表 4-3 から R_HFDSL_TH_RAW を削除 構造体 r_hfdsl_info_t の要素 pcb_raw の説明を変更 表 4-7 主要な定数に、FIFO_DEP_MAX, TIMEOUT_UNIT, TIMEOUT_COUNT を追加 図 4-8 判定条件の誤記を修正。Timeout 条件を追加 4.11.4(8) hfdsl_int_raw_callback フローチャートを追加 4.11.5(3) Sensor Hub channel データ取得シーケンスを追加 図 4-19 メッセージ送受信シーケンス図の記載を変更
2.00	Jun 07.24	5 24 37 - 43	使用ボードの表記を更新 固定幅整数の定義場所に関する記載を削除 シーケンス図の表記を更新
3.00	Oct 17.25	1, 4, 5 4, 27 8 - 15 42	商標の説明の記載方法を更新 HIPERFACE DSL MASTER Integration Manual の注記を更新 id 引数に関する説明を追加 コマンド実行例を追加
4.00	Apr 03.26	8 - 17, 22 27	ポインタ変数のプレフィクスを”p_”に変更 図 4-3 位置情報の変数名を修正

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。
5. クロックについて
   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
8. 製品間の相違について
   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

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

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。回路、ソフトウェアおよびこれらに関連する情報を使用する場合、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等
   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等
   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.5.0-1　2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。

© 2026 Renesas Electronics Corporation. All rights reserved.