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

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

Application Note

RENESAS

μITRON仕様OS移行マニュアル

HI7000シリーズからHI7000/4シリーズへの移行ガイド

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　2005.06

安全設計に関するお願い

1. 弊社は品質、信頼性の向上に努めておりますが、半導体製品は故障が発生したり、誤動作する場合があります。弊社の半導体製品の故障又は誤動作によって結果として、人身事故、火災事故、社会的損害などを生じさせないような安全性を考慮した冗長設計、延焼対策設計、誤動作防止設計などの安全設計に十分ご留意ください。

本資料ご利用に際しての留意事項

1. 本資料は、お客様が用途に応じた適切なルネサス テクノロジ製品をご購入いただくための参考資料であり、本資料中に記載の技術情報についてルネサス テクノロジが所有する知的財産権その他の権利の実施、使用を許諾するものではありません。
2. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例の使用に起因する損害、第三者所有の権利に対する侵害に関し、ルネサス テクノロジは責任を負いません。
3. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他全ての情報は本資料発行時点のものであり、ルネサス テクノロジは、予告なしに、本資料に記載した製品または仕様を変更することがあります。ルネサス テクノロジ半導体製品のご購入に当たりましては、事前にルネサス テクノロジ、ルネサス販売または特約店へ最新の情報をご確認頂きますとともに、ルネサス テクノロジホームページ (http://www.renesas.com)などを通じて公開される情報に常にご注意ください。
4. 本資料に記載した情報は、正確を期すため、慎重に制作したものですが万一本資料の記述誤りに起因する損害がお客様に生じた場合には、ルネサス テクノロジはその責任を負いません。
5. 本資料に記載の製品データ、図、表に示す技術的な内容、プログラム及びアルゴリズムを流用する場合は、技術内容、プログラム、アルゴリズム単位で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断してください。ルネサス テクノロジは、適用可否に対する責任は負いません。
6. 本資料に記載された製品は、人命にかかわるような状況の下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。本資料に記載の製品を運輸、移動体用、医療用、航空宇宙用、原子力制御用、海底中継用機器あるいはシステムなど、特殊用途へのご利用をご検討の際には、ルネサス テクノロジ、ルネサス販売または特約店へご照会ください。
7. 本資料の転載、複製については、文書によるルネサス テクノロジの事前の承諾が必要です。
8. 本資料に関し詳細についてのお問い合わせ、その他お気付きの点がございましたらルネサス テクノロジ、ルネサス販売または特約店までご照会ください。

【商標等について】

- TRON は The Realtime Operating System Nucleus の略称です。ITRON は Industrial TRON の略称です。μITRON は、Micro Industrial TRON の略称です。
- TRON、ITRON および μITRON は、特定の商品ないしは商品群を指す名称ではありません。
- μITRON4.0 仕様は、トロン協会 ITRON 部会が中心となって策定されたオープンなリアルタイムカーネル仕様です。μITRON4.0 仕様の仕様書は、ITRON プロジェクトホームページ (http://www.assoc.tron.org/itron/home-j.html)から入手が可能です。
- Microsoft® Windows® 95 operating system, Microsoft® Windows® 98 operating system, Microsoft® Windows® Millennium Edition(Windows® Me) operating system, Microsoft® Windows NT® operating system, Microsoft® Windows® 2000 operating system, Microsoft® Windows® XP operating system は、米国 Microsoft Corporation の米国およびその他の国における登録商標です。
- SuperH™ は、(株)ルネサス テクノロジの商標です。
- その他、本書で登場するシステム名、製品名は各社の登録商標または商標です。

目次

図目次

表目次

1. はじめに

本資料は、μITRON3.0 仕様準拠の HI7000 シリーズから、μITRON4.0 仕様準拠の HI7000/4 シリーズへの移行の方法について解説します。

本資料は、HI7000 シリーズから HI7000/4 シリーズへの移行を保証するものではありません。

詳細は、必ず双方のマニュアルで確認してください。

表 1-1　CPU と製品の対応表

CPU ＼ 製品	HI7000 シリーズ	HI7000/4 シリーズ
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	―	

2. 概要

ここでは HI7000 シリーズから HI7000/4 シリーズで削除、追加、変更となった機能の概要について説明します。
各機能の詳細についてはそれぞれのマニュアルをご覧ください。

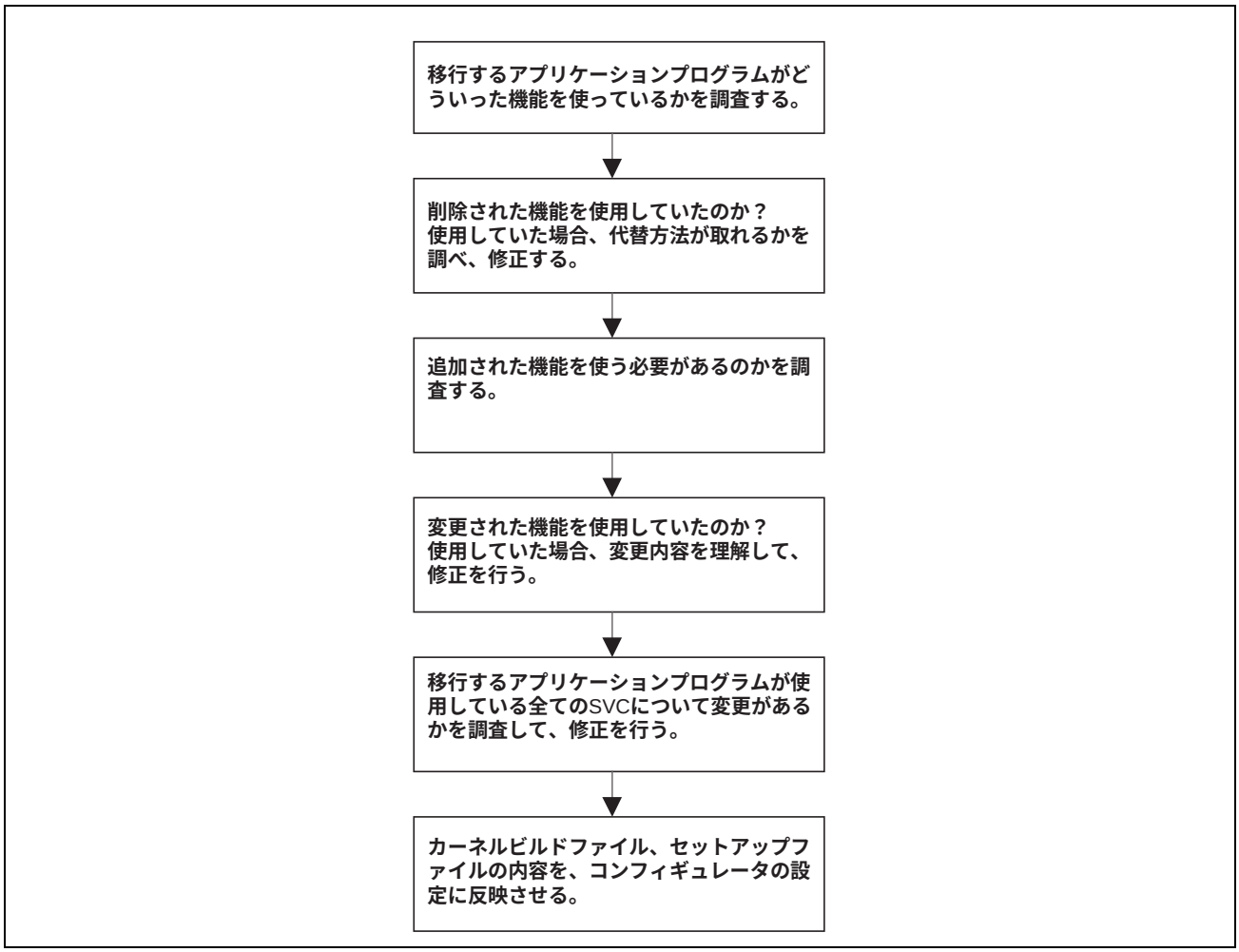

図 2-1　**移行の手順**

HI7000 シリーズと HI7000/4 シリーズの比較概要を**表** 2-1 に示します。

表 2-1　HI7000 **シリーズと** HI7000/4 **シリーズの比較概要**

番号	HI7000 シリーズの機能	HI7000/4 シリーズでの変更点	本資料での番号
1	**タイムスライス機能**	**代替えあり：オーバーランハンドラ機能**	3.1.1
2	**ページプール機能**	**代替え無し**	3.1.2
3	I/O **サポート機能**	**代替え無し**	3.1.3
4	**データキュー機能**	**新機能**	3.2.1
5	**ミューテックス機能**	**新機能**	3.2.2
6	**タスク例外機能**	**新機能**	3.2.3
7	**オーバーランハンドラ機能**	**新機能**	3.2.4
8	**タスクの多重起動要求**	**新機能**	3.2.5
9	“i”**のつくサービスコール**	**新機能**	3.2.6
10	**ディスパッチと** CPU **ロック**	**意味変更**	3.3.1
11	**拡張サービスコールルーチン**	**位置付け変更**	3.3.2
12	**イベントフラグ**	**属性が追加、クリア指定が変更**	3.3.3，3.3.4
13	**メールボックス**	**属性が追加、メッセージヘッダ型追加**	3.3.5，3.3.6
14	**メッセージバッファ**	**待ち行列が変更**	3.3.7
15	wup_tsk, sus_tsk	**自タスク指定可能**	3.3.8
16	**サンプルタイマドライバ**	**全般的に変更**	3.3.9
17	**時間管理**	**時間単位、時間管理が変更**	3.3.10
18	**周期ハンドラ**	**全般的に変更**	3.3.11
19	**アラームハンドラ**	**全般的に変更**	3.3.12
20	TA_COP1、TA_COP2 **属性**	**意味変更**	3.3.13
21	**オブジェクト名称機能**	**一部の機能の代替えあり**	3.3.14
22	**システムダウンルーチン**	**ルーチン名、システムダウン情報の変更**	3.3.15

3. 追加/削除/変更機能

ここでは HI7000/4 シリーズで削除、追加、変更となった機能について説明します。
各機能の詳細についてはそれぞれのマニュアルをご覧下さい。

3.1 削除機能

3.1.1 タイムスライス機能

HI7000 シリーズのタスク生成時でのタイムスライス時間設定が削除されました。HI7000/4 シリーズでタイムスライス機能を実現する場合には、オーバーランハンドラ機能を使用します。
表 3-1、**図** 3-1 に代替えの詳細を示します。

表 3-1　**タイムスライス機能の代替**

番号	タイムスライス機能	HI7000 シリーズ	HI7000/4 シリーズ
1	タイムスライス時間	タスク生成時に設定 (タイムスライス時間)	sta_ovr, ista_ovr で設定 (タスクの上限プロセッサ時間)
2	レディキュー回転	自動的に rot_rdq 実行	オーバーランハンドラ内でユーザが rot_rdq を実行。 起動の原因となったタスクの上限プロセッサ時間は解除されるので再設定する。

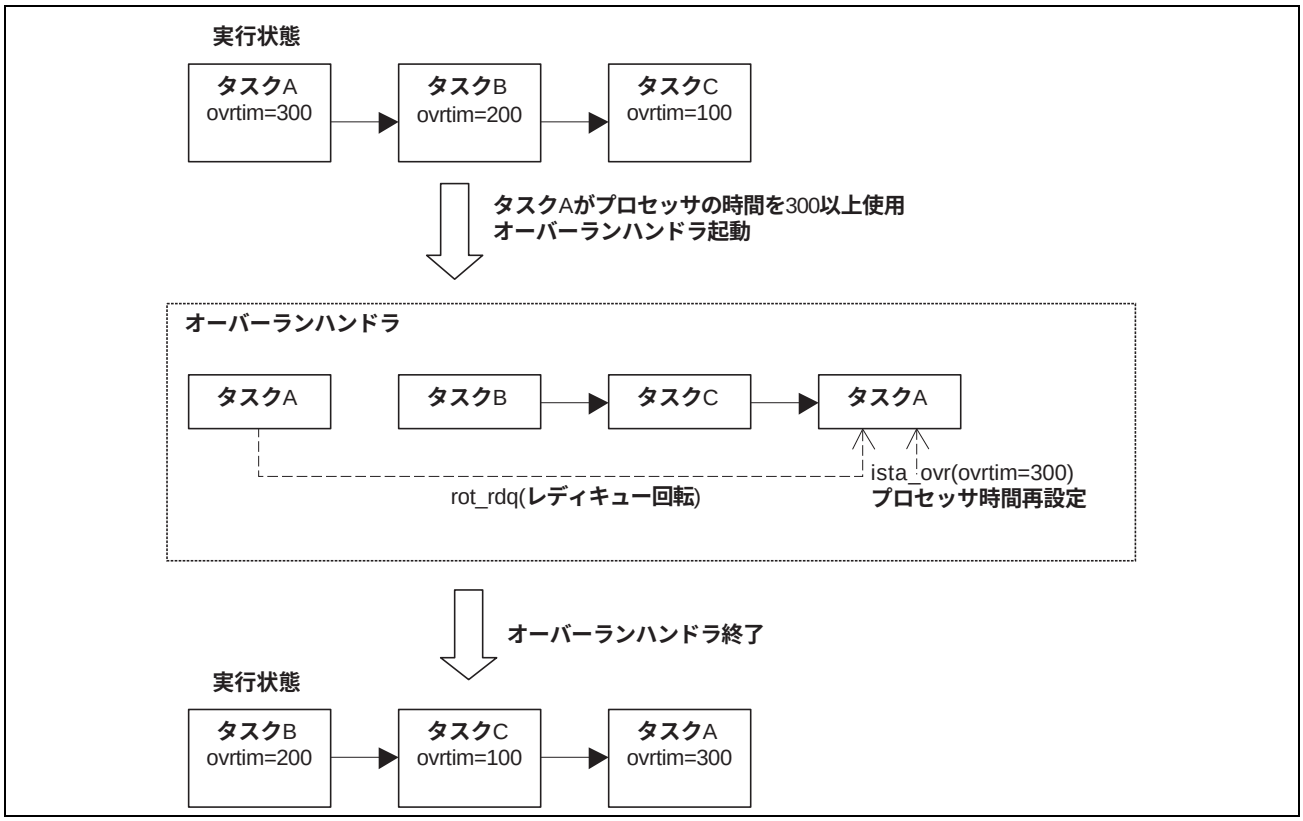

図 3-1　**オーバーランハンドラ機能によるタイムスライス機能実装例**

3.1.2 ページプール機能

HI7000/4 シリーズでの代替え機能はありません。
使用しているユーザは固定長メモリプール機能などで実現できないかを検討してください。

3.1.3 I/O サポート機能

HI7000/4 シリーズでの代替え機能はありません。
使用しているユーザは拡張 SVC で実現できないかを検討してください。

3.1.4 その他の機能（vrst_tsk, vjmp_msg, vasn_svc, ref_sys）

表 3-2 に HI7000 シリーズから削除された機能に対する、HI7000/4 シリーズでの代替え手段を示します。

表 3-2　HI7000/4 **シリーズとの対応表**

番号	SVC	代替え手段
1	vrst_tsk	**代替えはありません。いったん終了して再起動で代替できないか検討してください。**
2	vjmp_msg	**メッセージ優先度を最高にして送信／データキューを使用して** fsnd_dtq **による送信により代替えできます。**
3	vasn_svc	def_svc **で代用可能です。注意事項を以下に示します。** **未定義の拡張機能コードを検索し拡張** SVC **を割り当てるという動作はできませんが、拡張** SVC **の割り当てだけなら** def_svc **で実行できます。**
4	ref_sys	**代替えはありません。**sns_ctx,sns_loc,sns_dsp,sns_dpn **などで代替えできないか検討してください。**

3.2 追加機能

3.2.1 データキュー機能

HI7000/4 シリーズでは、新しくデータキュー機能が追加されました。
データキュー機能は、1 ワードのメッセージ（データ）受け渡しによる同期・通信ができます。
表 3-3 にデータキュー機能の概要を示します。

表 3-3 **データキュー機能**

番号	機能	SVC	API
1	**データキューの生成**	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	**データキューの生成** (ID **番号自動割り付け**)	acre_dtq	ER_ID dtqid = acre_dtq(T_CDTQ *pk_cdtq);
		iacre_dtq	ER_ID dtqid = iacre_dtq(T_CDTQ *pk_cdtq);
3	**データキューの削除**	del_dtq	ER ercd = del_dtq(ID dtqid);
4	**データキューへの送信**	snd_dtq	ER ercd = snd_dtq(ID dtqid, VP_INT data);
5	**同上(ポーリング)**	psnd_dtq	ER ercd = psnd_dtq(ID dtqid, VP_INT data);
		ipsnd_dtq	ER ercd = ipsnd_dtq(ID dtqid, VP_INT data);
6	**同上(タイムアウト有)**	tsnd_dtq	ER ercd = tsnd_dtq(ID dtqid, VP_INT data, TMO tmout);
7	**データキューへの強制送信**	fsnd_dtq	ER ercd = fsnd_dtq(ID dtqid, VP_INT data);
		ifsnd_dtq	ER ercd = ifsnd_dtq(ID dtqid, VP_INT data);
8	**データキューからの受信**	rcv_dtq	ER ercd = rcv_dtq(ID dtqid, VP_INT *p_data);
9	**同上(ポーリング)**	prcv_dtq	ER ercd = prcv_dtq(ID dtqid, VP_INT *p_data);
10	**同上(タイムアウト有)**	trcv_dtq	ER ercd = trcv_dtq(ID dtqid, VP_INT *p_data, TMO tmout);
11	**データキューの状態参照**	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 ミューテックス機能

HI7000/4 シリーズでは、新しくミューテックス機能が追加されました。
ミューテックス機能は、共有資源を使用する際にタスク間で排他制御を行うことができます。
表 3-4 にミューテックス機能の概要を示します。

表 3-4 **ミューテックス機能**

番号	機能	SVC	API
1	**ミューテックスの生成**	cre_mtx	ER ercd = cre_mtx(ID mtxid, T_CMTX *pk_cmtx);
2	**ミューテックスの生成** (ID **番号自動割り付け**)	acre_mtx	ER_ID mtxid = acre_mtx(T_CMTX *pk_cmtx);
3	**ミューテックスの削除**	del_mtx	ER ercd = del_mtx(ID mtxid);
4	**ミューテックスのロック**	loc_mtx	ER ercd = loc_mtx(ID mtxid);
5	**同上(ポーリング)**	ploc_mtx	ER ercd = ploc_mtx(ID mtxid);
6	**同上(タイムアウト有)**	tloc_mtx	ER ercd = tloc_mtx(ID mtxid, TMO tmout);
7	**ミューテックスのロック解除**	unl_mtx	ER ercd = unl_mtx(ID mtxid);
8	**ミューテックスの状態参照**	ref_mtx	ER ercd = ref_mtx(ID mtxid, T_RMTX *pk_rmtx);

3.2.3 タスク例外機能

HI7000/4 シリーズでは、新しくタスク例外機能が追加されました。
タスク例外機能は、タスクに発生した例外を処理することができます。
表 3-5 にタスク例外機能の概要を示します。

表 3-5 **タスク例外機能**

番号	機能	SVC	API
1	**タスク例外処理ルーチンの定義**	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	**タスク例外処理の要求**	ras_tex	ER ercd = ras_tex(ID tskid, TEXPTN rasptn);
		iras_tex	ER ercd = iras_tex(ID tskid, TEXPTN rasptn);
3	**タスク例外処理の禁止**	dis_tex	ER ercd = dis_tex();
4	**タスク例外処理の許可**	ena_tex	ER ercd = ena_tex();
5	**タスク例外禁止状態の参照**	sns_tex	BOOL state =sns_tex();
6	**タスク例外処理の状態参照**	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 オーバーランハンドラ機能

HI7000/4 シリーズでは、新しくオーバーラン機能が追加されました。
オーバーランハンドラ機能は､タスクが設定された時間を超えてプロセッサを使用した際に処理を行うことができます。
表 3-6 にオーバーランハンドラ機能の概要を示します。

表 3-6 **オーバーランハンドラ機能**

番号	機能	SVC	API
1	**オーバーランハンドラの定義**	def_ovr	ER ercd = def_ovr(T_DOVR *pk_dovr);
2	**オーバーランハンドラの動作開始**	sta_ovr	ER ercd = sta_ovr(ID tskid, OVRTIM ovrtim);
		ista_ovr	ER ercd = ista_ovr(ID tskid, OVRTIM ovrtim);
3	**オーバーランハンドラの動作停止**	stp_ovr	ER ercd = stp_ovr(ID tskid);
		istp_ovr	ER ercd = istp_ovr(ID tskid);
4	**オーバーランハンドラの状態参照**	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 タスクの多重起動要求機能

HI7000/4 シリーズでは、新たにタスクの多重起動要求が追加されました。
多重起動とは、すでに起動されているタスクに対して act_tsk を発行すると、そのタスクを起動しようとしたという記録が残り、後でそのタスクが終了したときに自動的に再起動する機能です。
この機能は、act_tsk サービスコールにて使用可能です。

3.2.6 先頭に"i"がつくサービスコール

HI7000/4 シリーズでは、先頭に"i"がつくサービスコールが追加されました。それらは非タスクコンテキスト専用のサービスコールです。非タスクコンテキスト内で呼び出すサービスコールは、非タスクコンテキスト専用のサービスコールを使用するようにしてください。**表 3-7** に一覧を示します。

表 3-7 非タスクコンテキスト専用サービスコール一覧

番号	サービスコール	機能	番号	サービスコール	機能
1	icre_tsk	タスク生成(ダイナミックスタック使用)	2	ivscr_tsk	タスク生成(スタティックスタック使用)
3	iacre_tsk	タスク生成(ID 番号自動割り付け)	4	iact_tsk	タスクの起動
5	ista_tsk	タスク起動(起動コード指定)	6	ican_act	タスク起動要求のキャンセル
7	ichg_pri	タスク優先度変更	8	iget_pri	タスク優先度の参照
9	iref_tsk	タスクの状態参照	10	iref_tst	タスクの状態参照(簡易版)
11	iwup_tsk	タスクの起床	12	ican_wup	タスク起床要求のキャンセル
13	irel_wai	待ち状態の強制解除	14	isus_tsk	強制待ち状態への移行
15	irsm_tsk	強制待ち状態からの再開	16	ifrsm_tsk	強制待ち状態からの強制再開
17	ivset_ftl	タスク付属イベントフラグのセット	18	ivclr_tfl	タスク付属イベントフラグのクリア
19	idef_tex	タスク例外処理ルーチンの定義	20	iras_tex	タスク例外処理の要求
21	iref_tex	タスク例外処理の状態参照	22	icre_sem	セマフォの生成
23	iacre_sem	セマフォの生成(ID 番号自動割り付け)	24	isig_sem	セマフォ資源の返却
25	ipol_sem	セマフォ資源の獲得(ポーリング)	26	iref_sem	セマフォの状態参照
27	icre_flg	イベントフラグの生成	28	iacre_flg	イベントフラグの生成 (ID 番号自動割り付け)
29	iset_flg	イベントフラグのセット	30	iclr_flg	イベントフラグのクリア
31	ipol_flg	イベントフラグ待ち(ポーリング)	32	iref_flg	イベントフラグの状態参照
33	icre_dtq	データキューの生成	34	iacre_dtq	データキューの生成(ID 番号自動割り付け)
35	ipsnd_dtq	データキューへの送信(ポーリング)	36	ifsnd_dtq	データキューへの強制送信
37	iref_dtq	データキューの状態参照	38	icre_mbx	メールボックスの生成
39	iacre_mbx	メールボックスの生成 (ID 番号自動割り付け)	40	isnd_mbx	メールボックスへの送信
41	iprcv_mbx	メールボックスからの受信(ポーリング)	42	iref_mbx	メールボックスの状態参照
43	icre_mbf	メッセージバッファの生成	44	iacre_mbf	メッセージバッファの生成 (ID 番号自動割り付け)
45	ipsnd_mbf	メッセージバッファへの送信 (ポーリング)	46	iref_mbf	メッセージバッファの状態参照
47	icre_mpf	固定長メモリプールの生成	48	iacre_mpf	固定長メモリプールの生成 (ID 番号自動割り付け)
49	ipget_mpf	固定長メモリブロックの獲得 (ポーリング)	50	irel_mpf	固定長メモリブロックの返却
51	iref_mpf	固定長メモリプールの状態参照	52	icre_mpl	可変長メモリプールの生成
53	iacre_mpl	可変長メモリプールの生成 (ID 番号自動割り付け)	54	ipget_mpl	可変長メモリブロックの獲得 (ポーリング)
55	irel_mpl	可変長メモリブロックの返却	56	iref_mpl	可変長メモリプールの状態参照
57	iset_tim	システム時刻の設定	58	iget_tim	システム時刻の参照
59	icre_cyc	周期ハンドラの生成	60	iacre_cyc	周期ハンドラの生成 (ID 番号自動割り付け)
61	ista_cyc	周期ハンドラの動作開始	62	istp_cyc	周期ハンドラの動作停止
63	iref_cyc	周期ハンドラの状態参照	64	icre_alm	アラームハンドラの生成
65	iacre_alm	アラームハンドラの生成 (ID 番号自動割り付け)	66	ista_alm	アラームハンドラの動作開始
67	istp_alm	アラームハンドラの動作停止	68	iref_alm	アラームハンドラの状態参照
69	ista_ovr	オーバーランハンドラの動作開始	70	istp_ovr	オーバーランハンドラの動作停止
71	iref_ovr	オーバーランハンドラの状態参照	72	irot_rdq	タスクの優先順位の回転

番号	サービスコール	機能	番号	サービスコール	機能
73	iget_tid	実行状態のタスク ID の参照	74	iloc_cpu	CPU ロック状態への移行
75	iunl_cpu	CPU ロック状態の解除	76	ivsta_knl	カーネルの起動(*1)
77	ivsys_dwn	システムダウン	78	ivget_trc	トレースの取得
79	idef_inh	割り込みハンドラの定義	80	ichg_ims	割り込みマスクの変更
81	iget_ims	割り込みマスクの参照	82	idef_svc	拡張サービスコールの定義
83	ical_svc	サービスコールの呼び出し	84	idef_exc	CPU 例外ハンドラの定義
85	ivdef_trp	CPU 例外ハンドラの定義 (TRAPA 命令例外)	86	iref_cfg	コンフィギュレーション情報の参照
87	iref_ver	バージョン情報の参照	88	ivini_cac	キャッシュの初期化
89	ivclr_cac	キャッシュのクリア	90	ivfls_cac	キャッシュのフラッシュ
91	ivinv_cac	キャッシュの無効化			

3.3 変更機能

3.3.1 ディスパッチ禁止と CPU ロックの意味変更

HI7000 シリーズでは、CPU ロック状態はディスパッチと割り込みが禁止された状態と扱っていたのを、HI7000/4 シリーズではそれぞれ独立した状態として扱います。

例えば、HI7000/4 シリーズでは、ディスパッチ禁止状態から CPU ロック状態になった場合、CPU ロックを解除してもディスパッチ禁止状態のままです。逆にディスパッチ許可状態から CPU ロック状態になった場合は、CPU ロックを解除するとディスパッチ許可状態になります。

移行の際は、unl_cpu、loc_cpu、dis_dsp、ena_dsp 使用後のディスパッチ状態、CPU ロック状態に注意して作業を行ってください。

表 3-8 HI7000 シリーズでの dis_dsp、loc_cpu システムコールによる状態遷移

状態番号	システム状態	現在の状態		発行するシステムコールと遷移先状態番号			
		割り込み	ディスパッチ	dis_dsp	ena_dsp	loc_cpu	unl_cpu
1	タスク実行状態	許可	許可	→2	→1	→3	→1
2	ディスパッチ禁止状態	許可	禁止	→2	→1	→3	→1
3	CPU ロック状態	禁止	禁止	E_CTX	E_CTX	→3	→1

表 3-9 HI7000/4 シリーズでの dis_dsp、loc_cpu サービスコールによる状態遷移

状態番号	システム状態	現在の状態		発行するシステムコールと遷移先状態番号			
		CPU ロック	ディスパッチ	dis_dsp	ena_dsp	loc_cpu	unl_cpu
1	ディスパッチ許可状態	—	許可	→2	→1	→4	→3
2	ディスパッチ禁止状態	—	禁止	→2	→1	→4	→3
3	CPU ロック解除状態	解除	—	→2	→1	→4	→3
4	CPU ロック状態	ロック	—	E_CTX	E_CTX	→4	→3

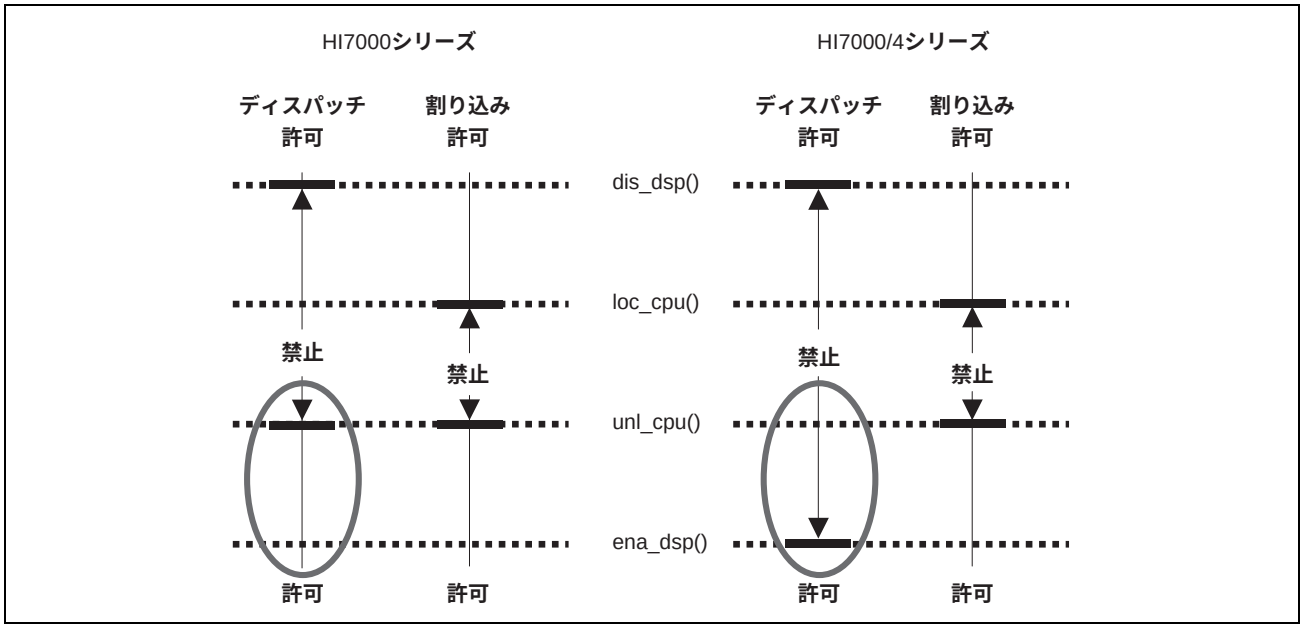

図 3-2 ディスパッチ禁止と CPU ロックの変更

3.3.2 拡張サービスコールルーチンの位置付け変更

HI7000/4 シリーズでは、拡張サービスコールルーチン（拡張 SVC）の位置付けが**表 3-10** のように変更されています。

HI7000 シリーズではディスパッチは起きませんでしたが、HI7000/4 シリーズではタスクコンテキストで発行した場合ディスパッチが起きるので、ディスパッチが起こらない前提で作成した拡張 SVC は修正が必要になります。

この場合は、dis_dsp()、ena_dsp()追加で HI7000 シリーズと同じ動きになります。

表 3-10 拡張サービスコールの実行コンテキスト

呼び出し元	HI7000 シリーズ	HI7000/4 シリーズ
タスクコンテキスト／タスク部	タスク独立部 (ディスパッチは起こらない)	タスクコンテキスト(ディスパッチが起こる)
非タスクコンテキスト／タスク独立部		非タスクコンテキスト(ディスパッチは起こらない)

```
ER_UINT Svcrtn(VP_INT par1, VP_INT par2)     ←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。
    /* 拡張ｻｰﾋﾞｽｺｰﾙﾙｰﾁﾝの処理 */
    return E_OK;                             ←発行元にリターン値を返します。
}
```

図 3-3 HI7000/4 シリーズでの拡張サービスコールルーチンの C 言語記述例

```
ER_UINT Svcrtn(VP_INT par1, VP_INT par2)
{
    dis_dsp()                                ←ディスパッチ禁止*

   /* 拡張ｻｰﾋﾞｽｺｰﾙﾙｰﾁﾝの処理 */

   ena_dsp()                                 ←ディスパッチ許可*

    return E_OK;                             ←発行元にリターン値を返します。
}
```

【注】 * タスクコンテキストから呼び出した場合のみ、ディスパッチ禁止・許可を行ってください。
非タスクコンテキストから呼び出した場合は不要です。

図 3-4 HI7000 シリーズからの移植の場合での記述例（タスクコンテキスト）

3.3.3 イベントフラグの属性追加

HI7000/4 シリーズでは、イベントフラグに新しい属性（グレーの部分）が追加されています。**表 3-11** に変更点を示します。

表 3-11 イベントフラグ

イベントフラグ属性	属性の意味	HI7000 シリーズ	HI7000/4 シリーズ
TA_TFIFO	待ちタスクのキューイングは FIFO	×*	○
TA_TPRI	待ちタスクのキューイングは優先度順	×	○
TA_WSGL	複数タスクの待ちを許さない	○	○
TA_WMUL	複数タスクの待ちを許す	○	○
TA_CLR	待ち解除時にイベントフラグを 0 クリア	×	○

【注】 * HI7000 シリーズでは TA_TFIFO と同等に振舞います。

3.3.4 イベントフラグのクリア指定変更

HI7000 シリーズでは、イベントフラグのクリア指定はイベント待ちにする際のモードで設定しましたが、HI7000/4 シリーズではイベントフラグにクリア指定があります。クリア指定の変更内容を**図** 3-5 に示します。

移行の際は、一つのイベントフラグでクリア指定付きの待ちとクリア指定無しの待ちが混在していないかを注意し、混在していた場合は、クリア指定の順番に注意して移行してください.

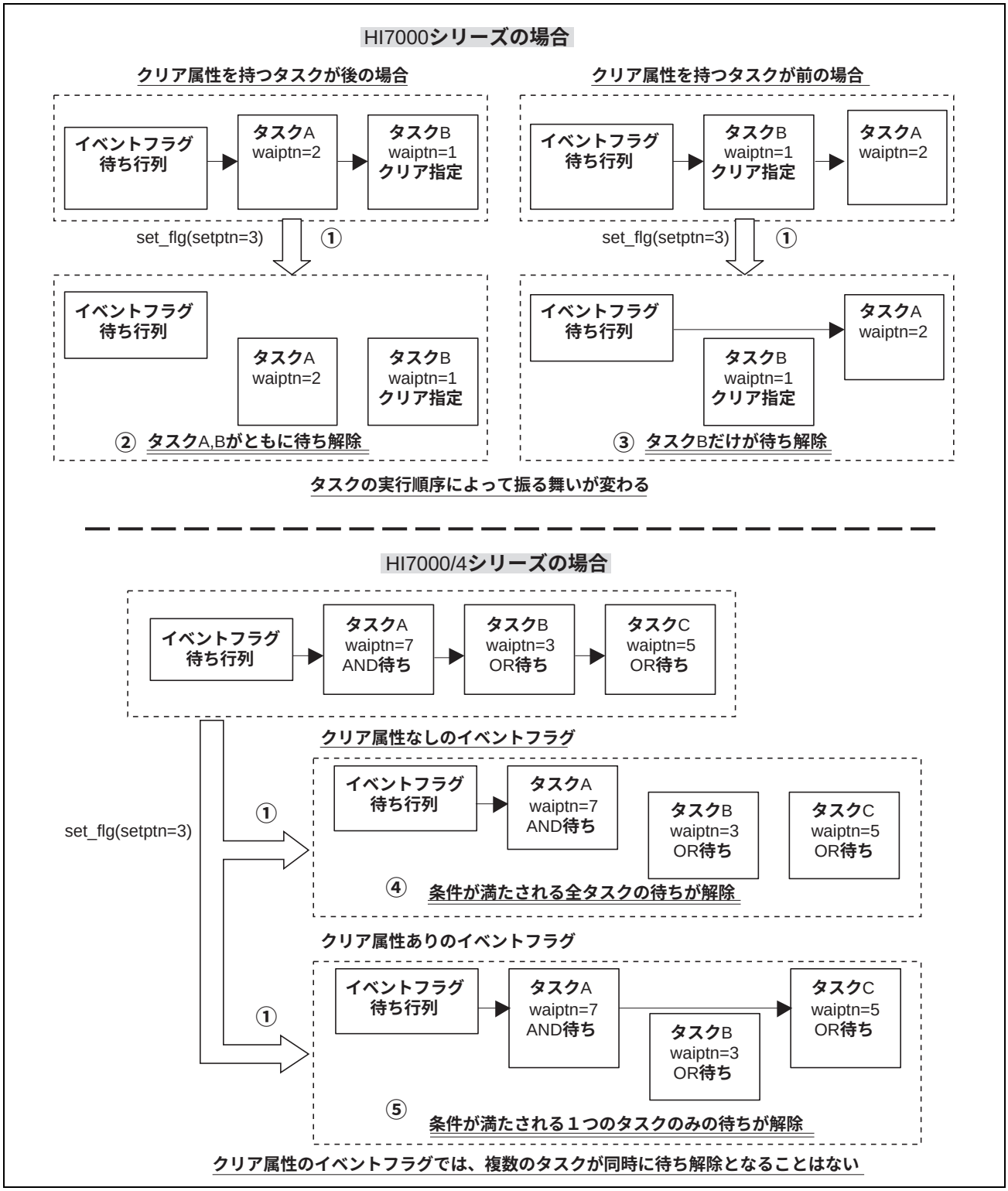

図 3-5 **イベントフラグのクリア指定変更**

①：イベントフラグにビットパターン（3）をセット。

②：タスク A はビットパターンが 2 のため、①のパターンで待ち状態が解除されます。クリア指定がないため次の待ちタスクについても判断を行います。
タスク B はビットパターンが 1 のため、①のパターンで待ち状態が解除されます。クリア指定がありますので処理を終了します。

③：タスク B はビットパターンが 1 のため、①のパターンで待ち状態が解除されます。タスク B にクリア指定があるため、後の待ち行列に繋がっているタスクについては判断を行わずに処理を終了します。

④：タスク A はビットパターンが 7 で AND 待ちのため、①のパターンとは一致せずに待ち状態のままです。
タスク B はビットパターンが 8 で OR 待ちのため、①のパターンで待ち解除されます。イベントフラグにクリア指定がないため待ち行列に繋がっているタスク全てについて判断を行います。
タスク C はビットパターンが 5 で OR 待ちのため、①のパターンで待ち解除されます。

⑤：タスク A はビットパターンが 7 で AND 待ちのため、①のパターンとは一致せずに待ち状態のままです。
タスク B はビットパターンが 8 で OR 待ちのため、①のパターンで待ち解除されます。イベントフラグにクリア指定があるため、タスク B 以降の待ち行列に繋がっているタスクについては判断を行わずに処理を終了します。

3.3.5 メールボックスの属性追加

HI7000/4 シリーズでは、メールボックスに新しい属性（グレーの部分）が追加されています。**表** 3-12 に変更点を示します。

表 3-12 **メールボックス**

メールボックス属性	属性の意味	HI7000 シリーズ	HI7000/4 シリーズ
TA_TFIFO	**受信待ちタスクのキューイングは** FIFO	○	○
TA_TPRI	**受信待ちタスクのキューイングは優先度順**	○	○
TA_MFIFO	**メッセージのキューイングは** FIFO	○	○
TA_MPRI	**メッセージのキューイングは優先度順**	×	○

3.3.6 メールボックスのメッセージヘッダ型追加

HI7000 シリーズではメイルボックスのメッセージヘッダは 1 種類でしたが、HI7000/4 シリーズのメールボックスのメッセージヘッダには、ヘッダのみと優先度付きヘッダの 2 種類があります。3.3.5 に伴う追加です。移植の際はアプリケーションプログラムの変更は不要です。

優先度付きのメッセージヘッダは、TA_MPRI 属性を指定した際に使用します。

T_MSG　　メールボックスのメッセージヘッダ
T_MSG_PRI　メールボックスの優先度付きメッセージヘッダ

```
typedef struct  t_msg {
   VP     msghead;     /* カーネル管理領域                        */
} T_MSG;
```

図 3-6　**メッセージヘッダ**

```
typedef   struct  t_msg_pri {
   T_MSG  msgque;     /* メッセージヘッダ                        */
   PRI    msgpri;     /* メッセージ優先度                        */
} T_MSG_PRI;
```

図 3-7　**優先度付きメッセージヘッダ**

以下にメッセージの例を示します。

```
typedef struct  user_msg {
   T_MSG  t_msg ;     /* T_MSG構造体                             */
   B      data[8];    /* ユーザメッセージデータ構造の例(任意の構造) */
} USER_MSG;
```

図 3-8　**メッセージの形式例**

```
typedef struct  user_msg {
   T_MSG_PRI    t_msg;     /* T_MSG_PRI構造体                          */
    B           data[8];  /* ユーザメッセージデータ構造の例(任意の構造) */
} USER_MSG;
```

図 3-9　**優先度付きメッセージの形式例**

3.3.7 メッセージバッファの待ち行列の変更

メッセージバッファの待ち行列に並ぶ際の並び方を指定する属性：TA_TPRI について、HI7000/4 シリーズでは「受信待ち行列が優先度順」から「送信待ち行列が優先度順」に変更になっています。詳しくは**表** 3-13 を参照してください。

表 3-13　**メッセージバッファの待ち行列意味付け変更**

メッセージバッファの待ち行列	HI7000 シリーズ	HI7000/4 シリーズ
受信待ち行列	TA_TFIFO 属性指定で FIFO／ TA_TPRI 属性指定で優先度順	FIFO のみ
送信待ち行列	FIFO のみ	TA_TFIFO 属性指定で FIFO／ TA_TPRI 属性指定で優先度順

3.3.8 wup_tsk, sus_tsk の変更

パラメータ：tskid=TSK_SELF（0）で自タスクの指定となります。
移行するアプリケーションプログラムが、E_ID のエラーによりプロセスを実行する場合は、修正する必要があります。

表 3-14　wup_tsk、sus_tsk **の変更**

パラメータ：tskid	HI7000 シリーズ	HI7000/4 シリーズ
TSK_SELF(0)	E_ID のエラー	自タスク指定

3.3.9 タイマドライバ（サンプル）の変更

タイマドライバ（サンプル）で明示的に vsys_clk(isig_tim)を発行しなくてもすむようになりました。
表 3-15 に、HI7000 シリーズと HI7000/4 シリーズの違いを示します。

表 3-15 **サンプルタイマドライバの変更**

番号	機能	HI7000 シリーズ	HI7000/4 シリーズ
1	タイマドライバの初期化	ユーザが、タイマ初期化ルーチンと、周期的に vsys_clk を発行するタイマ割り込みハンドラを作成して、システムに組み込む必要があります。	コンフィギュレータで CFG_TIMUSE を指定すると、タイマ初期化ルーチンとして_kernel_tmrini()が登録され、タイマ割り込みハンドラとして isig_tim が定義されます。
2	タイマ初期化ルーチン	名前は任意	_kernel_tmrini() ：固定
3	タイマ割り込みルーチン	名前は任意	_kernel_tmrint() ：固定
4	タイマ割り込み発生	ユーザが作成したタイマ割り込みハンドラを呼び出します。（ユーザがタイマ割り込みハンドラをシステムに組み込む必要あり）	isig_tim が割り込みハンドラとして起動し、isig_tim はタイマ割り込みルーチン_kernel_tmrint()を呼び出します。
5	システムクロックの更新	タイマ割り込みハンドラ内で明示的に vsys_clk を発行する必要があります。	isig_tim がシステムクロックを更新するので、明示的には必要ありません。

表 3-16 **サンプルタイマドライバのコーティング例**

ルーチン	HI7000 シリーズ	HI7000/4 シリーズ

タイマ初期化ルーチン	void timer_init(void)
	{
	/* タイマ初期化処理 */
	}
		void _kernel_tmrini(void)
		{
		/* タイマ初期化処理 */
		}

タイマ割り込みハンドラ	void timer_handor(void)
	{
	/* タイマ割り込みルーチン起動 */
	
	vsys_clk();
	}
		isig_tim()
		
		注：コンフィギュレータで設定されるので
		記述する必要はありません

タイマ割り込みルーチン	void timer_rutine(void)
	{
	/* タイマ割り込み処理 */
	}
		void _kernel_tmrint(void)
		{
		/* タイマ割り込み処理 */
		}

3.3.10 時間管理方法

時間パラメータの実時間と、時間管理が変更になっています。
HI7000 シリーズでの時間パラメータの単位時間は、ハードウェアタイマの割り込み周期の時間です。
HI7000/4 シリーズでの時間パラメータの単位時間は 1ms 固定です。
詳しくは**表** 3-17 を参照してください。

表 3-17　**時間パラメータの単位時間**

番号	変更点	HI7000 シリーズ	HI7000/4 シリーズ
1	**時間パラメータの単位時間**	**ハードウェアタイマの周期時間**	1ms(**固定**)
2	**タイムティック周期時間**	{（1／Q）×DIV} ×（n＋1）[ms] Q：**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：**タイマモジュールのカウンタレジスタに設定する値**	TIC_NUME／TIC_DENO [ms] TIC_NUME：**タイムティック周期の分子** TIC_DENO：**タイムティック周期の分母** **コンフィギュレータで設定します。なお** TIC_NUME, TIC_DENO **の少なくとも一方は** 1 **を指定しなければなりません**

時間管理の違いについて、HI7000 シリーズは、システムクロックが指定タイムアウト値を経過した時点でタイムアウトになりますが、HI7000/4 シリーズは、指定した時間が経過したことを保証しなければならないというμITRON4.0 仕様にあわせています。
HI7000 シリーズと HI7000/4 シリーズの時間管理の違いについては**図** 3-10 を参照してください。

移行の際は、例えばハードウェアタイマの周期時間とタイムティック周期時間が同じで、実際のタイムアウトまでの時間は、HI7000 シリーズの設定値から－1 したタイムアウト値を設定すれば、HI7000/4 シリーズでも同じ実時間が取れます。このように実際にタイムアウト時間を計算して移行を行ってください、

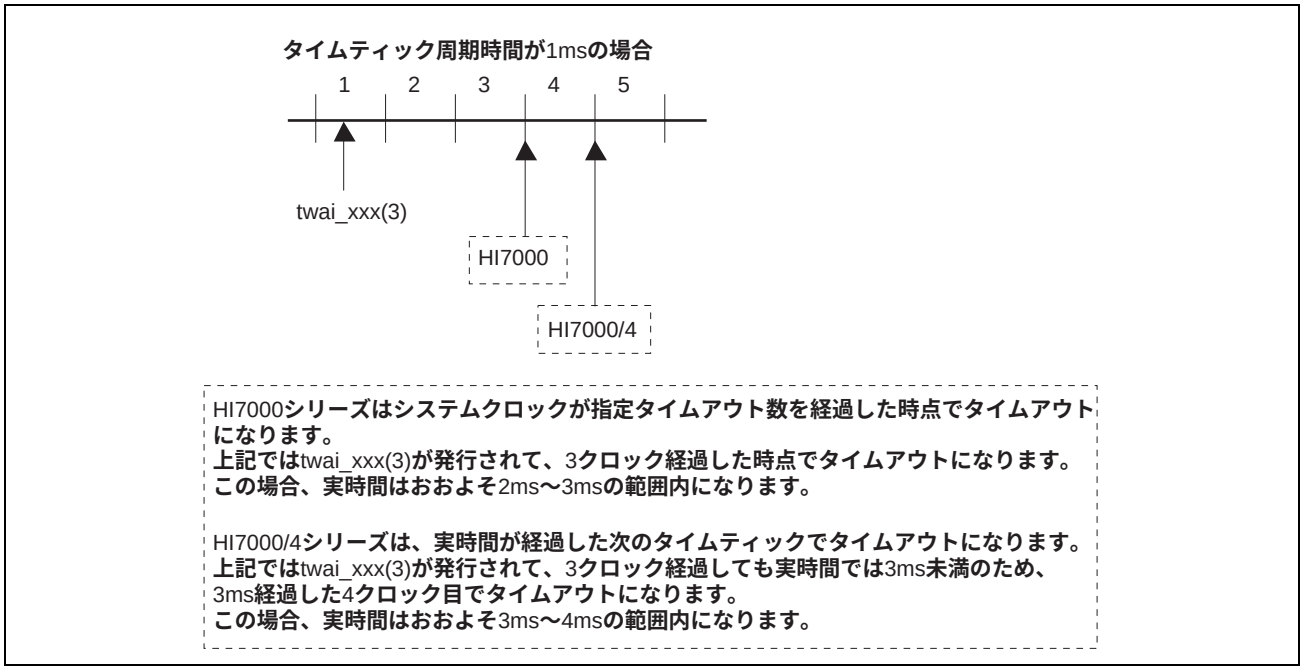

図 3-10　HI7000 **シリーズと** HI7000/4 **シリーズの時間管理の違い**

3.3.11 周期ハンドラ機能

HI7000/4 シリーズの周期ハンドラは、HI7000 シリーズに比べて全般に機能変更が行われています。
移行の際は、それぞれのマニュアルと、**表** 3-18、**表** 3-19、**図** 3-11 を参考にして修正を行ってください。

表 3-18　HI7000 **シリーズと** HI7000/4 **シリーズの周期ハンドラ比較表**

HI7000 シリーズの周期ハンドラ		HI7000/4 シリーズの周期ハンドラ	
def_cyc vasn_cyc	周期起動ハンドラ定義 周期起動ハンドラ指定番号の割り当てと定義	cre_cyc	周期ハンドラの生成
		icre_cyc	
		acre_cyc	周期ハンドラの生成
		iacre_cyc	(ID 自動割り付け)
act_cyc	周期起動ハンドラ活性制御	sta_cyc	周期ハンドラの動作開始
		ista_cyc	
		stp_cyc	周期ハンドラの動作停止
		istp_cyc	
ref_cyc ｆｆ	周期起動ハンドラ状態参照	ref_cyc	周期ハンドラの状態参照
		iref_cyc	
ret_tmr	タイマハンドラから復帰	—	自動的に終了
—	—	del_cyc	周期ハンドラの削除

表 3-19　**周期ハンドラ移行の設定**

動作	HI7000 シリーズの設定	HI7000/4 シリーズの設定
周期ハンドラ動作開始（起動位相保存）	act_cyc:TCY_ON	sta_cyc (cre_cyc で TA_PHS を指定)
周期ハンドラ動作開始（起動位相クリア）	act_cyc:TCY_INI	sta_cyc (cre_cyc で TA_PHS を指定しない)
周期ハンドラ動作停止	act_cyc:TCY_OFF	stp_cyc

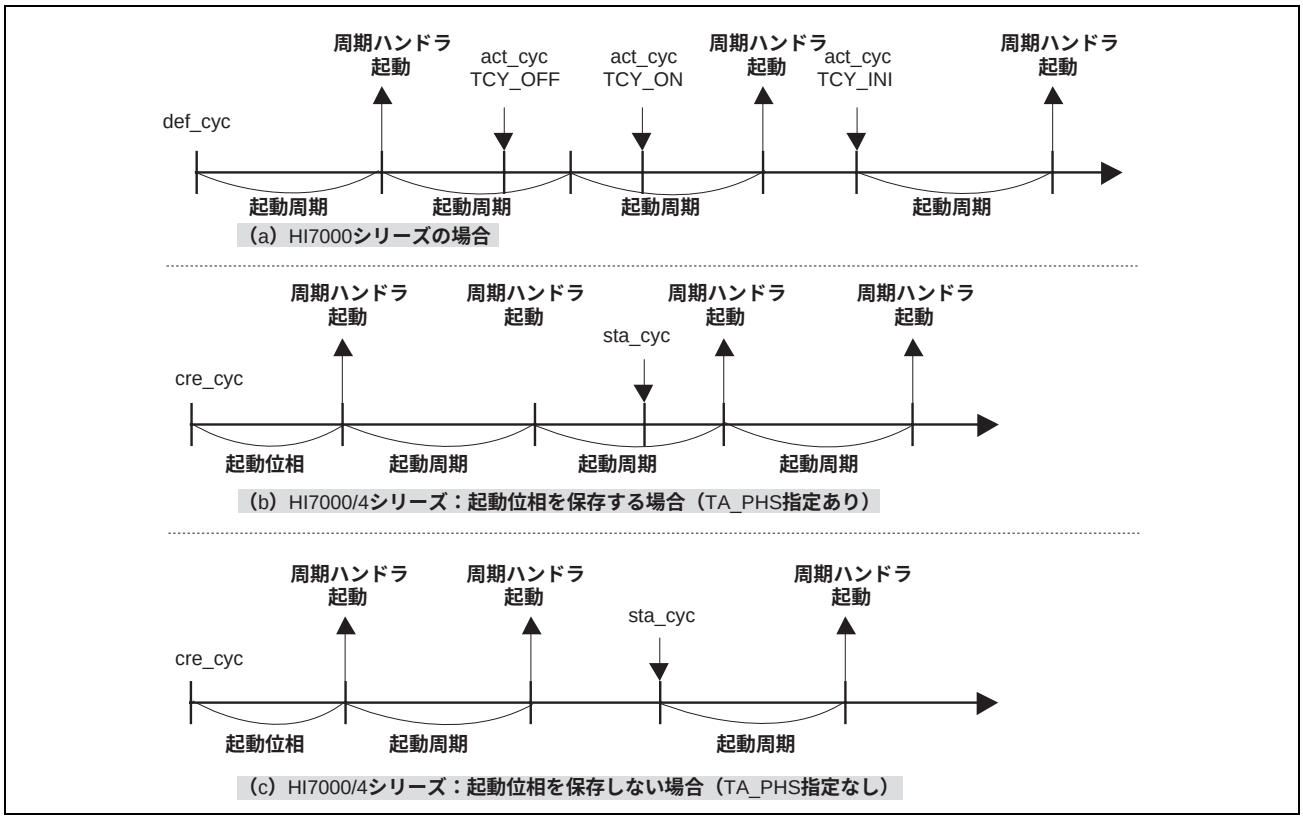

図 3-11　**周期ハンドラの動作の違い**

3.3.12 アラームハンドラ機能

HI7000/4 シリーズのアラームハンドラは、HI7000 シリーズに比べて全般に機能変更が行われています。
移行の際は、それぞれのマニュアルと**表** 3-20 を参考にして修正を行ってください。

表 3-20　HI7000 **シリーズと** HI7000/4 **シリーズのアラームハンドラ比較表**

HI7000 シリーズのアラームハンドラ		HI7000/4 シリーズのアラームハンドラ	
def_alm vasn_alm	**アラームハンドラ定義** **アラームハンドラ指定番号の割り当てと定義**	cre_alm	**アラームハンドラの生成**
		icre_alm	
		acre_alm	**アラームハンドラの生成** (ID **番号自動割り付け**)
		iacre_alm	
		sta_alm	**アラームハンドラの動作開始**
		ista_alm	
		stp_alm	**アラームハンドラの動作停止**
		istp_alm	
ref_alm	**アラームハンドラ状態参照**	ref_alm	**アラームハンドラの状態参照**
		iref_alm	
ret_tmr	**タイマハンドラから復帰**	—	**自動的に終了**
—	—	del_alm	**アラームハンドラの削除**

3.3.13 SH-4 **での** TA_COP1(FPU **ハンドラ** 0)、TA_COP2(FPU **バンク** 1)**属性**

HI7750 で通常の浮動小数点演算を使う場合、タスクやハンドラの起動では、TA_COP1 指定でバンク 0(FPSCR.FR=0)、TA_COP2 指定でバンク 1 FPSCR.FR=1)でした。

HI7750/4 で通常の浮動小数点演算を使う場合、ルーチンのエントリ関数の先頭で FPSCR を初期化し、バンク 0(FPSCR.FR=0)にする必要があります。属性の指定は、TA_COP1 固定になります。

表 3-21　TA_COP1、TA_COP2 **属性の意味変更**

属性指定	**タスク起動時の状態**	
	HI7750	HI7750/4
TA_COP1	**バンク 0**	**バンク 0**
TA_COP2	**バンク 1**	**指定できません**

```
#include <machine.h>            /* 組み込み関数set_fpscr()を使用するためにインクルード */
#define INI_FPSCR 0x00080000    /* FPSCR初期値(FR=0, PR=1, DN=0, SZ=0, RM=B00) */
#pragma noregsave(Task)
void Task(VP_INT exinf)
{
    set_fpscr(INI_FPSCR);       /* タスクの先頭でFPSCRを初期化 */
    /* タスクの処理 */
    ext_tsk();
}
```

図 3-12　**タスクにおける** FPSCR **の初期化例**

3.3.14 オブジェクト名称機能

HI7000 シリーズでのオブジェクト名称、オブジェクト名称機能が削除され、「オブジェクト名称から ID 番号を取得する」ことができなくなりました。HI7000/4 シリーズではこれらの機能の代替えとして、コンフィギュレータの機能を使います。

HI7000/4 シリーズでは、コンフィギュレータでオブジェクト生成時に、ID 番号の自動割り付け機能で ID 名称を指定します。その ID 名称は ID 番号として扱うことができます。**表** 3-22 に代替えの詳細を示します。

表 3-22 **オブジェクト名称機能の代替**

番号	機能	HI7000 シリーズ	HI7000/4 シリーズ
1	**オブジェクト**/ID **名称の設定**	**オブジェクトの生成時**	**コンフィギュレータ***
2	ID **番号の取得**	**オブジェクト名称管理サービスコールにより取得**	ID **名称ヘッダファイル**(kernel_id.h, kernel_id_sys.h)**をインクルードすることにより、**ID **名称が** ID **番号として使用可能**

【注】 * ID **名称を指定できるのは製品ごとにサポート範囲が異なります。詳しくは該当マニュアルのコンフィギュレーションの章をご覧ください。**

3.3.15 システムダウンルーチン

HI7000/4 シリーズのシステムダウンルーチンは、HI7000 シリーズと比べていくつかの変更が行われています。
表 3-23 にルーチン名の変更、**表** 3-24 に渡される情報の比較を示します。

表 3-23 **システムダウンルーチン名の変更**

OS	**システムダウンルーチン名（固定）**
HI7000 **シリーズ**	void hi_sysdwn(W type, ER ercd, VW inf1, VW inf2)
HI7000/4 **シリーズ**	void _kernel_sysdwn(W type, ER ercd, VW inf1, VW inf2)

HI7000/4 シリーズでは HI7000 シリーズに比べてシステムダウン要因の数が減っています。
表 3-24 では、HI7000/4 シリーズのシステムダウン情報にグレーの網掛けをしています。項番が同じ項目は同一のシステムダウン要因です。

表 3-24 **システムダウンルーチンに渡される情報**

項番	OS	**システムダウン要因**	**システムダウンルーチンに渡されるパラメータ**			
			エラー種別 W type (R4)	**エラーコード** ER ercd (R5)	**システムダウン情報** 1 VW inf1 (R6)	**システムダウン情報** 2 VW inf2 (R7)
1	HI7000 **シリーズ**	vsys_dwn **システムコール**	1〜H'7fffffff	vsys_dwn **システムコールのパラメータ**		
	HI7000/4 **シリーズ**	vsys_dwn, ivsys_dwn **サービスコール**	1〜H'7fffffff	vsys_dwn, ivsys_dwn **サービスコールのパラメータ**		
2	HI7000 **シリーズ**	**カーネルビルドファイルまたはセットアップファイルの初期登録情報に誤りがあった場合**	0	**初期登録に対するシステムコールのエラーコード**	0:**カーネルビルドファイルでのエラー** 1:**セットアップファイルでのエラー**	**カーネルビルドファイルまたはセットアップファイルで何番目の初期登録情報でエラーとなったかを示す**
	HI7000/4 **シリーズ**	**コンフィギュレータでの初期登録情報に誤りがあった場合**	0	**エラーコード**	0:**カーネル側** 1:**カーネル環境側**	**カーネル側またはカーネル環境側の何番目の登録でエラーとなったかを示す** *5
3	HI7000 **シリーズ**	**タスク独立部からの** ext_tsk **システムコールでコンテキストエラーが発生**	H'ffffffff(-1)	E_CTX (h'ffffffbb)	ext_tsk **を発行したアドレス**+2	**不定**
	HI7000/4 **シリーズ**	**非タスクコンテキストからの** ext_tsk **サービスコールでコンテキストエラーが発生**	H'ffffffff(-1)	E_CTX (h'ffffffe7)	ext_tsk **を呼び出したアドレス**	**不定**
4	HI7000 **シリーズ**	**タスク独立部からの** exd_tsk **システムコールでコンテキストエラーが発生**	H'fffffffe(-2)	E_CTX (h'ffffffbb)	exd_tsk **を発行したアドレス**+2	**不定**
	HI7000/4 **シリーズ**	**非タスクコンテキストからの** exd_tsk **サービスコールでコンテキストエラーが発生**	H'fffffffe(-2)	E_CTX (h'ffffffe7)	exd_tsk **を呼び出したアドレス**	**不定**
5	HI7000 **シリーズ**	**タスク独立部からの** ret_int **システムコールでコンテキストエラーが発生** *3	H'fffffffd(-3)	E_CTX (h'ffffffbb)	ret_int **を発行したアドレス**+2	**不定**
6	HI7000 **シリーズ**	**タスク独立部以外の状態からの** vsys_clk **システムコールでコンテキストエラーが発生**	H'fffffffc(-4)	E_CTX (h'ffffffbb)	vsys_clk **を発行したアドレス**+2	**不定**
7	HI7000 **シリーズ**	**拡張** SVC **ハンドラ**,I/O **ハンドラ以外からの** ret_svc **システムコールでコンテキストエラーが発生**	H'fffffffb(-5)	E_CTX (h'ffffffbb)	ret_svc **を発行したアドレス**+2	**不定**

項番	OS	システムダウン要因	システムダウンルーチンに渡されるパラメータ			
			エラー種別 W type (R4)	エラーコード ER ercd (R5)	システムダウン情報 1 VW inf1 (R6)	システムダウン情報 2 VW inf2 (R7)
8	HI7000 シリーズ	タスク独立部からの vrst_tsk システムコールでコンテキストエラーが発生、または vrst_tsk システムコールでパラメータエラーが発生	H'ffffffff9(-7)	E_CTX (h'ffffffbb) または E_PAR (h'ffffffdf)	vrst_tsk を発行したアドレス+2	不定
9	HI7000 シリーズ	未定義割り込み、例外が発生	H'ffffffff0(-16)	例外コード *2 またはベクタ番号	例外発生時の PC *1	例外発生時の SR *1
	HI7000/4 シリーズ	未定義割り込み、例外が発生	H'ffffffff0(-16)	例外コードまたはベクタ番号 *4	例外発生時の PC	例外発生時の SR

【注】*1 割り込み・例外・無条件トラップ発生時に、CPU が退避する情報です。

*2 HI7700、HI7750 の場合は例外コードが渡されます。割り込み・例外・無条件トラップ発生時に、CPU が割り込み事象レジスタ（INTEVT または INTEVT2）または例外事象レジスタ（EXPEVT）に設定する情報です。
HI7000 の場合はベクタ番号が渡されます。
カーネルビルドファイルで、マクロ hi_anavec を NOTUSE とした場合は-1 となります。

*3 HI7000 のみ発生するエラーです。

*4 HI7700/4、HI7750/4 の場合は、割り込み・例外・無条件トラップ発生時に、CPU が割り込み事象レジスタ（INTEVT または INTEVT2）または例外事象レジスタ（EXPEVT）に設定する情報です。

*5 初期登録は、カーネル側が先にカーネル環境側が後に処理されます。カーネル側およびカーネル環境側での初期登録の処理順は以下の通りで、それぞれの中の順序は各ビューでのリストに表示されている順序です。ただし、カーネル側の(1)～(3)でシステムダウンが発生することはありえないため、初期登録のカウントには含めません。
(1)割り込み,CPU 例外ハンドラビューでの割り込みハンドラの初期登録
(2)割り込み,CPU 例外ハンドラビューでの CPU 例外ハンドラの初期登録
(3)トラップ例外ハンドラビューでのトラップ用 CPU 例外ハンドラの初期登録
(4)タスクビューでのタスクおよびタスク例外処理ルーチンの初期登録
(5)セマフォビューでのセマフォの初期登録
(6)イベントフラグビューでのイベントフラグの初期登録
(7)データキュービューでのデータキューの初期登録
(8)メールボックスビューでのメールボックスの初期登録
(9)ミューテックスビューでのミューテックスの初期登録
(10)メッセージバッファビューでのメッセージバッファの初期登録
(11)固定長メモリプールビューでの固定長メモリプールの初期登録
(12)可変長メモリプールビューでの可変長メモリプールの初期登録
(13)周期ハンドラビューでの周期ハンドラの初期登録
(14)アラームハンドラビューでのアラームハンドラの初期登録
(15)オーバーランハンドラビューでのオーバーランハンドラの初期登録
(16)拡張サービスコールビューでの拡張サービスコールの初期登録

3.3.16 コーディング例

(1) タスク

タスクのコーディング例を**図** 3-13、**図** 3-14 に示します。

```
#include “itron.h”
#pragma noregsave(task)    ←タスク関数はリターンしないので、#pragma noregsaveを指定することができます。

void Task (INT stacd)
{
    /* タスクの処理 */
    ext_tsk();             ←タスクを終了する場合は、必ずext_tskまたはexd_tskシステムコールを発行してください。
}
```

図 3-13　HI7000 **シリーズ：タスクのコーディング例**

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

#pragma noregsave(Task)    ←タスク関数はレジスタを保証する必要はないので、#pragma noregsaveを指定することができます。

void  Task(VP_INTexinf)    ←タスク生成時にTA_ACT属性が指定されて起動された場合、およびact_tskで起動された場合には、
                             パラメータとしてタスク生成時に指定されたexinfが渡され、sta_tskで起動された場合はsta_tsk
                             で指定されたstacdが渡されます。
{
   /* タスクの処理 */
   ext_tsk();              ←タスクを終了する場合は、ext_tskまたはexd_tskサービスコールを呼び出してください。

}                          ←関数の終了で、ext_tskが自動的に呼び出されます。
```

図 3-14　HI7000/4 **シリーズ：タスクのコーディング例**

(2) **拡張サービスコール(拡張** SVC)

拡張サービスルーチンのコーディング例を**図** 3-15、**図** 3-16 に示します。

```
#include “itron.h”
#pragma noregsave(ExHandler)     ←拡張SVCハンドラ関数はリターンしないので、#pragma noregsaveを指定することができます。

void  ExHandler (ID reqid, UW    ←拡張SVCハンドラには、拡張SVCで指定したパラメータが渡されます。
par1)
{
    /* 拡張SVCハンドラの処理 */     ←拡張SVCハンドラの最後では、ret_svcシステムコールで終了してください。ret_svcのパラメータが、拡張SVCの呼び出し元にエラーコードとして返ります。
    ret_svc(error_code);
}
```

図 3-15　HI7000 **シリーズ：拡張** SVC **のコーディング例**

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

ER_UINT Svcrtn(VP_INT par1, VP_INT par2) ←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。
{
    /* 拡張ｻｰﾋﾞｽｺｰﾙﾙｰﾁﾝの処理 */              ←発行元にリターン値を返します。
    return E_OK;
}
```

図 3-16　HI7000/4 **シリーズ：拡張サービスコールルーチンのコーディング例**

(3) 割り込みハンドラ

割り込みハンドラのコーディング例を**図** 3-17、**図** 3-18、**図** 3-19 に示します。

```
#include “itron.h”
#define stksz 512                          ←割り込みハンドラで使用するスタックを確保します。
                                             同じ割り込みレベルのハンドラは、スタックを共有できます。
VW  stk[stksz / sizeof(VW)];
static const VP
    p_stk=(VP)&stk[stksz/sizeof(VW)];      ←スタックポインタの初期値をconst型として定義します。
#pragma interrupt(Isr(sp=p_stk,tn=25))
/* #pragma interrupt(Isr(sp=p_stk)) */     ←#pragma interruptにより、割り込みハンドラを割り込み関数として宣言します。割り
                                             込み仕様として以下を指定してください。

                                           ・スタック切り替え指定（sp=）
                                           ・トラップリターン指定（tn=25）
                                             カーネル割り込みマスクレベル以下のレベルの割り込みハンドラはret_intシステムコ
                                             ールで終了する必要があるので、“tn=25”を指定してください。カーネル割り込みマス
                                             クレベルよりも高いレベルの割り込みハンドラはRTE命令で終了する必要があるので、ト
                                             ラップリターン指定は行わないでください。

void Isr(void)                             ←割り込みハンドラは、void型の関数として記述します。
{
    /* 割り込みハンドラの処理 */
}                                          ←#pragma interruptの指定により、自動的にret_intシステムコールが実行されます。
```

図 3-17　HI7000 **シリーズ：割り込みハンドラのコーディング例**

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

#pragma noregsave(Inh)          ←SH-2A, SH2A-FPUでバンク割り込みを使用する場合(CFG_REGBANKをチェック)は、レジスタ
                                   を保証する必要はないので、#pragma noregsaveを指定することができます。

void Inh(void)                  ←割り込みハンドラは、void型の関数として記述します。
{
    /* 割り込みハンドラの処理 */
}
```

図 3-18　HI7000/4 **シリーズ：通常の割り込みハンドラのコーディング例**

```
#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)
{
    /* 割り込みハンドラの処理 */
}
```

図 3-19　HI7000/4 **シリーズ：ダイレクト割り込みハンドラのコーディング例**

（1）　割り込みハンドラで使用するスタックを確保します。NMI 以外の割り込みは、必ずスタックを切り替えなければなりません。これは、割り込まれる側のスタックのオーバフローを避けるためです。
同じ割り込みレベルのハンドラは、スタックを共有できます。ただし、NMI 割り込みハンドラは専用のスタックを持つことはできません。

（2）　スタックポインタの初期値を const 型として定義します。

（3）　#pragma interrupt により、割り込みハンドラを割り込み関数として宣言します。割り込み関数仕様として、以下を指定してください。

（a）"sp="指定（スタック切り替え）
スタックを切り替える場合は、（2）の変数を指定してください。

（b）"tn="指定（トラップリターン指定）
カーネル割り込みマスクレベルよりも高いレベル（NMI を含む）の割り込みハンドラは RTE 命令で終了する必要があるので、トラップリターン指定は行わないでください。
カーネル割り込みマスクレベル以下のレベルの割り込みの場合は、"tn=25"を指定してください。

（c）"resbank"指定（バンク割り込み）
SH-2A および SH2A-FPU において、バンク割り込みを使用する場合（CFG_REGBANK をチェック）に指定してください。ただし、NMI の場合は、指定してはなりません。

（4）　割り込み関数は、void 型の関数として記述します。

4. サービスコールの相違

4.1 SVC 相違一覧表

4.1.1 タスク管理機能

- グレーの網掛けがかかっているところは変更になっている箇所です。
- この SVC 相違一覧表では用語が以下のように置き換えられています。
  パラメータ → 引数　　リターンパラメータ → 返値

	HI7000 シリーズ（μITRON3.0 仕様準拠）		HI7000/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
1	**タスク生成(ダイナミックスタック使用)**		**タスク生成(ダイナミックスタック使用)**		**パケット構造　変更**
	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);	
		パケット構造 typedef struct t_ctsk { VP exinf; **拡張情報** ATR tskatr; **タスク属性** FP task; **タスク起動アドレス** PRI itskpri; **初期タスク優先度** UB cpumode; **タスク CPU モード** INT stksz; **タスクスタックサイズ** TMO quantum; **タイムスライス時間** B *name; **名称を格納した領域の先頭アドレス** } T_CTSK;		**パケット構造** typedef struct t_ctsk { ATR tskatr; **タスク属性** VP_INT exinf; **拡張情報** FP task; **タスク起動アドレス** PRI itskpri; **初期タスク優先度** SIZE stksz; **タスクスタックサイズ** VP stk; **タスクスタック領域の先頭アドレス** } T_CTSK;	
2	**タスク生成(スタティックスタック使用)**		**－**		**廃止**
	vcre_tsk	ER ercd = vcre_tsk(ID tskid, TASKP stadr, TPRI itskpri);	**該当なし**		
3	**タスク生成(スタティックスタック使用)**		**タスク生成(スタティックスタック使用)**		**パケット構造　変更**
	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);	
		パケット構造 typedef struct t_ctsk { VP exinf; **拡張情報** ATR tskatr; **タスク属性** FP task; **タスク起動アドレス** PRI itskpri; **初期タスク優先度** UB cpumode; **タスク CPU モード** INT stksz; **タスクスタックサイズ** TMO quantum; **タイムスライス時間** B *name; **名称を格納した領域の先頭アドレス** } T_CTSK;		**パケット構造** typedef struct t_ctsk { ATR tskatr; **タスク属性** VP_INT exinf; **拡張情報** FP task; **タスク起動アドレス** PRI itskpri; **初期タスク優先度** SIZE stksz; **タスクスタックサイズ** VP stk; **タスクスタック領域の先頭アドレス** } T_CTSK;	

	HI7000 シリーズ（μITRON3.0 仕様準拠）		HI7000/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
4	タスク ID の割り当てと生成(ダイナミックスタック使用)		タスク生成(ID 番号自動割り付け）(ダイナミックスタック使用)		SVC 名、引数、返値、パケット構造　変更
	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);	
	引数	ID *p_tskid; 生成したタスク ID を返す領域の先頭 T_CTSK *pk_ctsk; アドレス タスク生成情報の先頭アドレス	引数	T_CTSK *pk_ctsk; タスク生成情報を格納したパケットへのポインタ	
	返値	ER ercd; エラーコード ID *p_tskid; 生成したタスク ID を格納した領域の先頭アドレス	返値	ER_ID tskid; 生成したタスク ID(正の値)またはエラーコード	
	パケット構造	typedef struct t_ctsk { VP exinf; 拡張情報 ATR tskatr; タスク属性 FP task; タスク起動アドレス PRI itskpri; 初期タスク優先度 UB cpumode; タスク CPU モード INT stksz; タスクスタックサイズ TMO quantum; タイムスライス時間 B *name; 名称を格納した領域の先頭アドレス } T_CTSK;	パケット構造	typedef struct t_ctsk { ATR tskatr; タスク属性 VP_INT exinf; 拡張情報 FP task; タスク起動アドレス PRI itskpri; 初期タスク優先度 SIZE stksz; タスクスタックサイズ VP stk; タスクスタック領域の先頭アドレス } T_CTSK;	
5	タスク削除		タスク削除		
	del_tsk	ER ercd = del_tsk(ID tskid);	del_tsk	ER ercd = del_tsk(ID tskid);	
6	タスク起動		タスク起動		引数型変更
	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);	
	引数	INT stacd タスク起動コード	引数	VP_INT stacd タスク起動コード	
−	−		タスクの起動		新機能
	−	該当なし	act_tsk iact_tsk	ER ercd = act_tsk(ID tskid); ER ercd = iact_tsk(ID tskid);	
−	−		タスク起動要求のキャンセル		新機能
	−	該当なし	can_act ican_act	ER_UINT actcnt = can_act(ID tskid); ER_UINT actcnt = ican_act(ID tskid);	
7	自タスクの再起動		−		廃止
	vrst_tsk	void vrst_tsk(FP task, INT stacd);	−	該当なし	3.1.4 参照
8	自タスク終了		自タスク終了		
	ext_tsk	void ext_tsk(void);	ext_tsk	void ext_tsk();	
9	自タスクの終了と削除		自タスクの終了と削除		
	exd_tsk	void exd_tsk(void);	exd_tsk	void exd_tsk();	
10	他タスク強制終了		他タスク強制終了		
	ter_tsk	ER ercd = ter_tsk(ID tskid);	ter_tsk	ER ercd = ter_tsk(ID tskid);	
11	ディスパッチ禁止		ディスパッチ禁止		ディスパッチの禁止の意味が変更 3.3.1 参照
	dis_dsp	ER ercd = dis_dsp(void);	dis_dsp	ER ercd = dis_dsp();	
12	ディスパッチ許可		ディスパッチ許可		ディスパッチの許可の意味が変更 3.3.1 参照
	ena_dsp	ER ercd = ena_dsp(void);	ena_dsp	ER ercd = ena_dsp();	

	HI7000 シリーズ（μITRON3.0 仕様準拠）		HI7000/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
13	タスク優先度変更		タスク優先度変更		
	chg_pri	ER ercd = chg_pri(ID tskid, PRI tskpri);	chg_pri	ER ercd = chg_pri(ID tskid, PRI tskpri);	
			ichg_pri	ER ercd = ichg_pri(ID tskid, PRI tskpri);	
−	−		タスク優先度の参照		新機能
	−	該当なし	get_pri	ER ercd = get_pri(ID tskid, PRI *p_tskpri);	
			iget_pri	ER ercd = iget_pri(ID tskid, PRI *p_tskpri);	
14	タスクのレディキュー回転		タスクのレディキュー回転		
	rot_rdq	ER ercd = rot_rdq(PRI tskpri);	rot_rdq	ER ercd = rot_rdq(PRI tskpri);	
			irot_rdq	ER ercd = irot_rdq(PRI tskpri);	
15	他タスクの待ち状態解除		他タスクの待ち状態解除		
	rel_wai	ER ercd = rel_wai(ID tskid);	rel_wai	ER ercd = rel_wai(ID tskid);	
			irel_wai	ER ercd = irel_wai(ID tskid);	
16	自タスク ID 参照		実行状態のタスク ID の参照		非タスクコンテキストで動作が違う
	get_tid	ER ercd = get_tid(ID *p_tskid);	get_tid	ER ercd = get_tid(ID *p_tskid);	
			iget_tid	ER ercd = iget_tid(ID *p_tskid);	
		動作 タスク独立部から発行した場合は、タスク ID として FALSE（0）を返す。		動作 非タスクコンテキストから呼び出された場合はそのとき実行状態のタスク ID を返す。	
17	タスク状態参照		タスク状態参照		引数並び、パケット構造　変更
	ref_tsk	ER ercd = ref_tsk(T_RTSK *pk_rtsk, ID tskid);	ref_tsk	ER ercd = ref_tsk(ID tskid, T_RTSK *pk_rtsk);	
			iref_tsk	ER ercd = iref_tsk(ID tskid, T_RTSK *pk_rtsk);	
		パケット構造 typedef struct t_rtsk { VP exinf; 拡張情報 PRI tskpri; 現在のタスク優先度 UINT tskstat; タスク状態 UINT tskwait; 待ち要因 ID wid; 待ちオブジェクト ID H wupcnt; 起床要求カウント H suscnt; SUSPEND 要求カウント UH tskmode; タスクの実行モード UH pndptn; 遅延されている要求の有無 FN s_fncd; 現在実行中の拡張 SVC 番号 DVN iodvn; 現在実行中の I/O 番号 INT remstksz; 残りスタックサイズ UINT tflptn; 現在のタスク付属イベントフラグの値 FP task; タスク起動アドレス PRI itskpri; タスク起動時優先度 UB cpumode; タスク CPU モード TMO quantum; タイムスライス時間 B *name; 名称を返す領域の先頭アドレス } T_RTSK;		パケット構造 typedef struct t_rtsk { STAT tskstat; タスク状態 PRI tskpri; タスクの現在優先度 PRI tskbpri; タスクのベース優先度 STAT tskwait; 待ち要因 ID wobjid; 待ちオブジェクト ID TMO lefttmo; タイムアウトするまでの時間 UINT actcnt; 起動要求キューイング数 UINT wupcnt; 起床要求キューイング数 UINT suscnt; 強制待ち要求ネスト数 UINT tskmode; タスクの実行モード UINT tflptn; 現在のタスク付属イベントフラグの値 } T_CTSK;	
18	タスク状態参照		−		廃止
	vtsk_sts	ER ercd = vtsk_sts(UH *p_tskstat, TPRI *p_tskpri, ID tskid);	−	該当なし	

HI7000 シリーズ（μITRON3.0 仕様準拠）						HI7000/4 シリーズ（μITRON4.0 仕様準拠）					変更内容
	機能					機能					
	SVC	API				SVC	API				
―	―					タスクの状態参照(簡易版)					新機能
	―	該当なし				ref_tst	ER ercd = ref_tst(ID tskid, T_RTST *pk_rtst); ER ercd = iref_tst(ID tskid, T_RTST *pk_rtst);				
19	タスク実行モードの設定					タスク実行モードの変更					vchg_tmd に統合
	vset_tmd	ER ercd = vset_tmd(UINT settmd);				vchg_tmd	ER ercd = vchg_tmd(UINT tmd);				
		引数	UINT	settmd;	セットするタスク実行モード		引数	UINT	tmd;	変更するタスク実行モード	
		返値	ER	ercd;	エラーコード		返値	ER	ercd;	正常終了（E_OK）またはエラーコード	
20	タスク実行モードのクリア										
	vclr_tmd	ER ercd = vclr_tmd(UINT clrtmd);									
		引数	UINT	clrtmd;	クリアするタスク実行モード						
		返値	ER	ercd;	エラーコード						

4.1.2 タスク付属同期機能

- グレーの網掛けがかかっているところは変更になっている箇所です。
- この SVC 相違一覧表では用語が以下のように置き換えられています。
  パラメータ → 引数　　リターンパラメータ → 返値

	HI7000 シリーズ（μITRON3.0 仕様準拠）		HI7000/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
1	**他タスクを強制待ち状態へ移行**		**タスクを強制待ち状態へ移行**		**自タスク指定可能に** 3.3.8 **参照**
	sus_tsk	ER ercd = sus_tsk(ID tskid);	sus_tsk	ER ercd = sus_tsk(ID tskid);	
			isus_tsk	ER ercd = isus_tsk(ID tskid);	
		動作 **自タスクは指定不可**		**動作** tskid=0 **で自タスクが指定可能**	
2	**強制待ち状態のタスクを再開**		**強制待ち状態のタスクを再開**		
	rsm_tsk	ER ercd = rsm_tsk(ID tskid);	rsm_tsk	ER ercd = rsm_tsk(ID tskid);	
			irsm_tsk	ER ercd = irsm_tsk(ID tskid);	
3	**強制待ち状態のタスクを強制再開**		**強制待ち状態のタスクを強制再開**		
	frsm_tsk	ER ercd = frsm_tsk(ID tskid);	frsm_tsk	ER ercd = frsm_tsk(ID tskid);	
			ifrsm_tsk	ER ercd = ifrsm_tsk(ID tskid);	
4	**自タスクを起床待ち状態へ移行**		**自タスクを起床待ち状態へ移行**		
	slp_tsk	ER ercd = slp_tsk(void);	slp_tsk	ER ercd = slp_tsk();	
5	**自タスクを起床待ち状態へ移行(タイムアウト有)**		**自タスクを起床待ち状態へ移行(タイムアウト有)**		
	tslp_tsk	ER ercd = tslp_tsk(TMO tmout);	tslp_tsk	ER ercd = tslp_tsk(TMO tmout);	
6	**タスクの起床**		**タスクの起床**		**自タスク指定可能に** 3.3.8 **参照**
	wup_tsk	ER ercd = wup_tsk(ID tskid);	wup_tsk	ER ercd = wup_tsk(ID tskid);	
			iwup_tsk	ER ercd = iwup_tsk(ID tskid);	
		動作 **自タスクは指定不可**		**動作** tskid=0 **で自タスクが指定可能**	
7	**タスクの起床要求を無効化**		**タスクの起床要求を無効化**		**引数、返値変更**
	can_wup	ER ercd = can_wup(INT *p_wupcnt, ID tskid);	can_wup	ER_UINT wupcnt = can_wup(ID tskid);	
			ican_wup	ER_UINT wupcnt = ican_wup(ID tskid);	
		引数 INT *p_wupcnt; **起床要求回数を返す領域の先頭アドレス** ID tskid; **タスク** ID		**引数** ID tskid; **タスク** ID	
		返値 ER ercd; **エラーコード** INT *p_wupcnt; **起床要求回数を格納した領域の先頭アドレス**		**返値** ER_UINT wupcnt; **キューイングされていた起床要求の回数(正の値または 0) またはエラーコード**	
8	**タスク付属イベントフラグのセット**		**タスク付属イベントフラグのセット**		
	vset_tfl	ER ercd = vset_tfl(ID tskid, UINT setptn);	vset_tfl	ER ercd = vset_tfl(ID tskid, UINT setptn);	
			ivset_tfl	ER ercd = ivset_tfl(ID tskid, UINT setptn);	
9	**タスク付属イベントフラグのクリア**		**タスク付属イベントフラグのクリア**		
	vclr_tfl	ER ercd = vclr_tfl(ID tskid, UINT clrptn);	vclr_tfl	ER ercd = vclr_tfl(ID tskid, UINT clrptn);	
			ivclr_tfl	ER ercd = ivclr_tfl(ID tskid, UINT clrptn);	
10	**タスク付属イベントフラグ待ち**		**タスク付属イベントフラグ待ち**		
	vwai_tfl	ER ercd = vwai_tfl(UINT *p_tflptn, UINT waiptn);	vwai_tfl	ER ercd = vwai_tfl(UINT waiptn, UINT *p_tflptn);	
11	**タスク付属イベントフラグ待ち(ポーリング)**		**タスク付属イベントフラグ待ち(ポーリング)**		
	vpol_tfl	ER ercd = vpol_tfl(UINT *p_tflptn, UINT waiptn);	vpol_tfl	ER ercd = vpol_tfl(UINT waiptn, UINT *p_tflptn);	
12	**タスク付属イベントフラグ待ち(タイムアウト有)**		**タスク付属イベントフラグ待ち(タイムアウト有)**		
	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 タスク例外管理機能

- グレーの網掛けがかかっているところは変更になっている箇所です。
- この SVC 相違一覧表では用語が以下のように置き換えられています。
  パラメータ → 引数　　リターンパラメータ → 返値

HI7000 シリーズ（μITRON3.0 仕様準拠）			HI7000/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
−	−		タスク例外処理ルーチンの定義		新機能
	−	該当なし	def_tex	ER ercd = def_tex (ID tskid, T_DTEX *pk_dtex);	
			idef_tex	ER ercd = idef_tex (ID tskid, T_DTEX *pk_dtex);	
−	−		タスク例外処理の要求		新機能
	−	該当なし	ras_tex	ER ercd = ras_tex(ID tskid, TEXPTN rasptn);	
			iras_tex	ER ercd = iras_tex(ID tskid, TEXPTN rasptn);	
−	−		タスク例外処理の禁止		新機能
	−	該当なし	dis_tex	ER ercd = dis_tex();	
−	−		タスク例外処理の許可		新機能
	−	該当なし	ena_tex	ER ercd = ena_tex();	
−	−		タスク例外禁止状態の参照		新機能
	−	該当なし	sns_tex	BOOL state =sns_tex();	
−	−		タスク例外処理の状態参照		新機能
	−	該当なし	ref_tex	ER ercd = ref_tex(ID tskid, T_RTEX *pk_rtex);	
			iref_tex	ER ercd = iref_tex(ID tskid, T_RTEX *pk_rtex);	

4.1.4 同期･通信(セマフォ)機能

- グレーの網掛けがかかっているところは変更になっている箇所です。
- この SVC 相違一覧表では用語が以下のように置き換えられています。
  パラメータ → 引数　　リターンパラメータ → 返値

	HI7000 シリーズ（μITRON3.0 仕様準拠）		HI7000/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
1	**セマフォ生成**		**セマフォ生成**		**パケット構造　変更**
	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);	
		パケット構造 typedef struct t_csem { VP exinf; **拡張情報** ATR sematr; **セマフォ属性** INT isemcnt; **セマフォの初期値** B *name; **名称を格納した領域の先頭アドレス** } T_CSEM;		**パケット構造** typedef struct t_csem { ATR sematr; **セマフォ属性** UINT isemcnt; **セマフォの資源数の初期値** UINT maxsem; **セマフォの最大資源数** } T_CSEM;	
2	**セマフォ ID の割り当てと生成**		**セマフォ ID の割り当てと生成**		API **名、引数、返値、パケット構造変更**
	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);	
		引数 ID *p_semid; **生成したセマフォ ID を返す領域の先頭アドレス** T_CSEM *pk_csem; **セマフォ生成情報の先頭アドレス**		**引数** T_CSEM *pk_csem; **セマフォ生成情報を格納したパケットへのポインタ**	
		返値 ER ercd; **エラーコード** ID *p_semid; **生成したセマフォ ID を格納した領域の先頭アドレス**		**返値** ER_ID ercd **生成したセマフォの ID 番号（正の値）またはエラーコード**	
		パケット構造 typedef struct t_csem { VP exinf; **拡張情報** ATR sematr; **セマフォ属性** INT isemcnt; **セマフォの初期値** B *name; **名称を格納した領域の先頭アドレス** } T_CSEM;		**パケット構造** typedef struct t_csem { ATR sematr; **セマフォ属性** UINT isemcnt; **セマフォの資源数の初期値** UINT maxsem; **セマフォの最大資源数** } T_CSEM;	
3	**セマフォ削除**		**セマフォ削除**		
	del_sem	ER ercd = del_sem(ID semid);	del_sem	ER ercd = del_sem(ID semid);	
4	**セマフォ資源返却**		**セマフォ資源返却**		
	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	**セマフォ資源獲得**		**セマフォ資源獲得**		
	wai_sem	ER ercd = wai_sem(ID semid);	wai_sem	ER ercd = wai_sem(ID semid);	
6	**セマフォ資源獲得(ポーリング)**		**セマフォ資源獲得(ポーリング)**		SVC **名変更**
	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	**セマフォ資源獲得(タイムアウト有)**		**セマフォ資源獲得(タイムアウト有)**		
	twai_sem	ER ercd = twai_sem(ID semid, TMO tmout);	twai_sem	ER ercd = twai_sem(ID semid, TMO tmout);	

	HI7000 シリーズ（μITRON3.0 仕様準拠）			HI7000/4 シリーズ（μITRON4.0 仕様準拠）			変更内容
	機能			機能			
	SVC	API		SVC	API		
8	セマフォ状態参照			セマフォ状態参照			引数、パケット構造　変更
	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);		
		引数	T_RSEM *pk_rsem; セマフォ状態を返す領域の先頭アドレス ID semid; セマフォ ID		引数	ID semid; セマフォ ID T_RSEM *pk_rsem; セマフォ状態を返すパケットへのポインタ	
		パケット構造	typedef struct t_rsem { VP exinf; 拡張情報 BOOL_ID wtsk; 待ちタスク ID INT semcnt; 現在のセマフォカウント値 ATR sematr; セマフォ属性 B *name; 名称を返す領域の先頭アドレス } T_RSEM;		パケット構造	typedef struct t_rsem { ID wtskid; 待ちタスク ID UINT semcnt; 現在のセマフォカウント値 } T_RSEM;	
9	セマフォ状態参照			ー			廃止
	vsem_sts	ER ercd = vsem_sts(ID *p_wtskid, W *p_semcnt, ID semid);		ー	該当なし		

4.1.5 同期･通信(イベントフラグ)機能

- グレーの網掛けがかかっているところは変更になっている箇所です。
- この SVC 相違一覧表では用語が以下のように置き換えられています。
  パラメータ → 引数　　リターンパラメータ → 返値

HI7000 シリーズ（μITRON3.0 仕様準拠）			HI7000/4 シリーズ（μITRON4.0 仕様準拠）			変更内容
	機能			機能		
	SVC	API		SVC	API	
1	**イベントフラグ生成**			**イベントフラグ生成**		**パケット構造変更 クリア指定方法変更** 3.3.3, 3.3.4 **参照**
	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);	
		パケット構造 typedef struct t_cflg { VP exinf; **拡張情報** ATR flgatr; **イベントフラグ属性** UINT iflgptn; **イベントフラグの初期値** B *name; **名称を格納した領域の先頭アドレス** } T_CFLG;			**パケット構造** typedef struct t_cflg { ATR flgatr; **イベントフラグ属性** FLGPTN iflgptn; **イベントフラグの初期値** } T_CFLG;	
2	**イベントフラグ ID の割り当てと生成**			**イベントフラグ生成（ID 番号自動割り付け）**		SVC **名、引数、返値、パケット構造 変更 クリア指定方法変更** 3.3.3, 3.3.4 **参照**
	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);	
		引数 ID *p_flgid; **生成したイベントフラグ ID を返す領域の先頭アドレス** T_CFLG *pk_cflg; **イベントフラグ生成情報の先頭アドレス**			**引数** T_CFLG *pk_cflg **イベントフラグ生成情報を格納したパケットへのポインタ**	
		返値 ER ercd; **エラーコード** ID *p_flgid; **生成したイベントフラグ ID を格納した領域の先頭アドレス**			**返値** ER_ID flgid **生成したイベントフラグの ID 番号（正の値）またはエラーコード**	
		パケット構造 typedef struct t_cflg { VP exinf; **拡張情報** ATR flgatr; **イベントフラグ属性** UINT iflgptn; **イベントフラグの初期値** B *name; **名称を格納した領域の先頭アドレス** } T_CFLG;			**パケット構造** typedef struct t_cflg { ATR flgatr; **イベントフラグ属性** FLGPTN iflgptn; **イベントフラグの初期値** } T_CFLG;	
3	**イベントフラグ削除**			**イベントフラグ削除**		
	del_flg	ER ercd = del_flg(ID flgid);		del_flg	ER ercd = del_flg(ID flgid);	
4	**イベントフラグのセット**			**イベントフラグのセット**		**引数型変更**
	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);	
		引数 UINT setptn **セットするビットパターン**			**引数** FLGPTN setptn **セットするビットパターン**	
5	**イベントフラグのクリア**			**イベントフラグのクリア**		**引数型変更**
	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);	
		引数 UINT clrptn **クリアするビットパターン**			**引数** FLGPTN clrptn **クリアするビットパターン**	

	HI7000 シリーズ（μITRON3.0 仕様準拠）		HI7000/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
6	**イベントフラグの待ち**		**イベントフラグの待ち**		**引数型変更** **クリア指定方法変更** 3.3.3, 3.3.4 **参照**
	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);	
		引数 UINT *p_flgptn **待ち解除時のビットパターンを返す領域の先頭アドレス**		**引数** FLGPTN *p_flgptn **待ち解除時のビットパターンを返す領域へのポインタ**	
7	**イベントフラグの待ち(ポーリング)**		**イベントフラグの待ち(ポーリング)**		**引数並び、引数型変更** **クリア指定方法変更** 3.3.3, 3.3.4 **参照**
	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);	
		引数 UINT *p_flgptn **待ち解除時のビットパターンを返す領域の先頭アドレス**		**引数** FLGPTN *p_flgptn **待ち解除時のビットパターンを返す領域へのポインタ**	
8	**イベントフラグの待ち(タイムアウト有)**		**イベントフラグの待ち(タイムアウト有)**		**引数並び、引数型変更** **クリア指定方法変更** 3.3.3, 3.3.4 **参照**
	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);	
		引数 UINT *p_flgptn **待ち解除時のビットパターンを返す領域の先頭アドレス**		**引数** FLGPTN *p_flgptn **待ち解除時のビットパターンを返す領域へのポインタ**	
9	**イベントフラグ状態参照**		**イベントフラグ状態参照**		**引数並び、パケット構造　変更**
	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);	
		パケット構造 typedef struct t_rflg { VP exinf; **拡張情報** BOOL_ID wtsk; **待ちタスク** ID UINT flgptn; **イベントフラグの ビットパターン** ATR flgatr; **イベントフラグ属性** B *name; **名称を返す領域の先頭アドレス** } T_RFLG;		**パケット構造** typedef struct t_rflg { ID wtskid; **待ちタスク** ID FLGPTN flgptn; **イベントフラグの ビットパターン** } T_RFLG;	
10	**イベントフラグ状態参照**		**－**		**廃止**
	vflg_sts	ER ercd = vflg_sts(ID *p_wtskid, UW *p_flgptn, ID flgid);	**－**	**該当なし**	

4.1.6 同期･通信(データキュー)機能

- グレーの網掛けがかかっているところは変更になっている箇所です。
- この SVC 相違一覧表では用語が以下のように置き換えられています。
  パラメータ → 引数　　リターンパラメータ → 返値

HI7000 シリーズ（μITRON3.0 仕様準拠）			HI7000/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
−	−		データキューの生成		新機能
	−	該当なし	cre_dtq	ER ercd = cre_dtq(ID dtqid, T_CDTQ *pk_cdtq);	
			icre_dtq	ER ercd = icre_dtq(ID dtqid, T_CDTQ *pk_cdtq);	
−	−		データキューの生成(ID 番号自動割り付け)		新機能
	−	該当なし	acre_dtq	ER_ID dtqid = acre_dtq(T_CDTQ *pk_cdtq);	
			iacre_dtq	ER_ID dtqid = iacre_dtq(T_CDTQ *pk_cdtq);	
−	−		データキューの削除		新機能
	−	該当なし	del_dtq	ER ercd = del_dtq(ID dtqid);	
−	−		データキューへの送信		新機能
	−	該当なし	snd_dtq	ER ercd = snd_dtq(ID dtqid, VP_INT data);	
−	−		データキューへの送信(ポーリング)		新機能
	−	該当なし	psnd_dtq	ER ercd = psnd_dtq(ID dtqid, VP_INT data);	
			ipsnd_dtq	ER ercd = ipsnd_dtq(ID dtqid, VP_INT data);	
−	−		データキューへの送信(タイムアウト有)		新機能
	−	該当なし	tsnd_dtq	ER ercd = tsnd_dtq(ID dtqid, VP_INT data, TMO tmout);	
−	−		データキューへの強制送信		新機能
	−	該当なし	fsnd_dtq	ER ercd = fsnd_dtq(ID dtqid, VP_INT data);	
			ifsnd_dtq	ER ercd = ifsnd_dtq(ID dtqid, VP_INT data);	
−	−		データキューからの受信		新機能
	−	該当なし	rcv_dtq	ER ercd = rcv_dtq(ID dtqid, VP_INT *p_data);	
−	−		データキューからの受信(ポーリング)		新機能
	−	該当なし	prcv_dtq	ER ercd = prcv_dtq(ID dtqid, VP_INT *p_data);	
−	−		データキューからの受信(タイムアウト有)		新機能
	−	該当なし	trcv_dtq	ER ercd = trcv_dtq(ID dtqid, VP_INT *p_data, TMO tmout);	
−	−		データキューの状態参照		新機能
	−	該当なし	ref_dtq	ER ercd = ref_dtq(ID dtqid, T_RDTQ *pk_rdtq);	
			iref_dtq	ER ercd = iref_dtq(ID dtqid, T_RDTQ *pk_rdtq);	

4.1.7 同期･通信(メールボックス)機能

- グレーの網掛けがかかっているところは変更になっている箇所です。
- この SVC 相違一覧表では用語が以下のように置き換えられています。
  パラメータ → 引数　　リターンパラメータ → 返値

	HI7000 シリーズ（μITRON3.0 仕様準拠）		HI7000/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
1	**メイルボックス生成**		**メールボックス生成**		**パケット構造 変更** **属性追加** 3.3.5 **参照**
	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);	
		パケット構造 typedef struct t_cmbx { VP exinf; **拡張情報** ATR mbxatr; **メイルボックス属性** B *name; **名称を格納した領域の先頭アドレス** } T_CMBX;		**パケット構造** typedef struct t_cmbx { ATR mbxatr; **メールボックス属性** PRI maxmpri; **メッセージ優先度の最大値** VP mprihd; **優先度別メッセージキューヘッダの先頭アドレス** } T_CMBX;	
2	**メイルボックス ID の割り当てと生成**		**メールボックス ID の割り当てと生成**		SVC **名、引数、返値、パケット構造 変更** **属性追加** 3.3.5 **参照**
	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);	
		引数 ID *p_mbxid **生成したメイルボックス ID を返す領域の先頭アドレス** T_CMBX *pk_cmbx **メイルボックス生成情報の先頭アドレス**		**引数** T_CMBX *pk_cmbx **メールボックス生成情報を格納したパケットへのポインタ**	
		返値 ER ercd **エラーコード** ID *p_mbxid **生成したメイルボックス ID を格納した領域の先頭アドレス**		**返値** ER_ID mbxid **生成したメールボックスの ID 番号（正の値）、またはエラーコード**	
		パケット構造 typedef struct t_cmbx { VP exinf; **拡張情報** ATR mbxatr; **メイルボックス属性** B *name; **名称を格納した領域の先頭アドレス** } T_CMBX;		**パケット構造** typedef struct t_cmbx { ATR mbxatr; **メールボックス属性** PRI maxmpri; **メッセージ優先度の最大値** VP mprihd; **優先度別メッセージキューヘッダの先頭アドレス** } T_CMBX;	
3	**メイルボックス削除**		**メールボックス削除**		
	del_mbx	ER ercd = del_mbx(ID mbxid);	del_mbx	ER ercd = del_mbx(ID mbxid);	
4	**メイルボックスへ送信**		**メールボックスへ送信**		SVC **名変更** **メッセージヘッダ変更** 3.3.6 **参照**
	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);	
		パケット typedef struct t_msg { VP msghead; **カーネル管理領域** } T_MSG; **メイルボックスのメッセージヘッダ**		**パケット** typedef struct t_msg { VP msghead; **カーネル管理領域** } T_MSG; **メールボックスのメッセージヘッダ**	
				typedef struct t_msg_pri { T_MSG msgque; **メッセージヘッダ** PRI msgpri; **メッセージ優先度** } T_MSG_PRI; **メールボックスの優先度付きメッセージヘッダ**	

	HI7000 シリーズ（μITRON3.0 仕様準拠）		HI7000/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
5	**メイルボックスのメッセージ行列先頭へメッセージ送信**		**－**		**廃止** 3.1.4 参照
	vjmp_msg	ER ercd = vjmp_msg(ID mbxid, T_MSG *pk_msg);	－	**該当なし**	
6	**メイルボックスから受信**		**メールボックスから受信**		SVC **名、引数並び変更 メッセージヘッダ変更** 3.3.6 参照
	rcv_msg	ER ercd = rcv_msg(T_MSG **ppk_msg, ID mbxid);	rcv_mbx	ER ercd = rcv_mbx(ID mbxid, T_MSG **ppk_msg);	
		パケット typedef struct t_msg { VP msghead; **カーネル管理領域** } T_MSG; **メイルボックスのメッセージヘッダ**		**パケット** typedef struct t_msg { VP msghead; **カーネル管理領域** } T_MSG; **メールボックスのメッセージヘッダ**	
				typedef struct t_msg_pri { T_MSG msgque; **メッセージヘッダ** PRI msgpri; **メッセージ優先度** } T_MSG_PRI; **メールボックスの優先度付きメッセージヘッダ**	
7	**メイルボックスから受信(ポーリング)**		**メールボックスから受信(ポーリング)**		SVC **名、引数並び変更 メッセージヘッダ変更** 3.3.6 参照
	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);	
		パケット typedef struct t_msg { VP msghead; **カーネル管理領域** } T_MSG; **メイルボックスのメッセージヘッダ**		**パケット** typedef struct t_msg { VP msghead; **カーネル管理領域** } T_MSG; **メールボックスのメッセージヘッダ**	
				typedef struct t_msg_pri { T_MSG msgque; **メッセージヘッダ** PRI msgpri; **メッセージ優先度** } T_MSG_PRI; **メールボックスの優先度付きメッセージヘッダ**	
8	**メイルボックスから受信(タイムアウト有)**		**メールボックスから受信(タイムアウト有)**		SVC **名、引数並び変更 メッセージヘッダ変更** 3.3.6 参照
	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);	
		パケット typedef struct t_msg { VP msghead; **カーネル管理領域** } T_MSG; **メイルボックスのメッセージヘッダ**		**パケット** typedef struct t_msg { VP msghead; **カーネル管理領域** } T_MSG; **メールボックスのメッセージヘッダ**	
				typedef struct t_msg_pri { T_MSG msgque; **メッセージヘッダ** PRI msgpri; **メッセージ優先度** } T_MSG_PRI; **メールボックスの優先度付きメッセージヘッダ**	

	HI7000 シリーズ（μITRON3.0 仕様準拠）		HI7000/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
9	**メイルボックス状態参照**		**メールボックス状態参照**		**引数並び、パケット構造 変更メッセージヘッダ変更** 3.3.6 参照
	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);	
		パケット構造 typedef struct t_rmbx{ VP exinf; **拡張情報** BOOL_ID wtsk; **待ちタスク** ID T_MSG *pk_msg; **次に受信されるメッセージの先頭アドレス** ATR mbxatr; **メイルボックス属性** B *name; **名称を返す領域の先頭アドレス** } T_RMBX;		**パケット構造** typedef struct t_rmbx{ ID wtskid; **待ちタスク** ID T_MSG *pk_msg; **次に受信されるメッセージの先頭アドレス** } T_RMBX;	
		typedef struct t_msg { VP msghead; **カーネル管理領域** } T_MSG; **メイルボックスのメッセージヘッダ**		typedef struct t_msg { VP msghead; **カーネル管理領域** } T_MSG; **メールボックスのメッセージヘッダ**	
				typedef struct t_msg_pri { T_MSG msgque; **メッセージヘッダ** PRI msgpri; **メッセージ優先度** } T_MSG_PRI; **メールボックスの優先度付きメッセージヘッダ**	
10	**メイルボックス状態参照**		**－**		**廃止**
	vmbx_sts	ER ercd = vmbx_sts(ID *p_wtskid, T_MSG **ppk_msg, ID mbxid);	**－**	**該当なし**	

4.1.8 拡張同期･通信(ミューテックス)機能

HI7000 シリーズ（μITRON3.0 仕様準拠）			HI7000/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
−	−		ミューテックスの生成		新機能
	−	該当なし	cre_mtx	ER ercd = cre_mtx(ID mtxid, T_CMTX *pk_cmtx);	
−	−		ミューテックスの生成(ID 番号自動割り付け)		新機能
	−	該当なし	acre_mtx	ER_ID mtxid = acre_mtx(T_CMTX *pk_cmtx);	
−	−		ミューテックスの削除		新機能
	−	該当なし	del_mtx	ER ercd = del_mtx(ID mtxid);	
−	−		ミューテックスのロック		新機能
	−	該当なし	loc_mtx	ER ercd = loc_mtx(ID mtxid);	
−	−		同上(ポーリング)		新機能
	−	該当なし	ploc_mtx	ER ercd = ploc_mtx(ID mtxid);	
−	−		同上(タイムアウト有)		新機能
	−	該当なし	tloc_mtx	ER ercd = tloc_mtx(ID mtxid, TMO tmout);	
−	−		ミューテックスのロック解除		新機能
	−	該当なし	unl_mtx	ER ercd = unl_mtx(ID mtxid);	
−	−		ミューテックスの状態参照		新機能
	−	該当なし	ref_mtx	ER ercd = ref_mtx(ID mtxid, T_RMTX *pk_rmtx);	

4.1.9 同期･通信(メッセージバッファ)機能

- グレーの網掛けがかかっているところは変更になっている箇所です。
- この SVC 相違一覧表では用語が以下のように置き換えられています。
  パラメータ → 引数　　リターンパラメータ → 返値

HI7000 シリーズ（μITRON3.0 仕様準拠）		HI7000/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
機能		機能		
SVC	API	SVC	API	
1 メッセージバッファ生成		メッセージバッファ生成		パケット構造　変更 待ち行列動作変更 3.3.7 参照
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);	
	パケット構造 typedef struct t_cmbf { VP exinf; 拡張情報 ATR mbfatr; メッセージバッファ属性 INT bufsz; メッセージバッファのサイズ（バイト数） INT maxmsz; メッセージの最大長（バイト数） B *name; 名称を格納した領域の先頭アドレス } T_CMBF;		パケット構造 typedef struct t_cmbf { ATR mbfatr; メッセージバッファ属性 UINT maxmsz; メッセージの最大サイズ（バイト数） SIZE mbfsz; メッセージバッファ領域のサイズ（バイト数） VP mbf; メッセージバッファ領域の先頭アドレス } T_CMBF;	
2 メッセージバッファ ID の割り当てと生成		メッセージバッファ ID の割り当てと生成		SVC 名、引数、返値、パケット構造　変更 待ち行列動作変更 3.3.7 参照
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);	
	引数 ID *p_mbfid 生成したメッセージバッファ ID を返す領域の先頭アドレス T_CMBF *pk_cmbf メッセージバッファ生成情報の先頭アドレス		引数 T_CMBF *pk_cmbf メッセージバッファ生成情報を格納したパケットへのポインタ	
	返値 ER ercd エラーコード ID *p_mbfid 生成したメッセージバッファ ID を格納した領域の先頭アドレス		返値 ER_ID mbfid 生成したメッセージバッファの ID 番号（正の値）、またはエラーコード	
	パケット構造 typedef struct t_cmbf { VP exinf; 拡張情報 ATR mbfatr; メッセージバッファ属性 INT bufsz; メッセージバッファのサイズ（バイト数） INT maxmsz; メッセージの最大長（バイト数） B *name; 名称を格納した領域の先頭アドレス } T_CMBF;		パケット構造 typedef struct t_cmbf { ATR mbfatr; メッセージバッファ属性 UINT maxmsz; メッセージの最大サイズ（バイト数） SIZE mbfsz; メッセージバッファ領域のサイズ（バイト数） VP mbf; メッセージバッファ領域の先頭アドレス } T_CMBF;	
3 メッセージバッファ削除		メッセージバッファ削除		
del_mbf	ER ercd = del_mbf(ID mbfid);	del_mbf	ER ercd = del_mbf(ID mbfid);	
4 メッセージバッファへ送信		メッセージバッファへ送信		引数型変更 待ち行列動作変更 3.3.7 参照
snd_mbf	ER ercd = snd_mbf(ID mbfid, VP msg, INT msgsz);	snd_mbf	ER ercd = snd_mbf(ID mbfid, VP msg, UINT msgsz);	
	引数 INT msgsz; 送信メッセージのサイズ(バイト数)		引数 UINT msgsz; 送信メッセージのサイズ(バイト数)	

	HI7000 シリーズ（μITRON3.0 仕様準拠）					HI7000/4 シリーズ（μITRON4.0 仕様準拠）					変更内容
	機能					機能					
	SVC	API				SVC	API				
5	**メッセージバッファへ送信(ポーリング)**					**メッセージバッファへ送信(ポーリング)**					**引数型変更** **待ち行列動作変更** 3.3.7 **参照**
	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);				
		引数	INT	msgsz;	**送信メッセージのサイズ(バイト数)**		**引数**	UINT	msgsz;	**送信メッセージのサイズ(バイト数)**	
6	**メッセージバッファへ送信(タイムアウト有)**					**メッセージバッファへ送信(タイムアウト有)**					**引数型変更** **待ち行列動作変更** 3.3.7 **参照**
	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);				
		引数	INT	msgsz;	**送信メッセージのサイズ(バイト数)**		**引数**	UINT	msgsz;	**送信メッセージのサイズ(バイト数)**	
7	**メッセージバッファから受信**					**メッセージバッファから受信**					**引数、返値変更** **待ち行列動作変更** 3.3.7 **参照**
	rcv_mbf	ER ercd = rcv_mbf(VP msg, INT *p_msgsz, ID mbfid);				rcv_mbf	ER_UINT msgsz = rcv_mbf(ID mbfid, VP msg);				
		引数	VP	msg	**送信メッセージを返す領域の先頭アドレス**		**引数**	ID	mbfid	**メッセージバッファ** ID	
			INT	*p_msgsz	**受信したメッセージのサイズを返す領域の先頭アドレス**（**バイト数**）			VP	msg	**送信メッセージを返す領域への先頭アドレス**	
			ID	mbfid	**メッセージバッファ** ID						
		返値	ER	ercd	**エラーコード**		**返値**	ER_UINT	msgsz	**受信メッセージのサイズ（バイト数、正の値）またはエラーコード**	
			VP	msg	**送信メッセージを格納した領域の先頭アドレス**			VP	msg	**送信メッセージを格納した領域の先頭アドレス**	
			INT	*p_msgsz	**受信したメッセージのサイズを格納した領域の先頭アドレス(バイト数)**						
8	**メッセージバッファから受信(ポーリング)**					**メッセージバッファから受信(ポーリング)**					**引数、返値変更** **非タスクから発行不可** **待ち行列動作変更** 3.3.7 **参照**
	prcv_mbf	ER ercd = prcv_mbf(VP msg, INT *p_msgsz, ID mbfid);				prcv_mbf	ER_UINT msgsz = prcv_mbf(ID mbfid, VP msg);				
		引数	VP	msg	**送信メッセージを返す領域の先頭アドレス**		**引数**	ID	mbfid	**メッセージバッファ** ID	
			INT	*p_msgsz	**受信したメッセージのサイズを返す領域の先頭アドレス**（**バイト数**）			VP	msg	**送信メッセージを返す領域への先頭アドレス**	
			ID	mbfid	**メッセージバッファ** ID						
		返値	ER	ercd	**エラーコード**		**返値**	ER_UINT	msgsz	**受信メッセージのサイズ（バイト数、正の値）またはエラーコード**	
			VP	msg	**送信メッセージを格納した領域の先頭アドレス**			VP	msg	**送信メッセージを格納した領域の先頭アドレス**	
			INT	*p_msgsz	**受信したメッセージのサイズを格納した領域の先頭アドレス(バイト数)**						
		動作	**タスク独立部からも発行可能**				**動作**	**非タスクコンテキストから発行不可**			

	HI7000 シリーズ（μITRON3.0 仕様準拠）				HI7000/4 シリーズ（μITRON4.0 仕様準拠）				変更内容
	機能				機能				
	SVC	API			SVC	API			
9	**メッセージバッファから受信(タイムアウト有)**				**メッセージバッファから受信(タイムアウト有)**				**引数、返値変更 待ち行列動作変更** 3.3.7 **参照**
	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);			
		引数	VP	msg	**送信メッセージを返す領域の先頭アドレス**	**引数**	ID	mbfid	**メッセージバッファ** ID
			INT	*p_msgsz	**受信したメッセージのサイズを返す領域の先頭アドレス（バイト数)**		VP	msg	**送信メッセージを返す領域への先頭アドレス**
			ID	mbfid	**メッセージバッファ** ID		TMO	tmout	**タイムアウト指定**
			TMO	tmout	**タイムアウト指定**				
		返値	ER	ercd	**エラーコード**	**返値**	ER_UINT	msgsz	**受信メッセージのサイズ（バイト数、正の値）またはエラーコード**
			VP	msg	**送信メッセージを格納した領域の先頭アドレス**		VP	msg	**送信メッセージを格納した領域の先頭アドレス**
			INT	*p_msgsz	**受信したメッセージのサイズを格納した領域の先頭アドレス(バイト数)**				
10	**メッセージバッファ状態参照**				**メッセージバッファ状態参照**				**引数並び、パケット構造 変更待ち行列動作変更** 3.3.7 **参照**
	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);			
		typedef struct t_rmbf {				typedef struct t_rmbf {			
		VP	exinf;	**拡張情報**		ID	stskid;	**送信待ち行列の先頭タスク** ID	
		BOOL_ID	wtsk;	**受信待ちタスク** ID		ID	rtskid;	**受信待ち行列の先頭タスク** ID	
		BOOL_ID	stsk;	**送信待ちタスク** ID		UINT	smsgcnt;	**メッセージバッファに入っているメッセージの数**	
		INT	msgsz;	**次に受信されるメッセージのサイズ（バイト数）**		SIZE	fmbfsz;	**空きバッファのサイズ（バイト数）**	
		INT	frbufsz;	**空きバッファのサイズ（バイト数）**		} T_RMBF;			
		ATR	mbfatr;	**メッセージバッファ属性**					
		INT	bufsz;	**メッセージバッファのサイズ（バイト数）**					
		INT	maxmsz;	**メッセージの最大長（バイト数）**					
		B	*name;	**名称を返す領域の先頭アドレス**					
		} T_RMBF;							

4.1.10 割り込み管理機能

- グレーの網掛けがかかっているところは変更になっている箇所です。
- この SVC 相違一覧表では用語が以下のように置き換えられています。
  パラメータ → 引数　　リターンパラメータ → 返値

	HI7000 シリーズ（μITRON3.0 仕様準拠）		HI7000/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
1	割り込みハンドラ定義		割り込みハンドラ定義		SVC 名、パケット名変更
	def_int	ER ercd = def_int(UINT dintno, T_DINT *pk_dint); パケット typedef struct t_dint { ATR intatr; ハンドラ属性 FP inthdr; ハンドラアドレス UINT intsr; 起動時の SR(HI7000S/3 では無視されます) } T_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); パケット typedef struct t_dinh { ATR inhatr; ハンドラ属性 FP inthdr; ハンドラアドレス UINT inhsr; 起動時の SR(HI7000/4 では無視されます) } T_DINH;	
2	トラップ例外処理ルーチン定義[HI7700, HI7750]		CPU 例外ハンドラ（TRAP 例外）		SVC 名、パケット名変更
	vdef_trp	ER ercd = vdef_trp(UINT dtrpno, T_DTRP *pk_dtrp); パケット typedef struct t_dtrp { ATR trpatr; トラップ例外処理ルーチン属性 FP trphdr; トラップ例外処理ルーチンアドレス UINT trpsr; トラップ例外処理ルーチン起動時 SR } T_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); パケット typedef struct t_dtrp { ATR trpatr; CPU 例外(TRAPA 命令例外)ハンドラ属性 FP trphdr; CPU 例外(TRAPA 命令例外)ハンドラアドレス UINT trpsr; CPU 例外(TRAPA 命令例外)ハンドラ起動時 SR } T_DTRP;	
―	―		CPU 例外ハンドラの定義		新機能
	―	該当なし	def_exc idef_exc	ER ercd = def_exc(EXCNO excno, T_DEXC *pk_dexc); ER ercd = idef_exc(EXCNO excno, T_DEXC *pk_dexc);	
3	割り込みハンドラから復帰		割り込みハンドラから復帰		
	ret_int	[HI7000] "#pragma interrupt"文により、割り込みハンドラ関数の最後で自動的に発行されます [HI7700, HI7750] 割り込みハンドラ関数からのリターンで自動的に発行されます	―	[HI7000/4 ダイレクト割り込みハンドラ] "#pragma interrupt"文により、割り込みハンドラ関数の最後で自動的で復帰します [HI7000/4 シリーズ 通常割り込みハンドラ] 割り込みハンドラ関数からのリターンで自動的で復帰します	
4	例外処理から復帰		例外処理から復帰		
	vret_exc	[HI7000] "#pragma interrupt"文により、例外処理ルーチン関数の最後で自動的に発行されます [HI7700, HI7750] 例外処理ルーチン関数からのリターンで自動的に発行されます	―	CPU 例外ハンドラからのリターンで自動的で復帰します	
5	割り込みとディスパッチの禁止		CPU ロック状態への移行		CPU ロックの意味 変更 3.3.1 参照
	loc_cpu	ER ercd = loc_cpu(void);	loc_cpu iloc_cpu	ER ercd = loc_cpu(); ER ercd = iloc_cpu();	
6	割り込みとディスパッチの許可		CPU ロック状態の解除		CPU ロックの意味 変更 3.3.1 参照
	unl_cpu	ER ercd = unl_cpu(void);	unl_cpu iunl_cpu	ER ercd = unl_cpu(); ER ercd = iunl_cpu();	

	HI7000 シリーズ（μITRON3.0 仕様準拠）			HI7000/4 シリーズ（μITRON4.0 仕様準拠）			変更内容
	機能			機能			
	SVC	API		SVC	API		
7	割り込みマスクレベル変更			割り込みマスクレベル変更			
	chg_ims	ER ercd = chg_ims(UINT imask);		chg_ims	ER ercd = chg_ims(IMASK imask);		
				ichg_ims	ER ercd = ichg_ims(IMASK imask);		
8	割り込みマスクレベル参照			割り込みマスクレベル参照			SVC 名、引数型変更
	ref_ims	ER ercd = ref_ims(UINT *p_imask);		get_ims	ER ercd = get_ims(IMASK *p_imask);		
				iget_ims	ER ercd = iget_ims(IMASK *p_imask);		
		引数	UINT *p_imask; 割り込みマスクレベルを返す領域の先頭アドレス		引数	IMASK *p_imask; 割り込みマスクレベルを返す領域の先頭アドレス	

4.1.11 メモリプール管理(固定長メモリプール)機能

- グレーの網掛けがかかっているところは変更になっている箇所です。
- この SVC 相違一覧表では用語が以下のように置き換えられています。
  パラメータ → 引数　　リターンパラメータ → 返値

	HI7000 シリーズ（μITRON3.0 仕様準拠）					HI7000/4 シリーズ（μITRON4.0 仕様準拠）					変更内容
	機能					機能					
	SVC	API				SVC	API				
1	固定長メモリプール生成					固定長メモリプール生成					パケット構造変更
	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);				
		パケット構造	typedef struct t_cmpf {				パケット構造	typedef struct t_cmpf {			
			VP	exinf;	拡張情報			ATR	mpfatr;	固定長メモリプール属性	
			ATR	mpfatr;	固定長メモリプール属性			UINT	blkcnt;	メモリプール全体のブロック数	
			INT	mpfcnt;	メモリプール全体のブロック数			UINT	blksz;	固定長メモリブロックサイズ（バイト数）	
			INT	blfsz;	固定長メモリブロックサイズ（バイト数）			VP	mpf;	固定長メモリプール領域の先頭アドレス	
			B	*name;	名称を格納した領域の先頭アドレス			VP	mpfmb;	固定長メモリブロック管理領域の先頭アドレス (HI7000/4 で CFG_MPFMANAGE をチェックした場合のみある項目)	
			} T_CMPF;					} T_CMPF;			
2	固定長メモリプール ID の割り当てと生成					固定長メモリプール ID の割り当てと生成					SVC 名、引数、返値、パケット構造　変更
	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);				
		引数	ID	*p_mpfid	生成した固定長メモリプール ID を返す領域の先頭アドレス		引数	T_CMPF	*pk_cmpf	固定長メモリプール生成情報を格納したパケットへのポインタ	
			T_CMPF	*pk_cmpf	固定長メモリプール生成情報の先頭アドレス						
		返値	ER	ercd	エラーコード		返値	ER_ID	mpfid	生成した固定長メモリプールの ID 番号（正の値）またはエラーコード	
			ID	*p_mpfid	生成した固定長メモリプール ID を格納した領域の先頭アドレス						
3	固定長メモリプール削除					固定長メモリプール削除					
	del_mpf	ER ercd = del_mpf(ID mpfid);				del_mpf	ER ercd = del_mpf(ID mpfid);				
4	固定長メモリブロック獲得					固定長メモリブロック獲得					SVC 名、引数並び　変更
	get_blf	ER ercd = get_blf(VP *p_blf, ID mpfid);				get_mpf	ER ercd = get_mpf(ID mpfid, VP *p_blk);				
5	固定長メモリブロック獲得(ポーリング)					固定長メモリブロック獲得(ポーリング)					SVC 名、引数並び　変更
	pget_blf	ER ercd = pget_blf(VP *p_blf, ID mpfid);				pget_mpf	ER ercd = pget_mpf(ID mpfid, VP *p_blk);				
						ipget_mpf	ER ercd = ipget_mpf(ID mpfid, VP *p_blk);				
6	固定長メモリブロック獲得(タイムアウト有)					固定長メモリブロック獲得(タイムアウト有)					SVC 名、引数並び　変更
	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	固定長メモリブロック返却					固定長メモリブロック返却					SVC 名変更
	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 シリーズ（μITRON3.0 仕様準拠）		HI7000/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
8	固定長メモリプール状態参照		固定長メモリプール状態参照		引数並び、パケット構造変更
	ref_mpf	ER ercd = ref_mpf(T_RMPF *pk_rmpf, ID mpfid);	ref_mpf iref_mpf	ER ercd = ref_mpf(ID mpfid, T_RMPF *pk_rmpf); ER ercd = iref_mpf(ID mpfid, T_RMPF *pk_rmpf);	
		パケット構造 typedef struct t_rmpf { VP exinf; 拡張情報 BOOL_ID wtsk; 待ちタスク ID INT frbcnt; 空き領域のブロック数 ATR mpfatr; 固定長メモリプール属性 INT mpfcnt; メモリプール全体のブロック数 INT blfsz; 固定長メモリブロックサイズ（バイト数） B *name; 名称を返す領域の先頭アドレス } T_RMPF;		パケット構造 typedef struct t_rmpf { ID wtskid; 待ちタスク ID UINT fblkcnt; 空き領域のブロック数 } T_RMPF;	
9	固定長メモリプール状態参照		－		廃止
	vmpf_sts	ER ercd = vmpf_sts(ID *p_wtskid, W *p_frbcnt, ID mpfid);	－	該当なし	

4.1.12 メモリプール管理(可変長メモリプール)機能

- グレーの網掛けがかかっているところは変更になっている箇所です。
- この SVC 相違一覧表では用語が以下のように置き換えられています。
  パラメータ → 引数　　リターンパラメータ → 返値

	HI7000 シリーズ（μITRON3.0 仕様準拠）		HI7000/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
1	**可変長メモリプール生成**		**可変長メモリプール生成**		**パケット構造　変更**
	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);	
		パケット構造 typedef struct t_cmpl { VP exinf; **拡張情報** ATR mplatr; **可変長メモリプール属性** INT mplsz; **メモリプール全体のサイズ（バイト数）** B *name; **名称を格納した領域の先頭アドレス** } T_CMPL;		**パケット構造** typedef struct t_cmpl { ATR mplatr; **可変長メモリプール属性** SIZE mplsz; **メモリプール全体のサイズ（バイト数）** VP mpl; **可変長メモリプール領域の先頭アドレス** } T_CMPL;	
2	**可変長メモリプール ID の割り当てと生成**		**可変長メモリプール ID の割り当てと生成**		SVC **名、引数、返値、パケット構造　変更**
	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);	
		引数 ID *p_mplid **生成した可変長メモリプール ID を返す領域の先頭アドレス** T_CMPL *pk_cmpl **可変長メモリプール生成情報の先頭アドレス**		**引数** T_CMPL *pk_cmpl **可変長メモリプール生成情報を格納したパケットへのポインタ**	
		返値 ER ercd **エラーコード** ID *p_mplid **生成した可変長メモリプール ID を格納した領域の先頭アドレス**		**返値** ER ercd **エラーコード** ID *p_mplid **生成した可変長メモリプール ID を格納した領域の先頭アドレス**	
3	**可変長メモリプール削除**		**可変長メモリプール削除**		
	del_mpl	ER ercd = del_mpl(ID mplid);	del_mpl	ER ercd = del_mpl(ID mplid);	
4	**可変長メモリブロック獲得**		**可変長メモリブロック獲得**		SVC **名、引数並び　変更**
	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	**可変長メモリブロック獲得(ポーリング)**		**可変長メモリブロック獲得(ポーリング)**		SVC **名、引数並び　変更**
	pget_blk	ER ercd = pget_blk(VP *p_blk, ID mplid, INT blksz);	pget_mpl	ER ercd = pget_mpl(ID mplid, UINT blksz, VP *p_blk);	
			ipget_mpl	ER ercd = ipget_mpl(ID mplid, UINT blksz, VP *p_blk);	
6	**可変長メモリブロック獲得(タイムアウト有)**		**可変長メモリブロック獲得(タイムアウト有)**		SVC **名、引数並び　変更**
	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	**可変長メモリブロック返却**		**可変長メモリブロック返却**		SVC **名変更**
	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 シリーズ（μITRON3.0 仕様準拠）			HI7000/4 シリーズ（μITRON4.0 仕様準拠）			変更内容
	機能			機能			
	SVC	API		SVC	API		
8	可変長メモリプール状態参照			可変長メモリプール状態参照			引数並び、パケット構造　変更
	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);		
		パケット構造	typedef struct t_rmpl {		パケット構造	typedef struct t_rmpl {	
			VP exinf; 拡張情報			ID wtskid; 待ちタスク ID	
			BOOL_ID wtsk; 待ちタスク ID			SIZE fmplsz; 空き領域の合計サイズ（バイト数）	
			INT frsz; 空き領域の合計サイズ（バイト数）			UINT fblksz; 獲得可能な最大メモリブロックサイズ（バイト数）	
			INT maxsz; 最大の空き領域のサイズ（バイト数）			} T_RMPL;	
			ATR mplatr; 可変長メモリプール属性				
			INT mplsz; メモリプール全体のサイズ（バイト数）				
			B *name; 名称を返す領域の先頭アドレス				
			} T_RMPL;				

4.1.13 時間管理機能

- グレーの網掛けがかかっているところは変更になっている箇所です。
- この SVC 相違一覧表では用語が以下のように置き換えられています。
  パラメータ → 引数　　リターンパラメータ → 返値

	HI7000 シリーズ（μITRON3.0 仕様準拠）		HI7000/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
1	**システムクロック設定**		**システムクロック設定**		**時間管理変更** 3.3.10 **参照**
	set_tim	ER ercd = set_tim(SYSTIME *pk_tim);	set_tim	ER ercd = set_tim(SYSTIM *p_systim);	
			iset_tim	ER ercd = iset_tim(SYSTIM *p_systim);	
2	**システムクロック参照**		**システムクロック参照**		**時間管理変更** 3.3.10 **参照**
	get_tim	ER ercd = get_tim(SYSTIME *pk_tim);	get_tim	ER ercd = get_tim(SYSTIM *p_systim);	
			iget_tim	ER ercd = iget_tim(SYSTIM *p_systim);	
3	**タスク遅延**		**タスク遅延**		**時間管理変更** 3.3.10 **参照**
	dly_tsk	ER ercd = dly_tsk(DLYTIME dlytim);	dly_tsk	ER ercd = dly_tsk(RELTIM dlytim);	
4	**周期起動ハンドラ定義**		**周期ハンドラ定義**		**全般に機能変更** 3.3.10, 3.3.11 **参照**
	def_cyc	ER ercd = def_cyc(HNO cycno, T_DCYC *pk_dcyc);	cre_cyc	ER ercd = cre_cyc(ID cycid, T_CCYC *pk_ccyc);	
			icre_cyc	ER ercd = icre_cyc(ID cycid, T_CCYC *pk_ccyc);	
5	**周期起動ハンドラ指定番号の割り当てと定義**		**周期ハンドラ定義（ID 番号自動割り付け）**		
	vasn_cyc	ER ercd = vasn_cyc(HNO *p_cycno, T_DCYC *pk_dcyc);	acre_cyc	ER_ID cycid = acre_cyc(T_CCYC *pk_ccyc);	
			iacre_cyc	ER_ID cycid = iacre_cyc(T_CCYC *pk_ccyc);	
−	−		**周期起動ハンドラ削除**		
	−	**該当なし**	del_cyc	ER ercd = del_cyc(ID cycid);	
6	**周期起動ハンドラ活性制御**		**周期ハンドラ動作開始**		
	act_cyc	ER ercd = act_cyc(HNO cycno, UINT cycact);	sta_cyc	ER ercd = sta_cyc(ID cycid);	
			ista_cyc	ER ercd = ista_cyc(ID cycid);	
			周期ハンドラ動作停止		
			stp_cyc	ER ercd = stp_cyc(ID cycid);	
			istp_cyc	ER ercd = istp_cyc(ID cycid);	
7	**周期起動ハンドラ状態参照**		**周期ハンドラ状態参照**		
	ref_cyc	ER ercd = ref_cyc(T_RCYC *pk_rcyc, HNO cycno);	ref_cyc	ER ercd = ref_cyc(ID cycid, T_RCYC *pk_rcyc);	
			iref_cyc	ER ercd = iref_cyc(ID cycid, T_RCYC *pk_rcyc);	
8	**アラームハンドラ定義**		**アラームハンドラ定義**		**全般に機能変更** 3.3.10, 3.3.12 **参照**
	def_alm	ER ercd = def_alm(HNO almno, T_DALM *pk_dalm);	cre_alm	ER ercd = cre_alm(ID almid, T_CALM *pk_calm);	
			icre_alm	ER ercd = icre_alm(ID almid, T_CALM *pk_calm);	
9	**アラームハンドラ指定番号の割り当てと定義**		**アラームハンドラ定義（ID 番号自動割り付け）**		
	vasn_alm	ER ercd = vasn_alm(HNO *p_almno, T_DALM *pk_dalm);	acre_alm	ER_ID almid = acre_alm(T_CALM *pk_calm);	
			iacre_alm	ER_ID almid = iacre_alm(T_CALM *pk_calm);	
−	−		**アラームハンドラの削除**		
	−	**該当なし**	del_alm	ER ercd = del_alm(ID almid);	
−	−		**アラームハンドラの動作開始**		
	−	**該当なし**	sta_alm	ER ercd = sta_alm(ID almid, RELTIM almtim);	
			ista_alm	ER ercd = ista_alm(ID almid, RELTIM almtim);	
−	−		**アラームハンドラの動作停止**		
	−	**該当なし**	stp_alm	ER ercd = stp_alm(ID almid);	
			istp_alm	ER ercd = istp_alm(ID almid);	
10	**アラームハンドラ状態参照**		**アラームハンドラ状態参照**		
	ref_alm	ER ercd = ref_alm(T_RALM *pk_ralm, HNO almno);	ref_alm	ER ercd = ref_alm(ID almid, T_RALM *pk_ralm);	
			iref_alm	ER ercd = iref_alm(ID almid, T_RALM *pk_ralm);	

	HI7000 シリーズ（μITRON3.0 仕様準拠）		HI7000/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
11	タイマハンドラから復帰		周期ハンドラ、アラームハンドラから復帰		時間管理変更 3.3.10 参照
	ret_tmr	タイマハンドラ関数からのリターンで自動的に発行されます	－	自動的に発行されます	
12	自タスクのタイムスライス時間変更		－		オーバーランハンドラで同等の動作可能 3.1.1 参照
	vchg_qua	ER ercd = vchg_qua(ID tskid, TMO quantum);	－	該当なし	
13	システムクロック更新		システムクロック更新		タイマドライバ変更 3.3.9 参照
	vsys_clk	void vsys_clk(void);	isig_tim	CFG_TIMUSE を選択することで自動的に組み込まれます。	

4.1.14 時間管理機能(オーバーランハンドラ)

	HI7000 シリーズ（μITRON3.0 仕様準拠）		HI7000/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
－	－		オーバーランハンドラの定義		新機能
	－	該当なし	def_ovr	ER ercd = def_ovr(T_DOVR *pk_dovr);	
－	－		オーバーランハンドラの動作開始		新機能
	－	該当なし	sta_ovr	ER ercd = sta_ovr(ID tskid, OVRTIM ovrtim);	
			ista_ovr	ER ercd = ista_ovr(ID tskid, OVRTIM ovrtim);	
－	－		オーバーランハンドラの動作停止		新機能
	－	該当なし	stp_ovr	ER ercd = stp_ovr(ID tskid);	
			istp_ovr	ER ercd = istp_ovr(ID tskid);	
－	－		オーバーランハンドラの状態参照		新機能
	－	該当なし	ref_ovr	ER ercd = ref_ovr(ID tskid, T_ROVR *pk_rovr);	
			iref_ovr	ER ercd = iref_ovr(ID tskid, T_ROVR *pk_rovr);	

4.1.15 キャッシュサポート機能[HI7700, HI7750]

(HI7700/4: SH-3, SH3-DSP 用)　変更
(HI7750/4:SH-4 用)　変更
(HI7700/4:SH4AL-DSP, HI7750/4:SH-4A 用)が新規

- グレーの網掛けがかかっているところは変更になっている箇所です。
- この SVC 相違一覧表では用語が以下のように置き換えられています。
  パラメータ　→　引数　　リターンパラメータ　→　返値

項番	3.0	マイコン	4.0	マイコン	操作対象キャッシュ(HI7000/4)	
1	HI7700	SH-3, SH3-DSP	HI7700/4	SH-3, SH3-DSP	命令･オペランド混在キャッシュ	変更
2				SH4AL-DSP	命令キャッシュ、オペランドキャッシュ	対応デバイス追加
3	HI7750	SH-4	HI7750/4	SH-4	オペランドキャッシュ	変更
4				SH-4A	命令キャッシュ、オペランドキャッシュ	対応デバイス追加

	HI7000 シリーズ（μITRON3.0 仕様準拠）		HI7000/4 シリーズ（μITRON4.0 仕様準拠）			変更内容
	機能		機能			
	SVC	API	SVC	API		
1	キャッシュの初期化		キャッシュの初期化			引数追加
	vini_cac	void vini_cac(UW ccr_data);	[HI7700/4:SH-3, SH3-DSP 用]			
			vini_cac ivini_cac	void vini_cac(UW ccr_data, UW entnum, UW waynum); void ivini_cac(UW ccr_data, UW entnum, UW waynum);		
				引数	UW entnum; キャッシュのエントリ数 UW waynum; キャッシュのウェイ数	
			[HI7750/4:SH-4 用]			
			vini_cac ivini_cac	void vini_cac(UW ccr_data); void vini_cac(UW ccr_data);		
			[HI7700/4:SH4AL-DSP 用, HI7750/4:SH-4A 用]			対応デバイス追加
			vini_cac ivini_cac	ER vini_cac(ATR cacatr); ER ivini_cac(ATR cacatr);		
				引数	ATR cacatr; キャッシュ属性	
2	キャッシュクリア		キャッシュクリア			
	vclr_cac	ER ercd = vclr_cac(VP clradr1, VP clradr2);	[HI7700/4:SH-3, SH3-DSP 用]			
			vclr_cac ivclr_cac	ER ercd = vclr_cac(VP clradr1, VP clradr2); ER ercd = ivclr_cac(VP clradr1, VP clradr2);		
			[HI7750/4:SH-4 用]			オペランドキャッシュのクリア
			vclr_cac ivclr_cac	ER ercd = vclr_cac(VP clradr1, VP clradr2); ER ercd = ivclr_cac(VP clradr1, VP clradr2);		
			[HI7700/4:SH4AL-DSP 用, HI7750/4:SH-4A 用]			対応デバイス追加 命令・オペランドキャッシュのクリア
			vclr_cac ivclr_cac	ER ercd = vclr_cac(VP clradr1, VP clradr2, MODE mode); ER ercd = ivclr_cac(VP clradr1, VP clradr2, MODE mode);		
				引数	MODE mode; 対象キャッシュ指定	

	HI7000 シリーズ（μITRON3.0 仕様準拠）		HI7000/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
3	キャッシュフラッシュ		キャッシュフラッシュ		
	vfls_cac	ER ercd = vfls_cac(VP flsadr1, VP flsadr2);	[HI7700/4:SH-3, SH3-DSP 用]		
			vfls_cac	ER ercd = vfls_cac(VP flsadr1, VP flsadr2);	
			ivfls_cac	ER ercd = ivfls_cac(VP flsadr1, VP flsadr2);	
			[HI7750/4:SH-4 用]		オペランドキャッシュのフラッシュ
			vfls_cac	ER ercd = vfls_cac(VP flsadr1, VP flsadr2);	
			ivfls_cac	ER ercd = ivfls_cac(VP flsadr1, VP flsadr2);	
			[HI7700/4:SH4AL-DSP 用, HI7750/4:SH-4A 用]		対応デバイス追加 オペランドキャッシュのクリア
			vfls_cac	ER ercd = vfls_cac(VP flsadr1, VP flsadr2);	
			ivfls_cac	ER ercd = ivfls_cac(VP flsadr1, VP flsadr2);	
4	キャッシュの無効化		キャッシュの無効化		
	vinv_cac	[HI7700] ER ercd = vinv_cac(void);	[HI7700/4:SH-3, SH3-DSP 用]		
			vinv_cac	ER ercd = vinv_cac(void);	
			ivinv_cac	ER ercd = ivinv_cac(void);	
		[HI7750] ER ercd = vinv_cac(VP invadr1, VP invadr2);	[HI7750/4:SH-4 用]		オペランドキャッシュの無効化
			vinv_cac	ER ercd = vinv_cac(VP invadr1, VP invadr2);	
			ivinv_cac	ER ercd = ivinv_cac(VP invadr1, VP invadr2);	
			[HI7700/4:SH4AL-DSP 用, HI7750/4:SH-4A 用]		対応デバイス追加 命令・オペランドキャッシュの無効化
			vinv_cac ivinv_cac	ER ercd = vinv_cac(VP invadr1, VP invadr2, MODE mode); ER ercd = ivinv_cac(VP invadr1, VP invadr2, MODE mode);	
			引数	MODE mode; 対象キャッシュ指定	

4.1.16 システム管理機能

- グレーの網掛けがかかっているところは変更になっている箇所です。
- この SVC 相違一覧表では用語が以下のように置き換えられています。
  パラメータ → 引数　　リターンパラメータ → 返値

	HI7000 シリーズ（μITRON3.0 仕様準拠）		HI7000/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
1	バージョン参照		バージョン参照		SVC 名、パケット構造　変更
	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);	
		パケット構造 typedef struct t_ver { UH maker; メーカー UH id; 形式番号 UH spver; 仕様書バージョン UH prver; 製品バージョン UH prno[4]; 製品管理情報 UH cpu; CPU 情報 UH var; バリエーション記述子 } T_VER;		パケット構造 typedef struct t_rver { UH maker; メーカー UH prid; 形式番号 UH spver; 仕様書バージョン UH prver; 製品バージョン UH prno[4]; 製品管理情報 } T_RVER;	
2	システム状態参照		−		廃止
	ref_sys	ER ercd = ref_sys(T_RSYS *pk_rsys);	−	該当なし	
3	コンフィグレーション情報参照		コンフィグレーション情報参照		パケット構造　変更
	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);	
		パケット構造 typedef struct t_rcfg { UW knlmsklvl; カーネル割り込みマスクレベル UH timintvl; システムクロック周期時間 UB uppintnst; カーネル割り込みマスクレベルより高い割り込みのネスト数 UB lowintnst; カーネル割り込みマスクレベル以下の割り込みのネスト数 ID maxtskid; 最大タスク ID ID ststkid; 最大スタティックスタック使用タスク ID PRI maxtskpri; 最大タスク優先度 ID maxflgid; 最大イベントフラグ ID ID maxsemid; 最大セマフォ ID ID maxmbxid; 最大メイルボックス ID ID maxmbfid; 最大メッセージバッファ ID ID maxmplid; 最大可変長メモリプール ID ID maxmpfid; 最大固定長メモリプール ID HNO maxcycno; 最大周期起動ハンドラ指定番号 HNO maxalmno; 最大アラームハンドラ指定番号 DVN maxiodvn; 最大 I/O 番号 FN maxs_fncd; 最大拡張 SVC 機能コード } T_RCFG;		パケット構造 typedef struct t_rcfg { ID maxtskid; 最大タスク ID ID ststkid; 最大スタティックスタック使用タスク ID ID maxsemid; 最大セマフォ ID ID maxflgid; 最大イベントフラグ ID ID maxdtqid; 最大データキューID ID maxmbxid; 最大メールボックス ID ID maxmtxid; 最大ミューテックス ID ID maxmbfid; 最大メッセージバッファ ID ID maxmplid; 最大可変長メモリプール ID ID maxmpfid; 最大固定長メモリプール ID ID maxcycid; 最大周期ハンドラ ID ID maxalmid; 最大アラームハンドラ ID FN maxs_fncd; 最大拡張サービスコール機能コード } T_RCFG;	

	HI7000 シリーズ（μITRON3.0 仕様準拠）		HI7000/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
4	**システムダウン**		**システムダウン**		
	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);	
5	**拡張 SVC ハンドラ定義**		**拡張 SVC ハンドラ定義**		**パケット構造 変更** 3.3.2 **参照**
	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);	
		パケット構造 typedef struct t_dsvc { ATR svcatr; **拡張 SVC ハンドラ属性** FP svchdr; **拡張 SVC ハンドラアドレス** } T_DSVC;		**パケット構造** typedef struct t_dsvc { ATR svcatr; **拡張サービスコールルーチン属性** FP svcrtn; **拡張サービスコールルーチンアドレス** } T_DSVC;	
6	**拡張機能コードの割り当てと拡張 SVC ハンドラの定義**		**－**		**廃止** 3.1.4 **参照**
	vasn_svc	ER ercd = vasn_svc(FN *p_s_fncd, T_DSVC *pk_dsvc);	**－**	**該当なし**	
7	**拡張 SVC ハンドラ終了**		**－**		**廃止**
	ret_svc	void ret_svc(ER ercd);	**－**	**該当なし**	
8	**拡張 SVC の発行**		**拡張 SVC の発行**		SVC **名変更** **非タスクから発行可** 3.3.2 **参照**
	vsys_cal	ER ercd = vsys_cal(FN fncd, VW para1, VW para2, VW para3, VW para4);	cal_svc ical_svc	ER_UINT ercd = cal_svc(FN fncd, ...); ER_UINT ercd = ical_svc(FN fncd, ...);	
		動作 **非タスクからは発行できない**		**動作** **非タスクコンテキストから発行可能**	
9	**トレースの取得**		**トレースの取得**		SVC **名変更**
	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	**割り込みハンドラの開始をトレースに取得**		**割り込みハンドラの開始をトレースに取得**		SVC **名変更**
	vbgn_int	ER ercd = vbgn_int(UINT dintno);	ivbgn_int	ER ercd = ivbgn_int(UINT dintno);	
11	**割り込みハンドラの終了をトレースに取得**		**割り込みハンドラの終了をトレースに取得**		SVC **名変更**
	vend_int	ER ercd = vend_int(UINT dintno);	ivend_int	ER ercd = ivend_int(UINT dintno);	
12	**システムの再起動**		**－**		**廃止**
	vres_sys	void vres_sys();	**－**	**該当なし**	3.1.4 **参照**
－	**－**		**カーネルの起動**		**新機能**
	－	**該当なし**	vsta_knl ivsta_knl	void vsta_knl(); void ivsta_knl();	

4.1.17 システム状態管理機能

HI7000 シリーズ（μITRON3.0 仕様準拠）			HI7000/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
−	−		コンテキストの参照		新機能
	−	該当なし	sns_ctx	BOOL state = sns_ctx();	
−	−		CPU ロック状態の参照		新機能
	−	該当なし	sns_loc	BOOL state = sns_loc();	
−	−		ディスパッチ禁止状態の参照		新機能
	−	該当なし	sns_dsp	BOOL state = sns_dsp();	
−	−		ディスパッチ保留状態の参照		新機能
	−	該当なし	sns_dpn	BOOL state = sns_dpn();	

4.1.18 DSP スタンバイ制御機能 [HI7700/4]

HI7000 シリーズ（μITRON3.0 仕様準拠）			HI7000/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
−	−		TA_COP0 属性の変更		新機能
	−	該当なし	vchg_cop	ER_UINT oldatr = vchg_cop(ATR newatr);	

4.1.19 メモリプール管理(ページプール)機能 [HI7700, HI7750]

- グレーの網掛けがかかっているところは変更になっている箇所です。
- この SVC 相違一覧表では用語が以下のように置き換えられています。
  パラメータ → 引数　　リターンパラメータ → 返値

	HI7000 シリーズ（μITRON3.0 仕様準拠）		HI7000/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
1	MMU の初期化		−		廃止 3.1.2 参照
	vini_ppl	void vini_ppl(void);	−	該当なし	
2	ページブロック獲得		−		
	vget_pbl	ER ercd = vget_pbl(VP *p_pblk, INT pagcnt, UW pagatr);	−	該当なし	
3	ページブロック獲得(ポーリング)		−		
	vpget_pbl	ER ercd = vpget_pbl(VP *p_pblk, INT pagcnt, UW pagatr);	−	該当なし	
4	ページブロック獲得(タイムアウト有)		−		
	vtget_pbl	ER ercd = vtget_pbl(VP *p_pblk, INT pagcnt, UW pagatr, TMO tmout);	−	該当なし	
5	ページブロック返却		−		
	vrel_pbl	ER ercd = vrel_pbl(VP pblk);	−	該当なし	
6	ページ属性変更		−		
	vchg_pat	ER ercd = vchg_pat(VP pagtop, INT pagcnt, UW pagatr);	−	該当なし	
7	ページプール状態参照		−		
	vref_ppl	ER ercd = vref_ppl(T_RPPL *pk_rppl);	−	該当なし	
8	物理メモリアドレスへの変換		−		
	vtrn_adr	ER ercd = vtrn_adr(VP *p_phyadr, VP pagadr);	−	該当なし	

4.1.20 オブジェクト名称管理機能

	HI7000 シリーズ（μITRON3.0 仕様準拠）		HI7000/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
1	名称からタスク ID を得る		−		廃止 3.3.14 参照
	vinq_tsk	ER ercd = vinq_tsk(ID *p_tskid, B *name);	−	該当なし	
2	名称からセマフォ ID を得る		−		
	vinq_sem	ER ercd = vinq_sem(ID *p_semid, B *name);	−	該当なし	
3	名称からイベントフラグ ID を得る		−		
	vinq_flg	ER ercd = vinq_flg(ID *p_flgid, B *name);	−	該当なし	
4	名称からメイルボックス ID を得る		−		
	vinq_mbx	ER ercd = vinq_mbx(ID *p_mbxid, B *name);	−	該当なし	
5	名称からメッセージバッファ ID を得る		−		
	vinq_mbf	ER ercd = vinq_mbf(ID *p_mbfid, B *name);	−	該当なし	
6	名称から固定長メモリプール ID を得る		−		
	vinq_mpf	ER ercd = vinq_mpf(ID *p_mpfid, B *name);	−	該当なし	
7	名称から可変長メモリプール ID を得る		−		
	vinq_mpl	ER ercd = vinq_mpl(ID *p_mplid, B *name);	−	該当なし	
8	名称から周期起動ハンドラ指定番号を得る		−		
	vinq_cyc	ER ercd = vinq_cyc(ID *p_cycno, B *name);	−	該当なし	
9	名称からアラームハンドラ指定番号を得る		−		
	vinq_alm	ER ercd = vinq_alm(ID *p_almno, B *name);	−	該当なし	

4.1.21 I/O サポート機能

	HI7000 シリーズ（μITRON3.0 仕様準拠）		HI7000/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
1	I/O ハンドラの定義		−		廃止 3.1.3 参照
	vdef_dev	ER ercd = vdef_dev(DVN iodvn, T_DDEV *pk_ddev);	−	該当なし	
2	I/O 番号の割り当てと I/O ハンドラの定義		−		
	vasn_dev	ER ercd = vasn_dev(DVN *p_iodvn, T_DDEV *pk_ddev);	−	該当なし	
3	1バイト入力		−		
	get_chr	ER ercd = get_chr(DVN iodvn, W ioun, B *p_chr);	−	該当なし	
4	1バイト出力		−		
	put_chr	ER ercd = put_chr(DVN iodvn, W ioun, B chr);	−	該当なし	
5	1行入力		−		
	get_lin	ER ercd = get_lin(DVN iodvn, W ioun, B *lin, W linsz);	−	該当なし	
6	1行出力		−		
	put_lin	ER ercd = put_lin(DVN iodvn, W ioun, B *lin);	−	該当なし	
7	CIO 属性制御		−		
	ctl_cio	ER ercd = ctl_cio(DVN iodvn, W ioun, H ioatrcd, VW ioctl);	−	該当なし	
8	CIO 状態参照		−		
	ref_cio	ER ercd = ref_cio(DVN iodvn, W ioun, H ioatrcd, VW *p_iosts);	−	該当なし	
9	GIO 要求		−		
	req_gio	ER ercd = req_gio(DVN iodvn, T_RQGIO *pk_rqgio);	−	該当なし	

4.2 エラーコード相違点

HI7000 シリーズと HI7000/4 シリーズのシステムコール（サービスコール）エラーコードの一覧比較表を以下に示します。

番号	HI7000 シリーズ			HI7000/4 シリーズ			エラー内容
	エラーコード (ニーモニック)	エラーコード値		エラーコード (ニーモニック)	エラーコード値		
1	E_OK	H'00000000	(D'0)	E_OK	H'00000000	(D'0)	正常終了
2	E_NOMEM	H'ffffffdf	(-D'10)	E_NOMEM	H'fffffff6	(-D'33)	メモリ不足
3	E_NOSPT	H'fffffff7	(-D'17)	E_NOSPT	H'fffffef	(-D'9)	未サポート機能（機能が未登録）
4	E_RSFN	H'fffffff6	(-D'20)	E_RSFN	H'ffffffec	(-D'10)	システムコールが組み込まれていない
5	E_RSATR	H'fffffff5	(-D'24)	E_RSATR	H'ffffffe8	(-D'11)	予約属性（属性が不正）
6	E_PAR	H'fffffef	(-D'33)	E_PAR	H'ffffffdf	(-D'17)	パラメータエラー
7	E_ID	H'ffffffee	(-D'35)	E_ID	H'ffffffdd	(-D'18)	不正 ID 番号
8	E_NOEXS	H'ffffffd6	(-D'52)	E_NOEXS	H'ffffffcc	(-D'42)	オブジェクトが存在しない
9	E_OBJ	H'ffffffd7	(-D'63)	E_OBJ	H'ffffffc1	(-D'41)	オブジェクトの状態不正
10	E_CTX	H'ffffffe7	(-D'69)	E_CTX	H'ffffffbb	(-D'25)	コンテキストエラー
11	E_QOVR	H'ffffffd5	(-D'73)	E_QOVR	H'ffffffb7	(-D'43)	キューイングまたはネストのオーバフロー
12	E_DLT	H'ffffffcd	(-D'81)	E_DLT	H'ffffffaf	(-D'51)	待ちオブジェクトが削除された
13	E_TMOUT	H'ffffffce	(-D'85)	E_TMOUT	H'ffffffab	(-D'50)	ポーリング失敗またはタイムアウト
14	E_RLWAI	H'ffffffcf	(-D'86)	E_RLWAI	H'ffffffaa	(-D'49)	待ち状態強制解除
15	EV_NFOUND	H'ffffff1f	(-D'225)	—	—	—	空きの ID または指定番号がない
16	—	—	—	E_NOID	H'ffffffde	(-D'34)	ID 番号不足
17	EV_ILBLK	H'ffffff1e	(-D'226)	—	—	—	不正メモリブロックの操作
18	—	—	—	E_ILUSE	H'ffffffe4	(-D'28)	サービスコール不正使用

EV_NFOUND は廃止となります。E_NOID は意味が同じであり置き換えて使用することができます。
以下に EV_NFOUND を使用しているシステムコールと、置き換え可能なサービスコールを示します。

番号	EV_NFOUND があるシステムコール	HI7000/4 シリーズの同機能サービスコール	EV_NFOUND を置き換えたエラーコード
1	cre_tsk	cre_tsk, icre_tsk	E_NOID
2	vcre_tsk	(廃止・代替なし)	(廃止・代替なし)
3	vscr_tsk	vscr_tsk, ivscr_tsk(代替)	E_NOID
4	cre_flg	cre_flg, icre_flg	E_NOID
5	vasn_flg	acre_flg, iacre_flg(代替)	E_NOID
6	cre_mbx	cre_mbx, icre_mbx	E_NOID
7	vasn_mbx	acre_mbx, iacre_mbx(代替)	E_NOID
8	cre_mbf	cre_mbf, icre_mbf	E_NOID
9	vasn_mbf	iacre_mbf, iacre_mbf(代替)	E_NOID
10	cre_mpf	cre_mpf, icre_mpf	E_NOID
11	vasn_mpf	acre_mpf, iacre_mpf(代替)	E_NOID
12	cre_mpl	cre_mpl, icre_mpl	E_NOID
13	vasn_mpl	acre_mpl, iacre_mpl(代替)	E_NOID
14	def_cyc	cre_cyc, icre_cyc(代替)	E_NOID
15	vasn_cyc	acre_cyc, iacre_cyc(代替)	E_NOID
16	def_alm	cre_alm, icre_alm(代替)	E_NOID
17	vasn_alm	acre_alm, iacre_alm(代替)	E_NOID
18	vdef_dev	(廃止・代替なし)	(廃止・代替なし)
19	vasn_dev	(廃止・代替なし)	(廃止・代替なし)
20	def_svc	def_svc	(該当エラーコードなし)
21	vasn_svc	(廃止・代替なし)	(廃止・代替なし)

EV_ILBLK は該当するエラーが別のエラーコードと一緒の扱いになります。
EV_ILBLK を使用しているシステムコールと、置き換え可能なサービスコール、置き換えたエラーコードを以下に示します。

番号	EV_ILBLK があるシステムコール	HI7000/4 シリーズの同機能サービスコール	EV_ILBLK を置き換えたエラーコード
1	rel_blf	rel_mpf	E_PAR に含まれる
2	rel_blk	rel_mpl	E_PAR に含まれる
3	vrel_pbl	(廃止・代替なし)	−
4	vchg_pat	(廃止・代替なし)	−
5	vtrn_adr	(廃止・代替なし)	−

エラーコードが、新設、廃止もしくは別のエラーになったサービスコールを以下に示します。

番号	サービスコール	HI7000 シリーズでのエラーコード	HI7000/4 シリーズでのエラーコード
1	sus_tsk	(なし)	E_CTX (新設)
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	(廃止)
7	unl_cpu	E_CTX	(廃止)
8	chg_ims	E_CTX	(廃止)

パラメータチェック機能を有効にした場合のみ出るエラーコードの相違点を以下に示します。

番号	SVC 名	エラーコード	HI7000 シリーズ	SVC 名	エラーコード	HI7000/4 シリーズ
1	タスクの生成 cre_tsk vcre_tsk vscr_tsk	E_RAR	パラメータエラー(pk_ctsk が 4 の倍数以外、p_tskid または task が奇数、stksz が 4 の倍数以外、stksz≦0、itskpri≦0、itskpri＞hi_maxtskpri、quantum＜0)	タスクの生成 cre_tsk, icre_tsk acre_tsk, iacre_tsk vscr_tsk, ivcsr_tsk	E_RAR	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,)、stk が NULL 以外で 4 の倍数以外(HI7000/4 のみ))
2	タスクの起動 sta_tsk	E_ID	不正 ID 番号(tskid≦0、tskid>hi_maxtskid)	タスクの起動 act_tsk, iact_tsk	E_ID	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)
3	タスクのレディキュー回転 rot_rdq	E_PAR	パラメータエラー(tskpri＜0、tskpri＞hi_maxtskpri)	タスクの優先順位の回転 rot_rdq, irot_rdq	E_PAR	パラメータエラー(tskpri＜0、tskpri＞CFG_MAXTSKPRI、非タスクコンテキストで tskpri=TPRI_SELF(0)を指定)
4	タスク状態参照 ref_tsk	E_PAR	パラメータエラー(pk_rtsk が 4 の倍数以外、p_tskstat が奇数、p_tskpri が奇数)	タスクの状態参照 ref_tsk, iref_tsk	E_PAR	パラメータエラー(pk_rtsk が 4 の倍数以外)
		E_ID	不正 ID 番号(tskid≦0、tskid＞hi_maxtskid、tskid=0)		E_ID	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)
5	他タスクを強制待ち状態へ移行 sus_tsk	E_ID	不正 ID 番号(tskid≦0、tskid＞hi_maxtskid)	強制待ち状態への移行 sus_tsk, isus_tsk	E_ID	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)
6	強制待ち状態のタスクを再開 rsm_tsk	E_ID	不正 ID 番号(tskid≦0、tskid＞hi_maxtskid)	強制待ち状態からの再開 rsm_tsk, irsm_tsk	E_ID	不正 ID 番号(tskid＜0、tskid＞CFG_MAXTSKID)
7	他タスクの起床 wup_tsk	E_ID	不正 ID 番号(tskid≦0、tskid＞hi_maxtskid)	他タスクの起床 wup_tsk	E_ID	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)
8	タスクの起床要求を無効化 can_wup	E_PAR	パラメータエラー(p_wupcnt が 4 の倍数以外)	タスク起床要求のキャンセル can_wup, ican_wup	—	—
		E_ID	ID 範囲外(tskid<0、tskid＞hi_maxtskid、tskid=0)		E_ID	ID 範囲外(tskid＜0、tskid＞CFG_MAXTSKID、非タスクコンテキストで tskid=TSK_SELF(0)を指定)
9	セマフォ生成 cre_sem	E_NOSPT	未サポート機能(タスク優先度順機能が組み込まれていない)	セマフォの生成 cre_sem, icre_sem acre_sem, iacre_sem	—	—
		E_PAR	パラメータエラー(p_semid が奇数、pk_csem が 4 の倍数以外、isemcnt<0、isemcnt>H'ffff)		E_PAR	パラメータエラー(pk_csem が 4 の倍数以外、maxsem=0、maxsem＞H'ffff、isemcnt＜0、isemcnt＞maxsem)
10	セマフォ状態参照 ref_sem	E_PAR	パラメータエラー(pk_rsem が 4 の倍数以外、p_wtskid が奇数、p_semcnt が 4 の倍数以外)	セマフォの状態参照 ref_sem、 iref_sem	E_PAR	パラメータエラー(pk_rsem が 4 の倍数以外)
11	イベントフラグ生成 cre_flg	E_PAR	パラメータエラー(p_flgid が奇数、pk_cflg が 4 の倍数以外)	イベントフラグ生成 cre_flg、icre_flg	E_PAR	パラメータエラー(pk_cflg が 4 の倍数以外)
12	メールボックスの生成 cre_mbx vasn_mbx	E_NOSPT	未サポート機能(タスク優先度順機能が組み込まれていない)	メールボックスの生成 cre_mbx、 icre_mbx acre_mbx、 iacre_mbx	—	—
		E_PAR	パラメータエラー(pk_cmbx が 4 の倍数以外、p_mbxid が奇数)		E_PAR	パラメータエラー(pk_cmbx が 4 の倍数以外、maxmpri≦0、 maxmpri＞CFG_MAXMSGPRI)

番号	SVC 名	エラーコード	HI7000 シリーズ	SVC 名	エラーコード	HI7000/4 シリーズ
13	メッセージバッファ生成 cre_mbf vasn_mbf	E_NOSPT	未サポート機能(タスク優先度順機能が組み込まれていない)	メッセージバッファの生成 cre_mbf, icre_mbf acre_mbf, iacre_mbf	—	—
		E_PAR	パラメータエラー(pk_cmbf が 4 の倍数以外、bufsz が 4 の倍数以外または 0 未満、bufsz が 8 バイト未満、maxmsz が 0 以下、bufsz が 0 以外で maxmsz+4＞bufsz、p_mbfid が奇数)		E_PAR	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、mbfsz が 0 以外で maxmsz+4＞mbfsz)
14	メッセージバッファから受信 rcv_mbf、 prcv_mbf trcv_mbf	E_PAR	パラメータエラー(p_msgsz が 4 の倍数以外、msg が 4 の倍数以外、tmout≦-2)	メッセージバッファからの受信 rcv_mbf、 prcv_mbf trcv_mbf	E_PAR	パラメータエラー(msg が 4 の倍数以外、tmout≦-2)
15	割り込みハンドラの定義 def_int	E_PAR	パラメータエラー(pk_dint が 4 の倍数以外、inthdr が奇数) 〈HI7000〉 システム予約番号指定(16≦dintno≦31) 〈HI7700、HI7750〉 不正例外コード(dintno が H'20 の倍数以外、dintno＞hi_maxexccd)	割り込みハンドラの定義 def_inh、 idef_inh	E_PAR	パラメータエラー(pk_dinh が 4 の倍数以外、inthdr が奇数) 〈HI7000/4〉 不正番号指定(16≦inhno≦31、inhno＞CFG_MAXVCTNO) 〈HI7700/4、HI7750/4〉 不正番号指定(inhno が H'20 の倍数以外、inhno=H'160、inhno＞CFG_MAXVCTNO)
16	拡張 SVC の発行 vsys_cal	—	—	拡張サービスコールの発行 cal_svc、 ical_svc	E_RSFN	fncd が不正あるいは使用できない
17	固定長メモリプール生成 cre_mpf vasn_mbf	E_NOSPT	未サポート機能(タスク優先度順機能が組み込まれていない)	固定長メモリプール生成 cre_mpf、 icre_mpf acre_mpf、 iacre_mpf	—	—
		E_PAR	パラメータエラー(p_mpfid が奇数、pk_cmpf が 4 の倍数以外、mpfcnt≦0、blfsz が 4 の倍数以外または 0 以下)		E_PAR	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、mpf が NULL 以外で 4 の倍数以外(HI7000/4 のみ) mpfmb が 4 の倍数以外(HI7000/4 で CFG_MPFMANAGE チェック有りの場合のみ))
18	可変長メモリプール生成 cre_mpl vasn_mpl	E_PAR	パラメータエラー(pk_cmpl が 4 の倍数以外、mplsz が 4 の倍数以外または 20 バイト未満、p_mplid が奇数)	可変長メモリプール生成 cre_mpl、 icre_mpl acre_mpl、 iacre_mpl	E_PAR	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、 mpl が NULL 以外で 4 の倍数以外(HI7000/4 のみ))
19	システムクロック設定 set_tim	E_PAR	パラメータエラー(pk_tim が 4 の倍数以外、pk_tim で示される値が負)	システムクロック設定 set_tim、 iset_tim	E_PAR	パラメータエラー(p_systim が 4 の倍数以外)
20	タスク遅延 dly_tsk	E_PAR	パラメータエラー(dlytim＜0)	タスク遅延 dly_tsk	—	—
21	周期ハンドラの状態参照 ref_cyc	—	—	周期ハンドラの状態参照 ref_cyc、 iref_cyc	E_ID	不正 ID 番号(cycid≦0、cycid＞CFG_MAXCYCID)

5. コンフィギュレーション方法の相違

HI7000 シリーズでは、カーネルビルドファイルとセットアップファイルの 2 つを作成していましたが、HI7000/4 シリーズでは、コンフィギュレータを使ってこれらに相当するファイルを作ります。

コンフィギュレータは、カーネルの動作パラメータを設定するためのツールです。コンフィギュレータは、設定された内容にしたがっていくつかの C ソースファイルを生成します。生成されたファイルおよびアプリケーションプログラムをビルド（コンパイル・リンク）することで、システム（ロードモジュール）を作成します。

コンフィギュレータのウィンドウは、構築情報入力パート一覧ウィンドウ（左側）と構築情報入力ウィンドウ（右側）で構成されています。構築情報入力ウィンドウにより各種情報を入力後、「構築ファイルの生成」コマンド（メニューおよびツールボタン）により、構築ファイルを生成します。
詳細は HI7000/4 シリーズのマニュアルをご覧ください。

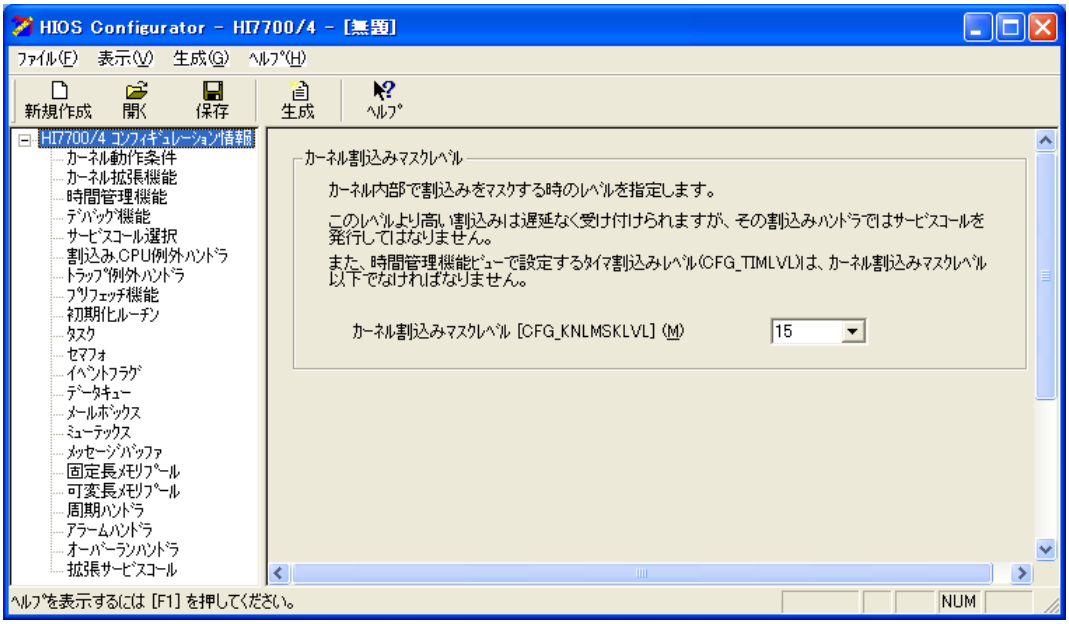

図 5-1　HI7000/4 シリーズのコンフィギュレータ概観

HI7000シリーズからHI7000/4シリーズへの移行ガイド
μITRON仕様OS移行マニュアル

発行年月日　2005 年 6 月 7 日　Rev.1.00
発　　行　株式会社ルネサス テクノロジ 営業企画統括部
　　〒100-0004 東京都千代田区大手町 2-6-2
編　　集　株式会社ルネサス小平セミコン 技術ドキュメント部

株式会社ルネサス テクノロジ 営業企画統括部　〒100-0004 東京都千代田区大手町2-6-2 日本ビル

営業お問合せ窓口
株式会社ルネサス販売

http://www.renesas.com

本社	〒100-0004	千代田区大手町2-6-2 (日本ビル)	(03) 5201-5350
京浜支社	〒212-0058	川崎市幸区鹿島田890-12 (新川崎三井ビル)	(044) 549-1662
西東京支社	〒190-0023	立川市柴崎町2-2-23 (第二高島ビル2F)	(042) 524-8701
東北支社	〒980-0013	仙台市青葉区花京院1-1-20 (花京院スクエア13F)	(022) 221-1351
いわき支店	〒970-8026	いわき市平小太郎町4-9 (平小太郎ビル)	(0246) 22-3222
茨城支店	〒312-0034	ひたちなか市堀口832-2 (日立システムプラザ勝田1F)	(029) 271-9411
新潟支店	〒950-0087	新潟市東大通1-4-2 (新潟三井物産ビル3F)	(025) 241-4361
松本支社	〒390-0815	松本市深志1-2-11 (昭和ビル7F)	(0263) 33-6622
中部支社	〒460-0008	名古屋市中区栄4-2-29 (名古屋広小路プレイス)	(052) 249-3330
関西支社	〒541-0044	大阪市中央区伏見町4-1-1 (明治安田生命大阪御堂筋ビル)	(06) 6233-9500
北陸支社	〒920-0031	金沢市広岡3-1-1 (金沢パークビル8F)	(076) 233-5980
広島支店	〒730-0036	広島市中区袋町5-25 (広島袋町ビルディング8F)	(082) 244-2570
鳥取支店	〒680-0822	鳥取市今町2-251 (日本生命鳥取駅前ビル)	(0857) 21-1915
九州支社	〒812-0011	福岡市博多区博多駅前2-17-1 (ヒロカネビル本館5F)	(092) 481-7695

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
　総合お問合せ窓口：コンタクトセンタ　E-Mail: csc@renesas.com

μITRON 仕様 OS 移行マニュアル
HI7000 シリーズから HI7000/4 シリーズへの移行ガイド

ルネサス エレクトロニクス株式会社
神奈川県川崎市中原区下沼部1753 〒211-8668

RJJ05B0777-0100