DMAC Repeat Transfer Sample Code (Using CMSIS Driver Package) for RE01 1500KB Group

DMAC Sample Code Using CMSIS Driver Package

Summary

This application note describes the DMAC repeat transfer sample code conforming to 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
Transfers ROM data to RAM using the DMAC driver.	DMAC	R_DMAC

Target Device

RE01 1500KB Group

Note

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

Related Document

Startup Guide to Development Using CMSIS Package for R7F0E01 Group (1.5 MB) (R01AN4660)

Contents

1. Specifications

1.1 Description of Project

A sample code project "an4703_hal_dmac_repeat_re" is provided with this application note.

The an4703_hal_dmac_repeat_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
P008	LED1
P009	LED0
P508	SW2

1.3 Folder Structure

The folder structure of the sample code is shown below.

```
DMAC sample program
├─Device
│  ├─ Config
│  │  ├─ r_dmac_cfg.h: R_DMAC configuration definition file
│  │  ├─ r_lpm_cfg.h: R_LPM configuration definition file
│  │  └─ r_system_cfg.h: System configuration definition file
│  │
│  ├─ Driver
│  │  ├─ Include
│  │  │  ├─ r_dma_common_api.h: R_DMAC/R_DTC common header file
│  │  │  ├─ r_dmac_api.h: R_DMAC header file
│  │  │  ├─ r_lpm_api.h: R_LPM header file
│  │  │  └─ r_system_api.h: R_SYSTEM header file
│  │  └─ Src
│  │     ├─ r_dmac
│  │     │  └─ r_dmac_api.C: R_DMAC 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 were added or modified for 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	
pin.c	I/O port setting	Changing pins assigned to IRQ4
r_system_cfg.h	System configuration	Registering IRQ interrupts and DMAC transfer end interrupts to NVIC

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 data transfer in repeat transfer mode using the R_DMAC driver.

It transfers 16 bytes of data from Flash to two RAM areas (destinations A and B) in block transfer mode.

The sample code performs the following operations.

- After release from the reset state, the DMAC transfer source is set to IRQ4, the transfer source address is set to the start of the data in the Flash memory, and the transfer destination address is set to destination A in the RAM.
- Every time SW2 (IRQ4) is pressed, one-byte data is transferred from the start address of the Flash to destination A in the RAM.
- Every time SW2 is pressed four times, data is transferred again from the start address of the Flash.
- On completion of DMAC transfer of 16 bytes (four repeated transfers), a callback function is called. In the callback function, the transfer destination of the DMAC is changed to destination B of RAM to reactivate the DMAC. At the same time, LED0 is turned on and LED1 is turned off.
- Every time SW2 is pressed again, one-byte data is transferred from Flash to destination B of RAM.
- After completion of 16-byte data transfer, the callback function is called again. In the callback function, the transfer destination of the DMAC is changed to destination A of RAM to reactivate the DMAC. At the same time, LED0 is turned on and LED1 is turned off.

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

Item	Setting
Transfer area	Data in the Flash is transferred to the RAM area
Transfer mode	Repeat transfer mode
Data transfer unit	Single data unit: 1 byte (8 bits) Single repeat size: 4 data units

Table 3-2 Information of Sample Program Operation (IRQ4)

Item	Setting
IRQ4 detection sense select	Falling edge
ICU event link select	PORT_IRQ4
IRQ4 interrupt priority level	3

3.1 System Configuration

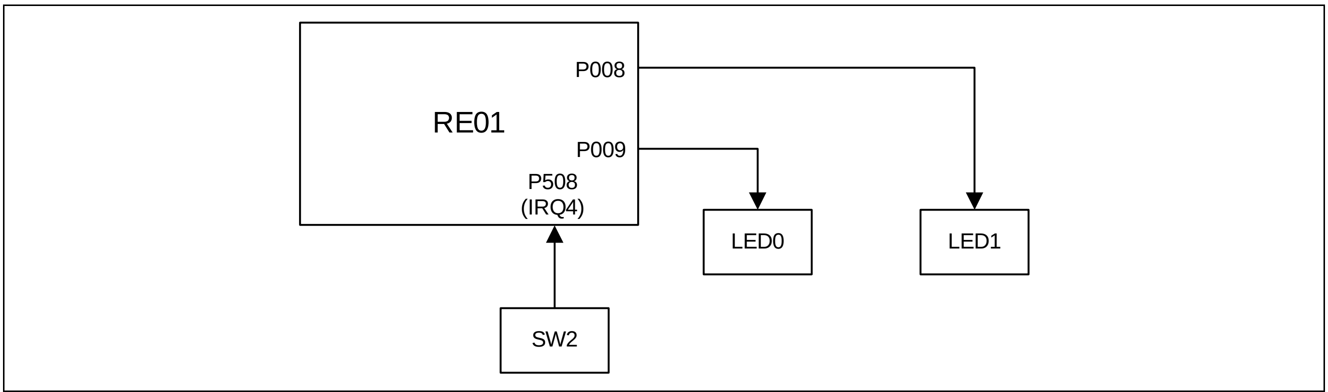


Figure 3.1 System Configuration

3.2 Driver Configurations

Table 3-3 Driver Configurations

Item	Location of Change	Details of Change

Changing the IRQ4 setting from default (P202) to P508	[pin.c]
	R_ICU_Pinset_CH4() function
		● Comment out followings
		// PFS->P202PFS_b.ASEL = 0U;
		// PFS->P202PFS_b.PSEL = 0U;
		// PFS->P202PFS_b.PDR = 0U;
		// PFS->P202PFS_b.PMR = 0U;

		● Comment out followings
		// PFS->P202PFS_b.EOFR = R_PIN_FALLING;
		// PFS->P202PFS_b.ISEL = 1U;

		● Validate followings PFS->P508PFS_b.ASEL = 0U; PFS->P508PFS_b.PSEL = 0U; PFS->P508PFS_b.PDR = 0U; PFS->P508PFS_b.PMR = 0U; PFS->P508PFS_b.EOFR = R_PIN_FALLING; PFS->P508PFS_b.ISEL = 1U;
Registering DMAC transfer end interrupts to NVIC	[r_system_cfg.h] SYSTEM_CFG_EVENT_NUMBER_DMAC0_INT	● Setting change (SYSTEM_IRQ_EVENT_NUMBER8)

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. It then sets up the DMAC transfer and starts DMAC transfer.
Argument	None
Return Value	None

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

dma_callback	
Overview	DMAC transfer end interrupt processing
Header	None
Declaration	static void dma_callback (void)
Description	After the end of DMAC repeat transfer, this function turns LED0 on and turns LED1 off if transfer to destination A has finished. Then, it changes the transfer destination address to the start address of destination B. If transfer to destination B has finished, this function turns LED0 off and turns LED1 on. Then, it changes the transfer destination address to the start address of destination A.
Argument	None
Return Value	None

.

3.4 List of Constants

Table 3-4 shows a list of constants.

Table 3-4 Constants (User Changeable) Used in Sample Code

Constant Name	Setting	Description
DMAC_LINK_TO_IRQ4	5	Argument to set IRQ4 to DMAC event link setting register
DMA_REPEAT_TIMES	4	Number of repeat transfers

3.5 Flowcharts

Figure 3.2 shows a flowchart of the main processing.

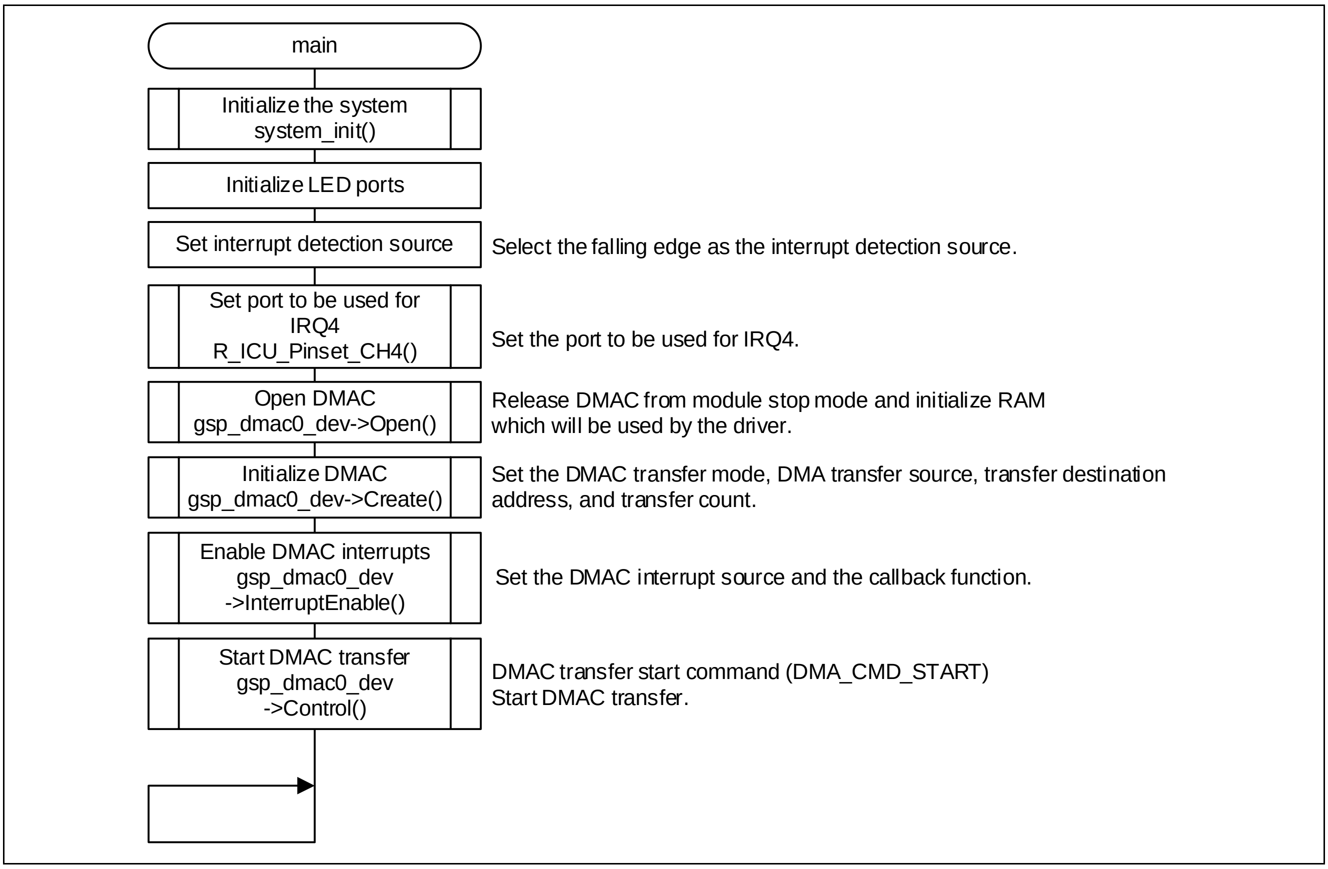


Figure 3.2 Main Processing

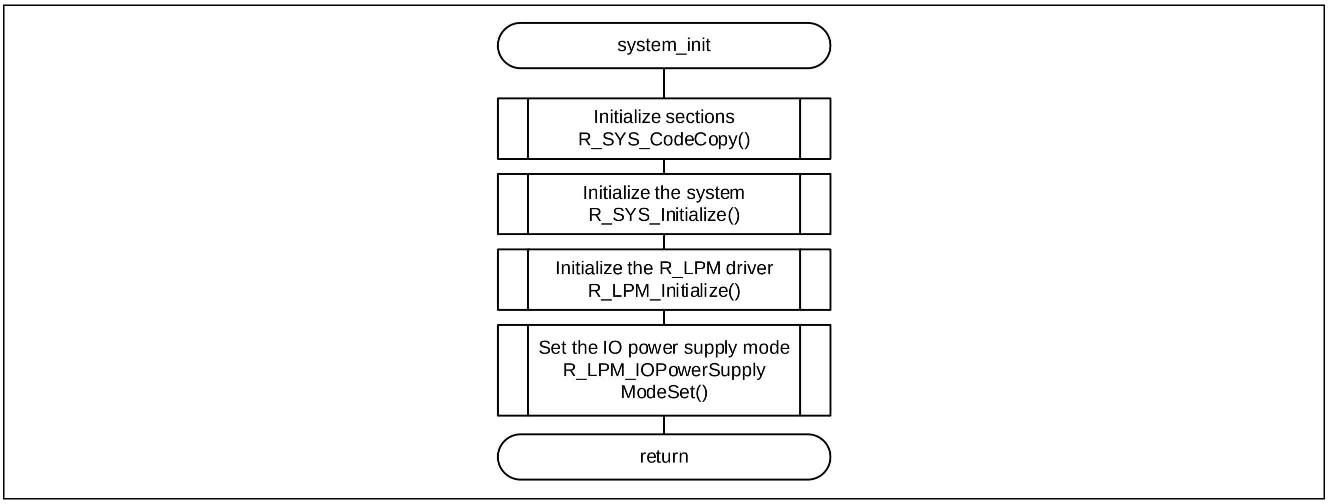


Figure 3.3 System Initialization Processing

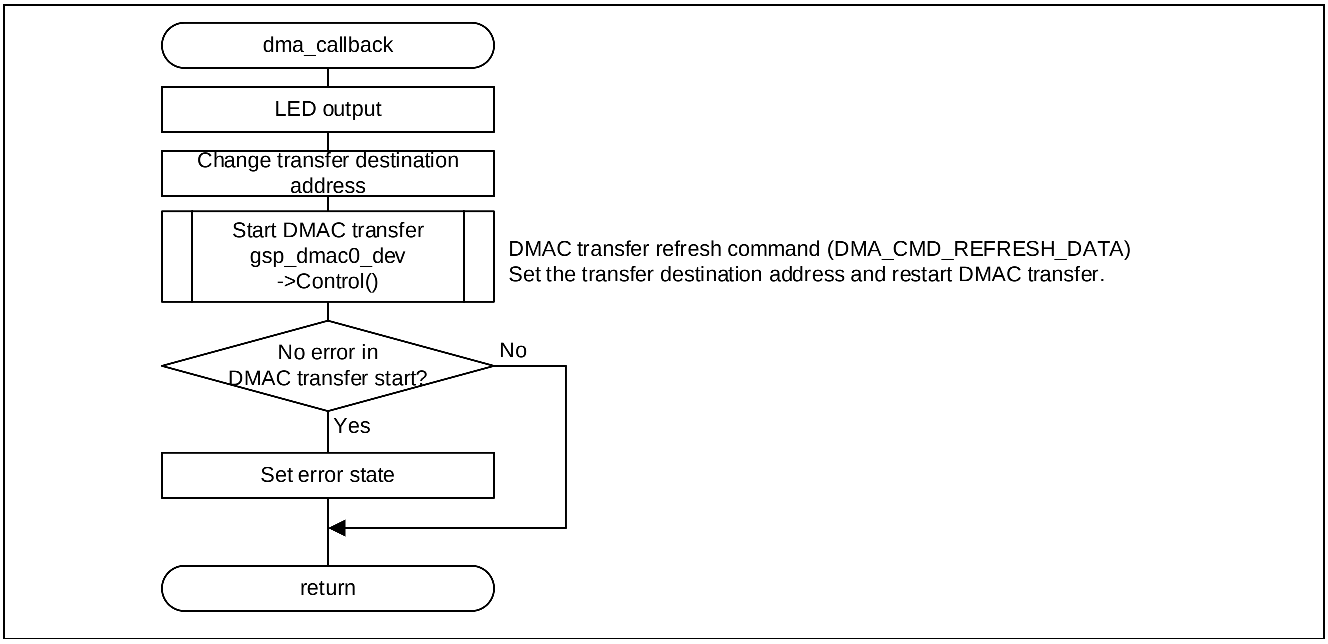


Figure 3.4 DMAC Transfer End Callback Processing

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.

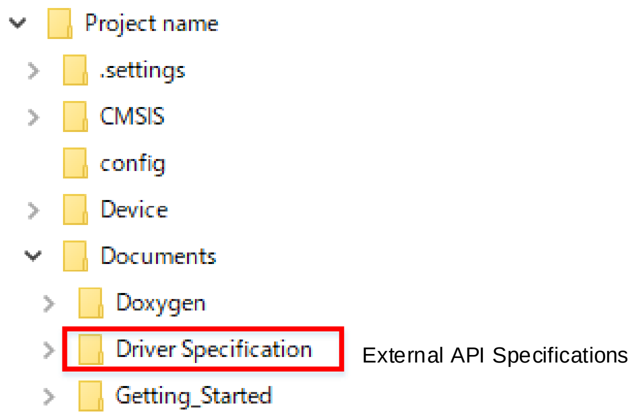


Figure 4-1 Location of External API Specifications

5. Usage Notes of R_DMAC Driver

This chapter covers the main points of concern regarding the R_I2C driver. Note: Not all the usage notes are given here in this note.

For additional information , see the external specification document described in "4.1 External Specification".

5.1 DMAC Interrupts

To use DMAC interrupts (transfer end interrupts or transfer escape interrupts), register them to the NVIC in r_system_cfg.c and then execute the InterruptEnable function.

Figure 5.1 shows an example of registering interrupts to the NVIC. **Figure 5.2** shows an example of enabling DMAC transfer end interrupts.

```
 . . .
#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 5.1 Example of Registering Interrupts to NVIC (Using DMAC0)

```
#include "r_dmac_api.h"
#include "r_dma_common_api.h"

static void dmac_callback(void);

// DMAC driver instance
extern DRIVER_DMA Driver_DMAC0;
static DRIVER_DMA *gsp_dmac0_dev = &Driver_DMAC0;

static const uint8_t gs_source_data[2] = {0xA0, 0x1B};
static uint8_t gs_dest_area[2] = {0};

main()
{
    IRQn_Type irq_type;
    st_dma_transfer_data_cfg_t config;

    /* Set parameters of Create process. */
    config.mode = (DMA_MODE_NORMAL | DMA_SIZE_BYTE |
                  DMA_SRC_INCR | DMA_DEST_INCR | DMA_REPEAT_BLOCK_SRC);
    /* Casting src_addr*/
    config.src_addr = (uint32_t)&gs_source_data[0];

    /* Casting dest_addr */
    config.dest_addr = (uint32_t)&gs_dest_area[0];
    config.transfer_count = 2;
    config.offset = 0;
    config.src_extended_repeat = 0;
    config.dest_extended_repeat = 0;

    (void) gsp_dmac0_dev ->Open();          /* DMAC driver is initialized */
    (void) gsp_dmac0_dev ->Create(SYSTEM_CFG_EVENT_NUMBER_PORT_IRQ4, &config);
                                         /* DMAC driver operation is set up */
    (void) gsp_ dmac0_dev->InterruptEnable(DMA_INT_COMPLETE, dmac_callback);
                                         /* DMAC transfer end interrupt is enabled */
    /* Set DMAC Transfer action source enable. */
    (void) gsp_dmac0_dev->Control(DMA_CMD_START, 0);  /* DMAC transfer is started */
    while(1);
}

/**************************************************************************************
* callback function
**************************************************************************************/
static void dmac_callback(void)
{
    /* Processing of DMAC transfer end interrupt is written */
}
```

Figure 5.2 Example of Enabling DMAC Interrupts

5.2 Transfer Escape Interrupt

If a transfer escape interrupt occurs, DMAC transfer is halted. To start transfer again, execute the Control function that takes the DMA_CMD_START command as an argument.

5.3 Setting Transfer Source and Transfer Destination

When the DMAC data transfer size is word (16 bits) (when DMA_SIZE_WORD is specified), specify the transfer destination and transfer source addresses so that bit 0 in the address is 0. When the DTC data transfer size is longword (32 bits) (when DMA_SIZE_LONG is specified), specify the transfer destination and transfer source addresses so that bits 0 and 1 in the address are 0.

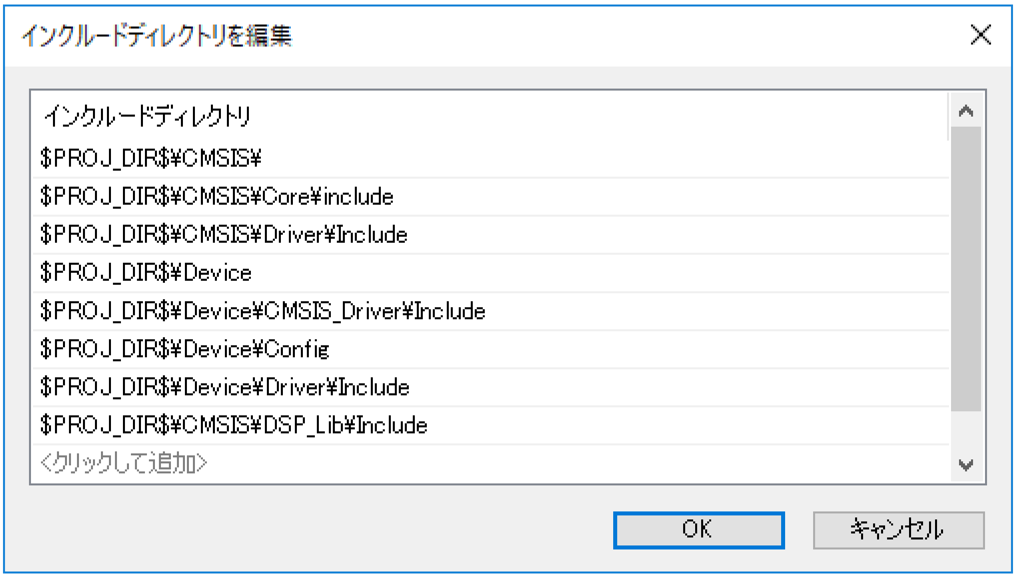
6. Troubleshooting

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 be 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.

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/.