RX Family

Using the Trusted Memory Function

Introduction

By storing, in part of the code flash memory, software that requires a high level of confidentiality such as cryptographic algorithm processing software, device control processing software involving know-how, paid middleware, and the like, the trusted memory (TM) function prevents unauthorized access and software alteration by third parties.
This application note explains how to store software in the area protected by TM and use software inside said area.

Target Devices

- RX64M Group
- RX71M Group
- RX65N Group
- RX651 Group
- RX66T Group
- RX72T Group
- RX72M Group
- RX72N Group
- RX66N Group

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

Contents

1. Specifications

The description in this application note assumes a first user, who enables the TM function, and a second user who uses the device with the TM function enabled. The first user writes a program to the area protected by TM, and then the second user executes writes to areas other than the area protected by TM.

Renesas Flash Programmer (RFP), a tool for programming flash memory, is used to write program data and to enable or disable the TM function.

Table 1.1 lists the peripheral functions used and their applications, and Figure 1.1 is an overview diagram.

Table 1.1 Peripheral Functions Used and Their Applications

Peripheral Function	Application
Trusted memory function	Prevents unauthorized reading of programs stored in the code flash memory or programming of additional data to the area protected by TM.

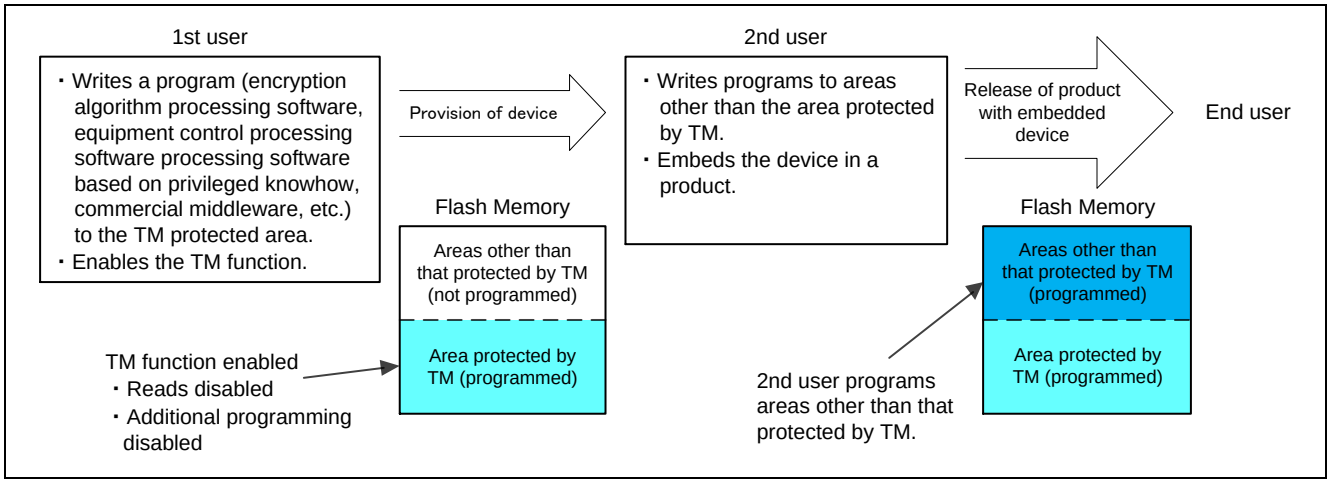


Figure 1.1 Overview Diagram

2. TM Function Operation Example

The first user (who supplies the program in the area protected by TM) develops an application and creates a library file (.lib) containing the variables and function tables used by the program in the area protected by TM. After developing the application, the first user uses RFP to write the program to the area protected by TM and enables the TM function.

The first user should provide the second user with the specifications of the program in the area protected by TM, sample code section information, and the library file (.lib) created as described above.

The second user (application developer) uses the provided program specifications, section information, and library file (.lib) to develop an application. After developing the application, the second user uses e2 studio to debug the application and uses RFP to write the program to an area other than that protected by TM.

The second user then embeds the device to which the program was written in the finished product and supplies it to the end user.

Figure 2.1 shows the development sequence.

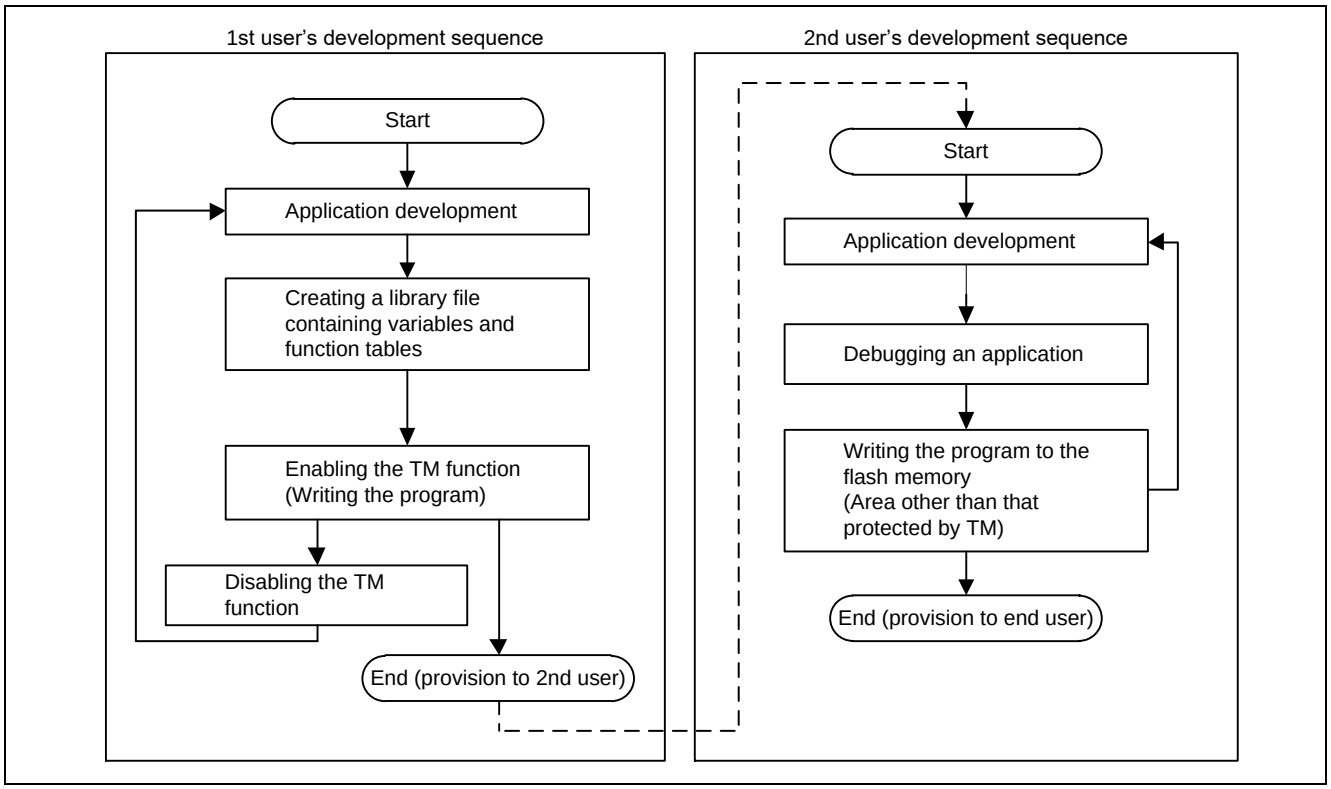


Figure 2.1 Development Sequence

2.1 Creating a New Project in e² studio

Both the first user and the second user use e2 studio to make section settings, specify ROM options, and initialize sections.

2.1.1 Section Settings

In order to clarify the areas of memory used for development by the first user and second user, separate sections should be specified for use by the first user's program in the area protected by TM and for application development by the second user.

Note that sections L and W are not used because they are #pragma sections that cannot be renamed. If switch statements are used, it is necessary to not specify table format (allocation to switch branching table area) but rather if_then format (allocation to program area).

This setting causes switch statements to be allocated to the program area and written to the device.

The switch statement code expansion setting is made by the first user only. The first user should provide the second user with information on the sections used.

The second user should then make settings for the sections used by the first user, based on the information provided. It is important that the addresses be accurate when making section settings. It is also important that the sections used by the second user be specified such that they do not overlap the areas used by the first user.

Figure 2.2 is a section diagram.

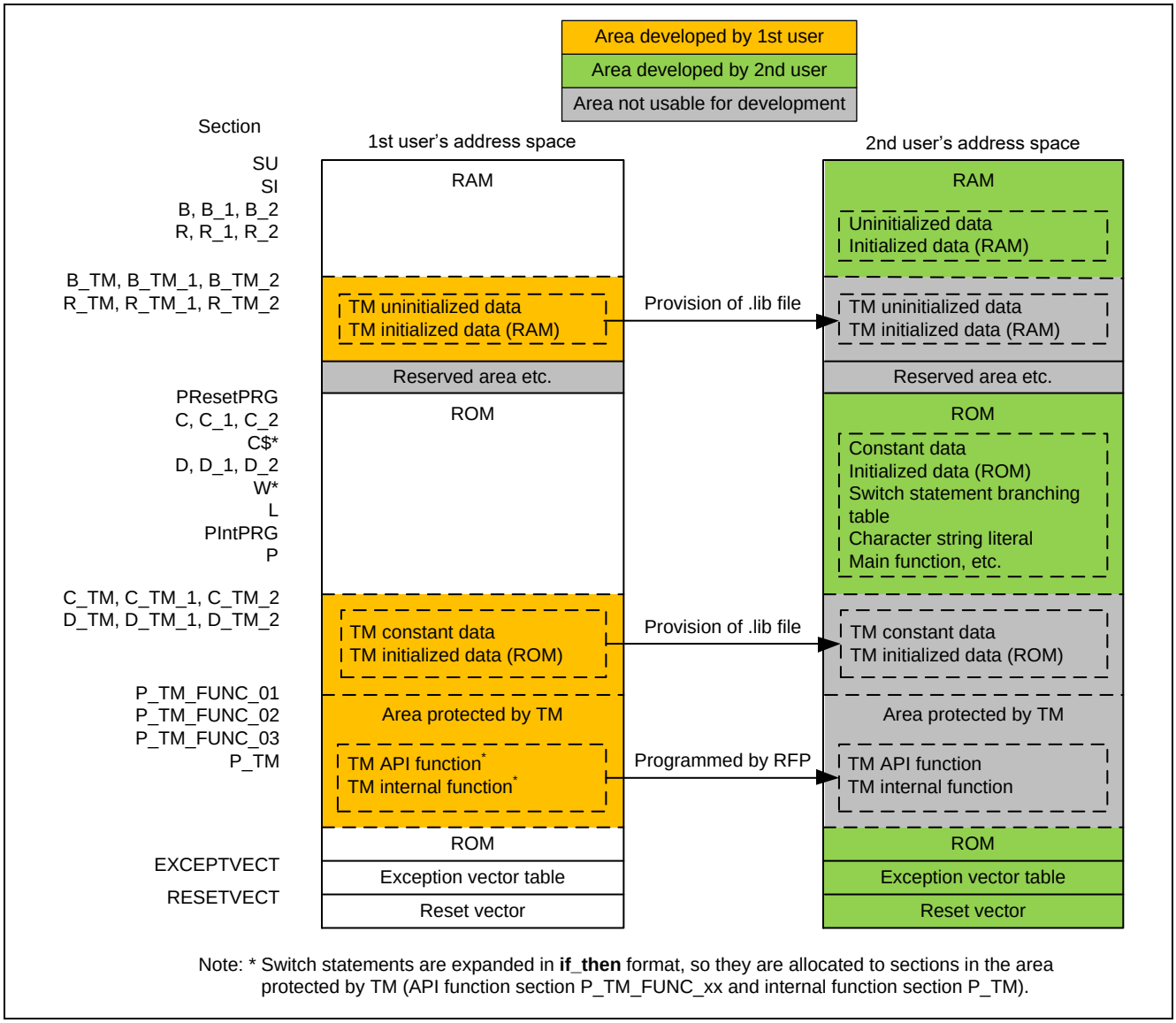


Figure 2.2 Section Diagram

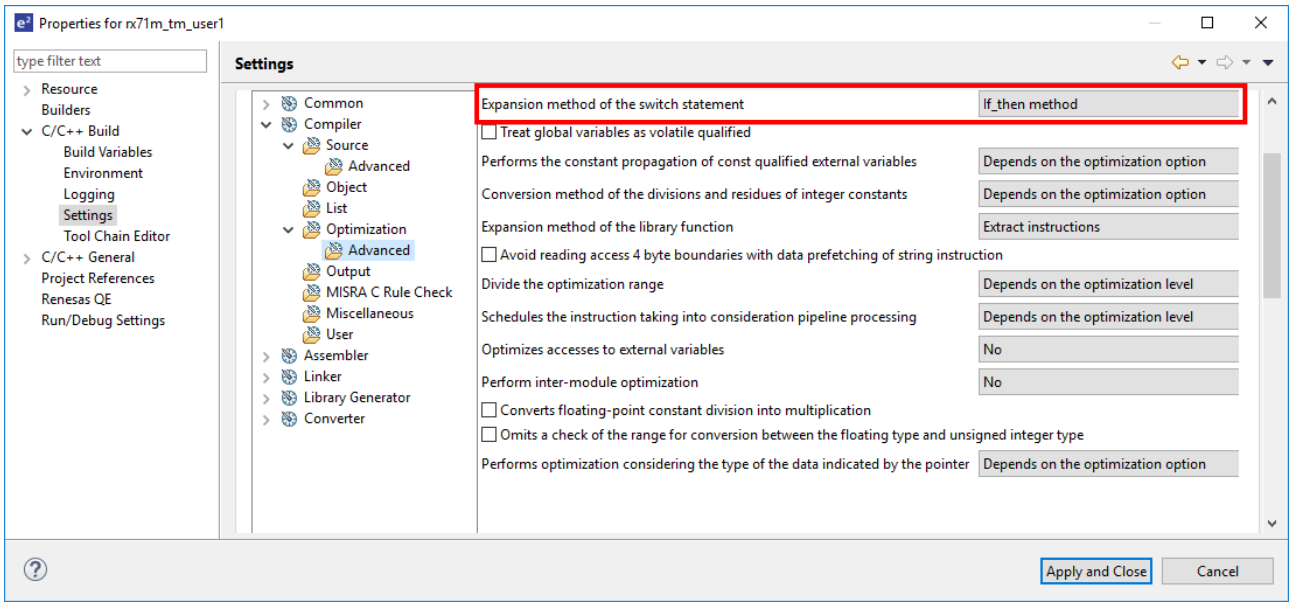
The procedure for setting the code expansion format for switch statements is described below:

1. In e^2 studio, right-click the target project and select **Properties**

2. Under **C/C++ Build** select **Settings**. Switch to the **Tool Settings** tab and select **Advanced** under **Optimize**.

3. In the **Expansion method of the switch statement**: list box, select **if_then method**.

4. Click the **Apply** button.



2.1.2 ROM Option Settings

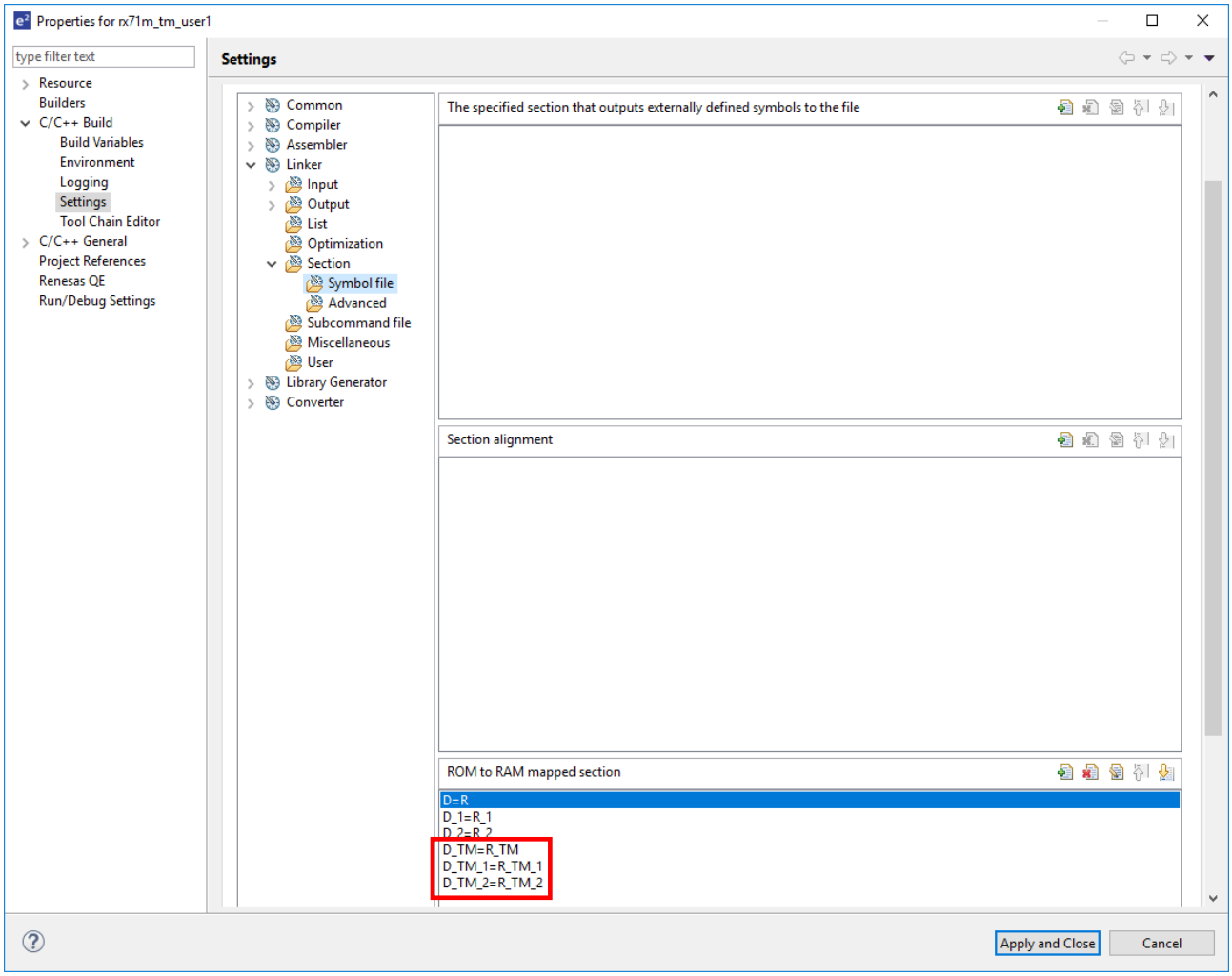
ROM options are used to secure areas in ROM and RAM for the initialized data area and to relocate symbols defined in ROM sections to addresses in RAM sections.

ROM option settings are specified by default for sections D and R only. When creating a new project, it is necessary to enter ROM option settings for the initialized data areas (D_TM and R_TM) used by the first user.

The procedure for setting ROM options is described below:

1. In e2 studio, right-click the target project and select **Properties**.
2. Under **C/C++ Build** select **Settings**. Switch to the **Tool Settings** tab and select **Output** under **Linker**.
3. Add the sections to be mapped from ROM to RAM.

   (In this application note the additions are D_TM = R_TM, D_TM_1 = R_TM_1, and D_TM_2 = R_TM_2.)
4. Click the **Apply** button.



For detailed information about ROM options, see the latest version of the RX Family C/C++ Compiler: User's Manual.

2.1.3 Section Initialization

When the _INITSCT function in **resetprg.c** a file generated automatically by e² studio, is called, uninitialized data sections are zero-initialized and initialized data sections are copied from the ROM area to the RAM area.

The user must declare the sections to be initialized in section initialization tables (DTBL and BTBL). Add the lines indicated below to the section initialization tables in the file **dbsct.c**, which is generated automatically by e² studio.

For detailed information about section initialization, see the latest version of the RX Family C/C++ Compiler: User's Manual.

```
#pragma section C C$DSEC
extern const struct {
    _UBYTE *rom_s;       /* Start address of the initialized data section in ROM */
    _UBYTE *rom_e;       /* End address of the initialized data section in ROM   */
    _UBYTE *ram_s;       /* Start address of the initialized data section in RAM */
}   _DTBL[] = {
    { __sectop("D"), __secend("D"), __sectop("R") },
    { __sectop("D_2"), __secend("D_2"), __sectop("R_2") },
    { __sectop("D_1"), __secend("D_1"), __sectop("R_1") },
    { __sectop("D_TM"), __secend("D_TM"), __sectop("R_TM") },
    { __sectop("D_TM_2"), __secend("D_TM_2"), __sectop("R_TM_2") },
    { __sectop("D_TM_1"), __secend("D_TM_1"), __sectop("R_TM_1") }
};
#pragma section C C$BSEC
extern const struct {
    _UBYTE *b_s;         /* Start address of non-initialized data section */
    _UBYTE *b_e;         /* End address of non-initialized data section */
}   _BTBL[] = {
    { __sectop("B"), __secend("B") },
    { __sectop("B_2"), __secend("B_2") },
    { __sectop("B_1"), __secend("B_1") },
    { __sectop("B_TM"), __secend("B_TM") },
    { __sectop("B_TM_2"), __secend("B_TM_2") },
    { __sectop("B_TM_1"), __secend("B_TM_1") }
};

#pragma section

/*
** CTBL prevents excessive output of L1100 messages when linking.
** Even if CTBL is deleted, the operation of the program does not change.
*/
_UBYTE * const _CTBL[] = {
    __sectop("C_1"), __sectop("C_2"), __sectop("C"),
    __sectop("W_1"), __sectop("W_2"), __sectop("W"),
    __sectop("L"), __sectop("SU"),
    __sectop("C$DSEC"), __sectop("C$BSEC"),
    __sectop("C$INIT"), __sectop("C$VTBL"), __sectop("C$VECT"),
    __sectop("C_TM_1"), __sectop("C_TM_2"), __sectop("C_TM")
};
```

2.2 Development Procedure for First User

2.2.1 Application Development

Generally speaking, symbols are used to call functions, but the second user cannot use this method to access functions in the area protected by TM because the second user is not provided with symbol information for these functions. In order to access functions in the area protected by TM, the second user requires address information for these functions.

The first user must therefore prepare function tables containing function address information for the area protected by TM. The function tables are allocated to the constant area, so they are provided to the second user in the library file (.lib). This enables the second user to call functions located in the area protected by TM by utilizing the function tables.

API functions in the area protected by TM are assigned sections and addresses individually. The first user should store the section start address of each API function in the function table. Information other than variables and function tables is omitted when the library file is created, so it is not possible to use function symbols in function tables.

An example of a function table is shown below:

```
void (*const tm_func_table1[])(void) = {
//;R_tm_func_01
    (void (*)())(__sectop("P_TM_FUNC_01")),
//;R_tm_func_02
    (void (*)())(__sectop("P_TM_FUNC_02")),
};
```

When the TM function is enabled, performing data access to the area protected by TM causes zeros to be read, so only section P should be allocated to the area protected by TM. All areas to which data access will be performed (section C, etc.) should be assigned to areas other than the area protected by TM.

2.2.2 Creating a Library File Containing Variables and Function Tables

To access functions in the area protected by TM, the second user requires address information for these functions. The first user must therefore prepare and supply to the second user a library file (.lib) containing variables and function tables for the area protected by TM.

The library file is created with the sample code **.lib file generation project provided to second user** by building the file **tm_api.c**, with information other than global variables, const variables, and function tables for the area protected by TM removed.

2.2.3 Enabling the TM Function

RFP is used to enable the TM function. Enabling of the TM and program writing to the area protected by TM take place at the same time.[*1]

The first user creates mot files from which all code except that related to the TM enable flag register (TMEF register), TM identification data register (TMINF register), and area protected by TM has been removed, and writes them to the target device.

The TM function is enabled by setting the TMEF bits in the TMEF register to 000b.[*2] The TMINF register stores the code used to identify programs in the area protected by TM.

To make settings to the TMEF register and TMINF register, make the modifications shown below to the file **vecttbl.c**, which is generated automatically by e^2 studio.

For RX64M, RX71M, RX66T, and RX72T.

```
#pragma address __TMEFreg=0x00120048          // TMEF register
const unsigned long __TMEFreg = 0xf8ffffff;
```

```
#pragma address __TMINFreg=0x00120060         // TMINF register
const unsigned long __TMINFreg = 0xffffffff;
```

For RX65N (Linear mode), RX651 (Linear mode), RX72M (Linear mode), RX72N (Linear mode), and RX66N (Linear mode).

```
#pragma address __TMEFreg=0xfe7f5d48          // TMEF register
const unsigned long __TMEFreg = 0xf8ffffff;
```

```
#pragma address __TMINFreg=0xfe7f5d10         // TMINF register
const unsigned long __TMINFreg = 0xffffffff;
```

Products having the dual bank function can be switched between the linear mode where the code flash memory is handled as one area and the dual mode where it is handled as two bank areas. The following products have the dual bank function:

- RX65N Group (Products with at least 1.5 Mbytes of code flash memory)
- RX651 Group (Products with at least 1.5 Mbytes of code flash memory)
- RX72M Group, RX72N Group, RX66N Group

For how to enable the TM function in dual mode, see 2.2.5(3) "Enabling the TM function in Dual Mode".

For details on enabling the TM function, see "How to Enable the TM Function" in the "Flash Memory" section in User's Manual: Hardware of the product used.

For detailed information on RFP, see the latest version of the Renesas Flash Programmer Flash memory programming software User's Manual.

Notes: 1. It is also possible to enable the TM function internally by means of self-programming.
2. It is not possible to enable the TM function by using the e2 studio environment (download). RFP must be used to write to the microcontroller.

2.2.4 Disabling the TM Function

Using RFP, all areas including the area protected by TM are erased and the TM function can be disabled.

For detailed information on RFP, see the latest version of the Renesas Flash Programmer Flash memory programming software User's Manual.

2.2.5 Using the TM Function in Dual Mode

When using the TM function in dual mode, if the second user switches the startup bank, API functions in the area protected by TM of Bank 1 are called. For that reason, the first user needs to program in the device by copying the area protected by TM of Bank 0 to the area protected by TM of Bank 1. Also, the first user needs to set the device to dual mode beforehand.

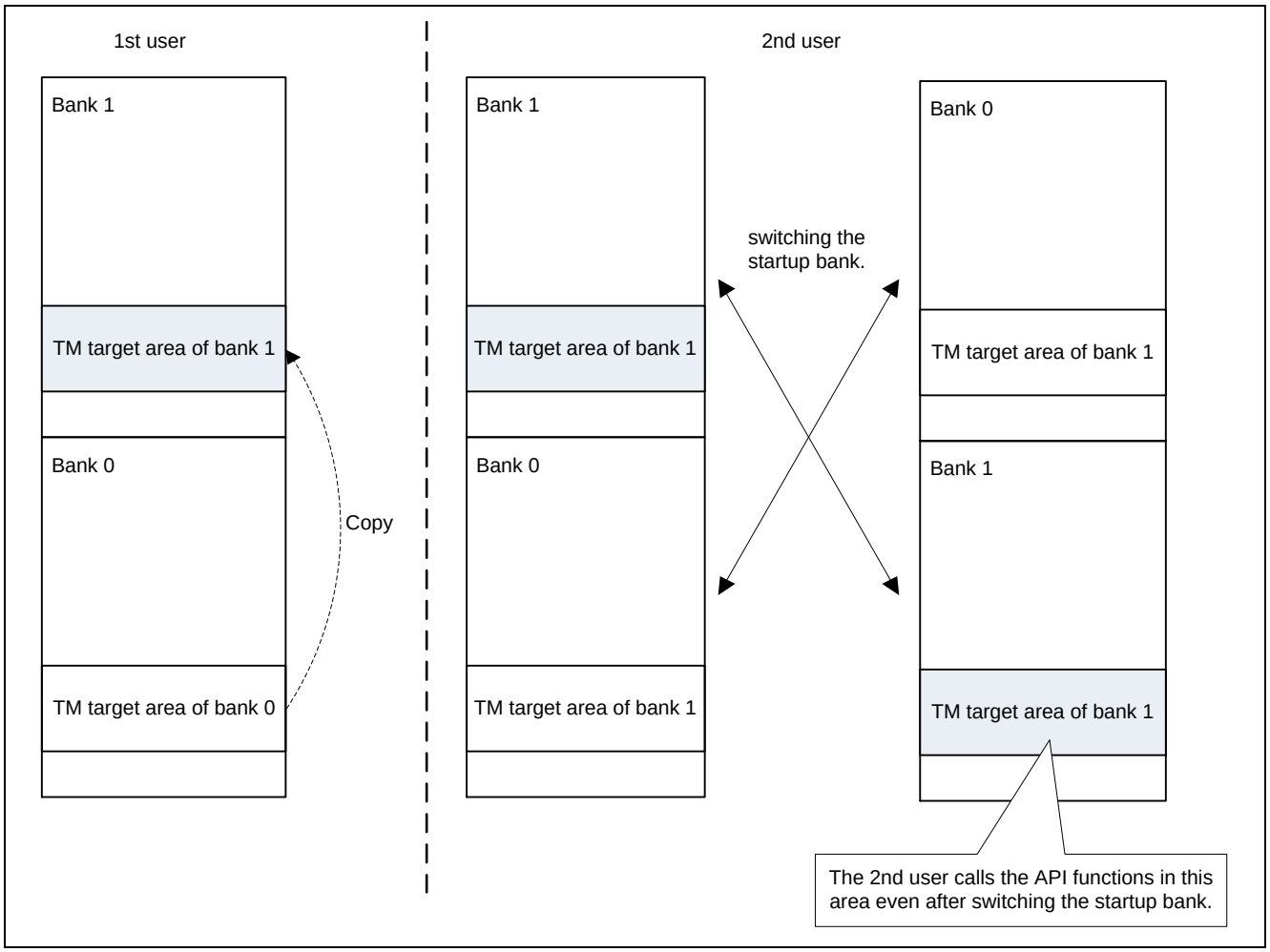


Figure 2.3 TM Target Area of Bank 0 and TM Target Area of Bank 1

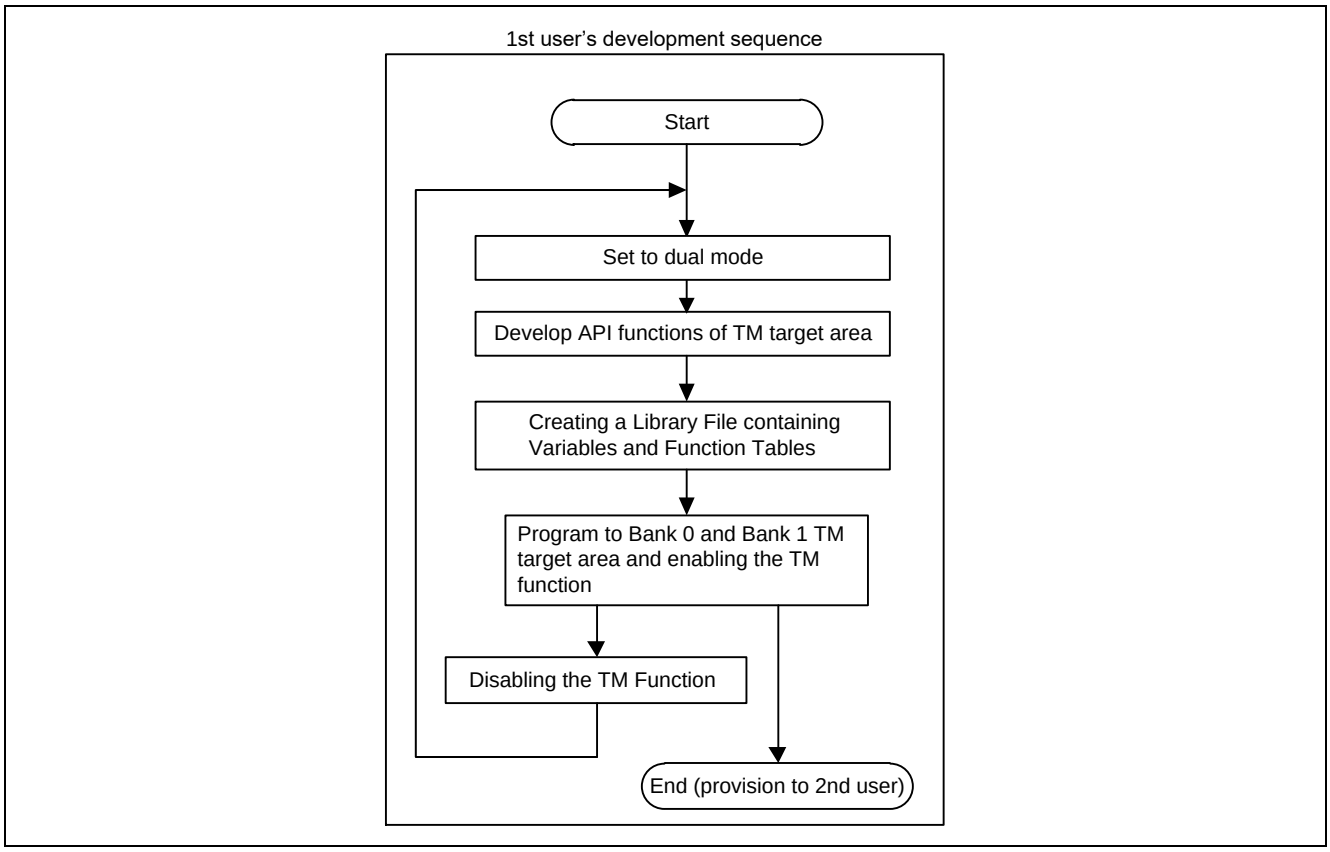


Figure 2.4 Development Sequence When Using TM Function in Dual Mode

(1) Setting Dual Mode

The first user sets the device to dual mode by creating mot files in which the BANKMD bit of the MDE register of the option setting memory is set to 000b, and using RFP to write the program, and then resetting.

Set the device to dual mode before writing the program in the area protected by TM and enabling the TM function.

To set the BANKMD bit of the MDE register, correct the auto generated **vecttbl.c** file as follows, and use RFP to write the program in the option setting memory.

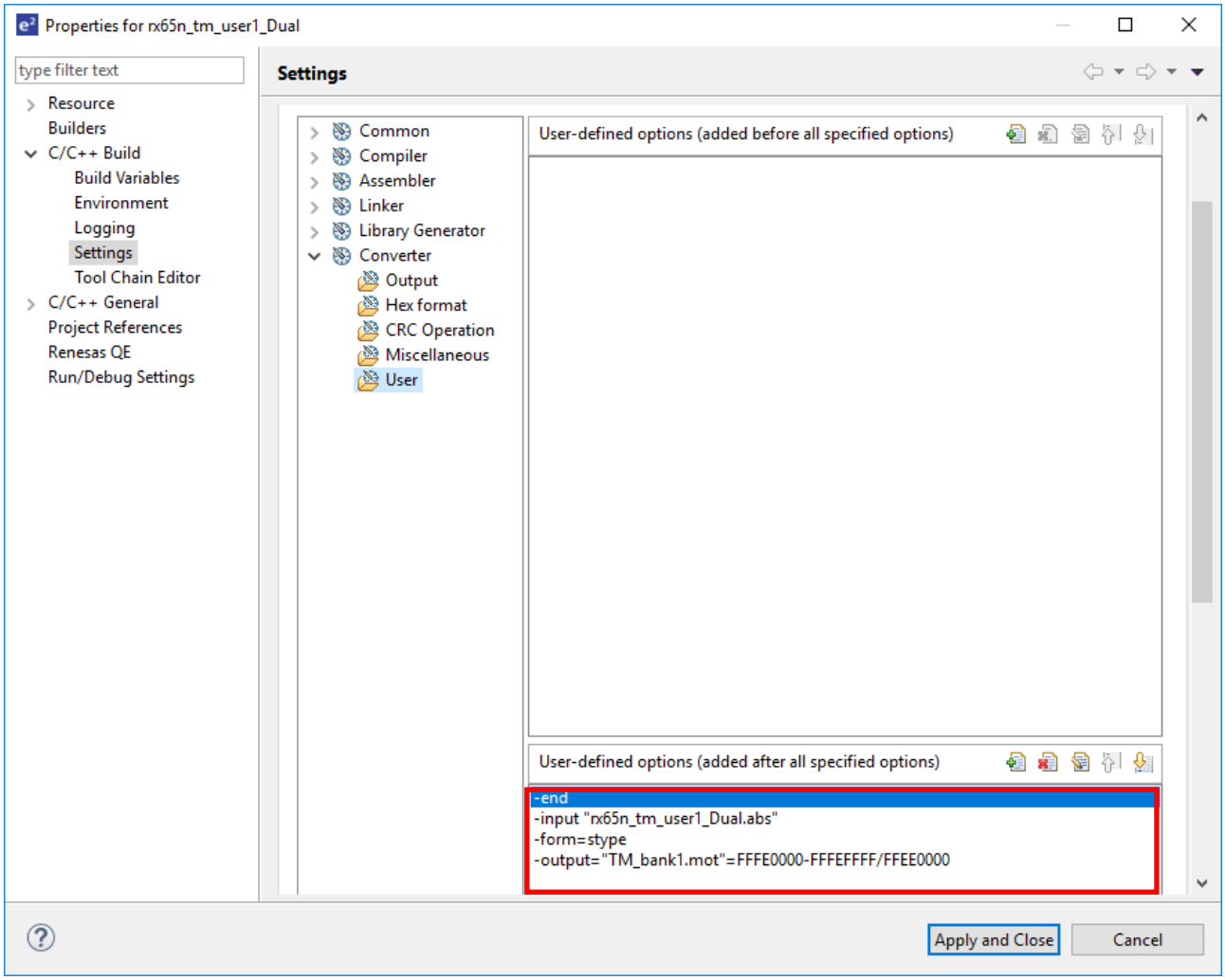
```
#pragma address __MDEreg=0xfe7f5d00          // MDE register (Single Chip Mode)
#ifdef __BIG
    const unsigned long __MDEreg = 0xffffff88;    // big
#else
    const unsigned long __MDEreg = 0xffffff8f;    // little
#endif
```

(2) Copying the area protected by TM

The first user creates mot files in which the area protected by TM of Bank 0 is copied to the area protected by TM of Bank 1, and uses RFP to write the program in the area protected by TM of Bank 1.

The following explains how to copy the area protected by TM and add the setting to output the mot files.

1. In e2 studio, right-click the target project and select **Properties**.
2. Under **C/C++ Build** select **Settings**. Switch to the **Tool Settings** tab and select **User** under **Converter**.
3. Add Options.
4. Click the **Apply** button.



The following indicates the content of options to be added.

Table 2.1 Content of Options to be Added

Options to be Added	Contents
-end	Executes the link option column that was set before -end.
-input "rx65n_tm_user1_Dual.abs"	Specifies the input file. In this example, "rx65n_tm_user1_Dual.abs" that was output by the rx65n_tm_user1_Dual sample code is specified as the input file.
-form=stype	Specifies Motorola S format file (mot file) as an output format.
-output="TM_bank1.mot"=FFFE0000-FFFEFFFF/FFEE0000	Outputs data to "TM_bank1.mot" file by specifying FFFE 0000h to FFFE FFFFh as an output range, which is the area protected by TM of Bank 0, and changing the first load address[*1] on the mot file to FFEE 0000h, which is the first address of the area protected by TM of Bank 1. The address of the area protected by TM differs depending on the product; change the setting according to the product you use.

Note: [*1] For the specification of the load address, use RX Family C/C++ compiler Version 3.00.00 or later. For detailed information about the load address and other options, see the latest version of the CC-RX Compiler User's Manual (R20UT3248).

(3) Enabling the TM function in Dual Mode

In order to enable the TM function in dual mode, set both the TMEF bit and TMEFDB bit in the TMEF register. Also, in the TMINF register, you can store code that can identify the program in the area protected by TM.

In order to set the TMEF bit and TMEFDB bit in the TMEF register and the TMINF register, correct the automatically created **vecttbl.c** file as follows, and use RFP to write the program in the option setting memory.

```
#pragma address __TMEFreg=0xfe7f5d48          // TMEF register
const unsigned long __TMEFreg = 0x88ffffff;
```

```
#pragma address __TMINFreg=0xfe7f5d10         // TMINF register
const unsigned long __TMINFreg = 0xffffffff;
```

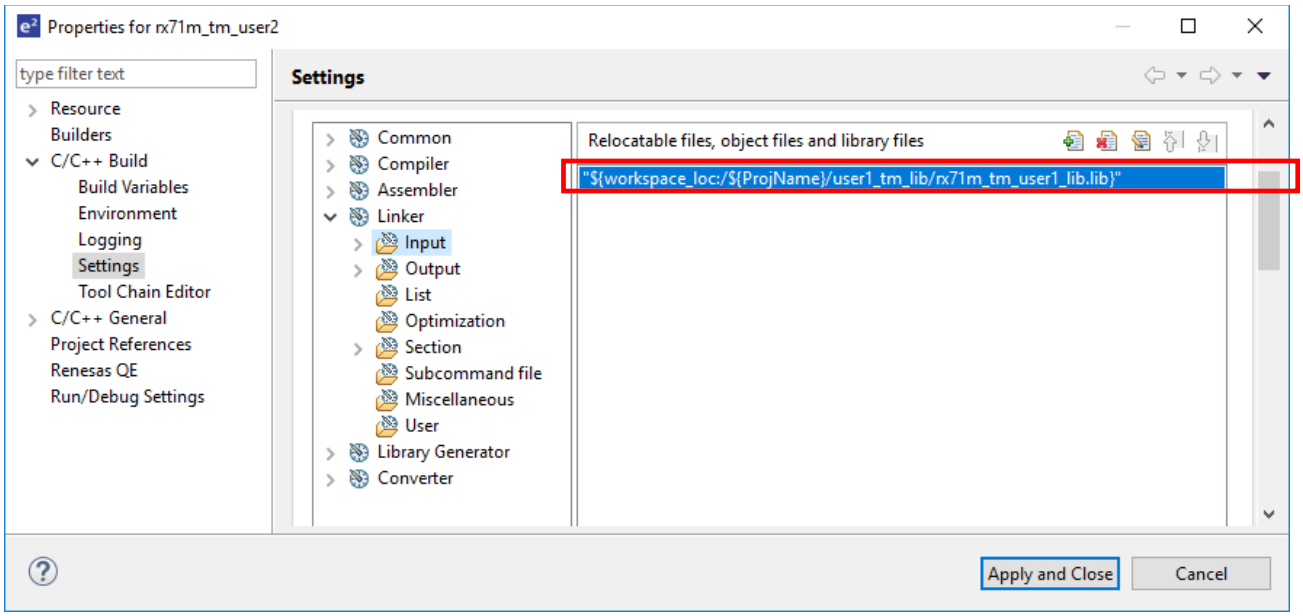
2.3 Development Procedure for Second User

2.3.1 Application Development

Using the provided library file (.lib) containing variables and function tables, the second user can develop an application.*

The procedure for linking the library file (.lib) is described below:

1. In e2 studio, right-click the target project and select **Properties**.
2. Under **C/C++ Build** select **Settings**. Switch to the **Tool Settings** tab and select **Input** under **Linker**.
3. Add the provided .lib file.
4. Click the **Apply** button.



The second user should use the function tables in the provided library file (.lib) to call API functions in the area protected by TM.

An example of calling API functions using function tables is shown below.

```
/* ==== Initialize LED ports ==== */
tm_func_table1[R_TM_PORT_INIT]();

/* ==== LEDs ON ==== */
tm_func_table2[R_TM_SET_LED](*table);

/* ==== LEDs OFF ==== */
tm_func_table1[R_TM_LED_OFF]();
```

Notes: *. The second user must not allocate programs to the area protected by TM when developing applications. This will result in errors during programming because the TM function is enabled.

2.3.2 Debugging an Application after Development

The e2 studio integrated development environment can be used to download programs to areas other than that protected by TM for application debugging.

The e2 studio integrated development environment offers the options when downloading programs to “overwrite without erasing” or to “erase and then overwrite.” The default setting is “erase and then overwrite,” but when the TM function is enabled it is not possible to erase programs in the area protected by TM.

3. Sample Code

Sample code for how to set sections, how to create a function table of API functions, and how to call API functions, which are explained in Section 2 of this application note, are provided for your reference.

Sample code can be downloaded from the Renesas Electronics website.

3.1 Sample Code Configuration

Table 3.1 shows sample code for RX71M, and **Table 3.2** shows sample code for RX65N.

Table 3.1 Sample code for RX71M[*1]

File Name	Outline
rx71m_tm_user1	Project developed by first user
rx71m_tm_user1_lib	.lib file generation project provided to second user
rx71m_tm_user2	Project developed by second user

Note: [*1] To use the linear mode of RX65N, RX651 RX72M, RX72N, and RX66N, and use RX64M, RX66T, and RX72T, refer to this sample code.

Table 3.2 Sample code for RX65N[*1]

File Name	Outline
DualMode.mot	mot file for the first user to set the device to the dual mode
rx65n_tm_user1_Dual	Project developed by first user (Dual mode)
rx65n_tm_user1_Dual_lib	.lib file generation project provided to second user
rx65n_tm_user2_Dual	Project developed by second user

Note: [*1] To use the dual mode of RX651, RX72M, RX72N, and RX66N, refer to this sample code.

3.2 Operation Confirmation Conditions

The sample code accompanying this application note has been run and confirmed under the conditions below.

Table 3.3 Operation Confirmation Conditions 1

Item	Contents
Target project	rx71m_tm_user1 rx71m_tm_user1_lib rx71m_tm_user2
MCU used	R5F571MLCDFC (RX71M Group)
Code flash memory capacity	4 MB
Operating frequency	Main clock: 24 MHz PLL: 240 MHz (main clock divided by 1 and multiplied by 10) System clock (ICLK): 120 MHz (PLL divided by 2) Peripheral module clock A (PCLKA): 120 MHz (PLL divided by 2) Peripheral module clock B (PCLKB): 60 MHz (PLL divided by 4) Flash interface clock (FCLK): 60 MHz (PLL divided by 4)
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics e^2 studio Version: 7.6.0
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V.3.01.00
iodefine.h version	V1.0A
Endian	Little endian
Operating mode	Single-chip mode
Processor mode	Supervisor mode
Sample code version	rx71m_tm_user1: Version 1.10 rx71m_tm_user1_lib: Version 1.10 rx71m_tm_user2: Version 1.11
Board used	Renesas Starter Kit+ for RX71M
Tools used	Renesas Flash Programmer V3.06.01

Table 3.4 Operation Confirmation Conditions 2

Item	Contents
Target project	DualMode.mot rx65n_tm_user1_Dual rx65n_tm_user1_Dual_lib rx65n_tm_user2_Dual
MCU used	R5F565NEDDFC (RX65N Group)
Code flash memory capacity	2MB
Operating frequency	Main clock: 24 MHz PLL: 240 MHz (main clock divided by 1 and multiplied by 10) System clock (ICLK): 120 MHz (PLL divided by 2) Peripheral module clock A (PCLKA): 120 MHz (PLL divided by 2) Peripheral module clock B (PCLKB): 60 MHz (PLL divided by 4) Flash interface clock (FCLK): 60 MHz (PLL divided by 4)
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics e^2 studio Version: 7.6.0
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V.3.01.00
iodefine.h version	V2.2
Endian	Little endian
Operating mode	Single-chip mode
Processor mode	Supervisor mode
Sample code version	rx65n_tm_user1_Dual: Version 1.10 rx65n_tm_user1_Dual_lib: Version 1.10 rx65n_tm_user2_Dual: Version 1.10
Board used	Renesas Starter Kit+ for RX65N-2MB
Tools used	Renesas Flash Programmer V3.06.01

3.3 Contents of Sample Code

Table 3.5 shows the contents of each sample code.

Table 3.5 Contents of Each Sample Code

Sample Code	Contents
rx71m_tm_user1	• Stores the API function that turns ON the LEDs loaded to Renesas Starter Kit+ in the area protected by TM. • Stores outside the area protected by TM an API function table, variables and constants used by API functions. (See 2.1.1 "Section Settings", 2.2.1 "Application Development" for details.) • Enabling the TM function (See 2.2.3 "Enabling the TM Function" for details.)
rx71m_tm_user1_lib	• Outputs the constants and the table of API functions used in the area protected by TM as a library file (.lib). (See 2.2.2 "Creating a Library File Containing Variables and Function Tables" for details.)
rx71m_tm_user2	• Link to the library file (See 2.3.1 "Application Development" for details.) • Calls the API functions in the area protected by TM. (See 2.3.1 "Application Development" for details.) When RX64M, RX71M, RX66T and RX72T are used, an error occurs in verification after writing TMEF bits are 111b. To verify, it is necessary to use a mot file with the TMEF bits set to 000b.
DualMode.mot	• mot file for setting to the dual mode (See 2.2.5(1) "Setting Dual Mode" for details.)
rx65n_tm_user1_Dual	• Stores the API function that turns ON the LEDs loaded to Renesas Starter Kit+ in the area protected by TM. • Stores outside the area protected by TM an API function table, variables and constants used by API functions. (See 2.1.1 "Section Settings", 2.2.1 "Application Development" for details.) • Enabling the TM function (See 2.2.3 "Enabling the TM Function" for details.) • Copies the area protected by TM of Bank 0 to the area protected by TM of Bank 1. (See 2.2.5(2) "Copying the area protected by TM" for details.)
rx65n_tm_user1_Dual_lib	• Outputs the constants and the table of API functions used in the area protected by TM as a library file (.lib). (See 2.2.2 "Creating a Library File Containing Variables and Function Tables" for details.)
rx65n_tm_user2_Dual	• Link to the library file (See 2.3.1 "Application Development" for details.) • Calls the API functions in the area protected by TM. (See 2.3.1 "Application Development" for details.)

3.4 Section Configuration

Table 3.6 lists section information for the sample code.

For information on how to add, modify, or erase sections, see the latest version of the RX Family C/C++ Compiler:User's Manual.

Table 3.6 Sample Code Section Information

Section Name	Area	First User	Second User	Description
SU	RAM	-	-	User stack area
SI		-	-	Interrupt stack area
B_1		-	-	Uninitialized data area Initialized data area (RAM)
R_1		-	-	
B_2		-	-	
R_2		-	-	
B		-	-	
R		-	-	
B_TM_1	0000 F000	Addition	Addition	Used by area protected by TM • Uninitialized data area • Initialized data area (RAM)
R_TM_1		Addition	Addition	
B_TM_2		Addition	Addition	
R_TM_2		Addition	Addition	
B_TM		Addition	Addition	
R_TM		Addition	Addition	
PResetPRG	User area	-	-	Power-on reset PC
C_1		-	-	Constant area
C_2		-	-	
C		-	-	
C$*		-	-	
D_1		Addition	Addition	Initialized data area (ROM) Changed to **D** from **D***
D_2		Addition	Addition	
D		Change	Change	
W*		-	-	Switch statement branching table area
L		-	-	Literal area
PIntPRG		-	-	Interrupt vector table
P		-	-	Program area
C_TM_1	FFFD F000	Addition	Addition	Used by area protected by TM • Constant area
C_TM_2		Addition	Addition	
C_TM		Addition	Addition	
D_TM_1		Addition	Addition	Used by area protected by TM • Initialized data area (ROM)
D_TM_2		Addition	Addition	
D_TM		Addition	Addition	
P_TM_FUNC_01	FFFE 0000	Addition	Addition	API function 01 in area protected by TM
P_TM_FUNC_02	FFFE 0100	Addition	Addition	API function 02 in area protected by TM
P_TM_FUNC_03	FFFE 0200	Addition	Addition	API function 03 in area protected by TM
P_TM	FFFE 0300	Addition	Not needed	Internal function in area protected by TM
EXCEPTVECT	User area	-	-	Exception vector table
RESETVECT		-	-	Reset vector

- : Default

3.5 Application Notes Used by Sample Code

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

- RX71M Group Initial Settings (R01AN2459EJ)
- RX65N Group, RX651 Group Initial Settings (R01AN3034)

The initial settings function of the above application note is used in the sample code of this application note.

If a newer version is available, replace the version included with this application note with the newer version. The latest version can be checked and downloaded from the Renesas Electronics website.

The sample code referenced in this application note includes partial modifications to the sample code provided in the application note RX71M Group Initial Settings Example. When upgrading to the latest version of the initial settings sample code, make sure to make the modifications listed below.

Table 3.7 lists the modifications to the initial settings sample code.

Table 3.7 Initial Settings Constants (r_init_clock.h) Modified in Sample Code

Constant Name	Value Before Change	Value After Change	Description
REG_SCKCR	20C2 1222h (PLL selected)	21C2 1222h (PLL selected)	Internal clock division ratio (SCKCR register setting value) Before change: ICLK = 240 MHz After change: ICLK = 120 MHz
REG_MEMWAIT	MEMWAIT_1WAIT	MEMWAIT_0WAIT	Memory wait cycles Before change: 1 wait cycle After change: 0 wait cycles

4. Cautions

The first user can also provide the second user with a section information file (.esi) containing section information.

To do this, the first user should export a section information file (.esi) from e^2 studio. The procedure for creating a section information file (.esi) is described below:

1. In e^2 studio, right-click the target project and select **Properties**.
2. Under **C/C++ Build** select **Settings**. Switch to the **Tool Settings** tab and select **Section** under **Linker**.
3. Click **Export**.
4. Select a destination to save the file.

The second user can then use e^2 studio to import the section information file (.esi).* The procedure for importing a section information file (.esi) is described below:

1. In e^2 studio, right-click the target project and select **Properties**.
2. Under **C/C++ Build** select **Settings**. Switch to the **Tool Settings** tab and select **Section** under **Linker**.
3. Click **Import**.
4. Open the provided section information file (.esi).

Notes: * Care should be exercised, because importing a section information file (.esi) causes all existing section settings to be erased and overwritten with the new ones.

5. Reference Documents

User's Manual: Hardware
RX64M Group User's Manual: Hardware (R01UH0377)
RX71M Group User's Manual: Hardware (R01UH0493)
RX65N Group, RX651 Group User's Manual: Hardware (R01UH0590)
RX66T Group User's Manual: Hardware (R01UH0749)
RX72T Group User's Manual: Hardware (R01UH0803)
RX72M Group User's Manual: Hardware (R01UH0804)
RX72N Group User's Manual: Hardware (R01UH0824)
RX66N Group User's Manual: Hardware (R01UH0825)
(The latest version can be downloaded from the Renesas Electronics website.)

User's Manual: Flash Memory
RX64M Group Flash Memory User's Manual: Hardware Interface (R01UH0435)
RX71M Group Flash Memory User's Manual: Hardware Interface (R01UH0435)
RX65N Group, RX651 Group Flash Memory User's Manual: Hardware Interface (R01UH0602)
(The latest version can be downloaded from the Renesas Electronics website.)

User's Manual: Renesas Flash Programmer
Renesas Flash Programmer Flash memory programming software. User's Manual (R20UT4307)
(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
RX Family C/C++ Compiler:User's Manual (R20UT3248)
(The latest version can be downloaded from the Renesas Electronics website.)

Revision History

		Description	
Rev.	Date	Page	Summary
1.00	Apr 01, 2015	-	First edition issued
1.01	Jun 18, 2015	3	Integrated development environment changed in table 2.1 Operation Confirmation Conditions
		17	4.4.2 Debugging an Application after Development changed
		-	Information on debugging removed from 7.2
2.00	Jun.06.19	-	Add Target Devices RX65N Group RX651 Group RX66T Group RX72T Group RX72M Group
3.00	Jan.20.20	-	Add Target Devices RX72N Group RX66N Group

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.

3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.

4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.

5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.

6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).

7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.

8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

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

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

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

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

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/.