To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

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

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

Real-time OS RI600/4 for RX600 Series

Application Transition Guide (HI1000/4 → RI600/4)

This document gives useful information about the transition of C-language applications from HI1000/4 to RI600/4, especially regarding the specifications changed from HI1000/4 to RI600/4.

Contents

1. Overview

Figure 1 gives an overview of application asset transition from HI1000/4 to RI600/4.

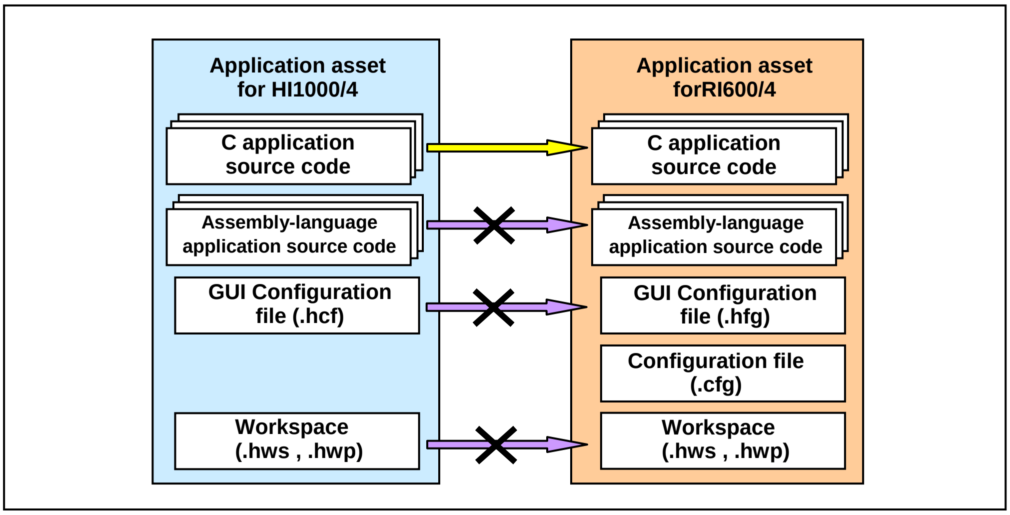

Figure 1 Overview of Application Asset Transition

(1) C Application Source Code

Some parts of the C source code that are related to the compiler differences should be modified in some cases. In addition, the code should be modified as necessary in accordance with the differences between OS specifications described in the following sections.

(2) Assembly-Language Application Source Code

The assembly languages of the H8SX, H8S family and RX family are not compatible; a new assembly-language code should be created for the RX family.

(3) GUI Configurator File (.hcf)

RI600/4 supports the GUI configurator, but the .hcf files for HI1000/4 and RI600/4 are not compatible.
See section 14.1, Difference between Configurators.

(4) Workspace (.hws, .hwp)

Due to the specifications of High-performance Embedded Workshop, the workspace created for the H8SX, H8S family cannot be used for the RX family; a new workspace should be created for the RX family.

2. Parameter Data Type and Size

Table 1 shows the differences in each parameter data type and size between HI1000/4 and RI600/4.
When the application uses the data types shaded in the table, check and change the code where such data types are used.
Note especially that the FLGPTN type (eventflag bit pattern) has been changed from 16 bits to 32 bits.

Table 1 Differences of Basic Data Types

	Type	HI1000/4	Remarks	RI600/4	Difference from HI1000/4
itron.h	B	signed char B;		signed char B;	
	H	signed short H;		signed short H;	
	W	signed long W;		signed long W;	
	D	(Not defined)	Not supported by compiler	signed long long D;	Added
	UB	unsigned char UB;		unsigned char UB;	
	UH	unsigned short UH;		unsigned short UH;	
	UW	unsigned long UW;		unsigned long UW;	
	UD	(Not defined)	Not supported by compiler	unsigned long long UD;	Added
	VB	signed char VB;		signed char VB;	
	VH	signed short VH;		signed short VH;	
	VW	signed long VW;		signed long VW;	
	VD	(Not defined)	Not supported by compiler	signed long long VD;	Added
	VP	void *VP;		void *VP;	
	FP	void (*FP)(void);		void (*FP)(void);	
	INT	H INT;		W INT;	16 bits → 32 bits
	UINT	UH UINT;		UW UINT;	16 bits → 32 bits
	BOOL	H BOOL;		W BOOL;	16 bits → 32 bits
	FN	H FN;	No function	(Not defined)	Deleted
	ER	H ER;		W ER;	16 bits → 32 bits
	ID	H ID;		H ID;	
	ATR	UH ATR;		UH ATR;	
	STAT	UH STAT;		UH STAT;	
	MODE	UH MODE;		UH MODE;	
	PRI	H PRI;		H PRI;	
	SIZE	UW SIZE;		UW SIZE;	
	TMO	W TMO;		W TMO;	
	RELTIM	UW RELTIM;		UW RELTIM;	
	SYSTIM	typedef struct { UH utime; UW ltime; } SYSTIM;		typedef struct { UH utime; UW ltime; } SYSTIM;	
	VP_INT	W VP_INT;		W VP_INT;	
	ER_BOOL	H ER_BOOL;	No function	(Not defined)	Deleted
	ER_ID	H ER_ID;	No function	(Not defined)	Deleted
	ER_UINT	H ER_UINT;		W ER_UINT;	16 bits → 32 bits
kernel.h	FLGPTN	UH FLGPTN;		UW FLGPTN	16 bits → 32 bits
	INHNO	UH INHNO;	No function	(Not defined)	Deleted
	EXCNO	UH EXCNO;	No function	(Not defined)	Deleted
	IMASK	UH IMASK;		UH IMASK;	

3. Functions Deleted or Diminished in RI600/4

3.1 Deletion of Shared Stack Function

RI600/4 does not support the shared stack function.
It has no substitute for the function.

3.2 Deletion of Service Call Trace Function and Related Service Calls

RI600/4 does not support the service call trace function.
It supports neither service call "**ivbgn_int**", which gets trace information at the start of the interrupt handler, nor "**ivend_int**", which gets trace information at the end of interrupt handler.
When the application uses these service calls, delete them.

3.3 Deletion of CPU Exception Handlers

RI600/4 does not support CPU exception handlers.
The CPU exceptions are treated as non-kernel interrupts.

3.4 Deletion of Initialization Routine

In HI1000/4, the initialization routine specified by the configurator can be executed before execution of the first task after initiation of the kernel, but RI600/4 does not support this function.
To get the same effect as the initialization routine, use the following procedure.

(1) Create a task for executing the necessary initialization processing. In configuration, specify "start after creation" only for that task, and specify only "create" for the other tasks.
(2) After the initialization processing by the task created in (1), start other tasks through the "**act_tsk**" or "**sta_tsk**" service call.

3.5 Not Making System Idling Routine Open

In HI1000/4, the KERNEL_H_SYSTEM_IDLE( ) function (sample program xxxx_idle.c provided in HI1000/4) is called when the kernel enters idle state, but RI600/4 does not support this function.
When some processing should be done in a non-load state, implement the processing as the lowest-priority task.

4. Restrictions on Object ID

HI1000/4 allows the existence of unused (non-created) ID numbers within the range between 1 to CFG_MAXxxxID (maximum ID) for each object type. If a service call is issued with a non-created ID number specified, the E_NOEXS error will be returned.
In RI600/4, the ID number specifications have been changed; there is no need to specify the "maximum ID for each object" during configuration but serial numbers starting from 1 should be assigned to objects in order without any unassigned number. Accordingly, no E_NOEXS error will occur and if an unassigned ID is found, the cfg600 will report an error.

5. Variable-Sized Memory Pool

5.1 Pool Size

The method for managing the variable-sized memory pool has been completely modified from HI1000/4 to RI600/4. Specifically, the RI600/4 only allows limited variations of the allocatable memory block size to reduce fragmentation of free space in comparison with HI1000/4.
However, the memory allocation efficiency may be degraded compared to that in HI1000/4 under some conditions. In this case, increase the memory pool size.

5.2 Maximum Size of Allocatable Block

In HI1000/4, the maximum size of the allocatable block is the pool size – 16 (bytes).
In RI600/4, the maximum block size should be defined during configuration. The memory block size variations described above are determined based on this maximum block size.
For details, refer to section 8.4.12, Variable-sized Memory Pool Definition (variable_memorypool[ ]), in the RI600/4 User's Manual.

6. Interrupt Priority Level and Interrupt Mask

As HI1000/4 and RI600/4 are targeted for different CPU cores, the interrupt priority level and mask values listed below have different meanings between HI1000/4 and RI600/4.
Note, however, that a value of 0 indicates the interrupt mask-canceled state in both HI1000/4 and RI600/4.

- Kernel interrupt mask level and timer interrupt priority level specified during configuration
- Interrupt masks processed by chg_ims, ichg_ims, get_ims, and iget_ims

7. E_CTX Error Caused by Interrupt Mask Value

In HI1000/4, service calls are prohibited while the interrupt mask level is higher than the kernel interrupt mask level.
In RI600/4, the E_CTX error will be returned if a service call is issued in such a case. Note that the E_CTX error will be returned also from the following service calls although the data type of the return value is BOOL (= signed long).
sns_ctx, sns_loc, sns_dsp, sns_dpn

8. Multiplex Interrupts

In HI1000/4, multiplex interrupts are enabled for all interrupt handlers according to the CPU specifications.
In the RX600 specifications, multiplex interrupts can be disabled.
For all interrupt handlers, multiplex interrupts are disabled by default in RI600/4. Multiplex interrupts can be enabled during configuration. For details, refer to section 8.4.15, Relocatable Vector Definition (interrupt_vector[]), in the RI600/4 User's Manual.

9. Timer Drivers

In HI1000/4, timer drivers should be created by the user and a sample timer driver for representative microcomputers is provided.
In RI600/4, as the configurator (cfg600) generates the driver for the on-chip timer (CMT) in the microcomputer, the user does not need to create a timer driver in most cases. Of course, the user can create a driver for an external timer as needed.
Note, however, that the timer interrupt handler is implemented in the kernel library and user-specified processing cannot be added to the timer interrupt handler in RI600/4 although it is allowed in HI1000/4. When such additional processing is necessary, replace the timer interrupt handler with a cyclic handler.
The timer initialization processing is automatically done in vsta_knl in HI1000/4, but the application should initialize the timer before starting the kernel (vsta_knl) in RI600/4. Specifically, include "ri_cmt.h" output from cfg600 and call _RI_init_cmt().

10. System-Down Routine

(1) Parameters

The amount of information passed to the system-down routine through parameters are almost the same between RI600/4 and HI1000/4, except that the "task ID when an undefined interrupt occurs" is not passed in RI600/4, but the specifications of the parameters differ between them. For details, refer to section 10, Information during System Down, in the HI1000/4 User's Manual and section 6.6, System-Down Routine, in the RI600/4 User's Manual, to compare the specifications.
Note that the -U option should be specified in cfg600 to receive the vector number of an undefined interrupt as a parameter in RI600/4.
For details, refer to section 8.5.4, Command Options, in the RI600/4 User's Manual.

(2) Function Name and File

In HI1000/4, the system-down routine is vsys_dwn( ) in sample program xxxx_sysdwn.c.
In RI600/4, the system-down routine is _RI_sys_dwn() in resetprg.c generated through the RI600/4 project creation function of High-performance Embedded Workshop.

11. Extended Language Specifications in Compiler

When the H8SX/H8S application uses extended language specifications (such as #pragma) dedicated for the H8SX/H8S compiler, the program parts where such specifications are used should be reviewed because the specifications differ between the H8SX/H8S and RX compilers. The RX compiler (ccrx) provides the "-check=ch38" option to check use of such extended specifications.

12. kernel_id.h

This header file is generated by the GUI configurator in the HI1000/4, but cfg600 generates it in RI600/4.

(1) In HI1000/4, inclusion of kernel_id.h in the application is optional, but it is mandatory in the task or handler entry function in RI600/4.
(2) As the prototype declarations of the task and handler entry functions are output to kernel_id.h in RI600/4, they should not be written in the application.
(3) For an interrupt handler, the user should write the #pragma interrupt declaration in HI1000/4. In RI600/4, the necessary #pragma declarations are output to kernel_id.h, and they should not be written in the application.

13. Kernel Configuration Macros and Information Acquired through ref_ver

The information of the kernel configuration macros listed in table 2 differs between HI1000/4 and RI600/4.
The GUI configurator for the HI1000/4 outputs the kernel configuration macros listed in table 2 to kernel_macro.h which is to be included in kernel.h. In comparison, cfg600 for the RI600/4 outputs these macros to kernel_id.h, which is not included in kernel.h.

Table 2 Kernel Configuration Macros that Differ between HI1000/4 and RI600/4

#	Macro	Description
1	TIC_NUME	Time tick numerator
2	TIC_DENO	Time tick denominator
3	TMAX_TPRI	Maximum task priority
4	TMAX_MPRI	Minimum message priority
5	VTKNL_LVL	Kernel interrupt mask level
6	VTIM_LVL	Timer interrupt level

14. Information on Build

14.1 Difference between Configurators

For HI1000/4, a GUI configurator is provided.
For RI600/4, a GUI configurator with a look and feel similar to those of the HI1000/4 GUI configurator is provided; command line configurator cfg600 is also provided.
cfg600 accepts a cfg file in text format as input. By using only cfg600, the configuration information can be written in a cfg file in text format, which makes configuration maintenance and modification control easy.
Accordingly, the GUI configurator for RI600/4 is regarded as a utility tool for creating a cfg file.

Note that the GUI configurator file (.hcf) for the HI1000/4 cannot be used for the RI600/4.

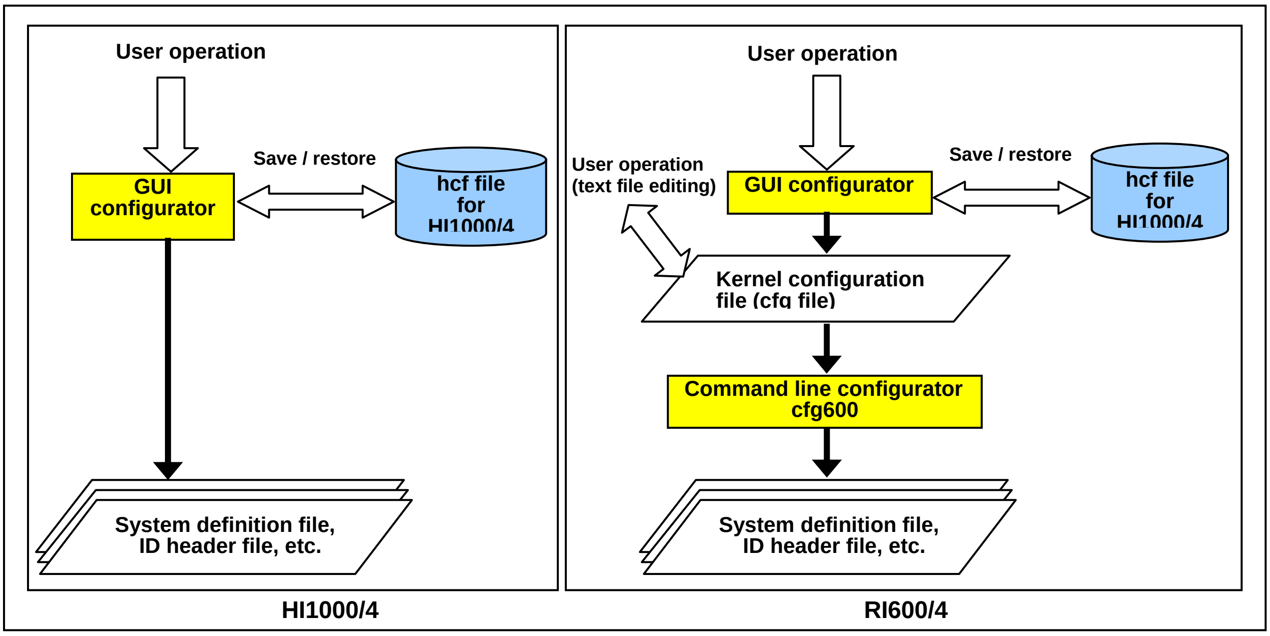

Figure 2 Difference in Role of Configurator

14.2 Difference in Build Procedure (Addition of mkritbl Utility)

As the service calls in HI1000/4 are implemented as a function library, they can be called by linking the library with the kernel library.
The service calls in RI600/4 are internally invoked with the INT instruction; the service call modules cannot be linked through symbolic linkage, so the mkritbl utility is provided for RI600/4.
When an application is compiled, an mrc file is generated, and this holds the information of the service calls used in the application source code. By inputting all mrc files to mkritbl, a file (ritable.src) is generated, which defines the linkage of the service call modules used in the system. For details, refer to section 9, Table Generation Utility, in the RI600/4 User's Manual.

14.3 Notes on Project Transition from HI1000/4 to RI600/4

The project for HI1000/4 (H8S, H8SX) cannot be used for RI600/4 (RX600) due to the specifications of High-performance Embedded Workshop, so a new RI600/4 project should be created.
Through High-performance Embedded Workshop provided in the RX family C/C++ compiler package, a template project for RI600/4 can be created. Create a project by using this function, then add the necessary application files and modify option settings.

The following describes important items that need special care.

14.3.1 Notes on Reset

(1) Reset Function Name and Source File

The reset function in HI1000/4 is _KERNEL_H_CPUINI in provided sample program xxxx_cpuasm.src. _KERNEL_H_CPUINI calls h_cpuini( ) in xxxx_cpu.c.
In RI600/4, the reset function is PowerON_Reset_PC( ) in resetprg.c, which is generated by the RI/600 project creation function of High-performance Embedded Workshop.

(2) Reset Vector Definition

The reset vector in HI1000/4 should be defined as vector #0 in the GUI configurator. In RI600/4, the GUI configurator or the cfg file defines it as fixed vector #31. When nothing is specified, PowerON_Reset_PC( ) is assigned by default; unless the user modifies the function name, the user does not need to explicitly specify fixed vector #31.

(3) Role of resetprg.c in RI600/4

The resetprg.c described above contains the following in addition to the reset function.

- System-down routine (see section 10, System-Down Routine)
- Inclusion of system definition files (kernel_rom.h and kernel_ram.h) output by cfg600

kernel_rom.h and kernel_ram.h have the same function as kernel_setup.src in HI1000/4.
For resetprg.c, compiler option "-nostuff" should be specified.
For details, refer to section 7.2, Creating Startup File (resetprg.c), in the RI600/4 User's Manual.

14.3.2 Compiler Options

In the RI600/4, the "-ri600_preinit_mrc" option should be specified for all files that include kernel.h.
For resetprg.c described above, the "-nostuff" option should be specified.

14.3.3 Correspondence of Files

The following shows the HI1000/4 sample files and their corresponding items in RI600/4.

#	Item	HI1000/4 Sample Files	RI600/4
1	Common type definition	(None)	typedefine.h *
2	I/O definition	(None)	iodefine.h *
3	Reset	xxxx_cpuasm.src, xxxx_cpu.h, xxxx_cpu,c	resetprg.c *, hwsetup.c *
4	System-down routine	xxxx_sysdwn.c	resetprg.c *
5	Undefined interrupt handler	xxxx_ilint.c	Generated by cfg600 and output to ri600.inc
6	Timer driver	xxxx_tmrdrv.c, xxxxtmrdrv.c	ri_cmt.h generated by cfg600
7	Idling	xxxx_idle.c	(No corresponding function)
8	Section initialization	(None)	dbsct.c *
9	Low-level standard I/O functions	(None)	lowsrc.h *, lowsrc.c *, lowlvl.src *
10	Low-level memory management functions for standard library	(None)	sbrk.h *, sbrk.c *
11	GUI configurator file	xxxx.hcf	None
12	cfg file	(None)	<project name>.cfg
13	Initialization routine	kernel_sysini.c (function that calls the initialization routine) generated by the GUI configurator	(No corresponding function)
14	Vector table	kenrel_vector.src generated by the GUI configurator	vector.tpl is generated by cfg600. Inputting it to mkritbl generates a vector table in ritable.src.
15	Header file	hihead¥itron.h	inc600¥itron.h
16	Header file	hihead¥kernel.h	inc600¥kernel.h
17	ID name header file	kernel_id.h generated by the GUI configurator	kernel_id.h generated by cfg600
18	Kernel configuration macros for application	kernel_macro.h generated by the GUI (to be included in kernel.h)	kernel_id.h generated by cfg600 (not to be included in kernel.h)
19	System definition file	kernel_setup.src generated by the GUI configurator	(1) kernel_rom.h and kernel_ram.h generated by cfg600 (to be included in resetprg.c) (2) ri600.inc generated by cfg600 (to be included in ritable.src, which is output by mkritbl)
20	Kernel library	A separate library is prepared for each operating mode in the hilib directory	lib600¥ri600big.lib (big endian) lib600¥ri600lit.lib (little endian)

Note: * indicates a file that is generated by the RI600/4 project creation function of High-performance Embedded Workshop.

15. Comparison of Specifications

Item		HI1000/4	RI600/4	Supplementary Description of Difference from HI1000/4	Remarks
Configuration information input		Input to the GUI configurator	Either of the following two ways (1) Input to the GUI configurator (2) Write to the cfg file	The way of writing to the cfg file is added.	See section 14.1, Difference between Configurators
Data type		See section 3, Parameter Data Type and Size			
Unused object ID		Allowed	Not allowed	Changed to "not allowed"	See section 4, Restrictions on Object ID.
Task management	Task ID	1 to 255	1 to 255		
	Task priority	1 to 31	1 to 255		
	Activation count	255	255		
	Extended information	16 bits	32 bits	16 bits → 32 bits	See section 3, Parameter Data Type and Size.
	Attribute	TA_HLNG TA_ASM	TA_HLNG TA_ASM		
	Shared stack	Supported	None	Deleted	See section3.1, Deletion of Shared Stack Function.
	act_tsk	TEDU	TEDU		
	iact_tsk	NEDU	NEDU		
	can_act	TEDU	TEDU		
	ican_act	(Not supported)	NEDU	Added	
	sta_tsk	TEDU	TEDU		
	ista_tsk	NEDU	NEDU		
	ext_tsk	TEDUL	TEDUL		
	ter_tsk	TEDU	TEDU		
	chg_pri	TEDU	TEDU		
	ichg_pri	(Not supported)	NEDU	Added	
	get_pri	TEDU	TEDU		
	iget_pri	(Not supported)	NEDU	Added	
	ref_tsk	TEDU	TEDU		
	iref_tsk	NEDU	NEDU		
	ref_tst	TEDU	TEDU		
	iref_tst	NEDU	NEDU		
Task-dependent synchronization	Wakeup count	255	255		
	Suspension count	1	1		
	slp_tsk	TEU	TEU		
	tslp_tsk	TEU	TEU		
	wup_tsk	TEDU	TEDU		
	iwup_tsk	NEDU	NEDU		
	can_wup	TEDU	TEDU		
	ican_wup	(Not supported)	NEDU	Added	
	rel_wai	TEDU	TEDU		
	irel_wai	NEDU	NEDU		
	sus_tsk	TEDU	TEDU		
	isus_tsk	(Not supported)	NEDU	Added	
	rsm_tsk	TEDU	TEDU		
	irsm_tsk	(Not supported)	NEDU	Added	
	frsm_tsk	TEDU	TEDU		
	ifrsm_tsk	(Not supported)	NEDU	Added	
	dly_tsk	TEU	TEU		
Semaphore	ID number	1 to 255	1 to 255		

Item		HI1000/4	RI600/4	Supplementary Description of Difference from HI1000/4	Remarks
	Maximum counter value	65535	65535		
	Attribute	TA_TFIFO	TA_TFIFO TA_TPRI	TA_TPRI (queued in order of task priority) is added	
	sig_sem	TEDU	TEDU		
	isig_sem	NEDU	NEDU		
	wai_sem	TEU	TEU		
	pol_sem	TEDU	TEDU		
	ipol_sem	NEDU	NEDU		
	twai_sem	TEU	TEU		
	ref_sem	TEDU	TEDU		
	iref_sem	NEDU	NEDU		
Eventflag	ID number	1 to 255	1 to 255		
	Flag length	16 bits	32 bits	16 bits → 32 bits	See section 3, Parameter Data Type and Size.
	Attribute	TA_TFIFO TA_WSGL TA_WMUL TA_CLR	TA_TFIFO TA_TPRI TA_WSGL TA_WMUL TA_CLR	TA_TPRI (queued in order of task priority) is added	
	set_flg	TEDU	TEDU		
	iset_flg	NEDU	NEDU		
	clr_clg	TEDU	TEDU		
	iclr_flg	NEDU	NEDU		
	wai_flg	TEU	TEU		
	pol_flg	TEDU	TEDU		
	ipol_flg	NEDU	NEDU		
	twai_flg	TEU	TEU		
	ref_flg	TEDU	TEDU		
	iref_flg	NEDU	NEDU		
Data queue	ID number	1 to 255	1 to 255		
	Data length	32 bits	32 bits		
	Attribute	TA_TFIFO	TA_TFIFO TA_TPRI	TA_TPRI (queued in order of task priority) is added	
	snd_dtq	TEU	TEU		
	psnd_dtq	TEDU	TEDU		
	ipsnd_dtq	NEDU	NEDU		
	tsnd_dtq	TEU	TEU		
	fsnd_dtq	TEDU	TEDU		
	ifsnd_dtq	NEDU	NEDU		
	rcv_dtq	TEU	TEU		
	prcv_dtq	TEDU	TEDU		
	iprcv_dtq	(Not supported)	NEDU	Added	
	trcv_dtq	TEU	TEU		
	ref_dtq	TEDU	TEDU		
	iref_dtq	NEDU	NEDU		
Mailbox	ID number	1 to 255	1 to 255		
	MSG priority	1 to 255	1 to 255		
	Attribute	TA_TFIFO TA_TPRI TA_MFIFO TA_MPRI	TA_TFIFO TA_TPRI TA_MFIFO TA_MPRI		
	snd_mbx	TEDU	TEDU		
	isnd_mbx	NEDU	NEDU		
	rcv_mbx	TEU	TEU		
	prcv_mbx	TEDU	TEDU		

Item		HI1000/4	RI600/4	Supplementary Description of Difference from HI1000/4	Remarks
	iprcv_mbx	NEDU	NEDU		
	trcv_mbx	TEU	TEU		
	ref_mbx	TEDU	TEDU		
	iref_mbx	NEDU	NEDU		
Mutex	ID number	1 to 255	1 to 255		
	Attribute	TA_CEILING	TA_CEILING		
	loc_mtx	TEU	TEU		
	ploc_mtx	TEDU	TEDU		
	tloc_mtx	TEU	TEU		
	unl_mtrx	TEDU	TEDU		
	ref_mtx	TEDU	TEDU		
Message buffer	ID number	(Not supported)	1 to 255	Message buffer function is added	
	Attribute		TA_TFIFO		
	snd_mbf		TEU		
	psnd_mbf		TEDU		
	ipsnd_mbf		NEDU		
	tsnd_mbf		TEU		
	rcv_mbf		TEU		
	prcv_mbf		TEDU		
	trcv_mbf		TEU		
	ref_mbf		TEDU		
	iref_mbf		NEDU		
Fixed-sized memory pool	ID number	1 to 255	1 to 255		
	Maximum block count	65535	65535		
	Maximum block size	65530	65535		
	Attribute	TA_TFIFO	TA_TFIFO TA_TPRI	TA_TPRI (queued in order of task priority) is added	
	get_mpf	TEU	TEU		
	pget_mpf	TEDU	TEDU		
	ipget_mpf	NEDU	NEDU		
	tget_mpf	TEU	TEU		
	rel_mpf	TEDU	TEDU		
	irel_mpf	(Not supported)	NEDU	Added	
	ref_mpf	TEDU	TEDU		
	iref_mpf	NEDU	NEDU		
Variable-sized memory pool	ID number	1 to 255	1 to 255		
	Management method	See section 5.1, Pool Size			
	Maximum pool size	65534	256MB	64 KB → 256 MB	
	Maximum size of allocatable block	Pool size – 16	Defined in configuration	The specification of the maximum size of allocatable block is changed to be defined in configuration.	See section 5.2, Maximum Size of Allocatable Block.
	Attribute	TA_TFIFO	TA_TFIFO		
	get_mpl	TEU	TEU		
	pget_mpl	TEDU	TEDU		
	ipget_mpl	NEDU	NEDU		
	tget_mpl	TEU	TEU		
	rel_mpl	TEDU	TEDU		
	ref_mpl	TEDU	TEDU		
	iref_mpl	NEDU	NEDU		
Time management	System time	Unsigned 48 bits	Unsigned 48 bits		
	Unit time	1 ms	1 ms		
	Update cycle of system time	TIC_NUME /TIC_DENO [ms]	TIC_NUME /TIC_DENO [ms]		
	set_tim	TEDU	TEDU		

Item		HI1000/4	RI600/4	Supplementary Description of Difference from HI1000/4	Remarks
	iset_tim	NEDU	NEDU		
	get_tim	TEDU	TEDU		
	iget_tim	NEDU	NEDU		
Cyclic handler	ID number	1 to 254	1 to 255		
	Extended information	32 bits	32 bits		
	Attribute	TA_HLNG TA_ASM TA_STA TA_PHS	TA_HLNG TA_ASM TA_STA TA_PHS		
	sta_cyc	TEDU	TEDU		
	ista_cyc	NEDU	NEDU		
	stp_cyc	TEDU	TEDU		
	istp_cyc	NEDU	NEDU		
	ref_cyc	TEDU	TEDU		
	iref_cyc	NEDU	NEDU		
Alarm handler	ID number	(Not supported)	1 to 255	The alarm handler function is added.	
	Extended information		32 bits		
	Attribute		TA_HLNG, TA_ASM		
	sta_alm		TEDU		
	ista_alm		NEDU		
	stp_alm		TEDU		
	istp_alm		NEDU		
	ref_alm		TEDU		
	iref_alm		NEDU		
System state management	rot_rdq	TEDU	TEDU		
	irot_rdq	NEDU	NEDU		
	get_tid	TEDU	TEDU		
	iget_tid	NEDUC	NEDU		
	loc_cpu	TEDUL	TEDUL		
	iloc_cpu	NEDUL	NEDUL		
	unl_cpu	TEDUL	TEDUL		
	iunl_cpu	NEDUL	NEDUL		
	dis_dsp	TEDU	TEDU		
	ena_dsp	TEDU	TEDU		
	sns_ctx	TNEDULC	TNEDUL		
	sns_loc	TNEDULC	TNEDUL		
	sns_dsp	TNEDULC	TNEDUL		
	sns_dpn	TNEDULC	TNEDUL		
	vsta_knl	Supported	Supported		
	ivsta_knl	Supported	Supported		
	vsys_dwn	TEDUL	TEDUL		
	ivsys_dwn	NEDULC	NEDUL		
	ivbgn_int	NEDU	(Not supported)	Deleted	See section 3.2, Deletion of Service Call Trace Function and Related Service Calls.
	ivend_ont	NEDU			
Interrupt management	chg_ims	TEU	TEDU		See section 6, Interrupt Priority Level and Interrupt Mask.
	ichg_ims	NEU	NEDU		
	get_ims	TEDU	TEDU		
	iget_ims	NEDU	NEDU		
	ret_int	N	N		
System configuration management	CPU exception handler	Supported	Treated as a non-kernel interrupt	See section 3.3, Deletion of CPU Exception Handler.	
	ref_ver	TEDU	TEDU		
	iref_ver	NEDU	NEDU		

Item		HI1000/4	RI600/4	Supplementary Description of Difference from HI1000/4	Remarks
Object reset function	vrst_dtq	(Not supported)	TEDU	The function for resetting objects to initial state is added.	
	vrst_mbx		TEDU		
	vrst_mpf		TEDU		
	vrst_mpl		TEDU		
	vrst_mbf		TEDU		

Note: N: Invocable from non-task contexts

T: Invocable from task contexts

E: Invocable from a dispatching-enabled state

D: Invocable from a dispatching-disabled state

U: Invocable from the CPU-unlocked state

L: Invocable from the CPU-locked state

C: Invocable from the CPU exception handler (only in the HI1000/4)

Website and Support

Renesas Technology Website

http://www.renesas.com/

Inquiries

http://www.renesas.com/inquiry

csc@renesas.com

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Feb. 16, 2010	—	First edition issued

Notes regarding these materials

1. This document is provided for reference purposes only so that Renesas customers may select the appropriate Renesas products for their use. Renesas neither makes warranties or representations with respect to the accuracy or completeness of the information contained in this document nor grants any license to any intellectual property rights or any other rights of Renesas or any third party with respect to the information in this document.
2. Renesas shall have no liability for damages or infringement of any intellectual property or other rights arising out of the use of any information in this document, including, but not limited to, product data, diagrams, charts, programs, algorithms, and application circuit examples.
3. You should not use the products or the technology described in this document for the purpose of military applications such as the development of weapons of mass destruction or for the purpose of any other military use. When exporting the products or technology described herein, you should follow the applicable export control laws and regulations, and procedures required by such laws and regulations.
4. All information included in this document such as product data, diagrams, charts, programs, algorithms, and application circuit examples, is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas products listed in this document, please confirm the latest product information with a Renesas sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas such as that disclosed through our website. (http://www.renesas.com)
5. Renesas has used reasonable care in compiling the information included in this document, but Renesas assumes no liability whatsoever for any damages incurred as a result of errors or omissions in the information included in this document.
6. When using or otherwise relying on the information in this document, you should evaluate the information in light of the total system before deciding about the applicability of such information to the intended application. Renesas makes no representations, warranties or guaranties regarding the suitability of its products for any particular application and specifically disclaims any liability arising out of the application and use of the information in this document or Renesas products.
7. With the exception of products specified by Renesas as suitable for automobile applications, Renesas products are not designed, manufactured or tested for applications or otherwise in systems the failure or malfunction of which may cause a direct threat to human life or create a risk of human injury or which require especially high quality and reliability such as safety systems, or equipment or systems for transportation and traffic, healthcare, combustion control, aerospace and aeronautics, nuclear power, or undersea communication transmission. If you are considering the use of our products for such purposes, please contact a Renesas sales office beforehand. Renesas shall have no liability for damages arising out of the uses set forth above.
8. Notwithstanding the preceding paragraph, you should not use Renesas products for the purposes listed below:
   (1) artificial life support devices or systems
   (2) surgical implantations
   (3) healthcare intervention (e.g., excision, administration of medication, etc.)
   (4) any other purposes that pose a direct threat to human life

   Renesas shall have no liability for damages arising out of the uses set forth in the above and purchasers who elect to use Renesas products in any of the foregoing applications shall indemnify and hold harmless Renesas Technology Corp., its affiliated companies and their officers, directors, and employees against any and all damages arising out of such applications.
9. You should use the products described herein within the range specified by Renesas, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas shall have no liability for malfunctions or damages arising out of the use of Renesas products beyond such specified ranges.
10. Although Renesas endeavors to improve the quality and reliability of its products, IC products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other applicable measures. Among others, since the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
11. In case Renesas products listed in this document are detached from the products to which the Renesas products are attached or affixed, the risk of accident such as swallowing by infants and small children is very high. You should implement safety measures so that Renesas products may not be easily detached from your products. Renesas shall have no liability for damages arising out of such detachment.
12. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written approval from Renesas.
13. Please contact a Renesas sales office if you have any questions regarding the information contained in this document, Renesas semiconductor products, or if you have any other inquiries.