RZ/N2H グループ

Dual Encoder 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/N2H の Encoder Interface を使用し、A-format™通信プロトコル仕様 Version 2.0（以下 A-format™ V2 仕様）に準拠したエンコーダと、EnDat 2.2 仕様に準拠したエンコーダの、異なるエンコーダ 2 種類を同時に接続し、それぞれの情報を取得・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。
・A-format™ V2 仕様に準拠したエンコーダと、EnDat 2.2 仕様に準拠したエンコーダとを同時に接続
・A-format™ V2 のコマンドコードに対応
・A-format™ V2 仕様に準拠したエンコーダ(Nikon 社製 MAR-M50A)から、角度情報等を取得
・EnDat 2.2 のモードコマンド、MRS コードに対応
・EnDat 2.2 仕様に準拠したエンコーダ(HEIDENHAIN 社製 EQN1035)から、角度情報等を取得

動作確認デバイス

RZ/N2H

目次

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
Multi-Protocol Encoder I/F	13 チャネルのエンコーダ I/F を備える。A-format V2 仕様のモジュールと EnDat 2.2 仕様のモジュールとをチャネル毎に選択することで、それぞれのエンコーダと通信を行う
割り込みコントローラ(ICU)	Multi-Protocol Encoder I/F 割り込み制御
汎用 PWM タイマ(GPT)	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 が出力するイベントと Multi-Protocol Encoder I/F 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

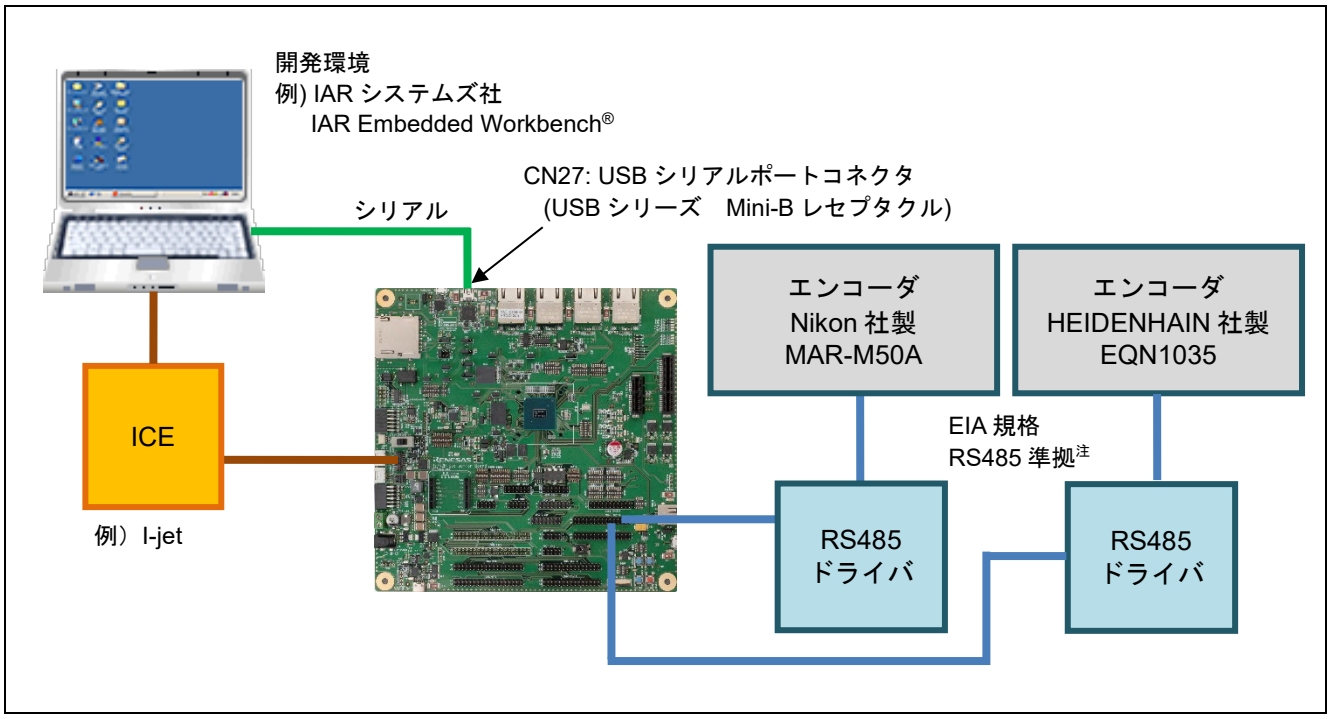


図 1-1　動作環境

【注】　送受信可能なケーブル長は、エンコーダの製造元に問い合わせてください。

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/N2H グループ
動作周波数 注 1	CR52 版	CPUCLK = 1000 MHz (Cortex®-R52 CPU0)
	CA55 版	CPUCLK = 1200 MHz (Cortex®-A55 Core0)
動作電圧		0.8V(Core) / 1.1V (DDR) / 1.8V (PLL, etc.) / 3.3V (I/O)
統合開発環境 注 2		IAR システムズ製　IAR Embedded Workbench® for Arm® RENESAS 製　e² studio
使用ボード		RZ/N2H Evaluation Board（RTK9RZN2Hxxxxxxxxx）
使用デバイス（ボード上で使用する機能）		なし

【注】 1. サンプルプログラムには、CPU コア Cortex-R52 で動作する CR52 版と、CPU コア Cortex-A55 で動作する CA55 版があります。CR52 版、CA55 版はそれぞれの版に関する説明です。
2. 統合開発環境のバージョンは、RZ/N2H グループ Encoder I/F Dual Encoder 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/N2H 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 に使用端子と機能を示します。

表 3.1 使用端子と機能 注

チャネル	端子名	I/O ポート	入出力	I/O 電源ドメイン	内容
AFMT0	ENCIFOE00 (D/R)	P14_3	出力	VDD33	データ出力許可設定
	ENCIFDO00 (REQ)	P14_4	出力	VDD33	データ出力
	ENCIFDI00 (SDAT)	P14_5	入力	VDD33	データ入力
ENDAT_CH1	ENCIFCK01 (TCLK1)	P02_0	出力	VDD1833_5	クロック出力
	ENCIFOE01 (DE1)	P02_1	出力	VDD1833_5	データ出力許可設定
	ENCIFDO01 (DATA_DV1)	P26_7	出力	VDD33	データ出力
	ENCIFDI01 (DATA_RC1)	P27_0	入力	VDD33	データ入力

【注】 A-format と EnDat 2.2 のチャネルを変更する方法は、4.15.11 チャネル変更手順 を参照してください。

4. ソフトウェア説明

4.1 A-format ドライバ機能

A-format ドライバの機能は以下です。

1. 初期設定
2. コマンドコードの送信
3. 受信データの取得

4.2 EnDat ドライバ機能

EnDat ドライバの機能は以下です。

1. 初期設定
   A) エンコーダの初期化 (バッテリー付きエンコーダは未対応)
   B) 伝送遅延補正の設定
2. リクエスト情報の送信
   A) モードコマンド
   B) MRS コード
   C) パラメータ
3. エンコーダデータの受信
   A) 位置値
   B) パラメータ
   C) 付加情報 注

【注】 本書では Additional Information 1 と Additional Information 2 を「付加情報」として表示しています。詳細は HEIDENHAIN 社に問い合わせることで入手可能な「EnDat Specification」を参照してください。

4.3 ファイル構成

ファイル構成は、RZ/N2H グループ Encoder I/F Dual Encoder sample program リリースノートを参照してください。

4.4 関数一覧

表 4.1 に A-format 関数一覧を、表 4.2 に EnDat 関数一覧を示します。

表 4.1 A-format 関数一覧

表 4.2　EnDat 関数一覧

4.5 A-format API 関数仕様

4.5.1 R_A_AS_Open

R_A_AS_Open	
概　要	エンコーダ制御の開始
ヘッダ	r_a_as_rzt2_if.h
宣　言	int32_t R_A_AS_Open(const int32_t id, r_a_as_info_t* p_info);
説　明	AFMT の初期設定を行います。 1 A-format Interface のパワーダウン設定解除 2 エンコーダインタフェースの設定 3 通信パラメータ(BRSEL レジスタ)の設定 4 割り込みイネーブルの設定
引　数	id ：使用する ID を指定します。(r_a_as_rzt2_dat.h で定義されています。) R_A_AS0_ID ：チャネル 0 を指定 R_A_AS1_ID ：チャネル 1 を指定 ： ： R_A_AS15_ID ：チャネル 15 を指定 上記以外 ：設定不可 p_info ：エンコーダの情報を設定します。 エンコーダの情報を格納した構造体 r_a_as_info_t のアドレスを指定してください。
リターン値	R_A_AS_SUCCESS：正常終了 R_A_AS_ERR_INVALID_ARG：異常終了 (id, p_info に指定した r_a_as_info_t 構造体のメンバ変数が無効、または規定されていない値) R_A_AS_ERR_ACCESS：異常終了（既に open されています）
注　意	本関数内でエンコーダ I/F の設定を行います。 エンコーダの電源投入後に本関数を実行した場合は、本関数の実行後に CDF8 を連続して 8 回エンコーダに送信し、ステータスフラグをクリアしてください。 本サンプルプログラムでは、チャネル 8, チャネル 12, チャネル 15 は無効です。 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。

4.5.2 R_A_AS_Close

R_A_AS_Close	
概　要	エンコーダの制御を終了
ヘッダ	r_a_as_rzt2_if.h
宣　言	int32_t R_A_AS_Close(const int32_t id);
説　明	指定されたチャネルの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_a_as_rzt2_dat.h で定義されています。) R_A_AS0_ID ：チャネル 0 を指定 R_A_AS1_ID ：チャネル 1 を指定 ： ： R_A_AS15_ID ：チャネル 15 を指定 上記以外 ：設定不可
リターン値	R_A_AS_SUCCESS：正常終了 R_A_AS_ERR_INVALID_ARG：異常終了 (id に指定した値が無効、または規定されていない値) R_A_AS_ERR_ACCESS：異常終了 (リクエストを送信中です。)
注　意	本サンプルプログラムでは、チャネル 8, チャネル 12, チャネル 15 は無効です。 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。

4.5.3 R_A_AS_GetVersion

R_A_AS_GetVersion	
概　要	エンコーダ IF ドライバのバージョンを取得
ヘッダ	r_a_as_rzt2_if.h
宣　言	uint32_t R_A_AS_GetVersion(const r_a_as_type_t type);
説　明	A-format ドライバのバージョンを取得します。
引　数	type　：R_A_AS_A_FORMAT を指定してください。
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）戻り値が 0x00010002 の場合、Ver.1.2
補　足	上記以外の type が指定された場合、戻り値は 0xFFFFFFFF となります。
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。

4.5.4 R_A_AS_Control

R_A_AS_Control	
概　要	エンコーダの制御
ヘッダ	r_a_as_rzt2_if.h
宣　言	int32_t R_A_AS_Control(const int32_t id, const r_a_as_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 4.5.5 A-format 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_a_as_rzt2_dat.h で定義されています。) R_A_AS0_ID　：チャネル 0 を指定 R_A_AS1_ID　：チャネル 1 を指定 ：　： R_A_AS15_ID　：チャネル 15 を指定 上記以外　：設定不可 cmd　：コマンド 内容は「表 4.10　R_A_AS_Control 関数の制御コマンド」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_A_AS_SUCCESS：正常終了 R_A_AS_ERR_INVALID_ARG：異常終了（id, cmd が無効、または規定されていない値） その他リターン値は 4.5.5 A-format 制御コマンド、表 4.10　R_A_AS_Control 関数の制御コマンドを参照してください。
注　意	本関数実行前に、必ず R_A_AS_Open を実行してください。 本サンプルプログラムでは、チャネル 8, チャネル 12, チャネル 15 は無効です。 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。

4.5.5　A-format 制御コマンド

(1)　R_A_AS_CMD_SET_PARAM

R_A_AS_CMD_SET_PARAM	
概　要	リクエスト情報を設定
ヘッダ	r_a_as_rzt2_if.h
宣　言	int32_t R_A_AS_Control(const int32_t id, const r_a_as_cmd_t cmd, void *const p_buf);
説　明	リクエスト情報を設定します。
引　数	id　：使用する ID を指定します。(r_a_as_rzt2_dat.h で定義されています。) R_A_AS0_ID　：チャネル 0 を指定 R_A_AS1_ID　：チャネル 1 を指定 ：　： R_A_AS15_ID　：チャネル 15 を指定 上記以外　：設定不可 cmd　：R_A_AS_CMD_SET_PARAM を指定します。 p_buf　：リクエスト情報 リクエスト情報を格納した r_a_as_req_t 構造体のポインタを指定します。詳細は「4.14.1(2) r_a_as_req_t」を参照してください。
リターン値	R_A_AS_SUCCESS：正常終了 R_A_AS_ERR_INVALID_ARG：異常終了（id が無効、または規定されていない値、p_buf が NULL、p_buf に指定された r_a_as_req_t 構造体のメンバ変数が規定されていない値） R_A_AS_ERR_ACCESS：異常終了（該当チャネルが開始されていません）
注意	本制御コマンドは、リクエスト情報の設定のみを行います。 エンコーダへコマンド送信するには、以下の制御コマンドをご使用ください。 ・R_A_AS_CMD_TX_TRG ・R_A_AS_CMD_TX_ELC 本サンプルプログラムでは、チャネル 8, チャネル 12, チャネル 15 は無効です。 コールバック関数内で、本制御コマンドを実行することは禁止します。

(2)　R_A_AS_CMD_ELC_DISABLE

R_A_AS_CMD_ELC_DISABLE	
概　要	ELC イベント入力トリガの無効
ヘッダ	r_a_as_rzt2_if.h
宣　言	int32_t R_A_AS_Control(const int32_t id, const r_a_as_cmd_t cmd, void *const p_buf);
説　明	ELC イベント入力トリガを無効にします。
引　数	id　：使用する ID を指定します。(r_a_as_rzt2_dat.h で定義されています。) R_A_AS0_ID　：チャネル 0 を指定 R_A_AS1_ID　：チャネル 1 を指定 ：　： R_A_AS15_ID　：チャネル 15 を指定 上記以外　：設定不可 cmd　：R_A_AS_CMD_ELC_DISABLE を指定します。 p_buf　：使用しません（NULL を指定してください）
リターン値	R_A_AS_SUCCESS：ELC イベント入力トリガを無効にしました。 R_A_AS_ERR_INVALID_ARG：異常終了（id が無効、または規定されていない値） R_A_AS_ERR_ACCESS：異常終了（ELC イベント入力トリガ動作中ではない、または、該当チャネルが開始されていません）
注　意	本サンプルプログラムでは、チャネル 8, チャネル 12, チャネル 15 は無効です。コールバック関数内で、本制御コマンドを実行することは禁止します。

(3) R_A_AS_CMD_TX_TRG

R_A_AS_CMD_TX_TRG	
概　要	エンコーダへのリクエスト送信を開始
ヘッダ	r_a_as_rzt2_if.h
宣　言	int32_t R_A_AS_Control(const int32_t id, const r_a_as_cmd_t cmd, void *const p_buf);
説　明	エンコーダへのリクエスト送信を開始します。 通常受信時はリクエスト送信 1 回に対して、割り込み許可中の割り込み要因に対応したコールバック関数が 1 回ずつコールされます。コールバック関数の詳細は、「4.6.1 a_as_txerr_callback」～「4.6.3 a_as_rxend_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_a_as_rzt2_dat.h で定義されています。) R_A_AS0_ID　：チャネル 0 を指定 R_A_AS1_ID　：チャネル 1 を指定 ：　： R_A_AS15_ID　：チャネル 15 を指定 上記以外　：設定不可 cmd　：R_A_AS_CMD_TX_TRG を指定します。 p_buf　：使用しません（NULL を指定してください）
リターン値	R_A_AS_SUCCESS：正常終了 R_A_AS_ERR_INVALID_ARG：異常終了（id が無効、または規定されていない値） R_A_AS_ERR_BUSY：異常終了（送信処理中） R_A_AS_ERR_ACCESS：異常終了（該当チャネルが開始されていません）
注　意	本制御コマンドは、エンコーダへのリクエスト送信の開始のみを行います。 制御コマンド R_A_AS_CMD_SET_PARAM でリクエスト情報を設定してからご使用ください。 本サンプルプログラムでは、チャネル 8, チャネル 12, チャネル 15 は無効です。 コールバック関数内で、本制御コマンドを実行することは禁止します。

(4) R_A_AS_CMD_TX_ELC

R_A_AS_CMD_TX_ELC	
概　要	ELC イベント入力トリガによるエンコーダへのリクエスト送信を開始
ヘッダ	r_a_as_rzt2_if.h
宣　言	int32_t R_A_AS_Control(const int32_t id, const r_a_as_cmd_t cmd, void *const p_buf);
説　明	ELC イベント入力トリガを許可し、エンコーダへのリクエスト送信を開始します。 通常受信時はリクエスト送信 1 回に対して、割り込み許可中の割り込み要因に対応したコールバック関数が 1 回ずつコールされます。コールバック関数の詳細は、「4.6.1 a_as_txerr_callback」～「4.6.3 a_as_rxend_callback」を参照してください。 ELC イベント入力トリガ動作中に本制御コマンドを実行した場合、R_A_AS_ERR_BUSY が発生します。
引　数	id　：使用する ID を指定します。(r_a_as_rzt2_dat.h で定義されています。) R_A_AS0_ID　：チャネル 0 を指定 R_A_AS1_ID　：チャネル 1 を指定 ：　： R_A_AS15_ID　：チャネル 15 を指定 上記以外　：設定不可 cmd　：R_A_AS_CMD_TX_ELC を指定します。 p_buf　：使用しません（NULL を指定してください）
リターン値	R_A_AS_SUCCESS：正常終了 R_A_AS_ERR_INVALID_ARG：異常終了（id が無効、または規定されていない値） R_A_AS_ERR_BUSY：異常終了（送信処理中、ELC イベント入力トリガ動作中） R_A_AS_ERR_ACCESS：異常終了（該当チャネルが開始されていません）
注　意	本制御コマンドは、エンコーダへのリクエスト送信の開始のみを行います。 制御コマンド R_A_AS_CMD_SET_PARAM でリクエスト情報を設定してからご使用ください。 本サンプルプログラムでは、チャネル 8, チャネル 12, チャネル 15 は無効です。コールバック関数内で、本制御コマンドを実行することは禁止します。

4.6 A-format ユーザー定義関数仕様

4.6.1 a_as_txerr_callback

a_as_txerr_callback	
概　要	通常受信でタイムアウトエラー時の送受信結果を通知
ヘッダ	-
宣　言	void a_as_txerr_callback (r_a_as_result_t * p_result);
説　明	R_A_AS_Control(R_A_AS_CMD_SET_PARAM) 関数で登録したコールバック関数です。 通常受信時の送受信結果を通知します。タイムアウトエラー割り込み(PERI_ERR0)が発生した場合にコールされます。本関数を処理後に、a_as_rxend_callback()関数もコールされます。 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 return 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	p_result　：送受信結果 構造体 r_a_as_result_t で宣言した送受信結果が格納されている配列のポインタです。配列の内容は表 4.3　配列の内容と送受信結果の対応を参照してください。 R_A_AS_Control(R_A_AS_CMD_SET_PARAM) 関数で指定したエンコーダアドレスに対応した送受信結果を更新します。 送受信結果の詳細は、表 4.4　送受信結果を参照してください。
リターン値	なし
注　意	ELC イベント入力トリガが有効の場合、タイマインターバル時間内に送受信結果を取得してください。 エンコーダからの応答タイミングによっては、R_A_AS_Close 関数や、R_A_AS_Control (R_A_AS_CMD_ELC_DISABLE) 関数を実行した後でも、本コールバック関数が呼ばれることがあります。

4.6.2 a_as_rxset_callback

a_as_rxset_callback	
概　要	通常受信で受信データ設定完了時の送受信結果を通知
ヘッダ	-
宣　言	void a_as_rxset_callback(r_a_as_result_t * p_result);
説　明	R_A_AS_Control (R_A_AS_CMD_SET_PARAM)関数で登録したコールバック関数です。 通常受信時の送受信結果を通知します。受信完了割り込み(AFMTi_EOF)が発生した場合にコールされます。本関数を処理後に、a_as_rxend_callback()関数もコールされます。 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 return 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	p_result　：送受信結果 構造体 r_a_as_result_t で宣言した送受信結果が格納されている配列のポインタです。 配列の内容は表 4.3　配列の内容と送受信結果の対応を参照してください。 R_A_AS_Control (R_A_AS_CMD_SET_PARAM) 関数で指定したエンコーダアドレスに対応した送受信結果を更新します。 送受信結果の詳細は、表 4.4　送受信結果を参照してください。
リターン値	なし
注　意	ELC イベント入力トリガが有効の場合、タイマインターバル時間内に送受信結果を取得してください。 エンコーダからの応答タイミングによっては、R_A_AS_Close 関数や、R_A_AS_Control (R_A_AS_CMD_ELC_DISABLE) 関数を実行した後でも、本コールバック関数が呼ばれることがあります。

4.6.3 a_as_rxend_callback

a_as_rxend_callback	
概　要	通常受信でデータ送受信完了時の送受信結果を通知
ヘッダ	-
宣　言	void a_as_rxend_callback(r_a_as_result_t * p_result);
説　明	R_A_AS_Control(R_A_AS_CMD_SET_PARAM) 関数で登録したコールバック関数です。 通常受信時の送受信結果を通知します。タイムアウトエラー割り込み(PERI_ERR0)や受信完了割り込み(AFMTi_EOF)が発生した場合に、a_as_txerr_callback()関数やa_as_rxset_callback()関数に続いてコールされます。 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 return 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	p_result　：送受信結果 構造体 r_a_as_result_t で宣言した送受信結果が格納されている配列のポインタです。 配列の内容は表 4.3　配列の内容と送受信結果の対応を参照してください。 R_A_AS_Control(R_A_AS_CMD_SET_PARAM) 関数で指定したエンコーダアドレスに対応した送受信結果を更新します。 送受信結果の詳細は、表 4.4　送受信結果を参照してください。
リターン値	なし
注　意	ELC イベント入力トリガが有効の場合、タイマインターバル時間内に送受信結果を取得してください。 エンコーダからの応答タイミングによっては、R_A_AS_Close 関数や、R_A_AS_Control(R_A_AS_CMD_ELC_DISABLE) 関数を実行した後でも、本コールバック関数が呼ばれることがあります。

4.6.4 a_as_elctimer_callback

a_as_elctimer_callback	
概　要	ELC イベント入力による連続受信でデータ送受信結果を通知
ヘッダ	-
宣　言	void a_as_elctimer_callback(r_a_as_result_t * p_result);
説　明	R_A_AS_Control (R_A_AS_CMD_SET_PARAM)関数で登録したコールバック関数です。 通常受信時の送受信結果を通知します。受信完了割り込み(AFMTi_EOF)が発生するたびにコールされます。 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 return 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	p_result　：送受信結果 構造体 r_a_as_result_t で宣言した送受信結果が格納されている配列のポインタです。 配列の内容は表 4.3　配列の内容と送受信結果の対応を参照してください。 R_A_AS_Control (R_A_AS_CMD_SET_PARAM) 関数で指定したエンコーダアドレスに対応した送受信結果を更新します。 送受信結果の詳細は、表 4.4　送受信結果を参照してください。
リターン値	なし
注　意	ELC イベント入力トリガが有効の場合、タイマインターバル時間内に送受信結果を取得してください。 エンコーダからの応答タイミングによっては、R_A_AS_Close 関数や、R_A_AS_Control(R_A_AS_CMD_ELC_DISABLE) 関数を実行した後でも、本コールバック関数が呼ばれることがあります。

表 4.3　配列の内容と送受信結果の対応

配列番号	内容
p_result[0]	エンコーダ区分 ENC1 の送受信結果
p_result[1]	エンコーダ区分 ENC2 の送受信結果
p_result[2]	エンコーダ区分 ENC3 の送受信結果
p_result[3]	エンコーダ区分 ENC4 の送受信結果
p_result[4]	エンコーダ区分 ENC5 の送受信結果
p_result[5]	エンコーダ区分 ENC6 の送受信結果
p_result[6]	エンコーダ区分 ENC7 の送受信結果
p_result[7]	エンコーダ区分 ENC8 の送受信結果

表 4.4　送受信結果

割り込み要因	送受信結果(p_result のメンバ変数)		
	result	data	status
タイムアウトエラー (PERI_ERR0)	コールバック関数内のみ有効	無効	コールバック関数内のみ有効
データ受信完了 (AFMTi_EOF) 注 1	コールバック関数内のみ有効	有効 注 2	コールバック関数内のみ有効

【注】 1. i = 00～15
2. ELC イベント入力トリガが無効の場合、次のリクエスト送信までデータ受信結果は有効です。
ELC イベント入力トリガが有効の場合、次の AFMTi_EOF 割り込みが発生するまでデータ受信結果は有効です。

4.7　A-format 割り込みハンドラ

4.7.1　a_as0_int_isr

a_as0_int_isr	
概　要	AFMT0_EOF 割り込みの受信完了割り込みハンドラ
ヘッダ	-
宣　言	void a_as0_int_isr(void);
説　明	A_AS Ch0 の受信完了割り込みに対する割り込みハンドラです。
引　数	なし
リターン値	なし

4.7.2　a_as1_int_isr

a_as1_int_isr	
概　要	AFMT1_EOF 割り込みの受信完了割り込みハンドラ
ヘッダ	-
宣　言	void a_as1_int_isr(void);
説　明	A_AS Ch1 の受信完了割り込みに対する割り込みハンドラです。注
引　数	なし
リターン値	なし

4.7.3 a_as2_int_isr

a_as2_int_isr	
概　要	AFMT2_EOF 割り込みの割り込みハンドラ
ヘッダ	-
宣　言	void a_as2_int_isr(void);
説　明	A_AS Ch2 の受信完了割り込みに対する割り込みハンドラです。注
引　数	なし
リターン値	なし

4.7.4 a_as3_int_isr

a_as3_int_isr	
概　要	AFMT3_EOF 割り込みの割り込みハンドラ
ヘッダ	-
宣　言	void a_as3_int_isr(void);
説　明	A_AS Ch3 の受信完了割り込みに対する割り込みハンドラです。注
引　数	なし
リターン値	なし

4.7.5 a_as4_int_isr

a_as4_int_isr	
概　要	AFMT4_EOF 割り込みの割り込みハンドラ
ヘッダ	-
宣　言	void a_as4_int_isr(void);
説　明	A_AS Ch4 の受信完了割り込みに対する割り込みハンドラです。注
引　数	なし
リターン値	なし

4.7.6 a_as5_int_isr

a_as5_int_isr	
概　要	AFMT5_EOF 割り込みの割り込みハンドラ
ヘッダ	-
宣　言	void a_as5_int_isr(void);
説　明	A_AS Ch5 の受信完了割り込みに対する割り込みハンドラです。注
引　数	なし
リターン値	なし

4.7.7 a_as6_int_isr

a_as6_int_isr	
概　要	AFMT6_EOF 割り込みの割り込みハンドラ
ヘッダ	-
宣　言	void a_as6_int_isr(void);
説　明	A_AS Ch6 の受信完了割り込みに対する割り込みハンドラです。注
引　数	なし
リターン値	なし

4.7.8 a_as7_int_isr

a_as7_int_isr	
概　要	AFMT7_EOF 割り込みの割り込みハンドラ
ヘッダ	-
宣　言	void a_as7_int_isr(void);
説　明	A_AS Ch7 の受信完了割り込みに対する割り込みハンドラです。注
引　数	なし
リターン値	なし

4.7.9 a_as9_int_isr

a_as9_int_isr	
概　要	AFMT9_EOF 割り込みの割り込みハンドラ
ヘッダ	-
宣　言	void a_as9_int_isr(void);
説　明	A_AS Ch9 の受信完了割り込みに対する割り込みハンドラです。注
引　数	なし
リターン値	なし

4.7.10 a_as10_int_isr

a_as10_int_isr	
概　要	AFMT10_EOF 割り込みの割り込みハンドラ
ヘッダ	-
宣　言	void a_as10_int_isr(void);
説　明	A_AS Ch10 の受信完了割り込みに対する割り込みハンドラです。注
引　数	なし
リターン値	なし

4.7.11 a_as11_int_isr

a_as11_int_isr	
概　要	AFMT11_EOF 割り込みの割り込みハンドラ
ヘッダ	-
宣　言	void a_as11_int_isr(void);
説　明	A_AS Ch11 の受信完了割り込みに対する割り込みハンドラです。注
引　数	なし
リターン値	なし

4.7.12 a_as13_int_isr

a_as13_int_isr	
概　要	AFMT13_EOF 割り込みの割り込みハンドラ
ヘッダ	-
宣　言	void a_as13_int_isr(void);
説　明	A_AS Ch13 の受信完了割り込みに対する割り込みハンドラです。注
引　数	なし
リターン値	なし

4.7.13 a_as14_int_isr

a_as14_int_isr	
概　要	AFMT14_EOF 割り込みの割り込みハンドラ
ヘッダ	-
宣　言	void a_as14_int_isr(void);
説　明	A_AS Ch14 の受信完了割り込みに対する割り込みハンドラです。注
引　数	なし
リターン値	なし

【注】 初期状態のチャネル設定では、使われません。使用するチャネルを変更したときに対応するチャネルの割り込みハンドラが使われます。

4.7.14 a_as_err_isr

a_as_err_isr	
概　要	PERI_ERR0 割り込みの割り込みハンドラ
ヘッダ	-
宣　言	void a_as_err_isr(void);
説　明	A_AS Ch0～15 のタイムアウト割り込みに対する割り込みハンドラです。
引　数	なし
リターン値	なし

4.8 EnDat API 関数仕様

4.8.1 R_ENDAT_Open

R_ENDAT_Open	
概　要	EnDat エンコーダ制御の開始
ヘッダ	r_endat_rzt2_if.h
宣　言	r_endat_err_t R_ENDAT_Open(const int32_t id, r_endat_info_t* p_info);
説　明	EnDat ドライバは下記の初期設定を行います。 1.　エンコーダの初期化 (バッテリー付きエンコーダは未対応) 2.　伝送遅延補正の設定 エンコーダの電源を投入後、1.3 秒経過後に本関数を実行してください。また、ケーブルの伝送遅延を自動測定しますが、R_ENDAT_CABLE_DELAY 回の測定の内、測定が失敗した場合はその分測定回数が減ります。測定がすべて失敗したらリターン値 ENDAT_ERR_DRV を返します。
引　数	id　：使用する ID を指定します。(r_endat_rzt2_dat.h で定義されています。) R_ENDAT0_ID　：チャネル 0 を指定 R_ENDAT1_ID　：チャネル 1 を指定 ：　： R_ENDAT15_ID　：チャネル 15 を指定 上記以外　：設定不可 p_info　：エンコーダの情報を設定します。 エンコーダの情報を格納した構造体 r_endat_info_t のアドレスを指定してください。
リターン値	ENDAT_SUCCESS　：正常終了 ENDAT_ERR_INVALID_ARG　：異常終了 (id, p_info に指定した r_endat_info_t 構造体のメンバ変数が無効、または規定されていない値です) ENDAT_ERR_ACCESS　：異常終了 (既に open されています) ENDAT_ERR_DRV　：異常終了 (エンコーダの初期化に失敗しました)
注　意	本サンプルプログラムでは、チャネル 8, チャネル 12, チャネル 15 は無効です。
補　足	エンコーダの初期化処理では、Mode command “Encoder receive reset” の送信、Word 13: “Number of clocks” のリード、Word 0 “Error messages”、Word 1: “Warning messages” のクリアを行っています。バッテリー付きエンコーダの初期化を行う際は、本関数実行後に HEIDENHAIN 社アプリケーションノート(v03)の 「Power-on procedure」を参考にして、バッテリー付きエンコーダに必要な処理を追加してください。

4.8.2 R_ENDAT_Close

R_ENDAT_Close		
概　要	EnDat エンコーダの制御を終了	
ヘッダ	r_endat_rzt2_if.h	
宣　言	r_endat_err_t R_ENDAT_Close(const int32_t id);	
説　明	指定されたチャンネルの EnDat 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_endat_rzt2_dat.h で定義されています。)
	R_ENDAT0_ID	：チャネル 0 を指定
	R_ENDAT1_ID	：チャネル 1 を指定
	：	：
	R_ENDAT15_ID	：チャネル 15 を指定
	上記以外	：設定不可
リターン値	ENDAT_SUCCESS	：正常終了
	ENDAT_ERR_INVALID_ARG	：異常終了 (指定した id が無効、または規定されていない値です)
	ENDAT_ERR_ACCESS	：異常終了 (リクエストを送信中です)
注　意	本サンプルプログラムでは、チャネル 8, チャネル 12, チャネル 15 は無効です。	

4.8.3 R_ENDAT_GetVersion

R_ENDAT_GetVersion		
概　要	エンコーダ I/F ドライバのバージョンを取得	
ヘッダ	r_endat_rzt2_if.h	
宣　言	uint32_t R_ENDAT_GetVersion(void);	
説　明	EnDat 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	バージョン情報	：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）戻り値が 0x00010002 の場合、Ver.1.2

4.8.4 R_ENDAT_Control

R_ENDAT_Control		
概　要	EnDat エンコーダの制御を実行	
ヘッダ	r_endat_rzt2_if.h	
宣　言	r_endat_err_t R_ENDAT_Control(const int32_t id, const r_endat_cmd_t cmd, void *const p_buf);	
説　明	引数 cmd を使って EnDat エンコーダを制御します。制御コマンドの動作は「4.8.5 EnDat 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_endat_rzt2_dat.h で定義されています。)
	R_ENDAT0_ID	：チャネル 0 を指定
	R_ENDAT1_ID	：チャネル 1 を指定
	：	：
	R_ENDAT15_ID	：チャネル 15 を指定
	上記以外	：設定不可
	cmd	：コマンド 内容は「4.14.6(2) r_endat_cmd_t」参照。
	p_buf	：各 cmd に対応する引数
リターン値	ENDAT_SUCCESS	：正常終了
	ENDAT_ERR_INVALID_ARG	：異常終了 (id, cmd が無効、または規定されていない値です)
	その他のリターン値は「4.8.5 EnDat 制御コマンド」を参照してください。	
注意	本関数実行前に、必ず R_ENDAT_Open を実行してください。 本サンプルプログラムでは、チャネル 8, チャネル 12, チャネル 15 は無効です。	

4.8.5 EnDat 制御コマンド

(1) ENDAT_CMD_REQ

ENDAT_CMD_REQ	
概　要	EnDat エンコーダへリクエストを送信
ヘッダ	r_endat_rzt2_if.h
宣　言	r_endat_err_t R_ENDAT_Control(const int32_t id, const r_endat_cmd_t cmd, void *const p_buf);
説　明	EnDat エンコーダへリクエストを送信します。 リクエスト送信 1 回に対して endat_callback 関数が 1 回コールされます。 Continuous モード設定を有効にした場合、ENDAT_CMD_POS_STOP を実行して endat_rdst_callback 関数がコールされるまで、endat_poscon_callback 関数がコールされ続けます。 ELC モード設定を有効にした場合、ENDAT_CMD_POS_STOP を実行して endat_rdst_callback 関数がコールされるまで、ELC から送られるイベントに同期してリクエストを送信し続けます。リクエスト送信のたびに、endat_poscon_callback 関数がコールされます。
引　数	id ：使用する ID を指定します。 (r_endat_rzt2_dat.h で定義されています。) R_ENDAT0_ID ：チャネル 0 を指定 R_ENDAT1_ID ：チャネル 1 を指定 ： ： R_ENDAT15_ID ：チャネル 15 を指定 上記以外 ：設定不可 cmd ：ENDAT_CMD_REQ p_buf ：リクエスト情報 リクエスト情報を格納した r_endat_req_t 構造体のポインタを指定します。 r_endat_req_t 構造体の詳細は「4.14.4(3) r_endat_req_t」参照。
リターン値	ENDAT_SUCCESS ：正常終了 ENDAT_ERR_INVALID_ARG ：異常終了 (id, cmd が無効、または規定されていない値、p_buf が NULL、p_buf に指定された r_endat_req_t 構造体のメンバ変数が規定されていない値です) ENDAT_ERR_BUSY ：異常終了 (送信処理中、または STATR レジスタ RDY4STRB ビットが 0 のため操作できません) ENDAT_ERR_ACCESS ：異常終了 (該当チャネルが開始されていません)
注　意	本サンプルプログラムでは、チャネル 8, チャネル 12, チャネル 15 は無効です。

(2) ENDAT_CMD_POS_STOP

ENDAT_CMD_POS_STOP	
概　要	位置値連続取得の停止
ヘッダ	r_endat_rzt2_if.h
宣　言	r_endat_err_t R_ENDAT_Control(const int32_t id, const r_endat_cmd_t cmd, void *const p_buf);
説　明	Continuous モードで受信処理中には Continuous モード設定を無効に、また、ELC モードでイベント同期送受信処理中には ELC モード設定を無効にし、EnDat エンコーダからの位置値の連続受信を停止します。 位置値の連続受信処理が行われていない場合には、エラーを返します。
引　数	id ：使用する ID を指定します。(r_endat_rzt2_dat.h で定義されています。) R_ENDAT0_ID ：チャネル 0 を指定 R_ENDAT1_ID ：チャネル 1 を指定 ： ： R_ENDAT15_ID ：チャネル 15 を指定 上記以外 ：設定不可 cmd ：ENDAT_CMD_POS_STOP p_buf ：使用しません (NULL を指定してください)
リターン値	ENDAT_SUCCESS ：正常終了 ENDAT_ERR_INVALID_ARG ：異常終了 (id, cmd が無効、または規定されていない値です) ENDAT_ERR_ACCESS ：異常終了 (Continuous モードや ELC モードが有効なリクエストが送信されていません)
注　意	本サンプルプログラムでは、チャネル 8, チャネル 12, チャネル 15 は無効です。

4.9 EnDat ユーザー定義関数仕様

4.9.1 enc_init_tclk_wait_callback

enc_init_tclk_wait_callback	
概　要	TCLK 端子出力開始後の待機時間生成関数
ヘッダ	-
宣　言	void enc_init_tclk_wait_callback(void);
説　明	R_ENDAT_Open 関数で登録するコールバック関数です。接続されたエンコーダの初期化処理で、エンコーダの TCLK 端子出力開始処理後に待機する時間を生成します。100us 以上待機する処理を行ってください。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

4.9.2 enc_init_reset_wait_callback

enc_init_reset_wait_callback	
概　要	エンコーダリセット後の待機時間生成関数
ヘッダ	-
宣　言	void enc_init_reset_wait_callback(void);
説　明	R_ENDAT_Open 関数で登録するコールバック関数です。接続されたエンコーダの初期化処理で、エンコーダのリセット処理後に待機する時間を生成します。60ms 以上待機する処理を行ってください。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

4.9.3 enc_init_mem_wait_callback

enc_init_mem_wait_callback	
概　要	エンコーダのメモリエリア選択のタイムアウトエラー検出用待機時間生成関数
ヘッダ	-
宣　言	void enc_init_mem_wait_callback(void);
説　明	R_ENDAT_Open 関数で登録するコールバック関数です。接続されたエンコーダの初期化処理で、メモリエリアを選択する処理のタイムアウトエラー検出に用いる待機時間を生成します。743us 注 以上待機する処理を行ってください。関数名は例であり、自由に設定できます。 【注】 (2clock + mode command(6clock) + MRS code(8clock) + 16clock + 2T(2clock) + 最大 7clock +Start(1clock) + MRS code(8clock) + 16clock + CRC(5clock))×(1/100kHz) + t_m(30us) + t_R(0.5us) + t_D(1.7us) = 742.2us を想定しています。 エンコーダの初期化処理時はドライバ内で送信クロック周波数を 100kHz に設定しています。遅延時間 t_D はケーブル長が 150m の場合を想定しています。ご利用のエンコーダ、ケーブル長に合わせて待機時間を調整してください。
引　数	なし
リターン値	なし

4.9.4 enc_init_pram_wait_callback

enc_init_pram_wait_callback	
概　要	エンコーダのパラメータ送受信のタイムアウトエラー検出用待機時間生成関数
ヘッダ	-
宣　言	void enc_init_pram_wait_callback(void);
説　明	R_ENDAT_Open 関数で登録するコールバック関数です。接続されたエンコーダの初期化処理で、エンコーダがパラメータを送受信する処理のタイムアウトエラー検出に用いる待機時間を生成します。13ms 注 以上待機する処理を行ってください。関数名は例であり、自由に設定できます。 【注】 メモリアクセス時間(12ms) + (Start(1clock) + Address(8clock) + Parameters(16clock) + CRC(5clock))×(1/100kHz) + t_m(30us) + t_R(0.5us) + t_D(1.7us) = 12.33ms を想定しています。 エンコーダの初期化処理時はドライバ内で送信クロック周波数を 100kHz に設定しています。遅延時間 t_D はケーブル長が 150m の場合を想定しています。ご利用のエンコーダ、ケーブル長に合わせて待機時間を調整してください。
引　数	なし
リターン値	なし

4.9.5　enc_init_cable_wait_callback

enc_init_cable_wait_callback	
概　要	ケーブル伝送遅延測定のタイムアウトエラー検出用待機時間生成関数
ヘッダ	-
宣　言	void enc_init_cable_wait_callback(void);
説　明	R_ENDAT_Open 関数で登録するコールバック関数です。接続されたエンコーダの初期化処理で、ケーブル伝送遅延を測定する処理のタイムアウトエラー検出に用いる待機時間を生成します。588us 注 以上待機する処理を行ってください。関数名は例であり、自由に設定できます。 【注】 t_{cal}(5us)+(Start(1clock) + Error(1clock) + 最大メイン受信データの bit 数(48bit)+CRCbit 数(5bit))×(1/100kHz)+ t_m(30us) + t_R(0.5us) + t_D(1.7us) = 587.2us を想定しています。 エンコーダの初期化処理時はドライバ内で送信クロック周波数を 100kHz に設定しています。遅延時間 t_D はケーブル長が 150m の場合を想定しています。ご利用のエンコーダ、ケーブル長に合わせて待機時間を調整してください。
引　数	なし
リターン値	なし

4.9.6　endat_callback

endat_callback		
概　要	リクエスト送信に対するデータ受信結果通知関数	
ヘッダ	-	
宣　言	void endat_callback(r_endat_result_t * p_result, r_endat_protocol_err_t *p_err);	
説　明	R_ENDAT_Control(ENDAT_CMD_REQ)関数で登録するコールバック関数です。リクエストに対するデータ受信結果を通知します。 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 return 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	p_result	: 送受信結果 送受信結果を格納した構造体 r_endat_result_t のポインタです。次のリクエスト送信までデータ受信結果は有効です。
	p_err	: エラー情報 エラー情報を格納した構造体 r_endat_protocol_err_t のポインタです。次のリクエスト送信までデータ受信結果は有効です。
リターン値	なし	

4.9.7 endat_poscon_callback

endat_poscon_callback	
概　要	リクエスト送信(Continuous モード, ELC モード)に対するデータ受信結果通知関数
ヘッダ	-
宣　言	void endat_poscon_callback(r_endat_result_t * p_result, r_endat_protocol_err_t *p_err);
説　明	Continuous モードや ELC モードでデータ送信を行う時に、R_ENDAT_Control (ENDAT_CMD_REQ)関数で登録するコールバック関数です。リクエストに対するデータ受信結果を通知します。 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 return 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	p_result : 送受信結果 送受信結果を格納した構造体 r_endat_result_t のポインタです。次のリクエスト送受信までデータ受信結果は有効です。 p_err : エラー情報 エラー情報を格納した構造体 r_endat_protocol_err_t のポインタです。次のリクエスト送受信までデータ受信結果は有効です。
リターン値	なし

4.9.8 endat_rdst_callback

endat_rdst_callback	
概　要	次のデータ通信が開始可能であることを通知するコールバック関数
ヘッダ	-
宣　言	void endat_rdst_callback(void);
説　明	R_ENDAT_Control(ENDAT_CMD_REQ)関数で登録するコールバック関数です。リクエスト送信に対するデータ受信が完了し、次のデータ通信が可能であることを通知します。割り込み発生時に STATR レジスタ RDY ビットが 1 のとき endat_callback 関数の後にコールされます。 Continuous モードや ELC モードで動作中は、データ受信が完了するたび、endat_poscon_callback 関数の後にコールされます。 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 return 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

4.10 EnDat 割り込みハンドラ

4.10.1 endat0_rx_int_isr

endat0_rx_int_isr	
概　要	チャネル 0 データ受信完了割り込みハンドラ
ヘッダ	-
宣　言	void endat0_rx_int_isr(void);
説　明	EnDat チャネル 0 の下記の割り込み要因に対する割り込みハンドラです。注 1. ERR1, IERR2 割り込み 2. WTDG 割り込み 3. EMPFR1-3 割り込み
引　数	なし
リターン値	なし

4.10.2 endat1_rx_int_isr

endat1_rx_int_isr	
概　要	チャネル 1 データ受信完了割り込みハンドラ
ヘッダ	-
宣　言	void endat1_rx_int_isr(void);
説　明	EnDat チャネル 1 の下記の割り込み要因に対する割り込みハンドラです。 1. ERR1, IERR2 割り込み 2. WTDG 割り込み 3. EMPFR1-3 割り込み
引　数	なし
リターン値	なし

4.10.3 endat2_rx_int_isr

endat2_rx_int_isr	
概　要	チャネル 2 データ受信完了割り込みハンドラ
ヘッダ	-
宣　言	void endat2_rx_int_isr(void);
説　明	EnDat チャネル 2 の下記の割り込み要因に対する割り込みハンドラです。注 1. ERR1, IERR2 割り込み 2. WTDG 割り込み 3. EMPFR1-3 割り込み
引　数	なし
リターン値	なし

4.10.4 endat3_rx_int_isr

endat3_rx_int_isr	
概　要	チャネル 3 データ受信完了割り込みハンドラ
ヘッダ	-
宣　言	void endat3_rx_int_isr(void);
説　明	EnDat チャネル 3 の下記の割り込み要因に対する割り込みハンドラです。注 1. ERR1, IERR2 割り込み 2. WTDG 割り込み 3. EMPFR1-3 割り込み
引　数	なし
リターン値	なし

4.10.5 endat4_rx_int_isr

endat4_rx_int_isr	
概　要	チャネル 4 データ受信完了割り込みハンドラ
ヘッダ	-
宣　言	void endat4_rx_int_isr(void);
説　明	EnDat チャネル 4 の下記の割り込み要因に対する割り込みハンドラです。注 1. ERR1, IERR2 割り込み 2. WTDG 割り込み 3. EMPFR1-3 割り込み
引　数	なし
リターン値	なし

4.10.6 endat5_rx_int_isr

endat5_rx_int_isr	
概　要	チャネル 5 データ受信完了割り込みハンドラ
ヘッダ	-
宣　言	void endat5_rx_int_isr(void);
説　明	EnDat チャネル 5 の下記の割り込み要因に対する割り込みハンドラです。注 1. ERR1, IERR2 割り込み 2. WTDG 割り込み 3. EMPFR1-3 割り込み
引　数	なし
リターン値	なし

4.10.7 endat6_rx_int_isr

endat6_rx_int_isr	
概　要	チャネル 6 データ受信完了割り込みハンドラ
ヘッダ	-
宣　言	void endat6_rx_int_isr(void);
説　明	EnDat チャネル 6 の下記の割り込み要因に対する割り込みハンドラです。注 1. ERR1, IERR2 割り込み 2. WTDG 割り込み 3. EMPFR1-3 割り込み
引　数	なし
リターン値	なし

4.10.8 endat7_rx_int_isr

endat7_rx_int_isr	
概　要	チャネル 7 データ受信完了割り込みハンドラ
ヘッダ	-
宣　言	void endat7_rx_int_isr(void);
説　明	EnDat チャネル 7 の下記の割り込み要因に対する割り込みハンドラです。注 1. ERR1, IERR2 割り込み 2. WTDG 割り込み 3. EMPFR1-3 割り込み
引　数	なし
リターン値	なし

4.10.9 endat9_rx_int_isr

endat9_rx_int_isr	
概　要	チャネル 9 データ受信完了割り込みハンドラ
ヘッダ	-
宣　言	void endat9_rx_int_isr(void);
説　明	EnDat チャネル 9 の下記の割り込み要因に対する割り込みハンドラです。注 1. ERR1, IERR2 割り込み 2. WTDG 割り込み 3. EMPFR1-3 割り込み
引　数	なし
リターン値	なし

4.10.10 endat10_rx_int_isr

endat10_rx_int_isr	
概　要	チャネル 10 データ受信完了割り込みハンドラ
ヘッダ	-
宣　言	void endat10_rx_int_isr(void);
説　明	EnDat チャネル 10 の下記の割り込み要因に対する割り込みハンドラです。注 1. ERR1, IERR2 割り込み 2. WTDG 割り込み 3. EMPFR1-3 割り込み
引　数	なし
リターン値	なし

4.10.11 endat11_rx_int_isr

endat11_rx_int_isr	
概　要	チャネル 11 データ受信完了割り込みハンドラ
ヘッダ	-
宣　言	void endat11_rx_int_isr(void);
説　明	EnDat チャネル 11 の下記の割り込み要因に対する割り込みハンドラです。注 1. ERR1, IERR2 割り込み 2. WTDG 割り込み 3. EMPFR1-3 割り込み
引　数	なし
リターン値	なし

4.10.12 endat13_rx_int_isr

endat13_rx_int_isr	
概　要	チャネル 13 データ受信完了割り込みハンドラ
ヘッダ	-
宣　言	void endat13_rx_int_isr(void);
説　明	EnDat チャネル 13 の下記の割り込み要因に対する割り込みハンドラです。注 1. ERR1, IERR2 割り込み 2. WTDG 割り込み 3. EMPFR1-3 割り込み
引　数	なし
リターン値	なし

4.10.13 endat14_rx_int_isr

endat14_rx_int_isr	
概　要	チャネル 14 データ受信完了割り込みハンドラ
ヘッダ	-
宣　言	void endat14_rx_int_isr(void);
説　明	EnDat チャネル 14 の下記の割り込み要因に対する割り込みハンドラです。注 1. ERR1, IERR2 割り込み 2. WTDG 割り込み 3. EMPFR1-3 割り込み
引　数	なし
リターン値	なし

【注】 初期状態のチャネル設定では、使われません。使用するチャネルを変更したときに対応するチャネルの割り込みハンドラが使われます。

4.11 使用割り込み一覧

表 4.5 に Dual Encoder ドライバで使用する割り込みを示します。チャネル変更を行った場合には、A-format エンコーダと EnDat エンコーダの割り込みチャネル変更に伴って、使われる ID も変わります。

表 4.5　Dual Encoder ドライバで使用する割り込み

割り込み	ID 注 1		概要
	CR52 版	CA55 版	
ENCIF_ERR0	388	417	Ch0～Ch15 のタイムアウトで割り込みが発生します注 2
ENCIF00_INT0	389	716	Ch0 A-format エンコーダの、Ch0 のデータ受信完了で割り込みが発生します。
ENCIF01_INT0	390	720	Ch1 EnDat エンコーダの下記の割り込み要因で割り込みが発生します 1. ERR1, IERR2 割り込み 2. WTDG 割り込み 3. EMPFR1-3 割り込み

【注】 1. サンプルプログラムには、CPU コア Cortex-R52 で動作する CR52 版と、CPU コア Cortex-A55 で動作する CA55 版があります。CR52 版、CA55 版はそれぞれの版に関する説明です。
2. 本サンプルプログラムでは、チャネル 8, チャネル 12, チャネル 15 は無効です。チャネル 8, チャネル 12, チャネル 15 による割り込みは発生しません。

4.12 定数/エラーコード一覧

表 4.6 に定数/エラーコード定義表の一覧を示します。各定義については、それぞれの表を参照してください。EnDat のエラーコードは「4.14.6(1) r_endat_err_t」を参照してください。

表 4.6　定数/エラーコード定義表の一覧

表番号	内容
表 4.7	A-format ドライバで使用するユーザー定義の定数(r_a_as_rzt2_config.h)
表 4.8	A-format ドライバの種類
表 4.9	A_AS とエンコーダの接続方式
表 4.10	R_A_AS_Control 関数の制御コマンド
表 4.11	A-format ビットレート
表 4.12	A-format エンコーダアドレス
表 4.13	A-format コマンド
表 4.14	A-format エラーコード
表 4.15	EnDat ドライバで使用するユーザー定義の定数 (r_endat_rzt2_config.h)
表 4.16	EnDat 2.2 モードコマンド
表 4.17	EnDat 送信クロック周波数
表 4.18	EnDat Watchdog Timer の時間の単位
表 4.19	EnDat データ送信開始時のデータ Low 期間
表 4.20	EnDat MRS コード一覧

表 4.7　A-format ドライバで使用するユーザー定義の定数(r_a_as_rzt2_config.h)

定数名	設定値	内容
R_A_AS_T2_ONE_2500KBPS	0x0000	ビットレートが 2.5Mbps、接続方式が 1 対 1 の場合の BRSEL レジスタ TM ビット設定値 注
R_A_AS_T2_ONE_4MBPS	0x0000	ビットレートが 4Mbps、接続方式が 1 対 1 の場合の BRSEL レジスタ TM ビット設定値 注
R_A_AS_T2_ONE_6670KBPS	0x0000	ビットレートが 6.67Mbps、接続方式が 1 対 1 の場合の BRSEL レジスタ TM ビット設定値 注
R_A_AS_T2_ONE_8MBPS	0x0000	ビットレートが 8Mbps、接続方式が 1 対 1 の場合の BRSEL レジスタ TM ビット設定値 注
R_A_AS_T2_BUS_2500KBPS	0x001A	ビットレートが 2.5Mbps、接続方式がバスの場合の BRSEL レジスタ TM ビット設定値 注
R_A_AS_T2_BUS_4MBPS	0x0010	ビットレートが 4Mbps、接続方式がバスの場合の BRSEL レジスタ TM ビット設定値 注
R_A_AS_T2_BUS_6670KBPS	0x0009	ビットレートが 6.67Mbps、接続方式がバスの場合の BRSEL レジスタ TM ビット設定値 注
R_A_AS_T2_BUS_8MBPS	0x0008	ビットレートが 8Mbps、接続方式がバスの場合の BRSEL レジスタ TM ビット設定値 注

【注】 サンプルプログラムでは推奨設定値を各レジスタに設定しています。

表 4.8　A-format ドライバの種類

定数名	設定値	内容
R_A_AS_A_FORMAT	0	A-format ドライバを指定

表 4.9　A_AS とエンコーダの接続方式

定数名	設定値	内容
R_A_AS_ONE_FOR_ONE	0	1 対 1 接続
R_A_AS_BUS	1	バス接続

表 4.10　R_A_AS_Control 関数の制御コマンド

定数名	設定値	内容
R_A_AS_CMD_SET_PARAM	0xAF000000	リクエスト情報を設定
R_A_AS_CMD_ELC_DISABLE	0xAF000002	ELC イベント入力トリガを無効化
R_A_AS_CMD_TX_TRG	0xAF000003	トリガによるコマンド送信を開始
R_A_AS_CMD_TX_ELC	0xAF000005	ELC イベント入力トリガによるコマンド送信を開始

表 4.11　A-format ビットレート

定数名	設定値	内容
R_A_AS_2500KBPS	0	2.5Mbps
R_A_AS_4MBPS	1	4Mbps
R_A_AS_6670KBPS	2	6.67Mbps
R_A_AS_8MBPS	3	8Mbps

表 4.12　A-format エンコーダアドレス

定数名	設定値	内容
R_A_AS_ECN1	0	エンコーダ区分 ENC1 のエンコーダアドレス
R_A_AS_ECN2	1	エンコーダ区分 ENC2 のエンコーダアドレス
R_A_AS_ECN3	2	エンコーダ区分 ENC3 のエンコーダアドレス
R_A_AS_ECN4	3	エンコーダ区分 ENC4 のエンコーダアドレス
R_A_AS_ECN5	4	エンコーダ区分 ENC5 のエンコーダアドレス
R_A_AS_ECN6	5	エンコーダ区分 ENC6 のエンコーダアドレス
R_A_AS_ECN7	6	エンコーダ区分 ENC7 のエンコーダアドレス
R_A_AS_ECN8	7	エンコーダ区分 ENC8 のエンコーダアドレス

表 4.13　A-format コマンド

定数名	設定値	内容
R_A_AS_CDF0	0	コマンドデータフレーム CDF0 の定義です。
R_A_AS_CDF1	1	コマンドデータフレーム CDF1 の定義です。
R_A_AS_CDF2	2	コマンドデータフレーム CDF2 の定義です。
R_A_AS_CDF3	3	コマンドデータフレーム CDF3 の定義です。
R_A_AS_CDF4	4	コマンドデータフレーム CDF4 の定義です。
R_A_AS_CDF5	5	コマンドデータフレーム CDF5 の定義です。
R_A_AS_CDF6	6	コマンドデータフレーム CDF6 の定義です。
R_A_AS_CDF7	7	コマンドデータフレーム CDF7 の定義です。
R_A_AS_CDF8	8	コマンドデータフレーム CDF8 の定義です。
R_A_AS_CDF9	9	コマンドデータフレーム CDF9 の定義です。
R_A_AS_CDF10	10	コマンドデータフレーム CDF10 の定義です。
R_A_AS_CDF11	11	コマンドデータフレーム CDF11 の定義です。
R_A_AS_CDF12	12	コマンドデータフレーム CDF12 の定義です。
R_A_AS_CDF13	13	コマンドデータフレーム CDF13 の定義です。
R_A_AS_CDF14	14	コマンドデータフレーム CDF14 の定義です。
R_A_AS_CDF15	15	コマンドデータフレーム CDF15 の定義です。
R_A_AS_CDF16	16	コマンドデータフレーム CDF16 の定義です。
R_A_AS_CDF17	17	コマンドデータフレーム CDF17 の定義です。
R_A_AS_CDF18	18	コマンドデータフレーム CDF18 の定義です。
R_A_AS_CDF19	19	コマンドデータフレーム CDF19 の定義です。
R_A_AS_CDF21	21	コマンドデータフレーム CDF21 の定義です。
R_A_AS_CDF22	22	コマンドデータフレーム CDF22 の定義です。
R_A_AS_CDF27	27	コマンドデータフレーム CDF27 の定義です。
R_A_AS_CDF28	28	コマンドデータフレーム CDF28 の定義です。
R_A_AS_CDF29	29	コマンドデータフレーム CDF29 の定義です。
R_A_AS_CDF30	30	コマンドデータフレーム CDF30 の定義です。

表 4.14　A-format エラーコード

定数名	設定値	内容
R_A_AS_SUCCESS	0	正常終了
R_A_AS_ERR_INVALID_ARG	-1	引数異常
R_A_AS_ERR_BUSY	-2	API を実行できない状態
R_A_AS_ERR_ACCESS	-3	API の実行順序エラー

表 4.15　EnDat ドライバで使用するユーザー定義の定数 (r_endat_rzt2_config.h)

定数名	設定値	内容
R_ENDAT_CABLE_DELAY	5	伝送遅延を自動で測定する回数です。5～255 回に設定してください。
R_ENDAT_ADD_NUM	0u	受信する付加情報数

表 4.16　EnDat 2.2 モードコマンド

定数名	設定値	内容
R_ENDAT_POS	0x07u	「Encoder send position values」コマンド
R_ENDAT_MEM	0x0Eu	「Selection of memory area」コマンド
R_ENDAT_RX_PARAM	0x1Cu	「Encoder receive parameter」コマンド
R_ENDAT_PARAM	0x23u	「Encoder send parameter」コマンド
R_ENDAT_RESET	0x2Au	「Encoder receive reset」コマンド
R_ENDAT_POS_ADD_DATA	0x38u	「Encoder send position values with additional data」コマンド
R_ENDAT_POS_MEM	0x09u	「Encoder send position values and selection of the memory area」コマンド
R_ENDAT_POS_RX_PARAM	0x1Bu	「Encoder send position values and receive parameter」コマンド
R_ENDAT_POS_PARAM	0x24u	「Encoder send position values and send parameter」コマンド
R_ENDAT_POS_RX_ERR_RESET	0x2Du	「Encoder send position values and receiver error reset」コマンド

【注】 詳細は HEIDENHAIN 社に問い合わせることで入手可能な「EnDat Specification」を参照してください。

表 4.17　EnDat 送信クロック周波数

定数名	設定値	内容
R_ENDAT_FTCLK_16670	0x3u	16.67 MHz 注
R_ENDAT_FTCLK_8330	0x6u	8.33 MHz 注
R_ENDAT_FTCLK_4160	0xBu	4.16 MHz 注
R_ENDAT_FTCLK_4000	0x8u	4 MHz 注
R_ENDAT_FTCLK_2000	0xCu	2 MHz
R_ENDAT_FTCLK_1000	0xDu	1 MHz
R_ENDAT_FTCLK_200	0xEu	0.2 MHz
R_ENDAT_FTCLK_100	0xFu	0.1 MHz

【注】 伝送遅延補正を有効 (delay_comp=true) にして使用してください。

表 4.18　EnDat Watchdog Timer の時間の単位

定数名	設定値	内容
R_ENDAT_WD_RANGE_US	0x00u	Watchdog Timer の時間の単位がマイクロ秒
R_ENDAT_WD_RANGE_MS	0x80u	Watchdog Timer の時間の単位がミリ秒

表 4.19　EnDat データ送信開始時のデータ Low 期間

定数名	設定値	内容
R_ENDAT_TST_HALF_TCLK	0x00u	1/2 TCLK
R_ENDAT_TST_500NS	0x01u	0.5 us 注
R_ENDAT_TST_1US	0x02u	1 us 注
R_ENDAT_TST_1500NS	0x03u	1.5 us 注
R_ENDAT_TST_2US	0x04u	2 us 注
R_ENDAT_TST_4US	0x05u	4 us 注
R_ENDAT_TST_8US	0x06u	8 us 注
R_ENDAT_TST_10US	0x07u	10 us 注

【注】 データ Low 期間に誤差があります。詳細はハードウェアマニュアルを参照してください。

表 4.20　EnDat MRS コード一覧

定数名	設定値	内容
R_ENDAT_MRS_INFO1_NOP	0x40u	Send additional info 1 without data contents (NOP)
R_ENDAT_MRS_DIA	0x41u	Send diagnostic values
R_ENDAT_MRS_POS2_LSB	0x42u	Send position value 2, word 1 LSB
R_ENDAT_MRS_POS2_CENTER	0x43u	Send position value 2, word 2
R_ENDAT_MRS_POS2_MSB	0x44u	Send position value 2, word 3 MSB
R_ENDAT_MRS_MEM_LSB	0x45u	Acknowledge memory content LSB
R_ENDAT_MRS_MEM_MSB	0x46u	Acknowledge memory content MSB
R_ENDAT_MRS_MRS_CODE	0x47u	Acknowledge MRS code
R_ENDAT_MRS_TEST_SMD	0x48u	Acknowledge test command
R_ENDAT_MRS_TEST_LSB	0x49u	Send test values, word 1 LSB
R_ENDAT_MRS_TEST_CENTER	0x4Au	Send test values, word 2
R_ENDAT_MRS_TEST_MSB	0x4Bu	Send test values, word 3 MSB
R_ENDAT_MRS_TEMP1	0x4Cu	Send temperature 1
R_ENDAT_MRS_TEMP2	0x4Du	Send temperature 2
R_ENDAT_MRS_ADD_SEN	0x4Eu	Additional sensors
R_ENDAT_MRS_NOT_INFO1	0x4Fu	Stop sending additional datum 1
R_ENDAT_MRS_INFO2_NOP	0x50u	Send additional datum 2 without data contents
R_ENDAT_MRS_COM	0x51u	Send commutation
R_ENDAT_MRS_ACC	0x52u	Send acceleration
R_ENDAT_MRS_COM_ACC	0x53u	Send commutation & acceleration
R_ENDAT_MRS_LIM_POS	0x54u	Send limit position signals
R_ENDAT_MRS_LIM_POS_ACC	0x55u	Send limit position signals & acceleration
R_ENDAT_MRS_ASY_POS_LSB	0x56u	Asynchronous position value, word 1 LSB
R_ENDAT_MRS_ASY_POS_CENTER	0x57u	Asynchronous position value, word 2
R_ENDAT_MRS_ASY_POS_MSB	0x58u	Asynchronous position value, word 3 MSB
R_ENDAT_MRS_OPE_STA_ERR	0x59u	Operating status error sources
R_ENDAT_MRS_TIM_STA	0x5Bu	Timestamp
R_ENDAT_MRS_NOT_INFO2	0x5Fu	Stop sending additional datum 2
R_ENDAT_MRS_OPE_STAT	0xB9u	Operating status
R_ENDAT_MRS_ENC_MANU1	0xA1u	Parameters of the encoder manufacturer 1
R_ENDAT_MRS_ENC_MANU2	0xA3u	Parameters of the encoder manufacturer 2
R_ENDAT_MRS_ENC_MANU3	0xA5u	Parameters of the encoder manufacturer 3
R_ENDAT_MRS_OPE_PARAM	0xA7u	Operating parameters
R_ENDAT_MRS_OEM1	0xA9u	Parameters of the OEM 1
R_ENDAT_MRS_OEM2	0xABu	Parameters of the OEM 2
R_ENDAT_MRS_OEM3	0xADu	Parameters of the OEM 3
R_ENDAT_MRS_OEM4	0xAFu	Parameters of the OEM 4
R_ENDAT_MRS_COMP_VAL1	0xB1u	Compensation Values of the encoder manufacturer 1
R_ENDAT_MRS_COMP_VAL2	0xB3u	Compensation Values of the encoder manufacturer 2
R_ENDAT_MRS_COMP_VAL3	0xB5u	Compensation Values of the encoder manufacturer 3
R_ENDAT_MRS_COMP_VAL4	0xB7u	Compensation Values of the encoder manufacturer 4
R_ENDAT_MRS_PARAM_ENDAT22	0xBDu	Parameters of the encoder manufacturer for EnDat 2.2
R_ENDAT_MRS_PARAM_SEC2	0xBFu	Parameters of the section 2 memory area
R_ENDAT_MRS_OPE_PARAM2	0xBBu	Operating parameters 2

【注】 詳細は HEIDENHAIN 社に問い合わせることで入手可能な「EnDat Specification」を参照してください。

4.13 固定幅整数一覧

表 4.21 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定数は、標準ライブラリで定義されています。

表 4.21　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（標準ライブラリにて定義）
int16_t	16 ビット整数、符号あり（標準ライブラリにて定義）
int32_t	32 ビット整数、符号あり（標準ライブラリにて定義）
int64_t	64 ビット整数、符号あり（標準ライブラリにて定義）
uint8_t	8 ビット整数、符号なし（標準ライブラリにて定義）
uint16_t	16 ビット整数、符号なし（標準ライブラリにて定義）
uint32_t	32 ビット整数、符号なし（標準ライブラリにて定義）
uint64_t	64 ビット整数、符号なし（標準ライブラリにて定義）

4.14 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。

4.14.1 A-format 用構造体

(1) r_a_as_info_t

A_AS 制御部の初期化情報。

```
typedef struct
{
    uint8_t     connect;    接続方式
                            A_AS とエンコーダの接続方式を指定してください。指定する値は
                            「表 4.9　A_AS とエンコーダの接続方式」を参照してください。
                            ※本設定は BRSEL レジスタに反映されます。
    uint8_t     bitrate;    ビットレート
                            エンコーダとの通信におけるビットレートを指定してください。指定
                            する値は「表 4.11　A-format ビットレート」を参照してください。
                            ※本設定は BRSEL レジスタに反映されます。
    uint16_t    ifmg;       マージン値
                            ※RZT2M グループ A-format ドライバ I/F との互換性のために設けら
                            れています。設定値は、RZ/N2H では使われません。
} r_a_as_info_t
```

(2) r_a_as_req_t

エンコーダに送信するリクエスト情報。

```
typedef struct
{
```

uint8_t	encadr;	エンコーダアドレス エンコーダアドレスを指定してください。指定する値は「表 4.12　A-format エンコーダアドレス」を参照してください。 この設定は COMMAND レジスタの XEA ビットに反映されます。
uint8_t	cmd;	コマンド エンコーダに送信するコマンドコードを指定してください。指定する値は「表 4.13　A-format コマンド」を参照してください。 「表 4.13　A-format コマンド」以外の値で 0x20 以上の値を指定すると、R_A_AS_ERR_INVALID_ARG が発生します。 接続方式によって使用できないコマンドがあります。
uint8_t	memadr;	メモリアドレス エンコーダのメモリアドレスを指定してください。 コマンドが以下の場合のみ設定してください。 cmd = R_A_AS_CDF13 cmd = R_A_AS_CDF14 ※ R_A_AS_CDF13 の場合、アクセス可能アドレス範囲は 0x00~0xFF までとなります。 R_A_AS_CDF14 の場合、アクセス可能アドレス範囲は 0x00~0xEF までとなります。
uint16_t	memdat;	メモリへ書き込むデータ メモリへ書き込むデータを指定してください。 コマンドが以下の場合のみ設定してください。 cmd = R_A_AS_CDF14
uint32_t	encid;	識別コード 識別コードの値は 24bit 長の値を指定してください。 コマンドが以下の場合のみ設定してください。 cmd = R_A_AS_CDF18 cmd = R_A_AS_CDF19
r_a_as_result_cb_t	cbadr_txerr	PERI_ERR0 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「4.6.1 a_as_txerr_callback」を参照してください。注 1
r_a_as_result_cb_t	cbadr_rxset;	AFMTi_EOF(i = 0～15)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「4.6.2 a_as_rxset_callback」を参照してください。注 1
r_a_as_result_cb_t	cbadr_rxend;	PERI_ERR0 割り込み発生時や AFMTi_EOF (i = 0～15)割り込み発生時に、cbadr_txerr()関数や cbadr_rxset() 関数に続いてコールされるコールバック関数のポインタ 詳細は「4.6.3 a_as_rxend_callback」を参照してください。注 1
bool	pre;	ELC イベント入力トリガによるデータ送受信中に、リクエスト情報を設定する場合は、true にしてください。 ELC イベント入力トリガによるデータ送受信中以外で、リクエスト情報を設定する場合は、false にしてください。 (true：ELC イベント入力トリガによるデータ送受信中にリクエスト設定)

```
} r_a_as_req_t
```

【注】 1　NULL を指定するとコールバックが発生しません。

(3) r_a_as_result_t

通常受信時の送受信結果

```
typedef struct
{
    r_a_as_req_err_t    result;     送受信結果
                                    詳細は列挙型「4.14.3(1) r_a_as_req_err_t」参照してください。
    r_a_as_data_t       data;       受信データ
                                    詳細は構造体「4.14.1(4) r_a_as_data_t」参照してください。
    r_a_as_status_t     status;     A_AS のステータス
                                    詳細は構造体「4.14.1(5) r_a_as_status_t」参照してください。
} r_a_as_result_t
```

(4) r_a_as_data_t

通常受信時の受信データ

```
typedef struct
{
    uint32_t            rxi;        ENCnRXDATA0 レジスタ値
                                    ENCnRXDATA0 レジスタの値が格納されます。
    uint32_t            rxd0;       ENCnRXDATA1 レジスタ値
                                    ENCnRXDATA1 レジスタの値が格納されます。
    uint32_t            rxd1;       ENCnRXDATA2 レジスタ値
                                    ENCnRXDATA2 レジスタの値が格納されます。
} r_a_as_data_t
```

(5) r_a_as_status_t

通常受信時の A_AS のステータス

```
typedef struct
{
    bool    iwdgerr;    IF Watchdog エラー情報。タイムアウトエラーに当たる。
    bool    dwdgerr;    DF Watchdog エラー情報。タイムアウトエラーに当たる。
    bool    starterr;   スタートビットエラー情報。CA[1]の FORM ステータスに当たる。
    bool    stoperr;    ストップビットエラー情報。CA[1]の FORM ステータスに当たる。
    bool    syncerr;    シンクコードエラー情報。CA[2]の SYNC ステータスに当たる。
    bool    rxeaerr;    受信エンコーダアドレスエラー情報。CA[0]の CMD ステータスに当たる。
    bool    crcerr;     CRC エラー情報を格納。CA[3]の CRC ステータスに当たる。
    bool    rxccerr;    受信コマンドコードエラー情報。CA[0]の CMD ステータスに当たる。
    bool    mdaterr;    EEPROM データエラー情報 注 1
    bool    madrerr;    EEPROM アドレスエラー情報 注 1
    bool    rxdzerr;    識別コードエラー情報 注 1
    bool    fd1err;     固定データエラー(1) 情報 注 1
    bool    fd2err;     固定データエラー(2) 情報 注 1
    bool    fd3err;     固定データエラー(3) 情報 注 1
    bool    fd5err;     固定データエラー(5) 情報 注 1
    bool    elcin;      ELC イベント入力情報 注 1
    uint8_t txcc;       送信コマンドコード (0 : CDF0～19,23～30、1:CDF21、2:CDF22)
    bool    rxset;      受信データ設定完了情報 (true：リード可能、false：リード不可)
    bool    timer;      タイマステータス情報 注 1
    bool    txerr;      送信エラー情報 注 1
    bool    rxend;      受信完了情報 (true：受信した、false：受信していない)
} r_a_as_status_t
```

【注】 1. RZ/T2M グループ A-format エンコーダドライバ I/F との互換性のために設けられています。RZ/N2H では使われません。常に false です。

4.14.2 A-format 用共用体

使用しません。

4.14.3 A-format 用列挙型

(1) r_a_as_req_err_t

A-format エンコーダからの受信結果。

```
typedef enum
{
    R_A_AS_REQ_SUCCESS = 0,   データ送受信正常終了
    R_A_AS_REQ_ERR            データ送受信エラー発生
                              構造体「4.14.1(5) r_a_as_status_t」のエラー情報(timer と rxend と rxset と elcin と txcc 以外)が 1 つでも true の場合、データ送受信エラー発生とします。
    R_A_AS_REQ_BP_ERR         FIFO が FULL
                              バイパス受信時のみ発生します。
} r_a_as_req_err_t
```

4.14.4 EnDat 用構造体

(1) r_endat_info_t

EnDat 制御部の初期化情報

```
typedef struct
{
    uint8_t             ftclk;                  送信クロック周波数設定
                                                「表 4.17　EnDat 送信クロック周波数」参照
                                                本設定は KONFR1 レジスタ FTCLK ビットに反映されます。
    bool                filter;                 ノイズフィルタの設定 (true: 有効, false: 無効)
                                                ※ RZ/T2M グループ EnDat ドライバとの互換性のために設けられています。設定値は、RZ/N2H では使われません。
    bool                delay_comp;             伝送遅延補正 (true: 有効, false: 無効)
                                                本設定は KONFR1 レジスタの DELAYCMP ビットに反映されます。
    uint8_t             tst;                    データ送信開始時の Low 期間を設定
                                                「表 4.19　EnDat データ送信開始時のデータ Low 期間」参照
                                                本設定は KONFR2 レジスタの RCVTIME ビットに反映されます。
    endat_wait_cb_t     p_enc_init_tclk_wait;   エンコーダの TCLK 端子出力開始後の待機時間を生成するコールバック関数のポインタ
                                                詳細は「4.9.1 enc_init_tclk_wait_callback」参照
                                                NULL は設定しないでください。
    endat_wait_cb_t     p_enc_init_reset_wait;  エンコーダリセット後の待機時間を生成するコールバック関数のポインタ
                                                詳細は「4.9.2enc_init_reset_wait_callback」参照
                                                NULL は設定しないでください。
    endat_wait_cb_t     p_enc_init_mem_wait;    エンコーダのメモリエリア選択のタイムアウトエラー検出用の待機時間を生成するコールバック関数のポインタ
                                                詳細は「4.9.3 enc_init_mem_wait_callback」参照
                                                NULL は設定しないでください。
    endat_wait_cb_t     p_enc_init_pram_wait;   エンコーダのパラメータ送受信のタイムアウトエラー検出用の待機時間を生成する関数のポインタ
                                                詳細は「4.9.4 enc_init_pram_wait_callback」参照
                                                NULL は設定しないでください。
    endat_wait_cb_t     p_enc_init_cable_wait;  ケーブル伝送遅延測定のタイムアウトエラー検出用の待機時間を生成する関数のポインタ。伝送遅延補正が無効 (delay_comp = false) の場合は設定を省略できます。
                                                詳細は「4.9.5 enc_init_cable_wait_callback」参照
                                                伝送遅延補正を有効にした場合には、NULL は設定しないでください。
} r_endat_info_t
```

(2) r_endat_watchdog_t

Watchdog Timer 設定時間

```
typedef struct
{
    uint8_t         range;      Watchdog Timer の時間の単位を設定
                                「表 4.18 EnDat Watchdog Timer の時間の単位」参照
    uint8_t         time;       Watchdog Timer の時間を設定
                                「表 4.22 Watchdog Timer 時間対応表」参照
} r_endat_watchdog_t
```

表 4.22 Watchdog Timer 時間対応表

time	Watchdog Timer の時間	
	range = R_ENDAT_WD_RANGE_US	range = R_ENDAT_WD_RANGE_MS
0	停止	停止
1	2 us	0.2 ms
2	4 us	0.4 ms
3	6 us	0.6 ms
:	:	:
10	20 us	2.0 ms
:	:	:
127	254 us	25.4 ms

【注】 停止以外の時間は誤差があります。詳細はハードウェアマニュアルを参照してください。

(3) r_endat_req_t

EnDat2.2 準拠エンコーダに送信するリクエスト情報

エンコーダに対してはモードコマンドと MRS コード、アドレス、ポートアドレスを組み合わせ送信します。組み合わせを「表 4.23 モードコマンド組み合わせ表」に示します。

```
typedef struct
{
    uint8_t                 mode_cmd;          EnDat 2.2 モードコマンド
                                               「表 4.16 EnDat 2.2 モードコマンド」参照
    bool                    dt;                Continuous モード設定 (true: 有効, false: 無効)
                                               mode_cmd=R_ENDAT_POS の場合のみ、本設定は有効です。
    uint8_t                 mrs;               MRS コード
                                               「表 4.20 EnDat MRS コード一覧」参照
                                               「表 4.23 モードコマンド組み合わせ表」で MRS コードと対になっているモードコマンドを
                                               mode_cmd に指定した場合のみ有効です。
    uint8_t                 addr;              アドレス設定 (0x00～0xFF)
                                               「表 4.23 モードコマンド組み合わせ表」でアドレスと対になっているモードコマンドを mode_cmd に指定した場合のみ有効です。
    uint16_t                param_instruction; エンコーダのメモリ領域に書き込むパラメータ
                                               「表 4.23 モードコマンド組み合わせ表」で
                                               Parameters や Block address と対になっているモードコマンドを mode_cmd に指定した場合のみ有効です。
    r_endat_watchdog_t      watchdog;          Watchdog Timer 設定時間
                                               「4.14.4(2) r_endat_watchdog_t」参照
                                               以下の設定のリクエスト送信時は無効(time=0)に設定してください。
                                                 mode_cmd=R_ENDAT_POS かつ dt=true の場合
                                                 mode_cmd=R_ENDAT_RESET の場合
                                                 mode_cmd=R_ENDAT_RX_PARAM の場合
                                                 mode_cmd=R_ENDAT_PARAM の場合
                                               本設定は KONFR2 レジスタ WTDOG ビットに反映されます。
    bool                    elc;               ELC モード設定 (true: 有効, false: 無効)
                                               mode_cmd=R_ENDAT_POS かつ dt=false の場合のみ、本設定は有効です。
    r_endat_isr_result_cb_t p_isr_result;      リクエストの結果を通知するコールバック関数へのポインタ
                                               詳細は「4.9.6 endat_callback」および
                                               「4.9.7 endat_poscon_callback」参照
                                               NULL は設定しないでください。
    r_endat_isr_rdst_cb_t   p_isr_rdst;        次のデータ通信が開始可能であることを通知するコールバック関数へのポインタ
                                               詳細は「4.9.8 endat_rdst_callback」参照
                                               NULL は設定しないでください。
} r_endat_req_t
```

表 4.23　モードコマンド組み合わせ表

mode_cmd	コマンドの値	mrs / addr	param_instruction
R_ENDAT_POS	0x07u	--	--
R_ENDAT_MEM	0x0Eu	MRS コード	--
R_ENDAT_RX_PARAM	0x1Cu	アドレス	Parameters 注 1
R_ENDAT_PARAM	0x23u	アドレス	--
R_ENDAT_RESET	0x2Au	アドレス	--
R_ENDAT_POS_ADD_DATA	0x38u	--	--
R_ENDAT_POS_MEM	0x09u	MRS コード	Block address 注 2
R_ENDAT_POS_RX_PARAM	0x1Bu	アドレス	Parameters 注 1
R_ENDAT_POS_PARAM	0x24u	アドレス	--
R_ENDAT_POS_RX_ERR_RESET	0x2Du	アドレス	--

【注】 1. アドレスによって設定値を考慮する必要があります。
2. MRS コードが R_ENDAT_MRS_PARAM_SEC2 の場合のみ使用します。

(4) r_endat_result_t

送受信結果

```
typedef struct
{
    r_endat_req_err_t       result;     リクエストの送受信結果
                                        「4.14.6(3) r_endat_req_err_t」参照
    r_endat_data_t          data;       受信データ
                                        「4.14.4(5) r_endat_data_t」参照
    r_endat_status_t        status;     エンコーダのステータス
                                        「4.14.4(6) r_endat_status_t」参照
} r_endat_result_t
```

(5) r_endat_data_t

受信データ

```
typedef struct
{
    uint64_t                pos;        受信した位置値データまたはテスト値
                                        下位 32bit に EMPFR1_L レジスタの RDATA_L ビット、上位 32bit に EMPFR1_H レジスタの RDATA_H ビットの値が格納されます。
    uint32_t                add_datum1; 付加情報 1
                                        EMPFR3 レジスタの D ビットの値が格納されます。
    uint32_t                add_datum2; 付加情報 2
                                        EMPFR2 レジスタの D ビットの値が格納されます。
} r_endat_data_t
```

(6) r_endat_status_t

エンコーダのステータス

```
typedef struct
{
    bool        busy;       エンコーダ内蔵メモリのステータス
                            (true: アクセス中, false: アクセス可能)
    bool        rm;         インクリメントエンコーダの原点ステータス
                            (true: 原点検出, false: 原点未検出)
    bool        wrn;        エンコーダ内部の警告ステータス
                            (true: 警告あり, false: 警告なし)
} r_endat_status_t
```

(7) r_endat_protocol_err_t

EnDat I/F およびエンコーダのエラー情報

```
typedef struct
{
    bool        err1;       Error1 ビットステータス (true: 発生, false: 未発生)
    bool        crc1;       位置値の CRC チェックエラー (true: 発生, false: 未発生)
    bool        ftype1;     EnDat TYPE1 エラー (true: 発生, false: 未発生)
    bool        ftype2;     EnDat TYPE2 エラー (true: 発生, false: 未発生)
    bool        msadr;      EnDat TYPE2 エラー内のアドレスエラー (true: 発生, false: 未発生)
    bool        err2;       Error2 ビットステータス (true: 発生, false: 未発生)
    bool        crc3;       付加情報 1 の CRC チェックエラー(true: 発生, false: 未発生)
    bool        crc2;       付加情報 2 の CRC チェックエラー(true: 発生, false: 未発生)
    bool        wdg;        ウォッチドッグエラー(true: 発生, false: 未発生)
    bool        ftype3;     EnDat TYPE3 エラー (true: 発生, false: 未発生)
    bool        modeerr;    モードコマンド送信エラー
                            (本ビットは使われません。常に false です) 注 1
} r_endat_protocol_err_t
```

【注】 1. RZ/T2M グループ EnDat エンコーダドライバ I/F との互換性のために設けられています。RZ/N2H では使われません。

4.14.5 EnDat 用共用体

使用しません。

4.14.6 EnDat 用列挙型

(1) r_endat_err_t

EnDat エンコーダ I/F のエラーコード

```
typedef enum
{
    ENDAT_SUCCESS           =0,   正常終了
    ENDAT_ERR_INVALID_ARG   ,     引数異常
    ENDAT_ERR_BUSY          ,     API を実行できない状態
    ENDAT_ERR_ACCESS        ,     API の実行順序エラー
    ENDAT_ERR_DRV                 ドライバ内部エラー
} r_endat_err_t
```

(2) r_endat_cmd_t

R_ENDAT_Control 関数を使用する時のコマンド設定

```
typedef enum
{
    ENDAT_CMD_REQ           ,     エンコーダへコマンド送信
    ENDAT_CMD_POS_STOP            エンコーダ制御部の位置値連続受信の停止
} r_endat_cmd_t
```

(3) r_endat_req_err_t

リクエストの送受信結果

```
typedef enum
{
    ENDAT_REQ_SUCCESS   =0,       データ送受信正常終了
    ENDAT_REQ_ERR                 データ送受信エラー発生
} r_endat_req_err_t
```

4.15 サンプルプログラムの説明

4.15.1 動作概要

本サンプルプログラムはバス接続した 1 台から 8 台までの A-format 仕様に準拠したエンコーダ(Nikon 社製 MAR-M50A), EnDat 2.2 準拠エンコーダ 「EQN1035」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。

・A-format

1) コンソールから入力したリクエスト情報をエンコーダへ送信(TXTRG レジスタへの書き込み動作による通常送受信)
2) エンコーダから受信したデータをコンソールに表示
3) AFMT の ELC イベント入力トリガ機能を使用してコマンドを送受信します。(入力イベントとして GPT のイベントをリンクしています。入力イベントの設定例は、「図 4-6 a_as_elctimer 関数のフローチャート」を参照してください。)

・EnDat

1) デバッガのターミナル I/O から入力したリクエストを EnDat エンコーダ(EQN1035)へ送信
2) EnDat エンコーダ(EQN1035)から受信したデータをデバッガのターミナル I/O に表示
3) EnDat I/F の ELC イベント入力トリガ機能を使用してコマンドを送受信します。(入力イベントとして GPT のイベントをリンクしています。)

(1) Dual Encoder システムブロック図

図 4-1 に Dual Encoder システムブロック図を示します。

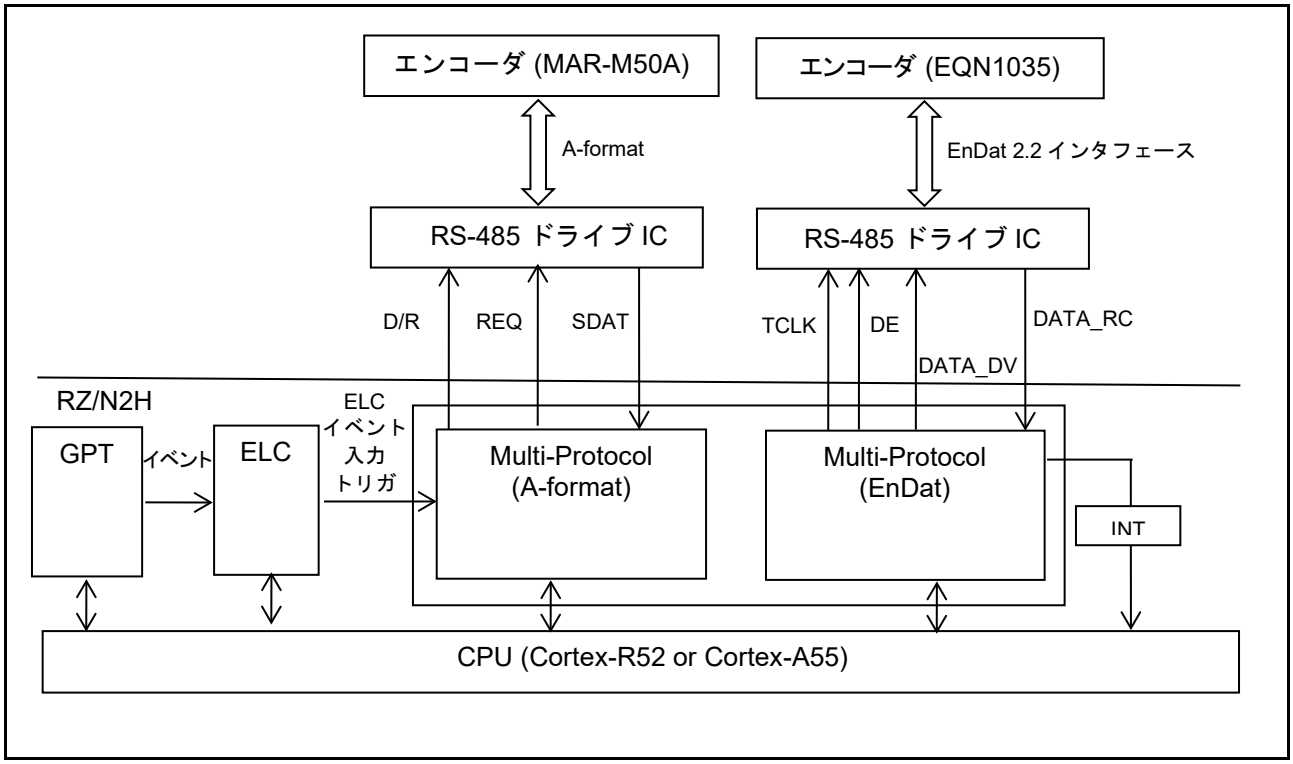


図 4-1 Dual Encoder システムブロック図

(2)　ソフトウェア構成図

図 4-2 に Dual Encoder ソフトウェア構成図を示します。

A-format ドライバには、R_A_AS_Open 関数で構成される開始処理部、R_A_AS_Close 関数で構成される終了処理部、R_A_AS_Control 関数で構成されるリクエスト送信部、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。

EnDat ドライバには、R_ENDAT_Open 関数で構成される開始処理部、R_ENDAT_Close 関数で構成される終了処理部、R_ENDAT_Control 関数で構成されるリクエスト送信部、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、A-format ドライバを制御し、リクエスト送信を行う A-format ドライバ制御部分、データ受信結果の表示を行う A-format 結果表示部分（コールバック）、EnDat ドライバを制御し、リクエスト送信を行う EnDat ドライバ制御部分、データ受信結果の表示を行う EnDat 結果表示部分（コールバック）があります。

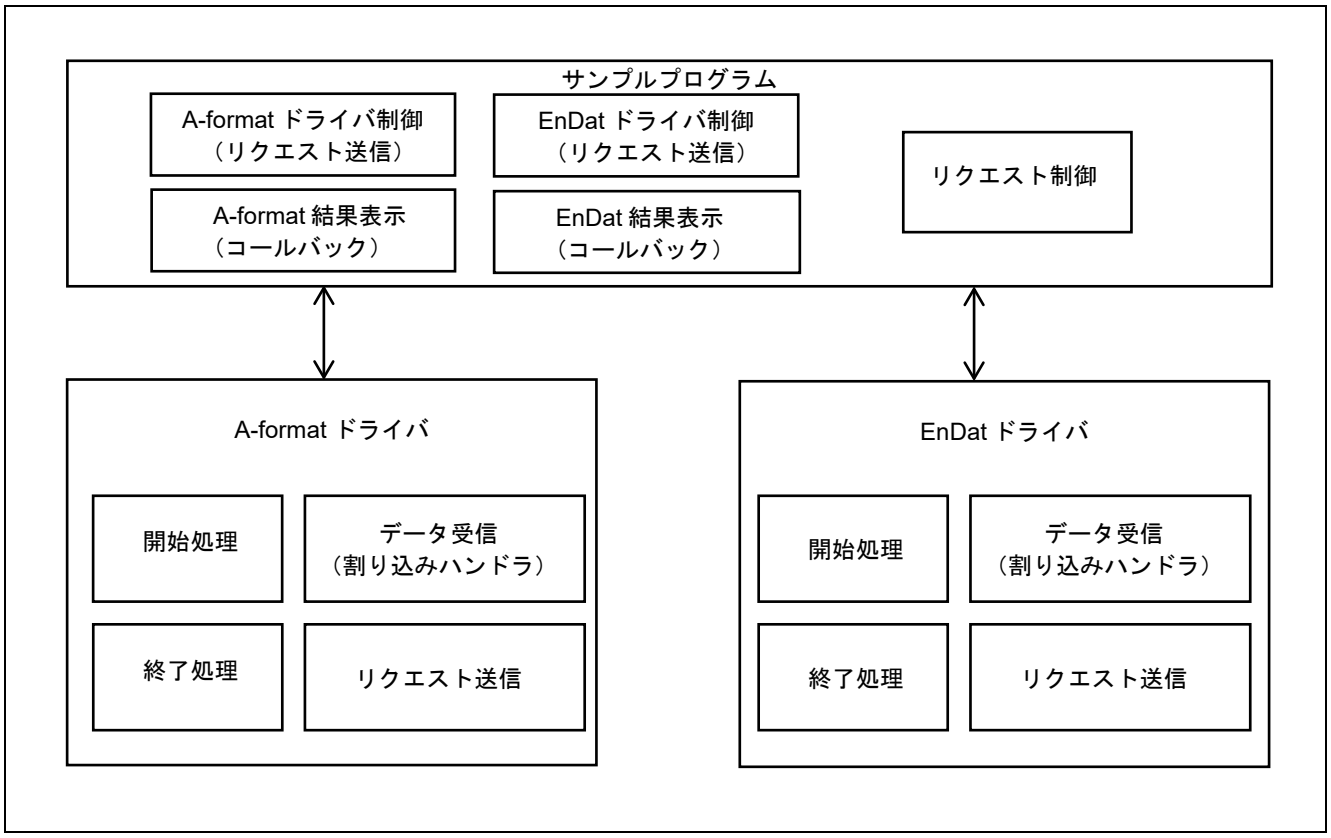


図 4-2　Dual Encoder ソフトウェア構成図

4.15.2 サンプルプログラム関数一覧

表 4.24 に主要なサンプルプログラム関数一覧を示します

表 4.24　サンプルプログラム関数一覧

関数分類	関数名	ページ番号	
		仕様	フローチャート
共通部関数	hal_entry	52	-
	enc_main	52	63
	dual_cmd_control	52	64
	get_cmd	52	-
	cmd_exit	53	-
	timer_start	53	-
	timer_stop	53	-
A-format 用関数	a_as_enc_init	53	-
	a_as_req	54	65
	a_as_elctimer	54	66
	a_as_elcstop	54	67
	a_as_txerr_callback	54	67
	a_as_rxset_callback	55	68
	a_as_rxend_callback	55	68
	a_as_elctimer_callback	55	69
EnDat 用関数	endat_power_on_wait	55	-
	enc_init_tclk_wait_callback	56	-
	enc_init_reset_wait_callback	56	-
	enc_init_mem_wait_callback	56	-
	enc_init_pram_wait_callback	56	-
	enc_init_cable_wait_callback	57	-
	endat_pos	57	70
	endat_poscon	57	71
	endat_elctimer	57	72
	endat_stop	58	73
	endat_temp	58	74
	endat_callback	58	75
	endat_poscon_callback	58	76
	endat_rdst_callback	59	76
	endat_result_display	59	-

4.15.3 サンプルプログラム関数仕様

(1) 共通部関数

(a) hal_entry

hal_entry	
概　要	Dual Encoder サンプルプログラムのエントリー関数
ヘッダ	-
宣　言	void hal_entry(void);
説　明	Dual Encoder サンプルプログラムのエントリー関数です。ここから、関数 enc_main() が呼び出されます。
引　数	なし
リターン値	なし

(b) enc_main

enc_main	
概　要	Dual Encoder サンプルプログラムのメイン関数
ヘッダ	-
宣　言	int32_t enc_main(void);
説　明	Dual Encoder サンプルプログラムのメイン関数です。詳細は、「4.15.8(1) enc_main フローチャート」参照。
引　数	なし
リターン値	0　：正常終了 0 以外　：異常終了（エンコーダ I/F ドライバのエラーコード）

(c) dual_cmd_control

dual_cmd_control	
概　要	Dual Encoder ドライバ制御関数
ヘッダ	-
宣　言	static void dual_cmd_control(void);
説　明	コンソールコマンドの入力処理を行います
引　数	なし
リターン値	なし

(d) get_cmd

get_cmd	
概　要	コンソールからコマンドを取得する関数
ヘッダ	-
宣　言	static uint32_t get_cmd(char_t *parg[], const uint32_t arg_max);
説　明	コンソールからコマンドを取得します。
引　数	parg　：コンソールから取得したコマンドを格納する配列のポインタ arg_max　：取得する最大文字列数
リターン値	コンソールから取得した文字列数

(e) cmd_exit

cmd_exit	
概　要	Dual Encoder サンプルプログラムの終了表示関数
ヘッダ	-
宣　言	static void cmd_exit(uint32_t arg_num, char_t *parg[]);
説　明	コンソールコマンド exit が入力された場合に実行される関数です。サンプルプログラムが終了したことをコンソールに表示します。
引　数	arg_num ：コンソールから入力された文字列の数 (未使用) *parg[] ：コンソールから入力された文字列の先頭アドレス (未使用)
リターン値	なし

(f) timer_start

timer_start	
概　要	GPT の周期設定/起動関数
ヘッダ	-
宣　言	static void timer_start(uint32_t ch, uint32_t us);
説　明	指定したチャネルのエンコーダに対応する GPT にタイマ周期を設定してタイマを起動します。使用していないチャネルが指定された場合には、何もせずにリターンします。
引　数	ch ：エンコーダのチャネル番号 us ：タイマ周期 [us]
リターン値	なし

(g) timer_stop

timer_stop	
概　要	GPT のタイマ停止
ヘッダ	-
宣　言	static void timer_stop(uint32_t ch);
説　明	指定したチャネルのエンコーダに対応する GPT タイマを停止します。使用していないチャネルが指定された場合には、何もせずにリターンします。
引　数	ch ：エンコーダのチャネル番号
リターン値	なし

(2) A-format 関数

(a) a_as_enc_init

a_as_enc_init	
概　要	エンコーダの初期化関数
ヘッダ	-
宣　言	static int32_t a_as_enc_init(int32_t id);
説　明	エンコーダを初期化します。コマンドデータフレーム CDF8 を連続して 8 回送信し、ステータスフラグをクリアします。
引　数	id ：エンコーダ ID R_A_AS0_ID: ch0 のエンコーダ ID を指定 R_A_AS1_ID: ch1 のエンコーダ ID を指定 ： R_A_AS15_ID: ch15 のエンコーダ ID を指定
リターン値	0：正常終了 0 以外：異常終了（エンコーダ I/F のエラーコード）
注　意	本サンプルプログラムでは、チャネル 8, チャネル 12, チャネル 15 は無効です。

(b) a_as_req

a_as_req	
概　要	req コンソールコマンド関数
ヘッダ	-
宣　言	static void a_as_req(uint32_t arg_num, char_t *parg[]);
説　明	コンソールコマンド req が入力された場合に実行される関数です。詳細は「4.15.8(3) a_as_req フローチャート」と「4.15.10 コンソールコマンド」を参照してください。
引　数	arg_num ：コンソールから入力された文字列の数 *parg[] ：コンソールから入力された文字列の先頭アドレス。
リターン値	なし

(c) a_as_elctimer

a_as_elctimer	
概　要	elctimer コンソールコマンド関数
ヘッダ	-
宣　言	static void a_as_elctimer (uint32_t arg_num, char_t *parg[]);
説　明	コンソールコマンド elctimer が入力された場合に実行される関数です。詳細は「4.15.8(4) a_as_elctimer フローチャート」と「4.15.10 コンソールコマンド」を参照してください。
引　数	arg_num ：コンソールから入力された文字列の数 *parg[] ：コンソールから入力された文字列の先頭アドレス。
リターン値	なし

(d) a_as_elcstop

a_as_elcstop	
概　要	elcstop コンソールコマンド関数
ヘッダ	-
宣　言	static void a_as_elcstop(uint32_t arg_num, char_t *parg[]);
説　明	コンソールコマンド elcstop が入力された場合に実行される関数です。詳細は「4.15.8(5) a_as_elcstop フローチャート」と「4.15.10 コンソールコマンド」を参照してください。
引　数	arg_num ：コンソールから入力された文字列の数 *parg[] ：コンソールから入力された文字列の先頭アドレス。
リターン値	なし

(e) a_as_txerr_callback

a_as_txerr_callback	
概　要	コンソールコマンド req で送信エラー発生時のコールバック関数
ヘッダ	-
宣　言	static void a_as_txerr_callback(r_a_as_result_t *p_result);
説　明	コンソールコマンド req が入力された場合に実行されるコールバック関数です。通常受信の A-format リクエストの送受信結果を a_as_result 変数に保存し、タイムアウトエラーが発生したことを通知します。詳細は「4.15.8(6) a_as_txerr_callback フローチャート」を参照してください。
引　数	*p_result ：リクエストの送受信結果が格納されている RAM の先頭アドレス
リターン値	なし

(f) a_as_rxset_callback

a_as_rxset_callback	
概　要	コンソールコマンド req で受信データ設定完了時のコールバック関数
ヘッダ	-
宣　言	static void a_as_rxset_callback(r_a_as_result_t *p_result);
説　明	コンソールコマンド req が入力された場合に実行されるコールバック関数です。通常受信の A-format リクエストの送受信結果を a_as_result 変数に保存し、受信データの設定が完了したことを通知します。詳細は「4.15.8(7) a_as_rxset_callback フローチャート」を参照してください。
引　数	*p_result　：リクエストの送受信結果が格納されている RAM の先頭アドレス
リターン値	なし

(g) a_as_rxend_callback

a_as_rxend_callback	
概　要	コンソールコマンド req でデータ送受信完了時のコールバック関数
ヘッダ	-
宣　言	static void a_as_rxend_callback(r_a_as_result_t *p_result);
説　明	コンソールコマンド req が入力された場合に実行されるコールバック関数です。通常受信の A-format リクエストに対するデータ送受信が完了したことを通知します。詳細は「4.15.8(8) a_as_rxend_callback フローチャート」を参照してください。
引　数	*p_result　：リクエストの送受信結果が格納されている RAM の先頭アドレス
リターン値	なし

(h) a_as_elctimer_callback

a_as_elctimer_callback	
概　要	elctimer コンソールコマンドのコールバック関数
ヘッダ	-
宣　言	static void a_as_elctimer_callback (r_a_as_result_t * p_result);
説　明	コンソールコマンド elctimer が入力された場合に実行されるコールバック関数です。リクエストの送受信結果を a_as_ti_result 変数と、a_as_ti_data 変数に保存します。詳細は「4.15.8(9) a_as_elctimer_callback フローチャート」を参照してください。
引　数	*p_result　：リクエストの送受信結果が格納されている RAM の先頭アドレス
リターン値	なし

(3) EnDat 関数

(a) endat_power_on_wait

endat_power_on_wait	
概　要	エンコーダ電源投入後の待機時間生成関数
ヘッダ	-
宣　言	static void endat_power_on_wait(void);
説　明	エンコーダの電源投入後に必要な 1.3s の待機時間を生成します。
引　数	なし
リターン値	なし

(b) enc_init_tclk_wait_callback

enc_init_tclk_wait_callback	
概　要	TCLK 端子出力開始後の待機時間生成関数
ヘッダ	-
宣　言	static void enc_init_tclk_wait_callback(void);
説　明	接続されたエンコーダの初期化処理でエンコーダの TCLK 端子出力開始処理後に待機する時間 100us を生成するコールバック関数です。 詳細は「4.9.1 enc_init_tclk_wait_callback」参照。
引　数	なし
リターン値	なし

(c) enc_init_reset_wait_callback

enc_init_reset_wait_callback	
概　要	エンコーダリセット後の待機時間生成関数
ヘッダ	-
宣　言	static void enc_init_reset_wait_callback(void);
説　明	接続されたエンコーダの初期化処理でエンコーダのリセット処理後に待機する時間 60ms を生成するコールバック関数です。 詳細は「4.9.2 enc_init_reset_wait_callback」参照。
引　数	なし
リターン値	なし

(d) enc_init_mem_wait_callback

enc_init_mem_wait_callback	
概　要	エンコーダのメモリエリア選択処理の待機時間生成関数
ヘッダ	-
宣　言	static void enc_init_mem_wait_callback(void);
説　明	接続されたエンコーダの初期化処理でメモリエリアを選択する処理のタイムアウトエラー検出用待機時間 743us を生成するコールバック関数です。 詳細は「4.9.3 enc_init_mem_wait_callback 」参照
引　数	なし
リターン値	なし

(e) enc_init_pram_wait_callback

enc_init_pram_wait_callback	
概　要	エンコーダのパラメータ送受信処理の待機時間生成関数
ヘッダ	-
宣　言	static void enc_init_pram_wait_callback(void);
説　明	接続されたエンコーダの初期化処理でエンコーダがパラメータを送受信する処理のタイムアウトエラー検出用待機時間 13ms を生成するコールバック関数です。 詳細は「4.9.4 enc_init_pram_wait_callback」参照
引　数	なし
リターン値	なし

(f) enc_init_cable_wait_callback

enc_init_cable_wait_callback	
概　要	エンコーダのケーブル伝送遅延測定処理の待機時間生成関数
ヘッダ	-
宣　言	static void enc_init_cable_wait_callback(void);
説　明	接続されたエンコーダの初期化処理でケーブル伝送遅延を測定する処理のタイムアウトエラー検出用待機時間 588us を生成するコールバック関数です。 詳細は「4.9.5 enc_init_cable_wait_callback」参照
引　数	なし
リターン値	なし

(g) endat_pos

endat_pos		
概　要	エンコーダから位置値を取得する関数	
ヘッダ	-	
宣　言	static void endat_pos(uint32_t arg_num, char_t *parg[]);	
説　明	コンソールコマンド pos が入力された場合に実行される関数です。エンコーダから位置値を取得します。	
引　数	arg_num	：コンソールから入力された文字列の数 (未使用)
	*parg[]	：コンソールから入力された文字列の先頭アドレス (未使用)
リターン値	なし	

(h) endat_poscon

endat_poscon		
概　要	エンコーダから連続して位置値を取得する関数	
ヘッダ	-	
宣　言	static void endat_poscon(uint32_t arg_num, char_t *parg[]);	
説　明	コンソールコマンド poscon が入力された場合に実行される関数です。Continuous モードを使って、エンコーダから連続して位置値を取得します。	
引　数	arg_num	：コンソールから入力された文字列の数 (未使用)
	*parg[]	：コンソールから入力された文字列の先頭アドレス (未使用)
リターン値	なし	

(i) endat_elctimer

endat_elctimer		
概　要	エンコーダから ELC イベントに同期して連続して位置値を取得する関数	
ヘッダ	-	
宣　言	static void endat_elctimer(uint32_t arg_num, char_t *parg[]);	
説　明	コンソールコマンド elctimer が入力された場合に実行される関数です。ELC モードを使って、エンコーダから ELC イベントに同期して連続して位置値を取得します。	
引　数	arg_num	：コンソールから入力された文字列の数
	*parg[]	：コンソールから入力された文字列の先頭アドレス
リターン値	なし	

(j) endat_stop

endat_stop	
概　要	エンコーダからの連続した位置値の取得を停止させる関数
ヘッダ	-
宣　言	static void endat_stop(uint32_t arg_num, char_t *parg[]);
説　明	コンソールコマンド stop が入力された場合に実行される関数です。Continuous モードで動作中には、エンコーダからの連続した位置値の送信を停止させます。また、ELC モードで動作中には、ELC モードを解除して連続した位置値取得コマンド発行を停止します。 エンコーダの連続位置値送信を停止させてから、過去 10 個分の位置値を表示します。
引　数	arg_num　：コンソールから入力された文字列の数 (未使用) *parg[]　：コンソールから入力された文字列の先頭アドレス (未使用)
リターン値	なし

(k) endat_temp

endat_temp	
概　要	エンコーダから温度情報を取得する関数
ヘッダ	-
宣　言	static void endat_temp(uint32_t arg_num, char_t *parg[]);
説　明	コンソールコマンド temp が入力された場合に実行される関数です。エンコーダから温度情報を取得します
引　数	arg_num　：コンソールから入力された文字列の数 (未使用) *parg[]　：コンソールから入力された文字列の先頭アドレス (未使用)
リターン値	なし

(l) endat_callback

endat_callback	
概　要	エンコーダへのリクエスト送信結果通知のコールバック関数
ヘッダ	-
宣　言	static void endat_callback(r_endat_result_t *p_result, r_endat_protocol_err_t *p_err);
説　明	結果をメモリに保存します。
引　数	p_result　：送受信結果 p_err　：EnDat I/F およびエンコーダのエラー情報
リターン値	なし

(m) endat_poscon_callback

endat_poscon_callback	
概　要	エンコーダへのリクエスト送信結果通知のコールバック関数
ヘッダ	-
宣　言	static void endat_poscon_callback(r_endat_result_t *p_result, r_endat_protocol_err_t *p_err);
説　明	連続して取得した結果をメモリに保存します。
引　数	p_result　：送受信結果 p_err　：EnDat I/F およびエンコーダのエラー情報
リターン値	なし

(n) endat_rdst_callback

endat_rdst_callback	
概　要	次のデータ通信が開始可能であることを通知するコールバック関数
ヘッダ	-
宣　言	static void endat_rdst_callback(void);
説　明	データ受信が完了し、次のデータ通信が可能であることを通知します。Continuous モードや ELC モードで動作中は、データ受信が完了するたびにコールされます。 取得完了フラグを立てます。
引　数	なし
リターン値	なし

(o) endat_result_display

endat_result_display		
概　要	データ受信結果を表示する関数	
ヘッダ	-	
宣　言	static void endat_result_display(r_endat_result_t *p_result, r_endat_protocol_err_t *p_err);	
説　明	エンコーダへのリクエスト送信に対するデータ受信結果を表示します。	
引　数	p_result	: 送受信結果
	p_err	: EnDat I/F およびエンコーダのエラー情報
リターン値	なし	

4.15.4 A-format サンプルプログラムの変数一覧

表 4.25 に static 型変数を示します。const 型は使用しません。

表 4.25　A-format の static 型変数

型	変数名	内容
bool	a_as_flg	送受信完了フラグ (true：送受信完了、false：送受信中)
r_a_as_result_t	a_as_result[A_AS_ENC_NUM]	データ取得結果を格納します。
bool	a_as_elc_flg	ELC イベント入力トリガフラグ (true：ELC イベント入力トリガ動作中、false：ELC イベント入力トリガ動作停止)
bool	elc_trans_flg	ELC イベント入力トリガ動作中のデータ送受信フラグ (true：送受信中、false：送受信完了)
r_a_as_req_t	a_as_req_elc	ELC イベント入力トリガ動作中のリクエスト情報を格納します。

4.15.5 A-format サンプルプログラムの定数一覧

表 4.26 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 4.26　A-format の主要な定数

定数名	設定値	内容
A_AS_ENC_NUM	8	エンコーダの接続数

4.15.6 EnDat サンプルプログラムの変数一覧

表 4.27 に static 型変数を示します。const 型は使用しません。

表 4.27　EnDat の static 型変数

型	変数名	内容	使用関数
bool	endat_flg	送受信完了フラグ (true: 送受信完了, false: 送受信中)	endat_pos endat_poscon endat_elctimer endat_stop endat_temp endat_callback endat_rdst_callback
bool	endat_elc_flg	ELC モード動作中フラグ (true: ELC モード動作中, false: ELC モード動作中ではない)	endat_pos endat_poscon endat_elctimer endat_stop
r_endat_result_t	*p_endat_result	データ取得結果を格納したアドレス	endat_pos endat_temp endat_callback
r_endat_protocol_err_t	*p_endat_err	エラー情報を格納したアドレス	endat_pos endat_temp endat_callback
r_endat_req_err_t	poscon_err[ENDAT_POS_NUM]	連続取得した位置値のエラー有無 要素数 10 の配列をリングバッファとして、最新の 10 回分の取得結果を格納します。	endat_poscon endat_elctimer endat_stop endat_poscon_callback
uint64_t	poscon[ENDAT_POS_NUM]	連続取得した位置値 要素数 10 の配列をリングバッファとして、最新の 10 回分の取得結果を格納します。	endat_poscon endat_elctimer endat_stop endat_poscon_callback
uint8_t	poscon_valid	poscon, poscon_err 配列の有効要素数 配列内に格納した位置値の有効要素数を示します。	endat_poscon endat_elctimer endat_stop endat_poscon_callback
uint8_t	poscon_num	poscon, poscon_err 配列の更新位置インデックス 次に取得した位置値によって更新するインデックスを示します。	endat_poscon endat_elctimer endat_stop endat_poscon_callback
bool	poscon_empty	poscon, poscon_err 配列の空き情報 (true: 空きあり, false: 空きなし)	endat_poscon endat_elctimer endat_poscon_callback
int32_t	endat_cur_id	EnDat I/F ドライバ使用 ID	enc_main endat_cmd_control endat_pos endat_poscon endat_elctimer endat_stop endat_temp

4.15.7 EnDat サンプルプログラムの定数一覧

表 4.28 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 4.28　EnDat の 主要な定数

定数名	設定値	内容
ENDAT_ENC_TSAT_WAIT	1300u	電源投入後の待機時間 (1.3s)
ENDAT_ENC_100US_WAIT	100u	TCLK 端子出力開始後の待機時間 (100us)
ENDAT_ENC_INIT_RESET_WAIT	60u	エンコーダのリセット処理後に待機する時間 (60ms)
ENDAT_ENC_INIT_MEM_WAIT	743u	エンコーダ初期化で、メモリエリアを選択する処理のタイムアウトエラー検出用待機時間 (743us)
ENDAT_ENC_INIT_PRAM_WAIT	13u	エンコーダ初期化で、パラメータを送受信する処理のタイムアウトエラー検出用待機時間 (13ms)
ENDAT_ENC_INIT_CABLE_WAIT	588u	エンコーダ初期化で、ケーブル伝送遅延を測定する処理のタイムアウトエラー検出用待機時間 (588us)
ENDAT_WDG_MAX	127u	Watchdog Timer 設定最大値
ENDAT_POS_NUM	10u	連続受信した位置値格納用配列の要素数
ENDAT_TEMP_SCA_FAC	0.1	温度データ分解能
ENDAT_TEMP_ABS_ZERO	273.2	温度データ単位変換用定数

4.15.8 メイン処理のフローチャート

(1) enc_main フローチャート

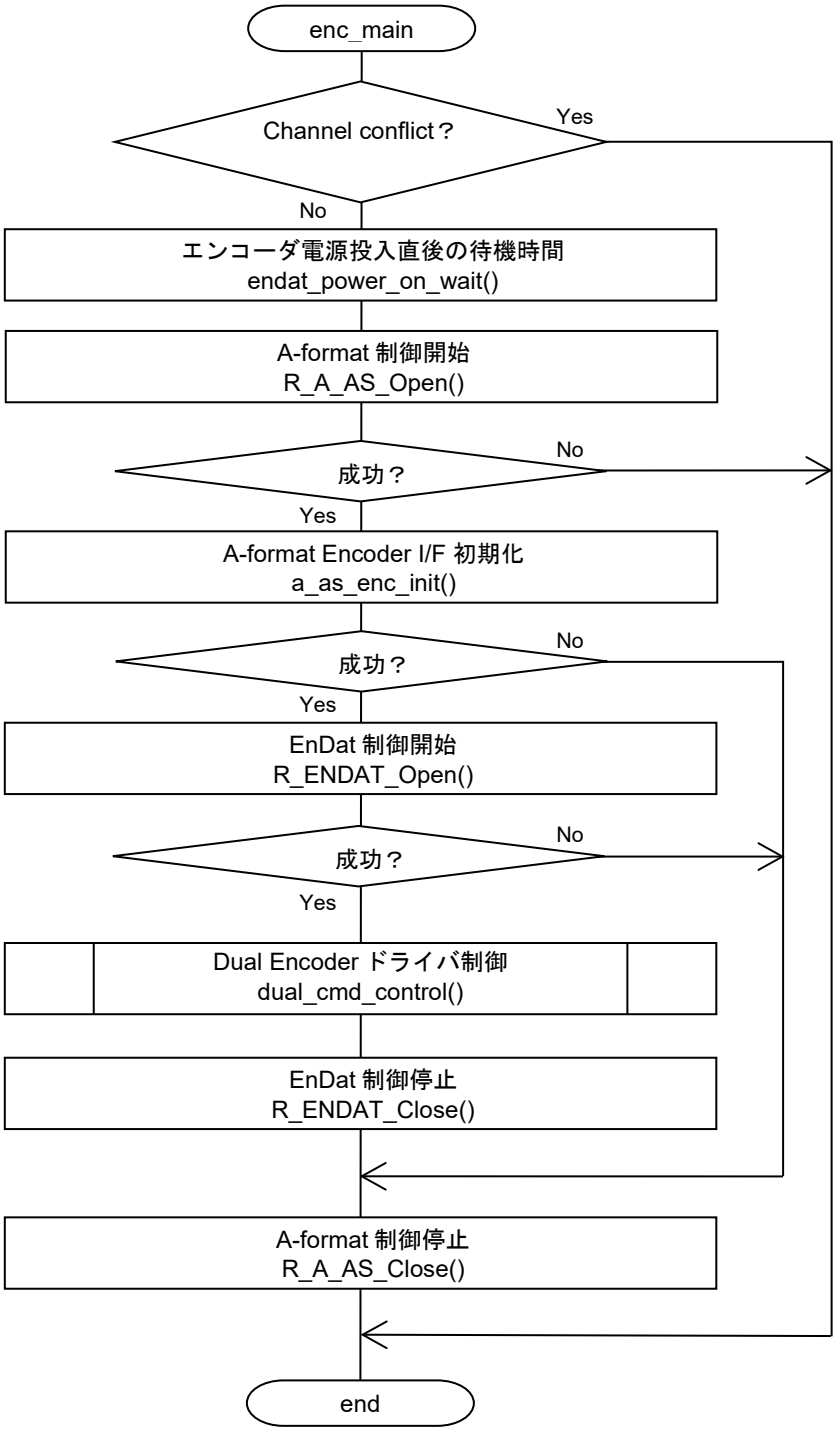


図 4-3 enc_main 関数のフローチャート

(2) dual_cmd_control フローチャート

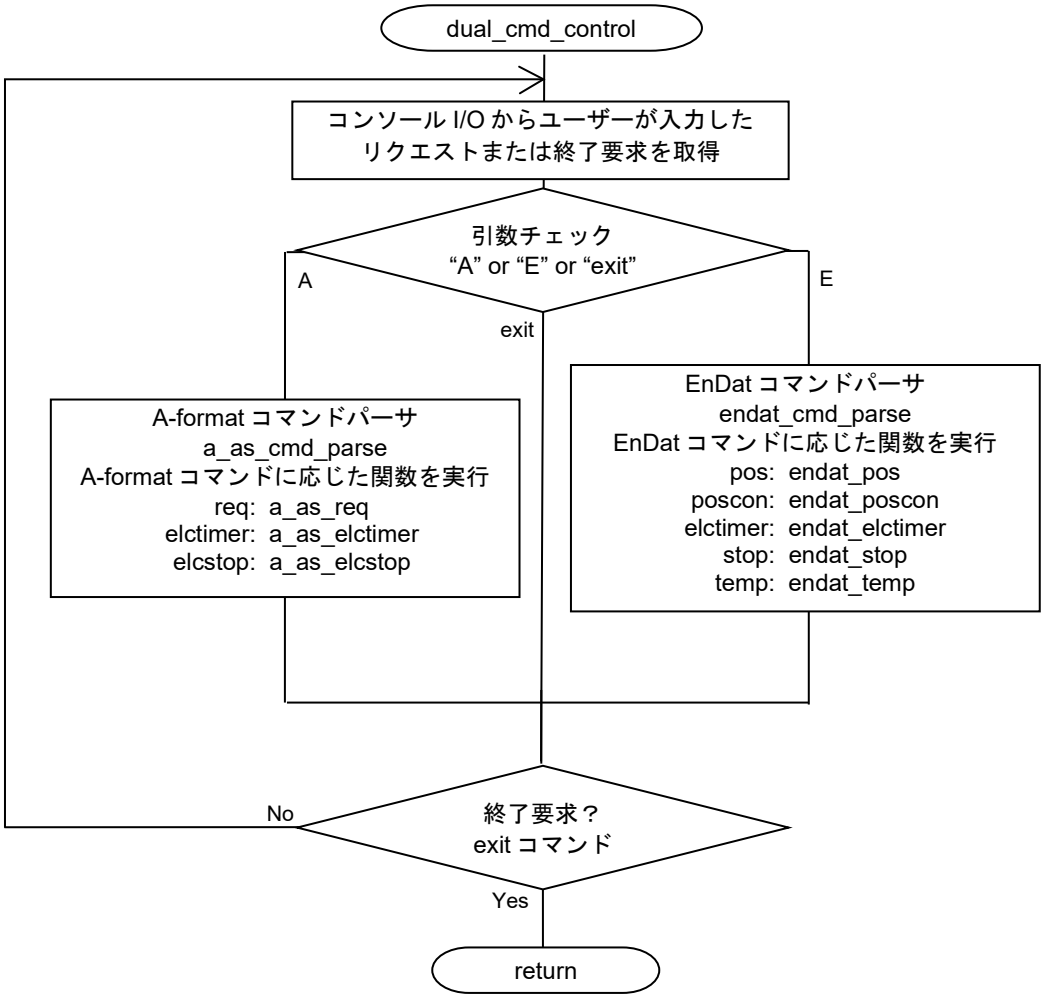


図 4-4 dual_cmd_control 関数のフローチャート

(3) a_as_req フローチャート

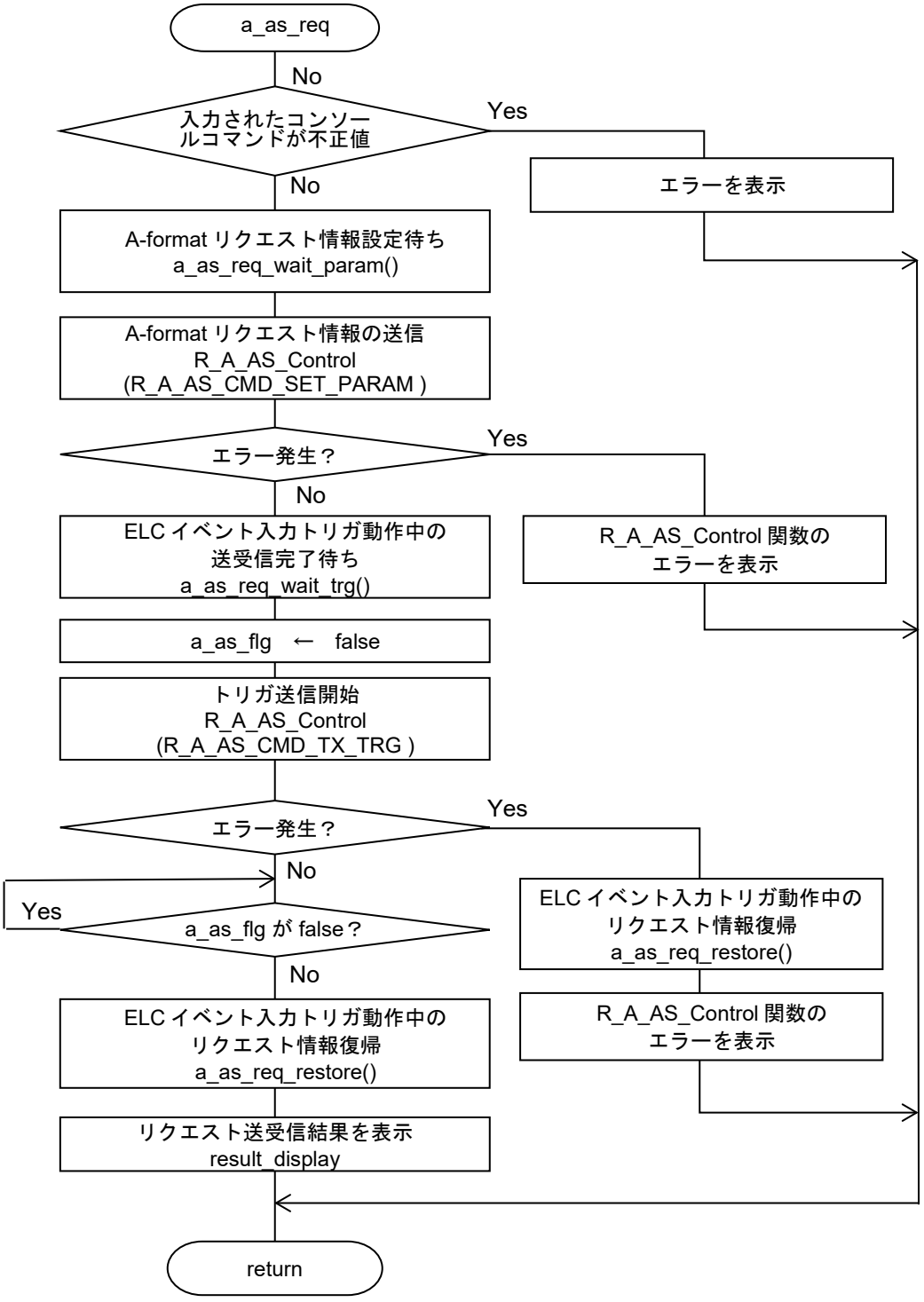


図 4-5 a_as_req 関数のフローチャート

(4) a_as_elctimer フローチャート

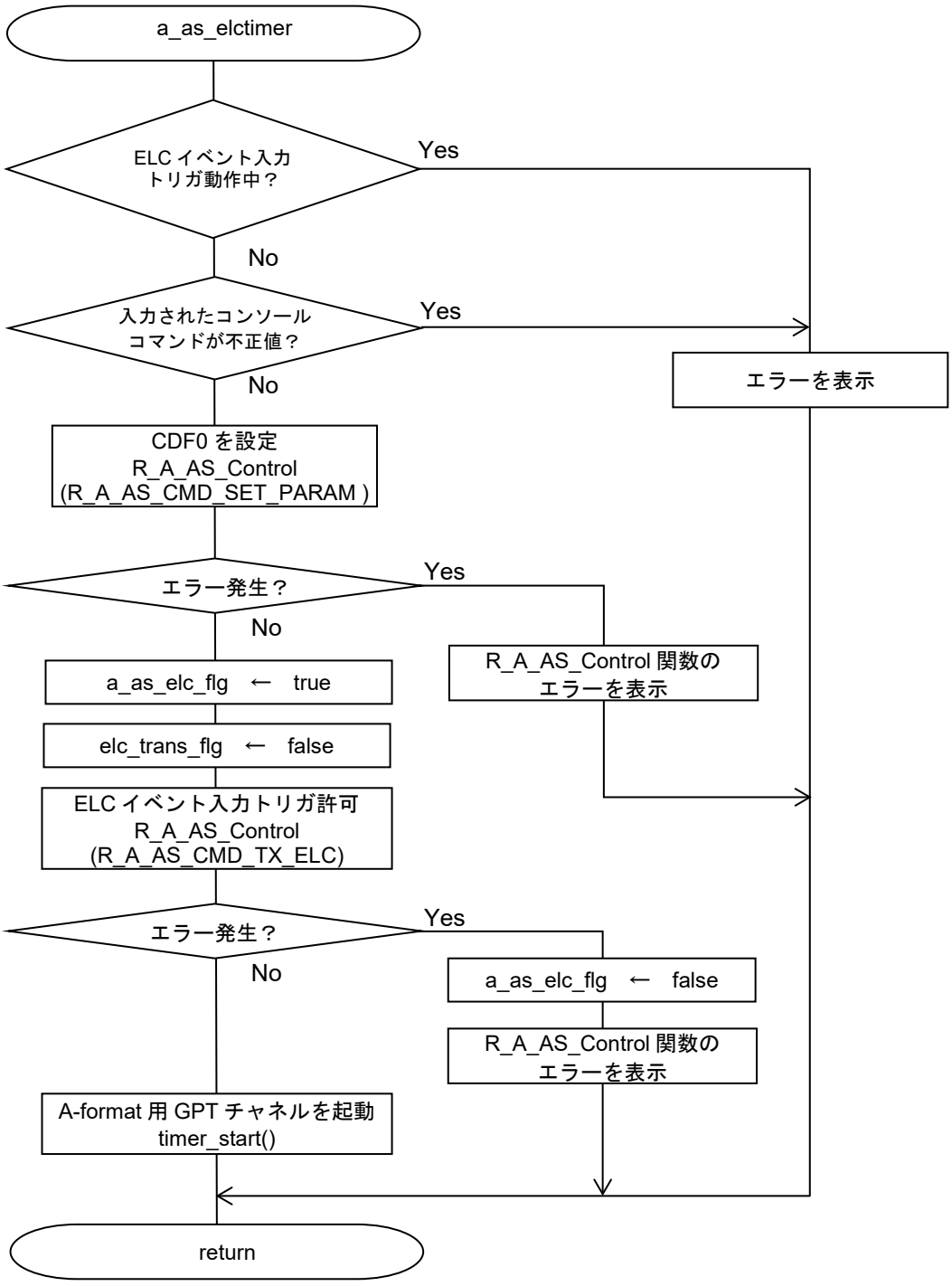


図 4-6 a_as_elctimer 関数のフローチャート

(5) a_as_elcstop フローチャート

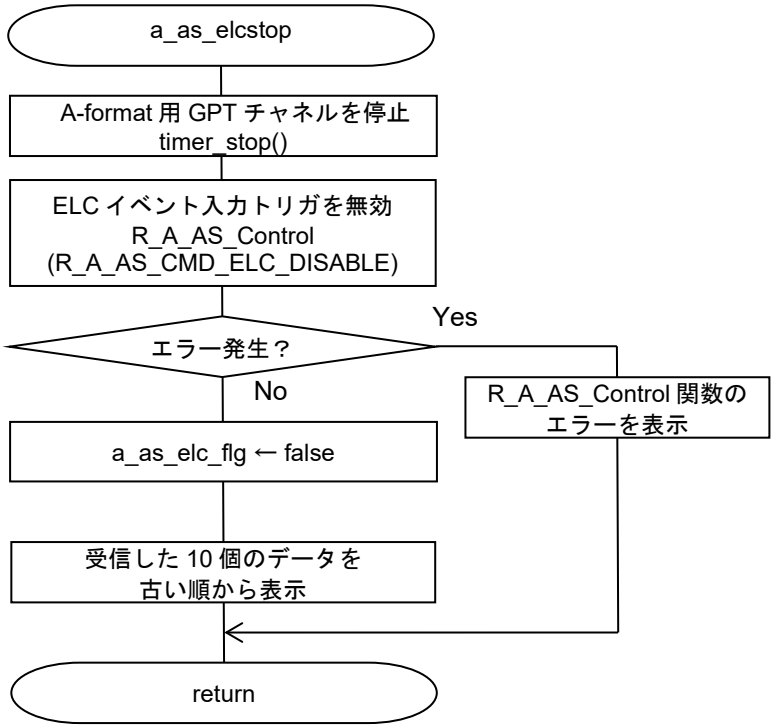


図 4-7 a_as_elcstop 関数のフローチャート

(6) a_as_txerr_callback フローチャート

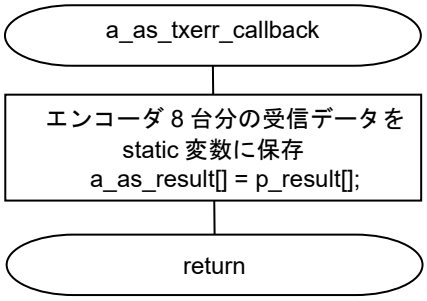


図 4-8 a_as_txerr_callback 関数のフローチャート

(7) a_as_rxset_callback フローチャート

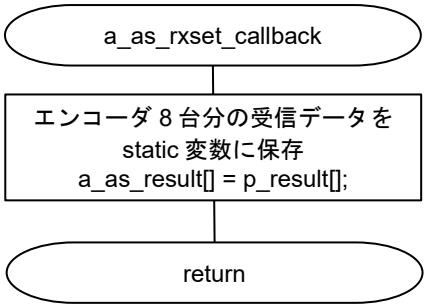


図 4-9 a_as_rxset_callback 関数のフローチャート

(8) a_as_rxend_callback フローチャート

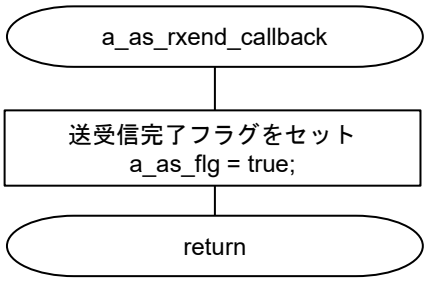


図 4-10 a_as_rxend_callback 関数のフローチャート

(9) a_as_elctimer_callback フローチャート

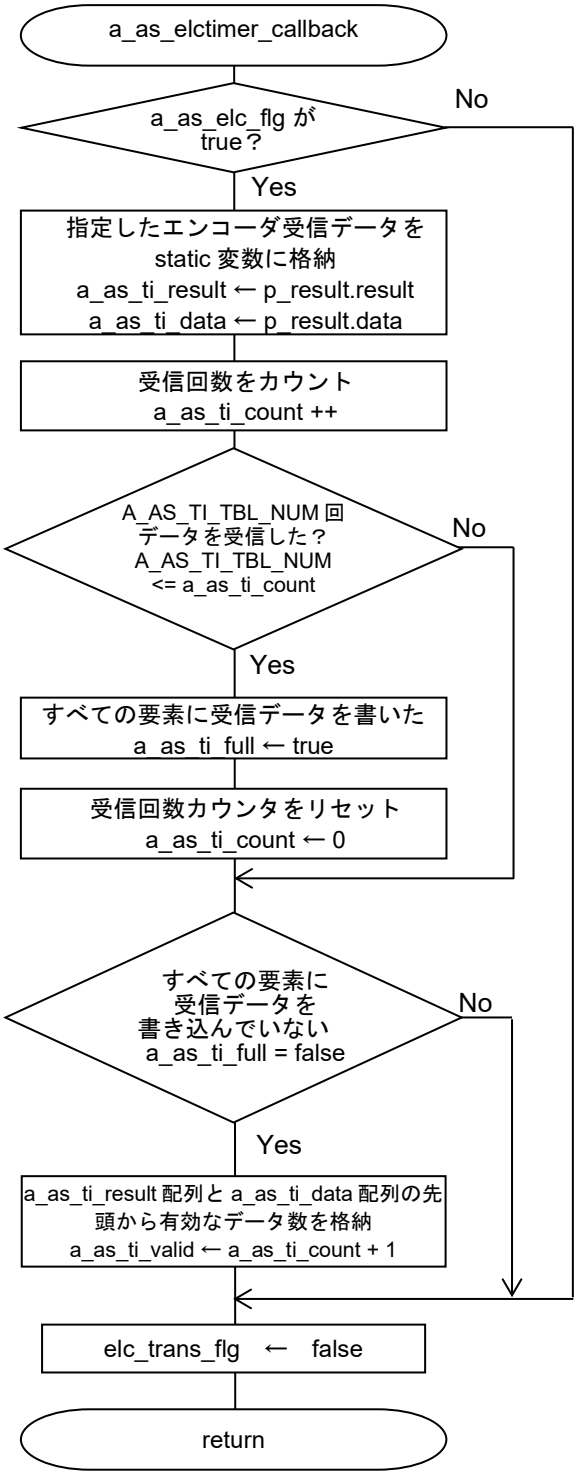


図 4-11 a_as_elctimer_callback 関数のフローチャート

(10) endat_pos フローチャート

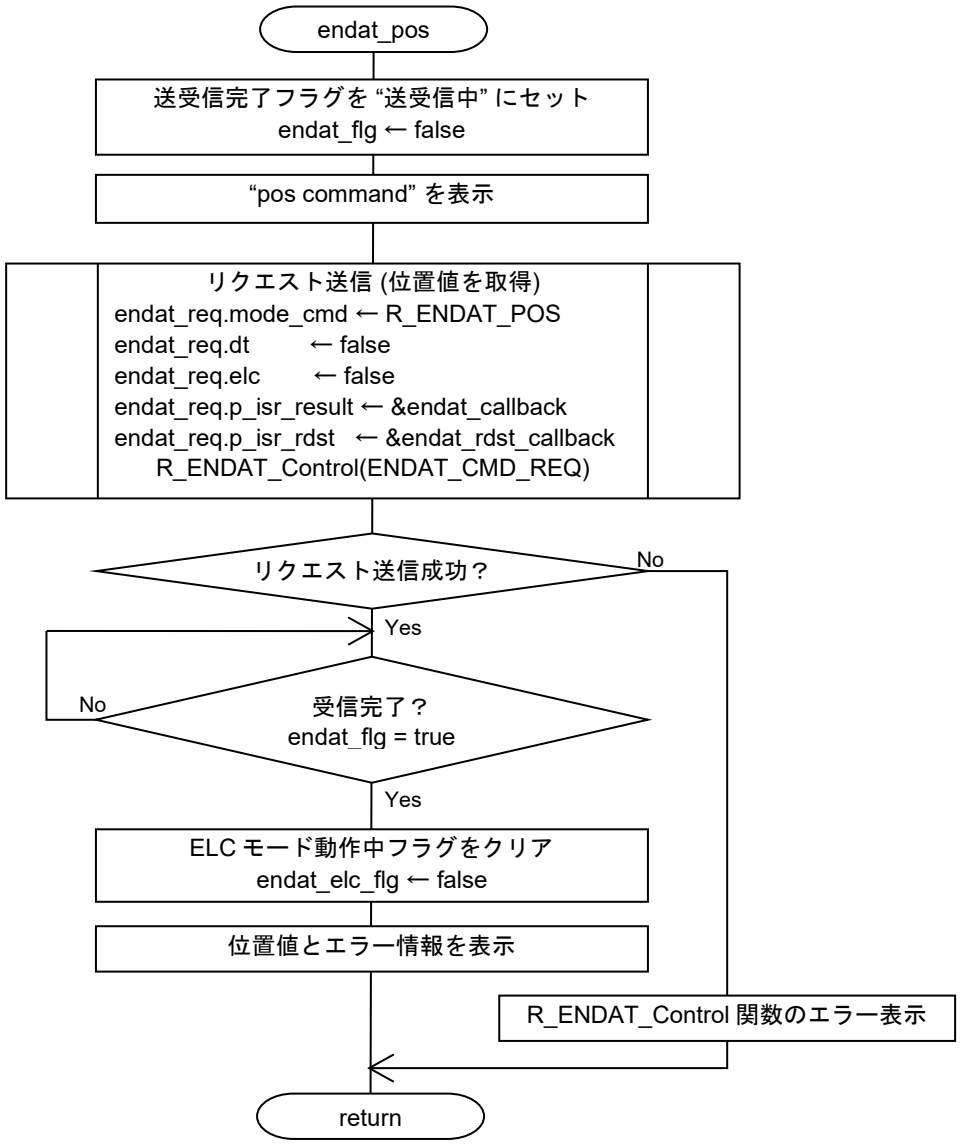


図 4-12　endat_pos 関数のフローチャート

(11) endat_poscon フローチャート

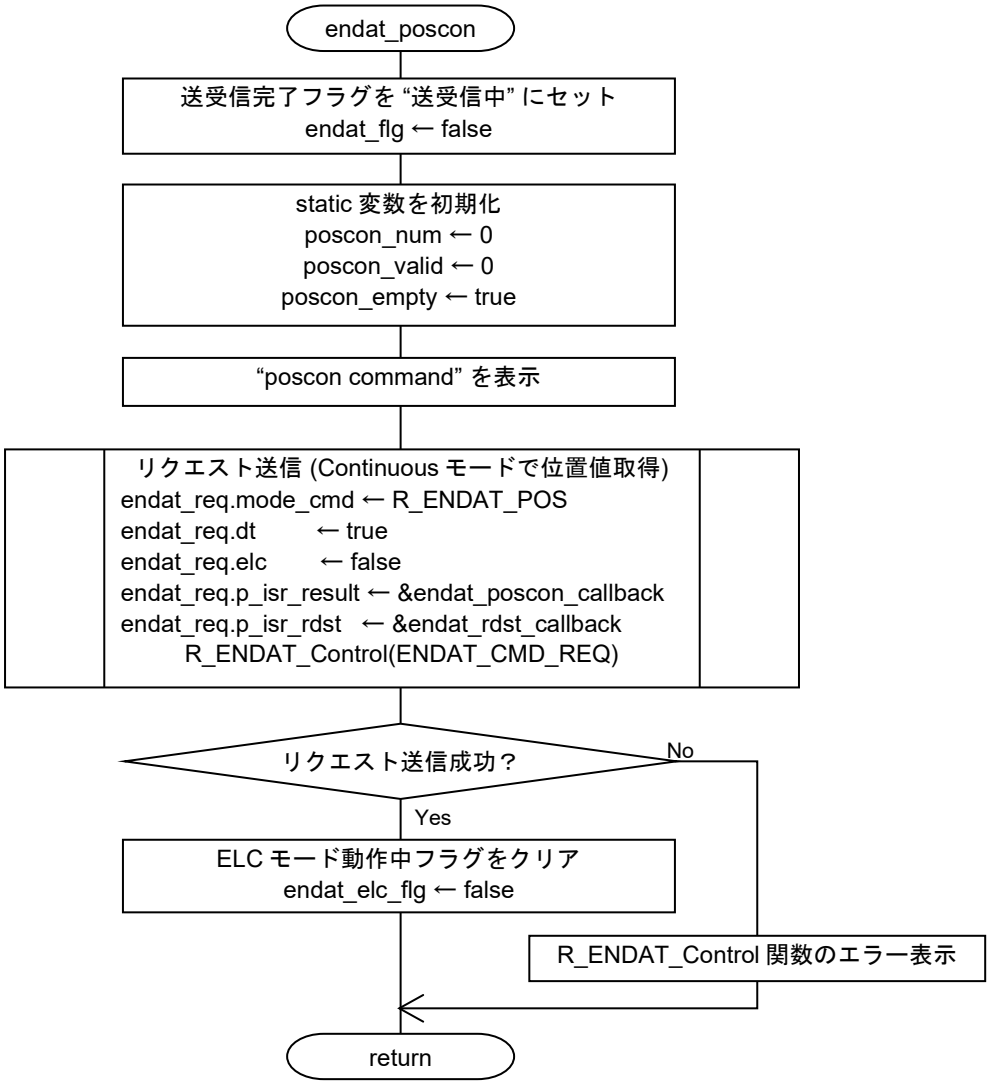


図 4-13　endat_poscon 関数のフローチャート

(12) endat_elctimer フローチャート

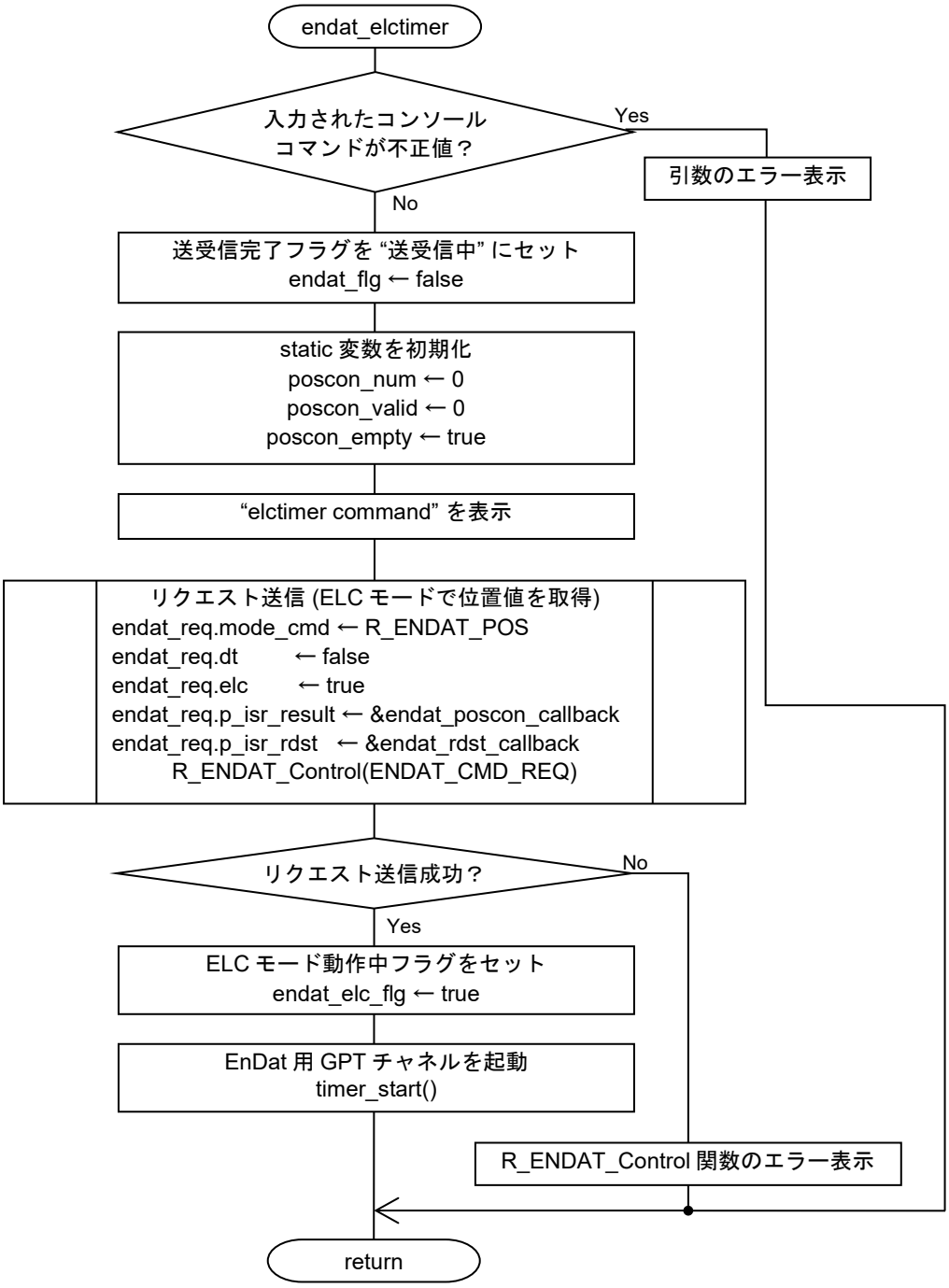


図 4-14　endat_elctimer 関数のフローチャート

(13) endat_stop フローチャート

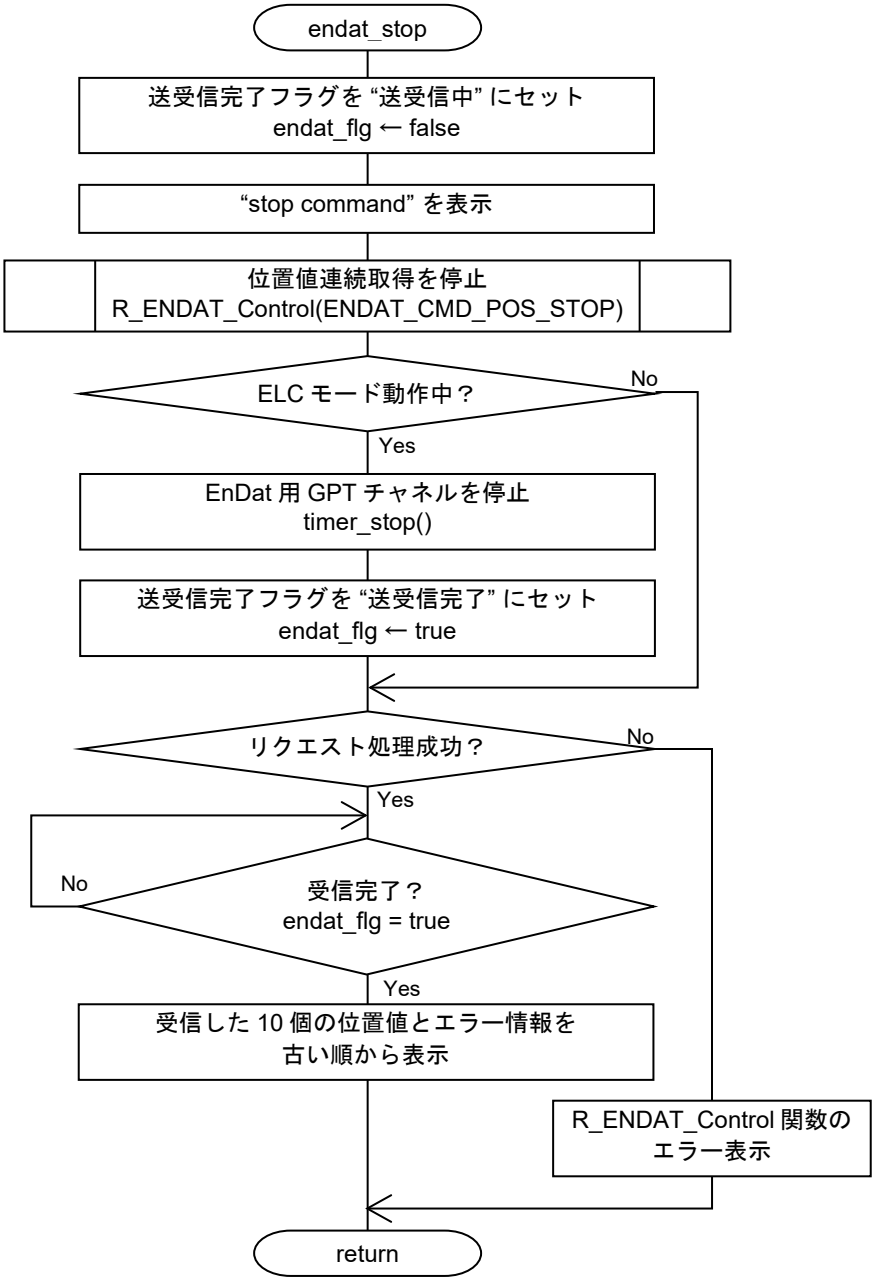


図 4-15　endat_stop 関数のフローチャート

(14) endat_temp フローチャート

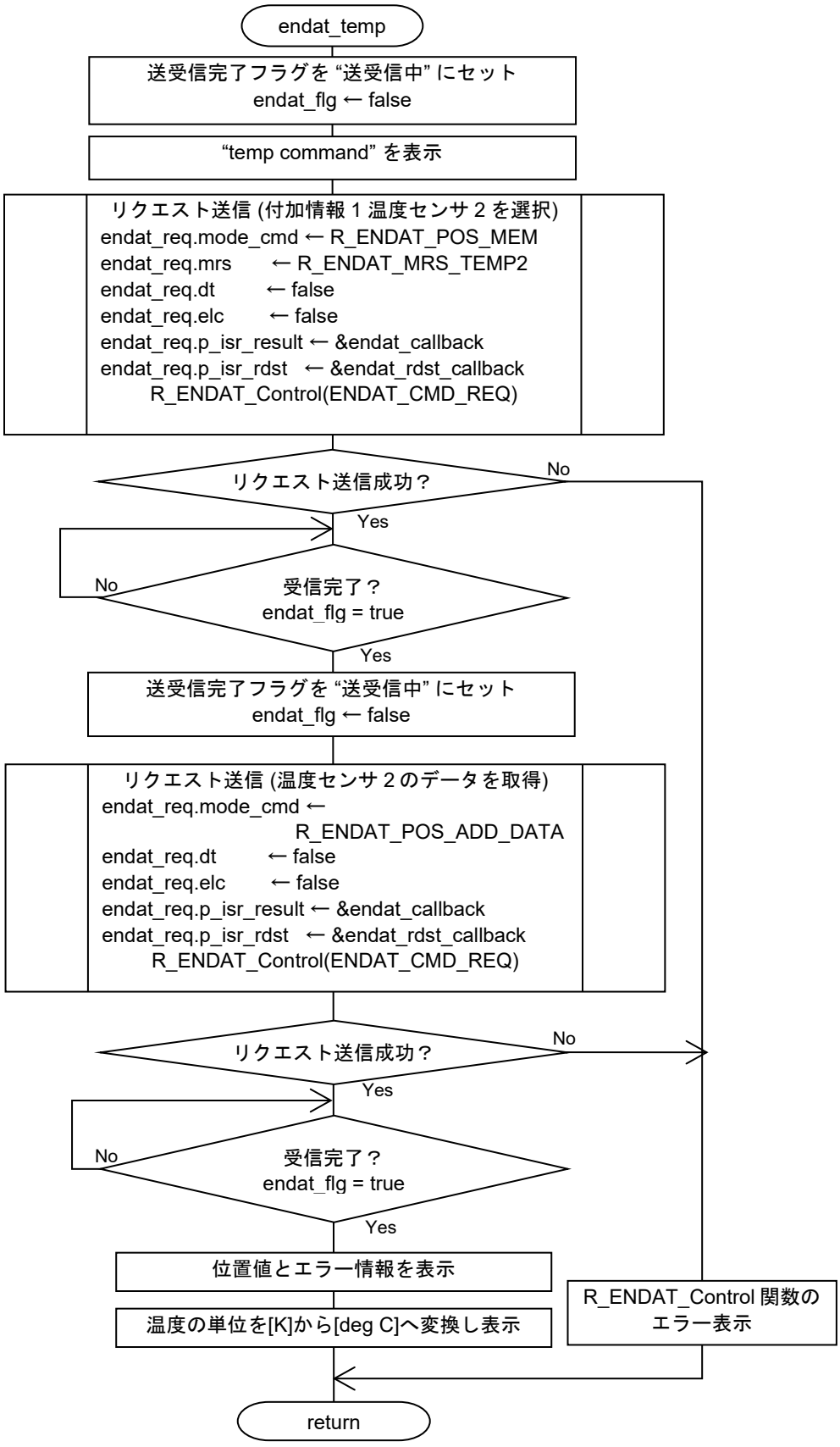


図 4-16　endat_temp 関数のフローチャート

(15) endat_callback フローチャート

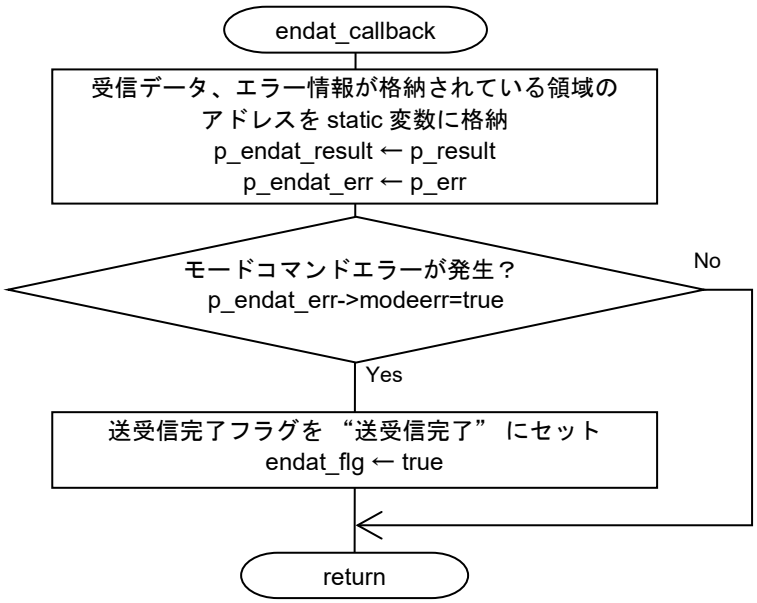


図 4-17　endat_callback 関数のフローチャート

(16) endat_poscon_callback フローチャート

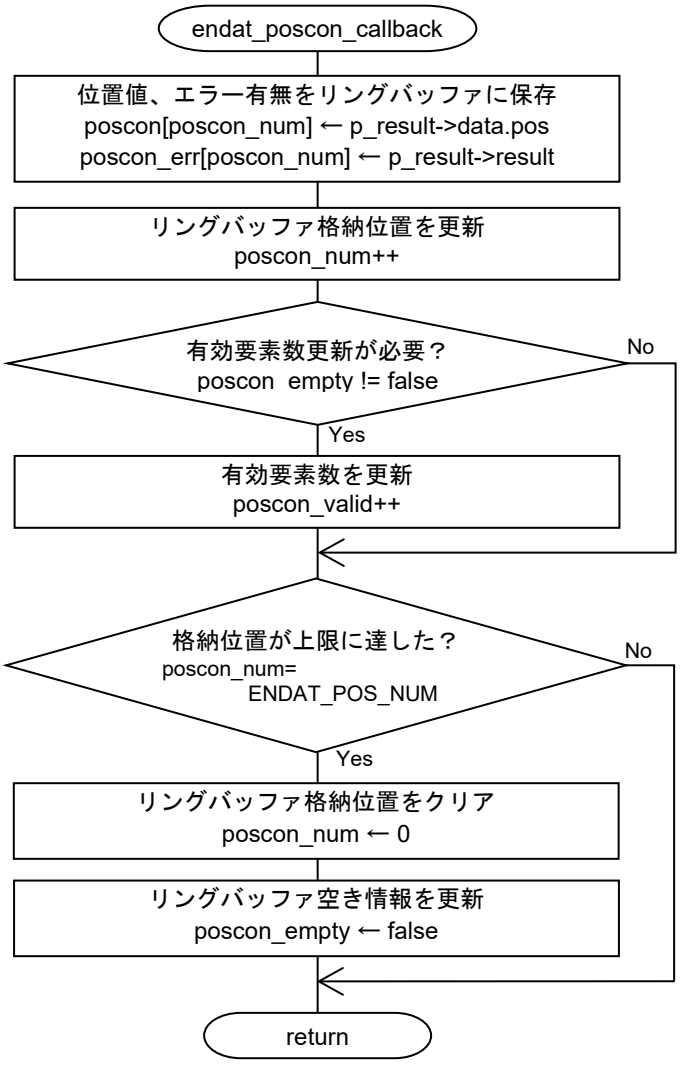


図 4-18　endat_poscon_callback 関数のフローチャート

(17) endat_rdst_callback フローチャート

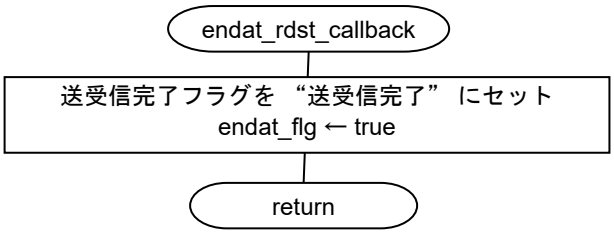


図 4-19　endat_rdst_callback 関数のフローチャート

4.15.9 Dual Encoder 開始シーケンス

(1) Dual Encoder 開始シーケンス

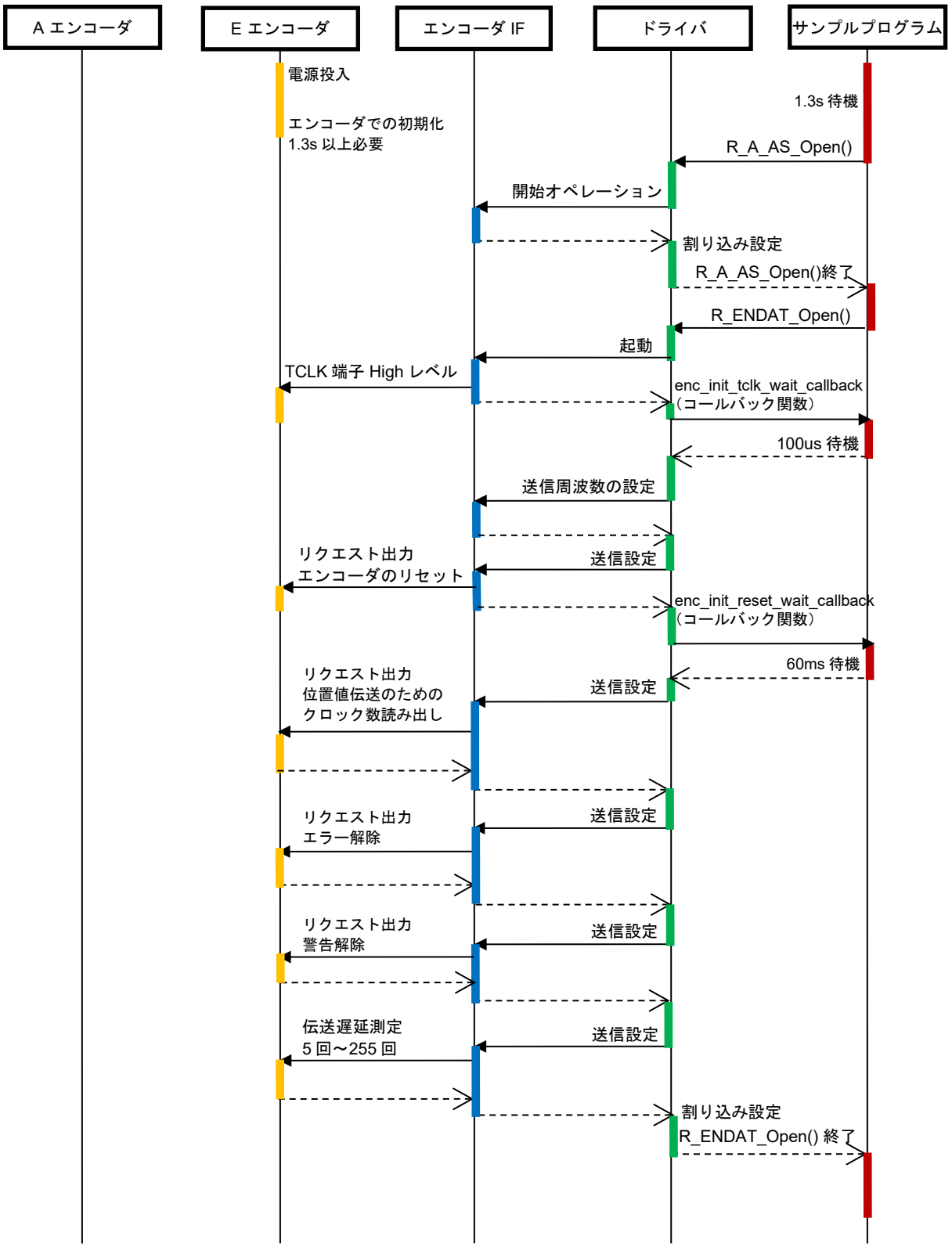


図 4-20 Dual Encoder 開始シーケンス図

(2) A-format リクエスト送信とデータ受信のシーケンス

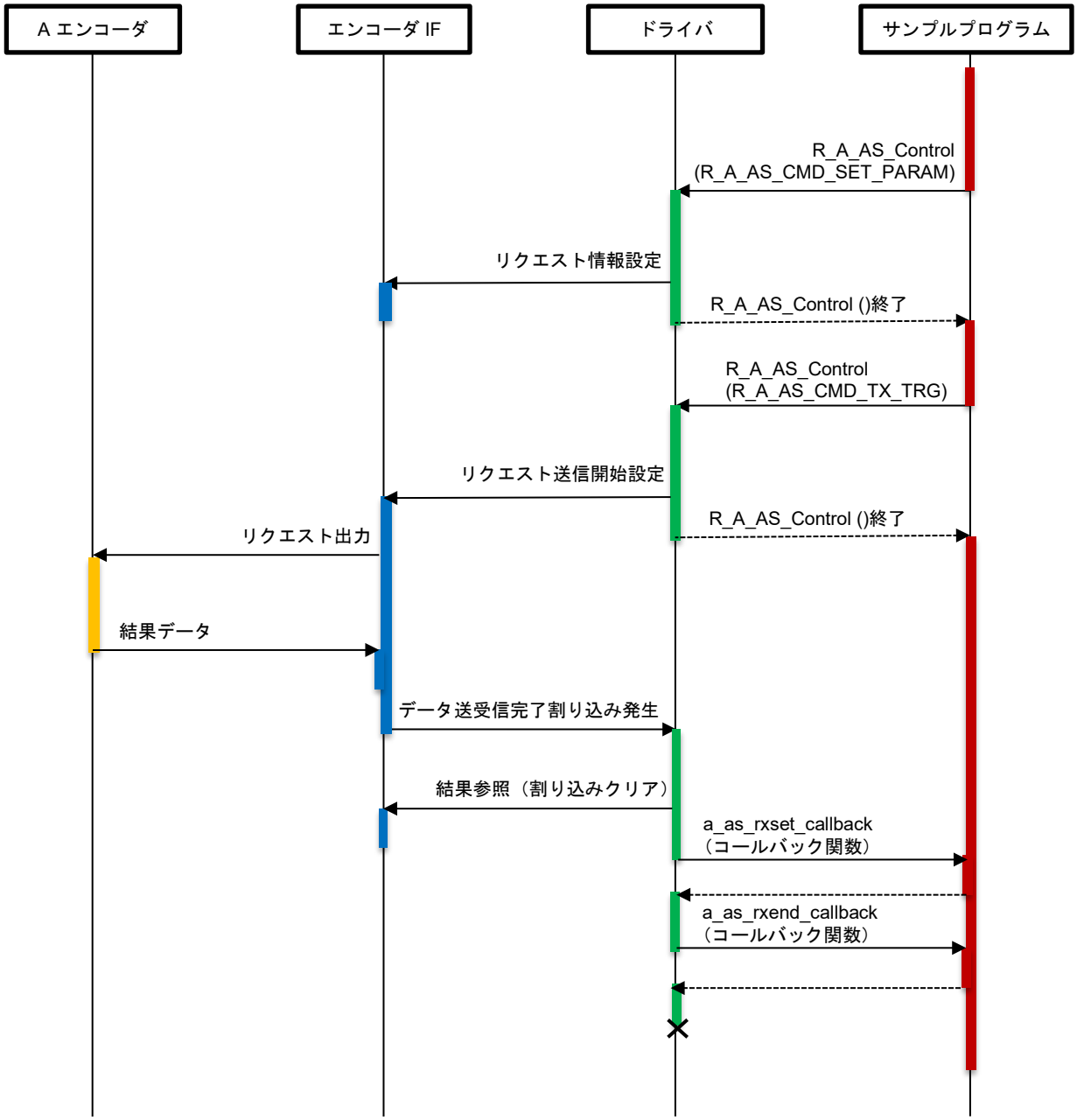


図 4-21 A-format リクエスト送信とデータ受信のシーケンス図

(3) EnDat リクエスト送信とデータ受信のシーケンス

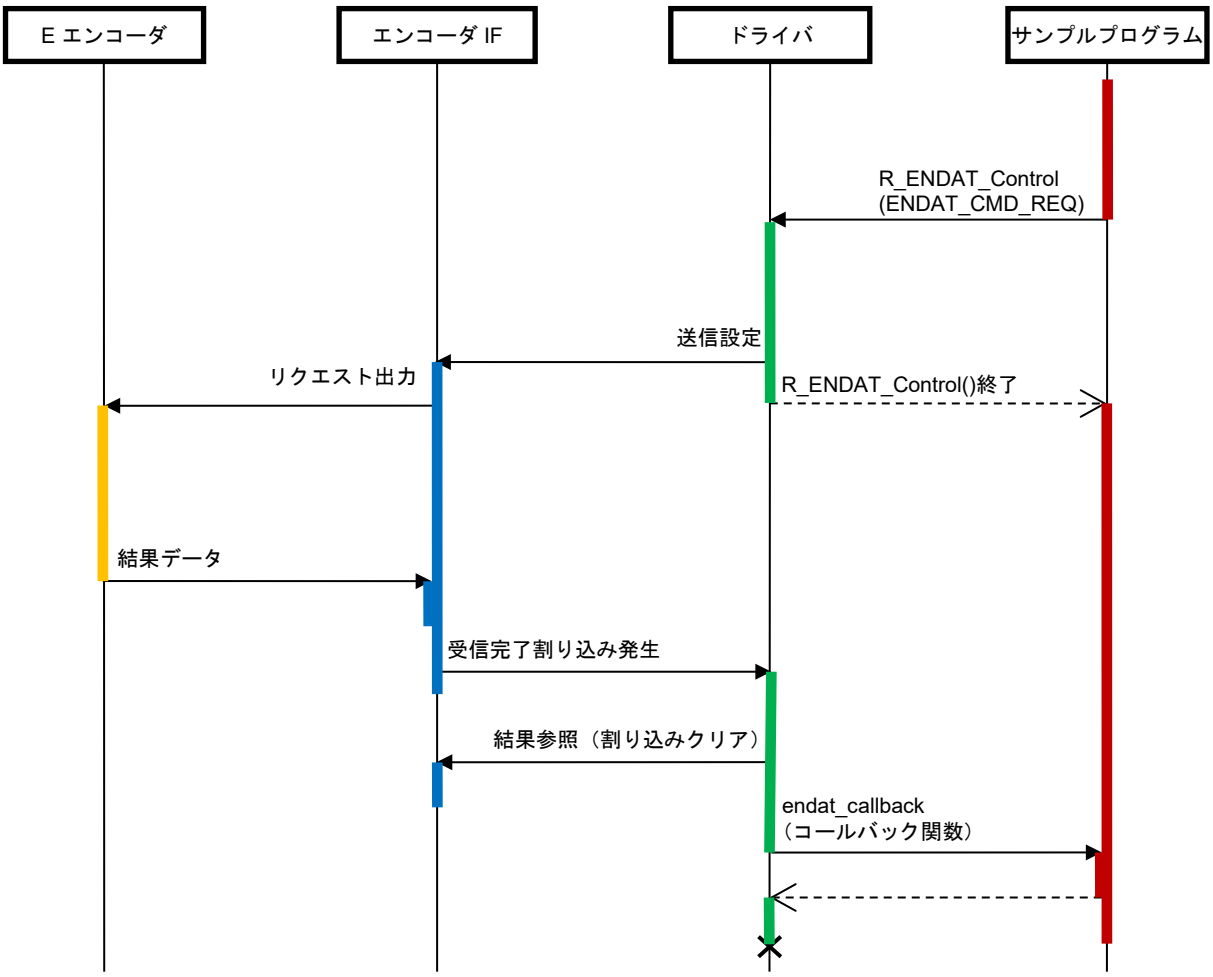


図 4-22 EnDat リクエスト送信とデータ受信のシーケンス図

(4) A-format ELC イベント入力トリガ動作のシーケンス

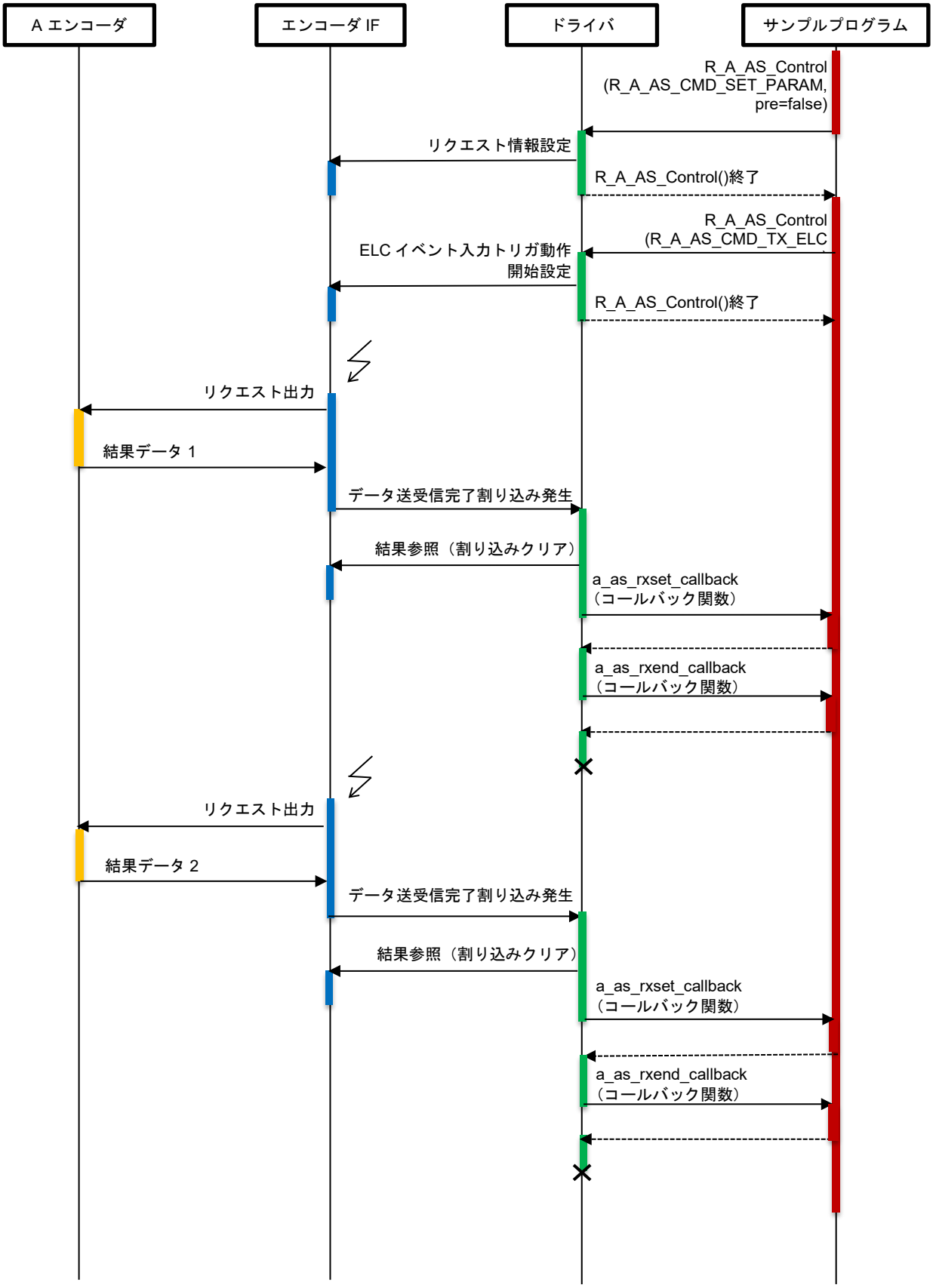


図 4-23 A-format ELC イベントトリガ動作のシーケンス図 (1/2)

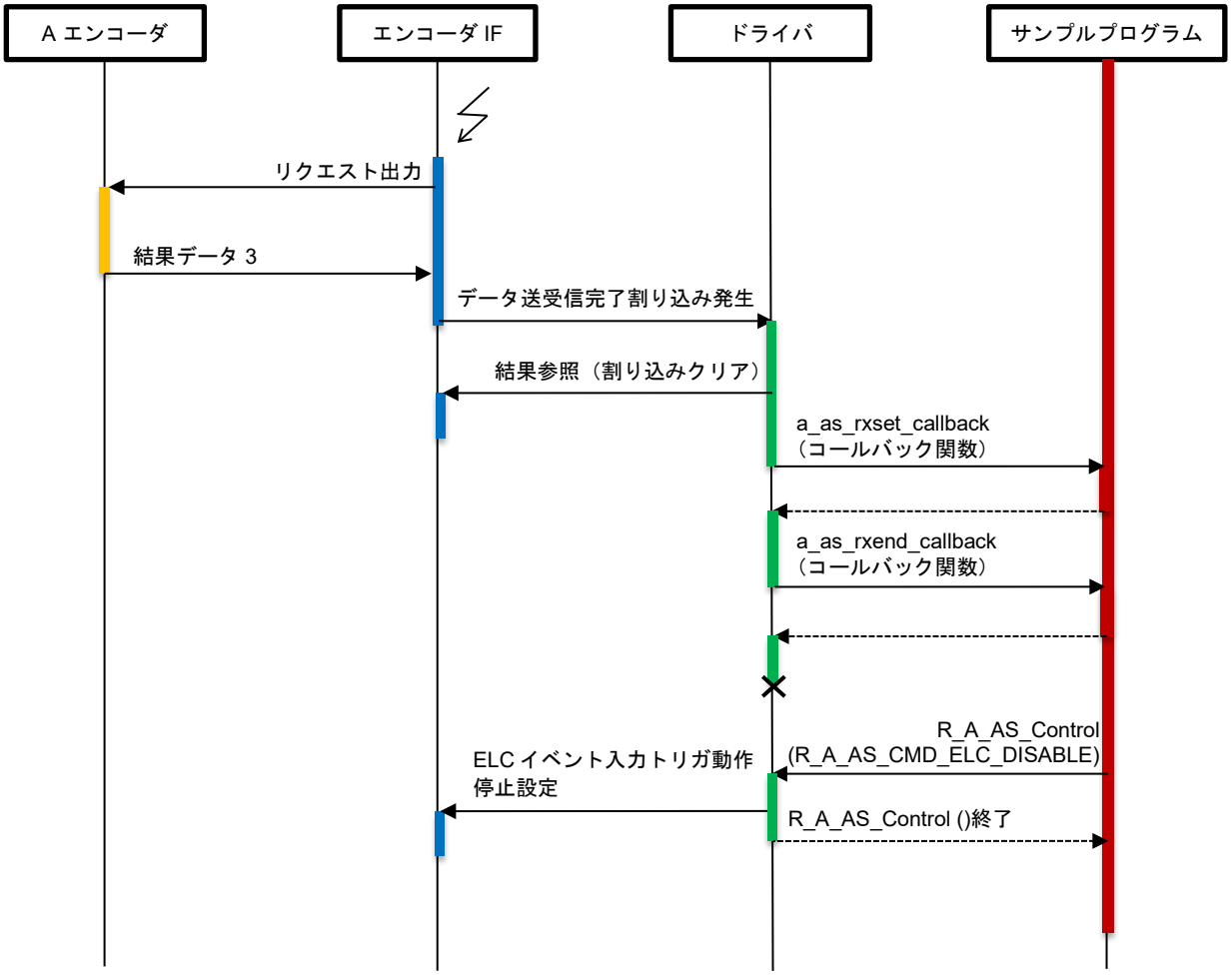


図 4-24　A-format ELC イベントトリガ動作のシーケンス図 (2/2)

(5) EnDat リクエスト送信(Continuous モード)とデータの連続受信シーケンス

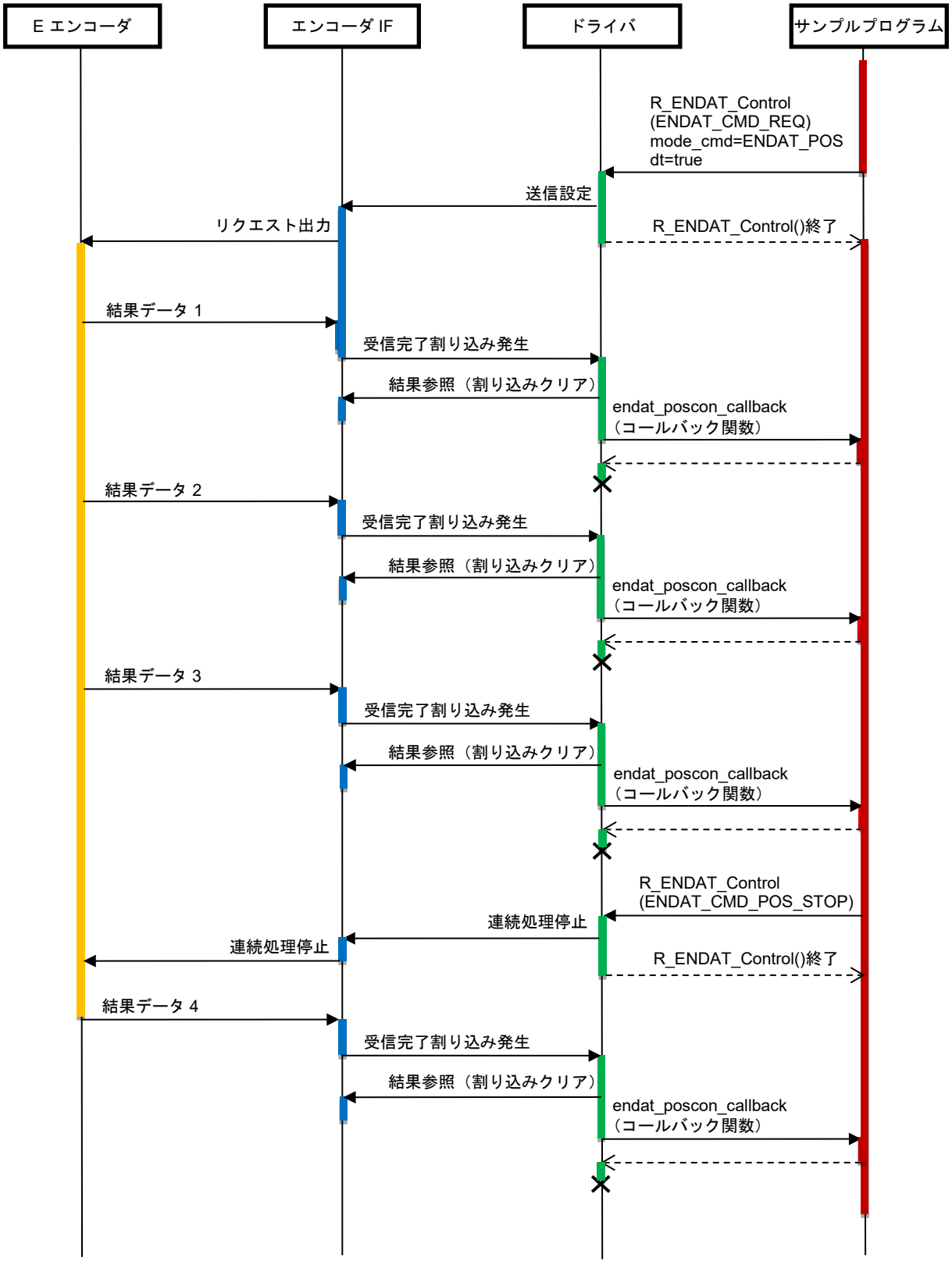


図 4-25 EnDat リクエスト送信(Continuous モード)とデータの連続受信シーケンス図

(6) EnDat リクエスト送信(ELC モード)とデータの連続受信シーケンス

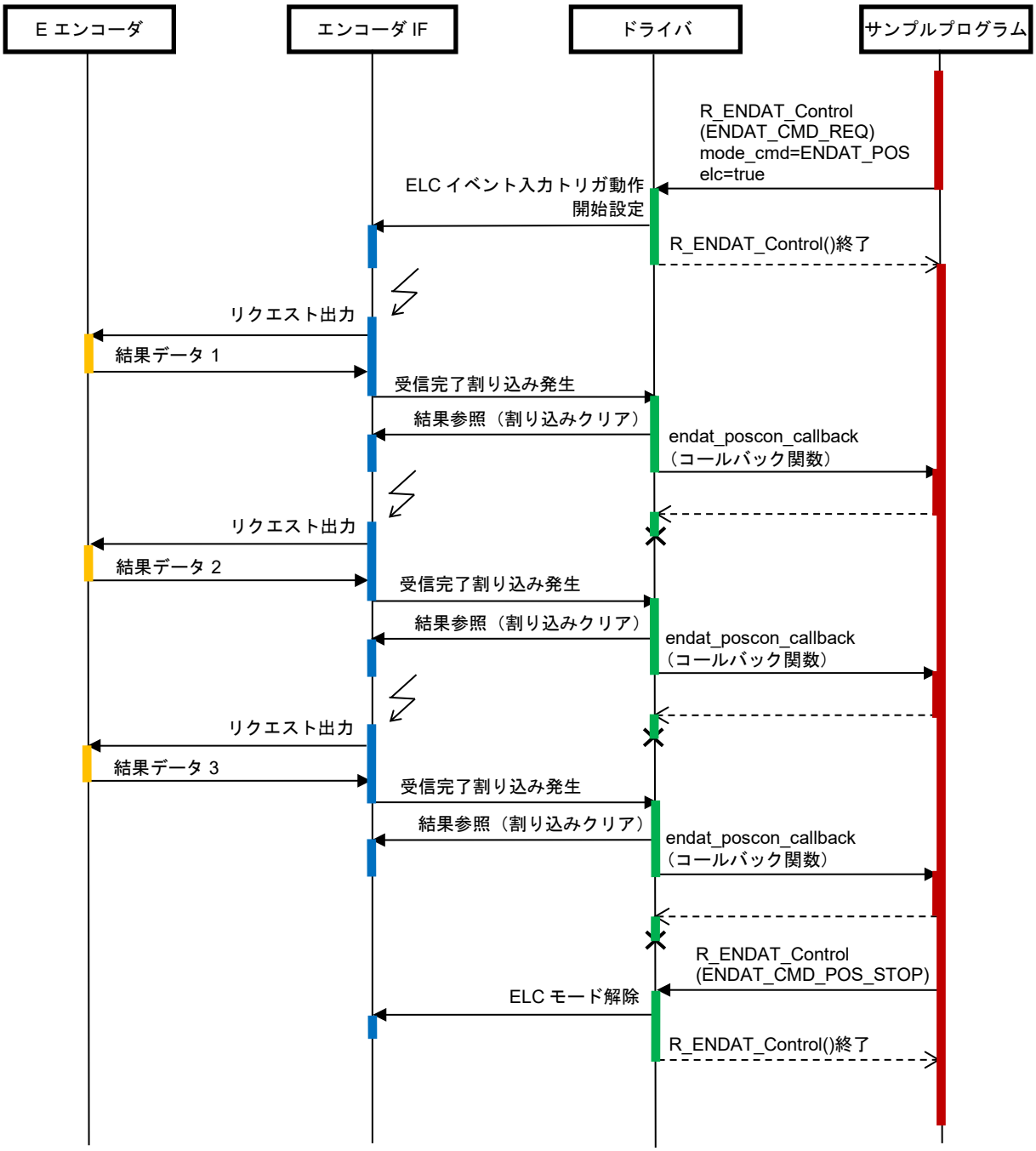


図 4-26 EnDat リクエスト送信(ELC モード)とデータの連続受信シーケンス図

(7) Dual Encoder 停止シーケンス

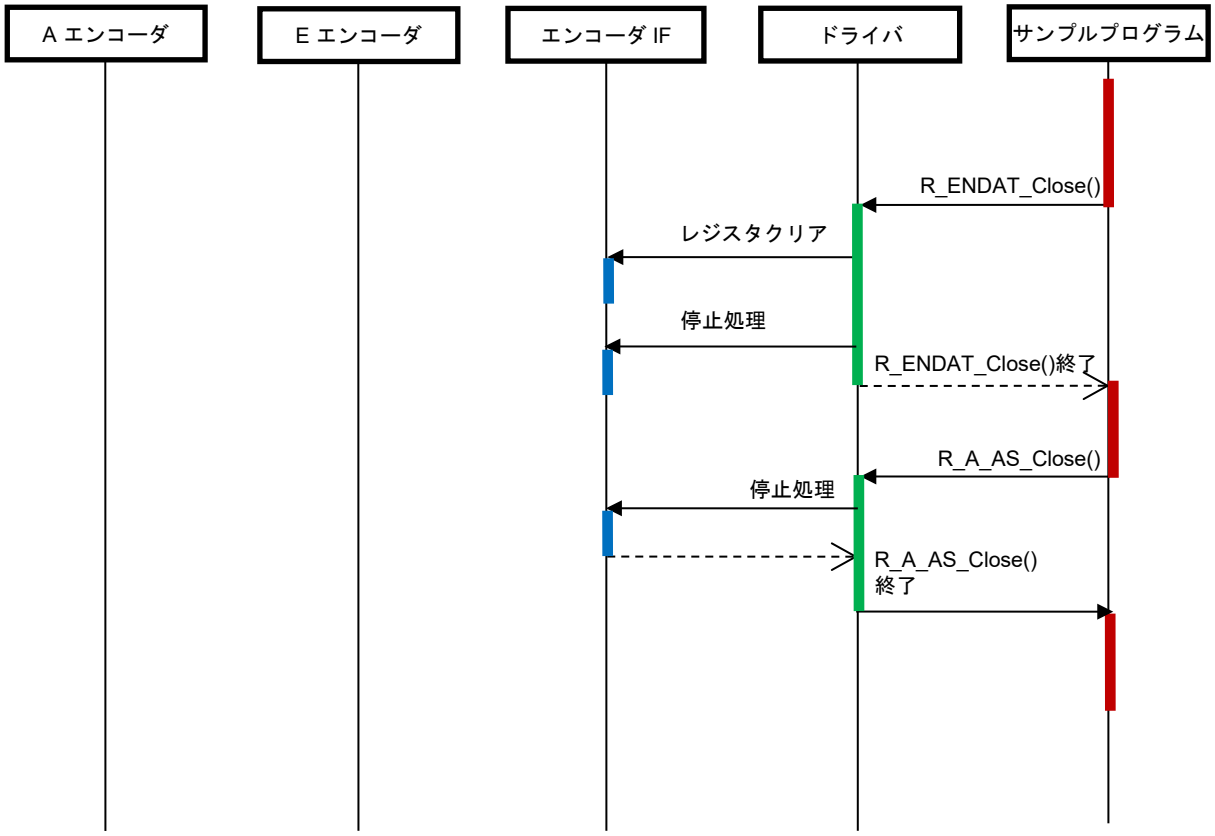


図 4-27 Dual Encoder 停止シーケンス図

4.15.10 コンソールコマンド

本サンプルプログラムは、A-format 仕様に準拠したエンコーダ(MAR-M50A)と、EnDat 2.2 仕様に準拠したエンコーダ(EQN1035)に対応しています。コンソールから入力可能なコマンドは以下の通りです。コマンドの先頭には識別コード "A", "E" を付けることで、どちらのエンコーダに対するコマンドかを区別しています。

表 4.29 A-format コンソールコマンド一覧

コマンド	内容
A req *ea cmd param1 param2*	エンコーダに対し *cmd* に入力したコマンドフレームを送信します。 *ea*：エンコーダ区分番号(10 進数) *cmd*：コマンドデータフレーム *param1*、*param2*：*cmd* によって入力値が異なります。詳細は「表 4.32 A-format req コマンドのパラメータ対応表」を参照してください。
A elctimer *ea val*	ELC イベント入力トリガ動作で、指定したエンコーダに対しコマンドデータフレーム CDF0 をビットレート 2.5Mbps で送信します。 *ea*：エンコーダ区分番号(10 進数) *val*：トリガ動作のタイミング （単位は us です。最大 6990us まで設定できます。） ELC イベント入力トリガ動作を停止させる場合はコンソールコマンド「elcstop」を実行してください。
A elcstop	ELC イベント入力トリガ動作を停止させます。

表 4.30 EnDat コンソールコマンド一覧

コマンド	内容
E pos	位置値を 1 回だけ取得します。
E poscon	位置値を連続して取得します。連続取得を停止する場合は「stop」コマンドを入力してください。
E elctimer *val*	ELC イベント入力トリガ動作として、位置値をタイマ周期で連続取得します。タイマ周期 *val* は us 単位(最大 6990us)で指定します。連続取得を停止する場合は「stop」コマンドを入力してください。
E stop	位置値の連続取得を停止します。
E temp	エンコーダから位置値とともに温度測定値を取得します。

表 4.31 共通コンソールコマンド一覧

コマンド	内容
exit	プログラムを終了します。

表 4.32　A-format req コマンドのパラメータ対応表

cmd	param1	param2
CDF0~CDF12	なし	なし
CDF13	EEPROM のアドレス(16 進数) 0x00~0xFF の範囲で指定しください。	なし
	使用例（エンコーダ区分番号 2 番の EEPROM の 0 番地のデータを読み込む） A req 2 CDF13 00	
CDF14	EEPROM のアドレス(16 進数) 0x00~0xEF の範囲の値を入力してください。	EEPROM へ書き込むデータ(16 進数) 0x0000~0xFFFF の範囲の値を入力してください。
	使用例（エンコーダ区分番号 2 番の EEPROM の 0 番地に 0x1234 を書き込む） A req 2 CDF14 00 1234	
CDF15~17	なし	なし
CDF18, CDF19	識別コード(16 進数) 0x00 0000~0xFF FFFF の範囲の値を入力してください。	なし
	使用例（エンコーダ区分番号 2 番の識別コードに 0x12 3456 を書き込む） A req 2 CDF18 123456	
CDF21, CDF27, CDF29	なし	なし

【注】 CDF11、CDF17、CDF19 は入力できますが、本サンプルプログラムの接続方式がバス接続のため使用できません。
複数伝送コマンド(CDF4～7, CDF22, CDF28, CDF30)および、CDF20 コマンドには対応していません。

(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。”Dual >”に続けてコマンドを入力してください。

```
Dual encoder sample program start
R_A_AS_GetVersion = 4.0


R_ENDAT_GetVersion = 4.0


Dual>
```

(2) コマンド実行例

A-format コンソールコマンドの、接続しているエンコーダ ENC1 に対して、コマンドフレーム CDF0 を送信するためのコンソール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、エンコーダアドレスやステータス情報とともに、角度データが表示されます。

```
Dual>A req 1 CDF0
A req command
A ------------------------------
A ENC1
A R_A_AS_REQ_SUCCESS
A EA : 0
A ES : 2
A CC : 0
A ABS 40bit [39:32] : 0x00000001
A ABS 40bit  [31:0] : 0x00B29A75
```

EnDat コンソールコマンドの、E 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、送受信結果、受信した位置値および付加情報が表示されます。

```
Dual>E pos
E pos command
E  result     : ENDAT_SUCCESS
E  pos_upper  : 0x00000000
E  pos_lower  : 0x00778D1B
E  add_datum1 : 0x00000000
E  add_datum2 : 0x00000000
```

4.15.11 チャネル変更手順

RZ/N2H には、エンコーダインタフェースとして、チャネル 0 から 15 のうちの、チャネル 8, チャネル 12, チャネル 15 を除く 13 チャネルが備えられています。チャネル 8, チャネル 12, チャネル 15 には、I/O ポートの割り当てがありません。Dual Encoder サンプルプログラムでは、初期状態で A-format がチャネル 0 に、EnDat がチャネル 1 に割り当てられています。チャネルの割り当てを変える場合には、ソースコード (dual_main.c), および各エンコーダドライバ (r_a_as_rzt2.c, r_endat_rzt2.c) 内のマクロ定義と、FSP Configurator 内の割り込みハンドラ, イベントリンク情報の指定を変更してください。

(1) 初期状態のチャネル設定。

初期状態のマクロ定義は、次の通りです。

```
dual_main.c :
  #define A_AS_CH (0)     /* A-format で使用するチャネルを設定 */
  #define ENDAT_CH (1)    /* EnDat で使用するチャネルを設定 */
r_a_as_rzt2.c :
  #define VECTOR_NUMBER_AFMTx_INT0 (VECTOR_NUMBER_ENCIF00_INT0)
                          /* A-format で、ENCIF00_INT0 ベクタを使用する */
r_endat_rzt2.c :
  #define VECTOR_NUMBER_ENDATx_INT0 (VECTOR_NUMBER_ENCIF01_INT0)
                          /* EnDat で、ENCIF01_INT0 ベクタを使用する */
```

端子や機能は初期状態で、表 4.33 のように割り当てられています。

表 4.33 使用端子と機能(初期状態)

チャネル	端子名	I/O ポート	入出力	I/O 電源ドメイン	内容
AFMT0	ENCIFOE00 (D/R)	P14_3	出力	VDD33	データ出力許可設定
	ENCIFDO00 (REQ)	P14_4	出力	VDD33	データ出力
	ENCIFDI00 (SDAT)	P14_5	入力	VDD33	データ入力
ENDAT_CH1	ENCIFCK01 (TCLK1)	P02_0	出力	VDD1833_5	クロック出力
	ENCIFOE01 (DE1)	P02_1	出力	VDD1833_5	データ出力許可設定
	ENCIFDO01 (DATA_DV1)	P26_7	出力	VDD33	データ出力
	ENCIFDI01 (DATA_RC1)	P27_0	入力	VDD33	データ入力

(2) チャネル変更例 1 の設定

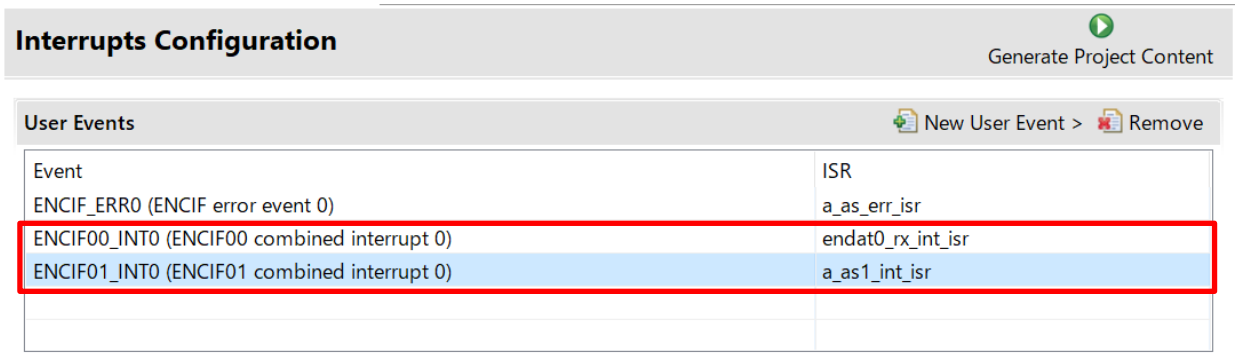
A-format をチャネル 1 に、EnDat をチャネル 0 に割り当てを変更する場合には、各ファイルのマクロ定義を次のように変更してください。

```
dual_main.c :
  #define A_AS_CH (1)     /* A-format で使用するチャネルを 1 に変更 */
  #define ENDAT_CH (0)    /* EnDat で使用するチャネルを 0 に変更 */
r_a_as_rzt2.c :
  #define VECTOR_NUMBER_AFMTx_INT0 (VECTOR_NUMBER_ENCIF01_INT0)
                          /* A-format で、ENCIF01_INT0 ベクタを使用する */
r_endat_rzt2.c :
  #define VECTOR_NUMBER_ENDATx_INT0 (VECTOR_NUMBER_ENCIF00_INT0)
                          /* EnDat で、ENCIF00_INT0 ベクタを使用する */
```

更に、FSP Configurator から、ENCIF00_INT0 イベント、ENCIF01_INT0 イベントに対応する割り込みハンドラ(ISR)を変更します。

表 4.34　割り込みハンドラ(ISR)の変更点

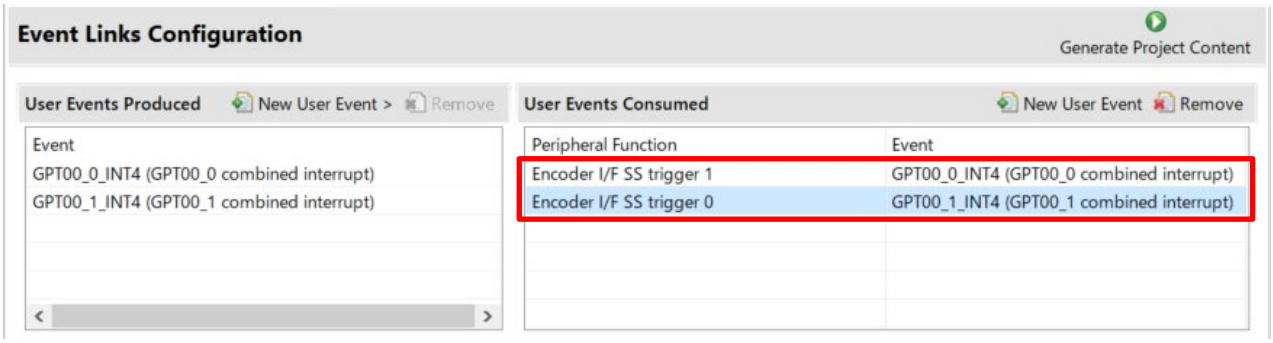
Event	初期状態の ISR	チャネル変更後の ISR
ENCIF00_INT0 (ENCIF00 combined Interrupt 0)	a_as0_int_isr	endat0_rx_int_isr
ENCIF01_INT0 (ENCIF01 combined Interrupt 0)	endat1_rx_int_isr	a_as1_int_isr



A-format および EnDat には、GPT が出力する周期的なイベントを ELC イベントとして受け取って動作する動作モードがあります。ELC イベントの割り当ても、FSP Configurator の Event Links タブで行います。A-format は GPT00_0_INT4、EnDat は GPT00_1_INT4 にそれぞれ対応しています。

表 4.35　Event Links の変更点

Event	初期状態の Peripheral Function	チャネル変更後の Peripheral Function
GPT00_0_INT4 (GPT00_0 combined interrupt)	Encoder I/F SS trigger 0	Encoder I/F SS trigger 1
GPT00_1_INT4 (GPT00_1 combined interrupt)	Encoder I/F SS trigger 1	Encoder I/F SS trigger 0



変更後の端子や機能は、表 4.36 のように割り当てられます。

表 4.36　使用端子と機能(チャネル変更例 1)

チャネル	端子名	I/O ポート	入出力	I/O 電源ドメイン	内容
AFMT1	ENCIFOE01 (D/R)	P02_1	出力	VDD1833_5	データ出力許可設定
	ENCIFDO01 (REQ)	P26_7	出力	VDD33	データ出力
	ENCIFDI01 (SDAT)	P27_0	入力	VDD33	データ入力
ENDAT_CH0	ENCIFCK00 (TCLK0)	P14_2	出力	VDD33	クロック出力
	ENCIFOE00 (DE0)	P14_3	出力	VDD33	データ出力許可設定
	ENCIFDO00 (DATA_DV0)	P14_4	出力	VDD33	データ出力
	ENCIFDI00 (DATA_RC0)	P14_5	入力	VDD33	データ入力

(3)　チャネル変更例 2 の設定

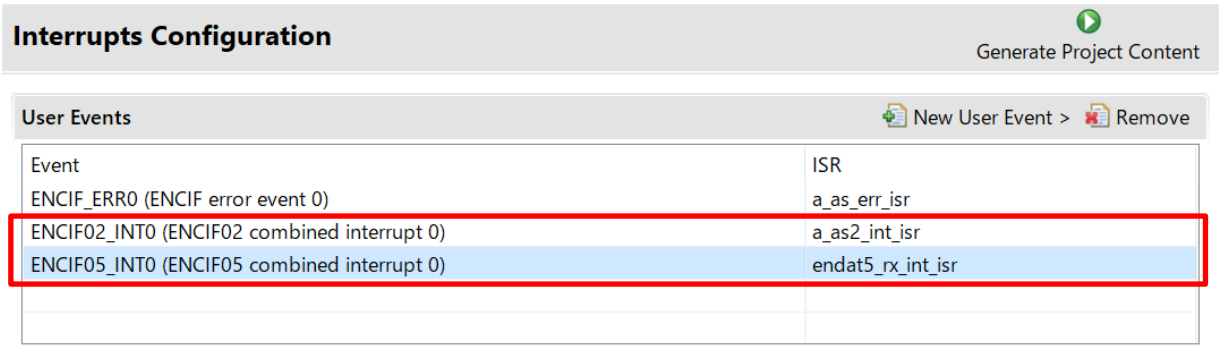
A-format をチャネル 2 に、EnDat をチャネル 5 に割り当てを変更する場合には、各ファイルのマクロ定義を次のように変更してください。

```
dual_main.c :
  #define A_AS_CH (2)     /* A-format で使用するチャネルを 2 に変更 */
  #define ENDAT_CH (5)    /* EnDat で使用するチャネルを 5 に変更 */
r_a_as_rzt2.c :
  #define VECTOR_NUMBER_AFMTx_INT0 (VECTOR_NUMBER_ENCIF02_INT0)
                          /* A-format で、ENCIF02_INT0 ベクタを使用する */
r_endat_rzt2.c :
  #define VECTOR_NUMBER_ENDATx_INT0 (VECTOR_NUMBER_ENCIF05_INT0)
                          /* EnDat で、ENCIF05_INT0 ベクタを使用する */
```

更に、FSP Configurator から、ENCIF00_INT0 イベント、ENCIF01_INT0 イベントに対応する割り込みハンドラ(ISR)を削除して、ENCIF02_INT0 イベント、ENCIF05_INT0 イベントに対応する割り込みハンドラ(ISR)を新しく登録します。

表 4.37　割り込みハンドラ(ISR)の変更点

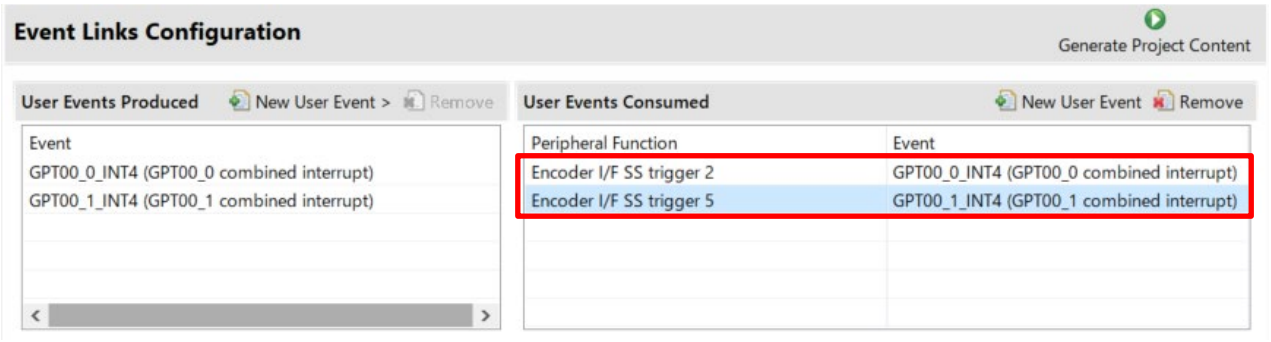
Event	初期状態の ISR	チャネル変更後の ISR
ENCIF00_INT0 (ENCIF00 combined Interrupt 0)	a_as0_int_isr	(削除)
ENCIF01_INT0 (ENCIF01 combined Interrupt 0)	endat1_rx_int_isr	(削除)
ENCIF02_INT0 (ENCIF02 combined Interrupt 0)	-	a_as2_int_isr
ENCIF05_INT0 (ENCIF05 combined Interrupt 0)	-	endat5_rx_int_isr



ELC イベントの割り当ての変更を、FSP Configurator の Event Links タブで行います。

表 4.38　Event Links の変更点

Event	初期状態の Peripheral Function	チャネル変更後の Peripheral Function
GPT00_0_INT4 (GPT00_0 combined interrupt)	Encoder I/F SS trigger 0	Encoder I/F SS trigger 2
GPT00_1_INT4 (GPT00_1 combined interrupt)	Encoder I/F SS trigger 1	Encoder I/F SS trigger 5



変更後の端子や機能は、表 4.39 のように割り当てられます。本サンプルプログラムでは、エンコーダ I/F の 13 チャネルに対応する I/O ポートを選択済みです。FSP Configurator の Pins タブで、I/O ポートの設定を変更する必要はありません。

表 4.39　使用端子と機能(チャネル入れ替え例 2) 注

チャネル	端子名	I/O ポート	入出力	I/O 電源ドメイン	内容
AFMT2	ENCIFOE02 (D/R)	P03_4	出力	VDD33	データ出力許可設定
	ENCIFDO02 (REQ)	P03_1	出力	VDD33	データ出力
	ENCIFDI02 (SDAT)	P03_2	入力	VDD33	データ入力
ENDAT_CH5	ENCIFCK05 (TCLK5)	P12_4	出力	VDD1833_6	クロック出力
	ENCIFOE05 (DE5)	P12_5	出力	VDD1833_6	データ出力許可設定
	ENCIFDO05 (DATA_DV5)	P12_6	出力	VDD1833_6	データ出力
	ENCIFDI05 (DATA_RC5)	P12_7	入力	VDD1833_6	データ入力

【注】　その他のチャネルを割り当てる場合の使用端子や機能については、EnDat では RZ/N2H グループ Encoder I/F EnDat sample program アプリケーションノートを、A-format では RZ/N2H グループ Encoder I/F A-format sample program アプリケーションノートを参照してください。

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
2.00	Mar 28.25	-	初版発行
3.00	Nov 28.25	42 48	構造体 r_a_as_status_t の説明を簡潔に変更 列挙型 r_endat_cmd_t の誤記を修正
4.00	May 15.26	5 10 - 28, 43, 45, 54 - 59 15, 16, 25 - 28	Cortex-A55 Core0 の周波数を 1200MHz に変更 ポインタ変数のプレフィクスを”p_”に変更 「ユーザー定義関数仕様」のヘッダ部分を修正

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。

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。