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。

User's Manual

μITRON 仕様 OS 移行マニュアル

MR32R から MR32R/4 への移行ガイド

アプリケーションノート

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

表目次

図目次

1. はじめに

本資料は、μITRON3.0 仕様準拠の MR32R から、μITRON4.0 仕様準拠の MR32R/4 への移行の方法について解説します。

本資料は、MR32R から MR32R/4 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。

2. 概要

ここでは MR32R/4 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。

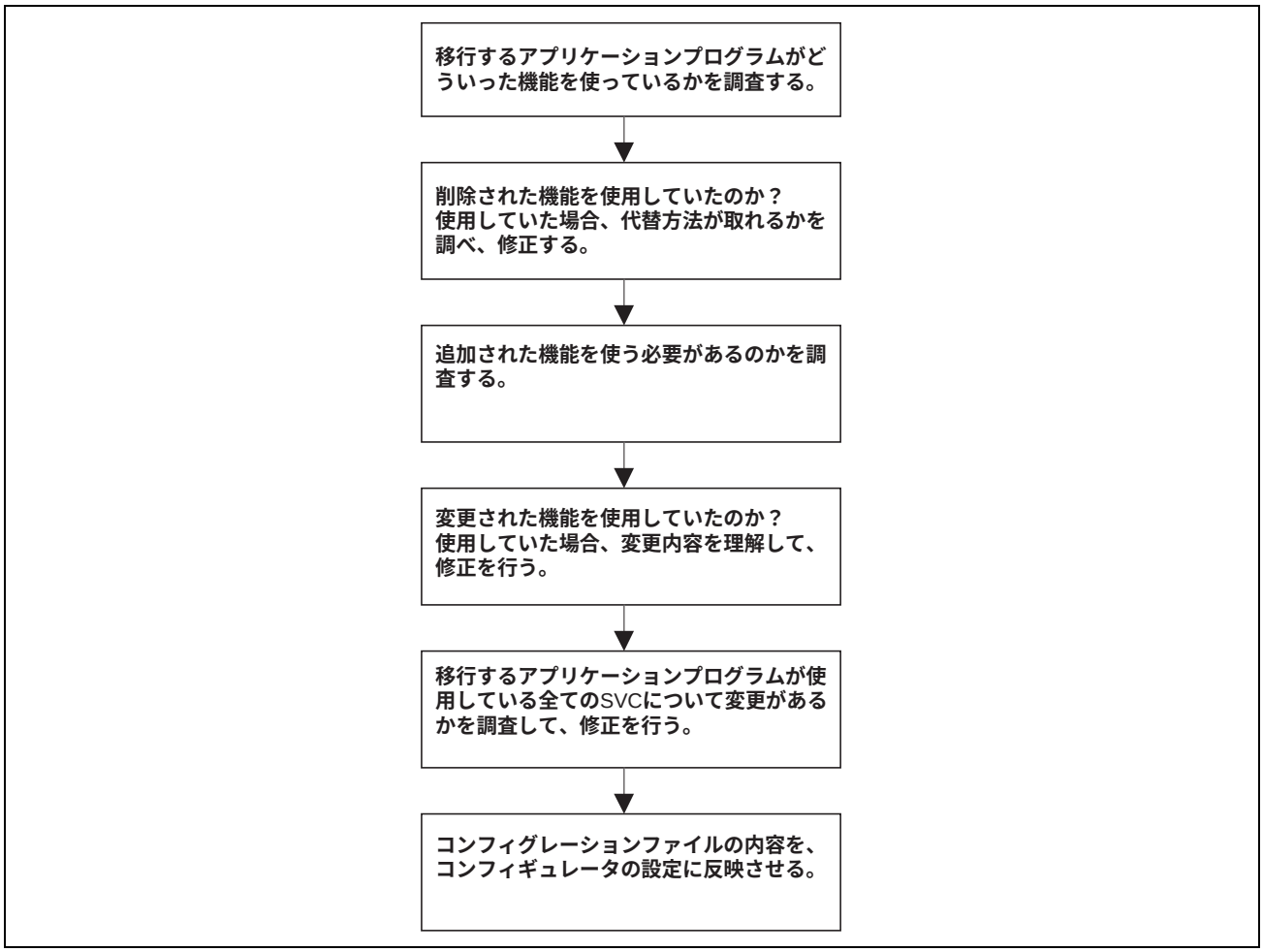

図 2-1　移行の手順

MR32R と MR32R/4 の比較概要を表 2-1 に示します。

表 2-1　MR32R と MR32R/4 の比較概要

番号	MR32R の機能	MR32R/4 での変更点	本資料での番号
1	拡張機能（優先度付きメールボックス）	メールボックス機能に統合	3.1.1
2	データキュー機能	新機能	3.2.1
3	タスク例外機能	新機能	3.2.2
4	タスクの多重起動要求	新機能	3.2.3
5	“i”のつくサービスコール	追加	3.2.4
6	ディスパッチと CPU ロック	意味変更	3.3.1
7	オブジェクトの属性	各オブジェクトに属性追加	3.3.2
8	wup_tsk, sus_tsk	自タスク指定可能	3.3.3
9	イベントフラグ	クリア指定が変更	3.3.4
10	メールボックス	メッセージヘッダ型追加	3.3.5
11	ランデブ	タイムアウトが変更	3.3.6
12	時間管理	時間単位、時間管理が変更	3.3.7
13	周期ハンドラ	全般的に変更	3.3.8
14	アラームハンドラ	全般的に変更	3.3.9

3. 追加/削除/変更機能

3.1 削除機能

3.1.1 拡張機能（優先度付きメールボックス）

MR32R/4 では、拡張機能（優先度付きメールボックス）が削除されました。
この機能は同期・通信機能（メールボックス）に含まれており、代替え可能です。

3.1.2 その他の機能

表 3-1 に MR32R から削除された機能に対する、MR32R/4 での代替え手段を示します。

表 3-1　MR32R/4 **との対応表**

番号	SVC	機能	代替え手段
1	def_exc	**例外ハンドラを定義**	**タスク例外機能で代替え可能**
2	vclr_ems	**例外マスクをクリア**	**タスク例外機能で代替え可能**
3	vset_ems	**例外マスクをセット**	**タスク例外機能で代替え可能**
4	vras_fex	**強制例外を起動**	**タスク例外機能で代替え可能**
5	ref_sys	CPU **や** OS **の状態参照**	**代替えはありません。**get_tid, sns_xxx **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

MR32R/4 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** 3-2 にデータキュー機能の概要を示します。

表 3-2 **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);
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);
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);
		iprcv_dtq	ER ercd = iprcv_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 タスク例外機能

MR32R/4 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** 3-3 にタスク例外機能の概要を示します。

表 3-3 **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);
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);

3.2.3 タスクの多重起動要求機能

MR32R/4 では、新たにタスクの多重起動要求が追加されています。

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。

3.2.4 先頭に"i"がつくサービスコール

MR32R/4 で、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** 3-4 に一覧を示します。

表 3-4 **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	iact_tsk	**タスクの起動**	2	ista_tsk	**タスク起動(起動コード指定)**
3	ican_act	**タスク起動要求のキャンセル**	4	ichg_pri	**タスク優先度変更**
5	iget_pri	**タスク優先度の参照**	6	iref_tsk	**タスクの状態参照**
7	iref_tst	**タスクの状態参照(簡易版)**	8	iwup_tsk	**タスクの起床**
9	ican_wup	**タスク起床要求のキャンセル**	10	irel_wai	**待ち状態の強制解除**
11	isus_tsk	**強制待ち状態への移行**	12	irsm_tsk	**強制待ち状態からの再開**
13	ifrsm_tsk	**強制待ち状態からの強制再開**	14	iras_tex	**タスク例外処理の要求**
15	iref_tex	**タスク例外処理の状態参照**	16	isig_sem	**セマフォ資源の返却**
17	ipol_sem	**セマフォ資源の獲得(ポーリング)**	18	iref_sem	**セマフォの状態参照**
19	iset_flg	**イベントフラグのセット**	20	iclr_flg	**イベントフラグのクリア**
21	ipol_flg	**イベントフラグ待ち(ポーリング)**	22	iref_flg	**イベントフラグの状態参照**
23	ipsnd_dtq	**データキューへの送信(ポーリング)**	24	ifsnd_dtq	**データキューへの強制送信**
25	iprcv_dtq	**データキューからデータ受信(ポーリング)**	26	iref_dtq	**データキューの状態参照**
27	isnd_mbx	**メールボックスへの送信**	28	iprcv_mbx	**メールボックスからの受信(ポーリング)**
29	iref_mbx	**メールボックスの状態参照**	30	iref_drv	**ランデヴ状態を参照**
31	iref_por	**ランデヴポート状態を参照**	32	ipget_mpf	**固定長メモリブロックの獲得(ポーリング)**
33	irel_mpf	**固定長メモリブロックの返却**	34	iref_mpf	**固定長メモリプールの状態参照**
35	iref_mpl	**可変長メモリプールの状態参照**	36	iset_tim	**システム時刻の設定**
37	iget_tim	**システム時刻の参照**	38	ista_cyc	**周期ハンドラの動作開始**
39	iref_cyc	**周期ハンドラの状態参照**	40	ista_alm	**アラームハンドラの動作開始**
41	istp_alm	**アラームハンドラの動作停止**	42	iref_alm	**アラームハンドラの状態参照**
43	irot_rdq	**タスクの優先順位の回転**	44	iget_tid	**実行状態のタスク ID の参照**
45	iloc_cpu	CPU **ロック状態への移行**	46	iunl_cpu	CPU **ロック状態の解除**
47	idef_int	**割り込みハンドラの定義**	48	iref_ver	**バージョン情報の参照**

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

MR32R では、CPU ロック状態はディスパッチと割込みが禁止された状態と扱っていたのを、MR32R/4 ではそれぞれ独立した状態として扱います。

例えば、MR32R/4 で、ディスパッチ禁止・CPU ロック状態の場合、CPU ロックを解除してもディスパッチ禁止は解除されません。

例えば、MR32R/4 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 3-5 MR32R 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 3-6 MR32R/4 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

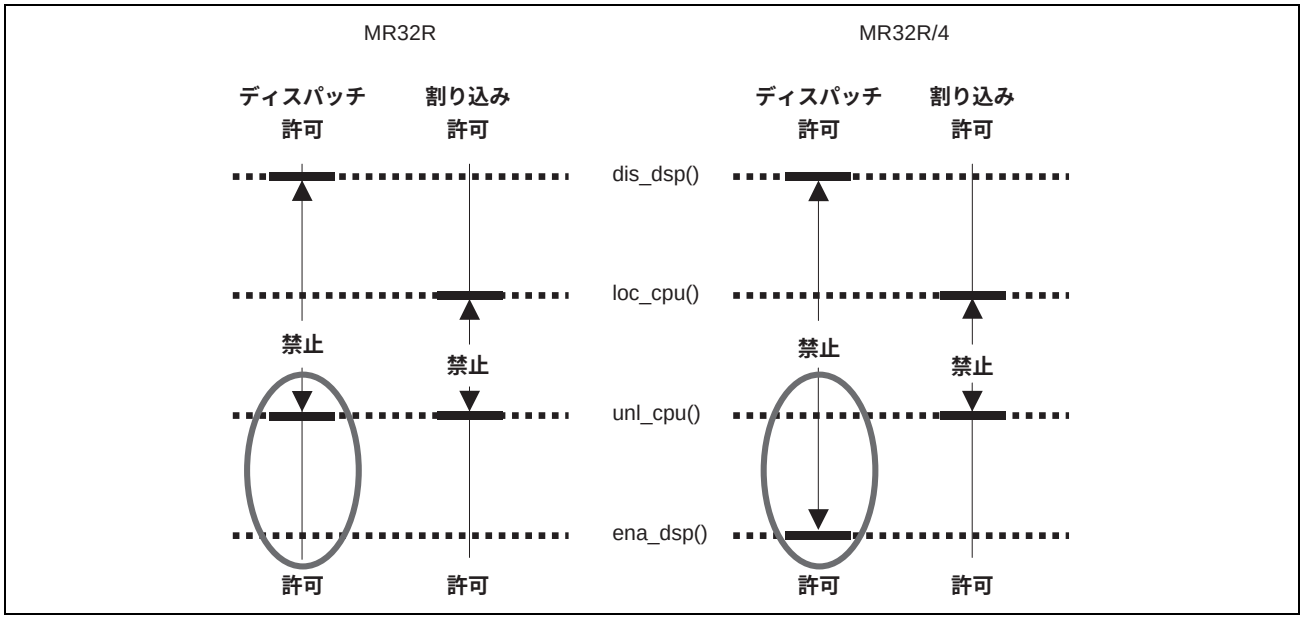

図 3-1 ディスパッチ禁止と CPU ロックの変更

3.3.2 オブジェクトの属性追加

各サービスコールにて、作成するオブジェクトに以下の属性が追加・削除されています。
MR32R では指定しても関知しないとなっていた属性ですが、MR32/4 では追加された属性を指定する必要があります。
以下にサービスコールと属性追加になるオブジェクト名、属性名、属性の説明を示します。

表 3-7　属性追加されたサービスコールと属性説明

サービスコール	機能	属性名	属性の説明
cre_tsk	タスク	TA_ACT	タスク生成後、act_tsk と同様の処理を行い、タスクを実行可能状態に移行します。
cre_sem	セマフォ	TA_TFIFO (*1)	待ちタスクのキューイングは FIFO 順
		TA_TPRI	待ちタスクのキューイングは優先度順
cre_flg	イベントフラグ	TA_TFIFO (*1)	待ちタスクのキューイングは FIFO 順
		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禁止
		TA_WMUL (*1)	複数タスクの待ちを許可
		TA_CLR	クリア指定
cre_mbx	メールボックス	TA_TFIFO (*1)	待ちタスクのキューイングは FIFO 順
		TA_TPRI	待ちタスクのキューイングは優先度順
		TA_MPRI	メッセージキューのキューイングは優先度順
		TA_MFIFO (*1)	メッセージキューのキューイングは FIFO 順
cre_mbf	メッセージバッファ	TA_TFIFO (*1)	送信待ちタスクのキューイングは FIFO 順
		TA_TPRI	送信待ちタスクのキューイングは優先度順
cre_por	ランデヴポート	TA_TFIFO (*1)	呼び出し待ちタスクのキューイングは FIFO 順
		TA_TPRI	呼び出し待ちタスクのキューイングは優先度順
cre_mpf	固定長メモリプール	TA_TFIFO (*1)	待ちタスクのキューイングは FIFO 順
		TA_TPRI	待ちタスクのキューイングは優先度順
cre_mpl	可変長メモリプール	TA_TFIFO (*1)	待ちタスクのキューイングは FIFO 順

*1：MR32R でのデフォルトの設定値と同じ機能の属性。

MR32R/4 で追加されたものの、カーネルが無視している属性と、その属性があるサービスコールを以下に示します。

表 3-8　カーネル無視の属性

サービスコール	機能	属性名	属性の説明
cre_tsk	タスク	TA_HLNG	高級言語で記述した際に指定。
cre_cyc	周期ハンドラ		
cre_alm	アラームハンドラ	TA_ASM	アセンブリ言語で記述した際に指定。
def_inh	割り込みハンドラ		

MR32R/4 で削除された属性を以下に示します。

表 3-9　MR32R/4 で削除された属性

サービスコール	機能	属性名	代替え方法
cre_tsk	タスク	__MR_USER	パケットのスタックの先頭アドレスを入れる変数に、ユーザが確保したスタックの先頭アドレスを入れる。 属性の値が、__MR_INT,__MR_EXT の場合はスタックの先頭アドレスを入れる変数に、NULL を入れる必要がある。
cre_mbx	メールボックス	__MR_USER	代替えはなし
cre_mbf	メッセージバッファ	__MR_USER	パケットのメッセージバッファ領域の先頭アドレスを入れる変数に、ユーザが確保したメッセージバッファ領域の先頭アドレスを入れる。 属性の値が、__MR_INT,__MR_EXT の場合はメッセージバッファ領域の先頭アドレスを入れる変数に、NULL を入れる必要がある。
cre_mpf	固定長メモリプール	__MR_USER	パケットの固定長メモリプール領域の先頭アドレスを入れる変数に、ユーザが確保した固定長メモリプール領域の先頭アドレスを入れる。 属性の値が、__MR_INT,__MR_EXT の場合は固定長メモリプール領域の先頭アドレスを入れる変数に、NULL を入れる必要がある。
cre_mpl	可変長メモリプール	__MR_USER	パケットの可変長メモリプール領域の先頭アドレスを入れる変数に、ユーザが確保した可変長メモリプール領域の先頭アドレスを入れる。 属性の値が、__MR_INT,__MR_EXT の場合は可変長メモリプール領域の先頭アドレスを入れる変数に、NULL を入れる必要がある。

3.3.3　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。

表 3-10　wup_tsk、sus_tsk の変更

パラメータ：tskid	MR32R	MR32R/4
TSK_SELF(0)	指定不可。正しく動作しない。	自タスク指定

3.3.4 イベントフラグのクリア指定変更

MR32R 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、MR32R/4 ではイベントフラグ自体にクリア指定があります。クリア指定の変更内容を**図** 3-2 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.

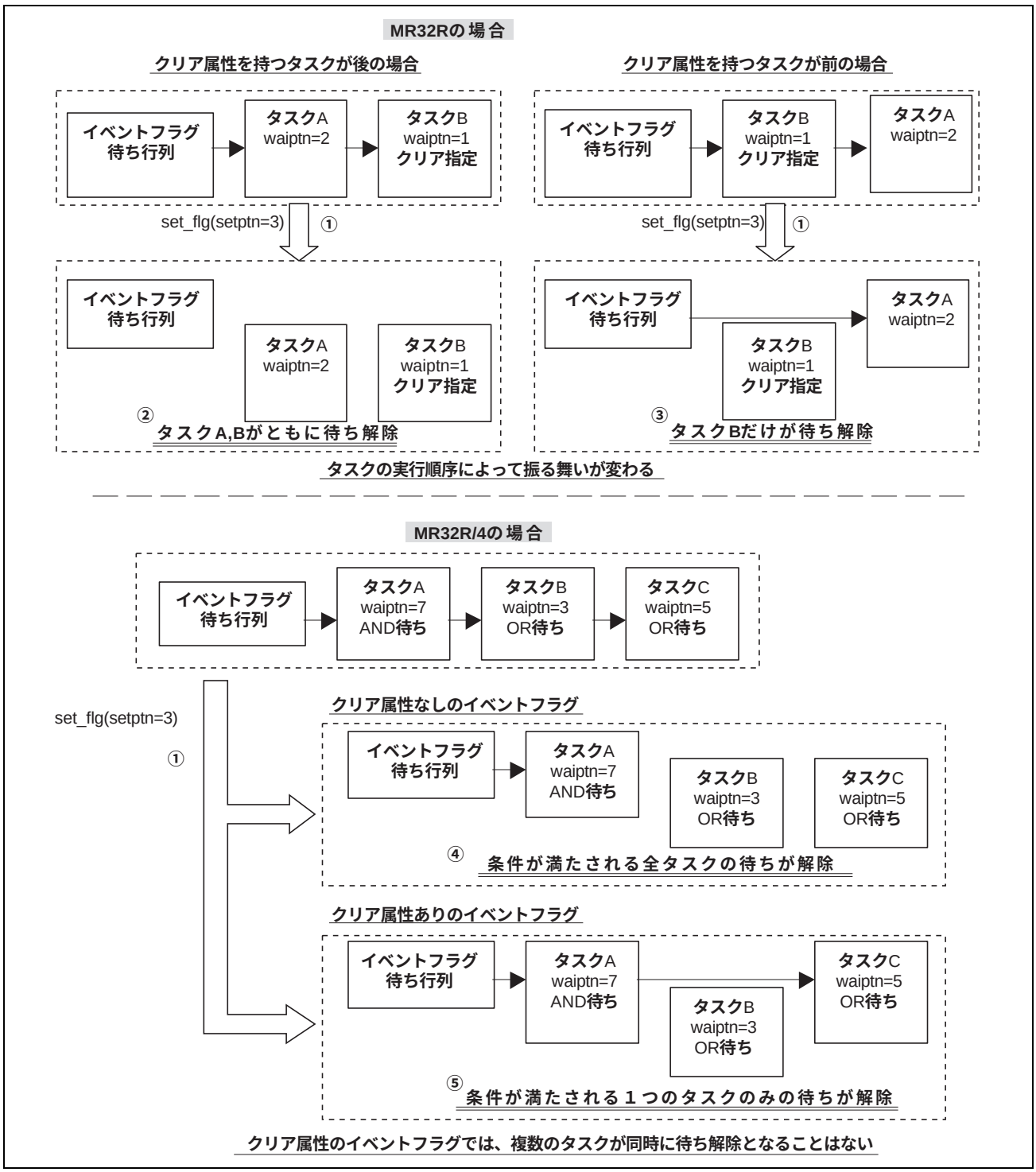

図 3-2 **イベントフラグのクリア指定変更**

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３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 3 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、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。

3.3.5 メールボックスの変更

MR32R のメールボックスは、メッセージが 32 ビットのデータとして受け渡しすることも可能でしたが、MR32R/4 のメールボックスは、メッセージの先頭アドレス受け渡しのみに変更されました。
データ受け渡しは、データキュー機能で代替え可能です。
また、MR32R の拡張機能（優先度付きメールボックス）は、MR32R/4 ではメールボックス機能に統合されています。

表 3-11 メールボックス関連機能の変更

機能	MR32R	MR32R/4
32 ビットのデータ受け渡し	メールボックス機能	データキュー機能
優先度付きのメールボックス	拡張機能（優先度付きメールボックス）	メールボックス機能

3.3.6 ランデブの変更

MR32R と MR32R/4 とではランデブ呼び出しでタイムアウトとなる箇所、動作が違います。以下の**表** 3-12、**図** 3-3 に詳細を示します。

このタイムアウト時の動作変更により、MR32R/4 では pcal_por がなくなり、tcal_por のタイムアウトが pcal_por の代用になります。

表 3-12 **ランデブの動作変更**

機能	MR32R **の動作**	MR32R/4 **の動作**
ランデブ呼び出しでのタイムアウトまでの時間	**ランデブ呼び出し待ち解除までの時間**	**ランデブ終了までの時間**
ランデブ呼び出しでのタイムアウト時の動作	**発行したタスクを呼出待ち行列につなぐ**	**処理をキャンセル**

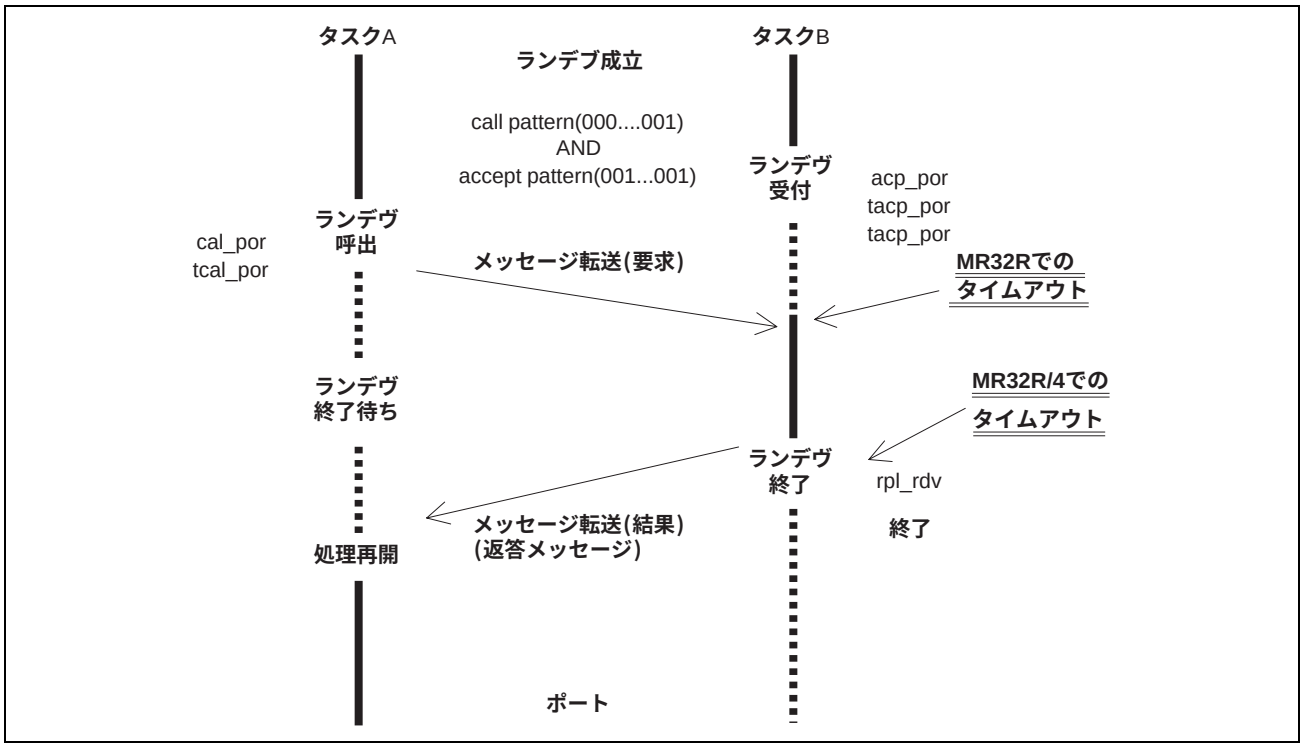

図 3-3 **ランデブのタイムアウト変更**

3.3.7 時間管理方法

時刻管理と、時間管理が変更になっています。

MR32R での時間および時刻の管理単位は、タイムティック数（カーネルが使用するハードウェアタイマの割り込み回数）です。MR32R/4 での管理単位は、1ms 単位となります。

表 3-13 時間パラメータの単位時間

番号	変更点	MR32R	MR32R/4
1	**時間パラメータの単位**	**タイムティック数**	ms

MR32R は、指定タイムティック数が経過した時点でタイムアウトとなりますが、MR32R/4 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。

MR32R と MR32R/4 の時間管理の違いについては**図 3-4** 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場合で、実際のタイムアウトまでの時間は、MR32R の設定値から－１したタイムアウト値を設定すれば、MR32R /4 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、

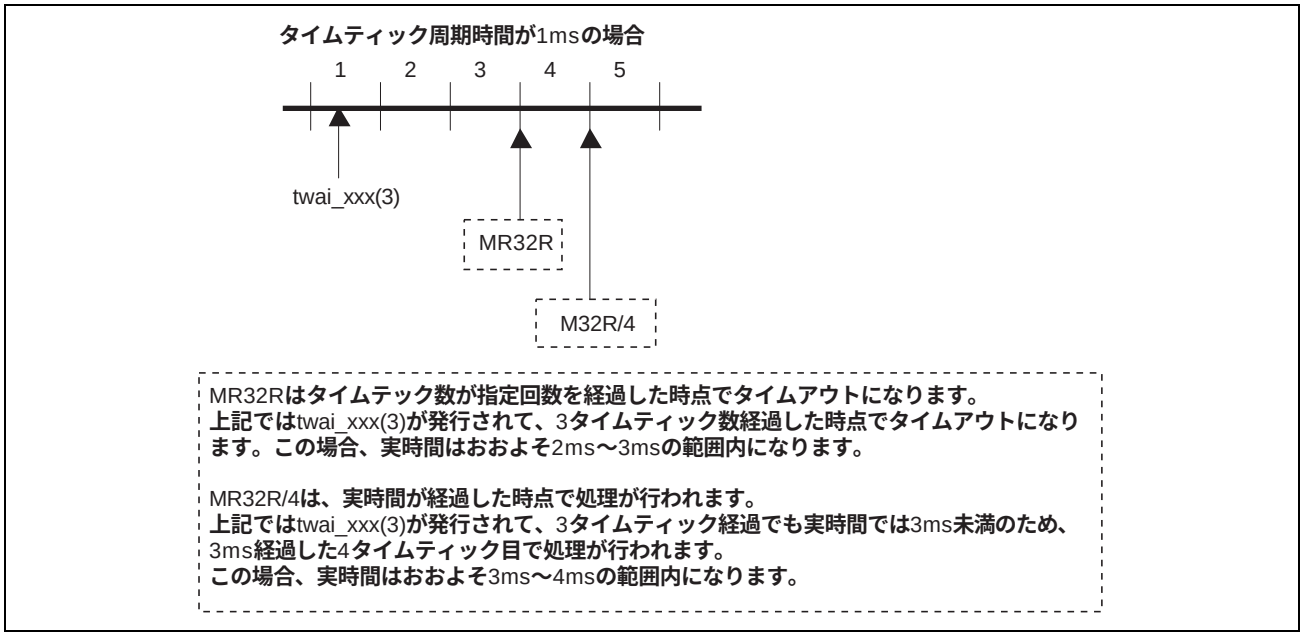

図 3-4 MR32R と MR32R/4 との時間管理の違い

3.3.8 周期ハンドラ機能

MR32R/4 の周期ハンドラは、MR32R 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を参考にして修正を行ってください。

表 3-14 MR32R と MR32R/4 の周期ハンドラ比較表

MR32R の周期ハンドラ		MR32R/4 の周期ハンドラ	
def_cyc	周期起動ハンドラ定義	cre_cyc	周期ハンドラの生成
		acre_cyc	周期ハンドラの生成 (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	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	
—	—	del_cyc	周期ハンドラの削除

表 3-15 移行の際の設定

動作	MR32R の設定	MR32R/4 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_ON	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

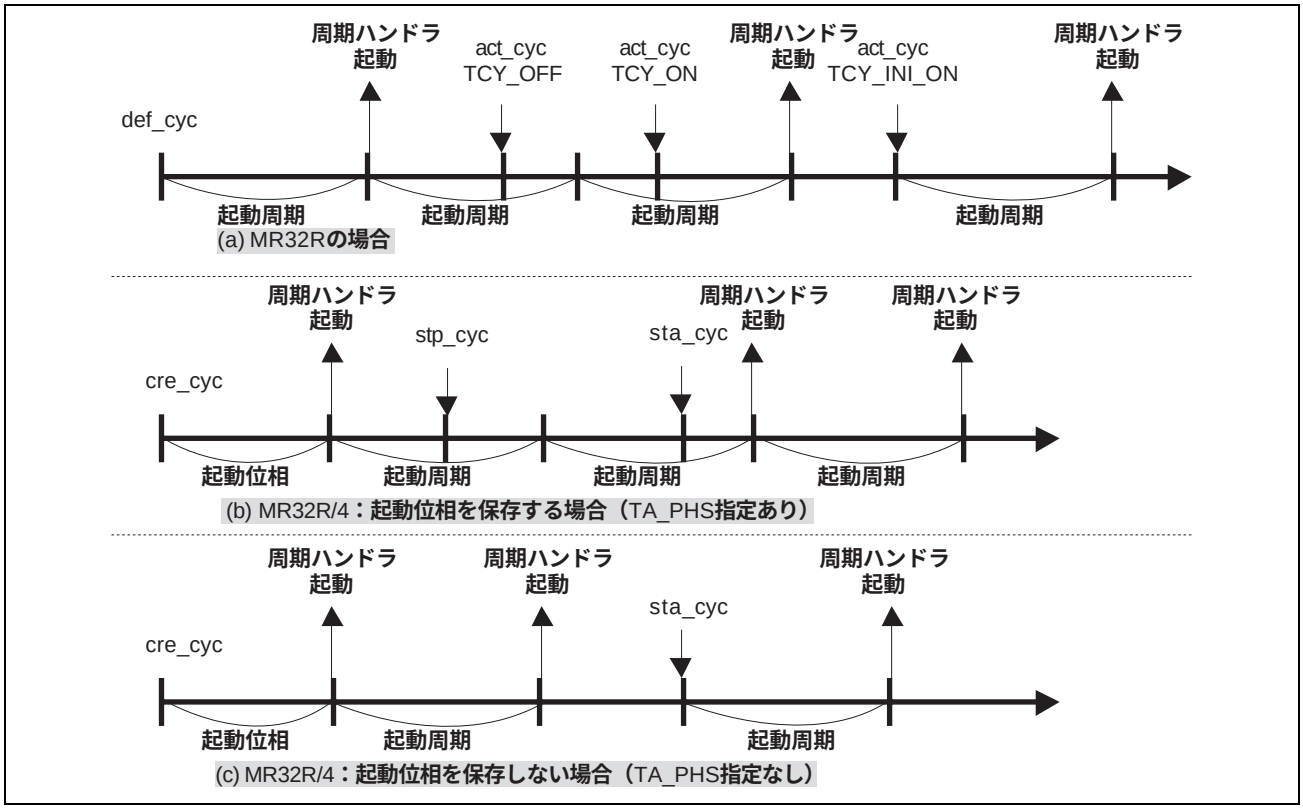

図 3-5 周期ハンドラの動作の違い

3.3.9 アラームハンドラ機能

MR32R/4 のアラームハンドラは、MR32R 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を参考にして修正を行ってください。

表 3-16 MR32R と MR32R/4 のアラームハンドラ比較表

MR32R のアラームハンドラ		MR32R/4 のアラームハンドラ	
―	―	cre_alm	**アラームハンドラの生成**
		acre_alm	**アラームハンドラの生成** (ID **番号自動割り付け**)
		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	
―	―	del_alm	**アラームハンドラの削除**

3.3.10 コーティング例

(1) タスク

タスクのコーディング例を**図 3-6**、**図 3-7** に示します。

```
#include <mr32r.h>          ←ファイル先頭で必ずシステムディレクトリのなかの“mr32r.h”
#include “id.h”               とカレントディレクトリ内の“id.h”をインクルードして
                              ください。



void task(void)             ←タスク開始関数の戻り値はありません。したがって、
{                             void型で宣言してください。
      /*処理*/
      ext_tsk();
}                           ←タスク開始関数の出口では、かならずext_tsk()を記述して
                              ください。
```

図 3-6 MR32R：タスクのコーティング例

```
#include <itron.h>          ←ファイル先頭で必ずシステムディレクトリのなかの
#include <kernel.h>           “itron.h、kernel.h”とカレントディレクトリ内の
#include “kernel_id.h”       “kernel_id.h”をインクルードしてください。


void task(VP_INT stacd)     ←タスク開始関数の戻り値はありません。したがって、void型で
{                             宣言してください。
      /*処理*/
}                           ←タスク開始関数の出口では、ext_tsk()を記述する必要は
                              ありません。カーネルが自動的にext_tskを呼び出します。
```

図 3-7 MR32R/4：タスクのコーティング例

(2) **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** 3-8、**図** 3-9 に示します。

```
#include <mr32r.h>          ←ファイル先頭で必ずシステムディレクトリのなかの“mr32r.h”
#include “id.h”                とカレントディレクトリ内の“id.h”をインクルードして
                                ください。

void int_handler(void)      ←割り込みハンドラ開始関数の戻り値および引き数は、
                                必ずvoid型で宣言してください。
{
       /* 処理 */
      iwup_tsk(ID_main);
}
```

図 3-8　MR32R：**割り込みハンドラのコーディング例**

```
#include <itron.h>          ←ファイル先頭で必ずシステムディレクトリのなかの“mr32r.h”
#include <kernel.h>             とカレントディレクトリ内の“id.h”をインクルードして
#include “kernel_id.h”          ください。

void int_handler(void)      ←割り込みハンドラ開始関数の戻り値および引き数は、
                                必ずvoid型で宣言してください。
{
       /* 処理 */
       iwup_tsk(ID_main);   ←タスク開始関数の出口では、かならずext_tsk()を記述して
                                ください。
}
```

図 3-9　MR32R/4：**割り込みハンドラのコーディング例**

4. サービスコールの相違

4.1 SVC 相違点一覧表

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。
- 変更内容には API レベルでの変更を記述しています。
- 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

MR32R シリーズ（μITRON3.0 仕様準拠）			MR32R/4 シリーズ（μITRON4.0 仕様準拠）			変更内容
	機能			機能		
	SVC	API		SVC	API	
1	**タスクを生成**			**タスクの生成**		**パケット構造　変更**
	cre_tsk	ER ercd = cre_tsk (ID tskid,T_CTSK *pk_ctsk);		cre_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; **タスク起動時優先度** INT stksz; **スタックサイズ** VP stk; **ユーザーが確保したスタックの先頭アドレス** } T_CTSK;			**パケット構造** typedef struct t_ctsk { ATR tskatr **タスク属性** VP_INT exinf **タスクの拡張情報** FP task **タスクの起動番地** PRI itskpri **タスクの起動時優先度** SIZE stksz **タスクのスタック領域のサイズ(バイト数)** VP stk **タスクのスタック領域の先頭番地** } T_CTSK;	
−	−			**タスクの生成**(ID **自動割り当て**)		**新機能**
	該当なし	−		acre_tsk	ER_ID tskid = acre_tsk(T_CTSK *pk_ctsk);	
2	**タスクを削除**			**タスクの削除**		
	del_tsk	ER ercd = del_tsk (ID tskid);		del_tsk	ER ercd = del_tsk(ID tskid);	
3	**タスクを起動**			**タスクの起動(起動コード指定)**		**引数型変更**
	sta_tsk	ER ercd = sta_tsk (ID tskid, INT stacd);		sta_tsk	ER ercd = sta_tsk(ID tskid,VP_INT stcd);	
		引数 INT stacd; **タスク起動コード**			**引数** VP_INT stcd **タスク起動コード**	
4	**タスクを起動(ハンドラ専用)**			**タスクの起動(起動コード指定)(ハンドラ専用)**		**引数型変更**
	ista_tsk	ER ercd = ista_tsk (ID tskid, INT stacd);		ista_tsk	ER ercd = ista_tsk (ID tskid,VP_INT stcd);	
		引数 INT stacd; **タスク起動コード**			**引数** VP_INT stcd **タスク起動コード**	
−	−			**タスクの起動**		**新機能**
	該当なし	−		act_tsk	ER ercd = act_tsk(ID tskid);	
				iact_tsk	ER ercd = iact_tsk(ID tskid);	
−	−			**起動要求カウントのキャンセル**		**新機能**
	該当なし	−		can_act	ER_UINT actcnt = can_act(ID tskid);	
				ican_act	ER_UINT actcnt = ican_act(ID tskid);	
5	**自タスクを正常終了**			**自タスクの終了**		
	ext_tsk	void ext_tsk ();		ext_tsk	ext_tsk();	
6	**自タスクを終了後削除**			**自タスクの終了および削除**		
	exd_tsk	void exd_tsk ();		exd_tsk	exd_tsk();	
7	**他タスクを強制的に異常終了**			**タスクの強制終了**		
	ter_tsk	ER ercd = ter_tsk (ID tskid);		ter_tsk	ER ercd = ter_tsk(ID tskid);	

	MR32R シリーズ（μITRON3.0 仕様準拠）		MR32R/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
8	タスクの優先度を変更		タスク優先度の変更		
	chg_pri	ER ercd = chg_pri (ID tskid, PRI tskpri);	chg_pri	ER ercd = chg_pri(ID tskid, PRI tskpri);	
9	タスクの優先度を変更(ハンドラ専用)		タスク優先度の変更(ハンドラ専用)		
	ichg_pri	ER ercd = ichg_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);	
10	タスクのディスパッチを禁止		ディスパッチの禁止		機能変更 3.3.1 参照
	dis_dsp	ER ercd = dis_dsp ();	dis_dsp	ER ercd = dis_dsp();	
11	タスクのディスパッチを許可		ディスパッチの許可		機能変更 3.3.1 参照
	ena_dsp	ER ercd = ena_dsp ();	ena_dsp	ER ercd = ena_dsp();	
12	レディキューを回転		タスク優先順位の回転		
	rot_rdq	ER ercd = rot_rdq (PRI tskpri);	rot_rdq	ER ercd = rot_rdq(PRI tskpri);	
13	レディキューを回転(ハンドラ専用)		タスク優先順位の回転(ハンドラ専用)		
	irot_rdq	ER ercd = irot_rdq (PRI tskpri);	irot_rdq	ER ercd = irot_rdq(PRI tskpri);	
14	待ち状態を強制解除		待ち状態の強制解除		
	rel_wai	ER ercd = rel_wai (ID tskid);	rel_wai	ER ercd = rel_wai(ID tskid);	
15	待ち状態を強制解除(ハンドラ専用)		待ち状態を強制解除(ハンドラ専用)		
	irel_wai	ER ercd = irel_wai (ID tskid);	irel_wai	ER ercd = irel_wai(ID tskid);	
16	自タスクの ID を取得		実行中タスク ID の参照		動作変更
	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時実行していたタスクの ID を返す。	
17	タスクの状態を参照		タスクの状態参照		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 PR tskpri 現在の優先度 UH tskstat タスク状態 UINT tskwait タスクの待ち要因 ID wid タスクの待ちオブジェクト ID INT wupcnt 起床要求キューイング数 ATR tskatr タスク属性 FP task タスク起動アドレス PRI itskpri タスク起動時優先度 INT stksz スタックサイズ UW epndptn 例外ペンディングパターン } 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 } T_RTSK;	
−	−		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);	
			iref_tst	ER ercd = iref_tst(ID tskid, T_RTST *pk_rtst);	

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。
- 変更内容には API レベルでの変更を記述しています。
- 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

	MR32R シリーズ（μITRON3.0 仕様準拠）			MR32R/4 シリーズ（μITRON4.0 仕様準拠）			変更内容
	機能			機能			
	SVC	API		SVC	API		
1	タスクを強制待ち状態を移行			強制待ち状態への移行			動作変更 3.3.3 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);		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			強制待ち状態からの再開			
	rsm_tsk	ER ercd = rsm_tsk (ID tskid);		rsm_tsk	ER ercd = rsm_tsk(ID tskid);		
	irsm_tsk	ER ercd = irsm_tsk (ID tskid);		irsm_tsk	ER ercd = irsm_tsk(ID tskid);		
―	―			強制待ち状態からの強制再開			新機能
	該当なし	―		frsm_tsk	ER ercd = frsm_tsk(ID tskid);		
				ifrsm_tsk	ER ercd = ifrsm_tsk(ID tskid);		
3	タスクを待ち状態へ移行			起床待ち			
	slp_tsk	ER ercd = slp_tsk ();		slp_tsk	ER ercd = slp_tsk();		
4	タスクを一定時間待ち状態へ移行			起床待ち(タイムアウト)			
	tslp_tsk	ER ercd = tslp_tsk (TMO tmout);		tslp_tsk	ER ercd = tslp_tsk(TMO tmout);		
5	待ち状態のタスクを起床			タスクの起床			動作変更 3.3.3 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);		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	
6	タスクの起床要求を無効			起床要求のキャンセル			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; 無効になった起床要求回数を格納する領域の先頭アドレス		引数	ID tskid; タスク ID 番号	
			ID tskid; タスク ID 番号				
		返値	ER ercd; エラーコード		返値	ER_UINT wupcnt; wupcnt>0 キャンセルされた起床要求カウント、wupcnt<0 エラーコード	
			INT *p_wupcnt 無効になった起床要求回数				

4.1.3 タスク例外処理

- 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。
- 変更内容には API レベルでの変更を記述しています。
- 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

MR32R シリーズ（μITRON3.0 仕様準拠）			MR32R/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);	3.2.2 参照
−	−		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);	3.2.2 参照
−	−		タスク例外処理の要求(ハンドラ専用)		新機能
	該当なし	−	iras_tex	ER ercd = iras_tex(ID tskid, TEXPTN rasptn);	3.2.2 参照
−	−		タスク例外処理の禁止		新機能
	該当なし	−	dis_tex	ER ercd = dis_tex();	3.2.2 参照
−	−		タスク例外処理の許可		新機能
	該当なし	−	ena_tex	ER ercd = ena_tex();	3.2.2 参照
−	−		タスク例外禁止状態の参照		新機能
	該当なし	−	sns_tex	ER ercd = sns_tex();	3.2.2 参照
−	−		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);	3.2.2 参照
−	−		タスク例外処理の状態参照(ハンドラ)		新機能
	該当なし	−	iref_tex	ER ercd = iref_tex(ID tskid, T_RTEX *pk_rtex);	3.2.2 参照

4.1.4 同期・通信機能(セマフォ)

- 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。
- 変更内容には API レベルでの変更を記述しています。
- 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

MR32R シリーズ（μITRON3.0 仕様準拠）			MR32R/4 シリーズ（μITRON4.0 仕様準拠）			変更内容
	機能			機能		
	SVC	API		SVC	API	
1	**セマフォを生成**			**セマフォの生成**		**パケット構造変更**
	cre_sem	ER ercd = cre_sem (ID semid, T_CSEM *pk_csem);		cre_sem	ER ercd = cre_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** INT maxsem; **セマフォの最大値** } T_CSEM;			**パケット構造** typedef struct t_csem { ATR sematr **セマフォ属性** UINT isemcnt **セマフォ資源数の初期値** UINT maxsem **セマフォ資源数の最大値** } T_CSEM;	
―	―			**セマフォの生成**(ID **自動割り当て**)		**新機能**
	該当なし	―		acre_sem	ER_ID semid = acre_sem(T_CSEM *pk_csem);	
2	**セマフォを削除**			**セマフォの削除**		
	del_sem	ER ercd = del_sem (ID semid);		del_sem	ER ercd = del_sem(ID semid);	
3	**セマフォ資源を返却**			**セマフォ資源の返却**		
	sig_sem	ER ercd = sig_sem (ID semid);		sig_sem	ER ercd = sig_sem(ID semid);	
4	**セマフォ資源を返却(ハンドラ専用)**			**セマフォ資源の返却(ハンドラ専用)**		
	isig_sem	ER ercd = isig_sem (ID semid);		isig_sem	ER ercd = isig_sem(ID semid);	
5	**セマフォ資源を獲得**			**セマフォ資源の獲得**		
	wai_sem	ER ercd = wai_sem (ID semid);		wai_sem	ER ercd = wai_sem(ID semid);	
6	**セマフォ資源を獲得(タイムアウトあり)**			**セマフォ資源の獲得(タイムアウト)**		
	twai_sem	ER ercd = twai_sem (ID semid, TMO tmout);		twai_sem	ER ercd = twai_sem(ID semid, TMO tmout);	
7	**セマフォ資源を獲得(ポーリング)**			**セマフォ資源の獲得(ポーリング)**		SVC **名変更**
	preq_sem	ER ercd = preq_sem (ID semid);		pol_sem	ER ercd = pol_sem(ID semid);	
				ipol_sem	ER ercd = ipol_sem(ID semid);	
8	**セマフォの状態を参照**			**セマフォの状態参照**		**パケット構造、引数並び変更**
	ref_sem	ER ercd = ref_sem (T_RSEM *pk_rsem, ID semid);		ref_sem	ER ercd = ref_sem(ID semid, T_RSEM *pk_rsem);	
				iref_sem	ER ercd = iref_sem(ID semid, T_RSEM *pk_rsem);	
		パケット構造 typedef struct t_rsem{ VP exinf; **拡張情報** BOOL_ID wtsk; **待ちタスクの有無** INT semcnt; **現在のセマフォカウント値** } 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タ値** } T_RSEM;	

4.1.5 同期・通信機能(イベントフラグ)

- 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。
- 変更内容には API レベルでの変更を記述しています。
- 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

	MR32R シリーズ（μITRON3.0 仕様準拠）			MR32R/4 シリーズ（μITRON4.0 仕様準拠）			変更内容
	機能			機能			
	SVC	API		SVC	API		
1	イベントフラグを生成			イベントフラグの生成			パケット構造　変更
	cre_flg	ER ercd = cre_flg (ID flgid,T_CFLG *pk_cflg);		cre_flg	ER ercd = cre_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 } 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;	
―	―			イベントフラグの生成(ID 自動割り当て)			新機能
	該当なし	―		acre_flg	ER_ID flgid = acre_flg(T_CFLG *pk_cflg);		
2	イベントフラグを削除			イベントフラグの削除			
	del_flg	ER ercd = del_flg (ID flgid);		del_flg	ER ercd = del_flg(ID flgid);		
3	イベントフラグをセット			イベントフラグのセット			引数型、動作変更 3.3.4 参照
	set_flg	ER ercd = set_flg (ID flgid, UINT setptn);		set_flg	ER ercd = set_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	
4	イベントフラグをセット（ハンドラ専用）			イベントフラグのセット(ハンドラ専用)			引数型、動作変更 3.3.4 参照
	iset_flg	ER ercd = iset_flg (ID flgid, UINT setptn);		iset_flg	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	UINT setptn; セットするビットパターン	
5	イベントフラグをクリア			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 setptn; クリアするビットパターン		引数	UINT setptn; クリアするビットパターン	
6	イベントフラグ待ち			イベントフラグ待ち			引数型、引数並び、動作変更 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 UINT waiptn; 待ちビットパターン UINT wfmode; 待ちモード		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 MODE waiptn; 待ちビットパターン FLGPTN wfmode; 待ちモード	
7	イベントフラグ待ち(タイムアウトあり)			イベントフラグ待ち(タイムアウト)			引数型、引数並び、動作変更 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 UINT waiptn; 待ちビットパターン UINT wfmode; 待ちモード		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 MODE waiptn; 待ちビットパターン FLGPTN wfmode; 待ちモード	

	MR32R シリーズ（μITRON3.0 仕様準拠）		MR32R/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
8	**イベントフラグを得ます(待ちなし)**		**イベントフラグ待ち(ポーリング)**		**引数型、引数並び、動作変更** 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** UINT waiptn; **待ちビットパターン** UINT wfmode; **待ちモード**		**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** MODE waiptn; **待ちビットパターン** FLGPTN wfmode; **待ちモード**	
9	**イベントフラグの状態を参照**		**イベントフラグの状態参照**		**引数型、引数並び変更**
	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);	
		引数 typedef struct t_rflg { VP exinf; **拡張情報** BOOL_ID wtsk; **待ちタスクの有無** UINT flgptn; **イベントフラグのビットパターン** } T_RFLG;		**引数** typedef struct t_rflg{ ID wtskid **待ちタスク** ID FLGPTN flgptn **現在のイベントフラグビットパターン** } T_RFLG;	

4.1.6 同期・通信機能(データキュー)

- 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。
- 変更内容には API レベルでの変更を記述しています。
- 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

MR32R シリーズ（μITRON3.0 仕様準拠）			MR32R/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 3.2.1 参照
	該当なし	−	cre_dtq	ER ercd = cre_dtq(ID dtqid, T_CDTQ *pk_cdtq);	
−	−		データキューの生成(ID 自動割り当て)		新機能 3.2.1 参照
	該当なし	−	acre_dtq	ER_ID dtqid = acre_dtq(T_CDTQ *pk_cdtq);	
−	−		データキューの削除		新機能 3.2.1 参照
	該当なし	−	del_dtq	ER ercd = del_dtq(ID dtqid);	
−	−		データキューへのデータ送信		新機能 3.2.1 参照
	該当なし	−	snd_dtq	ER ercd = snd_dtq(ID dtqid, VP_INT data);	
−	−		データキューへのデータ送信(ポーリング)		新機能 3.2.1 参照
	該当なし	−	psnd_dtq	ER ercd = psnd_dtq(ID dtqid, VP_INT data);	
−	−		データキューへのデータ送信(ハンドラ専用)		新機能 3.2.1 参照
	該当なし	−	ipsnd_dtq	ER ercd = ipsnd_dtq(ID dtqid, VP_INT data);	
−	−		データキューへのデータ送信(タイムアウト)		新機能 3.2.1 参照
	該当なし	−	tsnd_dtq	ER ercd = tsnd_dtq(ID dtqid, VP_INT data, TMO tmout);	
−	−		データキューへのデータ強制送信		新機能 3.2.1 参照
	該当なし	−	fsnd_dtq	ER ercd = fsnd_dtq(ID dtqid, VP_INT data);	
−	−		データキューへのデータ強制送信(ハンドラ専用)		新機能 3.2.1 参照
	該当なし	−	ifsnd_dtq	ER ercd = ifsnd_dtq(ID dtqid, VP_INT data);	
−	−		データキューからのデータ受信		新機能 3.2.1 参照
	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データ受信(ポーリング)		新機能 3.2.1 参照
	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データ受信(ハンドラ専用)		新機能 3.2.1 参照
	該当なし	−	iprcv_dtq	ER ercd = iprcv_dtq(ID dtqid, VP_INT *p_data);	
−	−		データキューからのデータ受信(タイムアウト)		新機能 3.2.1 参照
	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 3.2.1 参照
	該当なし	−	ref_dtq	ER ercd = ref_dtq(ID dtqid, T_RDTQ *pk_rdtq);	
−	−		データキューの状態参照(ハンドラ専用)		新機能 3.2.1 参照
	該当なし	−	iref_dtq	ER ercd = iref_dtq(ID dtqid, T_RDTQ *pk_rdtq);	

4.1.7 同期・通信機能(メールボックス)

- 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。
- 変更内容には API レベルでの変更を記述しています。
- 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

MR32R シリーズ（μITRON3.0 仕様準拠）			MR32R/4 シリーズ（μITRON4.0 仕様準拠）			変更内容
	機能			機能		
	SVC	API		SVC	API	
1	**メールボックスを生成**			**メールボックスの生成**		**パケット構造変更**
	cre_mbx	ER ercd = cre_mbx (ID mbxid, T_CMBX *pk_cmbx);		cre_mbx	ER ercd = cre_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; **メールボックス属性** INT bufcnt; **リングバッファの大きさ** VP mbx; **ユーザーが確保したメールボックス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;	
―	―			**メールボックスの生成**(ID **自動割り当て**)		**新機能**
	該当なし	―		acre_mbx	ER_ID mbxid = acre_mbx(T_CMBX *pk_cmbx);	
2	**メールボックスを削除**			**メールボックスの削除**		
	del_mbx	ER ercd = del_mbx (ID mbxid);		del_mbx	ER ercd = del_mbx(ID mbxid);	
3	**メッセージを送信**			**メールボックスへの送信**		SVC **名、メッセージヘッダ変更** 3.3.5 **参照**
	snd_msg	ER ercd = snd_msg (ID mbxid, T_MSG *pk_msg);		snd_mbx	ER ercd = snd_mbx(ID mbxid, T_MSG *pk_msg);	
		パケット構造 typedef UW T_MSG; **メッセージはパケットの先頭アドレス**			**パケット構造** 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）**	
		パケット構造 typedef void *PT_MSG; **メッセージは単なる** 32 **ビットデータ**			**パケット構造** 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）**	
4	**メッセージを送信(ハンドラ専用)**			**メールボックスへの送信(ハンドラ専用)**		SVC **名変更** **メッセージヘッダ変更** 3.3.5 **参照**
	isnd_msg	ER ercd = isnd_msg (ID mbxid, T_MSG *pk_msg);		isnd_mbx	ER ercd = isnd_mbx(ID mbxid, T_MSG *pk_msg);	
		パケット構造 typedef UW T_MSG; **メッセージはパケットの先頭アドレス**			**パケット構造** 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）**	
		パケット構造 typedef void *PT_MSG; **メッセージは単なる** 32 **ビットデータ**			**パケット構造** 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）**	

	MR32R シリーズ（μITRON3.0 仕様準拠）		MR32R/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
5	メッセージを受信		メールボックスからの受信		SVC 名、引数並び、メッセージヘッダ変更 3.3.5 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);	
	パケット構造	typedef UW T_MSG; メッセージはパケットの先頭アドレス	パケット構造	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 void *PT_MSG; メッセージは単なる 32 ビットデータ		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）	
6	メッセージを受信(タイムアウトあり)		メールボックスからの受信(タイムアウト)		SVC 名、引数並び、メッセージヘッダ変更 3.3.5 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);	
	パケット構造	typedef UW T_MSG; メッセージはパケットの先頭アドレス	パケット構造	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 void *PT_MSG; メッセージは単なる 32 ビットデータ		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	メッセージを受信(待ちなし)		メールボックスからの受信(ポーリング)		SVC 名、引数並び、メッセージヘッダ変更 3.3.5 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);	
	パケット構造	typedef UW T_MSG; メッセージはパケットの先頭アドレス	パケット構造	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 void *PT_MSG; メッセージは単なる 32 ビットデータ		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	メールボックスの状態を参照		メールボックスの状態参照		パケット構造、引数並び変更
	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; 待ちタスクの有無 T_MSG *pk_msg; 次に受信されるメッセージパケットの先頭アドレス INT msgcnt; メールボックスに蓄積されているメッセージの数 } T_RMBX;	パケット構造	typedef struct t_rmbx{ ID wtskid 受信待ちタスク ID T_MSG *pk_msg 次に受信されるメッセージパケット } T_RMBX;	

4.1.8 拡張同期・通信機能(メッセージバッファ)

- 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。
- 変更内容には API レベルでの変更を記述しています。
- 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

	MR32R シリーズ（μITRON3.0 仕様準拠）					MR32R/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を生成					メッセージバッファの生成					パケット構造変更
	cre_mbf	ER ercd = cre_mbf (ID mbfid, T_CMBF *pk_rmbf);				cre_mbf	ER ercd = cre_mbf(ID mbfid, T_CMBF *pk_cmbf);				
		パケット構造	typedef struct t_cmbf{				パケット構造	typedef struct t_cmbf {			
			VP	exinf;	拡張情報			ATR	mbfatr;	データキュー属性	
			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)	
			INT	bufsz;	メッセージバッファのサイズ			SIZE	mbfsz;	メッセージバッファ領域のサイズ(バイト数)	
			INT	maxmsz;	メッセージの最大長			VP	mbf;	メッセージバッファ領域の先頭アドレス	
			VP	mbf;	ユーザーが確保したメッセージバッファ領域の先頭アドレス			} T_CMBF;			
			} T_CMBF;								
−	−					メッセージバッファの生成(ID 自動割り当て)					新機能
	該当なし	−				acre_mbf	ER_ID mbfid = acre_mbf(T_CMBF *pk_cmbf);				
2	メッセージバッファを削除					メッセージバッファの削除					
	del_mbf	ER ercd = del_mbf (ID mbfid);				del_mbf	ER ercd = del_mbf(ID mbfid);				
3	メッセージバッファへ送信					メッセージバッファへの送信					引数型変更
	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		引数	UINT	msgsz;	送信するメッセージのサイズ	
4	メッセージバッファへ送信(タイムアウトあり)					メッセージバッファへの送信(タイムアウト)					引数型変更
	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		引数	UINT	msgsz;	送信するメッセージのサイズ	
5	メッセージバッファへ送信(待ちなし)					メッセージバッファへの送信(ポーリング)					引数型変更
	psnd_mbf	ER ercd = psnd_mbf (ID mbfid, VP msg, INT msgsz);				psnd_mbf	ER ercd = psnd_mbf(ID mbfid, VP msg, UINT msgsz);				
		引数	INT	msgsz;	送信メッセージのサイズ		引数	UINT	msgsz;	送信するメッセージのサイズ	
6	メッセージバッファから受信					メッセージバッファからの受信					引数、返値変更
	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;	受信メッセージを格納するアドレス		引数	ID	mbfid;	受信対象のメッセージバッファ ID 番号	
			INT	*p_msgsz;	受信メッセージのサイズ			VP	msg;	受信メッセージを格納した領域へのポインタ	
			ID	mbfid;	メッセージバッファ ID 番号						
		返値	ER	ercd;	エラーコード		返値	ER_UINT	msgsz;	受信したメッセージサイズ(正の値)、エラーコード(負の値)	
			INT	*p_msgsz;	受信メッセージのサイズ			VP	msg;	受信メッセージを格納した領域へのポインタ	
			VP	msg;	受信メッセージを格納するアドレス						

	MR32R シリーズ（μITRON3.0 仕様準拠）				MR32R/4 シリーズ（μITRON4.0 仕様準拠）				変更内容
	機能				機能				
	SVC	API			SVC	API			
7	**メッセージバッファから受信(タイムアウトあり)**				**メッセージバッファからの受信(タイムアウト)**				**引数、返値変更**
	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; **受信メッセージを格納するアドレス**		**引数**	ID	mbfid; **受信対象のメッセージバッファ ID 番号**	
			INT	*p_msgsz; **受信メッセージのサイズ**			TMO	tmout; **タイムアウト値**	
			ID	mbfid; **メッセージバッファ ID 番号**			VP	msg; **受信メッセージを格納した領域へのポインタ**	
			TMO	tmout **タイムアウト値**					
		返値	ER	ercd; **エラーコード**		**返値**	ER_UINT	msgsz; **受信したメッセージサイズ(正の値)、エラーコード(負の値)**	
			INT	*p_msgsz; **受信メッセージのサイズ**			VP	msg; **受信メッセージを格納した領域へのポインタ**	
			VP	msg; **受信メッセージを格納するアドレス**					
8	**メッセージバッファから受信(待ちなし)**				**メッセージバッファからの受信(ポーリング)**				**引数、返値変更**
	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; **受信メッセージを格納するアドレス**		**引数**	ID	mbfid; **受信対象のメッセージバッファ ID 番号**	
			INT	*p_msgsz; **受信メッセージのサイズ**			VP	msg; **受信メッセージを格納した領域へのポインタ**	
			ID	mbfid; **メッセージバッファ ID 番号**					
		返値	ER	ercd; **エラーコード**		**返値**	ER_UINT	msgsz; **受信したメッセージサイズ(正の値)、エラーコード(負の値)**	
			INT	*p_msgsz; **受信メッセージのサイズ**			VP	msg; **受信メッセージを格納した領域へのポインタ**	
			VP	msg; **受信メッセージを格納するアドレス**					
9	**メッセージバッファの状態を参照**				**メッセージバッファの状態参照**				**パケット構造、引数並び変更**
	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);			
		パケット構造	typedef struct t_rmbf {			**パケット構造**	typedef struct t_rmbf{		
			VP	exinf; **拡張情報**			ID	stskid; **送信待ちタスク ID**	
			BOOL_ID	wtsk; **受信待ちタスクの有無**			ID	rtskid; **受信待ちタスク ID**	
			BOOL_ID	stsk; **送信待ちタスクの有無**			UINT	smsgcnt; **メッセージバッファに入っているメッセージのカウント数**	
			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;		
		} T_RMBF;							

4.1.9 拡張同期・通信機能(ランデブ)

- 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。
- 変更内容には API レベルでの変更を記述しています。
- 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

	MR32R シリーズ（μITRON3.0 仕様準拠）				MR32R/4 シリーズ（μITRON4.0 仕様準拠）				変更内容
	機能				機能				
	SVC	API			SVC	API			
1	ランデブ用ポートを生成				ランデヴポートの生成				パケット構造変更
	cre_por	ER ercd = cre_por (ID porid,T_CPOR *pk_cpor);			cre_por	ER ercd = cre_por(ID porid, T_CPOR *pk_cpor);			
		パケット構造	typedef struct t_cpor {			パケット構造	typedef struct t_cpor {		
			VP	exinf; 拡張情報			ATR	poratr; ランデヴポート属性	
			ATR	poratr; ポート属性			UINT	maxcmsz; 呼び出しメッセージの最大サイズ(バイト数)	
			INT	maxcmsz; 呼出時のメッセージの最大数			UINT	maxrmsz; 返答メッセージの最大サイズ(バイト数)	
			INT	maxrmsz; 返答時のメッセージの最大数			} T_CPOR;		
			} T_CPOR;						
−	−				ランデヴポートの生成(ID 自動割り当て)				新機能
	該当なし	−			acre_por	ER_ID porid = acre_por(T_CPOR *pk_cpor);			
2	ランデブ用ポートを削除				ランデヴポートの削除				
	del_por	ER ercd = del_por (ID porid);			del_por	ER ercd = del_por(ID porid);			
3	ポートに対するランデブ呼出				ランデヴの呼び出し				引数、返値変更
	cal_por	ER ercd = cal_por (VP msg, INT *p_rmsgsz, ID porid, UINT calptn, INT cmsgsz);			cal_por	ER_UINT rmsgsz = cal_por(ID porid, RDVPTN calptn, VP msg, UINT cmsgsz);			
		引数	VP	msg; 呼出メッセージを格納したアドレス		引数	ID	porid; 呼び出し対象のランデヴポート ID 番号	
			INT	*p_rmsgsz; 返答メッセージのサイズを格納する領域のアドレス			RDVPTN	calptn; 呼び出し条件ビットパターン	
			ID	porid; ランデブ用ポート ID 番号			VP	msg; 呼び出しメッセージの先頭アドレス(返答メッセージを格納する先頭番地)	
			UINT	calptn; 呼出側選択条件を表わすビットパターン			UINT	cmsgsz; 呼び出しメッセージのサイズ(バイト数)	
			INT	cmsgsz; 呼出メッセージのサイズ					
		返値	ER	ecrd; エラーコード		返値	ER_UINT	rmsgsz; エラーコード	

	MR32R シリーズ（μITRON3.0 仕様準拠）					MR32R/4 シリーズ（μITRON4.0 仕様準拠）					変更内容
	機能					機能					
	SVC	API				SVC	API				
4	ポートに対するランデブ呼出(タイムアウトあり)					ランデヴの呼び出し(タイムアウト)					引数、返値、タイムアウト仕様変更 3.3.6 参照
	tcal_por	ER ercd = tcal_por (VP msg, INT *p_rmsgsz, ID porid, UINT calptn, INT cmsgsz, TMO tmout);				tcal_por	ER_UINT rmsgsz = tcal_por(ID porid, RDVPTN calptn, VP msg, UINT cmsgsz,TMO tmout);				
		引数	VP	msg;	呼出メッセージを格納したアドレス		引数	ID	porid;	呼び出し対象のランデヴポート ID 番号	
			INT	*p_rmsgsz;	返答メッセージのサイズを格納する領域のアドレス			RDVPTN	calptn;	呼び出し条件ビットパターン	
			ID	porid;	ランデブ用ポート ID 番号			VP	msg;	呼び出しメッセージの先頭アドレス(返答メッセージを格納する先頭番地)	
			UINT	calptn;	呼出側選択条件を表わすビットパターン			UINT	cmsgsz;	呼び出しメッセージのサイズ(バイト数)	
			INT	cmsgsz;	呼出メッセージのサイズ			TMO	tmout;	タイムアウト値(tcal_por の場合)	
			TMO	tmout;	タイムアウト値						
		返値	ER	ecrd;	エラーコード		返値	ER_UINT	rmsgsz;	エラーコード	
5	ポートに対するランデブ呼出(待ちなし)					−					廃止 3.3.6 参照
	pcal_por	ER ercd = pcal_por (VP msg, INT *p_rmsgsz, ID porid, UINT calptn, INT cmsgsz);				該当なし	−				
6	ポートに対するランデブ受付					ランデヴの受付					引数、返値変更
	acp_por	ER ercd = acp_por (RNO *p_rdvno, VP msg, INT *p_cmsgsz, ID porid, UINT acpptn);				acp_por	ER_UINT cmsgsz = acp_por(ID porid, RDVPTN acpptn, RDVNO *p_rdvno, VP msg);				
		引数	RNO	*p_rdvno;	ランデブ番号		引数	ID	porid;	受付対象のランデヴポート ID 番号	
			VP	msg;	呼出メッセージを格納する領域の先頭アドレス			RDVPTN	acpptn;	受付条件ビットパターン	
			INT	*p_cmsgsz;	呼出メッセージのサイズ			VP	msg;	呼び出しメッセージを格納する先頭アドレス	
			ID	porid;	ランデブ用ポート ID 番号			RDVNO	*p_rdvno;	成立したランデヴ番号へのポインタ	
			UINT	acpptn;	受付側選択条件を表わすビットパターン						
		返値	ER	ecrd;	エラーコード		返値	ER_UINT	cmsgsz;	エラーコード	
								RDVNO	*p_rdvno;	成立したランデヴ番号へのポインタ	

	MR32R シリーズ（μITRON3.0 仕様準拠）				MR32R/4 シリーズ（μITRON4.0 仕様準拠）				変更内容
	機能				機能				
	SVC	API			SVC	API			
7	ポートに対するランデブ受付(タイムアウトあり)				ランデヴの受付(タイムアウト)				引数、返値変更
	tacp_por	ER ercd = tacp_por (RNO *p_rdvno, VP msg, INT *p_cmsgsz, ID porid, UINT acpptn, TMO tmout);			tacp_por	ER_UINT cmsgsz = tacp_por(ID porid, RDVPTN acpptn, RDVNO *p_rdvno, VP msg, TMO tmout);			
		引数	RNO	*p_rdvno; ランデブ番号		引数	ID	porid; 受付対象のランデヴポート ID 番号	
			VP	msg; 呼出メッセージを格納する領域のアドレス			RDVPTN	acpptn; 受付条件ビットパターン	
			INT	*p_cmsgsz; 呼出メッセージのサイズ			VP	msg; 呼び出しメッセージを格納する先頭アドレス	
			ID	porid; ランデブ用ポート ID 番号			RDVNO	*p_rdvno; 成立したランデヴ番号へのポインタ	
			UINT	acpptn; 受付側選択条件を表わすビットパターン			TMO	tmout; タイムアウト値	
			TMO	tmout; タイムアウト値					
		返値	ER	ecrd; エラーコード		返値	ER_UINT	cmsgsz; エラーコード	
							RDVNO	*p_rdvno; 成立したランデヴ番号へのポインタ	
8	ポートに対するランデブ受付(待ちなし)				ランデヴの受付(ポーリング)				引数、返値変更
	pacp_por	ER ercd = pacp_por (RNO *p_rdvno, VP msg, INT *p_cmsgsz, ID porid, UINT acpptn);			pacp_por	ER_UINT cmsgsz = pacp_por(ID porid, RDVPTN acpptn, RDVNO *p_rdvno, VP msg);			
		引数	RNO	*p_rdvno; ランデブ番号		引数	ID	porid; 受付対象のランデヴポート ID 番号	
			VP	msg; 呼出メッセージを格納する領域の先頭アドレス			RDVPTN	acpptn; 受付条件ビットパターン	
			INT	*p_cmsgsz; 呼出メッセージのサイズ			VP	msg; 呼び出しメッセージを格納する先頭アドレス	
			ID	porid; ランデブ用ポート ID 番号			RDVNO	*p_rdvno; 成立したランデヴ番号へのポインタ	
			UINT	acpptn; 受付側選択条件を表わすビットパターン					
		返値	ER	ecrd; エラーコード		返値	ER_UINT	cmsgsz; エラーコード	
							RDVNO	*p_rdvno; 成立したランデヴ番号へのポインタ	
9	ランデブを別のポートに回送				ランデヴの回送				引数変更 3.3.6 参照
	fwd_por	ER ercd = fwd_por (ID porid, UINT calptn, RNO rdvno, VP msg, INT cmsgsz);			fwd_por	ER ercd = fwd_por(ID porid, RDVPTN calptn, RDVNO rdvno, VP msg, UINT cmsgsz);			
		引数	ID	porid; 回送先ポートの ID 番号		引数	ID	porid; 回送先のランデヴポート ID 番号	
			UINT	calptn; 呼出側選択条件を表わすビットパターン			RDVPTN	calptn; 呼び出し条件ビットパターン	
			RNO	rdvno; ランデブ番号			RDVNO	rdvno; 回送するランデヴ番号	
			VP	msg; 呼出メッセージを格納するアドレス			VP	msg; 呼び出しメッセージの先頭アドレス(返答メッセージを格納する先頭番地)	
			INT	cmsgsz; 呼出メッセージのサイズ			UINT	cmsgsz; 呼び出しメッセージのサイズ(バイト数)	

	MR32R シリーズ（μITRON3.0 仕様準拠）					MR32R/4 シリーズ（μITRON4.0 仕様準拠）					変更内容
	機能					機能					
	SVC	API				SVC	API				
10	ランデブ返答					ランデヴの終了					引数変更
	rpl_rdv	ER ercd = rpl_rdv (RNO rdvno, VP msg, INT rmsgsz);				rpl_rdv	ER ercd = rpl_rdv(RDVNO rdvno, VP msg, UINT rmsgsz);				
		引数	RNO	rdvno	ランデブ番号		引数	RDVNO	rdvno;	終了させるランデヴ番号	
			VP	msg	返答メッセージを格納するアドレス			VP	msg;	返答メッセージの先頭アドレス	
			INT	rmsgsz	返答メッセージのサイズ			UINT	rmsgsz;	返答メッセージのサイズ(バイト数)	
11	ランデブ用ポートを参照					ランデヴポートの状態参照					パケット構造、引数変更
	ref_por	ER ercd = ref_por (T_RPOR *pk_rpor, ID porid);				ref_por iref_por	ER ercd = ref_por(ID porid, T_RPOR *pk_rpor); ER ercd = iref_por(ID porid, T_RPOR *pk_rpor);				
		引数	T_RPOR	*pk_rpor;	ポートの状態を返す構造体の先頭アドレス		引数	ID	porid;	対象ランデヴポート ID 番号	
			ID	porid;	ランデブ用ポート ID 番号			T_RPOR	*pk_rpor;	ランデヴポート状態を返すパケットへのポインタ	
		パケット構造	typedef struct t_rpor{				パケット構造	typedef struct t_rpor{			
			VP	exinf;	拡張情報			ID	ctskid;	呼び出し待ちタスク ID	
			BOOL_ID	wtsk;	受付待ちタスクの有無			ID	atskid;	受付待ちタスク ID	
			BOOL_ID	atsk;	呼出待ちタスクの有無			} T_RPOR;			
			} T_RPOR								
−	−					ランデヴの状態参照					新機能
	該当なし	−				ref_rdv	ER ercd = ref_rdv(RDVNO rdvno, T_RRDV *pk_rrdv);				
−	−					ランデヴの状態参照(ハンドラ専用)					新機能
	該当なし	−				iref_rdv	ER ercd = iref_rdv(RDVNO rdvno, T_RRDV *pk_rrdv);				

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。
- 変更内容には API レベルでの変更を記述しています。
- 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

	MR32R シリーズ（μITRON3.0 仕様準拠）		MR32R/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
1	割り込みハンドラを定義		割り込みハンドラの定義		引数、パケット変更
	def_int	ER ercd = def_int (UINT dintno, T_DINT *pk_dint);	def_inh idef_inh	ER ercd = def_inh(INHNO inhno, T_DINH *pk_dinh); ER ercd = idef_inh(INHNO inhno, T_DINH *pk_dinh);	
		引数 UINT dintno; 定義する割り込みベクタ番号 T_DINT *pk_dint; 割り込み定義情報		引数 INHNO inhno; 定義する割り込ハンドラ番号 T_DINH *pk_dinh; 割り込みハンドラ定義情報を格納したパケットへのポインタ	
		パケット typedef struct t_dint { ATR intatr; 割り込みハンドラ属性 FP inthdr; 割り込みハンドラアドレス } T_DINT;		パケット typedef struct t_dinh { ATR inhatr; 割り込みハンドラ属性 FP inthdr; 割り込みハンドラエントリアドレス } T_DINH;	
2	割り込みハンドラから復帰		－		廃止（アセンブラで記述のみ ret_int が必要）
	ret_int	void ret_int ();	該当なし	－	
3	割り込みとディスパッチを禁止		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 ();	loc_cpu iloc_cpu	ER ercd = loc_cpu(); ER ercd = iloc_cpu();	
4	割り込みとディスパッチを許可		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 ();	unl_cpu iunl_cpu	ER ercd = unl_cpu(); ER ercd = iunl_cpu();	

4.1.11 固定長メモリプール管理機能

- 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。
- 変更内容には API レベルでの変更を記述しています。
- 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

	MR32R シリーズ（μITRON3.0 仕様準拠）		MR32R/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を生成**		**固定長メモリプールの生成**		**パケット構造変更**
	cre_mpf	ER ercd = cre_mpf (ID mpfid, T_CMPF *pk_cmpf); **パケット構造** typedef struct t_cmpf { VP exinf; **拡張情報** ATR mpfatr; **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** VP mpf; **ユーザーが確保したメモリプール領域の先頭アドレス** } T_CMPF;	cre_mpf	ER ercd = cre_mpf(ID mpfid, T_CMPF *pk_cmpf); **パケット構造** typedef struct t_cmpf { ATR mpfatr; **固定長メモリプール属性** UINT blkcnt; **獲得可能なメモリブロック数(個数)** UINT blksz; **メモリブロックのサイズ(バイト数)** VP mpf; **固定長メモリプール領域の先頭アドレス** } T_CMPF;	
–	–		**固定長メモリプールの生成(ID 自動割り当て)**		**新機能**
	該当なし	–	acre_mpf	ER_ID mpfid = acre_mpf(T_CMPF *pk_cmpf);	
2	**固定長メモリプールを削除**		**固定長メモリプールの削除**		
	del_mpf	ER ercd = del_mpf (ID mpfid);	del_mpf	ER ercd = del_mpf(ID mpfid);	
3	**メモリブロックを獲得**		**固定長メモリブロックの獲得**		SVC **名、引数変更**
	get_blf	ER ercd = get_blf (VP *p_blf, ID mpfid); **引数** VP *p_blf; **獲得したメモリブロックの先頭アドレスを格納する領域のアドレス** ID mpfid; **メモリプール ID 番号**	get_mpf	ER ercd = get_mpf(ID mpfid, VP *p_blk); **引数** ID mpfid; **獲得対象の固定長メモリプール ID 番号** VP *p_blk; **獲得したメモリブロック先頭アドレスへのポインタ**	
4	**メモリブロックを獲得(タイムアウトあり)**		**固定長メモリブロックの獲得(タイムアウト)**		SVC **名、引数変更**
	tget_blf	ER ercd = tget_blf (VP *p_blf, ID mpfid, TMO tmout); **引数** VP *p_blf; **獲得したメモリブロックの先頭アドレスを格納する領域のアドレス** ID mpfid; **メモリプール ID 番号** TMO tmout **タイムアウト値**	tget_mpf	ER ercd = tget_mpf(ID mpfid, VP *p_blk,TMO tmout); **引数** ID mpfid; **獲得対象の固定長メモリプール ID 番号** VP *p_blk; **獲得したメモリブロック先頭アドレスへのポインタ** TMO tmout; **タイムアウト値**	
5	**メモリブロックを獲得(待ちなし)**		**固定長メモリブロックの獲得(ポーリング)**		SVC **名、引数変更**
	pget_blf	ER ercd = pget_blf (VP *p_blf, ID mpfid); **引数** ID mpfid; **メモリプール ID 番号** VP *p_blf; **獲得したメモリブロックの先頭アドレスを格納する領域のアドレス**	pget_mpf ipget_mpf	ER ercd = pget_mpf(ID mpfid, VP *p_blk); ER ercd = ipget_mpf(ID mpfid, VP *p_blk); **引数** ID mpfid; **獲得対象の固定長メモリプール ID 番号** VP *p_blk; **獲得したメモリブロック先頭アドレスへのポインタ**	

	MR32R シリーズ（μITRON3.0 仕様準拠）		MR32R/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
6	**メモリブロックを解放**		**固定長メモリブロックの解放**		SVC 名、引数変更
	rel_blf	ER ercd = rel_blf (ID mpfid, VP p_blf); **引数** ID mpfid; **メモリプール ID 番号** VP p_blf; **解放するメモリブロックの先頭アドレス**	rel_mpf	ER ercd = rel_mpf(ID mpfid, VP blk); **引数** ID mpfid; **解放するメモリブロックの固定長メモリプール ID 番号** VP blk; **返却するメモリブロックの先頭アドレス**	
7	**メモリブロックを解放(ハンドラ専用)**		**固定長メモリブロックの解放(ハンドラ専用)**		SVC 名、引数変更
	irel_blf	ER ercd = irel_blf (ID mpfid, VP p_blf); **引数** ID mpfid; **メモリプール ID 番号** VP p_blf; **解放するメモリブロックの先頭アドレス**	irel_mpf	ER ercd = irel_mpf(ID mpfid, VP blk); **引数** ID mpfid; **解放するメモリブロックの固定長メモリプール ID 番号** VP blk; **返却するメモリブロックの先頭アドレス**	
8	**固定長メモリプールの状態を参照**		**固定長メモリプールの状態参照**		**引数並び、パケット構造変更**
	ref_mpf	ER ercd = ref_mpf (T_RMPF *pk_rmpf, ID mpfid); **引数** typedef struct t_rmpf { VP exinf; **拡張情報** BOOL_ID wtsk; **待ちタスクの有無** INT frbcnt; **空き領域のブロック数** INT blksz; **ブロックサイズ** } T_RMPF;	ref_mpf iref_mpf	ER ercd = ref_mpf(ID mpfid, T_RMPF *pk_rmpf); ER ercd = iref_mpf(ID mpfid, T_RMPF *pk_rmpf); **引数** typedef struct t_rmpf{ ID wtskid; **メモリブロック獲得待ちタスク ID** UINT fblkcnt; **空きメモリブロック数(個数)** } T_RMPF;	

4.1.12 可変長メモリプール管理機能

- 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。
- 変更内容には API レベルでの変更を記述しています。
- 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

	MR32R シリーズ（μITRON3.0 仕様準拠）					MR32R/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を生成**					**可変長メモリプールの生成**					**パケット構造変更**
	cre_mpl	ER ercd = cre_mpl (ID mplid, T_CMPL *pk_cmpl);				cre_mpl	ER ercd = cre_mpl(ID mplid, T_CMPL *pk_cmpl);				
		パケット構造	typedef struct t_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**	
			ATR	mplatr;	**メモリプール属性**			SIZE	mplsz	**確保するメモリプールサイズ**	
			INT	mplsz;	**メモリプール全体のサイズ**			UINT	maxblksz	**獲得するメモリサイズのうち最大サイズ**	
			INT	maxblksz;	**獲得するメモリブロックの最大サイズ**			VP	mpl	**可変長メモリプール領域の先頭アドレス**	
			VP	mpf;	**ユーザーが確保したメモリプール領域の先頭アドレス**			} T_CMPL;			
			} T_CMPL;								
–	–					**可変長メモリプールの生成**(ID **自動割り当て**)					**新機能**
	該当なし	–				acre_mpl	ER_ID mplid = acre_mpl(T_CMPL *pk_cmpl);				
2	**可変長メモリプールを削除**					**可変長メモリプールの削除**					
	del_mpl	ER ercd = del_mpl (ID mplid);				del_mpl	ER ercd = del_mpl(ID mplid);				
3	**メモリブロックを獲得**					**可変長メモリブロックの獲得**					SVC **名、引数並び、引数型変更**
	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);				
		引数	INT	blksz;	**獲得するメモリのサイズ**		**引数**	UINT	blksz;	**獲得するメモリのサイズ(バイト数)**	
4	**メモリブロックを獲得(タイムアウトあり)**					**可変長メモリブロックの獲得(タイムアウト)**					SVC **名、引数並び、引数型変更**
	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);				
		引数	INT	blksz;	**獲得するメモリのサイズ**		**引数**	UINT	blksz;	**獲得するメモリのサイズ(バイト数)**	
5	**メモリブロックを獲得(待ちなし)**					**可変長メモリブロックの獲得(ポーリング)**					SVC **名、引数並び、引数型変更**
	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);				
		引数	INT	blksz;	**獲得するメモリのサイズ**		**引数**	UINT	blksz;	**獲得するメモリのサイズ(バイト数)**	
6	**メモリブロックを解放**					**可変長メモリプールブロックの解放**					SVC **名変更**
	rel_blk	ER ercd = rel_blk (ID mplid, VP blk);				rel_mpl	ER ercd = rel_mpl(ID mplid, VP blk);				
7	**可変長メモリプールの状態を参照**					**可変長メモリプールの状態参照**					**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	ER ercd = ref_mpl(ID mplid, T_RMPL *pk_rmpl);				
						iref_mpl	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	**メモリ獲得待ちタスク** ID	
			BOOL_ID	wtsk;	**待ちタスクの有無**			SIZE	fmplsz	**空きメモリサイズ(バイト数)**	
			INT	frsz;	**空き領域の合計サイズ**			UINT	fblksz	**すぐに獲得可能なメモリの最大サイズ(バイト数)**	
			INT	maxsz;	**空き領域の最大サイズ**			} T_RMPL;			
			} 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。
- 変更内容には API レベルでの変更を記述しています。
- 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

MR32R シリーズ（μITRON3.0 仕様準拠）			MR32R/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能			機能	
	SVC	API	SVC	API	
1	**システムクロックを設定**		**システム時刻の設定**		**引数、パケット変更** 3.3.7 **参照**
	set_tim	ER ercd = set_tim (SYSTIME *pk_tim);	set_tim iset_tim	ER ercd = set_tim(SYSTIM *p_systim); ER ercd = iset_tim(SYSTIM *p_systim);	
		引数 SYSTIME *pk_tim; **設定するシステム時刻データの先頭アドレス**		**引数** SYSTIM *p_systim **設定するシステム時刻を示すパケットへのポインタ**	
		パケット typedef struct t_systime{ H utime; **上位 16 ビット** UW ltime; **下位 32 ビット** } SYSTIME;		**パケット** typedef struct t_systim { UH utime **上位 16bit** UW ltime **下位 32bit** } SYSTIM;	
2	**システムクロックを参照**		**システム時刻の取得**		**引数、パケット変更** 3.3.7 **参照**
	get_tim	ER ercd = get_tim (SYSTIME *pk_tim);	get_tim iget_tim	ER ercd = get_tim(SYSTIM *p_systim); ER ercd = iget_tim(SYSTIM *p_systim);	
		引数 SYSTIME *pk_tim; **設定するシステム時刻データの先頭アドレス**		**引数** SYSTIM *p_systim **設定するシステム時刻を示すパケットへのポインタ**	
		パケット typedef struct t_systime{ H utime; **上位 16 ビット** UW ltime; **下位 32 ビット** } SYSTIME;		**パケット** typedef struct t_systim { UH utime **上位 16bit** UW ltime **下位 32bit** } SYSTIM;	
−	−		**タイムティックの供給**		**新機能**
	該当なし	−	isig_tim	**カーネルに組み込まれている**	
3	**タスクの実行を一定時間遅延**		**タスクの遅延**		**引数変更** 3.3.7 **参照**
	dly_tsk	ER ercd = dly_tsk (DLYTIMe dlytim);	dly_tsk	ER ercd = dly_tsk(TMO tmout);	
		引数 DLYTIME dlytim; **遅延時間**		**引数** TMO tmout **遅延時間**	
4	**周期起動ハンドラを定義**		**周期ハンドラの生成**		**全般的に機能変更** 3.3.8 **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);	
−	−		**周期ハンドラの生成(ID 自動割り当て)**		
	該当なし	−	acre_cyc	ER_ID cycid = acre_cyc(T_CCYC *pk_ccyc);	
−	−		**周期ハンドラの削除**		
	該当なし	−	del_cyc	ER ercd = del_cyc(ID cycid);	
5	**周期起動ハンドラの活性制御を行います**		**周期ハンドラの動作開始**		
	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);	
			周期ハンドラの動作停止		
			stp_cyc	ER ercd = stp_cyc(ID cycid);	
			istp_cyc	ER ercd = istp_cyc(ID cycid);	
6	**周期起動ハンドラの状態を参照**		**周期ハンドラの状態参照**		
	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);	

MR32R シリーズ（μITRON3.0 仕様準拠）			MR32R/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
−	−		アラームハンドラの生成		全般的に機能変更 3.3.9 参照
	該当なし	−	cre_alm	ER ercd = cre_alm(ID almid, T_CALM *pk_calm);	
−	−		アラームハンドラの生成(ID 自動割り当て)		
	該当なし	−	acre_alm	ER_ID almid = acre_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);	
7	アラームハンドラの状態を参照		アラームハンドラの状態参照		
	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);	

4.1.14 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。
- 変更内容には API レベルでの変更を記述しています。
- 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

	MR32R シリーズ（μITRON3.0 仕様準拠）			MR32R/4 シリーズ（μITRON4.0 仕様準拠）			変更内容
	機能			機能			
	SVC	API		SVC	API		
1	OS のバージョンを参照			バージョン情報の参照			引数、パケット変更
	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);		
		引数	T_VER *pk_ver; バージョン管理情報を返す構造体の先頭アドレス		引数	T_RVER *pk_rver バージョン情報を返すパケットへのポインタ	
		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 カーネルのメーカーコード UH prid カーネルの識別番号 UH spver ITRON 仕様のバージョン番号 UH prver カーネルのバージョン番号 UH prno[4] カーネル製品の管理情報 } T_RVER;	
2	システムの状態を参照			−			廃止 3.1.2 参照
	ref_sys	ER ercd = ref_sys (T_RSYS *pk_rsys);		該当なし	−		
3	例外ハンドラの定義を行います			−			廃止 3.1.2 参照
	def_exc	ER ercd = def_exc (UINT exckind, T_DEXC *pk_dexc);		該当なし	−		

4.1.15 システム状態機能

MR32R シリーズ（μITRON3.0 仕様準拠）			MR32R/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	ER ercd = sns_ctx();	
−	−		CPU ロック状態の参照		新機能
	該当なし	−	sns_loc	ER ercd = sns_loc();	
−	−		ディスパッチ禁止状態の参照		新機能
	該当なし	−	sns_dsp	ER ercd = sns_dsp();	
−	−		ディスパッチ保留状態の参照		新機能
	該当なし	−	sns_dpn	ER ercd = sns_dpn();	

4.1.16 拡張機能

- 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。
- 変更内容には API レベルでの変更を記述しています。
- 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

	MR32R シリーズ（μITRON3.0 仕様準拠）		MR32R/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
1	**例外マスクをクリア**		**－**		**廃止** 3.1.2 **参照**
	vclr_ems	ER ercd = vclr_ems (ID tskid);	**該当なし**	**－**	
2	**例外マスクをセット**		**－**		**廃止** 3.1.2 **参照**
	vset_ems	ER ercd = vset_ems (ID tskid);	**該当なし**	**－**	
3	**強制例外ハンドラを起動**		**－**		**廃止** 3.1.2 **参照**
	vras_fex	ER ercd = vras_fex (ID tskid, UW exccd);	**該当なし**	**－**	
－	**－**		**データキュー領域の初期化**		**新機能**
	該当なし	**－**	vrst_dtq	ER ercd = vrst_dtq(ID dtqid);	
4	**メールボックスをクリア**		**メールボックス領域の初期化**		SVC **名変更**
	vrst_msg	ER ercd = vrst_msg (ID mbxid);	vrst_mbx	ER ercd = vrst_mbx(ID mbxid);	
5	**メッセージバッファをクリア**		**メッセージバッファ領域の初期化**		
	vrst_mbf	ER ercd = vrst_mbf (ID mbfid);	vrst_mbf	ER ercd = vrst_mbf(ID mbfid);	
6	**固定長メモリプールを解放**		**固定長メモリプール領域の初期化**		SVC **名変更**
	vrst_blf	ER ercd = vrst_blf (ID blfid);	vrst_mpf	ER ercd = vrst_mpf(ID mpfid);	
7	**可変長メモリプールを解放**		**可変長メモリプール領域の初期化**		SVC **名変更**
	vrst_blk	ER ercd = vrst_blk (ID blkid);	vrst_mpl	ER ercd = vrst_mpl(ID mplid);	

4.1.17 拡張機能(優先度付き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。
- 変更内容には API レベルでの変更を記述しています。
- 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

	MR32R シリーズ（μITRON3.0 仕様準拠）		MR32R/4 シリーズ（μITRON4.0 仕様準拠）		変更内容
	機能		機能		
	SVC	API	SVC	API	
1	優先度付きメールボックスを生成		−		廃止メールボックス機能で代用可 3.1.1 参照
	vcre_mbx	ER ercd = vcre_mbx (ID vmbxid, T_CVMBX *pk_rmbf);	該当なし	−	
2	優先度付きメールボックスを削除		−		
	vdel_mbx	ER ercd = vdel_mbx (ID vmbxid);	該当なし	−	
3	優先度付きメッセージを送信		−		
	vsnd_mbx	ER ercd = vsnd_mbx (ID vmbxid, T_MSG pk_msg);	該当なし	−	
4	優先度付きメッセージを送信(ハンドラ専用)		−		
	visnd_mbx	ER ercd = visnd_mbx (ID vmbxid, T_MSG pk_msg);	該当なし	−	
5	優先度付きメッセージを受信(タイムアウトなし)		−		
	vrcv_mbx	ER ercd = vrcv_mbx (T_MSG *ppk_msg, ID vmbxid);	該当なし	−	
6	優先度付きメッセージを受信(タイムアウトあり)		−		
	vtrcv_mbx	ER ercd = vtrcv_mbx (T_MSG *ppk_msg, ID vmbxid, TMO tmout);	該当なし	−	
7	優先度付きメッセージを受信(まちなし)		−		
	vprcv_mbx	ER ercd = vprcv_mbx (T_MSG *ppk_msg, ID vmbxid);	該当なし	−	
8	優先度付きメールボックスをクリア		−		
	vrst_mbx	ER ercd = vref_mbx (T_RVMBF *pk_rvmbf,ID vmbxid);	該当なし	−	
9	優先度付きメールボックスの状態を参照		−		
	vref_mbx	ER ercd = vrst_mbx (ID vmbxid);	該当なし	−	

4.2 エラーコード相違点

MR32R と MR32R/4 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。

表 4-1 エラーコード一覧比較表

番号	MR32R		MR32R/4		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	00000000H	E_OK	00000000H	正常終了
2	E_OBJ	0FFFFFFC1H	E_OBJ	0FFFFFFD7H	オブジェクトの状態が不正
3	E_QOVR	0FFFFFFB7H	E_QOVR	0FFFFFFD5H	キューイングまたはネストのオーバーフロー
4	E_TMOUT	0FFFFFFABH	E_TMOUT	0FFFFFFCEH	ポーリング失敗またはタイムアウト
5	E_RLWAI	0FFFFFFAAH	E_RLWAI	0FFFFFFCFH	待ち状態強制解除
6	E_NOEXS	0FFFFFFCCH	E_NOEXS	0FFFFFFD6H	オブジェクトが存在していない
7	E_DLT	0FFFFFFAFH	E_DLT	0FFFFFFCDH	待ちオブジェクトが削除された
8	E_NOMEM	0FFFFFFF6H	E_NOMEM	0FFFFFFDFH	メモリ不足
9	EV_RST	0FFFFFF02H	EV_RST	0FFFFFF81H	リセットのため待ちが解除された
10	−	−	E_ILUSE	0FFFFFFE4H	サービスコール不正使用
11	−	−	E_NOID	0FFFFFFDEH	割り当て可能な ID 番号がない

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。

表 4-2 新設、廃止のエラーコード

番号	サービスコール	MR32R でのエラーコード	MR32R/4 でのエラーコード	意味
1	ter_tsk	−	E_ILUSE	対象タスクが、自タスク ID または、TSK_SELF
2	wai_flg	−	E_ILUSE	サービスコール不正使用 (TA_WSGL 属性のイベントフラグに待ちタスクが存在)
3	twai_flg	−	E_ILUSE	サービスコール不正使用 (TA_WSGL 属性のイベントフラグに待ちタスクが存在)
4	pol_flg	−	E_ILUSE	サービスコール不正使用 (TA_WSGL 属性のイベントフラグに待ちタスクが存在)
5	cre_mbx	E_NOMEM	−	メモリ不足
6	snd_msg	E_QOVR	−	キューイングまたはネストのオーバーフロー
7	isnd_msg	E_QOVR	−	キューイングまたはネストのオーバーフロー

5. コンフィギュレーション方法の相違

MR32R では、テキストエディタなどでコンフィギュレーションファイルを作成する必要がありました。MR32R/4 では容易にコンフィギュレーションファイルを作成できるようにコンフィギュレーションファイルを作成するための GUI ツールを用意しました。

詳細は MR32R/4 の GUI コンフィギュレータヘルプファイルをご覧ください。

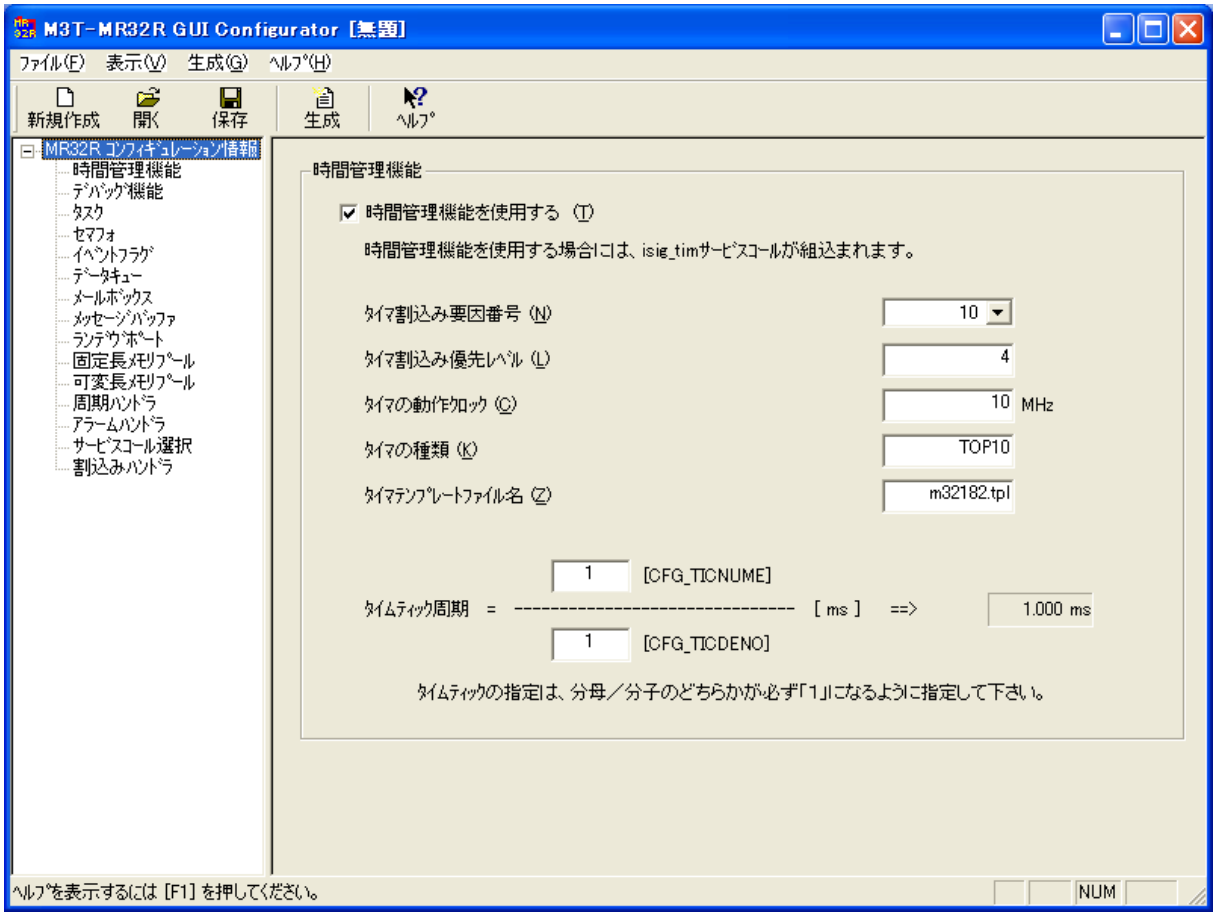

図 5-1　MR32R/4 のコンフィギュレータ概観

MR32RからMR32R/4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
MR32R から MR32R/4 への移行ガイド

ルネサス エレクトロニクス株式会社

RJJ05B0776-0100