RZ/T1 グループ

R01AN3539JJ0100
Rev.1.00
2016.12.22

CMSIS-RTOS RTX for Cortex-R4

CMT(W) & ELC & ADC サンプルプログラム

(EWARM / ICCARM, e2studio/RenesasGCC, DS-5/ARMCC)

要旨

本アプリケーションノートでは、RZ/T1 の 32 ビットタイマ(CMTW)、およびイベントリンクコントローラ(ELC)を用いて、タイマと ADC 機能を連動させ A/D 変換を行うサンプルプログラムについて説明します。

CMT(W)&ELC&ADC サンプルプログラムの特長を以下に示します。

・3 秒周期でタイマのコンペアマッチが発生します。

・タイマのコンペアマッチ発生時、ポテンショメータの入力電圧を A/D 変換します。

対象デバイス

RZ/T1

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

以下に、使用する周辺機能と用途を示します。

表 1-1 使用する周辺機能と用途

周辺機能	用途
RZ/T1 内蔵コンペアマッチタイマ(CMT)	16bit タイマカウンタで、設定カウント経過で割り込み出力や ELC へのイベント発行を実行
RZ/T1 内蔵コンペアマッチタイマ W(CMTW)	32bit タイマカウンタで、設定カウント経過で割り込み出力や ELC へのイベント発行を実行
RZ/T1 内蔵イベントリンクコントローラ(ELC)	イベントリンク元のモジュールからイベントを受付け、イベントリンク先モジュールへの連係動作を実行
消費電力低減機能	CMTW モジュール、ELC モジュールへのクロック供給／停止
割り込みコントローラ (ICUA)	●CMTW の割り込み制御(Unit0/Unit1) コンペアマッチ(ベクタ 25/30) インプットキャプチャ 0(ベクタ 26/31) インプットキャプチャ 1 (ベクタ 27/32) アウトプットコンペア 0 (ベクタ 28/33) アウトプットコンペア 1 (ベクタ 29/34) ●ELC 割り込みの制御 割り込み要求信号 1(ベクタ 242) 割り込み要求信号 2(ベクタ 243)
ADC（AN007）	ポテンショメータの入力電圧の A/D 変換
ポテンショメータ	ADC へ可変した電圧を入力

以下の基本的な内容は『RZ/T1 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。

・イベントリンクで使用する動作モード

・コンペアマッチタイマ W（CMTW）

・イベントリンクコントローラ（ELC）

・消費電力低減機能

・割り込みコントローラ（ICUA）

・I/O ポート

・マルチファンクションピンコントローラ（MPC）

・12 ビット A/D コンバータ（S12ADCa）

2. 動作環境

本アプリケーションノートのサンプルプログラムは、下記の条件で動作を確認しています。

表 2-1 動作環境

項目	内容
使用マイコン	RZ/T1 グループ
動作周波数	CPUCLK = 450MHz
動作電圧	3.3V
統合開発環境	IAR システムズ製 Embedded Workbench for ARM Version 7.80.2 （以下、EWARM と略す） RENESAS 製 e2studio 5.2.0.020（以下、e2studio と略す） ARM 製 DS-5 Version: 5.25.0（以下、DS-5 と略す）
動作モード	SPI ブートモード（シリアルフラッシュからのブート） SW4：ON/ON/ON
	RAM ブートモード（JTAG-ICE を用いた内蔵 RAM への直接ダウンロードによるブート）、または、16 ビットバスブートモード（NOR フラッシュからのブート） SW4：ON/OFF/ON
使用ボード	RZ/T1 CPU ボード (RTK7910018C00000BE)
使用デバイス （ボード上で使用する機能）	• NOR フラッシュメモリ (CS0、CS1 空間に接続) メーカ名: Macronix International Co., Ltd. 型名: MX29GL512FLT2I-10Q • SDRAM (CS2、CS3 空間に接続) メーカ名: Integrated Silicon Solution Inc. 型名: IS42S16320D-7TL • シリアルフラッシュメモリ メーカ名: Macronix International Co., Ltd. • 型名: MX25L51245GMI-10G • ポテンショメータ(AN007)
ICE	IAR システムズ製 I-jet JTAG エミュレータ SEGGER 社製 J-Link JTAG エミュレータ ARM 社製 KEIL ULINK2 エミュレータ

3. ドキュメント

3.1 関連文書一覧

本書に関連する文書を以下に示します。

- CMSIS-RTOS compliant Kernel Version 4.74

  本システムで使用している RTOS の仕様書です。サンプルアプリケーションは、CMSIS-RTOS RTX の機能を使用します。また、各サンプルアプリケーションは CMSIS-RTOS RTX を初期化します。

- RZ/T1 グループ ユーザーズマニュアル ハードウェア編（R01UH0483）

  RZ/T1 のハードウェア仕様を記載しています。

  （最新版をルネサス エレクトロニクスホームページから入手してください。）

- RZ/T1 グループ 初期設定 アプリケーションノート（R01AN2554）

  RZ/T1 の初期設定について記載しています。本書に記載のないものは、この「RZ/T1 グループ 初期設定アプリケーションノート」で設定した値を使用します。

  （最新版をルネサス エレクトロニクスホームページから入手してください。）

- RZ/T1 グループ CMSIS-RTOS RTX for Cortex-R4 RTX サンプルプログラム アプリケーションノート（R01AN3538JJ）

  RZ/T1 上で動作する CMSIS-RTOS RTX サンプルアプリケーションについて記載しています。

- テクニカルアップデート／テクニカルニュース

  （最新の情報をルネサス エレクトロニクスホームページから入手してください。）

- 開発環境に関わるユーザーズマニュアル

  IAR 統合開発環境（IAR Embedded Workbench for ARM）に関しては、最新版を IAR ホームページから入手してください。

  DS-5 統合開発環境（ARM Development Studio 5）に関しては、最新版を ARM ホームページから入手してください。

  ルネサス エレクトロニクスソフトウェア開発ツール（e2studio 等）に関しては、ルネサスエレクトロニクスホームページから最新版を入手してください。

4. ハードウェア説明

4.1 ハードウェア構成例

以下に、RZ/T1 CPU ボードを用いた基本的な接続例を示します。

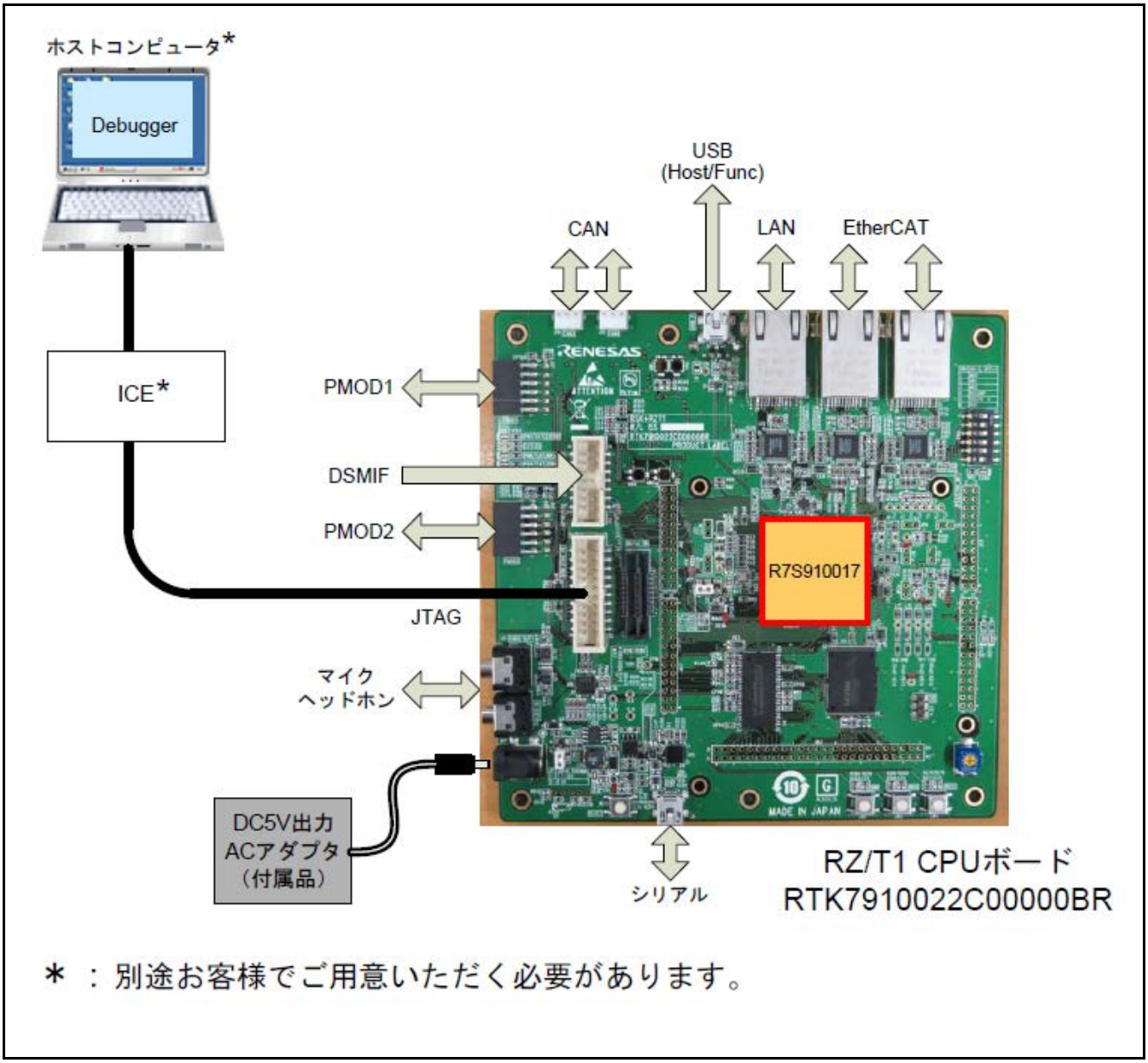

図 4.1 動作環境

以下に、本サンプルプログラムで使用されるハードウェア構成例を示します。

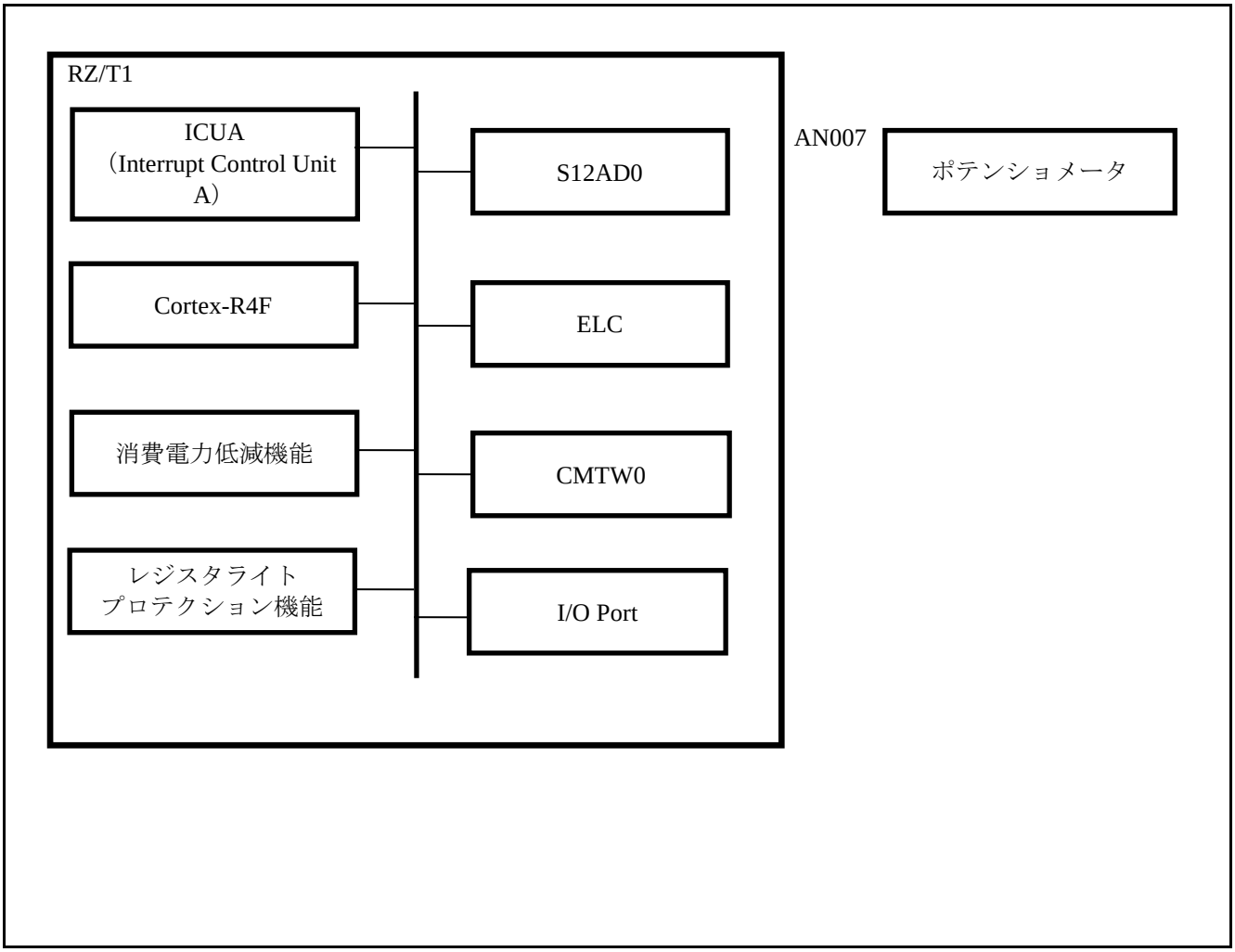

図 4.2　ハードウェア構成例

4.2　使用端子一覧

以下に使用端子と機能を示します。

表 4-1　使用端子と機能

端子名	入出力	内容
MD0	入力	動作モードの選択 MD0=“L”、MD1=“L”、MD2=“L” (SPI ブートモード) MD0=“L”、MD1=“H”、MD2=“L” (RAM ブートモード、または、16 ビットバスブートモード)
MD1	入力	
MD2	入力	
AN007	入力	ポテンショメータ

5. ソフトウェア説明

5.1 動作概要

以下に、本サンプルプログラムの動作概要、システムブロック図を示します。

表 5-1 動作概要

機能	概要
動作概要	・下記 1~4 を繰り返し実行 1. CMTW コンペアマッチによるイベント発生 2. ELC による ADC へのイベント振り分け 3. ADC での A/D 変換 4. A/D 変換結果を printf で表示する
チャネル番号（CMTW）	・ch0 を選択
動作モード（CMTW）	・コンペアマッチのみ設定
コンペアマッチ周期 （CMTW）	・約 3 秒 （タイマカウンタの入力クロック：PCLKD/8、コンペアマッチカウント設定：28125000）
タイマカウンタクリア要因 （CMTW）	・コンペアマッチ タイマカウンタ周期で動作
リンク元イベント（ELC）	・CMTW/チャネル 0/コンペアマッチ
リンク先イベント（ELC）	・S12AD0
A/D 変換開始条件	・同期トリガ起動
ADC ドライバ設定	・下記を設定 入力チャネル　：AN007 動作モード　：シングルスキャンモード 変換開始トリガ　：同期トリガ起動（ELC）

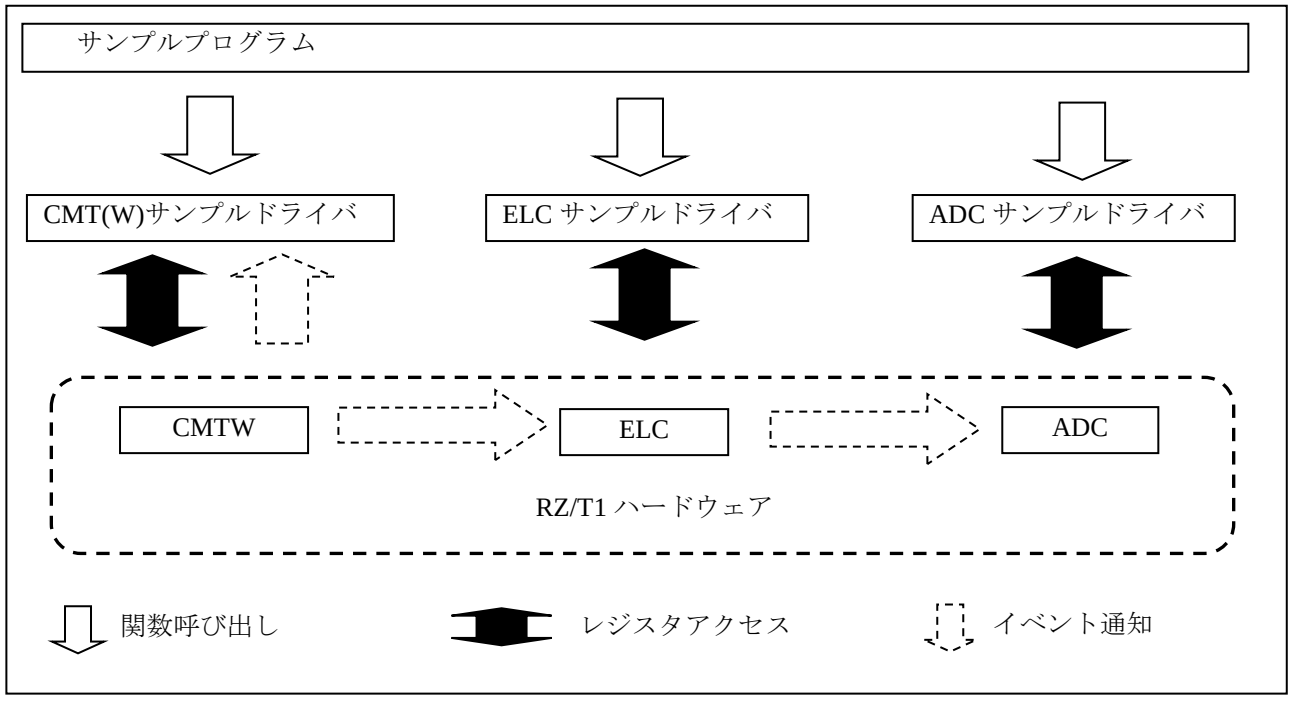

図 5.1　システムブロック図

5.2 プロジェクト設定

開発環境となるEWARM / e2studio / DS-5上で使用されるプロジェクト設定については、「RZ/T1グループ 初期設定 アプリケーションノート」に記載しています。

5.3 メモリマップ

RZ/T1 グループのアドレス空間と RZ/T1 CPU ボードのメモリマッピングについては「RZ/T1 グループ 初期設定 アプリケーションノート」に記載しています。

5.3.1 サンプルプログラムのセクション配置

サンプルプログラムで使用するセクションおよびサンプルプログラムの初期状態のセクション配置（ロードビュー）、スキャッタローディング機能を使用後のセクション配置（実行ビュー）は、「RZ/T1 グループ 初期設定 アプリケーションノート」に記載しています。

5.3.2 MPU の設定

MPU の設定は、「RZ/T1 グループ 初期設定 アプリケーションノート」に記載しています。

5.3.3 例外処理ベクタテーブル

例外処理ベクタテーブルについては、「RZ/T1 グループ 初期設定 アプリケーションノート」に記載しています。

5.3.4 必要メモリサイズ

必要メモリサイズは、RTX_ex1 サンプルプログラムの例が、「RZ/T1 グループ CMSIS-RTOS RTX for Cortex-R4 RTX サンプルプログラム アプリケーションノート」に記載されています。詳細な必要メモリサイズは、実際のプロジェクトを参照してください。

5.4 フォルダ構成

サンプルプログラム、各サンプルドライバのフォルダ構成を以下に示します。

```
[Drivers]          RTOS¥RTX¥Boards¥Renesas¥RZ_T1_RSK¥Renesas_example¥Drivers フォルダ
    │
    ├──[drv_example]
    │      │
    │      └──[RTX_CMT_ex1]          本サンプルプログラムソースコード
    │
    ├──[drv_inc]                     各サンプルドライバのヘッダファイル
    │
    └──[drv_src]                     各サンプルドライバのソースコード
           │
           ├──[adc]                  ADC サンプルドライバのソースコード
           │
           ├──[cmt]                  CMT サンプルドライバのソースコード
           │
           └──[elc]                  ELC サンプルドライバのソースコード
```

5.5 使用割り込み一覧

表 5-2 サンプルプログラムで使用する割り込み

割り込み(要因 ID)	優先度	処理概要
コンペアマッチ割り込み(25)	7	コールバック関数をコール
A/D 変換終了割り込み(35)	7	コールバック関数をコール

5.6 固定幅整数一覧

以下に、サンプルプログラムで使用する固定幅整数を示します。

表 5-3 サンプルプログラム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）

5.7 定数/エラーコード一覧

以下に、サンプルプログラム / サンプルドライバで使用する定数、エラーコードを記します。

5.7.1 サンプルプログラムで使用する定数

表 5-4 サンプルプログラムで使用する定数

定数名	設定値	内容
CMTW_MATCH_COUNT_3SEC	28125000	コンペアマッチのカウンタ値を設定します。サンプルプログラムで PCLKD の分周比設定との組み合わせでコンペアマッチ周期が約 3 秒となる値を設定しています。

5.7.2 サンプルプログラムで使用するエラーコード定数

表 5-5 サンプルプログラムで使用するエラーコード定数

定数名	設定値	内容
ESUCCESS	0	成功
EINVAL	-28	引数の不正
EACCES	-64	アクセス拒否
EBUSY	-67	ビジー
EFAULT	-97	失敗

5.7.3 ADC サンプルドライバの定数

表 5-6 バージョン定義定数（ADC）

定数名	値	内容
ADC_VERSION_MAJOR	1	ADC サンプルドライバのメジャーバージョン
CMT_VERSION_MINOR	0	ADC サンプルドライバのマイナーバージョン

表 5-7 マスク定義定数（ADC）

定数名	値	内容
ADC_MASK_GROUPB_OFF	0	グループ B のマスク解除
ADC_MASK_ADD_OFF	0	追加チャンネル／マスク解除
ADC_MASK_CH0	(1<<0)	チャンネル 0 マスク
ADC_MASK_CH1	(1<<1)	チャンネル 1 マスク
ADC_MASK_CH2	(1<<2)	チャンネル 2 マスク
ADC_MASK_CH3	(1<<3)	チャンネル 3 マスク
ADC_MASK_CH4	(1<<4)	チャンネル 4 マスク
ADC_MASK_CH5	(1<<5)	チャンネル 5 マスク
ADC_MASK_CH6	(1<<6)	チャンネル 6 マスク
ADC_MASK_CH7	(1<<7)	チャンネル 7 マスク

表 5-8 サンプリング時間（SST）定数（ADC）

定数名	値	内容
ADC_SST_CNT_MIN	5	最小サンプリング時間
ADC_SST_CNT_MAX	255	最大サンプリング時間
ADC_SST_CNT_DEFAULT	11	サンプリング時間デフォルト値

5.7.4 CMT(W)サンプルドライバの定数

表 5-9 バージョン定義定数（CMT(W)）

定数名	値	内容
CMT_VERSION_MAJOR	1	CMT(W)サンプルドライバのメジャーバージョン
CMT_VERSION_MINOR	0	CMT(W)サンプルドライバのマイナーバージョン

表 5-10 CMTW チャネル番号定義値（CMT(W)）

定数名	値	内容
CMTW_CHANNEL_0	6	CMT_CHANNEL_6
CMTW_CHANNEL_1	7	CMT_CHANNEL_7

※CMT_CHANNEL_6、CMT_CHANNEL_7 は 5.8.2 章参照。

表 5-11 コマンド定義（CMT(W)）

定数名	値	内容
CMT_CMD_SET_TIME_CNT	0	タイマカウント設定コマンド
CMT_CMD_SET_MODE	1	タイマモード設定コマンド
CMT_CMD_SET_PAUSE	2	一時停止コマンド
CMT_CMD_SET_RESUME	3	レジュームコマンド
CMT_CMD_SET_RESTART	4	リスタートコマンド
CMT_CMD_SET_ECM	5	ECM ダイナミックモードエラー設定
CMT_CMD_GET_STATUS	6	状態取得コマンド

表 5-12 タイマクロック分周比設定義値（CMT(W)）

定数名	値	内容
CMT_PCLK_DIV_8	0	タイマ分周比 PCLKD/8 を選択
CMT_PCLK_DIV_32	1	タイマ分周比 PCLKD/32 を選択
CMT_PCLK_DIV_128	2	タイマ分周比 PCLKD/128 を選択
CMT_PCLK_DIV_512	3	タイマ分周比 PCLKD/512 を選択値
CMT_PCLK_DIV_MAX_OVER	4	正常設定値範囲確認用

表 5-13 タイマカウンタ長設定定義値（CMT(W)）

定数名	値	内容
CMTW_CNT_SIZE_32BIT	0	カウンタサイズ 32 ビットを選択
CMTW_CNT_SIZE_16BIT	1	カウンタサイズ 16 ビットを選択
CMTW_CNT_SIZE_MAX_OVER	2	正常設定値範囲確認用

表 5-14 タイマカウンタクリア要因設定定義値（CMT(W)）

定数名	値	内容
CMT_CNT_CLEAR_COMPARE_MATCH	0	カウンタクリア要因にコンペアマッチを選択
CMTW_CNT_CLEAR_COMPARE_MATCH	0	カウンタクリア要因にコンペアマッチを選択
CMTW_CNT_CLEAR_INPUT_CAPTURE0	1	カウンタクリア要因にインプットキャプチャ 0 を選択
CMTW_CNT_CLEAR_INPUT_CAPTURE1	2	カウンタクリア要因にインプットキャプチャ 1 を選択
CMTW_CNT_CLEAR_OUTPUT_COMPARE0	3	カウンタクリア要因にアウトプットコンペア 0 を選択
CMTW_CNT_CLEAR_OUTPUT_COMPARE1	4	カウンタクリア要因にアウトプットコンペア 1 を選択
CMTW_CNT_CLEAR_NONE	5	カウンタクリア要因なしを選択
CMTW_CNT_CLEAR_MAX_OVER	6	正常設定値範囲確認用

表 5-15 アウトプットコンペア出力設定定義値（CMT(W)）

定数名	値	内容
CMTW_OUT_COMPARE_VALUE_HOLD	0	アウトプットコンペア時、出力値を変更しない
CMTW_OUT_COMPARE_VALUE_0_TO_TOGGLE	1	初期値 0 で開始し、アウトプットコンペア時にトグルを行う
CMTW_OUT_COMPARE_VALUE_1_TO_TOGGLE	2	初期値 1 で開始し、アウトプットコンペア時にトグルを行う
CMTW_OUT_COMPARE_VALUE_MAX_OVER	3	正常設定値範囲確認用

表 5-16 インプットキャプチャ検出トリガ設定定義値（CMT(W)）

定数名	値	内容
CMTW_IN_CAPTURE_TRIGGER_UP	0	インプットキャプチャトリガを立ち上がりエッジに設定
CMTW_IN_CAPTURE_TRIGGER_DOWN	1	インプットキャプチャトリガを立ち下がりエッジに設定
CMTW_IN_CAPTURE_TRIGGER_UP_DOWN	2	インプットキャプチャトリガを立ち上がり/立下りエッジに設定
CMTW_IN_CAPTURE_TRIGGER_MAX_OVER	3	正常設定値範囲確認用

表 5-17 ノイズフィルタクロック分周比設定定義値（CMT(W)）

定数名	値	内容
CMTW_NOISE_FILTER_PCLK_DIV_1	0	ノイズフィルタ分周比 PCLKD/1 を選択
CMTW_NOISE_FILTER_PCLK_DIV_8	1	ノイズフィルタ分周比 PCLKD/8 を選択
CMTW_NOISE_FILTER_PCLK_DIV_32	2	ノイズフィルタ分周比 PCLKD/32 を選択
CMTW_NOISE_FILTER_PCLK_DIV_64	3	ノイズフィルタ分周比 PCLKD/64 を選択
CMTW_NOISE_FILTER_PCLK_DIV_MAX_OVER	4	正常設定値範囲確認用

表 5-18 インプットキャプチャ番号定義値（CMT(W)）

定数名	値	内容
CMTW_INPUT_CAPTURE_NUM_0	0	インプットキャプチャ 0 を選択
CMTW_INPUT_CAPTURE_NUM_1	1	インプットキャプチャ 1 を選択
CMTW_INPUT_CAPTURE_NUM	2	正常設定値範囲確認に使用、またインプットキャプチャ本数を示す

表 5-19 アウトプットコンペア番号定義値（CMT(W)）

定数名	値	内容
CMTW_OUTPUT_COMPARE_NUM_0	0	アウトプットコンペア 0 を選択
CMTW_OUTPUT_COMPARE_NUM_1	1	アウトプットコンペア 1 を選択
CMTW_OUTPUT_COMPARE_NUM	2	正常設定値範囲確認に使用、またアウトプットコンペア本数を示す

表 5-20 タイマ動作状態定義値（CMT(W)）

定数名	値	内容
CMTW_STATUS_STOP	0	タイマ停止状態を示す
CMTW_STATUS_RUNNING	1	タイマ動作状態を示す

5.7.5 ELC サンプルドライバの定数

表 5-21 ドライババージョン定義値（ELC）

定数名	値	内容
ELC_VERSION_MAJOR	1	ELC サンプルドライバのメジャーバージョン
ELC_VERSION_MINOR	0	ELC サンプルドライバのマイナーバージョン

表 5-22 コマンド定義（ELC）

定数名	値	内容
ELC_CMD_SET_EVENT_MTU	0	MTU イベントリンク設定コマンド
ELC_CMD_SET_EVENT_CMT	1	CMT イベントリンク設定コマンド
ELC_CMD_SET_EVENT_DSMIF	2	ΔΣユニットイベントリンク設定コマンド
ELC_CMD_SET_EVENT_S12AD	3	12 ビット A/D コンバータイベントリンク設定コマンド
ELC_CMD_SET_EVENT_INTR	4	ELC 割り込み要求信号イベントリンク設定コマンド
ELC_CMD_SET_EVENT_OUT_PORT_GROUP	5	出力ポートグループイベントリンク設定コマンド
ELC_CMD_SET_EVENT_IN_PORT_GROUP	6	入力ポートグループイベントリンク設定コマンド
ELC_CMD_SET_EVENT_SINGLE_PORT	7	シングルポート登録、イベントリンク設定コマンド
ELC_CMD_SET_EVENT_CMTW	8	CMTW イベントリンク設定コマンド
ELC_CMD_SET_EVENT_TPU	9	TPU イベントリンク設定コマンド
ELC_CMD_SET_EVENT_GPT	10	GPT イベントリンク設定コマンド
ELC_CMD_SET_PORT_GROUP	11	ポートグループ設定コマンド
ELC_CMD_SET_SOFTWARE_EVENT	12	ソフトウェアイベント発行コマンド
ELC_CMD_GET_PORT_GROUP_VALUE	13	入力ポートグループ値取得コマンド

表 5-23 イベントリンクリソース設定定義値（ELC）

定数名	値	内容
ELC_RESOURCE_MTU0_COMPARE_MATCH_0A	0x01	イベントリンク元を MTU0・コンペアマッチ 0A に指定
ELC_RESOURCE_MTU0_COMPARE_MATCH_0B	0x02	イベントリンク元を MTU0・コンペアマッチ 0B に指定
ELC_RESOURCE_MTU0_COMPARE_MATCH_0C	0x03	イベントリンク元を MTU0・コンペアマッチ 0C に指定
ELC_RESOURCE_MTU0_COMPARE_MATCH_0D	0x04	イベントリンク元を MTU0・コンペアマッチ 0D に指定
ELC_RESOURCE_MTU0_COMPARE_MATCH_0E	0x05	イベントリンク元を MTU0・コンペアマッチ 0E に指定
ELC_RESOURCE_MTU0_COMPARE_MATCH_0F	0x06	イベントリンク元を MTU0・コンペアマッチ 0F に指定
ELC_RESOURCE_MTU0_OVERFLOW	0x07	イベントリンク元を MTU0・オーバーフローに指定
ELC_RESOURCE_MTU3_COMPARE_MATCH_3A	0x10	イベントリンク元を MTU3・コンペアマッチ 3A に指定
ELC_RESOURCE_MTU3_COMPARE_MATCH_3B	0x11	イベントリンク元を MTU3・コンペアマッチ 3B に指定
ELC_RESOURCE_MTU3_COMPARE_MATCH_3C	0x12	イベントリンク元を MTU3・コンペアマッチ 3C に指定

定数名	値	内容
ELC_RESOURCE_MTU3_COMPARE_MATCH_3D	0x13	イベントリンク元を MTU3・コンペアマッチ 3D に指定
ELC_RESOURCE_MTU3_OVERFLOW	0x14	イベントリンク元を MTU3・オーバーフローに指定
ELC_RESOURCE_MTU4_COMPARE_MATCH_4A	0x15	イベントリンク元を MTU4・コンペアマッチ 4A に指定
ELC_RESOURCE_MTU4_COMPARE_MATCH_4B	0x16	イベントリンク元を MTU4・コンペアマッチ 4B に指定
ELC_RESOURCE_MTU4_COMPARE_MATCH_4C	0x17	イベントリンク元を MTU4・コンペアマッチ 4C に指定
ELC_RESOURCE_MTU4_COMPARE_MATCH_4D	0x18	イベントリンク元を MTU4・コンペアマッチ 4D に指定
ELC_RESOURCE_MTU4_OVERFLOW	0x19	イベントリンク元を MTU4・オーバーフローに指定
ELC_RESOURCE_MTU4_UNDERFLOW	0x1A	イベントリンク元を MTU4・アンダーフローに指定
ELC_RESOURCE_CMT1_COMPARE_MATCH_1	0x1F	イベントリンク元を CMT1・コンペアマッチ 1 に指定
ELC_RESOURCE_ETHER_TIMER_SYNC	0x22	イベントリンク元を EtherMAC・IEEE1588 タイマ同期信号に指定
ELC_RESOURCE_RIIC0_EVENT	0x4E	イベントリンク元を RIIC0・通信エラー、イベント発生に指定
ELC_RESOURCE_RIIC0_RX_DATA_FULL	0x4F	イベントリンク元を RIIC0・受信データフルに指定
ELC_RESOURCE_RIIC0_TX_DATA_EMPTY	0x50	イベントリンク元を RIIC0・送信データエンプティに指定
ELC_RESOURCE_RIIC0_TX_END	0x51	イベントリンク元を RIIC0・送信終了に指定
ELC_RESOURCE_RSPI0_ERROR	0x52	イベントリンク元を RSPI0・エラーに指定
ELC_RESOURCE_RSPI0_IDLE	0x53	イベントリンク元を RSPI0・アイドルに指定
ELC_RESOURCE_RSPI0_RX_DATA_FULL	0x54	イベントリンク元を RSPI0・受信データフルに指定
ELC_RESOURCE_RSPI0_TX_DATA_EMPTY	0x55	イベントリンク元を RSPI0・送信データエンプティに指定
ELC_RESOURCE_RSPI0_TX_END	0x56	イベントリンク元を RSPI0・送信完了に指定
ELC_RESOURCE_SA12AD0_CONVERT_END	0x58	イベントリンク元を S12AD0・A/D 変換終了に指定
ELC_RESOURCE_INPUT_PORT_GROUP1	0x63	イベントリンク元を入力ポートグループ 1・入力エッジ検出に指定
ELC_RESOURCE_INPUT_PORT_GROUP2	0x64	イベントリンク元を入力ポートグループ 2・入力エッジ検出に指定
ELC_RESOURCE_SINGLE_PORT0	0x65	イベントリンク元を入力シングルポート 0・入力エッジ検出に指定
ELC_RESOURCE_SINGLE_PORT1	0x66	イベントリンク元を入力シングルポート 1・入力エッジ検出に指定
ELC_RESOURCE_SINGLE_PORT2	0x67	イベントリンク元を入力シングルポート 2・入力エッジ検出に指定
ELC_RESOURCE_SINGLE_PORT3	0x68	イベントリンク元を入力シングルポート 3・入力エッジ検出に指定
ELC_RESOURCE_ELC_SOFT_EVENT	0x69	イベントリンク元をソフトウェアイベントに指定
ELC_RESOURCE_DOC_DATA_CALCULATE	0x6A	イベントリンク元を DOC・データ演算条件成立に指定
ELC_RESOURCE_SA12AD1_CONVERT_END	0x6C	イベントリンク元を S12AD1・A/D 変換終了に指定

定数名	値	内容
ELC_RESOURCE_CMTW0_COMPARE_MATCH	0x7E	イベントリンク元を CMTW0・コンペアマッチに指定
ELC_RESOURCE_GPT0_COMPARE_MATCH_A	0x80	イベントリンク元を GPT0・コンペアマッチ A に指定
ELC_RESOURCE_GPT0_COMPARE_MATCH_B	0x81	イベントリンク元を GPT0・コンペアマッチ B に指定
ELC_RESOURCE_GPT0_COMPARE_MATCH_C	0x82	イベントリンク元を GPT0・コンペアマッチ C に指定
ELC_RESOURCE_GPT0_COMPARE_MATCH_D	0x83	イベントリンク元を GPT0・コンペアマッチ D に指定
ELC_RESOURCE_GPT0_OVERFLOW	0x86	イベントリンク元を GPT0・オーバーフローに指定
ELC_RESOURCE_GPT0_UNDERFLOW	0x87	イベントリンク元を GPT0・アンダーフローに指定
ELC_RESOURCE_GPT1_COMPARE_MATCH_A	0x88	イベントリンク元を GPT1・コンペアマッチ A に指定
ELC_RESOURCE_GPT1_COMPARE_MATCH_B	0x89	イベントリンク元を GPT1・コンペアマッチ B に指定
ELC_RESOURCE_GPT1_COMPARE_MATCH_C	0x8A	イベントリンク元を GPT1・コンペアマッチ C に指定
ELC_RESOURCE_GPT1_COMPARE_MATCH_D	0x8B	イベントリンク元を GPT1・コンペアマッチ D に指定
ELC_RESOURCE_GPT1_OVERFLOW	0x8E	イベントリンク元を GPT1・オーバーフローに指定
ELC_RESOURCE_GPT1_UNDERFLOW	0x8F	イベントリンク元を GPT1・アンダーフローに指定
ELC_RESOURCE_GPT2_COMPARE_MATCH_A	0x90	イベントリンク元を GPT2・コンペアマッチ A に指定
ELC_RESOURCE_GPT2_COMPARE_MATCH_B	0x91	イベントリンク元を GPT2・コンペアマッチ B に指定
ELC_RESOURCE_GPT2_COMPARE_MATCH_C	0x92	イベントリンク元を GPT2・コンペアマッチ C に指定
ELC_RESOURCE_GPT2_COMPARE_MATCH_D	0x93	イベントリンク元を GPT2・コンペアマッチ D に指定
ELC_RESOURCE_GPT2_OVERFLOW	0x96	イベントリンク元を GPT2・オーバーフローに指定
ELC_RESOURCE_GPT2_UNDERFLOW	0x97	イベントリンク元を GPT2・アンダーフローに指定
ELC_RESOURCE_GPT3_COMPARE_MATCH_A	0x98	イベントリンク元を GPT3・コンペアマッチ A に指定
ELC_RESOURCE_GPT3_COMPARE_MATCH_B	0x99	イベントリンク元を GPT3・コンペアマッチ B に指定
ELC_RESOURCE_GPT3_COMPARE_MATCH_C	0x9A	イベントリンク元を GPT3・コンペアマッチ C に指定
ELC_RESOURCE_GPT3_COMPARE_MATCH_D	0x9B	イベントリンク元を GPT3・コンペアマッチ D に指定
ELC_RESOURCE_GPT3_OVERFLOW	0x9E	イベントリンク元を GPT3・オーバーフローに指定
ELC_RESOURCE_GPT3_UNDERFLOW	0x9F	イベントリンク元を GPT3・アンダーフローに指定
ELC_RESOURCE_MTU6_COMPARE_MATCH_6A	0xA0	イベントリンク元を MTU6・コンペアマッチ 6A に指定
ELC_RESOURCE_MTU6_COMPARE_MATCH_6B	0xA1	イベントリンク元を MTU6・コンペアマッチ 6B に指定
ELC_RESOURCE_MTU6_COMPARE_MATCH_6C	0xA2	イベントリンク元を MTU6・コンペアマッチ 6C に指定

定数名	値	内容
ELC_RESOURCE_MTU6_COMPARE_MATCH_6D	0xA3	イベントリンク元を MTU6・コンペアマッチ 6D に指定
ELC_RESOURCE_MTU6_OVERFLOW	0xA4	イベントリンク元を MTU6・オーバーフローに指定
ELC_RESOURCE_MTU7_COMPARE_MATCH_7A	0xA5	イベントリンク元を MTU7・コンペアマッチ 7A に指定
ELC_RESOURCE_MTU7_COMPARE_MATCH_7B	0xA6	イベントリンク元を MTU7・コンペアマッチ 7B に指定
ELC_RESOURCE_MTU7_COMPARE_MATCH_7C	0xA7	イベントリンク元を MTU7・コンペアマッチ 7C に指定
ELC_RESOURCE_MTU7_COMPARE_MATCH_7D	0xA8	イベントリンク元を MTU7・コンペアマッチ 7D に指定
ELC_RESOURCE_MTU7_OVERFLOW	0xA9	イベントリンク元を MTU7・オーバーフローに指定
ELC_RESOURCE_MTU7_UNDERFLOW	0xAA	イベントリンク元を MTU7・アンダーフローに指定
ELC_RESOURCE_TPU0_COMPARE_MATCH_A	0xAC	イベントリンク元を TPU0・コンペアマッチ A に指定
ELC_RESOURCE_TPU0_COMPARE_MATCH_B	0xAD	イベントリンク元を TPU0・コンペアマッチ B に指定
ELC_RESOURCE_TPU0_COMPARE_MATCH_C	0xAE	イベントリンク元を TPU0・コンペアマッチ C に指定
ELC_RESOURCE_TPU0_COMPARE_MATCH_D	0xAF	イベントリンク元を TPU0・コンペアマッチ D に指定
ELC_RESOURCE_TPU0_OVERFLOW	0xB0	イベントリンク元を TPU0・オーバーフローに指定
ELC_RESOURCE_TPU1_COMPARE_MATCH_A	0xB1	イベントリンク元を TPU1・コンペアマッチ A に指定
ELC_RESOURCE_TPU1_COMPARE_MATCH_B	0xB2	イベントリンク元を TPU1・コンペアマッチ B に指定
ELC_RESOURCE_TPU1_OVERFLOW	0xB3	イベントリンク元を TPU1・オーバーフローに指定
ELC_RESOURCE_TPU1_UNDERFLOW	0xB4	イベントリンク元を TPU1・アンダーフローに指定
ELC_RESOURCE_TPU2_COMPARE_MATCH_A	0xB5	イベントリンク元を TPU2・コンペアマッチ A に指定
ELC_RESOURCE_TPU2_COMPARE_MATCH_B	0xB6	イベントリンク元を TPU2・コンペアマッチ B に指定
ELC_RESOURCE_TPU2_OVERFLOW	0xB7	イベントリンク元を TPU2・オーバーフローに指定
ELC_RESOURCE_TPU2_UNDERFLOW	0xB8	イベントリンク元を TPU2・アンダーフローに指定
ELC_RESOURCE_TPU3_COMPARE_MATCH_A	0xB9	イベントリンク元を TPU3・コンペアマッチ A に指定
ELC_RESOURCE_TPU3_COMPARE_MATCH_B	0xBA	イベントリンク元を TPU3・コンペアマッチ B に指定
ELC_RESOURCE_TPU3_COMPARE_MATCH_C	0xBB	イベントリンク元を TPU3・コンペアマッチ C に指定
ELC_RESOURCE_TPU3_COMPARE_MATCH_D	0xBC	イベントリンク元を TPU3・コンペアマッチ D に指定
ELC_RESOURCE_TPU3_OVERFLOW	0xBD	イベントリンク元を TPU3・オーバーフローに指定

表 5-24 MTU チャネル番号定義値（ELC）

定数名	値	内容
ELC_MTU_CH_0	0	MTU の ch0 を示す
ELC_MTU_CH_3	1	MTU の ch3 を示す
ELC_MTU_CH_4	2	MTU の ch4 を示す
ELC_MTU_CH_NUM	3	正常設定値範囲確認用

表 5-25 MTU イベントリンク動作設定定義値（ELC）

定数名	値	内容
ELC_MTU_COUNT_START	0	イベントリンク時、MTU のカウント開始
ELC_MTU_COUNT_RESTART	1	イベントリンク時、MTU のカウントをリスタート
ELC_MTU_INPUT_CAPTURE	2	イベントリンク時、MTU のインプットキャプチャを実行
ELC_MTU_MAX_OVER	3	正常設定値範囲確認用

表 5-26 CMT イベントリンク動作設定定義値（ELC）

定数名	値	内容
ELC_CMT_COUNT_START	0	イベントリンク時、CMT のカウント開始
ELC_CMT_COUNT_RESTART	1	イベントリンク時、CMT のカウントをリスタート
ELC_CMT_COUNT_INCREMENT	2	イベントリンク時、CMT のカウンタインクリメントを実行
ELC_CMT_MAX_OVER	3	正常設定値範囲確認用

表 5-27 ΔΣユニットチャネル・トリガ番号定義値（ELC）

定数名	値	内容
ELC_DSMIF0_TRIGGER_0	0	ΔΣユニット 0 のトリガ 0 を示す
ELC_DSMIF0_TRIGGER_1	1	ΔΣユニット 0 のトリガ 1 を示す
ELC_DSMIF1_TRIGGER_0	2	ΔΣユニット 1 のトリガ 0 を示す
ELC_DSMIF1_TRIGGER_1	3	ΔΣユニット 1 のトリガ 1 を示す
ELC_DSMIF_TRIGGER_NUM	4	正常設定値範囲確認用

表 5-28 12 ビット A/D コンバータチャネル番号定義値（ELC）

定数名	値	内容
ELC_S12AD_CH_0	0	12 ビット A/D コンバータの ch0 を示す
ELC_S12AD_CH_1	1	12 ビット A/D コンバータの ch1 を示す
ELC_S12AD_CH_NUM	2	正常設定値範囲確認用

表 5-29 割り込み番号定義値（ELC）

定数名	値	内容
ELC_INTR_NUM_1	0	割り込み要求信号 1 を示す
ELC_INTR_NUM_2	1	割り込み要求信号 2 を示す
ELC_INTR_NUM	2	正常設定値範囲確認用

表 5-30 ポートグループ番号定義値（ELC）

定数名	値	内容
ELC_PORT_GROUP_NUM_0	0	出力ポートグループ番号 0 を示す
ELC_PORT_GROUP_NUM_1	1	出力ポートグループ番号 1 を示す
ELC_PORT_GROUP_NUM	2	正常設定値範囲確認用

表 5-31 出力ポートグループイベントリンク動作設定定義値（ELC）

定数名	値	内容
ELC_OUT_GROUP_OUTPUT_0	0	イベントリンク時、出力グループポートは 0 を出力
ELC_OUT_GROUP_OUTPUT_1	1	イベントリンク時、出力グループポートは 1 を出力
ELC_OUT_GROUP_TOGGLE	2	イベントリンク時、出力ポートグループはトグル値を出力
ELC_OUT_GROUP_BUFFER	3	イベントリンク時、バッファ値を出力
ELC_OUT_GROUP_ROTATE	4	イベントリンク時、バッファ値をローテートした値を出力
ELC_OUT_GROUP_MAX_OVER	5	正常設定値範囲確認用

表 5-32 シングルポート番号定義値（ELC）

定数名	値	内容
ELC_SINGLE_PORT_NUM_0	0	シングルポート番号 0 を示す
ELC_SINGLE_PORT_NUM_1	1	シングルポート番号 1 を示す
ELC_SINGLE_PORT_NUM_2	2	シングルポート番号 2 を示す
ELC_SINGLE_PORT_NUM_3	3	シングルポート番号 3 を示す
ELC_SINGLE_PORT_NUM	4	正常設定値範囲確認用

表 5-33 シングルポートイベントリンク動作設定定義値（ELC）

定数名	値	内容
ELC_SINGLE_PORT_OUTPUT_0	0	イベントリンク時、0 を出力
ELC_SINGLE_PORT_OUTPUT_1	1	イベントリンク時、1 を出力
ELC_SINGLE_PORT_TOGGLE	2	イベントリンク時、トグル値を出力
ELC_SINGLE_PORT_ACTION_MAX_OVER	3	正常設定値範囲確認用

表 5-34 CMTW イベントリンク動作設定定義値（ELC）

定数名	値	内容
ELC_CMTW_COUNT_START	0	イベントリンク時、CMTW のカウント開始
ELC_CMTW_COUNT_RESTART	1	イベントリンク時、CMTW のカウントをリスタート
ELC_CMTW_COUNT_INCREMENT	2	イベントリンク時、CMTW のカウンタインクリメントを実行
ELC_CMTW_COUNT_MAX_OVER	3	正常設定値範囲確認用

表 5-35 TPU チャネル番号定義値（ELC）

定数名	値	内容
ELC_TPU_CH_0	0	TPU の ch0 を示す
ELC_TPU_CH_1	1	TPU の ch1 を示す
ELC_TPU_CH_2	2	TPU の ch2 を示す
ELC_TPU_CH_3	3	TPU の ch3 を示す
ELC_TPU_CH_NUM	4	正常設定値範囲確認用

表 5-36 TPU イベントリンク動作設定定義値（ELC）

定数名	値	内容
ELC_TPU_COUNT_START	0	イベントリンク時、TPU のカウント開始
ELC_TPU_COUNT_RESTART	1	イベントリンク時、TPU のカウントをリスタート
ELC_TPU_INPUT_CAPTURE	2	イベントリンク時、TPU のインプットキャプチャを実行
ELC_TPU_MAX_OVER	3	正常設定値範囲確認用

表 5-37 GPT チャネル番号定義値（ELC）

定数名	値	内容
ELC_GPT_CH_0	0	GPT の ch0 を示す
ELC_GPT_CH_1	1	GPT の ch1 を示す
ELC_GPT_CH_2	2	GPT の ch2 を示す
ELC_GPT_CH_3	3	GPT の ch3 を示す
ELC_GPT_CH_NUM	4	正常設定値範囲確認用

表 5-38 GPT イベントリンク動作設定定義値（ELC）

定数名	値	内容
ELC_GPT_COUNT_START	0	イベントリンク時、GPT のカウント開始
ELC_GPT_COUNT_RESTART	1	イベントリンク時、GPT のカウントをリスタート
ELC_GPT_COUNT_STOP	2	イベントリンク時、GPT のカウントをストップ
ELC_GPT_INPUT_CAPTURE	3	イベントリンク時、GPT のインプットキャプチャを実行
ELC_GPT_MAX_OVER	4	正常設定値範囲確認用

表 5-39 I/O ポートグループシンボル設定定義値（ELC）

定数名	値	内容
ELC_PORT_B	1	PORTB を指定
ELC_PORT_E	2	PORTE を指定
ELC_PORT_NUM	3	正常設定値範囲確認用

表 5-40 入力ポートグループイベント検出トリガ設定定義値（ELC）

定数名	値	内容
ELC_PORT_GROUP_TRIGGER_UP	0	入力ポートグループトリガを立ち上がりエッジに設定
ELC_PORT_GROUP_TRIGGER_DOWN	1	入力ポートグループトリガを立ち下がりエッジに設定
ELC_PORT_GROUP_TRIGGER_UP_DOWN	2	入力ポートグループトリガを立ち上がり/立下りエッジに設定
ELC_PORT_GROUP_MAX_OVER	3	正常設定値範囲確認用

表 5-41 シングルポートイベント検出トリガ設定定義値（ELC）

定数名	値	内容
ELC_SINGLE_EVENT_INPUT	0	イベント入力を選択設定
ELC_SINGLE_EVENT_OUTPUT	1	イベント出力設定
ELC_SINGLE_EVENT_MAX_OVER	2	正常設定値範囲確認用

表 5-42 シングルポートイベント検出トリガ設定定義値（ELC）

定数名	値	内容
ELC_SINGLE_PORT_TRIGGER_UP	0	入力シングルポートトリガを立ち上がりエッジに設定
ELC_SINGLE_PORT_TRIGGER_DOWN	1	入力シングルポートトリガを立ち下がりエッジに設定
ELC_SINGLE_PORT_TRIGGER_UP_DOWN	2	入力シングルポートトリガを立ち上がり/立下りエッジに設定
ELC_SINGLE_PORT_TRIGGER_MAX_OVER	3	正常設定値範囲確認用

5.8 構造体/共用体/列挙型

構造体/共用体/列挙型を記します。

5.8.1 ADC サンプルドライバの構造体/共用体/列挙型

以下に、ADC サンプルドライバの構造体/共用体/列挙型を示します。

```
/* ADC_OPEN() ARGUMENT DEFINITIONS */
typedef enum e_adc_mode
{
    ADC_MODE_SS_TEMPERATURE,                /* single scan temperature sensor */
    ADC_MODE_SS_ONE_CH,                     /* single scan one channel */
    ADC_MODE_SS_ONE_CH_DBLTRIG,             /* on even triggers save to ADDBLDR & interrupt */
    ADC_MODE_SS_MULTI_CH,                   /* 1 trigger source, scan multiple channels */
    ADC_MODE_SS_MULTI_CH_GROUPED,           /* 2 trigger sources, scan multiple channels */
    ADC_MODE_SS_MULTI_CH_GROUPED_DBLTRIG_A,
    ADC_MODE_CONT_ONE_CH,                   /* continuous scan one channel */
    ADC_MODE_CONT_MULTI_CH,                 /* continuous scan multiple channels */
    ADC_MODE_MAX
} adc_mode_t;

/* trigger sources */
typedef enum e_adc_trig
{
    ADC_TRIG_ADTRG0                 = 0,
    ADC_TRIG_TRGAON                 = 1,
    ADC_TRIG_TRGA1N                 = 2,
    ADC_TRIG_TRGA2N                 = 3,
    ADC_TRIG_TRGA3N                 = 4,
    ADC_TRIG_TRGA4N                 = 5,
    ADC_TRIG_TRGA6N                 = 6,
    ADC_TRIG_TRGA7N                 = 7,
    ADC_TRIG_TRGON                  = 8,
    ADC_TRIG_TRG4AN                 = 9,
    ADC_TRIG_TRG4BN                 = 10,
    ADC_TRIG_TRG4AN_OR_TRG4BN       = 11,
    ADC_TRIG_TRG4ABN                = 12,
    ADC_TRIG_TRG7AN                 = 13,
    ADC_TRIG_TRG7BN                 = 14,
    ADC_TRIG_TRG7AN_OR_TRG7BN       = 15,
    ADC_TRIG_TRG7ABN                = 16,
    ADC_TRIG_GTADTRAON              = 17,
    ADC_TRIG_GTADTRBON              = 18,
    ADC_TRIG_GTADTRA1N              = 19,
    ADC_TRIG_GTADTRB1N              = 20,
    ADC_TRIG_GTADTRA2N              = 21,
    ADC_TRIG_GTADTRB2N              = 22,
```

```
    ADC_TRIG_GTADTRA3N              = 23,
    ADC_TRIG_GTADTRB3N              = 24,
    ADC_TRIG_GTADTRAON_OR_GTADTRBON = 25,
    ADC_TRIG_GTADTRA1N_OR_GTADTRB1N = 26,
    ADC_TRIG_GTADTRA2N_OR_GTADTRB2N = 27,
    ADC_TRIG_GTADTRA3N_OR_GTADTRB3N = 28,
    ADC_TRIG_TPTRGAN_0              = 31,
    ADC_TRIG_TPTRG0AN_0             = 32,
    ADC_TRIG_TPTRGAN_1              = 33,
    ADC_TRIG_TPTRG6AN_1             = 34,
    ADC_TRIG_ELCTRG0                = 48,
    ADC_TRIG_SOFTWARE               = 63
} adc_trig_t;

typedef enum e_adc_add
{
    ADC_ADD_OFF           = 0,      /* addition is turned off for chans/sensors */
    ADC_ADD_TWO_SAMPLES,
    ADC_ADD_THREE_SAMPLES,
    ADC_ADD_FOUR_SAMPLES,
    ADC_ADD_MAX
} adc_add_t;

typedef enum e_adc_align
{
    ADC_ALIGN_RIGHT = 0x0000,
    ADC_ALIGN_LEFT  = 0x8000
} adc_align_t;

typedef enum e_adc_clear
{
    ADC_CLEAR_AFTER_READ_OFF = 0x0000,
    ADC_CLEAR_AFTER_READ_ON  = 0x0020
} adc_clear_t;

typedef struct st_adc_cfg
{
    adc_add_t       add_cnt;
    adc_align_t     alignment;          /* ignored if addition used */
    adc_clear_t     clearing;
    adc_trig_t      trigger;            /* default and Group A trigger source */
    adc_trig_t      trigger_groupb;     /* valid only for group modes */
    uint8_t         priority;           /* S12ADI0 interrupt priority; 0-15 */
    uint8_t         priority_groupb;    /* GBADI interrupt priority; 0-15 */
} adc_cfg_t;

/* CALLBACK FUNCTION ARGUMENT DEFINITIONS */
```

```
/* callback function events */
typedef enum e_adc_cb_evt
{
    ADC_EVT_SCAN_COMPLETE,              /* normal/Group A scan complete */
    ADC_EVT_SCAN_COMPLETE_GROUPB        /* Group B scan complete */
} adc_cb_evt_t;

typedef struct st_adc_cb_args           /* callback arguments */
{
    adc_cb_evt_t    event;
} adc_cb_args_t;

/* ADC_CONTROL() ARGUMENT DEFINITIONS */
typedef enum e_adc_cmd
{
    ADC_CMD_ENABLE_CHANS            = 1, /* enables chans and INT(s) if priority != 0 */
    ADC_CMD_ENABLE_TEMP_SENSOR,         /* enables sensor and INT if priority != 0 */
    ADC_CMD_SET_SAMPLE_STATE_CNT,
    ADC_CMD_ENABLE_TRIG,                /* allows an async/sync trigger to start scan */
    ADC_CMD_DISABLE_TRIG,               /* prevents an async/sync trigger to start scan */
    ADC_CMD_SCAN_NOW,                   /* issue software trigger */
    ADC_CMD_DISABLE_INT,                /* interrupt disable; ADCSR.ADIE=0 */
    ADC_CMD_ENABLE_INT,                 /* interrupt enable;  ADCSR.ADIE=1 */
    ADC_CMD_DISABLE_INT_GROUPB,         /* interrupt disable; ADCSR.GBADIE=0 */
    ADC_CMD_ENABLE_INT_GROUPB,          /* interrupt enable;  ADCSR.GBADIE=1 */
    ADC_CMD_CHECK_SCAN_DONE,            /* for Normal, GroupA or GroupB scan */
    ADC_CMD_CHECK_SCAN_DONE_GROUPA,
    ADC_CMD_CHECK_SCAN_DONE_GROUPB,
    ADC_CMD_MAX
} adc_cmd_t;

/* for ADC_CMD_ENABLE_CHANS */
typedef struct st_adc_ch_cfg            /* bit 0 is ch0; bit 7 is ch7 */
{
    uint32_t        chan_mask;          /* channels/bits 0-7 */
    uint32_t        chan_mask_groupb;   /* valid for group modes */
    uint32_t        add_mask;           /* valid if add enabled in Open() */
} adc_ch_cfg_t;




/* for ADC_CMD_SET_SAMPLE_STATE_CNT */
/* sample state registers */
typedef enum e_adc_sst_reg
```

```
{
    ADC_SST_CH0 = 0,
    ADC_SST_CH1,
    ADC_SST_CH2,
    ADC_SST_CH3,
    ADC_SST_CH4,
    ADC_SST_CH5,
    ADC_SST_CH6,
    ADC_SST_CH7,
    ADC_SST_TEMPERATURE,
    ADC_SST_NUM_REGS
} adc_sst_reg_t;

typedef struct st_adc_time
{
    adc_sst_reg_t   reg_id;
    uint8_t         num_states;         /* default=11 */
} adc_time_t;

/* ADC_READ() ARGUMENT DEFINITIONS */
typedef enum e_adc_reg
{
    ADC_REG_CH0 = 0,
    ADC_REG_CH1 = 1,
    ADC_REG_CH2 = 2,
    ADC_REG_CH3 = 3,
    ADC_REG_CH4 = 4,
    ADC_REG_CH5 = 5,
    ADC_REG_CH6 = 6,
    ADC_REG_CH7 = 7,
    ADC_REG_TEMP = 8,
    ADC_REG_DBLTRIG = 9,
    ADC_REG_MAX
} adc_reg_t;
/* ADC_READALL() ARGUMENT DEFINITIONS */
typedef struct st_adc_data
{
    uint16_t    chan[8];
    uint16_t    dbltrig;
} adc_data_t;
```

図 5.2　ADC サンプルドライバの構造体/共用体/列挙型

5.8.2 CMT(W)サンプルドライバの構造体/共用体/列挙型

以下に、CMT(W)サンプルドライバの構造体/共用体/列挙型を示します。

```
typedef enum e_cmt_channel
{
    CMT_CHANNEL_0,
    CMT_CHANNEL_1,
    CMT_CHANNEL_2,
    CMT_CHANNEL_3,
    CMT_CHANNEL_4,
    CMT_CHANNEL_5,
    CMT_CHANNEL_6,          /* CMTW */
    CMT_CHANNEL_7,          /* CMTW */
    CMT_CHANNEL_MAX
} cmt_channel_t;

/*-----------------------------------------------------------------------------*/
/*   Define cmt(w) timer count setting structure.                              */
/*-----------------------------------------------------------------------------*/
typedef struct
{
    uint32_t                pclk_div;
    uint32_t                cnt_size;
    uint32_t                clear_factor;
} cmt_time_cnt_t;
#define cmtw_time_cnt_t     cmt_time_cnt_t

/*-----------------------------------------------------------------------------*/
/*   Define cmt(w) compare match setting structure.                            */
/*-----------------------------------------------------------------------------*/
typedef struct
{
    int32_t                 mode_enable;
    uint32_t                compare_match_cnt;
    int32_t                 intr_priority;
    void                    (* p_callback)(void);
} cmt_compare_match_t;
#define cmtw_compare_match_t    cmt_compare_match_t
```

```
/*-----------------------------------------------------------------------------*/
/*   Define cmt(w) output compare setting structure.                           */
/*-----------------------------------------------------------------------------*/
typedef struct
{
    int32_t                 mode_enable;
    uint32_t                output_compare_cnt;
    uint32_t                output_signal;
    int32_t                 intr_priority;
    void                    (* p_callback)(void);
} cmtw_output_compare_t;

/*-----------------------------------------------------------------------------*/
/*   Define cmt(w) input capture setting structure.                            */
/*-----------------------------------------------------------------------------*/
typedef struct
{
    int32_t                 mode_enable;
    uint32_t                trigger;
    int32_t                 filter_enable;
    int32_t                 intr_priority;
    void                    (* p_callback)(uint32_t cnt_value);
} cmtw_input_capture_t;

/*-----------------------------------------------------------------------------*/
/*   Define ECM setting structure.                                             */
/*-----------------------------------------------------------------------------*/
typedef struct
{
    int32_t                 ecm_enable;
    uint32_t                output_compare_num;
} cmtw_ecm_t;

/*-----------------------------------------------------------------------------*/
/*   Define cmt(w) operation mode setting structure.                           */
/*-----------------------------------------------------------------------------*/
typedef struct
{
    cmt_compare_match_t     compare_match;
    cmtw_output_compare_t   output_compare[CMTW_OUTPUT_COMPARE_NUM];
    cmtw_input_capture_t    input_capture[CMTW_INPUT_CAPTURE_NUM];
    uint32_t                noise_filter_clk;
} cmt_mode_t;




/*-----------------------------------------------------------------------------*/
```

```
/*   Define cmt(w) configuration structure.                                    */
/*--------------------------------------------------------------------------*/
typedef struct
{
    cmt_time_cnt_t          time_cnt_param;
    cmt_mode_t              mode_param;
} cmt_cfg_t;
```

図 5.3　CMT(W)サンプルドライバの構造体/共用体/列挙型

5.8.3 ELC サンプルドライバの構造体/共用体/列挙型

以下に、ELC サンプルドライバの構造体/共用体/列挙型を示します。

```
/*-----------------------------------------------------------------------------*/
/*   Define MTU parameter for event link.                                      */
/*-----------------------------------------------------------------------------*/
typedef struct
{
    uint32_t        elc_mtu_ch;
    int32_t         event_link_enable;
    uint32_t        resource;
    uint32_t        action;
}elc_cmd_mtu_t;

/*-----------------------------------------------------------------------------*/
/*   Define CMT parameter for event link.                                      */
/*-----------------------------------------------------------------------------*/
typedef struct
{
    int32_t         event_link_enable;
    uint32_t        resource;
    uint32_t        action;
}elc_cmd_cmt_t;

/*-----------------------------------------------------------------------------*/
/*   Define delta sigma unit parameter for event link.                         */
/*-----------------------------------------------------------------------------*/
typedef struct
{
    uint32_t        elc_dsmif_ch;
    int32_t         event_link_enable;
    uint32_t        resource;
}elc_cmd_dsmif_t;

/*-----------------------------------------------------------------------------*/
/*   Define 12bit A/D converter parameter for event link.                      */
/*-----------------------------------------------------------------------------*/
typedef struct
{
    uint32_t        elc_s12ad_ch;
    int32_t         event_link_enable;
    uint32_t        resource;
}elc_cmd_s12ad_t;
```

```
/*-----------------------------------------------------------------------------*/
/*   Define ELC interrupt for event link.                                      */
/*-----------------------------------------------------------------------------*/
typedef struct
{
    uint32_t        elc_intr_num;
    int32_t         event_link_enable;
    uint32_t        resource;
    int32_t         intr_priority;
    void            (* p_callback)(void);
}elc_cmd_intr_t;

/*-----------------------------------------------------------------------------*/
/*   Define output port group parameter for event link.                        */
/*-----------------------------------------------------------------------------*/
typedef struct
{
    uint32_t        elc_out_port_group_num;
    int32_t         event_link_enable;
    uint32_t        resource;
    uint32_t        action;
    uint8_t         init_value;
}elc_cmd_out_port_group_t;

/*-----------------------------------------------------------------------------*/
/*   Define input port group parameter for event link.                         */
/*-----------------------------------------------------------------------------*/
typedef struct
{
    uint32_t        elc_in_port_group_num;
    int32_t         event_link_enable;
    uint8_t         resource;
    int32_t         overwrite_enable;
}elc_cmd_in_port_group_t;









/*-----------------------------------------------------------------------------*/
```

```
/*   Define input port group parameter for event link.                        */
/*---------------------------------------------------------------------------*/
typedef struct
{
    uint32_t        elc_single_port_num;
    uint32_t        port_symbol;
    uint32_t        port_num;
    int32_t         event_link_enable;
    uint32_t        event_direction;
    uint32_t        resource;
    uint32_t        output_action;
    uint32_t        input_trigger;
}elc_cmd_single_port_t;

/*---------------------------------------------------------------------------*/
/*   Define CMTW parameter for event link.                                   */
/*---------------------------------------------------------------------------*/
typedef struct
{
    int32_t         event_link_enable;
    uint32_t        resource;
    uint32_t        action;
}elc_cmd_cmtw_t;

/*---------------------------------------------------------------------------*/
/*   Define TPU parameter for event link.                                    */
/*---------------------------------------------------------------------------*/
typedef struct
{
    uint32_t        elc_tpu_ch;
    int32_t         event_link_enable;
    uint32_t        resource;
    uint32_t        action;
}elc_cmd_tpu_t;

/*---------------------------------------------------------------------------*/
/*   Define GPT parameter for event link.                                    */
/*---------------------------------------------------------------------------*/
typedef struct
{
    uint32_t        elc_gpt_ch;
    int32_t         event_link_enable;
    uint32_t        resource;
    uint32_t        action;
}elc_cmd_gpt_t;
/*---------------------------------------------------------------------------*/
/*   Define port group parameter for port group setting.                     */
```

```
/*-----------------------------------------------------------------------------*/
typedef struct
{
    uint32_t        port_group_num;
    uint8_t         port_group_bit;
    uint32_t        trigger;
}elc_cmd_port_group_t;

/*-----------------------------------------------------------------------------*/
/*   Define port group parameter for port group value get.                     */
/*-----------------------------------------------------------------------------*/
typedef struct
{
    uint32_t        elc_port_group_num;
    uint8_t         port_value;
}elc_get_port_value_t;
```

図 5.4　ELC サンプルドライバの構造体/共用体/列挙型

5.9 大域変数一覧

以下に大域変数一覧を示します。

表 5-43 大域変数一覧

型	変数名	内容	使用関数
static volatile int32_t	gb_cmtw_end_flag	CMT(W)サンプルドライバのコールバック情報	cmtw_elc_sample_cmtwi_callback
static volatile int32_t	gb_adc_end_flag	ADC サンプルドライバのコールバック情報	cmtw_elc_sample_adc_callback

5.10 関数一覧

以下に関数一覧を示します。

表 5-44 関数一覧

5.11 サンプルアプリケーションの関数仕様

サンプルアプリケーションコードの関数の仕様を記します。

5.11.1 main

main	
概　要	周期的なポテンショメータ入力電圧の A/D 変換を実行
ヘッダ	-
宣　言	int32_t main(void);
説　明	本関数は、以下の処理を行います。 以下ドライバの初期化を行います。 ・シリアル ・CMT(W) ・ELC ・ADC ELC により CMTW0 と ADC をリンクさせ、CMTW0 のコンペアマッチ周期で 評価ボード上で接続されているポテンショメータの入力電圧の A/D 変換結果を USB シリアルで出力します。
引　数	-
リターン値	本関数は無限ループによりリターンしません。
補　足	ADC ドライバは同期トリガ、シングルスキャンモードで起動します。

5.11.2 sample_setup

sample_setup	
概　要	CMT(W)、ELC、ADC ドライバの設定
ヘッダ	-
宣　言	static int32_t sample_setup(void);
説　明	本関数は、以下の処理を行います。 ADC ドライバは、ELC トリガ、シングルスキャンモードで起動します。 ELC ドライバは、CMTW0 コンペアマッチと ADC をイベントリンクし起動します。 CMT(W)ドライバは、CMTW0 を 3 秒周期のコンペアマッチで起動します。
引　数	-
リターン値	エラーコード ESUCCESS　：成功 EINVAL　：入力パラメータの不正 EACESS　：不正アクセス EFAULT　：失敗 EBUSY　：ビジー その他、エラーコード一覧を参照してください。
補　足	-

5.11.3 sample_process

sample_process	
概　要	AD 変換完了後、ADC 結果を USB シリアルに出力します。
ヘッダ	-
宣　言	static void sample_process(void);
説　明	CMTW0 のイベントを検出後、AD 変換の完了イベントを待機します。 AD 変換の完了イベント検出後、AD 結果を USB シリアル通信で AD 変換値を出力します。
引　数	-
リターン値	-
補　足	-

5.11.4 sample_release

sample_release	
概　要	CMT(W)、ELC、ADC ドライバの終了及び、解放
ヘッダ	-
宣　言	static void sample_release(void);
説　明	CMT(W)、ELC、ADC ドライバの終了し、各モジュールを停止、終了します。
引　数	-
リターン値	-
補　足	-

5.11.5 cmtw_ch0_inhr_callback

cmtw_ch0_inhr_callback	
概　要	CMTW0 の割り込みハンドラ
ヘッダ	-
宣　言	static void cmtw_ch0_inhr_callback(void);
説　明	R_CMT_Open で設定した CMTW0 のコンペアマッチ割り込みを検出するためのコールバック関数。関数呼び出し後、アプリケーションへ CMTW0 コンペアマッチイベントを通知します。
引　数	-
リターン値	-
補　足	-

5.11.6 adc_inhr_callback

adc_inhr_callback	
概　要	AD 変換完了の割り込みハンドラ
ヘッダ	-
宣　言	static void adc_inhr_callback(int32_t event);
説　明	R_ADC_Open で設定した AD 変換完了割り込みを検出するためのコールバック関数。関数呼び出し後、アプリケーションへ AD 変換完了を通知します。
引　数	-
リターン値	-
補　足	-

5.12 ADC サンプルドライバの関数仕様

ADC サンプルドライバの関数仕様を記します。

5.12.1 R_ADC_Init

R_ADC_Init	
概　要	ADC ドライバの初期化
ヘッダ	r_adc_if.h
宣　言	void R_ADC_Init(void);
説　明	ADC ドライバの初期化を行います。 ADC ドライバ I/F の排他制御用セマフォを生成します。
引　数	-
リターン値	-
補　足	-

5.12.2 R_ADC_Uninit

R_ADC_Uninit	
概　要	ADC ドライバの終了
ヘッダ	r_adc_if.h
宣　言	int32_t R_ADC_Uninit(void);
説　明	ADC ドライバを終了します。 ADC モジュールの停止及び、I/F 排他制御用セマフォを解放します。
引　数	-
リターン値	エラーコード：ESUCCESS、EACCES、EFAULT
補　足	-

5.12.3 R_ADC_Open

R_ADC_Open		
概　要	ADC の起動設定	
ヘッダ	r_adc_if.h	
宣　言	int32_t R_ADC_Open(adc_mode_t const mode, adc_cfg_t * const p_cfg, void (* const p_callback)(int32_t event));	
説　明	ADC の動作を開始するための設定を行います。	
引　数	mode	ADC 機能モード
	p_cfg	ADC 機能設定情報ポインタ
	p_callback	AD 変換完了イベントコールバック
リターン値	エラーコード：ESUCCESS、EBUSY、EACESS、EFAULT	
補　足	R_ADC_Init 関数を実行し、ADC ドライバを初期化してください。 ADC 機能の終了後、R_ADC_Close 関数を呼び出して終了してください。	

5.12.4 R_ADC_Close

R_ADC_Close	
概　要	ADC 機能の終了
ヘッダ	r_adc_if.h
宣　言	int32_t R_ADC_Close(void);
説　明	ADC 機能を停止し、終了します。
引　数	-
リターン値	エラーコード：ESUCCESS、EACESS、EFAULT
補　足	R_ADC_Open 関数を実行して ADC 機能を起動してください。

5.12.5 R_ADC_Control

R_ADC_Control		
概　要	ADC の機能設定	
ヘッダ	r_adc_if.h	
宣　言	int32_t R_ADC_Control(adc_cmd_t const cmd, void * const p_args);	
説　明	ADC の機能の設定を行います。 ADC コマンドの詳細については、「5.16 R_ADC_Control コマンド仕様」章を参照してください。	
引　数	cmd	ADC コントロールコマンド
	p_args	ADC コマンドパラメータ情報のポインタ
リターン値	エラーコード：ESUCCESS、EACESS、EFAULT	
補　足	R_ADC_Open 関数を実行して ADC 機能を起動してください。	

5.12.6 R_ADC_Read

R_ADC_Read		
概　要	1 チャンネルの A/D 変換値読み取り	
ヘッダ	r_adc_if.h	
宣　言	int32_t R_ADC_Read(adc_reg_t const reg_id, uint16_t * const p_data);	
説　明	指定した 1 チャンネルの A/D 変換値を読み取ります。	
引　数	reg_id	A/D 変換値の読み出しチャンネル
	p_data	A/D 変換値の格納先変数のポインタ
リターン値	エラーコード：ESUCCESS、EINVAL、EACESS、EFAULT	
補　足	R_ADC_Open 関数を実行して ADC 機能を起動してください。	

5.12.7 R_ADC_ReadAll

R_ADC_ReadAll		
概　要	全チャンネルの A/D 変換値読み取り	
ヘッダ	r_adc_if.h	
宣　言	int32_t R_ADC_ReadAll(adc_data_t * const p_all_data);	
説　明	全チャンネルの A/D 変換値を読み取ります。	
引　数	p_all_data	A/D 変換値の格納先変数のポインタ
リターン値	エラーコード：ESUCCESS、EINVAL、EACESS、EFAULT	
補　足	R_ADC_Open 関数を実行して ADC 機能を起動してください。	

5.12.8 R_ADC_GetVersion

R_ADC_GetVersion	
概　要	ADC ドライバのバージョン情報取得
ヘッダ	r_adc_if.h
宣　言	int32_t R_ADC_GetVersion(void);
説　明	ADC ドライバのバージョン情報を返します。
引　数	-
リターン値	ADC ドライバのバージョン情報 0-15bit　　: マイナーバージョン 16-31bit　　: メジャーバージョン
補　足	-

5.13 CMT(W)サンプルドライバの関数仕様

CMT(W)サンプルドライバの関数仕様を記します。

5.13.1 R_CMT_Init

R_CMT_Init	
概　要	CMT(W)ドライバの初期化
ヘッダ	r_cmt_if.h
宣　言	void R_CMT_Init(void);
説　明	CMT(W)ドライバの初期化を行います。 CMT(W)ドライバ I/F の排他制御用セマフォを生成します。
引　数	-
リターン値	-
補　足	-

5.13.2 R_CMT_Uninit

R_CMT_Uninit	
概　要	CMT(W)ドライバの終了
ヘッダ	r_cmt_if.h
宣　言	int32_t R_CMT_Uninit(void);
説　明	CMT(W)ドライバを終了します。 CMT(W)モジュールの停止及び、I/F 排他制御用セマフォを解放します。
引　数	-
リターン値	エラーコード：ESUCCESS、EINVAL（CMT_Close/CMTW_Close でエラー発生）、EFAULT
補　足	-

5.13.3 R_CMT_Open

R_CMT_Open	
概　要	CMT(W)の起動設定
ヘッダ	r_cmt_if.h
宣　言	int32_t R_CMT_Open(cmt_channel_t const channel, cmt_cfg_t * const p_cfg);
説　明	CMT(W)の動作を開始するための設定を行います。 指定したチャンネル番号の CMT 又は CMTW の起動設定を行います。引数の channel に以下を指定します。 コンペアマッチタイマ(CMT)は、CMT_CHANNEL_0～CMT_CHANNEL_5 コンペアマッチタイマ W(CMTW)は、CMT_CHANNEL_6、CMT_CHANNEL_7
引　数	channel　チャンネル番号 CMT_CHANNEL_0～CMT_CANNEL_7 p_cfg　CMT(W)機能設定情報ポインタ
リターン値	エラーコード：ESUCCESS、EINVAL、EACESS、EBUSY、EFAULT
補　足	R_CMT_Init 関数を実行し、CMT(W)ドライバを初期化してください。 CMT(W)機能の終了後、R_CMT_Close 関数を呼び出して終了してください。

5.13.4 R_CMT_Close

R_CMT_Close

概　要	CMT(W)機能の終了
ヘッダ	r_cmt_if.h
宣　言	int32_t　R_CMT_Close(cmt_channel_t const channel);
説　明	指定したチャンネルの CMT 又は CMTW 機能を停止し、終了します。 指定したチャンネル番号の CMT 又は CMTW の停止を行います。引数の channel に以下を指定します。 　コンペアマッチタイマ(CMT)は、CMT_CHANNEL_0～CMT_CHANNEL_5 　コンペアマッチタイマ W(CMTW)は、CMT_CHANNEL_6、CMT_CHANNEL_7
引　数	channel　チャンネル番号 CMT_CHANNEL_0～CMT_CANNEL_7
リターン値	エラーコード：ESUCCESS、EINVAL、EACESS、EBUSY、EFAULT
補　足	R_CMT_Open 関数を実行し、CMT(W)機能を起動してください。

5.13.5 R_CMT_Control

R_CMT_Control

概　要	CMT(W)機能の設定関数
ヘッダ	r_cmt_if.h
宣　言	int32_t R_CMT_Control(cmt_channel_t const channel, const uint32_t cmd, void * const p_data);
説　明	CMT(W)機能のための設定を行います。 指定されたコマンドに対応する設定を行います。 詳細については、「5.17 R_CMT_Control コマンド仕様」章を参照してください。
引　数	channel　チャンネル番号 CMT_CHANNEL_0～CMT_CANNEL_7 cmd　機能設定を行うためのコマンドを指定 p_data　機能設定を行うためのパラメータ情報を設定
リターン値	エラーコード：ESUCCESS、EACESS、、EBUSY、EFAULT
補　足	R_CMT_Open 関数を実行し、CMT(W)機能を起動してください。

5.13.6 R_CMT_StartPeriodic

R_CMT_StartPeriodic

概　要	タイマカウンタの周期動作開始関数
ヘッダ	r_cmt_if.h
宣　言	int32_t R_CMT_StartPeriodic(cmt_channel_t const channel);
説　明	タイマカウントの周期動作を開始します。
引　数	channel　チャンネル番号 CMT_CHANNEL_0～CMT_CANNEL_7
リターン値	エラーコード：ESUCCESS、EINVAL、EACESS、EBUSY、EFAULT
補　足	R_CMT_Open 関数を実行し、CMT(W)機能を起動してください。

5.13.7 R_CMT_StartOneShot

R_CMT_StartOneShot

概　要	タイマカウンタの非周期動作開始関数
ヘッダ	r_cmt_if.h
宣　言	int32_t R_CMT_StartOneShot(cmt_channel_t const channel);
説　明	タイマカウントの非周期動作を開始します。 タイマカウントクリア条件成立後、タイマは自動的に停止し、初期化状態へ遷移します。
引　数	channel　チャンネル番号 CMT_CHANNEL_0～CMT_CANNEL_7
リターン値	エラーコード：ESUCCESS、EINVAL、EACESS、EBUSY、EFAULT
補　足	R_CMT_Open 関数を実行し、CMT(W)機能を起動してください。

5.13.8 R_CMT_GetVersion

R_CMT_GetVersion

概　要	CMT(W)ドライバのバージョン情報取得
ヘッダ	r_cmt_if.h
宣　言	int32_t R_CMT_GetVersion(void);
説　明	CMT(W)ドライバのバージョン情報を返します。
引　数	-
リターン値	CMT(W)ドライバのバージョン情報 0-15bit　：マイナーバージョン 16-31bit　：メジャーバージョン
補　足	-

5.14 ELC サンプルドライバの関数仕様

ELC サンプルドライバの関数仕様を示します。

5.14.1 R_ELC_Init

R_ELC_Init	
概　要	ELC ドライバの初期化
ヘッダ	r_elc_if.h
宣　言	void R_ELC_Init(void);
説　明	ELC ドライバの初期化を行います。 ELC ドライバ I/F の排他制御用セマフォを生成します。
引　数	-
リターン値	-
補　足	-

5.14.2 R_ELC_Uninit

R_ELC_Uninit	
概　要	ELC ドライバの終了
ヘッダ	r_elc_if.h
宣　言	int32_t R_ELC_Uninit(void);
説　明	ELC ドライバを終了します。 ELC モジュールの停止及び、I/F 排他制御用セマフォを解放します。
引　数	-
リターン値	エラーコード：ESUCCESS、EFAULT
補　足	-

5.14.3 R_ELC_Open

R_ELC_Open	
概　要	ELC ドライバの初期化関数
ヘッダ	r_elc_if.h
宣　言	int32_t R_ELC_Open(void);
説　明	ELC の動作を開始するための設定を行います。
引　数	-
リターン値	エラーコード：ESUCCESS、EINVAL（ELC_Open 関数で ELC_ERR_MISSING_PTR または ELC_ERR_INVALID_ARG が発生）、EACESS、EBUSY、EFAULT
補　足	R_ELC_Init 関数を実行し、ELC ドライバを初期化してください。 ELC 機能の終了後、R_ELC_Close 関数を呼び出して終了してください。

5.14.4 R_ELC_Close

R_ELC_Close	
概　要	ELC 機能の終了
ヘッダ	r_elc_if.h
宣　言	int32_t R_ELC_Close(void);
説　明	ELC 機能を停止し、終了します。
引　数	-
リターン値	エラーコード：ESUCCESS、EBUSY、EACESS、、EFAULT
補　足	R_ELC_Open 関数を実行して ELC 機能を起動してください。

5.14.5 R_ELC_Control

R_ELC_Control	
概　要	ELC ドライバの機能設定関数
ヘッダ	r_elc_if.h
宣　言	int32_t　R_ELC_Control(uint32_t const cmd, void * const p_data);
説　明	ELC 機能設定のための処理を行います。 指定されたコマンドに対応する設定を行います。 詳細については、「5.18 R_ELC_Control コマンド仕様」章を参照してください。
引　数	cmd　ELC コントロールコマンド p_data　ELC コマンドパラメータ情報のポインタ
リターン値	エラーコード：ESUCCESS、EINVAL、EACESS、EBUSY、EFAULT
補　足	・R_ELC_Open()の実行後に本関数を実行してください。

5.14.6 R_ELC_LinkStart

R_ELC_LinkStart	
概　要	ELC のイベントリンク開始
ヘッダ	r_elc_if.h
宣　言	int32_t R_ELC_LinkStart(void);
説　明	ELC モジュールによるイベントリンク機能を開始します。
引　数	-
リターン値	エラーコード：ESUCCESS、EINVAL（ELC_LinkStart 関数で ELC_ERR_MISSING_PTR または、ELC_ERR_INVALID_ARG が発生）、EACESS、EBUSY、EFAULT
補　足	R_ELC_Open 関数を実行して ELC 機能を起動してください。

5.14.7 R_ELC_LinkStop

R_ELC_LinkStop	
概　要	ELC のイベントリンク停止
ヘッダ	r_elc_if.h
宣　言	int32_t R_ELC_LinkStop(void);
説　明	ELC モジュールによるイベントリンク機能を停止します。
引　数	-
リターン値	エラーコード：ESUCCESS、EINVAL（ELC_LinkStop 関数で ELC_ERR_MISSING_PTR または、ELC_ERR_INVALID_ARG が発生）、EACESS、EFAULT
補　足	R_ELC_LinkStart 関数を実行してイベントリンク機能を開始してください。

5.14.8 R_ELC_GetVersion

R_ELC_GetVersion	
概　要	ELC ドライバのバージョン情報取得
ヘッダ	r_elc_if.h
宣　言	int32_t R_ELC_GetVersion(void);
説　明	ELC ドライバのバージョン情報を返します。
引　数	-
リターン値	ELC ドライバのバージョン情報 0-15bit　：マイナーバージョン 16-31bit　：メジャーバージョン
補　足	-

5.15 フローチャート

5.15.1 サンプルプログラムメイン処理

図 5.5~図 5.7 にサンプルプログラムのメイン処理のフローチャートを示します。

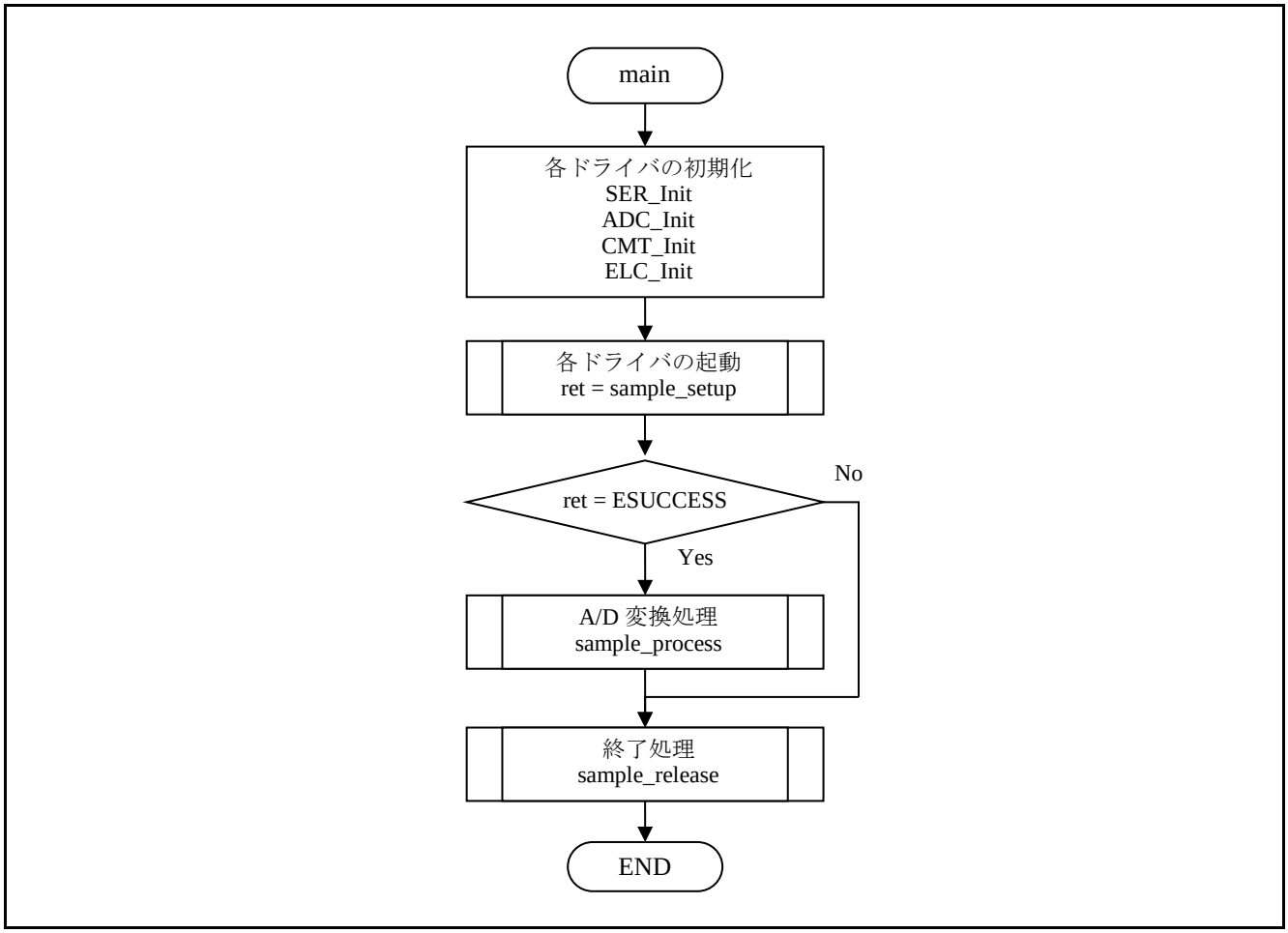

図 5.5 メイン処理

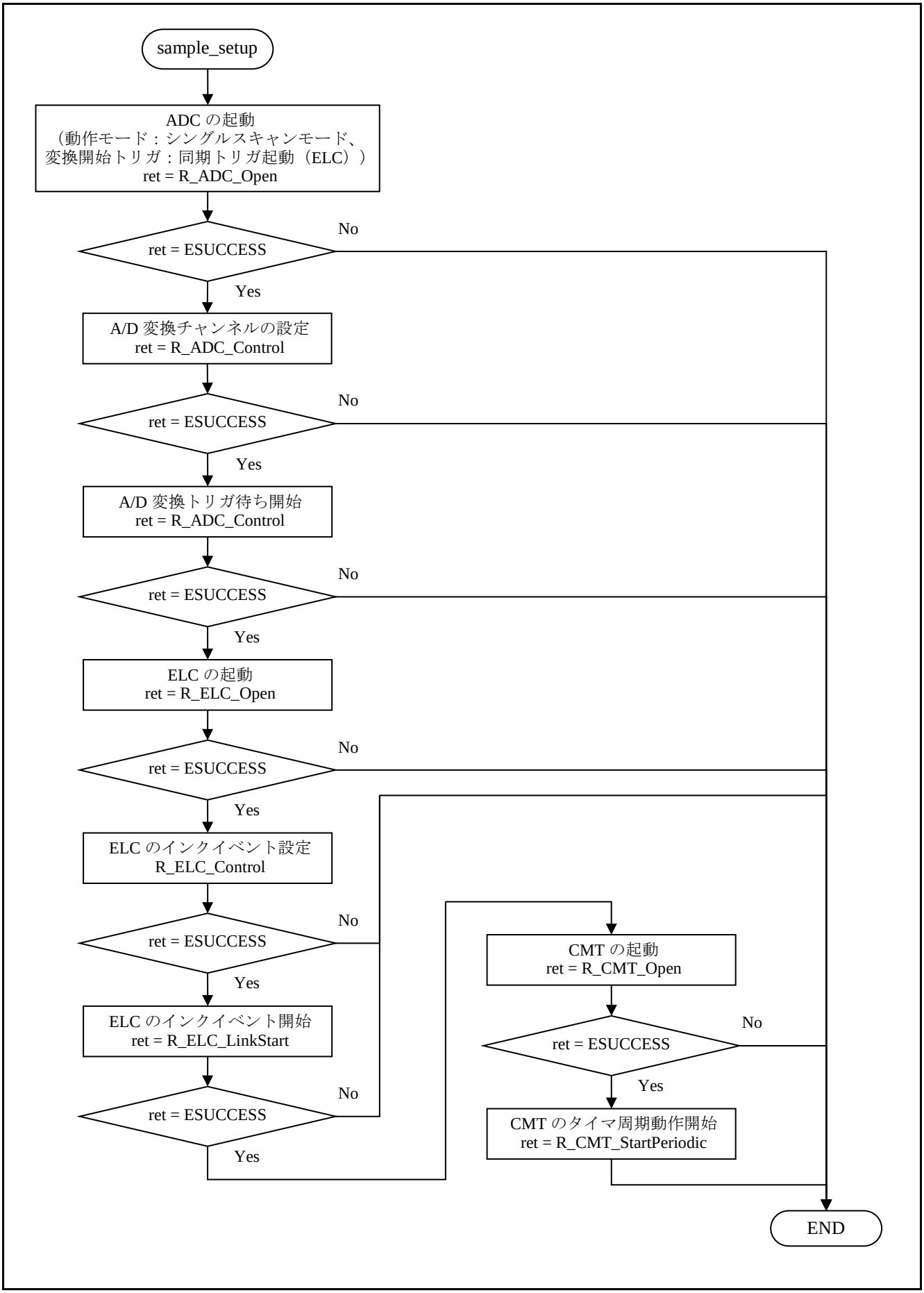

図 5.6　CMTW、ELC、ADC の起動

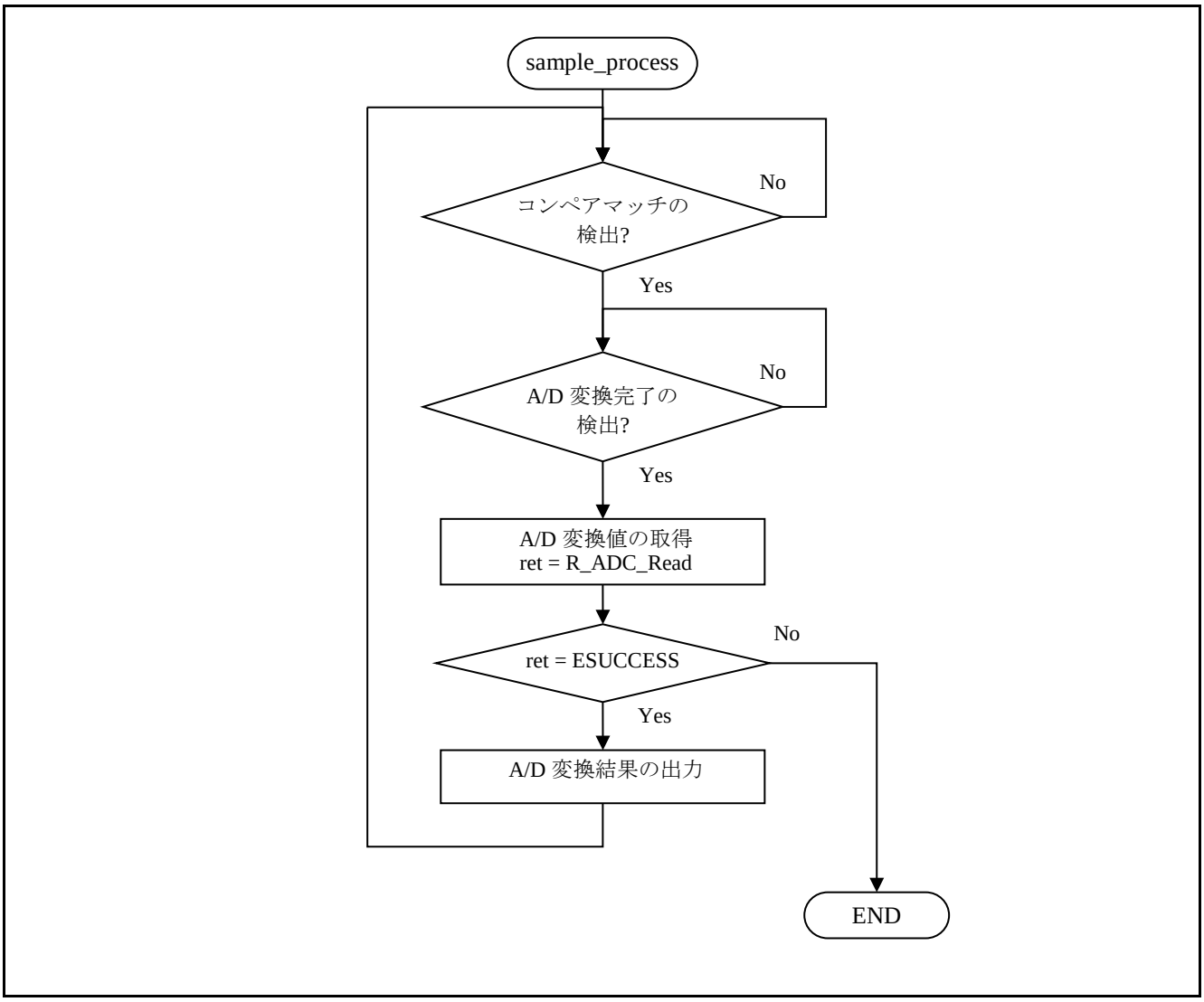
sample_process
コンペアマッチの
検出?
No
Yes
A/D 変換完了の
検出?
No
Yes
A/D 変換値の取得
ret = R_ADC_Read
ret = ESUCCESS
No
Yes
A/D 変換結果の出力
END

図 5.7　ADC 変換処理

5.15.2 cmtw_ch0_inhr_callback

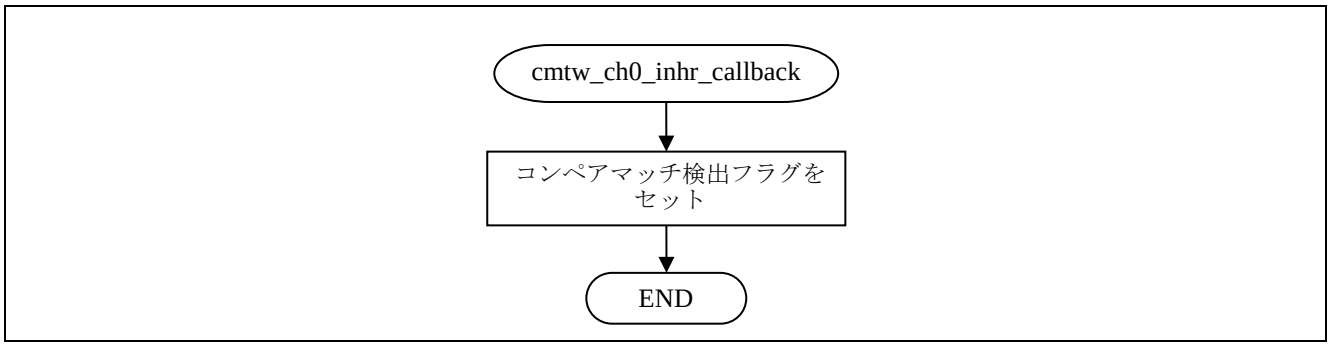

図 5.8 cmtw_ch0_inhr_callback

5.15.3 adc_inhr_callback

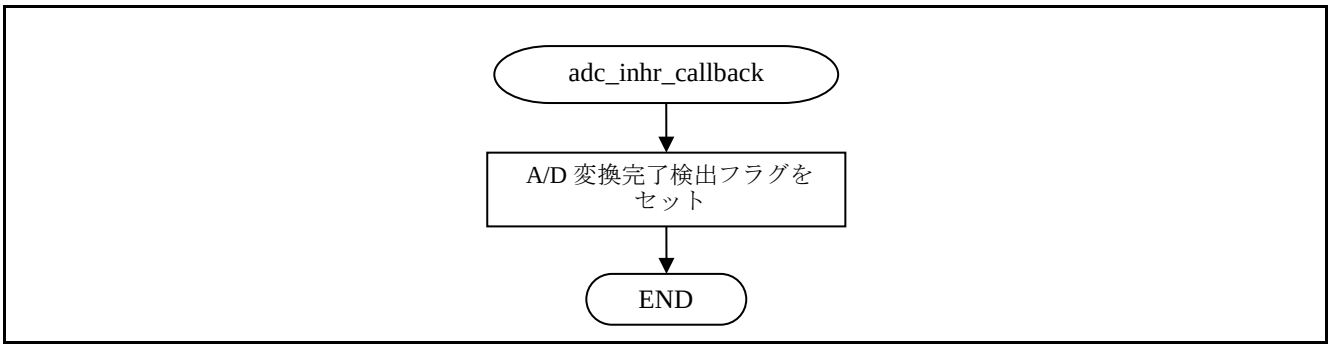

図 5.9 adc_inhr_callback

5.16 R_ADC_Control コマンド仕様

R_ADC_Control のコマンド仕様を記します。

表 5-45 R_ADC_Control のコマンド一覧

コマンド	概要
ADC_CMD_ENABLE_CHANS	A/D 変換チャネルの指定
ADC_CMD_ENABLE_TEMP_SENSOR	温度センサの初期設定
ADC_CMD_SET_SAMPLE_STATE_CNT	アナログ入力のサンプリング時間の設定
ADC_CMD_ENABLE_TRIG	同期、非同期トリガによる A/D 変換の開始を許可
ADC_CMD_DISABLE_TRIG	同期、非同期トリガによる A/D 変換の開始を禁止
ADC_CMD_SCAN_NOW	ソフトウェアトリガによる A/D 変換を開始
ADC_CMD_ENABLE_INT	スキャン終了後、S12ADI 割り込み発生の許可
ADC_CMD_DISABLE_INT	スキャン終了後、S12ADI 割り込み発生の禁止
ADC_CMD_ENABLE_INT_GROUPB	グループ B のスキャン終了後に S12GBADI 割り込み発生を許可
ADC_CMD_DISABLE_INT_GROUPB	グループ B のスキャン終了後に S12GBADI 割り込み発生を禁止
ADC_CMD_CHECK_SCAN_DONE	A/D 変換の確認
ADC_CMD_CHECK_SCAN_DONE_GROUPA	グループ A のスキャンを確認
ADC_CMD_CHECK_SCAN_DONE_GROUPB	グループ B のスキャンを確認

5.16.1 ADC_CMD_ENABLE_CHANS

ADC_CMD_ENABLE_CHANS

概　要	A/D 変換チャネルの指定
ヘッダ	r_adc_if.h
説　明	A/D 変換のチャネルを指定します。 パラメータは adc_ch_cfg_t 型変数の形で受け渡します。
パラメータ	adc_ch_cfg_t　p_args　A/D 変換を行うチャネルを指定します。
リターン値	ADC_SUCCESS　: チャネルの指定に成功 ADC_ERR_MISSING_PTR　: ポインタ引数が NULL ADC_ERR_ILLEGAL_ARG　: R_ADC_Open 関数の mode が ADC_MODE_SS_TEMPERATURE の場合 ADC_ERR_INVALID_ARG　: 引数の値が不正
補足	-

5.16.2 ADC_CMD_ENABLE_TEMP_SENSOR

ADC_CMD_ENABLE_TEMP_SENSOR		
概　要	温度センサの初期設定	
ヘッダ	r_adc_if.h	
説　明	温度センサの初期設定を行います。 パラメータはありません。NULL を指定してください。	
パラメータ	NULL	
リターン値	ADC_SUCCESS	: 温度センサの初期化に成功
	ADC_ERR_ILLEGAL_ARG	: R_ADC_Open 関数の mode が ADC_MODE_SS_TEMPERATURE 以外の場合
補足	-	

5.16.3 ADC_CMD_SET_SAMPLE_STATE_CNT

ADC_CMD_SET_SAMPLE_STATE_CNT		
概　要	アナログ入力のサンプリング時間の設定	
ヘッダ	r_adc_if.h	
説　明	アナログ入力のサンプリングの時間を設定します。 パラメータは adc_time_t 型変数の形で受け渡します。	
パラメータ	adc_time_t p_args	サンプリング時間を設定するチャネルと、サンプリング時間を指定します。 adc_time_t 構造体を参照
リターン値	ADC_SUCCESS	: アナログ入力のサンプリング時間の設定に成功
	ADC_ERR_MISSING_PTR	: ポインタ引数が NULL
	ADC_ERR_ILLEGAL_ARG	: 引数の値が不正
補足	-	

5.16.4 ADC_CMD_ENABLE_TRIG

ADC_CMD_ENABLE_TRIG		
概　要	同期、非同期トリガによる A/D 変換の開始を許可	
ヘッダ	r_adc_if.h	
説　明	同期、非同期トリガによる A/D 変換の開始を許可します。 パラメータはありません。NULL を指定してください。	
パラメータ	NULL	
リターン値	ADC_SUCCESS	: 同期、非同期トリガによる A/D 変換の開始の許可に成功
補足	-	

5.16.5 ADC_CMD_DISABLE_TRIG

ADC_CMD_DISABLE_TRIG		
概　要	同期、非同期トリガによる A/D 変換の開始を禁止	
ヘッダ	r_adc_if.h	
説　明	同期、非同期トリガによる A/D 変換の開始を禁止します。 パラメータはありません。NULL を指定してください。	
パラメータ	NULL	
リターン値	ADC_SUCCESS	: 同期、非同期トリガによる A/D 変換の開始の禁止に成功
補足	-	

5.16.6 ADC_CMD_SCAN_NOW

ADC_CMD_SCAN_NOW		
概　要	ソフトウェアトリガによる A/D 変換を開始	
ヘッダ	r_adc_if.h	
説　明	ソフトウェアトリガによる A/D 変換を開始します。 パラメータはありません。NULL を指定してください。	
パラメータ	NULL	
リターン値	ADC_SUCCESS	: A/D 変換の開始に成功
	ADC_ERR_SCAN_NOT_DONE	: A/D 変換中
補足	-	

5.16.7 ADC_CMD_ENABLE_INT

ADC_CMD_ENABLE_INT		
概　要	スキャン終了後、S12ADI 割り込み発生の許可	
ヘッダ	r_adc_if.h	
説　明	スキャン終了後、S12ADI 割り込み発生を許可します。 パラメータはありません。NULL を指定してください。	
パラメータ	NULL	
リターン値	ADC_SUCCESS	: スキャン終了後、S12ADI 割り込み発生の許可に成功
	ADC_ERR_ILLEGAL_ARG	: コールバック関数が未登録
補足	-	

5.16.8 ADC_CMD_DISABLE_INT

ADC_CMD_DISABLE_INT		
概　要	スキャン終了後、S12ADI 割り込み発生の禁止	
ヘッダ	r_adc_if.h	
説　明	スキャン終了後、S12ADI 割り込み発生を禁止します。 パラメータはありません。NULL を指定してください。	
パラメータ	NULL	
リターン値	ADC_SUCCESS	: スキャン終了後、S12ADI 割り込み発生の禁止に成功
補足	-	

5.16.9 ADC_CMD_ENABLE_INT_GROUPB

ADC_CMD_ENABLE_INT_GROUPB		
概　要	グループ B のスキャン終了後に S12GBADI 割り込み発生を許可	
ヘッダ	r_adc_if.h	
説　明	グループ B のスキャン終了後に S12GBADI 割り込み発生を許可します。 パラメータはありません。NULL を指定してください。	
パラメータ	NULL	
リターン値	ADC_SUCCESS	: グループ B のスキャン終了後の S12GBADI 割り込み発生の許可に成功
	ADC_ERR_ILLEGAL_ARG	: コールバック関数が未登録
補足	-	

5.16.10 ADC_CMD_DISABLE_INT_GROUPB

ADC_CMD_DISABLE_INT_GROUPB		
概　要	グループ B のスキャン終了後に S12GBADI 割り込み発生を禁止	
ヘッダ	r_adc_if.h	
説　明	グループ B のスキャン終了後に S12GBADI 割り込み発生を禁止します。 パラメータはありません。NULL を指定してください。	
パラメータ	NULL	
リターン値	ADC_SUCCESS	：グループ B のスキャン終了後の S12GBADI 割り込み発生の禁止に成功
補足	-	

5.16.11 ADC_CMD_CHECK_SCAN_DONE

ADC_CMD_CHECK_SCAN_DONE		
概　要	A/D 変換の確認	
ヘッダ	r_adc_if.h	
説　明	A/D 変換中か確認します。 パラメータはありません。NULL を指定してください。	
パラメータ	NULL	
リターン値	ADC_SUCCESS	：A/D 変換が終了
	ADC_ERR_SCAN_NOT_DONE	：A/D 変換中
補足	-	

5.16.12 ADC_CMD_CHECK_SCAN_DONE_GROUPA

ADC_CMD_CHECK_SCAN_DONE_GROUPA		
概　要	グループ A のスキャンを確認します。	
ヘッダ	r_adc_if.h	
説　明	グループ A のスキャンが終了したか確認します。 パラメータはありません。NULL を指定してください。	
パラメータ	NULL	
リターン値	ADC_SUCCESS	：グループ A のスキャンが終了
	ADC_ERR_SCAN_NOT_DONE	：グループ A がスキャン中
補足	-	

5.16.13 ADC_CMD_CHECK_SCAN_DONE_GROUPB

ADC_CMD_CHECK_SCAN_DONE_GROUPB		
概　要	グループ B のスキャンを確認します。	
ヘッダ	r_adc_if.h	
説　明	グループ B のスキャンが終了したか確認します。 パラメータはありません。NULL を指定してください。	
パラメータ	NULL	
リターン値	ADC_SUCCESS	：グループ B のスキャンが終了
	ADC_ERR_SCAN_NOT_DONE	：グループ B のスキャン中
補足	-	

5.17 R_CMT_Control コマンド仕様

R_CMT_Control のコマンド仕様を記します。

表 5-46 R_CMT_Control のコマンド一覧

定数名	内容
CMT_CMD_SET_TIME_CNT	CMT(W)のタイマカウント設定を行います。
CMT_CMD_SET_MODE	CMT(W)の動作モード、およびパラメータの設定を行います。
CMT_CMD_SET_PAUSE	タイマの一時停止を行います。
CMT_CMD_SET_RESUME	カウントを維持したままカウンタ動作の再開を行います。
CMT_CMD_SET_RESTART	カウントを 0 クリアし、カウンタ動作の再開を行います。
CMT_CMD_SET_ECM	ECM ダイナミックモードモードエラー出力の設定を行います。
CMTW_CMD_GET_STATUS	カウンタの動作状態を取得します。

5.17.1 CMT_CMD_SET_TIME_CNT

CMT_CMD_SET_TIME_CNT

概　要	CMT(W)のタイマカウント設定	
ヘッダ	r_cmt_if.h	
説　明	CMTW のタイマカウント設定を行います。	
	パラメータは cmtw_time_cnt_t 型変数の形で受け渡します。	
パラメータ	uint32_t pclk_div	PCLKD クロックの分周比を設定します。
	uint32_t cnt_size	カウンタサイズを設定します
	uint32_t clear_factor	タイマカウンタのクリア要因を設定します。
リターン値	CMT_SUCCESS	: 設定成功
	CMT_ERR_INVALID_ARG	: タイマカウント情報のメンバが不正な値
	CMT_ERR_TIMER_RUNNING	: タイマカウンタ動作中に実行
	CMT_ERR_MISSING_PTR	: ポインタ引数が不正
補足	-	

5.17.2 CMT_CMD_SET_MODE

CMT_CMD_SET_MODE

概　要	CMT(W)の動作モード設定	
ヘッダ	r_cmt_if.h	
説　明	CMT(W)の動作モードおよび各動作モードに対するパラメータの設定を行います。 パラメータは cmtw_mode_t 型変数の形で受け渡します。	
パラメータ	cmtw_compare_match_t compare_match	コンペアマッチパラメータを格納します。
	int32_t mode_enable	コンペアマッチ機能の ON/OFF を設定します。 true　: ON、false　: OFF
	uint32_t compare_match_cnt	コンペアマッチカウント値を設定します。 カウンタサイズが 16 ビットモードの場合、上位 16 ビットに設定した値は無視されます。
	int32_t intr_priority	コンペアマッチ割り込みの優先度を指定します。
	void (*p_ callback)(void)	コンペアマッチコールバック関数ポインタを設定します。 NULL を設定した場合、エラーとはなりませんがコンペアマッチ発生の通知は行われません。
	cmtw_output_compare_t output_compare[CMTW_OUTPUT_COMPARE_NUM]	アウトプットコンペアパラメータを格納します。 配列番号がアウトプットコンペア 0/1 に対応しています。
	int32_t mode_enable	アウトプットコンペア機能の ON/OFF を設定します。 true　: ON、false　: OFF
	uint32_t output_compare_cnt	アウトプットコンペアカウント値を設定します。 カウンタサイズが 16 ビットモードの場合、上位 16 ビットに設定した値は無視されます。
	uint32_t output_signal	アウトプットコンペア出力の信号値を設定します。
	int32_t intr_priority	アウトプットコンペア割り込みの優先度を指定します。
	void (*p_ callback)(void)	アウトプットコンペア 0/1 コールバック関数ポインタを設定します。 NULL を設定した場合、エラーとはなりませんがアウトプットコンペア発生の通知は行われません。
	cmtw_input_capture_t input_capture[CMTW_INPUT_CAPTURE_NUM]	インプットキャプチャパラメータを格納します。 配列番号がインプットキャプチャ 0/1 に対応しています。
	int32_t mode_enable	インプットキャプチャ機能の ON/OFF を設定します。 true　: ON　、false　: OFF
	uint32_t trigger	インプットキャプチャ実行のためのトリガを設定します。
	int32_t filter_enable	ノイズフィルタ機能の ON/OFF を設定します。 true　: ON　、false　: OFF
	int32_t intr_priority	インプットキャプチャ割り込みの優先度を指定します。
	void (*p_ callback) (uint32_t cnt_value)	インプットキャプチャ 0/1 コールバック関数ポインタを設定します。NULL を設定した場合、エラーとはなりませんがインプットキャプチャ発生の通知は行われません。
	uint32_t noise_filter_clk	ノイズフィルタに使用する PCLKD クロックの分周比を設定します。インプットキャプチャ 0/1 のノイズフィルタが 1 つ以上有効である場合に有効になります。
リターン値	CMT_SUCCESS	: 設定成功
	CMT_ERR_INVALID_ARG	: タイマカウント情報のメンバが不正な値
	CMT_ERR_TIMER_RUNNING	: タイマカウンタ動作中に実行
	CMT_ERR_MISSING_PTR	: ポインタ引数が不正
補足	-	

5.17.3 CMT_CMD_SET_PAUSE

CMT_CMD_SET_PAUSE

概　要	タイマー時停止	
ヘッダ	r_cmt_if.h	
説　明	CMT(W)のタイマカウントー時停止を行います。タイマカウンタが停止している最中に呼び出された場合は何も処理を実行せずにコマンドを終了します。	
パラメータ	なし	
リターン値	CMT_SUCCESS	：設定成功
	CMT_ERR_TIMER_STOP	：初期化後、一度もタイマカウントを開始せずに実行
補足	-	

5.17.4 CMT_CMD_SET_RESUME

CMT_CMD_SET_RESUME

概　要	タイマカウントを維持したままのカウント再開	
ヘッダ	r_cmt_if.h	
説　明	CMT(W)の一時停止実行時のタイマカウント値を維持したままカウントの再開を行います。タイマカウンタが動作している最中に呼び出された場合は何も処理を実行せずにコマンドを終了します。	
パラメータ	なし	
リターン値	CMT_SUCCESS	：設定成功
	CMT_ERR_TIMER_STOP	：初期化後、一度もタイマカウントを開始せずに実行
	CMT_ERR_TIMER_RUNNING	：タイマカウンタ動作中に実行
補足	-	

5.17.5 CMT_CMD_SET_RESTART

CMT_CMD_SET_RESTART

概　要	タイマカウントをクリア後、カウント再開	
ヘッダ	r_cmt_if.h	
説　明	CMT(W)の一時停止実行時のタイマカウント値をクリアし、カウントの再開を行います。タイマカウンタが動作している最中に呼び出された場合は何も処理を実行せずにコマンドを終了します。	
パラメータ	なし	
リターン値	CMT_SUCCESS	：設定成功
	CMT_ERR_TIMER_STOP	：初期化後、一度もタイマカウントを開始せずに実行
	CMT_ERR_TIMER_RUNNING	：タイマカウンタ動作中に実行
補足	-	

5.17.6 CMT_CMD_SET_ECM

CMT_CMD_SET_ECM

概　要	ECM ダイナミックモードエラー出力設定	
ヘッダ	r_cmt_if.h	
説　明	ECM ダイナミックモードエラー出力設定を行います。 パラメータは cmtw_ecm_t 型変数の形で受け渡します。	
パラメータ	int32_t ecm_enable	ECM ダイナミックモードエラー出力の有効/無効を設定します。 true　：ECM ダイナミックモードエラー出力有効 false　：ECM ダイナミックモードエラー出力無効
	uint32_t output_compare_num	ECM ダイナミックモードエラー出力を行うアウトプットコンペア番号を選択します。
リターン値	CMT_SUCCESS	：設定成功
	CMT_ERR_INVALID_ARG	：アウトプットコンペア番号設定が不正な値
	CMT_ERR_TIMER_RUNNING	：タイマカウンタ動作中に実行
	CMT_ERR_MISSING_PTR	：ポインタ引数が不正
補足	・タイマの動作開始前に、本コマンドで選択したアウトプットコンペア番号に対し、別途アウトプットコンペア出力設定を有効にしてください ・タイマが停止条件(R_CMT_Close 実行、タイマ非周期動作終了)を満たした際、設定は初期化されます ・一度に ECM ダイナミックモードエラー出力に設定できるアウトプットコンペア信号は HW 全体で１つのみで、最後に行った設定で上書きされます。 ・初期化直後、ECM ダイナミックモードエラー出力は disable に設定されています	

5.17.7 CMTW_CMD_GET_STATUS

CMTW_CMD_GET_STATUS

概　要	タイマの状態取得	
ヘッダ	r_cmt_if.h	
説　明	タイマの動作状態を取得します。 動作状態は uint32_t のポインタ変数で受け取ります。 下記パラメータの変数名は一例です。	
パラメータ	uint32_t ＊pcmt_status	タイマの動作状態を格納します。 動作状態として以下の値をリターンします。 CMTW_STATUS_STOP：タイマ停止中 CMTW_STATUS_RUNNING：タイマ動作中
リターン値	CMT_SUCCESS	：取得成功
	CMT_ERR_INVALID_ARG	：タイマ動作状態取得パラメータのポインタが異常値
補足	-	

5.18 R_ELC_Control コマンド仕様

R_ELC_Control のコマンド仕様を記します。

表 5-47 R_ELC_Control のコマンド一覧

定数名	内容
ELC_CMD_SET_EVENT_MTU	MTU モジュールに対するイベントリンク設定を行います。
ELC_CMD_SET_EVENT_CMT	CMT モジュールに対するイベントリンク設定を行います。
ELC_CMD_SET_EVENT_DSMIF	ΔΣユニットモジュールに対するイベントリンク設定を行います。
ELC_CMD_SET_EVENT_S12AD	12 ビット A/D コンバータに対するイベントリンク設定を行います。
ELC_CMD_SET_EVENT_INTR	ELC の割り込み要求信号に対するイベントリンク設定を行います。
ELC_CMD_SET_EVENT_OUT_PORT_GROUP	出力ポートグループに対するイベントリンク設定を行います。
ELC_CMD_SET_EVENT_IN_PORT_GROUP	入力ポートグループに対するイベントリンク設定を行います。
ELC_CMD_SET_EVENT_SINGLE_PORT	シングルポートの設定、およびポートに対するイベントリンク設定を行います。
ELC_CMD_SET_EVENT_CMTW	CMTW0 に対するイベントリンクパラメータ設定を行います。
ELC_CMD_SET_EVENT_TPU	TPU モジュールに対するイベントリンク設定を行います。
ELC_CMD_SET_EVENT_GPT	GPT モジュールに対するイベントリンク設定を行います。
ELC_CMD_SET_PORT_GROUP	ポートグループの設定を行います。
ELC_CMD_SET_SOFTWARE_EVENT	ELC のソフトウェアイベントの発行を行います。
ELC_CMD_GET_PORT_GROUP_VALUE	ポートグループの信号値を取得します。

5.18.1 ELC_CMD_SET_EVENT_MTU

ELC_CMD_SET_EVENT_MTU

概　要	MTU0、3、4 に対するイベントリンクパラメータ設定	
ヘッダ	r_elc_if.h	
説　明	MTU0、3、4 に対する ELC のイベントリンクパラメータの設定を行います。 パラメータは elc_cmd_mtu_t 型変数の形で受け渡します。	
パラメータ	uint32_t elc_mtu_ch	設定対象の MTU ユニット番号を指定します。
	int32_t event_link_enable	MTU に対するイベントリンクの ON/OFF を設定します。 true　: ON false　: OFF
	uint32_t resource	イベントリンク元となるイベント信号の設定を行います。
	uint32_t action	MTU に対するイベントリンク発生時の動作を設定します。
リターン値	ELC_SUCCESS	: 設定成功
	ELC_ERR_INVALID_ARG	: イベントリンク情報のメンバが不正な値
	ELC_ERR_MISSING_PTR	: ポインタ引数が不正
補足	-	

5.18.2 ELC_CMD_SET_EVENT_CMT

ELC_CMD_SET_EVENT_CMT		
概　要	CMT1 に対するイベントリンクパラメータ設定	
ヘッダ	r_elc_if.h	
説　明	CMT1 に対する ELC のイベントリンクパラメータの設定を行います。 パラメータは elc_cmd_cmt_t 型変数の形で受け渡します。	
パラメータ	int32_t event_link_enable	CMT1 に対するイベントリンクの ON/OFF を設定します。 true　: ON false　: OFF
	uint32_t resource	イベントリンク元となるイベント信号の設定を行います。
	uint32_t action	CMT1 に対するイベントリンク発生時の動作を設定します。
リターン値	ELC_SUCCESS	: 設定成功
	ELC_ERR_INVALID_ARG	: イベントリンク情報のメンバが不正な値
	ELC_ERR_MISSING_PTR	: ポインタ引数が不正
補足	-	

5.18.3 ELC_CMD_SET_EVENT_DSMIF

ELC_CMD_SET_ EVENT_DSMIF		
概　要	ΔΣユニット 0/1 トリガ 0/1 に対するイベントリンクパラメータ設定	
ヘッダ	r_elc_if.h	
説　明	ΔΣユニット 0/1 トリガ 0/1 に対する ELC のイベントリンクパラメータの設定を行います。 パラメータは elc_cmd_dsmif_t 型変数の形で受け渡します。	
パラメータ	uint32_t elc_dsmif_ch	設定対象のΔΣユニット 0/1 トリガ 0/1 の番号を指定します。
	int32_t event_link_enable	ΔΣユニットに対するイベントリンクの ON/OFF を設定します。 true　: ON false　: OFF
	uint32_t resource	イベントリンク元となるイベント信号の設定を行います。
リターン値	ELC_SUCCESS	: 設定成功
	ELC_ERR_INVALID_ARG	: イベントリンク情報のメンバが不正な値
	ELC_ERR_MISSING_PTR	: ポインタ引数が不正
補足	-	

5.18.4 ELC_CMD_SET_EVENT_S12AD

ELC_CMD_SET_EVENT_S12AD		
概　要	12 ビット A/D コンバータ 0、1 に対するイベントリンクパラメータ設定	
ヘッダ	r_elc_if.h	
説　明	12 ビット A/D コンバータ 0、1 に対する ELC のイベントリンクパラメータの設定を行います。パラメータは elc_cmd_s12ad_t 型変数の形で受け渡します。	
パラメータ	uint32_t elc_s12ad_ch	設定対象の 12 ビット A/D コンバータの番号を指定します。
	int32_t event_link_enable	12 ビット A/D コンバータに対するイベントリンクの ON/OFF を設定します。 true　: ON false　: OFF
	uint32_t resource	イベントリンク元となるイベント信号の設定を行います。
リターン値	ELC_SUCCESS	: 設定成功
	ELC_ERR_INVALID_ARG	: イベントリンク情報のメンバが不正な値
	ELC_ERR_MISSING_PTR	: ポインタ引数が不正
補足	-	

5.18.5 ELC_CMD_SET_EVENT_INTR

ELC_CMD_SET_EVENT_INTR		
概　要	ELC の割り込み要求信号 1、2 に対するイベントリンクパラメータ設定	
ヘッダ	r_elc_if.h	
説　明	ELC の割り込み要求信号 1、2 に対する ELC のイベントリンクパラメータの設定を行います。パラメータは elc_cmd_intr_t 型変数の形で受け渡します。	
パラメータ	uint32_t elc_intr_num	割り込み要求信号の番号を指定します。
	int32_t event_link_enable	割り込み要求信号 1、2 に対するイベントリンクの ON/OFF を設定します。 true　: ON false　: OFF
	uint32_t resource	イベントリンク元となるイベント信号の設定を行います。
	int32_t intr_priority	割り込みの優先度を指定します。
	void (*p_ callback) (void)	イベントリンクコールバック関数ポインタを設定します。NULL を設定した場合、エラーとはなりませんが割り込み発生通知は行われません。
リターン値	ELC_SUCCESS	: 設定成功
	ELC_ERR_INVALID_ARG	: イベントリンク情報のメンバが不正な値
	ELC_ERR_MISSING_PTR	: ポインタ引数が不正
補足	-	

5.18.6 ELC_CMD_SET_EVENT_OUT_PORT_GROUP

ELC_CMD_SET_EVENT_OUT_PORT_GROUP		
概　要	出力ポートグループ 1、2 に対するイベントリンクパラメータ設定	
ヘッダ	r_elc_if.h	
説　明	出力ポートグループ 1、2 に対する ELC のイベントリンクパラメータの設定を行います。パラメータは elc_cmd_out_port_group_t 型変数の形で受け渡します。	
パラメータ	uint32_t elc_out_port_group_num	出力ポートグループの番号を指定します。
	int32_t event_link_enable	出力ポートグループ 1、2 に対するイベントリンクの ON/OFF を設定します。 true　: ON false　: OFF
	uint32_t resource	イベントリンク元となるイベント信号の設定を行います。
	uint32_t action	出力ポートグループ 1、2 に対するイベントリンク発生時の動作を設定します。
	uint8_t init_value	出力ポートグループの初期出力値を設定します。action に、を設定している場合は設定値が無効になります。
リターン値	ELC_SUCCESS	: 設定成功
	ELC_ERR_INVALID_ARG	: イベントリンク情報のメンバが不正な値
	ELC_ERR_MISSING_PTR	: ポインタ引数が不正
補足	・ローテート出力を設定している際にローテート状態を初期状態に戻す場合、本コマンドで初期状態のパラメータを設定しなおしてください。	

5.18.7 ELC_CMD_SET_EVENT_IN_PORT_GROUP

ELC_CMD_SET_EVENT_IN_PORT_GROUP

概　要	入力ポートグループ 1、2 に対するイベントリンクパラメータ設定	
ヘッダ	r_elc_if.h	
説　明	出力ポートグループ 1、2 に対する ELC のイベントリンクパラメータの設定を行います。 パラメータは elc_cmd_in_port_group_t 型変数の形で受け渡します。	
パラメータ	uint32_t elc_in_port_group_num	出力ポートグループの番号を指定します。
	int32_t event_link_enable	入力ポートグループ 1、2 に対するイベントリンクの ON/OFF を設定します。 true　：ON false　：OFF
	uint32_t resource	イベントリンク元となるイベント信号の設定を行います。
	int32_t overwrite_enable	イベント発生時、バッファへの信号値の上書きを有効にするかを設定します。 true　：上書き有効 false　：上書き無効
リターン値	ELC_SUCCESS　：設定成功	
	ELC_ERR_INVALID_ARG　：イベントリンク情報のメンバが不正な値	
	ELC_ERR_MISSING_PTR　：ポインタ引数が不正	
補足	イベント発生時にバッファへの信号値の上書きを無効にする場合、バッファ値の読み出しを行うまで次のイベントが無効になります。 バッファ値の読み出しは ELC_OUT_GROUP_BUFFER を使用してください。	

5.18.8 ELC_CMD_SET_EVENT_SINGLE_PORT

ELC_CMD_SET_EVENT_SINGLE_PORT

概　要	シングルポートの登録、およびシングルポートに対するイベントリンクパラメータ設定	
ヘッダ	r_elc_if.h	
説　明	シングルポート 0、1、2、3 に対する I/O ポートの登録、および ELC のイベントリンクパラメータの設定を行います。パラメータは elc_cmd_single_port_t 型変数の形で受け渡します。	
パラメータ	uint32_t elc_single_port_num	シングルポートの番号を指定します。
	uint32_t port_symbol	シングルポートとして設定するポートシンボルを選択します。
	uint32_t port_num	シングルポートして設定する I/O ポート番号を 0～7 で指定します。
	int32_t event_link_enable	出力シングルポートに対するイベントリンクの ON/OFF を設定します。 ON に設定した場合はイベント発生待ち、および発生時のシングルポートからのデータ出力を行います OFF に設定した場合は入力シングルポートへのデータ入力待ちを行い、データ入力を検出後、イベントリンク要求の発行を行います。 true　：ON false　：OFF
	uint32_t event_direction	イベントリンク時のイベント入力/イベント出力の選択を行います。
	uint32_t resource	イベントリンク元となるイベント信号の設定を行います。 event_link_enable が true、かつ signal_direction が ELC_SINGLE_EVENT_OUTPUT の場合のみ有効です。
	uint32 _t output_action	イベントリンク発生時の出力シングルポートの動作を設定します。 event_link_enable が true、かつ signal_direction が ELC_SINGLE_EVENT_OUTPUT の場合のみ有効です。
	uint32_t input_trigger	入力シングルポートのデータ入力検出トリガを指定します。 event_link_enable が true、かつ signal_direction が ELC_SINGLE_EVENT_INPUT の場合のみ有効です。
リターン値	ELC_SUCCESS　：設定成功	
	ELC_ERR_INVALID_ARG　：イベントリンク情報のメンバが不正な値	
	ELC_ERR_MISSING_PTR　：ポインタ引数が不正	
補足	・シングルポートとして登録した I/O ポートのデータ入力 / 出力方向の設定は本サンプルドライバでは設定していません。I/O ドライバなどで設定を行ってください。 ・任意の I/O ポートに対し、シングルポートとポートグループの両方が設定されている場合、入力設定になっている場合は両方の機能が有効となります。 出力設定になっている場合はポートグループの設定のみが有効となります。	

5.18.9 ELC_CMD_SET_EVENT_CMTW

ELC_CMD_SET_EVENT_CMTW		
概　要	CMTW0 に対するイベントリンクパラメータ設定	
ヘッダ	r_elc_if.h	
説　明	CMTW0 に対する ELC のイベントリンクパラメータの設定を行います。 パラメータは elc_cmd_cmtw_t 型変数の形で受け渡します。	
パラメータ	int32_t event_link_enable	CMTW0 に対するイベントリンクの ON/OFF を設定します。 true　: ON false　: OFF
	uint32_t resource	イベントリンク元となるイベント信号の設定を行います。
	uint32_t action	CMTW0 に対するイベントリンク発生時の動作を設定します。
リターン値	ELC_SUCCESS	: 設定成功
	ELC_ERR_INVALID_ARG	: イベントリンク情報のメンバが不正な値
	ELC_ERR_MISSING_PTR	: ポインタ引数が不正
補足	-	

5.18.10 ELC_CMD_SET_EVENT_TPU

ELC_CMD_SET_EVENT_TPU		
概　要	TPU0、1、2、3 に対するイベントリンクパラメータ設定	
ヘッダ	r_elc_if.h	
説　明	TPU0、1、2、3 に対する ELC のイベントリンクパラメータの設定を行います。 パラメータは elc_cmd_tpu_t 型変数の形で受け渡します。	
パラメータ	uint32_t elc_tpu_ch	TPU の番号を指定します。
	int32_t event_link_enable	TPU0、1、2、3 に対するイベントリンクの ON/OFF を設定します。 true　: ON false　: OFF
	uint32_t resource	イベントリンク元となるイベント信号の設定を行います。
	uint32_t action	TPU0、1、2、3 に対するイベントリンク発生時の動作を設定します。
リターン値	ELC_SUCCESS	: 設定成功
	ELC_ERR_INVALID_ARG	: イベントリンク情報のメンバが不正な値
	ELC_ERR_MISSING_PTR	: ポインタ引数が不正
補足	-	

5.18.11 ELC_CMD_SET_EVENT_GPT

ELC_CMD_SET_EVENT_GPT

概　要	GPT0、1、2、3 に対するイベントリンクパラメータ設定	
ヘッダ	r_elc_if.h	
説　明	GPT0、1、2、3 に対する ELC のイベントリンクパラメータの設定を行います。 パラメータは elc_cmd_gpt_t 型変数の形で受け渡します。	
パラメータ	uint32_t elc_gpt_ch	GPT の番号を指定します。
	int32_t event_link_enable	GPT0、1、2、3 に対するイベントリンクの ON/OFF を設定します。 true　：ON false　：OFF
	uint32_t resource	イベントリンク元となるイベント信号の設定を行います。
	uint32_t action	GPT0、1、2、3 に対するイベントリンク発生時の動作を設定します。
リターン値	ELC_SUCCESS	：設定成功
	ELC_ERR_INVALID_ARG	：イベントリンク情報のメンバが不正な値
	ELC_ERR_MISSING_PTR	：ポインタ引数が不正
補足	-	

5.18.12 ELC_CMD_SET_PORT_GROUP

ELC_CMD_SET_PORT_GROUP

概　要	ポートグループの設定	
ヘッダ	r_elc_if.h	
説　明	ポートグループの設定を行います。 パラメータは elc_cmd_port_group_t 型変数の形で受け渡します。	
パラメータ	uint32_t port_group_num	設定対象となるポートグループ番号を選択します。
	uint8_t port_group_bit	ポートグループ指定するポート番号をビット値で指定します。 0～7 ビットが I/O ポートの 0～7 のポート番号に対応しており、各ビットが 1 の場合対応するポートがポートグループとして設定されます。
	uint32_t trigger	ポートグループがイベントリンク元として動作する場合、イベント信号を出力するトリガを指定します。
リターン値	ELC_SUCCESS	：設定成功
	ELC_ERR_INVALID_ARG	：イベントリンク情報のメンバが不正な値
	ELC_ERR_MISSING_PTR	：ポインタ引数が不正
補足	・ポートグループの入力/出力設定は本サンプルドライバでは設定していません。I/O ドライバなどで設定を行ってください。 ・任意の I/O ポートに対し、シングルポートとポートグループの両方が設定されている場合、入力設定になっている場合は両方の機能が有効となります。 出力設定になっている場合はポートグループの設定のみが有効となります。 ・ポートグループ 1 はポート番号：ポート B に、ポートグループ 2 はポート番号：ポート E に対応します。	

5.18.13 ELC_CMD_SET_SOFTWARE_EVENT

ELC_CMD_SET_SOFTWARE_EVENT	
概　要	ELC のソフトウェアイベントの発行
ヘッダ	r_elc_if.h
説　明	ELC のソフトウェアイベントの発行を行います。
パラメータ	-
リターン値	-
補足	-

5.18.14 ELC_CMD_GET_PORT_GROUP_VALUE

ELC_CMD_GET_PORT_GROUP_VALUE		
概　要	ポートグループ信号値の取得	
ヘッダ	r_elc_if.h	
説　明	ポートグループの信号値を取得します。 パラメータは elc_get_port_value_t 型変数の形で受け渡しします。	
パラメータ	uint32_t elc_port_group_num	値を取得するポートグループ番号を指定します。
	uint8_t port_value	入力ポートグループの信号値が格納されます。
リターン値	ELC_SUCCESS	: 取得成功
	ELC_ERR_INVALID_ARG	: ポートグループ番号指定が不正
	ELC_ERR_MISSING_PTR	: ポインタ引数が不正
補足	-	

6. サンプルプログラム

サンプルプログラムは、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。

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ

http://japan.renesas.com/

お問合せ先

http://japan.renesas.com/contact/

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2016.12.22	—	新規発行

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. 未使用端子の処理

【注意】未使用端子は、本文の「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。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。

2. 電源投入時の処置

【注意】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. リザーブアドレスのアクセス禁止

【注意】リザーブアドレスのアクセスを禁止します。

アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレスがあります。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。

4. クロックについて

【注意】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。

5. 製品間の相違について

【注意】型名の異なる製品に変更する場合は、事前に問題ないことをご確認ください。

同じグループのマイコンでも型名が違うと、内部メモリ、レイアウトパターンの相違などにより、特性が異なる場合があります。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。

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。お客様の機器・システムの設計において、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には、お客様の責任において行ってください。これらの使用に起因して、お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
2. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
3. 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に関し、当社は、何らの責任を負うものではありません。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. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製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。
5. 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）、交通用信号機器、
   防災・防犯装置、各種安全装置等
   当社製品は、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（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。たとえ、意図しない用途に当社製品を使用したことによりお客様または第三者に損害が生じても、当社は一切その責任を負いません。なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障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。
7. 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、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、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。
8. 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。お客様が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。
9. 本資料に記載されている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。また、当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途に使用しないでください。当社製品または技術を輸出する場合は、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
11. 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。

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その総株主の議決権の過半数を直接または間接に保有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。

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/contact/