USART Asynchronous Communication Sample Code (Using CMSIS Driver Package) for RE01 1500KB Group, 256KB Group

USART Sample Code Using CMSIS Driver Package

Summary

This application note describes the USART Asynchronous Communication sample code conforming to the RE01 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 serial communication with the PC using the USART driver.	USART	R_USART

Target Device

RE01 1500KB Group

RE01 256KB 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

RE01 1500KB, 256KB Group Startup Guide to Development Using CMSIS Package (R01AN4660)

Contents

1. Specifications

1.1 Description of Project

The following sample code projects are provided with this application note.

Sample code project for RE01 1500KB group : an4699_cmsis_usart_uart_re

Sample code project for RE01 256KB group : r01an4699_cmsis_usart_uart_re_256kb

The an4699_cmsis_usart_uart_re project has been tested using the Evaluation Kit RE01 1500KB. This project is configured to match the settings of R7F0E015D2CFB mounted on the Evaluation Kit RE01 1500KB.

The r01an4699_cmsis_usart_uart_re_256kb project has been tested using the Evaluation Kit RE01 256KB. This project is configured to match the settings of R7F0E01182CFP mounted on the Evaluation Kit RE01 256KB.

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.

Table 1-1 The pins used by the sample code for RE01 1500KB group

Pin Used	Purpose of Use
P812	TXD4
P813	RXD4

Table 1-2 The pins used by the sample code for RE01 256KB group

Pin Used	Purpose of Use
P703	TXD0
P702	RXD0

1.3 Folder Structure

The folder structure of the sample code is shown below.

```
USART sample program
├─CMSIS
│  └─ Driver
│       └─ Include
│            └─ Driver_USART.h: CMSIS driver header file
├─Device
│  ├─ CMSIS_Driver
│  │   ├─ Include
│  │   │   ├─ r_usart_cmsis.h: R_USART driver header file
│  │   │   └─ R_Driver_USART.h: CMSIS driver extended header file
│  │   └─ Src
│  │       └─ r_usart_cmsis.c: R_USART driver source file
│  ├─ Config
│  │   ├─ r_usart_cfg.h: R_USART configuration definition file
│  │   ├─ r_lpm_cfg.h: R_LPM configuration definition file
│  │   └─ r_system_cfg.h: System configuration definition file
│  ├─ Driver
│  │   ├─ Include
│  │   │   ├─ r_lpm_api.h: R_LPM driver header file
│  │   │   └─ r_system_api.h: System driver header file
│  │   └─ Src
│  │       ├─ 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 1-3 shows the files that are added or modified for this sample code.

Table 1-3 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 SCI4
r_system_cfg.h	System configuration	Registering SCI4 transfer interrupt to NVIC

1.5 Option-Setting Memory

Table 1-4 shows the option-setting memory setting for the sample code. Set suitable values for a user system if required.

Table 1-4 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 2-2).

Table 2-1 Operating Conditions for RE01 1500KB group

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 clock 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 8.32.1
C compiler	GCC	GCC ARM Embedded Version 6.3.1.20170620 GNU 6-2017-q2-update
	IAR	IAR C/C++ Compiler for ARM Version 8.32.1
Debugger		Segger J-Link OB
I/O header Version		Rev1.00
Sample code Version		Rev1.00

Table 2-2 Operating Conditions for RE01 256KB group

Item		Description
Microcontroller used		R7F0E01182CFP 100pin
Operating frequency	PLL is selected as the system clock	• HOCO: 64MHz • System clock (ICLK): 64 MHz (HOCO) • Peripheral module clock A (PCLKA): 64 MHz (HOCO is not divided) • Peripheral module clock B (PCLKB): 32 MHz (HOCO is divided by 2)
Operating voltage		• 3.3V
Target board		Evaluation Kit RE01 256KB (RTK70E0118CXXXXXBJ)
Integrated Development Environment	GCC	Renesas e^2 studio 2020-07
	IAR	IAR Embedded Workbench for ARM Version 8.40.2
C compiler	GCC	GCC ARM Embedded Version 6.3.1.20170620 GNU 6-2017-q2-update
	IAR	IAR C/C++ Compiler for ARM Version 8.40.2
Debugger		Segger J-Link OB
I/O header Version		Rev1.00
Sample code Version		Rev1.03

3. Description of Software

This sample code performs asynchronous communication with the PC using the R_USART driver.

The sample code performs the following operations.

- After release from the reset state, sets up SCI communication. Then, starts reception and waits for communication from the PC.
- Receives 1 byte of data from the PC and determines whether it is a line feed code. When a line feed code has been detected or reception of 50 bytes of data has finished, the received data is sent to the PC.
- Restarts reception and waits for communication from the PC.

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

Item	Setting
Communication device	PC
Transfer mode	Transfer rate: 9600 bps Data length: 8 bits Parity: None Stop bit: 2 bits Flow control: None
Data transfer units	Single data unit: 1 byte (8 bits)

3.1 System Configuration

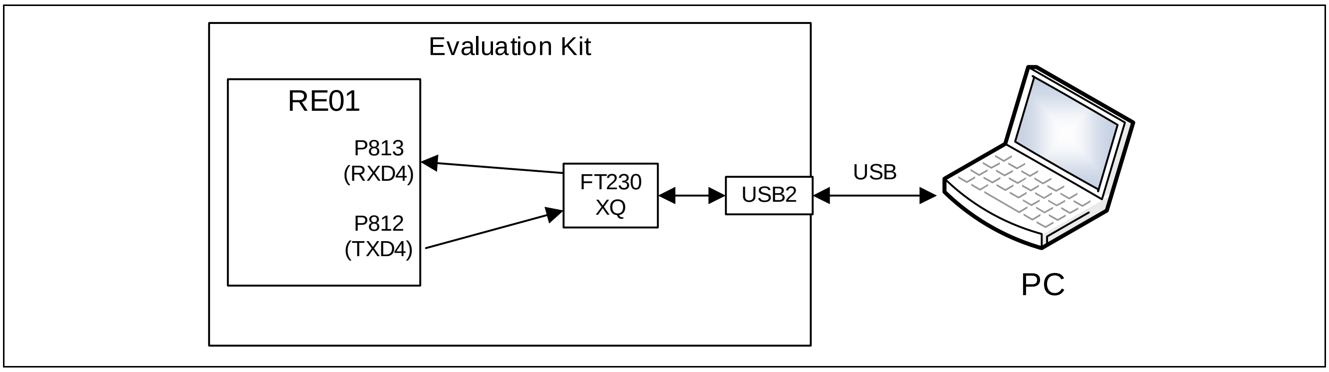


Figure 3.1 System Configuration for RE01 1500KB group

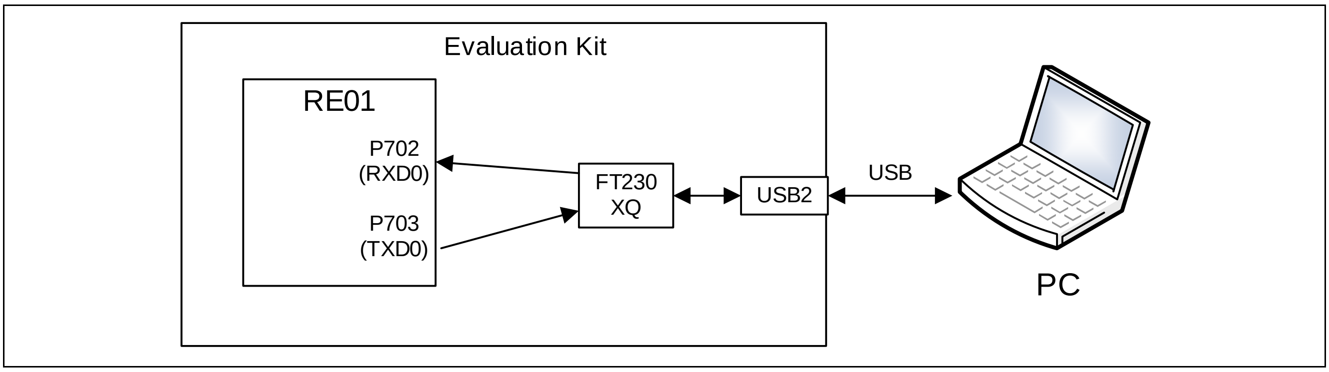


Figure 3.2 System Configuration for RE01 256KB group

3.2 Driver Configuration

3.2.1 Driver Configuration by the sample code for RE01 1500KB group

Table 3-2 Driver Configuration for 1500KB Group (1/2)

Item	Location of Change	Details of Change

Changing the CTS4 setting to unused	[pin.c]
	R_SCI_Pinset_CH4()
	function
		• The following was commented out.
		// PFS->P111PFS_b.ASEL = 0U;
		// PFS->P111PFS_b.ISEL = 0U;
		// PFS->P111PFS_b.PSEL
		= R_PIN_PRV_SCI_PSEL_04;
		// PFS->P111PFS_b.PMR = 1U;

Changing the TXD4 setting from default (P113) to P812	[pin.c]
	R_SCI_Pinset_CH4()
	function
		• The following was commented out.
		// PFS->P113PFS_b.ASEL = 0U;
		// PFS->P113PFS_b.ISEL = 0U;
		// PFS->P113PFS_b.PSEL
		= R_PIN_PRV_SCI_PSEL_04;
		// PFS->P113PFS_b.PMR = 1U;

		• The following was uncommented. PFS->P812PFS_b.ASEL = 0U; PFS->P812PFS_b.ISEL = 0U; PFS->P812PFS_b.PSEL = R_PIN_PRV_SCI_PSEL_04; PFS->P812PFS_b.PMR = 1U;

Table 3-3 Driver Configuration for 1500KB Group (2/2)

Item	Location of Change	Details of Change

Changing the RXD4 setting from default (P112) to P813	[pin.c]
	R_SCI_Pinset_CH4()
	function
		• The following was commented out.
		// PFS->P112PFS_b.ASEL = 0U;
		// PFS->P112PFS_b.ISEL = 0U;
		// PFS->P112PFS_b.PSEL
		= R_PIN_PRV_SCI_PSEL_04;
		// PFS->P112PFS_b.PMR = 1U;

		• The following was uncommented. PFS->P813PFS_b.ASEL = 0U; PFS->P813PFS_b.ISEL = 0U; PFS->P813PFS_b.PSEL = R_PIN_PRV_SCI_PSEL_04; PFS->P813PFS_b.PMR = 1U;

Changing the SCK4 setting to unused	[pin.c]
	R_SCI_Pinset_CH4()
	function
		• The following was commented out.
		// PFS->P108PFS_b.ASEL = 0U;
		// PFS->P108PFS_b.ISEL = 0U;
		// PFS->P108PFS_b.PSEL
		= R_PIN_PRV_SCI_PSEL_04;
		// PFS->P108PFS_b.PMR = 1U;

Registering SCI4 interrupts to NVIC	[r_system_cfg.h] SYSTEM_CFG_EVENT_ NUMBER_SCI4_RXI	• Setting change (SYSTEM_IRQ_EVENT_NUMBER4)
	[r_system_cfg.h] SYSTEM_CFG_EVENT_ NUMBER_SCI4_TXI	• Setting change (SYSTEM_IRQ_EVENT_NUMBER5)
	[r_system_cfg.h] SYSTEM_CFG_EVENT_ NUMBER_SCI4_ERI	• Setting change (SYSTEM_IRQ_EVENT_NUMBER7)

3.2.2 Driver Configuration by the sample code for RE01 256KB group

Table 3-4 Driver Configuration for 256KB Group

Item	Location of Change	Details of Change
Set P703 as TXD0 pin.	[pin.c] R_SCI_Pinset_CH0() function	• The following was uncommented. PFS->P703PFS_b.PMR = 0U; PFS->P703PFS_b.ASEL = 0U; PFS->P703PFS_b.ISEL = 0U; PFS->P703PFS_b.PSEL = R_PIN_PRV_SCI_PSEL_04; PFS->P703PFS_b.PMR = 1U;
Set P702 as RXD0 pin.	[pin.c] R_SCI_Pinset_CH0() function	• The following was uncommented. PFS->P702PFS_b.PMR = 0U; PFS->P702PFS_b.ASEL = 0U; PFS->P702PFS_b.ISEL = 0U; PFS->P702PFS_b.PSEL = R_PIN_PRV_SCI_PSEL_04; PFS->P702PFS_b.PMR = 1U;
Registering SCI0 interrupts to NVIC	[r_system_cfg.h] SYSTEM_CFG_EVENT_ NUMBER_SCI0_RXI	• Setting change (SYSTEM_IRQ_EVENT_NUMBER4)
	[r_system_cfg.h] SYSTEM_CFG_EVENT_ NUMBER_SCI0_TXI	• Setting change (SYSTEM_IRQ_EVENT_NUMBER5)
	[r_system_cfg.h] SYSTEM_CFG_EVENT_ NUMBER_SCI0_ERI	• Setting change (SYSTEM_IRQ_EVENT_NUMBER7)

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 USART operation and starts clock-synchronous transmission/reception. This function then confirms the received data and sends the data received so far back to the host device upon reception of a line feed code or 50 bytes of data.
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, the R_LPM driver, and calls the IO power supply setting function.
Argument	None
Return Value	None

usart_callback		
Overview	USART transfer end callback processing	
Header	None	
Declaration	static void usart_callback(uint32_t event)	
Description	This function executes processing for each event entered by an argument.	
Argument	uint32_t event	Cause of callback ARM_USART_EVENT_RECEIVE_COMPLETE Receive end ARM_USART_EVENT_SEND_COMPLETE Transmit end ARM_USART_EVENT_RX_OVERFLOW Overflow event ARM_USART_EVENT_RX_FRAMING_ERROR Framing error event ARM_USART_EVENT_RX_PARITY_ERROR Parity error
Return Value	None	

3.4 List of Constants

Table 3-5 shows a list of constants.

Table 3-5 Constants (User Changeable) Used in Sample Code

Constant Name	Setting	Description
RCV_CNT_MAX	50	Maximum data size received by USART

3.5 Flowcharts

Figure 3.3 shows a flowchart of the main processing.

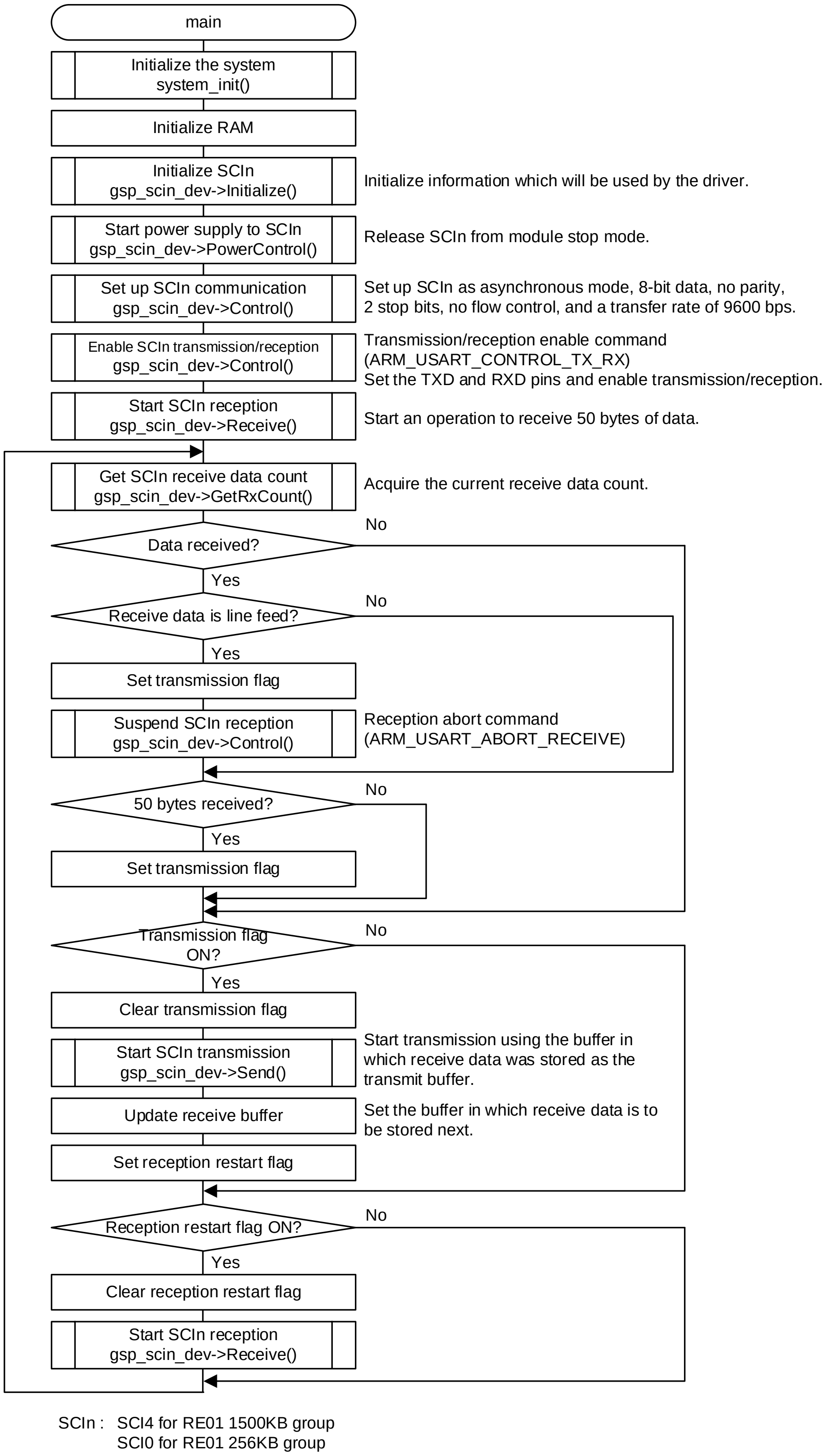


Figure 3.3 Main Processing

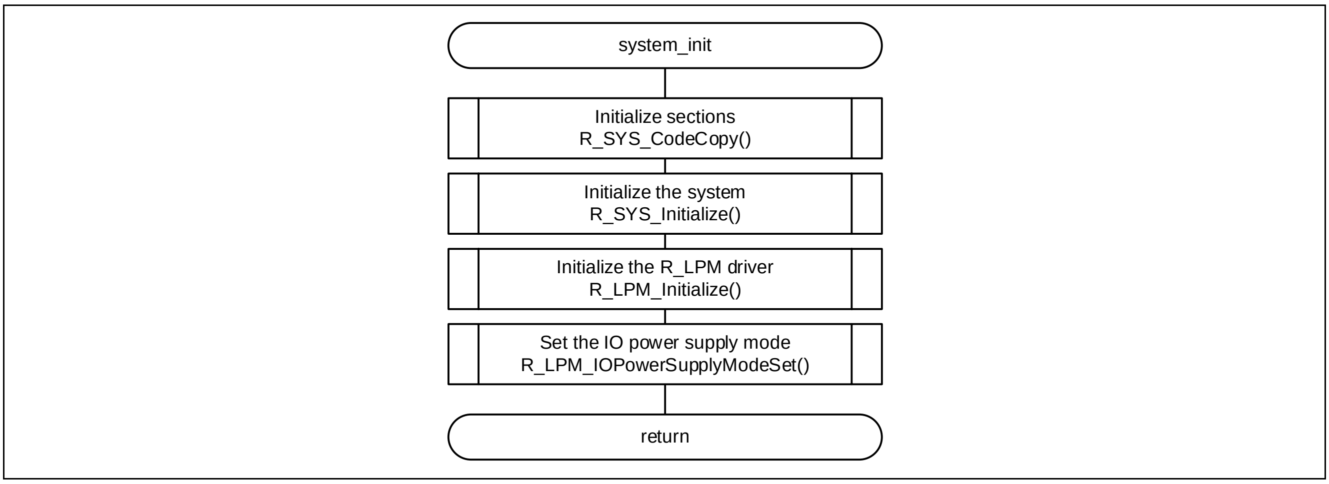


Figure 3.4 System Initialization Processing

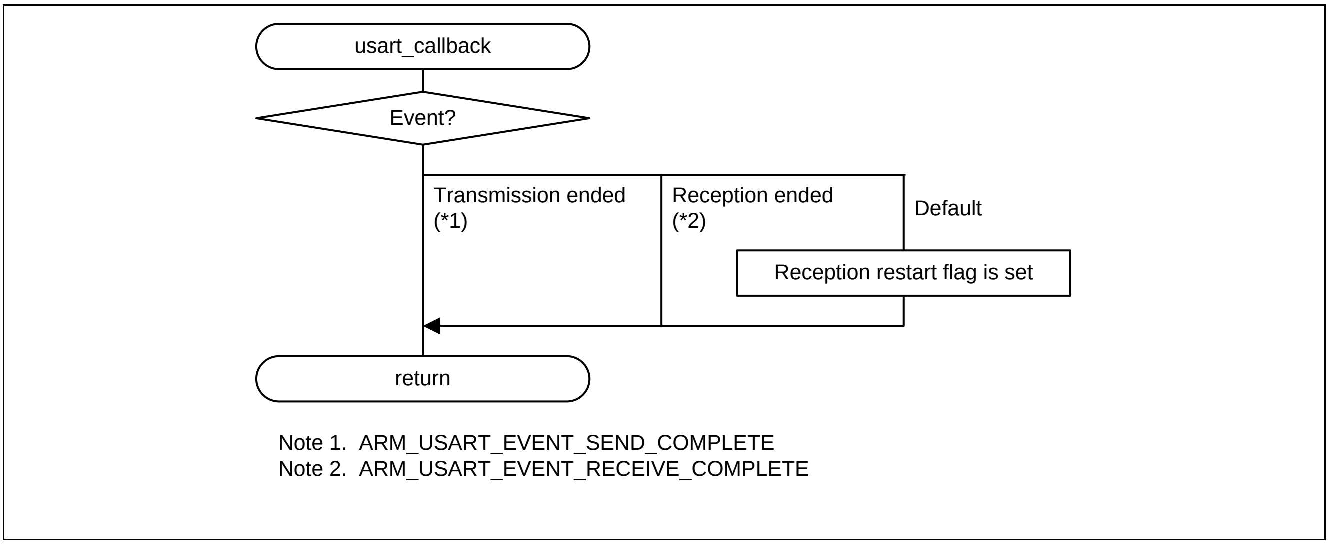


Figure 3.5 USART Transfer End Callback Processing

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.

5. Usage Notes of R_USART Driver

This chapter covers the main points regarding the R_USART driver. Note: not all the usage notes are given here in this note.

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

5.1 USART Communication Using DMAC or DTC

When transmitting or receiving data using the DMAC or DTC, change the settings for controlling transmission and reception in r_usart_cfg.h.

Table 5-1 shows the definitions of configuration parameters for transmission and reception control. Table 5-2 shows the definitions of values indicating methods of transmission and reception control.

Table 5-1 Definitions of Configuration Parameters for Transmission and Reception Control (n = 0 to 5, 9)

Definition	Initial Value	Description
SCIn_TRANSMIT_CONTROL	SCI_USED_INTERRUPT	Transmission control for SCIn (Initial value: Interrupt)
SCIn_RECEIVE_CONTROL	SCI_USED_INTERRUPT	Reception control for SCIn (Initial value: Interrupt)

Table 5-2 Definitions of Values Indicating Methods of Transmission and Reception Control

Definition	Value	Description
SCI_USED_INTERRUPT	(0)	An interrupt is used for transmission/reception control.
SCI_USED_DMAC0	(1<<0)	DMAC0 is used for transmission/reception control.
SCI_USED_DMAC1	(1<<1)	DMAC1 is used for transmission/reception control.
SCI_USED_DMAC2	(1<<2)	DMAC2 is used for transmission/reception control.
SCI_USED_DMAC3	(1<<3)	DMAC3 is used for transmission/reception control.
SCI_USED_DTC	(1<<15)	DTC is used for transmission/reception control.

5.2 Registering Interrupts to NVIC

When using the USART, register the interrupts to the NVIC in r_system_cfg.h.

Table 5-3 shows the definition 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 Each Intended Use

Mode	Intended Use	Definition of NVIC Registration	Remarks
Asynchronous	Used only for transmission	SYSTEM_CFG_EVENT_NUMBER_SCIn_TXI	(*1)
	Used only for reception	SYSTEM_CFG_EVENT_NUMBER_SCIn_RXI	(*2)
		SYSTEM_CFG_EVENT_NUMBER_SCIn_ERI	
	Used for transmission/ reception	SYSTEM_CFG_EVENT_NUMBER_SCIn_TXI	(*1)
		SYSTEM_CFG_EVENT_NUMBER_SCIn_RXI	(*2)
		SYSTEM_CFG_EVENT_NUMBER_SCIn_ERI	
Clock-synchronous/ Smart card interface	Used only for transmission	SYSTEM_CFG_EVENT_NUMBER_SCIn_TXI	(*1)
	Used for transmission/ reception or only for reception	SYSTEM_CFG_EVENT_NUMBER_SCIn_TXI	(*1)
		SYSTEM_CFG_EVENT_NUMBER_SCIn_RXI	(*2)
		SYSTEM_CFG_EVENT_NUMBER_SCIn_ERI	

Note 1. When DMACm (m = 0 to 3) is used for transmission control, register SYSTEM_CFG_EVENT_NUMBER_DMACm_INT to the NVIC.

Note 2. When DMACm (m = 0 to 3) is used for reception control, register SYSTEM_CFG_EVENT_NUMBER_DMACm_INT to the NVIC.

```
...
#define SYSTEM_CFG_EVENT_NUMBER_GPT_UVWEDGE
          (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)      /*!< Numbers 0/4/8/12/16/20/24/28 only */
#define SYSTEM_CFG_EVENT_NUMBER_SCI0_RXI
             (SYSTEM_IRQ_EVENT_NUMBER0)             /*!< Numbers 0/4/8/12/16/20/24/28 only */
#define SYSTEM_CFG_EVENT_NUMBER_SCI0_AM
              (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)  /*!< Numbers 0/4/8/12/16/20/24/28 only */
...
#define SYSTEM_CFG_EVENT_NUMBER_GPT2_CCMPB
           (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)      /*!< Numbers 1/5/9/13/17/21/25/29 only */
#define SYSTEM_CFG_EVENT_NUMBER_SCI0_TXI
             (SYSTEM_IRQ_EVENT_NUMBER1)             /*!< Numbers 1/5/9/13/17/21/25/29 only */
#define SYSTEM_CFG_EVENT_NUMBER_SPI0_SPTI
            (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)     /*!< Numbers 1/5/9/13/17/21/25/29 only */
...
#define SYSTEM_CFG_EVENT_NUMBER_GPT2_UDF
             (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)    /*!< Numbers 3/7/11/15/19/23/27/31 only */
#define SYSTEM_CFG_EVENT_NUMBER_SCI0_ERI
             (SYSTEM_IRQ_EVENT_NUMBER3)             /*!< Numbers 3/7/11/15/19/23/27/31 only */
#define SYSTEM_CFG_EVENT_NUMBER_SPI0_SPEI
            (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)     /*!< Numbers 3/7/11/15/19/23/27/31 only */
...
```

Figure 5.1 Example of Registering Interrupts to NVIC (When Using SCI0)

5.3 Setup for Reception in Clock-Synchronous or Smart Card Interface Mode

When performing simultaneous transmission and reception in the clock-synchronous mode, enable TE and RE in the Control function of the USART following the procedure below. TE and RE need to be enabled simultaneously due to a hardware restriction. When the procedure below is not followed, only the setting made first will be effective.

Even when only performing reception, please follow the same procedure because dummy data will be written.

Figure 5.2 shows an example of setup for performing reception in the clock-synchronous mode.

```
#include " R_Driver_USART.h"

static void usart_callback(uint32_t event);

// USART driver instance ( SCI0 )
extern ARM_DRIVER_USART Driver_USART0;
static ARM_DRIVER_USART *gsp_sci0_dev = &Driver_USART0;

// Receive data
static uint8_t rx_data[6];

main()
{
    uint32_t arg;
    /* Clock synchronous operation in master mode */
    arg = ARM_USART_MODE_SYNCHRONOUS_MASTER |
          ARM_USART_CPOL0 | ARM_USART_CPHA0 |
          ARM_USART_FLOW_CONTROL_NONE;

    (void)gsp_sci0_dev->Initialize(usart_callback);
                                        /* USART driver is initialized */
    (void)gsp_sci0_dev->PowerControl(ARM_POWER_FULL);
                                        /* USART is released from module stop mode */
    (void)gsp_sci0_dev->Control(arg, 100000);
                                        /* Clock synchronous operation in master */
                                        /* mode (100 kbps) */
    (void)gsp_sci0_dev->Control(ARM_USART_CONTROL_TX_RX,1);
                                                      /* Transmission and reception are */
                                                      /* enabled */

    (void) gsp_sci0_dev->Receive(rx_data, 6);  /* Reception is started */
    while(1);
}

/********************************************************************************
* callback function
*********************************************************************************/
static void usart_callback(uint32_t event)
{
    if (0 != (event & ARM_USART_EVENT_RECEIVE_COMPLETE))
    {
        /* Processing for successful reception is written */
    }
    else
    {
        /* Processing for communication error is written */
    }
}
```

Figure 5.2 Example of Setup for Reception in the Clock-Synchronous Mode

5.4 Pin Settings

The pins to be used by this driver will be set when transmission or reception has been enabled in the Control function. To change the pins to be used, modify code in the R_SCI_Pinset_CHn and R_SCI_Pinclr_CHn (n = 0 to 5, 9) functions of pin.c.

Figure 5.3, Figure 5.4, and Figure 5.5 show an example of changing pin settings, in which SCI0 is used in the asynchronous mode. In this example, the TXD0 pin is changed to P703, the RXD0 pin is changed to P702, and the CTS0 and SCK0 pin settings are changed so they are not used.

```
/**********************************************************************//**
* @brief This function sets Pin of SCI0.
***********************************************************************/
/* Function Name : R_SCI_Pinset_CH0 */
void R_SCI_Pinset_CH0(void)  // @suppress("API function naming") @suppress("Function length")
{
    /* Disable protection for PFS function (Set to PWPR register) */
    R_SYS_RegisterProtectDisable(SYSTEM_REG_PROTECT_MPC);

//    /* CTS0 : P107 */
//    PFS->P107PFS_b.PMR  = 0U;
//    PFS->P107PFS_b.ASEL = 0U;
//    PFS->P107PFS_b.ISEL = 0U;
//    PFS->P107PFS_b.PSEL = R_PIN_PRV_SCI_PSEL_04;
//    PFS->P107PFS_b.PMR  = 1U;

...

    /* TXD0 : P013 */
//    PFS->P013PFS_b.ASEL = 0U;
//    PFS->P013PFS_b.ISEL = 0U;

    /* When using SCI in I2C mode, set the pin to NMOS Open drain. */
////    PFS->P013PFS_b.NCODR = 1U;
////    PFS->P013PFS_b.PCODR = 0U;
//    PFS->P013PFS_b.PSEL = R_PIN_PRV_SCI_PSEL_04;
//    PFS->P013PFS_b.PMR  = 1U;

/* Set P703 as TXD0 pin.*/
    /* TXD0 : P703 */
    PFS->P703PFS_b.ASEL = 0U;
    PFS->P703PFS_b.ISEL = 0U;

    /* When using SCI in I2C mode, set the pin to NMOS Open drain. */
//    PFS->P703PFS_b.NCODR = 1U;
//    PFS->P703PFS_b.PCODR = 0U;
    PFS->P703PFS_b.PSEL = R_PIN_PRV_SCI_PSEL_04;
    PFS->P703PFS_b.PMR  = 1U;

//    /* RXD0 : P105 */
//    PFS->P105PFS_b.PMR  = 0U;
//    PFS->P105PFS_b.ASEL = 0U;
//    PFS->P105PFS_b.ISEL = 0U;

//    /* When using SCI in I2C mode, set the pin to NMOS Open drain. */
////    PFS->P105PFS_b.NCODR = 1U;
////    PFS->P105PFS_b.PCODR = 0U;
//    PFS->P105PFS_b.PSEL = R_PIN_PRV_SCI_PSEL_04;
//    PFS->P105PFS_b.PMR  = 1U;

...
```

Figure 5.3 Example of Changing Pin Settings (1/3)

```
/* Set P703 as TXD0 pin.*/
    /* RXD0 : P702 */
    PFS->P702PFS_b.ASEL = 0U;
    PFS->P702PFS_b.ISEL = 0U;

    /* When using SCI in I2C mode, set the pin to NMOS Open drain. */
//    PFS->P702PFS_b.NCODR = 1U;
//    PFS->P702PFS_b.PCODR = 0U;
    PFS->P702PFS_b.PSEL = R_PIN_PRV_SCI_PSEL_04;
    PFS->P702PFS_b.PMR  = 1U;

//    /* SCK0 : P104 */
//    PFS->P104PFS_b.PMR  = 0U;
//    PFS->P104PFS_b.ASEL = 0U;
//    PFS->P104PFS_b.ISEL = 0U;
//    PFS->P104PFS_b.PSEL = R_PIN_PRV_SCI_PSEL_04;
//    PFS->P104PFS_b.PMR  = 1U;

    /* SCK0 : P015 */
//    PFS->P015PFS_b.ASEL = 0U;
//    PFS->P015PFS_b.ISEL = 0U;
//    PFS->P015PFS_b.PSEL = R_PIN_PRV_SCI_PSEL_04;
//    PFS->P015PFS_b.PMR  = 1U;

    /* SCK0 : P700 */
//    PFS->P700PFS_b.ASEL = 0U;
//    PFS->P700PFS_b.ISEL = 0U;
//    PFS->P700PFS_b.PSEL = R_PIN_PRV_SCI_PSEL_04;
//    PFS->P700PFS_b.PMR  = 1U;

    /* Enable protection for PFS function (Set to PWPR register) */
    R_SYS_RegisterProtectEnable(SYSTEM_REG_PROTECT_MPC);

}/* End of function R_SCI_Pinset_CH0() */
```

Figure 5.4 Example of Changing Pin Settings (2/3)

```
/***************************************************************************//**
* @brief This function clears the pin setting of SCI0.
*******************************************************************************/
/* Function Name : R_SCI_Pinclr_CH0 */
void R_SCI_Pinclr_CH0(void)  // @suppress("API function naming")
{
    /* Disable protection for PFS function (Set to PWPR register) */
    R_SYS_RegisterProtectDisable(SYSTEM_REG_PROTECT_MPC);

//    /* SCK0 : P104 */
//    PFS->P104PFS &= R_PIN_PRV_CLR_MASK;

    /* SCK0 : P015 */
//    PFS->P015PFS &= R_PIN_PRV_CLR_MASK;

    /* SCK0 : P700 */
//    PFS->P700PFS &= R_PIN_PRV_CLR_MASK;

//    /* RXD0 : P105 */
//    PFS->P105PFS &= R_PIN_PRV_CLR_MASK;

    /* RXD0 : P014 */
//    PFS->P014PFS &= R_PIN_PRV_CLR_MASK;

/* Set P702 as RXD0 pin.*/
    /* RXD0 : P702 */
    PFS->P702PFS &= R_PIN_PRV_CLR_MASK;

//    /* TXD0 : P106 */
//    PFS->P106PFS &= R_PIN_PRV_CLR_MASK;

    /* TXD0 : P013 */
//    PFS->P013PFS &= R_PIN_PRV_CLR_MASK;

/* Set P703 as TXD0 pin.*/
    /* TXD0 : P703 */
    PFS->P703PFS &= R_PIN_PRV_CLR_MASK;

//    /* CTS0 : P107 */
//    PFS->P107PFS &= R_PIN_PRV_CLR_MASK;

    /* CTS0 : P500 */
//    PFS->P500PFS &= R_PIN_PRV_CLR_MASK;

    /* CTS0 : P704 */
//    PFS->P704PFS &= 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_SCI_Pinclr_CH0() */
```

Figure 5.5 Example of Changing Pin Settings (3/3)

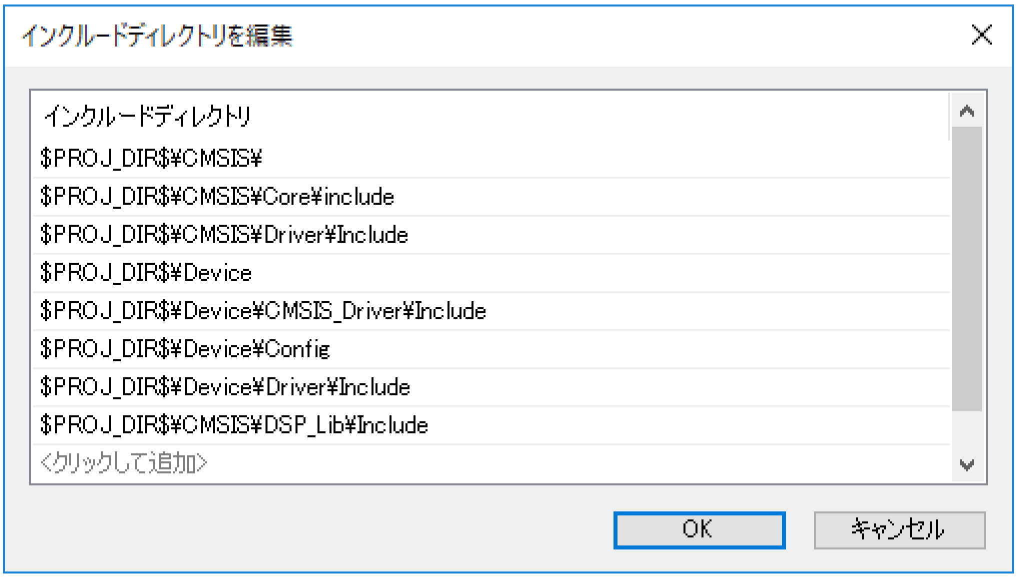
6. Troubleshooting

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.

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

RE01 256KB Group User's Manual: Hardware R01UH0894

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

RE01 1500KB, 256KB CMSIS Package Startup Guide

RE01 1500KB, 256KB 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
1.01	Jan.20.2020	—	Add RE01 256KB group.
1.02	Mar.19.2020	3,6,7 — Program (256KB)	RE01 256KB group target board changed to Evaluation Kit RE01 256KB Clerical error correction Replaced CMSIS Driver Package - RE01 256KB: CMSIS Driver Package Rev.0.80
1.03	Jun.01.2020	3 6 Program (256KB)	Change the name of the sample code project Updated "Operating Conditions" Replaced CMSIS Driver Package - RE01 256KB: CMSIS Driver Package Rev.1.00

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

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/.