RE01 1500KB、256KB グループ

CMSIS ドライバ S14AD 仕様書

要旨

本書では、RE01 1500KB、256KB グループ CMSIS software package の S14AD ドライバ（以下、S14AD ドライバ）の詳細仕様を説明します。

動作確認デバイス

RE01 1500KB グループ
RE01 256KB グループ

目次

1. 概要

S14AD ドライバは、RE01 1500KB および 256KB グループで 14 ビット A/D コンバータ(注)を使用するためのドライバです。

注　1500KB グループと 256KB グループでは、機能名が異なります。以下に各グループでの周辺機能名を示します。
1500KB グループ：S14AD 機能
256KB グループ：ADC14 機能

2. ドライバ構成

本章では、本ドライバ使用するために必要な情報を記載します。

2.1 ファイル構成

S14AD ドライバは CMSIS Driver Package の HAL_Driver に該当し、ベンダ独自ファイル格納ディレクトリ内の”r_adc_api.c”、 ”r_adc_api.h”、 ”r_adc_cfg.h” 、”pin.c”、”pin.h”の 5 個のファイルで構成されます。各ファイルの役割を表 2-1 に、ファイル構成を図 2-1 に示します。

表 2-1 R_S14AD ドライバ 各ファイルの役割

ファイル名	内容
r_adc_api.c	ドライバソースファイルです ドライバ関数の実体を用意します S14AD ドライバを使用する場合は、本ファイルをビルドする必要があります
r_adc_api.h	ドライバヘッダファイルです ドライバ内で使用するマクロ／型／プロトタイプ宣言が定義されています
r_adc_cfg.h	コンフィグレーション定義ファイルです ユーザが設定可能なコンフィグレーション定義を用意します
pin.c	端子設定ファイルです 各種機能の端子割り当て処理を用意します
pin.h	端子設定ヘッダファイルです

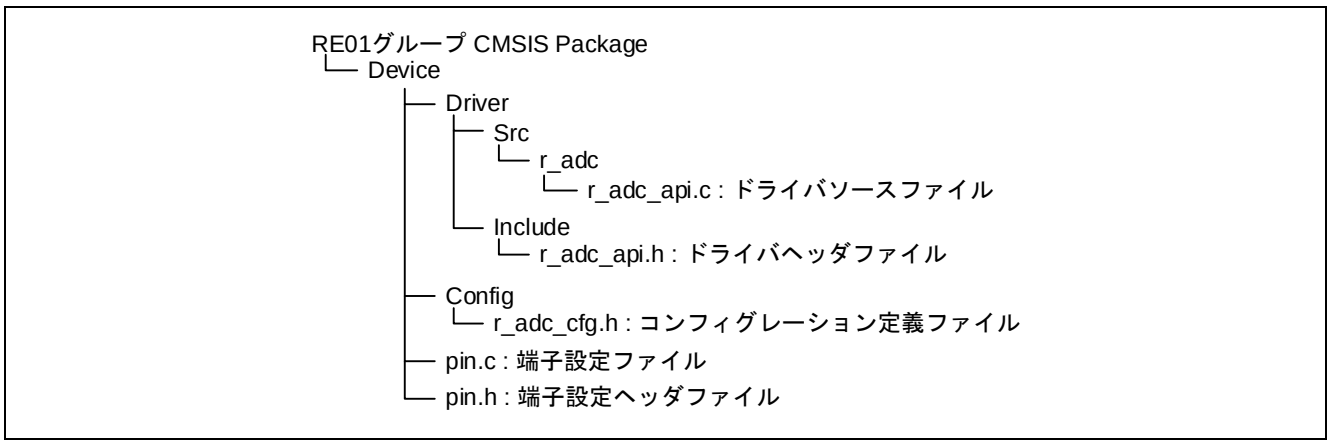


図 2-1 S14AD ドライバファイル構成

2.2 ドライバ API

S14AD ドライバには 1 ユニットのインスタンスがあります。ドライバを使用する場合は、インスタンス内の関数ポインタを使用して API にアクセスしてください。S14AD ドライバのインスタンス一覧を表 2-2 に、インスタンスの宣言例を図 2-2 に、インスタンスに含まれる API を表 2-3 に、S14AD ドライバへのアクセス例を図 2-3～図 2-10 に示します。

表 2-2 S14AD ドライバのインスタンス一覧

インスタンス	内容
Driver_S14AD	S14AD を使用する場合のインスタンス

```
#include "r_adc_api.h"

// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;
```

図 2-2 S14AD ドライバ インスタンス宣言例

表 2-3 S14AD ドライバ API

API	内容	参照
Open	S14AD ドライバの初期化（RAM の初期化、端子設定、モジュールストップ状態の解除）を行います S14AD ドライバで使用するスキャンモード、A/D 変換精度、A/D データレジスタフォーマット、デフォルトサンプリング時間の設定も行います 詳細は「2.4.1 Open 関数による A/D スキャンモード設定」参照	4.1.1
Close	S14AD ドライバを解放（端子解放）します モジュールストップ状態でない場合、モジュールストップ状態への遷移も行います また、DMA 転送による A/D データ自動読み取り機能を使用している場合、DMA ドライバの解放も行います	4.1.2
ScanSet	S14AD で使用するグループごとのスキャン設定（A/D 変換チャネル選択、温度センサ出力 A/D 変換選択、A/D 変換開始条件選択）を行います 詳細は「2.4.2 ScanSet 関数による A/D 入力端子および A/D 変換開始トリガ設定」参照	4.1.3
Start	A/D 変換開始条件にソフトウェアトリガを指定している場合、A/D 変換を開始します(注 1)	4.1.4
Stop	A/D 変換開始条件にソフトウェアトリガを指定している場合、A/D 変換を停止します(注 2)	4.1.5

Control	S14AD の制御コマンドを実行します(注 3)(注 4)
	第 2 引数には制御コマンドに応じた型の変数を用意し、その変数のポインタを指定してください
	各制御コマンドおよび使用方法については、「2.5 Control 関数による S14AD 機能設定」参照
	
	[AD_CMD_SET_AUTO_CLEAR コマンドの使用例]
	uint8_t arg = ADC_ENABLE; //A/D データレジスタ自動クリア有効
	Control(AD_CMD_SET_AUTO_CLEAR, &arg);
	//第 2 引数に変数のポインタを渡す
		4.1.6

Read	S14AD の A/D 変換結果を読み出します	4.1.7
GetVersion	S14AD ドライバのバージョンを取得します	4.1.8

注1. A/D 変換開始条件に非同期トリガまたは同期トリガを選択している場合、各トリガ入力（ELC イベント、TMR イベント、ADTRG 端子入力）で A/D 変換が開始されます。

注2. A/D 変換開始条件に非同期トリガまたは同期トリガを選択している場合、Control 関数制御コマンドの”AD_CMD_STOP_TRIG”で A/D 変換を停止させてください。制御コマンドについては「2.5 Control 関数による S14AD 機能設定」を参照してください。

注3. 機能やモードによっては同時に使用できないものがあります。機能の組み合わせについては「表 2-12 使用機能組み合わせ一覧」を参照してください。

注4. Control 関数による制御コマンドの実行は、Open 関数実行後にのみ有効です。

```
#include "r_adc_api.h"

// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

static void callback(uint32_t event);

main()
{
    st_adc_pins_t scanset_pin;  /* ScanSet()  A/D 変換チャネル */
    st_adc_status_info_t result_status;    /* AD_CMD_GET_AD_STATE 第 2 引数 */
    e_adc_int_method_t adc_int = ADC_INT_ENABLE;    /* 割り込み許可用変数 */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, callback);
                    /* S14AD ドライバ初期化
                    (RAM の初期化、端子設定、モジュールストップ状態の解除) を行います
                     また、引数で指定したモードの設定を行います */
    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                        /* A/D 変換に AN000、AN001、AN006、AN022 を使用 */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* A/D 変換に温度センサを使用しない */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);
                  /* グループ A で使用する端子の指定および A/D 変換開始トリガを設定します */

    (void)adcDev->Control(AD_CMD_SET_ADI_INT, &adc_int); /* スキャン終了割り込み許可 */

    while (1)
    {
        (void)adcDev->Start();  /* A/D 変換開始 */

        do
        {
            (void)adcDev->Control(AD_CMD_GET_AD_STATE, & result_status);    /* A/D の状態を取得 */
        }
        while (ADC_STATE_RUN == result_status.ad_info);    /* A/D 変換完了待ち */
    }
}
/***********************************************************************************************************
* callback function
***********************************************************************************************************/
static void callback(uint32_t event)
{
    switch(event)
    {
        case ADC_EVT_SCAN_COMPLETE:
        {
            /* A/D 変換が終了した場合の処理を記述 */
        }
        break;
        case ADC_EVT_SCAN_COMPLETE_GROUPB:
        case ADC_EVT_SCAN_COMPLETE_GROUPC:
        case ADC_EVT_CONDITION_MET:
        case ADC_EVT_CONDITION_METB:
        case ADC_EVT_WINDOW_CMP_MATCH:
        case ADC_EVT_WINDOW_CMP_UNMATCH:
        default:
        {
            /* 許可されていない割り込みイベントのため発生しません */
        }
        break;
    }
}
```

図 2-3 S14AD ドライバへのアクセス例（ソフトウェアトリガ）

```
#include "r_adc_api.h"

static void callback(uint32_t event);

// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

main()
{
    st_adc_pins_t groupa_pin;  /* グループ A の A/D 変換チャネル */
    st_adc_pins_t groupb_pin;  /* グループ B の A/D 変換チャネル */
    st_adc_pins_t groupc_pin;  /* グループ C の A/D 変換チャネル */
    e_adc_int_method_t adc_int = ADC_INT_ENABLE;    /* 割り込み許可用変数 */

    (void)adcDev->Open(ADC_GROUP_SCAN | ADC_14BIT, 0x10, callback);
                    /* S14AD ドライバ初期化
                    （RAM の初期化、端子設定、モジュールストップ状態の解除）を行います
                     また、引数で指定したモードの設定を行います */
    groupa_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01;    /* グループ A に AN00 と AN01 を指定 */
    groupa_pin.sensor   = ADC_SENSOR_NOTUSE; /* 温度センサ未使用 */
    (void)adcDev->ScanSet(ADC_GROUP_A, groupa_pin, ADC_TRIGER_ADTRG);
                /* グループ A で使用する端子の指定および A/D 変換開始トリガを設定します */

    groupb_pin.an_chans = ADC_MSEL_AN03 | ADC_MSEL_AN04;    /* グループ B に AN03 と AN04 を指定 */
    groupb_pin.sensor   = ADC_SENSOR_NOTUSE; /* 温度センサ未使用 */
    (void)adcDev->ScanSet(ADC_GROUP_B, groupb_pin, ADC_TRIGER_ELC);
                /* グループ B で使用する端子の指定および A/D 変換開始トリガを設定します */

    groupc_pin.an_chans = ADC_MSEL_AN22;   /* グループ C に AN22 を指定 */
    groupc_pin.sensor   = ADC_SENSOR_NOTUSE; /* 温度センサ未使用 */
    (void)adcDev->ScanSet(ADC_GROUP_C, groupc_pin, ADC_TRIGER_TMR);
                /* グループ C で使用する端子の指定および A/D 変換開始トリガを設定します */

    (void)adcDev->Control(AD_CMD_SET_ADI_INT, &adc_int);
                                            /* グループ A スキャン終了割り込み許可 */
    (void)adcDev->Control(AD_CMD_SET_GROUPB_INT, &adc_int);
                                            /* グループ B スキャン終了割り込み許可 */
    (void)adcDev->Control(AD_CMD_SET_GROUPC_INT, &adc_int);
                                            /* グループ C スキャン終了割り込み許可 */

    while(1)
    {
        /* A/D 変換開始トリガ入力待ち(注 1)
          [A/D 変換開始トリガ]
          グループ A：ADTRG0 端子入力
          グループ B：ELC_S14AD イベント
          グループ C：TMR_TCORA（TMR コンペアマッチ A） */
    }
}
/***********************************************************************************************
* callback function
***********************************************************************************************/
static void callback(uint32_t event)
{
    switch(event)
    {
        case ADC_EVT_SCAN_COMPLETE:
        {
            /* A/D 変換が終了
              （グループスキャンモードの場合、グループ A スキャン終了）した場合の処理を記述 */
        }
        break;
```

図 2-4 S14AD ドライバへのアクセス例（グループスキャン）(1/2)

```
        case ADC_EVT_SCAN_COMPLETE_GROUPB:
        {
            /* グループ B の A/D 変換が終了した場合の処理を記述 */
        }
        break;

        case ADC_EVT_SCAN_COMPLETE_GROUPC:
        {
            /* グループ C の A/D 変換が終了した場合の処理を記述 */
        }
        break;

        case ADC_EVT_CONDITION_MET:
        {
            /* コンペア A が一致した場合の処理(注 2) を記述 */
        }
        break;

        case ADC_EVT_CONDITION_METB
        {
            /* コンペア B が一致した場合の処理(注 2) を記述 */
        }
        break;

        case ADC_EVT_WINDOW_CMP_MATCH:
        {
            /* ウィンドウ A/B のコンペア条件に一致した場合の処理(注 3) を記述 */
        }
        break;

        case ADC_EVT_WINDOW_CMP_UNMATCH:
        {
            /* ウィンドウ A/B のコンペア条件に不一致した場合の処理(注 3) を記述 */
        }
        break;
        default:
        {
        }
        break;
    }
}
```

図 2-5 S14AD ドライバへのアクセス例（グループスキャン）(2/2)

注1. 同期トリガ（ELC_S14AD イベント、TMR_TCORA（TMR コンペアマッチ A））を使用する場合、各機能（ELC、TMR）設定を別途実施してください。

注2. 本例では、コンペアマッチ設定を行っていないためこのイベントは発生しません。

注3. 本例では、ウィンドウコンペアマッチ設定およびウィンドウ A/B コンペア一致/不一致ベントの割り込み許可設定を行っていないため、このイベントは発生しません。

```
#include "r_adc_api.h"
// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

static void callback(uint32_t event);

main()
{
    st_adc_pins_t groupa_pin;  /* グループ A の A/D 変換チャネル */
    st_adc_wina_t wina;    /* AD_CMD_SET_WINDOWA 第 2 引数 */
    st_adc_winb_t winb;    /* AD_CMD_SET_WINDOWB 第 2 引数 */
    st_adc_status_info_t result_status;    /* AD_CMD_GET_AD_STATE 第 2 引数 */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, callback);
                    /* S14AD ドライバ初期化
                    (RAM の初期化、端子設定、モジュールストップ状態の解除）を行います
                     また、引数で指定したモードの設定を行います */

    groupa_pin.an_chans = ADC_MSEL_AN02 | ADC_MSEL_AN06;    /* グループ A に AN02 と AN06 を指定 */
    groupa_pin.sensor   = ADC_SENSOR_NOTUSE; /* 温度センサ未使用 */
    (void)adcDev->ScanSet(ADC_GROUP_A, groupa_pin, ADC_TRIGER_ADTRG);
                  /* グループ A で使用する端子の指定および A/D 変換開始トリガを設定します */

    wina.inten             = ADC_ENABLE;      /* コンペア A 割り込み許可 */
    wina.cmp_mode       = ADC_CMP_WINDOW;    /* ウィンドウ比較 */
    wina.level1            = 0x02000;           /* コンペア比較基準値 1 */
    wina.level2            = 0x00100;           /* コンペア比較基準値 2 */
    wina.chans.an_chans    = ADC_MSEL_AN02;  /* 対象チャネルに AN02 を指定 */
    wina.chans.sensor       = ADC_SENSOR_NOTUSE;    /* 温度センサ未使用 */
    wina.chan_cond.an_chans = ADC_MSEL_AN02;
                          /* チャネル別比較条件   AN02 をレベル 1 とレベル 2 の範囲内に指定 */
    wina.chan_cond.sensor   = ADC_SENSOR_NOTUSE; /* 温度センサ未使用 */
    (void)adcDev->Control(AD_CMD_SET_WINDOWA, &wina);    /* ウィンドウコンペア A 設定 */

   winb.inten      = ADC_ENABLE;                  /* コンペア B 割り込み許可 */
   winb.winb_cond = ADC_WINB_WINDOW_BETWEEN;  /* ウィンドウ範囲内 */
   winb.comb      = ADC_COMB_OR;                /* 複合条件   OR 条件 */
   winb.level1      = 0x03000;                      /* コンペア比較基準値 1 */
   winb.level2      = 0x00010;                      /* コンペア比較基準値 2*/
   winb.channel    = ADC_SSEL_AN06;              /* 対象チャネルに AN06 を指定 */
   result = adcDev->Control(AD_CMD_SET_WINDOWB, &winb); /* ウィンドウコンペア B 設定 */

    while(1)
    {
        (void)adcDev->Start();  /* A/D 変換開始 */

        do
        {
            (void)adcDev->Control(AD_CMD_GET_AD_STATE, & result_status);    /* A/D の状態を取得 */
        }
        while (ADC_STATE_RUN == result_status.ad_info);    /* A/D 変換完了待ち */
    }
}
```

図 2-6 S14AD ドライバへのアクセス例（コンペアマッチ）(1/2)

```
/***********************************************************************************************************
* callback function
***********************************************************************************************************/
static void callback(uint32_t event)
{
    switch(event)
    {
        case ADC_EVT_CONDITION_MET:
        {
            /* ウィンドウ A 比較条件一致時の処理 */
        }
        break;
        case ADC_EVT_CONDITION_METB:
        {
            /* ウィンドウ B 比較条件一致時の処理 */
        }
        break;
        case ADC_EVT_SCAN_COMPLETE:
        case ADC_EVT_SCAN_COMPLETE_GROUPC:
        case ADC_EVT_SCAN_COMPLETE_GROUPB:
        case ADC_EVT_WINDOW_CMP_MATCH:
        case ADC_EVT_WINDOW_CMP_UNMATCH:
        default:
        {
            /* 許可されていない割り込みイベントのため発生しません */
        }
        break;
    }
}
```

図 2-7 S14AD ドライバへのアクセス例（コンペアマッチ）(2/2)

```
#include "r_adc_api.h"

static void callback(void);

// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

static uint16_t read_data[5]; /* DMA 転送先 RAM */

main()
{
    st_adc_pins_t scanset_pin;  /* ScanSet()  A/D 変換チャネル */
    st_adc_dma_read_info_t arg;    /* AD_CMD_AUTO_READ_NORMAL 第 2 引数 */
    st_adc_status_info_t result_status;    /* AD_CMD_GET_AD_STATE 第 2 引数 */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL);
                    /* S14AD ドライバ初期化
                    （RAM の初期化、端子設定、モジュールストップ状態の解除）を行います
                     また、引数で指定したモードの設定を行います */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                        /* A/D 変換に AN000、AN001、AN006、AN022 を使用 */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* A/D 変換に温度センサを使用しない */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin,  ADC_TRIGER_SOFT);
                  /* グループ A で使用する端子の指定および A/D 変換開始トリガを設定します */


    arg.cb_event     = callback;                 /* コールバック関数 */
    arg.dma_fact     = ADC_READ_ADI;         /* DMA 転送要因 */
    arg.src_addr     = (uint32_t)&S14AD->ADDR1; /* 転送元 A/D データレジスタ */
    arg.dest_addr    = (uint32_t)&read_data[0];    /* 転送先 RAM */
    arg.transfer_count = 5;                      /* 転送回数 5 回 */
    arg.block_size    = 0;                      /* ブロックサイズなし(0 固定) */
    arg.reg_size      = 0;                      /* 転送サイズ指定なし(16bit 固定) */

    (void)adcDev->Control(AD_CMD_AUTO_READ_NORMAL, &arg);
                                        /* DMA 転送による A/D 変換結果自動読み取り機能設定 */

    while (1)
    {
        (void)adcDev->Start();  /* A/D 変換開始 */

        do
        {
            (void)adcDev->Control(AD_CMD_GET_AD_STATE, & result_status);    /* A/D の状態を取得 */
        }
        while (ADC_STATE_RUN == result_status.ad_info);    /* A/D 変換完了待ち */
    }
}

/*****************************************************************************************************
* callback function
*****************************************************************************************************/
static void callback(void)
{
    /* DMA 転送が指定回数(5)分終了した場合の処理を記述 */
}
```

図 2-8 S14AD ドライバへのアクセス例（DMA による A/D データ自動読み取り）

```
#include "r_adc_api.h"

// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

main()
{
    st_adc_pins_t scanset_pin;  /* ScanSet()  A/D 変換チャネル */
    e_adc_int_method_t adc_int = ADC_INT_POLLING;    /* ポーリング設定 */
    st_adc_status_info_t result_status;    /* AD_CMD_GET_AD_STATE 第 2 引数 */


    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL);
                    /* S14AD ドライバ初期化
                    (RAM の初期化、端子設定、モジュールストップ状態の解除）を行います
                     また、引数で指定したモードの設定を行います */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                        /* A/D 変換に AN000、AN001、AN006、AN022 を使用 */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* A/D 変換に温度センサを使用しない */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_ADTRG);
                  /* グループ A で使用する端子の指定および A/D 変換開始トリガを設定します */

    (void)adcDev->Control(AD_CMD_SET_ADI_INT, &adc_int);
                                                /* スキャン終了割り込みポーリング使用 */

    while (1)
    {
        /* 外部トリガ(ADTRG)入力で A/D 変換を開始 */
        (void)adcDev->Control(AD_CMD_GET_AD_STATE, & result_status);    /* A/D の状態を取得 */
        if (ADC_CONV_COMPLETE == result_status.groupa_info)
        {
            /* グループ A の A/D 変換が完了した場合の処理を記述 */
        }
    }
    while(1);
}
```

図 2-9 S14AD ドライバへのアクセス例（ポーリング）

```
#include "r_adc_api.h"

// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;


static void callback(uint32_t event);

main()
{
    st_adc_pins_t scanset_pin;  /* ScanSet()  A/D 変換チャネル */
    st_adc_status_info_t result_status;    /* AD_CMD_GET_AD_STATE 第 2 引数 */
    e_adc_int_method_t adc_int = ADC_INT_ENABLE;    /* 割り込み許可用変数 */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, callback);
                    /* S14AD ドライバ初期化
                    (RAM の初期化、端子設定、モジュールストップ状態の解除) を行います
                     また、引数で指定したモードの設定を行います */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                        /* A/D 変換に AN000、AN001、AN006、AN022 を使用 */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* A/D 変換に温度センサを使用しない */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_ADTRG);
                  /* グループ A で使用する端子の指定および A/D 変換開始トリガを設定します */

    (void)adcDev->Control(AD_CMD_SET_ADI_INT, &adc_int);
                                                /* スキャン終了割り込み許可 */

    while (1)
    {

        do
        {
            /* ADTRG 端子入力で A/D 変換開始 */
            (void)adcDev->Control(AD_CMD_GET_AD_STATE, & result_status);    /* A/D の状態を取得 */
        }
        while (ADC_STATE_RUN == result_status.ad_info);    /* A/D 変換完了待ち */

    }
}
/***********************************************************************************************
* callback function
***********************************************************************************************/
static void callback(uint32_t event)
{
    switch(event)
    {
        case ADC_EVT_SCAN_COMPLETE:
        {
            /* A/D 変換が終了した場合の処理を記述 */
        }
        break;
        case ADC_EVT_SCAN_COMPLETE_GROUPB:
        case ADC_EVT_SCAN_COMPLETE_GROUPC:
        case ADC_EVT_CONDITION_MET:
        case ADC_EVT_CONDITION_METB:
        case ADC_EVT_WINDOW_CMP_MATCH:
        case ADC_EVT_WINDOW_CMP_UNMATCH:
        default:
        {
            /* 許可されていない割り込みイベントのため発生しません */
        }
        break;
    }
}
```

図 2-10 S14AD ドライバへのアクセス例（外部トリガ）

2.3 端子設定

本ドライバで使用する端子は、pin.c の R_S14AD_Pinset 関数で設定、R_S14AD_Pinclr 関数で解放されます。R_S14AD_Pinset 関数は Open 関数から、R_S14AD_Pinclr 関数は Close 関数から呼び出されます。

使用する端子は、pin.c の R_S14AD_Pinset、R_S14AD_Pinclr 関数内を修正して選択してください。アナログ入力端子の設定例を図 2-11、図 2-12 に示します。

```
    /*****************************************************************************//**
    * @brief This function sets Pin of S14AD.
    ********************************************************************************/
    /* Function Name : R_S14AD_Pinset */
    void R_S14AD_Pinset(void)  // @suppress("Source file naming") @suppress("API function naming")
@suppress("Function length")
    {
        /* Disable protection for PFS function (Set to PWPR register) */
        R_SYS_RegisterProtectDisable(SYSTEM_REG_PROTECT_MPC);

        /* P000 を AN000 用端子に設定 */
        /* AN000 : P000 */
        PFS->P000PFS_b.ISEL = 0U;
        PFS->P000PFS_b.PSEL = 0U;
        PFS->P000PFS_b.PMR  = 0U;
        PFS->P000PFS_b.PDR  = 0U;
        PFS->P000PFS_b.ASEL = 1U;

        /* P001 を AN001 用端子に設定 */
        /* AN001 : P001 */
        PFS->P001PFS_b.ISEL = 0U;
        PFS->P001PFS_b.PSEL = 0U;
        PFS->P001PFS_b.PMR  = 0U;
        PFS->P001PFS_b.PDR  = 0U;
        PFS->P001PFS_b.ASEL = 1U;

        /* P002 を AN002 用端子に設定 */
        /* AN002 : P002 */
        PFS->P002PFS_b.ISEL = 0U;
        PFS->P002PFS_b.PSEL = 0U;
        PFS->P002PFS_b.PMR  = 0U;
        PFS->P002PFS_b.PDR  = 0U;
        PFS->P002PFS_b.ASEL = 1U;

        /* AN003 : P003 */
    //    PFS->P003PFS_b.ISEL = 0U;
    //    PFS->P003PFS_b.PSEL = 0U;
    //    PFS->P003PFS_b.PMR  = 0U;
    //    PFS->P003PFS_b.PDR  = 0U;
    //    PFS->P003PFS_b.ASEL = 1U;
         . . .
    //    /* ADTRG0 : P204 */
    //    PFS->P204PFS_b.ISEL = 0U;
    //    PFS->P204PFS_b.ASEL = 0U;
    //    PFS->P204PFS_b.PSEL = R_PIN_PRV_S14AD_PSEL;
    //    PFS->P204PFS_b.PMR  = 1U;

    /* P500 を ADTRG0 用端子に設定 */
        /* ADTRG0 : P500 */
        PFS->P500PFS_b.ISEL = 0U;
        PFS->P500PFS_b.ASEL = 0U;
        PFS->P500PFS_b.PSEL = R_PIN_PRV_S14AD_PSEL;
        PFS->P500PFS_b.PMR  = 0U;

        /* Enable protection for PFS function (Set to PWPR register) */
        R_SYS_RegisterProtectEnable(SYSTEM_REG_PROTECT_MPC);
    }/* End of function R_S14AD_Pinset() */
```

図 2-11 端子設定例(1/2)

```
/*****************************************************************************//**
* @brief This function clears the pin setting of S14AD.
*****************************************************************************/
/* Function Name : R_S14AD_Pinclr */
void R_S14AD_Pinclr(void)  // @suppress("Source file naming") @suppress("API function naming")
{
    /* Disable protection for PFS function (Set to PWPR register) */
    R_SYS_RegisterProtectDisable(SYSTEM_REG_PROTECT_MPC);

    /* AN000 端子を解放 */
    /* AN000 : P000 */
    PFS->P000PFS &= R_PIN_PRV_CLR_MASK;

    /* AN001 端子を解放 */
    /* AN001 : P001 */
    PFS->P001PFS &= R_PIN_PRV_CLR_MASK;

    /* AN002 端子を解放 */
    /* AN002 : P002 */
    PFS->P002PFS &= R_PIN_PRV_CLR_MASK;

    /* AN003 : P003 */
//    PFS->P003PFS &= R_PIN_PRV_CLR_MASK;
    ・・・
    /* ADTRG0 : P204 */
//    PFS->P204PFS &= R_PIN_PRV_CLR_MASK;

    /* ADTRG0 端子を解放 */
    /* ADTRG0 : P500 */
    PFS->P500PFS &= R_PIN_PRV_CLR_MASK;

    /* Enable protection for PFS function (Set to PWPR register) */
    R_SYS_RegisterProtectEnable(SYSTEM_REG_PROTECT_MPC);
}/* End of function R_S14AD_Pinclr() */
```

図 2-12 端子設定例(2/2)

2.4 S14AD の初期化

2.4.1 Open 関数による A/D スキャンモード設定

Open 関数では、S14AD ドライバの初期化（RAM の初期化、端子設定、モジュールストップ状態の解除）を行います。同時にスキャンモード、A/D 変換精度、A/D データレジスタフォーマット、デフォルトサンプリング時間(ADSSTRn(注 1)の初期値)の設定も行われます。引数には、使用するモードの定義を組み合わせて指定してください。

使用例）

シングルスキャンモード、12 ビット精度、A/D データレジスタフォーマット左詰、デフォルトサンプリング(0x10)、コールバックなしで初期化する場合

```
adcDev->Open(ADC_SINGLE_SCAN | ADC_12BIT | ADC_LEFT, 0x10, NULL);
```

S14AD ドライバの各設定に対応する定義を表 2-4～表 2-6 に示します。定義を指定しなかった場合は、(デフォルト)と記載されている機能が有効になります。第 2 引数のデフォルトサンプリング時間(注 2)には、2～255(注 3)の値を設定してください。

注1. 256KB グループ：n = 0～7、L、T　　1500KB グループ：n = 0～6、L、T

注2. ADSSTRn の初期値

注3. サンプリング時間設定可能範囲は、サブクロックモード設定によって異なります。サブクロックモードの設定方法については「2.5.20 サブクロックモード設定(AD_CMD_SCLK_ENABLE)」を参照してください。以下に、モードごとのサンプリング時間設定可能範囲を示します。

サブクロックモード無効：5～255

サブクロックモード有効：2～8

表 2-4 スキャンモード定義一覧

コマンド	内容
ADC_SINGLE_SCAN(デフォルト)	シングルスキャンモード
ADC_GROUP_SCAN	グループスキャンモード
ADC_REPEAT_SCAN	連続スキャンモード

表 2-5 A/D 変換精度定義一覧

コマンド	内容
ADC_14BIT (デフォルト)	14 ビット精度
ADC_12BIT	12 ビット精度

表 2-6 A/D データレジスタフォーマット定義一覧

コマンド	内容
ADC_RIGHT (デフォルト)	右詰め
ADC_LEFT	左詰め

2.4.2 ScanSet 関数による A/D 入力端子および A/D 変換開始トリガ設定

ScanSet 関数では、A/D 変換グループ（グループ A、グループ B、グループ C）ごとの A/D 入力端子設定（A/D 変換チャネル選択、温度センサ出力 A/D 変換選択）および A/D 変換開始トリガ設定を行います。第 1 引数に指定するグループ定義を表 2-7 に、第 2 引数に指定する A/D 変換チャネル指定定義を表 2-8 に、第 3 引数に指定する A/D 変換開始条件設定定義を表 2-9 に示します。

使用例）

```
st_adc_pins_t scanset_pin;   /* A/D 変換チャネル設定用変数 */
scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN03 | ADC_MSEL_AN22;
                                    /* A/D 変換に AN000、AN001、AN003、AN022 を使用 */
scanset_pin.sensor     = ADC_SENSOR_NOTUSE;     /* A/D 変換に温度センサを使用しない */
adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);
```

表 2-7 A/D グループ定義一覧

コマンド	内容
ADC_GROUP_A	グループ A
ADC_GROUP_B(注)	グループ B
ADC_GROUP_C(注)	グループ C

注　グループスキャンモード時にのみ指定可能です。

表 2-8 A/D 変換チャネル指定定義一覧

要素	設定値	内容
an_chans	ADC_MSEL_ANn の組み合わせ(注) (1500KB グループ： n = 00～06、16、17、20～28 256KB グループ： n = 00～07、16、17、20～21)	対象となる A/D 変換チャネル [設定例] /* AN000、AN016 を対象 */ arg.chans.anchans = ADC_MSEL_AN00 | ADC_MSEL_AN16;
sensor	ADC_SENSOR_NOTUSE	温度センサ出力を使用しない
	ADC_MSEL_TEMP	温度センサ出力を使用する

注　256KB グループのみ、ADC_MSEL_VSC_VCC も組み合わせて指定可能

表 2-9 A/D 変換開始条件定義一覧

コマンド	内容
ADC_TRIGER_SOFT	ソフトウェアトリガ
ADC_TRIGER_TMR	TMR トリガ（TMR0_TCORA （TCORA レジスタと TCNT カウンタのコンペアマッチ））
ADC_TRIGER_ELC	ELC トリガ（ELC_S14AD）
ADC_TRIGER_ADTRG	外部トリガ入力（ADTRG0）
ADC_TRIGER_LOW_PRIORITY_CONT_SCAN	トリガ要因非選択(注)

注　A/D グループスキャン優先コントロール機能で低優先グループの連続動作を有効とする場合、最も優先度の低いグループのトリガ要因は非選択(ADC_TRIGER_LOW_PRIORITY_CONT_SCAN)としてください。

2.5 Control 関数による S14AD 機能設定

Control 関数で制御コマンドを実行することで、各機能を設定もしくは実行します。Control 関数は、Open 関数による初期化後に使用してください。その他の制限事項は、各制御コマンドの詳細でご確認ください。

Control 関数を実行する場合、第 1 引数に制御コマンドを指定します。第 2 引数には、制御コマンドに応じた型の変数を用意し、変数へのポインタを指定してください。S14AD ドライバの制御コマンド一覧を表 2-10、表 2-11 に示します。第 2 引数の型および制御コマンドの使用例は、各制御コマンドの詳細をご確認ください。

制御コマンドで設定する機能には、同時に使用できないものがあります。機能の組み合わせについては表 2-12 をご確認ください。

表 2-10 制御コマンド一覧(1/2)

コマンド	内容	詳細
AD_CMD_SET_ADD_MODE	A/D 変換値加算/平均モードの設定をします	0
AD_CMD_SET_DBLTRG	ダブルトリガモードの設定をします	2.5.2
AD_CMD_SET_DIAG	自己診断モードの設定をします	2.5.3
AD_CMD_SET_ADI_INT	グループ A の A/D スキャン終了割り込み(ADC140_ADI)設定を行います	2.5.4
AD_CMD_SET_GROUPB_INT	グループ B の A/D スキャン終了割り込み(ADC140_GBADI)設定を行います	2.5.5
AD_CMD_SET_GROUPC_INT	グループ C の A/D スキャン終了割り込み(ADC140_GCADI)設定を行います	2.5.6
AD_CMD_SET_WCMPM_INT	ウィンドウ A/B コンペア機能の条件一致割り込み(ADC140_WCMPM)設定を行います	2.5.7
AD_CMD_SET_WCMPUM_INT	ウィンドウ A/B コンペア機能の条件不一致割り込み(ADC140_WCMPUM)設定を行います	2.5.8
AD_CMD_SET_AUTO_CLEAR	A/D データレジスタ自動クリア設定を行います	2.5.9
AD_CMD_SET_SAMPLING_AN000	AN000、自己診断のサンプリング時間を設定します	2.5.10
AD_CMD_SET_SAMPLING_AN001	AN001 のサンプリング時間を設定します	
AD_CMD_SET_SAMPLING_AN002	AN002 のサンプリング時間を設定します	
AD_CMD_SET_SAMPLING_AN003	AN003 のサンプリング時間を設定します	
AD_CMD_SET_SAMPLING_AN004	AN004 のサンプリング時間を設定します	
AD_CMD_SET_SAMPLING_AN005	AN005 のサンプリング時間を設定します	
AD_CMD_SET_SAMPLING_AN006	AN006 のサンプリング時間を設定します	
AD_CMD_SET_SAMPLING_AN007(注 1)	AN007 のサンプリング時間を設定します	
AD_CMD_SET_SAMPLING_AN016_AN017_AN020_TO_AN028(注 2)	AN016～17、AN020～AN028 のサンプリング時間を設定します	
AD_CMD_SET_SAMPLING_AN016_AN017_AN020_AN021_VSC_VCC(注 1)	AN016～17、AN020～AN021、VSC_VCC 端子電圧出力のサンプリング時間を設定します	
AD_CMD_SET_SAMPLING_TEMP	温度センサ出力のサンプリング時間を設定します	
AD_CMD_SET_ADNDIS	断線検出アシストの設定を行います	2.5.11
AD_CMD_SET_GROUP_PRIORITY	グループ優先動作設定を行います	2.5.12
AD_CMD_SET_ADNDIS	断線検出アシストの設定を行います	2.5.11
AD_CMD_SET_GROUP_PRIORITY	グループ優先動作設定を行います	2.5.12
AD_CMD_SET_WINDOWA	コンペア機能ウィンドウ A の設定を行います	2.5.13
AD_CMD_SET_WINDOWB	コンペア機能ウィンドウ B の設定を行います	2.5.14

注1. 256KB グループのみ

注2. 1500KB グループのみ

表 2-11 制御コマンド一覧(2/2)

コマンド	内容	詳細
AD_CMD_SET_ELC	スキャン終了イベント(ADC140_ELC)のイベント発生条件を設定します	2.5.15
AD_CMD_GET_CMP_RESULT	A/D コンペア機能比較結果を取得します	2.5.16
AD_CMD_GET_AD_STATE	A/D コンバータの状態を取得します	2.5.17
AD_CMD_USE_VREFL0	低電位基準電圧に VREFL0 を選択します	2.5.18
AD_CMD_USE_VREFH0	高電位基準電圧に VREFH0 を選択します	2.5.19
AD_CMD_SCLK_ENABLE	A/D 変換にサブクロックを使用するよう設定します	2.5.20
AD_CMD_CALIBRATION	オフセットキャリブレーションを実施します	2.5.21
AD_CMD_STOP_TRIG	トリガ要因を解除し、A/D 変換を停止させます	2.5.22
AD_CMD_AUTO_READ_NORMAL	A/D 変換スキャン終了割り込み要求を DMA 起動要因とし、対象チャネルの A/D 変換結果を DMA 転送で読み取ります	2.5.23
AD_CMD_AUTO_READ_BLOCK	A/D 変換スキャン終了割り込み要求を DMA 起動要因とし、複数チャネルの A/D 変換結果を DMA 転送で読み取ります	2.5.24
AD_CMD_AUTO_READ_COMPARE	ウィンドウ A/B コンペア機能の条件一致/不一致イベントを DMA 起動要因とし、任意のレジスタの値を DMA 転送で取り込みます	2.5.25
AD_CMD_AUTO_READ_STOP	任意の DMA 起動要因による DMA 転送自動読み取り動作を停止させます	2.5.26
AD_CMD_AUTO_READ_RESTART	任意の DMA 起動要因による DMA 転送自動読み取り動作を再開します	2.5.27

表 2-12 使用機能組み合わせ一覧

	ダブルトリガ	グループスキャン	自己診断	加算/平均	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	オートクリア
ダブルトリガ			X			X	X		
グループスキャン									
自己診断	X			X			X	X	
加算/平均			X						
グループ A 優先制御									
センサ	X							X	
コンペアマッチ	X		X						
断線検出アシスト			X			X			
オートクリア									

X：同時に使用できません。Control 関数（該当機能の制御コマンド）実行時にエラーが返ります。

2.5.1 加算/平均モード設定(AD_CMD_SET_ADD_MODE)

AD_CMD_SET_ADD_MODE コマンドでは、A/D 変換値加算/平均モードを設定します。第 2 引数には st_adc_add_mode_t 型変数のポインタを指定してください。st_adc_add_mode_t 型引数の設定値を表 2-13 に、AD_CMD_SET_ADD_MODE コマンド使用例を図 2-13 に示します。

本コマンドには以下の制限があります。

- A/D 変換停止中(ADCSR.ADST = 0)にのみ実行できます。A/D 変換動作の状態遷移については、「2.9 状態遷移」を参照してください。

表 2-13 st_adc_add_mode_t 型引数の設定値

要素	設定値	内容
add_mode	ADC_ADD_OFF	A/D 変換加算/平均モードオフ
	ADC_ADD_2_SAMPLES	加算モード/2 回変換(1 回加算)
	ADC_ADD_4_SAMPLES	加算モード/4 回変換(3 回加算)
	ADC_ADD_16_SAMPLES(注)	加算モード/16 回変換(15 回加算)
	ADC_ADD_AVG_2_SAMPLES	平均モード/2 回変換(1 回加算)
	ADC_ADD_AVG_4_SAMPLES	平均モード/4 回変換(3 回加算)
	ADC_ADD_AVG_16_SAMPLES(注)	平均モード/16 回変換(15 回加算)
chans. an_chans	ADC_MSEL_ANn の組み合わせ (1500KB グループ： n = 00～06、16、17、20～28 256KB グループ： n = 00～07、16、17、20～21)	A/D 変換加算/平均の対象となる A/D 変換チャネル [設定例] /* AN000、AN016 を対象 */ arg.chans.anchans = ADC_MSEL_AN00 | ADC_MSEL_AN16;
chans. sensor	ADC_SENSOR_NOTUSE	温度センサ出力を使用しない
	ADC_MSEL_TEMP	温度センサ出力を使用する

注　A/D 変換精度が 12 ビットの場合のみ使用できます。

```
#include "r_adc_api.h"

// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

main()
{
    st_adc_pins_t scanset_pin;  /* ScanSet() A/D 変換チャネル */
    st_adc_add_mode_t arg;    /* AD_CMD_SET_ADD_MODE 第 2 引数 */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* S14AD ドライバ初期化 */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                        /* A/D 変換に AN000、AN001、AN006、AN022 を使用 */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* A/D 変換に温度センサを使用しない */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);

    arg.add_mode      = ADC_ADD_4_SAMPLES;    /* 加算モード/4 回変換(3 回加算) */
    arg.chans.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                            /* AN000、AN006、AN022 を加算対象 */
    arg.chans.sensor   = ADC_SENSOR_NOTUSE;   /* 温度センサを使用しない */
    (void)adcDev->Control(AD_CMD_SET_ADD_MODE, &arg);  /* A/D 加算/平均モード設定 */

    (void)adcDev->Start();  /* A/D 変換開始 */

    while(1);
}
```

図 2-13 AD_CMD_SET_ADD_MODE コマンド使用例

2.5.2 ダブルトリガモード設定(AD_CMD_SET_DBLTRG)

AD_CMD_SET_DBLTRG コマンドでは、ダブルトリガモードの設定をします。第 2 引数には int8_t 型変数のポインタを指定してください。第 2 引数の設定値を表 2-14 に、AD_CMD_SET_DBLTRG コマンド使用例を図 2-14 に示します。

本コマンドには以下の制限があります。

- シングルスキャンモードもしくはグループスキャンモードで使用できます。連続スキャンモードでは使用できません（エラーを返します）。
- A/D 変換開始条件には同期トリガ（TMR、ELC）を指定してください。ソフトウェアトリガの場合、使用できません（エラーを返します）。
- A/D 変換停止中(ADCSR.ADST = 0)にのみ実行できます。A/D 変換動作の状態遷移については、「2.9 状態遷移」を参照してください。

表 2-14 第 2 引数の設定値

設定値	内容
-1	ダブルトリガモード無効
1500KB グループ：0～06、16、17、20～28 256KB グループ：0～07、16、17、20～21	A/D 変換 2 重化チャネル

```
#include "r_adc_api.h"

// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

main()
{
    st_adc_pins_t scanset_pin;  /* ScanSet()  A/D 変換チャネル */
    int8_t arg;    /* AD_CMD_SET_DBLTRG 第 2 引数 */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* S14AD ドライバ初期化 */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                        /* A/D 変換に AN000、AN001、AN006、AN022 を使用 */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* A/D 変換に温度センサを使用しない */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);    /* ソフトウェアトリガ */

    arg.add_mode      = 6;    /* AN006 を A/D 変換 2 重化チャネルに指定 */
    (void)adcDev->Control(AD_CMD_SET_DBLTRG, &arg);  /* ダブルトリガモード設定 */

    (void)adcDev->Start();  /* A/D 変換開始 */

    while(1);
}
```

図 2-14 AD_CMD_SET_DBLTRG コマンド使用例

2.5.3 自己診断モード設定(AD_CMD_SET_DIAG)

AD_CMD_SET_DIAG コマンドでは、自己診断モードを設定します。第 2 引数には e_adc_diag_t 型変数のポインタを指定してください。e_adc_diag_t 型引数の設定値に表 2-15、AD_CMD_SET_DIAG コマンド使用例を図 2-15 に示します。

表 2-15 e_adc_diag_t 型引数の設定値

設定値	内容
AD_DIAG_DISABLE	自己診断無効
AD_DIAG_0V	0V 電圧
AD_DIAG_HARF	基準電圧電源(VREFH0)×1/2 電圧
AD_DIAG_BASE	基準電圧電源(VREFH0)電圧
AD_DIAG_ROTATE	自己診断電圧ローテーションモード

```
#include "r_adc_api.h"

// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

main()
{
    st_adc_pins_t scanset_pin;  /* ScanSet()  A/D 変換チャネル */
    e_adc_diag_t arg;    /* AD_CMD_SET_DIAG 第 2 引数 */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* S14AD ドライバ初期化 */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                        /* A/D 変換に AN000、AN001、AN006、AN022 を使用 */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* A/D 変換に温度センサを使用しない */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);

    arg                 = AD_DIAG_0V;                /* 0V 電圧を使って自己診断を行う */
    (void)adcDev->Control(AD_CMD_SET_DIAG, &arg);  /* 自己診断モード設定 */

    (void)adcDev->Start();  /* A/D 変換開始 */

    while(1);
}
```

図 2-15　AD_CMD_SET_DIAG コマンド使用例

2.5.4 ADI 割り込み機能設定(AD_CMD_SET_ADI_INT)

AD_CMD_SET_ADI_INT コマンドでは、シングルスキャンもしくはグループ A の A/D スキャン終了割り込み(ADC140_ADI)の設定をします。第 2 引数には、e_adc_int_method_t 型変数のポインタを指定してください。e_adc_int_method_t 型引数の設定値を表 2-16 に、AD_CMD_SET_ADI_INT コマンド使用例を図 2-17 に示します。

本コマンドには以下の制限があります。

- 割り込み機能とオートリード機能は排他的に制御されます。オートリード機能で A/D スキャン終了割り込み(ADC140_ADI)要求を使用している場合、本コマンドは使用できません（エラーが返ります）。
- 割り込み要因(ADC140_ADI)を r_system_cfg.h にて、ネスト型ベクタ割り込みコントローラ（以下、NVIC）に登録する必要があります。NVIC への割り込み登録例を図 2-16 に示します。各割り込み要因の NVIC 定義および割り込み登録例は「5.6 NVIC への割り込み登録について」を参照してください。

表 2-16 e_adc_int_method_t 型引数の設定値

コマンド	内容
ADC_INT_DISABLE	割り込み禁止
ADC_INT_POLLING	ポーリング
ADC_INT_ENABLE	割り込み許可

```
・・・
#define SYSTEM_CFG_EVENT_NUMBER_WDT_NMIUNDF
       (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)  /*!< Numbers 0/4/8/12/16/20/24/28 only */
#define SYSTEM_CFG_EVENT_NUMBER_ADC140_ADI
       (SYSTEM_IRQ_EVENT_NUMBER0)            /*!< Numbers 0/4/8/12/16/20/24/28 only */
#define SYSTEM_CFG_EVENT_NUMBER_ADC140_WCMPM
       (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)  /*!< Numbers 0/4/8/12/16/20/24/28 only */
・・・
```

図 2-16 NVIC への割り込み登録

```
#include "r_adc_api.h"

// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

static void callback(uint32_t event);

main()
{
    st_adc_pins_t scanset_pin;  /* ScanSet()  A/D 変換チャネル */
    e_adc_int_method_t adc_int = ADC_INT_ENABLE;    /* 割り込み許可用変数 */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* S14AD ドライバ初期化 */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                        /* A/D 変換に AN000、AN001、AN006、AN022 を使用 */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* A/D 変換に温度センサを使用しない */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);    /* ソフトウェアトリガ */

    (void)adcDev->Control(AD_CMD_SET_ADI_INT, &adc_int);
                                                /* スキャン終了割り込み許可 */


    (void)adcDev->Start();  /* A/D 変換開始 */

    while(1);
}

/***********************************************************************************************
* callback function
***********************************************************************************************/
static void callback(uint32_t event)
{
    switch(event)
    {
        case ADC_EVT_SCAN_COMPLETE:
        {
            /* A/D 変換が終了した場合の処理を記述 */
        }
        break;
        case ADC_EVT_SCAN_COMPLETE_GROUPB:
        case ADC_EVT_SCAN_COMPLETE_GROUPC:
        case ADC_EVT_CONDITION_MET:
        case ADC_EVT_CONDITION_METB:
        case ADC_EVT_WINDOW_CMP_MATCH:
        case ADC_EVT_WINDOW_CMP_UNMATCH:
        default:
        {
            /* 許可されていない割り込みイベントのため発生しません */
        }
        break;

    }
}
```

図 2-17 AD_CMD_SET_ADI_INT コマンド使用例

2.5.5 GBADI 割り込み機能設定(AD_CMD_SET_GROUPB_INT)

AD_CMD_SET_GROUPB_INT コマンドでは、グループ B の A/D スキャン終了割り込み(ADC140_GBADI)の設定をします。第 2 引数には、e_adc_int_method_t 型変数のポインタを指定してください。e_adc_int_method_t 型引数の設定値を表 2-17 に、AD_CMD_SET_GROUPB_INT コマンド使用例を図 2-19 に示します。

本コマンドには以下の制限があります。

- 割り込み機能とオートリード機能は排他的に制御されます。オートリード機能で A/D スキャン終了割り込み(ADC140_GBADI)要求を使用している場合、本コマンドは使用できません(エラーが返ります)。
- 割り込み要因(ADC140_GBADI)を r_system_cfg.h にて、ネスト型ベクタ割り込みコントローラ（以下、NVIC）に登録する必要があります。NVIC への割り込み登録例を図 2-18 に示します。各割り込み要因の NVIC 定義および割り込み登録例は「5.6 NVIC への割り込み登録について」を参照してください。
- グループ B の A/D スキャン終了割り込みはグループスキャンモードでのみ有効です。

表 2-17 e_adc_int_method_t 型引数の設定値

コマンド	内容
ADC_INT_DISABLE	割り込み禁止
ADC_INT_POLLING	ポーリング
ADC_INT_ENABLE	割り込み許可

```
・・・
#define SYSTEM_CFG_EVENT_NUMBER_RTC_ALM           (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)  /*!< Numbers
       1/5/9/13/17/21/25/29 only */
#define SYSTEM_CFG_EVENT_NUMBER_ADC140_GBADI
       (SYSTEM_IRQ_EVENT_NUMBER1)  /*!< Numbers 1/5/9/13/17/21/25/29 only */
#define SYSTEM_CFG_EVENT_NUMBER_ADC140_WCMPUM      (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)  /*!< Numbers
       1/5/9/13/17/21/25/29 only */
・・・
```

図 2-18 NVIC への割り込み登録

```
#include "r_adc_api.h"

// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

static void callback(uint32_t event);

main()
{
    st_adc_pins_t groupa_pin;  /* グループ A の A/D 変換チャネル */
    st_adc_pins_t groupb_pin;  /* グループ B の A/D 変換チャネル */
    e_adc_int_method_t adc_int = ADC_INT_ENABLE;    /* 割り込み許可用変数 */

    (void)adcDev->Open(ADC_GROUP_SCAN | ADC_14BIT, 0x10, callback); /* S14AD ドライバ初期化 */

    groupa_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01;    /* グループ A に AN00 と AN01 を指定 */
    groupa_pin.sensor   = ADC_SENSOR_NOTUSE; /* 温度センサ未使用 */
    (void)adcDev->ScanSet(ADC_GROUP_A, groupa_pin, ADC_TRIGER_ADTRG);

    groupb_pin.an_chans = ADC_MSEL_AN03 | ADC_MSEL_AN04;    /* グループ B に AN03 と AN04 を指定 */
    groupb_pin.sensor   = ADC_SENSOR_NOTUSE; /* 温度センサ未使用 */
    (void)adcDev->ScanSet(ADC_GROUP_B, groupb_pin, ADC_TRIGER_ELC);
    (void)adcDev->Control(AD_CMD_SET_GROUPB_INT, &adc_int);
                                            /* グループ B スキャン終了割り込み許可 */

    while(1)
    {
        /* A/D 変換開始トリガ入力待ち
          [A/D 変換開始トリガ]
          グループ A：ADTRG0 端子入力
          グループ B：ELC_S14AD イベント */
    }
}

/***********************************************************************************************
* callback function
***********************************************************************************************/
static void callback(uint32_t event)
{
    switch(event)
    {
        case ADC_EVT_SCAN_COMPLETE_GROUPB:
        {
            /* グループ B の A/D 変換が終了した場合の処理を記述 */
        }
        break;
        case ADC_EVT_SCAN_COMPLETE:
        case ADC_EVT_SCAN_COMPLETE_GROUPC:
        case ADC_EVT_CONDITION_MET:
        case ADC_EVT_CONDITION_METB:
        case ADC_EVT_WINDOW_CMP_MATCH:
        case ADC_EVT_WINDOW_CMP_UNMATCH:
        default:
        {
            /* 許可されていない割り込みイベントのため発生しません */
        }
        break;

    }
}
```

図 2-19 AD_CMD_SET_GROUPB_INT コマンド使用例

2.5.6 GCADI 割り込み機能設定(AD_CMD_SET_GROUPC_INT)

AD_CMD_SET_GROUPC_INT コマンドでは、グループ C の A/D スキャン終了割り込み(ADC140_GCADI)の設定をします。第 2 引数には、e_adc_int_method_t 型変数のポインタを指定してください。e_adc_int_method_t 型引数の設定値を表 2-18 に、AD_CMD_SET_GROUPC_INT コマンド使用例を図 2-21 に示します。

本コマンドには以下の制限があります。

- 割り込み機能とオートリード機能は排他的に制御されます。オートリード機能で A/D スキャン終了割り込み(ADC140_GCADI)要求を使用している場合、本コマンドは使用できません(エラーが返ります)。
- 割り込み要因(ADC140_GCADI)を r_system_cfg.h にて、ネスト型ベクタ割り込みコントローラ（以下、NVIC）に登録する必要があります。NVIC への割り込み登録例を図 2-20 に示します。各割り込み要因の NVIC 定義および割り込み登録例は「5.6 NVIC への割り込み登録について」を参照してください。
- グループ C の A/D スキャン終了割り込みはグループスキャンモードでのみ有効です。

表 2-18 e_adc_int_method_t 型引数の設定値

コマンド	内容
ADC_INT_DISABLE	割り込み禁止
ADC_INT_POLLING	ポーリング
ADC_INT_ENABLE	割り込み許可

```
・・・
#define SYSTEM_CFG_EVENT_NUMBER_ADC140_CMPAI       (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)  /*!< Numbers
       2/6/10/14/18/22/26/30 only */
#define SYSTEM_CFG_EVENT_NUMBER_ADC140_GCADI
       (SYSTEM_IRQ_EVENT_NUMBER2)  /*!< Numbers 2/6/10/14/18/22/26/30 only */
#define SYSTEM_CFG_EVENT_NUMBER_IIC0_TEI           (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)  /*!< Numbers
       2/6/10/14/18/22/26/30 only */
・・・
```

図 2-20 NVIC への割り込み登録

```
#include "r_adc_api.h"
// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

static void callback(uint32_t event);

main()
{
    st_adc_pins_t groupa_pin;  /* グループ A の A/D 変換チャネル */
    st_adc_pins_t groupb_pin;  /* グループ B の A/D 変換チャネル */
    st_adc_pins_t groupc_pin;  /* グループ C の A/D 変換チャネル */
    e_adc_int_method_t adc_int = ADC_INT_ENABLE;    /* 割り込み許可用変数 */
    (void)adcDev->Open(ADC_GROUP_SCAN | ADC_14BIT, 0x10, callback); /* S14AD ドライバ初期化 */

    groupa_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01;    /* グループ A に AN00 と AN01 を指定 */
    groupa_pin.sensor   = ADC_SENSOR_NOTUSE; /* 温度センサ未使用 */
    (void)adcDev->ScanSet(ADC_GROUP_A, groupa_pin, ADC_TRIGER_ADTRG);

    groupb_pin.an_chans = ADC_MSEL_AN03 | ADC_MSEL_AN04;    /* グループ B に AN03 と AN04 を指定 */
    groupb_pin.sensor   = ADC_SENSOR_NOTUSE; /* 温度センサ未使用 */
    (void)adcDev->ScanSet(ADC_GROUP_B, groupb_pin, ADC_TRIGER_ELC);

    groupc_pin.an_chans = ADC_MSEL_AN22;   /* グループ C に AN22 を指定 */
    groupc_pin.sensor   = ADC_SENSOR_NOTUSE; /* 温度センサ未使用 */
    (void)adcDev->ScanSet(ADC_GROUP_C, groupc_pin, ADC_TRIGER_TMR);

    (void)adcDev->Control(AD_CMD_SET_GROUPC_INT, &adc_int);
                                            /* グループ C スキャン終了割り込み許可 */
    while(1)
    {
        /* A/D 変換開始トリガ入力待ち
          [A/D 変換開始トリガ]
          グループ A：ADTRG0 端子入力
          グループ B：ELC_S14AD イベント
          グループ C：TMR_TCORA（TMR コンペアマッチ A） */
    }
}
/***********************************************************************************************
* callback function
***********************************************************************************************/
static void callback(uint32_t event)
{
    switch(event)
    {
        case ADC_EVT_SCAN_COMPLETE_GROUPC:
        {
            /* グループ C の A/D 変換が終了した場合の処理を記述 */
        }
        break;
        case ADC_EVT_SCAN_COMPLETE:
        case ADC_EVT_SCAN_COMPLETE_GROUPB:
        case ADC_EVT_CONDITION_MET:
        case ADC_EVT_CONDITION_METB:
        case ADC_EVT_WINDOW_CMP_MATCH:
        case ADC_EVT_WINDOW_CMP_UNMATCH:
        default:
        {
            /* 許可されていない割り込みイベントのため発生しません */
        }
        break;
    }
}
```

図 2-21 AD_CMD_SET_GROUPC_INT コマンド使用例

2.5.7 WCMPM 割り込み許可(AD_CMD_SET_WCMPM_INT)

AD_CMD_SET_WCMPM_INT コマンドでは、ウィンドウ A/B コンペア機能の条件一致割り込み(ADC140_WCMPM)の設定をします。第 2 引数には、e_adc_int_method_t 型変数のポインタを指定してください。e_adc_int_method_t 型引数の設定値を表 2-19 に、AD_CMD_SET_WCMPM_INT コマンド使用例を図 2-23、図 2-24 に示します。

本コマンドには以下の制限があります。

- 割り込み機能とオートリード機能は排他的に制御されます。オートリード機能で A/D スキャン終了割り込み(ADC140_WCMPM)要求を使用している場合、本コマンドは使用できません（エラーが返ります）。
- 割り込み要因(ADC140_WCMPM)を r_system_cfg.h にて、ネスト型ベクタ割り込みコントローラ（以下、NVIC）に登録する必要があります。NVIC への割り込み登録例を図 2-22 に示します。各割り込み要因の NVIC 定義および割り込み登録例は「5.6 NVIC への割り込み登録について」を参照してください。
- ウィンドウ A/B コンペア機能の条件一致割り込みを発生させるには、ウィンドウ A/B コンペア機能の設定（複合条件設定）を行う必要があります。コンペア機能の設定については「2.5.13 コンペア機能ウィンドウ A 設定(AD_CMD_SET_WINDOWA)」、「2.5.14 コンペア機能ウィンドウ B 設定(AD_CMD_SET_WINDOWB)」を参照してください。

表 2-19 e_adc_int_method_t 型引数の設定値

コマンド	内容
ADC_INT_DISABLE	割り込み禁止
ADC_INT_POLLING	ポーリング
ADC_INT_ENABLE	割り込み許可

```
. . .
#define SYSTEM_CFG_EVENT_NUMBER_ADC140_ADI         (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)  /*!< Numbers
       0/4/8/12/16/20/24/28 only */
#define SYSTEM_CFG_EVENT_NUMBER_ADC140_WCMPM
       (SYSTEM_IRQ_EVENT_NUMBER0)  /*!< Numbers 0/4/8/12/16/20/24/28 only */
#define SYSTEM_CFG_EVENT_NUMBER_IIC0_RXI           (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)  /*!< Numbers
       0/4/8/12/16/20/24/28 only */
. . .
```

図 2-22 NVIC への割り込み登録

```
#include "r_adc_api.h"
// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

static void callback(uint32_t event);

main()
{
    st_adc_pins_t groupa_pin;  /* グループ A の A/D 変換チャネル */
    st_adc_wina_t wina;    /* AD_CMD_SET_WINDOWA 第 2 引数 */
    st_adc_winb_t winb;    /* AD_CMD_SET_WINDOWB 第 2 引数 */
    st_adc_status_info_t result_status;    /* AD_CMD_GET_AD_STATE 第 2 引数 */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, callback); /* S14AD ドライバ初期化 */

    groupa_pin.an_chans = ADC_MSEL_AN02 | ADC_MSEL_AN06;    /* グループ A に AN02 と AN06 を指定 */
    groupa_pin.sensor   = ADC_SENSOR_NOTUSE; /* 温度センサ未使用 */
    (void)adcDev->ScanSet(ADC_GROUP_A, groupa_pin, ADC_TRIGER_ADTRG);

    wina.inten             = ADC_DISABLE;      /* コンペア A 割り込み禁止 */
    wina.cmp_mode        = ADC_CMP_WINDOW;    /* ウィンドウ比較 */
    wina.level1            = 0x02000;           /* コンペア比較基準値 1 */
    wina.level2            = 0x00100;           /* コンペア比較基準値 2 */
    wina.chans.an_chans    = ADC_MSEL_AN02;  /* 対象チャネルに AN02 を指定 */
    wina.chans.sensor       = ADC_SENSOR_NOTUSE;    /* 温度センサ未使用 */
    wina.chan_cond.an_chans = ADC_MSEL_AN02;
                         /* チャネル別比較条件  AN02 をレベル 1 とレベル 2 の範囲内に指定 */
    wina.chan_cond.sensor   = ADC_SENSOR_NOTUSE; /* 温度センサ未使用 */
    (void)adcDev->Control(AD_CMD_SET_WINDOWA, &wina);    /* ウィンドウコンペア A 設定 */

    winb.inten      = ADC_DISABLE;                  /* コンペア B 割り込み禁止 */
    winb.winb_cond = ADC_WINB_WINDOW_BETWEEN;  /* ウィンドウ範囲内 */
    winb.comb      = ADC_COMB_OR;                /* 複合条件  OR 条件 */
    winb.level1      = 0x03000;                     /* コンペア比較基準値 1 */
    winb.level2      = 0x00010;                     /* コンペア比較基準値 2*/
    winb.channel    = ADC_SSEL_AN06;
    (void)adcDev->Control(AD_CMD_SET_WINDOWB, &winb);

    (void)adcDev->Control(AD_CMD_SET_WCMPM_INT, &adc_int);
                                                /* ウィンドウ AB コンペア一致割り込み許可 */
    while(1)
    {
        (void)adcDev->Start();  /* A/D 変換開始 */

        do
        {
            (void)adcDev->Control(AD_CMD_GET_AD_STATE, & result_status);    /* A/D の状態を取得 */
        }
        while (ADC_STATE_RUN == result_status.ad_info);    /* A/D 変換完了待ち */
    }
}
```

図 2-23 AD_CMD_SET_WCMPM_INT コマンド使用例(1/2)

```
/***********************************************************************************************************
* callback function
***********************************************************************************************************/
static void callback(uint32_t event)
{
    switch(event)
    {
        case ADC_EVT_WINDOW_CMP_MATCH:
        {
            /* ウィンドウ AB コンペア条件一致時の処理を記述 */
        }
        break;
        case ADC_EVT_SCAN_COMPLETE:
        case ADC_EVT_SCAN_COMPLETE_GROUPC:
        case ADC_EVT_SCAN_COMPLETE_GROUPB:
        case ADC_EVT_CONDITION_MET:
        case ADC_EVT_CONDITION_METB:
        case ADC_EVT_WINDOW_CMP_UNMATCH:
        default:
        {
            /* 許可されていない割り込みイベントのため発生しません */
        }
        break;
    }
}
```

図 2-24 AD_CMD_SET_WCMPM_INT コマンド使用例(2/2)

2.5.8 WCMPUM 割り込み許可(AD_CMD_SET_WCMPUM_INT)

AD_CMD_SET_WCMPUM_INT コマンドでは、ウィンドウ A/B コンペア機能の条件不一致割り込み(ADC140_WCMPUM)の設定をします。第 2 引数には、e_adc_int_method_t 型変数のポインタを指定してください。e_adc_int_method_t 型引数の設定値を表 2-20 に、AD_CMD_SET_WCMPUM_INT コマンド使用例を図 2-26、図 2-27 に示します。

本コマンドには以下の制限があります。

- 割り込み機能とオートリード機能は排他的に制御されます。オートリード機能で A/D スキャン終了割り込み(ADC140_WCMPUM)要求を使用している場合、本コマンドは使用できません（エラーが返ります）。
- 割り込み要因(ADC140_WCMPUM)を r_system_cfg.h にて、ネスト型ベクタ割り込みコントローラ（以下、NVIC）に登録する必要があります。NVIC への割り込み登録例を図 2-25 に示します。各割り込み要因の NVIC 定義および割り込み登録例は「5.6 NVIC への割り込み登録について」を参照してください。
- ウィンドウ A/B コンペア機能の条件不一致割り込みを発生させるには、ウィンドウ A/B コンペア機能の設定（複合条件設定）を行う必要があります。コンペア機能の設定については「2.5.13 コンペア機能ウィンドウ A 設定(AD_CMD_SET_WINDOWA)」、「2.5.14 コンペア機能ウィンドウ B 設定(AD_CMD_SET_WINDOWB)」を参照してください。

表 2-20 e_adc_int_method_t 型引数の設定値

コマンド	内容
ADC_INT_DISABLE	割り込み禁止
ADC_INT_POLLING	ポーリング
ADC_INT_ENABLE	割り込み許可

```
. . .
#define SYSTEM_CFG_EVENT_NUMBER_ADC140_ADI         (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)  /*!< Numbers
       0/4/8/12/16/20/24/28 only */
#define SYSTEM_CFG_EVENT_NUMBER_ADC140_WCMPM
       (SYSTEM_IRQ_EVENT_NUMBER0)  /*!< Numbers 0/4/8/12/16/20/24/28 only */
#define SYSTEM_CFG_EVENT_NUMBER_IIC0_RXI           (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)  /*!< Numbers
       0/4/8/12/16/20/24/28 only */
. . .
```

図 2-25 NVIC への割り込み登録

```
#include "r_adc_api.h"
// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

static void callback(uint32_t event);

main()
{
    st_adc_pins_t groupa_pin;  /* グループ A の A/D 変換チャネル */
    e_adc_int_method_t adc_int = ADC_INT_ENABLE;    /* 割り込み許可用変数 */
    st_adc_wina_t wina;    /* AD_CMD_SET_WINDOWA 第 2 引数 */
    st_adc_winb_t winb;    /* AD_CMD_SET_WINDOWB 第 2 引数 */
    st_adc_status_info_t result_status;    /* AD_CMD_GET_AD_STATE 第 2 引数 */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, callback); /* S14AD ドライバ初期化 */

    groupa_pin.an_chans = ADC_MSEL_AN02 | ADC_MSEL_AN06;    /* グループ A に AN02 と AN06 を指定 */
    groupa_pin.sensor   = ADC_SENSOR_NOTUSE; /* 温度センサ未使用 */
    (void)adcDev->ScanSet(ADC_GROUP_A, groupa_pin, ADC_TRIGER_ADTRG);

    wina.inten             = ADC_DISABLE;      /* コンペア A 割り込み禁止 */
    wina.cmp_mode        = ADC_CMP_WINDOW;    /* ウィンドウ比較 */
    wina.level1            = 0x02000;          /* コンペア比較基準値 1 */
    wina.level2            = 0x00100;          /* コンペア比較基準値 2 */
    wina.chans.an_chans    = ADC_MSEL_AN02;  /* 対象チャネルに AN02 を指定 */
    wina.chans.sensor       = ADC_SENSOR_NOTUSE;    /* 温度センサ未使用 */
    wina.chan_cond.an_chans = ADC_MSEL_AN02;
                          /* チャネル別比較条件  AN02 をレベル 1 とレベル 2 の範囲内に指定 */
    wina.chan_cond.sensor   = ADC_SENSOR_NOTUSE; /* 温度センサ未使用 */
    (void)adcDev->Control(AD_CMD_SET_WINDOWA, &wina);    /* ウィンドウコンペア A 設定 */

    winb.inten      = ADC_DISABLE;                  /* コンペア B 割り込み禁止 */
    winb.winb_cond = ADC_WINB_WINDOW_BETWEEN;  /* ウィンドウ範囲内 */
    winb.comb      = ADC_COMB_OR;                 /* 複合条件  OR 条件 */
    winb.level1      = 0x03000;                      /* コンペア比較基準値 1 */
    winb.level2      = 0x00010;                      /* コンペア比較基準値 2*/
    winb.channel    = ADC_SSEL_AN06;
    (void)adcDev->Control(AD_CMD_SET_WINDOWB, &winb);

    (void)adcDev->Control(AD_CMD_SET_WCMPUM_INT, &adc_int);
                                                /* ウィンドウ AB コンペア不一致割り込み許可 */
    while(1)
    {
        (void)adcDev->Start();  /* A/D 変換開始 */

        do
        {
            (void)adcDev->Control(AD_CMD_GET_AD_STATE, & result_status);    /* A/D の状態を取得 */
        }
        while (ADC_STATE_RUN == result_status.ad_info);    /* A/D 変換完了待ち */
    }
}
```

図 2-26 AD_CMD_SET_WCMPUM_INT コマンド使用例(1/2)

```
/***********************************************************************************************************
* callback function
***********************************************************************************************************/
static void callback(uint32_t event)
{
    switch(event)
    {
        case ADC_EVT_WINDOW_CMP_UNMATCH:
        {
            /* ウィンドウ AB コンペア条件不一致時の処理を記述 */
        }
        break;
        case ADC_EVT_SCAN_COMPLETE:
        case ADC_EVT_SCAN_COMPLETE_GROUPC:
        case ADC_EVT_SCAN_COMPLETE_GROUPB:
        case ADC_EVT_CONDITION_MET:
        case ADC_EVT_CONDITION_METB:
        case ADC_EVT_WINDOW_CMP_MATCH:
        default:
        {
            /* 許可されていない割り込みイベントのため発生しません */
        }
        break;
    }
}
```

図 2-27 AD_CMD_SET_WCMPUM_INT コマンド使用例(2/2)

2.5.9 A/D データレジスタ自動クリア設定(AD_CMD_SET_AUTO_CLEAR)

AD_CMD_SET_AUTO_CLEAR コマンドでは、A/D データレジスタ自動クリアの設定をします。第 2 引数には uint8_t 型変数のポインタを指定してください。第 2 引数の設定値を表 2-21 に、AD_CMD_SET_AUTO_CLEAR 使用例を図 2-28 に示します。

表 2-21 第 2 引数の設定値

設定値	内容
ADC_ENABLE	A/D データレジスタ自動クリア有効
ADC_DISABLE	A/D データレジスタ自動クリア無効

```
#include "r_adc_api.h"

// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

main()
{
    st_adc_pins_t scanset_pin;  /* ScanSet()  A/D 変換チャネル */
    uint8_t arg;    /* AD_CMD_SET_AUTO_CLEAR 第 2 引数 */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* S14AD ドライバ初期化 */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                        /* A/D 変換に AN000、AN001、AN006、AN022 を使用 */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* A/D 変換に温度センサを使用しない */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);    /* ソフトウェアトリガ */

    arg      = ADC_ENABLE;    /* A/D データレジスタ自動クリア有効 */
    (void)adcDev->Control(AD_CMD_SET_AUTO_CLEAR, &arg);  /* A/D データレジスタ自動クリア設定 */

    (void)adcDev->Start();  /* A/D 変換開始 */

    while(1);
}
```

図 2-28 AD_CMD_SET_AUTO_CLEAR コマンド使用例

2.5.10 サンプリング時間設定(AD_CMD_SET_SAMPLING_XXX)

サンプリング時間設定コマンド(AD_CMD_SET_SAMPLING_XXX)には、AD_CMD_SET_SAMPLING_AN0yy(注 1)、AD_CMD_SET_SAMPLING_AN016_AN017_AN020_TO_AN028、AD_CMD_SET_SAMPLING_TEMP があります。

各チャネルのサンプリング時間には、Open 関数で指定したデフォルトサンプリング時間が設定されます。個別にサンプリング時間を変更する場合、本コマンドを使用してください。各コマンドに対応するサンプリング時間指定対象を表 2-22 に示します。

表 2-22 AD_CMD_SET_SAMPLING_XXX コマンド設定対象一覧

コマンド	内容
AD_CMD_SET_SAMPLING_AN0yy	AN0yy
AD_CMD_SET_SAMPLING_AN016_AN017_AN020_TO_AN028(注 2)	AN016、AN017、AN020～AN28
AD_CMD_SET_SAMPLING_AN016_AN017_AN020_AN021_VSC_VCC (注 3)	AN016、AN017、AN020、AN21、VSC_VCC 端子電圧出力
AD_CMD_SET_SAMPLING_TEMP	温度センサ出力

注1. 1500KB：yy = 0～6、16、17、20～28　256KB：yy = 0～7、16、17、20、21

注2. 1500KB のみ

注3. 256KB のみ

第 2 引数には uint8_t 型変数のポインタを指定してください。第 2 引数の設定値を表 2-23 に、サンプリング時間設定使用例を図 2-29 に示します。

表 2-23 第 2 引数の設定値

設定値	内容
2～255(注 1) (注 2)	サンプリング時間

注1. 範囲外の値を設定した場合、エラーが返ります。

注2. ADSCLKCR.SCLKEN ビットが 0（サブクロックモード無効）の場合、5～255 を設定してください。
ADSCLKCR.SCLKEN ビットが 1（サブクロックモード有効）の場合、2～8 を設定してください。

```
#include "r_adc_api.h"

// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

main()
{
    st_adc_pins_t scanset_pin;  /* ScanSet()  A/D 変換チャネル */
    uint8_t arg;    /* AD_CMD_SET_SAMPLING_001 第 2 引数 */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* S14AD ドライバ初期化 */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                        /* A/D 変換に AN000、AN001、AN006、AN022 を使用 */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* A/D 変換に温度センサを使用しない */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);    /* ソフトウェアトリガ */

    arg      = 5;    /* サンプリング時間(5) */
    (void)adcDev->Control(AD_CMD_SET_SAMPLING_AN001, &arg);  /* AN001 サンプリング時間設定 */

    (void)adcDev->Start();  /* A/D 変換開始 */

    while(1);
}
```

図 2-29 AD_CMD_SET_SAMPLING_AN001 コマンド使用例

2.5.11 断線検出アシスト設定(AD_CMD_SET_ADNDIS)

AD_CMD_SET_ADNDIS コマンドでは、断線検出アシストの設定をします。第 2 引数には uint8_t 型変数のポインタを指定してください。第 2 引数の設定値を表 2-24 に、AD_CMD_SET_ADNDIS 使用例を図 2-30 に示します。

本コマンドには以下の制限があります。

- A/D 変換停止中(ADCSR.ADST = 0)にのみ実行できます。A/D 変換動作の状態遷移については、「2.9 状態遷移」を参照してください。

表 2-24 第 2 引数の設定値

設定値	内容
ADC_DDA_OFF	断線検出アシスト機能無効
ADC_DDA_PRECHARGE(注)	プリチャージ
ADC_DDA_DISCHARGE(注)	ディスチャージ

注 プリチャージ/ディスチャージの設定を行う場合、プリチャージ/ディスチャージ期間と組み合わせて指定します。

使用例）プリチャージ期間 5 サイクル設定時

uint8_t arg = ADC_DDA_DISCHARGE | 5;

adcDrv -> Control(ADC_DDA_PRECHARGE, &arg);

```
#include "r_adc_api.h"

// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

main()
{
    st_adc_pins_t scanset_pin;  /* ScanSet()  A/D 変換チャネル */
    int8_t arg;    /* AD_CMD_SET_ADNDIS 第 2 引数 */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* S14AD ドライバ初期化 */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                        /* A/D 変換に AN000、AN001、AN006、AN022 を使用 */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* A/D 変換に温度センサを使用しない */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);    /* ソフトウェアトリガ */

    arg      = ADC_DDA_PRECHARGE | 5;    /* プリチャージ（チャージ期間(5)） */
    (void)adcDev->Control(AD_CMD_SET_ADNDIS, &arg);  /* 断線検出アシスト設定 */

    (void)adcDev->Start();  /* A/D 変換開始 */

    while(1);
}
```

図 2-30 AD_CMD_SET_ADNDIS コマンド使用例

2.5.12 グループ優先動作設定(AD_CMD_SET_GROUP_PRIORITY)

AD_CMD_SET_GROUP_PRIORITY コマンドでは、グループ優先動作を設定します。第 2 引数には e_adc_diag_t 型変数のポインタを指定してください。e_adc_gsp_t 型引数の設定値を表 2-25 に、AD_CMD_SET_GROUP_PRIORITY コマンド使用例を図 2-31 に示します。

本コマンドには以下の制限があります。

- A/D 変換停止中(ADCSR.ADST = 0)にのみ実行できます。A/D 変換動作の状態遷移については、「2.9 状態遷移」を参照してください。
- グループスキャンモード時にのみ実行できます。シングルスキャンモードおよび連続スキャンモードで実行した場合、エラーが返ります。

表 2-25 e_adc_gsp_t 型引数の設定値

設定値	内容
ADC_GSP_PRIORYTY_OFF	グループ優先動作無効
ADC_GSP_WAIT_TRG	低優先グループ再起動無効
ADC_GSP_SCAN_BIGIN	低優先グループ再起動有効
ADC_GSP_SCAN_RESTART	A/D 変換未終了チャネルから再スキャン
ADC_GSP_WAIT_TRG_CONT_SCAN	低優先グループのシングルスキャン連続動作有効かつ低優先グループ再起動無効(注)
ADC_GSP_SCAN_BIGIN_CONT_SCAN	低優先グループのシングルスキャン連続動作有効かつ低優先グループ再起動有効(注)
ADC_GSP_SCAN_RESTART_CONT_SCAN	低優先グループのシングルスキャン連続動作有効かつ A/D 変換未終了チャネルから再スキャン(注)

注　低優先グループのシングルスキャン連続動作を有効にする場合、最も優先度の低いグループの A/D 変換開始トリガを非選択にする必要があります。トリガ要因非選択の設定方法は「2.4.2 ScanSet 関数による A/D 入力端子および A/D 変換開始トリガ設定」を参照してください。

```
#include "r_adc_api.h"

// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

main()
{
    st_adc_pins_t groupa_pin;  /* グループ A の A/D 変換チャネル */
    st_adc_pins_t groupb_pin;  /* グループ B の A/D 変換チャネル */
    e_adc_gsp_t arg;    /* 割り込み許可用変数 */

    (void)adcDev->Open(ADC_GROUP_SCAN | ADC_14BIT, 0x10, callback); /* S14AD ドライバ初期化 */

    groupa_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01;    /* グループ A に AN00 と AN01 を指定 */
    groupa_pin.sensor   = ADC_SENSOR_NOTUSE; /* 温度センサ未使用 */
    (void)adcDev->ScanSet(ADC_GROUP_A, groupa_pin, ADC_TRIGER_ADTRG);

    groupb_pin.an_chans = ADC_MSEL_AN03 | ADC_MSEL_AN04;    /* グループ B に AN03 と AN04 を指定 */
    groupb_pin.sensor   = ADC_SENSOR_NOTUSE; /* 温度センサ未使用 */
    (void)adcDev->ScanSet(ADC_GROUP_B, groupb_pin, ADC_TRIGER_ELC);

    arg = ADC_GSP_WAIT_TRG;    /* 低優先グループの再起動無効 */
    (void)adcDev->Control(AD_CMD_SET_GROUP_PRIORITY, &arg);    /* グループ優先動作設定 */

    while(1)
    {
        /* A/D 変換開始トリガ入力待ち
        [A/D 変換開始トリガ]
        グループ A：ADTRG0 端子入力
        グループ B：ELC_S14AD イベント */
    }

}
```

図 2-31 AD_CMD_SET_GROUP_PRIORITY コマンド使用例

2.5.13 コンペア機能ウィンドウ A 設定(AD_CMD_SET_WINDOWA)

AD_CMD_SET_WINDOWA コマンドでは、デジタルコンペア機能（ウィンドウ A）を設定します。第 2 引数には st_adc_wina_t 型変数のポインタを指定してください。st_adc_wina_t 型引数の設定値を表 2-26 に、AD_CMD_SET_WINDOWA コマンド使用例を図 2-34 に示します。

本コマンドには以下の制限があります。

- A/D 変換停止中(ADCSR.ADST = 0)にのみ実行できます。A/D 変換動作の状態遷移については、「2.9 状態遷移」を参照してください。
- ウィンドウ B で温度センサを選択する場合は、ウィンドウ A 動作を使用することはできません（エラーが返ります）。
- ウィンドウ A およびウィンドウ B で同じチャネルを設定することはできません(エラーが返ります)。
- r_adc_cfg.h で ADC_CMPAI_SNOOZE_USE(注)を 0 に設定している場合、割り込みの許可/禁止にかかわらずウィンドウ A の比較条件一致割り込み(ADC140_CMPAI)の設定が必要です。割り込みの設定については図 2-33 を参照してください。各割り込み要因の NVIC 定義および割り込み登録例は「5.6 NVIC への割り込み登録について」を参照してください。
- r_adc_cfg.h で ADC_CMPAI_SNOOZE_USE(注)を 1 に設定している場合、ウィンドウ A の比較条件一致割り込み(ADC140_CMPAI)の設定をスキップすることができます。この場合、CMPAI 割り込みは無効になります。

注　ADC_CMPAI_SNOOZE_USE については「3.5.14 CMPAI スヌーズモード使用設定」参照

表 2-26 st_adc_wina_t 型引数の設定値

要素	設定値	内容
inten(注 1)	ADC_ENABLE	ウィンドウ A の比較条件一致割り込み(ADC140_CMPAI)許可
	ADC_DISABLE	ウィンドウ A の比較条件一致割り込み(ADC140_CMPAI)禁止
cmp_mode	ADC_CMP_OFF	コンペア比較しない
	ADC_CMP_LEVEL	レベル比較
	ADC_CMP_WINDOW	ウィンドウ比較
level1	uint16_t 型の値(注 2)	比較レベル(レベル 1)を指定(注 3)
level2	uint16_t 型の値(注 2)	比較レベル(レベル 2)を指定(注 3) (注 4)
chans. an_chans	ADC_MSEL_ANn の組み合わせ (1500KB グループ： n = 00～06、16、17、20～28 256KB グループ： n = 00～07、16、17、20～21(注 5))	A/D 変換加算/平均の対象となる A/D 変換チャネル [設定例] /* AN000、AN016 を対象 */ arg.chans.anchans = ADC_MSEL_AN00 | ADC_MSEL_AN16;
chans. sensor	ADC_SENSOR_NOTUSE	温度センサ出力を使用しない
	ADC_MSEL_TEMP	温度センサ出力を使用する
chan_cond. an_chans	ADC_MSEL_ANn の組み合わせ (1500KB グループ： n = 00～06、16、17、20～28 256KB グループ： n = 00～07、16、17、20～21(注 5))	チャネル別比較条件設定 [レベル比較] 指定したチャネル：レベル 1 以上 指定していないチャネル：レベル 1 以下 [ウィンドウ比較] 指定したチャネル：レベル 1 とレベル 2 の範囲内 指定していないチャネル：レベル 1 とレベル 2 の範囲外 [設定例] /* AN000、AN016 */ arg. chan_cond.anchans = ADC_MSEL_AN00 | ADC_MSEL_AN16;
chan_cond. sensor	ADC_SENSOR_NOTUSE	温度センサ出力の比較条件設定 [レベル比較] レベル 1 以下 [ウィンドウ比較] レベル 1 とレベル 2 の範囲外
	ADC_MSEL_TEMP	温度センサ出力の比較条件設定 [レベル比較] レベル 1 以上 [ウィンドウ比較] レベル 1 とレベル 2 の範囲内

注1. CMPAI スヌーズ使用時、割り込み設定は無効です。
注2. 比較レベル基準値の設定値および設定フォーマットは、A/D 変換精度、A/D データレジスタフォーマット、A/D 変換値加算モード設定によって異なります。
A/D 変換値加算モードを非選択とした場合
- 右詰めのフォーマット、14 ビット精度の場合：下位 14 ビット（b13 ～ b0）が有効
- 右詰めのフォーマット、12 ビット精度の場合：下位 12 ビット（b11 ～ b0）が有効
- 左詰めのフォーマット、14 ビット精度の場合：上位 14 ビット（b15 ～ b2）が有効
- 左詰めのフォーマット、12 ビット精度の場合：上位 12 ビット（b15 ～ b4）が有効

A/D 変換値加算モードを選択した場合
- 右詰めのフォーマット、14 ビット精度の場合：全ビット（b15 ～ b0）が有効
- 右詰めのフォーマット、12 ビット精度の場合：下位 14 ビット（b13 ～ b0）が有効
- 左詰めのフォーマット、14 ビット精度の場合：全ビット（b15 ～ b0）が有効
- 左詰めのフォーマット、12 ビット精度の場合：上位 14 ビット（b15 ～ b2）が有効

注3. ウィンドウ比較設定時、level1 と level2 の内値の大きい方をウィンドウ比較上位基準とします。
注4. レベル比較時は無効です。
注5. 256KB グループのみ、ADC_MSEL_VSC_VCC も組み合わせて指定可能

CMPAI スヌーズモード使用設定を行う場合は、r_adc_cfg.h で CMPAI スヌーズモードを有効にしてください。CMPAI スヌーズモード定義を表 2-27 に、スヌーズモード設定を図 2-32 に示します。

表 2-27 CMPAI スヌーズモード設定定義

定義	初期値	内容
ADC_CMPAI_SNOOZE_USE	0	0：コンペア機能ウィンドウ A を CMPAI 割り込みで使用する（CMPAI 割り込み設定を行う）(注) 1：コンペア機能ウィンドウ A を低消費電力モードのスヌーズモードでのみ使用する（CMPAI 割り込み設定をスキップする）

注　低消費電力モードのスヌーズモードでコンペア機能ウィンドウ A を使用し、かつ CMPAI 割り込み設定を行う場合、"0"を設定してください。"1"を設定した場合、CMPAI 割り込みは使用できません。

```
・・・
#define ADC_CMPAI_SNOOZE_USE  (1)
・・・
```

図 2-32 CMPAI スヌーズモード使用設定

ADC_CMPAI_SNOOZE_USE を 1 に設定すると、ウィンドウ A の比較条件一致割り込み(ADC140_CMPAI)の設定をスキップすることができます。この場合、CMPAI 割り込みは無効になります。

ADC_CMPAI_SNOOZE_USE を 0 に設定した場合は、必ず割り込み要因(ADC140_CMPAI)を r_system_cfg.h にて、ネスト型ベクタ割り込みコントローラ（以下、NVIC）に登録する必要があります。NVIC への割り込み登録例を図 2-33 に示します。

```
・・・
#define SYSTEM_CFG_EVENT_NUMBER_RTC_PRD            (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)  /*!< Numbers
       2/6/10/14/18/22/26/30 only */
#define SYSTEM_CFG_EVENT_NUMBER_ADC140_CMPAI
       (SYSTEM_IRQ_EVENT_NUMBER6)  /*!< Numbers 2/6/10/14/18/22/26/30 only */
#define SYSTEM_CFG_EVENT_NUMBER_ADC140_GCADI       (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)  /*!< Numbers
       2/6/10/14/18/22/26/30 only */
・・・
```

図 2-33 NVIC への割り込み登録

```
#include "r_adc_api.h"

// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

static void callback(uint32_t event);

main()
{
    st_adc_pins_t groupa_pin;  /* グループ A の A/D 変換チャネル */
    st_adc_wina_t wina;    /* AD_CMD_SET_WINDOWA 第 2 引数 */
    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, callback); /* S14AD ドライバ初期化 */

    groupa_pin.an_chans = ADC_MSEL_AN02 | ADC_MSEL_AN06;    /* グループ A に AN02 と AN06 を指定 */
    groupa_pin.sensor   = ADC_SENSOR_NOTUSE; /* 温度センサ未使用 */
    (void)adcDev->ScanSet(ADC_GROUP_A, groupa_pin, ADC_TRIGER_ADTRG);

    wina.inten              = ADC_ENABLE;      /* コンペア A 割り込み許可 */
    wina.cmp_mode        = ADC_CMP_WINDOW;    /* ウィンドウ比較 */
    wina.level1             = 0x02000;           /* コンペア比較基準値 1 */
    wina.level2             = 0x00100;           /* コンペア比較基準値 2 */
    wina.chans.an_chans     = ADC_MSEL_AN02;  /* 対象チャネルに AN02 を指定 */
    wina.chans.sensor       = ADC_SENSOR_NOTUSE;    /* 温度センサ未使用 */
    wina.chan_cond.an_chans = ADC_MSEL_AN02;
                          /* チャネル別比較条件  AN02 をレベル 1 とレベル 2 の範囲内に指定 */
    wina.chan_cond.sensor   = ADC_SENSOR_NOTUSE; /* 温度センサ未使用 */
    (void)adcDev->Control(AD_CMD_SET_WINDOWA, &wina);    /* ウィンドウコンペア A 設定 */

    (void)adcDev->Start();  /* A/D 変換開始 */

    while(1);
}

/***********************************************************************************************************
* callback function
***********************************************************************************************************/
static void callback(uint32_t event)
{
    switch(event)
    {
        case ADC_EVT_CONDITION_MET:
        {
            /* ウィンドウ A 比較条件一致時の処理を記述 */
        }
        break;
        case ADC_EVT_SCAN_COMPLETE:
        case ADC_EVT_SCAN_COMPLETE_GROUPB:
        case ADC_EVT_SCAN_COMPLETE_GROUPC:
        case ADC_EVT_CONDITION_METB:
        case ADC_EVT_WINDOW_CMP_MATCH:
        case ADC_EVT_WINDOW_CMP_UNMATCH:
        default:
        {
            /* 許可されていない割り込みイベントのため発生しません */
        }
        break;

    }
}
```

図 2-34　AD_CMD_SET_WINDOWA コマンド使用例

2.5.14 コンペア機能ウィンドウ B 設定(AD_CMD_SET_WINDOWB)

AD_CMD_SET_WINDOWB コマンドでは、デジタルコンペア機能（ウィンドウ B）を設定します。また、コンペア機能ウィンドウ A/B 複合条件設定も行います。第 2 引数には st_adc_winb_t 型変数のポインタを指定してください。st_adc_winb_t 型引数の設定値を表 2-28 に、AD_CMD_SET_WINDOWB コマンド使用例を図 2-37、図 2-38 に示します。

本コマンドには以下の制限があります。

- A/D 変換停止中(ADCSR.ADST = 0)にのみ実行できます。A/D 変換動作の状態遷移については、「2.9 状態遷移」を参照してください。
- コンペア機能イベント出力(ADC140_WCMPM/ADC140_WCMPUM)を使用する場合は、シングルスキャンモードに設定してください。
- ウィンドウ A およびウィンドウ B で同じチャネルを設定することはできません(エラーが返ります)。
- ウィンドウ A で温度センサを選択する場合は、ウィンドウ B 動作を使用することはできません（エラーが返ります）。
- r_adc_cfg.h で ADC_CMPBI_SNOOZE_USE(注)を 0 に設定している場合、割り込みの許可/禁止にかかわらずウィンドウ B の比較条件一致割り込み(ADC140_CMPBI)の設定が必要です。割り込みの設定については図 2-36 を参照してください。各割り込み要因の NVIC 定義および割り込み登録例は「5.6 NVIC への割り込み登録について」を参照してください。
- r_adc_cfg.h で ADC_CMPBI_SNOOZE_USE(注)を 1 に設定している場合、ウィンドウ B の比較条件一致割り込み(ADC140_CMPBI)の設定をスキップすることができます。この場合、CMPBI 割り込みは無効になります。

注　ADC_CMPBI_SNOOZE_USE については「3.5.15 CMPBI スヌーズモード使用設定」参照

表 2-28 st_adc_winb_t 型引数の設定値

要素	設定値	内容
inten(注 1)	ADC_ENABLE	ウィンドウ B の比較条件一致割り込み(ADC140_CMPBI)許可
	ADC_DISABLE	ウィンドウ B の比較条件一致割り込み(ADC140_CMPBI)禁止
winb_cond	ADC_WINB_LEVEL_BELOW	レベル 1 以下
	ADC_WINB_LEVEL_ABOVE	レベル 1 以上
	ADC_WINB_WINDOW_OUTSIDE	レベル 1 とレベル 2 の範囲外
	ADC_WINB_WINDOW_BETWEEN	レベル 1 とレベル 2 の範囲内
comb	ADC_COMB_OR	ウィンドウ A/B 複合条件設定 ウィンドウ A 比較条件に一致 OR ウィンドウ B 比較条件に一致で ADC140_WCMPM を出力、それ以外は ADC140_WCMPUM を出力
	ADC_COMB_EXOR	ウィンドウ A/B 複合条件設定 ウィンドウ A 比較条件に一致 EXOR ウィンドウ B 比較条件に一致で ADC140_WCMPM を出力､それ以外は ADC140_WCMPUM を出力
	ADC_COMB_AND	ウィンドウ A/B 複合条件設定 ウィンドウ A 比較条件に一致 AND ウィンドウ B 比較条件に一致で ADC140_WCMPM を出力、それ以外は ADC140_WCMPUM を出力
	ADC_COMB_NON_EVENT	ウィンドウ A/B 複合条件設定 ウィンドウ A のみでのイベント出力(注 2)
	ADC_COMB_OFF	ウィンドウ A/B 複合条件オフ
level1	uint16_t 型の値(注 3)	比較レベル(レベル 1)を指定(注 4)
level2	uint16_t 型の値(注 3)	比較レベル(レベル 2)を指定(注 4)
channel	ADC_SSEL_ANx (1500KB グループ： x = 00～06、16、17、20～28 256KB グループ： x = 00～07、16、17、20～21)	A/D 変換加算/平均の対象となる A/D 変換チャネルを 1 つ選択(注 5)
	ADC_SSEL_TEMP	温度センサ出力

注1. CMPBI スヌーズ使用時、割り込み設定は無効です。
コンペア機能のイベント出力(ADC140_WCMPM/ADC140_WCMPUM)をウィンドウ A でのみ使用する場合、ADC_COMB_NON_EVENT を指定してください。ADC_COMB_NON_EVENT を指定した場合、ウィンドウ B の設定は引数の設定値にかかわらず以下になります。
・ウィンドウ A/B 複合条件：OR 条件（ADCMPCR.CMPAB = 00b）
・ウィンドウ B 比較対象チャネル：非選択(ADCMPBNSR.CMPCHB = 0x3F)
・ウィンドウ B 比較条件：「常に不一致」となる「0 < 結果 < 0」
(ADCMPCR.WCMPE = 1、ADWINLLB[15:0] = ADWINULB[15:0] = 0000h、ADCMPBNSR.CMPLB = 1)

注2. 比較レベル基準値の設定値および設定フォーマットは、A/D 変換精度、A/D データレジスタフォーマット、A/D 変換値加算モード設定によって異なります。
A/D 変換値加算モードを非選択とした場合
・ 右詰めのフォーマット、14 ビット精度の場合：下位 14 ビット（b13 ～ b0）が有効
・ 右詰めのフォーマット、12 ビット精度の場合：下位 12 ビット（b11 ～ b0）が有効
・ 左詰めのフォーマット、14 ビット精度の場合：上位 14 ビット（b15 ～ b2）が有効
・ 左詰めのフォーマット、12 ビット精度の場合：上位 12 ビット（b15 ～ b4）が有効
A/D 変換値加算モードを選択した場合
・ 右詰めのフォーマット、14 ビット精度の場合：全ビット（b15 ～ b0）が有効
・ 右詰めのフォーマット、12 ビット精度の場合：下位 14 ビット（b13 ～ b0）が有効
・ 左詰めのフォーマット、14 ビット精度の場合：全ビット（b15 ～ b0）が有効
・ 左詰めのフォーマット、12 ビット精度の場合：上位 14 ビット（b15 ～ b2）が有効

注3. ウィンドウ比較設定時、level1 と level2 の内値の大きい方をウィンドウ比較上位基準とします。

注4. レベル比較時は無効です。

注5. 256KB グループのみ、ADC_SSEL_VSC_VCC も選択可能

CMPBI スヌーズモード使用設定を行う場合は、r_adc_cfg.h で CMPBI スヌーズモードを有効にしてください。CMPBI スヌーズモード定義を

表 2-29 に、スヌーズモード設定を図 2-35 に示します。

表 2-29 CMPBI スヌーズモード設定定義

定義	初期値	内容
ADC_CMPBI_SNOOZE_USE	0	0 : コンペア機能ウィンドウ B を CMPBI 割り込みで使用する（CMPBI 割り込み設定を行う）(注) 1 : コンペア機能ウィンドウ B を低消費電力モードのスヌーズモードでのみ使用する（CMPBI 割り込み設定をスキップする）

注　低消費電力モードのスヌーズモードでコンペア機能ウィンドウ B を使用し、かつ CMPBI 割り込み設定を行う場合、"0"を設定してください。"1"を設定した場合、CMPBI 割り込みは使用できません。

```
・・・
#define ADC_CMPBI_SNOOZE_USE  (1)
・・・
```

図 2-35 CMPBI スヌーズモード使用設定

ADC_CMPBI_SNOOZE_USE を 1 に設定すると、ウィンドウ B の比較条件一致割り込み(ADC140_CMPBI)の設定をスキップすることができます。この場合、CMPBI 割り込みは無効になります。

ADC_CMPBI_SNOOZE_USE を 0 に設定した場合は、必ず割り込み要因(ADC140_CMPBI)を r_system_cfg.h にて、ネスト型ベクタ割り込みコントローラ（以下、NVIC）に登録する必要があります。NVIC への割り込み登録例を図 2-36 に示します。

```
・・・
#define SYSTEM_CFG_EVENT_NUMBER_RTC_CUP            (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)  /*!< Numbers
       3/7/11/15/19/23/27/31 only */
#define SYSTEM_CFG_EVENT_NUMBER_ADC140_CMPBI
       (SYSTEM_IRQ_EVENT_NUMBER3)  /*!< Numbers 3/7/11/15/19/23/27/31 only */
#define SYSTEM_CFG_EVENT_NUMBER_ACMP_CMPI          (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)  /*!< Numbers
       3/7/11/15/19/23/27/31 only */
・・・
```

図 2-36 NVIC への割り込み登録

```
#include "r_adc_api.h"
// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

static void callback(uint32_t event);

main()
{
    st_adc_pins_t groupa_pin;  /* グループ A の A/D 変換チャネル */
    st_adc_wina_t wina;    /* AD_CMD_SET_WINDOWA 第 2 引数 */
    st_adc_winb_t winb;    /* AD_CMD_SET_WINDOWB 第 2 引数 */
    st_adc_status_info_t result_status;    /* AD_CMD_GET_AD_STATE 第 2 引数 */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, callback); /* S14AD ドライバ初期化 */

    groupa_pin.an_chans = ADC_MSEL_AN02 | ADC_MSEL_AN06;    /* グループ A に AN02 と AN06 を指定 */
    groupa_pin.sensor   = ADC_SENSOR_NOTUSE; /* 温度センサ未使用 */
    (void)adcDev->ScanSet(ADC_GROUP_A, groupa_pin, ADC_TRIGER_ADTRG);

    wina.inten             = ADC_DISABLE;      /* コンペア A 割り込み禁止 */
    wina.cmp_mode        = ADC_CMP_WINDOW;    /* ウィンドウ比較 */
    wina.level1            = 0x02000;           /* コンペア比較基準値 1 */
    wina.level2            = 0x00100;           /* コンペア比較基準値 2 */
    wina.chans.an_chans    = ADC_MSEL_AN02;  /* 対象チャネルに AN02 を指定 */
    wina.chans.sensor       = ADC_SENSOR_NOTUSE;    /* 温度センサ未使用 */
    wina.chan_cond.an_chans = ADC_MSEL_AN02;
                          /* チャネル別比較条件  AN02 をレベル 1 とレベル 2 の範囲内に指定 */
    wina.chan_cond.sensor   = ADC_SENSOR_NOTUSE; /* 温度センサ未使用 */
    (void)adcDev->Control(AD_CMD_SET_WINDOWA, &wina);    /* ウィンドウコンペア A 設定 */

    winb.inten      = ADC_ENABLE;                 /* コンペア B 割り込み許可 */
    winb.winb_cond = ADC_WINB_WINDOW_BETWEEN;  /* ウィンドウ範囲内 */
    winb.comb      = ADC_COMB_OR;                /* 複合条件  OR 条件 */
    winb.level1      = 0x03000;                    /* コンペア比較基準値 1 */
    winb.level2      = 0x00010;                    /* コンペア比較基準値 2*/
    winb.channel    = ADC_SSEL_AN06;
    (void)adcDev->Control(AD_CMD_SET_WINDOWB, &winb); /* ウィンドウコンペア B 設定 */

    while(1)
    {
        (void)adcDev->Start();  /* A/D 変換開始 */

        do
        {
            (void)adcDev->Control(AD_CMD_GET_AD_STATE, & result_status);    /* A/D の状態を取得 */
        }
        while (ADC_STATE_RUN == result_status.ad_info);    /* A/D 変換完了待ち */
    }
}
```

図 2-37 AD_CMD_SET_WINDOWB コマンド使用例(1/2)

```
/***********************************************************************************************************
* callback function
***********************************************************************************************************/
static void callback(uint32_t event)
{
    switch(event)
    {
        case ADC_EVT_CONDITION_METB:
        {
            /* ウィンドウ B 比較条件一致時の処理を記述 */
        }
        break;
        case ADC_EVT_SCAN_COMPLETE_GROUPC:
        case ADC_EVT_SCAN_COMPLETE:
        case ADC_EVT_SCAN_COMPLETE_GROUPB:
        case ADC_EVT_CONDITION_MET:
        case ADC_EVT_WINDOW_CMP_MATCH:
        case ADC_EVT_WINDOW_CMP_UNMATCH:
        default:
        {
            /* 許可されていない割り込みイベントのため発生しません */
        }
        break;
    }
}
```

図 2-38 AD_CMD_SET_WINDOWB コマンド使用例(2/2)

2.5.15 スキャン終了イベント(ADC140_ELC)発生条件設定(AD_CMD_SET_ELC)

AD_CMD_SET_ELC コマンドでは、スキャン終了イベント(ADC140_ELC)のイベント発生条件を設定します。第 2 引数には e_adc_elc_mode_t 型変数のポインタを指定してください。e_adc_elc_mode_t 型引数の設定を表 2-30 に、AD_CMD_SET_ELC コマンド使用例を図 2-39 に示します。

表 2-30 e_adc_elc_mode_t 型引数の設定値

設定値	内容
ADC_ELC_GROUPA	グループ B とグループ C のスキャン終了を除くスキャン終了時にイベント発生
ADC_ELC_GROUPB	グループ B スキャン終了時にイベント発生
ADC_ELC_GROUPC	グループ C スキャン終了時にイベント発生
ADC_ELC_ALL	すべてのスキャン終了時にイベント発生

```
#include "r_adc_api.h"

// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

main()
{
    st_adc_pins_t scanset_pin;  /* ScanSet()  A/D 変換チャネル */
    e_adc_elc_mode_t arg;    /* AD_CMD_SET_ELC 第 2 引数 */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* S14AD ドライバ初期化 */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                        /* A/D 変換に AN000、AN001、AN006、AN022 を使用 */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* A/D 変換に温度センサを使用しない */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_ADTRG);

    groupb_pin.an_chans = ADC_MSEL_AN03 | ADC_MSEL_AN04;    /* グループ B に AN03 と AN04 を指定 */
    groupb_pin.sensor   = ADC_SENSOR_NOTUSE; /* 温度センサ未使用 */
    (void)adcDev->ScanSet(ADC_GROUP_B, groupb_pin, ADC_TRIGER_ELC);


    arg                 = ADC_ELC_GROUPB;    /* グループ B スキャン終了時にイベント発生 */
    (void)adcDev->Control(AD_CMD_SET_ELC, &arg);
                                /* スキャン終了イベント(ADC140_ELC)のイベント発生条件設定 */

    (void)adcDev->Start();  /* A/D 変換開始 */

    while(1)
    {
        /* A/D 変換開始トリガ入力待ち
          [A/D 変換開始トリガ]
          グループ A：ADTRG0 端子入力
          グループ B：ELC_S14AD イベント */
    }
}
```

図 2-39　AD_CMD_SET_ELC コマンド使用例

2.5.16 A/D コンペア機能比較結果取得機能(AD_CMD_GET_CMP_RESULT)

AD_CMD_GET_CMP_RESULT コマンドでは、A/D コンペア機能比較結果を取得します。取得した A/D コンペア機能比較結果は第 2 引数に格納されます。第 2 引数には st_adc_cmp_result_t 型変数のポインタを指定してください。st_adc_cmp_result_t 型引数に格納される情報を表 2-31 に、AD_CMD_GET_CMP_RESULT コマンド使用例を図 2-40 に示します。

表 2-31 st_adc_cmp_result_t 型引数格納情報

要素	内容
wina_result. an_chans	A/D コンペア機能ウィンドウ A チャネルステータスレジスタ 0（ADCMPSR0）および A/D コンペア機能ウィンドウ A チャネルステータスレジスタ 1（ADCMPSR1）の値が格納されます 格納情報の各ビットは A/D チャネルに対応します（ビット 6 は AN006、ビット 0 は AN000 の比較結果）。 [端子ごとの比較結果] 0：比較条件不成立 1：比較条件成立
wina_result. sensor	A/D コンペア機能ウィンドウ A 拡張入力チャネルステータスレジスタ(ADCMPSER)の値が格納されます [温度センサ出力の比較結果] 0：比較条件不成立 1：比較条件成立
winb_result	A/D コンペア機能ウィンドウ B ステータスレジスタ（ADCMPBSR）の値が格納されます [ウィンドウ B 選択チャネル比較結果] 0：比較条件不成立 1：比較条件成立
comb_result	A/D コンペア機能ウィンドウ A/B ステータスモニタレジスタ（ADWINMON）の値が格納されます [組み合わせ結果(ビット 0)] 0：ウィンドウ A/ウィンドウ B の複合条件が不成立 1：ウィンドウ A/ウィンドウ B の複合条件が成立 [比較結果モニタ A(ビット 4)] 0：ウィンドウ A 比較条件が不成立 1：ウィンドウ A 比較条件が成立 [比較結果モニタ B(ビット 5)] 0：ウィンドウ B 比較条件が不成立 1：ウィンドウ B 比較条件が成立

```
#include "r_adc_api.h"
// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

main()
{
    st_adc_pins_t groupa_pin;  /* グループ A の A/D 変換チャネル */
    st_adc_wina_t wina;    /* AD_CMD_SET_WINDOWA 第 2 引数 */
    st_adc_winb_t winb;    /* AD_CMD_SET_WINDOWB 第 2 引数 */
    st_adc_cmp_result_t cmp_result; /* AD_CMD_GET_CMP_RESULT 第 2 引数 */
    st_adc_status_info_t result_status;    /* AD_CMD_GET_AD_STATE 第 2 引数 */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* S14AD ドライバ初期化 */

    groupa_pin.an_chans = ADC_MSEL_AN02 | ADC_MSEL_AN06;    /* グループ A に AN02 と AN06 を指定 */
    groupa_pin.sensor   = ADC_SENSOR_NOTUSE; /* 温度センサ未使用 */
    (void)adcDev->ScanSet(ADC_GROUP_A, groupa_pin, ADC_TRIGER_ADTRG);

    wina.inten             = ADC_DISABLE;      /* コンペア A 割り込み禁止 */
    wina.cmp_mode        = ADC_CMP_WINDOW;    /* ウィンドウ比較 */
    wina.level1            = 0x02000;           /* コンペア比較基準値 1 */
    wina.level2            = 0x00100;           /* コンペア比較基準値 2 */
    wina.chans.an_chans    = ADC_MSEL_AN02;  /* 対象チャネルに AN02 を指定 */
    wina.chans.sensor       = ADC_SENSOR_NOTUSE;    /* 温度センサ未使用 */
    wina.chan_cond.an_chans = ADC_MSEL_AN02;
                          /* チャネル別比較条件   AN02 をレベル 1 とレベル 2 の範囲内に指定 */
    wina.chan_cond.sensor   = ADC_SENSOR_NOTUSE; /* 温度センサ未使用 */
    (void)adcDev->Control(AD_CMD_SET_WINDOWA, &wina);    /* ウィンドウコンペア A 設定 */

    winb.inten      = ADC_DISABLE;                  /* コンペア B 割り込み禁止 */
    winb.winb_cond = ADC_WINB_WINDOW_BETWEEN;  /* ウィンドウ範囲内 */
    winb.comb      = ADC_COMB_OR;                /* 複合条件   OR 条件 */
    winb.level1      = 0x03000;                    /* コンペア比較基準値 1 */
    winb.level2      = 0x00010;                    /* コンペア比較基準値 2*/
    winb.channel    = ADC_SSEL_AN06;
    (void)adcDev->Control(AD_CMD_SET_WINDOWB, &winb); /* ウィンドウコンペア B 設定 */

    while(1)
    {
        (void)adcDev->Start();  /* A/D 変換開始 */

        do
        {
            (void)adcDev->Control(AD_CMD_GET_AD_STATE, & result_status);    /* A/D の状態を取得 */
        }
        while (ADC_STATE_RUN == result_status.ad_info);    /* A/D 変換完了待ち */
        (void)adcDev->Control(AD_CMD_GET_CMP_RESULT, & cmp_result);
        if (1 == cmp_result. winb_result)
        {
            /* コンペア機能ウィンドウ B の条件に一致した場合の処理を記載 */
        }
    }
}
```

図 2-40　AD_CMD_GET_CMP_RESULT コマンド使用例

2.5.17 A/D コンバータの状態取得機能(AD_CMD_GET_AD_STATE)

AD_CMD_GET_AD_STATE コマンドでは、A/D コンバータの状態を取得します。取得した A/D コンバータの状態情報は第 2 引数に格納されます。第 2 引数には st_adc_status_info_t 型変数のポインタを指定してください。st_adc_status_info_t 型引数に格納される情報を表 2-32 に、AD_CMD_GET_AD_STATE コマンド使用例を図 2-41 に示します。

表 2-32 st_adc_status_info_t 型引数格納情報

要素	内容
ad_info	A/D 変換の動作状態情報が格納されます ADC_STATE_STOP：A/D 変換停止中(ADCSR.ADST = 0) ADC_STATE_RUN：A/D 変換動作中(ADCSR.ADST = 1)
groupa_info	グループ A 動作状態情報が格納されます ADC_CONV_NOT_POLLING：ポーリング設定ではありません ADC_CONV_INCOMPLETE：グループ A の A/D スキャンが完了していません ADC_CONV_COMPLETE：グループ A の A/D スキャンが完了しています ADC_CONV_GET_FAILED：グループ A の状態取得に失敗しました
groupb_info	グループ B 動作状態情報が格納されます ADC_CONV_NOT_POLLING：ポーリング設定ではありません ADC_CONV_INCOMPLETE：グループ B の A/D スキャンが完了していません ADC_CONV_COMPLETE：グループ B の A/D スキャンが完了しています ADC_CONV_GET_FAILED：グループ B の状態取得に失敗しました
groupc_info	グループ C 動作状態情報が格納されます ADC_CONV_NOT_POLLING：ポーリング設定ではありません ADC_CONV_INCOMPLETE：グループ C の A/D スキャンが完了していません ADC_CONV_COMPLETE：グループ C の A/D スキャンが完了しています ADC_CONV_GET_FAILED：グループ C の状態取得に失敗しました
wcmpm_info	ウィンドウ A/B コンペアー致イベント情報が格納されます ADC_CONV_NOT_POLLING：ポーリング設定ではありません ADC_CONV_WCMP_DETECT：ウィンドウ A/B コンペアー致イベント検出 ADC_CONV_WCMP_NOT_DETECT：ウィンドウ A/B コンペアー致イベント未検出 ADC_CONV_GET_FAILED：ウィンドウ A/B コンペアー致イベントの状態取得に失敗しました
wcmpum_info	ウィンドウ A/B コンペア不一致イベント情報が格納されます ADC_CONV_NOT_POLLING：ポーリング設定ではありません ADC_CONV_WCMP_DETECT：ウィンドウ A/B コンペア不一致イベント検出 ADC_CONV_WCMP_NOT_DETECT：ウィンドウ A/B コンペア不一致イベント未検出 ADC_CONV_GET_FAILED：ウィンドウ A/B コンペア不一致イベントの状態取得に失敗しました

```
#include "r_adc_api.h"

// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

main()
{
    st_adc_pins_t scanset_pin;  /* ScanSet() A/D 変換チャネル */
    st_adc_status_info_t result_status;    /* AD_CMD_GET_AD_STATE 第 2 引数 */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* S14AD ドライバ初期化 */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                        /* A/D 変換に AN000、AN001、AN006、AN022 を使用 */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* A/D 変換に温度センサを使用しない */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);

    while(1)
    {
        (void)adcDev->Start();  /* A/D 変換開始 */
        do
        {
            (void)adcDev->Control(AD_CMD_GET_AD_STATE, & result_status);    /* A/D の状態を取得 */
        }
        while (ADC_STATE_RUN == result_status.ad_info);    /* A/D 変換完了待ち */
    }
}
```

図 2-41 AD_CMD_GET_AD_STATE コマンド使用例

2.5.18 低電位基準電圧設定(AD_CMD_USE_VREFL0)

AD_CMD_USE_VREFL0 コマンドでは、低電位基準電圧に VREFL0 を設定します。第 2 引数には NULL を指定してください。AD_CMD_USE_VREFL0 使用例を図 2-42 に示します。

本コマンドには以下の制限があります。

- A/D 変換停止中(ADCSR.ADST = 0)にのみ実行できます。A/D 変換動作の状態遷移については、「2.9 状態遷移」を参照してください。

```
#include "r_adc_api.h"

// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

main()
{
    st_adc_pins_t scanset_pin;  /* ScanSet()  A/D 変換チャネル */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* S14AD ドライバ初期化 */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                        /* A/D 変換に AN000、AN001、AN006、AN022 を使用 */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* A/D 変換に温度センサを使用しない */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);    /* ソフトウェアトリガ */

    (void)adcDev->Control(AD_CMD_USE_VREFL0, NULL);  /* 低電位基準電圧に VREFL0 を選択 */

    (void)adcDev->Start();  /* A/D 変換開始 */

    while(1);
}
```

図 2-42 AD_CMD_USE_VREFL0 コマンド使用例

2.5.19 高電位基準電圧設定(AD_CMD_USE_VREFH0)

AD_CMD_USE_VREFH0 コマンドでは、高電位基準電圧に VREFH0 を設定します。第 2 引数には NULL を指定してください。AD_CMD_USE_VREFH0 使用例を図 2-43 に示します。

本コマンドには以下の制限があります。

- A/D 変換停止中(ADCSR.ADST = 0)にのみ実行できます。A/D 変換動作の状態遷移については、「2.9 状態遷移」を参照してください。
- 以下の設定では使用できません（エラーが返ります）。
  - ・自己診断変換電圧に基準電圧×1/2 を選択
  - ・自己診断変換電圧に基準電圧を選択

```
#include "r_adc_api.h"

// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

main()
{
    st_adc_pins_t scanset_pin;  /* ScanSet()  A/D 変換チャネル */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* S14AD ドライバ初期化 */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                        /* A/D 変換に AN000、AN001、AN006、AN022 を使用 */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* A/D 変換に温度センサを使用しない */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);    /* ソフトウェアトリガ */

    (void)adcDev->Control(AD_CMD_USE_VREFH0, NULL);  /* 高電位基準電圧に VREFH0 を選択 */

    (void)adcDev->Start();  /* A/D 変換開始 */

    while(1);
}
```

図 2-43 AD_CMD_USE_VREFH0 コマンド使用例

2.5.20 サブクロックモード設定(AD_CMD_SCLK_ENABLE)

AD_CMD_SCLK_ENABLE コマンドでは、低周波クロック動作時（ADCLK = 32kHz)に内部生成クロック(サブクロック)を使用するよう設定します。第 2 引数には NULL を指定してください。AD_CMD_SCLK_ENABLE 使用例を図 2-44 に示します。

本コマンドには以下の制限があります。

- A/D 変換停止中(ADCSR.ADST = 0)にのみ実行できます。A/D 変換動作の状態遷移については、「2.9 状態遷移」を参照してください。
- 動作クロックが 32768Hz より小さい場合（低周波クロック動作時（ADCLK = 32kHz)）のみ実行できます。

```
#include "r_adc_api.h"

// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

main()
{
    st_adc_pins_t scanset_pin;  /* ScanSet()  A/D 変換チャネル */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* S14AD ドライバ初期化 */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                        /* A/D 変換に AN000、AN001、AN006、AN022 を使用 */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* A/D 変換に温度センサを使用しない */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);    /* ソフトウェアトリガ */

    (void)adcDev->Control(AD_CMD_SCLK_ENABLE, NULL);  /* A/D 変換時にサブクロック選択 */

    (void)adcDev->Start();  /* A/D 変換開始 */

    while(1);
}
```

図 2-44 AD_CMD_SCLK_ENABLE コマンド使用例

2.5.21 オフセットキャリブレーション設定(AD_CMD_CALIBRATION)

AD_CMD_CALIBRATION コマンドでは、オフセットキャリブレーションを実行します。第2引数にはNULLを指定してください。AD_CMD_CALIBRATION 使用例を図 2-45 に示します。

本コマンドには以下の制限があります。

- Open 関数での初期化後、ScanSet 関数実行前にのみ実行できます。

```
#include "r_adc_api.h"

// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

main()
{
    st_adc_pins_t scanset_pin;  /* ScanSet()  A/D 変換チャネル */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* S14AD ドライバ初期化 */

    (void)adcDev->Control(AD_CMD_CALIBRATION, NULL);  /* キャリブレーション実行 */
    /* 関数を抜けた時点でキャリブレーションは完了しています。ソフトによる完了待ち処理は不要です。 */

    while(1);
}
```

図 2-45 AD_CMD_CALIBRATION コマンド使用例

2.5.22 A/D 変換停止機能(AD_CMD_STOP_TRIG)

AD_CMD_STOP_TRIG コマンドでは、トリガ要因を解除し、A/D 変換を停止(ADCSR.ADST = 0)させます。オートリード機能を使用している場合は、DMA 転送要因を解放します(注 1)。同期トリガおよび非同期トリガによる A/D 変換を停止する場合は、本コマンドを使用してください(注 2)。第 2 引数には NULL を指定してください。AD_CMD_STOP_TRIG 使用例を図 2-46 に示します。

注1. オートリード機能で DMA 転送要因に指定した割り込み要求を割り込み機能で使用したい場合は、Close 関数を実行してから S14AD の初期化を行ってください。

注2. A/D 変換開始トリガ設定が初期化されます。再度 A/D 変換を実行する場合は、ScanSet 関数で A/D 変換開始トリガを再設定してください。オートリード機能を使用している場合は、オートリード機能の再設定も行ってください。

```
#include "r_adc_api.h"

// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

main()
{
    st_adc_pins_t scanset_pin;  /* ScanSet()  A/D 変換チャネル */
    e_adc_int_method_t adc_int = ADC_INT_ENABLE;    /* 割り込み許可用変数 */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* S14AD ドライバ初期化 */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                        /* A/D 変換に AN000、AN001、AN006、AN022 を使用 */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* A/D 変換に温度センサを使用しない */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_ELC);    /* ELC トリガ */

    /* ELC イベントで A/D 変換開始 */

    (void)adcDev->Control(AD_CMD_STOP_TRIG, NULL);  /* ELC トリガ要因を解除し A/D 変換を停止 */

    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_ELC);
                                    /* A/D 変換を再開する場合、もう一度 ScanSet 関数を実行する */


    while(1);
}
```

図 2-46 AD_CMD_STOP_TRIG コマンド使用例

2.5.23 オートリード機能(ノーマル)設定(AD_CMD_AUTO_READ_NORMAL)

AD_CMD_AUTO_READ_NORMAL コマンドでは、DMA 転送（DMAC もしくは DTC）による A/D 変換結果自動読み取り機能（オートリード機能(注)）の設定をします。第 2 引数には、st_adc_dma_read_info_t 型変数のポインタを指定してください。st_adc_dma_read_info_t 型引数の設定値を表 2-33 に、AD_CMD_AUTO_READ_NORMAL コマンド使用例を図 2-47 に示します。

本コマンドには以下の制限があります。

- オートリード機能と割り込み機能は、排他制御されます。割り込みで使用している A/D 割り込み要求を DMA 転送で使用することはできません（エラーが返ります）。また、DMA 起動要因に指定した割り込み要求を割り込みとして使用することはできません。
- r_adc_cfg.h の A/D コントロール定義の設定値を DMAC または DTC に変更する必要があります。DMA 転送による A/D 変換結果自動読み取り機能の動作および設定手順については「2.6.2 DMA 転送で A/D 変換結果を自動的に取り込む（オートリード機能）」を参照してください。
- DTC を使用する場合、DMA 起動要因に指定した割り込み要因を r_system_cfg.h にて、ネスト型ベクタ割り込みコントローラ（以下、NVIC）に登録する必要があります。DMA 起動要因ごとの NVIC 登録については「2.6.2 DMA 転送で A/D 変換結果を自動的に取り込む（オートリード機能）」を参照してください。

注　オートリード機能に関する詳細は「2.6.2 DMA 転送で A/D 変換結果を自動的に取り込む（オートリード機能）」をご確認ください。

表 2-33 st_adc_dma_read_info_t 型引数の設定値

要素	設定値	内容
cb_event	オートリード機能用コールバック関数	DMA 転送完了タイミングで呼び出されるコールバック関数 コールバック関数の引数には void を指定してください
dma_fact	ADC_READ_ADI	A/D スキャン終了割り込み要求(ADC140_ADI)を DMA 起動要因に設定
	ADC_READ_GBADI	グループ B の A/D スキャン終了割り込み要求(ADC140_GBADI)を DMA 起動要因に設定
	ADC_READ_GCADI	グループ C の A/D スキャン終了割り込み要求(ADC140_GCADI)を DMA 起動要因に設定
	ADC_READ_WCMPM(注 1)	ウィンドウ A/B コンペア機能の条件一致割り込み要求(ADC140_WCMPM)を DMA 起動要因に設定
	ADC_READ_WCMPUM(注 1)	ウィンドウ A/B コンペア機能の条件不一致割り込み要求(ADC140_WCMPUM)を DMA 起動要因に設定
src_addr	A/D データレジスタ y(ADDRy) (1500KB グループ： y = 00～06、16、17、20～28 256KB グループ： y = 00～07、16、17、20～21)	転送元アドレス DMA 転送で自動的に読み取る A/D データレジスタを 1 つ指定
	A/D データ 2 重化レジスタ(ADDBLDR)	
	A/D 温度センサデータレジスタ（ADTSDR）	
	A/D VSC_VCC 端子電圧データレジスタ（ADVSCDR）(注 2)	
dest_addr	任意の RAM	転送先アドレス
transfer_count	[DMA 転送に DMAC を使用した場合] 0～65535 [DMA 転送に DTC を使用した場合] 1～65536	転送回数を指定 DMAC 使用時に転送回数を 0 に指定すると、フリーランニングモードで動作します
block_size	- (注 3)	”0”(ブロックなし)固定
reg_size	- (注 3)	16 ビットサイズ固定

注1. 本コマンドでは使用できません。指定した場合、エラーが返ります。

注2. 256KB グループのみ

注3. 本コマンドでは、引数の値に関係なく設定は固定となります。

```
#include "r_adc_api.h"

static void callback(void);

// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

static uint16_t read_data[5]; /* DMA 転送先 RAM */

main()
{
    st_adc_pins_t scanset_pin;  /* ScanSet()  A/D 変換チャネル */
    st_adc_dma_read_info_t arg;    /* AD_CMD_AUTO_READ_NORMAL 第 2 引数 */

    (void)adcDev->Open(ADC_REPEAT_SCAN | ADC_14BIT, 0x10, NULL); /* S14AD ドライバ初期化 */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                        /* A/D 変換に AN000、AN001、AN006、AN022 を使用 */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* A/D 変換に温度センサを使用しない */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);

    arg.cb_event     = callback;                /* コールバック関数 */
    arg.dma_fact     = ADC_READ_ADI;         /* DMA 転送要因 */
    arg.src_addr     = (uint32_t)&S14AD->ADDR1; /* 転送元 A/D データレジスタ */
    arg.dest_addr    = (uint32_t)&read_data[0];    /* 転送先 RAM */
    arg.transfer_count = 5;                     /* 転送回数 5 回 */
    arg.block_size    = 0;                     /* ブロックサイズなし(0 固定) */
    arg.reg_size      = 0;                     /* 転送サイズ指定なし(16bit 固定) */

    (void)adcDev->Control(AD_CMD_AUTO_READ_NORMAL, &arg);
                                        /* DMA 転送による A/D 変換結果自動読み取り機能設定 */

    (void)adcDev->Start();  /* A/D 変換開始 */

    while(1);
}

/***********************************************************************************************
* callback function
***********************************************************************************************/
static void callback(void)
{
    /* DMA 転送が指定回数分終了した場合の処理を記述 */
}
```

図 2-47　AD_CMD_AUTO_READ_NORMAL コマンド使用例

2.5.24 オートリード機能(ブロック)設定(AD_CMD_AUTO_READ_BLOCK)

AD_CMD_AUTO_READ_BLOCK コマンドでは、DMA 転送（DMAC もしくは DTC）による A/D 変換結果自動読み取り機能（オートリード機能(注)）の設定をします。第 2 引数には、st_adc_dma_read_info_t 型変数のポインタを指定してください。st_adc_dma_read_info_t 型引数の設定値を表 2-34 に、AD_CMD_AUTO_READ_BLOCK コマンド使用例を図 2-48 に示します。

本コマンドには以下の制限があります。

- オートリード機能と割り込み機能は、排他制御されます。割り込みで使用している A/D 割り込み要求を DMA 転送で使用することはできません（エラーが返ります）。また、DMA 起動要因に指定した割り込み要求を割り込みとして使用することはできません。
- r_adc_cfg.h の A/D コントロール定義の設定値を DMAC または DTC に変更する必要があります。DMA 転送による A/D 変換結果自動読み取り機能の動作および設定手順については「2.6.2 DMA 転送で A/D 変換結果を自動的に取り込む（オートリード機能）」を参照してください。
- DTC を使用する場合、DMA 起動要因に指定した割り込み要因を r_system_cfg.h にて、ネスト型ベクタ割り込みコントローラ（以下、NVIC）に登録する必要があります。DMA 起動要因ごとの NVIC 登録については「2.6.2 DMA 転送で A/D 変換結果を自動的に取り込む（オートリード機能）」を参照してください。

注　オートリード機能に関する詳細は「2.6.2 DMA 転送で A/D 変換結果を自動的に取り込む（オートリード機能）」をご確認ください。

表 2-34 st_adc_dma_read_info_t 型引数の設定値

要素	設定値	内容
cb_event	オートリード機能用コールバック関数	DMA 転送完了タイミングで呼び出されるコールバック関数 コールバック関数の引数には void を指定してください
dma_fact	ADC_READ_ADI	A/D スキャン終了割り込み要求(ADC140_ADI)を DMA 起動要因に設定
	ADC_READ_GBADI	グループ B の A/D スキャン終了割り込み要求(ADC140_GBADI)を DMA 起動要因に設定
	ADC_READ_GCADI	グループ C の A/D スキャン終了割り込み要求(ADC140_GCADI)を DMA 起動要因に設定
	ADC_READ_WCMPM(注 1)	ウィンドウ A/B コンペア機能の条件一致割り込み要求(ADC140_WCMPM)を DMA 起動要因に設定
	ADC_READ_WCMPUM(注 1)	ウィンドウ A/B コンペア機能の条件不一致割り込み要求(ADC140_WCMPUM)を DMA 起動要因に設定
src_addr(注 3)	A/D データレジスタ y(ADDRy) (1500KB グループ： y = 00～06、16、17、20～28 256KB グループ： y = 00～07、16、17、20～21)	DMA 転送元レジスタ先頭アドレス DMA 転送で自動的に読み取る A/D データレジスタを 1 つ指定 ”block_size”と合わせて、複数の A/D データレジスタをブロックとして取得
	A/D データ 2 重化レジスタ(ADDBLDR)	
	A/D 温度センサデータレジスタ（ADTSDR）	
	A/D VSC_VCC 端子電圧データレジスタ（ADVSCDR）(注 2)	
dest_addr	任意の RAM	転送先アドレス
transfer_count	[DMA 転送に DMAC を使用した場合] 1～65536 [DMA 転送に DTC を使用した場合] 1～65536	転送回数を指定
block_size (注 3)(注 4)	[DMA 転送に DMAC を使用した場合] 1～256 [DMA 転送に DTC を使用した場合] 1～1024	転送ブロックサイズ ” src_addr”と合わせて、複数の A/D データレジスタをブロックとして取得
reg_size	- (注 5)	16 ビットサイズ固定

注1. 本コマンドでは使用できません。指定した場合、エラーが返ります。

注2. 256KB グループのみ

注3. 読み込み対象の A/D データレジスタは、レジスタの先頭アドレスおよびブロックサイズで指定します。
指定例）AN000～AN006 を 1 ブロックとして読み出す
arg. src_addr = (uint32_t)&S14AD->ADDR0; /* ブロックの先頭アドレスを指定 */
arg. block_size = 7; /* 先頭アドレスからのブロックサイズを指定 */

注4. 読み込み対象の A/D データブロックに存在しないチャネルの A/D データレジスタが指定されている場合、該当する RAM 領域には不正な値が転送されます。

例）AN017～AN020 をブロックとして読み込む場合、存在しないチャネルに対応する「AN018、AN019」に該当する RAM 領域には不正値が転送されます。

注5. 本コマンドでは、引数の値に関係なく設定は固定となります。

```
#include "r_adc_api.h"

static void callback(void);

// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

static uint16_t read_data[20]; /* DMA 転送先 RAM */

main()
{
    st_adc_pins_t scanset_pin;  /* ScanSet()  A/D 変換チャネル */
    st_adc_dma_read_info_t dma_read;    /* AD_CMD_AUTO_READ_BLOCK 第 2 引数 */

    (void)adcDev->Open(ADC_REPEAT_SCAN | ADC_14BIT, 0x10, NULL); /* S14AD ドライバ初期化 */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN02| ADC_MSEL_AN03;
                                    /* A/D 変換に AN000、AN001、AN002、AN003 を使用 */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* A/D 変換に温度センサを使用しない */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);

    dma_read.cb_event     = callback;                /* コールバック関数 */
    dma_read.dma_fact     = ADC_READ_ADI;        /* DMA 転送要因 */
    dma_read.src_addr     = (uint32_t)&S14AD->ADDR0; /* 転送元 A/D データレジスタ */
    dma_read.dest_addr    = (uint32_t)&read_data[0];    /* 転送先 RAM */
    dma_read.transfer_count = 5;                     /* 転送回数 5 回 */
    dma_read.block_size    = 4;                     /* ADDR0～ADDR3 を 1 ブロックとする */
    dma_read.reg_size      = 0;                     /* 転送サイズ指定なし(16bit 固定) */

    (void)adcDev->Control(AD_CMD_AUTO_READ_BLOCK, & dma_read);
                                        /* DMA 転送による A/D 変換結果自動読み取り機能設定 */

    (void)adcDev->Start();  /* A/D 変換開始 */

    while(1);
}

/***********************************************************************************************
* callback function
***********************************************************************************************/
static void callback(void)
{
    /* DMA 転送が指定回数分終了した場合の処理を記述 */
}
```

図 2-48　AD_CMD_AUTO_READ_BLOCK コマンド使用例

2.5.25 オートリード機能(コンペア)設定(AD_CMD_AUTO_READ_COMPARE)

AD_CMD_AUTO_READ_COMPARE コマンドでは、DMA 転送（DMAC もしくは DTC）による A/D 変換結果自動読み取り機能（オートリード機能(注)）の設定をします。第 2 引数には、st_adc_dma_read_info_t 型変数のポインタを指定してください。st_adc_dma_read_info_t 型引数の設定値を表 2-35 に、AD_CMD_AUTO_READ_COMPARE コマンド使用例を図 2-49 に示します。

本コマンドには以下の制限があります。

- オートリード機能と割り込み機能は、排他制御されます。割り込みで使用している A/D 割り込み要求を DMA 転送で使用することはできません（エラーが返ります）。また、DMA 起動要因に指定した割り込み要求を割り込みとして使用することはできません。
- r_adc_cfg.h の A/D コントロール定義の設定値を DMAC または DTC に変更する必要があります。DMA 転送による A/D 変換結果自動読み取り機能の動作および設定手順については「2.6.2 DMA 転送で A/D 変換結果を自動的に取り込む（オートリード機能）」を参照してください。
- DTC を使用する場合、DMA 起動要因に指定した割り込み要因を r_system_cfg.h にて、ネスト型ベクタ割り込みコントローラ（以下、NVIC）に登録する必要があります。DMA 起動要因ごとの NVIC 登録については「2.6.2 DMA 転送で A/D 変換結果を自動的に取り込む（オートリード機能）」を参照してください。
- コンペア機能（ウィンドウ A/B 複合条件）設定が必要です。コンペア機能の設定方法は「2.5.13 コンペア機能ウィンドウ A 設定(AD_CMD_SET_WINDOWA)」「2.5.14 コンペア機能ウィンドウ B 設定(AD_CMD_SET_WINDOWB)」を参照してください。

- オートリード機能に関する詳細は「2.6.2 DMA 転送で A/D 変換結果を自動的に取り込む（オートリード機能）」をご確認ください。

表 2-35 st_adc_dma_read_info_t 型引数の設定値

要素	設定値	内容
cb_event	オートリード機能用コールバック関数	DMA 転送完了タイミングで呼び出されるコールバック関数 コールバック関数の引数には void を指定してください
dma_fact	ADC_READ_ADI(注 1)	A/D スキャン終了割り込み要求(ADC140_ADI)を DMA 起動要因に設定
	ADC_READ_GBADI(注 1)	グループ B の A/D スキャン終了割り込み要求(ADC140_GBADI)を DMA 起動要因に設定
	ADC_READ_GCADI(注 1)	グループ C の A/D スキャン終了割り込み要求(ADC140_GCADI)を DMA 起動要因に設定
	ADC_READ_WCMPM	ウィンドウ A/B コンペア機能の条件一致割り込み要求(ADC140_WCMPM)を DMA 起動要因に設定
	ADC_READ_WCMPUM	ウィンドウ A/B コンペア機能の条件不一致割り込み要求(ADC140_WCMPUM)を DMA 起動要因に設定
src_addr	任意のレジスタ	転送元アドレス
dest_addr	任意の RAM	転送先アドレス
transfer_count	[DMA 転送に DMAC を使用した場合] 0～65535 [DMA 転送に DTC を使用した場合] 1～65536	転送回数を指定 DMAC 使用時に転送回数を 0 に指定すると、フリーランニングモードで動作します
block_size	- (注 2)	”0”(ブロックなし)固定
reg_size	ADC_READ_BYTE	8 ビット転送
	ADC_READ_WORD	16 ビット転送
	ADC_READ_LONG	32 ビット転送

注1. 本コマンドでは使用できません。指定した場合、エラーが返ります。

注2. 本コマンドでは、引数の値に関係なく設定は固定となります。

```
#include "r_adc_api.h"

static void callback(void);

// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

static uint8_t read_data[5]; /* DMA 転送先 RAM */

main()
{
    st_adc_pins_t groupa_pin;  /* グループ A の A/D 変換チャネル */
    st_adc_dma_read_info_t dma_read;    /* AD_CMD_AUTO_READ_COMPARE 第 2 引数 */
    st_adc_wina_t wina;    /* AD_CMD_SET_WINDOWA 第 2 引数 */
    st_adc_winb_t winb;    /* AD_CMD_SET_WINDOWB 第 2 引数 */
    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* S14AD ドライバ初期化 */

    groupa_pin.an_chans = ADC_MSEL_AN02 | ADC_MSEL_AN06;    /* グループ A に AN02 と AN06 を指定 */
    groupa_pin.sensor   = ADC_SENSOR_NOTUSE; /* 温度センサ未使用 */
    (void)adcDev->ScanSet(ADC_GROUP_A, groupa_pin, ADC_TRIGER_ADTRG);

    wina.inten             = ADC_DISABLE;      /* コンペア A 割り込み禁止 */
    wina.cmp_mode        = ADC_CMP_WINDOW;    /* ウィンドウ比較 */
    wina.level1            = 0x02000;           /* コンペア比較基準値 1 */
    wina.level2            = 0x00100;           /* コンペア比較基準値 2 */
    wina.chans.an_chans    = ADC_MSEL_AN02;  /* 対象チャネルに AN02 を指定 */
    wina.chans.sensor       = ADC_SENSOR_NOTUSE;    /* 温度センサ未使用 */
    wina.chan_cond.an_chans = ADC_MSEL_AN02;
                          /* チャネル別比較条件   AN02 をレベル 1 とレベル 2 の範囲内に指定 */
    wina.chan_cond.sensor   = ADC_SENSOR_NOTUSE; /* 温度センサ未使用 */
    (void)adcDev->Control(AD_CMD_SET_WINDOWA, &wina);    /* ウィンドウコンペア A 設定 */

    winb.inten      = ADC_DISABLE;                  /* コンペア B 割り込み禁止 */
    winb.winb_cond = ADC_WINB_WINDOW_BETWEEN;  /* ウィンドウ範囲内 */
    winb.comb      = ADC_COMB_OR;                /* 複合条件   OR 条件 */
    winb.level1      = 0x03000;                     /* コンペア比較基準値 1 */
    winb.level2      = 0x00010;                     /* コンペア比較基準値 2*/
    winb.channel    = ADC_SSEL_AN06;
    (void)adcDev->Control(AD_CMD_SET_WINDOWB, &winb); /* ウィンドウコンペア B 設定 */

    dma_read.cb_event     = callback;                       /* コールバック関数 */
    dma_read.dma_fact     = ADC_READ_WCMPUM;        /* DMA 転送要因 */
    dma_read.src_addr     = (uint32_t)&S14AD-> ADWINMON; /* 転送元 A/D データレジスタ */
    dma_read.dest_addr    = (uint32_t)&read_data[0];         /* 転送先 RAM */
    dma_read.transfer_count = 5;                           /* 転送回数 5 回 */
    dma_read.block_size    = 0;                           /* ブロックサイズなし(0 固定) */
    dma_read.reg_size      = ADC_READ_BYTE;            /* 8bit サイズ指定 */

    (void)adcDev->Control(AD_CMD_AUTO_READ_COMPARE & dma_read);
                                      /* DMA 転送による A/D 変換結果自動読み取り機能設定 */

    (void)adcDev->Start();  /* A/D 変換開始 */

    while(1);
}
/***********************************************************************************************
* callback function
***********************************************************************************************/
static void callback(void)
{
    /* DMA 転送が指定回数分終了した場合の処理を記述 */
}
```

図 2-49　AD_CMD_AUTO_READ_COMPARE コマンド使用例

2.5.26 オートリードストップ機能(AD_CMD_AUTO_READ_STOP)

AD_CMD_AUTO_READ_STOP コマンドでは、DMA 転送（DMAC もしくは DTC）による A/D 変換結果自動読み取り動作を停止させます。第 2 引数には、e_adc_dma_event_t 型変数のポインタを指定してください。e_adc_dma_event_t 型引数の設定値を表 2-36 に、AD_CMD_AUTO_READ_STOP コマンド使用例を図 2-50 に示します。

本コマンドには以下の制限があります。

- オートリード機能で設定済みの割り込み要因にのみ有効です。

表 2-36 e_adc_dma_event_t 型引数の設定値

設定値	内容
ADC_READ_ADI	A/D スキャン終了割り込み要求(ADC140_ADI)を DMA 起動要因としたオートリードを停止
ADC_READ_GBADI	グループ B の A/D スキャン終了割り込み要求(ADC140_GBADI) を DMA 起動要因としたオートリードを停止
ADC_READ_GCADI	グループ C の A/D スキャン終了割り込み要求(ADC140_GCADI) を DMA 起動要因としたオートリードを停止
ADC_READ_WCMPM	ウィンドウ A/B コンペア機能の条件一致割り込み要求 (ADC140_WCMPM) を DMA 起動要因としたオートリードを停止
ADC_READ_WCMPUM	ウィンドウ A/B コンペア機能の条件不一致割り込み要求 (ADC140_WCMPUM) を DMA 起動要因としたオートリードを停止

```
#include "r_adc_api.h"

static void callback(void);

// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

static uint16_t read_data[5]; /* DMA 転送先 RAM */

main()
{
    st_adc_pins_t scanset_pin;  /* ScanSet()  A/D 変換チャネル */
    st_adc_dma_read_info_t arg;    /* AD_CMD_AUTO_READ_NORMAL 第 2 引数 */
    e_adc_dma_event_t stop;    /* AD_CMD_AUTO_READ_STOP 第 2 引数 */

    (void)adcDev->Open(ADC_REPEAT_SCAN | ADC_14BIT, 0x10, NULL); /* S14AD ドライバ初期化 */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                        /* A/D 変換に AN000、AN001、AN006、AN022 を使用 */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* A/D 変換に温度センサを使用しない */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);

    arg.cb_event     = callback;                /* コールバック関数 */
    arg.dma_fact     = ADC_READ_ADI;         /* DMA 転送要因 */
    arg.src_addr     = (uint32_t)&S14AD->ADDR1; /* 転送元 A/D データレジスタ */
    arg.dest_addr    = (uint32_t)&read_data[0];    /* 転送先 RAM */
    arg.transfer_count = 5;                     /* 転送回数 5 回 */
    arg.block_size    = 0;                     /* ブロックサイズなし(0 固定) */
    arg.reg_size      = 0;                     /* 転送サイズ指定なし(16bit 固定) */

    (void)adcDev->Control(AD_CMD_AUTO_READ_NORMAL, &arg);
                                        /* DMA 転送による A/D 変換結果自動読み取り機能設定 */

    (void)adcDev->Start();  /* A/D 変換開始 */

    while(1)
    {
        if (/* オートリードを停止させる条件 */)
        {
            stop = ADC_READ_ADI;
            (void)adcDev->Control(AD_CMD_AUTO_READ_STOP, & stop);
                                            /* ADI 要求によるオートリードを停止 */
        }
    }
}

/***********************************************************************************************
* callback function
***********************************************************************************************/
static void callback(void)
{
    st_adc_dma_read_info_t restart = ADC_READ_ADI; /* AD_CMD_AUTO_READ_RESTART 第 2 引数 */
    (void)adcDev->Control(AD_CMD_AUTO_READ_RESTART, & restart);
                                            /* ADI 要求によるオートリードを再開 */
}
```

図 2-50　AD_CMD_AUTO_READ_STOP コマンド使用例

2.5.27 オートリードリスタート機能(AD_CMD_AUTO_READ_RESTART)

AD_CMD_AUTO_READ_RESTART コマンドでは、DMA 転送（DMAC もしくは DTC）による A/D 変換結果自動読み取り動作を最初から再度開始します(注)。第 2 引数には、e_adc_dma_event_t 型変数のポインタを指定してください。e_adc_dma_event_t 型引数の設定値を表 2-37 に、AD_CMD_AUTO_READ_RESTART コマンド使用例を図 2-51 に示します。

本コマンドには以下の制限があります。

- オートリード機能で設定済みの割り込み要因にのみ有効です。

注 転送元アドレス/転送先アドレス/転送回数/転送ブロックサイズを、オートリードコマンドで設定した値に書き戻し、再度動作を開始します。

表 2-37 e_adc_dma_event_t 型引数の設定値

設定値	内容
ADC_READ_ADI	A/D スキャン終了割り込み要求(ADC140_ADI)を DMA 起動要因としたオートリードを再開
ADC_READ_GBADI	グループ B の A/D スキャン終了割り込み要求(ADC140_GBADI) を DMA 起動要因としたオートリードを再開
ADC_READ_GCADI	グループ C の A/D スキャン終了割り込み要求(ADC140_GCADI) を DMA 起動要因としたオートリードを再開
ADC_READ_WCMPM	ウィンドウ A/B コンペア機能の条件一致割り込み要求 (ADC140_WCMPM) を DMA 起動要因としたオートリードを再開
ADC_READ_WCMPUM	ウィンドウ A/B コンペア機能の条件不一致割り込み要求 (ADC140_WCMPUM) を DMA 起動要因としたオートリードを再開

```
#include "r_adc_api.h"

static void callback(void);

// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

static uint16_t read_data[5]; /* DMA 転送先 RAM */

main()
{
    st_adc_pins_t scanset_pin;  /* ScanSet()  A/D 変換チャネル */
    st_adc_dma_read_info_t arg;    /* AD_CMD_AUTO_READ_NORMAL 第 2 引数 */

    (void)adcDev->Open(ADC_REPEAT_SCAN | ADC_14BIT, 0x10, NULL); /* S14AD ドライバ初期化 */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                        /* A/D 変換に AN000、AN001、AN006、AN022 を使用 */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* A/D 変換に温度センサを使用しない */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);

    arg.cb_event     = callback;                /* コールバック関数 */
    arg.dma_fact     = ADC_READ_ADI;         /* DMA 転送要因 */
    arg.src_addr     = (uint32_t)&S14AD->ADDR1; /* 転送元 A/D データレジスタ */
    arg.dest_addr    = (uint32_t)&read_data[0];    /* 転送先 RAM */
    arg.transfer_count = 5;                     /* 転送回数 5 回 */
    arg.block_size    = 0;                     /* ブロックサイズなし(0 固定) */
    arg.reg_size      = 0;                     /* 転送サイズ指定なし(16bit 固定) */

    (void)adcDev->Control(AD_CMD_AUTO_READ_NORMAL, &arg);
                                        /* DMA 転送による A/D 変換結果自動読み取り機能設定 */

    (void)adcDev->Start();  /* A/D 変換開始 */

    while(1);
}

/***********************************************************************************************
* callback function
***********************************************************************************************/
static void callback(void)
{
    e_adc_dma_event_t restart = ADC_READ_ADI; /* AD_CMD_AUTO_READ_RESTART 第 2 引数 */
    (void)adcDev->Control(AD_CMD_AUTO_READ_RESTART, & restart);
                                            /* ADI 要求によるオートリードを再開 */
}
```

図 2-51　AD_CMD_AUTO_READ_RESTART コマンド使用例

2.6 A/D 変換結果の取り込み方法

S14AD ドライバでは、2 種類の方法で A/D 変換結果を取り込むことができます。Read 関数を使用して任意のタイミングで A/D 変換結果を取り込む方法を 2.6.1 に、DMA 転送を使用して自動的に A/D 変換結果を取り込む方法（以下、オートリード機能）を 2.6.2 に示します。

2.6.1 Read 関数で A/D 変換結果を取り込む

S14AD ドライバでは、Read 関数を使用して A/D 変換結果を任意のタイミングで取り込むことができます。取り込むことのできる A/D データレジスタは、A/D データレジスタ y(ADDRy)(注)、A/D データ 2 重化レジスタ(ADDBLDR)、A/D 温度センサデータレジスタ(ADTSDR)、A/D 自己診断データレジスタ(ADRD)の 4 種類です。取り込み対象を指定する A/D データレジスタ定義を表 2-38 に、Read 関数の使用例を図 2-52 に示します。

表 2-38 A/D データレジスタ定義一覧

コマンド	内容
ADC_SSEL_ANy(注)	A/D データレジスタ y(ADDRy)(注)
ADC_SSEL_TEMP	A/D 温度センサデータレジスタ（ADTSDR）
ADC_SSEL_DBL	A/D データ 2 重化レジスタ（ADDBLDR）
ADC_SSEL_DIAG	A/D 自己診断データレジスタ（ADRD）
ADC_SSEL_VSC_VCC(注 2)	A/D VSC_VCC 端子電圧データレジスタ

注1. 1500KB グループ：y = 00～06、16、17、20～28
256KB グループ：y = 00～07、16、17、20～21

注2. 256KB のみ

```
#include "r_adc_api.h"

// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

main()
{
    st_adc_pins_t groupa_pin;  /* グループ A の A/D 変換チャネル */
    st_adc_status_info_t result_status; /* A/D ステータス格納変数 */
    uint16_t ad_data;

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* S14AD ドライバ初期化 */

    groupa_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN06;    /* A/D 変換に AN000 と AN006 を使用 */
    groupa_pin.sensor   = ADC_SENSOR_NOTUSE;   /* A/D 変換に温度センサを使用しない */
    (void)adcDev->ScanSet(ADC_GROUP_A, groupa_pin, ADC_TRIGER_SOFT); /* ソフトウェアトリガ */


    while (1)
    {
        (void)adcDev->Start();  /* A/D 変換開始 */

        do
        {
            (void)adcDev->Control(AD_CMD_GET_AD_STATE, & result_status);    /* A/D の状態を取得 */
        }
        while (ADC_STATE_RUN == result_status.ad_info);    /* A/D 変換完了待ち */

        (void)adcDev->Read(ADC_SSEL_AN00, &ad_data);    /* ADDR00 の A/D データを ad_data に取得 */

    }

}
```

図 2-52 Read 関数使用例

2.6.2 DMA 転送で A/D 変換結果を自動的に取り込む（オートリード機能）

S14AD ドライバでは DMA 転送（DMAC もしくは DTC）を使用して、A/D 変換結果を指定回数分自動的に取り込むことができます。オートリード機能を使用する場合、Control 関数でオートリードコマンド(AD_CMD_AUTO_READ_NORMAL/AD_CMD_AUTO_READ_BLOCK/AD_CMD_AUTO_READ_COMPARE)を実行し、設定を行ってください。オートリード機能では、DMA 転送起動要因(注 1)、転送元レジスタ、転送先 RAM、転送回数、コールバック関数(注 2)、転送サイズ(注 1)を指定できます。

オートリード機能を使用すると、任意の RAM に指定したレジスタの値が自動的に格納されます。指定回数分のオートリードが完了すると、オートリード動作は停止します。また、AD_CMD_AUTO_READ_STOP コマンドを実行することで、オートリード動作を停止させることができます。再度同じ設定でオートリード動作を行う場合は、AD_CMD_AUTO_READ_RESTART コマンドを実行してください。設定を変更する場合は、オートリードコマンドにて設定を上書きしてください。

注1. 使用するオートリードコマンドによって設定が異なります。各コマンドの設定値については表 2-39 で確認してください。

注2. DMA 転送完了を通知する割り込みには、Control 関数で指定したコールバック関数が使用されます。（Open 関数で指定したコールバック関数は使用されません。）

表 2-39 オートリードコマンド一覧

定義	DMA 起動要因(注 1)	転送サイズ	内容
AD_CMD_AUTO_READ_NORMAL	ADC140_ADI ADC140_GBADI ADC140_GCADI	16 ビット固定	DMA ノーマル転送モードを使用します 設定した DMA 起動要因の要求発生時に、指定した A/D データレジスタの値を RAM に転送します DMA 起動要因には A/D スキャン完了要求のみ指定できます
AD_CMD_AUTO_READ_BLOCK	ADC140_ADI ADC140_GBADI ADC140_GCADI	16 ビット固定	DMA ブロック転送モードを使用します 設定した DMA 起動要因の要求発生時に、指定した A/D データレジスタの値を RAM に転送します DMA 起動要因には A/D スキャン完了要求のみ指定できます
AD_CMD_AUTO_READ_COMPARE	ADC140_WCMPM ADC140_WCMPUM	8 ビット、16 ビット、32 ビット(注 2)	DMA ノーマル転送モードを使用します 設定した DMA 起動要因の要求発生時に、指定した A/D データレジスタの値を RAM に転送します DMA 起動要因にはウィンドウ A/B コンペアー致/不一致要求のみ指定できます
AD_CMD_AUTO_READ_STOP	ADC140_ADI ADC140_GBADI ADC140_GCADI ADC140_WCMPM ADC140_WCMPUM	-	オートリード動作をストップさせます
AD_CMD_AUTO_READ_RESTART	ADC140_ADI ADC140_GBADI ADC140_GCADI ADC140_WCMPM ADC140_WCMPUM	-	オートリード動作をリスタートします

注1. DMA 起動要因は Control 関数の第 2 引数で指定します。DMA 起動要因定義一覧は表 2-33 を参照してください。

注2. 転送サイズは Control 関数の第 2 引数(reg_size)で指定します。AD_CMD_AUTO_READ_COMPARE コマンドを使用する場合は、必ず転送サイズを指定して下さい。AD_CMD_AUTO_READ_NORMAL および AD_CMD_AUTO_READ_BLOCK コマンドを使用する場合、転送サイズは 16 ビットサイズ固定となるため設定不要です。

表 2-40 st_adc_dma_read_info_t 型引数の設定値

要素	設定値	内容
cb_event	オートリード機能用コールバック関数 (注 1)(注 2)	指定した転送回数分の DMA 転送が完了したタイミングで呼び出すコールバック関数
dma_fact	ADC_READ_ADI(注 3)	A/D スキャン終了割り込み(ADC140_ADI)を DMA 起動要因に設定
	ADC_READ_GBADI(注 3)	グループ B の A/D スキャン終了割り込み(ADC140_GBADI)を DMA 起動要因に設定
	ADC_READ_GCADI(注 3)	グループ C の A/D スキャン終了割り込み(ADC140_GCADI)を DMA 起動要因に設定
	ADC_READ_WCMPM(注 4)	ウィンドウ A/B コンペア機能の条件一致割り込み(ADC140_WCMPM)を DMA 起動要因に設定
	ADC_READ_WCMPUM(注 4)	ウィンドウ A/B コンペア機能の条件不一致割り込み(ADC140_WCMPUM)を DMA 起動要因に設定
src_addr	A/D データレジスタ y(ADDRy) (1500KB グループ： y = 00～06、16、17、20～28 256KB グループ： y = 00～07、16、17、20～21)	転送元アドレス DMA 転送で自動的に読み取る A/D データレジスタを 1 つ指定
	A/D データ 2 重化レジスタ(ADDBLDR)	
	A/D 温度センサデータレジスタ（ADTSDR）	
	A/D VSC_VCC 端子電圧データレジスタ（ADVSCDR）(注 5)	
dest_addr	任意の RAM	転送先アドレス DMA 転送で自動的に読み取った A/D データレジスタの情報を格納する RAM を指定
transfer_count	[DMA 転送に DMAC を使用した場合] 0～65535 [DMA 転送に DTC を使用した場合] 1～65536	転送回数 DMAC 使用時に転送回数を 0 に指定すると、フリーランニングモードで動作します
block_size (注 6)	[DMA 転送に DMAC を使用した場合] 1～256 [DMA 転送に DTC を使用した場合] 1～1024	転送ブロックサイズ
reg_size(注 7)	ADC_READ_BYTE	転送サイズ：8 ビット転送
	ADC_READ_WORD	転送サイズ：16 ビット転送
	ADC_READ_LONG	転送サイズ：32 ビット転送

注1. DMA 転送完了タイミングにコールバックを発生させたい場合に指定してください。DMA 転送完了タイミングの通知が不要な場合、コールバック関数に NULL を指定してください。（NULL を指定した場合、割り込みは発生しません。）

注2. オートリード機能のコールバックでは、Control 関数実行時に指定した関数が呼び出されます（Open 関数で指定した関数ではありません）。コールバック関数は使用する DMA 転送要因ごとに指定してください。

注3. AD_CMD_AUTO_READ_NORMAL コマンドおよび AD_CMD_AUTO_READ_BLOCK コマンドでのみ使用できます。AD_CMD_AUTO_READ_COMPARE コマンドでは使用できません。

注4. AD_CMD_AUTO_READ_COMPARE コマンドでのみ使用できます。AD_CMD_AUTO_READ_NORMAL コマンドおよび AD_CMD_AUTO_READ_BLOCK コマンドでは使用できません。

注5. 256KB グループのみ

注6. AD_CMD_AUTO_READ_BLOCK コマンドでのみ有効です。

注7. AD_CMD_AUTO_READ_COMPARE コマンドでのみ有効です。

S14AD ドライバで DMA 転送を使用する場合、r_adc_cfg.h で対象の割り込み要求制御を DMAC もしくは DTC に変更してください。

オートリード機能と割り込み機能は、排他制御されます。オートリードコマンドで DMA 起動要因に指定した割り込み要求を、割り込み機能で使用することはできません。

表 2-41 A/D 割り込み要求制御

定義（注）	初期値	内容
S14AD_ADI_CONTROL	S14AD_USED_INTERRUPT	ADI 割り込み要求制御（初期値：割り込み）
S14AD_GBADI_CONTROL	S14AD_USED_INTERRUPT	GBADI 割り込み要求制御（初期値：割り込み）
S14AD_GCADI_CONTROL	S14AD_USED_INTERRUPT	GCADI 割り込み要求制御（初期値：割り込み）
S14AD_WCMPM_CONTROL	S14AD_USED_INTERRUPT	ウィンドウ A/B コンペアマッチ割り込み要求制御（初期値：割り込み）
S14AD_WCMPUM_CONTROL	S14AD_USED_INTERRUPT	ウィンドウ A/B コンペアアンマッチ割り込み要求制御（初期値：割り込み）

注　CMPAI、CMPBI イベントは割り込みのみ使用可能（DMA 起動不可）

表 2-42 A/D 割り込み要求制御方法の定義

定義	値	内容
S14AD_USED_INTERRUPT	(0)	A/D スキャン終了またはウィンドウ A/B コンペア一致/不一致割り込み要求を、割り込みもしくはポーリングで使用
S14AD_USED_DMAC0	(1<<0)	A/D スキャン終了またはウィンドウ A/B コンペア一致/不一致割り込み要求で DMAC0 を起動 コールバック関数が指定されている場合、指定回数の DMAC0 転送指定回数完了タイミングを DMAC0 完了割り込みで通知
S14AD_USED_DMAC1	(1<<1)	A/D スキャン終了またはウィンドウ A/B コンペア一致/不一致割り込み要求で DMAC1 を起動 コールバック関数が指定されている場合、DMAC1 転送指定回数完了タイミングを DMAC1 完了割り込みで通知
S14AD_USED_DMAC2	(1<<2)	A/D スキャン終了またはウィンドウ A/B コンペア一致/不一致割り込み要求で DMAC2 を起動 コールバック関数が指定されている場合、DMAC2 転送指定回数完了タイミングを DMAC2 完了割り込みで通知
S14AD_USED_DMAC3	(1<<3)	A/D スキャン終了またはウィンドウ A/B コンペア一致/不一致割り込み要求で DMAC3 を起動 コールバック関数が指定されている場合、DMAC3 転送指定回数完了タイミングを DMAC3 完了割り込みで通知
S14AD_USED_DTC	(1<<15)	A/D スキャン終了またはウィンドウ A/B コンペア一致/不一致割り込み要求で DTC を起動 コールバック関数が指定されている場合、DTC 転送指定回数完了タイミングを DTC 起動要因の割り込みで通知

オートリードコマンド実行時にコールバック関数を指定した場合は、DMA 転送完了タイミングでコールバックが実行されます。コールバックを使用する場合は、DMA 転送完了を通知する割り込み要因を r_system_cfg.h にて、ネスト型ベクタ割り込みコントローラ（以下、NVIC）に登録する必要があります。DMA 起動要因に対する NVIC の登録定義を表 2-43 に、NVIC への割り込み登録例を図 2-53 に示します。

表 2-43　DMA 起動要因に対する NVIC の登録定義

DMA 転送制御	DMA 起動要因	NVIC 登録定義	備考
DTC	シングルスキャンもしくはグループ A の A/D スキャン終了割り込み要求(ADC140_ADI)	SYSTEM_CFG_EVENT_NUMBER_ADC140_ADI	
	グループ B の A/D スキャン終了割り込み要求(ADC140_GBADI)	SYSTEM_CFG_EVENT_NUMBER_ADC140_GBADI	
	グループ C の A/D スキャン終了割り込み(ADC140_GCADI)	SYSTEM_CFG_EVENT_NUMBER_ADC140_GCADI	
	ウィンドウ A/B コンペア機能の条件一致割り込み要求(ADC140_WCMPM)	SYSTEM_CFG_EVENT_NUMBER_ADC140_WCMPM	
	ウィンドウ A/B コンペア機能の条件不一致割り込み要求(ADC140_WCMPUM)	SYSTEM_CFG_EVENT_NUMBER_ADC140_WCMPUM	
DMAC	- (注 1)	SYSTEM_CFG_EVENT_NUMBER_DMACm_INT	m=0～3 (注 2)

注1. DMAC を使用する場合、起動要因ごとの割り込み設定は不要です。DMA 転送完了タイミングをコールバックで通知する場合は、SYSTEM_CFG_EVENT_NUMBER_DMACm_INT を設定してください。
注2. コールバック関数に NULL を設定した場合（コールバック未使用時）、この設定は不要です。

```
・・・
#define SYSTEM_CFG_EVENT_NUMBER_WDT_NMIUNDF
        (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)  /*!< Numbers 0/4/8/12/16/20/24/28 only */
#define SYSTEM_CFG_EVENT_NUMBER_ADC140_ADI
        (SYSTEM_IRQ_EVENT_NUMBER0)            /*!< Numbers 0/4/8/12/16/20/24/28 only */
#define SYSTEM_CFG_EVENT_NUMBER_ADC140_WCMPM
        (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)  /*!< Numbers 0/4/8/12/16/20/24/28 only */
・・・
```

図 2-53 NVIC への割り込み登録例（DTC 制御で ADC140_ADI を起動要因とする場合）

オートリード機能で DMAC を使用する場合のコンフィグ設定(r_adc_cfg.h、r_system_cfg.h)を図 2-54、図 2-55 に、DTC を使用する場合の設定を図 2-56、図 2-57 に示します。

```
・・・
#define S14AD_ADI_CONTROL    S14AD_USED_DMAC0   ///< Read control of AD conversion value by group A scan
end event
#define S14AD_GBADI_CONTROL  S14AD_USED_INTERRUPT   ///< Read control of AD conversion value by group B scan
end event
#define S14AD_GCADI_CONTROL  S14AD_USED_INTERRUPT   ///< Read control of AD conversion value by group C scan
end event
#define S14AD_WCMPM_CONTROL  S14AD_USED_INTERRUPT   ///< Read control of AD conversion value by WCMPM
#define S14AD_WCMPUM_CONTROL S14AD_USED_INTERRUPT   ///< Read control of AD conversion value by WCMPUM
・・・
```

図 2-54 r_adc_cfg.h 設定例（ADI 割り込み要求制御に DMAC0 を使用）

```
・・・
#define SYSTEM_CFG_EVENT_NUMBER_PORT_IRQ0   (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)  /*!< Numbers
0/4/8/12/16/20/24/28 only */
#define SYSTEM_CFG_EVENT_NUMBER_DMAC0_INT   (SYSTEM_IRQ_EVENT_NUMBER0)
/*!< Numbers 0/4/8/12/16/20/24/28 only */
#define SYSTEM_CFG_EVENT_NUMBER_DTC_COMPLETE   (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)  /*!< Numbers
0/4/8/12/16/20/24/28 only */
・・・
```

図 2-55 r_system_cfg.h 設定例（DMAC0 転送完了割り込みを使用）

```
・・・
#define S14AD_ADI_CONTROL    S14AD_USED_DTC   ///< Read control of AD conversion value by group A scan end
event
#define S14AD_GBADI_CONTROL  S14AD_USED_INTERRUPT   ///< Read control of AD conversion value by group B scan
end event
#define S14AD_GCADI_CONTROL  S14AD_USED_INTERRUPT   ///< Read control of AD conversion value by group C scan
end event
#define S14AD_WCMPM_CONTROL  S14AD_USED_INTERRUPT   ///< Read control of AD conversion value by WCMPM
#define S14AD_WCMPUM_CONTROL S14AD_USED_INTERRUPT   ///< Read control of AD conversion value by WCMPUM
・・・
```

図 2-56 r_adc_cfg.h 設定例（ADI 割り込み要求制御に DTC を使用）

```
・・・
#define SYSTEM_CFG_EVENT_NUMBER_WDT_NMIUNDF
       (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)  /*!< Numbers 0/4/8/12/16/20/24/28 only */
#define SYSTEM_CFG_EVENT_NUMBER_ADC140_ADI
       (SYSTEM_IRQ_EVENT_NUMBER0)            /*!< Numbers 0/4/8/12/16/20/24/28 only */
#define SYSTEM_CFG_EVENT_NUMBER_ADC140_WCMPM
       (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)  /*!< Numbers 0/4/8/12/16/20/24/28 only */
・・・
```

図 2-57 r_system_cfg.h 設定例（DTC 転送完了通知に ADI 割り込みを使用）

オートリード機能を使用する場合のサンプルコードを図 2-58 に示します。

```
#include "r_adc_api.h"

static void callback(void);

// S14AD driver instance
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *adcDev = &Driver_S14AD;

static uint16_t read_data[5]; /* DMA 転送先 RAM */

main()
{
    st_adc_pins_t scanset_pin;  /* ScanSet()  A/D 変換チャネル */
    st_adc_dma_read_info_t arg;    /* AD_CMD_AUTO_READ_NORMAL 第 2 引数 */

    (void)adcDev->Open(ADC_REPEAT_SCAN | ADC_14BIT, 0x10, NULL); /* S14AD ドライバ初期化 */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                        /* A/D 変換に AN000、AN001、AN006、AN022 を使用 */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* A/D 変換に温度センサを使用しない */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);

    arg.cb_event     = callback;                 /* コールバック関数 */
    arg.dma_fact     = ADC_READ_ADI;         /* DMA 転送要因 */
    arg.src_addr     = (uint32_t)&S14AD->ADDR1; /* 転送元 A/D データレジスタ */
    arg.dest_addr    = (uint32_t)&read_data[0];    /* 転送先 RAM */
    arg.transfer_count = 5;                      /* 転送回数 5 回 */
    arg.block_size    = 0;                      /* ブロックサイズなし(0 固定) */
    arg.reg_size      = 0;                      /* 転送サイズ指定なし(16 ビット固定) */

    (void)adcDev->Control(AD_CMD_AUTO_READ_NORMAL, &arg);
                                       /* DMA 転送による A/D 変換結果自動読み取り機能設定 */

    (void)adcDev->Start();  /* A/D 変換開始 */

    while(1);
}

/***********************************************************************************************
* callback function
***********************************************************************************************/
static void callback(void)
{
    /* DMA 転送が指定回数分終了した場合の処理を記述 */
    /* 以下、DMA 転送を再度実行する場合の処理例 */
    e_adc_dma_event_t restart = ADC_READ_ADI; /* AD_CMD_AUTO_READ_RESTART 第 2 引数 */
    (void)adcDev->Control(AD_CMD_AUTO_READ_RESTART, & restart);
                                            /* ADI 要求によるオートリードを再開 */
}
```

図 2-58 オートリード機能使用例（DMAC/DTC 共通）

2.7 マクロ/型定義

S14AD ドライバで、ユーザが参照可能なマクロ/型定義を r_adc_api.h ファイルで定義しています。

2.7.1 S14AD 初期設定コード定義

S14AD 初期設定コード定義は、Open 関数の第 1 引数で使用する A/D スキャンモードおよび A/D 初期設定定義です。S14AD 初期設定定義は、スキャンモード設定、A/D 変換精度定義、A/D データフォーマット定義で構成されます。

S14AD 初期設定コード定義の構成を図 2-59 に、各設定定義を表 2-44～表 2-46 に示します。

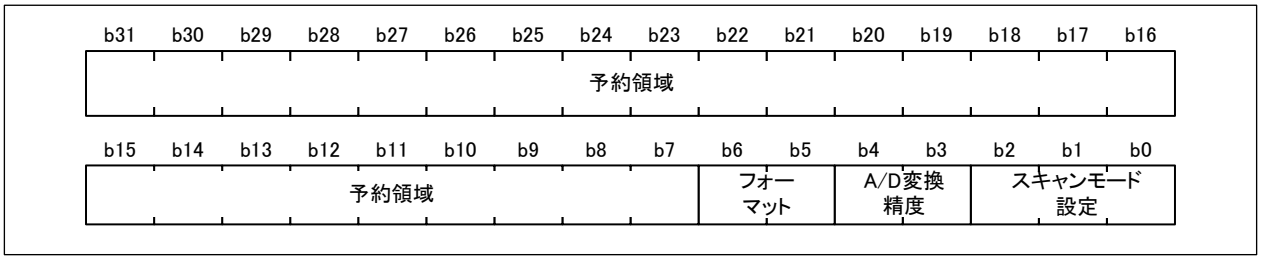


図 2-59 S14AD 初期設定コード定義の構成

表 2-44 スキャンモード定義一覧

コマンド	値	内容
ADC_SINGLE_SCAN(デフォルト)	0U << ADC_MODE_POS	シングルスキャンモード
ADC_GROUP_SCAN	1U << ADC_MODE_POS	グループスキャンモード
ADC_REPEAT_SCAN	2U << ADC_MODE_POS	連続スキャンモード

表 2-45 A/D 変換精度定義一覧

コマンド	値	内容
ADC_14BIT (デフォルト)	0U << ADC_RESOLUTION_POS	14 ビット精度
ADC_12BIT	1U << ADC_RESOLUTION_POS	12 ビット精度

表 2-46 A/D データレジスタフォーマット定義一覧

コマンド	値	内容
ADC_RIGHT (デフォルト)	0U << ADC_FORMAT_POS	右詰
ADC_LEFT	1U << ADC_FORMAT_POS	左詰

2.7.2 A/D 変換開始トリガ定義

A/D 変換開始トリガ定義は、ScanSet 関数の第 2 引数で使用する A/D 変換開始トリガを指定するための定義です。

表 2-47 A/D 変換開始条件定義一覧

コマンド	値	内容
ADC_TRIGER_SOFT	(00)	ソフトウェアトリガ
ADC_TRIGER_TMR	(01)	TCORA レジスタと TCNT カウンタのコンペアマッチ
ADC_TRIGER_ELC	(02)	ELC_S14AD
ADC_TRIGER_ADTRG	(03)	ADTRG0 入力
ADC_TRIGER_LOW_PRIORITY_CONT_SCAN	(04)	トリガ要因非選択

2.7.3 A/D 変換チャネル複数選択定義

A/D 変換チャネル複数選択定義は、A/D 変換チャネルを複数組み合わせて指定するための定義です。

表 2-48 A/D 変換チャネル複数選択定義一覧

定義	値	内容
ADC_MSEL_AN00	(1 << 0)	AN000 チャネル選択
ADC_MSEL_AN01	(1 << 1)	AN001 チャネル選択
ADC_MSEL_AN02	(1 << 2)	AN002 チャネル選択
ADC_MSEL_AN03	(1 << 3)	AN003 チャネル選択
ADC_MSEL_AN04	(1 << 4)	AN004 チャネル選択
ADC_MSEL_AN05	(1 << 5)	AN005 チャネル選択
ADC_MSEL_AN06	(1 << 6)	AN006 チャネル選択
ADC_MSEL_AN07(注 1)	(1 << 7)	AN007 チャネル選択
ADC_MSEL_AN16	(1 << 16)	AN016 チャネル選択
ADC_MSEL_AN17	(1 << 17)	AN017 チャネル選択
ADC_MSEL_AN20	(1 << 20)	AN020 チャネル選択
ADC_MSEL_AN21	(1 << 21)	AN021 チャネル選択
ADC_MSEL_AN22(注 2)	(1 << 22)	AN022 チャネル選択
ADC_MSEL_AN23(注 2)	(1 << 23)	AN023 チャネル選択
ADC_MSEL_AN24(注 2)	(1 << 24)	AN024 チャネル選択
ADC_MSEL_AN25(注 2)	(1 << 25)	AN025 チャネル選択
ADC_MSEL_AN26(注 2)	(1 << 26)	AN026 チャネル選択
ADC_MSEL_AN27(注 2)	(1 << 27)	AN027 チャネル選択
ADC_MSEL_AN28(注 2)	(1 << 28)	AN028 チャネル選択
ADC_MSEL_VSC_VCC(注 1)	(1 << 31)	VSC_VCC 端子電圧出力選択

注1. 256KB グループのみ

注2. 1500KB グループのみ

2.7.4 温度センサ出力使用定義

温度センサ出力使用定義は、A/D 変換に温度センサ出力を指定する場合に使用する定義です。

表 2-49 温度センサ出力使用定義一覧

コマンド	値	内容
ADC_SENSOR_NOTUSE	(00)	温度センサ出力を使用しない
ADC_MSEL_TEMP	(1 << 0)	温度センサ出力を使用する

2.7.5 A/D 変換チャネル単数選択定義

A/D 変換チャネル単数選択定義は、A/D 変換チャネルを単数指定するための定義です。

表 2-50 A/D 変換チャネル単数選択定義一覧

定義	値	内容
ADC_SSEL_AN00	(00)	AN000 チャネル選択
ADC_SSEL_AN01	(01)	AN001 チャネル選択
ADC_SSEL_AN02	(02)	AN002 チャネル選択
ADC_SSEL_AN03	(03)	AN003 チャネル選択
ADC_SSEL_AN04	(04)	AN004 チャネル選択
ADC_SSEL_AN05	(05)	AN005 チャネル選択
ADC_SSEL_AN06	(06)	AN006 チャネル選択
ADC_SSEL_AN07(注 1)	(07)	AN007 チャネル選択
ADC_SSEL_AN16	(16)	AN016 チャネル選択
ADC_SSEL_AN17	(17)	AN017 チャネル選択
ADC_SSEL_AN20	(20)	AN020 チャネル選択
ADC_SSEL_AN21	(21)	AN021 チャネル選択
ADC_SSEL_AN22(注 2)	(22)	AN022 チャネル選択
ADC_SSEL_AN23(注 2)	(23)	AN023 チャネル選択
ADC_SSEL_AN24(注 2)	(24)	AN024 チャネル選択
ADC_SSEL_AN25(注 2)	(25)	AN025 チャネル選択
ADC_SSEL_AN26(注 2)	(26)	AN026 チャネル選択
ADC_SSEL_AN27(注 2)	(27)	AN027 チャネル選択
ADC_SSEL_AN28(注 2)	(28)	AN028 チャネル選択
ADC_SSEL_VSC_VCC(注 1)	(31)	VSC_VCC 端子電圧出力
ADC_SSEL_TEMP	(32)	温度センサ出力選択
ADC_SSEL_DBL	(50)	ダブルトリガ選択
ADC_SSEL_DIAG	(51)	自己診断選択
ADC_SSEL_VSC_VCC	(32)	VSC_VCC 端子電圧出力選択

注1. 256KB グループのみ

注2. 1500KB グループのみ

2.7.6 A/D 変換グループ定義

A/D 変換グループ定義は、設定を行うグループを指定するための定義です。

表 2-51 A/D グループ定義一覧

コマンド	値	内容
ADC_GROUP_A	(00)	グループ A
ADC_GROUP_B	(01)	グループ B
ADC_GROUP_C	(02)	グループ C

2.7.7 Control 関数制御コマンド定義

Control 関数制御コマンド定義は、Control 関数の第 1 引数で使用する機能設定用コマンド定義です。本定義は"AD_CMD_SET_ADD_MODE"を先頭とした列挙型定義です。

表 2-52 Control 関数制御コマンド一覧

定義	内容
AD_CMD_SET_ADD_MODE	加算モード設定コマンド
AD_CMD_SET_DBLTRG	ダブルトリガモード有効設定コマンド
AD_CMD_SET_DIAG	自己診断モード有効設定コマンド
AD_CMD_SET_ADI_INT	ADI_INT の割り込み設定コマンド
AD_CMD_SET_GROUPB_INT	グループ B の割り込み設定コマンド
AD_CMD_SET_GROUPC_INT	グループ C の割り込み設定コマンド
AD_CMD_SET_WCMPM_INT	WCMPM の割り込み設定コマンド
AD_CMD_SET_WCMPUM_INT	WCMPUM の割り込み設定コマンド
AD_CMD_SET_AUTO_CLEAR	自動クリアの有効/無効設定コマンド
AD_CMD_SET_SAMPLING_AN000	AN000、自己診断のサンプリングステート変更コマンド
AD_CMD_SET_SAMPLING_AN001	AN001 のサンプリングステート変更コマンド
AD_CMD_SET_SAMPLING_AN002	AN002 のサンプリングステート変更コマンド
AD_CMD_SET_SAMPLING_AN003	AN003 のサンプリングステート変更コマンド
AD_CMD_SET_SAMPLING_AN004	AN004 のサンプリングステート変更コマンド
AD_CMD_SET_SAMPLING_AN005	AN005 のサンプリングステート変更コマンド
AD_CMD_SET_SAMPLING_AN006	AN006 のサンプリングステート変更コマンド
AD_CMD_SET_SAMPLING_AN007(注 1)	AN007 のサンプリングステート変更コマンド
AD_CMD_SET_SAMPLING_AN016_AN017_AN020_TO_AN028(注 2)	AN016、017、020～028 のサンプリングステート変更コマンド
AD_CMD_SET_SAMPLING_AN016_AN017_AN020_AN021_VSC_VCC(注 1)	AN016、017、020、021、VSC_VCC 端子出力のサンプリングステート変更コマンド
AD_CMD_SET_SAMPLING_TEMP	温度センサのサンプリングステート変更コマンド
AD_CMD_SET_ADNDIS	断線検出アシスト設定コマンド
AD_CMD_SET_GROUP_PRIORITY	グループ優先順位の設定コマンド
AD_CMD_SET_WINDOWA	ウィンドウ A の比較条件設定コマンド
AD_CMD_SET_WINDOWB	ウィンドウ B の比較条件設定コマンド
AD_CMD_SET_ELC	AD140_ELC の発生条件設定コマンド
AD_CMD_GET_CMP_RESULT	ウィンドウの比較結果取得コマンド
AD_CMD_GET_AD_STATE	AD コンバータの状態取得コマンド
AD_CMD_USE_VREFL0	低電位基準電圧 VREFL0 選択コマンド
AD_CMD_USE_VREFH0	高電位基準電圧 VREFH0 選択コマンド
AD_CMD_SCLK_ENABLE	サブクロック選択コマンド
AD_CMD_CALIBRATION	オフセットキャリブレーション実施コマンド
AD_CMD_STOP_TRIG	AD 変換停止、かつ A/D 変換トリガ要因クリアコマンド
AD_CMD_AUTO_READ_NORMAL	DMA 転送（ノーマル）による自動読み取り設定コマンド
AD_CMD_AUTO_READ_BLOCK	DMA 転送（ブロック）による自動読み取り設定コマンド
AD_CMD_AUTO_READ_COMPARE	WCMPM/WCMPUM を起動要因とする DMA 転送（ノーマル）による自動読み取り設定コマンド
AD_CMD_AUTO_READ_STOP	DMA 転送停止コマンド
AD_CMD_AUTO_READ_RESTART	DMA 転送再開コマンド

注1. 256KB グループのみ

注2. 1500KB グループのみ

2.7.8 加算/平均モード設定定義

加算/平均モード設定用定義は、AD_CMD_SET_ADD_MODE コマンド実行時の第 2 引数に使用する定義です。

表 2-53 加算/平均モード設定用定義一覧

定義	値	内容
ADC_ADD_OFF	(0)	A/D 変換加算/平均モードオフ
ADC_ADD_2_SAMPLES	(1)	加算モード/2 回変換(1 回加算)
ADC_ADD_4_SAMPLES	(3)	加算モード/4 回変換(3 回加算)
ADC_ADD_16_SAMPLES	(5)	加算モード/16 回変換(15 回加算)
ADC_ADD_AVG_2_SAMPLES	(0x81)	平均モード/2 回変換(1 回加算)
ADC_ADD_AVG_4_SAMPLES	(0x83)	平均モード/4 回変換(3 回加算)
ADC_ADD_AVG_16_SAMPLES	(0x85)	平均モード/16 回変換(15 回加算)

2.7.9 自己診断モード設定定義

自己診断モード設定定義は、AD_CMD_SET_DIAG コマンド実行時の第 2 引数に使用する定義です。

表 2-54 自己診断モード設定定義一覧

定義	値	内容
AD_DIAG_DISABLE	(00)	自己診断無効
AD_DIAG_0V	(01)	0V 電圧
AD_DIAG_HARF	(02)	基準電圧電源(VREFH0)×1/2 電圧
AD_DIAG_BASE	(03)	基準電圧電源(VREFH0)電圧
AD_DIAG_ROTATE	(04)	自己診断電圧ローテーションモード

2.7.10 割り込み機能設定定義

割り込み機能設定定義は、割り込み機能設定コマンド(注)実行時に使用する定義です。

注 AD_CMD_SET_ADI_INT、AD_CMD_SET_GROUPB_INT、AD_CMD_SET_GROUPC_INT、AD_CMD_SET_WCMPM_INT、AD_CMD_SET_WCMPUM_INT

表 2-55 割り込み機能設定定義一覧

定義	値	内容
ADC_INT_DISABLE	(00)	割り込み禁止
ADC_INT_POLLING	(01)	ポーリング
ADC_INT_ENABLE	(02)	割り込み許可

2.7.11 S14AD 許可/禁止定義

S14AD 許可/禁止定義は、機能の許可/禁止設定を行うための定義です。

表 2-56 第 2 引数の設定値

定義	値	内容
ADC_ENABLE	(01)	A/D データレジスタ自動クリア有効
ADC_DISABLE	(00)	A/D データレジスタ自動クリア無効

2.7.12 断線検出アシスト設定定義

断線検出アシスト設定定義は、AD_CMD_SET_ADNDIS コマンド実行時の第 2 引数に使用する定義です。プリチャージ/ディスチャージの設定を行う場合、プリチャージ/ディスチャージ期間と組み合わせて指定します。

使用例）

プリチャージ期間 5 サイクル

```
uint8_t arg = ADC_DDA_DISCHARGE | 5;
adcDrv -> Control(ADC_DDA_PRECHARGE, &arg);
```

表 2-57 断線検出アシスト設定定義一覧

定義	値	内容
ADC_DDA_OFF	(0x00)	断線検出アシスト機能無効
ADC_DDA_PRECHARGE	(0x10)	プリチャージ
ADC_DDA_DISCHARGE	(0x00)	ディスチャージ

2.7.13 グループ優先動作設定定義

グループ優先動作設定定義は、AD_CMD_SET_GROUP_PRIORITY コマンド実行時の第 2 引数に使用する定義です。

表 2-58 グループ優先動作設定定義一覧

定義	値	内容
ADC_GSP_PRIORYTY_OFF	(0x0000)	グループ優先動作無効
ADC_GSP_WAIT_TRG	(0x0001)	低優先グループ再起動無効
ADC_GSP_SCAN_BIGIN	(0x0003)	低優先グループ再起動有効
ADC_GSP_SCAN_RESTART	(0x4003)	A/D 変換未終了チャネルから再スキャン
ADC_GSP_WAIT_TRG_CONT_SCAN	(0x8001)	シングルスキャン連続動作有効かつ低優先グループ再起動無効
ADC_GSP_SCAN_BIGIN_CONT_SCAN	(0x8003)	シングルスキャン連続動作有効かつ低優先グループ再起動有効
ADC_GSP_SCAN_RESTART_CONT_SCAN	(0xC003)	シングルスキャン連続動作有効かつ A/D 変換未終了チャネルから再スキャン

2.7.14 コンペア機能ウィンドウ A 設定定義

コンペア機能ウィンドウ A 設定定義は、AD_CMD_SET_WINDOWA コマンド実行時の第 2 引数に使用する定義です。

表 2-59 コンペア機能ウィンドウ A 設定定義一覧

定義	値	内容
ADC_CMP_OFF	(00)	コンペア比較しない
ADC_CMP_LEVEL	(01)	レベル比較
ADC_CMP_WINDOW	(02)	ウィンドウ比較

2.7.15 コンペア機能ウィンドウ B 設定定義

コンペア機能ウィンドウ B 設定定義は、AD_CMD_SET_WINDOWB コマンド実行時の第 2 引数に使用する定義です。

表 2-60 コンペア機能ウィンドウ B 設定定義一覧

設定値	値	内容
ADC_WINB_LEVEL_BELOW	(00)	レベル 1 以下
ADC_WINB_LEVEL_ABOVE	(01)	レベル 1 以上
ADC_WINB_WINDOW_OUTSIDE	(02)	レベル 1 とレベル 2 の範囲外
ADC_WINB_WINDOW_BETWEEN	(03)	レベル 1 とレベル 2 の範囲内

2.7.16 コンペア機能ウィンドウ A/B 複合条件設定定義

コンペア機能ウィンドウ A/B 複合条件設定定義は、AD_CMD_SET_WINDOWB コマンド実行時の第 2 引数に使用する定義です。

表 2-61 コンペア機能ウィンドウ A/B 複合条件設定定義一覧

設定値	値	内容
ADC_COMB_OR	(00)	ウィンドウ A 比較条件に一致 OR ウィンドウ B 比較条件に一致で ADC140_WCMPM を出力、それ以外は ADC140_WCMPUM を出力
ADC_COMB_EXOR	(01)	ウィンドウ A 比較条件に一致 EXOR ウィンドウ B 比較条件に一致で ADC140_WCMPM を出力、それ以外は ADC140_WCMPUM を出力
ADC_COMB_AND	(02)	ウィンドウ A 比較条件に一致 AND ウィンドウ B 比較条件に一致で ADC140_WCMPM を出力、それ以外は ADC140_WCMPUM を出力
ADC_COMB_NON_EVENT	(03)	ウィンドウ A のみでのイベント出力(注)
ADC_COMB_OFF	(0xFF)	ウィンドウ A/B 複合条件オフ

注 ADC_COMB_NON_EVENT を指定した場合、ウィンドウ B の設定は引数の設定値にかかわらず以下になります。
・ウィンドウ A/B 複合条件：OR 条件（ADCMPCR.CMPAB = 00b）
・ウィンドウ B 比較対象チャネル：非選択(ADCMPBNSR.CMPCHB = 0x3F)
・ウィンドウ B 比較条件：「常に不一致」となる「0 < 結果 < 0」
(ADCMPCR.WCMPE = 1、ADWINLLB[15:0] = ADWINULB[15:0] = 0000h、ADCMPBNSR.CMPLB = 1)

2.7.17 S14AD ステータスコード定義

S14AD ステータスコードは、AD_CMD_GET_AD_STATE コマンドを実行時に取得する S14AD の状態を表すコードです。A/D 変換動作状態定義を表 2-62 に、割り込み要因ごとの状態定義を表 2-63 に示します。

表 2-62 A/D 変換動作状態定義一覧

定義	値	内容
ADC_STATE_STOP	(00)	A/D 変換停止中(ADCSR.ADST = 0)
ADC_STATE_RUN	(01)	A/D 変換動作中(ADCSR.ADST = 1)

表 2-63 割り込み要因ごとの状態定義一覧

定義	値	内容
ADC_CONV_NOT_POLLING	(00)	ポーリング設定ではありません
ADC_CONV_INCOMPLETE	(01)	A/D 変換が完了していません
ADC_CONV_COMPLETE	(02)	A/D 変換が完了しています
ADC_CONV_GET_FAILED	(03)	A/D 変換動作状態の取得に失敗しました
ADC_CONV_WCMP_DETECT	(04)	ウィンドウコンペアイベント（ADC140_WCMPM もしくは ADC140_WCMPUM）が検出されました
ADC_CONV_WCMP_NOT_DETECT	(05)	ウィンドウコンペアイベント（ADC140_WCMPM もしくは ADC140_WCMPUM）は検出されていません

2.7.18 オートリード機能設定定義

オートリード機能設定定義は、オートリード機能設定コマンド実行時の第 2 引数に使用する定義です。DMA 転送要因指定定義一覧を表 2-64 に、DMA 転送サイズ指定定義一覧を表 2-65 に示します。

表 2-64 DMA 転送要因指定定義一覧

定義	値	内容
ADC_READ_ADI	(00)	A/D スキャン終了割り込み要求(ADC140_ADI)を DMA 起動要因に設定
ADC_READ_GBADI	(01)	グループ B の A/D スキャン終了割り込み要求(ADC140_GBADI)を DMA 起動要因に設定
ADC_READ_GCADI	(02)	グループ C の A/D スキャン終了割り込み要求(ADC140_GCADI)を DMA 起動要因に設定
ADC_READ_WCMPM	(03)	ウィンドウ A/B コンペア機能の条件一致割り込み要求(ADC140_WCMPM)を DMA 起動要因に設定
ADC_READ_WCMPUM	(04)	ウィンドウ A/B コンペア機能の条件不一致割り込み要求(ADC140_WCMPUM)を DMA 起動要因に設定

表 2-65 DMA 転送サイズ指定定義一覧

定義	値	内容
ADC_READ_BYTE	(00)	8 ビット転送
ADC_READ_WORD	(01)	16 ビット転送
ADC_READ_LONG	(02)	32 ビット転送

2.7.19 スキャン終了イベント(ADC140_ELC)発生条件設定定義

スキャン終了イベント(ADC140_ELC)イベント発生条件設定定義は、AD_CMD_SET_ELC コマンド実行時の第 2 引数に使用する定義です。

表 2-66 スキャン終了イベント(ADC140_ELC)設定定義一覧

定義	値	内容
ADC_ELC_GROUPA	(0x00)	グループ B とグループ C のスキャン終了を除くスキャン終了時にイベント発生
ADC_ELC_GROUPB	(0x01)	グループ B スキャン終了時にイベント発生
ADC_ELC_GROUPC	(0x04)	グループ C スキャン終了時にイベント発生
ADC_ELC_ALL	(0x02)	すべてのスキャン終了時にイベント発生

2.7.20 S14AD イベントコード定義

Open 関数で指定したコールバック関数で通知されるイベント定義です。

表 2-67 S14AD イベントコード一覧

定義	値	内容
ADC_EVT_SCAN_COMPLETE	(00)	シングルスキャンもしくはグループ A の A/D スキャン完了
ADC_EVT_SCAN_COMPLETE_GROUPB	(01)	グループ B の A/D スキャン完了
ADC_EVT_SCAN_COMPLETE_GROUPC	(02)	グループ C の A/D スキャン完了
ADC_EVT_CONDITION_MET	(03)	ウィンドウ A の比較条件一致
ADC_EVT_CONDITION_METB	(04)	ウィンドウ B の比較条件一致
ADC_EVT_WINDOW_CMP_MATCH	(05)	ウィンドウ A/B コンペア機能の条件一致
ADC_EVT_WINDOW_CMP_UNMATCH	(06)	ウィンドウ A/B コンペア機能の条件不一致

2.7.21 S14AD エラーコード定義

S14AD のエラーコード定義です。

表 2-68 S14AD エラーコード一覧

定義	値	内容
ADC_OK	(00)	正常終了
ADC_ERROR	(01)	S14AD ドライバ設定が不正です
ADC_ERROR_BUSY	(02)	A/D 変換が実行中です
ADC_ERROR_PARAMETER	(03)	引数が不正です
ADC_ERROR_MODE	(04)	モード設定が不正です
ADC_ERROR_LOCKED	(05)	S14AD モジュールがロックされています
ADC_ERROR_SYSTEM_SETTING	(06)	システム設定が不正です

2.7.22 S14AD コールバック関数定義

S14AD のコールバック関数定義です。

表 2-69 adc_cd_event_t 型定義

型定義	内容
void (*adc_cd_event_t) (uint32_t event)	S14AD コールバック関数型定義

2.8 構造体定義

S14AD ドライバでは、ユーザが参照可能な構造体定義を r_adc_api.h ファイルで定義しています。

2.8.1 st_adc_pins_t 構造体

S14AD で使用するアナログ入力チャネル指定および温度センサ出力指定に使用する構造体です。

表 2-70st_adc_pins_t 構造体

要素名	型	内容
an_chans	uint32_t	アナログ入力チャネル
sensor	e_adc_msel_sensor_t	温度センサ有効/無効

2.8.2 st_adc_add_mode_t 構造体

AD_CMD_SET_ADD_MODE コマンドで、A/D 変換加算/平均モードを設定するときに第 2 引数の型として使用する構造体です。

表 2-71st_adc_add_mode_t 構造体

要素名	型	内容
add_mode	e_adc_add_t	A/D 変換加算/平均モード設定
chans	st_adc_pins_t	A/D 変換加算/平均の対象となる A/D 変換チャネル選択

2.8.3 st_adc_wina_t 構造体

AD_CMD_SET_WINDOWA コマンドで、コンペア機能ウィンドウ A を設定するときに第 2 引数の型として使用する構造体です。

表 2-72 st_adc_wina_t 構造体

要素名	型	内容
inten	uint8_t	ADC140_CMPAI 割り込みの許可/禁止
cmp_mode	e_adc_cmp_mode_t	コンペアモードを指定
level1	uint16_t	コンペアレベルを指定
level2	uint16_t	コンペアレベルを指定
chans	st_adc_pins_t	コンペア対象チャネルを組み合わせて指定
chan_cond	st_adc_pins_t	チャネルごとのコンペアモードを指定

2.8.4 st_adc_winb_t 構造体

AD_CMD_SET_WINDOWB コマンドで、コンペア機能ウィンドウ B を設定するときに第 2 引数の型として使用する構造体です。

表 2-73 st_adc_winb_t 構造体

要素名	型	内容
inten	uint8_t	ADC140_CMPBI 割り込みの許可/禁止
winb_cond	e_adc_winb_cond_t	コンペアモードを指定
comb	e_adc_comb_t	ウィンドウ A/B 複合条件を指定
level1	uint16_t	コンペアレベルを指定
level2	uint16_t	コンペアレベルを指定
channel	e_adc_ssel_ch_t	コンペア対象チャネルを指定

2.8.5 st_adc_wina_result_t 構造体

AD_CMD_GET_CMP_RESULT コマンドで、A/D コンペア機能ウィンドウ A の比較結果を取得するための構造体です。

表 2-74 st_adc_cmp_result_t 型引数格納情報

要素	型	内容
an_chans	uint32_t	A/D コンペア機能ウィンドウ A チャネルステータスレジスタ 0（ADCMPSR0）および A/D コンペア機能ウィンドウ A チャネルステータスレジスタ 1（ADCMPSR1）の値（ビット 6 は AN006、ビット 0 は AN000 に対応） [端子ごとの比較結果] 0：比較条件不成立 1：比較条件成立
sensor	uint8_t	A/D コンペア機能ウィンドウ B ステータスレジスタ（ADCMPBSR）の値 [ウィンドウ B 選択チャネル比較結果] 0：比較条件不成立 1：比較条件成立

2.8.6 st_adc_cmp_result_t 構造体

AD_CMD_GET_CMP_RESULT コマンドで、A/D コンペア機能比較結果を取得するための構造体です。

表 2-75 st_adc_cmp_result_t 型引数格納情報

要素	型	内容
wina_result	st_adc_wina_result_t	A/D コンペア機能ウィンドウ A の比較結果
winb_result	uint8_t	A/D コンペア機能ウィンドウ B ステータスレジスタ（ADCMPBSR）の値 [ウィンドウ B 選択チャネル比較結果] 0：比較条件不成立 1：比較条件成立
comb_result	uint8_t	A/D コンペア機能ウィンドウ A/B ステータスモニタレジスタ（ADWINMON） [組み合わせ結果(ビット 0)] 0：ウィンドウ A/ウィンドウ B の複合条件が不成立 1：ウィンドウ A/ウィンドウ B の複合条件が成立 [比較結果モニタ A(ビット 4)] 0：ウィンドウ A 比較条件が不成立 1：ウィンドウ A 比較条件が成立 [比較結果モニタ B(ビット 5)] 0：ウィンドウ B 比較条件が不成立 1：ウィンドウ B 比較条件が成立

2.8.7 st_adc_status_info_t 構造体

AD_CMD_GET_AD_STATE コマンドで、A/D コンバータの状態を取得するための構造体です。

表 2-76 st_adc_status_info_t 型引数格納情報

要素	型	内容
ad_info	e_adc_state_t	A/D スキャンの動作状態情報 ADC_STATE_STOP：A/D 変換停止中(ADCSR.ADST = 0) ADC_STATE_RUN：A/D 変換動作中(ADCSR.ADST = 1)
groupa_info	e_adc_conv_state_t	グループ A 動作状態情報 ADC_CONV_NOT_POLLING：ポーリング設定ではありません ADC_CONV_INCOMPLETE：グループ A の A/D スキャンが完了していません ADC_CONV_COMPLETE：グループ A の A/D スキャンが完了しています ADC_CONV_GET_FAILED：グループ A の状態取得に失敗しました
groupb_info	e_adc_conv_state_t	グループ B 動作状態情報 ADC_CONV_NOT_POLLING：ポーリング設定ではありません ADC_CONV_INCOMPLETE：グループ B の A/D スキャンが完了していません ADC_CONV_COMPLETE：グループ B の A/D スキャンが完了しています ADC_CONV_GET_FAILED：グループ B の状態取得に失敗しました
groupc_info	e_adc_conv_state_t	グループ C 動作状態情報 ADC_CONV_NOT_POLLING：ポーリング設定ではありません ADC_CONV_INCOMPLETE：グループ C の A/D スキャンが完了していません ADC_CONV_COMPLETE：グループ C の A/D スキャンが完了しています ADC_CONV_GET_FAILED：グループ C の状態取得に失敗しました
wcmpm_info	e_adc_conv_state_t	ウィンドウ A/B コンペアー致イベント発生情報 ADC_CONV_WCMP_DETECT：ウィンドウ A/B コンペアー致イベント発生 ADC_CONV_WCMP_NOT_DETECT : ウィンドウ A/B コンペアー致イベント未発生
wcmpum_info	e_adc_conv_state_t	ウィンドウ A/B コンペア不一致イベント発生情報 ADC_CONV_WCMP_DETECT：ウィンドウ A/B コンペア不一致イベント発生 ADC_CONV_WCMP_NOT_DETECT : ウィンドウ A/B コンペア不一致イベント未発生

2.8.8 st_adc_dma_read_info_t 構造体

AD_CMD_AUTO_READ_NORMAL コマンドで、DMA 転送（DMAC もしくは DTC）による A/D 変換結果自動読み取り機能（オートリード機能）の設定するときに第 2 引数の型として使用する構造体です。

表 2-77 st_adc_dma_read_info_t 型引数の設定値

要素	型	内容
cb_event	system_int_cb_t	DMA 転送完了タイミングで呼び出されるコールバック関数
dma_fact	e_adc_dma_event_t	DMA 起動要因を指定
src_addr	uint32_t	転送元アドレス
dest_addr	uint32_t	転送先アドレス
transfer_count	uint32_t	転送回数を指定
block_size	uint32_t	ブロックサイズを指定
reg_size	uint32_t	転送サイズを指定

2.9 状態遷移

S14AD ドライバの状態遷移図を図 2-60 に、各状態でのイベント動作を表 2-78 に示します。

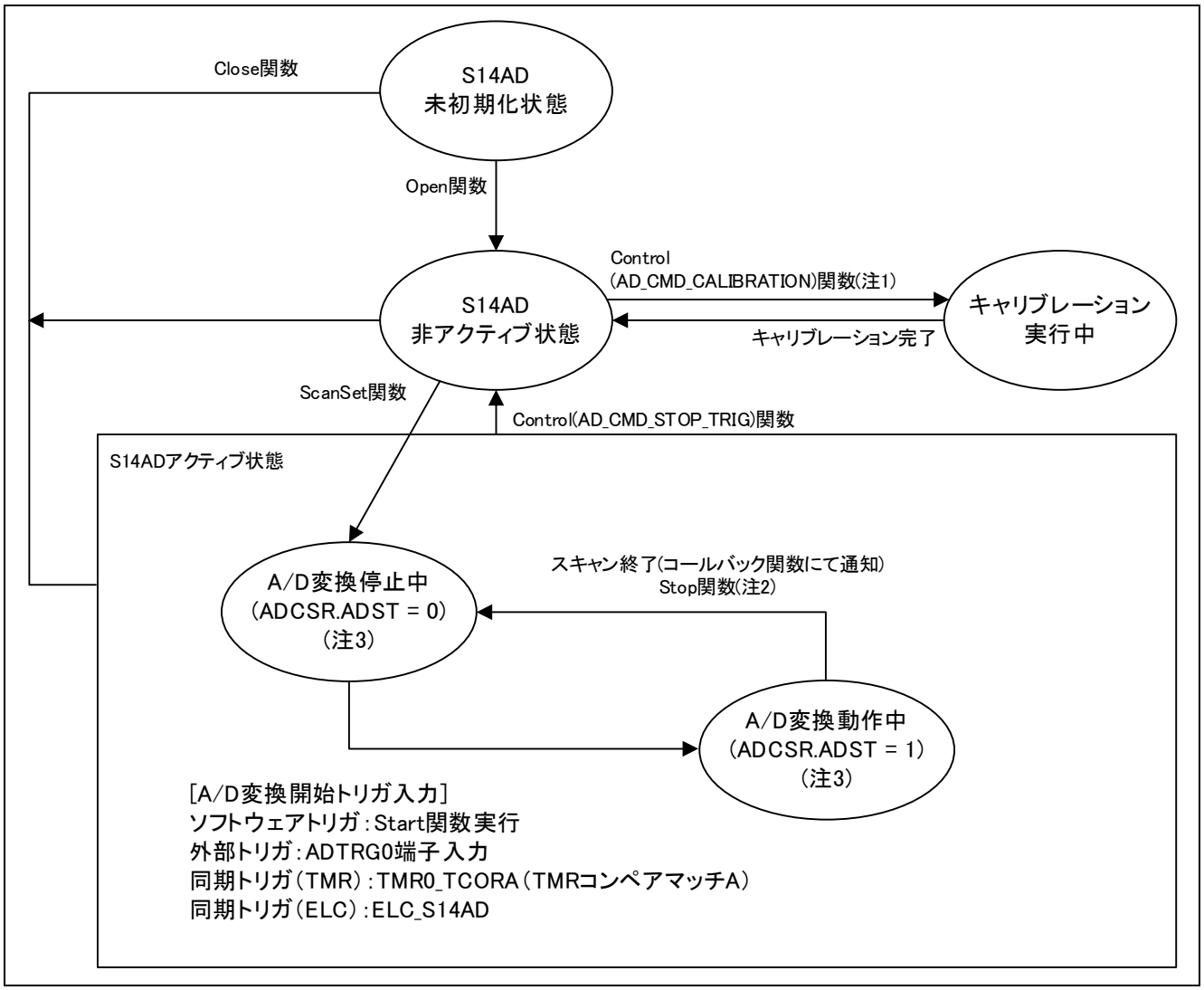


図 2-60 S14AD ドライバの状態遷移

注1. リセット解除後、オフセットキャリブレーションを実行してから S14AD アクティブ状態に遷移してください。

注2. Stop 関数による A/D 変換開始条件入力待ち状態への遷移は、ソフトウェアトリガ選択時のみ有効です。

注3. ADCSR.ADST は、以下のタイミングで”1”になります。指定したすべての A/D 変換が終了すると、ADCSR.ADST は自動的に”0”になります。

[ソフトウェアトリガ]

Start 関数実行時、Start 関数内で ADCSR.ADST = 1 を設定します

[同期トリガおよび外部トリガ]

指定したトリガ入力の検出で自動的に”1”になります

表 2-78 S14AD ドライバ状態でのイベント動作(注 1)

状態	概要	イベント	アクション
S14AD 未初期化状態	リセット解除後の S14AD ドライバの状態です	Open 関数の実行	S14AD 動作モード設定状態に遷移
S14AD 非アクティブ状態	S14AD モジュールにクロックが供給され、A/D 動作モードの設定がされている状態です	Close 関数の実行	S14AD 未初期化状態に遷移
		ScanSet 関数の実行	S14AD アクティブ状態（A/D 変換停止中）に遷移
		Control (AD_CMD_CALIBRATION)関数の実行	キャリブレーション実行中に遷移
キャリブレーション実行中	キャリブレーションを実行している状態です	キャリブレーション完了	S14AD 非アクティブ状態に遷移
S14AD アクティブ状態（A/D 変換停止中）	A/D 変換停止中です	Close 関数の実行	S14AD 未初期化状態に遷移
		[A/D 変換開始条件：ソフトウェアトリガ] Start 関数の実行	S14AD アクティブ状態（A/D 変換動作中）に遷移
		[A/D 変換開始条件：外部トリガ] ADTRG0 端子入力	S14AD アクティブ状態（A/D 変換動作中）に遷移
		[A/D 変換開始条件:8 ビットタイマトリガ] TMR0_TCORA(TMR コンペアマッチ A)	S14AD アクティブ状態（A/D 変換動作中）に遷移
		[A/D 変換開始条件：ELC トリガ] ELC_S14AD	S14AD アクティブ状態（A/D 変換動作中）に遷移
S14AD アクティブ状態（A/D 変換動作中）	A/D 変換実行中です	Close 関数の実行	S14AD 未初期化状態に遷移
		A/D 変換完了	S14AD アクティブ状態（A/D 変換停止中）に遷移し、コールバック関数を呼び出します(注 2)
		[A/D 変換開始条件：ソフトウェアトリガ] Stop 関数の実行	S14AD アクティブ状態（A/D 変換停止中）に遷移

注1. GetVersion、Read 関数はすべての状態で実行可能です。

注2. Open 関数実行時にコールバック関数を指定し、Control 関数で割り込み許可設定を行っていた場合のみコールバック関数を呼び出します。

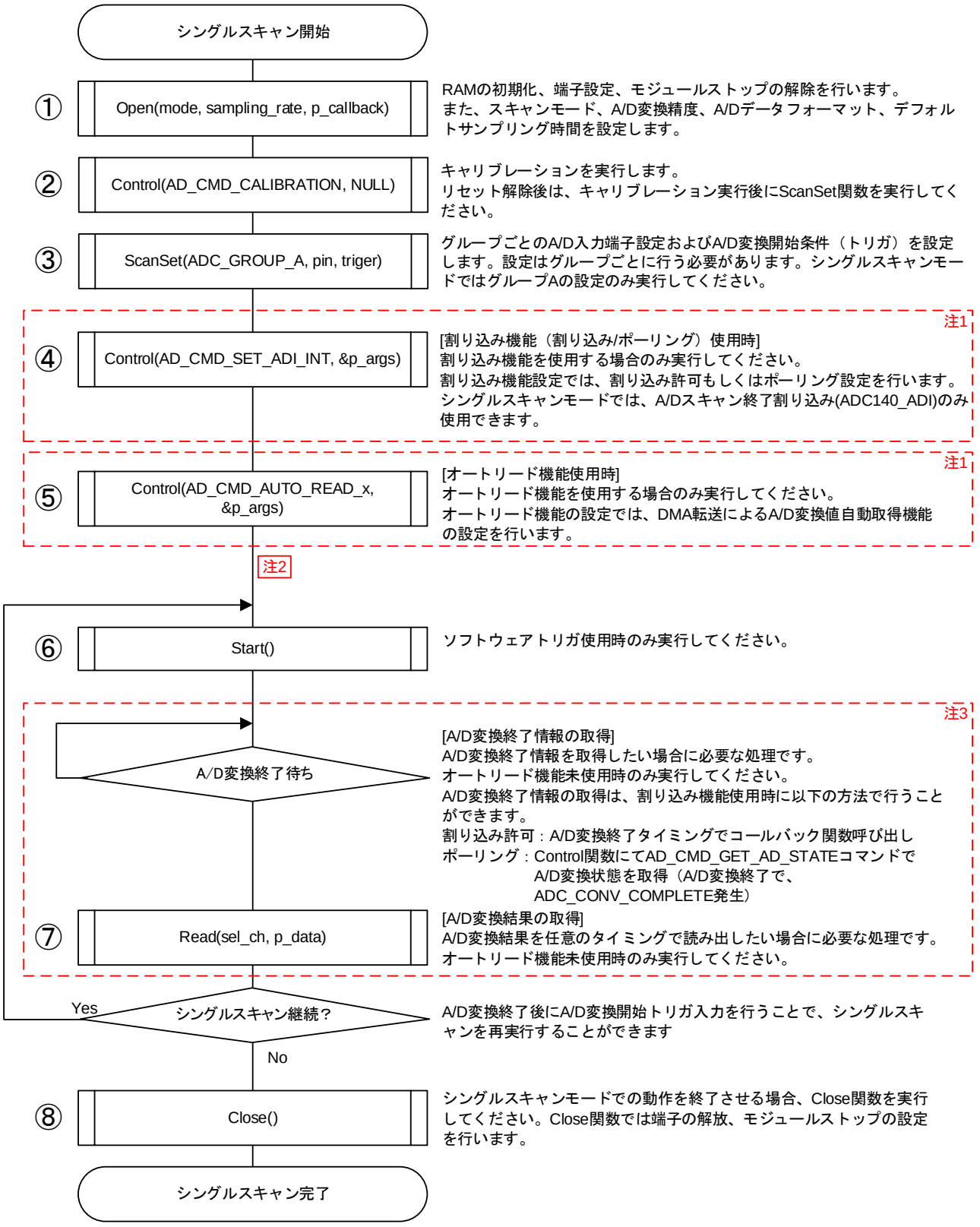
3. ドライバ動作説明

S14AD ドライバは、シングルスキャンモード、連続スキャンモード、グループスキャンモードによる A/D 変換機能を実現します。本章では、各スキャンモードおよび動作モードにおける実行手順と動作例を示します。

3.1 シングルスキャンモード

3.1.1 基本動作（グループ A/温度センサ出力）

シングルスキャンを実行すると、指定されたチャネルのアナログ入力を 1 サイクルのみ A/D 変換します。シングルスキャンの実行手順について図 3-1 に示します。



注1 割り込み機能とオートリード機能は排他的に制御されます。どちらか一方のみ使用することができます。
注2 処理⑤と処理⑥の間でTSCR.TSENビットの”１”設定および、
温度センサ基準電圧安定待ち処理をおこなってください。
注3 オートリード機能使用時は、自動的にA/D変換結果を取得するため本処理は不要です。

図 3-1 シングルスキャン実行手順

表 3-1 シングルスキャンモードの使用関数および引数一覧

関数	引数	内容	詳細
①Open 関数 関数詳細「2.4.1」	mode	uint32_t 型 スキャンモードにシングルスキャン（ADC_SINGLE_SCAN）を指定します 必要に応じて A/D 変換精度、A/D データフォーマットを組み合わせて指定してください	2.7.1
	sampling_rate	uint8_t 型 2～255 のデフォルトサンプリング時間(ADSSTRn(注)の初期値)を指定してください	-
	p_callback	adc_cd_event_t型 コールバック関数を指定してください NULLを指定するとS14ADのイベント発生時にコールバック関数は呼び出されません	2.7.22
②Control 関数 AD_CMD_CALIBRATION 機能詳細「2.5.21」	cmd	AD_CMD_CALIBRATION を指定してください	-
	p_args	NULL を指定してください	-
③ScanSet 関数 関数詳細「2.4.2」	group	e_adc_group_t 型 端子設定を行うグループを指定します シングルスキャンモードでは、グループ A 設定定義の ADC_GROUP_A のみ使用できます	2.7.6
	pins	st_adc_pins_t 型 A/D 変換入力端子を指定します	2.8.1
	triger	e_adc_triger_t 型 A/D 変換開始トリガを指定します	2.7.2
④Control 関数 AD_CMD_SET_ADI_INT 機能詳細「2.5.4」	cmd	AD_CMD_SET_ADI_INT を指定してください	-
	p_args	e_adc_int_method_t 型変数のポインタ 割り込み設定（許可ポーリング）を指定します 割り込み許可：ADC_INT_ENABLE ポーリング：ADC_INT_POLLING	2.7.10
⑤Control 関数 AD_CMD_AUTO_READ_x (x = NORMAL、BLOCK) 機能詳細(NORMAL)「2.5.23」 機能詳細(BLOCK)「2.5.24」	cmd	AD_CMD_AUTO_READ_x (x = NORMAL、BLOCK) を指定してください	-
	p_args	st_adc_dma_read_info_t 型変数のポインタ st_adc_dma_read_info_t 型構造体変数の各要素に、DMA 転送起動要因、転送元レジスタ、転送先 RAM、転送回数、コールバック関数、転送サイズを指定してください	2.8.8
⑥Start 関数	-	-	-
⑦Read 関数	sel_ch	e_adc_ssel_ch_t 型 A/D 変換結果を読み出すアナログ入力チャネルを一つ指定してください 温度センサ出力の A/D 変換結果を読み出す場合は、ADC_SSEL_TEMP を指定してください	2.7.5
	p_data	uint16_t 型変数のポインタ A/D 変換結果格納先アドレスを指定してください	-

注　256KB グループ：n = 0～7、L、T　　1500KB グループ：n = 0～6、L、T

AN000～AN002 までをシングルスキャンモードで A/D 変換する場合の動作例を図 3-2 に示します。

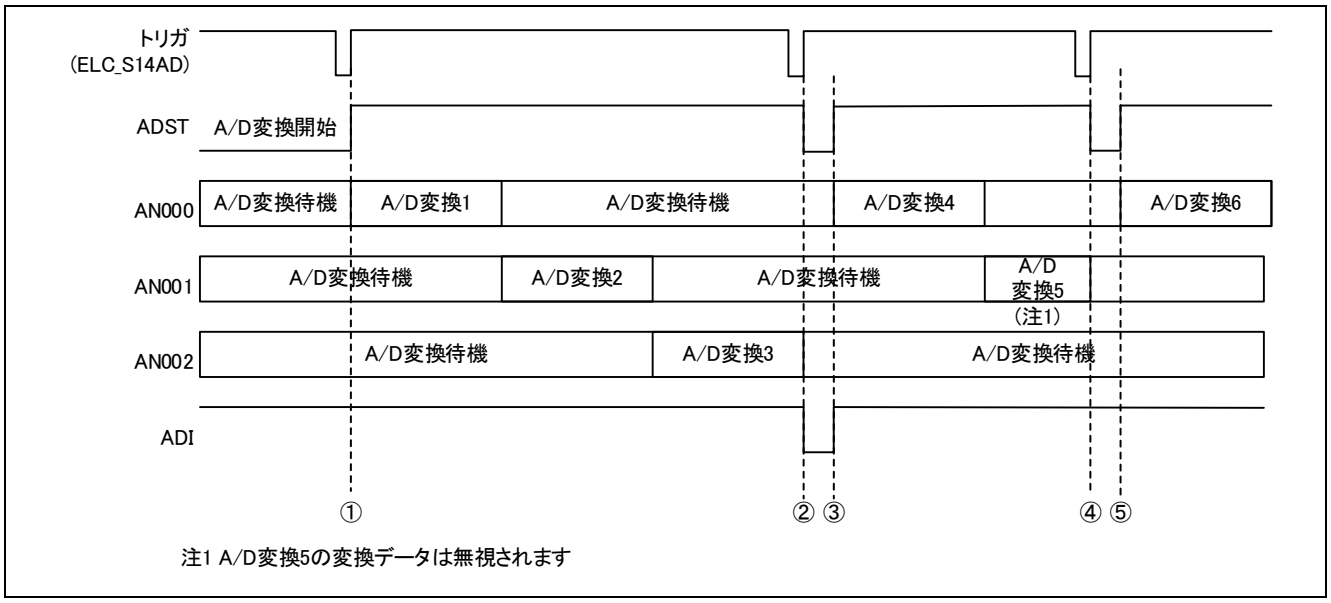


図 3-2 シングルスキャン動作

① A/D 変換開始トリガが入力されると ADST = 1 となり A/D 変換が開始されます。A/D 変換開始トリガ入力は、A/D 変換開始トリガ設定によって異なります。各設定における A/D 変換開始トリガ入力を以下に示します。

ソフトウェアトリガ：Start 関数実行
外部トリガ：ADTRG0 端子入力
同期トリガ（TMR）：TMR0_TCORA（TMR コンペアマッチ A）イベント検出
同期トリガ（ELC）：ELC_S14AD イベント検出

② A/D 変換が終了すると、ADST = 0 となります。この時、ADI 割り込みが許可状態(ADCSR.ADIE = 1）の場合 ADC140_ADI 割り込み要求が発生します。

A/D 変換終了タイミングの確認方法は、割り込み機能およびオートリード機能の設定によって異なります。各機能における A/D 変換終了タイミングの確認方法を以下に示します。

[割り込み]

Open 関数実行時にコールバック関数を指定し、割り込み機能設定にて割り込みを許可状態にしていた場合、A/D 変換終了タイミングでコールバック関数が実行されます。

[ポーリング]

割り込み機能設定にてポーリング設定を行っていた場合、Control 関数の AD_CMD_GET_AD_STATE コマンドを実行することで各 A/D 変換グループの動作状態を取得できます。A/D 変換が完了したグループの動作状態情報には、ADC_CONV_COMPLETE が格納されます。AD_CMD_GET_AD_STATE コマンドの使用例および取得できる情報の詳細は、「2.5.17 A/D コンバータの状態取得機能 (AD_CMD_GET_AD_STATE)」を参照してください。

[オートリード]

オートリード機能を使用している場合、A/D 変換終了タイミングで自動的に A/D 変換結果が RAM へ転送されます。A/D 変換終了ごとの通知はありません。オートリード機能設定時にコールバック関数を指定していた場合、オートリードが指定回数完了したタイミングでコールバック関数が実行されます。

[割り込みおよびオートリード機能未使用]

割り込みおよびオートリード機能を使用していない場合、A/D 変換動作状態(ADST = 1）から A/D 変換停止状態(ADST = 0）への遷移を検出することで A/D 変換終了を確認できます。A/D 変換動作状態は、Control 関数の AD_CMD_GET_AD_STATE コマンドで取得できます。

③ A/D 変換開始トリガ入力により、A/D 変換を再度実行します。

④ A/D 変換終了前に A/D 変換を停止させた場合､A/D 変換結果は不定値となります｡A/D 変換停止手順は、A/D 変換開始トリガ設定によって異なります。A/D 変換開始トリガ設定ごとの A/D 変換停止手順を以下に示します。

[ソフトウェアトリガ]
Stop 関数実行

[外部トリガ、同期トリガ(TMR、ELC)]
Control 関数で AD_CMD_STOP_TRIG コマンドを実行

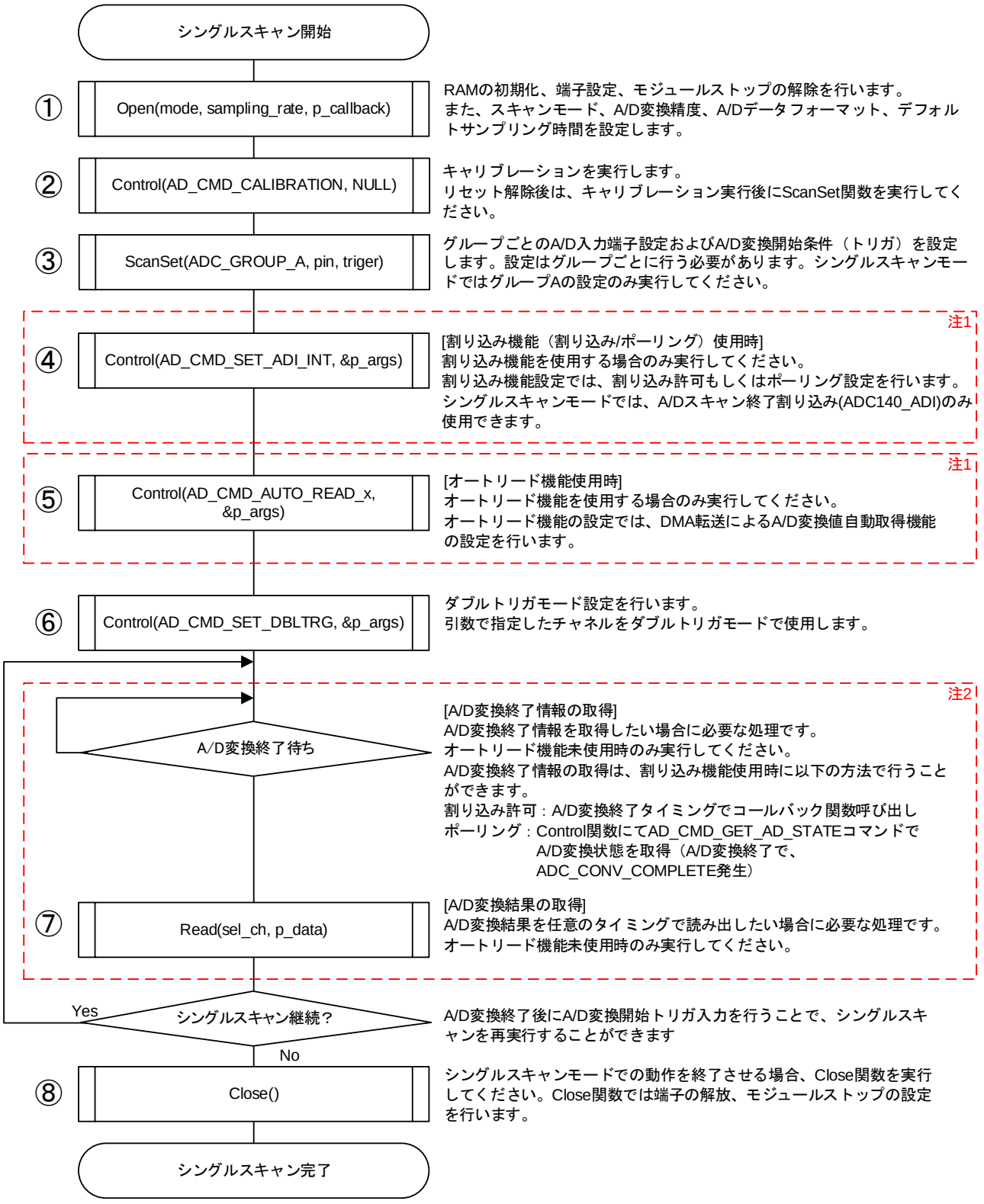
⑤ A/D 変換停止後 A/D 変換を再開させた場合、AN000 から A/D 変換を開始します。A/D 変換停止後の再開手順は、A/D 変換開始トリガ設定によって異なります。A/D 変換開始トリガ設定ごとの A/D 変換再開手順を以下に示します。

[ソフトウェアトリガ]
Start 関数実行

[外部トリガ、同期トリガ(TMR、ELC)]
ScanSet 関数にて A/D 変換開始トリガを再設定し、A/D 変換開始トリガ設定に応じたトリガ入力で A/D 変換を再開

3.1.2 ダブルトリガモード

シングルスキャンモードでダブルトリガモードを選択した場合は、同期トリガ（TMR、ELC）で開始するシングルスキャンモードの実行 2 回分を一連の動作として実行します。シングルスキャン(ダブルトリガモード）の実行手順を図 3-3 に示します。



注1 割り込み機能とオートリード機能は排他的に制御されます。どちらか一方のみ使用することができます。
注2 オートリード機能使用時は、自動的にA/D変換結果を取得するため本処理は不要です。

図 3-3 シングルスキャン（ダブルトリガモード）実行手順

シングルスキャン（ダブルトリガモード）設定で使用する関数および引数の一覧を表 3-2 に示します。

表 3-2 シングルスキャン（ダブルトリガモード）の使用関数および引数一覧

関数	引数	内容	詳細
①Open 関数 関数詳細「2.4.1」	mode	uint32_t型 スキャンモードにシングルスキャン（ADC_SINGLE_SCAN）を指定します 必要に応じてA/D変換精度、A/Dデータフォーマットを組み合わせて指定してください	2.7.1
	sampling_rate	uint8_t型 2～255のデフォルトサンプリング時間(ADSSTRn(注)の初期値)を指定してください	-
	p_callback	adc_cd_event_t型 コールバック関数を指定してください NULLを指定するとS14ADのイベント発生時にコールバック関数は呼び出されません	2.7.22
②Control 関数 AD_CMD_CALIBRATION 機能詳細「2.5.21」	cmd	AD_CMD_CALIBRATIONを指定してください	-
	p_args	NULLを指定してください	-
③ScanSet 関数 関数詳細「2.4.2」	group	e_adc_group_t型 端子設定を行うグループを指定します シングルスキャンモードでは、グループA設定定義のADC_GROUP_Aのみ使用できます	2.7.6
	pins	st_adc_pins_t型 A/D変換入力端子を指定します	2.8.1
	triger	e_adc_triger_t型 A/D変換開始トリガを指定します ダブルトリガモードでは同期トリガもしくは非同期トリガを使用してください ソフトウェアトリガは使用できません	2.7.2
④Control 関数 AD_CMD_SET_ADI_INT 機能詳細「2.5.4」	cmd	AD_CMD_SET_ADI_INTを指定してください	-
	p_args	e_adc_int_method_t型変数のポインタ 割り込み設定（許可ポーリング）を指定します 割り込み許可：ADC_INT_ENABLE ポーリング：ADC_INT_POLLING	2.7.10
⑤Control 関数 AD_CMD_AUTO_READ_x (x = NORMAL、BLOCK) 機能詳細(NORMAL)「2.5.23」 機能詳細(BLOCK)「2.5.24」	cmd	AD_CMD_AUTO_READ_x (x = NORMAL、BLOCK) を指定してください	-
	p_args	st_adc_dma_read_info_t型変数のポインタ st_adc_dma_read_info_t型構造体変数の各要素に、DMA転送起動要因、転送元レジスタ、転送先RAM、転送回数、コールバック関数、転送サイズを指定してください	2.8.8
⑥Control 関数 AD_CMD_SET_DBLTRG 機能詳細「2.5.2」	cmd	AD_CMD_SET_DBLTRGを指定してください	-
	p_args	int8_t型 ダブルトリガモードで使用するチャネルを指定してください ダブルトリガモードを無効にする場合は、”-1”を指定してください	-
⑦Read 関数	sel_ch	e_adc_ssel_ch_t型 A/D変換結果を読み出すアナログ入力チャネルを一つ指定してください 2回目のトリガによってA/D変換した結果を読み出す場合は、ADC_SSEL_DBLを指定してください	2.7.5
	p_data	uint16_t型変数のポインタ A/D変換結果格納先アドレスを指定してください	-
⑧Close 関数	-	-	-

注　256KB グループ：n = 0～7、L、T　　1500KB グループ：n = 0～6、L、T

AN000～AN002 までをシングルスキャンモードで A/D 変換する場合の動作例を図 3-4 に示します。

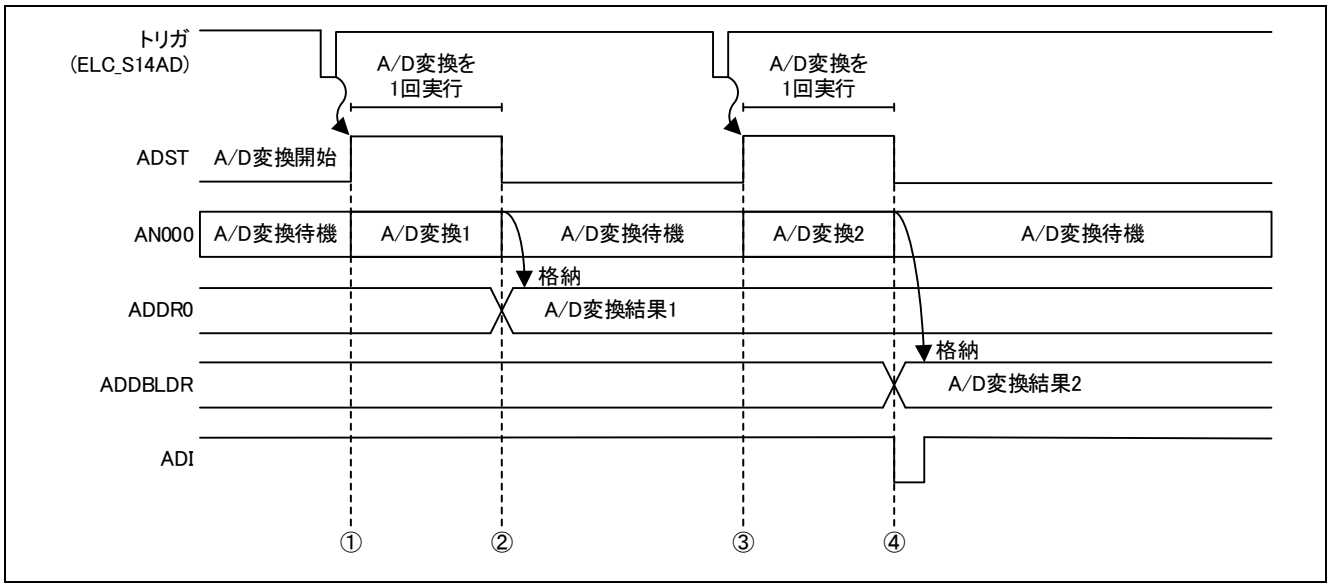


図 3-4 シングルスキャン（ダブルトリガモード）動作

① A/D 変換開始トリガが入力されると ADST = 1 となり A/D 変換が開始されます。A/D 変換開始トリガ入力は、A/D 変換開始トリガ設定によって異なります。各設定における A/D 変換開始トリガ入力を以下に示します。
ソフトウェアトリガ：Start 関数実行
外部トリガ：ADTRG0 端子入力
同期トリガ（TMR）：TMR0_TCORA（TMR コンペアマッチ A）イベント検出
同期トリガ（ELC）：ELC_S14AD イベント検出

② A/D 変換が終了すると、ADST = 0 となります。AN000 の A/D 変換結果が ADDR0 に格納されます。このタイミングでは、S140_ADI 割り込み要求は発生しません。

③ 2 回目の A/D 変換開始トリガ入力で ADST = 1 となり A/D 変換が開始されます。

④ A/D 変換が終了すると、ADST = 0 となります。AN000 の A/D 変換結果が ADDBLDR に格納されます。ADI 割り込みが許可状態(ADCSR.ADIE = 1）の場合、ADC140_ADI 割り込み要求が発生します。A/D 変換終了タイミングの確認方法は、割り込み機能およびオートリード機能の設定によって異なります。各機能における A/D 変換終了タイミングの確認方法を以下に示します。

[割り込み]

Open 関数実行時にコールバック関数を指定し、割り込み機能設定にて割り込みを許可状態にしていた場合、2 回目の A/D 変換終了タイミングでコールバック関数が実行されます。

[ポーリング]

割り込み機能設定にてポーリング設定を行っていた場合、Control 関数の AD_CMD_GET_AD_STATE コマンドを実行することで各 A/D 変換グループの動作状態を取得できます。ダブルトリガモードでは、2 回目の A/D 変換が終了している場合、グループ A の A/D 変換動作状態情報に ADC_CONV_COMPLETE が格納されます。AD_CMD_GET_AD_STATE コマンドの使用例および取得できる情報の詳細は、「2.5.17 A/D コンバータの状態取得機能(AD_CMD_GET_AD_STATE)」を参照してください。

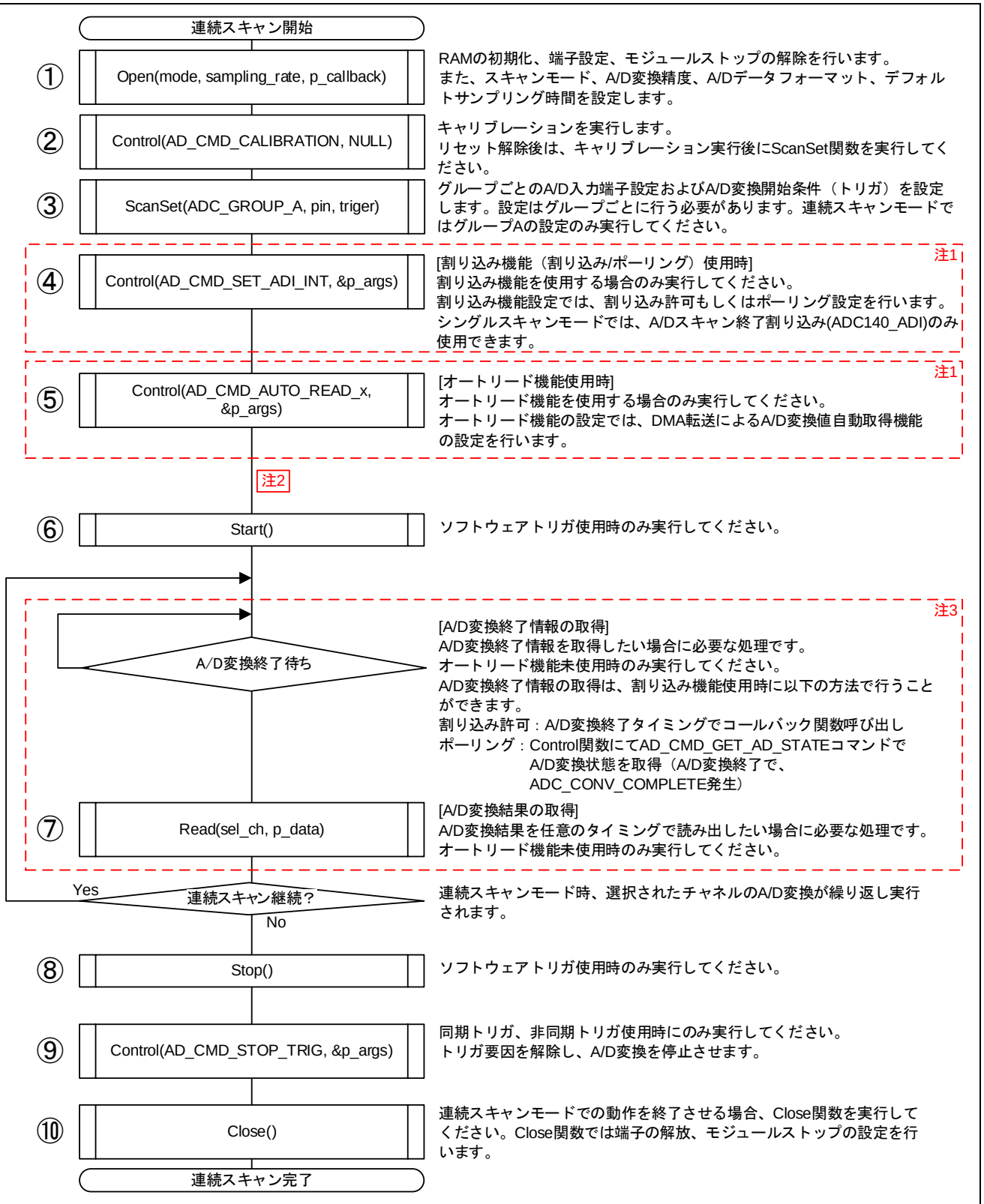
[オートリード]

オートリード機能を使用している場合、2 回目の A/D 変換終了タイミングで自動的に A/D 変換結果が RAM へ転送されます。A/D 変換終了ごとの通知はありません。オートリード機能設定時にコールバック関数を指定していた場合、オートリードが指定回数完了したタイミングでコールバック関数が実行されます。

3.2 連続スキャンモード

3.2.1 基本動作（グループ A、温度センサ出力）

連続スキャンモードでは、選択されたチャネルのアナログ入力を繰り返し A/D 変換します。



注1 割り込み機能とオートリード機能は排他的に制御されます。どちらか一方のみ使用することができます。
注2 処理⑤と処理⑥の間でTSCR.TSENビットの”1”設定および、
温度センサ基準電圧安定待ち処理をおこなってください。
注3 オートリード機能使用時は、自動的にA/D変換結果を取得するため本処理は不要です。

図 3-5 連続スキャン実行手順

連続スキャンモード設定で使用する関数および引数の一覧を表 3-3 に示します。

表 3-3 連続スキャンモードの使用関数および引数一覧

関数	引数	内容	詳細
①Open 関数 関数詳細「2.4.1」	mode	uint32_t 型 スキャンモードに連続スキャン（ADC_REPEAT_SCAN）を指定します 必要に応じて A/D 変換精度、A/D データフォーマットを組み合わせて指定してください	2.7.1
	sampling_rate	uint8_t 型 2～255 のデフォルトサンプリング時間(ADSSTRn(注)の初期値)を指定してください	-
	p_callback	adc_cd_event_t型 コールバック関数を指定してください NULLを指定するとS14ADのイベント発生時にコールバック関数は呼び出されません	2.7.22
②Control 関数 AD_CMD_CALIBRATION 機能詳細「2.5.21」	cmd	AD_CMD_CALIBRATION を指定してください	-
	p_args	NULL を指定してください	-
③ScanSet 関数 関数詳細「2.4.2」	group	e_adc_group_t 型 端子設定を行うグループを指定します 連続スキャンモードでは、グループ A 設定定義の ADC_GROUP_A のみ使用できます	2.7.6
	pins	st_adc_pins_t 型 A/D 変換入力端子を指定します	2.8.1
	triger	e_adc_triger_t 型 A/D 変換開始トリガを指定します	2.7.2
④Control 関数 AD_CMD_SET_ADI_INT 機能詳細「2.5.4」	cmd	AD_CMD_SET_ADI_INT を指定してください	-
	p_args	e_adc_int_method_t 型変数のポインタ 割り込み設定（許可ポーリング）を指定します 割り込み許可：ADC_INT_ENABLE ポーリング：ADC_INT_POLLING	2.7.10
⑤Control 関数 AD_CMD_AUTO_READ_x (x = NORMAL、BLOCK) 機能詳細(NORMAL)「2.5.23」 機能詳細(BLOCK)「2.5.24」	cmd	AD_CMD_AUTO_READ_x (x = NORMAL、BLOCK) を指定してください	-
	p_args	st_adc_dma_read_info_t 型変数のポインタ st_adc_dma_read_info_t 型構造体変数の各要素に、DMA 転送起動要因、転送元レジスタ、転送先 RAM、転送回数、コールバック関数、転送サイズを指定してください	2.8.8
⑥Start 関数	-	-	-
⑦Read 関数	sel_ch	e_adc_ssel_ch_t 型 A/D 変換結果を読み出すアナログ入力チャネルを一つ指定してください 温度センサ出力の A/D 変換結果を読み出す場合は、ADC_SSEL_TEMP を指定してください	2.7.5
	p_data	uint16_t 型変数のポインタ A/D 変換結果格納先アドレスを指定してください	-
⑧Stop 関数	-	-	-
⑨Control 関数 AD_CMD_STOP_TRIG 機能詳細「2.5.22」	cmd	AD_CMD_STOP_TRIG を指定してください	-
	p_args	NULL を指定してください	-
⑩Close 関数	-	-	-

注　256KB グループ：n = 0～7、L、T　　1500KB グループ：n = 0～6、L、T

AN000～AN002 までを連続スキャンモードで A/D 変換する場合の動作例を図 3-2 に示します。

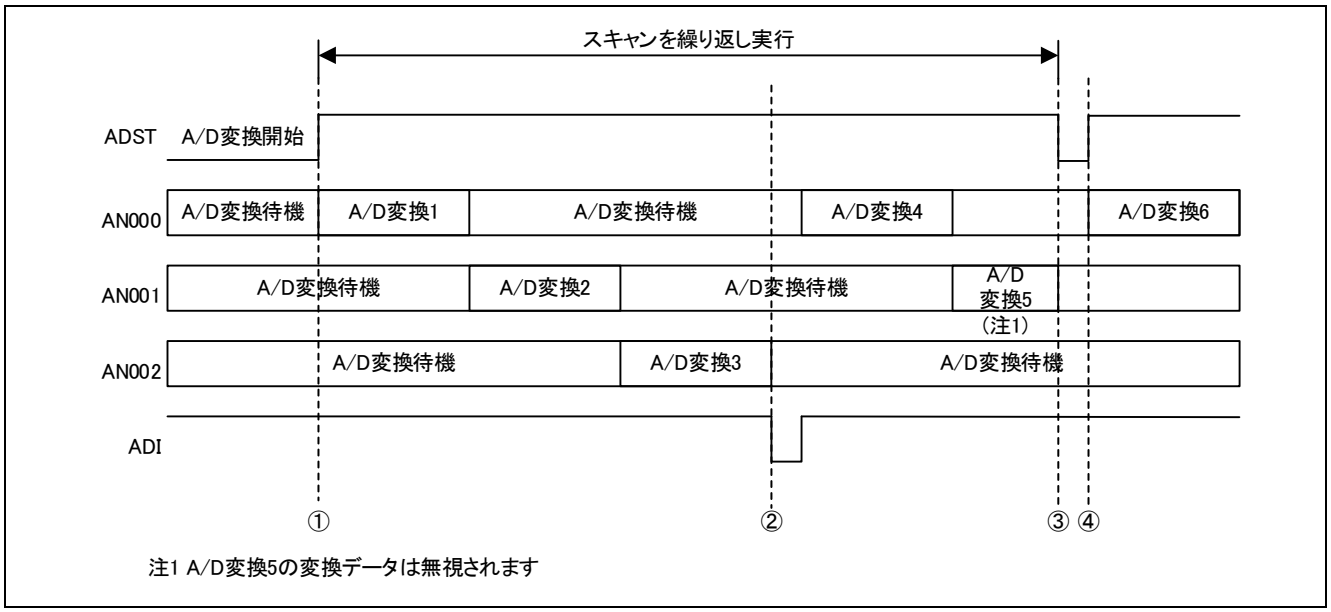


図 3-6 連続スキャン動作

① A/D 変換開始トリガが入力されると ADST = 1 となり A/D 変換が開始されます。A/D 変換開始トリガ入力は、A/D 変換開始トリガ設定によって異なります。各設定における A/D 変換開始トリガ入力を以下に示します。
ソフトウェアトリガ：Start 関数実行
外部トリガ：ADTRG0 端子入力
同期トリガ（TMR）：TMR0_TCORA（TMR コンペアマッチ A）イベント検出
同期トリガ（ELC）：ELC_S14AD イベント検出

② 選択したすべてのチャネルの A/D 変換終了後、ADI 割り込みが許可状態(ADCSR.ADIE = 1）の場合、ADI 割り込み要求が発生します。A/D 変換終了タイミングの確認方法は、割り込み機能およびオートリード機能の設定によって異なります。各機能における A/D 変換終了タイミングの確認方法を以下に示します。
[割り込み]
Open 関数実行時にコールバック関数を指定し、割り込み機能設定にて割り込みを許可状態にしていた場合、A/D 変換終了タイミングでコールバック関数が実行されます。
[ポーリング]
割り込み機能設定にてポーリング設定を行っていた場合、Control 関数の AD_CMD_GET_AD_STATE コマンドを実行することで各 A/D 変換グループの動作状態を取得できます。A/D 変換が完了したグループの動作状態情報には、ADC_CONV_COMPLETE が格納されます。AD_CMD_GET_AD_STATE コマンドの使用例および取得できる情報の詳細は、「2.5.17 A/D コンバータの状態取得機能 (AD_CMD_GET_AD_STATE)」を参照してください。
[オートリード]
オートリード機能を使用している場合、A/D 変換終了タイミングで自動的に A/D 変換結果が RAM へ転送されます。A/D 変換終了ごとの通知はありません。オートリード機能設定時にコールバック関数を指定していた場合、オートリードが指定回数完了したタイミングでコールバック関数が実行されます。

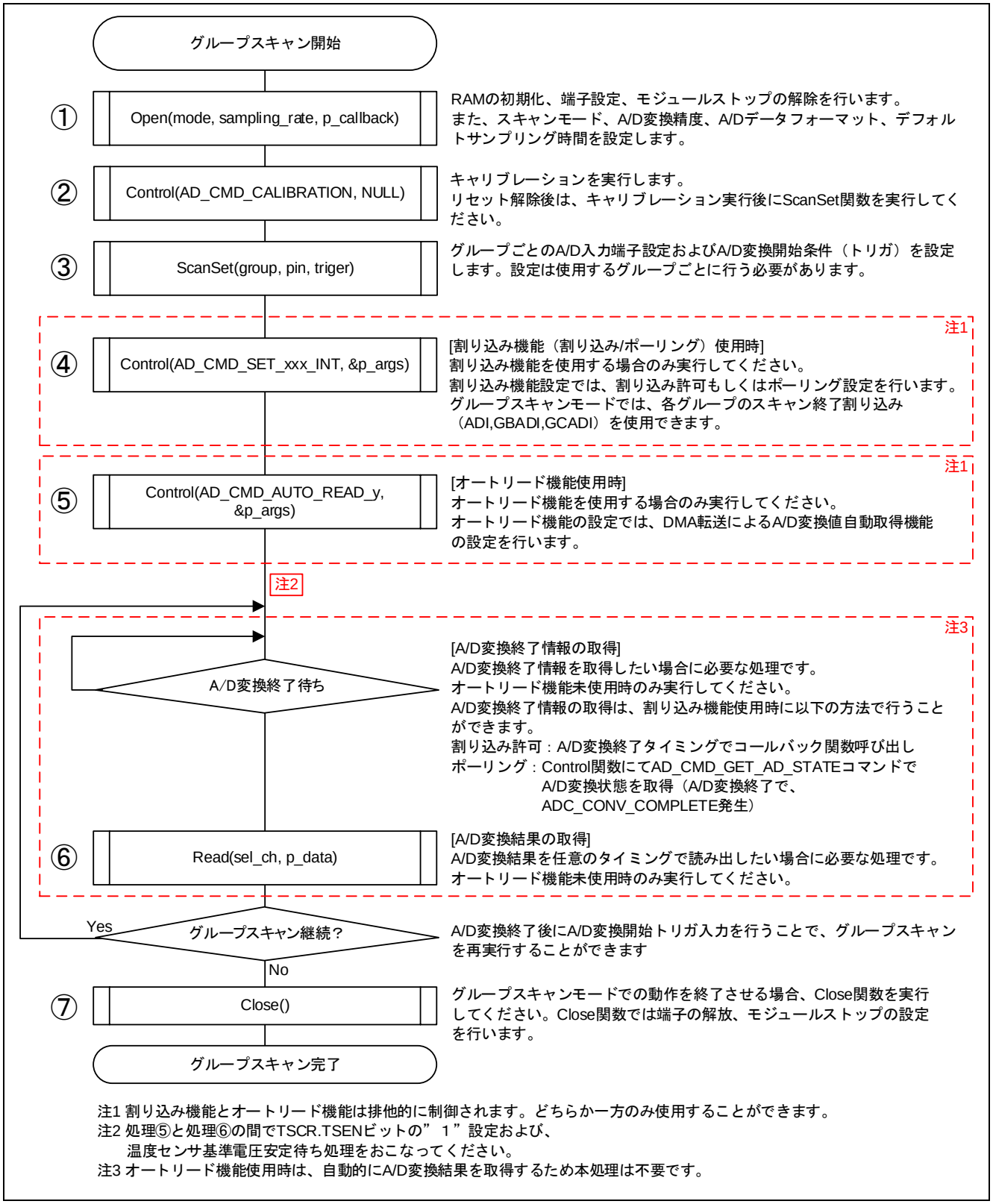
③ ADST = 0 になると、A/D 変換が停止します。A/D 変換を停止させる方法は、A/D 変換開始トリガ設定によって異なります。各 A/D 変換開始トリガ設定における A/D 変換停止方法を以下に示します。
[ソフトウェアトリガ]
Stop 関数実行
[同期トリガ/非同期トリガ]
Control 関数で AD_CMD_STOP_TRIG コマンドを実行

④ ソフトウェアトリガ設定時、Start 関数の実行で A/D 変換が再開されます。
同期トリガおよび非同期トリガ設定の場合、ScanSet 関数で再度 A/D 変換開始トリガを設定する必要があります。A/D 変換開始トリガの再設定後、A/D 変換開始トリガ入力により A/D 変換を開始します。

3.3 グループスキャンモード

3.3.1 基本動作（グループ A/グループ B/グループ C/温度センサ出力）

グループスキャンモードで使用するグループの数は 2 つ（グループ A、B）と 3 つ（グループ A、B、C）のどちらか一方を選択することができます。グループスキャンモードの基本動作は、同期トリガをスキャン開始条件とし、グループ A、B またはグループ A、B、C のそれぞれで選択したすべてのチャネルのアナログ入力を 1 回のみ A/D 変換します。グループ A、B、C のそれぞれのスキャン動作は、シングルスキャンモードと同じ動作になります。



注1 割り込み機能とオートリード機能は排他的に制御されます。どちらか一方のみ使用することができます。
注2 処理⑤と処理⑥の間でTSCR.TSENビットの”1”設定および、
温度センサ基準電圧安定待ち処理をおこなってください。
注3 オートリード機能使用時は、自動的にA/D変換結果を取得するため本処理は不要です。

図 3-7 グループスキャン実行手順

グループスキャンモード設定で使用する関数および引数の一覧を表 3-4 に示します。

表 3-4 グループスキャンモードの使用関数および引数一覧

関数	引数	内容	詳細
①Open 関数 関数詳細「2.4.1」	mode	uint32_t 型 スキャンモードにグループスキャン（ADC_GROUP_SCAN）を指定します 必要に応じて A/D 変換精度、A/D データフォーマットを組み合わせて指定してください	2.7.1
	sampling_rate	uint8_t 型 2～255 のデフォルトサンプリング時間(ADSSTRn(注)の初期値)を指定してください	-
	p_callback	adc_cd_event_t 型 コールバック関数を指定してください NULL を指定すると S14AD のイベント発生時にコールバック関数は呼び出されません	2.7.22
②Control 関数 AD_CMD_CALIBRATION 機能詳細「2.5.21」	cmd	AD_CMD_CALIBRATION を指定してください	-
	p_args	NULL を指定してください	-
③ScanSet 関数 関数詳細「2.4.2」	group	e_adc_group_t 型 端子設定を行うグループを指定します 2 つのグループを使用する場合は、グループ A およびグループ B の端子設定をそれぞれ実行してください 3 つのグループを使用する場合は、グループ A、グループ B、グループ C の端子設定をそれぞれ実行してください	2.7.6
	pins	st_adc_pins_t 型 A/D 変換入力端子を指定します	2.8.1
	triger	e_adc_triger_t 型 A/D 変換開始トリガを指定します グループスキャンモードでは、ソフトウェアトリガは使用できません	2.7.2
④Control 関数 AD_CMD_SET_xxx_INT (xxx = ADI、GROUPB、GROUPC) 機能詳細(ADI)「2.5.4」 機能詳細(GROUPB)「2.5.5」 機能詳細(GROUPC)「2.5.6」	cmd	e_adc_cmd_t 型 使用する割り込み要求ごとにコマンドを実行してください	-
	p_args	e_adc_int_method_t 型変数のポインタ 割り込み設定（許可ポーリング）を指定します 割り込み許可：ADC_INT_ENABLE ポーリング：ADC_INT_POLLING	2.7.10
⑤Control 関数 AD_CMD_AUTO_READ_y (y = NORMAL、BLOCK) 機能詳細(NORMAL)「2.5.23」 機能詳細(BLOCK)「2.5.24」	cmd	AD_CMD_AUTO_READ_y (y = NORMAL、BLOCK) を指定してください	-
	p_args	st_adc_dma_read_info_t 型変数のポインタ st_adc_dma_read_info_t 型構造体変数の各要素に、DMA 転送起動要因、転送元レジスタ、転送先 RAM、転送回数、コールバック関数、転送サイズを指定してください	2.8.8
⑥Read 関数	sel_ch	e_adc_ssel_ch_t 型 A/D 変換結果を読み出すアナログ入力チャネルを一つ指定してください 温度センサ出力の A/D 変換結果を読み出す場合は、ADC_SSEL_TEMP を指定してください	2.7.5
	p_data	uint16_t 型変数のポインタ A/D 変換結果格納先アドレスを指定してください	-
⑦Close 関数	-	-	-

注　256KB グループ：n = 0～7、L、T　　1500KB グループ：n = 0～6、L、T

グループスキャンモード（グループ A、B、C 使用）で A/D 変換する場合の動作例を図 3-8 に示します。動作例における各グループの A/D 変換開始トリガは以下のとおりです。

グループ A：非同期トリガ（ADTRG0 端子入力）
グループ B：同期トリガ（ELC_S14AD イベント）
グループ C：同期トリガ（TMR_TCORA（TMR コンペアマッチ A））

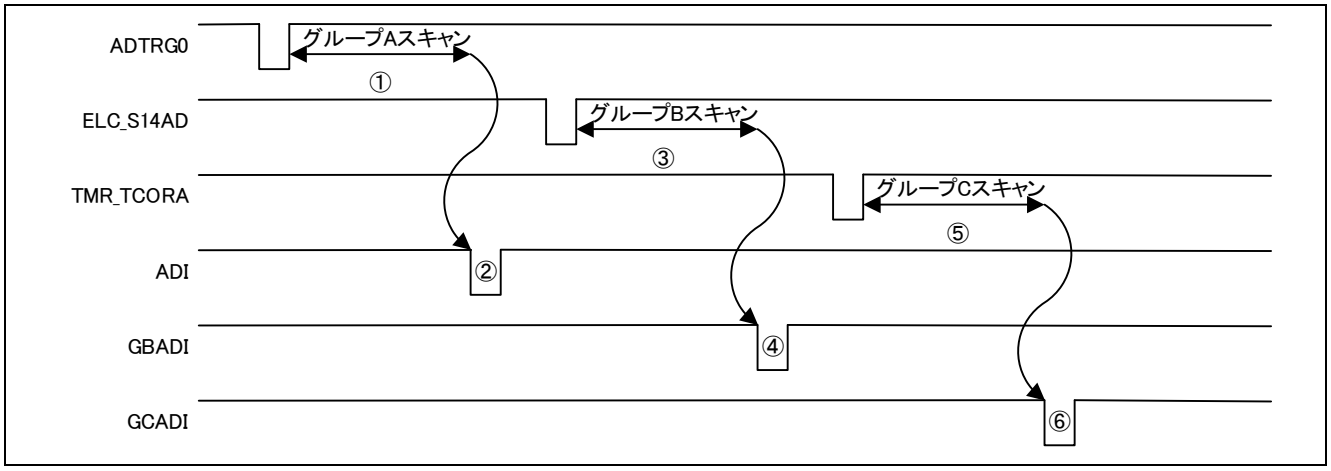


図 3-8 グループスキャン動作

① ADTRG0 入力を検出すると、グループ A の A/D 変換が実行されます。

② グループ A の A/D 変換終了後、ADI 割り込みが許可状態(ADCSR.ADIE = 1）の場合、ADC140_ADI 割り込み要求が発生します。A/D 変換終了タイミングを確認する方法は、割り込み機能およびオートリード機能の設定によって異なります。各機能における A/D 変換終了タイミングの確認を以下に示します。

[割り込み]

Open 関数実行時にコールバック関数を指定し、割り込み機能設定にて割り込みを許可状態にしていた場合、A/D 変換終了タイミングでコールバック関数が実行されます。

[ポーリング]

割り込み機能設定にてポーリング設定を行っていた場合、Control 関数の AD_CMD_GET_AD_STATE コマンドを実行することで各 A/D 変換グループの動作状態を取得できます。A/D 変換が完了したグループの動作状態情報には、ADC_CONV_COMPLETE が格納されます。AD_CMD_GET_AD_STATE コマンドの使用例および取得できる情報の詳細は、「2.5.17 A/D コンバータの状態取得機能 (AD_CMD_GET_AD_STATE)」を参照してください。

[オートリード]

オートリード機能を使用している場合、DMA 起動要因に指定したグループの A/D 変換終了タイミングで自動的に A/D 変換結果が RAM へ転送されます。A/D 変換終了ごとの通知はありません。オートリード機能設定時にコールバック関数を指定していた場合、オートリードが指定回数完了したタイミングでコールバック関数が実行されます。

③ ELC_S14AD イベントを検出すると、グループ B の A/D 変換が実行されます。

④ グループ B の A/D 変換終了後、GBADI 割り込みが許可状態(ADCSR.GBADIE = 1)の場合 ADC140_GBADI 割り込み要求が発生します。グループ B の A/D 変換終了タイミングを確認する方法は、割り込み機能およびオートリード機能の設定によって異なります。各機能における A/D 変換終了タイミングの確認方法は②と同様です。

⑤ TMR_TCORA イベントを検出すると、グループ C の A/D 変換が実行されます。

⑥ グループ C の A/D 変換終了後、GCADI 割り込みが許可状態(ADGCTRGR.GCADIE = 1)の場合 ADC140_GCADI 割り込み要求が発生します。グループ C の A/D 変換終了タイミングを確認する方法は、割り込み機能およびオートリード機能の設定によって異なります。各機能における A/D 変換終了タイミングの確認方法は②と同様です。

3.3.2 ダブルトリガモード

グループスキャンモードでダブルトリガモードを選択した場合は、グループ A は非同期または同期トリガ（TMR、ELC）で開始するシングルスキャンモードの実行 2 回分を一連の動作として実行します。グループ B とグループ C は同期トリガ（TMR、ELC）で開始するシングルスキャンモードと同じ動作になります。

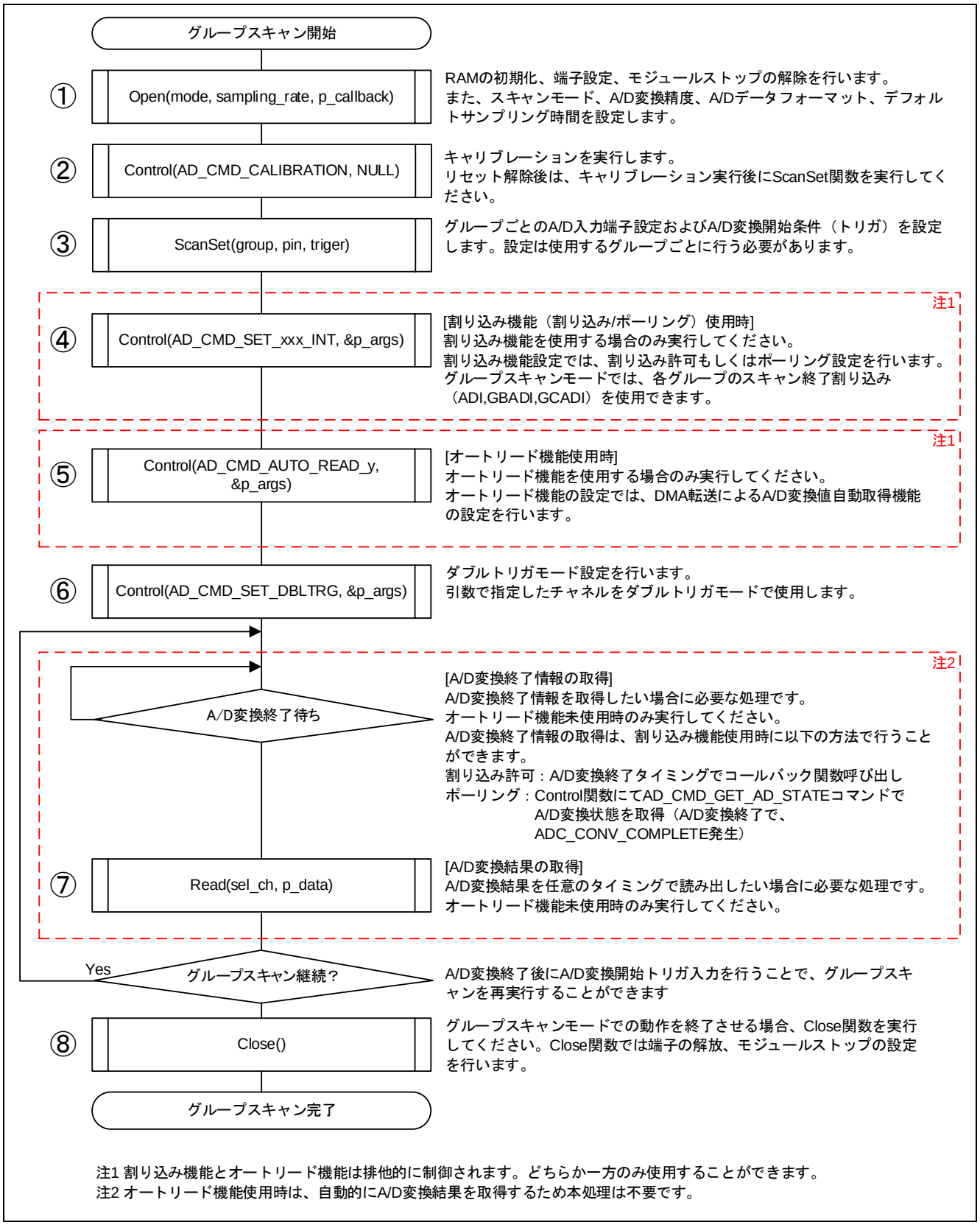


図 3-9 グループスキャン（ダブルトリガモード）実行手順

グループスキャンモード設定で使用する関数および引数の一覧を表 3-5、表 3-6 に示します。

表 3-5 グループスキャンモードの使用関数および引数一覧(1/2)

関数	引数	内容	詳細
①Open 関数 関数詳細「2.4.1」	mode	uint32_t 型 スキャンモードにグループスキャン（ADC_GROUP_SCAN）を指定します 必要に応じて A/D 変換精度、A/D データフォーマットを組み合わせて指定してください	2.7.1
	sampling_rate	uint8_t 型 2～255 のデフォルトサンプリング時間(ADSSTRn(注)の初期値)を指定してください	-
	p_callback	adc_cd_event_t型 コールバック関数を指定してください NULLを指定するとS14ADのイベント発生時にコールバック関数は呼び出されません	2.7.22
②Control 関数 AD_CMD_CALIBRATION 機能詳細「2.5.21」	cmd	AD_CMD_CALIBRATION を指定してください	-
	p_args	NULL を指定してください	-
③ScanSet 関数 関数詳細「2.4.2」	group	e_adc_group_t 型 端子設定を行うグループを指定します 2 つのグループを使用する場合は、グループ A およびグループ B の端子設定をそれぞれ実行してください 3 つのグループを使用する場合は、グループ A、グループ B、グループ C の端子設定をそれぞれ実行してください	2.7.6
	pins	st_adc_pins_t 型 A/D 変換入力端子を指定します	2.8.1
	triger	e_adc_triger_t 型 A/D 変換開始トリガを指定します グループスキャンモードでは、ソフトウェアトリガは使用できません	2.7.2
④Control 関数 AD_CMD_SET_xxx_INT (xxx = ADI、GROUPB、GROUPC) 機能詳細(ADI)「2.5.4」 機能詳細(GROUPB)「2.5.5」 機能詳細(GROUPC)「2.5.6」	cmd	e_adc_cmd_t 型 使用する割り込み要求ごとにコマンドを実行してください グループスキャンモードでは、以下の割り込み要求を使用できます AD_CMD_SET_ADI_INT：グループ A スキャン終了割り込み AD_CMD_SET_GROUPB_INT：グループ B スキャン終了割り込み AD_CMD_SET_GROUPC_INT：グループ C スキャン終了割り込み	-
	p_args	e_adc_int_method_t 型変数のポインタ 割り込み設定（許可ポーリング）を指定します 割り込み許可：ADC_INT_ENABLE ポーリング：ADC_INT_POLLING	2.7.10
⑤Control 関数 AD_CMD_AUTO_READ_y (y = NORMAL、BLOCK) 機能詳細(NORMAL)「2.5.23」 機能詳細(BLOCK)「2.5.24」	cmd	AD_CMD_AUTO_READ_y (y = NORMAL、BLOCK) を指定してください	-
	p_args	st_adc_dma_read_info_t 型変数のポインタ st_adc_dma_read_info_t 型構造体変数の各要素に、DMA 転送起動要因、転送元レジスタ、転送先 RAM、転送回数、コールバック関数、転送サイズを指定してください	2.8.8

注　256KB グループ：n = 0～7、L、T　　1500KB グループ：n = 0～6、L、T

表 3-6 グループスキャンモードの使用関数および引数一覧(2/2)

関数	引数	内容	詳細
⑥Control 関数 AD_CMD_SET_DBLTRG 機能詳細「2.5.2」	cmd	AD_CMD_SET_DBLTRG を指定してください	-
	p_args	int8_t 型 ダブルトリガモードで使用するチャネルを指定してください ダブルトリガモードで使用するチャネルには、グループ A のチャネルのみ指定できます ダブルトリガモードを無効にする場合は、”-1”を指定してください	-
⑦Read 関数	sel_ch	e_adc_ssel_ch_t型 A/D変換結果を読み出すアナログ入力チャネルを一つ指定してください 2 回目のトリガによって A/D 変換した結果を読み出す場合は、ADC_SSEL_DBL を指定してください	2.7.5
	p_data	uint16_t 型変数のポインタ A/D 変換結果格納先アドレスを指定してください	-
⑧Close 関数	-	-	-

グループスキャンモード（グループ A、B、C 使用）で A/D 変換する場合の動作例を図 3-10 グループスキャン（ダブルトリガモード）動作に示します。動作例における各グループの A/D 変換開始トリガを以下のとおりです。

・グループ A：非同期トリガ（ADTRG0 端子入力）
・グループ B：同期トリガ（ELC_S14AD イベント）
・グループ C：同期トリガ（TMR_TCORA（TMR コンペアマッチ A））

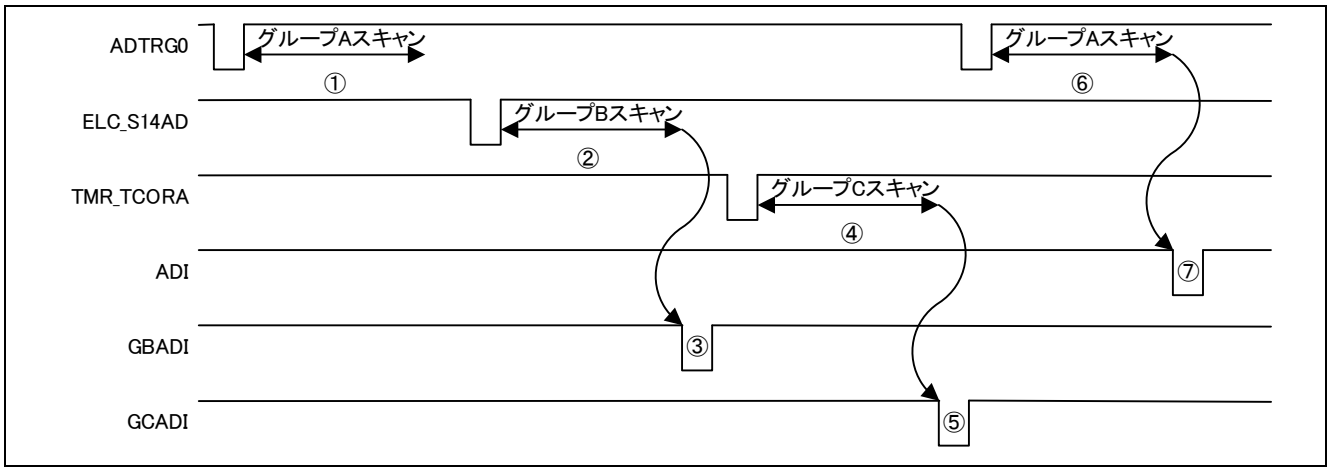


図 3-10 グループスキャン（ダブルトリガモード）動作

① ADTRG0 入力を検出すると、グループ A の A/D 変換が実行されます。A/D 変換の結果はグループ A の変換チャネルに対応した ADDRn(注)に格納されます。

② ELC_S14AD イベントを検出すると、グループ B の A/D 変換が実行されます。

③ グループ B の A/D 変換終了後、GBADI 割り込みが許可状態(ADCSR.GBADIE = 1）の場合 ADC140_GBADI 割り込み要求が発生します。A/D 変換終了タイミングを確認する方法は、割り込み機能およびオートリード機能の設定によって異なります。各機能における A/D 変換終了タイミングの確認を以下に示します。

[割り込み]

Open 関数実行時にコールバック関数を指定し、割り込み機能設定にて割り込みを許可状態にしていた場合、A/D 変換終了タイミングでコールバック関数が実行されます。

[ポーリング]

割り込み機能設定にてポーリング設定を行っていた場合、Control 関数の AD_CMD_GET_AD_STATE コマンドを実行することで各 A/D 変換グループの動作状態を取得できます。A/D 変換が完了したグループの動作状態情報には、ADC_CONV_COMPLETE が格納されます。AD_CMD_GET_AD_STATE コマンドの使用例および取得できる情報の詳細は、「2.5.17 A/D コンバータの状態取得機能 (AD_CMD_GET_AD_STATE)」を参照してください。

[オートリード]

オートリード機能を使用している場合、DMA 起動要因に指定したグループの A/D 変換終了タイミングで自動的に A/D 変換結果が RAM へ転送されます。A/D 変換終了ごとの通知はありません。オートリード機能設定時にコールバック関数を指定していた場合、オートリードが指定回数完了したタイミングでコールバック関数が実行されます。

④ TMR_TCORA イベントを検出すると、グループ C の A/D 変換が実行されます。

⑤ グループ C の A/D 変換終了後、GCADI 割り込みが許可状態(ADGCTRGR.GCADIE = 1)の場合 ADC140_GCADI 割り込み要求が発生します。グループ C の A/D 変換終了タイミングを確認する方法は、割り込み機能およびオートリード機能の設定によって異なります。各機能における A/D 変換終了タイミングの確認方法は③と同様です。

⑥ 2 回目の ADTRG0 入力を検出すると、グループ A の A/D 変換が実行されます。A/D 変換の結果は ADDBLDR に格納されます。

⑦ 2 回目のグループ A スキャン終了後、ADI 割り込みが許可状態(ADCSR.ADIE = 1）の場合 ADC140_ADI 割り込み要求が発生します。グループ A の A/D 変換終了タイミングを確認する方法は、割り込み機能およびオートリード機能の設定によって異なります。各機能における A/D 変換終了タイミングの確認方法は③と同様です。

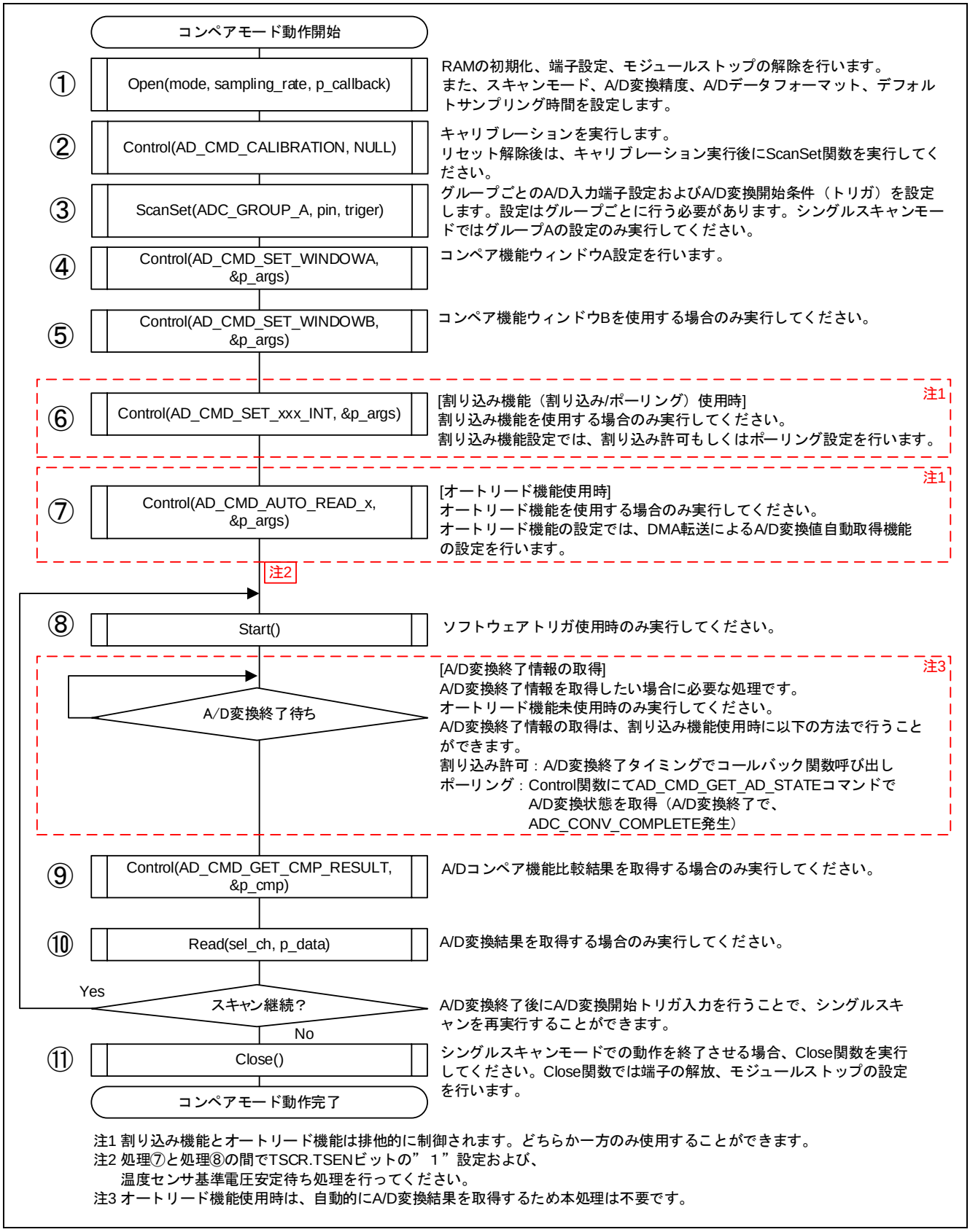
注 256KB グループ：n = 00 to 07,16,17,20 to 21　　1500KB グループ：n = 00 to 06,16,17,20 to 28

3.4 コンペア機能

3.4.1 コンペア機能（ウィンドウ A/ウィンドウ B/温度センサ出力）

コンペア機能は、レジスタに設定した基準値と A/D 変換結果を比較する機能です。基準値はウィンドウ A およびウィンドウ B それぞれに設定することができます。

コンペア機能のイベント出力は、ウィンドウ A およびウィンドウ B の比較条件成立/不成立からウィンドウ A/B 複合条件（A or B、A and B、A exor B）に応じて ADC140_WCMPM もしくは ADC140_WCMPUM イベントを出力します。コンペア機能のイベント出力を使用する場合は、シングルキャンモードを使用してください。



注1 割り込み機能とオートリード機能は排他的に制御されます。どちらか一方のみ使用することができます。
注2 処理⑦と処理⑧の間でTSCR.TSENビットの”1”設定および、
温度センサ基準電圧安定待ち処理を行ってください。
注3 オートリード機能使用時は、自動的にA/D変換結果を取得するため本処理は不要です。

図 3-11 コンペアモード実行手順

コンペアモード設定で使用する関数および引数の一覧を表 3-7、表 3-8 に示します。

表 3-7 コンペア機能使用関数および引数一覧(1/2)

関数	引数	内容	詳細
①Open 関数 関数詳細「2.4.1」	mode	uint32_t 型 スキャンモードを指定します コンペア機能のイベント出力（ADC140_WCMPM/ADC140_WCMPUM）を使用する場合は、シングルスキャンモードに設定してください 必要に応じて A/D 変換精度、A/D データフォーマットを組み合わせて指定してください	2.7.1
	sampling_rate	uint8_t 型 2～255 のデフォルトサンプリング時間(ADSSTRn(注)の初期値)を指定してください	-
	p_callback	adc_cd_event_t型 コールバック関数を指定してください NULL を指定すると S14AD のイベント発生時にコールバック関数は呼び出されません	2.7.22
②Control 関数 AD_CMD_CALIBRATION 機能詳細「2.5.21」	cmd	AD_CMD_CALIBRATION を指定してください	-
	p_args	NULL を指定してください	-
③ScanSet 関数 関数詳細「2.4.2」	group	e_adc_group_t 型 端子設定を行うグループを指定します 端子設定は使用するグループごとに実行してください	2.7.6
	pins	st_adc_pins_t 型 A/D 変換入力端子を指定します	2.8.1
	triger	e_adc_triger_t 型 A/D 変換開始トリガを指定します	2.7.2
④Control 関数 AD_CMD_SET_WINDOWA 機能詳細「2.5.13」	cmd	AD_CMD_SET_WINDOWA を指定してください	-
	p_args	st_adc_wina_t 型変数のポインタ st_adc_wina_t 型構造体変数の各要素に、CMPAI 割り込み設定、比較条件設定、比較レベル設定、比較端子設定、温度センサ使用設定、チャネル別比較条件設定、温度センサ比較条件設定を指定してください	2.8.3
⑤Control 関数 AD_CMD_SET_WINDOWB 機能詳細「2.5.14」	cmd	AD_CMD_SET_WINDOWA を指定してください	-
	p_args	st_adc_winb_t 型変数のポインタ st_adc_winb_t 型構造体変数の各要素に、CMPBI 割り込み設定、比較条件設定、ウィンドウ A/B 複合条件設定、比較レベル、比較端子設定、温度センサ使用設定を指定してください	2.8.4

注　256KB グループ：n = 0～7、L、T　　1500KB グループ：n = 0～6、L、T

表 3-8 コンペア機能使用関数および引数一覧(2/2)

関数	引数	内容	詳細
⑥Control 関数 AD_CMD_SET_xxx_INT (xxx = ADI、GROUPB、GROUPC、WCMPM、WCMPUM) 機能詳細(ADI)「2.5.4」 機能詳細(GROUPB)「2.5.5」 機能詳細(GROUPC)「2.5.6」 機能詳細(WCMPM)「2.5.7」 機能詳細(WCMPUM)「2.5.8」	cmd	e_adc_cmd_t 型 使用する割り込み要求ごとにコマンドを実行してください	-
	p_args	e_adc_int_method_t 型変数のポインタ 割り込み設定（許可ポーリング）を指定します 割り込み許可：ADC_INT_ENABLE ポーリング：ADC_INT_POLLING	2.7.10
⑦Control 関数 AD_CMD_AUTO_READ_y (y = NORMAL、BLOCK、COMPARE) 機能詳細(NORMAL)「2.5.23」 機能詳細(BLOCK)「2.5.24」 機能詳細(COMPARE)「2.5.25」	cmd	AD_CMD_AUTO_READ_y (y = NORMAL、BLOCK、COMPARE) のいずれか指定してください	-
	p_args	st_adc_dma_read_info_t 型変数のポインタ st_adc_dma_read_info_t 型構造体変数の各要素に、DMA 転送起動要因、転送元レジスタ、転送先 RAM、転送回数、コールバック関数、転送サイズを指定してください	2.8.8
⑧Start 関数	-	-	-
⑨Control 関数 AD_CMD_GET_CMP_RESULT 機能詳細「2.5.16」	cmd	AD_CMD_GET_CMP_RESULT	
	p_cmp	st_adc_cmp_result_t 型変数のポインタ 取得するコンペア機能比較結果の格納先アドレスを指定してください	2.8.6
⑩Read 関数	sel_ch	e_adc_ssel_ch_t 型 A/D 変換結果を読み出すアナログ入力チャネルを一つ指定してください	2.7.5
	p_data	uint16_t 型変数のポインタ A/D 変換結果格納先アドレスを指定してください	-
⑪Close 関数	-	-	-

シングルスキャンモードでコンペア機能（ウィンドウ A/B）を使用する場合の動作例を図 3-12 に示します。また、動作例における S14AD のモードおよびコンペア機能設定を以下に示します。
・コンペア機能ウィンドウ A/B：許可
・コンペア機能ウィンドウ A チャネル：AN000
・コンペア機能ウィンドウ B チャネル：AN001
・コンペア機能ウィンドウ A/B 複合条件：ウィンドウ A 比較条件に一致 AND ウィンドウ B 比較条件に一致で ADC140_WCMPM を出力、それ以外は ADC140_WCMPUM を出力

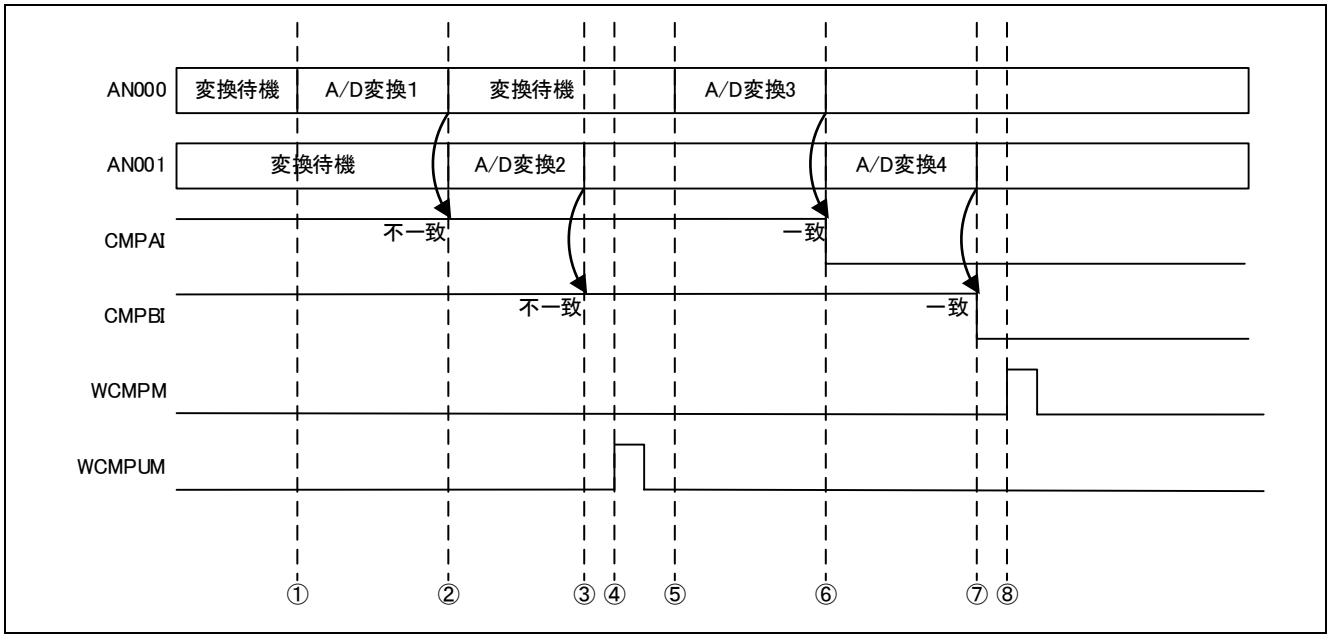


図 3-12 コンペア機能ウィンドウ A/B 動作

① A/D 変換開始トリガ入力によって A/D 変換が開始されます。

② AN000 チャネルの A/D 変換完了後、A/D 変換結果がコンペア機能ウィンドウ A の比較条件と一致しない場合、CMPAI 割り込み要求は発生しません。

③ AN001 チャネルの A/D 変換完了後、A/D 変換結果がコンペア機能ウィンドウ B の比較条件と一致しない場合、CMPBI 割り込み要求は発生しません。

④ コンペア機能ウィンドウ A/B の比較結果が複合条件に一致しない場合、WCMPUM 割り込み要求が発生します。

WCMPUM 割り込み要求の発生を確認する方法は、割り込み機能設定およびオートリード機能の設定によって異なります。各機能における割り込み要求の発生確認方法を以下に示します。

[割り込み]

Open 関数実行時にコールバック関数を指定し、割り込み機能設定にて割り込みを許可状態にしていた場合、A/D 変換終了タイミングでコールバック関数が実行されます。

[ポーリング]

割り込み機能設定にてポーリング設定を行っていた場合、Control 関数の AD_CMD_GET_AD_STATE コマンドを実行することで、ウィンドウ A/B コンペア一致/不一致イベント情報を取得できます。イベントを検出した場合、ウィンドウ A/B コンペア一致もしくは不一致イベント情報に ADC_CONV_WCMP_DETECT が格納されます。AD_CMD_GET_AD_STATE コマンドの使用例および取得できる情報の詳細は、「2.5.17 A/D コンバータの状態取得機能(AD_CMD_GET_AD_STATE)」を参照してください。

[オートリード]

オートリード機能を使用している場合、DMA 起動要因に指定した割り込み要求発生タイミングで自動的に任意のレジスタ値が RAM へ転送されます。割り込み要求発生ごとの通知はありません。オートリード機能設定時にコールバック関数を指定していた場合、オートリードが指定回数完了したタイミングでコールバック関数が実行されます。

⑤ 2 回目の A/D 変換開始トリガ入力によって A/D 変換が開始されます。

⑥ AN000 チャネルの A/D 変換完了後、A/D 変換結果がコンペア機能ウィンドウ A の比較条件と一致すると、コールバック関数が呼び出されます(注)。

⑦ AN001 チャネルの A/D 変換完了後、A/D 変換結果がコンペア機能ウィンドウ B の比較条件と一致すると、コールバック関数が呼び出されます(注)。

⑧ コンペア機能ウィンドウ A/B の比較結果が複合条件に一致した場合、WCMPM 割り込み要求が発生します。WCMPM 割り込み要求は、A/D 変換スキャン終了イベント（ADC140_ELC）から 1PCLKB 遅れて発生します。

WCMPUM 割り込み要求の発生を確認する方法は、割り込み機能設定およびオートリード機能の設定によって異なります。各機能における割り込み要求の発生確認方法は④と同様です。

注 以下の条件を満たした場合のみ、コールバック関数が呼び出されます。

・Open 関数でコールバック関数を指定していた場合

・Control 関数のコンペア機能設定コマンド(AD_CMD_SET_WINDOWA/AD_CMD_SET_WINDOWB)を実行する際に割り込み許可にしていた場合

3.5 コンフィグレーション

S14AD ドライバは、ユーザが設定可能なコンフィグレーションを r_adc_cfg.h ファイルに用意します。

3.5.1 パラメータチェック

S14AD ドライバにおけるパラメータチェックの有効/無効を設定します。

名称：ADC_CFG_PARAM_CHECKING_ENABLE

表 3-9 ADC_CFG_PARAM_CHECKING_ENABLE の設定

設定値	内容
(0)	パラメータチェックを無効にします 関数仕様に記載している引数の妥当性判定に関するエラーの検出を行いません
(1) （初期値）	パラメータチェックを有効にします 関数仕様に記載している引数の妥当性判定に関するエラーの検出を行います

3.5.2 ADI 割り込み要求制御

シングルスキャンもしくはグループ A スキャン終了割り込み要求（ADC140_ADI）で起動させる機能を選択します。

名称：S14AD_ADI_CONTROL

表 3-10 A/D 割り込み要求制御方法の定義

定義	値	内容
S14AD_USED_INTERRUPT （初期値）	(0)	ADC140_ADI 割り込み要求を割り込みもしくはポーリングで使用
S14AD_USED_DMAC0	(1<<0)	ADC140_ADI 割り込み要求で DMAC0 を起動
S14AD_USED_DMAC1	(1<<1)	ADC140_ADI 割り込み要求で DMAC1 を起動
S14AD_USED_DMAC2	(1<<2)	ADC140_ADI 割り込み要求で DMAC2 を起動
S14AD_USED_DMAC3	(1<<3)	ADC140_ADI 割り込み要求で DMAC3 を起動
S14AD_USED_DTC	(1<<15)	ADC140_ADI 割り込み要求で DTC を起動

3.5.3 GBADI 割り込み要求制御

グループ B スキャン終了割り込み要求（ADC140_GBADI）で起動させる機能を選択します。

名称：S14AD_GBADI_CONTROL

表 3-11 A/D 割り込み要求制御方法の定義

定義	値	内容
S14AD_USED_INTERRUPT （初期値）	(0)	ADC140_GBADI 割り込み要求を割り込みもしくはポーリングで使用
S14AD_USED_DMAC0	(1<<0)	ADC140_GBADI 割り込み要求で DMAC0 を起動
S14AD_USED_DMAC1	(1<<1)	ADC140_GBADI 割り込み要求で DMAC1 を起動
S14AD_USED_DMAC2	(1<<2)	ADC140_GBADI 割り込み要求で DMAC2 を起動
S14AD_USED_DMAC3	(1<<3)	ADC140_GBADI 割り込み要求で DMAC3 を起動
S14AD_USED_DTC	(1<<15)	ADC140_GBADI 割り込み要求で DTC を起動

3.5.4 GCADI 割り込み要求制御

グループ C スキャン終了割り込み要求（ADC140_GCADI）で起動させる機能を選択します。

名称：S14AD_GCADI_CONTROL

表 3-12 A/D 割り込み要求制御方法の定義

定義	値	内容
S14AD_USED_INTERRUPT（初期値）	(0)	ADC140_GCADI 割り込み要求を割り込みもしくはポーリングで使用
S14AD_USED_DMAC0	(1<<0)	ADC140_GCADI 割り込み要求で DMAC0 を起動
S14AD_USED_DMAC1	(1<<1)	ADC140_GCADI 割り込み要求で DMAC1 を起動
S14AD_USED_DMAC2	(1<<2)	ADC140_GCADI 割り込み要求で DMAC2 を起動
S14AD_USED_DMAC3	(1<<3)	ADC140_GCADI 割り込み要求で DMAC3 を起動
S14AD_USED_DTC	(1<<15)	ADC140_GCADI 割り込み要求で DTC を起動

3.5.5 WCMPM 割り込み要求制御

ウィンドウ A/B コンペア機能の条件一致割り込み要求（ADC140_WCMPM）で起動させる機能を選択します。

名称：S14AD_WCMPM_CONTROL

表 3-13 A/D 割り込み要求制御方法の定義

定義	値	内容
S14AD_USED_INTERRUPT（初期値）	(0)	ADC140_WCMPM 割り込み要求を割り込みもしくはポーリングで使用
S14AD_USED_DMAC0	(1<<0)	ADC140_WCMPM 割り込み要求で DMAC0 を起動
S14AD_USED_DMAC1	(1<<1)	ADC140_WCMPM 割り込み要求で DMAC1 を起動
S14AD_USED_DMAC2	(1<<2)	ADC140_WCMPM 割り込み要求で DMAC2 を起動
S14AD_USED_DMAC3	(1<<3)	ADC140_WCMPM 割り込み要求で DMAC3 を起動
S14AD_USED_DTC	(1<<15)	ADC140_WCMPM 割り込み要求で DTC を起動

3.5.6 WCMPUM 割り込み要求制御

ウィンドウ A/B コンペア機能の条件不一致割り込み要求（ADC140_WCMPUM）で起動させる機能を選択します。

名称：S14AD_WCMPUM_CONTROL

表 3-14 A/D 割り込み要求制御方法の定義

定義	値	内容
S14AD_USED_INTERRUPT（初期値）	(0)	ADC140_WCMPUM 割り込み要求を割り込みもしくはポーリングで使用
S14AD_USED_DMAC0	(1<<0)	ADC140_WCMPUM 割り込み要求で DMAC0 を起動
S14AD_USED_DMAC1	(1<<1)	ADC140_WCMPUM 割り込み要求で DMAC1 を起動
S14AD_USED_DMAC2	(1<<2)	ADC140_WCMPUM 割り込み要求で DMAC2 を起動
S14AD_USED_DMAC3	(1<<3)	ADC140_WCMPUM 割り込み要求で DMAC3 を起動
S14AD_USED_DTC	(1<<15)	ADC140_WCMPUM 割り込み要求で DTC を起動

3.5.7 ADI 割り込み優先レベル

ADI 割り込みの優先レベルを設定します。

名称：S14AD_ADI_PRIORITY

表 3-15 S14AD_ADI_PRIORITY の設定

設定値	内容
0	割り込み優先レベルを 0（最高）に設定
1	割り込み優先レベルを 1 に設定
2	割り込み優先レベルを 2 に設定
3（初期値）	割り込み優先レベルを 3（最低）に設定

3.5.8 GBADI 割り込み優先レベル

GBADI 割り込みの優先レベルを設定します。

名称：S14AD_GBADI_PRIORITY

表 3-16 S14AD_GBADI_PRIORITY の設定

設定値	内容
0	割り込み優先レベルを 0（最高）に設定
1	割り込み優先レベルを 1 に設定
2	割り込み優先レベルを 2 に設定
3（初期値）	割り込み優先レベルを 3（最低）に設定

3.5.9 GCADI 割り込み優先レベル

GCADI 割り込みの優先レベルを設定します。

名称：S14AD_GCADI_PRIORITY

表 3-17 S14AD_GCADI_PRIORITY の設定

設定値	内容
0	割り込み優先レベルを 0（最高）に設定
1	割り込み優先レベルを 1 に設定
2	割り込み優先レベルを 2 に設定
3（初期値）	割り込み優先レベルを 3（最低）に設定

3.5.10 CMPAI 割り込み優先レベル

CMPAI 割り込みの優先レベルを設定します。

名称：S14AD_CMPAI_PRIORITY

表 3-18 S14AD_CMPAI_PRIORITY の設定

設定値	内容
0	割り込み優先レベルを 0（最高）に設定
1	割り込み優先レベルを 1 に設定
2	割り込み優先レベルを 2 に設定
3（初期値）	割り込み優先レベルを 3（最低）に設定

3.5.11 CMPBI 割り込み優先レベル

CMPBI 割り込みの優先レベルを設定します。

名称：S14AD_CMPBI_PRIORITY

表 3-19 S14AD_CMPBI_PRIORITY の設定

設定値	内容
0	割り込み優先レベルを 0（最高）に設定
1	割り込み優先レベルを 1 に設定
2	割り込み優先レベルを 2 に設定
3（初期値）	割り込み優先レベルを 3（最低）に設定

3.5.12 WCMPM 割り込み優先レベル

WCMPM 割り込みの優先レベルを設定します。

名称：S14AD_WCMPM_PRIORITY

表 3-20 S14AD_WCMPM_PRIORITY の設定

設定値	内容
0	割り込み優先レベルを 0（最高）に設定
1	割り込み優先レベルを 1 に設定
2	割り込み優先レベルを 2 に設定
3（初期値）	割り込み優先レベルを 3（最低）に設定

3.5.13 WCMPUM 割り込み優先レベル

WCMPUM 割り込みの優先レベルを設定します。

名称：S14AD_WCMPUM_PRIORITY

表 3-21 S14AD_WCMPUM_PRIORITY の設定

設定値	内容
0	割り込み優先レベルを 0（最高）に設定
1	割り込み優先レベルを 1 に設定
2	割り込み優先レベルを 2 に設定
3（初期値）	割り込み優先レベルを 3（最低）に設定

3.5.14 CMPAI スヌーズモード使用設定

低消費電力モードのスヌーズモードでのみコンペア機能ウィンドウ A を使用するかどうかを設定します。

名称：ADC_CMPAI_SNOOZE_USE

表 3-22 ADC_CMPAI_SNOOZE_USE の設定

設定値	内容
0（初期値）	コンペア機能ウィンドウ A を CMPAI 割り込みで使用する （CMPAI 割り込み設定を行う）(注 1)
1	コンペア機能ウィンドウ A を低消費電力モードのスヌーズモードでのみ使用する （CMPAI 割り込み設定をスキップする）(注 2)

注1. 低消費電力モードのスヌーズモードでコンペア機能ウィンドウ A を使用し、かつ CMPAI 割り込み設定を行う場合、"0"を設定してください。"1"を設定した場合、CMPAI 割り込みは使用できません。

注2. CMPAI 割り込みを使用せず、AD_CMD_GET_CMP_RESULT コマンドを使用してポーリングを行う場合、"1"を設定してください。

3.5.15 CMPBI スヌーズモード使用設定

低消費電力モードのスヌーズモードでのみコンペア機能ウィンドウ B を使用するかどうかを設定します。

名称：ADC_CMPBI_SNOOZE_USE

表 3-23 ADC_CMPBI_SNOOZE_USE の設定

設定値	内容
0（初期値）	コンペア機能ウィンドウ B を CMPBI 割り込みで使用する （CMPBI 割り込み設定を行う）(注 1)
1	コンペア機能ウィンドウ B を低消費電力モードのスヌーズモードでのみ使用する （CMPBI 割り込み設定をスキップする）(注 2)

注1. 低消費電力モードのスヌーズモードでコンペア機能ウィンドウ B を使用し、かつ CMPBI 割り込み設定を行う場合、"0"を設定してください。"1"を設定した場合、CMPBI 割り込みは使用できません。

注2. CMPBI 割り込みを使用せず、AD_CMD_GET_CMP_RESULT コマンドを使用してポーリングを行う場合、"1"を設定してください。

3.5.16 関数の RAM 配置

S14AD ドライバの特定関数を RAM で実行するための設定を行います。

関数の RAM 配置を設定するコンフィグレーションは、関数ごとに定義を持ちます。

名称：S14AD_CFG_xxx

xxx には関数名をすべて大文字で記載

例） R_ADC_Open 関数 → S14AD_CFG_R_ADC_OPEN

表 3-24 ADC_CFG_SECTION_xxx の設定

設定値	内容
SYSTEM_SECTION_CODE	関数を RAM に配置しません
SYSTEM_SECTION_RAM_FUNC	関数を RAM に配置します

表 3-25 各関数の RAM 配置初期状態

番号	関数名	RAM 配置
1	R_ADC_Open	
2	R_ADC_ScanSet	
3	R_ADC_Start	
4	R_ADC_Stop	
5	R_ADC_Control	
6	R_ADC_Read	
7	R_ADC_Close	
8	R_ADC_GetVersion	
9	adc_s14adi0_isr (ADI 割り込み処理)	✔
10	adc_gbadi_isr(GBADI 割り込み処理)	✔
11	adc_gcadi_isr(GCADI 割り込み処理)	✔
12	adc_cmpai_isr(CMPAI 割り込み処理)	✔
13	adc_cmpbi_isr(CMPBI 割り込み処理)	✔
14	adc_wcmpm_isr(WCMPM 割り込み処理)	✔
15	adc_wcmpum_isr(WCMPUM 割り込み処理)	✔

4. ドライバ詳細情報

本章では、本ドライバ機能を構成する詳細仕様について説明します。

4.1 関数仕様

S14AD ドライバの各関数の仕様と処理フローを示します。

処理フロー内では条件分岐などの判定方法の一部を省略して記述しているため、実際の処理と異なる場合があります。

4.1.1 R_ADC_Open 関数

表 4-1 R_ADC_Open 関数仕様

書式	e_adc_err_t R_ADC_Open(uint32_t mode, uint8_t default_sampling_rate, adc_cd_event_t const p_callback, st_adc_resources_t const * const p_adc)
仕様説明	S14AD ドライバの初期化（RAM の初期化、レジスタ設定、端子設定、モジュールストップの解除）を行います 同時にスキャンモード、A/D 変換精度、A/D データレジスタフォーマット、デフォルトサンプリング時間の設定も行います
引数	uint32_t mode：スキャンモード、A/D 変換精度、A/D データレジスタフォーマットを組み合わせて指定します
	uint8_t default_sampling_rate：デフォルトサンプリング時間 2～255(注 1)のデフォルトサンプリング時間(ADSSTRn(注 2)の初期値)を指定します
	adc_cd_event_t const p_callback：コールバック関数 S14AD のイベント発生時のコールバック関数を指定します。NULL を設定した場合、S14AD のイベント発生時にコールバック関数が実行されません。
	st_adc_resources_t const * const p_adc：S14AD のリソース 初期化する S14AD のリソースを指定します
戻り値	ADC_OK S14AD の初期化成功
	ADC_ERROR_PARAMETER パラメータエラー 以下のいずれかの条件を検出するとパラメータエラーとなります ・スキャンモードに規定外の値を指定した場合 ・A/D 変換精度設定に規定外の値を指定した場合 ・A/D データレジスタフォーマットに規定外の値を指定した場合 ・デフォルトサンプリング時間設定に 0 または 1 を設定した場合
	ADC_ERROR S14AD の初期化失敗 以下のいずれかの条件を検出するとパラメータエラーとなります ・Open 関数をすでに実行していた場合 ・すでにモジュールストップが解除されていた場合

備考	インスタンスからのアクセス時は S14AD リソースの指定は不要です。
	
	[インスタンスからの関数呼び出し例]
	static void callback(uint32_t event);
	
	// S14AD driver instance
	extern DRIVER_S14AD Driver_S14AD;
	DRIVER_S14AD *adcDev = &Driver_S14AD;
	
	main()
	{
	adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, callback);
	}

注1. ADSCLKCR.SCLKEN ビットが 0（サブクロックモード無効）の場合、5～255 を設定してください。
ADSCLKCR.SCLKEN ビットが 1（サブクロックモード有効）の場合、2～8 を設定してください。

注2. 256KB グループ：n = 0～7、L、T 1500KB グループ：n = 0～6、L、T

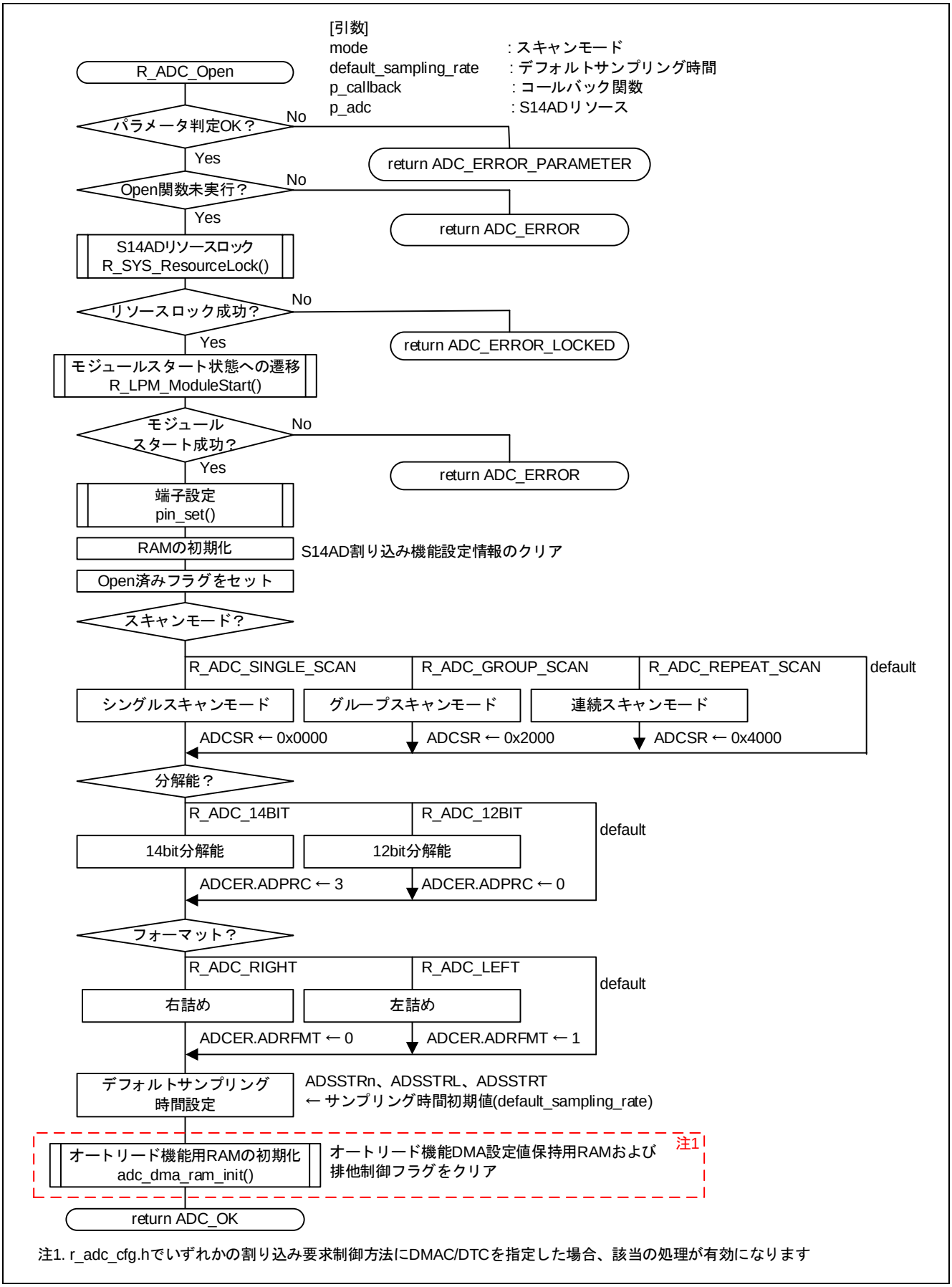


図 4-1 R_ADC_Open 関数処理フロー

4.1.2 R_ADC_Close 関数

表 4-2 R_ADC_Close 関数仕様

書式	e_adc_err_t R_ADC_Close(st_adc_resources_t * const p_adc)
仕様説明	S14AD ドライバを解放します
引数	st_adc_resources_t *p_adc：S14AD のリソース 解放する S14AD のリソースを指定します
戻り値	ADC_OK　　S14AD の解放成功

備考	インスタンスからのアクセス時は S14AD リソースの指定は不要です。
	
	[インスタンスからの関数呼び出し例]
	// S14AD driver instance
	extern DRIVER_S14AD Driver_S14AD;
	DRIVER_S14AD *adcDev = &Driver_S14AD;
	
	main()
	{
	　adcDev->Close();
	}

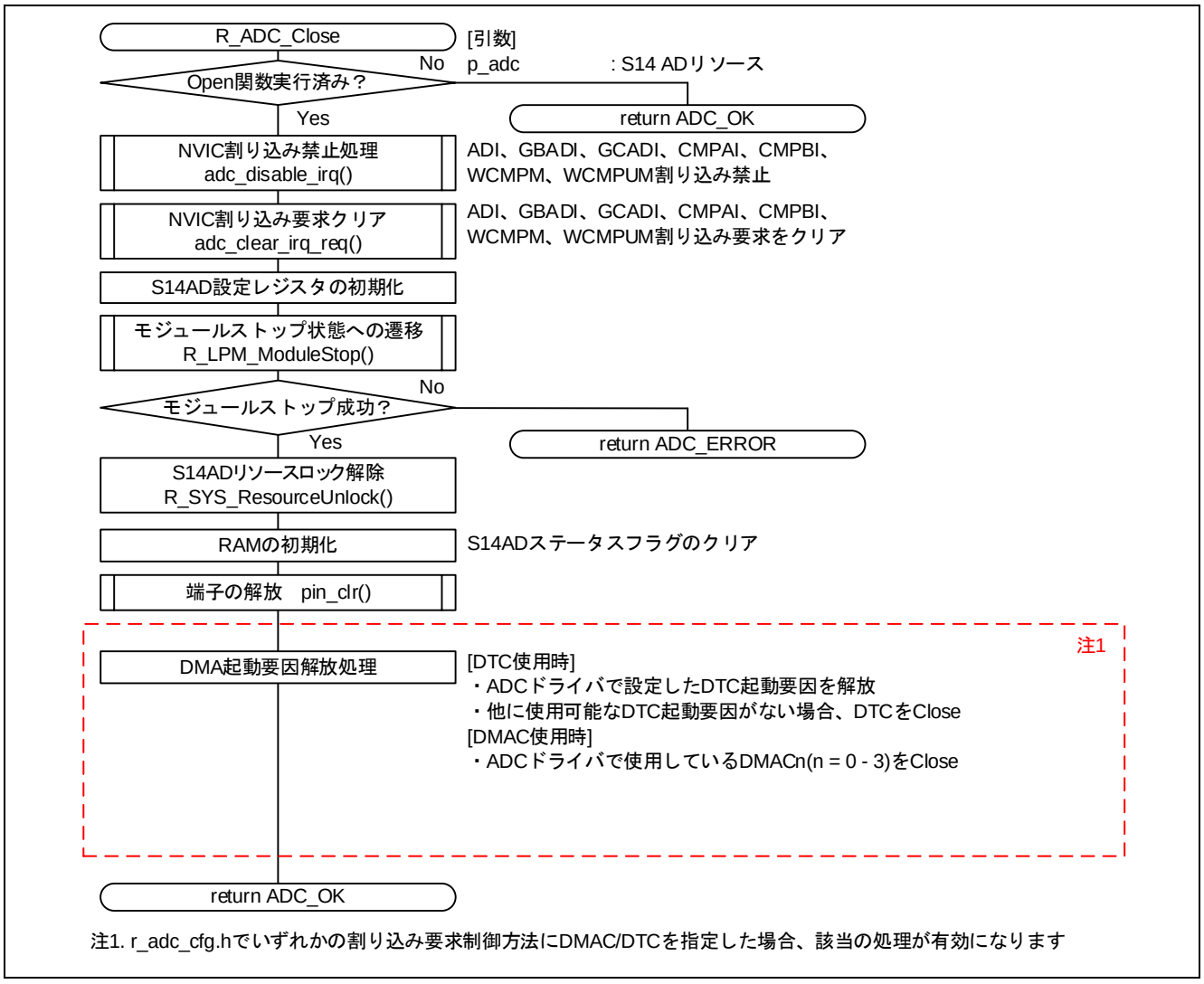


図 4-2 R_ADC_Close 関数処理フロー

4.1.3 R_ADC_ScanSet 関数

表 4-3 R_ADC_ScanSet 関数仕様

書式	e_adc_err_t R_ADC_ScanSet(e_adc_group_t group, st_adc_pins_t pins , e_adc_triger_t triger, st_adc_resources_t * const p_adc)
仕様説明	グループごとの A/D 変換対象チャネル、A/D 変換開始トリガを設定します
引数	e_adc_group_t group：設定を行う A/D 変換グループを指定します 以下のいずれかを設定します ADC_GROUP_A：グループ A ADC_GROUP_B：グループ B ADC_GROUP_C：グループ C
	st_adc_pins_t pins：A/D 変換対象チャネルを指定します
	e_adc_triger_t triger：A/D 変換開始トリガを指定します 以下のいずれか設定します ADC_TRIGER_SOFT：ソフトウェアトリガ ADC_TRIGER_TMR：TCORA レジスタと TCNT カウンタのコンペアマッチ ADC_TRIGER_ELC：ELC_S14AD ADC_TRIGER_ADTRG：ADTRG0 入力 ADC_TRIGER_LOW_PRIORITY_CONT_SCAN：トリガ要因非選択
	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	ADC_OK　A/D 変換対象チャネルおよび A/D 変換開始トリガ設定成功
	ADC_ERROR　A/D 変換対象チャネルおよび A/D 変換開始トリガ設定失敗 S14AD の未初期化状態で実行した場合、A/D 変換対象チャネルおよび A/D 変換開始トリガ設定失敗となります。
	ADC_ERROR_PARAMETER　パラメータエラー 以下のいずれかの条件を検出するとパラメータエラーとなります ・A/D 変換対象チャネルに端子が指定されていない場合 ・存在しないトリガ設定を指定した場合 ・グループ B およびグループ C の A/D 変換開始トリガにソフトウェアトリガを指定した場合 ・グループ B およびグループ C の A/D 変換開始トリガに外部トリガを指定した場合
	ADC_ERROR_MODE　モードエラー 以下のいずれかの条件を検出するとモードエラーとなります ・グループスキャンモードでグループ A にソフトウェアトリガを指定した場合 ・グループスキャンモード以外でグループ B およびグループ C を指定した場合

備考	インスタンスからのアクセス時は S14AD リソースの指定は不要です。
	
	[インスタンスからの関数呼び出し例]
	// S14AD driver instance
	extern DRIVER_S14AD Driver_S14AD;
	DRIVER_S14AD *adcDev = &Driver_S14AD;
	
	main()
	{
	st_adc_pins_t scanset_pin;　/* A/D 変換チャネル設定用変数 */
	scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN03;
	/* A/D 変換に AN000、AN001、AN03 を使用 */
	scanset_pin.sensor　= ADC_SENSOR_NOTUSE;　/* A/D 変換に温度センサを使用しない */
	adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);
	}

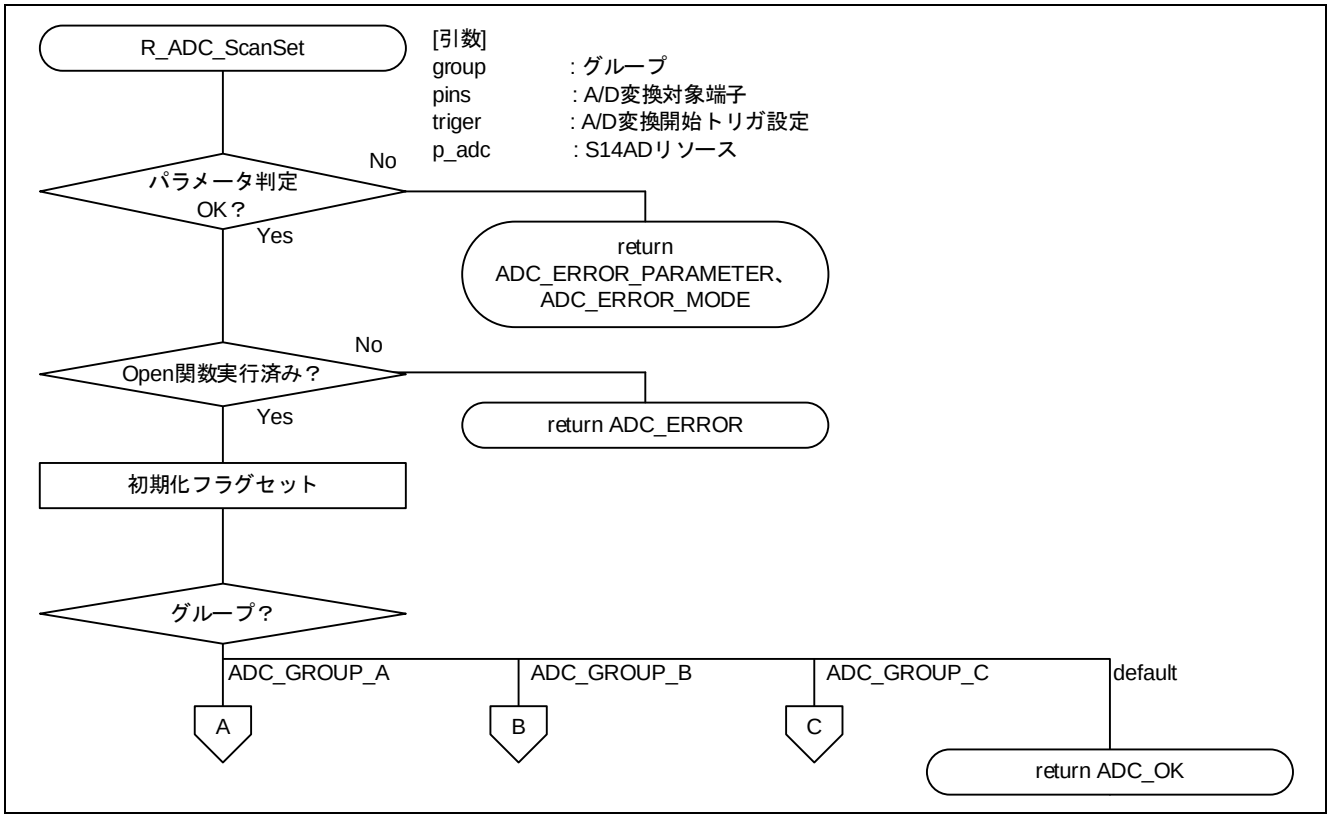


図 4-3 R_ADC_ScanSet 関数処理フロー(1/4)

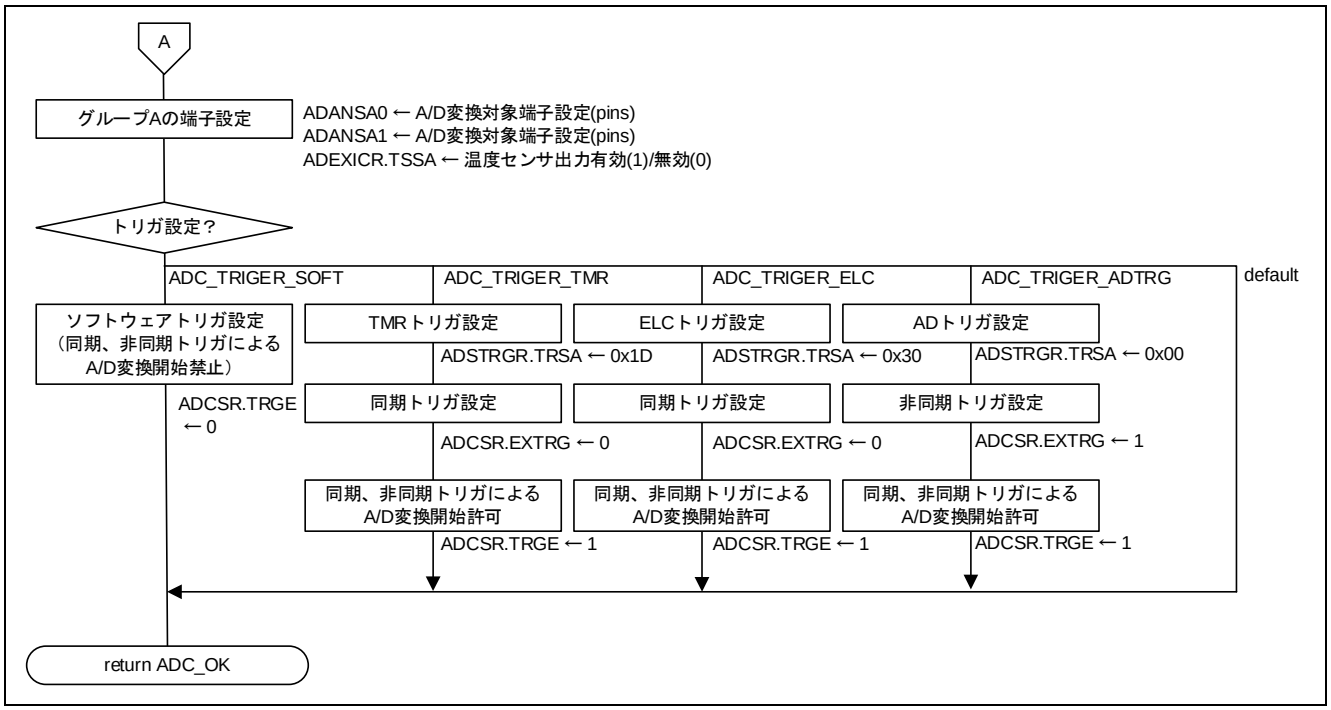


図 4-4 R_ADC_ScanSet 関数処理フロー(2/4)

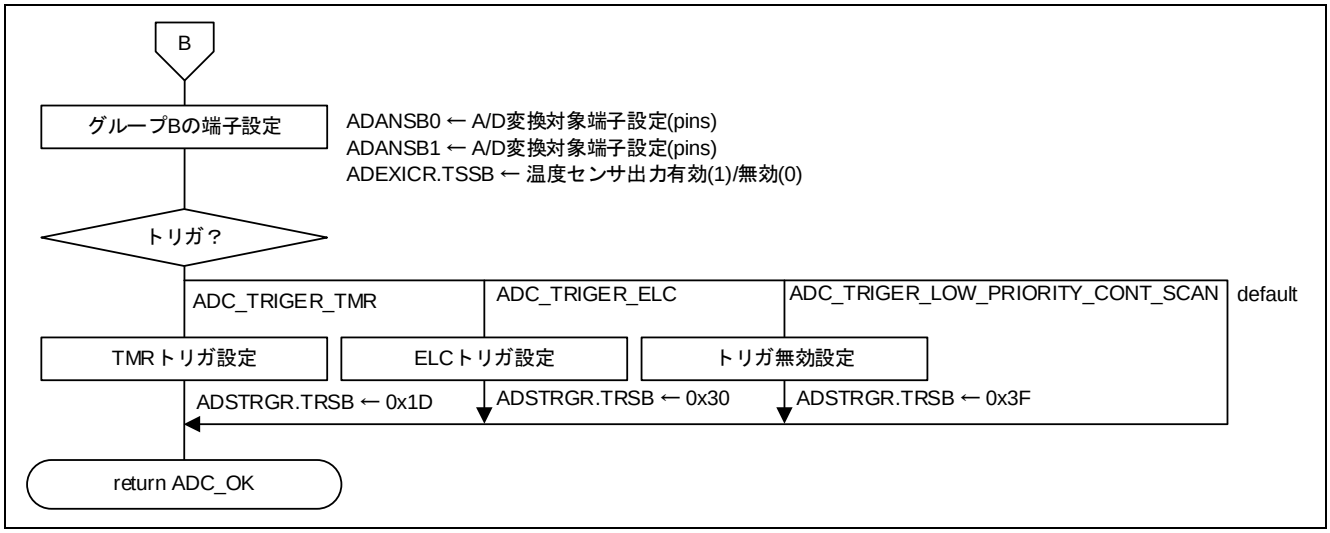


図 4-5 R_ADC_ScanSet 関数処理フロー(3/4)

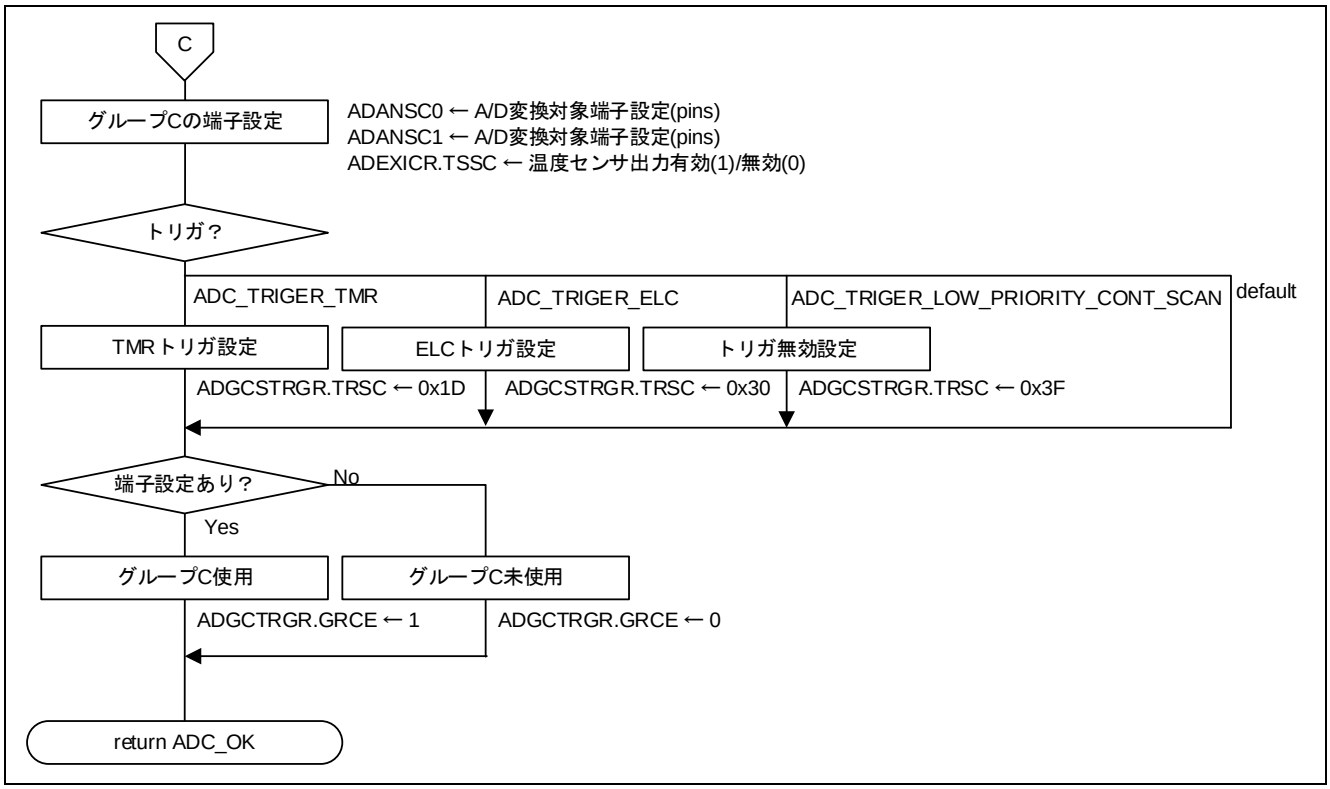


図 4-6 R_ADC_ScanSet 関数処理フロー(4/4)

4.1.4 R_ADC_Start 関数

表 4-4 R_ADC_Start 関数仕様

書式	e_adc_err_t R_ADC_Start(st_adc_resources_t * const p_adc)
仕様説明	ソフトウェアトリガによる A/D 変換を開始します
引数	st_adc_resources_t *p_adc：S14AD のリソース 送信する S14AD のリソースを指定します
戻り値	ADC_OK　A/D 変換開始成功 ADC_ERROR　A/D 変換開始失敗 S14AD の未初期化状態で実行した場合、A/D 変換開始失敗となります

備考	インスタンスからのアクセス時は S14AD リソースの指定は不要です。
	
	[インスタンスからの関数呼び出し例]
	// S14AD driver instance
	extern DRIVER_S14AD Driver_S14AD;
	DRIVER_S14AD *adcDev = &Driver_S14AD;
	
	main()
	{
	　adcDev->Start();
	}

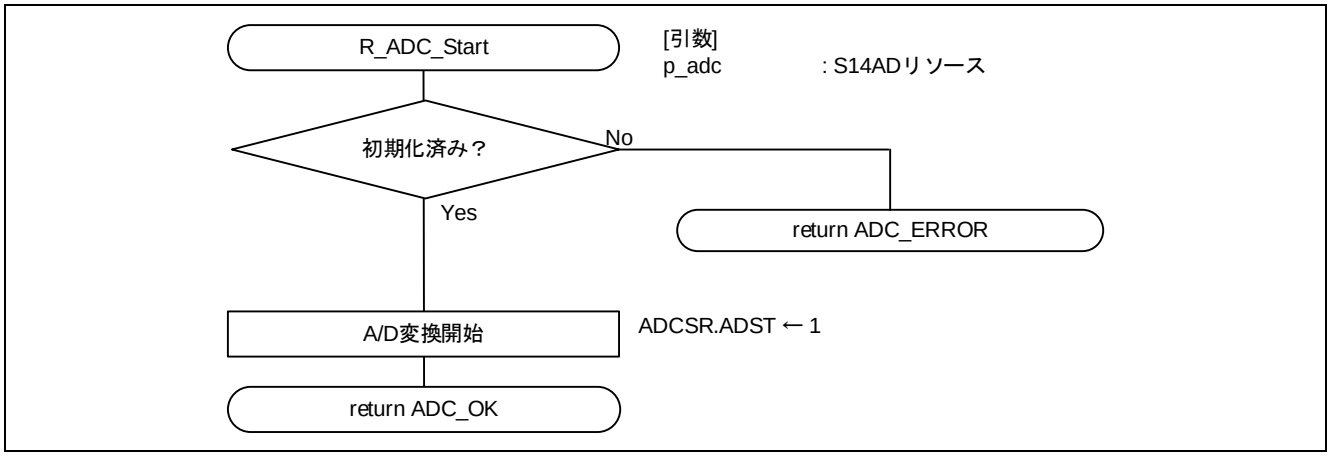


図 4-7 R_ADC_Start 関数処理フロー

4.1.5 R_ADC_Stop 関数

表 4-5 R_ADC_Stop 関数仕様

書式	e_adc_err_t R_ADC_Stop(st_adc_resources_t * const p_adc)
仕様説明	ソフトウェアトリガによる A/D 変換を停止します
引数	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	ADC_OK　A/D 変換停止成功 ADC_ERROR　A/D 変換停止失敗 S14AD の未初期化状態で実行した場合、A/D 変換停止失敗となります

備考	インスタンスからのアクセス時は S14AD リソースの指定は不要です。
	
	[インスタンスからの関数呼び出し例]
	// S14AD driver instance
	extern DRIVER_S14AD Driver_S14AD;
	DRIVER_S14AD *adcDev = &Driver_S14AD;
	
	main()
	{
	　adcDev->Stop();
	}

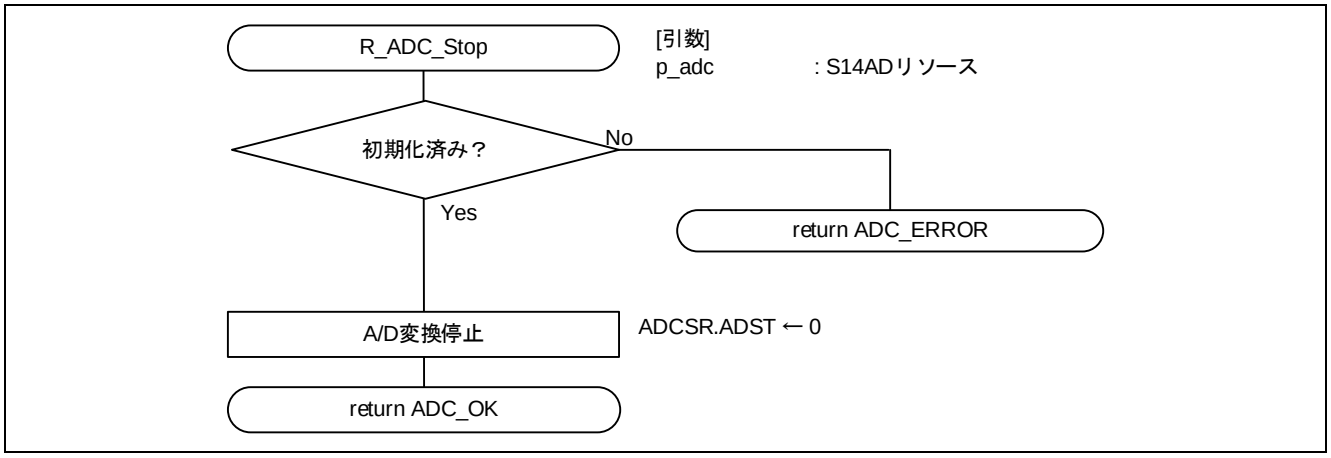


図 4-8 R_ADC_Stop 関数処理フロー

4.1.6 R_ADC_Control 関数

表 4-6 R_ADC_Control 関数仕様

書式	e_adc_err_t R_ADC_Control(e_adc_cmd_t const cmd, void const * const p_args, st_adc_resources_t const * const p_adc)
仕様説明	S14AD の各機能を設定します
引数	e_adc_cmd_t const cmd：設定を行う機能の制御コマンドを指定します
	void const * const p_args：制御コマンドに応じた引数を指定します
	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	ADC_OK　機能設定成功
	ADC_ERROR　機能設定失敗 以下のいずれかの状態を検出すると機能設定失敗となります ・Open 関数未実行状態で実行した場合 ・各機能設定関数の戻り値が ADC_ERROR の場合
	ADC_ERROR_BUSY　ビジー状態による機能設定失敗 A/D 変換実行中に機能設定を行えない制御コマンドを指定した場合、ビジー状態による機能設定失敗となります
	ADC_ERROR_SYSTEM_SETTING　システム設定エラー システム設定が不正な場合、システム設定エラーとなります
	ADC_ERROR_MODE　モードエラー モード設定が不正な場合、モードエラーとなります
	ADC_ERROR_PARAMETER　パラメータエラー 引数が不正な場合、パラメータエラーとなります

備考	インスタンスからのアクセス時は S14AD リソースの指定は不要です。
	
	[インスタンスからの関数呼び出し例]
	// S14AD driver instance
	extern DRIVER_S14AD Driver_S14AD;
	DRIVER_S14AD *adcDev = &Driver_S14AD;
	
	main()
	{
	　　e_adc_int_method_t adc_int = ADC_INT_ENABLE;
	　　adcDev->Control(AD_CMD_SET_ADI_INT, &adc_int);
	}

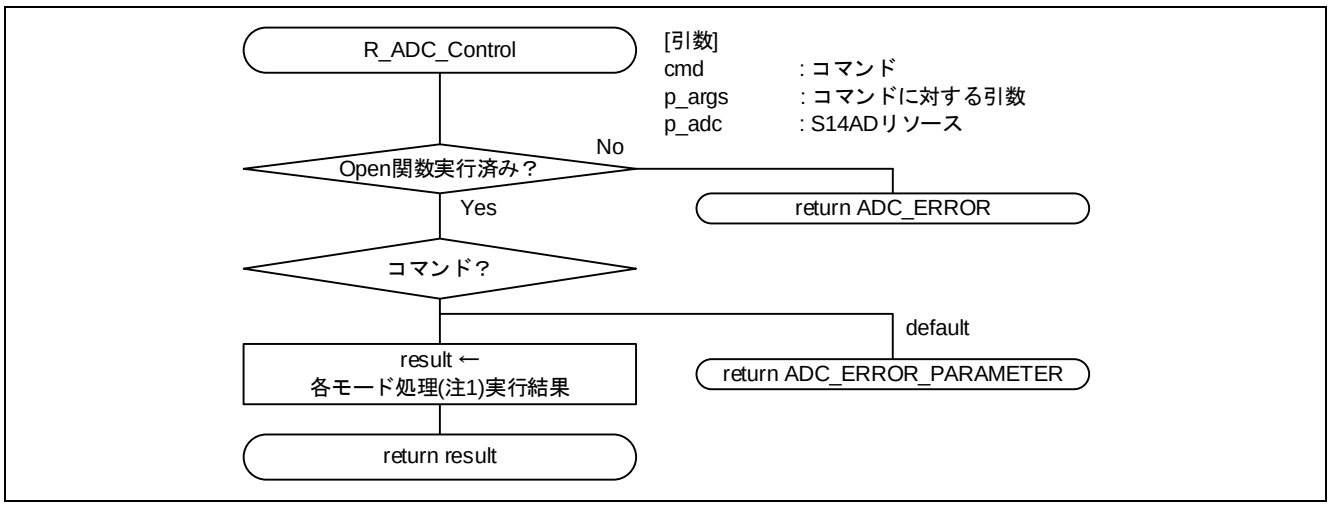


図 4-9 R_ADC_Control 関数処理フロー

注1. result には、各コマンド用関数の戻り値が格納されます。各コマンドにおける呼び出し関数および戻り値一覧を表 4-7、表 4-8 に示します。

表 4-7 Control コマンド別呼び出し関数および戻り値一覧(1/2)

コマンド（cmd）	実行関数	戻り値
AD_CMD_SET_ADD_MODE	adc_cmd_add_mode	ADC_ERROR_PARAMETER
		ADC_ERROR_MODE
		ADC_ERROR_BUSY
		ADC_OK
AD_CMD_SET_DBLTRG	adc_cmd_dbltrg	ADC_ERROR_PARAMETER
		ADC_ERROR_MODE
		ADC_ERROR_BUSY
		ADC_OK
AD_CMD_SET_DIAG	adc_cmd_diag	ADC_ERROR_PARAMETER
		ADC_ERROR_MODE
		ADC_ERROR_BUSY
		ADC_OK
AD_CMD_SET_ADI_INT	adc_cmd_set_adi	ADC_ERROR_PARAMETER
		ADC_ERROR
		ADC_ERROR_SYSTEM_SETTING
		ADC_OK
AD_CMD_SET_GROUPB_INT	adc_cmd_set_gbadi	ADC_ERROR_PARAMETER
		ADC_ERROR
		ADC_ERROR_SYSTEM_SETTING
		ADC_OK
AD_CMD_SET_GROUPC_INT	adc_cmd_set_gcadi	ADC_ERROR_PARAMETER
		ADC_ERROR
		ADC_ERROR_SYSTEM_SETTING
		ADC_OK
AD_CMD_SET_WCMPM_INT	adc_cmd_set_wcmpm	ADC_ERROR_PARAMETER
		ADC_ERROR
		ADC_ERROR_SYSTEM_SETTING
		ADC_OK
AD_CMD_SET_WCMPUM_INT	adc_cmd_set_wcmpum	ADC_ERROR_PARAMETER
		ADC_ERROR
		ADC_ERROR_SYSTEM_SETTING
		ADC_OK
AD_CMD_SET_AUTO_CLEAR	adc_cmd_auto_clear	ADC_ERROR_PARAMETER
		ADC_OK
AD_CMD_SET_SAMPLING_AN000	adc_cmd_sampling	ADC_ERROR_PARAMETER
		ADC_OK
AD_CMD_SET_SAMPLING_AN001	adc_cmd_sampling	ADC_ERROR_PARAMETER
		ADC_OK
AD_CMD_SET_SAMPLING_AN002	adc_cmd_sampling	ADC_ERROR_PARAMETER
		ADC_OK
AD_CMD_SET_SAMPLING_AN003	adc_cmd_sampling	ADC_ERROR_PARAMETER
		ADC_OK
AD_CMD_SET_SAMPLING_AN004	adc_cmd_sampling	ADC_ERROR_PARAMETER
		ADC_OK
AD_CMD_SET_SAMPLING_AN005	adc_cmd_sampling	ADC_ERROR_PARAMETER
		ADC_OK
AD_CMD_SET_SAMPLING_AN006	adc_cmd_sampling	ADC_ERROR_PARAMETER
		ADC_OK
AD_CMD_SET_SAMPLING_AN007(注 1)	adc_cmd_sampling	ADC_ERROR_PARAMETER
		ADC_OK
AD_SST_AN016_AN017_AN020_TO_AN028(注 2)	adc_cmd_sampling	ADC_ERROR_PARAMETER
		ADC_OK
AD_CMD_SET_SAMPLING_AN016_AN017_ AN020_AN021_VSC_VCC(注 1)	adc_cmd_sampling	ADC_ERROR_PARAMETER
		ADC_OK
AD_CMD_SET_SAMPLING_TEMP	adc_cmd_sampling	ADC_ERROR_PARAMETER
		ADC_OK

注1. 256KB グループのみ

注2. 1500KB グループのみ

表 4-8 Control コマンド別呼び出し関数および戻り値一覧(2/2)

コマンド（cmd）	実行関数	戻り値
AD_CMD_SET_ADNDIS	adc_cmd_charge	ADC_ERROR_PARAMETER
		ADC_ERROR_MODE
		ADC_ERROR_BUSY
		ADC_OK
AD_CMD_SET_GROUP_PRIORITY	adc_cmd_group_priority	ADC_ERROR_PARAMETER
		ADC_ERROR_MODE
		ADC_ERROR_BUSY
		ADC_OK
AD_CMD_SET_WINDOWA	adc_cmd_set_windowa	ADC_ERROR_PARAMETER
		ADC_ERROR_MODE
		ADC_ERROR_BUSY
		ADC_ERROR_SYSTEM_SETTING
		ADC_OK
AD_CMD_SET_WINDOWB	adc_cmd_set_windowb	ADC_ERROR_PARAMETER
		ADC_ERROR_MODE
		ADC_ERROR_BUSY
		ADC_ERROR_SYSTEM_SETTING
		ADC_OK
AD_CMD_GET_CMP_RESULT	adc_cmd_get_cmp_result	ADC_OK
AD_CMD_GET_AD_STATE	adc_cmd_get_state	ADC_OK
AD_CMD_USE_VREFL0	adc_cmd_set_vrefl0	ADC_ERROR_BUSY
		ADC_OK
AD_CMD_USE_VREFH0	adc_cmd_set_vrefh0	ADC_ERROR_MODE
		ADC_ERROR_BUSY
		ADC_OK
AD_CMD_SCLK_ENABLE	adc_cmd_set_sclk	ADC_ERROR
		ADC_ERROR_BUSY
		ADC_OK
AD_CMD_CALIBRATION	adc_cmd_calibration	ADC_ERROR
		ADC_ERROR_BUSY
		ADC_OK
AD_CMD_SET_ELC	adc_cmd_set_elc	ADC_ERROR_PARAMETER
		ADC_OK
AD_CMD_STOP_TRIG	adc_cmd_stop_trigger	ADC_ERROR
		ADC_OK
AD_CMD_AUTO_READ_NORMAL	adc_cmd_auto_read	ADC_ERROR_PARAMETER
		ADC_ERROR
		ADC_ERROR_SYSTEM_SETTING
		ADC_OK
AD_CMD_AUTO_READ_BLOCK	adc_cmd_auto_read	ADC_ERROR_PARAMETER
		ADC_ERROR
		ADC_ERROR_SYSTEM_SETTING
		ADC_OK
AD_CMD_AUTO_READ_COMPARE	adc_cmd_auto_read	ADC_ERROR_PARAMETER
		ADC_ERROR
		ADC_ERROR_SYSTEM_SETTING
		ADC_OK
AD_CMD_AUTO_READ_STOP	adc_cmd_auto_read_stop	ADC_ERROR_PARAMETER
		ADC_ERROR_SYSTEM_SETTING
		ADC_OK
AD_CMD_AUTO_READ_RESTART	adc_cmd_auto_read_restart	ADC_ERROR_PARAMETER
		ADC_ERROR_SYSTEM_SETTING
		ADC_OK

4.1.7 R_ADC_Read 関数

表 4-9 R_ADC_Read 関数仕様

書式	e_adc_err_t R_ADC_Read(e_adc_ssel_ch_t sel_ch, uint16_t * const p_data, st_adc_resources_t const * const p_adc)
仕様説明	指定したチャネルの A/D 変換結果を読み出します
引数	e_adc_ssel_ch_t sel_ch：読み取りチャネル 以下のいずれかを指定します ・ADC_SSEL_ANn：指定チャネルの A/D データレジスタ(ADDRn)を読み出す (1500KB グループ：n = 00～06、16、17、20～28 256KB グループ：n = 00～07、16、17、20～21) ・ADC_SSEL_TEMP：温度センサデータレジスタ(ADTSDR)を読み出す ・ADC_SSEL_VSC_VCC(注)：VSC_VCC 端子電圧出力データレジスタ (ADVSCDR)を読み出す ・ADC_SSEL_DBL：A/D データ 2 重化レジスタ(ADDBLDR)を読み出す ・ADC_SSEL_DIAG：A/D 自己診断データレジスタ(ADRD)を読み出す
	uint16_t * const p_data：読み取りデータ格納先アドレス 読み取りデータを格納する変数のアドレスを指定します
	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	ADC_OK A/D 変換結果読み出し成功
	ADC_ERROR_PARAMETER パラメータエラー 以下のいずれかの条件を検出するとパラメータエラーとなります ・読み取りデータ格納先アドレスが NULL の場合 ・存在しない A/D 変換チャネルを指定した場合

備考	インスタンスからのアクセス時は S14AD リソースの指定は不要です。
	
	[インスタンスからの関数呼び出し例]
	// S14AD driver instance
	extern DRIVER_S14AD Driver_S14AD;
	DRIVER_S14AD *adcDev = &Driver_S14AD;
	uint8_t read_data;
	
	main()
	{
	adcDev->Read(ADC_SSEL_AN00, &read_data);
	}

注 256KB グループのみ

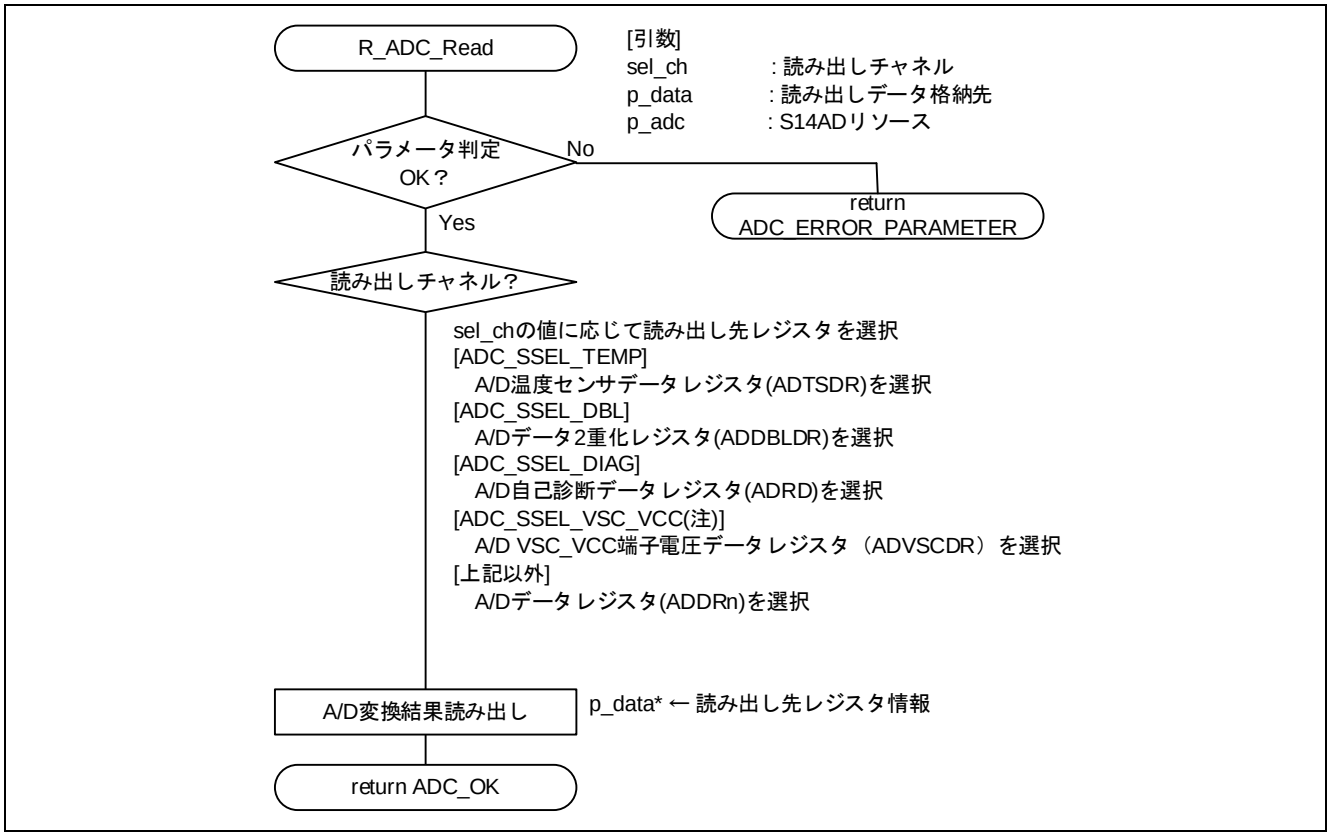


図 4-10 R_ADC_Read 関数処理フロー

注　256KB グループのみ

4.1.8 R_ADC_GetVersion 関数

表 4-10 R_ADC_GetVersion 関数仕様

書式	uint32_t　R_ADC_GetVersion(void)
仕様説明	S14AD ドライバのバージョンを取得します
引数	なし
戻り値	S14AD ドライバのバージョン

備考	インスタンスからのアクセス時は S14AD リソースの指定は不要です。
	
	[インスタンスからの関数呼び出し例]
	// S14AD driver instance
	extern DRIVER_S14AD Driver_S14AD;
	DRIVER_S14AD *adcDev = &Driver_S14AD;
	
	main()
	{
	　　DRIVER_VERSION version;
	　　version = adcDev->GetVersion();
	
	}

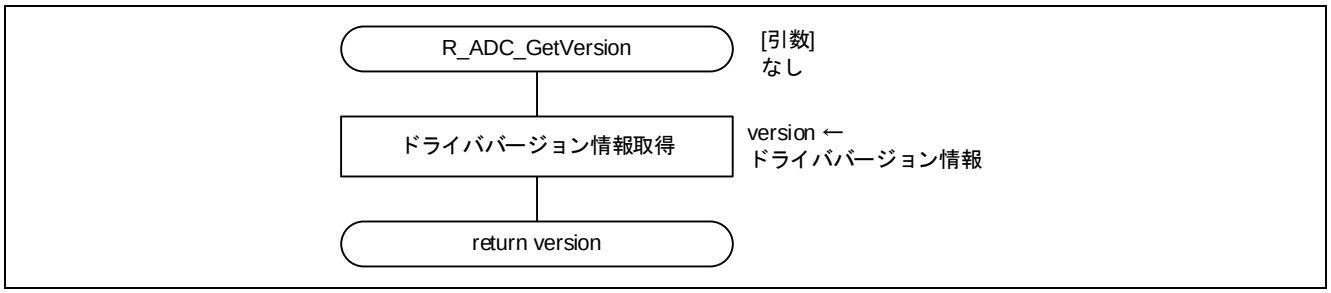


図 4-11 R_ADC_GetVersion 関数処理フロー

4.1.9 adc_cmd_add_mode 関数

表 4-11 adc_cmd_add_mode 関数仕様

書式	static e_adc_err_t adc_cmd_add_mode(st_adc_add_mode_t const * const p_add_m, st_adc_resources_t * const p_adc)
仕様説明	加算/平均モード設定を行います
引数	st_adc_add_mode_t const * const p_add_m：加算/平均モード設定値
	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	ADC_OK　加算/平均モード設定成功
	ADC_ERROR_MODE　モードエラー 以下のいずれかの条件を検出するとモードエラーとなります ・A/D 変換精度が 14 ビット精度設定の時、16 回変換を指定した場合 ・自己診断モードで実行した場合
	ADC_ERROR_PARAMETER　パラメータエラー 以下のいずれかの条件を検出するとパラメータエラーとなります ・加算/平均モード設定に規定外の値を指定した場合 ・加算/平均モード対象チャネルに存在しないチャネルを指定した場合
	ADC_ERROR_BUSY　ビジー状態による設定失敗 A/D 変換実行中に実行した場合、ビジー状態による設定失敗となります
備考	-

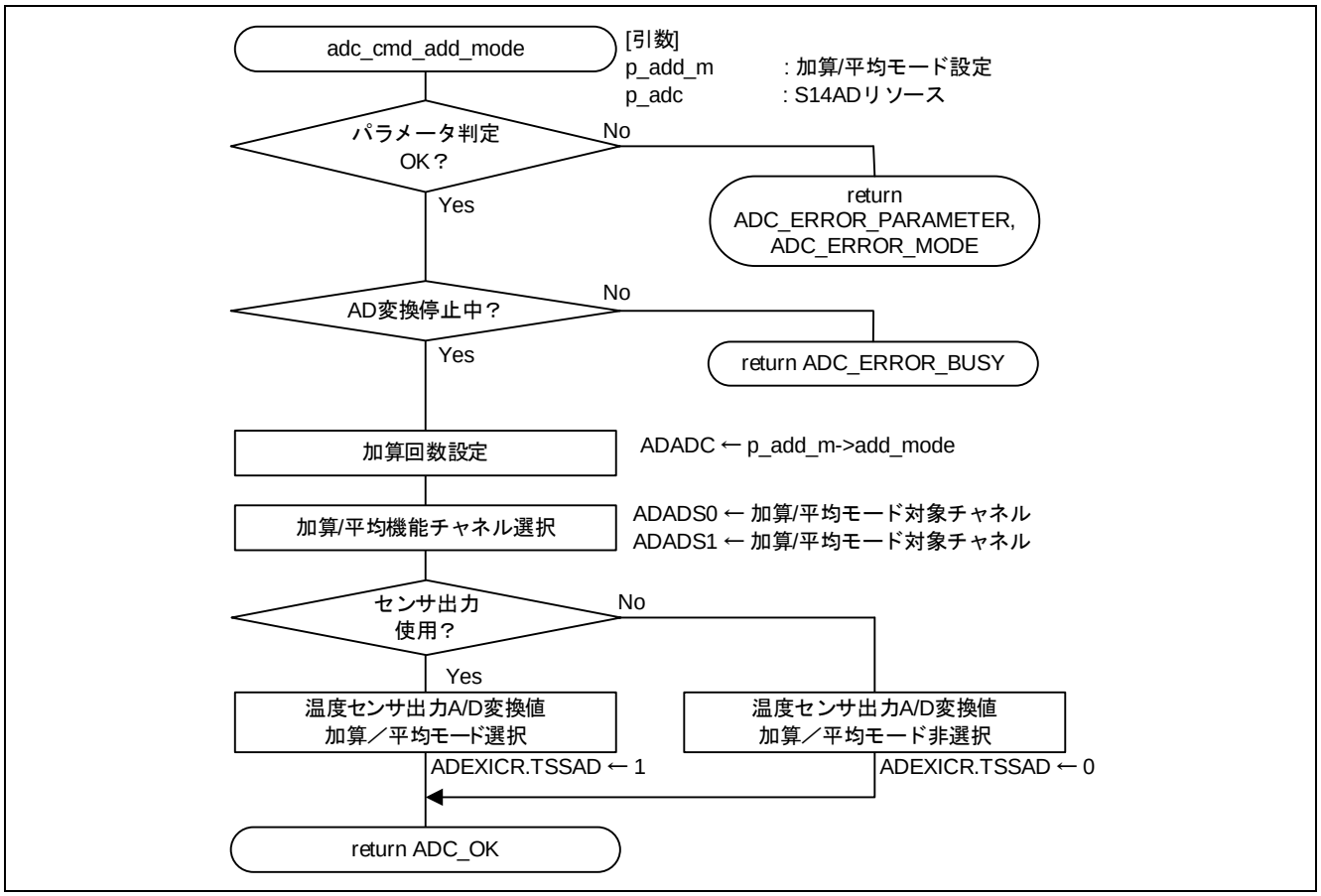


図 4-12 adc_cmd_add_mode 関数処理フロー

4.1.10 adc_cmd_dbltrg 関数

表 4-12 adc_cmd_dbltrg 関数仕様

書式	static e_adc_err_t adc_cmd_dbltrg(int8_t const * const p_dbltrg_ch, st_adc_resources_t * const p_adc)
仕様説明	ダブルトリガモード設定を行います
引数	int8_t const * const p_dbltrg_ch：A/D 変換データ 2 重化チャネルを指定します
	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	ADC_OK　ダブルトリガモード設定成功
	ADC_ERROR_PARAMETER　パラメータエラー A/D 変換データ 2 重化チャネルに存在しないチャネルが指定された場合、パラメータエラーとなります
	ADC_ERROR_MODE　モードエラー 以下のいずれかの条件を検出するとモードエラーとなります ・連続スキャンモードで実行した場合 ・ソフトウェアトリガ設定で実行した場合 ・自己診断モードで実行した場合 ・温度センサ出力を使用時に実行した場合 ・コンペアモードで実行した場合 ・VSC_VCC 端子電圧出力を使用時に実行した場合(注)
	ADC_ERROR_BUSY　ビジー状態による設定失敗 A/D 変換実行中に実行した場合、ビジー状態による設定失敗となります
備考	-

注　256KB グループのみ

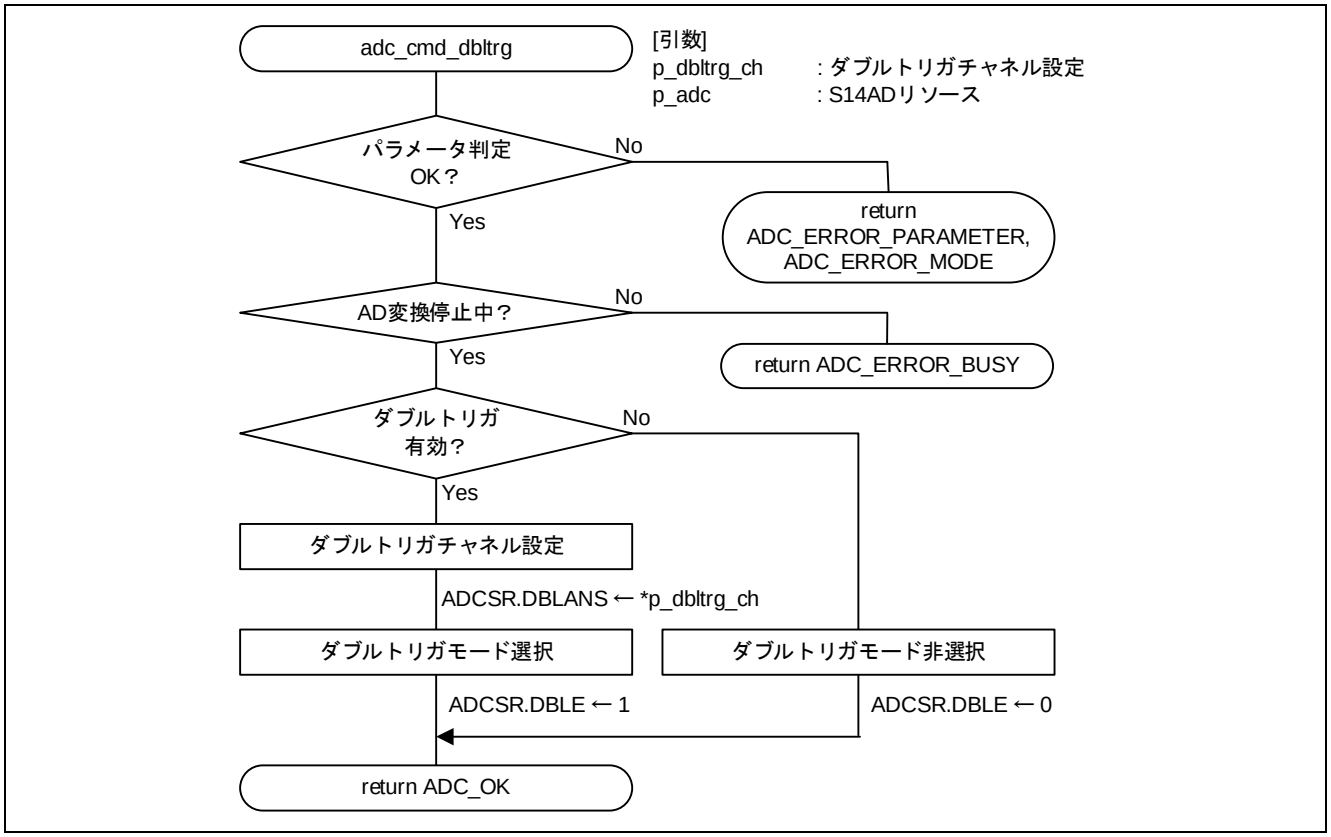


図 4-13 adc_cmd_dbltrg 関数処理フロー

4.1.11 adc_cmd_diag 関数

表 4-13 adc_cmd_diag 関数仕様

書式	static e_adc_err_t adc_cmd_diag(e_adc_diag_t const * const p_diag, st_adc_resources_t * const p_adc)	
仕様説明	自己診断モード設定を行います	
引数	e_adc_diag_t const * const p_diag：自己診断モード設定値 以下のいずれかを設定します ・AD_DIAG_DISABLE：自己診断無効 ・AD_DIAG_0V：0V 電圧 ・AD_DIAG_HARF：基準電圧電源(VREFH0)×1/2 電圧 ・AD_DIAG_BASE：基準電圧電源(VREFH0)電圧 ・AD_DIAG_ROTATE：自己診断電圧ローテーションモード	
	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します	
戻り値	ADC_OK	自己診断モード設定成功
	ADC_ERROR_MODE	モードエラー 以下のいずれかの条件を検出するとモードエラーとなります ・高位電圧基準に VREFH が指定されている場合 ・ダブルトリガモードで実行した場合 ・断線検知アシスト機能が有効の場合 ・コンペア機能が有効の場合
	ADC_ERROR_BUSY	ビジー状態による設定失敗 A/D 変換実行中に実行した場合、ビジー状態による設定失敗となります
	ADC_ERROR_PARAMETER	パラメータエラー 自己診断モード設定に規定外の値を指定した場合、パラメータエラーとなります
備考	-	

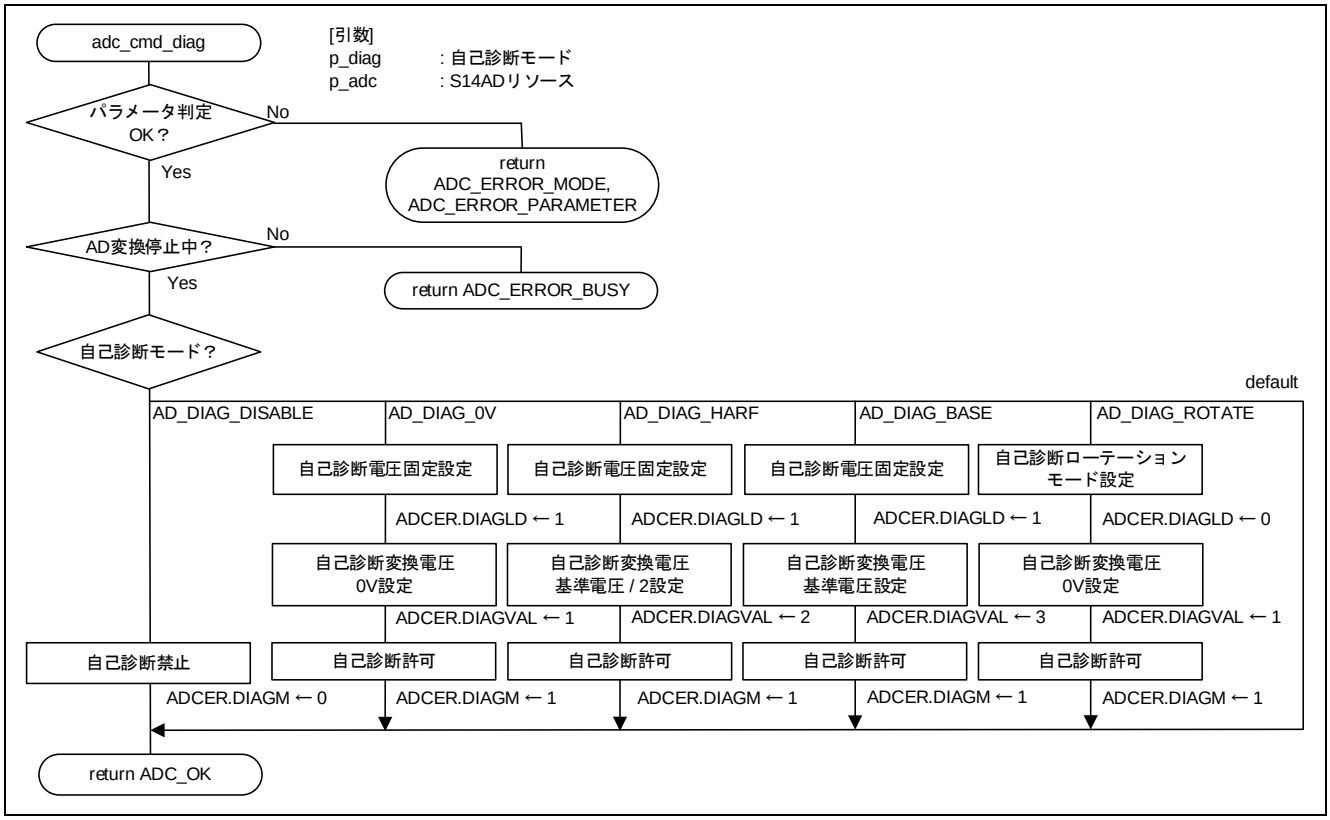


図 4-14 adc_cmd_diag 関数処理フロー

4.1.12 adc_cmd_auto_clear 関数

表 4-14 adc_cmd_auto_clear 関数仕様

書式	static e_adc_err_t adc_cmd_auto_clear(uint8_t const * const p_en, st_adc_resources_t * const p_adc)
仕様説明	A/D データレジスタ自動クリア設定を行います
引数	uint8_t const * const p_en：A/D データレジスタ自動クリア許可/禁止を指定します ADC_ENABLE：許可 ADC_DISABLE：禁止
	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	ADC_ERROR_PARAMETER　　パラメータエラー 以下のいずれかの条件を検出するとパラメータエラーとなります ・引数が NULL の場合 ・引数に規定外の値が指定された場合
	ADC_OK　　A/D データレジスタ自動クリア設定成功
備考	-

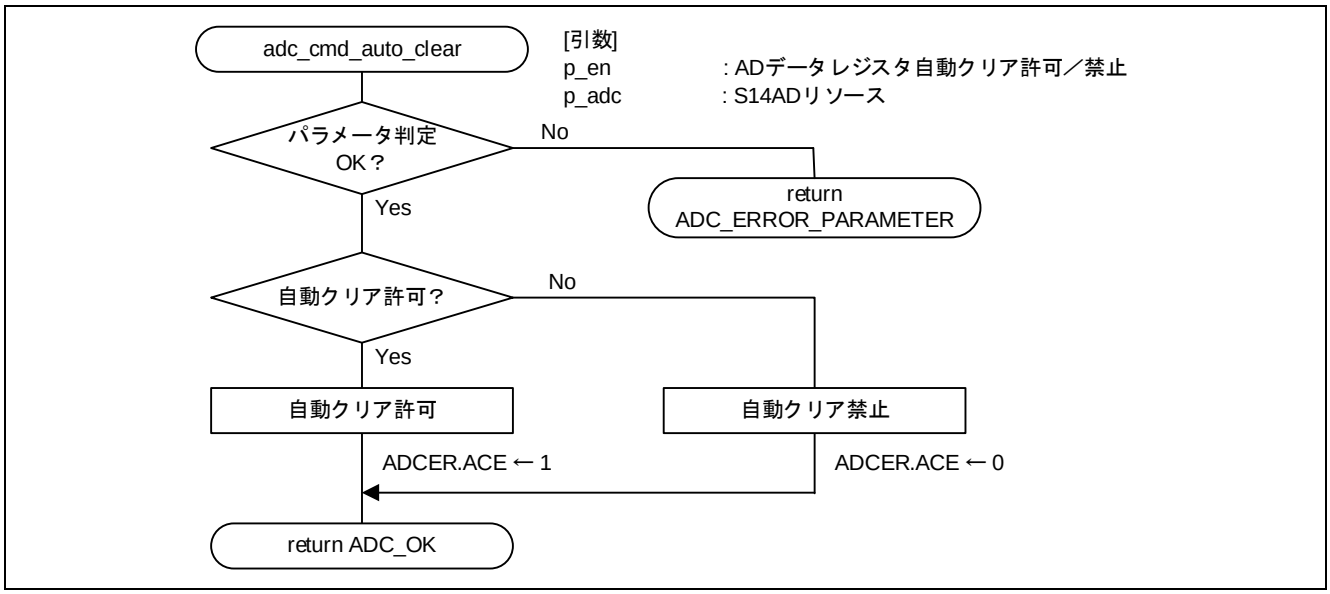


図 4-15 adc_cmd_auto_clear 関数処理フロー

4.1.13 adc_cmd_sampling 関数

表 4-15 adc_cmd_sampling 関数仕様

書式	static e_adc_err_t adc_cmd_sampling(e_adc_sst_t sst , uint8_t const * const p_sst_val, st_adc_resources_t * const p_adc)
仕様説明	A/D サンプリング時間設定を行います
引数	e_adc_sst_t sst：サンプリング時間設定対象チャネル
	uint8_t const * const p_sst_val：サンプリング時間 2～255 のサンプリング時間を指定します
	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	ADC_ERROR_PARAMETER　　パラメータエラー 以下のいずれかの条件を検出するとパラメータエラーとなります ・引数が NULL の場合 ・引数に規定外の値が指定された場合
	ADC_OK　　A/D サンプリング時間設定成功
備考	-

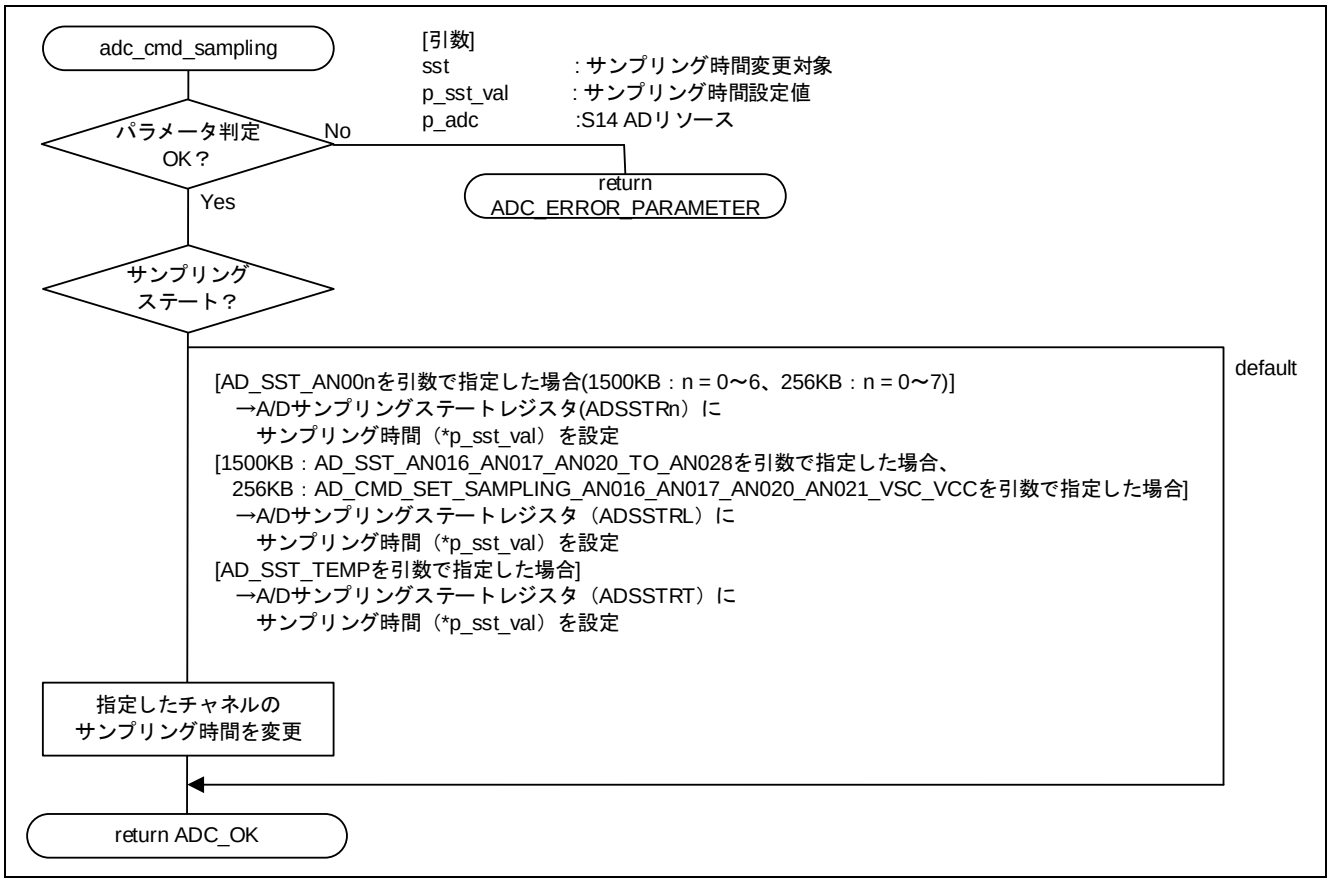


図 4-16 adc_cmd_sampling 関数処理フロー

4.1.14 adc_cmd_charge 関数

表 4-16 adc_cmd_charge 関数仕様

書式	static e_adc_err_t adc_cmd_charge(uint8_t const * const p_charge, st_adc_resources_t * const p_adc)
仕様説明	断線検出アシスト機能設定を行います
引数	uint8_t const * const p_charge：断線検出アシスト設定値
	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	ADC_OK　　断線検出アシスト機能設定成功
	ADC_ERROR_PARAMETER　　パラメータエラー 以下のいずれかの条件を検出するとパラメータエラーとなります ・引数が NULL の場合 ・引数に規定外の値が指定された場合
	ADC_ERROR_MODE　　モードエラー 以下のいずれかの条件を検出するとモードエラーとなります ・温度センサ出力を使用している場合 ・自己診断モードで実行した場合
	ADC_ERROR_BUSY　　ビジー状態による設定失敗 A/D 変換実行中に実行した場合、ビジー状態による設定失敗となります
備考	-

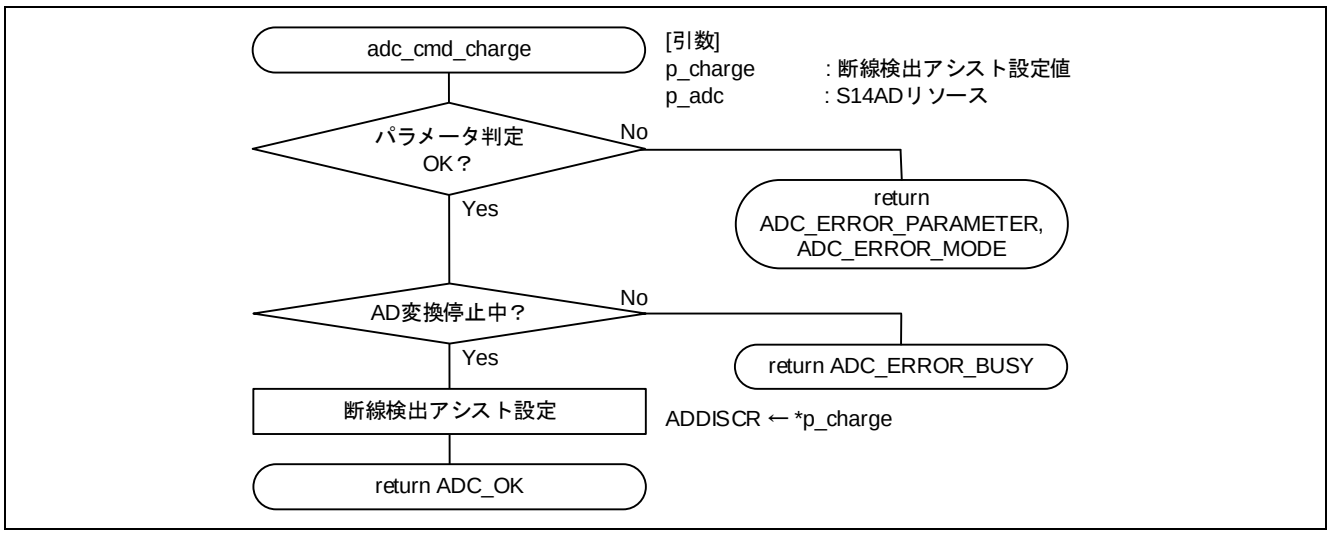


図 4-17 adc_cmd_charge 関数処理フロー

4.1.15 adc_cmd_group_priority 関数

表 4-17 adc_cmd_group_priority 関数仕様

書式	static e_adc_err_t adc_cmd_group_priority(e_adc_gsp_t const * const p_gsp, st_adc_resources_t * const p_adc)
仕様説明	A/D グループスキャン優先動作設定を行います
引数	e_adc_gsp_t const * const p_gsp：A/D グループスキャン優先制御設定値 以下のいずれかを設定します ・ADC_GSP_PRIORYTY_OFF：グループ優先動作無効 ・ADC_GSP_WAIT_TRG：低優先グループ再起動無効 ・ADC_GSP_SCAN_BIGIN：低優先グループ再起動有効 ・ADC_GSP_SCAN_RESTART：A/D 変換未終了チャネルから再スキャン ・ADC_GSP_WAIT_TRG_CONT_SCAN ：シングルスキャン連続動作有効かつ低優先グループ再起動無効 ・ADC_GSP_SCAN_BIGIN_CONT_SCAN ：シングルスキャン連続動作有効かつ低優先グループ再起動有効 ・ADC_GSP_SCAN_RESTART_CONT_SCAN ：シングルスキャン連続動作有効かつ A/D 変換未終了チャネルから再スキャン
	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	ADC_OK　A/D グループスキャン優先動作設定成功
	ADC_ERROR_PARAMETER　パラメータエラー 以下のいずれかの条件を検出するとパラメータエラーとなります ・引数が NULL の場合 ・引数に規定外の値が指定された場合
	ADC_ERROR_MODE　モードエラー グループスキャンモード以外で実行した場合、モードエラーとなります
	ADC_ERROR_BUSY　ビジー状態による設定失敗 A/D 変換実行中に実行した場合、ビジー状態による設定失敗となります
備考	-

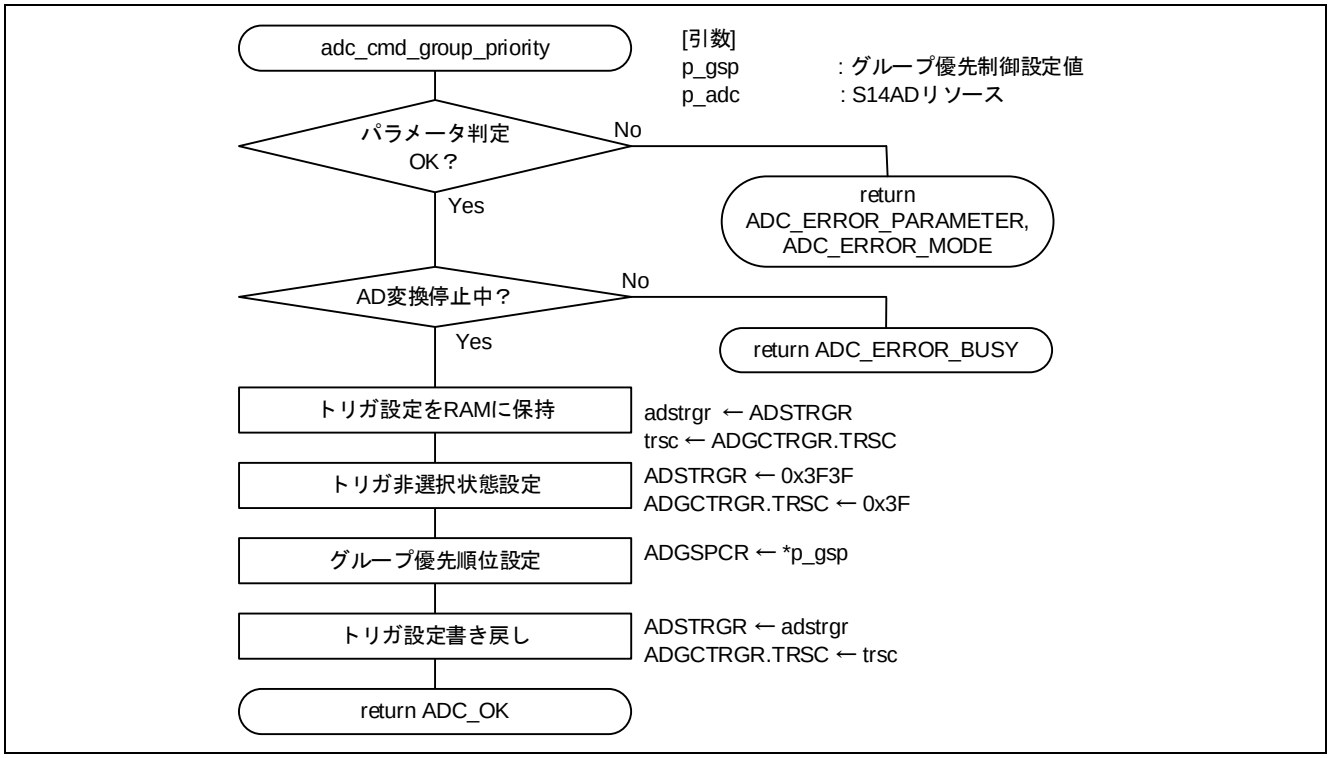


図 4-18 adc_cmd_group_priority 関数処理フロー

4.1.16 adc_cmd_set_windowa 関数

表 4-18 adc_cmd_set_windowa 関数仕様

書式	static e_adc_err_t adc_cmd_set_windowa(st_adc_wina_t const * const p_wina, st_adc_resources_t * const p_adc)
仕様説明	コンペア機能ウィンドウ A の設定を行います
引数	st_adc_wina_t const * const p_wina：コンペア機能ウィンドウ A 設定値
	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	ADC_OK　コンペアウィンドウ A 設定成功
	ADC_ERROR_PARAMETER　パラメータエラー 以下のいずれかの条件を検出するとパラメータエラーとなります ・引数が NULL の場合 ・ADC_CMP_WINDOW 指定時、上位側基準レベルと下位側基準レベルの値が等しい場合 ・ADC_CMP_WINDOW もしくは ADC_CMP_LEVEL 指定時、対象チャネルが指定されていない場合 ・ADC_CMP_WINDOW もしくは ADC_CMP_LEVEL 指定時、対象チャネルが有効になっていない場合 ・引数に規定外の値が指定された場合
	ADC_ERROR_MODE　モードエラー グループスキャンモード以外で実行した場合、モードエラーとなります
	ADC_ERROR_BUSY　ビジー状態による設定失敗 A/D 変換実行中に実行した場合、ビジー状態による設定失敗となります
	ADC_ERROR_SYSTEM_SETTING　システム設定エラー 以下のいずれかの条件を検出するとシステム設定エラーとなります ・r_system_cfg.h で CMPAI 割り込みが未使用定義（SYSTEM_IRQ_EVENT_NUMBER_NOT_USED）になっている場合 ・r_adc_cfg.h で S14AD_CMPAI_PRIORITY の設定が定義範囲を超えている場合
備考	-

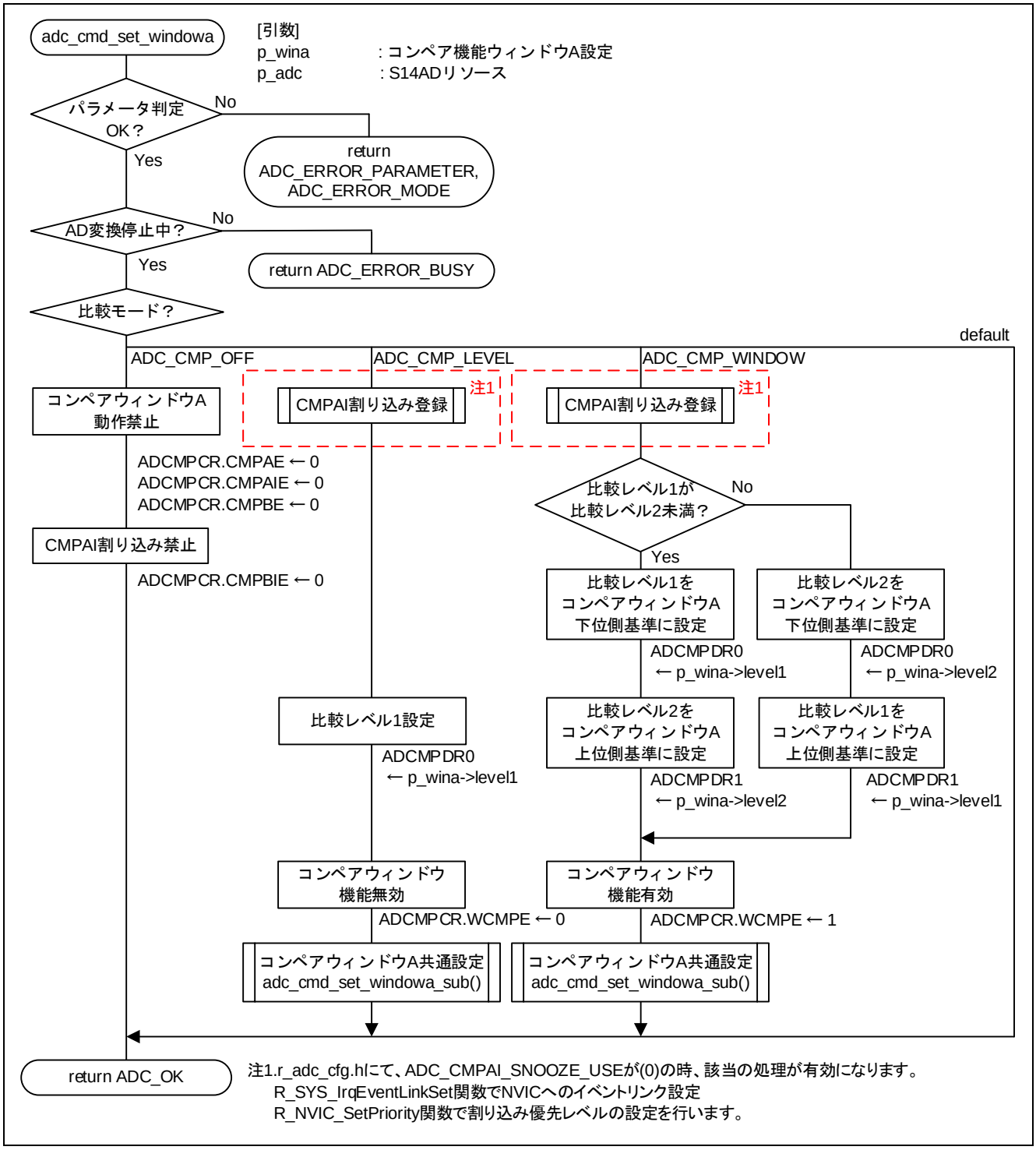


図 4-19 adc_cmd_set_windowa 関数処理フロー

4.1.17 adc_cmd_set_windowa_sub 関数

表 4-19 adc_cmd_set_windowa_sub 関数仕様

書式	static void adc_cmd_set_windowa_sub(st_adc_wina_t const * const p_wina, st_adc_resources_t * const p_adc)
仕様説明	コンペア機能ウィンドウ A の設定を行います
引数	st_adc_wina_t const * const p_wina：コンペア機能ウィンドウ A 設定値
	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	-
備考	-

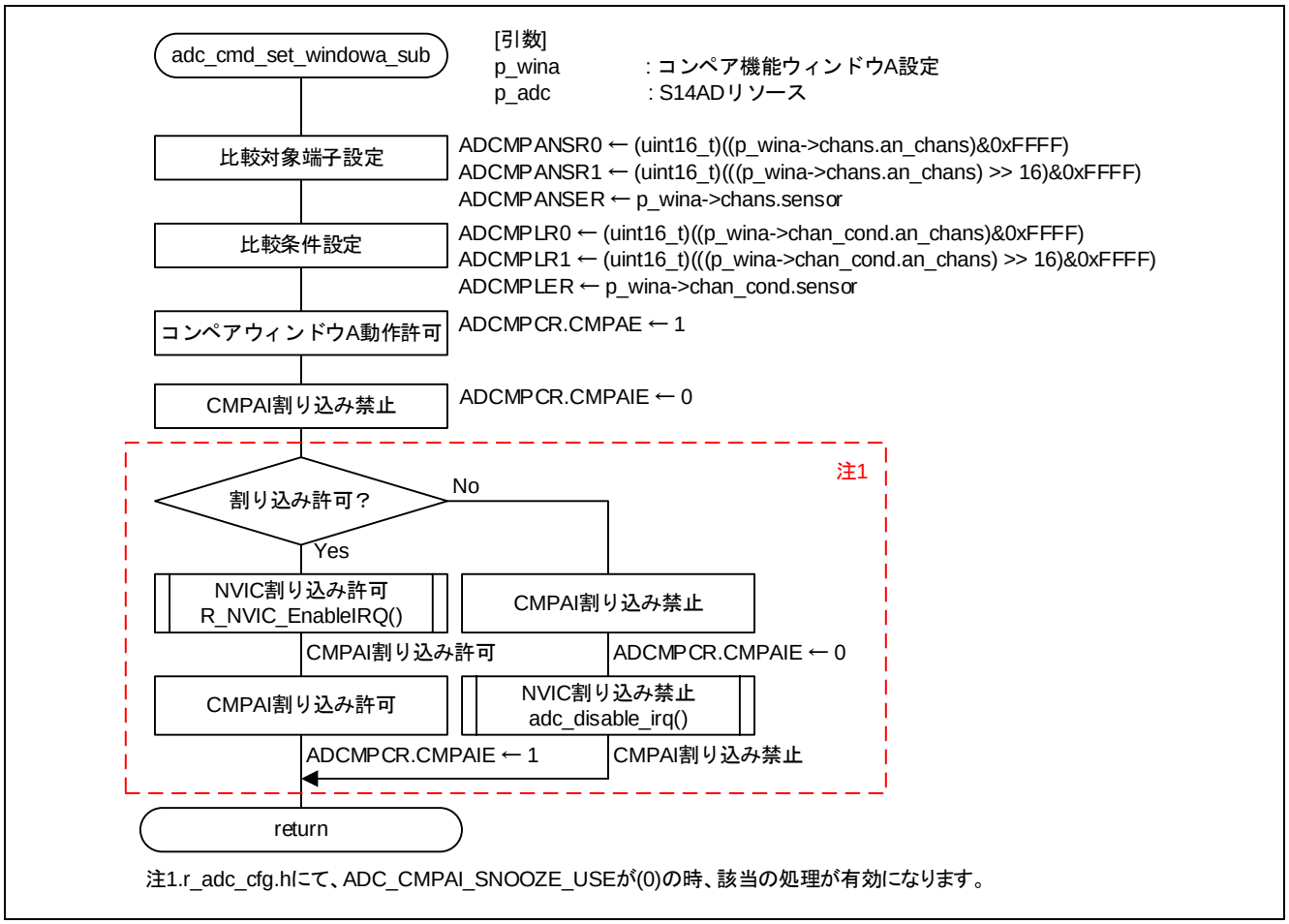


図 4-20 adc_cmd_set_windowa_sub 関数処理フロー

4.1.18 adc_cmd_set_windowb 関数

表 4-20 adc_cmd_set_windowb 関数仕様

書式	static e_adc_err_t adc_cmd_set_windowb(st_adc_winb_t const * const p_winb, st_adc_resources_t * const p_adc)
仕様説明	コンペア機能ウィンドウ B の設定を行います
引数	st_adc_winb_t const * const p_winb：コンペア機能ウィンドウ B 設定値
	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	ADC_OK　コンペア機能ウィンドウ B 設定成功
	ADC_ERROR_PARAMETER　パラメータエラー 以下のいずれかの条件を検出するとパラメータエラーとなります ・引数が NULL の場合 ・引数に規定外の値が指定された場合
	ADC_ERROR_MODE　モードエラー 以下のいずれかの条件を検出するとモードエラーとなります ・指定したチャネルがウィンドウ A で使用されている場合 ・ウィンドウ A が無効の場合 ・シングルスキャンモード以外で実行した場合 ・ウィンドウ A で温度センサ出力を使用している場合
	ADC_ERROR_BUSY　ビジー状態による設定失敗
	A/D 変換実行中に実行した場合、ビジー状態による設定失敗となります
	ADC_ERROR_SYSTEM_SETTING　システム設定エラー 以下のいずれかの条件を検出するとシステム設定エラーとなります ・r_system_cfg.h で CMPBI 割り込みが未使用定義 （SYSTEM_IRQ_EVENT_NUMBER_NOT_USED）になっている場合 ・r_adc_cfg.h で S14AD_CMPBI_PRIORITY の設定が定義範囲を超えている場合
備考	-

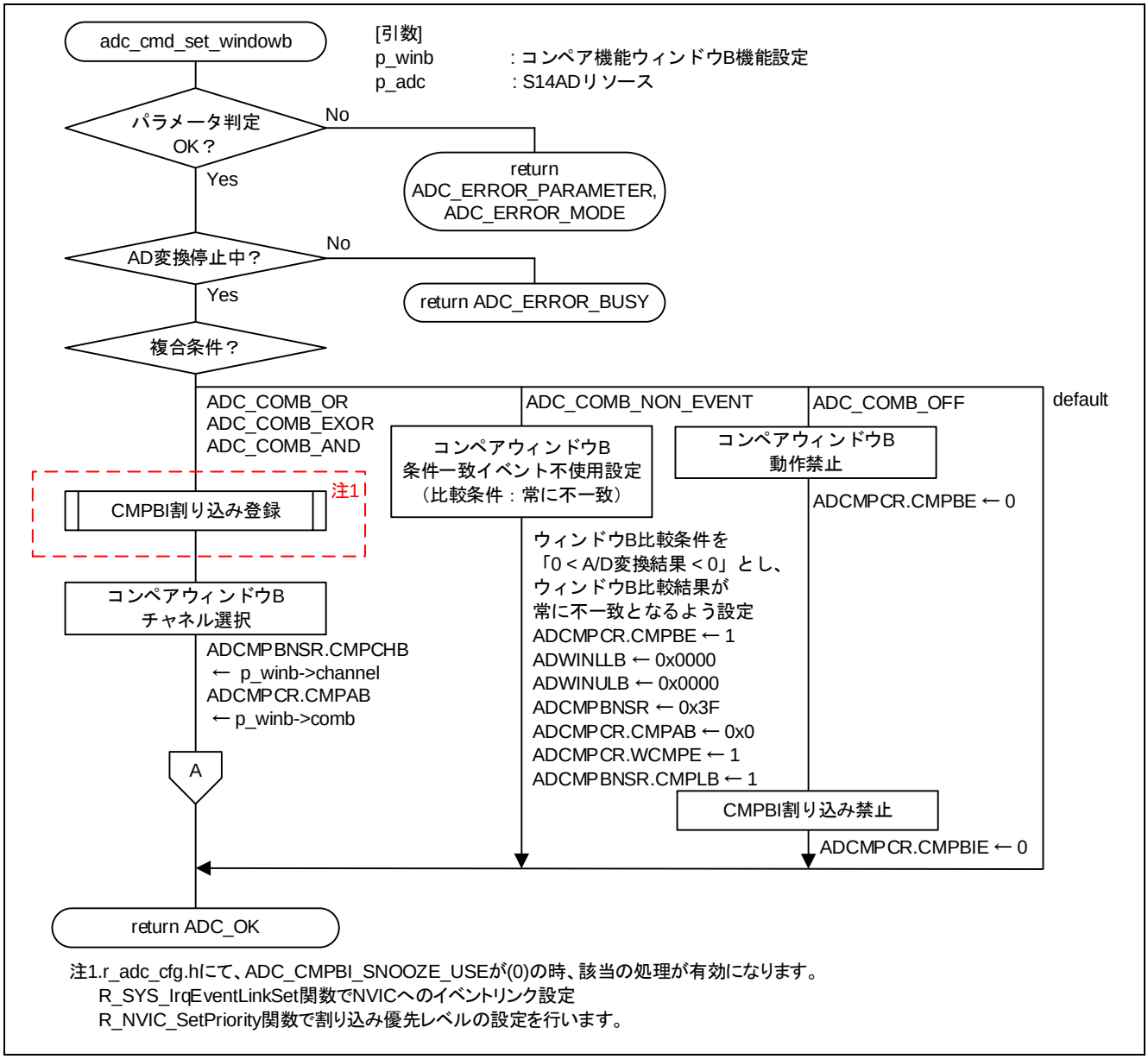


図 4-21 adc_cmd_set_windowb 関数処理フロー(1/2)

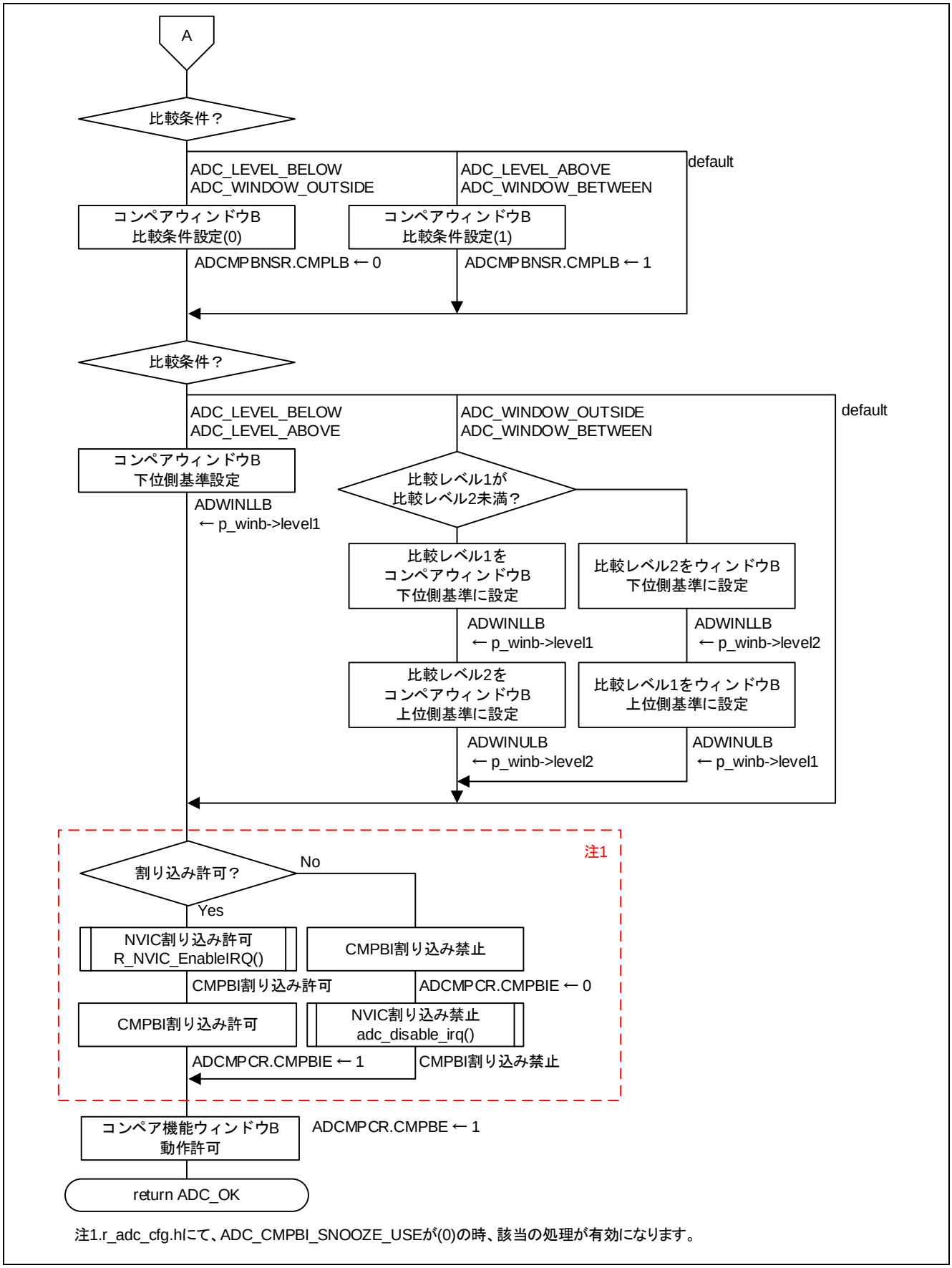


図 4-22 adc_cmd_set_windowb 関数処理フロー(2/2)

4.1.19 adc_cmd_get_cmp_result 関数

表 4-21 adc_cmd_get_cmp_result 関数仕様

書式	static e_adc_err_t adc_cmd_get_cmp_result(st_adc_cmp_result_t * const p_cmp_result, st_adc_resources_t * const p_adc)
仕様説明	コンペア機能比較結果を取得します
引数	st_adc_cmp_result_t * const p_cmp_result：コンペア機能比較結果格納先アドレス
	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	ADC_OK　　コンペア機能比較結果取得成功
備考	-

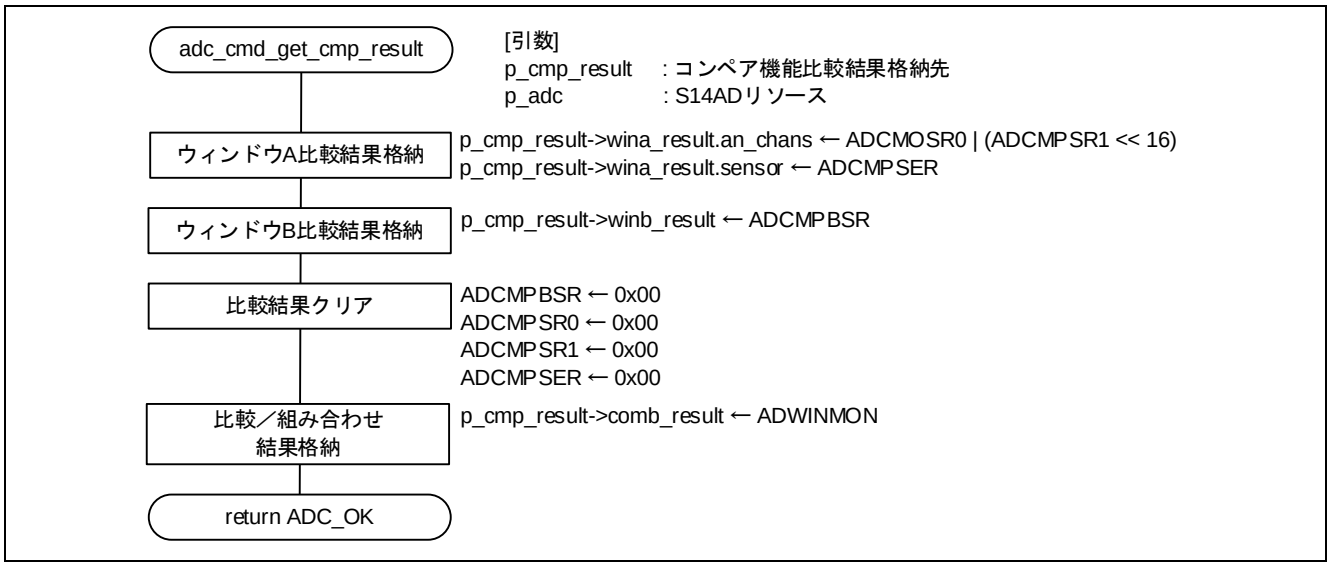


図 4-23 adc_cmd_get_cmp_result 関数処理フロー

4.1.20 adc_cmd_get_state 関数

表 4-22 adc_cmd_get_state 関数仕様

書式	static e_adc_err_t adc_cmd_get_state(st_adc_status_info_t * const p_state, st_adc_resources_t * const p_adc)
仕様説明	S14AD の状態を取得します
引数	st_adc_status_info_t * const p_state：S14AD 状態情報格納先アドレス
	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	ADC_OK　コンペア機能比較結果取得成功
備考	-

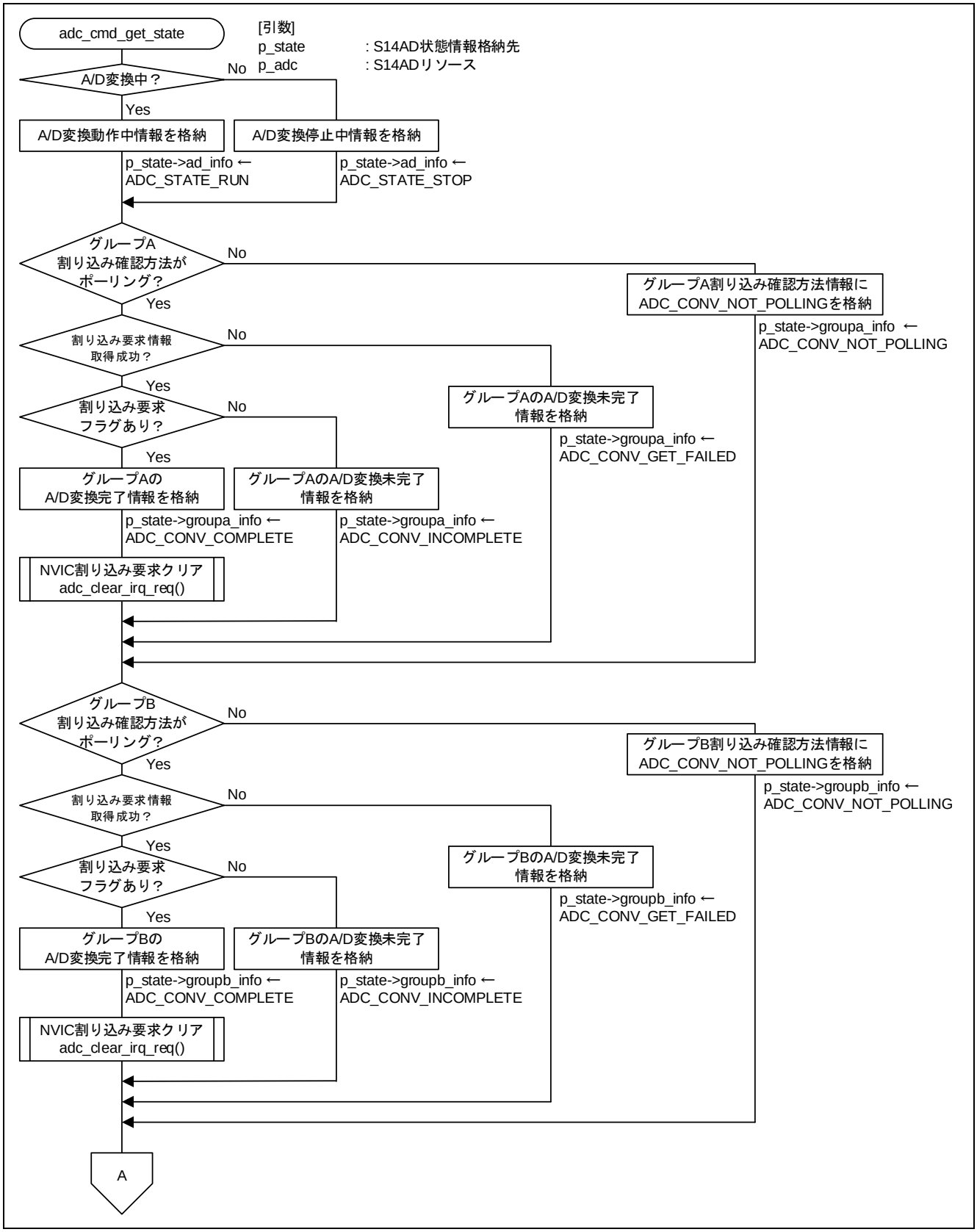


図 4-24 adc_cmd_get_state 関数処理フロー(1/3)

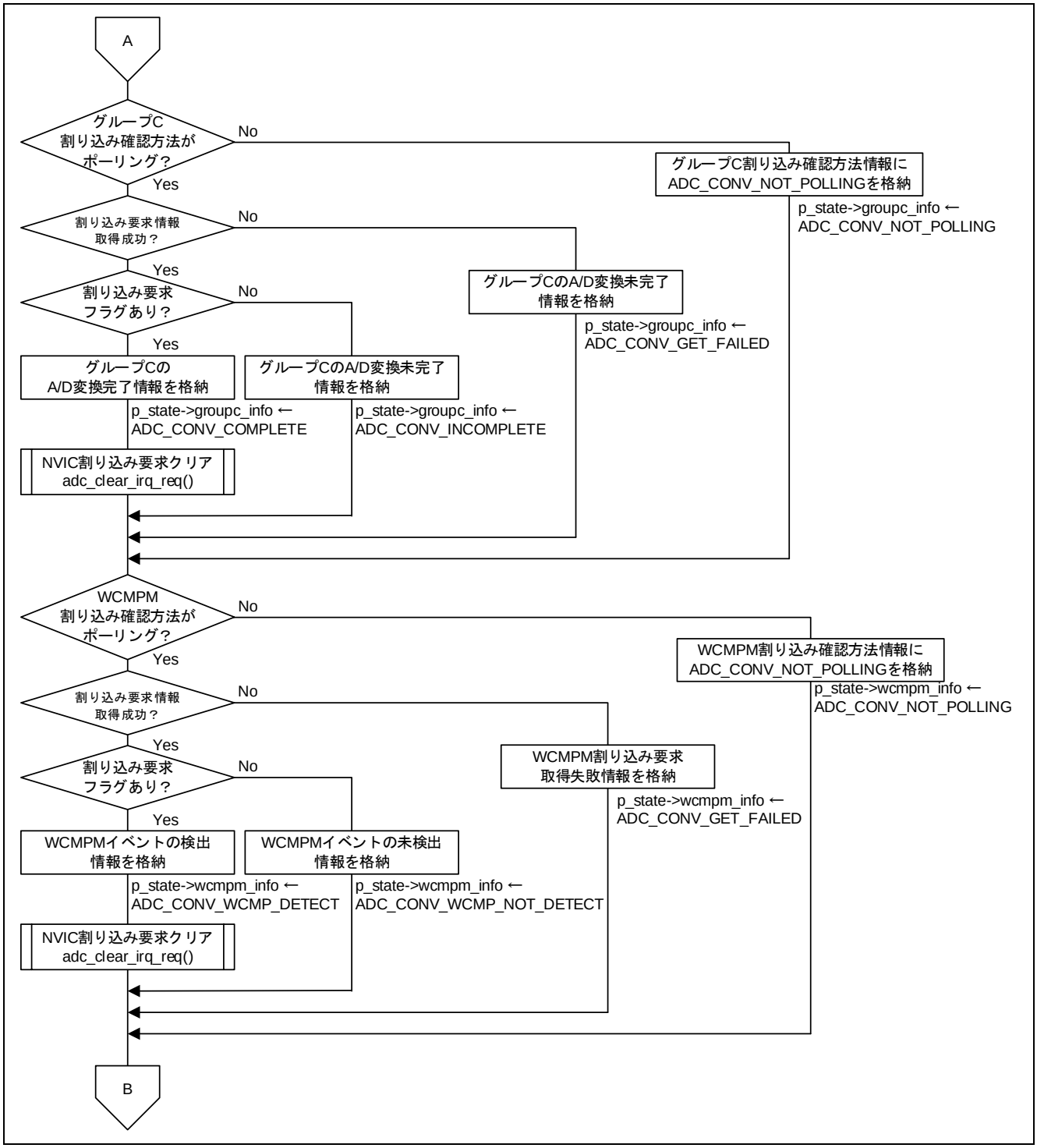


図 4-25 adc_cmd_get_state 関数処理フロー(2/3)

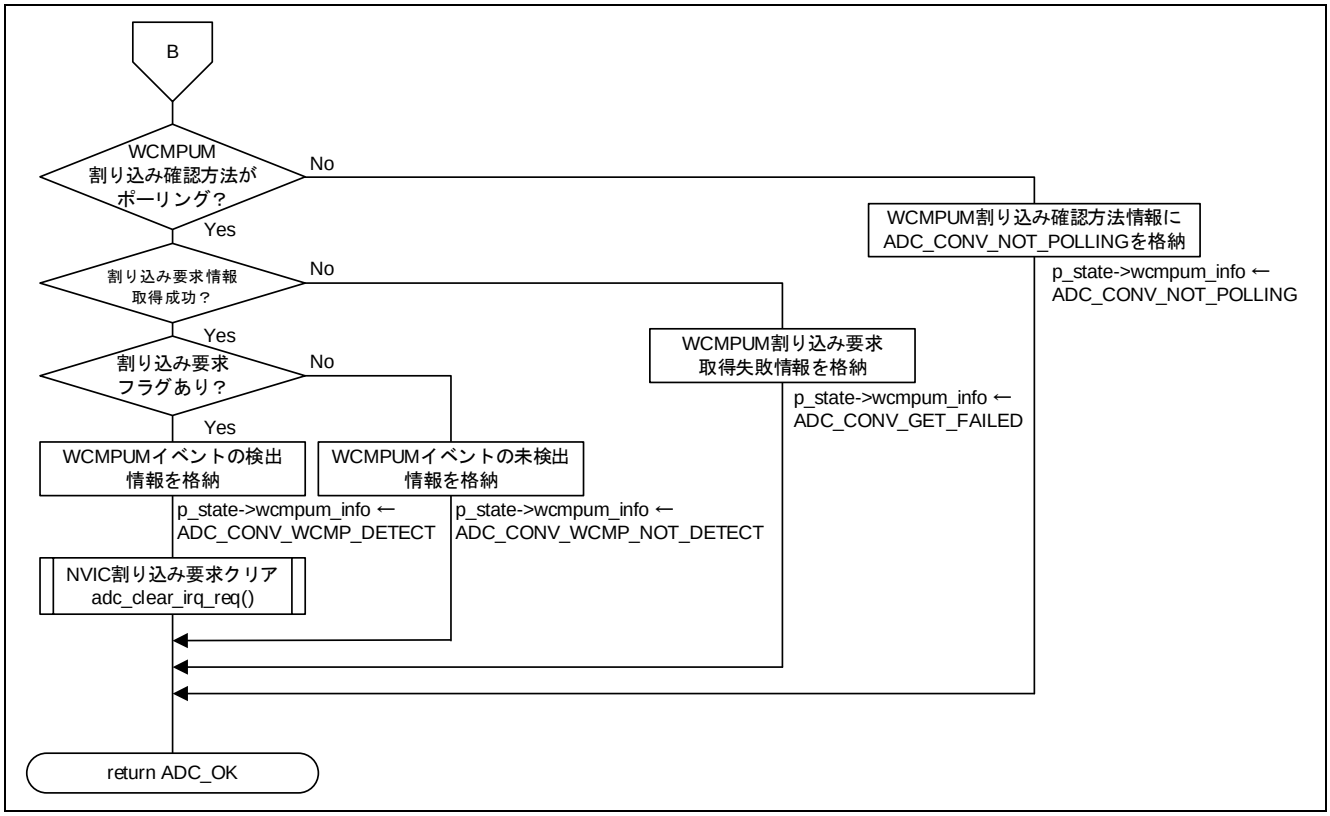


図 4-26 adc_cmd_get_state 関数処理フロー(3/3)

4.1.21 adc_cmd_set_vrefl0 関数

表 4-23 adc_cmd_set_vrefl0 関数仕様

書式	static e_adc_err_t adc_cmd_set_vrefl0(st_adc_resources_t * const p_adc)
仕様説明	コンペア機能比較結果を取得します
引数	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	ADC_OK　コンペア機能比較結果取得成功
	ADC_ERROR_BUSY　ビジー状態による設定失敗 A/D 変換実行中に実行した場合、ビジー状態による設定失敗となります
備考	-

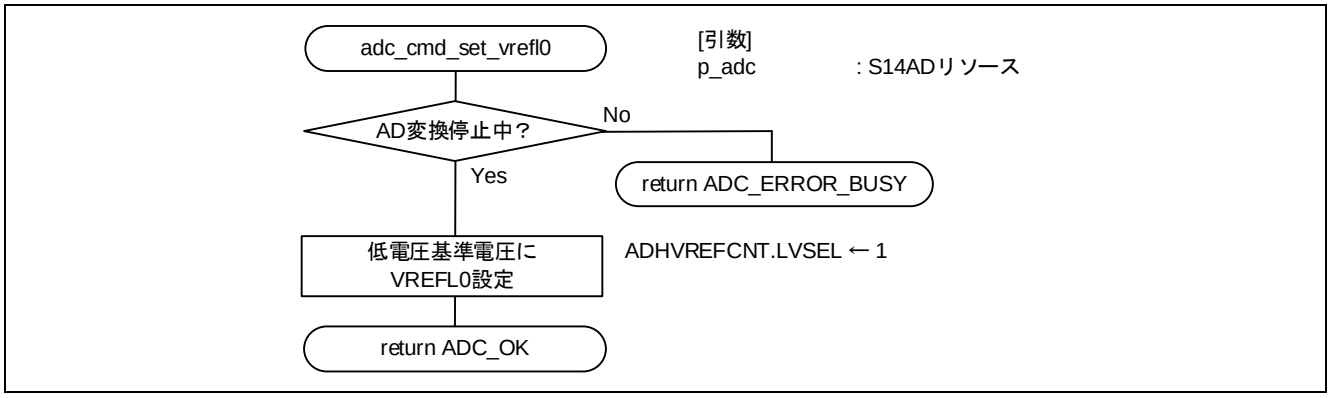


図 4-27 adc_cmd_set_vrefl0 関数処理フロー

4.1.22 adc_cmd_set_vrefh0 関数

表 4-24 adc_cmd_set_vrefh0 関数仕様

書式	static e_adc_err_t adc_cmd_set_vrefh0(st_adc_resources_t * const p_adc)
仕様説明	コンペア機能比較結果を取得します
引数	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	ADC_OK　コンペア機能比較結果取得成功
	ADC_ERROR_MODE　モードエラー 以下のいずれかの条件を検出するとモードエラーとなります ・自己診断許可状態かつ自己診断電圧ローテーションモードの場合 ・自己診断許可状態かつ自己診断変換電圧に基準電圧×1/2 を選択している場合 ・自己診断許可状態かつ自己診断変換電圧に基準電圧を選択している場合
	ADC_ERROR_BUSY　ビジー状態による設定失敗 A/D 変換実行中に実行した場合、ビジー状態による設定失敗となります
備考	-

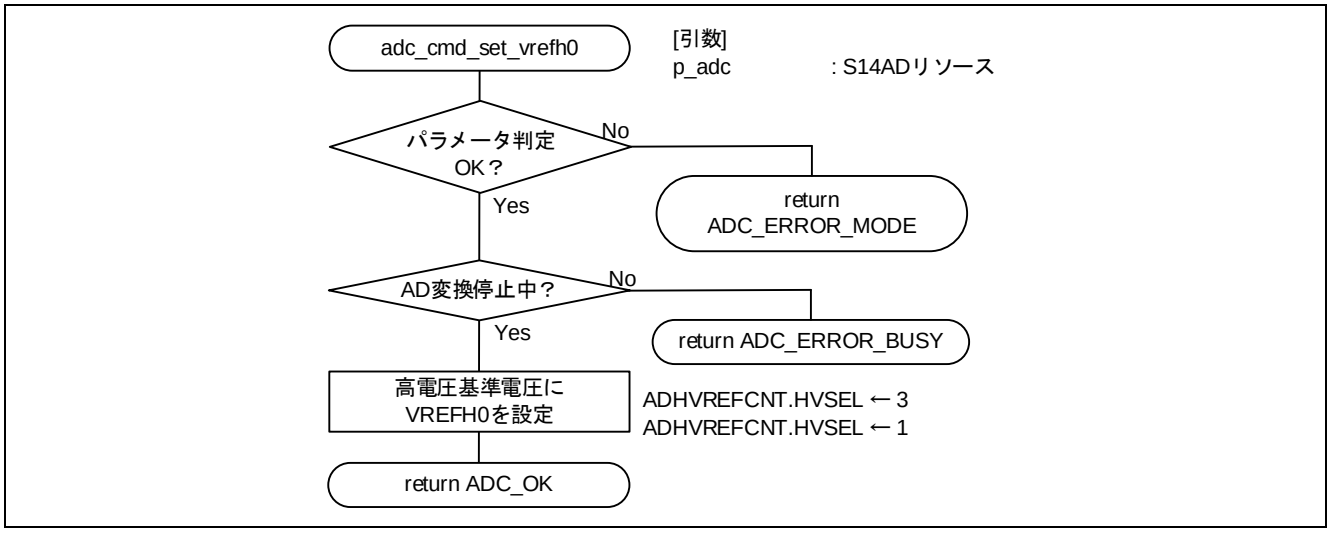


図 4-28 adc_cmd_set_vrefh0 関数処理フロー

4.1.23 adc_cmd_set_sclk 関数

表 4-25 adc_cmd_set_sclk 関数仕様

書式	static e_adc_err_t adc_cmd_set_sclk(st_adc_resources_t * const p_adc)
仕様説明	サブクロックモードの動作設定を行います
引数	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	ADC_OK　サブクロックモード設定成功
	ADC_ERROR　サブクロックモード設定失敗 動作クロック(ADCLK)が 32768Hz より大きい(低周波動作クロック(32kHz)動作設定ではない)場合、サブクロックモード失敗となります。
	ADC_ERROR_BUSY　ビジー状態による設定失敗 A/D 変換実行中に実行した場合、ビジー状態による設定失敗となります
備考	-

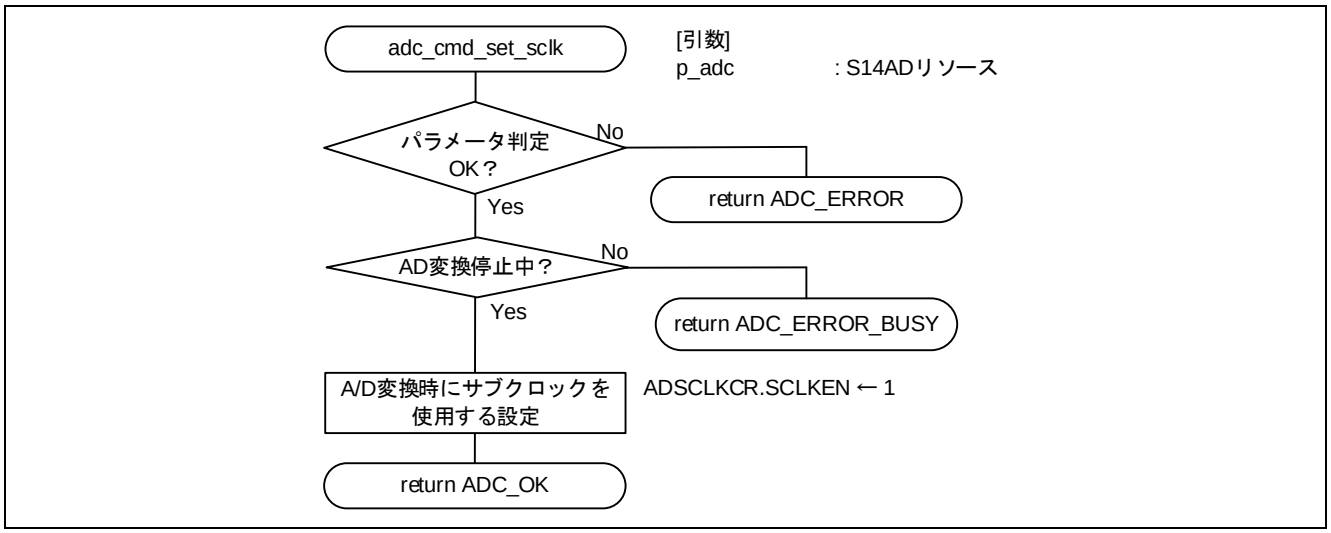


図 4-29 adc_cmd_set_sclk 関数処理フロー

4.1.24 adc_cmd_calibration 関数

表 4-26 adc_cmd_calibration 関数仕様

書式	static e_adc_err_t adc_cmd_calibration(st_adc_resources_t * const p_adc)
仕様説明	オフセットキャリブレーションを実行します
引数	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	ADC_OK　オフセットキャリブレーション成功
	ADC_ERROR　オフセットキャリブレーション失敗 ScanSet 関数による A/D ドライバの初期化（A/D 変換端子設定および A/D 変換開始トリガ設定）を行っていた場合、オフセットキャリブレーション失敗となります
	ADC_ERROR_BUSY　ビジー状態による設定失敗 A/D 変換実行中に実行した場合、ビジー状態による設定失敗となります
備考	-

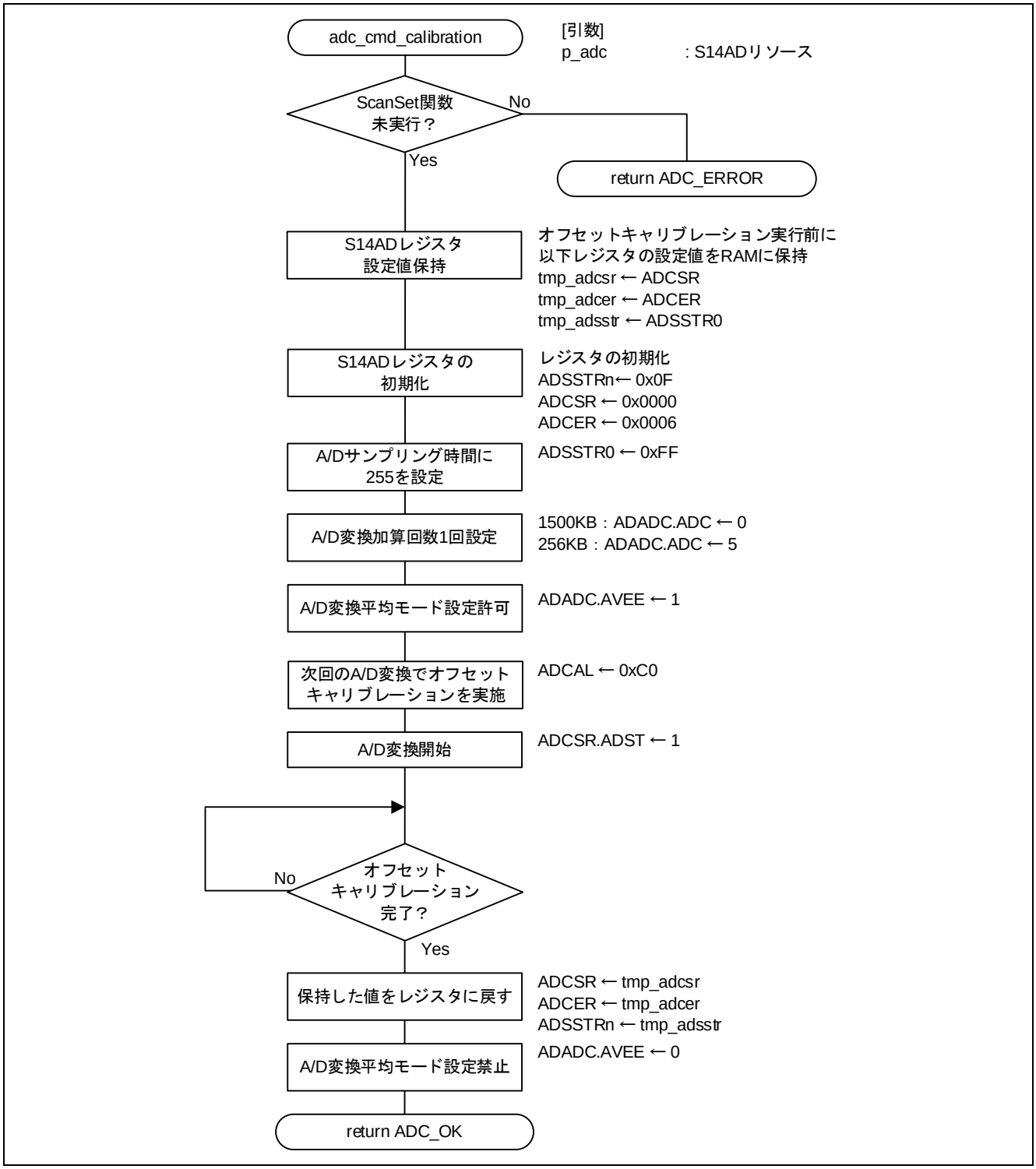
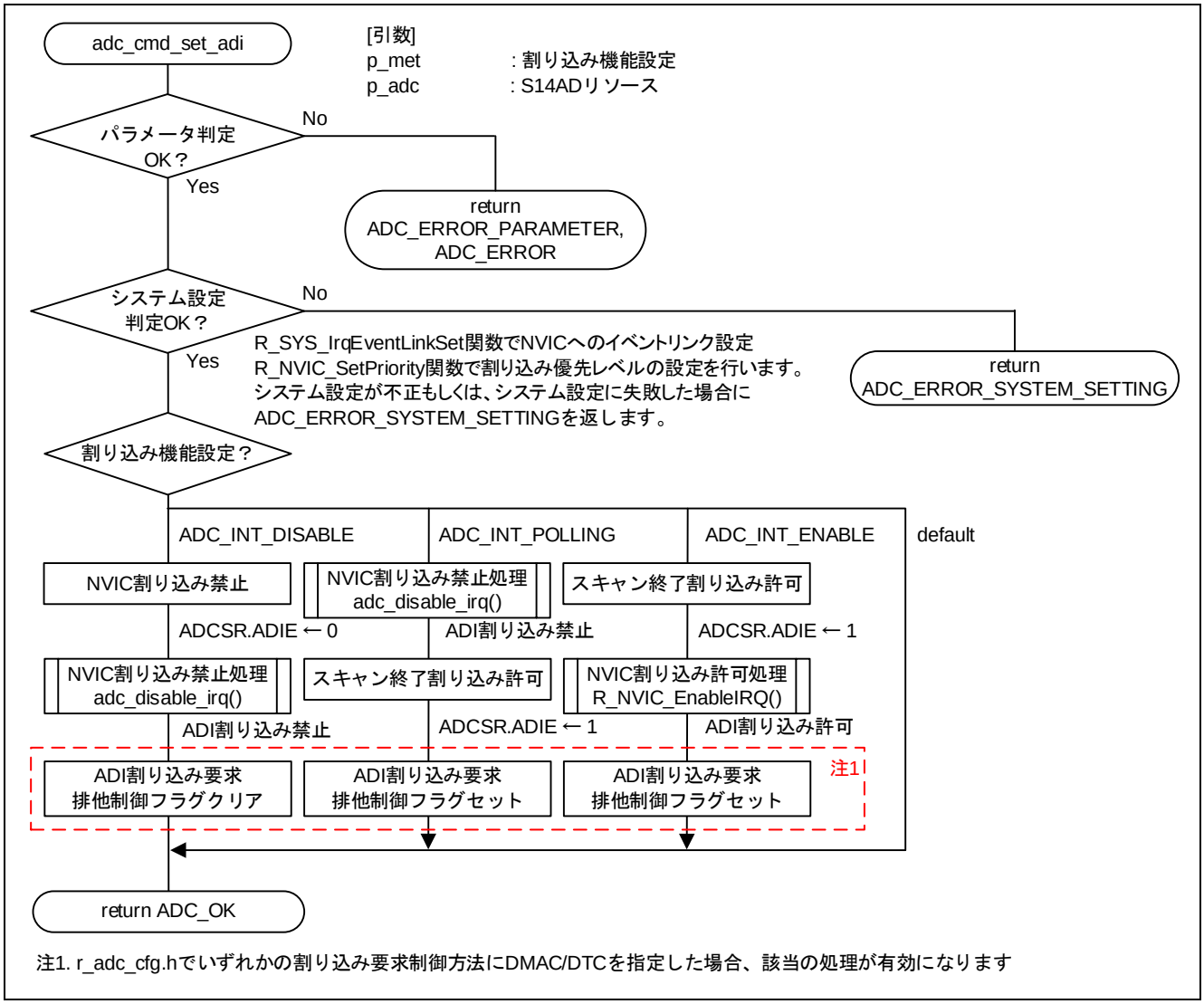


図 4-30 adc_cmd_calibration 関数処理フロー

4.1.25 adc_cmd_set_adi 関数

表 4-27 adc_cmd_set_adi 関数仕様

書式	static e_adc_err_t adc_cmd_set_adi (e_adc_int_method_t const * const p_met, st_adc_resources_t * const p_adc)
仕様説明	シングルスキャンもしくはグループ A スキャン終了割り込み(ADC140_ADI)（以下、ADI 割り込み）の設定を行います
引数	e_adc_int_method_t const * const p_met：割り込み機能設定 以下のいずれかを設定します ADC_INT_DISABLE：割り込み禁止 ADC_INT_POLLING：ポーリング ADC_INT_ENABLE：割り込み許可
	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	ADC_OK　ADI 割り込み機能設定成功
	ADC_ERROR　ADI 割り込み機能設定失敗 オートリード機能で ADI 割り込みを DMA 起動要因に使用していた場合、ADI 割り込み機能設定失敗となります
	ADC_ERROR_PARAMETER　パラメータエラー 引数に規定外の値が指定された場合、パラメータエラーとなります
	ADC_ERROR_SYSTEM_SETTING　システムエラー 以下のいずれかの条件を検出するとシステムエラーとなります ・r_system_cfg.h で ADI 割り込みが未使用定義（SYSTEM_IRQ_EVENT_NUMBER_NOT_USED）になっている場合 ・r_adc_cfg.h で S14AD_ADI_PRIORITY の設定が定義範囲を超えている場合
備考	-



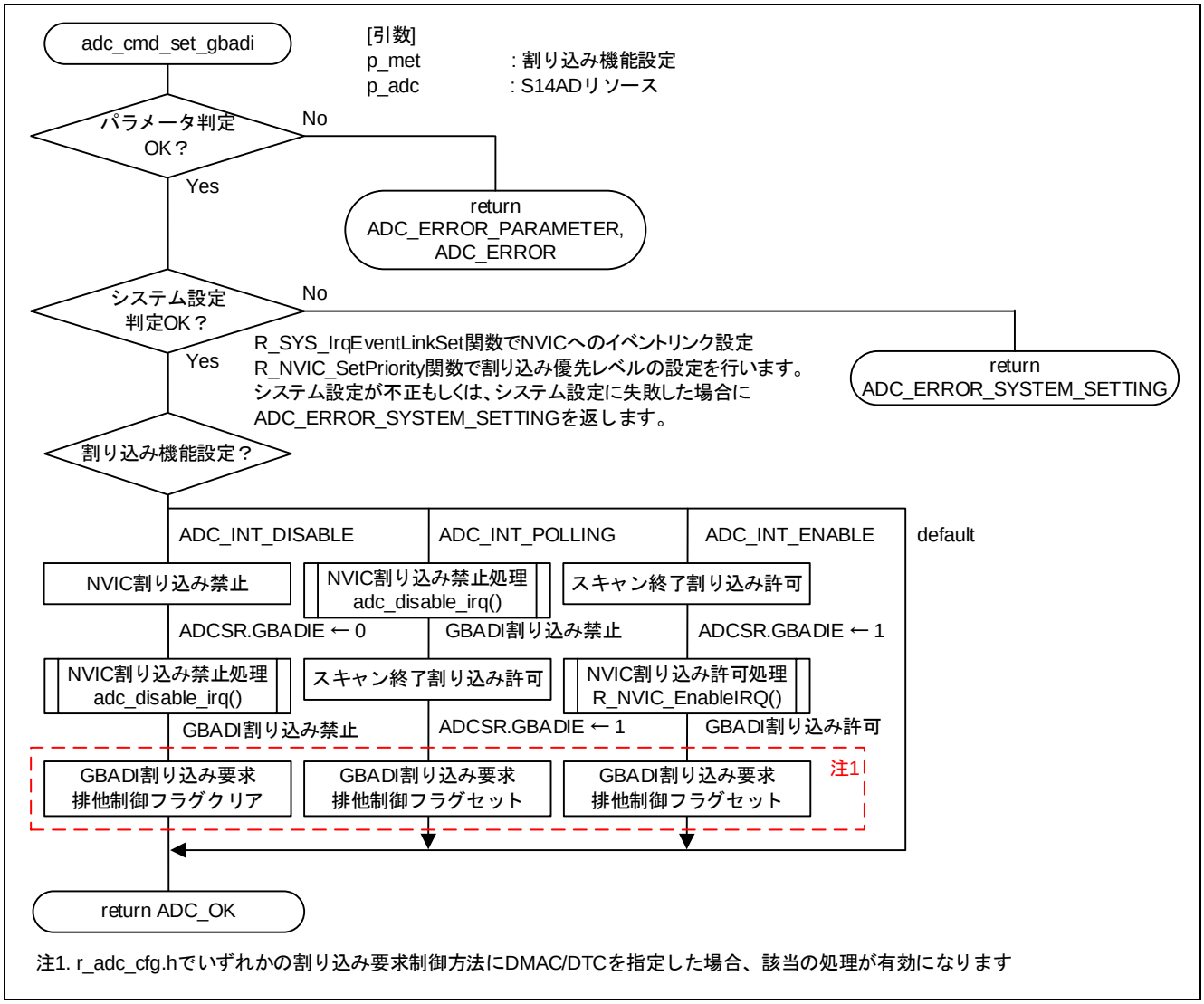
注1. r_adc_cfg.hでいずれかの割り込み要求制御方法にDMAC/DTCを指定した場合、該当の処理が有効になります

図 4-31 adc_cmd_set_adi 関数処理フロー

4.1.26 adc_cmd_set_gbadi 関数

表 4-28 adc_cmd_set_gbadi 関数仕様

書式	static e_adc_err_t adc_cmd_set_gbadi (e_adc_int_method_t const * const p_met, st_adc_resources_t * const p_adc)
仕様説明	グループ B スキャン終了割り込み(ADC140_GBADI)（以下、GBADI 割り込み）の設定を行います
引数	e_adc_int_method_t const * const p_met：割り込み機能設定 以下のいずれかを設定します ADC_INT_DISABLE：割り込み禁止 ADC_INT_POLLING：ポーリング ADC_INT_ENABLE：割り込み許可
	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	ADC_OK　GBADI 割り込み機能設定成功
	ADC_ERROR　GBADI 割り込み機能設定失敗 オートリード機能で GBADI 割り込みを DMA 起動要因に使用していた場合、GBADI 割り込み機能設定失敗となります
	ADC_ERROR_PARAMETER　パラメータエラー 引数に規定外の値が指定された場合、パラメータエラーとなります
	ADC_ERROR_SYSTEM_SETTING　システムエラー 以下のいずれかの条件を検出するとシステムエラーとなります ・r_system_cfg.h で GBADI 割り込みが未使用定義（SYSTEM_IRQ_EVENT_NUMBER_NOT_USED）になっている場合 ・r_adc_cfg.h で S14AD_GBADI_PRIORITY の設定が定義範囲を超えている場合
備考	-



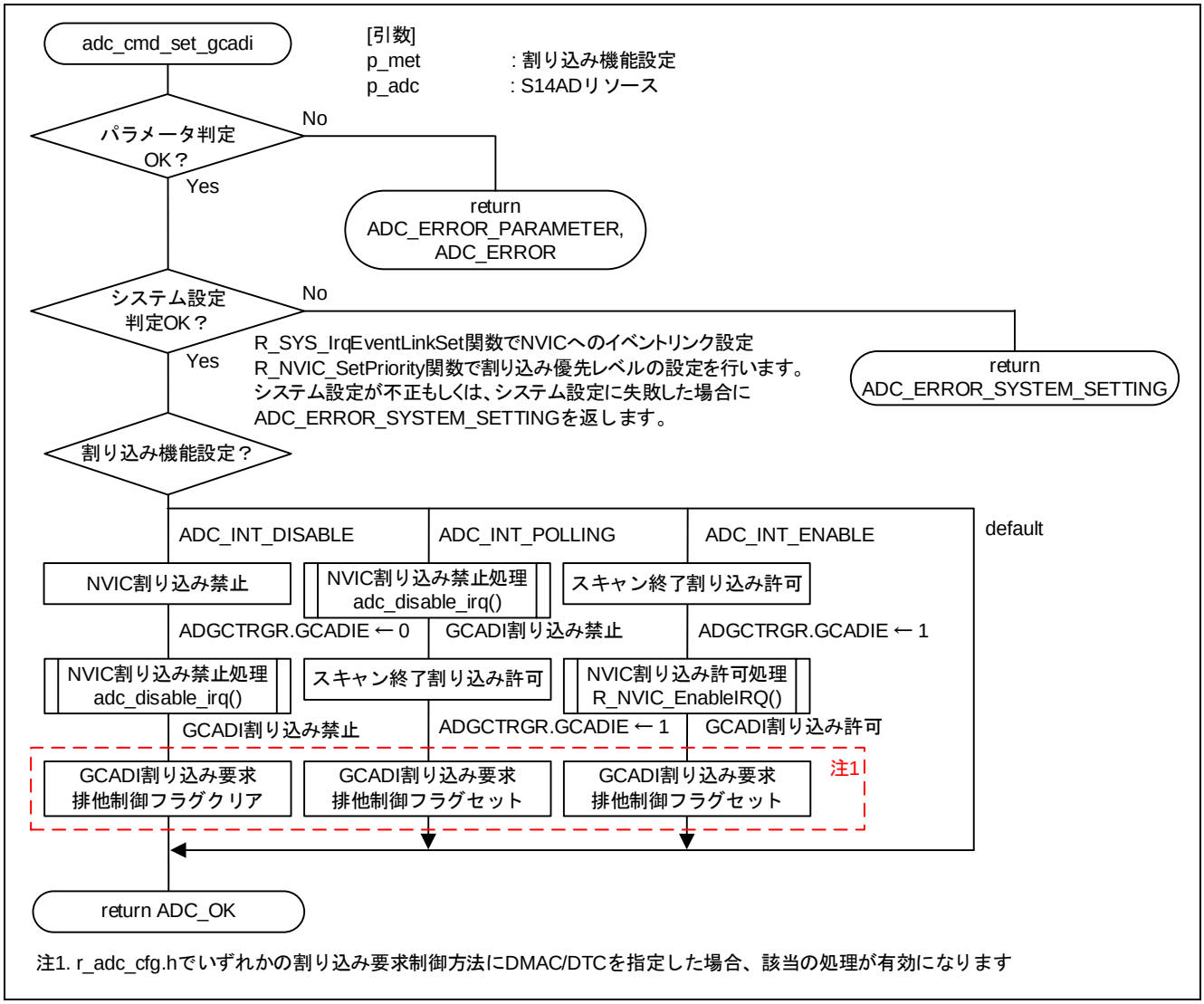
注1. r_adc_cfg.hでいずれかの割り込み要求制御方法にDMAC/DTCを指定した場合、該当の処理が有効になります

図 4-32 adc_cmd_set_gbadi 関数処理フロー

4.1.27 adc_cmd_set_gcadi 関数

表 4-29 adc_cmd_set_gcadi 関数仕様

書式	static e_adc_err_t adc_cmd_set_gcadi (e_adc_int_method_t const * const p_met, st_adc_resources_t * const p_adc)
仕様説明	グループ C スキャン終了割り込み(ADC140_GCADI)（以下、GCADI 割り込み）の設定を行います
引数	e_adc_int_method_t const * const p_met：割り込み機能設定 以下のいずれかを設定します ADC_INT_DISABLE：割り込み禁止 ADC_INT_POLLING：ポーリング ADC_INT_ENABLE：割り込み許可
	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	ADC_OK　GCADI 割り込み機能設定成功
	ADC_ERROR　GCADI 割り込み機能設定失敗 オートリード機能で GCADI 割り込みを DMA 起動要因に使用していた場合、GCADI 割り込み機能設定失敗となります
	ADC_ERROR_PARAMETER　パラメータエラー 引数に規定外の値が指定された場合、パラメータエラーとなります
	ADC_ERROR_SYSTEM_SETTING　システムエラー 以下のいずれかの条件を検出するとシステムエラーとなります ・r_system_cfg.h で GCADI 割り込みが未使用定義（SYSTEM_IRQ_EVENT_NUMBER_NOT_USED）になっている場合 ・r_adc_cfg.h で S14AD_GCADI_PRIORITY の設定が定義範囲を超えている場合
備考	-



注1. r_adc_cfg.hでいずれかの割り込み要求制御方法にDMAC/DTCを指定した場合、該当の処理が有効になります

図 4-33 adc_cmd_set_gcadi 関数処理フロー

4.1.28 adc_cmd_set_wcmpm 関数

表 4-30 adc_cmd_set_wcmpm 関数仕様

書式	static e_adc_err_t adc_cmd_set_wcmpm (e_adc_int_method_t const * const p_met, st_adc_resources_t * const p_adc)
仕様説明	ウィンドウ A/B コンペア機能の条件一致割り込み(ADC140_WCMPM)（以下、WCMPM 割り込み）の設定を行います
引数	e_adc_int_method_t const * const p_met：割り込み機能設定 以下のいずれかを設定します ADC_INT_DISABLE：割り込み禁止 ADC_INT_POLLING：ポーリング ADC_INT_ENABLE：割り込み許可
	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	ADC_OK　WCMPM 割り込み機能設定成功
	ADC_ERROR　WCMPM 割り込み機能設定失敗 オートリード機能で WCMPM 割り込みを DMA 起動要因に使用していた場合、WCMPM 割り込み機能設定失敗となります
	ADC_ERROR_PARAMETER　パラメータエラー 引数に規定外の値が指定された場合、パラメータエラーとなります
	ADC_ERROR_SYSTEM_SETTING　システムエラー 以下のいずれかの条件を検出するとシステムエラーとなります ・r_system_cfg.h で WCMPM 割り込みが未使用定義（SYSTEM_IRQ_EVENT_NUMBER_NOT_USED）になっている場合 ・r_adc_cfg.h で S14AD_WCMPM_PRIORITY の設定が定義範囲を超えている場合
備考	-

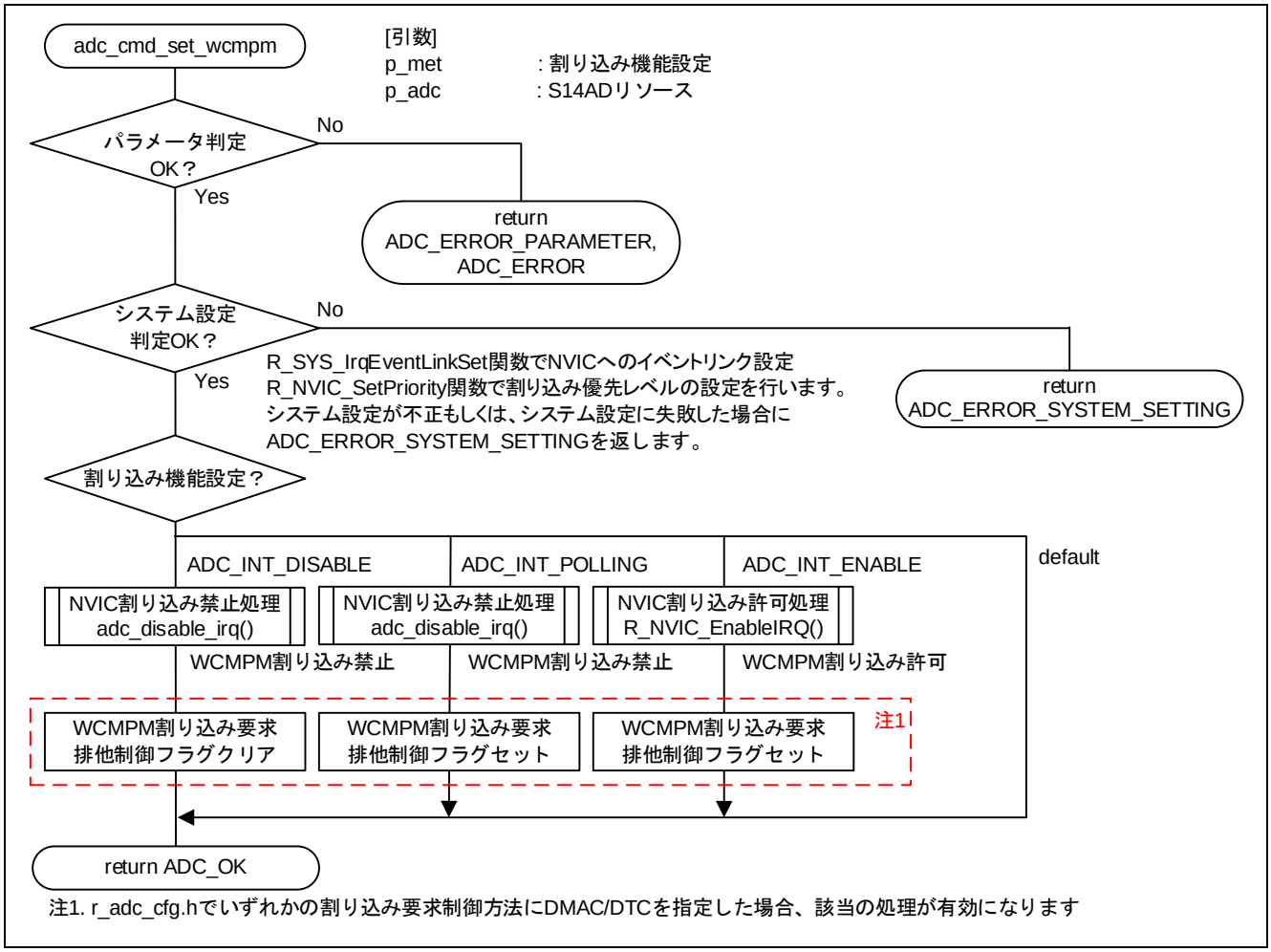


図 4-34 adc_cmd_set_wcmpm 関数処理フロー

4.1.29 adc_cmd_set_wcmpum 関数

表 4-31 adc_cmd_set_wcmpum 関数仕様

書式	static e_adc_err_t adc_cmd_set_wcmpum (e_adc_int_method_t const * const p_met, st_adc_resources_t * const p_adc)
仕様説明	ウィンドウ A/B コンペア機能の条件不一致割り込み(ADC140_WCMPUM)（以下、WCMPUM 割り込み）の設定を行います
引数	e_adc_int_method_t const * const p_met：割り込み機能設定 以下のいずれかを設定します ADC_INT_DISABLE：割り込み禁止 ADC_INT_POLLING：ポーリング ADC_INT_ENABLE：割り込み許可
	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	ADC_OK　WCMPUM 割り込み機能設定成功
	ADC_ERROR　WCMPUM 割り込み機能設定失敗 オートリード機能で WCMPUM 割り込みを DMA 起動要因に使用していた場合、WCMPUM 割り込み機能設定失敗となります
	ADC_ERROR_PARAMETER　パラメータエラー 引数に規定外の値が指定された場合、パラメータエラーとなります
	ADC_ERROR_SYSTEM_SETTING　システムエラー 以下のいずれかの条件を検出するとシステムエラーとなります ・r_system_cfg.h で WCMPUM 割り込みが未使用定義（SYSTEM_IRQ_EVENT_NUMBER_NOT_USED）になっている場合 ・r_adc_cfg.h で S14AD_WCMPUM_PRIORITY の設定が定義範囲を超えている場合
備考	-

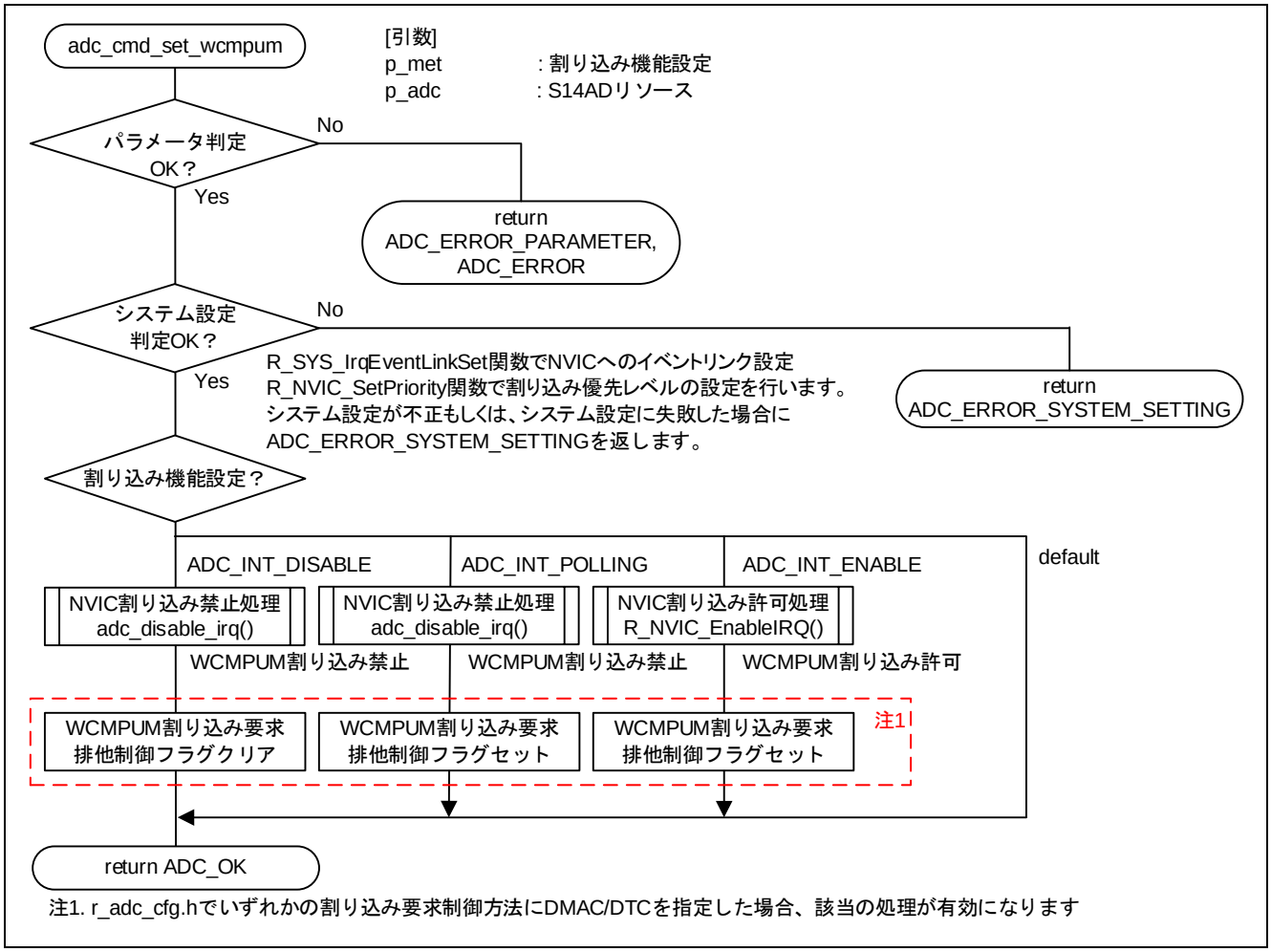


図 4-35 adc_cmd_set_wcmpum 関数処理フロー

4.1.30 adc_cmd_set_elc 関数

表 4-32 adc_cmd_set_elc 関数仕様

書式	static e_adc_err_t adc_cmd_set_elc(e_adc_elc_mode_t const * const p_elc, st_adc_resources_t * const p_adc)
仕様説明	コンペア機能比較結果を取得します
引数	e_adc_elc_mode_t const * const p_elc：スキャン終了イベント発生条件設定 以下のいずれかを設定してください ADC_ELC_GROUPA：グループ B とグループ C のスキャン終了を除くスキャン終了時にイベント発生 ADC_ELC_GROUPB：グループ B スキャン終了時にイベント発生 ADC_ELC_GROUPC：グループ C スキャン終了時にイベント発生 ADC_ELC_ALL：すべてのスキャン終了時にイベント発生
	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	ADC_OK　スキャン終了イベント発生条件設定成功
	ADC_ERROR_PARAMETER　パラメータエラー 以下のいずれかの条件を検出するとパラメータエラーとなります ・引数が NULL の場合 ・引数に規定外の値が指定された場合
備考	-

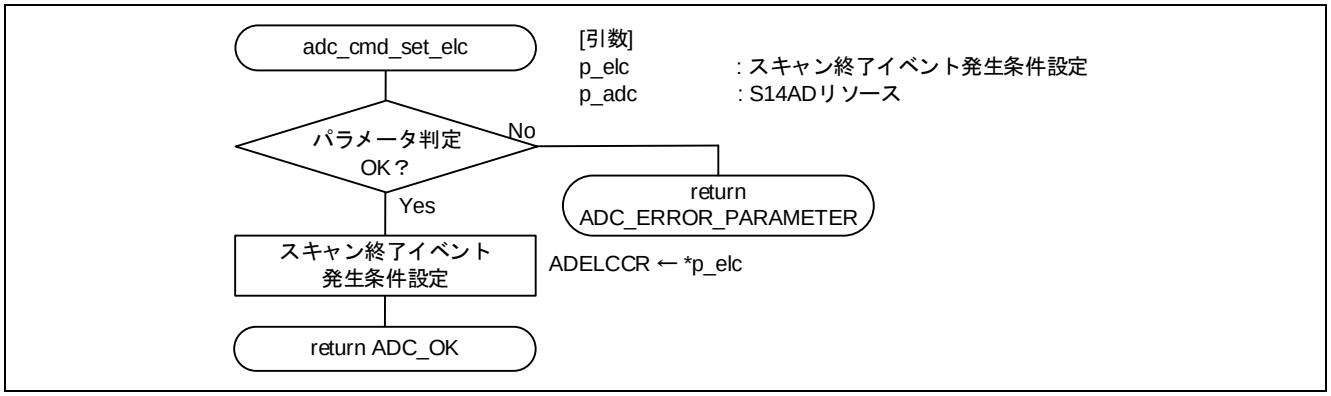


図 4-36 adc_cmd_set_elc 関数処理フロー

4.1.31 adc_cmd_stop_trigger 関数

表 4-33 adc_cmd_stop_trigger 関数仕様

書式	static e_adc_err_t adc_cmd_stop_trigger(st_adc_resources_t * const p_adc)
仕様説明	全グループの A/D 変換開始トリガ設定を解除し、A/D 変換を停止させます
引数	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	ADC_OK　A/D 変換停止成功
	ADC_ERROR　A/D 変換停止失敗 ScanSet 関数による初期が未実施の場合、A/D 変換停止失敗となります
備考	-

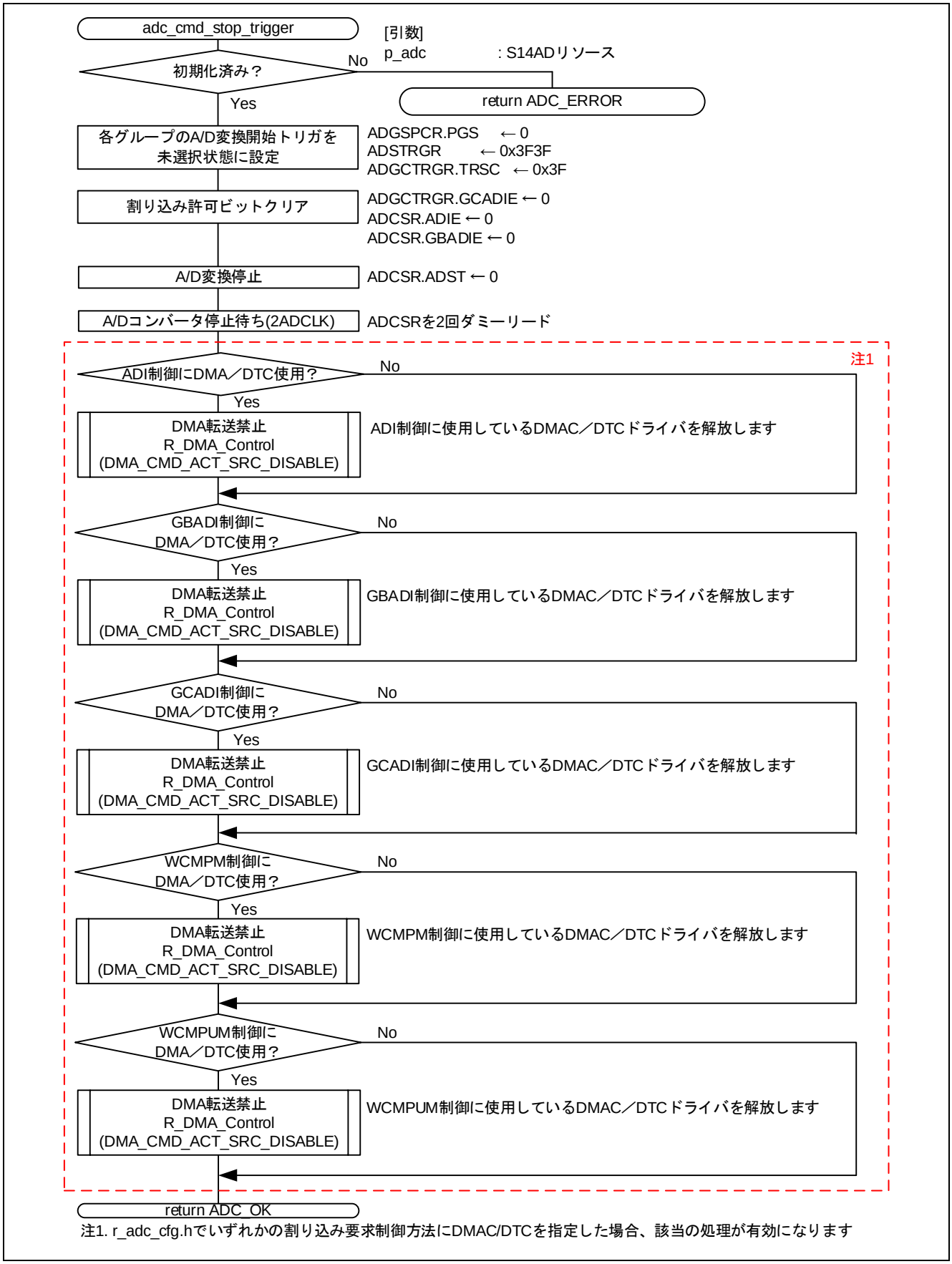


図 4-37 adc_cmd_stop_trigger 関数処理フロー

4.1.32 adc_cmd_auto_read 関数

表 4-34 adc_cmd_auto_read 関数仕様

書式	static e_adc_err_t adc_cmd_auto_read(st_adc_dma_read_info_t const * const p_dma, e_adc_dma_cmd_t cmd, st_adc_resources_t * const p_adc)
仕様説明	オートリード機能の設定を行います
引数	st_adc_dma_read_info_t const * const p_dma：オートリード機能設定
	e_adc_dma_cmd_t cmd：オートリード機能コマンド
	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	ADC_OK　オートリード機能設定成功
	ADC_ERROR　オートリード機能設定失敗 DMA 起動要因に指定した割り込み要求が割り込み機能（割り込みもしくはポーリング）で使用されている場合、オートリード機能設定失敗となります
	ADC_ERROR_SYSTEM_SETTING　システムエラー 以下のいずれかの条件を検出するとシステムエラーとなります [共通] ・r_adc_cfg.h で DMA 起動要因とする割り込み要求制御設定(S14AD_xxx_CONTROL)が割り込み設定(S14AD_USED_INTERRUPT)になっている場合(xxx = ADI、GBADI、GCADI、WCMPM、WCMPUM) [DTC 使用] ・r_system_cfg.h で DMA 起動要因の割り込みが未使用定義（SYSTEM_IRQ_EVENT_NUMBER_NOT_USED）になっている場合 ・r_adc_cfg.h で DMA 起動要因の割り込み優先順位設定が定義範囲を超えている場合 [DMAC 使用] ・コールバック関数を指定したとき、r_system_cfg.h で DMACn_INT(n = 0～3)の割り込みが未使用定義（SYSTEM_IRQ_EVENT_NUMBER_NOT_USED）になっている場合 ・コールバック関数を指定したとき、r_dma_api_cfg.h で DMACn_INT_PRIORITY の設定が定義範囲を超えている場合
	ADC_ERROR_PARAMETER　パラメータエラー 以下のいずれかの条件を検出するとパラメータエラーとなります ・引数に規定外の値が指定された場合 ・オートリード機能（コンペア）で DMA 起動要因に ADI、GBADI、GCADI 割り込み要求を指定した場合 ・オートリード機能（コンペア）以外で DMA 起動要因に WCMPM、WCMPUM 割り込み要求をしてした場合 ・オートリード機能（コンペア）で転送サイズ設定が不正な場合
備考	r_adc_cfg.h でいずれかの割り込み要求制御に DMAC/DTC を指定していた場合のみ有効

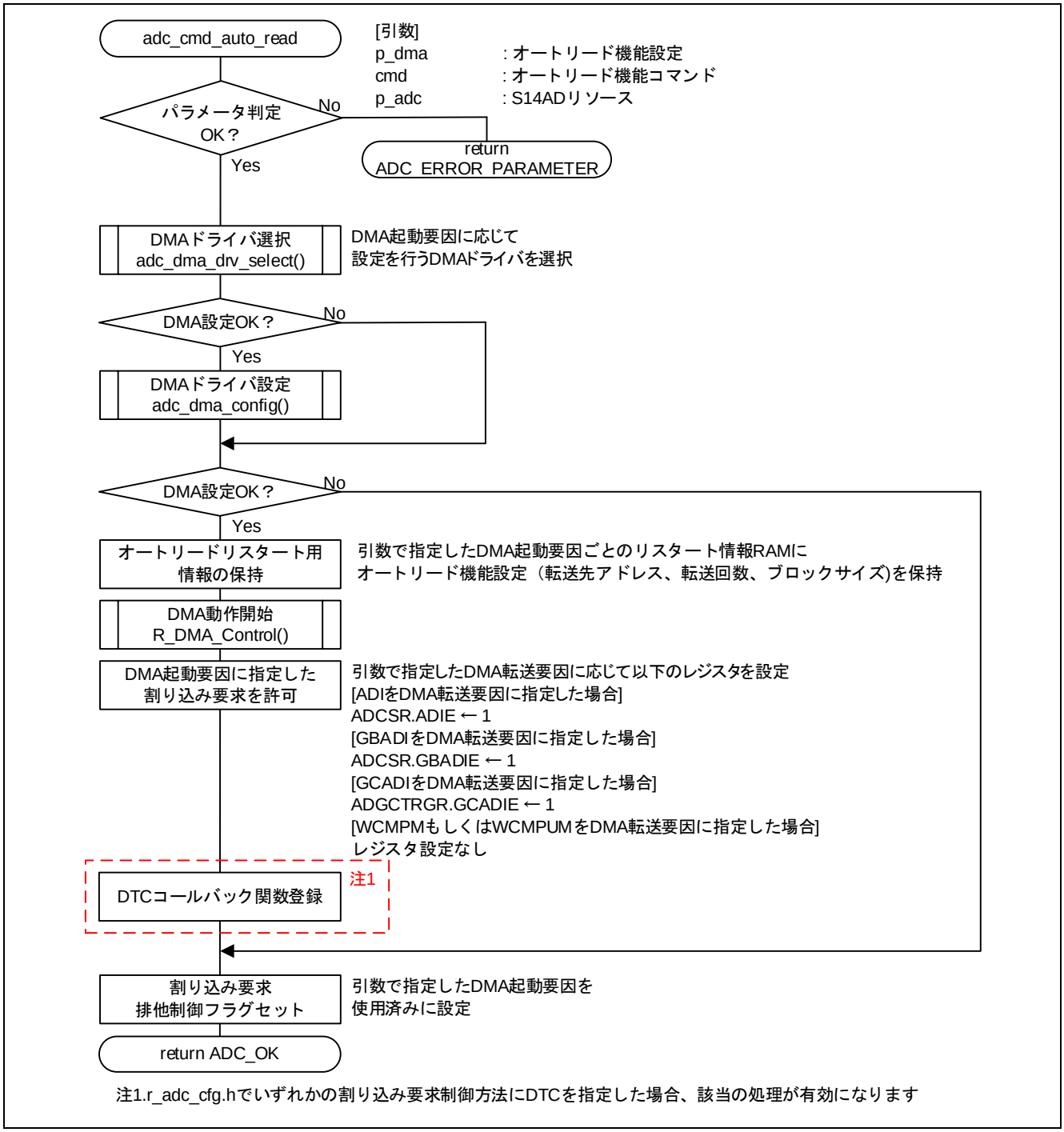


図 4-38 adc_cmd_auto_read 関数処理フロー

4.1.33 adc_cmd_auto_read_stop 関数

表 4-35 adc_cmd_auto_read_stop 関数仕様

書式	static e_adc_err_t adc_cmd_auto_read_stop(e_adc_dma_event_t const * const event, st_adc_resources_t * const p_adc)
仕様説明	指定した DMA 起動要因によるオートリードを停止します
引数	e_adc_dma_event_t const * const event：DMA 起動要因
	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	ADC_OK　オートリード機能停止成功
	ADC_ERROR_SYSTEM_SETTING　システムエラー r_adc_cfg.h で DMA 起動要因とする割り込み要求制御設定(S14AD_xxx_CONTROL)が割り込み設定(S14AD_USED_INTERRUPT)になっている場合(xxx = ADI、GBADI、GCADI、WCMPM、WCMPUM)、システムエラーとなります
	ADC_ERROR_PARAMETER　パラメータエラー 引数に規定外の値が指定された場合パラメータエラーとなります
備考	r_adc_cfg.h でいずれかの割り込み要求制御に DMAC/DTC を指定していた場合のみ有効

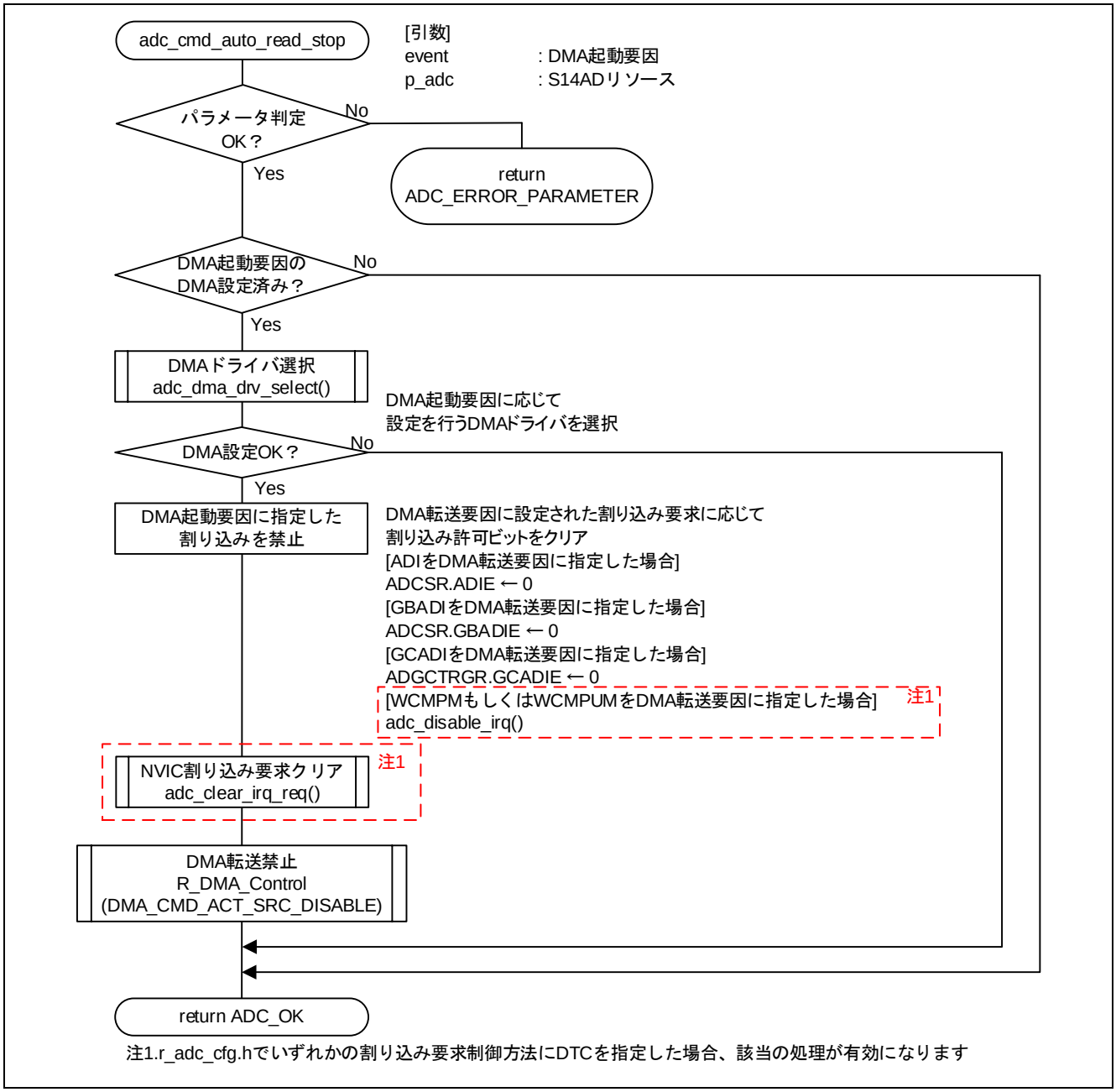


図 4-39 adc_cmd_auto_read_stop 関数処理フロー

4.1.34 adc_cmd_auto_read_restart 関数

表 4-36 adc_cmd_auto_read_restart 関数仕様

書式	static e_adc_err_t adc_cmd_auto_read_restart(e_adc_dma_event_t const * const event, st_adc_resources_t * const p_adc)
仕様説明	指定した DMA 起動要因によるオートリードを停止します
引数	e_adc_dma_event_t const * const event：DMA 起動要因
	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	ADC_OK　　オートリード機能停止成功
	ADC_ERROR_SYSTEM_SETTING　　システムエラー 以下のいずれかの条件を検出するとシステムエラーとなります ・DMA 起動要因に WCMPM を指定したとき、r_system_cfg.h で WCMPM 割り込みが未使用定義（SYSTEM_IRQ_EVENT_NUMBER_NOT_USED）になっている場合 ・DMA 起動要因に WCMPUM を指定したとき、r_system_cfg.h で WCMPUM 割り込みが未使用定義（SYSTEM_IRQ_EVENT_NUMBER_NOT_USED）になっている場合 ・r_adc_cfg.h で DMA 起動要因とする割り込み要求制御設定(S14AD_xxx_CONTROL)が割り込み設定(S14AD_USED_INTERRUPT)になっている場合(xxx = ADI、GBADI、GCADI、WCMPM、WCMPUM)、システムエラーとなります
	ADC_ERROR_PARAMETER　　パラメータエラー 引数に規定外の値が指定された場合パラメータエラーとなります
備考	r_adc_cfg.h でいずれかの割り込み要求制御に DMAC/DTC を指定していた場合のみ有効

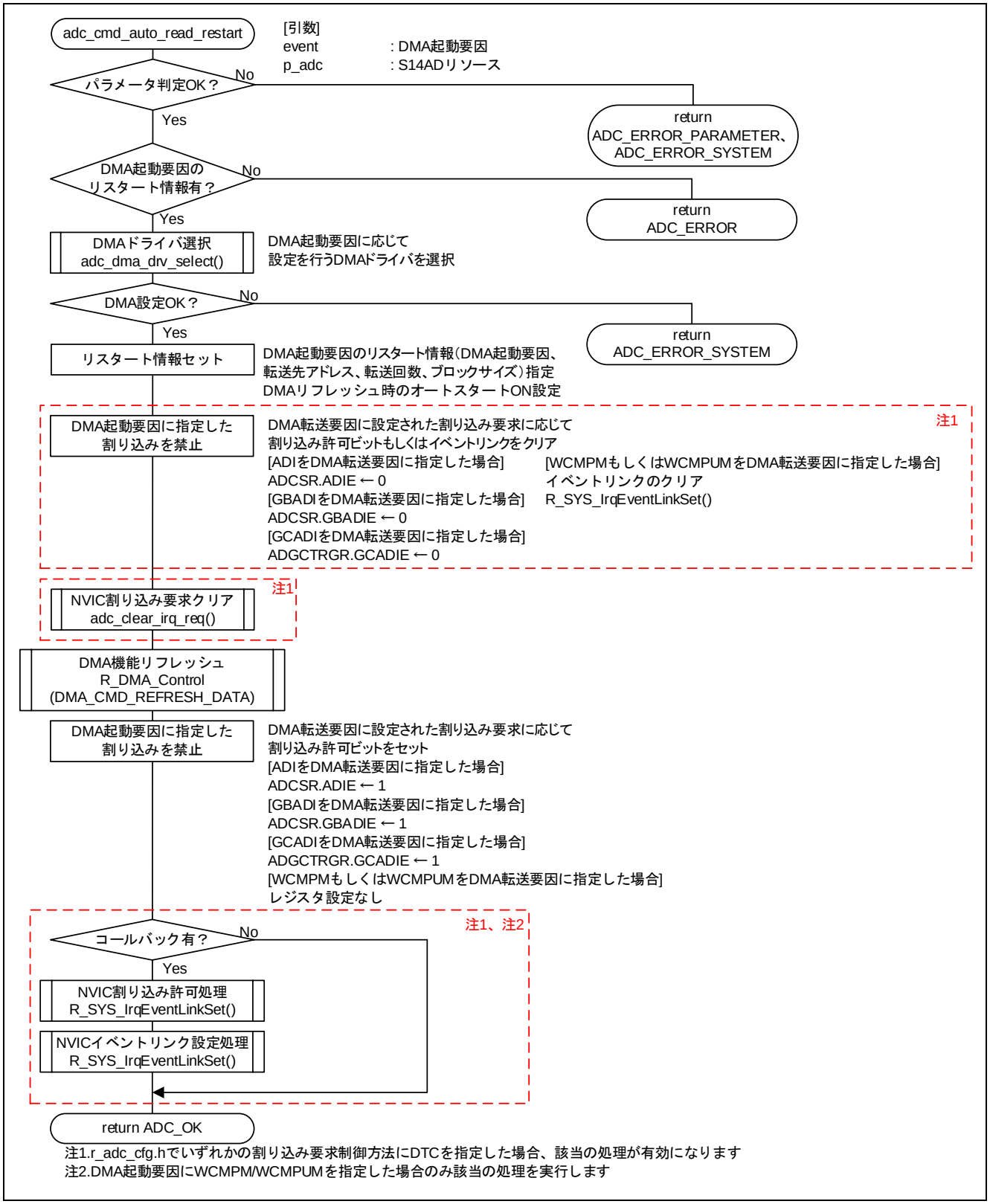


図 4-40 adc_cmd_auto_read_restart 関数処理フロー

4.1.35 adc_dma_ram_init 関数

表 4-37 adc_dma_ram_init 関数仕様

書式	void adc_dma_ram_init(st_adc_resources_t * const p_adc)
仕様説明	オートリード用 RAM の初期化を行います
引数	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	-
備考	r_adc_cfg.h でいずれかの割り込み要求制御に DMAC/DTC を指定していた場合のみ有効

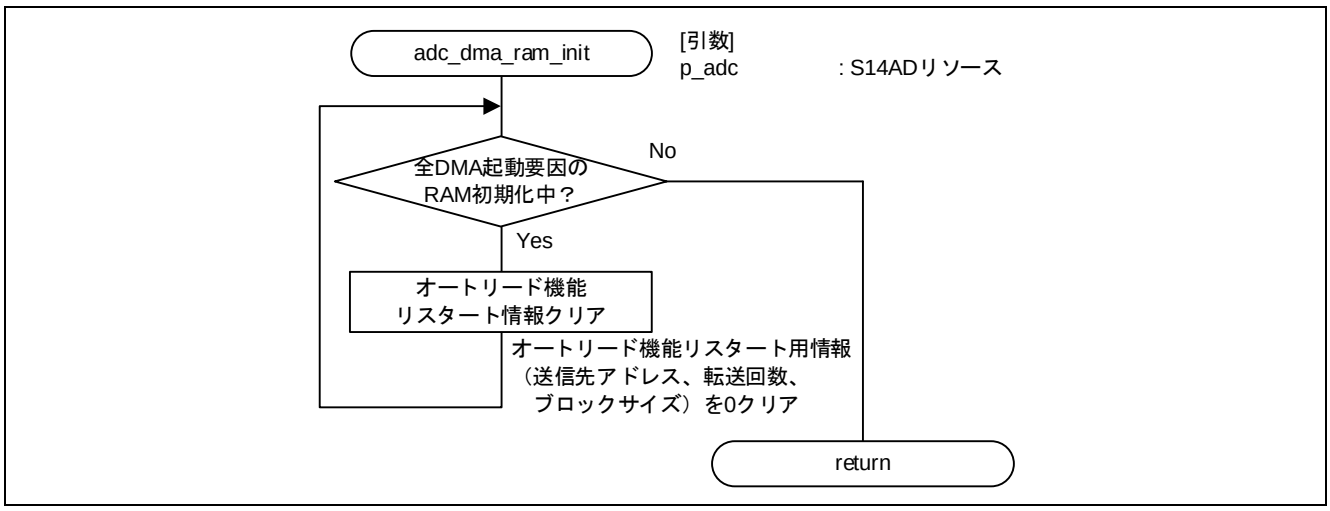


図 4-41 adc_dma_ram_init 関数処理フロー

4.1.36 adc_dma_drv_select 関数

表 4-38 adc_dma_drv_select 関数仕様

書式	static e_adc_err_t adc_dma_drv_select(st_adc_dma_set * const setting, e_adc_dma_event_t event, st_adc_resources_t * const p_adc)
仕様説明	指定した DMA 起動要因の DMA ドライバを選択します
引数	st_adc_dma_set * const setting：DMA 転送設定
	e_adc_dma_event_t const * const event：DMA 起動要因
	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	ADC_OK　オートリード機能停止成功
	ADC_ERROR_SYSTEM_SETTING　システムエラー r_adc_cfg.h で DMA 起動要因とする割り込み要求制御設定(S14AD_xxx_CONTROL)が割り込み設定(S14AD_USED_INTERRUPT)になっている場合(xxx = ADI、GBADI、GCADI、WCMPM、WCMPUM)、システムエラーとなります
備考	r_adc_cfg.h でいずれかの割り込み要求制御に DMAC/DTC を指定していた場合のみ有効

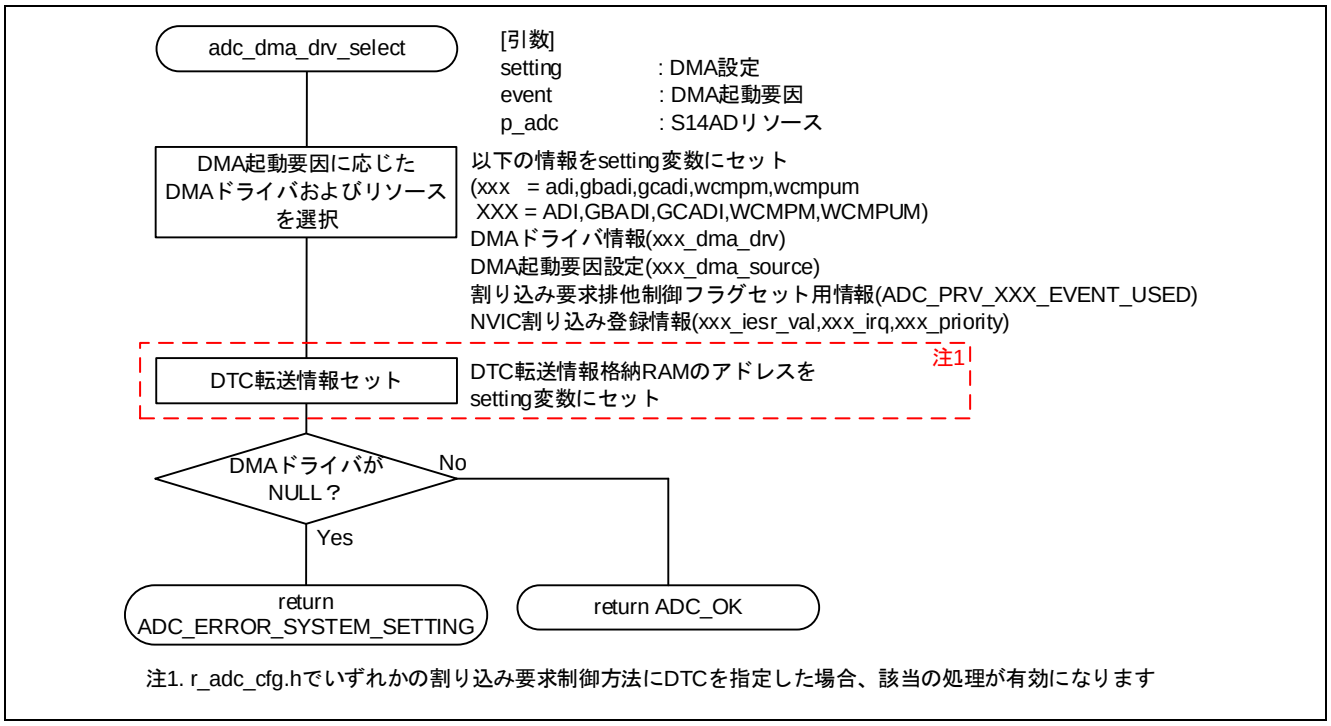


図 4-42 adc_dma_drv_select 関数処理フロー

4.1.37 adc_dma_config 関数

表 4-39 adc_dma_config 関数仕様

書式	static e_adc_err_t adc_dma_config(st_adc_dma_read_info_t const * const p_dma, st_adc_dma_set const * const setting, e_adc_dma_cmd_t cmd, st_adc_resources_t * const p_adc)
仕様説明	オートリード機能で使用する DMA ドライバを初期化します
引数	st_adc_dma_read_info_t const * const p_dma：オートリード機能設定
	st_adc_dma_set const * const setting：DMA 転送設定
	e_adc_dma_cmd_t cmd：
	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	ADC_OK　オートリード機能設定成功
	ADC_ERROR　オートリード機能設定失敗 DMA 起動要因に指定した割り込み要求が割り込み機能（割り込みもしくはポーリング）で使用されている場合、オートリード機能設定失敗となります
	ADC_ERROR_SYSTEM_SETTING　システムエラー 以下のいずれかの条件を検出するとシステムエラーとなります [共通] ・r_adc_cfg.h で DMA 起動要因とする割り込み要求制御設定(S14AD_xxx_CONTROL)が割り込み設定(S14AD_USED_INTERRUPT)になっている場合(xxx = ADI、GBADI、GCADI、WCMPM、WCMPUM) [DTC 使用] ・r_system_cfg.h で DMA 起動要因の割り込みが未使用定義（SYSTEM_IRQ_EVENT_NUMBER_NOT_USED）になっている場合 ・r_adc_cfg.h で DMA 起動要因の割り込み優先順位設定が定義範囲を超えている場合 [DMAC 使用] ・コールバック関数を指定したとき、r_system_cfg.h で DMACn_INT(n = 0～3)の割り込みが未使用定義（SYSTEM_IRQ_EVENT_NUMBER_NOT_USED）になっている場合 ・コールバック関数を指定したとき、r_dma_api_cfg.h で DMACn_INT_PRIORITY の設定が定義範囲を超えている場合
	ADC_ERROR_PARAMETER　パラメータエラー 以下のいずれかの条件を検出するとパラメータエラーとなります ・引数に規定外の値が指定された場合 ・オートリード機能（コンペア）で DMA 起動要因に ADI、GBADI、GCADI 割り込み要求を指定した場合 ・オートリード機能（コンペア）以外で DMA 起動要因に WCMPM、WCMPUM 割り込み要求をしてした場合 ・オートリード機能（コンペア）で転送サイズ設定が不正な場合
備考	r_adc_cfg.h でいずれかの割り込み要求制御に DMAC/DTC を指定していた場合のみ有効

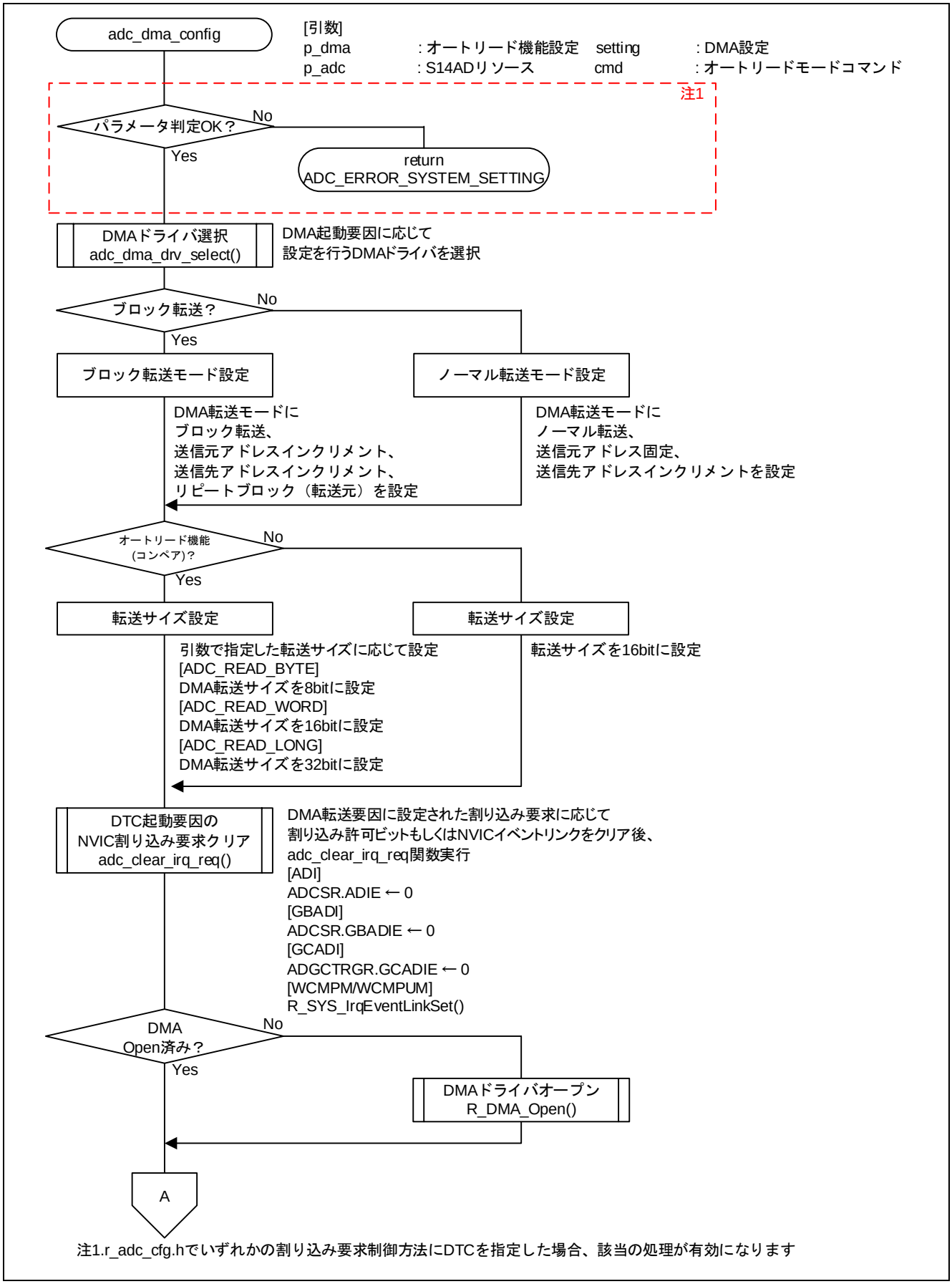


図 4-43 adc_dma_config 関数処理フロー(1/2)

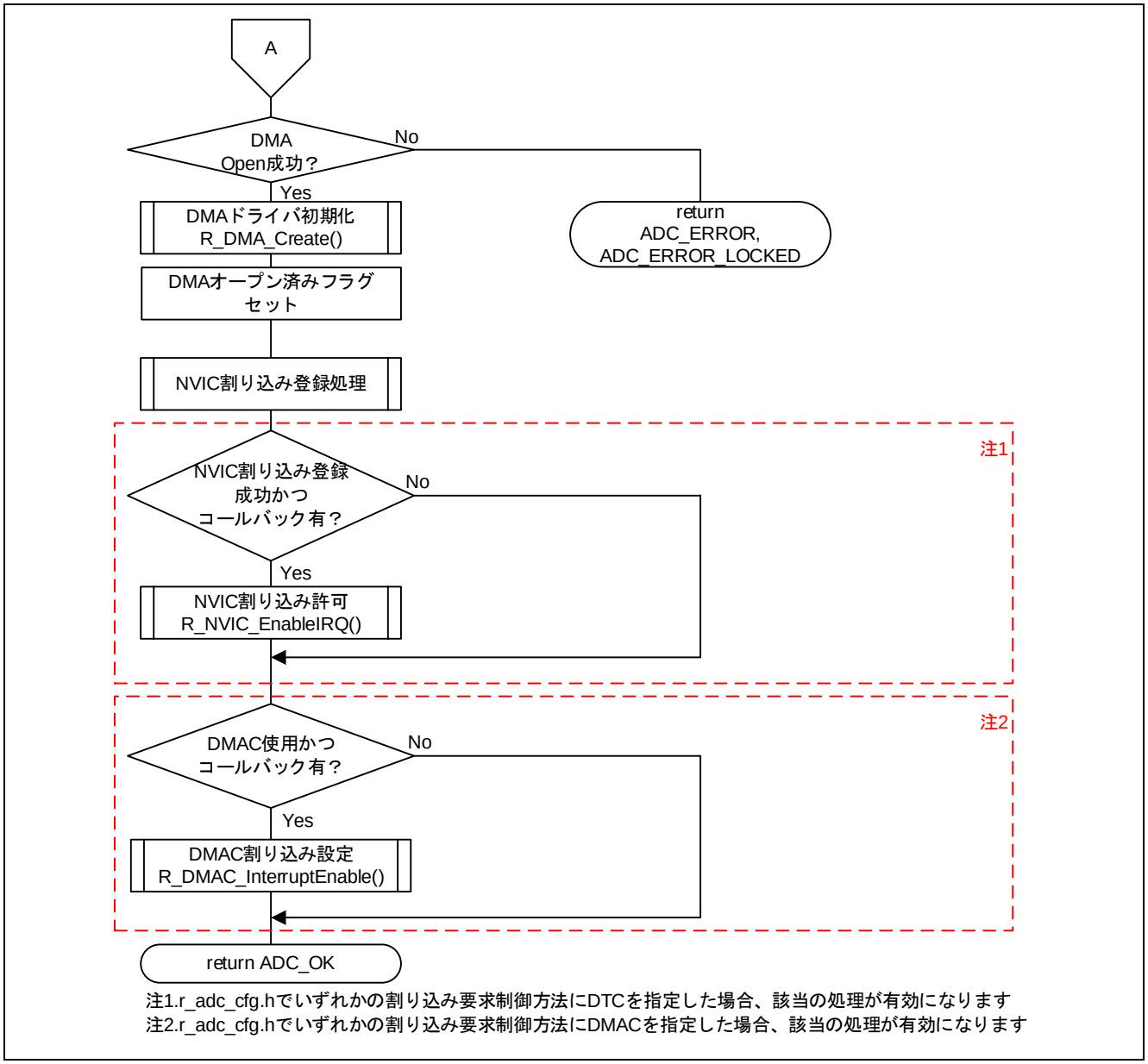


図 4-44 adc_dma_config 関数処理フロー(2/2)

4.1.38 adc_s14adi0_isr 関数

表 4-40 adc_s14adi0_isr 関数仕様

書式	static void adc_s14adi0_isr(st_adc_resources_t * const p_adc)
仕様説明	ADI 割り込み処理
引数	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	なし
備考	-

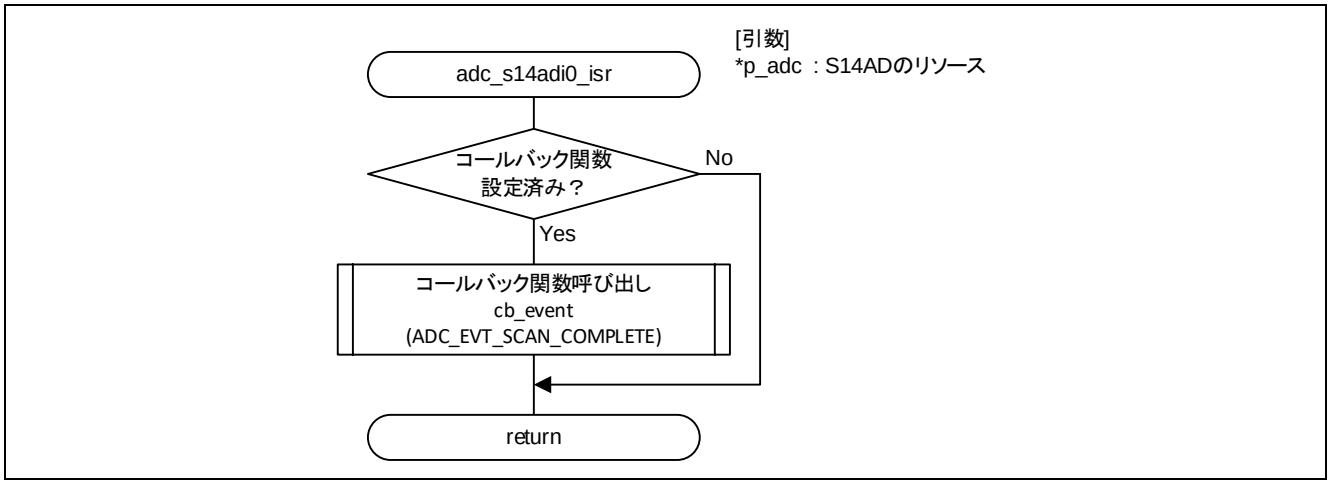


図 4-45 adc_s14adi0_isr 関数処理フロー

4.1.39 adc_gbadi_isr 関数

表 4-41 adc_gbadi_isr 関数仕様

書式	static void adc_gbadi_isr(st_adc_resources_t * const p_adc)
仕様説明	GBADI 割り込み処理
引数	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	なし
備考	-

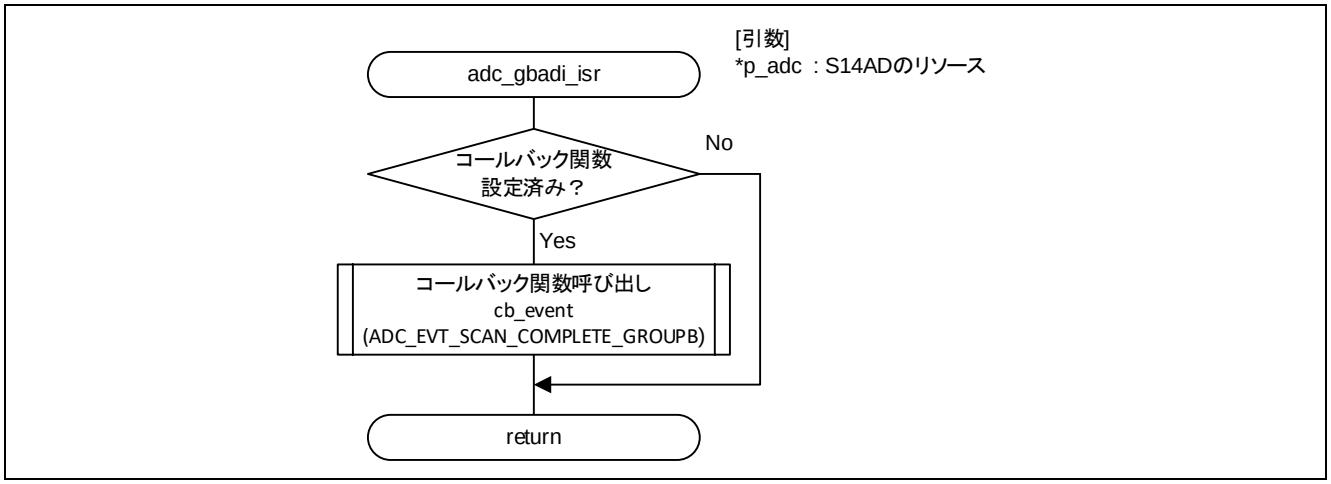


図 4-46 adc_gbadi_isr 関数処理フロー

4.1.40 adc_gcadi_isr 関数

表 4-42 adc_gcadi_isr 関数仕様

書式	static void adc_gcadi_isr(st_adc_resources_t * const p_adc)
仕様説明	GCADI 割り込み処理
引数	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	なし
備考	-

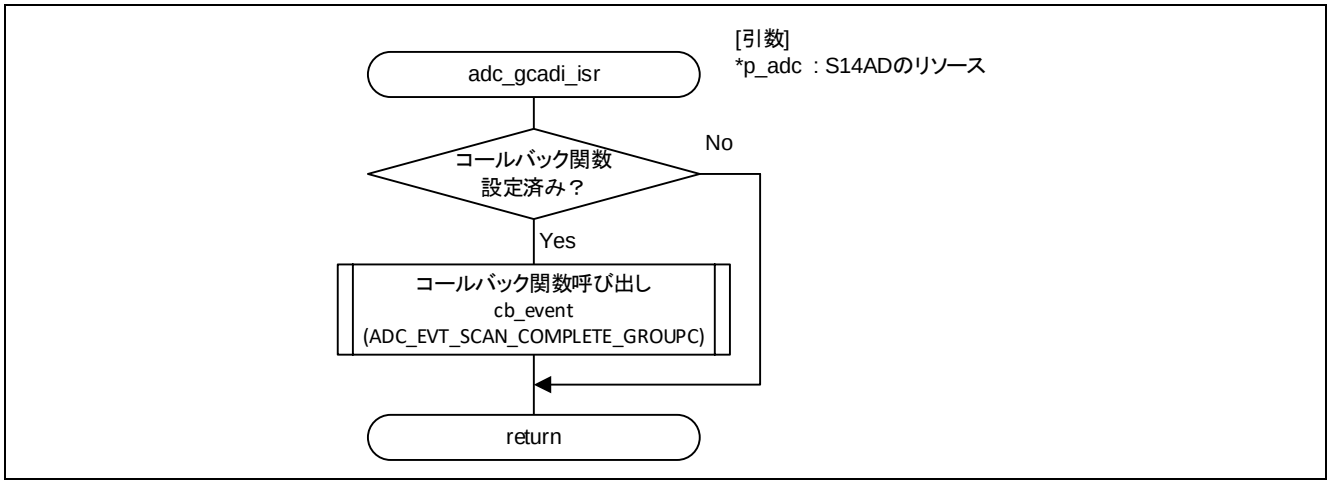


図 4-47 adc_gcadi_isr 関数処理フロー

4.1.41 adc_wcmpm_isr 関数

表 4-43 adc_wcmpm_isr 関数仕様

書式	static void adc_wcmpm_isr(st_adc_resources_t * const p_adc)
仕様説明	WCMPM 割り込み処理
引数	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	なし
備考	-

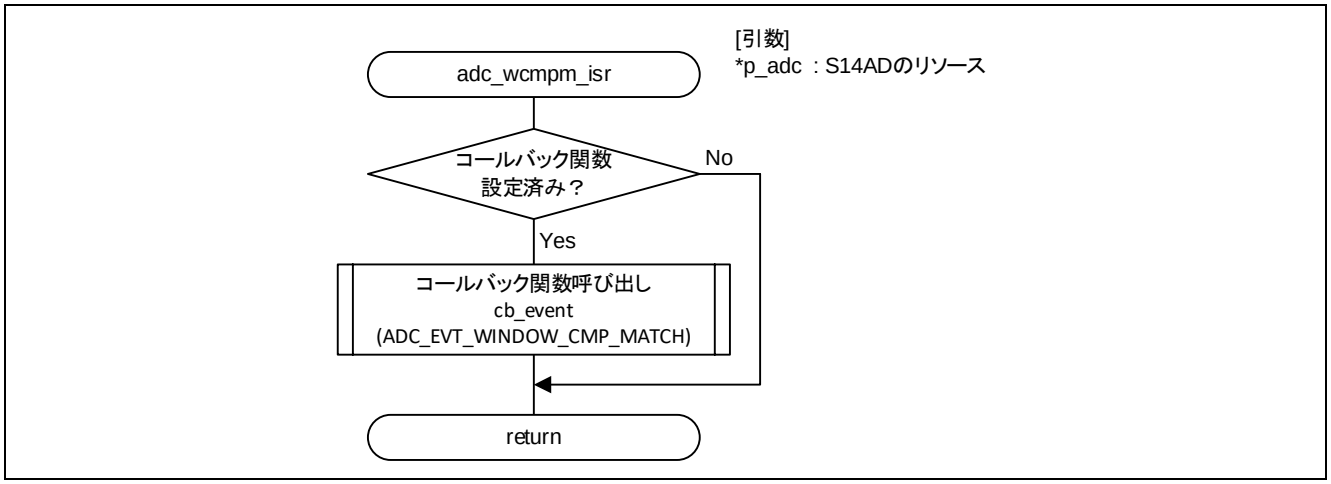


図 4-48 adc_wcmpm_isr 関数処理フロー

4.1.42 adc_wcmpum_isr 関数

表 4-44 adc_wcmpum_isr 関数仕様

書式	static void adc_wcmpum_isr(st_adc_resources_t * const p_adc)
仕様説明	WCMPUM 割り込み処理
引数	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	なし
備考	-

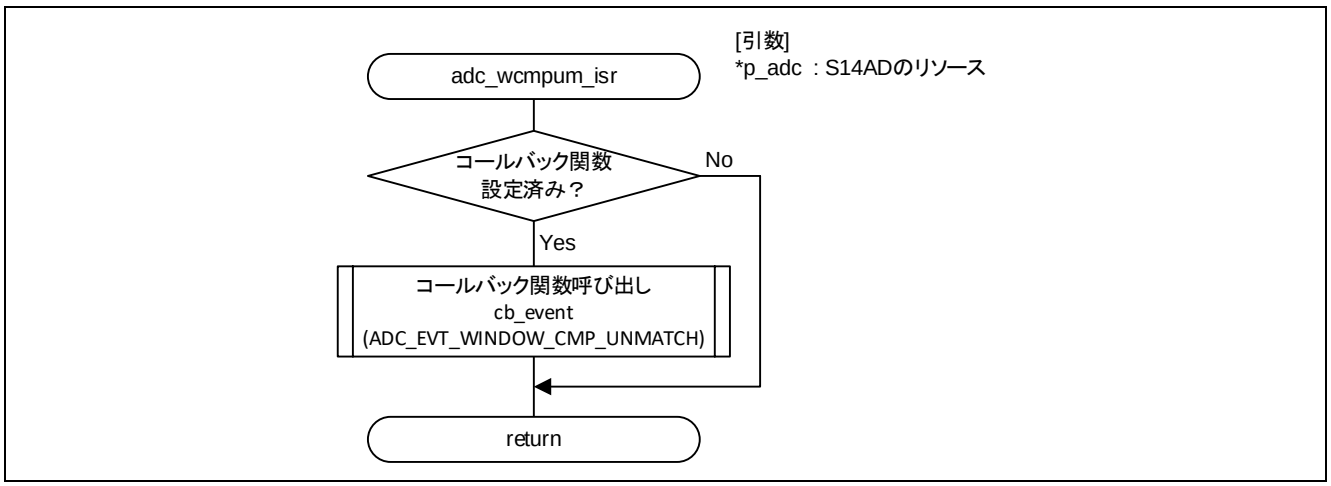


図 4-49 adc_wcmpum_isr 関数処理フロー

4.1.43 adc_cmpai_isr 関数

表 4-45 adc_cmpai_isr 関数仕様

書式	static void adc_cmpai_isr(st_adc_resources_t * const p_adc)
仕様説明	CMPAI 割り込み処理
引数	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	なし
備考	-

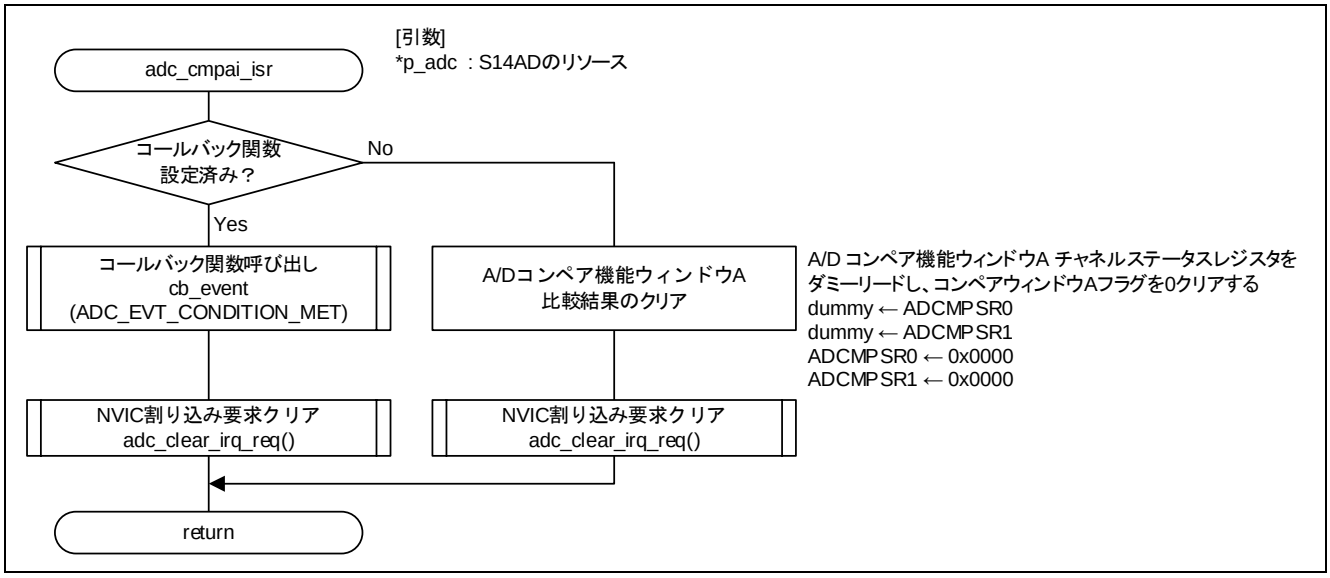


図 4-50 adc_cmpai_isr 関数処理フロー

4.1.44 adc_cmpbi_isr 関数

表 4-46 adc_cmpbi_isr 関数仕様

書式	static void adc_cmpbi_isr(st_adc_resources_t * const p_adc)
仕様説明	CMPBI 割り込み処理
引数	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	なし
備考	-

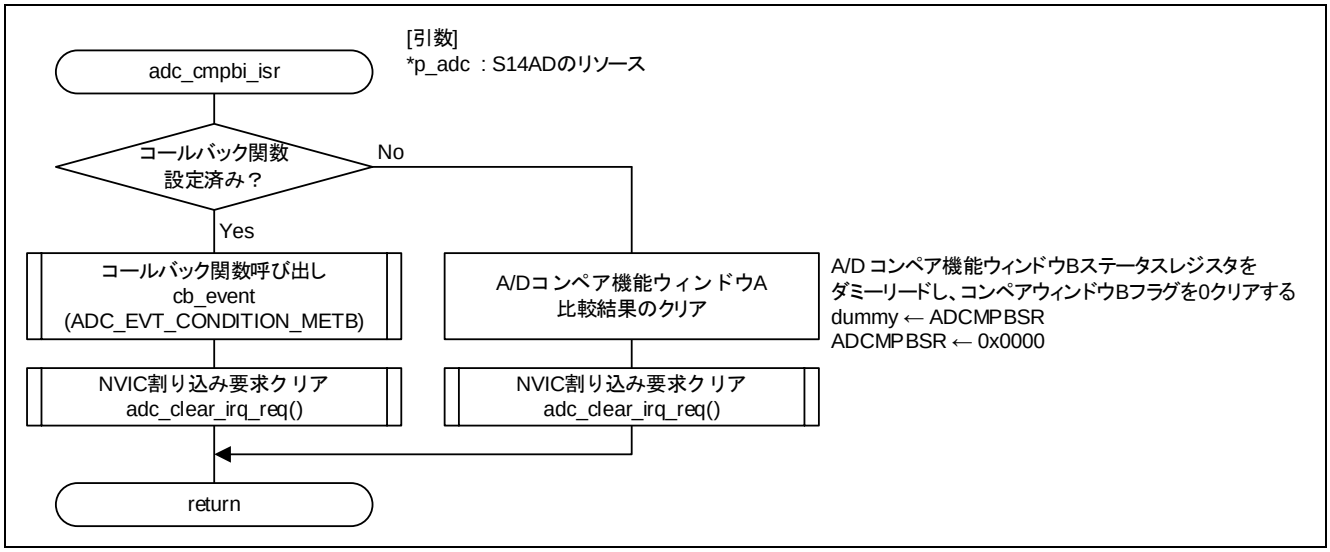


図 4-51 adc_cmpbi_isr 関数処理フロー

4.1.45 adc_dma_adi_isr 関数

表 4-47 adc_dma_adi_isr 関数仕様

書式	static void adc_dma_adi_isr(st_adc_resources_t * const p_adc)
仕様説明	オートリード機能 ADI 割り込み処理
引数	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	なし
備考	r_adc_cfg.h でいずれかの割り込み要求制御に DMAC/DTC を指定していた場合のみ有効

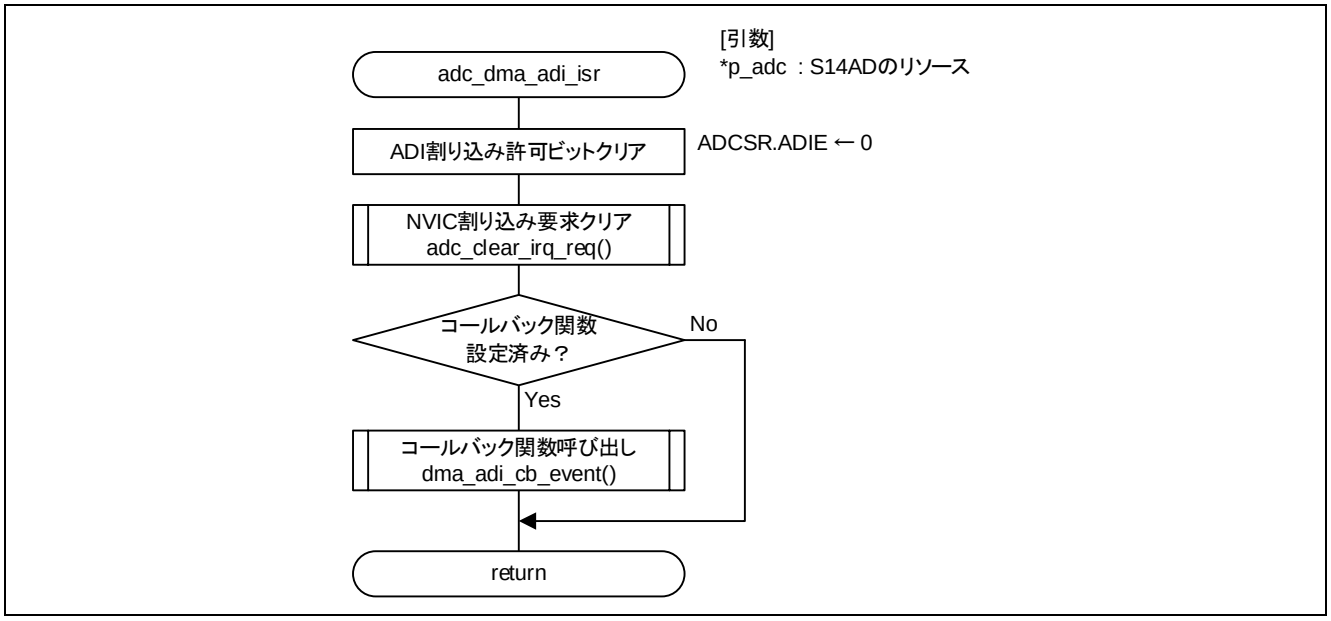


図 4-52 adc_dma_adi_isr 関数処理フロー

4.1.46 adc_dma_gbadi_isr 関数

表 4-48 adc_dma_gbadi_isr 関数仕様

書式	static void adc_dma_gbadi_isr(st_adc_resources_t * const p_adc)
仕様説明	オートリード機能 GBADI 割り込み処理
引数	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	なし
備考	r_adc_cfg.h でいずれかの割り込み要求制御に DMAC/DTC を指定していた場合のみ有効

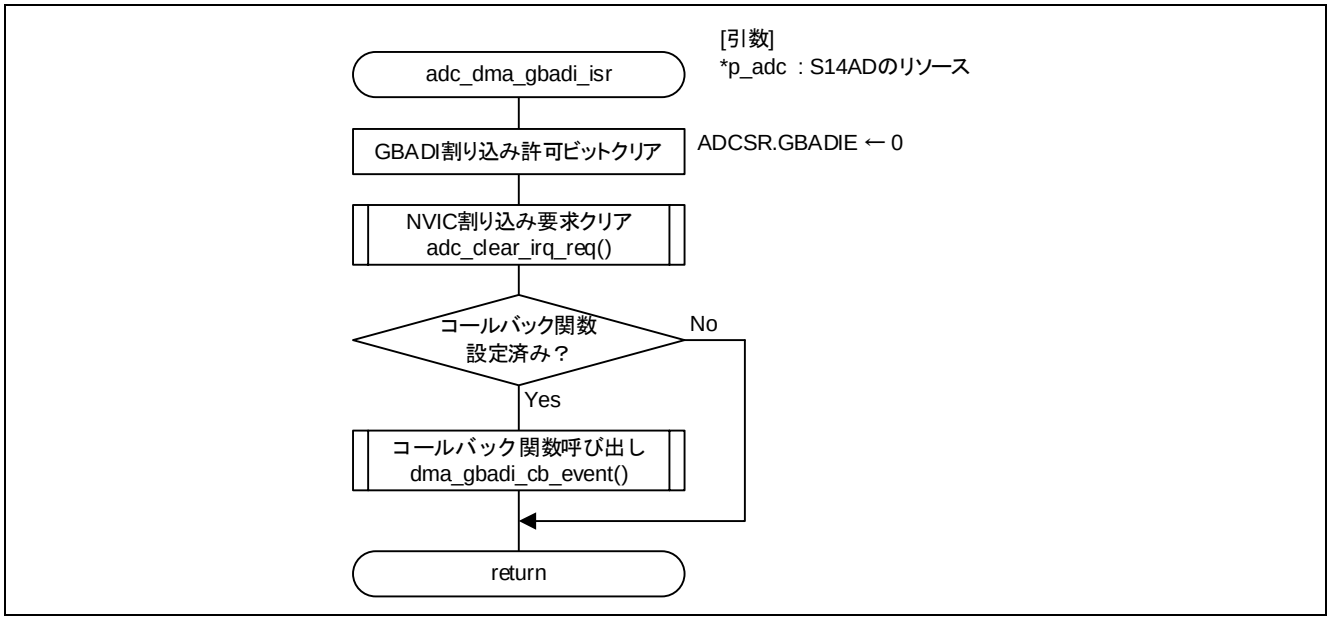


図 4-53 adc_dma_gbadi_isr 関数処理フロー

4.1.47 adc_dma_gcadi_isr 関数

表 4-49 adc_dma_gcadi_isr 関数仕様

書式	static void adc_dma_gcadi_isr(st_adc_resources_t * const p_adc)
仕様説明	オートリード機能 GCADI 割り込み処理
引数	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	なし
備考	r_adc_cfg.h でいずれかの割り込み要求制御に DMAC/DTC を指定していた場合のみ有効

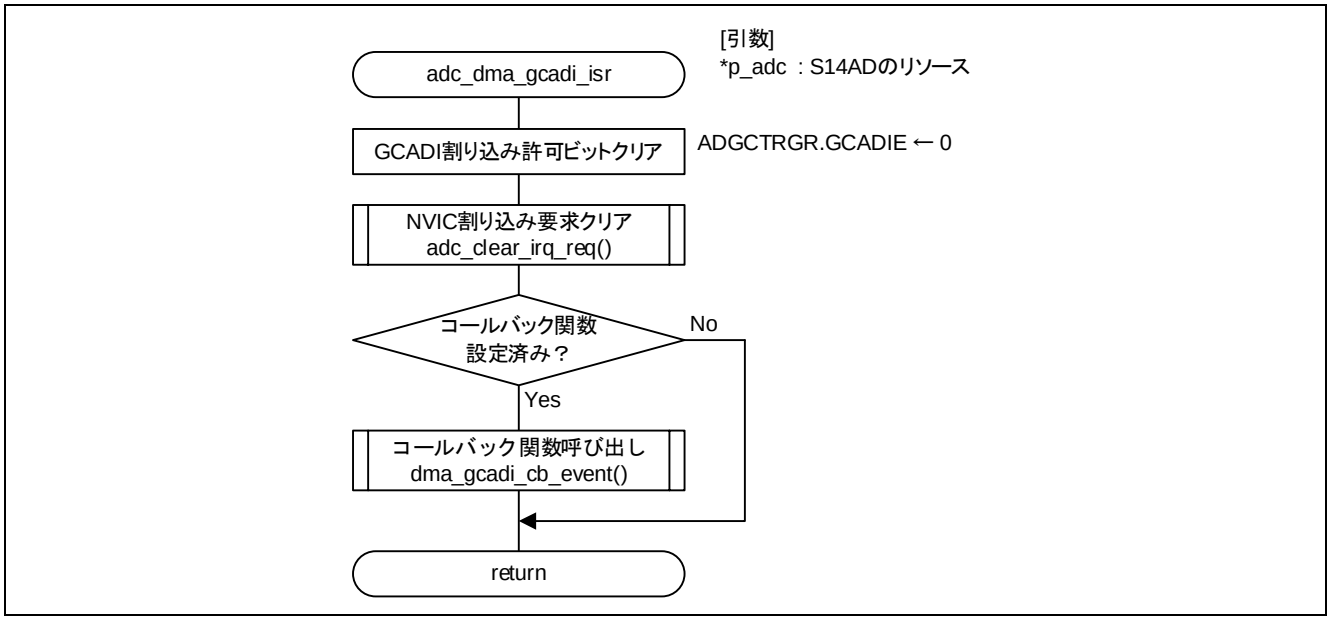


図 4-54 adc_dma_gcadi_isr 関数処理フロー

4.1.48 adc_dma_wcmpm_isr 関数

表 4-50 adc_dma_wcmpm_isr 関数仕様

書式	static void adc_dma_wcmpm_isr(st_adc_resources_t * const p_adc)
仕様説明	オートリード機能 WCMPM 割り込み処理
引数	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	なし
備考	r_adc_cfg.h でいずれかの割り込み要求制御に DMAC/DTC を指定していた場合のみ有効

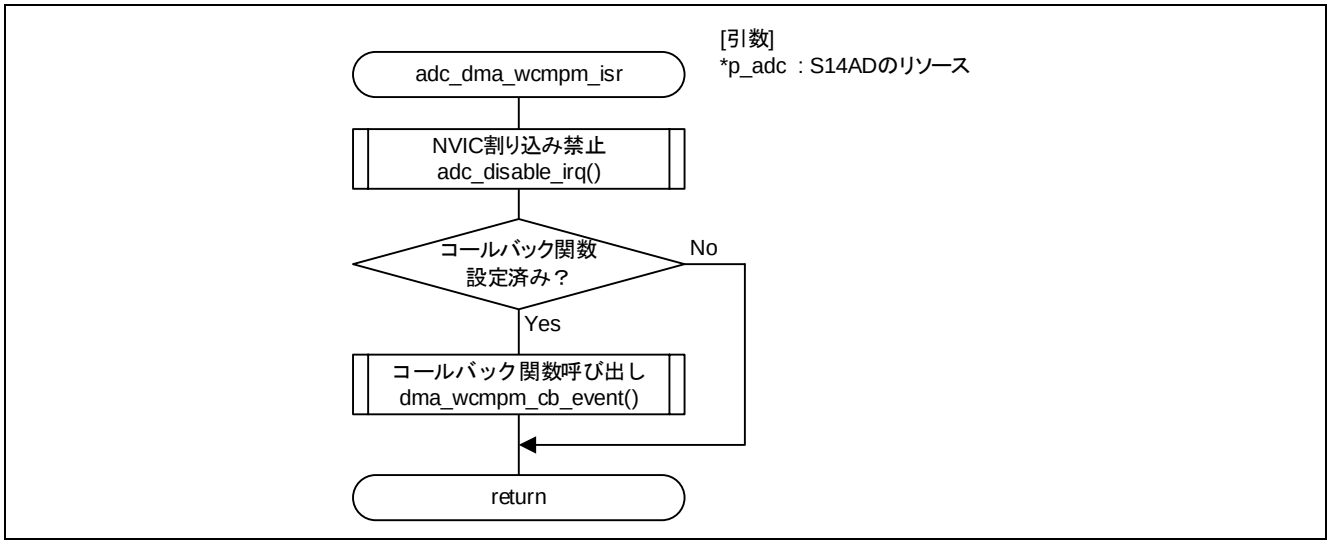


図 4-55 adc_dma_wcmpm_isr 関数処理フロー

4.1.49 adc_dma_wcmpum_isr 関数

表 4-51 adc_dma_wcmpum_isr 関数仕様

書式	static void adc_dma_wcmpum_isr(st_adc_resources_t * const p_adc)
仕様説明	オートリード機能 WCMPUM 割り込み処理
引数	st_adc_resources_t *p_adc：S14AD のリソース S14AD のリソースを指定します
戻り値	なし
備考	r_adc_cfg.h でいずれかの割り込み要求制御に DMAC/DTC を指定していた場合のみ有効

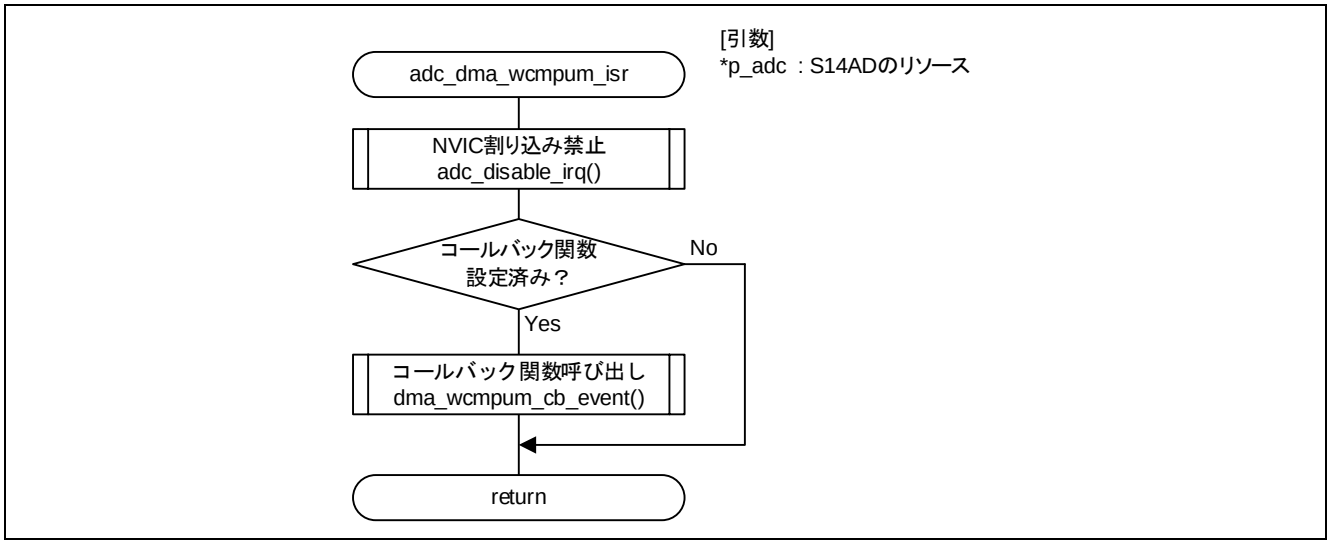


図 4-56 adc_dma_wcmpum_isr 関数処理フロー

4.2 マクロ/型定義

ドライバ内部で使用するマクロ/型定義を示します。

4.2.1 マクロ定義一覧

表 4-52 マクロ定義一覧(1/2)

定義	値	内容
ADC_FLAG_OPENED	(1U << 0)	Open 実行済みフラグ
ADC_FLAG_INITIALIZED	(1U << 1)	S14AD 初期化フラグ
ADC_FLAG_CONFIGURATION	(1U << 2)	S14AD コンフィグレーションフラグ
ADC_FLAG_DMA_ADI_OPEN	(1U << 3)	ADI 割り込み要求の DMA ドライバ Open 済みフラグ
ADC_FLAG_DMA_GBADI_OPEN	(1U << 4)	GBADI 割り込み要求の DMA ドライバ Open 済み フラグ
ADC_FLAG_DMA_GCADI_OPEN	(1U << 5)	GCADI 割り込み要求の DMA ドライバ Open 済み フラグ
ADC_FLAG_DMA_WCMPM_OPEN	(1U << 6)	WCMPM 割り込み要求の DMA ドライバ Open 済み フラグ
ADC_FLAG_DMA_WCMPUM_OPEN	(1U << 7)	WCMPUM 割り込み要求の DMA ドライバ Open 済み フラグ
ADC_FLAG_DMA_POS	(3)	オートリード機能用 DMA ドライバフラグ位置
ADC_MODE_POS	(0)	モード設定位置
ADC_MODE_MASK	(0x3UL << ADC_MODE_POS)	モード設定マスク
ADC_RESOLUTION_POS	(2)	分解能設定位置
ADC_RESOLUTION_MASK	(0x1UL << ADC_RESOLUTION_POS)	分解能設定マスク
ADC_FORMAT_POS	(3)	A/D データレジスタフォーマット設定位置
ADC_FORMAT_MASK	(0x1UL << ADC_FORMAT_POS)	A/D データレジスタフォーマット設定マスク
ADC_DDA_MASK	(0x0F)	自己診断モード設定マスク
ADC_VERSION_MAJOR	(注)	S14AD ドライバメジャーバージョン情報
ADC_VERSION_MINOR	(注)	S14AD ドライバマイナーバージョン情報
ADC_PRV_CHAN_LOW_MASK	1500KB:(0x007F) 256KB:(0x00FF)	A/D チャネル下位側マスク(AN000～AN006)
ADC_PRV_CHAN_HIGH_MASK	1500KB: (0x01FF3) 256KB:(0x08033)	A/D チャネル上位側マスク(AN016、AN017、AN020～AN028)
ADC_PRV_CHAN_ALL_MASK	((ADC_PRV_CHAN_HIGH_MASK << 16) | ADC_PRV_CHAN_LOW_MASK)	全 A/D チャネル用マスク(AN000～AN006、AN016、AN017、AN020～AN028)
ADC_PRV_SENSOR_TEMP	(0x01)	温度センサ出力チャネル 定義
ADC_PRV_TRIGER_TMR_VAL	(0x1D)	TMR トリガ選択用 A/D 変換開始トリガ設定値

注　本ドライバのバージョンに応じた値が設定されています
例）ドライババージョン 1.01 の場合
ADC_VERSION_MAJOR (1)
ADC_VERSION_MINOR (01)

表 4-53 マクロ定義一覧(2/2)

定義	値	内容
ADC_PRV_TRIGER_ELC_VAL	(0x30)	ELC トリガ選択用 A/D 変換開始トリガ設定値
ADC_PRV_TRIGER_ADTRG_VAL	(0x00)	ADTRG トリガ選択用 A/D 変換開始トリガ設定値
ADC_PRV_TRIGER_DISABLE_VAL	(0x3F)	トリガ非選択用 A/D 変換開始トリガ設定値
ADC_PRV_SENSOR_MASK	(0x01)	温度センサ出力チャネル用マスク
ADC_PRV_ADI_EVENT_USED	(1 << ADC_READ_ADI)	排他制御用 ADI 割り込み要求使用済みフラグ
ADC_PRV_GBADI_EVENT_USED	(1 << ADC_READ_GBADI)	排他制御用 GBADI 割り込み要求使用済みフラグ
ADC_PRV_GCADI_EVENT_USED	(1 << ADC_READ_GCADI)	排他制御用 GCADI 割り込み要求使用済みフラグ
ADC_PRV_WCMPM_EVENT_USED	(1 << ADC_READ_WCMPM)	排他制御用 WCMPM 割り込み要求使用済みフラグ
ADC_PRV_WCMPUM_EVENT_USED	(1 << ADC_READ_WCMPUM)	排他制御用 WCMPUM 割り込み要求使用済みフラグ
ADC_PRV_ADI_DMAC_SOURCE_ID	(0x23)	ADI 用 DELS ビット設定値
ADC_PRV_GBADI_DMAC_SOURCE_ID	(0x24)	GBADI 用 DELS ビット設定値
ADC_PRV_GCADI_DMAC_SOURCE_ID	(0x29)	GCADI 用 DELS ビット設定値
ADC_PRV_WCMPM_DMAC_SOURCE_ID	(0x27)	WCMPM 用 DELS ビット設定値
ADC_PRV_WCMPUM_DMAC_SOURCE_ID	(0x28)	WCMPUM 用 DELS ビット設定値
ADC_PRV_USED_DMAC_DRV	(0x00FF & (S14AD_ADI_CONTROL | S14AD_GBADI_CONTROL | S14AD_GCADI_CONTROL | S14AD_WCMPM_CONTROL | S14AD_WCMPUM_CONTROL))	DMAC ドライバ使用判定定義
ADC_PRV_USED_DTC_DRV	(S14AD_USED_DTC & (S14AD_ADI_CONTROL | S14AD_GBADI_CONTROL | S14AD_GCADI_CONTROL | S14AD_WCMPM_CONTROL | S14AD_WCMPUM_CONTROL))	DTC ドライバ使用判定定義
ADC_PRV_USED_DTC_DMAC_DRV	(ADC_PRV_USED_DMAC_DRV | ADC_PRV_USED_DTC_DRV)	DMAC/DTC ドライバ使用判定定義
ADC_PRV_WCMP_USED_DMAC_DRV	(0x00FF & (S14AD_WCMPM_CONTROL | S14AD_WCMPUM_CONTROL))	ウィンドウコンペア用 DMAC ドライバ使用定義
ADC_PRV_WCMP_USED_DTC_DRV	(S14AD_USED_DTC & (S14AD_WCMPM_CONTROL | S14AD_WCMPUM_CONTROL))	ウィンドウコンペア用 DTC ドライバ使用定義
ADC_PRV_WCMP_USED_DTC_DMAC_DRV	(ADC_PRV_WCMP_USED_DMAC_DRV | ADC_PRV_WCMP_USED_DTC_DRV)	ウィンドウコンペア用 DMAC/DTC ドライバ使用定義

4.2.2 e_adc_sst 定義

AD_CMD_SET_SAMPLING_ANn(注)、AD_CMD_SET_SAMPLING_AN016_AN017_AN020_TO_AN028、AD_CMD_SET_SAMPLING_TEMP コマンドで使用するサンプリング時間変更対象定義です。

注　1500KB グループ：n = 0～6、L、T　　256KB グループ：n = 0～7、L、T

表 4-54 e_adc_sst 定義一覧(1500KB グループ)

定義	値	内容
AD_SST_AN000	0	サンプリングステートレジスタ(ADSSTR0)を変更
AD_SST_AN001	1	サンプリングステートレジスタ(ADSSTR1)を変更
AD_SST_AN002	2	サンプリングステートレジスタ(ADSSTR2)を変更
AD_SST_AN003	3	サンプリングステートレジスタ(ADSSTR3)を変更
AD_SST_AN004	4	サンプリングステートレジスタ(ADSSTR4)を変更
AD_SST_AN005	5	サンプリングステートレジスタ(ADSSTR5)を変更
AD_SST_AN006	6	サンプリングステートレジスタ(ADSSTR6)を変更
AD_SST_AN016_AN017_AN020_TO_AN028	7	サンプリングステートレジスタ(ADSSTRL)を変更
AD_SST_TEMP	8	サンプリングステートレジスタ(ADSSTRT)を変更

表 4-55 e_adc_sst 定義一覧(256KB グループ)

定義	値	内容
AD_SST_AN000	0	サンプリングステートレジスタ(ADSSTR0)を変更
AD_SST_AN001	1	サンプリングステートレジスタ(ADSSTR1)を変更
AD_SST_AN002	2	サンプリングステートレジスタ(ADSSTR2)を変更
AD_SST_AN003	3	サンプリングステートレジスタ(ADSSTR3)を変更
AD_SST_AN004	4	サンプリングステートレジスタ(ADSSTR4)を変更
AD_SST_AN005	5	サンプリングステートレジスタ(ADSSTR5)を変更
AD_SST_AN006	6	サンプリングステートレジスタ(ADSSTR6)を変更
AD_SST_AN007	7	サンプリングステートレジスタ(ADSSTR7)を変更
AD_SST_AN016_AN017_AN020_AN021_VSC_VCC	8	サンプリングステートレジスタ(ADSSTRL)を変更
AD_SST_TEMP	9	サンプリングステートレジスタ(ADSSTRT)を変更

4.2.3 e_adc_dma_cmd_t 定義

オートリード機能用のモード定義です。

表 4-56 e_adc_sst 定義一覧

定義	値	内容
ADC_READ_NORMAL	0	オートリード機能ノーマル転送モード
ADC_READ_BLOCK	1	オートリード機能ブロック転送モード
ADC_READ_COMPARE	2	オートリード機能コンペアモード

4.3 構造体定義

4.3.1 st_adc_resources_t 構造体

S14AD のリソースを構成する構造体です。

表 4-57 st_adc_resources_t 構造体(1/3)

要素名	型	内容
*reg	volatile S14AD_Type	S14AD レジスタを示します
pin_set	r_pinset_t	端子設定用関数ポインタ
pin_clr	r_pinclr_t	端子解除用関数ポインタ
*adc_info	st_adc_mode_info_t	S14AD 状態情報
*dma_info	st_adc_dma_isr_info_t	オートリード機能用 DMA 状態情報
lock_id	e_system_mcu_lock_t	S14AD ロック ID
mstp_id	e_lpm_mstp_t	S14AD モジュールストップ ID
adi_irq	IRQn_Type	ADI 割り込みの NVIC 割り当て番号
gbadi_irq	IRQn_Type	GBADI 割り込みの NVIC 割り当て番号
gcadi_irq	IRQn_Type	GCADI 割り込みの NVIC 割り当て番号
cmpai_irq	IRQn_Type	CMPAI 割り込みの NVIC 割り当て番号
cmpbi_irq	IRQn_Type	CMPBI 割り込みの NVIC 割り当て番号
wcmpm_irq	IRQn_Type	WCMPM 割り込みの NVIC 割り当て番号
wcmpum_irq	IRQn_Type	WCMPUM 割り込みの NVIC 割り当て番号
adi_iesr_val	uint32_t	ADI 割り込みの IESR レジスタ設定値
gbadi_iesr_val	uint32_t	GBADI 割り込みの IESR レジスタ設定値
gcadi_iesr_val	uint32_t	GCADI 割り込みの IESR レジスタ設定値
cmpai_iesr_val	uint32_t	CMPAI 割り込みの IESR レジスタ設定値
cmpbi_iesr_val	uint32_t	CMPBI 割り込みの IESR レジスタ設定値
wcmpm_iesr_val	uint32_t	WCMPM 割り込みの IESR レジスタ設定値
wcmpum_iesr_val	uint32_t	WCMPUM 割り込みの IESR レジスタ設定値
adi_priority	uint32_t	ADI 割り込み優先レベル
gbadi_priority	uint32_t	GBADI 割り込み優先レベル
gcadi_priority	uint32_t	GCADI 割り込み優先レベル
cmpai_priority	uint32_t	CMPAI 割り込み優先レベル
cmpbi_priority	uint32_t	CMPBI 割り込み優先レベル
wcmpm_priority	uint32_t	WCMPM 割り込み優先レベル
wcmpum_priority	uint32_t	WCMPUM 割り込み優先レベル

表 4-58 st_adc_resources_t 構造体(2/3)

要素名	型	内容
*adi_dma_drv	DRIVER_DMA	ADI 割り込み要求用 DMA ドライバ ADI 割り込み要求を割り込みで使用する場合は NULL が設定されます
adi_dma_source	uint16_t	ADI 用 DELS ビット設定値
*adi_dtc_info	st_dma_transfer_data_t	ADI 用 DTC 転送情報格納番地
*gbadi_dma_drv	DRIVER_DMA	GBADI 割り込み要求用 DMA ドライバ GBADI 割り込み要求を割り込みで使用する場合は NULL が設定されます
gbadi_dma_source	uint16_t	GBADI 用 DELS ビット設定値
*gbadi_dtc_info	st_dma_transfer_data_t	GBADI 用 DTC 転送情報格納番地
*gcadi_dma_drv	DRIVER_DMA	GCADI 割り込み要求用 DMA ドライバ GCADI 割り込み要求を割り込みで使用する場合は NULL が設定されます
gcadi_dma_source	uint16_t	GCADI 用 DELS ビット設定値
*gcadi_dtc_info	st_dma_transfer_data_t	GCADI 用 DTC 転送情報格納番地
*wcmpm_dma_drv	DRIVER_DMA	WCMPM 割り込み要求用 DMA ドライバ WCMPM 割り込み要求を割り込みで使用する場合は NULL が設定されます
wcmpm_dma_source	uint16_t	WCMPM 用 DELS ビット設定値
*wcmpm_dtc_info	st_dma_transfer_data_t	WCMPM 用 DTC 転送情報格納番地
*wcmpum_dma_drv	DRIVER_DMA	WCMPUM 割り込み要求用 DMA ドライバ WCMPUM 割り込み要求を割り込みで使用する場合は NULL が設定されます
wcmpum_dma_source	uint16_t	WCMPUM 用 DELS ビット設定値
*wcmpum_dtc_info	st_dma_transfer_data_t	WCMPUM 用 DTC 転送情報格納番地
*dma_restart_data	st_adc_dma_restart	オートリード機能用 DMA リスタート情報
adi_callback	system_int_cb_t	ADI コールバック関数
gbadi_callback	system_int_cb_t	GBADI コールバック関数
gcadi_callback	system_int_cb_t	GCADI コールバック関数
cmpai_callback	system_int_cb_t	CMPAI コールバック関数
cmpbi_callback	system_int_cb_t	CMPBI コールバック関数
wcmpm_callback	system_int_cb_t	WCMPM コールバック関数
wcmpum_callback	system_int_cb_t	WCMPUM コールバック関数

表 4-59 st_adc_resources_t 構造体(3/3)

要素名	型	内容
dma_adi_callback	system_int_cb_t	オートリード機能用 ADI コールバック関数
dma_gbadi_callback	system_int_cb_t	オートリード機能用 GBADI コールバック関数
dma_gcadi_callback	system_int_cb_t	オートリード機能用 GCADI コールバック関数
dma_wcmpm_callback	system_int_cb_t	オートリード機能用 WCMPM コールバック関数
dma_wcmpum_callback	system_int_cb_t	オートリード機能用 WCMPUM コールバック関数

4.3.2 st_adc_dma_set 構造体

オートリード機能で設定する DMA 設定定義です。

表 4-60 st_adc_dma_set 構造体

要素名	型	内容
flg	uint8_t	DMA 転送要因排他制御用フラグ設定値
*dma_drv	DRIVER_DMA	DMA 転送要因 DMA ドライバ
dma_source	uint16_t	DMA 転送要因 DELS ビット設定値
irq	IRQn_Type	DMA 転送要因 NVIC 割り当て番号
iesr	uint32_t	DMA 転送要因 IESR 設定値
priority	uint32_t	DMA 転送要因割り込み優先レベル
*transfer_info	st_dma_transfer_data_t	DMA 転送要因 DTC 転送情報格納番地

4.4 外部関数の呼び出し

S14AD ドライバ API から呼び出される外部関数を示します。

表 4-61 S14AD ドライバ API から呼び出す外部関数と呼び出し条件(1/2)

API	呼び出し関数	条件（注）
Open	R_SYS_ResourceLock	なし
	R_LPM_ModuleStart	なし
	R_SYS_ResourceUnlock	モジュールストップ解除状態で Open 関数実行時
	R_S14AD_Pinset	なし
Close	R_S14AD_Pinclr	なし
	R_SYS_ResourceUnlock	なし
	R_NVIC_DisableIRQ	なし
	R_LPM_ModuleStop	なし
	R_DMAC_Close	割り込み要求制御に DMAC を指定していた場合
	R_DTC_Close	割り込み要求制御に DTC を指定していた場合
	R_DTC_Release	
	R_DTC_GetAvailabilitySrc	
ScanSet	-	-
Start	-	-
Stop	-	-
Control	R_NVIC_DisableIRQ	以下のコマンドを実行した場合 AD_CMD_SET_ADI_INT AD_CMD_SET_GBADI_INT AD_CMD_SET_GCADI_INT AD_CMD_SET_WCMPM_INT AD_CMD_SET_WCMPUM_INT AD_CMD_SET_WINDOWA AD_CMD_SET_WINDOWB
	R_SYS_IrqStatusClear	
	R_NVIC_ClearPendingIRQ	
	R_NVIC_EnableIRQ	
	R_SYS_IrqStatusGet	AD_CMD_GET_AD_STATE を実行した場合
	R_SYS_PeripheralClockFreqGet	AD_CMD_SCLK_ENABLE を実行した場合
	R_DMAC_Open	割り込み要求制御に DMAC を指定し、以下のコマンドを実行した場合 AD_CMD_AUTO_READ_NORMAL AD_CMD_AUTO_READ_BLOCK AD_CMD_AUTO_READ_COMPARE
	R_DTC_Open	割り込み要求制御に DTC を指定し、以下のコマンドを実行した場合 AD_CMD_AUTO_READ_NORMAL AD_CMD_AUTO_READ_BLOCK AD_CMD_AUTO_READ_COMPARE
	R_DMAC_Create	割り込み要求制御に DMAC を指定し、以下のコマンドを実行した場合 AD_CMD_AUTO_READ_NORMAL AD_CMD_AUTO_READ_BLOCK AD_CMD_AUTO_READ_COMPARE
	R_DTC_Create	割り込み要求制御に DTC を指定し、以下のコマンドを実行した場合 AD_CMD_AUTO_READ_NORMAL AD_CMD_AUTO_READ_BLOCK AD_CMD_AUTO_READ_COMPARE

注　条件なしの場合でも、パラメータチェックによるエラー終了発生時には呼び出し関数が実行されない可能性があります。

表 4-62 S14AD ドライバ API から呼び出す外部関数と呼び出し条件(2/2)

API	呼び出し関数	条件（注）
	R_DMAC_Control	割り込み要求制御に DMAC を指定し、以下のコマンドを実行した場合 AD_CMD_AUTO_READ_NORMAL AD_CMD_AUTO_READ_BLOCK AD_CMD_AUTO_READ_COMPARE AD_CMD_AUTO_READ_STOP AD_CMD_AUTO_READ_RESTART AD_CMD_STOP_TRIG
	R_DTC_Control	割り込み要求制御に DTC を指定し、以下のコマンドを実行した場合 AD_CMD_AUTO_READ_NORMAL AD_CMD_AUTO_READ_BLOCK AD_CMD_AUTO_READ_COMPARE AD_CMD_AUTO_READ_STOP AD_CMD_AUTO_READ_RESTART AD_CMD_STOP_TRIG
	R_DMAC_ InterruptEnable	割り込み要求制御に DMAC を指定し、以下のコマンドを実行した場合 AD_CMD_AUTO_READ_NORMAL AD_CMD_AUTO_READ_BLOCK AD_CMD_AUTO_READ_COMPARE
Read	-	-
GetVersion	-	-

注　条件なしの場合でも、パラメータチェックによるエラー終了発生時には呼び出し関数が実行されない可能性があります。

5. 使用上の注意

5.1 引数について

各関数の引数で指定する構造体の内、使用しない要素には 0 を設定してください。

5.2 端子設定について

本ドライバで使用する端子は、pin.c の R_S14AD_Pinset 関数で設定、R_S14AD_Pinclr 関数で解放されます。R_S14AD_Pinset 関数は Open 関数から、R_S14AD_Pinclr 関数は Close 関数から呼び出されます。

使用端子の設定方法は、「図 2-11 端子設定例(1/2)」「図 2-12 端子設定例(2/2)」をご確認ください。

5.3 A/D 変換開始条件について

A/D 変換は、A/D 変換開始トリガ入力の検出によって実行されます。各設定における A/D 変換開始トリガを以下に示します。A/D 変換開始トリガに同期トリガを使用する場合、トリガ要因に応じて 8 ビットタイマ(TMR)もしくはイベントリンクコントローラ(ELC)の設定を行ってください。

ソフトウェアトリガ：Start 関数実行

外部トリガ：ADTRG0 端子入力

同期トリガ（TMR）：TMR0_TCORA（TMR コンペアマッチ A）イベント検出

同期トリガ（ELC）：ELC_S14AD イベント検出

5.4 S14AD 機能組み合わせおよび各機能の制限事項について

S14AD の各機能には、組み合わせて使用できない機能があります。各機能の組み合わせについては「表 2-12 使用機能組み合わせ一覧」を、機能設定時の制限事項については「2.5 Control 関数による S14AD 機能設定」の各制御コマンドの詳細をご確認ください。

5.5 電源オープン制御レジスタ(VOCR)設定について

本ドライバは、電源オープン制御レジスタ（VOCR）の設定を行った上で使用してください。

VOCR レジスタは、電源供給されていない電源ドメインから不定な入力が入ることを阻止するレジスタです。このため、VOCR レジスタはリセット後、入力信号を遮断する設定になっています。この状態では入力信号がデバイス内部に伝搬されません。詳細は「RE01 1500KB、256KB グループ CMSIS パッケージを用いた開発スタートアップガイド（r01an4660）」の「IO 電源ドメイン不定値伝搬抑止制御」を参照してください。

5.6 NVIC への割り込み登録について

S14AD ドライバにて A/D 割り込み要因を割り込み、もしくはポーリングで使用する場合は r_system_cfg.h にて NVIC への登録を行ったうえ、Control 関数にて割り込みを許可にしてください。詳細は、

「RE01 1500KB、256KB グループ CMSIS パッケージを用いた開発スタートアップガイド（r01an4660）」の「割り込み制御」の設定を参照してください。

NVIC に S14AD 割り込みが登録されていない場合、Control 関数割り込み許可コマンド実行時に ADC_ERROR_SYSTEM_SETTING が戻ります。

使用用途に対する NVIC の登録定義を**表 5-1** に、NVIC への割り込み登録例を図 5-1 に示します。

表 5-1 使用用途に対する NVIC の登録定義

使用用途	NVIC 登録定義	備考
ADI 割り込み使用時	SYSTEM_CFG_EVENT_NUMBER_ADC140_ADI	
GBADI 割り込み使用時	SYSTEM_CFG_EVENT_NUMBER_ADC140_GBADI	
GCADI 割り込み使用時	SYSTEM_CFG_EVENT_NUMBER_ADC140_GCADI	
CMPAI 割り込み使用時	SYSTEM_CFG_EVENT_NUMBER_ADC140_CMPAI	
CMPBI 割り込み使用時	SYSTEM_CFG_EVENT_NUMBER_ADC140_CMPBI	
WCMPM 割り込み使用時	SYSTEM_CFG_EVENT_NUMBER_ADC140_WCMPM	
WCMPUM 割り込み使用時	SYSTEM_CFG_EVENT_NUMBER_ADC140_WCMPUM	
オートリードコマンド(注 1) 使用時（DMAC 使用）	SYSTEM_CFG_EVENT_NUMBER_DMACm_INT	m=0～3 (注 2)
オートリードコマンド(注 1) 使用時（DTC 使用）	(注 3)	

注1. オートリードコマンドは、以下のいずれかのコマンドを指します。
- AD_CMD_AUTO_READ_NORMAL
- AD_CMD_AUTO_READ_BLOCK
- AD_CMD_AUTO_READ_COMPARE

注2. コールバック関数に NULL を設定した場合（コールバック未使用）、設定は不要です。

注3. オートリード対象の要因に対応した NVIC 登録を行ってください。
要因に ADI を使用した場合：SYSTEM_CFG_EVENT_NUMBER_ADC140_ADI
要因に GBADI を使用した場合：SYSTEM_CFG_EVENT_NUMBER_ADC140_GBADI
要因に GCADI を使用した場合：SYSTEM_CFG_EVENT_NUMBER_ADC140_GCADI
要因に WCMPM を使用した場合：SYSTEM_CFG_EVENT_NUMBER_ADC140_WCMPM
要因に WCMPUM を使用した場合：SYSTEM_CFG_EVENT_NUMBER_ADC140_WCMPUM

```
・・・
#define SYSTEM_CFG_EVENT_NUMBER_WDT_NMIUNDF
           (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)  /*!< Numbers 0/4/8/12/16/20/24/28 only */
#define SYSTEM_CFG_EVENT_NUMBER_ADC140_ADI
           (SYSTEM_IRQ_EVENT_NUMBER0)            /*!< Numbers 0/4/8/12/16/20/24/28 only */
#define SYSTEM_CFG_EVENT_NUMBER_ADC140_WCMPM
      (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)  /*!< Numbers 0/4/8/12/16/20/24/28 only */
・・・
```

図 5-1 r_system_cfg.h での NVIC への割り込み登録例（ADI を使用）

5.7 割り込み機能とオートリード機能の排他制御について

割り込み機能とオートリード機能は排他的に制御されます。オートリード機能で DMA 起動要因に指定した割り込み要求は、割り込みおよびポーリングで使用できません。DMA 起動要因に指定した割り込み要求は、Close 関数を実行することで解放されます。DMA 起動要因に指定した割り込み要求を割り込み機能で使用する場合、Close 関数および S14AD の初期化を実行後、割り込み機能設定を行う必要があります。

5.8 コールバック関数のイベント判定について

各イベント発生時、Open 関数で指定したコールバック関数が呼び出されます。どのイベントから呼び出されたかは、引数にて判定してください。

各イベント発生時に実行されるコールバック関数を表 5-2 に示します。Open 関数で指定したコールバック関数でのイベント判定例を図 5-2 に示します。

表 5-2 各イベント発生時に実行されるコールバック一覧

イベント	コールバック関数	イベントコード
ADC140_ADI	Open 関数で指定したコールバック関数	ADC_EVT_SCAN_COMPLETE
ADC140_GBADI		ADC_EVT_SCAN_COMPLETE_GROUPB
ADC140_GCADI		ADC_EVT_SCAN_COMPLETE_GROUPC
ADC140_CMPAI		ADC_EVT_CONDITION_MET
ADC140_CMPBI		ADC_EVT_CONDITION_METB
ADC140_WCMPM		ADC_EVT_WINDOW_CMP_MATCH
ADC140_WCMPUM		ADC_EVT_WINDOW_CMP_UNMATCH
オートリード機能で すべての転送が完了したとき	[DMAC] DMACm_INT のコールバック関数 (m=0～3) [DTC] DTC 転送要因に指定した割り込みのコールバック関数	-

```
/***********************************************************************************************************
* callback function
***********************************************************************************************************/
static void callback(uint32_t event)
{
    switch(event)
    {
        case ADC_EVT_SCAN_COMPLETE:
        {
            /* A/D 変換が終了
              （グループスキャンモードの場合、グループ A スキャン終了）した場合の処理を記述 */
        }
        break;

        case ADC_EVT_SCAN_COMPLETE_GROUPB:
        {
            /* グループ B の A/D 変換が終了した場合の処理を記述 */
        }
        break;

        case ADC_EVT_SCAN_COMPLETE_GROUPC:
        {
            /* グループ C の A/D 変換が終了した場合の処理を記述 */
        }
        break;

        case ADC_EVT_CONDITION_MET:
        {
            /* コンペア A が一致した場合の処理を記述 */
        }
        break;

        case ADC_EVT_CONDITION_METB
        {
            /* コンペア B が一致した場合の処理を記述 */
        }
        break;

        case ADC_EVT_WINDOW_CMP_MATCH:
        {
            /* ウィンドウ A/B のコンペア条件に一致した場合の処理を記述 */
        }
        break;

        case ADC_EVT_WINDOW_CMP_UNMATCH:
        {
            /* ウィンドウ A/B のコンペア条件に不一致した場合の処理を記述 */
        }
        break;
        default:
        {
        }
        break;
    }
}
```

図 5-2 Open 関数で指定したコールバック関数でのイベント判定例

5.9 DTC 使用時の注意事項

本ドライバの r_adc_cfg.h ファイルにて送信制御もしくは受信制御に DTC を選択した場合、ADC ドライバ内で DTC ドライバを使用します。ユーザプログラムにて DTC ドライバを併用する場合、R_DTC_Close 関数を実行するとすべての DTC 転送が停止します。R_DTC_Close 関数が実行されると、ADC で使用している DTC 起動要因が解放されるため、オートリード動作が停止します。

6. 参考ドキュメント

ユーザーズマニュアル：ハードウェア

RE01 1500KB グループ ユーザーズマニュアル　ハードウェア編 R01UH0796
RE01 256KB グループ ユーザーズマニュアル　ハードウェア編 R01UH0894

（最新版をルネサス エレクトロニクスホームページから入手してください。）

RE01 グループ CMSIS Package スタートアップガイド

RE01 1500KB、256KB グループ CMSIS パッケージを用いた開発スタートアップガイド R01AN4660

（最新版をルネサス エレクトロニクスホームページから入手してください。）

テクニカルアップデート／テクニカルニュース

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

ユーザーズマニュアル：開発環境

（最新版をルネサス エレクトロニクスホームページから入手してください。）

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Oct.10.2019	ー	初版
1.01	Oct.31.2019	プログラム 215 17,18,222	adc_dma_config 関数の DMACn ドライバ使用判定式を修正 バージョン定義に関する注記を追加 pin.c のデフォルト端子設定コメントアウト化にともなう修正
1.02	Dec.16.2019	ー プログラム (256KB)	256KB グループに対応 256KB グループの仕様を以下に示す ・周辺機能名（ADC14） ・レジスタ名（ADC140） ・アナログ入力端子　最大 12 チャネル (AN000~AN007(高精度), AN016~AN017(標準精度) ,AN020~AN021(標準精度)) ・VSC_VCC 端子電圧出力変換機能 ・オフセットキャリブレーション機能設定 - 加算回数 16 回
1.03	Apr.17.2020	プログラム	DMAC および DTC ドライバがない状態でビルドできる形に構成を変更
1.04	Jul.28.2020	ー	誤記修正
1.05	Nov.05.2020	225 ー	"5.9 DTC 使用時の注意"を追加 誤記修正
1.06	Jan.11.2021	138 プログラム	R_ADC_Close 関数のフローをプログラムに合わせて更新 DTC 使用時の DTC ドライバ Close 処理を以下のように修正 - ADC ドライバで使用している起動要因を解放 - 他に有効な DTC 起動要因がない場合のみ DTC を Close

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策

   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。

2. 電源投入時の処置

   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。

3. 電源オフ時における入力信号

   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。

4. 未使用端子の処理

   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。

5. クロックについて

   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。

6. 入力端子の印加波形

   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。

7. リザーブアドレス（予約領域）のアクセス禁止

   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。

8. 製品間の相違について

   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。回路、ソフトウェアおよびこれらに関連する情報を使用する場合、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

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

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

(Rev.5.0-1　2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

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

www.renesas.com/contact/

商標について

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