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.

Notice

1. All information included in this document 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 Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of 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. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may 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.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

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

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

Application Note

µITRON Specification OS Transition Manual Transition Guide from HI7000 Series to HI7000/4 Series

Application Note

Renesas Microcomputer Development Environment System

Renesas Electronics
www.renesas.com

Rev.1.00 2005.09

Keep safety first in your circuit designs!

1. Renesas Technology Corp. puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage.
   Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

1. These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corp. product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corp. or a third party.
2. Renesas Technology Corp. assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
3. All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corp. without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corp. or an authorized Renesas Technology Corp. product distributor for the latest product information before purchasing a product listed herein.
   The information described here may contain technical inaccuracies or typographical errors.
   Renesas Technology Corp. assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
   Please also pay attention to information published by Renesas Technology Corp. by various means, including the Renesas Technology Corp. Semiconductor home page (http://www.renesas.com).
4. When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corp. assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corp. semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corp. or an authorized Renesas Technology Corp. product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
6. The prior written approval of Renesas Technology Corp. is necessary to reprint or reproduce in whole or in part these materials.
7. If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
   Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
8. Please contact Renesas Technology Corp. for further details on these materials or the products contained therein.

1. TRON is an acronym of “The Realtime Operating system Nucleus”, ITRON is an acronym of the “Industrial TRON”, and µITRON is an acronym of the “Micro Industrial TRON”.
2. TRON, ITRON, and µITRON are the names of computer specifications and do not indicate a specific group of the commodity or the commodity.
3. The µITRON4.0 specification and µITRON4.0 protection function extension are open realtime-kernel specifications defined by the TRON association. The specifications of µITRON4.0 and µITRON4.0 protection function extension can be downloaded from the TRON association homepage (http://www.ertl.jp/ITRON/home-e.html).
4. The copyright of the µITRON specification belongs to the TRON association.
5. Microsoft® Windows® 98, Microsoft® Windows® Millennium Edition (Windows® Me) operating system, Microsoft® Windows NT® operating system, Microsoft® Windows® 2000 operating system, and Microsoft® Windows® XP operating system are registered trademarks of Microsoft Corporation in the United States and/or other countries.
6. SuperHTM is a trademark of Renesas Technology Corp..
7 All other product names are trademarks or registered trademarks of the respective holders.

Contents

Section 1 Preface

This guide describes how to shift from the HI7000 series based on the µITRON3.0 specifications to the HI7000/4 series based on the µITRON4.0 specifications.

This guide does not guarantee correct system transition from the HI7000 series to the HI7000/4 series.

For detailed specifications, be sure to refer to the manuals of both series.

Table 1.1 Correspondence between CPU and HI Series

	HI Series Product	
CPU	**HI7000 Series**	**HI7000/4 Series**
SH-1, SH-2, SH-DSP	HI7000	HI7000/4
SH-2A, SH2A-FPU	—	
SH-3, SH3-DSP	HI7700	HI7700/4
SH4AL-DSP	—	
SH-4	HI7750	HI7750/4
SH-4A	—	

Section 2 Overview

This section gives an overview of the functions that are deleted, added, or modified with regard to the shift from the HI7000 series to HI7000/4 series.

For the details of each function, refer to the respective manual.

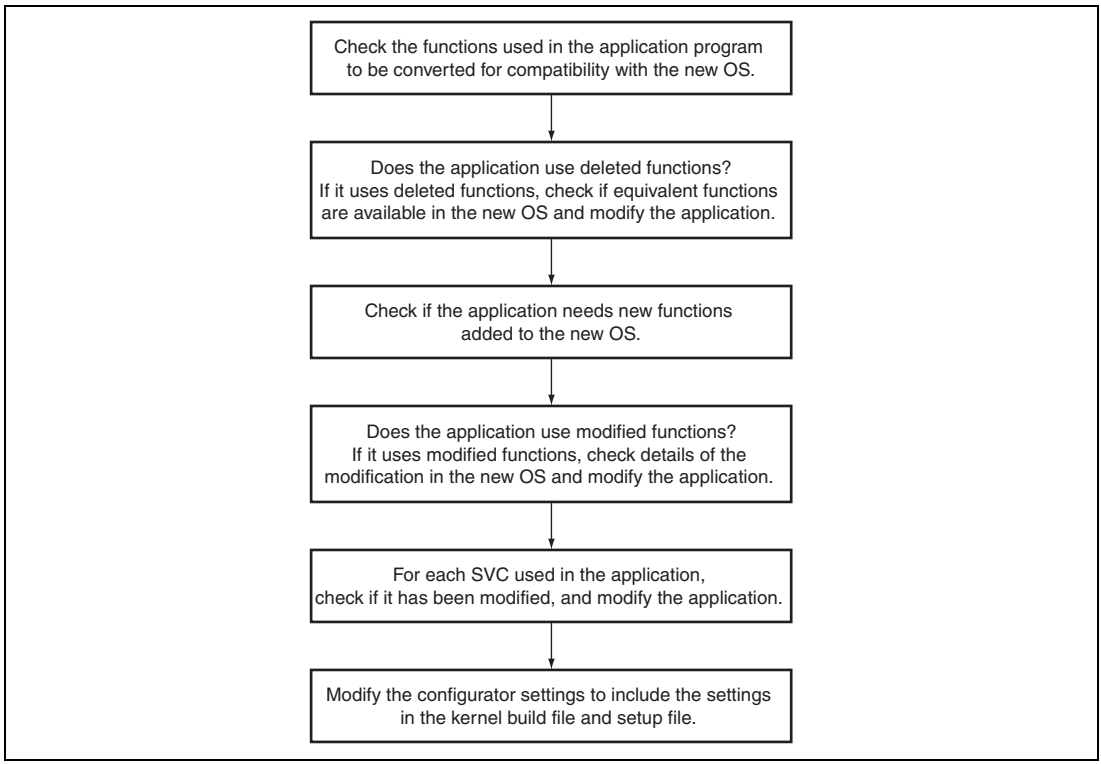

Figure 2.1 Transition Steps

Table 2.1 gives a comparative overview of the HI7000 series and HI7000/4 series.

Table 2.1 Comparison between HI7000 Series and HI7000/4 Series

No.	HI7000 Series Function	Modification in HI7000/4 Series	Section in This Guide
1	Time-slicing	Substitute function available: Overrun handler	3.1.1
2	Page pool	No substitute function	3.1.2
3	I/O support	No substitute function	3.1.3
4	Data queue	New function	3.2.1
5	Mutex	New function	3.2.2
6	Task exception	New function	3.2.3
7	Overrun handler	New function	3.2.4
8	Multiple task start requests	New function	3.2.5
9	Service calls starting with "i"	New function	3.2.6
10	Dispatch and CPU lock	Meanings of the words are modified	3.3.1
11	Extended service call routine	Context is modified	3.3.2
12	Event flag	Attributes are added and clearing specification is modified	3.3.3, 3.3.4
13	Mailbox	An attribute and new message header type are added	3.3.5, 3.3.6
14	Message buffer	Wait queue is modified	3.3.7
15	wup_tsk, sus_tsk	Current task can be specified	3.3.8
16	Sample timer driver	Modifications made throughout the function	3.3.9
17	Time management	Time unit and management are modified	3.3.10
18	Cyclic handler	Modifications made throughout the function	3.3.11
19	Alarm handler	Modifications made throughout the function	3.3.12
20	TA_COP1 and TA_COP2 attributes	Meanings of the attributes are modified	3.3.13
21	Object name	Substitute means available for part of the function	3.3.14
22	System down routine	Routine name and system down information are modified	3.3.15

Section 3 Added, Deleted, and Modified Functions

This section describes the functions deleted, added, or modified in the HI7000/4 series.

For details of each function, refer to the respective manual.

3.1 Deleted Functions

3.1.1 Time-Slicing

The time-slice time setting for task creation in the HI7000 series is deleted. To achieve time-slicing in the HI7000/4 series, use the overrun handler instead.

Table 3.1 and figure 3.1 show how to use the overrun handler instead of the time-slice function.

Table 3.1 Substitute for Time-Slice Function

No.	Time-Slice Function	HI7000 Series	HI7000/4 Series
1	Time-slice time	Specified at task creation (time-slice time)	Specified through sta_ovr or ista_ovr (upper limit processor time for the task)
2	Ready queue rotation	rot_rdq executed automatically	The user must execute rot_rdq in the overrun handler. The upper limit processor time for the task that has caused initiation of the overrun handler is cleared and must be specified again.

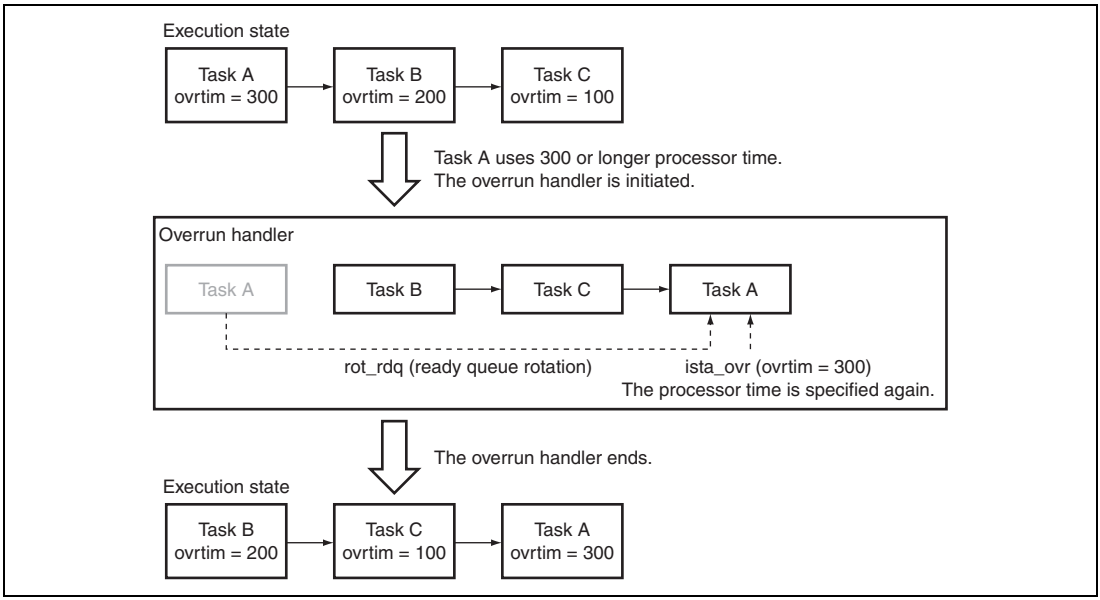

Figure 3.1 Example of Time-Slice Function Implemented through Overrun Handler Function

3.1.2 Page Pool

The HI7000/4 series does not provide a substitute means for it.

If it is used in the application, consider to use a fixed-size memory pool to implement this function.

3.1.3 I/O Support

The HI7000/4 series does not provide a substitute means for it.

If it is used in the application, consider to use an extended SVC to implement this function.

3.1.4 Other Functions (vrst_tsk, vjmp_msg, vasn_svc, ref_sys)

Table 3.2 shows the HI7000/4 series substitute means for the functions deleted from the HI7000 series.

Table 3.2 Correspondence to HI7000/4 Series Functions

No.	SVC	Substitute Means
1	vrst_tsk	No substitute means available. Consider to terminate and then restart the task.
2	vjmp_msg	Transmit the message with the highest message priority or execute fsnd_dtq through the data queue.
3	vasn_svc	Use def_svc. Note the following. The function for searching for an undefined extended function code and assigning it to an extended SVC cannot be implemented through def_svc, but an extended SVC can be assigned to a function code through def_svc.
4	ref_sys	No substitute means available. Consider to use sns_ctx, sns_loc, sns_dsp, or sns_dpn to implement this function.

3.2 Added Functions

3.2.1 Data Queue

The HI7000/4 series has a new function called a data queue.

This function provides synchronization and communication through 1-word message (data) transfer.

Table 3.3 gives an overview of the data queue function.

Table 3.3 Data Queue Function

No.	Function	SVC	API
1	Create a data queue	cre_dtq	ER ercd = cre_dtq(ID dtqid, T_CDTQ *pk_cdtq);
		icre_dtq	ER ercd = icre_dtq(ID dtqid, T_CDTQ *pk_cdtq);
2	Create a data queue (assign ID automatically)	acre_dtq	ER_ID dtqid = acre_dtq(T_CDTQ *pk_cdtq);
		iacre_dtq	ER_ID dtqid = iacre_dtq(T_CDTQ *pk_cdtq);
3	Delete a data queue	del_dtq	ER ercd = del_dtq(ID dtqid);
4	Send data to a data queue	snd_dtq	ER ercd = snd_dtq(ID dtqid, VP_INT data);
5	Send data to a data queue (polling)	psnd_dtq	ER ercd = psnd_dtq(ID dtqid, VP_INT data);
		ipsnd_dtq	ER ercd = ipsnd_dtq(ID dtqid, VP_INT data);
6	Send data to a data queue (with timeout)	tsnd_dtq	ER ercd = tsnd_dtq(ID dtqid, VP_INT data, TMO tmout);
7	Forcibly send data to a data queue	fsnd_dtq	ER ercd = fsnd_dtq(ID dtqid, VP_INT data);
		ifsnd_dtq	ER ercd = ifsnd_dtq(ID dtqid, VP_INT data);
8	Receive data from a data queue	rcv_dtq	ER ercd = rcv_dtq(ID dtqid, VP_INT *p_data);
9	Receive data from a data queue (polling)	prcv_dtq	ER ercd = prcv_dtq(ID dtqid, VP_INT *p_data);
10	Receive data from a data queue (with timeout)	trcv_dtq	ER ercd = trcv_dtq(ID dtqid, VP_INT *p_data, TMO tmout);
11	Refer to the data queue state	ref_dtq	ER ercd = ref_dtq(ID dtqid, T_RDTQ *pk_rdtq);
		iref_dtq	ER ercd = iref_dtq(ID dtqid, T_RDTQ *pk_rdtq);

3.2.2 Mutex

The HI7000/4 series has a new function called a mutex.

This function provides exclusive control for resources shared among tasks.

Table 3.4 gives an overview of the mutex function.

Table 3.4 Mutex Function

No.	Function	SVC	API
1	Create a mutex	cre_mtx	ER ercd = cre_mtx(ID mtxid, T_CMTX *pk_cmtx);
2	Create a mutex (assign ID automatically)	acre_mtx	ER_ID mtxid = acre_mtx(T_CMTX *pk_cmtx);
3	Delete a mutex	del_mtx	ER ercd = del_mtx(ID mtxid);
4	Lock a mutex	loc_mtx	ER ercd = loc_mtx(ID mtxid);
5	Lock a mutex (polling)	ploc_mtx	ER ercd = ploc_mtx(ID mtxid);
6	Lock a mutex (with timeout)	tloc_mtx	ER ercd = tloc_mtx(ID mtxid, TMO tmout);
7	Unlock a mutex	unl_mtx	ER ercd = unl_mtx(ID mtxid);
8	Refer to the mutex state	ref_mtx	ER ercd = ref_mtx(ID mtxid, T_RMTX *pk_rmtx);

3.2.3 Task Exception Processing

The HI7000/4 series has a new function called task exception processing.

This function processes exceptions generated in tasks.

Table 3.5 gives an overview of the task exception processing.

Table 3.5 Task Exception Processing

No.	Function	SVC	API
1	Define a task exception processing routine	def_tex	ER ercd = def_tex (ID tskid, T_DTEX *pk_dtex);
		idef_tex	ER ercd = idef_tex (ID tskid, T_DTEX *pk_dtex);
2	Request task exception processing	ras_tex	ER ercd = ras_tex(ID tskid, TEXPTN rasptn);
		iras_tex	ER ercd = iras_tex(ID tskid, TEXPTN rasptn);
3	Disable task exception processing	dis_tex	ER ercd = dis_tex();
4	Enable task exception processing	ena_tex	ER ercd = ena_tex();
5	Refer to the task exception disable state	sns_tex	BOOL state =sns_tex();
6	Refer to the task exception processing state	ref_tex	ER ercd = ref_tex(ID tskid, T_RTEX *pk_rtex);
		iref_tex	ER ercd = iref_tex(ID tskid, T_RTEX *pk_rtex);

3.2.4 Overrun Handler

The HI7000/4 series has a new function called the overrun handler.

This function performs processing when a task uses the processor for a time exceeding the predetermined time.

Table 3.6 gives an overview of the overrun handler function.

Table 3.6 Overrun Handler Function

No.	Function	SVC	API
1	Define the overrun handler	def_ovr	ER ercd = def_ovr(T_DOVR *pk_dovr);
2	Start the overrun handler	sta_ovr	ER ercd = sta_ovr(ID tskid, OVRTIM ovrtim);
		ista_ovr	ER ercd = ista_ovr(ID tskid, OVRTIM ovrtim);
3	Stop the overrun handler	stp_ovr	ER ercd = stp_ovr(ID tskid);
		istp_ovr	ER ercd = istp_ovr(ID tskid);
4	Refer to the overrun handler state	ref_ovr	ER ercd = ref_ovr(ID tskid, T_ROVR *pk_rovr);
		iref_ovr	ER ercd = iref_ovr(ID tskid, T_ROVR *pk_rovr);

3.2.5 Multiple Start Requests for a Task

The HI7000/4 series enables multiple start requests to be issued for a task.

By using this function, when act_tsk is issued to a task that has already been started, the attempted request is recorded and the task is automatically started again when the task is terminated.

This function is implemented by the act_tsk service call.

3.2.6 Service Call Names Starting with "i"

The HI7000/4 series provides new service calls whose names start with "i". These service calls are dedicated to a non-task context. In a non-task context, only these dedicated service calls should be used. Table 3.7 is a list of these service calls.

Table 3.7 List of Service Calls Dedicated to Non-Task Context

No.	Service Call	Function	No.	Service Call	Function
1	icre_tsk	Create a task (using a dynamic stack)	2	ivscr_tsk	Create a task (using a static stack)
3	iacre_tsk	Create a task (assign ID automatically)	4	iact_tsk	Start a task
5	ista_tsk	Start a task (specifying start code)	6	ican_act	Cancel start task request
7	ichg_pri	Change the task priority	8	iget_pri	Refer to the task priority
9	iref_tsk	Refer to the task state	10	iref_tst	Refer to the task state (simple version)
11	iwup_tsk	Wake up a task	12	ican_wup	Cancel wakeup task
13	irel_wai	Release WAITING state forcibly	14	isus_tsk	Suspend a task
15	irsm_tsk	Resume a suspended task	16	ifrsm_tsk	Resume a suspended task forcibly
17	ivset_ftl	Set a task event flag	18	ivclr_tfl	Clear a task event flag
19	idef_tex	Define a task exception processing routine	20	iras_tex	Request task exception processing
21	iref_tex	Refer to the task exception processing	22	icre_sem	Create a semaphore
23	iacre_sem	Create a semaphore (assign ID automatically)	24	isig_sem	Release a semaphore resource

No.	Service Call	Function	No.	Service Call	Function
25	ipol_sem	Get a semaphore resource (polling)	26	iref_sem	Refer to the semaphore state
27	icre_flg	Create an event flag	28	iacre_flg	Create an event flag (assign ID automatically)
29	iset_flg	Set an event flag	30	iclr_flg	Clear an event flag
31	ipol_flg	Wait for an event flag (polling)	32	iref_flg	Refer to the event flag state
33	icre_dtq	Create a data queue	34	iacre_dtq	Create a data queue (assign ID automatically)
35	ipsnd_dtq	Send data to a data queue (polling)	36	ifsnd_dtq	Send data to a data queue forcibly
37	iref_dtq	Refer to the data queue state	38	icre_mbx	Create a mailbox
39	iacre_mbx	Create a mailbox (assign ID automatically)	40	isnd_mbx	Send data to a mailbox
41	iprcv_mbx	Receive data from a mailbox (polling)	42	iref_mbx	Refer to the mailbox state
43	icre_mbf	Create a message buffer	44	iacre_mbf	Create a message buffer (assign ID automatically)
45	ipsnd_mbf	Send data to a message buffer (polling)	46	iref_mbf	Refer to the message buffer state
47	icre_mpf	Create a fixed-size memory pool	48	iacre_mpf	Create a fixed-size memory pool (assign ID automatically)
49	ipget_mpf	Get a fixed-size memory block (polling)	50	irel_mpf	Release a fixed-size memory block
51	iref_mpf	Refer to the fixed-size memory pool state	52	icre_mpl	Create a variable-size memory pool
53	iacre_mpl	Create a variable-size memory pool (assign ID automatically)	54	ipget_mpl	Get a variable-size memory block (polling)
55	irel_mpl	Release a variable-size memory block	56	iref_mpl	Refer to the variable-size memory pool state
57	iset_tim	Set the system clock	58	iget_tim	Refer to the system clock
59	icre_cyc	Create a cyclic handler	60	iacre_cyc	Create a cyclic handler (assign ID automatically)
61	ista_cyc	Start a cyclic handler	62	istp_cyc	Stop a cyclic handler
63	iref_cyc	Refer to the cyclic handler state	64	icre_alm	Create an alarm handler
65	iacre_alm	Create an alarm handler (assign ID automatically)	66	ista_alm	Start an alarm handler
67	istp_alm	Stop an alarm handler	68	iref_alm	Refer to the alarm handler state

No.	Service Call	Function	No.	Service Call	Function
69	ista_ovr	Start the overrun handler	70	istp_ovr	Stop the overrun handler
71	iref_ovr	Refer to the overrun handler state	72	irot_rdq	Rotate the ready queue
73	iget_tid	Refer to the task ID in RUNNING state	74	iloc_cpu	Lock the CPU
75	iunl_cpu	Unlock the CPU	76	ivsta_knl	Start the kernel (*1)
77	ivsys_dwn	Terminate the system	78	ivget_trc	Get trace information
79	idef_inh	Define an interrupt handler	80	ichg_ims	Change the interrupt mask
81	iget_ims	Refer to the interrupt mask	82	idef_svc	Define an extended service call
83	ical_svc	Issue a service call	84	idef_exc	Define a CPU exception handler
85	ivdef_trp	Define a CPU exception handler (TRAPA instruction exception)	86	iref_cfg	Refer to the configuration information
87	iref_ver	Refer to the version information	88	ivini_cac	Initialize the cache
89	ivclr_cac	Clear the cache	90	ivfls_cac	Flush the cache
91	ivinv_cac	Invalidate the cache			

3.3 Modified Functions

3.3.1 Meanings of Dispatch-Disabled State and CPU-Locked State

In the HI7000 series, the CPU-locked state indicates the state where the dispatch and interrupts are disabled. In the HI7000/4 series, the CPU-locked state and dispatch-disabled state are managed as independent states.

For example, in the HI7000/4 series, after the system enters the CPU-locked state from the dispatch-disabled state, if the CPU-locked state is canceled, the system remains in the dispatch-disabled state. On the other hand, after the system enters the CPU-locked state from the dispatch-enabled state, if the CPU-locked state is canceled, the system enters the dispatch-enabled state.

When shifting to the HI7000/4 series, note the dispatch-disabled state and CPU-locked state after unl_cpu, loc_cpu, dis_dsp, or ena_dsp execution.

Table 3.8 State Transition by dis_dsp and loc_cpu Service Calls in HI7000 Series

State No.	System State	Current State		Service Call Issued and State No. Entered			
		Interrupt	Dispatch	dis_dsp	ena_dsp	loc_cpu	unl_cpu
1	RUNNING	Enabled	Enabled	→2	→1	→3	→1
2	Dispatch disabled	Enabled	Disabled	→2	→1	→3	→1
3	CPU locked	Disabled	Disabled	E_CTX	E_CTX	→3	→1

Table 3.9 State Transition by dis_dsp and loc_cpu Service Calls in HI7000/4 Series

State No.	System State	Current State		Service Call Issued and State No. Entered			
		CPU Lock	Dispatch	dis_dsp	ena_dsp	loc_cpu	unl_cpu
1	Dispatch enabled	—	Enabled	→2	→1	→4	→3
2	Dispatch disabled	—	Disabled	→2	→1	→4	→3
3	CPU unlocked	Unlocked	—	→2	→1	→4	→3
4	CPU locked	Locked	—	E_CTX	E_CTX	→4	→3

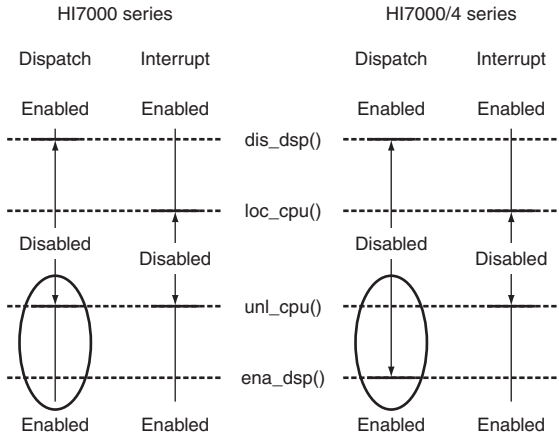

Figure 3.2 Modification of Dispatch-Disabled State and CPU-Locked State

3.3.2 Context for Execution of Extended Service Call Routines

The context in which extended service call (extended SVC) routines are executed is modified in the HI7000/4 series as shown in table 3.10.

In the HI7000 series, task dispatch is not performed. In the HI7000/4 series, note that task dispatch is performed if an extended service call is issued in a task context; modify the extended SVC routines that were created assuming that no task dispatch is performed.

When modifying such extended SVC routines, add dis_dsp() and ena_dsp() to achieve the same behavior as in the HI7000 series.

Table 3.10 Context for Execution of Extended Service Calls

Caller	HI7000 Series	HI7000/4 Series
Task context or task portion	Task-independent portion (no task dispatch)	Task context (dispatch performed)
Non-task context or task-independent portion		Non-task context (no task dispatch)

```
ER_UINT Svcrtn(VP_INT par1, VP_INT par2)   ← The parameters specified in cal_svc are
{                                             passed to the extended service call routine.
   /* Extended SVC routine processing */
    return E_OK;                           ← Passes a return value to the caller.
}
```

Figure 3.3 Example of C-Language Extended Service Call Routine in HI7000/4 Series

```
ER_UINT Svcrtn(VP_INT par1, VP_INT par2)
{
    dis_dsp()                              ← Disables dispatch (*)

   /* Extended SVC routine processing */

    ena_dsp()                              ← Enables dispatch (*)

    return E_OK;                           ← Passes a return value to the caller.
}
```

Figure 3.4 Example of Extended Service Call Routine Modified from HI7000 Series (Task Context)

Note: * Disable and enable task dispatch only when the extended service call routine is called in a task context. When called in a non-task context, these disabling and enabling service calls are not necessary.

3.3.3 New Attributes for Event Flags

New attributes (shaded in the table) can be used for event flags in the HI7000/4 series. Table 3.11 shows the event flag attributes.

Table 3.11 Event Flag Attributes

Event Flag Attribute	Description	HI7000 Series	HI7000/4 Series
TA_TFIFO	Task wait queue is managed on a FIFO basis	× (*)	Available
TA_TPRI	Task wait queue is managed on priority	×	Available
TA_WSGL	Does not permit multiple tasks to wait for the event flag	Available	Available
TA_WMUL	Permits multiple tasks to wait for the event flag	Available	Available
TA_CLR	The event flag is cleared to 0 when waiting state is canceled	×	Available

Note: * In the HI7000 series, the task wait queue is always managed in the same way as when TA_TFIFO is specified.

3.3.4 Clearing Specification for Event Flags

In the HI7000 series, clearing of an event flag is specified through the wait mode when an event-waiting system call is issued. In the HI7000/4 series, each event flag has a clearing specification attribute. Figure 3.5 shows how the clearing specification differs.

When shifting to the HI7000/4 series, check if one event flag is waited for both with and without clearing specifications; if both waiting modes are used for one event flag, consider the order of the clearing specifications.

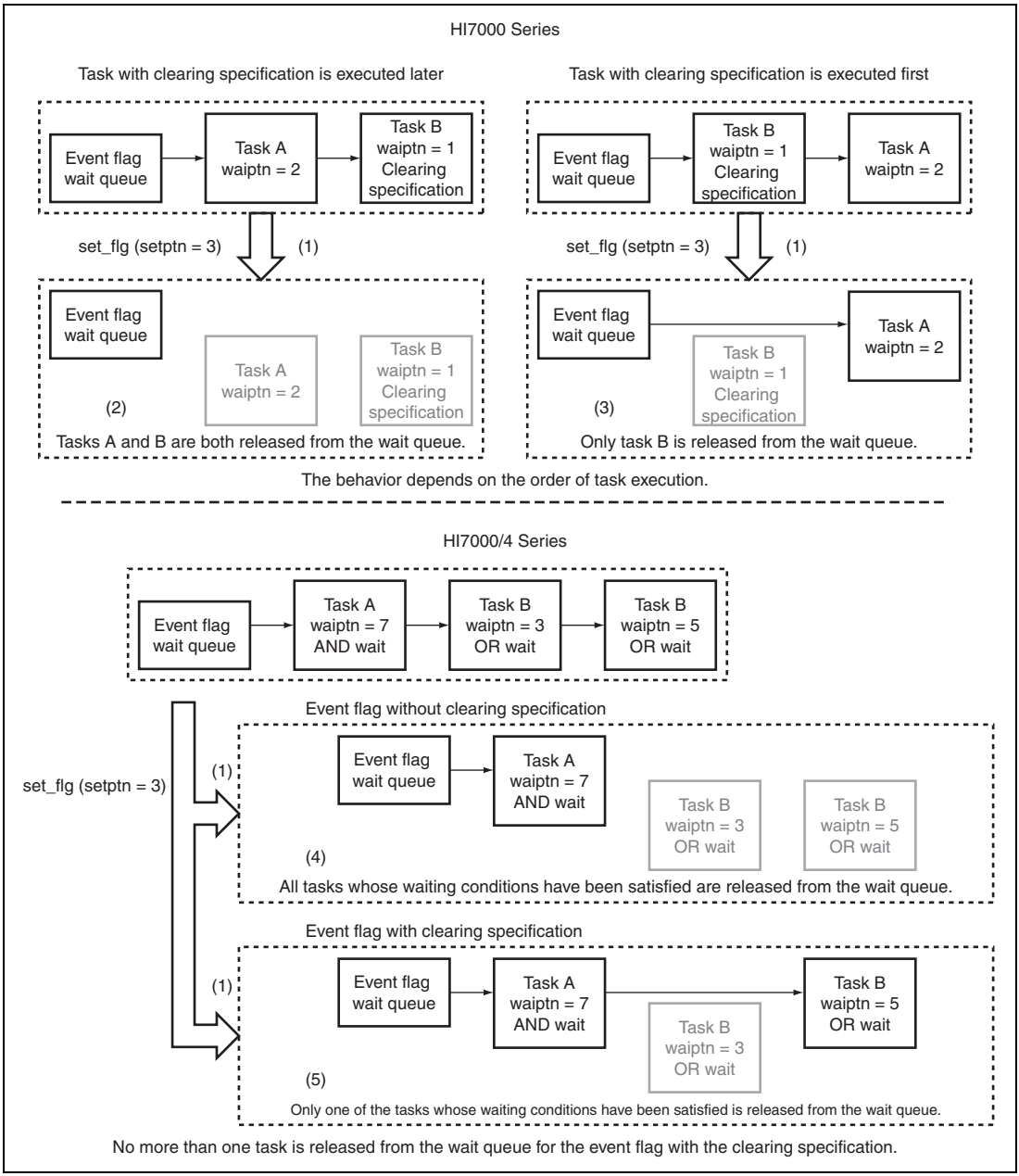

Figure 3.5 Differences in Event Flag Clearing Specification

1. Set bit pattern (3) in the event flag.
2. Task A is released from the wait queue by the bit pattern set in step 1 because it waits for bit pattern 2. As flag clearing is not specified, the next wait task is also checked.

   Task B is released from the wait queue by the bit pattern set in step 1 because it waits for bit pattern 1. As flag clearing is specified, processing ends.
3. Task B is released from the wait queue by the bit pattern set in step 1 because it waits for bit pattern 1. As flag clearing is specified in task B, the tasks behind task B in the wait queue are not checked and processing ends.
4. Task A remains in the wait queue because it waits for bit pattern 7 with the AND condition, which does not match the pattern set in step 1.

   Task B is released from the wait queue by the bit pattern set in step 1 because it waits for bit pattern 3 with the OR condition. As flag clearing is not specified for the event flag, all tasks in the wait queue are checked.

   Task C is released from the wait queue by the bit pattern set in step 1 because it waits for bit pattern 5 with the OR condition.
5. Task A remains in the wait queue because it waits for bit pattern 7 with the AND condition, which does not match the pattern set in step 1.

   Task B is released from the wait queue by the bit pattern set in step 1 because it waits for bit pattern 3 with the OR condition. As flag clearing is specified for the event flag, the tasks behind task B in the wait queue are not checked and processing ends.

3.3.5 New Attribute for Mailboxes

A new attribute (shaded in the table) can be used for mailboxes in the HI7000/4 series. Table 3.12 shows the mailbox attributes.

Table 3.12 Mailbox Attributes

Mailbox Attribute	Description	HI7000 Series	HI7000/4 Series
TA_TFIFO	Task queue waiting for receiving is managed on a FIFO basis	Available	Available
TA_TPRI	Task queue waiting for receiving is managed on priority	Available	Available
TA_MFIFO	Message queue is managed on a FIFO basis	Available	Available
TA_MPRI	Message queue is managed on priority	x	Available

3.3.6 New Message Header Type for Mailboxes

The HI7000 series has only one type of message header for mailboxes; the HI7000/4 series provides two types of message headers, the ordinary header and the header with priority. This new header type is added to implement the new management method shown in section 3.3.5. The application does not need to be modified for this new type when shifting to the HI7000/4 series.

A message header with priority should be used when the TA_MPRI attribute is specified.

T_MSG: Message header for a mailbox
T_MSG_PRI: Message header with priority for a mailbox

```
typedef struct  t_msg {
    VP      msghead;    /* Kernel management area              */
} T_MSG;
```

Figure 3.6 Message Header

```
typedef   struct  t_msg_pri {
   T_MSG  msgque;      /* Message header                       */
   PRI    msgpri;      /* Message header with priority         */
} T_MSG_PRI;
```

Figure 3.7 Message Header with Priority

The following shows sample messages.

```
typedef struct  user_msg {
   T_MSG  t_msg;       /* T_MSG structure                      */
   B      data[8] ;    /* Example of user message data
                          structure (any structure)            */
} USER_MSG;
```

Figure 3.8 Sample Message

```
typedef struct  user_msg {

   T_MSG_PRI  t_msg;    /* T_MSG_PRI structure                    */

   B          data[8];  /* Example of user message data
                           structure (any structure)              */

} USER_MSG;
```

Figure 3.9 Sample Message with Priority

3.3.7 Wait Queues for Message Buffers

The TA_TPRI attribute specifies the order of tasks waiting for a message buffer. In the HI7000 series, when this attribute is specified, the wait queue for "receiving" is managed on priority, but in the HI7000/4 series, the wait queue for "sending" is managed on priority. For details, see table 3.13.

Table 3.13 Difference in Wait Queue Managing Attribute for Message Buffer

Wait Queue for Message Buffer	HI7000 Series	HI7000/4 Series
Wait queue for receiving	TA_TFIFO: Managed on a FIFO basis TA_TPRI: Managed on priority	Always managed on a FIFO basis
Wait queue for sending	Always managed on a FIFO basis	TA_TFIFO: Managed on a FIFO basis TA_TPRI: Managed on priority

3.3.8 wup_tsk and sus_tsk

Parameter tskid = TSK_SELF (0) specifies the current task in the HI7000/4 series.

If the application executes processing when error E_ID occurs, the application must be modified.

Table 3.14 Difference in wup_tsk and sus_tsk

Parameter tskid	HI7000 Series	HI7000/4 Series
TSK_SELF(0)	Error E_ID	Current task specification

3.3.9 Timer Driver (Sample)

In the sample timer driver, no explicit vsys_clk (isig_tim) is necessary in the HI7000/4 series.

Table 3.15 shows the differences in the sample timer driver between the HI7000 series and HI7000/4 series.

Table 3.15 Differences in Sample Timer Driver

No.	Function	HI7000 Series	HI7000/4 Series
1	Timer driver initialization	The user must create a timer initialization routine and a timer interrupt handler that periodically issues vsys_clk and install them in the system.	By specifying CFG_TIMUSE in the configurator, _kernel_tmrini() is registered as the timer initialization routine and isig_tim is defined as the timer interrupt handler.
2	Timer initialization routine	Any name can be used.	The name is fixed to _kernel_tmrini().
3	Timer interrupt routine	Any name can be used.	The name is fixed to _kernel_tmrint().
4	Timer interrupt occurrence	The user-created timer interrupt handler is called. (The user must install the timer interrupt handler in the system.)	isig_tim is initiated as the interrupt handler and calls timer interrupt routine _kernel_tmrint().
5	System clock updating	vsys_clk must be issued explicitly in the timer interrupt handler.	isig_tim updates the system clock and explicit vsys_clk call is not necessary.

Table 3.16 Example of Sample Timer Driver Coding

Routine	HI7000 Series	HI7000/4 Series

Timer initialization routine	`void timer_init(void)`
	`{`
	`  /* Timer initialization */`
	`}`
		`void _kernel_tmrini(void)`
		`{`
		`  /* Timer initialization */`
		`}`

Timer interrupt handler	`void timer_handor(void)`
	`{`
	`  /* Initiate timer interrupt routine */`
	`  vsys_clk();`
	`}`
		`isig_tim()`
		Note: There is no need to create a timer interrupt handler because it is automatically registered by the configurator.

Timer interrupt routine	`void timer_rutine(void)`
	`{`
	`  /* Timer interrupt processing */`
	`}`
		`void _kernel_tmrint(void)`
		`{`
		`  /* Timer interrupt processing */`
		`}`

3.3.10 Time Management

The actual time for the time parameters and time management method are modified in the HI7000/4 series.

In the HI7000 series, the time unit for the time parameters is the interrupt cycle in the hardware timer.

In the HI7000/4 series, the time unit is fixed to 1 ms.

For details, see table 3.17.

Table 3.17 Time Unit for Time Parameters

No.	Modified Item	HI7000 Series	HI7000/4 Series
1	Time unit for time parameters	Hardware timer cycle	1 ms (fixed)
2	Time tick cycle	{ (1 / Q) × DIV} × (n + 1) [ms] Q: Clock frequency supplied to timer module DIV: Division ratio according to register setting in timer module n: Value specified in counter register in timer module	TIC_NUME/TIC_DENO [ms] TIC_NUME: Numerator of time tick cycle TIC_DENO: Denominator of time tick cycle Must be specified through the configurator. Either TIC_NUME or TIC_DENO must be set to 1.

Difference in Time Management:

In the HI7000 series, a timeout occurs when the system clock has passed the specified timeout value. The HI7000/4 series conforms to the µITRON4.0 specifications; that is, the system must guarantee that the specified time has elapsed.

See figure 3.10 for the difference in time management between the HI7000 series and HI7000/4 series.

For example, if the hardware timer cycle is the same as the time tick cycle, specify (timeout value set in HI7000 series - 1) as the timeout value in the HI7000/4 series to achieve the same actual time as in the HI7000 series. Calculate the timeout time in this way and modify the application.

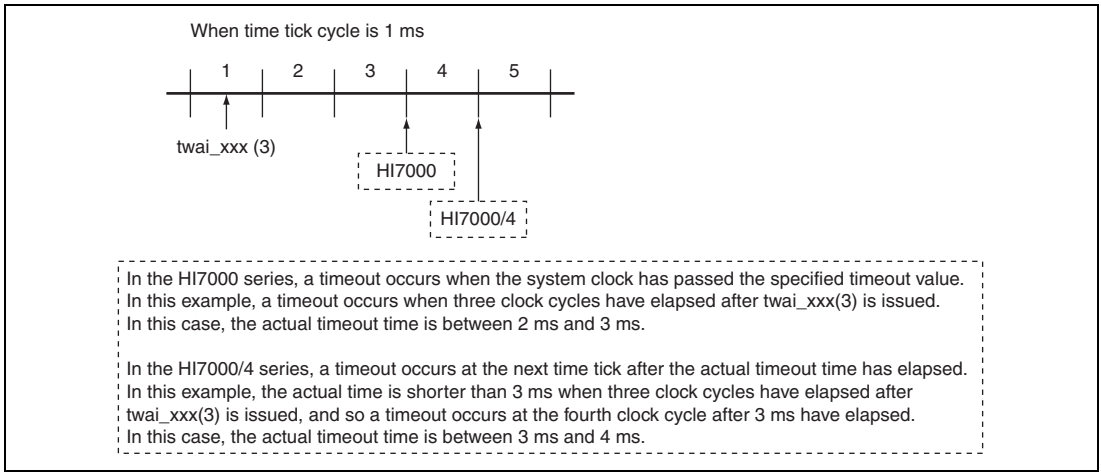

Figure 3.10 Difference in Time Management between HI7000 Series and HI7000/4 Series

3.3.11 Cyclic Handler

Modifications have been made throughout the cyclic handler function in the HI7000/4 series.

Modify the application by referring to the respective manuals, tables 3.18 and 3.19, and figure 3.11.

Table 3.18 Comparison of Cyclic Handler between HI7000 Series and HI7000/4 Series

Cyclic Handler in HI7000 Series		Cyclic Handler in HI7000/4 Series	
def_cyc, vasn_cyc	Define a cyclic handler Assign a cyclic handler number and define a cyclic handler	cre_cyc	Create a cyclic handler
		icre_cyc	
		acre_cyc	Create a cyclic handler (assign ID automatically)
		iacre_cyc	
act_cyc	Activate a cyclic handler	sta_cyc	Start a cyclic handler
		ista_cyc	
		stp_cyc	Stop a cyclic handler
		istp_cyc	
ref_cycff	Refer to the cyclic handler state	ref_cyc	Refer to the cyclic handler state
		iref_cyc	
ret_tmr	Return from the timer handler	—	Handler is terminated automatically
—	—	del_cyc	Delete a cyclic handler

Table 3.19 Settings for Cyclic Handler

Operation	Settings in HI7000 Series	Settings in HI7000/4 Series
Start a cyclic handler (storing initiation phase)	act_cyc: TCY_ON	sta_cyc (specifying TA_PHS in cre_cyc)
Start a cyclic handler (clearing initiation phase)	act_cyc: TCY_INI	sta_cyc (not specifying TA_PHS in cre_cyc)
Stop a cyclic handler	act_cyc: TCY_OFF	stp_cyc

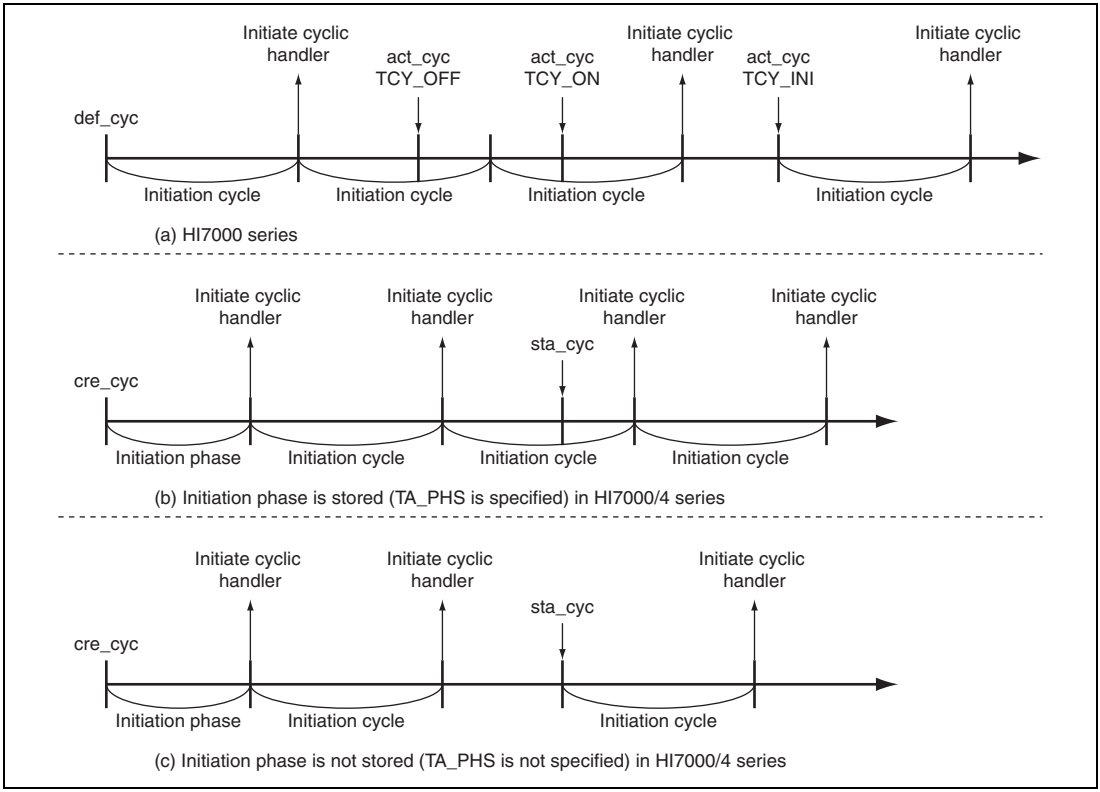

Figure 3.11 Differences in Cyclic Handler Operation

3.3.12 Alarm Handler

Modifications have been made throughout the alarm handler function in the HI7000/4 series.

Modify the application by referring to the respective manuals and table 3.20.

Table 3.20 Comparison of Alarm Handler between HI7000 Series and HI7000/4 Series

Alarm Handler in HI7000 Series		Alarm Handler in HI7000/4 series	
def_alm, vasn_alm	Define an alarm handler Assign an alarm handler number and define an alarm handler	cre_alm	Create an alarm handler
		icre_alm	
		acre_alm	Create an alarm handler (assign ID automatically)
		iacre_alm	
		sta_alm	Start an alarm handler
		ista_alm	
		stp_alm	Stop an alarm handler
		istp_alm	
ref_alm	Refer to the alarm handler state	ref_alm	Refer to the alarm handler state
		iref_alm	
ret_tmr	Return from the timer handler	—	Handler is terminated automatically
—	—	del_alm	Delete an alarm handler

3.3.13 TA_COP1 (FPU Bank 0) and TA_COP2 (FPU Bank 1) Attributes for SH-4

When ordinary floating-point operation is performed in the HI7750, the TA_COP1 attribute specifies bank 0 (FPSCR.FR = 0) or the TA_COP2 specifies bank 1 (FPSCR.FR = 1) when initiating a task or a handler.

When ordinary floating-point operation is performed in the HI7750/4, FPSCR must be initialized at the start of the entry function of the routine to specify bank 0 (FPSCR.FR = 0). The attribute is fixed to TA_COP1.

Table 3.21 Modification of TA_COP1 and TA_COP2 Attributes

	At Task Initiation	
Attribute	**HI7750**	**HI7750/4**
TA_COP1	Bank 0	Bank 0
TA_COP2	Bank 1	Cannot be specified

```
#include <machine.h>            /* Include machine.h to use intrinsic function set_fpscr()        */
#define INI_FPSCR 0x00080000    /* Initial FPSCR value (FR=0, PR=1, DN=0, SZ=0, RM=B'00) */
#pragma noregsave(Task)
void Task(VP_INT exinf)
{
    set_fpscr(INI_FPSCR);       /* Initialize FPSCR at the start of the task                      */
    /* Task processing */
    ext_tsk();}
```

Figure 3.12 Example of FPSCR Initialization in Task

3.3.14 Object Name

The object name function in the HI7000 series is deleted, and ID numbers cannot be acquired from the corresponding object names in the HI7000/4 series. As a substitute means for this function, configurator functions should be used in the HI7000/4 series.

In the HI7000/4 series, an ID name is specified through the automatic ID assignment function when an object is created by the configurator. That ID name can be used as an ID number. Table 3.22 shows this substitute means.

Table 3.22 Substitute Means for Object Names

No.	Function	HI7000 Series	HI7000/4 Series
1	Specify an object/ID name	At object creation	By the configurator *
2	Acquire an ID number	Acquired through the object name management service call	By including ID name header files (kernel_id.h and kernel_id_sys.h), ID names can be used as ID numbers

Note: * The range of ID name assignment depends on the product. For details, refer to the description of configuration in the respective manual.

3.3.15 System Down Routine

Some modifications are made to the system down routine in the HI7000/4 series in comparison with the HI7000 series.

Table 3.23 shows the modification in the routine name, and table 3.24 shows the comparison of information passed.

Table 3.23 Modification in System Down Routine Name

OS	System Down Routine Name (Fixed Name)
HI7000 series	void hi_sysdwn(W type, ER ercd, VW inf1, VW inf2)
HI7000/4 series	void _kernel_sysdwn(W type, ER ercd, VW inf1, VW inf2)

The number of system down causes is reduced in the HI7000/4 series in comparison with the HI7000 series.

The shaded rows in table 3.24 show the system down information in the HI7000/4 series. The same system down causes are shown under the same No.

Table 3.24 Information Passed to System Down Routine

			Parameters Passed to System Down Routine			
No.	OS	System Down Cause	Error Type W type (R4)	Error Code ER ercd (R5)	System Down Information 1 VW inf1 (R6)	System Down Information 2 VW inf2 (R7)
1	HI7000 series	System call vsys_dwn	1 to H'7fffffff	Parameters of system call vsys_dwn		
	HI7000/4 series	Service call vsys_dwn or ivsys_dwn	1 to H'7fffffff	Parameters of service call vsys_dwn or ivsys_dwn		
2	HI7000 series	An error in initial definition information in kernel build file or setup file	0	Error code of the system call for the initial definition	0: Error in kernel build file 1: Error in setup file	Indicates which initial definition information in kernel build file or setup file generated an error
	HI7000/4 series	An error in initial definition of configurator	0	Error code	0: Kernel side 1: Kernel environment side	Indicates the definition number where an error has occurred in the kernel side or kernel environment side *5
3	HI7000 series	A context error generated in system call ext_tsk issued in task-independent portion	H'ffffffff(-1)	E_CTX (h'ffffffbb)	Address calling ext_tsk + 2	Undefined
	HI7000/4 series	A context error generated in service call ext_tsk issued in non-task context	H'ffffffff(-1)	E_CTX (h'ffffffe7)	Address calling ext_tsk	Undefined

			Parameters Passed to System Down Routine			
No.	**OS**	**System Down Cause**	**Error Type W type (R4)**	**Error Code ER ercd (R5)**	**System Down Information 1 VW inf1 (R6)**	**System Down Information 2 VW inf2 (R7)**
4	HI7000 series	A context error generated in system call exd_tsk issued in task-independent portion	H'fffffffe (-2)	E_CTX (h'ffffffbb)	Address calling exd_tsk + 2	Undefined
	HI7000/4 series	A context error generated in service call exd_tsk issued in non-task context	H'fffffffe (-2)	E_CTX (h'ffffffe7)	Address calling exd_tsk	Undefined
5	HI7000 series	A context error generated in system call ret_int issued in task-independent portion*3	H'fffffffd (-3)	E_CTX (h'ffffffbb)	Address calling ret_int + 2	Undefined
6	HI7000 series	A context error generated in system call vsys_clk outside task-independent portion	H'fffffffc (-4)	E_CTX (h'ffffffbb)	Address calling vsys_clk + 2	Undefined
7	HI7000 series	A context error generated in system call ret_svc issued outside extended SVC handler and I/O handler	H'fffffffb (-5)	E_CTX (h'ffffffbb)	Address calling ret_svc + 2	Undefined
8	HI7000 series	A context error generated in system call vrst_tsk issued in task-independent portion or a parameter error generated in system call vrst_tsk	H'fffffff9 (-7)	E_CTX (h'ffffffbb) or E_PAR (h'ffffffdf)	Address calling vrst_tsk + 2	Undefined
9	HI7000 series	Undefined interrupt or exception generated	H'fffffff0 (-16)	Exception code *2 or vector number	PC at exception occurrence *1	SR at exception occurrence *1
	HI7000/4 series	Undefined interrupt or exception generated	H'fffffff0 (-16)	Exception code or vector number *4	PC at exception occurrence	SR at exception occurrence

Notes: 1. Information saved by the CPU in stack when an interrupt, exception, or unconditional trap occurs.

2. In the HI7700 and HI7750, an exception code is passed, which indicates the information set by the CPU in the interrupt event register (INTEVT or INTEVT2) or exception event register (EXPEVT) when an interrupt, exception, or unconditional trap occurs.

   In the HI7000, a vector number is passed.

   This value is –1 when NOTUSE is selected for macro hi_anavec in the kernel build file.

3. This error occurs only in the HI7000.

4. Information set by the CPU in the interrupt event register (INTEVT or INTEVT2) or exception event register (EXPEVT) when an interrupt, exception, or unconditional trap occurs in the HI7700/4 and HI7750/4.

5. The initial definition is first processed on the kernel side, then is processed on the kernel environment side. The order of initial definition on the kernel side and kernel environment side is shown below. The order in each process is the order of the list in each view. However, the system never goes down in steps (1) to (3) on the kernel side; they are not counted as the initial definition.

   (1) Initial definition of interrupt handlers in the interrupt and CPU exception handler view

   (2) Initial definition of CPU exception handlers in the interrupt and CPU exception handler view

   (3) Initial definition of CPU exception handlers for trap in the trap exception handler view

   (4) Initial definition of tasks and task exception processing routines in the task view

   (5) Initial definition of semaphores in the semaphore view

   (6) Initial definition of event flags in the event flag view

   (7) Initial definition of data queues in the data queue view

   (8) Initial definition of mailboxes in the mailbox view

   (9) Initial definition of mutexes in the mutex view

   (10) Initial definition of message buffers in the message buffer view

   (11) Initial definition of fixed-size memory pools in the fixed-size memory pool view

   (12) Initial definition of variable-size memory pools in the variable-size memory pool view

   (13) Initial definition of cyclic handlers in the cyclic handler view

   (14) Initial definition of alarm handlers in the alarm handler view

   (15) Initial definition of the overrun handler in the overrun handler view

   (16) Initial definition of extended service calls in the extended service call view

3.3.16 Coding Examples

(1) Task

Figures 3.13 and 3.14 show examples of task coding.

```
#include “itron.h”
#pragma  noregsave(task)          ← #pragma noregsave can be used because
                                    the task function does not return a value.
void  Task(INT stacd)
{
    /* Task processing */         ← Be sure to issue an ext_tsk or exd_tsk system call
    ext_tsk();                      at the end of the task.
}
```

Figure 3.13 Example of Task Coding in HI7000 Series

```
#include “itron.h”
#include “kernel.h”
#include “hkernel_id.h”

#pragma noregsave(Task)           ← #pragma noregsave can be used because the task function
                                    does not need to guarantee register contents.
void Task(VP_INT exinf)           ← When a task is initiated by TA_ACT attribute or act_tsk,
                                    exinf specified at task creation is passed as a parameter;
                                    when a task is initiated by sta_tsk, stacd specified by sta_tsk
{                                   is passed as a parameter.
    /* Task processing */         ← Be sure to issue an ext_tsk or exd_tsk service call at the end
    ext_tsk();                      of the task.
}                                 ← ext_tsk is automatically issued at the end of the function.
```

Figure 3.14 Example of Task Coding in HI7000/4 Series

(2) Extended Service Call (Extended SVC)

Figures 3.15 and 3.16 show examples of extended service call routine coding.

```
#include “itron.h”
#pragma noregsave(ExHandler)        ← #pragma noregsave can be used because the extended SVC
                                      handler function does not return a value.
void ExHandler(ID reqid,            ← Parameters specified in the extended SVC are passed to
UW par1)                              the extended SVC handler.
{
    /* Extended SVC handler
       processing           */      ← Be sure to issue a ret_svc system call at the end of
    ret_svc(error_code);              the extended SVC handler. The parameter of ret_svc is
}                                     passed to the caller of the extended SVC as an error code.
```

Figure 3.15 Example of Extended SVC Coding in HI7000 Series

```
#include “itron.h”
#include “kernel.h”
#include “hkernel_id.h”

ER_UINT Svcrtn(VP_INT par1,         ← Parameters specified in cal_svc are passed to the extended
VP_INT par2)                          service call routine.
{
    /* Extended service call       ← A return value is passed to the caller.
       routine             */
    return E_OK;
}
```

Figure 3.16 Example of Extended Service Call Routine Coding in HI7000/4 Series

(3) Interrupt Handler

Figures 3.17 to 3.19 show examples of interrupt handler coding.

```
#include “itron.h”
#define stksz 512                              ← (1)
VW  stk[stksz / sizeof(VW)];
static const VP
    p_stk=(VP)&stk[stksz/sizeof(VW)];          ← (2)

 #pragma interrupt(Isr(sp=p_stk,tn=25))        ← (3)
/* #pragma interrupt(Isr(sp=p_stk)) */

void Isr(void)                                 ← (4)
{
    /* Interrupt handler processing */
}                                              ← (5)
```

Figure 3.17 Example of Interrupt Handler Coding in HI7000 Series

Description:

(1) The stack size used by the interrupt handler is defined.
Interrupt handlers of the same interrupt level can share a stack.

(2) The initial value of the stack pointer is defined as a const type.

(3) The #pragma interrupt directive declares the interrupt function to be used as an interrupt handler. The following interrupt specifications are necessary:

— Stack switching specification (sp=)

— Trap return specification (tn=25)

The interrupt handler having a level equal to or lower than the kernel interrupt mask level must be terminated by the ret_int system call, and therefore tn=25 must be specified in this case. The interrupt handler having a higher level than the kernel interrupt mask level must be terminated by the RTE instruction, and therefore do not use trap return specification in this case.

(4) Write the interrupt handler as a void-type function.

(5) As #pragma interrupt is used, the ret_int system call is automatically issued.

```
#include “itron.h”
#include “kernel.h”
#include “hkernel_id.h”

#pragma noregsave(Inh)                          ← (1)

void Inh(void)                                  ← (2)
{
/* Interrupt handler processing */
}
```

Figure 3.18 Example of Ordinary Interrupt Handler Coding in HI7000/4 Series

Description:

(1) When a bank interrupt is used in the SH-2A or SH2A-FPU (CFG_REGBANK is checked), #pragma noregsave can be used because the handler does not have to guarantee registers.

(2) Write the interrupt handler as a void-type function.

```
#include “itron.h”
#include “kernel.h”
#include “hkernel_id.h”

#define stksz 512                                       ← (1)
VW  stk[stksz / sizeof(VW)];
static const VP  p_stk=(VP)&stk[stksz/sizeof(VW)];      ← (2)
#pragma interrupt(DirectInh(sp=p_stk,tn=25))            ← (3)
void DirectInh(void)                                    ← (4)
{
    /* Interrupt handler processing */
}
```

Figure 3.19 Example of Direct Interrupt Handler Coding in HI7000/4 Series

Description:

(1) The stack size used by the interrupt handler is defined.

The handler except NMI must switch stacks in order to avoid overflow of the interrupted program's stack. Interrupt handlers of the same interrupt level can share a stack. Note, however, that NMI interrupt handler cannot have its dedicated stack.

(2) The initial value of the stack pointer is defined as a const type.

(3) The #pragma interrupt directive declares the interrupt function to be used as an interrupt handler. The following interrupt specifications are necessary:

(a) "sp=" specification (stack switching)

Specify the variable defined in (2) when the stack is switched.

(b) "tn=" specification (trap return specification)

The interrupt handler having a higher level than the kernel interrupt mask level must be terminated by the RTE instruction, and therefore do not use trap return specification in this case.

For the interrupt handler having a level equal to or lower than the kernel interrupt mask level, specify "tn=25".

(c) "resbank" specification (bank interrupt)

When a bank interrupt is used in the SH-2A or SH2A-FPU (CFG_REGBANK is checked), "resbank" must be specified.

However, for NMI, "resbank" must not be specified.

(4) Write the interrupt function as a void-type function.

Section 4 Service Call Differences

4.1 SVC Differences

In the following tables, the shaded rows indicate the changed functions.
The following terms are used in these tables.
Prm: Parameter R_Prm: Return parameter struct: Structure

4.1.1 Task Management

	HI7000 Series (µITRON 3.0)		HI7000/4 Series (µITRON 4.0)		
	Function		Function		
	SVC	API	SVC	API	Change
1	Create task (with dynamic stack)		Create task (with dynamic stack)		Packet struct changed
	cre_tsk	ER ercd = cre_tsk (ID tskid, T_CTSK *pk_ctsk);	cre_tsk icre_tsk	ER ercd = cre_tsk (ID tskid, T_CTSK *pk_ctsk);	
		Packet struct typedef struct t_ctsk {		Packet struct typedef struct t_ctsk {	
		VP exinf; Extended information		ATR tskatr; Task attribute	
		ATR tskatr; Task attribute		VP_INT exinf; Extended information	
		FP task; Task start address		FP task; Task start address	
		PRI itskpri; Priority at task initiation		PRI itskpri; Priority at task initiation	
		UB cpumode; Task CPU mode		SIZE stksz; Task stack size	
		INT stksz; Task stack size		VP stk; Start address of task stack area	
		TMO quantum; Time slice value		} T_CTSK;	
		B *name; Start address of area storing the name			
		} T_CTSK;			
2	Create task (with static stack)		—		Deleted
	vcre_tsk	ER ercd = vcre_tsk (ID tskid, TASKP stadr, TPRI itskpri);	Not available		

	HI7000 Series (µITRON 3.0)				HI7000/4 Series (µITRON 4.0)				
	Function				Function				
	SVC	API			SVC	API			Change
3	Create task (with static stack)				Create task (with static stack)				Packet struct changed
	vscr_tsk	ER ercd = vscr_tsk (ID tskid, T_CTSK *pk_ctsk);			vscr_tsk ivscr_tsk	ER ercd = vscr_tsk (ID tskid, T_CTSK *pk_ctsk); ER ercd = ivscr_tsk (ID tskid, T_CTSK *pk_ctsk);			
		Packet struct	typedef struct t_ctsk {			Packet struct	typedef struct t_ctsk {		
			VP	exinf;	Extended information		ATR	tskatr;	Task attribute
			ATR	tskatr;	Task attribute		VP_INT	exinf;	Extended information
			FP	task;	Task start address		FP	task;	Task start address
			PRI	itskpri;	Priority at task initiation		PRI	itskpri;	Priority at task initiation
			UB	cpumode;	Task CPU mode		SIZE	stksz;	Task stack size
			INT	stksz;	Task stack size		VP	stk;	Start address of task stack area
			TMO	quantum;	Time slice value		} T_CTSK;		
			B	*name;	Start address of area storing the name				
		} T_CTSK;							
4	Create task (assign task ID automatically) (with dynamic stack)				Create task (assign task ID automatically) (with dynamic stack)				SVC name, Prm, R_Prm, and packet struct changed
	vasn_tsk	ER ercd = vasn_tsk (ID *p_tskid, T_CTSK *pk_ctsk);			acre_tsk iacre_tsk	ER_ID tskid = acre_tsk (T_CTSK *pk_ctsk); ER_ID tskid = iacre_tsk (T_CTSK *pk_ctsk);			
		Prm	ID	*p_tskid;	Start address of area where created task ID is to be returned	Prm	T_CTSK	*pk_ctsk;	Pointer to packet storing task creation information
			T_CTSK	*pk_ctsk;	Start address of task creation information				
		R_Prm	ER	ercd;	Error code	R_Prm	ER_ID	tskid;	Created task ID (a positive value) or error code
			ID	*p_tskid;	Start address of area storing created task ID				
		Packet struct	typedef struct t_ctsk {			Packet struct	typedef struct t_ctsk {		
			VP	exinf;	Extended information		ATR	tskatr;	Task attribute
			ATR	tskatr;	Task attribute		VP_INT	exinf;	Extended information
			FP	task;	Task start address		FP	task;	Task start address
			PRI	itskpri;	Priority at task initiation		PRI	itskpri;	Priority at task initiation
			UB	cpumode;	Task CPU mode		SIZE	stksz;	Task stack size
			INT	stksz;	Task stack size		VP	stk;	Start address of task stack area
			TMO	quantum;	Time slice value		} T_CTSK;		
			B	*name;	Start address of area storing the name				
		} T_CTSK;							
5	Delete task				Delete task				
	del_tsk	ER ercd = del_tsk (ID tskid);			del_tsk	ER ercd = del_tsk (ID tskid);			

	HI7000 Series (µITRON 3.0)		HI7000/4 Series (µITRON 4.0)		
	Function		Function		
	SVC	API	SVC	API	Change
6	Start task		Start task		Prm type changed
	sta_tsk	ER ercd = sta_tsk (ID tskid, INT stacd);	sta_tsk ista_tsk	ER ercd = sta_tsk (ID tskid, VP_INT stacd); ER ercd = ista_tsk (ID tskid, VP_INT stacd);	
		Prm INT stacd Task start code		Prm VP_INT stacd Task start code	
—	—	—	Initiate task		New function
	—	Not available	act_tsk iact_tsk	ER ercd = act_tsk (ID tskid); ER ercd = iact_tsk (ID tskid);	
—	—	—	Cancel task initiation request		New function
	—	Not available	can_act ican_act	ER_UINT actcnt = can_act (ID tskid); ER_UINT actcnt = ican_act (ID tskid);	
7	Restart current task		—		Deleted. See 3.1.4.
	vrst_tsk	void vrst_tsk (FP task, INT stacd);	—	Not available	
8	Exit current task		Exit current task		
	ext_tsk	void ext_tsk(void);	ext_tsk	void ext_tsk();	
9	Exit and delete current task		Exit and delete current task		
	exd_tsk	void exd_tsk(void);	exd_tsk	void exd_tsk();	
10	Terminate task		Terminate task		
	ter_tsk	ER ercd = ter_tsk (ID tskid);	ter_tsk	ER ercd = ter_tsk (ID tskid);	
11	Disable task dispatch		Disable task dispatch		Meaning of "dispatch" changed. See 3.3.1.
	dis_dsp	ER ercd = dis_dsp(void);	dis_dsp	ER ercd = dis_dsp();	
12	Enable task dispatch		Enable task dispatch		Meaning of "dispatch" changed. See 3.3.1.
	ena_dsp	ER ercd = ena_dsp(void);	ena_dsp	ER ercd = ena_dsp();	
13	Change task priority		Change task priority		
	chg_pri	ER ercd = chg_pri (ID tskid, PRI tskpri);	chg_pri ichg_pri	ER ercd = chg_pri (ID tskid, PRI tskpri); ER ercd = ichg_pri (ID tskid, PRI tskpri);	
—	—	—	Refer to task priority		New function
	—	Not available	get_pri iget_pri	ER ercd = get_pri (ID tskid, PRI *p_tskpri); ER ercd = iget_pri (ID tskid, PRI *p_tskpri);	
14	Rotate ready queue		Rotate ready queue		
	rot_rdq	ER ercd = rot_rdq (PRI tskpri);	rot_rdq irot_rdq	ER ercd = rot_rdq (PRI tskpri); ER ercd = irot_rdq (PRI tskpri);	
15	Cancel WAITING state forcibly		Cancel WAITING state forcibly		
	rel_wai	ER ercd = rel_wai (ID tskid);	rel_wai irel_wai	ER ercd = rel_wai (ID tskid); ER ercd = irel_wai (ID tskid);	
16	Get current task ID		Get task ID in RUNNING state		Operation changed in a non-task context
	get_tid	ER ercd = get_tid (ID *p_tskid);	get_tid iget_tid	ER ercd = get_tid (ID *p_tskid); ER ercd = iget_tid (ID *p_tskid);	
		Operation Returns FALSE(0) as the task ID if the service call is issued in a task-independent portion.		Operation Returns the ID of the task that is being executed if the service call is issued in a non-task context.	

	HI7000 Series (µITRON 3.0)				HI7000/4 Series (µITRON 4.0)				
	Function				Function				
	SVC	API			SVC	API			Change
17	Refer to task state				Refer to task state				Prm sequence and packet struct changed
	ref_tsk	ER ercd = ref_tsk (T_RTSK *pk_rtsk, ID tskid);			ref_tsk iref_tsk	ER ercd = ref_tsk (ID tskid, T_RTSK *pk_rtsk); ER ercd = iref_tsk (ID tskid, T_RTSK *pk_rtsk);			
		Packet struct typedef struct t_rtsk {				Packet struct typedef struct t_rtsk {			
		VP	exinf;	Extended information		STAT	tskstat;	Task state	
		PR	tskpri;	Current task priority		PRI	tskpri;	Current priority of task	
		UINT	tskstat;	Task state		PRI	tskbpri;	Base priority of task	
		UINT	tskwait;	Wait cause		STAT	tskwait;	Wait cause	
		ID	wid;	Wait object ID		ID	wobjid;	Wait object ID	
		H	wupcnt;	Number of wakeup requests		TMO	lefttmo;	Time to timeout	
		H	suscnt;	Number of suspension requests		UINT	actcnt;	Number of queued initiation requests	
		UH	tskmode;	Task execution mode		UINT	wupcnt;	Number of queued wakeup requests	
		UH	pndptn;	Whether a request is pended		UINT	suscnt;	Suspend request nest count	
		FN	s_fncd;	Current extended SVC number being executed		UINT	tskmode;	Task execution mode	
		DVN	iodvn;	Current I/O number being executed		UINT	tflptn;	Current task event flag value	
		INT	remstksz;	Remaining stack size		} T_CTSK;			
		UINT	tflptn;	Current task event flag value					
		FP	task;	Task start address					
		PRI	itskpri;	Task priority at initiation					
		UB	cpumode;	Task CPU mode					
		TMO	quantum;	Time slice value					
		B	*name;	Start address of area where the name is to be returned					
		} T_RTSK;							
18	Refer to task state				—				Deleted
	vtsk_sts	ER ercd = vtsk_sts (UH *p_tskstat, TPRI *p_tskpri, ID tskid);			—	Not available			
—	—				Refer to task state (simple version)				New function
	—	Not available			ref_tst	ER_ercd = ref_tst (ID tskid, T_RTST *pk_rtst); ER_ercd = iref_tst (ID tskid, T_RTST *pk_rtst);			

	HI7000 Series (µITRON 3.0)				HI7000/4 Series (µITRON 4.0)				
	Function				Function				
	SVC	API			SVC	API			Change
19	Set task execution mode				Change task execution mode				Integrated into vchg_tmd
	vset_tmd	ER ercd = vset_tmd (UINT settmd);			vchg_tmd	ER ercd = vchg_tmd (UINT tmd);			
		Prm	UINT	settmd; Task execution mode to set		Prm	UINT	tmd; Task execution mode to change	
		R_Prm	ER	ercd; Error code		R_Prm	ER	ercd; Normal termination (E_OK) or error code	
20	Clear task execution mode								
	vclr_tmd	ER ercd = vclr_tmd (UINT clrtmd);							
		Prm	UINT	clrtmd; Task execution mode to clear					
		R_Prm	ER	ercd; Error code					

4.1.2 Task Synchronization

	HI7000 Series (µTRON 3.0)					HI7000/4 Series (µTRON 4.0)					
	Function					Function					
	SVC	API				SVC	API				Change
1	Suspend task					Suspend task					Current task can be specified. See 3.3.8.
	sus_tsk	ER ercd = sus_tsk (ID tskid);				sus_tsk isus_tsk	ER ercd = sus_tsk (ID tskid); ER ercd = isus_tsk (ID tskid);				
		Operation	Current task cannot be specified.				Operation	Current task can be specified through tskid = 0.			
2	Resume task					Resume task					
	rsm_tsk	ER ercd = rsm_tsk (ID tskid);				rsm_tsk irsm_tsk	ER ercd = rsm_tsk (ID tskid); ER ercd = irsm_tsk (ID tskid);				
3	Resume task forcibly					Resume task forcibly					
	frsm_tsk	ER ercd = frsm_tsk (ID tskid);				frsm_tsk ifrsm_tsk	ER ercd = frsm_tsk (ID tskid); ER ercd = ifrsm_tsk (ID tskid);				
4	Sleep task					Sleep task					
	slp_tsk	ER ercd = slp_tsk(void);				slp_tsk	ER ercd = slp_tsk();				
5	Sleep task with timeout					Sleep task with timeout					
	tslp_tsk	ER ercd = tslp_tsk (TMO tmout);				tslp_tsk	ER ercd = tslp_tsk (TMO tmout);				
6	Wake up task					Wake up task					Current task can be specified. See 3.3.8.
	wup_tsk	ER ercd = wup_tsk (ID tskid);				wup_tsk iwup_tsk	ER ercd = wup_tsk (ID tskid); ER ercd = iwup_tsk (ID tskid);				
		Operation	Current task cannot be specified.				Operation	Current task can be specified through tskid = 0.			
7	Cancel wakeup request					Cancel wakeup request					Prm and R_Prm changed
	can_wup	ER ercd = can_wup (INT *p_wupcnt, ID tskid);				can_wup ican_wup	ER_UINT wupcnt = can_wup (ID tskid); ER_UINT wupcnt = ican_wup (ID tskid);				
		Prm	INT	*p_wupcnt;	Start address of area where the number of wakeup requests is to be returned		Prm	ID	tskid;	Task ID	
			ID	tskid;	Task ID						
		R_Prm	ER	ercd;	Error code		R_Prm	ER_UINT	wupcnt;	Number of queued task wakeup requests (0 or a positive value) or error code	
			INT	*p_wupcnt;	Start address of area where the number of wakeup requests is returned						
8	Set task event flag					Set task event flag					
	vset_tfl	ER ercd = vset_tfl (ID tskid, UINT setptn);				vset_tfl ivset_tfl	ER ercd = vset_tfl (ID tskid, UINT setptn); ER ercd = ivset_tfl (ID tskid, UINT setptn);				
9	Clear task event flag					Clear task event flag					
	vclr_tfl	ER ercd = vclr_tfl (ID tskid, UINT clrptn);				vclr_tfl ivclr_tfl	ER ercd = vclr_tfl (ID tskid, UINT clrptn); ER ercd = ivclr_tfl (ID tskid, UINT clrptn);				
10	Wait for task event flag					Wait for task event flag					
	vwai_tfl	ER ercd = vwai_tfl (UINT *p_tflptn, UINT waiptn);				vwai_tfl	ER ercd = vwai_tfl (UINT waiptn, UINT *p_tflptn);				
11	Wait for task event flag (polling)					Wait for task event flag (polling)					
	vpol_tfl	ER ercd = vpol_tfl (UINT *p_tflptn, UINT waiptn);				vpol_tfl	ER ercd = vpol_tfl (UINT waiptn, UINT *p_tflptn);				
12	Wait for task event flag with timeout					Wait for task event flag with timeout					
	vtwai_tfl	ER ercd = vtwai_tfl (UINT *p_tflptn, UINT waiptn, TMO tmout);				vtwai_tfl	ER ercd = vtwai_tfl (UINT waiptn, UINT *p_tflptn, TMO tmout);				

4.1.3 Task Exception Processing

HI7000 Series (µITRON 3.0)			HI7000/4 Series (µITRON 4.0)		
Function			Function		
SVC	API		SVC	API	Change
—	—		Define task exception processing routine		New function
—	Not available		def_tex idef_tex	ER ercd = def_tex (ID tskid, T_DTEX *pk_dtex); ER ercd = idef_tex (ID tskid, T_DTEX *pk_dtex);	
—	—		Request task exception processing		New function
—	Not available		ras_tex iras_tex	ER ercd = ras_tex (ID tskid, TEXPTN rasptn); ER ercd = iras_tex (ID tskid, TEXPTN rasptn);	
—	—		Disable task exception processing		New function
—	Not available		dis_tex	ER ercd = dis_tex();	
—	—		Enable task exception processing		New function
—	Not available		ena_tex	ER ercd = ena_tex();	
—	—		Refer to task exception processing disabled state		New function
—	Not available		sns_tex	BOOL state = sns_tex();	
—	—		Refer to task exception processing state		New function
—	Not available		ref_tex iref_tex	ER ercd = ref_tex (ID tskid, T_RTEX *pk_rtex); ER ercd = iref_tex (ID tskid, T_RTEX *pk_rtex);	

4.1.4 Synchronization and Communication (Semaphore)

	HI7000 Series (µITRON 3.0)					HI7000/4 Series (µITRON 4.0)					
	Function					Function					
	SVC	API				SVC	API				Change
1	Create semaphore					Create semaphore					Packet struct changed
	cre_sem	ER ercd = cre_sem (ID semid, T_CSEM *pk_csem);				cre_sem icre_sem	ER ercd = cre_sem (ID semid, T_CSEM *pk_csem); ER ercd = icre_sem (ID semid, T_CSEM *pk_csem);				
		Packet struct	typedef struct t_csem {				Packet struct	typedef struct t_csem {			
			VP	exinf;	Extended information			ATR	sematr;	Semaphore attribute	
			ATR	sematr;	Semaphore attribute			UINT	isemcnt;	Initial value of semaphore resource count	
			INT	isemcnt;	Initial value of semaphore			UINT	maxsem;	Maximum number of semaphore resources	
			B	*name;	Start address of area storing the name			} T_CSEM;			
			} T_CSEM;								
2	Assign semaphore ID and create semaphore					Assign semaphore ID and create semaphore					API name, Prm, R_Prm, and packet struct changed
	vasn_sem	ER ercd = vasn_sem (ID *p_semid, T_CSEM *pk_csem);				acre_sem iacre_sem	ER_ID semid = acre_sem (T_CSEM *pk_csem); ER_ID semid = iacre_sem (T_CSEM *pk_csem);				
		Prm	ID	*p_semid;	Start address of area where created semaphore ID is to be returned		Prm	T_CSEM	*pk_csem;	Pointer to packet storing semaphore creation information	
			T_CSEM	*pk_csem;	Start address of semaphore creation information						
		R_Prm	ER	ercd;	Error code		R_Prm	ER_ID	ercd	Created semaphore ID (a positive value) or error code	
			ID	*p_semid;	Start address of area storing created semaphore ID						
		Packet	typedef struct t_csem {				Packet	typedef struct t_csem {			
			VP	exinf;	Extended information			ATR	sematr;	Semaphore attribute	
			ATR	sematr;	Semaphore attribute			UINT	isemcnt;	Initial value of semaphore resource count	
			INT	isemcnt;	Initial value of semaphore			UINT	maxsem;	Maximum number of semaphore resources	
			B	*name;	Start address of area storing the name			} T_CSEM;			
			} T_CSEM;								
3	Delete semaphore					Delete semaphore					
	del_sem	ER ercd = del_sem (ID semid);				del_sem	ER ercd = del_sem (ID semid);				
4	Return semaphore resource					Return semaphore resource					
	sig_sem	ER ercd = sig_sem (ID semid);				sig_sem isig_sem	ER ercd = sig_sem (ID semid); ER ercd = isig_sem (ID semid);				
5	Wait for semaphore resource					Wait for semaphore resource					
	wai_sem	ER ercd = wai_sem (ID semid);				wai_sem	ER ercd = wai_sem (ID semid);				

	HI7000 Series (µITRON 3.0)					HI7000/4 Series (µITRON 4.0)					
	Function					Function					
	SVC	API				SVC	API				Change
6	Wait for semaphore resource (polling)					Wait for semaphore resource (polling)					SVC name changed
	preq_sem	ER ercd = preq_sem (ID semid);				pol_sem ipol_sem	ER ercd = pol_sem (ID semid); ER ercd = ipol_sem (ID semid);				
7	Wait for semaphore resource with timeout					Wait for semaphore resource with timeout					
	twai_sem	ER ercd = twai_sem (ID semid, TMO tmout);				twai_sem	ER ercd = twai_sem (ID semid, TMO tmout);				
8	Refer to semaphore state					Refer to semaphore state					Prm and packet struct changed
	ref_sem	ER ercd = ref_sem (T_RSEM *pk_rsem, ID semid);				ref_sem iref_sem	ER ercd = ref_sem (ID semid, T_RSEM *pk_rsem); ER ercd = iref_sem (ID semid, T_RSEM *pk_rsem);				
		Prm	T_RSEM	*pk_rsem;	Start address of area where semaphore state is to be returned		Prm	ID	semid;	Semaphore ID	
			ID	semid;	Semaphore ID			T_RSEM	*pk_rsem;	Pointer to packet where semaphore state is to be returned	
		Packet struct	typedef struct t_rsem {				Packet struct	typedef struct t_rsem {			
			VP	exinf;	Extended information			ID	wtskid;	Wait task ID	
			BOOL_ID	wtsk;	Wait task ID			UINT	semcnt;	Current semaphore count value	
			INT	semcnt;	Current semaphore count value			} T_RSEM;			
			ATR	sematr;	Semaphore attribute						
			B	*name;	Start address of area where the name is to be returned						
			} T_RSEM;								
9	Refer to semaphore state					—					Deleted
	vsem_sts	ER ercd = vsem_sts (ID *p_wtskid, W *p_semcnt, ID semid);				—	Not available				

4.1.5 Synchronization and Communication (Event Flag)

	HI7000 Series (µITRON 3.0)				HI7000/4 Series (µITRON 4.0)				
	Function				Function				
	SVC	API			SVC	API			Change
1	Create event flag				Create event flag				Packet struct changed. Clearing specification changed. See 3.3.3 and 3.3.4.
	cre_flg	ER ercd = cre_flg (ID flgid, T_CFLG *pk_cflg);			cre_flg icre_flg	ER ercd = cre_flg (ID flgid, T_CFLG *pk_cflg); ER ercd = icre_flg (ID flgid, T_CFLG *pk_cflg);			
		Packet struct typedef struct t_cflg {				Packet struct typedef struct t_cflg {			
		VP	exinf;	Extended information		ATR	flgatr;	Event flag attribute	
		ATR	flgatr;	Event flag attribute		FLGPTN	iflgptn;	Initial value of event flag	
		UINT	iflgptn;	Initial value of event flag		} T_CFLG;			
		B	*name;	Start address of area storing the name					
		} T_CFLG;							
2	Assign event flag ID and create event flag				Create event flag (assign event flag ID automatically)				SVC name, Prm, R_Prm, and packet struct changed. Clearing specification changed. See 3.3.3 and 3.3.4.
	vasn_flg	ER ercd = vasn_flg (ID *p_flgid, T_CFLG *pk_cflg);			acre_flg iacre_flg	ER_ID flgid= acre_flg (T_CFLG *pk_cflg); ER_ID flgid= iacre_flg (T_CFLG *pk_cflg);			
		Prm	ID	*p_flgid; Start address of area where created event flag ID is to be returned		Prm	T_CFLG	*pk_cflg	Pointer to packet storing event flag creation information
			T_CFLG	*pk_cflg; Start address of event flag creation information					
		R_Prm	ER	ercd; Error code		R_Prm	ER_ID	flgid	Created event flag ID (a positive value) or error code
			ID	*p_flgid; Start address of area storing created event flag ID					
		Packet struct typedef struct t_cflg {				Packet struct typedef struct t_cflg {			
		VP	exinf;	Extended information		ATR	flgatr;	Event flag attribute	
		ATR	flgatr;	Event flag attribute		FLGPTN	iflgptn;	Initial value of event flag	
		UINT	iflgptn;	Initial value of event flag		} T_CFLG;			
		B	*name;	Start address of area storing the name					
		} T_CFLG;							
3	Delete event flag				Delete event flag				
	del_flg	ER ercd = del_flg (ID flgid);			del_flg	ER ercd = del_flg (ID flgid);			
4	Set event flag				Set event flag				Prm type changed
	set_flg	ER ercd = set_flg (ID flgid, UINT setptn);			set_flg iset_flg	ER ercd = set_flg (ID flgid, FLGPTN setptn); ER ercd = iset_flg (ID flgid, FLGPTN setptn);			
		Prm	UINT	setptn; Bit pattern to set		Prm	FLGPTN	setptn;	Bit pattern to set

	HI7000 Series (µITRON 3.0)				HI7000/4 Series (µITRON 4.0)				
	Function				Function				
	SVC	API			SVC	API			Change
5	Clear event flag				Clear event flag				Prm type changed
	clr_flg	ER ercd = clr_flg (ID flgid, UINT clrptn);			clr_flg iclr_flg	ER ercd = clr_flg (ID flgid, FLGPTN clrptn); ER ercd = iclr_flg (ID flgid, FLGPTN clrptn);			
		Prm	UINT	clrptn; Bit pattern to be cleared		Prm	UINT	clrptn; Bit pattern to be cleared	
6	Wait for event flag				Wait for event flag				Prm type changed. Clearing specification changed. See 3.3.3 and 3.3.4.
	wai_flg	ER ercd = wai_flg (UINT *p_flgptn, ID flgid, UINT waiptn, UINT wfmode);			wai_flg	ER ercd = wai_flg (ID flgid, FLGPTN waiptn, MODE wfmode, FLGPTN *p_flgptn);			
		Prm	UINT	*p_flgptn; Start address of area where the bit pattern at waiting release is to be returned		Prm	FLGPTN	*p_flgptn; Pointer to area where the bit pattern at waiting release is to be returned	
7	Wait for event flag (polling)				Wait for event flag (polling)				Prm sequence and Prm type changed. Clearing specification changed. See 3.3.3 and 3.3.4.
	pol_flg	ER ercd = pol_flg (UINT *p_flgptn, ID flgid, UINT waiptn, UINT wfmode);			pol_flg ipol_flg	ER ercd = pol_flg (ID flgid, FLGPTN waiptn, MODE wfmode, FLGPTN *p_flgptn); ER ercd = ipol_flg (ID flgid, FLGPTN waiptn, MODE wfmode, FLGPTN *p_flgptn);			
		Prm	UINT	*p_flgptn; Start address of area where the bit pattern at waiting release is to be returned		Prm	FLGPTN	*p_flgptn; Pointer to area where the bit pattern at waiting release is to be returned	
8	Wait for event flag with timeout				Wait for event flag with timeout				Prm sequence and Prm type changed. Clearing specification changed. See 3.3.3 and 3.3.4.
	twai_flg	ER ercd = twai_flg (UINT *p_flgptn, ID flgid, UINT waiptn, UINT wfmode, TMO tmout);			twai_flg	ER ercd = twai_flg (ID flgid, FLGPTN waiptn, MODE wfmode, FLGPTN *p_flgptn, TMO tmout);			
		Prm	UINT	*p_flgptn; Start address of area where the bit pattern at waiting release is to be returned		Prm	FLGPTN	*p_flgptn; Pointer to area where the bit pattern at waiting release is to be returned	
9	Refer to event flag state				Refer to event flag state				Prm sequence, packet struct changed
	ref_flg	ER ercd = ref_flg (T_RFLG *pk_rflg, ID flgid);			ref_flg iref_flg	ER ercd = ref_flg (ID flgid, T_RFLG *pk_rflg); ER ercd = iref_flg (ID flgid, T_RFLG *pk_rflg);			
		Packet struct	typedef struct t_rflg{			Packet struct	typedef struct t_rflg{		
			VP	exinf; Extended information			ID	wtskid; Wait task ID	
			BOOL_ID	wtsk; Wait task ID			FLGPTN	flgptn; Event flag bit pattern	
			UINT	flgptn; Event flag bit pattern			} T_RFLG;		
			ATR	flgatr; Event flag attribute					
			B	*name; Start address of area where the name is to be returned					
		} T_RFLG;							
10	Refer to event flag state				—				Deleted
	vflg_sts	ER ercd = vflg_sts (ID *p_wtskid, UW *p_flgptn, ID flgid);			—	Not available			

4.1.6 Synchronization and Communication (Data Queue)

HI7000 Series (µITRON 3.0)			HI7000/4 Series (µITRON 4.0)		
Function			Function		
SVC	API		SVC	API	Change
—	—		Create data queue		New function
—	Not available		cre_dtq icre_dtq	ER ercd = cre_dtq (ID dtqid, T_CDTQ *pk_cdtq); ER ercd = icre_dtq (ID dtqid, T_CDTQ *pk_cdtq);	
—	—		Create data queue (assign data queue ID automatically)		New function
—	Not available		acre_dtq iacre_dtq	ER_ID dtqid = acre_dtq (T_CDTQ *pk_cdtq); ER_ID dtqid = acre_dtq (T_CDTQ *pk_cdtq);	
—	—		Delete data queue		New function
—	Not available		del_dtq	ER ercd = del_dtq (ID dtqid);	
—	—		Send data to data queue		New function
—	Not available		snd_dtq	ER ercd = snd_dtq (ID dtqid, VP_INT data);	
—	—		Send data to data queue (polling)		New function
—	Not available		psnd_dtq ipsnd_dtq	ER ercd = psnd_dtq (ID dtqid, VP_INT data); ER ercd = ipsnd_dtq (ID dtqid, VP_INT data);	
—	—		Send data to data queue with timeout		New function
—	Not available		tsnd_dtq	ER ercd = tsnd_dtq (ID dtqid, VP_INT data, TMO tmout);	
—	—		Forcibly send data to data queue		New function
—	Not available		fsnd_dtq ifsnd_dtq	ER ercd = fsnd_dtq (ID dtqid, VP_INT data); ER ercd = ifsnd_dtq (ID dtqid, VP_INT data);	
—	—		Receive data from data queue		New function
—	Not available		rcv_dtq	ER ercd = rcv_dtq (ID dtqid, VP_INT *p_data);	
—	—		Receive data from data queue (polling)		New function
—	Not available		prcv_dtq	ER ercd = prcv_dtq (ID dtqid, VP_INT *p_data);	
—	—		Receive data from data queue with timeout		New function
—	Not available		trcv_dtq	ER ercd = trcv_dtq (ID dtqid, VP_INT *p_data, TMO tmout);	
—	—		Refer to data queue state		New function
—	Not available		ref_dtq iref_dtq	ER ercd = ref_dtq (ID dtqid, T_RDTQ *pk_rdtq); ER ercd = iref_dtq (ID dtqid, T_RDTQ *pk_rdtq);	

4.1.7 Synchronization and Communication (Mailbox)

	HI7000 Series (µITRON 3.0)					HI7000/4 Series (µITRON 4.0)					
	Function					Function					
	SVC	API				SVC	API				Change
1	Create mailbox					Create mailbox					Packet struct changed. Attribute added. See 3.3.5.
	cre_mbx	ER ercd = cre_mbx (ID mbxid, T_CMBX *pk_cmbx);				cre_mbx icre_mbx	ER ercd = cre_mbx (ID mbxid, T_CMBX *pk_cmbx); ER ercd = icre_mbx (ID mbxid, T_CMBX *pk_cmbx);				
		Packet struct	typedef struct t_cmbx {				Packet struct	typedef struct t_cmbx {			
			VP	exinf;	Extended information			ATR	mbxatr;	Mailbox attribute	
			ATR	mbxatr;	Mailbox attribute			PRI	maxmpri;	Maximum value of message priority	
			B	*name;	Start address of area storing the name			VP	mprihd;	Start address of message queue header with priority	
			} T_CMBX;					} T_CMBX;			
2	Create mailbox (assign mailbox ID automatically)					Create mailbox (assign mailbox ID automatically)					SVC name, Prm, R_Prm, and packet struct changed. Attribute added. See 3.3.5.
	vasn_mbx	ER ercd= vasn_mbx (ID *p_mbxid, T_CMBX *pk_cmbx);				acre_mbx iacre_mbx	ER_ID mbxid = acre_mbx (T_CMBX *pk_cmbx); ER_ID mbxid = iacre_mbx (T_CMBX *pk_cmbx);				
		Prm	ID	*p_mbxid	Start address of area where created mailbox ID is to be returned		Prm	T_CMBX	*pk_cmbx	Pointer to packet storing mailbox creation information	
			T_CMBX	*pk_ cmbx	Start address of mailbox creation information						
		R_Prm	ER	ercd	Error code		R_Prm	ER_ID	mbxid	Created mailbox ID (a positive value) or error code	
			ID	*p_mbxid	Start address of area storing created mailbox ID						
		Packet struct	typedef struct t_cmbx {				Packet struct	typedef struct t_cmbx {			
			VP	exinf;	Extended information			ATR	mbxatr;	Mailbox attribute	
			ATR	mbxatr;	Mailbox attribute			PRI	maxmpri;	Maximum value of message priority	
			B	*name;	Start address of area storing the name			VP	mprihd;	Start address of message queue header with priority	
			} T_CMBX;					} T_CMBX;			
3	Delete mailbox					Delete mailbox					
	del_mbx	ER ercd = del_mbx (ID mbxid);				del_mbx	ER ercd = del_mbx (ID mbxid);				

	HI7000 Series (µITRON 3.0)			HI7000/4 Series (µITRON 4.0)			
	Function			Function			
	SVC	API		SVC	API		Change
4	Send data to mailbox			Send data to mailbox			SVC name changed. Message header changed. See 3.3.6.
	snd_msg	ER ercd = snd_msg (ID mbxid, T_MSG *pk_msg);		snd_mbx isnd_mbx	ER ercd = snd_mbx (ID mbxid, T_MSG *pk_msg); ER ercd = isnd_mbx (ID mbxid, T_MSG *pk_msg);		
		Packet struct typedef struct t_msg {			Packet struct typedef struct t_msg {		
		VP msghead;	Kernel management area		VP msghead;	Kernel management area	
		} T_MSG;	Message header in mailbox		} T_MSG;	Message header in mailbox	
					typedef struct t_msg_pri {		
					T_MSG msgque;	Message header	
					PRI msgpri;	Message priority	
					} T_MSG_PRI;	Mailbox message header with priority	
5	Send message to head of message queue in mailbox			—			Deleted. See 3.1.4.
	vjmp_msg	ER ercd = vjmp_msg (ID mbxid, T_MSG *pk_msg);		—	Not available		
6	Receive data from mailbox			Receive data from mailbox			SVC name and Prm sequence changed. Message header changed. See 3.3.6.
	rcv_msg	ER ercd = rcv_msg (T_MSG **ppk_msg, ID mbxid);		rcv_mbx	ER ercd = rcv_mbx (ID mbxid, T_MSG **ppk_msg);		
		Packet struct typedef struct t_msg {			Packet struct typedef struct t_msg {		
		VP msghead;	Kernel management area		VP msghead;	Kernel management area	
		} T_MSG;	Message header in mailbox		} T_MSG;	Message header in mailbox	
					typedef struct t_msg_pri {		
					T_MSG msgque;	Message header	
					PRI msgpri;	Message priority	
					} T_MSG_PRI;	Mailbox message header with priority	
7	Receive data from mailbox (polling)			Receive data from mailbox (polling)			SVC name and Prm sequence changed. Message header changed. See 3.3.6.
	prcv_msg	ER ercd = prcv_msg (T_MSG **ppk_msg, ID mbxid);		prcv_mbx iprcv_mbx	ER ercd = prcv_mbx (ID mbxid, T_MSG **ppk_msg); ER ercd = iprcv_mbx (ID mbxid, T_MSG **ppk_msg);		
		Packet struct typedef struct t_msg {			Packet struct typedef struct t_msg {		
		VP msghead;	Kernel management area		VP msghead;	Kernel management area	
		} T_MSG;	Message header in mailbox		} T_MSG;	Message header in mailbox	
					typedef struct t_msg_pri {		
					T_MSG msgque;	Message header	
					PRI msgpri;	Message priority	
					} T_MSG_PRI;	Mailbox message header with priority	

	HI7000 Series (µITRON 3.0)		HI7000/4 Series (µITRON 4.0)		
	Function		Function		
	SVC	API	SVC	API	Change
8	Receive data from mailbox with timeout		Receive data from mailbox with timeout		SVC name and Prm sequence changed. Message header changed. See 3.3.6.
	trcv_msg	ER ercd = trcv_msg (T_MSG **ppk_msg, ID mbxid, TMO tmout);	trcv_mbx	ER ercd = trcv_mbx (ID mbxid, T_MSG **ppk_msg, TMO tmout);	
		Packet struct typedef struct t_msg { VP msghead; Kernel management area } T_MSG; Message header in mailbox		Packet struct typedef struct t_msg { VP msghead; Kernel management area } T_MSG; Message header in mailbox	
				typedef struct t_msg_pri { T_MSG msgque; Message header PRI msgpri; Message priority } T_MSG_PRI; Mailbox message header with priority	
9	Refer to mailbox state		Refer to mailbox state		Prm sequence and packet struct changed. Message header changed. See 3.3.6.
	ref_mbx	ER ercd = ref_mbx (T_RMBX *pk_rmbx, ID mbxid);	ref_mbx iref_mbx	ER ercd = ref_mbx (ID mbxid, T_RMBX *pk_rmbx); ER ercd = iref_mbx (ID mbxid, T_RMBX *pk_rmbx);	
		Packet struct typedef struct t_rmbx { VP exinf; Extended information BOOL_ID wtsk; Wait task ID T_MSG *pk_msg; Start address of next message to be received ATR mbxatr; Mailbox attribute B *name; Start address of area where the name is to be returned } T_RMBX;		Packet struct typedef struct t_rmbx { ID wtskid Wait task ID T_MSG *pk_msg; Start address of next message to be received } T_RMBX;	
		typedef struct t_msg { VP msghead; Kernel management area } T_MSG; Message header in mailbox		typedef struct t_msg { VP msghead; Kernel management area } T_MSG; Message header in mailbox	
				typedef struct t_msg_pri { T_MSG msgque; Message header PRI msgpri; Message priority } T_MSG_PRI; Mailbox message header with priority	
10	Refer to mailbox state		—		Deleted
	vmbx_sts	ER ercd = vmbx_sts (ID *p_wtskid, T_MSG **ppk_msg, ID mbxid);	—	Not available	

4.1.8 Synchronization and Communication (Mutex)

HI7000 Series (µITRON 3.0)			HI7000/4 Series (µITRON 4.0)		
Function			Function		
SVC	API		SVC	API	Change
—	—		Create mutex		New function
—	Not available		cre_mtx	ER ercd = cre_mtx (ID mtxid, T_CMTX *pk_cmtx);	
—	—		Create mutex (assign mutex ID automatically)		New function
—	Not available		acre_mtx	ER_ID mtxid = acre_mtx (T_CMTX *pk_cmtx);	
—	—		Delete mutex		New function
—	Not available		del_mtx	ER ercd = del_mtx (ID mtxid);	
—	—		Lock mutex		New function
—	Not available		loc_mtx	ER ercd = loc_mtx (ID mtxid);	
—	—		Lock mutex (polling)		New function
—	Not available		ploc_mtx	ER ercd = ploc_mtx (ID mtxid);	
—	—		Lock mutex with timeout		New function
—	Not available		tloc_mtx	ER ercd = tloc_mtx (ID mtxid, TMO tmout);	
—	—		Unlock mutex		New function
—	Not available		unl_mtx	ER ercd = unl_mtx (ID mtxid);	
—	—		Refer to mutex state		New function
—	Not available		ref_mtx	ER ercd = ref_mtx (ID mtxid, T_RMTX *pk_rmtx);	

4.1.9 Synchronization and Communication (Message Buffer)

	HI7000 Series (µITRON 3.0)				HI7000/4 Series (µITRON 4.0)				
	Function				Function				
	SVC	API			SVC	API			Change
1	Create message buffer				Create message buffer				Packet struct changed. Wait queue operation changed. See 3.3.7.
	cre_mbf	ER ercd = cre_mbf (ID mbfid, T_CMBF *pk_cmbf);			cre_mbf icre_mbf	ER ercd = cre_mbf (ID mbfid, T_CMBF *pk_cmbf); ER ercd = icre_mbf (ID mbfid, T_CMBF *pk_cmbf);			
		Packet struct typedef struct t_cmbf {				Packet struct typedef struct t_cmbf {			
			VP	exinf;	Extended information		ATR	mbfatr;	Message buffer attribute
			ATR	mbfatr;	Message buffer attribute		UINT	maxmsz;	Maximum message size (bytes)
			INT	bufsz;	Message buffer size (bytes)		SIZE	mbfsz;	Message buffer area size (bytes)
			INT	maxmsz;	Maximum message size (bytes)		VP	mbf;	Start address of message buffer area
			B	*name;	Start address of area storing the name		} T_CMBF;		
		} T_CMBF;							
2	Create message buffer (assign message buffer ID automatically)				Create message buffer (assign message buffer ID automatically)				SVC name, Prm, R_Prm, and packet struct changed. Wait queue operation changed. See 3.3.7.
	vasn_mbf	ER ercd = vasn_mbf (ID *p_mbfid, T_CMBF *pk_cmbf);			acre_mbf iacre_mbf	ER_ID mbfid = acre_mbf (T_CMBF *pk_cmbf); ER_ID mbfid = iacre_mbf (T_CMBF *pk_cmbf);			
		Prm	ID	*p_mbfid	Start address of area where created message buffer ID is to be returned		Prm	T_CMBF	*pk_cmbf	Pointer to packet storing the message buffer creation information
			T_CMBF	*pk_cmbf	Start address of message buffer creation information					
		R_Prm	ER	ercd	Error code		R_Prm	ER_ID	mbfid	Created message buffer ID (a positive value) or error code
			ID	*p_mbfid	Start address of area storing created message buffer ID					
		Packet struct typedef struct t_cmbf {					Packet struct typedef struct t_cmbf {			
			VP	exinf;	Extended information			ATR	mbfatr;	Message buffer attribute
			ATR	mbfatr;	Message buffer attribute			UINT	maxmsz;	Maximum message size (bytes)
			INT	bufsz;	Message buffer size (bytes)			SIZE	mbfsz;	Message buffer area size (bytes)
			INT	maxmsz;	Maximum message size (bytes)			VP	mbf;	Start address of message buffer area
			B	*name;	Start address of area storing the name			} T_CMBF;		
		} T_CMBF;								
3	Delete message buffer				Delete message buffer				
	del_mbf	ER ercd = del_mbf (ID mbfid);			del_mbf	ER ercd = del_mbf (ID mbfid);			
4	Send data to message buffer				Send data to message buffer				Prm type changed. Wait queue operation changed. See 3.3.7.
	snd_mbf	ER ercd = snd_mbf (ID mbfid, VP msg, INT msgsz);			snd_mbf	ER ercd = snd_mbf (ID mbfid, VP msg, UINT msgsz);			
		Prm	INT	msgsz;	Size of the message to send (bytes)		Prm	UINT	msgsz;	Size of the message to send (bytes)

	HI7000 Series (µITRON 3.0)				HI7000/4 Series (µITRON 4.0)				
	Function				Function				
	SVC	API			SVC	API			Change
5	Send data to message buffer (polling)				Send data to message buffer (polling)				Prm type changed. Wait queue operation changed. See 3.3.7.
	psnd_mbf	ER ercd = psnd_mbf (ID mbfid, VP msg, INT msgsz);			psnd_mbf ipsnd_mbf	ER ercd = psnd_mbf (ID mbfid, VP msg, UINT msgsz); ER ercd = ipsnd_mbf (ID mbfid, VP msg, UINT msgsz);			
		Prm	INT	msgsz; Size of the message to send (bytes)		Prm	UINT	msgsz; Size of the message to send (bytes)	
6	Send data to message buffer with timeout				Send data to message buffer with timeout				Prm type changed. Wait queue operation changed. See 3.3.7.
	tsnd_mbf	ER ercd = tsnd_mbf (ID mbfid, VP msg, INT msgsz, TMO tmout);			tsnd_mbf	ER ercd = tsnd_mbf (ID mbfid, VP msg, UINT msgsz, TMO tmout);			
		Prm	INT	msgsz; Size of the message to send (bytes)		Prm	UINT	msgsz; Size of the message to send (bytes)	
7	Receive data from message buffer				Receive data from message buffer				Prm and R_Prm changed. Wait queue operation changed. See 3.3.7.
	rcv_mbf	ER ercd = rcv_mbf (VP msg, INT *p_msgsz, ID mbfid);			rcv_mbf	ER_UINT msgsz = rcv_mbf (ID mbfid, VP msg);			
		Prm	VP	msg Start address of area where received message is to be returned		Prm	ID	mbfid Message buffer ID	
			INT	*p_msgsz Start address of area where the size (bytes) of received message is to be returned			VP	msg Start address of area where received message is to be returned	
			ID	mbfid Message buffer ID					
		R_Prm	ER	ercd Error code		R_Prm	ER_UINT	msgsz Received message size (bytes, a positive value) or error code	
			VP	msg Start address of area storing received message			VP	msg Start address of area storing received message	
			INT	*p_msgsz Start address of area storing the size (bytes) of received message					
8	Receive data from message buffer (polling)				Receive data from message buffer (polling)				Prm and R_Prm changed. Cannot be issued in a non-task context. Wait queue operation changed. See 3.3.7.
	prcv_mbf	ER ercd = prcv_mbf (VP msg, INT *p_msgsz, ID mbfid);			prcv_mbf	ER_UINT msgsz = prcv_mbf (ID mbfid, VP msg);			
		Prm	VP	msg Start address of area where received message is to be returned		Prm	ID	mbfid Message buffer ID	
			INT	*p_msgsz Start address of area where the size (bytes) of received message is to be returned			VP	msg Start address of area where received message is to be returned	
			ID	mbfid Message buffer ID					
		R_Prm	ER	ercd Error code		R_Prm	ER_UINT	msgsz Received message size (bytes, a positive value) or error code	
			VP	msg Start address of area storing received message			VP	msg Start address of area storing received message	
			INT	*p_msgsz Start address of area storing the size (bytes) of received message					
		Operation	Can be issued in a task-independent portion.			Operation	Cannot be issued in a non-task context		

	HI7000 Series (µITRON 3.0)				HI7000/4 Series (µITRON 4.0)				
	Function				Function				
	SVC	API			SVC	API			Change
9	Receive data from message buffer with timeout				Receive data from message buffer with timeout				Prm and R_Prm changed. Wait queue operation changed. See 3.3.7.
	trcv_mbf	ER ercd = trcv_mbf (VP msg, INT *p_msgsz, ID mbfid, TMO tmout);			trcv_mbf	ER_UINT msgsz = trcv_mbf (ID mbfid, VP msg, TMO tmout);			
		Prm	VP	msg	Start address of area where received message is to be returned	Prm	ID	mbfid	Message buffer ID
			INT	*p_msgsz	Start address of area where the size (bytes) of received message is to be returned		VP	msg	Start address of area where received message is to be returned
			ID	mbfid	Message buffer ID		TMO	tmout	Timeout specification
			TMO	tmout	Timeout specification				
		R_Prm	ER	ercd	Error code	R_Prm	ER_UINT	msgsz	Received message size (bytes, a positive value) or error code
			VP	msg	Start address of area storing received message		VP	msg	Start address of area storing received message
			INT	*p_msgsz	Start address of area storing the size (bytes) of received message				
10	Refer to message buffer state				Refer to message buffer state				Prm sequence and packet struct changed. Wait queue operation changed. See 3.3.7.
	ref_mbf	ER ercd = ref_mbf (T_RMBF *pk_rmbf, ID mbfid);			ref_mbf iref_mbf	ER ercd = ref_mbf (ID mbfid, T_RMBF *pk_rmbf); ER ercd = iref_mbf (ID mbfid, T_RMBF *pk_rmbf);			
		Packet struct	typedef struct t_rmbf {			Packet struct	typedef struct t_rmbf {		
			VP	exinf;	Extended information		ID	stskid;	Start task ID of the queue waiting for sending
			BOOL_ID	wtsk;	Task ID waiting for receiving		ID	rtskid;	Start task ID of the queue waiting for receiving
			BOOL_ID	stsk;	Task ID waiting for sending		UINT	smsgcnt;	Number of messages in message buffer
			INT	msgsz;	Size of next message to be received (bytes)		SIZE	fmbfsz;	Size of free buffer (bytes)
			INT	frbufsz;	Size of free buffer (bytes)		} T_RMBF;		
			ATR	mbfatr;	Message buffer attribute				
			INT	bufsz;	Message buffer size (bytes)				
			INT	maxmsz;	Maximum message size (bytes)				
			B	*name;	Start address of area where the name is to be returned				
		} T_RMBF;							

4.1.10 Interrupt Management

	HI7000 Series (µITRON 3.0)		HI7000/4 Series (µITRON 4.0)		
	Function		Function		
	SVC	API	SVC	API	Change
1	Define interrupt handler		Define interrupt handler		SVC name and packet name changed.
	def_int	ER ercd = def_int (UINT dintno, T_DINT *pk_dint);	def_inh idef_inh	ER ercd = def_inh (INHNO inhno, T_DINH *pk_dinh); ER ercd = idef_inh (INHNO inhno, T_DINH *pk_dinh);	
		Packet struct typedef struct t_dint { ATR intatr; Handler attribute FP inthdr; Handler address UINT intsr; SR at initiation (ignored in HI7000S/3) } T_DINT;		Packet struct typedef struct t_dinh { ATR inhatr; Handler attribute FP inthdr; Handler address UINT inhsr; SR at initiation (ignored in HI7000/4) } T_DINH;	
2	Define trap exception processing routine [HI7700, HI7750]		Define CPU exception handler (TRAP exception)		SVC name and packet name changed.
	vdef_trp	ER ercd = vdef_trp (UINT dtrpno, T_DTRP *pk_dtrp);	vdef_trp ivdef_trp	ER ercd = vdef_trp (UINT dtrpno, T_DTRP *pk_dtrp); ER ercd = ivdef_trp (UINT dtrpno, T_DTRP *pk_dtrp);	
		Packet struct typedef struct t_dtrp { ATR trpatr; Attribute of trap exception processing routine FP trphdr; Address of trap exception processing routine UINT trpsr; SR at initiation of trap exception processing routine } T_DTRP;		Packet struct typedef struct t_dtrp { ATR trpatr; Attribute of CPU exception (TRAPA instruction exception) handler FP trphdr; Address of CPU exception (TRAPA instruction exception) handler UINT trpsr; SR at initiation of CPU exception (TRAPA instruction exception) handler } T_DTRP;	
—	—		Define CPU exception handler		New function
	—	Not available	def_exc idef_exc	ER ercd = def_exc (EXCNO excno, T_DEXC *pk_dexc); ER ercd = def_exc (EXCNO excno, T_DEXC *pk_dexc);	
3	Return from interrupt handler		Return from interrupt handler		
	ret_int	[HI7000] This SVC is automatically issued through the #pragma interrupt statement at the end of the interrupt handler function. [HI7700, HI7750] This SVC is automatically issued when execution returns from the interrupt handler function.	—	[HI7000/4 direct interrupt handler] Control automatically returns to the caller through the #pragma interrupt statement at the end of the interrupt handler function. [HI7000/4 series ordinary interrupt handler] Control automatically returns to the caller when execution returns from the interrupt handler function.	
4	Return from exception processing		Return from exception processing		
	vret_exc	[HI7000] This SVC is automatically issued through the #pragma interrupt statement at the end of the exception processing routine function. [HI7700, HI7750] This SVC is automatically issued when execution returns from the exception processing routine function.	—	Control automatically returns to the caller when execution returns from the CPU exception handler.	
5	Disable interrupt and task dispatch		Lock CPU		Meaning of CPU lock changed. See 3.3.1.
	loc_cpu	ER ercd = loc_cpu(void);	loc_cpu iloc_cpu	ER ercd = loc_cpu(); ER ercd = iloc_cpu();	
6	Enable interrupt and task dispatch		Unlock CPU		Meaning of CPU lock changed. See 3.3.1.
	unl_cpu	ER ercd = unl_cpu(void);	unl_cpu iunl_cpu	ER ercd = unl_cpu(); ER ercd = iunl_cpu();	

	HI7000 Series (µITRON 3.0)				HI7000/4 Series (µITRON 4.0)				
	Function				Function				
	SVC	API			SVC	API			Change
7	Change interrupt mask				Change interrupt mask				
	chg_ims	ER ercd = chg_ims (UINT imask);			chg_ims ichg_ims	ER ercd = chg_ims (IMASK imask); ER ercd = ichg_ims (IMASK imask);			
8	Refer to interrupt mask				Refer to interrupt mask				SVC name and Prm type changed.
	ref_ims	ER ercd = ref_ims (UINT *p_imask);			get_ims iget_ims	ER ercd = get_ims (IMASK *p_imask); ER ercd = iget_ims (IMASK *p_imask);			
		Prm	UINT	*p_imask; Start address of area where interrupt mask level is to be returned		Prm	IMASK	*p_imask; Start address of area where interrupt mask level is to be returned	

4.1.11 Memory Pool Management (Fixed-Size Memory Pool)

	HI7000 Series (µITRON 3.0)				HI7000/4 Series (µITRON 4.0)				
	Function				Function				
	SVC	API			SVC	API			Change
1	Create fixed-size memory pool				Create fixed-size memory pool				Packet struct changed
	cre_mpf	ER ercd = cre_mpf (ID mpfid, T_CMPF *pk_cmpf);			cre_mpf icre_mpf	ER ercd = cre_mpf (ID mpfid, T_CMPF *pk_cmpf); ER ercd = icre_mpf (ID mpfid, T_CMPF *pk_cmpf);			
		Packet struct typedef struct t_cmpf {				Packet struct typedef struct t_cmpf {			
			VP	exinf;	Extended information		ATR	mpfatr;	Fixed-size memory pool attribute
			ATR	mpfatr;	Fixed-size memory pool attribute		UINT	blkcnt;	Number of blocks in memory pool
			INT	mpfcnt;	Number of blocks in memory pool		UINT	blksz;	Block size of fixed-size memory pool (bytes)
			INT	blfsz;	Block size of fixed-size memory pool (bytes)		VP	mpf;	Start address of fixed-size memory pool area
			B	*name;	Start address of area storing the name		VP	mpfmb;	Start address of fixed-size memory block management area (only when CFG_MPFMANAGE is selected in HI7000/4)
		} T_CMPF;					} T_CMPF;		
2	Create fixed-size memory pool (assign memory pool ID automatically)				Create fixed-size memory pool (assign memory pool ID automatically)				SVC name, Prm, R_Prm, and packet struct changed
	vasn_mpf	ER ercd = vasn_mpf (ID *p_mpfid, T_CMPF *pk_cmpf);			acre_mpf iacre_mpf	ER_ID mpfid = acre_mpf (T_CMPF *pk_cmpf); ER_ID mpfid = iacre_mpf (T_CMPF *pk_cmpf);			
		Prm	ID	*p_mpfid	Start address of area where created fixed-size memory pool ID is to be returned	Prm	T_CMPF	*pk_cmpf	Pointer to packet storing fixed-size memory pool creation information
			T_CMPF	*pk_cmpf	Start address of fixed-size memory pool creation information				
		R_Prm	ER	ercd	Error code	R_Prm	ER_ID	mpfid	Created fixed-size memory pool ID (a positive value) or error code
			ID	*p_mpfid	Start address of area storing created fixed-size memory pool ID				
3	Delete fixed-size memory pool				Delete fixed-size memory pool				
	del_mpf	ER ercd = del_mpf (ID mpfid);			del_mpf	ER ercd = del_mpf (ID mpfid);			
4	Get fixed-size memory block				Get fixed-size memory block				SVC name and Prm sequence changed.
	get_blf	ER ercd = get_blf (VP *p_blf, ID mpfid);			get_mpf	ER ercd = get_mpf (ID mpfid, VP *p_blk);			
5	Get fixed-size memory block (polling)				Get fixed-size memory block (polling)				SVC name and Prm sequence changed.
	pget_blf	ER ercd = pget_blf (VP *p_blf, ID mpfid);			pget_mpf ipget_mpf	ER ercd = pget_mpf (ID mpfid, VP *p_blk); ER ercd = ipget_mpf (ID mpfid, VP *p_blk);			
6	Get fixed-size memory block with timeout				Get fixed-size memory block with timeout				SVC name and Prm sequence changed.
	tget_blf	ER ercd = tget_blf (VP *p_blf, ID mpfid, TMO tmout);			tget_mpf	ER ercd = tget_mpf (ID mpfid, VP *p_blk, TMO tmout);			
7	Release fixed-size memory block				Release fixed-size memory block				SVC name changed
	rel_blf	ER ercd = rel_blf (ID mpfid, VP blf);			rel_mpf irel_mpf	ER ercd = rel_mpf (ID mpfid, VP blk); ER ercd = irel_mpf (ID mpfid, VP blk);			

	HI7000 Series (µITRON 3.0)		HI7000/4 Series (µITRON 4.0)		
	Function		Function		
	SVC	API	SVC	API	Change
8	Refer to fixed-size memory pool state		Refer to fixed-size memory pool state		Prm sequence and packet struct changed
	ref_mpf	ER ercd = ref_mpf (T_RMPF *pk_rmpf, ID mpfid); Packet struct typedef struct t_rmpf { VP exinf; Extended information BOOL_ID wtsk; Wait task ID INT frbcnt; Number of blocks of memory space available ATR mpfatr; Fixed-size memory pool attribute INT mpfcnt; Number of blocks in memory pool INT blfsz; Block size of fixed-size memory pool (bytes) B *name; Start address of area where the name is to be returned } T_RMPF;	ref_mpf iref_mpf	ER ercd = ref_mpf (ID mpfid, T_RMPF *pk_rmpf); ER ercd = iref_mpf (ID mpfid, T_RMPF *pk_rmpf); Packet struct typedef struct t_rmpf { ID wtskid; Wait task ID UINT fblkcnt; Number of blocks of memory space available } T_RMPF;	
9	Refer to fixed-size memory pool state		—		Deleted
	vmpf_sts	ER ercd = vmpf_sts (ID *p_wtskid, W *p_frbcnt, ID mpfid);	—	Not available	

4.1.12 Memory Pool Management (Variable-Size Memory Pool)

	HI7000 Series (µITRON 3.0)				HI7000/4 Series (µITRON 4.0)				
	Function				Function				
	SVC	API			SVC	API			Change
1	Create variable-size memory pool				Create variable-size memory pool				Packet struct changed
	cre_mpl	ER ercd = cre_mpl (ID mplid, T_CMPL *pk_cmpl);			cre_mpl icre_mpl	ER ercd = cre_mpl (ID mplid, T_CMPL *pk_cmpl); ER ercd = icre_mpl (ID mplid, T_CMPL *pk_cmpl);			
		Packet struct	typedef struct t_cmpl {			Packet struct	typedef struct t_cmpl {		
			VP	exinf;	Extended information		ATR	mplatr;	Variable-size memory pool attribute
			ATR	mplatr;	Variable-size memory pool attribute		SIZE	mplsz;	Size of memory pool (bytes)
			INT	mplsz;	Size of memory pool (bytes)		VP	mpl;	Start address of variable-size memory pool area
			B	*name;	Start address of area storing the name		} T_CMPL;		
			} T_CMPL;						
2	Create variable-size memory pool (assign memory pool ID automatically)				Create variable-size memory pool (assign memory pool ID automatically)				SVC name, Prm, R_Prm, and packet struct changed
	vasn_mpl	ER ercd = vasn_mpl (ID *p_mplid, T_CMPL *pk_cmpl);			acre_mpl iacre_mpl	ER_ID mplid = acre_mpl (T_CMPL *pk_cmpl); ER_ID mplid = iacre_mpl (T_CMPL *pk_cmpl);			
		Prm	ID	*p_mplid	Start address of area where created variable-size memory pool ID is to be returned	Prm	T_CMPL	*pk_cmpl	Pointer to packet storing variable-size memory pool creation information
			T_CMPL	*pk_cmpl	Start address of variable-size memory pool creation information				
		R_Prm	ER	ercd	Error code		ER	ercd	Error code
			ID	*p_mplid	Start address of area storing created variable-size memory pool ID		ID	*p_mplid	Start address of area storing created variable-size memory pool ID
3	Delete variable-size memory pool				Delete variable-size memory pool				
	del_mpl	ER ercd = del_mpl (ID mplid);			del_mpl	ER ercd = del_mpl (ID mplid);			
4	Get variable-size memory block				Get variable-size memory block				SVC name and Prm sequence changed
	get_blk	ER ercd = get_blk (VP *p_blk, ID mplid, INT blksz);			get_mpl	ER ercd = get_mpl (ID mplid, UINT blksz, VP *p_blk);			
5	Get variable-size memory block (polling)				Get variable-size memory block (polling)				SVC name and Prm sequence changed
	pget_blk	ER ercd = pget_blk (VP *p_blk, ID mplid, INT blksz);			pget_mpl ipget_mpl	ER ercd = pget_mpl (ID mplid, UINT blksz, VP *p_blk); ER ercd = ipget_mpl (ID mplid, UINT blksz, VP *p_blk);			
6	Get variable-size memory block with timeout				Get variable-size memory block with timeout				SVC name and Prm sequence changed
	tget_blk	ER ercd = tget_blk (VP *p_blk, ID mplid, INT blksz, TMO tmout);			tget_mpl	ER ercd = tget_mpl (ID mplid, UINT blksz, VP *p_blk, TMO tmout);			
7	Release variable-size memory block				Release variable-size memory block				SVC name changed
	rel_blk	ER ercd = rel_blk (ID mplid, VP blk);			rel_mpl irel_mpl	ER ercd = rel_mpl (ID mplid, VP blk); ER ercd = irel_mpl (ID mplid, VP blk);			

	HI7000 Series (µITRON 3.0)				HI7000/4 Series (µITRON 4.0)				
	Function				Function				
	SVC	API			SVC	API			Change
8	Refer to variable-size memory pool				Refer to variable-size memory pool				Prm sequence and packet struct changed
	ref_mpl	ER ercd = ref_mpl (T_RMPL *pk_rmpl, ID mplid);			ref_mpl iref_mpl	ER ercd = ref_mpl (ID mplid, T_RMPL *pk_rmpl); ER ercd = iref_mpl (ID mplid, T_RMPL *pk_rmpl);			
		Packet struct	typedef struct t_rmpl {			Packet struct	typedef struct t_rmpl {		
			VP	exinf;	Extended information		ID	wtskid;	Wait task ID
			BOOL_ID	wtsk;	Wait task ID		SIZE	fmplsz;	Total size of available memory area (bytes)
			INT	frsz;	Total size of available memory area (bytes)		UINT	fblksz;	Maximum memory block size available (bytes)
			INT	maxsz;	Maximum memory area available (bytes)		} T_RMPL;		
			ATR	mplatr;	Variable-size memory pool attribute				
			INT	mplsz;	Size of memory pool (bytes)				
			B	*name;	Start address of area where the name is to be returned				
		} T_RMPL;							

4.1.13 Time Management

	HI7000 Series (µITRON 3.0)		HI7000/4 Series (µITRON 4.0)		
	Function		Function		
	SVC	API	SVC	API	Change
1	Set system clock		Set system clock		Time management changed. See 2.3.10.
	set_tim	ER ercd = set_tim (SYSTIME *pk_tim);	set_tim iset_tim	ER ercd = set_tim (SYSTIM *p_systim); ER ercd = iset_tim (SYSTIM *p_systim);	
2	Get system clock		Get system clock		Time management changed. See 2.3.10.
	get_tim	ER ercd = get_tim (SYSTIME *pk_tim);	get_tim iget_tim	ER ercd = get_tim (SYSTIM *p_systim); ER ercd = iget_tim (SYSTIM *p_systim);	
3	Delay task		Delay task		Time management changed. See 2.3.10.
	dly_tsk	ER ercd = dly_tsk (DLYTIME dlytim);	dly_tsk	ER ercd = dly_tsk (RELTIM dlytim);	
4	Define cyclic handler		Define cyclic handler		Modifications made throughout the function. See 2.3.10 and 2.3.11.
	def_cyc	ER ercd = def_cyc (HNO cycno, T_DCYC *pk_dcyc);	cre_cyc icre_cyc	ER ercd = cre_cyc (ID cycid, T_CCYC *pk_ccyc); ER ercd = icre_cyc (ID cycid, T_CCYC *pk_ccyc);	
5	Define cyclic handler (assign cyclic handler number automatically)		Define cyclic handler (assign cyclic handler number automatically)		
	vasn_cyc	ER ercd = vasn_cyc (HNO *p_cycno, T_DCYC *pk_dcyc);	acre_cyc iacre_cyc	ER_ID cycid = acre_cyc (T_CCYC *pk_ccyc); ER_ID cycid = iacre_cyc (T_CCYC *pk_ccyc);	
—	—	—	Delete cyclic handler		
	—	Not available	del_cyc	ER ercd = del_cyc (ID cycid);	
6	Activate cyclic handler		Start cyclic handler		
	act_cyc	ER ercd = act_cyc (HNO cycno, UINT cycact);	sta_cyc ista_cyc	ER ercd = sta_cyc (ID cycid); ER ercd = ista_cyc (ID cycid);	
			Stop cyclic handler		
			stp_cyc istp_cyc	ER ercd = stp_cyc (ID cycid); ER ercd = istp_cyc (ID cycid);	
7	Refer to cyclic handler state		Refer to cyclic handler state		
	ref_cyc	ER ercd = ref_cyc (T_RCYC *pk_rcyc, HNO cycno);	ref_cyc iref_cyc	ER ercd = ref_cyc (ID cycid, T_RCYC *pk_rcyc); ER ercd = iref_cyc (ID cycid, T_RCYC *pk_rcyc);	

	HI7000 Series (µITRON 3.0)		HI7000/4 Series (µITRON 4.0)		
	Function		Function		
	SVC	API	SVC	API	Change
8	Define alarm handler		Define alarm handler		Modifications made throughout the function. See 2.3.10 and 2.3.12.
	def_alm	ER ercd = def_alm (HNO almno, T_DALM *pk_dalm);	cre_alm icre_alm	ER ercd = cre_alm (ID almid, T_CALM *pk_calm); ER ercd = icre_alm (ID almid, T_CALM *pk_calm);	
9	Define alarm handler (assign alarm handler number automatically)		Define alarm handler (assign alarm handler number automatically)		
	vasn_alm	ER ercd = vasn_alm (HNO *p_almno, T_DALM *pk_dalm);	acre_alm iacre_alm	ER_ID almid = acre_alm (T_CALM *pk_calm); ER_ID almid = iacre_alm (T_CALM *pk_calm);	
—	—		Delete alarm handler		
	—	Not available	del_alm	ER ercd = del_alm (ID almid);	
—	—		Start alarm handler		
	—	Not available	sta_alm ista_alm	ER ercd = sta_alm (ID almid, RELTIM almtim); ER ercd = ista_alm (ID almid, RELTIM almtim);	
—	—		Stop alarm handler		
	—	Not available	stp_alm istp_alm	ER ercd = stp_alm (ID almid); ER ercd = istp_alm (ID almid);	
10	Refer to alarm handler state		Refer to alarm handler state		
	ref_alm	ER ercd = ref_alm (T_RALM *pk_ralm, HNO almno);	ref_alm iref_alm	ER ercd = ref_alm (ID almid, T_RALM *pk_ralm); ER ercd = iref_alm (ID almid, T_RALM *pk_ralm);	
11	Return from timer handler		Return from timer handler		Time management changed. See 2.3.10.
	ret_tmr	The SVC is issued automatically when execution returns from the timer handler function.	—	Control automatically returns.	
12	Change time slice value of current task		—		Equivalent function available through an overrun handler. See 2.1.2.
	vchg_qua	ER ercd = vchg_qua (ID tskid, TMO quantum);	—	Not available	
13	Update system clock		Update system clock		Timer driver changed. See 2.3.9.
	vsys_clk	void vsys_clk(void);	isig_tim	The SVC is automatically included by selecting CFG_TIMUSE.	

4.1.14 Time Management (Overrun Handler)

	HI7000 Series (µITRON 3.0)		HI7000/4 Series (µITRON 4.0)		
	Function		Function		
	SVC	API	SVC	API	Change
—	—		Define overrun handler		New function
	—	Not available	def_ovr	ER ercd = def_ovr (T_DOVR *pk_dovr);	
—	—		Start overrun handler		New function
	—	Not available	sta_ovr ista_ovr	ER ercd = sta_ovr (ID tskid, OVRTIM ovrtim); ER ercd = ista_ovr (ID tskid, OVRTIM ovrtim);	
—	—		Stop overrun handler		New function
	—	Not available	stp_ovr istp_ovr	ER ercd = stp_ovr (ID tskid); ER ercd = istp_ovr (ID tskid);	
—	—		Refer to overrun handler state		New function
	—	Not available	ref_ovr iref_ovr	ER ercd = ref_ovr (ID tskid, T_ROVR *pk_rovr); ER ercd = ref_ovr (ID tskid, T_ROVR *pk_rovr);	

4.1.15 Cache Support Function [HI7700, HI7750]

HI7700/4 for SH-3 and SH3-DSP: Changed
HI7750/4 for SH-4: Changed
HI7700/4 for SH4AL-DSP and HI7750/4 for SH-4A: Newly added

No.	3.0	CPU	4.0	CPU	Target Cache (HI7000/4)	
1	HI7700	SH-3, SH3-DSP	HI7700/4	SH-3, SH3-DSP	Mixed instruction and operand cache	Changed
2				SH4AL-DSP	Instruction cache and operand cache	Applicable CPU added
3	HI7750	SH-4	HI7750/4	SH-4	Operand cache	Changed
4				SH-4A	Instruction cache and operand cache	Applicable CPU added

	HI7000 Series (µITRON 3.0)		HI7000/4 Series (µITRON 4.0)		
	Function		Function		
	SVC	API	SVC	API	Change
1	Initialize cache		Initialize cache		Prm added
	vini_cac	void vini_cac (UW ccr_data);	[HI7700/4 (for SH-3, SH3-DSP)]		
			vini_cac ivini_cac	void vini_cac (UW ccr_data, UW entnum, UW waynum); void ivini_cac (UW ccr_data, UW entnum, UW waynum); Prm UW entnum; Number of cache entries UW waynum; Number of cache ways	
			[HI7750/4 (for SH-4)]		
			vini_cac ivini_cac	void vini_cac (UW ccr_data); void ivini_cac (UW ccr_data);	
			[HI7700/4 (for SH4AL-DSP) and HI7750/4 (for SH-4A)]		Applicable CPU added.
			vini_cac ivini_cac	ER vini_cac (ATR cacatr); ER ivini_cac (ATR cacatr); Prm ATR cacatr; Cache attribute	
2	Clear cache		Clear cache		
	vclr_cac	ER ercd = vclr_cac (VP clradr1, VP clradr2);	[HI7700/4 (for SH-3, SH3-DSP)]		
			vclr_cac ivclr_cac	ER ercd = vclr_cac (VP clradr1, VP clradr2); ER ercd = ivclr_cac (VP clradr1, VP clradr2);	
			[HI7750/4 (for SH-4)]		Operand cache is cleared.
			vclr_cac ivclr_cac	ER ercd = vclr_cac (VP clradr1, VP clradr2); ER ercd = ivclr_cac (VP clradr1, VP clradr2);	
			[HI7700/4 (for SH4AL-DSP) and HI7750/4 (for SH-4A)]		Applicable CPU added. Instruction cache and operand cache are cleared.
			vclr_cac ivclr_cac	ER ercd = vclr_cac (VP clradr1, VP clradr2, MODE mode); ER ercd = ivclr_cac (VP clradr1, VP clradr2, MODE mode); Prm MODE mode; Target cache	

	HI7000 Series (µITRON 3.0)		HI7000/4 Series (µITRON 4.0)		
	Function		Function		
	SVC	API	SVC	API	Change
3	Flush cache		Flush cache		
	vfls_cac	ER ercd = vfls_cac (VP flsadr1, VP flsadr2);	[HI7700/4 (for SH-3, SH3-DSP)]		
			vfls_cac ivfls_cac	ER ercd = vfls_cac (VP flsadr1, VP flsadr2); ER ercd = ivfls_cac (VP flsadr1, VP flsadr2);	
			[HI7750/4 (for SH-4)]		Operand cache is flushed.
			vfls_cac ivfls_cac	ER ercd = vfls_cac (VP flsadr1, VP flsadr2); ER ercd = ivfls_cac (VP flsadr1, VP flsadr2);	
			[HI7700/4 (for SH4AL-DSP) and HI7750/4 (for SH-4A)]		Applicable CPU added. Operand cache is cleared.
			vfls_cac ivfls_cac	ER ercd = vfls_cac (VP flsadr1, VP flsadr2); ER ercd = ivfls_cac (VP flsadr1, VP flsadr2);	
4	Invalidate cache		Invalidate cache		
	vinv_cac	[HI7700] ER ercd = vinv_cac(void);	[HI7700/4 (for SH-3, SH3-DSP)]		
			vinv_cac ivinv_cac	ER ercd = vinv_cac(void); ER ercd = ivinv_cac(void);	
		[HI7750] ER ercd = vinv_cac (VP invadr1, VP invadr2);	[HI7750/4 (for SH-4)]		Operand cache is invalidated.
			vinv_cac ivinv_cac	ER ercd = vinv_cac (VP invadr1, VP invadr2); ER ercd = ivinv_cac (VP invadr1, VP invadr2);	
			[HI7700/4 (for SH4AL-DSP) and HI7750/4 (for SH-4A)]		Applicable CPU added. Instruction cache and operand cache are invalidated.
			vinv_cac ivinv_cac	ER ercd = vinv_cac (VP invadr1, VP invadr2, MODE mode); ER ercd = ivinv_cac (VP invadr1, VP invadr2, MODE mode);	
				Prm MODE mode; Target cache	

4.1.16 System Configuration Management

	HI7000 Series (µITRON 3.0)				HI7000/4 Series (µITRON 4.0)				
	Function				Function				
	SVC	API			SVC	API			Change
1	Refer to version information				Refer to version information				SVC name and packet struct changed
	get_ver	ER ercd = get_ver (T_VER *pk_ver);			ref_ver iref_ver	ER ercd = ref_ver (T_RVER *pk_rver); ER ercd = iref_ver (T_RVER *pk_rver);			
		Packet struct typedef struct t_ver {				Packet struct typedef struct t_rver {			
		UH	maker;	Manufacturer		UH	maker;	Manufacturer	
		UH	id;	Identification number		UH	prid;	Identification number	
		UH	spver;	Specification version		UH	spver;	Specification version	
		UH	prver;	Product version		UH	prver;	Product version	
		UH	prno[4];	Product management information		UH	prno[4];	Product management information	
		UH	cpu;	CPU information		} T_RVER;			
		UH	var;	Variation descriptor					
		} T_VER;							
2	Refer to system state				—				Deleted
	ref_sys	ER ercd = ref_sys (T_RSYS *pk_rsys);			—	Not available			

	HI7000 Series (µITRON 3.0)				HI7000/4 Series (µITRON 4.0)				
	Function				Function				
	SVC	API			SVC	API			Change
3	Refer to configuration information				Refer to configuration information				Packet struct changed
	ref_cfg	ER ercd = ref_cfg (T_RCFG *pk_rcfg);			ref_cfg iref_cfg	ER ercd = ref_cfg (T_RCFG *pk_rcfg); ER ercd = iref_cfg (T_RCFG *pk_rcfg);			
		Packet struct typedef struct t_rcfg {				Packet struct typedef struct t_rcfg {			
		UW	knlmsklvl;	Kernel interrupt mask level		ID	maxtskid;	Maximum task ID	
		UH	timintvl;	System clock interval		ID	ststkid;	Maximum ID of task using static stack	
		UB	uppintnst;	Number of nested interrupts that are higher in priority than the kernel interrupt mask level		ID	maxsemid;	Maximum semaphore ID	
		UB	lowintnst;	Number of nested interrupts that are, in priority, equal to or lower than the kernel interrupt mask level		ID	maxflgid;	Maximum event flag ID	
		ID	maxtskid;	Maximum task ID		ID	maxdtqid;	Maximum data queue ID	
		ID	ststkid;	Maximum ID of task using static stack		ID	maxmbxid;	Maximum mailbox ID	
		PRI	maxtskpri;	Maximum task priority		ID	maxmtxid;	Maximum mutex ID	
		ID	maxflgid;	Maximum event flag ID		ID	maxmbfid;	Maximum message buffer ID	
		ID	maxsemid;	Maximum semaphore ID		ID	maxmplid;	Maximum variable-size memory pool ID	
		ID	maxmbxid;	Maximum mailbox ID		ID	maxmpfid;	Maximum fixed-size memory pool ID	
		ID	maxmbfid;	Maximum message buffer ID		ID	maxcycid;	Maximum cyclic handler ID	
		ID	maxmplid;	Maximum variable-size memory pool ID		ID	maxalmid;	Maximum alarm handler ID	
		ID	maxmpfid;	Maximum fixed-size memory pool ID		FN	maxs_fncd;	Maximum function code of extended service call	
		HNO	maxcycno;	Maximum cyclic handler number		} T_RCFG;			
		HNO	maxalmno;	Maximum alarm handler number					
		DVN	maxiodvn;	Maximum I/O number					
		FN	maxs_fncd;	Maximum extended SVC function code					
		} T_RCFG;							
4	System down				System down				
	vsys_dwn	void vsys_dwn (W type, ER ercd, VW inf1, VW inf2);			vsys_dwn ivsys_dwn	void vsys_dwn (W type, ER ercd, VW inf1, VW inf2); void ivsys_dwn (W type, ER ercd, VW inf1, VW inf2);			

	HI7000 Series (µITRON 3.0)		HI7000/4 Series (µITRON 4.0)		
	Function		Function		
	SVC	API	SVC	API	Change
5	Define extended SVC handler		Define extended SVC handler		Packet struct changed. See 3.3.2.
	def_svc	ER ercd = def_svc (FN s_fncd, T_DSVC *pk_dsvc);	def_svc idef_svc	ER ercd = def_svc (FN fncd, T_DSVC *pk_dsvc); ER ercd = idef_svc (FN fncd, T_DSVC *pk_dsvc);	
		Packet struct typedef struct t_dsvc { ATR svcatr; Extended SVC handler attribute FP svchdr; Extended SVC handler address } T_DSVC;		Packet struct typedef struct t_dsvc { ATR svcatr; Extended service call routine attribute FP svcrtn; Extended service call routine address } T_DSVC;	
6	Define extended SVC handler (assign extended function code automatically)		—		Deleted. See 3.1.4.
	vasn_svc	ER ercd = vasn_svc (FN *p_s_fncd, T_DSVC *pk_dsvc);	—	Not available	
7	Terminate extended SVC handler		—		Deleted
	ret_svc	void ret_svc (ER ercd);	—	Not available	
8	Issue extended SVC		Issue extended SVC		SVC name changed. Can be issued in a non-task context. See 3.3.2.
	vsys_cal	ER ercd = vsys_cal (FN fncd, VW para1, VW para2, VW para3, VW para4); Operation Cannot be issued in a non-task context.	cal_svc ical_svc	ER_UINT ercd = cal_svc (FN fncd, …); ER_UINT ercd = ical_svc (FN fncd, …); Operation Can be issued in a non-task context.	
9	Acquire trace information		Acquire trace information		SVC name changed.
	vget_trc	ER ercd = vget_trc (VW para1, VW para2, VW para3, VW para4);	vget_trc ivget_trc	ER ercd = vget_trc (VW para1, VW para2, VW para3, VW para4); ER ercd = ivget_trc (VW para1, VW para2, VW para3, VW para4);	
10	Acquire start of interrupt handler as trace information		Acquire start of interrupt handler as trace information		SVC name changed.
	vbgn_int	ER ercd = vbgn_int (UINT dintno);	ivbgn_int	ER ercd = ivbgn_int (UINT dintno);	
11	Acquire end of interrupt handler as trace information		Acquire end of interrupt handler as trace information		SVC name changed.
	vend_int	ER ercd = vend_int (UINT dintno);	ivend_int	ER ercd = ivend_int (UINT dintno);	
12	Restart system		—		Deleted. See 3.1.4.
	vres_sys	void vres_sys();	—	Not available	
13	—		Start kernel		New function
	—	Not available	vsta_knl ivsta_knl	void vsta_knl(); void ivsta_knl();	

4.1.17 System State Management

HI7000 Series (µITRON 3.0)			HI7000/4 Series (µITRON 4.0)		
Function			Function		
	SVC	API	SVC	API	Change
—	—	—	Refer to context		New function
	—	Not available	sns_ctx	BOOL state = sns_ctx();	
—	—	—	Refer to CPU-locked state		New function
	—	Not available	sns_loc	BOOL state = sns_loc();	
—	—	—	Refer to dispatch-disabled state		New function
	—	Not available	sns_dsp	BOOL state = sns_dsp();	
—	—	—	Refer to dispatch-pended state		New function
	—	Not available	sns_dpn	BOOL state = sns_dpn();	

4.1.18 DSP Standby Control [HI7700/4]

HI7000 Series (µITRON 3.0)			HI7000/4 Series (µITRON 4.0)		
Function			Function		
	SVC	API	SVC	API	Change
—	—	—	Change TA_COP0 attribute		New function
	—	Not available	vchg_cop	ER_UINT oldatr = vchg_cop (ATR newatr);	

4.1.19 Memory Pool Management (Page Pool) [HI7700, HI7750]

	HI7000 Series (µITRON 3.0)		HI7000/4 Series (µITRON 4.0)		
	Function		Function		
	SVC	API	SVC	API	Change
1	Initialize MMU			—	Deleted. See 3.1.2.
	vini_ppl	void vini_ppl(void);	—	Not available	
2	Get page block			—	
	vget_pbl	ER ercd = vget_pbl (VP *p_pblk, INT pagcnt, UW pagatr);	—	Not available	
3	Get page block (polling)			—	
	vpget_pbl	ER ercd = vpget_pbl (VP *p_pblk, INT pagcnt, UW pagatr);	—	Not available	
4	Get page block with timeout			—	
	vtget_pbl	ER ercd = vtget_pbl (VP *p_pblk, INT pagcnt, UW pagatr, TMO tmout);	—	Not available	
5	Release page block			—	
	vrel_pbl	ER ercd = vrel_pbl (VP pblk);	—	Not available	
6	Change page attribute			—	
	vchg_pat	ER ercd = vchg_pat (VP pagtop, INT pagcnt, UW pagatr);	—	Not available	
7	Refer to page pool state			—	
	vref_ppl	ER ercd = vref_ppl (T_RPPL *pk_rppl);	—	Not available	
8	Translate to physical memory address			—	
	vtrn_adr	ER ercd = vtrn_adr (VP *p_phyadr, VP pagadr);	—	Not available	

4.1.20 Object Name Management

	HI7000 Series (µITRON 3.0)		HI7000/4 Series (µITRON 4.0)		
	Function		Function		
	SVC	API	SVC	API	Change
1	Get task ID of task name			—	Deleted. See 3.1.14.
	vinq_tsk	ER ercd = vinq_tsk (ID *p_tskid, B *name);	—	Not available	
2	Get semaphore ID of semaphore name			—	
	vinq_sem	ER ercd = vinq_sem (ID *p_semid, B *name);	—	Not available	
3	Get event flag ID of event flag name			—	
	vinq_flg	ER ercd = vinq_flg (ID *p_flgid, B *name);	—	Not available	
4	Get mailbox ID of mailbox name			—	
	vinq_mbx	ER ercd = vinq_mbx (ID *p_mbxid, B *name);	—	Not available	
5	Get message buffer ID of message buffer name			—	
	vinq_mbf	ER ercd = vinq_mbf (ID *p_mbfid, B *name);	—	Not available	
6	Get fixed-size memory pool ID of fixed-size memory pool name			—	
	vinq_mpf	ER ercd = vinq_mpf (ID *p_mpfid, B *name);	—	Not available	
7	Get variable-size memory pool ID of variable-size memory pool name			—	
	vinq_mpl	ER ercd = vinq_mpl (ID *p_mplid, B *name);	—	Not available	
8	Get cyclic handler specification number of cyclic handler name			—	
	vinq_cyc	ER ercd = vinq_cyc (ID *p_cycno, B *name);	—	Not available	
9	Get alarm handler specification number of alarm handler name			—	
	vinq_alm	ER ercd = vinq_alm (ID *p_almno, B *name);	—	Not available	

4.1.21 I/O Support

	HI7000 Series (µITRON 3.0)		HI7000/4 Series (µITRON 4.0)		
	Function		Function		
	SVC	API	SVC	API	Change
1	Define I/O handler			—	Deleted. See 2.1.3.
	vdef_dev	ER ercd = vdef_dev (DVN iodvn, T_DDEV *pk_ddev);	—	Not available	
2	Assign I/O number and define I/O handler			—	
	vasn_dev	ER ercd = vasn_dev (DVN *p_iodvn, T_DDEV *pk_ddev);	—	Not available	
3	Input one byte			—	
	get_chr	ER ercd = get_chr (DVN iodvn, W ioun, B *p_chr);	—	Not available	
4	Output one byte			—	
	put_chr	ER ercd = put_chr (DVN iodvn, W ioun, B chr);	—	Not available	
5	Input one line			—	
	get_lin	ER ercd = get_lin (DVN iodvn, W ioun, B *lin, W linsz);	—	Not available	
6	Output one line			—	
	put_lin	ER ercd = put_lin (DVN iodvn, W ioun, B *lin);	—	Not available	
7	Control CIO attribute			—	
	ctl_cio	ER ercd = ctl_cio (DVN iodvn, W ioun, H ioatrcd, VW ioctl);	—	Not available	
8	Refer to CIO state			—	
	ref_cio	ER ercd = ref_cio (DVN iodvn, W ioun, H ioatrcd, VW *p_iosts);	—	Not available	
9	Request GIO			—	
	req_gio	ER ercd = req_gio (DVN iodvn, T_RQGIO *pk_rqgio);	—	Not available	

4.2 Error Code Differences

The following list shows the comparison of the error codes for the system calls (service calls) between the HI7000 series and the HI7000/4 series.

	HI7000 Series (µITRON 3.0)			HI7000/4 Series (µITRON 4.0)			
No.	Error Code (Mnemonic)	Error Code Value		Error Code (Mnemonic)	Error Code Value		Description
1	E_OK	H'00000000	(D'0)	E_OK	H'00000000	(D'0)	Normal termination
2	E_NOMEM	H'ffffffdf	(–D'10)	E_NOMEM	H'fffffff6	(–D'33)	Memory insufficient
3	E_NOSPT	H'fffffff7	(–D'17)	E_NOSPT	H'ffffffef	(–D'9)	Unsupported function (function is not installed)
4	E_RSFN	H'fffffff6	(–D'20)	E_RSFN	H'ffffffec	(–D'10)	System call is not included
5	E_RSATR	H'fffffff5	(–D'24)	E_RSATR	H'ffffffe8	(–D'11)	Reserved attribute (invalid attribute)
6	E_PAR	H'ffffffef	(–D'33)	E_PAR	H'ffffffdf	(–D'17)	Parameter error
7	E_ID	H'ffffffee	(–D'35)	E_ID	H'ffffffdd	(–D'18)	Invalid ID number
8	E_NOEXS	H'ffffffd6	(–D'52)	E_NOEXS	H'ffffffcc	(–D'42)	Object does not exist
9	E_OBJ	H'ffffffd7	(–D'63)	E_OBJ	H'ffffffc1	(–D'41)	Object state is invalid
10	E_CTX	H'ffffffe7	(–D'69)	E_CTX	H'ffffffbb	(–D'25)	Context error
11	E_QOVR	H'ffffffd5	(–D'73)	E_QOVR	H'ffffffb7	(–D'43)	Queuing or nest overflow
12	E_DLT	H'ffffffcd	(–D'81)	E_DLT	H'ffffffaf	(–D'51)	Waiting object deleted
13	E_TMOUT	H'ffffffce	(–D'85)	E_TMOUT	H'ffffffab	(–D'50)	Polling failed or timeout
14	E_RLWAI	H'ffffffcf	(–D'86)	E_RLWAI	H'ffffffaa	(–D'49)	WAITING state is forcibly cancelled
15	EV_NFOUND	H'ffffff1f	(–D'225)	—	—	—	An unused ID or specification number does not exist
16	—	—	—	E_NOID	H'ffffffde	(–D'34)	No ID available
17	EV_ILBLK	H'ffffff1e	(–D'226)	—	—	—	Returns illegal memory block
18	—	—	—	E_ILUSE	H'ffffffe4	(–D'28)	Illegal use of service call

EV_NFOUND is deleted; E_NOID has the same meaning as EV_NFOUND and can be used instead of EV_NFOUND.

The following shows the system calls to return EV_NFOUND and their substitute service calls.

No.	System Call to Return EV_NFOUND	Equivalent Service Call in HI7000/4 Series	Error Code Substituting for EV_NFOUND
1	cre_tsk	cre_tsk, icre_tsk	E_NOID
2	vcre_tsk	(Deleted, no substitute)	(Deleted, no substitute)
3	vscr_tsk	vscr_tsk, ivscr_tsk (substitute)	E_NOID
4	cre_flg	cre_flg, icre_flg	E_NOID
5	vasn_flg	acre_flg, iacre_flg (substitute)	E_NOID
6	cre_mbx	cre_mbx, icre_mbx	E_NOID
7	vasn_mbx	acre_mbx, iacre_mbx (substitute)	E_NOID
8	cre_mbf	cre_mbf, icre_mbf	E_NOID
9	vasn_mbf	iacre_mbf, iacre_mbf (substitute)	E_NOID
10	cre_mpf	cre_mpf, icre_mpf	E_NOID
11	vasn_mpf	acre_mpf, iacre_mpf (substitute)	E_NOID
12	cre_mpl	cre_mpl, icre_mpl	E_NOID
13	vasn_mpl	acre_mpl, iacre_mpl (substitute)	E_NOID
14	def_cyc	cre_cyc, icre_cyc (substitute)	E_NOID
15	vasn_cyc	acre_cyc, iacre_cyc (substitute)	E_NOID
16	def_alm	cre_alm, icre_alm (substitute)	E_NOID
17	vasn_alm	acre_alm, iacre_alm (substitute)	E_NOID
18	vdef_dev	(Deleted, no substitute)	(Deleted, no substitute)
19	vasn_dev	(Deleted, no substitute)	(Deleted, no substitute)
20	def_svc	def_svc	(No error code)
21	vasn_svc	(Deleted, no substitute)	(Deleted, no substitute)

The error indicated by EV_ILBLK is included in another error code.

The following shows the system calls to return EV_ILBLK, their substitute service calls, and the error code including the EV_ILBLK error.

No.	System Call to Return EV_ILBLK	Equivalent Service Call in HI7000/4 Series	Error Code Substituting for EV_ILBLK
1	rel_blf	rel_mpf	Included in E_PAR
2	rel_blk	rel_mpl	Included in E_PAR
3	vrel_pbl	(Deleted, no substitute)	—
4	vchg_pat	(Deleted, no substitute)	—
5	vtrn_adr	(Deleted, no substitute)	—

The following shows the service calls for which a new error code is added, an error code is deleted, or an error code is replaced with another code.

No.	Service Call	Error Code in HI7000 Series	Error Code in HI7000/4 Series
1	sus_tsk	(None)	E_CTX (added)
2	wai_flg	E_OBJ	E_ILUSE
3	pol_flg	E_OBJ	E_ILUSE
4	ipol_flg	E_OBJ	E_ILUSE
5	twai_flg	E_OBJ	E_ILUSE
6	loc_cpu	E_CTX	(Deleted)
7	unl_cpu	E_CTX	(Deleted)
8	chg_ims	E_CTX	(Deleted)

The following shows the differences in error codes that can be detected only when the parameter check function is enabled.

	HI7000 Series			HI7000/4 Series		
No.	SVC Name	Error Code	Description	SVC Name	Error Code	Description
1	Create task cre_tsk vcre_tsk vscr_tsk	E_RAR	Parameter error (pk_ctsk is not a multiple of four, p_tskid or task is an odd number, stksz is not a multiple of four, stksz ≤ 0, itskpri ≤ 0, itskpri > hi_maxtskpri, or quantum < 0)	Create task cre_tsk, icre_tsk acre_tsk, iacre_tsk vscr_tsk, ivcsr_tsk	E_RAR	Parameter error (pk_ctsk is not a multiple of four, task is an odd number, stksz is not a multiple of four, stksz = 0, stksz ≥ 0x80000000, itskpri ≤ 0, itskpri > CFG_MAXTSKPRI, or stk is not a multiple of four if stk is not NULL (only in HI7000/4))
2	Start task sta_tsk	E_ID	Invalid ID number (tskid ≤ 0 or tskid > hi_maxtskid)	Start task act_tsk, iact_tsk	E_ID	Invalid ID number (tskid < 0, tskid > CFG_MAXTSKID, or tskid = TSK_SELF(0) is specified in a non-task context)
3	Rotate ready queue rot_rdq	E_PAR	Parameter error (tskpri < 0 or tskpri > hi_maxtskpri)	Rotate ready queue rot_rdq, irot_rdq	E_PAR	Parameter error (tskpri < 0, tskpri > CFG_MAXTSKPRI, or tskpri = TPRI_SELF(0) is specified in a non-task context)
4	Refer to task state ref_tsk	E_PAR	Parameter error (pk_rtsk is not a multiple of four, p_tskstat is an odd number, or p_tskpri is an odd number)	Refer to task state ref_tsk, iref_tsk	E_PAR	Parameter error (pk_rtsk is not a multiple of four)
		E_ID	Invalid ID number (tskid ≤ 0, tskid > hi_maxtskid, or tskid = 0)		E_ID	Invalid ID number (tskid < 0, tskid > CFG_MAXTSKID, or tskid = TSK_SELF(0) is specified in a non-task context)
5	Suspend task sus_tsk	E_ID	Invalid ID number (tskid ≤ 0 or tskid > hi_maxtskid)	Suspend task sus_tsk, isus_tsk	E_ID	Invalid ID number (tskid < 0, tskid > CFG_MAXTSKID, or tskid = TSK_SELF(0) is specified in a non-task context)
6	Resume task rsm_tsk	E_ID	Invalid ID number (tskid ≤ 0 or tskid > hi_maxtskid)	Resume task rsm_tsk, irsm_tsk	E_ID	Invalid ID number (tskid < 0 or tskid > CFG_MAXTSKID)
7	Wake up task wup_tsk	E_ID	Invalid ID number (tskid ≤ 0 or tskid > hi_maxtskid)	Wake up task wup_tsk	E_ID	Invalid ID number (tskid < 0, tskid > CFG_MAXTSKID, or tskid = TSK_SELF(0) is specified in a non-task context)
8	Cancel wakeup task can_wup	E_PAR	Parameter error (p_wupcnt is not a multiple of four)	Cancel wakeup task can_wup, ican_wup	—	—
		E_ID	Out of ID range (tskid < 0, tskid > hi_maxtskid, or tskid = 0)		E_ID	Out of ID range (tskid < 0, tskid > CFG_MAXTSKID, or tskid = TSK_SELF(0) is specified in a non-task context)
9	Create semaphore cre_sem	E_NOSPT	Unsupported function (task priority function is not installed)	Create semaphore cre_sem, icre_sem acre_sem, iacre_sem	—	—
		E_PAR	Parameter error (p_semid is an odd number, pk_csem is not a multiple of four, isemcnt < 0, or isemcnt > H'ffff)		E_PAR	Parameter error (pk_csem is not a multiple of four, maxsem = 0, maxsem > H'ffff, isemcnt < 0, or isemcnt > maxsem)
10	Refer to semaphore state ref_sem	E_PAR	Parameter error (pk_rsem is not a multiple of four, p_wtskid is an odd number, or p_semcnt is not a multiple of four)	Refer to semaphore state ref_sem, iref_sem	E_PAR	Parameter error (pk_rsem is not a multiple of four)
11	Create event flag cre_flg	E_PAR	Parameter error (p_flgid is an odd number or pk_cflg is not a multiple of four)	Create event flag cre_flg, icre_flg	E_PAR	Parameter error (pk_cflg is not a multiple of four)
12	Create mailbox cre_mbx vasn_mbx	E_NOSPT	Unsupported function (task priority function is not installed)	Create mailbox cre_mbx, icre_mbx acre_mbx, iacre_mbx	—	—
		E_PAR	Parameter error (pk_cmbx is not a multiple of four or p_mbxid is an odd number)		E_PAR	Parameter error (pk_cmbx is not a multiple of four, maxmpri ≤ 0, or maxmpri > CFG_MAXMSGPRI)
13	Create message buffer cre_mbf vasn_mbf	E_NOSPT	Unsupported function (task priority function is not installed)	Create message buffer cre_mbf, icre_mbf acre_mbf, iacre_mbf	—	—
		E_PAR	Parameter error (pk_cmbf is not a multiple of four, bufsz is not a multiple of four or smaller than 0, bufsz is smaller than eight bytes, maxmsz is 0 or smaller, maxmsz+4 > bufsz when bufsz is not 0, or p_mbfid is an odd number)		E_PAR	Parameter error (pk_cmbf is not a multiple of four, mbfsz is not a multiple of four, maxmsz = 0, maxmsz ≥ H'80000000, or maxmsz+4 > mbfsz when mbfsz is not 0)
14	Receive message from message buffer rcv_mbf, prcv_mbf trcv_mbf	E_PAR	Parameter error (p_msgsz is not a multiple of four, msg is not a multiple of four, or tmout ≤ –2)	Receive message from message buffer rcv_mbf, prcv_mbf trcv_mbf	E_PAR	Parameter error (msg is not a multiple of four or tmout ≤ –2)

	HI7000 Series			HI7000/4 Series		
No.	SVC Name	Error Code	Description	SVC Name	Error Code	Description
15	Define interrupt handler def_int	E_PAR	Parameter error (pk_dint is not a multiple of four or inthdr is an odd number) [HI7000] Number reserved by the system (16 ≤ dintno ≤ 31) [HI7700, HI7750] Invalid exception code (dintno is not a multiple of H'20 or dintno > hi_maxexccd)	Define interrupt handler def_inh, idef_inh	E_PAR	Parameter error (pk_dinh is not a multiple of four or inthdr is an odd number) [HI7000/4] Invalid number specification (16 ≤ inhno ≤ 31 or inhno > CFG_MAXVCTNO) [HI7700/4, HI7750/4] Invalid number specification (inhno is not a multiple or H'20, inhno = H'160, or inhno > CFG_MAXVCTNO)
16	Issue extended SVC vsys_cal	—	—	Issue extended SVC cal_svc, ical_svc	E_RSFN	fncd is invalid or cannot be used.
17	Create fixed-size memory pool cre_mpf vasn_mbf	E_NOSPT	Unsupported function (task priority function is not installed)	Create fixed-size memory pool cre_mpf, icre_mpf acre_mpf, iacre_mpf	—	—
		E_PAR	Parameter error (p_mpfid is an odd number, pk_cmpf is not a multiple of four, mpfcnt ≤ 0, or blfsz is not a multiple of four or 0 or smaller)		E_PAR	Parameter error (pk_cmpf is not a multiple of four, blkcnt = 0, blksz is not a multiple of four, blksz = 0, mpf is not a multiple of four when mpf is not NULL (only in HI7000/4), or mpfmb is not a multiple of four (when CFG_MPFMANAGE is checked in HI7000/4))
18	Create variable-size memory pool cre_mpl	E_PAR	Parameter error (pk_cmpl is not a multiple of four, mplsz is not a multiple of four or is smaller than 20 bytes, or p_mplid is an odd number)	Create variable-size memory pool cre_mpl, icre_mpl	E_PAR	Parameter error (pk_cmpl is not a multiple of four, mplsz is not a multiple of four, mplsz < 20, mplsz ≥ H'80000000, or mpl is not a multiple of four when mpl is not NULL (only in HI7000/4))
19	Set system clock set_tim	E_PAR	Parameter error (pk_tim is not a multiple of four or value specified by pk_tim is negative)	Set system clock set_tim, iset_tim	E_PAR	Parameter error (p_systim is not a multiple of four)
20	Delay task dly_tsk	E_PAR	Parameter error (dlytim < 0)	Delay task dly_tsk	—	—
21	Refer to cyclic handler state	—	—	Refer to cyclic handler state	E_ID	Invalid ID number (cycid ≤ 0 or cycid > CFG_MAXCYCID)

Section 5 Difference in Configuration Operation

In the HI7000 series, two files, kernel build file and setup file, are created; in the HI7000/4 series, the configurator should be used to create the equivalent files.

The configurator is a tool for specifying kernel operation parameters. The configurator generates several C source files according to the user settings. The created files and application programs are built (compiled and linked) into a system (load module).

The configurator window consists of a list window of configuration information input parts (on the left side) and a configuration information input window (on the right side). Input data in the configuration information input window and execute the Configuration File Creation command with the menu or the tool button. The configuration files are then created.

For details, refer to the user's manual of the HI7000/4 series.

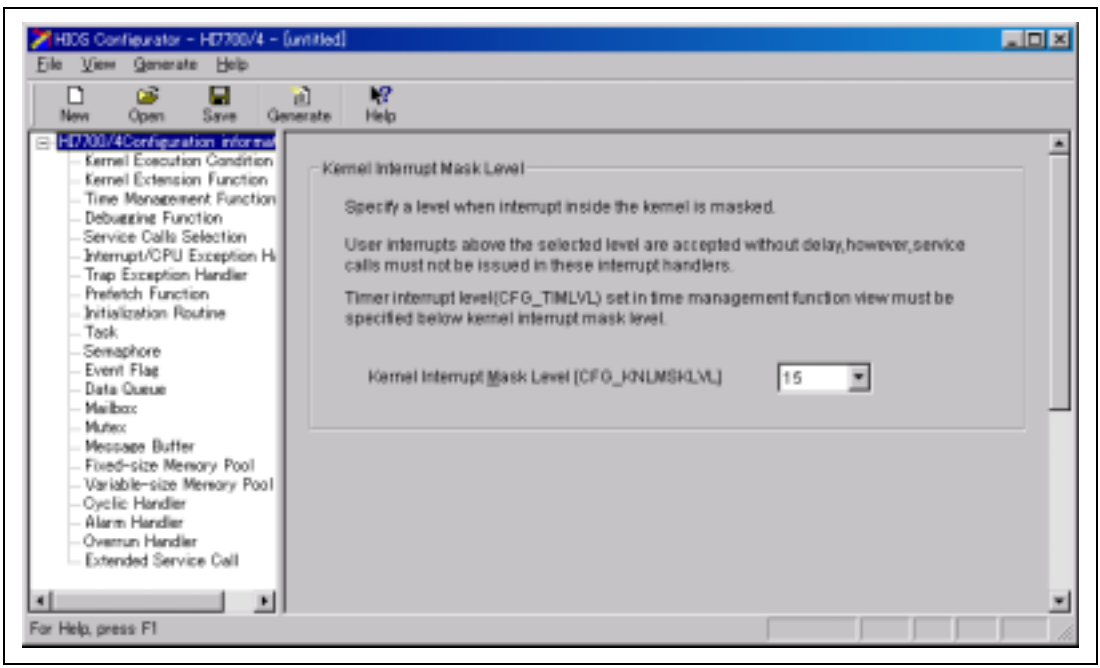

Figure 5.1 HI7000/4 Series Configurator Window

μITRON Specification OS Transition Manual
Transition Guide from HI7000 Series to HI7000/4 Series

Publication Date: Rev.1.00, Sep. 14, 2005
Published by: Sales Strategic Planning Div.
Renesas Technology Corp.
Edited by: Customer Support Department
Global Strategic Communication Div.
Renesas Solutions Corp.

Renesas Technology Corp. Sales Strategic Planning Div. Nippon Bldg., 2-6-2, Ohte-machi, Chiyoda-ku, Tokyo 100-0004, Japan

RENESAS SALES OFFICES

http://www.renesas.com

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

Renesas Technology America, Inc.
450 Holger Way, San Jose, CA 95134-1368, U.S.A
Tel: <1> (408) 382-7500, Fax: <1> (408) 382-7501

Renesas Technology Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, United Kingdom
Tel: <44> (1628) 585-100, Fax: <44> (1628) 585-900

Renesas Technology Hong Kong Ltd.
7th Floor, North Tower, World Finance Centre, Harbour City, 1 Canton Road, Tsimshatsui, Kowloon, Hong Kong
Tel: <852> 2265-6688, Fax: <852> 2730-6071

Renesas Technology Taiwan Co., Ltd.
10th Floor, No.99, Fushing North Road, Taipei, Taiwan
Tel: <886> (2) 2715-2888, Fax: <886> (2) 2713-2999

Renesas Technology (Shanghai) Co., Ltd.
Unit2607 Ruijing Building, No.205 Maoming Road (S), Shanghai 200020, China
Tel: <86> (21) 6472-1001, Fax: <86> (21) 6415-2952

Renesas Technology Singapore Pte. Ltd.
1 Harbour Front Avenue, #06-10, Keppel Bay Tower, Singapore 098632
Tel: <65> 6213-0200, Fax: <65> 6278-8001

Renesas Technology Korea Co., Ltd.
Kukje Center Bldg. 18th Fl., 191, 2-ka, Hangang-ro, Yongsan-ku, Seoul 140-702, Korea
Tel: <82> 2-796-3115, Fax: <82> 2-796-2145

Renesas Technology Malaysia Sdn. Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No.18, Jalan Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: <603> 7955-9390, Fax: <603> 7955-9510

µITRON Specification OS Transition Manual
Transition Guide from HI7000 Series to HI7000/4 Series

Application Note

Renesas Electronics Corporation
1753, Shimonumabe, Nakahara-ku, Kawasaki-shi, Kanagawa 211-8668 Japan

REJ05B0764-0100