RX Family Real-Time OS RI600PX V1.03.00

Release Notes

R20UT4377EJ0100
Rev.1.00
Oct 1, 2018

Contents

1. Packaged Tools

Depending on the model name, RI600PX has different forms of contract and offer as follows.

Product Name	Agreement Type	Contents
R0R5RX00PCW011	Evaluation License, Limited 1 host	A
R0R5RX00PCW01A	Evaluation License, Unlimited hosts	A
R0R5RX00PCW01K	Mass-production License, 3000 copies	A
R0R5RX00PCW01U	Mass-production License, Unlimited copies	A
R0R5RX00PCW01Z	Mass-production License, Unlimited copies, With source code	B

The following tools are provided.

Contents		Name	Version
B	A	Real-Time OS RI600PX Kernel Object	V1.03.00
		Command-line Configurator “cfg600px”	V1.01.01.001
		Plug-ins for CS+ for CC	
		Realtime OS Build Tool Plug-in (Common)	V3.02.01.01
		Realtime OS Build Tool Plug-in (RI600PX)	V3.00.00.06
		Realtime OS Analysis Control Plug-in (Common)	V3.00.00.03
		Realtime OS Analysis Control Plug-in (μITRON4)	V3.00.00.02
		Realtime OS Analysis Control Plug-in (RI600PX)	V3.00.00.02
		Realtime OS Resource Information Displaying Plug-in (Common)	V3.01.00.01
		Realtime OS Resource Information Displaying Plug-in (μITRON4)	V3.00.00.06
		Real-Time OS RI600PX Kernel Source Code	V1.03.00

2. User's Manual

The following user's manuals are included with this version. Please read these manuals together with this document.

Manual Name	Document Number
RI Series Real-Time Operating System User's Manual: Start	R20UT0751EJ0106
RI600PX Real-Time Operating System User's Manual: Coding	R20UT0964EJ0101
RI600PX Real-Time Operating System User's Manual: Debug	R20UT0950EJ0100
RI Series Real-Time Operating System User's Manual: Message	R20UT0756EJ0105

These PDF files are provided by this package or Renesas Electronics Home page. You can read them using the Windows Start Menu after Installing this package.

3. Target Devices

The following devices are supported by the product.

- RX700 Series MCU with Memory Protection Unit
- RX600 Series MCU with Memory Protection Unit
- RX200 Series MCU with Memory Protection Unit

4. Operating Environment

Below is described the operating environment for using the product.

4.1. Hardware Environment

- Processor: At least 1GHz (supported for hyper threading/multicore CPU)
- Memory capacity: 2 GB or more recommended. Minimum requirement is 1 GB or more （64-bit Windows® requires 2 GB or more）
- Display: Resolution at least 1024 x 768; at least 65,536 colors

4.2. Software Environment

The following OS are supported.

- Windows 7（32bit, 64bit）
- Windows 8.1（32bit, 64bit）
- Windows Vista（32bit, 64bit）
- Windows 10（32bit, 64bit）

Remark: It is recommended that the latest service pack is installed on any OS.

The following runtime libraries are required.

- .NET Framework 4.5.2
- Runtime library of Microsoft Visual C++ 2010 SP1

4.3. Supported Tools

The following tools are supported.

Tool Name	Manufacturer	Version
Integrated development environment CS+ for CC	Renesas Electronics	V3.02.00 or later
C/C++ Compiler CC-RX	Renesas Electronics	V2.04.01 or later recommended

5. Installation Notes

This section provides cautions for installation and uninstallation

5.1. Cautions for Installation

5.1.1. Caution for administrator privileges

Windows® administrator privileges are required to install the software.

5.1.2. Caution for execution environment

The .NET Framework and the Visual C++ runtime libraries are required to run the installer.

5.1.3. Caution for network drives

The software cannot be installed from a network drive.

It also cannot be installed to a network drive.

5.1.4. Caution for installation folder name

The available characters for specifying the installation folder are the same as for Windows®.

The 11 characters / * : < > ? | " \ ; , cannot be used. Folder names also cannot start or end with a space.

Specify folders as absolute paths. Do not use relative paths.

Use the backslash character (\) as the path separator for the installation folder. Do not use the forward slash (/).

5.1.5. Caution for modifying and repairing functions

To modify or repair the function of a tool that has already been installed, have the tool's installer package on hand, and run the installation program. The program maintenance program will start; select Modify or Repair.

Uninstall or change a program dialog boxes will cause an error.

5.1.6. Caution for required files after installation

The following folder is created after installation. Do not delete it, because it contains files that are necessary for the tools to run.

- If Windows® is 32bit and the installation drive is C:
  C:\Program Files\Common Files\Renesas Electronics CubeSuite+\

- If Windows® is 64bit and the installation drive is C:
  C:\Program Files (x86)\Common Files\Renesas Electronics CubeSuite+\

5.1.7. Caution for version of installed tools

If the newer version tool is already installed, the older version tool may not be installed.

5.1.8. Caution for starting installer

If the installer is started on a non-Japanese version of Windows®, then if the path contains multi-byte characters it will cause an error, and the installer will not start.

5.1.9. Enable Plug-ins

Plug-ins of this product may be disabled immediately after installation of this product. Please enable Plug-ins of this product. For details, refer to “7.7 Enable Plug-ins”.

5.2. Cautions for Uninstallation

5.2.1. Caution for administrator privileges

Windows® administrator privileges are required to uninstall the software.

5.2.2. Caution for uninstallation folder name

Depending on the order in which tools are uninstalled, the folders may not be completely deleted. If this happens, remove any remaining folders via Explorer or the like.

5.2.3. Caution for adding/repairing via other than the installer

If you added or modified files to the folders in which tools and manuals were installed using other means than the installers, they cannot be deleted during uninstallation.

5.2.4. Key Word for Uninstallation

There are two ways to uninstall this product.

- Use the integrated uninstaller (uninstalls CS+ for CC)
- Use separate uninstaller (uninstalls this product only)

To use the separate uninstaller, select the following from the Control Panel:

- Programs and Features

After the applet appears, delete the followings.

- CS+ Realtime OS Common Plugins
- CS+ Realtime OS RI600PX Plugins
- CS+ Realtime OS RI600PX Object Release, or CS+ Realtime OS RI600PX Source Release

6. Changes from previous released versions

This section provides changes in each release version of this product.

6.1. Changes in RI600PX V1.02.00

6.1.1. Kernel

There is no difference in the kernel.

6.1.2. Configurator

There is no difference in the configurator.

For the reason of 6.1.1 and 6.1.2, the package version is same as the previous version.

6.1.3. Realtime OS Build Tool Plug-in

(1) The “CS+ for CC” tools are supported

The “CS+ for CC” tools are supported. In addition, this plug-in of this version does not operate on the “CubeSuite+”.

(2) The help can be opened from [Realtime OS] tab and [System Configuration File Related Information] tab

6.1.4. Realtime OS Resource Information Displaying Plug-in

(1) The “CS+ for CC” tools are supported

The “CS+ for CC” tools are supported. In addition, this plug-in of this version does not operate on the “CubeSuite+”.

(2) The waiting factor which are showed by the “ID” are changed to “name”

The waiting factor which are showed by the “ID” are changed to “name”. It became intelligible the waiting factor.

(3) It became intelligible the tabs

The tab selection area is divided into two columns, and adds the icon to each tab.

(4) A part of messages is improved

A part of messages, for example error message, is improved.

(5) The following restriction is canceled.

The resource information panel does not get focus even if a display menu or a display button on toolbar is selected.

6.1.5. Sample programs of CS+

(1) Add new sample programs using FIT (Firmware Integration Technology) modules.

Add new sample programs using FIT modules. For details, refer to “9.2 Sample programs using Firmware Integration Technology”.

6.2. Changes in RI600PX V1.03.00

6.2.1. Kernel

（1） The RXv3 architecture is Supported.

To support the RXv3 architecture, we have changed that the RXv2 architecture’s library is linked when using the RXv3 architecture. The RXv3 architecture is compatible with the RXv2 architecture.

It should be noted that Table 2-1 Kernel libraries in 2.6.3. Kernel library of “RI600PX Real-Time Operating System User's Manual: Coding” (R20UT0964EJ0101) shall be replaced with the below table.

	Folder	Compiler version corresponding to the library	Corresponding CPU core	File name	Description
1	<ri_root>\library\rxv1	V1.02.01 or later	・RXv1 architecture	ri600lit.lib	For little endian
				ri600big.lib	For big endian
2	<ri_root>\library\rxv2	V2.01.00 or later	・RXv1 architecture ・RXv2 architecture ・RXv3 architecture	ri600lit.lib	For little endian
				ri600big.lib	For big endian

（2） The kernel version information

The version change is as follows.

Item	Before	After
TKERNEL_PRVER, T_RVER prver (the return value of “ref_ver” and “iref_ver”)	0x120	0x130

6.2.2. Sample programs for CS+

（1） Add new sample programs for RX66T

Because of supporting the RXv3 architecture, we have added new sample programs for CS+ for RX66T.

7. Cautions

7.1. Distinction of Version

By referring to this variable, the version of the kernel is distinguishable.

```
const UW _RI600PX_VERSION = < Value>;
```

The version of the kernel is denoted by form of “X,YY,ZZ,aa”. The bit31-24of _RI600PX_VERSION expresses “X”, the bit 23-16 expresses “YY”, the bit 15-8 expresses “ZZ”, and the bit 7-0 expresses “aa”

The actual versions are as follows.

Kernel version (Product version)	_RI600PX_VERSION	Note
V1.01.00（V1.01.00, V1.01.01）	(Not defined)	The past version
V1.02.00.03（V1.02.00）	0x01020004	The past version
V1.03.00（V1.03.00）	0x01030000	This version

7.2. Shift from a Previous Version

When you shift from a previous version, please be sure to re-build.

7.3. Timer Template File

The relation between timer template file provided by RI600PX and corresponded MCUs is shown as follows.

The timer template file is specified to “clock.template” in the system configuration file.

Please check the latest information of the timer template file on the product website of RI600PX.

figure 7-1 Timer template file

Template File	Corresponded MCUs
rx62t.tpl *1	RX600 Series RX62T Group
rx62n.tpl	RX600 Series RX62G Group RX600 Series RX62N Group RX600 Series RX621 Group
rx630.tpl	RX700 Series RX71M Group *2 RX600 Series RX66T Group *2 RX600 Series RX65N Group *2 RX600 Series RX651 Group *2 RX600 Series RX64M Group *2 RX600 Series RX630 Group RX600 Series RX63N Group RX600 Series RX631 Group

	RX600 Series RX634 Group RX600 Series RX63T Group RX200 Series RX21A Group RX200 Series RX230 Group RX200 Series RX231 Group RX200 Series RX23T Group RX200 Series RX24T Group RX200 Series RX24U Group

*1 Since this file is not included in RI600PX V1.02.00, it is necessary to get it separately.
Please contact our sales or distributor.

*2 Don't specify “CMT2” and “CMT3” as “clock.timer” in the system configuration file.

7.4. How to Build Kernel Source Code

Since the RI600PX kernel is provided in the library form, it does not usually need to build the kernel. The source code is only attached to R0R5RX00PCW01Z.

The kernel source code is stored in "< installation folder >\src600”. To build the kernel, set current folder to this folder, and run "nmake.exe"[1] as follows. The libraries will be generated under “< installation folder >\library".

- Command to generate libraries at “< installation folder >\library\rxv1"

  ```
  nmake release_install(RET)
  ```

  Note, the libraries attached to this product was built by using CC-RX V1.02.01.

- Command to generate libraries at "< installation folder >\library\rxv2"

  ```
  nmake –f make_rxv2.mak release_install(RET)
  ```

  Note, the libraries attached to this product was built by using CC-RX V2.01.00.

Please copy the installation folder to the writable folder if you don't have the write-access permission to the installation folder. After the build, copy the generated libraries to the "library\rxv1” or “library\rxv2” folder under the installation folder by the user who has write-access permission to the product installation folder.

[1] "nmake.exe" is a tool to build the project provided by Microsoft Corporation in United States.
"nmake.exe" is included in Microsoft Visual Studio 2008 etc.

7.5. Stack Consumption

7.5.1. Stack consumption of base clock interrupt handler (*clocksz1*, *clocksz2*, *clocksz3*)

The value of *clocksz1*, *clocksz2* and *clocksz3* described in appendix D.4 of “RI600PX Real-Time Operating System User's Manual: Coding” are as follows.

- *clocksz1*=120
- *clocksz2*=120
- *clocksz3*=200

7.5.2. Stack consumption of service calls (*svcsz*)

The kernel uses the system stack.

Please apply the maximum value of consumption of service calls used with the system and the following expression to *svcsz* described in appendix D.4 of “RI600PX Real-Time Operating System User's Manual: Coding”.

- Size consumed by function tree that makes the access exception handler (_RI_sys_access_exception()) + 16
  Size consumed by function tree that makes the timer initialization call-back function (_RI_init_cmt_knl()) + 8

figure 7-2 Stack usage of service-call

	Service call	Consumption	Note
Task management function			
1	cre_tsk	28	
2	acre_tsk	28	
3	del_tsk	28	
4	act_tsk	28	
5	iact_tsk	24	
6	can_act	24	
7	ican_act	24	
8	sta_tsk	28	
9	ista_tsk	24	
10	ext_tsk	60	The ext_tsk is called at the return from the task entry function.
11	exd_tsk	56	
12	ter_tsk	108	
13	chg_pri	36	
14	ichg_pri	52	
15	get_pri	28	
16	iget_pri	28	
17	ref_tsk	36	
18	iref_tsk	36	
19	ref_tst	28	
20	iref_tst	28	
Task dependent synchronization function			
21	slp_tsk	28	
22	tslp_tsk	28	
23	wup_tsk	32	
24	iwup_tsk	48	
25	can_wup	24	

	Service call	Consumption	Note
26	ican_wup	24	
27	rel_wai	104	
28	irel_wai	120	
29	sus_tsk	28	
30	isus_tsk	24	
31	rsm_tsk	28	
32	irsm_tsk	24	
33	frsm_tsk	28	
34	ifrsm_tsk	24	
35	dly_tsk	28	
Task exception handling function			
36	def_tex	28	
37	ras_tex	28	
38	iras_tex	24	
39	dis_tex	24	
40	ena_tex	28	
41	sns_tex	24	
42	ref_tex	24	
43	iref_tex	24	
Semaphore			
44	cre_sem	28	
45	acre_sem	28	
46	del_sem	48	
47	sig_sem	32	
48	isig_sem	48	
49	wai_sem	28	
50	pol_sem	24	
51	ipol_sem	24	
52	twai_sem	32	
53	ref_sem	28	
54	iref_sem	28	
Eventflag			
55	cre_flg	28	
56	acre_flg	28	
57	del_flg	48	
58	set_flg	48	
59	iset_flg	64	
60	clr_flg	24	
61	iclr_flg	24	
62	wai_flg	32	
63	pol_flg	28	
64	ipol_flg	28	
65	twai_flg	36	
66	ref_flg	28	
67	iref_flg	28	
Data queue			
68	cre_dtq	28	
69	acre_dtq	28	
70	del_dtq	48	
71	snd_dtq	32	
72	psnd_dtq	32	
73	ipsnd_dtq	48	

	Service call	Consumption	Note
74	tsnd_dtq	36	
75	fsnd_dtq	32	
76	ifsnd_dtq	52	
77	rcv_dtq	32	
78	prcv_dtq	32	
79	iprcv_dtq	48	
80	trcv_dtq	32	
81	ref_dtq	32	
82	iref_dtq	32	
Mailbox			
83	cre_mbx	28	
84	acre_mbx	28	
85	del_mbx	48	
86	snd_mbx	32	
87	isnd_mbx	52	
88	rcv_mbx	28	
89	prcv_mbx	28	
90	iprcv_mbx	28	
91	trcv_mbx	32	
92	ref_mbx	28	
93	iref_mbx	28	
Mutex			
94	cre_mtx	28	
95	acre_mtx	28	
96	del_mtx	52	
97	loc_mtx	28	
98	ploc_mtx	28	
99	tloc_mtx	32	
100	unl_mtx	44	
101	ref_mtx	28	
Message buffer			
102	cre_mbf	28	
103	acre_mbf	28	
104	del_mbf	48	
105	snd_mbf	36	
106	psnd_mbf	36	
107	ipsnd_mbf	56	
108	tsnd_mbf	36	
109	rcv_mbf	56	
110	prcv_mbf	56	
111	trcv_mbf	56	
112	ref_mbf	28	
113	iref_mbf	28	
Fixed-sized memory pool			
114	cre_mpf	28	
115	acre_mpf	28	
116	del_mpf	48	
117	get_mpf	28	
118	pget_mpf	28	
119	ipget_mpf	28	
120	tget_mpf	32	
121	rel_mpf	32	

	Service call	Consumption	Note
122	irel_mpf	48	
123	ref_mpf	28	
124	iref_mpf	28	
Variable-sized memory pool			
125	cre_mpl	80	
126	acre_mpl	80	
127	del_mpl	48	
128	get_mpl	88	
129	pget_mpl	104	
130	ipget_mpl	104	
131	tget_mpl	88	
132	rel_mpl	100	
133	ref_mpl	28	
134	iref_mpl	28	
Time management function			
135	set_tim	28	
136	iset_tim	28	
137	get_tim	28	
138	iget_tim	28	
Cyclic handler			
139	cre_cyc	28	
140	acre_cyc	28	
141	del_cyc	28	
142	sta_cyc	24	
143	ista_cyc	24	
144	stp_cyc	24	
145	istp_cyc	24	
146	ref_cyc	28	
147	iref_cyc	28	
Alarm handler			
148	cre_alm	28	
149	acre_alm	28	
150	del_alm	28	
151	sta_alm	24	
152	ista_alm	24	
153	stp_alm	24	
154	istp_alm	24	
155	ref_alm	28	
156	iref_alm	28	
System state management function			
157	rot_rdq	28	
158	irot_rdq	24	
159	get_tid	28	
160	iget_tid	28	
161	loc_cpu	24	
162	iloc_cpu	16	
163	unl_cpu	28	
164	iunl_cpu	24	
165	dis_dsp	16	
166	ena_dsp	28	
167	sns_ctx	24	
168	sns_loc	24	

	Service call	Consumption	Note
169	sns_dsp	24	
170	sns_dpn	24	
171	vsta_knl	88	The system stack is used after initializing ISP.
172	ivsta_knl	88	
173	vsys_dwn	24	
174	ivsys_dwn	24	
Interrupt management function			
175	chg_ims	28	
176	ichg_ims	16	
177	get_ims	28	
178	iget_ims	28	
179	Kernel interrupt handler	36	When a kernel interrupt handler ends, 36 bytes of the system stack is consumed from just before generating of the interrupt.
System configuration management function			
180	ref_ver	28	
181	iref_ver	28	
Object reset function			
182	vrst_dtq	40	
183	vrst_mbx	28	
184	vrst_mbf	40	
185	vrst_mpf	40	
186	vrst_mpl	76	
Memory object management function			
187	ata_mem	48	
188	det_mem	44	
189	sac_mem	60	
190	vprb_mem	28	
191	ref_mem	52	

7.5.3. When the kernel library is built

Please note that the stack consumption might change when a version and/or an optional setting of the compiler are changed and the kernel library is built.

7.6. Cautions When Using global optimization of compile option

It is not able to specify global optimization (-ip_optimize, -merge_files, -whole_program) to the program embedded RI600PX.

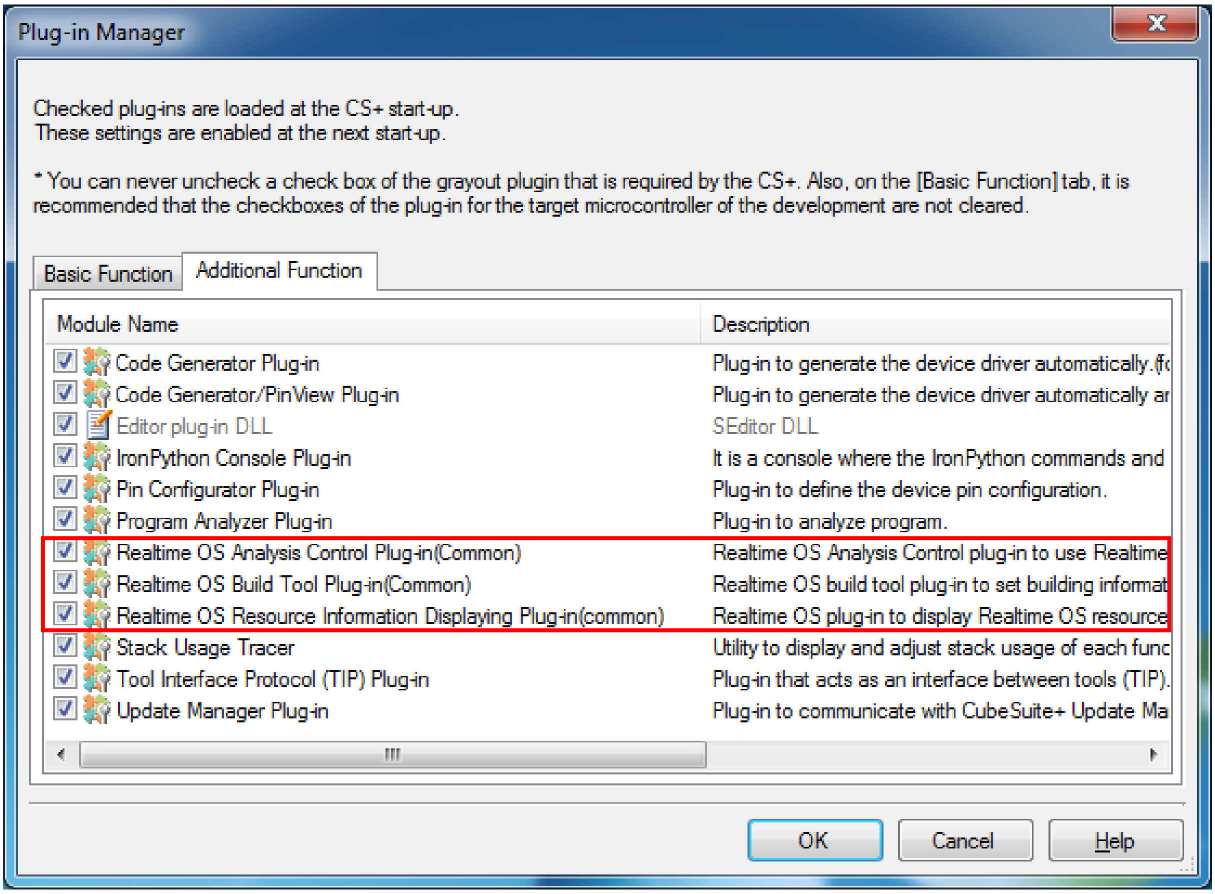
7.7. Enable Plug-ins

Plug-ins of this product may be disabled immediately after installation of this product. If plug-ins are disabled, the problem of being unable to build arises.

Please enable following Plug-ins by [Additional Function] tab in [Plug-in Manager] dialog box of the CS+ for CC

- Realtime OS Analysis Control Plug-in(Common)
- Realtime OS Build Tool Plug-in(Common)
- Realtime OS Resource Information Displaying Plug-in(common)

figure 7-3 Plug-in Manager



Plug-in Manager

Checked plug-ins are loaded at the CS+ start-up.
These settings are enabled at the next start-up.

* You can never uncheck a check box of the grayout plugin that is required by the CS+. Also, on the [Basic Function] tab, it is recommended that the checkboxes of the plug-in for the target microcontroller of the development are not cleared.

Module Name	Description
☑ Code Generator Plug-in	Plug-in to generate the device driver automatically.(f
☑ Code Generator/PinView Plug-in	Plug-in to generate the device driver automatically ar
☑ Editor plug-in DLL	SEditor DLL
☑ IronPython Console Plug-in	It is a console where the IronPython commands and
☑ Pin Configurator Plug-in	Plug-in to define the device pin configuration.
☑ Program Analyzer Plug-in	Plug-in to analyze program.
☑ Realtime OS Analysis Control Plug-in(Common)	Realtime OS Analysis Control plug-in to use Realtime
☑ Realtime OS Build Tool Plug-in(Common)	Realtime OS build tool plug-in to set building informat
☑ Realtime OS Resource Information Displaying Plug-in(common)	Realtime OS plug-in to display Realtime OS resource
☑ Stack Usage Tracer	Utility to display and adjust stack usage of each func
☑ Tool Interface Protocol (TIP) Plug-in	Plug-in that acts as an interface between tools (TIP).
☑ Update Manager Plug-in	Plug-in to communicate with CubeSuite+ Update Ma

7.8. Create a CS+ Project

To create a project which uses this product, there are the following two methods.

- Divert the sample project attached to this product.
- Create a new project

7.8.1. Divert the sample project attached to this product

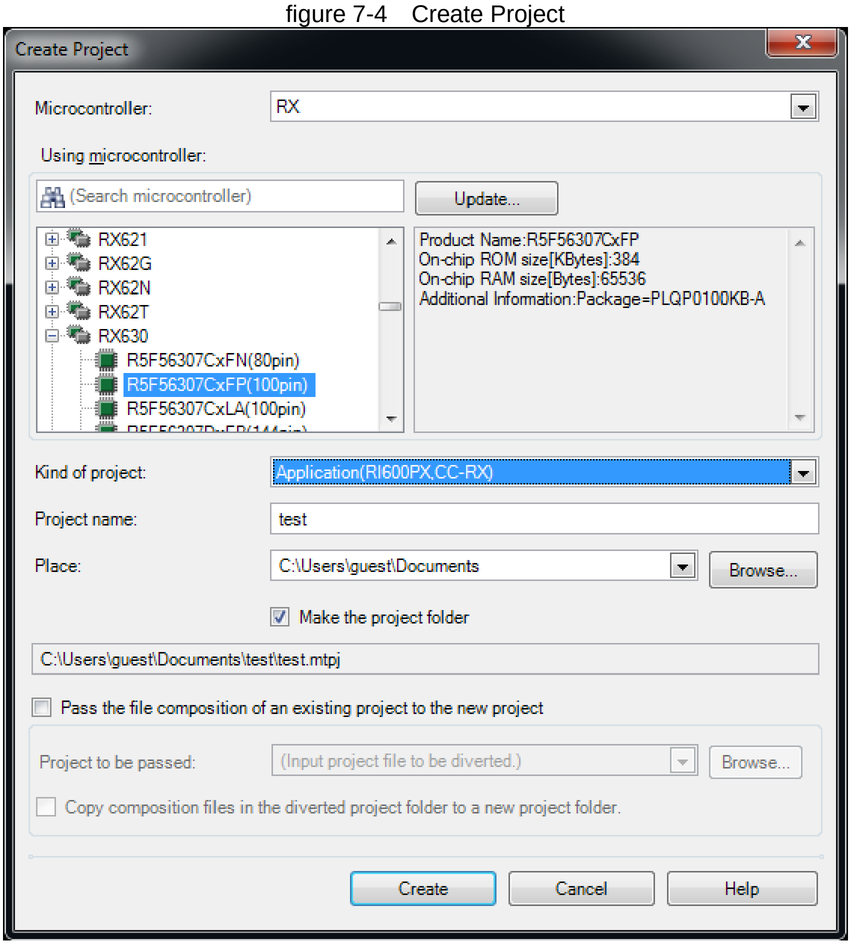
Select [RX] tab in [Open Sample Project] area of [Start] panel of the CS+, and choose the project named "RX???_RI600PX".

7.8.2. Create a new project

(1) Create a project

Press [Go] button in [Create New Project] area of [Start] panel of the CS+, then [Create Project] dialog box will be opened.

figure 7-4 Create Project



Create Project

Microcontroller: RX

Using microcontroller:

(Search microcontroller)

- RX621
- RX62G
- RX62N
- RX62T
- RX630
  - R5F56307CxFN(80pin)
  - R5F56307CxFP(100pin)
  - R5F56307CxLA(100pin)

Product Name:R5F56307CxFP
On-chip ROM size[KBytes]:384
On-chip RAM size[Bytes]:65536
Additional Information:Package=PLQP0100KB-A

Kind of project: Application(RI600PX,CC-RX)

Project name: test

Place: C:\Users\guest\Documents

☑ Make the project folder

C:\Users\guest\Documents\test\test.mtpj

□ Pass the file composition of an existing project to the new project

Project to be passed: (Input project file to be diverted.)

□ Copy composition files in the diverted project folder to a new project folder.

- [Microcontroller] : Select "RX"
- [Kind of project] : Select "Application(RI600PX,CC-RX)"

Press [Create] button, then a project will be generated.

(2) Register files

No files are registered immediately after project creation. Please register the following files according to “CHAPTER 2 SYSTEM BUILDING” in “RI600PX User's Manual: Coding”.

- Processing programs, such as tasks and handlers (refer to section 2.2 in “RI600PX User's Manual: Coding”)
- System configuration file (refer to section 2.3 in “RI600PX User's Manual: Coding”)
- User-own coding module (refer to section 2.4 in “RI600PX User's Manual: Coding”)

(3) Build options

Please set up suitable build options according to “2.5 Creating Load Module” and “2.6 Build Options” in “RI600PX User's Manual: Coding”.

7.9. Cautions for Realtime OS Resource Information Panel

7.9.1. View after Real-Time OS is initialized

View the Realtime OS Resource Information Panel after the Real-Time OS has been initialized. Before the Real-Time OS has been initialized, the information in the Realtime OS Resource Information Panel is undefined.

7.9.2. Use programs with debug information generated

When using the Realtime OS Resource Information Panel, download a program for which debug information has been generated. Downloading a program without debug information and viewing it in the Realtime OS Resource Information Panel will cause an error.

To generate debug information, under Build Tool, under the Link Options properties, set "Generate debug information" to "Yes".

8. Restrictions

8.1. Restrictions of CS+ for CC

8.1.1. Realtime OS Build Tool Plug-in

(1) **Multiple build modes**

Do not use multiple build modes for the following reasons.

- The configurator options are common to all build modes. Even if multiple build modes are used, the same configurator options are applied.
- Every time the build mode is changed, the path to the kernel_id.h file is added to [Additional include paths] of the build tool. Although the build-setting plug-in sets the correct path in [System include paths], the IDE adds the old path prior to the change of the build mode to [Additional include paths]. In the process of building, the build tool refers to the old path set by the IDE. This means that editing the configuration file to change the build mode before editing kernel_id.h, for example, will not be reflected in building.

(2) **Utilizing existing projects**

If you choose to recycle as the basis of a new project an existing project that does not contain any files such as sit.s which are generated by the configurator, and you select copy processing for the files you will be reusing, the missing files such as sit.s that are supposed to be grayed out in the project tree will be deleted from the project tree.

8.1.2. Realtime OS Resource Information Displaying Plug-in

(1) **Effect of resetting the display of waiting tasks (child nodes) on the display of the [Task] tabbed page**

Resetting the display of waiting tasks also resets the display of other tasks in the [Task] tabbed page. However, the information being displayed will be correct.

(2) **"Time Left" in "Realtime OS Resource Information Panel"**

The value displayed on the following items may become larger TIC_NUME than the original value at the maximum.

- "Time Left" item in [Task] tab
- "Time Left" item in [Cyclic Handler] tab
- "Time Left" item in [Alarm Handler] tab

The original value can be calculated by the following formulas.

- When (The value displayed on "Time Left") > TIC_NUME
  The original value = (The value displayed on "Time Left") – TIC_NUME
- When (The value displayed on "Time Left") ≤ TIC_NUME
  The original value = 0

9. Sample Programs

This section describes the sample program “RX630_RI600PX” which is provided by RI600PX V1.02.00

9.1. Sample programs of CS+

9.1.1. Summary

There are three domains, "Master domain", "Domain-A" and "Domain-B".

The master domain (domid #1) is "trusted domain". The master domain creates various objects that are required to execute domain-A and -B. The task that belongs to the master domain (MasterDom_Task) is created and activated by the system configuration file.

The domain-A (domid #2) and domain-B (domid #3) are not "trusted domain".

The task that belongs to the domain-A is AppDomA_Task, and the task that belongs to the domain-B is AppDomB_Task.

AppDomA_Task and AppDomB_Task access the global variable "g_ulSharedData" by using the semaphore (ID_SEM1) while controlling it exclusively.

And AppDomA_Task sends data to the data queue (ID_DTQ1), AppDomB_Task receives it.

Table 9-1 List of Objects (1/2)

Type	ID number, etc.	Description
Domain	1	Master domain Trusted domain Belonging task : "MasterDom_Task"
	2	Domain-A Untrusted domain Belonging task : "AppDomA_Task"
	3	Domain-B Untrusted domain Belonging task : "AppDomB_Task"
Task	ID_MASTERDOMTASK	Created and activated by the system configuration file
	ID_DOM_A_TASK	Created and activated by MasterDom_Task
	ID_DOM_B_TASK	Created and activated by MasterDom_Task
Semaphore	ID_SEM1	Created by MasterDom_Task Control to access to variable "g_ulSharedData" from AppDomA_Task and AppDomB_Task
Data Queue	ID_DTQ1	Created by MasterDom_Task Used to communicate between AppDomA_Task and AppDomB_Task The data queue area is generated in the "BS" section. This section is outside of memory objects.
Variable-sized memory pool	ID_MPL1	Created by MasterDom_Task It is dummy. The pool area is generated in the "BU_SH" section. This section is inside of memory_object[4].

Table 9-1 List of Objects (2/2)

Type	ID number, etc.	Description
Cyclic handler	ID_CYC1	Created and started by the system configuration file Rotate ready queue for AppDomA_Task and AppDomB_Task
Alarm handler	ID_ALM1	Created by the system configuration file It is dummy.
Interrupt handler	Relocatable vector #64	Defined by the system configuration file It is dummy.

9.1.2. File Composition

The “RX630_RI600PX” sample programs are stored in the folder shown below.

```
<CS+_root>\SampleProjects\RX\RX630_RI600PX
```

- <CS+_root>

  Indicates the installation folder of CS+.

  The default folder is "C:\Program Files\Renesas Electronics\CS+”.

(1) appli\source\reset folder

- resetprg.c

  This is the boot processing file. For details, refer to section 17.2 of “RI600PX Real-Time Operating System User's Manual: Coding”.

- dbsct.c

  This is the section information file. For details, refer to section 17.4 of “RI600PX Real-Time Operating System User's Manual: Coding”.

(2) appli\source\kernel folder

- sample.cfg

  This is the system configuration file. For details, refer to chapter 20 of “RI600PX Real-Time Operating System User's Manual: Coding”.

- access_exc.c

  This is the access exception handler. For details, refer to section 3.10 of “RI600PX Real-Time Operating System User's Manual: Coding”.

- init_cmt.c

  This is the base clock timer initialization routine. For details, refer to section 10.9 of “RI600PX Real-Time Operating System User's Manual: Coding”.

- sysdwn.c

  This is the system down routine. For details, refer to section 15.2 of “RI600PX Real-Time Operating System User's Manual: Coding”.

- handler.c

  Various kinds of handlers are implemented in this file.

(3) appli\source\master_dom folder

- master_dom.c

  The task which belongs to “Master domain” is implemented in this file.

(4) appli\source\dom_A folder

- dom_A.c

  The task which belongs to “Domain-A” is implemented in this file.

(5) appli\source\dom_B folder

- dom_B.c

  The task which belongs to “Domain-B” is implemented in this file.

(6) appli\source\common folder

- common.c

  Functions and variables shared by two or more domains are implemented in this file.

9.1.3. Memory Map

The “aligned section” linker option, which aligns the start address of the section to 16-bytes boundary, is specified for the sections indicated by parentheses “[ ]”. For details, see section 2.6.4 of “RI600PX Real-Time Operating System User's Manual: Coding”.

9.1.3.1. RAM area

Table 9-2 RAM area

Address	Section Order (setting for linker)	Description	Memory Object
0～0x0001FFFF	SI	System stack	Non-memory object
	BRI_RAM, RRI_RAM	Kernel data	
	BS, BS_1, BS_2, RS, RS_1, RS_2	Data only for handlers	
	[SURI_STACK]	User stack	
	[BU_MASTERDOM],BU_MASTERDOM_1, BU_MASTERDOM_2, RU_MASTERDOM,RU_MASTERDOM_1, RU_MASTERDOM_2	Data only for the master domain	memory_object[1]
	[BU_DOM_A], BU_DOM_A_1, BU_DOM_A_2, RU_DOM_A, RU_DOM_A_1, RU_DOM_A_2	Data only for the domain-A	memory_object [2]
	[BU_DOM_B], BU_DOM_B_1, BU_DOM_B_2, RU_DOM_B,RU_DOM_B_1,RU_DOM_B_2	Data only for the domain-B	memory_object [3]
	[BURI_HEAP], BU_SH, BU_SH_1, BU_SH_2, RU_SH,RU_SH_1,RU_SH_2	Shared data	memory_object[4]

9.1.3.2. ROM area

Table 9-3 ROM area

Address	Section Order (setting for linker)	Description	Memory Object
0xFFFF0000～ 0xFFFFFF7F	[PU_MASTERDOM], CU_MASTERDOM, CU_MASTERDOM_1, CU_MASTERDOM_2, DU_MASTERDOM, DU_MASTERDOM_1, DU_MASTERDOM_2	Code and constant only for the master domain	memory_object[5]
	[PU_DOM_A],CU_DOM_A,CU_DOM_A_1, CU_DOM_A_2,DU_DOM_A,DU_DOM_A_1, DU_DOM_A_2	Code and constant only for the domain-A	memory_object[6]
	[PU_DOM_B],CU_DOM_B,CU_DOM_B_1, CU_DOM_B_2,DU_DOM_B,DU_DOM_B_1, DU_DOM_B_2	Code and constant only for the domain-B	memory_object[7]
	[PU_SH], WU_SH, WU_SH_1, WU_SH_2, LU_SH, CU_SH, CU_SH_1, CU_SH_2, DU_SH,DU_SH_1,DU_SH_2	Shared code and constant	memory_object[8]
	INTERRUPT_VECTOR	Relocatable vector table	非メモリ・オブジェクト
	PRI_KERNEL	RI600PX code	
	CRI_ROM, DRI_ROM	RI600PX constant	
	C$*,PS,CS,CS_1,CS_2,DS,DS_1,DS_2	Code and constant only for handlers	
0xFFFFFF80～ 0xFFFFFFFF	FIX_INTERRUPT_VECTOR	Fixed vector table	

9.1.3.3. Memory objects

There are eight memory objects, these are registered in the system configuration file. The contents of registration of memory objects in the system configuration file are shown in below.

(1) memory_object[1] : Data only for the master domain

```
memory_object[1]{
    start_address = BU_MASTERDOM;
    end_address   = RU_MASTERDOM_2;
    acptn1      = 0x0001;        The operand-read access is permitted only to the master domain.
    acptn2      = 0x0001;        The operand-write access is permitted only to the master domain.
    acptn3      = 0;             The execution access is permitted to no domain.
};
```

(2) memory_object[2] : Data only for the domain-A

```
memory_object[2]{
    start_address = BU_DOM_A;
    end_address   = RU_DOM_A_2;
    acptn1      = 0x0002;        The operand-read access is permitted only to the domain-A.
    acptn2      = 0x0002;        The operand-write access is permitted only to the domain-A.
    acptn3      = 0;             The execution access is permitted to no domain.
};
```

(3) memory_object[3] : Data only for the domain-B

```
memory_object[3]{
    start_address = BU_DOM_B;
    end_address   = RU_DOM_B_2;
    acptn1       = 0x0004;        The operand-read access is permitted only to the domain-B.
    acptn2       = 0x0004;        The operand-write access is permitted only to the domain-B.
    acptn3       = 0;             The execution access is permitted to no domain.
};
```

(4) memory_object[4] : Shared data

```
memory_object[4]{
    start_address = BURI_HEAP;
    end_address   = RU_SH_2;
    acptn1       = TACP_SHARED;   The operand-read access is permitted to all the domains.
    acptn2       = TACP_SHARED;   The operand-write access is permitted to all the domains.
    acptn3       = 0;             The execution access is permitted to no domain.
};
```

(5) memory_object[5] : Code and constant only for the master domain

```
memory_object[5]{
    start_address = PU_MASTERDOM;
    end_address   = DU_MASTERDOM_2;
    acptn1       = 0x0001;        The operand-read access is permitted only to the master domain.
    acptn2       = 0;             The operand-write access is permitted to no domain.
    acptn3       = 0x0001;        The execution access is permitted only to the master domain.
};
```

(6) memory_object[6] : Code and constant only for the domain-A

```
memory_object[6]{
    start_address = PU_DOM_A;
    end_address   = DU_DOM_A_2;
    acptn1       = 0x0002;        The operand-read access is permitted only to the domain-A.
    acptn2       = 0;             The operand-write access is permitted to no domain.
    acptn3       = 0x0002;        The execution access is permitted only to the domain-A.
};
```

(7) memory_object[7] : Code and constant only for the domain-B

```
memory_object[7]{
    start_address = PU_DOM_B;
    end_address   = DU_DOM_B_2;
    acptn1        = 0x0004;
    acptn2        = 0;
    acptn3        = 0x0004;
};
```

The operand-read access is permitted only to the domain-B.

The operand-write access is permitted to no domain.

The execution access is permitted only to the domain-B.

(8) memory_object[8] : Shared code and data

```
memory_object[8]{
    start_address = PU_SH;
    end_address   = DU_SH_2;
    acptn1        = TACP_SHARED;
    acptn2        = 0;
    acptn3        = TACP_SHARED;
};
```

The operand-read access is permitted to all the domains.

The operand-write access is permitted to no domain.

The execution access is permitted to all the domains.

9.1.3.4. User stacks

The user stacks must be allocated to the outside of memory objects. In this sample, user stacks for all tasks are generated in SURI_STACK section that is the default setting.

(1) User stack for MasterDom_Task

MasterDom_Task is created statically by the system configuration file.

```
task[]{
    name            = ID_MASTERDOMTASK;
    entry_address   = MasterDom_Task();
    initial_start   = ON;
    stack_size      = 256;
    priority        = 1;
//  stack_section   = SURI_STACK;
    exinf           = 1;
    domain_num      = 1;
};
```

The user stack is generated in the SURI_STACK section when "stack_section" is omitted.

(2) User stack for AppDomA_Task and AppDomB_Task

AppDomA_Task and AppDomB_Task are generated by acre_tsk which is called by MasterDom_Task. The start address and size of user stack for each task is passed to acre_tsk.

User stack area for both AppDomA_Task and AppDomB_Task are generated in SURI_STACK section by using #pragma section directive in “master_dom.c”.

```
////////////////////////////////////////////////////////////////////////////
// Stack for AppDomA_Task and AppDomB_Task
////////////////////////////////////////////////////////////////////////////
#pragma section B SURI_STACK
static UW   s_ulDomA_Stk[DOM_A_STKSZ/sizeof(UW)];   // Stack area for AppDomA_Task
static UW   s_ulDomB_Stk[DOM_B_STKSZ/sizeof(UW)];   // Stack area for AppDomB_Task
```

9.1.4. Setting of Build Tools concerning Sections

9.1.4.1. Standard library generator

The section of standard library is made memory objects that can be accessed from all domains.

Table 9-4 Sections of Standard Library

Area	Section	Memory object
Code	PU_SH	memory_object[8]
Constant	CU_SH	
Literal	LU_SH	
Switch branch table	WU_SH, WU_SH_1, WU_SH_2	
Initialized data	DU_SH,DU_SH_1,DU_SH_2	
Uninitialized data	BU, BU_SH_1, BU_SH_2	memory_object[4]
Initialized data (RAM) (specify for linker)	RU_SH,RU_SH_1,RU_SH_2	

9.1.4.2. C/C++ compiler

The default section is same as Table 9-4. When other than this section is required, the section is changed by using "#pragma section" directive.

9.1.4.3. Linker

The “aligned_section” option is required for the following sections as shown in section 2.6.4 of “RI600PX Real-Time Operating System User's Manual: Coding”.

- The section specified for “memory_object[].start_address”
- The section specified for “task[].stack_section” (In this sample, this is not specified.)
- SURI_STACK

Therefore, in this sample, the “aligned_section” is specified for the start sections of memory objects and SURI_STACK.

9.1.5. Example of Dealing with Access Violation

This sample implements following example. For details, refer to sample code.

- AppDomA_Task : Raise task exception, and the task exception handling routine re-activates itself.
- AppDomB_Task : Do processing over again from normal point by using longjmp().

9.2. Sample programs using Firmware Integration Technology

This section describes the sample programs using FIT (Firmware Integration Technology) modules.

9.2.1. Summary

This sample program can run on the Renesas Starter Kits. It is possible to develop applications quickly and easily by using this sample. This sample program is that the minimum FIT modules are added to original sample program like existing "RX630_RI600PX". The basic operation is the same except that the hardware is initialized by FIT modules.

9.2.2. Structure of sample programs using FIT

Sample programs are provided as CS+ sample projects. The details of RI600PX sample project using FIT and used FIT modules are as follows.

- RI600PX Sample Projects using FIT

Sample Projects	Renesas Starter Kits
RX63N_RI600PX_FIT	Renesas Starter Kit+ for RX63N
RX64M_RI600PX_FIT	Renesas Starter Kit+ for RX64M
RX65N_RI600PX_FIT	Renesas Starter Kit+ for RX65N
RX71M_RI600PX_FIT	Renesas Starter Kit+ for RX71M

- Used FIT Modules

FIT Modules	FIT module name	Revision
Board Support Package (BSP)	r_bsp	Dec.01.15 Rev.3.10 Oct.01.16 Rev.3.40 (for RX65N)
Compare Match Timer (CMT)	r_cmt_rx	Jun.30.15 Rev.2.60 Oct.01.16 Rev.3.00 (for RX65N)

9.2.3. Directory structure of RI600PX sample projects using FIT

The directory structure of RI600PX sample project using FIT for RX63N is shown below.

```
RX63N_RI600PX_FIT   (*1)
├─appli
│  ├─include
│  └─source
│       ├─common
│       ├─dom_A
│       ├─dom_B
│       ├─kernel
│       └─master_dom
├─DefaultBuild
├─r_bsp
│  ├─board
│  │  ├─rskrx63n   (*1)
│  │  └─user
│  ├─doc
│  └─mcu
│       ├─all
│       └─rx63n   (*1)
│            └─register_access
├─r_cmt_rx
│  ├─doc
│  ├─ref
│  └─src
└─r_config
```

*1 In the other sample project, MCU name or RSK Board name in a folder name is replaced with respectively.

FIT modules (r_bsp and r_cmt_rx) are located in the root directory of a project.

The header files for FIT module settings are stored in r_config folder together.

9.2.4. Changes to RI600PX sample project using FIT

In RI600PX sample project using FIT, modules and sample program are modified partially. Basically, FIT modules have been changed to be available in a project with/without RTOS.

This chapter describes the major changes.

（1） RSK board initialization by FIT modules

Target files:

r_bsp\board\<RSK board name>\resetprg.c

r_bsp\board\<RSK board name>\dbsct.c

** RSK board name: rskrx63n, rskrx64m, rskrx65n, rskrx71m

Changes:

RSK Board is initialized by using board support package (r_bsp).

Therefore the following duplicate files are removed from the original sample program.

appli\source\reset\resetprg.c

appli\source\reset\dbsct.c appli

In the startup routine (PowerON_Reset_PC function in resetprg.c), the operation is switched by using BSP_CFG_RTOS_USED macro in case of RI600PX or OS-less.

The starting addresses of interrupt vector and fixed/exception vector are different in case of RI600PX or OS-less.

main function is called in user mode in case of OS-less, and vsta_knl is called in supervisor mode in case of RI600PX.

```
void PowerON_Reset_PC(void)
{
#if BSP_CFG_RTOS_USED == 0        /* Non-OS */
        set_intb((void *)__sectop("C$VECT"));
#ifdef __RXV2
        set_extb((void *)__sectop("EXCEPTVECT"));/* RXv2 command */
#endif/* __RXV2 */
#elif BSP_CFG_RTOS_USED == 1    /* FreeRTOS */
#elif BSP_CFG_RTOS_USED == 2    /* SEGGER embOS */
#elif BSP_CFG_RTOS_USED == 3    /* Micrium MicroC/OS */
#elif BSP_CFG_RTOS_USED == 4    /* Renesas RI600V4 & RI600PX */
        set_intb((void *)__sectop("INTERRUPT_VECTOR"));
#ifdef __RXV2
        set_extb((void *)__sectop("FIX_INTERRUPT_VECTOR"));/* RXv2 command */
#endif/* __RXV2 */
#endif/* BSP_CFG_RTOS_USED */

        (omission)

#if BSP_CFG_RTOS_USED == 0        /* Non-OS */
        nop();
        set_psw(PSW_init);
#if BSP_CFG_RUN_IN_USER_MODE==1
        chg_pmusr() ;
#endif
        main();
#if BSP_CFG_IO_LIB_ENABLE == 1
        _CLOSEALL();
#endif
        while(1)
        {
                /* Infinite loop. Put a breakpoint here if you want to catch an exit of main(). */
        }
#elif BSP_CFG_RTOS_USED == 1    /* FreeRTOS */
#elif BSP_CFG_RTOS_USED == 2    /* SEGGER embOS */
#elif BSP_CFG_RTOS_USED == 3    /* Micrium MicroC/OS */
#elif BSP_CFG_RTOS_USED == 4    /* Renesas RI600PX */
        /* Make sure to disable interrupt. */
        clrpsw_i();
        vsta_knl();             /* Start RI600PX and never return */
        brk();
#endif/* BSP_CFG_RTOS_USED */
}
```

The clock initialization routines are skipped by USE_SIM_DEBUG macro to prevent an endless waiting when debugging with RX Simulator.

For details, refer to "（6）RX simulator debug" in “9.2.5 Cautions of RI600PX sample project using FIT”.

The settings for section initialization in dbsct.c is switched by using BSP_CFG_RTOS_USED macro in case of RI600PX or OS-less.

（2） The aggregation of interrupt vector

Target files:

r_bsp\board\<RSK board name>\vecttbl.c

r_bsp\mcu\<MCU name>\mcu_interrupts.c

r_cmt_rx\src\r_cmt_rx.c

appli\source\kernel\sample.cfg

** RSK board name: rskrx63n, rskrx64m, rskrx65n, rskrx71m

** MCU name: rx63n, rx64m, rx65n, rx71m

Changes:

Interrupt vectors defined in FIT module are aggregated to RTOS system configuration file (sample.cfg).

The following descriptions are excluded by using BSP_CFG_RTOS_USED macro in vecttbl.c.

- "#pragma interrupt" line of interrupt handler function
- The definition of interrupt vector table (since "#pragma section C FIXEDVECT" line）

```
#if BSP_CFG_RTOS_USED == 0        /* Non-OS */
#pragma interrupt (non_maskable_isr)
#elif BSP_CFG_RTOS_USED == 1    /* FreeRTOS */
#elif BSP_CFG_RTOS_USED == 2    /* SEGGER embOS */
#elif BSP_CFG_RTOS_USED == 3    /* Micrium MicroC/OS */
#elif BSP_CFG_RTOS_USED == 4    /* Renesas RI600V4 & RI600PX */
#endif
void non_maskable_isr(void)
{
        :
}

#if BSP_CFG_RTOS_USED == 0        /* Non-OS */
#pragma section C FIXEDVECT

void * const Fixed_Vectors[] =
{
        :
        (void *) non_maskable_isr,          /* 0xfffffff8    NMI */
        (void *) PowerON_Reset_PC       /* 0xfffffffc    RESET */
};
#elif BSP_CFG_RTOS_USED == 1    /* FreeRTOS */
#elif BSP_CFG_RTOS_USED == 2    /* SEGGER embOS */
#elif BSP_CFG_RTOS_USED == 3    /* Micrium MicroC/OS */
#elif BSP_CFG_RTOS_USED == 4    /* Renesas RI600V4 & RI600PX */
#endif
```

"#pragma interrupt" lines of the following group-interrupt-handlers are excluded in mcu_interrupts.c for RX64M, RX65N and RX71M.

group_al0_handler_isr

group_al1_handler_isr

group_bl0_handler_isr

group_bl1_handler_isr

group_bl2_handler_isr (for RX65N only))

All of the above interrupt handler functions are registered in sample.cfg.

```
// BSP Interrupt Handler Definition (VECT_ICU_GROUPBL0)
interrupt_vector[110]{
        os_int                  = YES;
        entry_address       = group_bl0_handler_isr();
        pragma_switch       = E,ACC;
};
```

"#pragma interrupt" lines and static declarations on the following FIT timer API interrupt handlers are excluded in the r_cmt_rx.c.

cmt0_isr

cmt1_isr

cmt2_isr

cmt3_isr

The interrupt handler functions listed above are registered as follows in the sample.cfg file.

[RX71M, RX65N, RX64M]

Interrupt vector 128: cmt2_isr

Interrupt vector 129: cmt3_isr

[RX63N]

Interrupt vector 30: cmt2_isr

Interrupt vector 31: cmt3_isr

（3） Including RTOS header files in FIT

Target files:

r_bsp\board\<RSK board name>\r_bsp.h

** RSK board name: rskrx63n, rskrx64m, rskrx65n, rskrx71m

Changes:

Including the following header files in r_bsp.h

kernel.h

kernel_id.h

```
#if BSP_CFG_RTOS_USED == 0      /* Non-OS */
#elif BSP_CFG_RTOS_USED == 1    /* FreeRTOS */
#elif BSP_CFG_RTOS_USED == 2    /* SEGGER embOS */
#elif BSP_CFG_RTOS_USED == 3    /* Micrium MicroC/OS */
#elif BSP_CFG_RTOS_USED == 4    /* Renesas RI600V4 & RI600PX */
#include        "kernel.h"
#include        "kernel_id.h"
#endif/* BSP_CFG_RTOS_USED */
```

platform.h just need to be included in RTOS source, because r_bsp.h has been included in it.

（4） Excluding timer resource used by RI600PX in r_cmt_rx module

Target files:

r_cmt_rx\src\r_cmt_rx.c

r_config\r_cmt_rx_config.h

Changes:

In the timer API of this module, CMT channel used by RI600PX for system time update is excluded.

_RI_TRACE_TIMER macro (dummy) indicates CMT channel for tracing.

It is defined in r_cmt_rx_config.h to share FIT modules with RI600V4.

_RI_CLOCK_TIMER and _RI_TRACE_TIMER are set the same value.

```
bool R_CMT_Stop (uint32_t channel)
{
    /* Make sure valid channel number was input. */
    if (channel >= CMT_RX_NUM_CHANNELS)
    {
        /* Invalid channel number was used. */
        return false;
    }

#if BSP_CFG_RTOS_USED == 0      /* Non-OS */
#elif BSP_CFG_RTOS_USED == 1    /* FreeRTOS */
#elif BSP_CFG_RTOS_USED == 2    /* SEGGER embOS */
#elif BSP_CFG_RTOS_USED == 3    /* Micrium MicroC/OS */
#elif BSP_CFG_RTOS_USED == 4    /* Renesas RI600V4 & RI600PX */
    /* Exclude RTOS timers */
    if (channel == _RI_CLOCK_TIMER || channel == _RI_TRACE_TIMER)
    {
        return false;
    }
#endif/* BSP_CFG_RTOS_USED */

    /* Stop counter. */
    cmt_counter_stop(channel);
```

(5) Set the initial value of RTOS reserved channel in r_cmt_rx module.

Target files:

r_cmt_rx\src\r_cmt_rx.c

Changes:

CMT channel activity is stored in g_cmt_modes array.

CMT_RX_MODE_PERIODIC is set in the array as the initial value of the CMT channel used by RTOS.

** To prevent power down of a pair of CMT channels.

(6) LED control on RSK board

Target files:

appli\include\hw_control.h

appli\source\common\hw_control.c

Changes:

set_LED function is created to control LED's on/off on RSK board.

(7) Message output for debugging

Target files:

appli\include\rtos_sample_config.h

Changes:

In this sample, It is available to output any messages to debug-console by using printf function during debagging with RX simulator/E1 emulator.

printf function is not called directly from a task in a sample program.

It is used via DEBUG_print macro defined in rtos_sample_config.h.

DEBUG_print macro is controlled to enable or disable by following macros.

USE_DEBUG_MESSAGE (Enable output message to debug-console, if defined.)

（8） Handler calling at memory access exception

Target files:

r_bsp\board\<RSK board name>\vecttbl.c

appli\source\kernel\access_exc.c

** RSK board name: rskrx64m, rskrx65n, rskrx71m

Changes:

The access-exception handler excep_access_isr is registered at exception vector 21 in FIT.

However, the exception vector 21 is also used for task exception function in RI600PX.

So decided to call excep_access_isr function from access-exception handler _RI_sys_access_exception (access_exc.c) of RI600PX.

（9） Changes of sample program

Target files:

appli\include\rtos_sample_config.h

appli\source\common\common.c

appli\source\dom_A\dom_A.c

appli\source\dom_B\dom_B.c

appli\source\kernel\access_exc.c

appli\source\kernel\handler.c

appli\source\kernel\init_cmt.c

appli\source\kernel\sysdwn.c

appli\source\master_dom\master_dom.c

Changes:

- Including platform.h instead of kernel.h and kernel_id.h in all C sources.
- Add message outputs by using DEBUG_print macro to tasks and task-exception handlers.
- Light up LED1 at the end of task Mastrdom_task(mastr_dom.c).
- Goes off LED2 at the top of task AppDomA_Task(dom_A.c).
- Light up LED2 during task-exception handler AppDomA_TaskTex(dom_A.c).
- Goes off LED3 at the top of task AppDomB_Task(dom_B.c).
- Light up LED3 during task-exception handler AppDomB_TaskTex(dom_B.c).
- Dividing the number of calls of cyclic handler cyh1(handler.c) to switch LED0 on and off.
  LED_BLINK_DIV_RATIO macro defined in rtos_sample_config.h is used for dividing.
- Output error messages in _RI_sys_dwn__ (sysdwn.c) to debug-console.
- Locking CMT channel by using r_bsp API in _RI_init_cmt_knl function (init_cmt.c)

（10） Setting of Individual compile options

Target files:

r_bsp\board\<RSK board name>resetprg.c

** RSK board name: rskrx63n, rskrx64m, rskrx65n, rskrx71m

Changes:

- The individual compile options are set to place the stack area on a 4-byte boundary.
  The setting “-nostuff” is add to [Others]-[Other additional options].

（11） Changes of “CC-RX(Build tool)” option

The following options are changed in relation to Build.

Compile option:

Setting has changed from "C89" to "C99" in [Source]-[C source language]

Setting has changed from "2" to "0" in [Optimization]-[Optimization level]

Link option:

Setting has changed to "Yes" in [List]-[Output symbol information]

Library generate option:

Setting has changed from “C89” to "C99" in [Standard library]-[Library structure]

9.2.5. Cautions of RI600PX sample project using FIT

This chapter describes the cautions on use of RI600PX sample project using FIT.

（1） Section names used in RI600PX sample

Most of section names are application-dependent.

Please do not forget the following changes when adding the section name or changing it.

- Specifying sections arrangement in a source file. (#pragma section)
- Section setting of the link option
  1. Section start address (order)
  2. Section alignment
  3. ROM to RAM mapping section
- Initialization section settings in dbsct.c
- Memory object settings in system configuration file (sample.cfg).

（2） Section arrangement

Section arrangement is closely related with memory objects in sample.cfg.

The address range is specified by using two section names in a memory object.

The range is across multiple sections.

Be careful enough when changing section order.

（3） CMT channel limitation

CMT0 is used to update the system time by default in RI600PX.

FIT timer API dynamically uses the other CMT channels greater than 0.

（4） Power supply from emulator

Default setting is power supply from the emulator (USB) in this sample.

USB power may become insufficient capacity by increased current consumption during development.

Please use an external power supply on RX64M or later RSK Board.

[Connection with target board]-[Power target from the emulator.(MAX 200mA)]

in "Connect Settings" tab of "RX E1(JTAG) (Debug Tool)" property

（5） FIT module update

The FIT modules r_bsp and r_cmt_rx attached to this sample are customized for RTOS.

Therefore, do not overwrite them with the latest version of the corresponding FIT module.

（6） Debugging on RX simulator

The clock initialization routine for RX71M, RX65N, RX64M, RX113 assumes executing on emulator.

When debugging on RX simulator, it enters an infinite wait loop by register readout.

To avoid this problem, build after uncommenting the following definition in resetprg.c,

or specify it in the compile option [Source] - [Macro Definition].

```
//#define USE_SIM_DEBUG
```

The clock initialization routine is skipped by the USE_SIM_DEBUG macro definition.

(7) Message output function during debugging

DEBUG_print macro function enables debug log and error message output.

However, DEBUG_print is disabled by default in this sample.

This is because debug console plug-in is disabled by default on CS+.

DEBUG_print can be enabled with the following steps.

1.Enable debug console plug-in

Select CS+ menu [Tool]-[Plug-in Setting…] to open "Plug-in Manager" dialog box.

Check the box [Debugging Console Plug-in] in "Additional Function" tab.

2.Build with USE_DEBUG_MESSAGE macro definition

Build after uncommenting the following definition in rtos_sample_config.h,

or specify USE_DEBUG_MESSAGE macro when compiling.

```
//#define USE_DEBUG_MESSAGE
```

(8) Task stack size

This sample assumes the use of the standard function printf.

Standard function printf consumes more stack. (More than 400 bytes)

Therefore, task stack size will ensure more than expected. (over 400bytes)

(9) Limitation of R_CMT_Control

FIT timer API R_CMT_Control function returns an error when specifying the same value of _RI_CLOCK_TIMER(0) as CMT channel.

This is because RTOS reserved channel is excluded internally.

（10） FIT API restrictions

FIT provides various API.

However, there are the following restrictions on RI600PX.

Table 9-5 Usable or not about r_bsp API on RI600PX

r_bsp API name	Before starting kernel	After starting kernel	
	Startup routine （PowerON_Reset_PC）	Task (User mode)	Non-task (Supervisor mode)
R_BSP_GetVersion	✓	✓	✓
R_BSP_InterruptsDisable	✓	(No effect)	✓
R_BSP_InterruptsEnable	✓	(No effect)	✓
R_BSP_CpuInterruptLevelRead	✓	✓	✓
R_BSP_CpuInterruptLevelWrite	✓	(No effect) ***1**	(Not recommend) ***1**
R_BSP_RegisterProtectEnable	✓	✓	✓
R_BSP_RegisterProtectDisable	✓	✓	✓
R_BSP_SoftwareLock	✓	✓	✓
R_BSP_SoftwareUnlock	✓	✓	✓
R_BSP_HardwareLock	✓	✓	✓
R_BSP_HardwareUnlock	✓	✓	✓
R_BSP_InterruptWrite	✓	✓	✓
R_BSP_InterruptRead	✓	✓	✓
R_BSP_InterruptControl	✓	✓	✓
R_BSP_SoftwareDelay	✓	✓	✓

*1 Use service-calls of RI600PX instead, such as chg_ims or ichg_ims.

Table 9-6 Usable or not about r_cmt_rx API on RI600PX

r_cmt_rx API name	Before starting kernel	After starting kernel	
	Startup routine (PowerON_Reset_PC)	Task (User mode)	Non-task (Supervisor mode)
R_CMT_CreatePeriodic	✓	✓ *1	✓ *1
R_CMT_CreateOneShot	✓	✓ *1	✓ *1
R_CMT_Control	✓	✓	✓
R_CMT_Stop	✓	✓	✓
R_CMT_GetVersion	✓	✔	✔

*1 Use cyclic handler or alarm handler of RI600PX instead.

（11） Undefined interrupt handler

When an undefined interrupt occurs in this sample, the system-down-routine (_RI_sys_dwn _ _) is called from a handler inside the kernel. The undefined interrupt handler (undefined_interrupt_source_isr) in r_bsp module is not called. For this reason, the callback function that is registered as undefined-interrupt-handler is never called.

Please describe the processing of undefined interrupt in the system-down-routine.

（12） Setting of the base clock cycle

It is recommended to set the cycle of base clock timer interrupt to 1ms on RI600PX.

In RI600PX sample project using FIT, set the clock frequency of PCLKB set by the following file to clock.timer_clock in sample.cfg.

r_config\r_bsp_config.h

This will cause system time to be in milliseconds.

```
// Setting example for RX64M in sample.cfg
clock{
      template         = rx630.tpl;
      timer            = CMT0;
      timer_clock      = 60MHz; // PCLKB frequency
      IPL              = 13;
};
```

As a confirmation method, set 125 as the cycle of cyclic handler, and define the macro LED_BLINK_DIV_RATIO as 2 in the following file.

appli\include\rtos_sample_config.h

This causes LED0 to blink at approximately 1 second intervals.

9.2.6. How to add a new FIT module

This chapter describes the sequences to add a new FIT module.

（1） Get the source ZIP file of FIT module.

（2） Unzip the source files of FIT module to the root directory of CS+ project.

（3） Create a new configuration file of FIT module in r_config folder.

There is a reference file usually. Copy the file and rename it.

Original: ref\<FIT module name>_config_reference.h

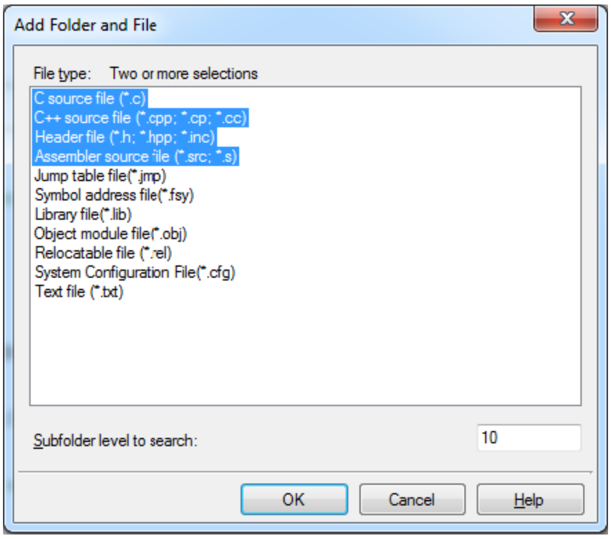
Copied : r_config\<FIT module name>_config.h

（4） By using Windows explorer, Execute Drag&Drop to add the top directory of FIT module to the “File” of CS+ Project tree.

（5） In “Add folder and file” dialog, Set the followings and click OK button.

- Select the file types to add to the project.
- Set a value over maximum number of layers to “Detecting the number of sub-folder layers”.

Table 9-1 "Add Folder and File" dialog



Add Folder and File

File type: Two or more selections

- C source file (*.c)
- C++ source file (*.cpp; *.cp; *.cc)
- Header file (*.h; *.hpp; *.inc)
- Assembler source file (*.src; *.s)
- Jump table file(*.jmp)
- Symbol address file(*.fsy)
- Library file(*.lib)
- Object module file(*.obj)
- Relocatable file (*.rel)
- System Configuration File(*.cfg)
- Text file (*.txt)

Subfolder level to search: 10

（6） Confirm [Source]-[Additional include path] settings in "Compile option" tab of "CC-RX(Build tool)"

By above method, CS+ adds all relative paths in FIT module to [Additional include path].

（7） Register all interrupt handlers inside FIT modules in the system configuration file.

When a new FIT module is added, please aggregate all interrupt handlers to cfg file.

Please change sources by following these steps.

1. Exclude "#pragma interrupt" lines in a source file of FIT module.

   Notes: Including platform.h or kernel_id.h is required in the source. Please confirm it.

2. Remove "static" declaration of a handler function.
3. Register interrupt handlers in sample.cfg.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Oct 1,2018	-	New Publication

Website and Support

- Renesas Electronics Website
  http://www.renesas.com/

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

All trademarks and registered trademarks are the property of their respective owners.

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.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

RENESAS

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

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

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 7.1