RZ/N2L グループ

FA-CODER 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/N2L のシリアル通信 I/F（以下 SCI）を使用し、多摩川精機製位置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。
・多摩川精機製位置エンコーダ（FA-CODER®）に対応（データフィールド 8 個まで対応）
・多摩川精機製位置エンコーダから、角度情報等を取得可能

対象デバイス

RZ/N2L

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
割り込みコントローラ(ICU)	GPT, SCI の割り込み制御
汎用 PWM タイマ(GPT)	GPT ユニット 0 チャネル 0 で、DMAC に入力するイベント周期を生成する GPT ユニット 0 チャネル 1 で、FA-CODER 送受信のタイムアウトを監視する
DMA コントローラ(DMAC)	GPT ユニット 0 チャネル 0 が出力するイベントで起動し、イベントに同期した送信トリガに使用する
シリアル通信 I/F (SCI) UART	SCI の調歩同期式 I/F を使用し、チャネル 0 で USB インタフェースによる COM ポート通信を行う チャネル 0 はサンプルプログラムのコンソールインタフェース用 チャネル 3 および 4 を、FA-CODER のエンコーダ I/F に使用する

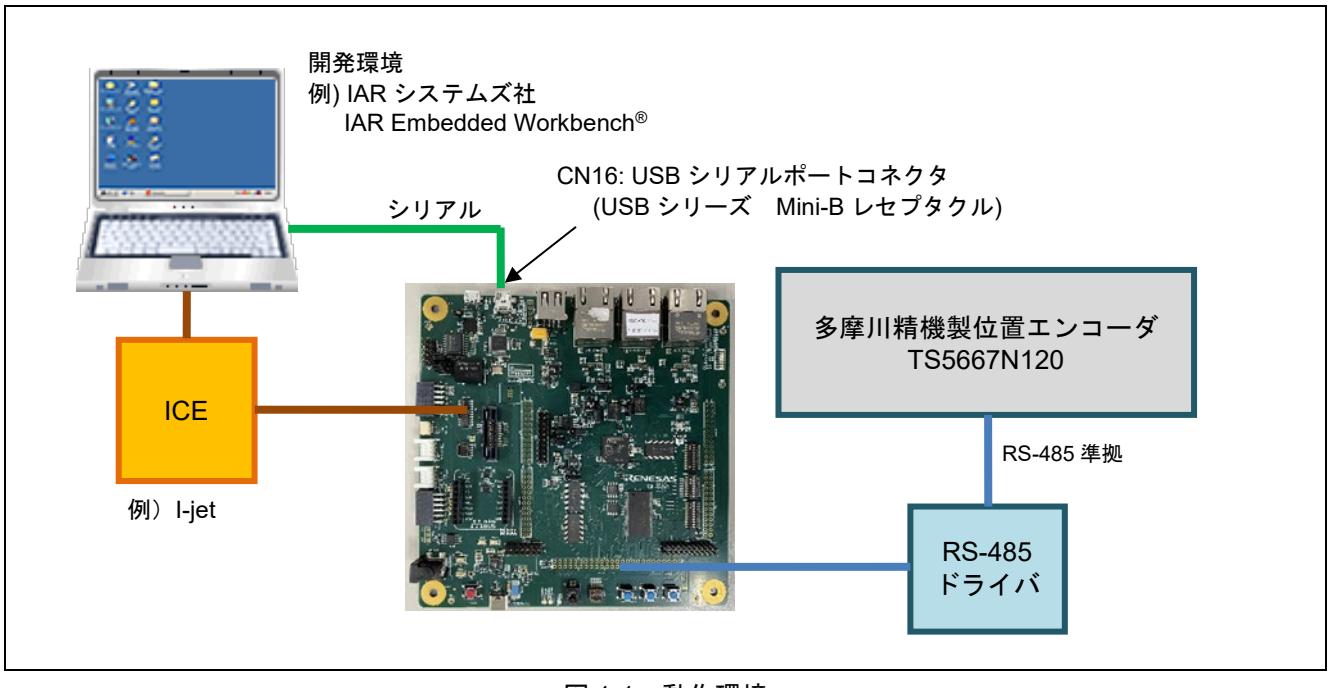


図 1-1　動作環境

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/N2L グループ
動作周波数	CPUCLK = 400MHz
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 Embedded Workbench® for Arm®
	RENESAS 製 e² studio
使用ボード	RSK+RZN2L（RTK9RZN2L0C00000BE）
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/N2L グループ Encoder I/F FA-CODER 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/N2L グループ・ユーザーズマニュアル ハードウェア編に記載しています。

3.1 使用端子一覧

表 3.1 に使用端子と機能を示します。

表 3.1 使用端子と機能

チャネル	端子名	I/O ポート	入出力	内容
FACODER0	RXD3	P04_0	入力	データ受信端子
	TXD3	P04_1	出力	リクエスト出力端子
	DE3	P04_5	出力	ドライブ/レシーブ制御端子
FACODER1	RXD4	P18_5	入力	データ受信端子
	TXD4	P18_4	出力	リクエスト出力端子
	DE4	P18_6	出力	ドライブ/レシーブ制御端子

4. ソフトウェア説明

4.1 FA-CODER ドライバ機能

FA-CODER ドライバの機能は以下です。

1) Encoder I/F、割り込みコントローラ、端子の初期化
2) FA-CODER へのリクエスト送信、結果の受信
3) FA-CODER との通信時のエラー通知

4.2 ファイル構成

ファイル構成は、RZ/N2L グループ Encoder I/F FA-CODER 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

4.4 API 関数仕様

4.4.1 R_FAC_Open

R_FAC_Open		
概　要	エンコーダ制御の開始関数	
ヘッダ	r_fac_rzt2_if.h	
宣　言	r_fac_err_t R_FAC_Open(const int32_t id, const r_fac_info_t *p_info);	
説　明	引数で指定された FA-CODER の開始処理を行います。	
引　数	id	：使用する ID を指定します。（r_fac_rzt2_dat.h で定義されています。）
	R_FAC0_ID	：チャネル 0 を指定
	R_FAC1_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_fac_info_t のアドレスを指定してください。
リターン値	R_FAC_SUCCESS	：正常終了
	R_FAC_ERR_ACCESS	：異常終了（既に open されています）
	R_FAC_ERR_INVALID_ARG	：異常終了（引数 id, p_info に指定した構造体のメンバ変数が規定されていない値）

4.4.2 R_FAC_Close

R_FAC_Close		
概　要	エンコーダ制御の終了処理関数	
ヘッダ	r_fac_rzt2_if.h	
宣　言	r_fac_err_t R_FAC_Close(const int32_t id);	
説　明	エンコーダ I/F ドライバの終了処理を行います。	
引　数	id	：使用する ID を指定します。（r_fac_rzt2_dat.h で定義されています。）
	R_FAC0_ID	：チャネル 0 を指定
	R_FAC1_ID	：チャネル 1 を指定
	上記以外	：設定不可
リターン値	R_FAC_SUCCESS	：正常終了
	R_FAC_ERR_INVALID_ARG	：異常終了（引数 id が規定されていない値）
	R_FAC_ERR_BUSY	：異常終了（転送中のため、操作できません）
注　意	エンコーダと送受信中の場合は Close できません。 Close 中に本関数を実行した場合は、処理は行わず R_FAC_SUCCESS を返します。	

4.4.3 R_FAC_GetVersion

R_FAC_GetVersion		
概　要	エンコーダドライバのバージョン取得関数	
ヘッダ	r_fac_rzt2_if.h	
宣　言	uint32_t R_FAC_GetVersion(void);	
説　明	エンコーダ I/F 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_FAC_Control

R_FAC_Control		
概　要	エンコーダの制御関数	
ヘッダ	r_fac_rzt2_if.h	
宣　言	r_fac_err_t R_FAC_Control(const int32_t id, const r_fac_cmd_t cmd, void *const p_buf);	
説　明	エンコーダ I/F ドライバの動作をコントロールします。引数 cmd にコマンドを指定することによって動作が変わります。各コマンド動作の詳細は「4.4.5 Control コマンド一覧」を参照してください。	
引　数	id	：使用する ID を指定します。（r_fac_rzt2_dat.h で定義されています。）
		R_FAC0_ID ：チャネル 0 を指定
		R_FAC1_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_fac_cmd_t」参照。
	p_buf	：各 cmd に対応する引数
リターン値	R_FAC_SUCCESS	：正常終了
	R_FAC_ERR_ACCESS	：異常終了（該当チャネルが open されていません）
	R_FAC_ERR_BUSY	：異常終了（エンコーダへの送受信中）
	R_FAC_ERR_INVALID_ARG	：異常終了（引数 id, cmd が規定されていない値、p_buf が NULL）

4.4.5 Control コマンド一覧

(1) R_FAC_CMD_REQ

R_FAC_CMD_REQ		
概　要	FA-CODER へのリクエスト送信	
ヘッダ	r_fac_rzt2_if.h	
宣　言	r_fac_err_t R_FAC_Control(const int32_t id, const r_fac_cmd_t cmd, void *const p_buf);	
説　明	FA-CODER へリクエストを送信します。	
引　数	id	：使用する ID を指定します。（r_fac_rzt2_dat.h で定義されています。）
		R_FAC0_ID ：チャネル 0 を指定
		R_FAC1_ID ：チャネル 1 を指定
		上記以外 ：設定不可
	cmd	：R_FAC_CMD_REQ
	p_buf	：リクエスト情報 リクエスト情報を格納した r_fac_req_t 構造体のポインタを指定します。 r_fac_req_t 構造体の詳細は「4.10.1(2) r_fac_req_t」参照。
リターン値	R_FAC_SUCCESS	：正常終了
	R_FAC_ERR_ACCESS	：異常終了（該当チャネルが open されていません）
	R_FAC_ERR_INVALID_ARG	：異常終了（引数 id, cmd が規定されていない値、p_buf が NULL、p_buf に指定された r_fac_req_t 構造体のメンバ変数が規定されていない値）
	R_FAC_ERR_BUSY	：異常終了（転送中のため、操作できません）

(2) R_FAC_CMD_E2PROM

R_FAC_CMD_E2PROM		
概　要	FA-CODER への E2PROM アクセスリクエスト送信	
ヘッダ	r_fac_rzt2_if.h	
宣　言	r_fac_err_t R_FAC_Control(const int32_t id, const r_fac_cmd_t cmd, void *const p_buf);	
説　明	FA-CODER へ E2PROM アクセスリクエストを送信します。	
引　数	id	：使用する ID を指定します。（r_fac_rzt2_dat.h で定義されています。）
		R_FAC0_ID　：チャネル 0 を指定
		R_FAC1_ID　：チャネル 1 を指定
		上記以外　：設定不可
	cmd	：R_FAC_CMD_E2PROM
	p_buf	：E2PROM 情報
		E2PROM 情報を格納した r_fac_e2prom_data_t 構造体のポインタを指定します。r_fac_e2prom_data_t 構造体の詳細は「4.10.1(3) r_fac_e2prom_data_t」参照。
リターン値	R_FAC_SUCCESS	：正常終了
	R_FAC_ERR_ACCESS	：異常終了（該当チャネルが open されていません）
	R_FAC_ERR_INVALID_ARG	：異常終了（引数 id, cmd が規定されていない値、p_buf が NULL、p_buf に指定された r_fac_e2prom_data_t 構造体のメンバ変数が規定されていない値）
	R_FAC_ERR_BUSY	：異常終了（転送中のため、操作できません）

(3) R_FAC_CMD_ELCTIMER

R_FAC_CMD_ELCTIMER		
概　要	FA-CODER へのタイマイベント同期リクエスト送信	
ヘッダ	r_fac_rzt2_if.h	
宣　言	r_fac_err_t R_FAC_Control(const int32_t id, const r_fac_cmd_t cmd, void *const p_buf);	
説　明	FA-CODER へタイマイベントに同期してリクエストを送信します。 コマンド名には、RZ/T2M グループ FA-CODER ドライバとの互換性のために、ELC の名称が使われています。RZ/N2L グループ FA-CODER ドライバでは、イベントリンクコントローラ(ELC)による送信トリガに対応していませんが、イベントに同期して DMA を起動することで、CPU を経由しない送信トリガ生成を実現します。	
引　数	id	：使用する ID を指定します。（r_fac_rzt2_dat.h で定義されています。）
		R_FAC0_ID　：チャネル 0 を指定
		R_FAC1_ID　：チャネル 1 を指定
		上記以外　：設定不可
	cmd	：R_FAC_CMD_ELCTIMER
	p_buf	：リクエスト情報
		リクエスト情報を格納した r_fac_req_t 構造体のポインタを指定します。 r_fac_req_t 構造体の詳細は「4.10.1(2) r_fac_req_t」参照。
リターン値	R_FAC_SUCCESS	：正常終了
	R_FAC_ERR_ACCESS	：異常終了（該当チャネルが open されていません）
	R_FAC_ERR_INVALID_ARG	：異常終了（引数 id, cmd が規定されていない値、p_buf が NULL、p_buf に指定された r_fac_req_t 構造体のメンバ変数が規定されていない値）
	R_FAC_ERR_BUSY	：異常終了（転送中のため、操作できません）

(4) R_FAC_CMD_ELCSTOP

R_FAC_CMD_ELCSTOP		
概　要	FA-CODER へのタイマイベント同期リクエスト停止	
ヘッダ	r_fac_rzt2_if.h	
宣　言	r_fac_err_t R_FAC_Control(const int32_t id, const r_fac_cmd_t cmd, void *const p_buf);	
説　明	FA-CODER へのタイマイベント同期リクエストを停止します。	
引　数	id	：使用する ID を指定します。（r_fac_rzt2_dat.h で定義されています。）
		R_FAC0_ID　：チャネル 0 を指定
		R_FAC1_ID　：チャネル 1 を指定
		上記以外　：設定不可
	cmd	：R_FAC_CMD_ELCSTOP
	p_buf	：未使用
		NULL を指定してください。
リターン値	R_FAC_SUCCESS	：正常終了
	R_FAC_ERR_ACCESS	：異常終了（該当チャネルが open されていません，タイマイベント同期リクエストが送信されていません）
	R_FAC_ERR_INVALID_ARG	：異常終了

4.5 ユーザー定義関数仕様

4.5.1 callback_req_result

callback_req_result		
概　要	FA-CODER へのリクエスト送信に対するデータ受信結果を通知するコールバック関数	
ヘッダ	-	
宣　言	void callback_req_result(r_fac_result_t * p_result, uint8_t * p_rxdf);	
説　明	R_FAC_Control(R_FAC_CMD_REQ)関数で登録したコールバック関数です。リクエストに対するデータ受信結果を通知します。 本関数は割り込みハンドラのコンテキストとなります。割り込みの応答性を確保するため、速やかに return するようにしてください。関数名は例であり、自由に設定できます。	
引　数	p_result	：受信結果 「4.10.1(4) r_fac_result_t」参照。本構造体は次回コマンド実行時まで有効です。次回コマンド実行後に参照する場合は、適当なメモリへコピーを行ってください。
	p_rxdf	：受信データ 要素数 8 の配列となります。有効データ数はエンコーダの種類と送信 ID コードに依存します。本配列は次回コマンド実行時まで有効です。次回コマンド実行後に参照する場合は、適当なメモリへコピーを行ってください。
リターン値	なし	

4.5.2 callback_e2prom_result

callback_e2prom_result	
概　要	FA-CODER へのリクエスト送信に対するデータ受信結果を通知するコールバック関数
ヘッダ	-
宣　言	void callback_e2prom_result(r_fac_result_t * p_result, uint8_t adf, uint8_t edf);
説　明	R_FAC_Control(R_FAC_CMD_E2PROM)関数で登録したコールバック関数です。リクエストに対するデータ受信結果を通知します。 本関数は割り込みハンドラのコンテキストとなります。割り込みの応答性を確保するため、速やかに return するようにしてください。関数名は例であり、自由に設定できます。
引　数	p_result ：受信結果 「4.10.1(4) r_fac_result_t」参照。本構造体は次回コマンド実行時まで有効です。次回コマンド実行後に参照する場合は、適当なメモリへコピーを行ってください。 adf ：E2PROM の受信データ(ADF) edf ：E2PROM の受信データ(EDF)
リターン値	なし

4.5.3 callback_elctimer_result

callback_elctimer_result	
概　要	FA-CODER へのリクエスト送信に対するデータ受信結果を通知するコールバック関数
ヘッダ	-
宣　言	void callback_elctimer_result(r_fac_result_t * p_result, uint8_t *p_rxdf);
説　明	R_FAC_Control(R_FAC_CMD_ELCTIMER)関数で登録したコールバック関数です。リクエストに対するデータ受信結果を通知します。 本関数は割り込みハンドラのコンテキストとなります。割り込みの応答性を確保するため、速やかに return するようにしてください。関数名は例であり、自由に設定できます。
引　数	p_result ：受信結果 「4.10.1(4) r_fac_result_t」参照。本構造体は次回コマンド実行時まで有効です。次回コマンド実行後に参照する場合は、適当なメモリへコピーを行ってください。 p_rxdf ：受信データ 要素数 8 の配列となります。有効データ数はエンコーダの種類と送信 ID コードに依存します。 本配列は次回コマンド実行時まで有効です。次回コマンド実行後に参照する場合は、適当なメモリへコピーを行ってください。
リターン値	なし

4.6 割り込みハンドラ

4.6.1 fac0_eri_isr

fac0_eri_isr	
概　要	チャネル 0 受信エラー割り込みハンドラ
ヘッダ	-
宣　言	void fac0_eri_isr(void);
説　明	FA-CODER チャネル 0 からの受信エラー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 fac0_rxi_isr

fac0_rxi_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 fac0_rxi_isr(void);
説　明	FA-CODER チャネル 0 から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

4.6.3 fac0_txi_isr

fac0_txi_isr	
概　要	チャネル 0 データ送信割り込みハンドラ
ヘッダ	-
宣　言	void fac0_txi_isr(void);
説　明	FA-CODER チャネル 0 からのデータ送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 fac0_tei_isr

fac0_tei_isr	
概　要	チャネル 0 データ送信完了割り込みハンドラ
ヘッダ	-
宣　言	void fac0_tei_isr(void);
説　明	FA-CODER チャネル 0 からのデータ送信完了割り込みに対する割り込みハンドラです。
引　数	なし
リターン値	なし

4.6.5 fac1_eri_isr

fac1_eri_isr	
概　要	チャネル 1 受信エラー割り込みハンドラ
ヘッダ	-
宣　言	void fac1_eri_isr(void);
説　明	FA-CODER チャネル 1 からの受信エラー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 fac1_rxi_isr

fac1_rxi_isr	
概　要	チャネル 1 データ受信割り込みハンドラ
ヘッダ	-
宣　言	void fac1_rxi_isr(void);
説　明	FA-CODER チャネル 1 から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

4.6.7 fac1_txi_isr

fac1_txi_isr	
概　要	チャネル 1 データ送信割り込みハンドラ
ヘッダ	-
宣　言	void fac1_txi_isr(void);
説　明	FA-CODER チャネル 1 からのデータ送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

4.6.8 fac1_tei_isr

fac1_tei_isr	
概　要	チャネル 1 データ送信完了割り込みハンドラ
ヘッダ	-
宣　言	void fac1_tei_isr(void);
説　明	FA-CODER チャネル 1 からのデータ送信完了割り込みに対する割り込みハンドラです。
引　数	なし
リターン値	なし

4.6.9 fac_gpt1_isr

fac_gpt1_isr	
概　要	GPT1 タイムアウト割り込みハンドラ
ヘッダ	-
宣　言	void fac_gpt1_isr(void);
説　明	FA-CODER 受信タイムアウト割り込みに対する割り込みハンドラです。
引　数	なし
リターン値	なし

4.7 使用割り込み一覧

表 4.2 に FA-CODER ドライバで使用する割り込みを示します。

表 4.2 FA-CODER 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 受信エラー	300	チャネル 0 (SCI3)でデータ受信エラー発生
チャネル 0 データ受信	301	チャネル 0 (SCI3)でデータ受信
チャネル 0 データ送信	302	チャネル 0 (SCI3)でデータ送信
チャネル 0 データ送信完了	303	チャネル 0 (SCI3)でデータ送信完了
チャネル 1 受信エラー	304	チャネル 1 (SCI4)でデータ受信エラー発生
チャネル 1 データ受信	305	チャネル 1 (SCI4)でデータ受信
チャネル 1 データ送信	306	チャネル 1 (SCI4)でデータ送信
チャネル 1 データ送信完了	307	チャネル 1 (SCI4)でデータ送信完了
GPT1 タイムアウト	131	データ受信時にタイムアウト発生

4.8 定数/エラーコード一覧

表 4.3 に FA-CODER ドライバで使用する定数(r_fac_rzt2.c)、表 4.4 にビットレート指定(r_fac_rzt2_if.h)を示します。エラーコードは「4.10.3(1) r_fac_err_t」を参照してください。

表 4.3 FA-CODER ドライバで使用する定数(r_fac_rzt2.c)

定数名	設定値	内容
FAC_ID_NUM	2	FA-CODER コンフィグレーション ID の総数
FAC_CMD_NUM	4	FA-CODER ドライバのコマンド総数
FAC_RXDF_MAX	8	最大データフィールド数（変更不可）

表 4.4 ビットレート指定(r_fac_rzt2_if.h)

定数名	設定値	内容
R_FAC_2500KBPS	0	2.5Mbps
R_FAC_5MBPS	1	5Mbps
R_FAC_BITRATE_NUM	2	ビットレート定数総数

4.9 固定幅整数一覧

表 4.5 にサンプルコードで使用する固定幅整数を示します。サンプルコードで使用する固定幅定数は、標準ライブラリで定義されています。

表 4.5 サンプルコードで使用する固定幅整数

シンボル	内容
int8_t	8 ビット整数、符号あり（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_fac_info_t

FA-CODER 制御部の初期化情報。

```
typedef struct
{
    uint8_t             bitrate;        ビットレート
                                        エンコーダとの通信におけるビットレートを指定してください。
                                        指定する値は「表 4.4 ビットレート指定(r_fac_rzt2_if.h)」を参照してください。
} r_fac_info_t
```

(2) r_fac_req_t

FA-CODER に送信するリクエスト情報。

```
typedef struct
{
    uint8_t             txid;           送信 ID コード（0～15）
    uint8_t             dfnum;          データフィールド数（1～8）
                                        エンコーダの種類と送信 ID コードで一意に決まるデータフィールド数を設定します。
    uint16_t            timotn;         送信開始から受信完了までの制限時間
                                        timotn×50(ns)が制限時間となります。推奨値は 4000(200us)。
    r_fac_result_cb_t   p_result_cb;    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「4.5.1 callback_req_result」参照。
                                        NULL を指定するとコールバックが発生しません。
} r_fac_req_t
```

(3) r_fac_e2prom_data_t

E2PROM に送信するリクエスト情報。

```
typedef struct
{
    uint16_t                timotn;         送信開始から受信完了までの制限時間
                                            timotn×50(ns)が制限時間となります。推奨値は
                                            4000(200us)。
    uint8_t                 adr;            E2PROM のアドレス
    uint8_t                 data;           E2PROM のデータ
    r_fac_e2prom_dir_t      dir;            E2PROM 読み書きの方向
                                            「4.10.3(4) r_fac_e2prom_dir_t」参照
    r_fac_e2prom_result_cb_t p_result_cb;   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 callback_e2prom_result」参照。
                                            NULL を指定するとコールバックが発生しません。
} r_fac_e2prom_data_t
```

(4) r_fac_result_t

FA-CODER との通信結果ステータス。

```
typedef struct
{
    r_fac_rx_err_t      result;     受信結果
                                    内容は「4.10.3(3) r_fac_rx_err_t」参照。
    bool                rse;        リードシーケンスエラー 注1
    bool                ide;        受信 ID エラー 注1
    bool                ebusy;      E2PROM アクセス受信ビジーステータスビット
    uint8_t             rxid;       受信したリクエスト ID
    uint8_t             rxidp;      受信したリクエスト ID のパリティビット
    uint8_t             rxsfic;     受信したインフォメーションコード
    uint8_t             rxsfea;     受信したエンコーダアラーム
    uint8_t             rxsfca;     受信した通信アラーム
    uint8_t             crc;        受信した CRC データ
    bool                conte;      コントロールフィールドエラー 注1
    bool                crce;       CRC エラー 注1
    bool                fome;       フォームエラー 注1
    bool                sfome;      ショートフォームエラー 注1
    bool                timote;     タイムアウトエラー 注1
    bool                rxedfe;     受信 EDF エラー 注1
    bool                rxadfe;     受信 ADF エラー 注1
    bool                dfovfe;     受信データフィールド数オーバーフローエラー
                                    (本ビットは使われません。常に false です) 注2
    bool                orer;       オーバーランエラー 注1
} r_fac_result_t
```

【注】 1. エラー発生時に true になります。
2. RZ/T2M グループ FA-CODER エンコーダドライバ I/F との互換性のために設けられています。
RZ/N2L では使われません。

4.10.2 共用体

使用しません。

4.10.3 列挙型

(1) r_fac_err_t

FA-CODER I/F ドライバのエラーコード。

```
typedef enum
{
    R_FAC_SUCCESS              =0,    正常終了
    R_FAC_ERR_INVALID_ARG      ,      引数異常
    R_FAC_ERR_BUSY             ,      API を実行できない状態
    R_FAC_ERR_ACCESS                  API の実行順序エラー
} r_fac_err_t
```

(2) r_fac_cmd_t

R_FAC_Control 関数を使用する時のコマンド設定。

```
typedef enum
{
    R_FAC_CMD_REQ              ,      エンコーダへのリクエスト送信
    R_FAC_CMD_E2PROM           ,      E2PROM へのアクセス
    R_FAC_CMD_ELCTIMER         ,      タイマイベント入力トリガによるリクエスト送受信開始 注
    R_FAC_CMD_ELCSTOP                 タイマイベント入力トリガによるリクエスト送受信停止 注
} r_fac_cmd_t
```

【注】 RZ/T2M グループ FA-CODER エンコーダドライバ I/F との互換性のために ELC の名称が使われています。RZ/N2L グループ FA-CODER ドライバでは、イベントリンクコントローラ(ELC)による送信トリガに対応していませんが、イベントに同期して DMA を起動することで、CPU を経由しない送信トリガを実現します。

(3) r_fac_rx_err_t

エンコーダ I/F のエラーコード。

```
typedef enum
{
    R_FAC_SUCCESS              =0,    正常終了
    R_FAC_ERR                         エラー終了
} r_fac_rx_err_t
```

(4) r_fac_e2prom_dir_t

E2PROM 読み書きの方向。

```
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 FA-CODER へ送信
2) FA-CODER から受信したデータをコンソールに表示
3) タイマイベントで DMA を起動することにより、イベントに同期してコマンドを送受信します。

(1) システムブロック図

図 4-1 にシステムブロック図を示します。

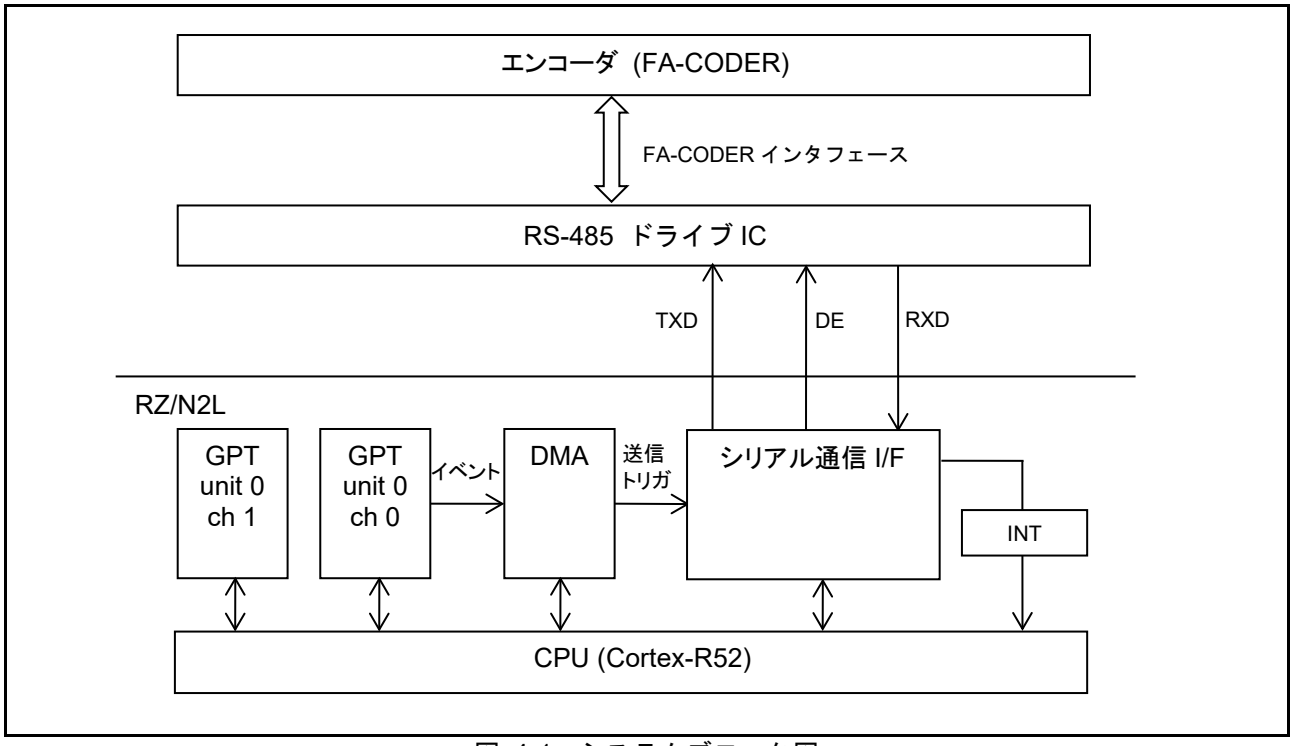


図 4-1 システムブロック図

(2) ソフトウェア構成図

図 4-2 にソフトウェア構成図を示します。

FA-CODER ドライバには、R_FAC_Open 関数で構成される開始処理部、R_FAC_Close 関数で構成される終了処理部、R_FAC_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送受信部分（割り込みハンドラ）があります。

サンプルプログラムには、FA-CODER ドライバを制御し、リクエスト送信を行う FA-CODER 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。

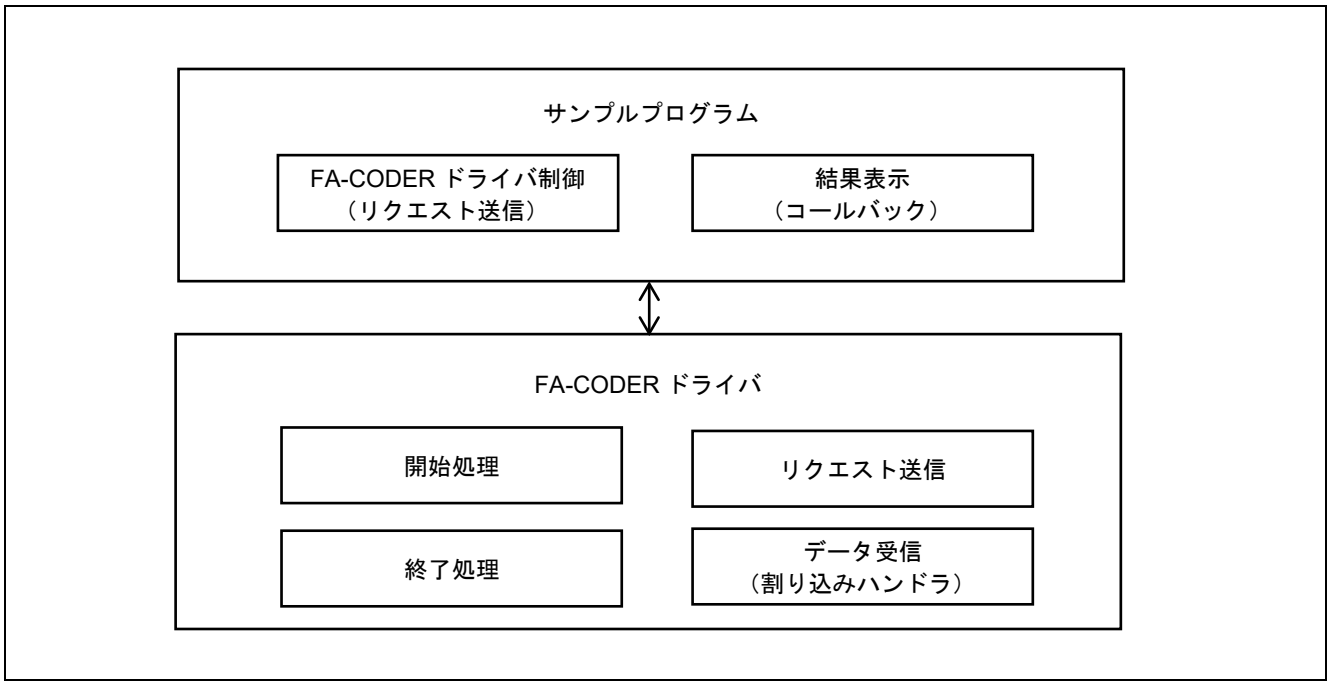


図 4-2 ソフトウェア構成図

4.11.2 サンプルプログラム関数一覧

表 4.6 に主要なサンプルプログラム関数一覧を示します

表 4.6　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	22	-
enc_main	22	29
fac_get_cmd	22	30
fac_cmd_single	22	31
fac_cmd_multi	23	
fac_cmd_encid	23	
fac_cmd_req	23	32
fac_cmd_e2prom_write	23	33
fac_cmd_e2prom_read	24	34
fac_cmd_exit	25	38
fac_cmd_reset_single	24	35
fac_cmd_reset_multi	24	
fac_cmd_reset_all	24	
fac_cmd_elctimer	25	36
fac_cmd_elcstop	25	37
fac_trans_req	25	38
fac_trans_e2prom	26	
fac_trans_timer	26	39
fac_elc_stop	26	39
callback_req_result	26	40
callback_e2prom_result	27	
callback_elctimer_result	27	40

4.11.3 サンプルプログラム関数仕様

(1) hal_entry

hal_entry	
概　要	FA-CODER 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	FA-CODER 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	FA-CODER 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	FA-CODER サンプルプログラムのメイン関数です。詳細は、「4.11.5(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（FA-CODER I/F ドライバのエラーコード）

(3) fac_get_cmd

fac_get_cmd		
概　要	コマンドの取得関数	
ヘッダ	-	
宣　言	static uint32_t fac_get_cmd(char_t *p_arg[], const uint32_t arg_max);	
説　明	入力されたコマンドの取得を行います。	
引　数	p_arg	：取得したコマンドを格納する RAM の先頭アドレス
	arg_max	：引数の最大個数
リターン値	：引数の個数	

(4) fac_cmd_single

fac_cmd_single		
概　要	シングルコマンドの実行関数	
ヘッダ	-	
宣　言	static void fac_cmd_single(char_t *p_arg[], const uint32_t arg_num);	
説　明	シングルコマンドを実行します。	
引　数	p_arg	：未使用
	arg_num	：引数の個数
リターン値	なし	

(5) fac_cmd_multi

fac_cmd_multi		
概　要	マルチコマンドの実行関数	
ヘッダ	-	
宣　言	static void fac_cmd_multi(char_t *p_arg[], const uint32_t arg_num);	
説　明	マルチコマンドを実行します。	
引　数	p_arg	: 未使用
	arg_num	: 引数の個数
リターン値	なし	

(6) fac_cmd_encid

fac_cmd_encid		
概　要	エンコーダの ID 取得コマンドの実行関数	
ヘッダ	-	
宣　言	static void fac_cmd_encid(char_t *p_arg[], const uint32_t arg_num);	
説　明	エンコーダの ID 取得コマンドを実行します。	
引　数	p_arg	: 未使用
	arg_num	: 引数の個数
リターン値	なし	

(7) fac_cmd_req

fac_cmd_req		
概　要	リクエストコマンドの実行関数	
ヘッダ	-	
宣　言	static void fac_cmd_req(char_t *p_arg[], const uint32_t arg_num);	
説　明	リクエストコマンドを実行します。	
引　数	p_arg	: コマンドの引数が格納された配列の先頭アドレス
	arg_num	: 引数の個数
リターン値	なし	

(8) fac_cmd_e2prom_write

fac_cmd_e2prom_write		
概　要	E2PROM の書き込みコマンドの実行関数	
ヘッダ	-	
宣　言	static void fac_cmd_e2prom_write(char_t *p_arg[], const uint32_t arg_num);	
説　明	E2PROM の書き込みコマンドを実行します。	
引　数	p_arg	: コマンドの引数が格納された配列の先頭アドレス
	arg_num	: 引数の個数
リターン値	なし	

(9) fac_cmd_e2prom_read

fac_cmd_e2prom_read		
概　要	E2PROM の読み出しコマンドの実行関数	
ヘッダ	-	
宣　言	static void fac_cmd_e2prom_read(char_t *p_arg[], const uint32_t arg_num);	
説　明	E2PROM の読み出しコマンドを実行します。	
引　数	p_arg	：コマンドの引数が格納された配列の先頭アドレスを指定します。
	arg_num	：引数の個数
リターン値	なし	

(10) fac_cmd_reset_single

fac_cmd_reset_single		
概　要	１回転データリセットコマンドの実行関数	
ヘッダ	-	
宣　言	static void fac_cmd_reset_single(char_t *p_arg[], const uint32_t arg_num);	
説　明	１回転データリセットコマンドを実行します。	
引　数	p_arg	：未使用
	arg_num	：引数の個数
リターン値	なし	

(11) fac_cmd_reset_multi

fac_cmd_reset_multi		
概　要	多回転データおよびオールエラーリセットコマンドの実行関数	
ヘッダ	-	
宣　言	static void fac_cmd_reset_multi(char_t *p_arg[], const uint32_t arg_num);	
説　明	多回転データおよびオールエラーリセットコマンドを実行します。	
引　数	p_arg	：未使用
	arg_num	：引数の個数
リターン値	なし	

(12) fac_cmd_reset_all

fac_cmd_reset_all		
概　要	オールエラーリセットコマンドの実行関数	
ヘッダ	-	
宣　言	static void fac_cmd_reset_all(char_t *p_arg[], const uint32_t arg_num);	
説　明	オールエラーリセットコマンドを実行します。	
引　数	p_arg	：未使用
	arg_num	：引数の個数
リターン値	なし	

(13) fac_cmd_elctimer

fac_cmd_elctimer		
概　要	ELC タイマコマンドの実行関数	
ヘッダ	-	
宣　言	static void fac_cmd_elctimer(char_t *p_arg[], const uint32_t arg_num);	
説　明	ELC タイマコマンドを実行するための関数です。	
引　数	p_arg	：コマンドの引数が格納された配列の先頭アドレスを指定します。
	arg_num	：引数の個数
リターン値	なし	

(14) fac_cmd_elcstop

fac_cmd_elcstop		
概　要	ELC ストップコマンドの実行関数	
ヘッダ	-	
宣　言	static void fac_cmd_elcstop(char_t *p_arg[], const uint32_t arg_num);	
説　明	ELC ストップコマンドを実行するための関数です。	
引　数	p_arg	：未使用
	arg_num	：引数の個数
リターン値	なし	

(15) fac_cmd_exit

fac_cmd_exit		
概　要	終了コマンドの実行関数	
ヘッダ	-	
宣　言	static void fac_cmd_exit(char_t *p_arg[], const uint32_t arg_num);	
説　明	終了コマンドを実行します。	
引　数	p_arg	：未使用
	arg_num	：引数の個数
リターン値	なし	

(16) fac_trans_req

fac_trans_req		
概　要	コマンドの転送要求関数	
ヘッダ	-	
宣　言	static r_fac_err_t fac_trans_req(r_fac_req_t *const p_req);	
説　明	コマンドの転送要求を行います。E2PROM コマンド以外は本関数を使用します。	
引　数	p_req	：転送要求データの先頭アドレス
リターン値	err_code	：エラーコード

(17) fac_trans_e2prom

fac_trans_e2prom		
概　要	E2PROM コマンドの転送要求関数	
ヘッダ	-	
宣　言	static r_fac_err_t fac_trans_e2prom(r_fac_e2prom_data_t *const p_e2prom_data);	
説　明	E2PROM コマンドの転送要求を行います。	
引　数	p_e2prom_data	: E2PROM 転送要求データの先頭アドレス。
リターン値	err_code	: エラーコード

(18) fac_trans_timer

fac_trans_timer		
概　要	ELC タイマコマンドの転送要求関数	
ヘッダ	-	
宣　言	static r_fac_err_t fac_trans_timer(r_fac_req_t *const p_req);	
説　明	ELC タイマコマンドの転送要求を行います。	
引　数	p_req	: ELC タイマ転送要求データの先頭アドレス。
リターン値	err_code	: エラーコード

(19) fac_elc_stop

fac_elc_stop		
概　要	ELC タイマコマンドの停止関数	
ヘッダ	-	
宣　言	static r_fac_err_t fac_elc_stop(void);	
説　明	ELC タイマコマンドの転送要求を停止します。	
引　数	なし	
リターン値	err_code	: エラーコード

(20) callback_req_result

callback_req_result		
概　要	FA-CODER 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 callback_req_result(r_fac_result_t *p_result, uint8_t *p_rxdf);	
説　明	結果をメモリに保存し、取得完了フラグを立てます。	
引　数	p_result	: 通信結果の先頭アドレス 「4.10.1(4) r_fac_result_t」参照
	p_rxdf	: 受信データの先頭アドレス
リターン値	なし	

(21) callback_e2prom_result

callback_e2prom_result		
概　要	FA-CODER の E2PROM との送受信結果通知のコールバック関数	
ヘッダ	-	
宣　言	static void callback_e2prom_result(r_fac_result_t *p_result, uint8_t adf, uint8_t edf);	
説　明	結果をメモリに保存し、取得完了フラグを立てます。	
引　数	p_result	: 通信結果の先頭アドレス「4.10.1(4) r_fac_result_t」参照
	adf	: アドレスフィールドの値
	edf	: E2PROM フィールドの値
リターン値	なし	

(22) callback_elctimer_result

callback_elctimer_result		
概　要	FA-CODER 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 callback_elctimer_result(r_fac_result_t *p_result, uint8_t *p_rxdf);	
説　明	結果をメモリに保存し、カウンタを更新します。	
引　数	p_result	: 通信結果の先頭アドレス「4.10.1(4) r_fac_result_t」参照
	p_rxdf	: 受信データの先頭アドレス
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 4.7 に static 型変数を示します。const 型は使用しません。

表 4.7　static 型変数

型	変数名	内容	使用関数
bool	fac_done	結果取得完了フラグ 初期値：false	fac_trans_req fac_trans_e2prom fac_trans_timer callback_req_result callback_e2prom_result
r_fac_result_t*	p_fac_result	データ取得結果の先頭アドレス	fac_cmd_single fac_cmd_multi fac_cmd_encid fac_cmd_req fac_cmd_e2prom_write fac_cmd_e2prom_read fac_cmd_reset_single fac_cmd_reset_multi fac_cmd_reset_all callback_req_result callback_e2prom_result
uint8_t*	p_fac_rxdf	取得データの先頭アドレス	fac_cmd_single fac_cmd_multi fac_cmd_encid fac_cmd_req callback_req_result
uint8_t	fac_adf	アドレスフィールドの値	fac_cmd_e2prom_write fac_cmd_e2prom_read callback_e2prom_result
uint8_t	fac_edf	E2PROM フィールドの値	fac_cmd_e2prom_write fac_cmd_e2prom_read callback_e2prom_result
bool	fac_elc_flg	タイマイベント入力動作フラグ 初期値：false	fac_cmd_elctimer fac_cmd_elcstop
r_fac_result_t	fac_ti_result[]	タイマイベント入力動作時用 データ取得結果リングバッファ	fac_cmd_elcstop callback_elctimer_result
uint32_t	fac_ti_single_turn[]	タイマイベント入力動作時用 取得データリングバッファ	fac_cmd_elcstop callback_elctimer_result
uint32_t	fac_ti_count	リングバッファ格納位置カウンタ	fac_cmd_elctimer fac_cmd_elcstop callback_elctimer_result
uint32_t	fac_ti_valid	リングバッファ有効データ数	fac_cmd_elctimer fac_cmd_elcstop callback_elctimer_result
bool	fac_ti_full	リングバッファ full フラグ	fac_cmd_elctimer callback_elctimer_result

4.11.5 メイン処理のフローチャート

(1) enc_main フローチャート

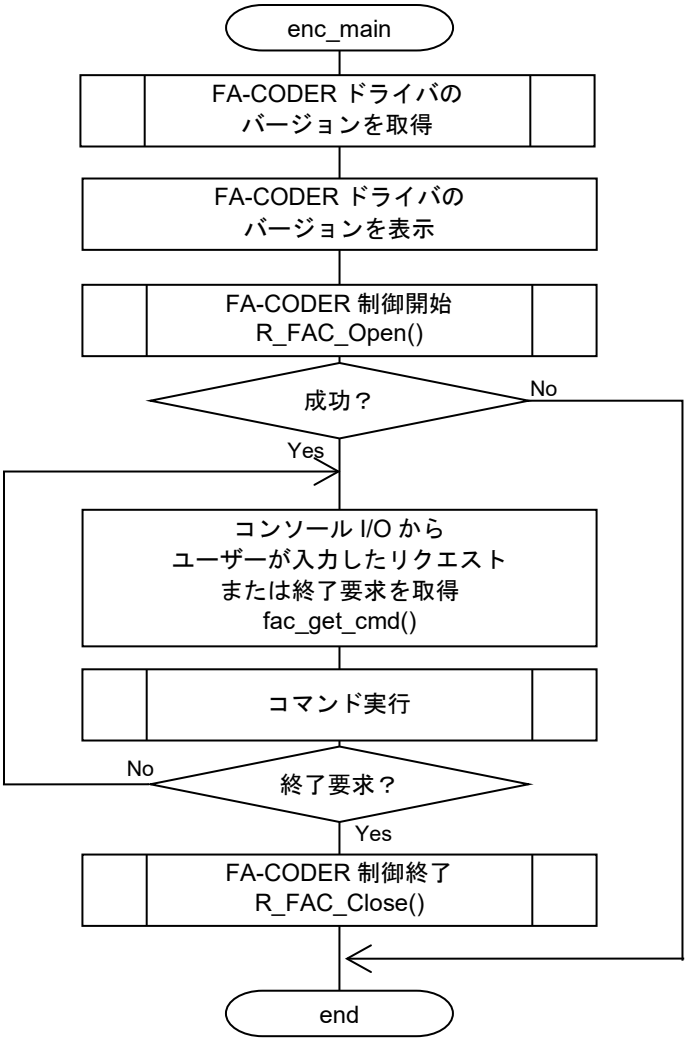


図 4-3 enc_main 関数のフローチャート

(2) fac_get_cmd フローチャート

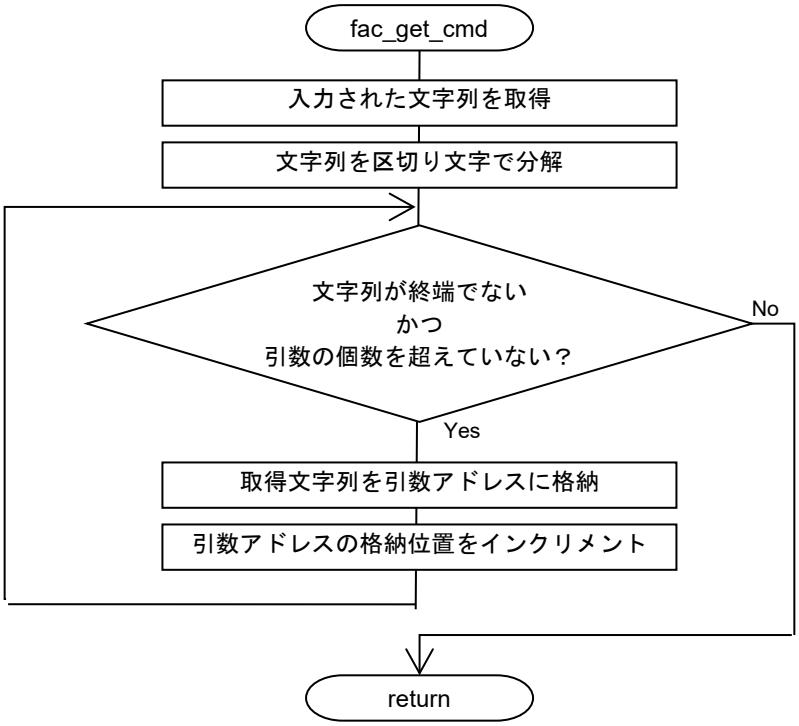


図 4-4 fac_get_cmd 関数のフローチャート

(3) fac_cmd_xxx フローチャート

4.11.3(4) fac_cmd_single～4.11.3(6) fac_cmd_encid の関数のフローチャートを本章に記載します。

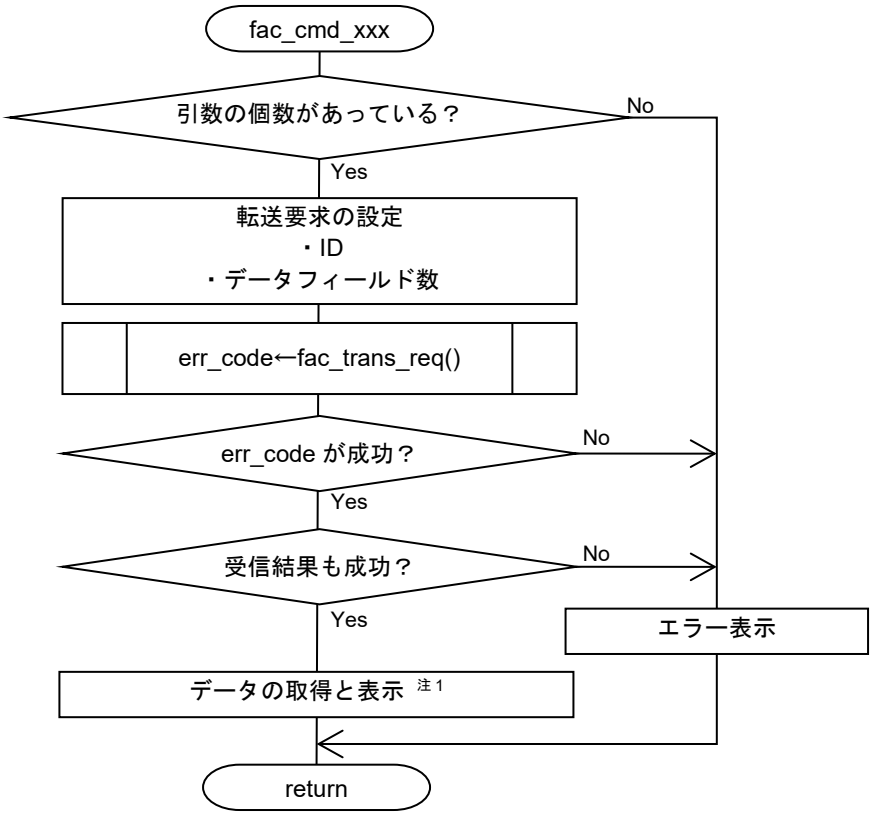


図 4-5 fac_cmd_xxx 関数のフローチャート

【注】 1. シングルコマンドは 1 回転内の絶対値データを、マルチコマンドは多回転データを、エンコード ID コマンドはエンコード ID を取得し表示します。

(4) fac_cmd_req フローチャート

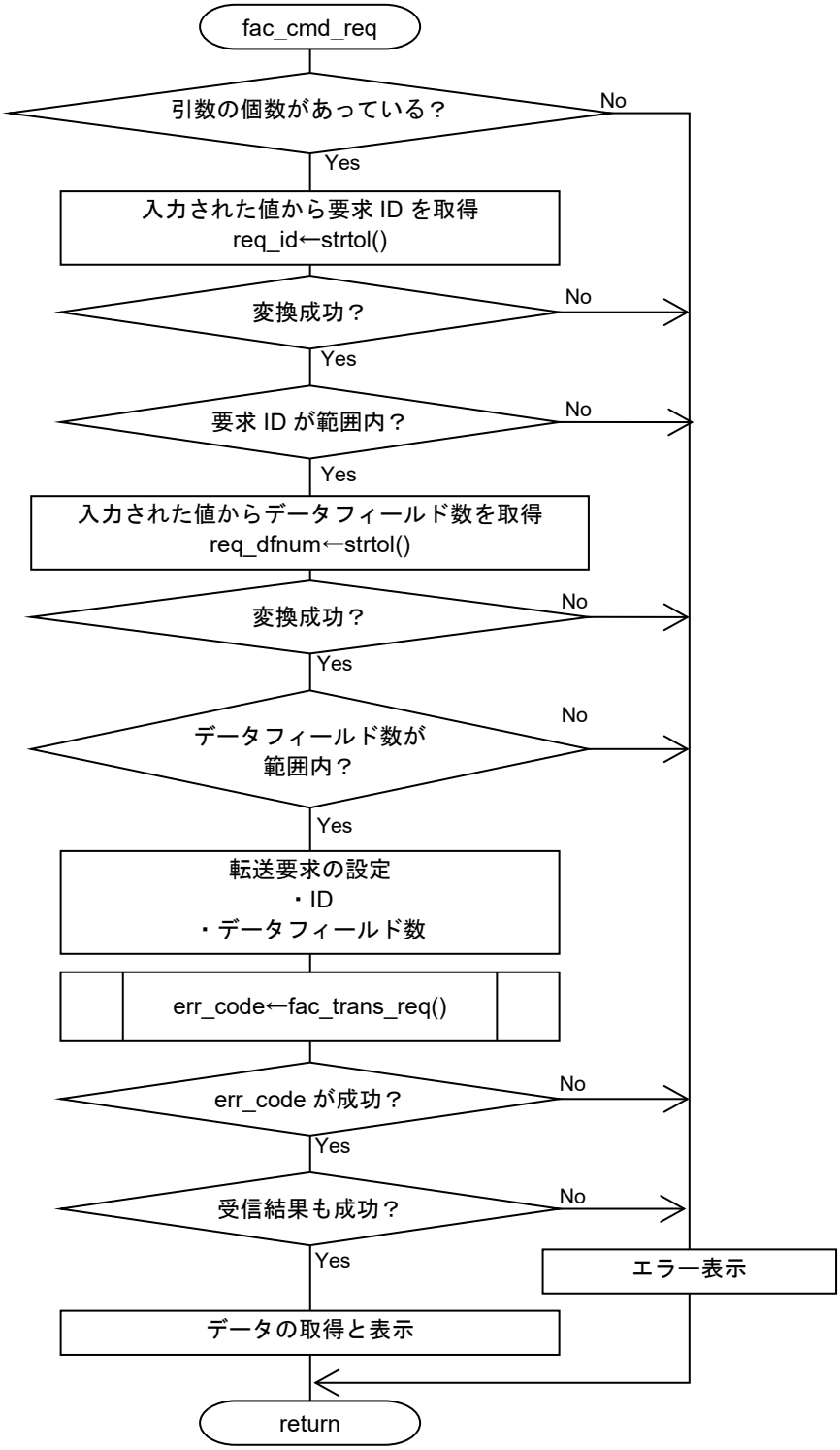


図 4-6 fac_cmd_req 関数のフローチャート

(5) fac_cmd_e2prom_write フローチャート

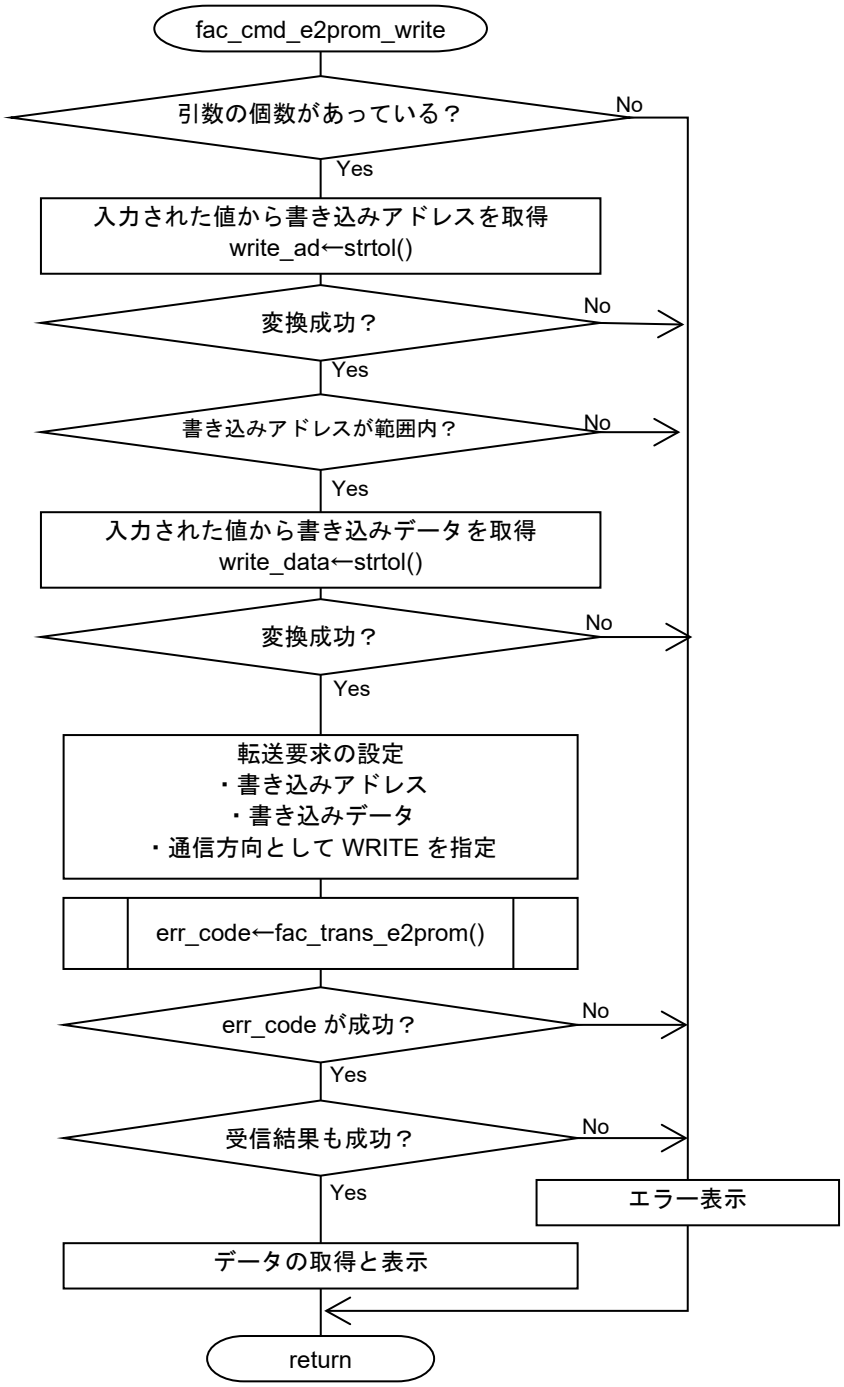


図 4-7 fac_cmd_e2prom_write 関数のフローチャート

(6) fac_cmd_e2prom_read フローチャート

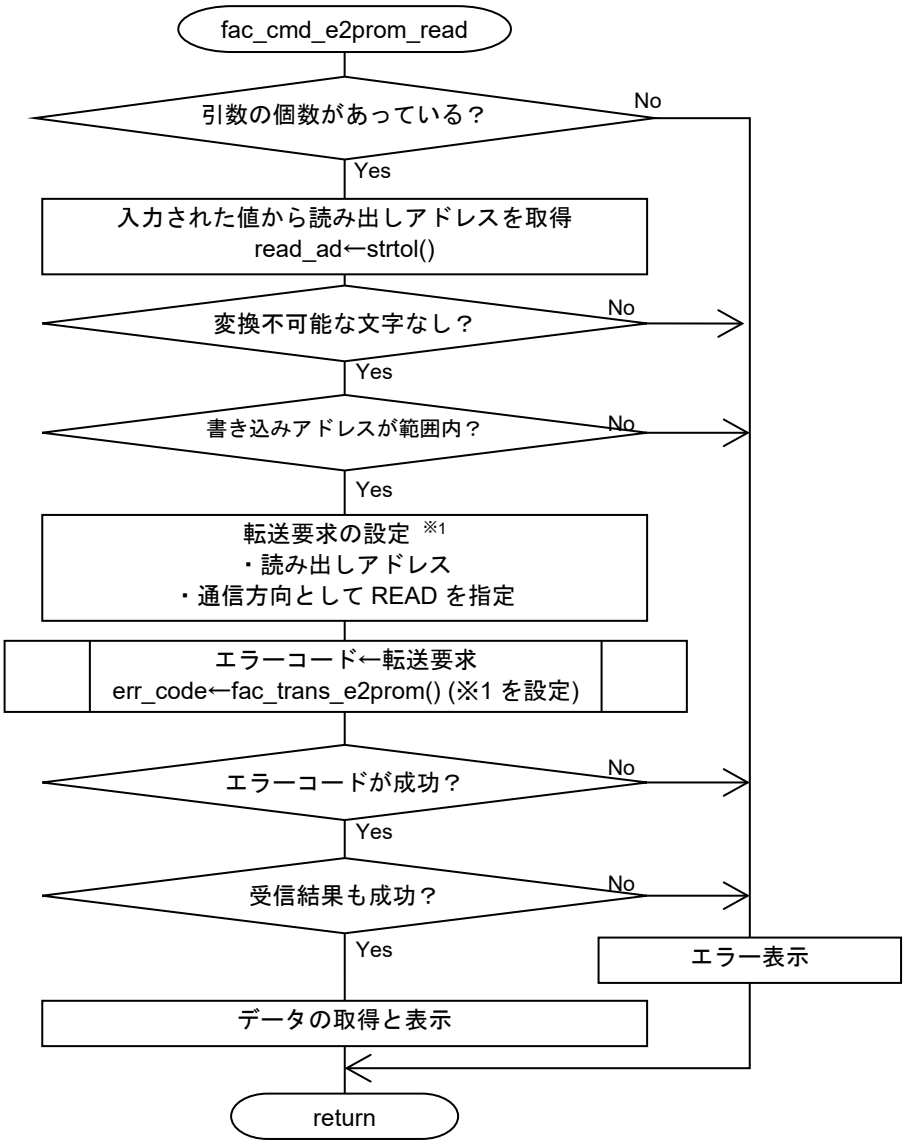


図 4-8 fac_cmd_e2prom_read 関数のフローチャート

(7) fac_cmd_reset_xxx フローチャート

4.11.3(10) fac_cmd_reset_single～4.11.3(12) fac_cmd_reset_all の関数のフローチャートを本章に記載します。

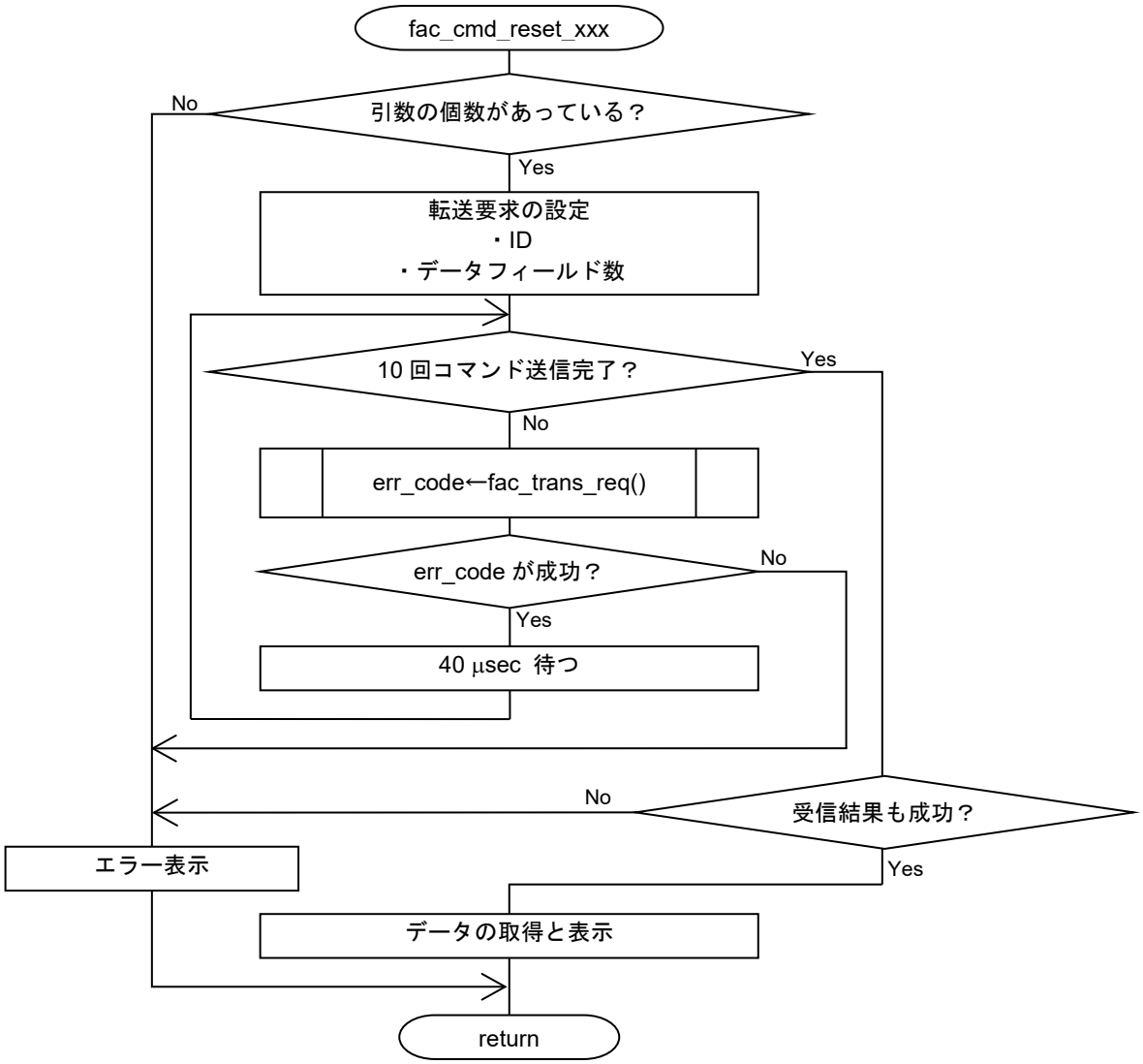


図 4-9 fac_cmd_reset_xxx 関数のフローチャート

(8) fac_cmd_elctimer フローチャート

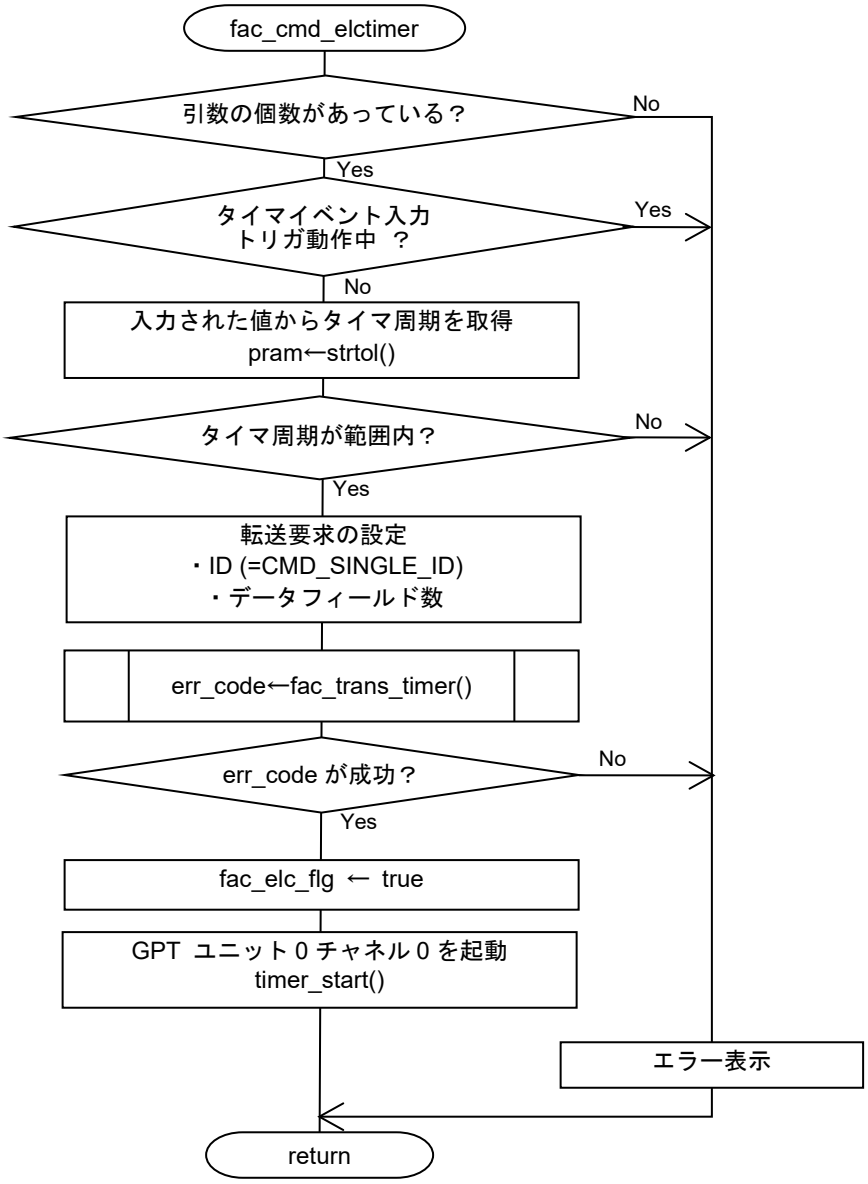


図 4-10 fac_cmd_elctimer 関数のフローチャート

(9) fac_cmd_elcstop フローチャート

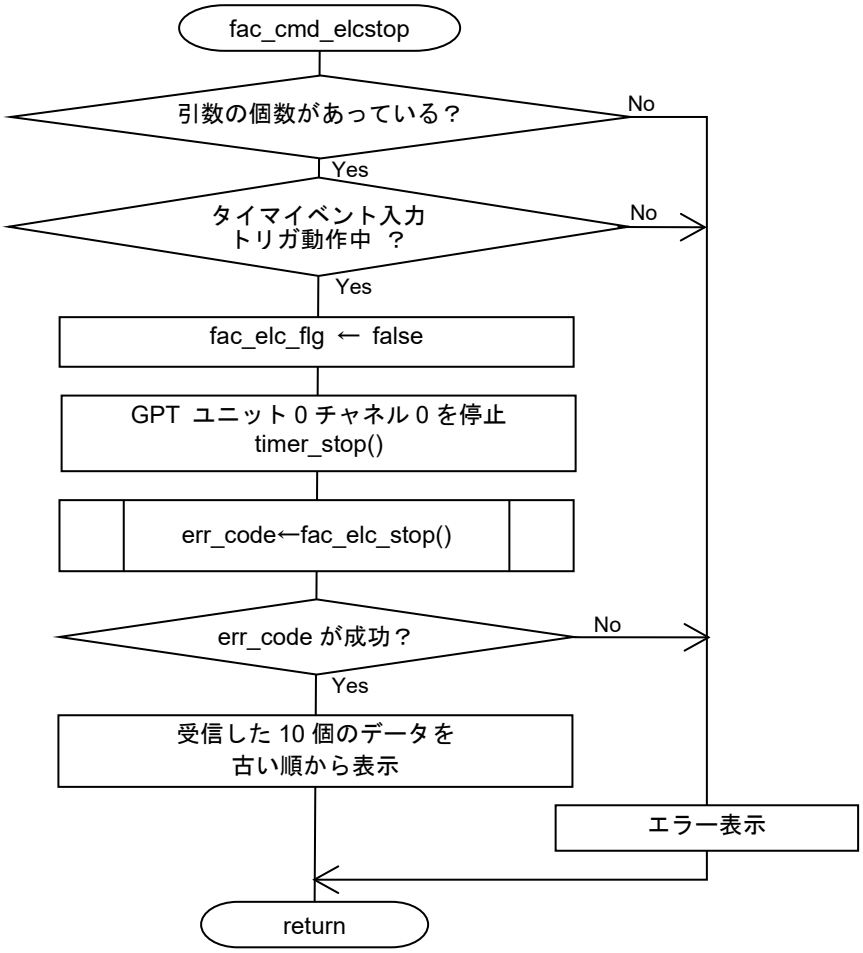


図 4-11 fac_cmd_elcstop 関数のフローチャート

(10) fac_cmd_exit フローチャート

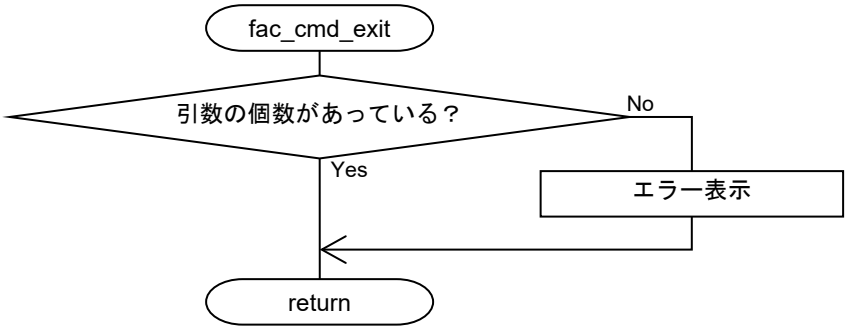


図 4-12　fac_cmd_exit 関数のフローチャート

(11) fac_trans_xxx フローチャート

4.11.3(16) fac_trans_req、4.11.3(17) fac_trans_e2prom の関数は本章のフローチャートに記載します。

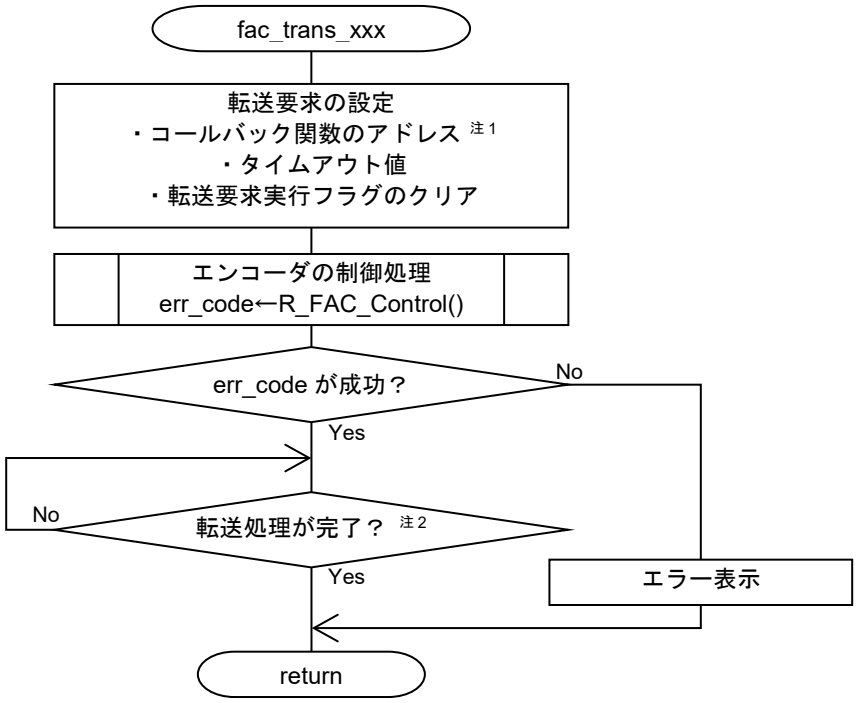


図 4-13　fac_trans_xxx 関数のフローチャート

【注】 1. fac_trans_req 関数では callback_req_result を fac_cmd_e2prom_write と fac_cmd_e2prom_read 関数では callback_e2prom_result を設定します。
2. コールバック関数内で、転送処理の完了が設定されます。

(12) fac_trans_timer フローチャート

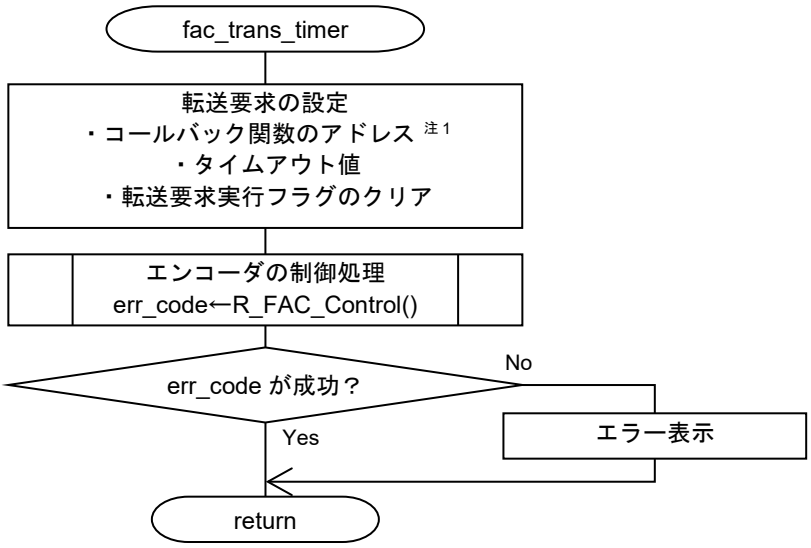


図 4-14 fac_trans_timer 関数のフローチャート

【注】 1. コールバック関数として、callback_elctimer_result を設定します。

(13) fac_elc_stop フローチャート

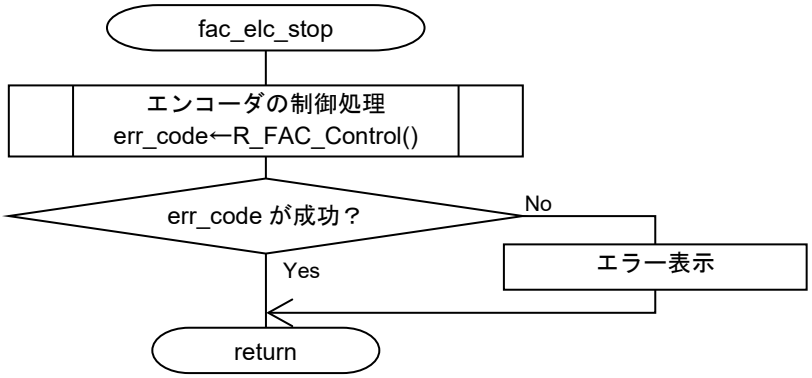


図 4-15 fac_elc_stop 関数のフローチャート

(14) callback_xxx_result フローチャート

4.11.3(20) callback_req_result 、4.11.3(21) callback_e2prom_result の関数のフローチャートを本章に記載します。

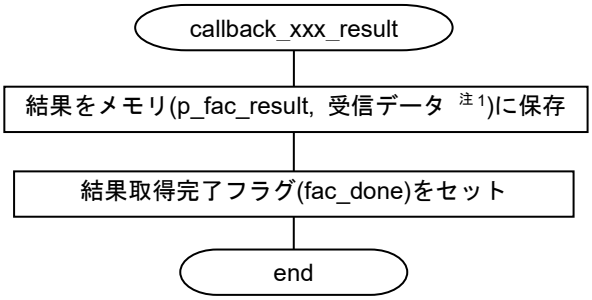


図 4-16 callback_xxx_result 関数のフローチャート

【注】 1. callback_req_result 関数は p_fac_rxdf（受信データフィールド）を、callback_e2prom_result 関数は fac_adf（ADF レジスタ）と fac_edf（EDF レジスタ）のデータを保存します。

(15) callback_elctimer_result フローチャート

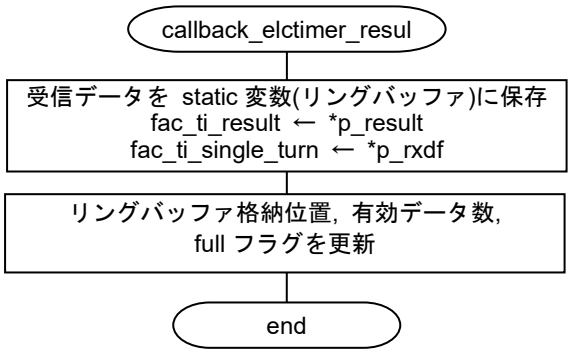


図 4-17 callback_elctimer_result 関数のフローチャート

4.11.6 動作シーケンス

(1) 開始シーケンス

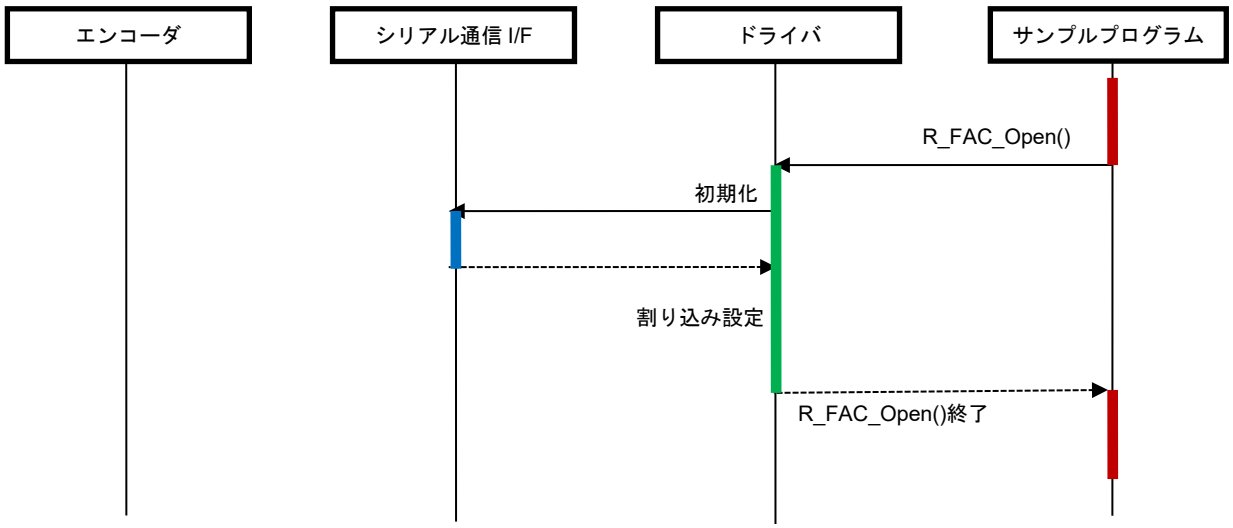


図 4-18 開始シーケンス図

(2) リクエスト・データ受信シーケンス

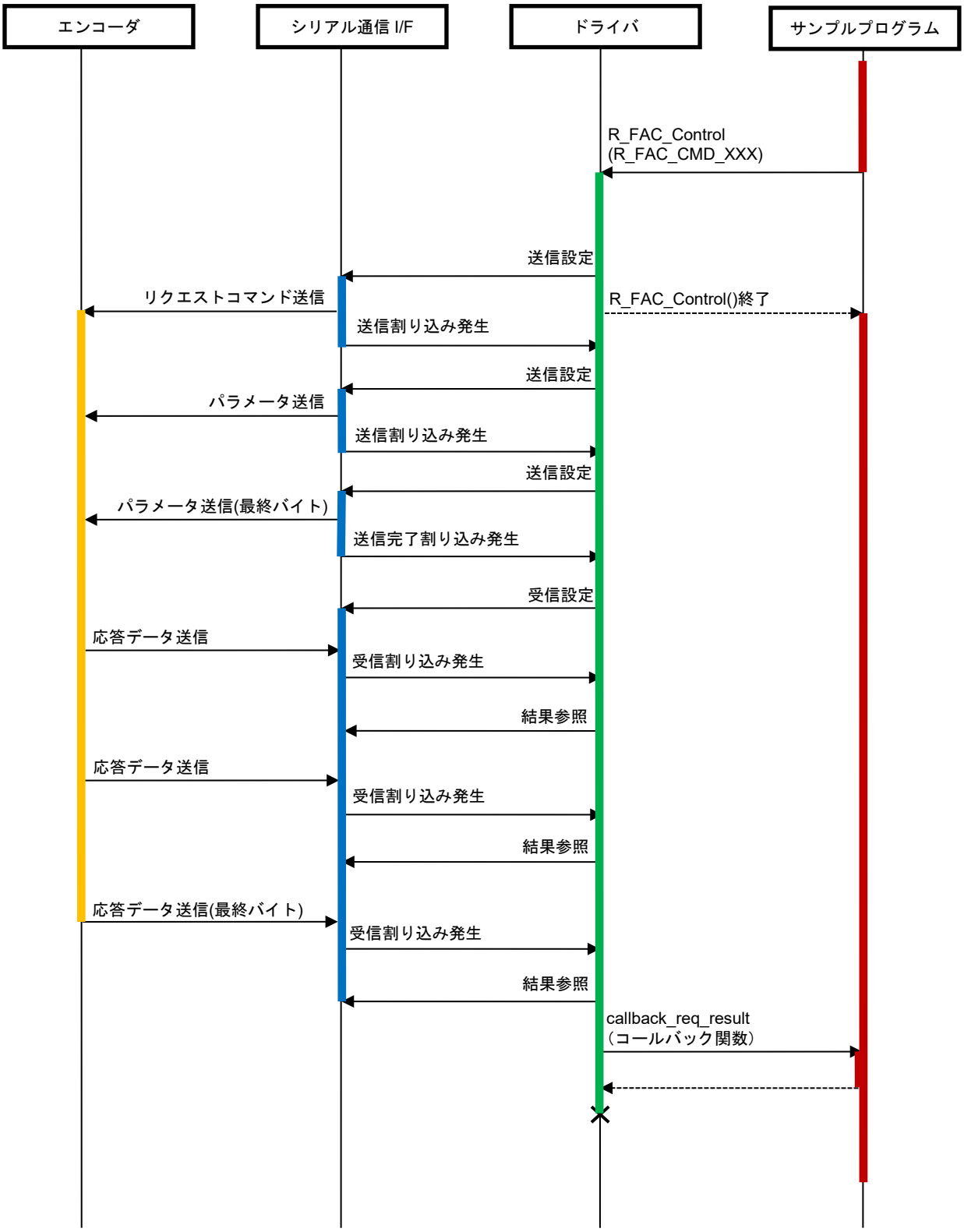
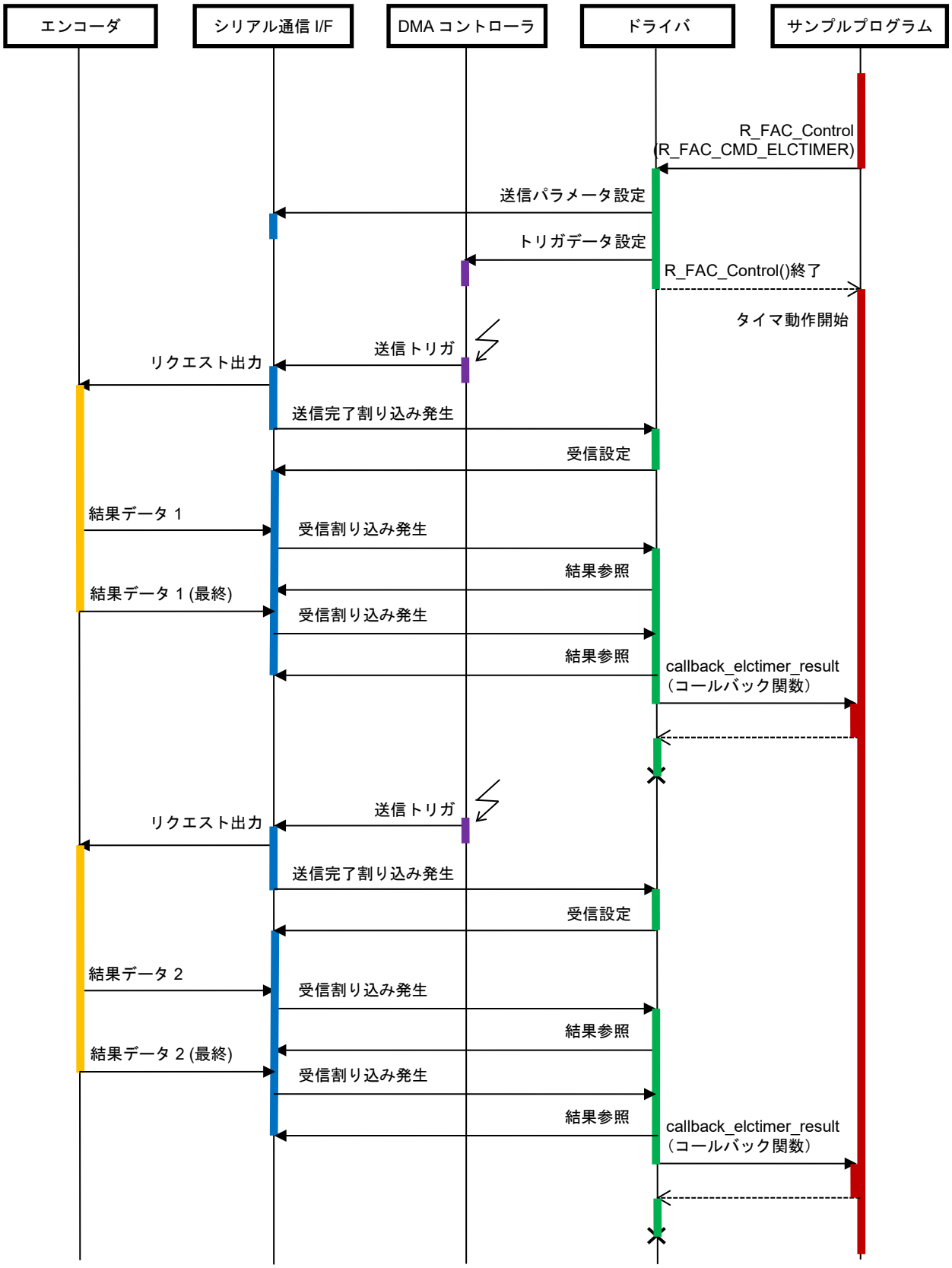


図 4-19 リクエスト・データ受信シーケンス図

(3) タイマイベント入力トリガ動作のシーケンス



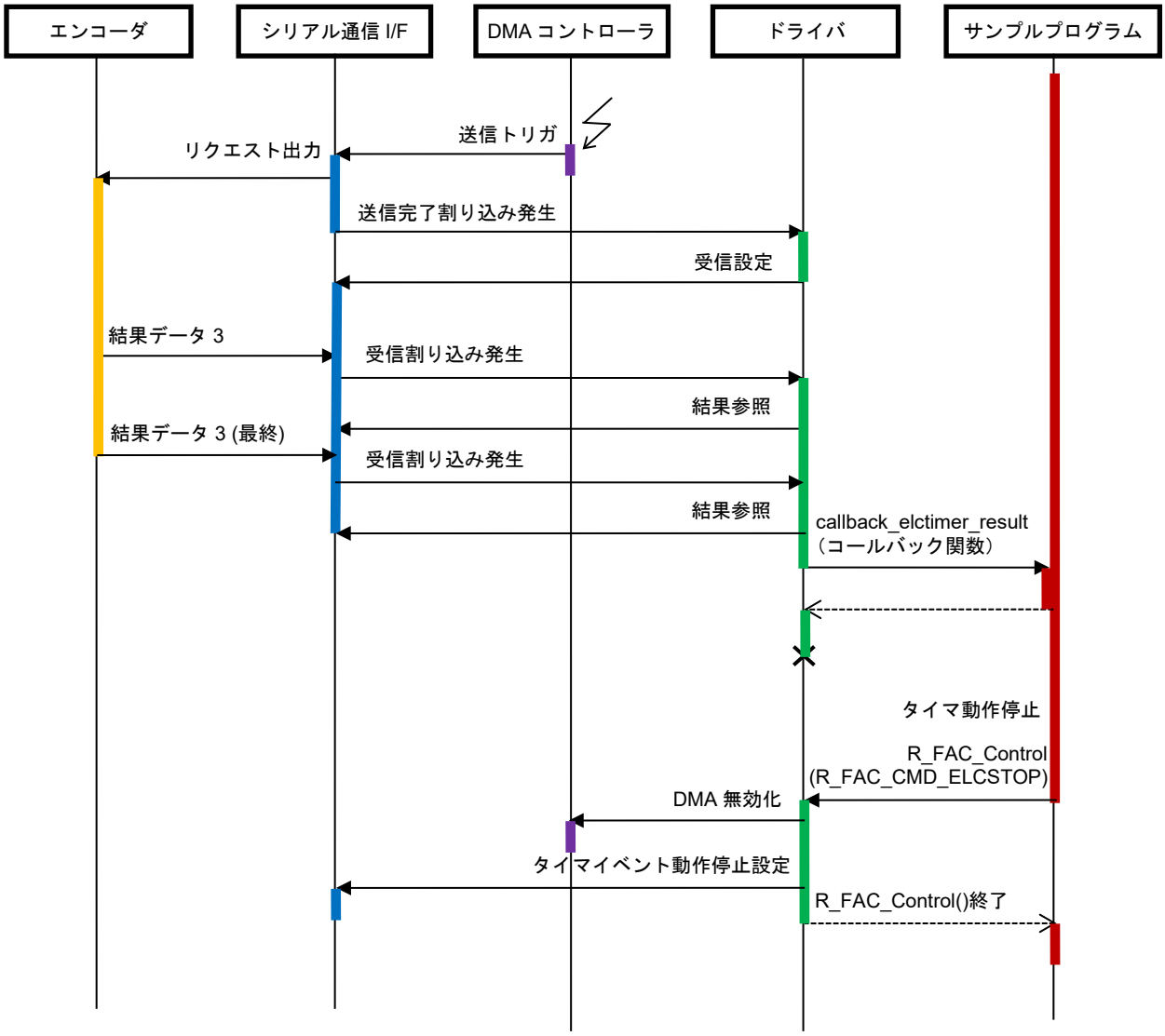


図 4-20　タイマイベント入力トリガ動作のシーケンス図

(4) 停止シーケンス

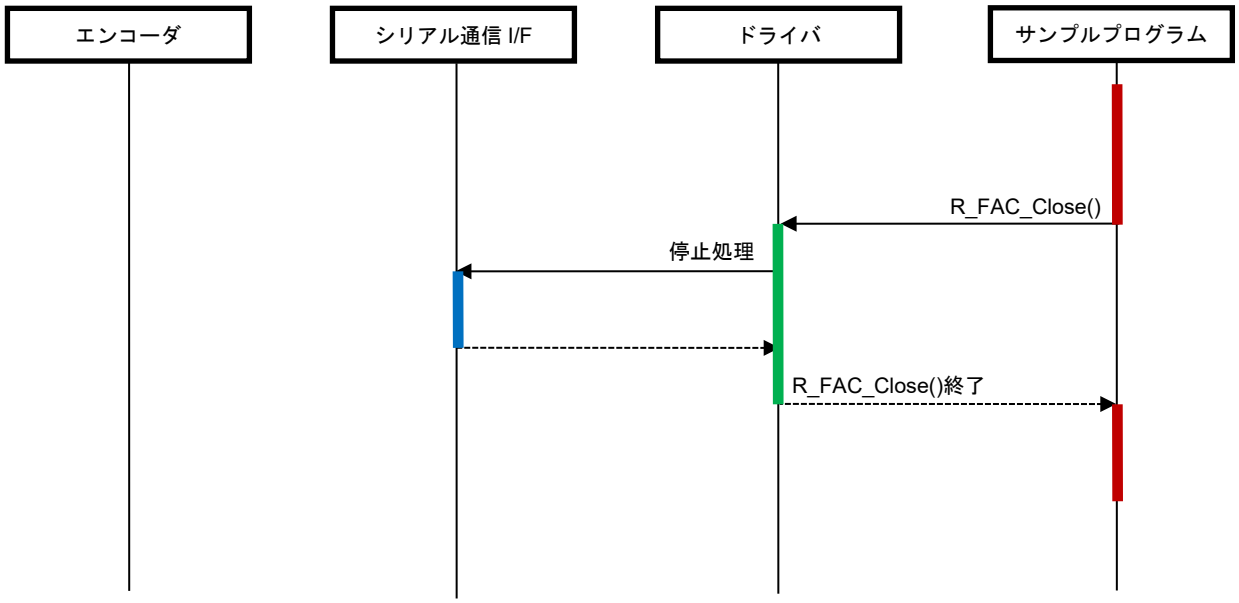


図 4-21 停止シーケンス図

4.11.7 コンソールコマンド

本サンプルプログラムは多摩川精機製位置エンコーダ（FA-CODER）「TS5667N120」「TS5702N142」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となります。

表 4.8　コンソールコマンド一覧

コマンド	内容
single	シングルターンデータ取得 注1
multi	マルチターンデータ取得 注1
encid	エンコーダ ID 取得 注1
write *x y*	データ書き込み 注1 書き込みアドレス *x* とデータ *y*（0～255）を指定して実行 注2
read *x*	データ読み込み 注1 読み出しアドレス *x* を指定して実行 注2
req *x y*	リクエスト実行 リクエスト ID *x* とデータフィールド数 *y* を指定してリクエストを実行する 具体的なリクエスト ID およびデータフィールド数の設定値については、「表 4.9 req コマンド引数一覧」参照
reset_single	１回転データリセット 注1
reset_multi	多回転データおよびオールエラーリセット 注1
reset_all	オールエラーリセット 注1
elctimer *val*	タイマイベント入力トリガ動作として、シングルターンデータをタイマ周期で取得 注1 タイマ周期 *val* を us 単位（最大 6990us）で指定して実行
elcstop	タイマイベント入力トリガ動作を終了して取得データ表示
exit	プログラム終了

【注】 1. エンコーダ「TS5667N120」「TS5702N142」用コマンド
2. 書き込み/読み出しアドレス x の有効範囲は、エンコーダが「TS5667N120」のとき 0～79、「TS5702N142」のとき 0～127

表 4.9　req コマンド引数一覧

リクエスト ID	データフィールド数	内容
0	3	シングルターンデータ取得 注1
1	3	マルチターンデータ取得 注1
2	1	エンコーダ ID 取得 注1
3	8	シングルターンデータ，エンコーダ ID，マルチターンデータ，エンコーダエラー情報取得 注1
7	3	オールエラーリセット 注1
8	3	１回転データリセット 注1
C	3	多回転データおよびオールエラーリセット 注1

【注】 1. エンコーダ「TS5667N120」「TS5702N142」用コマンド

(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。”tamagawa >”に続けてコマンドを入力してください。

```
Tama sample program start
R_FAC_GetVersion = 4.0


tamagawa >
```

(2) コマンド実行例

single 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、シングルターンデータ、リクエスト ID やアラーム情報などが表示されます。

```
tamagawa >single
single command
  result:success
    single turn data:             764010
    request id:                    0H
    parity bit(request id):        0H
    information code:              0H
    encoder alarm:                 2H
    communication alarm:           0H
    crc data:                     EBH
tamagawa >
```

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	Mar 31.23	-	初版発行
1.10	Jun.30.23	8, 14, 16, 19, 20 42	R_FAC_Open()関数の引数に p_info を追加 表 4.4 ビットレート指定を追加 r_fac_info_t 構造体の記載を追加 図の参照番号誤りを修正 表 4.8 コンソールコマンドの注記を更新
1.20	Mar.01.24	4, 4, 6 7, 11, 13, 20 16, 17, 21, 23, 27 - 30, 38, 39, 41, 42 45, 46, 48	タイマイベント同期リクエストに対応して、表 1.1 を更新 FA-CODER のエンコーダ I/F に使用する SCI チャネルを 3 と 4 に変更 タイマイベント同期リクエストに関連するコマンドおよび、コールバック関数の記載を追加 使用する SCI チャネル変更に伴い、表 4.2 を更新 固定幅整数に関する説明を修正 動作概要, システムブロック図を更新 タイマイベント同期リクエストに関連する関数および、変数の説明を更新 fac_cmd_elctimer, fac_cmd_elcstop フローチャートの説明を更新、fac_elc_stop, callback_fac_elctimer_result フローチャートを追加 タイマイベント入力トリガ動作のシーケンス図を追加 コンソールコマンドに elctimer, elcstop を追加
3.00	Oct 31.25	1, 4, 5 8 - 12 32 46 47	商標の説明の記載方法を更新 id 引数に関する説明を追加 fac_cmd_req フローチャートの誤記を修正 表 4.8 を修正し、req コマンド引数一覧 表 4.9 を追加 コマンド実行例を追加
4.00	Apr 10.26	7 - 40	ポインタ変数のプレフィクスを”p_”に変更 (例. pinfo -> p_info)
4.01	May 22.26	16 31 - 35	r_fac_e2prom_data_t 構造体の adr の型の誤記を修正 フローチャート内の関数名の誤りを修正 (fac_trans_reg → fac_trans_req, fac_trans_e2prom)

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。

・FA-CODER is a registered trademark of Tamagawa Seiki Co., Ltd.
・IAR Embedded Workbench is a registered trademark of IAR Systems.
・Arm and Cortex are registered trademarks of Arm Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved.
・Additionally all product names and service names in this document are a trademark or a registered trademark which belongs to the respective owners.

ご注意書き

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

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

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

(Rev.5.0-1　2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

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

www.renesas.com/contact/

商標について

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

© 2026 Renesas Electronics Corporation. All rights reserved.