HITACHI SEMICONDUCTOR TECHNICAL UPDATE

Classification of Production	Development Environment			No	TN-OS*-058A/E	
THEME	HI7700/4 V1.1.00 Revision up		Classification of Information	1. (Spec change) 2. Supplement of Documents 3. Limitation of Use 4. Change of Mask 5. Change of Production Line		
PRODUCT NAME	HS0770ITI41SRE, HS0770ITI41SRB, HS0770ITI41SRS, HS0770ITI41SRE-E, HS0770ITI41SRB-E, HS0770ITI41SRS-E	Lot No. V1.00r1, V1.01r1, V1.0Ar1, V1.0Br1, V1.0Cr1	Reference Documents	HI7000/4 Series User's Manual(ADE-702-248)	Rev. Rev.1	Effective Date Forever

We release HI7700/4 V1.1.00.

If the following conditions are satisfiled, please request V1.1.00 to us. If the following conditions are not satisfiled, do not have to use V1.1.00.

(1) Use SH-3 with 32k-bytes cache

(2) Use 32k-bytes mode for cache

(3) Use either vini_cac, vclr_cac, or vfls_cac service calls

The changes of the V1.1.00 are shown as follows;

(1) The cache support service calls supports 32kB cache.

(2) When vdef_trp is selected and CFG_MAXTRPNO is less than 24 in the configurator, the vsta_knl writes to kernel stack illegally. This bug is corrected. Note, this bug does not raise abnormal phenomenon because the data in the written address are always invalid.

(3) kernel version is V1.1.00. T_VER.prver, which is returned from the ref_ver system call, was modified from 0x010c to 0x0110. Similarly, macro TKERNEL_PRVER for configuring the kernel was also modified from 0x010c to 0x0110.

(4) In connection with (1), "HI7000/4 series User's Manual supplemental information" is updated to Rev.4.

Attached document : HI7000/4 series User's Manual supplemental information (Rev.4)

HI7000/4 series User's Manual supplemental information (Rev.4)

This document supplements the "HI7000/4 User's Manual (ADE-702-248)".

1. Supplemental Information

1.1 Basic Data Type

The basic data type defined in the HI7000/4 series are shown in the table1.

Table1 Basic Data Type

No.	Data type	Meaning	No.	Data type	Meaning
1	B	8-bit signed integer	21	PRI	16-bit signed integer
2	H	16-bit signed integer	22	SIZE	32-bit unsigned integer
3	W	32-bit signed integer	23	TMO	32-bit signed integer
4	UB	8-bit unsigned integer	24	RELTIM	32-bit unsigned integer
5	UH	16-bit unsigned integer	25	SYSTIM	A structure which contains following members
6	UW	32-bit unsigned integer			Upper : 16-bit unsigned integer
7	VB	8-bit signed integer *			Lower : 32-bit unsigned integer
8	VH	16-bit signed integer *	26	VP_INT	32-bit signed integer *
9	VW	32-bit signed integer *	27	ER_BOOL	32-bit signed integer
10	VP	pointer to void type	28	ER_ID	32-bit signed integer
11	FP	pointer to void type function	29	ER_UINT	32-bit signed integer
12	INT	32-bit signed integer	30	TEXPTN	32-bit unsigned integer
13	UINT	32-bit unsigned integer	31	FLGPTN	32-bit unsigned integer
14	BOOL	32-bit signed integer	32	RDVPTN	32-bit unsigned integer
15	FN	32-bit signed integer	33	RDVNO	32-bit unsigned integer
16	ER	32-bit signed integer	34	OVRTIM	32-bit unsigned integer
17	ID	16-bit signed integer	35	INHNO	32-bit unsigned integer
18	ATR	32-bit unsigned integer	36	EXCNO	32-bit unsigned integer
19	STAT	32-bit unsigned integer	37	IMASK	32-bit unsigned integer
20	MODE	32-bit unsigned integer			

* When the variable values of these data types are referred to or substituted, the type must be explicitly converted (casted).

1.2 Processing precedence

In the HI7000/4 series, each processing unit is processed with the following precedences:

(1) Interrupt handler, time event handler, and CPU exception handler

(2) Dispatcher (part of kernel processing)

(3) Task

The dispatcher is a kernel processing that switches a task to be executed.

The precedence of an interrupt handler becomes higher when an interrupt level is higher.

The precedence of a time event handler is the same as a timer interrupt level (CFG_TIMINTLVL).

The precedence of a CPU exception handler is higher than that of the processing where the CPU exception occurred and of the dispatcher. The precedence of a CPU exception handler is also lower than that of other processings which have the higher precedences than those where the CPU exception occurred.

The precedence between tasks depends on the priority of these tasks.

The precedence of an extended service call routine is higher than that of the processing where the extended service call was called. The precedence of an extended service call routine is also lower than that of other processings which have the higher precedences than those where the extended service call was called.

The precedence of a task's exception processing routine is higher than that of the task and lower than that of other higher-level tasks.

When the following service calls are called, the precedence which does not apply above can be temporarily generated:

(a) When dis_dsp is called, the precedence will be the middle of (1) and (2) above. The state returns to former state by calling dis_dsp.
(b) When loc_cpu or iloc_cpu is called, the precedence will be the same as that of the interrupt handler of which interrupt level is same as CFG_KNLMSKLVL. The state returns to former state by calling unl_cpu or iunl_cpu.
(c) While the interrupt mask level is changed to other than 0 by chg_ims, the precedence is same as an interrupt handler which has the same level.

1.3 Additional service calls which begin with "i"

Tabel2 shows added service calls. The specification of the service calls are same as the base service calls except "i".
The added service calls are supported in V1.0Ar1 or later.

Table2 Additional service calls

Added service call	Base service call	Function	Note
ivsta_knlk	vsta_knl	Start kernel	
ivsys_dwn	vsys_dwn	System down	
ivini_cac	vini_cac	Initialize cache	Only in HI7700/4 and HI7750/4
ivclr_cac	vclr_cac	Clear cache	Only in HI7700/4 and HI7750/4
ivfls_cac	vfls_cac	Flush cache	Only in HI7700/4 and HI7750/4
ivinv_cac	vinv_cac	Invalidate cache	Only in HI7700/4 and HI7750/4

Related descriptions : "3.19.11 Start Kernel"(p.248), "3.19.12 System Down"(p.249), and "3.23 Cache Support Function"(p.277)

1.4 TA_CEILING attribute in the mutex

When the TA_CEILING attrubute is specified, the mutex is managed by "simplified priority control rule". Under this rule, the management which changes the task's current priority to higher value is always operated, but the management which changes the task's priority to lower value is not operated onlly when the task releases all of mutexes.

1.5 System clock

The system clock is expressed as a 48-bit unsigned integer value by using the data type "SYSTIM". The maximum value of the system clock is shown as follows,

[Case of "CFG_TICNUME/CFG_TICDENO <= 1"]
The maximum value = H'7fffffffffff/CFG_TICDENO
[Case of "CFG_TICNUME/CFG_TICDENO > 1"]
The maximum value = H'7fffffffffff

When the system clock exceeds the above maximum value at timer interrupt (isig_tim), the system clock is initialized to 0.
If a value larger than the above maximum value is specified in the set_tim service call, the system operation is not guaranteed.

Related desciption : "3.15 Time Management (System Clock)" (p.204)

1.6 Sample timer driver

The timer interrupt period(T) is calculated by follow expression.

T [sec] = CFG_TICNUME/CFG_TICDDENO/1000 = { (1 / PCLOCK) x DIV} x (COUNTER + 1)

- CFG_TICNUME, CFG_TICDENO : Numerator, denominator of timer tick [ms]
  These are set in the Configurator.
- PCLOCK[Hz] : Frequency which is supplied to the timer module
  This is defined in the ***nnnn***_tmrdef.h
- DIV : Divided ratio by setting of the timer module register
  In the HI7000/4, the DIV which is defined in the ***nnnn***_tmrdrv.c
  In the HI7700/4 and HI7750/4, the DIV is selected automatically in order to PCLOCK

- COUNTER : The number of timer clock counts which expires the period(T)
  In the sample timer drivers, COUNTER is calculated automatically by the following expression that is deduced from the above expression.
  COUNTER = CFG_TICNUME x PCLOCK / CFG_TICDENO / DIV / 1000 - 1

Note, when the caluclated value as COUNTER is larger than the counter register size, the period is illegal. Do confirm this. And, when the caluclated value as COUNTER is not integer, the value is rolled to integer. In this case, the period at run-time is shorter than expected period(CFG_TICNUME/CFG_TICDENO).

Related desciption : "4.10.2 Sample Timer Driver" (p.315)

1.7 Direct interrupt handler (HI7000/4)

When a direct interrupt handler is written in assembly language, do satisfy the specification shown in table 3.

Table 3 The methods to end direct interrupt handler

Item	Secification
Instruction to end	(1) The interrupt level is less than or equal to the kernel interrupt mask level(CFG_KNLMSKLVL) : TRAPA #25 instruction (2) Other than (1) : RTE instruction
Status at ended	The value of all of CPU registers except PC and SR must be same as the value at the handler initiated when the instruction to end is executed.

Related desciption : "4.7.2 Direct Interrupt Handler (HI7000/4)" (p.300)

1.8 Reseved TRAP

The TRAPA #16 - #31 instructions are reserved by the HI7000/4 series. An application must not use these instructions. When these instructions are executed, the system will be down except the case in the above 1.7.

1.9 The macro that begins with "TSZ_"

In the µITRON4.0 specification, the area for some object types can be prepared by an application. In the object creation service call such as cre_dtq, application can specify the area address. And the µITRON4.0 specification defines the macro that begins with "TSZ_" for calculating the area size. Note that these are out of the standard profile of the µITRON4.0 specification.

But the current HI7000/4 series don't support this function. Therefore, these macros have no meaning.

The macros defined by the µITRON4.0 specification are shown in table 4.

Table 4 The macros defined by the µITRON4.0 specification

Macro	Object	Purpose
TSZ_DTQ	Data queue	Calculating the data queue area size
TSZ_MPRIHD	Mailbox	Calculating the message queue header area size
TSZ_MBF	Message buffer	Calculating the message buffer area size
TSZ_MPF	Fixed-size memory pool	Calculating the fixed-size memory pool area size
TSZ_MPL	Variable-size memory pool	Calculating the fixed-size memory pool area size

Related desciption : Table 4.1 (p.286)

1.10 Service Calls Selection view of the configurator

(1) In the Service Calls Selection view, there is no selection items of service call that begins with 'i'. When a service call without 'i' is selected, it means that a service call that begins with 'i' is automatically selected.

(2) To use kernel objects except task, cre_??? or def_??? service call which creates or defines the object must be selected in the Service Call Selection view. Otherwise, if cre_??? or def_??? service call is not selected, some items such as maximum object ID are treated as 0 regardless window contents of the Configurator.. The details are shown in table 5.

Table 5 The service calls which have to be selected to use the objects

Objects	Required service calls	The items treated as 0 when required service call is not selected
Semaphore	cre_sem	CFG_MAXSEMID in the Semaphore view
Event flag	cre_flg	CFG_MAXFLGID in the Event Flag view
Data queue	cre_dtq	CFG_MAXDTQID and CFG_DTQSZ in the Data Queue view
Mailbox	cre_mbx	CFG_MAXMBXID in the Mailbox view
Mutex	cre_mtx	CFG_MAXMTXID in the Mutex view
Message buffer	cre_mbf	CFG_MAXMBFID and CFG_MBFSZ in the Message Buffer view
Fixed-size memory pool	cre_mpf	CFG_MAXMPFID and CFG_MPFSZ in the Fixed-size Mempry Pool view
Variable-size memory pool	cre_mpl	CFG_MAXMPLID and CFG_MPLSZ in the Variable-size Memory Pool view
Cyclic handler	cre_cyc	CFG_MAXCYCID in the Cyclic Handler view
Alarm handler	cre_alm	CFG_MAXALMID in the Alarm Handler view
Extended service call routine	def_svc	CFG_MAXSVCCD in the Extended Service Call view

1.11 HI7700/4 Cache support service calls

The table 6 shows caches supported by HI7700/4 cache support service calls, and parameter specification of vini_cac

Table 6 HI7700/4 Cache support service calls

Cache size	Supported kernel	Conditions	Parameters of vini_cac		
			ccr_data	entnum	waynum
8kB	V1.0 or later	Don't use internal RAM mode	Specify in order to device specification	4	128
		Use internal RAM mode		2	128
16kB		---		4	256
32kB *	V1.1.00 or later	32kB mode		4	512
		16kB mode		4	256

* When CCR3 register must be set, set CCR3 register before vini_cac. And don't accress cache between CCR3 register setting and vini_cac. For example, set CCR3 register and vini_cac in cache disable state.

2. Manual Correction

2.1 Figure 2.10 (p.29)

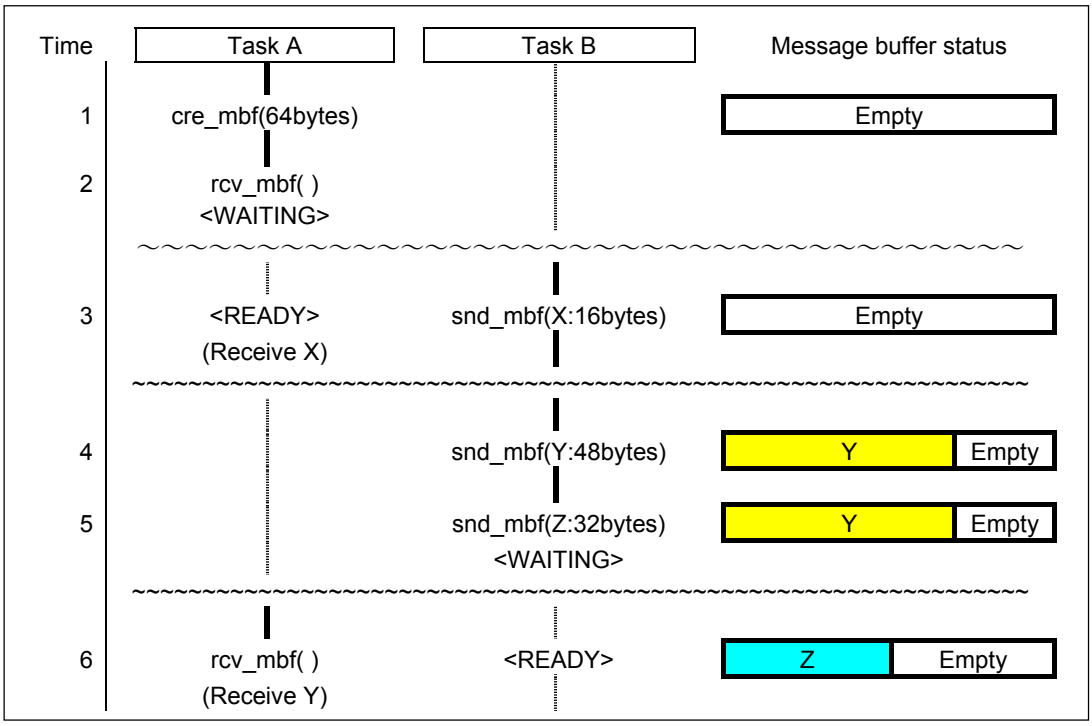

Figure 2.10 Example of Using a Message Buffer

Description:

Bold lines represent executed process. The following describes the message buffer operation with respect to time.

1. Task A creates a 64-byte message buffer, which deals the message where the maximum size is 48 bytes, by issuing cre_mbf.
2. Task A receives a message by preparing the 48-byte memory and issuing rcv_mbf. Task A is placed in the WAITING state since there are no messages in the message buffer.
3. Task B sends a 16-byte message X by issuing snd_mbf. At this time, task A is released from the WAITING state and message X is copied to the memory prepared by task A. Task A receives message size 16 as the return parameter.
4. Task B sends a 48-byte message Y by issuing snd_mbf. At this time, since there are no tasks waiting for a message, message Y is copied to the message buffer. In this case, the kernel uses 4-byte message buffer area to copy message Y to the message buffer, however this is not indicated in figure 2.10.
5. Task B attempts to send 32-byte message Z by issuing snd_mbf. At this time, since the message buffer is not large enough to store message Z, task B is placed in the WAITING state.
6. Task A prepares the 48-byte memory and issues rcv_mbf to receive a message, 48-byte message Y stored in the message buffer is copied to the memory prepared by task A. Task A receives message size 48 as the return parameter. At this time, since the message buffer has sufficient space to store message Z, task B is released from the WAITING state and message Z is copied to the message buffer.

2.2 Error Code Type in Figure 3.2 (p.58)

Before	After
Error Code Type [k] indicates an error that is always detected. [p] indicates an error that is detected only when the kernel with parameter check function (CFG_PARCHK) is installed.	Error Code Type [k] indicates an error that is always detected. [p] indicates an error that is detected only when the kernel with parameter check function (CFG_PARCHK) is installed. ***When the conditions which cause [p] type errors occurred in the configuration which does not check CFG_PARCHK, the system operation is not guaranteed. And the cases of [p] type errors are not detected when [k] type errors are detected.***

2.3 E_PAR error description in "3.4.1 Create Task" (p.62)

Before	After
E_PAR [p] Parameter error (pk_ctsk is other than a multiple of four, task is an odd address, stksz is other than a multiple of four, stksz ≤ 0, itskpri ≤ 0, or itskpri > CFG_MAXTSKPRI)	E_PAR [p] Parameter error (pk_ctsk is other than a multiple of four, task is an odd address, stksz is other than a multiple of four, ***stksz == 0, or stksz ≥ H'0x80000000,*** itskpri ≤ 0, or itskpri > CFG_MAXTSKPRI)

2.4 E_ID error description in "3.4.5 Start Task(Start Code Specified)" (p.69)

Before	After
E_ID [p] Invalid ID number (tskid < 0, tskid > CFG_MAXTSKID~~, or tskid = TSK_SELF(0) is specified in a non-task context~~)~~	E_ID [p] Invalid ID number (***tskid≤0***, tskid > CFG_MAXTSKID)

2.5 The last paragraph in "3.4.8 Change Task Priority" (p.75)

Before	After
If the base priority specified in the parameter tskpri is higher than the ceiling priority of one of the mutexes when the object task locks the mutexes with the TA_CEILING attribute, E_ILUSE is returned.	If the base priority specified in the parameter tskpri is higher than the ceiling priority of one of the mutexes when the object task ***locks or wait to lock*** the mutexes with the TA_CEILING attribute, E_ILUSE is returned.

2.6 Add E_PAR error to "3.4.9 Refet to Task Priority"(p.76)

Add
E_PAR [p] Parameter error (p_tskpri is not even)

2.7 E_ID error description in "3.4.5 Start Task(Start Code Specified)" (p.92)

Before	After
E_ID [p] Invalid ID number (tskid < 0, tskid > CFG_MAXTSKID~~, or tskid = TSK_SELF(0) is specified in a non-task context~~)	E_ID [p] Invalid ID number (***tskid≤0***, tskid > CFG_MAXTSKID)

2.8 E_PAR error description in "3.7.1 Create Semaphore" (p.114)

Before	After
E_PAR [p] Parameter error (pk_csem is other than a multiple of four, maxsem ≤ 0, maxsem > H'ffff, isemcnt < 0, or isemcnt > maxsem)	E_PAR [p] Parameter error (pk_csem is other than a multiple of four, ***maxsem == 0***, maxsem > H'ffff, isemcnt < 0, or isemcnt > maxsem)

2.9 The data type "T_CFLG" in "3.8.1 Create Event Flag" (p.123)

Before	After
Packet Structure: typedef struct t_cflg { ATR flgatr; +0 4 Event flag attribute UINT iflgptn; +4 4 Initial value of event flag } T_CFLG;	Packet Structure: typedef struct t_cflg { ATR flgatr; +0 4 Event flag attribute ***FLGPTN*** iflgptn; +4 4 Initial value of event flag } T_CFLG;;

2.10 Table 3.28 Service Calls for Data Queue (p.134)

The prcv_dtq can be called from non-task context. The correct contents of Table 3.28 (abstract) is shown as follows;

			System State
Service Call		Description	T/N/E/D/U/L/C
		(omission)	
prcv_dtq	[S]	Polls and receives data from data queue	***T/E/D/U***
		(omission)	

2.11 Title correction in "3.9.3 Send Data to Data Queue" (p.140)

Before	3.9.3 Send Data to Data Queue (snd_dtq, psnd_dtq, ipsnd_dtq, tsnd_dtq)
After	3.9.3 Send Data to Data Queue (snd_dtq, psnd_dtq, ipsnd_dtq, tsnd_dtq***, fsnd_dtq, ifsnd_dtq***)

2.12 Add E_ILUSE error in "3.9.3 Send Data to Data Queue" (p.140)

Add
E_ILUSE [k] Illegal use of service call(fsnd_dtq, ifsnd_dtq is issued for the data queue which dtqcnt is 0)

2.13 E_PAR error description in "3.10.3 Create Message Buffer" (p.152)

Before	After
E_PAR [p] Parameter error (pk_msg is other than a multiple of four or the first four bytes of the message is other than 0) [k] (msgpri > CFG_MAXMSGPRI)	E_PAR [p] Parameter error (pk_msg is other than a multiple of four or the first four bytes of the message is other than 0) [k] (***msgpri ≤ 0,*** msgpri > CFG_MAXMSGPRI)

2.14 The last paragraph in "3.11.1 Create Mutex" (p.161)

Before	After
ceilpri specifies the ceiling priority for the mutex to be created. The range of values which can be specified is 0 to CFG_MAXTSKPRI.	ceilpri specifies the ceiling priority for the mutex to be created. The range of values which can be specified is ***1 to CFG_MAXTSKPRI***.

2.15 E_PAR error description in "3.12.1 Create Message Buffer" (p.170)

Before	After
E_PAR [p] Parameter error (pk_cmbf is other than a multiple of four, mbfsz is other than a multiple of four, maxmsz ≤ 0, or mbfsz is other than 0 and maxmsz + 4 > mbfsz or mbfsz < 8)	E_PAR [p] Parameter error (pk_cmbf is other than a multiple of four, mbfsz is other than a multiple of four, ***maxmsz = = 0, or maxmsz ≥ H'80000000*** mbfsz is other than 0 and maxmsz + 4 > mbfsz or mbfsz < 8)

2.16 Add description in "3.12.3 Send Message to Message Buffer" (p.173)

Add
ipsnd_mbf can also be issued from a non-task context. Since the priority of a non-task context is higher than that of a task, when the target message buffer has TA_TPRI attribute and the buffer has enough free size for required size (msgsz + 4), the specified message is copied to the buffer even if there exists a task that has been waiting to be transmitted.

2.17 E_PAR error description in "3.13.1 Create Fixed-Size Memory Pool" (p.184)

Before	After
E_PAR [p] Parameter error (pk_cmpf is other than a multiple of four, blkcnt = 0, blksz is other than a multiple of four or blksz = 0) [k] blkcnt × blkcnt exceeds 32-bit range	E_PAR [p] Parameter error (pk_cmpf is other than a multiple of four, blkcnt = 0, blksz is other than a multiple of four or blksz = 0) [k] ***blkcnt × (blksz + 4)*** exceeds 32-bit range

2.18 E_PAR error description in "3.14.1 Create Variable-Size memory Pool" (p.195)

Before	After
E_PAR [p] Parameter error (pk_cmpl is other than a multiple of four, mplsz is other than a multiple of four, or mplsz < 20)	E_PAR [p] Parameter error (pk_cmpl is other than a multiple of four, mplsz is other than a multiple of four, or mplsz < 20***, mplsz≥H'80000000***)

2.19 E_ID and E_NOEXS error description in "3.18.2 Start Overrun Handler" (p.231)

Before	After
E_ID [p] Invalid ID number (tskid ≤ 0, tskid > CFG_MAXTSKID, or tskid = TSK_SELF(0) is specified in a non-task context) E_NOEXS [k] Undefined (Overrun handler specified by ovrid does not exist)	E_ID [p] Invalid ID number (***tskid < 0***, tskid > CFG_MAXTSKID, or tskid = TSK_SELF(0) is specified in a non-task context) E_NOEXS [k] Undefined (***Task specified by tskid does not exist***)

2.20 E_ID error description in "3.18.3 Stop Overrun Handler Operation" (p.233)

Before	After
E_ID [p] Invalid ID number (tskid ≤ 0, tskid > CFG_MAXTSKID, or tskid = TSK_SELF(0) is specified in a non-task context)	E_ID [p] Invalid ID number (***tskid < 0***, tskid > CFG_MAXTSKID, or tskid = TSK_SELF(0) is specified in a non-task context)

2.21 E_ID error description in "3.18.4 Refer Overrun Handler State" (p.234)

Before	After
E_ID [p] Invalid ID number (tskid ≤ 0, tskid > CFG_MAXTSKID)	E_ID [p] Invalid ID number (***tskid < 0***, tskid > CFG_MAXTSKID***, or*** ***tskid = TSK_SELF(0) is specified in a*** ***non-task context***)

2.22 Add description in "3.18.4 Refer Overrun Handler State" (p.235)

Add
By specifying tskid = TSK_SELF (0), the current task can be specified.

2.23 Cautions of CPU-locked state in "3.19.3 Lock CPU" (p.240)

Before	After
When the system makes a transition to the CPU-locked state by calling service call iloc_cpu in the interrupt handler, service call iunl_cpu must be called to unlock the CPU before returning from the interrupt handler.	***The transition between CPU-Locked state and CPU-unlocked state is occurred only when loc_cpu, iloc_cpu, unl_cpu, iunl_cpu, ext_tsk, or exd_tsk service call is called.*** ***An interupt handler which level is equal or lower than the kernel interrupt mask level(CFG_KNLMSKLVL), time event handler, initialization routine, and task exception routine must unlock the CPU at termination. If the CPU is locked at termination, normal system operation cannot be guaranteed. Note, the CPU at the start of these handlers is unlocked.*** ***If the CPU exception handler changes CPU-locked/unlocked state, the handler must return to former state. If don't return to former state, normal system operation cannot be guaranteed.***

2.24 Cautions of dispatch-disabled state in "3.19.5 Lock CPU" (add) (p.242)

Add
The transition between dispatch-disabled state and dispatch-enabled state is occurred only when dis_dsp, ena_dsp, ext_tsk, or exd_tsk service call is called. If the CPU exception handler changes dispatch-disabled/enabled state, the handler must return to former state. If don't return to former state, normal system operation cannot be guaranteed.

2.25 Data type of the return value from the cal_svc and ical_svc service call

(1) "3.21.2 Call Service Call" (p.262)

Before	After
ER ercd R0 Return value from service call	***ER_UINT*** ercd R0 Return value from service call

(2) Figure4.3 (p.298)

[Before]

ER Svcrtn(VP_INT par1, VP_INT par2)
{
/* Extended service call routine */
 	
return E_OK;
}
	<-Parameters specified by
	cal_svc are passed to the
	extended service call
	routine.
	<- Passes the return value to the caller

[After]

ER_UINT Svcrtn(VP_INT par1, VP_INT par2)
{
/* Extended service call routine */
 	
return E_OK;
}
	<-Parameters specified by
	cal_svc are passed to the
	extended service call
	routine.
	<- Passes the return value to the caller

2.26 "4.3 Reserved Names" (p.289)

The title of this chapter is changed to "4.3 System Reserve".

And the existing explanation is classified as "4.3.1 Reserved Names", and the following contents are added as "4.3.2 Reserved TRAP".

Add
The TRAPA #16 - #31 instructions are reserved by the HI7000/4 series. An application must not use these instructions. When these instructions are executed, the system will be down except the TRAPA #25 at the end of ta direct interrupt handler in the HI7000/4.

2.27 Figure 4.7 (p.304)

[Before]

```
 ... (omitted) ...
TRAPA    #D'25                  <- An interrupt handler with a level equal to or lower
                                   than the kernel interrupt mask level completes execution
                                   with TRAPA #25.
```

[After]

```
 ... (omitted) ...
TRAPA    #D'25                  <- An interrupt handler with a level equal to or lower
                                   than the kernel interrupt mask level completes execution
                                   with TRAPA #25. When the TRAPA #25 instruction is
                                   executed in the task context, the system is down because of
                                   undefined exception.
```

2.28 Figure 5.1 (p.322), Figure 5.2 (p.323), Figure 5.3 (p.327)

Before	After
kernel_cfg.c	***kernel_def.c***
kernel_env.c	***kernel_cfg.c***
kernel_cfg_main.h	***kernel_def_main.h***
kernel_env_main.h	***kernel_cfg_main.h***
*nnnn*_cfg	***nnnn_def***
*nnnn*_env	***nnnn_cfg***

2.29 C language API of the vini_cac service call (p.366)

[Before]

149	vini_cac	[HI7700/4] void vini_cac (UW ccr_data); [HI7750/4] void vini_cac (UW ccr_data, UW entnum, UW waynum);	Initialize Cache

[After]

149	vini_cac	[HI7700/4] void vini_cac ***(UW ccr_data, UW entnum, UW waynum)***; [HI7750/4] void vini_cac ***(UW ccr_data)***;	Initialize Cache