
RZ/T2M グループ

SSI サンプルプログラム

要旨

本アプリケーションノートでは、RZ/T2M の Encoder I/F Configuration Library (以下 EC-Lib) を使用して、同期式シリアルインタフェース (Synchronized Serial Interface、以下 SSI) 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。

本サンプルプログラムは、RZ/T2M Encoder I/F Configuration Data (SSI) Ver.1.0 以降に対応しています。

プログラムの特徴を以下に示します。

- ・ SSI 仕様の通信方式に対応
- ・ SSI 仕様に準拠したエンコーダから、角度情報等を取得

対象デバイス

RZ/T2M

目次

1. 仕様	4
2. 動作環境	5
3. 周辺機能説明	6
3.1 使用端子一覧	6
4. ソフトウェア説明	7
4.1 SSI ドライバ機能	7
4.2 ファイル構成	7
4.3 関数一覧	7
4.4 API 関数仕様	8
4.4.1 R_SSI_Open	8
4.4.2 R_SSI_Close	8
4.4.3 R_SSI_GetVersion	9
4.4.4 R_SSI_GetFirmVersion	9
4.4.5 R_SSI_Control	9
4.4.6 R_SSI_GetStatus	13
4.4.7 R_SSI_GetRxd	13
4.5 ユーザー定義関数仕様	14
4.5.1 ssi_int_ssi_callback	14
4.5.2 ssi_int_fifo_callback	14
4.6 割り込みハンドラ	15
4.6.1 enc_ch0_int_ssi_isr	15
4.6.2 enc_ch0_int_fifo_isr	15
4.6.3 enc_ch1_int_ssi_isr	15
4.6.4 enc_ch1_int_fifo_isr	15
4.7 使用割り込み一覧	16
4.8 定数/エラーコード一覧	16
4.9 固定幅整数一覧	18
4.10 構造体/共用体/列挙型一覧	19
4.10.1 構造体	19
4.10.2 共用体	21
4.10.3 列挙型	21
4.11 サンプルプログラムの説明	22
4.11.1 動作概要	22
4.11.2 サンプルプログラム関数一覧	24
4.11.3 サンプルプログラム関数仕様	25
4.11.4 サンプルプログラムの変数一覧	31
4.11.5 サンプルプログラムの定数一覧	31
4.11.6 メイン処理のフローチャート	32
4.11.7 動作シーケンス	42
4.11.8 コンソールコマンド	48
5. サンプルコード	50

改訂記録51

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
SSI 通信コントローラ (SSI)	SSI 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)	SSI 割り込み制御
汎用 PWM タイマ (GPT)	連続処理時の通信周期の生成、および、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 チャンネル 1 が出力するイベントと SSI 通信コントローラをリンク
シリアル通信インタフェース (SCI) UART	SCI の調歩同期式 I/F を使用し、USB インタフェースによる COM ポート通信に使用 サンプルプログラムのコンソールインタフェース用

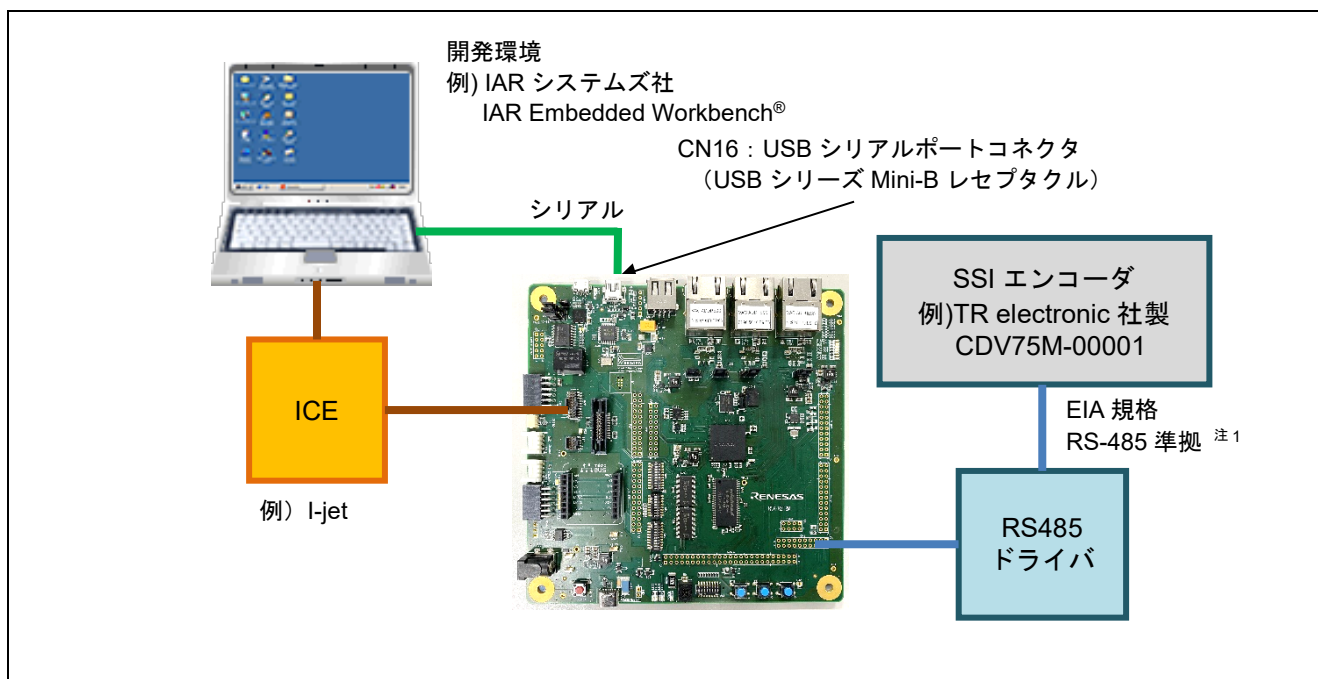


図 1-1 動作環境

【注】 1. 送受信可能なケーブル長は、エンコーダの製造元に問い合わせてください。

2. 動作環境

本アプリケーションノートのサンプルコードは、下記の環境を想定しています。

表 2.1 動作環境

項目	内容
使用マイコン	RZ/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 SSI sample program リリースノート
を参照してください。

3. 周辺機能説明

周辺機能、動作モード、レジスタについての基本的な内容は、RZ/T2M グループ・ユーザーズマニュアルハードウェア編を参照してください。

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

チャンネル	端子名 (機能ピン名)	I/O ポート	入出力	内容
SSI0	RX0 (ENCIF0)	P01_6	入力	データ入力
	CLK0 (ENCIF4)	P02_3	出力	クロック出力
SSI1	RX1 (ENCIF5)	P17_3	入力	データ入力
	CLK1 (ENCIF9)	P03_3	出力	クロック出力

4. ソフトウェア説明

4.1 SSI ドライバ機能

SSI ドライバの機能は以下です。

1. 初期設定
2. 受信データの取得
3. 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/T2M グループ Encoder I/F SSI application 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
SSI I/F ドライバ API 関数	R_SSI_Open	8
	R_SSI_Close	8
	R_SSI_GetVersion	9
	R_SSI_GetFirmVersion	9
	R_SSI_Control	9
	R_SSI_GetStatus	13
	R_SSI_GetRxd	13
ユーザー定義関数	ssi_int_ssi_callback	14
	ssi_int_fifo_callback	14
割り込みハンドラ	enc_ch0_int_ssi_isr	15
	enc_ch0_int_fifo_isr	15
	enc_ch1_int_ssi_isr	15
	enc_ch1_int_fifo_isr	15

4.4 API 関数仕様

4.4.1 R_SSI_Open

R_SSI_Open	
概要	エンコーダ制御の開始
ヘッダ	r_ssi_rzt2_if.h
宣言	int32_t R_SSI_Open(const int32_t id);
説明	引数で指定された SSI 通信チャンネルの開始処理を行います。 1. 通信パラメータの初期設定 2. ELC 設定の無効化 3. FIFO 初期化 4. 割り込みコントローラへの登録 5. 割り込みマスク (MASK レジスタ) の設定
引数	id : 使用する ID を指定します。(r_ssi_rzt2_dat.h で定義されています。) R_SSI0_ID : チャンネル 0 を指定 R_SSI1_ID : チャンネル 1 を指定 上記以外 : 設定不可
リターン値	R_SSI_SUCCESS : 正常終了 R_SSI_ERR_ACCESS : 異常終了 (既に open されています) R_SSI_ERR_INVALID_ARG : 異常終了 (引数 id が規定されていない値)
注意	本関数実行前に、必ず EC-Lib を用いて SSI Encoder I/F のコンフィギュレーションと起動を行ってください。 本関数内でエンコーダ I/F の設定を行います。 コールバック関数内で、本 API 関数を実行することは禁止します。

4.4.2 R_SSI_Close

R_SSI_Close	
概要	エンコーダ制御を終了
ヘッダ	r_ssi_rzt2_if.h
宣言	int32_t R_SSI_Close(const int32_t id);
説明	引数で指定された SSI 通信チャンネルの制御を終了します。
引数	id : 使用する ID を指定します。(r_ssi_rzt2_dat.h で定義されています。) R_SSI0_ID : チャンネル 0 を指定 R_SSI1_ID : チャンネル 1 を指定 上記以外 : 設定不可
リターン値	R_SSI_SUCCESS : 正常終了 R_SSI_ERR_ACCESS : 異常終了 (通信中または ELC 受付中です。) R_SSI_ERR_INVALID_ARG : 異常終了 (引数 id が規定されていない値)
注意	エンコーダと送受信中の場合は Close できません。 コールバック関数内で、本 API 関数を実行することは禁止します。

4.4.3 R_SSI_GetVersion

R_SSI_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_ssi_rzt2_if.h
宣言	uint32_t R_SSI_GetVersion();
説明	SSI ドライバのバージョンを取得します。
引数	なし
リターン値	上位 16 ビットにメジャーバージョン、下位 16 ビットにマイナーバージョンが格納されます。 例) 戻り値が 0x00010002 の場合、Ver.1.2
補足	
注意	

4.4.4 R_SSI_GetFirmVersion

R_SSI_GetFirmVersion

概要	SSI 通信コントローラのバージョンを取得
ヘッダ	r_ssi_rzt2_if.h
宣言	uint32_t R_SSI_GetFirmVersion(const int32_t id);
説明	引数で指定された SSI 通信コントローラのバージョンを取得します。
引数	id : 使用する ID を指定します。(r_ssi_rzt2_dat.h で定義されています。) R_SSI0_ID : チャンネル 0 を指定 R_SSI1_ID : チャンネル 1 を指定 上記以外 : 設定不可
リターン値	上位 16 ビットにメジャーバージョン、下位 16 ビットにマイナーバージョンが格納されます。 例) 戻り値が 0x00010002 の場合、Ver.1.2
補足	不正な id が指定された場合、戻り値は 0xFFFFFFFF となります。
注意	本関数実行前に、必ず EC-Lib を用いて SSI Encoder I/F のコンフィギュレーションと起動を行ってください。

4.4.5 R_SSI_Control

R_SSI_Control

概要	エンコーダの制御
ヘッダ	r_ssi_rzt2_if.h
宣言	int32_t R_SSI_Control(const int32_t id, const r_ssi_cmd_t cmd, const void *p_buf);
説明	引数 cmd を使用して SSI 通信チャンネルの制御を行います。
引数	id : 使用する ID を指定します。(r_ssi_rzt2_dat.h で定義されています。) R_SSI0_ID : チャンネル 0 を指定 R_SSI1_ID : チャンネル 1 を指定 上記以外 : 設定不可 cmd : コマンド 内容は「表 4.10 r_ssi_cmd_t」を参照してください。 p_buf 各 cmd に対応する引数
リターン値	R_SSI_SUCCESS : 正常終了 R_SSI_ERR_ACCESS : 異常終了 (Open されていません。) R_SSI_ERR_INVALID_ARG : 異常終了 (引数 id, cmd が規定されていない値) その他リターン値は「4.4.5(1) 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_SSI_Open を実行してください。

(1) 制御コマンド

(a) R_SSI_CMD_SET_PARAM

R_SSI_CMD_SET_PARAM

概要	SSI 通信パラメータの設定
ヘッダ	r_ssi_rzt2_if.h
宣言	int32_t R_SSI_Control(const int32_t id, const r_ssi_cmd_t cmd, const void *p_buf);
説明	引数により指定された SSI 通信パラメータを SSI 通信チャンネルに設定します。 SSI 通信チャンネルの CFG、BR、FLDn、RXDSEL、NF、LATDLY レジスタの設定を行います。
引数	id : 使用する ID を指定します。(r_ssi_rzt2_dat.h で定義されています。) R_SSI0_ID : チャンネル 0 を指定 R_SSI1_ID : チャンネル 1 を指定 上記以外 : 設定不可 cmd : R_SSI_CMD_SET_PARAM を指定します。 p_buf : 通信パラメータ 通信パラメータを格納した r_ssi_param_t 構造体のポインタを指定します。詳細は「4.10.1(1) r_ssi_param_t」を参照してください。
リターン値	R_SSI_SUCCESS : 正常終了 R_SSI_ERR_ACCESS : 異常終了 (Open されていません。) R_SSI_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 または構造体のメンバ変数が規定されていない値) R_SSI_ERR_BUSY : 異常終了 (通信中またはタイミイベント受付中)
注意	本関数実行前に、必ず R_SSI_Open を実行してください。 コールバック関数内で、本制御コマンドを実行することは禁止します。

(b) R_SSI_CMD_SET_CALLBACK

R_SSI_CMD_SET_CALLBACK

概要	エンコーダ制御のコールバック関数を設定
ヘッダ	r_ssi_rzt2_if.h
宣言	int32_t R_SSI_Control(const int32_t id, const r_ssi_cmd_t cmd, const void *p_buf);
説明	引数で指定された SSI 通信チャンネルのコールバック関数を設定します。
引数	id : 使用する ID を指定します。(r_ssi_rzt2_dat.h で定義されています。) R_SSI0_ID : チャンネル 0 を指定 R_SSI1_ID : チャンネル 1 を指定 上記以外 : 設定不可 cmd : R_SSI_CMD_SET_CALLBACK を指定します。 p_buf : コールバック関数情報 コールバック関数情報を格納した r_ssi_callback_t 構造体のポインタを指定します。詳細は「4.10.1(5) r_ssi_callback_t」を参照してください。 NULL を指定した場合、コールバック関数の登録は解除されます。
リターン値	R_SSI_SUCCESS : 正常終了 R_SSI_ERR_ACCESS : 異常終了 (Open されていません。) R_SSI_ERR_INVALID_ARG : 異常終了 (引数 id, cmd が規定されていない値) R_SSI_ERR_BUSY : 異常終了 (通信中またはタイミイベント受付中)
注意	本関数実行前に、必ず R_SSI_Open を実行してください。 コールバック関数内で、本制御コマンドを実行することは禁止します。

(c) R_SSI_CMD_SET_MASK

R_SSI_CMD_SET_MASK	
概要	エンコーダ制御の SSI ステータス割り込みマスクを設定
ヘッダ	r_ssi_rzt2_if.h
宣言	int32_t R_SSI_Control(const int32_t id, const r_ssi_cmd_t cmd, const void *p_buf);
説明	引数で指定された SSI 通信チャンネルで、SSI ステータス割り込みの割り込みマスクを設定します。 SSI 通信チャンネルの MASK レジスタの設定を行います。
引数	id : 使用する ID を指定します。(r_ssi_rzt2_dat.h で定義されています。) R_SSI0_ID : チャンネル 0 を指定 R_SSI1_ID : チャンネル 1 を指定 上記以外 : 設定不可 cmd : R_SSI_CMD_SET_MASK を指定します。 p_buf : マスク設定 マスク設定を格納した r_ssi_mask_t 構造体のポインタを指定します。詳細は「4.10.1(6) r_ssi_mask_t」を参照してください。
リターン値	R_SSI_SUCCESS : 正常終了 R_SSI_ERR_ACCESS : 異常終了 (Open されていません。) R_SSI_ERR_INVALID_ARG : 異常終了 (引数 id, cmd が規定されていない値、または p_buf が NULL)
注意	本関数実行前に、必ず R_SSI_Open を実行してください。 コールバック関数内で、本制御コマンドを実行することは禁止します。

(d) R_SSI_CMD_START

R_SSI_CMD_START	
概要	SSI 通信を開始
ヘッダ	r_ssi_rzt2_if.h
宣言	int32_t R_SSI_Control(const int32_t id, const r_ssi_cmd_t cmd, const void *p_buf);
説明	引数で指定された SSI 通信チャンネルの通信を開始します。 通信により、R_SSI_CMD_SET_CALLBACK コマンドにより指定されたコールバック関数がコールされます。コールバック関数の詳細は、「4.5.1 ssi_int_ssi_callback」～「4.5.2 ssi_int_fifo_callback」を参照してください。
引数	id : 使用する ID を指定します。(r_ssi_rzt2_dat.h で定義されています。) R_SSI0_ID : チャンネル 0 を指定 R_SSI1_ID : チャンネル 1 を指定 上記以外 : 設定不可 cmd : R_SSI_CMD_START を指定します。 p_buf : 使用しません。(NULL を指定してください)
リターン値	R_SSI_SUCCESS : 正常終了 R_SSI_ERR_ACCESS : 異常終了 (Open されていません。) R_SSI_ERR_INVALID_ARG : 異常終了 (引数 id, cmd が規定されていない値) R_SSI_ERR_BUSY : 異常終了 (通信中またはタイミイベント受付中)
注意	本関数実行前に、必ず R_SSI_Open を実行してください。

(e) R_SSI_CMD_ELC_START

R_SSI_CMD_ELC_START

概要	タイマイイベント同期 SSI 通信の受付を開始
ヘッダ	r_ssi_rzt2_if.h
宣言	int32_t R_SSI_Control(const int32_t id, const r_ssi_cmd_t cmd, const void *p_buf);
説明	引数で指定された SSI 通信チャンネルで、ELC イベント同期通信を許可します。 通信により、R_SSI_CMD_SET_CALLBACK コマンドにより指定されたコールバック関数がコールされます。コールバック関数の詳細は、「4.5.1 ssi_int_ssi_callback」～「4.5.2 ssi_int_fifo_callback」を参照してください。
引数	id : 使用する ID を指定します。(r_ssi_rzt2_dat.h で定義されています。) R_SSI0_ID : チャンネル 0 を指定 R_SSI1_ID : チャンネル 1 を指定 上記以外 : 設定不可 cmd : R_SSI_CMD_ELC_START を指定します。 p_buf : 使用しません。(NULL を指定してください)
リターン値	R_SSI_SUCCESS : 正常終了 R_SSI_ERR_ACCESS : 異常終了 (Open されていません。) R_SSI_ERR_INVALID_ARG : 異常終了 (引数 id, cmd が規定されていない値) R_SSI_ERR_BUSY : 異常終了 (通信中またはタイマイイベント受付中)
注意	本関数実行前に、必ず R_SSI_Open を実行してください。

(f) R_SSI_CMD_ELC_STOP

R_SSI_CMD_ELC_STOP

概要	タイマイイベント同期 SSI 通信の受付を停止
ヘッダ	r_ssi_rzt2_if.h
宣言	int32_t R_SSI_Control(const int32_t id, const r_ssi_cmd_t cmd, const void *p_buf);
説明	引数で指定された SSI 通信チャンネルの ELC イベント同期通信を停止します。
引数	id : 使用する ID を指定します。(r_ssi_rzt2_dat.h で定義されています。) R_SSI0_ID : チャンネル 0 を指定 R_SSI1_ID : チャンネル 1 を指定 上記以外 : 設定不可 cmd : R_SSI_CMD_ELC_STOP を指定します。 p_buf : 使用しません。(NULL を指定してください)
リターン値	R_SSI_SUCCESS : 正常終了 R_SSI_ERR_ACCESS : 異常終了 (Open されていません。) R_SSI_ERR_INVALID_ARG : 異常終了 (引数 id, cmd が規定されていない値) R_SSI_ERR_BUSY : 異常終了 (通信中です。)
注意	本関数実行前に、必ず R_SSI_Open を実行してください。

4.4.6 R_SSI_GetStatus

R_SSI_GetStatus

概要	エンコーダステータスの取得
ヘッダ	r_ssi_rzt2_if.h
宣言	int32_t R_SSI_GetStatus(const int32_t id, r_ssi_status_t *p_status);
説明	引数で指定された SSI 通信チャネルのステータス情報を取得します。 SS レジスタの内容を引数で指定したアドレスに格納します。
引数	id : 使用する ID を指定します。(r_ssi_rzt2_dat.h で定義されています。) R_SSI0_ID : チャンネル 0 を指定 R_SSI1_ID : チャンネル 1 を指定 上記以外 : 設定不可 p_status : ステータス取得結果 ステータス取得結果の格納先のポインタを指定します。構造体 r_ssi_status_t で宣言されたステータス取得結果が格納されます。 詳細は「4.10.1(3) r_ssi_status_t」を参照してください。
リターン値	R_SSI_SUCCESS : 正常終了 R_SSI_ERR_INVALID_ARG : 異常終了 (引数 id が規定されていない値、 または p_status が NULL)

4.4.7 R_SSI_GetRxd

R_SSI_GetRxd

概要	エンコーダ受信結果 RXD レジスタの取得
ヘッダ	r_ssi_rzt2_if.h
宣言	int32_t R_SSI_GetRxd(const int32_t id, r_ssi_rxd_t *p_rxd);
説明	引数で指定された SSI 通信チャネルのエンコーダ受信結果 RXD レジスタを取得します。
引数	id : 使用する ID を指定します。(r_ssi_rzt2_dat.h で定義されています。) R_SSI0_ID : チャンネル 0 を指定 R_SSI1_ID : チャンネル 1 を指定 上記以外 : 設定不可 p_rxd : RXD レジスタ取得結果 RXD レジスタ取得結果の格納先のポインタを指定します。構造体 r_ssi_rxd_t で宣言された RXD レジスタ取得結果が格納されます。 詳細は「0 r_ssi_rxd_t」を参照してください。
リターン値	R_SSI_SUCCESS : 正常終了 R_SSI_ERR_INVALID_ARG : 異常終了 (引数 id が規定されていない値、 または p_rxd が NULL)

4.5 ユーザー定義関数仕様

4.5.1 ssi_int_ssi_callback

ssi_int_ssi_callback	
概要	SSI 通信によるステータス割り込みを通知
ヘッダ	-
宣言	static void ssi_int_ssi_callback(r_ssi_status_t *p_status);
説明	R_SSI_Control(R_SSI_CMD_SET_CALLBACK)関数で登録するコールバック関数です。SSI 通信によりステータスが変化して割り込みが発生した際にコールされます。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、自由に設定できません。
引数	p_status : ステータス取得結果 ステータス取得結果の格納先のポインタを指定します。構造体 r_ssi_status_t で宣言されたステータス取得結果が格納されます。
リターン値	なし
注意	タイマイイベント同期通信が有効の場合、次のイベントが発生する前に受信結果を取得してください。

4.5.2 ssi_int_fifo_callback

ssi_int_fifo_callback	
概要	SSI 通信による FIFO 割り込みを通知
ヘッダ	-
宣言	static void ssi_int_fifo_callback(r_ssi_result_t *p_result);
説明	R_SSI_Control(R_SSI_CMD_SET_CALLBACK)関数で登録するコールバック関数です。SSI 通信によりデータ受信が完了した際にコールされます。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、自由に設定できません。
引数	p_result : SSI 受信結果 SSI 受信時の受信結果の格納先のポインタを指定します。構造体 r_ssi_result_t で宣言されたフィールドデータとステータス取得結果が格納されます。
リターン値	なし
注意	タイマイイベント同期通信が有効の場合、次のイベントが発生する前に受信結果を取得してください。

4.6 割り込みハンドラ

4.6.1 enc_ch0_int_ssi_isr

enc_ch0_int_ssi_isr	
概要	チャンネル 0 の INT_SSI 割り込みの割り込みハンドラ
ヘッダ	-
宣言	void enc_ch0_int_ssi_isr(void);
説明	SSI 通信チャンネルの下記の割り込み要因に対する割り込みハンドラです。 R_SSI_Control(R_SSI_CMD_SET_MASK)によりマスクされます。 1. SSI ステータス割り込み(FIFO エラー) 2. SSI ステータス割り込み(エラーフラグ) 3. SSI ステータス割り込み(継続判定) 4. SSI ステータス割り込み(受信完了)
引数	なし
リターン値	なし

4.6.2 enc_ch0_int_fifo_isr

enc_ch0_int_fifo_isr	
概要	チャンネル 0 の INT_FIFO 割り込みの割り込みハンドラ
ヘッダ	-
宣言	void enc_ch0_int_fifo_isr(void);
説明	SSI 通信チャンネル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. FIFO 受信完了割り込み
引数	なし
リターン値	なし

4.6.3 enc_ch1_int_ssi_isr

enc_ch1_int_ssi_isr	
概要	チャンネル 1 の INT_SSI 割り込みの割り込みハンドラ
ヘッダ	-
宣言	void enc_ch1_int_ssi_isr(void);
説明	SSI 通信チャンネルの下記の割り込み要因に対する割り込みハンドラです。 R_SSI_Control(R_SSI_CMD_SET_MASK)によりマスクされます。 1. SSI ステータス割り込み(FIFO エラー) 2. SSI ステータス割り込み(エラーフラグ) 3. SSI ステータス割り込み(継続判定) 4. SSI ステータス割り込み(受信完了)
引数	なし
リターン値	なし

4.6.4 enc_ch1_int_fifo_isr

enc_ch1_int_fifo_isr	
概要	チャンネル 1 の INT_FIFO 割り込みの割り込みハンドラ
ヘッダ	-
宣言	void enc_ch1_int_fifo_isr(void);
説明	SSI 通信チャンネル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. FIFO 受信完了割り込み
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 SSI ドライバで使用する割り込みを示します。

表 4.2 SSI ドライバで使用する割り込み

割り込み	ID	概要
ENCIF_INT0	372	チャンネル 0 の SSI 通信ステータスにより割り込みが発生します。
ENCIF_INT2	374	チャンネル 0 の受信が完了しデータ取得が可能となると割り込みが発生します。
ENCIF_INT4	376	チャンネル 1 の SSI 通信ステータスにより割り込みが発生します。
ENCIF_INT6	378	チャンネル 1 の受信が完了しデータ取得が可能となると割り込みが発生します。

4.8 定数/エラーコード一覧

表 4.3 に定数/エラーコード定義表の一覧を示します。各定義については、それぞれの表を参照してください。

表 4.3 定数/エラーコード定義表の一覧

表番号	内容
表 4.4	SSI ドライバで使用する定数(r_ssi_rzt2_dat.h)
表 4.5	SSI ドライバで使用するユーザー定義の定数(r_ssi_rzt2_config.h)
表 4.6	ビットレート
表 4.7	エラーコード
表 4.8	SSI 通信コントローラ仕様による規定値

表 4.4 SSI ドライバで使用する定数(r_ssi_rzt2_dat.h)

定数名	設定値	内容
R_SSI0_ID	0x01	EC-Lib チャンネル 0 で指定するコンフィギュレーション ID
R_SSI1_ID	0x02	EC-Lib チャンネル 1 で指定するコンフィギュレーション ID
R_SSI_FREQ	50000UL	SSI 通信コントローラの動作周波数 50.0MHz

表 4.5 SSI ドライバで使用するユーザー定義の定数(r_ssi_rzt2_config.h)

定数名	設定値	内容
R_SSI_INT_SSI_PRI	11	INT_SSI 割り込みの優先度を指定(0~31)
R_SSI_INT_FIFO_PRI	11	INT_FIFO 割り込みの優先度を指定(0~31)
R_SSI_NFINTV_2MHz	0	ビットレートが 2MHz の場合の NFINTV ビット設定値
R_SSI_NFINTV_1MHz	1	ビットレートが 1MHz の場合の NFINTV ビット設定値
R_SSI_NFINTV_400kHz	2	ビットレートが 400kHz の場合の NFINTV ビット設定値
R_SSI_NFINTV_299kHz	2	ビットレートが 299.40kHz の場合の NFINTV ビット設定値
R_SSI_NFINTV_200kHz	3	ビットレートが 200kHz の場合の NFINTV ビット設定値
R_SSI_NFINTV_100kHz	4	ビットレートが 100kHz の場合の NFINTV ビット設定値
R_SSI_NFINTV_80kHz	4	ビットレートが 80.12kHz の場合の NFINTV ビット設定値
R_SSI_NFSCNT_2MHz	9	ビットレートが 2MHz の場合の NFSCNT ビット設定値
R_SSI_NFSCNT_1MHz	9	ビットレートが 1MHz の場合の NFSCNT ビット設定値
R_SSI_NFSCNT_400kHz	11	ビットレートが 400kHz の場合の NFSCNT ビット設定値
R_SSI_NFSCNT_299kHz	15	ビットレートが 299.40kHz の場合の NFSCNT ビット設定値
R_SSI_NFSCNT_200kHz	11	ビットレートが 200kHz の場合の NFSCNT ビット設定値
R_SSI_NFSCNT_100kHz	11	ビットレートが 100kHz の場合の NFSCNT ビット設定値
R_SSI_NFSCNT_80kHz	15	ビットレートが 80.12kHz の場合の NFSCNT ビット設定値

表 4.6 ビットレート

定数名	設定値	内容
R_SSI_2MHz	0x063	ビットレート設定値(2MHz)
R_SSI_1MHz	0x0C7	ビットレート設定値(1MHz)
R_SSI_400kHz	0x17C	ビットレート設定値(400kHz)
R_SSI_299kHz	0x1A6	ビットレート設定値(299.40kHz)
R_SSI_200kHz	0x1F9	ビットレート設定値(200kHz)
R_SSI_100kHz	0x27C	ビットレート設定値(100kHz)
R_SSI_80kHz	0x29B	ビットレート設定値(80.12kHz)

表 4.7 エラーコード

定数名	設定値	内容
R_SSI_SUCCESS	0	正常終了
R_SSI_ERR_INVALID_ARG	-1	引数異常
R_SSI_ERR_BUSY	-2	API を実行できない状態
R_SSI_ERR_ACCESS	-3	API の実行順序エラー

表 4.8 SSI 通信コントローラ仕様による規定値

定数名	設定値	内容
R_SSI_FLD_MAX_NUM	8	フィールド数最大値
R_SSI_FLDSIZE_MAX_NUM	32	各フィールドのビット数最大値

4.9 固定幅整数一覧

表 4.9 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 4.9 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_ssi_param_t

SSI 通信設定用のパラメータ情報。

```
typedef struct
{
    uint16_t  bitrate;          ビットレート設定
                                SSI 通信チャネルのビットレート設定レジスタ(BR)に設定する値を設定
                                します。指定する値は「表 4.6 ビットレート」を参照してください。
    uint8_t   fldnum;          データフィールド数 (1~8)
                                SSI 通信チャネルの通信設定レジスタ(CFG)に設定するフィールド数を設定し
                                ます。
    uint8_t   fldsize[R_SSI_F フィールドサイズ (1~32)
                                LD_MAX_NUM]; 各フィールドのビット数を設定します。SSI 通信チャネルの FLDn レジスタに
                                は 1 引いた値が設定されます。
    uint8_t   gry[R_SSI_FLD_ グレイコード変換設定 (0 or 1)
                                MAX_NUM]; 0: 各フィールドのデータ取得時にグレイコード->バイナリ変換を行う機能を
                                無効とします。
                                1: 各フィールドのデータ取得時にグレイコード->バイナリ変換を行う機能を
                                有効とします。
    uint8_t   rxd0sel;         フィールド番号 (0~7)
                                受信データレジスタ RXD0 に表示するフィールド番号を指定します。
    uint8_t   rxd1sel;         フィールド番号 (0~7)
                                受信データレジスタ RXD1 に表示するフィールド番号を指定します。
    uint8_t   nfintv;          ノイズフィルタのサンプリング周期
                                SSI 通信チャネルのノイズフィルタ設定レジスタ(NF)の NFINTV に設定する値
                                を指定します。指定する値は「表 4.5 SSI ドライバで使用するユーザー定義
                                の定数(r_ssi_rzt2_config.h)」を参照してください。
    uint8_t   nfscnt           ノイズフィルタのサンプリング段数
                                SSI 通信チャネルのノイズフィルタ設定レジスタ(NF)の NFSCNT に設定する
                                値を指定します。指定する値は「表 4.5 SSI ドライバで使用するユーザー定
                                義の定数(r_ssi_rzt2_config.h)」を参照してください。
} r_ssi_param_t
```

(2) r_ssi_rxd_t

SSI 受信時の RXD レジスタデータ

```
typedef struct
{
    uint32_t  rxd0;           受信データレジスタ 0(RXD0)の値が格納されます。
    uint32_t  rxd1           受信データレジスタ 1(RXD1)の値が格納されます。
}
r_ssi_rxd_t
```

(3) r_ssi_status_t

SSI 受信時のステータス

メンバ変数が表す情報の詳細は、「RZ/T2M グループ SSI インタフェース(SSI) ユーザーズマニュアル」の「ステータスレジスタ(SS)」を参照してください。

```
typedef struct
{
    bool        fifoerr;        FIFO がオーバーフローした状態を示すステータスを格納
                                (true: オーバーフローあり)
    bool        errflg;        SSI 受信後のエラーフラグを示すステータスを格納
                                (true: エラーあり)
    bool        cont;          前回通信の継続であることを示すステータスを格納
                                (true: 継続)
    bool        end;           SSI 受信が完了したことを示すステータスを格納
                                (true: 受信完了)
    bool        standby        通信開始待ち状態 (スタンバイ状態) であることを示すステータスを格納
                                (true: スタンバイ)
} r_ssi_status_t
```

(4) r_ssi_result_t

SSI 受信時の受信結果

```
typedef struct
{
    r_ssi_status_t    status;    受信ステータス
                                詳細は構造体「r_ssi_status_t」を参照してください。
    uint32_t          fdat[R_SSI_FLD 受信データ
                            _MAX_NUM] 受信した各フィールドのデータが格納されます。
} r_ssi_result_t
```

(5) r_ssi_callback_t

SSI ドライバのコールバック関数設定用構造体

```
typedef struct
{
    r_ssi_int_ssi_cb_t    cbadr_int_ssi;  SSI ステータス割り込みでコールされるコールバック関数のポインタ
                                詳細は「4.5.1 ssi_int_ssi_callback」を参照してください。
    r_ssi_int_fifo_cb_t   cbadr_int_fifo  FIFO 割り込みでコールされるコールバック関数のポインタ
                                詳細は「4.5.2 ssi_int_fifo_callback」を参照してください。
} r_ssi_callback_t
```

(6) r_ssi_mask_t

SSI ドライバのステータス割り込みマスク設定用構造体

```
typedef struct
{
    bool    fifoerr;    FIFO がオーバーフローした状態を示すステータスによる割り込み許可
                    (true: 割り込み許可)
    bool    errflg;    SSI 受信後のエラーフラグを示すステータスによる割り込み許可
                    (true: 割り込み許可)
    bool    cont;      前回通信の継続であることを示すステータスによる割り込み許可
                    (true: 割り込み許可)
    bool    end        SSI 受信が完了したことを示すステータスによる割り込み許可
                    (true: 割り込み許可)
} r_ssi_mask_t
```

4.10.2 共用体

使用しません。

4.10.3 列挙型

(1) r_ssi_cmd_t

制御コマンドを表す列挙型です。表 4.10 に内容を示します。

表 4.10 r_ssi_cmd_t

定数名	設定値	内容
R_SSI_CMD_SET_PARAM	0	パラメータ設定
R_SSI_CMD_SET_CALLBACK	1	コールバック関数設定
R_SSI_CMD_SET_MASK	2	SSI ステータス割り込みマスク設定
R_SSI_CMD_START	3	通信開始
R_SSI_CMD_ELC_START	4	タイマイベント同期通信の受付
R_SSI_CMD_ELC_STOP	5	タイマイベント同期通信の終了
R_SSI_CMD_MAX	6	コマンド数

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) コンソールからのコマンドにより通信パラメータを設定
- 2) コンソールからのコマンドによりエンコーダ通信を実施
 - a) FIFO 受信完了割り込み(INT_FIFO)によりデータを取得
 - b) SSI ステータス割り込み(INT_SSI)によりデータ取得を中止
 - c) 受信データ、ステータスをコンソールに表示
- 3) コンソールからのコマンドにより GPT タイマ機能を使用して定期的にエンコーダ通信を実施
 - a) FIFO 受信完了割り込み(INT_FIFO)によりデータを複数回取得
 - b) SSI ステータス割り込み(INT_SSI)によりデータ取得を中止
 - c) 複数の受信データ、ステータスをコンソールに表示
- 4) ELC イベント入カトリガ機能を使用して定期的にエンコーダ通信を実施
 - a) FIFO 受信完了割り込み(INT_FIFO)によりデータを複数回取得
 - b) SSI ステータス割り込み(INT_SSI)によりデータ取得を中止
 - c) 複数の受信データ、ステータスをコンソールに表示

(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。

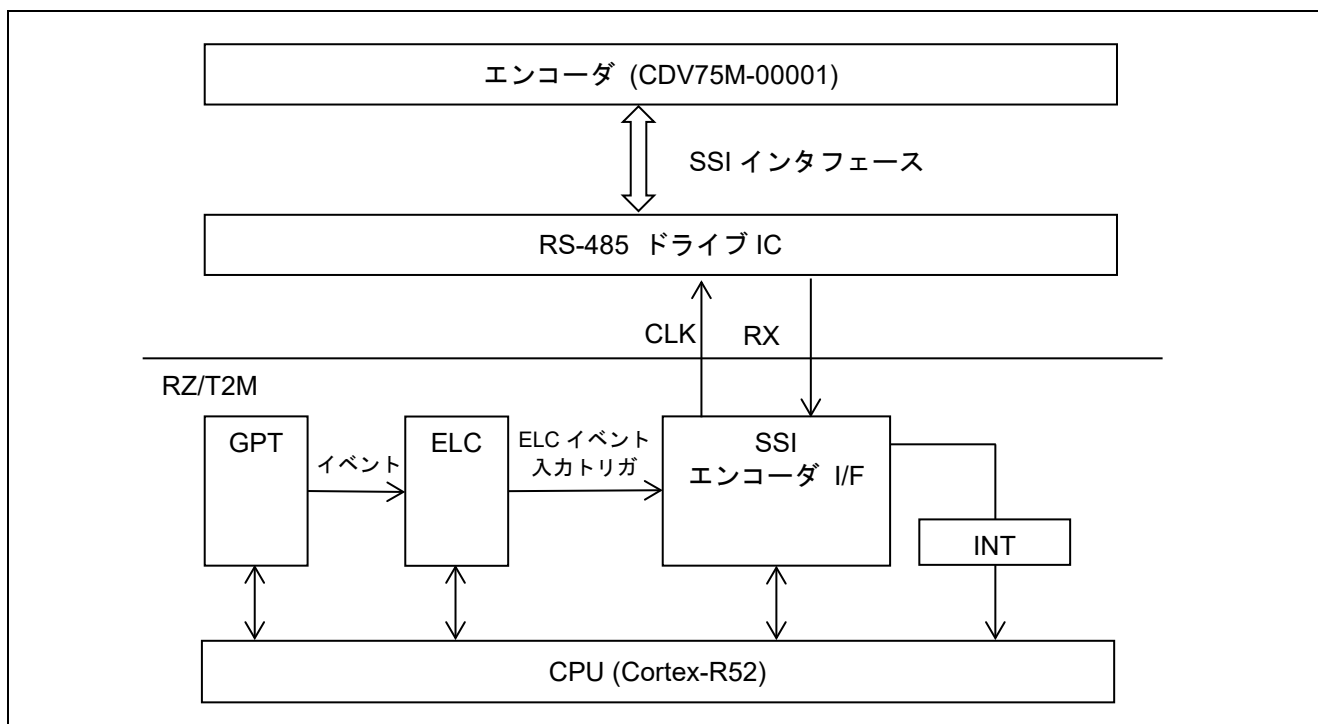


図 4-1 システムブロック図

(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。

SSI ドライバには、R_SSI_Open 関数で構成される開始処理部、R_SSI_Close 関数で構成される終了処理部、R_SSI_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（割り込みハンドラ）があります。

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

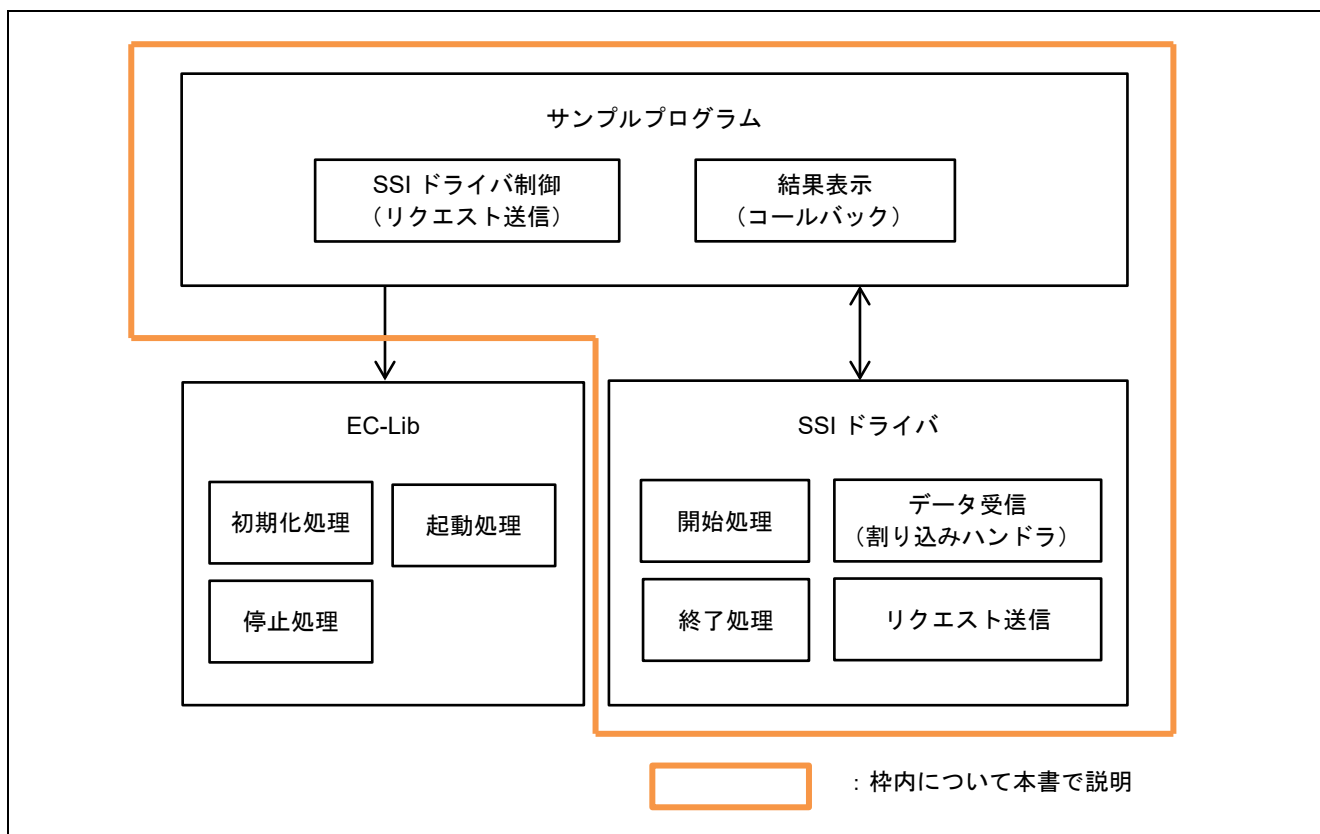


図 4-2 ソフトウェア構成図

4.11.2 サンプルプログラム関数一覧

表 4.11 にサンプルプログラム関数一覧を示します

表 4.11 サンプルプログラム関数一覧

関数名	ページ番号
hal_entry	25
enc_main	25
ssi_cmd_control	25
ssi_br	25
ssi_flg	26
ssi_gry	26
ssi_mask	26
ssi_start	26
ssi_start_cont	27
ssi_timer_start	27
ssi_elc_start	27
ssi_exit	27
ssi_int_ssi_callback	28
ssi_int_fifo_callback	28
get_cmd	28
show_all	28
show_result	29
show_status	29
show_rxd	29
r_g_timer0_callback	29
timer_start	30
timer_stop	30

4.11.3 サンプルプログラム関数仕様

(1) hal_entry

hal_entry	
概要	SSI 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	SSI 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	SSI 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	SSI 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」を参照してください。
引数	ch エンコーダチャンネル番号 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 (エンコーダ I/F のエラーコード)

(3) ssi_cmd_control

ssi_cmd_control	
概要	SSI サンプルプログラムのドライバ制御関数
ヘッダ	-
宣言	static void ssi_cmd_control(void);
説明	以下の処理を行います。 <ul style="list-style-type: none"> ● エンコーダ制御の開始 ● コンソールコマンドの入力処理 ● エンコーダ制御の終了
引数	なし
リターン値	なし

(4) ssi_br

ssi_br	
概要	br コンソールコマンド関数
ヘッダ	-
宣言	static void ssi_br(uint32_t arg_num, char_t *p_arg[]);
説明	コンソールコマンド br が入力された場合に実行される関数です。詳細は「4.11.8 コンソールコマンド」を参照してください。
引数	arg_num コンソールから入力された文字列の数 *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

(5) ssi_fld

ssi_fld	
概要	fld コンソールコマンド関数
ヘッダ	-
宣言	static void ssi_fld(uint32_t arg_num, char_t *p_arg[]);
説明	コンソールコマンド fld が入力された場合に実行される関数です。詳細は「4.11.8 コンソールコマンド」を参照してください。
引数	arg_num コンソールから入力された文字列の数 *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

(6) ssi_gry

ssi_gry	
概要	gry コンソールコマンド関数
ヘッダ	-
宣言	static void ssi_gry(uint32_t arg_num, char_t *p_arg[]);
説明	コンソールコマンド gry が入力された場合に実行される関数です。詳細は「4.11.8 コンソールコマンド」を参照してください。
引数	arg_num コンソールから入力された文字列の数 *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) ssi_mask

ssi_mask	
概要	mask コンソールコマンド関数
ヘッダ	-
宣言	static void ssi_mask(uint32_t arg_num, char_t *p_arg[]);
説明	コンソールコマンド mask が入力された場合に実行される関数です。詳細は「4.11.8 コンソールコマンド」を参照してください。
引数	arg_num コンソールから入力された文字列の数 *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) ssi_start

ssi_start	
概要	start コンソールコマンド関数
ヘッダ	-
宣言	static void ssi_start(uint32_t arg_num, char_t *p_arg[]);
説明	コンソールコマンド start が入力された場合に実行される関数です。詳細は「4.11.8 コンソールコマンド」を参照してください。
引数	arg_num コンソールから入力された文字列の数 *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) ssi_start_cont

ssi_start_cont	
概要	start_cont コンソールコマンド関数
ヘッダ	-
宣言	static void ssi_start_cont(uint32_t arg_num, char_t *p_arg[]);
説明	コンソールコマンド start_cont が入力された場合に実行される関数です。詳細は「4.11.8 コンソールコマンド」を参照してください。
引数	arg_num コンソールから入力された文字列の数 *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) ssi_timer_start

ssi_timer_start	
概要	timer_start コンソールコマンド関数
ヘッダ	-
宣言	static void ssi_timer_start(uint32_t arg_num, char_t *p_arg[]);
説明	コンソールコマンド timer_start が入力された場合に実行される関数です。詳細は「4.11.8 コンソールコマンド」を参照してください。
引数	arg_num コンソールから入力された文字列の数 *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) ssi_elc_start

ssi_elc_start	
概要	elc_start コンソールコマンド関数
ヘッダ	-
宣言	static void ssi_elc_start(uint32_t arg_num, char_t *p_arg[]);
説明	コンソールコマンド elc_start が入力された場合に実行される関数です。詳細は「4.11.8 コンソールコマンド」を参照してください。
引数	arg_num コンソールから入力された文字列の数 *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) ssi_exit

ssi_exit	
概要	exit コンソールコマンド関数
ヘッダ	-
宣言	static void ssi_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。詳細は「4.11.8 コンソールコマンド」を参照してください。
引数	arg_num コンソールから入力された文字列の数 *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

(13) ssi_int_ssi_callback

ssi_int_ssi_callback	
概要	SSI ドライバのステータス割り込み発生時のコールバック関数
ヘッダ	-
宣言	static void ssi_int_ssi_callback(r_ssi_status_t *p_status);
説明	SSI ドライバにて、SSI ステータス割り込み発生時に実行されるコールバック関数です。
引数	*p_status 受信ステータスが格納されている RAM の先頭アドレス
リターン値	なし

(14) ssi_int_fifo_callback

ssi_int_fifo_callback	
概要	SSI ドライバの FIFO 割り込み発生時のコールバック関数
ヘッダ	-
宣言	static void ssi_int_fifo_callback(r_ssi_result_t *p_result);
説明	SSI ドライバにて、FIFO に受信データが格納されたときに実行されるコールバック関数です。
引数	*p_result 受信結果が格納されている RAM の先頭アドレス
リターン値	なし

(15) get_cmd

get_cmd	
概要	コマンドの取得関数
ヘッダ	-
宣言	static uint32_t 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

(16) show_all

show_all	
概要	全受信結果を表示する関数
ヘッダ	-
宣言	static void show_all(const uint32_t num);
説明	全受信結果の各フィールド、ステータス、RXD レジスタ内容を表示します。
引数	num 表示する受信結果の総数
リターン値	なし

(17) show_result

show_result	
概要	受信結果（フィールド、ステータス）を表示する関数
ヘッダ	-
宣言	static void show_result(const r_ssi_result_t *p_result, uint8_t fld_num);
説明	受信結果の各フィールド、ステータス内容を表示します。
引数	*p_result 表示する受信結果 fld_num 表示するフィールド数
リターン値	なし

(18) show_status

show_status	
概要	受信結果（ステータス）を表示する関数
ヘッダ	-
宣言	static void show_status(const r_ssi_status_t *p_status);
説明	受信結果のステータス内容を表示します。
引数	*p_status 表示するステータス
リターン値	なし

(19) show_rxd

show_rxd	
概要	受信結果（RXD レジスタ）を表示する関数
ヘッダ	-
宣言	static void show_rxd(const r_ssi_rxd_t *p_rxd);
説明	受信結果の RXD レジスタデータを表示します。
引数	*p_rxd 表示する RXD レジスタデータ
リターン値	なし

(20) r_g_timer0_callback

r_g_timer0_callback	
概要	サンプルプログラムのタイマ動作を行うコールバック関数
ヘッダ	-
宣言	void r_g_timer0_callback(timer_callback_args_t *p_args);
説明	一定周期で呼び出される GPT0 のコールバック関数です。 タイマにより周期的に SSI 通信リクエストを行います。
引数	*p_args r_gpt ドライバのタイマコールバック引数
リターン値	なし

(21) timer_start

timer_start	
概要	サンプルプログラムのタイマ周期動作開始を行う関数
ヘッダ	-
宣言	static int32_t timer_start(timer_ctrl_t * const p_ctrl, timer_cfg_t const * const p_cfg, uint32_t usec);
説明	引数で指定された GPT(GPT0 または GPT1)を起動します。
引数	*p_ctrl r_gpt ドライバのコントロールデータ *p_cfg r_gpt ドライバのコンフィギュレーションデータ usec タイマ周期 [usec]
リターン値	R_SSI_SUCCESS : 正常終了 R_SSI_ERR_ACCESS : 異常終了

(22) timer_stop

timer_stop	
概要	サンプルプログラムのタイマ周期動作停止を行う関数
ヘッダ	-
宣言	static int32_t timer_stop(timer_ctrl_t * const p_ctrl);
説明	引数で指定された GPT(GPT0 または GPT1)を停止します。
引数	*p_ctrl r_gpt ドライバのコントロールデータ
リターン値	R_SSI_SUCCESS : 正常終了 R_SSI_ERR_ACCESS : 異常終了

4.11.4 サンプルプログラムの変数一覧

表 4.12 に static 型変数を示します。

表 4.12 static 型変数

型	変数名	内容
enum ssi_mode	ssi_mode	SSI 通信を行うモードを格納します。 SSI_MODE_SINGLE 1 回通信を行うモード SSI_MODE_CONT 2 連続で通信を行うモード SSI_MODE_TIMER 複数回通信を行うモード
r_ssi_param_t	ssi_param	SSI ドライバの設定パラメータを格納します。
r_ssi_callback_t	ssi_callback_dat	SSI ドライバへのコールバック設定を格納します。
r_ssi_mask_t	ssi_mask_dat	SSI ドライバへの割り込みマスク設定を格納します。
r_ssi_status_t	ssi_status	SSI 受信結果の SSI ステータスを格納します。
r_ssi_result_t	ssi_result[SSI_TIMER_COUNT_MAX]	SSI 受信結果を格納します。 最大 SSI_TIMER_COUNT_MAX 回の通信結果を格納します。
r_ssi_rxd_t	ssi_rxd[SSI_TIMER_COUNT_MAX]	SSI 受信結果を格納します。 最大 SSI_TIMER_COUNT_MAX 回の通信結果を格納します。
uint16_t	ssi_result_index	SSI 受信結果を格納する格納先 index を示します。
int32_t	ssi_err_code	SSI 通信で発生したエラーコードを保持します。

4.11.5 サンプルプログラムの定数一覧

表 4.13 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 4.13 主要な定数

定義名	設定値	内容
SSI_CONT_COUNT_MAX	2	start_cont コマンドで連続通信を行う際の通信回数
SSI_TIMER_COUNT_MAX	5	timer_start, elc_start コマンドで複数回通信を行う際の通信回数
SSI_TIMER_PERIOD	1000	start_cont, timer_start, elc_start コマンドで通信を行う際のデフォルト通信間隔 [usec]

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

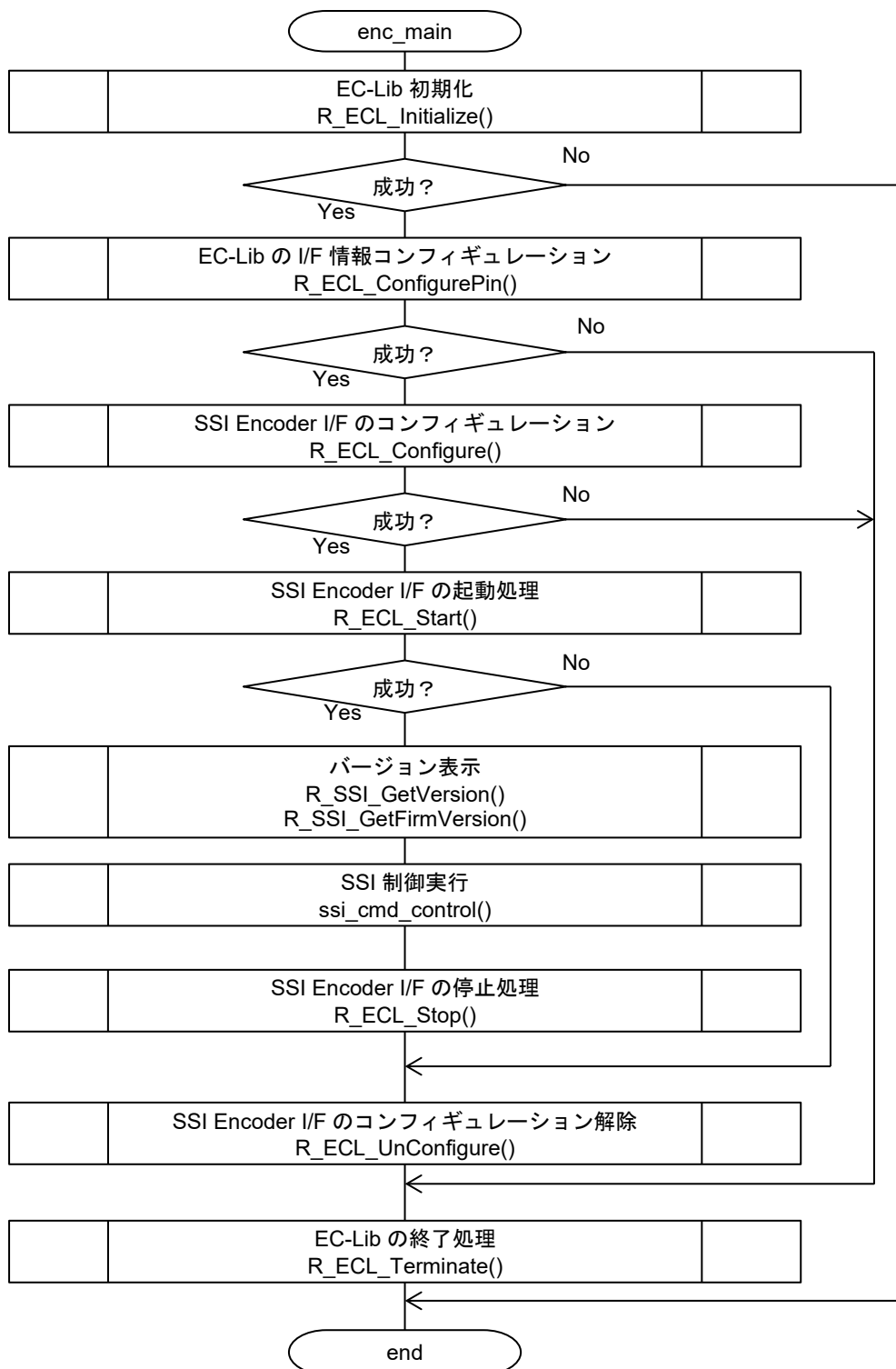


図 4-3 enc_main 関数のフローチャート

(2) ssi_cmd_control フローチャート

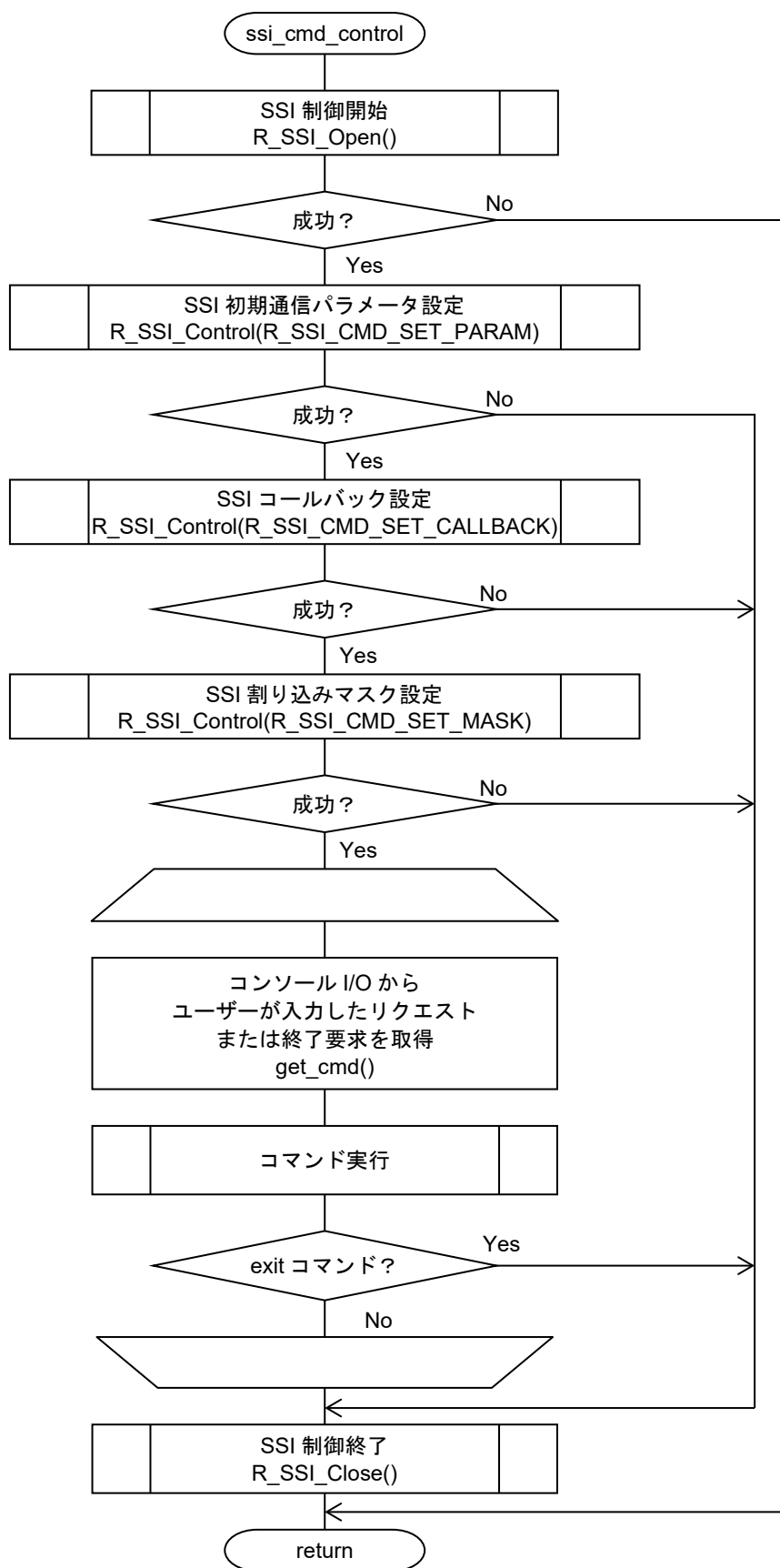


図 4-4 ssi_cmd_control 関数のフローチャート

(3) ssi_br, ssi_fld, ssi_gry フローチャート

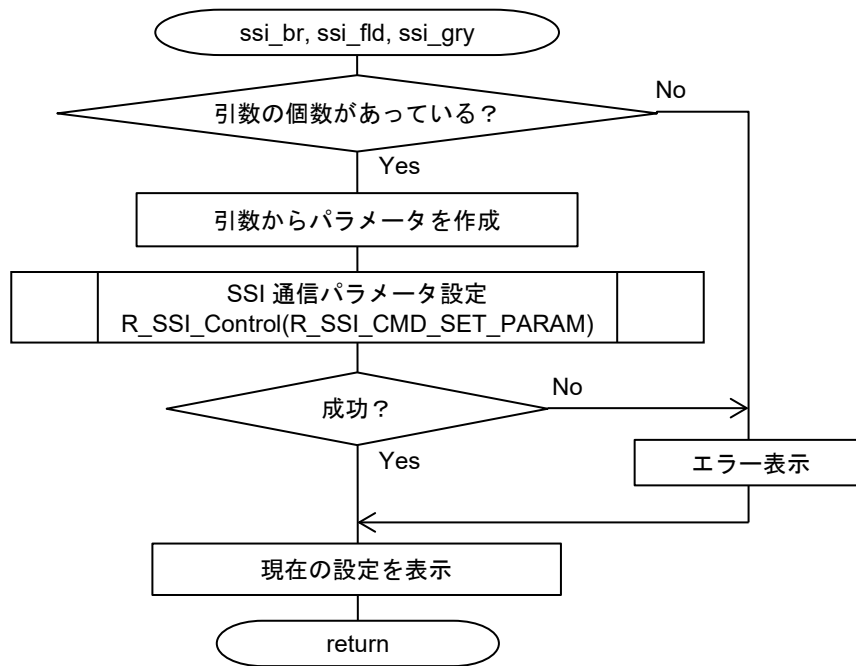


図 4-5 ssi_br, ssi_fld, ssi_gry 関数のフローチャート

(4) ssi_mask フローチャート

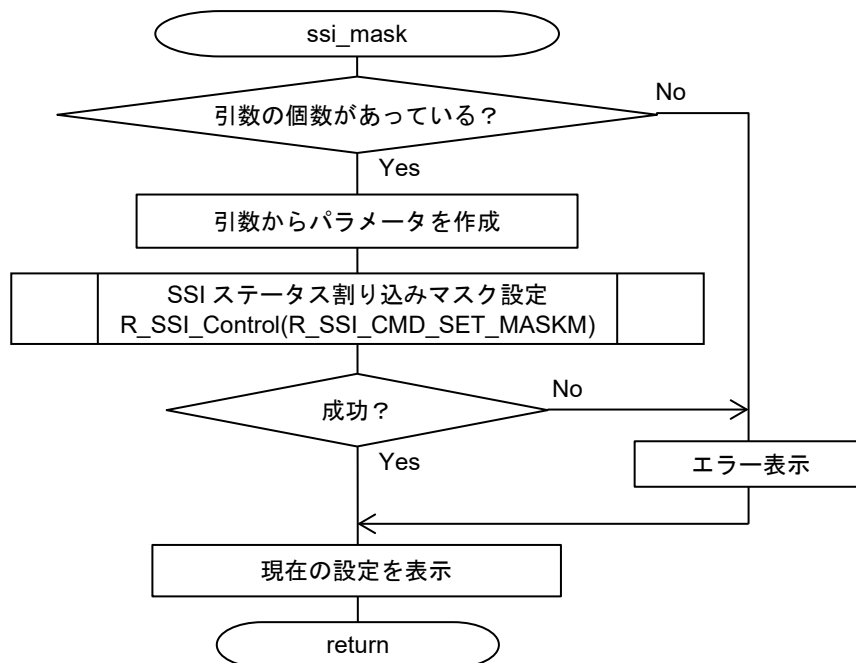


図 4-6 ssi_mask 関数のフローチャート

(5) ssi_start フローチャート

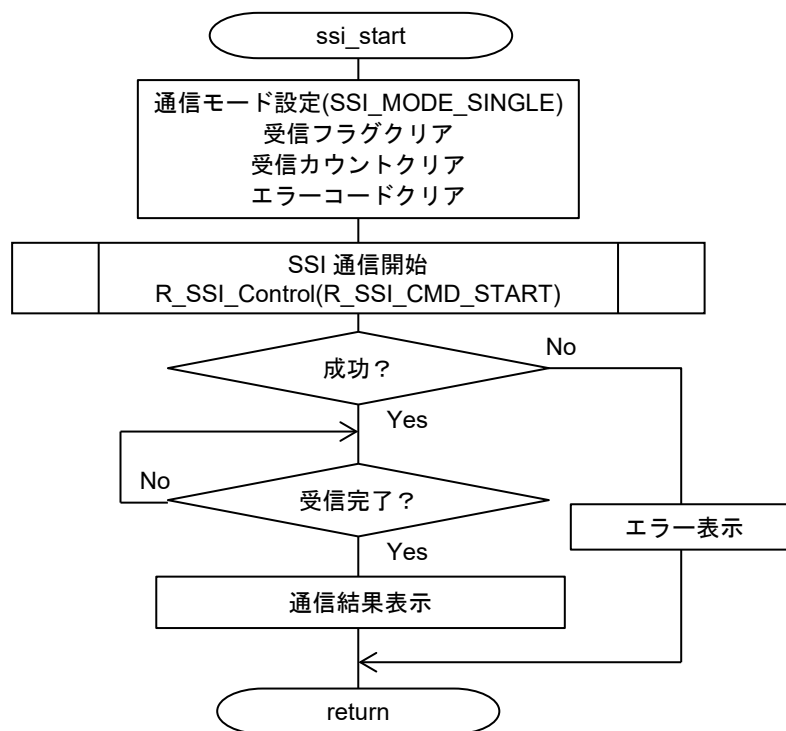


図 4-7 ssi_start 関数のフローチャート

(6) ssi_start_cont, ssi_timer_start フローチャート

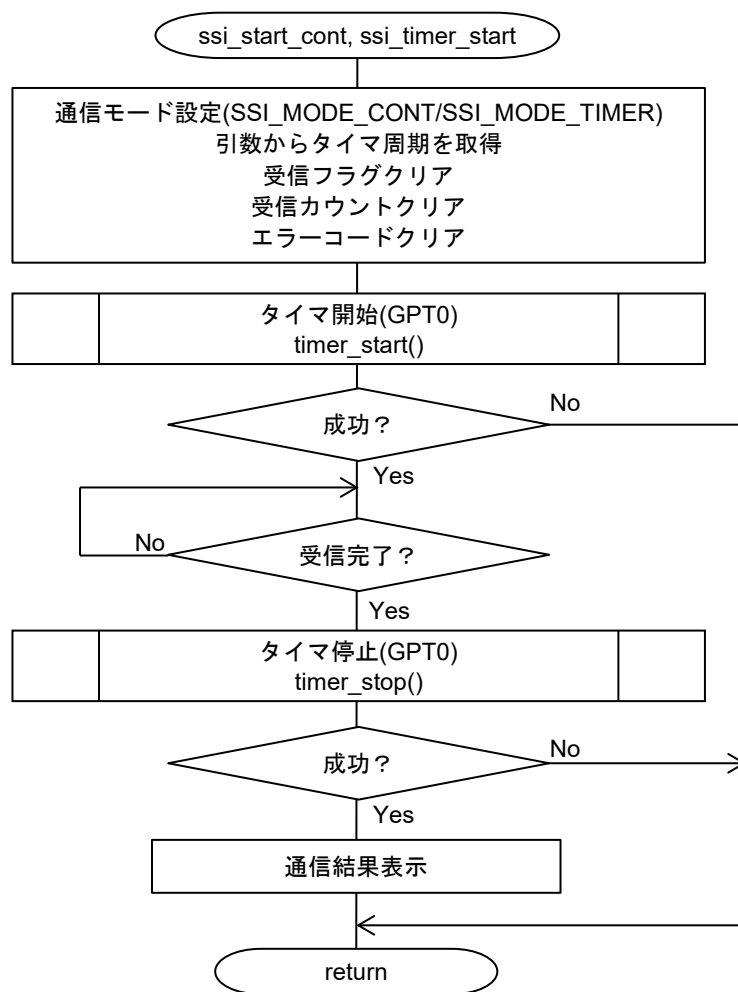


図 4-8 ssi_start_cont, ssi_timer_start 関数のフローチャート

(7) ssi_elc_start フローチャート

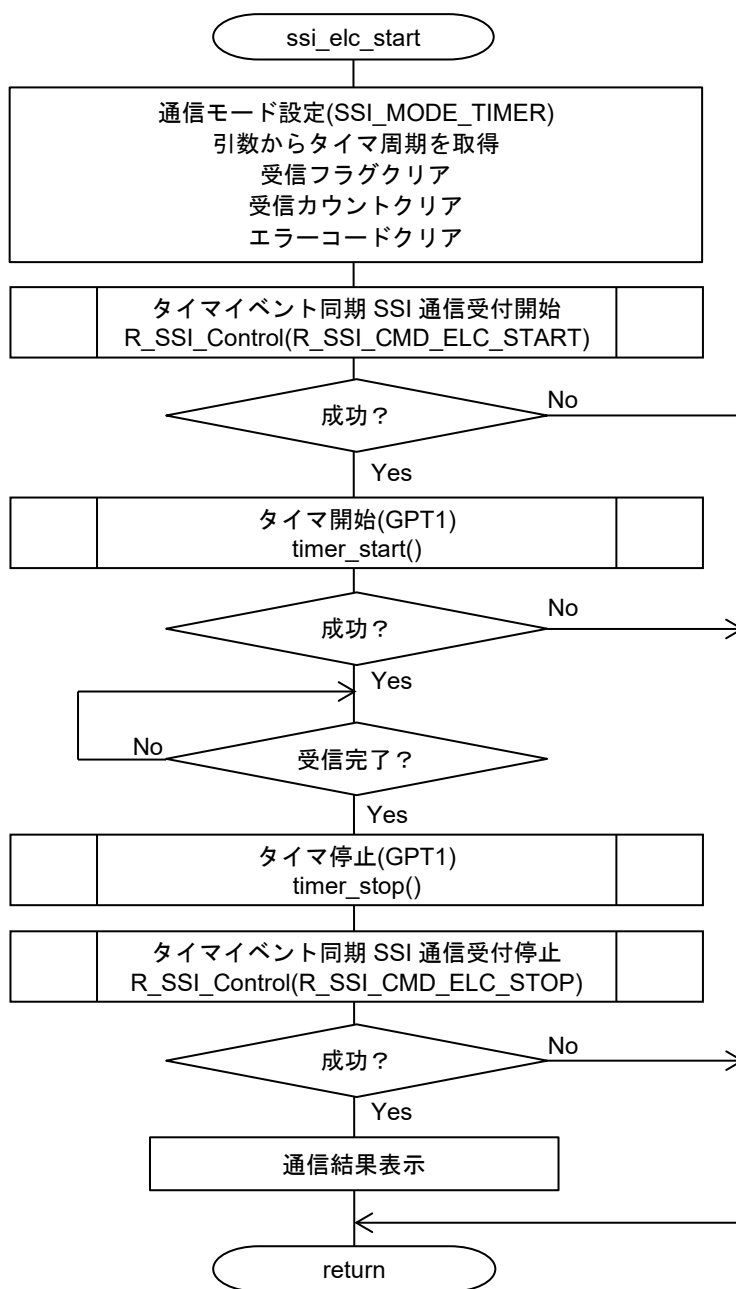


図 4-9 ssi_elc_start 関数のフローチャート

(8) ssi_exit フローチャート

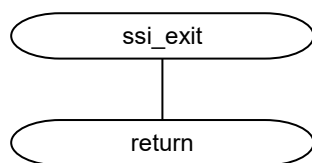


図 4-10 ssi_exit 関数のフローチャート

(9) ssi_int_ssi_callback フローチャート

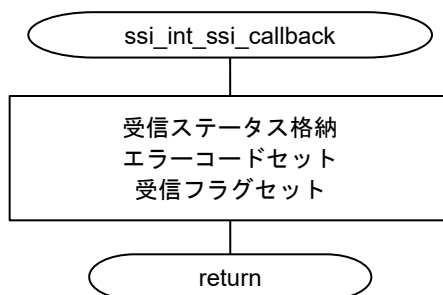


図 4-11 ssi_int_ssi_callback 関数のフローチャート

(10) ssi_int_fifo_callback フローチャート

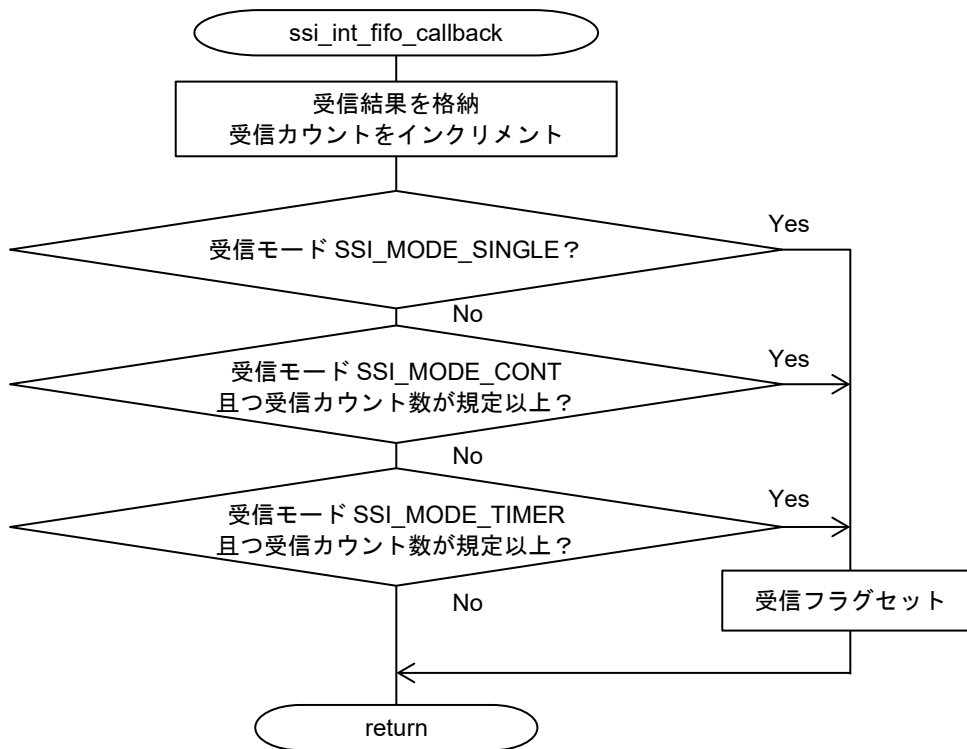


図 4-12 ssi_int_fifo_callback 関数のフローチャート

(11) get_cmd フローチャート

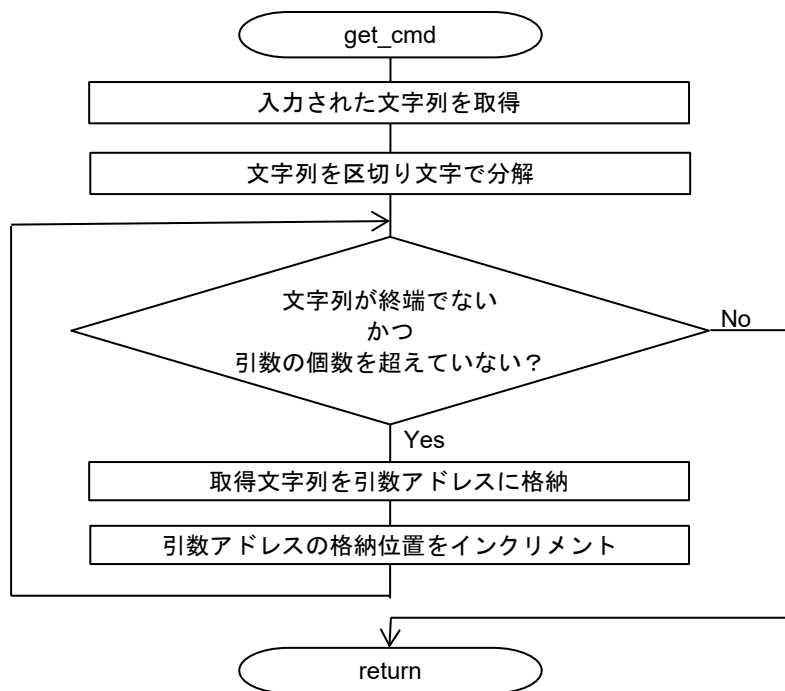


図 4-13 get_cmd 関数のフローチャート

(12) show_all フローチャート

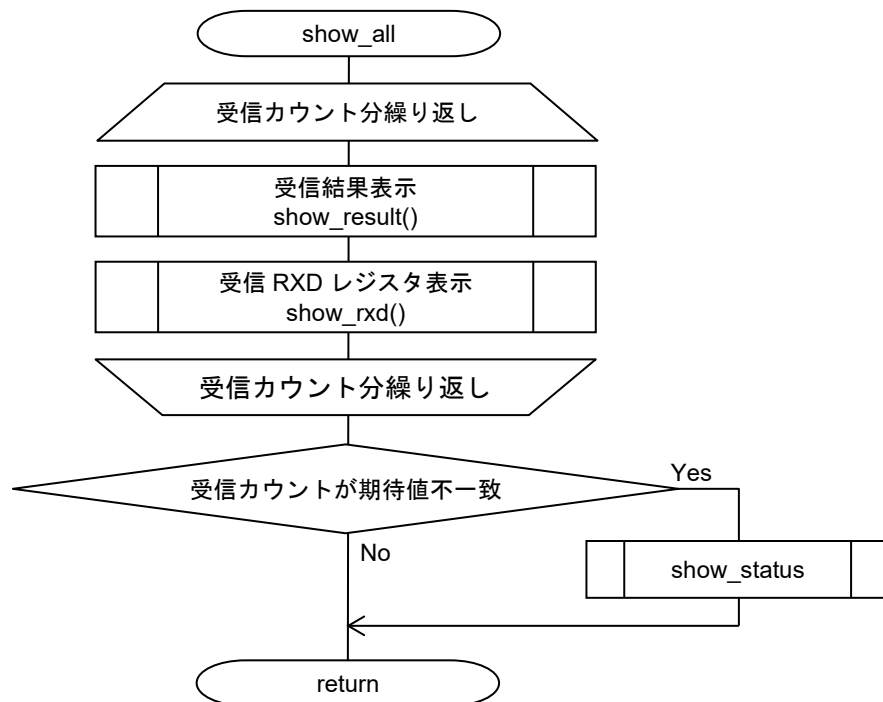


図 4-14 show_all 関数のフローチャート

(13) show_result, show_status, show_rxd フローチャート

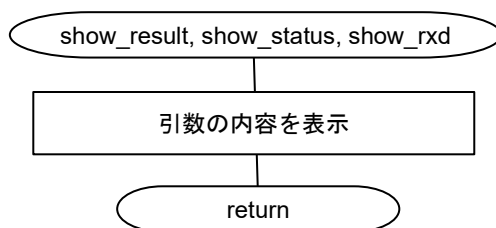


図 4-15 show_result, show_status, show_rxd 関数のフローチャート

(14) r_g_timer0_callback フローチャート

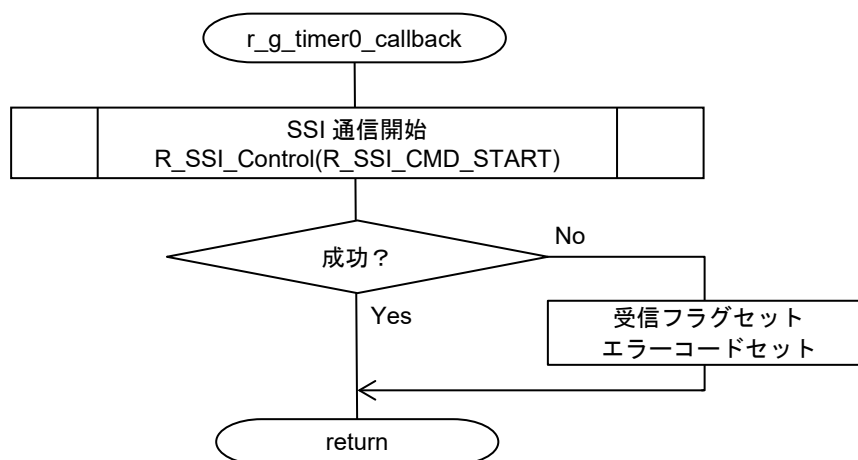


図 4-16 r_g_timer0_callback 関数のフローチャート

(15) timer_start フローチャート

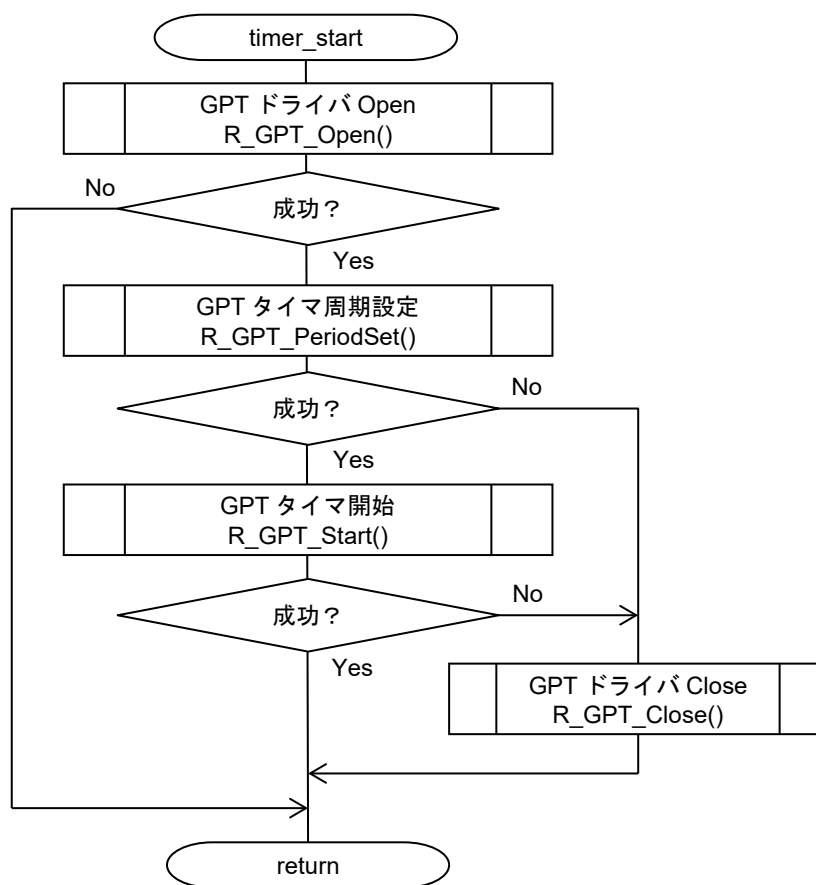


図 4-17 timer_start 関数のフローチャート

(16) timer_stop フローチャート

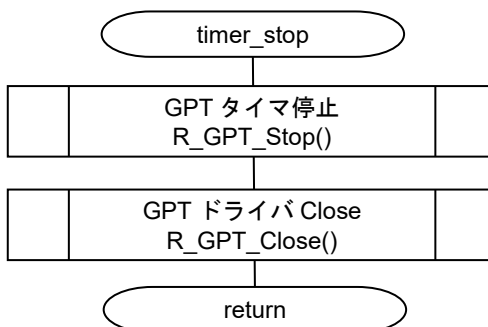


図 4-18 timer_stop 関数のフローチャート

4.11.7 動作シーケンス

(1) 開始シーケンス

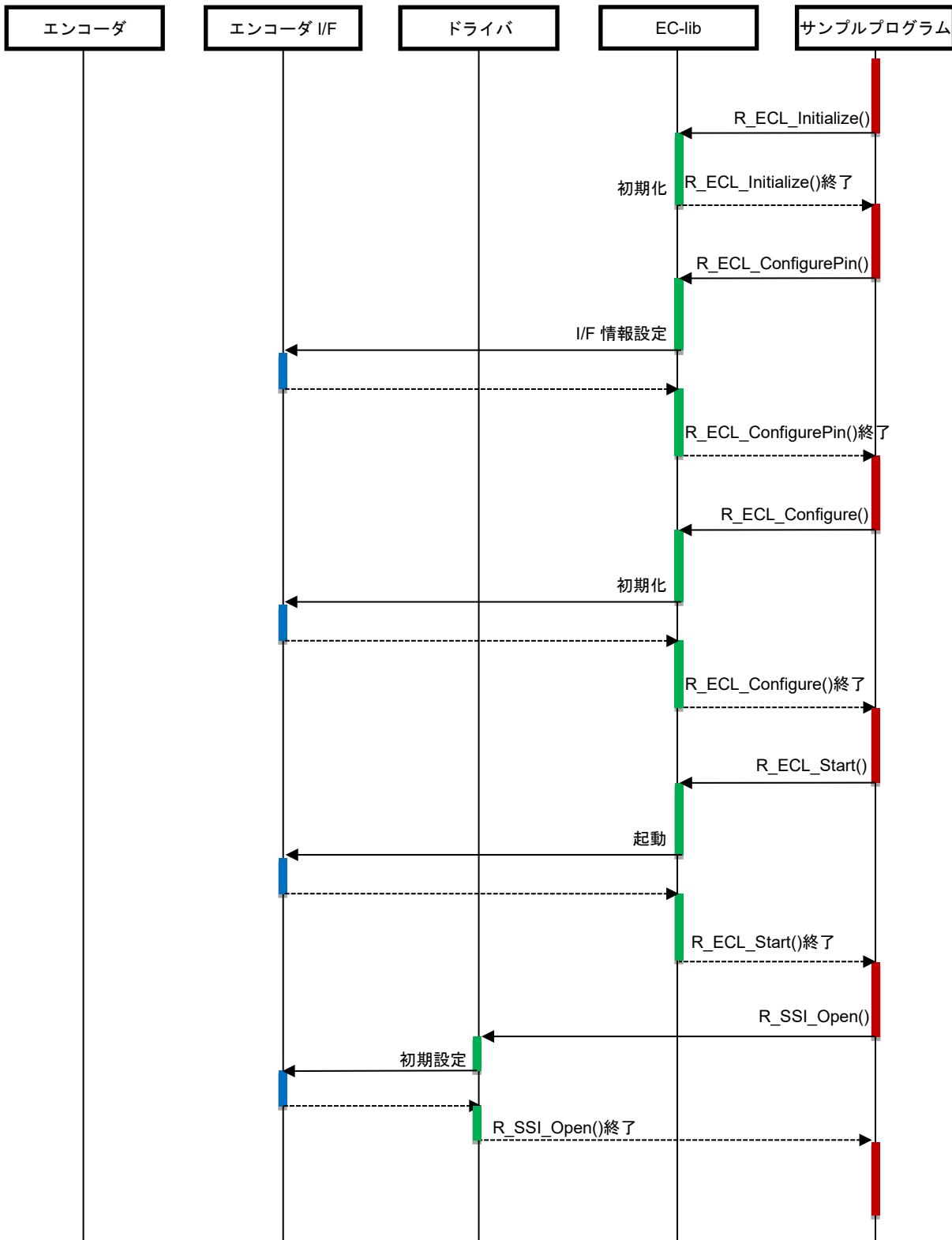


図 4-19 開始シーケンス図

(2) データ受信のシーケンス

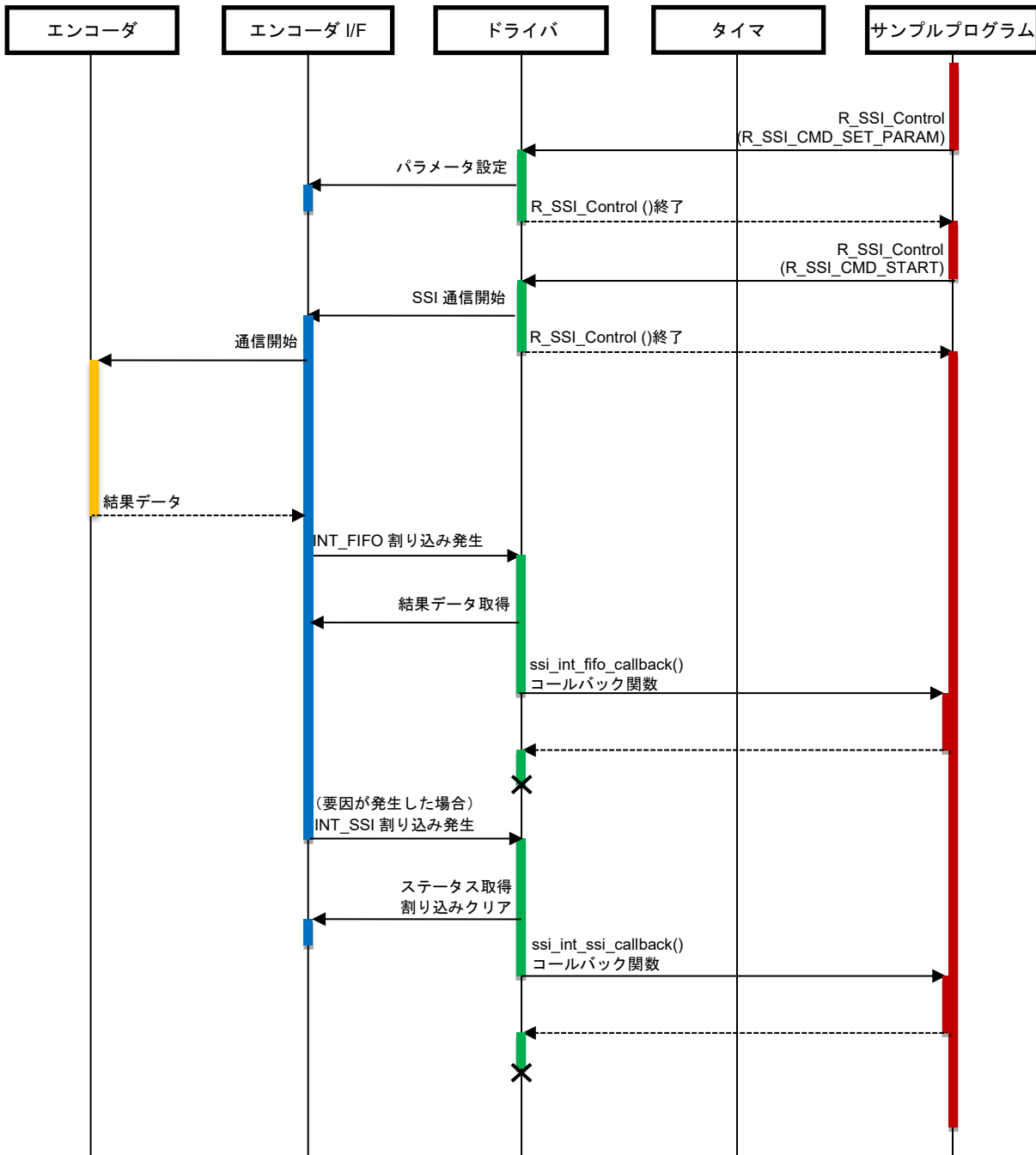
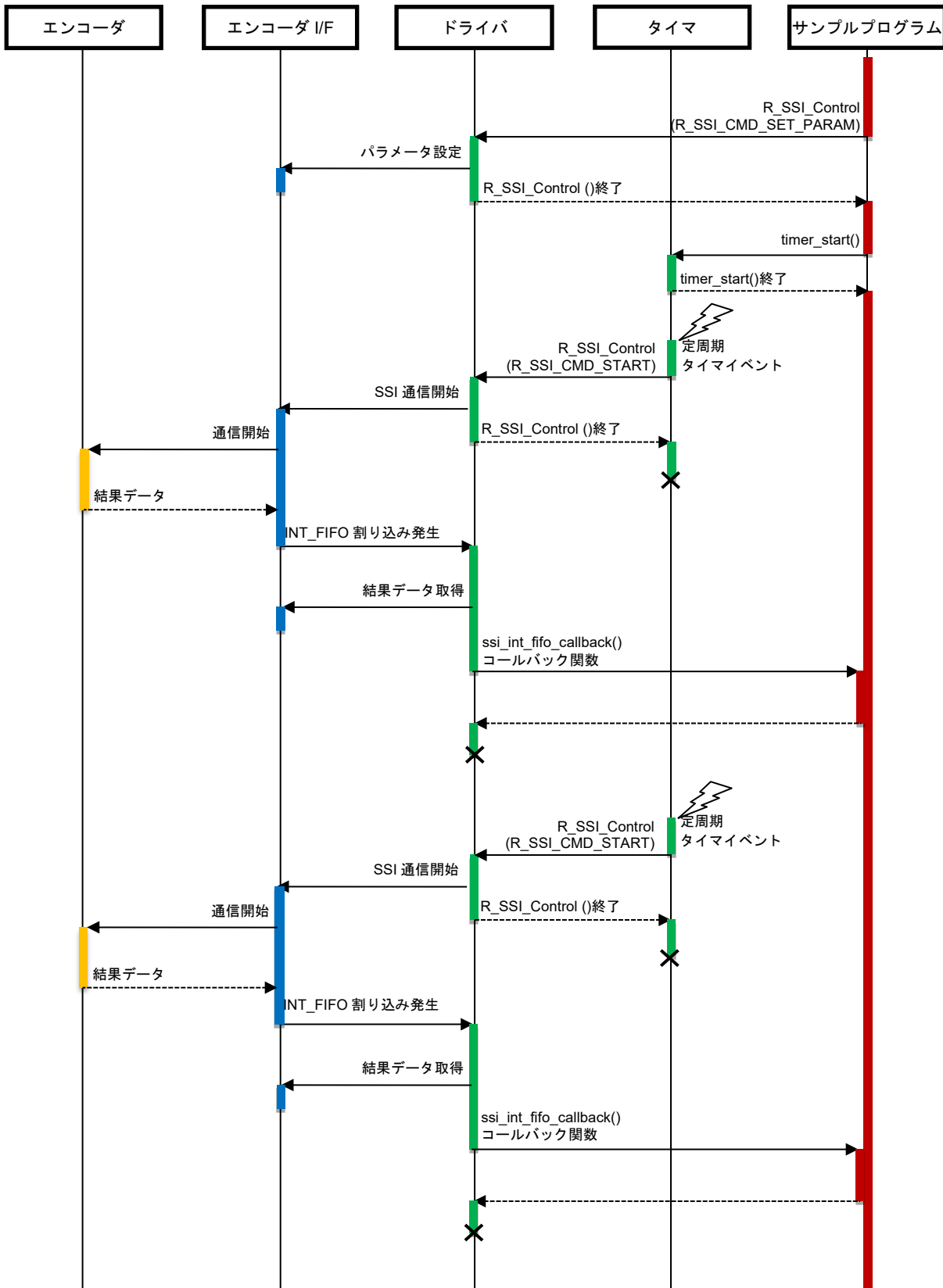


図 4-20 データ受信のシーケンス図

(3) データ受信(タイマ動作)のシーケンス



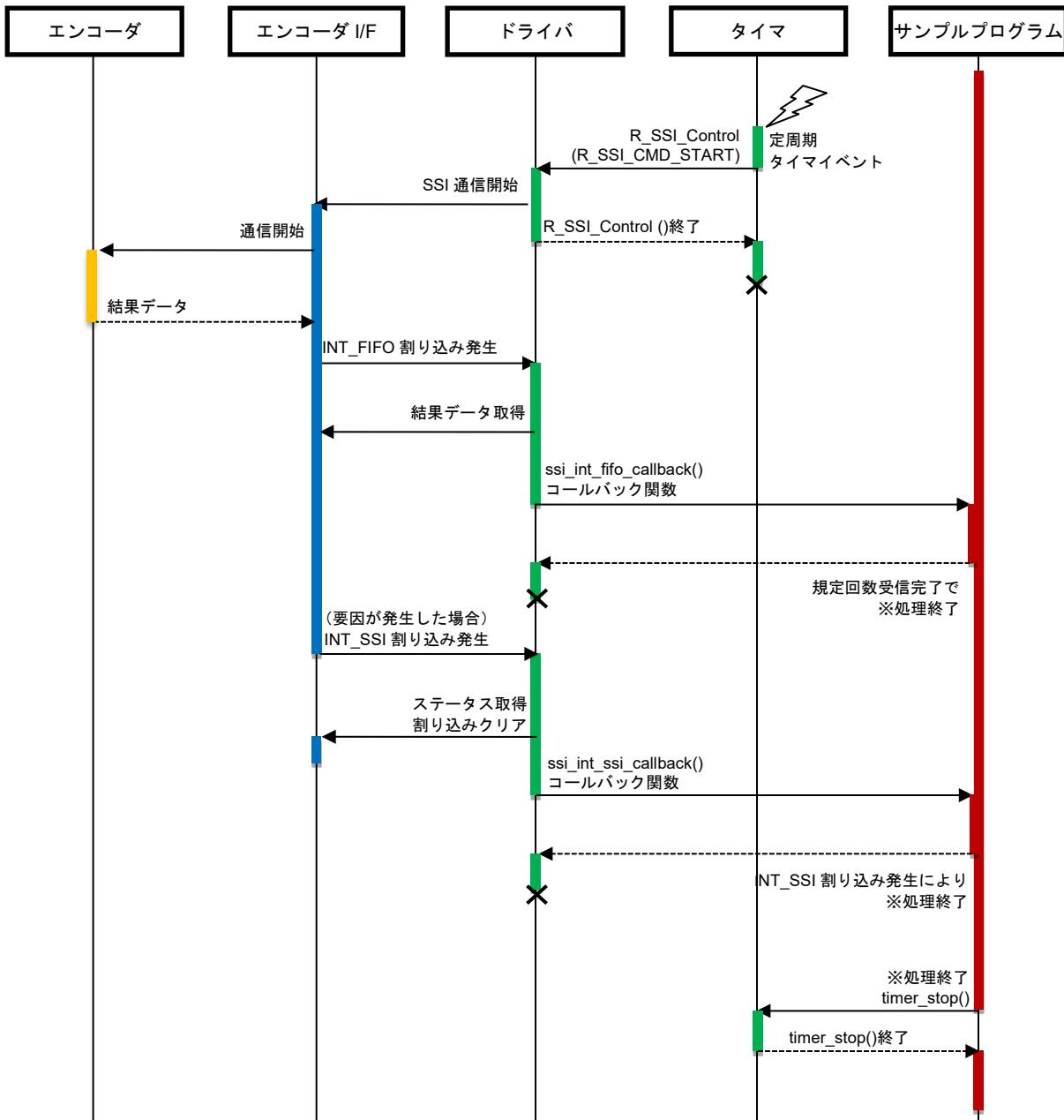


図 4-21 データ受信(タイマ動作)のシーケンス図

(4) データ受信(ELC 動作)のシーケンス

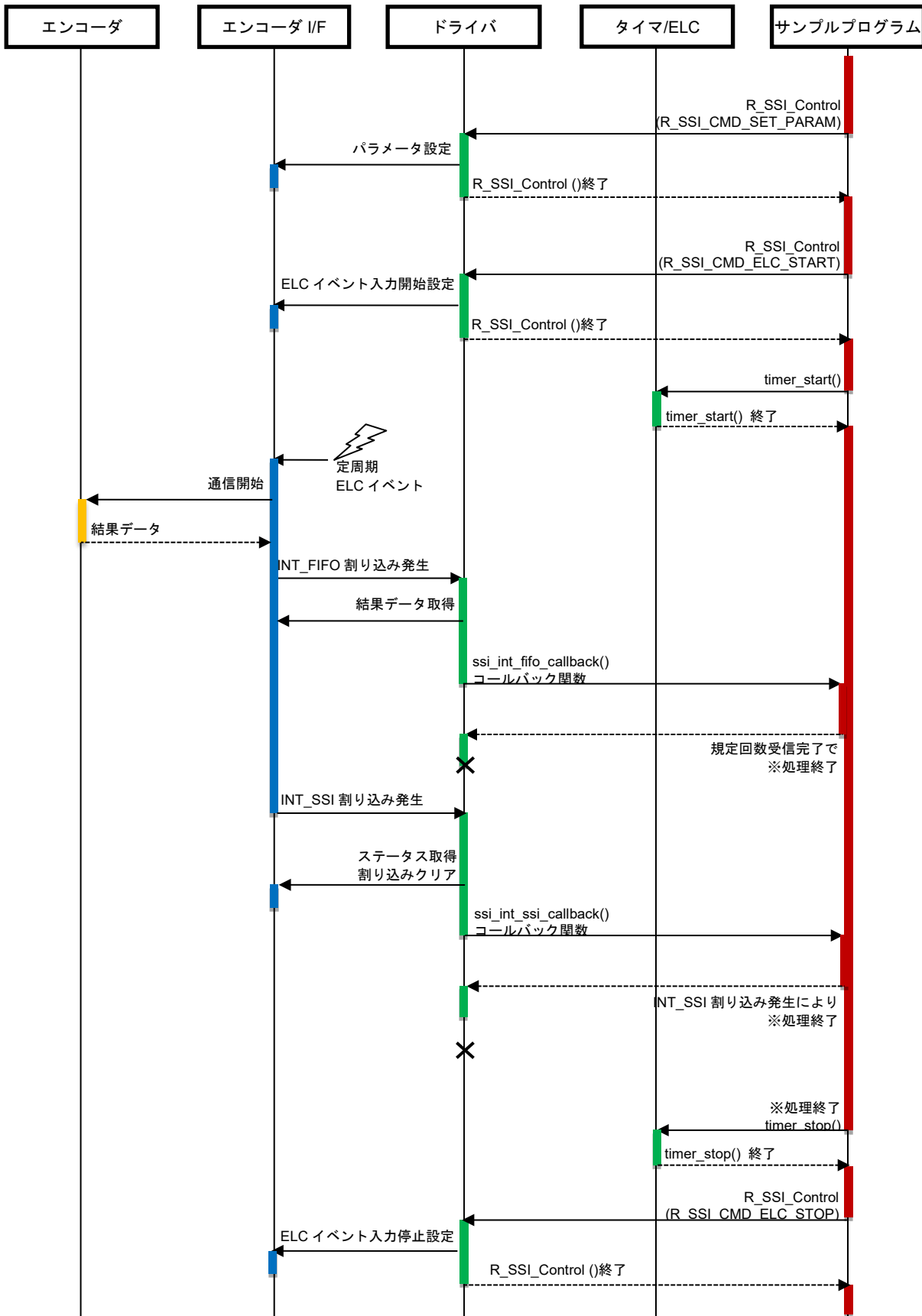


図 4-22 データ受信(ELC 動作)のシーケンス図

(5) 停止シーケンス

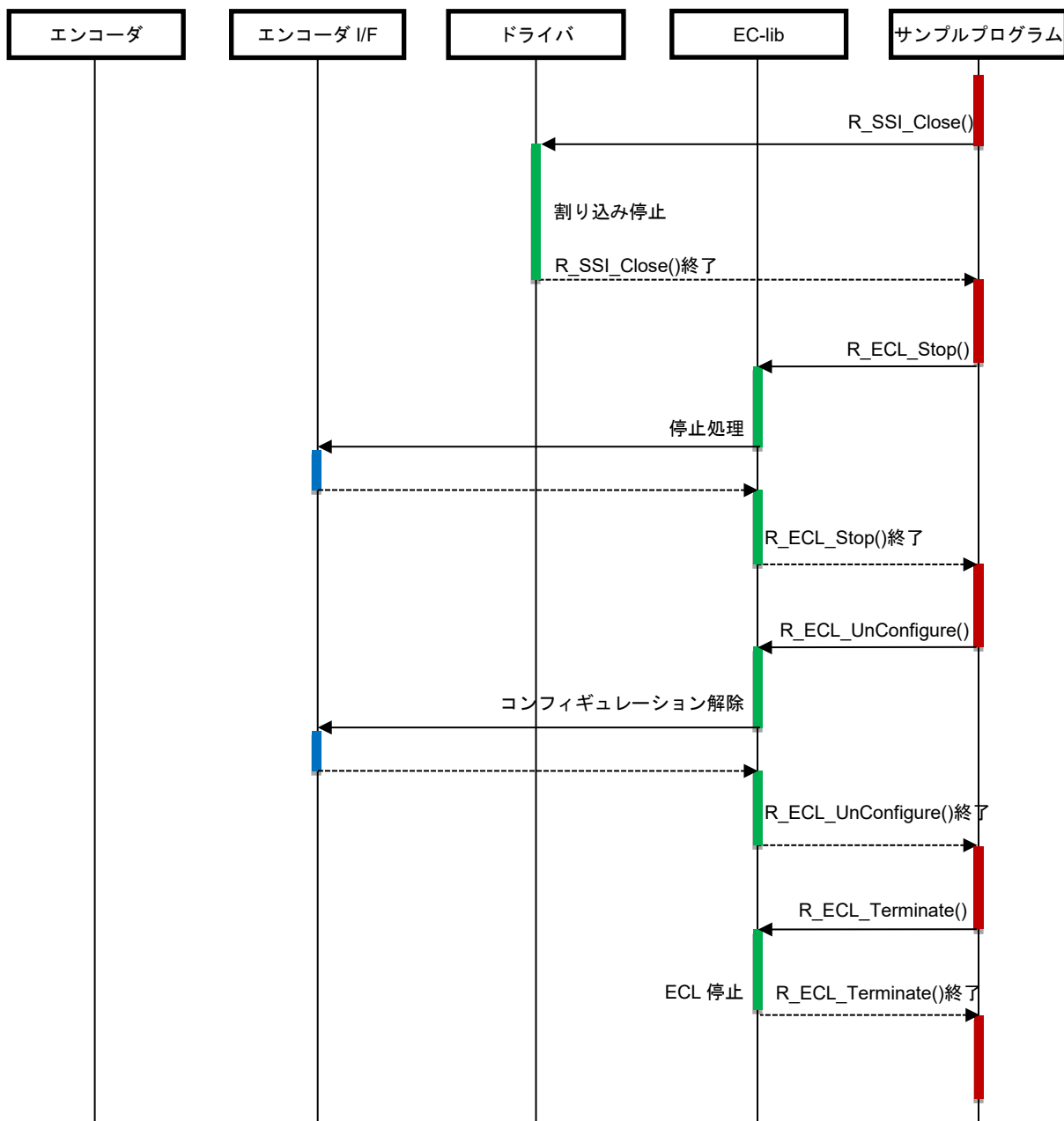


図 4-23 停止シーケンス図

4.11.8 コンソールコマンド

本サンプルプログラムは TR electronic 社製エンコーダ「CDV75M-00001」および Kübler 社製エンコーダ「8.5853FS3.1232.B723」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 4.14 コンソールコマンド一覧

コマンド	内容
br [bitrate]	ビットレートの設定を行います。 bitrate: ビットレートを指定します。 引数がない場合、入力可能な一覧と現在の設定値が表示されます。
fld [fldsize0] [fldsize1] .. [fldsize7]	フィールド数と各フィールドのビット数の設定を行います。 フィールドを指定した個数によりフィールド数が設定されます。 fldsize0: 0 番目のフィールドのビット数を指定します。 fldsize1: 1 番目のフィールドのビット数を指定します。 ... fldsize7: 7 番目のフィールドのビット数を指定します。 引数がない場合、現在の設定値が表示されます。
gry [gry0] [gry1] .. [gry7]	グレイコード/バイナリ変換設定を行います。 gry0: 0 番目のフィールドの変換有無を指定します。(1: 変換有、0: 変換無) gry1: 1 番目のフィールドの変換有無を指定します。(1: 変換有、0: 変換無) ... gry7: 7 番目のフィールドの変換有無を指定します。(1: 変換有、0: 変換無) 引数がない場合、現在の設定値が表示されます。
mask [fifoerr] [errflg] [cont] [end]	SSI ステータス割り込みの設定を行います。 fifoerr: FIFO エラーフラグの割り込みマスクを指定します。(1: 割り込み有効、0: 無効) errflg: 受信後のエラーフラグの割り込みマスクを指定します。(1: 割り込み有効、0: 無効) cont: 継続ステータスの割り込みマスクを指定します。(1: 割り込み有効、0: 無効) end: 受信完了の割り込みマスクを指定します。(1: 割り込み有効、0: 無効) 引数がない場合、現在の設定値が表示されます。
start	エンコーダに対し通信を行い、結果を表示します。
start_cont [period]	エンコーダに対し 2 回連続で通信を行い、結果を表示します。 period: 通信間隔を指定します。[usec] 引数が無ければデフォルト 1000 [usec]の間隔で通信を行います。
timer_start [period]	エンコーダに対し 5 回連続で通信を行い、結果を表示します。 period: 通信間隔を指定します。[usec] 引数が無ければデフォルト 1000 [usec]の間隔で通信を行います。
elc_start [period]	ELC を使用し、エンコーダに対し 5 回連続で通信を行い、結果を表示します。 period: 通信間隔を指定します。[usec] 引数が無ければデフォルト 1000 [usec]の間隔で通信を行います。
exit	プログラム終了

(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。"ssi >"に続けてコマンドを入力してください。

```
SSI sample program start
R_SSI_GetVersion = 4.0
R_SSI_GetFirmVersion = 1.0
ssi >
```

(2) コマンド実行例

コンソールコマンド "start" を実行した例です。エンコーダからの応答に基づき、ステータス情報とともに、角度データが表示されます。

```
ssi >start
start command
- 0 -----
FLD0 : 0x0000006D
FLD1 : 0x00000E8B
FLD2 : 0x00000000
FLD3 : 0x00000000
FLD4 : 0x00000001
FLD5 : 0x00000037
FIFOERR : 0
ERRFLG : 0
CONT : 0
END : 1
STANDBY : 0
RXD0:FLD0 12bit : 0x0000006D
RXD1:FLD1 13bit : 0x00000E8B

ssi >
```

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
Rev.1.00	Jun.28.22	-	初版発行
Rev.2.00	Jun 14.24	5	使用ボードの表記を更新
Rev.3.00	Oct 17.25	1, 4, 5 8 - 13 49	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 コマンド実行例を追加
Rev.4.00	Apr 03.26	9 - 14, 25 - 30 14, 15 16	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」 「4.6 割り込みハンドラ」 のヘッダ部分を修正 SSI 通信コントローラの動作周波数を 50.0MHz に変更

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しないでください。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えば、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、ノイズ輻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。

- 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まれます。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、変更、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、変更、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）、交通管制（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、これに限られません。）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た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。