RX Family

Simple I^{2}C Module for EEPROM Access
Using Firmware Integration Technology

R01AN1811EJ0131
Rev. 1.31
Feb. 1, 2019

Introduction

This application note describes the simple I^{2}C module for EEPROM access using firmware integration technology (FIT) for programming and reading an EEPROM using the serial communications interface (SCI).

Target Device

- RX110 Group
- RX111 Group
- RX113 Group

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Related Documents

For additional information associated with this document, refer to the following application notes.

- Firmware Integration Technology User's Manual Rev.1.00 (R01AN1833EU)
- Board Support Package Module Using Firmware Integration Technology Rev.2.70 (R01AN1685EU)
- Adding Firmware Integration Technology Modules to Projects Rev.1.10 (R01AN1723EU)
- Adding Firmware Integration Technology Modules to CubeSuite+ Projects Rev. 1.00 (R01AN1826EJ)
- RX Family Simple I^{2}C Module Using Firmware Integration Technology Rev. 1.30 (R01AN1691EJ)

Contents

1. Overview

The simple I^{2}C module for EEPROM access using firmware integration technology (EEPROM access (simple I^{2}C) FIT module (1)) provides the API functions to program and read an EEPROM using the SCI. The simple I^{2}C module using FIT (SCI simple I^{2}C mode FIT module) is used for the SCI simple I^{2}C mode which is in compliance with single master mode of the NXP I^{2}C-bus (Inter-IC-Bus) interface.

Note:

1. When the description says only “module” in this document, it indicates the EEPROM access (simple I^{2}C) FIT module.

Features supported by this module are as follows:

- Programming and reading an EEPROM
- Supports EEPROMs with 1-byte and 2-byte memory addresses.
- Supports EEPROMs whose one block size is up to 255 bytes.
- Multiple EEPROMs can be accessed.
- Communication mode can be standard or fast mode and the maximum communication rate is 384 kbps.

Limitations

This module has the following limitations:

- This module does not support EEPROMs with the 10-bit device address.
- Limitations on the SCI simple I^{2}C mode FIT module apply.

1.1 EEPROM Access (Simple I^{2}C) FIT Module

This module is implemented in a project and used as the API. Refer to 2.10 Adding the FIT Module to Your Project for details on implementing the module to the project.

1.2 Outline of the API

Table 1.1 lists the API Functions and Table 1.2 lists the Required Memory Size.

Table 1.1 API Functions

Item	Contents
R_EEPROM_SCI_IIC_Open()	Initializes the SCI to get this module ready for communication.
R_EEPROM_SCI_IIC_Write()	Starts programming an EEPROM.
R_EEPROM_SCI_IIC_Read()	Starts reading an EEPROM.
R_EEPROM_SCI_IIC_Advance()	Proceeds with programming or reading an EEPROM.
R_EEPROM_SCI_IIC_Close()	Releases the SCI channel used.
R_EEPROM_SCI_IIC_GetVersion()	Returns the module version.

Table 1.2 Required Memory Size

Memory Used	Size	Remarks
ROM	1393 bytes	Aside from the memories specified here, memories for the SCI simple I^{2}C mode FIT module are required. Refer to the SCI simple I^{2}C mode FIT Module application note for the required memories.
RAM	12 bytes + 96 bytes × number of channels	
Maximum user stack usage	136 bytes	
Maximum interrupt stack usage	212 bytes	

* The configuration options when measuring each memory size are default values listed in 2.6 Configuration Overview.

* The table lists values when the default values are set to the compile options.
The required memory size varies depending on the C compiler version and compile options.

1.3 Overview of the EEPROM Access (Simple I^{2}C) FIT Module

1.3.1 Specifications of the EEPROM Access (Simple I^{2}C) FIT Module

- This module supports programming and reading an EEPROM.
  - Programming an EEPROM is described in section 1.4.1.
  - Reading an EEPROM is described in section 1.4.2.
- The memory address can be specified with 1 byte or 2 bytes of data (00h to FFh or 0000h to FFFFh). When the device address includes the memory address in the EEPROM used, refer to the note described in 1.3.2.
- When programming or reading an EEPROM, the total number of bytes and 1 block size of the EEPROM need to be specified on the call of the function to start programming or reading. A start condition or stop condition is generated every block according to the setting. Data size for one block (page size) can be specified up to 255 bytes.
- When a NACK is detected, retry is performed. Refer to 2.6 Configuration Overview for details on the following retry settings.
  - Wait time before performing retry can be specified.
  - The number of retries can be specified.
- Multiple EEPROMs on the same channel bus can be controlled. However, while communication is in progress (the period from start condition generation to stop condition generation), communication with another device is not available. Figure 1.1 shows an Example of Controlling Multiple EEPROMs.

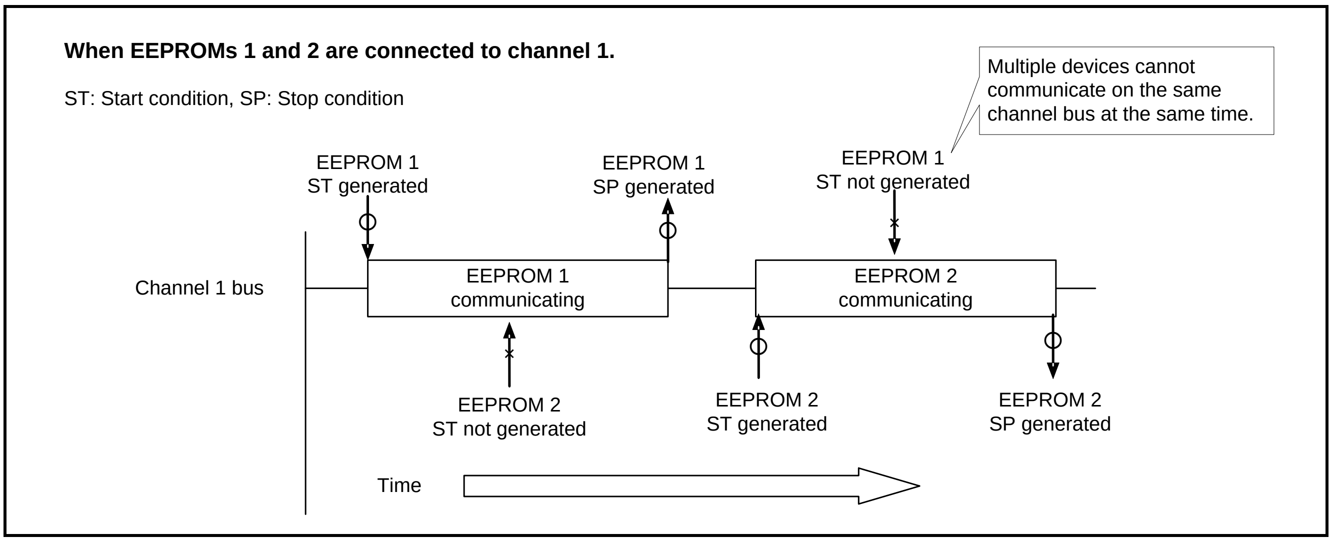
When EEPROMs 1 and 2 are connected to channel 1.
ST: Start condition, SP: Stop condition
Multiple devices cannot communicate on the same channel bus at the same time.
EEPROM 1 ST generated
EEPROM 1 SP generated
EEPROM 1 ST not generated
Channel 1 bus
EEPROM 1 communicating
EEPROM 2 communicating
EEPROM 2 ST not generated
EEPROM 2 ST generated
EEPROM 2 SP generated
Time

Figure 1.1 Example of Controlling Multiple EEPROMs

1.3.2 Note when Using an EEPROM with the Memory Address Included in the Device Address

This module cannot update the memory address included in the device address automatically. For example, when 3 bits out of 11 bits of the memory address are included in the device address (a8 to a10 in Figure 1.2), specify the memory address size to 1 byte. Then specify the following so that the 1-byte memory address does not overflow.

- Memory address
- Block size for programming or reading
- Total data size for programming or reading

After the above have been specified and a programming or reading has completed, update the memory address in the device address with the callback function or other method, and perform a programming or reading repeatedly.

Figure 1.2 shows an Example when the Memory Address is Included in the Device Address.

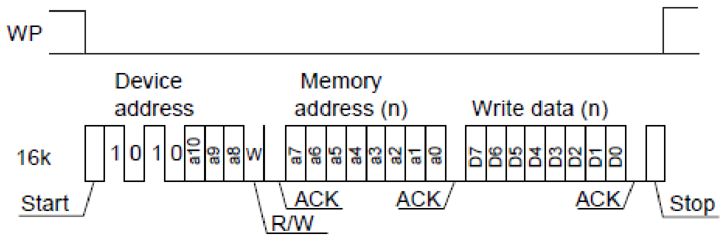

Figure 1.2 Example when the Memory Address is Included in the Device Address

1.4 Operation Examples

1.4.1 Programming an EEPROM

Figure 1.3 shows Programming an EEPROM. The callback function is called when all data have been programmed or a timeout has occurred. Set the function name to "callbackfunc" which is a member of the EEPROM information structure.

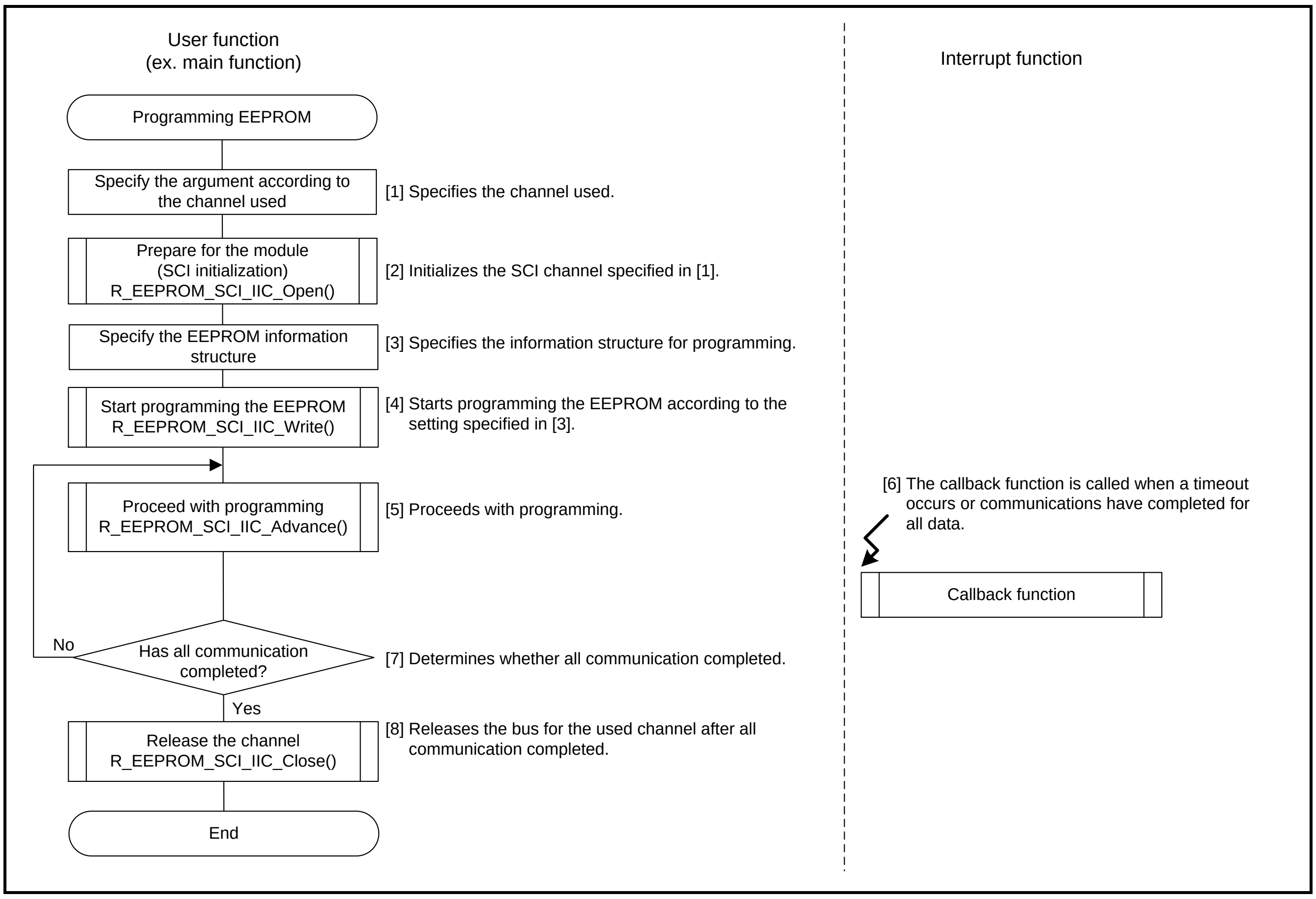
User function
(ex. main function)
Interrupt function
Programming EEPROM
Specify the argument according to the channel used
[1] Specifies the channel used.
Prepare for the module (SCI initialization) R_EEPROM_SCI_IIC_Open()
[2] Initializes the SCI channel specified in [1].
Specify the EEPROM information structure
[3] Specifies the information structure for programming.
Start programming the EEPROM R_EEPROM_SCI_IIC_Write()
[4] Starts programming the EEPROM according to the setting specified in [3].
Proceed with programming R_EEPROM_SCI_IIC_Advance()
[5] Proceeds with programming.
[6] The callback function is called when a timeout occurs or communications have completed for all data.
Callback function
No
Has all communication completed?
[7] Determines whether all communication completed.
Yes
Release the channel R_EEPROM_SCI_IIC_Close()
[8] Releases the bus for the used channel after all communication completed.
End

Figure 1.3 Programming an EEPROM

1.4.2 Reading an EEPROM

Figure 1.4 shows Reading an EEPROM. The callback function is called when all data have been read or a timeout has occurred. Set the function name to “callbackfunc” which is a member of the EEPROM information structure.

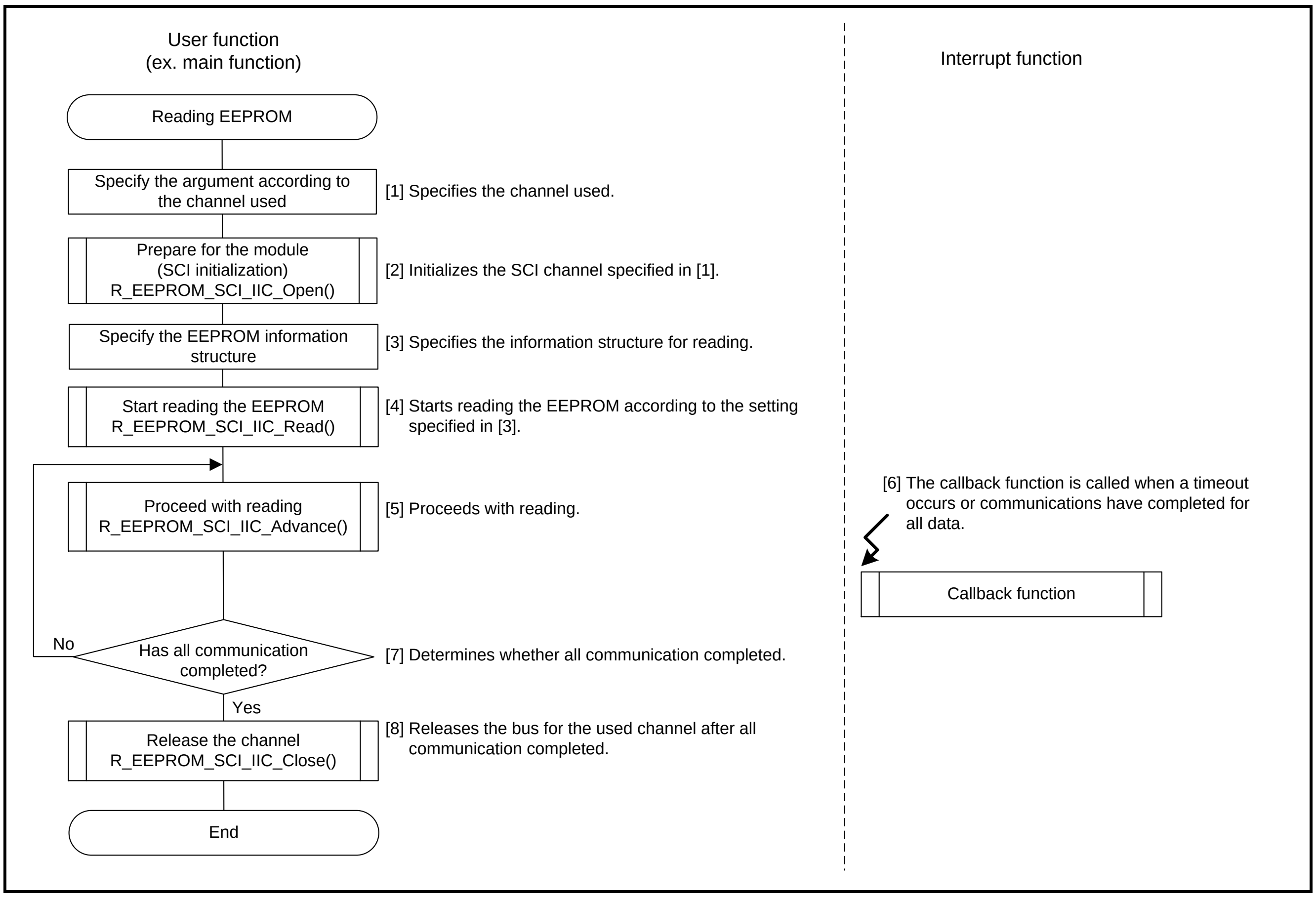
User function
(ex. main function)
Interrupt function
Reading EEPROM
Specify the argument according to the channel used
[1] Specifies the channel used.
Prepare for the module (SCI initialization) R_EEPROM_SCI_IIC_Open()
[2] Initializes the SCI channel specified in [1].
Specify the EEPROM information structure
[3] Specifies the information structure for reading.
Start reading the EEPROM R_EEPROM_SCI_IIC_Read()
[4] Starts reading the EEPROM according to the setting specified in [3].
Proceed with reading R_EEPROM_SCI_IIC_Advance()
[5] Proceeds with reading.
[6] The callback function is called when a timeout occurs or communications have completed for all data.
Callback function
No
Has all communication completed?
[7] Determines whether all communication completed.
Yes
Release the channel R_EEPROM_SCI_IIC_Close()
[8] Releases the bus for the used channel after all communication completed.
End

Figure 1.4 Reading an EEPROM

1.4.3 State Transition

Figure 1.5 shows the EEPROM Access (Simple I^2C) FIT Module State Transition Diagram.

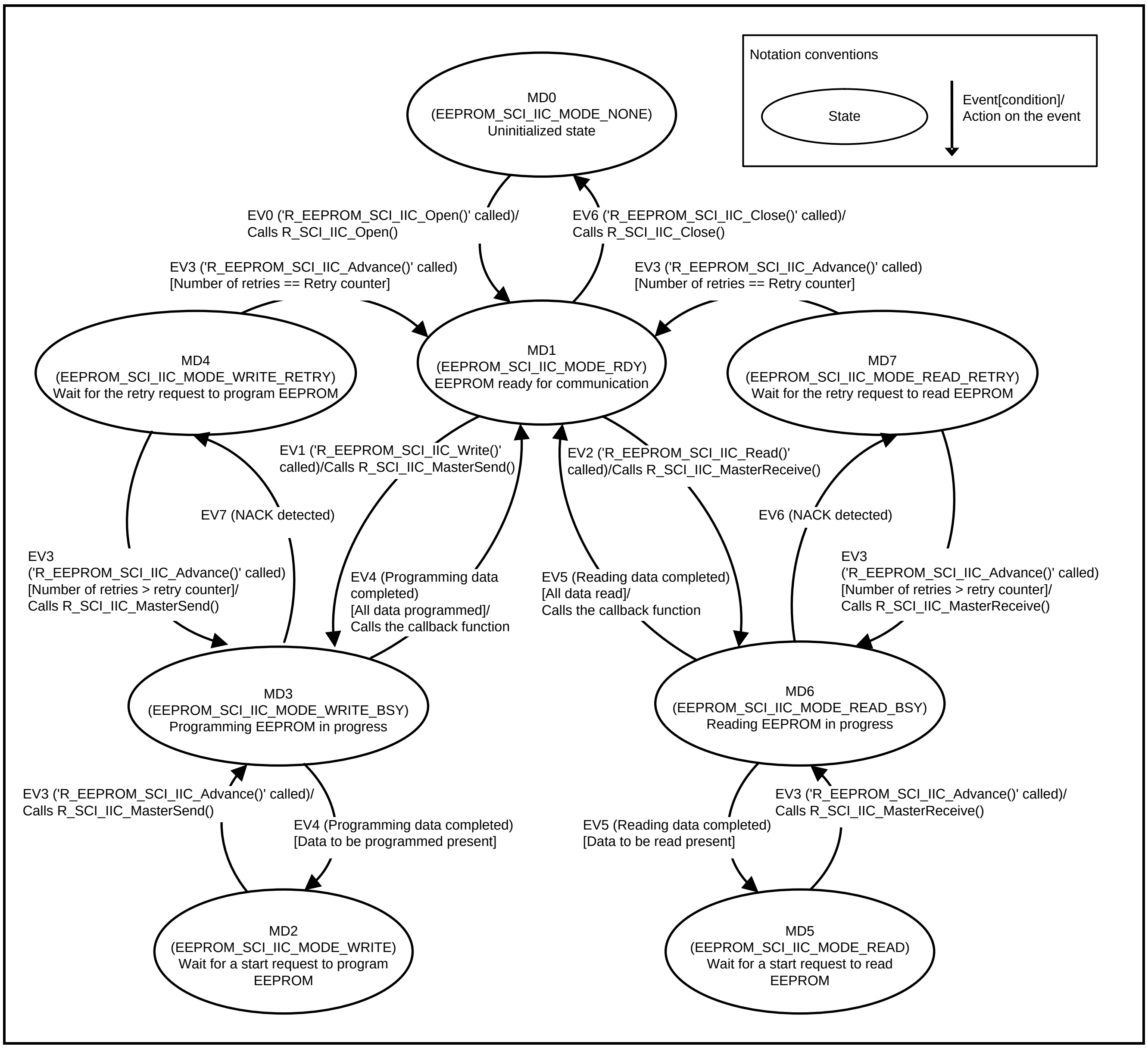
Notation conventions
State
Event[condition]/ Action on the event
MD0 (EEPROM_SCI_IIC_MODE_NONE) Uninitialized state
EV0 ('R_EEPROM_SCI_IIC_Open()' called)/ Calls R_SCI_IIC_Open()
EV6 ('R_EEPROM_SCI_IIC_Close()' called)/ Calls R_SCI_IIC_Close()
EV3 ('R_EEPROM_SCI_IIC_Advance()' called) [Number of retries == Retry counter]
EV3 ('R_EEPROM_SCI_IIC_Advance()' called) [Number of retries == Retry counter]
MD4 (EEPROM_SCI_IIC_MODE_WRITE_RETRY) Wait for the retry request to program EEPROM
MD1 (EEPROM_SCI_IIC_MODE_RDY) EEPROM ready for communication
MD7 (EEPROM_SCI_IIC_MODE_READ_RETRY) Wait for the retry request to read EEPROM
EV1 ('R_EEPROM_SCI_IIC_Write()' called)/Calls R_SCI_IIC_MasterSend()
EV2 ('R_EEPROM_SCI_IIC_Read()' called)/Calls R_SCI_IIC_MasterReceive()
EV7 (NACK detected)
EV6 (NACK detected)
EV3 ('R_EEPROM_SCI_IIC_Advance()' called) [Number of retries > retry counter]/ Calls R_SCI_IIC_MasterSend()
EV4 (Programming data completed) [All data programmed]/ Calls the callback function
EV5 (Reading data completed) [All data read]/ Calls the callback function
EV3 ('R_EEPROM_SCI_IIC_Advance()' called) [Number of retries > retry counter]/ Calls R_SCI_IIC_MasterReceive()
MD3 (EEPROM_SCI_IIC_MODE_WRITE_BSY) Programming EEPROM in progress
MD6 (EEPROM_SCI_IIC_MODE_READ_BSY) Reading EEPROM in progress
EV3 ('R_EEPROM_SCI_IIC_Advance()' called)/ Calls R_SCI_IIC_MasterSend()
EV4 (Programming data completed) [Data to be programmed present]
EV5 (Reading data completed) [Data to be read present]
EV3 ('R_EEPROM_SCI_IIC_Advance()' called)/ Calls R_SCI_IIC_MasterReceive()
MD2 (EEPROM_SCI_IIC_MODE_WRITE) Wait for a start request to program EEPROM
MD5 (EEPROM_SCI_IIC_MODE_READ) Wait for a start request to read EEPROM

Figure 1.5 EEPROM Access (Simple I^2C) FIT Module State Transition Diagram

2. API Information

This driver API adheres to the Renesas API naming standards.

2.1 Hardware Requirements

This driver requires your MCU support the following feature:

- SCI

2.2 Software Requirements

This driver is dependent upon the following packages:

- r_bsp
- r_sci_iic_rx

2.3 Supported Toolchains

This driver is tested and works with the following toolchain:

- Renesas RX Toolchain v.2.02.00

2.4 Header Files

All API calls and their supporting interface definitions are located in r_eeprom_sci_iic_rx_if.h.

2.5 Integer Types

This project uses ANSI C99. These types are defined in stdint.h.

2.6 Configuration Overview

The configuration options in this module are specified in r_eeprom_sci_iic_rx_config.h. The option names and setting values are listed in the table below.

Configuration options in *r_eeprom_sci_iic_rx_config.h*	
EEPROM_SCI_IIC_CFG_PARAM_CHECKING **- Default value = 1**	Selectable whether to include parameter checking in the code. - When this is set to 0, parameter checking is omitted. - When this is set to 1, parameter checking is included.
EEPROM_SCI_IIC_CFG_COUNT_RETRY **- Default value = 100**	Specifies the number of retries (reprogramming/rereading) when a NACK is detected. Specify a value less than or equal to 0xFFFFFFFF.
EEPROM_SCI_IIC_CFG_WAIT_CNT **- Default value = 1000**	Specifies the given time (number of loops) which is inserted before retry ('for loop' processing). Specify a value less than or equal to 0xFFFFFFFF.
EEPROM_SCI_IIC_CFG_CHi_TABLE_NO **i = 0 to 12** **- When i = 1, the default value is 0.** **- When i = 5, the default value is 1.** **- When i = 12, the default value is 2.** **- When i = value other than above, the default value is "DUMMY".**	Specifies whether to use SCIi. - When not using SCIi, set this to "DUMMY". - When using SCIi, specify from 0 to a value smaller than the value of MAX_EEPROM_SCI_IIC_CH_NUM. Do not set the same value for multiple channels.
EEPROM_SCI_IIC_CH_MAXUSE_NUM **- Default value = 3**	Specifies the number of SCI channels used. Specify from 1 to a value smaller than the value of MAX_EEPROM_SCI_IIC_CH_NUM.

2.7 Parameters

This section describes the structure whose members are API parameters. This structure is located in r_eeprom_sci_iic_rx_if.h as are the prototype declarations of API functions.

```
typedef volatile struct
{
   uint8_t rsv1; /* Reserved area */
   uint8_t ch_no; /* SCI channel number */
   uint8_t size_wr_blk_byte; /* Programming block size */
   uint8_t size_mem_adr_byte; /* Memory address size */
   eeprom_sci_iic_callback callbackfunc; /* Callback function */
   uint32_t cnt_all_wr_data; /* Total number of data to be programmed */
   uint8_t * p_wr_data; /* Pointer to the programming data */
   uint16_t mem_adr; /* Memory address */
   uint16_t dev_adr; /* Device address */
} eeprom_sci_iic_wr_info_t; // Structure used when programming an EEPROM

typedef volatile struct
{
   uint8_t rsv1; /* Reserved area */
   uint8_t ch_no; /* SCI channel number */
   uint8_t size_rd_blk_byte; /* Read block size */
   uint8_t size_mem_adr_byte; /* Memory address size */
   eeprom_sci_iic_callback callbackfunc; /* Callback function */
   uint32_t cnt_all_rd_data; /* Total number of data to be read */
   uint8_t * p_rd_data; /* Pointer to the read data */
   uint16_t mem_adr; /* Memory address */
   uint16_t dev_adr; /* Device address */
} eeprom_sci_iic_rd_info_t; // Structure used when reading an EEPROM
```

2.8 Return Values

This section describes return values of API functions. This enumeration is located in r_eeprom_sci_iic_rx_if.h as are the prototype declarations of API functions.

```
typedef enum
{
   EEPROM_SCI_IIC_SUCCESS = 0U, /* Function is called successfully */
   EEPROM_SCI_IIC_ERR_LOCK_FUNC, /* The SCI is used by another module */
   EEPROM_SCI_IIC_ERR_INVALID_CHAN, /* Nonexistent channel */
   EEPROM_SCI_IIC_ERR_INVALID_ARG, /* Invalid parameter */
   EEPROM_SCI_IIC_ERR_NO_INIT, /* Uninitialized state */
   EEPROM_SCI_IIC_ERR_BUS_BUSY, /* Bus is busy */
   EEPROM_SCI_IIC_ERR_OTHER, /* Other error */
} eeprom_sci_iic_return_t;
```

2.9 Callback Function

In this module, the callback function is called when programming or reading an EEPROM for all data (1) have been completed.

To specify the callback function, store the address of the function to be registered as the callback function in the structure member “callbackfunc”. For the structure member, refer to 2.7 Parameters.

When the callback function is called, the variable which stores the constants listed in Table 2.1 is passed as an argument. After the callback function is executed, the content of the variable is initialized to “EEPROM_SCI_IIC_NONE”. When the argument is required for other than the callback function, copy the value to somewhere such as global variable.

Refer to the Example in 3.2 R_EEPROM_SCI_IIC_Write() and 3.3 R_EEPROM_SCI_IIC_Read() for usage examples.

Note:

1. The number of data is specified in the structure member “cnt_all_wr_data” when programming and in “cnt_all_rd_data” when reading. For the structure members, refer to 2.7 Parameters.

Table 2.1 Arguments Used for the Callback Function (enum eeprom_sci_iic_status_t)

Constant Definition	Changed Timing to the Definition
EEPROM_SCI_IIC_NONE	After the callback function is executed. This is the initial value.
EEPROM_ SCI_IIC _FINISH	When programming/reading for all data is completed.
EEPROM_ SCI_IIC _TIMEOUT	When the number of retries on a NACK detection exceeds the setting value of the configuration option “EEPROM_SCI_IIC_CFG_COUNT_RETRY”.

2.10 Adding the FIT Module to Your Project

The module must be added to an existing e² studio project.

It is best to use the e2Studio FIT plug-in to add the FIT module to your project as that will automatically update the include file paths for you. To add the FIT module using the plug-in, refer to chapter 2 in the “Adding FIT Modules to Projects (r01an1723eu0100.pdf)” which is in the same level of the folder structure as this document.

Alternatively, the driver can be added manually. To add the FIT module manually, refer to chapter 3 in the “Adding FIT Modules to Projects (r01an1723eu0100.pdf)”.

When using the EEPROM Access (Simple I²C) FIT module, the BSP, CGC, and SCI simple I²C mode modules also need to be added. The BSP module is easily configured through the platform.h header file which is located in the r_bsp folder. For details of the CGC and SCI simple I²C mode modules, refer to application notes regarding these modules.

Configuring the BSP module

Open platform.h and uncomment the #include for the board you are using. For example, to use a RSK RX111 board, uncomment the #include for ‘./board/rskrx111/r_bsp.h’ macro and make sure all other board #includes are commented out.

3. API Functions

3.1 R_EEPROM_SCI_IIC_Open()

This function initializes the SCI to get this module ready for communication.

Format

```
eeprom_sci_iic_return_t R_EEPROM_SCI_IIC_Open (
        uint8_t ch /* Channel number */
)
```

Parameters

uint8_t ch

Specifies the channel number.

Return Values

EEPROM_SCI_IIC_SUCCESS, / The function is called successfully */*
EEPROM_SCI_IIC_ERR_LOCK_FUNC, / The SCI is used by another module */*
EEPROM_SCI_IIC_ERR_INVALID_CHAN, / Nonexistent channel */*
EEPROM_SCI_IIC_ERR_ARG, / Invalid parameter */*
EEPROM_SCI_IIC_ERR_BUS_BUSY, / Bus is busy */*
EEPROM_SCI_IIC_ERR_OTHER, / Other error */*

Properties

Prototyped in r_ eeprom_sci_iic_rx_if.h.

Description

Configures settings to start communication with an EEPROM. Initializes the SCI channel specified by the parameter. This function performs the following processes:

- Passes the channel number specified to the I^2C information structure in the SCI simple I^2C mode module.
- Updates the operating mode to “communication ready (EEPROM_SCI_IIC_MODE_RDY)”.
- Calls the R_SCI_IIC_Open function (initialization) of the SCI simple I^2C mode module.

Reentrant

None

Example

```
eeprom_sci_iic_wr_info_t     eep_w;

eep_w.ch_no = 1;

/* Open channel for writing EEPROM */
ret = R_EEPROM_SCI_IIC_Open(eep_w.ch_no);
```

Special Notes

None

3.2 R_EEPROM_SCI_IIC_Write()

This function starts programming an EEPROM.

Format

```
eeprom_sci_iic_return_t R_EEPROM_SCI_IIC_Write(
        eeprom_sci_iic_wr_info_t * p_eeprom_sci_iic_info  /* Structure data */
)
```

Parameters

*eeprom_sci_iic_wr_info_t * p_eeprom_sci_iic_info*

This is the pointer to the EEPROM information structure. Refer to 2.7 Parameters for details on the structure.

When setting the address of the EEPROM device, store it without shifting 1 bit to left.

```
uint8_t rsv1; /* Reserved area */
uint8_t ch_no; /* Channel number */
uint8_t size_wr_blk_byte; /* Programming block size (in bytes) */
uint8_t size_mem_adr_byte; /* Memory address size (in bytes) */
eeprom_sci_iic_callback callbackfunc; /* Callback function */
uint32_t cnt_all_wr_data; /* Total number of data to be programmed */
uint8_t * p_wr_data; /* Address of the programming data */
uint16_t mem_adr; /* Memory address */
uint16_t dev_adr; /* Address of the EEPROM device */
```

Return Values

EEPROM_SCI_IIC_SUCCESS, / Function is called successfully */*
EEPROM_SCI_IIC_ERR_INVALID_CHAN, / Nonexistent channel */*
EEPROM_SCI_IIC_ERR_INVALID_ARG, / Invalid parameter */*
EEPROM_SCI_IIC_ERR_NO_INIT, / Uninitialized state */*
EEPROM_SCI_IIC_ERR_BUS_BUSY, / Bus is busy */*
EEPROM_SCI_IIC_ERR_OTHER, / Other error */*

Properties

Prototyped in r_eeprom_sci_iic_rx_if.h.

Description

Starts programming the EEPROM with the SCI channel specified by the parameter. If the state of the channel is “communication ready (EPROM_SCI_IIC_MODE_RDY)”, the following processes are performed.

- Initializes global variables used by the API.
- Updates the operation mode.
- Passes the EEPROM information structure to the I^2C information structure of the SCI simple I^2C mode module.
- Calls the R_SCI_IIC_MasterSend function (master transmission) of the SCI simple I^2C mode module. The master transmission starts by the called API.

Reentrant

None

Example

```
void CallbackEEPROM(void *);
eeprom_sci_iic_status_t g_event; //source to terminate the EEPROM communication

void main(void)
{
    volatile eeprom_sci_iic_return_t   ret;
    eeprom_sci_iic_wr_info_t     eep_w;
    uint16_t        addr_eeprom = 0x0050;
    uint16_t        access_addr = 0x0000;
    uint8_t         wr_data[9]={0x81,0x82,0x83,0x84,0x85,0x86,0x87,0x88,0x89};

    /* Set Informations */
        eep_w.ch_no = 1;
        eep_w.callbackfunc = &CallbackEEPROM;
        eep_w.size_wr_blk_byte = 2;
        eep_w.size_mem_adr_byte = 1;
        eep_w.cnt_all_wr_data = 8;
        eep_w.p_wr_data = wr_data;
        eep_w.mem_adr = access_addr;
        eep_w.dev_adr = addr_eeprom;

        /* Open channel for writing EEPROM (refer to section 3.1) */
        ret = R_EEPROM_SCI_IIC_Open(eep_w.ch_no);

    /* Start writing to EEPROM */
        ret = R_EEPROM_SCI_IIC_Write(&eep_w);

        while(EEPROM_SCI_IIC_FINISH != g_event)
        {
    /* Proceeds with programming an EEPROM. (refer to section 3.4) */
            ret = R_EEPROM_SCI_IIC_Advance(eep_w.ch_no);
        }

    /* Close Channel (refer to section 3.5) */
        ret = R_EEPROM_SCI_IIC_Close(eep_w.ch_no);

        while(1)
        {
             /* Do Nothing */
        }

}

/* Callback function */
void CallbackEEPROM(void * p_eeprom_status)
{
       g_event = *(eeprom_sci_iic_status_t *)p_eeprom_status;
}
```

Special Notes

None

3.3 R_EEPROM_SCI_IIC_Read()

This function starts reading an EEPROM.

Format

eeprom_sci_iic_return_t R_EEPROM_SCI_IIC_Read(

eeprom_sci_iic_rd_info_t * p_eeprom_sci_iic_info /* Structure data */

)

Parameters

*eeprom_sci_iic_rd_info_t * p_eeprom_sci_iic_info*

This is the pointer to the EEPROM information structure. Refer to 2.7 Parameters for details on the structure.

When setting the address of the EEPROM device, store it without shifting 1 bit to left.

```
uint8_t rsv1; /* Reserved area */
uint8_t ch_no; /* Channel number */
uint8_t size_rd_blk_byte; /* Read block size (in bytes) */
uint8_t size_mem_adr_byte; /* Memory address size (in bytes) */
eeprom_sci_iic_callback callbackfunc; /* Callback function */
uint32_t cnt_all_rd_data; /* Total number of data to be read */
uint8_t * p_rd_data; /* Address of the read data */
uint16_t mem_adr; /* Memory address */
uint16_t dev_adr; /* Address of the EEPROM device */
```

Return Values

EEPROM_SCI_IIC_SUCCESS, / Function is called successfully */*
EEPROM_SCI_IIC_ERR_INVALID_CHAN, / Nonexistent channel */*
EEPROM_SCI_IIC_ERR_INVALID_ARG, / Invalid parameter */*
EEPROM_SCI_IIC_ERR_NO_INIT, / Uninitialized state */*
EEPROM_SCI_IIC_ERR_BUS_BUSY, / Bus is busy */*
EEPROM_SCI_IIC_ERR_OTHER, / Other error */*

Properties

Prototyped in r_eeprom_sci_iic_rx_if.h.

Description

Starts reading the EEPROM with the SCI channel specified by the parameter. If the state of the channel is "communication ready (EPROM_SCI_IIC_MODE_RDY)", the following processes are performed.

- Initializes global variables used by the API.
- Updates the operation mode.
- Passes the EEPROM information structure to the I^{2}C information structure of the SCI simple I^{2}C mode module.
- Calls the R_SCI_IIC_MasterReceive function (master reception) of the SCI simple I^{2}C mode module. The SCI interrupts are enabled and a start condition is generated by the called API.

Reentrant

None

Example

```
void CallbackEEPROM(void *);
eeprom_sci_iic_status_t g_event; //source to terminate the EEPROM communication

void main(void)
{
    volatile eeprom_sci_iic_return_t   ret;
    eeprom_sci_iic_rd_info_t     eep_r;
    uint16_t        addr_eeprom = 0x0050;
    uint16_t        access_addr = 0x0000;
    uint8_t         store_area[9]={0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF};

    /* Set Informations */
    eep_r.ch_no = 1;
    eep_r.callbackfunc = &CallbackEEPROM;
    eep_r.size_rd_blk_byte = 2;
    eep_r.size_mem_adr_byte = 1;
    eep_r.cnt_all_rd_data = 8;
    eep_r.p_rd_data = store_area;
    eep_r.mem_adr = access_addr;
    eep_r.dev_adr = addr_eeprom;

    /* Open channel for reading EEPROM (refer to section 3.1)*/
    ret = R_EEPROM_SCI_IIC_Open(eep_r.ch_no);

    /* Start reading from EEPROM */
    ret = R_EEPROM_SCI_IIC_Read(&eep_r);

    while(EEPROM_SCI_IIC_FINISH != g_event)
    {
    /* Proceeds with reading an EEPROM (refer to section 3.4) */
        ret = R_EEPROM_SCI_IIC_Advance(eep_r.ch_no);
    }

    /* Close Channel (refer to section 3.5) */
    ret = R_EEPROM_SCI_IIC_Close(eep_r.ch_no);

    while(1)
    {
       /* Do Nothing */
    }

}

/* Callback function */
void CallbackEEPROM(void * p_eeprom_status)
{
       g_event = *(eeprom_sci_iic_status_t *)p_eeprom_status;
}
```

Special Notes

None

3.4 R_EEPROM_SCI_IIC_Advance()

This function proceeds with programming or reading an EEPROM.

Format

```
eeprom_sci_iic_return_t R_EEPROM_SCI_IIC_Advance(
          uint8_t ch  /* Channel number */
)
```

Parameters

uint8_t ch

Specifies the channel number.

Return Values

EEPROM_SCI_IIC_SUCCESS, / Function is called successfully */*
EEPROM_SCI_IIC_ERR_INVALID_CHAN, / Nonexistent channel */*
EEPROM_SCI_IIC_ERR_INVALID_ARG, / Invalid parameter */*
EEPROM_SCI_IIC_ERR_NO_INIT, / Uninitialized state */*
EEPROM_SCI_IIC_ERR_BUS_BUSY, / Bus is busy */*
EEPROM_SCI_IIC_ERR_OTHER, / Other error */*

Properties

Prototyped in r_eeprom_sci_iic_rx_if.h.

Description

Proceeds with programming or reading after the R_EEPROM_SCI_IIC_Write function (start programming EEPROM) or the R_EEPROM_SCI_IIC_Read function (start reading EEPROM) is executed. Performs the following processes depending on the operating mode:

- When the operating mode is EEPROM_SCI_IIC_MODE_WRITE (Wait for the start request to program EEPROM)
    - Clears the retry counter.
    - Updates the operating mode to EEPROM_SCI_IIC_MODE_WRITE_BSY (programming EEPROM in progress).
    - Calls the R_SCI_IIC_MasterSend function (master transmission) of the SCI simple I^{2}C mode module. It proceeds with programming the EEPROM.
- When the operating mode is EEPROM_SCI_IIC_MODE_WRITE_RETRY (Wait for the retry request to program EEPROM)
    - Increments the retry counter.
    - When the retry counter value exceeds the specified maximum value, updates the operating mode to EEPROM_SCI_IIC_MODE_RDY (EEPROM ready for communication).
    - When the retry counter value is less than or equal to the specified maximum value, performs the following:
        - Updates the operating mode to EEPROM_SCI_IIC_MODE_WRITE_BSY (programming EEPROM in progress).
        - Calls the R_SCI_IIC_MasterSend function (master transmission) of the SCI simple I^{2}C mode module.
- When the operating mode is EEPROM_SCI_IIC_MODE_READ (Wait for the start request to read EEPROM)
    - Clears the retry counter.
    - Updates the operating mode to EEPROM_SCI_IIC_MODE_READ_BSY (reading EEPROM in progress).
    - Calls the R_SCI_IIC_MasterReceive function (master reception) of the SCI simple I^{2}C mode module. It proceeds with reading the EEPROM.

- When the operating mode is EEPROM_SCI_IIC_MODE_READ_RETRY (Wait for the retry request to read EEPROM)
    - Increments the retry counter.
    - When the retry counter value exceeds the specified maximum value, updates the operating mode to EEPROM_SCI_IIC_MODE_RDY (EEPROM ready for communication).
    - When the retry counter value is less than or equal to the specified maximum value, performs the following:
        - Updates the operating mode to EEPROM_SCI_IIC_MODE_READ_BSY (reading EEPROM in progress).
        - Calls the R_SCI_IIC_MasterReceive function (master reception) of the SCI simple I^{2}C mode module.

Reentrant
None

Example
Refer to the Example in the R_EEPROM_SCI_IIC_Write and R_EEPROM_SCI_IIC_Read functions for using this API.

Special Notes
None

3.5 R_EEPROM_SCI_IIC_Close ()

This function completes the communication with the EEPROM device and releases the SCI channel used.

Format

```
eeprom_sci_iic_return_t R_EEPROM_SCI_IIC_Close(
        uint8_t ch /* Channel number */
)
```

Parameters

uint8_t ch

Specifies the channel number.

Return Values

EEPROM_SCI_IIC_SUCCESS, / Function is called successfully */*
EEPROM_SCI_IIC_ERR_INVALID_CHAN, / Nonexistent channel */*
EEPROM_SCI_IIC_ERR_INVALID_ARG, / Invalid parameter */*
EEPROM_SCI_IIC_ERR_BUS_BUSY, / Bus is busy */*
EEPROM_SCI_IIC_ERR_OTHER, / Other error */*

Properties

Prototyped in r_eeprom_sci_iic_rx_if.h.

Description

Configures settings to complete the communication with the EEPROM device. Releases the SCI channel specified by the parameter. This function performs the following processes:

- Passes the channel number specified to the I^2C information structure in the SCI simple I^2C mode module.
- Updates the operating mode to "uninitialized state (EEPROM_SCI_IIC_MODE_NONE)".
- Calls the R_SCI_IIC_Close function of the SCI simple I^2C mode module. The I^2C output port is released and the SCI interrupts are disabled by the called API.

When starting a communication with the EEPROM again, the R_EEPROM_SCI_IIC_Open (initialization) function needs to be called. If a communication was forcibly terminated, the communication cannot be guaranteed.

Reentrant

None

Example

```
eeprom_sci_iic_rd_info_t     eep_r;
eep_r.ch_no = 1;

/* Close Channel */
ret = R_EEPROM_SCI_IIC_Close(eep_r.ch_no);
```

Special Notes

None

3.6 R_EEPROM_SCI_IIC_GetVersion()

This function returns the API version.

Format

uint32_t R_EEPROM_SCI_IIC_GetVersion(void)

Parameters

None

Return Values

Version number

Properties

Prototyped in r_eeprom_sci_iic_rx_if.h.

Description

Returns the API version number.

Reentrant

None

Example

```
uint32_t   version;

version = R_EEPROM_SCI_IIC_GetVersion();
```

Special Notes

This function is inlined using ‘#pragma inline’.

4. Provided Modules

The modules provided can be downloaded from the Renesas Electronics website.

5. Reference Documents

User’s Manual: Hardware
The latest version can be downloaded from the Renesas Electronics website.

Technical Update/Technical News
The latest information can be downloaded from the Renesas Electronics website.

User’s Manual: Development Tools
[e^2 studio] RX Family Compiler CC-RX V2.01.00 User's Manual: RX Coding (R20UT2748EJ)
The latest version can be downloaded from the Renesas Electronics website.

Website and Support

Renesas Electronics website
http://www.renesas.com

Inquiries
http://www.renesas.com/contact/

REVISION HISTORY	RX Family Application Note Simple I^2C Module for EEPROM Access Using Firmware Integration Technology

Rev.	**Date**	**Description**	
		Page	**Summary**
1.00	Sep. 30, 2013	—	First edition issued
1.10	Nov. 15, 2013	—	Changed the title of the document from “EEPROM SCI Simple I2C Mode FIT Module” to “Simple I^2C Module for EEPROM Access Using Firmware Integration Technology”.
		4	Table 1.2 Required Memory Size: Changed the ROM and RAM sizes to show the sizes only for this module.
		12	Added 2.9 Callback Function.
1.20	Oct. 1, 2014	1	Target Device: Added the RX110 Group.
		1	Related Documents: Added.
		4	Table 1.2 Required Memory Size: Changed the size information.
		10	2.2 Software Requirements: Deleted “r_cgc_rx” since this module is independent of the r_cgc_rx.
		10	2.3 Supported Toolchains: Changed the compiler version.
		10	2.6 Configuration Overview: Added the following configuration options: - EEPROM_SCI_IIC_CFG_PARAM_CHECKING - EEPROM_SCI_IIC_CFG_CHi_TABLE_NO - EEPROM_SCI_IIC_CH_MAXUSE_NUM
		22	5. Reference Documents: Changed the reference document for the User’s Manual: Development Tools.
1.30	Dec. 1, 2014	—	Added support for the RX113 Group.
1.31	Feb. 1, 2019	—	Changes associated with functions: Added support setting function of configuration option Using GUI on Smart Configurator. [Description] Added a setting file to support configuration option setting function by GUI.

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.

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics Corporation
TOYOSU FORESIA, 3-2-24 Toyosu, Koto-ku, Tokyo 135-0061, Japan

Renesas Electronics America Inc.
1001 Murphy Ranch Road, Milpitas, CA 95035, U.S.A.
Tel: +1-408-432-8888, Fax: +1-408-434-5351

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709 Quantum Plaza, No.27 ZhichunLu, Haidian District, Beijing, 100191 P. R. China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, 200333 P. R. China
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL 2nd Stage, Indiranagar, Bangalore 560 038, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
17F, KAMCO Yangjae Tower, 262, Gangnam-daero, Gangnam-gu, Seoul, 06265 Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5338

Colophon 7.2