RZ/T2M グループ

EnDat Safety 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 2.2 仕様に準拠したエンコーダから Safety 仕様のデータを含む情報を取得・表示するサンプルプログラムについて説明します。

プログラムの特長を以下に示します。
・EnDat 2.2 のモードコマンド、MRS コードに対応
・EnDat 2.2 仕様に準拠した機能安全対応エンコーダ(HEIDENHAIN 社製 ECN1123 FS)から、角度情報等を取得

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 I/F	EnDat 2.2 (インクリメンタル信号には未対応) 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 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 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

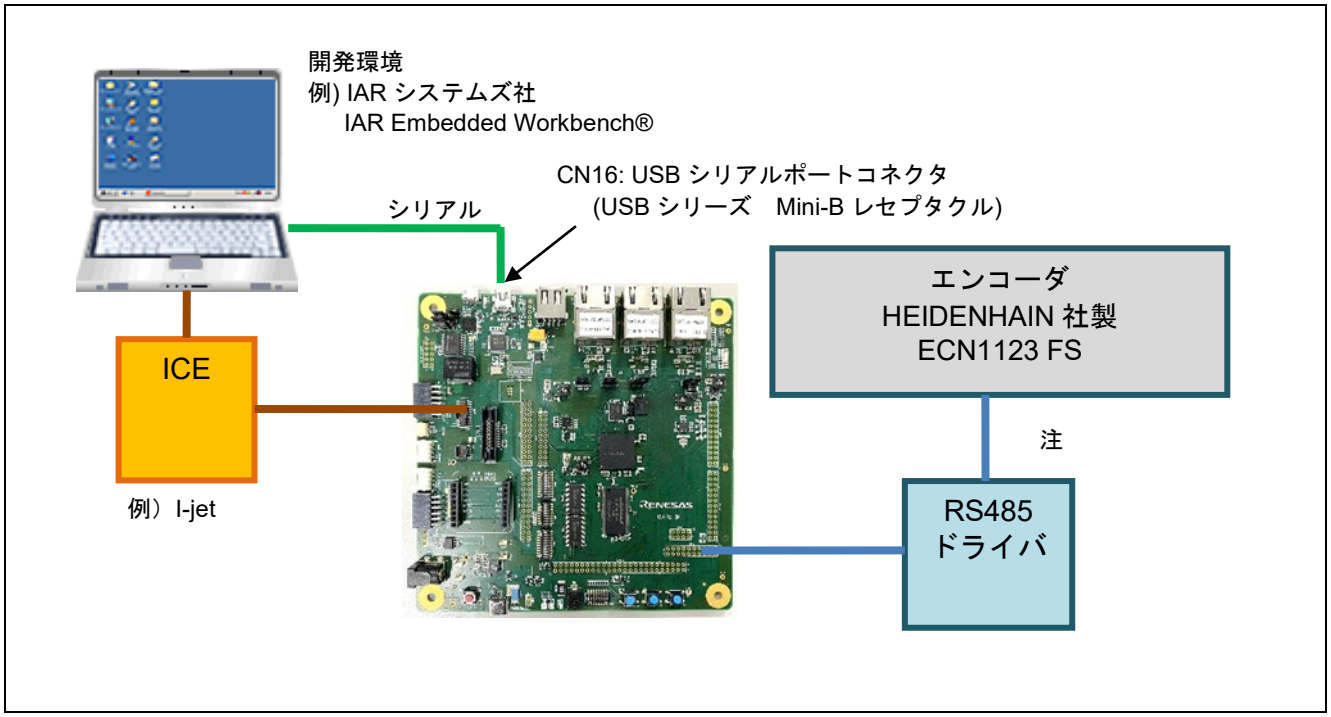

図1-1　動作環境

【注】　送受信可能なケーブル長は、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。

IAR Embedded Workbench は IAR システムズ社の登録商標です。

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
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 e^2 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 Safety 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―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
ENDAT_CH0	TCLK0 (ENCIF4)	P02_3	出力	クロック出力端子
	DE0 (ENCIF3)	P02_2	出力	データ出力許可設定端子
	DATA_DV0 (ENCIF2)	P02_0	出力	データ出力端子
	DATA_RC0 (ENCIF0)	P01_6	入力	データ入力端子
ENDAT_CH1	TCLK1 (ENCIF9)	P03_3	出力	クロック出力端子
	DE1 (ENCIF8)	P03_0	出力	データ出力許可設定端子
	DATA_DV1 (ENCIF7)	P17_5	出力	データ出力端子
	DATA_RC1 (ENCIF5)	P17_3	入力	データ入力端子

4. ソフトウェア説明

4.1 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) ノイズフィルタの設定
   B) 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)
   C) 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。

4.2 ファイル構成

ファイル構成は、RZ/T2M グループ Encoder I/F EnDat Safety 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 ドライバ API 関数	R_ENDAT_Open	8
	R_ENDAT_Close	8
	R_ENDAT_GetVersion	9
	R_ENDAT_Control	9
ユーザー定義関数	enc_init_reset_wait_callback	12
	enc_init_mem_wait_callback	12
	enc_init_pram_wait_callback	12
	enc_init_cable_wait_callback	13
	endat_callback	13
	endat_poscon_callback	14
	endat_fifodt_callback	14
	endat_rdst_callback	14
割り込みハンドラ	endat0_rx_int_isr	15
	endat1_rx_int_isr	15
	endat0_fifo_int_isr	15
	endat1_fifo_int_isr	15

4.4　API 関数仕様

4.4.1　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　r_endat_rzt2_dat,h
宣　言	r_endat_err_t R_ENDAT_Open(const int32_t id, r_endat_info_t* pinfo);
説　明	EnDat ドライバは下記の初期設定を行います。 1.　ノイズフィルタの初期化 2.　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) 3.　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_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 pinfo　：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, pinfo に指定した e_endat_info_t 構造体のメンバ変数が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)
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。 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。

4.4.2　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_rz2_if.h　r_endat_rzt2_dat.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_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	ENDAT_SUCCESS　：正常終了 ENDAT_ERR_INVALID_ARG　：異常終了 (指定した id が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)

4.4.3 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_rz2_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

4.4.4 R_ENDAT_Control

R_ENDAT_Control	
概　要	EnDat エンコーダの制御
ヘッダ	r_endat_rz2_if.h　r_endat_rzt2_dat.h
宣　言	r_endat_err_t R_ENDAT_Control(const int32_t id, const r_endat_cmd_t cmd, void *const pbuf);
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「4.4.5 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_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_endat_cmd_t」参照。 pbuf　：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_ENDAT_Open を実行してください。

4.4.5　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_rz2_if.h　r_endat_rz2_dat.h	
宣　言	r_endat_err_t R_ENDAT_Control(const int32_t id, const r_endat_cmd_t cmd, void *const pbuf);	
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_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	：ENDAT_CMD_REQ
	pbuf	：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「4.10.1(1) r_endat_info_t」参照。
リターン値	ENDAT_SUCCESS	：正常終了
	ENDAT_ERR_INVALID_ARG	：異常終了 (id, cmd が規定されていない値、pbuf が NULL、pbuf 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 STAT レジスタ READY 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)

(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_rz2_if.h　r_endat_rz2_dat.h	
宣　言	r_endat_err_t R_ENDAT_Control(const int32_t id, const r_endat_cmd_t cmd, void *const pbuf);	
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_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	：ENDAT_CMD_POS_STOP
	pbuf	：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)

4.5 ユーザー定義関数仕様

4.5.1 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	r_endat_rzt2_if.h
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

4.5.2 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	r_endat_rzt2_if.h
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

4.5.3 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	r_endat_rzt2_if.h
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

4.5.4　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	r_endat_rzt2_if.h
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

4.5.5　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	r_endat_rzt2_if.h	
宣　言	void endat_callback(r_endat_result_t * presult, r_endat_protocol_err_t *perr);	
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。MBERR 割り込み、WDG 割り込み、RXEND 割り込み発生時にコールされます。 本関数は割り込みハンドラのコンテキストとなります。割り込みの応答性を確保するため、速やかに 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	presult	：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。
	perr	：エラー情報 送受信結果を格納した構造体 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	

4.5.6 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	r_endat_rzt2_if.h
宣　言	void endat_poscon_callback(r_endat_result_t * presult, endat_protocol_err_t *perr);
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。MBERR 割り込み、WDG 割り込み、RXEND 割り込み発生時にコールされます。 本関数は割り込みハンドラのコンテキストとなります。割り込みの応答性を確保するため、速やかに 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	presult : 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。 perr : エラー情報 送受信結果を格納した構造体 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

4.5.7 endat_fifodt_callback

endat_fifodt_callback	
概　要	FIFO データ受信結果通知関数
ヘッダ	r_endat_rzt2_if.h
宣　言	void endat_fifodt_callback(r_endat_fifodt_t *pfdat);
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。FIFO データ受信時にコールされます。 本関数は割り込みハンドラのコンテキストとなります。割り込みの応答性を確保するため、速やかに 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	pfdat : 受信結果 FIFO による受信結果を格納した構造体 r_endat_fifodt_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

4.5.8 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	r_endat_rzt2_if.h
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。 RDSTC 割り込み発生時に 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

4.6 割り込みハンドラ

4.6.1 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。 1. MBERR 割り込み 2. WDG 割り込み 3. RXEND 割り込み 4. RDSTC 割り込み
引　数	なし
リターン値	なし

4.6.2 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. MBERR 割り込み 2. WDG 割り込み 3. RXEND 割り込み 4. RDSTC 割り込み
引　数	なし
リターン値	なし

4.6.3 endat0_fifo_int_isr

endat0_fifo_int_isr	
概　要	チャネル 0 FIFO データ受信完了割り込みハンドラ
ヘッダ	-
宣　言	void endat0_fifo_int_isr(void);
説　明	EnDat チャネル 0 の FIFO 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

4.6.4 endat1_fifo_int_isr

endat1_fifo_int_isr	
概　要	チャネル 1 FIFO データ受信完了割り込みハンドラ
ヘッダ	-
宣　言	void endat1_fifo_int_isr(void);
説　明	EnDat チャネル 1 の FIFO 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ドライバで使用する割り込みを示します。

表 4.2 EnDat ドライバで使用する割り込み

割り込み	ID	概要
チャネル 0 データ受信	372	下記の割り込み要因で割り込みが発生します 1. MBERR 割り込み 2. WDG 割り込み 3. RXEND 割り込み 4. RDSTC 割り込み
チャネル 1 データ受信	376	下記の割り込み要因で割り込みが発生します 1. MBERR 割り込み 2. WDG 割り込み 3. RXEND 割り込み 4. RDSTC 割り込み
チャネル 0 FIFO データ受信	374	FIFO への受信データ格納完了で割り込みが発生します
チャネル 1 FIFO データ受信	378	FIFO への受信データ格納完了で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 4.4にEnDat 2.2モードコマンド、表 4.5に送信クロック周波数、表 4.6にWatchdog Timerの時間の単位、表 4.7にデータ送信開始時のデータLow期間、表 4.8にMRSコード一覧、表 4.9にFIFOデータ受信用設定値を示します。エラーコードは「4.10.3(1)r_endat_err_t」を参照してください。

表 4.3 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 4.4 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 4.5 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 4.6 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 4.7 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 4.8 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。

表 4.9 FIFO データ受信用設定値

定数名	設定値	内容
R_ENDAT_FIFO_POS_PAETH	1u	POS コマンド用 FIFO 受信閾値
R_ENDAT_FIFO_POS_NORM	2u	POS コマンド用 FIFO データ読み出しワード数
R_ENDAT_FIFO_PAETH	3u	FIFO 受信閾値 (POS コマンド以外)
R_ENDAT_FIFO_NORM	4u	FIFO データ読み出しワード数 (POS コマンド以外)
R_ENDAT_FIFO_CLR	22u	FIFO データクリア時の読み出し回数

4.9 固定幅整数一覧

表 4.10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 4.10 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_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 4.5　送信クロック周波数」参照
                                        本設定は CFG1 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)
                                        本設定は NF レジスタの INF ビット, NFINF ビット,
                                        NFSCNT ビットに反映されます。
    bool            delay_comp;         伝送遅延補正 (true: 有効, false: 無効)
                                        本設定は CFG1 レジスタの DLY 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 4.7　データ送信開始時のデータLow期間」参照
                                        本設定は CFG2 レジスタの TST ビットに反映されます。
    endat_wait_cb_t penc_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「4.5.1 enc_init_reset_wait_callback」参照
                                        NULL は設定しないでください。
    endat_wait_cb_t penc_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「4.5.2 enc_init_mem_wait_callback」参照
                                        NULL は設定しないでください。
    endat_wait_cb_t penc_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「4.5.3 enc_init_pram_wait_callback」参照
                                        NULL は設定しないでください。
    endat_wait_cb_t penc_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「4.5.4 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 4.6 Watchdog Timerの時間の単位」参照
    uint8_t     time;       Watchdog Timer の時間を設定
                            「表 4.11 Watchdog Timer時間対応表」参照
} r_endat_watchdog_t
```

表 4.11 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 4.12 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 4.4 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 4.8 MRSコード一覧」参照
                                            「表 4.12 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 4.12 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 4.12 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
                                            「4.10.1(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 CFG2 レジスタ WDG 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。
    bool                fifo_enable;        FIFO 受信設定 (true: 有効, false: 無効)
    bool                rtcnt_enable;       リカバリータイム計測設定 (true: 有効, false: 無効)
    r_endat_isr_result_cb_t pisr_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「4.5.5 endat_callback」および
                                            「4.5.6 endat_poscon_callback」参照
                                            NULL は設定しないでください。
    r_endat_isr_fifodt_cb_t pisr_fifodt;    FIFO 受信データを通知するコールバック関数へのポインタ
                                            詳細は「4.5.7 endat_fifodt_callback」参照
    r_endat_isr_rdst_cb_t   pisr_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「4.5.8 endat_rdst_callback」参照
                                            NULL は設定しないでください。
} r_endat_req_t
```

表 4.12　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
                                        「4.10.3(3) r_endat_req_err_t」参照
    r_endat_data_t          data;       受信データ
                                        「4.10.1(5) r_endat_data_t」参照
    r_endat_status_t        status;     エンコーダのステータス
                                        「4.10.1(7) 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 RECV1L レジスタの RXD1 ビット、上位
                                        32bit に RECV1U レジスタの RXD1 ビットの値が格納
                                        されます。
    uint32_t                add_datum1; 付加情報 1
                                        RECV3 レジスタの RXD3 ビットの値が格納されます。
    uint32_t                add_datum2; 付加情報 2
                                        RECV2 レジスタの RXD2 ビットの値が格納されます。
} r_endat_data_t
```

(6) r_endat_fifodt_t

FIFO 受信データ

```
typedef struct
{
    uint8_t     mode_cmd;                   EnDat 2.2 モードコマンド
                                            「表 4.4 EnDat 2.2モードコマンド」参照
    uint16_t    rtcnt_pre;                  リカバリータイムカウント (実行前)
                                            モードコマンド実行直前の RTCNT レジスタの値が格納されます。
    uint16_t    rtcnt;                      リカバリータイムカウント (FIFO データ読み出し時)
                                            FIFO データ読み出し直後の RTCNT レジスタの値が格納されます。本カウント値を格納した時点では、リカバリータイムが続いており、RTCNT レジスタのカウントアップが完了していない場合があります。カウントアップ完了後の値は、次の FIFO 受信データの、モードコマンド実行前カウント値 (rtcnt_pre) で確認してください。
                                            リクエスト情報で、リカバリータイム計測設定を無効 (rtcnt_enable を false) に設定した場合や、Continuous モード (dt を true) に設定した場合には、カウントアップは行われません。
    uint32_t    fdat [R_ENDAT_FIFO_NORM]; FIFO 受信データ配列
                                            FIFO に格納された受信データが読み出して格納されます。R_ENDAT_POS コマンドでは、位置情報が、その他の位置データ取得コマンドでは、位置情報とともに付加情報が格納されます。
} r_endat_fifodt_t
```

(7) 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
```

(8) 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        mrsadr;     EnDat TYPE2 エラー内のアドレスエラー (true: 発生, false: 未発生)
    bool        fifoerr;    FIFO オーバーフローエラー (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(true: 発生, false: 未発生)
} r_endat_protocol_err_t
```

4.10.2 共用体

使用しません。

4.10.3 列挙型

(1) r_endat_err_t

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
{
    R_FAC_CMD_REQ          ,         エンコーダへコマンド送信
    R_FAC_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) r_fac_e2prom_dir_t

E2PROM の方向。

```
typedef enum
{
    R_FAC_E2PROM_READ    ,           読み出し
    R_FAC_E2PROM_WRITE               書き込み
} r_fac_e2prom_dir_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 2.2 に準拠した機能安全対応エンコーダ 「ECN1123 FS」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 Safety エンコーダ(ECN1123)へ送信
2) EnDat Safety エンコーダ(ECN1123)から受信したデータを表示
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) 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。

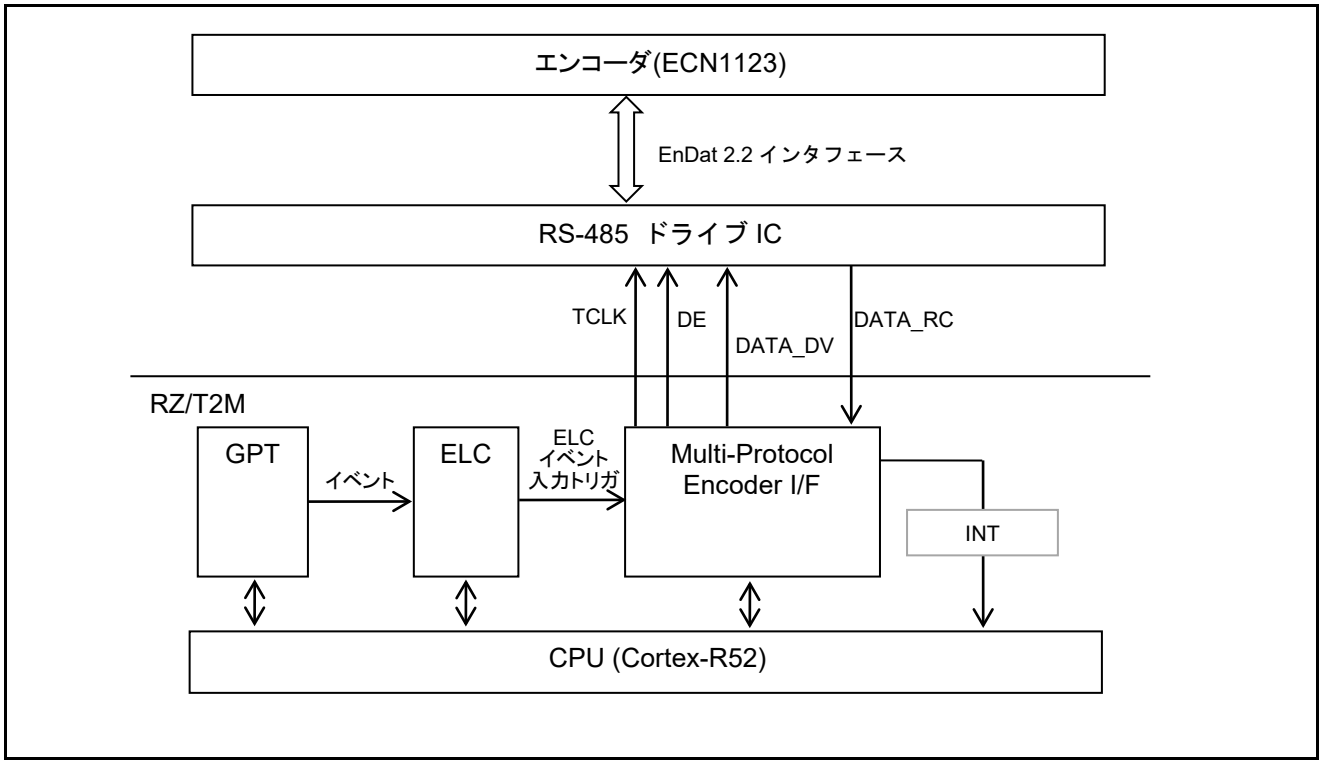

図 4-1 システムブロック図

(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 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、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。

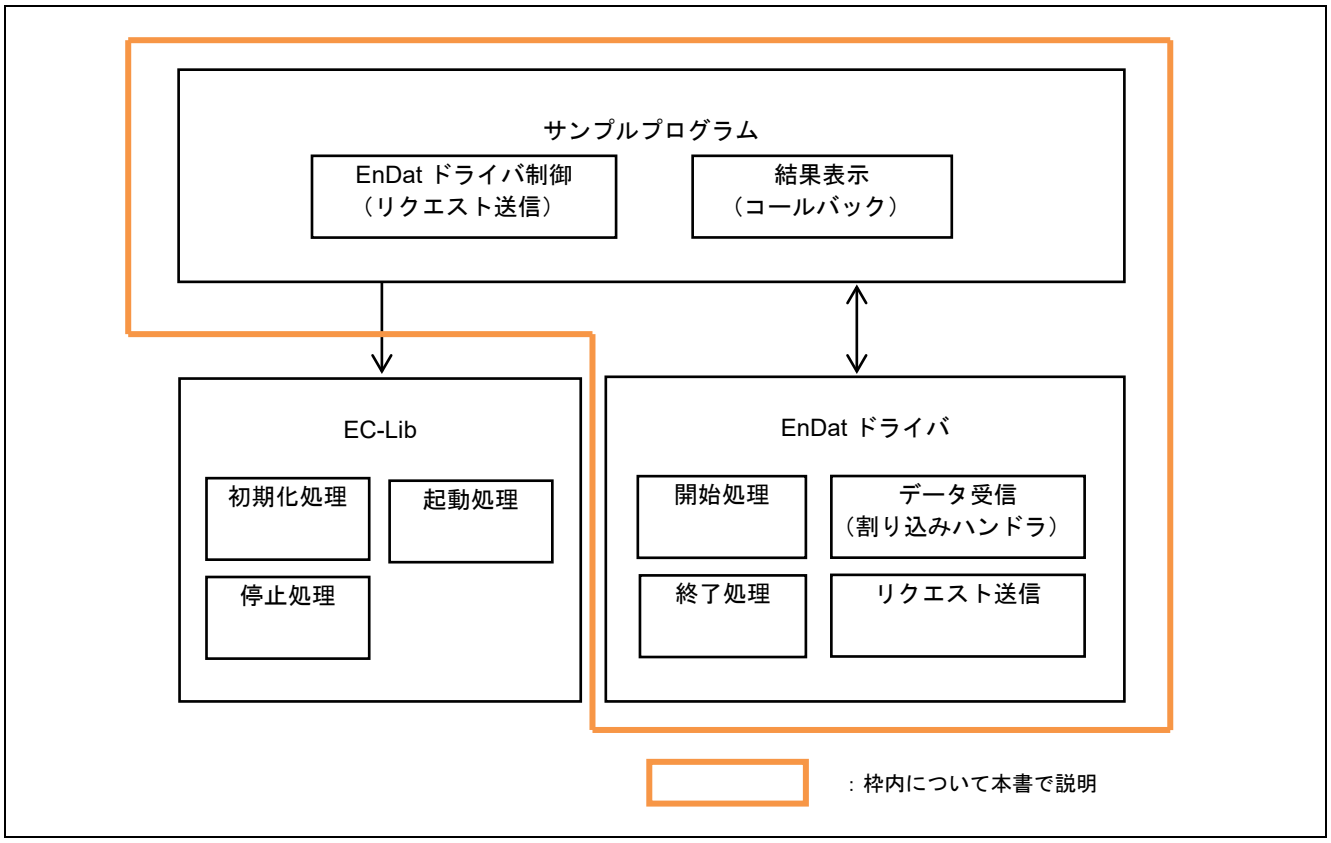

図 4-2　ソフトウェア構成図

4.11.2 サンプルプログラム関数一覧

表 4.13に主要なサンプルプログラム関数一覧を示します

表 4.13　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	31	-
enc_main	31	41
endat_cmd_control	31	42
endat_power_on_wait	31	-
enc_init_reset_wait_callback	32	-
enc_init_mem_wait_callback	32	-
enc_init_pram_wait_callback	32	-
enc_init_cable_wait_callback	32	-
endat_pos	33	43
endat_poscon	33	44
endat_elctimer	33	45
endat_stop	34	46
endat_temp	34	47
endat_read	34	48
endat_write	35	49
endat_spos	35	50
endat_pos_safe	35	51
endat_sel_info	36	52
endat_callback	36	53
endat_poscon_callback	36	54
endat_fifodt_callback	36	54
endat_rdst_callback	37	55
get_cmd	37	-
cmd_exit	37	-
result_display	37	-
result_display_param	38	-
result_fifo_display	38	-
timer_start	38	-
timer_stop	38	-

4.11.3 サンプルプログラム関数仕様

(1) hal_entry

hal_entry	
概　要	EnDat 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 サンプルプログラムのエントリー関数です。ここから、メイン関数 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 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 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 I/F ドライバのエラーコード）

(3) endat_cmd_control

endat_cmd_control	
概　要	EnDat ドライバ制御関数
ヘッダ	-
宣　言	static void endat_cmd_control(void);
説　明	以下の処理を行います。 • EnDat 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 エンコーダ制御の終了
引　数	なし
リターン値	なし

(4) 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

(5) 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「4.5.1 enc_init_reset_wait_callback」参照。
引　数	なし
リターン値	なし

(6) 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「4.5.2 enc_init_mem_wait_callback」参照
引　数	なし
リターン値	なし

(7) 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「4.5.3 enc_init_pram_wait_callback」参照
引　数	なし
リターン値	なし

(8) 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「4.5.4 enc_init_cable_wait_callback」参照
引　数	なし
リターン値	なし

(9) 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

(10) 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

(11) 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

(12) 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

(13) 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

(14) endat_read

endat_read	
概　要	エンコーダからパラメータを取得する関数
ヘッダ	-
宣　言	static void endat_read(uint32_t arg_num, char_t *parg[]);
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。エンコーダの指定アドレスからパラメータを読み出します
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

(15) endat_write

endat_write	
概　要	エンコーダにパラメータを書き込む関数
ヘッダ	-
宣　言	static void endat_write(uint32_t arg_num, char_t *parg[]);
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。エンコーダの指定アドレスにパラメータを書き込みます
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

(16) endat_spos

endat_spos	
概　要	エンコーダから Safety 仕様のデータとともに位置値を取得する関数
ヘッダ	-
宣　言	static void endat_spos(uint32_t arg_num, char_t *parg[]);
説　明	コンソールコマンド spos が入力された場合に実行される関数です。以下の処理を行います。 1. 付加情報 1 に Position 2 word 1 を選択 2. 位置値とともに付加情報を取得 (リカバリータイム計測) 3. 付加情報 1 に Position 2 word 2 を選択 4. 位置値とともに付加情報を取得 5. 付加情報 1 に Position 2 word 3 を選択 6. 位置値とともに付加情報を取得 7. 付加情報 1 に NOP (データなし) を選択 8. 位置値とともに付加情報を取得
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

(17) endat_pos_safe

endat_pos_safe	
概　要	エンコーダから位置値とともに付加情報を取得する関数
ヘッダ	-
宣　言	static void endat_pos_safe(uint32_t arg_num, char_t *parg[]);
説　明	コンソールコマンド pos_safe が入力された場合に実行される関数です。位置値とともに、選択されている付加情報を取得します。
引　数	arg_num　コンソールから入力された文字列の数 *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

(18) endat_sel_info

endat_sel_info	
概　要	エンコーダから取得する付加情報を選択する関数
ヘッダ	-
宣　言	static void endat_sel_info(uint32_t arg_num, char_t *parg[]);
説　明	コンソールコマンド sel_info が入力された場合に実行される関数です。付加情報 1 または付加情報 2 として取得する情報を選択します。
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

(19) 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 *presult, r_endat_protocol_err_t *perr);
説　明	結果をメモリに保存します。
引　数	presult　：送受信結果 perr　：EnDat I/F およびエンコーダのエラー情報
リターン値	なし

(20) 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 *presult, r_endat_protocol_err_t *perr);
説　明	連続して取得した結果をメモリに保存します。
引　数	presult　：送受信結果 perr　：EnDat I/F およびエンコーダのエラー情報
リターン値	なし

(21) endat_fifodt_callback

endat_fifodt_callback	
概　要	FIFO データ受信結果通知のコールバック関数
ヘッダ	-
宣　言	static void endat_fifodt_callback(r_endat_fifodt_t *pfdat);
説　明	取得した FIFO データのポインタを保存します。
引　数	pfdat　：取得した FIFO データ
リターン値	なし

(22) 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

(23) 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

(24) cmd_exit

cmd_exit	
概　要	EnDat 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。EnDat 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

(25) 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(r_endat_result_t *presult, r_endat_protocol_err_t *perr);
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	presult　　：送受信結果 perr　　：EnDat I/F およびエンコーダのエラー情報
リターン値	なし

(26) result_display_param

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

(27) result_fifo_display

result_fifo_display	
概　要	FIFO データ受信結果を表示する関数
ヘッダ	-
宣　言	static void result_fifo_display(r_endat_result_t *presult);
説　明	エンコーダへのリクエスト送受信が成功しているとき、受信 FIFO データを表示します。
引　数	presult　：FIFO データ受信結果
リターン値	なし

(28) 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

(29) 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 4.14に static 型変数を示します。const 型は使用しません。

表 4.14 static 型変数

型	変数名	内容	使用関数
bool	endat_flag	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_read endat_write endat_spos endat_pos_safe endat_sel_info 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	*pendat_result	データ取得結果を格納したアドレス	endat_pos endat_temp endat_read endat_write endat_spos endat_pos_safe endat_sel_info endat_callback
r_endat_protocol_err_t	*pendat_err	エラー情報を格納したアドレス	endat_pos endat_temp endat_read endat_write endat_spos endat_pos_safe endat_sel_info endat_callback
r_endat_fifodt_t	*pendat_result_fifo	FIFO データ取得結果を格納したアドレス	endat_fifodt_callback result_fifo_display
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

型	変数名	内容	使用関数
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	cur_id	EnDat I/F ドライバ 使用 ID	enc_main endat_cmd_control endat_pos endat_pocson endat_elctimer endat_stop endat_temp endat_read endat_write endat_spos endat_pos_safe endat_sel_info

4.11.5 サンプルプログラムの定数一覧

表 4.15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 4.15 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	EC-Lib 起動後の待機時間 (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

4.11.6 メイン処理のフローチャート

(1) enc_mainフローチャート

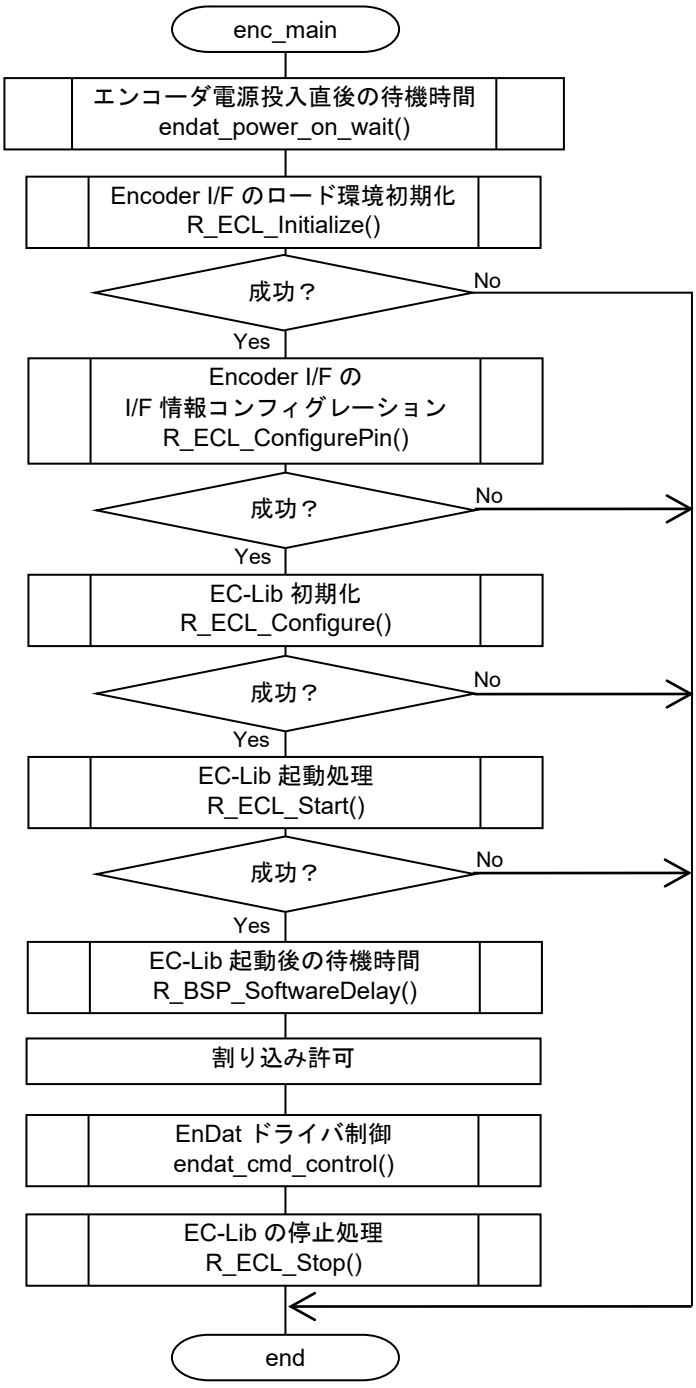

図 4-3 enc_main関数のフローチャート

(2) endat_cmd_control フローチャート

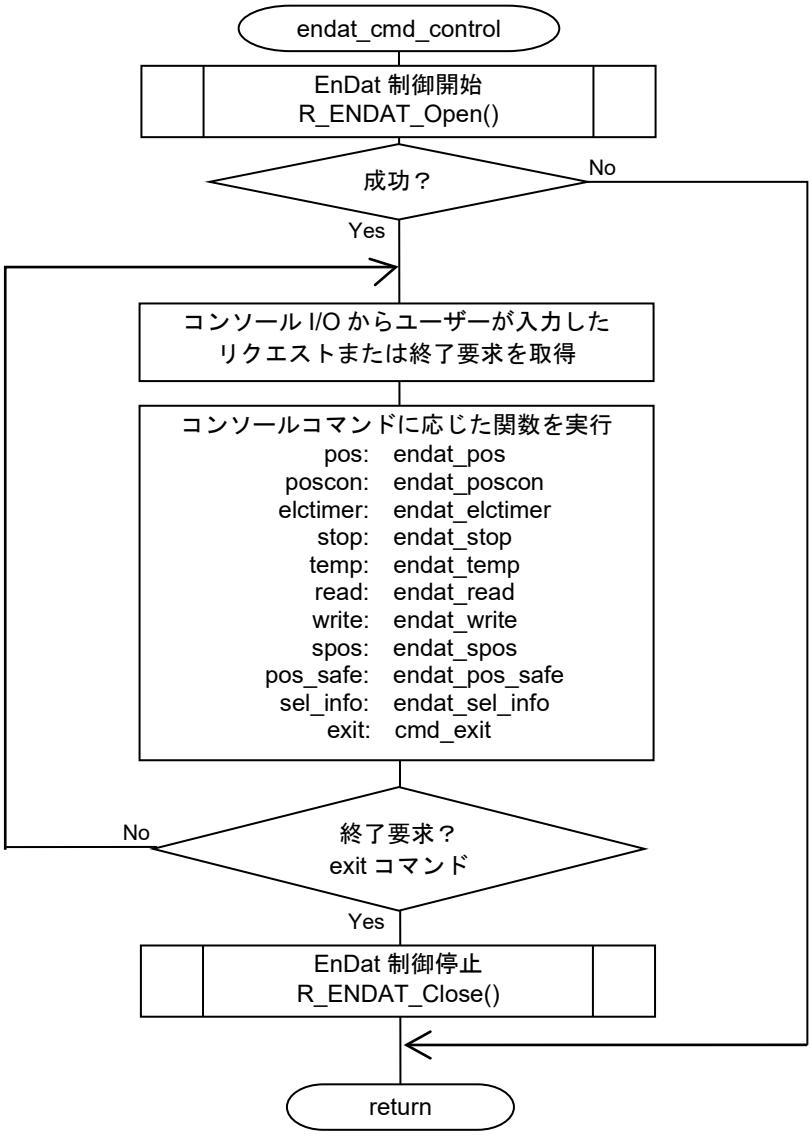

図 4-4 endat_cmd_control 関数のフローチャート

(3) endat_pos フローチャート

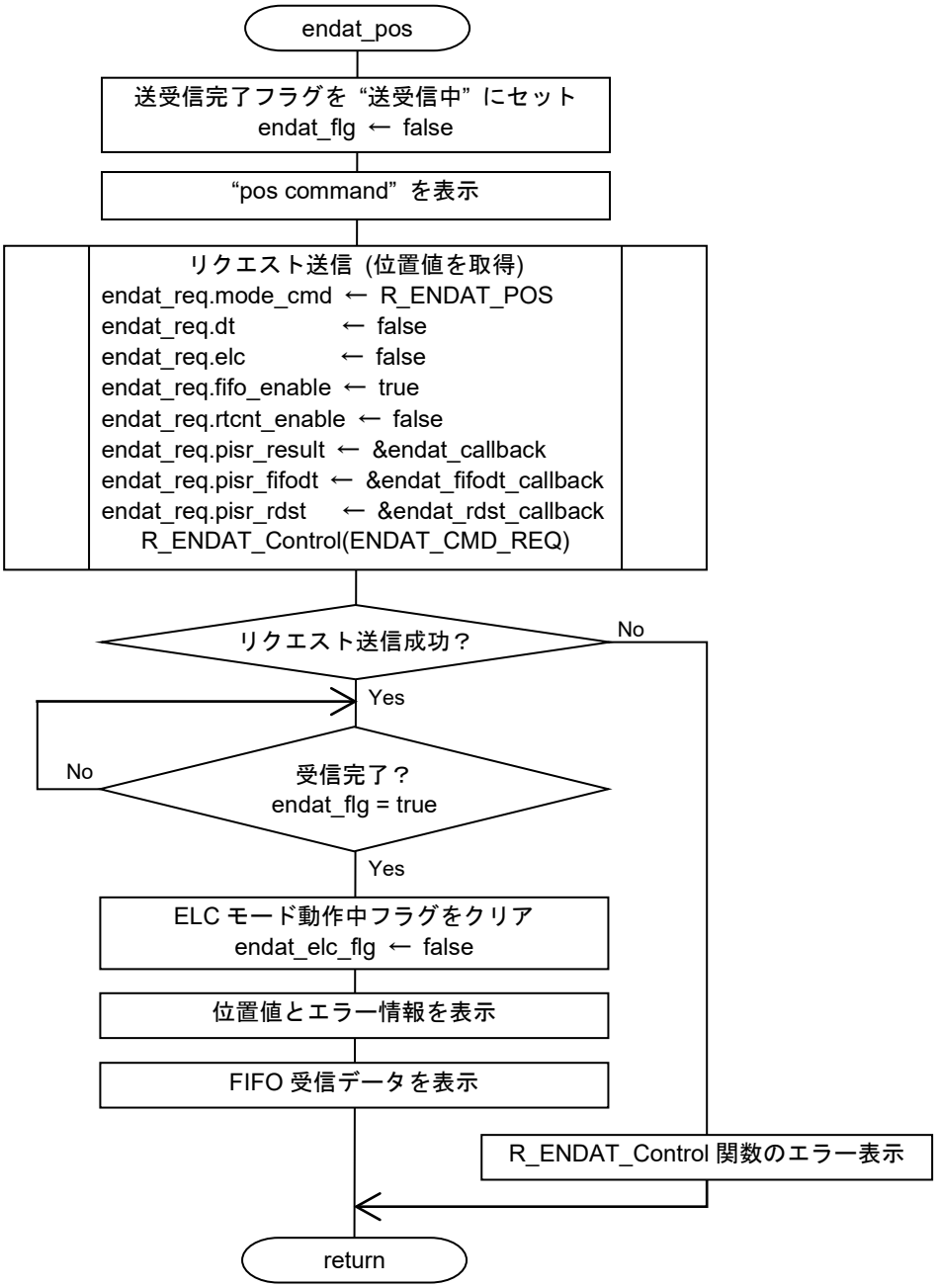

図 4-5 endat_pos 関数のフローチャート

(4) endat_poscon フローチャート

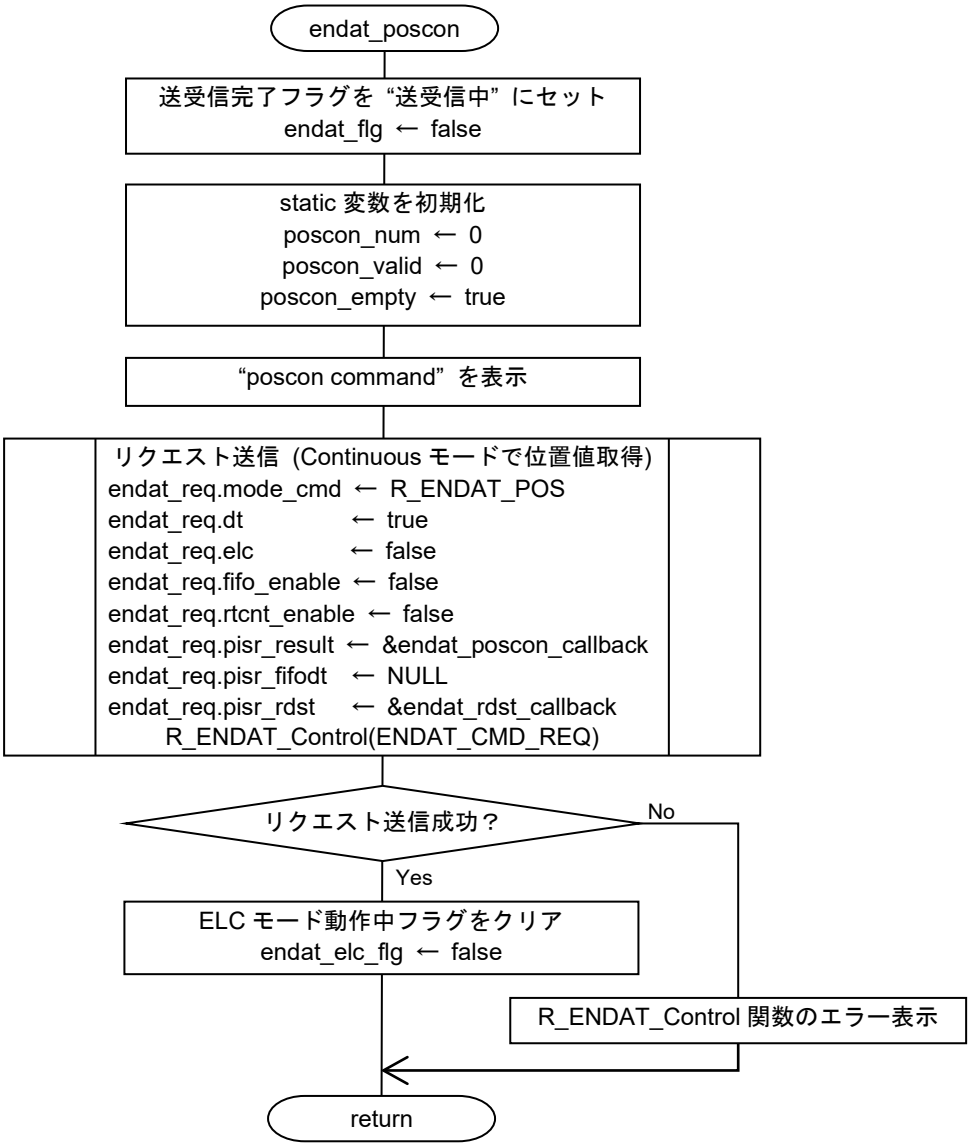

図 4-6 endat_poscon 関数のフローチャート

(5) endat_elctimer フローチャート

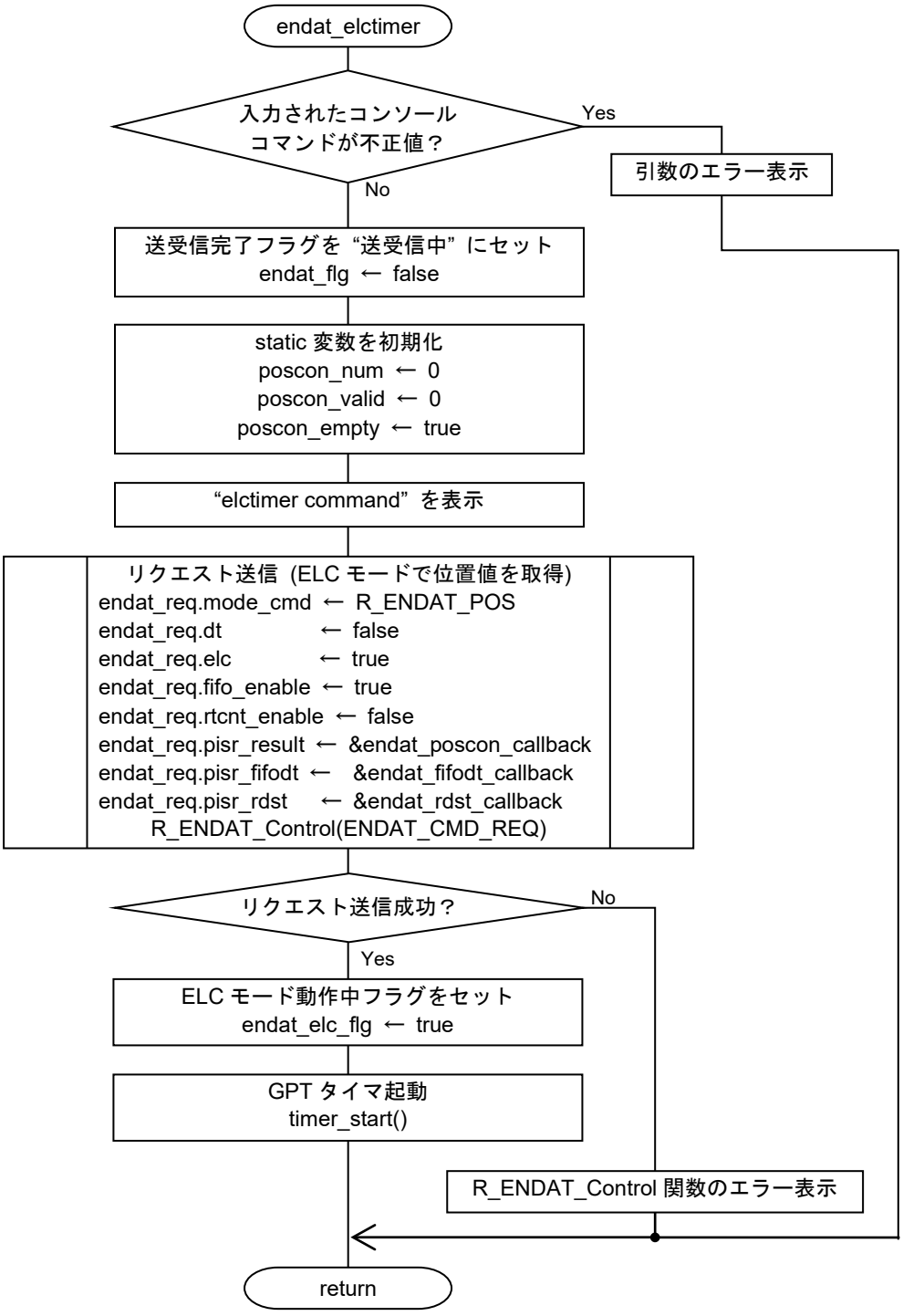

図 4-7 endat_elctimer 関数のフローチャート

(6) endat_stop フローチャート

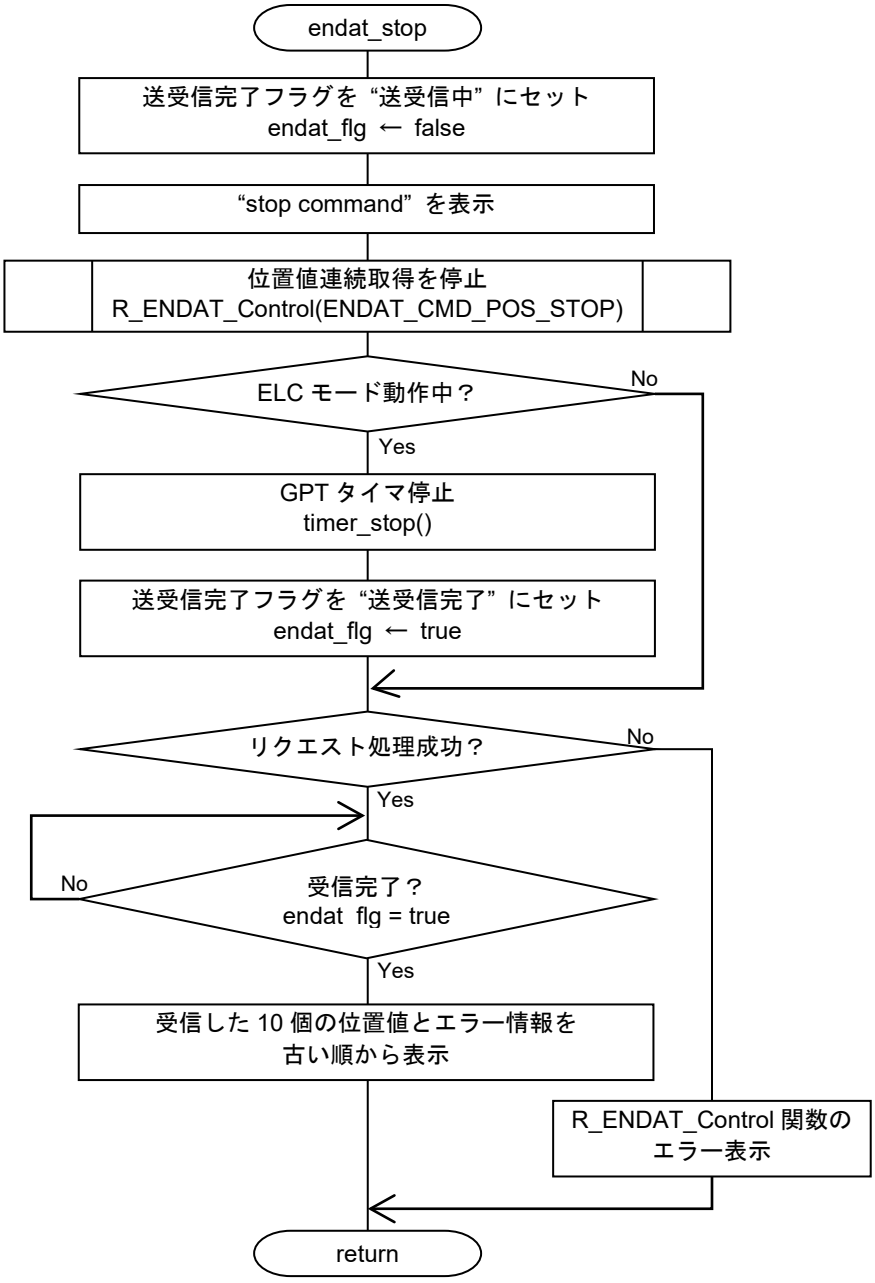

図 4-8 endat_stop 関数のフローチャート

(7) endat_temp フローチャート

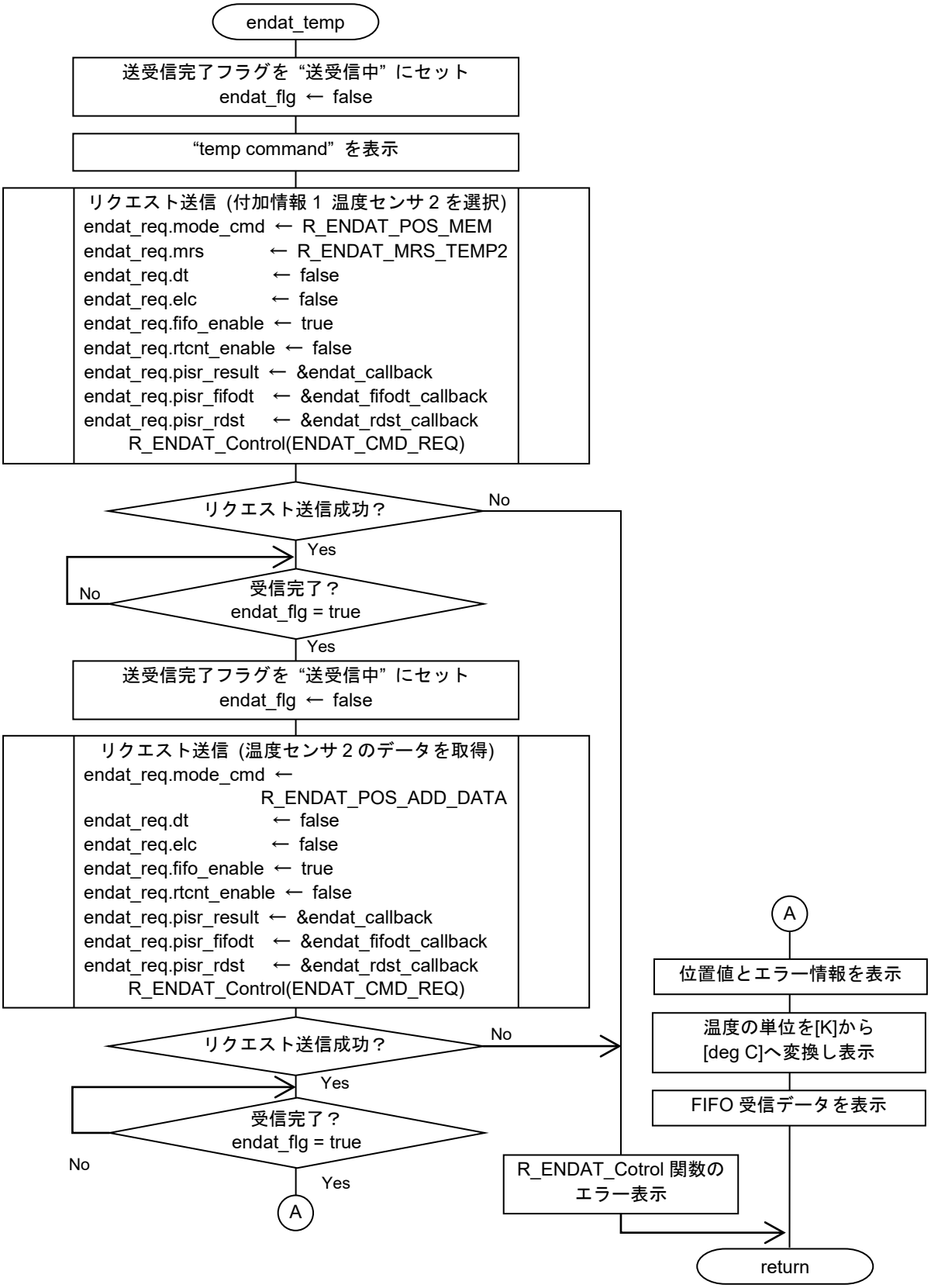

図 4-9 endat_temp 関数のフローチャート

(8) endat_read フローチャート

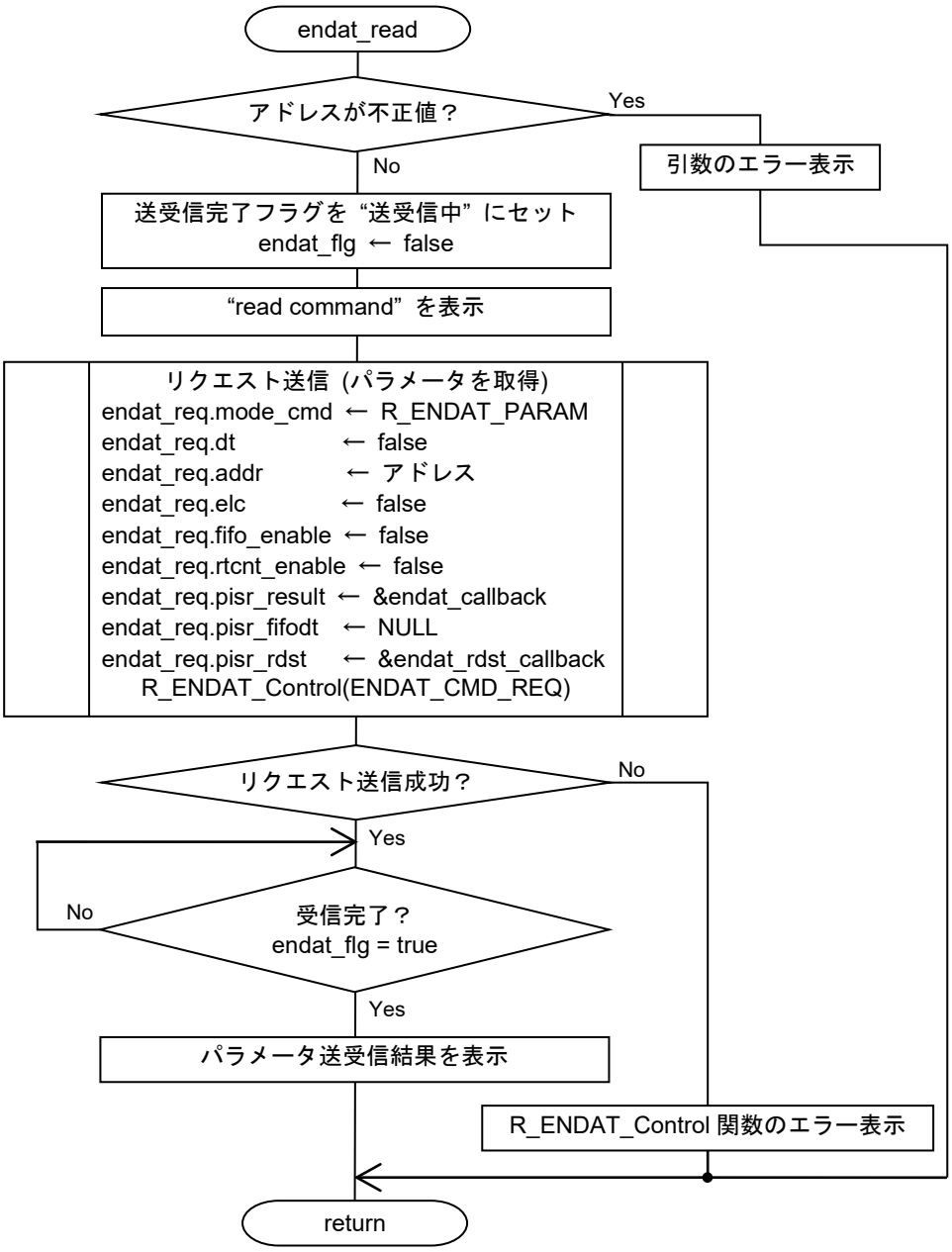

図 4-10 endat_read 関数のフローチャート

(9) endat_write フローチャート

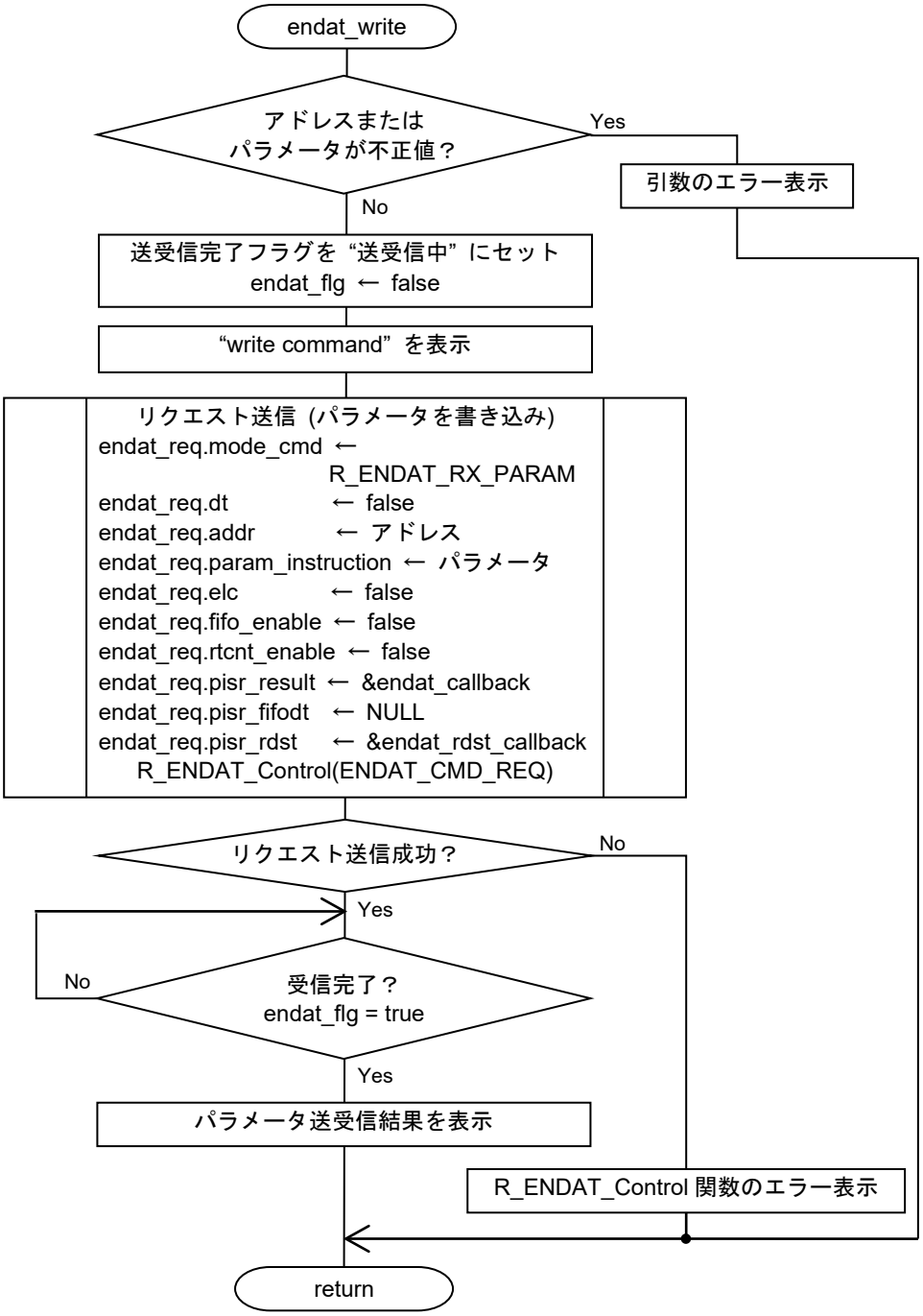

図 4-11 endat_write 関数のフローチャート

(10) endat_spos フローチャート

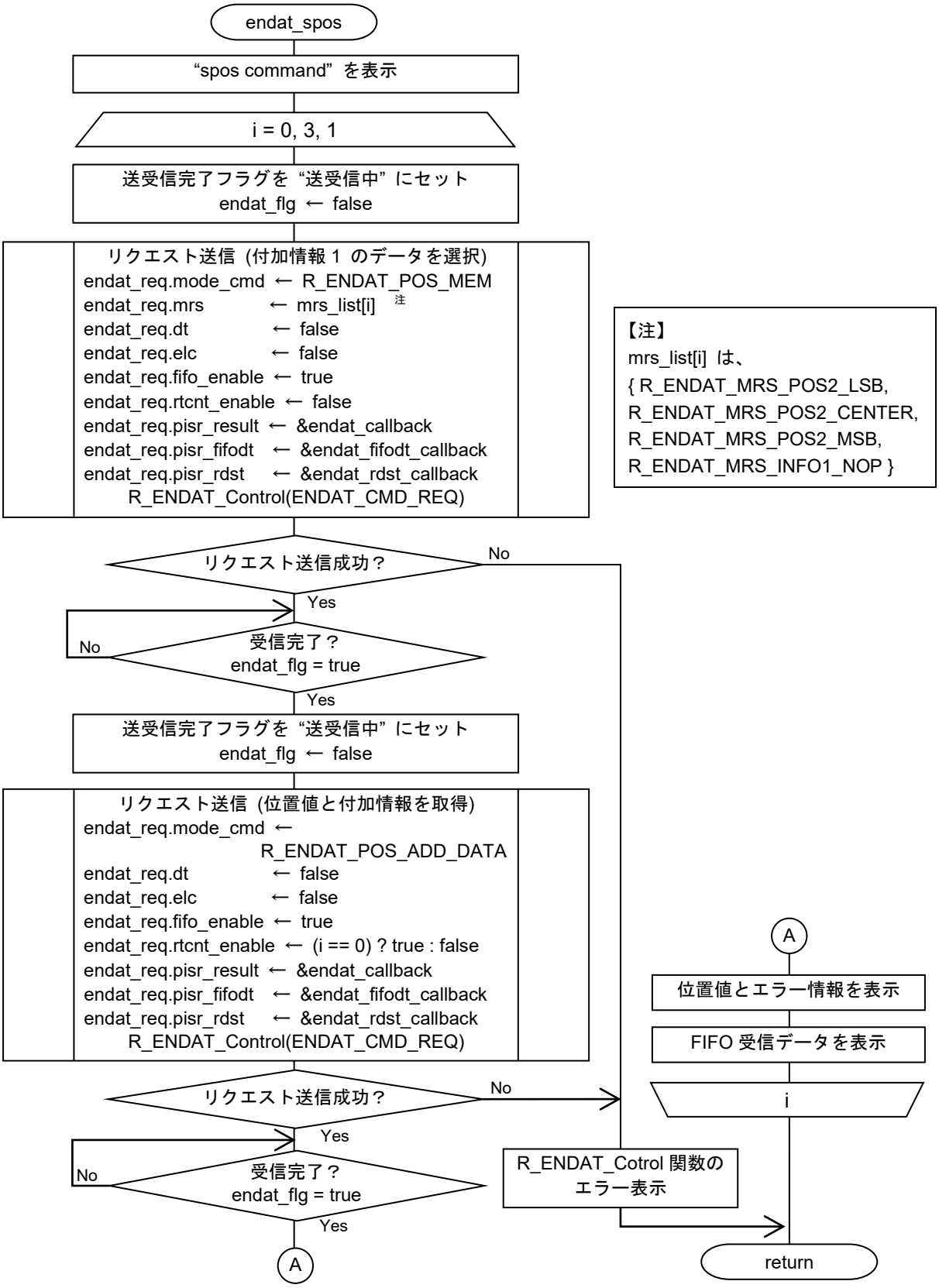

図 4-12　endat_spos 関数のフローチャート

(11) endat_pos_safe フローチャート

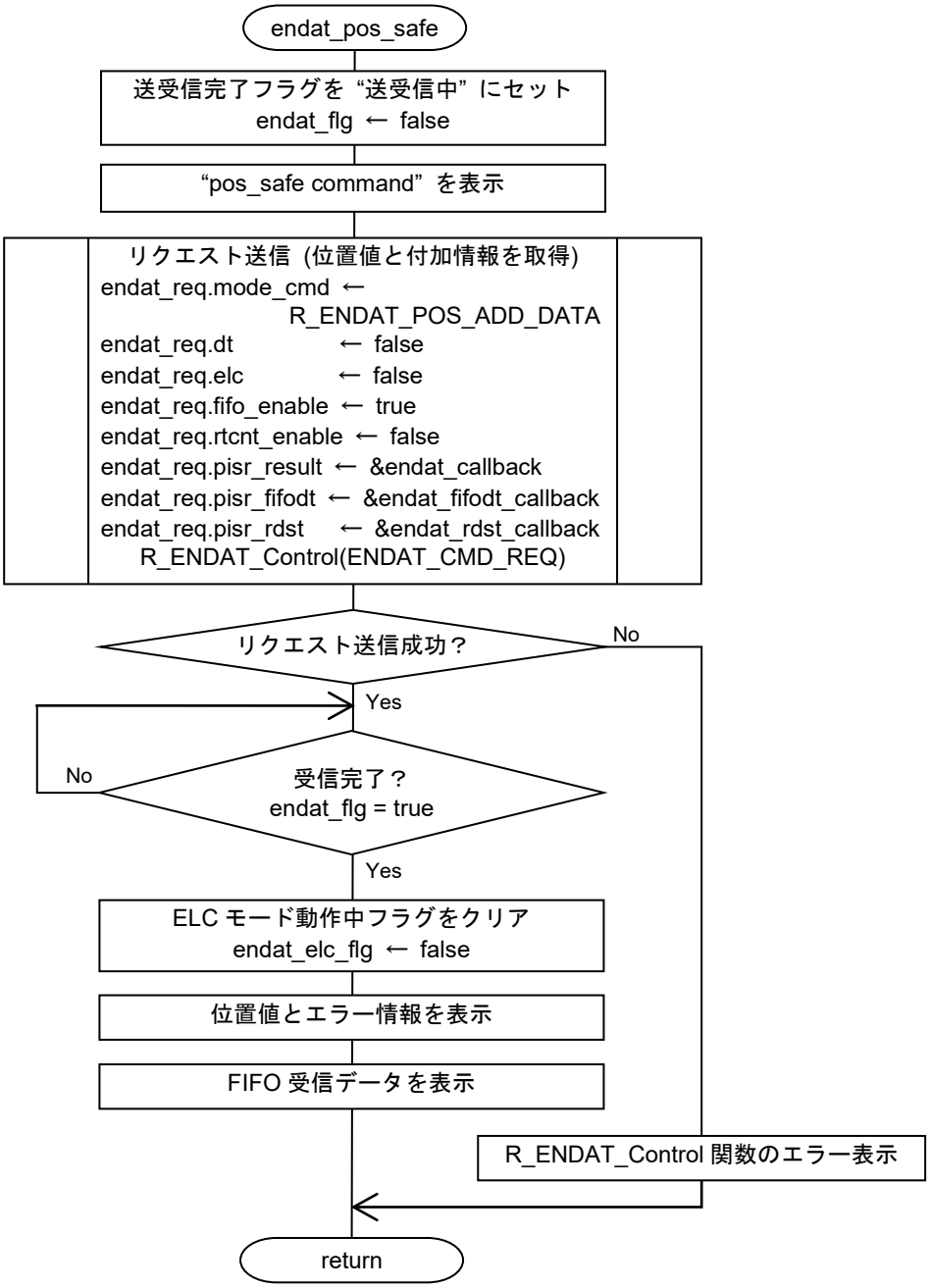

図 4-13 endat_pos_safe 関数のフローチャート

(12) endat_sel_info フローチャート

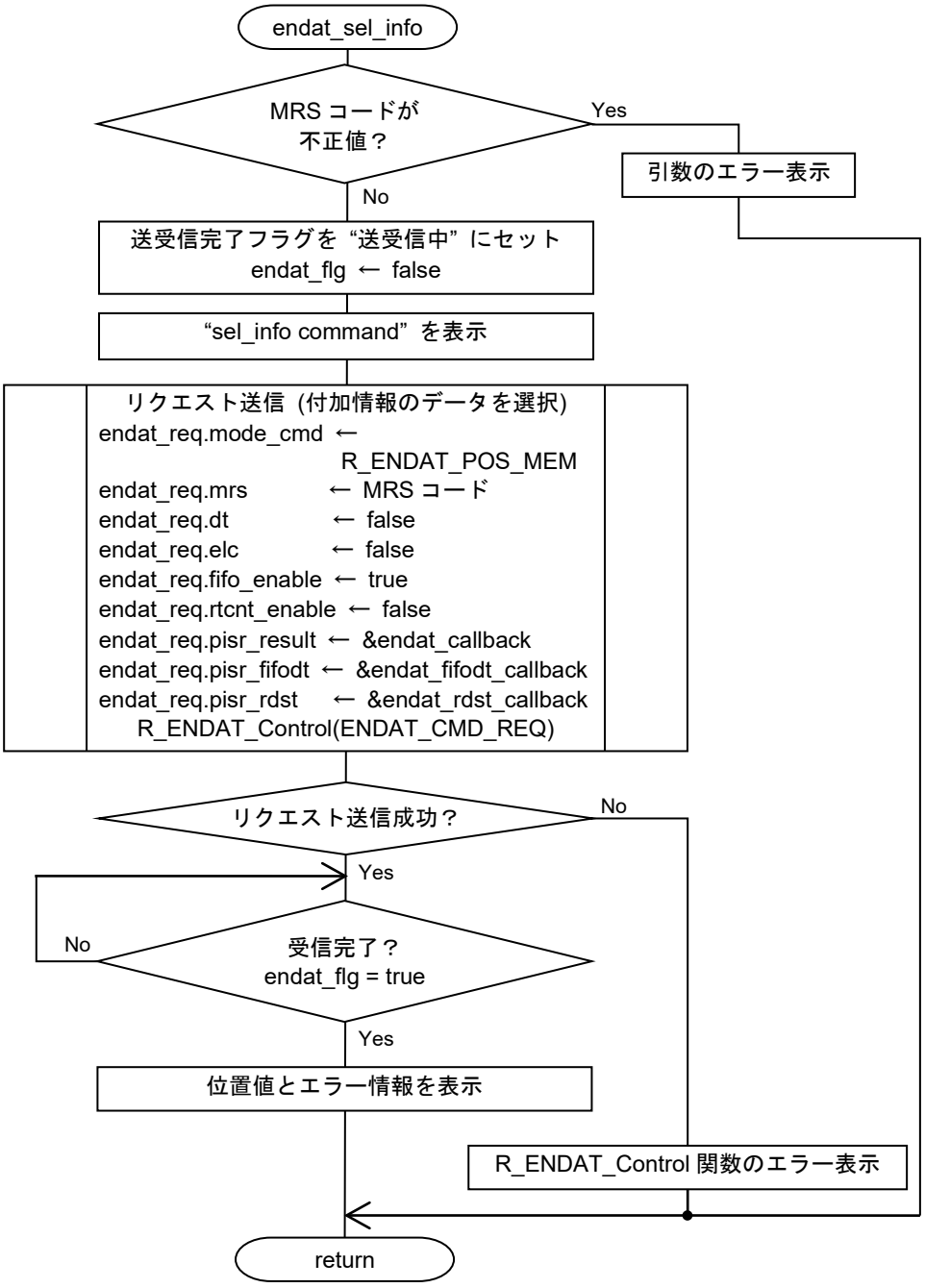

図 4-14 endat_sel_info 関数のフローチャート

(13) endat_callback フローチャート

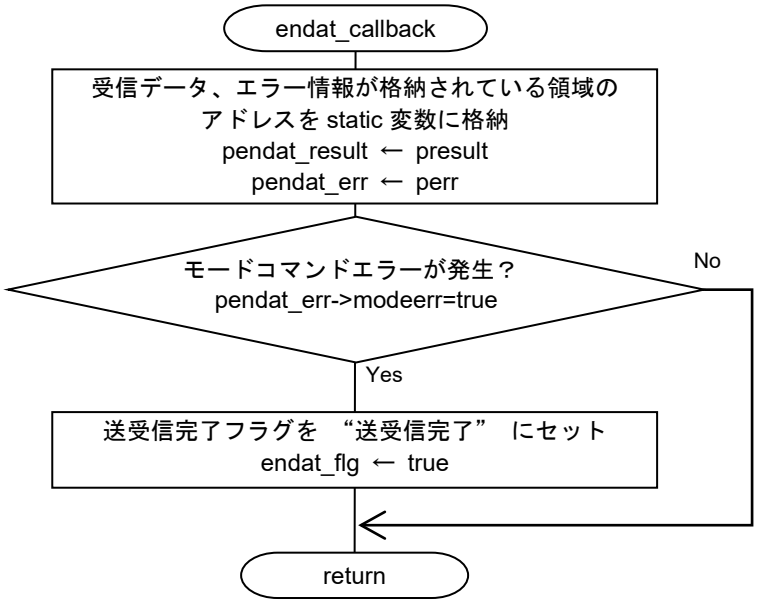

図 4-15 endat_callback 関数のフローチャート

(14) endat_poscon_callback フローチャート

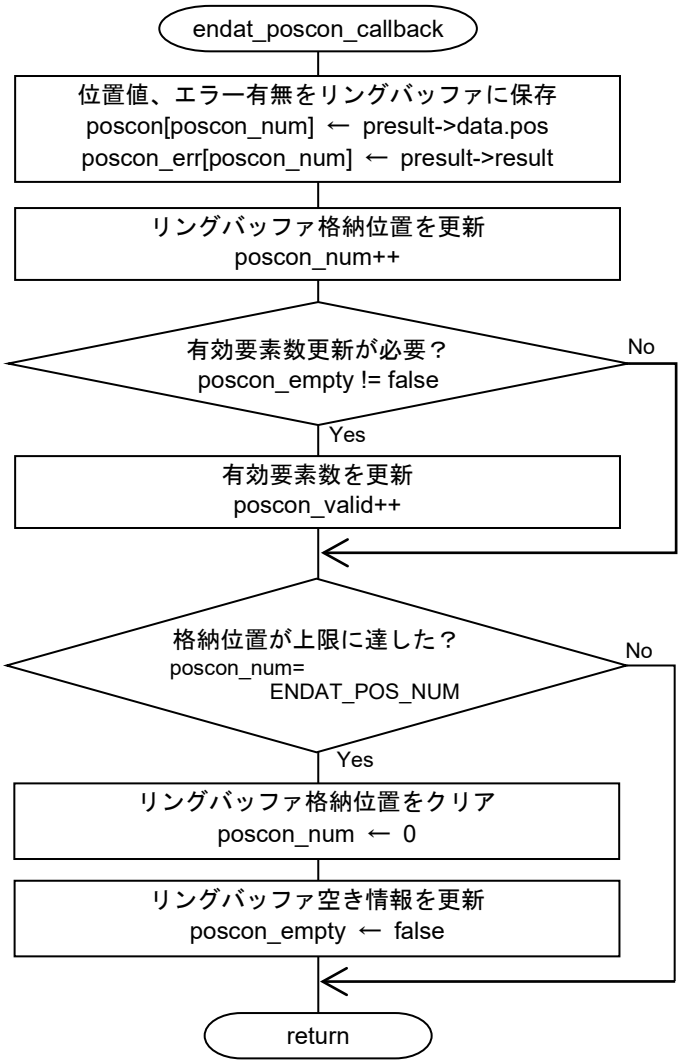

図 4-16　endat_poscon_callback 関数のフローチャート

(15) endat_fifodt_callback フローチャート

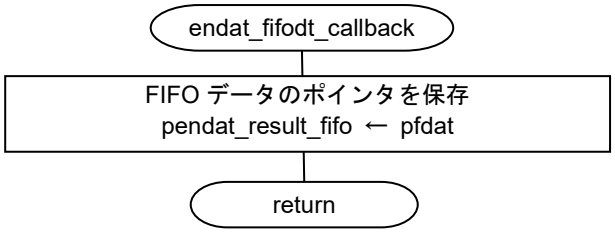

図 4-17　endat_fifodt_callback 関数のフローチャート

(16) endat_rdst_callback フローチャート

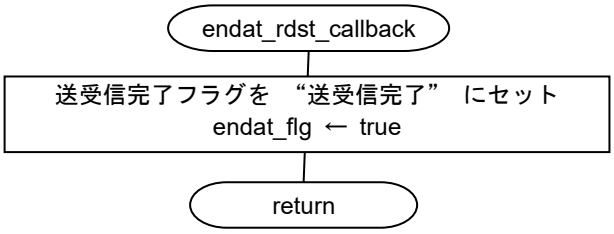

図 4-18 endat_rdst_callback 関数のフローチャート

4.11.7 動作シーケンス

(1) 起動シーケンス

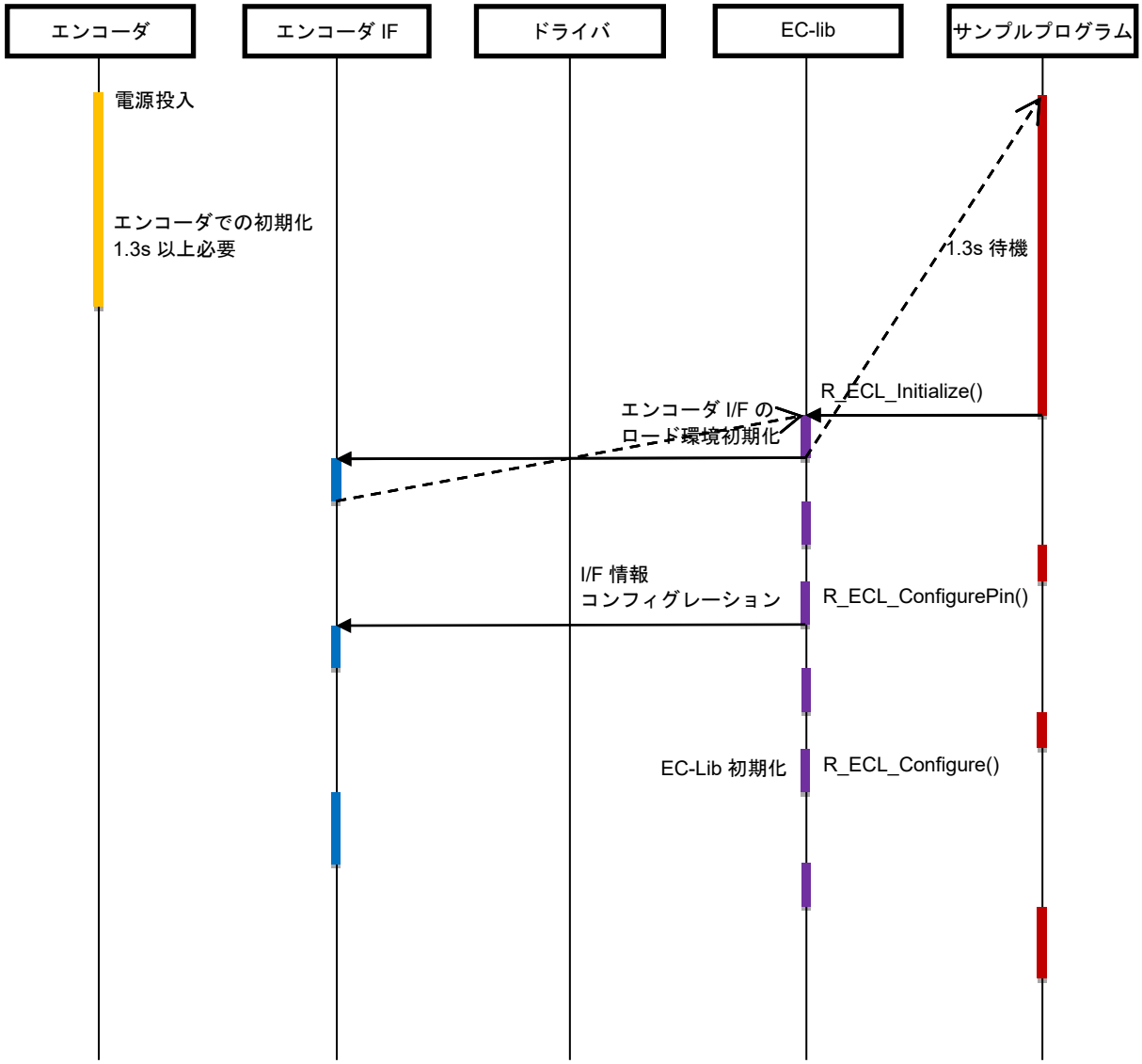

図 4-19 起動シーケンス図

(2) 開始シーケンス

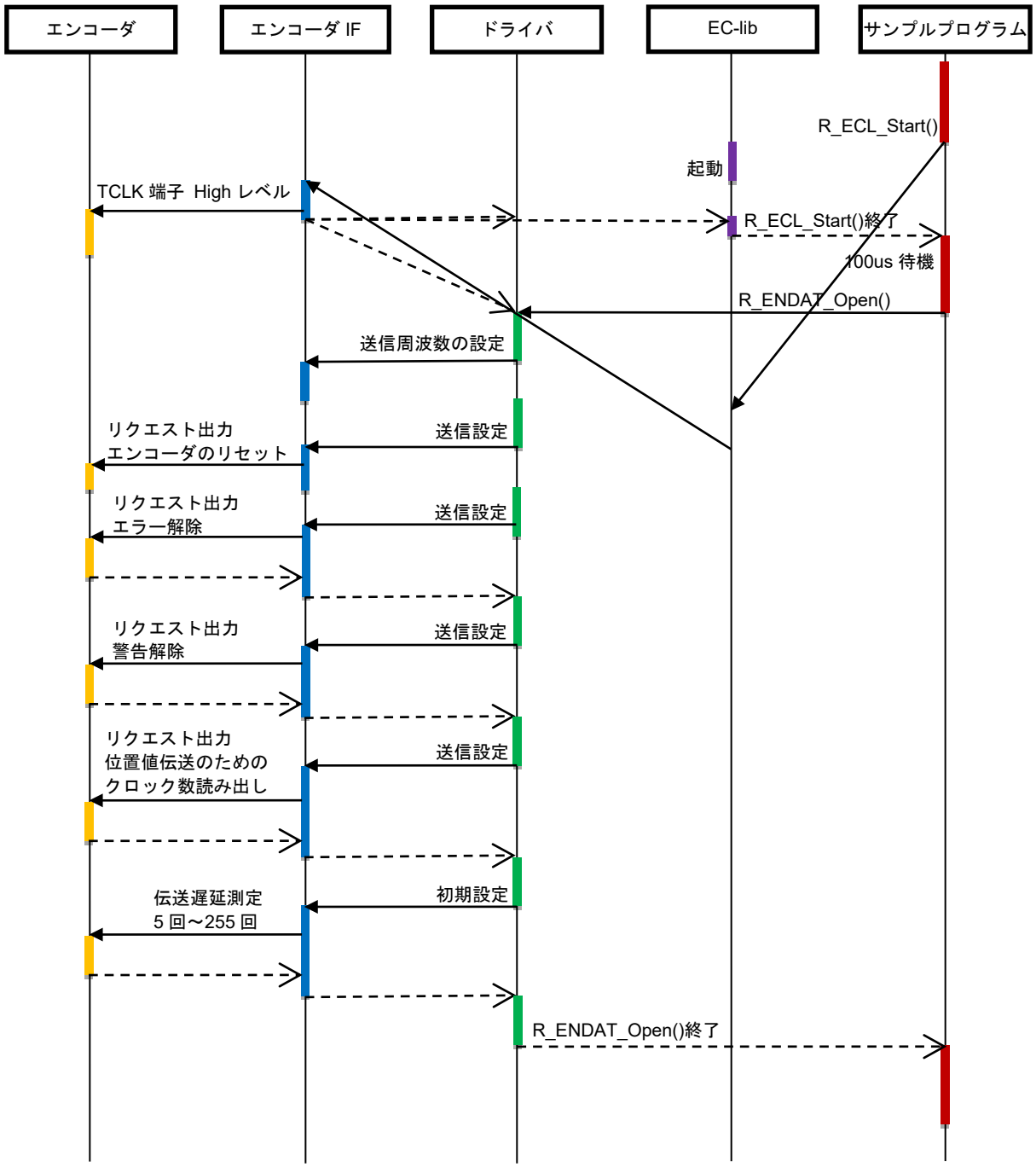

図 4-20 開始シーケンス図

(3) リクエスト送信とデータ受信のシーケンス

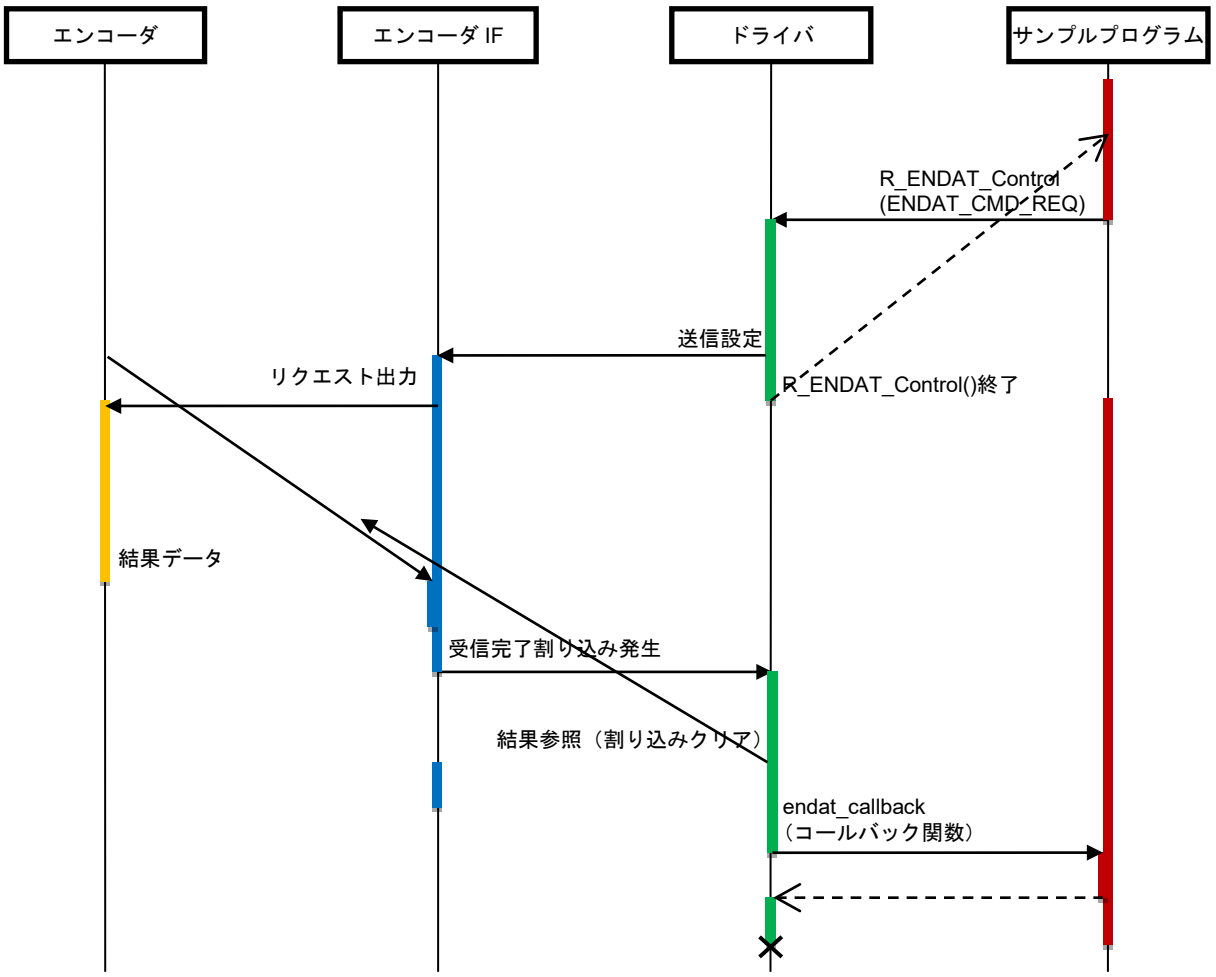

図 4-21 リクエスト送信とデータ受信のシーケンス図

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

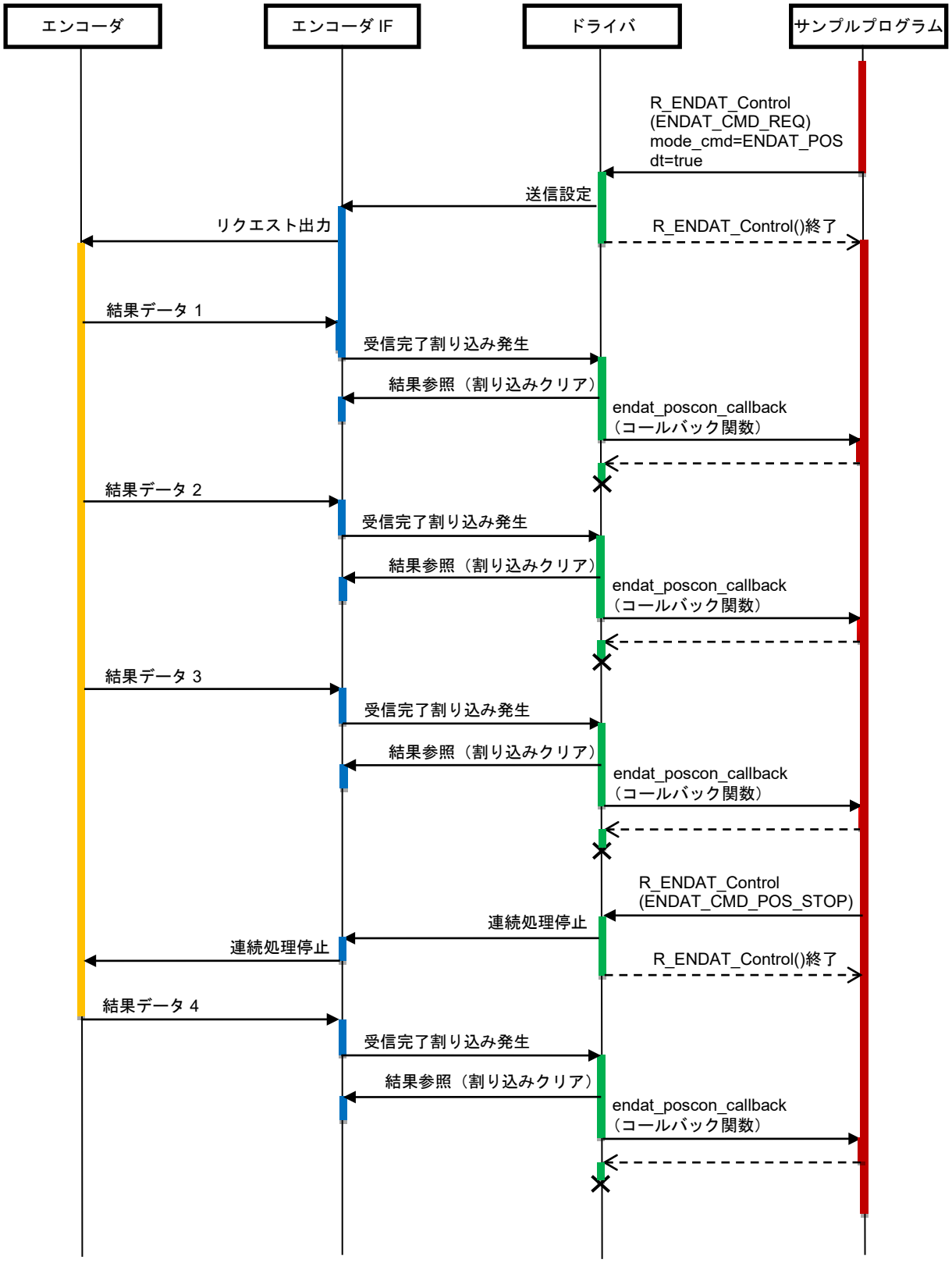

図 4-22 リクエスト送信(Continuous モード)とデータの連続受信シーケンス図

(5) 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

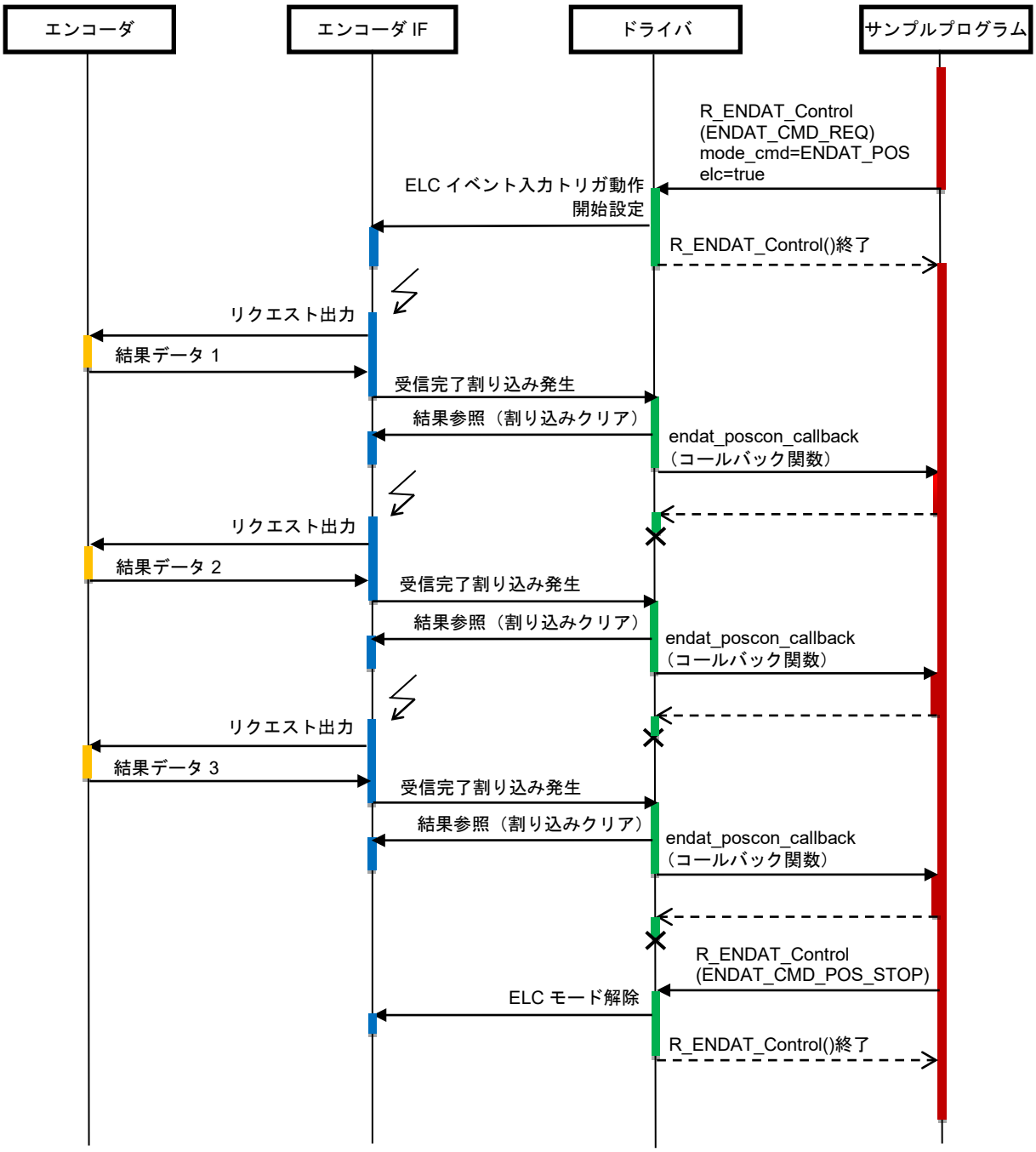

図 4-23 リクエスト送信(ELC モード)とデータの連続受信シーケンス図

(6) 停止シーケンス

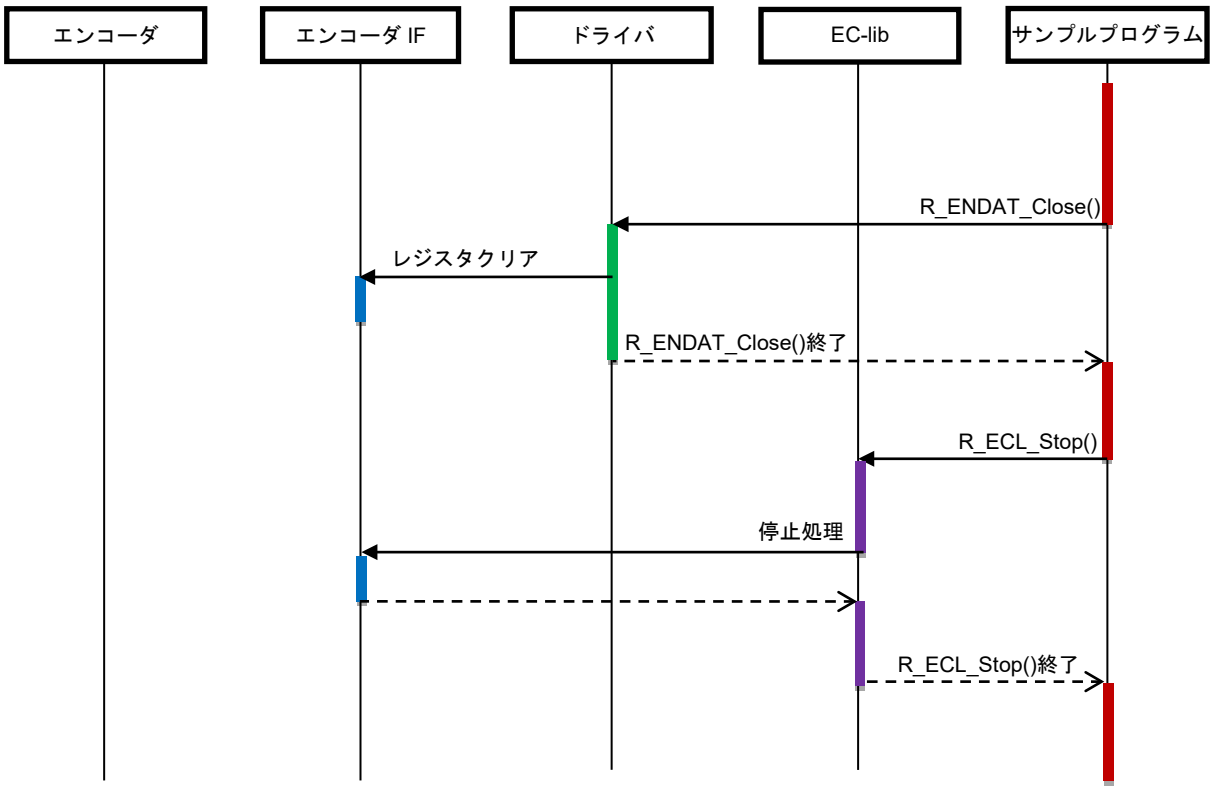

図 4-24 停止シーケンス図

4.11.8 コンソールコマンド

本サンプルプログラムは EnDat 2.2 準拠エンコーダ「ECN1123」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 4.16　コンソールコマンド一覧

コマンド	内容
pos	位置値を 1 回だけ取得します。
spos	付加情報 1 の内容を Position 2 の word 1, Position 2 の word 2, Position 2 の word 3, NOP に切り替えながら、位置値と付加情報 1 とを取得します。Position 2 の word 1 取得時には、リカバリータイム計測を有効にします。 コマンド内で、メモリ領域選択と位置値，付加情報取得が行われます。 リカバリータイム計測は、Position 2 の word 1 データ引き取り時には、まだ終わっていません。word 2 データ以降に RTCNT として示される値が、計測完了後の値です。
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。
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。
stop	位置値の連続取得を停止します。
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。 コマンド内で、メモリ領域選択と位置値，付加情報取得が行われます。
sel_info *MRS*	メモリ領域選択コード (*MRS*)を、16 進数で指定します。
pos_safe	位置値とともに、MRS コードで指定された付加情報の内容を 1 回だけ取得します。
read *addr*	MRS コードで指定されたメモリ領域の、16 進数 *addr* のアドレスからパラメータを読み出します。
write *addr param*	MRS コードで指定されたメモリ領域の、16 進数 *addr* のアドレスに、16 進数 *param* のパラメータを書き込みます。
exit	プログラムを終了します。

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	Feb 28.23	-	初版発行
2.00	Jul 05.24	5 20	使用ボードの表記を更新 固定幅整数の定義場所に関する記載を削除

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。