Rx Family
Setting for Using Firmware Integration Technology with the Code Generator

Abstract
This application note describes procedures for setting firmware integration technology (FIT) and the code generator to be used together. The descriptions in this document assume that the product used is the RX64M Group. When using a product other than the RX64M Group, replace the description of the RX64M Group with the product used.

Products
RX Family

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

1. Overview ............................................................................................................................................. 3

2. Confirmed Operating Conditions ........................................................................................................ 3

3. Reference Application Notes .............................................................................................................. 3

4. Procedure When Using the BSP Module for Initial Settings and the Code Generator for the Peripheral Functions ........................................................................................................................... 4
   4.1 Procedure for Creating a New Project ......................................................................................... 4
   4.2 Procedure When Changing the Settings ................................................................................... 16
   4.3 Procedure When Adding Peripheral Functions .......................................................................... 18

5. Reference Documents ...................................................................................................................... 21
1. Overview

This document describes the setting procedures in three cases listed below when using the Board Support Package (BSP) FIT module for initial setting and clock settings, and using the code generator for setting the peripheral functions.

1. Procedure when starting a new project
   This is the procedure from generating a new project to the build. Refer to section 4.1 for details.

2. Procedure when changing the settings
   This procedure is used for setting the clocks and changing the conditions of peripheral functions for the project generated in (1) above. Refer to section 4.2 for details.

3. Procedure when adding peripheral functions
   This procedure is used for adding peripheral functions to the project generated in (1) above. Refer to section 4.3 for details.

Restriction

The BSP FIT module cannot be used in conjunction with the code generator of the realtime clock (RTC)

2. Confirmed Operating Conditions

The sample code accompanying this application note has been run and confirmed under the conditions below. Operation is not guaranteed if the e² studio or BSP module version is changed.

<table>
<thead>
<tr>
<th>Table 2.1 Confirmed Operating Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Item</strong></td>
</tr>
<tr>
<td>MCU used</td>
</tr>
<tr>
<td>Integrated development environment</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>C compiler</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>BSP module version</td>
</tr>
<tr>
<td>iodefine.h version</td>
</tr>
<tr>
<td>Endian</td>
</tr>
<tr>
<td>Operating mode</td>
</tr>
<tr>
<td>Processor mode</td>
</tr>
<tr>
<td>Board used</td>
</tr>
</tbody>
</table>

3. Reference Application Notes

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

- Firmware Integration Technology User’s Manual (R01AN1833EU)
- RX Family Board Support Package Module Using Firmware Integration Technology (R01AN1685EU)
4. Procedure When Using the BSP Module for Initial Settings and the Code Generator for the Peripheral Functions

Sections 4.1 to 4.3 describe setting procedures when using the Board Support Package (BSP) FIT module for initial setting and the code generator for the peripheral functions. The setting procedures in this section are described for the RX64M Group as an example. When using a product other than the RX64M Group, replace the description of the RX64M Group with the product used.

4.1 Procedure for Creating a New Project

This section describes the procedure from generating a new project, through setting the peripheral functions with the code generator, to the build. Figure 4.1 shows the flow for generating a new project. This section explains processing method described in the figure.

![Figure 4.1 Procedure When Creating a New Project](image-url)
1. Creating a new project.

Create a new project using project generation for the FIT

(1) From the e² studio toolbar, select File > New > C Project.

(2) Input a project name, select Renesas RXC toolchain as the toolchain, and click Next.

(3) Select a target.

(4) Uncheck the Release (no debug) in the configuration options and click Next.

(5) Check the Use Peripheral code Generator box, and click Next.

---

**Note:** When creating a project only for use with FIT, the box does not need to be checked, but in this case the code generator is used for the peripheral functions, so the box needs to be checked.

(6) Change the Select Additional CPU Options and Global Options Settings as needed.

(7) In the Standard Header Files, select ‘C(C99)’ for the library structure, and click Next.

(8) Uncheck the Use User Stack, Use Heap Memory, Vector Definition Files, and I/O Register Definition Files, and click Finish.

(9) When a project is created, delete the dbsct.c, project name.c, and typedefine.h files from the src folder.

(10) Right click on the project, and select C/C++ Build > Settings > Linker > Section from Properties.

(11) Click on Remove Section in the PResetPRG section and PIntPRG section.

(12) Change P section to "P* section".

(13) Click Apply.
2. Adding the BSP module.

The BSP module is added to the project created in step 1.

(1) From the e2 studio toolbar, select File > New > Renesas FIT Module.

(2) Select the project name to which the FIT module will be added.

(3) Select the Family, Series, Group, and Target Board.

(4) Select the r_bsp module.

(5) Click Finish.

(6) Copy files r_bsp_config_reference.h and r_bsp_interruput_config_reference.h from the r_bsp/board/rskrx64m folder to the r_config folder and rename the files to "r_bsp_config.h" and "r_bsp_interruput_config.h", respectively.
Configure the clock in `r_bsp_config.h` (select the clock source, set the frequency of the main clock, select the division ratios, etc.).

```c
#include <stdio.h>

int main() {
    // Code for configuring the clock
    return 0;
}
```

/* Clock source select (CKSEL)*/
0 = Low Speed On-Chip Oscillator (LOCO)
1 = High Speed On-Chip Oscillator (HOCO)
2 = Main Clock Oscillator
3 = Sub-Clock Oscillator
4 = PLL Circuit
*/
#define BSP_CFG_CLOCK_SOURCE (4)

/* Clock configuration options.*/
The input clock frequency is specified and then the system clocks are set by specifying the multipliers used. The multiplier settings are used to set the clock registers in resetprg.c. If a 24MHz clock is used and the ICLK is 120MHz, PCLKA is 120MHz, PCLKB is 60MHz, PCLKC is 60MHz, PCLKD is 60MHz, FCLK is 60MHz, USB Clock is 48MHz, and BCLK is 120MHz then the settings would be:

```c
#define BSP_CFG_XTAL_HZ (24000000)
#define BSP_CFG_PLL_DIV (1) (no division)
#define BSP_CFG_PLL_MUL (10.0 (24MHz x 10.0 = 240MHz)
#define BSP_CFG_ICLK_DIV (2): System clock (ICLK) =
    ((BSP_CFG_XTAL_HZ/BSP_CFG_PLL_DIV) * BSP_CFG_PLL_MUL / (BSP_CFG_ICLK_DIV) = 120MHz
#define BSP_CFG_PCKA_DIV (2): Peripheral Clock A (PCLKA) =
    ((BSP_CFG_XTAL_HZ/BSP_CFG_PLL_DIV) * BSP_CFG_PLL_MUL / (BSP_CFG_PCKA_DIV) = 120MHz
#define BSP_CFG_PCKB_DIV (4): Peripheral Clock B (PCLKB) =
    ((BSP_CFG_XTAL_HZ/BSP_CFG_PLL_DIV) * BSP_CFG_PLL_MUL / (BSP_CFG_PCKB_DIV) = 48MHz
#define BSP_CFG_PCKC_DIV (4): Peripheral Clock C (PCLKC) =
    ((BSP_CFG_XTAL_HZ/BSP_CFG_PLL_DIV) * BSP_CFG_PLL_MUL / (BSP_CFG_PCKC_DIV) = 60MHz
#define BSP_CFG_PCKD_DIV (4): Peripheral Clock D (PCLKD) =
    ((BSP_CFG_XTAL_HZ/BSP_CFG_PLL_DIV) * BSP_CFG_PLL_MUL / (BSP_CFG_PCKD_DIV) = 60MHz
#define BSP_CFG_FCK_DIV (4): Flash IF Clock (FCLK) =
    ((BSP_CFG_XTAL_HZ/BSP_CFG_PLL_DIV) * BSP_CFG_PLL_MUL / (BSP_CFG_FCK_DIV) = 60MHz
#define BSP_CFG_BCK_DIV (2): External Bus Clock (BCLK) =
    ((BSP_CFG_XTAL_HZ/BSP_CFG_PLL_DIV) * BSP_CFG_PLL_MUL / (BSP_CFG_BCK_DIV) = 120MHz
#define BSP_CFG_UCK_DIV (5): USB Clock (UCLK) =
    ((BSP_CFG_XTAL_HZ/BSP_CFG_PLL_DIV) * BSP_CFG_PLL_MUL / (BSP_CFG_UCK_DIV) = 48MHz
```

/* XTAL - Input clock frequency in Hz */
#define BSP_CFG_XTAL_HZ (24000000)

/* The HOCO can operate at several different frequencies. Choose which one using the macro below. Available frequency settings:*/
0 = 16MHz (default)
1 = 18MHz
2 = 20MHz
*/
#define BSP_CFG_HOCO_FREQUENCY (0)

/* PLL clock source (PLLSRCSEL). Choose which clock source to input to the PLL circuit.*/
Available clock sources:
0 = Main clock (default)
1 = HOCO
*/
#define BSP_CFG_PLL_SRC (0)

/* PLL Input Frequency Division Ratio Select (PLDIV).*/
Available divisors = /1 (no division), /2, /3
*/
#define BSP_CFG_PLL_DIV (1)

/* PLL Frequency Multiplication Factor Select (STC).*/
Available multipliers = x10.0 to x30.0 in 0.5 increments
(e.g. 10.0, 10.5, 11.0, 11.5, ..., 29.0, 29.5, 30.0)
*/
#define BSP_CFG_PLL_MUL (10.0)
(8) Select the platform. To select the platform, uncomment the \#include for the board used.

Before

```c
/* RDKRX63N */
//#include "./board/rdkrx63n/r_bsp.h"

/* RSKRX64M */
//#include "./board/rskrx64m/r_bsp.h"

/* RSKRX210 */
//#include "./board/rskrx210/r_bsp.h"
```

After

```c
/* RDKRX63N */
//#include "./board/rdkrx63n/r_bsp.h"

/* RSKRX64M */
#include "./board/rskrx64m/r_bsp.h"

/* RSKRX210 */
#include "./board/rskrx210/r_bsp.h"
```

Uncommented
3. Changing the code generator properties.

Change the Generate File Mode in the Code Generator Property to Merge file.

(1) In the project created, right click on the Code Generator and select Property.

(2) In Properties view, change File generation control to Merge file.

The three options in the File Generation Control are explained in the following table.

<table>
<thead>
<tr>
<th>File Generation Control</th>
<th>Processing After Code is Generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do nothing if file exists</td>
<td>If a file with the same name exists, a new file will not be output.</td>
</tr>
<tr>
<td>Merge file</td>
<td>If code is written in between the specified comments, that part is left as is and the code is updated.</td>
</tr>
<tr>
<td>Overwrite file</td>
<td>If a file with the same name exists, the existing file is overwritten by a new file.</td>
</tr>
</tbody>
</table>
4. Generating code for the peripheral functions

When code is generated, clocks and peripheral functions are set.

(1) Click on the Code Generator in the project > Peripheral Functions > Clock Generator.

(2) Configure the settings in those set to r_bsp_config.h in (7) of step 2.

**NOTE:** The peripheral function settings (counter value, bit rate, etc.) are calculated using the code generation settings. For proper operation under the BSP module settings, the clock generator setting of the code generator must match the setting of the BSP module.

(3) In the project, click on Code Generator > Peripheral Functions > peripheral function used.

(4) Use the graphic user interface (GUI) to select the desired conditions.

(5) Click Generate Code.

(6) Confirm that the source code has been generated in the project’s src folder.
5. Modifying the file that includes the main function.

The file that was created by the code generator and includes the main function is modified. The default file name is r_cg_main.c (hereinafter referred to as the main file). When adding code, refer to the following note.

Note when adding code:

Code MUST be added between the comments below. If code is added someplace other than between the comments below, then the code is deleted when code generation is performed again.

/* Start user code for include. Do not edit comment generated here */

When adding code, add the code in between these two comments.

/* End user code. Do not edit comment generated here */

(1) Delete #includes for the r_cg_cgc.h.

(2) Add #include for platform.h.

Before change

```
/******************************
 Includes
 ******************************
#include "r_cg_macrodriver.h"
#include "r_cg_cgc.h"   Edit
#include "r_cg_mtu3.h"   Delete
/* Start user code for include. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
#include "r_cg_userdefined.h"
```

After change

```
/******************************
 Includes
 ******************************
#include "r_cg_mtu3.h"
/* Start user code for include. Do not edit comment generated here */
#include "platform.h" Add
/* End user code. Do not edit comment generated here */
#include "r_cg_userdefined.h"
```
(3) Adding macro definitions

Copy the code and comments in the red box from r_cg_macrodriver.h and paste them in the main file.

(Extract part of the code for r_cg_macrodriver)

```c
.getOrElse(apply)''
```
(4) Adding the initialization function to peripheral functions

(4-1) Copy the code for the R_Systeminit function in r_cg_hardware_setup.c and paste it in the R_MAIN_UserInit function.

```c
void R_Systeminit(void) {
    /* Enable writing to registers related to operating modes, LPC, CGC and software reset */
    SYSTEM.PRCR.WORD = 0xA50BU;

    /* Enable writing to MPC pin function control registers */
    MPC.PWPR.BIT.B0WI = 0U;
    MPC.PWPR.BIT.PFSWE = 1U;

    /* Initialize non-existent pins */
    PORT5.PDR.BYTE = 0x70U;

    /* Set peripheral settings */
    R_CGC_Create();
    R_MTU3_Create();

    /* Disable writing to MPC pin function control registers */
    MPC.PWPR.BIT.PFSWE = 0U;
    MPC.PWPR.BIT.B0WI = 1U;

    /* Disable protection */
    SYSTEM.PRCR.WORD = 0xA500U;
}
```

(4-2) Delete the unused pin processing, and delete the call for the R_CGC_Create function.

```c
void R_Systeminit(void) {
    /* Enable writing to registers related to operating modes, LPC, CGC and software reset */
    SYSTEM.PRCR.WORD = 0xA50BU;

    /* Enable writing to MPC pin function control registers */
    MPC.PWPR.BIT.B0WI = 0U;
    MPC.PWPR.BIT.PFSWE = 1U;

    /* Initialize non-existent pins */
    PORT5.PDR.BYTE = 0x70U;

    /* Set peripheral settings */
    R_CGC_Create();
    R_MTU3_Create();

    /* Disable writing to MPC pin function control registers */
    MPC.PWPR.BIT.PFSWE = 0U;
    MPC.PWPR.BIT.B0WI = 1U;

    /* Disable protection */
    SYSTEM.PRCR.WORD = 0xA500U;

    /* End user code. Do not edit comment generated here */
}
```

Make sure to paste the code in between these two comments.
(4-3) Add other processing
Add other processing that is executed before the infinite loop to (4-2).
(e.g. start function call for timers, turning on LEDs, etc.)

6. Modifying files other than the main file
Modify files that were created at the code generation except the main file.
(1) Modify \_\_\_rg\_macrodriver.h
Change the include path of iodefine.h (./../r_bsp/mcu/rx64m/register_access)

Before change

```c
/**************************/
Includes
**************************
#include "iodefine.h"
#include <machine.h>
```

After change

```c
/****************************/
Includes
*******************************
#include "../iodefine.h"
#include <machine.h>
```

7. Adding processing for interrupt functions
Interrupt handling for peripheral functions is performed in \_\_\_rg\_peripheral function name\_user.c.
Interrupt functions are generated by the code generator. Add processing for the interrupt functions.

8. Deleting unnecessary files
The initial setting and clock setting can be set both by the BSP module and code generator. Since the settings are conflicting, delete the following files in the e\^3 studio.
\_\_\_rg\_cgc\_user.c, \_\_\_rg\_cgc\_user.h, \_\_\_rg\_cgc\_c, \_\_\_rg\_cgc\_h, \_\_\_rg\_dbsct\_c, \_\_\_rg\_dbsct\_h, \_\_\_rg\_hardware\_setup\_c, \_\_\_rg\_intrprg\_c, \_\_\_rg\_intrprg\_h, \_\_\_rg\_resetprg\_c, \_\_\_rg\_resetprg\_h, \_\_\_rg\_sbrc\_c, \_\_\_rg\_sbrc\_h, \_\_\_rg\_stacksct\_h, \_\_\_rg\_vect\_h, \_\_\_rg\_vecttbl\_c
Note on warnings displayed after the build

If `#includes` for stdint.h and r_cg_macrodriver.h are added to the same file, the warnings below appear in Problems view after the build. This is because the same typedef is declared in both stdint.h generated by selecting C99 at project generation and r_cg_macrodriver.h generated by the code generator.

Follow the procedure below to delete these warnings. However, after regenerating the code in sections 4.2 Procedure When Changing the Settings and 4.3 Procedure When Adding Peripheral Functions, the procedure below must be performed again. Therefore, this procedure should be performed after all code generation is complete.

1. Delete the following code from the r_cg_macrodriver.h file.

```c
#ifndef TYPEDEF
typedef signed char int8_t;
typedef unsigned char uint8_t;
typedef signed short int16_t;
typedef unsigned short uint16_t;
typedef signed long int32_t;
typedef unsigned long uint32_t;
typedef unsigned short MD_STATUS;
#define __TYPEDEF__
#endif
```

2. Add `#include` for stdint.h to the r_cg_macrodriver.h file.

```c
#include <stdint.h>
```

4.2 Procedure When Changing the Settings

This section describes the procedures for setting clocks and changing the conditions of the peripheral functions after a new project has been created (see section 4.1). Figure 4.2 shows the Procedure When Changing the Settings. This section also describes detailed processing for each step of the procedure.

![Procedure Diagram](image)

**Start**
1. Modify r_bsp_config.h (only when changing the clock settings).
2. Modify the code generator.
3. Modify files that include the main function.
4. Modify files other than the main file.
5. Add processing for interrupt functions (only when changing the peripheral functions).
6. Delete unnecessary files. **(build project)**

**Figure 4.2 Procedure When Changing the Settings**

1. **Modifying r_bsp_config.h**

   When changing the clock setting (PLL clock division and multiplication, PCLKB division, etc.), change r_bsp_config.h.

   ```
   /* Peripheral Module Clock D Divider (PCKD).*/
   // Available divisors = /1 (no division), /2, /4, /8, /16, /32, /64
   #define BSP_CFG_PLL_SRC (4)
   
   /* External Bus Clock Divider (BCLK).*/
   // Available divisors = /1 (no division), /2, /4, /8, /16, /32, /64
   #define BSP_CFG_BCK_DIV (4)  // Example: Change the BCLK division from divided by 2 to divided by 4.
   
   /* Flash IF Clock Divider (FCK).*/
   // Available divisors = /1 (no division), /2, /4, /8, /16, /32, /64
   #define BSP_CFG_FCK_DIV (4)
   ```
2. Modifying the Code Generator

(1) When setting the clocks, set the Clock Generator to the same settings as those in r_bsp_config.h.

<table>
<thead>
<tr>
<th>System clock setting</th>
<th>PLL circuit</th>
<th>FLL circuit</th>
</tr>
</thead>
<tbody>
<tr>
<td>System clock ([I/CLK])</td>
<td>x 1/2</td>
<td>120</td>
</tr>
<tr>
<td>Peripheral module clock (PCLKA)</td>
<td>x 1/2</td>
<td>120</td>
</tr>
<tr>
<td>Peripheral module clock (PCLKB)</td>
<td>x 1/4</td>
<td>60</td>
</tr>
<tr>
<td>Peripheral module clock for ADC (PCLKC)</td>
<td>x 1/4</td>
<td>60</td>
</tr>
<tr>
<td>Peripheral module clock for ADC (PCLKD)</td>
<td>x 1/4</td>
<td>60</td>
</tr>
<tr>
<td>External bus clock (BCLK)</td>
<td>x 1/4</td>
<td>80</td>
</tr>
<tr>
<td>Flash IF clock (FCLK)</td>
<td>x 1/4</td>
<td>80</td>
</tr>
<tr>
<td>USB clock (UCLK)</td>
<td>x 1/5</td>
<td>48</td>
</tr>
</tbody>
</table>

Example: Change the BCLK division from divided by 2 to divided by 4

(2) Change the peripheral functions as needed.

(3) Click Generate Code.

3. Modifying files that include the main function

When code generation is performed again, codes that are not placed between the comments described in “Note when adding code” in Step 5 of 4-1 Procedure When Creating a New Project are regenerated, so the modification below must be made. This modification is also performed when creating a new project, but they are regenerated when code generation is performed again.

(1) Delete #includes for r_cg_cgc.h.

4. Modifying files other than the main file

When code generation is performed again, the modification below must be made. These are also performed when creating a new project, but they are regenerated when code generation is performed again.

(1) Modify r_cg_macrodriver.h

Change the include path for iodefine.h (../../../r_bsp/mcu/rx64m/register_access)

5. Changing processing for interrupt functions

Change processing for interrupt functions when necessary.

6. Deleting unnecessary files

Perform the same processing as step 8 in section 4.1 Procedure for Creating a New Project.
4.3 Procedure When Adding Peripheral Functions

This section describes the procedure for adding additional peripheral functions after a new project has been created (section 4.1 above). Figure 4.3 shows the Procedure When Adding Peripheral Functions. This section also describes detailed processing for each step of the procedure.

Figure 4.3 Procedure When Adding Peripheral Functions

1. Generating code for peripheral functions to be added

   (1) Configure settings for the peripheral functions to be added by the code generator.
   
   (2) Click Generate code.

2. Modify files that contain the main function.
3. Modify files other than the main file.
4. Change the processing of the interrupt functions for the added peripheral functions (only when necessary).
5. Delete unnecessary files.

End (build project)
2. Modifying Files That Include the main Function

Add code for the added peripheral function.

(1) Copy only the initial setting for the added peripheral function in the R_Systeminit function in r_cg_hardware_setup.c. (the initial setting function name for the peripheral functions is r_cg_<peripheral function>_Creat function).

```c
/*********************************************************************************
 * Function Name: R_Systeminit
 * Description : This function initializes every macro.
 * Arguments : None
 * Return Value : None
*******************************************************************************/
void R_Systeminit(void)
{
    /* Enable writing to registers related to operating modes, LPC, CGC, and software reset */
    SYSTEM.PRCR.WORD = 0xA50BU;
    /* Enable writing to MPC pin function control registers */
    MPC.PWPR.BIT.B0WI = 0U;
    MPC.PWPR.BIT.PFSWE = 1U;
    /* Initialize non-existent pins */
    PORT5.PDR.BYTE = 0x70U;
    /* Set peripheral settings */
    R_CGC_Create();
    R_LVDI_Create();
    /* Disable writing to MPC pin function control registers */
    MPC.PWPR.BIT.PFSWE = 0U;
    MPC.PWPR.BIT.B0WI = 1U;
    /* Disable protection */
    SYSTEM.PRCR.WORD = 0xA500U;
}
```

(2) Add the function copied in (2-1) to the R_MAIN_UserInit function.

```c
/*********************************************************************************
 * Function Name: R_MAIN_UserInit
 * Description : This function adds user code before implementing main function.
 * Arguments : None
 * Return Value : None
*******************************************************************************
void R_MAIN_UserInit(void)
{
    /* Start user code. Do not edit comment generated here */
    /* Enable writing to registers related to operating modes, LPC, CGC, and software reset */
    SYSTEM.PRCR.WORD = 0xA50BU;
    /* Enable writing to MPC pin function control registers */
    MPC.PWPR.BIT.B0WI = 0U;
    MPC.PWPR.BIT.PFSWE = 1U;
    /* Set peripheral settings */
    R_CGC_Create();
    R_LVDI_Create();
    R_MTU3_Create();
    R_MTU3_C0_Start();
    LED0 = LED_ON;
    /* Disable writing to MPC pin function control registers */
    MPC.PWPR.BIT.PFSWE = 0U;
    MPC.PWPR.BIT.B0WI = 1U;
    /* Disable protection */
    SYSTEM.PRCR.WORD = 0xA500U;
    /* End user code. Do not edit comment generated here */
}
```
The following processing needs to be performed every code generation.

(3) Delete `#includes` for `r_cg_cgc.h`.

3. Modifying Files Other Than the main File

When code generation is performed again, the modification below must be made. These are also performed when creating a new project, but they are regenerated when code generation is performed again.

(1) Modify `r_cg_macrodriver.h`
Change the include path for `iodefine.h` (`../../r_bsp/mcu/rx64m/register_access`)

4. Changing the Processing of the Interrupt Functions for the Added Peripheral Function

Add processing when the added peripheral function uses the interrupt functions.

Interrupt handling for peripheral functions is performed in "r_cg_name of peripheral function_user.c".

5. Deleting Unnecessary Files

Perform the same processing as step 8 in section 4.1 Procedure for Creating a New Project.
5. Reference Documents

User’s Manual: Hardware
RX64M Group User’s Manual: Hardware Rev.1.00 (R01UH0377EJ)
When using a product other than the RX64M Group, refer to User’s Manual: Hardware for the product used.
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 Environment
RX Family Compiler CC-RX V2.01.00 User’s Manual: RX Coding Rev.1.00 (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

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Page</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Dec. 26, 2014</td>
<td>—</td>
<td>First edition issued</td>
</tr>
<tr>
<td>1.10</td>
<td>Mar. 2, 2015</td>
<td>—</td>
<td>Changed the target device from the RX64M Group to the RX Family</td>
</tr>
</tbody>
</table>

All trademarks and registered trademarks are the property of their respective owners.
General Precautions in the Handling of MPU/MCU Products
The following usage notes are applicable to all MPU/MCU 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. 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 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. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on
   The state of the product is undefined at the moment 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 moment 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 moment 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 moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   - The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable.
   - When switching the clock signal during program execution, wait until the target clock signal has 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. Moreover, 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.

5. Differences between Products
   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.
   - The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the 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 of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.

2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.

3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties or arising from the use of Renesas Electronics products or technical information described in this document. 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 should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.

5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended 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; and industrial robots etc.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.

Renesas Electronics products are neither intended nor 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 damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.

6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.

7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, 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, please evaluate 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. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.

9. Renesas Electronics products and technology may not be used or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.

10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.

11. This document may not be 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, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics products” means any product developed or manufactured by or for Renesas Electronics.

Renesas Electronics Corporation

http://www.renesas.com

© 2015 Renesas Electronics Corporation. All rights reserved.

Colophon 5.0