S14AD Scan Conversion and DTC Transfer Sample Code (Using CMSIS Driver Package) for RE01 1500KB Group

R_ADC Sample Code Using CMSIS Driver Package

Summary

This application note describes the S14AD Scan conversion and DTC Transfer sample code using the RE01 1500KB Group CMSIS driver package. The sample code can be found in the project delivered with this application note.

The overview of this sample code is shown in the table below.

Table Overview of Sample Code

Overview of Sample Code Operation	Peripheral Module Mainly Used	Driver Module Mainly Used
Performs A/D conversion using the ADC driver.	ADC	R_ADC
Performs DTC transfer of A/D conversion results to RAM using the ADC driver.	DTC	R_DTC

Target Device

RE01 1500KB Group

Note

When applying the sample code covered in this application note to another microcontroller, please modify the code according to the specifications for the target microcontroller and conduct an extensive evaluation of the modified program.

Related Document

Startup Guide to Development Using CMSIS Package for RE01 1500KB Group(R01AN4660)

Contents

1. Specifications

1.1 Description of Project

A sample code project "an4702_hal_ad_dtc_re" is provided with this application note.

The an4702_hal_ad_dtc_re project has been tested using the Evaluation Kit RE01 1500KB (RTK70E015DSxxxxxBE). This project is configured to match the settings of R7F0E015D2CFB mounted on the Evaluation Kit RE01 1500KB. When using another device, change the device settings in the project to those of the target device.

1.2 Pins Used

The pins used by the sample code are shown below.

Pin Used	Purpose of Use
P007	LED2
P008	LED1
P000	AN000
P001	AN001

1.3 Folder Structure

The folder structure of the sample code is shown below.

```
ADC+DTC sample program
 ├─Device
 │  ├─ Config
 │  │  ├─ r_adc_cfg.h: R_ADC configuration definition file
 │  │  ├─ r_dtc_cfg.h: R_DTC configuration definition file
 │  │  ├─ r_lpm_cfg.h: R_LPM configuration definition file
 │  │  └─ r_system_cfg.h: System configuration definition file
 │  │
 │  ├─ Driver
 │  │  ├─ Include
 │  │  │  ├─ r_adc_api.h: R_ADC header file
 │  │  │  ├─ r_dma_common_api.h: R_DMAC/R_DTC common header file
 │  │  │  ├─ r_dtc_api.h: R_DTC header file
 │  │  │  ├─ r_lpm_api.h: R_LPM header file
 │  │  │  └─ r_system_api.h: R_SYSTEM header file
 │  │  └─ Src
 │  │     ├─ r_adc
 │  │     │  └─ r_adc_api.C: R_ADC driver source file
 │  │     ├─ r_dtc
 │  │     │  └─ r_dtc_api.C: R_DTC driver source file
 │  │     ├─ r_lpm
 │  │     │  └─ r_lpm_api.C: R_LPM driver source file
 │  │     └─ r_system
 │  │        └─ r_system_api.c: System driver source file
 │  ├─ pin.c: Pin setting file
 │  └─ pin.h: Pin setting header file
 │
 └─ src
    └─ main.c: Sample program source file
```

Figure 1.1 Folder Structure

1.4 File Configuration

Table 1-1 shows the files that are added or modified in this sample code.

Table 1-1 Files Added or Modified for this Sample Code

File Name	Overview of Processing or Configuration	Remarks
main.c	Main processing	

1.5 Option-Setting Memory

Table 1-2 shows the option-setting memory setting for the sample code. Set suitable values for a user system if required.

Table 1-2 Option-Setting Memory Setting for Sample Code

Symbol	Address	Setting	Description
AWS	0100A164h to 0100A167h	FFFF FFFFh	No access window settings
OSIS	0100A150h to 0100A15Fh	FFFF FFFFh	No ID code protection (All FFh)
SECMPUxxx	00000408h to 0000043Bh	FFFF FFFFh	MPU is disabled.
OFS1	00000404h to 00000407h	FFFF FFFFh	After a reset, the voltage monitor 0 reset is disabled. After a reset, HOCO oscillation is disabled.
OFS0	00000400h to 00000403h	FFFF FFFFh	Automatic activation of IWDT is disabled. Automatic activation of WDT is disabled.

2. Operating Conditions

The operation of the sample code provided with this application note has been tested under the following conditions (Table 2-1).

Table 2-1 Operating Conditions

Item		Description
Microcontroller used		R7F0E015D2CFB 144pin
Operating frequency	PLL is selected as the system clock	• Main clock: 32 MHz • PLL: 64 MHz (main clock frequency is divided by 4 and then multiplied by 8) • System clock (ICLK): 64 MHz (PLL) • Peripheral module clock A (PCLKA): 64 MHz (PLL frequency is not divided) • Peripheral module clocks B(PCLKB): 32 MHz (PLL frequency is divided by 2)
Operating voltage		• 3.3V
Target board		Evaluation Kit RE01 1500KB （RTK70E015DSxxxxxBE）
Integrated Development Environment	GCC	Renesas e^2 studio Version 7
	IAR	IAR Embedded Workbench for ARM Version 8.32
C compiler	GCC	GCC ARM Embedded Version 6.3.1.20170620 GNU 6-2017-q2-update
	IAR	IAR C/C++ Compiler for ARM Version 8.32
Debugger		Segger J-Link OB
I/O header Version		Rev1.00
Sample code Version		Rev1.00

3. Description of Software

This sample code performs A/D conversion in continuous-scan mode using the R_ADC driver. The converted results are transferred to RAM using a DTC transfer.

The pins to be used in A/D conversion are AN000 and AN001.

The sample code performs the following operations.

- After release from the reset state, initializes the A/D converter so that AN000 and AN001 are converted in continuous-scan mode and the converted results are transferred to RAM by DTC transfer. The DTC is initialized so that transfer stops after the conversion results of AN000 and AN001 are acquired once.
- Starts A/D conversion.
- After completion of A/D conversion, a callback function is called. In the callback function, the A/D conversion end flag is set.
- The sample software performs the following functions every 100 ms.
    - Acquires the A/D-converted value of AN000 that has been transferred to RAM, and turns LED0 on if the value is lower than 1/2 AVCC and turns LED0 off if the value is 1/2 AVCC or higher.
    - Acquires the A/D-converted value of AN001 that has been transferred to RAM, and turns LED1 on if the value is lower than 1/2 AVCC and turns LED1 off if the value is 1/2 AVCC or higher.
    - Restarts DTC transfer.

Table 3-1 Information of Sample Program Operation (ADC)

Item	Setting
Number of A/D conversion channels	2
A/D conversion mode	Continuous-scan mode
Conditions for A/D conversion start	Software trigger
DTC transfer source	Completion of A/D conversion
DTC transfer count	1
DTC transfer block size	2
DTC transfer unit	2 bytes

3.1 System Configuration

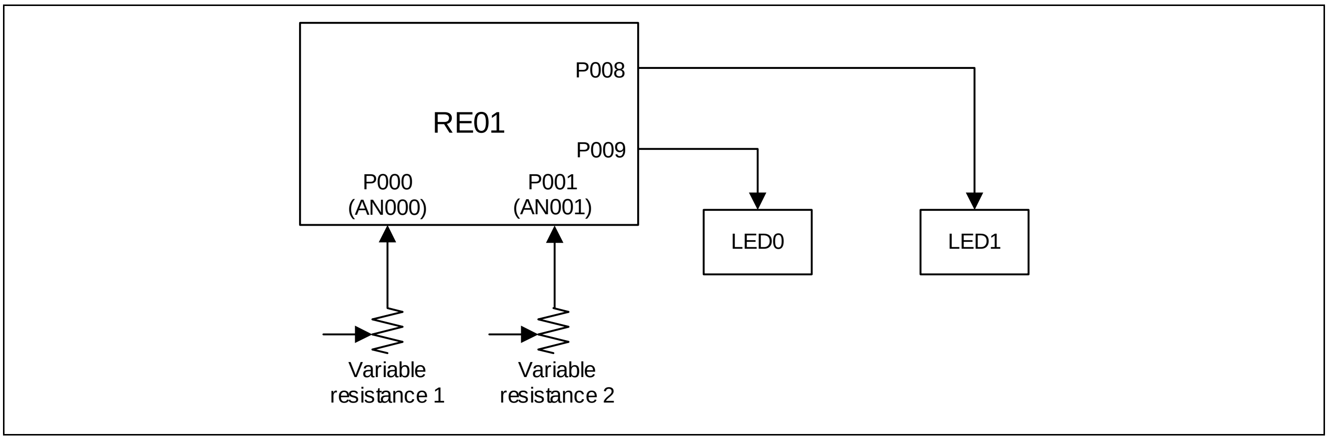


Figure 3.1 System Configuration

3.2 Driver Configuration

Table 3-2 Driver Configuration

Item	Location of Change	Details of Change
Transferring A/D conversion results using DTC	[r_adc_cfg.h] S14AD_ADI_CONTROL	● Setting change S14AD_USED_DTC
Registering ADI interrupts to NVIC	[r_system_cfg.h] SYSTEM_CFG_EVENT_NUMBER_ADC140_ADI	● Setting change (SYSTEM_IRQ_EVENT_NUMBER4)

3.3 List of Functions

The functions added to the sample code are described here.

main	
Overview	Main processing
Header	None
Declaration	void main(void)
Description	This function calls the system initialization function to make initial settings of the ADC and DTC after a reset. After that, this function performs A/D conversion every 100 ms and turns LED1 and LED2 on according to the conversion result.
Argument	None
Return Value	None

system_init	
Overview	System initialization processing
Header	None
Declaration	static void system_init(void)
Description	This function initializes sections, initializes the system, the R_LPM driver, and calls the IO power supply setting function.
Argument	None
Return Value	None

dtc_callback	
Overview	DTC transfer end callback processing
Header	None
Declaration	static void dtc_callback(void)
Description	This function sets the DTC transfer end flag.
Argument	None
Return Value	None

3.4 Flowcharts

Figure 3.2 shows a flowchart of the main processing.

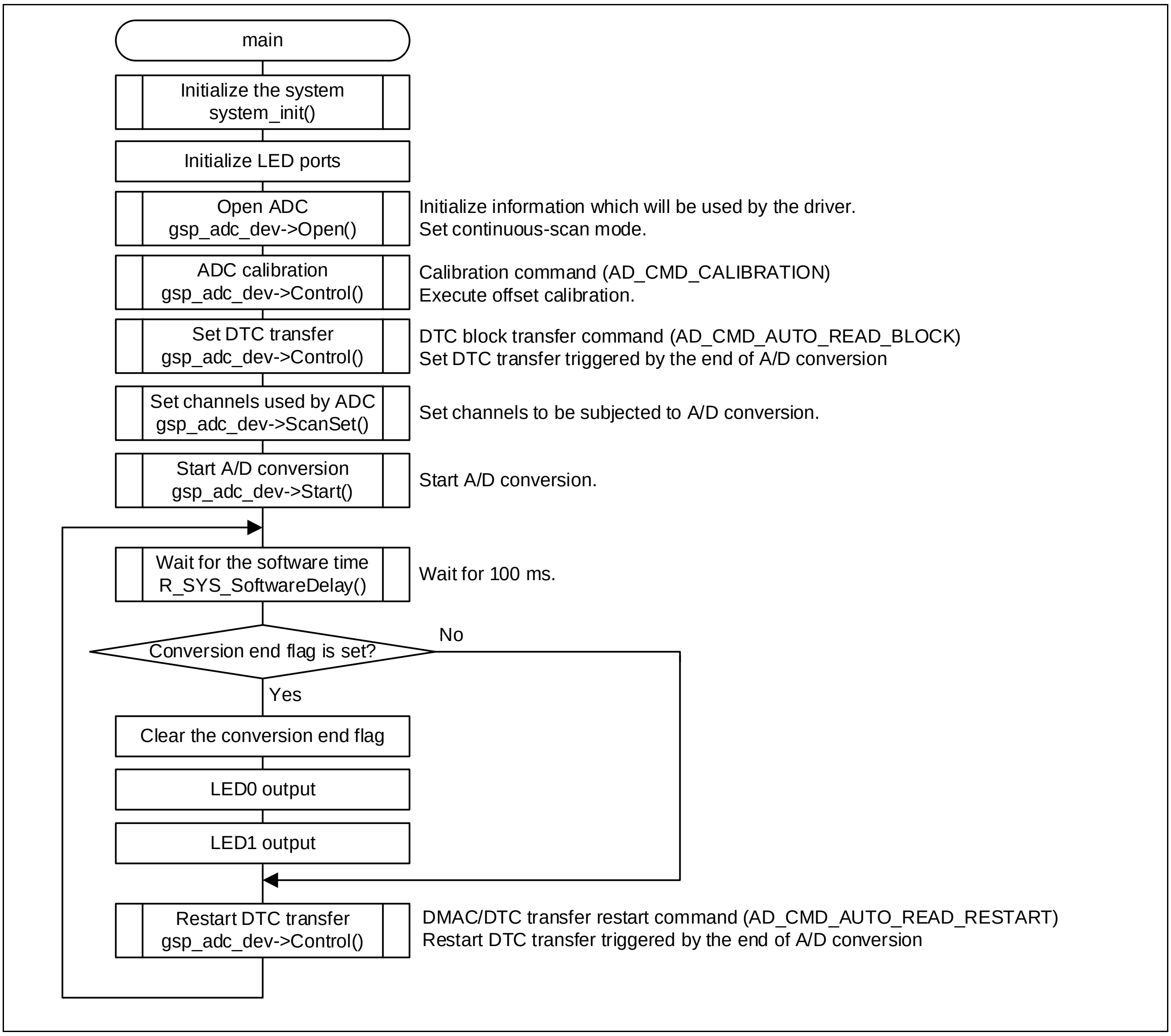


Figure 3.2 Main Processing

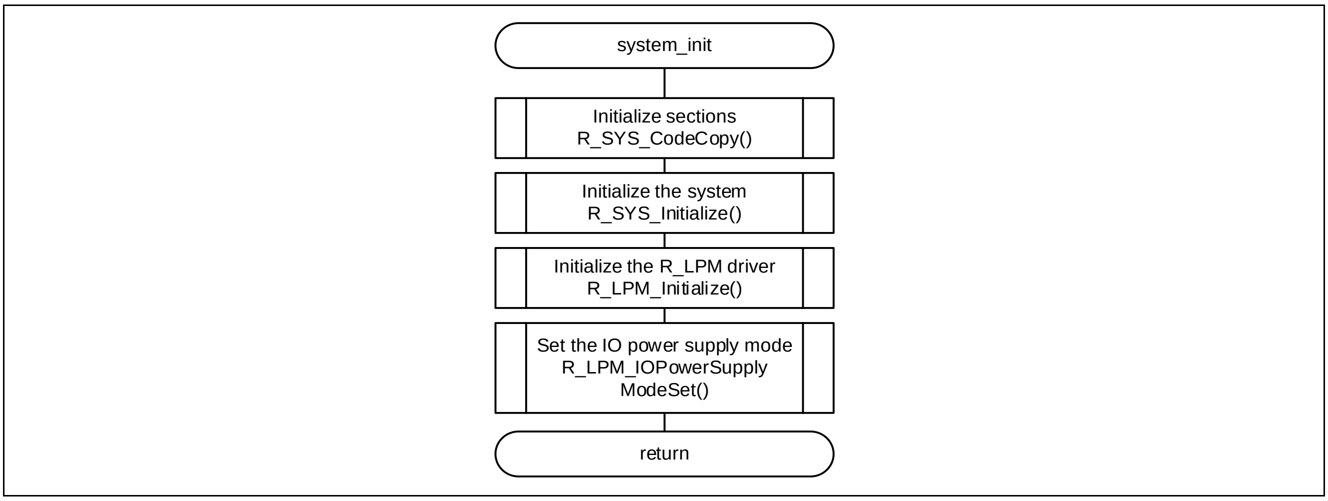


Figure 3.3 System Initialization Processing

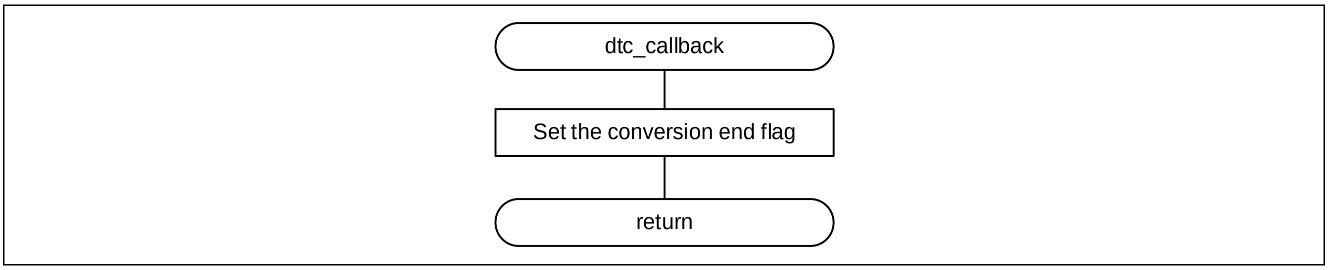


Figure 3.4 DTC Transfer End Callback Processing

4. Specifications of Driver APIs

4.1 External Specification

This driver contains documents that describes the external API specification. These files are contained in the Driver Specification folder within the Documents.

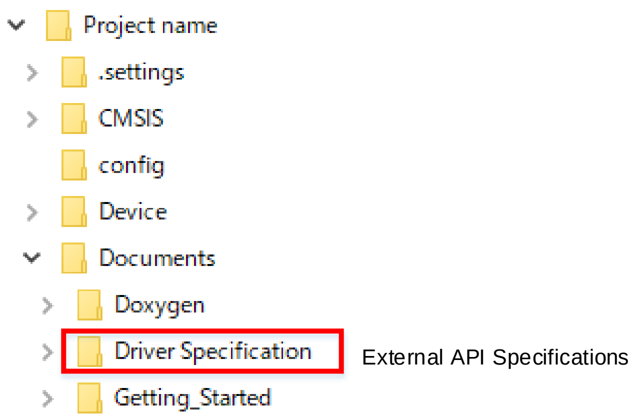


Figure 4-1 Location of External API Specifications

5. Usage Notes of R_ADC Driver

This chapter introduces the main points to concern regarding the R_ADC driver. Note that not all the usage notes are given here.

For other notes, see the external specification document described in "4 Specifications of Driver APIs".

5.1 Usage of Auto-Read Commands Using DMAC

When acquiring the A/D conversion result by the DMAC using an auto-read command (AD_CMD_AUTO_READ_NORMAL, AD_CMD_AUTO_READ_BLOCK, or AD_CMD_AUTO_READ_COMPARE), set the DMA resource to be used for the target interrupt source in r_adc_cfg.h.

Table 5-1 shows the definition of configuration parameters for A/D conversion end or compare match event notification. Table 5-2 shows the definition of values indicating methods of A/D conversion end or compare match event notification.

Table 5-1 Definitions of Configuration Parameters for A/D Conversion End or Compare Match Event Notification

Definition (*)	Initial Value	Description
S14AD_ADI_CONTROL	S14AD_USED_INTERRUPT	ADI conversion end notification (Initial value: Interrupt)
S14AD_GBADI_CONTROL	S14AD_USED_INTERRUPT	GBADI conversion end notification (Initial value: Interrupt)
S14AD_GCADI_CONTROL	S14AD_USED_INTERRUPT	GCADI conversion end notification (Initial value: Interrupt)
S14AD_WCMPM_CONTROL	S14AD_USED_INTERRUPT	Compare match notification (Initial value: Interrupt)
S14AD_WCMPUM_CONTROL	S14AD_USED_INTERRUPT	Compare mismatch notification (Initial value: Interrupt)

Note. Only an interrupt can be used for notification of CMPAI or CMPBI (notification by DMA is not possible).

Table 5-2 Definitions of Values Indicating Methods of A/D Conversion End or Compare Match Event Notification

Definition	Value	Description
SCI_USED_INTERRUPT	(0)	A/D conversion end or compare result is notified through an interrupt or polling.
SCI_USED_DMAC0	(1<<0)	A/D conversion end or compare result is notified through DMAC0.
SCI_USED_DMAC1	(1<<1)	A/D conversion end or compare result is notified through DMAC1.
SCI_USED_DMAC2	(1<<2)	A/D conversion end or compare result is notified through DMAC2.
SCI_USED_DMAC3	(1<<3)	A/D conversion end or compare result is notified through DMAC3.
SCI_USED_DTC	(1<<15)	A/D conversion end or compare result is notified through DTC.

5.2 Registering Interrupts to NVIC

When using interrupts in the ADC or polling for interrupts, register the interrupts to the NVIC in r_system_cfg.h and then enable the interrupts in the Control function.

Table 5-3 shows the definition name of NVIC registration for each intended use. **Figure 5.1** shows an example of registering interrupts to the NVIC.

Table 5-3 Definition of NVIC Registration for Intended Use

Intended Use	Definition of NVIC Registration	Remarks
When using the ADI interrupt	SYSTEM_CFG_EVENT_NUMBER_ADC140_ADI	
When using the GBADI interrupt	SYSTEM_CFG_EVENT_NUMBER_ADC140_GBADI	
When using the GCADI interrupt	SYSTEM_CFG_EVENT_NUMBER_ADC140_GCADI	
When using the CMPAI interrupt	SYSTEM_CFG_EVENT_NUMBER_ADC140_CMPAI	
When using the CMPBI interrupt	SYSTEM_CFG_EVENT_NUMBER_ADC140_CMPBI	
When using the WCMPM interrupt	SYSTEM_CFG_EVENT_NUMBER_ADC140_WCMPM	
When using the WCMPUM interrupt	SYSTEM_CFG_EVENT_NUMBER_ADC140_WCMPUM	
When using an auto-read command (*1) (DMAC is used)	SYSTEM_CFG_EVENT_NUMBER_DMACm_INT	m = 0 to 3 (*2)
When using an auto-read command (*1) (DTC is used)	(*3)	

Note 1. The auto-read commands are as follows:
- 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 Registering Interrupts to NVIC (When Using ADI)

5.3 Using A/D Converter in Snooze Mode

When performing A/D conversion in snooze mode, set ADC_CMPAI_SNOOZE_USE or ADC_CMPBI_SNOOZE_USE in r_adc_cfg.h to (1).

When (1) is set, registering CMPAI or CMPBI to the NVIC is skipped. Therefore, they cannot be used in interrupts.

5.4 Pin Settings

The pins to be used by this driver will be set when the Open function is executed. Table 5-4 shows the pins that are set by default. To change the pins to be used, modify code in the R_S14AD_Pinset and R_S14AD_Pinclr functions of pin.c.

Figure 5.2 and **Figure 5.3** show an example of pin settings in which AN000 to AN003 are the only analog input pins used and the ADTRG pin is changed to P500.

Table 5-4 Pins Used by Default

Channel	Pin Function	Assigned Port
ADC	AN000	P000
	AN001	P001
	AN002	P002
	AN003	P003
	AN004	P004
	AN005	P005
	AN006	P006
	AN016	P010
	AN017	P011
	AN020	P014
	AN021	P015
	AN022	P500
	AN023	P501
	AN024	P502
	AN025	P503
	AN026	P504
	AN027	P505
	AN028	P506
	ADTRG0	P204

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

    /* 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;

    /* 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;

    /* 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;

/* Commented out because AN003 to AN028 are not used */
#if 0
    /* 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;
    ・・・
    /* AN028 : P506 */
    PFS->P506PFS_b.ISEL = 0U;
    PFS->P506PFS_b.PSEL = 0U;
    PFS->P506PFS_b.PMR  = 0U;
    PFS->P506PFS_b.PDR  = 0U;
    PFS->P506PFS_b.ASEL = 1U;
#endif

/* Commented out because ADTRG0 is changed from P204 (default) to P500 */
//    /* 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;

/* Uncommented because ADTRG0 is changed from P204 (default) to P500 */
    /* 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 5.2 Example of Changing Pin Settings (1/2)

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

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

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

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

/* Commented out because AN003 to AN028 are not used */
#if 0
    /* AN003 : P003 */
    PFS->P003PFS &= R_PIN_PRV_CLR_MASK;
     . . .
    /* AN028 : P506 */
    PFS->P506PFS &= R_PIN_PRV_CLR_MASK;
#endif

/* Commented out because ADTRG0 is changed from P204 (default) to P500 */
//    /* ADTRG0 : P204 */
//    PFS->P204PFS &= R_PIN_PRV_CLR_MASK;

/* Uncommented because ADTRG0 is changed from P204 (default) to P500 */
    /* 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 5.3 Example of Changing Pin Settings (2/2)

5.5 Calibration

After the Open function has been executed following release from the reset state, execute calibration (execute the Control function with the AD_CMD_CALIBRATION command set as the argument) before the ScanSet function is executed.

Figure 5.4 shows an example of performing calibration.

```
extern DRIVER_S14AD Driver_S14AD;
DRIVER_S14AD *gsp_adc_dev = &Driver_S14AD;

/*******************************************************************************
* Function Name: main
* Description  : Main Function
* Arguments    : none
* Return Value : none
*******************************************************************************/
void main(void)
{
    st_adc_pins_t scanset_pin;

    gsp_adc_dev->Open(ADC_REPEAT_SCAN, 0x10, NULL); /* ADC driver is opened */
    gsp_adc_dev->Control(AD_CMD_CALIBRATION, NULL); /* Calibration is executed */

    /** Channel Select */
    scanset_pin.an_chans = ADC_MSEL_AN00 | ADC_MSEL_AN01;
    scanset_pin.sensor   = ADC_SENSOR_NOTUSE;
    gsp_adc_dev->ScanSet(ADC_GROUP_A, scanset_pin, ADC_TRIGER_SOFT);
    …

}
```

Figure 5.4 Example of Executing Calibration

5.6 Combinations of Functions to Be Used

Some functions cannot be used at the same time. Confirm the possible combinations in the table below.

	Double-trigger	Group-scan	Self-diagnosis	Addition/ Average	Group A priority control	Sensor	Compare match	Disconnec-tion detection assist	Automatic clear
Double-trigger			X			X	X		
Group-scan									
Self-diagnosis	X			X			X	X	
Addition/Average			X						
Group A priority control									
Sensor	X							X	
Compare match	X		X						
Disconnection detection assist			X			X			
Automatic clear									

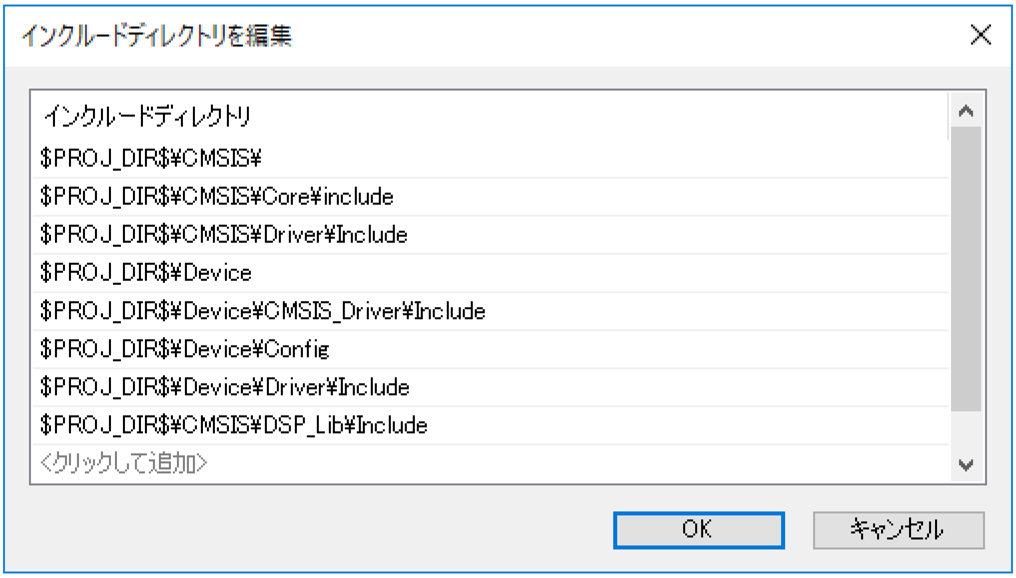
6. Troubleshooting

6.1 Occurrence of Build Error with IAR Compiler

A-1) Have the include directories been specified correctly?

When using EWARM, we recommend that the include directories are specified as shown in the example below.

The include directories can be specified from IDE Options [C/C++ Compiler] → [Preprocessor].



6.2 Occurrence of HardFault Error when API of CMSIS Driver Is Called

A) The API has possibly not been copied to RAM.

Before calling an API function that is mapped to RAM, make sure that it has been copied to RAM by the R_SYS_CodeCopy function. For details, refer to the related document No. R01AN4660.

6.3 Peripheral Function Fails to Operate when API Is Called

A) Has the API been set up correctly?

Check the API's return value to see if an error has occurred.

In particular, errors are often caused by problems related to interrupts not being set in r_system_cfg.h. For details, refer to the related document No. R01AN4660.

6.4 Normal API Return Value But No Pin Output from Peripheral Function

A) Are the pin settings correct?

Check to make sure the pins have been set up correctly by the functions in pin.c.

For details, refer to the related document No. R01AN4660.

6.5 Peripheral Function's Input or Output Does Not Operate as Expected

A) Check to make sure the VOCR register has been set up correctly before making the initial settings for peripheral functions.

For details, refer to the related document No. R01AN4660.

7. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

8. Reference Documents

User's Manual: Hardware

RE01 1500KB Group User's Manual: Hardware R01UH0796

(The latest version can be downloaded from the Renesas Electronics website.)

RE01 1500KB CMSIS Package Startup Guide

RE01 1500KB Group Startup 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	Sep. 19, 2019	—	First edition issued

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

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/.