RX ファミリ

ADC モジュール Firmware Integration Technology

R01AN1666JJ0230
Rev.2.30
2017.07.24

要旨

本アプリケーションノートは Firmware Integration Technology (FIT)を使った ADC モジュールについて説明します。

本モジュールでは、以下に示すグループにおいて、12 ビット A/D コンバータのすべての機能をサポートします: RX110、RX111、RX113、RX130、RX210、RX230、RX231、RX631、RX63N、RX64M、RX65N、RX71M。

以降、本モジュールを ADC FIT モジュールと称します。

対象デバイス

- RX110、RX111、RX113、RX130 グループ
- RX210 グループ
- RX230、RX231 グループ
- RX631、RX63N グループ
- RX64M グループ
- RX65N グループ
- RX71M グループ

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。

関連アプリケーションノート

- Firmware Integration Technology ユーザーズマニュアル(R01AN1833)
- ボードサポートパッケージモジュール Firmware Integration Technology (R01AN1685)
- e² studio に組み込む方法 Firmware Integration Technology (R01AN1723)
- CS+に組み込む方法 Firmware Integration Technology (R01AN1826)
- Renesas e² studio スマート・コンフィグレータ ユーザーガイド(R20AN0451)

目次

1. 概要

ADC FIT モジュールは、RX63x では S12ADa を、RX110/RX111/RX113/RX210 では S12ADb を、RX64M/RX71M では S12ADC を、RX130/RX230/RX231 では S12ADE を、RX65x では S12ADFa をサポートしています。

シングルスキャン、グループシングルスキャン、連続スキャンに限らず、選択した MCU に応じて機能が使用できます。周辺機能には、レジスタの左詰め、右詰め、レジスタ読み出し後のクリア、変換結果の加算、平均値の演算が含まれます。また、チャネルを交互にトリガしてデータを格納することができます。チャネル、温度センサ、内部基準電圧センサで使用される機能には、ステート数を使ったサンプリング時間の設定、A/D 変換値の加算結果の書き出しが含まれます。ボードサポートパッケージ（r_bsp モジュール）の他に本 FIT モジュールが依存しているソフトウェアはありません。

トリガを受信すると、12 ビット A/D コンバータ（S12AD）は変換を開始します。変換が完了すると、割り込み要求が発生し、許可されている場合は割り込みが発生します。S12AD がシングルスキャンモードで動作している場合、トリガごとに A/D 変換が 1 回実行されます。S12AD が連続スキャンモードで動作している場合、最初のトリガ発生後、無期限で A/D 変換が継続されます。

本 FIT モジュールでは、S12AD を初期化し、変換結果を読み出す関数を提供します。変換アラインメントや加算カウントなど、全チャネルに共通の設定は、R_ADC_Open()関数で設定します。特定チャネルの有効化は R_ADC_Control()関数を使用します。A/D 変換の結果を取得する場合、単一の変換値を取得する R_ADC_Read 関数、チャネルの有効／無効に関わらず全変換レジスタの値を取得する R_ADC_ReadAll 関数のどちらかを使用します。

ADC FIT モジュールは、RX MCU のグループに応じて、下記の 12 ビット A/D コンバータをサポートしています。

表 1.1　MCU グループに対応する 12 ビット A/D コンバータの一覧

	S12ADa	S12ADb	S12ADC	S12ADE	S12ADFa
RX110		○			
RX111		○			
RX113		○			
RX130				○	
RX210		○			
RX230				○	
RX231				○	
RX63x	○				
RX64M			○		
RX65x					○
RX71M			○		

2. API 情報

本アプリケーションノートのサンプルコードは、下記の条件で動作を確認しています。

2.1 ハードウェアの要求

ご使用になる MCU が以下の機能をサポートしている必要があります。

- S12ADa、S12ADb、S12ADC、S12ADE、S12ADFa

2.2 ハードウェアリソースの要求

ここでは、本 FIT モジュールが要求するハードウェアの周辺機能について説明します。特に記載がない場合、ここで説明するリソースは本 FIT モジュールが使用できるように、ユーザのプログラムでは使用しないでください。

2.2.1 S12ADa/S12ADb/S12ADC/S12ADE/S12ADFa

本 FIT モジュールでは、S12ADa、S12ADb、S12ADC、S12ADE、S12ADFa の全機能に対応します。

2.2.2 GPIO

本 FIT モジュールでは各アナログチャネルに対応するポート端子を使用します。

2.3 ソフトウェアの要求

本 FIT モジュールは以下のパッケージに依存しています。

- ルネサスボードサポートパッケージ（r_bsp）

2.4 制限事項

12 ビット A/D コンバータで使用するモードによって、レジスタ、設定、使用上の注意事項が異なります。本アプリケーションノートの API は、ご使用の MCU のユーザーズマニュアル　ハードウェア編の 12 ビット A/D コンバータの章に準拠してご使用ください。

RX130-512KB を使用する場合は、ルネサスボードサポートパッケージ（r_bps）の Ver.3.60 以降をご使用ください。

2.5 対応ツールチェーン

本 FIT モジュールは「6.1 動作確認環境」に示すツールチェーンで動作確認を行っています。

2.6　使用する割り込みベクタ

R_ADC_Open 関数で割り込み優先順位に 0 以外を設定した場合、割り込み発生要因に応じた割り込み（S12ADIn、S12GBADIn、GCADIn）が有効になります。

表 2.1 に本 FIT モジュールが使用する割り込みベクタを示します。

表 2.1　使用する割り込みベクター覧

デバイス	割り込みベクタ
RX110、RX111、RX113、RX130、RX210、RX230、RX231	S12ADI0 割り込み（ベクタ番号:102） GBADI 割り込み（ベクタ番号:103）
RX631、RX63N	S12ADI0 割り込み（ベクタ番号:102）
RX64M、RX71M	S12ADI0 割り込み（ベクタ番号:190）（注 1） S12ADI1 割り込み（ベクタ番号:192）（注 1） S12GBADI0 割り込み（ベクタ番号:191）（注 1） S12GBADI1 割り込み（ベクタ番号:193）（注 1） GROUPBL1 割り込み（ベクタ番号: 111） ● S12CMPI0 割り込み（グループ割り込み要因番号：20） ● S12CMPI1 割り込み（グループ割り込み要因番号：22）
RX65N	S12ADI0 割り込み（ベクタ番号:186）（注 1） S12ADI1 割り込み（ベクタ番号:189）（注 1） S12GBADI0 割り込み（ベクタ番号:187）（注 1） S12GBADI1 割り込み（ベクタ番号:190）（注 1） S12GCADI0 割り込み（ベクタ番号:188）（注 1） S12GCADI1 割り込み（ベクタ番号:191）（注 1） GROUPBL1 割り込み（ベクタ番号: 111） ● S12CMPAI 割り込み（グループ割り込み要因番号：20） ● S12CMPBI 割り込み（グループ割り込み要因番号：21） ● S12CMPAI1 割り込み（グループ割り込み要因番号：22） ● S12CMPBI1 割り込み（グループ割り込み要因番号：23）

注1.　選択型割り込み B に割り当てられている割り込みのベクタ番号については、ボードサポートパッケージ FIT モジュール(BSP モジュール)で割り当てられているデフォルト設定を記載しています。

2.7 ヘッダファイル

すべての API 呼び出しは r_s12ad_rx_if.h に記載されています。このファイルはユーザアプリケーションにインクルードする必要があります。

r_ s12ad_rx_config.h ファイルで、ビルド時に設定可能なコンフィギュレーションオプションを選択あるいは定義できます。

2.8 整数型

コードをわかりやすく、また移植が容易に行えるように、本プロジェクトでは ANSI C99 (Exact width integer types (固定幅の整数型)) を使用しています。これらの型は stdint.h で定義されています。

2.9 コンパイル時の設定

本モジュールのコンフィギュレーションオプションの設定は、r_s12ad_rx_config.h で行います。
オプション名および設定値に関する説明を、下表に示します。

コンフィギュレーションオプション（r_s12ad_rx_config.h）	
#define ADC_CFG_PARAM_CHECKING_ENABLE (1)	本定義を 1 に設定するとパラメータチェック処理のコードを生成し、0 に設定すると生成しません。 本定義を BSP_CFG_PARAM_CHECKING_ENABLE に設定するとシステムのデフォルト設定が使用されます。
// 1.8V <= AVcc0 < 2.7V #define ADC_CFG_PGA_GAIN　0 // 2.7V <= AVcc0 < 3.6V //#define ADC_CFG_PGA_GAIN　1 // 3.6V <= AVcc0 < 4.5V //#define ADC_CFG_PGA_GAIN　2 // 4.5V <= AVcc0 <= 5.5V //#define ADC_CFG_PGA_GAIN　3	本定義は RX210 用で、温度センサのゲインの増幅に関する定義です。デフォルト値には、すべての対象電圧に対して有効な“0”が設定されています。温度の分解能が最適となるように、使用する AVCC0 にあった電圧範囲を示すコメントを外してください。

2.10 コードサイズ

ツールチェーン（セクション 2.5 に記載）でのコードサイズは、最適化レベル 2、およびコードサイズ重視の最適化を前提としたサイズです。ROM (コードおよび定数) と RAM (グローバルデータ) のサイズは、本モジュールのコンフィギュレーションヘッダファイルで設定される、ビルド時のコンフィギュレーションオプションによって決まります。

ROM および RAM のコードサイズ

	パラメータチェックあり	パラメータチェックなし
RX110	ROM: 1354 バイト	ROM: 983 バイト
	RAM: 12 バイト	RAM: 12 バイト
RX111	ROM: 1234 バイト	ROM: 950 バイト
	RAM: 124 バイト	RAM: 124 バイト
RX113	ROM: 1471 バイト	ROM: 1100 バイト
	RAM: 12 バイト	RAM: 12 バイト
RX130	ROM: 2674 バイト	ROM: 2125 バイト
	RAM: 12 バイト	RAM: 12 バイト
RX210	ROM: 1671 バイト	ROM: 1200 バイト
	RAM: 12 バイト	RAM: 12 バイト
RX230, RX231	ROM: 2676 バイト	ROM: 2127 バイト
	RAM: 12 バイト	RAM: 12 バイト
RX63N	ROM: 1030 バイト	ROM: 792 バイト
	RAM: 12 バイト	RAM: 12 バイト
RX64M	ROM: 3667 バイト	ROM: 2962 バイト
	RAM: 32 バイト	RAM: 32 バイト
RX65N	ROM: 5325 バイト	ROM: 4284 バイト
	RAM: 40 バイト	RAM: 40 バイト
RX71M	ROM: 3667 バイト	ROM: 2962 バイト
	RAM: 32 バイト	RAM:32 バイト

2.11 API データ構造体

本モジュールの API で使用されるデータ構造体について説明します。

API 関数で使用するパラメータの多くは、enum 型で定義しています。これは型チェックを行い、エラーを減少させるためです。使用可能な値は、インタフェースファイルに定義されます。

2.11.1 コールバック関数のイベント定義（ 全 MCU 共通 ）

```
/* コールバック関数の引数定義 */
typedef enum e_adc_cb_evt            // コールバック関数のイベント
{
    ADC_EVT_SCAN_COMPLETE,
    ADC_EVT_SCAN_COMPLETE_GROUPB,
#if (defined(BSP_MCU_RX65_ALL))
    ADC_EVT_SCAN_COMPLETE_GROUPC,
#endif
#if (defined(BSP_MCU_RX64M) || defined(BSP_MCU_RX71M) || defined(BSP_MCU_RX65_ALL))
    ADC_EVT_CONDITION_MET
#endif
defined(BSP_MCU_RX65_ALL))
    ADC_EVT_CONDITION_METB
#endif
} adc_cb_evt_t;
```

列挙型名	説明
ADC_EVT_SCAN_COMPLETE	シングルスキャンによる A/D 変換、またはグループ A の A/D 変換完了を示します
ADC_EVT_SCAN_COMPLETE_GROUPB	グループ B の A/D 変換完了を示します
ADC_EVT_SCAN_COMPLETE_GROUPC	グループ C の A/D 変換完了を示します S12ADFa でのみ有効です
ADC_EVT_CONDITION_MET	ウィンドウ A 比較条件成立を示します。 S12ADC、S12ADFa でのみ有効です。
ADC_EVT_CONDITION_METB	ウィンドウ B 比較条件成立を示します。 S12ADFa でのみ有効です。

2.11.2　コールバック関数の引数定義（ 全 MCU 共通 ）

```
typedef struct st_adc_cb_args        // コールバックの引数
{
    adc_cb_evt_t   event;
#if (defined(BSP_MCU_RX64M) || defined(BSP_MCU_RX71M) || defined(BSP_MCU_RX65_ALL))
    uint32_t       compare_flags;
#if (defined(BSP_MCU_RX65_ALL))
    uint32_t       compare_flagsb;
#endif
    uint8_t        unit;
#endif
} adc_cb_args_t;
```

メンバ	説明
event	イベント内容を示します。
compare_flags	ウィンドウ A のチャネルごとの比較結果が格納されます。 チャネル n の比較結果がビット n に対応します。 0: 比較条件不一致 1: 比較条件一致 S12ADC、S12ADFa でのみ有効です。
compare_flagsb	ウィンドウ B のチャネルごとの比較結果が格納されます。 チャネル n の比較結果がビット n に対応します。 0: 比較条件不一致 1: 比較条件一致 S12ADFa でのみ有効です。
unit	イベントが発生したユニットを示します。 S12ADC、S12ADFa でのみ有効です。

2.11.3　S12AD 動作モード指定（S12ADb、S12ADE）

```
/* ADC_OPEN()の引数定義*/
typedef enum e_adc_mode
{
    ADC_MODE_SS_TEMPERATURE,
    ADC_MODE_SS_INT_REF_VOLT,
    ADC_MODE_SS_ONE_CH,
    ADC_MODE_SS_MULTI_CH,
    ADC_MODE_CONT_ONE_CH,
    ADC_MODE_CONT_MULTI_CH,
    ADC_MODE_SS_ONE_CH_DBLTRIG,
    ADC_MODE_SS_MULTI_CH_GROUPED,
    ADC_MODE_SS_MULTI_CH_GROUPED_DBLTRIG_A,
    ADC_MODE_MAX                    // この定義はR_ADC_Open()の引数には使用不可
} adc_mode_t;
```

列挙型名	説明
ADC_MODE_SS_TEMPERATURE	温度センサをシングルスキャンモードで A/D 変換します。 チャネルには温度センサを選択してください。
ADC_MODE_SS_INT_REF_VOLT	内部基準電圧をシングルスキャンモードで A/D 変換します。 チャネルには内部基準電圧を選択してください。
ADC_MODE_SS_ONE_CH	1 チャネルをシングルスキャンモードで A/D 変換します。 チャネルには 1 つのチャネルのみ選択してください。内部基準電圧および温度センサは選択できません。
ADC_MODE_SS_MULTI_CH	複数のチャネルをシングルスキャンモードで A/D 変換します。 内部基準電圧および温度センサは選択できません。
ADC_MODE_CONT_ONE_CH	1 チャネルを連続スキャンモードで A/D 変換します。 チャネルには 1 つのチャネルのみ選択してください。内部基準電圧および温度センサは選択できません。
ADC_MODE_CONT_MULTI_CH	複数のチャネルを連続スキャンモードで A/D 変換します。 内部基準電圧および温度センサは選択できません。
ADC_MODE_SS_ONE_CH_DBLTRIG	1 チャネルをダブルトリガモードで A/D 変換します。 チャネルには 1 つのチャネルのみ選択してください。内部基準電圧および温度センサは選択できません。
ADC_MODE_SS_MULTI_CH_GROUPED	複数チャネルをグループスキャンモードで A/D 変換します。 グループ A とグループ B には、それぞれ異なるチャネルを選択してください。内部基準電圧および温度センサは選択できません。
ADC_MODE_SS_MULTI_CH_GROUPED_DBLTRIG_A	複数チャネルをグループスキャンモードで A/D 変換します。グループ A はダブルトリガモードで動作します。 グループ A には 1 チャネルのみを、グループ B にはグループ A で選択したチャネル以外を選択してください。内部基準電圧および温度センサは選択できません。

2.11.4　S12AD 動作モード指定（S12ADa）

```
/* ADC_OPEN()の引数定義*/
typedef enum e_adc_mode
{
    ADC_MODE_SS_TEMPERATURE,
    ADC_MODE_SS_INT_REF_VOLT,
    ADC_MODE_SS_ONE_CH,
    ADC_MODE_SS_MULTI_CH,
    ADC_MODE_CONT_ONE_CH,
    ADC_MODE_CONT_MULTI_CH,
    ADC_MODE_MAX                    // この定義は R_ADC_Open()の引数には使用不可
} adc_mode_t;
```

列挙型名	説明
ADC_MODE_SS_TEMPERATURE	温度センサをシングルスキャンモードで A/D 変換します。 チャネルには温度センサを選択してください。
ADC_MODE_SS_INT_REF_VOLT	内部基準電圧をシングルスキャンモードで A/D 変換します。 チャネルには内部基準電圧を選択してください。
ADC_MODE_SS_ONE_CH	1 チャネルをシングルスキャンモードで A/D 変換します。 チャネルには 1 つのチャネルのみ選択してください。内部基準電圧および温度センサは選択できません。
ADC_MODE_SS_MULTI_CH	複数のチャネルをシングルスキャンモードで A/D 変換します。 内部基準電圧および温度センサは選択できません。
ADC_MODE_CONT_ONE_CH	1 チャネルを連続スキャンモードで A/D 変換します。 チャネルには 1 つのチャネルのみ選択してください。内部基準電圧および温度センサは選択できません。
ADC_MODE_CONT_MULTI_CH	複数のチャネルを連続スキャンモードで A/D 変換します。 内部基準電圧および温度センサは選択できません。

2.11.5　S12AD 動作モード指定（S12ADC）

```
typedef enum e_adc_mode
{
    ADC_MODE_SS_ONE_CH,
    ADC_MODE_SS_MULTI_CH,
    ADC_MODE_CONT_ONE_CH,
    ADC_MODE_CONT_MULTI_CH,
    ADC_MODE_SS_ONE_CH_DBLTRIG,
    ADC_MODE_SS_MULTI_CH_GROUPED,
    ADC_MODE_SS_MULTI_CH_GROUPED_DBLTRIG_A,
    ADC_MODE_MAX                        // この定義は R_ADC_Open()の引数には使用不可
} adc_mode_t;
```

列挙型名	説明
ADC_MODE_SS_ONE_CH	1 チャネルをシングルスキャンモードで A/D 変換します。 チャネルには 1 つのチャネルのみ選択してください。
ADC_MODE_SS_MULTI_CH	複数のチャネルをシングルスキャンモードで A/D 変換します。
ADC_MODE_CONT_ONE_CH	1 チャネルを連続スキャンモードで A/D 変換します。 チャネルには 1 つのチャネルのみ選択してください。
ADC_MODE_CONT_MULTI_CH	複数のチャネルを連続スキャンモードで A/D 変換します。
ADC_MODE_SS_ONE_CH_DBLTRIG	1 チャネルをダブルトリガモードで A/D 変換します。 チャネルには 1 つのチャネルのみ選択してください。
ADC_MODE_SS_MULTI_CH_GROUPED	複数チャネルをグループスキャンモードで A/D 変換します。 グループ A とグループ B には、異なるチャネルを選択してください。
ADC_MODE_SS_MULTI_CH_GROUPED_DBLTRIG_A	複数チャネルをグループスキャンモードで A/D 変換します。グループ A はダブルトリガモードで動作します。 グループ A には 1 チャネルのみを、グループ B にはグループ A で選択したチャネル以外を選択してください。

2.11.6　S12AD 動作モード指定（S12ADFa）

```
/* ADC_OPEN()の引数定義*/
typedef enum e_adc_mode
{
    ADC_MODE_SS_ONE_CH,
    ADC_MODE_SS_MULTI_CH,
    ADC_MODE_CONT_ONE_CH,
    ADC_MODE_CONT_MULTI_CH,
    ADC_MODE_SS_ONE_CH_DBLTRIG,
    ADC_MODE_SS_MULTI_CH_GROUPED,
    ADC_MODE_SS_MULTI_CH_GROUPED_GROUPC,
    ADC_MODE_SS_MULTI_CH_GROUPED_DBLTRIG_A,
    ADC_MODE_SS_MULTI_CH_GROUPED_DBLTRIG_A_GROUPC,
    ADC_MODE_MAX                        // この定義はR_ADC_Open()の引数には使用不可
} adc_mode_t;
```

列挙型名	説明
ADC_MODE_SS_ONE_CH	1 チャネルをシングルスキャンモードで A/D 変換します。 チャネルには 1 つのチャネルのみ選択してください。
ADC_MODE_SS_MULTI_CH	複数のチャネルをシングルスキャンモードで A/D 変換します。
ADC_MODE_CONT_ONE_CH	1 チャネルを連続スキャンモードで A/D 変換します。 チャネルには 1 つのチャネルのみ選択してください。
ADC_MODE_CONT_MULTI_CH	複数のチャネルを連続スキャンモードで A/D 変換します。
ADC_MODE_SS_ONE_CH_DBLTRIG	1 チャネルをダブルトリガモードで A/D 変換します。 チャネルには 1 つのチャネルのみ選択してください。
ADC_MODE_SS_MULTI_CH_GROUPED	複数チャネルを 2 つのグループ（グループ A、グループ B）を使用して A/D 変換します。 グループ A とグループ B には、異なるチャネルを選択してください。
ADC_MODE_SS_MULTI_CH_GROUPED_GROUPC	複数チャネルを 3 つのグループ（グループ A、グループ B、グループ C）を使用して A/D 変換します。 それぞれのグループには、異なるチャネルを選択してください。
ADC_MODE_SS_MULTI_CH_GROUPED_DBLTRIG_A	複数チャネルを 2 つのグループ（グループ A、グループ B）を使用して A/D 変換します。グループ A はダブルトリガモードで動作します。 グループ A には 1 チャネルのみを、グループ B にはグループ A で選択したチャネル以外を選択してください。
ADC_MODE_SS_MULTI_CH_GROUPED_DBLTRIG_A_GROUPC	複数チャネルを 3 つのグループ（グループ A、グループ B、グループ C）を使用して A/D 変換します。 グループ A には 1 チャネルのみを選択してください。また、それぞれのグループには異なるチャネルを選択してください。

2.11.7　S12AD 機能設定用構造体（S12ADb、ただし RX210 を除く）

```
typedef struct st_adc_cfg
{
    adc_add_t       add_cnt;
    adc_align_t     alignment;
    adc_clear_t     clearing;
    adc_speed_t     conv_speed;
    adc_trig_t      trigger;
    adc_trig_t      trigger_groupb;
    uint8_t         priority;
    uint8_t         priority_groupb;
} adc_cfg_t;
```

メンバ	説明
add_cnt	加算モードを指定します。
alignment	フォーマットを指定します。 加算モードが有効の場合、本メンバの設定は無効になります。
clearing	A/D データレジスタの自動クリアの有効／無効を指定します。
conv_speed	A/D 変換動作モード（通常動作モード／高速動作モード）を指定します。
trigger	A/D 変換の開始トリガを指定します。
trigger_groupb	グループ B の A/D 変換開始トリガを指定します。
priority	S12ADI0 割り込みの優先順位を設定します。（0～15） 0 を指定した場合、S12ADI0 割り込みは禁止されます。
priority_groupb	GBADI 割り込みの優先順位を指定します。（0～15） 0 を指定した場合、GBADI 割り込みは禁止されます。

2.11.8　S12AD 機能設定用構造体（RX210 のみ）

```
typedef struct st_adc_cfg
{
    adc_add_t       add_cnt;
    adc_align_t     alignment;
    adc_clear_t     clearing;
    adc_trig_t      trigger;
    adc_trig_t      trigger_groupb;
    uint8_t         priority;
    uint8_t         priority_groupb;
} adc_cfg_t;
```

メンバ	説明
add_cnt	加算モードを指定します。
alignment	フォーマットを指定します。 加算モードが有効の場合、本メンバの設定は無効になります。
clearing	A/D データレジスタ自動クリアの有効／無効を指定します。
trigger	A/D 変換の開始トリガを指定します。
trigger_groupb	グループ B の A/D 変換開始トリガを指定します。
priority	S12ADI0 割り込みの優先順位を設定します。（0～15） 0 を指定した場合、S12ADI0 割り込みは禁止されます。
priority_groupb	GBADI 割り込みの優先順位を指定します。（0～15） 0 を指定した場合、GBADI 割り込みは禁止されます。

2.11.9　S12AD 機能設定用構造体（S12ADE）

```
typedef struct st_adc_cfg
{
    adc_speed_t     conv_speed;
    adc_align_t     alignment;
    adc_add_t       add_cnt;
    adc_clear_t     clearing;
    adc_trig_t      trigger;
    adc_trig_t      trigger_groupb;
    uint8_t         priority;
    uint8_t         priority_groupb;
} adc_cfg_t;
```

メンバ	説明
conv_speed	A/D 変換動作モード（通常動作モード／高速動作モード）を指定します。
alignment	フォーマットを指定します。
add_cnt	加算モードを指定します。
clearing	A/D データレジスタ自動クリアの有効／無効を指定します。
trigger	A/D 変換の開始トリガを指定します。
trigger_groupb	グループ B の A/D 変換開始トリガを指定します。
priority	S12ADI0 割り込みの優先順位を設定します。（0～15） 0 を指定した場合、S12ADI0 割り込みは禁止されます。
priority_groupb	GBADI 割り込みの優先順位を指定します。（0～15） 0 を指定した場合、GBADI 割り込みは禁止されます。

2.11.10　S12AD 機能設定用構造体（S12ADa）

```
typedef struct st_adc_cfg
{
    adc_add_t       add_cnt;
    adc_align_t     alignment;
    adc_clear_t     clearing;
    adc_speed_t     conv_speed;
    adc_trig_t      trigger;
    uint8_t         priority;

} adc_cfg_t;
```

メンバ	説明
add_cnt	加算モードを指定します。
alignment	フォーマットを指定します。 加算モードが有効の場合、本メンバの設定は無効になります。
clearing	A/D データレジスタ自動クリアの有効／無効を指定します。
conv_speed	A/D 変換動作モード（通常動作モード／高速動作モード）を指定します。
trigger	A/D 変換の開始トリガを指定します。
priority	S12ADI0 割り込みの優先順位を設定します。（0～15） 0 を指定した場合、S12ADI0 割り込みは禁止されます。

2.11.11　S12AD 機能設定用構造体（S12ADC）

```
typedef struct st_adc_cfg
{
    adc_res_t       resolution;
    adc_align_t     alignment;
    adc_add_t       add_cnt;
    adc_clear_t     clearing;
    adc_trig_t      trigger;
    adc_trig_t      trigger_groupb;
    uint8_t         priority;
    uint8_t         priority_groupb;
} adc_cfg_t;
```

メンバ	説明
resolution	A/D 変換精度（8 ビット、10 ビット、12 ビット）を指定します。 分解能が低いほど、変換時間が短くなります。
alignment	フォーマットを指定します。
add_cnt	加算モードを指定します。
clearing	A/D データレジスタ自動クリアの有効／無効を指定します。
trigger	A/D 変換の開始トリガを指定します。
trigger_groupb	グループ B の A/D 変換開始トリガを指定します。
priority	S12ADI 割り込みの優先順位を設定します。（0～15） 0 を指定した場合、S12ADI 割り込みは禁止されます。
priority_groupb	S12GBADI 割り込みの優先順位を指定します。（0～15） 0 を指定した場合、S12GBADI 割り込みは禁止されます。

2.11.12　S12AD 機能設定用構造体（S12ADFa）

```
typedef struct st_adc_cfg
{
    adc_res_t       resolution;
    adc_align_t     alignment;
    adc_add_t       add_cnt;
    adc_clear_t     clearing;
    adc_trig_t      trigger;
    adc_trig_t      trigger_groupb;
    adc_trig_t      trigger_groupc;
    uint8_t         priority;
    uint8_t         priority_groupb;
    uint8_t         priority_groupc;
} adc_cfg_t;
```

メンバ	説明
resolution	A/D 変換精度（8 ビット、10 ビット、12 ビット）を指定します。 分解能が低いほど、変換速度が短くなります。
alignment	フォーマットを指定します。
add_cnt	加算モードを指定します。
clearing	A/D データレジスタ自動クリアの有効／無効を指定します。
trigger	A/D 変換の開始トリガを指定します。
trigger_groupb	グループ B の A/D 変換開始トリガを指定します。
trigger_groupc	グループ C の A/D 変換開始トリガを指定します。
priority	S12ADI 割り込みの優先順位を設定します。（0～15） 0 を指定した場合、S12ADI 割り込みは禁止されます。
priority_groupb	S12GBADI 割り込みの優先順位を指定します。（0～15） 0 を指定した場合、S12GBADI 割り込みは禁止されます。
priority_groupc	S12GCADI 割り込みの優先順位を指定します。（0～15） 0 を指定した場合、S12GCADI 割り込みは禁止されます。

2.11.13　トリガ指定

S12AD に使用できるトリガは MCU ごとに異なります。RX110 でのトリガ指定定義を以下に示します。

```
typedef enum e_adc_trig             // トリガ要因
{
    ADC_TRIG_ASYNC_ADTRG    = 0,
    ADC_TRIG_SYNC_TRG0AN    = 1,
    ADC_TRIG_SYNC_TRG0BN    = 2,
    ADC_TRIG_SYNC_TRGAN     = 3,
    ADC_TRIG_SYNC_TRG0EN    = 4,
    ADC_TRIG_SYNC_TRG0FN    = 5,
    ADC_TRIG_SOFTWARE       = 16

} adc_trig_t;
```

S12AD のトリガ一覧および使用可能 MCU を以下に示します。

列挙型名	説明	使用できる MCU
ADC_TRIG_NONE	トリガ要因非選択	RX130、RX230、RX231 RX64M、RX71M、RX65N
ADC_TRIG_SOFTWARE	ソフトウェアトリガ	すべての MCU
ADC_TRIG_ASYNC_ADTRG	外部トリガ（ADTRG#）	すべての MCU
ADC_TRIG_SYNC_TRG0AN	MTU0 TRGA	すべての MCU
ADC_TRIG_SYNC_TRG0BN	MTU0 TRGB	RX110、RX111、RX113 RX210、RX63x、RX130、 RX230、RX231
ADC_TRIG_SYNC_TRG1AN	MTU1 TRGA	RX64M、RX71M、RX65N
ADC_TRIG_SYNC_TRG2AN	MTU2 TRGA	RX64M、RX71M、RX65N
ADC_TRIG_SYNC_TRG3AN	MTU3 TRGA	RX64M、RX71M、RX65N
ADC_TRIG_SYNC_TRGAN	MTUx TRGA	RX110、RX63x
ADC_TRIG_SYNC_TRGAN_OR_UDF4N	MTUx TRGA または MTU4 アンダーフロー(相補 PWM)	RX111、RX113、RX210 RX130、RX230、RX231
ADC_TRIG_SYNC_TRG4AN_OR_UDF4N	MTU4 TRGA または MTU4 アンダーフロー(相補 PWM)	RX64M、RX71M、RX65N
ADC_TRIG_SYNC_TRG6AN	MTU6 TRGA	RX64M、RX71M、RX65N
ADC_TRIG_SYNC_TRG7AN_OR_UDF7N	MTU7 TRGA または MTU7 アンダーフロー(相補 PWM)	RX64M、RX71M、RX65N
ADC_TRIG_SYNC_TRG0EN	MTU0 TRGE	すべての MCU
ADC_TRIG_SYNC_TRG0FN	MTU0 TRGF	RX110、RX111、RX113 RX210、RX63x、RX130、 RX230、RX231
ADC_TRIG_SYNC_TRG4AN	MTU4 TADCORA	RX111、RX113、RX210 RX130、RX230、RX231 RX64M、RX71M、RX65N
ADC_TRIG_SYNC_TRG4BN	MTU4 TADCORB	RX111、RX113、RX210 RX130、RX230、RX231 RX64M、RX71M、RX65N

列挙型名	説明	使用できる MCU
ADC_TRIG_SYNC_TRG4AN_OR_TRG4BN	MTU4 TADCORA または TADCORB	RX63x、RX64M、RX71M RX65N
ADC_TRIG_SYNC_TRG4AN_AND_TRG4BN	MTU4 TADCORA かつ TADCORB	RX111、RX113、RX210 RX130、RX230、RX231 RX64M、RX71M、RX65N
ADC_TRIG_SYNC_TRG7AN	MTU7 TADCORA	RX64M、RX71M、RX65N
ADC_TRIG_SYNC_TRG7BN	MTU7 TADCORB	RX64M、RX71M、RX65N
ADC_TRIG_SYNC_TRG7AN_OR_TRG7BN	MTU7 TADCORA または TADCORB	RX64M、RX71M、RX65N
ADC_TRIG_SYNC_TRG7AN_AND_TRG7BN	MTU7 TADCORA かつ TADCORB	RX64M、RX71M、RX65N
ADC_TRIG_SYNC_GTADTR0AN	GPT0 GTADTRA	RX64M、RX71M
ADC_TRIG_SYNC_GTADTR0BN	GPT0 GTADTRB	RX64M、RX71M
ADC_TRIG_SYNC_GTADTR1AN	GPT1 GTADTRA	RX64M、RX71M
ADC_TRIG_SYNC_GTADTR1BN	GPT1 GTADTRB	RX64M、RX71M
ADC_TRIG_SYNC_GTADTR2AN	GPT2 GTADTRA	RX64M、RX71M
ADC_TRIG_SYNC_GTADTR2BN	GPT2 GTADTRB	RX64M、RX71M
ADC_TRIG_SYNC_GTADTR3AN	GPT3 GTADTRA	RX64M、RX71M
ADC_TRIG_SYNC_GTADTR3BN	GPT3 GTADTRB	RX64M、RX71M
ADC_TRIG_SYNC_GTADTR0AN_OR_GTADTR0BN	GPT0 GTADTRA または GTADTRB	RX64M、RX71M
ADC_TRIG_SYNC_GTADTR1AN_OR_GTADTR1BN	GPT1 GTADTRA または GTADTRB	RX64M、RX71M
ADC_TRIG_SYNC_GTADTR2AN_OR_GTADTR2BN	GPT2 GTADTRA または GTADTRB	RX64M、RX71M
ADC_TRIG_SYNC_GTADTR3AN_OR_GTADTR3BN	GPT3 GTADTRA または GTADTRB	RX64M、RX71M
ADC_TRIG_SYNC_TMRTRG0AN	TMR0 TCORA	RX63x、RX64M、RX71M RX65N
ADC_TRIG_SYNC_TMRTRG2AN	TMR2 TCORA	RX63x、RX64M、RX71M RX65N
ADC_TRIG_SYNC_TPUTRG0AN	TPU0 TRGA	RX210、RX230、RX231 RX63x、RX64M、RX71M RX65N
ADC_TRIG_SYNC_TPUTRGAN	TPUx TRGA	RX210、RX230、RX231 RX63x、RX64M、RX71M RX65N
ADC_TRIG_SYNC_TEMPS	温度センサ	RX210
ADC_TRIG_SYNC_ELC	ELC	RX111、RX113、RX210 RX130、RX230、RX231 RX64M、RX71M、RX65N

2.11.14　加算モード指定（S12ADa、S12ADb）

```
typedef enum e_adc_add
{
    ADC_ADD_OFF = 0,
    ADC_ADD_TWO_SAMPLES = 1,
    ADC_ADD_THREE_SAMPLES = 2,
    ADC_ADD_FOUR_SAMPLES = 3
} adc_add_t;
```

列挙型名	説明
ADC_ADD_OFF	加算モードを使用しない
ADC_ADD_TWO_SAMPLES	2 回変換（1 回加算を行う）
ADC_ADD_THREE_SAMPLES	3 回変換（2 回加算を行う）
ADC_ADD_FOUR_SAMPLES	4 回変換（3 回加算を行う）

2.11.15　加算モード指定（S12ADC）

```
typedef enum e_adc_add
{
    ADC_ADD_OFF = 0,
    ADC_ADD_TWO_SAMPLES = 1,
    ADC_ADD_THREE_SAMPLES = 2,
    ADC_ADD_FOUR_SAMPLES = 3,
    ADC_ADD_AVG_2_SAMPLES = 0x81,
    ADC_ADD_AVG_4_SAMPLES = 0x83,
} adc_add_t;
```

列挙型名	説明
ADC_ADD_OFF	加算モードを使用しない
ADC_ADD_TWO_SAMPLES	2 回変換（1 回加算を行う）
ADC_ADD_THREE_SAMPLES	3 回変換（2 回加算を行う）
ADC_ADD_FOUR_SAMPLES	4 回変換（3 回加算を行う）
ADC_ADD_AVG_2_SAMPLES	2 回変換の平均値を使用する
ADC_ADD_AVG_4_SAMPLES	4 回変換の平均値を使用する

2.11.16　加算モード指定（S12ADE、S12ADFa）

```
typedef enum e_adc_add
{
    ADC_ADD_OFF = 0,
    ADC_ADD_TWO_SAMPLES = 1,
    ADC_ADD_THREE_SAMPLES = 2,
    ADC_ADD_FOUR_SAMPLES = 3,
    ADC_ADD_SIXTEEN_SAMPLES = 5,
    ADC_ADD_AVG_2_SAMPLES = 0x81,
    ADC_ADD_AVG_4_SAMPLES = 0x83,
} adc_add_t;
```

列挙型名	説明
ADC_ADD_OFF	加算モードを使用しない
ADC_ADD_TWO_SAMPLES	2 回変換（1 回加算を行う）
ADC_ADD_THREE_SAMPLES	3 回変換（2 回加算を行う）
ADC_ADD_FOUR_SAMPLES	4 回変換（3 回加算を行う）
ADC_ADD_ SIXTEEN _SAMPLES	16 回変換（15 回加算を行う）
ADC_ADD_AVG_2_SAMPLES	2 回変換の平均値を使用する
ADC_ADD_AVG_4_SAMPLES	4 回変換の平均値を使用する

2.11.17　A/D 変換精度指定（S12ADC、S12ADFa）

```
typedef enum e_adc_res
{
    ADC_RESOLUTION_12_BIT = 0,
    ADC_RESOLUTION_10_BIT = 1,
    ADC_RESOLUTION_8_BIT = 2
} adc_res_t;
```

列挙型名	説明
ADC_RESOLUTION_12_BIT	12 ビット精度で A/D 変換を実施します
ADC_RESOLUTION_10_BIT	10 ビット精度で A/D 変換を実施します
ADC_RESOLUTION_8_BIT	8 ビット精度で A/D 変換を実施します

2.11.18　フォーマット指定（全 MCU 共通）

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

列挙型名	説明
ADC_ALIGN_RIGHT	A/D 変換結果を右詰めで格納する
ADC_ALIGN_LEFT	A/D 変換結果を左詰めで格納する

2.11.19　自動クリア指定（全 MCU 共通）

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

列挙型名	説明
ADC_CLEAR_AFTER_READ_OFF	A/D データレジスタ自動クリアしない
ADC_CLEAR_AFTER_READ_ON	A/D データレジスタ自動クリアする

2.11.20　A/D 変換動作指定（S12ADb、ただし RX210 を除く）

```
typedef enum e_adc_speed
{
    ADC_CONVERT_SPEED_DEFAULT = 0x0000,
    ADC_CONVERT_SPEED_NORM = 0x0000,
    ADC_CONVERT_SPEED_HIGH = 0x0400
} adc_speed_t;
```

列挙型名	説明
ADC_CONVERT_SPEED_DEFAULT	デフォルト設定（通常変換動作）
ADC_CONVERT_SPEED_NORM	通常変換動作
ADC_CONVERT_SPEED_HIGH	高速変換動作

2.11.21　A/D 変換動作指定（S12ADE）

```
typedef enum e_adc_speed
{
    ADC_CONVERT_SPEED_DEFAULT = 0,
    ADC_CONVERT_SPEED_HIGH    = 0,
    ADC_CONVERT_CURRENT_LOW   = 1
} adc_speed_t;
```

列挙型名	説明
ADC_CONVERT_SPEED_DEFAULT	デフォルト設定（高速変換動作）
ADC_CONVERT_SPEED_HIGH	高速変換動作
ADC_CONVERT_CURRENT_LOW	低電流変換動作

2.11.22　A/D 変換動作指定（S12ADa）

```
typedef enum e_adc_speed
{
    ADC_CONVERT_SPEED_PCLK_DIV8 = 0x00,
    ADC_CONVERT_SPEED_PCLK_DIV4 = 0x01,
    ADC_CONVERT_SPEED_PCLK_DIV2 = 0x02,
    ADC_CONVERT_SPEED_PCLK = 0x03
} adc_speed_t;
```

列挙型名	説明
ADC_CONVERT_SPEED_PCLK_DIV8	A/D 変換クロックに PCLK/8 を選択
ADC_CONVERT_SPEED_PCLK_DIV4	A/D 変換クロックに PCLK/4 を選択
ADC_CONVERT_SPEED_PCLK_DIV2	A/D 変換クロックに PCLK/2 を選択
ADC_CONVERT_SPEED_PCLK	A/D 変換クロックに PCLK を選択

2.11.23　R_ADC_Control 関数で使用するコマンド

R_ADC_Control で使用するコマンドは MCU ごとに異なります。RX110 でのトリガ指定定義を以下に示します。

```
typedef enum e_adc_cmd
{
    // ハードウェアの設定に使用するコマンド
    ADC_CMD_SET_SAMPLE_STATE_CNT,

    // チャネル、センサの有効設定に使用するコマンド
    ADC_CMD_ENABLE_CHANS,
    ADC_CMD_ENABLE_TEMP_SENSOR,
    ADC_CMD_ENABLE_VOLT_SENSOR,

    // ハードウェアトリガの有効設定、ソフトウェアトリガの発生に使用するコマンド
    ADC_CMD_ENABLE_TRIG,
    ADC_CMD_SCAN_NOW,

     // A/D 変換完了のポーリングに使用するコマンド
     ADC_CMD_CHECK_SCAN_DONE,
     ADC_CMD_CHECK_SCAN_DONE_GROUPA,
     ADC_CMD_CHECK_SCAN_DONE_GROUPB,

     // 高度な制御系コマンド
     ADC_CMD_DISABLE_TRIG,
     ADC_CMD_DISABLE_INT,
     ADC_CMD_ENABLE_INT,
     ADC_CMD_DISABLE_INT_GROUPB,
     ADC_CMD_ENABLE_INT_GROUPB
} adc_cmd_t;
```

使用するコマンドにより、R_ADC_Control 関数の第 3 引数(p_args)に指定する内容が異なります。コマンド一覧および使用可能 MCU を以下に示します。なお、パラメータを使用しないコマンドに関しては、R_ADC_Control 関数の第 3 引数に FIT_NO_PTR を指定してください。

列挙型名	説明	使用できる MCU
ADC_CMD_USE_INT_VOLT_AS_HVREF	高電位側基準電圧に内部基準電圧を使用する パラメータは使用しません。	RX113
ADC_CMD_USE_VREFL0	低電位側基準電圧に VREFL0 を使用する。 パラメータは使用しません。	RX130、RX230、RX231
ADC_CMD_USE_VREFH0	高電位側基準電圧に VREFH0 を使用する。 パラメータは使用しません。	RX130、RX230、RX231

列挙型名	説明	使用できる MCU
ADC_CMD_SET_DDA_STATE_CNT	A/D 断線検出アシスト機能の設定を行います。 パラメータには断線検出設定用構造体(adc_dda_t)を指定してください。	RX130、RX210、RX230 RX231、RX64M、RX65N RX71M
ADC_CMD_SET_SAMPLE_STATE_CNT	A/D サンプリングステートを変更します。 パラメータにはサンプリングステート設定用構造体(adc_time_t、または adc_sst_t)を指定してください。	すべての MCU
ADC_CMD_ENABLE_CHANS	A/D 変換するチャネルの設定を行います。 パラメータには変換チャネル設定用構造体(adc_ch_cfg_t)を指定してください。	すべての MCU
ADC_CMD_ENABLE_TEMP_SENSOR	温度センサを有効にします。 パラメータは使用しません。	RX110、RX111、RX113 RX130、RX210、RX230 RX231、RX63N
ADC_CMD_ENABLE_VOLT_SENSOR	内部基準電圧センサを有効にします。 パラメータは使用しません。	RX110、RX111、RX113 RX130、RX210、RX230 RX231、RX63N
ADC_CMD_EN_COMPARATOR_LEVEL	コンペア機能をウィンドウ機能無効（しきい値比較）で使用します。 パラメータにはコンペア機能設定用構造体（adc_cmpwin_t）を指定してください。	RX130、RX230、RX231 RX64M、RX65N、RX71M
ADC_CMD_EN_COMPARATOR_WINDOW	コンペア機能をウィンドウ機能有効（範囲比較）で使用します。 パラメータにはコンペア機能設定用構造体（adc_cmpwin_t）を指定してください。	RX130、RX230、RX231 RX64M、RX65N、RX71M
ADC_CMD_COMP_COMB_STATUS	ウィンドウ A/B の複合条件結果を取得します。 パラメータには、組み合わせ結果モニタ（adc_comp_stat_t）変数へのポインタを指定してください。	RX65N
ADC_CMD_ENABLE_TRIG	同期、非同期トリガによる A/D 変換の開始を許可にします。 パラメータは使用しません。	すべての MCU
ADC_CMD_SCAN_NOW	ソフトウェアトリガによる A/D 変換を開始します。 パラメータは使用しません。	すべての MCU
ADC_CMD_CHECK_SCAN_DONE	シングルスキャンモードにて、A/D 変換中かを確認します。 パラメータは使用しません。	すべての MCU

列挙型名	説明	使用できる MCU
ADC_CMD_CHECK_SCAN_DONE_GROUPA	グループスキャンモードにて、グループ A が A/D 変換中かを確認します。 パラメータは使用しません。	RX110、RX111、RX113 RX130、RX210、RX230 RX231、RX64M、RX65N RX71M
ADC_CMD_CHECK_SCAN_DONE_GROUPB	グループスキャンモードにて、グループ B が A/D 変換中かを確認します。 パラメータは使用しません。	RX110、RX111、RX113 RX130、RX210、RX230 RX231、RX64M、RX65N RX71M
ADC_CMD_CHECK_SCAN_DONE_GROUPC	グループスキャンモードにて、グループ C が A/D 変換中かを確認します。 パラメータは使用しません。	RX65N
ADC_CMD_CHECK_CONDITION_MET	コンペア機能による比較結果を取得します。（注 1） パラメータには、比較結果を格納する uint32_t 型変数へのポインタを指定してください。	RX130、RX230、RX231 RX64M、RX65N、RX71M
ADC_CMD_CHECK_CONDITION_METB	グループ B のコンペア機能による比較結果を取得します。（注 1） パラメータには、比較結果を格納する uint32_t 型変数へのポインタを指定してください。	RX65N
ADC_CMD_DISABLE_TRIG	同期、非同期トリガによる A/D 変換の開始を無効にします。 パラメータは使用しません。	すべての MCU
ADC_CMD_DISABLE_INT	S12ADI 割り込みを禁止にします。 パラメータは使用しません。	すべての MCU
ADC_CMD_ENABLE_INT	S12ADI 割り込みを許可にします。 パラメータは使用しません。	すべての MCU
ADC_CMD_DISABLE_INT_GROUPB	GBADI 割り込みを禁止にします。 パラメータは使用しません。	RX110、RX111、RX113 RX130、RX210、RX230 RX231、RX64M、RX65N RX71M
ADC_CMD_ENABLE_INT_GROUPB	GBADI/S12GBADI 割り込みを許可にします。 パラメータは使用しません。	RX110、RX111、RX113 RX130、RX210、RX230 RX231、RX64M、RX65N RX71M
ADC_CMD_DISABLE_INT_GROUPC	S12GCADI 割り込みを禁止にします。 パラメータは使用しません。	RX65N
ADC_CMD_ENABLE_INT_GROUPC	S12GCADI 割り込みを許可にします。 パラメータは使用しません。	RX65N

注1. 本コマンド実行後、比較結果を“0”（比較条件不成立）に初期化します。そのため、A/D 変換完了ごとに、本コマンドを 1 度だけ実行してください。

2.11.24　断線検出設定用構造体（S12ADC、S12ADE、S12ADFa）

```
typedef struct st_adc_dda
{
    adc_charge_t    method;
    uint8_t         num_states;
} adc_dda_t;
```

メンバ	説明
method	断線検出アシストの設定（ディスチャージ／プリチャージ）を設定します。
num_states	ディスチャージ／プリチャージ期間を設定します。 0、または 2～15 の範囲で設定してください。 0 を設定すると、断線検出アシスト機能は無効となります。

2.11.25　断線検出設定用構造体（RX210）

```
typedef struct st_adc_dda
{
    adc_charge_t    method;
    uint8_t         num_states;
} adc_dda_t;
```

メンバ	説明
method	断線検出アシストの設定（ディスチャージ／プリチャージ）を設定します。
num_states	ディスチャージ／プリチャージ期間を設定します。 0～15 の範囲で設定してください。 0 を設定すると、断線検出アシスト機能は無効となります。

2.11.26　サンプリングステート設定用構造体（S12ADb、ただし RX210 は除く）

```
typedef struct st_adc_time
{
    adc_sst_reg_t   reg_id;
    uint8_t         num_states;
} adc_time_t;
```

メンバ	説明
reg_id	サンプリングステートを設定するチャネルを選択します。
num_states	サンプリング時間を設定します。 6～255 の範囲で設定してください。

2.11.27　サンプリングステート設定用構造体（RX210）

```
typedef struct st_adc_time
{
    adc_sst_reg_t   reg_id;
    uint8_t         num_states;
} adc_time_t;
```

メンバ	説明
reg_id	サンプリングステートを設定するチャネルを選択します。
num_states	サンプリング時間を設定します。 12～255 の範囲で設定してください。

2.11.28　サンプリングステート設定用構造体（S12ADa）

```
typedef struct st_adc_time
{
    adc_sst_reg_t   reg_id;
    uint8_t         num_states;
} adc_time_t;
```

メンバ	説明
reg_id	サンプリングステートを設定するチャネルを選択します。
num_states	サンプリング時間を設定します。 10～255 の範囲で設定してください。

2.11.29　サンプリングステート設定用構造体（S12ADC、S12ADE、S12ADFa）

```
typedef struct st_adc_time
{
    adc_sst_reg_t   reg_id;
    uint8_t         num_states;
} adc_sst_t;
```

メンバ	説明
reg_id	サンプリングステートを設定するチャネルを選択します。
num_states	サンプリング時間を設定します。 5～255 の範囲で設定してください。（注 1）

注1.　S12ADE で PCLK : ADCLK 周波数比= 1 : 2、1 : 4 の場合、6 ステート以上の値を設定してください。

2.11.30　変換チャネル設定用構造体（S12ADb、ただし RX210 は除く）

```
typedef struct st_adc_ch_cfg            // bit 0 = ch0; bit 15 = ch15
{
    uint32_t        chan_mask;
    uint32_t        chan_mask_groupb;
    uint32_t        add_mask;
} adc_ch_cfg_t;
```

メンバ	説明
chan_mask(注 1)	使用するチャネルを選択します。 選択例）チャネル 1 とチャネル 3 を選択する場合 (ADC_MASK_CH1 | ADC_MASK_CH3)
chan_mask_groupb(注 1)	グループ B で使用するチャネルを選択します。 グループ B を使用しない場合は ADC_MASK_GROUPB_OFF を指定してください。 選択例）チャネル 4 とチャネル 5 を選択する場合 (ADC_MASK_CH4 | ADC_ MASK_CH5)
add_mask(注 1)	加算モードを行うチャネルを選択します。 加算モードを使用しない場合は、ADC_MASK_ADD_OFF を指定してください。 加算モードを使用する場合は、chan_mask で選択したチャネルから選択してください。

注1.　チャネルの指定は ADC_MASK_CHn (n はチャネル番号)を使用してください。

2.11.31　変換チャネル設定用構造体（RX210）

```
typedef struct st_adc_ch_cfg
{
    uint32_t       chan_mask;
    uint32_t       chan_mask_groupb;
    uint32_t       add_mask;
    adc_diag_t     diag_method;
    uint8_t        sample_hold_mask;
    uint8_t        sample_hold_states;
} adc_ch_cfg_t;
```

メンバ	説明
chan_mask(注 1)	使用するチャネルを選択します。 選択例）チャネル 1 とチャネル 3 を選択する場合 (ADC_MASK_CH1 | ADC_ MASK_CH3)
chan_mask_groupb(注 1)	グループ B で使用するチャネルを選択します。 グループ B を使用しない場合は、ADC_MASK_GROUPB_OFF を指定してください。 選択例）チャネル 4 とチャネル 5 を選択する場合 (ADC_MASK_CH4 | ADC_ MASK_CH5)
add_mask(注 1)	加算モードを行うチャネルを選択します。 加算モードを使用しない場合は、ADC_MASK_ADD_OFF を指定してください。 加算モードを使用する場合は、chan_mask で選択したチャネルから選択してください。
diag_method	自己診断モードの設定を行います。
sample_hold_mask	サンプル&ホールド回路を使用するチャネルを指定します。 ビット 2～0 がチャネル 2～0 に対応しています。 0: チャネル専用サンプル&ホールド回路をバイパス 1: チャネル専用サンプル&ホールド回路を使用
sample_hold_states	サンプリング時間を設定します。 4～255 の範囲で設定してください。

注1.　チャネルの指定は ADC_MASK_CHn (n はチャネル番号)の組み合わせを使用してください。

2.11.32　変換チャネル設定用構造体（S12ADE）

```
typedef struct st_adc_ch_cfg
{
    uint32_t        chan_mask;
    uint32_t        chan_mask_groupb;
    adc_grpa_t      priority_groupa;
    uint32_t        add_mask;
    adc_diag_t      diag_method;
    adc_elc_t       signal_elc;
} adc_ch_cfg_t;
```

メンバ	説明
chan_mask(注 1)	使用するチャネルを選択します。 選択例）チャネル 1 とチャネル 3 を選択する場合 (ADC_MASK_CH1 | ADC_ MASK_CH3)
chan_mask_groupb(注 1)	グループ B で使用するチャネルを選択します。 グループ B を使用しない場合は、ADC_MASK_GROUPB_OFF を指定してください。 選択例）チャネル 4 とチャネル 5 を選択する場合 (ADC_MASK_CH4 | ADC_ MASK_CH5)
priority_groupa	グループ A 優先制御動作の設定を行います。
add_mask(注 1)	加算モードを行うチャネルを選択します。 加算モードを使用しない場合は、ADC_MASK_ADD_OFF を指定してください。 加算モードを使用する場合は、chan_mask で選択したチャネルから選択してください。
diag_method	自己診断モードの設定を行います。
signal_elc	ELC 用スキャン終了イベントのイベント発生条件を設定します。

注1.　チャネルの指定は ADC_MASK_CHn (n はチャネル番号)を使用してください。
温度センサをシングルスキャンモード（ADC_MODE_SS_TEMPERATURE）で使用している場合、ADC_MASK_TEMP を指定してください。
同様に内部基準電圧センサをシングルスキャンモード（ADC_MODE_SS_INT_REF_VOLT）で使用している場合、ADC_MASK_VOLT を指定してください。

2.11.33　変換チャネル設定用構造体（S12ADa）

```
typedef struct st_adc_ch_cfg
{
    uint32_t        chan_mask;
    uint32_t        add_mask;
} adc_ch_cfg_t;
```

メンバ	説明
chan_mask(注 1)	使用するチャネルを選択します。 選択例）チャネル 1 とチャネル 3 を選択する場合 (ADC_MASK_CH1 | ADC_ MASK_CH3)
add_mask(注 1)	加算モードを行うチャネルを選択します。 加算モードを使用しない場合は、ADC_MASK_ADD_OFF を指定してください。 加算モードを使用する場合は、chan_mask で選択したチャネルから選択してください。

注1.　チャネルの指定は ADC_MASK_CHn (n はチャネル番号)を使用してください。

2.11.34　変換チャネル設定用構造体（S12ADC）

```
typedef struct st_adc_ch_cfg
{
    uint32_t       scan_mask;
    uint32_t       scan_mask_groupb;
    adc_grpa_t     priority_groupa;
    uint32_t       add_mask;
    adc_diag_t     diag_method;
    bool           anex_enable;
    uint8_t        sample_hold_mask;
    uint8_t        sample_hold_states;
} adc_ch_cfg_t;
```

メンバ	説明
scan_mask(注 1)	使用するチャネルを選択します。 選択例）チャネル 1 とチャネル 3 を選択する場合 (ADC_MASK_CH1 | ADC_ MASK_CH3)
scan_mask_groupb(注 1)	グループ B で使用するチャネルを選択します。 グループ B を使用しない場合は、ADC_MASK_GROUPB_OFF を指定してください。 選択例）チャネル 4 とチャネル 5 を選択する場合 (ADC_MASK_CH4 | ADC_ MASK_CH5)
priority_groupa	グループ A 優先制御動作の設定を行います。
add_mask(注 1)	加算モードを行うチャネルを選択します。 加算モードを使用しない場合は、ADC_MASK_ADD_OFF を指定してください。 加算モードを使用する場合は、scan_mask で選択したチャネルから選択してください。
diag_method	自己診断モードの設定を行います。
anex_enable	拡張アナログ入力(ANEX1)の使用有無を指定します。
sample_hold_mask(注 1)	サンプル&ホールド回路を使用するチャネルを指定します。 チャネル専用サンプル&ホールド回路を使用するチャネルを 0～2 から選択してください。
sample_hold_states	サンプリング時間を設定します。 4～255 の範囲で設定してください。

注1.　チャネルの指定は ADC_MASK_CHn (n はチャネル番号)、ADC_MASK_TEMP（温度センサ）、ADC_MASK_VOLT（内部基準電圧センサ）のいずれか、もしくは組み合わせで指定してください。

2.11.35　変換チャネル設定用構造体（S12ADFa）

```
typedef struct st_adc_ch_cfg
{
    uint32_t        scan_mask;
    uint32_t        scan_mask_groupb;
    uint32_t        scan_mask_groupc;
    adc_grpa_t      priority_groupa;
    uint32_t        add_mask;
    adc_diag_t      diag_method;
    bool           anex_enable;
    uint8_t         sample_hold_mask;
    uint8_t         sample_hold_states;
} adc_ch_cfg_t;
```

メンバ	説明
scan_mask(注 1)	使用するチャネルを選択します。 選択例）チャネル 1 とチャネル 3 を選択する場合 (ADC_MASK_CH1 | ADC_ MASK_CH3)
scan_mask_groupb(注 1)	グループ B で使用するチャネルを選択します。 グループ B を使用しない場合は、ADC_MASK_GROUPB_OFF を指定してください。 選択例）チャネル 4 とチャネル 5 を選択する場合 (ADC_MASK_CH4 | ADC_ MASK_CH5)
scan_mask_groupc(注 1)	グループ C で使用するチャネルを選択します。 グループ C を使用しない場合は、ADC_MASK_GROUPC_OFF を指定してください。 選択例）チャネル 8 とチャネル 9 を選択する場合 (ADC_MASK_CH8 | ADC_ MASK_CH9)
priority_groupa	グループ優先制御動作の設定を行います。
add_mask	加算モードを行うチャネルを選択します。 加算モードを使用しない場合は、ADC_MASK_ADD_OFF を指定してください。 加算モードを使用する場合は、scan_mask で選択したチャネルから選択してください。
diag_method	自己診断モードの設定を行います。
anex_enable	拡張アナログ入力(ANEX1)の使用有無を指定します。
sample_hold_mask(注 1)	サンプル&ホールド回路を使用するチャネルを指定します。 チャネル専用サンプル&ホールド回路を使用するチャネルを 0～2 から選択してください。
sample_hold_states	サンプリング時間を設定します。 4～255 の範囲で設定してください。

注1.　チャネルの指定は ADC_MASK_CHn (n はチャネル番号)、ADC_MASK_TEMP（温度センサ）、ADC_MASK_VOLT（内部基準電圧センサ）のいずれか、もしくは組み合わせで指定してください。

2.11.36　コンペア機能設定用構造体（S12ADE）

```
typedef struct st_adc_cmpwin_cfg
{
    uint32_t       compare_mask;
    uint32_t       inside_window_mask;
    uint16_t       level_lo;
    uint16_t       level_hi;
    bool           windowa_enable;
} adc_cmpwin_t;
```

メンバ	説明
compare_mask(注 1)	コンペア機能を使用するチャネルを選択します。 選択例）チャネル 1 とチャネル 3 を選択する場合 (ADC_MASK_CH1 | ADC_ MASK_CH3)
inside_window_mask	各チャネルに対するコンペア条件を選択します。 ビット n がチャネル n に対応します。 ・ウィンドウ機能無効時の場合 （ADC_CMD_EN_COMPARATOR_LEVEL コマンド） 0: level_lo > A/D 変換値のとき合致 1: level_lo < A/D 変換値のとき合致 ・ウィンドウ機能有効時の場合 （ADC_CMD_EN_COMPARATOR_WINDOW コマンド） 0: A/D 変換値 < level_lo または level_hi < A/D 変換値のとき合致 1: level_lo < A/D 変換値 < level_hi のとき合致
level_lo(注 2)	コンペアウィンドウ A の下位側レベルを設定します。
level_hi(注 2)	コンペアウィンドウ A の上位側レベルを設定します。 ADC_CMD_EN_COMPARATOR_WINDOW コマンド使用時のみ有効です。
windowa_enable	コンペアウィンドウ A 機能の有効／無効を選択します。

注1.　チャネルの指定は ADC_MASK_CHn (n はチャネル番号)を使用してください。
温度センサをシングルスキャンモード（ADC_MODE_SS_TEMPERATURE）で使用している場合、ADC_MASK_TEMP を指定してください。
同様に内部基準電圧センサをシングルスキャンモード（ADC_MODE_SS_INT_REF_VOLT）で使用している場合、ADC_MASK_VOLT を指定してください。

注2.　A/D データレジスタのフォーマット選択（右詰め／左詰め）や A/D 変換値加算モードの設定により設定内容が異なります。詳細はユーザーズマニュアル ハードウェア編を参照してください。

2.11.37　コンペア機能設定用構造体（S12ADC）

```
typedef struct st_adc_cmpwin_cfg
{
    uint32_t       compare_mask;
    uint32_t       inside_window_mask;
    uint16_t       level_lo;
    uint16_t       level_hi;
    uint8_t        int_priority;
} adc_cmpwin_t;
```

メンバ	説明
compare_mask(注 1)	コンペア機能を使用するチャネルを選択します。 選択例）チャネル 1 とチャネル 3 を選択する場合 (ADC_MASK_CH1 | ADC_ MASK_CH3)
inside_window_mask	各チャネルに対するコンペア条件を選択します。 ビット n がチャネル n に対応します。 ・ウィンドウ機能無効時の場合 （ADC_CMD_EN_COMPARATOR_LEVEL コマンド） 0: level_lo > A/D 変換値のとき合致 1: level_lo < A/D 変換値のとき合致 ・ウィンドウ機能有効時の場合 （ADC_CMD_EN_COMPARATOR_WINDOW コマンド） 0: A/D 変換値 < level_lo または level_hi < A/D 変換値のとき合致 1: level_lo < A/D 変換値 < level_hi のとき合致
level_lo(注 2)	コンペアウィンドウ A の下位側レベルを設定します。
level_hi(注 2)	コンペアウィンドウ A の上位側レベルを設定します。 ADC_CMD_EN_COMPARATOR_WINDOW コマンド使用時のみ有効です。
int_priority	S12CMPI 割り込みの優先順位を設定します。（0～15） 0 を指定した場合、S12CMPI 割り込みは禁止されます。

注1.　チャネルの指定は ADC_MASK_CHn (n はチャネル番号)、ADC_MASK_TEMP（温度センサ）、ADC_MASK_VOLT（内部基準電圧センサ）のいずれか、もしくは組み合わせで指定してください。

注2.　A/D データレジスタのフォーマット選択（右詰め／左詰め）や A/D 変換精度、A/D 変換値加算モードの設定により設定内容が異なります。詳細はユーザーズマニュアル ハードウェア編を参照してください。

2.11.38　コンペア機能設定用構造体（S12ADFa）

```
typedef struct st_adc_cmpwin_cfg
{
    uint32_t       compare_mask;
    uint32_t       compare_maskb;
    uint32_t       inside_window_mask;
    uint32_t       inside_window_maskb;
    uint16_t       level_lo;
    uint16_t       level_lob;
    uint16_t       level_hi;
    uint16_t       level_hib;
    adc_comp_cond_t comp_cond;
    uint8_t        int_priority;
    bool           windowa_enable;
    bool           windowb_enable;
} adc_cmpwin_t;
```

メンバ	説明
compare_mask(注 1)	ウィンドウ A のコンペア機能を使用するチャネルを選択します。 選択例）チャネル 1 とチャネル 3 を選択する場合 (ADC_MASK_CH1 | ADC_ MASK_CH3)
compare_maskb(注 2)	ウィンドウ B のコンペア機能を使用するチャネルを選択します。 選択例）チャネル 4 を選択する場合 ADC_COMP_WINB_CH4
inside_window_mask	ウィンドウ A の各チャネルに対するコンペア条件を選択します。 ビット n がチャネル n に対応します。 ・ウィンドウ機能無効時の場合 （ADC_CMD_EN_COMPARATOR_LEVEL コマンド） 0: level_lo > A/D 変換値のとき合致 1: level_lo < A/D 変換値のとき合致 ・ウィンドウ機能有効時の場合 （ADC_CMD_EN_COMPARATOR_WINDOW コマンド） 0: A/D 変換値 < level_lo または level_hi < A/D 変換値のとき合致 1: level_lo < A/D 変換値 < level_hi のとき合致
inside_window_maskb	ウィンドウ B のコンペア条件を選択します。 ・ウィンドウ機能無効時の場合 （ADC_CMD_EN_COMPARATOR_LEVEL コマンド） ADC_COMP_WINB_COND_BELOW： level_lo > A/D 変換値のとき合致 ADC_COMP_WINB_COND_ABOVE： level_lo < A/D 変換値のとき合致 ・ウィンドウ機能有効時の場合 （ADC_CMD_EN_COMPARATOR_WINDOW コマンド） ADC_COMP_WINB_COND_BELOW： A/D 変換値 < level_lo または level_hi < A/D 変換値のとき合致 ADC_COMP_WINB_COND_ABOVE： level_lo < A/D 変換値 < level_hi のとき合致

メンバ	説明
level_lo(注 3)	コンペアウィンドウ A の下位側レベルを設定します。
level_lob(注 3)	コンペアウィンドウ B の下位側レベルを設定します。 ADC_CMD_EN_COMPARATOR_WINDOW コマンド使用時のみ有効です。
level_hi(注 3)	コンペアウィンドウ A の上位側レベルを設定します。
level_hib(注 3)	コンペアウィンドウ B の上位側レベルを設定します。 ADC_CMD_EN_COMPARATOR_WINDOW コマンド使用時のみ有効です。
comp_cond	ウィンドウ A/B 複合条件を設定します。
int_priority	S12CMPAI 割り込みおよび S12CMPBI 割り込みの優先順位を設定します。（0～15） 0 を指定した場合、S12CMPAI 割り込みおよび S12CMPBI 割り込みは禁止されます。
windowa_enable	コンペアウィンドウ A 機能の有効／無効を選択します。
windowb_enable	コンペアウィンドウ B 機能の有効／無効を選択します。

注1. チャネルの指定は ADC_MASK_CHn (n はチャネル番号)、ADC_MASK_TEMP（温度センサ）、ADC_MASK_VOLT（内部基準電圧センサ）のいずれか、もしくは組み合わせで指定してください。

注2. ウィンドウ B のチャネル指定は ADC_COMP_WINB_CHn（n はチャネル番号）、ADC_COMP_WINB_TEMP（温度センサ）、ADC_COMP_WINB_VOLT（内部基準電圧センサ）のいずれか 1 つを指定してください。

注3. A/D データレジスタのフォーマット選択（右詰め／左詰め）や A/D 変換精度、A/D 変換値加算モードの設定により設定内容が異なります。詳細はユーザーズマニュアル ハードウェア編を参照してください。

2.11.39　ウィンドウ A/B 複合条件指定（S12ADFa）

```
typedef enum e_adc_comp_cond
{
    ADC_COND_OR = 0x00,
    ADC_COND_EXOR = 0x01,
    ADC_COND_AND = 0x02
} adc_comp_cond_t;
```

列挙型名	説明
ADC_COND_OR	ウィンドウ A 比較条件一致 OR ウィンドウ B 比較条件一致
ADC_COND_EXOR	ウィンドウ A 比較条件一致 EXOR ウィンドウ B 比較条件一致
ADC_COND_AND	ウィンドウ A 比較条件一致 AND ウィンドウ B 比較条件一致

2.11.40　ウィンドウ A/B 組み合わせ結果モニタ用定義（S12ADFa）

```
typedef enum e_adc_comp_stat
{
    ADC_COMP_COND_NOTMET = 0x00,
    ADC_COMP_COND_MET    = 0x01
} adc_comp_stat_t;
```

列挙型名	説明
ADC_COMP_COND_NOTMET	ウィンドウ A/B の複合条件不成立
ADC_COMP_COND_MET	ウィンドウ A/B の複合条件成立

2.11.41　断線検出ディスチャージ／プリチャージ指定（S12ADC、S12ADE、S12ADFa、RX210）

```
typedef enum e_adc_charge
{
    ADC_DDA_DISCHARGE = 0x00,
    ADC_DDA_PRECHARGE = 0x01,
    ADC_DDA_OFF = 0x02
} adc_charge_t;
```

列挙型名	説明
ADC_DDA_DISCHARGE	ディスチャージ選択
ADC_DDA_PRECHARGE	プリチャージ選択
ADC_DDA_OFF	断線検出機能使用しない

2.11.42　サンプリングステートチャネル指定（S12ADb）

サンプリングステートチャネル指定は、MCU ごとに異なります。RX113 でのトリガ指定定義を以下に示します。

```
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_CH8_TO_15,
    ADC_SST_CH21,
    ADC_SST_TEMPERATURE,
    ADC_SST_VOLTAGE,
    ADC_SST_REG_MAX = ADC_SST_VOLTAGE
} adc_sst_reg_t;
```

列挙型名	説明
ADC_SST_CHn	チャネル n を選択 RX110、RX111：n = 0 ～ 4、6 RX113：n = 0 ～ 7、21 RX210：n = 0 ～ 7 使用する MCU に存在するチャネルのみ使用してください
ADC_SST_CH8_TO_15	チャネル 8～15 を選択
ADC_SST_TEMPERATURE	温度センサを選択
ADC_SST_VOLTAGE	内部基準電圧を選択

2.11.43　サンプリングステートチャネル指定（S12ADa）

```
typedef enum e_adc_sst_reg
{
    ADC_SST_CH0_TO_20,
    ADC_SST_TEMPERATURE
} adc_sst_reg_t;
```

列挙型名	説明
ADC_SST_CH0_TO_20	チャネル 0～20 を選択 使用する MCU に存在するチャネルのみ使用してください
ADC_SST_TEMPERATURE	温度センサを選択

2.11.44　サンプリングステートチャネル指定（S12ADE）

```
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_CH16_TO_31,
    ADC_SST_TEMPERATURE,
    ADC_SST_VOLTAGE,
    ADC_SST_REG_MAX = ADC_SST_VOLTAGE
} adc_sst_reg_t;
```

列挙型名	説明
ADC_SST_CHn	チャネル n を選択（n=0～7） 使用する MCU に存在するチャネルのみ使用してください
ADC_SST_CH16_TO_31	チャネル 16～31 を選択
ADC_SST_TEMPERATURE	温度センサを選択
ADC_SST_VOLTAGE	内部基準電圧を選択

2.11.45　サンプリングステートチャネル指定（S12ADC）

```
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_CH8_TO_20,
    ADC_SST_TEMPERATURE,
    ADC_SST_VOLTAGE,
    ADC_SST_REG_MAX = ADC_SST_VOLTAGE
} adc_sst_reg_t;
```

列挙型名	説明
ADC_SST_CHn	チャネル n を選択（n=0～7） 使用する MCU に存在するチャネルのみ使用してください
ADC_SST_CH8_TO_20	チャネル 8～20 を選択（ユニット 1 のみ使用可）
ADC_SST_TEMPERATURE	温度センサを選択
ADC_SST_VOLTAGE	内部基準電圧を選択

2.11.46　サンプリングステートチャネル指定（S12ADFa）

```
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_CH8,
    ADC_SST_CH9,
    ADC_SST_CH10,
    ADC_SST_CH11,
    ADC_SST_CH12,
    ADC_SST_CH13,
    ADC_SST_CH14,
    ADC_SST_CH15,
    ADC_SST_CH16_TO_20,
    ADC_SST_TEMPERATURE,
    ADC_SST_VOLTAGE,
    ADC_SST_REG_MAX
} adc_sst_reg_t;
```

列挙型名	説明
ADC_SST_CHn	チャネル n を選択（n=0～15） 使用する MCU に存在するチャネルのみ使用してください。 ユニット 1 にはチャネル 0～7 まで選択可能です。
ADC_SST_CH16_TO_20	チャネル 16～20 を選択（ユニット 1 のみ使用可）
ADC_SST_TEMPERATURE	温度センサを選択
ADC_SST_VOLTAGE	内部基準電圧を選択

2.11.47　グループ A 優先制御動作指定（S12ADC、S12ADE）

```
typedef enum e_adc_grpa
{
    ADC_GRPA_PRIORITY_OFF = 0,
    ADC_GRPA_GRPB_WAIT_TRIG = 1,
    ADC_GRPA_GRPB_RESTART_SCAN = 3,
    ADC_GRPA_GRPB_CONT_SCAN= 0x8001,
} adc_grpa_t;
```

列挙型名	説明
ADC_GRPA_PRIORITY_OFF	グループ A の優先制御動作を行わない
ADC_GRPA_GRPB_WAIT_TRIG	グループ A の優先制御でグループ B の A/D 変換動作中断後の再起動をしない
ADC_GRPA_GRPB_RESTART_SCAN（注 1）	グループ A の優先制御でグループ B の A/D 変換動作中断後の再起動をする
ADC_GRPA_GRPB_CONT_SCAN	グループ B のシングルスキャン連続動作（グループ A の A/D 変換要求発生時、グループ A を優先動作）

注1.　本設定を行う場合は、周辺モジュールクロック PCLK と A/D 変換クロック ADCLK の周波数比を 1：1 にしてください。

2.11.48　グループ優先制御動作指定（S12ADFa）

```
typedef enum e_adc_grpa
{
    ADC_GRPA_PRIORITY_OFF = 0,
    ADC_GRPA_GRPB_GRPC_WAIT_TRIG = 1,
    ADC_GRPA_GRPB_GRPC_TOP_RESTART_SCAN = 3,
    ADC_GRPA_GRPB_GRPC_RESTART_TOP_CONT_SCAN = 0x8003,
    ADC_GRPA_GRPB_GRPC_RESTART_SCAN = 0x4003,
    ADC_GRPA_GRPB_GRPC_TOP_CONT_SCAN = 0x8001,
    ADC_GRPA_GRPB_GRPC_RESTART_CONT_SCAN = 0xC003,
} adc_grpa_t;
```

列挙型名	説明
ADC_GRPA_PRIORITY_OFF	優先制御動作を行わない
ADC_GRPA_GRPB_GRPC_WAIT_TRIG	グループ優先制御で低優先グループの A/D 変換動作中断後の再起動をしない
ADC_GRPA_GRPB_GRPC_TOP_RESTART_SCAN	グループ優先制御で低優先グループの A/D 変換動作中断後、先頭チャネルから再起動をする
ADC_GRPA_GRPB_GRPC_RESTART_TOP_CONT_SCAN（注 1）	最も優先度の低いグループのシングルスキャン連続動作 グループ優先制御で低優先グループの A/D 変換動作中断後、先頭チャネルから再起動する
ADC_GRPA_GRPB_GRPC_RESTART_SCAN	グループ優先制御で低優先グループの A/D 変換動作中断後、未終了チャネルから再起動をする
ADC_GRPA_GRPB_GRPC_TOP_CONT_SCAN（注 1）	最も優先度の低いグループのシングルスキャン連続動作 グループ優先制御で低優先グループの A/D 変換動作中断後の再起動をしない
ADC_GRPA_GRPB_GRPC_RESTART_CONT_SCAN（注 1）	最も優先度の低いグループのシングルスキャン連続動作 グループ優先制御で低優先グループの A/D 変換動作中断後、未終了チャネルから再起動をする

注1.　本設定を行う場合は、周辺モジュールクロック PCLK と A/D 変換クロック ADCLK の周波数比を 1：1 にしてください。

2.11.49　自己診断モード指定（S12ADC、S12ADE、S12ADFa）

```
typedef enum e_adc_diag
{
    ADC_DIAG_OFF         = 0x00,
    ADC_DIAG_0_VOLT      = 0x01,
    ADC_DIAG_HALF_VREFH0 = 0x02,
    ADC_DIAG_VREFH0      = 0x03,
    ADC_DIAG_ROTATE_VOLTS = 0x04
} adc_diag_t;
```

列挙型名	説明
ADC_DIAG_OFF	自己診断を使用しない
ADC_DIAG_0_VOLT	0V の電圧を使って自己診断を行う
ADC_DIAG_HALF_VREFH0	基準電源×1/2 の電圧を使って自己診断を行う
ADC_DIAG_VREFH0	基準電源の電圧を使って自己診断を行う
ADC_DIAG_ROTATE_VOLTS	自己診断ローテーションモードを使用する

2.11.50　R_ADC_Read 関数でのチャネル指定（S12ADb、ただし RX210 を除く）

```
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_CH6 = 6,
    ADC_REG_CH8 = 8,
    ADC_REG_CH9 = 9,
    ADC_REG_CH10 = 10,
    ADC_REG_CH11 = 11,
    ADC_REG_CH12 = 12,
    ADC_REG_CH13 = 13,
    ADC_REG_CH14 = 14,
    ADC_REG_CH15 = 15,
    ADC_REG_TEMP = 16,
    ADC_REG_VOLT = 17,
    ADC_REG_DBLTRIG = 18,
    ADC_REG_MAX = ADC_REG_DBLTRIG
} adc_reg_t;
```

列挙型名	説明
ADC_REG_CHn（注 1）	チャネル n の A/D 変換値を指定
ADC_REG_TEMP	温度センサの A/D 変換値を指定
ADC_REG_VOLT	内部基準電圧センサの A/D 変換値を指定
ADC_REG_DBLTRIG	ダブルトリガの A/D 変換値を指定

注1.　使用できるチャネルは MCU やピン数により異なります。本章では RX110 の定義を記載しています。

2.11.51　R_ADC_Read 関数でのチャネル指定（S12ADE、RX210）

```
typedef enum e_adc_reg
{
    ADC_REG_CH0  = 0,
    ADC_REG_CH1,
    ADC_REG_CH2,
    ADC_REG_CH3,
    ADC_REG_CH4,
    ADC_REG_CH5,
    ADC_REG_CH6,
    ADC_REG_CH7,
    ADC_REG_CH16,
    ADC_REG_CH17,
    ADC_REG_CH18,
    ADC_REG_CH19,
    ADC_REG_CH20,
    ADC_REG_CH21,
    ADC_REG_CH24,
    ADC_REG_CH25,
    ADC_REG_CH26,
    ADC_REG_TEMP,
    ADC_REG_VOLT,
    ADC_REG_DBLTRIG,
    ADC_REG_SELF_DIAG,
    ADC_REG_MAX = ADC_REG_SELF_DIAG
} adc_reg_t;
```

列挙型名	説明
ADC_REG_CHn（注 1）	チャネル n の A/D 変換値を指定
ADC_REG_TEMP	温度センサの A/D 変換値を指定
ADC_REG_VOLT	内部基準電圧センサの A/D 変換値を指定
ADC_REG_DBLTRIG	ダブルトリガの A/D 変換値を指定
ADC_REG_SELF_DIAG	自己診断の A/D 変換値を指定

注1.　使用できるチャネルは MCU やピン数により異なります。本章では RX130 の定義を記載しています。

2.11.52　R_ADC_Read 関数でのチャネル指定（S12ADa）

```
typedef enum e_adc_reg
{
    ADC_REG_CH0 = 0,
    ADC_REG_CH1,
    ADC_REG_CH2,
    ADC_REG_CH3,
    ADC_REG_CH4,
    ADC_REG_CH5,
    ADC_REG_CH6,
    ADC_REG_CH7,
    ADC_REG_CH8,
    ADC_REG_CH9,
    ADC_REG_CH10,
    ADC_REG_CH11,
    ADC_REG_CH12,
    ADC_REG_CH13,
    ADC_REG_CH14,
    ADC_REG_CH15,
    ADC_REG_CH16,
    ADC_REG_CH17,
    ADC_REG_CH18,
    ADC_REG_CH19,
    ADC_REG_CH20,
    ADC_REG_TEMP,
    ADC_REG_VOLT,
    ADC_REG_MAX = ADC_REG_VOLT
} adc_reg_t;
```

列挙型名	説明
ADC_REG_CHn（注 1）	チャネル n の A/D 変換値を指定
ADC_REG_TEMP	温度センサの A/D 変換値を指定
ADC_REG_VOLT	内部基準電圧センサの A/D 変換値を指定

注1.　使用できるチャネルは MCU やピン数により異なります。本章では RX63N の定義を記載しています。

2.11.53　R_ADC_Read 関数でのチャネル指定（S12ADC、S12ADFa）

```
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_CH8 = 8,
    ADC_REG_CH9 = 9,
    ADC_REG_CH10 = 10,
    ADC_REG_CH11 = 11,
    ADC_REG_CH12 = 12,
    ADC_REG_CH13 = 13,
    ADC_REG_CH14 = 14,
    ADC_REG_CH15 = 15,
    ADC_REG_CH16 = 16,
    ADC_REG_CH17 = 17,
    ADC_REG_CH18 = 18,
    ADC_REG_CH19 = 19,
    ADC_REG_CH20 = 20,
    ADC_REG_TEMP,
    ADC_REG_VOLT,
    // ユニット 0、1
    ADC_REG_DBLTRIG,
    ADC_REG_DBLTRIGA,
    ADC_REG_DBLTRIGB,
    ADC_REG_SELF_DIAG,
    ADC_REG_MAX = ADC_REG_SELF_DIAG
} adc_reg_t;
```

列挙型名	説明
ADC_REG_CHn（注 1）	チャネル n の A/D 変換値を指定
ADC_REG_TEMP	温度センサの A/D 変換値を指定
ADC_REG_VOLT	内部基準電圧センサの A/D 変換値を指定
ADC_REG_DBLTRIG	ダブルトリガの A/D 変換値を指定
ADC_REG_DBLTRIGA	ダブルトリガ拡張モードの A/D 変換値を指定（ADDBLDRA レジスタ）
ADC_REG_DBLTRIGB	ダブルトリガ拡張モードの A/D 変換値を指定（ADDBLDRB レジスタ）
ADC_REG_SELF_DIAG	自己診断の A/D 変換値を指定

注1.　使用できるチャネルは MCU やピン数により異なります。本章では RX64M の定義を記載しています。

2.11.54　R_ADC_ReadAll 関数での A/D 変換結果格納用構造体（S12ADb、ただし RX210 を除く）

```
typedef struct st_adc_data
{
    uint16_t    chan[ADC_REG_ARRAY_MAX];
    uint16_t    temp;
    uint16_t    volt;
    uint16_t    dbltrig;
} adc_data_t;
```

メンバ	説明
chan[ADC_REG_ARRAY_MAX]	各チャネルの A/D 変換結果（注 1）
temp	温度センサの A/D 変換結果
volt	内部基準電圧の A/D 変換結果
dbltrig	ダブルトリガの A/D 変換結果

注1.　チャネルの指定は ADC_REG_CHn（n はチャネル番号）で指定してください。

2.11.55　R_ADC_ReadAll 関数での A/D 変換結果格納用構造体（S12ADE、RX210）

```
typedef struct st_adc_data
{
    uint16_t    chan[ADC_REG_ARRAY_MAX];
    uint16_t    temp;
    uint16_t    volt;
    uint16_t    dbltrig;
    uint16_t    self_diag;
} adc_data_t;
```

メンバ	説明
chan[ADC_REG_ARRAY_MAX]	各チャネルの A/D 変換結果（注 1）
temp	温度センサの A/D 変換結果
volt	内部基準電圧の A/D 変換結果
dbltrig	ダブルトリガの A/D 変換結果
self_diag	自己診断の A/D 変換結果

注1.　チャネルの指定は ADC_REG_CHn（n はチャネル番号）で指定してください。

2.11.56　R_ADC_ReadAll 関数での A/D 変換結果格納用構造体（S12ADa）

```
typedef struct st_adc_data
{
    uint16_t    chan[ADC_REG_ARRAY_MAX];
    uint16_t    temp;
    uint16_t    volt;
} adc_data_t;
```

メンバ	説明
chan[ADC_REG_ARRAY_MAX]	各チャネルの A/D 変換結果（注 1）
temp	温度センサの A/D 変換結果
volt	内部基準電圧の A/D 変換結果

注1.　チャネルの指定は ADC_REG_CHn（n はチャネル番号）で指定してください。

2.11.57　R_ADC_ReadAll 関数での A/D 変換結果格納用構造体（S12ADC、S12ADFa）

```
typedef struct st_adc_data
{
    adc_unit0_data_t   unit0;
    adc_unit1_data_t   unit1;
} adc_data_t;
```

メンバ	説明
unit0	ユニット 0 用 A/D 変換結果格納用構造体
unit1	ユニット 1 用 A/D 変換結果格納用構造体

2.11.58　ユニット 0 用 A/D 変換結果格納用構造体（S12ADC、S12ADFa）

```
typedef struct st_adc_unit0_data
{
    uint16_t    chan[ADC_0_REG_ARRAY_MAX];
    uint16_t    dbltrig;
    uint16_t    dbltrigA;
    uint16_t    dbltrigB;
    uint16_t    self_diag;
} adc_unit0_data_t;
```

メンバ	説明
chan[ADC_REG_ARRAY_MAX]	各チャネルの A/D 変換結果（注 1）
dbltrig	ダブルトリガの A/D 変換結果
dbltrigA	ダブルトリガ拡張モードの A/D 変換結果（ADDBLDRA レジスタ）
dbltrigB	ダブルトリガ拡張モードの A/D 変換結果（ADDBLDRB レジスタ）
self_diag	自己診断の A/D 変換結果

注1.　チャネルの指定は ADC_REG_CHn（n はチャネル番号）で指定してください。

2.11.59　ユニット 1 用 A/D 変換結果格納用構造体（S12ADC、S12ADFa）

```
typedef struct st_adc_unit1_data
{
    uint16_t    chan[ADC_1_REG_ARRAY_MAX];
    uint16_t    temp;
    uint16_t    volt;
    uint16_t    dbltrig;
    uint16_t    dbltrigA;
    uint16_t    dbltrigB;
    uint16_t    self_diag;
} adc_unit1_data_t;
```

メンバ	説明
chan[ADC_REG_ARRAY_MAX]	各チャネルの A/D 変換結果（注 1）
temp	温度センサの A/D 変換結果
volt	内部基準電圧の A/D 変換結果
dbltrig	ダブルトリガの A/D 変換結果
dbltrigA	ダブルトリガ拡張モードの A/D 変換結果（ADDBLDRA レジスタ）
dbltrigB	ダブルトリガ拡張モードの A/D 変換結果（ADDBLDRB レジスタ）
self_diag	自己診断の A/D 変換結果

注1.　チャネルの指定は ADC_REG_CHn（n はチャネル番号）で指定してください。

2.12 戻り値

API 関数の戻り値を示します。この列挙型は、API 関数の宣言とともに r_s12ad_rx_if.h に記載されています。

```
typedef enum e_adc_err          // ADC API エラーコード
{
ADC_SUCCESS = 0,
ADC_ERR_AD_LOCKED,              //他の処理で R_ADC_Open()を呼び出し中です。
ADC_ERR_AD_NOT_CLOSED,          //周辺機能が別のモードで動作中です。
ADC_ERR_MISSING_PTR,            //要求される引数のポインタがありません。
ADC_ERR_INVALID_ARG,            //パラメータに対して引数が無効です。
ADC_ERR_ILLEGAL_ARG,            //モードに対して引数が不正です。
ADC_ERR_SCAN_NOT_DONE,          //A/D 変換が未完了です。
ADC_ERR_TRIG_ENABLED,           //A/D 変換実行中のため、コンペアマッチを設定できません。
ADC_ERR_CONDITION_NOT_MET,      //いずれのチャネル/センサもコンペアマッチの条件を
                                //満たしていません。
ADC_ERR_UNKNOWN                 //不明なハードウェアエラー
} adc_err_t;
```

2.13 FIT モジュールの追加方法

本モジュールは、使用するプロジェクトごとに追加する必要があります。ルネサスでは、Smart Configurator を使用した(1)、(3)の追加方法を推奨しています。ただし、Smart Configurator は、一部の RX デバイスのみサポートしています。サポートされていない RX デバイスについては(2)、(4)の方法を使用してください。

(1) e² studio 上で Smart Configurator を使用して FIT モジュールを追加する場合
e² studio の Smart Configurator を使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「Renesas e² studio スマート・コンフィグレータ ユーザーガイド (R20AN0451)」を参照してください。

(2) e² studio 上で FIT Configurator を使用して FIT モジュールを追加する場合
e² studio の FIT Configurator を使用して、自動的にユーザプロジェクトに FIT モジュールを追加することができます。詳細は、アプリケーションノート「RX ファミリ e² studio に組み込む方法 Firmware Integration Technology (R01AN1723)」を参照してください。

(3) CS+上で Smart Configurator を使用して FIT モジュールを追加する場合
CS+上で、スタンドアロン版 Smart Configurator を使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「Renesas e² studio スマート・コンフィグレータ ユーザーガイド (R20AN0451)」を参照してください。

(4) CS+上で FIT モジュールを追加する場合
CS+上で、手動でユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「RX ファミリ CS+に組み込む方法 Firmware Integration Technology (R01AN1826)」を参照してください。

3. API 関数

3.1 概要

本モジュールには以下の関数が含まれます。

関数	説明
R_ADC_Open()	12 ビット A/D コンバータ (および、温度センサ (RX210、RX63x))を起動し、動作モード、トリガ要因、割り込み優先順位と全チャネルおよびセンサに共通の設定を行います。また、コールバック関数を設定すると、A/D 変換完了時、またはコンペアマッチの条件が合致したとき、割り込み処理でコールバック関数が呼び出されます。
R_ADC_Control()	12 ビット A/D コンバータの動作に関するコマンドを提供します。コマンドには、使用するチャネルまたはセンサの設定、トリガ要因および割り込みの有効／無効設定、ソフトウェアトリガ開始、A/D 変換完了の確認に関するコマンドが含まれます。
R_ADC_Read()	単一のチャネル、センサ、ダブルトリガ、または自己診断のいずれかのレジスタから変換結果を読み出します。
R_ADC_ReadAll()	MCU で対応しているすべての変換結果格納レジスタを読み出します。このとき、チャネルの有効／無効は関係ありません。
R_ADC_Close()	処理中の A/D 変換を終了し、割り込みを無効にして、A/D コンバータを終了します。
R_ADC_GetVersion()	本 FIT モジュールのバージョン番号を返します。

3.2 R_ADC_Open ()

12 ビット A/D コンバータを起動し、動作モード、トリガ要因、割り込み優先順位、コールバック関数と全チャネルおよびセンサに共通の設定を行います。コールバック関数は、割り込み優先順位が 0 以外の場合、A/D 変換完了時、またはコンペアマッチの条件が合致したときに割り込み処理にて呼び出されます。

この関数は ADC FIT モジュールを初期化する関数です。この関数は他の API 関数を使用する前に実行される必要があります。

Format

```
adc_err_t R_ADC_Open(uint8_t             unit,
                     adc_mode_t const    mode,
                     adc_cfg_t * const   p_cfg,
                     void                (* const p_callback)(void *p_args));
```

Parameters

unit

“0”または“1”を設定します。ユニットを 1 つしか持たない MCU では、“0”を設定してください。RX64M/RX71M/RX65x のみがユニットを 2 つ持ちます。

mode

動作モードを指定します。動作モードについては、「2.11.3 S12AD 動作モード指定（S12ADb、S12ADE）」～「2.11.6 S12AD 動作モード指定（S12ADFa）」を参照ください。

本パラメータは実行される A/D 変換のタイプを示します。

“ADC_MODE_SS_MULTI_CH_GROUPED_DBLTRIG_A”、または

“ADC_MODE_SS_MULTI_CH_GROUPED_DBLTRIG_A_GROUPC” を指定した場合、グループ A には 1 チャネルのみ指定可能です。

p_cfg

12 ビット A/D の機能設定用構造体へのポインタ。機能設定用構造体については、「2.11.7 S12AD 機能設定用構造体（S12ADb、ただし RX210 を除く）」～「2.11.12 S12AD 機能設定用構造体（S12ADFa）」を参照ください。

p_callback

A/D 変換完了時、またはコンペアマッチの条件が合致したとき、割り込みから呼び出される関数のポインタ。使用しない場合は、FIT_NO_PTR を設定してください。

Return Values

ADC_SUCCESS	*/*処理が正常に完了 */*
ADC_ERR_AD_LOCKED	*/*他の処理で R_ADC_Open()関数を実行中です。*/*
ADC_ERR_AD_NOT_CLOSED	*/*周辺機能が別のモードで動作中です。先に R_ADC_Close()を*/*
	*/*実行してください。*/*
ADC_ERR_INVALID_ARG	*/* "p_cfg"構造体に指定した設定は無効です。。*/*
ADC_ERR_ILLEGAL_ARG	*/*モードに対して引数が不正です。*/*
ADC_ERR_MISSING_PTR	*/* "p_cfg"のポインタが FIT_NO_PTR/NULL です。

Properties

r_s12ad_rx_if.h にプロトタイプ宣言されています。

Description

MCU 周辺機能の A/D コンバータを起動し、動作モード、トリガ要因、割り込み優先順位と全チャネルおよびセンサに共通の設定を行います。割り込み優先順位に 0 以外を設定している場合、A/D 変換完了時、またはコンペアマッチの条件が合致したときに、割り込み処理にてコールバック関数を呼び出します。割り込み優先順位を 0 に設定している場合、コールバック関数は呼び出されません。必要に応じて、A/D 変換の完了を R_ADC_Control 関数で確認してください。

この関数で使用する引数の値を設定するときは、最初に引数の全メンバを"0"でクリアしてから値を設定してください。

Reentrant

この関数は、再入不可です。

Example (S12ADb、ただし RX210 を除く)

```
adc_cfg_t   config;

      /* adc_cfg_t 構造体のすべてのメンバをクリア */
      memset(&config, 0, sizeof(config));

    /* 温度センサ出力をシングルスキャンするための初期化
     *  - スキャン開始にソフトウェアトリガを使用、A/D 変換完了をポーリング
     *  - A/D 変換値の加算はしない
     *  - A/D データレジスタは右詰め、読み出し後に自動クリアしない
     *  - 通常変換動作
     */
    config.trigger = ADC_TRIG_SOFTWARE;
    config.priority = 0;              // ポーリングを表す
    config.add_cnt = ADC_ADD_OFF;
    config.alignment = ADC_ALIGN_RIGHT;
    config.clearing = ADC_CLEAR_AFTER_READ_OFF;
    config.conv_speed = ADC_CONVERT_SPEED_NORM;

    R_ADC_Open(0, ADC_MODE_SS_TEMPERATURE, &config, FIT_NO_FUNC);
```

Special Notes (RX 共通):

R_ADC_Open()関数を呼び出す前に、アプリケーションで MPC および PORT を設定してください。アナログ端子と同じポートで出力端子を使用する場合は、ユーザーズマニュアル　ハードウェア編で制限事項をご確認ください。以下に RSKRX111 Rev 1 ボードの初期設定サンプルを示します。

```
    R_BSP_RegisterProtectDisable(BSP_REG_PROTECT_MPC);

    PORT4.PDR.BIT.B0 = 0;       // A/D 変換ポートを入力に設定する
    PORT4.PMR.BIT.B0 = 0;       // A/D 変換ポートを汎用入出力に設定する
    MPC.P40PFS.BYTE = 0x80;     // P40 ポートの機能選択を A/D 変換ポート(AN000)にする

    MPC.PB0PFS.BIT.PSEL = 0x09; // PB0 ポートの機能選択を ADTRIG0 にする
                                // (RSKRX111 ボードの SW3)
    PORTB.PDR.BIT.B0 = 0;       // ADTRIG0 を入力に設定する
    PORTB.PMR.BIT.B0 = 1;       // ADTRIG0 を周辺機能に設定する

    R_BSP_RegisterProtectEnable(BSP_REG_PROTECT_MPC);
```

R_ADC_Open()関数を呼び出す前に、A/D 変換クロックを設定してください。

連続スキャンモードで A/D 変換を開始後に、A/D 変換を停止させる場合は、R_ADC_Close 関数を呼び出してください。

連続スキャンモードを選択した場合、A/D 変換完了が連続して発生するため、S12ADI 割り込みの使用は推奨されません。

割り込みを使用する場合、単一の引数を取るコールバック関数が必要です。この引数は構造体へのポインタで、他の FIT モジュールのコールバック関数と合わせるために void ポインタにキャストされます。割り込み処理内で adc_cb_args_t ポインタにキャストして使用してください。adc_cb_args_t の内容については、「2.11.1 コールバック関数のイベント定義 (全 MCU 共通)」を参照してください。

以下にコールバック関数のテンプレートの例を示します。

```
void MyCallback(void *p_args)
{
adc_cb_args_t    *args;

    args = (adc_cb_args_t *)p_args;

    if (args->event == ADC_EVT_SCAN_COMPLETE)
    {
        // ここで A/D 変換結果を読み出し
        nop();
    }
    else if (args->event == ADC_EVT_GROUPB_SCAN_COMPLETE)
    {
        // ここでグループ B の A/D 変換結果を読み出し
        nop();
    }
    else if (args->event == ADC_EVT_CONDITION_MET)
    {
        // args->compare_flags がコンペアマッチの条件を満たしたチャネル/センサを示す
        nop();
    }
}
```

Special Notes (S12ADa):

温度センサ出力と内部基準電圧のみレジスタ自動クリアを有効にしても A/D 変換結果がクリアされません。

R_ADC_Open 関数実行後、10ms 以上待ってから A/D 変換を実行してください。

Special Notes (S12ADb、A12ADC、S12ADE、S12ADFa):

R_ADC_Open 関数実行後、1μs 以上待ってから A/D 変換を実行してください。

3.3 R_ADC_Control()

12 ビット A/D コンバータの機能を設定するための関数です。

Format

```
adc_err_t R_ADC_Control(uint8_t          unit,
                        adc_cmd_t const  cmd,
                        void * const     p_args);
```

Parameters

unit

“0”または“1”を設定します。ユニットを 1 つしか持たない MCU では、“0”を設定してください。RX64M/RX71M/RX65x のみがユニットを 2 つ持ちます。

cmd

実行するコマンドを指定します。コマンドおよびコマンドで使用する引数については、「2.11.23 R_ADC_Control 関数で使用するコマンド」を参照ください。

p_args

任意の設定用構造体へのポインタ。最初に引数の全メンバを“0”でクリアしてから値を設定してください。コマンドが引数を取らない場合、引数には FIT_NO_PTR を設定してください。

Return Values

ADC_SUCCESS	*/*処理が正常に完了*/*
ADC_ERR_MISSING_PTR	*/*“p_args”のポインタが FIT_NO_PTR/NULL です。*/*
	*/*引数が必要です。*/*
ADC_ERR_INVALID_ARG	*/* “p_args”構造体に指定した設定は無効です。*/*
ADC_ERR_ILLEGAL_ARG	*/*モードに対して“cmd”の引数が不正です。*/*
ADC_ERR_SCAN_NOT_DONE	*/*要求された A/D 変換が完了していません。*/*
ADC_ERR_TRIG_ENABLED	*/*A/D 変換実行中のため、コンペアマッチを設定できません。*/*
ADC_ERR_CONDITION_NOT_MET,	*/*いずれのチャネル/センサもコンペアマッチの条件を*/*
	*/*満たしていません。*/*

Properties

r_s12ad_rx_if.h にプロトタイプ宣言されています。

Description

12 ビット A/D コンバータの動作に関するコマンドを提供します。コマンドには、使用するチャネルまたはセンサの設定、トリガ要因および割り込みの有効／無効設定、ソフトウェアトリガ開始、A/D 変換完了の確認に関するコマンドが含まれます。

R_ADC_Open()関数呼び出し後、R_ADC_Control()関数で下記のコマンドの発行が可能です。

なお、コマンドは以下の No.順に、必要な No.のコマンドのみを発行してください。

No	コマンド	説明	第 3 引数(p_args)
1	ADC_CMD_SET_DDA_STATE_CNT	S12ADC、S12ADE、S12ADFa、および RX210 で有効です。 断線検出アシスト機能の設定。 断線検出アシスト機能を使用しない場合はコマンドの発行は不要です。	断線検出設定用構造体（adc_dda_t）
2	ADC_CMD_SET_SAMPLE_STATE_CNT	アナログ入力のサンプリング時間を設定します。 初期値（ADSSTRn レジスタのリセット後の値）から変更しない場合は、コマンドの発行は不要です。	サンプリングステート設定用構造体（adc_sst_t、または adc_time_t）
3	ADC_CMD_USE_VREFL0	S12ADE で有効です。 高電位側基準電圧を VREFH0 に設定します。AVCC0 を選択する場合は、コマンドの発行は不要です。	FIT_NO_PTR
4	ADC_CMD_USE_VREFH0	S12ADE で有効です。 低電位側基準電圧を VREFL0 に設定します。AVSS0 を選択する場合は、コマンドの発行は不要です。	FIT_NO_PTR
5	ADC_CMD_ENABLE_CHANS	A/D 変換を行うチャネルの選択および設定を行います。	変換チャネル設定用構造体（adc_ch_cfg_t）
6	ADC_CMD_ENABLE_TEMP_SENSOR	S12ADa、S12ADb、S12ADE で有効です。 温度センサを有効にします。 温度センサを使用しない場合、コマンドの発行は不要です。	FIT_NO_PTR
7	ADC_CMD_ENABLE_VOLT_SENSOR	S12ADa、S12ADb、S12ADE で有効です。 内部基準電圧センサを有効にします。 内部基準電圧センサを使用しない場合、コマンドの発行は不要です。	FIT_NO_PTR
8	ADC_CMD_EN_COMPARATOR_LEVEL	S12ADC、S12ADE、S12ADFa で有効です。 コンペア機能（ウィンドウ機能無効）の設定を行います。 コンペア機能を使用しない場合はコマンドの発行は不要です。	コンペア機能設定用構造体（adc_cmpwin_t）
9	ADC_CMD_EN_COMPARATOR_WINDOW	S12ADC、S12ADE、S12ADFa で有効です。 コンペア機能（ウィンドウ機能有効）の設定を行います。 コンペア機能を使用しない場合はコマンドの発行は不要です。	コンペア機能設定用構造体（adc_cmpwin_t）
10	ADC_CMD_TRIG_ENABLE	ハードウェアトリガを有効にします。 ソフトウェアトリガを選択した場合は、コマンドの発行は不要です。	FIT_NO_PTR
11	ADC_CMD_SCAN_NOW	ソフトウェアトリガによる A/D 変換を開始します。 ハードウェアトリガを選択した場合は、コマンドの発行は不要です。	FIT_NO_PTR

No	コマンド	説明	第 3 引数(p_args)
12	ADC_CMD_CHECK_SCAN_DONE	A/D 変換が完了したかを確認します。 コールバック関数を使用せず、ポーリングで A/D 変換完了を確認する場合に使用してください。	FIT_NO_PTR
13	ADC_CMD_CHECK_SCAN_DONE_GROUPA	S12ADb、S12ADC、S12ADE、S12ADFa で使用できます。 グループ A の A/D 変換が完了したかを確認します。 グループ A の割り込み優先レベルを 0 に設定し、ポーリングで A/D 変換を確認する場合に使用してください。	FIT_NO_PTR
14	ADC_CMD_CHECK_SCAN_DONE_GROUPB	S12ADb、S12ADC、S12ADE、S12ADFa で使用できます。 グループ B の A/D 変換が完了したかを確認します。 グループ B の割り込み優先レベルを 0 に設定し、ポーリングで A/D 変換を確認する場合に使用してください。	FIT_NO_PTR
15	ADC_CMD_CHECK_SCAN_DONE_GROUPC	S12ADFa で使用できます。 グループ C の A/D 変換が完了したかを確認します。 グループ C の割り込み優先レベルを 0 に設定し、ポーリングで A/D 変換を確認する場合に使用してください。	FIT_NO_PTR
16	ADC_CMD_CHECK_CONDITION_MET	S12ADC、S12ADE、S12ADFa で使用できます。 コンペア機能による比較結果を引数に指定した変数に格納します。 比較結果はチャネル n の結果がビット n に格納されます。（注 1） 0: 比較条件不成立 1: 比較条件成立	比較結果を格納する uint32_t 型変数のポインタ
17	ADC_CMD_CHECK_CONDITION_METB	S12ADFa で使用できます。 グループ B のコンペア機能による比較結果を取得します。 引数に指定した変数にグループ B の比較結果が格納されます。（注 1） 0x0000: 比較条件不成立 0x0001: 比較条件成立	比較結果を格納する uint32_t 型変数のポインタ
18	ADC_CMD_COMP_COMB_STATUS	S12ADFa で使用できます。 ウィンドウ A/B 複合条件結果を取得します。 引数に指定した変数にウィンドウ A/B の組み合わせ結果が格納されます。 ADC_COMP_COND_NOTMET: ウィンドウ A/B の複合条件不成立 ADC_COMP_COND_MET: ウィンドウ A/B の複合条件成立	ウィンドウ A/B 組み合わせ結果を格納する adc_comp_stat_t 型変数のポインタ

注1. 本コマンド実行後、比較結果を“0”（比較条件不成立）に初期化します。そのため、A/D 変換完了ごとに、本コマンドを 1 度だけ実行してください。

Reentrant

この関数は、再入不可です。ただし、ADC_CMD_CHECK_SCAN_DONE_GROUPA、ADC_CMD_CHECK_SCAN_DONE_GROUPB、ADC_CMD_CHECK_SCAN_DONE_GROUPC のコマンド実行中のみ再入可能です。

Example 1: ユニット 0、単一チャネルを使用してポーリングを行う場合 (RX64M、RX71M、RX65x)

```
uint16_t      data;
adc_cfg_t     config;
adc_ch_cfg_t   ch_cfg;
adc_err_t     err;

/* S12AD をオープン */

/* adc_cfg_t 構造体のすべてのメンバをクリア */
memset(&config, 0, sizeof(config));

/* S12AD をソフトウェアトリガ、シングルスキャン(1 チャネルのみ)、ポーリングでオープンする */
config.resolution = ADC_RESOLUTION_12_BIT;
config.trigger    = ADC_TRIG_SOFTWARE;
config.priority   = 0;                          // ポーリングを表す
config.add_cnt    = ADC_ADD_OFF;
config.alignment  = ADC_ALIGN_RIGHT;
config.clearing   = ADC_CLEAR_AFTER_READ_OFF;
err = R_ADC_Open(0,ADC_MODE_SS_ONE_CH, &config, NULL);

/* チャネルを有効にする */

/* adc_ch_cfg_t 構造体のすべてのメンバをクリア */
memset(&ch_cfg, 0, sizeof(ch_cfg));

/* RSKRX64M ボードに実装されているボリュームのチャネルを有効にする */
ch_cfg.chan_mask       = ADC_MASK_CH0;
ch_cfg.diag_method     = ADC_DIAG_OFF;
ch_cfg.anex_enable     = false;
ch_cfg.sample_hold_mask = 0;
err = R_ADC_Control(0, ADC_CMD_ENABLE_CHANS, &ch_cfg);

/*  Open 後、A/D 変換開始まで 1μs 以上の待ち時間を設けること */

/* 繰り返しトリガを発生させ、A/D 変換の完了を待って結果を読み出す */
while(1)
{
  /* ソフトウェアトリガを発生 */
  err = R_ADC_Control(0, ADC_CMD_SCAN_NOW, NULL);

  /* A/D 変換完了待ち */
  while (R_ADC_Control(0,ADC_CMD_CHECK_SCAN_DONE,NULL) == ADC_ERR_SCAN_NOT_DONE)
  {
  }

  /* 結果読み出し */
  err = R_ADC_Read(0, ADC_REG_CH0, &data);
}
```

Example 2: ユニット 1、温度センサを使用して、ポーリング、およびステート数の設定を行う場合 (RX64M、RX71M、RX65x)

```
uint16_t      data;
adc_cfg_t     config;
adc_sst_t     sst;          // サンプリングステート
adc_ch_cfg_t   ch_cfg;
adc_err_t    adc_err;

/* S12AD をオープン */

/* adc_cfg_t 構造体のすべてのメンバをクリア */
memset(&config, 0, sizeof(config));

/* S12AD をソフトウェアトリガ、シングルスキャン(温度センサ出力)、ポーリングでオープンする */
config.resolution = ADC_RESOLUTION_10_BIT;
config.trigger  = ADC_TRIG_SOFTWARE;
config.priority = 0;                        // ポーリングを表す
config.add_cnt  = ADC_ADD_OFF;
config.alignment = ADC_ALIGN_RIGHT;
config.clearing  = ADC_CLEAR_AFTER_READ_OFF;
adc_err = R_ADC_Open(1, ADC_MODE_SS_ONE_CH, &config, NULL);


/* ハードウェアに特化した設定 */

/* adc_sst_t 構造体のすべてのメンバをクリア */
memset(&sst, 0, sizeof(sst));

/* adc_ch_cfg_t 構造体のすべてのメンバをクリア */
memset(&ch_cfg, 0, sizeof(ch_cfg));

/* 4us サンプルになるようにサンプリングステート数を設定 *
/* PCLKD が 60MHz の場合、1 ステート = 1/60MHz = 16.7ns, 4us/16.7ns = 240 ステート */
sst.reg_id = ADC_SST_TEMPERATURE;
sst.num_states = 240;
adc_err = R_ADC_Control(1, ADC_CMD_SET_SAMPLE_STATE_CNT, &sst);

/* スキャンの設定 */
ch_cfg.chan_mask = ADC_MASK_TEMP;
ch_cfg.diag_method = ADC_DIAG_OFF;
ch_cfg.anex_enable = false;
ch_cfg.sample_hold_mask = 0;         // ユニット 1 では使用できない
adc_err = R_ADC_Control(1, ADC_CMD_ENABLE_CHANS, &ch_cfg);

/* Open 後、A/D 変換開始まで 1µs 以上の待ち時間を設けること */

/* ソフトウェアトリガを発生 */
adc_err = R_ADC_Control(1, ADC_CMD_SCAN_NOW, NULL);

/* A/D 変換完了待ち */
while (R_ADC_Control(1, ADC_CMD_CHECK_SCAN_DONE, NULL) == ADC_ERR_SCAN_NOT_DONE)
{
}

/* 結果読み出し */
adc_err = R_ADC_Read(1, ADC_REG_TEMP, &data);
```

Example 3: グループスキャンモードを割り込みトリガあり、ダブルトリガモード（グループ A）、かつ 4 回平均に設定する場合 (RX64M、RX71M、RX65x)

```
adc_cfg_t     config;
adc_ch_cfg_t   ch_cfg;

/* トリガソースを設定するため、MTU の初期化をここで行うこと */

/* S12AD のオープン */

/* 各構造体のすべてのメンバをクリア */
memset(&config, 0, sizeof(config));
memset(&ch_cfg, 0, sizeof(ch_cfg));

/* S12AD をグループスキャンモード（ダブルトリガモード使用）で初期化
 *  - 同期トリガ（TRGA0N）でグループ A のスキャンを開始（割り込み優先順位：4）
 *  - 同期トリガ（TRG0N）でグループ B のスキャンを開始（割り込み優先順位：5）
 *  - 許可した各チャネルで次のチャネルをスキャンする前に 4 回スキャンして平均をとる
 *  - A/D 変換値を読み出した後に A/D データレジスタをクリアしない
 */
config.resolution = ADC_RESOLUTION_8_BIT;
config.trigger    = ADC_TRIG_SYNC_TRG0AN;
config.priority   = 4;
config.trigger_groupb = ADC_TRIG_SYNC_TRG0EN;
config.priority_groupb= 5;
config.add_cnt    = ADC_ADD_AVG_4_SAMPLES;
config.alignment  = ADC_ALIGN_RIGHT;
config.clearing   = ADC_CLEAR_AFTER_READ_OFF;
R_ADC_Open(1, ADC_MODE_SS_MULTI_CH_GROUPED_DBLTRIG_A, &config, MyCallback);


/* スキャンの設定 */

/* ダブルトリガにはグループ A の 1 チャネルしか指定できない
   グループ A にはチャネル 8、グループ B にはチャネル 2、3、9 を指定
   加算/平均はチャネル 9 を除く全てのチャネルで行う
*/
ch_cfg.chan_mask = ADC_MASK_CH8;
ch_cfg.chan_mask_groupb = ADC_MASK_CH2 | ADC_MASK_CH3 | ADC_MASK_CH9;
ch_cfg.priority_groupa = ADC_GRPA_PRIORITY_OFF;
ch_cfg.add_mask = ADC_MASK_CH8 | ADC_MASK_CH2 | ADC_MASK_CH3;
ch_cfg.diag_method = ADC_DIAG_OFF;
ch_cfg.anex_enable = false;
ch cfg.sample hold mask = 0;
R_ADC_Control(1, ADC_CMD_ENABLE_CHANS, &ch_cfg);

/* Open 後、A/D 変換開始まで 1µs 以上の待ち時間を設けること */

/* トリガを有効にする */
R_ADC_Control(1, ADC_CMD_ENABLE_TRIG, NULL);

/* スキャン完了時に割り込み発生 */

/* コールバックは割り込みにより 2 回呼ばれる
 *  （各グループのスキャン完了時に一回呼ばれる。呼ばれる順番はトリガの順番に依存する）
 */
void MyCallback(void *p_args)
{
```

```
adc_cb_args_t  *args;
uint16_t       dbltrg,data2,data3,data8,data9;

    args = (adc_cb_args_t *)p_args;

    /* A/D 変換結果を読み出し */

    if (args->event == ADC_EVT_SCAN_COMPLETE)
    {
        /* S12ADIO 割り込み（グループ A スキャン完了）、レジスタ読み出し */
        R_ADC_Read(1, ADC_REG_CH8, &data8);
        R_ADC_Read(1, ADC_REG_DBLTRIG, &dbltrg);
    }
    else if (args->event == ADC_EVT_SCAN_COMPLETE_GROUPB)
    {
        /* GBADI 割り込み（グループ B スキャン完了）、レジスタ読み出し */
        R_ADC_Read(1, ADC_REG_CH2, &data2);
        R_ADC_Read(1, ADC_REG_CH3, &data3);
        R_ADC_Read(1, ADC_REG_CH9, &data9);
    }

    /* アプリケーションのデータ処理、もしくはフラグセットを行う */
}
```

Example 4: グループスキャンモードを割り込みトリガありで設定する場合 (RX65x)

```
adc_cfg_t     config;
adc_ch_cfg_t   ch_cfg;

/* トリガソースを設定するため、MTU の初期化をここで行うこと */

/* S12AD のオープン */

/* adc_cfg_t 構造体のすべてのメンバをクリア */
memset(&config, 0, sizeof(config));

/* S12AD をグループスキャンモード（ダブルトリガモード使用）で初期化
 *  - 同期トリガ（TRGA0N）でグループ A のスキャンを開始（割り込み優先順位：4）
 *  - 同期トリガ（TRGA1N）でグループ B のスキャンを開始（割り込み優先順位：5）
 *  - 同期トリガ（TRGA2N）でグループ C のスキャンを開始（割り込み優先順位：6）
 *  - 許可した各チャネルで次のチャネルをスキャンする前に 4 回スキャンして平均をとる
 *  - A/D 変換値を読み出した後に A/D データレジスタをクリアしない
 */
config.resolution = ADC_RESOLUTION_8_BIT;
config.trigger    = ADC_TRIG_SYNC_TRG0AN;
config.priority   = 4;
config.trigger_groupb = ADC_TRIG_SYNC_TRG1AN;
config.priority_groupb= 5;
config.trigger groupc = ADC TRIG SYNC TRG2AN;
config.priority_groupc= 6;
config.add_cnt    = ADC_ADD_OFF;
config.alignment  = ADC_ALIGN_RIGHT;
config.clearing   = ADC_CLEAR_AFTER_READ_OFF;
R_ADC_Open(0, ADC_MODE_SS_MULTI_CH_GROUPED_GROUPC, &config, MyCallback);

/* スキャンの設定 */

/* adc_ch_cfg_t 構造体のすべてのメンバをクリア */
memset(&ch_cfg, 0, sizeof(ch_cfg));

/* グループ A にはチャネル 1 と 2、グループ B にはチャネル 3 と 4、グループ C にはチャネル 5 と 6 を指定
   加算/平均はチャネル 9 を除く全てのチャネルで行う
*/
ch_cfg.scan_mask = ADC_MASK_CH1 | ADC_MASK_CH2;
ch_cfg.scan_mask_groupb = ADC_MASK_CH3 | ADC_MASK_CH4;
ch_cfg.scan_mask_groupc = ADC_MASK_CH5 | ADC_MASK_CH6;
ch_cfg.priority_groupa = ADC_GRPA_PRIORITY_OFF;
ch_cfg.add_mask = 0;
ch_cfg.diag_method = ADC_DIAG_OFF;
ch_cfg.anex_enable = false;
ch_cfg.sample_hold_mask = 0;
R_ADC_Control(0, ADC_CMD_CONFIGURE_SCAN, &ch_cfg);

/* Open 後、A/D 変換開始まで 1µs 以上の待ち時間を設けること */

/* トリガを有効にする */
R_ADC_Control(0, ADC_CMD_ENABLE_TRIG, NULL);


/* スキャン完了時に割り込み発生 */
```

```
/* コールバックは割り込みにより 2 回呼ばれる
 *  （各グループのスキャン完了時に一回呼ばれる。呼ばれる順番はトリガの順番に依存する）
 */
void MyCallback(void *p_args)
{
adc_cb_args_t  *args;
uint16_t       data1,data2,data3,data4,data5,data6;


    args = (adc_cb_args_t *)p_args;

    /* A/D 変換結果を読み出し */

    if (args->event == ADC_EVT_SCAN_COMPLETE)
    {
        /* S12ADIO 割り込み（グループ A スキャン完了）、レジスタ読み出し */
        R_ADC_Read(0, ADC_REG_CH1, &data1);
        R_ADC_Read(0, ADC_REG_CH2, &data2);
    }
    else if (args->event == ADC_EVT_SCAN_COMPLETE_GROUPB)
    {
        /* GBADI 割り込み（グループ B スキャン完了）、レジスタ読み出し */
        R_ADC_Read(0, ADC_REG_CH3, &data3);
        R_ADC_Read(0, ADC_REG_CH4, &data4);
    }
    else if (args->event == ADC_EVT_SCAN_COMPLETE_GROUPC)
    {
        /* GCADI 割り込み（グループ C スキャン完了）、レジスタ読み出し */
        R_ADC_Read(0, ADC_REG_CH5, &data5);
        R_ADC_Read(0, ADC_REG_CH6, &data6);
}
    /* アプリケーションのデータ処理、もしくはフラグセットを行う */
}
```

Example 5: 複数チャネルを割り込みトリガありで設定し、コンペアマッチの合致を確認する場合 (RX64M、RX71M)

```
adc_cfg_t     config;
adc_ch_cfg_t   ch_cfg;
adc_cmpwin_t   cmpwin;

/* トリガソースを設定するため、MTU の初期化をここで行うこと */

/* ユニット 0 のオープン */

/* adc_cfg_t 構造体のすべてのメンバをクリア */
memset(&config, 0, sizeof(config));

config.resolution = ADC_RESOLUTION_12_BIT;
config.trigger = ADC_TRIG_SYNC_TRG0AN;
config.priority = 4;
config.add_cnt = ADC_ADD_OFF;
config.alignment = ADC_ALIGN_RIGHT;
config.clearing = ADC_CLEAR_AFTER_READ_OFF;
R_ADC_Open(0, ADC_MODE_SS_MULTI_CH, &config, MyCallback);
```

```
/* チャネル 3～5 のスキャンの設定 */

/* adc_ch_cfg_t 構造体のすべてのメンバをクリア */
memset(&ch_cfg, 0, sizeof(ch_cfg));

ch_cfg.chan_mask = ADC_MASK_CH3 | ADC_MASK_CH4 | ADC_MASK_CH5;
ch_cfg.diag_method = ADC_DIAG_OFF;
ch_cfg.anex_enable = false;
ch_cfg.sample_hold_mask = 0;
R_ADC_Control(0, ADC_CMD_ENABLE_CHANS, &ch_cfg);


/* チャネル 3、4 が 1.65V 未満に下がったかコンパレータでチェックする */

/* adc_cmpwin_t 構造体のすべてのメンバをクリア */
memset(&cmpwin, 0, sizeof(cmpwin));

cmpwin.compare_mask = ADC_MASK_CH3 | ADC_MASK_CH4;
cmpwin.inside_window_mask = 0;        // 設定レベルを下回ったら条件一致
cmpwin.level_lo = 0x7FF;              // 12-bit の場合、3.3V=0xFFF、1.65V=0x7FF
cmpwin.int_priority = 3;
R_ADC_Control(0, ADC_CMD_EN_COMPARATOR_LEVEL, &cmpwin);


/* トリガを有効にする */
R_ADC_Control(0, ADC_CMD_ENABLE_TRIG, NULL);

/* スキャン完了時に割り込み発生 */

  :

/* コールバックは割り込みから呼ばれる */

void MyCallback(void *p_args)
{
adc_cb_args_t  *args;
uint16_t       data3,data4,data5;


    args = (adc_cb_args_t *)p_args;


    /* A/D 変換結果を読み出し */

    if (args->event == ADC_EVT_SCAN_COMPLETE)
    {
        R_ADC_Read(0, ADC_REG_CH3, &data3);
        R_ADC_Read(0, ADC_REG_CH4, &data4);
        R_ADC_Read(0, ADC_REG_CH5, &data5);
    }
```

```
    if (args->event == ADC_EVT_CONDITION_MET)
    {
        if (args->compare_flags & ADC_MASK_CH3)
        {
            // チャネル 3 の電圧が下回った場合の処理
        }
        else
        {
            // チャネル 4 の電圧が下回った場合の処理
        }
    }
}
```

Example 6: 複数チャネルを割り込みトリガありで設定し、コンペアマッチの 2 回合致を確認する場合(RX65x)

```
adc_cfg_t     config;
adc_ch_cfg_t   ch_cfg;
adc_cmpwin_t   cmpwin;

/* 各構造体のすべてのメンバをクリア */
memset(&config, 0, sizeof(config));
memset(&ch_cfg, 0, sizeof(ch_cfg));
memset(&cmpwin, 0, sizeof(cmpwin));

/* トリガソースを設定するため、MTU の初期化をここで行うこと */

/* ユニット 0 のオープン */

config.resolution = ADC_RESOLUTION_12_BIT;
config.trigger = ADC_TRIG_SYNC_TRG0AN;
config.priority = 4;
config.add_cnt = ADC_ADD_OFF;
config.alignment = ADC_ALIGN_RIGHT;
config.clearing = ADC_CLEAR_AFTER_READ_OFF;
R_ADC_Open(0, ADC_MODE_SS_MULTI_CH, &config, MyCallback);


/* チャネル 3～4 のスキャンの設定 */

ch_cfg.chan_mask = ADC_MASK_CH3 | ADC_MASK_CH4 | ADC_MASK_CH5;
ch_cfg.diag_method = ADC_DIAG_OFF;
ch_cfg.anex_enable = false;
ch_cfg.sample_hold_mask = 0;
R_ADC_Control(0, ADC_CMD_ENABLE_CHANS, &ch_cfg);

/* チャネル 3、4 が 1.65V 未満に下がったかコンパレータでチェックする */

cmpwin.compare_mask = ADC_MASK_CH3 | ADC_MASK_CH4;
cmpwin.compare_maskb = ADC_COMP_WINB_CH5;
cmpwin.inside_window_mask = 0;          // 設定レベルを下回ったら条件一致
cmpwin.inside_window_maskb = ADC_COMP_WINB_COND_BELOW;
cmpwin.level_lo = 0x7FF;               // 12-bit の場合、3.3V=0xFFF、1.65V=0x7FF
cmpwin.level_lob = 0x7FF;              // 12-bit の場合、3.3V=0xFFF、1.65V=0x7FF
cmpwin.int_priority = 3;
cmpwin.windowa_enable = true;
cmpwin.windowb_enable = true;
R_ADC_Control(0, ADC_CMD_EN_COMPARATOR_LEVEL, &cmpwin);
```

```
/* Open 後、A/D 変換開始まで 1μs 以上の待ち時間を設けること */

/* トリガを有効にする */
R_ADC_Control(0, ADC_CMD_ENABLE_TRIG, NULL);

/* スキャン完了時に割り込み発生 */
  :

/* コールバックは割り込みから呼ばれる */

void MyCallback(void *p_args)
{
adc_cb_args_t  *args;
uint16_t       data3,data4,data5;


    args = (adc_cb_args_t *)p_args;

    /* A/D 変換結果の読み出し */

    if (args->event == ADC_EVT_SCAN_COMPLETE)
    {
        R_ADC_Read(0, ADC_REG_CH3, &data3);
        R_ADC_Read(0, ADC_REG_CH4, &data4);
        R_ADC_Read(0, ADC_REG_CH5, &data5);
    }

    if (args->event == ADC_EVT_CONDITION_MET)
    {
        if (args->compare_flags & ADC_MASK_CH3)
        {
            // チャネル 3 の電圧が下回った場合の処理
        }
        if (args->compare_flags & ADC_MASK_CH4)
        {
            // チャネル 4 の電圧が下回った場合の処理
        }
    }

    if (args->event == ADC_EVT_CONDITION_METB)
    {
        // チャネル 5 の電圧が下回った場合の処理
    }
}
```

Special Notes (RX 共通):

ADST ビットが 1 のときは、本関数でモードなどの設定を変更しないでください。なお、変換状態やコンペア結果の取得は可能です。

A/D 変換またはその設定に使用するチャネルを切り替える場合、一度 R_ADC_Close()関数を呼び出した後に、再度 R_ADC_Open()関数を呼び出して開始してください。

R_ADC_Control 関数で A/D 変換完了待ちを行う場合、以下のコマンドを使用してください。

A/D 変換チャネルの設定			R_ADC_Control 関数のコマンド	
モード	A/D 変換開始トリガ	割り込み	A/D 変換開始	A/D 変換完了待ち
シングルスキャン	ソフトウェアトリガ	―	ADC_CMD_SCAN_NOW	ADC_CMD_CHECK_SCAN_DONE
	ソフトウェアトリガ以外	無効	ADC_CMD_ENABLE_TRIG	ADC_CMD_CHECK_SCAN_DONE_GROUPA（注 1）
連続スキャン	ソフトウェアトリガ	無効	ADC_CMD_SCAN_NOW	ADC_CMD_CHECK_SCAN_DONE_GROUPA（注 1）
	ソフトウェアトリガ以外	無効	ADC_CMD_ENABLE_TRIG	ADC_CMD_CHECK_SCAN_DONE_GROUPA（注 1）
グループスキャン	ソフトウェアトリガ以外	無効	ADC_CMD_ENABLE_TRIG	ADC_CMD_CHECK_SCAN_DONE_GROUPA（注 1） ADC_CMD_CHECK_SCAN_DONE_GROUPB（注 2） ADC_CMD_CHECK_SCAN_DONE_GROUPC（注 3）

注1. S12ADa では ADC_CMD_CHECK_SCAN_DONE_GROUPA は使用できません。割り込み要求フラグを直接参照して、A/D 変換完了待ちを行ってください。

注2. ADC_CMD_CHECK_SCAN_DONE_GROUPB はグループ B の A/D 変換完了待ちの時に使用してください。

注3. ADC_CMD_CHECK_SCAN_DONE_GROUPC は S12ADFa のみ使用可能です。

A/D 変換割り込みが有効の場合、シングルスキャン、かつソフトウェアトリガの場合を除き R_ADC_Control 関数で A/D 変換完了待ちを行うことができません。

そのため、A/D 変換割り込みのコールバック関数を使って A/D 変換完了待ちを行ってください。

Special Notes (S12ADC、S12ADFa):

同じユニットで、複数のチャネルとセンサを組み合わせて使用することができます。

ELC は S12ADI とのみ使用でき、S12GBADI、S12CMPI とは使用できません（S12ADC）。

ELC は S12ADI とのみ使用でき、GBADI、GCADI、S12CMPAI、S12CMPBI とは使用できません（S12ADFa）。

最適な結果を得るためには、温度センサのトリガを有効にする前に、A/D 変換設定後 30μs 待つようにしてください。

グループ B が連続スキャンモードで動作していて、グループ A 優先制御動作が選択されている場合、頻発して割り込み処理が実行されるため、S12GBADI 割り込み（S12ADC）、および GBADI 割り込み（S12ADFa）の使用は推奨されません。

トリガを有効にする前に、コンペアマッチを有効にしてください。

機能によっては同時に使用できないものがあります。以下の表でご確認ください。

	ダブルトリガ	グループスキャン	自己診断	加算/平均	ANEX	サンプル&ホールド	グループ A 優先制御	センサ	コンペアマッチ	断線検出アシスト
ダブルトリガ			X			*B		X	X	
グループスキャン					X	*S				
自己診断	X			X	X				X	X
加算/平均			X							
ANEX		X	X					X		X
サンプル&ホールド	*B	*S					*A			
グループ A 優先制御						*A				
センサ	X				X					X
コンペアマッチ	X		X							
断線検出アシスト			X		X			X		

X: 組み合わせての使用は不可。例えば、ANEX はグループスキャンモード、自己診断機能、センサ、断線検出アシストと一緒に使用することはできません。
*A: サンプル&ホールドに使用するチャネルはグループ A にしてください。
*B: サンプル&ホールドに使用するチャネルはグループ B かグループ C にしてください。
*S: サンプル&ホールドに使用するチャネルはグループをまたがないでください。

Special Notes (S12ADE):

本関数では以下の機能はサポートしていません。

- コンペア機能ウィンドウ B
- コンペア機能ウィンドウ A/B の複合条件

Special Notes (S12ADC/S12ADE/S12ADFa):

コンペア機能を使用する場合、チャネル設定の後にコンペア設定を行ってください。

Special Notes (S12ADa):

サンプリングステート数の設定は AN008～AN020 のみ可能です。
（AN000～AN007 は設定値にかかわらず 20 ステート固定となります。）

Special Notes (S12ADb、ただし RX210 は除く):

温度センサ出力、および内部基準電圧の場合、サンプリングステート数が 5μs 以上になるように設定してください。

3.4 R_ADC_Read()

単一のチャネル、センサ、ダブルトリガ、または自己診断の変換結果格納レジスタから変換結果を読み出します。

Format

```
adc_err_t R_ADC_Read(uint8_t          unit,
                     adc_reg_t const  reg_id,
                     uint16_t * const p_data);
```

Parameters

unit

“0”または“1”を設定します。ユニットを 1 つしか持たない MCU では、“0”を設定してください。RX64M/RX71M/RX65x のみがユニットを 2 つ持ちます。

reg_id

読み出すレジスタの ID。レジスタの ID に関しては「2.11.50 R_ADC_Read 関数でのチャネル指定（S12ADb、ただし RX210 を除く）」～「2.11.53 R_ADC_Read 関数でのチャネル指定（S12ADC、S12ADFa）」を参照してください。

p_data

値を入れる変数へのポインタ

Return Values

ADC_SUCCESS	*/*正常に処理を完了*/*
ADC_ERR_INVALID_ARG	*/* “unit”または“reg_id”の値が無効です。 */*
ADC_ERR_MISSING _PTR	*/* “p_data”の値が FIT_NO_PTR/NULL です。 */*

Properties

r_s12ad_rx_if.h にプロトタイプ宣言されています。

Description

単一のチャネル、センサ、ダブルトリガ、または自己診断のいずれかのレジスタから変換結果を読み出します。

Reentrant

この関数は、再入可能（リエントラント）です。

Example

```
uint16_t data;
    :
   /* Read channel 0 on unit 0 */
   R_ADC_Read(0, ADC_CH0_REG, &data);     // “data”には変換値が入っている
```

Special Notes (S12ADb、ただし RX210 を除く):

温度センサ出力、および内部基準電圧は、オープン後の初回変換を読み捨て、2 回目以降の A/D 変換結果を使用するようにしてください。

3.5 R_ADC_ReadAll()

MCU で対応しているすべての変換結果格納レジスタを読み出します。このとき、チャネルの有効／無効は関係ありません。

Format

```
adc_err_t R_ADC_ReadAll(adc_data_t * const p_data);
```

Parameters

p_data

レジスタ値を入れる構造体へのポインタ。構造体については「2.11.54 R_ADC_ReadAll 関数での A/D 変換結果格納用構造体（S12ADb、ただし RX210 を除く）」～「2.11.57 R_ADC_ReadAll 関数での A/D 変換結果格納用構造体（S12ADC、S12ADFa）」を参照してください。

Return Values

ADC_SUCCESS　　*/*正常に処理を完了*/*

ADC_ERR_MISSING _PTR　　*/* “p_data”の値が FIT_NO_PTR/NULL です。*/*

Properties

r_s12ad_rx_if.h にプロトタイプ宣言されています。

Description

チャネルの有効／無効に関わらず、MCU で対応しているすべての変換結果格納レジスタを読み出します。

Reentrant

この関数は、再入可能（リエントラント）です。

Example

```
adc_data_t data;
    :
   /* ハードウェアで使用可能なすべてのチャネルレジスタを読み込む */
   R_ADC_ReadAll(&data);       // “data”にすべての変換レジスタの値が入る
```

Special Notes:

なし

3.6 R_ADC_Close()

処理中の A/D 変換を終了し、割り込みを無効にして、A/D コンバータを終了します。

Format

```
adc_err_t R_ADC_Close(uint8_t unit);
```

Parameters

unit

“0”または“1”を設定します。ユニットを 1 つしか持たない MCU では、“0”を設定してください。RX64M/RX71M/RX65x のみがユニットを 2 つ持ちます。

Return Values

ADC_ERR_INVALID_ARG　　/ “unit”の値が “0”または “1”ではありません。*/*

Properties

r_s12ad_rx_if.h にプロトタイプ宣言されています。

Description

処理中の A/D 変換を終了し、割り込みを無効にして、A/D コンバータを終了します。A/D 変換の設定を変更する場合は、本関数を呼び出した後に、再度 R_ADC_Open()関数を呼び出してください。

Reentrant

本関数は、R_ADC_Open()関数実行後、ユニットごとに 1 度だけ呼び出すことができます。

Example

```
:
   err = R_ADC_Open(1, ADC_MODE_SS_MULTI_CH_GROUPED, &config, MyCallback);
    :
   R_ADC_Close(1);
```

Special Notes:

本関数は処理中の A/D 変換を中止します。

3.7 R_ADC_GetVersion ()

この関数は本 FIT モジュールのバージョン番号を返します。

Format

```
uint32_t R_ADC_GetVersion(void)
```

Parameters

なし

Return Values

バージョン番号

Properties

r_s12ad_rx_if.h にプロトタイプ宣言されています。

Description

この関数は本 FIT モジュールのバージョン番号を返します。バージョン番号は符号化され、最上位の 2 バイトがメジャーバージョン番号を、最下位の 2 バイトがマイナーバージョン番号を示しています。

Reentrant

この関数は、再入可能（リエントラント）です。

Example

```
uint32_t version;
      :
version = R_ADC_GetVersion();
```

Special Notes:

この関数は“#pragma inline”を使用してインライン化されています。

4. 端子設定

ADC FIT モジュールを使用するためには、マルチファンクションピンコントローラ（MPC）で周辺機能の入出力信号を端子に割り付ける（以下、端子設定と称す）必要があります。端子設定は、R_ADC_Open 関数を呼び出した後に行ってください。

e^2 studio の場合は「FIT Configurator」または「Smart Configurator」の端子設定機能を使用することができます。FIT Configurator、Smart Configurator の端子設定機能を使用すると、端子設定画面で選択したオプションに応じて、ソースファイルが出力されます。そのソースファイルで定義された関数を呼び出すことにより端子を設定できます。詳細は表 4.1 を参照してください。

表 4.1　FIT コンフィグレータが出力する関数一覧

使用マイコン	選択したオプション	出力される関数名	備考
RX64M RX71M RX65N	ユニット 0	R_ADC_PinSet_S12AD0()	
	ユニット 1	R_ADC_PinSet_S12AD1()	
RX110 RX111 RX113 RX130 RX210 RX230 RX231 RX63x	ユニット 0	R_ADC_PinSet_S12AD0()	

5. デモプロジェクト

デモプロジェクトはスタンドアロンプログラムです。デモプロジェクトには、FIT モジュールとそのモジュールが依存するモジュール（例 : r_bsp）を使用する main()関数が含まれます。デモプロジェクトの標準的な命名規則は、<module>_demo_<board>となり、<module>は周辺の略語(例: s12ad、CMT、SCI）、<board>は標準 RSK（例: rskrx113）です。例えば、RSKRX113 用の s12ad FIT モジュールのデモプロジェクトは s12ad_demo_rskrx113 となります。同様にエクスポートされた.zip ファイルは <module>_demo_<board>.zip となります。例えば、zip 形式のエクスポート/インポートされたファイルは s12ad_demo_rskrx113.zip となります。

5.1 s12ad_int_demo_rskrx113

本デモは MTU0 の周期割り込みを使って、定期的に A/D 変換を行います。A/D 変換完了時、デモプログラムでは、コールバック関数の割り込みで変換値を読み出し、グローバル変数 “data”に A/D 変換結果を格納します。プログラム実行後、ボリュームを調整し A/D 入力チャネルの電圧を変化させ、エミュレータ上で”data”を確認してください。

5.2 s12ad_poll_demo_rskrx113

本デモは、無限ループを使用し、ソフトウェアトリガによって A/D 変換を行います。A/D 変換完了時、デモプログラムでは、アプリケーションで変換値を読み出し、グローバル変数 “data”に A/D 変換結果を格納します。プログラム実行後、ボリュームを調整し A/D 入力チャネルの電圧を変化させ、エミュレータ上で“data”を確認してください。

5.3 s12ad_poll_demo_rskrx130

本デモは、無限ループを使用し、ソフトウェアトリガによって A/D 変換を行います。A/D 変換完了時、デモプログラムでは、アプリケーションで変換値を読み出し、グローバル変数 “data”に A/D 変換結果を格納します。プログラム実行後、ボリュームを調整し A/D 入力チャネルの電圧を変化させ、エミュレータ上で“data”を確認してください。

5.4 s12ad_demo_rskrx64m

RSKRX64M（FIT モジュール “r_s12ad_rx”）向けの RX64M A/D コンバータ（S12AD）のシンプルなデモです。デモではマルチファンクションタイマパルスユニット 3（MTU3a）を使用して、ボリュームに接続されているチャネル 0 で定期的に A/D 変換を行います。A/D 変換完了時、デモプログラムでは、コールバック関数の割り込みで変換値を読み出し、グローバル変数 “g_data”に A/D 変換結果を格納します。プログラム実行後、ボリュームを調整し A/D 入力チャネルの電圧を変化させ、エミュレータ上で”g_data”を確認してください。

5.5 s12ad_demo_rskrx71m

RSKRX71M（FIT モジュール “r_s12ad_rx”）向けの RX71M A/D コンバータ（S12AD）のシンプルなデモです。デモではマルチファンクションタイマパルスユニット 3（MTU3a）を使用して、ボリュームに接続されているチャネル 0 で定期的に A/D 変換を行います。A/D 変換完了時、デモプログラムでは、コールバック関数の割り込みで変換値を読み出し、グローバル変数 “g_data”に A/D 変換結果を格納します。プログラム実行後、ボリュームを調整し A/D 入力チャネルの電圧を変化させ、エミュレータ上で”g_data”を確認してください。

5.6 s12ad_demo_rskrx231

RSKRX231（FIT モジュール “r_s12ad_rx”）向けの RX231 A/D コンバータ（S12AD）のシンプルなデモです。デモではマルチファンクションタイマパルスユニット 2（MTU2a）を使用して、ボリュームに接続されているチャネル 0 で定期的に A/D 変換を行います。A/D 変換完了時、デモプログラムでは、コールバック関数の割り込みで変換値を読み出し、グローバル変数 “g_data”に A/D 変換結果を格納します。プログラム実行後、ボリュームを調整し A/D 入力チャネルの電圧を変化させ、エミュレータ上で”g_data”を確認してください。

5.7 ワークスペースにデモを追加する

デモプロジェクトは、e2 studio のインストールディレクトリ内の FITDemos サブディレクトリにあります。ワークスペースにデモプロジェクトを追加するには、「ファイル」→「インポート」を選択し、「インポート」ダイアログから「一般」の「既存プロジェクトをワークスペースへ」を選択して「次へ」ボタンをクリックします。「インポート」ダイアログで「アーカイブ・ファイルの選択」ラジオボタンを選択し、「参照」ボタンをクリックして FITDemos サブディレクトリを開き、使用するデモの zip ファイルを選択して「完了」をクリックします。

5.8　デモのダウンロード方法

デモプロジェクトは、RX Driver Package には同梱されていません。デモプロジェクトを使用する場合は、個別に各 FIT モジュールをダウンロードする必要があります。「スマートブラウザ」の「アプリケーションノート」タブから、本アプリケーションノートを右クリックして「サンプル・コード（ダウンロード）」を選択することにより、ダウンロードできます。

6. 付録

6.1 動作確認環境

本 FIT モジュールの動作確認環境を以下に示します。

表 6.1 動作確認環境 (Rev.2.30)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e^2 studio V5.4.0 (WS パッチ仕様)
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V2.07.00
	コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev2.30
使用ボード	Renesas Starter Kit+ for RX65N-2MB (型名：RTK50565N2SxxxxxBE) Renesas Starter Kit for RX130-512KB (型名：RTK5051308SxxxxxBE)

6.2　トラブルシューティング

(1) Q : 本 FIT モジュールをプロジェクトに追加しましたが、ビルド実行すると「Could not open source file "platform.h"」エラーが発生します。

A : FIT モジュールがプロジェクトに正しく追加されていない可能性があります。プロジェクトへの追加方法をご確認ください。

- CS+を使用している場合
  アプリケーションノート RX ファミリ CS+に組み込む方法 Firmware Integration Technology (R01AN1826)」
- e^2 studio を使用している場合
  アプリケーションノート RX ファミリ e^2 studio に組み込む方法 Firmware Integration Technology (R01AN1723)」

また、本 FIT モジュールを使用する場合、ボードサポートパッケージ FIT モジュール(BSP モジュール)もプロジェクトに追加する必要があります。BSP モジュールの追加方法は、アプリケーションノート「ボードサポートパッケージモジュール(R01AN1685)」を参照してください。

(2) Q : 本 FIT モジュールをプロジェクトに追加しましたが、ビルド実行すると「This MCU is not supported by the current r_s12ad_rx module.」エラーが発生します。

A : 追加した FIT モジュールがユーザプロジェクトのターゲットデバイスに対応していない可能性があります。追加した FIT モジュールの対象デバイスを確認してください。

(3) Q : アナログ入力端子に入力した電圧と A/D 変換結果が一致しません。

A : 正しく端子設定が行われていない可能性があります。本 FIT モジュールを使用する場合は端子設定が必要です。詳細は「4 端子設定」を参照してください。

テクニカルアップデートの対応について

本モジュールは以下のテクニカルアップデートの内容を反映しています。

- TN-RX*-A124A/J
- TN-RX*-A117A/J

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
2.20	2016.12.1	—	初版発行
		ﾌﾟﾛｸﾞﾗﾑ	コメント行の誤字を修正しました。
			R_ADC_Open 関数の初期化処理を見直しました。
			以下の不具合を修正しました。 ■対象デバイス RX64M/RX71M/RX230/RX231 ■内容 引数の範囲チェックに誤りがあるため、グループ B のトリガにトリガ要因非選択状態を設定すると R_ADC_Open 関数がエラーを返します。 ■発生条件 R_ADC_Open 関数の引数を以下の組み合わせに設定した場合に発生します。 第 2 引数(mode) ADC_MODE_SS_MULTI_CH_GROUPED、または ADC_MODE_SS_MULTI_CH_GROUPED_DBLTRIG_A 第 3 引数(p_cfg->trigger_groupb) ADC_TRIG_NONE_GROUPB ■対策 adc_check_open_cfg 関数の引数チェックを修正しました。 Rev2.20 以降の ADC FIT モジュールを使用してください。
			以下の不具合を修正しました。 ■対象デバイス RX230/RX231 ■内容 コンペアウィンドウ A 動作許可ビットを有効に設定していないため、コンペア機能(レベル比較、およびウィンドウ比較)が動作しません。 ■発生条件 条件に関わらず、コンペア機能は動作しません。 ■対策 コンペア機能を選択された場合、adc_control 関数で CMPAE ビットを　有効にするように修正しました。 Rev2.20 以降の ADC FIT モジュールを使用してください。

Rev.	発行日	改訂内容	
		ページ	ポイント
2.20	2016.12.1	ﾌﾟﾛｸﾞﾗﾑ	以下の不具合を修正しました。 ■対象デバイス RX64M/RX71M/RX230/RX231 ■内容 断線検出アシスト機能設定後、レジスタ初期化が無いため断線検出アシスト機能の設定がレジスタにそのまま残り、自己診断を設定する時に組み合わせ異常で R_ADC_Control 関数がエラーを返します。 ■発生条件 断線検出アシスト機能設定後に FIT モジュールをクローズし、再度オープンした後に自己診断を設定した場合に発生します。 ■対策 adc_open 関数に S12AD の全レジスタを初期化する処理を追加し、adc_check_scan_config 関数の自己診断設定時のエラーチェックから断線検出アシスト機能動作中のチェックを削除しました。 Rev2.20 以降の ADC FIT モジュールを使用してください。
			以下の不具合を修正しました。 ■対象デバイス RX230/RX231 ■内容 レジスタテーブルを修飾するための引数(enum 値)の数がレジスタテーブルと合っていないため、レジスタテーブルの範囲外を修飾してしまい R_ADC_Read 関数で自己診断結果が正常に取得できません。 ■発生条件 条件に関わらず、常に発生します。 ■対策 レジスタテーブルの修飾に使う列挙型(adc_reg_t)から不要な定義を削除しました。 Rev2.20 以降の ADC FIT モジュールを使用してください。
			以下の不具合を修正しました。 ■対象デバイス RX210 ■内容 コンパイルに必要な定数が Rev2.10 で削除されているため、RX210 を使用した場合にビルドエラーが発生します。 ■発生条件 Rev2.10、または Rev2.11 の ADC FIT モジュールを組み込んだプロジェクトをビルドした場合に発生します。 ■対策 r_s12ad_rx_config.h に ADC_CFG_PGA_GAIN を追加しました。 Rev2.20 以降の ADC FIT モジュールを使用してください。
			不要な define 定義を削除しました。
			不要なメンバ変数を削除しました。
			A/D 変換停止時の処理手順をハードウェアマニュアル記載の方法に合わせました。
			低消費電力状態への遷移時の処理手順をハードウェアマニュアル記載の方法に合わせました。
			ADHSC ビットの書き換え手準をハードウェアマニュアル記載の方法に合わせました。

Rev.	発行日	改訂内容	
		ページ	ポイント
2.20	2016.12.1	ﾌﾟﾛｸﾞﾗﾑ	以下の不具合を修正しました。 ■対象デバイス RX64M/RX71M/RX230/RX231 ■内容 上限電圧を下限電圧未満に設定しないように制限している処理の等号記号を間違えていたため、コンペア機能設定時に上限電圧と下限電圧を同じ電圧に設定できません。 ■発生条件 コンペア機能(ウィンドウ比較)を使用した場合に発生します。 ■対策 Rev2.20 以降の ADC FIT モジュールを使用してください。
			RX64M/RX71M の温度センサ変換時にディレイ時間を適切に設定するように修正しました。
			RX64M/RX71M の拡張アナログ入力使用時に無効チャネルをチェックする処理を修正しました。
			各チップ間で意味は同じだが定義名が異なっている定数があるため共通化しました。 <u>RX63x</u> ADC_TRIG_ASYNC_ADTRG0 → ADC_TRIG_ASYNC_ADTRG ADC_TRIG_SYNC_TRG0AN_0 → ADC_TRIG_SYNC_TRG0AN ADC_TRIG_SYNC_TRG0BN_0 → ADC_TRIG_SYNC_TRG0BN ADC_TRIG_SYNC_TRGAN_0 → ADC_TRIG_SYNC_TRGAN ADC_TRIG_SYNC_TRGAN_1 → ADC_TRIG_SYNC_TPUTRGAN ADC_TRIG_SYNC_TRG0EN_0 → ADC_TRIG_SYNC_TRG0EN ADC_TRIG_SYNC_TRG0FN_0 → ADC_TRIG_SYNC_TRG0FN ADC_TRIG_SYNC_TRG4ABN_0 → ADC_TRIG_SYNC_TRG4AN_OR_TRG4BN ADC_TRIG_SYNC_TRG4ABN_1 → ADC_TRIG_SYNC_TPUTRG0AN ADC_TRIG_SYNC_TMRTRG0AN_0 → ADC_TRIG_SYNC_TMRTRG0AN ADC_TRIG_SYNC_TMRTRG0AN_1 → ADC_TRIG_SYNC_TMRTRG2AN <u>RX110</u> ADC_CONVERT_SPEED_HI → ADC_CONVERT_SPEED_HIGH ADC_TRIG_NONE_GROUPB → 削除 ADC_TRIG_ASYNC_ADTRG0 → ADC_TRIG_ASYNC_ADTRG <u>RX111</u> ADC_CONVERT_SPEED_HI → ADC_CONVERT_SPEED_HIGH ADC_TRIG_NONE_GROUPB → 削除 ADC_TRIG_ASYNC_ADTRG0 → ADC_TRIG_ASYNC_ADTRG ADC_TRIG_SYNC_TRGAN → ADC_TRIG_SYNC_TRGAN_OR_UDF4N ADC_TRIG_SYNC_TRG4ABN → ADC_TRIG_SYNC_TRG4AN_AND_TRG4BN <u>RX113</u> ADC_CONVERT_SPEED_HI → ADC_CONVERT_SPEED_HIGH ADC_TRIG_NONE_GROUPB → 削除 ADC_TRIG_ASYNC_ADTRG0 → ADC_TRIG_ASYNC_ADTRG ADC_TRIG_SYNC_TRGAN → ADC_TRIG_SYNC_TRGAN_OR_UDF4N ADC_TRIG_SYNC_TRG4ABN → ADC_TRIG_SYNC_TRG4AN_AND_TRG4BN

Rev.	発行日	改訂内容	
		ページ	ポイント
2.20	2016.12.1	ﾌﾟﾛｸﾞﾗﾑ	RX210 ADC_TRIG_NONE_GROUPB → 削除 ADC_TRIG_ASYNC_ADTRG0 → ADC_TRIG_ASYNC_ADTRG ADC_TRIG_SYNC_TRGAN → ADC_TRIG_SYNC_TRGAN_OR_UDF4N ADC_TRIG_SYNC_TRG4ABN → ADC_TRIG_SYNC_TRG4AN_AND_TRG4BN ADC_TRIG_PLACEHOLDER → ADC_TRIG_SYNC_TEMPS ADC_TRIG_SYNC_TRGAN1 → ADC_TRIG_SYNC_TPUTRGAN ADC_TRIG_SYNC_TRG4ABN1 → ADC_TRIG_SYNC_TPUTRG0AN RX64M ADC_CMD_CONFIGURE_SCAN → ADC_CMD_ENABLE_CHANS ADC_TRIG_NONE_GROUPB → ADC_TRIG_NONE ADC_TRIG_ASYNC_ADTRG0 → ADC_TRIG_ASYNC_ADTRG ADC_TRIG_SYNC_TRGA0N → ADC_TRIG_SYNC_TRG0AN ADC_TRIG_SYNC_TRGA1N → ADC_TRIG_SYNC_TRG1AN ADC_TRIG_SYNC_TRGA2N → ADC_TRIG_SYNC_TRG2AN ADC_TRIG_SYNC_TRGA3N → ADC_TRIG_SYNC_TRG3AN ADC_TRIG_SYNC_TRGA4N → ADC_TRIG_SYNC_TRG4AN_OR_UDF4N ADC_TRIG_SYNC_TRGA6N → ADC_TRIG_SYNC_TRG6AN ADC_TRIG_SYNC_TRGA7N → ADC_TRIG_SYNC_TRG7AN_OR_UDF7N ADC_TRIG_SYNC_TRG0N → ADC_TRIG_SYNC_TRG0EN ADC_TRIG_SYNC_TRG4ABN → ADC_TRIG_SYNC_TRG4AN_AND_TRG4BN ADC_TRIG_SYNC_TRG7ABN → ADC_TRIG_SYNC_TRG7AN_AND_TRG7BN ADC_TRIG_SYNC_GTADTRA0N → ADC_TRIG_SYNC_GTADTR0AN ADC_TRIG_SYNC_GTADTRB0N → ADC_TRIG_SYNC_GTADTR0BN ADC_TRIG_SYNC_GTADTRA1N → ADC_TRIG_SYNC_GTADTR1AN ADC_TRIG_SYNC_GTADTRB1N → ADC_TRIG_SYNC_GTADTR1BN ADC_TRIG_SYNC_GTADTRA2N → ADC_TRIG_SYNC_GTADTR2AN ADC_TRIG_SYNC_GTADTRB2N → ADC_TRIG_SYNC_GTADTR2BN ADC_TRIG_SYNC_GTADTRA3N → ADC_TRIG_SYNC_GTADTR3AN ADC_TRIG_SYNC_GTADTRB3N → ADC_TRIG_SYNC_GTADTR3BN ADC_TRIG_SYNC_GTADTRA0N_OR_GTADTRB0N → ADC_TRIG_SYNC_GTADTR0AN_OR_GTADTR0BN ADC_TRIG_SYNC_GTADTRA1N_OR_GTADTRB1N → ADC_TRIG_SYNC_GTADTR1AN_OR_GTADTR1BN ADC_TRIG_SYNC_GTADTRA2N_OR_GTADTRB2N → ADC_TRIG_SYNC_GTADTR2AN_OR_GTADTR2BN ADC_TRIG_SYNC_GTADTRA3N_OR_GTADTRB3N → ADC_TRIG_SYNC_GTADTR3AN_OR_GTADTR3BN ADC_TRIG_SYNC_TMTRG0AN_0 → ADC_TRIG_SYNC_TMRTRG0AN ADC_TRIG_SYNC_TMTRG0AN_1 → ADC_TRIG_SYNC_TMRTRG2AN ADC_TRIG_SYNC_TPTRGAN → ADC_TRIG_SYNC_TPUTRGAN ADC_TRIG_SYNC_TPTRG0AN → ADC_TRIG_SYNC_TPUTRG0AN ADC_TRIG_SYNC_ELCTRG → ADC_TRIG_SYNC_ELC

Rev.	発行日	改訂内容	
		ページ	ポイント
2.20	2016.12.1	ﾌﾟﾛｸﾞﾗﾑ	RX71M ADC_CMD_CONFIGURE_SCAN → ADC_CMD_ENABLE_CHANS ADC_TRIG_NONE_GROUPB → ADC_TRIG_NONE ADC_TRIG_ASYNC_ADTRG0 → ADC_TRIG_ASYNC_ADTRG ADC_TRIG_SYNC_TRGA0N → ADC_TRIG_SYNC_TRG0AN ADC_TRIG_SYNC_TRGA1N → ADC_TRIG_SYNC_TRG1AN ADC_TRIG_SYNC_TRGA2N → ADC_TRIG_SYNC_TRG2AN ADC_TRIG_SYNC_TRGA3N → ADC_TRIG_SYNC_TRG3AN ADC_TRIG_SYNC_TRGA4N → ADC_TRIG_SYNC_TRG4AN_OR_UDF4N ADC_TRIG_SYNC_TRGA6N → ADC_TRIG_SYNC_TRG6AN ADC_TRIG_SYNC_TRGA7N → ADC_TRIG_SYNC_TRG7AN_OR_UDF7N ADC_TRIG_SYNC_TRG0N → ADC_TRIG_SYNC_TRG0EN ADC_TRIG_SYNC_TRG4ABN → ADC_TRIG_SYNC_TRG4AN_AND_TRG4BN ADC_TRIG_SYNC_TRG7ABN → ADC_TRIG_SYNC_TRG7AN_AND_TRG7BN ADC_TRIG_SYNC_GTADTRA0N → ADC_TRIG_SYNC_GTADTR0AN ADC_TRIG_SYNC_GTADTRB0N → ADC_TRIG_SYNC_GTADTR0BN ADC_TRIG_SYNC_GTADTRA1N → ADC_TRIG_SYNC_GTADTR1AN ADC_TRIG_SYNC_GTADTRB1N → ADC_TRIG_SYNC_GTADTR1BN ADC_TRIG_SYNC_GTADTRA2N → ADC_TRIG_SYNC_GTADTR2AN ADC_TRIG_SYNC_GTADTRB2N → ADC_TRIG_SYNC_GTADTR2BN ADC_TRIG_SYNC_GTADTRA3N → ADC_TRIG_SYNC_GTADTR3AN ADC_TRIG_SYNC_GTADTRB3N → ADC_TRIG_SYNC_GTADTR3BN ADC_TRIG_SYNC_GTADTRA0N_OR_GTADTRB0N → ADC_TRIG_SYNC_GTADTR0AN_OR_GTADTR0BN ADC_TRIG_SYNC_GTADTRA1N_OR_GTADTRB1N → ADC_TRIG_SYNC_GTADTR1AN_OR_GTADTR1BN ADC_TRIG_SYNC_GTADTRA2N_OR_GTADTRB2N → ADC_TRIG_SYNC_GTADTR2AN_OR_GTADTR2BN ADC_TRIG_SYNC_GTADTRA3N_OR_GTADTRB3N → ADC_TRIG_SYNC_GTADTR3AN_OR_GTADTR3BN ADC_TRIG_SYNC_TMTRG0AN_0 → ADC_TRIG_SYNC_TMRTRG0AN ADC_TRIG_SYNC_TMTRG0AN_1 → ADC_TRIG_SYNC_TMRTRG2AN ADC_TRIG_SYNC_TPTRGAN → ADC_TRIG_SYNC_TPUTRGAN ADC_TRIG_SYNC_TPTRG0AN → ADC_TRIG_SYNC_TPUTRG0AN ADC_TRIG_SYNC_ELCTRG → ADC_TRIG_SYNC_ELC RX130 ADC_TRIG_NONE_GROUPB → ADC_TRIG_NONE ADC_TRIG_ASYNC_ADTRG0 → ADC_TRIG_ASYNC_ADTRG ADC_TRIG_SYNC_TRGAN → ADC_TRIG_SYNC_TRGAN_OR_UDF4N ADC_TRIG_SYNC_TRG4ABN → ADC_TRIG_SYNC_TRG4AN_AND_TRG4BN ADC_TRIG_SYNC_ELCTRG0 → ADC_TRIG_SYNC_ELC

Rev.	発行日	改訂内容	
		ページ	ポイント
2.20	2016.12.1	ﾌﾟﾛｸﾞﾗﾑ	RX230 ADC_TRIG_NONE_GROUPB → ADC_TRIG_NONE ADC_TRIG_ASYNC_ADTRG0 → ADC_TRIG_ASYNC_ADTRG ADC_TRIG_SYNC_TRGAN → ADC_TRIG_SYNC_TRGAN_OR_UDF4N ADC_TRIG_SYNC_TRG4ABN → ADC_TRIG_SYNC_TRG4AN_AND_TRG4BN ADC_TRIG_SYNC_ELCTRG0N_OR_ELCTRG1N → ADC_TRIG_SYNC_ELC ADC_TRIG_SYNC_TRGAN1 → ADC_TRIG_SYNC_TPUTRGAN ADC_TRIG_SYNC_TRG4ABN1 → ADC_TRIG_SYNC_TPUTRG0AN RX231 ADC_TRIG_NONE_GROUPB → ADC_TRIG_NONE ADC_TRIG_ASYNC_ADTRG0 → ADC_TRIG_ASYNC_ADTRG ADC_TRIG_SYNC_TRGAN → ADC_TRIG_SYNC_TRGAN_OR_UDF4N ADC_TRIG_SYNC_TRG4ABN → ADC_TRIG_SYNC_TRG4AN_AND_TRG4BN ADC_TRIG_SYNC_ELCTRG0N_OR_ELCTRG1N → ADC_TRIG_SYNC_ELC ADC_TRIG_SYNC_TRGAN1 → ADC_TRIG_SYNC_TPUTRGAN ADC_TRIG_SYNC_TRG4ABN1 → ADC_TRIG_SYNC_TPUTRG0AN
			adc_ch_cfg_t 構造体のメンバ名がチップ間で異なっているため共通化しました。 RX64M/RX71M scan_mask → chan_mask scan_mask_groupb → chan_mask_groupb
			チェック処理簡略化のため enum 値による範囲チェック処理を削除しました。 ※enum 値の範囲外はコンパイル時のワーニングで判断してください。
			以下の不具合を修正しました。 ■対象デバイス RX210 ■内容 引数チェックをする時に ADC_TRIG_SYNC_TEMPS が有効かどうかのチェックを trigger_groupb ではなく trigger でチェックしているため有効な設定をしているにも関わらず R_ADC_Open 関数がエラーを返します。 ■発生条件 A/D 変換のトリガ要因に ADC_TRIG_SYNC_TEMPS を設定した場合に発生します。 ■対策 adc_open 関数から ADC_TRIG_SYNC_TEMPS のチェックを削除しました。 ※trigger_groupb はグループスキャンモード以外の場合は無視され、グループスキャンモードの場合は trigger_groupb に ADC_TRIG_SYNC_TEMPS が設定されているとエラーを返すため、引数チェックでの判断は不要です。 Rev2.20 以降の ADC FIT モジュールを使用してください。

Rev.	発行日	改訂内容	
		ページ	ポイント
2.20	2016.12.1	ﾌﾟﾛｸﾞﾗﾑ	R_ADC_ReadAll 関数の振る舞いを全チップで統一するため、RX63x/RX110/RX111/RX113/RX210 の adc_data_t 構造体に温度センサ(temp)と内部基準電圧(volt)を追加しました。
			以下の不具合を修正しました。 ■対象デバイス RX64M/RX71M ■内容 引数チェックをする時に ADC_TRIG_NONE が有効かどうかのチェックを trigger でチェックしているため､有効な設定をしているにも関わらず R_ADC_Open 関数がエラーを返します。 ■発生条件 A/D 変換のトリガ要因に ADC_TRIG_NONE を設定した場合に発生します。 ■対策 ADC_TRIG_NONE は TRSA、TRSB の何れのレジスタにも設定可能なため、adc_open 関数から ADC_TRIG_NONE をチェックする処理を削除しました。 Rev2.20 以降の ADC FIT モジュールを使用してください。
			RX130/RX230/RX231/RX64M/RX71M でグループスキャンモード以外を設定した時に ADGSPCR レジスタを初期化するように修正しました。
			RX130/RX230/RX231 のコンペア機能の引数の構造体を RX65N に合わせました。adc_cmplvl_t 構造体は廃止になりましたので、コンペア機能のレベル比較を使用する場合は adc_cmpwin_t 構造体を使用してください。
			以下の不具合を修正しました。 ■対象デバイス RX130/RX230/RX231 ■内容 コンペアウィンドウ動作許可ビットを停止に設定する処理がないため、一度コンペア機能を設定すると再オープンする以外にコンペア機能を停止することができません。 また、RX230/RX231 は再オープンしても停止することができません。 ■発生条件 コンペア機能を使用すると必ず発生します。 ■対策 コンペア機能の引数の構造体に windowa_enable を追加し、windowa_enable の true/false によってコンペアウィンドウ動作許可ビットを許可/停止するようにしました(RX65N と同等の処理)。 Rev2.20 以降の ADC FIT モジュールを使用してください。

Rev.	発行日	改訂内容	
		ページ	ポイント
2.20	2016.12.1	ﾌﾟﾛｸﾞﾗﾑ	以下の不具合を修正しました。 ■対象デバイス RX64M/RX71M ■内容 WCMPE ビットを"0"(レベル比較)に設定する箇所が無いため、一度ウィンドウ比較に設定するとレベル比較を設定することができなくなります。 ■発生条件 コンペア機能をウィンドウ比較に設定した後、レベル比較に再設定した場合に発生します。 ■対策 ウィンドウ比較､およびレベル比較を選択した場合に WCMPE ビットを適切に設定するように修正しました。 Rev2.20 以降の ADC FIT モジュールを使用してください。
			RX64M/RX71M のコンペア割り込みを使用する際、BSP の提供するインタフェース(R_BSP_InterruptControl 関数)を使用して割り込み許可ビット、割り込み優先順位の設定を行うように変更しました。
			以下の不具合を修正しました。 ■対象デバイス RX64M/RX71M ■内容 コンペア割り込みイネーブルビットを停止に設定する処理がないため、一度コンペア機能を設定するとコンペア割り込みを禁止することができません。 ■発生条件 コンペア機能設定時に割り込み優先順位を"1"以上に設定すると発生します。 ■対策 adc_close 関数実行時にコンペア割り込みを禁止し、他にグループ割り込みを使っている FIT モジュールが無ければ、グループ割り込みを禁止するように修正しました。 Rev2.20 以降の ADC FIT モジュールを使用してください。
			以下の不具合を修正しました。 ■対象デバイス RX130/RX230/RX231/RX64M/RX71M ■内容 未設定(NULL)のコールバック関数を実行し､不正割り込みが発生します。 ■発生条件 割り込み禁止でオープンした後､コンペア機能の優先順位を"1"以上に設定すると発生します。 ■対策 コールバック関数を実行する前に NULL チェックを行い、コールバック関数未設定の場合には何もせず割り込み処理を終了するように修正しました。 Rev2.20 以降の ADC FIT モジュールを使用してください。

Rev.	発行日	改訂内容	
		ページ	ポイント
2.20	2016.12.1	ﾌﾟﾛｸﾞﾗﾑ	RX210 の温度センサ出力を有効にする際、0 に初期化済みのレジスタを再度 0 に初期化していたため、不要な初期化を削除しました。
			RX113 で独自の時間待ち関数(adc_delay)を使っていたため、BSP 提供の時間待ち関数(R_BSP_SoftwareDelay)に置き換えました。 ※独自の時間待ち関数(adc_delay)は削除しました。
			以下の不具合を修正しました。 ■対象デバイス RX210 ■内容 不要なエラー判定処理を行っているため、チャネル専用サンプル＆ホールドをありに設定すると、R_ADC_Control 関数がエラーを返します。 ■発生条件 グループスキャンモード、かつグループ A とグループ B の A/D 変換チャネルを、両方サンプル＆ホールドありにすると発生します。 ■対策 HWM には該当する制限事項がないため、不要なエラー判定処理を削除しました。 Rev2.20 以降の ADC FIT モジュールを使用してください。
			以下の不具合を修正しました。 ■対象デバイス RX210 ■内容 エラー判定処理が抜けているため、自己診断が動作しないモードで自己診断を設定しても R_ADC_Control 関数がエラーを返しません。 ■発生条件 シングルスキャン、かつダブルトリガ、もしくはグループスキャン、かつダブルトリガの時に自己診断を設定すると発生します。 ■対策 自己診断が設定された場合のエラー判定処理を追加しました。 Rev2.20 以降の ADC FIT モジュールを使用してください。
			以下の不具合を修正しました。 ■対象デバイス RX130/RX230/RX231 ■内容 不要なエラー判定処理を行っているため、自己診断設定後に断線検出アシスト機能を設定しようとすると R_ADC_Control 関数がエラーを返します。 ■発生条件 自己診断を設定した後に断線検出アシスト機能をディスチャージ、もしくはプリチャージに設定すると発生します。 ■対策 断線検出アシスト機能設定時に行っていた不要な判断処理を削除しました。 Rev2.20 以降の ADC FIT モジュールを使用してください。

Rev.	発行日	改訂内容	
		ページ	ポイント
2.20	2016.12.1	ﾌﾟﾛｸﾞﾗﾑ	以下の不具合を修正しました。 ■対象デバイス RX63x ■内容 有効チャネルの判別を行うための定義が間違っていたため、チャネル 20 が選択できません。 ■発生条件 177pin、176pin、145pin、144pin のチップを選択した場合に発生します。 ■対策 有効チャネルの判別を行うための定義を修正しました。 Rev2.20 以降の ADC FIT モジュールを使用してください。
			以下の不具合を修正しました。 ■対象デバイス RX631 ■内容 有効チャネルの判別を行うための定義が無いため、コンパイルエラーが発生します。 ■発生条件 64pin、48pin のチップを選択した場合に発生します。 ■対策 有効チャネルの判別を行うための定義を追加しました。 Rev2.20 以降の ADC FIT モジュールを使用してください。
			以下の不具合を修正しました。 ■対象デバイス RX64M/RX71M/RX65x ■内容 コンペア結果取得時にコンペアチャネルをクリアするため、次回からコンペアが行われません。 ■発生条件 ユニット 1 のチャネル 16～チャネル 20 の何れかをコンペアチャネルに指定した時に、条件一致しコンペア割り込みが発生した、もしくは ADC_CMD_CHECK_CONDITION_MET を設定して R_ADC_Control 関数を実行した場合に発生します。 ■対策 コンペア結果取得時に初期化するレジスタを修正しました。 Rev2.20 以降の ADC FIT モジュールを使用してください。
			以下の不具合を修正しました。 ■対象デバイス RX64M/RX71M/RX65x/RX130/RX230/RX231 ■内容 設定禁止条件中に自己診断を設定すると、正常終了します。 ■発生条件 シングルスキャンモード、かつダブルトリガを設定している時に自己診断を設定すると発生します。 ■対策 自己診断設定時のエラー条件チェック処理を修正しました。 Rev2.20 以降の ADC FIT モジュールを使用してください。

Rev.	発行日	改訂内容	
		ページ	ポイント
2.20	2016.12.1	ﾌﾟﾛｸﾞﾗﾑ	RX63x/RX210 で、温度センサモジュールが有効な場合のみ温度センサのレジスタを変更するように変更しました。
			RX110/RX111/RX113 の A/D 変換速度の定義に "ADC_CONVERT_SPEED_DEFAULT"を追加しました。 "ADC_CONVERT_SPEED_DEFAULT"の値は "ADC_CONVERT_SPEED_NORM"と同じになります。
			以下の不具合を修正しました。 ■対象デバイス RX110 ■内容 サンプリングステート数の最小値を設定しようとするとエラーになります。 ■発生条件 サンプリングステート数を設定可能な状態ならば、常時発生します。 ■対策 サンプリングステート数の最小値チェックの定義を修正しました。 Rev2.20 以降の ADC FIT モジュールを使用してください。
			RX64M/RX71M/RX65x/RX130/RX230/RX231 で関数宣言とプロトタイプ宣言が異なっているものがあったので、関数の宣言をプロトタイプ宣言に合わせました。
2.30	2017.7.24	—	説明が適用される対象を MCU ではなく S12AD 周辺ごとに記載。
		—	RX65N-2MB で追加されたパッケージ(177 ピン、176 ピン)に対応。
		—	RX130-512KB で追加されたパッケージ(100 ピン)に対応。
		1	「関連ドキュメント」に以下のドキュメントを追加: Renesas e^2 studio スマート・コンフィグレータ ユーザーガイド (R20AN0451)
		4	1.概要: 内容見直し
		5	2.5 対応ツールチェーン: 新規対応に伴う見直し
		6	2.6 使用する割り込みベクタ: 追加
		8	2.10 コードサイズ プログラム変更に伴うサイズ見直し
		9-46	2.11 API データ構造体: 構造体ごとの説明に変更
		47	2.12 戻り値: コメント見直し
		47	2.13 FIT モジュールの追加方法: 見直し
		48	3.1 概要: 説明見直し
		49-52	3.2 R_ADC_Open: 内容見直し
		53-66	3.3 R_ADC_Control: 内容見直し
		71	4 端子設定: 内容見直し
		73	5.8 デモのダウンロード方法: 追加
		74-75	6.付録: 追加
		ﾌﾟﾛｸﾞﾗﾑ	RX65N で、チェック処理簡略化のため enum 値による範囲チェック処理を削除しました。 ※enum 値の範囲外はコンパイル時のワーニングで判断してください

Rev.	発行日	改訂内容	
		ページ	ポイント
2.30	2017.7.24	ﾌﾟﾛｸﾞﾗﾑ	以下の不具合を修正しました。 ■対象デバイス RX130/RX230/RX231/RX64M/RX71M/RX65N ■内容 グループスキャンモード以外でオープンしたチャネルに、グループスキャンモードの時のみ有効なパラメータを設定した場合、エラーなりません。 ■発生条件 グループスキャンモード以外の時に、グループ B のチャネル、グループ C のチャネル(RX65N のみ)、グループ優先制御を設定すると発生します。 ■対策 グループスキャンモード時の無効な組み合わせのチェック処理を修正し、エラーを返すようにしました。 Rev2.30 以降の ADC FIT モジュールを使用してください。
			以下の不具合を修正しました。 ■対象デバイス RX130/RX230/RX231/RX64M/RX71M/RX65N ■内容 グループ優先制御のレジスタ設定手順がハードウェアマニュアルの手順を守れていないため、スキャンの動作および格納されるデータが保証されません。 ■発生条件 グループ優先制御を設定すると発生します。 ■対策 グループ優先制御のレジスタ設定手順を修正しました。 Rev2.30 以降の ADC FIT モジュールを使用してください。
			以下の不具合を修正しました。 ■対象デバイス RX65N ■内容 コールバック関数を設定せずに割り込み優先レベルを設定(割り込みを有効)した場合、エラーになりません。 ■発生条件 割り込み優先レベルを 1 以上に設定すると発生します。 ■対策 オープン時のチェック処理を修正し、エラーを返すようにしました。 Rev2.30 以降の ADC FIT モジュールを使用してください。

Rev.	発行日	改訂内容	
		ページ	ポイント
2.30	2017.7.24	ﾌﾟﾛｸﾞﾗﾑ	以下の不具合を修正しました。 ■対象デバイス RX65N ■内容 加算モードで無効な組み合わせを設定した場合エラーになりません。 ■発生条件 加算モードで 16 回サンプルを選択した時に、変換精度を 10 ビット、または 8 ビットに設定すると発生します。 ■対策 オープン時のチェック処理を修正し、エラーを返すようにしました。 Rev2.30 以降の ADC FIT モジュールを使用してください。
			以下の不具合を修正しました。 ■対象デバイス RX65N ■内容 A/D 変換停止時の手順がハードウェアマニュアルの手順を守れていないため、意図しない動作をする可能性があります。 ■発生条件 グループ優先制御を有効にした状態でクローズを行うと発生します。 ■対策 クローズ時のレジスタ設定手順を修正しました。 Rev2.30 以降の ADC FIT モジュールを使用してください。
			以下の不具合を修正しました。 ■対象デバイス RX65N ■内容 コンペア機能ウィンドウ B の比較条件が正しく設定出来ない可能性があります。 ■発生条件 コンペア機能設定時にウィンドウ B の比較条件を"2"以上に設定した場合に発生します。 ■対策 ウィンドウ B の比較条件に範囲チェックが無かったため、範囲外であればエラーを返すように修正しました。 Rev2.30 以降の ADC FIT モジュールを使用してください。

Rev.	発行日	改訂内容	
		ページ	ポイント
2.30	2017.7.24	ﾌﾟﾛｸﾞﾗﾑ	以下の不具合を修正しました。 ■対象デバイス RX65N ■内容 グループ A のトリガを外部トリガに設定する事ができません。 ■発生条件 グループスキャンモード、かつダブルトリガ無効の場合に発生します。 ■対策 RX64M と同等の実装になっていたため外部トリガの設定が許可されていなかったので、RX65N の場合はグループ A のみ外部トリガを設定可能に修正しました。 Rev2.30 以降の ADC FIT モジュールを使用してください。
			以下の不具合を修正しました。 ■対象デバイス RX65N ■内容 コンペア機能のウィンドウ A/B の複合条件を設定しても結果を知る事が出来ません ■発生条件 常時発生します。 ■対策 ウィンドウ A/B の複合条件の結果を取得するための I/F を R_ADC_Control 関数に追加しました。 Rev2.30 以降の ADC FIT モジュールを使用してください。

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、その他の不適切に使用しないでください。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、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、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。また、当社製品および技術を、(1)核兵器、化学兵器、生物兵器等の大量破壊兵器およびこれらを運搬することができるミサイル（無人航空機を含みます。）の開発、設計、製造、使用もしくは貯蔵等の目的、(2)通常兵器の開発、設計、製造または使用の目的、または(3)その他の国際的な平和および安全の維持の妨げとなる目的で、自ら使用せず、かつ、第三者に使用、販売、譲渡、輸出、賃貸もしくは使用許諾しないでください。

   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. お客様の転売、貸与等により、本書（本ご注意書きを含みます。）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、当社は一切その責任を負わず、お客様にかかる使用に基づく当社への請求につき当社を免責いただきます。
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. 本資料に記載された情報または当社製品に関し、ご不明点がある場合には、当社営業にお問い合わせください。

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。

(Rev.3.0-1　2016.11)

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/