DTC Block Transfer Sample Code (Using CMSIS Driver Package) for RE01 1500KB Group

DTC Sample Code Using CMSIS Driver Package

Summary

This application note describes the DTC Block Transfer sample code conforming the RE01 1500KB Group CMSIS driver package. This 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 DTC driver.	DTC	R_DTC

Target Device

RE01 1500KB Group

Note

When applying the sample code covered in this application note to another microcomputer, 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 RE01 1500KB Group(R01AN4660)

Contents

1. Specifications

1.1 Description of Project

A sample code project "an4704_hal_dtc_block_re" is provided with this application note.

The an4697_cmsis_iic_re project 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.

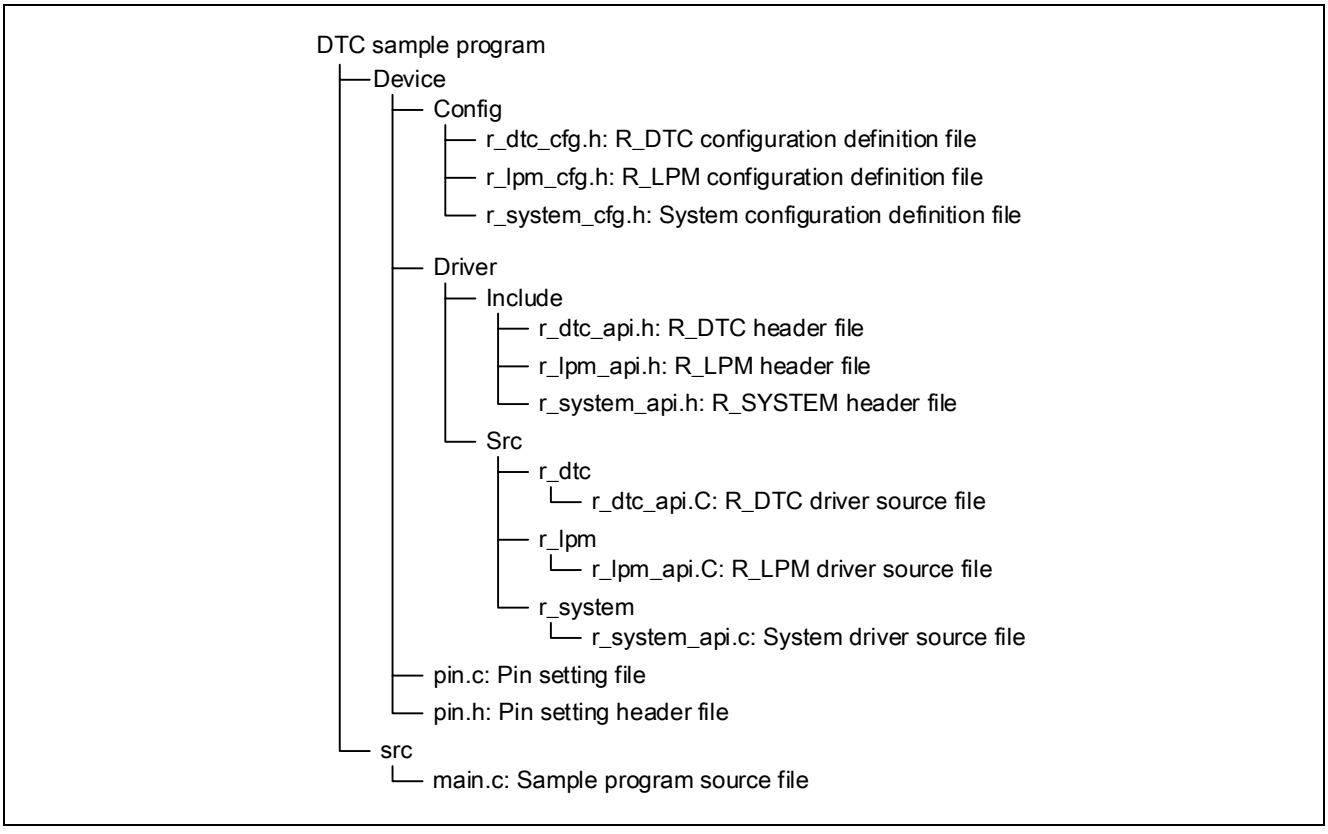


Figure 1-1 Folder Structure

1.4 File Configuration

Table 1-1 shows the files that are 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 to NVIC
r_dtc_cfg.h	DTC configuration	Optionally move code to SRAM

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 block transfer mode using the R_DTC driver.

It transfers 48 bytes of data from ROM to two areas (destinations A and B) of RAM in block transfer mode.

The sample code performs the following operations.

- After release from the reset state, the DTC transfer source is set to IRQ4, the transfer source address is set to ROM, and the transfer destination address is set to destination A of RAM.
- When SW2 (IRQ4) is pressed, 48 bytes of data is transferred from ROM to destination A of RAM.
- After DTC transfer has finished, the IRQ4 interrupt occurs and the DTC transfer end callback function is called.
- The transfer destination of the DTC is changed to destination B of RAM and the DTC is reactivated in the callback function. At the same time, LED0 is turned on and LED1 is turned off.
- When SW2 is pressed again, data is transferred from ROM to destination B of RAM. After completion of data transfer in block transfer mode, the callback function is called again. The transfer destination of the DTC is changed to destination A of RAM and the DTC is reactivated in the callback function. At the same time, LED0 is turned off and LED1 is turned on.

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

Item	Setting
Transfer area	ROM data is transferred to the RAM area
Transfer mode	Block transfer mode
Data transfer units	Single data unit: 1 byte (8 bits) Single block size: 16 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

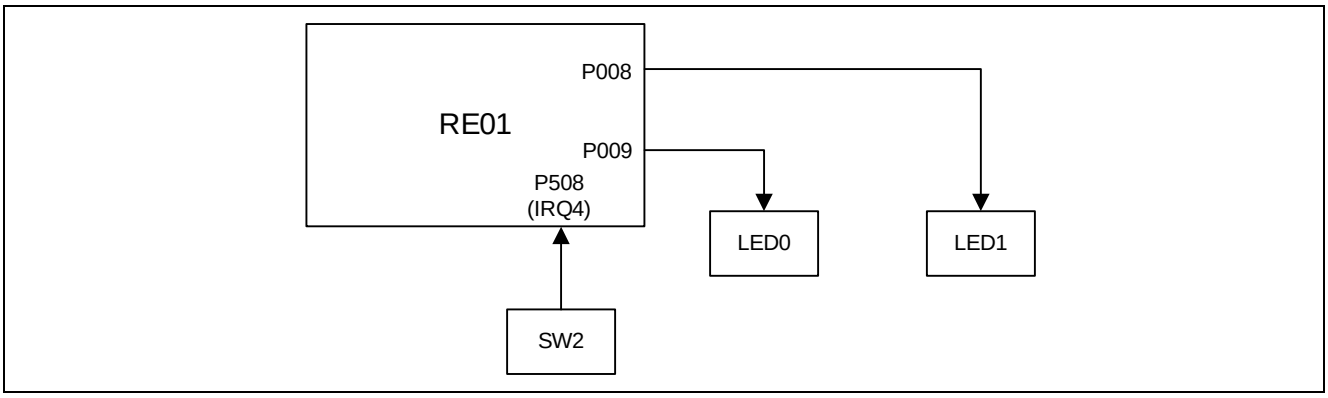


Figure 3-1 System Configuration

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 IRQ4 interrupts to NVIC	[r_system_cfg.h] SYSTEM_CFG_EVENT_ NUMBER_PORT_IRQ4	● 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. Then, it sets up DTC transfer and starts DTC transfer.
Argument	None
Return Value	None

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

irq4_enable	
Overview	IRQ4 interrupt enabling
Header	None
Declaration	static void irq4_enable(void)
Description	This function enables IRQ4 interrupt.
Argument	None
Return Value	None

irq4_disable	
Overview	IRQ4 interrupt disabling
Header	None
Declaration	static void irq4_disable(void)
Description	This function disables IRQ4 interrupt.
Argument	None
Return Value	None

dtc_callback	
Overview	DTC transfer end callback processing
Header	None
Declaration	static void dtc_callback(void)
Description	After the end of DTC block transfer, this function turns LED0 on and turns LED1 off if transfer to destination A has finished. Then, it changes the transfer destination RAM 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 RAM 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
DTC_BLOCK_SIZE	16	Number of blocks transferred in DTC transfer
DTC_REPEAT_TIMES	3	Number of repeat transfers

3.5 Flowcharts

Figure 3-2 shows a flowchart of the main processing.

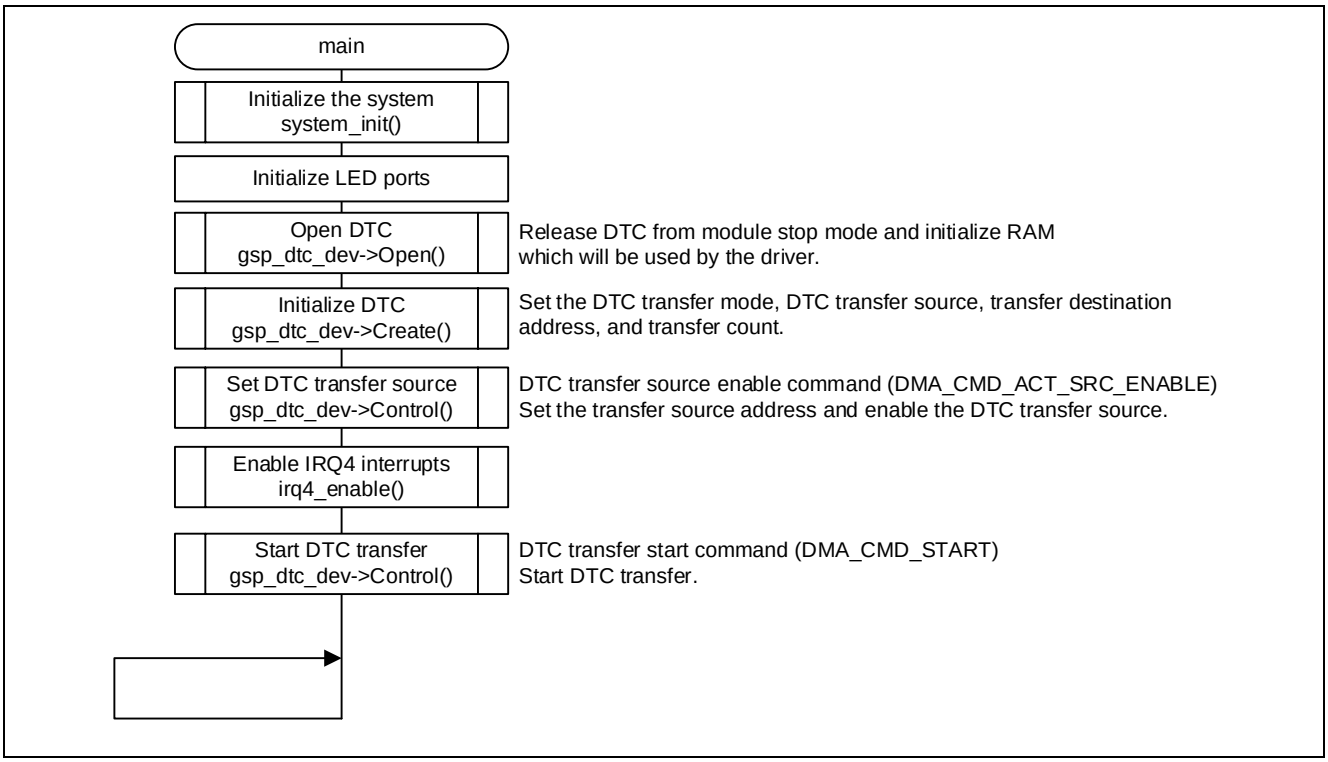


Figure 3-2 Main Processing

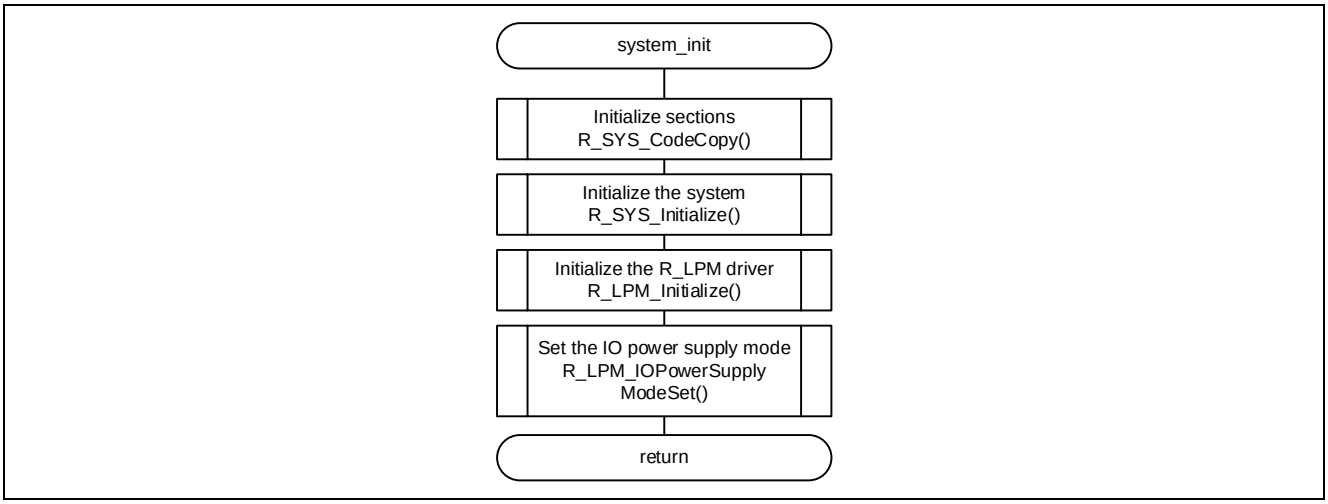


Figure 3-3 System Initialization Processing

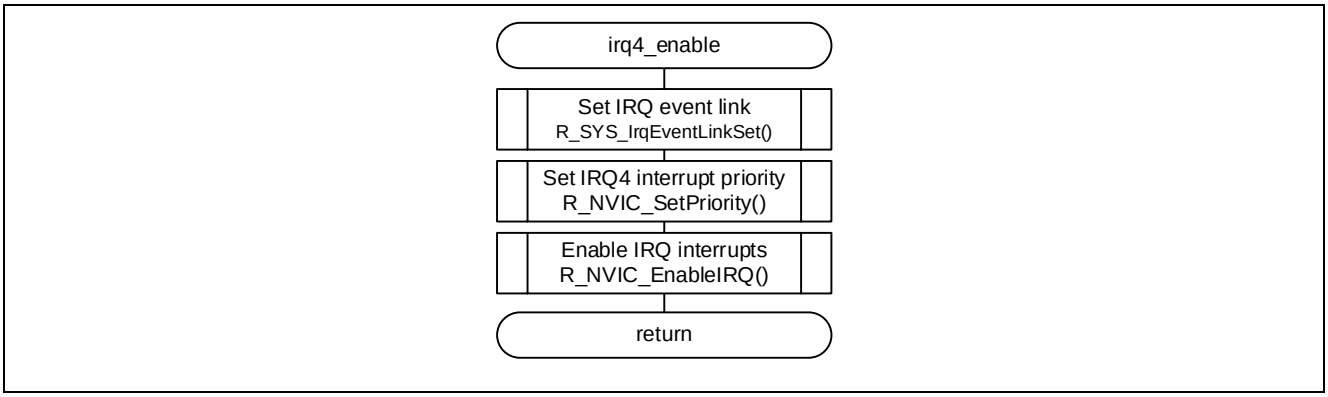


Figure 3-4 IRQ4 Interrupt Enabling

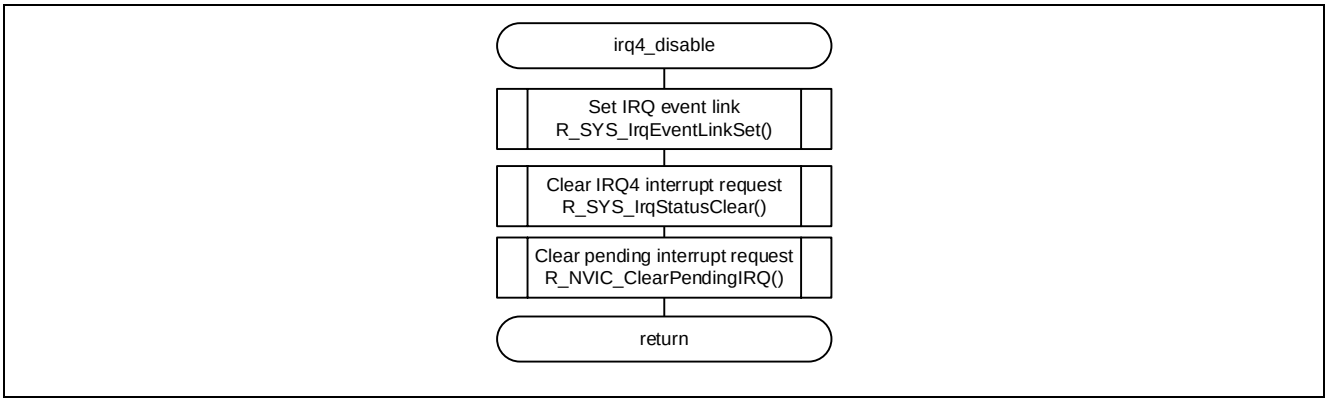


Figure 3-5 IRQ4 Interrupt Disabling

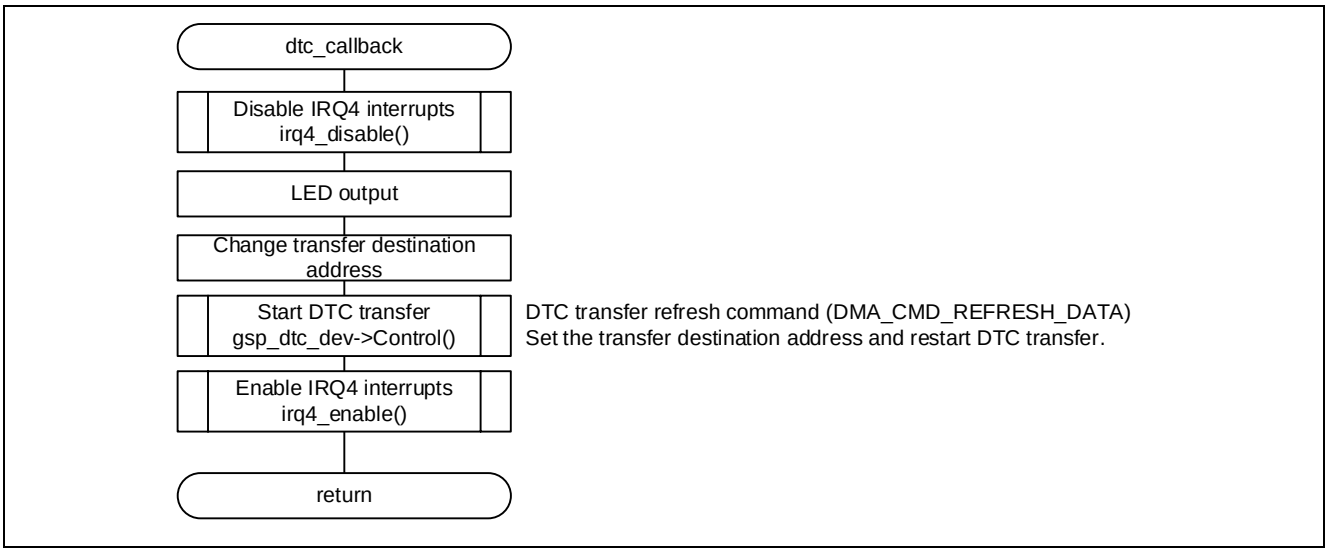


Figure 3-6 DTC Transfer End Callback Processing

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.

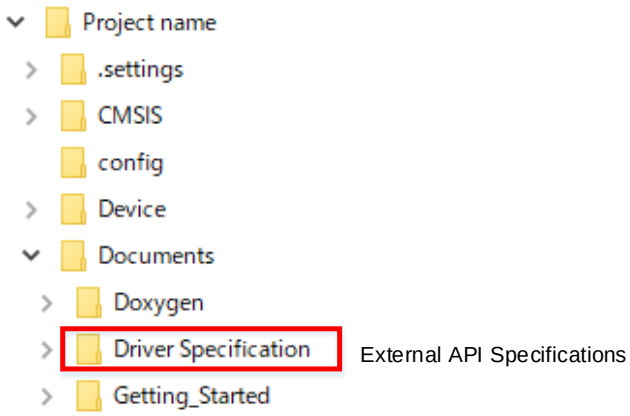


Figure 4-1 Location of External API Specifications

5. Usage Notes of R_DTC Driver

This chapter covers the main points to concern regarding the R_DTC driver. Note that not all notes are given here.

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

5.1 DTC Interrupts

When data transfer of the specified count is complete (DMA_INT_AFTER_ALL_COMPLETE is specified in the Create function) or when data transfer is complete (DMA_INT_PER_SINGLE_TRANSFER is specified in the Create function), an interrupt is signaled to the CPU. Two types of interrupt to the CPU are available: an interrupt triggered by DTC activation (for each channel) and an interrupt triggered by the event signal DTC_COMPLETE (common for all channels).

To use DTC_COMPLETE, register it 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 DTC interrupts.

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

```
#include "r_dtc_api.h"
#include "r_dma_common_api.h"

static void dtc_callback(void);

// DTC driver instance
extern DRIVER_DMA Driver_DTC;
static DRIVER_DMA *gsp_dtc_dev = Driver_DTC;

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

main()
{
    IRQn_Type irq_type;
    st_dma_transfer_data_cfg_t config;

    /* Set parameters of Create process. */
    config.mode = (DMA_MODE_NORMAL |
                   DMA_SIZE_WORD |
                   DMA_SRC_FIXED |
                   DMA_DEST_FIXED |
                   DMA_REPEAT_BLOCK_DEST |
                   DMA_INT_PER_SINGLE_TRANSFER |
                   DMA_CHAIN_DISABLE);
    /* 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 = 1;
    config.block_size = 0;
    config.offset = 0;
    config.src_extended_repeat = 0;
    config.dest_extended_repeat = 0;
    config.p_transfer_data = &transfer_data;     /* Start address of transfer information storage area */

    (void) gsp_dtc_dev->Open();              /* DTC driver is initialized */
    (void) gsp_dtc_dev->Create(SYSTEM_CFG_EVENT_NUMBER_PORT_IRQ4, &config);
                                        /* DTC driver operation is set up */
    (void) gsp_dtc_dev->InterruptEnable(DMA_INT_COMPLETE, dtc_callback);
                                        /* DTC transfer end interrupt is enabled */
    /* Set DTC Transfer action source enable. */
    irq_type = SYSTEM_CFG_EVENT_NUMBER_PORT_IRQ4;
    (void)gsp_dtc_dev->Control(DMA_CMD_ACT_SRC_ENABLE, &irq_type); /* DTC transfer source (IRQ4) */
                                                                   /* is enabled */
    (void) gsp_dtc_dev->Control(DMA_CMD_START, 0);  /* DTC transfer is started */
    while(1);
}

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

Figure 5-2 Example of Enabling DTC Interrupts

5.2 Setting Transfer Information

The DTC transfer information start address (p_transfer_data) must be word-aligned (set to a multiple of 4) when executing the Create function. Specify an address of 4n, which is a static variable.

Figure 5-1 shows possible and impossible patterns for setting transfer information.

Table 5-1 Possible and Impossible Patterns for Setting Transfer Information

Code Example	Judgment	Reason

main()
{
st_dma_transfer_data_cfg_t config;
st_dma_transfer_data_t transfer_data;
...
config.p_transfer_data = &transfer_data;
(void) gsp_dtc_dev->Create(DTC_EVENT, &config);
}
	Incorrect	Transfer information “config” is an auto / local variable and it may be invalid when the DTC is active.
		“transfer_data” is an auto / local variable and it may be invalid when the DTC is active.

uint8_t transfer_data[16];
 	
main()
{
st_dma_transfer_data_cfg_t config;
...
config.p_transfer_data = (st_dma_transfer_data_t *)transfer_data;
(void) gsp_dtc_dev->Create(DTC_EVENT, &config);
}
	Incorrect	The start address for “transfer_data” is not a word-aligned address.

st_dma_transfer_data_t transfer_data;
 	
main()
{
st_dma_transfer_data_cfg_t config;
...
config.p_transfer_data = &transfer_data;
(void) gsp_dtc_dev->Create(DTC_EVENT, &config);
}
	Possible (*)	Implicit alignment by the compiler is performed.

st_dma_transfer_data_t transfer_data __attribute__ (aligned (4));
 	
main()
{
st_dma_transfer_data_cfg_t config;
...
config.p_transfer_data = &transfer_data;
(void) gsp_dtc_dev->Create(DTC_EVENT, &config);
}
	Possible	Alignment to 4n addresses is explicitly specified.

st_dma_transfer_data_t transfer_data __attribute__ (aligned (4));
st_dma_transfer_data_cfg_t config;
 	
any_function() // other examples work only when executed in main()
{ ...
config.p_transfer_data = &transfer_data;
(void) gsp_dtc_dev->Create(DTC_EVENT, &config);
}
		Most reliable: Define “config” as a global variable.
		Force alignment for “transfer_data”.

DTC_EVENT stands for the NVIC registration number
(e.g., SYSTEM_CFG_EVENT_NUMBER_PORT_IRQ4) of the DTC transfer source.

Note. Note that the start address may not be a 4n address depending on the compiler type and optimization level.

5.3 Setting Transfer Source and Transfer Destination Addresses

When the DTC 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.

5.4 Occurrence of Transfer Request Source Interrupt before Setting up DTC Transfer

If an interrupt occurs in the DTC transfer request source before the Create function is executed or an interrupt reoccurs in the DTC transfer request source before the DTC is set up again in the callback function called after completion of DTC transfer, the interrupt request bit is set to "1" without DTC transfer being performed.

If DTC transfer request source interrupts are enabled, an interrupt will be generated at that point. (If the DTC was set up again in the callback function, etc., the callback function will be re-executed before DTC transfer is performed.)

If DTC transfer request source interrupts are disabled, the interrupt request bit remains set to "1" and DTC transfer is not performed.

Accordingly, avoid that DTC transfer request source interrupts occur before DTC transfer is set up.

Examples of initial setting and re-setting of the DTC are as follows.

[At initial setting]

(1) Disable DTC transfer request source interrupts. (*1)

(2) Clear the interrupt request of the DTC transfer request source. (*1)

(3) Initialize the DTC in the Open function.

(4) Specify the DTC transfer request source in the Create function.

(5) Execute the DTC transfer request enable command (DMA_CMD_ACT_SRC_ENABLE) in the Control function.

(6) Enable DTC transfer request source interrupts.

(7) Execute the DTC transfer start command (DMA_CMD_START) in the Control function.

Note 1. If the DTC transfer request source is not operating after release from the reset state, this processing is unnecessary.

[At re-setting]

(1) Disable DTC transfer request source interrupts.

(2) Clear the interrupt request of the DTC transfer request source.

(3) Execute the DTC transfer refresh command (DMA_CMD_REFRESH_DATA) in the Control function.

(4) Enable DTC transfer request source interrupts.

```
/* (1) Disable DTC transfer request source interrupts. */
R_SYS_IrqEventLinkSet(SYSTEM_CFG_EVENT_NUMBER_PORT_IRQ4, 0x00 , dtc_callback)

/* (2) Clear the interrupt request of the DTC transfer request source. */
R_SYS_IrqStatusClear(SYSTEM_CFG_EVENT_NUMBER_PORT_IRQ4);
R_NVIC_ClearPendingIRQ(SYSTEM_CFG_EVENT_NUMBER_PORT_IRQ4);

/* (3) Initialize the DTC in the Open function. */
(void) gsp_dtc_dev->Open();             /* DTC driver is initialized */

/* (4) Specify the DTC transfer request source in the Create function. */
(void) gsp_dtc_dev->Create(SYSTEM_CFG_EVENT_NUMBER_PORT_IRQ4, &config);
                                    /* DTC driver operation is set up */
/* (5) Execute the DTC transfer request enable command (DMA_CMD_ACT_SRC_ENABLE) in the */
/* Control function. */
(void)gsp_dtc_dev->Control(DMA_CMD_ACT_SRC_ENABLE, &irq_type); /* DTC transfer source (IRQ4)
is */
                                                            /* enabled */
/* (6) Enable DTC transfer request source interrupts. */
R_SYS_IrqEventLinkSet(SYSTEM_CFG_EVENT_NUMBER_PORT_IRQ4, 0x13, dtc_callback)

/* (7) Execute the DTC transfer start command (DMA_CMD_START) in the Control function. */
(void) gsp_dtc_dev->Control(DMA_CMD_START, 0);  /* DTC transfer is started */
```

Figure 5-3 Example of Initial Setting of DTC (When IRQ4 Is Used as DTC Transfer Request Source)

```
/* (1) Disable DTC transfer request source interrupts. */
R_SYS_IrqEventLinkSet(SYSTEM_CFG_EVENT_NUMBER_PORT_IRQ4, 0x00 , dtc_callback)

/* (2) Clear the interrupt request of the DTC transfer request source. */
R_SYS_IrqStatusClear(SYSTEM_CFG_EVENT_NUMBER_PORT_IRQ4);
R_NVIC_ClearPendingIRQ(SYSTEM_CFG_EVENT_NUMBER_PORT_IRQ4);

/* (3) Initialize the DTC in the Open function. */
(void) gsp_dtc_dev->Control(DMA_CMD_REFRESH_DATA, &config);     /* DTC driver is initialized
*/

/* (4) Enable DTC transfer request source interrupts. */
R_SYS_IrqEventLinkSet(SYSTEM_CFG_EVENT_NUMBER_PORT_IRQ4, 0x13, dtc_callback)
```

Figure 5-4 Example of Re-setting of DTC (When IRQ4 Is Used as DTC Transfer Request Source)

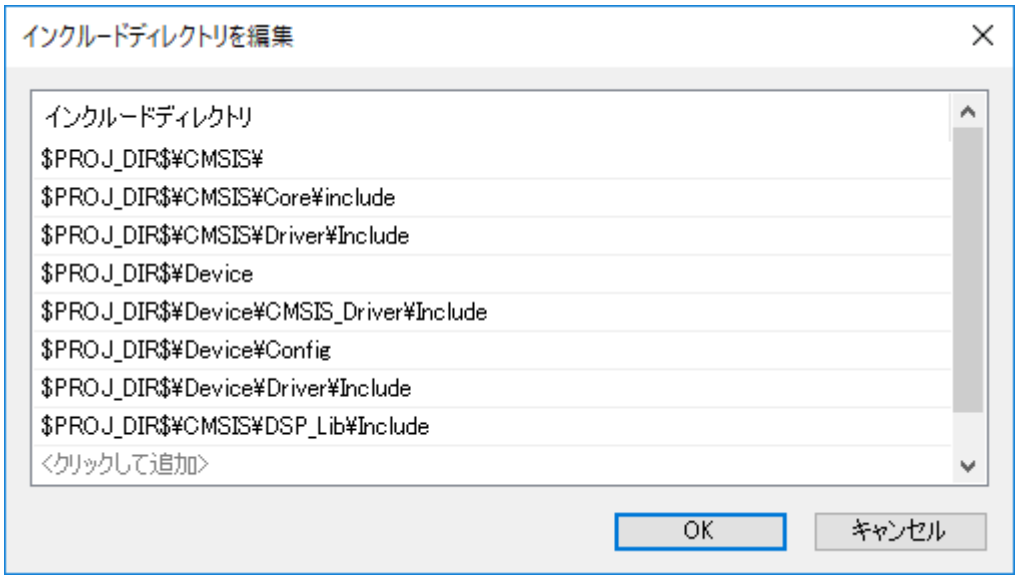
6. Troubleshooting

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. 19th, 19	—	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.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

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/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.