RZ/T1 グループ
CMTW & ELC サンプルプログラム

R01AN2600JJ0140
Rev.1.40
2018.06.07

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。

CMTW & ELC サンプル 2014.11.04 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、ポテンションメータの入力電圧を A/D 変換します。
- 変換結果を 4 段階に識別し LED0、LED1、LED2、LED3 に表示します。

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。

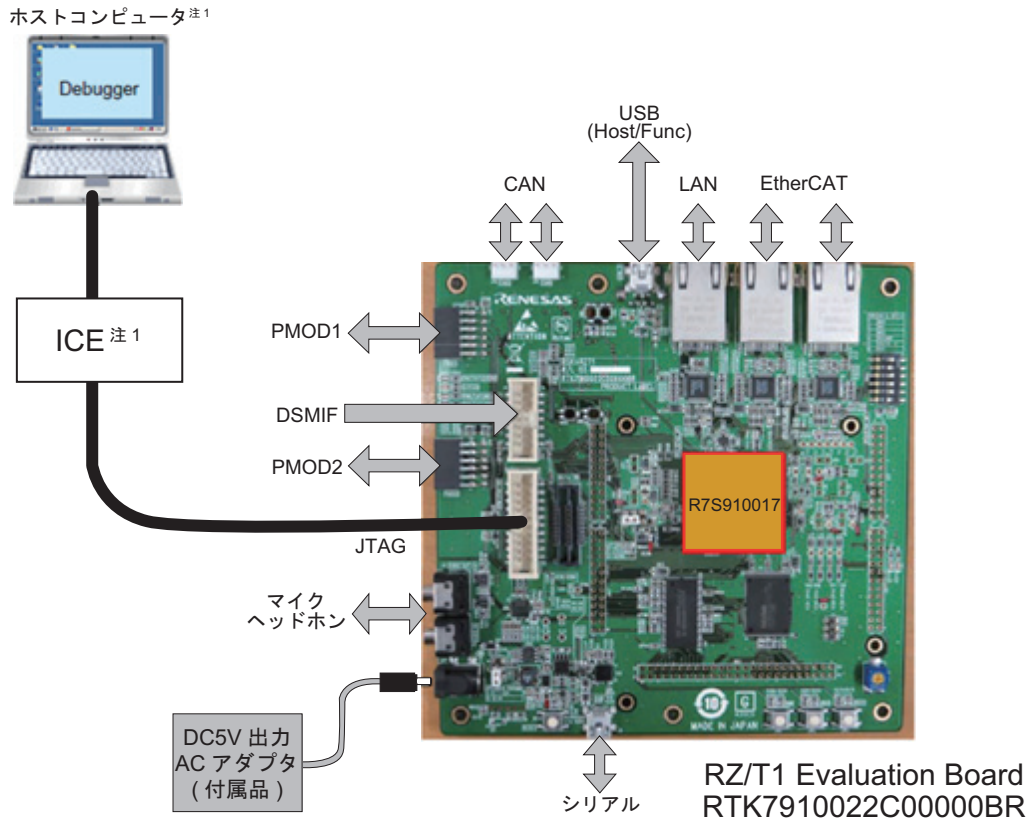
目　次

1.　仕様

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
RZ/T1内臓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内臓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
低消費電力低減機能	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）
I/Oポート（PM3,PM2,P56,PF7）	LED制御
ADC（AN007）	ポテンションメータの入力電圧のA/D変換
ポテンションメータ	ADCへ可変した電圧を入力



注 1.　別途お客様でご用意いただく必要があります。

図 1.1　動作環境

2. 動作環境

本アプリケーションノートのサンプルコードは、下記の環境を想定しています。

表2.1 動作環境

項目	内容
使用マイコン	RZ/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 8.20.2 Arm製 DS-5™ 5.26.2 RENESAS製 e2studio 6.1.0
動作モード	SPIブートモード 16ビットバスブートモード
使用ボード	RZ/T1 Evaluation Board （RTK7910022C00000BR）
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..、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..、型名 : MX25L51245G • ポテンショメータ（AN007） • LED LED0（PF7）、LED1（P56）、LED2（P77）、LED3（PA0）

3. 関連アプリケーションノート

本アプリケーションノートに関連するアプリケーションノートを以下に示します。併せて参照してください。

- RZ/T1 グループ 初期設定 アプリケーションノート（R01AN2554JJ）
- RZ/T1 グループ ADC サンプルプログラム アプリケーションノート（R01AN2599JJ）

注 . 本アプリケーションノートで記載しないレジスタに関しては、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のまま使用します。

4. 周辺機能説明

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）、低消費電力低減機能、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）についての基本的な内容は、『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。

5. ハードウェア説明

5.1 ハードウェア構成例

図 5.1 にサンプルコードで使用されるハードウェアのハードウェア構成例を示します。

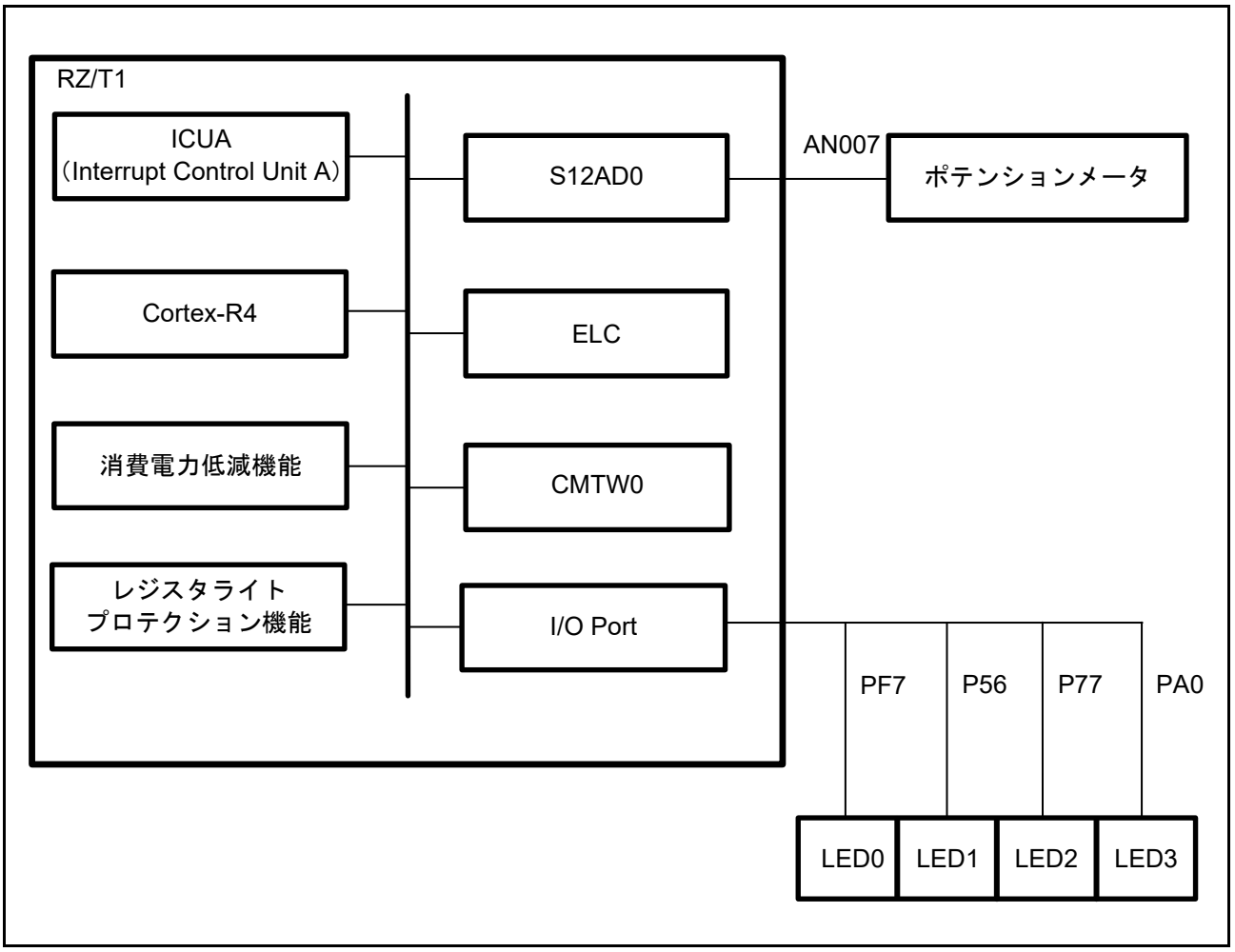


図 5.1 ハードウェア構成例

5.2 使用端子一覧

表 5.1 に使用端子と機能を示します。

表5.1 使用端子と機能

端子名	入出力	内容
AN007	入力	ポテンションメータ
PF7	出力	LED0制御
P56	出力	LED1制御
P77	出力	LED2制御
PA0	出力	LED3制御

6.　ソフトウェア説明

6.1　動作概要

CMTW & ELC サンプルプログラムの機能概要を**表 6.1　動作概要**に示します。また、**図 6.1** にシステムブロック図を示します。

表6.1　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4段階に識別し、それぞれの段階に割り当てたLEDを点灯させる 5. A/D変換結果が最大値（第4識別レベル）を超えたときはサンプルプログラムの処理を終了する
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動作で動作
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）

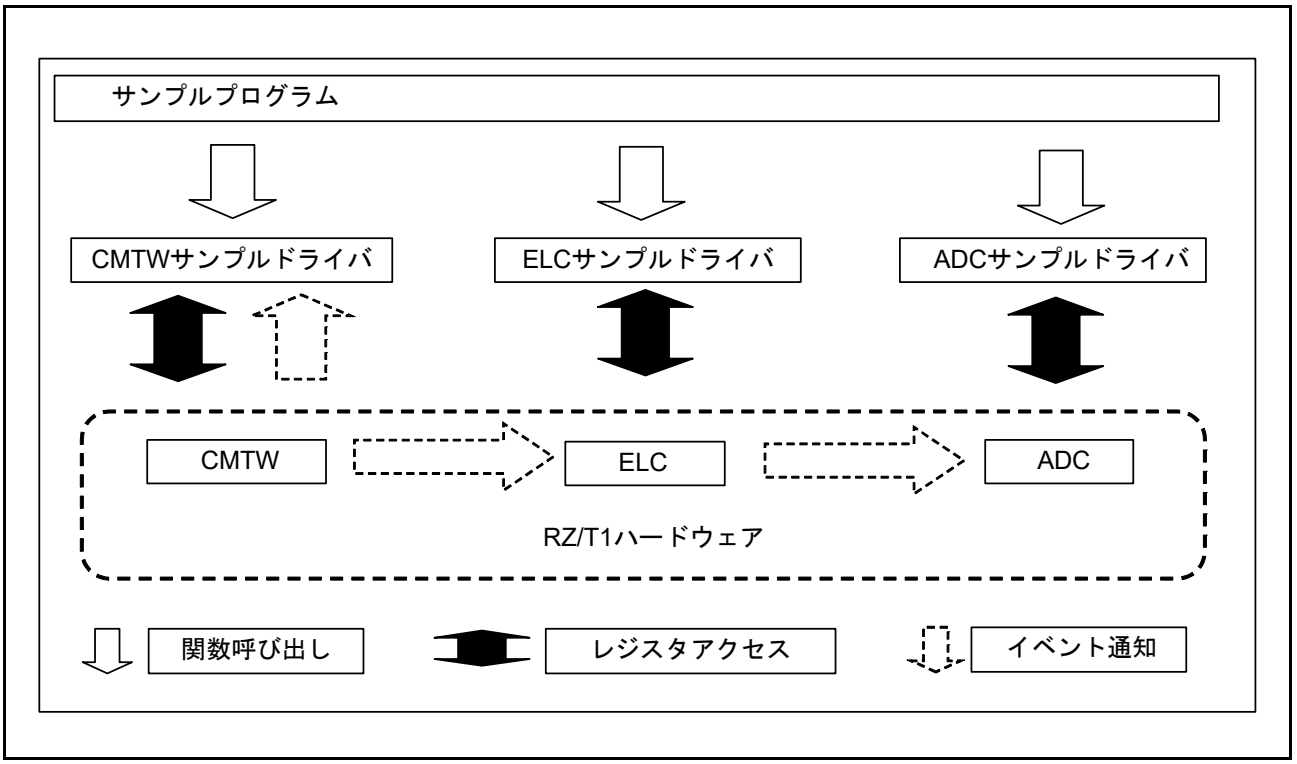


図 6.1　システムブロック図

6.1.1 プロジェクト設定

開発環境となる EWARM 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。

6.1.2 使用準備

本サンプルプログラムの実行準備は必要ありません。

6.2 メモリマップ

RZ/T1 グループのアドレス空間と RZ/T1 評価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。

6.2.1 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。

6.2.2 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。

6.2.3 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。

6.3 使用割り込み一覧

表 6.2 にサンプルコードで使用する割り込みを示します。

表6.2 サンプルコード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

6.4 固定幅整数一覧

表 6.3 にサンプルコードで使用する固定幅整数を示します。

表6.3 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）

6.5　定数／エラーコード一覧

表 6.4 にサンプルコード上で各ドライバに対して使用する定数、**表 6.5** ～**表 6.17** に CMTW ドライバで使用する定数、**表 6.18** ～**表 6.40** に ELC ドライバで対して使用する定数を示します。

なお、本サンプルコードでは ELC に対して定数は使用しません。

6.5.1　サンプルプログラムで使用する定数

表 6.4　サンプルコードで使用する定数

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。
CMTW_INTR_PRI	7	CMTW ドライバに設定する割り込み優先度
ELC_INTR_PRI	7	ELC ドライバに設定する割り込み優先度
ADC_PORT_PDR_OUT	3	LED に対応するポート方向設定 出力方向を設定
ADC_PORT_PMR_IO_SET	0	LED に対応するポート端子機能設定 汎用入出力ポートとして使用
ADC_LED_OFF	0	LED に対応するポート出力値設定 LED 消灯
ADC_LED_ON	1	LED に対応するポート出力値設定 LED 点灯
ADC_ADI_PRI	7	ADC ドライバ割り込み優先度設定
ADC_LVL0	0	A/D 変換値レベル 0 値（変換最小値）
ADC_LVL1	820	A/D 変換値レベル 1 の閾値
ADC_LVL2	1639	A/D 変換値レベル 2 の閾値
ADC_LVL3	2458	A/D 変換値レベル 3 の閾値
ADC_LVL4	3277	A/D 変換値レベル 4 の閾値

6.5.2 CMTW ドライバで使用する定数

表6.5 サンプルコードのエラーコード（CMTW）

定数名	設定値	内容
CMTW_SUCCESS	0	関数正常終了
CMTW_ERR_INVALID_CH	1	存在しないチャネルを指定
CMTW_ERR_INVALID_ARG	2	設定パラメータが異常な値
CMTW_ERR_NOT_OPENED	3	未初期化状態で関数を実行
CMTW_ERR_NOT_CLOSED	4	APIの二重初期化を実行
CMTW_ERR_TIMER_RUNNING	5	タイマ動作中に関数を実行
CMTW_ERR_TIMER_STOP	6	タイマカウント停止状態での関数実行
CMTW_ERR_MISSING_PTR	7	ポインタ引数が不正

表6.6 CMTWバージョン定義定数

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

表6.7 CMTW ドライバチャネル番号定義値

定数名	値	内容
CMTW_CH_0	0	チャネル0を指定
CMTW_CH_1	1	チャネル1を指定
CMTW_CH_NUM	2	正常設定値範囲確認に使用、またCMTWチャネル数を示す

表6.8 CMTWコマンド定義

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

表6.9 タイマクロック分周比設定定義値

定数名	値	内容
CMTW_PCLK_DIV_8	0	タイマ分周比PCLKD/8を選択
CMTW_PCLK_DIV_32	1	タイマ分周比PCLKD/32を選択
CMTW_PCLK_DIV_128	2	タイマ分周比PCLKD/128を選択
CMTW_PCLK_DIV_512	3	タイマ分周比PCLKD/512を選択値
CMTW_PCLK_DIV_MAX_OVER	4	正常設定値範囲確認用

表6.10 タイマカウンタ長設定定義値

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6.11 タイマカウンタクリア要因設定定義値

定数名	値	内容
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6.12 アウトプットコンペア出力設定定義値

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6.13 インプットキャプチャ検出トリガ設定定義値

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／立ち下がりエッジに設定
CMTW_IN_CAPTURE_TRIGGER_MAX_OVER	3	正常設定値範囲確認用

表6.14 ノイズフィルタクロック分周比設定定義値

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6.15 インプットキャプチャ番号定義値

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6.16 アウトプットコンペア番号定義値

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6.17 タイマ動作状態定義値

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

6.5.3 ELC ドライバで使用する定数

表6.18 サンプルコードのエラーコード（ELC）

定数名	設定値	内容
ELC_SUCCESS_OK	0	関数正常終了
ELC_ERR_INVALID_ARG	1	設定パラメータが異常な値
ELC_ERR_NOT_OPENED	2	未初期化状態で関数を実行
ELC_ERR_NOT_CLOSED	3	APIの二重初期化を実行
ELC_ERR_MISSING_PTR	4	ポインタ引数が不正

表6.19 ELC ドライババージョン定義値

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6.20 ELCコマンド定義

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6.21 イベントリンクリソース設定定義値 (1 / 3)

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
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
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

表6.21 イベントリンクリソース設定定義値 (2 / 3)

定数名	値	内容
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
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

表6.21 イベントリンクリソース設定定義値 (3 / 3)

定数名	値	内容
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6.22 MTUチャネル番号定義値

定数名	値	内容
ELC_MTU_CH_0	0	MTUのch0を示す
ELC_MTU_CH_3	1	MTUのch1を示す
ELC_MTU_CH_4	2	MTUのch2を示す
ELC_MTU_CH_NUM	3	正常設定値範囲確認用

表6.23 MTUイベントリンク動作設定定義値

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6.24 CMTイベントリンク動作設定定義値

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6.25 ΔΣユニットチャネル・トリガ番号定義値

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6.26 12ビットA/Dコンバータチャネル番号定義値

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6.27 ELC割り込み番号定義値

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6.28 ポートグループ番号定義値

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6.29 出力ポートグループイベントリンク動作設定定義値

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6.30 シングルポート番号定義値

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6.31 シングルポートイベントリンク動作設定定義値

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6.32 CMTWイベントリンク動作設定定義値

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6.33 TPUチャネル番号定義値

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6.34 TPUイベントリンク動作設定定義値

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6.35 GPTチャネル番号定義値

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6.36 GPTイベントリンク動作設定定義値

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6.37 I/Oポートグループシンボル設定定義値

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6.38 入力ポートグループイベント検出トリガ設定定義値

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／立ち下がりエッジに設定
ELC_PORT_GROUP_MAX_OVER	3	正常設定値範囲確認用

表6.39 シングルポートイベント検出トリガ設定定義値

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6.40 シングルポートイベント検出トリガ設定定義値

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／立ち下がりエッジに設定
ELC_SINGLE_PORT_TRIGGER_MAX_OVER	3	正常設定値範囲確認用

6.6 構造体／共用体／列挙型一覧

図 6.2 にサンプルコードで使用する構造体／共用体／列挙型を示します。

```
typedef struct
{
    uint32_t        pclk_div;     // PCLK divide rate
    uint32_t        cnt_size;     // CMTW counter bit size
    uint32_t        clear_factor; // counter clear factor
} cmtw_time_cnt_t;

typedef struct
{
    int32_t         mode_enable;                // ON/OFF setting of compare match
    uint32_t        compare_match_cnt;          // counter value of compare match
    int32_t         intr_priority               // compare match interrupt priority
    void            (*p_callback)(void);        // callback function of compare match
} cmtw_compare_match_t;

typedef struct
{
    int32_t         mode_enable;                // ON/OFF setting of output compare
    uint32_t        output_compare_cnt;         // counter value of output compare
    uint32_t        output_signal               // signal value of output compare
    int32_t         intr_priority               // output compare interrupt priority
    void            (*p_callback)(void);        // callback function of output compare
} cmtw_output_compare_t;

typedef struct
{
    int32_t         mode_enable;                // ON/OFF setting of output compare
    uint32_t        trigger;                    // trigger of input capture
    int32_t         filter enable               // ON/OFF setting of noise filter
    int32_t         intr_priority               //input capture interrupt priority
    void            (*p_callback)(uint32_t cnt_value); // callback function of input capture
} cmtw_input_capture_t;

typedef struct
{
    int32_t             ecm_enable;             // ECM dynamic mode error enable/disable
    uint32_t            output_compare_num;     // output compare number of ECM
} cmtw_ecm_t;



typedef struct
```

```
{
    cmtw_compare_match_t   compare_match;   // setting of compare match
    cmtw_output_compare_t
                output_compare[CMTW_OUTPUT_COMPARE_NUM_MAX]; // setting of output compare
    cmtw_input_capture_t
              input_capture[CMTW_INPUT_CAPTURE_NUM_MAX];  // setting of input capture
    uint32_t  noise_filter_clk                          // noise filter setting of input capture
} cmtw_mode_t;

typedef struct
{
    cmtw_time_cnt_t   time_cnt_param;   // setting of timer count
    cmtw_mode_t       mode_param;      // setting of CMTW mode
} cmtw_cfg_t;

typedef struct
{
    uint32_t          elc_mtu_ch;           // MTU number for event link
    int32_t           event_link_enable;    // ON/OFF setting of MTU event link
    uint32_t          resource;             // resource of event link
    uint32_t          action;               // action when event link
} elc_cmd_mtu_t;

typedef struct
{
    int32_t           event_link_enable;   // ON/OFF setting of CMT event link
    uint32_t          resource;            // resource of event link
    uint32_t          action;              // action when event link
} elc_cmd_cmt_t;

typedef struct
{
    uint32_t          elc_dsmif_ch;        // DSMIF number for event link
    int32_t           event_link_enable;   // ON/OFF setting of DSMIF event link
    uint32_t          resource;            // resource of event link
} elc_cmd_dsmif_t;

typedef struct
{
    uint32_t          elc_s12ad_ch;        // S12AD number for event link
    int32_t           event_link_enable;   // ON/OFF setting of S12AD event link
    uint32_t          resource;            // resource of event link
} elc_cmd_s12ad_t;
```

```
typedef struct
{
    uint32_t            elc_intr_num;          // interrupt number for event link
    int32_t             event_link_enable;     // ON/OFF setting of MTU event link
    uint32_t            resource;              // resource of event link
    int32_t             intr_priority          // interrupt priority
    void                (*p_callback);         // callback function for interrupt
} elc_cmd_intr_t;

typedef struct
{
    uint32_t            elc_out_port_group_num;  // output port group number
    int32_t             event_link_enable;       // ON/OFF setting of output port group event link
    uint32_t            resource;                // resource of event link
    uint32_t            action;                  // action when event link
    uint8_t             init_value;              // initial output value
} elc_cmd_out_port_group_t;

typedef struct
{
    uint32_t            elc_in_port_group_num;  // input port group number
    int32_t             event_link_enable;      // ON/OFF setting of input port group event link
    uint32_t            resource;               // resource of event link
    int32_t             overwrite_enable;       // overwrite permittion for input buffer
} elc_cmd_in_port_group_t;

typedef struct
{
    uint32_t            elc_single_port_num;    // input port group number
    uint32_t            port_symbol;            // I/O port symbol information
    uint32_t            port_num;               // I/O port number for single port
    int32_t             event_link_enable;      // ON/OFF setting of single port event link
    uint32_t            event_direction;        // select event input / output
    uint32_t            resource;               // resource of event link
    uint32_t            output_action;          // action when event link for output
    uint32_t            input_trigger;          // trigger for input event link signal
} elc_cmd_single_port_t;

typedef struct
{
    int32_t             event_link_enable;      // ON/OFF setting of CMTW event link
    uint32_t            resource;               // resource of event link
    uint32_t            action;                 // action when event link
} elc_cmd_cmtw_t;
```

```
typedef struct
{
    uint32_t        elc_tpu_ch;         // TPU number for event link
    int32_t         event_link_enable;  // ON/OFF setting of TPU event link
    uint32_t        resource;           // resource of event link
    uint32_t        action;             // action when event link
} elc_cmd_tpu_t;

typedef struct
{
    uint32_t        elc_gpt_ch;         // GPT number for event link
    int32_t         event_link_enable;  // ON/OFF setting of GPT event link
    uint32_t        resource;           // resource of event link
    uint32_t        action;             // action when event link
} elc_cmd_gpt_t;

typedef struct
{
    uint32_t        port_symbol;        // I/O port symbol number
    uint8_t         port_group_bit;     // I/O port number
    uint32_t        trigger;            // trigger for signal input
} elc_cmd_port_group_t;

typedef struct
{
    uint32_t        port_group_num;     // port group number for get port value
    uint8_t         *port_value;        // port value
} elc_get_port_value_t;
```

```
typedef enum                    /* CMTW error codes */
{
        CMTW_SUCCESS                    = 0,
        CMTW_ERR_INVALID_CH             = 1,
        CMTW_ERR_INVALID_ARG            = 2,
        CMTW_ERR_NOT_OPENED             = 3,
        CMTW_ERR_NOT_CLOSED             = 4,
        CMTW_ERR_TIMER_RUNNING  = 5,
        CMTW_ERR_TIMER_STOP             = 6,
        CMTW_ERR_MISSING_PTR            = 7
} cmtw_err_t;

typedef enum                    /* ELC error codes */
{
        ELC_SUCCESS                     = 0,
        ELC_ERR_INVALID_ARG = 1,
        ELC_ERR_NOT_OPENED  = 2,
        ELC_ERR_NOT_CLOSED  = 3,
        ELC_ERR_MISSING_PTR  = 4
} elc_err_t;
```

図 6.2　サンプルコードで使用する構造体／共用体／列挙型

6.7 大域変数一覧

表 6.41 に大域変数一覧を示します。

表6.41 大域変数一覧

型	変数名	内容	使用関数
static volatile int32_t	gb_cmtw_end_flag	CMTWサンプルドライバのコールバック情報	cmtw_elc_sample_led_control 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_led_control cmtw_elc_sample_adc_callback
static const cmtw_intr_t	gb_cmtw_intr_table[][]	CMTWサンプルドライバ割り込み登録テーブル	cmtw_intr_register cmtw_intr_enable cmtw_intr_disable
static cmtw_info_ch_t	gb_cmtw_info[]	CMTWサンプルドライバ情報	R_CMTW_Open R_CMTW_Close R_CMTW_Control cmtw_init cmtw_set_count_param cmtw_set_compare_match_param cmtw_set_input_capture_param cmtw_set_output_compare_param cmtw_count_start cmtw_count_start_hw cmtw_count_stop_hw cmtw_get_driver_info
static const elc_intr_t	gb_elc_intr_table[]	ELCサンプルドライバ割り込み登録テーブル	elc_intr_register elc_intr_enable elc_intr_disable
static elcinfo_ch_t	gb_elc_info	ELCサンプルドライバ情報	R_ELC_Open R_ELC_Close R_ELC_LinkStart R_ELC_LinkStop R_ELC_Control elc_init elc_set_intr_param elc_get_driver_info

6.8 関数一覧

表 6.42 に関数一覧を示します。

表 6.42 関数一覧

関数名	ページ番号
main	38
R_CMTW_Open	30
R_CMTW_Close	31
R_CMTW_StartPeriodic	31
R_CMTW_StartOneShot	32
R_CMTW_Control	32
R_CMTW_GetVersion	33
R_ELC_Open	34
R_ELC_Close	35
R_ELC_LinkStart	35
R_ELC_LinkStop	36
R_ELC_Control	36
R_ELC_GetVersion	37
cmtw<n>_cmwi_isr	33
cmtw<n>_ic<m>i_isr	33
cmtw<n>_oc<m>i_isr	34
elc_elci<n>_isr	37
CMTW_Cmwi_Callback	38
CMTW_Ic0i_Callback	39
CMTW_Oc0i_Callback	39
ELC_Elci_Callback	39

6.9　関数仕様

サンプルコードの関数仕様を示します。

サンプルコードでは ADC ドライバの関数を使用します。ADC ドライバの関数は RZ/T1 ADC ドライバサンプル・アプリケーションノートを参照してください。

6.9.1　R_CMTW_Open

R_CMTW_Open		
概　要	CMTW ドライバの初期化関数	
ヘッダ	r_cmtw_rzt1_if.h	
宣　言	cmtw_err_t R_CMTW_Open(const uint32_t channel, const cmtw_cfg_t * const p_cfg);	
説　明	CMTW 動作開始のための初期設定を行います。 詳細は以下の通りです。 • 引数の確認 • CMTW の初期設定 - 各動作モードの設定 - コンペアマッチ周期の設定 - カウンタクリア要因の設定 - デジタルノイズフィルタ設定 • ICUA の初期設定 - コンペアマッチ割り込み、インプットキャプチャ割り込み 0/1、アウトプットコンペア割り込み 0/1 を有効にし、優先度を設定 • 消費電力低減機能設定 - MSTPCRA0 レジスタを操作し、ストップ状態を解除	
引　数	const uint32_t channel	CMTW のチャネル番号を指定
	const cmtw_cfg_t * const p_cfg	CMTW の初期設定情報ポインタ
リターン値	CMTW_SUCCESS	: 関数が正常終了 CMTW_ERR_NOT_CLOSED : 二重初期化を実行
	CMTW_ERR_INVALID_CH	: 存在しないチャネルを指定
	CMTW_ERR_INVALID_ARG	: 初期設定情報のメンバが不正な値
	CMTW_ERR_MISSING_PTR	: ポインタ引数が不正
補足	• すべての CMTW ドライバの API 関数を実行する前に本関数を実行してください。	

6.9.2 R_CMTW_Close

R_CMTW_Close		
概　要	CMTW ドライバの終了関数	
ヘッダ	r_cmtw_rzt1_if.h	
宣　言	cmtw_err_t R_CMTW_Close(const uint32_t channel);	
説　明	CMTW 動作終了のための処理を行います。 詳細は以下の通りです。 • ICUA の終了処理 - コンペアマッチ割り込み、インプットキャプチャ割り込み 0/1、アウトプットコンペア割り込み 0/1 を割り込み無効に設定 • CMTW の終了処理 - タイマカウンタの停止 • 消費電力低減機能設定 - MSTPCRA0 レジスタを操作し、ストップ状態に設定	
引　数	const uint32_t channel	CMTW のチャネル番号を指定
リターン値	CMTW_SUCCESS CMTW_ERR_INVALID_CH	：関数が正常終了 ：存在しないチャネルを指定
補足	• R_CMTW_Open の実行後に本関数を実行してください。	

6.9.3 R_CMTW_StartPeriodic

R_CmMTW_Start_Periodic		
概　要	タイマカウンタの周期動作開始関数	
ヘッダ	r_cmtw_rzt1_if.h	
宣　言	cmtw_err_t R_CMTW_StartPeriodic(const uint32_t channel);	
説　明	タイマカウントの周期動作を開始します。	
引　数	const uint32_t channel	CMTW のチャネル番号を指定
リターン値	CMTW_SUCCESS CMTW_ERR_NOT_OPENED CMTW_ERR_INVALID_CH CMTW_ERR_TIMER_RUNNING	：関数が正常終了 ：未初期化状態で実行 ：存在しないチャネルを指定 ：タイマカウンタ動作中に実行
補足	• R_CMTW_Open の実行後に本関数を実行してください。 • 周期動作を停止する場合は R_CMTW_Close() を呼び出してください。	

6.9.4 R_CMTW_StartOneShot

R_CMTW_StartOneShot		
概　要	タイマカウンタの非周期動作開始関数	
ヘッダ	r_cmtw_rzt1_if.h	
宣　言	cmtw_err_t R_CMTW_StartOneShot(const uint32_t 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	const uint32_t channel	CMTW のチャネル番号を指定
リターン値	CMTW_SUCCESS CMTW_ERR_NOT_OPENED CMTW_ERR_INVALID_CH CMTW_ERR_TIMER_RUNNING	：関数が正常終了 ：未初期化状態で実行 ：存在しないチャネルを指定 ：タイマカウンタ動作中に実行
補足	• R_CMTW_Open の実行後に本関数を実行してください。	

6.9.5 R_CMTW_Control

R_CMTW_Control		
概　要	CMTW ドライバの機能設定関数	
ヘッダ	r_cmtw_rzt1_if.h	
宣　言	cmtw_err_t R_CMTW_Control(const uint32_t channel, const uint32_t cmd, void * const p_data);	
説　明	CMTW 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。 詳細は 6.11.1 ～ 6.11.7 を参照してください。	
引　数	const uint32_t channel	CMTW のチャネル番号を指定
	const uint32_t cmd	機能設定を行うためのコマンドを指定 6.11.1 ～ 6.11.7 を参照してください
	void * const p_data	機能設定を行うためのパラメータ情報を設定 6.11.1 ～ 6.11.7 を参照してください
リターン値	CMTW_SUCCESS CMTW_ERR_NOT_OPENED CMTW_ERR_INVALID_ARG CMTW_ERR_INVALID_CH	：関数が正常終了 ：未初期化状態で実行 ：存在しないコマンドを指定 ：存在しないチャネルを指定
	上記の他に、指定するコマンド毎に別個のリターン値が返る場合があります。 詳細は 6.11.1 ～ 6.11.7 を参照してください。	
補足	• R_CMTW_Open の実行後に本関数を実行してください。	

6.9.6 R_CMTW_GetVersion

R_CMTW_GetVersion	
概　要	CMTW ドライバのバージョン情報取得関数
ヘッダ	
宣　言	uint32_t R_CMTW_GetVersion(void);
説　明	CMTW ドライバのバージョン情報を取得します。
引　数	なし
リターン値	符号化されたバージョン情報 ：0-15bit Minor Version ：16-31bit Major Version
補足	—

6.9.7 cmtw<n>_cmwi_isr

cmtw<n>_cmwi_isr	
概　要	CMTW・チャネル <n> コンペアマッチ割り込みハンドラ（n = 0、1）
ヘッダ	—
宣　言	void cmtw<n>_cmwi_isr (void);
説　明	R_CMTW_Open()、R_CMTW_Control() 関数で登録した、コンペアマッチコールバック関数を呼びだします。 また、タイマクリア要因にコンペアマッチを設定し、かつ非周期動作を行っている場合はタイマを停止し、初期化状態へ遷移します。
引　数	なし
リターン値	なし

6.9.8 cmtw<n>_ic<m>i_isr

cmtw<n>_ic<m>i_isr	
概　要	CMTW・チャネル <n> インプットキャプチャ <m> 割り込みハンドラ（n = 0、1、m = 0、1）
ヘッダ	—
宣　言	void cmtw<n>_ic<m>i_isr (void);
説　明	R_CMTW_Open()、R_CMTW_Control() 関数で登録した、インプットキャプチャコールバック関数を呼びだします。 また、タイマクリア要因にインプットキャプチャを設定し、かつ非周期動作を行っている場合はタイマを停止し、初期化状態へ遷移します。
引　数	なし
リターン値	なし

6.9.9 cmtw<n>_oc<m>i_isr

cmtw<n>_oc<m>i_isr	
概　要	CMTW・チャネル <n> アウトプットコンペア <m> 割り込みハンドラ（n = 0、1、m = 0、1）
ヘッダ	—
宣　言	void cmtw<n>_oc<m>i_isr (void);
説　明	R_CMTW_Open()、R_CMTW_Control() 関数で登録した、アウトプットコンペアコールバック関数を呼びだします。 また、タイマクリア要因にアウトプットコンペアを設定し、かつ非周期動作を行っている場合はタイマを停止し、初期化状態へ遷移します。
引　数	なし
リターン値	なし

6.9.10 R_ELC_Open

R_ELC_Open	
概　要	ELC ドライバの初期化関数
ヘッダ	r_elc_rzt1_if.h
宣　言	elc_err_t R_ELC_Open(void);
説　明	ELC 動作開始のための初期設定を行います。 詳細は以下の通りです。 • 消費電力低減機能設定 - MSTPCRC6 レジスタを操作し、ストップ状態を解除
引　数	なし
リターン値	ELC_SUCCESS　：関数が正常終了 ELC_ERR_NOT_CLOSED　：二重初期化を実行
補足	• すべての ELC ドライバの API 関数を実行する前に本関数を実行してください。

6.9.11 R_ELC_Close

R_ELC_Close	
概　要	ELC ドライバの終了関数
ヘッダ	r_elc_rzt1_if.h
宣　言	void R_ELC_Close(void);
説　明	ELC 動作終了のための処理を行います。 詳細は以下のとおりです。 • ICUA の終了処理 - 割り込み要求信号 1、2 を無効に設定 • ELC の終了処理 - ELC 機能を無効に設定 • 消費電力低減機能設定 - MSTPCRC6 レジスタを操作し、ストップ状態に設定
引　数	なし
リターン値	なし
補足	• R_ELC_Open の実行後に本関数を実行してください。

6.9.12 R_ELC_LinkStart

R_ELC_LinkStart	
概　要	ELC ドライバのイベントリンク開始関数
ヘッダ	r_elc_rzt1_if.h
宣　言	elc_err_t R_ELC_LinkStart(void);
説　明	ELC モジュールによるイベントリンク動作を開始します。
引　数	なし
リターン値	ELC_SUCCESS ：関数が正常終了 ELC_ERR_NOT_OPENED ：未初期化状態で実行
補足	• R_ELC_Open() の実行後に本関数を実行してください。 • イベントリンク実行中に6.12.1～6.12.11のコマンドによってイベントリンク設定を追加する場合、イベントリンク元となる信号が出力されないことを保障できる状態で設定を追加してください。

6.9.13 R_ELC_LinkStop

R_ELC_LinkStop		
概　要	ELC ドライバのイベントリンク終了関数	
ヘッダ	r_elc_rzt1_if.h	
宣　言	elc_err_t R_ELC_LinkStop(void);	
説　明	ELC モジュールによるイベントリンク動作を停止します。 イベントリンク動作が開始されていない状態で呼び出された場合、何もせずに関数を終了します。	
引　数	なし	
リターン値	ELC_SUCCESS ELC_ERR_NOT_OPENED	：関数が正常終了 ：未初期化状態で実行
補足	• R_ELC_Open() の実行後に本関数を実行してください。	

6.9.14 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_rzt1_if.h	
宣　言	elc_err_t R_ELC_Control(const uint32_t 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。 詳細は 6.12.1 ～ 6.12.14 を参照してください。	
引　数	const uint32_t cmd	機能設定を行うためのコマンドを指定 6.12.1 ～ 6.12.14 を参照してください
	void * const p_data	機能設定を行うためのパラメータ情報を設定 6.12.1 ～ 6.12.14 を参照してください
リターン値	ELC_SUCCESS ELC_ERR_INVALID_ARG ELC_ERR_NOT_OPENED	：関数が正常終了 ：存在しないコマンドを指定 ：未初期化状態で実行
	上記の他に、指定するコマンド毎に別個のリターン値が返る場合があります。 詳細は 6.12.1 ～ 6.12.14 を参照してください。	
補足	• R_ELC_Open() の実行後に本関数を実行してください。	

6.9.15 R_ELC_GetVersion

R_ELC_GetVersion	
概　要	ELC ドライバのバージョン情報取得関数
ヘッダ	
宣　言	uint32_t R_ELC_GetVersion(void);
説　明	ELC ドライバのバージョン情報を取得します。
引　数	なし
リターン値	符号化されたバージョン情報 ：0-15bit Minor Version ：16-31bit Major Version
補足	—

6.9.16 elc_elci<n>_isr

elc_elci<n>_isr	
概　要	ELC イベントリンク割り込みハンドラ（n = 1、2）
ヘッダ	—
宣　言	void elc_elci<n>_isr (void);
説　明	R_ELC_Open()、R_ELC_Control() 関数で登録したコールバック関数を呼びだします。
引　数	なし
リターン値	なし

6.9.17 main

main	
概　要	周期的なポテンションメータ入力電圧の 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。 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ンメータの入力電圧の A/D 変換結果を 5 段階に識別し、下記に示すようにそれぞれの段階に割り当てた評価ボード上の LED に表示します。 第 4 識別レベルを超える結果になる場合はプログラムを終了します。 A/D 変換レベル ADC_LVL0 ～ ADC_LVL1（第 1 識別レベル）: LED0 点灯 A/D 変換レベル ADC_LVL1 ～ ADC_LVL2（第 2 識別レベル）: LED1 点灯 A/D 変換レベル ADC_LVL2 ～ ADC_LVL3 （第 3 識別レベル）: LED2 点灯 A/D 変換レベル ADC_LVL3 ～ ADC_LVL4 （第 4 識別レベル）: LED3 点灯 A/D 変換レベル ADC_LVL4 以上 : プログラム終了
引　数	なし
リターン値	なし
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。

6.9.18 CMTW_Cmwi_Callback

CMTW_Cmwi_Callbak	
概　要	CMTW サンプルプログラムコンペアマッチコールバック関数
ヘッダ	—
宣　言	void CMTW_Cmwi_Callback(void);
説　明	CMTW のコンペアマッチ割り込みの発生を通知するコールバック関数です。
引　数	なし
リターン値	なし
補足	• 関数名はドライバに登録する際の例であり、名前に制限はありません。 • サンプルプログラム上では関数名 CMTW_Sample_Callback として使用しています。

6.9.19 CMTW_Ic0i_Callback

CMTW_Ic0i_Callbak	
概　要	CMTW サンプルプログラムインプットキャプチャコールバック関数
ヘッダ	—
宣　言	void CMTW_Ic0i_Callback(uint32_t cnt_value);
説　明	CMTW のインプットキャプチャ割り込みの発生を通知するコールバック関数です。 インプットキャプチャ時のカウンタ値を引数として渡します。
引　数	uint32_t cnt_value　インプットキャプチャ時のカウンタ値
リターン値	なし
補足	• 関数名はドライバに登録する際の例であり、名前に制限はありません。 • サンプルプログラムでは使用しません。

6.9.20 CMTW_Oc0i_Callback

CMTW_Oc0i_Callback	
概　要	CMTW アウトプットコンペアコールバック関数
ヘッダ	—
宣　言	void CMTW_Oc0i_Callback(void);
説　明	CMTW のアウトプットコンペア割り込みの発生を通知するコールバック関数です。
引　数	なし
リターン値	なし
補足	• 関数名はドライバに登録する際の例であり、名前に制限はありません。 • サンプルプログラムでは使用しません。

6.9.21 ELC_Elci_Callback

ELC_Elci_Callback	
概　要	ELC イベントリンクコールバック関数
ヘッダ	—
宣　言	void ELC_Elci_Callback(void);
説　明	ELC のイベントリンク割り込みの発生を通知するコールバック関数です。
引　数	なし
リターン値	なし
補足	• 関数名はドライバに登録する際の例であり、名前に制限はありません。 • サンプルプログラムでは使用しません。

6.10 フローチャート

6.10.1 サンプルプログラムメイン処理

図 6.3、図 6.4 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。

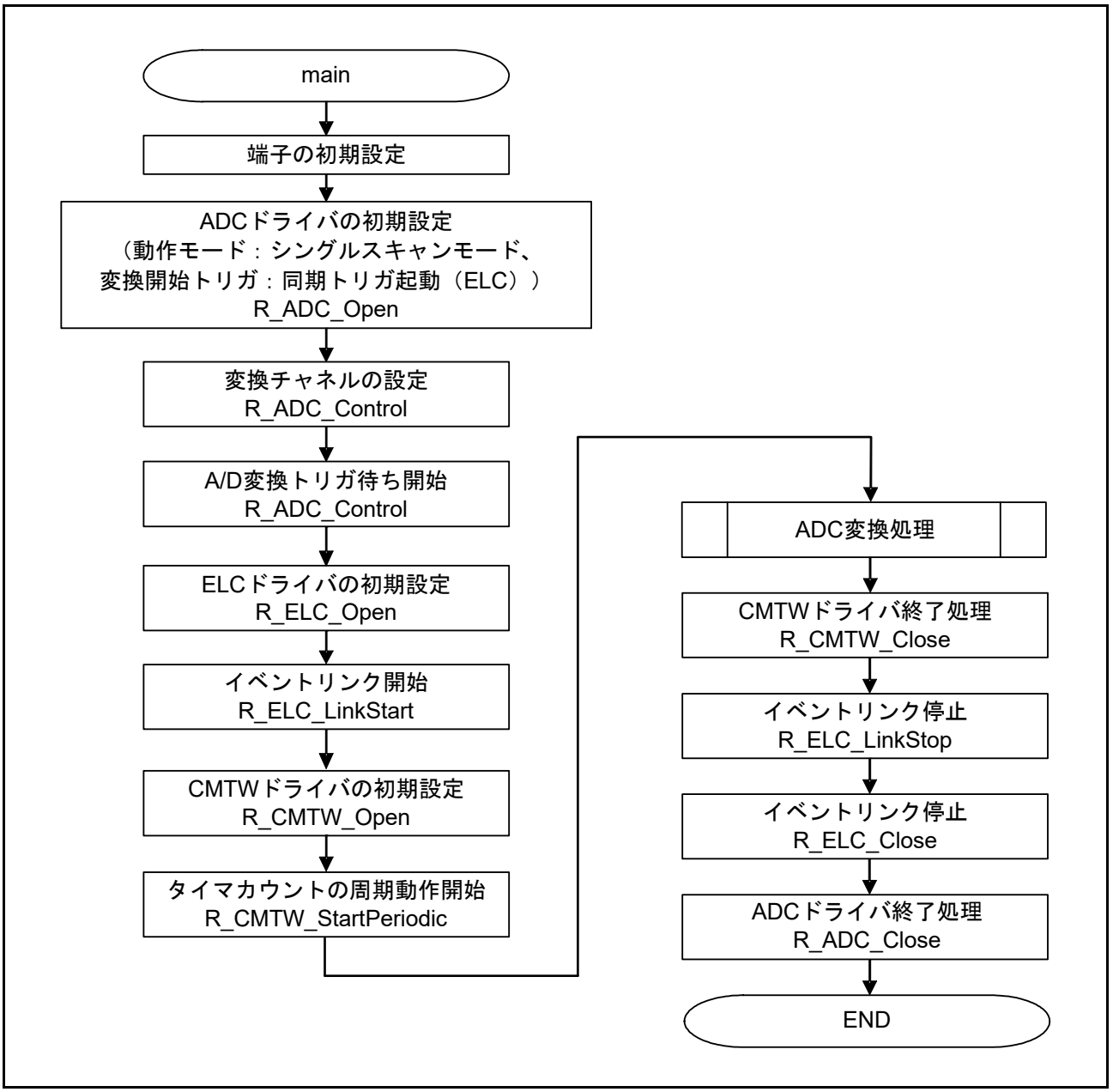


図 6.3 メイン処理

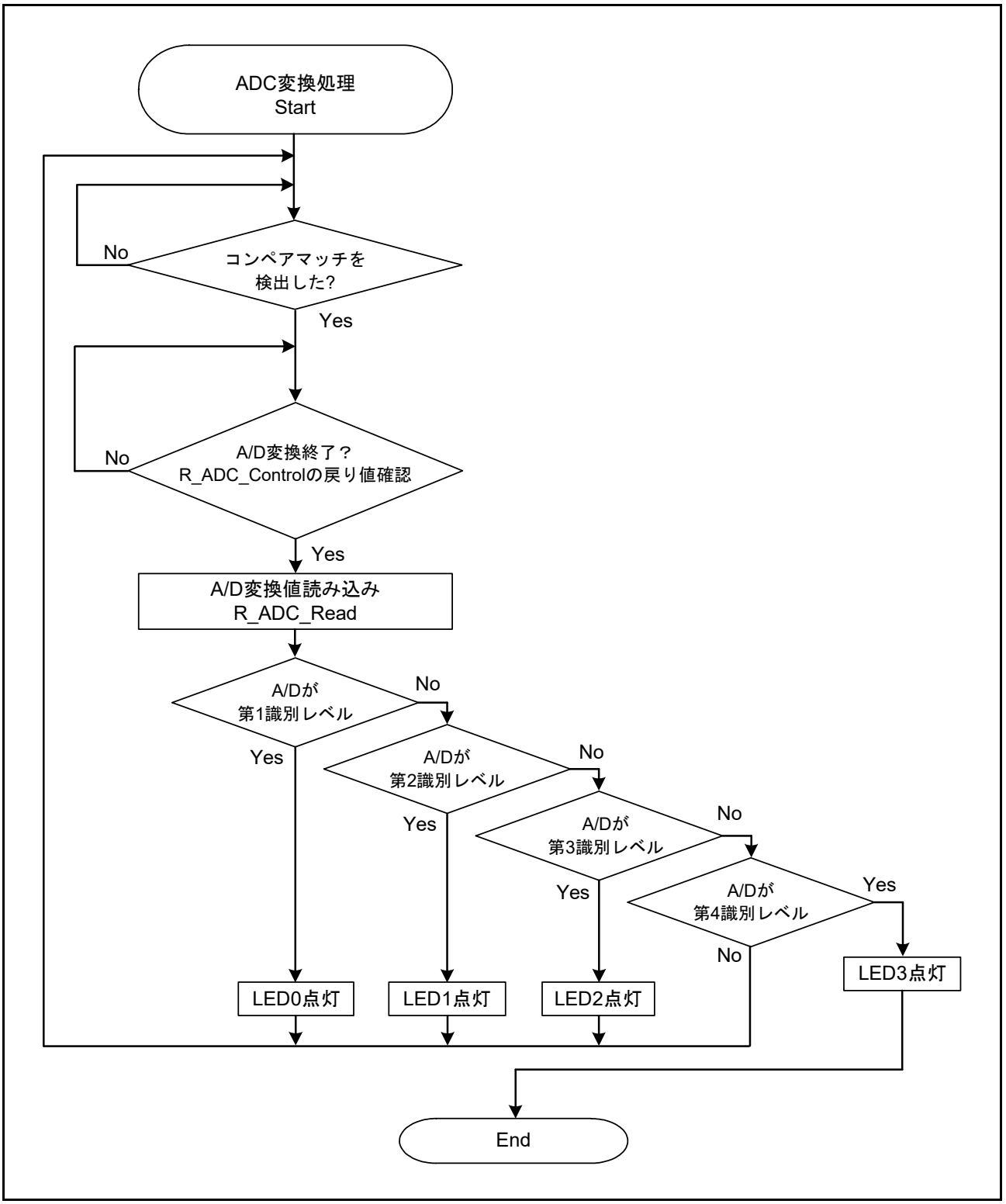

ADC変換処理
Start
No
コンペアマッチを
検出した?
Yes
No
A/D変換終了？
R_ADC_Controlの戻り値確認
Yes
A/D変換値読み込み
R_ADC_Read
A/Dが
第1識別レベル
No
Yes
A/Dが
第2識別レベル
No
Yes
A/Dが
第3識別レベル
No
Yes
A/Dが
第4識別レベル
Yes
No
LED0点灯
LED1点灯
LED2点灯
LED3点灯
End

図 6.4　ADC 変換処理

6.10.2 CMTW_Sample_Callback

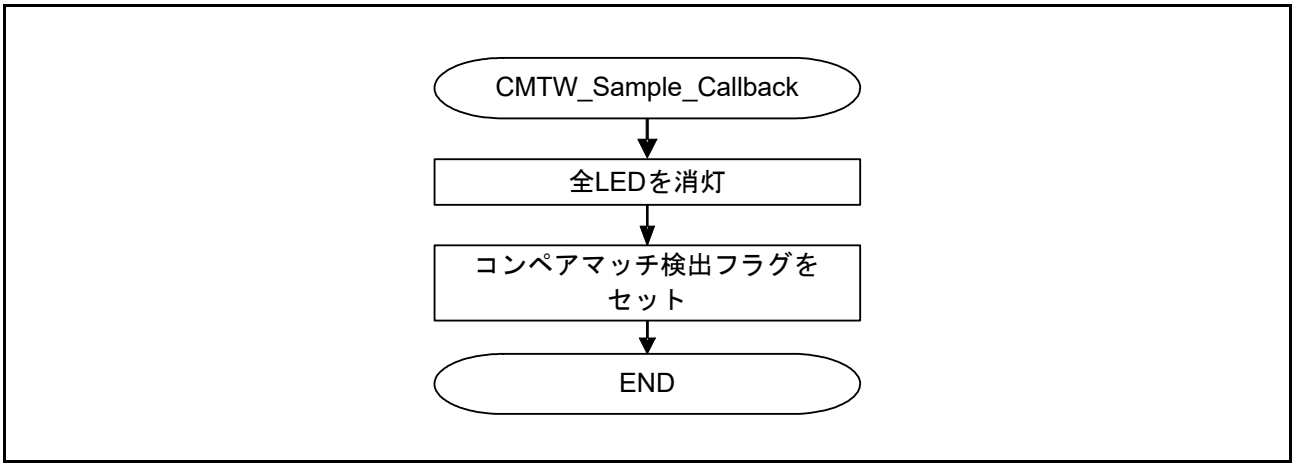


図 6.5 CMTW_Smple_Callback

6.10.3 cmtw0_cmwi_isr

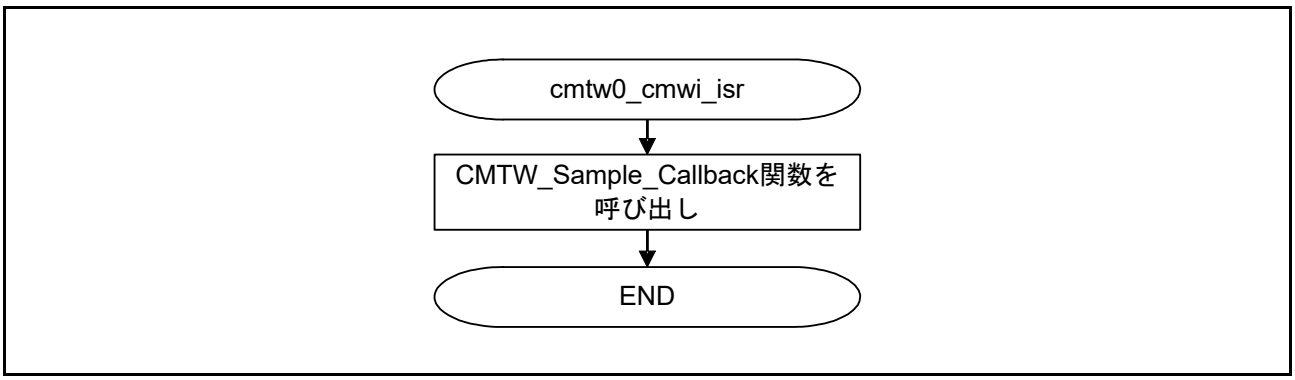


図 6.6 cmtw0_cmwi_isr

6.11 R_CMTW_Control コマンド一覧

CMTW サンプルドライバで使用するコマンド一覧を下記に示します。

表6.43 CMTWサンプルドライバで使用するコマンド一覧

定数名	内容
CMTW_CMD_SET_TIME_CNT	CMTWのタイマカウント設定を行います。
CMTW_CMD_SET_MODE	CMTWの動作モード、およびパラメータの設定を行います。
CMTW_CMD_SET_PAUSE	タイマの一時停止を行います。
CMTW_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。
CMTW_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。
CMTW_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。

6.11.1 CMTW_CMD_SET_TIME_CNT

CMTW_CMD_SET_TIME_CNT

概　要	CMTW のタイマカウント設定		
ヘッダ	r_cmtw_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	CMTW_ERR_INVALID_ARG		：タイマカウント情報のメンバが不正な値
	CMTW_ERR_TIMER_RUNNING		：タイマカウンタ動作中に実行
	CMTW_ERR_MISSING_PTR		：ポインタ引数が不正
補足	—		

6.11.2 CMTW_CMD_SET_MODE

CMTW_CMD_SET_MODE			
概　要	CMTW の動作モード設定		
ヘッダ	r_cmtw_if.h		
説　明	CMTW 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)	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。 関数の詳細は 6.9.18　CMTW_Cmwi_Callbackを参照してください。
	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)	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。 関数の詳細は 6.9.20　CMTW_Oc0i_Callback を参照してください。
	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)	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。 関数の詳細は 6.9.19　CMTW_Ic0i_Callback を参照してください。
	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	CMTW_ERR_INVALID_ARG CMTW_ERR_TIMER_RUNNING CMTW_ERR_MISSING_PTR		：タイマカウント情報のメンバが不正な値 ：タイマカウンタ動作中に実行 ：ポインタ引数が不正
補足	—		

6.11.3 CMTW_CMD_SET_PAUSE

CMTW_CMD_SET_TIME_PAUSE		
概　要	タイマー時停止	
ヘッダ	r_cmtw_if.h	
説　明	CMTW のタイマカウント一時停止を行います。 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	CMTW_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	—	

6.11.4 CMTW_CMD_SET_RESUME

CMTW_CMD_SET_TIME_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_cmtw_if.h	
説　明	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	CMTW_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
	CMTW_ERR_TIMER_RUNNING	：タイマカウンタ動作中に実行
補足	—	

6.11.5 CMTW_CMD_SET_RESTART

CMTW_CMD_SET_TIME_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_cmtw_if.h	
説　明	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	CMTW_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
	CMTW_ERR_TIMER_RUNNING	：タイマカウンタ動作中に実行
補足	—	

6.11.6　CMTW_CMD_SET_ECM

CMTW_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_cmtw_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	CMTW_ERR_INVALID_ARG CMTW_ERR_TIMER_RUNNING CMTW_ERR_MISSING_PTR		：アウトプットコンペア番号設定が不正な値 ：タイマカウンタ動作中に実行 ：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 6.11.2　CMTW_CMD_SET_MODE などで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_CMTW_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 つのみです 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		

6.11.7　CMTW_CMD_GET_STATUS

CMTW_CMD_GET_STATUS			
概　要	タイマの状態取得		
ヘッダ	r_cmtw_if.h		
説　明	タイマの動作状態を取得します。 動作状態は cmtw_status_t 型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	*p_cmtw_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	CMTW_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	—		

6.12 R_ELC_Control コマンド一覧

ELC サンプルドライバで使用するコマンド一覧を下記に示します。

表6.44 ELCサンプルドライバで使用するコマンド一覧

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。

6.12.1 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_ERR_INVALID_ARG ELC_ERR_MISSING_PTR		：イベントリンク情報のメンバが不正な値 ：ポインタ引数が不正
補足	—		

6.12.2 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_ERR_INVALID_ARG ELC_ERR_MISSING_PTR		：イベントリンク情報のメンバが不正な値 ：ポインタ引数が不正
補足	—		

6.12.3 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_ERR_INVALID_ARG ELC_ERR_MISSING_PTR		：イベントリンク情報のメンバが不正な値 ：ポインタ引数が不正
補足	—		

6.12.4 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_ERR_INVALID_ARG ELC_ERR_MISSING_PTR		：イベントリンク情報のメンバが不正な値 ：ポインタ引数が不正
補足	—		

6.12.5　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)	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。 関数の詳細は 6.9.21　ELC_Elci_Callback を参照してください。
リターン値	ELC_ERR_INVALID_ARG ELC_ERR_MISSING_PTR		：イベントリンク情報のメンバが不正な値 ：ポインタ引数が不正
補足	—		

6.12.6 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、ELC_OUT_GROUP_OUTPUT_0、ELC_OUT_GROUP_OUTPUT_1 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_ERR_INVALID_ARG ELC_ERR_MISSING_PTR		：イベントリンク情報のメンバが不正な値 ：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、本コマンドをで初期状態のパラメータを設定しなおしてください。		

6.12.7　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	入力ポートグループ 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	入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_ERR_INVALID_ARG ELC_ERR_MISSING_PTR		：イベントリンク情報のメンバが不正な値 ：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 6.12.14　ELC_CMD_GET_PORT_GROUP_VALUE を使用してください。		

6.12.8　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。
	uunt32_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	signal_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_ERR_INVALID_ARG ELC_ERR_MISSING_PTR		：イベントリンク情報のメンバが不正な値 ：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。

6.12.9 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_ERR_INVALID_ARG ELC_ERR_MISSING_PTR		：イベントリンク情報のメンバが不正な値 ：ポインタ引数が不正
補足	—		

6.12.10 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_ERR_INVALID_ARG ELC_ERR_MISSING_PTR		：イベントリンク情報のメンバが不正な値 ：ポインタ引数が不正
補足	—		

6.12.11 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_ERR_INVALID_ARG ELC_ERR_MISSING_PTR		：イベントリンク情報のメンバが不正な値 ：ポインタ引数が不正
補足	—		

6.12.12 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_ERR_INVALID_ARG ELC_ERR_MISSING_PTR		：イベントリンク情報のメンバが不正な値 ：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。		

6.12.13 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	なし
リターン値	なし
補足	—

6.12.14 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	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	*p_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_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	—		

7. 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。

8. 参考ドキュメント

- ユーザーズマニュアル：ハードウェア
  RZ/T1 グループ ユーザーズマニュアル ハードウェア編
  （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 Evaluation Board RTK7910022C00000BR ユーザーズマニュアル
  （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。）

- 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。）

- ユーザーズマニュアル：開発環境
  IAR 統合開発環境（IAR Embedded Workbench® for Arm）に関しては、IAR ホームページから入手してください。
  （最新版を IAR 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	CMTW & ELCサンプルプログラム アプリケーションノート

Rev.	発行日	改訂内容	
		ページ	ポイント
0.10	2015.03.09	—	初版発行
1.00	2015.04.10	—	Web掲載に際しRevのみ変更
1.10	2015.07.06	2. 動作環境	
		5	表2.1 動作環境　統合開発環境　表記一部修正、追加
		6. ソフトウェア説明	
		10	6.2.4 説明文 参照を追加
		10	表6.2 タイトル、サイズを一部修正
		11	表6.3 追加
		11	表6.4 追加
1.20	2015.12.03	2. 動作環境	
		5	表2.1 動作環境　統合開発環境　一部修正
1.30	2017.04.05	2. 動作環境	
		5	表2.1 動作環境　統合開発環境の内容変更
		6. ソフトウェア説明	
		—	6.2.4 必要メモリサイズ　削除
1.40	2018.06.07	2.動作環境	
		5	表2.1 動作環境　統合開発環境の内容変更
		5. ハードウェア説明	
		8	図5.1 ハードウェア構成例 モジュール名変更
		8. 参考ドキュメント	
		60	IAR 統合開発環境名変更

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. 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（予約領域）のアクセス禁止
   【注意】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、将来の機能拡張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。
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、製品型名ごとにシステム評価試験を実施してください。
   同じグループのマイコンでも型名が違うと、内部 ROM、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。

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生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品、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、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
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）、交通制御（信号）、大規模通信機器、
   金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
6. 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。
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、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。
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。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. 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。
10. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
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。
12. 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.4.0-1 2017.11)

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：https://www.renesas.com/contact/