お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

2002年　6月　14日

― 日立半導体技術情報 ―

〒１００－０００４
東京都千代田区大手町2丁目6番2号
(日本ビル)
TEL　(03)5201-5022（ダイヤルイン）
株式会社 日立製作所 半導体グループ

製品分類	開発環境		発行番号	TN-0S*-058A		
題名	HI7700/4 V1.1.00 へのリビジョンアップのお知らせ		情報分類	①. 仕様変更 2. ドキュメント訂正追加等 3. 使用上の注意事項 4. マスク変更 5. ライン変更		
適用製品	HS0770ITI41SRE, HS0770ITI41SRB, HS0770ITI41SRS, HS0770ITI41SRE-E, HS0770ITI41SRB-E, HS0770ITI41SRS-E	対象ロット等 V1.00rl, V1.01rl, V1.0Arl, V1.0Brl, V1.0Crl	間連資料	HI7700/4 シリーズ ユーザーズマニュアル (ADJ-702-298A)　第2版	Rev. 第1版	有効期限 永年

拝啓、貴社益々御清栄のこととお慶び申し上げます。平素は格別の御高配を賜り、感謝申し上げます。
このたび、HI7700/4 を V1.0Crl から V1.1.00 にリビジョンアップ致しました。以下の全条件を満たす場合は、担当営業にリビジョンアップ版の入手をご依頼ください。それ以外の場合には、リビジョンアップは不要です。

(1)32kB キャッシュを搭載した SH-3, SH3-DSP マイコンを使用する
(2)キャッシュを 32k バイトモードで使用する
(3)vini_cac, vclr_cac, vfls_cac のいずれかのサービスコールを使用する

以下に、今回のリビジョンアップでの変更点を示します。

(1) キャッシュサポートサービスコールが、32k バイトのキャッシュに対応しました。
　キャッシュサポートサービスコール：vini_cac, vclr_cac, vfls_cac, vinv_cac
(2) コンフィギュレータで vdef_trp が選択されており、かつ CFG_MAXTRPNO に 24 未満の値を指定していた場合、vsta_knl でカーネルスタック領域に不正にデータライトを行う場合がありましたが、これを対策しました。ただし、不正ライトするスタック領域に有効なデータが格納されているケースはありえないため、対策前でも不具合現象は発生しません。
(3) カーネルのバージョンを V1.1.00 としました。ref_ver システムコールで返る T_VER.prver は、0x010c から 0x0110 に変更しました。同様に、カーネル構成マクロ TKERNEL_PRVER も 0x010c から 0x0110 に変更しました。
(4) (1)に関連し、添付のように「HI7000/4 シリーズユーザーズマニュアル補足説明書」を Rev.4 に改版しました。Rev.4 では、1.11 節に HI7700/4 の vini_cac サービスコールに関する説明を追加しました。

添付資料：HI7000/4 シリーズユーザーズマニュアル補足説明書(Rev.4)

以上

HI7000/4 シリーズ ユーザーズマニュアル(第 2 版)補足説明書(Rev.4)

製品添付の「HI7000/4 シリーズユーザーズマニュアル第 2 版(ADJ-702-298A)」につき、以下に訂正・補足説明を示します。
また、下記の弊社ホームページもあわせてご利用下さい。

http://www.hitachisemiconductor.com/sic/jsp/japan/jpn/Sicd/Japanese/Products/micom/dev_env/index.htm

1. 補足説明

1.1 基本データ型

HI7000/4 シリーズで定義されている基本データ型の詳細を、表 1に示します。

表 1 基本データ型

No.	データ型	定義内容	No.	データ型	定義内容
1	B	符号付き 8 ビット整数	21	PRI	符号付き 16 ビット整数
2	H	符号付き 16 ビット整数	22	SIZE	符号無し 32 ビット整数
3	W	符号付き 32 ビット整数	23	TMO	符号付き 32 ビット整数
4	UB	符号無し 8 ビット整数	24	RELTIM	符号無し 32 ビット整数
5	UH	符号無し 16 ビット整数	25	SYSTIM	以下のメンバから構成される構造体
6	UW	符号無し 32 ビット整数			上位：符号無し 16 ビット整数
7	VB	符号付き 8 ビット整数 *			下位：符号無し 32 ビット整数
8	VH	符号付き 16 ビット整数 *	26	VP_INT	符号付き 32 ビット整数 *
9	VW	符号付き 32 ビット整数 *	27	ER_BOOL	符号付き 32 ビット整数
10	VP	void 型へのポインタ	28	ER_ID	符号付き 32 ビット整数
11	FP	void 型関数へのポインタ	29	ER_UINT	符号付き 32 ビット整数
12	INT	符号付き 32 ビット整数	30	TEXPTN	符号無し 32 ビット整数
13	UINT	符号無し 32 ビット整数	31	FLGPTN	符号無し 32 ビット整数
14	BOOL	符号付き 32 ビット整数	32	RDVPTN	符号無し 32 ビット整数
15	FN	符号付き 32 ビット整数	33	RDVNO	符号無し 32 ビット整数
16	ER	符号付き 32 ビット整数	34	OVRTIM	符号無し 32 ビット整数
17	ID	符号付き 16 ビット整数	35	INHNO	符号無し 32 ビット整数
18	ATR	符号無し 32 ビット整数	36	EXCNO	符号無し 32 ビット整数
19	STAT	符号無し 32 ビット整数	37	IMASK	符号無し 32 ビット整数
20	MODE	符号無し 32 ビット整数			

【注】* これらのデータタイプの変数の値を参照する場合や代入する場合には、明示的に型変換(キャスト)を行う必要があります。

1.2 処理の優先順位

HI7000/4 シリーズでは、各処理単位は以下の優先順位で処理されます。

(1) 割込みハンドラ、タイムイベントハンドラ、CPU 例外ハンドラ
(2) ディスパッチャ (カーネルの処理の一部)
(3) タスク

なお、ディスパッチャとは、実行するタスクを切り換えるカーネルの処理のことです。
割込みハンドラは、割込みレベルが高いほど優先順位が高くなります。
タイムイベントハンドラの優先順位は、タイマ割込みレベル(CFG_TIMINTLVL)と同じとなります。
CPU 例外ハンドラの優先順位は、CPU 例外が発生した処理の優先順位と、ディスパッチャの優先順位のいずれよりも高く、かつ CPU 例外が発生した処理よりも高い優先順位を持つ他のいずれの処理よりも低くなります。
タスクの間の優先順位は、タスクに付与された優先度に従います。
拡張サービスコールルーチンの優先順位は、拡張サービスコールを呼び出した処理の優先順位よりも高く、かつ拡張サービスコールルーチンを呼び出した処理よりも高い優先順位を持つ他のいずれの処理よりも低くなります。
タスク例外処理ルーチンの優先順位は、当該タスクよりも高くかつ他の優先度の高いタスクよりも低くなります。
また以下のサービスコールを呼び出した場合は、一時的に上記に該当しない優先順位を作り出すことができます。

(a)dis_dsp を呼び出すと、その処理の優先順位は上記の(1)と(2)の間となります。この状態は、dis_dsp を呼び出すことによって元に戻ります。
(b)loc_cpu, iloc_cpu を呼び出すと、その処理の優先順位は割込みレベルが CFG_KNLMSKLVL である割込みハンドラと同じになります。この状態は、unl_cpu, iunl_cpu を呼び出すことによって元に戻ります。
(c)chg_ims によって割込みマスクレベルを 0 以外に変更している間の優先順位は、そのレベルの割込みハンドラと同じとなります。

1.3 "i"で始まるサービスコールの追加

表 2に示す"i"で始まる名称のサービスコールを追加します。追加サービスコールの仕様は、"i"がつく点を除いてベースとなるサービスコールと全く同じです。
なお、追加サービスコールは V1.0Ar1 以降でサポートされます。

表2 追加サービスコール

追加サービスコール	ベースサービスコール	機能	備考
ivsta_knl	vsta_knl	カーネルの起動	
ivsys_dwn	vsys_dwn	システムダウン	
ivini_cac	vini_cac	キャッシュの初期化	HI7700/4, HI7750/4 のみ
ivclr_cac	vclr_cac	キャッシュクリア	HI7700/4, HI7750/4 のみ
ivfls_cac	vfls_cac	キャッシュフラッシュ	HI7700/4, HI7750/4 のみ
ivinv_cac	vinv_cac	キャッシュの無効化	HI7700/4, HI7750/4 のみ

マニュアル関連箇所→「3.19.11 カーネルの起動」(p.178)、「3.19.12 システムダウン」(p.179)、「3.23 キャッシュサポート機能」(p.200)

1.4 ミューテックスの TA_CEILING 属性

HI7000/4 シリーズのミューテックスにおける TA_CEILING 属性(優先度上限プロトコル)では、簡略化した優先度制御規則を採用しています。簡略化した優先度制御規則では、タスクの優先度を高くする制御はすべて行われますが、タスクの優先度を低くする制御は、タスクがロックしていたミューテックスが無くなったとき(複数のミューテックスをロックしていた場合は、それら全てを解放したとき)にのみ行われます。

マニュアル関連箇所→「3.11 拡張同期・通信(ミューテックス)機能」(p.113)

1.5 システム時刻

システム時刻は SYSTIM 型の構造体によって符号無し 48bit 整数として表現されますが、その最大値は以下のようになります。

[CFG_TICNUME/CFG_TICDENO ≦ 1 の場合]
最大値＝H'7fffffffffff/CFG_TICDENO
[CFG_TICNUME/CFG_TICDENO ＞ 1 の場合]
最大値＝H'7fffffffffff

タイマ割込み(isig_tim)によってシステム時刻を更新する際に上記最大値を超える場合には、システム時刻は 0 に戻ります。
また、set_tim サービスコールで、上記最大値を超える値を指定した場合の動作は保証されません。

マニュアル関連箇所→「3.15 時間管理機能(システム時刻管理)」(p.144)

1.6 サンプルタイマドライバ

タイマ割込み周期時間(T)は、以下の式で表されます。

T＝ {(１／PCLOCK) ×DIV} ×N

ここで、それぞれの記号の意味は以下の通りです。

・PCLOCK：タイマモジュールへ供給されるクロック周波数
・DIV：タイマモジュールのレジスタ設定による分周比
・N：周期時間に相当するクロックカウント数

一般にタイマモジュールのカウンタレジスタに n を設定すると、n+1 回のカウントが行われます。したがって、

n = N - 1

となります。

HI7000/4 シリーズで提供するサンプルタイマドライバでは、上記の計算方法にしたがって n を算出し、タイマモジュールのレジスタに設定します。サンプルタイマドライバでは、上式中の記号を以下のように対応させています。

T : CFG_TICNUME/CFG_TICDENO [ミリ秒]
PCLOCK : ***nnnn***_tmrdef.h で定義している PCLOCK [Hz]
DIV : HI7000/4 では、***nnnn***_tmrdrv.c 中で定義しています。
HI7700/4, HI7750/4 では、PCLOCK に応じて自動選択するようになっています。
n : ***nnnn***_tmrdrv.c で、上記計算式を COUNTER に定義しています。

なお、COUNTER の計算結果がタイマカウンタレジスタのサイズを超える場合には、周期は期待した通りになりません。COUNTER の計算結果がレジスタのサイズを超えないかを必ず確認してください。
また、COUNTER の計算結果が整数とならない場合、整数に丸められますので、周期 T は期待した時間よりもわずかに短くなることになります。

マニュアル関連箇所→「4.10.2 サンプルタイマドライバ」 (p.224)

1.7 ダイレクト割込みハンドラの終了方法(HI7000/4)

アセンブリ言語でダイレクト割込みハンドラを記述する場合には、表 3の仕様にしたがってください。

表 3 ダイレクト割込みハンドラの終了方法

項目	仕様
終了命令	(1) 割込みレベルがカーネル割込みマスクレベル(CFG_KNLMSKLVL)以下の場合：TRAPA #25 (2) (1)以外の場合：RTE 命令
終了時の状態	終了命令を実行する時点で、PC,SR 以外の全ての CPU レジスタがハンドラ起動前と同じ

マニュアル関連箇所→「4.7.2 ダイレクト割込みハンドラ(HI7000/4)」(p.215)

1.8 予約トラップ

TRAPA #16～31 のトラップは HI7000/4 シリーズで予約されているので、アプリケーションでは使用できません。これらの命令を実行した場合、1.7に記載のケースを除いて未定義例外と扱われ、システムダウンとなります。

1.9 "TSZ_"で始まるマクロ

μITRON4.0 仕様では、いくつかのオブジェクトについては、アプリケーション側で用意した領域を使用することができる機能があります。オブジェクト生成のサービスコール(cre_dtq など)では、その領域のアドレスを指定します。また、μITRON4.0 仕様では、このような場合にその領域のサイズを求めるために、"TSZ_"で始まるマクロを規定しています。なお、これらの機能はμITRON4.0 仕様のスタンダードプロファイル外の機能です。
現在の HI7000/4 シリーズでは、アプリケーション側で用意した領域を使用する機能はサポートしていないため、"TSZ_"で始まるマクロは意味を持ちません。
参考のため、μITRON4.0 仕様で規定されているメモリ領域サイズの算出するためのマクロを表 2 に示します。

表 4 μITRON4.0 仕様で規定されているメモリ領域サイズの算出するためのマクロ

マクロ	対象オブジェクト	マクロの目的
TSZ_DTQ	データキュー	データキュー領域サイズの算出
TSZ_MPRIHD	メールボックス	メッセージヘッダ領域サイズの算出
TSZ_MBF	メッセージバッファ	メッセージバッファ領域サイズの算出
TSZ_MPF	固定長メモリプール	固定長メモリプール領域サイズの算出
TSZ_MPL	可変長メモリプール	可変長メモリプール領域サイズの算出

マニュアル関連箇所→表 4.1 (p.206)

1.10 コンフィギュレータのサービスコール選択ビュー

(1) サービスコール選択ビューでは、"i"で始まるサービスコールの選択項目はありません。"i"のないサービスコールを選択することで、自動的に対応する"i"で始まるサービスコールも選択された意味になります。
(2) 各オブジェクトを使用する場合は、必ずそのオブジェクトを生成・定義する cre_???, def_サービスコール(表 5参照)を選択してください。

表 5 オブジェクトを使用するために選択必要なサービスコール

オブジェクト	必要なサービスコール
セマフォ	cre_sem
イベントフラグ	cre_flg
データキュー	cre_dtq
メールボックス	cre_mbx
ミューテックス	cre_mtx
メッセージバッファ	cre_mbf

オブジェクト	必要なサービスコール
固定長メモリプール	cre_mpf
可変長メモリプール	cre_mpl
周期ハンドラ	cre_cyc
アラームハンドラ	cre_alm
拡張 SVC ハンドラ	def_svc

1.11 HI7700/4 のキャッシュサポートサービスコール

表 6に、HI7700/4 のキャッシュサポートサービスコールが対応しているキャッシュと、vini_cac で指定が必要なパラメータを示します。

表 6 HI7700/4 のキャッシュサポートサービスコール

キャッシュサイズ	サポートしているカーネル	条件	vini_cac パラメータ		
			ccr_data	entnum	waynum
8kB	V1.0 以降	内蔵 RAM モード未使用	デバイス仕様に合わせて指定してください。	4	128
		内蔵 RAM モード使用		2	128
16kB		－		4	256
32kB *	V1.1.00 以降	32kB モードを使用		4	512
		16kB モードを使用		4	256

* CCR3 レジスタの設定が必要な場合は、必ず vini_cac 発行前に CCR3 レジスタを設定してください。また、CCR3 レジスタの設定から vini_cac からリターンするまでの期間では、キャッシュをアクセスしないようにしてください。例えば、キャッシュディスエーブル状態で CCR3 レジスタの設定と vini_cac を行ってください。

2. 誤記訂正

2.1 p.21 図 2.10

図 2.10 全体を、以下に訂正します。

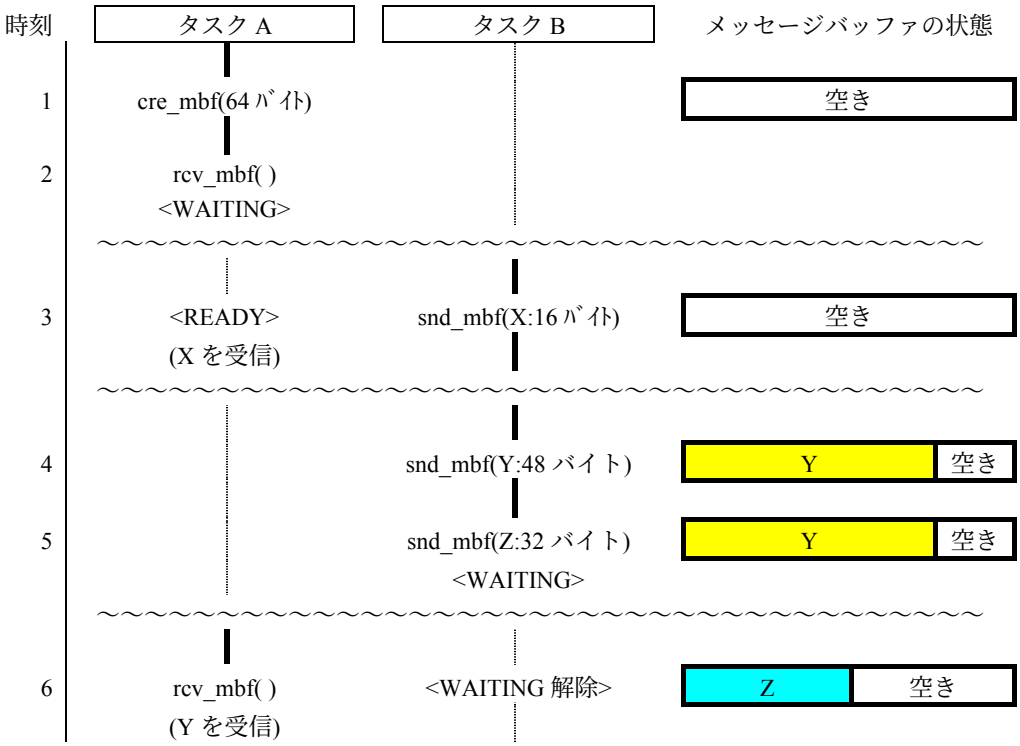

図2.1 メッセージバッファ動作例

図の解説

太実線は実際に実行したことを示しています。以下、各時刻での説明を補足します。

1. タスクAがcre_mbfでバッファサイズが64バイトで、扱えるメッセージの最大サイズが48バイトのメッセージバッファを生成します。
2. タスクAはメッセージを受信するために、48バイトのメモリを用意してrcv_mbfを発行します。この時点ではメッセージバッファにメッセージが無いので、タスクAはメッセージ受信待ち状態になります。
3. タスクBがsnd_mbfで16バイトのメッセージを送信すると、受信を待っていたタスクAの待ち状態が解除され、タスクAが用意したメモリにXがコピーされます。また、タスクAは受信したメッセージサイズ16をリターンパラメータとして受け取ります。
4. タスクBがsnd_mbfで48バイトのメッセージYを送信します。受信待ちタスクは存在しないので、メッセージYはメッセージバッファにコピーされます。なお、メッセージをメッセージバッファにコピーする際、カーネルはメッセージバッファ領域を4バイト消費しますが、図中ではこの表記はしていません。
5. タスクBがsnd_mbfで32バイトのメッセージZを送信しようとしますが、メッセージバッファの空きが足りないのでタスクBは送信待ち状態になります。
6. タスクAが48バイトのメモリを用意してrcv_mbfを発行すると、メッセージバッファに蓄えられていたメッセージYがタスクAが用意したメモリにコピーされます。また、タスクAは受信したメッセージサイズ48をリターンパラメータとして受け取ります。また、メッセージYをタスクAに渡したことによってメッセージバッファに空きが生じたので、タスクBの送信待ち状態は解除され、メッセージZがメッセージバッファにコピーされます。

2.2 p.42 図 3.2 のエラー種別

訂正前	訂正後
エラー種別は以下の意味を持ちます。 [k] 常にチェックされるエラー [p] パラメータチェック機能付きのカーネル(CFG_PARCHK をチェック) の場合のみ検出されるエラー	エラー種別は以下の意味を持ちます。 [k] 常にチェックされるエラー [p] パラメータチェック機能付きのカーネル(CFG_PARCHK をチェック) の場合のみ検出されるエラー ***CFG_PARCHK をチェックしない場合、[p]に分類されるエラーが発生する状況が生じた場合の動作は保証されません。***

2.3 p.46「3.4.1 タスクの生成」の E_PAR エラーの説明

訂正前	訂正後
E_PAR [p] パラメータエラー(pk_ctsk が 4 の倍数以外、task が奇数、stksz が 4 の倍数以外、stksz ≦0 、itskpri ≦0 、itskpri ＞CFG_MAXTSKPRI)	E_PAR [p] パラメータエラー(pk_ctsk が 4 の倍数以外、task が奇数、stksz が 4 の倍数以外、***stksz==0*** 、***stksz≧0x80000000***、itskpri≦0 、itskpri＞CFG_MAXTSKPRI)

2.4 p.51「3.4.5 タスクの起動(起動コード指定)」の E_ID エラーの説明

訂正前	訂正後
E_ID [p] 不正 ID 番号(tskid≦0、tskid＞CFG_MAXTSKID~~、非タスクコンテキストで tskid=TSK_SELF(0)を指定~~)	E_ID [p] 不正 ID 番号(tskid≦0、tskid＞CFG_MAXTSKID)

2.5 p.54 「3.4.8 タスク優先度の変更」の本文の最終段落

訂正前	訂正後
対象タスクが TA_CEILING 属性のミューテックスをロックしている場合で、tskpri に指定されたベース優先度が、それらのミューテックスのいずれかの上限優先度よりも高い場合には、E_ILUSE を返します。	対象タスクが TA_CEILING 属性のミューテックスを***ロックしているかロックを待っている場合で***、tskpri に指定されたベース優先度が、それらのミューテックスのいずれかの上限優先度よりも高い場合には、E_ILUSE を返します。

2.6 p.55「3.4.9 タスク優先度の参照」に E_PAR エラーを追記

追記内容
E_PAR [p] パラメータエラー(p_tskpri が 2 の倍数で無い)

2.7 p.66「3.5.4 待ち状態の強制解除」の E_ID エラーの説明

訂正前	訂正後
E_ID [p] 不正 ID 番号(tskid＜0、tskid＞CFG_MAXTSKID~~、非タスクコンテキストで tskid=TSK_SELF(0)を指定~~)	E_ID [p] 不正 ID 番号(***tskid≦0***、tskid＞CFG_MAXTSKID)

2.8 p.81「3.7.1 セマフォの生成」の E_PAR エラーの説明

訂正前	訂正後
E_PAR [p] パラメータエラー(pk_csem が 4 の倍数以外、maxsem≦0、maxsem＞H'ffff)	E_PAR [p] パラメータエラー(pk_csem が 4 の倍数以外、***maxsem==0***、maxsem＞H'ffff)

2.9 p.88「3.8.1 イベントフラグの生成」の T_CFLG 構造体

訂正前	訂正後
■パケットの構造 typedef struct t_cflg { ATR flgatr; +0 4 イベントフラグ属性 UINT iflgptn; +4 4 イベントフラグの初期値 } T_CFLG;	■パケットの構造 typedef struct t_cflg { ATR flgatr; +0 4 イベントフラグ属性 ***FLGPTN*** iflgptn; +4 4 イベントフラグの初期値 } T_CFLG;

2.10 p.96 表 3.16 の項番 9 の訂正(prcv_dtq を呼び出し可能なシステム状態)

prcv_dtq は、ディスパッチ禁止状態からも発行可能なので、表 3.16 の項番 9 の"D"の欄に"○"を追加します。以下に変更後の表 3.16 の抜粋を示します。

項番	サービスコール	機 能	呼び出し可能なシステム状態						
			T	N	E	D	U	L	C
...									
9	prcv_dtq [S]	同上(ポーリング)	○		○	***○***	○		
...									

2.11 p.100「3.9.3 データキューへの送信」のタイトル訂正

訂正前	3.9.3 データキューへの送信(snd_dtq,psnd_dtq,ipsnd_dtq,tsnd_dtq)
訂正後	3.9.3 データキューへの送信(snd_dtq,psnd_dtq,ipsnd_dtq,tsnd_dtq***,fsnd_dtq,ifsnd_dtq***)

2.12 p.100「3.9.3 データキューへの送信」の E_ILUSE エラーの追加

追記内容
E_ILUSE [k] サービスコール不正使用(dtqcnt が 0 のデータキューに対する fsnd_dtq,ifsnd_dtq の発行)

2.13 p.108「3.10.3 メールボックスへの送信」の E_PAR エラーの説明

訂正前	訂正後
E_PAR [p] パラメータエラー(pk_msg が 4 の倍数以外、メッセージ先頭 4 バイトが 0 以外) [k] (msgpri＞CFG_MAXMSGPRI)	E_PAR [p] パラメータエラー(pk_msg が 4 の倍数以外、メッセージ先頭 4 バイトが 0 以外) [k] (***msgpri≦0***、msgpri＞CFG_MAXMSGPRI)

2.14 p.115「3.11.1 ミューテックスの生成」本文

訂正前	訂正後
ceilpri には、生成するミューテックスの上限優先度を指定します。指定できる値の範囲は、0～CFG_MAXTSKPRI です。	ceilpri には、生成するミューテックスの上限優先度を指定します。指定できる値の範囲は、***1～CFG_MAXTSKPRI*** です。

2.15 p.121「3.12.1 メッセージバッファの生成」の E_PAR エラーの説明

訂正前	訂正後
E_PAR [p] パラメータエラー(pk_cmbf が 4 の倍数以外、mbfsz が 4 の倍数以外、maxmsz ≦ 0)	E_PAR [p] パラメータエラー(pk_cmbf が 4 の倍数以外、mbfsz が 4 の倍数以外、***maxmsz == 0、maxmsz≧H'80000000***)

2.16 p.124「3.12.3 メッセージバッファへの送信」の本文追記

追記内容
ipsnd_mbf は、非タスクコンテキストからも発行可能です。対象のメッセージバッファに TA_TPRI 属性が指定されている場合は、非タスクコンテキストはタスクよりも優先順位が高いため、バッファに空きがあれば、先に送信を待っているタスクが存在しても、バッファにメッセージがコピーされます。

2.17 p.131「3.13.1 固定長メモリプールの生成」の E_PAR エラーの説明

訂正前	訂正後
E_PAR [p] パラメータエラー(pk_cmpf が 4 の倍数以外、blkcnt=0、blksz が 4 の倍数以外、blksz=0) [k] (blkcnt×blksz が 32 ビット範囲を超えている)	E_PAR [p] パラメータエラー(pk_cmpf が 4 の倍数以外、blkcnt=0、blksz が 4 の倍数以外、blksz=0) [k] (***blkcnt×(blksz+4) が 32 ビット範囲を超えている***)

2.18 p.137「3.14.1 可変長メモリプールの生成」の E_PAR エラーの説明

訂正前	訂正後
E_PAR [p] パラメータエラー(pk_cmpl が 4 の倍数以外、mplsz が 4 の倍数以外、mplsz＜20)	E_PAR [p] パラメータエラー(pk_cmpl が 4 の倍数以外、mplsz が 4 の倍数以外、mplsz＜20、***mplsz≧H'80000000***)

2.19 p.164「3.18.2 オーバーランハンドラの動作開始」の E_ID, E_NOEXS エラーの説明

訂正前	訂正後
E_PAR [p] 不正 ID 番号(tskid≦0、tskid＞CFG_MAXTSKID、非タスクコンテキストで tskid=TSK_SELF(0)を指定) E_NOEXS [k] 未登録(ovrid のオーバーランハンドラが存在しない)	E_PAR [p] 不正 ID 番号(***tskid＜0***、tskid＞CFG_MAXTSKID、非タスクコンテキストで tskid=TSK_SELF(0)を指定) E_NOEXS [k] 未登録(***tskid のタスクが存在しない***)

2.20 p.165「3.18.3 オーバーランハンドラの動作停止」の E_ID エラーの説明

訂正前	訂正後
E_PAR [p] 不正 ID 番号(tskid≦0、tskid＞CFG_MAXTSKID、非タスクコンテキストで tskid=TSK_SELF(0)を指定)	E_PAR [p] 不正 ID 番号(***tskid＜0***、tskid＞CFG_MAXTSKID、非タスクコンテキストで tskid=TSK_SELF(0)を指定)

2.21 p.166「3.18.4 オーバーランハンドラの状態参照」の E_ID,エラーの説明

訂正前	訂正後
E_PAR [p] 不正 ID 番号(tskid≦0、tskid＞CFG_MAXTSKID)	E_PAR [p] 不正 ID 番号(***tskid＜0***、tskid＞CFG_MAXTSKID、***非タスクコンテキストで tskid=TSK_SELF(0)を指定***)

2.22 p.166「3.18.4 オーバーランハンドラの状態参照」の本文

追記内容
tskid=TSK_SELF（0）の指定により、自タスクの指定になります。

2.23 p.170「3.19.3 CPU ロック状態への移行」本文追記

訂正前	訂正後
割込みハンドラ内で iloc_cpu を呼び出し、CPU ロック状態に移行した場合は、割込みハンドラからリターンする前に必ず iunl_cpu を呼び出し、CPU ロック状態を解除してください。	***CPU ロック状態と CPU ロック解除状態の間の遷移は、loc_cpu,iloc_cpu, unl_cpu,iunl_cpu,ext_tsk,exd_tsk サービスコールによってのみ発生します。カーネル割込みマスクレベル(CFG_KNLMSKLVL)以下の割込みハンドラ、タイムイベントハンドラ、初期化ルーチン、タスク例外処理ルーチンの終了時には、必ず CPU ロック解除状態でなければなりません。CPU ロック状態の場合、動作は保証されません。なお、これらのハンドラ開始時は、常に CPU ロック解除状態です。*** ***CPU 例外ハンドラで CPU ロック/ロック解除状態を変更した場合、必ず終了前に元に状態に戻さなくてはなりません。戻さない場合、動作は保証されません。***

2.24 p.172「3.19.5 ディスパッチの禁止」の本文追記

追記内容
ディスパッチ禁止状態とディスパッチ許可解除状態の間の遷移は、dis_dsp, ena_dsp, ext_tsk, exd_tsk サービスコールによってのみ発生します。 CPU 例外ハンドラでディスパッチ禁止/許可状態を変更した場合、必ず終了前に元に状態に戻さなくてはなりません。戻さない場合、動作は保証されません。

2.25 cal_svc, ical_svc サービスコールのリターン値の型

2.25.1 p.190 「サービスコールの呼び出し」

訂正前	訂正後
ER ercd R0 サービスコールからのリターン値	***ER_UINT*** ercd R0 サービスコールからのリターン値

2.25.2 p.211 図 4.3

[訂正前]

```
ER Svcrtn(VP_INT par1, VP_INT par2)
{
    /* 拡張ｻｰﾋﾞｽｺｰﾙﾙｰﾁﾝの処理 */
    return E_OK;
}
```

←拡張サービスコールルーチンには、cal_svc で指定したパラメータが渡されます。cal_svc で指定するパラメータと数を合わせてください。
←発行元にリターン値を返します。

[訂正後]

```
ER_UINT Svcrtn(VP_INT par1, VP_INT par2)
{
    /* 拡張ｻｰﾋﾞｽｺｰﾙﾙｰﾁﾝの処理 */
    return E_OK;
}
```

←拡張サービスコールルーチンには、cal_svc で指定したパラメータが渡されます。cal_svc で指定するパラメータと数を合わせてください。
←発行元にリターン値を返します。

2.26 p.208 「4.3 予約名」

節のタイトルを「4.3 システム予約」に訂正します。
また、既存の内容を「4.3.1 予約名」とし、新たに「4.3.2 予約トラップ」として以下の内容を追加します。

追記内容
TRAPA #16～31 のトラップは HI7000/4 シリーズで予約されているので、アプリケーションでは使用できません。これらの命令を実行した場合、HI7000/4 におけるダイレクト割込みハンドラ終了時の TRAPA #25 を除き、未定義例外と扱われてシステムダウンとなります。

2.27 図 4.7 (p.217)

[訂正前]

```
    .INCLUDE "itron.inc"
            ...(省略)...
    TRAPA #D'25                    ←カーネル割込みマスクレベル以下の割込みハンドラ
                                     は TRAPA #25 で終了
```

[訂正後]

```
    .INCLUDE "itron.inc"
            ...(省略)...
    TRAPA #D'25                    ←カーネル割込みマスクレベル以下の割込みハンドラ
                                     は TRAPA #25 で終了(タスクコンテキストから TRAPA #25
                                     を実行した場合は、未定義例外としてシステムダウンとな
                                     ります)
```

2.28 図 5.1 (p.229), 図 5.2 (p.230), 図 5.3 (p.233)

図中のファイル名を、以下のように訂正します。

訂正前	訂正後
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 vini_cac サービスコールの C 言語 API (p.259)

[訂正前]

149	vini_cac	[HI7700/4] void vini_cac(UW ccr_data);	キャシュの初期化
		[HI7750/4] void vini_cac(UW ccr_data, UW entnum, UW waynum);	

[訂正後]

149	vini_cac	[HI7700/4] void vini_cac***(UW ccr_data, UW entnum, UW waynum)***;	キャシュの初期化
		[HI7750/4] void vini_cac***(UW ccr_data)***;	

以上