



V.1.00 Release 01

February 1, 2010

RI600/4 V.1.00 Release 01

Release Notes

Contents

Section 1	Provided Form	1	Section 7	Notes	7
Section 2	Required System	1	7.1	Notes when RX610 group microcontroller is used.....	7
Section 3	Tool News	1	7.2	Notes concerning ID code protecting.....	7
Section 4	Installation and Uninstallation	2	7.3	Notes concerning address 0.....	7
Section 5	How to build kernel source code 2		Section 8	Correction of user's manual.....	8
Section 6	About stack size.....	3	8.1	"Table 5.34" (163 Page).....	8
6.1	Stack size of system clock interrupt handler.....	3	8.2	"11.1 Overview" (230 Page)	8
6.2	Stack size used by service call	3	Section 9	History.....	9
6.3	When the kernel library is built.....	6	9.1	V.1.00 Release 00 (October 2009)	9
			9.2	V.1.00 Release 01 (February 2010)	10
			9.2.1	Kernel.....	10

WARNING

Take a countermeasure such as a fail safe on the system against personal injury, fire hazard, or other damages of your system due to the operation of this product. If the countermeasure cannot be taken, do not use this product.

Microsoft, Windows and Windows Vista are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

IBM PC is a registered trademark of International Business Machines Corporation.

All other company and product names are registered trademarks or trademarks of their respective companies.

Section 1 Provided Form

The RI600/4 is provided as shown in table 1.1.

Table 1.1 RI600/4 Provided Form List

Type	Host Computer	Medium	Number of Volume
R0R5RX00TRW011 *1 License for evaluation; can be installed only to one host computer.	Windows(R)	CD-R	1
R0R5RX00TRW015 *1 License for evaluation, can be installed only to 5 host computers.	Windows(R)	CD-R	1
R0R5RX00TRW01A *1 License for evaluation; can be installed only to 10 host computers.	Windows(R)	CD-R	1
R0R5RX00TRW01K *1 Mass-production license; can be embedded up to a total of 1,000 productions of product model(s) with the source code not disclosed.	Windows(R)	CD-R	1
R0R5RX00TRW01U *1 Mass-production license; can be embedded up to unlimited productions of product model(s) with the source code not disclosed.	Windows(R)	CD-R	1
R0R5RX00TRW01Z Mass-production license; can be embedded up to unlimited productions of product model(s) with the source code disclosed.	Windows(R)	CD-R	1

*1 These contents of offer are the same.

Section 2 Required System

- Host computer : IBM PC or compatible machine, which runs following, host OS.
Microsoft(R) Windows(R) 2000, Windows(R) XP, or Windows Vista(R)
- Memory capacity : 128 Mbytes at least; 256 Mbytes or more recommended.
- I/O device: CD-ROM drive
- Other requirement: A pointing device such as a mouse
- C/C++ Compiler package for RX Family.

Section 3 Tool News

Tool News provides information on our products so that customers can use the products more efficiently.

The Tool News pages are available on our Web site.

<http://tool-support.renesas.com/eng/toolnews/index.htm>

Get the latest information about new products, upgraded versions and precautions from Tool News, and take advantage of it in your development projects. Since the release notes do not include information issued after the release of the product, be sure to check the latest issue of Tool News.

Section 4 Installation and Uninstallation

The administrator authority of the computer that uses it is required for installing this product.

Start 'setup.exe' in the root directory of the CD-R, and then follow the instructions displayed on the screen. When installation, close all applications.

The following components are installed by setup.exe.

- Kernel library file, standard header file, and system definition file.
- Command line configurator (cfg600)
- Table generation utility (mkritbl)
- GUI configurator
- Kernel source code (It is attached only to R0R5RX00TRW01Z).

To uninstall, select [RI600/4 V.1.00 Release 01] from [Add/Remove Program] of the Control-Pannel.

Section 5 How to build kernel source code¹

The kernel source code is stored in "< installation directory > \src600". It moves to the source code installation directory to build the kernel, and "Nmake²" is executed.

The environment variable settings are needed by compiler when building the kernel.

Example:

```
C:\RENESAS\src600>nmake
```

After the building the kernel, the kernel library is generated to the following directories.

Kernel library name	Contents
product\big\debug\ri600big.lib	Big endian library with debugging information
product\big\release\ri600big.lib	Big endian library without debugging information
product\little\debug\ri600lit.lib	Little endian library with debugging information
product\little\release\ri600lit.lib	Little endian library without debugging information

[Note]

Please copy "src600" directory to the writable directory if you don't have the write-access permission to the product installation directory. After the build, copy the generated library to the "lib600" directory under the product installation directory by the user who has write-access permission to the product installation directory.

¹ The source code is only attached to a mass production contract version (R0R5RX00TRW01Z).

² "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.

Section 6 About stack size

6.1 Stack size of system clock interrupt handler

The value of $\epsilon 1$, $\epsilon 2$, and $\epsilon 3$ described in the manual paragraph 12.4 are as follows.

$$\epsilon 1 = 104$$

$$\epsilon 2 = 104$$

$$\epsilon 3 = 196$$

6.2 Stack size used by service call

In the service call, the stack is used as follows.

- (1) Called from the task context

The stack in the task context execution is a user stack. The service call is using following.

- (a) User stack (Former call stack)

- (b) System stack

- (2) Called from the non-task context

The stack in the non-task context execution is a system stack. The service call is using following.

- (a) System stack (Former call stack)

The use size of former stack ((a), (c)) which the service call uses is displayed by Call Walker. However, when the service call is called from the application of the assembly language, it is necessary to add the value of (a) and (c) by hand because it is not displayed in Call Walker.

Moreover, to calculate the system stack size described in manual 12.4, the size of (b) and (c) is needed. (Paragraph 12.4 has described as α .) The size of (a), (b) and (c) of each service call is shown as follows.

	Service call	The use size of User stack(a)	The use size of System stack ((b),(c))	Note
Task Management Function				
1	act_tsk	0	24	
2	iact_tsk	0	24	
3	can_act	0	24	
4	ican_act	0	24	
5	sta_tsk	0	24	
6	ista_tsk	0	24	
7	ext_tsk	0	64	ext_tsk is called at the return from the task beginning function.
8	ter_tsk	0	132	
9	chg_pri	0	36	
10	ichg_pri	0	52	
11	get_pri	0	24	
12	iget_pri	0	24	
13	ref_tsk	0	28	
14	iref_tsk	0	28	
15	ref_tst	0	24	
16	iref_tst	0	24	
Task Dependent Synchronization Function				
17	slp_tsk	0	24	
18	tslp_tsk	0	24	
19	wup_tsk	0	40	
20	iwup_tsk	0	52	

	Service call	The use size of User stack(a)	The use size of System stack ((b),(c))	Note
21	can_wup	0	24	
22	ican_wup	0	24	
23	rel_wai	0	116	
24	irel_wai	0	132	
25	sus_tsk	0	24	
26	isus_tsk	0	24	
27	rsm_tsk	0	24	
28	irms_tsk	0	24	
29	frsm_tsk	0	24	
30	ifrsn_tsk	0	24	
31	dly_tsk	0	24	
Semaphore				
32	sig_sem	0	44	
33	isig_sem	0	60	
34	wai_sem	0	32	
35	pol_sem	0	24	
36	ipol_sem	0	24	
37	twai_sem	0	36	
38	ref_sem	0	24	
39	iref_sem	0	24	
Eventflag				
40	set_flg	0	48	
41	iset_flg	0	64	
42	clr_flg	0	24	
43	iclr_flg	0	24	
44	wai_flg	0	44	
45	pol_flg	0	24	
46	ipol_flg	0	24	
47	twai_flg	0	48	
48	ref_flg	0	24	
49	iref_flg	0	24	
Data Queue				
50	snd_dtq	0	36	
51	psnd_dtq	0	32	
52	ipsnd_dtq	0	52	
53	tsnd_dtq	0	40	
54	fsnd_dtq	0	32	
55	ifsnd_dtq	0	52	
56	rcv_dtq	0	32	
57	prcv_dtq	0	32	
58	iprcv_dtq	0	52	
59	trcv_dtq	0	32	
60	ref_dtq	0	24	
61	iref_dtq	0	24	
Mailbox				
62	snd_mbx	0	40	
63	isnd_mbx	0	56	
64	rcv_mbx	0	32	
65	prcv_mbx	0	28	
66	iprcv_mbx	0	28	
67	trcv_mbx	0	36	
68	ref_mbx	0	24	
69	iref_mbx	0	24	
Mutex				

	Service call	The use size of User stack(a)	The use size of System stack ((b),(c))	Note
70	loc_mtx	0	40	
71	ploc_mtx	0	24	
72	tlloc_mtx	0	44	
73	unl_mtx	0	56	
74	ref_mtx	0	24	
Message Buffer				
75	snd_mbf	0	44	
76	psnd_mbf	0	44	
77	ipsnd_mbf	0	60	
78	tsnd_mbf	0	44	
79	rcv_mbf	0	56	
80	prcv_mbf	0	56	
81	trcv_mbf	0	56	
82	ref_mbf	0	24	
83	iref_mbf	0	24	
Fixed-sized Memory Pool				
84	get_mpf	0	48	
85	pget_mpf	0	36	
86	ipget_mpf	0	36	
87	tget_mpf	0	48	
88	rel_mpf	20	36	
89	irel_mpf	0	52	
90	ref_mpf	0	24	
91	iref_mpf	0	24	
Variable Size Memory Pool				
92	get_mpl	36	96	
93	pget_mpl	0	112	
94	ipget_mpl	0	112	
95	tget_mpl	36	96	
96	rel_mpl	0	108	
97	ref_mpl	0	24	
98	iref_mpl	0	24	
Time Management Function				
99	set_tim	0	24	
100	iset_tim	0	24	
101	get_tim	0	24	
102	iget_tim	0	24	
Cyclic Handler				
103	sta_cyc	0	24	
104	ista_cyc	0	24	
105	stp_cyc	0	24	
106	istp_cyc	0	24	
107	ref_cyc	0	24	
108	iref_cyc	0	24	
Alarm Handler				
109	sta_alm	0	24	
110	ista_alm	0	24	
111	stp_alm	0	24	
112	istp_alm	0	24	
113	ref_alm	0	24	
114	iref_alm	0	24	
System State Management Function				
115	rot_rdq	0	24	
116	irotd_rdq	0	24	

	Service call	The use size of User stack(a)	The use size of System stack ((b),(c))	Note
117	get_tid	0	24	
118	iget_tid	0	24	
119	loc_cpu	0	16	
120	iloc_cpu	0	16	
121	unl_cpu	0	24	
122	iunl_cpu	0	24	
123	dis_dsp	0	16	
124	ena_dsp	0	24	
125	sns_ctx	0	24	
126	sns_loc	0	24	
127	sns_dsp	0	24	
128	sns_dpn	0	24	
129	vsta_knl	0	52	After the system stack pointer is initialized, it uses it.
130	ivsta_knl	0	52	
131	vsys_dwn	0	16	
132	ivsys_dwn	0	16	
Interrupt Management Function				
133	chg_ims	0	28	
134	ichg_ims	0	16	
135	get_ims	4	0	The Call Walker displays the stack size used by get_ims and iget_ims when these functions are called from assembly program.
136	iget_ims	0	4	
137	ret_int	0	32	
System Configuration Management Function				
138	ref_ver	0	24	
139	iref_ver	0	24	
Object Reset Function				
140	vrst_dtq	0	48	
141	vrst_mbx	0	24	
142	vrst_mbf	0	48	
143	vrst_mpf	0	48	
144	vrst_mpl	0	68	

6.3 When the kernel library is built

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

Section 7 Notes

7.1 Notes when RX610 group microcontroller is used

In the RX610 group, the processor interrupt priority level is limited to 0-7. Even if the value of 8-15 is set when RX610 is used, it doesn't detect it as an error though it is possible to use for 0-15 as a processor interrupt priority level in the GUI configurator, cfg600, and the chg_ims•ichg_ims service call. Please confirm the value of 8-15 is not set on the user side enough.

7.2 Notes concerning ID code protecting

In RI600/4, when the handler that uses a fixed vector is not defined, the handler address of default is set. The handler address of default is similarly set as for the ID code area. Please correspond as follows to set the ID code area to the arbitrary value.

1. The arbitrary symbol of assembly language is defined as an interrupt handler of a fixed vector.
2. The given value is set to the defined symbol.

Example:

To set 0xFFFFFFFF into the address 0xFFFFFA0,

- Description of configuration file
interrupt_fvector[8]{
 entry_address = ID_CODE1;
};
- Linkage editor option
-define=ID_CODE1=0xFFFFFFFF

7.3 Notes concerning address 0

Please do not put the following in address 0.

1. Variable-sized memory pool section
2. Fixed-sized memory pool section
3. Message sent to the mailbox

Section 8 Correction of user's manual

The user's manual (The document number: REJ10J2052-0100) is corrected as follows.

8.1 "Table 5.34" (163 Page)

Correct:

Table 5.34 Constants and Macros

Classification	Macro	Definition	Where	Description
...
Kernel configuration	VTMAX_AREASIZE	0x10000000	kernel.h	Maximum size of various areas
...
Error codes	E_NOSPT	-9	itron.h	Unsupported function
...

Incorrect:

Table 5.34 Constants and Macros

Classification	Macro	Definition	Where	Description
...
Kernel configuration	VTMAX_AREASIZE	0x20000000	kernel.h	Maximum size of various areas
...
Error codes	E_NOSPT	-11	itron.h	Unsupported function
...

8.2 "11.1 Overview" (230 Page)

Correct:

Table 11.1 Functions in the Sample Program

Function	Type	ID number	Task priority	Description
task1()	Task	1	1	Outputs "task1 running"
task2()	Task	2	2	Outputs "task2 running"
cyh1()	Cyclic handler	1	-	Wakes up task1

Incorrect:

Table 11.1 Functions in the Sample Program

Function	Type	ID number	Task priority	Description
main()	Task	1	1	Activates task1 and task2 and cyh1
task1()	Task	2	2	Outputs "task1 running"
task2()	Task	3	3	Outputs "task2 running"
cyh1()	Cyclic handler	1	-	Wakes up task1

Correct:

The content of processing is described below.

- The task1 operates in order to following.

Incorrect:

The content of processing is described below.

- The main task activates task1, task2, and cyh1, and then terminates itself.
- The task1 operates in order to following.

Section 9 History

9.1 V.1.00 Release 00 (October 2009)

The first release.

9.2 V.1.00 Release 01 (February 2010)

9.2.1 Kernel

(1) Problem Fixed Concerning Service Calls With a Time Waiting

The following problem is fixed.

The following problems might occur when a task enters to the WAITING state by either of the service call with a time waiting (*), and the WAITING state of the task is released by the methods other than the passing of time.

- (a) Even if the specified time passes, the task that calls the service call with a time waiting is not released from the WAITING state.
- (b) Time when the WAITING state of the task that calls the service call with a time waiting is released is delayed.
- (c) The operation of unspecified task becomes illegal.

* Service calls with a time waiting:

tslp_tsk, dly_tsk, twai_sem, twai_flg, tsnd_dtq, trcv_dtq, trcv_mbx, tloc_mtx, tsnd_mbf, trcv_mbf, tget_mpf, tget_mpl

(2) Problem Fixed Concerning Cyclic Handler and Alarm Handler [1]

The following problem is fixed.

When a cyclic handler or alarm handler updates the R5 register, the kernel might access (read/write) to unspecified address.

(3) Problem Fixed Concerning Cyclic Handler and Alarm Handler [2]

The following problem is fixed.

When a cyclic handler or alarm handler updates the R8 - R13 registers, the R8 - R13 registers in the program which is interrupted by the system-clock interrupt handler are destroyed.

(4) Problem Fixed Concerning System Clock Interrupt Handler

The following problem is fixed.

When application uses DSP instructions, the ACC register in the program which is interrupted by the system-clock interrupt handler might be destroyed.

(5) Problem Fixed Concerning irel_mpf

The following problem is fixed.

The R6 register might not be guaranteed before and after the call of iref_mpf.

(6) Change in Stack Size

The contents of "6.1 Stack size of system clock interrupt handler" is changed as follows.

Before:

The value of ϵ_1 described in the manual paragraph 12.4, ϵ_2 , and ϵ_3 is as follows.

$$\epsilon_1 = 72$$

$$\epsilon_2 = 72$$

$$\epsilon_3 = 144$$

After:

The value of ϵ_1 , ϵ_2 , and ϵ_3 described in the manual paragraph 12.4 are as follows.

$$\epsilon_1 = 104$$

$$\epsilon_2 = 104$$

$$\epsilon_3 = 196$$

And the table in "6.2 Stack size used by service call" is changed.

The changed service calls are shown as follows.

- The size of System stack used by `ter_tsk` : 112 → 132
- The size of System stack used by `rel_wai` : 96 → 116
- The size of System stack used by `irel_wai` : 112 → 132 *
- The size of System stack used by `irel_mpf` : 48 → 52 *
- The size of System stack used by `get_mpl` : 76 → 96
- The size of System stack used by `pget_mpl` : 92 → 112
- The size of System stack used by `ipget_mpl` : 92 → 112 *
- The size of System stack used by `tget_mpl` : 76 → 96
- The size of System stack used by `rel_mpl` : 88 → 108

The stack size displayed by "Call Walker" is also changed about the service calls shown in "*".

(7) Addition of Notes

"7.3 Notes concerning address 0" of this book is added.

(8) Others

Following values is changed.

Item	Before	After
The <code>TKERNEL_PRVER</code> macro, and <code>T_RVER.prver</code> , which is returned by the <code>ref_ver</code>	0x0100	0x0101

RI600/4 Release Notes

Revision Date : February 1, 2010

Copyright (C) 2009(2010) Renesas Technology Corp. and Renesas Solutions Corp. All rights reserved

<http://www.renesas.com/>