RE01 1500KB, 256KB Group

CMSIS Driver R_S14AD Specifications

Summary

This document describes the detailed specifications of the S14AD driver provided in the RE CMSIS software package (hereinafter referred to as the S14AD driver).

Target Device

RE01 1500KB Group
RE01 256KB Group

Contents

1. Overview 6

2. Driver Configuration 6
2.1 File Configuration 6
2.2 Driver API 7
2.3 Pin Configuration.. 17
2.4 Initialization of S14AD 19
2.4.1 Setting A/D Scan Mode by Open Function 19
2.4.2 Setting A/D Input Pins and A/D Conversion Start Trigger by ScanSet Function 20
2.5 Setting S14AD Features by Control Function 21
2.5.1 Addition/Average Mode Setting (AD_CMD_SET_ADD_MODE) 23
2.5.2 Double-Trigger Mode Setting (AD_CMD_SET_DBLTRG) 24
2.5.3 Self-Diagnosis Mode Setting (AD_CMD_SET_DIAG) 25
2.5.4 ADI Interrupt Function Setting (AD_CMD_SET_ADI_INT) 27
2.5.5 GBADI Interrupt Function Setting (AD_CMD_SET_GROUPB_INT) 29
2.5.6 GCADI Interrupt Function Setting (AD_CMD_SET_GROUPC_INT)................................. 31
2.5.7 WCMPM Interrupt Enable (AD_CMD_SET_WCMPM_INT) 33
2.5.8 WCMPUM Interrupt Enable (AD_CMD_SET_WCMPUM_INT)... 36
2.5.9 A/D Data Register Auto-Clear (AD_CMD_SET_AUTO_CLEAR)...................................... 39
2.5.10 Sampling Time Setting (AD_CMD_SET_SAMPLING_XXX) 39
2.5.11 Disconnection Detection Assist Setting (AD_CMD_SET_ADNDIS).................................. 41
2.5.12 Group Priority Operation Setting (AD_CMD_SET_GROUP_PRIORITY)............................ 42
2.5.13 Compare Function Window A Setting (AD_CMD_SET_WINDOWA).................................. 44
2.5.14 Compare Function Window B Setting (AD_CMD_SET_WINDOWB).................................. 48
2.5.15 Scan End Event (ADC140_ELC) Generating Condition Setting... 52
2.5.16 Acquiring Comparison Results of A/D Compare Function (AD_CMD_GET_CMP_RESULT) 54
2.5.17 A/D Converter State Acquisition (AD_CMD_GET_AD_STATE) 56
2.5.18 Low-Potential Side Reference Voltage Setting (AD_CMD_USE_VREFL0)......................... 58
2.5.19 High-Potential Side Reference Voltage Setting (AD_CMD_USE_VREFH0) 59
2.5.20 Sub-Clock Mode Setting (AD_CMD_SCLK_ENABLE)... 60
2.5.21 Offset Calibration (AD_CMD_CALIBRATION) 61
2.5.22 A/D Conversion Stop Function (AD_CMD_STOP_TRIG) 62
2.5.23 Auto-Read Function (Normal) (AD_CMD_AUTO_READ_NORMAL).................................. 63
2.5.24 Auto-Read Function (Block) Setting (AD_CMD_AUTO_READ_BLOCK) 66
2.5.25 Auto-Read Function (Compare) Setting (AD_CMD_AUTO_READ_COMPARE) 69
2.5.26 Auto-Read Stop Function (AD_CMD_AUTO_READ_STOP)... 72
2.5.27 Auto-Read Restart Function (AD_CMD_AUTO_READ_RESTART).................................. 74
2.6 Method for Fetching A/D Conversion Result.. 76
2.6.1 Fetching A/D Conversion Result with Read Function.. 76
2.6.2 Automatically Fetching A/D Conversion Result Using DMA Transfer (Auto-Read Function)77
2.7 Macro and Type Definitions 84
2.7.1 S14AD Initial Setting Code Definitions 84
2.7.2 A/D Conversion Start Trigger Definition... 85
2.7.3 Multiple A/D Conversion Channel Selection Definition 85

2.7.4 Temperature Sensor Output Usage Definition 86
2.7.5 Single A/D Conversion Channel Selection Definition 86
2.7.6 A/D Conversion Group Definition 86
2.7.7 Control Command Definition of Control Function 87
2.7.8 Addition/Average Mode Setting Definition 88
2.7.9 Self-Diagnosis Mode Setting Definition 88
2.7.10 Interrupt Function Setting Definition 88
2.7.11 S14AD Enable/Disable Definition 88
2.7.12 Disconnection Detection Assist Setting Definition 89
2.7.13 Group Priority Operation Setting Definition 89
2.7.14 Compare Function Window A Setting Definition 89
2.7.15 Compare Function Window B Setting Definition 90
2.7.16 Definition for Setting Compare Function Window A/B Complex conditions 90
2.7.17 S14AD Status Code Definition 91
2.7.18 Auto-Read Function Setting Definition 92
2.7.19 Definition to Specify Scan End Event (ADC140_ELC) Generating Condition 92
2.7.20 S14AD Event Code Definition 93
2.7.21 S14AD Error Code Definition 93
2.7.22 S14AD Callback Function Definition 93
2.8 Structure Definitions 94
2.8.1 st_adc_pins_t Structure 94
2.8.2 st_adc_add_mode_t Structure 94
2.8.3 st_adc_wina_t Structure 94
2.8.4 st_adc_winb_t Structure 95
2.8.5 st_adc_wina_result_t Structure 95
2.8.6 st_adc_cmp_result_t Structure 96
2.8.7 st_adc_status_info_t Structure 97
2.8.8 st_adc_dma_read_info_t Structure 98
2.9 State Transitions 99

3. Descriptions of Driver Operations 101
3.1 Single-Scan Mode 101
3.1.1 Basic Operation (Group A/Temperature Sensor Output) 101
3.1.2 Double-Trigger Mode 106
3.2 Continuous-Scan Mode 110
3.2.1 Basic Operation (Group A/Temperature Sensor Output) 110
3.3 Group-Scan Mode 114
3.3.1 Basic Operation (Group A/ Group B/Group C/Temperature Sensor Output) 114
3.3.2 Double-Trigger Mode 118
3.4 Compare Function 122
3.4.1 Compare Function (Window A/Window B/Temperature Sensor Output) 122
3.5 Configurations 128
3.5.1 Parameter Check 128
3.5.2 ADI Interrupt Request Control 128
3.5.3 GBADI Interrupt Request Control 128
3.5.4 GCADI Interrupt Request Control 129
3.5.5 WCMPM Interrupt Request Control 129

3.5.6 WCMPUM Interrupt Request Control ... 130
3.5.7 ADI Interrupt Priority Level ... 130
3.5.8 GBADI Interrupt Priority Level ... 130
3.5.9 GCADI Interrupt Priority Level ... 131
3.5.10 CMPAI Interrupt Priority Level ... 131
3.5.11 CMPBI Interrupt Priority Level ... 131
3.5.12 WCMPM Interrupt Priority Level ... 132
3.5.13 WCMPUM Interrupt Priority Level ... 132
3.5.14 CMPAI Snooze Mode ... 133
3.5.15 CMPBI Snooze Mode ... 133
3.5.16 Function Allocation to RAM ... 134

4. Detailed Information of Driver ... 135
4.1 Function Specifications ... 135
4.1.1 R_ADC_Open Function ... 136
4.1.2 R_ADC_Close Function ... 138
4.1.3 R_ADC_ScanSet Function ... 139
4.1.4 R_ADC_Start Function ... 142
4.1.5 R_ADC_Stop Function ... 143
4.1.6 R_ADC_Control Function ... 144
4.1.7 R_ADC_Read Function ... 148
4.1.8 R_ADC_GetVersion Function ... 150
4.1.9 adc_cmd_add_mode Function ... 151
4.1.10 adc_cmd_dbltrg Function ... 152
4.1.11 adc_cmd_diag Function ... 153
4.1.12 adc_cmd_auto_clear Function ... 155
4.1.13 adc_cmd_sampling Function ... 156
4.1.14 adc_cmd_charge Function ... 157
4.1.15 adc_cmd_group_priority Function ... 158
4.1.16 adc_cmd_set_windowa Function ... 160
4.1.17 adc_cmd_set_windowa_sub Function ... 162
4.1.18 adc_cmd_set_windowb Function ... 163
4.1.19 adc_cmd_get_cmp_result Function ... 166
4.1.20 adc_cmd_get_state Function ... 167
4.1.21 adc_cmd_set_vrefl0 Function ... 171
4.1.22 adc_cmd_set_vrefh0 Function ... 172
4.1.23 adc_cmd_set_sclk Function ... 173
4.1.24 adc_cmd_calibration Function ... 174
4.1.25 adc_cmd_set_adi Function ... 176
4.1.26 adc_cmd_set_gbadi Function ... 178
4.1.27 adc_cmd_set_gcadi Function ... 180
4.1.28 adc_cmd_set_wcmpm Function ... 182
4.1.29 adc_cmd_set_wcmpum Function ... 184
4.1.30 adc_cmd_set_elc Function ... 186
4.1.31 adc_cmd_stop_trigger Function ... 187
4.1.32 adc_cmd_auto_read Function ... 189
4.1.33 adc_cmd_auto_read_stop Function ... 191

4.1.34 adc_cmd_auto_read_restart Function 193
4.1.35 adc_dma_ram_init Function 195
4.1.36 adc_dma_drv_select Function 196
4.1.37 adc_dma_config Function 197
4.1.38 cadc_s14adi0_isr Function 200
4.1.39 adc_gbadi_isr Function 201
4.1.40 adc_gcadi_isr Function 202
4.1.41 adc_wcmpm_isr Function 203
4.1.42 adc_wcmpum_isr Function 204
4.1.43 adc_cmpai_isr Function 205
4.1.44 adc_cmpbi_isr Function 206
4.1.45 adc_dma_adi_isr Function 207
4.1.46 adc_dma_gbadi_isr Function 208
4.1.47 adc_dma_gcadi_isr Function 209
4.1.48 adc_dma_wcmpm_isr Function 210
4.1.49 adc_dma_wcmpum_isr Function 211
4.2 Macro and Type Definitions 212
4.2.1 Macro Definition List 212
4.2.2 e_adc_sst Definition 214
4.2.3 e_adc_dma_cmd_t Definition 214
4.3 Structure Definitions 215
4.3.1 st_adc_resources_t Structure 215
4.3.2 st_adc_dma_set Structure 217
4.4 Calling External Functions 218

5. Usage Notes 220
5.1 Arguments 220
5.2 Pin Settings 220
5.3 Conditions for Starting A/D Conversion 220
5.4 Restrictions on S14AD Function Combinations and Each Function 220
5.5 Power supply open control register (VOCR) setting 220
5.6 Registering Interrupts to NVIC 221
5.7 Exclusive Control of Interrupt Function and Auto-Read Function 222
5.8 Event Decision of Callback Functions 222
5.9 Notes on the use of DTC 224

6. Reference Documents 225

1. Overview

The S14AD driver allows the RE01 1500KB and 256KB Group devices to use the on-chip 14-bit A/D converter (note).

Note. The function names differ between the 1500KB group and the 256KB group.
The peripheral function names in each group are shown below.
1500KB Group : S14AD
256KB Group : ADC14

2. Driver Configuration

This chapter describes the information required for using this driver.

2.1 File Configuration

The R_S14AD drive corresponds to DeviceHAL in the CMSIS Driver Package and consists of five files: r_adc_api.c, r_adc_api.h, r_adc_cfg.h, pin.c, and pin.h in the vendor-specific file storage directory. The functions of the files are shown in Table 2-1. Figure 2-1 shows the file structure.

Table 2-1 Functions of R_S14AD Driver Files

File Name	Description
r_adc_api.c	Driver source file This file provides the main function of the driver. To use the S14AD driver, it is necessary to build this file.
r_adc_api.h	Driver header file This file defines macro, type, and prototype declarations used in the driver here.
r_adc_cfg.h	Configuration definition file This file provides configuration definitions that can be modified by the user.
pin.c	Pin setting file This file provides pin assignment processing for the driver.
pin.h	Pin setting header file

```
RE01 Group CMSIS Pack
    Device
      ├── Driver
      │     ├── Src
      │     │     └── r_adc
      │     │           └── r_adc_api.c: Driver source file
      │     └── Include
      │           └── r_adc_api.h: Driver header file
      ├── Config
      │     └── r_adc_cfg.h: Configuration definition file
      ├── pin.c: Pin setting file
      └── pin.h: Pin setting header file
```

Figure 2-1 File Configuration of S14AD Driver

2.2 Driver API

The S14AD driver has a single instance. To use this driver, access the API using function pointer to the instance. Table 2-2 shows the S14AD driver instance. Figure 2-2 shows an example of instance declaration. Table 2-3 lists the APIs contained in the instance. Figure 2-3 to Figure 2-10 show examples of access to the S14AD driver.

Table 2-2 List of S14AD Driver Instances

Instance	Description
Driver_S14AD	Instance for using S14AD

```
#include "r_adc_api.h"

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

Figure 2-2 Example of S14AD Driver Instance Declaration

Table 2-3 S14AD Driver APIs

API	Description	Reference
Open	Initializes the S14AD driver (initializes RAM, makes pin settings, and releases the module stop state). It also specifies the scan mode, A/D conversion accuracy, format of A/D data registers, and default sampling time for the S14AD driver. For details, see section 2.4.1, Setting A/D Scan Mode by Open Function.	4.1.1
Close	Releases the S14AD driver (releases the pins). It will also cause a transition to the module stop state if the A/D converter is not in the module stop state. It will also release the DMA driver if auto read of A/D data by DMA transfer is being used.	4.1.2
ScanSet	Makes scanning settings (selects the channels on which A/D conversion is performed, specifies whether to perform A/D conversion for the temperature sensor output, and selects the conditions for starting A/D conversion) for each group used in the S14AD driver. For details, see section 2.4.2, Setting A/D Input Pins and A/D Conversion Start Trigger by ScanSet Function.	4.1.3
Start	When a software trigger is specified as the A/D conversion start condition, it will start the A/D conversion (Note 1).	4.1.4
Stop	When a software trigger is specified as the A/D conversion start condition, the A/D conversion is stopped (Note 2).	4.1.5

Control	Executes a control command of the S14AD (Notes 3 and 4).
	Provides a variable whose type complies with the control command and specify a pointer to that variable in the second argument.
	For each control command and how to use it, see section 2.5, Setting S14AD Features by Control Function.
	
	[Example of using AD_CMD_SET_AUTO_CLEAR command]
	uint8_t arg = ADC_ENABLE; //Enable the A/D data register auto clear.
	Control(AD_CMD_SET_AUTO_CLEAR, &arg);
	//Pass a variable pointer to the second argument.
		4.1.6

Read	Reads the A/D conversion results of the S14AD.	4.1.7
GetVersion	Obtains the version of the S14AD driver.	4.1.8

Note 1. When an asynchronous trigger or a synchronous trigger is selected as the start condition of A/D conversion, A/D conversion is started by a trigger input (ELC event, TMR event, or ADTRG pin input).

Note 2. When an asynchronous trigger or synchronous trigger is selected as the start condition of A/D conversion, A/D conversion should be stopped using the AD_CMD_STOP_TRIG control command of the Control function. For control commands, see section 2.5, Setting S14AD Features by Control Function.

Note 3. Some functions or modes cannot be used simultaneously. For a combination of functions, see Table 2-12 .

Note 4. Executing control commands by the Control function is enabled only after the Open function has been executed.

```
#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 conversion channels */
    st_adc_status_info_t result_status;    /* 2nd argument of AD_CMD_GET_AD_STATE */
    e_adc_int_method_t adc_int = ADC_INT_ENABLE;    /* Variable to enable interrupts */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, callback);
                     /* Initialize the S14AD driver
                     Initialize RAM, make pin settings, and release the module stop state
                      Also set the mode specified with argument */
    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                        /* Use AN000, AN001, AN006 and AN02 for A/D conversion */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* Temperature sensor not used for A/D conversion */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);
                  /* Specify pins used for group A and set the A/D conversion start trigger */

    (void)adcDev->Control(AD_CMD_SET_ADI_INT, &adc_int); /* Enable a scan end interrupt */

    while (1)
    {
        (void)adcDev->Start();  /* Start A/D conversion */

        do
        {
            (void)adcDev->Control(AD_CMD_GET_AD_STATE, & result_status);    /* Acquire the A/D status */
        }
        while (ADC_STATE_RUN == result_status.ad_info);    /* Wait for completion of A/D conversion */
    }
}
/*****************************************************************************************************
* callback function
*****************************************************************************************************/
static void callback(uint32_t event)
{
    switch(event)
    {
        case ADC_EVT_SCAN_COMPLETE:
        {
            /* Describe the processing to be performed when A/D conversion is finished */
        }
        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:
        {
            /* Interrupt event does not occur because it is not enabled */
        }
        break;
    }
}
```

Figure 2-3 Example of Access to S14AD Driver (Software Trigger)

```
#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;  /* A/D conversion channel of group A */
    st_adc_pins_t groupb_pin;  /* A/D conversion channel of group B */
    st_adc_pins_t groupc_pin;  /* A/D conversion channel of group C */
    e_adc_int_method_t adc_int = ADC_INT_ENABLE;    /* Variable to enable interrupts */

    (void)adcDev->Open(ADC_GROUP_SCAN | ADC_14BIT, 0x10, callback);
                    /* Initialize the S14AD driver
                    Initialize RAM, make pin settings, and release the module stop state
                     Also set the mode specified with argument */
    groupa_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01;    /* Specify AN00 and AN01 for group A */
    groupa_pin.sensor   = ADC_SENSOR_NOTUSE; /* Temperature sensor not used */
    (void)adcDev->ScanSet(ADC_GROUP_A, groupa_pin, ADC_TRIGER_ADTRG);
                 /* Specify pins to be used for group A and set the A/D conversion start trigger */

    groupb_pin.an_chans = ADC_MSEL_AN03 | ADC_MSEL_AN04;    /* Specify AN03 and AN04 for group B */
    groupb_pin.sensor   = ADC_SENSOR_NOTUSE; /* Temperature sensor not used */
    (void)adcDev->ScanSet(ADC_GROUP_B, groupb_pin, ADC_TRIGER_ELC);
                 /* Specify pins to be used for group B and set the A/D conversion start trigger */

    groupc_pin.an_chans = ADC_MSEL_AN22;   /* Specify AN22 for group C */
    groupc_pin.sensor   = ADC_SENSOR_NOTUSE; /* Temperature sensor not used */
    (void)adcDev->ScanSet(ADC_GROUP_C, groupc_pin, ADC_TRIGER_TMR);
                 /* Specify pins to be used for group C and set the A/D conversion start trigger */

    (void)adcDev->Control(AD_CMD_SET_ADI_INT, &adc_int);
                                             /* Enable the group-A scan end interrupt */
    (void)adcDev->Control(AD_CMD_SET_GROUPB_INT, &adc_int);
                                             /* Enable the group-B scan end interrupt */
    (void)adcDev->Control(AD_CMD_SET_GROUPC_INT, &adc_int);
                                             /* Enable the group-C scan end interrupt */

    while(1)
    {
        /* Wait for an input of the A/D conversion start trigger (Note1)
          [A/D conversion start trigger]
          Group A: ADTRG0 pin input
          Group B: ELC_S14AD event
          Group C: TMR_TCORA (TMR compare match A) */
    }
}
/*******************************************************************************************************
* callback function
*******************************************************************************************************/
static void callback(uint32_t event)
{
    switch(event)
    {
        case ADC_EVT_SCAN_COMPLETE:
        {
            /* Describe the processing to be performed
               when A/D conversion terminates (group-A scan terminates
               in group-scan mode) */
        }
        break;
```

Figure 2-4 Example of Access to S14AD Driver (Group Scan) (1/2)

```
        case ADC_EVT_SCAN_COMPLETE_GROUPB:
        {
            /* Describe the processing to be performed
               when A/D conversion of group B is completed */
        }
        break;

        case ADC_EVT_SCAN_COMPLETE_GROUPC:
        {
            /* Describe the processing to be performed
               when A/D conversion of group C is completed */
        }
        break;

        case ADC_EVT_CONDITION_MET:
        {
            /* Describe the processing (Note 2) to be performed
               when window A comparison condition is met */
        }
        break;

        case ADC_EVT_CONDITION_METB
        {
            /* Describe the processing (Note 2) to be performed
                when window A comparison condition is not met */
        }
        break;

        case ADC_EVT_WINDOW_CMP_MATCH:
        {
            /* Describe the processing (Note 3) to be performed
                when window A or B comparison conditions are met */
        }
        break;

        case ADC_EVT_WINDOW_CMP_UNMATCH:
        {
            /* Describe the processing (Note 3) to be performed
                when window A or B comparison conditions are not met */
        }
        break;
        default:
        {
        }
        break;
    }
}
```

Figure 2-5 Example of Access to S14AD Driver (Group Scan) (2/2)

Note 1. When using a synchronous trigger (ELC_S14AD event or TMR_TCORA (TMR compare match A)), set each function (ELC or TMR) separately.

Note 2. In this example, this event does not occur because no window compare match event is set.

Note 3. In this example, this event does not occur because no window compare match event is set and interrupts are not enabled for window A/B compare function match/mismatch events.

```
#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;   /* A/D conversion channel of group A */
    st_adc_wina_t wina;      /* 2nd argument of AD_CMD_SET_WINDOW */
    st_adc_winb_t winb;      /* 2nd argument of AD_CMD_SET_WINDOWB */
    st_adc_status_info_t result_status;      /* 2nd argument of AD_CMD_GET_AD_STATE */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, callback);
                                /* Initialize the S14AD driver
                                Initialize RAM, make pin settings, and release the module stop state
                                Also set the mode specified with argument */

    groupa_pin.an_chans = ADC_MSEL_AN02 | ADC_MSEL_AN06;      /* Specify AN02 and AN06 channels */
    groupa_pin.sensor     = ADC_SENSOR_NOTUSE; /* Temperature sensor not used */
    (void)adcDev->ScanSet(ADC_GROUP_A, groupa_pin, ADC_TRIGER_ADTRG);
                            /* Specify pins to be used for group A and set the A/D conversion start trigger */

    wina.inten                  = ADC_ENABLE;         /* Enable the compare A interrupt */
    wina.cmp_mode               = ADC_CMP_WINDOW;      /* Compare windows */
    wina.level1                 = 0x02000;            /* Comparison reference value 1 */
    wina.level2                 = 0x00100;            /* Comparison reference value 2 */
    wina.chans.an_chans         = ADC_MSEL_AN02;   /* Specify AN02 as the channel for A/D conversion */
    wina.chans.sensor           = ADC_SENSOR_NOTUSE;      /* Temperature sensor not used */
    wina.chan_cond.an_chans = ADC_MSEL_AN02;
                                /* Channel-specific comparison condition: AN02 is specified within a range from level 1 to level 2 */
    wina.chan_cond.sensor     = ADC_SENSOR_NOTUSE; /* Temperature sensor not used */
    (void)adcDev->Control(AD_CMD_SET_WINDOWA, &wina);      /* Set the window A comparison */

   winb.inten          = ADC_ENABLE;                          /* Enable the compare B interrupt */
   winb.winb_cond = ADC_WINB_WINDOW_BETWEEN;   /* Within a range of windows */
   winb.comb           = ADC_COMB_OR;                         /* Complex condition: OR condition */
   winb.level1          = 0x03000;                            /* Comparison reference value 1 */
   winb.level2          = 0x00010;                            /* Comparison reference value 2 */
   winb.channel       = ADC_SSEL_AN06;                        /* Specify AN06 as a channel to be used */
   result = adcDev->Control(AD_CMD_SET_WINDOWB, &winb); /* Set the window B comparison */

    while(1)
    {
        (void)adcDev->Start();   /* Start A/D conversion */

        do
        {
            (void)adcDev->Control(AD_CMD_GET_AD_STATE, & result_status);      /* Acquire A/D status */
        }
        while (ADC_STATE_RUN == result_status.ad_info);      /* Wait for completion of A/D conversion */
    }
}
```

Figure 2-6 S14AD Example of Access to S14AD Driver (Compare Match) (1/2)

```
/**********************************************************************************************************
* callback function
**********************************************************************************************************/
static void callback(uint32_t event)
{
    switch(event)
    {
        case ADC_EVT_CONDITION_MET:
        {
            /* Describe the processing to be performed when window A comparison condition is met */
        }
        break;
        case ADC_EVT_CONDITION_METB:
        {
            /* Describe the processing to be performed when window B comparison condition is met */
        }
        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:
        {
            /* Interrupt event does not occur because it is not enabled */
        }
        break;
    }
}
```

Figure 2-7 S14AD Example of Access to S14AD Driver (Compare Match) (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 transfer destination: RAM */

main()
{
    st_adc_pins_t scanset_pin;   /* ScanSet() A/D conversion channels */
    st_adc_dma_read_info_t arg;     /* 2nd argument of AD_CMD_AUTO_READ_NORMAL */
    st_adc_status_info_t result_status;     /* 2nd argument of AD_CMD_GET_AD_STATE */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL);
                        /* Initialize the S14AD driver
                        Initialize RAM, make pin settings, and release the module stop state
                        Also set the mode specified with argument */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                    /* Use AN000, AN001, AN006 and AN022 for A/D conversion */
    scanset_pin.sensor    = ADC_SENSOR_NOTUSE;    /* Temperature sensor not used for A/D conversion */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin,   ADC_TRIGER_SOFT);
                    /* Specify pins to be used for group A and set the A/D conversion start trigger */


    arg.cb_event       = callback;                       /* Callback function */
    arg.dma_fact       = ADC_READ_ADI;                  /* DMA transfer trigger */
    arg.src_addr       = (uint32_t)&S14AD->ADDR1; /* Transfer source: A/D data register */
    arg.dest_addr      = (uint32_t)&read_data[0];      /* Transfer destination: RAM */
    arg.transfer_count = 5;                              /* Transfer count: 5 times */
    arg.block_size     = 0;                              /* No block size specified (fixed at 0) */
    arg.reg_size       = 0;                              /* No transfer size specified (fixed at word) */

    (void)adcDev->Control(AD_CMD_AUTO_READ_NORMAL, &arg);
                                    /* Specify the auto-read function for A/D conversion results with DMA
transfer */

    while (1)
    {
        (void)adcDev->Start();   /* Start A/D conversion */

        do
        {
            (void)adcDev->Control(AD_CMD_GET_AD_STATE, & result_status);      /* Acquire the A/D status */
        }
        while (ADC_STATE_RUN == result_status.ad_info);      /* Wait for completion of A/D conversion */
    }
}

/******************************************************************************************************
* callback function
******************************************************************************************************/
static void callback(void)
{
    /* Describe the processing to be performed when DMA transfer for the specified count (5) terminates */
}
```

Figure 2-8 Example of Access to S14AD Driver (Auto-Read of A/D Data with DMA)

```
#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 conversion channels */
    e_adc_int_method_t adc_int = ADC_INT_POLLING;       /* Specify polling */
    st_adc_status_info_t result_status;       /* 2nd argument of AD_CMD_GET_AD_STATE */


    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL);
                          /* Initialize the S14AD driver
                          Initialize RAM, make pin settings, and release the module stop state
                          And set the mode specified with argument */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                              /* Use AN000, AN001, AN006 and AN022 for A/D conversion */
    scanset_pin.sensor     = ADC_SENSOR_NOTUSE;     /* Temperature sensor not used for A/D conversion */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_ADTRG);
                      /* Specify pins to be used for group A and sets the A/D conversion start trigger */

    (void)adcDev->Control(AD_CMD_SET_ADI_INT, &adc_int);
                                                    /* Use a scan end interrupt with polling enabled */

    while (1)
    {
        /* Start A/D conversion by an input of external trigger (ADTRG) */
        (void)adcDev->Control(AD_CMD_GET_AD_STATE, & result_status);      /* Acquire the A/D status */
        if (ADC_CONV_COMPLETE == result_status.groupa_info)
        {
            /* Describe the processing to be performed when A/D conversion of group A is completed */
        }
    }
    while(1);
}
```

Figure 2-9 Example of Access to S14AD Driver (Polling)

```
#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 conversion channels */
    st_adc_status_info_t result_status;      /* 2nd argument of AD_CMD_GET_AD_STATE */
    e_adc_int_method_t adc_int = ADC_INT_ENABLE;      /* Variable to enable interrupts */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, callback);
                            /* Initialize the S14AD driver
                            Initialize RAM, make pin settings, and release the module stop state
                            Also set the mode specified with argument */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                                    /* Use AN000, AN001, AN006 and AN022 for A/D conversion */
    scanset_pin.sensor     = ADC_SENSOR_NOTUSE;     /* Temperature sensor not used for A/D conversin */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_ADTRG);
                        /* Specify pins to be used for group A and set the A/D conversion start trigger */

    (void)adcDev->Control(AD_CMD_SET_ADI_INT, &adc_int);
                                                            /* Enable a scan end interrupt */

    while (1)
    {

        do
        {
            /* Start A/D conversion by ADTRG pin input */
            (void)adcDev->Control(AD_CMD_GET_AD_STATE, & result_status);      /* Acquire the A/D status */
        }
        while (ADC_STATE_RUN == result_status.ad_info);      /* Wait for completion of A/D conversion */

    }
}
/*******************************************************************************************************
* callback function
*******************************************************************************************************/
static void callback(uint32_t event)
{
    switch(event)
    {
        case ADC_EVT_SCAN_COMPLETE:
        {
            /* Describe the processing to be performed when A/D conversion is completed */
        }
        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:
        {
            /* Interrupt event does not occur because it is not enabled */
        }
        break;
    }
}
```

Figure 2-10 Example of Access to S14AD Driver (External Trigger)

2.3 Pin Configuration

The pins to be used by this driver are set with the R_S14AD_Pinset function in pin.c and released with the R_S14AD_Pinclr function in pin.c. The R_S14AD_Pinset function is called from the Open function and the R_S14AD_Pinclr function is called from the Close function.

Select the pin to be used by editing the R_S14AD_Pinset and R_S14AD_Pinclr functions of pin.c. Figure 2-11 and Figure 2-12 show examples of pin settings.

```
    /**************************************************************************//**
    * @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);

        /* Set P000 as AN000 pin. */
        /* 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;

        /* Set P001 as AN001 pin. */
        /* 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;

        /* Set P002 as AN002 pin. */
        /* 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;

        /* Set P500 as ADTRG0 pin. */
        /* 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() */
```

Figure 2-11 Examples of Setting Pin (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);

    /* Release AN000 pin */
    /* AN000 : P000 */
    PFS->P000PFS &= R_PIN_PRV_CLR_MASK;

    /* Release AN001 pin */
    /* AN001 : P001 */
    PFS->P001PFS &= R_PIN_PRV_CLR_MASK;

    /* Release AN002 pin */
    /* AN002 : P002 */
    PFS->P002PFS &= R_PIN_PRV_CLR_MASK;

    /* AN003 : P003 */
//    PFS->P003PFS &= R_PIN_PRV_CLR_MASK;
    ...
    /* AN028 : P506 */
//    PFS->P506PFS &= R_PIN_PRV_CLR_MASK;

      /* ADTRG0 : P204 */
//    PFS->P204PFS &= R_PIN_PRV_CLR_MASK;

    /* Release ADTRG0 pin */
    /* 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() */
```

Figure 2-12 Examples of Setting Pin (2/2)

2.4 Initialization of S14AD

2.4.1 Setting A/D Scan Mode by Open Function

The Open function initializes the S14AD driver (initializes RAM, makes pin settings, and cancels the module stop state). At the same time, it also specifies the scan mode, A/D conversion accuracy, format of A/D data registers, and default sampling time (initial value of ADSSTRn (Note 1)). Specify definitions of the mode to be used in combination in arguments.

Example:

To initialize this driver when single-scan mode, 12-bit accuracy, flush-left data for A/D data registers, default sampling time (0x10), and no callback are set:

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

Definitions that can be specified by the S14AD driver are shown in Table 2-4 to Table 2-6. When definitions are not specified, the function indicated with (default) is enabled. Set a value from 2 to 255 (Note 3) for the default sampling time (Note 2) in the second parameter.

Note 1. 256KB Group : n = 0 to 7,L,T 1500KB Group : n = 0 to 6,L,T

Note 2. Initial value of ADSSTRn

Note 3. The specifiable range of the sampling time depends on the setting of sub-clock mode. For setting the sub-clock mode, see section 2.5.20, Sub-Clock Mode Setting (AD_CMD_SCLK_ENABLE). The specifiable range of the sampling time for each mode is shown below.

When sub-clock mode is disabled: 5 to 255

When sub-clock mode is enabled: 2 to 8

Table 2-4 Scan Mode Definitions

Command	Description
ADC_SINGLE_SCAN (default)	Single-scan mode
ADC_GROUP_SCAN	Group-scan mode
ADC_REPEAT_SCAN	Continuous-scan mode

Table 2-5 List of A/D Conversion Accuracy Definitions

Command	Description
ADC_14BIT (default)	14-bit accuracy
ADC_12BIT	12-bit accuracy

Table 2-6 List of A/D Data Register Format Definitions

Command	Description
ADC_RIGHT (default)	Flush right
ADC_LEFT	Flush left

2.4.2 Setting A/D Input Pins and A/D Conversion Start Trigger by ScanSet Function

The ScanSet function sets the A/D input pins (selects the channels on which A/D conversion is performed and specifies whether to perform A/D conversion for the temperature sensor output) for each A/D conversion group (group A, group B, or group C) and sets the trigger for starting A/D conversion. Table 2-7 shows the definitions of groups that are to be specified in the first argument. Table 2-8 shows the definitions for setting A/D conversion channels that are to be specified in the second argument. Table 2-9 shows the definitions of A/D conversion start conditions that are to be specified in the third argument.

Example:

```
st_adc_pins_t scanset_pin;    /* Variable to specify A/D conversion channel */
scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN03 | ADC_MSEL_AN22;
                                         /* Use AN000, AN001, AN003, and AN022 for A/D conversion */
scanset_pin.sensor    = ADC_SENSOR_NOTUSE;    /* Temperature sensor not used for A/D conversion
*/
adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);
```

Table 2-7 List of A/D Group Definitions

Command	Description
ADC_GROUP_A	Group A
ADC_GROUP_B (Note)	Group B
ADC_GROUP_C (Note)	Group C

Note. Can only be specified in group-scan mode.

Table 2-8 Definitions to Set A/D Conversion Channels

Element	Setting Value	Description
an_chans	Combination of ADC_MSEL_ANn channels(Note) (1500KB Group : n = 00 to 06,16,17,20 to 28 256KB Group : n = 00 to 07,16,17,20 to 21)	Target channels for A/D conversion [Setting example] /* AN000 and AN016 are used */ arg.chans.anchans = ADC_MSEL_AN00 | ADC_MSEL_AN16;
sensor	ADC_SENSOR_NOTUSE	Temperature sensor output is not used.
	ADC_MSEL_TEMP	Temperature sensor output is used.

Note. Only 256KB group can be specified with ADC_MSEL_VSC_VCC

Table 2-9 Definitions of A/D Conversion Start Conditions

Command	Description
ADC_TRIGER_SOFT	Software trigger
ADC_TRIGER_TMR	TMR trigger (TMR0_TCORA (compare match between TCORA register and TCNT counter))
ADC_TRIGER_ELC	ELC trigger (ELC_S14AD)
ADC_TRIGER_ADTRG	External trigger input (ADTRG0)
ADC_TRIGER_LOW_PRIORITY_CONT_SCAN	Trigger source deselected (Note)

Note. When continuous operation for a low-priority group is enabled in the A/D group-scan priority control function, the A/D conversion trigger source for the group with the lowest priority should be deselected (ADC_TRIGER_LOW_PRIORITY_CONT_SCAN).

2.5 Setting S14AD Features by Control Function

Each feature can be set or executed by executing its corresponding control command in the Control function. The Control function should be used after performing initialization by the Open function. For other restrictions, see the details of each control command.

When executing the Control function, specify a control command in the first argument. Provide a variable whose type complies with the control command and specify a pointer to that variable in the second argument. Table 2-10 and Table 2-11 show a list of control commands for the S14AD driver. For the type of the second argument and usage example of the control command, see the reference section containing details of each control command.

Not all features set by control commands can be used simultaneously. For combination of features, see Table 2-12.

Table 2-10 List of Control Commands (1/2)

Command	Description	Details
AD_CMD_SET_ADD_MODE	Sets the A/D-converted value addition/average mode.	2.5.1
AD_CMD_SET_DBLTRG	Sets the double-trigger mode.	2.5.2
AD_CMD_SET_DIAG	Sets the self-diagnosis mode.	2.5.3
AD_CMD_SET_ADI_INT	Sets an A/D scan end interrupt (ADC140_ADI) of group A..	2.5.4
AD_CMD_SET_GROUPB_INT	Sets an A/D scan end interrupt (ADC140_GBADI) of group B.	2.5.5
AD_CMD_SET_GROUPC_INT	Sets an A/D scan end interrupt (ADC140_GCADI) of group C.	2.5.6
AD_CMD_SET_WCMPM_INT	Sets a condition match interrupt (ADC140_WCMPM) of the window A/B compare function.	2.5.7
AD_CMD_SET_WCMPUM_INT	Sets a condition mismatch interrupt (ADC140_WCMPUM) of the window A/B compare function.	2.5.8
AD_CMD_SET_AUTO_CLEAR	Sets automatic clearing of A/D data registers.	2.5.9
AD_CMD_SET_SAMPLING_AN000	Sets the sampling time of AN000 or self-diagnosis.	2.5.10
AD_CMD_SET_SAMPLING_AN001	Sets the sampling time of AN001.	
AD_CMD_SET_SAMPLING_AN002	Sets the sampling time of AN002.	
AD_CMD_SET_SAMPLING_AN003	Sets the sampling time of AN003.	
AD_CMD_SET_SAMPLING_AN004	Sets the sampling time of AN004.	
AD_CMD_SET_SAMPLING_AN005	Sets the sampling time of AN005.	
AD_CMD_SET_SAMPLING_AN006	Sets the sampling time of AN006.	
AD_CMD_SET_SAMPLING_AN007 (Note 1)	Sets the sampling time of AN007.	
AD_CMD_SET_SAMPLING_AN016_ AN017_AN020_TO_AN028(Note 2)	Sets the sampling time of AN016, AN017, and AN020 to AN028.	
AD_CMD_SET_SAMPLING_AN016_ AN017_AN020_AN021_VSC_VCC (Note 1)	Sets the sampling time of AN016, AN017, AN020, AN021 and VSC_VCC voltage	
AD_CMD_SET_SAMPLING_TEMP	Sets the sampling time for the temperature sensor output.	
AD_CMD_SET_ADNDIS	Sets the disconnection detection assist function.	2.5.11
AD_CMD_SET_GROUP_PRIORITY	Sets group priority operation.	2.5.12
AD_CMD_SET_WINDOWA	Sets the compare function window A.	2.5.13
AD_CMD_SET_WINDOWB	Sets the compare function window B.	2.5.14

Note 1. 256KB Group Only

Note 2. 1500KB Group Only

Table 2-11 List of Control Commands (2/2)

Command	Description	Details
AD_CMD_SET_ELC	Sets the generating condition of a scan end event (ADC140_ELC).	2.5.15
AD_CMD_GET_CMP_RESULT	Acquires comparison results of A/D compare functions.	2.5.16
AD_CMD_GET_AD_STATE	Acquires the state of the A/D converter.	2.5.17
AD_CMD_USE_VREFL0	Sets VREFL0 as the reference voltage on the low-potential side.	2.5.18
AD_CMD_USE_VREFH0	Sets VREFH0 as the reference voltage on the high-potential side.	2.5.19
AD_CMD_SCLK_ENABLE	Specifies a sub-clock to be used for A/D conversion.	2.5.20
AD_CMD_CALIBRATION	Executes offset calibration.	2.5.21
AD_CMD_STOP_TRIG	Releases a trigger source and stops A/D conversion.	2.5.22
AD_CMD_AUTO_READ_NORMAL	Sets an A/D conversion scan end interrupt request as the DMA start trigger and reads the A/D conversion result of the specified channel using DMA transfer.	2.5.23
AD_CMD_AUTO_READ_BLOCK	Sets an A/D conversion scan end interrupt request as the DMA start trigger and reads the A/D conversion results of multiple channels using DMA transfer.	2.5.24
AD_CMD_AUTO_READ_COMPARE	Sets a condition match/mismatch event of the window A/B compare function as the DMA start trigger and reads the values of the desired registers using DMA transfer.	2.5.25
AD_CMD_AUTO_READ_STOP	Stops auto read using DMA transfer due to a desired DMA start trigger.	2.5.26
AD_CMD_AUTO_READ_RESTART	Restarts auto-read using DMA transfer due to a desired DMA start trigger.	2.5.27

Table 2-12 Combination List of Functions Used

	Double trigger	Group scan	Self-diagnosis	Addition/ averaging	Group A priority control	Sensor	Compare match	Disconnection detection assist	Auto clear
Double trigger			X			X	X		
Group scan									
Self-diagnosis	X			X			X	X	
Addition/ averaging			X						
Group A priority control									
Sensor	X							X	
Compare match	X		X						
Disconnection detection assist			X			X			
Auto clear									

X: Cannot be used simultaneously. An error is returned when the Control function (control command for the applicable function) is executed.

2.5.1 Addition/Average Mode Setting (AD_CMD_SET_ADD_MODE)

The AD_CMD_SET_ADD_MODE command selects the A/D-converted value addition mode or average mode. Specify a pointer to the st_adc_add_mode_t type variable in the second argument. Table 2-13 shows settings of the st_adc_add_mode_t type argument. Figure 2-13 shows a usage example of the AD_CMD_SET_ADD_MODE command.

The following restriction applies to this command.

- Can be executed only when A/D conversion is halted (ADCSR.ADST = 0). For state transitions of the A/D conversion operation, see section 2.9, State Transitions.

Table 2-13 Settings of st_adc_add_mode_t Type Argument

Element	Setting	Description
add_mode	ADC_ADD_OFF	A/D-converted value addition mode or average mode is disabled.
	ADC_ADD_2_SAMPLES	Addition mode, 2-time conversion (addition of one time)
	ADC_ADD_4_SAMPLES	Addition mode, 4-time conversion (addition of 3 times)
	ADC_ADD_16_SAMPLES (Note)	Addition mode, 16-time conversion (addition of 15 times)
	ADC_ADD_AVG_2_SAMPLES	Average mode, 2-time conversion (addition of one time)
	ADC_ADD_AVG_4_SAMPLES	Average mode, 4-time conversion (addition of 3 times)
	ADC_ADD_AVG_16_SAMPLES(Note)	Average mode, 16-time conversion (addition of 15 times)
chans. an_chans	Combination of ADC_MSEL_ANn channels (1500KB Group : n = 00 to 06,16,17,20 to 28 256KB Group : n = 00 to 07,16,17,20 to 21)	A/D conversion channels subject to A/D-converted value addition/averaging [Setting example] /* AN000 and AN016 are used */ arg.chans.anchans = ADC_MSEL_AN00 | ADC_MSEL_AN16;
chans. sensor	ADC_SENSOR_NOTUSE	Temperature sensor output is not used.
	ADC_MSEL_TEMP	Temperature sensor output is used.

Note. Can be used only when the A/D conversion accuracy is 12-bit accuracy.

```
#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 conversion channels */
    st_adc_add_mode_t arg;    /* 2nd argument of AD_CMD_SET_ADD_MODE */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* Initialize the S14AD 
driver */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                            /* Use AN000, AN001, AN006 and AN022 for A/D 
conversion */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* Temperature sensor not used for A/D conversion 
*/
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);

    arg.add_mode      = ADC_ADD_4_SAMPLES;    /* Addition mode/4-time conversion (add 3 
times) */
    arg.chans.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                                   /* Select AN000, AN006, and AN022 for 
addition */
    arg.chans.sensor   = ADC_SENSOR_NOTUSE;   /* Temperature sensor not used */
    (void)adcDev->Control(AD_CMD_SET_ADD_MODE, &arg);  /* Set A/D-converted value 
addition/average mode */

    (void)adcDev->Start();  /* Start A/D conversion */

    while(1);
}
```

Figure 2-13 Example of AD_CMD_SET_ADD_MODE Command

2.5.2 Double-Trigger Mode Setting (AD_CMD_SET_DBLTRG)

The AD_CMD_SET_DBLTRG sets the double-trigger mode. Specify a pointer to the int8_t type variable in the second argument. Table 2-14 shows settings of the second argument. Figure 2-14 shows a usage example of the AD_CMD_SET_DBLTRG command.

The following restrictions apply to this command.

- Can be used in single-scan mode or group-scan mode. Cannot be used in continuous-scan mode (error is returned).
- Specify a synchronous trigger (TMR or ELC) as the start condition of A/D conversion. A synchronous trigger cannot be used when it is a software trigger (error is returned).
- Can only be executed when A/D conversion is halted (ADCSR.ADST = 0). For state transitions of the A/D conversion operation, see section 2.9, State Transitions.

Table 2-14 Settings of 2nd Argument

Setting	Description
-1	Double-trigger mode disabled
1500KB Group : 00 to 06,16,17,20 to 28 256KB Group : 00 to 07,16,17,20 to 21	Channel for A/D conversion data duplication

```
#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 conversion channels */
    int8_t arg;    /* 2nd argument of AD_CMD_SET_DBLTRG */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* Initialize the S14AD driver */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                            /* Use AN000, AN001, AN006 and AN022 for A/D conversion */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* Temperature sensor not used for A/D conversion */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);    /* Software trigger */

    arg.add_mode      = 6;    /* Specify AN006 as a channel for A/D conversion data duplication */
    (void)adcDev->Control(AD_CMD_SET_DBLTRG, &arg);  /* Set double-trigger mode */

    (void)adcDev->Start();  /* Start A/D conversion */

    while(1);
}
```

Figure 2-14 Example of AD_CMD_SET_DBLTRG Command

2.5.3 Self-Diagnosis Mode Setting (AD_CMD_SET_DIAG)

The AD_CMD_SET_DIAG command sets the self-diagnosis mode. Specify a pointer to the e_adc_diag_t type variable in the second argument. Table 2-15 shows settings of the e_adc_diag_t type argument. Figure 2-15 shows a usage example of the AD_CMD_SET_DIAG command.

Table 2-15 Settings of e_adc_diag_t Type Argument

Setting	Description
AD_DIAG_DISABLE	Self-diagnosis disabled
AD_DIAG_0V	0 V voltage
AD_DIAG_HARF	Reference power supply (VREFH0) × 1/2 voltage
AD_DIAG_BASE	Reference power supply (VREFH0) voltage
AD_DIAG_ROTATE	Self-diagnosis voltage rotation mode

```
#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 conversion channels */
    e_adc_diag_t arg;    /* 2nd argument of AD_CMD_SET_DIAG */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* Initialize the S14AD driver */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                    /* Use AN000, AN001, AN006 and AN022 for A/D conversion */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* Temperature sensor not used for A/D conversion */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);

    arg               = AD_DIAG_0V;               /* Execute self-diagnosis using a voltage of 0 V */
    (void)adcDev->Control(AD_CMD_SET_DIAG, &arg);  /* Set self-diagnosis mode */

    (void)adcDev->Start();  /* Start A/D conversion */

    while(1);
}
```

Figure 2-15 Example of AD_CMD_SET_DIAG Command

2.5.4 ADI Interrupt Function Setting (AD_CMD_SET_ADI_INT)

The AD_CMD_SET_ADI_INT command sets a single-scan or group A A/D scan end interrupt (ADC140_ADI). Specify a pointer to the e_adc_int_method_t type variable in the second argument. Table 2-16 shows settings of the e_adc_int_method_t type argument. Figure 2-17 shows a usage example of the AD_CMD_SET_ADI_INT command.

The following restrictions apply to this command.

- The interrupt function and auto-read function are controlled exclusively. When the auto-read function is using the A/D scan end interrupt (ADC140_ADI) request, it cannot be used by this command (error is returned).
- The interrupt source (ADC140_GBADI) must be registered in the Nested Vector Interrupt Controller (hereinafter called the NVIC) by modifying r_system_cfg.h. An example of registering interrupts to the NVIC is shown in Figure 2-16. For the NVIC definition of each interrupt source and an example of registering interrupts, refer to section 5.6, Registering Interrupts to NVIC.

Table 2-16 Settings of e_adc_int_method_t Type Argument

Command	Description
ADC_INT_DISABLE	Disables interrupts.
ADC_INT_POLLING	Polling
ADC_INT_ENABLE	Enables interrupts.

```
...
#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 */
...
```

Figure 2-16 Registering Interrupts with NVIC

```
#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 conversion channels */
    e_adc_int_method_t adc_int = ADC_INT_ENABLE;    /* Variable to enable interrupts */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* Initialize the S14AD driver */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                    /* Use AN000, AN001, AN006 and AN022 for A/D conversion */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* Temperature sensor not used for A/D conversion */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);    /* Software trigger */

    (void)adcDev->Control(AD_CMD_SET_ADI_INT, &adc_int);
                                             /* Enable a scan end interrupt */


    (void)adcDev->Start();  /* Start A/D conversion */

    while(1);
}

/****************************************************************************************************
* callback function
****************************************************************************************************/
static void callback(uint32_t event)
{
    switch(event)
    {
        case ADC_EVT_SCAN_COMPLETE:
        {
            /* Describe the processing to be performed when A/D conversion is finished */
        }
        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:
        {
            /* Interrupt event does not occur because it is not enabled */
        }
        break;

    }
}
```

Figure 2-17 Example of AD_CMD_SET_ADI_INT Command

2.5.5 GBADI Interrupt Function Setting (AD_CMD_SET_GROUPB_INT)

The AD_CMD_SET_GROUPB_INT command sets the A/D scan end interrupt of group B (ADC140_GBADI). Specify a pointer to the e_adc_int_method_t type variable in the second argument. Table 2-17 shows settings of the e_adc_int_method_t type argument. Figure 2-19 shows a usage example of the AD_CMD_SET_GROUPB_INT command.

The following restrictions apply to this command.

- The interrupt function and auto-read function are controlled exclusively. When the auto-read function is using the A/D scan end interrupt (ADC140_GBADI) request, it cannot be used by this command (error is returned).
- The interrupt source (ADC140_GBADI) must be registered in the Nested Vector Interrupt Controller (hereinafter called the NVIC) by modifying r_system_cfg.h. An example of registering interrupts to the NVIC is shown in Figure 2-18. For the NVIC definition of each interrupt source and an example of registering interrupts, refer to section 5.6, Registering Interrupts to NVIC.
- The A/D scan end interrupt of group B is valid only in group-scan mode.

Table 2-17 Settings of e_adc_int_method_t Type Argument

Command	Description
ADC_INT_DISABLE	Disables interrupts.
ADC_INT_POLLING	Polling
ADC_INT_ENABLE	Enables interrupts.

```
...
#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 */
...
```

Figure 2-18 Registering Interrupts with NVIC

```
#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;  /* A/D conversion channel of group A */
    st_adc_pins_t groupb_pin;  /* A/D conversion channel of group B */
    e_adc_int_method_t adc_int = ADC_INT_ENABLE;    /* Variable to enable interrupts */

    (void)adcDev->Open(ADC_GROUP_SCAN | ADC_14BIT, 0x10, callback); /* Initialize the S14AD driver */

    groupa_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01;    /* Specify AN00 and AN01 for group A */
    groupa_pin.sensor   = ADC_SENSOR_NOTUSE; /* Temperature sensor not used */
    (void)adcDev->ScanSet(ADC_GROUP_A, groupa_pin, ADC_TRIGER_ADTRG);

    groupb_pin.an_chans = ADC_MSEL_AN03 | ADC_MSEL_AN04;    /* Specify AN03 and AN04 for group B */
    groupb_pin.sensor   = ADC_SENSOR_NOTUSE; /* Temperature sensor not used */
    (void)adcDev->ScanSet(ADC_GROUP_B, groupb_pin, ADC_TRIGER_ELC);
    (void)adcDev->Control(AD_CMD_SET_GROUPB_INT, &adc_int);
                                             /* Enable a scan end interrupt of group B */

    while(1)
    {
        /* Wait for an input of A/D conversion start trigger
          [A/D conversion start trigger]
          Group A: ADTRG0 pin input
          Group B: ELC_S14AD event */
    }
}

/*******************************************************************************************************
* callback function
******************************************************************************************************/
static void callback(uint32_t event)
{
    switch(event)
    {
        case ADC_EVT_SCAN_COMPLETE_GROUPB:
        {
            /* Describe the processing to be performed when A/D conversion of group B is finished */
        }
        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:
        {
            /* Interrupt event does not occur because it is not enabled */
        }
        break;

    }
```

Figure 2-19 Example of AD_CMD_SET_GROUPB_INT Command

2.5.6 GCADI Interrupt Function Setting (AD_CMD_SET_GROUPC_INT)

The AD_CMD_SET_GROUPC_INT command sets the A/D scan end interrupt of group C (ADC140_GCADI). Specify a pointer to the e_adc_int_method_t type variable in the second argument. Table 2-18 shows settings of the e_adc_int_method_t type argument. Figure 2-21 shows a usage example of the AD_CMD_SET_GROUPC_INT command.

The following restrictions apply to this command.

- The interrupt function and auto-read function are controlled exclusively. When the auto-read function is using the A/D scan end interrupt (ADC140_GCADI) request, it cannot be used by this command (error is returned).
- The interrupt source (ADC140_GBADI) must be registered in the Nested Vector Interrupt Controller (hereinafter called the NVIC) by modifying r_system_cfg.h. An example of registering interrupts to the NVIC is shown in Figure 2-20. For the NVIC definition of each interrupt source and an example of registering interrupts, refer to section 5.6, Registering Interrupts to NVIC.
- The A/D scan end interrupt of group C is valid only in group-scan mode.

Table 2-18 Settings of e_adc_int_method_t Type Argument

Command	Description
ADC_INT_DISABLE	Disables interrupts.
ADC_INT_POLLING	Polling
ADC_INT_ENABLE	Enables interrupts.

```
...
#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 */
...
```

Figure 2-20 Registering Interrupts with NVIC

```
#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;  /* A/D conversion channel of group A */
    st_adc_pins_t groupb_pin;  /* A/D conversion channel of group B */
    st_adc_pins_t groupc_pin;  /* A/D conversion channel of group C */
    e_adc_int_method_t adc_int = ADC_INT_ENABLE;    /* Variable to enable interrupts */
    (void)adcDev->Open(ADC_GROUP_SCAN | ADC_14BIT, 0x10, callback); /* Initialize the S14AD driver */

    groupa_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01;    /* Specify AN00 and AN01 for group A */
    groupa_pin.sensor   = ADC_SENSOR_NOTUSE; /* Temperature sensor not used */
    (void)adcDev->ScanSet(ADC_GROUP_A, groupa_pin, ADC_TRIGER_ADTRG);

    groupb_pin.an_chans = ADC_MSEL_AN03 | ADC_MSEL_AN04;    /* Specify AN03 and AN04 for group B */
    groupb_pin.sensor   = ADC_SENSOR_NOTUSE; /* Temperature sensor not used */
    (void)adcDev->ScanSet(ADC_GROUP_B, groupb_pin, ADC_TRIGER_ELC);

    groupc_pin.an_chans = ADC_MSEL_AN22;   /* Specify AN22 for group C */
    groupc_pin.sensor   = ADC_SENSOR_NOTUSE; /* Temperature sensor not used */
    (void)adcDev->ScanSet(ADC_GROUP_C, groupc_pin, ADC_TRIGER_TMR);

    (void)adcDev->Control(AD_CMD_SET_GROUPC_INT, &adc_int);
                                             /* Enable a scan end interrupt of group C */
    while(1)
    {
        /* Wait for an input of the A/D conversion start trigger
          [A/D conversion start trigger]
          Group A: ADTRG0 pin input
          Group B: ELC_S14AD event
           Group C: TMR_TCORA (TMR compare match A) */
    }
}
/******************************************************************************************************
* callback function
******************************************************************************************************/
static void callback(uint32_t event)
{
    switch(event)
    {
        case ADC_EVT_SCAN_COMPLETE_GROUPC:
        {
            /* Describe the processing to be performed when A/D conversion of group C is finished */
        }
        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:
        {
            /* Interrupt event does not occur because it is not enabled */
        }
        break;
    }
}
```

Figure 2-21 Example of AD_CMD_SET_GROUPC_INT Command

2.5.7 WCMPM Interrupt Enable (AD_CMD_SET_WCMPM_INT)

The AD_CMD_SET_WCMPM_INT command sets the condition match interrupt (ADC140_WCMPM) of the window A/B compare function. Specify a pointer to the e_adc_int_method_t type variable in the second argument. Table 2-19 shows settings of the e_adc_int_method_t type argument. Figure 2-23 and Figure 2-24 show usage examples of the AD_CMD_SET_WCMPM_INT command.

The following restrictions apply to this command.

- The interrupt function and auto-read function are controlled exclusively. When the auto-read function is using the A/D scan end interrupt (ADC140_ WCMPM) request, it cannot be used by this command (error is returned).
- The interrupt source (ADC140_WCMPM) must be registered in the Nested Vector Interrupt Controller (hereinafter called the NVIC) by modifying r_system_cfg.h. An example of registering interrupts to the NVIC is shown in Figure 2-22. For the NVIC definition of each interrupt source and an example of registering interrupts, refer to section 5.6, Registering Interrupts to NVIC.
- The window A/B compare function must be set (complex conditions have to be set) in order to generate a condition match interrupt of the window A/B compare function. For information on setting a compare function, see sections 2.5.13, Compare Function Window A Setting (AD_CMD_SET_WINDOWA), and 2.5.14 Compare Function Window B Setting (AD_CMD_SET_WINDOWB).

Table 2-19 Settings of e_adc_int_method_t Type Argument

Command	Description
ADC_INT_DISABLE	Disables interrupts.
ADC_INT_POLLING	Polling
ADC_INT_ENABLE	Enable interrupts.

```
...
#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 */
...
```

Figure 2-22 Registering Interrupts with NVIC

```
#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;  /* A/D conversion channel of group A */
    st_adc_wina_t wina;    /* 2nd argument of AD_CMD_SET_WINDOWA */
    st_adc_winb_t winb;    /* 2nd argument of AD_CMD_SET_WINDOWB */
    st_adc_status_info_t result_status;    /* 2nd argument of AD_CMD_GET_AD_STATE */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, callback); /* Initialize the S14AD driver */

    groupa_pin.an_chans = ADC_MSEL_AN02 | ADC_MSEL_AN06;    /* Specify AN02 and AN06 for group A */
    groupa_pin.sensor   = ADC_SENSOR_NOTUSE; /* Temperature sensor not used */
    (void)adcDev->ScanSet(ADC_GROUP_A, groupa_pin, ADC_TRIGER_ADTRG);

    wina.inten             = ADC_DISABLE;      /* Disable the compare A interrupt */
    wina.cmp_mode        = ADC_CMP_WINDOW;    /* Compare windows */
    wina.level1            = 0x02000;           /* Comparison reference value 1 */
    wina.level2            = 0x00100;           /* Comparison reference value 2 */
    wina.chans.an_chans    = ADC_MSEL_AN02;  /* Specify AN02 as the target channel */
    wina.chans.sensor       = ADC_SENSOR_NOTUSE;    /* Temperature sensor not used */
    wina.chan_cond.an_chans = ADC_MSEL_AN02;
                         /* Channel-specific comparison condition: AN02 is specified within a range from
level 1 to level 2 */
    wina.chan_cond.sensor   = ADC_SENSOR_NOTUSE; /* Temperature sensor not used */
    (void)adcDev->Control(AD_CMD_SET_WINDOWA, &wina);    /* Set the window A comparison */

    winb.inten      = ADC_DISABLE;                  /* Disable the compare B interrupts */
    winb.winb_cond = ADC_WINB_WINDOW_BETWEEN;  /* Within a range of windows */
    winb.comb      = ADC_COMB_OR;                 /* Complex condition: OR condition */
    winb.level1      = 0x03000;                        /* Comparison reference value 1 */
    winb.level2      = 0x00010;                        /* Comparison reference value 2*/
    winb.channel    = ADC_SSEL_AN06;
    (void)adcDev->Control(AD_CMD_SET_WINDOWB, &winb);

    (void)adcDev->Control(AD_CMD_SET_WCMPM_INT, &adc_int);
                                             /* Enable a compare match interrupt for window A/B */
    while(1)
    {
        (void)adcDev->Start();  /* Start A/D conversion */

        do
        {
            (void)adcDev->Control(AD_CMD_GET_AD_STATE, & result_status);    /* Acquire the A/D status */
        }
        while (ADC_STATE_RUN == result_status.ad_info);    /* Wait for completion of A/D conversion */
    }
}
```

Figure 2-23 Example of AD_CMD_SET_WCMPM_INT Command (1/2)

```
/******************************************************************************************************
* callback function
******************************************************************************************************/
static void callback(uint32_t event)
{
    switch(event)
    {
        case ADC_EVT_WINDOW_CMP_MATCH:
        {
            /* Describe the processing to be performed when the window A/B compare conditions are met */
        }
        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:
        {
            /* Interrupt event does not occur because it is not enabled */
        }
        break;
    }
}
```

Figure 2-24 Example of AD_CMD_SET_WCMPM_INT Command (2/2)

2.5.8 WCMPUM Interrupt Enable (AD_CMD_SET_WCMPUM_INT)

AD_CMD_SET_WCMPUM_INT command sets the condition mismatch interrupt (ADC140_WCMPUM) of the window A/B compare function. Specify a pointer to the e_adc_int_method_t type variable in the second argument. Table 2-20 shows settings of the e_adc_int_method_t type argument. Figure 2-26 and Figure 2-27 show usage examples of the AD_CMD_SET_WCMPUM_INT command.

The following restrictions apply to this command.

- The interrupt function and auto-read function are controlled exclusively. When the auto-read function is using the A/D scan end interrupt (ADC140_ WCMPUM) request, it cannot be used by this command (error is returned).
- The interrupt source (ADC140_WCMPUM) must be registered in the Nested Vector Interrupt Controller (hereinafter called the NVIC) by modifying r_system_cfg.h. An example of registering interrupts to the NVIC is shown in Figure 2-25. For the NVIC definition of each interrupt source and an example of registering interrupts, refer to section 5.6, Registering Interrupts to NVIC.
- The window A/B compare function must be set (complex conditions have to be set) in order to generate a condition match interrupt of the window A/B compare function. For information on setting a compare function, see sections 2.5.13, Compare Function Window A Setting (AD_CMD_SET_WINDOWA), and 2.5.14 Compare Function Window B Setting (AD_CMD_SET_WINDOWB).

Table 2-20 Settings of e_adc_int_method_t Type Argument

Command	Description
ADC_INT_DISABLE	Disables interrupts.
ADC_INT_POLLING	Polling
ADC_INT_ENABLE	Enables interrupts.

```
...
#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 */
...
```

Figure 2-25 Registering Interrupts with NVIC

```
#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;  /* A/D conversion channel of group A */
    e_adc_int_method_t adc_int = ADC_INT_ENABLE;    /* Variable to enable interrupts */
    st_adc_wina_t wina;    /* 2nd argument of AD_CMD_SET_WINDOWA */
    st_adc_winb_t winb;    /* 2nd argument of AD_CMD_SET_WINDOWB */
    st_adc_status_info_t result_status;    /* 2nd argument of AD_CMD_GET_AD_STATE */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, callback); /* Initialize the S14AD driver */

    groupa_pin.an_chans = ADC_MSEL_AN02 | ADC_MSEL_AN06;    /* Specify AN02 and AN06 for group A */
    groupa_pin.sensor   = ADC_SENSOR_NOTUSE; /* Temperature sensor not used */
    (void)adcDev->ScanSet(ADC_GROUP_A, groupa_pin, ADC_TRIGER_ADTRG);

    wina.inten             = ADC_DISABLE;      /* Disable the compare A interrupt */
    wina.cmp_mode        = ADC_CMP_WINDOW;    /* Compare windows */
    wina.level1            = 0x02000;           /* Comparison reference value 1 */
    wina.level2            = 0x00100;           /* Comparison reference value 2 */
    wina.chans.an_chans    = ADC_MSEL_AN02;  /* Specify AN02 as the target channel */
    wina.chans.sensor       = ADC_SENSOR_NOTUSE;    /* Temperature sensor not used */
    wina.chan_cond.an_chans = ADC_MSEL_AN02;
                         /* Channel-specific comparison condition: AN02 is specified within a range from
level 1 to level 2 */
    wina.chan_cond.sensor   = ADC_SENSOR_NOTUSE; /* Temperature sensor not used */
    (void)adcDev->Control(AD_CMD_SET_WINDOWA, &wina);    /* Set window A comparison */

    winb.inten      = ADC_DISABLE;                  /* Disable the compare B interrupt */
    winb.winb_cond = ADC_WINB_WINDOW_BETWEEN;  /* Within a range of windows */
    winb.comb      = ADC_COMB_OR;                /* Complex condition: OR condition */
    winb.level1      = 0x03000;                       /* Comparison reference value 1 */
    winb.level2      = 0x00010;                       /* Comparison reference value 2*/
    winb.channel     = ADC_SSEL_AN06;
    (void)adcDev->Control(AD_CMD_SET_WINDOWB, &winb);

    (void)adcDev->Control(AD_CMD_SET_WCMPUM_INT, &adc_int);
                                            /* Enable the window A/B compare mismatch interrupt */
    while(1)
    {
        (void)adcDev->Start();  /* Start A/D conversion */

        do
        {
            (void)adcDev->Control(AD_CMD_GET_AD_STATE, & result_status);    /* Acquire the A/D status */
        }
        while (ADC_STATE_RUN == result_status.ad_info);    /* Wait for completion of A/D conversion */
    }
}
```

Figure 2-26 Example of AD_CMD_SET_WCMPUM_INT Command (1/2)

```
/*****************************************************************************************************
* callback function
*****************************************************************************************************/
static void callback(uint32_t event)
{
    switch(event)
    {
        case ADC_EVT_WINDOW_CMP_UNMATCH:
        {
            /* Describe the processing to be performed
               when the window A/B compare conditions are not met */
        }
        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:
        {
            /* Interrupt event does not occur because it is not enabled */
        }
        break;
    }
}
```

Figure 2-27 Example of AD_CMD_SET_WCMPUM_INT Command (2/2)

2.5.9 A/D Data Register Auto-Clear (AD_CMD_SET_AUTO_CLEAR)

The AD_CMD_SET_AUTO_CLEAR command enables or disables automatic clearing of A/D data registers. Specify a pointer to the uint8_t type variable in the second argument. Table 2-21 shows settings of the second argument. Figure 2-28 shows a usage example of the AD_CMD_SET_AUTO_CLEAR command.

Table 2-21 Settings of 2nd Argument

Setting	Description
ADC_ENABLE	Enables A/D data register auto-clear
ADC_DISABLE	Disables A/D data register auto-clear

```
#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 conversion channels */
    uint8_t arg;    /* 2nd argument of AD_CMD_SET_AUTO_CLEAR */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* Initialize the S14AD driver */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                                /* Use AN000, AN001, AN006 and AN022 for A/D conversion */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* Temperature sensor not used for A/D conversion */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);    /* Software trigger */

    arg      = ADC_ENABLE;    /* Enable A/D data register auto-clear */
    (void)adcDev->Control(AD_CMD_SET_AUTO_CLEAR, &arg);  /* Set A/D data register auto-clear */

    (void)adcDev->Start();  /* Start A/D conversion */

    while(1);
}
```

Figure 2-28 Example of AD_CMD_SET_AUTO_CLEAR Command

2.5.10 Sampling Time Setting (AD_CMD_SET_SAMPLING_XXX)

The sampling time setting commands (AD_CMD_SET_SAMPLING_XXX) are AD_CMD_SET_SAMPLING_AN0yy (Note), AD_CMD_SET_SAMPLING_AN016_AN017_AN020_TO_AN028, and AD_CMD_SET_SAMPLING_TEMP.

The default sampling time specified in the Open function is set as the sampling time of each channel. This command can be used to individually change the sampling time. Table 2-22 shows the targets for which sampling time can be specified for each command.

Table 2-22 List of Targets for Setting AD_CMD_SET_SAMPLING_XXX Commands

Command	Description
AD_CMD_SET_SAMPLING_AN0yy (Note)	AN0yy (Note)
AD_CMD_SET_SAMPLING_AN016_AN017_AN020_TO_AN028(Note2)	AN016, AN017, and AN020 to AN28
AD_CMD_SET_SAMPLING_AN016_AN017_AN020_AN021_VSC_VCC (Note3)	AN016, AN017, AN020, AN21, VSC_VCC voltage
AD_CMD_SET_SAMPLING_TEMP	Temperature sensor output

Note 1. 1500KB : yy = 0 to 6,16,17,20 to 28 256KB : yy = 0 to 7,16,17,20,21

Note 2. 1500KB Only

Note 3. 256KB Only

Specify a pointer to the e_adc_int_method_t type variable in the second argument. Specify a pointer to the uint8_t type variable in the second argument. Table 2-23 shows settings of the second argument. Figure 2-29 shows a setting example of sampling time.

Table 2-23 Settings of 2nd Argument

Setting	Description
2 to 255 (Note 1) (Note 2)	Sampling time

Note 1. When an out-of-range value is specified, an error is returned.

Note 2. Specify a value from 5 to 255 when the ADSCLKCR.SCLKEN bit is set to 0 (sub-clock mode is disabled). Specify a value from 2 to 8 when the ADSCLKCR.SCLKEN bit is set to 1 (sub-clock mode is enabled).

```
#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 conversion channels */
    uint8_t arg;    /* 2nd argument of AD_CMD_SET_SAMPLING_001 */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* Initialize the S14AD driver */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                         /* Use AN000, AN001, AN006 and AN022 for A/D conversion */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* Temperature sensor not used for A/D conversion */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);    /* Software trigger */

    arg      = 5;    /* Sampling time (5) */
    (void)adcDev->Control(AD_CMD_SET_SAMPLING_AN001, &arg);  /* Set the sampling time for AN001 */

    (void)adcDev->Start();  /* Start A/D conversion */

    while(1);
}
```

Figure 2-29 Example of AD_CMD_SET_SAMPLING_AN001 Command

2.5.11 Disconnection Detection Assist Setting (AD_CMD_SET_ADNDIS)

The AD_CMD_SET_ADNDIS command sets the disconnection detection assist function. Specify a pointer to the uint8_t type variable in the second argument. Table 2-24 shows settings of the second argument. Figure 2-30 shows a usage example of the AD_CMD_SET_ADNDIS command.

The following restriction applies to this command.

- Can be executed only when A/D conversion is halted (ADCSR.ADST = 0). For state transitions of the A/D conversion operation, see section 2.9, State Transitions.

Table 2-24 Settings of 2nd Argument

Setting	Description
ADC_DDA_OFF	Disables disconnection detection assist function.
ADC_DDA_PRECHARGE (Note 1)	Precharge
ADC_DDA_DISCHARGE (Note 1)	Discharge

Note 1. When selecting either precharge or discharge, the period of precharge/discharge should be specified together.

Example: To set a precharge period of five cycles:

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 conversion channels */
    int8_t arg;    /* 2nd argument of AD_CMD_SET_ADNDIS */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* Initialize the S14AD driver */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                                /* Use AN000, AN001, AN006 and AN022 for A/D conversion */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* Temperature sensor not used for A/D conversion */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);    /* Software trigger */

    arg      = ADC_DDA_PRECHARGE | 5;    /* Precharge (charge period (5)) */
    (void)adcDev->Control(AD_CMD_SET_ADNDIS, &arg);  /* Set the disconnection detection assist */

    (void)adcDev->Start();  /* Start A/D conversion */

    while(1);
}
```

Figure 2-30 Example of AD_CMD_SET_ADNDIS Command

2.5.12 Group Priority Operation Setting (AD_CMD_SET_GROUP_PRIORITY)

The AD_CMD_SET_GROUP_PRIORITY command sets group priority operation. Specify a pointer to the e_adc_diag_t type variable in the second argument. Table 2-25 shows settings of the e_adc_gsp_t type argument. Figure 2-31 shows a usage example of the AD_CMD_SET_GROUP_PRIORITY command.

The following restrictions apply to this command.

- Can be executed only when A/D conversion is stopped (ADCSR.ADST = 0). For state transitions of the A/D conversion operation, see section 2.9, State Transitions.
- Can only be executed in group-scan mode. When executed in single-scan mode or continuous-scan mode, an error is returned.

Table 2-25 Settings of e_adc_gsp_t Type Argument

Setting	Description
ADC_GSP_PRIORYTY_OFF	Group priority operation disabled
ADC_GSP_WAIT_TRG	Scanning for a low-priority group not restarted
ADC_GSP_SCAN_BIGIN	Scanning for a low-priority group restarted
ADC_GSP_SCAN_RESTART	Rescanning from the channel for which A/D conversion is not completed
ADC_GSP_WAIT_TRG_CONT_SCAN	Continuous single scanning for low-priority group enabled and low-priority group restart disabled (Note)
ADC_GSP_SCAN_BIGIN_CONT_SCAN	Continuous single scanning for low-priority group enabled and low-priority group restart enabled (Note)
ADC_GSP_SCAN_RESTART_CONT_SCAN	Continuous single scanning for low-priority group enabled and rescanning is started from the channel for which A/D conversion has not completed (Note)

Note. When continuous single scanning for a low-priority group is enabled, the A/D conversion start trigger for the group with the lowest priority must be deselected. To deselect the trigger source, see section 2.4.2, Setting A/D Input Pins and A/D Conversion Start Trigger by ScanSet.

```
#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;  /* A/D conversion channel of group A */
    st_adc_pins_t groupb_pin;  /* A/D conversion channel of group B */
    e_adc_gsp_t arg;    /* Variable to enable interrupts */

    (void)adcDev->Open(ADC_GROUP_SCAN | ADC_14BIT, 0x10, callback); /* Initialize the S14AD driver */

    groupa_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01;    /* Specify AN00 and AN01 for group A */
    groupa_pin.sensor   = ADC_SENSOR_NOTUSE; /* Temperature sensor not used */
    (void)adcDev->ScanSet(ADC_GROUP_A, groupa_pin, ADC_TRIGER_ADTRG);

    groupb_pin.an_chans = ADC_MSEL_AN03 | ADC_MSEL_AN04;    /* Specify AN03 and AN04 for group B */
    groupb_pin.sensor   = ADC_SENSOR_NOTUSE; /* Temperature sensor not used */
    (void)adcDev->ScanSet(ADC_GROUP_B, groupb_pin, ADC_TRIGER_ELC);

    arg = ADC_GSP_WAIT_TRG;    /* Disable the low-priority group restart */
    (void)adcDev->Control(AD_CMD_SET_GROUP_PRIORITY, &arg);  /* Specify the group priority operation */

    while(1)
    {
        /* Wait for an input of A/D conversion start trigger
        [A/D conversion start trigger]
        Group A: ADTRG0 pin input
        Group B: ELC_S14AD event */
    }

}
```

Figure 2-31 Example of AD_CMD_SET_GROUP_PRIORITY Command

2.5.13 Compare Function Window A Setting (AD_CMD_SET_WINDOWA)

The AD_CMD_SET_WINDOWA command sets the digital comparison function (window A). Specify a pointer to the st_adc_wina_t type variable in the second argument. Table 2-26 shows settings of the st_adc_wina_t type argument. Figure 2-34 shows a usage example of the AD_CMD_SET_WINDOWA command.

The following restrictions apply to this command.

- Can be executed only when A/D conversion is stopped (ADCSR.ADST = 0). For state transitions of the A/D conversion operation, see section 2.9, State Transitions.
- When the temperature sensor is selected in window B, it cannot be used in window A operation (error is returned).
- The same channel cannot be set in window A and window B (error is returned).
- When ADC_CMPAI_SNOOZE_USE (Note) is set to 0 in r_adc_cfg.h, the comparison condition match interrupt of window A (ADC140_CMPAI) has to be set regardless of whether interrupts are enabled or disabled. For information on setting interrupts, see Figure 2-33. For the NVIC definition of each interrupt source and an example of registering interrupts, see section 5.6, Registering Interrupts to NVIC.
- When ADC_CMPAI_SNOOZE_USE (Note) is set to 1 in r_adc_cfg.h, setting of the comparison condition match interrupt of window A (ADC140_CMPAI) can be skipped. When the setting is skipped, the CMPAI interrupt is disabled.

Note. For information on ADC_CMPAI_SNOOZE_USE, see section 3.5.14, CMPAI Snooze Mode .

Table 2-26 Settings of st_adc_wina_t Type Argument

Element	Setting	Description
Inten (Note 1)	ADC_ENABLE	Enables the comparison condition match interrupt for window A (ADC140_CMPAI)
	ADC_DISABLE	Disables the comparison condition match interrupt for window A (ADC140_CMPAI)
cmp_mode	ADC_CMP_OFF	Comparison not performed
	ADC_CMP_LEVEL	Level comparison
	ADC_CMP_WINDOW	Window comparison
level1	uint16_t type value (Note 2)	Specifies the comparison level (level 1) (Note 3)
level2	uint16_t type value (Note 2)	Specifies the comparison level (level 2) (Note 3) (Note 4)
chans. an_chans	Combination of ADC_MSEL_ANn channels (1500KB Group : n = 00 to 06,16,17,20 to 28 256KB Group : n = 00 to 07,16,17,20 to 21(Note5))	Target channels for A/D-converted value addition/average [Setting example] /* AN000 and AN016 are used */ arg.chans.anchans = ADC_MSEL_AN00 | ADC_MSEL_AN16;
chans. sensor	ADC_SENSOR_NOTUSE	Temperature sensor output not used
	ADC_MSEL_TEMP	Temperature sensor output used
chan_cond. an_chans	Combination of ADC_MSEL_ANn channels (1500KB Group : n = 00 to 06,16,17,20 to 28 256KB Group : n = 00 to 07,16,17,20 to 21(Note5))	Specifies the channel-specific comparison condition [Level comparison] Specified channel: Level 1 or higher Other channels: Level 1 or lower [Window comparison] Specified channel: Within a range from level 1 to level 2 Other channels: Outside a range from level 1 to level 2 [Setting example] /* AN000 and AN016 */ arg. chan_cond.anchans = ADC_MSEL_AN00 | ADC_MSEL_AN16;
chan_cond. sensor	ADC_SENSOR_NOTUSE	Specifies the comparison conditions of temperature sensor output [Level comparison] Level 1 or lower [Window comparison] Outside a range from level 1 to level 2
	ADC_MSEL_TEMP	Specifies the comparison conditions of temperature sensor output [Level comparison] Level 1 or higher [Window comparison] Within a range from level 1 to level 2

Note 1. When CMPAI snooze mode is used, the interrupt setting is invalid.

Note 2. The setting and format of the reference value for the comparison level differ depending on the A/D conversion accuracy, format of A/D data registers, and A/D-converted value addition mode setting.
When A/D-converted value addition mode is not selected:
- Flush-right data with 14-bit accuracy: Lower 14 bits (b13 to b0) are valid.
- Flush-right data with 12-bit accuracy: Lower 12 bits (b11 to b0) are valid.
- Flush-left data with 14-bit accuracy: Upper 14 bits (b15 to b2) are valid.
- Flush-left data with 12-bit accuracy: Upper 12 bits (b15 to b4) are valid.

When A/D-converted value average mode is selected:
- Flush-right data with 14-bit accuracy: All bits (b15 to b0) are valid
- Flush-right data with 12-bit accuracy: Lower 14 bits (b13 to b0) are valid.
- Flush-left data with 14-bit accuracy: All bits (b15 to b0) are valid.
- Flush-left data with 12-bit accuracy: Upper 14 bits (b15 to b2) are valid.

Note 3. The larger value among level 1 and level 2 is the upper reference for window comparison when window comparison has been specified.

Note 4. Invalid at level comparison.

Note 5. Only 256KB group can be specified with ADC_MSEL_VSC_VCC

To make settings for using CMPAI snooze mode, enable the CMPAI snooze mode in r_adc_cfg.h. Table 2-27 shows the CMPAI snooze mode definition. Figure 2-32 shows the setting of snooze mode.

Table 2-27 Definition of CMPAI Snooze Mode

Definition	Initial Value	Description
ADC_CMPAI_SNOOZE_USE	0	0: Compare function window A is used for CMPAI interrupt (CMPAI interrupt is set). (Note) 1: Compare function window A is used only in Snooze mode, which is a low power consumption mode (CMPAI interrupt setting is skipped).

Note. Set this to 0 when using compare function window A in Snooze mode, which is a low power consumption mode, and also setting the CMPAI interrupt. When 1is set, the CMPAI interrupt cannot be used.

```
...
#define ADC_CMPAI_SNOOZE_USE  (1)
...
```

Figure 2-32 Setting for Using CMPAI Snooze Mode

When ADC_CMPAI_SNOOZE_USE is set to 1, the setting of the comparison condition match interrupt for window A (ADC140_CMPAI) can be skipped. When the setting is skipped, the CMPAI interrupt is disabled.

When ADC_CMPAI_SNOOZE_USE is set to 0, the interrupt source (ADC140_CMPAI) must be registered in the Nested Vector Interrupt Controller (hereinafter called the NVIC) by modifying r_system_cfg.h. Figure 2-33 shows an example for registering interrupts with NVIC.

```
...
#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 */
...
```

Figure 2-33 Registering Interrupts with NVIC

```
#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;  /* A/D conversion channel of group A */
    st_adc_wina_t wina;    /* 2nd argument of AD_CMD_SET_WINDOWA */
    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, callback); /* Initialize the S14AD driver */

    groupa_pin.an_chans = ADC_MSEL_AN02 | ADC_MSEL_AN06;    /* Specify AN02 and AN06 for group A */
    groupa_pin.sensor   = ADC_SENSOR_NOTUSE; /* Temperature sensor not used */
    (void)adcDev->ScanSet(ADC_GROUP_A, groupa_pin, ADC_TRIGER_ADTRG);

    wina.inten             = ADC_ENABLE;      /* Enable compare A interrupt */
    wina.cmp_mode        = ADC_CMP_WINDOW;    /* Compare windows */
    wina.level1            = 0x02000;           /* Comparison reference value 1 */
    wina.level2            = 0x00100;           /* Comparison reference value 2 */
    wina.chans.an_chans    = ADC_MSEL_AN02;  /* Specify AN02 as the target channel */
    wina.chans.sensor       = ADC_SENSOR_NOTUSE;    /* Temperature sensor not used */
    wina.chan_cond.an_chans = ADC_MSEL_AN02;
                         /* Channel-specific comparison condition: AN02 is specified within a range from
level 1 to level 2 */
    wina.chan_cond.sensor   = ADC_SENSOR_NOTUSE; /* Temperature sensor not used */
    (void)adcDev->Control(AD_CMD_SET_WINDOWA, &wina);    /* Set the window A comparison */

    (void)adcDev->Start();  /* Start A/D conversion */

    while(1);
}

/****************************************************************************************************
* callback function
****************************************************************************************************/
static void callback(uint32_t event)
{
    switch(event)
    {
        case ADC_EVT_CONDITION_MET:
        {
            /* Describe the processing to be performed when the window A comparison condition is met */
        }
        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:
        {
            /* Interrupt event does not occur because it is not enabled */
        }
        break;

    }
}
```

Figure 2-34 Example of AD_CMD_SET_WINDOWA Command

2.5.14 Compare Function Window B Setting (AD_CMD_SET_WINDOWB)

The AD_CMD_SET_WINDOWB command sets the digital comparison function (window B). It also sets the complex conditions for window A/B compare function. Specify a pointer to the st_adc_winb_t type variable in the second argument. Table 2-28 shows settings of the st_adc_winb_t type argument. Figure 2-37 and Figure 2-38 show usage examples of the AD_CMD_SET_WINDOWB command.

The following restrictions apply to this command.

- Can be executed only when A/D conversion is stopped (ADCSR.ADST = 0). For state transitions of the A/D conversion operation, see section 2.9, State Transitions.
- To use the compare function event output (ADC140_WCMPM/ADC140_WCMPUM), set the single-scan mode.
- The same channel cannot be set in window A and window B (error is returned).
- When the temperature sensor is selected for window A, window B operations are disabled (error is returned).
- When ADC_CMPBI_SNOOZE_USE (Note) is set to 0 in r_adc_cfg.h, the comparison condition match interrupt of window B (ADC140_CMPBI) has to be set regardless of whether interrupts are enabled or disabled. For information on setting interrupts, see Figure 2-36. For the NVIC definition of each interrupt source and an example of registering interrupts, see section 5.6, Registering Interrupts to NVIC.
- When ADC_CMPBI_SNOOZE_USE (Note) is set to 1 in r_adc_cfg.h, setting of the comparison condition match interrupt of window B (ADC140_CMPBI) can be skipped. When the setting is skipped, the CMPBI interrupt is disabled.

Note. For information on ADC_CMPBI_SNOOZE_USE, see section 3.5.15, CMPBI.

Table 2-28 Settings of st_adc_winb_t Type Argument

Element	Setting	Description
Inten (Note 1)	ADC_ENABLE	Enables the comparison condition match interrupt for window B (ADC140_CMPBI)
	ADC_DISABLE	Disables the comparison condition match interrupt for window B (ADC140_CMPBI)
winb_cond	ADC_WINB_LEVEL_BELOW	Level 1 or lower
	ADC_WINB_LEVEL_ABOVE	Level 1 or higher
	ADC_WINB_WINDOW_OUTSIDE	Outside a range from level 1 to level 2
	ADC_WINB_WINDOW_BETWEEN	Within a range from level 1 to level 2
comb	ADC_COMB_OR	Window A/B complex conditions setting Outputs ADC140_WCMPM when window A OR window B comparison conditions are met; otherwise, outputs ADC140_WCMPUM.
	ADC_COMB_EXOR	Window A/B complex conditions setting Outputs ADC140_WCMPM when window A EXOR window B comparison conditions are met; otherwise, outputs ADC140_WCMPUM.
	ADC_COMB_AND	Window A/B complex conditions setting Outputs ADC140_WCMPM when window A AND window B comparison conditions are met; otherwise, outputs ADC140_WCMPUM.
	ADC_COMB_NON_EVENT	Window A/B complex conditions setting Outputs an event only in window A (Note 2)
	ADC_COMB_OFF	Window A/B complex conditions are invalid
level1	uint16_t type value (Note 3)	Specifies the comparison level (level 1) (Note 4)
level2	uint16_t type value (Note 3)	Specifies the comparison level (level 2) (Note 4) (Note 5)
channel	ADC_SSEL_ANx (1500KB Group : n = 00 to 06,16,17,20 to 28 256KB Group : n = 00 to 07,16,17,20 to 21(Note 5))	Selects one channel for A/D-converted value addition/averaging
	ADC_SSEL_TEMP	Temperature sensor output

Note 1. When CMPBI snooze mode is used, the interrupt setting is invalid.

When the compare function event output (ADC140_WCMPM or ADC140_WCMPUM) is used only in window A, specify ADC_COMB_NON_EVENT. When ADC_COMB_NON_EVENT is specified, the window B settings become as shown below regardless of the values set in the arguments.

- Complex condition of window A and window B: OR condition (ADCMPCR.CMPAB = 00b)
- Compared channel of window B: Not selected (ADCMPBNSR.CMPCHB = 0x3F)
- Comparison condition of window B: Always mismatch (0 < results < 0) (ADCMPCR.WCMPE = 1, ADWINLLB[15:0] = ADWINULB[15:0] = 0000h, ADCMPBNSR.CMPLB = 1)

Note 2. The setting and format of the reference value for the comparison level differ depending on the A/D conversion accuracy, format of A/D data registers, and A/D-converted value addition mode setting.

When A/D-converted value addition mode is not selected:

- Flush-right data with 14-bit accuracy: Lower 14 bits (b13 to b0) are valid.
- Flush-right data with 12-bit accuracy: Lower 12 bits (b11 to b0) are valid.
- Flush-left data with 14-bit accuracy: Upper 14 bits (b15 to b2) are valid.
- Flush-left data with 12-bit accuracy: Upper 12 bits (b15 to b4) are valid.

When A/D-converted value average mode is selected:

- Flush-right data with 14-bit accuracy: All bits (b15 to b0) are valid
- Flush-right data with 12-bit accuracy: Lower 14 bits (b13 to b0) are valid.
- Flush-left data with 14-bit accuracy: All bits (b15 to b0) are valid.
- Flush-left data with 12-bit accuracy: Upper 14 bits (b15 to b2) are valid.

Note 3. The larger value among level 1 and level 2 is the upper reference for window comparison when window comparison has been specified.

Note 4. Invalid at level comparison.

Note 5. Only 256KB group, ADC_SSEL_VSC_VCC can also be selected

To make settings for using CMPBI snooze mode, enable the CMPBI snooze mode in r_adc_cfg.h. Table 2-29 shows the CMPBI snooze mode definition. Figure 2-35 shows the setting of snooze mode.

Table 2-29 Definition of CMPBI Snooze Mode

Definition	Initial Value	Description
ADC_CMPBI_SNOOZE_USE	0	0: Compare function window A is used for CMPBI interrupt (CMPBI interrupt is set). (Note) 1: Compare function window B is used only in Snooze mode, which is a low power consumption mode (CMPBI interrupt setting is skipped).

Note. Set this to 0 when using compare function window B in Snooze mode, which is a low power consumption mode, and also setting the CMPBI interrupt. When 1is set, the CMPBI interrupt cannot be used.

```
...
#define ADC_CMPBI_SNOOZE_USE  (1)
...
```

Figure 2-35 Setting for Using CMPBI Snooze Mode

When ADC_CMPBI_SNOOZE_USE is set to 1, the setting of the comparison condition match interrupt for window A (ADC140_CMPBI) can be skipped. When the setting is skipped, the CMPBI interrupt is disabled.

When ADC_CMPBI_SNOOZE_USE is set to 0, the interrupt source (ADC140_CMPBI) must be registered in the Nested Vector Interrupt Controller (hereinafter called the NVIC) by modifying r_system_cfg.h. Figure 2-36 shows an example for registering the interrupts with NVIC.

```
...
#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 */
...
```

Figure 2-36 Registering Interrupts with NVIC

```
#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;  /* A/D conversion channel of group A */
    st_adc_wina_t wina;    /* 2nd argument of AD_CMD_SET_WINDOWA */
    st_adc_winb_t winb;    /* 2nd argument of AD_CMD_SET_WINDOWB */
    st_adc_status_info_t result_status;    /* 2nd argument of AD_CMD_GET_AD_STATE */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, callback); /* Initialize the S14AD driver */

    groupa_pin.an_chans = ADC_MSEL_AN02 | ADC_MSEL_AN06;    /* Specify AN02 and AN06 for group A */
    groupa_pin.sensor   = ADC_SENSOR_NOTUSE; /* Temperature sensor not used */
    (void)adcDev->ScanSet(ADC_GROUP_A, groupa_pin, ADC_TRIGER_ADTRG);

    wina.inten             = ADC_DISABLE;      /* Disable the compare A interrupt */
    wina.cmp_mode        = ADC_CMP_WINDOW;    /* Compare windows */
    wina.level1            = 0x02000;           /* Comparison reference value 1 */
    wina.level2            = 0x00100;           /* Comparison reference value 2 */
    wina.chans.an_chans    = ADC_MSEL_AN02;  /* Specify AN02 as the target channel */
    wina.chans.sensor       = ADC_SENSOR_NOTUSE;    /* Temperature sensor not used */
    wina.chan_cond.an_chans = ADC_MSEL_AN02;
                         /* Channel-specific comparison condition: AN02 is specified within a range from
level 1 to level 2 */
    wina.chan_cond.sensor   = ADC_SENSOR_NOTUSE; /* Temperature sensor not used */
    (void)adcDev->Control(AD_CMD_SET_WINDOWA, &wina);    /* Set the window A comparison */

    winb.inten      = ADC_ENABLE;                  /* Enable the compare B interrupt */
    winb.winb_cond = ADC_WINB_WINDOW_BETWEEN;  /* Within a range of windows */
    winb.comb      = ADC_COMB_OR;                /* Complex condition: OR condition */
    winb.level1      = 0x03000;                     /* Comparison reference value 1 */
    winb.level2      = 0x00010;                     /* Comparison reference value 2*/
    winb.channel    = ADC_SSEL_AN06;
    (void)adcDev->Control(AD_CMD_SET_WINDOWB, &winb); /* Set the window B comparison */

    while(1)
    {
        (void)adcDev->Start();  /* Start A/D conversion */

        do
        {
            (void)adcDev->Control(AD_CMD_GET_AD_STATE, & result_status);    /* Acquire the A/D status */
        }
        while (ADC_STATE_RUN == result_status.ad_info);    /* Wait for completion of A/D conversion */
    }
}
```

Figure 2-37 Example of AD_CMD_SET_WINDOWB Command (1/2)

```
/*******************************************************************************************************
* callback function
*******************************************************************************************************/
static void callback(uint32_t event)
{
    switch(event)
    {
        case ADC_EVT_CONDITION_METB:
        {
            /* Describe the processing to be performed when the window B comparison condition is met */
        }
        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:
        {
            /* Interrupt event does not occur because it is not enabled */
        }
        break;
    }
}
```

Figure 2-38 Example of AD_CMD_SET_WINDOWB Command (2/2)

2.5.15 Scan End Event (ADC140_ELC) Generating Condition Setting

The AD_CMD_SET_ELC command sets the generating condition of a scan end event (ADC140_ELC). Specify a pointer to the e_adc_elc_mode_t type variable in the second argument. Table 2-30 shows settings of the e_adc_elc_mode_t type argument. Figure 2-39 shows a usage example of the AD_CMD_SET_ELC command.

Table 2-30 Settings of e_adc_elc_mode_t Type Argument

Setting	Description
ADC_ELC_GROUPA	Generates an event upon completion of scanning other than group B and group C scans.
ADC_ELC_GROUPB	Generates an event on completion of group B scan.
ADC_ELC_GROUPC	Generates an event on completion of group C scan.
ADC_ELC_ALL	Generates an event on completion of all scans.

```
#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 conversion channels */
    e_adc_elc_mode_t arg;    /* 2nd argument of AD_CMD_SET_ELC */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* Initialize the S14AD driver */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                    /* Use AN000, AN001, AN006 and AN022 for A/D conversion */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* Temperature sensor not used for A/D conversion */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_ADTRG);

    groupb_pin.an_chans = ADC_MSEL_AN03 | ADC_MSEL_AN04;    /* Specify AN03 and AN04 for group B */
    groupb_pin.sensor   = ADC_SENSOR_NOTUSE; /* Temperature sensor not used */
    (void)adcDev->ScanSet(ADC_GROUP_B, groupb_pin, ADC_TRIGER_ELC);


    arg                = ADC_ELC_GROUPB;    /* Generate an event on completion of group B scan */
    (void)adcDev->Control(AD_CMD_SET_ELC, &arg);
                             /* Set the generating condition of scan end event (ADC140_ELC) */

    (void)adcDev->Start();  /* Start A/D conversion */

    while(1)
    {
        /* Wait for an input of the A/D conversion start trigger
          [A/D conversion start trigger]
          Group A: ADTRG0 pin input
          Group B: ELC_S14AD event */
    }
}
```

Figure 2-39 Example of AD_CMD_SET_ELC Command

2.5.16 Acquiring Comparison Results of A/D Compare Function (AD_CMD_GET_CMP_RESULT)

The AD_CMD_GET_CMP_RESULT command acquires the comparison results of A/D compare functions. The acquired comparison results of A/D compare function are stored in the second argument. Specify a pointer to the st_adc_cmp_result_t type variable in the second argument. Table 2-31 shows the information stored in the st_adc_cmp_result_t type argument. Figure 2-40 shows a usage example of the AD_CMD_GET_CMP_RESULT command.

Table 2-31 Information Stored in st_adc_cmp_result_t Type Argument

Setting	Description
wina_result. an_chans	The values of the A/D compare function window A channel status register 0 (ADCMPSR0) and window A channel status register 1 (ADCMPSR1) are stored. Each bit of the stored information corresponds to an A/D channel (bit 6 and bit 0 indicate the comparison results of AN006 and AN000). [Comparison result for each pin] 0: Comparison conditions are not met 1: Comparison conditions are met
wina_result. sensor	The value of the A/D compare function window A extended input channel status register (ADCMPSER) is stored. [Comparison result of temperature sensor output] 0: Comparison conditions are not met 1: Comparison conditions are met
winb_result	The value of the A/D compare function window B channel status register (ADCMPBSR) is stored. [Comparison result of channel selected for window B] 0: Comparison conditions are not met 1: Comparison conditions are met
comb_result	The value of the A/D compare function window A/B channel status monitor register (ADWINMON) is stored. [Combined results (bit 0)] 0: Window A/B complex conditions are not met. 1: Window A/B complex conditions are met. [Comparison result monitor A (bit 4)] 0: Window A comparison conditions are not met. 1: Window A comparison conditions are met. [Comparison result monitor B (bit 5)] 0: Window B comparison conditions are not met. 1: Window B comparison conditions are met.

```
#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;  /* A/D conversion channel of group A */
    st_adc_wina_t wina;    /* 2nd argument of AD_CMD_SET_WINDOWA */
    st_adc_winb_t winb;    /* 2nd argument of AD_CMD_SET_WINDOWB */
    st_adc_cmp_result_t cmp_result; /* 2nd argument of AD_CMD_GET_CMP_RESULT */
    st_adc_status_info_t result_status;    /* 2nd argument of AD_CMD_GET_AD_STATE */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* Initialize the S14AD driver */

    groupa_pin.an_chans = ADC_MSEL_AN02 | ADC_MSEL_AN06;    /* Specify AN02 and AN06 for group A */
    groupa_pin.sensor   = ADC_SENSOR_NOTUSE; /* Temperature sensor not used */
    (void)adcDev->ScanSet(ADC_GROUP_A, groupa_pin, ADC_TRIGER_ADTRG);

    wina.inten             = ADC_DISABLE;      /* Disable the compare A interrupt */
    wina.cmp_mode        = ADC_CMP_WINDOW;    /* Compare windows */
    wina.level1            = 0x02000;           /* Comparison reference value 1 */
    wina.level2            = 0x00100;           /* Comparison reference value 2 */
    wina.chans.an_chans    = ADC_MSEL_AN02;  /* Specify AN02 as the target channel */
    wina.chans.sensor       = ADC_SENSOR_NOTUSE;    /* Temperature sensor not used */
    wina.chan_cond.an_chans = ADC_MSEL_AN02;
                         /* Channel-specific comparison condition: AN02 is specified within a range from
level 1 to level 2 */
    wina.chan_cond.sensor   = ADC_SENSOR_NOTUSE; /* Temperature sensor not used */
    (void)adcDev->Control(AD_CMD_SET_WINDOWA, &wina);    /* Set the window A comparison */

    winb.inten      = ADC_DISABLE;                  /* Disable the compare B interrupt */
    winb.winb_cond = ADC_WINB_WINDOW_BETWEEN;  /* Within a range of windows */
    winb.comb      = ADC_COMB_OR;                /* Complex condition: OR condition */
    winb.level1      = 0x03000;                      /* Comparison reference value 1 */
    winb.level2      = 0x00010;                      /* Comparison reference value 2*/
    winb.channel    = ADC_SSEL_AN06;
    (void)adcDev->Control(AD_CMD_SET_WINDOWB, &winb); /* Set the window B comparison */

    while(1)
    {
        (void)adcDev->Start();  /* Start A/D conversion */

        do
        {
            (void)adcDev->Control(AD_CMD_GET_AD_STATE, & result_status);    /* Acquire the A/D status */
        }
        while (ADC_STATE_RUN == result_status.ad_info);    /* Wait for completion of A/D conversion */
        (void)adcDev->Control(AD_CMD_GET_CMP_RESULT, & cmp_result);
        if (1 == cmp_result. winb_result)
        {
            /* Describe the processing to be performed when the compare function window B condition is
met */
        }
    }
}
```

Figure 2-40 Example of AD_CMD_GET_CMP_RESULT Command

2.5.17 A/D Converter State Acquisition (AD_CMD_GET_AD_STATE)

The AD_CMD_GET_AD_STATE acquires the state of the A/D converter. The acquired state information is stored in the second argument. Specify a pointer to the st_adc_status_info_t type variable in the second argument. Table 2-32 shows the information stored in the st_adc_status_info_t type argument. Figure 2-41 shows a usage example of the AD_CMD_GET_AD_STATE command.

Table 2-32 Information Stored in st_adc_status_info_t Type Argument

Setting	Description
ad_info	The state information of A/D conversion operation is stored. ADC_STATE_STOP: A/D conversion halted (ADCSR.ADST = 0) ADC_STATE_RUN: A/D conversion in progress (ADCSR.ADST = 1)
groupa_info	The state information of group A operation is stored. ADC_CONV_NOT_POLLING: Polling not set ADC_CONV_INCOMPLETE: A/D scan of group A not completed ADC_CONV_COMPLETE: A/D scan of group A completed ADC_CONV_GET_FAILED: Failed in acquiring the state of group A
groupb_info	The state information of group B operation is stored. ADC_CONV_NOT_POLLING: Polling not set ADC_CONV_INCOMPLETE: A/D scan of group B not completed ADC_CONV_COMPLETE: A/D scan of group B completed ADC_CONV_GET_FAILED: Failed in acquiring the state of group B
groupc_info	The state information of group C operation is stored. ADC_CONV_NOT_POLLING: Polling not set ADC_CONV_INCOMPLETE: A/D scan of group C not completed ADC_CONV_COMPLETE: A/D scan of group C completed ADC_CONV_GET_FAILED: Failed in acquiring the state of group C
wcmpm_info	The information of window A/B compare match event is stored. ADC_CONV_NOT_POLLING: Polling not set ADC_CONV_WCMP_DETECT: Window A/B compare match event detected ADC_CONV_WCMP_NOT_DETECT: Window A/B compare match event not detected ADC_CONV_GET_FAILED: Failed in acquiring the state of a window A/B compare match event
wcmpum_info	The information of window A/B compare mismatch event is stored. ADC_CONV_NOT_POLLING: Polling not set ADC_CONV_WCMP_DETECT: Window A/B compare mismatch event detected ADC_CONV_WCMP_NOT_DETECT: Window A/B compare mismatch event not detected ADC_CONV_GET_FAILED: Failed in acquiring the state of a window A/B compare match event

```
#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 conversion channels */
    st_adc_status_info_t result_status;    /* 2nd argument of AD_CMD_GET_AD_STATE */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* Initialize the S14AD driver */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                    /* Use AN000, AN001, AN006 and AN022 for A/D conversion */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* Temperature sensor not used for A/D conversion */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);

    while(1)
    {
        (void)adcDev->Start();  /* Start A/D conversion */
        do
        {
            (void)adcDev->Control(AD_CMD_GET_AD_STATE, & result_status);    /* Acquire the A/D status */
        }
        while (ADC_STATE_RUN == result_status.ad_info);    /* Wait for completion of A/D conversion */
    }
}
```

Figure 2-41 Example of AD_CMD_GET_AD_STATE Command

2.5.18 Low-Potential Side Reference Voltage Setting (AD_CMD_USE_VREFL0)

The AD_CMD_USE_VREFL0 command sets VREFL0 as the reference voltage on the low-potential side. Specify NULL in the second argument. Figure 2-42 shows a usage example of the AD_CMD_USE_VREFL0 command.

The following restriction applies to this command.

- Can be executed only when A/D conversion is stopped (ADCSR.ADST = 0). For state transitions of the A/D conversion operation, see section 2.9, State Transitions.

```
#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 conversion channels */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* Initialize the S14AD driver */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                        /* Use AN000, AN001, AN006, and AN022 for A/D conversion */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* Temperature sensor not used for A/D conversion */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);    /* Software trigger */

    (void)adcDev->Control(AD_CMD_USE_VREFL0, NULL); /* Select VREFL0 as reference voltage on the low-
potential side */

    (void)adcDev->Start();  /* Start A/D conversion */

    while(1);
}
```

Figure 2-42 Example of AD_CMD_USE_VREFL0 Command

2.5.19 High-Potential Side Reference Voltage Setting (AD_CMD_USE_VREFH0)

The AD_CMD_USE_VREFH0 command sets VREFH0 as the reference voltage on the high-potential side. Specify NULL in the second argument. Figure 2-43 shows a usage example of the AD_CMD_USE_VREFH0 command.

- The following restrictions apply to this command.
- Can be executed only when A/D conversion is stopped (ADCSR.ADST = 0). For state transitions of the A/D conversion operation, see section 2.9, State Transitions.
- Cannot be used while the followings are set (error is returned).

  - Reference voltage × 1/2 is selected as the self-diagnosis conversion voltage.

  - Reference voltage is selected as the self-diagnosis conversion voltage.

```
#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 conversion channels */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* Initialize the S14AD driver */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                        /* Use AN000, AN001, AN006 and AN022 for A/D conversion */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* Temperature sensor not used for A/D conversion */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);    /* Software trigger */

    (void)adcDev->Control(AD_CMD_USE_VREFH0, NULL);  /* Select VREFH0 as the high-potential reference
voltage */

    (void)adcDev->Start();  /* Start A/D conversion */

    while(1);
}
```

Figure 2-43 Example of AD_CMD_USE_VREFH0 Command

2.5.20 Sub-Clock Mode Setting (AD_CMD_SCLK_ENABLE)

The AD_CMD_SCLK_ENABLE command specifies an internally generated clock (sub-clock) to be used at low-frequency clock operation (ADCLK = 32 kHz). Specify NULL in the second argument. Figure 2-44 shows a usage example of the AD_CMD_SCLK_ENABLE command.

The following restrictions apply to this command.

- Can be executed only while A/D conversion is stopped (ADCSR.ADST = 0). For state transitions of the A/D conversion operation, see section 2.9, State Transitions.
- Can be executed only when the operating clock frequency is lower than 32768 Hz (when a low-frequency clock is operating (ADCLK = 32 kHz))/

```
#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 conversion channels */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* Initialize the S14AD driver */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                         /* Use AN000, AN001, AN006 and AN022 for A/D conversion */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* Temperature sensor not used for A/D conversion */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);    /* Software trigger */

    (void)adcDev->Control(AD_CMD_SCLK_ENABLE, NULL);  /* Select the sub-clock for A/D conversion */

    (void)adcDev->Start();  /* Start A/D conversion */

    while(1);
}
```

Figure 2-44 Example of AD_CMD_SCLK_ENABLE Command

2.5.21 Offset Calibration (AD_CMD_CALIBRATION)

The AD_CMD_CALIBRATION performs offset calibration. Specify NULL in the second argument. Figure 2-45 shows a usage example of the AD_CMD_CALIBRATION command.

The following restriction applies to this command.

- Can be executed only in the period from after initialization in the Open function up to before the ScanSet function is executed.

```
#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 conversion channels */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* Initialize the S14AD driver */

    (void)adcDev->Control(AD_CMD_CALIBRATION, NULL);  /* Perform calibration */
    /* Calibration is completed when the function is exited. Software processing to wait for completion
is not required. */

    while(1);
}
```

Figure 2-45 Example of AD_CMD_CALIBRATION Command

2.5.22 A/D Conversion Stop Function (AD_CMD_STOP_TRIG)

The AD_CMD_STOP_TRIG command releases the trigger source and stops A/D conversion (ADCSR.ADST = 0). When the auto-read function is used, it also releases the DMA transfer trigger (Note 1). Use this command to stop A/D conversion that was started with a synchronous trigger or asynchronous trigger (Note 2). Specify NULL in the second argument. Figure 2-46 shows a usage example of the AD_CMD_STOP_TRIG command.

Note 1. For the interrupt function to use an interrupt request that was specified as a DMA transfer trigger by the auto-read function, execute the Close function and then initialize the S14AD driver.

Note 2. This command initializes the A/D conversion start trigger setting. To execute A/D conversion again, re-set the A/D conversion start trigger in the ScanSet function. When using the auto-read function, also make settings for the auto-read function again.

```
#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 conversion channels */
    e_adc_int_method_t adc_int = ADC_INT_ENABLE;    /* Variable to enable interrupts */

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* Initialize the S14AD driver */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                        /* Use AN000, AN001, AN006 and AN022 for A/D conversion */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* Temperature sensor not used for A/D conversion */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_ELC);    /* ELC trigger */

    /* Start A/D conversion with ELC event */

    (void)adcDev->Control(AD_CMD_STOP_TRIG, NULL);  /* Release the ELC trigger source and stop A/D
conversion */

    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_ELC);
                                   /* To resume A/D conversion, execute the ScanSet function again */


    while(1);
}
```

Figure 2-46 Example of AD_CMD_STOP_TRIG Command

2.5.23 Auto-Read Function (Normal) (AD_CMD_AUTO_READ_NORMAL)

The AD_CMD_AUTO_READ_NORMAL command sets up the function for automatically reading (auto-read function (Note)) the A/D conversion result by DMA transfer (DMAC or DTC). Specify a pointer to the st_adc_dma_read_info_t type variable in the second argument. Table 2-33 shows settings of the st_adc_dma_read_info_t type argument. Figure 2-47 shows a usage example of the AD_CMD_AUTO_READ_NORMAL command.

The following restrictions apply to this command.

- The auto-read function and interrupt function are controlled exclusively. An A/D interrupt request used as an interrupt cannot be used in DMA transfer (error is returned). An interrupt request specified as a DMA start trigger can also not be used as an interrupt.
- The setting of the A/D control definition in r_adc_cfg.h needs to be changed to DMAC or DTC. For information on the operation and setting procedure of A/D conversion result auto-read using DMA transfer, see section 2.6.2, Automatically Fetching A/D Conversion Result Using DMA Transfer (Auto-Read Function).
- When the DTC is used, the interrupt source specified as the DMA start trigger must be registered in the Nested Vector Interrupt Controller (hereinafter called the NVIC) by modifying r_system_cfg.h. For information on the NVIC registration for each DMA start trigger, see section 2.6.2, Automatically Fetching A/D Conversion Result Using DMA Transfer (Auto-Read Function).

Note. For details of the auto-read function, see section 2.6.2, Automatically Fetching A/D Conversion Result Using DMA Transfer (Auto-Read Function).

Table 2-33 Settings of st_adc_dma_read_info_t Type Argument

Element	Setting	Description
cb_event	Callback function for auto-read function	Callback function called when DMA transfer has been completed Specifies void in the argument of the callback function.
dma_fact	ADC_READ_ADI	Specifies the A/D scan end interrupt request (ADC140_ADI) as DMA start trigger.
	ADC_READ_GBADI	Specifies the A/D scan end interrupt request of group B (ADC140_GBADI) as DMA start trigger.
	ADC_READ_GCADI	Specifies the A/D scan end interrupt request of group C (ADC140_GCADI) as DMA start trigger.
	ADC_READ_WCMPM(Note 1)	Specifies the condition match interrupt request (ADC140_WCMPM) of the window A/B compare function as DMA start trigger.
	ADC_READ_WCMPUM(Note 1)	Specifies the condition mismatch interrupt request (ADC140_WCMPUM) of the window A/B compare function as DMA start trigger.
src_addr	A/D data register y (ADDRy) (1500KB Group : y = 00 to 06,16,17,20 to 28 256KB Group : y = 00 to 07,16,17,20 to 21)	Transfer source address Specifies one A/D data register that will be read automatically in DMA transfer.
	A/D data duplexing register (ADDBLDR)	
	A/D temperature sensor data register (ADTSDR)	
	A/D VSC_VCC Voltage Voltage Data Register (ADVSCDR) (Note 2)	
dest_addr	Desired RAM	Transfer destination address
transfer_count	[When using DMAC for DMA transfer] 0 to 65535 [When using DTC for DMA transfer] 1 to 65536	Specifies the transfer count. If the transfer count is specified as 0 when using the DMAC, data transfer is in free running mode.
block_size	- (Note 3)	Fixed at 0 (no block)
reg_size	- (Note 3)	Fixed at 16-bit size

Note 1. Cannot be used in this command. If specified, an error will be returned.

Note 2. 256KB Only

Note 3. The setting is fixed in this command regardless of the argument value.

```
#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 transfer destination: RAM */

main()
{
    st_adc_pins_t scanset_pin;  /* ScanSet() A/D conversion channels */
    st_adc_dma_read_info_t arg;    /* 2nd argument of AD_CMD_AUTO_READ_NORMAL */

    (void)adcDev->Open(ADC_REPEAT_SCAN | ADC_14BIT, 0x10, NULL); /* Initialize the S14AD driver */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                        /* Use AN000, AN001, AN006 and AN022 for A/D conversion */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* Temperature sensor not used for A/D conversion */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);

    arg.cb_event      = callback;                /* Callback function */
    arg.dma_fact      = ADC_READ_ADI;        /* DMA transfer trigger */
    arg.src_addr      = (uint32_t)&S14AD->ADDR1; /* Transfer source: A/D data register */
    arg.dest_addr     = (uint32_t)&read_data[0];    /* Transfer destination: RAM */
    arg.transfer_count = 5;                      /* Transfer count: 5 times */
    arg.block_size     = 0;                     /* No block size specified (fixed at 0) */
    arg.reg_size      = 0;                      /* No transfer size specified (fixed at word)  */

    (void)adcDev->Control(AD_CMD_AUTO_READ_NORMAL, &arg);
                                          /* Specify auto-read of the A/D conversion results using DMA
transfer */

    (void)adcDev->Start();  /* Start A/D conversion */

    while(1);
}

/*******************************************************************************************************
* callback function
*******************************************************************************************************/
static void callback(void)
{
    /* Describe the processing to be performed when DMA transfer for the specified count has finished
*/
}
```

Figure 2-47 Example of AD_CMD_AUTO_READ_NORMAL Command

2.5.24 Auto-Read Function (Block) Setting (AD_CMD_AUTO_READ_BLOCK)

The AD_CMD_AUTO_READ_BLOCK command sets up the function for automatically reading (auto-read function (Note)) the A/D conversion result by DMA transfer (DMAC or DTC). Specify a pointer to the st_adc_dma_read_info_t type variable in the second argument. Table 2-34 shows settings of the st_adc_dma_read_info_t type argument. Figure 2-48 shows a usage example of the AD_CMD_AUTO_READ_BLOCK command.

The following restrictions apply to this command.

- The auto-read function and interrupt function are controlled exclusively. An A/D interrupt request used as an interrupt cannot be used in DMA transfer (error is returned). An interrupt request specified as a DMA start trigger can also not be used as an interrupt.
- The setting of the A/D control definition in r_adc_cfg.h needs to be changed to DMAC or DTC. For information on the operation and setting procedure of A/D conversion result auto-read using DMA transfer, see section 2.6.2, Automatically Fetching A/D Conversion Result Using DMA Transfer (Auto-Read Function).
- When the DTC is used, the interrupt source specified as the DMA start trigger must be registered in the Nested Vector Interrupt Controller (hereinafter called the NVIC) by modifying r_system_cfg.h. For information on the NVIC registration for each DMA start trigger, see section 2.6.2, Automatically Fetching A/D Conversion Result Using DMA Transfer (Auto-Read Function).

Note. For details of the auto-read function, see section 2.6.2, Automatically Fetching A/D Conversion Result Using DMA Transfer (Auto-Read Function).

Table 2-34 Settings of st_adc_dma_read_info_t Type Argument

Element	Setting	Description
cb_event	Callback function for auto-read function	Callback function called when DMA transfer has been completed Specifies void in the argument of the callback function.
dma_fact	ADC_READ_ADI	Specifies the A/D scan end interrupt request (ADC140_ADI) as DMA start trigger.
	ADC_READ_GBADI	Specifies the A/D scan end interrupt request of group B (ADC140_GBADI) as DMA start trigger.
	ADC_READ_GCADI	Specifies the A/D scan end interrupt request of group C (ADC140_GCADI) as DMA start trigger.
	ADC_READ_WCMPM (Note 1)	Specifies the condition match interrupt request (ADC140_WCMPM) of the window A/B compare function as DMA start trigger.
	ADC_READ_WCMPUM (Note 1)	Specifies the condition mismatch interrupt request (ADC140_WCMPUM) of the window A/B compare function as DMA start trigger.
src_addr (Note 3)	A/D data register y (ADDRy) (1500KB Group : y = 00 to 06,16,17,20 to 28 256KB Group : sy = 00 to 07,16,17,20 to 21)	Start address in the DMA transfer source register Specifies one A/D data register that will be read automatically in DMA transfer. In combination with the setting of "block_size", multiple A/D data registers can be read as a block.
	A/D data duplexing register (ADDBLDR)	
	A/D temperature sensor data register (ADTSDR)	
	A/D VSC_VCC Voltage Voltage Data Register(ADVSCDR)(Note 2)	
dest_addr	Desired RAM	Transfer destination address
transfer_count	[When using DMAC for DMA transfer] 1 to 65536 [When using DTC for DMA transfer] 1 to 65536	Specifies transfer count.
block_size (Note 3) (Note 4)	[When using DMAC for DMA transfer] 1 to 256 [When using DTC for DMA transfer] 1 to 1024	Transfer block size In combination with the setting of “src_addr”, multiple A/D data registers can be read as a block.
reg_size	- (Note 5)	Fixed at 16-bit size

Note 1. Cannot be used in this command. If specified, an error will be returned.

Note 2. 256KB Only

Note 3. The A/D data register to be read is specified with the register start address and the block size.

Example: Read channels from AN000 to AN006 as one block.

```
arg. src_addr = (uint32_t)&S14AD->ADDR0;    /* Specify the start address of a block */
arg. block_size = 7;                        /* Specify the block size from the start address */
```

Note 4. If a non-existent channel is specified in the A/D data block to be read, an invalid value is transferred to the RAM area corresponding to the channel.

Example: When the channels in the block of AN017 to AN020 are read, an invalid value is transferred to the RAM areas corresponding to the non-existent channels AN018 and AN019.

Note 5. The setting is fixed in this command regardless of the argument value.

```
#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 transfer destination: RAM */

main()
{
    st_adc_pins_t scanset_pin;  /* ScanSet() A/D conversion channels */
    st_adc_dma_read_info_t dma_read;    /* 2nd argument of AD_CMD_AUTO_READ_BLOCK */

    (void)adcDev->Open(ADC_REPEAT_SCAN | ADC_14BIT, 0x10, NULL); /* Initialize the S14AD driver */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN02| ADC_MSEL_AN03;
                                        /* Use AN000, AN001, AN002 and AN003 for A/D conversion */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* Temperature sensor not used for A/D conversion */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);

    dma_read.cb_event     = callback;                /* Callback function */
    dma_read.dma_fact     = ADC_READ_ADI;        /* DMA transfer trigger */
    dma_read.src_addr     = (uint32_t)&S14AD->ADDR0; /* Transfer source: A/D data register */
    dma_read.dest_addr    = (uint32_t)&read_data[0];    /* Transfer destination: RAM */
    dma_read.transfer_count = 5;                     /* Transfer count: 5 times */
    dma_read.block_size    = 4;                    /* ADDR0 to ADDR3 are treated as one block */
    dma_read.reg_size      = 0;                    /* No transfer size specified (fixed at word) */

    (void)adcDev->Control(AD_CMD_AUTO_READ_BLOCK, & dma_read);
                                          /* Specify auto-read of the A/D conversion results using DMA
transfer */

    (void)adcDev->Start();  /* Start A/D conversion */

    while(1);
}

/*****************************************************************************************************
* callback function
*****************************************************************************************************/
static void callback(void)
{
    /* Describe the processing to be performed when DMA transfer for the specified count has finished
*/
}
```

Figure 2-48 Example of AD_CMD_AUTO_READ_BLOCK Command

2.5.25 Auto-Read Function (Compare) Setting (AD_CMD_AUTO_READ_COMPARE)

The AD_CMD_AUTO_READ_COMPARE command sets up the function for automatically reading (auto-read function (Note)) the A/D conversion result by DMA transfer (DMAC or DTC). Specify a pointer to the st_adc_dma_read_info_t type variable in the second argument. Table 2-35 shows settings of the st_adc_dma_read_info_t type argument. Figure 2-49 shows a usage example of the AD_CMD_AUTO_READ_COMPARE command.

The following restrictions apply to this command.

- The auto-read function and interrupt function are controlled exclusively. An A/D interrupt request used as an interrupt cannot be used in DMA transfer (error is returned). An interrupt request specified as a DMA start trigger can also not be used as an interrupt.
- The setting of the A/D control definition in r_adc_cfg.h needs to be changed to DMAC or DTC. For information on the operation and setting procedure of A/D conversion result auto-read using DMA transfer, see section 2.6.2, Automatically Fetching A/D Conversion Result Using DMA Transfer (Auto-Read Function).
- When the DTC is used, the interrupt source specified as the DMA start trigger must be registered in the Nested Vector Interrupt Controller (hereinafter called the NVIC) by modifying r_system_cfg.h. For information on the NVIC registration for each DMA start trigger, see section 2.6.2, Automatically Fetching A/D Conversion Result Using DMA Transfer (Auto-Read Function)
- The compare function (window A/B complex condition) is required to be set. For information on setting the compare function, see sections 2.5.13, Compare Function Window A Setting (AD_CMD_SET_WINDOWA) and 2.5.14, Compare Function Window B Setting (AD_CMD_SET_WINDOWB).

Note. For details of the auto-read function, see section 2.6.2, Automatically Fetching A/D Conversion Result Using DMA Transfer (Auto-Read Function).

Table 2-35 Settings of st_adc_dma_read_info_t Type Argument

Element	Setting	Description
cb_event	Callback function for auto-read function	Callback function called when DMA transfer has been completed Specifies void in the argument of the callback function.
dma_fact	ADC_READ_ADI (Note1)	Specifies the A/D scan end interrupt request (ADC140_ADI) as DMA start trigger.
	ADC_READ_GBADI (Note1)	Specifies the A/D scan end interrupt request of group B (ADC140_GBADI) as a DMA start trigger.
	ADC_READ_GCADI (Note1)	Specifies the A/D scan end interrupt request of group C (ADC140_GCADI) as a DMA start trigger.
	ADC_READ_WCMPM	Specifies the condition match interrupt request (ADC140_WCMPM) of the window A/B compare function as a DMA start trigger.
	ADC_READ_WCMPUM	Specifies the condition mismatch interrupt request (ADC140_WCMPUM) of the window A/B compare function as a DMA start trigger.
src_addr	Desired register	Transfer source address
dest_addr	Desired RAM	Transfer destination address
transfer_count	[When using DMAC for DMA transfer] 0 to 65535 [When using DTC for DMA transfer] 1 to 65536	Specifies the transfer count. If the transfer count is specified as 0 when using the DMAC, data transfer is in free running mode.
block_size	- (Note 2)	Fixed at 0 (no block)
reg_size	ADC_READ_BYTE	8-bit transfer
	ADC_READ_WORD	16-bit transfer
	ADC_READ_LONG	32-bit transfer

Note 1. Cannot be used in this command. If specified, an error will be returned.

Note 2. The setting is fixed in this command regardless of the argument value.

```
#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 transfer destination: RAM */

main()
{
    st_adc_pins_t groupa_pin;  /* A/D conversion channel of group A */
    st_adc_dma_read_info_t dma_read;    /* 2nd argument of AD_CMD_AUTO_READ_COMPARE */
    st_adc_wina_t wina;    /* 2nd argument of AD_CMD_SET_WINDOWA */
    st_adc_winb_t winb;    /* 2nd argument of AD_CMD_SET_WINDOWB */
    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* Initialize the S14AD driver */

    groupa_pin.an_chans = ADC_MSEL_AN02 | ADC_MSEL_AN06;    /* Specify AN02 and AN06 for group A */
    groupa_pin.sensor   = ADC_SENSOR_NOTUSE; /* Temperature sensor not used */
    (void)adcDev->ScanSet(ADC_GROUP_A, groupa_pin, ADC_TRIGER_ADTRG);

    wina.inten             = ADC_DISABLE;      /* Disable the compare A interrupt */
    wina.cmp_mode        = ADC_CMP_WINDOW;    /* Compare windows */
    wina.level1            = 0x02000;           /* Comparison reference value 1 */
    wina.level2            = 0x00100;           /* Comparison reference value 2 */
    wina.chans.an_chans    = ADC_MSEL_AN02;  /* Specify AN02 as the target channel */
    wina.chans.sensor       = ADC_SENSOR_NOTUSE;    /* Temperature sensor not used */
    wina.chan_cond.an_chans = ADC_MSEL_AN02;
                     /* Channel-specific comparison condition: AN02 is specified within a range from
level 1 to level 2 */
    wina.chan_cond.sensor   = ADC_SENSOR_NOTUSE; /* Temperature sensor not used */
    (void)adcDev->Control(AD_CMD_SET_WINDOWA, &wina);    /* Set window A comparison */

    winb.inten      = ADC_DISABLE;                  /* Disable the compare B interrupt */
    winb.winb_cond = ADC_WINB_WINDOW_BETWEEN;  /* Within a range of windows */
    winb.comb      = ADC_COMB_OR;                 /* Complex condition: OR condition */
    winb.level1      = 0x03000;                      /* Comparison reference value 1 */
    winb.level2      = 0x00010;                      /* Comparison reference value 2*/
    winb.channel    = ADC_SSEL_AN06;
    (void)adcDev->Control(AD_CMD_SET_WINDOWB, &winb); /* Set window B comparison */

    dma_read.cb_event     = callback;                      /* Callback function */
    dma_read.dma_fact     = ADC_READ_WCMPUM;         /* DMA transfer trigger */
    dma_read.src_addr     = (uint32_t)&S14AD-> ADWINMON; /* Transfer source: A/D data register */
    dma_read.dest_addr    = (uint32_t)&read_data[0];          /* Transfer destination: RAM */
    dma_read.transfer_count = 5;                             /* Transfer count: 5 times */
    dma_read.block_size    = 0;                            /* No block size specified (fixed at 0) */
    dma_read.reg_size      = ADC_READ_BYTE;             /* Specified as 8 bits */

    (void)adcDev->Control(AD_CMD_AUTO_READ_COMPARE & dma_read);
                                      /* Specify auto-read of the A/D conversion results using DMA
transfer */

    (void)adcDev->Start();  /* Start A/D conversion */

    while(1);
}
/*****************************************************************************************************
* callback function
*****************************************************************************************************/
static void callback(void)
{
    /* Describe the processing to be performed when DMA transfer for the specified count has finished
*/
}
```

Figure 2-49 Example of AD_CMD_AUTO_READ_COMPARE Command

2.5.26 Auto-Read Stop Function (AD_CMD_AUTO_READ_STOP)

The AD_CMD_AUTO_READ_STOP command stops operation of auto-reading of the A/D conversion result by DMA transfer (DMAC or DTC). Specify a pointer to the e_adc_dma_event_t type variable in the second argument. Table 2-36 shows settings of the e_adc_dma_event_t type argument. Figure 2-50 shows a usage example of the AD_CMD_AUTO_READ_STOP command.

The following restriction applies to this command.

- Valid only for interrupt sources that have already been set by the auto-read function.

Table 2-36 Settings of e_adc_dma_event_t Type Argument

Setting	Description
ADC_READ_ADI	Stops auto-read for which an A/D scan end interrupt request (ADC140_ADI) was set as the DMA start trigger.
ADC_READ_GBADI	Stops auto-read for which an A/D scan end interrupt request of group B (ADC140_GBADI) was set as the DMA start trigger.
ADC_READ_GCADI	Stops auto-read for which an A/D scan end interrupt request of group C (ADC140_GCADI) was set as the DMA start trigger.
ADC_READ_WCMPM	Stops auto-read for which a condition match interrupt request (ADC140_WCMPM) of the window A/B compare function was set as the DMA start trigger.
ADC_READ_WCMPUM	Stops auto-read for which a condition mismatch interrupt request (ADC140_WCMPUM) of the window A/B compare function was set as the DMA start trigger.

```
#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 transfer destination: RAM */

main()
{
    st_adc_pins_t scanset_pin;  /* ScanSet() A/D conversion channels */
    st_adc_dma_read_info_t arg;    /* 2nd argument of AD_CMD_AUTO_READ_NORMAL */
    e_adc_dma_event_t stop;    /* 2nd argument of AD_CMD_AUTO_READ_STOP */

    (void)adcDev->Open(ADC_REPEAT_SCAN | ADC_14BIT, 0x10, NULL); /* Initialize the S14AD driver */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                        /* Use AN000, AN001, AN006 and AN022 for A/D conversion */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* Temperature sensor not used for A/D conversion */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);

    arg.cb_event     = callback;                /* Callback function */
    arg.dma_fact     = ADC_READ_ADI;        /* DMA transfer trigger */
    arg.src_addr     = (uint32_t)&S14AD->ADDR1; /* Transfer source: A/D data register */
    arg.dest_addr    = (uint32_t)&read_data[0];    /* Transfer destination: RAM */
    arg.transfer_count = 5;                     /* Transfer count: 5 times */
    arg.block_size    = 0;                    /* No block size specified (fixed at 0) */
    arg.reg_size     = 0;                     /* No transfer size specified (fixed at word) */

    (void)adcDev->Control(AD_CMD_AUTO_READ_NORMAL, &arg);
                                          /* Specify auto-read of the A/D conversion results using DMA
transfer */

    (void)adcDev->Start();  /* Start A/D conversion */

    while(1)
    {
        if (/* Condition to stop auto-read */)
        {
            stop = ADC_READ_ADI;
            (void)adcDev->Control(AD_CMD_AUTO_READ_STOP, & stop);
                                               /* Stop auto-read by ADI request */
        }
    }
}

/*****************************************************************************************************
* callback function
*****************************************************************************************************/
static void callback(void)
{
    st_adc_dma_read_info_t restart = ADC_READ_ADI; /* 2nd argument of AD_CMD_AUTO_READ_RESTART */
    (void)adcDev->Control(AD_CMD_AUTO_READ_RESTART, & restart);
                                               /* Restart auto-read by ADI request */
}
```

Figure 2-50 Example of AD_CMD_AUTO_READ_STOP Command

2.5.27 Auto-Read Restart Function (AD_CMD_AUTO_READ_RESTART)

The AD_CMD_AUTO_READ_RESTART command restarts from the beginning of the operation to automatically read the A/D conversion result using DMA transfer (DMAC or DTC) (Note). Specify a pointer to the e_adc_dma_event_t type variable in the second argument. Table 2-37 shows settings of the e_adc_dma_event_t type argument. Figure 2-51 shows a usage example of the AD_CMD_AUTO_READ_RESTART command.

The following restriction applies to this command.

- Valid only for interrupt sources that have already been set by the auto-read function.

Note. Writes back the transfer source address, transfer destination address, transfer count, and transfer block size to the values that were set by the auto-read command and restarts operation.

Table 2-37 Settings of e_adc_dma_event_t Type Argument

Setting	Description
ADC_READ_ADI	Restarts auto-read for which an A/D scan end interrupt request (ADC140_ADI) was set as the DMA start trigger.
ADC_READ_GBADI	Restarts auto-read for which an A/D scan end interrupt request of group B (ADC140_GBADI) was set as the DMA start trigger.
ADC_READ_GCADI	Restarts auto-read for which an A/D scan end interrupt request of group C (ADC140_GCADI) was set as the DMA start trigger.
ADC_READ_WCMPM	Restarts auto-read for which a condition match interrupt request (ADC140_WCMPM) of the window A/B compare function was set as the DMA start trigger.
ADC_READ_WCMPUM	Restarts auto-read for which a condition mismatch interrupt request (ADC140_WCMPUM) of the window A/B compare function was set as the DMA start trigger.

```
#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 transfer destination: RAM */

main()
{
    st_adc_pins_t scanset_pin;  /* ScanSet() A/D conversion channels */
    st_adc_dma_read_info_t arg;    /* 2nd argument of AD_CMD_AUTO_READ_NORMAL */

    (void)adcDev->Open(ADC_REPEAT_SCAN | ADC_14BIT, 0x10, NULL); /* Initialize the S14AD driver */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                        /* Use AN000, AN001, AN006 and AN022 for A/D conversion */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* Temperature sensor not used for A/D conversion */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);

    arg.cb_event     = callback;                /* Callback function */
    arg.dma_fact     = ADC_READ_ADI;         /* DMA transfer trigger */
    arg.src_addr     = (uint32_t)&S14AD->ADDR1; /* Transfer source: A/D data register */
    arg.dest_addr    = (uint32_t)&read_data[0];    /* Transfer destination: RAM */
    arg.transfer_count = 5;                     /* Transfer count: 5 times */
    arg.block_size    = 0;                    /* No block size specified (fixed at 0)*/
    arg.reg_size     = 0;                     /* No transfer size specified (fixed at word) */

    (void)adcDev->Control(AD_CMD_AUTO_READ_NORMAL, &arg);
                                          /* Specify auto-read of the A/D conversion results using DMA
transfer */

    (void)adcDev->Start();  /* Start A/D conversion */

    while(1);
}

/****************************************************************************************************
* callback function
****************************************************************************************************/
static void callback(void)
{
    e_adc_dma_event_t restart = ADC_READ_ADI; /* 2nd argument of AD_CMD_AUTO_READ_RESTART */
    (void)adcDev->Control(AD_CMD_AUTO_READ_RESTART, & restart);
                                               /* Restart auto-read by ADI request */
}
```

Figure 2-51 Example of AD_CMD_AUTO_READ_RESTART Command

2.6 Method for Fetching A/D Conversion Result

The S14AD driver has two methods for fetching the A/D conversion result. Section 2.6.1 shows the method for fetching the A/D conversion result at a desired timing using the Read function. Section 2.6.2 shows the method for automatically fetching the A/D conversion result using DMA transfer (this method is hereinafter called the auto-read function).

2.6.1 Fetching A/D Conversion Result with Read Function

The S14AD driver can fetch the A/D conversion result at a desired timing using the Read function. There are four types of A/D data registers from which data can be read: A/D data registers y (ADDRy) (Note 1), A/D data duplexing register (ADDBLDR), A/D temperature sensor data register (ADTSDR), and A/D self-diagnosis data register (ADRD). Table 2-38 shows the definitions of A/D data registers specifying the register from which data is to be read. Figure 2-52 shows a usage example of the Read function.

Table 2-38 A/D Data Register Definitions

Command	Description
ADC_SSEL_Ay (Note 1)	A/D data register y (ADDRy) (Note 1)
ADC_SSEL_TEMP	A/D temperature sensor data register (ADTSDR)
ADC_SSEL_DBL	A/D data duplexing register (ADDBLDR)
ADC_SSEL_DIAG	A/D self-diagnosis data register (ADRD)
ADC_SSEL_VSC_VCC(Note 2)	A/D VSC_VCC Voltage Voltage Data Register(ADVSCDR)

Note 1. 1500KB Group : y = 00 to 06,16,17,20 to 28
256KB Group : y = 00 to 07,16,17,20 to 21

Note 2. 256KB Only

```
#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;  /* A/D conversion channel for group A */
    st_adc_status_info_t result_status; /* A/D status storage variable */
    uint16_t ad_data;

    (void)adcDev->Open(ADC_SINGLE_SCAN | ADC_14BIT, 0x10, NULL); /* S14AD driver is initialized. */

    groupa_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN06;    /* Use AN000 and AN006 for A/D conversion */
    groupa_pin.sensor   = ADC_SENSOR_NOTUSE;   /* Temperature sensor not used for A/D conversion. */
    (void)adcDev->ScanSet(ADC_GROUP_A, groupa_pin, ADC_TRIGER_SOFT); /* Software trigger */


    while (1)
    {
        (void)adcDev->Start();  /* Start A/D conversion */

        do
        {
            (void)adcDev->Control(AD_CMD_GET_AD_STATE, & result_status);    /* Acquire the A/D status */
        }
        while (ADC_STATE_RUN == result_status.ad_info);    /* Wait for completion of A/D conversion */

        (void)adcDev->Read(ADC_SSEL_AN00, &ad_data);    /* Store A/D data in ADDR00 in ad_data */

    }

}
```

Figure 2-52 Example of Read Function

2.6.2 Automatically Fetching A/D Conversion Result Using DMA Transfer (Auto-Read Function)

The S14AD driver can automatically fetch the A/D conversion result for the specified number of times using DMA transfer (DMAC or DTC). When using the auto-read function, execute an auto-read command (AD_CMD_AUTO_READ_NORMAL, AD_CMD_AUTO_READ_BLOCK, or AD_CMD_AUTO_READ_COMPARE) in the Control function and make the settings. The DMA transfer start trigger (Note 1), transfer source register, transfer destination RAM, transfer count, callback function (Note 2), and transfer size (Note 1) can be specified for the auto-read function.

When the auto-read function is used, the value of the specified register is automatically stored in an arbitrary RAM address. When auto-reading for the specified number of times has finished, auto-reading stops. Auto-reading can also be stopped by executing the AD_CMD_AUTO_READ_STOP command. When auto-reading is to be performed again with the same settings, execute the AD_CMD_AUTO_READ_RESTART command. When the settings are to be changed, overwrite the settings in an auto-read command.

Note 1. The settings differ according to the auto-read command used. For information on each command setting, see Table 2-39.

Note 2. The callback function specified in the Control function is used as the interrupt to notify the end of DMA transfer. (The callback function that was specified in the Open function is not used.)

Table 2-39 List of Auto-Read Commands

Definition	DMA Start Trigger (Note 1)	Transfer Size	Description
AD_CMD_AUTO_READ_NORMAL	ADC140_ADI ADC140_GBADI ADC140_GCADI	Fixed at 16 bits	Uses DMA normal transfer mode. When the specified DMA start trigger request occurs, the value of the specified A/D data register is transferred to RAM. Only an A/D scan end interrupt request can be specified as a DMA start trigger.
AD_CMD_AUTO_READ_BLOCK	ADC140_ADI ADC140_GBADI ADC140_GCADI	Fixed at 16 bits	Uses DMA block transfer mode. When the specified DMA start trigger request occurs, the value of the specified A/D data register is transferred to RAM. Only an A/D scan end interrupt request can be specified as a DMA start trigger.
AD_CMD_AUTO_READ_COMPARE	ADC140_WCMPM ADC140_WCMPUM	8 bits, 16 bits, and 32 bits (Note 2)	Uses DMA normal transfer mode. When the specified DMA start trigger request occurs, the value of the specified A/D data register is transferred to RAM. Only a window A/B compare match/mismatch request can be specified as a DMA start trigger.
AD_CMD_AUTO_READ_STOP	ADC140_ADI ADC140_GBADI ADC140_GCADI ADC140_WCMPM ADC140_WCMPUM	-	Stops auto-read operation.
AD_CMD_AUTO_READ_RESTART	ADC140_ADI ADC140_GBADI ADC140_GCADI ADC140_WCMPM ADC140_WCMPUM	-	Restarts auto-read operation.

Note 1. Specify the DMA start trigger in the second argument of the Control function. For a list of DMA start trigger definitions, see Table 2-33.

Note 2. Specify the transfer size in the second argument (reg_size) of the Control function. When using the AD_CMD_AUTO_READ_COMPARE command, be sure to specify the transfer size. When using the AD_CMD_AUTO_READ_NORMAL or AD_CMD_AUTO_READ_BLOCK command, the transfer size does not need to be specified because it is fixed at 16 bits.

Table 2-40 Setting of st_adc_dma_read_info_t Type Argument

Element	Setting	Description
cb_event	Callback function for auto-read function (Note 1) (Note 2)	Callback function called when DMA transfer for the specified transfer count has been completed
dma_fact	ADC_READ_ADI (Note 3)	Specifies an A/D scan end interrupt (ADC140_ADI) as DMA start trigger.
	ADC_READ_GBADI (Note 3)	Specifies an A/D scan end interrupt of group B (ADC140_GBADI) as DMA start trigger.
	ADC_READ_GCADI (Note 3)	Specifies an A/D scan end interrupt of group C (ADC140_GCADI) as DMA start trigger.
	ADC_READ_WCMPM (Note 4)	Specifies a condition match interrupt (ADC140_WCMPM) of the window A/B compare function as DMA start trigger.
	ADC_READ_WCMPUM (Note 4)	Specifies a condition mismatch interrupt (ADC140_WCMPUM) of the window A/B compare function as DMA start trigger.
src_addr	A/D data register y (ADDRy) (1500KB Group : y = 00 to 06,16,17,20 to 28 256KB Group : y = 00 to 07,16,17,20 to 21)	Transfer source address Specifies one A/D data register that will be read automatically by DMA transfer.
	A/D data duplexing register (ADDBLDR)	
	A/D temperature sensor data register (ADTSDR)	
	A/D VSC_VCC Voltage Voltage Data Register (ADVSCDR) (Note 5)	
dest_addr	Desired RAM	Transfer destination address Specifies RAM in which information automatically read from the A/D data register by DMA transfer will be stored.
transfer_count	[When using DMAC for DMA transfer] 0 to 65535 [When using DTC for DMA transfer] 1 to 65536	Specifies the transfer count. If the transfer count is specified as 0 when using the DMAC, data transfer is in free running mode.
block_size (Note 6)	[When using DMAC for DMA transfer] 1 to 256 [When using DTC for DMA transfer] 1 to 1024	Transfer block size
reg_size (Note 7)	ADC_READ_BYTE	Transfer size: 8-bit transfer
	ADC_READ_WORD	Transfer size: 16-bit transfer
	ADC_READ_LONG	Transfer size: 32-bit transfer

Note 1. Specify this to generate a callback when DMA transfer has been completed. If the completion of the DMA transfer does not have to be reported, specify NULL in the callback function. (If NULL is specified, an interrupt does not occur.)

Note 2. In a callback by the auto-read feature, the function that was specified when executing the Control function is called. (This is not the function that was specified in the Open function.) A callback function should be specified for each DMA transfer trigger used.

Note 3. Can be used only in the AD_CMD_AUTO_READ_NORMAL and AD_CMD_AUTO_READ_BLOCK commands. It cannot be used in the AD_CMD_AUTO_READ_COMPARE command.

Note 4. Can be used only in the AD_CMD_AUTO_READ_COMPARE command. It cannot be used in the AD_CMD_AUTO_READ_NORMAL and AD_CMD_AUTO_READ_BLOCK commands.

Note 5. 256KB Only

Note 6. Valid only in the AD_CMD_AUTO_READ_BLOCK command.

Note 7. Valid only in the AD_CMD_AUTO_READ_COMPARE command.

When the S14AD driver uses DMA transfer, change the interrupt request control module to DMAC or DTC in r_adc_cfg.h.

The auto-read function and interrupt function are controlled exclusively. An interrupt request specified as the DMA start trigger cannot be used in the interrupt function.

Table 2-41 A/D Interrupt Request Control

Definition (Note)	Initial Value	Description
S14AD_ADI_CONTROL	S14AD_USED_INTERRUPT	ADI interrupt request control (initial value: interrupt)
S14AD_GBADI_CONTROL	S14AD_USED_INTERRUPT	GBADI interrupt request control (initial value: interrupt)
S14AD_GCADI_CONTROL	S14AD_USED_INTERRUPT	GCADI interrupt request control (initial value: interrupt)
S14AD_WCMPM_CONTROL	S14AD_USED_INTERRUPT	Compare match interrupt request control for window A/B (initial value: interrupt)
S14AD_WCMPUM_CONTROL	S14AD_USED_INTERRUPT	Compare mismatch interrupt request control for window A/B (initial value: interrupt)

Note. The CMPAI or CMPBI event can be used only for an interrupt (DMA activation is not possible).

Table 2-42 Definitions of A/D Interrupt Request Control Methods

Definition	Value	Description
S14AD_USED_INTERRUPT	(0)	Uses an A/D scan end interrupt request or window A/B compare function match/mismatch interrupt request in an interrupt or polling.
S14AD_USED_DMAC0	(1<<0)	Activates the DMAC0 by an A/D scan end interrupt request or window A/B compare function match/mismatch interrupt request. If a callback function has been specified, the timing at which DMAC0 transfer for the specified count finished is notified by a DMAC0 transfer end interrupt.
S14AD_USED_DMAC1	(1<<1)	Activates the DMAC1 by an A/D scan end interrupt request or window A/B compare function match/mismatch interrupt request. If a callback function has been specified, the timing at which DMAC1 transfer for the specified count finished is notified by a DMAC1 transfer end interrupt.
S14AD_USED_DMAC2	(1<<2)	Activates the DMAC2 by an A/D scan end interrupt request or window A/B compare function match/mismatch interrupt request. If a callback function has been specified, the timing at which DMAC2 transfer for the specified count finished is notified by a DMAC2 transfer end interrupt.
S14AD_USED_DMAC3	(1<<3)	Activates the DMAC3 by an A/D scan end interrupt request or window A/B compare function match/mismatch interrupt request. If a callback function has been specified, the timing at which DMAC3 transfer for the specified count finished is notified by a DMAC3 transfer end interrupt.
S14AD_USED_DTC	(1<<15)	Activates the DTC by an A/D scan end interrupt request or window A/B compare function match/mismatch interrupt request. If a callback function has been specified, the timing at which DTC transfer for the specified count finished is notified by an interrupt that was specified as a DTC activation source.

If a callback function is specified when executing an auto-read command, the callback will be executed when DMA transfer has been completed. When using a callback, the interrupt source that indicates the completion of the DMA transfer must be registered in the Nested Vector Interrupt Controller (hereinafter called the NVIC) by modifying r_system_cfg.h. Table 2-43 shows the definitions of NVIC registration for DMA start triggers. Figure 2-53 shows an example of registering an interrupt in NVIC.

Table 2-43 Definitions of NVIC Registration for DMA Start Triggers

DMA Transfer Control	DMA Start Trigger	Definition of NVIC Registration	Remark
DTC	Single scan or group A A/D scan end interrupt request (ADC140_ADI)	SYSTEM_CFG_EVENT_NUMBER_ADC140_ADI	
	A/D scan end interrupt request of group B (ADC140_GBADI)	SYSTEM_CFG_EVENT_NUMBER_ADC140_GBADI	
	A/D scan end interrupt request of group C (ADC140_GCADI)	SYSTEM_CFG_EVENT_NUMBER_ADC140_GCADI	
	Condition match interrupt request of the window A/B compare function (ADC140_WCMPM)	SYSTEM_CFG_EVENT_NUMBER_ADC140_WCMPM	
	Condition mismatch interrupt request of the window A/B compare function (ADC140_WCMPUM)	SYSTEM_CFG_EVENT_NUMBER_ADC140_WCMPU M	
DMAC	- (Note 1)	SYSTEM_CFG_EVENT_NUMBER_DMACm_INT	m = 0 to 3 (Note 2)

Note 1. When the DMAC is used, interrupts do not have to be set for each activation source. Set SYSTEM_CFG_EVENT_NUMBER_DMACm_INT when notifying the timing at which DMA transfer ends using a callback.

Note 2. When NULL is set as the argument in the callback function (callback function is not used), this setting is not needed.

```
...
#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 */
...
```

Figure 2-53 Example of Registering Interrupts with NVIC
(When Specifying ADC140_ADI as Start Trigger under DTC Control)

Figure 2-54 and Figure 2-55 respectively show the configuration definition files r_adc_cfg.h and r_system_cfg.h when using the DMAC by the auto-read function. Figure 2-56 and Figure 2-57 respectively show them when using the DTC.

```
...
#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
...
```

Figure 2-54 r_adc_cfg.h Setting Example (When DMAC0 is Used for ADI Interrupt Request Control)

```
...
#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 */
...
```

Figure 2-55 r_system_cfg.h Setting Example (When Using DMAC0 Transfer End Interrupt)

```
...
#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
...
```

Figure 2-56 r_adc_cfg.h Setting Example (When DTC is Used for ADI Interrupt Request Control)

```
...
#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 */
...
```

Figure 2-57 r_system_cfg.h Setting Example
(When ADI Interrupt is Used for DTC Transfer End Notification)

Figure 2-58 shows the sample code for using the auto-read function.

```
#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 transfer destination: RAM */

main()
{
    st_adc_pins_t scanset_pin;  /* ScanSet() A/D conversion channels */
    st_adc_dma_read_info_t arg;    /* 2nd argument of AD_CMD_AUTO_READ_NORMAL */

    (void)adcDev->Open(ADC_REPEAT_SCAN | ADC_14BIT, 0x10, NULL); /* Initialize the S14AD driver */

    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN06 | ADC_MSEL_AN22;
                                        /* Use AN000, AN001, AN006 and AN022 for A/D conversion */
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;   /* Temperature sensor not used for A/D conversion */
    (void)adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);

    arg.cb_event     = callback;                 /* Callback function */
    arg.dma_fact     = ADC_READ_ADI;         /* DMA transfer trigger */
    arg.src_addr     = (uint32_t)&S14AD->ADDR1; /* Transfer source: A/D data register */
    arg.dest_addr    = (uint32_t)&read_data[0];    /* Transfer destination: RAM */
    arg.transfer_count = 5;                      /* Transfer count: 5 times */
    arg.block_size    = 0;                     /* No block size specified (fixed at 0) */
    arg.reg_size      = 0;                     /* No transfer size specified (fixed at 16 bits) */

    (void)adcDev->Control(AD_CMD_AUTO_READ_NORMAL, &arg);
                                          /* Specify auto-read of the A/D conversion result using DMA
transfer */

    (void)adcDev->Start();  /* Start A/D conversion */

    while(1);
}

/******************************************************************************************************
* callback function
******************************************************************************************************/
static void callback(void)
{
    /* Describe the processing to be performed when DMA transfer for the specified count has finished
*/
    /* Processing example for executing DMA transfer again */
    e_adc_dma_event_t restart = ADC_READ_ADI; /* 2nd argument of AD_CMD_AUTO_READ_RESTART */
    (void)adcDev->Control(AD_CMD_AUTO_READ_RESTART, & restart);
                                                /* Restart auto-read by ADI request */
}
```

Figure 2-58 Example for Using Auto-Read Function (Common to DMAC/DTC)

2.7 Macro and Type Definitions

For the S14AD driver, the macro and types that can be referenced by the user are defined in the r_adc_api.h file.

2.7.1 S14AD Initial Setting Code Definitions

The definitions of the S14AD initial setting codes are definitions for A/D scan mode setting and A/D initial setting that are used in the first argument of the Open function. The S14AD initial setting definitions consist of definitions of scan mode setting, A/D conversion accuracy, and A/D data format.

Figure 2-59 shows a format of a S14AD initial setting code definition. Table 2-44 to Table 2-46 show various setting definitions.

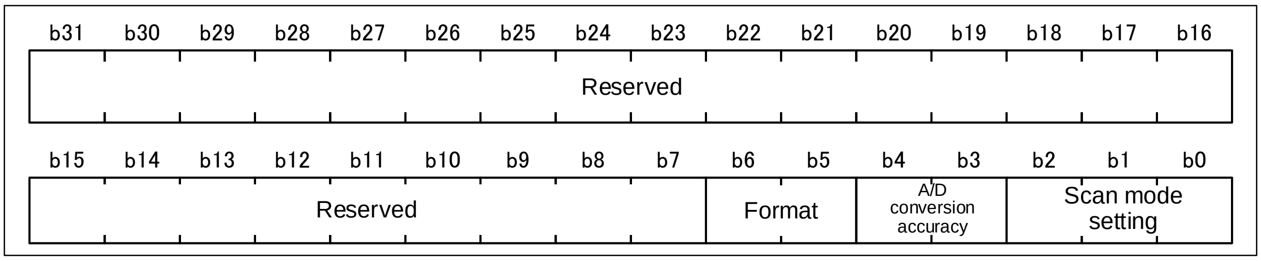

b31 b30 b29 b28 b27 b26 b25 b24 b23 b22 b21 b20 b19 b18 b17 b16
Reserved
b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0
Reserved
Format
A/D conversion accuracy
Scan mode setting

Figure 2-59 Structure of S14AD Initial Setting Code Definitions

Table 2-44 List of Scan Mode Definitions

Command	Value	Description
ADC_SINGLE_SCAN (default)	0U << ADC_MODE_POS	Single-scan mode
ADC_GROUP_SCAN	1U << ADC_MODE_POS	Group-scan mode
ADC_REPEAT_SCAN	2U << ADC_MODE_POS	Continuous-scan mode

Table 2-45 List of A/D Conversion Accuracy Definitions

Command	Value	Description
ADC_14BIT (default)	0U << ADC_RESOLUTION_POS	14-bit accuracy
ADC_12BIT	1U << ADC_RESOLUTION_POS	12-bit accuracy

Table 2-46 List of A/D Data Register Format Definitions

Command	Value	Description
ADC_RIGHT (default)	0U << ADC_FORMAT_POS	Flush right
ADC_LEFT	1U << ADC_FORMAT_POS	Flush left

2.7.2 A/D Conversion Start Trigger Definition

The definitions of A/D conversion start triggers are for specifying an A/D conversion start trigger to be used in the second argument of the ScanSet function.

Table 2-47 Definitions of A/D Conversion Start Conditions

Command	Value	Description
ADC_TRIGER_SOFT	(00)	Software trigger
ADC_TRIGER_TMR	(01)	Compare match between the TCORA register and the TCNT counter
ADC_TRIGER_ELC	(02)	ELC_S14AD
ADC_TRIGER_ADTRG	(03)	ADTRG0 input
ADC_TRIGER_LOW_PRIORITY_CONT_SCAN	(04)	Trigger source not selected

2.7.3 Multiple A/D Conversion Channel Selection Definition

The definitions of multiple A/D conversion channel selection are for specifying multiple A/D conversion channels by combining them.

Table 2-48 A/D Conversion Channel Selection Definitions

Definition	Value	Description
ADC_MSEL_AN00	(1 << 0)	AN000 channel selected
ADC_MSEL_AN01	(1 << 1)	AN001 channel selected
ADC_MSEL_AN02	(1 << 2)	AN002 channel selected
ADC_MSEL_AN03	(1 << 3)	AN003 channel selected
ADC_MSEL_AN04	(1 << 4)	AN004 channel selected
ADC_MSEL_AN05	(1 << 5)	AN005 channel selected
ADC_MSEL_AN06	(1 << 6)	AN006 channel selected
ADC_MSEL_AN07 (Note 1)	(1 << 7)	AN007 channel selected
ADC_MSEL_AN16	(1 << 16)	AN016 channel selected
ADC_MSEL_AN17	(1 << 17)	AN017 channel selected
ADC_MSEL_AN20	(1 << 20)	AN020 channel selected
ADC_MSEL_AN21	(1 << 21)	AN021 channel selected
ADC_MSEL_AN22 (Note 2)	(1 << 22)	AN022 channel selected
ADC_MSEL_AN23 (Note 2)	(1 << 23)	AN023 channel selected
ADC_MSEL_AN24 (Note 2)	(1 << 24)	AN024 channel selected
ADC_MSEL_AN25 (Note 2)	(1 << 25)	AN025 channel selected
ADC_MSEL_AN26 (Note 2)	(1 << 26)	AN026 channel selected
ADC_MSEL_AN27 (Note 2)	(1 << 27)	AN027 channel selected
ADC_MSEL_AN28 (Note 2)	(1 << 28)	AN028 channel selected
ADC_MSEL_VSC_VCC (Note 1)	(1 << 31)	VSC_VCC Voltage selected

Note 1. 256KB Only

Note 2. 1500KB Only

2.7.4 Temperature Sensor Output Usage Definition

The definitions of temperature sensor output usage are used when specifying temperature sensor output in A/D conversion.

Table 2-49 Temperature Sensor Output Usage Definitions

Command	Value	Description
ADC_SENSOR_NOTUSE	(00)	Temperature sensor output not used
ADC_MSEL_TEMP	(1 << 0)	Temperature sensor output used

2.7.5 Single A/D Conversion Channel Selection Definition

The definitions of single A/D conversion channel selection are for specifying a single A/D conversion channel.

Table 2-50 Single A/D Conversion Channel Selection Definitions

Definition	Value	Description
ADC_SSEL_AN00	(00)	AN000 channel selected
ADC_SSEL_AN01	(01)	AN001 channel selected
ADC_SSEL_AN02	(02)	AN002 channel selected
ADC_SSEL_AN03	(03)	AN003 channel selected
ADC_SSEL_AN04	(04)	AN004 channel selected
ADC_SSEL_AN05	(05)	AN005 channel selected
ADC_SSEL_AN06	(06)	AN006 channel selected
ADC_SSEL_AN07 (Note 1)	(07)	AN007 channel selected
ADC_SSEL_AN16	(16)	AN016 channel selected
ADC_SSEL_AN17	(17)	AN017 channel selected
ADC_SSEL_AN20	(20)	AN020 channel selected
ADC_SSEL_AN21	(21)	AN021 channel selected
ADC_SSEL_AN22 (Note 2)	(22)	AN022 channel selected
ADC_SSEL_AN23 (Note 2)	(23)	AN023 channel selected
ADC_SSEL_AN24 (Note 2)	(24)	AN024 channel selected
ADC_SSEL_AN25 (Note 2)	(25)	AN025 channel selected
ADC_SSEL_AN26 (Note 2)	(26)	AN026 channel selected
ADC_SSEL_AN27 (Note 2)	(27)	AN027 channel selected
ADC_SSEL_AN28 (Note 2)	(28)	AN028 channel selected
ADC_SSEL_TEMP	(32)	Temperature sensor output selected
ADC_SSEL_DBL	(50)	Double trigger selected
ADC_SSEL_DIAG	(51)	Self-diagnosis selected
ADC_SSEL_VSC_VCC (Note 1)	(32)	VSC_VCC Voltage selected

Note 1. 256KB Only

Note 2. 1500KB Only

2.7.6 A/D Conversion Group Definition

The definitions of A/D conversion groups are for specifying the group for which settings are made.

Table 2-51 A/D Group Definitions

Command	Value	Description
ADC_GROUP_A	(00)	Group A
ADC_GROUP_B	(01)	Group B
ADC_GROUP_C	(02)	Group C

2.7.7 Control Command Definition of Control Function

The definitions of control commands of the Control function are those for setting the features that are used in the first argument of the Control function.

Table 2-52 List of Control Commands of Control Function

Definition	Description
AD_CMD_SET_ADD_MODE	Command for setting addition mode
AD_CMD_SET_DBLTRG	Commend for enabling double-trigger mode
AD_CMD_SET_DIAG	Command for enabling self-diagnosis mode
AD_CMD_SET_ADI_INT	Command for setting an ADI_INT interrupt
AD_CMD_SET_GROUPB_INT	Command for setting a group B interrupt
AD_CMD_SET_GROUPC_INT	Command for setting a group C interrupt
AD_CMD_SET_WCMPM_INT	Command for setting a WCMPM interrupt
AD_CMD_SET_WCMPUM_INT	Command for setting a WCMPUM interrupt
AD_CMD_SET_AUTO_CLEAR	Command for enabling or disabling auto clear
AD_CMD_SET_SAMPLING_AN000	Command for changing the sampling state of AN000 or self-diagnosis
AD_CMD_SET_SAMPLING_AN001	Command for changing the sampling state of AN001
AD_CMD_SET_SAMPLING_AN002	Command for changing the sampling state of AN002
AD_CMD_SET_SAMPLING_AN003	Command for changing the sampling state of AN003
AD_CMD_SET_SAMPLING_AN004	Command for changing the sampling state of AN004
AD_CMD_SET_SAMPLING_AN005	Command for changing the sampling state of AN005
AD_CMD_SET_SAMPLING_AN006	Command for changing the sampling state of AN006
AD_CMD_SET_SAMPLING_AN007(Note1)	Command for changing the sampling state of AN007
AD_CMD_SET_SAMPLING_AN016_AN017_AN020_TO_AN028(Note 2)	Command for changing the sampling state of AN16, 017, and AN020 to AN028
AD_CMD_SET_SAMPLING_AN016_AN017_AN020_AN021_VSC_VCC (Note1)	Sets the sampling time of AN016, AN017, AN020, AN021 and VSC_VCC voltage
AD_CMD_SET_SAMPLING_TEMP	Command for changing the sampling state of temperature sensor
AD_CMD_SET_ADNDIS	Command for setting the disconnection detection assistance
AD_CMD_SET_GROUP_PRIORITY	Command for setting the group priority
AD_CMD_SET_WINDOWA	Command for setting the comparison condition for window A
AD_CMD_SET_WINDOWB	Command for setting the comparison condition for window B
AD_CMD_SET_ELC	Command for setting the generating condition of AD140_ELC
AD_CMD_GET_CMP_RESULT	Command for acquiring the window comparison result
AD_CMD_GET_AD_STATE	Command for acquiring the A/D converter status
AD_CMD_USE_VREFL0	Command for selecting the low-potential reference voltage VREFL0
AD_CMD_USE_VREFH0	Command for selecting the high-potential reference voltage VREFH0
AD_CMD_SCLK_ENABLE	Command for selecting a sub-clock
AD_CMD_CALIBRATION	Command for performing offset calibration
AD_CMD_STOP_TRIG	Command for stopping A/D conversion and clearing the source of the A/D conversion start trigger
AD_CMD_AUTO_READ_NORMAL	Command for setting auto-read by DMA transfer (normal)
AD_CMD_AUTO_READ_BLOCK	Command for setting auto-read by DMA transfer (block)
AD_CMD_AUTO_READ_COMPARE	Command for setting auto-read by DMA transfer (normal) with WCMPM or WCMPUM as the start trigger
AD_CMD_AUTO_READ_STOP	Command for stopping DMA transfer
AD_CMD_AUTO_READ_RESTART	Command for restarting DMA transfer

Note 1. 256KB Only

Note 2. 1500KB Only

2.7.8 Addition/Average Mode Setting Definition

The definitions for setting addition mode or average mode are to be used in the second argument when executing the AD_CMD_SET_ADD_MODE command.

Table 2-53 Addition/Average Mode Setting Definitions

Definition	Value	Description
ADC_ADD_OFF	(0)	A/D-converted value addition/average mode disabled
ADC_ADD_2_SAMPLES	(1)	Addition mode/2-time conversion (addition of one time)
ADC_ADD_4_SAMPLES	(3)	Addition mode/4-time conversion (addition of 3 times)
ADC_ADD_16_SAMPLES	(5)	Addition mode/16-time conversion (addition of 15 times)
ADC_ADD_AVG_2_SAMPLES	(0x81)	Average mode/2-time conversion (addition of one time)
ADC_ADD_AVG_4_SAMPLES	(0x83)	Average mode/4-time conversion (addition of 3 times)
ADC_ADD_AVG_16_SAMPLES	(0x85)	Average mode/16-time conversion (addition of 15 times)

2.7.9 Self-Diagnosis Mode Setting Definition

The definitions for setting self-diagnosis mode are to be used in the second argument when executing the AD_CMD_SET_DIAG command.

Table 2-54 Self-Diagnosis Setting Definitions

Definition	Value	Description
AD_DIAG_DISABLE	(00)	Self-diagnosis disabled
AD_DIAG_0V	(01)	0 V voltage
AD_DIAG_HARF	(02)	Reference power supply (VREFH0) × 1/2 voltage
AD_DIAG_BASE	(03)	Reference power supply (VREFH0) voltage
AD_DIAG_ROTATE	(04)	Self-diagnosis voltage rotation mode

2.7.10 Interrupt Function Setting Definition

The definitions for setting interrupt function are to be used when executing interrupt function setting command (Note).

Note. 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

Table 2-55 Interrupt Function Setting Definitions

Definition	Value	Description
ADC_INT_DISABLE	(00)	Interrupt disabled
ADC_INT_POLLING	(01)	Polling
ADC_INT_ENABLE	(02)	Interrupt enabled

2.7.11 S14AD Enable/Disable Definition

The S14AD enable/disable definitions are for setting function to be enabled or disabled.

Table 2-56 Settings of 2nd Argument

Definition	Value	Description
ADC_ENABLE	(01)	A/D data register auto-clear enabled
ADC_DISABLE	(00)	A/D data register auto-clear disabled

2.7.12 Disconnection Detection Assist Setting Definition

The definitions for setting disconnection detection assistance are to be used in the second argument when executing the AD_CMD_SET_ADNDIS command. When a precharge or discharge is set, it should be specified in combination with a precharge or discharge period.

Example:

Precharge period: 5 cycles

uint8_t arg = ADC_DDA_DISCHARGE | 5;

adcDrv -> Control(ADC_DDA_PRECHARGE, &arg);

Table 2-57 Disconnection Detection Assist Setting Definitions

Definition	Value	Description
ADC_DDA_OFF	(0x00)	Disconnection detection assist function disabled
ADC_DDA_PRECHARGE	(0x10)	Precharge
ADC_DDA_DISCHARGE	(0x00)	Discharge

2.7.13 Group Priority Operation Setting Definition

The definitions for setting group priority operations are used in the second argument when executing the AD_CMD_SET_GROUP_PRIORITY command.

Table 2-58 Group Priority Operation Setting Definitions

Definition	Value	Description
ADC_GSP_PRIORYTY_OFF	(0x0000)	Group priority operation disabled
ADC_GSP_WAIT_TRG	(0x0001)	Scanning for a low-priority group not restarted
ADC_GSP_SCAN_BIGIN	(0x0003)	Scanning for a low-priority group restarted
ADC_GSP_SCAN_RESTART	(0x4003)	Rescanning started from the channel for which A/D conversion is not completed
ADC_GSP_WAIT_TRG_CONT_SCAN	(0x8001)	Continuous single scanning enabled and low-priority group restart disabled
ADC_GSP_SCAN_BIGIN_CONT_SCAN	(0x8003)	Continuous single scanning enabled and low-priority group restart enabled
ADC_GSP_SCAN_RESTART_CONT_SCAN	(0xC003)	Continuous single scanning enabled and rescanning is started from the channel for which A/D conversion has not completed

2.7.14 Compare Function Window A Setting Definition

The definitions for setting the compare function window A are used in the second argument when executing the AD_CMD_SET_WINDOWA command.

Table 2-59 List of Compare Function Window A Setting Definitions

Definition	Value	Description
ADC_CMP_OFF	(00)	Not compared
ADC_CMP_LEVEL	(01)	Level comparison
ADC_CMP_WINDOW	(02)	Window comparison

2.7.15 Compare Function Window B Setting Definition

The definitions for setting the compare function window A are used in the second argument when executing the AD_CMD_SET_WINDOWB command.

Table 2-60 Compare Function Window B Setting Definitions

Setting	Value	Description
ADC_WINB_LEVEL_BELOW	(00)	Level 1 or lower
ADC_WINB_LEVEL_ABOVE	(01)	Level 1 or higher
ADC_WINB_WINDOW_OUTSIDE	(02)	Outside a range from level 1 to level 2
ADC_WINB_WINDOW_BETWEEN	(03)	Within a range from level 1 to level 2

2.7.16 Definition for Setting Compare Function Window A/B Complex conditions

The definitions for setting the complex conditions for window A/B compare function are used in the second argument when executing the AD_CMD_SET_WINDOWB command.

Table 2-61 Definitions to Set Compare Function Window A/B Complex conditions

Setting	Value	Description
ADC_COMB_OR	(00)	Outputs ADC140_WCMPM when window A OR window B comparison conditions are met; otherwise, outputs ADC140_WCMPUM
ADC_COMB_EXOR	(01)	Outputs ADC140_WCMPM when window A EXOR window B comparison conditions are met; otherwise, outputs ADC140_WCMPUM
ADC_COMB_AND	(02)	Outputs ADC140_WCMPM when window A AND window B comparison conditions are met; otherwise, outputs ADC140_WCMPUM
ADC_COMB_NON_EVENT	(03)	Outputs an event only in window A (Note)
ADC_COMB_OFF	(0xFF)	Window A/B complex conditions are invalid

Note. When ADC_COMB_NON_EVENT is specified, the window B settings become as shown below regardless of the values set in the arguments.

- Complex condition of window A and window B: OR condition (ADCMPCR.CMPAB = 00b)
- Compared channel of window B: Not selected (ADCMPBNSR.CMPCHB = 0x3F)
- Comparison condition of window B: Always mismatch (0 < results < 0)
  (ADCMPCR.WCMPE = 1, ADWINLLB[15:0] = ADWINULB[15:0] = 0000h,
  ADCMPBNSR.CMPLB = 1)

2.7.17 S14AD Status Code Definition

The S14AD status codes indicate the status of the S14AD driver that is acquired when executing the AD_CMD_GET_AD_STATE command. Table 2-62 shows the status definitions of A/D conversion operation. Table 2-63 shows the status definitions for each interrupt source.

Table 2-62 A/D Conversion Status Definitions

Definition	Value	Description
ADC_STATE_STOP	(00)	A/D conversion halted (ADCSR.ADST = 0)
ADC_STATE_RUN	(01)	A/D conversion in progress (ADCSR.ADST = 1)

Table 2-63 Status Definitions for Each Interrupt Source

Definition	Value	Description
ADC_CONV_NOT_POLLING	(00)	Polling not set
ADC_CONV_INCOMPLETE	(01)	A/D conversion not completed
ADC_CONV_COMPLETE	(02)	A/D conversion completed
ADC_CONV_GET_FAILED	(03)	Failed in acquiring the status of A/D conversion operation
ADC_CONV_WCMP_DETECT	(04)	Window compare event (ADC140_WCMPM or ADC140_WCMPUM) detected
ADC_CONV_WCMP_NOT_DETECT	(05)	Window compare event (ADC140_WCMPM or ADC140_WCMPUM) not detected

2.7.18 Auto-Read Function Setting Definition

The definitions for setting the auto-read function are used in the second argument when executing the auto-read function setting command. Table 2-64 shows a list of definitions to specify a DMA transfer trigger. Table 2-65 shows a list of definitions to specify the DMA transfer size.

Table 2-64 DMA Transfer Trigger Setting Definitions

Definition	Value	Description
ADC_READ_ADI	(00)	Specifies the A/D scan end interrupt request (ADC140_ADI) as DMA start trigger
ADC_READ_GBADI	(01)	Specifies the A/D scan end interrupt request of group B (ADC140_GBADI) as DMA start trigger
ADC_READ_GCADI	(02)	Specifies the A/D scan end interrupt request of group C (ADC140_GCADI) as DMA start trigger
ADC_READ_WCMPM	(03)	Specifies the condition match interrupt request (ADC140_WCMPM) of the window A/B compare function as DMA start trigger
ADC_READ_WCMPUM	(04)	Specifies the condition mismatch interrupt request (ADC140_WCMPUM) of the window A/B compare function as DMA start trigger

Table 2-65 Definitions to Specify DMA Transfer Size

Definition	Value	Description
ADC_READ_BYTE	(00)	8-bit transfer
ADC_READ_WORD	(01)	16-bit transfer
ADC_READ_LONG	(02)	32-bit transfer

2.7.19 Definition to Specify Scan End Event (ADC140_ELC) Generating Condition

The definitions to specify generating conditions of the scan end event (ADC140_ELC) are used in the second argument when executing the AD_CMD_SET_ELC command.

Table 2-66 Definitions to Specify Scan End Event (ADC140_ELC)

Definition	Value	Description
ADC_ELC_GROUPA	(0x00)	Generates an event upon completion of scanning other than group B and group C scans.
ADC_ELC_GROUPB	(0x01)	Generates an event on completion of scanning of group B.
ADC_ELC_GROUPC	(0x04)	Generates an event on completion of scanning of group C.
ADC_ELC_ALL	(0x02)	Generates an event on completion of all scans.

2.7.20 S14AD Event Code Definition

The following shows definitions of events that will be notified by the callback function, which was specified in the Open function.

Table 2-67 List of S14AD Event Codes

Definition	Value	Description
ADC_EVT_SCAN_COMPLETE	(00)	Single scanning or A/D scan of group A completed
ADC_EVT_SCAN_COMPLETE_GROUPB	(01)	A/D scan of group B completed
ADC_EVT_SCAN_COMPLETE_GROUPC	(02)	A/D scan of group C completed
ADC_EVT_CONDITION_MET	(03)	Comparison condition for window A are met
ADC_EVT_CONDITION_METB	(04)	Comparison condition for window B are met
ADC_EVT_WINDOW_CMP_MATCH	(05)	Condition for window A/B compare function are met
ADC_EVT_WINDOW_CMP_UNMATCH	(06)	Condition for window A/B compare function are not met

2.7.21 S14AD Error Code Definition

The following shows definitions of the S14AD error codes.

Table 2-68 List of S14AD Error Codes

Definition	Value	Description
ADC_OK	(00)	Normal completion
ADC_ERROR	(01)	S14AD driver setting invalid
ADC_ERROR_BUSY	(02)	A/D conversion in progress
ADC_ERROR_PARAMETER	(03)	Invalid argument
ADC_ERROR_MODE	(04)	Invalid mode setting
ADC_ERROR_LOCKED	(05)	S14AD module locked
ADC_ERROR_SYSTEM_SETTING	(06)	Invalid system setting

2.7.22 S14AD Callback Function Definition

The following is the S14AD callback function definition.

Table 2-69 adc_cd_event_t Type Definition

Type Definition	Description
void (*adc_cd_event_t) (uint32_t event)	Definition of S14AD callback function type

2.8 Structure Definitions

For the S14AD driver, the structures that can be referenced by the user are defined in the r_adc_api.h file.

2.8.1 st_adc_pins_t Structure

This structure is used to specify the analog input channel to be used by the S14AD driver and enable or disable the temperature sensor output

Table 2-70 st_adc_pins_t Structure

Element Name	Type	Description
an_chans	uint32_t	Analog input channel
sensor	e_adc_msel_sensor_t	Enables or disables temperature sensor

2.8.2 st_adc_add_mode_t Structure

This structure is used as the second argument type when setting the A/D-converted value addition/average mode in the AD_CMD_SET_ADD_MODE command.

Table 2-71 st_adc_add_mode_t Structure

Element Name	Type	Description
add_mode	e_adc_add_t	Sets the A/D-converted value addition/average mode
chans	st_adc_pins_t	Selects A/D conversion channels subject to A/D-converted value addition/averaging

2.8.3 st_adc_wina_t Structure

This structure is used as the second argument type when setting the compare function window A in the AD_CMD_SET_WINDOWA command.

Table 2-72 st_adc_wina_t Structure

Element Name	Type	Description
inten	uint8_t	Enables or disables the ADC140_CMPAI interrupt
cmp_mode	e_adc_cmp_mode_t	Specifies compare mode
level1	uint16_t	Specifies compare level
level2	uint16_t	Specifies compare level
chans	st_adc_pins_t	Specifies a combination of channels for comparison
chan_cond	st_adc_pins_t	Specifies compare mode for each channel

2.8.4 st_adc_winb_t Structure

This structure is used as the second argument type when setting the compare function window B in the AD_CMD_SET_WINDOWB command.

Table 2-73 st_adc_winb_t Structure

Element Name	Type	Description
inten	uint8_t	Enables or disables the ADC140_CMPBI interrupt
winb_cond	e_adc_winb_cond_t	Specifies compare mode
comb	e_adc_comb_t	Specifies window A/B complex conditions
level1	uint16_t	Specifies compare level
level2	uint16_t	Specifies compare level
channel	e_adc_ssel_ch_t	Specifies channels for comparison

2.8.5 st_adc_wina_result_t Structure

This structure is used to acquire the comparison result of A/D compare function window A in the AD_CMD_GET_CMP_RESULT command.

Table 2-74 Information Stored in st_adc_cmp_result_t Type Argument

Element	Type	Description
an_chans	uint32_t	Values of A/D compare function window A channel status register 0 (ADCMPSR0) and A/D compare function window A channel status register 1 (ADCMPSR1) (bit 6 corresponds to AN006 and bit 0 corresponds to AN000) [Comparison result for each pin] 0: Comparison condition is not met 1: Comparison condition is met
sensor	uint8_t	Values of A/D compare function window B channel status register (ADCMPBSR) [Comparison result of selected channel in window B] 0: Comparison condition is not met 1: Comparison condition is met

2.8.6 st_adc_cmp_result_t Structure

This structure is used to acquire the comparison results of the A/D compare function in the AD_CMD_GET_CMP_RESULT command.

Table 2-75 Information Stored in st_adc_cmp_result_t Type Argument

Element	Type	Description
wina_result	st_adc_wina_result_t	Comparison result of A/D compare function window A
winb_result	uint8_t	Value of A/D compare function window B status register (ADCMPBSR) [Comparison result of channel selected for window B] 0: Comparison conditions are not met. 1: Comparison conditions are met.
comb_result	uint8_t	A/D compare function window A/B status monitor register (ADWINMON) [Combined results (bit 0)] 0: Window A/B complex conditions are not met. 1: Window A/B complex conditions are met. [Comparison result monitor A (bit 4)] 0: Window A comparison conditions are not met. 1: Window A comparison conditions are met. [Comparison result monitor B (bit 5)] 0: Window B comparison conditions are not met. 1: Window B comparison conditions are met.

2.8.7 st_adc_status_info_t Structure

This structure is used to acquire the status of the A/D converter in the AD_CMD_GET_AD_STATE command.

Table 2-76 Information Stored in st_adc_status_info_t Type Argument

Element	Type	Description
ad_info	e_adc_state_t	Status information on the A/D scan operation ADC_STATE_STOP: A/D conversion halted (ADCSR.ADST = 0) ADC_STATE_RUN: A/D conversion in progress (ADCSR.ADST = 1)
groupa_info	e_adc_conv_state_t	Status information of group A operation ADC_CONV_NOT_POLLING: Polling not set ADC_CONV_INCOMPLETE: A/D scan of group A not completed ADC_CONV_COMPLETE: A/D scan of group A completed ADC_CONV_GET_FAILED: Failed in acquiring the group A status
groupb_info	e_adc_conv_state_t	Status information of group B operation ADC_CONV_NOT_POLLING: Polling not set ADC_CONV_INCOMPLETE: A/D scan of group B not completed ADC_CONV_COMPLETE: A/D scan of group B completed ADC_CONV_GET_FAILED: Failed in acquiring the group B status
groupc_info	e_adc_conv_state_t	Status information of group C operation ADC_CONV_NOT_POLLING: Polling not set ADC_CONV_INCOMPLETE: A/D scan of group C not completed ADC_CONV_COMPLETE: A/D scan of group C completed ADC_CONV_GET_FAILED: Failed in acquiring the group B status
wcmpm_info	e_adc_conv_state_t	Information on window A/B compare match event occurrence ADC_CONV_WCMP_DETECT: Window A/B compare match event occurred ADC_CONV_WCMP_NOT_DETECT: Window A/B compare match event not occurred
wcmpum_info	e_adc_conv_state_t	Information on window A/B compare mismatch event occurrence ADC_CONV_WCMP_DETECT: Window A/B compare mismatch event occurred ADC_CONV_WCMP_NOT_DETECT: Window A/B compare mismatch event not occurred

2.8.8 st_adc_dma_read_info_t Structure

This structure is used as the second argument type when setting the function for automatically reading (auto-read function) the A/D conversion result by DMA transfer (DMAC or DTC) in the AD_CMD_AUTO_READ_NORMAL command.

Table 2-77 Settings of st_adc_dma_read_info_t Type Argument

Element	Type	Description
cb_event	system_int_cb_t	Callback function called when DMA transfer has been completed
dma_fact	e_adc_dma_event_t	Specifies a DMA start trigger
src_addr	uint32_t	Transfer source address
dest_addr	uint32_t	Transfer destination address
transfer_count	uint32_t	Specifies transfer count
block_size	uint32_t	Specifies block size
reg_size	uint32_t	Specifies transfer size

2.9 State Transitions

The state transition diagram of the S14AD driver is shown in Figure 2-60, and state-specific events and actions are shown in Table 2-78.

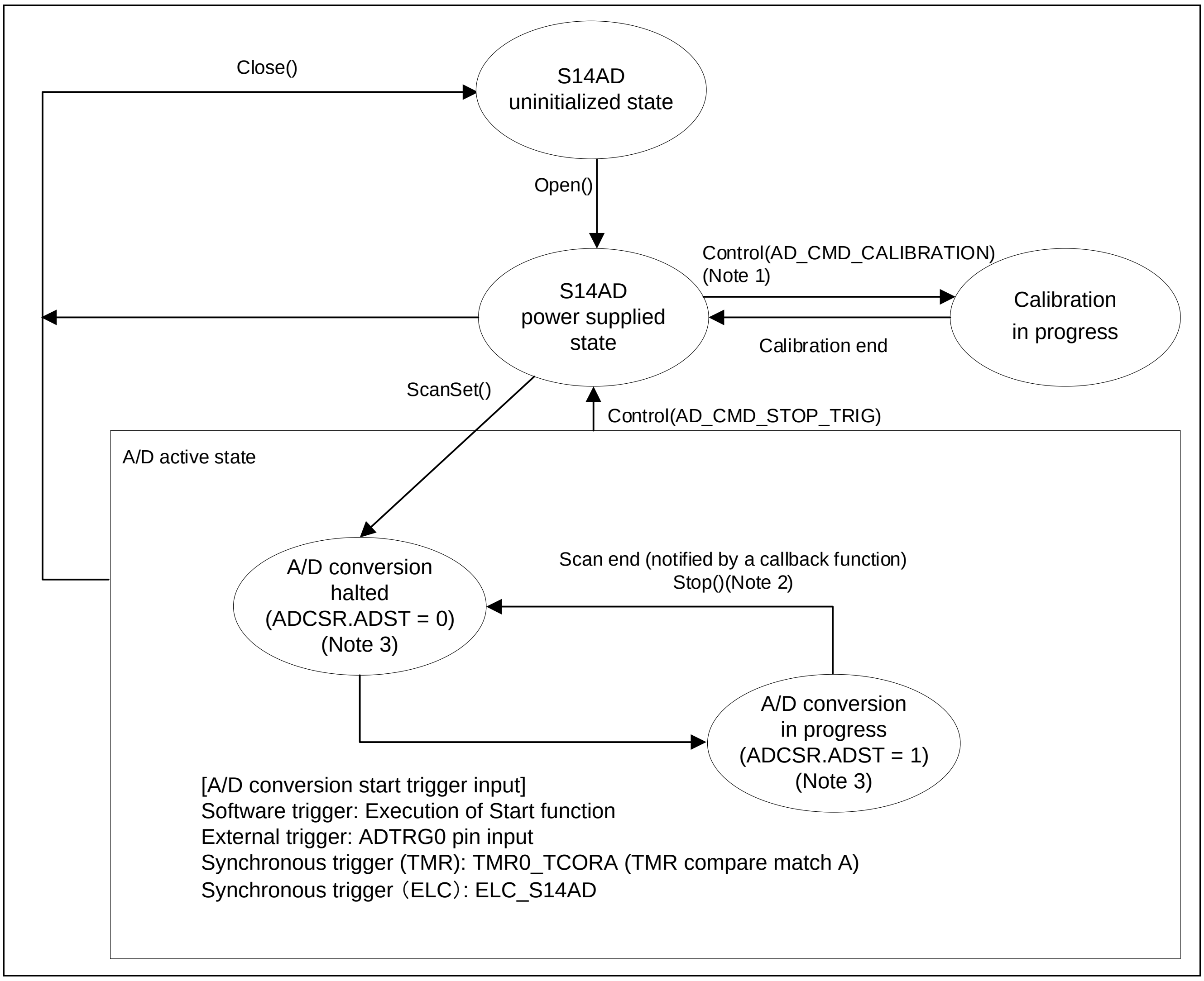

Close()
S14AD uninitialized state
Open()
Control(AD_CMD_CALIBRATION) (Note 1)
S14AD power supplied state
Calibration in progress
Calibration end
ScanSet()
Control(AD_CMD_STOP_TRIG)
A/D active state
A/D conversion halted (ADCSR.ADST = 0) (Note 3)
Scan end (notified by a callback function)
Stop()(Note 2)
A/D conversion in progress (ADCSR.ADST = 1) (Note 3)
[A/D conversion start trigger input]
Software trigger: Execution of Start function
External trigger: ADTRG0 pin input
Synchronous trigger (TMR): TMR0_TCORA (TMR compare match A)
Synchronous trigger（ELC）: ELC_S14AD

Figure 2-60 State Transitions of S14AD Driver

Note 1. After release from the reset state, perform offset calibration before making a transition to the S14AD active state.

Note 2. Transition to the state waiting for an A/D conversion start condition to be entered by Stop() is enabled only when a software trigger has been selected.

Note 3. ADCSR.ADST is set to 1 at the following timing. When all specified A/D conversion operations have finished, ADCSR.ADST automatically becomes 0.

[Software trigger]

When the Start function is executed, set ADCSR.ADST to 1 in the Start function.

[Synchronous trigger and external trigger]

ADCSR.ADST is automatically set to 1 when input of the specified trigger is detected.

Table 2-78 Events and Actions Specific to S14AD Driver State (Note1)

State	Overview	Event	Action
S14AD uninitialized state	The S14D driver is in this state after release from a reset.	Execution of Open() function	Enters the S14AD power supplied state.
S14AD power supplied state	Clock is supplied to the S14AD module in this state.	Execution of Close() function	Enters the S14AD uninitialized state.
		Execution of ScanSet() function	Enters the A/D active state (A/D conversion halted).
		Execution of Control (AD_CMD_CALIBRATION)	Enters the calibration in progress.
Calibration in progress	Calibration is in progress in this state.	Completion of calibration	Enters the S14AD power supplied state.
A/D active state (A/D conversion halted)	A/D conversion is halted.	Execution of Close() function	Enters the S14AD uninitialized state.
		[A/D conversion start condition: Software trigger] Execution of Start() function	Enters the A/D active state (A/D conversion in progress).
		[A/D conversion start condition: External trigger] ADTRG0 pin input	Enters the A/D active state (A/D conversion in progress).
		[A/D conversion start condition: 8-bit timer trigger] TMR0_TCORA (TMR compare match A)	Enters the A/D active state (A/D conversion in progress).
		[A/D conversion start condition: ELC trigger] ELC_S14AD	Enters the A/D active state (A/D conversion in progress).
A/D active state (A/D conversion in progress)	A/D conversion is in progress.	Execution of Close() function	Enters the S14AD uninitialized state.
		Completion of A/D conversion	Enters the A/D active state (A/D conversion halted) and calls the callback function. (Note 2)
		[A/D conversion start condition: Software trigger] Execution of Stop() function	Enters the A/D active state (A/D conversion halted).

Note 1. The GetVersion and Read functions can be executed in any state.

Note 2. Only if a callback function is specified when the Open function is executed and an interrupt is set to be enabled with the Control function, the callback function will be called.

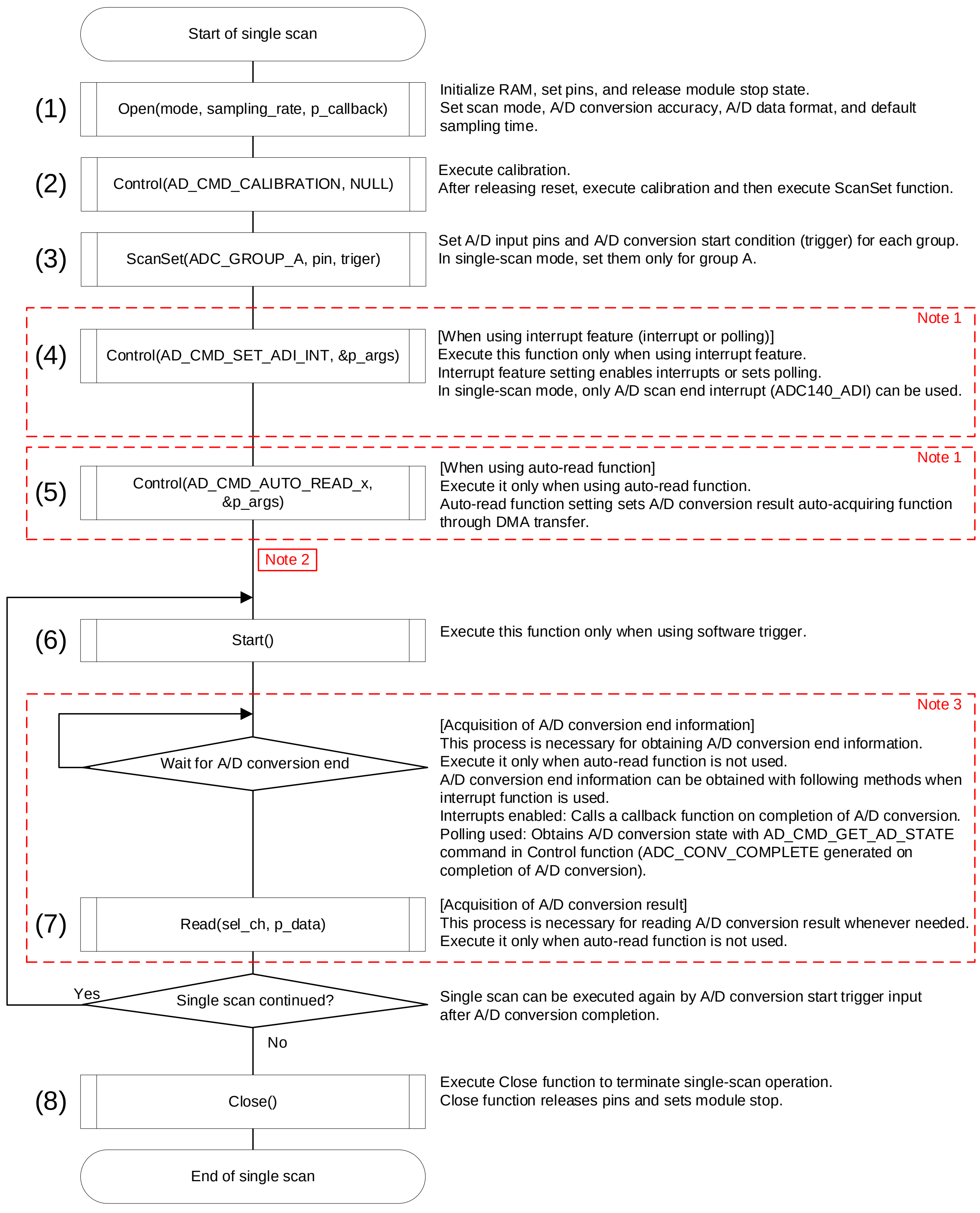
3. Descriptions of Driver Operations

The S14AD driver provides the AD conversion capabilities in single-scan, continuous-scan, and group-scan modes. This chapter describes the execution procedures and operation examples in each scan mode and operating mode.

3.1 Single-Scan Mode

3.1.1 Basic Operation (Group A/Temperature Sensor Output)

When single scan mode is executed, analog input on the specified channels is AD-converted for one cycle only. The single scanning procedure is shown in Figure 3-1.


Start of single scan
(1)
Open(mode, sampling_rate, p_callback)
Initialize RAM, set pins, and release module stop state.
Set scan mode, A/D conversion accuracy, A/D data format, and default sampling time.
(2)
Control(AD_CMD_CALIBRATION, NULL)
Execute calibration.
After releasing reset, execute calibration and then execute ScanSet function.
(3)
ScanSet(ADC_GROUP_A, pin, triger)
Set A/D input pins and A/D conversion start condition (trigger) for each group.
In single-scan mode, set them only for group A.
Note 1
(4)
Control(AD_CMD_SET_ADI_INT, &p_args)
[When using interrupt feature (interrupt or polling)]
Execute this function only when using interrupt feature.
Interrupt feature setting enables interrupts or sets polling.
In single-scan mode, only A/D scan end interrupt (ADC140_ADI) can be used.
Note 1
(5)
Control(AD_CMD_AUTO_READ_x, &p_args)
[When using auto-read function]
Execute it only when using auto-read function.
Auto-read function setting sets A/D conversion result auto-acquiring function through DMA transfer.
Note 2
(6)
Start()
Execute this function only when using software trigger.
Note 3
Wait for A/D conversion end
[Acquisition of A/D conversion end information]
This process is necessary for obtaining A/D conversion end information.
Execute it only when auto-read function is not used.
A/D conversion end information can be obtained with following methods when interrupt function is used.
Interrupts enabled: Calls a callback function on completion of A/D conversion.
Polling used: Obtains A/D conversion state with AD_CMD_GET_AD_STATE command in Control function (ADC_CONV_COMPLETE generated on completion of A/D conversion).
(7)
Read(sel_ch, p_data)
[Acquisition of A/D conversion result]
This process is necessary for reading A/D conversion result whenever needed.
Execute it only when auto-read function is not used.
Yes
Single scan continued?
Single scan can be executed again by A/D conversion start trigger input after A/D conversion completion.
No
(8)
Close()
Execute Close function to terminate single-scan operation.
Close function releases pins and sets module stop.
End of single scan

Note 1: The interrupt function and auto-read function are controlled mutually exclusively; only one of them can be used.
Note 2: Between processes (5) and (6), set the TSCR.TSEN bit to 1 and perform the process to wait for temperature sensor reference voltage to stabilize.
Note 3: When the auto-read function is used, A/D conversion result is obtained automatically; this process is not necessary.

Figure 3-1 Single Scanning Procedure

Table 3-1 List of Functions and Commands Used in Single-Scan Mode

Function and Command	Argument	Description	Details
(1) Open function Function details in 2.4.1	mode	uint32_t type Specifies single-scan mode (ADC_SINGLE_SCAN). Specify A/D conversion accuracy and data format as necessary.	2.7.1
	sampling_rate	uint8_t type Specify default sampling time from among 2 to 255 (ADSSTRn (Note) initial value).	-
	p_callback	adc_cd_event_t type Specify callback function. If NULL is specified, no callback function will be called upon occurrence of an S14AD event.	2.7.22
(2) Control function AD_CMD_CALIBRATION Function details in 2.5.21	cmd	Specify AD_CMD_CALIBRATION.	-
	p_args	Specify NULL.	-
(3) ScanSet function Function details in 2.4.2	group	e_adc_group_t type Specifies group which contains pins to be set. In single-scan mode, only ADC_GROUP_A can be used, which is definition for setting group A.	2.7.6
	pins	st_adc_pins_t type Specifies A/D conversion input pins.	2.8.1
	triger	e_adc_triger_t type Specifies A/D conversion start trigger.	2.7.2
(4) Control function AD_CMD_SET_ADI_INT Function details in 2.5.4	cmd	Specify AD_CMD_SET_ADI_INT.	-
	p_args	e_adc_int_method_t type variable pointer Specifies interrupt setting (enabling or polling). Enabling interrupts: ADC_INT_ENABLE Polling: ADC_INT_POLLING	2.7.10
(5) Control function AD_CMD_AUTO_READ_x (x = NORMAL, BLOCK) Function details (NORMAL) in 2.4.23 Function details (BLOCK) in 2.5.24	cmd	Specify AD_CMD_AUTO_READ_x (x = NORMAL, BLOCK).	-
	p_args	st_adc_dma_read_info_t type variable pointer As each element of st_adc_dma_read_info_t type variable structure, specify DMA transfer trigger, transfer source register, transfer destination RAM, transfer count, callback function, and transfer size.	2.8.8
(6) Start function	-	-	-
(7) Read function	sel_ch	e_adc_ssel_ch_t type Specify one analog input channel from which A/D conversion result is to be read. Specify ADC_SSEL_TEMP to read A/D conversion result of temperature sensor output.	2.7.5
	p_data	uint16_t type variable pointer Specify address in which A/D conversion result is to be stored.	-

Note. 256KB Group : n = 0 to 7,L,T 1500KB Group : n = 0 to 6,L,T

Figure 3-2 shows an example of A/D-conversion on AN000 to AN002 in single-scan mode.

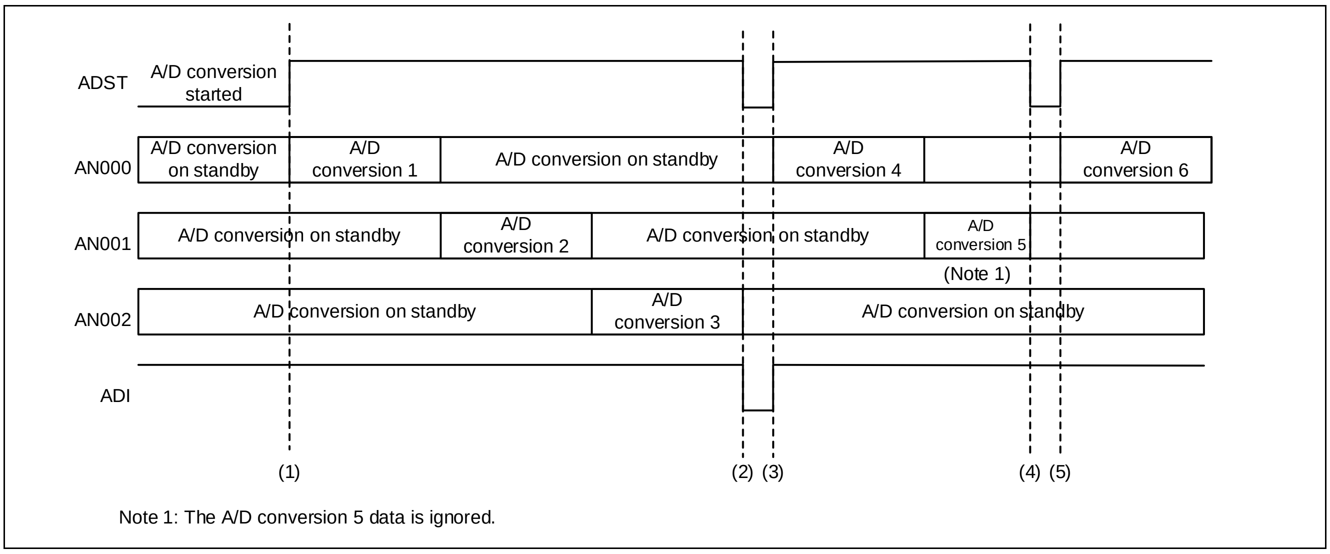

ADST
A/D conversion started
AN000
A/D conversion on standby
A/D conversion 1
A/D conversion on standby
A/D conversion 4
A/D conversion 6
AN001
A/D conversion on standby
A/D conversion 2
A/D conversion on standby
A/D conversion 5
(Note 1)
AN002
A/D conversion on standby
A/D conversion 3
A/D conversion on standby
ADI
(1)
(2) (3)
(4) (5)
Note 1: The A/D conversion 5 data is ignored.

Figure 3-2 Single-Scan Operation

(1) When ADST is set to 1 by A/D conversion start trigger input, A/D conversion is started. The A/D conversion start trigger input differs depending on the A/D conversion start trigger setting. The following shows the A/D conversion start trigger inputs for each setting.

Software trigger: Execution of Start function
External trigger: ADTRG0 pin input
Synchronous trigger (TMR): TMR0_TCORA (TMR compare match A) event detection
Synchronous trigger (ELC): ELC_S14AD event detection

(2) When A/D conversion is completed, ADST is cleared to 0. Here, an ADC140_ADI interrupt request is generated if the ADI interrupt is enabled (ADCSR.ADIE = 1).

A/D conversion end timing is checked by the different manners depending on the interrupt function and auto-read function setting. The following describes how A/D conversion end timing is checked for each function.

[Interrupts]

If a callback function has been specified when executing the Open function and interrupts have been enabled by interrupt function setting, the callback function will be executed at the end of A/D conversion.

[Polling]

If polling has been set by interrupt function setting, the operating state of each A/D conversion group can be obtained by executing the AD_CMD_GET_AD_STATE command of the Control function. ADC_CONV_COMPLETE will be stored as the operating state information of the group whose A/D conversion has been completed. For the application example of the AD_CMD_GET_AD_STATE command and the details of the obtainable information, see section 2.5.17, A/D Converter State Acquisition (AD_CMD_GET_AD_STATE).

[Auto-read]

If the auto-read function is used, the A/D conversion result will be automatically transferred to the RAM at the end of A/D conversion. A notice is not issued each time A/D conversion ends. If a callback function has been specified when setting the auto-read function, the callback function will be executed upon completion of auto-reading for the specified times.

[Interrupts or auto-read function not used]

If neither interrupts nor auto-read function is used, A/D conversion end can be confirmed by detecting the transition from the A/D conversion-in-progress state (ADST = 1) to the A/D conversion stop state (ADST = 0). The state of A/D conversion can be obtained using the AD_CMD_GET_AD_STATE command of the Control function.

(3) When the A/D conversion start trigger is input, A/D conversion is performed again.

(4) If A/D conversion is terminated before A/D conversion is completed, the A/D conversion result will be undefined. The A/D conversion termination procedure differs depending on the A/D conversion start trigger setting. The following shows the A/D conversion termination procedure for each A/D conversion start trigger setting.

[Software trigger]
Executes the Stop function.

[External trigger, synchronous trigger (TMR or ELC)]
Executes the AD_CMD_STOP_TRIG command in the Control function.

(5) If A/D conversion is terminated and then restarted, A/D conversion is started at AN000. The A/D conversion restarting procedure differs depending on the A/D conversion start trigger setting. The following shows the A/D conversion restarting procedure for each A/D conversion start trigger setting.

[Software trigger]
Executes the Start function.

[External trigger, synchronous trigger (TMR or ELC)]
Sets the A/D conversion start trigger again with the ScanSet function, and restarts A/D conversion upon input of the trigger according to the A/D conversion start trigger setting.

3.1.2 Double-Trigger Mode

When double-trigger mode is selected in single-scan mode, two rounds of single-scan operation started by a synchronous trigger (TMR or ELC) are performed as a sequence. The single scanning procedure (double-trigger mode) is shown in Figure 3-3.

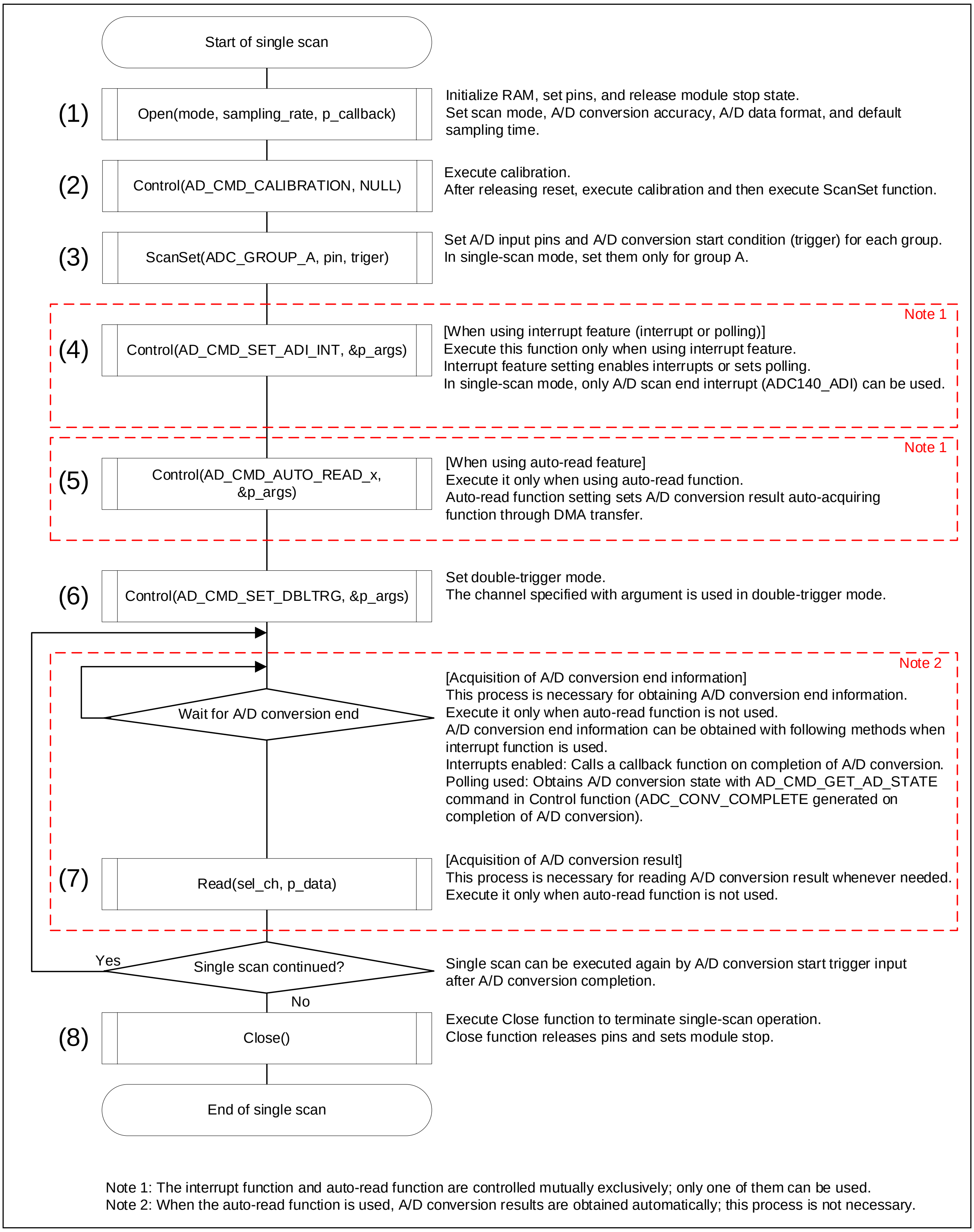

Start of single scan
(1)
Open(mode, sampling_rate, p_callback)
Initialize RAM, set pins, and release module stop state.
Set scan mode, A/D conversion accuracy, A/D data format, and default sampling time.
(2)
Control(AD_CMD_CALIBRATION, NULL)
Execute calibration.
After releasing reset, execute calibration and then execute ScanSet function.
(3)
ScanSet(ADC_GROUP_A, pin, triger)
Set A/D input pins and A/D conversion start condition (trigger) for each group.
In single-scan mode, set them only for group A.
Note 1
(4)
Control(AD_CMD_SET_ADI_INT, &p_args)
[When using interrupt feature (interrupt or polling)]
Execute this function only when using interrupt feature.
Interrupt feature setting enables interrupts or sets polling.
In single-scan mode, only A/D scan end interrupt (ADC140_ADI) can be used.
Note 1
(5)
Control(AD_CMD_AUTO_READ_x, &p_args)
[When using auto-read feature]
Execute it only when using auto-read function.
Auto-read function setting sets A/D conversion result auto-acquiring function through DMA transfer.
(6)
Control(AD_CMD_SET_DBLTRG, &p_args)
Set double-trigger mode.
The channel specified with argument is used in double-trigger mode.
Note 2
Wait for A/D conversion end
[Acquisition of A/D conversion end information]
This process is necessary for obtaining A/D conversion end information.
Execute it only when auto-read function is not used.
A/D conversion end information can be obtained with following methods when interrupt function is used.
Interrupts enabled: Calls a callback function on completion of A/D conversion.
Polling used: Obtains A/D conversion state with AD_CMD_GET_AD_STATE command in Control function (ADC_CONV_COMPLETE generated on completion of A/D conversion).
(7)
Read(sel_ch, p_data)
[Acquisition of A/D conversion result]
This process is necessary for reading A/D conversion result whenever needed.
Execute it only when auto-read function is not used.
Yes
Single scan continued?
Single scan can be executed again by A/D conversion start trigger input after A/D conversion completion.
No
(8)
Close()
Execute Close function to terminate single-scan operation.
Close function releases pins and sets module stop.
End of single scan
Note 1: The interrupt function and auto-read function are controlled mutually exclusively; only one of them can be used.
Note 2: When the auto-read function is used, A/D conversion results are obtained automatically; this process is not necessary.

Figure 3-3 Single Scanning Procedure (Double-Trigger Mode)

Table 3-2 lists the functions and arguments used in single-scan mode (double-trigger mode).

Table 3-2 List of Functions and Commands Used in Single-Scan Mode (Double-Trigger Mode)

Function and Command	Argument	Description	Details
(1) Open function Function details in 2.4.1	mode	uint32_t type Specifies single-scan mode (ADC_SINGLE_SCAN). Specify A/D conversion accuracy and data format as necessary.	2.7.1
	sampling_rate	uint8_t type Specify default sampling time from among 2 to 255 (ADSSTRn (Note) initial value).	-
	p_callback	adc_cd_event_t type Specify callback function. If NULL is specified, no callback function will be called upon occurrence of an S14AD event.	2.7.22
(2) Control function AD_CMD_CALIBRATION Function details in 2.5.21	cmd	Specify AD_CMD_CALIBRATION.	-
	p_args	Specify NULL.	-
(3) ScanSet function Function details in 2.4.2	group	e_adc_group_t type Specifies group which contains pins to be set. In single-scan mode, only ADC_GROUP_A can be used, which is definition for setting group A.	2.7.6
	pins	st_adc_pins_t type Specifies A/D conversion input pins.	2.8.1
	triger	e_adc_triger_t type Specifies A/D conversion start trigger. Use synchronous or asynchronous trigger in double-trigger mode. Software trigger cannot be used.	2.7.2
(4) Control function AD_CMD_SET_ADI_INT Function details in 2.5.4	cmd	Specify AD_CMD_SET_ADI_INT.	-
	p_args	e_adc_int_method_t type variable pointer Specifies interrupt setting (enabling and polling). Enabling interrupts: ADC_INT_ENABLE Polling: ADC_INT_POLLING	2.7.10
(5) Control function AD_CMD_AUTO_READ_x (x = NORMAL, BLOCK) Function details (NORMAL) in 2.4.23 Function details (BLOCK) in 2.5.24	cmd	Specify AD_CMD_AUTO_READ_x (x = NORMAL, BLOCK).	-
	p_args	st_adc_dma_read_info_t type variable pointer As each element of st_adc_dma_read_info_t type variable structure, specify DMA transfer trigger, transfer source register, transfer destination RAM, transfer count, callback function, and transfer size.	2.8.8
(6) Control function AD_CMD_SET_DBLTRG Function details in 2.5.2	cmd	Specify AD_CMD_SET_DBLTRG.	-
	p_args	int8_t type Specify channels to use in double-trigger mode. Specify "-1" to cancel double-trigger mode.	-
(7) Read function	sel_ch	e_adc_ssel_ch_t type Specify one analog input channel from which A/D conversion result is to be read. Specify ADC_SSEL_DBL to read result of A/D conversion triggered by the second trigger.	2.7.5
	p_data	uint16_t type variable pointer Specify address in which A/D conversion result is to be stored	-
(8) Close function	-	-	-

Note. 256KB Group : n = 0 to 7,L,T 1500KB Group : n = 0 to 6,L,T

Figure 3-2 shows an example of A/D-conversion on AN000 to AN002 in single-scan mode.

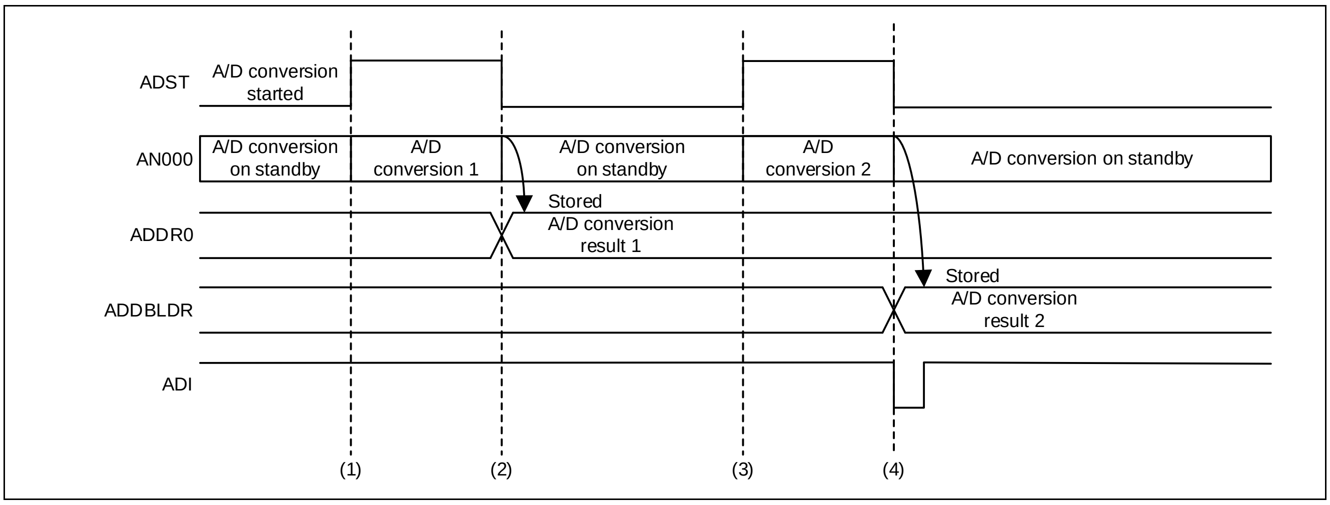

ADST
A/D conversion started
AN000
A/D conversion on standby
A/D conversion 1
A/D conversion on standby
A/D conversion 2
A/D conversion on standby
ADDR0
Stored A/D conversion result 1
ADDBLDR
Stored A/D conversion result 2
ADI
(1)
(2)
(3)
(4)

Figure 3-4 Single-Scan Operation (Double-Trigger Mode)

(1) When ADST is set to 1 by A/D conversion start trigger input, A/D conversion is started. The A/D conversion start trigger input differs depending on the A/D conversion start trigger setting. The following shows the A/D conversion start trigger inputs for each setting.
Software trigger: Execution of Start function
External trigger: ADTRG0 pin input
Synchronous trigger (TMR): TMR0_TCORA (TMR compare match A) event detection
Synchronous trigger (ELC): ELC_S14AD event detection
(2) When A/D conversion is completed, ADST is cleared to 0. The A/D conversion result of AN000 is stored in ADDR0. Here, an S140_ADI interrupt request is not generated.
(3) When ADST is set to 1 by the second A/D conversion start trigger input, A/D conversion is started.
(4) When A/D conversion is completed, ADST is cleared to 0. The A/D conversion result of AN000 is stored in ADDBLDR. Here, an ADC140_ADI interrupt request is generated if the ADI interrupt is enabled (ADCSR.ADIE = 1). A/D conversion end timing is checked by the different manners depending on the interrupt function and auto-read function setting. The following describes how A/D conversion end timing is checked for each function.
[Interrupts]

If a callback function has been specified when executing the Open function and interrupts have been enabled by interrupt function setting, the callback function will be executed at the end of the second A/D conversion.

[Polling]

If polling has been set by interrupt function setting, the operating state of each A/D conversion group can be obtained by executing the AD_CMD_GET_AD_STATE command of the Control function. In double-trigger mode, if the second A/D conversion is completed, ADC_CONV_COMPLETE will be stored as the operating state information of group A. For the application example of the AD_CMD_GET_AD_STATE command and the details of the obtainable information, see section 2.5.17, A/D Converter State Acquisition (AD_CMD_GET_AD_STATE).

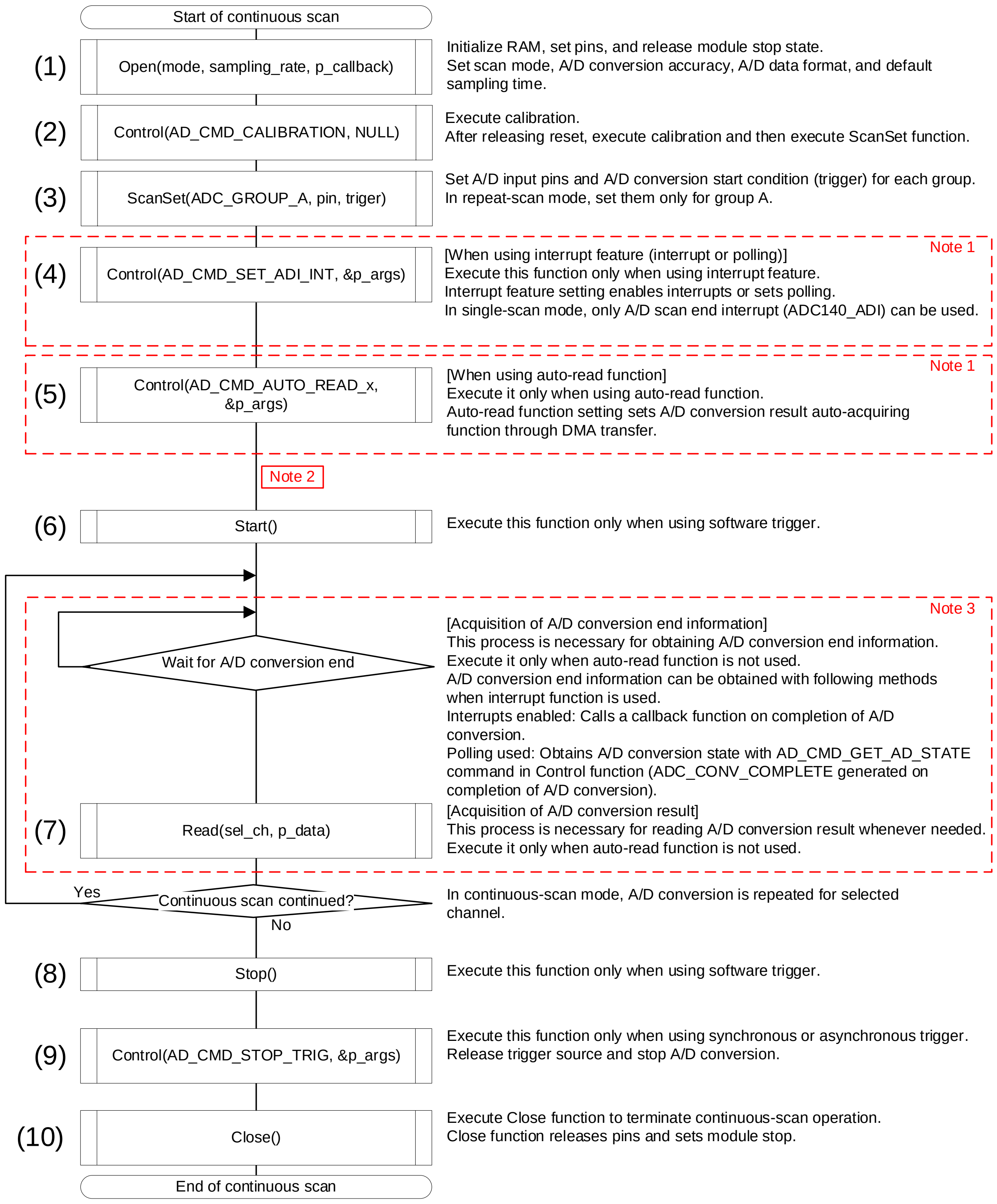
[Auto-read]

If the auto-read function is used, the A/D conversion result will be automatically transferred to the RAM at the end of the second A/D conversion. A notice is not issued each time A/D conversion ends. If a callback function has been specified when setting the auto-read function, the callback function will be executed upon completion of auto-reading for the specified times.

3.2 Continuous-Scan Mode

3.2.1 Basic Operation (Group A/Temperature Sensor Output)

In continuous-scan mode, A/D conversion is performed repeatedly on the analog input of the specified channels.


Start of continuous scan
(1)
Open(mode, sampling_rate, p_callback)
Initialize RAM, set pins, and release module stop state.
Set scan mode, A/D conversion accuracy, A/D data format, and default sampling time.
(2)
Control(AD_CMD_CALIBRATION, NULL)
Execute calibration.
After releasing reset, execute calibration and then execute ScanSet function.
(3)
ScanSet(ADC_GROUP_A, pin, triger)
Set A/D input pins and A/D conversion start condition (trigger) for each group.
In repeat-scan mode, set them only for group A.
(4)
Control(AD_CMD_SET_ADI_INT, &p_args)
Note 1
[When using interrupt feature (interrupt or polling)]
Execute this function only when using interrupt feature.
Interrupt feature setting enables interrupts or sets polling.
In single-scan mode, only A/D scan end interrupt (ADC140_ADI) can be used.
(5)
Control(AD_CMD_AUTO_READ_x, &p_args)
Note 1
[When using auto-read function]
Execute it only when using auto-read function.
Auto-read function setting sets A/D conversion result auto-acquiring function through DMA transfer.
Note 2
(6)
Start()
Execute this function only when using software trigger.
Note 3
Wait for A/D conversion end
[Acquisition of A/D conversion end information]
This process is necessary for obtaining A/D conversion end information.
Execute it only when auto-read function is not used.
A/D conversion end information can be obtained with following methods when interrupt function is used.
Interrupts enabled: Calls a callback function on completion of A/D conversion.
Polling used: Obtains A/D conversion state with AD_CMD_GET_AD_STATE command in Control function (ADC_CONV_COMPLETE generated on completion of A/D conversion).
(7)
Read(sel_ch, p_data)
[Acquisition of A/D conversion result]
This process is necessary for reading A/D conversion result whenever needed.
Execute it only when auto-read function is not used.
Yes
Continuous scan continued?
No
In continuous-scan mode, A/D conversion is repeated for selected channel.
(8)
Stop()
Execute this function only when using software trigger.
(9)
Control(AD_CMD_STOP_TRIG, &p_args)
Execute this function only when using synchronous or asynchronous trigger.
Release trigger source and stop A/D conversion.
(10)
Close()
Execute Close function to terminate continuous-scan operation.
Close function releases pins and sets module stop.
End of continuous scan

Note 1: The interrupt function and auto-read function are controlled mutually exclusively; only one of them can be used.
Note 2: Between processes (5) and (6), set the TSCR.TSEN bit to 1 and perform the process to wait for temperature sensor reference voltage to stabilize.
Note 3: When the auto-read function is used, A/D conversion result is obtained automatically; this process is not necessary.

Figure 3-5 Continuous Scanning Procedure

TableTable 3-3 lists the functions and arguments used in continuous-scan mode.

Table 3-3 List of Functions and Commands Used in Continuous-Scan Mode

Function and Command	Argument	Description	Details
(1) Open function Function details in 2.4.1	mode	uint32_t type Specifies continuous-scan mode (ADC_REPEAT_SCAN). Specify A/D conversion accuracy and data format as necessary.	2.7.1
	sampling_rate	uint8_t type Specify default sampling time from among 2 to 255 (ADSSTRn (Note) initial value).	-
	p_callback	adc_cd_event_t type Specify callback function. If NULL is specified, no callback function will be called upon occurrence of an S14AD event.	2.7.22
(2) Control function AD_CMD_CALIBRATION Function details in 2.5.21	cmd	Specify AD_CMD_CALIBRATION.	-
	p_args	Specify NULL.	-
(3) ScanSet function Function details in 2.4.2	group	e_adc_group_t type Specifies group which contains pins to be set. In continuous-scan mode, only ADC_GROUP_A can be used, which is definition for setting group A.	2.7.6
	pins	st_adc_pins_t type Specifies A/D conversion input pins.	2.8.1
	triger	e_adc_triger_t type Specifies A/D conversion start trigger.	2.7.2
(4) Control function AD_CMD_SET_ADI_INT Function details in 2.5.4	cmd	Specify AD_CMD_SET_ADI_INT.	-
	p_args	e_adc_int_method_t type variable pointer Specifies interrupt setting (enabling and polling). Enabling interrupts: ADC_INT_ENABLE Polling: ADC_INT_POLLING	2.7.10
(5) Control function AD_CMD_AUTO_READ_x (x = NORMAL, BLOCK) Function details (NORMAL) in 2.4.23 Function details (BLOCK) in 2.5.24	cmd	Specify AD_CMD_AUTO_READ_x (x = NORMAL, BLOCK).	-
	p_args	st_adc_dma_read_info_t type variable pointer As each element of st_adc_dma_read_info_t type variable structure, specify DMA transfer trigger, transfer source register, transfer destination RAM, transfer count, callback function, and transfer size.	2.8.8
(6) Start function	-	-	-
(7) Read function	sel_ch	e_adc_ssel_ch_t type Specify one analog input channel from which A/D conversion result is to be read. Specify ADC_SSEL_TEMP to read A/D conversion result of temperature sensor output.	2.7.5
	p_data	uint16_t type variable pointer Specify address in which A/D conversion result is to be stored.	-
(8) Stop function	-	-	-
(9) Control function AD_CMD_STOP_TRIG Function details in 2.5.22	cmd	Specify AD_CMD_STOP_TRIG.	-
	p_args	Specify NULL.	-
(10) Close function	-	-	-

Note. 256KB Group : n = 0 to 7,L,T　1500KB Group : n = 0 to 6,L,T

Figure 3-6 shows an example of A/D-conversion on AN000 to AN002 in continuous-scan mode.

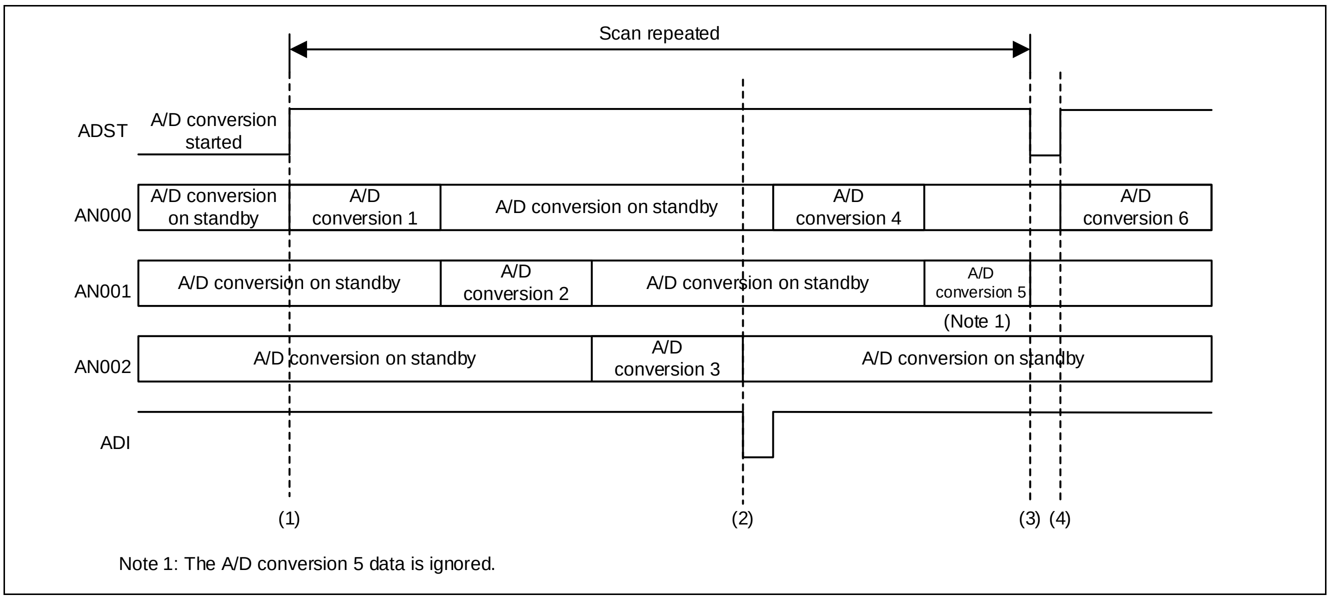

Scan repeated
ADST
A/D conversion started
AN000
A/D conversion on standby
A/D conversion 1
A/D conversion on standby
A/D conversion 4
A/D conversion 6
AN001
A/D conversion on standby
A/D conversion 2
A/D conversion on standby
A/D conversion 5
(Note 1)
AN002
A/D conversion on standby
A/D conversion 3
A/D conversion on standby
ADI
(1)
(2)
(3)
(4)
Note 1: The A/D conversion 5 data is ignored.

Figure 3-6 Continuous-Scan Operation

(1) When ADST is set to 1 by A/D conversion start trigger input, A/D conversion is started. The A/D conversion start trigger input differs depending on the A/D conversion start trigger setting. The following shows the A/D conversion start trigger inputs for each setting.
Software trigger: Execution of Start function
External trigger: ADTRG0 pin input
Synchronous trigger (TMR): TMR0_TCORA (TMR compare match A) event detection
Synchronous trigger (ELC): ELC_S14AD event detection

(2) When A/D conversion of all the selected channels is completed, an ADI interrupt request is generated if an ADI interrupt is enabled (ADCSR.ADIE = 1). A/D conversion end timing is checked by the different manners depending on the interrupt function and auto-read function setting. The following describes how A/D conversion end timing is checked for each function.
[Interrupts]
If a callback function has been specified when executing the Open function and interrupts have been enabled by interrupt function setting, the callback function will be executed at the end of A/D conversion.
[Polling]
If polling has been set by interrupt function setting, the operating state of each A/D conversion group can be obtained by executing the AD_CMD_GET_AD_STATE command of the Control function. ADC_CONV_COMPLETE will be stored as the operating state information of the group whose A/D conversion has been completed. For the application example of the AD_CMD_GET_AD_STATE command and the details of the obtainable information, see section 2.5.17, A/D Converter State Acquisition (AD_CMD_GET_AD_STATE).
[Auto-read]
If the auto-read function is used, the A/D conversion result will be automatically transferred to the RAM at the end of A/D conversion. A notice is not issued each time A/D conversion ends. If a callback function has been specified when setting the auto-read function, the callback function will be executed upon completion of auto-reading for the specified times.

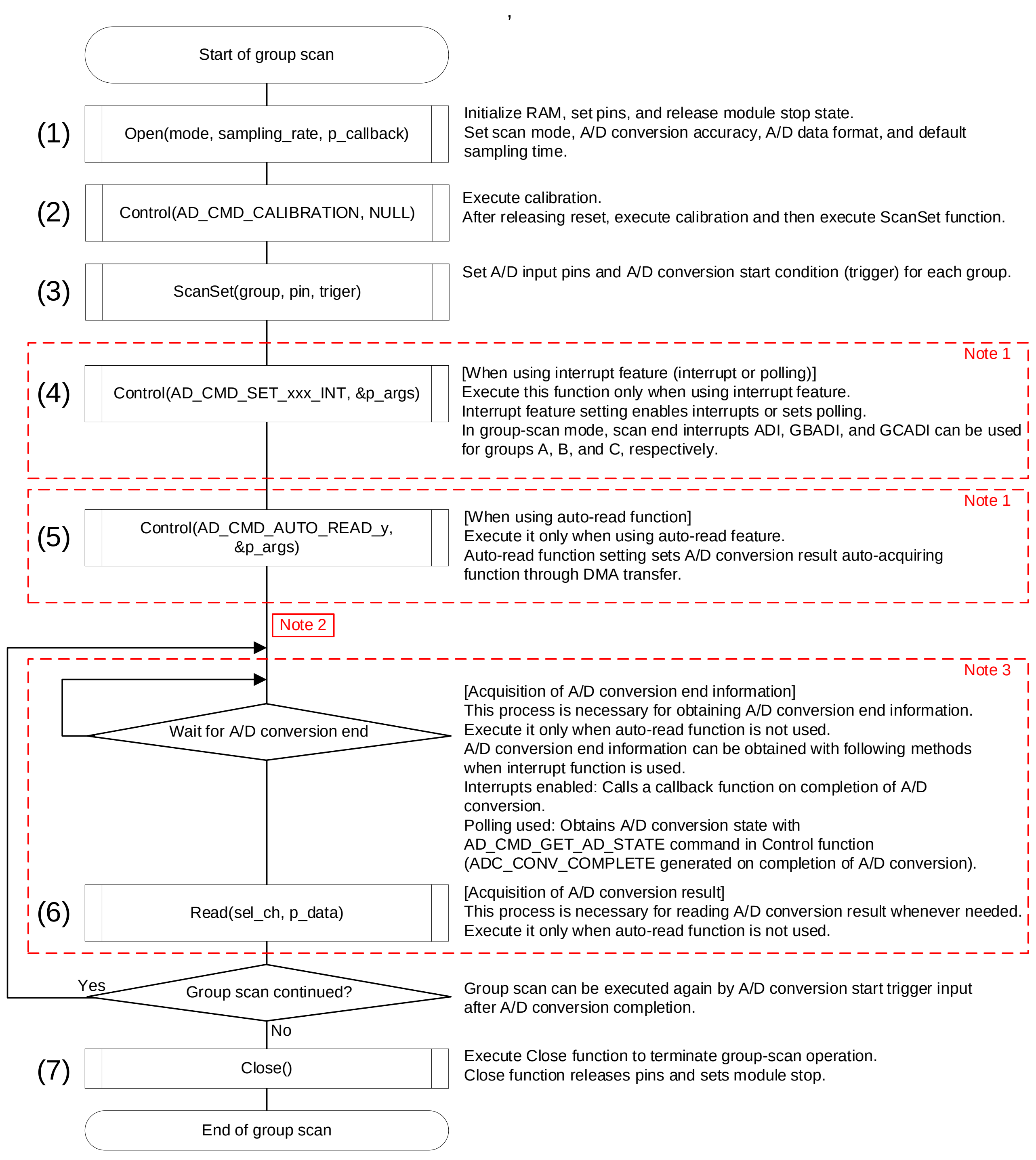
(3) When ADST is cleared to 0, A/D conversion is terminated. The A/D conversion termination procedure differs depending on the A/D conversion start trigger setting. The following shows the A/D conversion termination procedure for each A/D conversion start trigger setting.
[Software trigger]
Executes the Stop function.
[Synchronous trigger/asynchronous trigger]
Executes the AD_CMD_STOP_TRIG command in the Control function.

(4) If a software trigger has been set, the A/D conversion is restarted by execution of the Start function.
If a synchronous or an asynchronous trigger has been set, the A/D conversion start trigger needs to be set again with the ScanSet function. After the A/D conversion start trigger is set, A/D conversion is started upon A/D conversion start trigger input.

3.3 Group-Scan Mode

3.3.1 Basic Operation (Group A/ Group B/Group C/Temperature Sensor Output)

The number of groups to be used in group-scan mode can be selected from two (groups A and B) and three (groups A, B, and C). As the basic operation in group-scan mode, A/D conversion is performed once on the analog inputs of all the specified channels in groups A and B or groups A, B, and C after scanning is started by the respective synchronous trigger. The scan operation of each group is similar to the scan operation in single-scan mode.


Start of group scan
(1)
Open(mode, sampling_rate, p_callback)
Initialize RAM, set pins, and release module stop state.
Set scan mode, A/D conversion accuracy, A/D data format, and default sampling time.
(2)
Control(AD_CMD_CALIBRATION, NULL)
Execute calibration.
After releasing reset, execute calibration and then execute ScanSet function.
(3)
ScanSet(group, pin, triger)
Set A/D input pins and A/D conversion start condition (trigger) for each group.
Note 1
(4)
Control(AD_CMD_SET_xxx_INT, &p_args)
[When using interrupt feature (interrupt or polling)]
Execute this function only when using interrupt feature.
Interrupt feature setting enables interrupts or sets polling.
In group-scan mode, scan end interrupts ADI, GBADI, and GCADI can be used for groups A, B, and C, respectively.
Note 1
(5)
Control(AD_CMD_AUTO_READ_y, &p_args)
[When using auto-read function]
Execute it only when using auto-read feature.
Auto-read function setting sets A/D conversion result auto-acquiring function through DMA transfer.
Note 2
Note 3
Wait for A/D conversion end
[Acquisition of A/D conversion end information]
This process is necessary for obtaining A/D conversion end information.
Execute it only when auto-read function is not used.
A/D conversion end information can be obtained with following methods when interrupt function is used.
Interrupts enabled: Calls a callback function on completion of A/D conversion.
Polling used: Obtains A/D conversion state with AD_CMD_GET_AD_STATE command in Control function (ADC_CONV_COMPLETE generated on completion of A/D conversion).
(6)
Read(sel_ch, p_data)
[Acquisition of A/D conversion result]
This process is necessary for reading A/D conversion result whenever needed.
Execute it only when auto-read function is not used.
Yes
Group scan continued?
Group scan can be executed again by A/D conversion start trigger input after A/D conversion completion.
No
(7)
Close()
Execute Close function to terminate group-scan operation.
Close function releases pins and sets module stop.
End of group scan

Note 1: The interrupt function and auto-read function are controlled mutually exclusively; only one of them can be used.
Note 2: Between processes (5) and (6), set the TSCR.TSEN bit to 1 and perform the process to wait for temperature sensor reference voltage to stabilize.
Note 3: When the auto-read function is used, A/D conversion result is obtained automatically; this process is not necessary.

Figure 3-7 Group Scanning Procedure

Table 3-4 lists the functions and arguments used in group-scan mode.

Table 3-4 List of Functions and Commands Used in Group-Scan Mode

Function and Command	Argument	Description	Details
(1) Open function Function details in 2.4.1	mode	uint32_t type Specifies group-scan mode (ADC_GROUP_SCAN). Specify A/D conversion accuracy and data format as necessary.	2.7.1
	sampling_rate	uint8_t type Specify default sampling time from among 2 to 255 (ADSSTRn (Note) initial value).	-
	p_callback	adc_cd_event_t type Specify callback function. If NULL is specified, no callback function will be called upon occurrence of an S14AD event.	2.7.22
(2) Control function AD_CMD_CALIBRATION Function details in 2.5.21	cmd	Specify AD_CMD_CALIBRATION.	-
	p_args	Specify NULL.	-
(3) ScanSet function Function details in 2.4.2	group	e_adc_group_t type Specifies group which contains pins to be set. When using two groups, set pins for groups A and B separately. When using three groups, set pins for groups A, B, and C separately.	2.7.6
	pins	st_adc_pins_t type Specifies A/D conversion input pins.	2.8.1
	triger	e_adc_triger_t type Specifies A/D conversion start trigger. In group-scan mode, a software trigger cannot be used.	2.7.2
(4) Control function AD_CMD_SET_xxx_INT (xxx = ADI, GROUPB, GROUPC) Function details (ADI) in 2.5.4 Function details (GROUPB) in 2.5.5 Function details (GROUPC) in 2.5.6	cmd	e_adc_cmd_t type Execute a command for each interrupt request to use.	-
	p_args	e_adc_int_method_t type variable pointer Specifies interrupt setting (enabling and polling). Enabling interrupts: ADC_INT_ENABLE Polling: ADC_INT_POLLING	2.7.10
(5) Control function AD_CMD_AUTO_READ_y (y = NORMAL, BLOCK) Function details (NORMAL) in 2.4.23 Function details (BLOCK) in 2.5.24	cmd	Specify AD_CMD_AUTO_READ_y (y = NORMAL, BLOCK).	-
	p_args	st_adc_dma_read_info_t type variable pointer As each element of st_adc_dma_read_info_t type variable structure, specify DMA transfer trigger, transfer source register, transfer destination RAM, transfer count, callback function, and transfer size.	2.8.8
(6) Read function	sel_ch	e_adc_ssel_ch_t type Specify one analog input channel from which A/D conversion result is to be read. Specify ADC_SSEL_TEMP to read A/D conversion result of temperature sensor output.	2.7.5
	p_data	uint16_t type variable pointer Specify address in which A/D conversion result is to be stored.	-
(7) Close function	-	-	-

Note. 256KB Group : n = 0 to 7,L,T 1500KB Group : n = 0 to 6,L,T

Figure 3-8 shows an example of A/D-conversion in group-scan mode (groups A, B, and C used). In the shown example, the A/D conversion start triggers used for each group are as follows.
Group A: Asynchronous trigger (ADTRG0 pin input)
Group B: Synchronous trigger (ELC_S14AD event)
Group C: Synchronous trigger (TMR_TCORA (TMR compare match A))

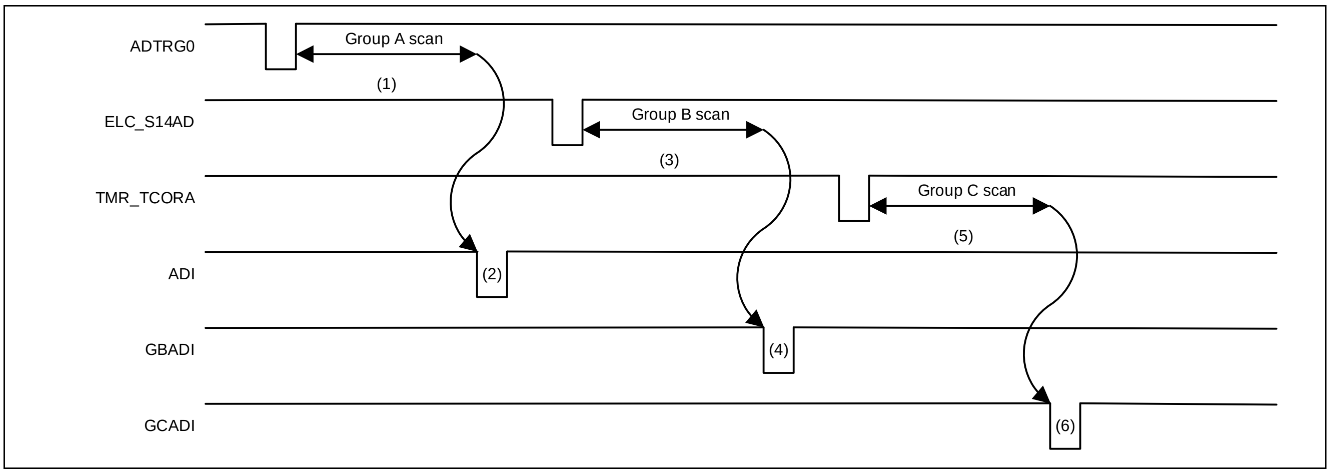


Figure 3-8 Group-Scan Operation

(1) When ADTRG0 input is detected, A/D conversion of group A is performed.
(2) When A/D conversion of group A is completed, an ADC140_ADI interrupt request is generated if an ADI interrupt is enabled (ADCSR.ADIE = 1). A/D conversion end timing is checked by the different manners depending on the interrupt function and auto-read function setting. The following describes how A/D conversion end timing is checked for each function.
[Interrupts]
If a callback function has been specified when executing the Open function and interrupts have been enabled by interrupt function setting, the callback function will be executed at the end of A/D conversion.
[Polling]
If polling has been set by interrupt function setting, the operating state of each A/D conversion group can be obtained by executing the AD_CMD_GET_AD_STATE command of the Control function. ADC_CONV_COMPLETE will be stored as the operating state information of the group whose A/D conversion has been completed. For the application example of the AD_CMD_GET_AD_STATE command and the details of the obtainable information, see section 2.5.17, A/D Converter State Acquisition (AD_CMD_GET_AD_STATE).
[Auto-read]
If the auto-read function is used, the A/D conversion result will be automatically transferred to the RAM at the end of the A/D conversion on the group specified as the DMA activation source. A notice is not issued each time A/D conversion ends. If a callback function has been specified when setting the auto-read function, the callback function will be executed upon completion of auto-reading for the specified times.
(3) When ELC_S14AD event is detected, A/D conversion of group B is performed.
(4) When A/D conversion of group B is completed, an ADC140_GBADI interrupt request is generated if a GBADI interrupt is enabled (ADCSR.GBADIE = 1). A/D conversion end timing can be checked by the different manners depending on the interrupt function and auto-read function setting. A/D conversion end timing for each function can be checked in the same manner as that described in 2.
(5) When TMR_TCORA event is detected, A/D conversion of group C is performed.
(6) When A/D conversion of group C is completed, an ADC140_GCADI interrupt request is generated if a GCADI interrupt is enabled (ADCSR.GCADIE = 1). A/D conversion end timing can be checked by the different manners depending on the interrupt function and auto-read function setting. A/D conversion end timing for each function can be checked in the same manner as that described in 2.

3.3.2 Double-Trigger Mode

When double-trigger mode is selected in double-scan mode, two rounds of single-scan operation started by an asynchronous trigger or a synchronous trigger (TMR or ELC) are performed in sequence for group A. Scan operations in groups B and C are the same as operations in single-scan mode started by a synchronous trigger (TMR or ELC).

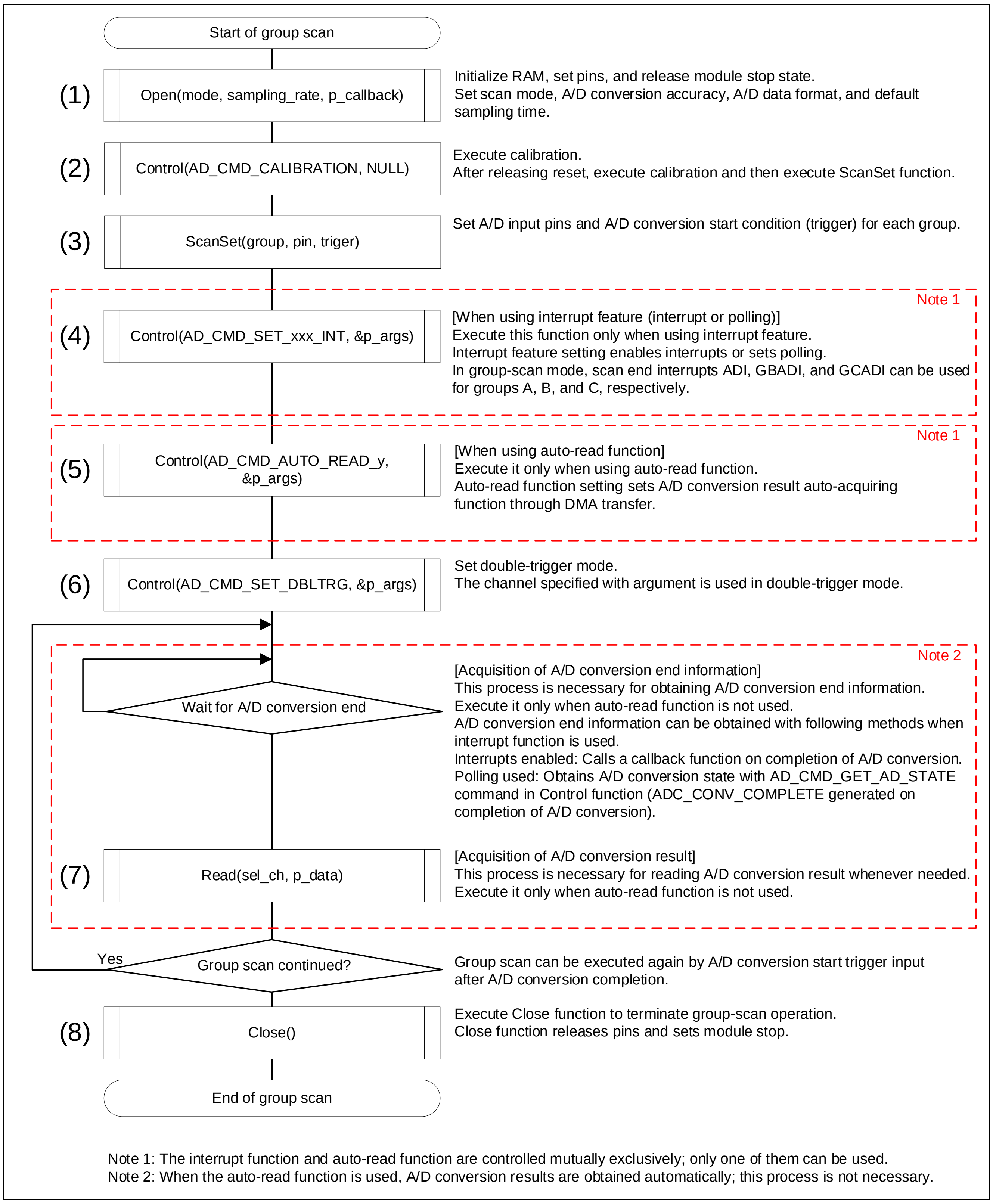

Start of group scan
(1)
Open(mode, sampling_rate, p_callback)
Initialize RAM, set pins, and release module stop state.
Set scan mode, A/D conversion accuracy, A/D data format, and default sampling time.
(2)
Control(AD_CMD_CALIBRATION, NULL)
Execute calibration.
After releasing reset, execute calibration and then execute ScanSet function.
(3)
ScanSet(group, pin, triger)
Set A/D input pins and A/D conversion start condition (trigger) for each group.
Note 1
(4)
Control(AD_CMD_SET_xxx_INT, &p_args)
[When using interrupt feature (interrupt or polling)]
Execute this function only when using interrupt feature.
Interrupt feature setting enables interrupts or sets polling.
In group-scan mode, scan end interrupts ADI, GBADI, and GCADI can be used for groups A, B, and C, respectively.
Note 1
(5)
Control(AD_CMD_AUTO_READ_y, &p_args)
[When using auto-read function]
Execute it only when using auto-read function.
Auto-read function setting sets A/D conversion result auto-acquiring function through DMA transfer.
(6)
Control(AD_CMD_SET_DBLTRG, &p_args)
Set double-trigger mode.
The channel specified with argument is used in double-trigger mode.
Note 2
Wait for A/D conversion end
[Acquisition of A/D conversion end information]
This process is necessary for obtaining A/D conversion end information.
Execute it only when auto-read function is not used.
A/D conversion end information can be obtained with following methods when interrupt function is used.
Interrupts enabled: Calls a callback function on completion of A/D conversion.
Polling used: Obtains A/D conversion state with AD_CMD_GET_AD_STATE command in Control function (ADC_CONV_COMPLETE generated on completion of A/D conversion).
(7)
Read(sel_ch, p_data)
[Acquisition of A/D conversion result]
This process is necessary for reading A/D conversion result whenever needed.
Execute it only when auto-read function is not used.
Yes
Group scan continued?
Group scan can be executed again by A/D conversion start trigger input after A/D conversion completion.
(8)
Close()
Execute Close function to terminate group-scan operation.
Close function releases pins and sets module stop.
End of group scan
Note 1: The interrupt function and auto-read function are controlled mutually exclusively; only one of them can be used.
Note 2: When the auto-read function is used, A/D conversion results are obtained automatically; this process is not necessary.

Figure 3-9 Group Scanning Procedure (Double-Trigger Mode)

Table 3-3 and Table 3-6 list the functions and arguments used in group-scan mode.

Table 3-5 List of Functions and Commands Used in Group-Scan Mode (1/2)

Function and Command	Argument	Description	Details
(1) Open function Function details in 2.4.1	mode	uint32_t type Specifies group-scan mode (ADC_GROUP_SCAN). Specify A/D conversion accuracy and data format as necessary.	2.7.1
	sampling_rate	uint8_t type Specify default sampling time from among 2 to 255 (ADSSTRn (Note) initial value).	-
	p_callback	adc_cd_event_t type Specify callback function. If NULL is specified, no callback function will be called upon occurrence of an S14AD event.	2.7.22
(2) Control function AD_CMD_CALIBRATION Function details in 2.5.21	cmd	Specify AD_CMD_CALIBRATION.	-
	p_args	Specify NULL.	-
(3) ScanSet function Function details in 2.4.2	group	e_adc_group_t type Specifies group which contains pins to be set. When using two groups, set pins for groups A and B separately. When using three groups, set pins for groups A, B, and C separately.	2.7.6
	pins	st_adc_pins_t type Specifies A/D conversion input pins.	2.8.1
	triger	e_adc_triger_t type Specifies A/D conversion start trigger. Software trigger cannot be used in group-scan mode.	2.7.2
(4) Control function AD_CMD_SET_xxx_INT (xxx = ADI, GROUPB, GROUPC) Function details (ADI) in 2.5.4 Function details (GROUPB) in 2.5.5 Function details (GROUPC) in 2.5.6	cmd	e_adc_cmd_t type Execute a command for each interrupt request to use. In group-scan mode, the following interrupt requests can be used. AD_CMD_SET_ADI_INT: Group A scan end interrupt AD_CMD_SET_GROUPB_INT: Group B scan end interrupt AD_CMD_SET_GROUPC_INT: Group C scan end interrupt	-
	p_args	e_adc_int_method_t type variable pointer Specifies interrupt setting (enabling and polling). Enabling interrupts: ADC_INT_ENABLE Polling: ADC_INT_POLLING	2.7.10
(5) Control function AD_CMD_AUTO_READ_y (y = NORMAL, BLOCK) Function details (NORMAL) in 2.4.23 Function details (BLOCK) in 2.5.24	cmd	Specify AD_CMD_AUTO_READ_y (y = NORMAL, BLOCK).	-
	p_args	st_adc_dma_read_info_t type variable pointer As each element of st_adc_dma_read_info_t type variable structure, specify DMA transfer trigger, transfer source register, transfer destination RAM, transfer count, callback function, and transfer size.	2.8.8

Note. 256KB Group : n = 0 to 7,L,T 1500KB Group : n = 0 to 6,L,T

Table 3-6 List of Functions and Commands Used in Group-Scan Mode (2/2)

Function and Command	Argument	Description	Details
(6) Control function AD_CMD_SET_DBLTRG Function details in 2.5.2	cmd	Specify AD_CMD_SET_DBLTRG.	-
	p_args	int8_t type Specify channels to use in double-trigger mode. Only group A channels can be specified in double-trigger mode. Specify ”-1” to cancel double-trigger mode.	-
(7) Read function	sel_ch	e_adc_ssel_ch_t type Specify one analog input channel from which A/D conversion result is to be read. Specify ADC_SSEL_DBL to read result of A/D conversion triggered by the second trigger.	2.7.5
	p_data	uint16_t type variable pointer Specify address in which A/D conversion result is to be stored.	-
(8) Close function	-	-	-

Figure 3-8, Group-Scan Operation (Double-Trigger Mode), shows an example of A/D conversion in group-scan mode (groups A, B, and C used). In the shown example, the A/D conversion start triggers used for each group are as follows.

Group A: Asynchronous trigger (ADTRG0 pin input)
Group B: Synchronous trigger (ELC_S14AD event)
Group C: Synchronous trigger (TMR_TCORA (TMR compare match A))

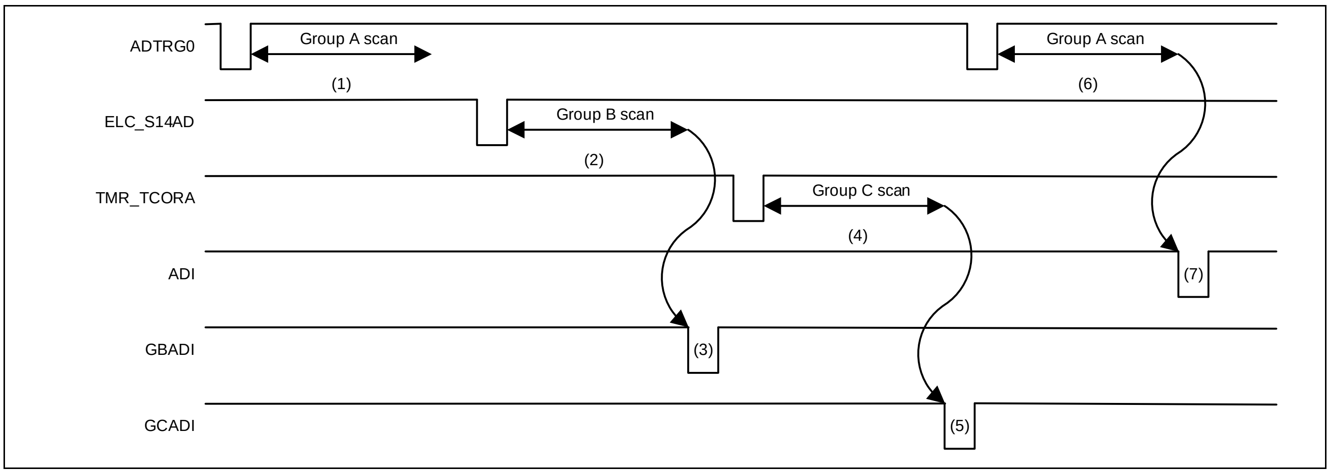


Figure 3-10 Group-Scan Operation (Double-Trigger Mode)

(1) When ADTRG0 input is detected, A/D conversion of group A is performed. The A/D conversion result will be transferred to the ADDRn (Note) corresponding to the group A conversion channels.
(2) When the ELC_S14AD event is detected, A/D conversion of group B is performed.
(3) When A/D conversion of group B is completed, an ADC140_GBADI interrupt request is generated if a GBADI interrupt is enabled (ADCSR.GBADIE = 1). A/D conversion end timing is checked by the different manners depending on the interrupt function and auto-read function setting. The following describes how A/D conversion end timing is checked for each function.
[Interrupts]

If a callback function has been specified when executing the Open function and interrupts have been enabled by interrupt function setting, the callback function will be executed at the end of A/D conversion.

[Polling]

If polling has been set by interrupt function setting, the operating state of each A/D conversion group can be obtained by executing the AD_CMD_GET_AD_STATE command of the Control function. ADC_CONV_COMPLETE will be stored as the operating state information of the group whose A/D conversion has been completed. For the application example of the AD_CMD_GET_AD_STATE command and the details of the obtainable information, see section 2.5.17, A/D Converter State Acquisition (AD_CMD_GET_AD_STATE).

[Auto-read]

If the auto-read function is used, the A/D conversion result will be automatically transferred to the RAM at the end of the A/D conversion on the group specified as the DMA activation source. A notice is not issued each time A/D conversion ends. If a callback function has been specified when setting the auto-read function, the callback function will be executed upon completion of auto-reading for the specified times.
(4) When TMR_TCORA event is detected, A/D conversion of group C is performed.
(5) When the A/D conversion of group C is completed, an ADC140_GCADI interrupt request is generated if a GCADI interrupt is enabled (ADGCTRGR.GCADIE = 1). A/D conversion end timing of group C is checked by the different manners depending on the interrupt function and auto-read function setting. A/D conversion end timing for each function is checked in the same manner as that described in 3.
(6) When the second ADTRG0 input is detected, A/D conversion of group A is performed. The A/D conversion result will be transferred to the ADDBLDR.
(7) When the second scanning of group A is completed, an ADC140_ADI interrupt request is generated if an ADI interrupt is enabled (ADCSR.ADIE = 1). A/D conversion end timing of group A is checked by the different manners depending on the interrupt function and auto-read function setting. A/D conversion end timing for each function is checked in the same manner as that described in 3.

Note. 256KB Group : n = 00 to 07,16,17,20 to 21 1500KB Group : n = 00 to 06,16,17,20 to 28

3.4 Compare Function

3.4.1 Compare Function (Window A/Window B/Temperature Sensor Output)

The compare function compares a reference value set in a register with the A/D conversion result. The reference value can be set for window A and window B independently.

The event output of the compare function outputs the ADC140_WCMPM or ADC140_WCMPUM event according to the window complex conditions (A or B, A and B, or A exor B) depending on whether the comparison result of window A and window B is met. When using the event output of the compare function, use single-scan mode.

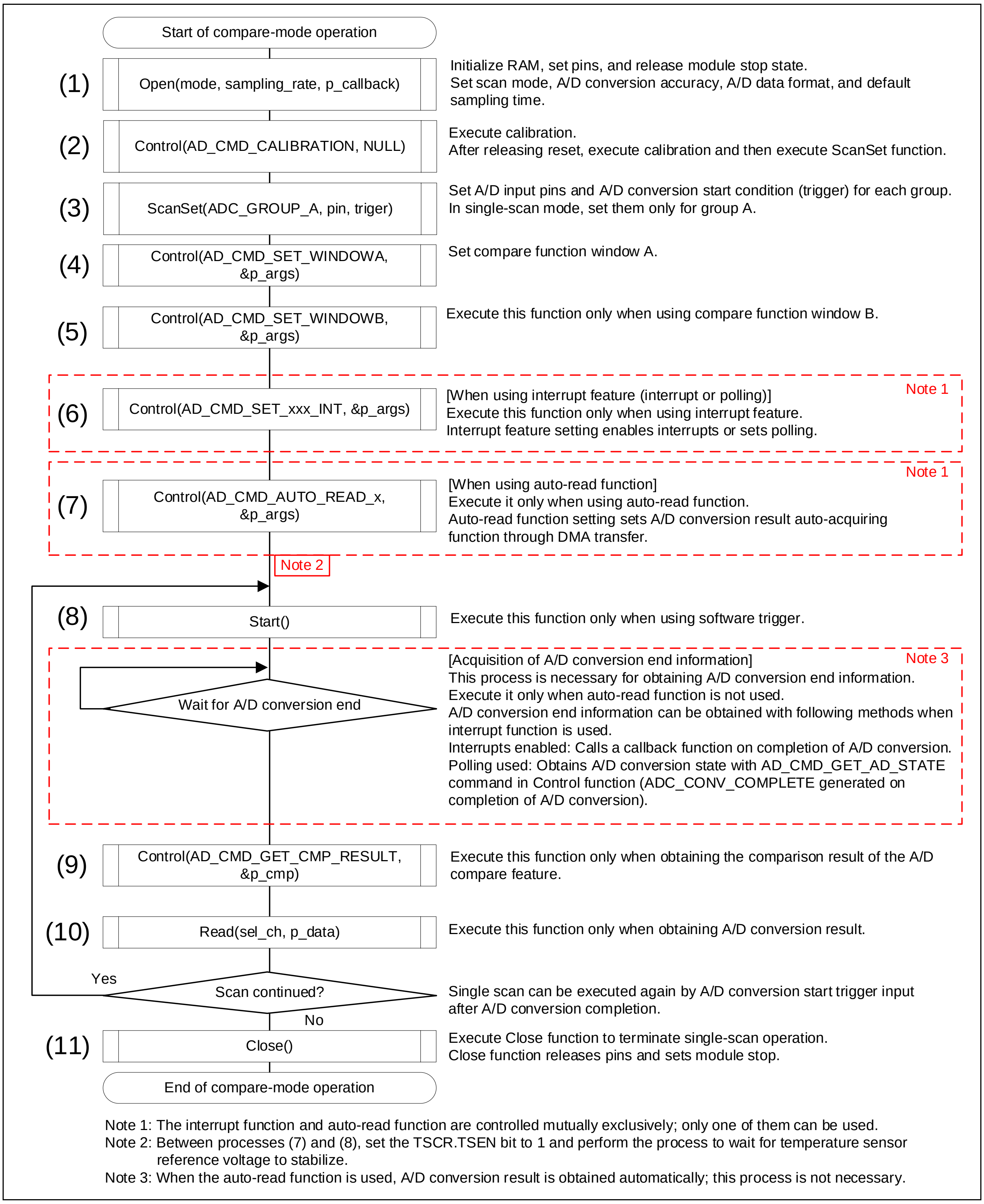

Start of compare-mode operation
(1)
Open(mode, sampling_rate, p_callback)
Initialize RAM, set pins, and release module stop state.
Set scan mode, A/D conversion accuracy, A/D data format, and default sampling time.
(2)
Control(AD_CMD_CALIBRATION, NULL)
Execute calibration.
After releasing reset, execute calibration and then execute ScanSet function.
(3)
ScanSet(ADC_GROUP_A, pin, triger)
Set A/D input pins and A/D conversion start condition (trigger) for each group.
In single-scan mode, set them only for group A.
(4)
Control(AD_CMD_SET_WINDOWA, &p_args)
Set compare function window A.
(5)
Control(AD_CMD_SET_WINDOWB, &p_args)
Execute this function only when using compare function window B.
(6)
Control(AD_CMD_SET_xxx_INT, &p_args)
Note 1
[When using interrupt feature (interrupt or polling)]
Execute this function only when using interrupt feature.
Interrupt feature setting enables interrupts or sets polling.
(7)
Control(AD_CMD_AUTO_READ_x, &p_args)
Note 1
[When using auto-read function]
Execute it only when using auto-read function.
Auto-read function setting sets A/D conversion result auto-acquiring function through DMA transfer.
Note 2
(8)
Start()
Execute this function only when using software trigger.
Wait for A/D conversion end
Note 3
[Acquisition of A/D conversion end information]
This process is necessary for obtaining A/D conversion end information.
Execute it only when auto-read function is not used.
A/D conversion end information can be obtained with following methods when interrupt function is used.
Interrupts enabled: Calls a callback function on completion of A/D conversion.
Polling used: Obtains A/D conversion state with AD_CMD_GET_AD_STATE command in Control function (ADC_CONV_COMPLETE generated on completion of A/D conversion).
(9)
Control(AD_CMD_GET_CMP_RESULT, &p_cmp)
Execute this function only when obtaining the comparison result of the A/D compare feature.
(10)
Read(sel_ch, p_data)
Execute this function only when obtaining A/D conversion result.
Yes
Scan continued?
No
Single scan can be executed again by A/D conversion start trigger input after A/D conversion completion.
(11)
Close()
Execute Close function to terminate single-scan operation.
Close function releases pins and sets module stop.
End of compare-mode operation
Note 1: The interrupt function and auto-read function are controlled mutually exclusively; only one of them can be used.
Note 2: Between processes (7) and (8), set the TSCR.TSEN bit to 1 and perform the process to wait for temperature sensor reference voltage to stabilize.
Note 3: When the auto-read function is used, A/D conversion result is obtained automatically; this process is not necessary.

Figure 3-11 Compare Mode Execution Procedure

Table 3-7 and Table 3-8 list the functions and arguments used in compare mode.

Table 3-7 List of Functions and Commands Used in Compare Mode (1/2)

Function and Command	Argument	Description	Details
(1) Open function Function details in 2.4.1	mode	uint32_t type Specifies scan mode. Set to single-scan mode to use event output of compare function (ADC140_WCMPM/ADC140_WCMPUM). Specify A/D conversion accuracy and data format as necessary.	2.7.1
	sampling_rate	uint8_t type Specify default sampling time from among 2 to 255 (ADSSTRn (Note) initial value).	-
	p_callback	adc_cd_event_t type Specify callback function. If NULL is specified, no callback function will be called upon occurrence of an S14AD event.	2.7.22
(2) Control function AD_CMD_CALIBRATION Function details in 2.5.21	cmd	Specify AD_CMD_CALIBRATION.	-
	p_args	Specify NULL.	-
(3) ScanSet function Function details in 2.4.2	group	e_adc_group_t type Specifies group which contains pins to be set. Pin setting is required for each group to be used.	2.7.6
	pins	st_adc_pins_t type Specifies A/D conversion input pins.	2.8.1
	triger	e_adc_triger_t type Specifies A/D conversion start trigger.	2.7.2
(4) Control function AD_CMD_SET_WINDOWA Function details in 2.5.13	cmd	Specify AD_CMD_SET_WINDOWA.	-
	p_args	st_adc_wina_t type variable pointer As each element of st_adc_wina_t type structure variables, specify CMPAI interrupt setting, compare conditions, compared levels, compared pins, temperature sensor setting, compare conditions for each channel, and temperature sensor compare conditions.	2.8.3
(5) Control function AD_CMD_SET_WINDOWB Function details in 2.5.14	cmd	Specify AD_CMD_SET_WINDOWA.	-
	p_args	st_adc_winb_t type variable pointer As each element of st_adc_winb_t type structure variables, specify CMPBI interrupt setting, compare conditions, window A/B complex conditions, compared levels, compared pins, and temperature sensor setting.	2.8.4

Note 256KB Group : n = 0 to 7,L,T   1500KB Group : n = 0 to 6,L,T

Table 3-8 List of Functions and Commands Used in Compare Mode (2/2)

Function and Command	Argument	Description	Details
(6) Control function AD_CMD_SET_xxx_INT (xxx = ADI, GROUPB, GROUPC, WCMPM, WCMPUM) Function details (ADI) in 2.5.4 Function details (GROUPB) in 2.5.5 Function details (GROUPC) in 2.5.6 Function details (WCMPM) in 2.5.7 Function details (WCMPUM) in 2.5.8	cmd	e_adc_cmd_t type Execute a command for each interrupt request to use.	-
	p_args	e_adc_int_method_t type variable pointer Specifies interrupt setting (enabling and polling). Enabling interrupts: ADC_INT_ENABLE Polling: ADC_INT_POLLING	2.7.10
(7) Control function AD_CMD_AUTO_READ_y (y = NORMAL, BLOCK, COMPARE) Function details (NORMAL) in 2.4.23 Function details (BLOCK) in 2.5.24 Function details (COMPARE) in 2.5.25	cmd	Specify AD_CMD_AUTO_READ_y (y = NORMAL, BLOCK, or COMPARE).	-
	p_args	st_adc_dma_read_info_t type variable pointer As each element of st_adc_dma_read_info_t type variable structure, specify DMA transfer trigger, transfer source register, transfer destination RAM, transfer count, callback function, and transfer size.	2.8.8
(8) Start function	-	-	-
(9) Control function AD_CMD_GET_CMP_RESULT Function details in 2.5.16	cmd	AD_CMD_GET_CMP_RESULT	
	p_cmp	st_adc_cmp_result_t type variable pointer Specify address in which obtained comparison result is to be stored.	2.8.6
(10) Read function	sel_ch	e_adc_ssel_ch_t type Specify one analog input channel from which A/D conversion result is to be read.	2.7.5
	p_data	uint16_t type variable pointer Specify address in which A/D conversion result is to be stored.	-
(11) Close function	-	-	-

Figure 3-12 shows an example of compare operation (window A/B) in single-scan mode. In the shown example, the S14AD modes and compare functions are set as follows.

- Compare function window A/B: Enabled
- Compare function window A channel: AN000
- Compare function window B channel: AN001
- Compare function window A/B complex conditions: ADC140_WCMPM is output when the window A compare condition is met AND window B compare condition is met, whereas ADC140_WCMPUM is output in the other cases.

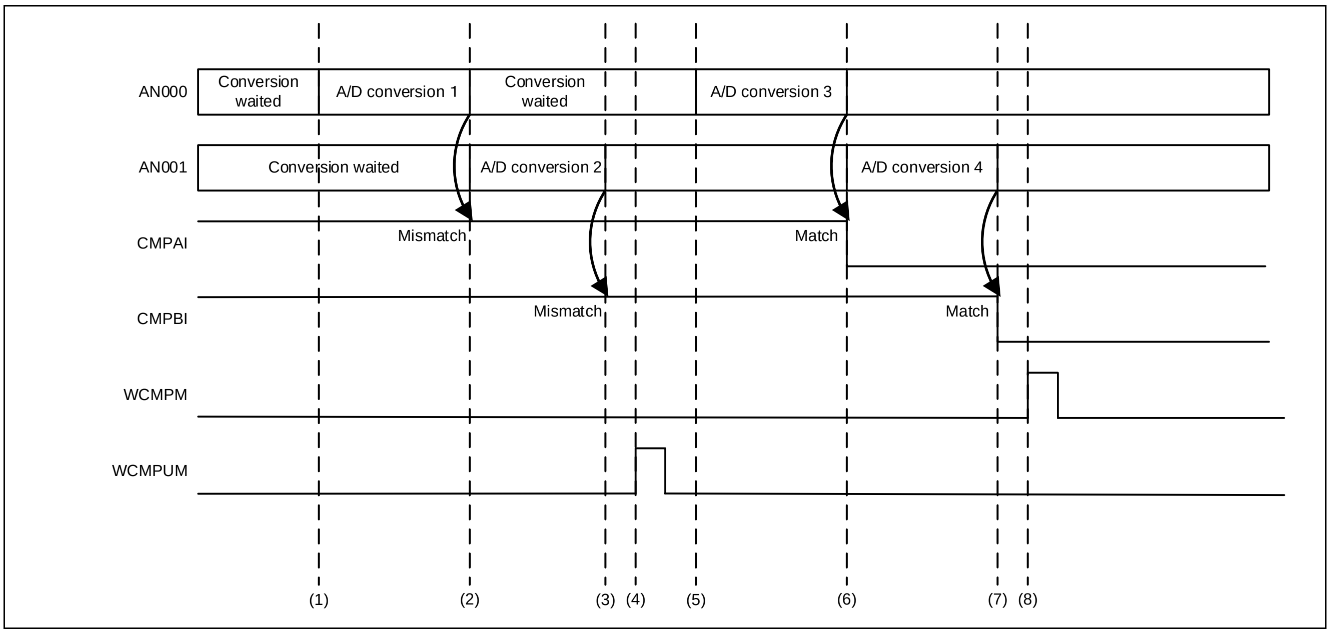


Figure 3-12 Compare Function Window A/B Operation

(1) A/D conversion is started by A/D conversion start trigger input.

(2) If the A/D conversion result of the AN000 channel after A/D conversion completion does not meet the condition of the compare function window A, the CMPAI interrupt request will not be generated.

(3) If the A/D conversion result of the AN001 channel after A/D conversion completion does not meet the condition of the compare function window B, the CMPBI interrupt request will not be generated.

(4) If the comparison result of the compare function window A/B does not meet the complex condition, the WCMPUM interrupt request will be generated.

Generation of the WCMPUM interrupt request is checked by the different manners depending on the interrupt function and auto-read function setting. The following describes how generation of the interrupt request is checked for each function.

[Interrupts]

If a callback function has been specified when executing the Open function and interrupts have been enabled by interrupt feature setting, the callback function will be executed at the end of A/D conversion.

[Polling]

If polling has been set by interrupt function setting, the window A/B compare match or mismatch event information can be obtained by executing the AD_CMD_GET_AD_STATE command of the Control function. If the event is detected, ADC_CONV_WCMP_DETECT will be stored as window A/B compare match or mismatch event information. For the application example of the AD_CMD_GET_AD_STATE command and the details of the obtainable information, see section 2.5.17, A/D Converter State Acquisition (AD_CMD_GET_AD_STATE).

[Auto-read]

If the auto-read function is used, the value of the predetermined register will be automatically transferred to the RAM upon generation of the interrupt request specified as the DMA activation source. A notice is not issued each time an interrupt request is generated. If a callback function has been specified when setting the auto-read function, the callback function will be executed upon completion of auto-reading for the specified times.

(5) A/D conversion is started by the second A/D conversion start trigger input.

(6) If the A/D conversion result meets the comparison condition of the compare function window A after completion of A/D conversion of AN000 channel, the callback function will be called. (Note)

(7) If the A/D conversion result meets the comparison condition of the compare function window B after completion of A/D conversion of AN001 channel, the callback function will be called. (Note 1)

(8) If the comparison result of the compare function window A/B meets the complex condition, the WCMPM interrupt request will be generated 1PCLKB after the A/D conversion scan end event (ADC140_ELC).

Generation of the WCMPUM interrupt request is checked by the different manners depending on the interrupt function and auto-read function setting. Generation of the interrupt request for each function is checked in the same manner as that described in 4.

Note. A callback function is called only when the following conditions are satisfied.

- A callback function has been specified with the Open function.
- An interrupt has been enabled when executing the compare feature setting command (AD_CMD_SET_WINDOWA/AD_CMD_SET_WINDOWB) of the Control function.

3.5 Configurations

For S14AD driver, configuration definitions that can be modified by the user are provided in the r_adc_cfg.h file.

3.5.1 Parameter Check

This enables or disables parameter checking for the S14AD driver.

Name: ADC_CFG_PARAM_CHECKING_ENABLE

Table 3-9 Settings of ADC_CFG_PARAM_CHECKING_ENABLE

Setting	Description
(0)	Disables parameter check. Does not detect the errors related to validity decision of the arguments described in the function specifications.
(1) (initial value)	Enables parameter check. Detects the errors related to validity validity of the arguments described in the function specifications.

3.5.2 ADI Interrupt Request Control

This selects which function is to be activated by the single-scan or group A scan end interrupt request (ADC140_ADI).

Name: S14AD_ADI_CONTROL

Table 3-10 Definitions of A/D Interrupt Request Control

Definition	Value	Description
S14AD_USED_INTERRUPT (initial value)	(0)	ADC140_ADI interrupt request is used for interrupts or polling.
S14AD_USED_DMAC0	(1<<0)	DMAC0 is activated by ADC140_ADI interrupt request.
S14AD_USED_DMAC1	(1<<1)	DMAC1 is activated by ADC140_ADI interrupt request
S14AD_USED_DMAC2	(1<<2)	DMAC2 is activated by ADC140_ADI interrupt request
S14AD_USED_DMAC3	(1<<3)	DMAC3 is activated by ADC140_ADI interrupt request
S14AD_USED_DTC	(1<<15)	DTC is activated by ADC140_ADI interrupt request

3.5.3 GBADI Interrupt Request Control

This selects which function is to be activated by the group B scan end interrupt request (ADC140_GBADI).

Name: S14AD_GBADI_CONTROL

Table 3-11 Definitions of A/D Interrupt Request Control

Definition	Value	Description
S14AD_USED_INTERRUPT (initial value)	(0)	ADC140_GBADI interrupt request is used for interrupts or polling.
S14AD_USED_DMAC0	(1<<0)	DMAC0 is activated by ADC140_GBADI interrupt request.
S14AD_USED_DMAC1	(1<<1)	DMAC1 is activated by ADC140_GBADI interrupt request.
S14AD_USED_DMAC2	(1<<2)	DMAC2 is activated by ADC140_GBADI interrupt request.
S14AD_USED_DMAC3	(1<<3)	DMAC3 is activated by ADC140_GBADI interrupt request.
S14AD_USED_DTC	(1<<15)	DTC is activated by ADC140_GBADI interrupt request.

3.5.4 GCADI Interrupt Request Control

This selects which function is to be activated by the group C scan end interrupt request (ADC140_GCADI).

Name: S14AD_GCADI_CONTROL

Table 3-12 Definitions of A/D Interrupt Request Control

Definition	Value	Description
S14AD_USED_INTERRUPT (initial value)	(0)	ADC140_GCADI interrupt request is used for interrupts or polling.
S14AD_USED_DMAC0	(1<<0)	DMAC0 is activated by ADC140_GCADI interrupt request.
S14AD_USED_DMAC1	(1<<1)	DMAC1 is activated by ADC140_GCADI interrupt request.
S14AD_USED_DMAC2	(1<<2)	DMAC2 is activated by ADC140_GCADI interrupt request.
S14AD_USED_DMAC3	(1<<3)	DMAC3 is activated by ADC140_GCADI interrupt request.
S14AD_USED_DTC	(1<<15)	DTC is activated by ADC140_GCADI interrupt request.

3.5.5 WCMPM Interrupt Request Control

This selects which function is to be activated by the condition match interrupt request (ADC140_WCMPM) of the window A/B compare function.

Name: S14AD_WCMPM_CONTROL

Table 3-13 Definitions of A/D Interrupt Request Control

Definition	Value	Description
S14AD_USED_INTERRUPT (initial value)	(0)	ADC140_WCMPM interrupt request is used for interrupts or polling.
S14AD_USED_DMAC0	(1<<0)	DMAC0 is activated by ADC140_WCMPM interrupt request.
S14AD_USED_DMAC1	(1<<1)	DMAC1 is activated by ADC140_WCMPM interrupt request.
S14AD_USED_DMAC2	(1<<2)	DMAC2 is activated by ADC140_WCMPM interrupt request.
S14AD_USED_DMAC3	(1<<3)	DMAC3 is activated by ADC140_WCMPM interrupt request.
S14AD_USED_DTC	(1<<15)	DTC is activated by ADC140_WCMPM interrupt request.

3.5.6 WCMPUM Interrupt Request Control

This selects which function is to be activated by the condition mismatch interrupt request (ADC140_WCMPUM) of the window A/B compare function.

Name: S14AD_WCMPUM_CONTROL

Table 3-14 Definitions of A/D Interrupt Request Control

Definition	Value	Description
S14AD_USED_INTERRUPT (initial value)	(0)	ADC140_WCMPUM interrupt request is used for interrupts or polling.
S14AD_USED_DMAC0	(1<<0)	DMAC0 is activated by ADC140_WCMPUM interrupt request.
S14AD_USED_DMAC1	(1<<1)	DMAC1 is activated by ADC140_WCMPUM interrupt request.
S14AD_USED_DMAC2	(1<<2)	DMAC2 is activated by ADC140_WCMPUM interrupt request.
S14AD_USED_DMAC3	(1<<3)	DMAC3 is activated by ADC140_WCMPUM interrupt request.
S14AD_USED_DTC	(1<<15)	DTC is activated by ADC140_WCMPUM interrupt request.

3.5.7 ADI Interrupt Priority Level

This sets the priority level of the ADI interrupt.

Name: S14AD_ADI_PRIORITY

Table 3-15 Settings of S14AD_ADI_PRIORITY

Setting	Description
0	Sets the interrupt priority level to 0. (Highest)
1	Sets the interrupt priority level to 1.
2	Sets the interrupt priority level to 2.
3 (initial value)	Sets the interrupt priority level to 3.

3.5.8 GBADI Interrupt Priority Level

This sets the priority level of the GBADI interrupt.

Name: S14AD_GBADI_PRIORITY

Table 3-16 Settings of S14AD_GBADI_PRIORITY

Setting	Description
0	Sets the interrupt priority level to 0. (Highest)
1	Sets the interrupt priority level to 1.
2	Sets the interrupt priority level to 2.
3 (initial value)	Sets the interrupt priority level to 3.

3.5.9 GCADI Interrupt Priority Level

This sets the priority level of the GCADI interrupt.

Name: S14AD_GCADI_PRIORITY

Table 3-17 Settings of S14AD_GCADI_PRIORITY

Setting	Description
0	Sets the interrupt priority level to 0. (Highest)
1	Sets the interrupt priority level to 1.
2	Sets the interrupt priority level to 2.
3 (initial value)	Sets the interrupt priority level to 3.

3.5.10 CMPAI Interrupt Priority Level

This sets the priority level of the CMPAI interrupt.

Name: S14AD_CMPAI_PRIORITY

Table 3-18 Settings of S14AD_CMPAI_PRIORITY

Setting	Description
0	Sets the interrupt priority level to 0. (Highest)
1	Sets the interrupt priority level to 1.
2	Sets the interrupt priority level to 2.
3 (initial value)	Sets the interrupt priority level to 3.

3.5.11 CMPBI Interrupt Priority Level

This sets the priority level of the CMPBI interrupt.

Name: S14AD_CMPBI_PRIORITY

Table 3-19 Settings of S14AD_CMPBI_PRIORITY

Setting	Description
0	Sets the interrupt priority level to 0. (Highest)
1	Sets the interrupt priority level to 1.
2	Sets the interrupt priority level to 2.
3 (initial value)	Sets the interrupt priority level to 3.

3.5.12 WCMPM Interrupt Priority Level

This sets the priority level of the WCMPM interrupt.

Name: S14AD_WCMPM_PRIORITY

Table 3-20 Settings of S14AD_WCMPM_PRIORITY

Setting	Description
0	Sets the interrupt priority level to 0. (Highest)
1	Sets the interrupt priority level to 1.
2	Sets the interrupt priority level to 2.
3 (initial value)	Sets the interrupt priority level to 3.

3.5.13 WCMPUM Interrupt Priority Level

This sets the priority level of the WCMPUM interrupt.

Name: S14AD_WCMPUM_PRIORITY

Table 3-21 Settings of S14AD_WCMPUM_PRIORITY

Setting	Description
0	Sets the interrupt priority level to 0. (Highest)
1	Sets the interrupt priority level to 1.
2	Sets the interrupt priority level to 2.
3 (initial value)	Sets the interrupt priority level to 3.

3.5.14 CMPAI Snooze Mode

This sets whether to use compare function window A only in snooze mode in low power consumption mode.

Name: ADC_CMPAI_SNOOZE_USE

Table 3-22 Settings of ADC_CMPAI_SNOOZE_USE

Setting	Description
0 (initial value)	Uses compare function window A by the CMPAI interrupt (performs CMPAI interrupt setting). (Note 1)
1	Uses compare function window A only in snooze mode in low power consumption mode (skips CMPAI interrupt setting). (Note 2)

Note 1. Set 0 to use compare function window A in snooze mode in low power consumption mode and to perform CMPAI interrupt setting. The CMPAI interrupt cannot be used when 1 is set.

Note 2. When checking A/D conversion status (CMPAI) by S/W polling, set "1".

3.5.15 CMPBI Snooze Mode

This sets whether to use compare function window B only in snooze mode in low power consumption mode.

Name: ADC_CMPBI_SNOOZE_USE

Table 3-23 Settings of ADC_CMPBI_SNOOZE_USE

Setting	Description
0 (initial value)	Uses compare function window B by the CMPBI interrupt (performs CMPBI interrupt setting). (Note 1)
1	Uses compare function window B only in snooze mode in low power consumption mode (skips CMPBI interrupt setting). (Note 2)

Note 1. Set 0 to use compare function window B in snooze mode in low power consumption mode and to perform CMPBI interrupt setting. The CMPBI interrupt cannot be used when 1 is set.

Note 2. When checking A/D conversion status (CMPBI) by S/W polling, set "1".

3.5.16 Function Allocation to RAM

This initializes the settings for executing specific functions of the S14AD driver in RAM.

This configuration definition for setting function allocation to RAM has function-specific definitions.

Name: S14AD_CFG_xxx

A function name xxx should be written in all capital letters.

Example: R_ADC_Open function → S14AD_CFG_R_ADC_OPEN

Table 3-24 Settings of ADC_CFG_SECTION_xxx

Setting	Description
SYSTEM_SECTION_CODE	Does not allocate the function to RAM.
SYSTEM_SECTION_RAM_FUNC	Allocates the function to RAM.

Table 3-25 Initial State of Function Allocation to RAM

No.	Function Name	Allocation to RAM
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 interrupt handling process)	√
10	adc_gbadi_isr (GBADI interrupt handling process)	√
11	adc_gcadi_isr (GCADI interrupt handling process)	√
12	adc_cmpai_isr (CMPAI interrupt handling process)	√
13	adc_cmpbi_isr (CMPBI interrupt handling process)	√
14	adc_wcmpm_isr (WCMPM interrupt handling process)	√
15	adc_wcmpum_isr (WCMPUM interrupt handling process)	√

4. Detailed Information of Driver

This chapter describes the detailed specifications implementing the capabilities of this driver.

4.1 Function Specifications

The specifications and processing flow of each function of the S14AD driver are described in this section.

In the flow of processing, some decision methods such as conditional branching are omitted, and consequently the flowcharts do not exactly show the actual processing.

4.1.1 R_ADC_Open Function

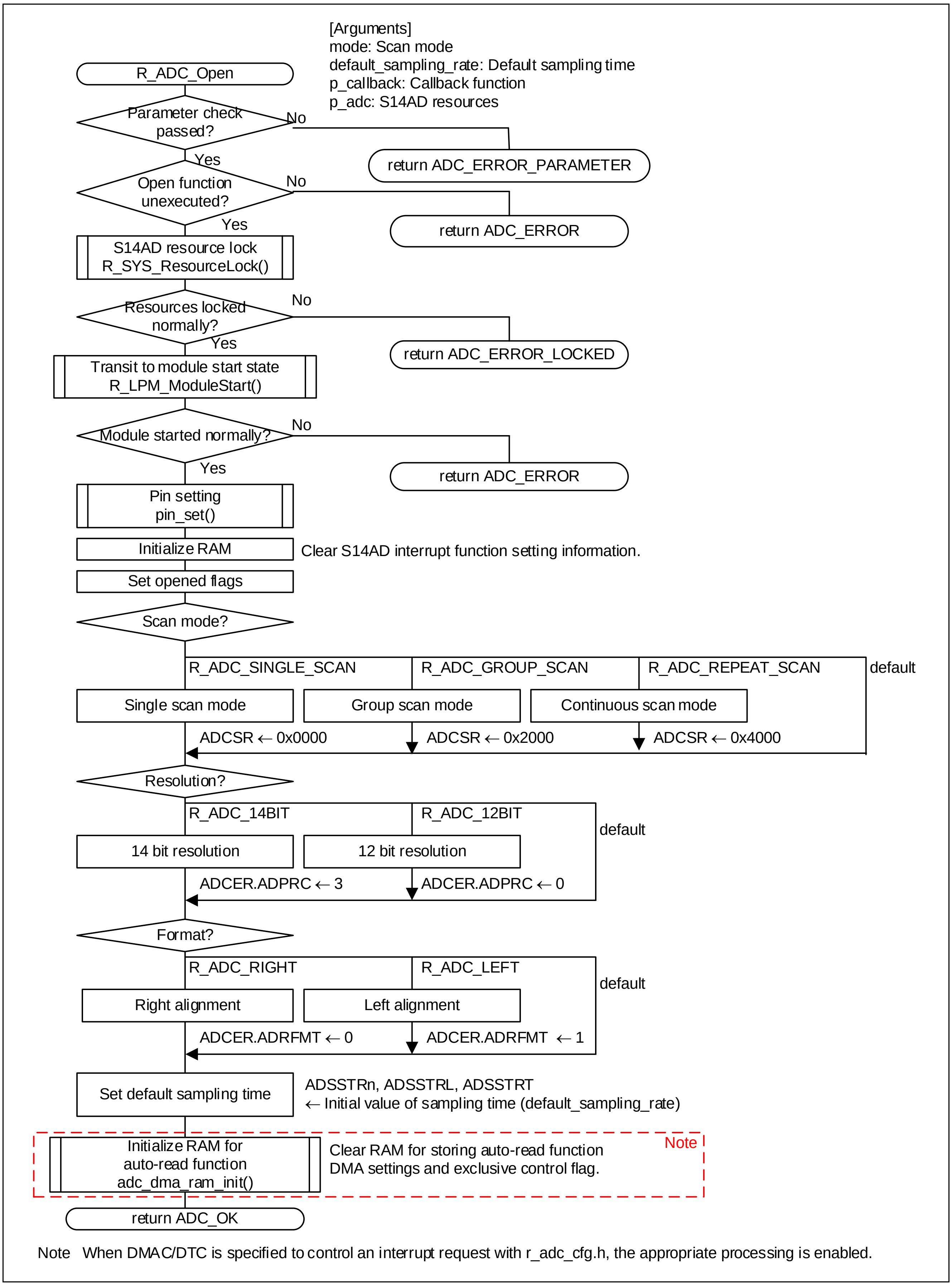
Table 4-1 R_ADC_Open Function Specifications

Format	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)
Description	Initializes the S14AD driver (initializes RAM, makes register settings, makes pin settings, and cancels the module stop state). Also sets scan mode, A/D conversion accuracy, A/D data register format, and default sampling time.
Argument	uint32_t mode: Scan mode, A/D conversion accuracy, and A/D data register format are specified in combination.
	uint8_t default_sampling_rate: default sampling time A default sampling time of 2 to 255 (Note 1) is specified (initial value of ADSSTRn (Note 2)).
	adc_cd_event_t const p_callback: callback function Specifies the callback function at the time of S14AD event occurrence. When NULL is specified, a callback function is not executed at the time of S14AD event occurrence.
	st_adc_resources_t const * const p_adc: Resources of S14AD Specify the resources of the S14AD to initialize.
Return value	ADC_OK S14AD initialization completed
	ADC_ERROR_PARAMETER Parameter error If one of the following conditions is detected, a parameter error will occur. • If a value other than those stipulated is specified for scan mode. • If a value other than those stipulated is specified for A/D conversion accuracy. • If a value other than those stipulated is specified for A/D data register format. • If 0 or 1 is set for the default sampling time.
	ADC_ERROR S14AD initialization failed If one of the following conditions is detected, a parameter error will occur. • If the Open function is already being executed • If module stop state has already been cancelled.

Remarks	When this function is accessed, specifying the S14AD resources is not required.
	
	[Example of calling function from instance]
	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);
	}

Note 1. When the ADSCLKCR.SCLKEN bit is 0 (sub clock mode disabled), specify 5 to 255. When the ADSCLKCR.SCLKEN bit is 1 (sub clock mode enabled), specify 2 to 8.

Note 2. 256KB Group : n = 0 to 7,L,T 1500KB Group : n = 0 to 6,L,T



Note When DMAC/DTC is specified to control an interrupt request with r_adc_cfg.h, the appropriate processing is enabled.

Figure 4-1 R_ADC_Open Function Processing Flow

4.1.2 R_ADC_Close Function

Table 4-2 R_ADC_Close Function Specifications

Format	e_adc_err_t R_ADC_Close(st_adc_resources_t * const p_adc)
Description	Releases the S14AD driver.
Argument	st_adc_resources_t *p_adc: Resources of S14AD Specifies the resources of the S14AD to be released.
Return value	ADC_OK S14AD released normally

Remarks	When this function is accessed, specifying the S14AD resources is not required.
	
	[Example of calling function from instance]
	// S14AD driver instance
	extern DRIVER_S14AD Driver_S14AD;
	DRIVER_S14AD *adcDev = &Driver_S14AD;
	
	main()
	{
	    adcDev->Close();
	}

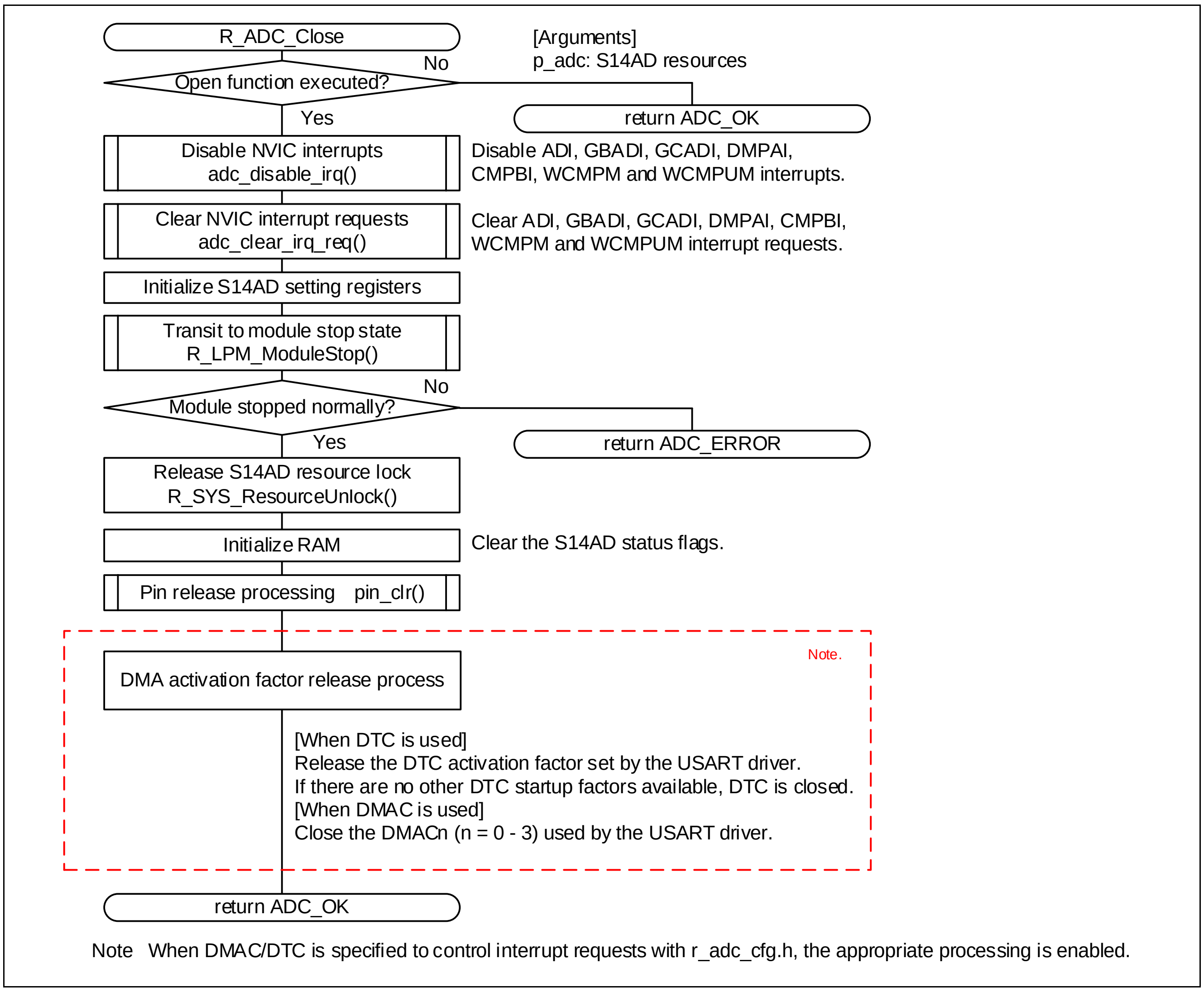

R_ADC_Close
[Arguments]
p_adc: S14AD resources
Open function executed?
No
Yes
return ADC_OK
Disable NVIC interrupts
adc_disable_irq()
Disable ADI, GBADI, GCADI, DMPAI, CMPBI, WCMPM and WCMPUM interrupts.
Clear NVIC interrupt requests
adc_clear_irq_req()
Clear ADI, GBADI, GCADI, DMPAI, CMPBI, WCMPM and WCMPUM interrupt requests.
Initialize S14AD setting registers
Transit to module stop state
R_LPM_ModuleStop()
Module stopped normally?
No
Yes
return ADC_ERROR
Release S14AD resource lock
R_SYS_ResourceUnlock()
Initialize RAM
Clear the S14AD status flags.
Pin release processing pin_clr()
Note.
DMA activation factor release process
[When DTC is used]
Release the DTC activation factor set by the USART driver.
If there are no other DTC startup factors available, DTC is closed.
[When DMAC is used]
Close the DMACn (n = 0 - 3) used by the USART driver.
return ADC_OK
Note When DMAC/DTC is specified to control interrupt requests with r_adc_cfg.h, the appropriate processing is enabled.

Figure 4-2 R_ADC_Close Function Processing Flow

4.1.3 R_ADC_ScanSet Function

Table 4-3 R_ADC_ScanSet Function Specifications

Format	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)	
Description	Sets the target channel for A/D conversion and A/D conversion start trigger for each group.	
Argument	e_adc_group_t group: Specifies the A/D conversion group to be set. Set one of the following. ADC_GROUP_A: Group A ADC_GROUP_B: Group B ADC_GROUP_C: Group C	
	st_adc_pins_t pins: Specifies the target channel for A/D conversion.	
	e_adc_triger_t triger: Specifies A/D conversion start trigger. Set one of the following. ADC_TRIGER_SOFT: Software trigger ADC_TRIGER_TMR: Compare match of the TCORA register and TCNT counter ADC_TRIGER_ELC: ELC_S14AD ADC_TRIGER_ADTRG: ADTRG0 input ADC_TRIGER_LOW_PRIORITY_CONT_SCAN: Trigger source not selected	
	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.	
Return value	ADC_OK	A/D conversion channel and A/D conversion start trigger set normally
	ADC_ERROR	A/D conversion channel and A/D conversion start trigger setting failed. If this function is executed with S14AD uninitialized, A/D conversion channel and A/D conversion start trigger setting will fail.
	ADC_ERROR_PARAMETER	Parameter error If one of the following conditions is detected, a parameter error will occur. • If any pin is not specified for the A/D conversion channel • If a non-existent trigger setting is specified • If a software trigger is specified for the group B and group C A/D conversion start trigger • If an external trigger is specified for the group B and group C A/D conversion start trigger
	ADC_ERROR_MODE	Mode error If one of the following conditions is detected, a mode error will occur. • If a software trigger is specified for group A in group scan mode • If group B and group C are specified in other than group scan mode

Remarks	When this function is accessed, specifying the S14AD resources is not required.
	
	[Example of calling function from instance]
	`// S14AD driver instance`
	`extern DRIVER_S14AD Driver_S14AD;`
	`DRIVER_S14AD *adcDev = &Driver_S14AD;`
	
	`main()`
	` {`
	`      st_adc_pins_t scanset_pin;    /* Variable for setting A/D conversion channel */`
	`      scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01 | ADC_MSEL_AN03;`
	`                                   /* AN000, AN001 and AN03 are used for A/D conversion */`
	`      scanset_pin.sensor      = ADC_SENSOR_NOTUSE;    /* No temperature sensor is used`
	`for A/D conversion */`
	`      adcDev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);`
	`}`	

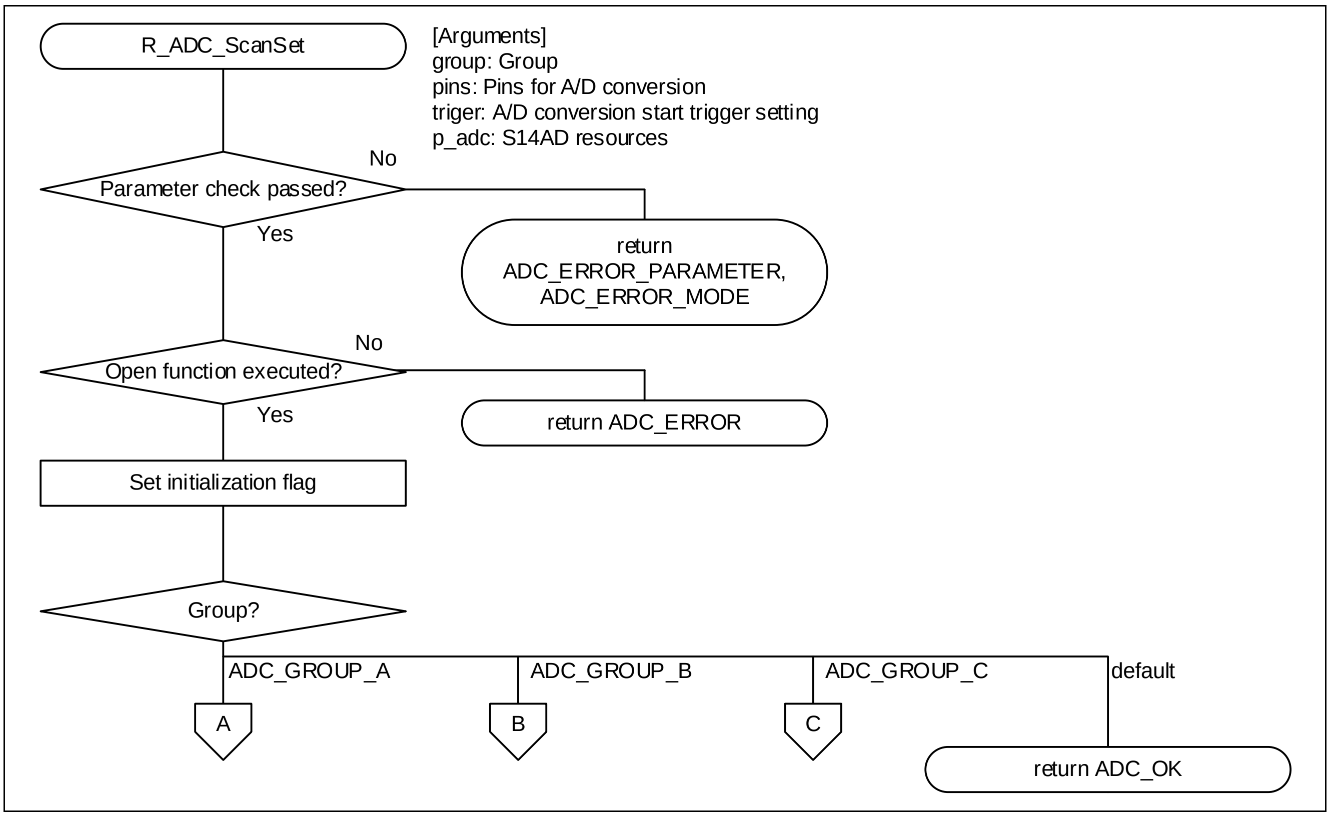


Figure 4-3 R_ADC_ScanSet Function Processing Flow (1/4)

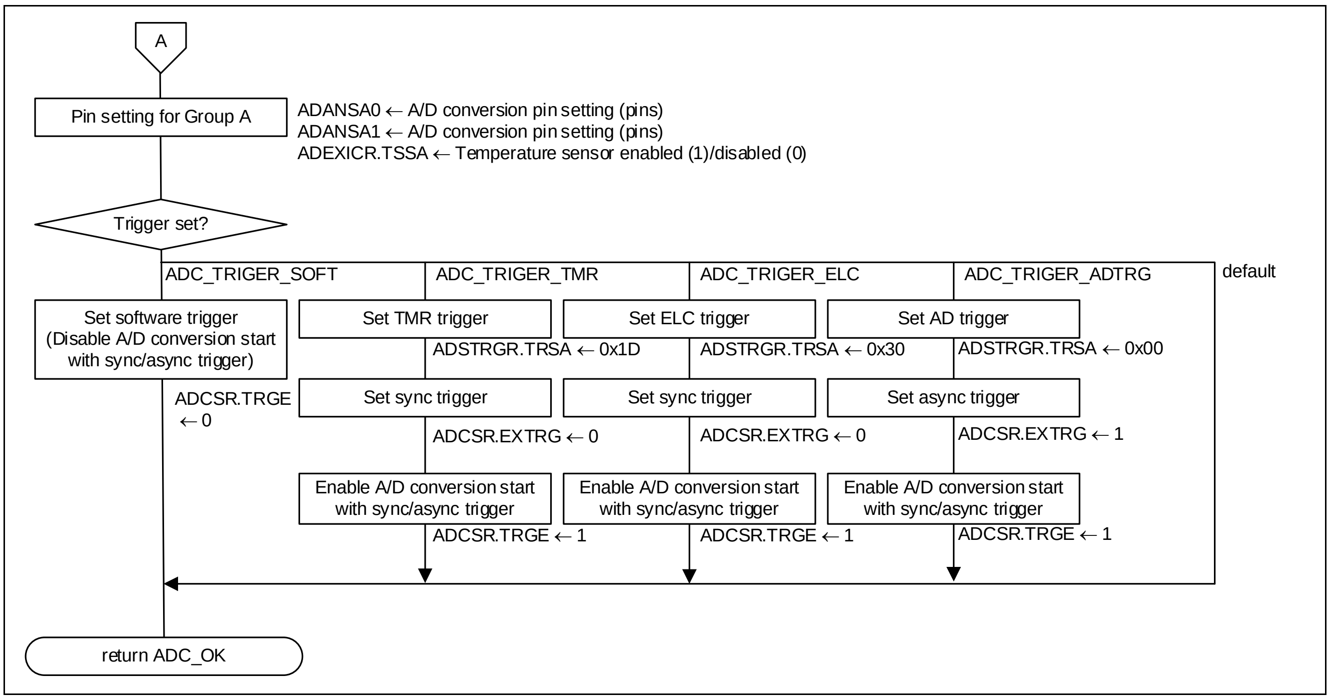

A
Pin setting for Group A
ADANSA0 ← A/D conversion pin setting (pins)
ADANSA1 ← A/D conversion pin setting (pins)
ADEXICR.TSSA ← Temperature sensor enabled (1)/disabled (0)
Trigger set?
ADC_TRIGER_SOFT
ADC_TRIGER_TMR
ADC_TRIGER_ELC
ADC_TRIGER_ADTRG
default
Set software trigger (Disable A/D conversion start with sync/async trigger)
ADCSR.TRGE ← 0
Set TMR trigger
ADSTRGR.TRSA ← 0x1D
Set sync trigger
ADCSR.EXTRG ← 0
Enable A/D conversion start with sync/async trigger
ADCSR.TRGE ← 1
Set ELC trigger
ADSTRGR.TRSA ← 0x30
Set sync trigger
ADCSR.EXTRG ← 0
Enable A/D conversion start with sync/async trigger
ADCSR.TRGE ← 1
Set AD trigger
ADSTRGR.TRSA ← 0x00
Set async trigger
ADCSR.EXTRG ← 1
Enable A/D conversion start with sync/async trigger
ADCSR.TRGE ← 1
return ADC_OK

Figure 4-3 R_ADC_ScanSet Function Processing Flow (2/4)

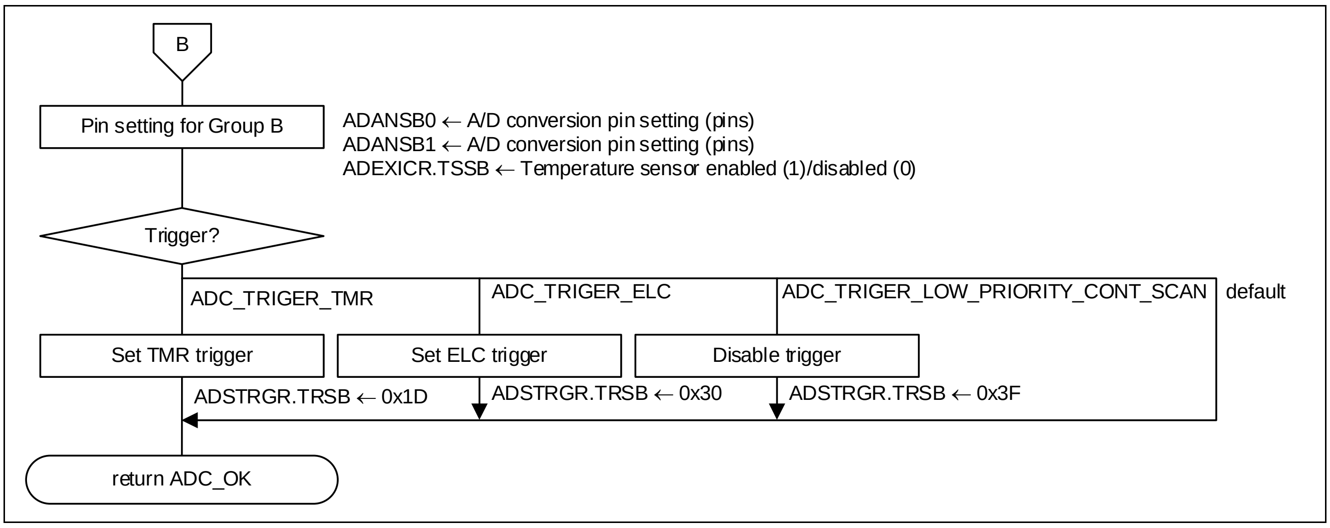


Figure 4-4 R_ADC_ScanSet Function Processing Flow (3/4)

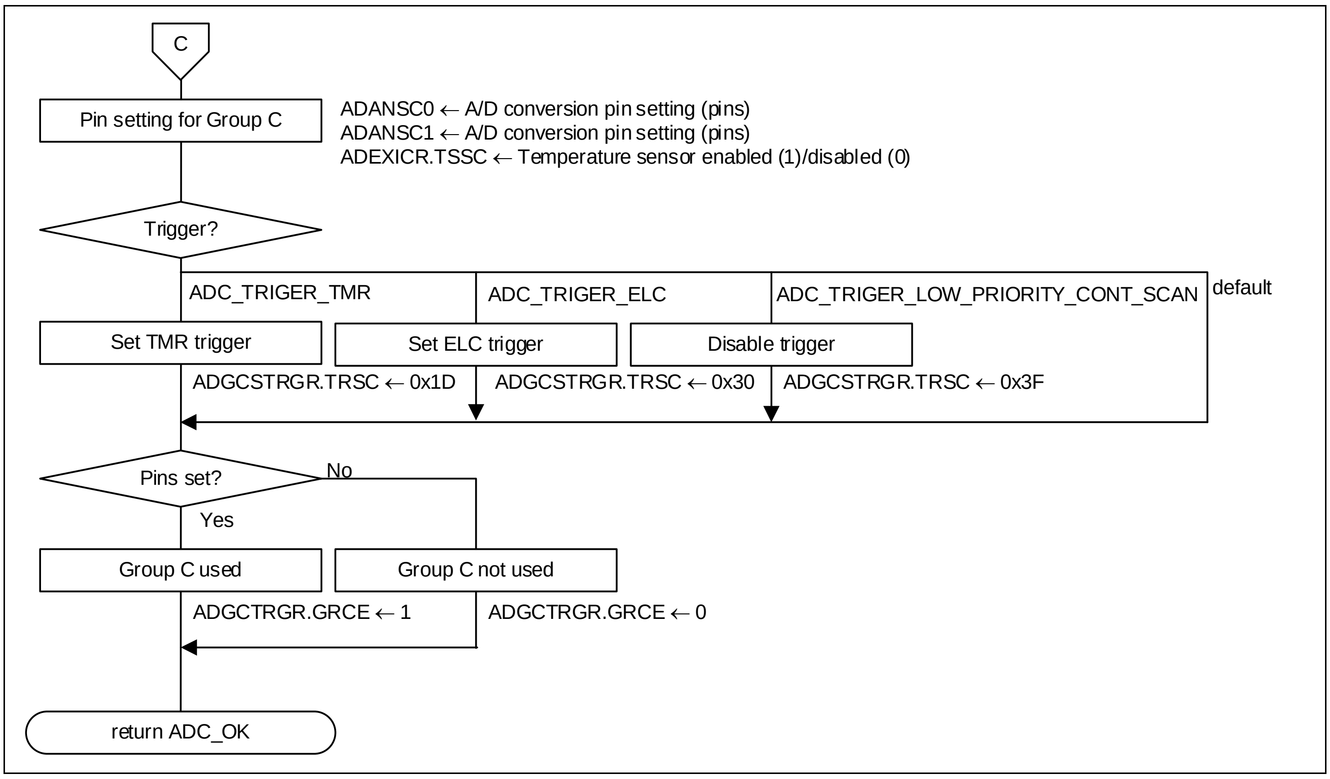


Figure 4-5 R_ADC_ScanSet Function Processing Flow (4/4)

4.1.4 R_ADC_Start Function

Table 4-4 R_ADC_Start Function Specifications

Format	e_adc_err_t R_ADC_Start(st_adc_resources_t * const p_adc)	
Description	Starts A/D conversion using a software trigger	
Argument	st_adc_resources_t *p_adc: Resources of S14AD Specifies the S14AD resource to transmit	
Return value	ADC_OK	A/D conversion started normally
	ADC_ERROR If this function is executed with S14AD uninitialized, A/D conversion start will fail.	A/D conversion start failed

Remarks	When this function is accessed, specifying the S14AD resources is not required.
	
	[Example of calling function from instance]
	// S14AD driver instance
	extern DRIVER_S14AD Driver_S14AD;
	DRIVER_S14AD *adcDev = &Driver_S14AD;
	
	main()
	{
	    adcDev->Start();
	}	

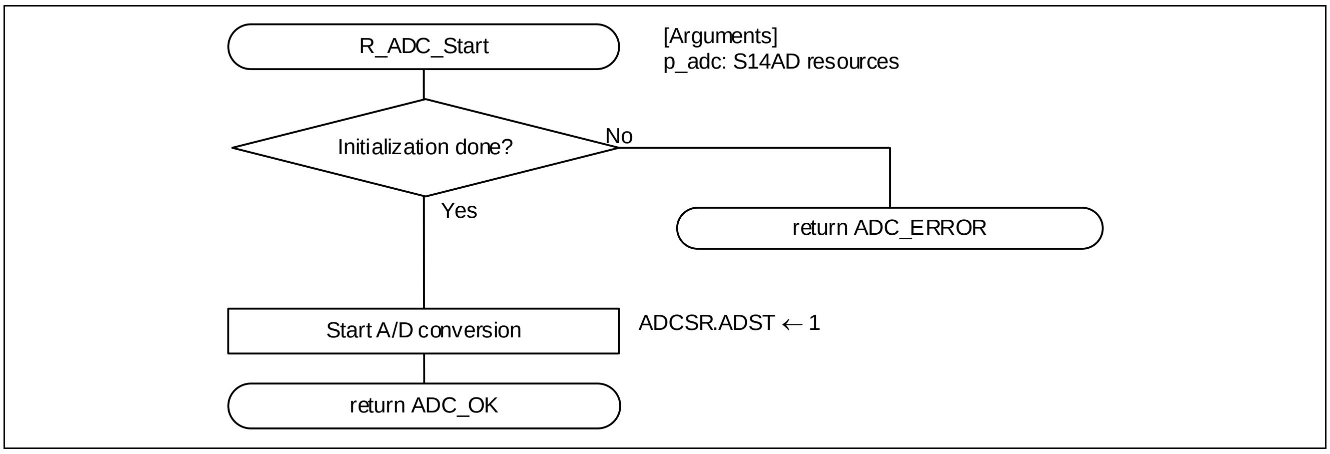


Figure 4-6 R_ADC_Start Function Processing Flow

4.1.5 R_ADC_Stop Function

Table 4-5 R_ADC_Stop Function Specifications

Format	e_adc_err_t R_ADC_Stop(st_adc_resources_t * const p_adc)	
Description	Stops A/D conversion using a software trigger	
Argument	st_adc_resources_t *p_adc: Resources of S14AD Specifies the resources of S14AD.	
Return value	ADC_OK	A/D conversion stopped normally
	ADC_ERROR If this function is executed with S14AD uninitialized, A/D conversion stop will fail.	A/D conversion stop failed

Remarks	When this function is accessed, specifying the S14AD resources is not required
	
	[Example of calling function from instance]
	// S14AD driver instance
	extern DRIVER_S14AD Driver_S14AD;
	DRIVER_S14AD *adcDev = &Driver_S14AD;
	
	main()
	{
	    adcDev->Stop();
	}	

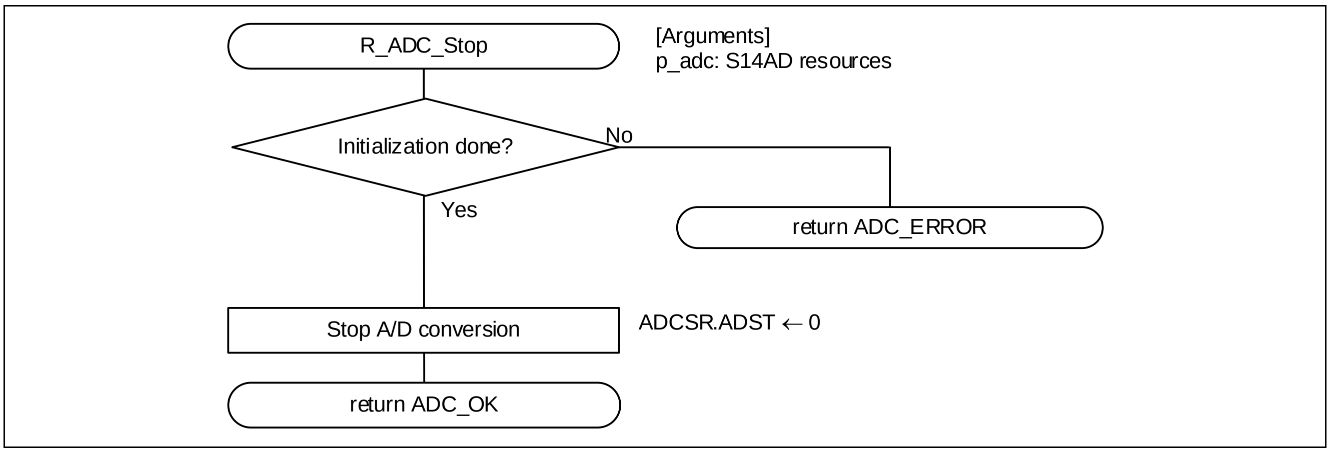


Figure 4-7 R_ADC_Stop Function Processing Flow

4.1.6 R_ADC_Control Function

Table 4-6 R_ADC_Control Function Specifications

Format	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)	
Description	Sets S14AD features	
Argument	e_adc_cmd_t const cmd: Specifies the control command to set a S14AD feature	
	void const * const p_args: Specifies arguments for the control command	
	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD	
Return value	ADC_OK	Feature setting completed
	ADC_ERROR If one of the following conditions is detected, feature setting will fail. • If executed in a state in which the Open function has not been executed • If the return value from each feature setting function is ADC_ERROR	Feature setting failed
	ADC_ERROR_BUSY When a control command that cannot set a feature during A/D conversion is specified, feature setting failure due to a busy state results.	Feature setting failure due to a busy state
	ADC_ERROR_SYSTEM_SETTING When a system setting is incorrect, a system setting error results.	System setting error
	ADC_ERROR_MODE When a mode setting is incorrect, a mode error results.	Mode error
	ADC_ERROR_PARAMETER When an argument is incorrect, a parameter error results.	Parameter error

Remarks	When this function is accessed, specifying the S14AD resources is not required
	
	[Example of calling function from instance]
	`// 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);`
	`}`	

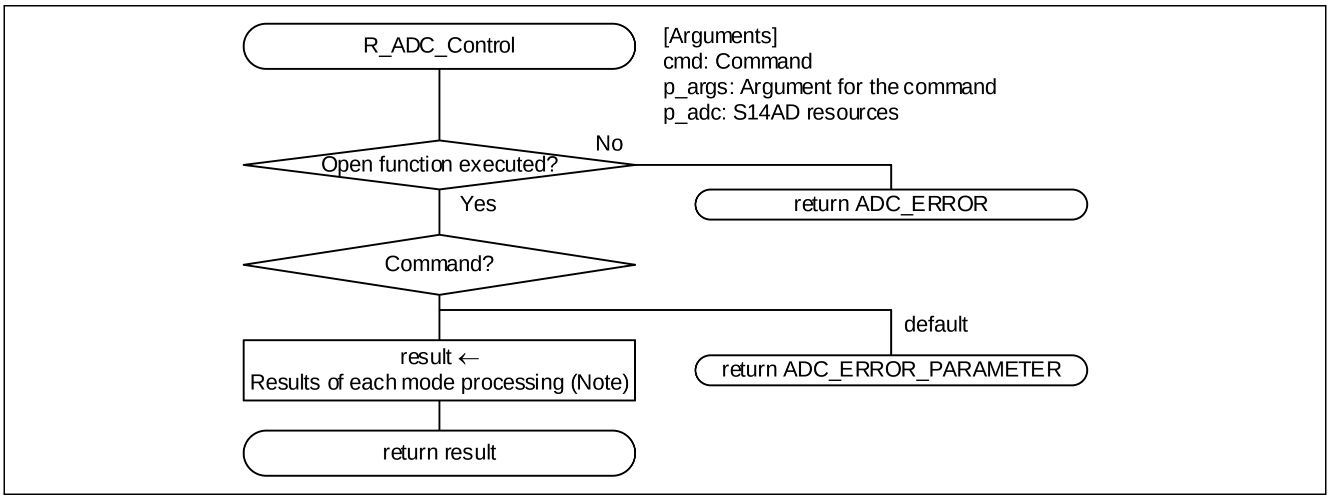


Figure 4-8 R_ADC_Control Function Processing Flow

Note “result” stores the return value from the function for each command. The functions called and return values for each command are listed in Table 4-7 and Table 4-8.

Table 4-7 Functions Called and Return Values for Each Control Command (1/2)

Command (cmd)	Functions Executed	Return Values
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 (Note 1)	adc_cmd_sampling	ADC_ERROR_PARAMETER
		ADC_OK
AD_SST_AN016_AN017_AN020_TO_AN028 (Note 2)	adc_cmd_sampling	ADC_ERROR_PARAMETER
		ADC_OK
AD_CMD_SET_SAMPLING_AN016_AN017_AN020_AN021_VSC_VCC (Note 1)	adc_cmd_sampling	ADC_ERROR_PARAMETER
		ADC_OK
AD_CMD_SET_SAMPLING_TEMP	adc_cmd_sampling	ADC_ERROR_PARAMETER
		ADC_OK
AD_CMD_SET_ADNDIS	adc_cmd_charge	ADC_ERROR_PARAMETER
		ADC_ERROR_MODE
		ADC_ERROR_BUSY
		ADC_OK

Note 1. 256KB Only

Note 2. 1500KB Only

Table 4-8 Functions Called and Return Values for Each Control Command (2/2)

Command (cmd)	Functions Executed	Return Values
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 Function

Table 4-9 R_ADC_Read Function Specifications

Format	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)
Description	Reads out the A/D conversion result for the specified channel.
Argument	e_adc_ssel_ch_t sel_ch: Channel for readout One of the following is specified DC_SSEL_Ann: Reads out the A/D data register (ADDRn) for the specified channel (1500KB Group : nn = 00 to 06,16,17,20 to 28 256KB Group : nn = 00 to 07,16,17,20 to 21)l • ADC_SSEL_TEMP: Reads out the temperature sensor data register (ADTSDR) • ADC_SSEL_VSC_VCC: Reads out the A/D VSC_VCC Voltage Voltage Data Register (ADVSCDR)(Note) • ADC_SSEL_DBL: Reads out the A/D data duplexing register (ADDBLDR) • ADC_SSEL_DIAG: Reads A/D self-diagnosis data register (ADRD)
	uint16_t * const p_data: Read data storage address Specifies the address of a variable for storing the read data
	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.
Return value	ADC_OK A/D conversion result read normally
	ADC_ERROR_PARAMETER Parameter error If one of the following conditions is detected, a parameter error will occur • If the read data storage address is NULL • If a non-existent A/D conversion channel is specified

Remarks	When this function is accessed, specifying the S14AD resources is not required
	
	[Example of calling function from instance]
	// 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);
	}

Note. 256KB Only

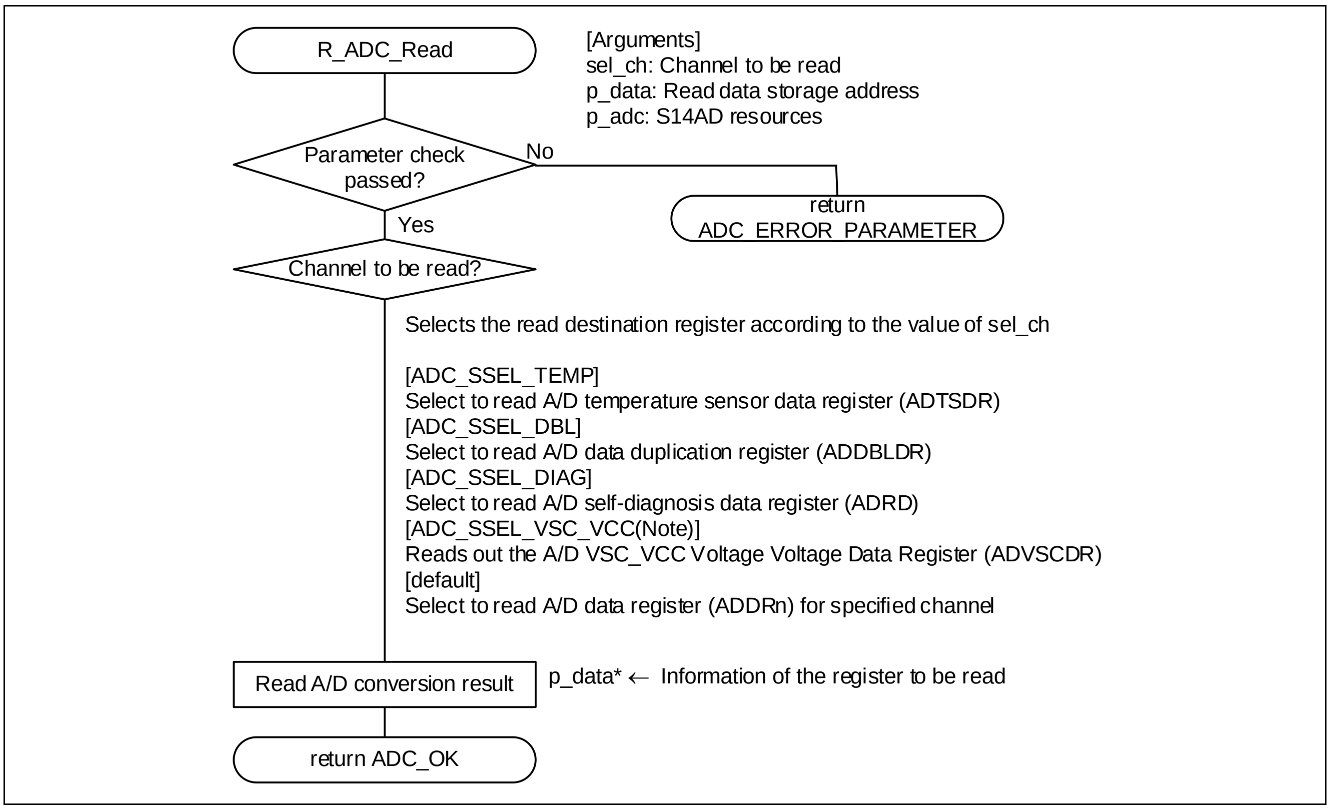

R_ADC_Read
[Arguments]
sel_ch: Channel to be read
p_data: Read data storage address
p_adc: S14AD resources
Parameter check passed?
No
return ADC_ERROR_PARAMETER
Yes
Channel to be read?
Selects the read destination register according to the value of sel_ch
[ADC_SSEL_TEMP]
Select to read A/D temperature sensor data register (ADTSDR)
[ADC_SSEL_DBL]
Select to read A/D data duplication register (ADDBLDR)
[ADC_SSEL_DIAG]
Select to read A/D self-diagnosis data register (ADRD)
[ADC_SSEL_VSC_VCC(Note)]
Reads out the A/D VSC_VCC Voltage Voltage Data Register (ADVSCDR)
[default]
Select to read A/D data register (ADDRn) for specified channel
Read A/D conversion result
p_data* ← Information of the register to be read
return ADC_OK

Figure 4-9 R_ADC_Read Function Processing Flow

Note. 256KB Only

4.1.8 R_ADC_GetVersion Function

Table 4-10 R_ADC_GetVersion Function Specifications

Format	uint32_t R_ADC_GetVersion(void)
Description	Acquires S14AD driver version
Argument	None
Return value	The version of the S14AD driver

Remarks	When this function is accessed, specifying the S14AD resources is not required
	
	[Example of calling function from instance]
	`// S14AD driver instance`
	`extern DRIVER_S14AD Driver_S14AD;`
	`DRIVER_S14AD *adcDev = &Driver_S14AD;`
	
	`main()`
	`{`
	`    DRIVER_VERSION version;`
	`    version = adcDev->GetVersion();`
	
	`}`

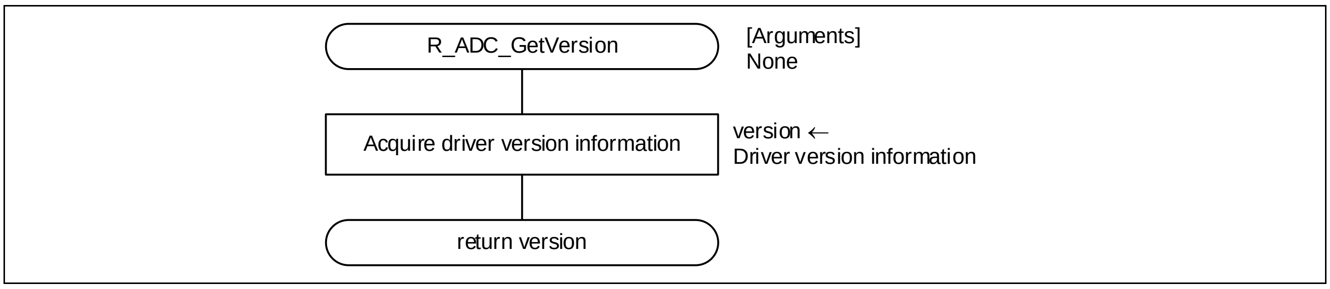

R_ADC_GetVersion
[Arguments]
None
Acquire driver version information
version ←
Driver version information
return version

Figure 4-10 R_ADC_GetVersion Function Processing Flow

4.1.9 adc_cmd_add_mode Function

Table 4-11 adc_cmd_add_mode Function Specifications

Format	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)
Description	Specifies addition/average mode settings.
Argument	st_adc_add_mode_t const * const p_add_m: Addition/average mode setting value
	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.
Return value	ADC_OK Addition/average mode set normally.
	ADC_ERROR_MODE Mode error If one of the following conditions is detected, a mode error will occur • If the A/D conversion accuracy is 14 bits, but 16 conversions were specified • If this function is executed in self-diagnosis mode
	ADC_ERROR_PARAMETER Parameter error If one of the following conditions is detected, a parameter error will occur • If a value other than those stipulated is specified for an addition/average mode setting • If a channel that is not an addition/average mode channel is specified.
	ADC_ERROR_BUSY The setting failed because of busy state. If this function is executed during A/D conversion, setting failure due to a busy state results.
Remarks	-

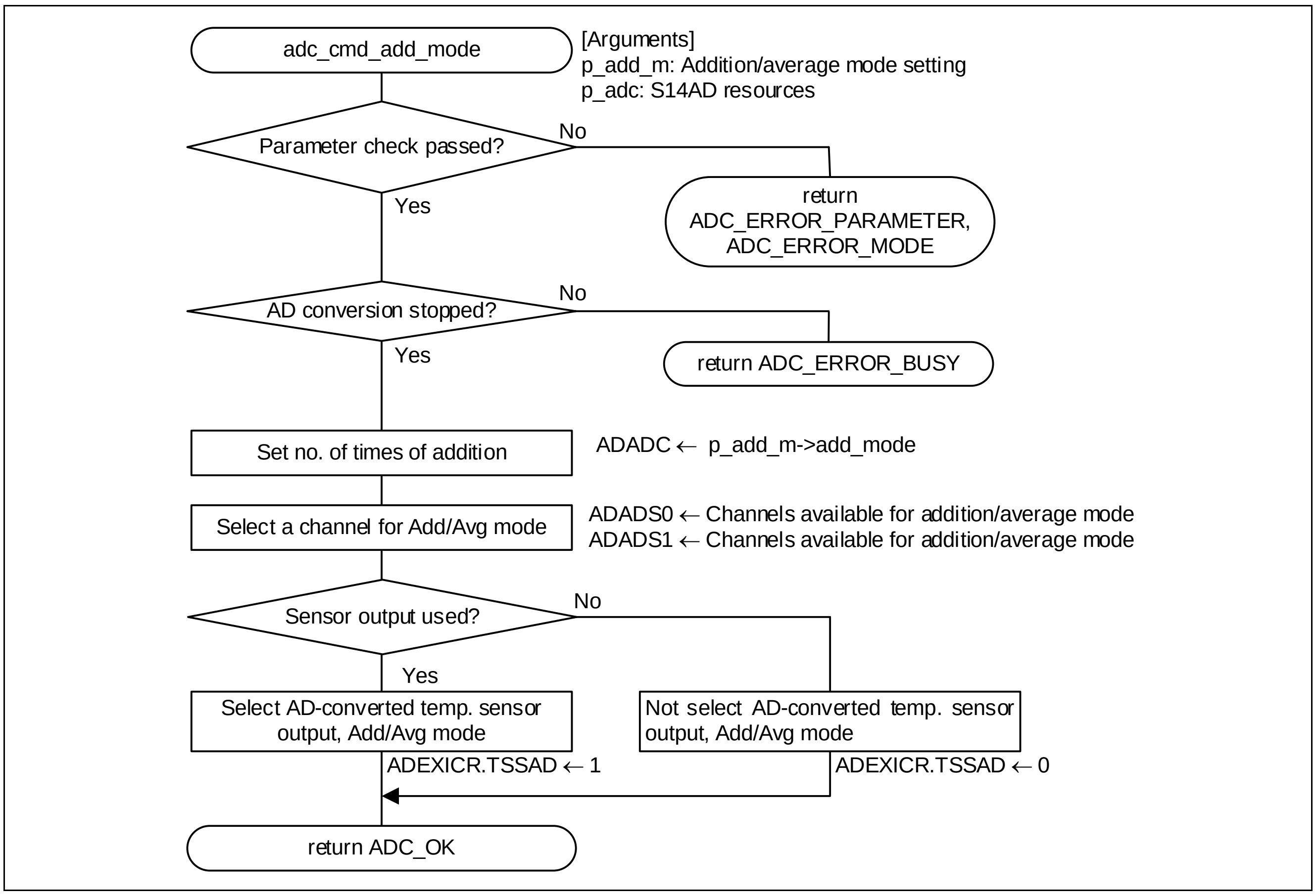

adc_cmd_add_mode
[Arguments]
p_add_m: Addition/average mode setting
p_adc: S14AD resources
Parameter check passed?
No
Yes
return ADC_ERROR_PARAMETER, ADC_ERROR_MODE
AD conversion stopped?
No
Yes
return ADC_ERROR_BUSY
Set no. of times of addition
ADADC ← p_add_m->add_mode
Select a channel for Add/Avg mode
ADADS0 ← Channels available for addition/average mode
ADADS1 ← Channels available for addition/average mode
Sensor output used?
No
Yes
Select AD-converted temp. sensor output, Add/Avg mode
Not select AD-converted temp. sensor output, Add/Avg mode
ADEXICR.TSSAD ← 1
ADEXICR.TSSAD ← 0
return ADC_OK

Figure 4-11 adc_cmd_add_mode Function Processing Flow

4.1.10 adc_cmd_dbltrg Function

Table 4-12 adc_cmd_dbltrg Function Specifications

Format	static e_adc_err_t adc_cmd_dbltrg(int8_t const * const p_dbltrg_ch, st_adc_resources_t * const p_adc)	
Description	Specifies double-trigger mode settings.	
Argument	int8_t const * const p_dbltrg_ch: Specifies the channel for A/D conversion data duplication.	
	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.	
Return value	ADC_OK	Double-trigger mode set normally
	ADC_ERROR_PARAMETER When a channel that does not exist is specified as an A/D conversion data duplication channel, a parameter error results	Parameter error
	ADC_ERROR_MODE If one of the following conditions is detected, a mode error will occur. • If this function is executed in continuous-scan mode • If this function is executed with software trigger setting • If this function is executed in self-diagnosis mode • If this function is executed when temperature sensor output is used • If this function is executed in compare mode • When executed when VSC_VCC pin voltage output is used(Note)	Mode error
	ADC_ERROR_BUSY If this function is executed during A/D conversion, setting failure due to a busy state results.	The setting failed because of busy state
Remarks	-	

Note. 256KB Only

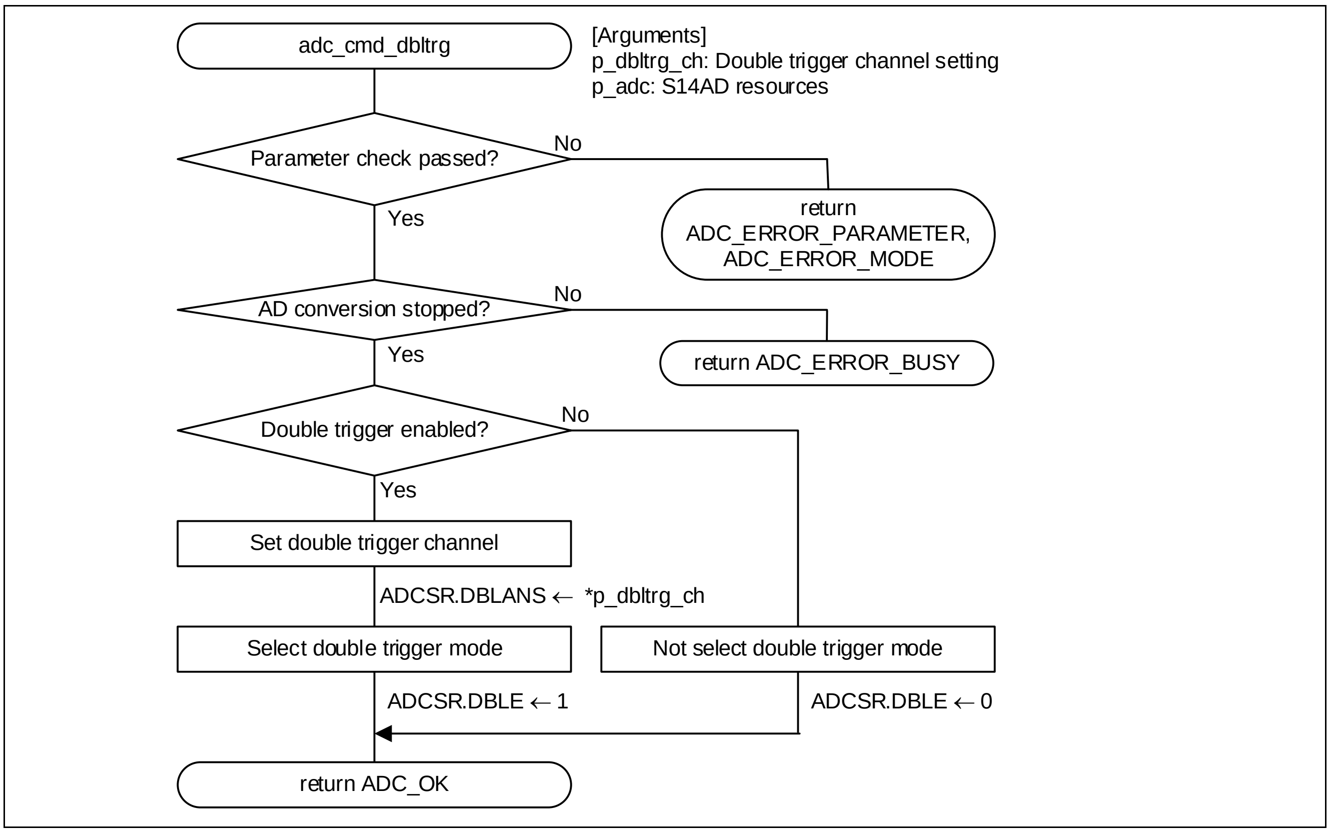

adc_cmd_dbltrg
[Arguments]
p_dbltrg_ch: Double trigger channel setting
p_adc: S14AD resources
Parameter check passed?
No
Yes
return ADC_ERROR_PARAMETER, ADC_ERROR_MODE
AD conversion stopped?
No
Yes
return ADC_ERROR_BUSY
Double trigger enabled?
No
Yes
Set double trigger channel
ADCSR.DBLANS ← *p_dbltrg_ch
Select double trigger mode
Not select double trigger mode
ADCSR.DBLE ← 1
ADCSR.DBLE ← 0
return ADC_OK

Figure 4-12 adc_cmd_dbltrg Function Processing Flow

4.1.11 adc_cmd_diag Function

Table 4-13 adc_cmd_diag Function Specifications

Format	static e_adc_err_t adc_cmd_diag(e_adc_diag_t const * const p_diag, st_adc_resources_t * const p_adc)	
Description	Specifies self-diagnosis mode settings.	
Argument	e_adc_diag_t const * const p_diag: Self-diagnosis mode setting value Set one of the following. • AD_DIAG_DISABLE: Self-diagnosis disabled • AD_DIAG_0V: 0V • AD_DIAG_HARF: Reference voltage (VREFH0) x 1/2 • AD_DIAG_BASE: Reference voltage (VREFH0) • AD_DIAG_ROTATE: Self-diagnostic voltage rotation mode	
	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.	
Return value	ADC_OK	Self-diagnosis mode set normally.
	ADC_ERROR_MODE If one of the following conditions is detected, a mode error will occur. • If VREFH is specified as the reference voltage on the high-potential side. • If executed in double-trigger mode • If disconnection detection assist function is enabled • If compare function is enabled	Mode error
	ADC_ERROR_BUSY If executed during A/D conversion, setting failure due to a busy state results.	The setting failed because of busy state.
	ADC_ERROR_PARAMETER If a value other than those stipulated is specified for self-diagnosis mode setting, a parameter will occur.	Parameter error
Remarks	-	

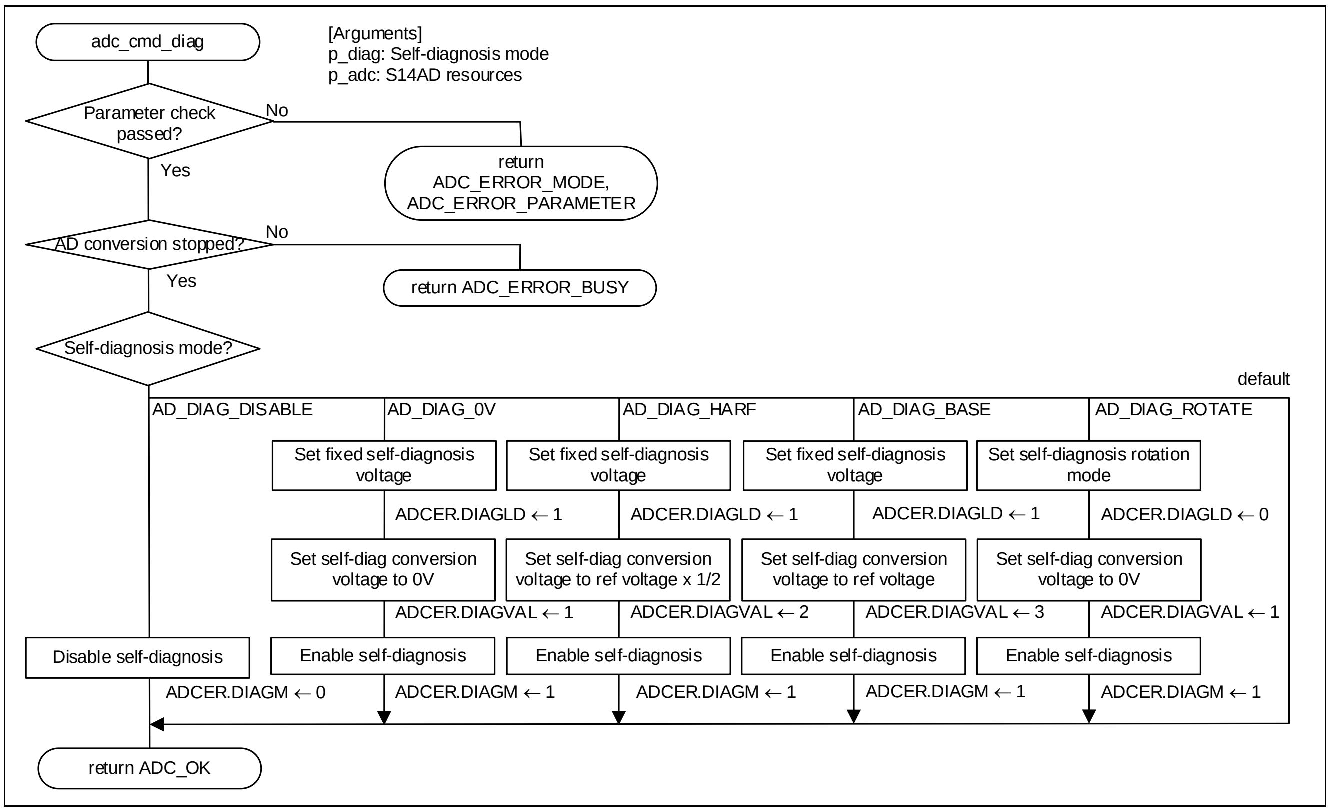


Figure 4-13 adc_cmd_diag Function Processing Flow

4.1.12 adc_cmd_auto_clear Function

Table 4-14 adc_cmd_auto_clear Function Specifications

Format	static e_adc_err_t adc_cmd_auto_clear(uint8_t const * const p_en, st_adc_resources_t * const p_adc)	
Description	Enables or disables automatic clearing of A/D data registers	
Argument	uint8_t const * const p_en: Enables/disables automatic clearing of A/D data registers ADC_ENABLE: Enable ADC_DISABLE: Disable	
	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.	
Return value	ADC_ERROR_PARAMETER If one of the following conditions is detected, a parameter error will occur. • If an argument is NULL • If a value other than those stipulated is specified for an argument	Parameter error
	ADC_OK	Automatic clearing of A/D data register enabled/disabled normally.
Remarks	-	

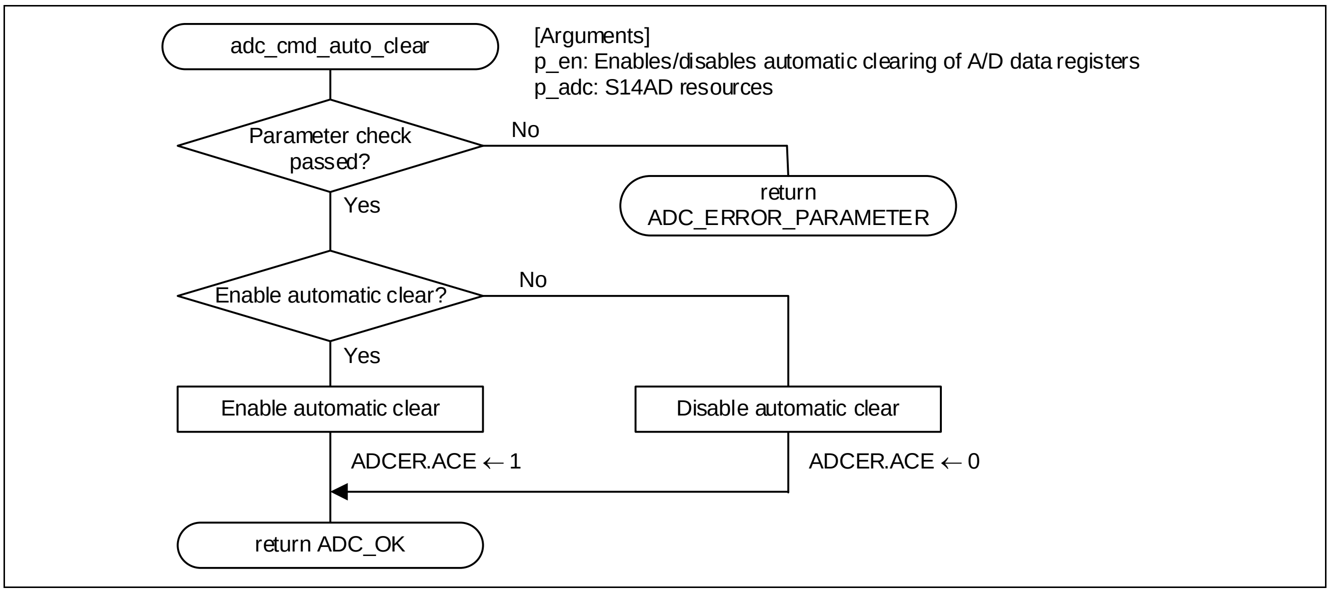


Figure 4-14 adc_cmd_auto_clear Function Processing Flow

4.1.13 adc_cmd_sampling Function

Table 4-15 adc_cmd_sampling Function Specifications

Format	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)	
Description	Specifies A/D sampling time settings.	
Argument	e_adc_sst_t sst: Channel for which the sampling time is set	
	uint8_t const * const p_sst_val: Sampling time Specify a sampling time from 2 to 255.	
	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.	
Return value	ADC_ERROR_PARAMETER If one of the following conditions is detected, a parameter error will occur. • If an argument is NULL. • If a value other than those stipulated is specified for an argument.	Parameter error
	ADC_OK	A/D sampling time set normally.
Remarks	-	

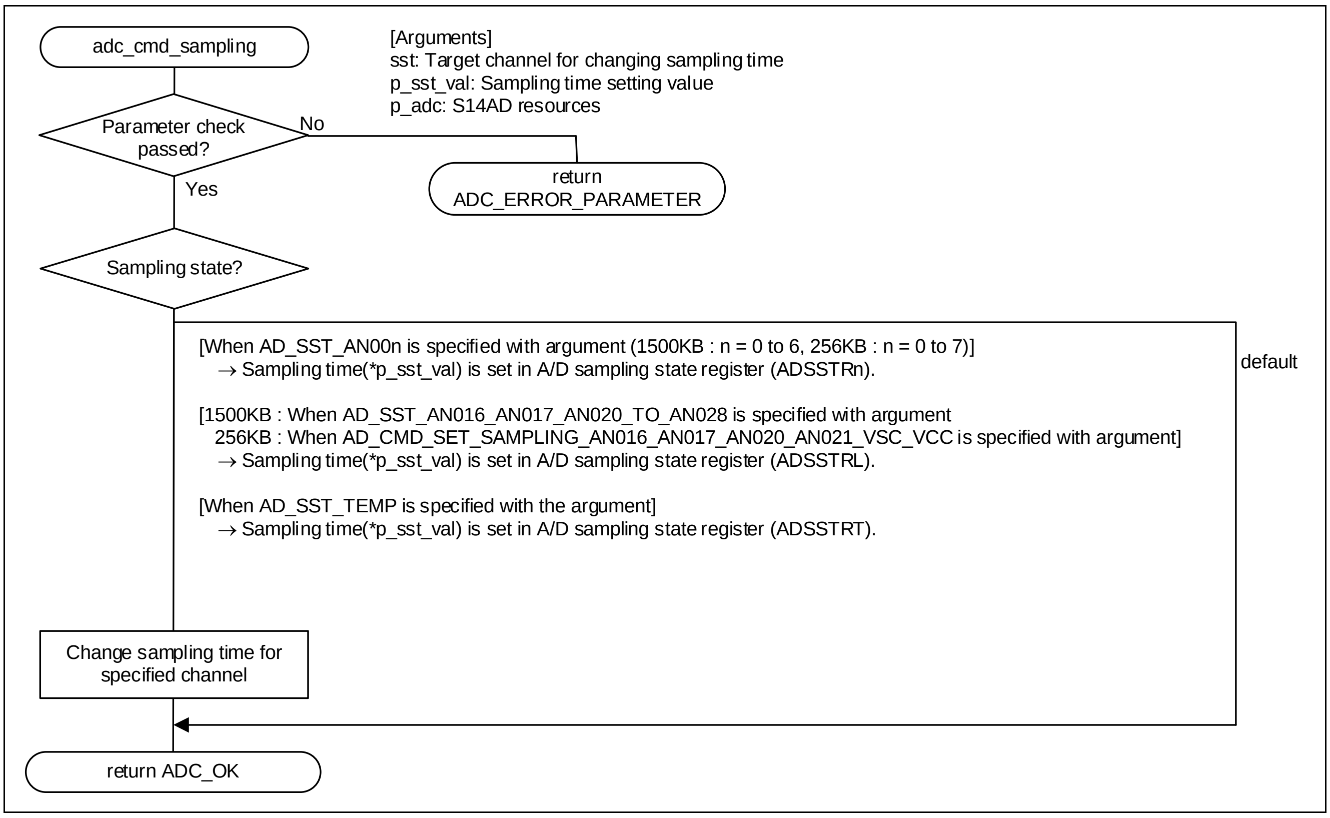

adc_cmd_sampling
[Arguments]
sst: Target channel for changing sampling time
p_sst_val: Sampling time setting value
p_adc: S14AD resources
Parameter check passed?
No
Yes
return ADC_ERROR_PARAMETER
Sampling state?
[When AD_SST_AN00n is specified with argument (1500KB : n = 0 to 6, 256KB : n = 0 to 7)]
→ Sampling time(*p_sst_val) is set in A/D sampling state register (ADSSTRn).
[1500KB : When AD_SST_AN016_AN017_AN020_TO_AN028 is specified with argument
256KB : When AD_CMD_SET_SAMPLING_AN016_AN017_AN020_AN021_VSC_VCC is specified with argument]
→ Sampling time(*p_sst_val) is set in A/D sampling state register (ADSSTRL).
[When AD_SST_TEMP is specified with the argument]
→ Sampling time(*p_sst_val) is set in A/D sampling state register (ADSSTRT).
default
Change sampling time for specified channel
return ADC_OK

Figure 4-15 adc_cmd_sampling Function Processing Flow

4.1.14 adc_cmd_charge Function

Table 4-16 adc_cmd_charge Function Specifications

Format	static e_adc_err_t adc_cmd_charge(uint8_t const * const p_charge, st_adc_resources_t * const p_adc)
Description	Sets the disconnection detection assist function.
Argument	uint8_t const * const p_charge: Disconnection detection assist setting value
	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.
Return vale	ADC_OK Disconnection detection assist set normally.
	ADC_ERROR_PARAMETER Parameter error If one of the following conditions is detected, a parameter error will occur. • If an argument is NULL. • If a value other than those stipulated is specified for an argument.
	ADC_ERROR_MODE Mode error If one of the following conditions is detected, a parameter error will occur. • If temperature sensor output is used. • If executed in self-diagnosis mode
	ADC_ERROR_BUSY The setting failed because of busy state. If executed during A/D conversion, setting failure due to a busy state results.
Remarks	-

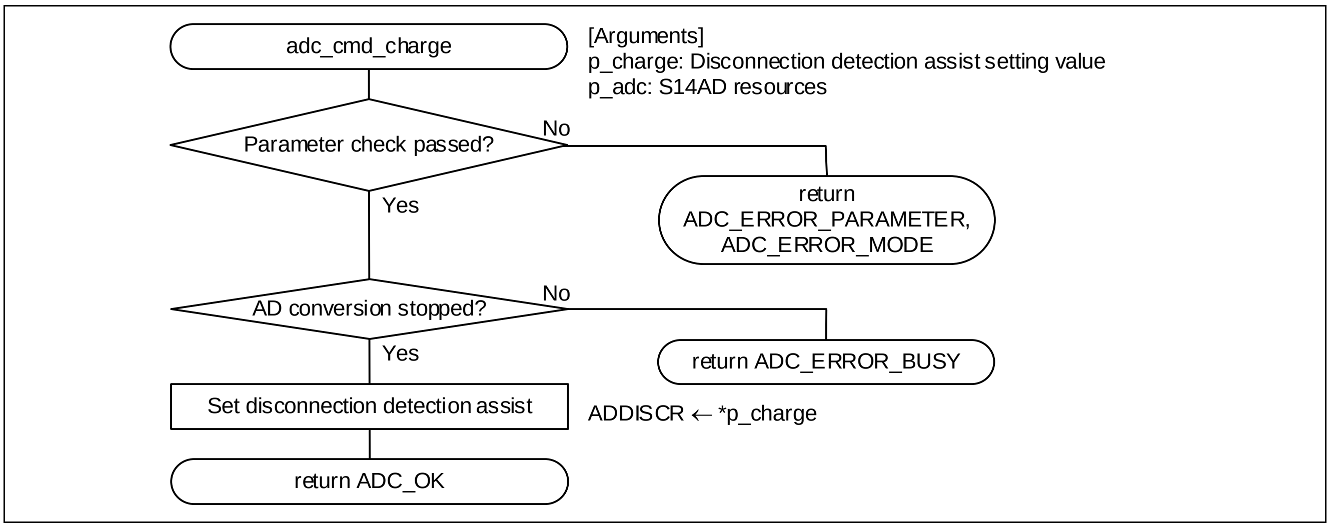


Figure 4-16 adc_cmd_charge Function Processing Flow

4.1.15 adc_cmd_group_priority Function

Table 4-17 adc_cmd_group_priority Function Specifications

Format	static e_adc_err_t adc_cmd_group_priority(e_adc_gsp_t const * const p_gsp, st_adc_resources_t * const p_adc)	
Description	Sets A/D group scan priority operation.	
Argument	e_adc_gsp_t const * const p_gsp: A/D group scan priority setting value. Set one of the following. • ADC_GSP_PRIORYTY_OFF: Group priority operation disabled • ADC_GSP_WAIT_TRG: Low-priority group restart disabled • ADC_GSP_SCAN_BIGIN: Low-priority group restart enabled • ADC_GSP_SCAN_RESTART: Rescanning is started from the channel for which A/D conversion has not completed. • ADC_GSP_WAIT_TRG_CONT_SCAN : Continuous single scanning enabled and low-priority group restart disabled • ADC_GSP_SCAN_BIGIN_CONT_SCAN : Continuous single scanning enabled and low-priority group restart enabled • ADC_GSP_SCAN_RESTART_CONT_SCAN : Continuous single scanning is enabled and rescanning is started from the channel for which A/D conversion has not completed.	
	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.	
Return value	ADC_OK	A/D group scan priority operation set normally.
	ADC_ERROR_PARAMETER If one of the following conditions is detected, a parameter error will occur. • If an argument is NULL. • If a value other than those stipulated is specified for an argument.	Parameter value
	ADC_ERROR_MODE If this function is executed in other than group scan mode, a mode error will occur.	Mode error
	ADC_ERROR_BUSY If this function is executed during A/D conversion, setting failure due to a busy state results.	The setting failed because of busy state.
Remarks	-	

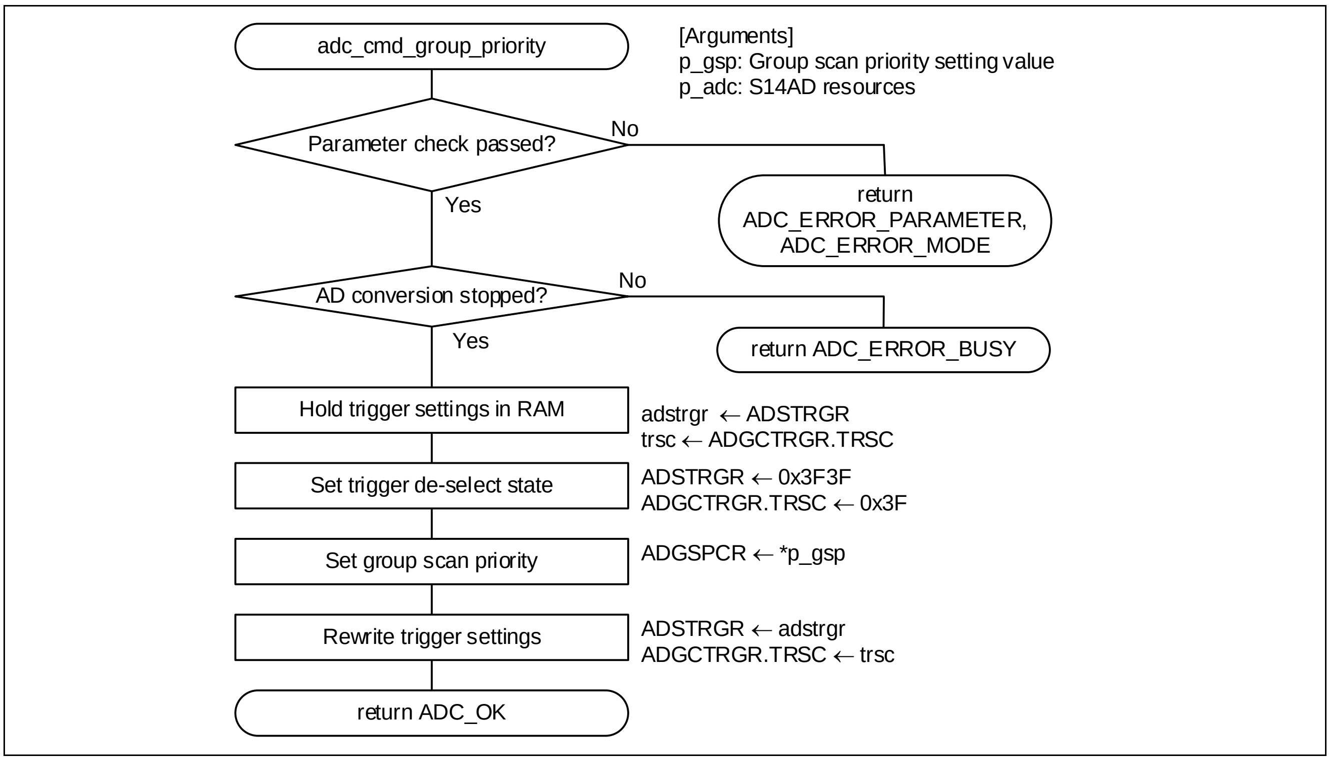


Figure 4-17 adc_cmd_group_priority Function Processing Flow

4.1.16 adc_cmd_set_windowa Function

Table 4-18 adc_cmd_set_windowa Function Specifications

Format	static e_adc_err_t adc_cmd_set_windowa(st_adc_wina_t const * const p_wina, st_adc_resources_t * const p_adc)	
Description	Sets compare function window A.	
Argument	st_adc_wina_t const * const p_wina: Compare function window A setting value	
	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.	
Return value	ADC_OK	Compare function window A set normally
	ADC_ERROR_PARAMETER If one of the following conditions is detected, a parameter error will occur. • If an argument is NULL. • If ADC_CMP_WINDOW is specified, and the upper reference level and lower reference level values are equal. • If ADC_CMP_WINDOW or ADC_CMP_LEVEL is specified, and the relevant channel is not specified. • If ADC_CMP_WINDOW or ADC_CMP_LEVEL is specified, and the relevant channel is not enabled. • If a value other than those stipulated is specified for an argument.	Parameter error
	ADC_ERROR_MODE If executed in other than group scan mode, a mode error will occur.	Mode error
	ADC_ERROR_BUSY If executed during A/D conversion, setting failure due to a busy state results.	The setting failed because of busy state.
	ADC_ERROR_SYSTEM_SETTING If one of the following conditions is detected, a system setting error will occur. • If CMPAI interrupt is defined to be unused • (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED) with r_system_cfg.h. • If, in r_adc_cfg.h, the setting for S14AD_CMPAI_PRIORITY exceeds the defined range.	System setting error
Remarks	-	

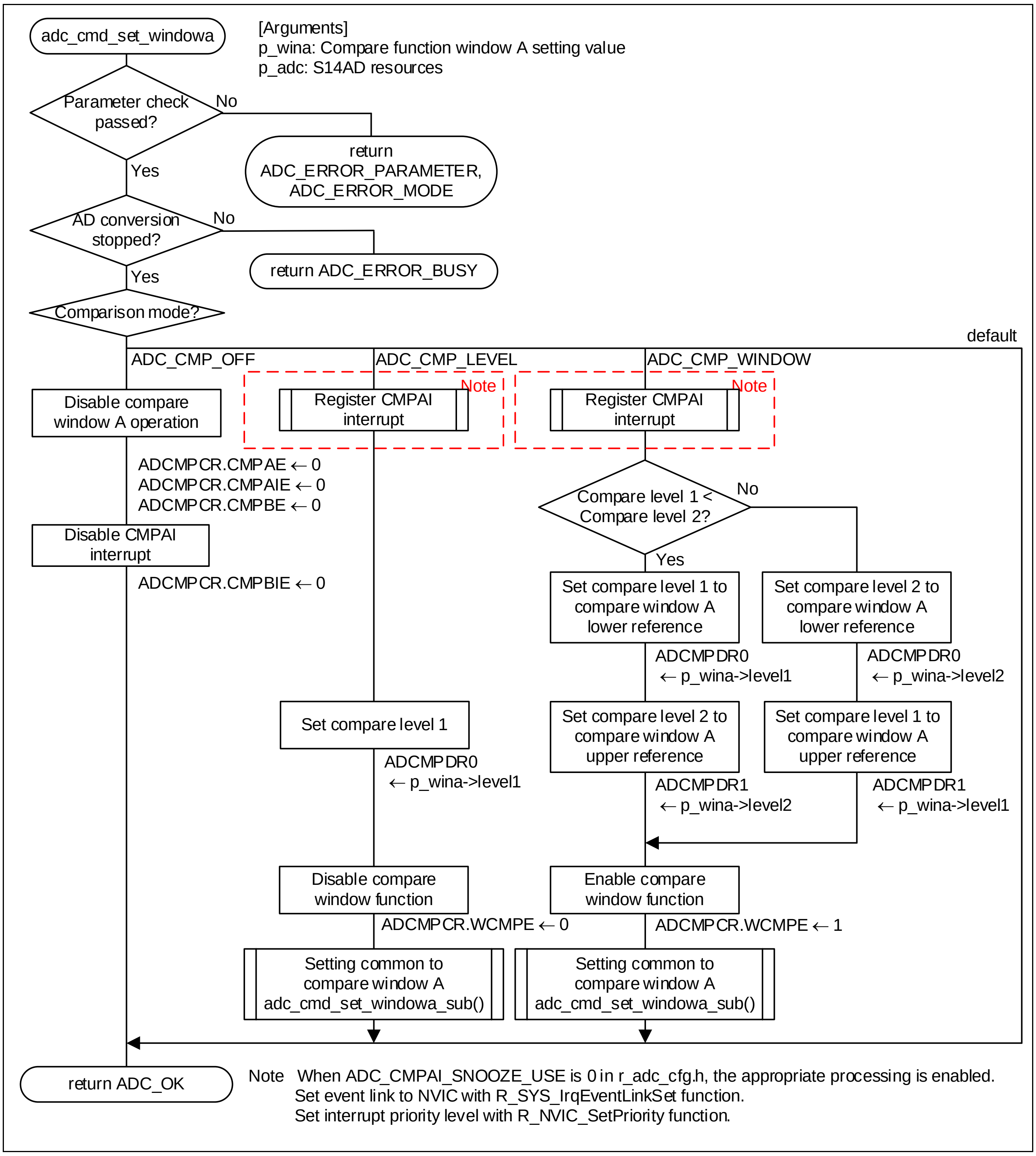

adc_cmd_set_windowa
[Arguments]
p_wina: Compare function window A setting value
p_adc: S14AD resources
Parameter check passed?
No
return ADC_ERROR_PARAMETER, ADC_ERROR_MODE
Yes
AD conversion stopped?
No
return ADC_ERROR_BUSY
Yes
Comparison mode?
default
ADC_CMP_OFF
ADC_CMP_LEVEL
ADC_CMP_WINDOW
Disable compare window A operation
Note
Register CMPAI interrupt
Note
Register CMPAI interrupt
ADCMPCR.CMPAE ← 0
ADCMPCR.CMPAIE ← 0
ADCMPCR.CMPBE ← 0
Compare level 1 < Compare level 2?
No
Disable CMPAI interrupt
Yes
ADCMPCR.CMPBIE ← 0
Set compare level 1 to compare window A lower reference
Set compare level 2 to compare window A lower reference
ADCMPDR0 ← p_wina->level1
ADCMPDR0 ← p_wina->level2
Set compare level 1
Set compare level 2 to compare window A upper reference
Set compare level 1 to compare window A upper reference
ADCMPDR0 ← p_wina->level1
ADCMPDR1 ← p_wina->level2
ADCMPDR1 ← p_wina->level1
Disable compare window function
Enable compare window function
ADCMPCR.WCMPE ← 0
ADCMPCR.WCMPE ← 1
Setting common to compare window A adc_cmd_set_windowa_sub()
Setting common to compare window A adc_cmd_set_windowa_sub()
return ADC_OK
Note When ADC_CMPAI_SNOOZE_USE is 0 in r_adc_cfg.h, the appropriate processing is enabled.
Set event link to NVIC with R_SYS_IrqEventLinkSet function.
Set interrupt priority level with R_NVIC_SetPriority function.

Figure 4-18 adc_cmd_set_windowa Function Processing Flow

4.1.17 adc_cmd_set_windowa_sub Function

Table 4-19 adc_cmd_set_windowa_sub Function Specifications

Format	static void adc_cmd_set_windowa_sub(st_adc_wina_t const * const p_wina, st_adc_resources_t * const p_adc)
Description	Sets compare function window A.
Argument	st_adc_wina_t const * const p_wina: Compare function window A setting value
	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.
Return value	-
Remarks	-

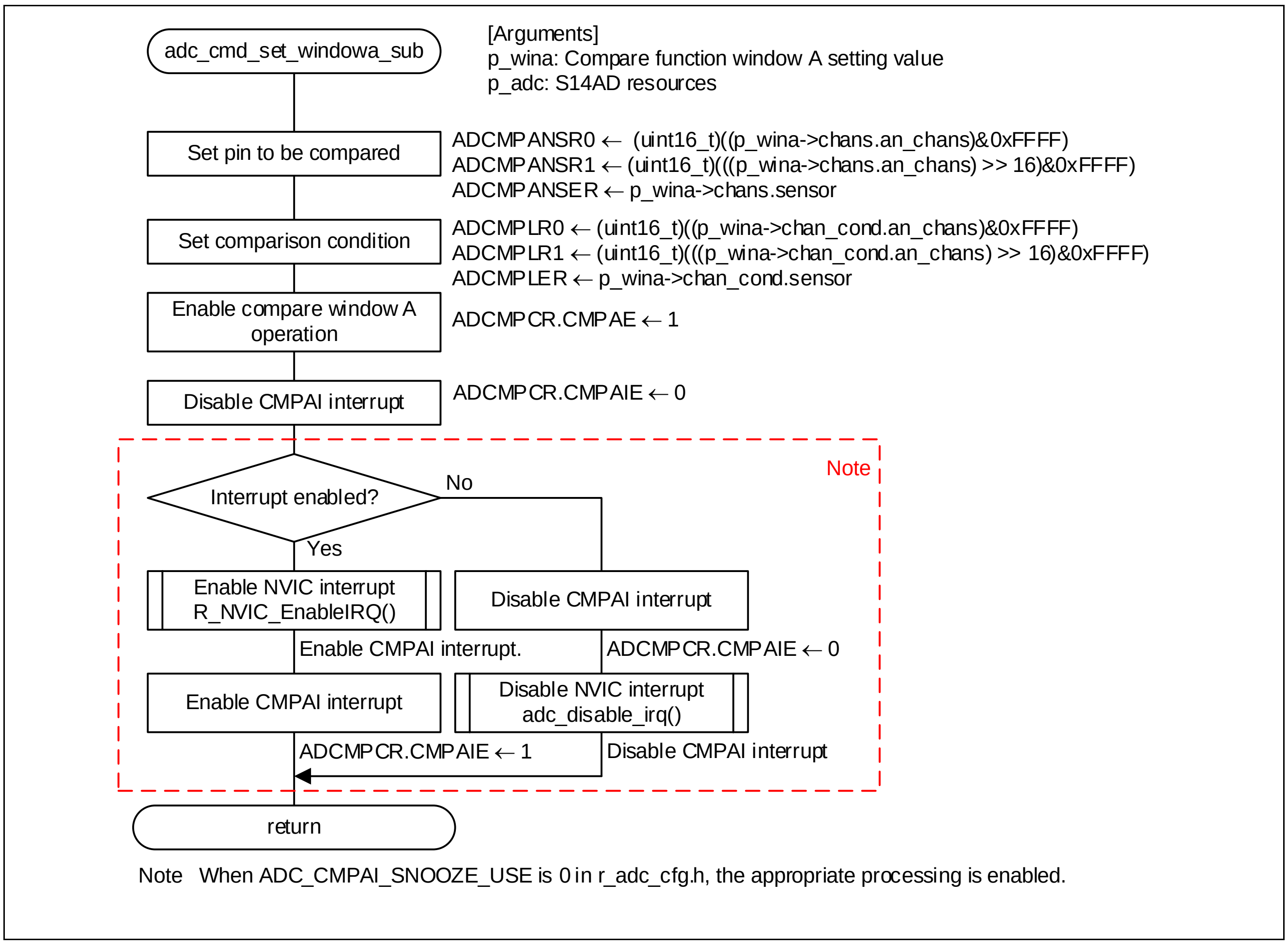
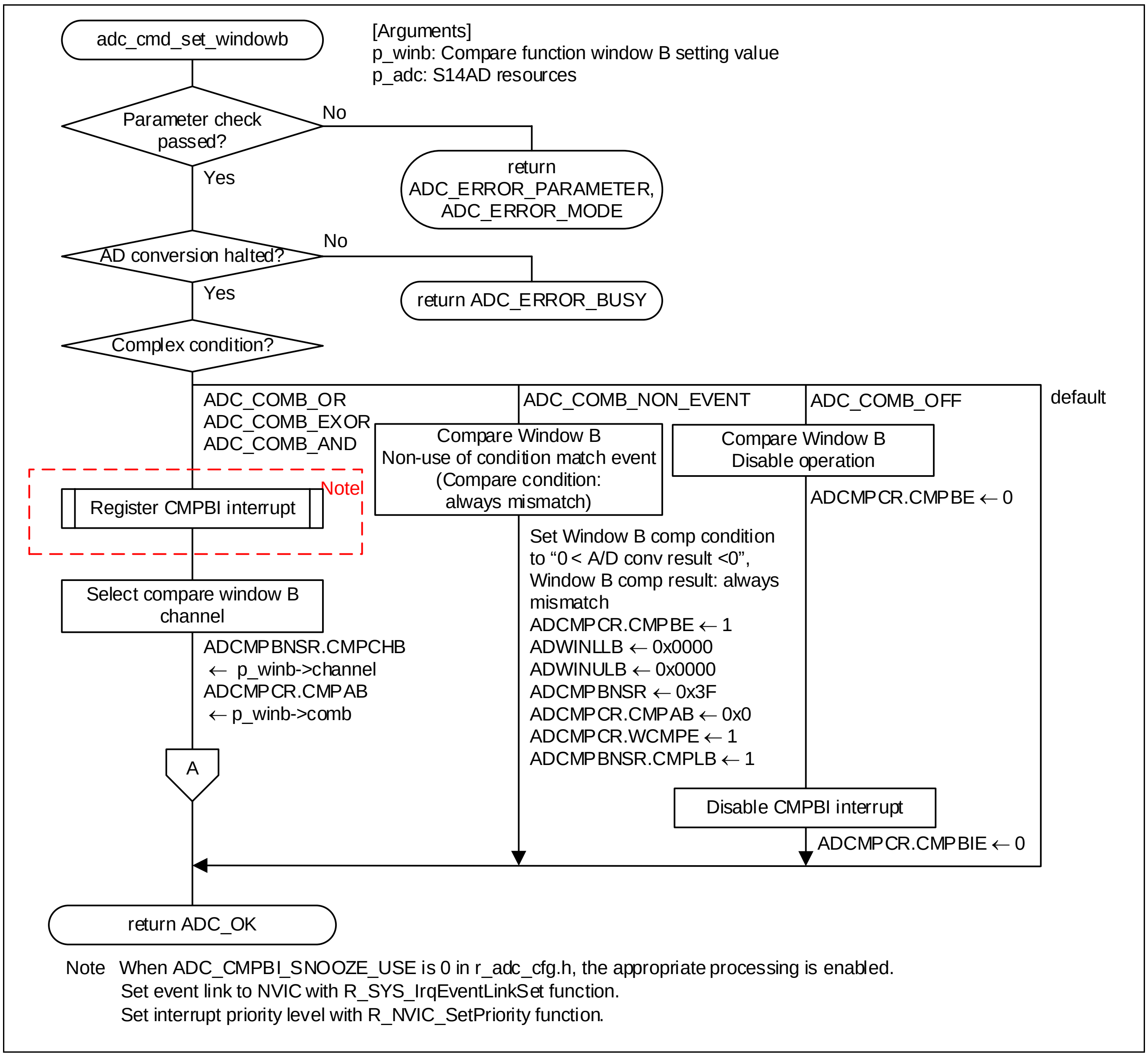

adc_cmd_set_windowa_sub
[Arguments]
p_wina: Compare function window A setting value
p_adc: S14AD resources
Set pin to be compared
ADCMPANSR0 ← (uint16_t)((p_wina->chans.an_chans)&0xFFFF)
ADCMPANSR1 ← (uint16_t)(((p_wina->chans.an_chans) >> 16)&0xFFFF)
ADCMPANSER ← p_wina->chans.sensor
Set comparison condition
ADCMPLR0 ← (uint16_t)((p_wina->chan_cond.an_chans)&0xFFFF)
ADCMPLR1 ← (uint16_t)(((p_wina->chan_cond.an_chans) >> 16)&0xFFFF)
ADCMPLER ← p_wina->chan_cond.sensor
Enable compare window A operation
ADCMPCR.CMPAE ← 1
Disable CMPAI interrupt
ADCMPCR.CMPAIE ← 0
Note
Interrupt enabled?
No
Yes
Enable NVIC interrupt R_NVIC_EnableIRQ()
Disable CMPAI interrupt
Enable CMPAI interrupt.
ADCMPCR.CMPAIE ← 0
Enable CMPAI interrupt
Disable NVIC interrupt adc_disable_irq()
ADCMPCR.CMPAIE ← 1
Disable CMPAI interrupt
return
Note When ADC_CMPAI_SNOOZE_USE is 0 in r_adc_cfg.h, the appropriate processing is enabled.

Figure 4-19 adc_cmd_set_windowa_sub Function Processing Flow

4.1.18 adc_cmd_set_windowb Function

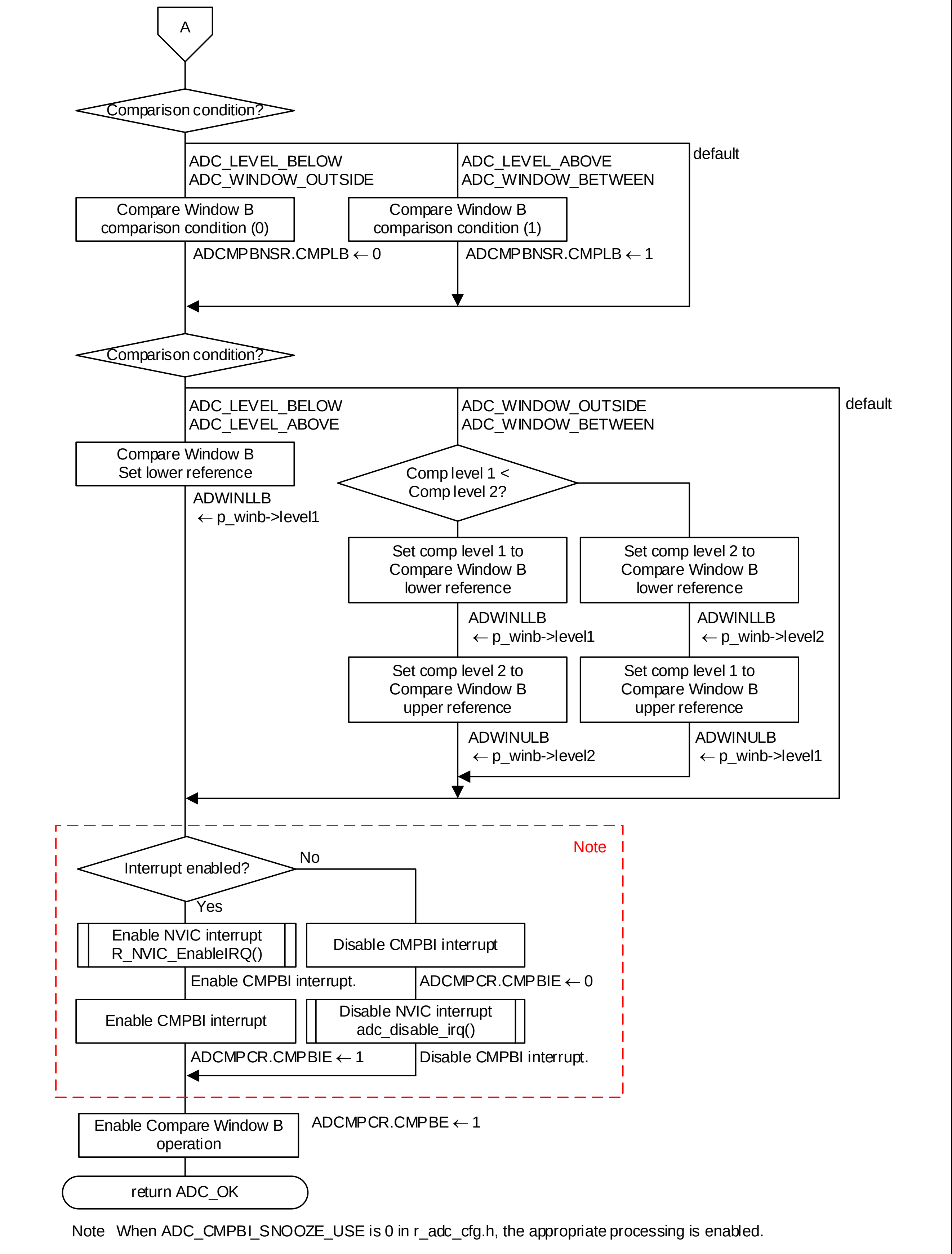
Table 4-20 adc_cmd_set_windowb Function Specifications

Format	static e_adc_err_t adc_cmd_set_windowb(st_adc_winb_t const * const p_winb, st_adc_resources_t * const p_adc)	
Description	Sets compare function window B.	
Argument	st_adc_winb_t const * const p_winb: Compare function window B setting value	
	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.	
Return value	ADC_OK	Compare function window B set normally
	ADC_ERROR_PARAMETER If one of the following conditions is detected, a parameter error will occur. • If an argument is NULL. • If a value other than those stipulated is specified for an argument.	Parameter error
	ADC_ERROR_MODE If one of the following conditions is detected, a mode error will occur. • If the specified channel is being used in window A. • If window A is disabled. • If executed in other than single scan mode. • If temperature sensor output is being used in window A.	Mode error
	ADC_ERROR_BUSY	The setting failed because of busy state.
	If executed during A/D conversion, setting failure due to a busy state results.	
	ADC_ERROR_SYSTEM_SETTING If one of the following conditions is detected, a system setting error will occur. • If CMPBI interrupt is defined to be unused (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED) with r_system_cfg.h. • If, in r_adc_cfg.h, the setting for S14AD_CMPBI_PRIORITY exceeds the defined range.	System setting error
Remarks	-	



Note When ADC_CMPBI_SNOOZE_USE is 0 in r_adc_cfg.h, the appropriate processing is enabled.
Set event link to NVIC with R_SYS_IrqEventLinkSet function.
Set interrupt priority level with R_NVIC_SetPriority function.

Figure 4-20 adc_cmd_set_windowb Function Processing Flow (1/2)


A
Comparison condition?
ADC_LEVEL_BELOW
ADC_WINDOW_OUTSIDE
ADC_LEVEL_ABOVE
ADC_WINDOW_BETWEEN
default
Compare Window B comparison condition (0)
Compare Window B comparison condition (1)
ADCMPBNSR.CMPLB ← 0
ADCMPBNSR.CMPLB ← 1
Comparison condition?
ADC_LEVEL_BELOW
ADC_LEVEL_ABOVE
ADC_WINDOW_OUTSIDE
ADC_WINDOW_BETWEEN
default
Compare Window B Set lower reference
ADWINLLB ← p_winb->level1
Comp level 1 < Comp level 2?
Set comp level 1 to Compare Window B lower reference
Set comp level 2 to Compare Window B lower reference
ADWINLLB ← p_winb->level1
ADWINLLB ← p_winb->level2
Set comp level 2 to Compare Window B upper reference
Set comp level 1 to Compare Window B upper reference
ADWINULB ← p_winb->level2
ADWINULB ← p_winb->level1
Note
Interrupt enabled?
No
Yes
Enable NVIC interrupt R_NVIC_EnableIRQ()
Disable CMPBI interrupt
Enable CMPBI interrupt.
ADCMPCR.CMPBIE ← 0
Enable CMPBI interrupt
Disable NVIC interrupt adc_disable_irq()
ADCMPCR.CMPBIE ← 1
Disable CMPBI interrupt.
Enable Compare Window B operation
ADCMPCR.CMPBE ← 1
return ADC_OK

Note When ADC_CMPBI_SNOOZE_USE is 0 in r_adc_cfg.h, the appropriate processing is enabled.

Figure 4-21 adc_cmd_set_windowb Function Processing Flow (2/2)

4.1.19 adc_cmd_get_cmp_result Function

Table 4-21 adc_cmd_get_cmp_result Function Specifications

Format	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)	
Description	Acquires the comparison results of compare function.	
Argument	st_adc_cmp_result_t * const p_cmp_result: Storage destination address for the comparison results of compare function	
	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.	
Return value	ADC_OK	Comparison results of compare function acquired normally
Remarks	-	

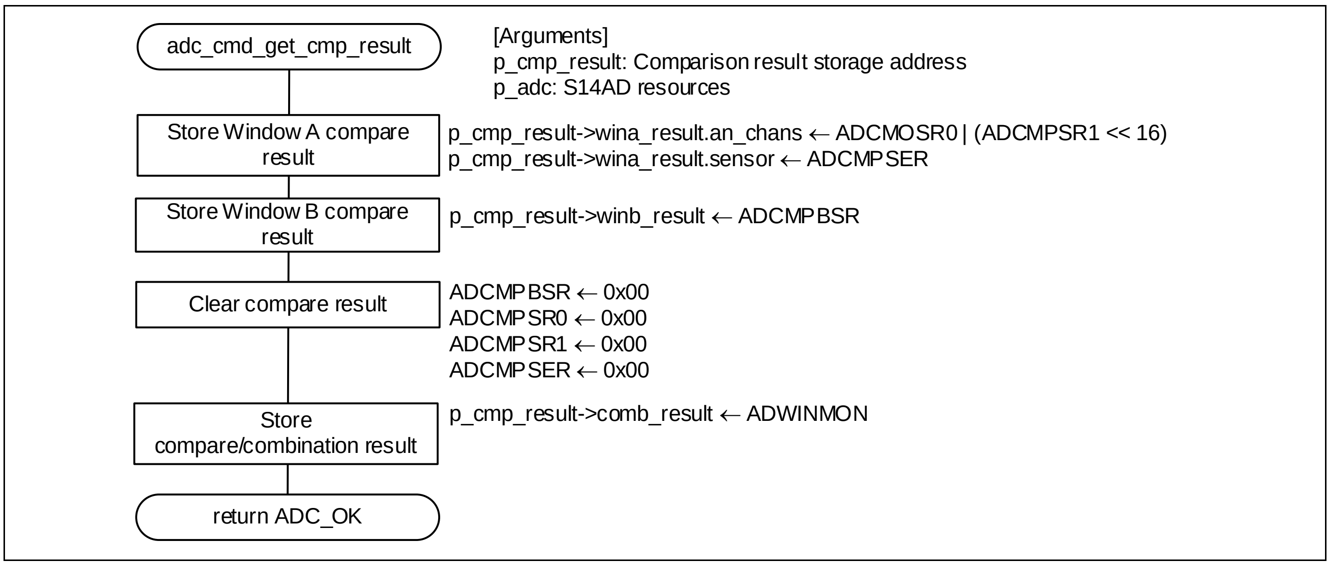

adc_cmd_get_cmp_result
[Arguments]
p_cmp_result: Comparison result storage address
p_adc: S14AD resources
Store Window A compare result
p_cmp_result->wina_result.an_chans ← ADCMOSR0 | (ADCMPSR1 << 16)
p_cmp_result->wina_result.sensor ← ADCMPSER
Store Window B compare result
p_cmp_result->winb_result ← ADCMPBSR
Clear compare result
ADCMPBSR ← 0x00
ADCMPSR0 ← 0x00
ADCMPSR1 ← 0x00
ADCMPSER ← 0x00
Store compare/combination result
p_cmp_result->comb_result ← ADWINMON
return ADC_OK

Figure 4-22 adc_cmd_get_cmp_result Function Processing Flow

4.1.20 adc_cmd_get_state Function

Table 4-22 adc_cmd_get_state Function Specifications

Format	static e_adc_err_t adc_cmd_get_state(st_adc_status_info_t * const p_state, st_adc_resources_t * const p_adc)	
Description	Acquires S14AD status.	
Argument	st_adc_status_info_t * const p_state: Storage address for S14AD status information	
	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.	
Return value	ADC_OK	Comparison results of compare function acquired normally
Remarks	-	

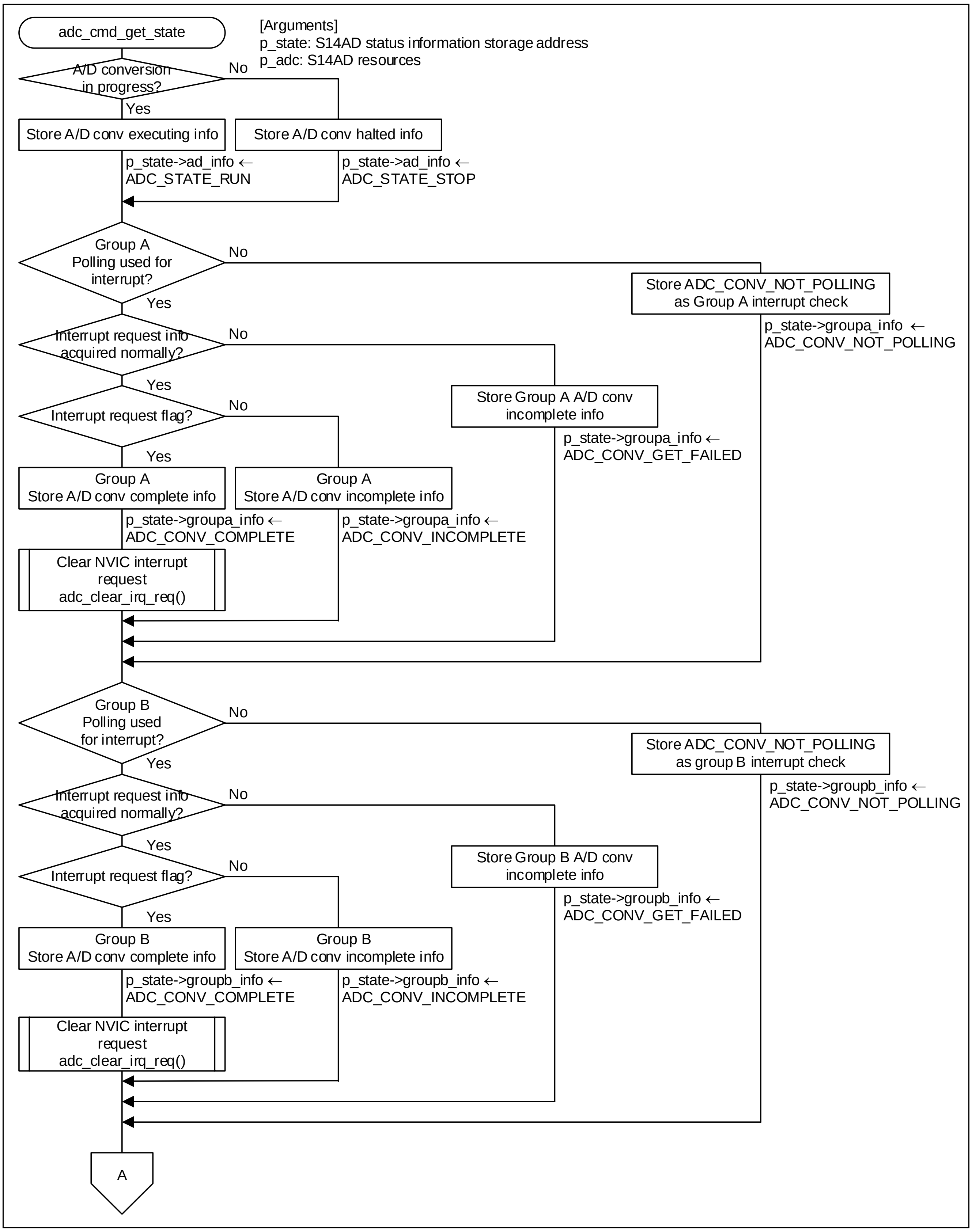

adc_cmd_get_state
[Arguments]
p_state: S14AD status information storage address
p_adc: S14AD resources
A/D conversion in progress?
No
Yes
Store A/D conv executing info
Store A/D conv halted info
p_state->ad_info ← ADC_STATE_RUN
p_state->ad_info ← ADC_STATE_STOP
Group A Polling used for interrupt?
No
Yes
Store ADC_CONV_NOT_POLLING as Group A interrupt check
p_state->groupa_info ← ADC_CONV_NOT_POLLING
Interrupt request info acquired normally?
No
Yes
Store Group A A/D conv incomplete info
p_state->groupa_info ← ADC_CONV_GET_FAILED
Interrupt request flag?
No
Yes
Group A Store A/D conv complete info
Group A Store A/D conv incomplete info
p_state->groupa_info ← ADC_CONV_COMPLETE
p_state->groupa_info ← ADC_CONV_INCOMPLETE
Clear NVIC interrupt request adc_clear_irq_req()
Group B Polling used for interrupt?
No
Yes
Store ADC_CONV_NOT_POLLING as group B interrupt check
p_state->groupb_info ← ADC_CONV_NOT_POLLING
Interrupt request info acquired normally?
No
Yes
Store Group B A/D conv incomplete info
p_state->groupb_info ← ADC_CONV_GET_FAILED
Interrupt request flag?
No
Yes
Group B Store A/D conv complete info
Group B Store A/D conv incomplete info
p_state->groupb_info ← ADC_CONV_COMPLETE
p_state->groupb_info ← ADC_CONV_INCOMPLETE
Clear NVIC interrupt request adc_clear_irq_req()
A

Figure 4-23 adc_cmd_get_state Function Processing Flow (1/3)

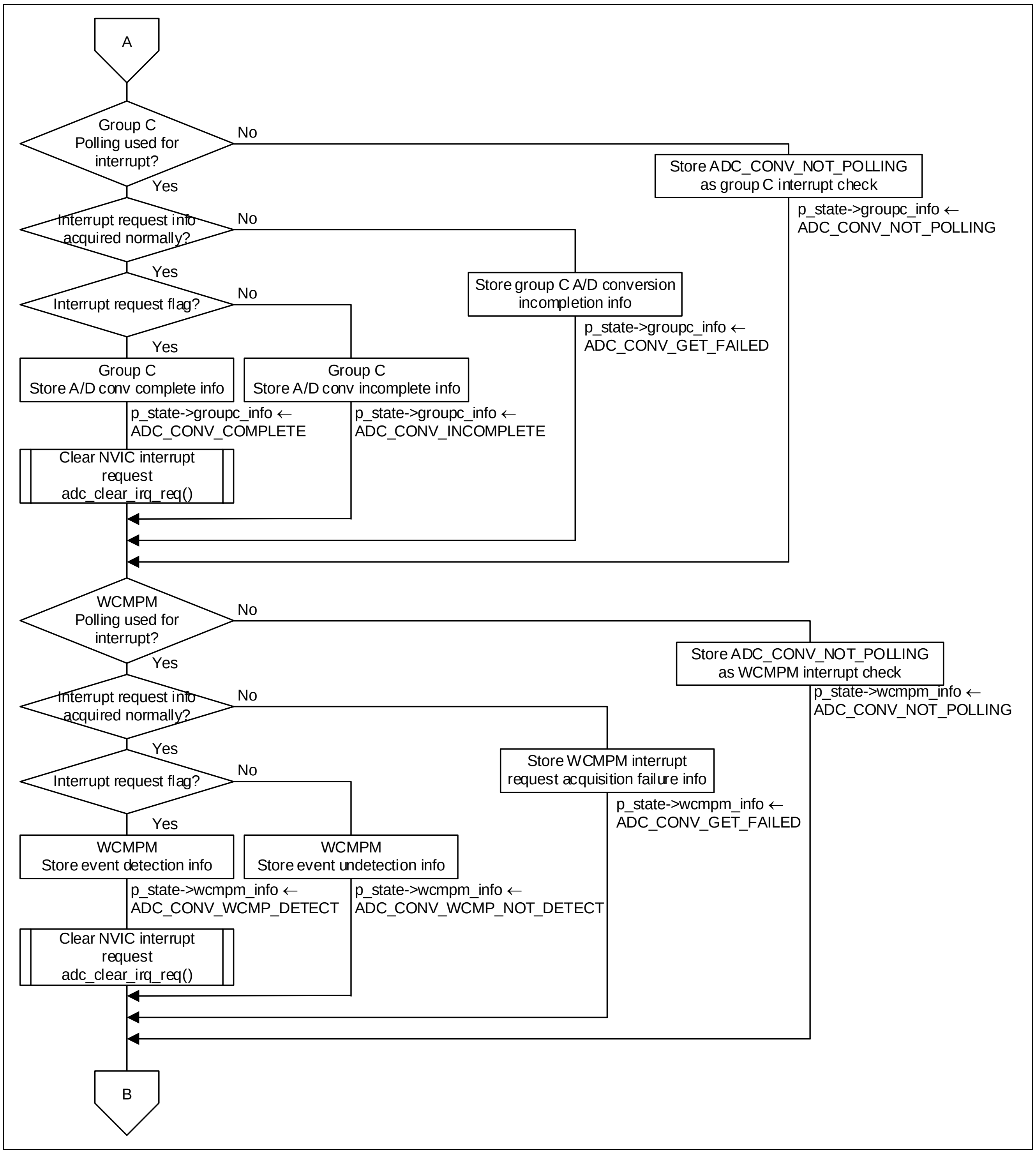

A
Group C Polling used for interrupt?
No
Yes
Store ADC_CONV_NOT_POLLING as group C interrupt check
p_state->groupc_info ← ADC_CONV_NOT_POLLING
Interrupt request info acquired normally?
No
Yes
Store group C A/D conversion incompletion info
p_state->groupc_info ← ADC_CONV_GET_FAILED
Interrupt request flag?
No
Yes
Group C Store A/D conv complete info
p_state->groupc_info ← ADC_CONV_COMPLETE
Group C Store A/D conv incomplete info
p_state->groupc_info ← ADC_CONV_INCOMPLETE
Clear NVIC interrupt request adc_clear_irq_req()
WCMPM Polling used for interrupt?
No
Yes
Store ADC_CONV_NOT_POLLING as WCMPM interrupt check
p_state->wcmpm_info ← ADC_CONV_NOT_POLLING
Interrupt request info acquired normally?
No
Yes
Store WCMPM interrupt request acquisition failure info
p_state->wcmpm_info ← ADC_CONV_GET_FAILED
Interrupt request flag?
No
Yes
WCMPM Store event detection info
p_state->wcmpm_info ← ADC_CONV_WCMP_DETECT
WCMPM Store event undetection info
p_state->wcmpm_info ← ADC_CONV_WCMP_NOT_DETECT
Clear NVIC interrupt request adc_clear_irq_req()
B

Figure 4-24 adc_cmd_get_state Function Processing Flow (2/3)

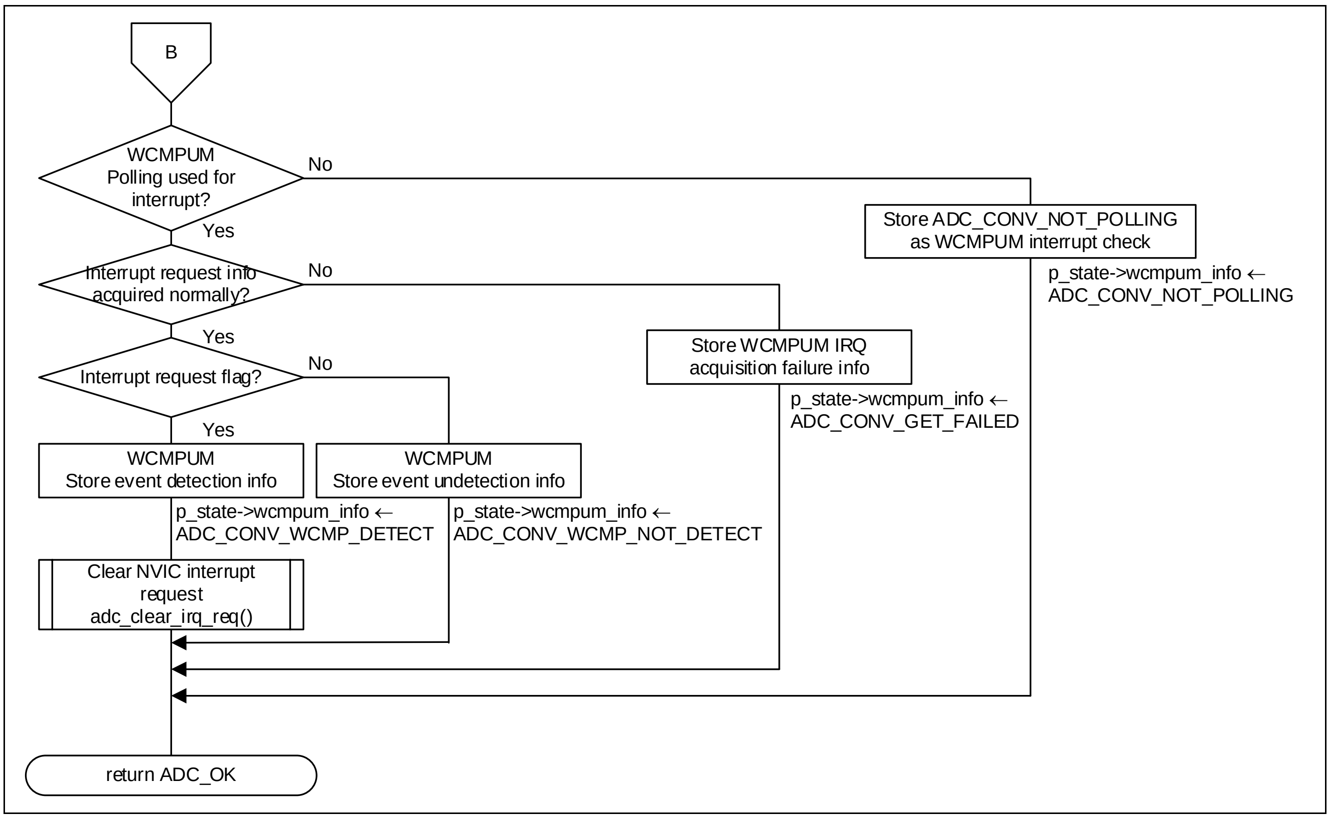

B
WCMPUM Polling used for interrupt?
No
Yes
Store ADC_CONV_NOT_POLLING as WCMPUM interrupt check
p_state->wcmpum_info ← ADC_CONV_NOT_POLLING
Interrupt request info acquired normally?
No
Yes
Store WCMPUM IRQ acquisition failure info
p_state->wcmpum_info ← ADC_CONV_GET_FAILED
Interrupt request flag?
No
Yes
WCMPUM Store event detection info
WCMPUM Store event undetection info
p_state->wcmpum_info ← ADC_CONV_WCMP_DETECT
p_state->wcmpum_info ← ADC_CONV_WCMP_NOT_DETECT
Clear NVIC interrupt request adc_clear_irq_req()
return ADC_OK

Figure 4-25 adc_cmd_get_state Function Processing Flow (3/3)

4.1.21 adc_cmd_set_vrefl0 Function

Table 4-23 adc_cmd_set_vrefl0 Function Specifications

Format	static e_adc_err_t adc_cmd_set_vrefl0(st_adc_resources_t * const p_adc)	
Description	Acquires the comparison results of compare function.	
Argument	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.	
Return value	ADC_OK	Comparison results of compare function acquired normally
	ADC_ERROR_BUSY If executed during A/D conversion, setting failure due to a busy state results.	The setting failed because of busy state.
Remarks	-	

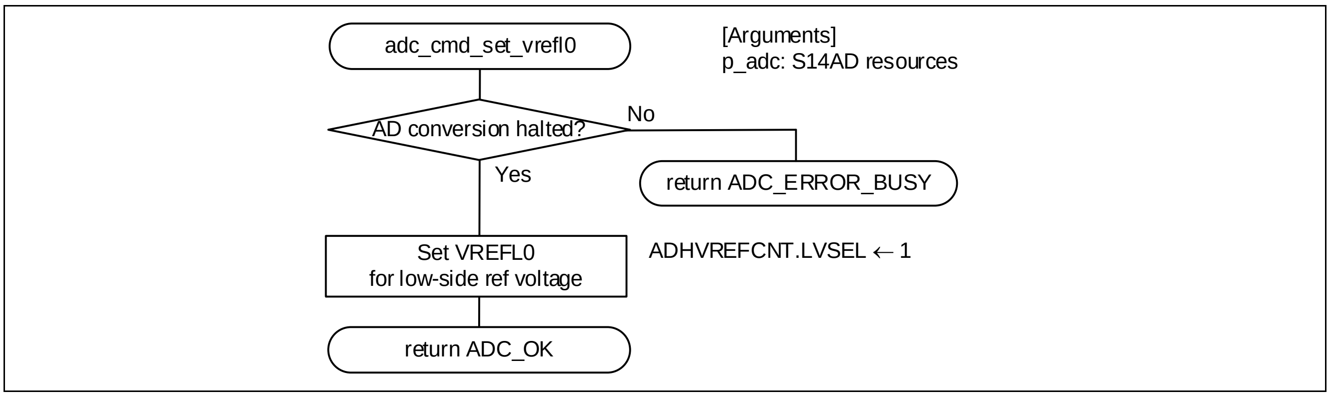


Figure 4-26 adc_cmd_set_vrefl0 Function Processing Flow

4.1.22 adc_cmd_set_vrefh0 Function

Table 4-24 adc_cmd_set_vrefh0 Function Specifications

Format	static e_adc_err_t adc_cmd_set_vrefh0(st_adc_resources_t * const p_adc)	
Description	Acquires the comparison results of compare function.	
Argument	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.	
Return value	ADC_OK	Comparison results of compare function acquired normally
	ADC_ERROR_MODE	Mode error
	If one of the following conditions is detected, a mode error will occur. • If self-diagnosis is enabled in self-diagnosis voltage rotation mode. • If self-diagnosis is enabled and reference voltage ×1/2 is selected as the self-diagnosis conversion voltage • If self-diagnosis is enabled and reference voltage is selected as the self-diagnosis conversion voltage	
	ADC_ERROR_BUSY	The setting failed because of busy state.
	If executed during A/D conversion, setting failure due to a busy state results.	
Remarks	-	

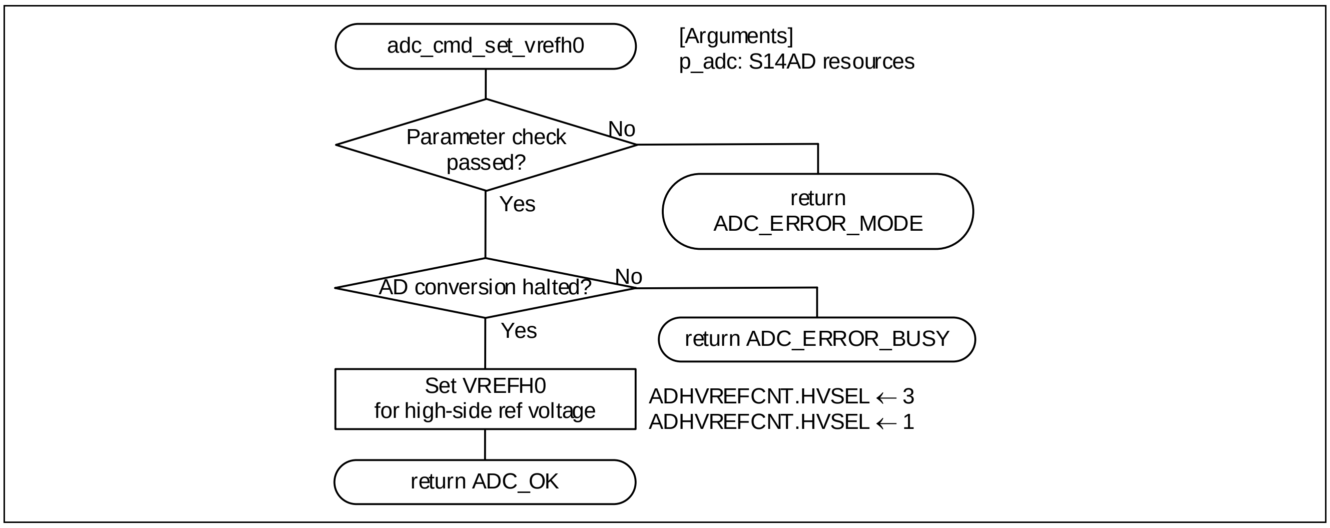


Figure 4-27 adc_cmd_set_vrefh0 Function Processing Flow

4.1.23 adc_cmd_set_sclk Function

Table 4-25 adc_cmd_set_sclk Function Specifications

Format	static e_adc_err_t adc_cmd_set_sclk(st_adc_resources_t * const p_adc)	
Description	Sets sub-clock mode operation.	
Argument	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.	
Return value	ADC_OK	Sub-clock mode set normally
	ADC_ERROR When the operation clock (ADCLK) is higher than 32768 Hz (is not set to the low-frequency operation clock (32 kHz)), sub-clock mode failure results.	Sub-clock mode setting failed.
	ADC_ERROR_BUSY If this function is executed during A/D conversion, setting failure due to a busy state results.	The setting failed because of busy state.
Remarks	-	

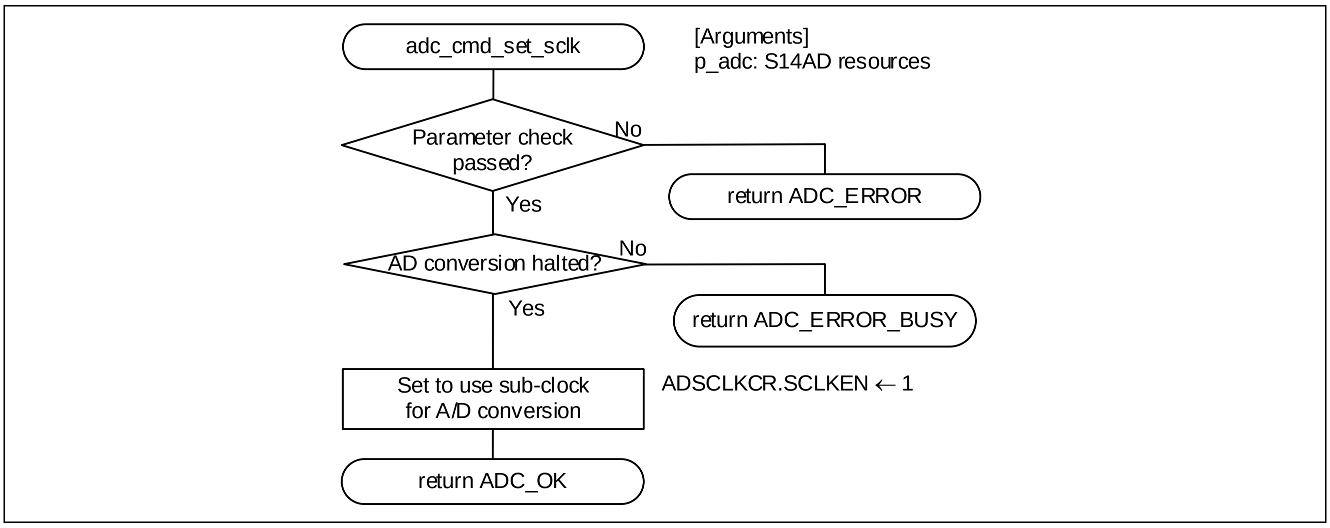


Figure 4-28 adc_cmd_set_sclk Function Processing Flow

4.1.24 adc_cmd_calibration Function

Table 4-26 adc_cmd_calibration Function Specifications

Format	static e_adc_err_t adc_cmd_calibration(st_adc_resources_t * const p_adc)	
Description	Performs offset calibration.	
Argument	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.	
Return value	ADC_OK	Offset calibration completed.
	ADC_ERROR	Offset calibration failed.
	When using the ScanSet function for A/D driver initialization (A/D conversion pin settings, A/D conversion start trigger settings), offset calibration failure results	
	ADC_ERROR_BUSY	The setting failed because of busy state
	If this function is executed during A/D conversion, setting failure due to a busy state results.	
Remarks	-	

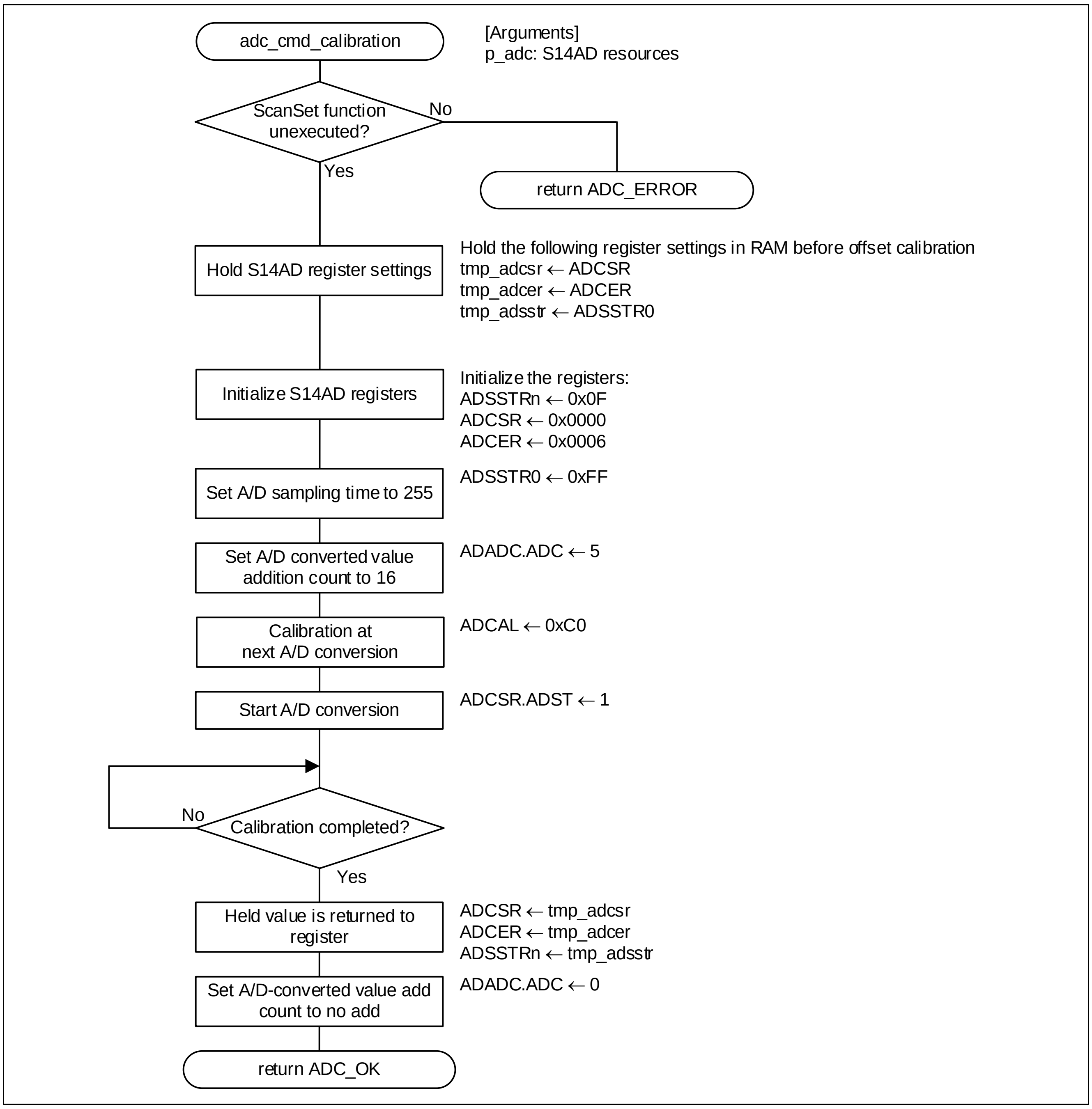

adc_cmd_calibration
[Arguments]
p_adc: S14AD resources
ScanSet function unexecuted?
No
Yes
return ADC_ERROR
Hold S14AD register settings
Hold the following register settings in RAM before offset calibration
tmp_adcsr ← ADCSR
tmp_adcer ← ADCER
tmp_adsstr ← ADSSTR0
Initialize S14AD registers
Initialize the registers:
ADSSTRn ← 0x0F
ADCSR ← 0x0000
ADCER ← 0x0006
Set A/D sampling time to 255
ADSSTR0 ← 0xFF
Set A/D converted value addition count to 16
ADADC.ADC ← 5
Calibration at next A/D conversion
ADCAL ← 0xC0
Start A/D conversion
ADCSR.ADST ← 1
Calibration completed?
No
Yes
Held value is returned to register
ADCSR ← tmp_adcsr
ADCER ← tmp_adcer
ADSSTRn ← tmp_adsstr
Set A/D-converted value add count to no add
ADADC.ADC ← 0
return ADC_OK

Figure 4-29 adc_cmd_calibration Function Processing Flow

4.1.25 adc_cmd_set_adi Function

Table 4-27 adc_cmd_set_adi Function Specifications

Format	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)	
Description	Sets a single scan or group A scan end interrupt (ADC140_ADI) (hereafter, ADI interrupt)	
Argument	e_adc_int_method_t const * const p_met: Interrupt function setting Set one of the following. ADC_INT_DISABLE: Interrupt disable ADC_INT_POLLING: Polling ADC_INT_ENABLE: Interrupt enable	
	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.	
Return value	ADC_OK	ADI interrupt function set normally
	ADC_ERROR When an ADI interrupt is being used as a DMA transfer trigger for auto-read function, an ADI interrupt setting failure results	ADI interrupt function setting failed.
	ADC_ERROR_PARAMETER If a value other than those stipulated is specified for an argument, a parameter error will occur.	Parameter error
	ADC_ERROR_SYSTEM_SETTING If one of the following conditions is detected, a system error will occur. • If ADI interrupt is defined to be unused (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED) with r_system_cfg.h. • If, in r_adc_cfg.h, the setting for S14AD_ADI_PRIORITY exceeds the defined range	System error
Remarks	-	

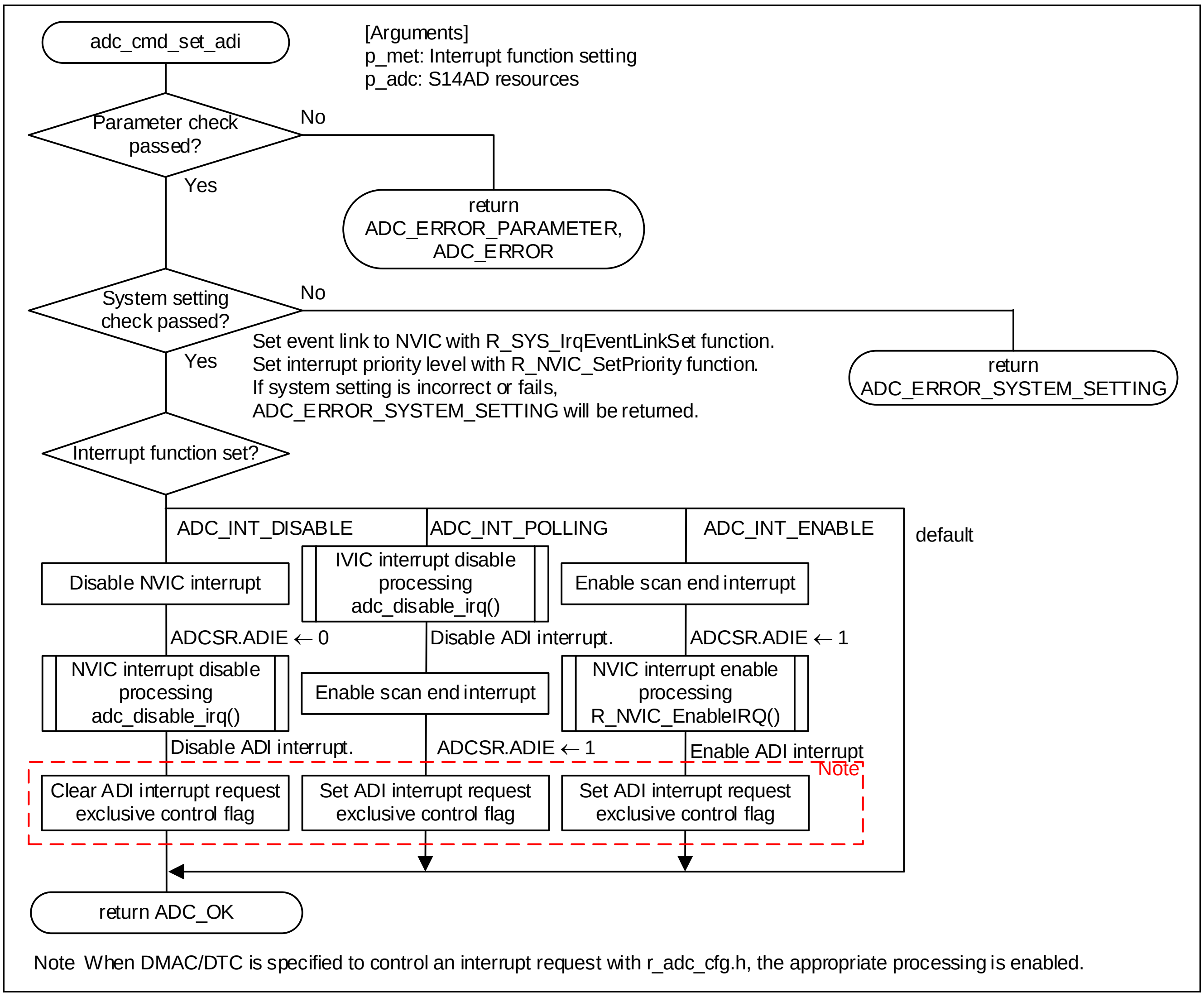
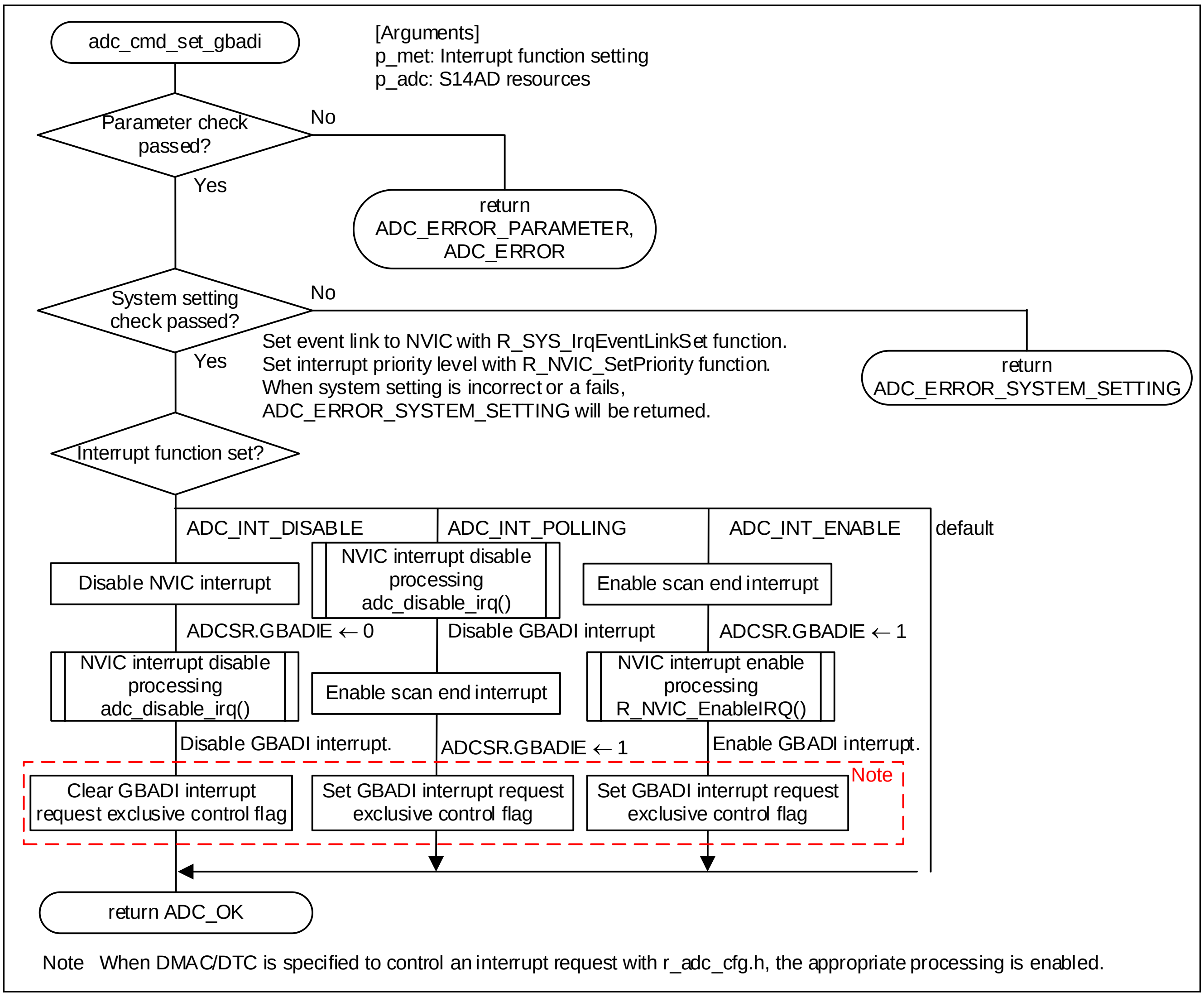

adc_cmd_set_adi
[Arguments]
p_met: Interrupt function setting
p_adc: S14AD resources
Parameter check passed?
No
Yes
return ADC_ERROR_PARAMETER, ADC_ERROR
System setting check passed?
No
Yes
Set event link to NVIC with R_SYS_IrqEventLinkSet function.
Set interrupt priority level with R_NVIC_SetPriority function.
If system setting is incorrect or fails,
ADC_ERROR_SYSTEM_SETTING will be returned.
return ADC_ERROR_SYSTEM_SETTING
Interrupt function set?
ADC_INT_DISABLE
ADC_INT_POLLING
ADC_INT_ENABLE
default
Disable NVIC interrupt
IVIC interrupt disable processing adc_disable_irq()
Enable scan end interrupt
ADCSR.ADIE ← 0
Disable ADI interrupt.
ADCSR.ADIE ← 1
NVIC interrupt disable processing adc_disable_irq()
Enable scan end interrupt
NVIC interrupt enable processing R_NVIC_EnableIRQ()
Disable ADI interrupt.
ADCSR.ADIE ← 1
Enable ADI interrupt
Note
Clear ADI interrupt request exclusive control flag
Set ADI interrupt request exclusive control flag
Set ADI interrupt request exclusive control flag
return ADC_OK
Note When DMAC/DTC is specified to control an interrupt request with r_adc_cfg.h, the appropriate processing is enabled.

Figure 4-30 adc_cmd_set_adi Function Processing Flow

4.1.26 adc_cmd_set_gbadi Function

Table 4-28 adc_cmd_set_gbadi Function Specifications

Format	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)	
Description	Sets a group B scan end interrupt (ADC140_GBADI) (hereafter GBADI interrupt)	
Argument	e_adc_int_method_t const * const p_met: Interrupt function setting Set one of the following. ADC_INT_DISABLE: Interrupt disable ADC_INT_POLLING: Polling ADC_INT_ENABLE: Interrupt enable	
	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.	
Return value	ADC_OK	GBADI interrupt function set normally
	ADC_ERROR	GBADI interrupt function setting failed.
	When a GBADI interrupt is being used as a DMA transfer trigger for auto-read function, a GBADI interrupt setting failure results	
	ADC_ERROR_PARAMETER	Parameter error
	If a value other than those stipulated is specified for an argument, a parameter error will occur.	
	ADC_ERROR_SYSTEM_SETTING	System error
	If one of the following conditions is detected, a system error will occur. • If GBADI interrupt is defined to be unused (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED) with r_system_cfg.h. • If, in r_adc_cfg.h, the setting for S14AD_GBADI_PRIORITY exceeds the defined range.	
Remarks	-	


adc_cmd_set_gbadi
[Arguments]
p_met: Interrupt function setting
p_adc: S14AD resources
Parameter check passed?
No
Yes
return ADC_ERROR_PARAMETER, ADC_ERROR
System setting check passed?
No
Yes
Set event link to NVIC with R_SYS_IrqEventLinkSet function.
Set interrupt priority level with R_NVIC_SetPriority function.
When system setting is incorrect or a fails, ADC_ERROR_SYSTEM_SETTING will be returned.
return ADC_ERROR_SYSTEM_SETTING
Interrupt function set?
ADC_INT_DISABLE
ADC_INT_POLLING
ADC_INT_ENABLE
default
Disable NVIC interrupt
NVIC interrupt disable processing adc_disable_irq()
Enable scan end interrupt
ADCSR.GBADIE ← 0
Disable GBADI interrupt
ADCSR.GBADIE ← 1
NVIC interrupt disable processing adc_disable_irq()
Enable scan end interrupt
NVIC interrupt enable processing R_NVIC_EnableIRQ()
Disable GBADI interrupt.
ADCSR.GBADIE ← 1
Enable GBADI interrupt.
Note
Clear GBADI interrupt request exclusive control flag
Set GBADI interrupt request exclusive control flag
Set GBADI interrupt request exclusive control flag
return ADC_OK

Note When DMAC/DTC is specified to control an interrupt request with r_adc_cfg.h, the appropriate processing is enabled.

Figure 4-31 adc_cmd_set_gbadi Function Processing Flow

4.1.27 adc_cmd_set_gcadi Function

Table 4-29 adc_cmd_set_gcadi Function Specifications

Format	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)	
Description	Sets a group C scan end interrupt (ADC140_GCADI) (hereafter GCADI interrupt).	
Argument	e_adc_int_method_t const * const p_met: Interrupt function setting Set one of the following. ADC_INT_DISABLE: Interrupt disable ADC_INT_POLLING: Polling ADC_INT_ENABLE: Interrupt enable	
	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.	
Return value	ADC_OK	GCADI interrupt function set normally
	ADC_ERROR	GCADI interrupt function setting failed
	When a GCADI interrupt is being used as a DMA transfer trigger for auto-read function, a GCADI interrupt setting failure results	
	ADC_ERROR_PARAMETER	Parameter error
	If a value other than those stipulated is specified for an argument, a parameter error will occur.	
	ADC_ERROR_SYSTEM_SETTING	System error
	If one of the following conditions is detected, a system error will occur. • If GCADI interrupt is defined to be unused (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED) with r_system_cfg.h. • If, in r_adc_cfg.h, the setting for S14AD_GCADI_PRIORITY exceeds the defined range.	
Remarks	-	

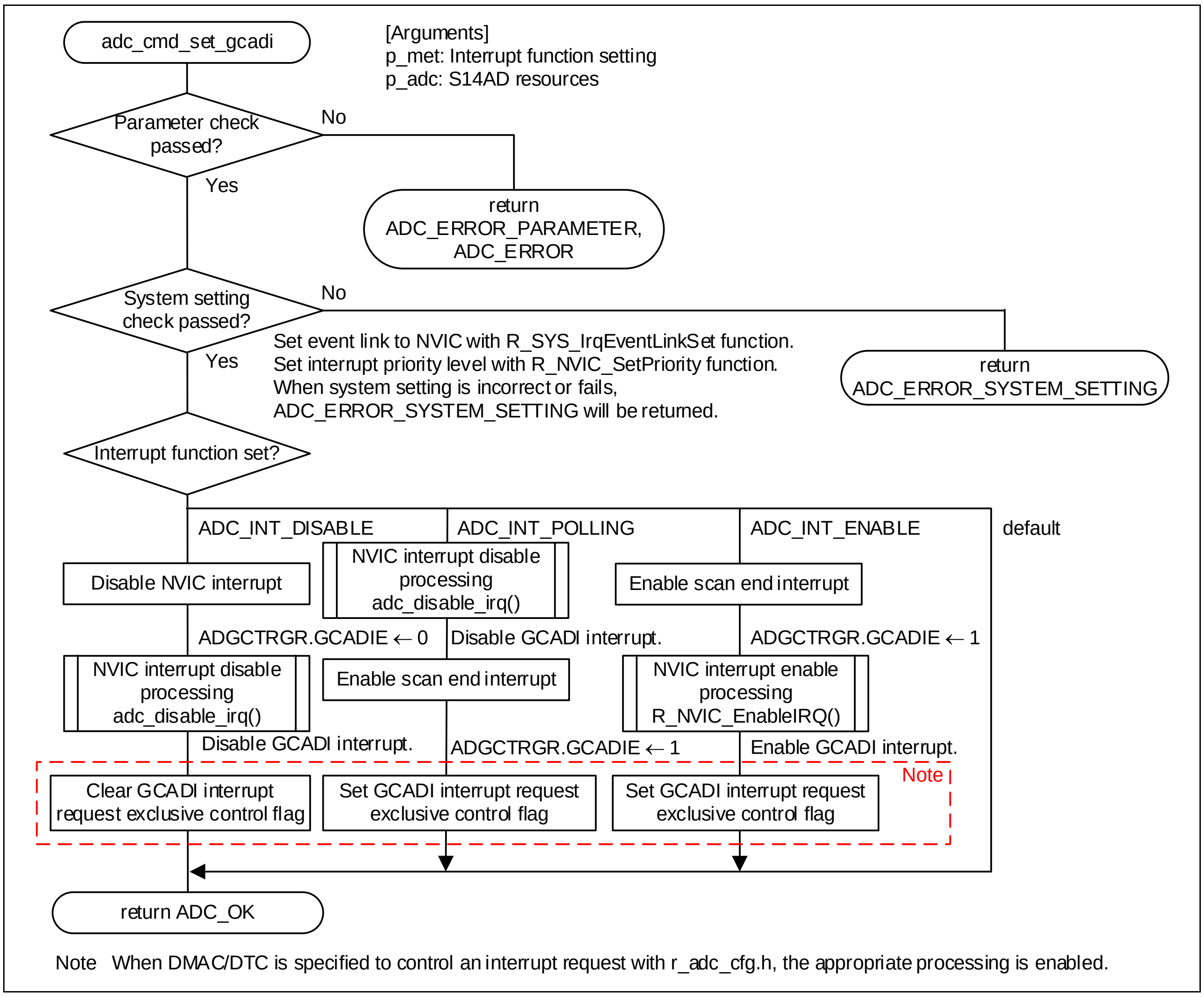

adc_cmd_set_gcadi
[Arguments]
p_met: Interrupt function setting
p_adc: S14AD resources
Parameter check passed?
No
Yes
return ADC_ERROR_PARAMETER, ADC_ERROR
System setting check passed?
No
Yes
Set event link to NVIC with R_SYS_IrqEventLinkSet function.
Set interrupt priority level with R_NVIC_SetPriority function.
When system setting is incorrect or fails,
ADC_ERROR_SYSTEM_SETTING will be returned.
return ADC_ERROR_SYSTEM_SETTING
Interrupt function set?
ADC_INT_DISABLE
ADC_INT_POLLING
ADC_INT_ENABLE
default
Disable NVIC interrupt
NVIC interrupt disable processing adc_disable_irq()
Enable scan end interrupt
ADGCTRGR.GCADIE ← 0
Disable GCADI interrupt.
ADGCTRGR.GCADIE ← 1
NVIC interrupt disable processing adc_disable_irq()
Enable scan end interrupt
NVIC interrupt enable processing R_NVIC_EnableIRQ()
Disable GCADI interrupt.
ADGCTRGR.GCADIE ← 1
Enable GCADI interrupt.
Note
Clear GCADI interrupt request exclusive control flag
Set GCADI interrupt request exclusive control flag
Set GCADI interrupt request exclusive control flag
return ADC_OK
Note When DMAC/DTC is specified to control an interrupt request with r_adc_cfg.h, the appropriate processing is enabled.

Figure 4-32 adc_cmd_set_gcadi Function Processing Flow

4.1.28 adc_cmd_set_wcmpm Function

Table 4-30 adc_cmd_set_wcmpm Function Specifications

Format	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)	
Description	Sets a comparison condition match interrupt of compare function window A/B (ADC140_WCMPM) (hereafter WCMPM interrupt).	
Argument	e_adc_int_method_t const * const p_met: Interrupt function setting Set one of the following. ADC_INT_DISABLE: Interrupt disable ADC_INT_POLLING: Polling ADC_INT_ENABLE: Interrupt enable	
	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.	
Return value	ADC_OK	WCMPM interrupt function set normally
	ADC_ERROR When a WCMPM interrupt is being used as a DMA transfer trigger for auto-read function, a WCMPM interrupt setting failure results	WCMPM interrupt function setting failed.
	ADC_ERROR_PARAMETER If a value other than those stipulated is specified for an argument, a parameter error will occur.	Parameter error
	ADC_ERROR_SYSTEM_SETTING If one of the following conditions is detected, a system error will occur. • If WCMPM interrupt is defined to be unused (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED) with r_system_cfg.h • If, in r_adc_cfg.h, the setting for S14AD_WCMPM_PRIORITY exceeds the defined range.	System error
Remarks	-	

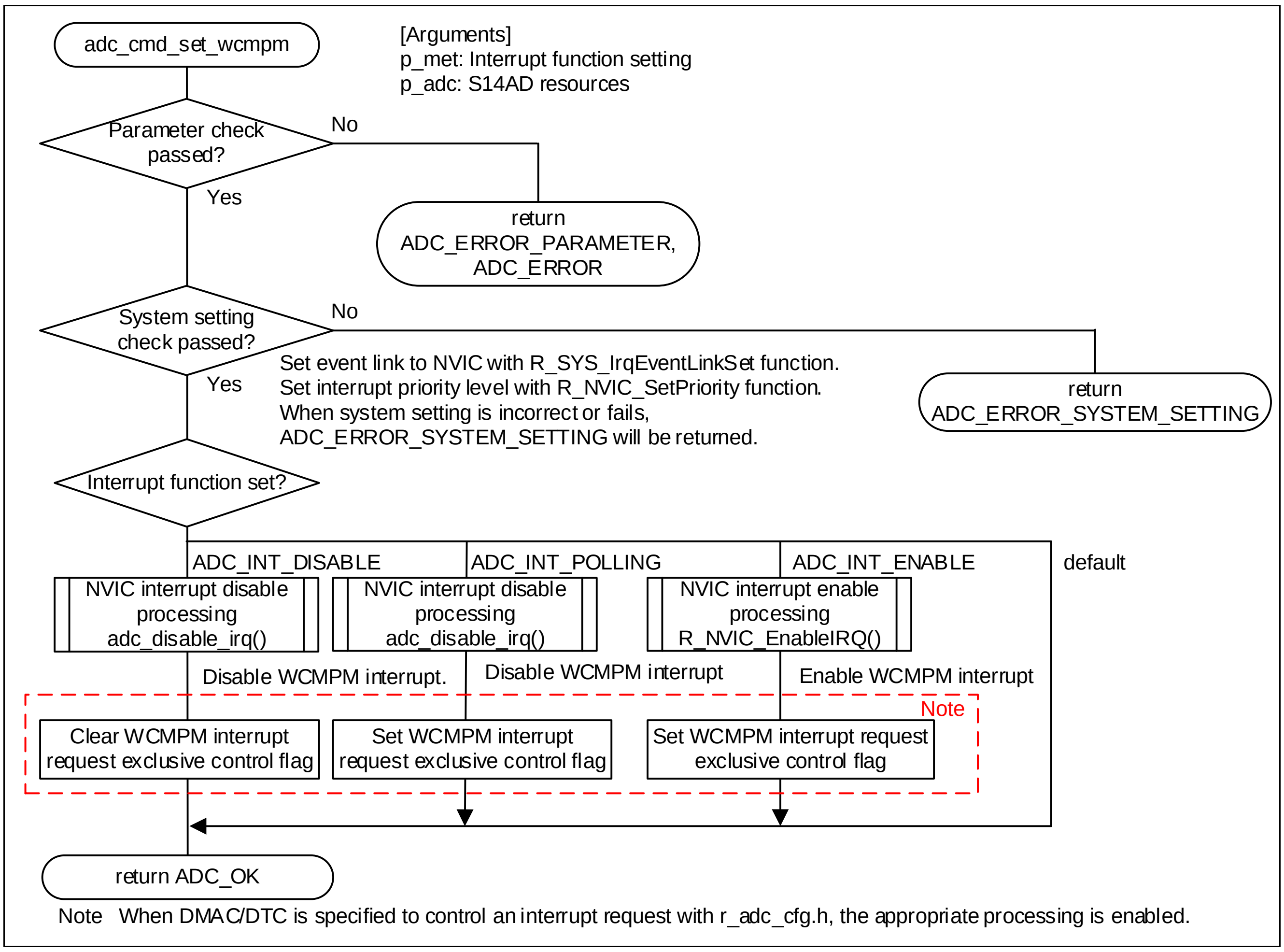
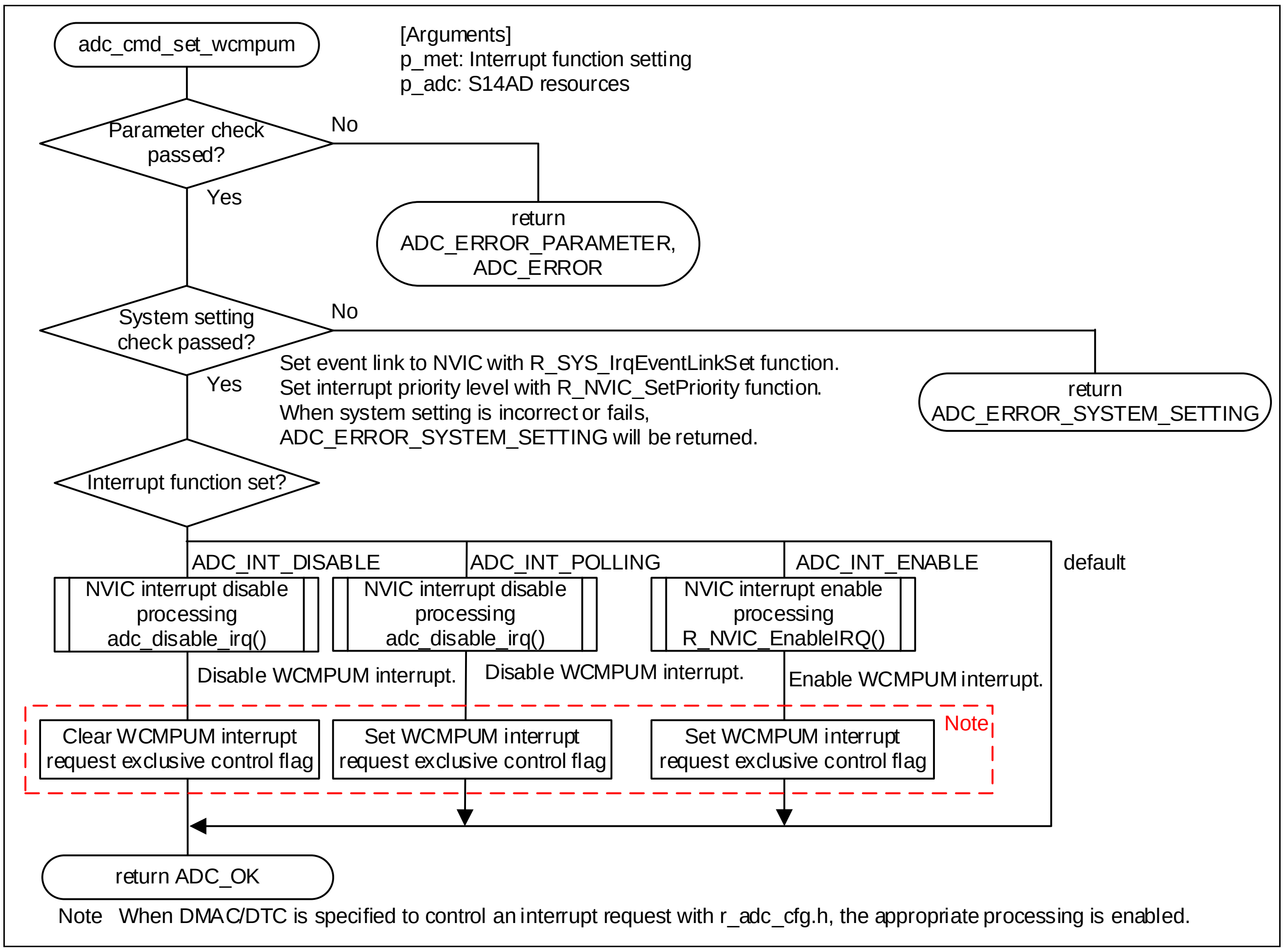

adc_cmd_set_wcmpm
[Arguments]
p_met: Interrupt function setting
p_adc: S14AD resources
Parameter check passed?
No
Yes
return ADC_ERROR_PARAMETER, ADC_ERROR
System setting check passed?
No
Yes
Set event link to NVIC with R_SYS_IrqEventLinkSet function.
Set interrupt priority level with R_NVIC_SetPriority function.
When system setting is incorrect or fails,
ADC_ERROR_SYSTEM_SETTING will be returned.
return ADC_ERROR_SYSTEM_SETTING
Interrupt function set?
ADC_INT_DISABLE
ADC_INT_POLLING
ADC_INT_ENABLE
default
NVIC interrupt disable processing adc_disable_irq()
NVIC interrupt disable processing adc_disable_irq()
NVIC interrupt enable processing R_NVIC_EnableIRQ()
Disable WCMPM interrupt.
Disable WCMPM interrupt
Enable WCMPM interrupt
Note
Clear WCMPM interrupt request exclusive control flag
Set WCMPM interrupt request exclusive control flag
Set WCMPM interrupt request exclusive control flag
return ADC_OK
Note When DMAC/DTC is specified to control an interrupt request with r_adc_cfg.h, the appropriate processing is enabled.

Figure 4-33 adc_cmd_set_wcmpm Function Processing Flow

4.1.29 adc_cmd_set_wcmpum Function

Table 4-31 adc_cmd_set_wcmpum Function Specifications

Format	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)	
Description	Sets a comparison condition mismatch interrupt of compare function window A/B(ADC140_WCMPUM) (hereafter WCMPUM interrupt)	
Argument	e_adc_int_method_t const * const p_met: Interrupt function setting Set one of the following. ADC_INT_DISABLE: Interrupt disable ADC_INT_POLLING: Polling ADC_INT_ENABLE: Interrupt enable	
	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.	
Return value	ADC_OK	WCMPUM interrupt function set normally
	ADC_ERROR When a WCMPUM interrupt is being used as a DMA transfer trigger for auto-read function, a WCMPUM interrupt setting failure results	WCMPUM interrupt function setting failed.
	ADC_ERROR_PARAMETER If a value other than those stipulated is specified for an argument, a parameter error will occur.	Parameter error
	ADC_ERROR_SYSTEM_SETTING If one of the following conditions is detected, a system error will occur. • If WCMPUM interrupt is defined to be unused (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED) with r_system_cfg.h • If, in r_adc_cfg.h, the setting for S14AD_WCMPUM_PRIORITY exceeds the defined range	System error
Remarks	-	


adc_cmd_set_wcmpum
[Arguments]
p_met: Interrupt function setting
p_adc: S14AD resources
Parameter check passed?
No
Yes
return ADC_ERROR_PARAMETER, ADC_ERROR
System setting check passed?
No
Yes
Set event link to NVIC with R_SYS_IrqEventLinkSet function.
Set interrupt priority level with R_NVIC_SetPriority function.
When system setting is incorrect or fails,
ADC_ERROR_SYSTEM_SETTING will be returned.
return ADC_ERROR_SYSTEM_SETTING
Interrupt function set?
ADC_INT_DISABLE
ADC_INT_POLLING
ADC_INT_ENABLE
default
NVIC interrupt disable processing adc_disable_irq()
NVIC interrupt disable processing adc_disable_irq()
NVIC interrupt enable processing R_NVIC_EnableIRQ()
Disable WCMPUM interrupt.
Disable WCMPUM interrupt.
Enable WCMPUM interrupt.
Note
Clear WCMPUM interrupt request exclusive control flag
Set WCMPUM interrupt request exclusive control flag
Set WCMPUM interrupt request exclusive control flag
return ADC_OK

Note When DMAC/DTC is specified to control an interrupt request with r_adc_cfg.h, the appropriate processing is enabled.

Figure 4-34 adc_cmd_set_wcmpum Function Processing Flow

4.1.30 adc_cmd_set_elc Function

Table 4-32 adc_cmd_set_elc Function Specifications

Format	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)	
Description	Acquires the comparison results of compare function.	
Argument	e_adc_elc_mode_t const * const p_elc: Scan end event occurrence condition setting Set one of the following. ADC_ELC_GROUPA: Event occurrence at the time of scan end, excluding scan end of group B and group C. ADC_ELC_GROUPB: Event occurrence at the time of scan end of group B. ADC_ELC_GROUPC: Event occurrence at the time of scan end of group C. ADC_ELC_ALL: Event occurrence at the time of scan end of all the groups.	
	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.	
Return value	ADC_OK	Scan end event occurrence condition set normally
	ADC_ERROR_PARAMETER If one of the following conditions is detected, a parameter error will occur. • If an argument is NULL. • If a value other than those stipulated is specified for an argument.	Parameter error
Remarks	-	

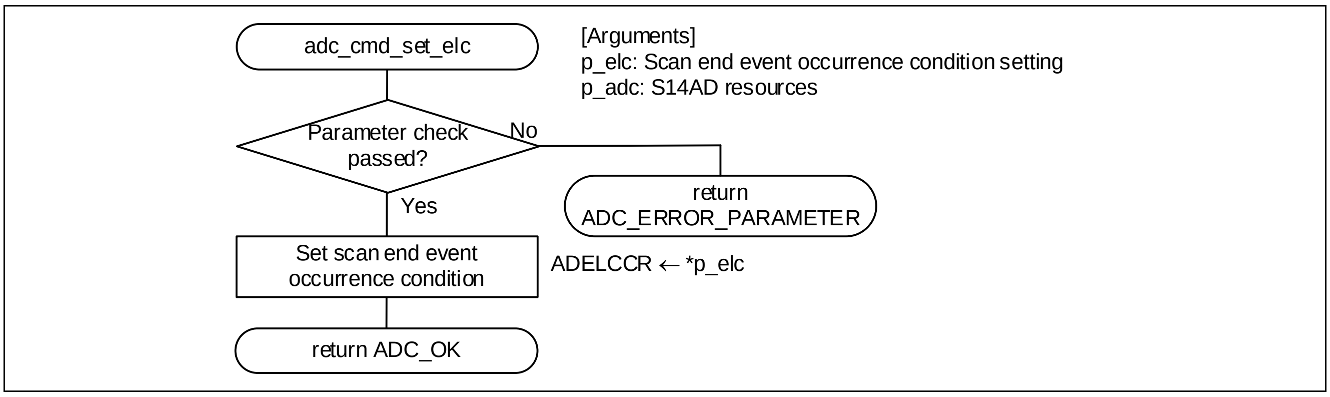

adc_cmd_set_elc
[Arguments]
p_elc: Scan end event occurrence condition setting
p_adc: S14AD resources
Parameter check passed?
No
Yes
return ADC_ERROR_PARAMETER
Set scan end event occurrence condition
ADELCCR ← *p_elc
return ADC_OK

Figure 4-35 adc_cmd_set_elc Function Processing Flow

4.1.31 adc_cmd_stop_trigger Function

Table 4-33 adc_cmd_stop_trigger Function Specifications

Format	static e_adc_err_t adc_cmd_stop_trigger(st_adc_resources_t * const p_adc)	
Description	Cancels the A/D conversion start trigger setting for all groups and stops A/D conversion.	
Argument	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.	
Return value	ADC_OK	A/D conversion stopped normally.
	ADC_ERROR	A/D conversion stop failed. If initialization using the ScanSet function has not be completed, A/D conversion stop failed.
Remarks	-	

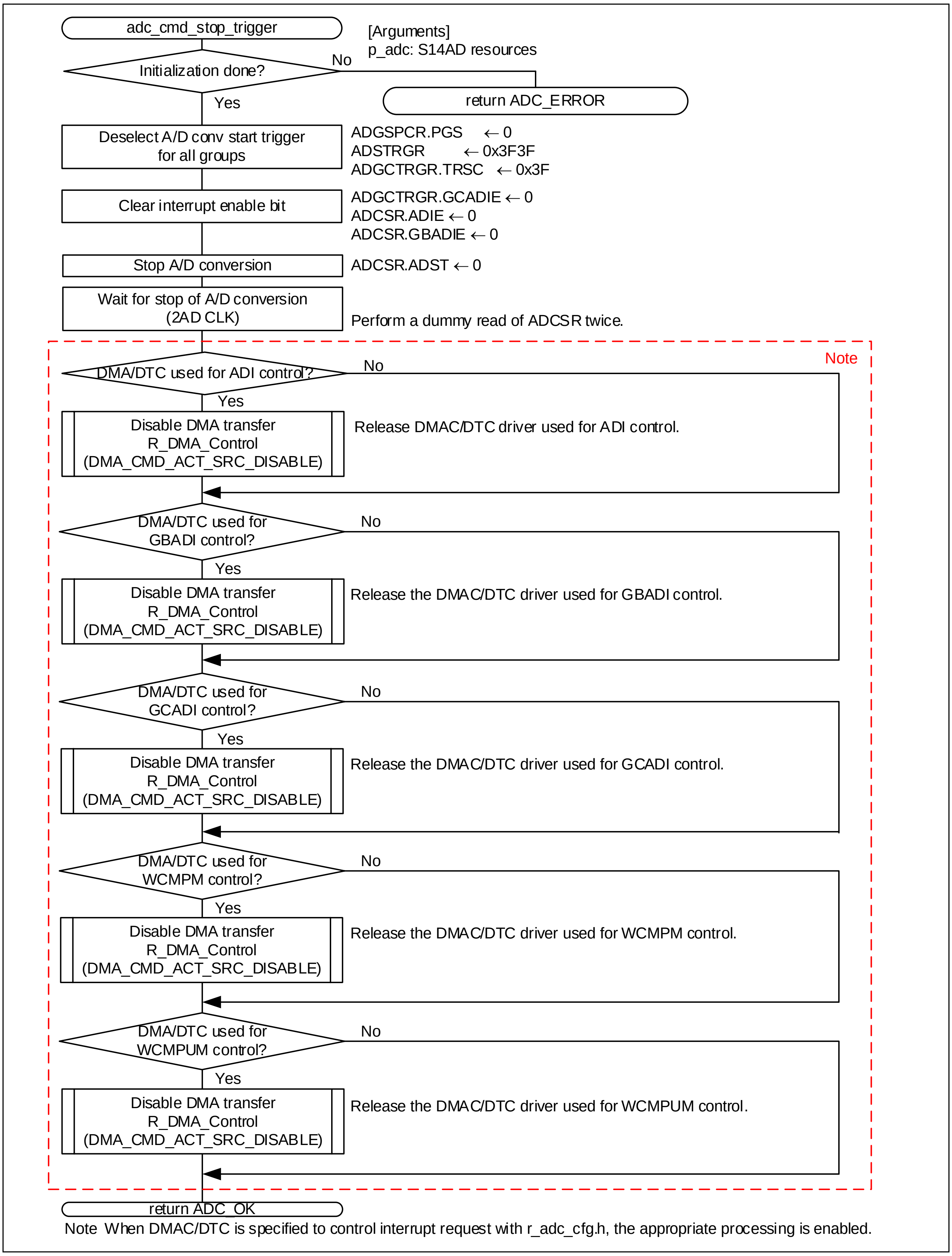

adc_cmd_stop_trigger
[Arguments]
p_adc: S14AD resources
Initialization done?
No
Yes
return ADC_ERROR
Deselect A/D conv start trigger for all groups
ADGSPCR.PGS ← 0
ADSTRGR ← 0x3F3F
ADGCTRGR.TRSC ← 0x3F
Clear interrupt enable bit
ADGCTRGR.GCADIE ← 0
ADCSR.ADIE ← 0
ADCSR.GBADIE ← 0
Stop A/D conversion
ADCSR.ADST ← 0
Wait for stop of A/D conversion (2AD CLK)
Perform a dummy read of ADCSR twice.
Note
DMA/DTC used for ADI control?
No
Yes
Disable DMA transfer R_DMA_Control (DMA_CMD_ACT_SRC_DISABLE)
Release DMAC/DTC driver used for ADI control.
DMA/DTC used for GBADI control?
No
Yes
Disable DMA transfer R_DMA_Control (DMA_CMD_ACT_SRC_DISABLE)
Release the DMAC/DTC driver used for GBADI control.
DMA/DTC used for GCADI control?
No
Yes
Disable DMA transfer R_DMA_Control (DMA_CMD_ACT_SRC_DISABLE)
Release the DMAC/DTC driver used for GCADI control.
DMA/DTC used for WCMPM control?
No
Yes
Disable DMA transfer R_DMA_Control (DMA_CMD_ACT_SRC_DISABLE)
Release the DMAC/DTC driver used for WCMPM control.
DMA/DTC used for WCMPUM control?
No
Yes
Disable DMA transfer R_DMA_Control (DMA_CMD_ACT_SRC_DISABLE)
Release the DMAC/DTC driver used for WCMPUM control.
return ADC_OK
Note When DMAC/DTC is specified to control interrupt request with r_adc_cfg.h, the appropriate processing is enabled.

Figure 4-36 adc_cmd_stop_trigger Function Processing Flow

4.1.32 adc_cmd_auto_read Function

Table 4-34 adc_cmd_auto_read Function Specifications

Format	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)
Description	Specifies auto-read function settings
Argument	st_adc_dma_read_info_t const * const p_dma: Auto-read function setting
	e_adc_dma_cmd_t cmd: Auto-read command
	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.
Return value	ADC_OK Auto-read function set normally
	ADC_ERROR Auto-read function setting failed. When an interrupt request specified as a DMA transfer trigger is used for an interrupt purpose (interrupt or polling), an auto-read function setting will fail.
	ADC_ERROR_SYSTEM_SETTING System error If one of the following conditions is detected, a system error will occur. [Common] • If, in r_adc_cfg.h, an interrupt request control (S14AD_xxx_CONTROL) specified as a DMA transfer trigger is an interrupt setting (S14AD_USED_INTERRUPT) (xxx = ADI, GBADI, GCADI, WCMPM, WCMPUM). [DTC use] • If , in r_system_cfg.h, DMA transfer trigger interrupt is defined to be unused (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED). • If, in r_adc_cfg.h, the DMA transfer trigger interrupt priority order setting exceeds the defined range. [DMAC use] • If a callback function has been specified, and, in r_system_cfg.h, a DMACn_INT (n = 0 to 3) interrupt is defined to be unused (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED) • If a callback function has been specified, and, in r_dma_api_cfg.h, the DMACn_INT_PRIORITY setting exceeds the defined range
	ADC_ERROR_PARAMETER Parameter error If one of the following conditions is detected, a parameter error will occur. • If a value other than those stipulated is specified for an argument. • If an ADI, GBADI or GCADI interrupt request is specified as an argument • DMA transfer trigger for auto-read function (compare). • If a WCMPM or WCMPUM interrupt request is specified as a DMA transfer trigger for other than auto-read function (compare). • If there is an incorrect transfer size setting for auto-read function (compare)
Remarks	Valid only when, in r_adc_cfg.h, DMAC/DTC is specified for interrupt request control

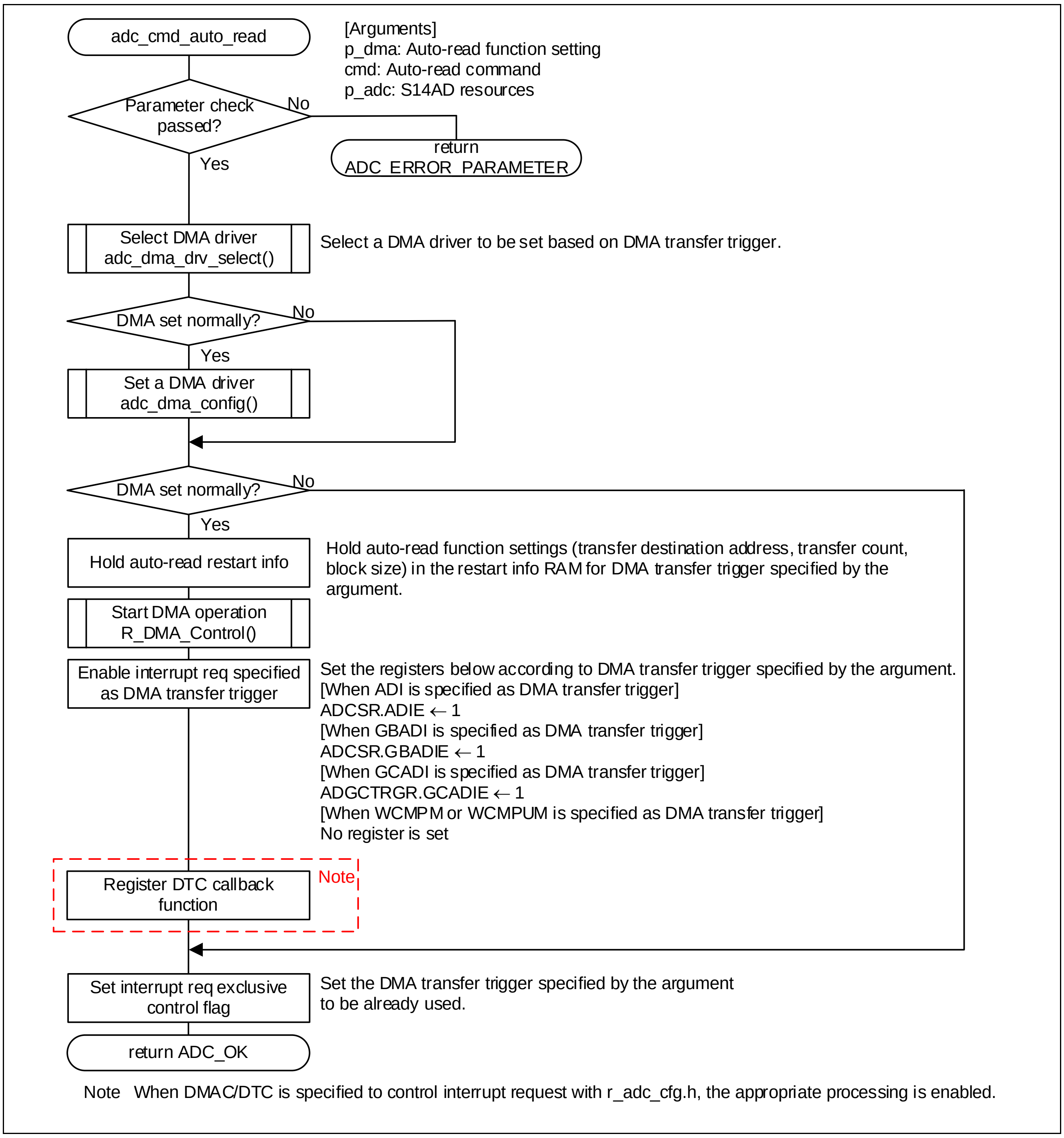

adc_cmd_auto_read
[Arguments]
p_dma: Auto-read function setting
cmd: Auto-read command
p_adc: S14AD resources
Parameter check passed?
No
Yes
return ADC_ERROR_PARAMETER
Select DMA driver adc_dma_drv_select()
Select a DMA driver to be set based on DMA transfer trigger.
DMA set normally?
No
Yes
Set a DMA driver adc_dma_config()
DMA set normally?
No
Yes
Hold auto-read restart info
Hold auto-read function settings (transfer destination address, transfer count, block size) in the restart info RAM for DMA transfer trigger specified by the argument.
Start DMA operation R_DMA_Control()
Enable interrupt req specified as DMA transfer trigger
Set the registers below according to DMA transfer trigger specified by the argument.
[When ADI is specified as DMA transfer trigger]
ADCSR.ADIE ← 1
[When GBADI is specified as DMA transfer trigger]
ADCSR.GBADIE ← 1
[When GCADI is specified as DMA transfer trigger]
ADGCTRGR.GCADIE ← 1
[When WCMPM or WCMPUM is specified as DMA transfer trigger]
No register is set
Register DTC callback function
Note
Set interrupt req exclusive control flag
Set the DMA transfer trigger specified by the argument to be already used.
return ADC_OK
Note When DMAC/DTC is specified to control interrupt request with r_adc_cfg.h, the appropriate processing is enabled.

Figure 4-37 adc_cmd_auto_read Function Processing Flow

4.1.33 adc_cmd_auto_read_stop Function

Table 4-35 adc_cmd_auto_read_stop Function Specifications

Format	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)	
Description	Stops auto read according to a specified DMA transfer trigger.	
Argument	e_adc_dma_event_t const * const event: DMA transfer trigger	
	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.	
Return value	ADC_OK	Auto read stopped normally.
	ADC_ERROR_SYSTEM_SETTING • If, in r_adc_cfg.h, an interrupt request control (S14AD_xxx_CONTROL) specified as a DMA transfer trigger is an interrupt setting (S14AD_USED_INTERRUPT) (xxx = ADI, GBADI, GCADI, WCMPM, WCMPUM), a system error will occur.	System error
	ADC_ERROR_PARAMETER If a value other than those stipulated is specified for an argument, a parameter error will occur.	Parameter error
Remarks	Valid only when, in r_adc_cfg.h, DMAC/DTC is specified for interrupt request control.	

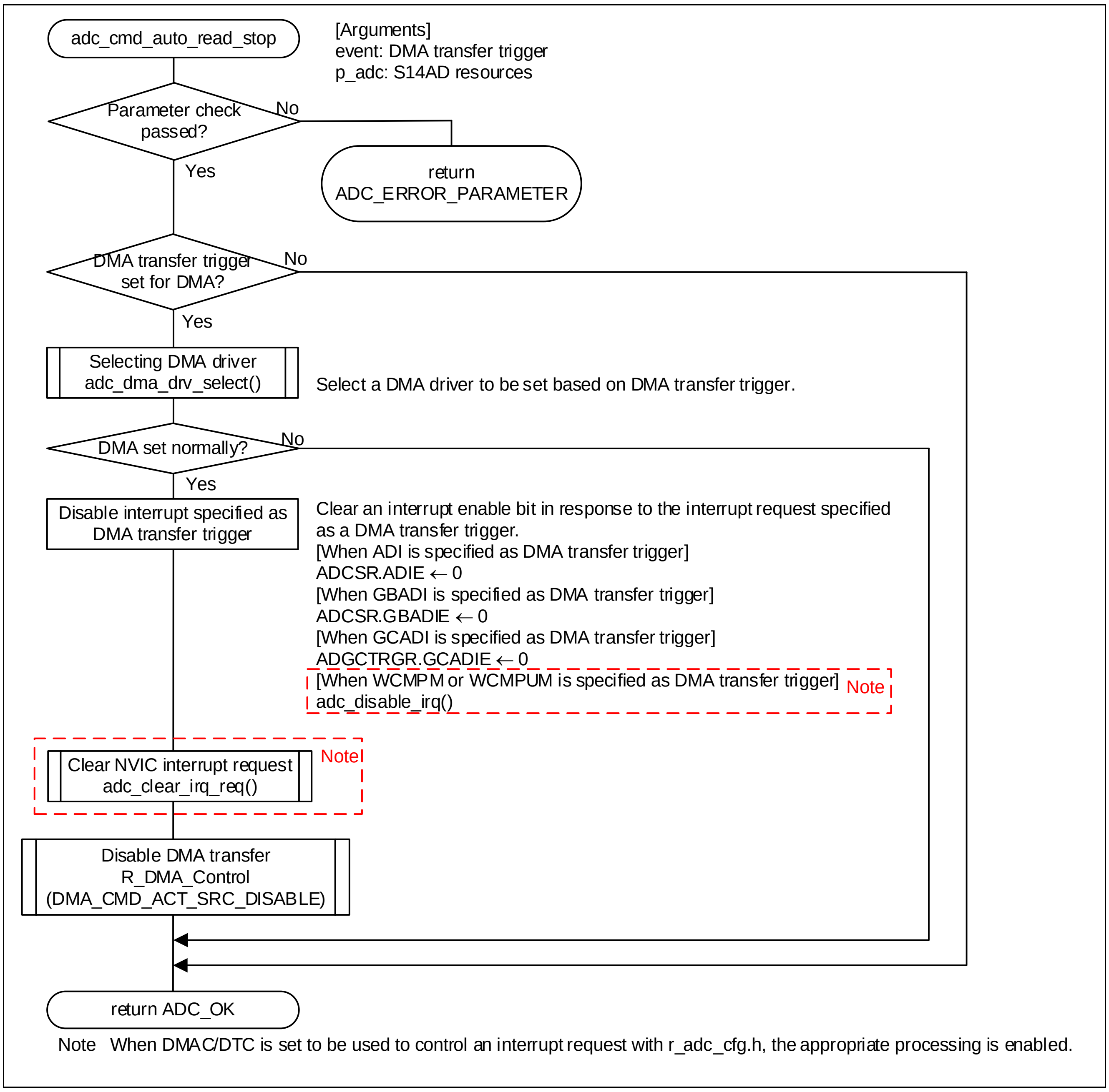
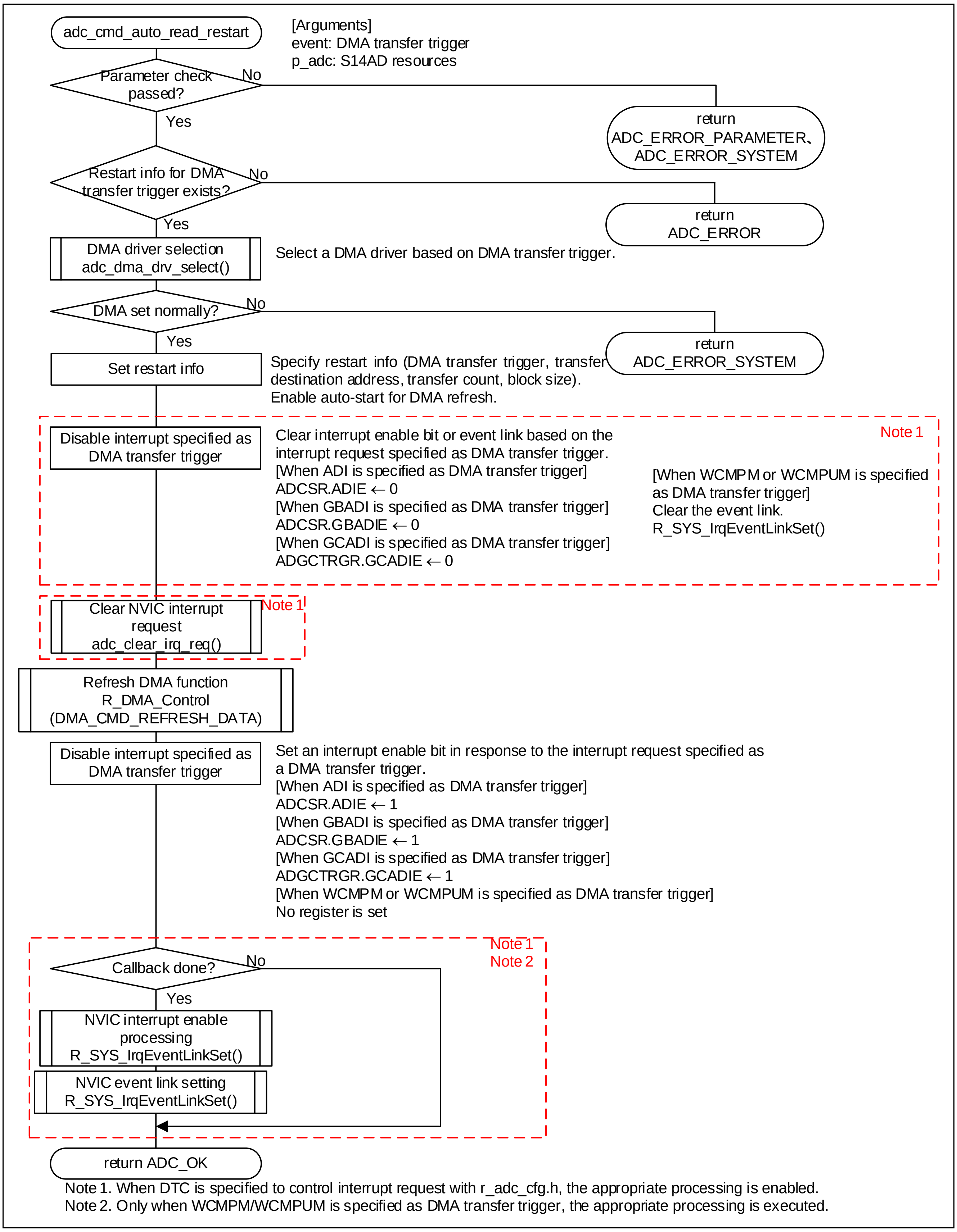

adc_cmd_auto_read_stop
[Arguments]
event: DMA transfer trigger
p_adc: S14AD resources
Parameter check passed?
No
Yes
return ADC_ERROR_PARAMETER
DMA transfer trigger set for DMA?
No
Yes
Selecting DMA driver adc_dma_drv_select()
Select a DMA driver to be set based on DMA transfer trigger.
DMA set normally?
No
Yes
Disable interrupt specified as DMA transfer trigger
Clear an interrupt enable bit in response to the interrupt request specified as a DMA transfer trigger.
[When ADI is specified as DMA transfer trigger]
ADCSR.ADIE ← 0
[When GBADI is specified as DMA transfer trigger]
ADCSR.GBADIE ← 0
[When GCADI is specified as DMA transfer trigger]
ADGCTRGR.GCADIE ← 0
[When WCMPM or WCMPUM is specified as DMA transfer trigger]
adc_disable_irq()
Note
Clear NVIC interrupt request adc_clear_irq_req()
Note
Disable DMA transfer R_DMA_Control (DMA_CMD_ACT_SRC_DISABLE)
return ADC_OK
Note When DMAC/DTC is set to be used to control an interrupt request with r_adc_cfg.h, the appropriate processing is enabled.

Figure 4-38 adc_cmd_auto_read_stop Function Processing Flow

4.1.34 adc_cmd_auto_read_restart Function

Table 4-36 adc_cmd_auto_read_restart Function Specifications

Format	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)	
Description	Stops auto read according to a specified DMA transfer trigger.	
Argument	e_adc_dma_event_t const * const event: DMA transfer trigger	
	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.	
Return value	ADC_OK	Auto read stopped normally.
	ADC_ERROR_SYSTEM_SETTING	System error
	If one of the following conditions is detected, a system error will occur. • When WCMPM is specified as a DMA transfer trigger, in r_system_cfg.h, WCMPM interrupt is defined to be unused (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED). • When WCMPUM is specified as a DMA transfer trigger, in r_system_cfg.h, WCMPUM interrupt is defined to be unused (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED). • If, in r_adc_cfg.h, an interrupt request control (S14AD_xxx_CONTROL) specified as a DMA transfer trigger is an interrupt setting (S14AD_USED_INTERRUPT) (xxx = ADI, GBADI, GCADI, WCMPM, WCMPUM).	
	ADC_ERROR_PARAMETER	Parameter error
	If a value other than those stipulated is specified for an argument, a parameter error will occur.	
Remarks	Valid only when, in r_adc_cfg.h, DMAC/DTC is specified for interrupt request control.	


adc_cmd_auto_read_restart
[Arguments]
event: DMA transfer trigger
p_adc: S14AD resources
Parameter check passed?
No
Yes
return ADC_ERROR_PARAMETER、ADC_ERROR_SYSTEM
Restart info for DMA transfer trigger exists?
No
Yes
return ADC_ERROR
DMA driver selection adc_dma_drv_select()
Select a DMA driver based on DMA transfer trigger.
DMA set normally?
No
Yes
return ADC_ERROR_SYSTEM
Set restart info
Specify restart info (DMA transfer trigger, transfer destination address, transfer count, block size).
Enable auto-start for DMA refresh.
Note 1
Disable interrupt specified as DMA transfer trigger
Clear interrupt enable bit or event link based on the interrupt request specified as DMA transfer trigger.
[When ADI is specified as DMA transfer trigger]
ADCSR.ADIE ← 0
[When GBADI is specified as DMA transfer trigger]
ADCSR.GBADIE ← 0
[When GCADI is specified as DMA transfer trigger]
ADGCTRGR.GCADIE ← 0
[When WCMPM or WCMPUM is specified as DMA transfer trigger]
Clear the event link.
R_SYS_IrqEventLinkSet()
Clear NVIC interrupt request adc_clear_irq_req()
Note 1
Refresh DMA function R_DMA_Control (DMA_CMD_REFRESH_DATA)
Disable interrupt specified as DMA transfer trigger
Set an interrupt enable bit in response to the interrupt request specified as a DMA transfer trigger.
[When ADI is specified as DMA transfer trigger]
ADCSR.ADIE ← 1
[When GBADI is specified as DMA transfer trigger]
ADCSR.GBADIE ← 1
[When GCADI is specified as DMA transfer trigger]
ADGCTRGR.GCADIE ← 1
[When WCMPM or WCMPUM is specified as DMA transfer trigger]
No register is set
Note 1
Note 2
Callback done?
No
Yes
NVIC interrupt enable processing R_SYS_IrqEventLinkSet()
NVIC event link setting R_SYS_IrqEventLinkSet()
return ADC_OK

Note 1. When DTC is specified to control interrupt request with r_adc_cfg.h, the appropriate processing is enabled.
Note 2. Only when WCMPM/WCMPUM is specified as DMA transfer trigger, the appropriate processing is executed.

Figure 4-39 adc_cmd_auto_read_restart Function Processing Flow

4.1.35 adc_dma_ram_init Function

Table 4-37 adc_dma_ram_init Function Specifications

Format	void adc_dma_ram_init(st_adc_resources_t * const p_adc)
Description	Initializes the RAM used for auto read.
Argument	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.
Return value	-
Remarks	Valid only when, in r_adc_cfg.h, DMAC/DTC is specified for interrupt request control

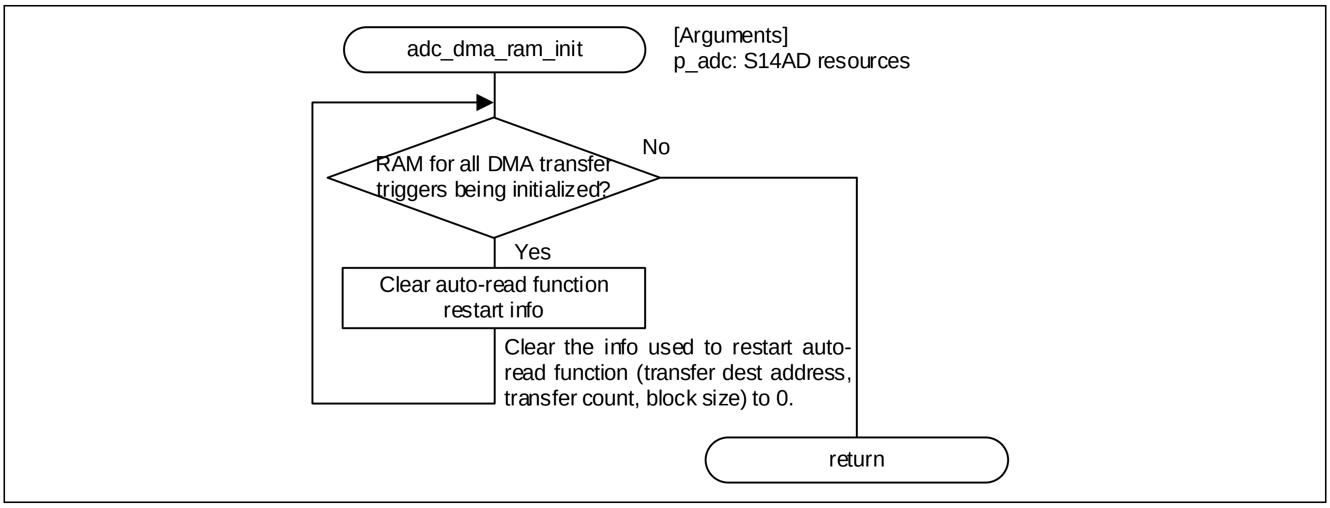

adc_dma_ram_init
[Arguments]
p_adc: S14AD resources
RAM for all DMA transfer triggers being initialized?
No
Yes
Clear auto-read function restart info
Clear the info used to restart auto-read function (transfer dest address, transfer count, block size) to 0.
return

Figure 4-40 adc_dma_ram_init Function Processing Flow

4.1.36 adc_dma_drv_select Function

Table 4-38 adc_dma_drv_select Function Specifications

Format	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)	
Description	Selects a DMA driver for the specified DMA transfer trigger.	
Argument	st_adc_dma_set * const setting: DMA transfer setting	
	e_adc_dma_event_t const * const event: DMA transfer trigger	
	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.	
Return value	ADC_OK	Auto read stopped normally.
	ADC_ERROR_SYSTEM_SETTING • If, in r_adc_cfg.h, an interrupt request control (S14AD_xxx_CONTROL) specified as a DMA transfer trigger is an interrupt setting (S14AD_USED_INTERRUPT) (xxx = ADI, GBADI, GCADI, WCMPM, WCMPUM), a system error will occur.	System error
Remarks	Valid only when, in r_adc_cfg.h, DMAC/DTC is specified for interrupt request control	

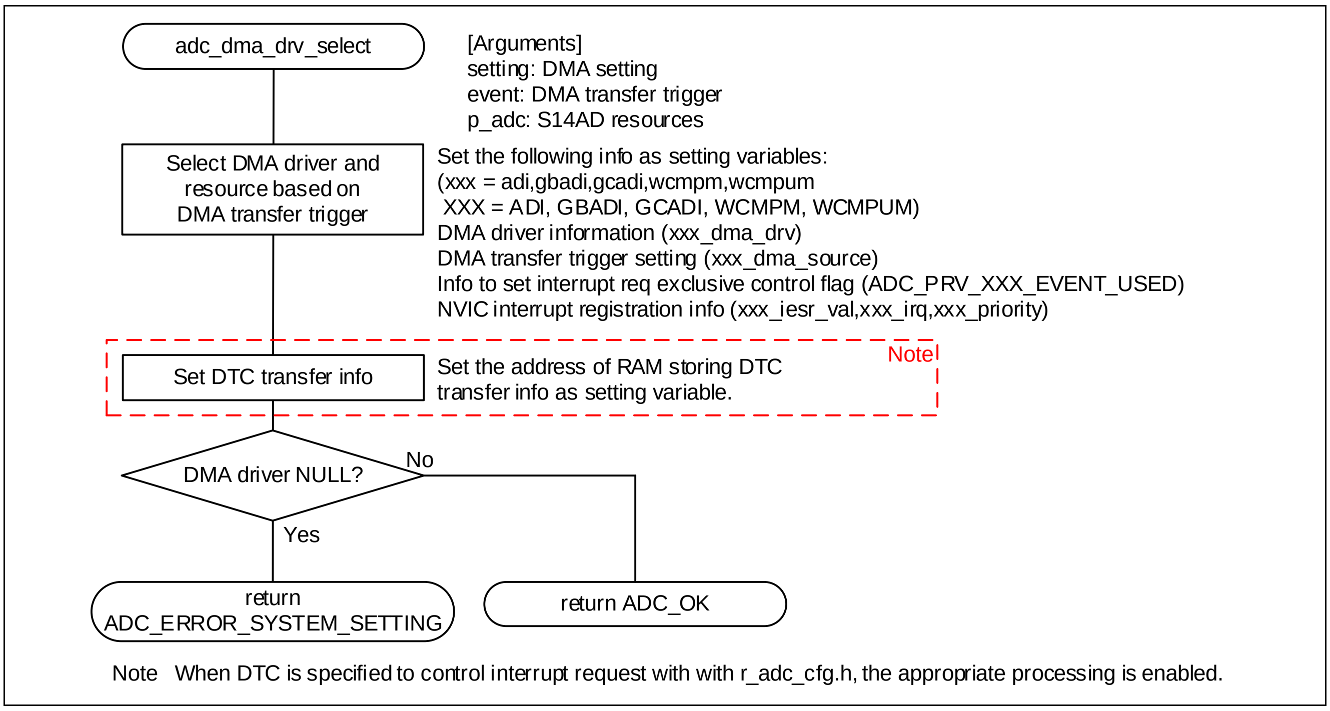

adc_dma_drv_select
[Arguments]
setting: DMA setting
event: DMA transfer trigger
p_adc: S14AD resources
Select DMA driver and resource based on DMA transfer trigger
Set the following info as setting variables:
(xxx = adi,gbadi,gcadi,wcmpm,wcmpum
XXX = ADI, GBADI, GCADI, WCMPM, WCMPUM)
DMA driver information (xxx_dma_drv)
DMA transfer trigger setting (xxx_dma_source)
Info to set interrupt req exclusive control flag (ADC_PRV_XXX_EVENT_USED)
NVIC interrupt registration info (xxx_iesr_val,xxx_irq,xxx_priority)
Set DTC transfer info
Set the address of RAM storing DTC transfer info as setting variable.
Note
DMA driver NULL?
No
Yes
return ADC_ERROR_SYSTEM_SETTING
return ADC_OK
Note When DTC is specified to control interrupt request with with r_adc_cfg.h, the appropriate processing is enabled.

Figure 4-41 adc_dma_drv_select Function Processing Flow

4.1.37 adc_dma_config Function

Table 4-39 adc_dma_config Function Specifications

Format	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)	
Description	Initializes a DMR driver to be used for auto-read function.	
Argument	st_adc_dma_read_info_t const * const p_dma: Auto-read function setting	
	st_adc_dma_set const * const setting: DMA transfer setting	
	e_adc_dma_cmd_t cmd:	
	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.	
Return value	ADC_OK	Auto-read function set normally
	ADC_ERROR When an interrupt request specified as a DMA transfer trigger is used for an interrupt purpose (interrupt or polling), an auto-read setting failure results	Auto-read function setting failed.
	ADC_ERROR_SYSTEM_SETTING If one of the following conditions is detected, a system error will occur. [Common] • If, in r_adc_cfg.h, an interrupt request control (S14AD_xxx_CONTROL) specified as a DMA transfer trigger is an interrupt setting (S14AD_USED_INTERRUPT) (xxx = ADI, GBADI, GCADI, WCMPM, WCMPUM). [DTC use] • If , in r_system_cfg.h, DMA transfer trigger interrupt is defined to be unused (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED). • If, in r_adc_cfg.h, the DMA transfer trigger interrupt priority order setting exceeds the defined range. [DMAC use] • If a callback function has been specified, and, in r_system_cfg.h, a DMACn_INT (n = 0 to 3) interrupt is defined to be unused (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED). • If a callback function has been specified, and, in r_dma_api_cfg.h, the DMACn_INT_PRIORITY setting exceeds the defined range	System error
	ADC_ERROR_PARAMETER If one of the following conditions is detected, a parameter error will occur. • If a value other than those stipulated is specified for an argument. • If an ADI, GBADI or GCADI interrupt request is specified as a DMA transfer trigger for auto-read function (compare). • If a WCMPM or WCMPUM interrupt request is specified as a DMA transfer trigger for other than auto-read function (compare). • If there is an incorrect transfer size setting for auto-read function (compare)	Parameter error
Remarks	Valid only when, in r_adc_cfg.h, DMAC/DTC is specified for interrupt request control.	

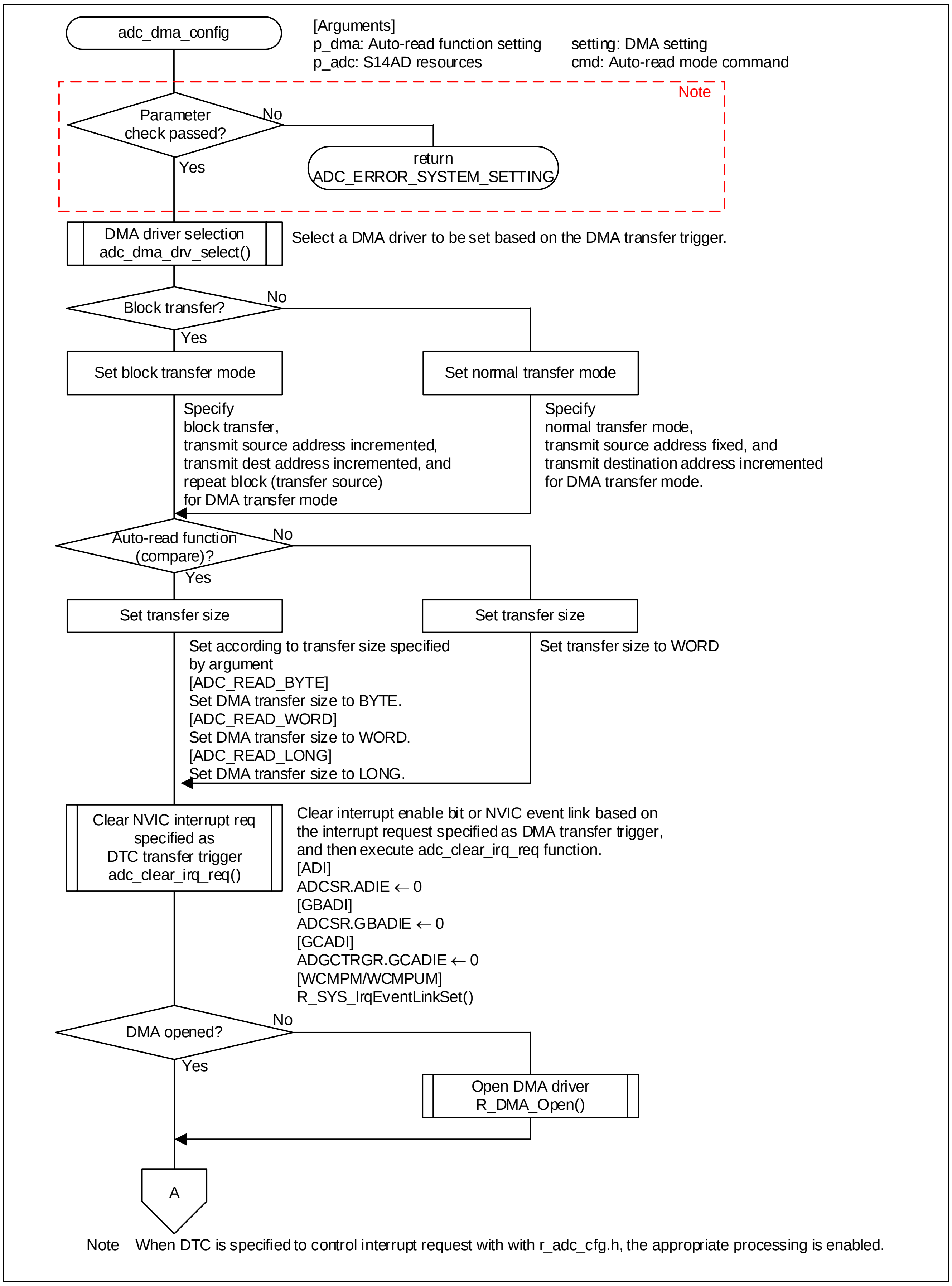
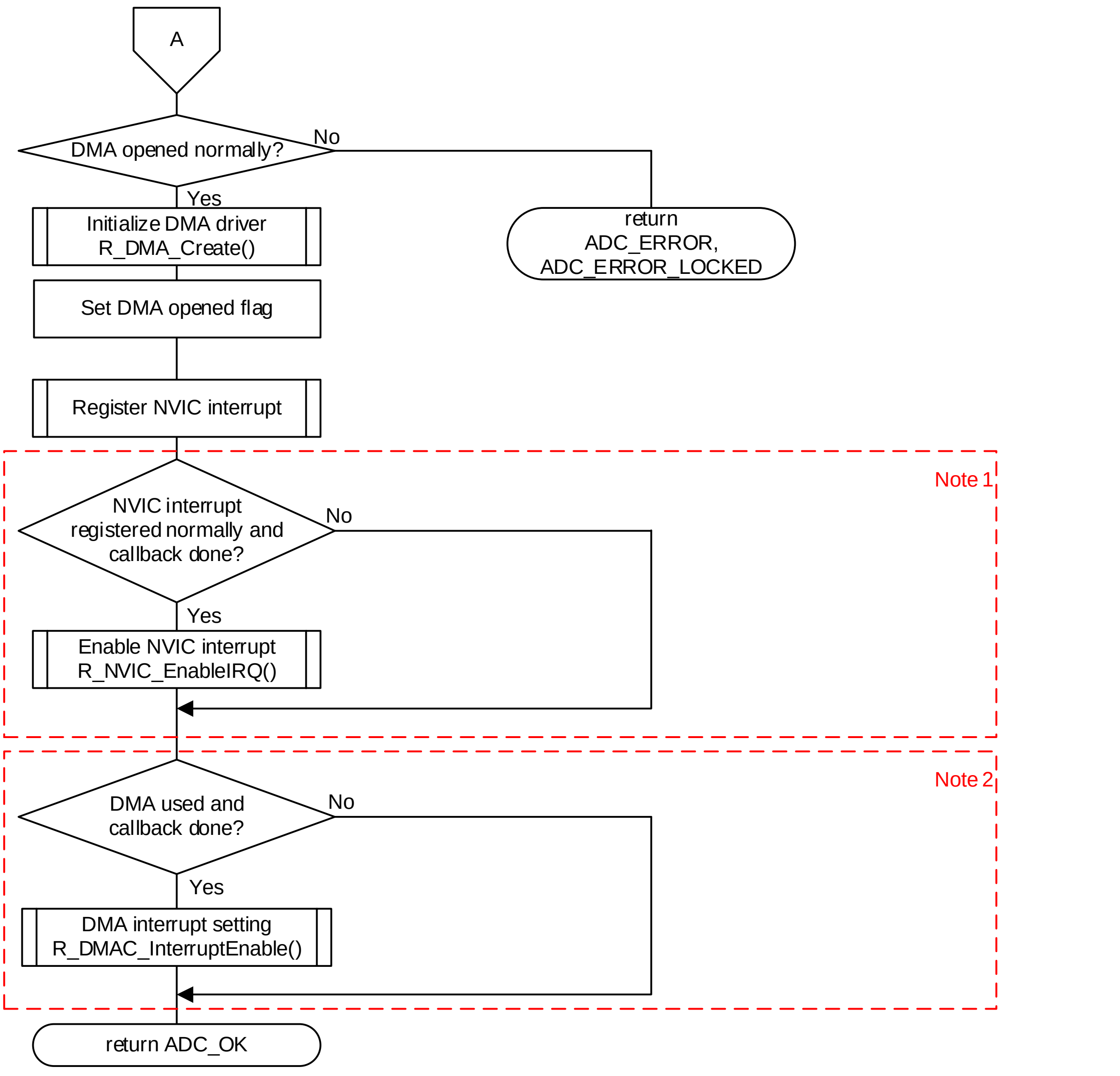

adc_dma_config
[Arguments]
p_dma: Auto-read function setting
p_adc: S14AD resources
setting: DMA setting
cmd: Auto-read mode command
Note
Parameter check passed?
No
Yes
return ADC_ERROR_SYSTEM_SETTING
DMA driver selection adc_dma_drv_select()
Select a DMA driver to be set based on the DMA transfer trigger.
Block transfer?
No
Yes
Set block transfer mode
Set normal transfer mode
Specify block transfer, transmit source address incremented, transmit dest address incremented, and repeat block (transfer source) for DMA transfer mode
Specify normal transfer mode, transmit source address fixed, and transmit destination address incremented for DMA transfer mode.
Auto-read function (compare)?
No
Yes
Set transfer size
Set transfer size
Set according to transfer size specified by argument
[ADC_READ_BYTE]
Set DMA transfer size to BYTE.
[ADC_READ_WORD]
Set DMA transfer size to WORD.
[ADC_READ_LONG]
Set DMA transfer size to LONG.
Set transfer size to WORD
Clear NVIC interrupt req specified as DTC transfer trigger adc_clear_irq_req()
Clear interrupt enable bit or NVIC event link based on the interrupt request specified as DMA transfer trigger, and then execute adc_clear_irq_req function.
[ADI]
ADCSR.ADIE ← 0
[GBADI]
ADCSR.GBADIE ← 0
[GCADI]
ADGCTRGR.GCADIE ← 0
[WCMPM/WCMPUM]
R_SYS_IrqEventLinkSet()
DMA opened?
No
Yes
Open DMA driver R_DMA_Open()
A
Note When DTC is specified to control interrupt request with with r_adc_cfg.h, the appropriate processing is enabled.

Figure 4-42 adc_dma_config Function Processing Flow (1/2)



Note 1. When DTC is specified to control interrupt request with with r_adc_cfg.h, the appropriate processing is enabled.
Note 2. When DMA is specified to control interrupt request with with r_adc_cfg.h, the appropriate processing is enabled.

Figure 4-43 adc_dma_config Function Processing Flow (2/2)

4.1.38 cadc_s14adi0_isr Function

Table 4-40 adc_s14adi0_isr Function Specifications

Format	static void adc_s14adi0_isr(st_adc_resources_t * const p_adc)
Description	ADI interrupt processing
Argument	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.
Return value	None
Remarks	-

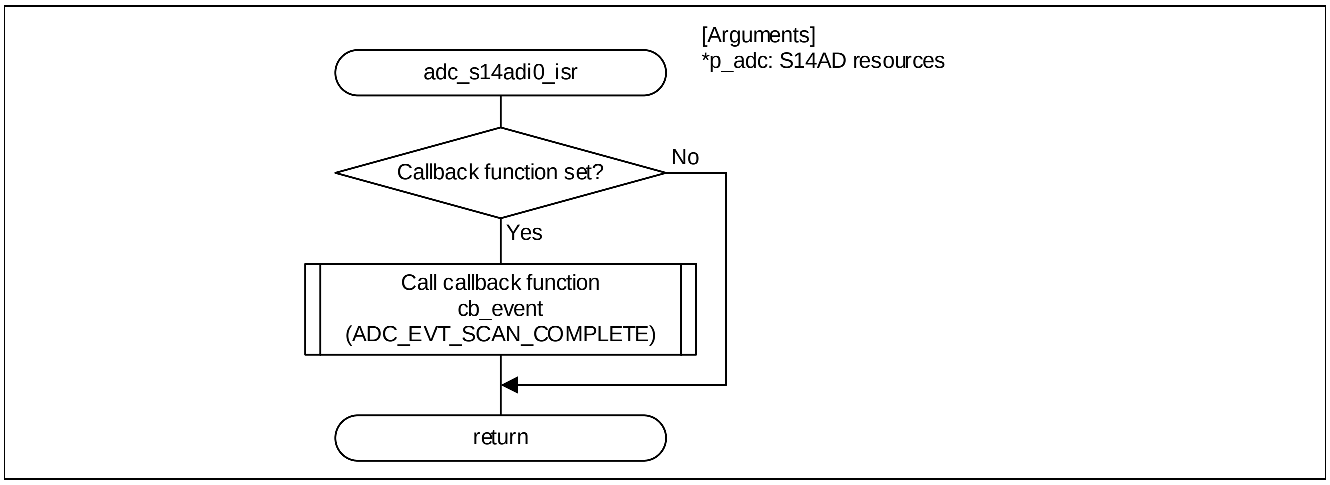


Figure 4-44 adc_s14adi0_isr Function Processing Flow

4.1.39 adc_gbadi_isr Function

Table 4-41 adc_gbadi_isr Function Specifications

Format	static void adc_gbadi_isr(st_adc_resources_t * const p_adc)
Description	GBADI interrupt processing
Argument	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.
Return value	None
Remarks	-

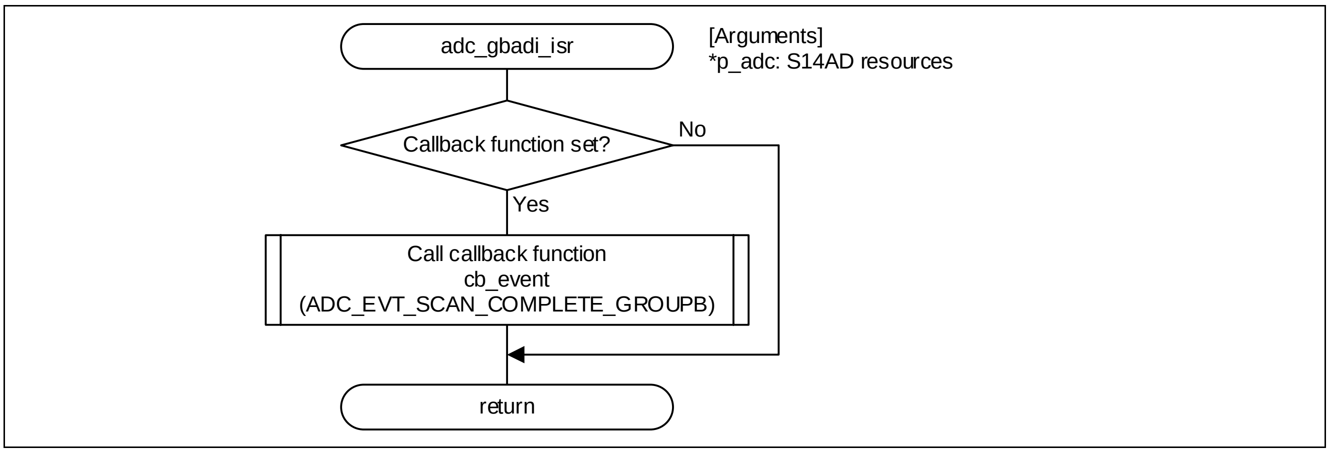


Figure 4-45 adc_gbadi_isr Function Processing Flow

4.1.40 adc_gcadi_isr Function

Table 4-42 adc_gcadi_isr Function Specifications

Format	static void adc_gcadi_isr(st_adc_resources_t * const p_adc)
Description	GCADI interrupt processing
Argument	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.
Return value	None
Remarks	-

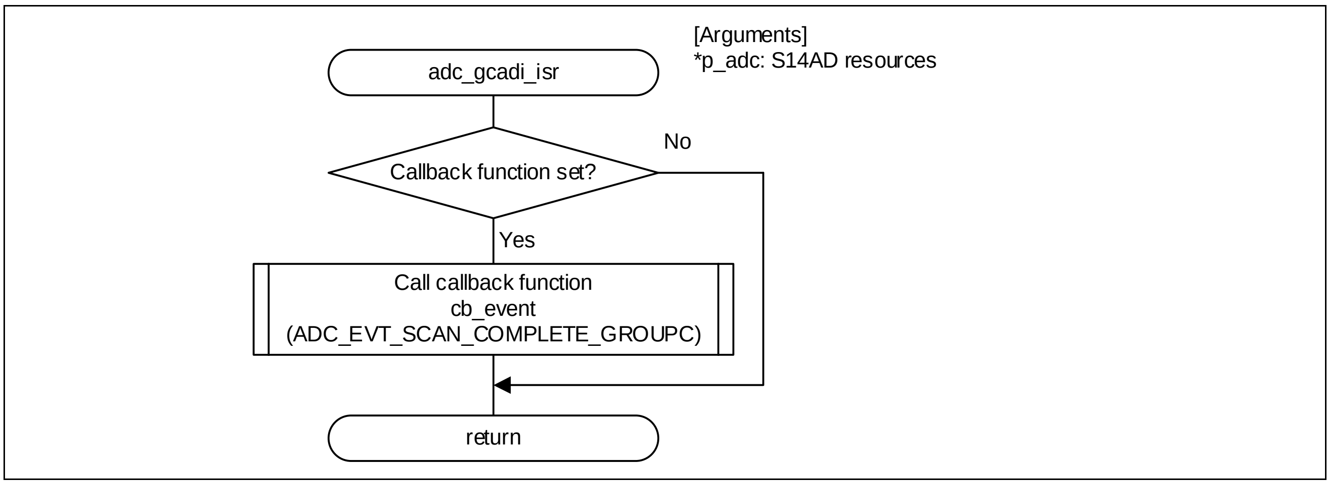


Figure 4-46 adc_gcadi_isr Function Processing Flow

4.1.41 adc_wcmpm_isr Function

Table 4-43 adc_wcmpm_isr Function Specifications

Format	static void adc_wcmpm_isr(st_adc_resources_t * const p_adc)
Description	WCMPM interrupt processing
Argument	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.
Return value	None
Remarks	-

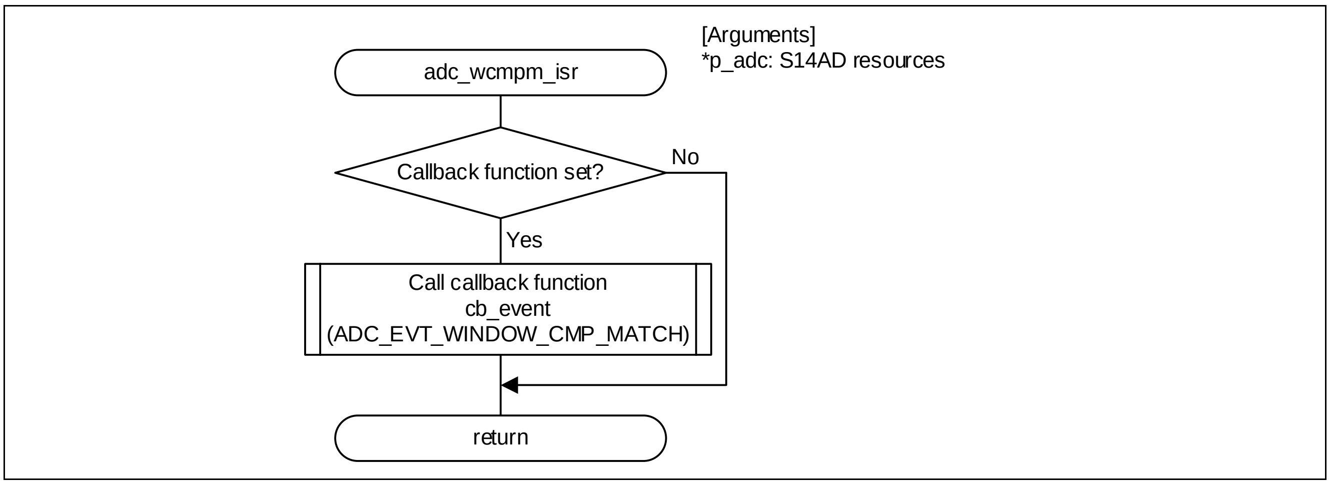


Figure 4-47 adc_wcmpm_isr Function Processing Flow

4.1.42 adc_wcmpum_isr Function

Table 4-44 adc_wcmpum_isr Function Specifications

Format	static void adc_wcmpum_isr(st_adc_resources_t * const p_adc)
Description	WCMPUM interrupt processing
Argument	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.
Return value	None
Remarks	-

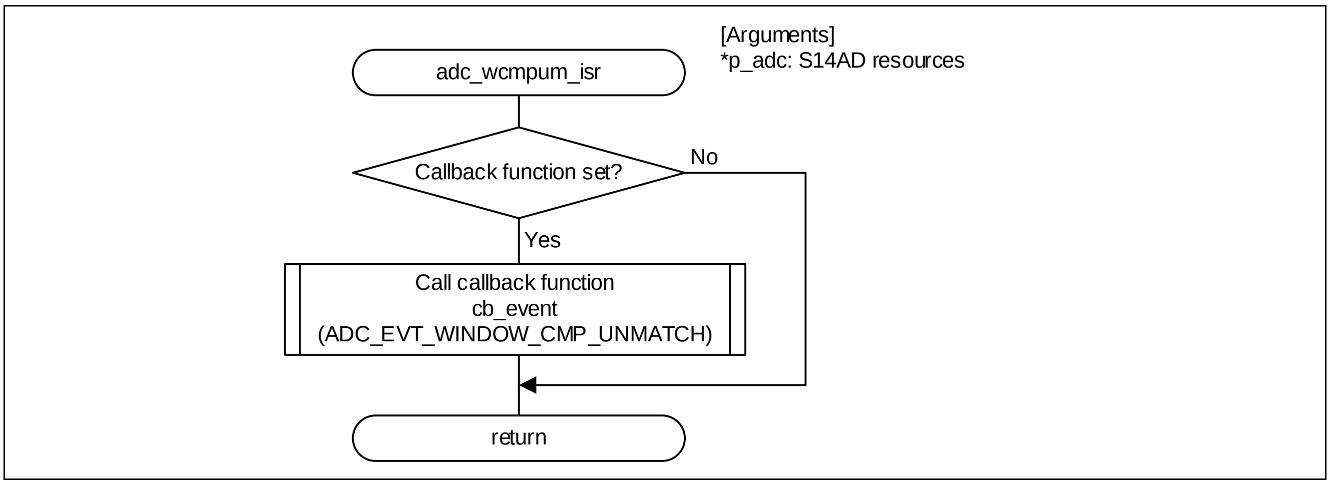


Figure 4-48 adc_wcmpum_isr Function Processing Flow

4.1.43 adc_cmpai_isr Function

Table 4-45 adc_cmpai_isr Function Specifications

Format	static void adc_cmpai_isr(st_adc_resources_t * const p_adc)
Description	CMPAI interrupt processing
Argument	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.
Return value	None
Remarks	-

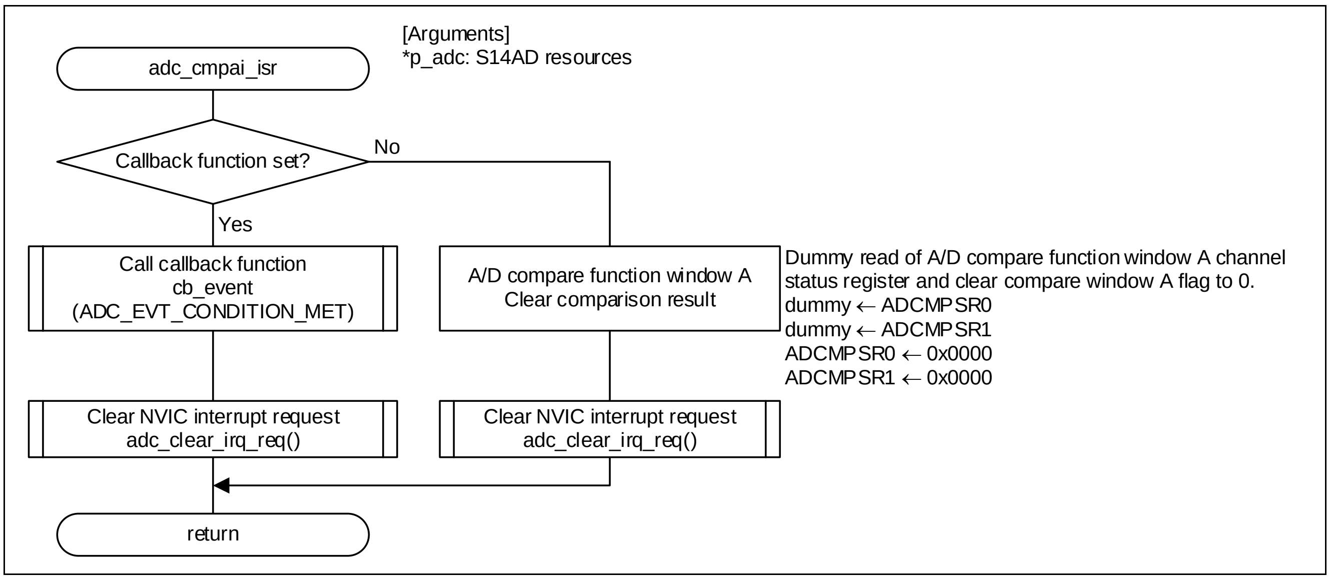


Figure 4-49 adc_cmpai_isr Function Processing Flow

4.1.44 adc_cmpbi_isr Function

Table 4-46 adc_cmpbi_isr Function Specifications

Format	static void adc_cmpbi_isr(st_adc_resources_t * const p_adc)
Description	CMPBI interrupt processing
Argument	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.
Return value	None
Remarks	-

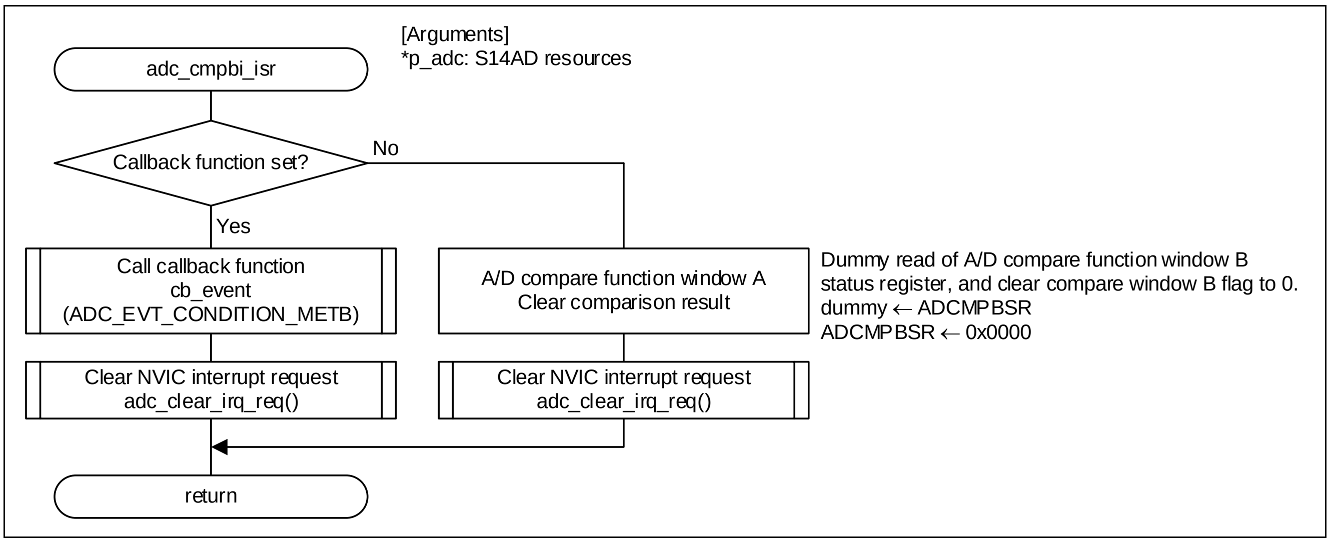


Figure 4-50 adc_cmpbi_isr Function Processing Flow

4.1.45 adc_dma_adi_isr Function

Table 4-47 adc_dma_adi_isr Function Specifications Function Specifications

Format	static void adc_dma_adi_isr(st_adc_resources_t * const p_adc)
Description	Auto-read function ADI interrupt processing
Argument	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.
Return value	None
Remarks	Valid only when, in r_adc_cfg.h, DMAC/DTC is specified for interrupt request control

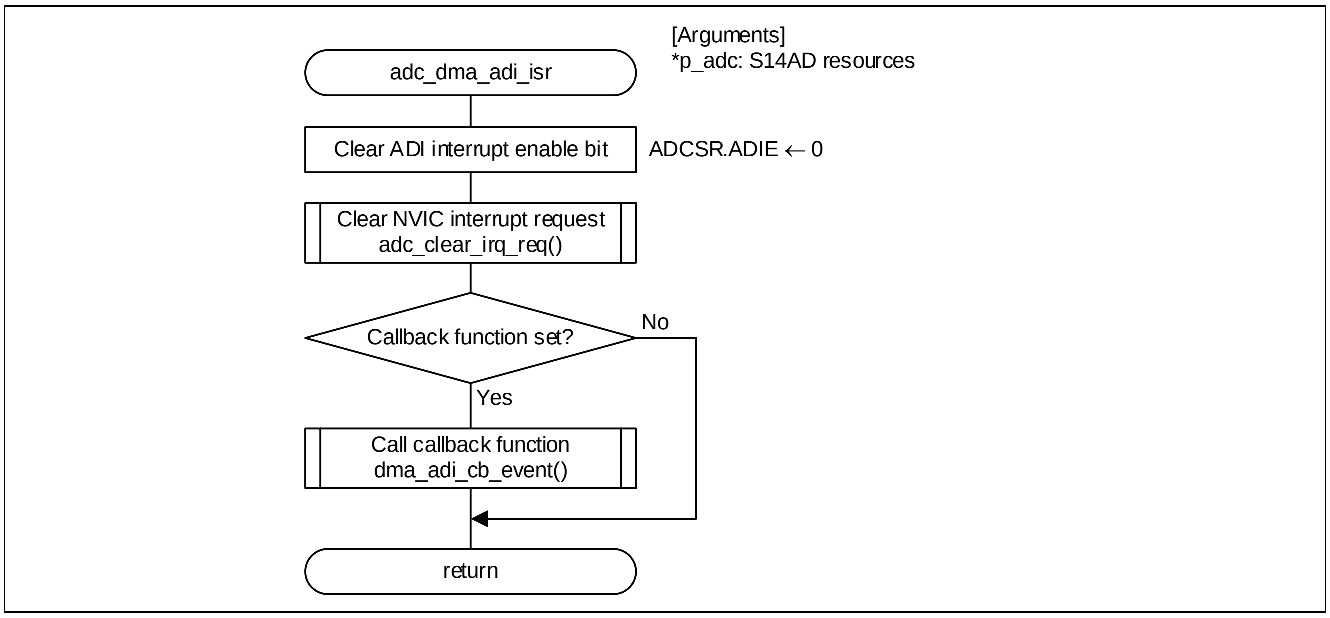


Figure 4-51 adc_dma_adi_isr Function Processing Flow

4.1.46 adc_dma_gbadi_isr Function

Table 4-48 adc_dma_gbadi_isr Function Specifications

Format	static void adc_dma_gbadi_isr(st_adc_resources_t * const p_adc)
Description	Auto-read function GBADI interrupt processing
Argument	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.
Return value	None
Remarks	Valid only when, in r_adc_cfg.h, DMAC/DTC is specified for interrupt request control

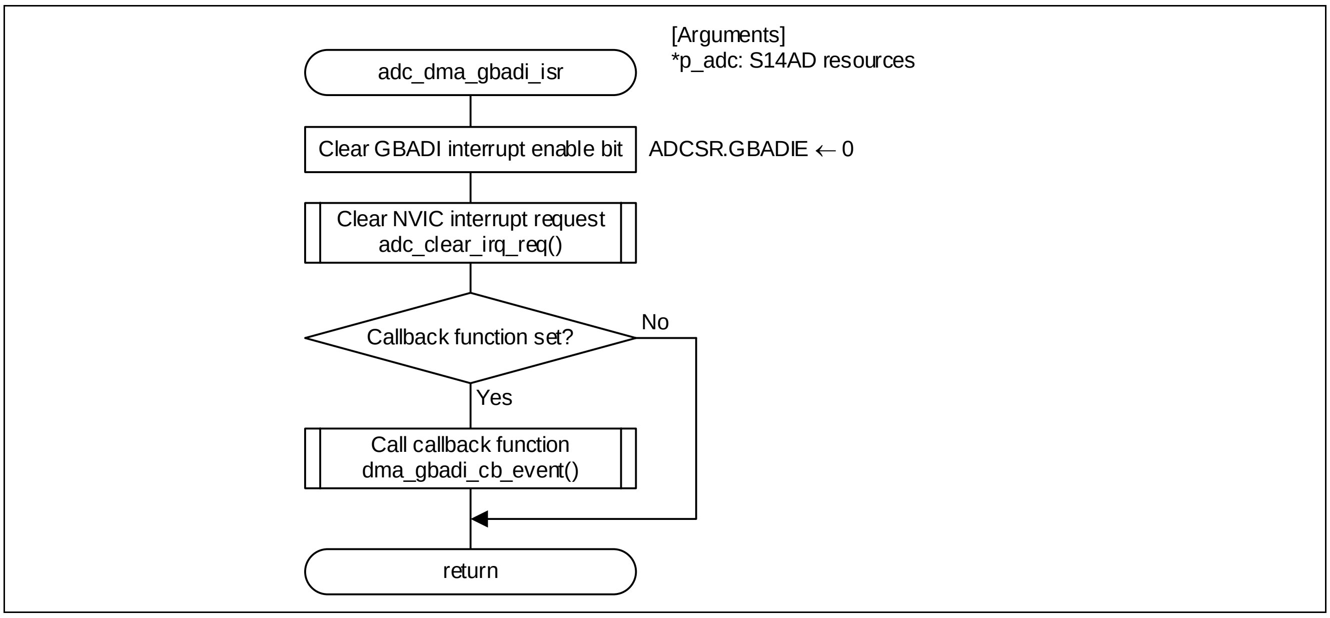


Figure 4-52 adc_dma_gbadi_isr Function Processing Flow

4.1.47 adc_dma_gcadi_isr Function

Table 4-49 adc_dma_gcadi_isr Function Specifications

Format	static void adc_dma_gcadi_isr(st_adc_resources_t * const p_adc)
Description	Auto-read function GCADI interrupt processing
Argument	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.
Return value	None
Remarks	Valid only when, in r_adc_cfg.h, DMAC/DTC is specified for interrupt request control.

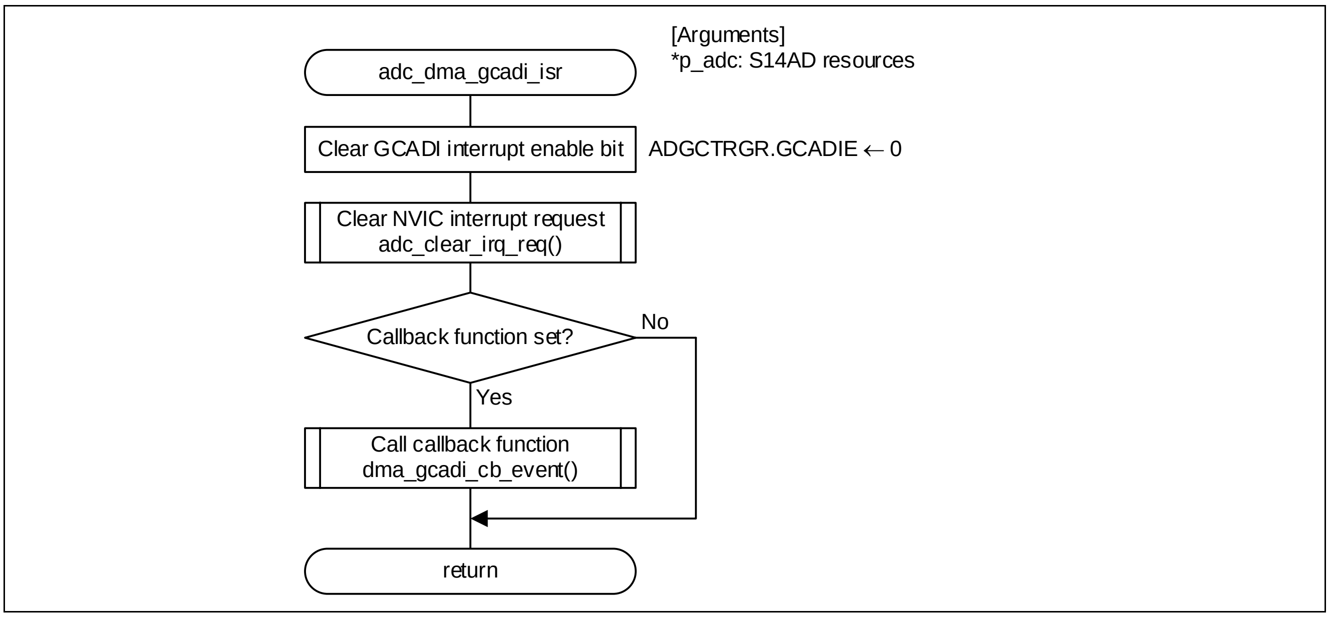


Figure 4-53 adc_dma_gcadi_isr Function Processing Flow

4.1.48 adc_dma_wcmpm_isr Function

Table 4-50 adc_dma_wcmpm_isr Function Specifications

Format	static void adc_dma_wcmpm_isr(st_adc_resources_t * const p_adc)
Description	Auto-read function WCMPM interrupt processing
Argument	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.
Return value	None
Remarks	Valid only when, in r_adc_cfg.h, DMAC/DTC is specified for interrupt request control

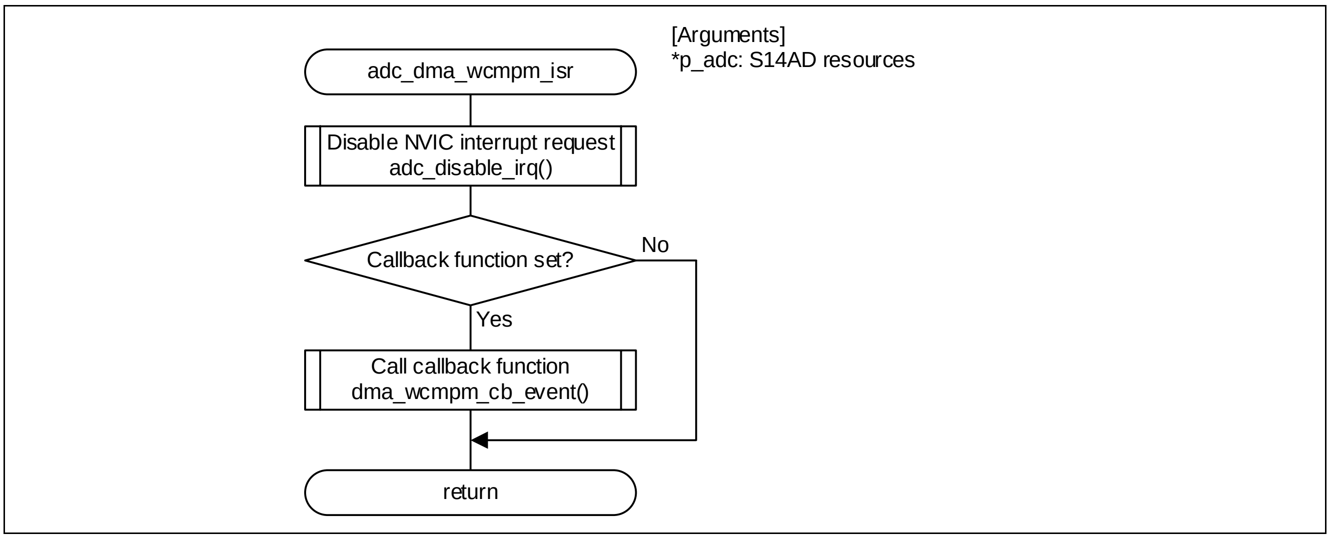


Figure 4-54 adc_dma_wcmpm_isr Function Processing Flow

4.1.49 adc_dma_wcmpum_isr Function

Table 4-51 adc_dma_wcmpum_isr Function Specifications

Format	static void adc_dma_wcmpum_isr(st_adc_resources_t * const p_adc)
Description	Auto-read function WCMPUM interrupt processing
Argument	st_adc_resources_t *p_adc: Resources of S14AD Specify the resources of the S14AD.
Return value	None
Remarks	Valid only when, in r_adc_cfg.h, DMAC/DTC is specified for interrupt request control

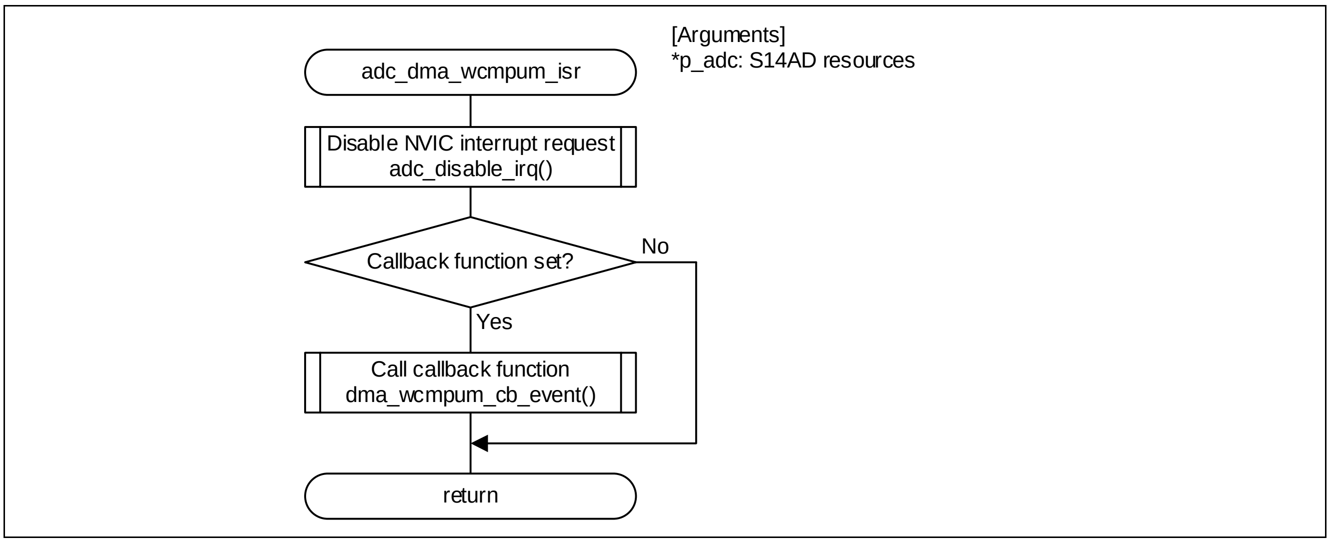


Figure 4-55 adc_dma_wcmpum_isr Function Processing Flow

4.2 Macro and Type Definitions

This section shows the definitions of the macros used in the driver and the types of them.

4.2.1 Macro Definition List

Table 4-52 List of Macro Definitions (1/2)

Definition	Value	Description
ADC_FLAG_OPENED	(1U << 0)	Opened flag
ADC_FLAG_INITIALIZED	(1U << 1)	S14AD initialized flag
ADC_FLAG_CONFIGURATION	(1U << 2)	S14AD configuration flag
ADC_FLAG_DMA_ADI_OPEN	(1U << 3)	ADI interrupt request DMA driver opened flag
ADC_FLAG_DMA_GBADI_OPEN	(1U << 4)	GBADI interrupt request DMA driver opened flag
ADC_FLAG_DMA_GCADI_OPEN	(1U << 5)	GCADI interrupt request DMA driver opened flag
ADC_FLAG_DMA_WCMPM_OPEN	(1U << 6)	WCMPM interrupt request DMA driver opened flag
ADC_FLAG_DMA_WCMPUM_OPEN	(1U << 7)	WCMPUM interrupt request DMA driver opened flag
ADC_FLAG_DMA_POS	(3)	DMA driver flag position for auto-read function
ADC_MODE_POS	(0)	Mode setting position
ADC_MODE_MASK	(0x3UL << ADC_MODE_POS)	Mode setting mask
ADC_RESOLUTION_POS	(2)	Resolution setting position
ADC_RESOLUTION_MASK	(0x1UL << ADC_RESOLUTION_POS)	Resolution setting mask
ADC_FORMAT_POS	(3)	A/D data register format setting position
ADC_FORMAT_MASK	(0x1UL << ADC_FORMAT_POS)	A/D data register format setting mask
ADC_DDA_MASK	(0x0F)	Self-diagnosis mode setting mask
ADC_VERSION_MAJOR	- (Note)	S14AD driver major version information
ADC_VERSION_MINOR	- (Note)	S14AD driver minor version information
ADC_PRV_CHAN_LOW_MASK	1500KB : (0x007F) 256KB : (0x00FF)	Mask for lower A/D channels (AN000 to AN006)
ADC_PRV_CHAN_HIGH_MASK	1500KB : (0x01FF3) 256KB : (0x08033)	Mask for higher A/D channels (AN016, AN017, AN020 to AN028)
ADC_PRV_CHAN_ALL_MASK	((ADC_PRV_CHAN_HIGH_MASK << 16) | ADC_PRV_CHAN_LOW_MASK)	Mask for all A/D channels (AN000 to AN006、AN016, AN017, AN020 to AN028)
ADC_PRV_SENSOR_TEMP	(0x01)	Temperature sensor input channel definition
ADC_PRV_TRIGER_TMR_VAL	(0x1D)	A/D conversion start trigger setting value for selected TMR trigger

Note. The value is set according to the version of this driver.

Example: Driver version 1.01

ADC_VERSION_MAJOR (1)

ADC_VERSION_MINOR (1)

Table 4-53 List of Macro Definitions (2/2)

Definition	Value	Description
ADC_PRV_TRIGER_ELC_VAL	(0x30)	A/D conversion start trigger setting value for selected ELC trigger
ADC_PRV_TRIGER_ADTRG_VAL	(0x00)	A/D conversion start trigger setting value for selected ADTRG trigger
ADC_PRV_TRIGER_DISABLE_VAL	(0x3F)	A/D conversion start trigger setting value for selected trigger
ADC_PRV_SENSOR_MASK	(0x01)	Mask for temperature sensor input channel
ADC_PRV_ADI_EVENT_USED	(1 << ADC_READ_ADI)	ADI interrupt request used flag for exclusive control
ADC_PRV_GBADI_EVENT_USED	(1 << ADC_READ_GBADI)	GBADI interrupt request used flag for exclusive control
ADC_PRV_GCADI_EVENT_USED	(1 << ADC_READ_GCADI)	GCADI interrupt request used flag for exclusive control
ADC_PRV_WCMPM_EVENT_USED	(1 << ADC_READ_WCMPM)	WCMPM interrupt request used flag for exclusive control
ADC_PRV_WCMPUM_EVENT_USED	(1 << ADC_READ_WCMPUM)	WCMPUM interrupt request used flag for exclusive control
ADC_PRV_ADI_DMAC_SOURCE_ID	(0x23)	DELS bit setting for ADI
ADC_PRV_GBADI_DMAC_SOURCE_ID	(0x24)	DELS bit setting for GBADI
ADC_PRV_GCADI_DMAC_SOURCE_ID	(0x29)	DELS bit setting for GCADI
ADC_PRV_WCMPM_DMAC_SOURCE_ID	(0x27)	DELS bit setting for WCMPM
ADC_PRV_WCMPUM_DMAC_SOURCE_ID	(0x28)	DELS bit setting for WCMPUM
ADC_PRV_USED_DMAC_DRV	(0x00FF & (S14AD_ADI_CONTROL | S14AD_GBADI_CONTROL | S14AD_GCADI_CONTROL | S14AD_WCMPM_CONTROL | S14AD_WCMPUM_CONTROL))	Definition for DMAC driver availability judgment
ADC_PRV_USED_DTC_DRV	(S14AD_USED_DTC & (S14AD_ADI_CONTROL | S14AD_GBADI_CONTROL | S14AD_GCADI_CONTROL | S14AD_WCMPM_CONTROL | S14AD_WCMPUM_CONTROL))	Definition for DTC driver availability judgment
ADC_PRV_USED_DTC_DMAC_DRV	(ADC_PRV_USED_DMAC_DRV | ADC_PRV_USED_DTC_DRV)	Definition for DMAC/DTC driver availability judgment
ADC_PRV_WCMP_USED_DMAC_DRV	(0x00FF & (S14AD_WCMPM_CONTROL | S14AD_WCMPUM_CONTROL))	Definition for window compare DMAC driver usage
ADC_PRV_WCMP_USED_DTC_DRV	(S14AD_USED_DTC & (S14AD_WCMPM_CONTROL | S14AD_WCMPUM_CONTROL))	Definition for window compare DTC driver usage
ADC_PRV_WCMP_USED_DTC_DMAC_DRV	(ADC_PRV_WCMP_USED_DMAC_DRV | ADC_PRV_WCMP_USED_DTC_DRV)	Definition for window compare DMAC/DTC driver usage

4.2.2 e_adc_sst Definition

This definition is used to specify a register for which sampling time is changed using the AD_CMD_SET_SAMPLING_ANn (Note), AD_CMD_SET_SAMPLING_AN016_AN017_AN020_TO_AN028, and AD_CMD_SET_SAMPLING_TEMP commands.

Note. 256KB Group : n = 0 to 7,L,T 1500KB Group : n = 0 to 6,L,T

Table 4-54 List of e_adc_sst Definitions (1500KB Group)

Definition	Value	Description
AD_SST_AN000	0	Sampling state register (ADSSTR0) changed
AD_SST_AN001	1	Sampling state register (ADSSTR1) changed
AD_SST_AN002	2	Sampling state register (ADSSTR2) changed
AD_SST_AN003	3	Sampling state register (ADSSTR3) changed
AD_SST_AN004	4	Sampling state register (ADSSTR4) changed
AD_SST_AN005	5	Sampling state register (ADSSTR5) changed
AD_SST_AN006	6	Sampling state register (ADSSTR6) changed
AD_SST_AN016_AN017_AN020_TO_AN028	7	Sampling state register (ADSSTRL) changed
AD_SST_TEMP	8	Sampling state register (ADSSTRT) changed

Table 4-55 List of e_adc_sst Definitions (256KB Group)

Definition	Value	Description
AD_SST_AN000	0	Sampling state register (ADSSTR0) changed
AD_SST_AN001	1	Sampling state register (ADSSTR1) changed
AD_SST_AN002	2	Sampling state register (ADSSTR2) changed
AD_SST_AN003	3	Sampling state register (ADSSTR3) changed
AD_SST_AN004	4	Sampling state register (ADSSTR4) changed
AD_SST_AN005	5	Sampling state register (ADSSTR5) changed
AD_SST_AN006	6	Sampling state register (ADSSTR6) changed
AD_SST_AN007	7	Sampling state register (ADSSTR7) changed
AD_SST_AN016_AN017_AN020_AN021_VSC_VCC	8	Sampling state register (ADSSTRL) changed
AD_SST_TEMP	9	Sampling state register (ADSSTRT) changed

4.2.3 e_adc_dma_cmd_t Definition

This definition specifies the modes for auto-read function.

Table 4-56 List of e_adc_sst Definitions

Definition	Value	Description
ADC_READ_NORMAL	0	Normal transfer mode for auto-read function
ADC_READ_BLOCK	1	Block transfer mode for auto-read function
ADC_READ_COMPARE	2	Compare mode for auto-read function

4.3 Structure Definitions

4.3.1 st_adc_resources_t Structure

This structure configures the resources of the S14AD.

Table 4-57 st_adc_resources_t Structure (1/3)

Element Name	Type	Description
*reg	volatile S14AD_Type	Indicates the S14AD register
pin_set	r_pinset_t	Function pointer for setting pins
pin_clr	r_pinclr_t	Function pointer for releasing pins
*adc_info	st_adc_mode_info_t	S14AD state information
*dma_info	st_adc_dma_isr_info_t	DMA state information for auto-read function
lock_id	e_system_mcu_lock_t	S14AD lock ID
mstp_id	e_lpm_mstp_t	S14AD module stop ID
adi_irq	IRQn_Type	ADI interrupt number assigned in NVIC
gbadi_irq	IRQn_Type	GBADI interrupt number assigned in NVIC
gcadi_irq	IRQn_Type	GCADI interrupt number assigned in NVIC
cmpai_irq	IRQn_Type	CMPAI interrupt number assigned in NVIC
cmpbi_irq	IRQn_Type	CMPBI interrupt number assigned in NVIC
wcmpm_irq	IRQn_Type	WCMPM interrupt number assigned in NVIC
wcmpum_irq	IRQn_Type	WCMPUM interrupt number assigned in NVIC
adi_iesr_val	uint32_t	IESR register setting value for ADI interrupt
gbadi_iesr_val	uint32_t	IESR register setting value for GBADI interrupt
gcadi_iesr_val	uint32_t	IESR register setting value for GCADI interrupt
cmpai_iesr_val	uint32_t	IESR register setting value for CMPAI interrupt
cmpbi_iesr_val	uint32_t	IESR register setting value for CMPBI interrupt
wcmpm_iesr_val	uint32_t	IESR register setting value for WCMPM interrupt
wcmpum_iesr_val	uint32_t	IESR register setting value for WCMPUM interrupt
adi_priority	uint32_t	ADI interrupt priority level
gbadi_priority	uint32_t	GBADI interrupt priority level
gcadi_priority	uint32_t	GCADI interrupt priority level
cmpai_priority	uint32_t	CMPAI interrupt priority level
cmpbi_priority	uint32_t	CMPBI interrupt priority level
wcmpm_priority	uint32_t	WCMPM interrupt priority level
wcmpum_priority	uint32_t	WCMPUM interrupt priority level

Table 4-58 st_adc_resources_t Structure (2/3)

Element Name	Type	Description
*adi_dma_drv	DRIVER_DMA	DMA driver for ADI interrupt request If ADI interrupt request is used for interrupt, NULL is set.
adi_dma_source	uint16_t	DELS bit setting for ADI
*adi_dtc_info	DELS bit setting for	Address where DTC transfer information for ADI is stored
*gbadi_dma_drv	DRIVER_DMA	DMA driver for GBADI interrupt request If GBADI interrupt request is used for interrupt, NULL is set.
gbadi_dma_source	uint16_t	DELS bit setting for GBADI
*gbadi_dtc_info	st_dma_transfer_data_t	Address where DTC transfer information for GBADI is stored
*gcadi_dma_drv	DRIVER_DMA	DMA driver for GCADI interrupt request If GCADI interrupt request is used for interrupt, NULL is set.
gcadi_dma_source	uint16_t	DELS bit setting for GCADI
*gcadi_dtc_info	st_dma_transfer_data_t	Address where DTC transfer information for GCADI is stored
*wcmpm_dma_drv	DRIVER_DMA	DMA driver for WCMPM interrupt request If WCMPM interrupt request is used for interrupt, NULL is set.
wcmpm_dma_source	uint16_t	DELS bit setting for WCMPM
*wcmpm_dtc_info	st_dma_transfer_data_t	Address where DTC transfer information for WCMPM is stored
*wcmpum_dma_drv	DRIVER_DMA	DMA driver for WCMPUM interrupt request If WCMPUM interrupt request is used for interrupt, NULL is set.
wcmpum_dma_source	uint16_t	DELS bit setting for WCMPUM
*wcmpum_dtc_info	st_dma_transfer_data_t	Address where DTC transfer information for WCMPUM is stored
*dma_restart_data	st_adc_dma_restart	DMA restart information for auto-read function
adi_callback	system_int_cb_t	ADI callback function
gbadi_callback	system_int_cb_t	GBADI callback function
gcadi_callback	system_int_cb_t	GCADI callback function
cmpai_callback	system_int_cb_t	CMPAI callback function
cmpbi_callback	system_int_cb_t	CMPBI callback function
wcmpm_callback	system_int_cb_t	WCMPM callback function
wcmpum_callback	system_int_cb_t	WCMPUM callback function

Table 4-59 st_adc_resources_t Structure (3/3)

Element Name	Type	Description
dma_adi_callback	system_int_cb_t	ADI callback function for auto-read function
dma_gbadi_callback	system_int_cb_t	GBADI callback function for auto-read function
dma_gcadi_callback	system_int_cb_t	GCADI callback function for auto-read function
dma_wcmpm_callback	system_int_cb_t	WCMPM callback function for auto-read function
dma_wcmpum_callback	system_int_cb_t	WCMPUM callback function for auto-read function

4.3.2 st_adc_dma_set Structure

This defines DMA setting to be set for auto-read function.

Table 4-60 st_adc_dma_set Structure

Element Name	Type	Description
flg	uint8_t	Flag setting for exclusive control of DMA transfer trigger
*dma_drv	DRIVER_DMA	DMA driver for DMA transfer trigger
dma_source	uint16_t	DELS bit setting for DMA transfer trigger
irq	IRQn_Type	DMA transfer trigger number assigned to NVIC
iesr	uint32_t	IESR setting for DMA transfer trigger
priority	uint32_t	DMA transfer trigger interrupt priority level
*transfer_info	st_dma_transfer_data_t	DTC transfer information storage address for DMA transfer trigger

4.4 Calling External Functions

This section shows the external functions to be called from the S14AD driver API.

Table 4-61 External Functions Called from S14AD Driver APIs and Calling Conditions (1/2)

API	Functions Called	Conditions (Note)
Open	R_SYS_ResourceLock	None
	R_LPM_ModuleStart	None
	R_SYS_ResourceUnlock	When open function is executed with module stop state released.
	R_S14AD_Pinset	None
Close	R_S14AD_Pinclr	None
	R_SYS_ResourceUnlock	None
	R_NVIC_DisableIRQ	None
	R_LPM_ModuleStop	None
	R_DMAC_Close	DMAC has been specified for interrupt request control.
	R_DTC_Close	DTC has been specified for interrupt request control.
	R_DTC_Release	
	R_DTC_GetAvailabilitySrc	
ScanSet	-	-
Start	-	-
Stop	-	-
Control	R_NVIC_DisableIRQ R_SYS_IrqStatusClear R_NVIC_ClearPendingIRQ R_NVIC_EnableIRQ	One of the following commands was executed: 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_IrqStatusGet	AD_CMD_GET_AD_STATE was executed.
	R_SYS_PeripheralClockFreqGet	AD_CMD_SCLK_ENABLE was executed.
	R_DMAC_Open	DMAC was specified for interrupt request control, and one of the following commands was executed: AD_CMD_AUTO_READ_NORMAL AD_CMD_AUTO_READ_BLOCK AD_CMD_AUTO_READ_COMPARE
	R_DTC_Open	DTC was specified for interrupt request control, and one of the following commands was executed: AD_CMD_AUTO_READ_NORMAL AD_CMD_AUTO_READ_BLOCK AD_CMD_AUTO_READ_COMPARE
	R_DMAC_Create	DMAC was specified for interrupt request control, and one of the following commands was executed: AD_CMD_AUTO_READ_NORMAL AD_CMD_AUTO_READ_BLOCK AD_CMD_AUTO_READ_COMPARE
	R_DTC_Create	DTC was specified for interrupt request control, and one of the following commands was executed: AD_CMD_AUTO_READ_NORMAL AD_CMD_AUTO_READ_BLOCK AD_CMD_AUTO_READ_COMPARE

Note If operation terminates due to a parameter check error, the functions may not be called even when no condition is specified.

Table 4-62 External Functions Called from S14AD Driver APIs and Calling Conditions (2/2)

API	Function Called	Conditions (Note)
	R_DMAC_Control	DMAC was specified for interrupt request control, and one of the following commands was executed: 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	DTC was specified for interrupt request control, and one of the following commands was executed: 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	DMAC was specified for interrupt request control, and one of the following commands was executed: AD_CMD_AUTO_READ_NORMAL AD_CMD_AUTO_READ_BLOCK AD_CMD_AUTO_READ_COMPARE
Read	-	-
GetVersion	-	-

Note If operation terminates due to a parameter check error, the functions may not be called even when no condition is specified.

5. Usage Notes

5.1 Arguments

Initialize all elements of the structures used for the arguments of each function to 0 before use.

5.2 Pin Settings

The pins to be used by this driver are set and released with the R_S14AD_Pinset and R_S14AD_Pinclr functions in pin.c. The R_S14AD_Pinset and R_S14AD_Pinclr functions are called from the Open and Close functions.

Refer to "Figure 2-11 Examples of Setting Pin (1/2)" and "Figure 2-12 Examples of Setting Pin (2/2)" for how to set the pins used..

5.3 Conditions for Starting A/D Conversion

A/D conversion is performed upon detection of the A/D conversion start trigger input. The following shows the A/D conversion start triggers for each setting. When using a synchronous trigger as the A/D conversion start trigger, set the 8-bit timer (TMR) or event link controller (ELC) depending on the trigger source used.

Software trigger: Execution of Start function

External trigger: ADTRG0 pin input

Synchronous trigger (TMR): TMR0_TCORA (TMR compare match A) event detection

Synchronous trigger (ELC): ELC_S14AD event detection

5.4 Restrictions on S14AD Function Combinations and Each Function

Some S14AD function combinations cannot be used. For the function combinations, see Table 2-12, Combination List of Functions Used. For the restrictions of function setting, see the details of the pertinent control commands described in section 2.5, Setting S14AD Features by Control Function.

5.5 Power supply open control register (VOCR) setting

Use this driver after setting the power supply open control register (VOCR).

The VOCR register prevents indefinite inputs from entering the power domain that is not supplied with power. For this reason, the VOCR register is set to shut off the input signal after reset. In this state, the input signal is not propagated inside the device. For details, refer to “Control of Undefined Value Propagation Suppression in I/O Power Supply Domains” in “RE01 1500KB, 256KB Group Getting Started Guide to Development Using CMSIS Package R01AN4660”.

5.6 Registering Interrupts to NVIC

When allowing the S14AD driver to use the A/D interrupt sources for interrupts or polling, register the interrupts to the NVIC in r_system_cfg.h and then enable the interrupts in the Control function. For details, refer to

"Interrupt Control" in the RE01 1500KB, 256KB Group Getting Started Guide to Development Using CMSIS Package.

If the S14AD interrupt is not registered in NVIC, ADC_ERROR_SYSTEM_SETTING will return when an interrupt enable command of the Control function is executed.

Table 5-1 shows the definition of NVIC registration for each intended use and Figure 5-1 shows the coding example for registering the interrupts to the NVIC.

Table 5-1 Definitions of NVIC Registration for Each Intended Use

Intended Use	NVIC Registration Definition	Remarks
When using ADI interrupt	SYSTEM_CFG_EVENT_NUMBER_ADC140_ADI	
When using GBADI interrupt	SYSTEM_CFG_EVENT_NUMBER_ADC140_GBADI	
When using GCADI interrupt	SYSTEM_CFG_EVENT_NUMBER_ADC140_GCADI	
When using CMPAI interrupt	SYSTEM_CFG_EVENT_NUMBER_ADC140_CMPAI	
When using CMPBI interrupt	SYSTEM_CFG_EVENT_NUMBER_ADC140_CMPBI	
When using WCMPM interrupt	SYSTEM_CFG_EVENT_NUMBER_ADC140_WCMPM	
When using WCMPUM interrupt	SYSTEM_CFG_EVENT_NUMBER_ADC140_WCMPUM	
When using auto-read command (Note 1) (DMAC used)	SYSTEM_CFG_EVENT_NUMBER_DMACm_INT	m = 0 to 3 (Note 2)
When using auto-read command (Note 1) (DTC used)	(Note 3)	

Note 1. The auto-read command refers to one of the following commands.

- AD_CMD_AUTO_READ_NORMAL
- AD_CMD_AUTO_READ_BLOCK
- AD_CMD_AUTO_READ_COMPARE

Note 2. When NULL is set as the argument in the callback function (callback function is not used), this setting is not needed.

Note 3. Perform NVIC registration in accordance with the source of auto read.

When ADI is used as the source: SYSTEM_CFG_EVENT_NUMBER_ADC140_ADI
When GBADI is used as the source: SYSTEM_CFG_EVENT_NUMBER_ADC140_GBADI
When GCADI is used as the source: SYSTEM_CFG_EVENT_NUMBER_ADC140_GCADI
When WCMPM is used as the source: SYSTEM_CFG_EVENT_NUMBER_ADC140_WCMPM
When WCMPUM is used as the source:
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 */
...
```

Figure 5-1 Example of Interrupt Registration to NVIC (ADI used) in r_system_cfg.h

5.7 Exclusive Control of Interrupt Function and Auto-Read Function

The interrupt function and auto-read function are exclusively controlled. The interrupt requests that are specified as DMA start triggers for the auto-read function cannot be used for the interrupt and polling functions. Such interrupt requests can be released by executing the Close function. To use the interrupt requests specified as DMA start triggers for the interrupt function, the interrupt function needs to be set after execution of the Close function and initialization of the S14AD.

5.8 Event Decision of Callback Functions

When an event occurs, the callback function specified by the Open function is called. Decide from which event is the callback function called according to the arguments.

Table 5-2 lists the callback functions executed upon each event. Figure 5-2 shows an example of event decision of the callback function specified by the Open function.

Table 5-2 List of Callback Functions Executed upon Event Occurrence

Event	Callback Functions	Event Code
ADC140_ADI	Callback function specified by Open function	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
All transfers are completed for auto-read function	[DMAC] Callback function of DMACm_INT (m=0 to 3) [DTC] Callback function of interrupt specified as DTC transfer trigger	-

```
/*****************************************************************************************************
* callback function
*****************************************************************************************************/
static void callback(uint32_t event)
{
    switch(event)
    {
        case ADC_EVT_SCAN_COMPLETE:
        {
            /* Describe the processing when A/D conversion is completed
              (or when group A scanning is completed in group-scan mode). */
        }
        break;

        case ADC_EVT_SCAN_COMPLETE_GROUPB:
        {
            /* Describe the processing when A/D conversion of group B is completed. */
        }
        break;

        case ADC_EVT_SCAN_COMPLETE_GROUPC:
        {
            /* Describe the processing when A/D conversion of group C is completed. */
        }
        break;

        case ADC_EVT_CONDITION_MET:
        {
            /* Describe the processing when compare A match occurs. */
        }
        break;

        case ADC_EVT_CONDITION_METB
        {
            /* Describe the processing when compare B match occurs. */
        }
        break;

        case ADC_EVT_WINDOW_CMP_MATCH:
        {
            /* Describe the processing when window A/D compare condition match occurs. */
        }
        break;

        case ADC_EVT_WINDOW_CMP_UNMATCH:
        {
            /* Describe the processing when window A/D compare condition mismatch occurs. */
        }
        break;
        default:
        {
        }
        break;
    }
}
```

Figure 5-2 Example of Event Decision of Callback Function Specified by Open Function

5.9 Notes on the use of DTC

If DTC is selected for transmit control or receive control in the r_adc_cfg.h file of this driver, the DTC driver is used in the ADC driver.When the DTC driver is used together in the user program, execution of the R_DTC_Close function will stop all DTC transmissions.When the R_DTC_Close function is executed, the DTC startup factor used in the ADC is released, and thus the auto-read operation stops.

6. Reference Documents

User's Manual: Hardware

RE01 1500KB Group User's Manual: Hardware R01UH0796
RE01 256KB Group User's Manual: Hardware R01UH0894
(The latest version can be downloaded from the Renesas Electronics website.)

RE01 Group CMSIS Package Getting Started Guide

RE01 1500KB, 256KB Group Getting Started Guide to Development Using CMSIS Package R01AN4660

(The latest version can be downloaded from the Renesas Electronics website.)

Technical Update/Technical News

(The latest version can be downloaded from the Renesas Electronics website.)

User's Manual: Development Tools

(The latest version can be downloaded from the Renesas Electronics website.)

Revision History

Rev.	Date	Description	
		Page	**Summary**
1.00	Oct.10.2019	—	First edition issued
1.01	Oct.31.2019	program 216 17,18,222	Modified condition judgment formula of adc_dma_config function Added note about version definition Modification to comment out default pin setting of pin.c
1.02	Dec.16.2019	— program (256KB)	Compatible with 256KB group The 256KB group specifications are shown below. - Peripheral function name (ADC14) • Register name (ADC140) • Analog input terminal up to 12 channels (AN000 to AN007 (high accuracy), (AN016 to AN017 (standard accuracy), AN020 to AN021 (standard accuracy)) • VSC_VCC pin voltage output conversion function • Offset calibration function setting - 16 additions
1.03	Apr.17.2020	program	Changed the configuration so that it can be built without DMAC and DTC drivers.
1.04	Jul.28.2020	—	Error correction
1.05	Nov.05.2020	225 —	Added "5.9 Notes on the use of DTC" Error correction
1.06	Mar.05.2021	138 program	Update the flow of the R_ADC_Close function to match the program DTC driver Close processing when DTC is used is modified as follows. - Release the startup factor used by the ADC driver. - Close DTC only when there are no other valid DTC startup factors.

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.