User's Manual

R-IN32M4-CL3

プログラミング・マニュアル　（OS 編）

本資料に記載の全ての情報は本資料発行時点のものであり、ルネサス エレクトロニクスは、
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 エレクトロニクスのホームページなどにより公開される最新情報をご確認ください。

資料番号：R18UZ0072JJ0100
発行年月：2019.10.30

ルネサス エレクトロニクス
www.renesas.com

ご注意書き

1. 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・システムの設計において、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て、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。
2. 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。
3. 本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害に関し、当社は、何らの責任を負うものではありません。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。
4. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   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、産業用ロボット等
   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、各種安全装置等

   当社製品は、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等） 、もしくは多大な物的損害を発生させるおそれのある機器・システム（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。 たとえ、意図し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損害が生じても、当社は一切その責任を負いません。 なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の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。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、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、社会的損害等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、お客様の機器・システムとしての出荷保証を行ってください。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、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の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。
9. 本資料に記載されている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・システムに使用することはできません。また、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に使用しないでください。当社製品または技術を輸出する場合は、「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。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
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禁じます。

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。

製品ご使用上の注意事項

ここでは、CMOS デバイスの一般的注意事項について説明します。個別の使用上の注意事項については、本文を参照してください。なお、本マニュアルの本文と異なる記載がある場合は、本文の記載が優先するものとします。

1. 未使用端子の処理
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】電源投入時は、製品の状態は不定です。
   電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。
   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. リザーブアドレスのアクセス禁止
   【注意】リザーブアドレスのアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレスがあります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. クロックについて
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。
   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。

このマニュアルの使い方

1. 目的と対象者

このマニュアルはイーサネット通信 LSI「R-IN32M4-CL3」の機能を理解し、それを用いた応用設計をするユーザを対象とします。このマニュアルを使用するには、電気回路、論理回路マイクロコンピュータに関する基本的な知識が必要です。

本製品は、注意事項を十分確認の上、使用してください。注意事項は、各章の本文中、各章の最後、注意事項の章に記載しています。

改訂記録は旧版の記載内容に対して訂正または追加した主な箇所をまとめたものです。改訂内容すべてを記録したものではありません。詳細は、このマニュアルの本文でご確認ください。
本文中の★印は、本版で改訂された主な箇所を示しています。この"★"を PDF 上でコピーして「検索する文字列」に指定することによって、改版箇所を容易に検索できます

関連資料　関連資料は暫定版の場合がありますが、この資料では「暫定」の表示をしておりません。あらかじめご了承ください。また各コアの開発・企画段階で資料を作成しているため、関連資料は個別のお客様向け資料の場合があります。下記資料番号の末尾****部分は版数です。当社ホームページより最新版を参照してください。

R-IN32M4-CL3に関する資料

資料名	資料番号
R-IN32M4-CL3　ユーザーズ・マニュアル	R18UZ0073JJ****
R-IN32M4-CL3　プログラミング・マニュアル（ドライバ編）	R18UZ0076JJ****
R-IN32M4-CL3　プログラミング・マニュアル（OS 編）	このマニュアル

OSに関する資料

資料名	資料番号
μITRON4.0 仕様　Ver.4.02.00　（(社)トロン協会　ITRON 仕様検討グループ）	-

μITRON4.0仕様は、トロン協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。
本書にて記すμITRON4.0仕様は、μITRON4.0仕様の仕様書からの抜粋となっております。仕様の全容につきましては、μITRON4.0仕様の仕様書を参照してください。
なお、μITRON仕様の仕様書は、トロン協会のホームページから入手することができます。

2. 数や記号の表記

データ表記の重み：左が上位桁、右が下位桁
アクティブ・ローの表記：
　　xxxZ（端子、信号名称のあとにZ ）
　　またはxxx_N (端子、信号名称のあとに_N ）
　　またはxxnx (端子、信号名称にnを含む）
注：
　　本文中につけた注の説明
注意：
　　気をつけて読んでいただきたい内容
備考：
　　本文の補足説明
数の表記：
　　2 進数 … xxxx , xxxxBまたはn’bxxxx(nビット)
　　10 進数 … xxxx
　　16 進数 … xxxxHまたはn’hxxxx(nビット)

2のべき数を示す接頭語（アドレス空間、メモリ容量）：
　　K （キロ）… $2^{10} = 1024$
　　M （メガ）… $2^{20} = 1024^2$
　　G （ギガ）… $2^{30} = 1024^3$
データ・タイプ：
　　ワード … 32 ビット
　　ハーフワード … 16 ビット
　　バイト … 8 ビット

目次

図の目次

表の目次

1.　概説

本書では、イーサネット通信 LSI「R-IN32M4-CL3」にてリアルタイム OS（μITRON Ver. 4.0）を利用するための手順、および対応しているサービスコールについて説明します。

ハードウェア・リアルタイム OS と OS ライブラリを組み合わせることで、RTOS の機能を無償で実現できます。ライセンス料も保守費用も不要です。（ただし、無保証）

1.1　ハードウェア・リアルタイム OS の特徴

R-IN32M4-CL3 は、リアルタイム OS の処理を高速化するハードウェア・リアルタイム OS (HW-RTOS) を搭載しています。ハードウェア・リアルタイム OS は、タスクやイベントフラグ等のオブジェクトやタスク・スケジューリングをハードウェアにて処理するため、応答性の良い OS 処理が可能となっています。

設定可能なオブジェクトの個数を表 1.1 に示します。ハードウェア・リアルタイム OS では、セマフォとミューテックスを同一のハードウェアにて実現しているため、設定可能個数はあわせて 128 となります。例えば、セマフォ用として 100 個使用したとすると、ミューテックス用に使用できる個数は 28 個となります。尚、セマフォとミューテックスは同じ ID を使用することはできません（セマフォで ID の 1 ～10 を使用した場合、ミューテックスで使用可能な ID は、11～になります）。

表 1.1　設定可能なオブジェクト数

オブジェクト	設定可能個数
タスク数	64
イベントフラグ数	64
メールボックス数	64
セマフォ数	あわせて128
ミューテックス数	

また、ハードウェア・リアルタイム OS の特徴として、ハードウェア割り込みサービスルーチン(以下、Hardware ISR)機能があります。

Hardware ISR 機能とは、割り込みサービスルーチンと、そのルーチン内で実行されるサービスコールの一部をハードウェア化したものです。Hardware ISR 機能を使用すると、割り込みが発生したときに、事前に登録しておいたサービスコールをハードウェア・リアルタイム OS が自動実行することができます。例えば、割り込み処理内で set_flg のみを行っている場合には、代わりに本機能を使用することによって、CPU による割り込み処理を介さずに set_flg を実行することが可能になります。また、同サービスコール実行に伴うタスク・スケジューリングもハードウェア・リアルタイム OS が行うので、割り込み応答性の良いサービスコールを実現できます。

表 1.2 に Hardware ISR に設定できるサービスコール一覧を示します。設定方法につきましては、6. Hardware ISR の章を参照してください。

表 1.2　Hardware ISR に設定できるサービスコール一覧

サービスコール名	機能
set_flg	イベントフラグのセット
sig_sem	セマフォ資源の返却
rel_wai	待ち状態の強制解除
wup_tsk	タスクの起床

1.2　OS ライブラリ

OS ライブラリは、HW-RTOS を制御するソフトウェアで、μITRON のサービスコールを提供します。
このドキュメントでは、HW-RTOS+OS ライブラリで実現する RTOS の API 仕様について記載します。

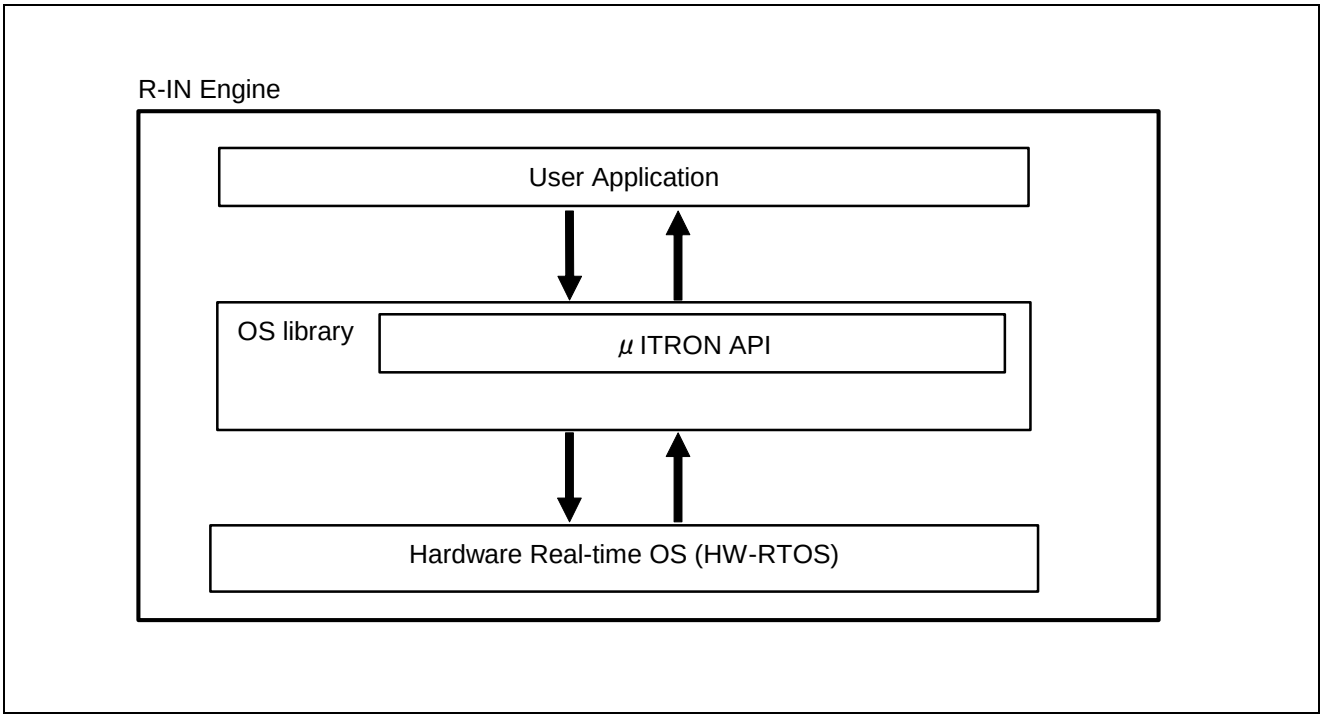

図 1.1　システム構成図

1.2.1　OS ライブラリのバージョン

本書が対象とする OS ライブラリのバージョンは以下です。

OSライブラリのファイル名	バージョン
libos.a	2.03

1.3　対応サービスコール一覧

スタンダードプロファイルに対し、R-IN32M4 にて対応するサービスコールの一覧を以下に記します。

表 1.3　対応サービスコール一覧(1/3)

項目		名称	機能	R-IN32M4 μITRON	μITRON Ver. 4.0 スタンダード プロファイル
タスク管理機能		act_tsk	タスクの起動	－	○
		iact_tsk	タスクの起動	－	○
		can_act	タスク起動要求のキャンセル	－	○
		sta_tsk	タスクの起動（起動コード指定）	○	－
		ext_tsk	自タスクの終了	○	○
		ter_tsk	タスクの強制終了	○	○
		chg_pri	タスク優先度の変更	○	○
		get_pri	タスク優先度の参照	○	○
タスク付属同期機能		slp_tsk	起床待ち	○	○
		tslp_tsk	起床待ち（タイムアウトあり）	○	○
		wup_tsk	タスクの起床	○	○
		iwup_tsk	タスクの起床	○	○
		can_wup	タスク起床要求のキャンセル	○	○
		rel_wai	待ち状態の強制解除	○	○
		irel_wai	待ち状態の強制解除	○	○
		sus_tsk	強制待ち状態への移行	－	○
		frsm_tsk	強制待ち状態からの強制再開	－	○
		rsm_tsk	強制待ち状態からの再開	－	○
		dly_tsk	自タスクの遅延	－	○
タスク例外処理機能		ras_tex	タスク例外処理の要求	－	○
		iras_tex	タスク例外処理の要求	－	○
		dis_tex	タスク例外処理の禁止	－	○
		ena_tex	タスク例外処理の許可	－	○
		sns_tex	タスク例外処理禁止状態の参照	－	○
同期通信機能	セマフォ	cre_sem	セマフォの生成	－	－
		del_sem	セマフォの削除	○	－
		wai_sem	セマフォ資源の獲得	○	○
		pol_sem	セマフォ資源の獲得（ポーリング）	○	○
		twai_sem	セマフォ資源の獲得（タイムアウトあり）	○	○
		sig_sem	セマフォ資源の返却	○	○
		isig_sem	セマフォ資源の返却	○	○

【注】 ○：対応、－：非対応

表 1.3　対応サービスコール一覧(2/3)

項目		名称	機能	R-IN32M4 μITRON	μITRON Ver. 4.0 スタンダード プロファイル
同期通信機能	イベントフラグ	cre_flg	イベントフラグの生成	－	－
		del_flg	イベントフラグの削除	○	－
		set_flg	イベントフラグのセット	○	○
		iset_flg	イベントフラグのセット	○	○
		clr_flg	イベントフラグのクリア	○	○
		wai_flg	イベントフラグ待ち	○	○
		pol_flg	イベントフラグ待ち（ポーリング）	○	○
		twai_flg	イベントフラグ待ち（タイムアウトあり）	○	○
	データ・キュー	snd_dtq	データ・キューへの送信	－	○
		psnd_dtq	データ・キューへの送信（ポーリング）	－	○
		ipsnd_dtq	データ・キューへの送信	－	○
		tsnd_dtq	データ・キューへの送信（タイムアウトあり）	－	○
		fsnd_dtq	データ・キューへの強制送信	－	○
		ifsnd_dtq	データ・キューへの強制送信	－	○
		rcv_dtq	データ・キューからの受信	－	○
		prcv_dtq	データ・キューからの受信（ポーリング）	－	○
	メールボックス	cre_mbx	メールボックスの生成	－	－
		del_mbx	メールボックスの削除	○	－
		snd_mbx	メールボックスへの送信	○	○
		rcv_mbx	メールボックスからの受信	○	○
		prcv_mbx	メールボックスからの受信（ポーリング）	○	○
		trcv_mbx	メールボックスからの受信（タイムアウトあり）	○	○
拡張同期・通信機能	ミューテックス	cre_mtx	ミューテックスの生成	－	－
		del_mtx	ミューテックスの削除	○	－
		loc_mtx	ミューテックスのロック	○	－
		ploc_mtx	ミューテックスのロック（ポーリング）	○	－
		tloc_mtx	ミューテックスのロック（タイムアウトあり）	○	－
		unl_mtx	ミューテックスのロック解除	○	－

【注】 ○：対応、－：非対応

表 1.3　対応サービスコール一覧(3/3)

項目		名称	機能	R-IN32M4 μITRON	μITRON Ver. 4.0 スタンダード プロファイル
メモリプール管理機能	固定長	get_mpf	固定長メモリブロックの獲得	―	○
		pget_mpf	固定長メモリブロックの獲得（ポーリング）	―	○
		tget_mpf	固定長メモリブロックの獲得（タイムアウトあり）	―	○
		rel_mpf	固定長メモリブロックの返却	―	○
時間管理機能	システム時刻管理	set_tim	システム時刻の設定	○	○
		get_tim	システム時刻の参照	○	○
		isig_tim	タイムティックの供給	―	○
	周期ハンドラ	sta_cyc	周期ハンドラの動作開始	―	○
		stp_cyc	周期ハンドラの動作停止	―	○
システム状態管理機能		rot_rdq	タスク優先順位の回転	○	○
		irot_rdq	タスク優先順位の回転	○	○
		get_tid	実行状態のタスクIDの参照	○	○
		iget_tid	実行状態のタスクIDの参照	○	○
		loc_cpu	CPUロック状態への移行	○	○
		iloc_cpu	CPUロック状態への移行	―	○
		unl_cpu	CPUロック状態の解除	○	○
		iunl_cpu	CPUロック状態の解除	―	○
		dis_dsp	ディスパッチの禁止	○	○
		ena_dsp	ディスパッチの許可	○	○
		sns_ctx	コンテキストの参照	―	○
		sns_loc	CPUロック状態の参照	○	○
		sns_dsp	ディスパッチ禁止状態の参照	―	○
		sns_dpn	ディスパッチ保留状態の参照	―	○

【注】 ○：対応、―：非対応

1.4　対応静的 API 一覧

スタンダードプロファイルに対し、R-IN32M4 にて対応する静的 API 一覧を以下に示します。設定方法は、5. オブジェクト静的生成を参照してください。

表 1.4　対応静的 API 一覧

項目		名称	機能	R-IN32M4 μITRON	μITRON Ver. 4.0 スタンダードプロファイル
タスク管理機能		CRE_TSK	タスクの生成	○	○
タスク例外処理機能		DEF_TEX	タスク例外処理ルーチンの定義	－	○
同期通信機能	セマフォ	CRE_SEM	セマフォの生成	○	○
	イベントフラグ	CRE_FLG	イベントフラグの生成	○	○
	データ・キュー	CRE_DTQ	データキューの生成	－	○
	メールボックス	CRE_MBX	メールボックスの生成	○	○
拡張同期・通信機能	ミューテックス	CRE_MTX	ミューテックスの生成	○	－
メモリプール管理機能	固定長	CRE_MPF	固定長メモリプールの生成	－	○
時間管理機能	周期ハンドラ	CRE_CYC	周期ハンドラの生成	－	○
割り込み管理機能		DEF_INH	割り込みハンドラの定義	○	○
システム構成管理機能		DEF_EXC	CPU例外ハンドラの定義	－	○
		ATT_INI	初期化ルーチンの追加	－	○

【注】 ○：対応、－：非対応

1.5　スタンダードプロファイルとの相違点

	R-IN32M4 μITRON仕様	μITRON Ver. 4.0 スタンダードプロファイル
起動要求のキューイング	非対応	対応
タスク優先度	1～15	1～16
セマフォの最大資源数	31	65535以上
メッセージ優先度	1～7	1～16 (タスク優先度以上)
イベントフラグ属性	TA_WSGL非対応 TA_WMULのみ対応	TA_WSGL

スタンダードプロファイルを拡張しており、タスクコンテキスト専用サービスのうち以下の関数は、非タスクコンテキストからの使用が可能です。

sta_tsk	wup_tsk	pol_flg	rot_rdq
ter_tsk	can_wup	sig_sem	get_tid
chg_pri	rel_wai	set_flg	prcv_mbx
get_pri	pol_sem	clr_flg	snd_mbx

1.6　プロセッサの動作モード

ARM プロセッサは、2 つの動作モード（スレッドモード／ハンドラモード）や、アクセスモード（特権／

非特権）をサポートしています。OS ライブラリでは、以下のように使用します。

- タスク
  タスクは、スレッドモード、特権アクセスで動作します。非特権アクセスへの変更は行いません。
  スレッドモードでは、プロセススタックを使用します。

- 非タスク
  割り込みハンドラやディスパッチ処理では、ハンドラモード、特権アクセスで動作します。
  ハンドラモードでは、メインスタックを使用します。

また、メモリ保護管理は行わず、メモリ保護ユニット（MPU）は、使用していません。

1.7　OS の Tick

OS の Tick はハードウェアで行われるため、Tick 割り込みは発生しません。Tick 用タイマはハードウェア RTOS に内蔵されており、ハードウェア RTOS の起動処理と共に設定されます。

Tick 周期はデフォルトでは 1ms に設定されていますが、RTOS の起動前に hwos_set_tick_time 関数を実行することで変更できます。

1.8　開発環境

OS ライブラリの作成に使用したソフトウェア開発ツールを以下に示します。

表 1.5　ソフトウェア開発ツール一覧

ツール ベンダ	コンパイラ
IAR	Embedded Workbench for ARM V8.42.1 (IAR Systems)

2.　ソフトウェア開発手順

ここでは、ソフトウェア開発の一連の手順を説明します。

2.1　設計フロー

ファイル相関図を図 2.1 に示します。

ファイル構成につきましては、別冊の「R-IN32M4-CL3　プログラミング・マニュアル（ドライバ編）」を参照してください。

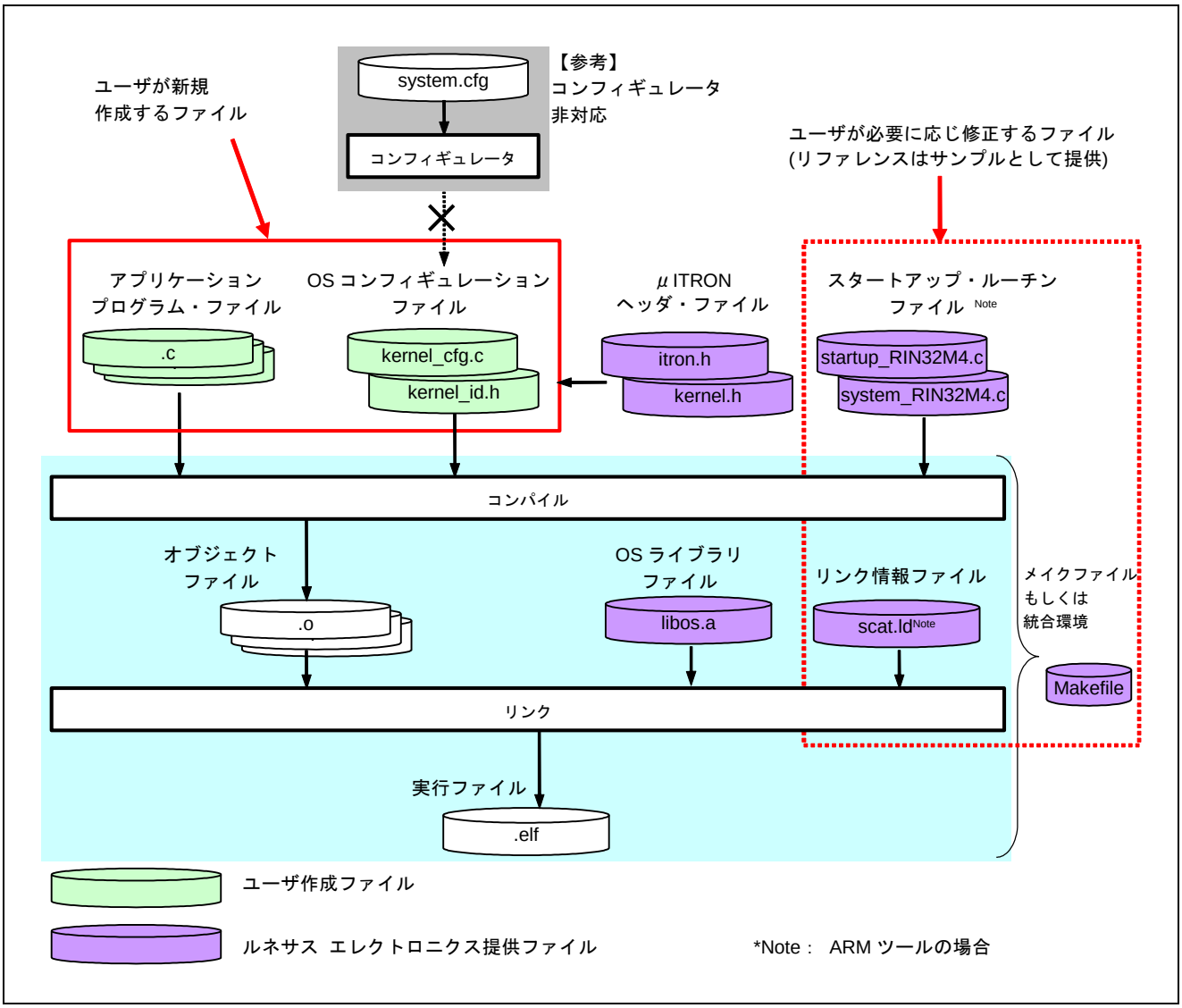

図 2.1　ファイル相関図

2.2　OS コンフィギュレーションファイルの作成

静的生成するオブジェクト、割り込みハンドラ、および Hardware ISR 定義は、kernel_cfg.c にて定義します。

定義方法についての詳細は、5. オブジェクト静的生成、6. Hardware ISR の章を参照してください。

2.3　OS の開始

OS の利用を開始するには、スタートアップ中で hwos_setup 関数を実行します。これにより、OS 利用のための初期設定が完了します。hwos_init 関数は、以前のバージョンとの互換性を保つための空関数です。

2.3.1　OS のセットアップ

hwos_setup

(1)　概要

ハードウェア OS のセットアップ

(2)　C 言語形式

```
ER hwos_setup(void);
```

(3)　パラメータ

なし

(4)　機能

OS コンフィギュレーションファイル(kernel_cfg.c)をもとに、OS 資源の設定を行います。
本関数で行う設定は以下のとおりです。

- スタックポインタの設定
  - スタック領域の最下位アドレスから順に、タスク用スタック領域を確保
  - メインスタックポインタ(MSP)を、割り込み用スタック領域の最上位アドレスに設定
  - プロセススタックポインタ(PSP)を、最初に起動するタスクのスタックポインタに設定
- セマフォの設定
- イベントフラグの設定
- メールボックスの設定
- ミューテックスの設定
- カーネル管理割り込みの設定
- Hardware ISR の設定

(5)　戻り値

戻り値	意味
ER_OK	セットアップ成功

2.3.2　OS の初期設定

hwos_init

(1)　概要

R-IN32M3 シリーズ用の OS ライブラリとの互換性を保つための空関数

(2)　C 言語形式

```
ER hwos_init(void);
```

(3)　パラメータ

なし

(4)　機能

なし

(5)　戻り値

戻り値	意味
ER_OK	なし

2.4　OS の再起動

ファームウェアのアップデートなどで OS を再起動したい場合には、下記の手順を実施してください。

- HW-RTOS モジュールのリセットレジスタで、HW-RTOS 本体をリセット
- CPU の割り込みコントローラ（NVIC）で、HWRTOS 割り込みの無効化と保留要因をクリア
- hwos_setup 関数を実行

以下に、再起動処理のコード記述例を示します。

```
/* Release register protection */
RIN_SYS->SYSPCMD = 0x000000A5;
RIN_SYS->SYSPCMD = 0x00000001;
RIN_SYS->SYSPCMD = 0x0000FFFE;
RIN_SYS->SYSPCMD = 0x00000001;

/* Assert reset */
RINACS->RTOSRST.LONG = 0x00000000;
/* Disable interrupt */
NVIC_DisableIRQ(HWRTOS_IRQn);
NVIC_ClearPendingIRQ(HWRTOS_IRQn);
/* Deassert reset */
RINACS->RTOSRST.LONG = 0x00000001;

/* Set register protection */
RIN_SYS->SYSPCMD = 0x00000000;

/* Start HW-RTOS */
hwos_setup();
```

図 2.2　再起動処理のコード記述例

2.5　注意事項

OS ライブラリを使用する際の注意事項を以下に示します。

- HW-RTOS の周辺レジスタ領域（アドレス：0x40080000～0x4008FFFF）に対して、アクセスしないようにしてください。デバッグ時には、デバッガからアクセスが発生する可能性があります。該当アドレスをメモリ表示させない、ウォッチ対象にしないなど、注意してください。
- デバッガでブレークして停止した際でも、HW-RTOS の Hardware ISR は停止しません。
- イーサネット MAC 機能を使用している場合、ディスパッチ禁止状態や CPU ロック状態では、イーサネット MAC 機能のハードウェア・ファンクションによるコマンドが実行できず、エラーが返ります。ディスパッチ許可、CPU ロック解除の状態で、実行するようにしてください。
- OS ライブラリは、ベクタテーブル(シンボル名__vector_table)内の SVCall 割り込みハンドラと HWRTOS 割り込みハンドラを参照しています。この 2 つの割り込みベクタには下記の関数名を配置してコンパイルしてください。
  - SVCall 割り込みベクタ：SVC_Handler
  - HWRTOS 割り込みベクタ：HWRTOS_IRQHandler
- OS 起動時に、static_interrupt_table に登録された割り込みハンドラで、ベクタテーブルを書き換えます。このとき OS ライブラリは、ベクタテーブルのシンボル名”__vector_table”を参照しますので、このシンボル名を変更しないでください。
- SVCall 割り込みが禁止された状態では OS サービスコールは発行できませんのでご注意ください。
  （例：__disable_irq 実行後のサービスコール発行、など）

3.　データ・タイプとマクロ

本章では、OS ライブラリが提供するサービスコールを発行する際に使用するデータ・タイプ、データ構造体、マクロについて解説しています。

3.1　データ・タイプ

以下に，サービスコールを発行する際に指定する各種パラメータのデータ・タイプ一覧を示します。

表 3.1　データ・タイプ

マクロ	型	意味
B	signed char	符号付き8ビット整数
H	signed short	符号付き16ビット整数
W	signed long	符号付き32ビット整数
UB	unsigned char	符号なし8ビット整数
UH	unsigned short	符号なし16ビット整数
UW	unsigned long	符号なし32ビット整数
VB	char	データ・タイプが定まらない値（8ビット）
VH	short	データ・タイプが定まらない値（16ビット）
VW	long	データ・タイプが定まらない値（32ビット）
VP	void *	データ・タイプが定まらない値（ポインタ）
FP	void (*)	処理プログラムの起動アドレス（ポインタ）
INT	signed int	符号付き32ビット整数
UINT	unsigned int	符号なし32ビット整数
BOOL	INT	真偽値（TRUE，またはFALSE）
FN	INT	機能コード
ER	INT	エラー・コード（サービスコールからの戻り値）
ID	INT	管理オブジェクトのID
ATR	UINT	管理オブジェクトの属性
STAT	UINT	管理オブジェクトの状態
MODE	UINT	サービスコールの動作モード
PRI	INT	タスク，またはメッセージの優先度
SIZE	UINT	領域のサイズ（単位：バイト）
TMO	INT	タスクの待ち時間（単位：ミリ秒）
RELTIM	UINT	相対時間（単位：ミリ秒）
SYSTIM	UINT	システム時間（単位：ミリ秒）
VP_INT	VP	データ・タイプが定まらない値（ポインタ）， または符号付き32 ビット整数
ER_BOOL	ER	エラー・コード、または真偽値（TRUE，またはFALSE）
ER_ID	ER	エラー・コード、または管理オブジェクトのID
ER_UINT	ER	エラー・コード、または符号なし整数
FLGPTN	UINT	イベントフラグのビットパターン

3.2　定数

以下に，本 OS ライブラリにて既定している定数一覧を示します。

表 3.2　定数（一般）

定数	値	意味
NULL	0	無効ポインタ
TRUE	1	真
FALSE	0	偽
E_OK	0	正常終了

表 3.3　定数（オブジェクト属性）

定数	値	意味
TA_NULL	0	オブジェクト属性を指定しない
タスク／ハンドラ　生成時指定属性		
TA_HLNG	0x00	高級言語用のインタフェースで処理単位を起動
TA_ASM	0x01	アセンブリ言語用のインタフェースで処理単位を起動
TA_ACT	0x02	タスクを起動された状態で生成
TA_RSTR	0x04	制約タスク（非対応）
同期／拡張同期通信機能（セマフォ、イベントフラグ、メールボックス、ミューテックス）　生成時指定属性		
TA_TFIFO	0x00	タスクの待ち行列をFIFO順に
TA_TPRI	0x01	タスクの待ち行列をタスクの優先度順に
同期通信機能（イベントフラグ）　生成時指定属性		
TA_WSGL	0x00	イベントフラグを複数のタスクが待つことを許さない（非対応）
TA_WMUL	0x02	イベントフラグを複数のタスクが待つことを許す
TA_CLR	0x04	待ち解除時にイベントフラグをクリア
同期通信機能（メールボックス）　生成時指定属性		
TA_MFIFO	0x00	メッセージのキューをFIFO順に
TA_MPRI	0x02	メッセージのキューをメッセージの優先度順に
拡張同期通信機能（ミューテックス）　生成時指定属性		
TA_INHERIT	0x02	ミューテックスが優先度継承プロトコルをサポート（非対応）
TA_CEILING	0x03	ミューテックスが優先度上限プロトコルをサポート（非対応）
周期ハンドラ　生成時指定属性（非対応）		
TA_STA	0x02	周期ハンドラを動作している状態で生成（非対応）
TA_PHS	0x04	周期ハンドラの位相を保存（非対応）

表 3.4　定数（タイムアウト指定）

定数	値	意味
TMO_POL	0	ポーリング
TMO_FEVR	-1	永久待ち
TMO_NBLK	-2	ノンブロッキング（非対応）

表 3.5　定数（サービスコールの動作モード）

定数	値	意味
TWF_ANDW	0x00	イベントフラグのAND待ち
TWF_ORW	0x01	イベントフラグのOR待ち

表 3.6　定数（その他の定数）

定数	値	意味
TSK_SELF	0	自タスク指定
TSK_NONE	0	該当するタスクがない（未使用）
TPRI_SELF	0	自タスクのベース優先度の指定
TPRI_INI	0	タスクの起動時優先度の指定

表 3.7　定数（エラー・コード）

定数	値	意味
E_SYS	-5	システムエラー
E_RSATR	-11	予約属性
E_PAR	-17	パラメータエラー
E_ID	-18	不正ID番号
E_CTX	-25	コンテキストエラー
E_ILUSE	-28	サービスコール不正使用
E_OBJ	-41	オブジェクト状態エラー
E_NOEXS	-42	オブジェクト未生成
E_QOVR	-43	キューイングオーバフロー
E_RLWAI	-49	待ち状態の強制解除
E_TMOUT	-50	ポーリング失敗またはタイムアウト
E_DLT	-51	待ちオブジェクトの削除
E_UNKNOWN	-99	不明なエラー（H/Wの異常等で、HW-RTOSの応答が不正）

3.3　データ構造体

3.3.1　μITRON V4 定義構造体

T_CTSK

概要

タスク生成情報

構造体宣言

```
typedefstruct t_ctsk {
    ATR    tskatr;  /*!< Task attribute                  */
    VP_INT exinf;   /*!< Task extended information        */
    FP     task;    /*!< Task start address              */
    PRI    itskpri; /*!< Task initial priority           */
    SIZE   stksz;   /*!< Task stack size                 */
    VP     stk;     /*!< Base address of task stack space */
} T_CTSK;
```

構造体メンバ

メンバ		説明
ATR	tskatr	タスク属性 TA_ACTを指定した場合、タスクは起動された状態で生成されます。 TA_ACT（2）：タスクを起動された状態で生成 以下定義を指定する事もできますが、動作に変わりはありません。 TA_HLNG（0）：高級言語用のインタフェースで処理単位を起動（未使用） TA_ASM（1）：アセンブリ言語用のインタフェースで処理単位を起動（未使用）
VP_INT	exinf	タスクの拡張情報(TA_ACT指定時にタスクの引数として渡されます)
FP	task	タスクの起動番地
PRI	itskpri	タスクの起動時優先度 数値（1～15）注：タスクの優先度
SIZE	stksz	タスクのスタックサイズ（バイト数）
VP	stk	タスクのスタック領域の先頭番地 NULL（0）：カーネルによる先頭番地設定（推奨） 数値：本数値を先頭番地として設定

注. μITRON4.0 仕様では、タスクの優先度の範囲は 1～16 です。

T_CSEM

概要

セマフォ生成情報

構造体宣言

```
typedef struct t_csem {
    ATR  sematr;   /*!< Semaphore attribute              */
    UINT isemcnt;  /*!< Initial semaphore resource count */
    UINT maxsem;   /*!< Maximum semaphore resource count */
}T_CSEM;
```

構造体メンバ

メンバ		説明			
ATR	sematr	セマフォ属性			
		タスクの待ち行列	TA_TFIFO	0x00	FIFO順
			TA_TPRI	0x01	タスクの優先度順
UINT	isemcnt	セマフォの資源数の初期値（設定可能最大値：maxsem設定値）			
UINT	maxsem	セマフォの最大資源数（設定可能最大値：31）			

T_CFLG

概要

イベントフラグ生成情報

構造体宣言

```
typedef struct t_cflg {
    ATR    flgatr;   /*!< Eventflag attribute                    */
    FLGPTN iflgptn;  /*!< Initial value of eventflag bit pattern */
}T_CFLG;
```

構造体メンバ

メンバ		説明			
ATR	flgatr	イベントフラグ属性			
		タスクの待ち行列	TA_TFIFO	0x00	FIFO順
			TA_TPRI	0x01	タスクの優先度順
			TA_WMUL	0x02	複数のタスク待ち許可
			TA_CLR	0x04	待ち解除時にビットパターンをクリアする
FLGPTN	iflgptn	イベントフラグのビットパタンの初期値(有効ビット長：16bit)			

制限事項

TA_WSGL は非対応です。TA_WSGL を指定した場合、TA_WMUL として動作します。

T_CMBX

概要

メールボックス生成情報

構造体宣言

```
typedef struct t_cmbx {
    ATR mbxatr;         /*!< Mailbox attribute                          */
    PRI maxmpri;        /*!< Maximum message priority                   */
    VP  mprihd;         /*!< Start address of the area for message
                             queue headers for each message priority */
}T_CMBX;
```

構造体メンバ

メンバ		説明			
ATR	mbxatr	メールボックス属性			
		タスクの待ち行列	TA_TFIFO	0x00	FIFO順
			TA_TPRI	0x01	タスクの優先度順
		メッセージのキュー	TA_MFIFO	0x00	FIFO順
			TA_MPRI	0x02	メッセージの優先度順
PRI	maxmpri	メッセージの優先度最大値（設定可能値：1～7）			
VP	mprihd	優先度別のメッセージキューヘッダ領域の先頭番地（未使用）			

注意　メールボックスの TA_MPRI 属性は、デフォルトでは使用できません。
使用する場合は、8 ユーティリティ関数の「hwos_set_mpri_operation」を参照してください。

T_CMTX

概要

ミューテックス生成情報

構造体宣言

```
typedef struct t_cmtx {
    ATR mtxatr;         /*!< Mutex attribute        */
    PRI ceilpri;        /*!< Mutex ceiling priority */
}T_CMTX;
```

構造体メンバ

メンバ		説明			
ATR	mtxatr	ミューテックス属性			
		タスクの待ち行列	TA_TFIFO	0x00	FIFO順
			TA_TPRI	0x01	タスクの優先度順
PRI	ceilpri	ミューテックスの上限優先度（未使用）			

注意　ミューテックス属性の、TA_INHERIT または TA_CEILING は非対応です。

T_DINH

概要

割り込みハンドラ定義情報のパケット形式

構造体宣言

```
typedef struct t_dinh {
    ATR inhatr;         /*!< Interrupt handler attribute*/
    FP  inthdr;         /*!< Interrupt handler start address*/
} T_DINH;
```

構造体メンバ

メンバ		説明
ATR	inhatr	割り込みハンドラ属性（未使用）
FP	inthdr	割り込みハンドラの起動番地

T_MSG

概要

メッセージヘッダ情報

構造体宣言

```
typedef struct t_msg {
    struct t_msg    *next;
} T_MSG;
```

構造体メンバ

メンバ	説明
t_msg　　*next	メールボックスのメッセージパケットの先頭番地

3.3.2　R-IN32M4 固有定義構造体

TSK_TBL

概要

タスク静的生成情報（CRE_TSK の引数を構造体として定義）

構造体宣言

```
typedef struct task_table {
    ID     id;      /*!< Task ID                          */
    T_CTSK t_ctsk;  /*!< Task creation information packet */
}TSK_TBL;
```

構造体メンバ

メンバ		説明
ID	id	静的生成するタスクのID 数値（1～64）：対象タスクID
T_CTSK	t_ctsk	タスク生成情報

SEM_TBL

概要

セマフォ静的生成情報（CRE_SEM の引数を構造体として定義）

構造体宣言

```
typedef struct semaphore_table {
    ID     id;       /*!< Semaphore ID                          */
    T_CSEM pk_csem;  /*!< Semaphore creation infomation packet */
}SEM_TBL;
```

構造体メンバ

メンバ		説明
ID	id	静的生成するセマフォのID 数値（1～128）：対象セマフォID (ミューテックスIDと重ならないこと)
T_CSEM	pk_csem	セマフォ生成情報

FLG_TBL

概要

フラグ静的生成情報（CRE_FLG の引数を構造体として定義）

構造体宣言

```
typedef struct flag_table {
    ID     id;       /*!< Eventflag ID                          */
    T_CFLG pk_cflg;  /*!< Eventflag creation infomation packet */
}FLG_TBL;
```

構造体メンバ

メンバ		説明
ID	id	静的生成するフラグのID 数値（1～64）：対象フラグID
T_CFLG	pk_cflg	フラグ生成情報

MBX_TBL

概要

メールボックス静的生成情報（CRE_MBX の引数を構造体として定義）

構造体宣言

```
typedef struct mailbox_table {
    ID     id;       /*!< Mailbox ID                          */
    T_CMBX pk_cmbx;  /*!< Mailbox creation infomation packet */
}MBX_TBL;
```

構造体メンバ

メンバ		説明
ID	id	静的生成するメールボックスのID 数値（1～64）：対象メールボックスID
T_CMBX	pk_cmbx	メールボックス生成情報

MTX_TBL

概要

ミューテックス静的生成情報（CRE_MTX の引数を構造体として定義）

構造体宣言

```
typedef struct mutex_table {
  ID     id;       /*!< Mutex ID                          */
  T_CMTX pk_cmtx;  /*!< Mutex creation infomation packet */
}MTX_TBL;
```

構造体メンバ

メンバ		説明
ID	id	静的生成するミューテックスのID 数値（1～128）：対象ミューテックスID (セマフォIDと重ならないこと)
T_CMTX	pk_cmtx	ミューテックス生成情報

INT_TBL

概要

割り込みハンドラ生成情報（DEF_INH の引数を構造体として定義）

構造体宣言

```
typedef struct interrupt_table {
    INHNO  inhno;    /*!< Interrupt handler number to be defined                          */
    T_DINH pk_dinh;  /*!< Pointer to the packet containing the interrupt handler definition
                          information */
}INT_TBL;
```

構造体メンバ

メンバ	説明
INHNO　inhno	割り込みハンドラ生成対象割り込み番号 数値（0～152）：対象割り込み番号 ※
T_DINH　pk_dinh	割り込みハンドラ定義情報を入れたパケットへのポインタ

※割り込みは、例外番号の 16 以降に割り当たっています。
対象割り込み番号は、例外番号 16 以降を 0 番から付け替えた番号です。
例外番号から、-16 した値を設定してください。
対象の割り込みは、ユーザーズマニュアルの『割り込み一覧』を参照してください。

HWISR_TBL

概要

Hardware ISR オブジェクト生成情報

構造体宣言

```
typedef struct hwisr_table {
    INHNO  inhno;          /*!< Interrupt handler number to be defined */
    UINT   hwisr_syscall;  /*!< System call                            */
    ID     id;             /*!< Terget ID                              */
    FLGPTN setptn;         /*!< Bit pattern (only set_flg)             */
}HWISR_TBL;
```

構造体メンバ

メンバ		説明
INHNO	inhno	Hardware ISR対象割り込み番号 数値（0～152）：対象割り込み番号
UINT	hwisr_syscall	割り込み発生時自動実行するサービスコール HWISR_SET_FLG（1）：set_flg() HWISR_SIG_SEM（2）：sig_sem() HWISR_REL_WAI（3）：rel_wai() HWISR_WUP_TSK（4）：wup_tsk()
ID	id	割り込み発生時に自動実行するサービスコールの対象オブジェクトのID
FLGPTN	setptn	セットするビットパタン（set_flg時のみ有効）

3.4　グローバル変数

OS ライブラリで使用されているグローバル変数は以下の通りです。ユーザーアプリケーションのシンボルと、シンボル名が重ならないようにしてください。

表 3.8　グローバル変数

変数名
HWRTOS_Sbt
HWRTOS_Sit
HWRTOS_IntTable

4. サービスコール

4.1 タスク管理機能

タスク管理機能として提供しているサービスコールの一覧を示します。

表 4.1　タスク管理機能

サービスコール名	機能	発行有効範囲
sta_tsk	タスクの起動（起動コード指定）	タスク、非タスク
ext_tsk	自タスクの終了	タスク
ter_tsk	タスクの強制終了	タスク、非タスク
chg_pri	タスク優先度の変更	タスク、非タスク
get_pri	タスク優先度の参照	タスク、非タスク

タスク管理機能の仕様を以下に示します。

表 4.2　タスク管理機能の仕様

項番	項目	内容
1	タスクID	1～64
2	タスク優先度	1～15
3	タスクの起動要求キューイング数	63
4	タスク属性	TA_HLNG : 高級言語記述（未使用） TA_ASM : アセンブリ言語記述（未使用） TA_ACT : タスク生成後にタスクを実行可能状態へ

sta_tsk

概要

タスクの起動

C 言語形式

ER sta_tsk(ID tskid, VP_INT stacd);

パラメータ

I/O	パラメータ		説明
I	ID	tskid	タスクのID 数値（1～64）：対象タスクID
I	VP_INT	stacd	タスクの起動コード

機能

tskid で指定されたタスクを DORMANT 状態から READY 状態へと遷移させます。
stacd には，対象タスクに引き渡す拡張情報を指定します。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_ID	-18	不正ID番号（tskidが不正あるいは使用できない）
E_OBJ	-41	オブジェクト状態エラー（対象タスクがDORMANT状態ではない）
E_NOEXS	-42	オブジェクト未生成（対象タスクが未登録）

ext_tsk

概要
タスクの終了

C 言語形式
```
void ext_tsk(void);
```

パラメータ
なし

機能
自タスクを RUN 状態から DORMANT 状態へ遷移させます。

本サービスコールは、呼び出し元には戻りません。ただし、エラーを検出した場合には、リターンします。CPU ロック、ディスパッチ禁止、割込みハンドラの各状態で発行した場合、エラーとなります。

戻り値
なし

制限事項
ミューテックスをロックしたままタスク終了した場合、ロック解除処理が行われません。必ずロックを解除してからタスクを終了してください。

注意事項
CPU ロック、ディスパッチ禁止、割込みハンドラの各状態でこの関数を発行しないでください。これらの状態で発行した場合の動作は保証しません。

ter_tsk

概要

タスクの強制終了

C 言語形式

ER ter_tsk(ID tskid);

パラメータ

I/O	パラメータ	説明
I	ID tskid	タスクのID 数値（1～64）：対象タスクID

機能

tskid で指定されたタスクを強制的に DORMANT 状態へと遷移させます。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_ID	-18	不正ID番号（tskidが不正、あるいは使用できない）
E_CTX	-25	CPUロック状態で実行した
E_ILUSE	-28	サービスコール不正使用（対象タスクが自タスク）
E_OBJ	-41	オブジェクト状態エラー（対象タスクがDORMANT状態）
E_NOEXS	-42	オブジェクト未生成（対象タスクが未登録）

制限事項

ミューテックスをロックしたままタスク終了した場合、ロック解除処理が行われません。必ずロックを解除してからタスクを終了してください。

chg_pri

概要

タスク優先度の変更

C 言語形式

ER chg_pri(ID tskid, PRI tskpri);

パラメータ

I/O	パラメータ		説明
I	ID	tskid	タスクのID TSK_SELF（0）：自タスクID 数値（1～64）：対象タスクID
I	PRI	tskpri	変更後のベース優先度[注1] TPRI_INI（0）：対象タスクの初期優先度 数値（1～15）[注2]：対象タスクのベース優先度

注 1　優先度継承等の優先度制御機構は対応していないため、ベース優先度と現在優先度は常に同じです。
注 2. μITRON4.0 仕様では、タスクの優先度の範囲は 1～16 です。

機能

tskid で指定されたタスクのベース優先度を、tskpri で指定される値に変更します。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_PAR	-17	パラメータエラー（tskpriが不正）
E_ID	-18	不正ID番号（tskidが不正、あるいは使用できない） 割り込みハンドラからTSK_SELFを指定した
E_CTX	-25	CPUロック状態で実行した
E_OBJ	-41	オブジェクト状態エラー（対象タスクがDORMANT状態）
E_NOEXS	-42	オブジェクト未生成（対象タスクが未登録）

制限事項

TA_TFIFO 属性のオブジェクトで待ち状態のタスクに対して chg_pri を発行した場合、対象タスクのオブジェクト待ち順位が最低になります。　μITRON(ver4.03)仕様では待ち順位に変更はありません。

get_pri

概要

タスク優先度の参照

C 言語形式

ER get_pri(ID tskid, PRI *p_tskpri);

パラメータ

I/O	パラメータ		説明
I	ID	tskid	タスクのID TSK_SELF（0）：自タスクID 数値（1～64）：対象タスクID
O	PRI	*p_tskpri	対象タスクの現在優先度

機能

tskid で指定されたタスクの現在優先度を参照し、p_tskpri に返します。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_PAR	-17	p_tskpriにNULLポインタを指定した
E_ID	-18	不正ID番号（tskidが不正、あるいは使用できない） 割り込みハンドラからTSK_SELFを指定した
E_CTX	-25	CPUロック状態で実行した
E_OBJ	-41	オブジェクト状態エラー（対象タスクがDORMANT状態）
E_NOEXS	-42	オブジェクト未生成（対象タスクが未登録）

4.2　タスク付属同期機能

タスク付属同期機能として提供しているサービスコールの一覧を示します。

表 4.3　タスク付属同期機能

サービスコール名	機能	発行有効範囲
slp_tsk	起床待ち	タスク
tslp_tsk	起床待ち（タイムアウトあり）	タスク
wup_tsk	タスクの起床	タスク、非タスク
iwup_tsk	タスクの起床	非タスク
can_wup	タスク起床要求のキャンセル	タスク、非タスク
rel_wai	待ち状態の強制解除	タスク、非タスク
irel_wai	待ち状態の強制解除	非タスク

タスク付属同期機能の仕様を以下に示します。

表 4.4　タスク付属同期機能の仕様

項番	項目	内容
1	タスクの起床要求キューイング数	63

slp_tsk

概要
タスク起床待ち

C 言語形式
ER slp_tsk(void);

パラメータ
なし

機能
自タスクを RUNNING 状態から起床待ち状態へと遷移させます。

ただし，本サービスコールを発行した際，自タスクの起床要求がキューイングされていた（起床要求キューイング数が 0x0 以外）場合には，状態操作処理は行わず，起床要求キューイング数から 0x1 を減算し実行を継続します。

slp_tsk()は、tslp_tsk(TMO_FEVR)と同一です。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_CTX	-25	CPUロック状態/ディスパッチ禁止状態/割り込みハンドラから実行した
E_RLWAI	-49	待ち状態の強制解除（待ち状態の間にrel_waiを受け付け）

tslp_tsk

概要

タスク起床待ち（タイムアウトあり）

C 言語形式

ER tslp_tsk(TMO tmout);

パラメータ

I/O	パラメータ	説明
I	TMO tmout	タイムアウト指定（単位はms） TMO_FEVR（-1）：永久待ち（slp_tsk()と同等の処理） TMO_POL(0)：ポーリング 数値：待ち時間

機能

自タスクを RUNNING 状態から起床待ち状態へと遷移させます。

ただし，本サービスコールを発行した際，自タスクの起床要求がキューイングされていた（起床要求キューイング数が 0x0 以外）場合には，状態操作処理は行わず，起床要求キューイング数から 0x1 を減算し実行を継続します。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_PAR	-17	パラメータエラー（tmoutが不正）
E_CTX	-25	CPUロック状態/ディスパッチ禁止状態/割り込みハンドラから実行した
E_RLWAI	-49	待ち状態の強制解除（待ち状態の間にrel_waiを受け付け）
E_TMOUT	-50	タイムアウト

wup_tsk / iwup_tsk

概要

タスクの起床

C 言語形式

ER wup_tsk(ID tskid);
ER iwup_tsk(ID tskid);

パラメータ

I/O	パラメータ	説明
I	ID tskid	タスクのID TSK_SELF（0）：自タスクID 数値（1～64）：対象タスクID

機能

tskid で指定されるタスクを起床待ち状態から READY 状態に遷移させます。

ただし，本サービスコールを発行した際，対象タスクが起床待ち状態以外の場合には，状態操作処理は行わず，起床要求キューイング数に 0x1 を加算します。

起床要求キューイング数の最大値を超える起床要求を行った場合は、エラー（E_QOVR）となります。

備考 起床要求キューイング数の最大値は 63 です。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_ID	-18	不正ID番号（tskidが不正あるいは使用できない） 割り込みハンドラからTSK_SELFを指定した
E_CTX	-25	CPUロック状態から実行した タスクから実行した(iwup_tskのみ)
E_OBJ	-41	オブジェクト状態エラー（対象タスクがDORMANT状態）
E_NOEXS	-42	オブジェクト未生成（対象タスクが未登録）
E_QOVR	-43	キューイングオーバフロー（起床要求回数が63回を超えた）

can_wup

概要

タスク起床要求のキャンセル

C 言語形式

ER_UINT can_wup(ID tskid);

パラメータ

I/O	パラメータ	説明
I	ID tskid	タスクのID TSK_SELF（0）：自タスクID 数値（1～64）：対象タスクID

機能

tskid で指定されるタスクに対してキューイングされている起床要求をキャンセル（起床要求キューイング数を 0 にクリア）し、キューイングされていた起床要求の回数を返します。

戻り値

マクロ	数値	意味
-	0または正の整数	キューイングされていた起床要求回数
E_ID	-18	不正ID番号（tskidが不正あるいは使用できない） 割り込みハンドラからTSK_SELFを指定した
E_CTX	-25	CPUロック状態から実行した
E_OBJ	-41	オブジェクト状態エラー（対象タスクがDORMANT状態）
E_NOEXS	-42	オブジェクト未生成（対象タスクが未登録）

rel_wai / irel_wai

概要

待ち状態の強制解除

C 言語形式

ER rel_wai(ID tskid);
ER irel_wai(ID tskid);

パラメータ

I/O	パラメータ	説明
I	ID tskid	タスクのID 数値（1～64）：対象タスクID

機能

tskid で指定されたタスクの WAITING 状態（起床待ち状態、セマフォ待ち状態、イベントフラグ待ち状態、メッセージ待ち状態）を強制的に解除します。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_ID	-18	不正ID番号（tskidが不正あるいは使用できない）
E_CTX	-25	CPUロック状態から実行した タスクから実行した(irel_waiのみ)
E_OBJ	-41	オブジェクト状態エラー（対象タスクがWAITING状態でない）
E_NOEXS	-42	オブジェクト未生成（対象タスクが未登録）

4.3　同期通信機能（セマフォ）

同期通信機能（セマフォ）として提供しているサービスコールの一覧を示します。

表 4.5　同期通信機能（セマフォ）

サービスコール名	機能	発行有効範囲
del_sem	セマフォの削除	タスク
wai_sem	セマフォ資源の獲得	タスク
pol_sem	セマフォ資源の獲得（ポーリング）	タスク、非タスク
twai_sem	セマフォ資源の獲得（タイムアウトあり）	タスク
sig_sem	セマフォ資源の返却	タスク、非タスク
isig_sem	セマフォ資源の返却	非タスク

セマフォ機能の仕様を以下に示します。

表 4.6　セマフォ機能の仕様

項番	項目	内容
1	セマフォID	1～128（ミューテックスと合わせて）
2	セマフォの最大資源数	31
3	サポート属性	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

del_sem

概要

セマフォの削除

C 言語形式

ER del_sem(ID semid);

パラメータ

I/O	パラメータ	説明
I	ID semid	セマフォのID 数値（1～128）：対象セマフォID

機能

semid で指定された ID を持つセマフォを削除します。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_ID	-18	不正ID番号（semidが不正、あるいは使用できない）
E_CTX	-25	CPUロック状態/割り込みハンドラから実行した
E_NOEXS	-42	オブジェクト未生成

wai_sem

概要

セマフォ資源の獲得

C 言語形式

ER wai_sem(ID semid);

パラメータ

I/O	パラメータ		説明
I	ID	semid	セマフォのID 数値（1～128）：対象セマフォID

機能

semid で指定されるセマフォから、資源を 1 つ獲得します。
対象セマフォの資源数が 0 の場合は、自タスクをセマフォ待ち状態に遷移させます。
wai_sem(smid)は twai_sem(semid, TMO_FEVR)と同一です。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_ID	-18	不正ID番号（semidが不正、あるいは使用できない）
E_CTX	-25	CPUロック状態/ディスパッチ禁止状態/割り込みハンドラから実行した
E_NOEXS	-42	オブジェクト未生成
E_RLWAI	-49	待ち状態の強制解除（待ち状態の間にrel_waiを受け付け）
E_DLT	-51	待ちオブジェクトの削除（待ち状態の間に対象セマフォが削除）

pol_sem

概要

セマフォ資源の獲得（ポーリング）

C 言語形式

ER pol_sem(ID semid);

パラメータ

I/O	パラメータ	説明
I	ID semid	セマフォのID 数値（1～128）：対象セマフォID

機能

semid で指定されるセマフォから、資源を 1 つ獲得します。

対象セマフォの資源数が 0 の場合は、自タスクをセマフォ待ち状態には移行させず、ポーリング失敗となります。

pol_sem(semid)は twai_sem(semid, TMO_POL)と同一です。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_ID	-18	不正ID番号（semidが不正、あるいは使用できない）
E_CTX	-25	CPUロック状態から実行した
E_NOEXS	-42	オブジェクト未生成
E_TMOUT	-50	ポーリング失敗

twai_sem

概要

セマフォ資源の獲得（タイムアウトあり）

C 言語形式

ER twai_sem(ID semid, TMO tmout);

パラメータ

I/O	パラメータ		説明
I	ID	semid	セマフォのID 数値（1～128）：対象セマフォID
I	TMO	tmout	タイムアウト指定（単位はms） TMO_POL（0）：ポーリング（pol_sem()と同等の処理となる） TMO_FEVR（-1）：永久待ち（wai_sem()と同等の処理となる） 数値：待ち時間

機能

semid で指定されるセマフォから、資源を 1 つ獲得します。

対象セマフォの資源数が 0 の場合は、自タスクをセマフォ待ち状態には移行させます。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_PAR	-17	パラメータエラー（tmoutが不正）
E_ID	-18	不正ID番号（semidが不正、あるいは使用できない）
E_CTX	-25	CPUロック状態/ディスパッチ禁止状態/割り込みハンドラから実行した（TMO_POL指定時は割込みハンドラから実行可能）
E_NOEXS	-42	オブジェクト未生成
E_RLWAI	-49	待ち状態の強制解除（待ち状態の間にrel_waiを受け付け）
E_TMOUT	-50	タイムアウトまたはポーリング失敗
E_DLT	-51	待ちオブジェクトの削除（待ち状態の間に対象セマフォが削除）

sig_sem / isig_sem

概要

セマフォ資源の返却

C 言語形式

ER sig_sem(ID semid);

ER isig_sem(ID semid);

パラメータ

I/O	パラメータ	説明
I	ID semid	セマフォのID 数値（1～128）：対象セマフォID

機能

semid で指定されるセマフォから、資源を 1 つ返却します。

対象セマフォに対して資源の獲得を待っているタスクがある場合には、待ち行列の先頭タスクを待ち解除します。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_ID	-18	不正ID番号（semidが不正、あるいは使用できない）
E_CTX	-25	CPUロック状態から実行した タスクから実行した（isig_semのみ）
E_NOEXS	-42	オブジェクト未生成（対象セマフォが未登録）
E_QOVR	-43	キューイングオーバフロー（最大資源数(31)を超える返却）

4.4　同期通信機能（イベントフラグ）

同期通信機能（イベントフラグ）として提供しているサービスコールの一覧を示します。

表 4.7　同期通信機能（イベントフラグ）

サービスコール名	機能	発行有効範囲
del_flg	イベントフラグの削除	タスク
set_flg	イベントフラグのセット	タスク、非タスク
iset_flg	イベントフラグのセット	非タスク
clr_flg	イベントフラグのクリア	タスク、非タスク
wai_flg	イベントフラグ待ち	タスク
pol_flg	イベントフラグ待ち（ポーリング）	タスク、非タスク
twai_flg	イベントフラグ待ち（タイムアウトあり）	タスク

イベントフラグ機能の仕様を以下に示します。

表 4.8　イベントフラグ機能の仕様

項番	項目	内容
1	イベントフラグID	1～64
2	イベントフラグのビット数	16ビット
3	サポート属性	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_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

del_flg

概要
イベントフラグの削除

C 言語形式
ER del_flg(ID flgid);

パラメータ

I/O	パラメータ		説明
I	ID	flgid	削除対象のイベントフラグのID番号 数値（1～64）：対象イベントフラグID

機能
flgid で指定された ID を持つイベントフラグを削除します。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_ID	-18	不正ID番号（flgidが不正、あるいは使用できない）
E_CTX	-25	CPUロック状態/割り込みハンドラから実行した
E_NOEXS	-42	オブジェクト未生成

set_flg / iset_flg

概要

イベントフラグのセット

C 言語形式

ER set_flg(ID flgid, FLGPTN setptn)

ER iset_flg(ID flgid, FLGPTN setptn)

パラメータ

I/O	パラメータ		説明
I	ID	flgid	セット対象のイベントフラグのID番号 数値（1～64）：対象イベントフラグID
I	FLGPTN	setptn	セットするビットパターン（下位16bitが有効）

機能

flgid で指定されたイベントフラグに対して、setptn で指定されたビットパターンをセットします。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_PAR	-17	パラメータエラー（setptnが不正、bit16以上に1指定）
E_ID	-18	不正ID番号（flgidが不正、あるいは使用できない）
E_CTX	-25	CPUロック状態から実行した タスクから実行した（iset_flgのみ）
E_NOEXS	-42	オブジェクト未生成

clr_flg

概要

イベントフラグのクリア

C 言語形式

ER clr_flg(ID flgid, FLGPTN clrptn)

パラメータ

I/O	パラメータ	説明
I	ID flgid	クリア対象のイベントフラグのID番号 数値（1～64）：対象イベントフラグID
I	FLGPTN clrptn	クリアするビットパターン（下位16bitが有効）

機能

flgid で指定されたイベントフラグに対して、clrptn の対応するビットが 0 になっているビットをクリアします。

set_flg サービスコールと異なり、clrptn の bit16 以上に 1 がセットされていてもエラーにはなりません。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_ID	-18	不正ID番号（flgidが不正、あるいは使用できない）
E_CTX	-25	CPUロック状態から実行した
E_NOEXS	-42	オブジェクト未生成

wai_flg

概要

イベントフラグ待ち

C 言語形式

ER wai_flg(ID flgid, FLGPTN waiptn, MODE wfmode, FLGPTN *p_flgptn)

パラメータ

I/O	パラメータ		説明
I	ID	flgid	待ち対象のイベントフラグのID番号 数値（1～64）：対象イベントフラグID
I	FLGPTN	waiptn	待ちビットパターン（下位16bitが有効） （bit16以上に1を指定、あるいは0x0000を指定するとエラー）
I	MODE	wfmode	待ちモード TWF_ANDW（0）：イベントフラグのAND待ち TWF_ORW（1）：イベントフラグのOR待ち
O	FLGPTN	*p_flgptn	待ち解除時のビットパターン格納領域へのポインタ

機能

flgid で指定されるイベントフラグのビットパタンが、waiptn と wfmode で指定される待ち条件を満たすのを待ちます。

wai_flg(～)は、twai_flg(～, TMO_FEVR)と同一です。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_PAR	-17	パラメータエラー（waiptnが不正、あるいはwfmodeが不正）
E_ID	-18	不正ID番号（flgidが不正、あるいは使用できない）
E_CTX	-25	CPUロック状態/ディスパッチ禁止状態/割り込みハンドラから実行した
E_NOEXS	-42	オブジェクト未生成
E_RLWAI	-49	待ち状態の強制解除（待ち状態の間にrel_waiを受け付け）
E_DLT	-51	待ちオブジェクトの削除（待ち状態の間に対象イベントフラグが削除）

pol_flg

概要

イベントフラグ待ち（ポーリング）

C 言語形式

ER pol_flg(ID flgid, FLGPTN waiptn, MODE wfmode, FLGPTN *p_flgptn)

パラメータ

I/O	パラメータ		説明
I	ID	flgid	待ち対象のイベントフラグのID番号 数値（1～64）：対象イベントフラグID
I	FLGPTN	waiptn	待ちビットパターン（下位16bitが有効） （bit16以上に1を指定、あるいは0x0000を指定するとエラー）
I	MODE	wfmode	待ちモード TWF_ANDW（0）：イベントフラグのAND待ち TWF_ORW（1）：イベントフラグのOR待ち
O	FLGPTN	*p_flgptn	待ち解除時のビットパターン格納領域へのポインタ

機能

flgid で指定されるイベントフラグのビットパタンが、waiptn と wfmode で指定される待ち条件を満たしているかどうかをポーリングします。

pol_flg(～)は、twai_flg(～, TMO_POL)と同一です。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_PAR	-17	パラメータエラー（waiptnが不正、あるいはwfmodeが不正）
E_ID	-18	不正ID番号（flgidが不正、あるいは使用できない）
E_CTX	-25	CPUロック状態から実行した
E_NOEXS	-42	オブジェクト未生成
E_TMOUT	-50	ポーリング失敗

twai_flg

概要

イベントフラグ待ち（タイムアウトあり）

C 言語形式

ER twai_flg(ID flgid, FLGPTN waiptn, MODE wfmode, FLGPTN *p_flgptn, TMO tmout)

パラメータ

I/O	パラメータ		説明
I	ID	flgid	待ち対象のイベントフラグのID番号 数値（1～64）：対象イベントフラグID
I	FLGPTN	waiptn	待ちビットパターン（下位16bitが有効） （bit16以上に1を指定、あるいは0x0000を指定するとエラー）
I	MODE	wfmode	待ちモード TWF_ANDW（0）：イベントフラグのAND待ち TWF_ORW（1）：イベントフラグのOR待ち
O	FLGPTN	*p_flgptn	待ち解除時のビットパターン格納領域へのポインタ
I	TMO	tmout	タイムアウト指定（単位はms） TMO_POL（0）：ポーリング（pol_flg()と同等の処理となる） TMO_FEVR（-1）：永久待ち（wai_flg()と同等の処理となる） 数値：待ち時間

機能

flgid で指定されるイベントフラグのビットパタンが、waiptn と wfmode で指定される待ち条件を満たすのを待ちます。*p_flgptn には、条件成立時のイベントフラグのビットパターンを返します。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_PAR	-17	パラメータエラー（waiptnが不正）
E_ID	-18	不正ID番号
E_CTX	-25	CPUロック状態/ディスパッチ禁止状態/割り込みハンドラから実行した （TMO_POL指定時は割り込みハンドラから実行可能）
E_NOEXS	-42	オブジェクト未生成
E_RLWAI	-49	待ち状態の強制解除（待ち状態の間にrel_waiを受け付け）
E_TMOUT	-50	タイムアウト、またはポーリング失敗
E_DLT	-51	待ちオブジェクトの削除（待ち状態の間に対象イベントフラグが削除）

4.5　同期通信機能（メールボックス）

同期通信機能（メールボックス）として提供しているサービスコールの一覧を示します。

表 4.9　同期通信機能（メールボックス）

サービスコール名	機能	発行有効範囲
del_mbx	メールボックスの削除	タスク
snd_mbx	メールボックスへの送信	タスク、非タスク
isnd_mbx	メールボックスへの送信	非タスク
rcv_mbx	メールボックスからの受信	タスク
prcv_mbx	メールボックスからの受信（ポーリング）	タスク、非タスク
trcv_mbx	メールボックスからの受信（タイムアウトあり）	タスク

メールボックス機能の仕様を以下に示します。

表 4.10　メールボックス機能の仕様

項番	項目	内容
1	メールボックスID	1～64
2	メッセージ優先度	1～7
3	メッセージのキューイング数	192
4	サポート属性	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_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（制限事項あり）

del_mbx

概要

メールボックスの削除

C 言語形式

ER del_mbx(ID mbxid);

パラメータ

I/O	パラメータ	説明
I	ID mbxid	削除対象のメールボックスのID番号 数値（1～64）：対象メールボックスID

機能

mbxid で指定された ID を持つメールボックスを削除します。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_ID	-18	不正ID番号（mbxidが不正、あるいは使用できない）
E_CTX	-25	CPUロック状態/割り込みハンドラから実行した
E_NOEXS	-42	オブジェクト未生成

snd_mbx / isnd_mbx

概要

メールボックスへの送信

C 言語形式

ER snd_mbx(ID mbxid, T_MSG *pk_msg);
ER isnd_mbx(ID mbxid, T_MSG *pk_msg);

パラメータ

I/O	パラメータ	説明
I	ID mbxid	送信対象のメールボックスのID番号 数値（1～64）：対象メールボックスID
I	T_ MSG *pk_msg	メールボックスへ送信するメッセージパケットの先頭アドレス

機能

mbxid で指定されるメールボックスに、pk_msg を先頭アドレスとするメッセージを送信します。

戻り値の E_RSATR は本サービスコールの実装依存で追加したエラーコードです。ユーティリティ関数の hwos_set_mpri_operation で TA_MPRI 属性の使用を禁止したメールボックスへメッセージパケットを送信すると E_RSATR が返ります。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_RSATR	-11	hwos_set_mpri_operation関数でHWOS_DISABLE_MPRIを指定時に、TA_MPRI属性で生成したメールボックスへメッセージを送信した
E_PAR	-17	パラメータエラー（pk_msgが不正、あるいはメッセージプライオリティが不正）
E_ID	-18	不正ID番号（mbxidが不正、あるいは使用できない）
E_CTX	-25	CPUロック状態から実行した タスクから実行した（isnd_mbxのみ）
E_NOEXS	-42	オブジェクト未生成
E_QOVR	-43	キューイングオーバフロー(メッセージのキューイング数が最大値(192)を超えた）

rcv_mbx

概要

メールボックスからの受信

C 言語形式

ER rcv_mbx(ID mbxid, T_MSG **ppk_msg);

パラメータ

I/O	パラメータ	説明
I	ID　mbxid	受信対象のメールボックスのID番号 数値（1～64）：対象メールボックスID
O	T_ MSG　**ppk_msg	メールボックスから受信したメッセージパケットの先頭アドレス格納領域へのポインタ

機能

mbxid で指定されるメールボックスからメッセージを受信し、その先頭番地を ppk_msg に返す。

対象メールボックスにメッセージがない場合は、自タスクをメッセージ待ち状態に移行させます。

rcv_mbx(～)は、trcv_mbx(～, TMO_FEVR)と同一です。

戻り値の E_RSATR は本サービスコールの実装依存で追加したエラーコードです。ユーティリティ関数の hwos_set_mpri_operation で TA_MPRI 属性の使用を禁止したメールボックスからメッセージパケットを受信すると E_RSATR が返ります。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_RSATR	-11	hwos_set_mpri_operation関数でHWOS_DISABLE_MPRIを指定時に、TA_MPRI属性で生成したメールボックスからメッセージを受信した
E_PAR	-17	パラメータエラー（ppk_msgが不正）
E_ID	-18	不正ID番号（mbxidが不正、あるいは使用できない）
E_CTX	-25	CPUロック状態/ディスパッチ禁止状態/割り込みハンドラから実行した
E_NOEXS	-42	オブジェクト未生成
E_RLWAI	-49	待ち状態の強制解除（待ち状態の間にrel_waiを受け付け）
E_DLT	-51	待ちオブジェクトの削除（待ち状態の間に対象メールボックスが削除された）

prcv_mbx

概要

メールボックスからの受信（ポーリング）

C 言語形式

ER prcv_mbx(ID mbxid, T_MSG **ppk_msg);

パラメータ

I/O	パラメータ	説明
I	ID mbxid	受信対象のメールボックスのID番号 数値（1～64）：対象メールボックスID
O	T_ MSG **ppk_msg	メールボックスから受信したメッセージパケットの先頭アドレス格納領域へのポインタ

機能

mbxid で指定されるメールボックスからメッセージを受信し、その先頭番地を ppk_msg に返す。

対象メールボックスにメッセージがない場合は、自タスクをメッセージ待ち状態には移行させず、ポーリング失敗となります。

prcv_mbx(～)は、trcv_mbx(～, TMO_POL)と同一です。

戻り値の E_RSATR は本サービスコールの実装依存で追加したエラーコードです。ユーティリティ関数の hwos_set_mpri_operation で TA_MPRI 属性の使用を禁止したメールボックスからメッセージパケットを受信すると E_RSATR が返ります。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_RSATR	-11	hwos_set_mpri_operation関数でHWOS_DISABLE_MPRIを指定時に、TA_MPRI属性で生成したメールボックスからメッセージを受信（ポーリング）した
E_PAR	-17	パラメータエラー（ppk_msgが不正）
E_ID	-18	不正ID番号（mbxidが不正、あるいは使用できない）
E_CTX	-25	CPUロック状態から実行した
E_NOEXS	-42	オブジェクト未生成
E_TMOUT	-50	ポーリング失敗

trcv_mbx

概要

メールボックスからの受信（タイムアウトあり）

C 言語形式

ER trcv_mbx(ID mbxid, T_MSG **ppk_msg, TMO tmout);

パラメータ

I/O	パラメータ		説明
I	ID	mbxid	受信対象のメールボックスのID番号 数値（1～64）：対象メールボックスID
O	T_ MSG	**ppk_msg	メールボックスから受信したメッセージパケットの先頭アドレス格納領域へのポインタ
I	TMO	tmout	タイムアウト指定（単位はms） TMO_POL（0）：ポーリング（prcv_mbx ()と同等の処理となる） TMO_FEVR（-1）：永久待ち（rcv_mbx()と同等の処理となる） 数値：待ち時間

機能

mbxid で指定されるメールボックスからメッセージを受信し、その先頭番地を ppk_msg に返す。

対象メールボックスにメッセージがない場合は、自タスクをメッセージ待ち状態に移行させます。

戻り値の E_RSATR は本サービスコールの実装依存で追加したエラーコードです。ユーティリティ関数の hwos_set_mpri_operation で TA_MPRI 属性の使用を禁止したメールボックスからメッセージパケットを受信すると E_RSATR が返ります。

TMO_POL 指定時のみ、割り込みハンドラから実行が可能です。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_RSATR	-11	hwos_set_mpri_operation関数でHWOS_DISABLE_MPRIを指定時に、TA_MPRI属性で生成したメールボックスからメッセージを受信（タイムアウトあり）した
E_PAR	-17	パラメータエラー（ppk_msgが不正、あるいはtmoutが不正）
E_ID	-18	不正ID番号（mbxidが不正、あるいは使用できない）
E_CTX	-25	CPUロック状態/ディスパッチ禁止状態/割り込みハンドラから実行した
E_NOEXS	-42	オブジェクト未生成
E_RLWAI	-49	待ち状態の強制解除（待ち状態の間にrel_waiを受け付け）
E_TMOUT	-50	タイムアウト、またはポーリング失敗
E_DLT	-51	待ちオブジェクトの削除（待ち状態の間に対象メールボックスが削除）

4.6　拡張同期通信機能（ミューテックス）

拡張同期通信機能（ミューテックス）として提供しているサービスコールの一覧を示します。

表 4.11　拡張同期通信機能（ミューテックス）

サービスコール名	機能	発行有効範囲
del_mtx	ミューテックスの削除	タスク
loc_mtx	ミューテックスのロック	タスク
ploc_mtx	ミューテックスのロック（ポーリング）	タスク
tloc_mtx	ミューテックスのロック（タイムアウトあり）	タスク
unl_mtx	ミューテックスのロック解除	タスク

ミューテックス機能の仕様を以下に示します。

表 4.12　ミューテックス機能の仕様

項番	項目	内容
1	ミューテックスID	1～128（セマフォと合わせて）
2	サポート属性	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

del_mtx

概要

ミューテックスの削除

C 言語形式

ER del_mtx(ID mtxid);

パラメータ

I/O	パラメータ	説明	
I	ID mtxid	削除対象のミューテックスのID	
		数値（1～128）	対象ミューテックスID

機能

mtxid で指定された ID を持つミューテックスを削除します。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_ID	-18	不正ID番号（mtxidが不正、あるいは使用できない）
E_CTX	-25	CPUロック状態/割り込みハンドラから実行した
E_NOEXS	-42	オブジェクト未生成

loc_mtx

概要

ミューテックスのロック

C 言語形式

ER loc_mtx(ID mtxid);

パラメータ

I/O	パラメータ	説明
I	ID mtxid	ロック対象のミューテックスのID 数値（1～128）：対象ミューテックスID

機能

mtxid で指定されるミューテックスをロックします。

対象ミューテックスが他のタスクにロックされている場合は、自タスクをミューテックス待ち状態に移行させます。

loc_mtx(mtxid)は、tloc_mtx(mtxid, TMO_FEVR)と同一です。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_ID	-18	不正ID番号（mtxidが不正、あるいは使用できない）
E_CTX	-25	CPUロック状態/ディスパッチ禁止状態/割り込みハンドラから実行した
E_ILUSE	-28	サービスコール不正使用（既に自タスクがロックしている）
E_NOEXS	-42	オブジェクト未生成
E_RLWAI	-49	待ち状態の強制解除（待ち状態の間にrel_waiを受け付け）
E_DLT	-51	待ちオブジェクトの削除（待ち状態の間に対象ミューテックスが削除）

ploc_mtx

概要

ミューテックスのロック（ポーリング）

C 言語形式

ER ploc_mtx(ID mtxid);

パラメータ

I/O	パラメータ	説明
I	ID mtxid	ロック対象のミューテックスのID 数値（1～128）：対象ミューテックスID

機能

mtxid で指定されるミューテックスをロックします。

対象ミューテックスが他のタスクにロックされている場合は、自タスクをミューテックス待ち状態には移行させず、ポーリング失敗となります。

ploc_mtx(mtxid)は、tloc_mtx(mtxid, TMO_POL)と同一です。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_ID	-18	不正ID番号（mtxidが不正、あるいは使用できない）
E_CTX	-25	CPUロック状態/割り込みハンドラから実行した
E_ILUSE	-28	サービスコール不正使用（既に自タスクがロックしている）
E_NOEXS	-42	オブジェクト未生成
E_TMOUT	-50	ポーリング失敗

tloc_mtx

概要

ミューテックスのロック（タイムアウトあり）

C 言語形式

ER tloc_mtx(ID mtxid, TMO tmout);

パラメータ

I/O	パラメータ		説明
I	ID	mtxid	ロック対象のミューテックスのID 数値（1～128）：対象ミューテックスID
I	TMO	tmout	タイムアウト指定（単位はms） TMO_POL（0）：ポーリング（ploc_mtx()と同等の処理となる） TMO_FEVR（-1）：永久待ち（loc_mtx()と同等の処理となる） 数値：待ち時間

機能

mtxid で指定されるミューテックスをロックします。

対象ミューテックスが他のタスクにロックされている場合は、自タスクをミューテックス待ち状態に移行させます。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_PAR	-17	パラメータエラー（tmoutが不正）
E_ID	-18	不正ID番号（mtxidが不正、あるいは使用できない）
E_CTX	-25	CPUロック状態/ディスパッチ禁止状態/割り込みハンドラから実行した （ディスパッチ禁止状態でもTMO_POL指定時は実行可能）
E_ILUSE	-28	サービスコール不正使用（既に自タスクがロックしている）
E_NOEXS	-42	オブジェクト未生成
E_RLWAI	-49	待ち状態の強制解除（待ち状態の間にrel_waiを受け付け）
E_TMOUT	-50	タイムアウト
E_DLT	-51	待ちオブジェクトの削除（待ち状態の間に対象ミューテックスが削除）

unl_mtx

概要

ミューテックスのロック解除

C 言語形式

ER unl_mtx(ID mtxid);

パラメータ

I/O	パラメータ	説明
I	ID mtxid	ロック解除対象のミューテックスのID 数値（1～128）：対象ミューテックスID

機能

mtxid で指定されるミューテックスをロック解除します。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_ID	-18	不正ID番号（mtxidが不正、あるいは使用できない）
E_CTX	-25	CPUロック状態/割り込みハンドラから実行した
E_ILUSE	-28	サービスコール不正使用（対象ミューテックスをロックしていない）
E_NOEXS	-42	オブジェクト未生成

制限事項

ミューテックスをロックしたままタスク終了（ext_tsk や ter_tsk）した場合、本来の仕様ではミューテックスのロック解除処理が行われますが、ハードウエアリアルタイム OS では行われません。必ずロック解除してからタスクを終了してください。

4.7　システム時刻管理

システム時刻管理機能として提供しているサービスコールの一覧を示します。

表 4.13　システム時刻管理機能

サービスコール名	機能	発行有効範囲
set_tim	システム時刻の設定	タスク、非タスク
get_tim	システム時刻の参照	タスク、非タスク

システム時刻管理の仕様を以下に示します。

表 4.14　システム時刻管理の仕様

項番	項目	内容
1	システム時刻値	符号なし32 ビット
2	システム時刻の単位※	デフォルト1 [ms] （10[us]～100[ms]で1[us]単位で設定可能）
3	システム時刻初期値（初期起動時）	0x00000000

※：Tick 周期の事。HW-RTOS のセットアップ前に hwos_set_tick_time 関数をコールする事で変更可能

set_tim

概要

システム時刻の設定

C 言語形式

ER set_tim(SYSTIM *p_systim);

パラメータ

I/O	パラメータ	説明
I	SYSTIM　*p_systim	システム時刻に設定する時刻情報へのポインタ

機能

現在のシステム時刻を p_systim で指定される時刻に設定します。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_PAR	-17	パラメータエラー（p_systimが不正）
E_CTX	-25	CPUロック状態から実行した

get_tim

概要

システム時刻の参照

C 言語形式

ER get_tim(SYSTIM *p_systim);

パラメータ

I/O	パラメータ	説明
O	SYSTIM　*p_systim	現在のシステム時刻情報格納領域へのポインタ

機能

現在のシステム時刻を呼び出し、p_systim に返します。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_PAR	-17	パラメータエラー（p_systimが不正）
E_CTX	-25	CPUロック状態から実行した

4.8　システム状態管理機能

システム状態管理機能として提供しているサービスコールの一覧を示します。

表 4.15　システム状態管理機能

サービスコール名	機能	発行有効範囲
rot_rdq	タスク優先順位の回転	タスク、非タスク
irot_rdq	タスク優先順位の回転	非タスク
get_tid	実行状態のタスクIDの参照	タスク、非タスク
iget_tid	実行状態のタスクIDの参照	非タスク
loc_cpu	CPUロック状態への移行	タスク
unl_cpu	CPUロック状態の解除	タスク
sns_loc	CPUロック状態の参照	タスク、非タスク
dis_dsp	ディスパッチの禁止	タスク
ena_dsp	ディスパッチの許可	タスク

rot_rdq / irot_rdq

概要

タスク優先順位の回転

C 言語形式

ER rot_rdq(PRI tskpri) ;

ER irot_rdq(PRI tskpri) ;

パラメータ

I/O	パラメータ	説明
I	PRI tskpri	優先順位を回転する対象の優先度 TPRI_SELF（0）：自タスクのベース優先度を対象優先度とする 数値（1～15）：優先順位を回転する対象の優先度

機能

tskpri で指定される優先度のタスクの優先順位を回転します。具体的には、tskpri で指定された優先度に対応したレディキューの先頭タスクを最後尾につなぎかえ、次につながれているタスクに実行を切り替えます。

tskpri=TPRI_SELF（0）を指定すると、自タスクのベース優先度のレディキューを回転します。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_PAR	-17	パラメータエラー （tskpriが不正、あるいは割り込みハンドラからTPRI_SELFを指定した）
E_CTX	-25	CPUロック状態から実行した タスクから実行した（irot_rdqのみ）

get_tid / iget_tid

概要

実行状態のタスク ID の参照

C 言語形式

ER get_tid(ID *p_tskid) ;
ER iget_tid(ID *p_tskid) ;

パラメータ

I/O	パラメータ		説明
O	ID	*p_tskid	実行状態のタスクのID番号

機能

実行状態のタスクの ID 番号を参照し、p_tskid に返します。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_PAR	-17	パラメータエラー（p_tskidが不正）
E_CTX	-25	CPUロック状態から実行した タスクから実行した（iget_tidのみ）

制限事項

アイドルタスク実行中に実行した場合、TSK_NONE(=0)ではなく、アイドルタスクとして定義されたタスクの ID を返します。

loc_cpu

概要
CPU ロック状態への移行

C 言語形式
ER loc_cpu(void);

パラメータ
なし

機能
システム状態を CPU ロック状態へ移行します。CPU ロック状態では、カーネル管理割り込みとタスクのディスパッチが禁止されます。つまり、カーネル管理外割り込みハンドラを除くすべての処理プログラムに対して、排他的に処理を行うことができます。

CPU ロック状態では、発行可能なサービスコールは、loc_cpu , unl_cpu , sns_loc のみに制限されます。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_CTX	-25	割り込みハンドラから実行した

制限事項
割り込みハンドラから実行できません。

unl_cpu

概要
CPU ロック状態の解除

C 言語形式
ER unl_cpu(void);

パラメータ
なし

機能
CPU ロック状態を解除します。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_CTX	-25	割り込みハンドラから実行した

制限事項
割り込みハンドラから実行できません。

sns_loc

概要
CPU ロック状態の参照

C 言語形式
```
BOOL sns_loc(void);
```

パラメータ
なし

機能
CPU ロック状態を獲得します。

戻り値

マクロ	数値	意味
TRUE	1	CPUロック状態
FALSE	0	CPU非ロック状態

dis_dsp

概要
ディスパッチの禁止

C 言語形式
ER dis_dsp(void) ;

パラメータ
なし

機能
ディスパッチ禁止状態に移行します。

ディスパッチ禁止状態では、タスクのスケジューリングが禁止されるため、他のタスクに対して排他的に処理を行うことができます。

本サービスコールでは、禁止要求のキューイングが行われません。このため、ディスパッチ禁止状態で呼び出された場合には、何も処理は行わず、エラーとしても扱いません。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_CTX	-25	CPUロック状態/割り込みハンドラから実行した

ena_dsp

概要
ディスパッチの許可

C 言語形式
ER ena_dsp(void) ;

パラメータ
なし

機能
ディスパッチ許可状態に移行します。

戻り値

マクロ	数値	意味
E_OK	0	正常終了
E_CTX	-25	CPUロック状態/割り込みハンドラから実行した

5.　オブジェクト静的生成

5.1　タスク生成

カーネルにより予約された static_task_table 配列にタスク情報を記すことにより、OS 起動時（スタートアップ・ルーチンでの hwos_setup()関数実行）にタスクを生成することができます。

static_task_table 配列は TSK_TBL 構造体にて定義されており、タスク ID と T_CTSK 構造体のメンバを羅列するようにして設定することができます。

tskid に TASK_TBL_END（-1）を設定すると、カーネルはテーブルの最終と見なします。

```
//-------------------------------------
// Task information
//-------------------------------------
const TSK_TBL static_task_table[] = {
// CRE_TSK( tskid,         {tskatr          , exinf, task,          itskpri, stksz, stk});
           {ID_TASK_INIT, {TA_HLNG | TA_ACT, 0,     (FP)init_task, 1,       0x400, NULL}},
           {ID_TASK_MAIN, {TA_HLNG | TA_ACT, 0,     (FP)main_task, 2,       0x400, NULL}},
           {ID_TASK_IDLE, {TA_HLNG | TA_ACT, 0,     (FP)idle_task,15,       0x100, NULL}},
           {TASK_TBL_END, {0,                0,     (FP)NULL,      0,           0, NULL}}
};
```

図 5.1　static_task_table 配列設定例

注意　本 OS ではカーネル内にアイドル・タスクを持っていないため、アプリケーション側で定義が必要です。図 5.2 にアイドル・タスクの定義例を示します。

```
void idle_task(int exinf)
{
    while (1) {
        __NOP();
    }
}
```

図 5.2　アイドル・タスク定義例

注意　同一プライオリティで複数のタスクを TA_ACT 指定で生成した場合、タスクがレディ状態になる順序は配列での定義順にはなりません。レディ状態になる順序を制御するには、TA_ACT を指定せずに生成し、sta_tsk サービスコールでタスクを起動してください。
タスクは静的生成のみ行えます。システム起動後に動的生成はできません。
また、少なくとも 1 つ以上のタスクが TA_ACT 指定されている必要があります

5.2　セマフォ生成

カーネルにより予約された static_semaphore_table 配列にセマフォ情報を記すことにより、OS 起動時（スタートアップ・ルーチンでの hwos_setup()関数実行）にセマフォを生成することができます。

static_semaphore_table 配列は SEM_TBL 構造体にて定義されており、セマフォ ID と T_CSEM 構造体のメンバを羅列するようにして設定することができます。

semid に SEMAPHORE_TBL_END（-1）を設定すると、カーネルはテーブルの最終と見なします。

```
//-------------------------------------
// Semaphore information
//-------------------------------------
const SEM_TBL static_semaphore_table[] = {
// CRE_SEM( semid,              {sematr,    isemcnt, maxsem});
           {ID_APL_SEM1,        {TA_TFIFO, 0,        1}},
           {SEMAPHORE_TBL_END, {0,         0,        0}}
};
```

図 5.3　static_semaphore_table 配列設定例

5.3　イベントフラグ生成

カーネルにより予約された static_eventflag_table 配列にイベントフラグ情報を記すことにより、OS 起動時（スタートアップ・ルーチンでの hwos_setup()関数実行）にイベントフラグを生成することができます。

static_eventflag_table 配列は FLG_TBL 構造体にて定義されており、イベントフラグ ID と T_CFLG 構造体のメンバを羅列するようにして設定することができます。

flgid に EVENTFLAG_TBL_END（-1）を設定すると、カーネルはテーブルの最終と見なします。

```
//-------------------------------------
// Eventflag information
//-------------------------------------
const FLG_TBL static_eventflag_table[] = {
// CRE_FLG( flgid,              {flgatr,             iflgptn});
           {ID_APL_FLG1,        {TA_TFIFO | TA_WMUL | TA_CLR, 0}},
           {EVENTFLAG_TBL_END, {0,                  0}}
};
```

図 5.4　static_eventflag_table 配列設定例

注意　TA_WSGL が指定された場合でも、TA_WMUL として動作します。

5.4　メールボックス生成

カーネルにより予約された static_mailbox_table 配列にメールボックス情報を記すことにより、OS 起動時（スタートアップ・ルーチンでの hwos_setup()関数実行）にメールボックスを生成することができます。

static_mailbox_table 配列は MBX_TBL 構造体にて定義されており、メールボックス ID と T_CMBX 構造体のメンバを羅列するようにして設定することができます。

mbxid に MAILBOX_TBL_END（-1）を設定すると、カーネルはテーブルの最終と見なします。

```
//-----------------------------------
// Mailbox information
//-----------------------------------
const MBX_TBL static_mailbox_table[] = {
// CRE_MBX(  mbxid,           {mbxatr,    maxmpri, mprihd});
            {ID_APL_MBX1,     {TA_TFIFO | TA_MFIFO,  0,       NULL}},
            {MAILBOX_TBL_END, {0,         0,       NULL}}
};
```

図 5.5　static_mailbox_table 配列設定例

注意　メールボックスの TA_MPRI 属性は、デフォルトでは使用できません。
使用する場合は、8 ユーティリティ関数の「hwos_set_mpri_operation」を参照してください。

5.5　ミューテックス生成

カーネルにより予約された static_mutex_table 配列にミューテックス情報を記すことにより、OS 起動時（スタートアップ・ルーチンでの hwos_setup()関数実行）にミューテックスを生成することができます。

static_mutex_table 配列は MBX_TBL 構造体にて定義されており、ミューテックス ID と T_CMTX 構造体のメンバを羅列するようにして設定することができます。

mtxid に MUTEX_TBL_END（-1）を設定すると、カーネルはテーブルの最終と見なします。

```
//-----------------------------------
// Mutex information
//-----------------------------------
const MTX_TBL static_mutex_table[] = {
// CRE_MTX( mtxid,          {mtxatr,    ceilpri});
           {ID_APL_MTX1,    {TA_TFIFO, 0}},
           {MUTEX_TBL_END,  {0,         0}}
};
```

図 5.6　static_mutex_table 配列設定例

注意　ミューテックスの属性の TA_INHERIT または TA_CEILING は対応していません。
設定された場合は、無視されます。

5.6　割り込みハンドラ定義

カーネルにより予約された static_interrupt_table 配列に割り込みハンドラ情報を記すことにより、OS 起動時（スタートアップ・ルーチンでの hwos_setup()関数実行）に OS の監視下にある割り込み（カーネル管理割り込み）ハンドラを生成することができます。

本定義により生成された割り込みハンドラ内では、OS のサービスコールを発行することができます。

static_interrupt_table 配列は INT_TBL 構造体にて定義されており、割り込みハンドラ番号と T_DINH 構造体のメンバを羅列するようにして設定することができます。

inhno に INT_TBL_END（0xFFFFFFFF）を設定すると、カーネルはテーブルの最終と見なします。

```
//-------------------------------------
// Interrupt handler information
//-------------------------------------
const INT_TBL static_interrupt_table[] = {
// DEF_INH( inhno,        {inhatr,    inthdr});
           {INTPZ0_IRQn, {TA_HLNG, (FP)int_task}},
           {INT_TBL_END, {0,        (FP)NULL}}
};
```

図 5.7　static_interrupt_table 配列設定例

注意　カーネル管理割り込みは、NVIC で優先度はすべて 15(最低優先度)に設定されます。
したがって、割り込みの横取り（多重割り込み）は発生しません。

以下に割り込みハンドラのコード記述例を示します。

```
void int_task(void)
{
    iset_flg(ID_APL_FLG1, 0x0001);
};
```

図 5.8　割り込みハンドラのコード記述例

6.　Hardware ISR

カーネルにより予約された static_hwisr_table 配列に Hardware ISR 情報を記すことにより、OS 起動時（スタートアップ・ルーチンでの hwos_setup()関数実行）に Hardware ISR の登録をすることができます。

Hardware ISR は、割り込みが発生したときに、事前に登録しておいたサービスコールをハードウェア・リアルタイム OS が自動実行する機能です。本機能を使用することにより、CPU が割り込み処理を行うオーバヘッドを削減することができます。

Hardware ISR が実行できるサービスコールは、表 1.2 にも示したとおり、set_flg()、sig_sem()、rel_wai()、wup_tsk()の 4 種類です。

Hardware ISR は、割り込み信号の立ち上がりで検出されます。

static_hwisr_table 配列は HWISR_TBL 構造体にて定義されており、32 個の Hardware ISR を設定することができます。例えば、図 6.1 の 1 つ目の設定例では、INTPZ1 割り込みが発生したときにハードウェア・リアルタイム OS が自動的に「set_flg(ID_APL_FLG1, 0x0001);」サービスコールを発行します。2 つ目の設定例では、INTPZ2 割り込みが発生したときにハードウェア・リアルタイム OS が自動的に「wup_tsk(ID_TASK_MAIN);」サービスコールを発行します。

inhno に HWISR_TBL_END（0xFFFFFFFF）を設定すると、カーネルはテーブルの最終と見なします。

```
//-------------------------------------
// Hardware ISR
//-------------------------------------
const HWISR_TBL static_hwisr_table[] = {
// inhno,          hwisr_syscall, id,            setptn
  {INTPZ1_IRQn,    HWISR_SET_FLG, ID_APL_FLG1,   0x0001},
  {INTPZ2_IRQn,    HWISR_WUP_TSK, ID_TASK_MAIN, 0},
  {HWISR_TBL_END, 0,             0,             0}
};
```

図 6.1　static_hwisr_table 配列設定例

7. 割り込み管理機能

7.1 割り込みの種類

μITRON4.0 の割り込みは、カーネル管理割り込みとカーネル管理外割り込みに分類されます。これに加えて、HW-RTOS 特有の機能として Hardware ISR があります。

- カーネル管理割り込み
  割り込みハンドラを OS に登録した割り込みを、カーネル管理割り込みといいます。
  static_interrupt_table にハンドラを登録することで実現します。
  カーネル管理割り込みハンドラでは、サービス・コールを呼び出すことができます。
  サービス・コール処理中にカーネル管理割り込みが発生した場合、カーネル管理割り込みを受け付け可能となるまで割り込み受理が遅延されます。
  カーネル管理割り込みの割り込み優先度は、15 になるようにカーネルが設定します。

- カーネル管理外割り込み
  カーネル動作中にもマスクされない割り込みを、カーネル管理外割り込みといいます。
  カーネル管理外割り込みハンドラでは、サービス・コールを呼び出してはなりません。
  サービス・コール処理中にカーネル管理外割り込みが発生した場合、直ちに割り込み要求が受理されるため、カーネル処理に依存しない高速な割り込み応答が可能です。
  カーネル管理外割り込みの割り込み優先度は、14 より高優先（0～13）になるようにユーザーが設定してください。

- Hardware ISR
  HW-RTOS 単体で処理が完結する ISR です。
  特定のサービスコールしか実行できませんが、CPU を介さずに処理できます。(詳細は 6 章を参照)

7.2 CPU 例外の扱い

Cortex-M の例外番号 1～15 のうち、例外番号 11 のSVCall 例外は、カーネルが使用します。
それ以外は、カーネル管理外割り込みの扱いとなります。

7.3 多重割り込み

カーネル管理割り込みの割り込み優先度はすべて同一優先度に設定されます。したがって、割り込みの横取り（多重割り込み）は、発生しません。

7.4 割り込みハンドラ

割り込みハンドラは、割り込みが発生した際に起動される割り込み処理専用ルーチンです。
カーネルは、割り込みハンドラの前に行うべき処理を行った後、割り込みハンドラを起動します。
割り込みハンドラは、初期化時に登録することができます。「5.6 割り込みハンドラ定義」を参照してください。

8.　ユーティリティ関数

rin_hwos_get_version

概要

バージョン情報の取得

C 言語形式

```
char *rin_hwos_get_version(uint8_t mode);
```

パラメータ

I/O	パラメータ	説明
I	uint8_t　mode	出力するバージョンの様式 0：バージョンのみ 1：ビルド日時付きバージョン

機能

OS ライブラリのバージョン情報を、パラメータで指定した様式に合わせて取得します。

戻り値

バージョン情報の文字列

使用例

```
printf("R-IN HWRTOS lib ver%s¥n",rin_hwos_get_version(0));
```

hwos_set_mpri_operation

概要

TA_MPRI 属性の使用設定

C 言語形式

void hwos_set_mpri_operation(int32_t flag);

パラメータ

I/O	パラメータ	説明
I	int32_t　flag	TA_MPRI属性の使用設定 HWOS_DISABLE_MPRI　（0）：無効（デフォルト） HWOS_ENABLE_MPRI　（1）：有効

機能

TA_MPRI 属性の Mailbox の使用有無を選択します。

本 API をコールしない場合は、デフォルトで無効が選択されます。

本 API はメールの送受信を行う前に呼び出す必要があります。メール動作中に本 API を発行した場合の動作は不定ですので、発行しないでください。

TA_MPRI 属性の Mailbox を使用可能にした場合、Mailbox のキューに 1 つ以上のメッセージが残っている状態で 256 回以上送信 API を発行したとき、E_QOVR を返します。Mailbox が空になるまで同じエラーを返すため、ユーザーは Mailbox が空になるまで受信する必要が生じます。

項目	パラメータ	
	HWOS_DISABLE_MPRI	HWOS_ENABLE_MPRI
TA_MPRI 属性の Mailbox生成	成功します	成功します
TA_MPRI 属性の Mailbox への メール送信	失敗します APIの戻り値：E_RSATR (-11)	Mailbox にメールが 1 つ以上ある状態でメールの送信を繰り返し行うと 256 回目に API が失敗します。 APIの戻り値：E_QOVR (-43)
TA_MPRI 属性の Mailbox からの メール受信	失敗します APIの戻り値：E_RSATR (-11)	成功します
補足	ー	APIがE_QOVRを返したときは、Mailboxの中身が空になるまでメール送信は失敗（E_QOVR）します。

戻り値

なし

hwos_set_tick_time

概要

Tick Time の設定

C 言語形式

int32_t hwos_set_tick_time(uint32_t tick_time);

パラメータ

I/O	パラメータ	説明
I	uint32_t　tick_time	HW-RTOSのTick Timeを[μs]単位で指定します。 指定可能な範囲は10 ~100000（10[μs]~100[ms]）になります。

機能

HW-RTOS の Tick Time を設定します。

本 API は HW-RTOS のセットアップ前（hwos_setup 関数コール前）に呼び出してください。

本 API をコールしない場合、又は無効なパラメータが指定されたときは、HW-RTOS の Tick Time は 1[ms]になります。

戻り値

マクロ	数値	意味
TRUE	1	HW-RTOSのTick Time の設定が成功
FALSE	0	HW-RTOSのTick Time の設定が失敗（パラメータが不正）

注意事項

HW-RTOS のセットアップ完了後は、本 API を呼び出し戻り値に TRUE が返されても、Tick Time の変更は行われません。

9.　開発ツール依存設定

本章では、開発ツール毎の差分について説明します。
IAR では、スタートアップ中にコンパイラが用意しているライブラリを使用しています。

9.1　IAR

9.1.1　スタートアップ

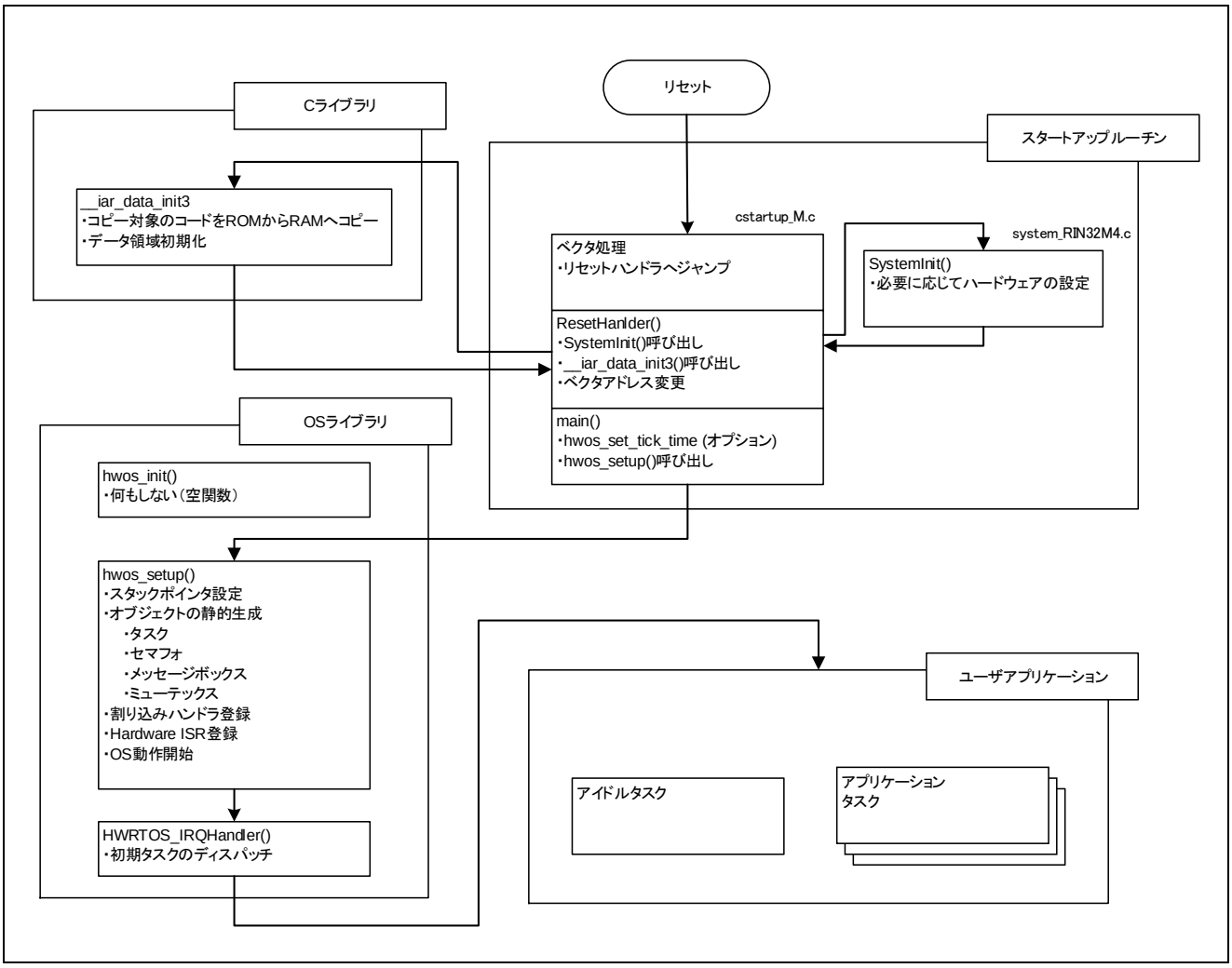

図 9.1　IAR 使用時のスタートアップ・ルーチン

9.1.2　スタック領域

OS 起動時（hwos_setup 実行後）のスタック領域の状態を以下に示します。図中の矢印は、ポインタの進行方向を示しています。

OS ライブラリは、MSP と PSP の 2 つのスタックポインタを使用しており、IAR コンパイラの場合、1 つの領域（CSTACK）を共有して使用します。

PSP は通常のタスク処理、MSP は割り込み等のタスク以外の処理で使用されます。PSP の初期値は、最初に起動するタスクのスタックポインタになります。タスクのディスパッチ時に、遷移先タスクのスタック領域終端に PSP を切り替えて使用します。

OS ライブラリは、セクションの先頭／最終アドレス値を得るために、下記のセクションのシンボルを参照しています。リンカ設定ファイル（*.icf）に、同じセクション名で定義するようにしてください。

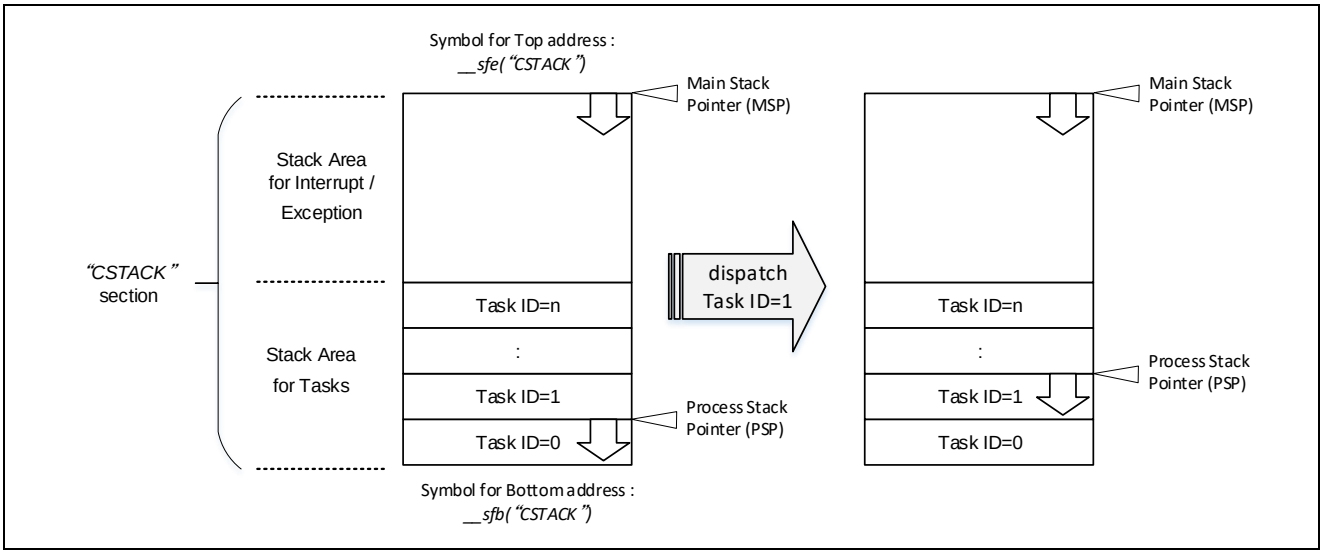

図 9.2　OS 起動時のスタック領域(IAR 使用時)

9.1.3　コンパイル・オプション

OS ライブラリ生成時のコンパイル・オプションを以下に示します。

--cpu=Cortex-M4F	ターゲットCPU　：Cortex-M4
--fpu=VFPv4_sp	FPUの型　：単精度浮動小数点
--endian=little	生成コードのエンディアン　：リトルエンディアン
-e	言語拡張を有効化
-Ohs	最適化レベル
--no_size_constraints	最適化時のコードサイズの制限を緩和

10.　リソース

10.1　H/W 資源

OS ライブラリが使用する H/W 資源を以下に示します。

表 10.1　H/W 使用資源

資源名	内容
HW-RTOS	Hardware Real-time OS
例外（割り込み）	SVCall（例外番号11）：SVC命令によるシステム・サービスの呼び出し INTHWRTOS（例外番号92）：HW-RTOS割り込み

備考：　Tick 用タイマは、HW-RTOS の機能を使用し、内蔵周辺機能のタイマは、使用しません。

10.2　メモリ

OS ライブラリが使用するメモリリソースを以下に示します。

表 10.2　メモリ使用量

種類	サイズ [byte]		
	ARM	GNU	IAR
Code	—	—	7,240
RO Data	—	—	4
RW Data	—	—	0
ZI Data	—	—	3,744

上記の他に、スタック用のメモリが必要になります。「10.3 スタック」を参照してください。

10.3　スタック

スタックには、プロセススタックとメインスタックの 2 種類があります。

スタック使用量の算出方法を以下に示します。スタック領域の定義は、コンパイラにより異なります。「9. 開発ツール依存設定」を参照してください。

10.3.1　プロセススタックの算出方法

プロセススタックは、タスクで使用します。

以下の a+b が、1 つのタスクでの使用量になり、生成する全タスク分を合計します。（タスク生成用の static_task_table 配列で定義した stksz の合計です。）

a) タスク開始関数を起点とする関数コールツリーで、消費されるスタック最大値
b) タスクのコンテキスト・レジスタのサイズ。72 バイト。

10.3.2　メインスタックの算出方法

メインスタックは、非タスクの処理で使用します。

多重割り込みは発生しないので、各割り込み処理の使用量の最大値です。以下の各サイズを合計します。

a) 各割り込みハンドラ開始関数を起点とする関数コールツリーで、消費されるスタックの最大値
b) 割り込みハンドラ前処理のレジスタ退避。4 バイト。

改訂記録	R-IN32M4-CL3　プログラミング・マニュアル（OS 編）

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2019.10.30	ー	初版発行

[メ　モ]

R-IN32M4-CL3
プログラミング・マニュアル（OS編）

発行年月日　2019年 10月30日　　Rev.1.00

発行　　ルネサス エレクトロニクス株式会社
　　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

R-IN32M4-CL3
プログラミング・マニュアル（OS 編）

ルネサス エレクトロニクス株式会社

R18UZ0040JJ0102