RZ/T2M グループ

EnDat 3 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、EnDat 3 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。

- EnDat 3 のフォアグラウンド通信、バックグラウンド通信に対応
- EnDat 3 仕様に準拠したエンコーダ(HEIDENHAIN 社製 ECI 1319 E30-R2)や、EnDat 3 仕様のバス接続に対応したエンコーダ(HEIDENHAIN 社製 EQN 1337 E30-RB)から、角度情報等を取得
- EnDat 3 仕様に準拠したエンコーダのメモリアクセスに対応

動作確認デバイス

RZ/T2M

本アプリケーションノートを他のマイコンへ適用する場合、そのマイコンの仕様にあわせて変更し、十分評価してください。

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、図 1-2 にバス接続の例を示します。

表 1.1　使用する周辺機能と用途

周辺機能	用途
EnDat 3 I/F	EnDat 3 仕様に準拠したエンコーダとの通信
割り込みコントローラ(ICU)	EnDat 3 I/F 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 EnDat 3 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

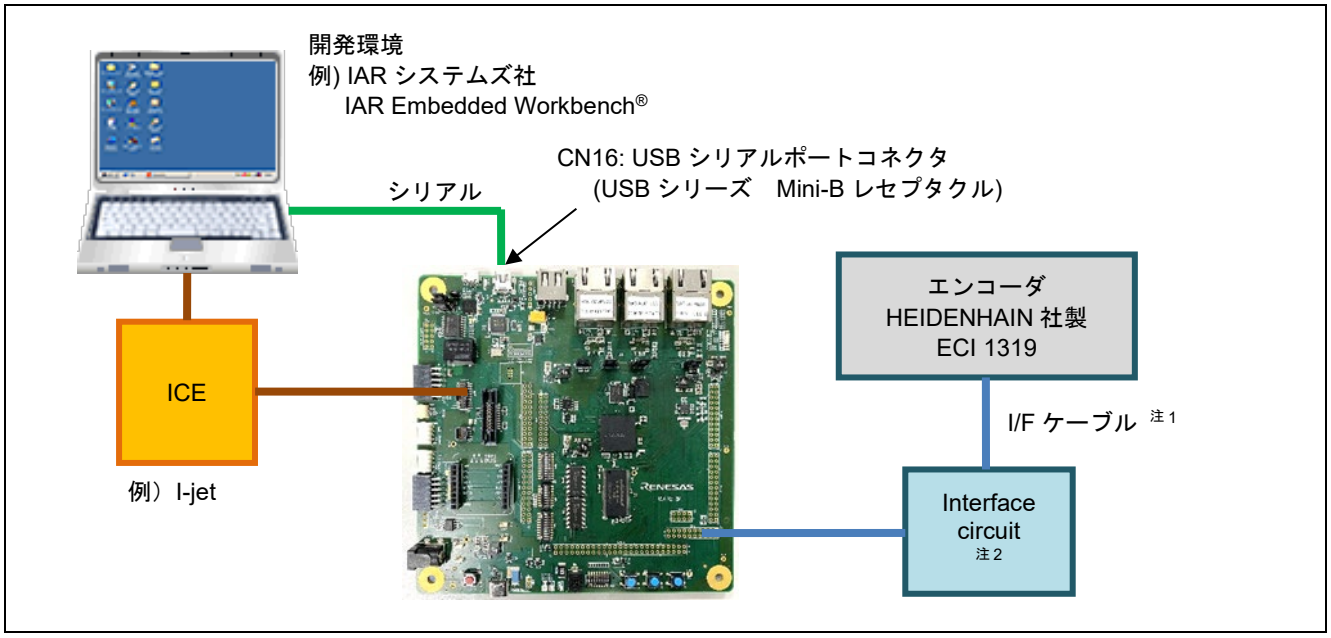

図 1-1　動作環境

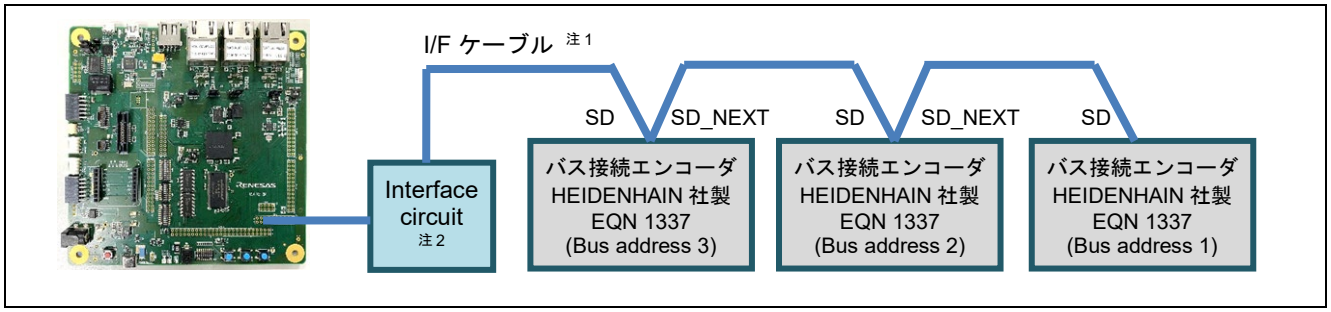

図 1-2　バス接続例

【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. Interface circuit については、HEIDENHAIN 社に問い合わせることで入手可能な「EnDat 3 Hardware Specification」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	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】 統合開発環境のバージョンは、RZ/T2M グループ Encoder I/F EnDat 3 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

チャネル	端子名 (機能ピン名)	I/O ポート	入出力	内容
ENDAT3_CH0	soen0 (ENCIF4)	P02_3	出力	データ出力許可設定端子
	sdout0 (ENCIF3)	P02_2	出力	データ出力端子
	sdin00 (ENCIF0)	P01_6	入力	データ入力端子 0 注
	sdin10 (ENCIF1)	P01_7	入力	データ入力端子 1 注
ENDAT3_CH1	soen1 (ENCIF9)	P03_3	出力	データ出力許可設定端子
	sdout1 (ENCIF8)	P03_0	出力	データ出力端子
	sdin01 (ENCIF5)	P17_3	入力	データ入力端子 0 注
	sdin11 (ENCIF6)	P17_4	入力	データ入力端子 1 注

【注】 データ入力端子 0 とデータ入力端子 1 には、同一の受信信号を入力して使用してください。

4. ソフトウェア説明

4.1 EnDat 3 ドライバ機能

EnDat 3 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. メモリ領域アクセス
4. 付加情報の受信 注

【注】 本書では様々な Frame ID (FIDs) が割り当てられている Low Priority Frame (LPF) のコンテンツを「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 3 Interface Specification」を参照してください。

4.2 ファイル構成

ファイル構成は、RZ/T2M グループ Encoder I/F EnDat 3 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
EnDat 3 ドライバ API 関数	R_ENDAT3_Open	8
	R_ENDAT3_Close	8
	R_ENDAT3_GetVersion	8
	R_ENDAT3_Control	9
ユーザー定義関数	endat3_init_reset_wait_callback	13
	endat3_callback	13
	endat3_pos_callback	14
	endat3_ready_callback	14
割り込みハンドラ	enc_ch0_fg_isr	15
	enc_ch1_fg_isr	15
	enc_ch0_bg_isr	15
	enc_ch1_bg_isr	15
	enc_ch0_fifo_isr	16
	enc_ch1_fifo_isr	16

4.4　API 関数仕様

4.4.1　R_ENDAT3_Open

R_ENDAT3_Open		
概　要	EnDat 3 エンコーダ制御の開始	
ヘッダ	r_endat3_rzt2_if.h	
宣　言	r_endat3_err_t R_ENDAT3_Open(const int32_t id, r_endat3_info_t* p_info);	
説　明	EnDat 3 ドライバは下記の初期設定を行います。 1.　エンコーダ I/F の初期化 2.　HELLO リクエストの発行 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、この HELLO リクエスト送信後には、300 ミリ秒以上の待ち時間を挿入してください。HELLO リクエストに対するレスポンスデータは取得しません。	
引　数	id	：使用する ID を指定します。(r_endat3_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
	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_endat3_info_t のアドレスを指定してください。
リターン値	ENDAT3_SUCCESS	：正常終了
	ENDAT3_ERR_INVALID_ARG	：異常終了 (id, p_info に指定した e_endat3_info_t 構造体のメンバ変数が規定されていない値です)
	ENDAT3_ERR_ACCESS	：異常終了 (既に open されています)
補足	本関数実行前に、必ず EC-Lib を用いて Multi-Protocol Encoder IF のコンフィグレーションと起動を行ってください。	

4.4.2　R_ENDAT3_Close

R_ENDAT3_Close		
概　要	EnDat 3 エンコーダの制御を終了	
ヘッダ	r_endat3_rz2_if.h	
宣　言	r_endat3_err_t R_ENDAT3_Close(const int32_t id);	
説　明	指定されたチャンネルの EnDat 3 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_endat3_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
	上記以外	：設定不可
リターン値	ENDAT3_SUCCESS	：正常終了
	ENDAT3_ERR_INVALID_ARG	：異常終了 (指定した id が規定されていない値です)
	ENDAT3_ERR_ACCESS	：異常終了 (リクエストを送信中です)

4.4.3 R_ENDAT3_GetVersion

R_ENDAT3_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_endat3_rz2_if.h	
宣　言	uint32_t R_ENDAT3_GetVersion(void);	
説　明	EnDat 3 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

4.4.4 R_ENDAT3_Control

R_ENDAT3_Control		
概　要	EnDat 3 エンコーダの制御	
ヘッダ	r_endat3_rz2_if.h	
宣　言	r_endat3_err_t R_ENDAT3_Control(const int32_t id, const r_endat3_cmd_t cmd, r_endat3_req_t *p_req);	
説　明	引数 cmd を使って EnDat 3 エンコーダを制御します。制御コマンドの動作は「4.4.5 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_endat3_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
	上記以外	：設定不可
	cmd	：コマンド 内容は「4.10.3(2) r_endat3_cmd_t」参照。
	p_req	：各 cmd に対応する引数
リターン値	ENDAT3_SUCCESS	：正常終了
	ENDAT3_ERR_INVALID_ARG	：異常終了 (id, cmd が規定されていない値です)
	その他のリターン値は「4.4.5 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_ENDAT3_Open を実行してください。	

4.4.5 制御コマンド

(1) ENDAT3_CMD_REQ

ENDAT3_CMD_REQ	
概　要	EnDat 3 エンコーダへリクエストを送信
ヘッダ	r_endat3_rz2_if.h
宣　言	r_endat3_err_t R_ENDAT3_Control(const int32_t id, const r_endat3_cmd_t cmd, r_endat3_req_t *p_req);
説　明	EnDat 3 エンコーダへフォアグラウンドリクエストを送信します。 リクエスト送信 1 回に対して endat3_callback 関数と endat3_ready_callback 関数が 1 回ずつコールされます。 ELC モードを有効にした場合、ENDAT3_CMD_STOP を実行するまで、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、endat3_pos_callback 関数と endat3_ready_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_endat3_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 上記以外 ：設定不可 cmd ：ENDAT3_CMD_REQ p_req ：リクエスト情報 リクエスト情報を格納した r_endat3_req_t 構造体のポインタを指定します。r_endat3_req_t 構造体の詳細は「4.10.1(2) r_endat3_req_t」参照。
リターン値	ENDAT3_SUCCESS ：正常終了 ENDAT3_ERR_INVALID_ARG ：異常終了 (id, cmd が規定されていない値、p_req が NULL、p_req に指定された r_endat3_req_t 構造体のメンバ変数が規定されていない値です) ENDAT3_ERR_BUSY ：異常終了 (送信処理中、または FG_STATUS レジスタ MASTER_READY ビットが 0 のため操作できません) ENDAT3_ERR_ACCESS ：異常終了 (該当チャネルが開始されていません)

(2) ENDAT3_CMD_BGREQ

ENDAT3_CMD_BGREQ	
概　要	EnDat 3 エンコーダへバックグラウンドリクエストを送信
ヘッダ	r_endat3_rz2_if.h
宣　言	r_endat3_err_t R_ENDAT3_Control(const int32_t id, const r_endat3_cmd_t cmd, r_endat3_req_t *p_req);
説　明	EnDat 3 エンコーダへバックグラウンドリクエストを送信します。 バックグラウンドリクエストは、3 回のフォアグラウンドリクエストに分割して送信されます。更に、バックグラウンドレスポンスが更新されるまで、フォアグラウンドリクエストを送信し続けます。フォアグラウンドリクエスト送信のたびに、endat3_callback 関数がコールされます。 バックグラウンドレスポンスを受信すると、endat3_callback 関数と endat3_ready_callback 関数をコールして終了します。
引　数	id　：使用する ID を指定します。(r_endat3_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 上記以外　：設定不可 cmd　：ENDAT3_CMD_BGREQ p_req　：リクエスト情報 リクエスト情報を格納した r_endat3_req_t 構造体のポインタを指定します。r_endat3_req_t 構造体の詳細は「4.10.1(2) r_endat3_req_t」参照。
リターン値	ENDAT3_SUCCESS　：正常終了 ENDAT3_ERR_INVALID_ARG　：異常終了 (id, cmd が規定されていない値、p_req が NULL、p_req に指定された r_endat3_req_t 構造体のメンバ変数が規定されていない値です) ENDAT3_ERR_BUSY　：異常終了 (送信処理中、または FG_STATUS レジスタ MASTER_READY ビットが 0 のため操作できません) ENDAT3_ERR_ACCESS　：異常終了 (該当チャネルが開始されていません)

(3) ENDAT3_CMD_RATE

ENDAT3_CMD_RATE		
概　要	エンコーダ I/F のデータレートを変更	
ヘッダ	r_endat3_rz2_if.h	
宣　言	r_endat3_err_t R_ENDAT3_Control(const int32_t id, const r_endat3_cmd_t cmd, r_endat3_req_t * p_req);	
説　明	EnDat 3 エンコーダ I/F のデータレートを、リクエスト情報で指定した値に切り替えます。 エンコーダ本体のデータレートは、事前にフォアグラウンドリクエストを使って切り替えてください。	
引　数	id	：使用する ID を指定します。(r_endat3_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
		上記以外 ：設定不可
	cmd	：ENDAT3_CMD_RATE
	p_req	：リクエスト情報 リクエスト情報を格納した r_endat3_req_t 構造体のポインタを指定します。構造体のメンバのうち、req_data のみが参照されます。 req_data が 0 のとき、12.5 Mbps に、 req_data が 1 のとき、25 Mbps に切り替えます、
リターン値	ENDAT3_SUCCESS	：正常終了
	ENDAT3_ERR_INVALID_ARG	：異常終了 (id, cmd が規定されていない値です)
	ENDAT3_ERR_BUSY	：異常終了 (送信処理中)
	ENDAT3_ERR_ACCESS	：異常終了 (該当チャネルが開始されていません)

(4) ENDAT3_CMD_STOP

ENDAT3_CMD_STOP		
概　要	位置値連続取得の停止	
ヘッダ	r_endat3_rz2_if.h	
宣　言	r_endat3_err_t R_ENDAT3_Control(const int32_t id, const r_endat3_cmd_t cmd, r_endat3_req_t * p_req);	
説　明	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 3 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_endat3_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
		上記以外 ：設定不可
	cmd	：ENDAT3_CMD_STOP
	p_req	：使用しません (NULL を指定してください)
リターン値	ENDAT3_SUCCESS	：正常終了
	ENDAT3_ERR_INVALID_ARG	：異常終了 (id, cmd が規定されていない値です)
	ENDAT3_ERR_ACCESS	：異常終了 (ELC モードが有効なリクエストが送信されていません)

4.5 ユーザー定義関数仕様

4.5.1 endat3_init_reset_wait_callback

endat3_init_reset_wait_callback	
概　要	初回の HELLO リクエスト送信後の待機時間生成関数
ヘッダ	-
宣　言	void endat3_init_reset_wait_callback(void);
説　明	R_ENDAT3_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、HELLO リクエスト送信後に待機する時間を生成します。300 ミリ秒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

4.5.2 endat3_callback

endat3_callback		
概　要	リクエストに対する受信結果通知関数	
ヘッダ	-	
宣　言	void endat3_callback(uint16_t num_result, r_endat3_result_t * p_result, r_endat3_protocol_err_t *p_err, uint16_t runtime);	
説　明	R_ENDAT3_Control 関数で ENDAT3_CMD_REQ コマンドや ENDAT3_CMD_BGREQ コマンドを引数として実行する時に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。フォアグラウンド割り込み発生時にコールされます。 ENDAT3_CMD_BGREQ コマンドを引数としてバックグラウンドリクエストを実行中には、コールバックが複数回発生します。バックグラウンドレスポンスは、最後のコールバックの送受信結果として通知されます。 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	num_result	: 送受信結果総数
	p_result	: 送受信結果 送受信結果を格納した構造体 r_endat3_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 送受信結果を格納した構造体 r_endat3_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。
	runtime	: 伝搬時間表示 リクエストを送信してから、レスポンスを受信するまでの伝搬時間が通知されます。
リターン値	なし	

4.5.3 endat3_pos_callback

endat3_pos_callback	
概　要	ELC モードのフォアグラウンドリクエストに対する受信結果通知関数
ヘッダ	-
宣　言	void endat3_pos_callback(uint16_t num_result, r_endat3_result_t * p_result, endat3_protocol_err_t *p_err, uint16_t runtime);
説　明	ELC モードでデータ送信を行う時に、R_ENDAT3_Control (ENDAT3_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。フォアグラウンド割り込み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	num_result : 送受信結果総数 p_result : 送受信結果 送受信結果を格納した構造体 r_endat3_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 送受信結果を格納した構造体 r_endat3_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。 runtime : 伝搬時間表示 リクエストを送信してから、レスポンスを受信するまでの伝搬時間が通知されます。
リターン値	なし

4.5.4 endat3_ready_callback

endat3_ready_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 endat3_ready_callback(void);
説　明	R_ENDAT3_Control 関数で ENDAT3_CMD_REQ コマンドや ENDAT3_CMD_BGREQ コマンドを実行する時に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。 ENDAT3_CMD_REQ コマンドでは、フォアグラウンド割り込み発生時に endat3_callback 関数の後にコールされます。ELC モードで動作中は、フォアグラウンド割り込みが発生するたびに、endat3_pos_callback 関数の後にコールされます。 ENDAT3_CMD_BGREQ コマンドでは、バックグラウンドレスポンス受信後のフォアグラウンド割り込み発生時のみで、endat3_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

4.6 割り込みハンドラ

4.6.1 enc_ch0_fg_isr

enc_ch0_fg_isr	
概　要	チャネル 0 フォアグラウンド割り込みハンドラ
ヘッダ	-
宣　言	void enc_ch0_fg_isr(void);
説　明	EnDat 3 チャネル 0 のフォアグラウンド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

4.6.2 enc_ch1_fg_isr

enc_ch1_fg_isr	
概　要	チャネル 1 フォアグラウンド割り込みハンドラ
ヘッダ	-
宣　言	void enc_ch1_fg_isr(void);
説　明	EnDat 3 チャネル 1 のフォアグラウンド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

4.6.3 enc_ch0_bg_isr

enc_ch0_bg_isr	
概　要	チャネル 0 バックグラウンド割り込みハンドラ
ヘッダ	-
宣　言	void enc_ch0_bg_isr(void);
説　明	EnDat 3 チャネル 0 のバックグラウンド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

4.6.4 enc_ch1_bg_isr

enc_ch1_bg_isr	
概　要	チャネル 1 バックグラウンド割り込みハンドラ
ヘッダ	-
宣　言	void enc_ch1_bg_isr(void);
説　明	EnDat 3 チャネル 1 のバックグラウンド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

4.6.5 enc_ch0_fifo_isr

enc_ch0_fifo_isr	
概　要	チャネル 0 データ受信割り込みハンドラ
ヘッダ	-
宣　言	void enc_ch0_fifo_isr(void);
説　明	EnDat 3 チャネル 0 のデータ受信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

4.6.6 enc_ch1_fifo_isr

enc_ch1_fifo_isr	
概　要	チャネル 0 データ受信割り込みハンドラ
ヘッダ	-
宣　言	void enc_ch1_fifo_isr(void);
説　明	EnDat 3 チャネル 1 のデータ受信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 使用割り込み一覧

表 4.2 に EnDat 3 ドライバで使用する割り込みを示します。

表 4.2　EnDat 3 ドライバで使用する割り込み

割り込み	ID	概要
ENCIF_INT0	372	フォアグラウンド通信の終了やエラー発生によって、チャネル 0 が通信待機状態になると割り込みが発生します。
ENCIF_INT1	373	チャネル 0 が通信待機状態になったときのバックグラウンド通信が、以下の場合に割り込みが発生します。 1. バックグラウンドレスポンスを受信した 2. エンコーダがバックグラウンドリクエストを受信した 3. バックグラウンドリクエストでエラーが発生した
ENCIF_INT2	374	チャネル 0 のデータ受信に伴って割り込みが発生します。
ENCIF_INT4	376	フォアグラウンド通信の終了やエラー発生によって、チャネル 1 が通信待機状態になると割り込みが発生します。
ENCIF_INT5	377	チャネル 1 が通信待機状態になったときのバックグラウンド通信が、以下の場合に割り込みが発生します。 1. バックグラウンドレスポンスを受信した 2. エンコーダがバックグラウンドリクエストを受信した 3. バックグラウンドリクエストでエラーが発生した
ENCIF_INT6	378	チャネル 1 のデータ受信に伴って割り込みが発生します。

4.8 定数/エラーコード一覧

主要な定数とエラーコードの一覧を記載します。表 4.3 に EnDat 3 リクエストコード / データ、表 4.4 に EnDat 3 バックグラウンドリクエストコード、表 4.5 に EnDat 3 フレーム ID、表 4.6 に EnDat 3 インタフェース レジスタ設定値を示します。エラーコードは「4.10.3(1) r_endat3_err_t」を参照してください。

表 4.3 EnDat 3 リクエストコード / データ

定数名	設定値	内容 注
R_ENDAT3_REQC_DATA0	0x00	DATA0 リクエスト, LPF 送信リスト 0 を選択する
R_ENDAT3_REQC_DATA1	0x01	DATA1 リクエスト, LPF 送信リスト 1 を選択する
R_ENDAT3_REQC_DATA2	0x02	DATA2 リクエスト, LPF 送信リスト 2 を選択する
R_ENDAT3_REQC_DATA3	0x03	DATA3 リクエスト, LPF 送信リスト 3 を選択する
R_ENDAT3_REQC_DATA4	0x04	DATA4 リクエスト, LPF 送信リスト 4 を選択する
R_ENDAT3_REQC_DATA5	0x05	DATA5 リクエスト, LPF 送信リスト 5 を選択する
R_ENDAT3_REQC_DATA6	0x06	DATA6 リクエスト, LPF 送信リスト 6 を選択する
R_ENDAT3_REQC_DATA7	0x07	DATA7 リクエスト, LPF 送信リスト 7 を選択する
R_ENDAT3_REQC_DATA	0x08	DATA リクエスト
R_ENDAT3_REQC_DATANOP	0x09	DATANOP リクエスト
R_ENDAT3_REQC_RESET	0x0B	RESET リクエスト, エンコーダをリセットする
R_ENDAT3_REQC_CLEAR	0x0C	CLEAR リクエスト, エンコーダの状態をリセットする
R_ENDAT3_REQC_ECHO	0x0E	ECHO リクエスト, 遅延時間測定用
R_ENDAT3_REQC_RATE	0x10	RATE リクエスト, データレートを切り替える
R_ENDAT3_REQC_HELLO	0x22	HELLO リクエスト, エンコーダの応答確認用
R_ENDAT3_REQC_BUSBC	0x80	BUSBC リクエスト, ブロードキャスト用バスコマンド
R_ENDAT3_REQC_BUSP2P	0x81	BUSP2P リクエスト, P2P 通信用バスコマンド
R_ENDAT3_REQC_BUSINIT	0x82	BUSINIT リクエスト, バス設定を初期化する
R_ENDAT3_REQC_FORCE	0x90	FORCE リクエスト, ダイナミックサンプリングを行う
R_ENDAT3_REQD_DATANOP	0x0000	DATANOP リクエスト用のデータ部
R_ENDAT3_REQD_RESET	0xBBBB	RESET リクエスト用のデータ部
R_ENDAT3_REQD_CLEAR_MAX	0x0007	CLEAR リクエスト用データ部の最大値
R_ENDAT3_REQD_RATE_MAX	0x0001	RATE リクエスト用データ部の最大値
R_ENDAT3_REQD_HELLO	0x2222	HELLO リクエスト用のデータ部
R_ENDAT3_REQD_BUSINIT	0x8282	BUSINIT リクエスト用のデータ部
R_ENDAT3_REQD_FORCE	0x0000	FORCE リクエスト用のデータ部

【注】 詳細は HEIDENHAIN 社に問い合わせることで入手可能な「EnDat 3 Interface Specification」を参照してください。

表 4.4 EnDat 3 バックグラウンドリクエストコード

定数名	設定値	内容 注
R_ENDAT3_BGREQ_NOP	0x01	NOP リクエスト
R_ENDAT3_BGREQ_READ	0x02	READ リクエスト, メモリ領域のリード
R_ENDAT3_BGREQ_WRITE	0x03	WRITE リクエスト, メモリ領域のライト
R_ENDAT3_BGREQ_RECONFIGURE	0x04	RECONFIGURE リクエスト
R_ENDAT3_BGREQ_AUTH	0x80	AUTH リクエスト
R_ENDAT3_BGREQ_PROTECT	0x81	PROTECT リクエスト
R_ENDAT3_BGREQ_SETPASS	0x83	SETPASS リクエスト
R_ENDAT3_BGREQ_LOCATE	0x84	LOCATE リクエスト

【注】 詳細は HEIDENHAIN 社に問い合わせることで入手可能な「EnDat 3 Interface Specification」を参照してください。

表 4.5 EnDat 3 フレーム ID

定数名	設定値	内容 注
R_ENDAT3_FID_NOP	0x00	空のコンテンツ
R_ENDAT3_FID_POS1	0x01	Position 1
R_ENDAT3_FID_POS2	0x02	Position 2
R_ENDAT3_FID_TOUCHPROBE	0x03	タッチプローブのプローブ信号とタイムスタンプ
R_ENDAT3_FID_POS_ABS	0x04	インクリメンタルエンコーダの絶対位置
R_ENDAT3_FID_ZERODATA	0x05	データ値 0x00
R_ENDAT3_FID_ERRMSG	0x0A	エラーメッセージと警告メッセージ
R_ENDAT3_FID_EVALNUM	0x11	Valuation ナンバー
R_ENDAT3_FID_MOUNT0	0x12	Mounting パラメータ 0～2
R_ENDAT3_FID_MOUNT1	0x13	Mounting パラメータ 3～5
R_ENDAT3_FID_COMMU	0x1A	Commutation と Limit position
R_ENDAT3_FID_SENSOR_TEMP_MAX	0x20	モータ巻線温度最大値
R_ENDAT3_FID_SENSOR_TEMP_INT	0x21	エンコーダ内部温度
R_ENDAT3_FID_SENSOR_TEMP_M1	0x22	モータ巻線温度 1
R_ENDAT3_FID_SENSOR_TEMP_M2	0x23	モータ巻線温度 2
R_ENDAT3_FID_SENSOR_TEMP_M3	0x24	モータ巻線温度 3
R_ENDAT3_FID_SF_POS1	0x50	Position 1 セーフティフレーム
R_ENDAT3_FID_BGRSP	0x60	最新のバックグラウンドレスポンス
R_ENDAT3_FID_BGREQ	0x68	最新のバックグラウンドリクエスト
R_ENDAT3_FID_BUS_ADDR	0x70	バスアドレス

【注】 詳細は HEIDENHAIN 社に問い合わせることで入手可能な「EnDat 3 Interface Specification」を参照してください。

表 4.6 EnDat 3 インタフェース レジスタ設定値

定数名	設定値	内容
R_ENDAT3_BRATE_25M	0x0001u	データレート 25 Mbps のレジスタ設定値
R_ENDAT3_BRATE_12_5M	0x0003u	データレート 12.5 Mbps のレジスタ設定値
R_ENDAT3_P2P_MODE	false	P2P モードのトポロジー設定値
R_ENDAT3_BUS_MODE	true	バスモードのトポロジー設定値
R_ENDAT3_NUM_BUS_SLAVES	1 (P2P モード) 3 (バスモード)	接続しているスレーブ数
R_ENDAT3_WD_11US	0x016Fu	Watchdog Timer の待ち時間 11 us の設定値

4.9 固定幅整数一覧

表 4.7 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 4.7 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）

4.10 構造体/共用体/列挙型一覧

4.10.1 構造体

(1) r_endat3_info_t

EnDat 3 制御部の初期化情報

```
typedef struct
{
```

bool	topology;	P2P モード、またはバスモードの指定 P2P モードのとき R_ENDAT3_P2P_MODE を指定してください。バスモードのとき、R_ENDAT3_BUS_MODE を指定してください。
uint16_t	num_bus_slaves;	バスモードのスレーブ数設定 P2P モードでは 1 を設定してください。バスモードでは、接続するスレーブの数を設定してください。バスモードのとき、本設定は CONFIG_0 レジスタの NUM_BUS_SLAVES ビットに反映されます。
uint16_t	bitrate;	データレート設定 「表 4.6 EnDat 3 インタフェース レジスタ設定値」参照 本設定は BRATE レジスタの CLKSEL ビット, CLKDIV ビットに反映されます。
endat3_wait_cb_t	p_enc_init_reset_wait;	初期の HELLO リクエスト発行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「4.5.1 endat3_init_reset_wait_callback」参照 NULL は設定しないでください。
uint16_t *	p_rd_buf;	受信データ転送用バッファのポインタ EnDat 3 エンコーダからの受信データ転送用に確保したバッファへのポインタを指定してください。受信データを格納するのに十分な長さのバッファを用意してください。受信データ長は使用する LPF のフレーム数に応じて変わります。LPF を n フレーム使うときに必要なバッファ長は、接続したエンコーダ 1 台当たり 9+4n です。バスモードでは、各エンコーダからの受信データがまとめて格納されます。
uint16_t	rd_buf_length;	受信データ転送用バッファ長 p_rd_buf にポインタを格納したバッファの長さを設定してください。

```
} r_endat3_info_t
```

(2) r_endat3_req_t

EnDat 3 準拠エンコーダに送信するリクエスト情報

フォアグラウンドリクエスト送信コマンドでは、エンコーダに対してリクエストコードとリクエストデータを組み合わせて送信します。バックグラウンドリクエスト送信コマンドでは、バックグラウンドリクエストコードとバックグラウンドリクエストデータを 3 回のフォアグラウンドリクエストに分割して、バックグラウンドデータ送信を示すリクエストコードと組み合わせて送信します。

```
typedef struct
{
    bool            en_bus_req;   バスリクエスト設定 (true: 有効、false: 無効)
    uint8_t         bus_code;     バスコード
                                  バスリクエストのリクエストコードを指定します。「表 4.3 EnDat 3 リクエストコード / データ」参照。バスリクエストでのみ有効です。BUSBC または BUSP2P リクエストを設定してください。
    uint8_t         bus_addr;     バスアドレス
                                  バスリクエストのバスアドレスを指定します。BUSBC リクエストではフォアグラウンドアドレスとして、BUSP2P リクエストではバスアドレスとして使われます。その他の場合、設定値は無視されます。
    uint8_t         bus_bgaddr;   バスバックグラウンドアドレス
                                  バスリクエスト、BUSBC リクエストのバックグラウンドアドレスとして使われます。その他の場合、設定値は無視されます。
    uint8_t         req_code;     リクエストコード
                                  「表 4.3 EnDat 3 リクエストコード / データ」参照。バックグラウンドリクエスト送信コマンド(「4.4.5(2) ENDAT3_CMD_BGREQ」)使用時には、DATA0～DATA7 リクエストまたは DATA リクエストの何れかを設定してください。
    uint16_t        req_data;     リクエストデータ
                                  「表 4.3 EnDat 3 リクエストコード / データ」参照。リクエストコードによって、データの値や範囲が示されているものは、指定範囲のデータのみ有効です。バックグラウンドリクエスト送信コマンド使用時には、設定値は無視されます。
    uint16_t        watchdog;     Watchdog timer 設定時間
                                  「表 4.6 EnDat 3 インタフェース レジスタ設定値」参照。リクエスト送信完了から High Priority Frame (HPF) 受信完了までの監視時間を設定します。監視時間は設定値×30 ns となります。
    uint8_t         cmd;          バックグラウンドリクエストコード
                                  「表 4.4 EnDat 3 バックグラウンドリクエストコード」参照。バックグラウンドリクエスト送信コマンド(「4.4.5(2) ENDAT3_CMD_BGREQ」)使用時のみ有効です。
    uint8_t         args[5];      バックグラウンドリクエストデータ
                                  バックグラウンドリクエスト送信コマンド使用時のみ有効です。
    bool            elc;          ELC モード設定 (true: 有効, false: 無効)
                                  フォアグラウンドリクエスト送信コマンド(「4.4.5(1) ENDAT3_CMD_REQ」)使用時のみ有効です。
    r_endat3_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「4.5.2 endat3_callback」および「4.5.3 endat3_pos_callback」参照。NULL は設定しないでください。
    r_endat3_isr_ready_cb_t   p_isr_ready;    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「4.5.4 endat3_ready_callback」参照。NULL は設定しないでください。
} r_endat3_req_t
```

(3) r_endat3_result_t

送受信結果

```
typedef struct
{
    r_endat3_req_err_t      result;     リクエストの送受信結果
                                        「4.10.3(3) r_endat3_req_err_t」参照
    r_endat3_data_t         data;       受信データ
                                        「4.10.1(4) r_endat3_data_t」参照
    r_endat3_status_t       status;     エンコーダのステータス
                                        「4.10.1(5) r_endat3_status_t」参照
} r_endat3_result_t
```

(4) r_endat3_data_t

受信データ

```
typedef struct
{
    uint8_t     timestamp;  タイムスタンプ表示
    uint64_t    data;       位置データ (Position 1)
                            High Priority Frame (HPF) の第 5～0 バイトが格納されます。
    uint8_t     status;     HPF の Status バイトが格納されます。
    uint32_t    lph;        Low Priority Header (LPH) の第 3～0 バイトが格納されます。
    uint8_t     lph_bg;     LPH の BG フィールド(BG.ERR_EXEC および BG.STATUS)が格納されます。
    uint8_t     lph_zact;   LPH の ZACT フィールドが格納されます。
    uint8_t     lph_yact;   LPH の YACT フィールドが格納されます。
    uint8_t     lph_xdim;   LPH の XDIM フィールドが格納されます。
    uint16_t    lpf[60];    Low Priority Frames (LPFs) を 2 バイト単位に分割して配列に格納します。1 フレームは 8 バイトのため、4 要素で 1 フレームに当たります。1 回のフォアグラウンドリクエストで受信される LPFs は最大 15 フレームで、配列の先頭から lph_xdim フレーム分が有効です。
    uint32_t    crcerr;     HPF, LPF, HPH 各フレームの CRC エラーフラグが格納されます。
    uint64_t    bgrsp;      バックグラウンドレスポンスの第 5～0 バイトが格納されます。バックグラウンドリクエスト送信コマンド(「4.4.5(2) ENDAT3_CMD_BGREQ」)に対するコールバックでのみ有効です。
} r_endat3_data_t
```

(5) r_endat3_status_t

エンコーダのステータスおよび High Priority Frame (HPF) の有効性 (HPF.STATUS)

```
typedef struct
{
    bool        err_req;    リクエストコードのサポートエラー
                            (true: リクエストコードは非サポート, false: サポート)
    bool        rm;         絶対位置の有効性
                            (true: 絶対位置有効, false: 有効ではない)
    bool        hpfv;       HPF データの有効性
                            (true: HPF データ有効, false: HPF データ無効)
    bool        w;          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)
    bool        f;          エンコーダのデバイスエラー
                            (true: エラーあり, false: エラーなし)
} r_endat3_status_t
```

(6) r_endat3_protocol_err_t

EnDat 3 I/F およびエンコーダのフォアグラウンド通信エラー情報

```
typedef struct
{
    bool        comm_error;  通信エラー (cs_error, phy_error, wd_error, strobe_error, buff_ovr の
                             論理和) (true: エラーあり, false: エラーなし)
    bool        cs_error;    CRC エラー (true: エラーあり, false: エラーなし)
    bool        phy_error;   マンチェスター符号エラー (true: エラーあり, false: エラーなし)
    bool        wd_error;    ウォッチドッグエラー(true: エラーあり, false: エラーなし)
    bool        strobe_error; ストローブエラー (通信中に、通信開始のストローブを検出した)
                             (true: エラーあり, false: エラーなし)
    bool        buffer_ovr;  受信バッファオーバーラン (true: エラーあり, false: エラーなし)
} r_endat3_protocol_err_t
```

4.10.2 共用体

使用しません。

4.10.3 列挙型

(1) r_endat3_err_t

エンコーダ I/F のエラーコード

```
typedef enum
{
    ENDAT3_SUCCESS            =0,    正常終了
    ENDAT3_ERR_INVALID_ARG    ,      引数異常
    ENDAT3_ERR_BUSY           ,      API を実行できない状態
    ENDAT3_ERR_ACCESS         ,      API の実行順序エラー
    ENDAT3_ERR_DRV                   ドライバ内部エラー
} r_endat3_err_t
```

(2) r_endat3_cmd_t

R_ENDAT3_Control 関数を使用する時のコマンド設定

```
typedef enum
{
    ENDAT3_CMD_REQ        ,    エンコーダへフォアグラウンドリクエストを送信
    ENDAT3_CMD_BGREQ      ,    エンコーダへバックグラウンドリクエストを送信
    ENDAT3_CMD_RATE       ,    エンコーダインタフェースのデータレートを変更する
    ENDAT3_CMD_STOP            ELC モードによるリクエスト送受信停止
} r_endat3_cmd_t
```

(3) r_endat3_req_err_t

リクエストの送受信結果

```
typedef enum
{
    ENDAT3_REQ_SUCCESS  =0,    データ送受信正常終了
    ENDAT3_REQ_ERR             データ送受信エラー発生
} r_endat3_req_err_t
```

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 EnDat 3 準拠エンコーダ 「ECI 1319 E30-R2」および、バス接続エンコーダ「EQN 1337 E30-RB」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リクエストを EnDat 3 エンコーダへ送信する。
2) コンソールから入力したコマンドにより EnDat 3 エンコーダのメモリにリード・ライトを行う。
3) EnDat 3 エンコーダから受信したデータをコンソールに表示する。
4) EnDat 3 I/F の ELC イベント入力トリガ機能を使用してリクエストを送信する。(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) 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。

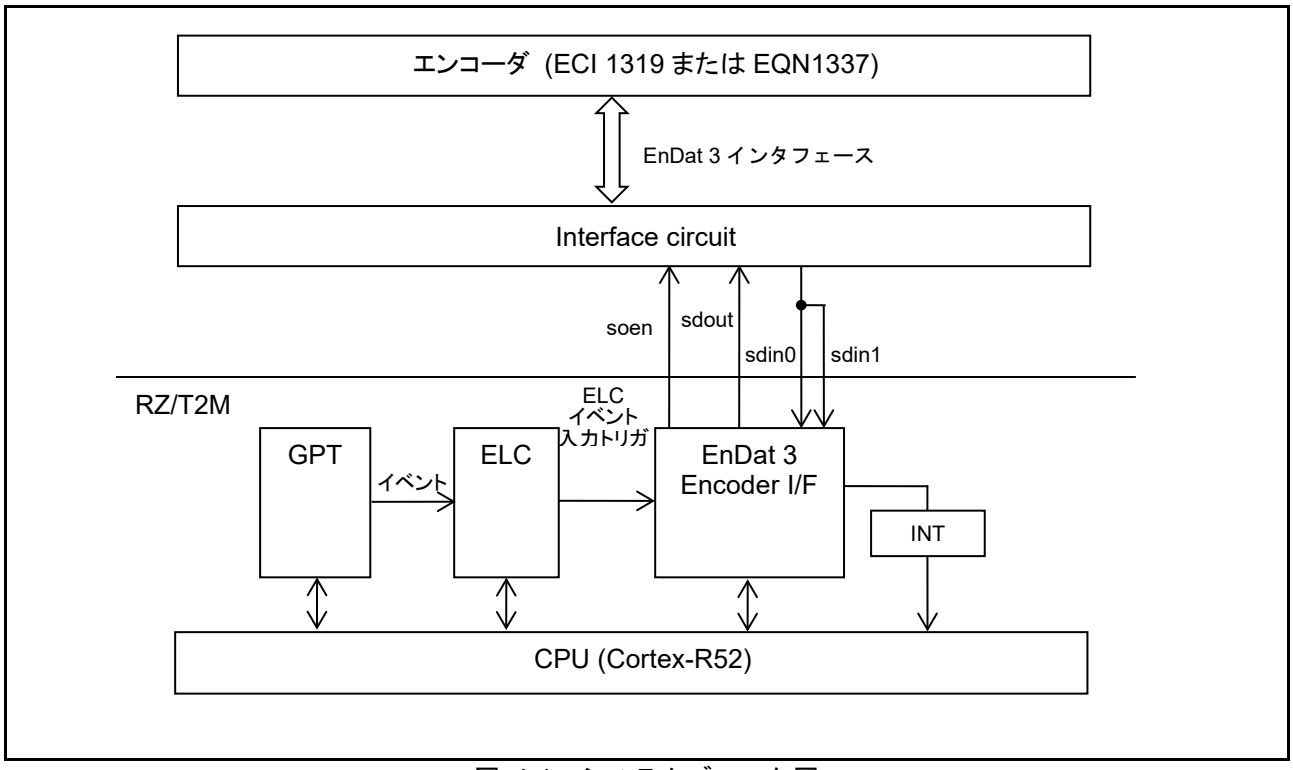

図 4-1 システムブロック図

(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。

EnDat 3 ドライバには、R_ENDAT3_Open 関数で構成される開始処理部、R_ENDAT3_Close 関数で構成される終了処理部、R_ENDAT3_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、EnDat 3 ドライバを制御し、リクエスト送信を行う EnDat 3 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。

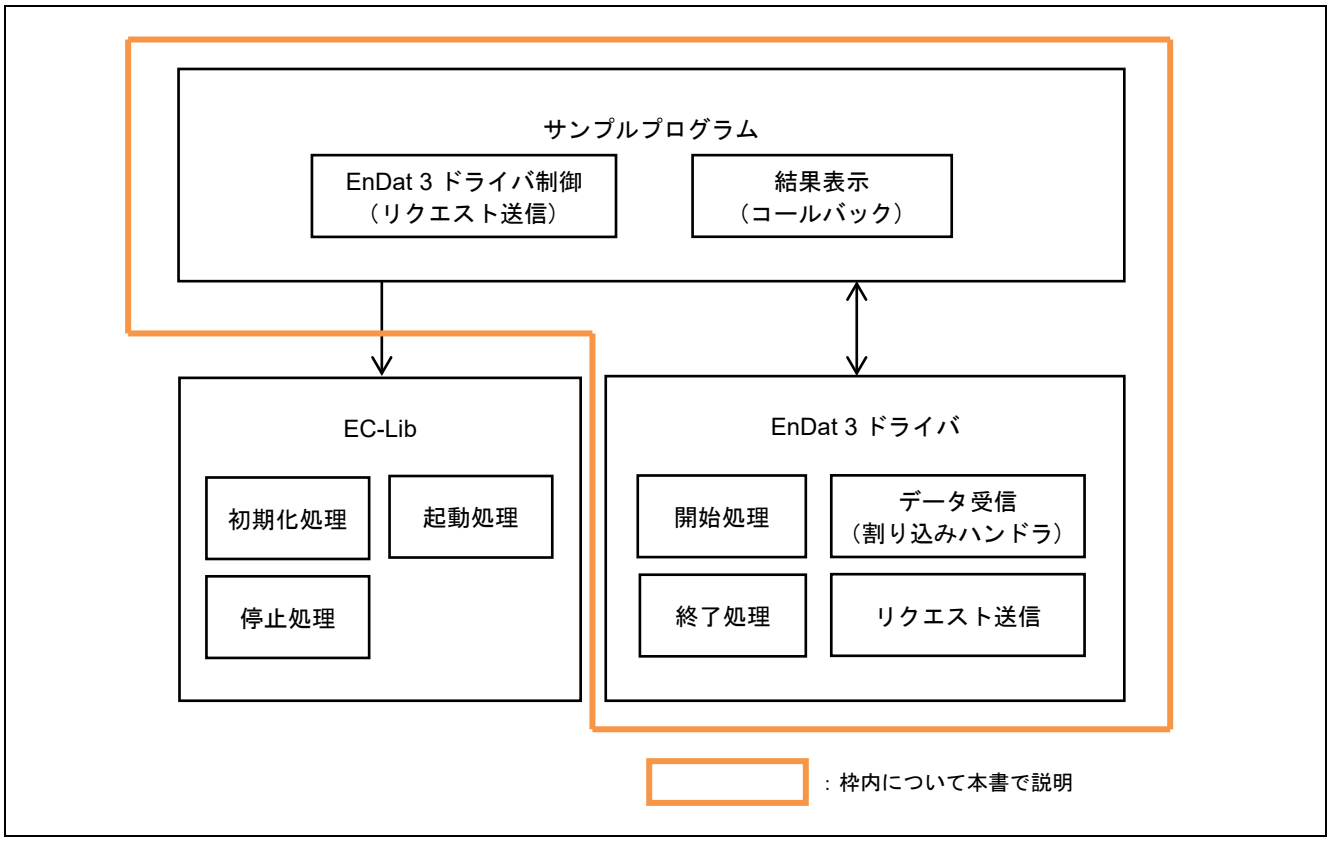

図 4-2　ソフトウェア構成図

4.11.2 サンプルプログラム関数一覧

表 4.8 に主要なサンプルプログラム関数一覧を示します

表 4.8　サンプルプログラム関数一覧

関数名	ページ番号	
	仕様	フローチャート
hal_entry	28	-
enc_main	28	37
endat3_cmd_control	28	38
endat3_power_on_wait	28	-
endat3_init_reset_wait_callback	29	-
endat3_pos	29	39
endat3_read	29	40
endat3_write	30	41
endat3_bus_addr	30	42
endat3_rate	30	43
endat3_elctimer	31	44
endat3_stop	31	45
endat3_req	31	46
endat3_bus_req	32	47
endat3_callback	32	48
endat3_pos_callback	32	49
endat3_ready_callback	33	49
get_cmd	33	-
cmd_exit	33	-
result_display	33	-
result_access_display	34	-
result_fg_display	34	-
timer_start	34	-
timer_stop	34	-

4.11.3 サンプルプログラム関数仕様

(1) hal_entry

hal_entry	
概　要	EnDat 3 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	EnDat 3 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

(2) enc_main

enc_main		
概　要	EnDat 3 サンプルプログラムのメイン関数	
ヘッダ	-	
宣　言	int32_t enc_main(uint8_t ch);	
説　明	EnDat 3 サンプルプログラムのメイン関数です。詳細は、「4.11.6(1) enc_main フローチャート」参照。	
引　数	ch	：エンコーダチャネル番号 0: ch0 を指定, 1: ch1 を指定
リターン値	0	：正常終了
	0 以外	：異常終了（EnDat 3 I/F ドライバのエラーコード）

(3) endat3_cmd_control

endat3_cmd_control	
概　要	EnDat 3 ドライバ制御関数
ヘッダ	-
宣　言	static void endat3_cmd_control(void);
説　明	以下の処理を行います。 • EnDat 3 エンコーダ制御の開始 • コンソールコマンドの入力処理 • EnDat 3 エンコーダ制御の終了
引　数	なし
リターン値	なし

(4) endat3_power_on_wait

endat3_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 endat3_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

(5) endat3_init_reset_wait_callback

endat3_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 endat3_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 300ms を生成するコールバック関数です。 詳細は「4.5.1 endat3_init_reset_wait_callback」を参照してください。
引　数	なし
リターン値	なし

(6) endat3_pos

endat3_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 endat3_pos(uint32_t arg_num, char_t *p_arg[]);
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。取得のため、リクエストコードを R_ENDAT3_REQC_DATANOP としてフォアグラウンドリクエストを送信しています。詳細は「4.11.6(3) endat3_pos フローチャート」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 *p_arg[]　：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

(7) endat3_read

endat3_read	
概　要	エンコーダのメモリからデータを指定ワード数読み出す関数
ヘッダ	-
宣　言	static void endat3_read(uint32_t arg_num, char_t *p_arg[]);
説　明	コンソールコマンド read が入力された場合に実行される関数です。読み出しのため、リクエストコードを R_ENDAT3_REQC_DATA0 としてバックグラウンドリクエスト R_ENDAT3_BGREQ_READ を送信しています。詳細は「4.11.6(4) endat3_read フローチャート」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 *p_arg[]　：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

(8) endat3_write

endat3_write	
概　要	エンコーダのメモリにデータを書き込む関数
ヘッダ	-
宣　言	static void endat3_write(uint32_t arg_num, char_t *p_arg[]);
説　明	コンソールコマンド write が入力された場合に実行される関数です。書き込みのため、リクエストコードを R_ENDAT3_REQC_DATA0 としてバックグラウンドリクエスト R_ENDAT3_BGREQ_WRITE を送信しています。詳細は「4.11.6(5) endat3_write フローチャート」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 *p_arg[]　：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

(9) endat3_bus_addr

endat3_bus_addr	
概　要	エンコーダ I/F のバスアドレスを変更する関数
ヘッダ	-
宣　言	static void endat3_bus_addr(uint32_t arg_num, char_t *p_arg[]);
説　明	コンソールコマンド bus_addr が入力された場合に実行される関数です。エンコーダがバス接続されている場合に、read, write コマンドの対象となるエンコーダを変更します。詳細は「4.11.6(6) endat3_bus_addr フローチャート」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 *p_arg[]　：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

(10) endat3_rate

endat3_rate	
概　要	エンコーダ I/F のデータレートを変更する関数
ヘッダ	-
宣　言	static void endat3_rate(uint32_t arg_num, char_t *p_arg[]);
説　明	コンソールコマンド rate が入力された場合に実行される関数です。エンコーダ I/F のデータレートを変更します。詳細は「4.11.6(7) endat3_rate フローチャート」を参照してください。 (エンコーダ本体のデータレートは、事前にフォアグラウンドリクエストを使って変更してください。エンコーダ I/F のデータレートは、本体のデータレートと一致していないと通信できません。)
引　数	arg_num　：コンソールから入力された文字列の数 *p_arg[]　：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

(11) endat3_elctimer

endat3_elctimer	
概　要	ELC イベントに同期してエンコーダから位置情報を取得する関数
ヘッダ	-
宣　言	static void endat3_elctimer(uint32_t arg_num, char_t *p_arg[]);
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。リクエストコードを R_ENDAT3_REQC_DATANOP としたフォアグラウンドリクエストを使用します。詳細は「4.11.6(8) endat3_elctimer フローチャート」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 *p_arg[]　：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

(12) endat3_stop

endat3_stop	
概　要	ELC イベントに同期した位置情報の取得を終了する関数
ヘッダ	-
宣　言	static void endat3_stop(uint32_t arg_num, char_t *p_arg[]);
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。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。詳細は「4.11.6(9) endat3_stop フローチャート」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 (未使用) *p_arg[]　：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

(13) endat3_req

endat3_req	
概　要	エンコーダにフォアグラウンドリクエストを送信する関数
ヘッダ	-
宣　言	static void endat3_req(uint32_t arg_num, char_t *p_arg[]);
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。エンコーダにフォアグラウンドリクエストを送信し、受信したフォアグラウンドレスポンスを表示します。詳細は「4.11.6(10) endat3_req フローチャート」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 *p_arg[]　：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

(14) endat3_bus_req

endat3_bus_req		
概　要	エンコーダにバスモードでフォアグラウンドリクエストを送信する関数	
ヘッダ	-	
宣　言	static void endat3_bus_req(uint32_t arg_num, char_t *p_arg[]);	
説　明	コンソールコマンド bus_req が入力された場合に実行される関数です。エンコーダにバスモードでフォアグラウンドリクエストを送信し、受信したフォアグラウンドレスポンスを表示します。詳細は「4.11.6(11) endat3_bus_req フローチャート」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
	*p_arg[]	: 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	

(15) endat3_callback

endat3_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 endat3_callback(uint16_t num_result, r_endat3_result_t *p_result, r_endat3_protocol_err_t *p_err, uint16_t runtime);	
説　明	結果をメモリに保存します。	
引　数	num_result	: 送受信結果総数
	p_result	: 送受信結果
	p_err	: EnDat 3 I/F およびエンコーダのエラー情報
	runtime	: 伝搬時間表示
リターン値	なし	

(16) endat3_pos_callback

endat3_pos_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 endat3_pos_callback(uint16_t num_result, r_endat3_result_t *p_result, r_endat3_protocol_err_t *p_err, uint16_t runtime);	
説　明	連続して取得した結果をメモリに保存します。	
引　数	num_result	: 送受信結果総数
	p_result	: 送受信結果
	p_err	: EnDat 3 I/F およびエンコーダのエラー情報
	runtime	: 伝搬時間表示
リターン値	なし	

(17) endat3_ready_callback

endat3_ready_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 endat3_ready_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。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

(18) 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 *p_arg[], 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	p_arg　：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

(19) cmd_exit

cmd_exit	
概　要	EnDat 3 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 *p_arg[]);
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。EnDat 3 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) *p_arg[]　：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

(20) result_display

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 result_display(uint16_t num_result, r_endat3_result_t *p_result, r_endat3_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	num_result　：送受信結果総数 p_result　：送受信結果 p_err　：EnDat 3 I/F およびエンコーダのエラー情報
リターン値	なし

(21) result_access_display

result_access_display	
概　要	エンコーダのメモリアクセス結果を表示する関数
ヘッダ	-
宣　言	static void result_access_display(uint32_t addr, uint8_t num_words, r_endat3_result_t *p_result, r_endat3_protocol_err_t *p_err);
説　明	エンコーダへのメモリアクセス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	addr : メモリアドレス num_words : アクセスワード数 p_result : 送受信結果 p_err : EnDat 3 I/F およびエンコーダのエラー情報
リターン値	なし

(22) result_fg_display

result_fg_display	
概　要	エンコーダへのフォアグラウンドリクエスト結果を表示する関数
ヘッダ	-
宣　言	static void result_fg_display(uint16_t num_result, r_endat3_result_t *p_result, r_endat3_protocol_err_t *p_err, uint16_t runtime);
説　明	エンコーダへのフォアグラウンド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	num_result : 送受信結果総数 p_result : 送受信結果 p_err : EnDat 3 I/F およびエンコーダのエラー情報 runtime : 伝搬時間表示
リターン値	なし

(23) timer_start

timer_start	
概　要	GPT チャネル 0 の周期設定/起動関数
ヘッダ	bsp_api.h　hal_data.h
宣　言	static void timer_start(uint32_t us);
説　明	GPT チャネル 0 にタイマ周期を設定してタイマを起動します。
引　数	us : タイマ周期 [us]
リターン値	なし

(24) timer_stop

timer_stop	
概　要	GPT チャネル 0 のタイマ停止
ヘッダ	bsp_api.h　hal_data.h
宣　言	static void timer_stop(void);
説　明	GPT チャネル 0 のタイマを停止します。
引　数	なし
リターン値	なし

4.11.4 サンプルプログラムの変数一覧

表 4.9 に static 型変数を示します。

表 4.9 static 型変数

型	変数名	内容	使用関数
bool	endat3_flg	送受信完了フラグ (true: 送受信完了, false: 送受信中)	endat3_pos endat3_read endat3_write endat3_stop endat3_req endat3_bus_req endat3_callback endat3_ready_callback
uint16_t	endat3_num_result	データ取得結果総数	endat3_pos endat3_req endat3_bus_req endat3_callback
r_endat3_result_t	*p_endat3_result	データ取得結果を格納したアドレス	endat3_pos endat3_read endat3_write endat3_req endat3_bus_req endat3_callback
r_endat3_protocol_err_t	*p_endat3_err	エラー情報を格納したアドレス	endat3_pos endat3_read endat3_write endat3_req endat3_bus_req endat3_callback
uint16_t	endat3_runtime	伝搬時間表示	endat3_req endat3_bus_req endat3_callback
r_endat3_req_err_t	result_ti[ENDAT3_POS_NUM][ENDAT3_NUM_BUS_SLAVES]	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。	endat3_elctimer endat3_stop endat3_pos_callback
uint64_t	pos_ti[ENDAT3_POS_NUM][ENDAT3_NUM_BUS_SLAVES]	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。	endat3_elctimer endat3_stop endat3_pos_callback
r_endat3_status_t	status_ti[ENDAT3_POS_NUM][ENDAT3_NUM_BUS_SLAVES]	連続取得時のステータス 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。	endat3_elctimer endat3_stop endat3_pos_callback
uint8_t	pos_ti_valid	result_ti, pos_ti, status_ti 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。	endat3_elctimer endat3_stop endat3_pos_callback

型	変数名	内容	使用関数
uint8_t	pos_ti_num	result_ti, pos_ti, status_ti 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。	endat3_elctimer endat3_stop endat3_pos_callback
bool	pos_ti_full	result_ti, pos_ti, status_ti 配列の空き情報 (true: 空きなし, false: 空きあり)	endat3_elctimer endat3_pos_callback
int32_t	cur_id	EnDat 3 I/F ドライバ 使用 ID	enc_main endat3_cmd_control endat3_pos endat3_read endat3_write endat3_rate endat3_elctimer endat3_stop endat3_req endat3_bus_req

4.11.5 サンプルプログラムの定数一覧

表 4.10 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 4.10 主要な定数

定数名	設定値	内容
ENDAT3_ENC_TSAT_WAIT	1300u	電源投入後の待機時間 (1.3s)
ENDAT3_ENC_100US_WAIT	100u	EC-Lib 起動後の待機時間 (100us)
ENDAT3_ENC_INIT_RESET_WAIT	300u	エンコーダのリセット処理後に待機する時間 (300ms)
ENDAT3_POS_NUM	10u	連続受信した位置値格納用配列の要素数
ENDAT3_ADDR_MAX	0xFFFFFF	read, write コマンドのアドレス最大値
ENDAT3_DATA_MAX	0xFFFF	write コマンドのデータ最大値
ENDAT3_NUM_WORDS_MIN	1	read コマンドの読み出しワード数最小値
ENDAT3_NUM_WORDS_MAX	3	read コマンドの読み出しワード数最大値

4.11.6 メイン処理のフローチャート

(1) enc_main フローチャート

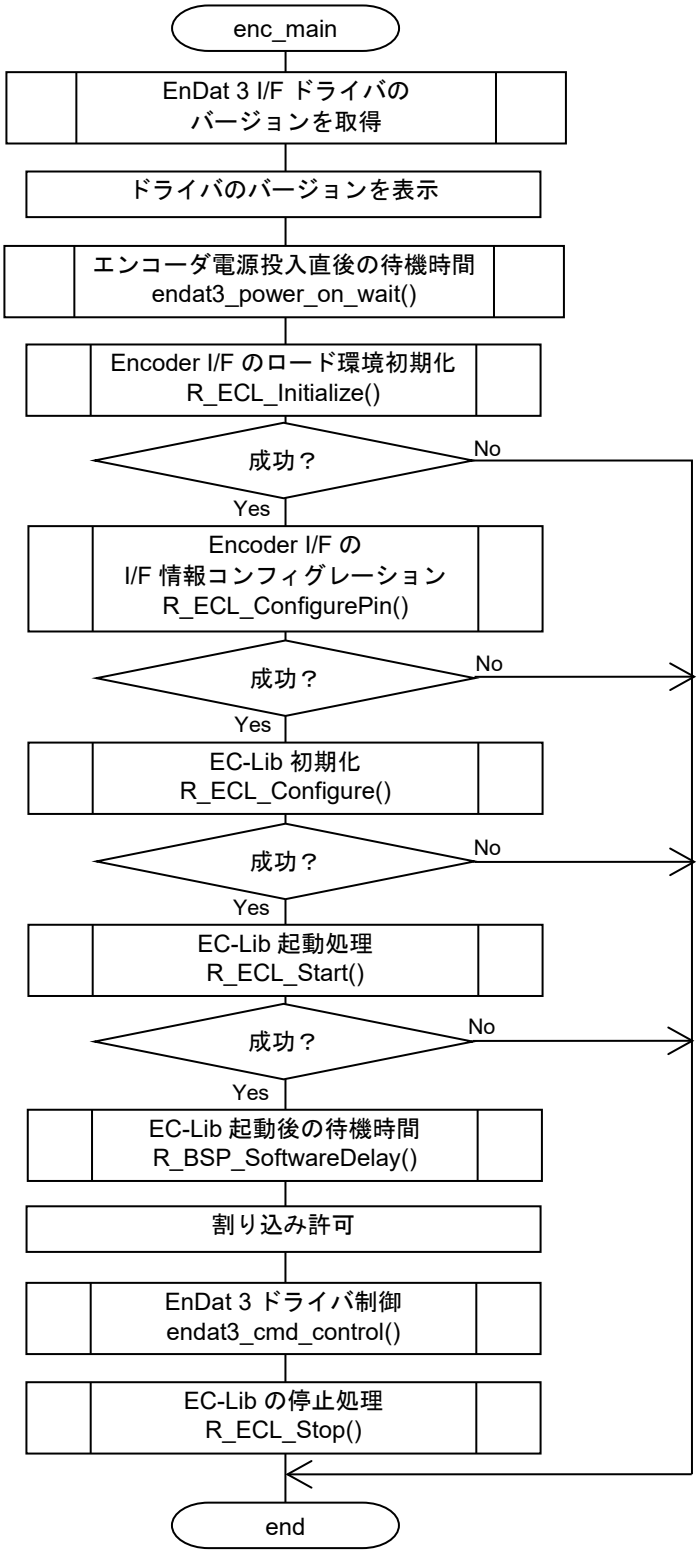

図 4-3 enc_main 関数のフローチャート

(2) endat3_cmd_control フローチャート

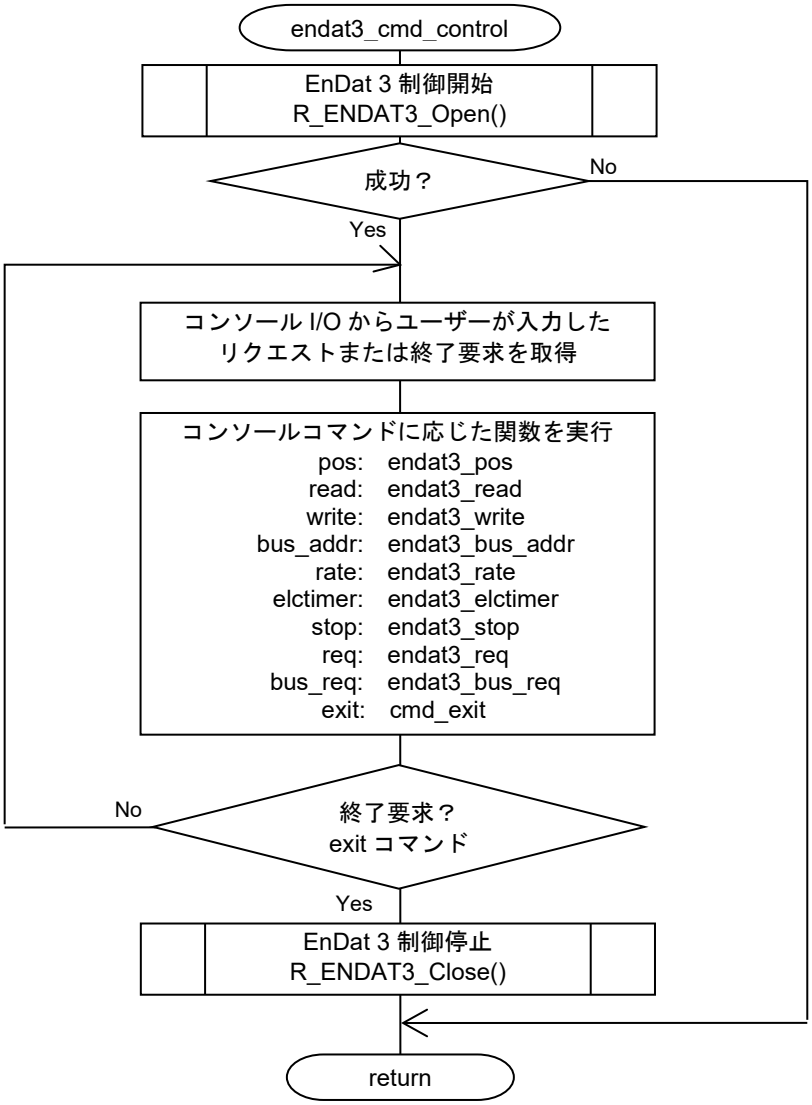

図 4-4 endat3_cmd_control 関数のフローチャート

(3) endat3_pos フローチャート

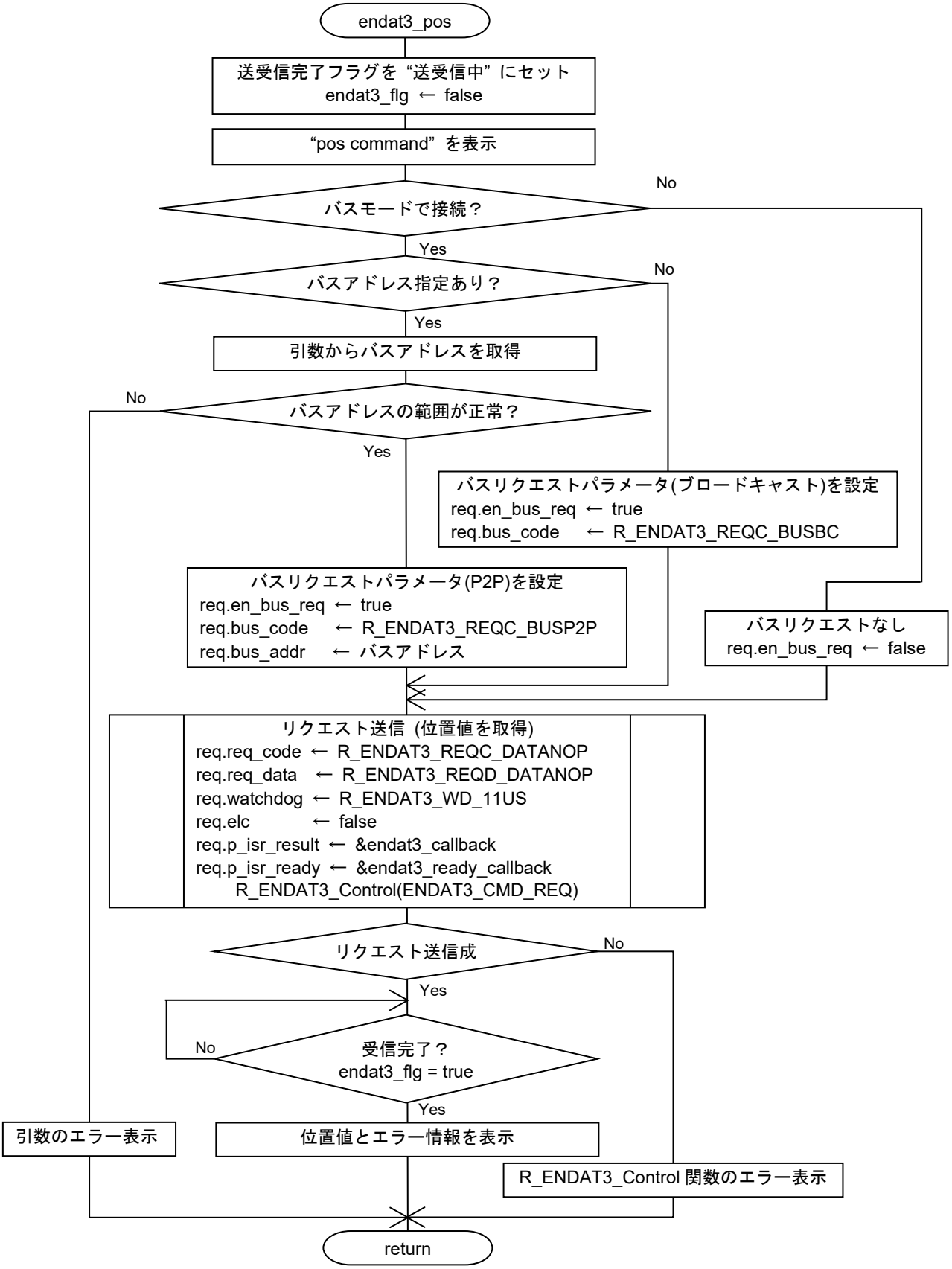

図 4-5 endat3_pos 関数のフローチャート

(4) endat3_read フローチャート

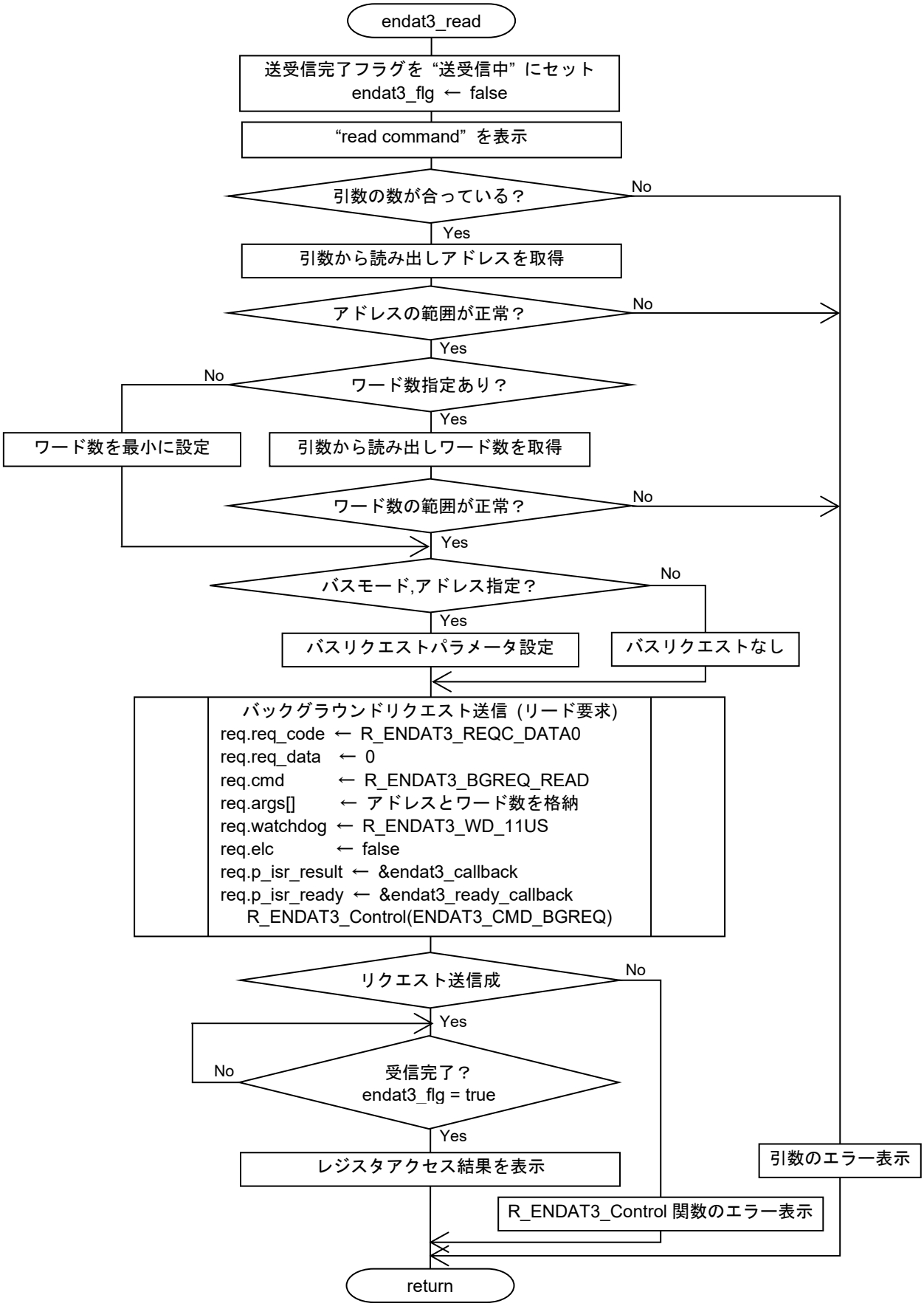

図 4-6 endat3_read 関数のフローチャート

(5) endat3_write フローチャート

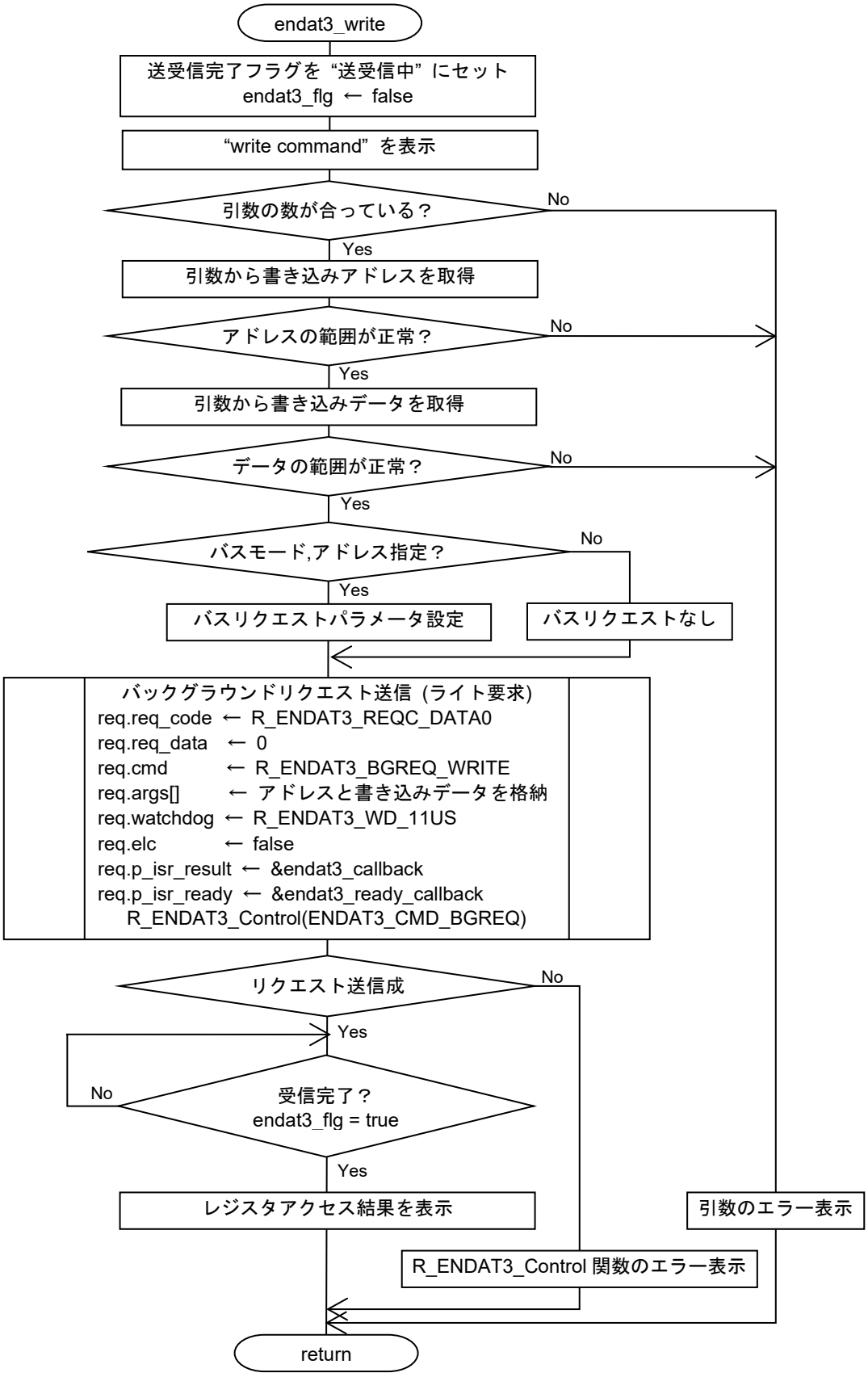

図 4-7 endat3_write 関数のフローチャート

(6) endat3_bus_addr フローチャート

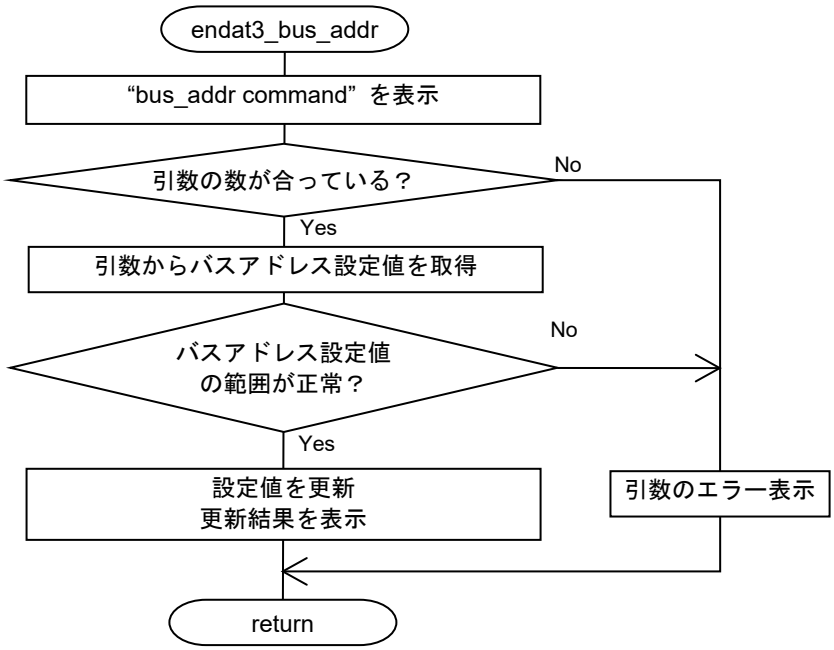

図 4-8 endat3_bus_addr 関数のフローチャート

(7) endat3_rate フローチャート

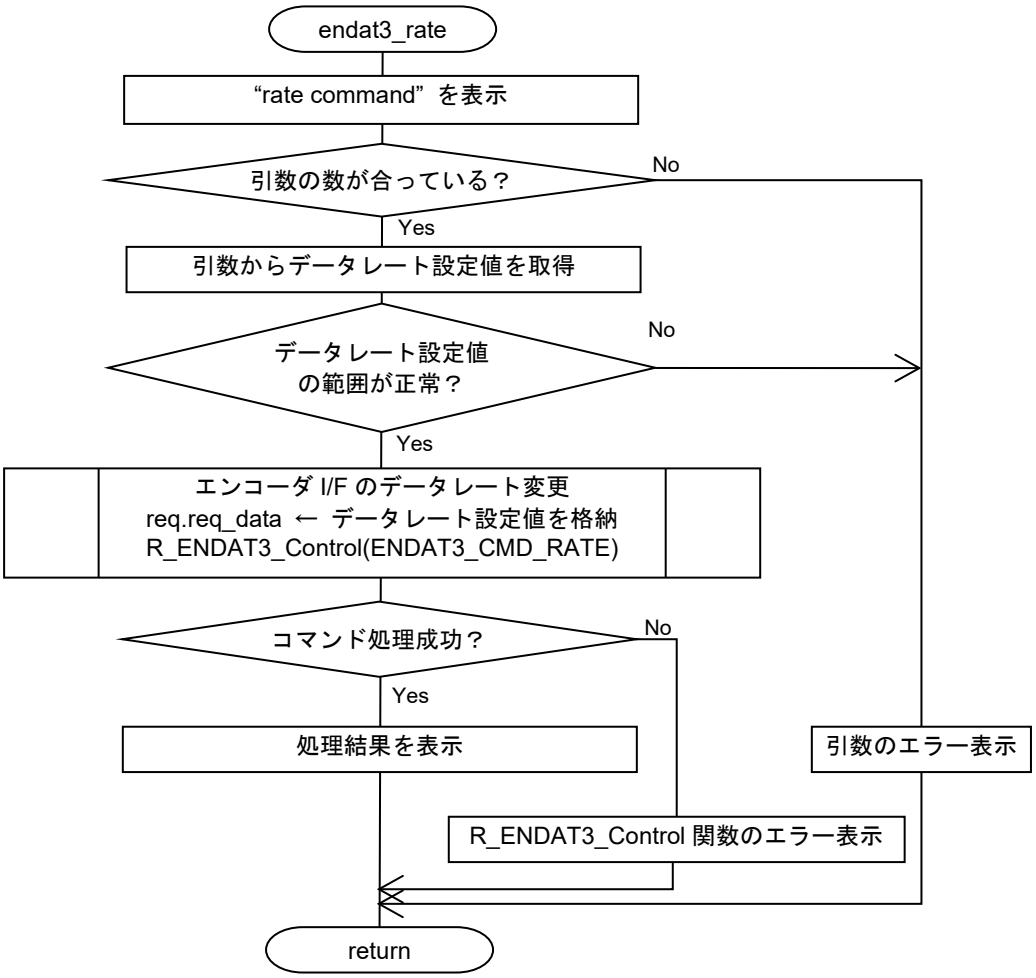

図 4-9 endat3_rate 関数のフローチャート

(8) endat3_elctimer フローチャート

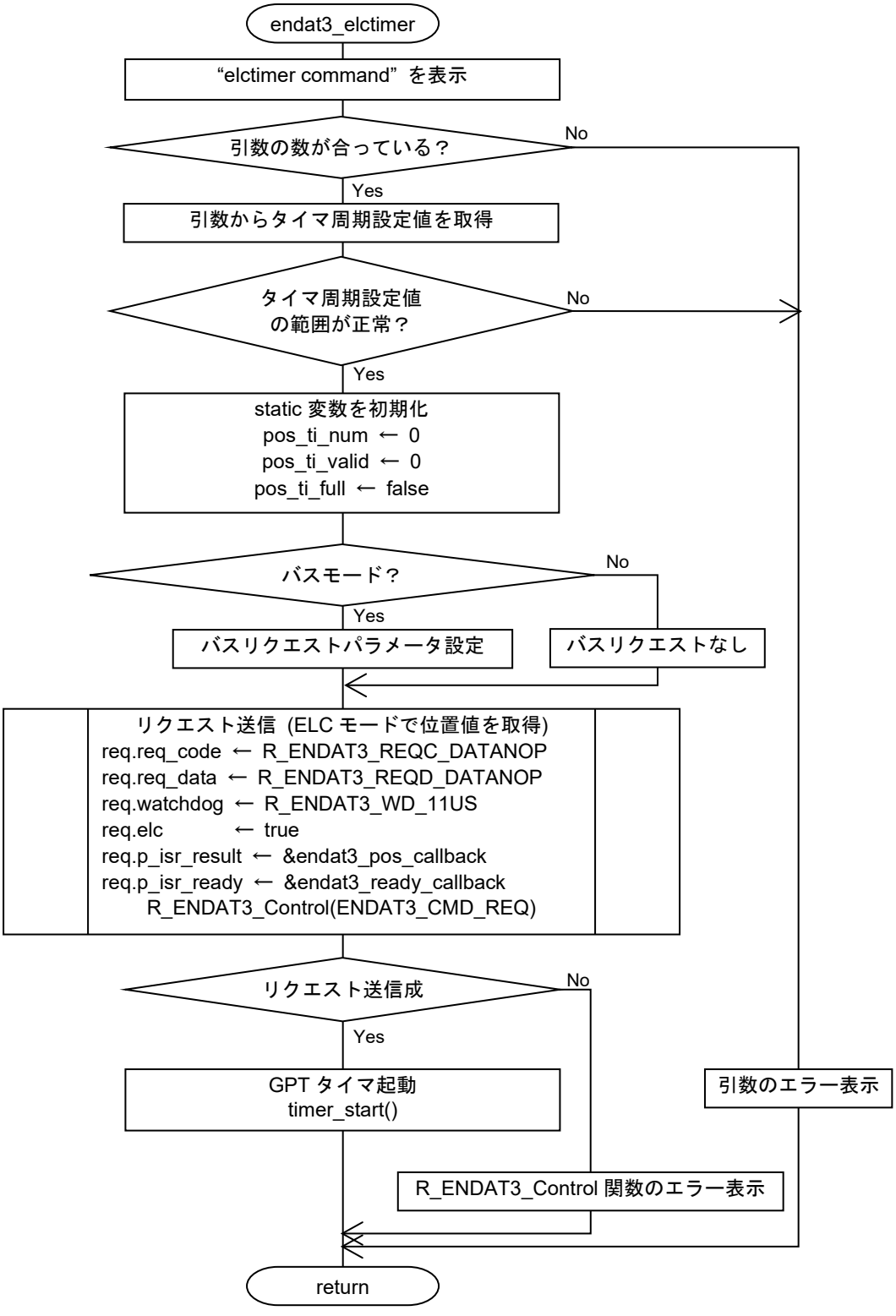

図 4-10 endat3_elctimer 関数のフローチャート

(9) endat3_stop フローチャート

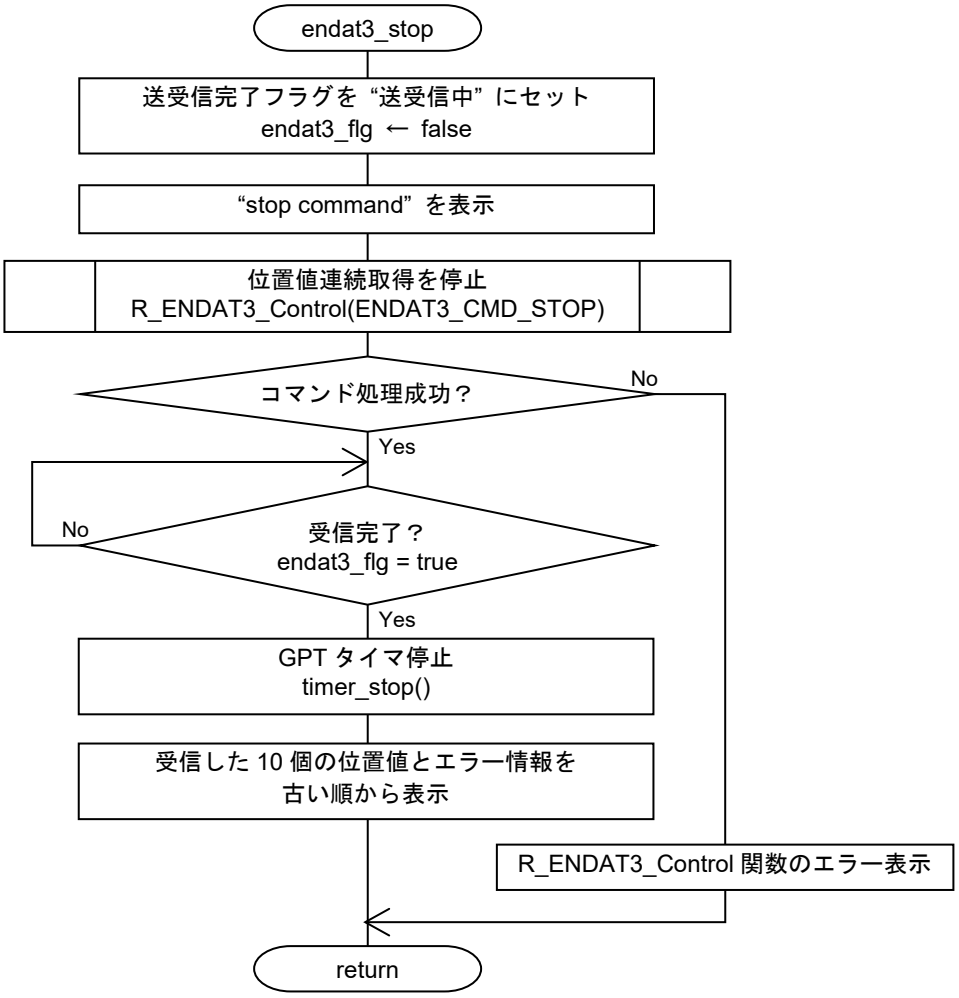

図 4-11 endat3_stop 関数のフローチャート

(10) endat3_req フローチャート

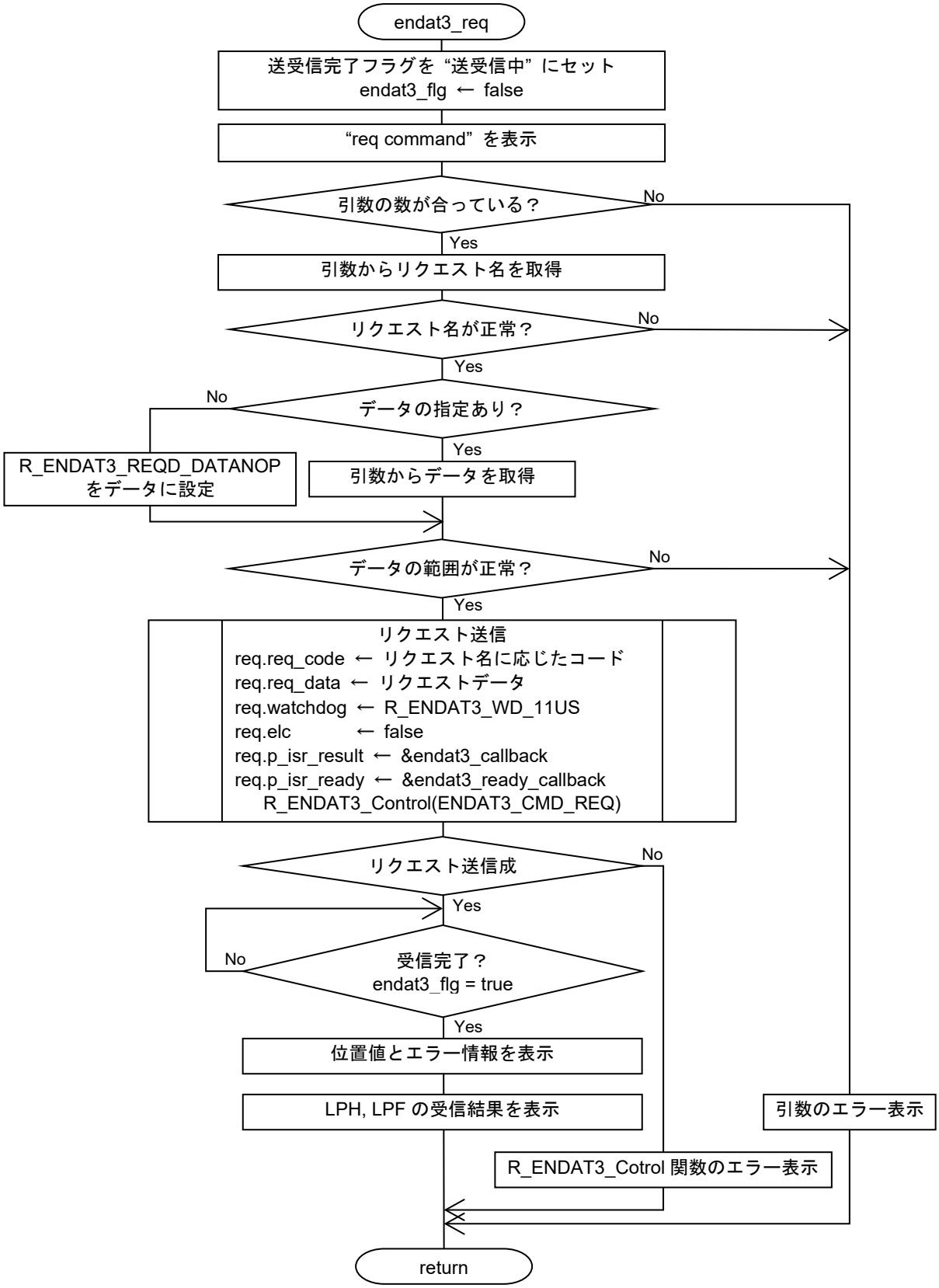

図 4-12 endat3_req 関数のフローチャート

(11) endat3_bus_req フローチャート

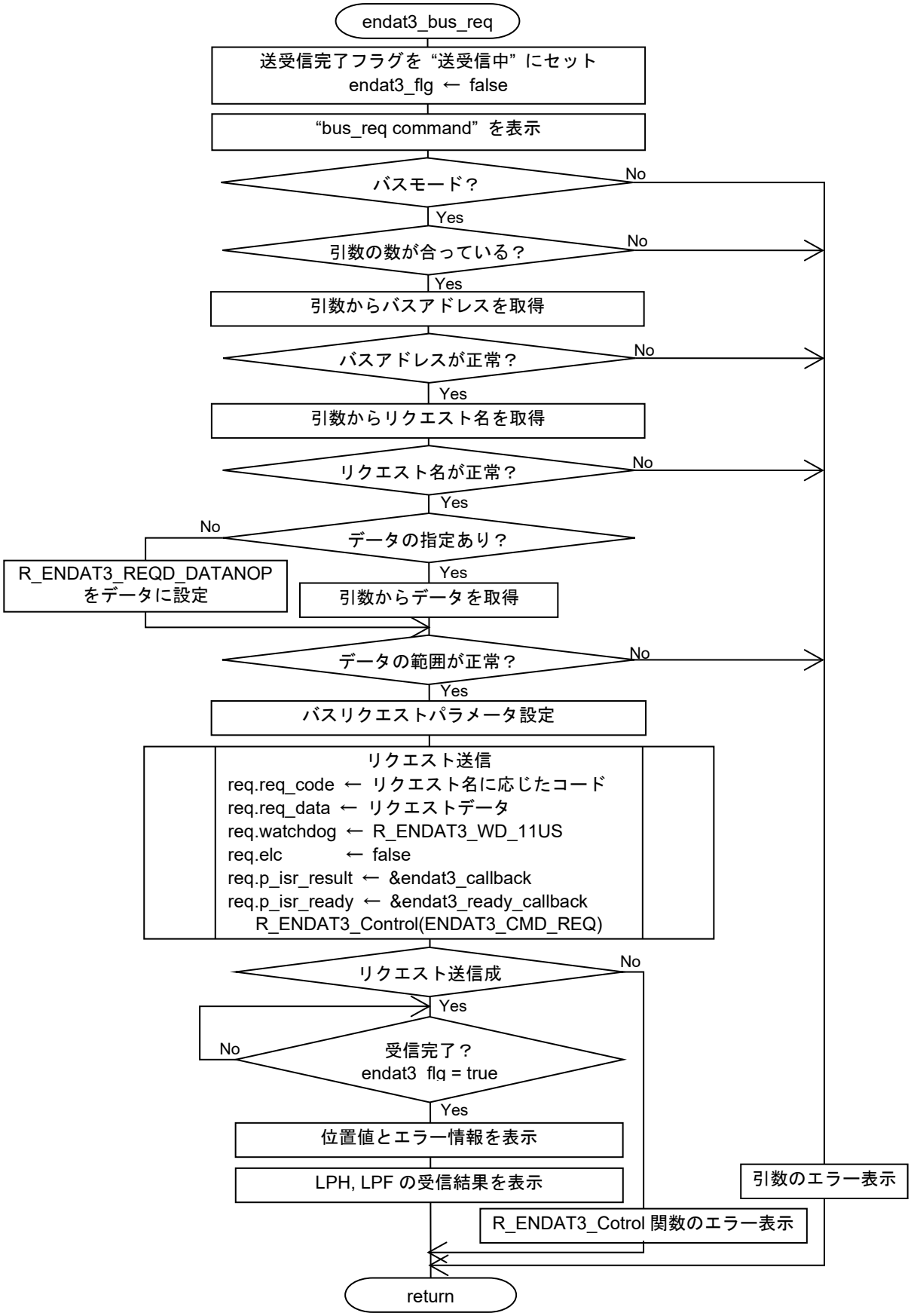

図 4-13 endat3_bus_req 関数のフローチャート

(12) endat3_callback フローチャート

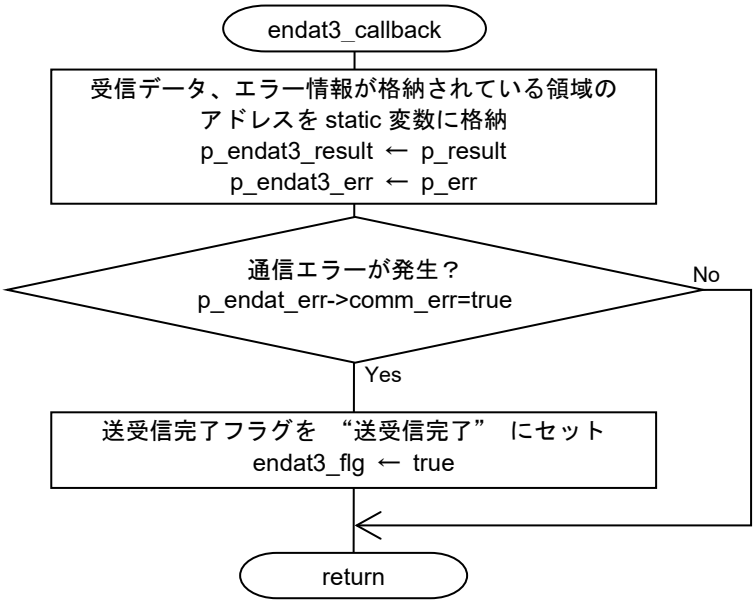

図 4-14 endat3_callback 関数のフローチャート

(13) endat3_pos_callback フローチャート

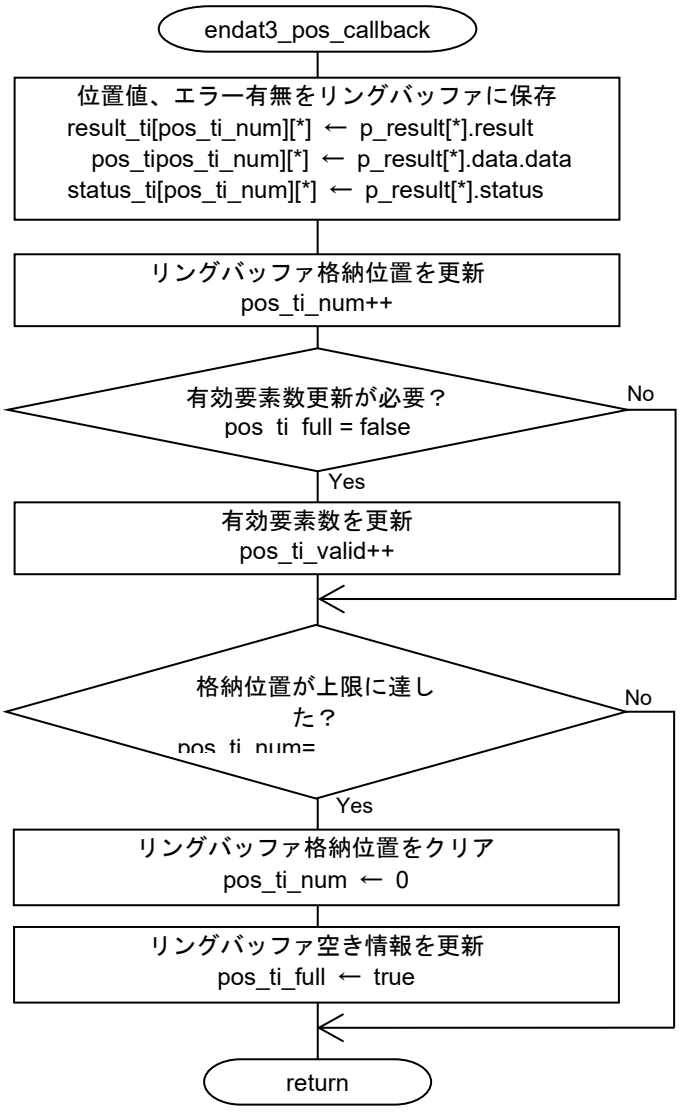

図 4-15 endat3_pos_callback 関数のフローチャート

(14) endat3_ready_callback フローチャート

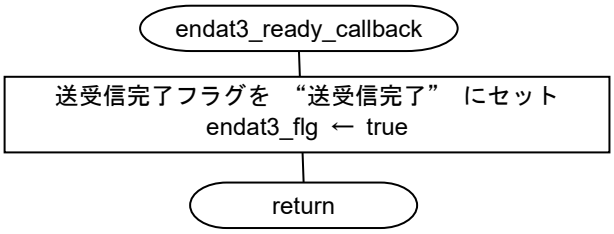

図 4-16 endat3_ready_callback 関数のフローチャート

4.11.7 動作シーケンス

(1) 起動シーケンス

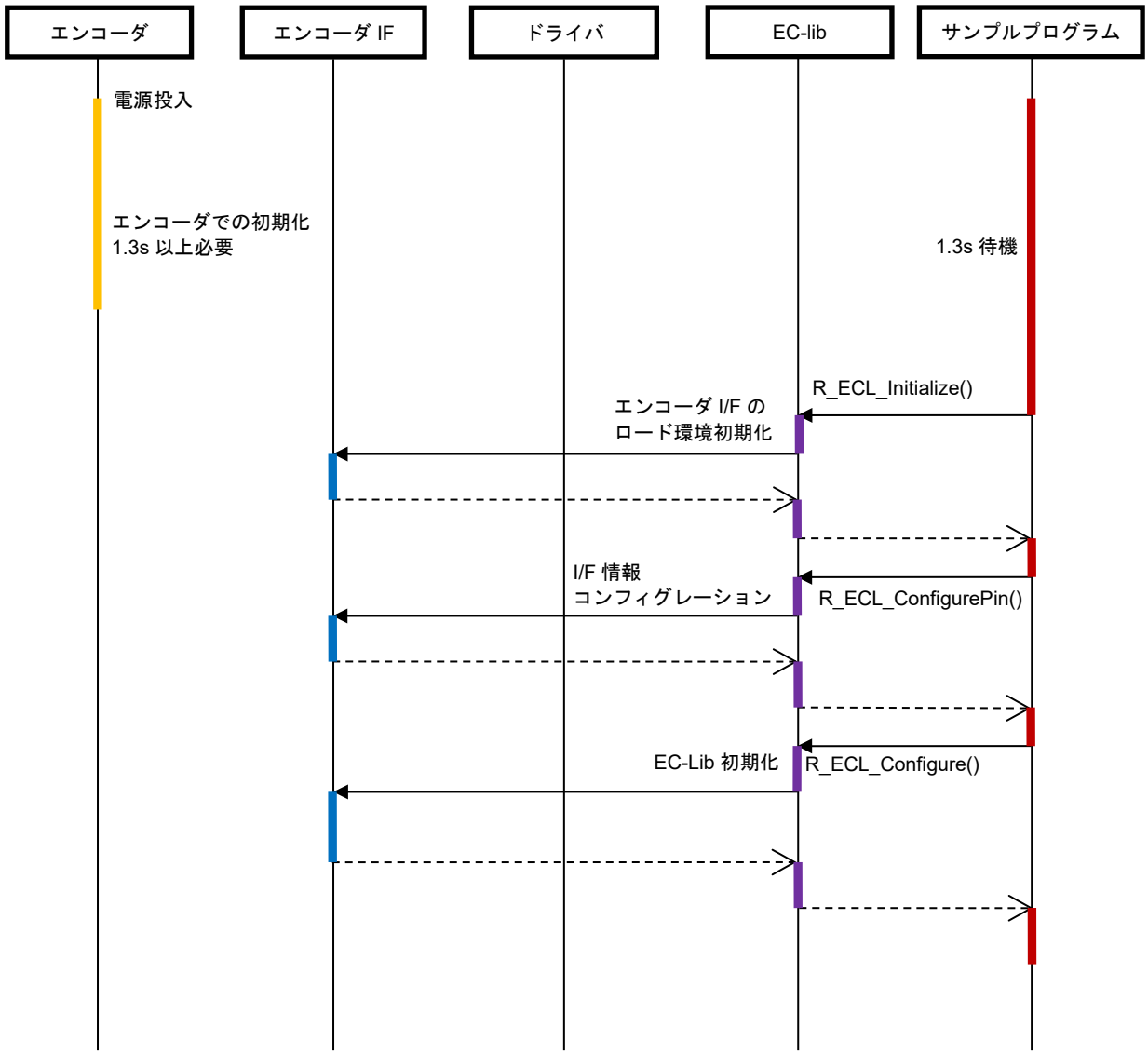

図 4-17 起動シーケンス図

(2) 開始シーケンス

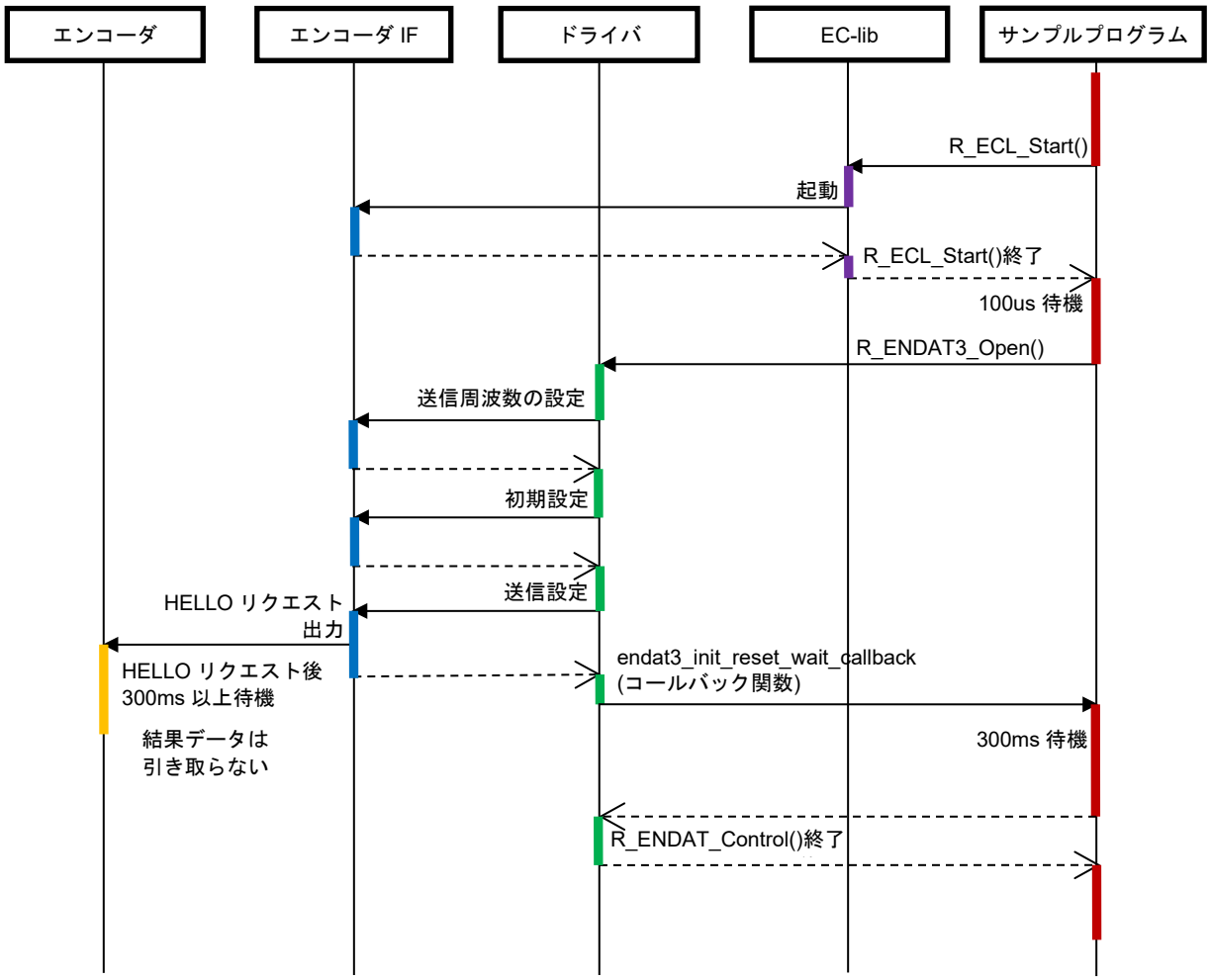

図 4-18 開始シーケンス図

(3) リクエスト送信とデータ受信のシーケンス

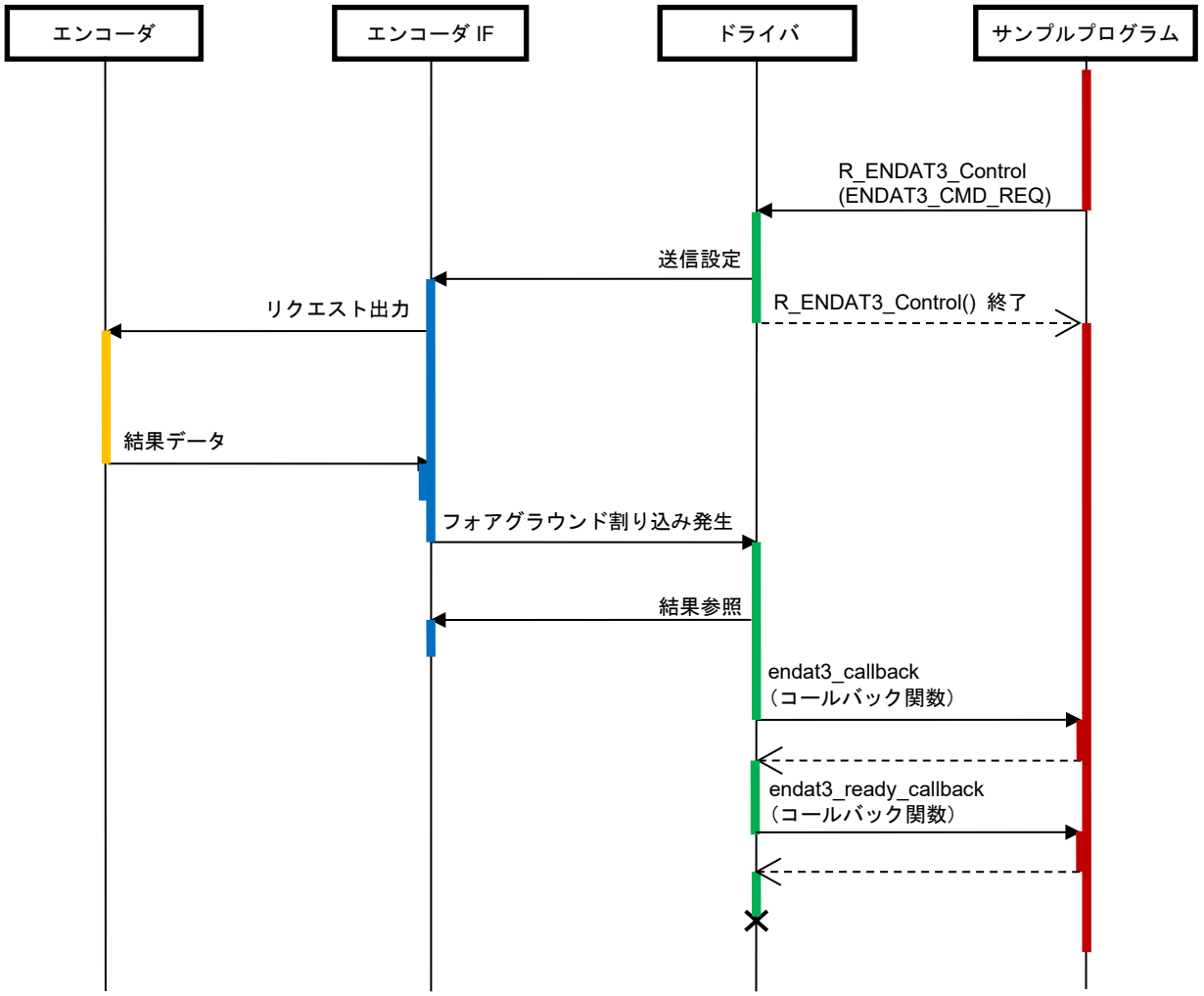

図 4-19 リクエスト送信とデータ受信のシーケンス図

(4)　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

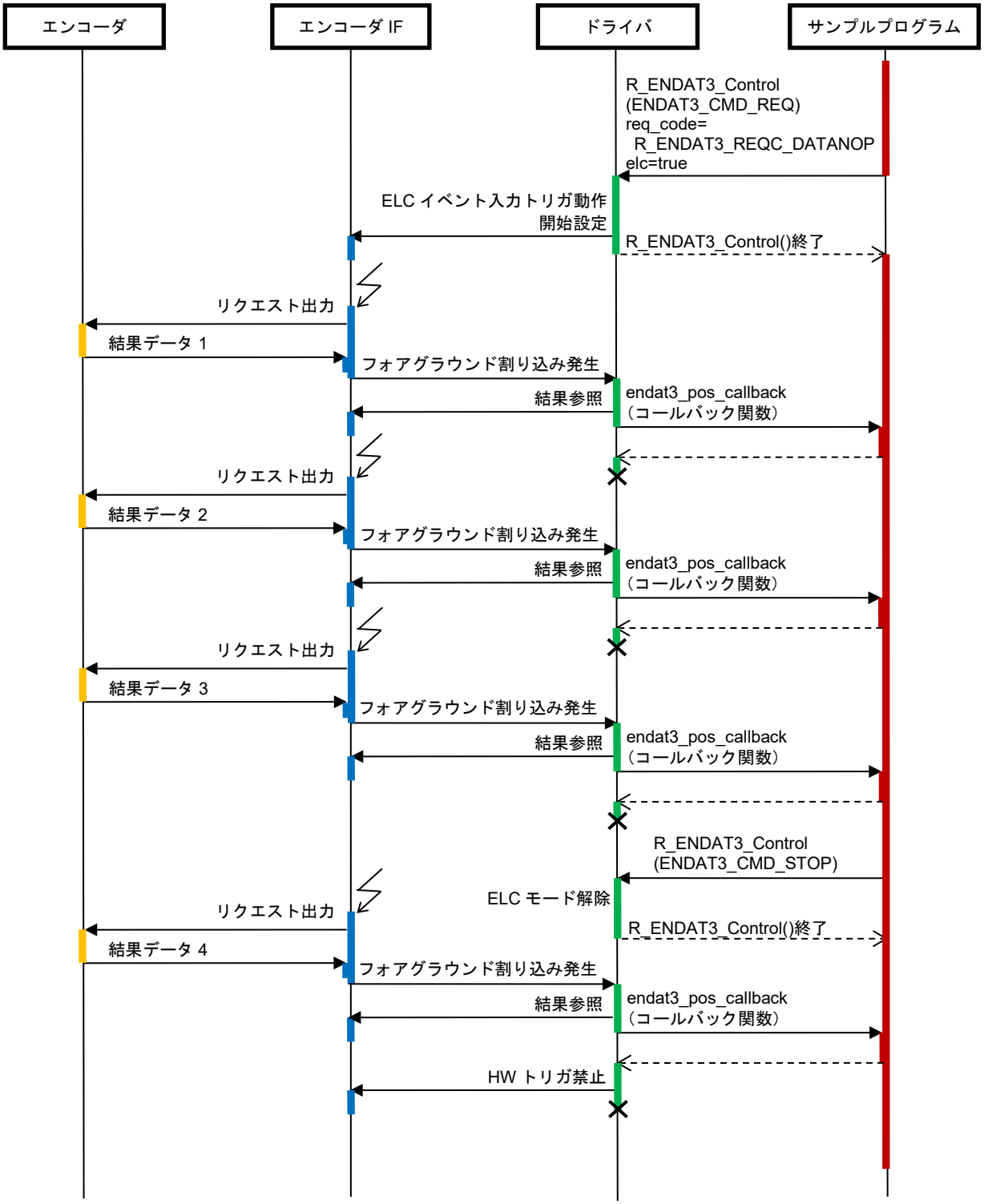

図 4-20　リクエスト送信(ELC モード)とデータの連続受信シーケンス図

(5) バックグラウンドリクエスト送信とデータ受信のシーケンス

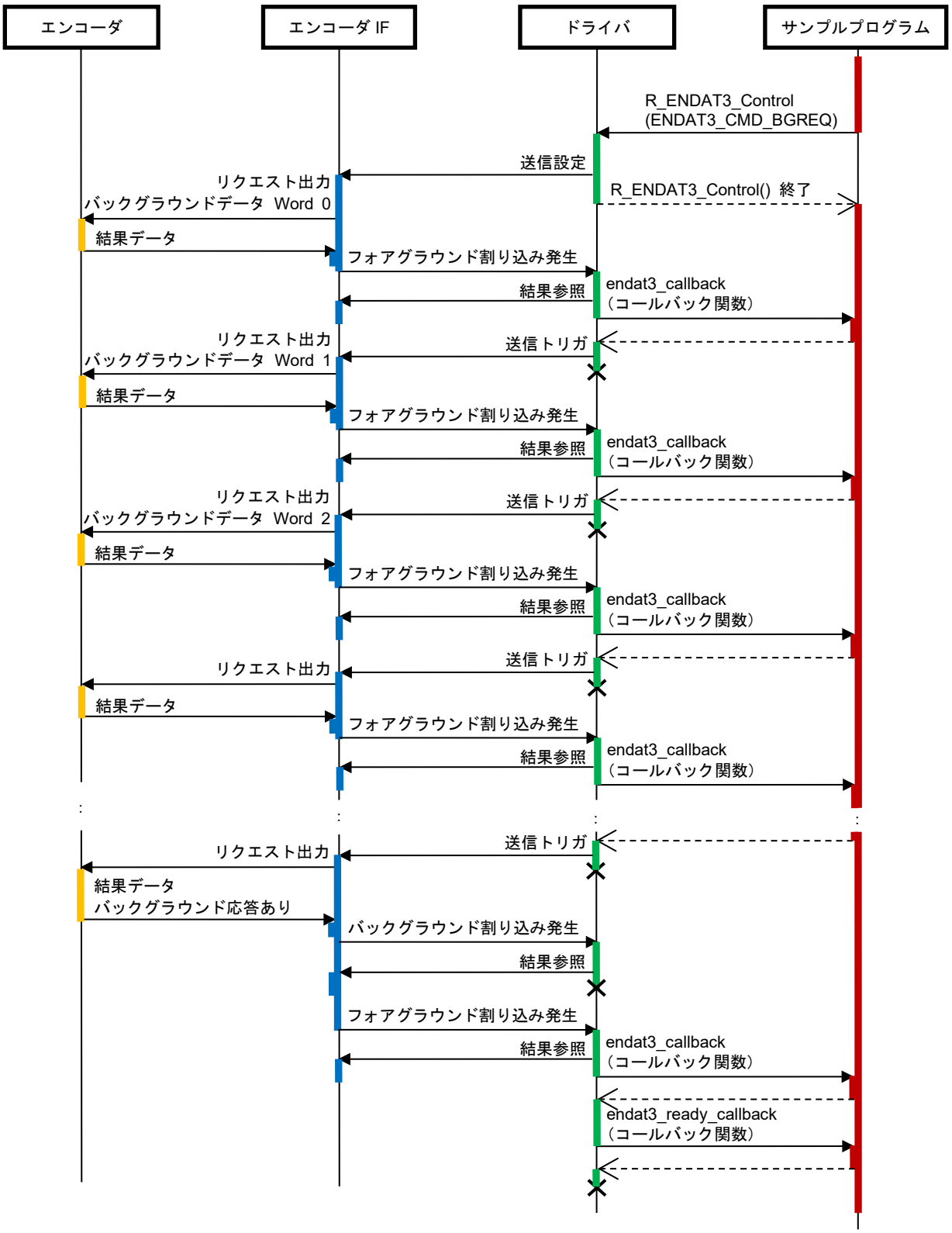

図 4-21 バックグラウンドリクエスト送信とデータ受信のシーケンス図

(6) 停止シーケンス

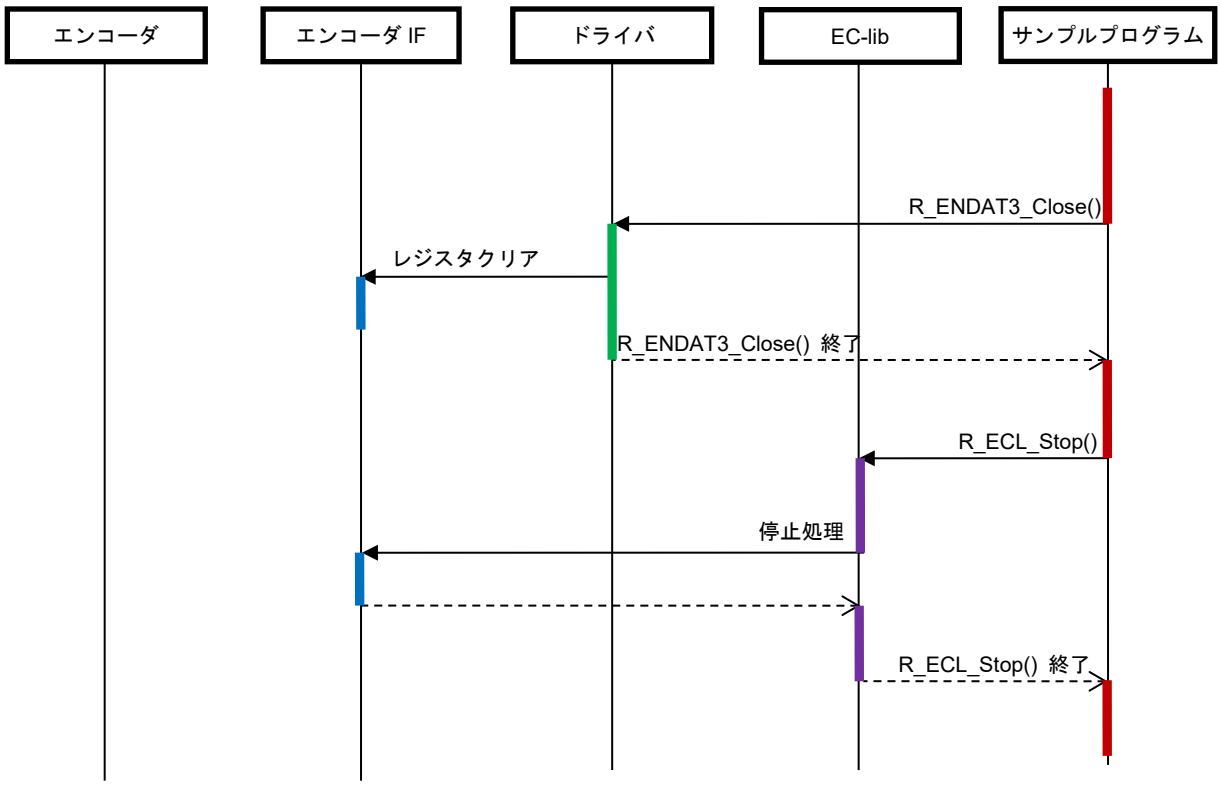

図 4-22 停止シーケンス図

4.11.8 コンソールコマンド

本サンプルプログラムは EnDat 3 準拠エンコーダ「ECI 1319 E30-R2」および、バス接続エンコーダ「EQN 1337 E30-RB」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。

表 4.11 コンソールコマンド一覧

コマンド	内容
pos *ba*	P2P モードのとき、位置値を 1 回だけ取得します。バスアドレス *ba* の指定は無効です。 バスモードのとき、*ba* で指定したバスアドレスのエンコーダから、位置値を 1 回だけ取得します。バスアドレスの指定を省略した場合は、ブロードキャストですべてのエンコーダからの位置値を 1 回取得します。
read *addr num*	エンコーダのメモリからデータを読み出します。読み出しアドレス *addr* は 16 進数で 000000 ～ FFFFFF の範囲で、読み出しワード数 *num* は 1 ～ 3 の範囲で指定します。ワード数の指定を省略した場合は、1 が使われます。
write *addr data*	エンコーダのメモリにデータを書き込みます。書き込みアドレス *addr* は 16 進数で 000000 ～ FFFFFF の範囲で、書き込みデータ *data* は 16 進数で 0000 ～ FFFF の範囲で指定します。
bus_addr *ba*	バスモードのとき read, write コマンドでアクセス対象となるエンコーダを設定します。バスアドレス *ba* は 16 進数で 0 ～ エンコーダ数の範囲で指定します。バスアドレスとして 0 が設定されている場合には、バスリクエストを使わずにアクセスを試みます。バスアドレスの初期値は 0 です。
rate *data*	エンコーダ I/F のデータレートを設定します。データレート *data* は 0 または 1 から選択します。0 のとき 12.5 Mbps に、1 のとき 25 Mbps に切り替えられます。エンコーダ本体のデータレートは、事前にフォアグラウンドリクエストで切り替えてください。データレートの初期値は 12.5 Mbps です。
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。バスモードのときは、ブロードキャストですべてのエンコーダからの位置値を連続取得します。タイマ周期 *val* は 10 進数で us 単位で指定します。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。
stop	位置値の連続取得を停止します。
req *name data*	フォアグラウンドリクエストを送信します。リクエスト名 *name* およびリクエストデータ *data* は、「表 4.12 フォアグラウンドリクエスト名 / リクエストデーター覧」を参照してください。リクエスト名はテキストで、リクエストデータは 16 進数で指定します。リクエストデータの指定を省略した場合は、0000 が使われます。
bus_req *ba name data*	バスモードでフォアグラウンドリクエストを送信します。バスアドレス *ba* は 16 進数で 0 ～ エンコーダ数 または FF の中から指定します。FF を指定した場合には、ブロードキャストが行われます。リクエスト名 *name* およびリクエストデータ *data* は、「表 4.12 フォアグラウンドリクエスト名 / リクエストデーター覧」を参照してください。リクエスト名はテキストで、リクエストデータは 16 進数で指定します。リクエストデータの指定を省略した場合は、0000 が使われます。
exit	プログラムを終了します。

表 4.12 フォアグラウンドリクエスト名 / リクエストデーター覧

リクエスト名	リクエストデータ	内容
DATA0	BGD	Low Priority Frame (LPF) 送信リスト 0 を選択する
DATA1	BGD	Low Priority Frame (LPF) 送信リスト 1 を選択する
DATA2	BGD	Low Priority Frame (LPF) 送信リスト 2 を選択する
DATA3	BGD	Low Priority Frame (LPF) 送信リスト 3 を選択する
DATA4	BGD	Low Priority Frame (LPF) 送信リスト 4 を選択する
DATA5	BGD	Low Priority Frame (LPF) 送信リスト 5 を選択する
DATA6	BGD	Low Priority Frame (LPF) 送信リスト 6 を選択する
DATA7	BGD	Low Priority Frame (LPF) 送信リスト 7 を選択する
DATA	BGD	送信リストを変更しない
DATANOP	0x0000	送信リストを変更せず、BGD も送信しない
RESET	0xBBBB	エンコーダリセットする
CLEAR	Type	エンコーダの状態をリセットする Type bit 0 : エラーフラグをリセット, Type bit 1 : 警告フラグをリセット Type bit 2 : 絶対位置をリセット Type bit 15～3 : 予約 (これらのビットには 0 を設定する)
ECHO	Data	遅延時間測定用にリクエストをエコーする
RATE	Type	データレートを切り替える Type = 0x0000 : 12.5 Mbps に切り替える Type = 0x0001 : 25 Mbps に切り替える (上記以外のリクエストデータは予約済みで、設定不可)
HELLO	0x2222	エンコーダの応答確認用

【注】 BGD はバックグラウンドデータです。詳細は HEIDENHAIN 社に問い合わせることで入手可能な「EnDat 3 Interface Specification」を参照してください。

(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。”endat3 >”に続けてコマンドを入力してください。

```
EnDat3 sample program start
R_ENDAT3_GetVersion = 4.0

endat3 >
```

(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、送受信結果、受信した位置値およびステータス情報が表示されます。

```
endat3 >pos
pos command
  result     : ENDAT3_REQ_SUCCESS
  pos mt     : 0x0D13
  pos st     : 0xDA50DE80
  rm        : true
  hpfv      : true
endat3 >
```

4.11.9 バス接続エンコーダ初期化手順

バス接続に対応した EnDat 3 エンコーダでは、エンコーダをデイジーチェインによって連結することで、バスモードによるアクセスが行えます。

バス接続した 3 台のエンコーダ (バスアドレスは RZ/T2M に遠い方から昇順に割り当てる) にアクセスするためには、最初に以下のコマンドを実行してください。

```
req HELLO 2222
bus_req 3 HELLO 2222
req HELLO 2222
bus_req 2 HELLO 2222
req HELLO 2222
bus_req 1 HELLO 2222
```

これ以降、各エンコーダがバスモードでのコマンドを受け付けるようになります。バス接続したエンコーダとの通信に関する詳細は、HEIDENHAIN 社に問い合わせることで入手可能な「EnDat 3 Interface Specification」および、「EnDat 3 Application Notes」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
1.00	Nov 15.24	-	初版発行
3.00	Oct 24.25	1, 4, 5	商標の説明の記載方法を更新
4.00	Mar 13.26	8 - 35 13, 14	ポインタ変数のプレフィクスを”p_”に変更 「4.5 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。