User's Manual

RI850MP

リアルタイム・オペレーティング・システム
ユーザーズマニュアル　コーディング編

対象ツール

RI850MP

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　2011.04

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。

このマニュアルの使い方

対 象 者　このマニュアルは，V850マイクロコントローラの各製品の応用システムを設計，開発するユーザを対象としています。

目　　的　このマニュアルは，次の構成に示すルネサス エレクトロニクス製リアルタイムOS　RI850MPの機能をユーザに理解していただくことを目的としています。

構　　成　このマニュアルは，大きく分けて次の内容で構成しています。

読 み 方　このマニュアルの読者には，電気，論理回路，マイクロコンピュータ，C言語，アセンブラの一般知識を必要とします。

V850マイクロコントローラのハードウエア機能を知りたいとき
→各製品の**ユーザーズ・マニュアル**を参照してください。

凡　　例

データ表記の重み	：左が上位桁，右が下位桁
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またはXXXXB
	10進数…XXXX
	16進数…0xXXXX

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）　$2^{20} = 1024^2$

関連資料　このマニュアルを使用する場合は，次の資料もあわせてご覧ください。
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。

開発ツールの資料（ユーザーズ・マニュアル）

資　料　名		資料番号	
		和　文	英　文
RIシリーズ	起動編	R20UT0509J	R20UT0509E
	メッセージ編	R20UT0510J	R20UT0510E
RI78V4	コーディング編	R20UT0511J	R20UT0511E
	デバッグ編	R20UT0520J	R20UT0520E
	解析編	R20UT0513J	R20UT0513E
	内部構造編	R20UT0514J	R20UT0514E
RI850V4	コーディング編	R20UT0515J	R20UT0515E
	デバッグ編	R20UT0516J	R20UT0516E
	解析編	R20UT0517J	R20UT0517E
	内部構造編	R20UT0518J	R20UT0518E
RI850MP	コーディング編	このマニュアル	R20UT0519E
CubeSuite+統合開発環境	起動編	R20UT0545J	R20UT0545E
	78K0設計編	R20UT0546J	R20UT0546E
	78K0R設計編	R20UT0547J	R20UT0547E
	RL78設計編	R20UT0548J	R20UT0548E
	V850設計編	R20UT0549J	R20UT0549E
	R8C設計編	R20UT0550J	R20UT0550E
	78K0コーディング編	R20UT0551J	R20UT0551E
	RL78,78K0Rコーディング編	R20UT0552J	R20UT0552E
	V850コーディング編	R20UT0553J	R20UT0553E
	コーディング編（CXコンパイラ）	R20UT0554J	R20UT0554E
	R8Cコーディング編	R20UT0576J	R20UT0576E
	78K0ビルド編	R20UT0555J	R20UT0555E
	RL78,78K0Rビルド編	R20UT0556J	R20UT0556E
	V850ビルド編	R20UT0557J	R20UT0557E
	ビルド編（CXコンパイラ）	R20UT0558J	R20UT0558E
	R8Cビルド編	R20UT0575J	R20UT0575E
	78K0デバッグ編	R20UT0559J	R20UT0559E
	78K0Rデバッグ編	R20UT0560J	R20UT0560E
	RL78デバッグ編	R20UT0561J	R20UT0561E

注意　上記関連資料は，予告なしに内容を変更することがあります。設計などには，必ず最新の資料を使用してください。

〔メ　　モ〕

〔メ　　モ〕

〔メ　　モ〕

目　　次

第 1 章 概 説

1.1 概 要

RI850MP は，効率の良いリアルタイム処理環境，および，マルチタスク処理環境を提供するとともに，対象デバイスの制御機器分野における応用範囲を拡大することを目的として開発された“リアルタイム・マルチタスク OS”です。

また，実行環境に組み込んで使用することを前提として開発されているため，ROM 化を意識し，コンパクトな設計が行われています。

第２章　タスク管理機能

本章では，RI850MP が提供するタスク管理機能について解説しています。

2.1　概　　要

RI850MP におけるタスク管理機能では，タスクの状態を操作／参照するための機能を提供しています。

備考　RI850MP がタスク管理機能として提供しているサービス・コールについての詳細は，「13. 4. 1　タスク管理機能」を参照してください。

2.2　タスク

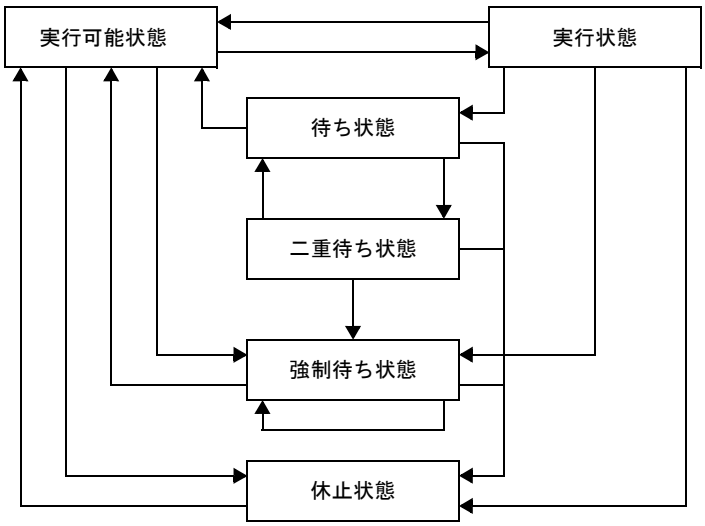
タスクは，他の処理プログラム（周期ハンドラ，割り込みハンドラ，CPU 例外ハンドラなど）とは異なり，RI850MP が提供しているサービス・コールを使用して明示的に操作しないかぎり実行されることのない処理プログラムです。

2.2.1　タスクの状態

タスクは，処理を実行するうえで必要となる資源の獲得状況，および事象の発生有無などにより，さまざまな状態へと遷移していきます。

なお，RI850MP では，タスクが取り得る状態を以下に示した 6 種類に分類し，管理しています。

図 2—1　タスクの状態遷移



- 休止状態

タスクとして起動されていない状態，またはタスクとしての処理を終了した際に遷移する状態です。

なお，休止状態のタスクは，RI850MP の管理下にありながらも，スケジューリング対象からは除外された状態です。

- 実行可能状態

処理を実行するうえで必要となる条件は整っているが，より高い優先度のタスクが処理を実行中のため，対象デバイスの使用権が割り当てられるのを待っている状態です。

- 実行状態

対象デバイスの使用権が割り当てられ，処理を実行中の状態です。

- 待ち状態

処理を実行するうえで必要となる条件が整わないため，処理の実行が中断した状態です。
なお，待ち状態からの処理再開は，処理の実行が中断した箇所からの再開となります。
RI850MP では，要求条件の種類により，待ち状態を以下に示す 9 種類に分類し，管理しています。

表 2—1　待ち状態の種類

状態種別	概要
起床待ち状態	slp_tsk，または tslp_tsk を発行した際，自タスクの起床要求ネスト・カウンタが 0x0 の場合に遷移する状態です。
時間経過待ち状態	dly_tsk を発行した際に遷移する状態です。
資源獲得待ち状態	wai_sem，または twai_sem を発行した際，対象セマフォから資源を獲得することができなかった場合に遷移する状態です。
イベントフラグ待ち状態	wai_flg，または twai_flg を発行した際，対象イベントフラグのビット・パターンが要求条件を満足していなかった場合に遷移する状態です。
データ送信待ち状態	snd_dtq，または tsnd_dtq を発行した際，対象データ・キューにデータを送信することができなかった場合に遷移する状態です。
データ受信待ち状態	rcv_dtq，または trcv_dtq を発行した際，対象データ・キューからデータを受信することができなかった場合に遷移する状態です。
メッセージ受信待ち状態	rcv_mbx，または trcv_mbx を発行した際，対象メールボックスからメッセージを受信することができなかった場合に遷移する状態です。
ミューテックス獲得待ち状態	loc_mtx，または tloc_mtx を発行した際，対象ミューテックスを獲得することができなかった場合に遷移する状態です。
固定長メモリ・ブロック獲得待ち状態	get_mpf，または tget_mpf を発行した際，対象固定長メモリ・プールから固定長メモリ・ブロックを獲得することができなかった場合に遷移する状態です。

- 強制待ち状態

  強制的に処理の実行を中断させられた状態です。

  なお，強制待ち状態からの処理再開は，処理の実行が中断した箇所からの再開となります。

- 二重待ち状態

  待ち状態と強制待ち状態が複合した状態です。

  なお，待ち状態が解除された際には強制待ち状態へ，強制待ち状態が解除された際には待ち状態へと遷移します。

2. 2. 2　タスクの優先度

タスクには，処理を実行するうえでの優先順位が割り付けられています。そこで，RI850MP では，実行可能な状態（実行可能状態，または実行状態）にあるタスクの優先度を参照し，その中から最も高い優先度を持つタスクを選び出し，対象デバイスの使用権を割り当てています。

なお，RI850MP では，タスクの優先度を以下に示した 2 種類に分類し，管理しています。

- 初期優先度

  コンフィギュレーション・ファイルで静的 API“CRE_TSK”を使用して定義した値となります。

- 現在優先度

  タスクの起動後，RI850MP が各種操作を実行する際に参照する優先度です。

備考 1. RI850MP におけるタスクの優先度は，その値が小さいほど，高い優先度であることを意味しています。

2. システム内で利用可能な優先度の範囲は，コンフィギュレーション・ファイルで静的 API“MAX_PRI”を使用して定義した値となります。

なお，静的 API“MAX_PRI”についての詳細は，「B. 3. 5　最大優先度情報」を参照してください。

2. 2. 3　タスクの基本型

タスクを記述する場合，VP_INT 型の引数を 1 つ持った void 型の関数として記述します。

なお，引数 *exinf* には，“タスク情報で定義した拡張情報”が設定されます。

以下に，タスクを C 言語で記述する場合の基本型を示します。

【CX 対応版】

```
#include    <kernel.h>
#pragma     rtos_task   task
void
task(VP_INT exinf){
        ..........................
        ..........................
        ext_tsk();
}
```

【CCV850E 対応版】

```
#include    <kernel.h>
void
task(VP_INT exinf){
        ……
        ……
        ext_tsk();
}
```

2. 2. 4　タスクの生成

RI850MP では，タスクの静的な生成のみをサポートしています。したがって，処理プログラムからサービス・コールを発行して動的に生成することはできません。

なお，タスクの静的な生成とは，コンフィギュレーション・ファイルで静的 API“CRE_TSK”を使用してタスクに関する情報を定義することをいいます。

備考　静的 API“CRE_TSK”についての詳細は，「B. 5. 1　タスク情報」を参照してください。

第 3 章　タスク付属同期機能

本章では，RI850MP が提供するタスク付属同期機能について解説しています。

3.1　概　　要

RI850MP におけるタスク付属同期機能では，タスクの状態操作にともなうタスク間の同期を実現するための機能を提供しています。

備考　RI850MP がタスク付属同期機能として提供しているサービス・コールについての詳細は，「13. 4. 2　タスク付属同期機能」を参照してください。

第４章　同期・通信機能

本章では，RI850MP が提供する同期・通信機能について解説しています。

4.1　概　　要

RI850MP における同期・通信機能では，タスク間の同期／通信を実現するための機能としてセマフォ，イベントフラグ，データ・キュー，メールボックスを提供しています。

備考　RI850MP が同期・通信機能として提供しているサービス・コールについての詳細は，「13. 4. 3　同期・通信機能（セマフォ）」，「13. 4. 4　同期・通信機能（イベントフラグ）」，「13. 4. 5　同期・通信機能（データ・キュー）」，「13. 4. 6　同期・通信機能（メールボックス）」を参照してください。

4.2　セマフォ

セマフォは，タスク間の同期を実現するための機能です。

備考　RI850MP では，非負数の計数型セマフォを提供しています。

4.2.1　セマフォの生成

RI850MP では，セマフォの静的な生成のみをサポートしています。したがって，処理プログラムからサービス・コールを発行して動的に生成することはできません。

なお，セマフォの静的な生成とは，コンフィギュレーション・ファイルで静的 API“CRE_SEM”を使用してセマフォに関する情報を定義することをいいます。

備考　静的 API“CRE_SEM”についての詳細は，「B. 5. 2　セマフォ情報」を参照してください。

4.3　イベントフラグ

イベントフラグは，タスク間の同期を実現するための機能です。

備考　RI850MP では，32 ビット幅のイベントフラグを提供しています。

4.3.1　イベントフラグの生成

RI850MP では，イベントフラグの静的な生成のみをサポートしています。したがって，処理プログラムからサービス・コールを発行して動的に生成することはできません。

なお，イベントフラグの静的な生成とは，コンフィギュレーション・ファイルで静的 API“CRE_FLG”を使用してイベントフラグに関する情報を定義することをいいます。

備考　静的 API“CRE_FLG”についての詳細は，「B. 5. 3　イベントフラグ情報」を参照してください。

4. 4　データ・キュー

データ・キューは，タスク間の同期／通信を実現するための機能です。

備考　RI850MP では，規定サイズ（4 バイト）のデータを送信／受信することが可能なデータ・キューを提供しています。

4. 4. 1　データ・キューの生成

RI850MP では，データ・キューの静的な生成のみをサポートしています。したがって，処理プログラムからサービス・コールを発行して動的に生成することはできません。

なお，データ・キューの静的な生成とは，コンフィギュレーション・ファイルで静的 API“CRE_DTQ”を使用してデータ・キューに関する情報を定義することをいいます。

備考　静的 API“CRE_DTQ”についての詳細は，「B. 5. 4　データ・キュー情報」を参照してください。

4. 5　メールボックス

メールボックスは，タスク間の同期／通信を実現するための機能です。

備考　RI850MP では，任意サイズのデータを送信／受信することが可能なメールボックスを提供しています。

4. 5. 1　メールボックスの生成

RI850MP では，メールボックスの静的な生成のみをサポートしています。したがって，処理プログラムからサービス・コールを発行して動的に生成することはできません。

なお，メールボックスの静的な生成とは，コンフィギュレーション・ファイルで静的 API“CRE_MBX”を使用してメールボックスに関する情報を定義することをいいます。

備考　静的 API“CRE_MBX”についての詳細は，「B. 5. 5　メールボックス情報」を参照してください。

第 5 章　拡張同期・通信機能

本章では，RI850MP が提供する拡張同期・通信機能について解説しています。

5. 1　概　　要

RI850MP における拡張同期・通信機能では，タスク間の同期を実現するための機能としてミューテックスを提供しています。

備考　RI850MP が拡張同期・通信機能として提供しているサービス・コールについての詳細は，「13. 4. 7　拡張同期・通信機能」を参照してください。

5. 2　ミューテックス

ミューテックスは，タスク間の同期を実現するための機能です。

5. 2. 1　ミューテックスの生成

RI850MP では，ミューテックスの静的な生成のみをサポートしています。したがって，処理プログラムからサービス・コールを発行して動的に生成することはできません。

なお，ミューテックスの静的な生成とは，コンフィギュレーション・ファイルで静的 API“CRE_MTX”を使用してミューテックスに関する情報を定義することをいいます。

備考　静的 API“CRE_MTX”についての詳細は，「B. 5. 6　ミューテックス情報」を参照してください。

第 6 章　メモリ・プール管理機能

本章では，RI850MP が提供するメモリ・プール管理機能について解説しています。

6. 1　概　　要

RI850MP におけるメモリ・プール管理機能では，処理プログラムから動的にメモリ領域の獲得／開放を実現するための機能として固定長メモリ・プールを提供しています。

備考　RI850MP がメモリ・プール管理機能として提供しているサービス・コールについての詳細は，「13. 4. 8　メモリ・プール管理機能」を参照してください。

6. 2　固定長メモリ・プール

固定長メモリ・プールは，処理プログラムから動作的に固定サイズのメモリ・ブロックを獲得／返却することが可能なメモリ領域です。

6. 2. 1　固定長メモリ・プールの生成

RI850MP では，固定長メモリ・プールの静的な生成のみをサポートしています。したがって，処理プログラムからサービス・コールを発行して動的に生成することはできません。

なお，固定長メモリ・プールの静的な生成とは，コンフィギュレーション・ファイルで静的 API“CRE_MPF”を使用して固定長メモリ・プールに関する情報を定義することをいいます。

備考　静的 API“CRE_MPF”についての詳細は，「B. 5. 7　固定長メモリ・プール情報」を参照してください。

第 7 章　時間管理機能

本章では，RI850MP が提供する時間管理機能について解説しています。

7.1　概　　要

RI850MP における時間管理機能では，一定周期で発生するタイマ割り込みを利用することにより，時間に依存した処理を実現するための機能を提供しています。

備考　RI850MP が時間管理機能として提供しているサービス・コールについての詳細は，「13. 4. 9　時間管理機能」を参照してください。

7.2　タイマ割り込み

RI850MP では，一定周期で発生するタイマ割り込みを利用することにより，時間に依存した処理（システム時刻の更新，タスクのタイムアウト，周期ハンドラの起動など）を実現しています。

7.2.1　タイマ割り込みの登録

RI850MP では，タイマ割り込みの静的な登録のみをサポートしています。したがって，処理プログラムからサービス・コールを発行して動的に登録することはできません。

なお，タイマ割り込みの静的な登録とは，コンフィギュレーション・ファイルで静的 API“DEF_TIM”，および“CLK_INTNO”を使用してタイマ割り込みに関する情報を定義することをいいます。

備考 1.　静的 API“DEF_TIM”についての詳細は，「B. 3. 2　基本クロック周期情報」を参照してください。

2.　静的 API“CLK_INTNO”についての詳細は，「B. 3. 3　タイマ割り込み情報」を参照してください。

7.3　周期ハンドラ

周期ハンドラは，一定の時間が経過するたびに呼び出される周期処理専用ルーチンです。

なお，RI850MP では，周期ハンドラを“タスクとは独立したもの（非タスク）”として位置づけています。このため，一定の時間が経過した際には，システム内で最高優先度を持つタスクが処理を実行中であっても，その処理は中断され，周期ハンドラに制御が移ります。

7.3.1　周期ハンドラの状態

RI850MP では，周期ハンドラが取り得る状態を以下に示した 2 種類に分類し，管理しています。

- 停止状態

一定の時間が経過しても周期ハンドラの起動が行われない状態です。

- 動作状態

一定の時間が経過した際，周期ハンドラの起動が行われる状態です。

なお，停止状態から動作状態へと遷移した際の 1 回目の起動が行われるまでの間隔は，対象周期ハンドラに TA_PHS 属性（保存有無：起動位相を保存）が付与されているか否かにより異なります。

7. 3. 2 周期ハンドラの基本型

周期ハンドラを記述する場合，VP_INT 型の引数を 1 つ持った void 型の関数として記述します。

なお，引数 *exinf* には，“周期ハンドラ情報で定義した拡張情報”が設定されます。

以下に，周期ハンドラを C 言語で記述する場合の基本型を示します。

```
#include    <kernel.h>
void
cychdr(VP_INT exinf){
        ……
        ……
        return;
}
```

7. 3. 3 周期ハンドラの登録

RI850MP では，周期ハンドラの静的な登録のみをサポートしています。したがって，処理プログラムからサービス・コールを発行して動的に登録することはできません。

なお，周期ハンドラの静的な登録とは，コンフィギュレーション・ファイルで静的 API“CRE_CYC”を使用して周期ハンドラに関する情報を定義することをいいます。

備考 静的 API“CRE_CYC”についての詳細は，「B. 5. 8 周期ハンドラ情報」を参照してください。

第 8 章　システム状態管理機能

本章では，RI850MP が提供するシステム状態機能について解説しています。

8. 1　概　　要

RI850MP におけるシステム状態機能では，システムの状態を操作／参照するための機能を提供しています。

備考　RI850MP がシステム状態管理機能として提供しているサービス・コールについての詳細は，「13. 4. 10　システム状態管理機能」を参照してください。

第 9 章　割り込み管理機能

本章では，RI850MP が提供する割り込み管理機能について解説しています。

9. 1　概　　要

RI850MP における割り込み管理機能では，割り込みが発生した際に起動する割り込みハンドラに関連した機能を提供しています。

備考　RI850MP が割り込み管理機能として提供しているサービス・コールについての詳細は，「13. 4. 11　割り込み管理機能」を参照してください。

9. 2　ユーザ・オウン・コーディング部

RI850MP では，割り込み管理機能のうち，ユーザの実行環境に依存した処理をユーザ・オウン・コーディング部として切り出し，サンプル・ソース・ファイルを提供しています。

これにより，様々な実行環境への移行性を向上させるとともに，カスタマイズを容易なものとしています。

9. 2. 1　割り込みマスクの論理和

割り込みマスクの論理和は，該当ユーザ・オウン関数のパラメータで指定された割り込みマスク・パターンと対象デバイスの割り込みマスク・パターン（割り込みコントローラ制御レジスタ EIC*n*，または割り込みマスク・レジスタ IMR*m* の割り込みマスク・フラグ EIMK*n* の値）の論理和をとり，その結果を対象レジスタの割り込みマスク・フラグ EIMK*n* に設定するために切り出された割り込みマスク・パターン設定処理専用ルーチンであり，処理プログラムからサービス・コール loc_cpu，iloc_cpu が発行された際に呼び出されます。

- 基本型

割り込みマスクの論理和を記述する場合，VP 型の引数を 1 つ持った void 型の関数（関数名：_kernel_usr_msk_intmsk）として記述します。

なお，引数 *p_intms* には，“設定する割り込みマスク・パターンを格納した領域へのポインタ”が設定されます。

以下に，割り込みマスクの論理和を C 言語で記述する場合の基本型を示します。

```
#include    <kernel.h>
void
_kernel_usr_msk_intmsk(VP p_intms){
        ..........................
        ..........................
        return;
}
```

9. 2. 2　割り込みマスクの参照

割り込みマスクの参照は，該当ユーザ・オウン関数のパラメータで指定された領域に対象デバイスの割り込みマスク・パターン（割り込みコントローラ制御レジスタ EIC*n*，または割り込みマスク・レジスタ IMR*m* の割り込みマスク・フラグ EIMK*n* の値）を設定するために切り出された割り込みマスク・パターン獲得処理専用ルーチンであり，処理プログラムからサービス・コール loc_cpu，iloc_cpu が発行された際に呼び出されます。

- 基本型

割り込みマスクの参照を記述する場合，VP 型の引数を 1 つ持った void 型の関数（関数名：_kernel_usr_get_intmsk）として記述します。

なお，引数 *p_intms* には，“獲得した割り込みマスク・パターンを設定する領域へのポインタ”が設定されます。

以下に，割り込みマスクの参照を C 言語で記述する場合の基本型を示します。

```
#include    <kernel.h>
void
_kernel_usr_get_intmsk(VP p_intms){
        ..........................
        ..........................
        return;
}
```

9. 2. 3　割り込みマスクの上書き

割り込みマスクの上書きは，該当ユーザ・オウン関数のパラメータで指定された割り込みマスク・パターンを対象デバイスの割り込みマスク・パターン（割り込みコントローラ制御レジスタ EIC*n*，または割り込みマスク・レジスタ IMR*m* の割り込みマスク・フラグ EIMK*n* の値）として設定するために切り出された割り込みマスク・パターン設定処理専用ルーチンであり，処理プログラムからサービス・コール unl_cpu，iunl_cpu が発行された際に呼び出されます。

- 基本型

割り込みマスクの上書きを記述する場合，VP 型の引数を 1 つ持った void 型の関数（関数名：_kernel_usr_set_intmsk）として記述します。

なお，引数 *p_intms* には，“設定する割り込みマスク・パターンを格納した領域へのポインタ”が設定されます。

以下に，割り込みマスクの上書きを C 言語で記述する場合の基本型を示します。

```
#include    <kernel.h>
void
_kernel_usr_set_intmsk(VP p_intms){
        ..........................
        ..........................
        return;
}
```

9. 2. 4　マスカブル割り込みの受け付け禁止

マスカブル割り込みの受け付け禁止は，マスカブル割り込みの受け付けを許可状態から禁止状態へと変更するために切り出されたマスカブル割り込みの受け付け操作処理専用ルーチンであり，処理プログラムからサービス・コール dis_int が発行された際に呼び出されます。

- 基本型

マスカブル割り込みの受け付け禁止を記述する場合，INTNO 型の引数を 1 つ持った void 型の関数（関数名：_kernel_usr_dis_int）として記述します。

なお，引数 *intno* には，“許可状態から禁止状態へと変更するマスカブル割り込みに対応した例外要因コード”が設定されます。

以下に，マスカブル割り込みの受け付け禁止を C 言語で記述する場合の基本型を示します。

```
#include    <kernel.h>
void
_kernel_usr_set_dis_int(INTNO intno){
        ..........................
        ..........................
        return;
}
```

9. 2. 5　マスカブル割り込みの受け付け許可

マスカブル割り込みの受け付け許可は，マスカブル割り込みの受け付けを禁止状態から許可状態へと変更するために切り出されたマスカブル割り込みの受け付け操作処理専用ルーチンであり，処理プログラムからサービス・コール ena_int が発行された際に呼び出されます。

- 基本型

マスカブル割り込みの受け付け許可を記述する場合，INTNO 型の引数を 1 つ持った void 型の関数（関数名：_kernel_usr_ena_int）として記述します。

なお，引数 *intno* には，“禁止状態から許可状態へと変更するマスカブル割り込みに対応した例外要因コード”が設定されます。

以下に，マスカブル割り込みの受け付け許可を C 言語で記述する場合の基本型を示します。

```
#include    <kernel.h>
void
_kernel_usr_set_ena_int(INTNO intno){
        ..........................
        ..........................
        return;
}
```

9. 2. 6　割り込みエントリ

割り込みエントリは，割り込みが発生した際に対象デバイスが強制的に制御を移すハンドラ・アドレスに対して割り込み前処理への分岐処理を割り付けるために切り出されたエントリ処理専用ルーチンです。

ただし，割り込みハンドラ情報で定義された割り込みハンドラに対応した例外要因コードに関する割り込みエントリはコンフィギュレータを実行することにより出力されるエントリ・ファイルに内包されています。したがって，割り込みエントリをカスタマイズする必要がない場合には，該当エントリ・ファイルを利用することにより，割り込みエントリの記述が不要となります。

- 基本型

割り込みエントリを記述する場合，割り込みが発生した際に対象デバイスが強制的に制御を移すハンドラ・アドレスに対して割り込み前処理への分岐処理を割り付けます。

以下に，割り込みエントリをアセンブリ言語で記述する場合の基本型を示します。

【CX 対応版】

```
inhno       .cseg   text        -- inhno：例外要因名
jr          __kernel_int_entry  -- 割り込み前処理への分岐
```

【CCV850E 対応版】

```
.org        inthdr              -- inthdr：起動アドレス
jr          __kernel_int_entry  -- 割り込み前処理への分岐
```

9. 3　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呼び出される割り込み処理専用ルーチンです。

なお，RI850MP では，割り込みハンドラを“タスクとは独立したもの（非タスク）”として位置づけています。このため，割り込みが発生した際には，システム内で最高優先度を持つタスクが処理を実行中であっても，その処理は中断され，割り込みハンドラに制御が移ります。

9. 3. 1　割り込みハンドラの基本型

割り込みハンドラを記述する場合，引数を持たない void 型の関数として記述します。

以下に，割り込みハンドラを C 言語で記述する場合の基本型を示します。

```
#include    <kernel.h>
void
inthdr(void){
        ……
        ……
        return;
}
```

9. 3. 2　割り込みハンドラの登録

RI850MP では，割り込みハンドラの静的な登録のみをサポートしています。したがって，処理プログラムからサービス・コールを発行して動的に登録することはできません。

なお，割り込みハンドラの静的な登録とは，コンフィギュレーション・ファイルで静的 API“DEF_INH”を使用して割り込みハンドラに関する情報を定義することをいいます。

備考 1. 静的 API“DEF_INH”についての詳細は，「B. 5. 9　割り込みハンドラ情報」を参照してください。

2. RI850MP では，一定周期で発生するタイマ割り込みを利用することにより，時間管理機能を実現しています。このため，タイマ割り込み情報で定義した例外要因コードに対して割り込みハンドラを登録することは禁止されています。

第 10 章　システム構成管理機能

本章では，RI850MP が提供するシステム構成管理機能について解説しています。

10. 1　概　　要

RI850MP におけるシステム構成管理機能では，CPU 例外が発生した際に起動する CPU 例外ハンドラ，およびシステム初期化処理から呼び出される初期化ルーチンに関連した機能を提供しています。

10. 2　ユーザ・オウン・コーディング部

RI850MP では，システム構成管理機能のうち，ユーザの実行環境に依存した処理をユーザ・オウン・コーディング部として切り出し，サンプル・ソース・ファイルを提供しています。

これにより，様々な実行環境への移行性を向上させるとともに，カスタマイズを容易なものとしています。

10. 2. 1　CPU 例外エントリ

CPU 例外エントリは，CPU 例外が発生した際に対象デバイスが強制的に制御を移すハンドラ・アドレスに対して割り込み前処理への分岐処理を割り付けるために切り出されたエントリ処理専用ルーチンです。

ただし，CPU 例外ハンドラ情報で定義された CPU 例外ハンドラに対応した例外要因コードに関する CPU 例外エントリはコンフィギュレータを実行することにより出力されるエントリ・ファイルに内包されています。したがって，CPU 例外エントリをカスタマイズする必要がない場合には，該当エントリ・ファイルを利用することにより，CPU 例外エントリの記述が不要となります。

- 基本型

CPU 例外エントリを記述する場合，CPU 例外が発生した際に対象デバイスが強制的に制御を移すハンドラ・アドレスに対して割り込み前処理への分岐処理を割り付けます。

以下に，CPU 例外エントリをアセンブリ言語で記述する場合の基本型を示します。

【CX 対応版】

```
excno       .cseg   text          -- excno：例外要因名
jr          __kernel_int_entry    -- 割り込み前処理への分岐
```

【CCV850E 対応版】

```
.org        exchdr                -- exchdr：起動アドレス
jr          __kernel_int_entry    -- 割り込み前処理への分岐
```

10. 3　CPU 例外ハンドラ

CPU 例外ハンドラは，CPU 例外が発生した際に呼び出される CPU 例外処理専用ルーチンです。

なお，RI850MP では，CPU 例外ハンドラを“タスクとは独立したもの（非タスク）”として位置づけています。このため，CPU 例外が発生した際には，システム内で最高優先度を持つタスクが処理を実行中であっても，その処理は中断され，CPU 例外ハンドラに制御が移ります。

10. 3. 1　CPU 例外ハンドラの基本型

CPU 例外ハンドラを記述する場合，引数を持たない void 型の関数として記述します。

以下に，CPU 例外ハンドラを C 言語で記述する場合の基本型を示します。

```
#include    <kernel.h>
void
exchdr(void){
        ……
        ……
        return;
}
```

10. 3. 2　CPU 例外ハンドラの登録

RI850MP では，CPU 例外ハンドラの静的な登録のみをサポートしています。したがって，処理プログラムからサービス・コールを発行して動的に登録することはできません。

なお，CPU 例外ハンドラの静的な登録とは，コンフィギュレーション・ファイルで静的 API“DEF_EXC”を使用して CPU 例外ハンドラに関する情報を定義することをいいます。

備考　静的 API“DEF_EXC”についての詳細は，「B. 5. 10　CPU 例外ハンドラ情報」を参照してください。

10. 4　初期化ルーチン

初期化ルーチンは，ユーザの実行環境に依存したハードウエア，およびソフトウエアを初期化するために切り出された初期化処理専用ルーチンであり，システム初期化処理から呼び出されます。

10. 4. 1　初期化ルーチンの基本型

初期化ルーチンを記述する場合，VP_INT 型の引数を 1 つ持った void 型の関数として記述します。

なお，引数 *exinf* には，“初期化ルーチン情報で定義した拡張情報”が設定されます。

以下に，初期化ルーチンを C 言語で記述する場合の基本型を示します。

```
#include    <kernel.h>
void
inirtn(VP_INT exinf){
        ..........................
        ..........................
        return;
}
```

10. 4. 2　初期化ルーチンの登録

RI850MP では，初期化ルーチンの静的な登録のみをサポートしています。したがって，処理プログラムからサービス・コールを発行して動的に登録することはできません。

なお，初期化ルーチンの静的な登録とは，コンフィギュレーション・ファイルで静的 API“ATT_INI”を使用して初期化ルーチンに関する情報を定義することをいいます。

備考　静的 API“ATT_INI”についての詳細は，「B. 5. 11　初期化ルーチン情報」を参照してください。

第 11 章　スケジューリング機能

本章では，RI850MP が提供するスケジューリング機能について解説しています。

11. 1　概　　要

RI850MP におけるスケジューリング機能では，動的に変化していくタスクの状態を直接参照することにより，タスクの実行順序を管理／決定し，最適なタスクに対象デバイスの使用権を割り当てる機能を提供しています。

11. 2　駆動方式

RI850MP では，スケジューラの駆動方式として何らかの事象（きっかけ）が発生した際に起動する“イベント・ドリブン方式”を採用しています。

以下に，RI850MP がスケジューラを起動する際のきっかけを示します。

- タスクの状態遷移を引き起こす可能性があるサービス・コールの発行
- 非タスク（周期ハンドラ，割り込みハンドラなど）からの復帰命令の発行
- 時間管理を実現する際に利用しているタイマ割り込みの発生

11. 3　スケジューリング方式

RI850MP では，タスクのスケジューリング方式として各タスクに付与されている優先度（現在優先度）を利用した“優先度方式”，および RI850MP のスケジューリング対象となってからの経過時間を利用した“FCFS 方式”の 2 種類を採用しています。

- 優先度方式

  実行可能な状態（実行状態，または実行可能状態）にあるタスクの中から“最も高い優先度（現在優先度）が付与されているタスク”を選び出し，対象デバイスの使用権を割り当てます。

- FCFS 方式

  実行可能状態へと遷移してから“最も時間が経過しているタスク”を選び出し，対象デバイスの使用権を割り当てます。

  なお，FCFS 方式によるタスクのスケジューリングは，優先度方式における“タスクの選び出し基準”である最も高い優先度（現在優先度）のタスクが複数存在する場合に実行されます。

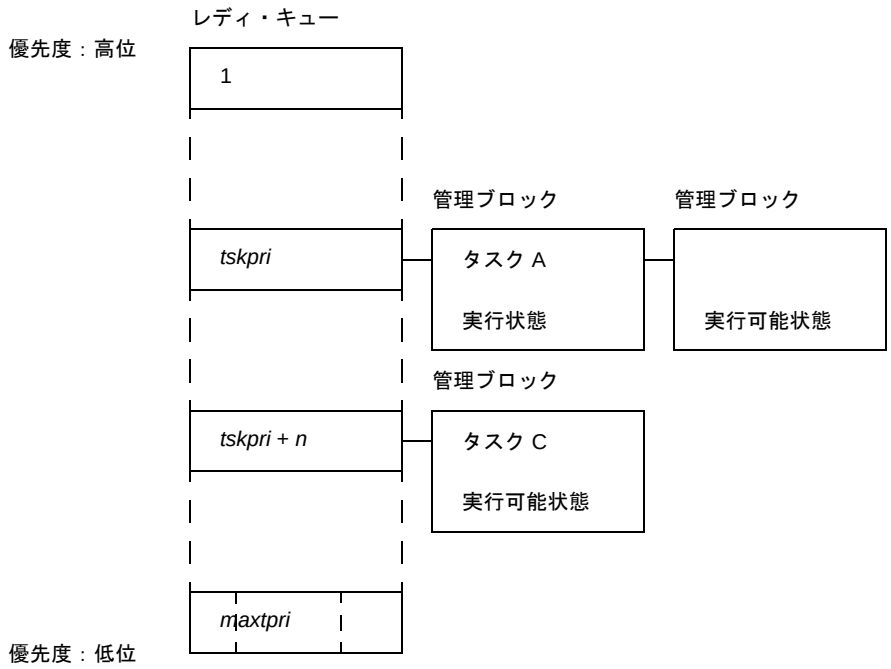
11. 4　レディ・キュー

RI850MP では，タスクのスケジューリング方式を実現する手段として“レディ・キュー”を採用しています。

なお，RI850MP におけるレディ・キューは，優先度をキーとしたハッシュ・テーブルであり，実行可能な状態（実行状態，または実行可能状態）へと遷移したタスクの管理ブロックは，該当優先度位置に FIFO 順でキューイングされます。

したがって，RI850MP のスケジューラは，レディ・キューの優先度高位からタスクの検出処理を実行し，レディ・キューにキューイングされているタスクを検出した際には，該当優先度の先頭にキューイングされているタスクに対象デバイスの使用権を割り当てます。

図 11—1　レディ・キュー



11. 4. 1　レディ・キューの生成

RI850MP では，レディ・キューの静的な生成のみをサポートしています。したがって，処理プログラムからサービス・コールを発行して動的に生成することはできません。

なお，レディ・キューの静的な生成とは，コンフィギュレーション・ファイルで静的 API“MAX_PRI”を使用して優先度に関する情報を定義することをいいます。

備考　静的 API“MAX_PRI”についての詳細は，「B. 3. 5　最大優先度情報」を参照してください。

11. 5　スケジューリングのロック

RI850MP では，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（loc_cpu，iloc_cpu，unl_cpu，iunl_cpu，dis_dsp，ena_dsp）を利用することにより，処理プログラムからスケジューラの状態を明示的に操作し，ディスパッチ処理を禁止／許可する機能を提供しています。

11. 6　アイドル・ルーチン

アイドル・ルーチンは，対象デバイスが提供しているスタンバイ機能（低消費電力システムの実現）を有効活用するために切り出されたアイドル処理専用ルーチンであり，RI850MP のスケジューリング対象となるタスクがシステム内に 1 つも存在しなくなった際にスケジューラから呼び出されます。

11. 6. 1　初期化ルーチンの基本型

アイドル・ルーチンを記述する場合，引数を持たない void 型の関数として記述します。

以下に，アイドル・ルーチンを C 言語で記述する場合の基本型を示します。

```
#include    <kernel.h>
void
idlrtn(void){
        ..........................
        ..........................
        return;
}
```

11. 6. 2　アイドル・ルーチンの登録

RI850MP では，アイドル・ルーチンの静的な登録のみをサポートしています。したがって，処理プログラムからサービス・コールを発行して動的に登録することはできません。

なお，アイドル・ルーチンの静的な登録とは，コンフィギュレーション・ファイルで静的 API“VATT_IDL”を使用してアイドル・ルーチンに関する情報を定義することをいいます。

備考　静的 API“VATT_IDL”についての詳細は，「B. 5. 12　アイドル・ルーチン情報」を参照してください。

第 12 章　システム初期化処理

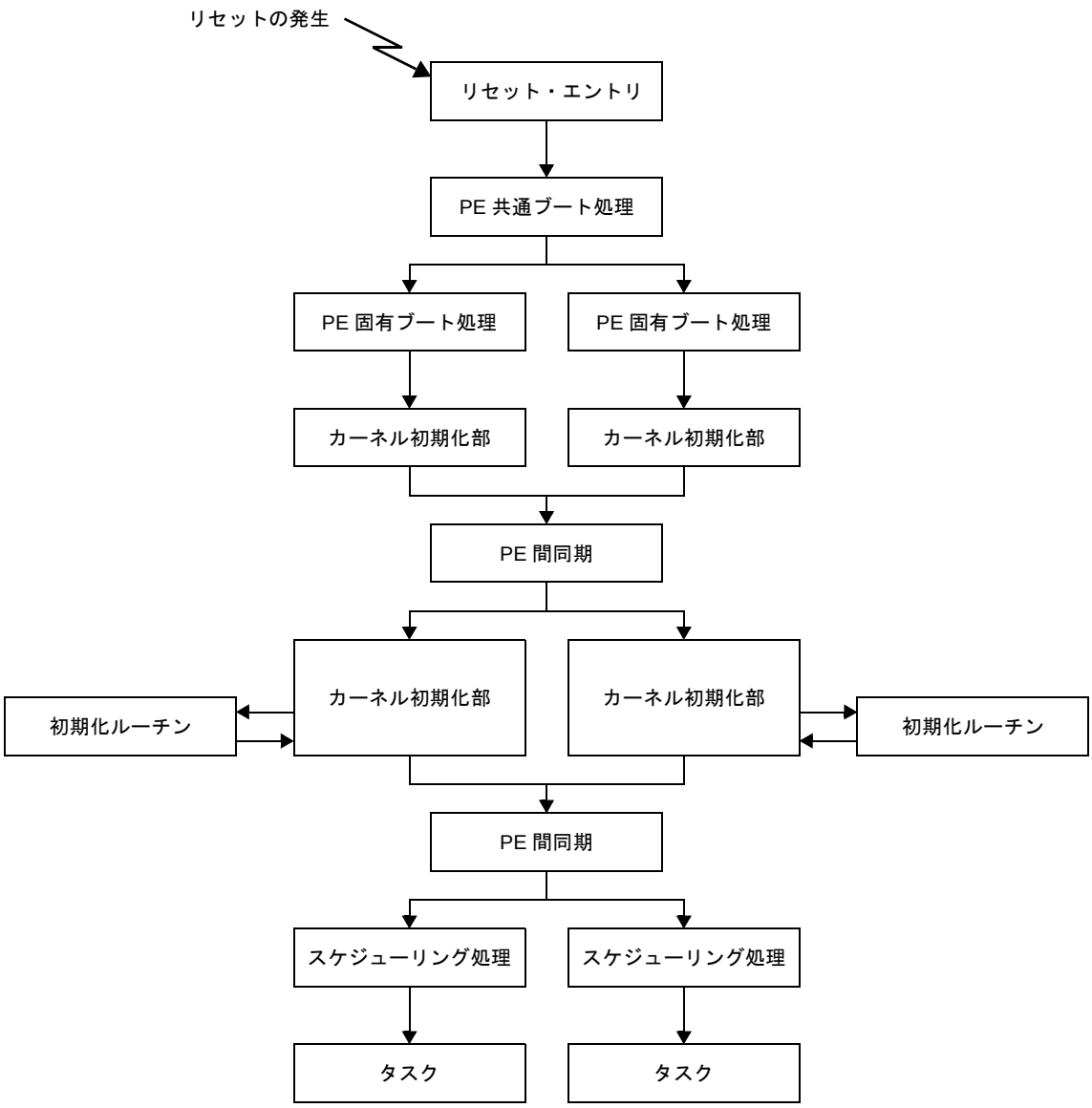
本章では，RI850MP が提供するシステム初期化処理について解説しています。

12. 1　概　　要

RI850MP におけるシステム初期化処理では，リセットの発生から処理プログラム（タスク）に制御を移すまでに必要となる“RI850MP が処理を実行するうえで必要となるハードウエア，およびソフトウエアの初期化処理”を意味しています。

以下に，リセットの発生から処理プログラムに制御を移すまでの処理の流れを示します。

図 12—1　システム初期化処理の流れ



12. 2　ユーザ・オウン・コーディング部

RI850MP では，システム初期化処理のうち，ユーザの実行環境に依存した処理をユーザ・オウン・コーディング部として切り出し，サンプル・ソース・ファイルを提供しています。

これにより，様々な実行環境への移行性を向上させるとともに，カスタマイズを容易なものとしています。

12. 2. 1　リセット・エントリ

リセット・エントリは，リセットが発生した際に対象デバイスが強制的に制御を移すハンドラ・アドレスに対して PE 共通ブート処理への分岐処理を割り付けるために切り出されたエントリ処理専用ルーチンです。

- 基本型

リセット・エントリを記述する場合，リセットが発生した際に対象デバイスが強制的に制御を移すハンドラ・アドレスに対して PE 共通ブート処理への分岐処理を割り付けます。

ただし，PE 共通ブート処理を CPU 例外ハンドラ情報で CPU 例外ハンドラとして定義した際には，リセット・エントリはコンフィギュレータを実行することにより出力されるエントリ・ファイルに内包されています。したがって，リセット・エントリをカスタマイズする必要がない場合には，該当エントリ・ファイルを利用することにより，リセット・エントリの記述が不要となります。

以下に，リセット・エントリをアセンブリ言語で記述する場合の基本型を示します。

【CX 対応版】

```
RESET       .cseg   text
jr          __boot              -- PE 共通ブート処理への分岐
```

【CCV850E 対応版】

```
.org        0x00000000
jr          __boot              -- PE 共通ブート処理への分岐
```

12. 2. 2　ブート処理

ブート処理は，RI850MP が処理を実行するうえで必要となる最低限のハードウエアを初期化するために切り出された初期化処理専用ルーチンであり，PE 共通ブート処理，および PE 固有ブート処理から構成されています。

- PE 共通ブート処理

PE 共通ブート処理は，PE 共通で必要となるハードウエアを初期化するために切り出された初期化処理専用ルーチンであり，リセット・エントリから呼び出されます。

以下に，PE 共通ブート処理をアセンブリ言語で記述する場合の基本型を示します。

```
        .text
        .align      0x4
        .global     __boot
__boot:
        mov         __boot_PE1, r10
        mov         __boot_PE2, r20
        mov         PEID, r1
        ld.h        0[r1], r2
        cmp         1, r2
        cmove       r10, r20, r21
        jmp         [r21]               -- PE 固有ブート処理処理への分岐
```

- PE 固有ブート処理

PE 固有ブート処理は，PE 単位に必要となるハードウエア，およびソフトウエアを初期化するために切り出された初期化処理専用ルーチンであり，PE 共通ブート処理から呼び出されます。

以下に，PE 固有ブート処理をアセンブリ言語で記述する場合の基本型を示します。

【CX 対応版】

```
        .text
        .align      0x4
        .global     __boot_PEn
__boot_PEn:
        .extern     __kernel_sit
        ……
        ……
        mov         #__kernel_sit, r6
        jarl        __kernel_start, lp  -- カーネル初期化部への分岐
```

【CCV850E 対応版】

```
        .text
        .align      0x4
        .global     __boot_PEn
__boot_PEn:
        .extern     __kernel_sit
        ……
        ……
        mov         __kernel_sit, r6
        jarl        __kernel_start, lp  -- カーネル初期化部への分岐
```

12. 3　カーネル初期化部

カーネル初期化部は，RI850MP が処理を実行するうえで必要となる最低限のハードウエアを初期化するために切り出された初期化処理専用ルーチンであり，PE 固有ブート処理から呼び出されます。

なお，カーネル初期化部では，以下に示した処理が行われます。

- 浮動小数点設定／状態レジスタの初期化
- システム時刻の初期化
- タスクの生成／起動
- セマフォの生成
- イベントフラグの生成
- データ・キューの生成
- メールボックスの生成
- ミューテックスの生成
- 固定長メモリ・プールの生成
- 周期ハンドラの登録／起動
- 割り込みハンドラの登録
- CPU 例外ハンドラの登録
- 初期化ルーチンの登録
- アイドル・ルーチンの登録
- 初期化ルーチンの呼び出し
- スケジューラの起動

備考　カーネル初期化部は，RI850MP が提供しているシステム初期化処理に内包されています。したがって，ユーザがカーネル初期化部を記述する必要はありません。

12. 4　初期化ルーチン

初期化ルーチンは，ユーザの実行環境に依存したハードウエア，およびソフトウエアを初期化するために切り出された初期化処理専用ルーチンであり，カーネル初期化部から呼び出されます。

備考　初期化ルーチンについての詳細は，「10. 4　初期化ルーチン」を参照してください。

第 13 章　サービス・コール

本章では，RI850MP が提供するサービス・コールについて解説しています。

13. 1　概　　要

RI850MP が提供するサービス・コールは，ユーザが記述した処理プログラムから RI850MP が管理している資源（タスク，セマフォ，データ・キューなど）を操作するために用意されたものです。

以下に，RI850MP が提供しているサービス・コールを機能別に示します。

- タスク管理機能

act_tsk，iact_tsk，can_act，ican_act，ext_tsk，ter_tsk，chg_pri，ichg_pri，get_pri，iget_pri，ref_tsk，iref_tsk

- タスク付属同期機能

slp_tsk，tslp_tsk，wup_tsk，iwup_tsk，can_wup，ican_wup，rel_wai，irel_wai，sus_tsk，isus_tsk，rsm_tsk，irsm_tsk，frsm_tsk，ifrsm_tsk，dly_tsk

- 同期・通信機能（セマフォ）

sig_sem，isig_sem，wai_sem，pol_sem，ipol_sem，twai_sem，ref_sem，iref_sem

- 同期・通信機能（イベントフラグ）

set_flg，iset_flg，clr_flg，iclr_flg，wai_flg，pol_flg，ipol_flg，twai_flg，ref_flg，iref_flg

- 同期・通信機能（データ・キュー）

snd_dtq，psnd_dtq，ipsnd_dtq，tsnd_dtq，fsnd_dtq，ifsnd_dtq，rcv_dtq，prcv_dtq，iprcv_dtq，trcv_dtq，ref_dtq，iref_dtq

- 同期・通信機能（メールボックス）

snd_mbx，isnd_mbx，rcv_mbx，prcv_mbx，iprcv_mbx，trcv_mbx，ref_mbx，iref_mbx

- 拡張同期・通信機能

loc_mtx，ploc_mtx，tloc_mtx，unl_mtx，ref_mtx，iref_mtx

- メモリ・プール管理機能

get_mpf，pget_mpf，ipget_mpf，tget_mpf，rel_mpf，irel_mpf，ref_mpf，iref_mpf

- 時間管理機能

set_tim，iset_tim，get_tim，iget_tim，sta_cyc，ista_cyc，stp_cyc，istp_cyc，ref_cyc，iref_cyc

- システム状態管理機能

rot_rdq，irot_rdq，get_tid，iget_tid，loc_cpu，iloc_cpu，unl_cpu，iunl_cpu，dis_dsp，ena_dsp，sns_ctx，sns_loc，sns_dsp，sns_dpn

- 割り込み管理機能

dis_int，ena_int，chg_ipm，ichg_ipm，get_ipm，iget_ipm

13. 1. 1　サービス・コールの呼び出し

サービス・コールを C 言語，およびアセンブリ言語で記述された処理プログラムから発行する場合の呼び出し方法を以下に示します。

- C 言語で記述された処理プログラム

通常の C 言語関数と同様の方法で呼び出しを行うことにより，サービス・コールの引数は RI850MP に渡り，該当処理が実行されます。

- アセンブリ言語で記述された処理プログラム

使用する C コンパイラ・パッケージの関数呼び出し規約にしたがった引数，および戻り番地の設定を行ったのち，jarl 命令による呼び出しを行うことにより，サービス・コールの引数は RI850MP に渡り，該当処理が実行されます。

備考　RI850MP が提供するサービス・コールを処理プログラムから発行する場合，以下に示したヘッダ・ファイルの定義（インクルード処理）が必要となります。

kernel.h：　標準ヘッダ・ファイル
kernel_id.h：　システム情報ヘッダ・ファイル

13. 2　データ・マクロ

RI850MP が提供しているサービス・コールを発行する際に使用するデータ・マクロについて以下に示します。

13. 2. 1　データ・タイプ

データ・タイプは，標準ヘッダ・ファイル“kernel.h”から呼び出されるヘッダ・ファイル“types.h”に定義されています。

表 13—1　データ・タイプ

マクロ	型	説明
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	signed char	データ・タイプが定まらない 8 ビット値
VH	signed short	データ・タイプが定まらない 16 ビット値
VW	signed long	データ・タイプが定まらない 32 ビット値
VP	void *	データ・タイプが定まらないものへのポインタ
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	signed int	真偽値（TRUE，または FALSE）
FN	signed short	サービス・コールの機能コード
ER	signed long	サービス・コールの戻り値
ID	signed short	オブジェクトの ID
ATR	unsigned short	オブジェクトの属性
STAT	unsigned short	オブジェクトの状態
MODE	unsigned short	サービス・コールの動作モード
PRI	signed short	オブジェクトの優先度
SIZE	unsigned long	メモリ領域のサイズ
TMO	signed long	待ち時間
RELTIM	unsigned long	相対時間
SYSTIM	－	SYSTIM についての詳細は，「13. 3. 11　システム時刻 SYSTIM」を参照
VP_INT	signed int	データ・タイプが定まらないものへのポインタ，または符号付き 32 ビット整数

マクロ	型	説明
ER_BOOL	signed long	サービス・コールの戻り値，または真偽値（TRUE，またはFALSE）
ER_ID	signed long	サービス・コールの戻り値，またはオブジェクトの ID
ER_UINT	unsigned int	サービス・コールの戻り値，または符号なし 32 ビット整数
FLGPTN	unsigned int	ビット・パターン
T_MSG	－	T_MSG についての詳細は，「13. 3. 9　メッセージ（優先度なし）T_MSG」を参照
T_MSG_PRI	－	T_MSG_PRI についての詳細は，「13. 3. 10　メッセージ（優先度あり）T_MSG_PRI」を参照
INTNO	unsigned short	例外要因コード
INTPMR	unsigned short	レジスタ値
PE_ID	unsigned char	PE 番号

13. 2. 2　戻り値

戻り値は，標準ヘッダ・ファイル“kernel.h”から呼び出されるヘッダ・ファイル“errcd.h”，“option.h”に定義されています。

表 13―2　戻り値

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能
E_PAR	-17	引数が不正
E_ID	-18	ID が不正
E_CTX	-25	コンテキスト・エラー
E_ILUSE	-28	サービス・コールの使用方法が不正
E_OBJ	-41	オブジェクト状態エラー
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
TRUE	1	真
FALSE	0	偽
NULL	0	無効ポインタ

13. 2. 3　オブジェクトの属性

オブジェクトの属性は，標準ヘッダ・ファイル“kernel.h”から呼び出されるヘッダ・ファイル“option.h”に定義されています。

表 13—3　オブジェクトの属性

マクロ	数値	説明
TA_HLNG	0x0	C 言語
TA_ASM	0x1	アセンブリ言語
TA_TFIFO	0x0	タスクを FIFO 順にキューイング
TA_TPRI	0x1	タスクを優先度順にキューイング
TA_MFIFO	0x0	メッセージを FIFO 順にキューイング
TA_MPRI	0x2	メッセージを優先度順にキューイング
TA_ACT	0x2	実行可能状態
TA_WSGL	0x0	1 個のタスク
TA_WMUL	0x2	複数のタスク
TA_CLR	0x4	要求条件を満足した際，ビット・パターンをクリア
TA_STA	0x2	動作状態
TA_PHS	0x4	起動位相を保存
TA_ENAINT	0x0	割り込みを許可
TA_DISINT	0x8000	割り込みを禁止

13. 2. 4　タスクの待ち時間

タスクの待ち時間は，標準ヘッダ・ファイル“kernel.h”から呼び出されるヘッダ・ファイル“option.h”に定義されています。

表 13—4　タスクの待ち時間

マクロ	数値	説明
TMO_POL	0	ポーリング
TMO_FEVR	-1	永久待ち

13. 2. 5　タスクの要求条件

タスクの要求条件は，標準ヘッダ・ファイル“kernel.h”から呼び出されるヘッダ・ファイル“option.h”に定義されています。

表 13―5　タスクの要求条件

マクロ	数値	説明
TWF_ANDW	0x0	AND 待ち
TWF_ORW	0x1	OR 待ち

13. 2. 6　タスクの現在状態

タスクの現在状態は，標準ヘッダ・ファイル“kernel.h”から呼び出されるヘッダ・ファイル“option.h”に定義されています。

表 13―6　タスクの現在状態

マクロ	数値	説明
TTS_RUN	0x1	実行状態
TTS_RDY	0x2	実行可能状態
TTS_WAI	0x4	待ち状態
TTS_SUS	0x8	強制待ち状態
TTS_WAS	0xC	二重待ち状態
TTS_DMT	0x10	休止状態

13. 2. 7　タスクの待ち要因

タスクの待ち要因（待ち状態の種類）は，標準ヘッダ・ファイル“kernel.h”から呼び出されるヘッダ・ファイル“option.h”に定義されています。

表 13—7　タスクの待ち要因

マクロ	数値	説明
TTW_NONE	0x0	待ち状態でない
TTW_SLP	0x1	起床待ち状態
TTW_DLY	0x2	時間経過待ち状態
TTW_SEM	0x4	資源獲得待ち状態
TTW_FLG	0x8	イベントフラグ待ち状態
TTW_SDTQ	0x10	データ送信待ち状態
TTW_RDTQ	0x20	データ受信待ち状態
TTW_MBX	0x40	メッセージ受信待ち状態
TTW_MTX	0x80	ミューテックス獲得待ち状態
TTW_MPF	0x2000	固定長メモリ・ブロック獲得待ち状態

13. 2. 8　周期ハンドラの現在状態

周期ハンドラの現在状態は，標準ヘッダ・ファイル“kernel.h”から呼び出されるヘッダ・ファイル“option.h”に定義されています。

表 13—8　周期ハンドラの現在状態

マクロ	数値	説明
TCYC_STP	0x0	停止状態
TCYC_STA	0x1	動作状態

13. 2. 9　その他の定数

その他の定数は，標準ヘッダ・ファイル“kernel.h”から呼び出されるヘッダ・ファイル“option.h”に定義されています。

表 13—9　その他の定数

マクロ	数値	説明
TSK_SELF	0	自タスク
TSK_NONE	0	タスクはキューイングされていない
TPRI_SELF	0	タスクの現在優先度
TPRI_INI	0	タスクの初期優先度

13. 2. 10　条件コンパイル用マクロ

RI850MP では，以下の条件コンパイル用マクロが用意されています。

表 13—10　条件コンパイル用マクロ

マクロ	説明
__cx__	C コンパイラ・パッケージに CX を使用
__ccv850e__	C コンパイラ・パッケージに CCV850E を使用
__v850e2m__	ターゲット・デバイスに V850E2M を使用

13. 3　データ構造体

RI850MP が提供しているサービス・コールを処理プログラムから発行する際に使用するデータ構造体について以下に示します。

13. 3. 1　タスク情報 T_RTSK

タスク情報は，標準ヘッダ・ファイル“kernel.h”から呼び出されるヘッダ・ファイル“packet.h”に定義されています。

```
typedef struct  t_rtsk {
        STAT    tskstat;    /* 現在状態 */
        PRI     tskpri;     /* 現在優先度 */
        PRI     tskbpri;    /* システム予約領域 */
        STAT    tskwait;    /* 待ち要因 */
        ID      wobjid;     /* オブジェクトの ID*/
        TMO     lefttmo;    /* 残り時間 */
        UINT    actcnt;     /* 起動要求ネスト数 */
        UINT    wupcnt;     /* 起床要求ネスト数 */
        UINT    suscnt;     /* 強制待ち要求ネスト数 */
        ATR     tskatr;     /* 属性 */
        PRI     itskpri;    /* 初期優先度 */
        PE_ID   peid;       /*PE 番号 */
} T_RTSK;
```

以下に，タスク情報の詳細を示します。

- tskstat

タスクの現在状態が格納されます。

マクロ	数値	説明
TTS_RUN	0x1	実行状態
TTS_RDY	0x2	実行可能状態
TTS_WAI	0x4	待ち状態
TTS_SUS	0x8	強制待ち状態
TTS_WAS	0xC	二重待ち状態
TTS_DMT	0x10	休止状態

- tskpri

タスクの現在優先度が格納されます。

なお，該当タスクが休止状態の場合には，初期優先度が格納されます。

- tskbpri

システム予約領域です。

- tskwait

タスクの待ち要因（待ち状態の種類）が格納されます。
なお，該当タスクが待ち状態，または二重待ち状態以外の場合には，TTW_NONE（= 0x0）が格納されます。

マクロ	数値	説明
TTW_NONE	0x0	待ち状態でない
TTW_SLP	0x1	起床待ち状態
TTW_DLY	0x2	時間経過待ち状態
TTW_SEM	0x4	資源獲得待ち状態
TTW_FLG	0x8	イベントフラグ待ち状態
TTW_SDTQ	0x10	データ送信待ち状態
TTW_RDTQ	0x20	データ受信待ち状態
TTW_MBX	0x40	メッセージ受信待ち状態
TTW_MTX	0x80	ミューテックス獲得待ち状態
TTW_MPF	0x2000	固定長メモリ・ブロック獲得待ち状態

備考　tslp_tsk，twai_sem，twai_flg などの発行に伴い，時限付きの待ち状態へと遷移しているタスクについては，該当待ち状態を示す値“TTW_SLP，TTW_SEM，TTW_FLG など”と時間経過待ち状態を示す値“TTW_DLY”との論理和が格納されます。

- wobjid

タスクが待ち状態へと遷移するきっかけとなったオブジェクト（セマフォ，イベントフラグ，データ・キューなど）の ID が格納されます。
なお，該当タスクが資源獲得待ち状態，イベントフラグ待ち状態，データ送信待ち状態，データ受信待ち状態，メッセージ受信待ち状態，ミューテックス獲得待ち状態，固定長メモリ・ブロック獲得待ち状態以外の場合には，0 が格納されます。

- lefttmo

タスクが時間経過待ち状態（tslp_tsk，dly_tsk，twai_sem などの発行に伴い遷移）を解除されるまでの残り時間（単位：ミリ秒）が格納されます。
なお，該当タスクが時間経過待ち状態以外の場合には，0 が格納されます。

備考　該当タスクが永久待ちの場合には，TMO_FEVR（= -1）が格納されます。

- actcnt

タスクに保持されている起動要求ネスト数（act_tsk ／ iact_tsk のネスト数）が格納されます。
なお，該当タスクが休止状態の場合には，0 が格納されます。

- wupcnt

タスクに保持されている起床要求ネスト数（wup_tsk ／ iwup_tsk のネスト数）が格納されます。

なお，該当タスクが休止状態の場合には，0 が格納されます。

- suscnt

タスクに保持されている強制待ち要求ネスト数（sus_tsk ／ isus_tsk のネスト数）が格納されます。

なお，該当タスクが休止状態の場合には，0 が格納されます。

- tskatr

タスクの属性（記述言語，初期状態，初期割り込み状態）が格納されます。

- タスクの記述言語（ビット 0）

TA_HLNG： C 言語

TA_ASM： アセンブリ言語

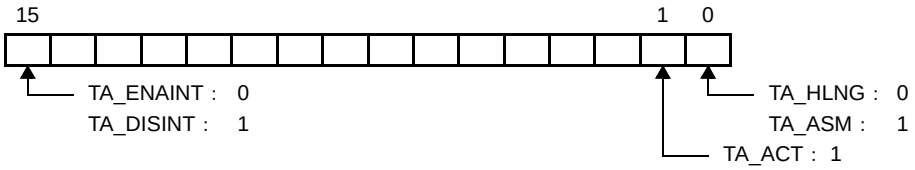
- タスクの初期状態（ビット 1）

TA_ACT： 実行可能状態

- タスクの初期割り込み状態（ビット 15）

TA_ENAINT： 割り込みを許可

TA_DISINT： 割り込みを禁止



備考 タスクの初期状態が休止状態の場合には，ビット 1 に 0 が格納されます。

- itskpri

タスクの初期優先度が格納されます。

- peid

タスクが所属しているプロセッサ・エレメントの PE 番号が格納されます。

13. 3. 2 セマフォ情報 T_RSEM

セマフォ情報は，標準ヘッダ・ファイル“kernel.h”から呼び出されるヘッダ・ファイル“packet.h”に定義されています。

```
typedef struct  t_rsem {
        ID      wtskid;   /* 待ちタスクの有無 */
        UINT    semcnt;   /* 現在資源数 */
        ATR     sematr;   /* 属性 */
        UINT    maxsem;   /* 最大資源数 */
} T_RSEM;
```

以下に，セマフォ情報の詳細を示します。

- wtskid

セマフォの待ちキューにタスクがキューイングされているか否かが格納されます。

マクロ	数値	説明
TSK_NONE	0	待ちキューにタスクはキューイングされていない
－	0 以外	待ちキューの先頭にキューイングされているタスクの ID

- semcnt

セマフォの現在資源数が格納されます。

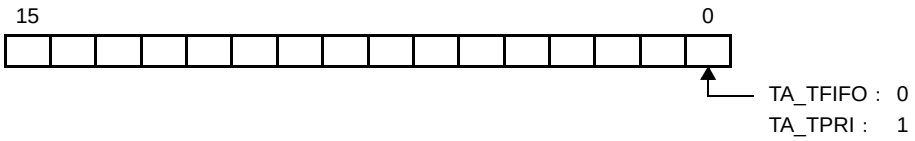
- sematr

セマフォの属性（キューイング方法）が格納されます。

- タスクのキューイング方法（ビット 0）

TA_TFIFO： 資源の獲得要求を行った順

TA_TPRI： タスクの優先度順



- maxsem

セマフォの最大資源数が格納されます。

13. 3. 3 イベントフラグ情報 T_RFLG

イベントフラグ情報は，標準ヘッダ・ファイル“kernel.h”から呼び出されるヘッダ・ファイル“packet.h”に定義されています。

```
typedef struct  t_rflg {
        ID      wtskid;   /* 待ちタスクの有無 */
        FLGPTN  flgptn;   /* 現在ビット・パターン */
        ATR     flgatr;   /* 属性 */
} T_RFLG;
```

以下に，イベントフラグ情報の詳細を示します。

- wtskid

イベントフラグの待ちキューにタスクがキューイングされているか否かが格納されます。

マクロ	数値	説明
TSK_NONE	0	待ちキューにタスクはキューイングされていない
—	0 以外	待ちキューの先頭にキューイングされているタスクの ID

- flgptn

イベントフラグの現在ビット・パターンが格納されます。

- flgatr

イベントフラグの属性（キューイング方法，最大タスク数，クリア方法）が格納されます。

- タスクのキューイング方法（ビット 0）

TA_TFIFO： ビット・パターンの判定要求を行った順

TA_TPRI： タスクの優先度順

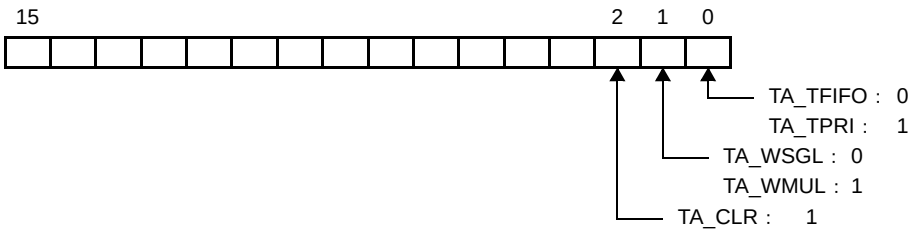
- キューイング可能な最大タスク数（ビット 1）

TA_WSGL： 1 個のタスク

TA_WMUL： 複数のタスク

- ビット・パターンのクリア方法（ビット 2）

TA_CLR： 要求条件を満足した際，ビット・パターンをクリア



備考　ビット・パターンのクリア方法が“要求条件を満足した際，ビット・パターンを非クリア”の場合には，ビット 2 に 0 が格納されます。

13. 3. 4 データ・キュー情報 T_RDTQ

データ・キュー情報は，標準ヘッダ・ファイル“kernel.h”から呼び出されるヘッダ・ファイル“packet.h”に定義されています。

```
typedef struct  t_rdtq {
        ID      stskid;     /* 送信待ちタスクの有無 */
        ID      rtskid;     /* 受信待ちタスクの有無 */
        UINT    sdtqcnt;    /* データ数 */
        ATR     dtqatr;     /* 属性 */
        UINT    dtqcnt;     /* 最大データ数 */
} T_RDTQ;
```

以下に，データ・キュー情報の詳細を示します。

- stskid

データ・キューの待ちキューに送信待ちタスクがキューイングされているか否かが格納されます。

マクロ	数値	説明
TSK_NONE	0	待ちキューに送信待ちタスクはキューイングされていない
−	0 以外	待ちキューの先頭にキューイングされている送信待ちタスクの ID

- rtskid

データ・キューの待ちキューに受信待ちタスクがキューイングされているか否かが格納されます。

マクロ	数値	説明
TSK_NONE	0	待ちキューに受信待ちタスクはキューイングされていない
−	0 以外	待ちキューの先頭にキューイングされている受信待ちタスクの ID

- sdtqcnt

データ・キューの待ちキューにキューイングされているデータの数が格納されます。

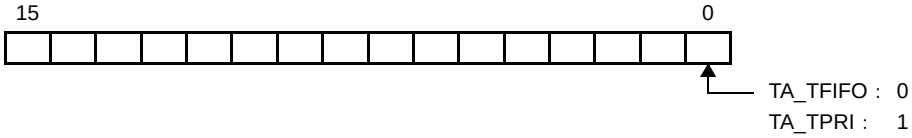
- dtqatr

データ・キューの属性（キューイング方法）が格納されます。

- タスクのキューイング方法（ビット 0）

TA_TFIFO： データの送信要求を行った順

TA_TPRI： タスクの優先度順



- dtqcnt

  データ・キューの待ちキューにキューイング可能なデータの最大データ数が格納されます。

13. 3. 5　メールボックス情報 T_RMBX

メールボックス情報は，標準ヘッダ・ファイル“kernel.h”から呼び出されるヘッダ・ファイル“packet.h”に定義されています。

```
typedef struct  t_rmbx {
        ID      wtskid;     /* 待ちタスクの有無 */
        T_MSG   *pk_msg;    /* 待ちメッセージの有無 */
        ATR     mbxatr;     /* 属性 */
} T_RMBX;
```

以下に，メールボックス情報の詳細を示します。

- wtskid

メールボックスの待ちキューにタスクがキューイングされているか否かが格納されます。

マクロ	数値	説明
TSK_NONE	0	待ちキューにタスクはキューイングされていない
－	0 以外	待ちキューの先頭にキューイングされているタスクの ID

- pk_msg

メールボックスの待ちキューにメッセージがキューイングされているか否かが格納されます。

マクロ	数値	説明
NULL	0	待ちキューにタスクはキューイングされていない
－	0 以外	待ちキューの先頭にキューイングされているメッセージの先頭アドレス

- mbxatr

メールボックスの属性（キューイング方法）が格納されます。

- タスクのキューイング方法（ビット 0）

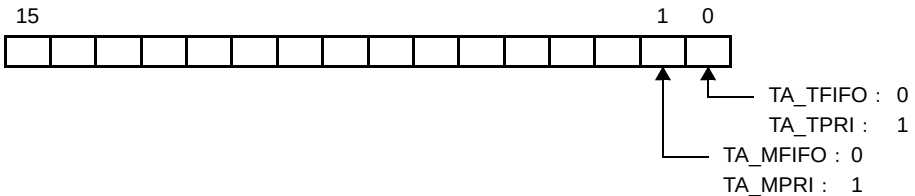
TA_TFIFO：　メッセージの受信要求を行った順

TA_TPRI：　タスクの優先度順

- メッセージのキューイング方法（ビット 1）

TA_MFIFO：　メッセージの送信要求を行った順

TA_MPRI：　メッセージの優先度順



13. 3. 6　ミューテックス情報 T_RMTX

ミューテックス情報は，標準ヘッダ・ファイル“kernel.h”から呼び出されるヘッダ・ファイル“packet.h”に定義されています。

```
typedef struct  t_rmtx {
        ID      htskid;     /* 獲得タスクの有無 */
        ID      wtskid;     /* 待ちタスクの有無 */
        ATR     mtxatr;     /* 属性 */
        PRI     ceilpri;    /* システム予約領域 */
} T_RMTX;
```

以下に，ミューテックス情報の詳細を示します。

- htskid

ミューテックスを獲得しているタスクがいるか否かが格納されます。

マクロ	数値	説明
TSK_NONE	0	獲得しているタスクはいない
－	0 以外	獲得しているタスクの ID

- wtskid

ミューテックスの待ちキューにタスクがキューイングされているか否かが格納されます。

マクロ	数値	説明
TSK_NONE	0	待ちキューにタスクはキューイングされていない
－	0 以外	待ちキューの先頭にキューイングされているタスクの ID

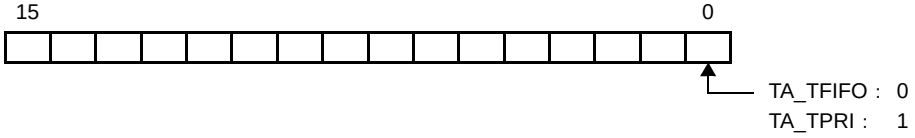
- mtxatr

ミューテックスの属性（キューイング方法）が格納されます。

- タスクのキューイング方法（ビット 0）

TA_TFIFO :　ミューテックスの獲得要求を行った順

TA_TPRI :　タスクの優先度順



- ceilpri

システム予約領域です。

13. 3. 7 固定長メモリ・プール情報 T_RMPF

固定長メモリ・プール情報は，標準ヘッダ・ファイル“kernel.h”から呼び出されるヘッダ・ファイル“packet.h”に定義されています。

```
typedef struct  t_rmpf {
        ID      wtskid;     /* 待ちタスクの有無 */
        UINT    fblkcnt;    /* 残りブロック数 */
        ATR     mpfatr;     /* 属性 */
} T_RMPF;
```

以下に，固定長メモリ・プール情報の詳細を示します。

- wtskid

固定長メモリ・プールの待ちキューにタスクがキューイングされているか否かが格納されます。

マクロ	数値	説明
TSK_NONE	0	待ちキューにタスクはキューイングされていない
－	0 以外	待ちキューの先頭にキューイングされているタスクの ID

- fblkcnt

獲得可能な残りブロック数が格納されます。

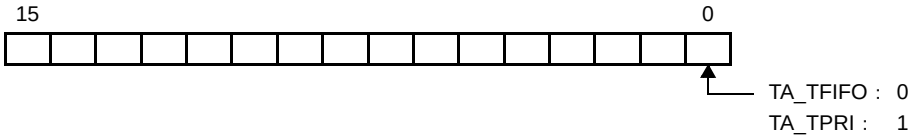
- mpfatr

固定長メモリ・プールの属性（キューイング方法）が格納されます。

- タスクのキューイング方法（ビット 0）

TA_TFIFO： 固定長メモリ・ブロックの獲得要求を行った順

TA_TPRI： タスクの優先度順



13. 3. 8 周期ハンドラ情報 T_RCYC

周期ハンドラ情報は，標準ヘッダ・ファイル“kernel.h”から呼び出されるヘッダ・ファイル“packet.h”に定義されています。

```
typedef struct  t_rcyc {
        STAT    cycstat;    /* 現在状態 */
        RELTIM  lefttim;    /* 残り時間 */
        ATR     cycatr;     /* 属性 */
        RELTIM  cyctim;     /* 起動周期 */
        RELTIM  cycphs;     /* 起動位相 */
        PE_ID   peid;       /*PE 番号 */
} T_RCYC;
```

以下に，周期ハンドラ情報の詳細を示します。

- cycstat

周期ハンドラの現在状態が格納されます。

マクロ	数値	説明
TCYC_STP	0x0	停止状態
TCYC_STA	0x1	動作状態

- lefttim

周期ハンドラが次に起動するまでの残り時間（単位：ミリ秒）が格納されます。

- cycatr

周期ハンドラの属性（記述言語，初期状態，保存有無）が格納されます。

- 周期ハンドラの記述言語（ビット 0）

TA_HLNG： C 言語

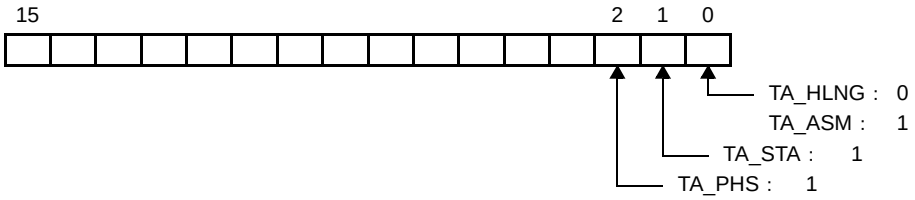
TA_ASM： アセンブリ言語

- 周期ハンドラの初期状態（ビット 1）

TA_STA： 動作状態

- 起動位相の保存有無（ビット 2）

TA_PHS： 起動位相を保存



備考 1. 周期ハンドラの初期状態が停止状態の場合には，ビット 1 に 0 が格納されます。

2. 周期ハンドラの起動位相保存の有無が“起動位相を保存しない”の場合には，ビット 2 に 0 が格納されます。

- cyctim

周期ハンドラの起動周期（単位：ミリ秒）が格納されます。

- cycphs

周期ハンドラの起動位相（単位：ミリ秒）が格納されます。

- peid

周期ハンドラが所属しているプロセッサ・エレメントの PE 番号が格納されます。

13. 3. 9　メッセージ（優先度なし）T_MSG

メッセージ（優先度なし）は，標準ヘッダ・ファイル“kernel.h”から呼び出されるヘッダ・ファイル“types.h”に定義されています。

```
typedef struct  t_msg {
        struct  t_msg   *msgque;    /* システム予約領域 */
        ……                          /* メッセージの本体 */
        ……
} T_MSG;
```

以下に，メッセージ（優先度なし）の詳細を示します。

- msgque

  システム予約領域です。

- ……

  メッセージの本体が格納されます。

  なお，“メッセージの本体”の構成，データ・タイプなどについては，送受信を行う処理プログラム間で規定します。

13. 3. 10　メッセージ（優先度あり）T_MSG_PRI

メッセージ（優先度あり）は，標準ヘッダ・ファイル“kernel.h”から呼び出されるヘッダ・ファイル“types.h”に定義されています。

```
typedef struct  t_msg_pri {
        T_MSG   msgque;             /* システム予約領域 */
        PRI     msgpri;             /* 優先度 */
        ……                          /* メッセージの本体 */
        ……
} T_MSG;
```

以下に，メッセージ（優先度あり）の詳細を示します。

- msgque

  システム予約領域です。

- msgpri

  メッセージの優先度が格納されます。

  備考　RI850MP における“メッセージの優先度”は，その値が小さいほど，高い優先度であることを意味します。

- ……

  メッセージの本体が格納されます。
  なお，“メッセージの本体”の構成，データ・タイプなどについては，送受信を行う処理プログラム間で規定します。

13. 3. 11　システム時刻 SYSTIM

システム時刻は，標準ヘッダ・ファイル“kernel.h”から呼び出されるヘッダ・ファイル“types.h”に定義されています。

```
typedef struct  t_systim {
        UW      ltime;    /* システム時刻（下位 32 ビット）*/
        UH      utime;    /* システム時刻（上位 16 ビット）*/
} SYSTIM;
```

以下に，システム時刻の詳細を示します。

- ltime

  システム時刻（単位：ミリ秒）の下位 32 ビットが格納されます。

- utime

  システム時刻（単位：ミリ秒）の上位 16 ビットが格納されます。

備考　システム時刻とは，カーネル初期化部における“システム時刻の初期化”，または set_tim ／ iset_tim の発行による“システム時刻の変更”からの相対的な経過時間であり，例外要因コード tintno で指定されたタイマ割り込みが発生した際，基本クロック周期 tbase で指定された値で更新されます。

13. 4　サービス・コール・リファレンス

本節では，RI850MP が提供するサービス・コールについて，次の記述フォーマットに従って説明します。

図 13—1　サービス・コールの記述フォーマット

(1) → ....

(2) → **[概要]**

....
....

(3) → **[記述形式]**

```
…  .........
```

(4) → **[引数]**

I/O	引数	説明
…	…　.........	....

(5) → **[機能]**

....
....

(6) → **[戻り値]**

マクロ	数値	説明
....	…	....

(1) 名称

サービス・コールの名称を示しています。

(2) [概要]

サービス・コールの機能概要を示しています。

(3)［記述形式］

サービス・コールを C 言語で記述された処理プログラムから発行する際の記述形式を示しています。

(4)［引数］

サービス・コールの引数を次の形式で示しています。

I/O	引数	説明
(a)	(b)	(c)

(a) I/O

引数の種類

I　…　入力引数

O　…　出力引数

(b) 引数

引数のデータ・タイプ

(c) 説明

引数の説明

(5)［機能］

サービス・コールの機能詳細を示しています。

(6)［戻り値］

サービス・コールからの戻り値を次の形式で示しています。

マクロ	数値	説明
(a)	(b)	(c)

(a) マクロ

戻り値のマクロ

(b) 数値

マクロの定義値

(c) 説明

戻り値の説明

13. 4. 1　タスク管理機能

以下に，RI850MP がタスク管理機能として提供しているサービス・コールを示します。

表 13—11　タスク管理機能

サービス・コール名	機能概要
act_tsk ／ iact_tsk	タスクの起動
can_act ／ ican_act	起動要求のキャンセル
ext_tsk	自タスクの終了
ter_tsk	タスクの強制終了
chg_pri ／ ichg_pri	現在優先度の変更
get_pri ／ iget_pri	現在優先度の参照
ref_tsk ／ iref_tsk	タスク情報の参照

act_tsk
iact_tsk

[概要]

タスクの起動

[記述形式]

```
ER  act_tsk(ID tskid);
ER  iact_tsk(ID tskid);
```

[引数]

I/O	引数	説明
I	ID *tskid*;	タスクの ID TSK_SELF： 自タスク 数値： タスクの ID

[機能]

tskid で指定されたタスクに起動要求を発行し，休止状態から実行可能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，対象タスクが休止状態以外であった場合には，タスクの状態操作は行わず，起動要求ネスト・カウンタに 1 を加算します。

備考 1. 実行可能状態へと遷移したタスクは，初期優先度 itskpri に対応したレディ・キューの末尾にキューイングされます。

2. 本サービス・コールの発行に伴い起動要求ネスト数が 127 を越える場合には，起動要求の発行操作は行わず，戻り値として“E_QOVR（= -43）”を返却します。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_ID	-18	*tskid* が不正 - タスク情報で未定義の ID - 非タスクからの発行時に TSK_SELF を指定

マクロ	数値	説明
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行
E_QOVR	-43	キューイング・オーバフロー - 起動要求ネスト数が最大起動要求ネスト数“127”を越える

can_act
ican_act

[概要]

起動要求のキャンセル

[記述形式]

```
ER_UINT can_act(ID tskid);
ER_UINT ican_act(ID tskid);
```

[引数]

I/O	引数	説明
I	ID *tskid*;	タスクの ID TSK_SELF： 自タスク 数値： タスクの ID

[機能]

tskid で指定されたタスクに保持されている起動要求をキャンセル（起動要求ネスト・カウンタに 0 を設定）し，戻り値として“キャンセルした起動要求ネスト数（act_tsk ／ iact_tsk のネスト数）”を返却します。

備考 1. 対象タスクが休止状態の場合には，戻り値として“0”を返却します。

2. 本サービス・コールでは，タスクの状態操作（実行可能状態から休止状態への遷移など）は行われません。

[戻り値]

マクロ	数値	説明
－	－	正常終了 - キャンセルした起動要求ネスト数
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_ID	-18	*tskid* が不正 - タスク情報で未定義の ID - 非タスクからの発行時に TSK_SELF を指定
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行

ext_tsk

［概要］

自タスクの終了

［記述形式］

```
void  ext_tsk(void);
```

［引数］

なし

［機能］

自タスクを実行状態から休止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，自タスクに起動要求が保持されていた（カウント値が 0 以外であった）場合には，タスクの状態操作を行ったのち，act_tsk ／ iact_tsk と同等の処理が行われます。

備考 1. 本サービス・コールでは，自タスクを休止状態へと遷移させる際，以下の操作も併せて行います。

- 獲得しているミューテックスの返却
- 現在優先度を初期優先度 itskpri と同値に変更
- 起床要求ネスト・カウンタに 0 を設定
- 強制待ち要求ネスト・カウンタに 0 を設定
- 割り込み状態を属性 tskatr の初期割り込み状態と同状態に変更

2. RI850MP では，タスク内で return 命令が呼び出された場合，本サービス・コールと同等の処理を行います。

3. SCT 情報で本サービス・コールを使用する旨の定義が行われていない場合，以後の動作が保証されません。

4. 本サービス・コールを非タスク，CPU ロック状態，またはディスパッチ禁止状態から発行した場合，以後の動作が保証されません。

［戻り値］

なし

ter_tsk

[概要]

タスクの強制終了

[記述形式]

```
ER  ter_tsk(ID tskid);
```

[引数]

I/O	引数	説明
I	ID *tskid*;	タスクの ID

[機能]

tskid で指定されたタスクを休止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，対象タスクに起動要求が保持されていた（カウント値が 0 以外であった）場合には，タスクの状態操作を行ったのち，act_tsk ／ iact_tsk と同等の処理が行われます。

備考 本サービス・コールでは，対象タスクを休止状態へと遷移させる際，以下の操作も併せて行います。

- 獲得しているミューテックスの返却
- 現在優先度を初期優先度 itskpri と同値に変更
- 起床要求ネスト・カウンタに 0 を設定
- 強制待ち要求ネスト・カウンタに 0 を設定
- 割り込み状態を属性 tskatr の初期割り込み状態と同状態に変更

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_ID	-18	*tskid* が不正 - タスク情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - 非タスクから発行 - CPU ロック状態から発行

マクロ	数値	説明
E_ILUSE	-28	サービス・コールの使用方法が不正 - 対象タスクが他 PE に所属 - *tskid* に自タスクを指定
E_OBJ	-41	オブジェクト状態エラー - 対象タスクが休止状態

chg_pri
ichg_pri

[概要]

現在優先度の変更

[記述形式]

```
ER  chg_pri(ID tskid, PRI tskpri);
ER  ichg_pri(ID tskid, PRI tskpri);
```

[引数]

I/O	引数	説明
I	ID *tskid*;	タスクの ID TSK_SELF： 自タスク 数値： タスクの ID
I	PRI *tskpri*;	変更後の現在優先度 TPRI_INI： 初期優先度 数値： 変更後の現在優先度

[機能]

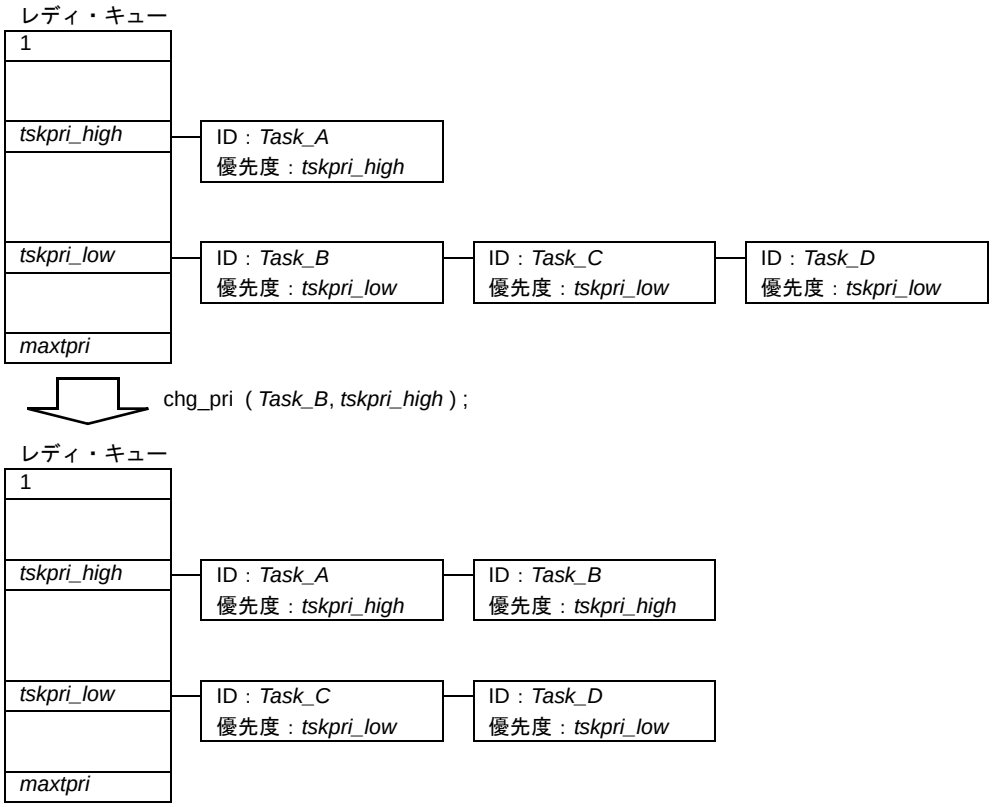
tskid で指定されたタスクの現在優先度を *tskpri* で指定された値に変更します。

ただし，本サービス・コールを発行した際，対象タスクが実行状態，または実行可能状態であった場合には，現在優先度の変更操作を行ったのち，該当タスクを *tskpri* で指定された優先度に対応したレディ・キューの末尾へとつなぎ替える処理が行われます。

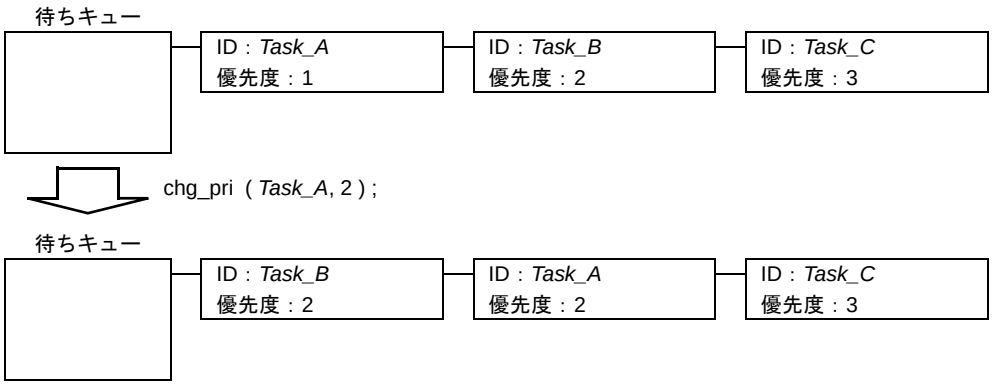
また，本サービス・コールを発行した際，対象タスクが“タスクの優先度順”で待ちキューにキューイングされていた場合には，現在優先度の変更操作を行ったのち，該当タスクを待ちキューの適切な箇所へとつなぎ替える処理が行われます。

備考 1. RI850MP における“タスクの優先度”は，その値が小さいほど，高い優先度であることを意味します。

2. 以下に，本サービス・コールの発行に伴う，レディ・キューの状態変更イメージを示します。



3. 以下に，本サービス・コールの発行に伴う，待ちキューの状態変更イメージを示します。



［戻り値］

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない

マクロ	数値	説明
E_PAR	-17	*tskpri* が不正 - *tskpri* ＜ 0 - *tskpri* ＞ 最大優先度 maxtpri
E_ID	-18	*tskid* が不正 - タスク情報で未定義の ID - 非タスクからの発行時に TSK_SELF を指定
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行
E_ILUSE	-28	サービス・コールの使用方法が不正 - 対象タスクが他 PE に所属
E_OBJ	-41	オブジェクト状態エラー - 対象タスクが休止状態

get_pri
iget_pri

[概要]

現在優先度の参照

[記述形式]

```
ER  get_pri(ID tskid, PRI *p_tskpri);
ER  iget_pri(ID tskid, PRI *p_tskpri);
```

[引数]

I/O	引数	説明
I	`ID tskid;`	タスクの ID TSK_SELF： 自タスク 数値： タスクの ID
O	`PRI *p_tskpri;`	現在優先度を格納する領域へのポインタ

[機能]

tskid で指定されたタスクの現在優先度を獲得し，*p_tskpri* で指定された領域に格納します。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_PAR	-17	*p_tskpri* が不正 - *p_tskpri* = 0
E_ID	-18	*tskid* が不正 - タスク情報で未定義の ID - 非タスクからの発行時に TSK_SELF を指定
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行
E_OBJ	-41	オブジェクト状態エラー - 対象タスクが休止状態

ref_tsk
iref_tsk

[概要]

タスク情報の参照

[記述形式]

```
ER  ref_tsk(ID tskid, T_RTSK *pk_rtsk);
ER  iref_tsk(ID tskid, T_RTSK *pk_rtsk);
```

[引数]

I/O	引数	説明
I	ID tskid;	タスクの ID TSK_SELF : 自タスク 数値 : タスクの ID
O	T_RTSK *pk_rtsk;	タスク情報を格納する領域へのポインタ

【タスク情報 T_RTSK】

```
typedef struct  t_rtsk {
        STAT    tskstat;    /* 現在状態 */
        PRI     tskpri;     /* 現在優先度 */
        PRI     tskbpri;    /* システム予約領域 */
        STAT    tskwait;    /* 待ち要因 */
        ID      wobjid;     /* オブジェクトの ID*/
        TMO     lefttmo;    /* 残り時間 */
        UINT    actcnt;     /* 起動要求ネスト数 */
        UINT    wupcnt;     /* 起床要求ネスト数 */
        UINT    sus_cnt;    /* 強制待ち要求ネスト数 */
        ATR     tskatr;     /* 属性 */
        PRI     itskpri;    /* 初期優先度 */
        PE_ID   peid;       /*PE 番号 */
} T_RTSK;
```

[機能]

tskid で指定されたタスクのタスク情報（現在状態など）を *pk_rtsk* で指定された領域に格納します。

備考　タスク情報についての詳細は，「13. 3. 1　タスク情報 T_RTSK」を参照してください。

［戻り値］

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_PAR	-17	*pk_rtsk* が不正 - *pk_rtsk* = 0
E_ID	-18	*tskid* が不正 - タスク情報で未定義の ID - 非タスクからの発行時に TSK_SELF を指定
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行

13. 4. 2　タスク付属同期機能

以下に，RI850MP がタスク付属同期機能として提供しているサービス・コールを示します。

表 13—12　タスク付属同期機能

サービス・コール名	機能概要
slp_tsk	起床待ち状態への移行
tslp_tsk	起床待ち状態への移行（タイムアウトあり）
wup_tsk ／ iwup_tsk	起床待ち状態からの再開
can_wup ／ ican_wup	起床要求のキャンセル
rel_wai ／ irel_wai	待ち状態の強制解除
sus_tsk ／ isus_tsk	強制待ち状態への移行
rsm_tsk ／ irsm_tsk	強制待ち状態からの再開
frsm_tsk ／ ifrsm_tsk	強制待ち状態からの強制再開
dly_tsk	時間経過待ち状態への移行

slp_tsk

[概要]

起床待ち状態への移行

[記述形式]

```
ER  slp_tsk(void);
```

[引数]

なし

[機能]

自タスクを実行状態から起床待ち状態へと遷移させます。

なお，起床待ち状態の解除は，以下の場合に行われ，起床待ち状態から実行可能状態へと遷移します。

起床待ち状態の解除	戻り値
wup_tsk ／ iwup_tsk の発行により，起床要求が発行された	E_OK
rel_wai ／ irel_wai の発行により，起床待ち状態を強制的に解除された	E_RLWAI

備考 本サービス・コールを発行した際，自タスクに起床要求が保持されていた（カウント値が 0 以外であった）場合には，タスクの状態操作は行わず，起床要求ネスト・カウンタから 1 を減算します。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_CTX	-25	コンテキスト・エラー - 非タスクから発行 - CPU ロック状態から発行 - ディスパッチ禁止状態から発行
E_RLWAI	-49	起床待ち状態の強制解除 - rel_wai ／ irel_wai の発行

tslp_tsk

[概要]

起床待ち状態への移行（タイムアウトあり）

[記述形式]

```
ER  tslp_tsk(TMO tmout);
```

[引数]

I/O	引数	説明
I	TMO *tmout*;	待ち時間（単位：ミリ秒） TMO_FEVR： 永久待ち TMO_POL： ポーリング 数値： 待ち時間

[機能]

自タスクを実行状態から起床待ち状態へと遷移させます。

なお，起床待ち状態の解除は，以下の場合に行われ，起床待ち状態から実行可能状態へと遷移します。

起床待ち状態の解除	戻り値
wup_tsk ／ iwup_tsk の発行により，起床要求が発行された	E_OK
rel_wai ／ irel_wai の発行により，起床待ち状態を強制的に解除された	E_RLWAI
tmout で指定された待ち時間が経過した	E_TMOUT

備考 1. 本サービス・コールを発行した際，自タスクに起床要求が保持されていた（カウント値が 0 以外であった）場合には，タスクの状態操作は行わず，起床要求ネスト・カウンタから 1 を減算します。

2. *tmout* に TMO_FEVR を指定した場合は slp_tsk と同等の処理が行われます。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_PAR	-17	*tmout* が不正 - *tmout* < TMO_FEVR

マクロ	数値	説明
E_CTX	-25	コンテキスト・エラー - 非タスクから発行 - CPU ロック状態から発行 - ディスパッチ禁止状態から発行
E_RLWAI	-49	起床待ち状態の強制解除 - rel_wai ／ irel_wai の発行
E_TMOUT	-50	タイムアウト - *tmout* で指定された待ち時間が経過

wup_tsk
iwup_tsk

[概要]

起床待ち状態からの再開

[記述形式]

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

[引数]

I/O	引数	説明
I	ID *tskid*;	タスクの ID TSK_SELF： 自タスク 数値： タスクの ID

[機能]

tskid で指定されたタスクに起床要求を発行し，起床待ち状態から実行可能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，対象タスクが起床待ち状態以外であった場合には，タスクの状態操作は行わず，起床要求ネスト・カウンタに 1 を加算します。

備考 本サービス・コールの発行に伴い起床要求ネスト数が 127 を越える場合には，起床要求の発行操作は行わず，戻り値として“E_QOVR（= -43）”を返却します。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_ID	-18	*tskid* が不正 - タスク情報で未定義の ID - 非タスクからの発行時に TSK_SELF を指定
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行

マクロ	数値	説明
E_OBJ	-41	オブジェクト状態エラー - 対象タスクが休止状態
E_QOVR	-43	キューイング・オーバフロー - 起床要求ネスト数が最大起床要求ネスト数“127”を越える

can_wup
ican_wup

[概要]

起床要求のキャンセル

[記述形式]

```
ER_UINT can_wup(ID tskid);
ER_UINT ican_wup(ID tskid);
```

[引数]

I/O	引数	説明
I	ID *tskid*;	タスクの ID TSK_SELF： 自タスク 数値： タスクの ID

[機能]

tskid で指定されたタスクに保持されている起床要求をキャンセル（起床要求ネスト・カウンタに 0 を設定）し，戻り値として“キャンセルした起床要求ネスト数（wup_tsk ／ iwup_tsk のネスト数）”を返却します。

備考 本サービス・コールでは，タスクの状態操作（待ち状態から実行可能状態への遷移など）は行われません。

[戻り値]

マクロ	数値	説明
－	－	正常終了 - キャンセルした起床要求ネスト数
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_ID	-18	*tskid* が不正 - タスク情報で未定義の ID - 非タスクからの発行時に TSK_SELF を指定
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行
E_OBJ	-41	オブジェクト状態エラー - 対象タスクが休止状態

rel_wai
irel_wai

[概要]

待ち状態の強制解除

[記述形式]

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

[引数]

I/O	引数	説明
I	ID *tskid*;	タスクの ID

[機能]

tskid で指定されたタスクの待ち状態を強制的に解除します。これにより，該当タスクは待ちキューから外れ，待ち状態から実行可能状態へ，または二重待ち状態から強制待ち状態へと遷移します。

備考 1. 本サービス・コールでは，強制待ち状態の解除は行いません。

2. 本サービス・コールの発行に伴い待ち状態を強制的に解除されたタスクには，待ち状態へと遷移するきっかけとなったサービス・コール（slp_tsk，dly_tsk，wai_sem など）の戻り値として“E_RLWAI（= -49）”を返却します。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_ID	-18	*tskid* が不正 - タスク情報で未定義の ID - 非タスクからの発行時に TSK_SELF を指定
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行

マクロ	数値	説明
E_OBJ	-41	オブジェクト状態エラー - 対象タスクが待ち状態，二重待ち状態以外

sus_tsk
isus_tsk

[概要]

強制待ち状態への移行

[記述形式]

```
ER  sus_tsk(ID tskid);
ER  isus_tsk(ID tskid);
```

[引数]

I/O	引数	説明
I	ID *tskid*;	タスクの ID TSK_SELF： 自タスク 数値： タスクの ID

[機能]

tskid で指定されたタスクに強制待ち要求を発行し，強制待ち状態，または二重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，対象タスクが強制待ち状態，または二重待ち状態であった場合には，タスクの状態操作は行わず，強制待ち要求ネスト・カウンタに 1 を加算します。

備考 本サービス・コールの発行に伴い強制待ち要求ネスト数が 127 を越える場合には，強制待ち要求の発行操作は行わず，戻り値として“E_QOVR（= -43）”を返却します。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_ID	-18	*tskid* が不正 - タスク情報で未定義の ID - 非タスクからの発行時に TSK_SELF を指定
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行 - ディスパッチ禁止状態からの発行時に自タスクを指定

マクロ	数値	説明
E_OBJ	-41	オブジェクト状態エラー - 対象タスクが休止状態
E_QOVR	-43	キューイング・オーバフロー - 強制待ち要求ネスト数が最大強制待ち要求ネスト数“127”を越える

rsm_tsk
irsm_tsk

[概要]

強制待ち状態からの再開

[記述形式]

```
ER  rsm_tsk(ID tskid);
ER  irsm_tsk(ID tskid);
```

[引数]

I/O	引数	説明
I	ID *tskid*;	タスクの ID

[機能]

tskid で指定されたタスクに強制待ち解除要求を発行し，実行可能状態，または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，対象タスクに強制待ち要求が保持されていた（カウント値が 0 以外であった）場合には，タスクの状態操作は行わず，強制待ち要求ネスト・カウンタから 1 を減算します。

備考 本サービス・コールでは，強制待ち解除要求のキューイングが行われません。このため，対象タスクが強制待ち状態，または二重待ち状態以外の場合には，戻り値として“E_OBJ（= -41）”を返却します。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_ID	-18	*tskid* が不正 - タスク情報で未定義の ID - 非タスクからの発行時に TSK_SELF を指定
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行
E_OBJ	-41	オブジェクト状態エラー - 対象タスクが強制待ち状態，二重待ち状態以外

frsm_tsk
ifrsm_tsk

[概要]

強制待ち状態からの強制再開

[記述形式]

```
ER  frsm_tsk(ID tskid);
ER  ifrsm_tsk(ID tskid);
```

[引数]

I/O	引数	説明
I	ID *tskid*;	タスクの ID

[機能]

tskid で指定されたタスクの強制待ち状態を強制的に解除（強制待ち要求ネスト・カウンタに 0 を設定）します。これにより，対象タスクは，強制待ち状態から実行可能状態へ，または二重待ち状態から待ち状態へと遷移します。

備考 1. 本サービス・コールでは，待ち状態の解除は行いません。

2. 本サービス・コールでは，強制待ち解除要求のキューイングが行われません。このため，対象タスクが強制待ち状態，または二重待ち状態以外の場合には，戻り値として“E_OBJ（= -41）”を返却します。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_ID	-18	*tskid* が不正 - タスク情報で未定義の ID - 非タスクからの発行時に TSK_SELF を指定
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行
E_OBJ	-41	オブジェクト状態エラー - 対象タスクが強制待ち状態，二重待ち状態以外

dly_tsk

[概要]

時間経過待ち状態への移行

[記述形式]

```
ER  dly_tsk(RELTIM dlytim);
```

[引数]

I/O	引数	説明
I	RELTIM *dlytim*;	待ち時間（単位：ミリ秒）

[機能]

自タスクを実行状態から時間経過待ち状態へと遷移させます。

なお，時間経過待ち状態の解除は，以下の場合に行われ，時間経過待ち状態から実行可能状態へと遷移します。

時間経過待ち状態の解除	戻り値
dlytim で指定された待ち時間が経過した	E_OK
rel_wai ／ irel_wai の発行により，時間経過待ち状態を強制的に解除された	E_RLWAI

備考 *dlytim* に 0 を指定した場合であっても，タスクの状態操作は行われます。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了 - *dlytim* で指定された待ち時間が経過
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_CTX	-25	コンテキスト・エラー - 非タスクから発行 - CPU ロック状態から発行 - ディスパッチ禁止状態から発行
E_RLWAI	-49	時間経過待ち状態の強制解除 - rel_wai ／ irel_wai の発行

13. 4. 3　同期・通信機能（セマフォ）

以下に，RI850MP が同期・通信機能（セマフォ）として提供しているサービス・コールを示します。

表 13—13　同期・通信機能（セマフォ）

サービス・コール名	機能概要
sig_sem ／ isig_sem	資源の返却
wai_sem	資源の獲得
pol_sem ／ ipol_sem	資源の獲得（ポーリング）
twai_sem	資源の獲得（タイムアウトあり）
ref_sem ／ iref_sem	セマフォ情報の参照

sig_sem
isig_sem

[概要]

資源の返却

[記述形式]

```
ER  sig_sem(ID semid);
ER  isig_sem(ID semid);
```

[引数]

I/O	引数	説明
I	ID *semid*;	セマフォの ID

[機能]

semid で指定されたセマフォに資源を返却（セマフォ・カウンタに 1 を加算）します。

ただし，本サービス・コールを発行した際，対象セマフォの待ちキューにタスクがキューイングされていた場合には，資源の返却操作は行わず，該当タスク（待ちキューの先頭タスク）に資源を渡します。これにより，該当タスクは待ちキューから外れ，資源獲得待ち状態から実行可能状態へ，または二重待ち状態から強制待ち状態へと遷移します。

備考 本サービス・コールの発行に伴い資源数が最大資源数 maxsem を越える場合には，資源の返却操作は行わず，戻り値として“E_QOVR（= -43）”を返却します。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_ID	-18	*semid* が不正 - セマフォ情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行
E_QOVR	-43	キューイング・オーバフロー - 資源数が最大資源数 maxsem を越える

wai_sem

［概要］

資源の獲得

［記述形式］

```
ER  wai_sem(ID semid);
```

［引数］

I/O	引数	説明
I	ID *semid*;	セマフォの ID

［機能］

semid で指定されたセマフォから資源を獲得（セマフォ・カウンタから 1 を減算）します。

ただし，本サービス・コールを発行した際，対象セマフォから資源を獲得することができなかった（既にカウント値が 0 であった）場合には，自タスクをセマフォの待ちキューにキューイングしたのち，実行状態から資源獲得待ち状態へと遷移させます。

なお，資源獲得待ち状態の解除は，以下の場合に行われ，資源獲得待ち状態から実行可能状態へと遷移します。

資源獲得待ち状態の解除	戻り値
sig_sem ／ isig_sem の発行により，対象セマフォに資源が返却された	E_OK
rel_wai ／ irel_wai の発行により，資源獲得待ち状態を強制的に解除された	E_RLWAI

備考　タスクをセマフォの待ちキューにキューイングする順序は，属性 sematr で指定したキューイング方法（資源の獲得要求を行った順，またはタスクの優先度順）となります。

［戻り値］

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_ID	-18	*semid* が不正 - セマフォ情報で未定義の ID

マクロ	数値	説明
E_CTX	-25	コンテキスト・エラー - 非タスクから発行 - CPU ロック状態から発行 - ディスパッチ禁止状態から発行
E_RLWAI	-49	資源獲得待ち状態の強制解除 - rel_wai ／ irel_wai の発行

pol_sem
ipol_sem

[概要]

資源の獲得（ポーリング）

[記述形式]

```
ER  pol_sem(ID semid);
ER  ipol_sem(ID semid);
```

[引数]

I/O	引数	説明
I	ID *semid*;	セマフォの ID

[機能]

semid で指定されたセマフォから資源を獲得（セマフォ・カウンタから 1 を減算）します。

ただし，本サービス・コールを発行した際，対象セマフォから資源を獲得することができなかった（既にカウント値が 0 であった）場合には，戻り値として“E_TMOUT（= -50）”を返却します。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_ID	-18	*semid* が不正 - セマフォ情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行
E_TMOUT	-50	ポーリング失敗 - 対象セマフォの資源数が 0

twai_sem

［概要］

資源の獲得（タイムアウトあり）

［記述形式］

```
ER  twai_sem(ID semid, TMO tmout);
```

［引数］

I/O	引数	説明
I	ID *semid*;	セマフォの ID
I	TMO *tmout*;	待ち時間（単位：ミリ秒） TMO_FEVR：　永久待ち TMO_POL：　ポーリング 数値：　待ち時間

［機能］

semid で指定されたセマフォから資源を獲得（セマフォ・カウンタから 1 を減算）します。

ただし，本サービス・コールを発行した際，対象セマフォから資源を獲得することができなかった（既にカウント値が 0 であった）場合には，自タスクをセマフォの待ちキューにキューイングしたのち，実行状態から資源獲得待ち状態へと遷移させます。

なお，資源獲得待ち状態の解除は，以下の場合に行われ，資源獲得待ち状態から実行可能状態へと遷移します。

資源獲得待ち状態の解除	戻り値
sig_sem ／ isig_sem の発行により，対象セマフォに資源が返却された	E_OK
rel_wai ／ irel_wai の発行により，資源獲得待ち状態を強制的に解除された	E_RLWAI
tmout で指定された待ち時間が経過した	E_TMOUT

備考 1. タスクをセマフォの待ちキューにキューイングする順序は，属性 sematr で指定したキューイング方法（資源の獲得要求を行った順，またはタスクの優先度順）となります。

2. *tmout* に TMO_FEVR を指定した場合は wai_sem と，TMO_POL を指定した場合は pol_sem ／ ipol_sem と同等の処理が行われます。

［戻り値］

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_PAR	-17	*tmout* が不正 - *tmout* ＜ TMO_FEVR
E_ID	-18	*semid* が不正 - セマフォ情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - 非タスクから発行 - CPU ロック状態から発行 - ディスパッチ禁止状態から発行
E_RLWAI	-49	資源獲得待ち状態の強制解除 - rel_wai ／ irel_wai の発行
E_TMOUT	-50	タイムアウト - *tmout* で指定された待ち時間が経過

ref_sem
iref_sem

[概要]

セマフォ情報の参照

[記述形式]

```
ER  ref_sem(ID semid, T_RSEM *pk_rsem);
ER  iref_sem(ID semid, T_RSEM *pk_rsem);
```

[引数]

I/O	引数	説明
I	ID semid;	セマフォの ID
O	T_RSEM *pk_rsem;	セマフォ情報を格納する領域へのポインタ

【セマフォ情報 T_RSEM】

```
typedef struct  t_rsem {
        ID      wtskid;   /* 待ちタスクの有無 */
        UINT    semcnt;   /* 現在資源数 */
        ATR     sematr;   /* 属性 */
        UINT    maxsem;   /* 最大資源数 */
} T_RSEM;
```

[機能]

semid で指定されたセマフォのセマフォ情報（待ちタスクの有無など）を *pk_rsem* で指定された領域に格納します。

備考 セマフォ情報についての詳細は，「13. 3. 2 セマフォ情報 T_RSEM」を参照してください。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない

マクロ	数値	説明
E_PAR	-17	*pk_rsem* が不正 - *pk_rsem* = 0
E_ID	-18	*semid* が不正 - セマフォ情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行

13. 4. 4　同期・通信機能（イベントフラグ）

以下に，RI850MP が同期・通信機能（イベントフラグ）として提供しているサービス・コールを示します。

表 13—14　同期・通信機能（イベントフラグ）

サービス・コール名	機能概要
set_flg ／ iset_flg	ビット・パターンの変更
clr_flg ／ iclr_flg	ビット・パターンのクリア
wai_flg	ビット・パターンの判定
pol_flg ／ ipol_flg	ビット・パターンの判定（ポーリング）
twai_flg	ビット・パターンの判定（タイムアウトあり）
ref_flg ／ iref_flg	イベントフラグ情報の参照

set_flg
iset_flg

［概要］

ビット・パターンの変更

［記述形式］

```
ER  set_flg(ID flgid, FLGPTN setptn);
ER  iset_flg(ID flgid, FLGPTN setptn);
```

［引数］

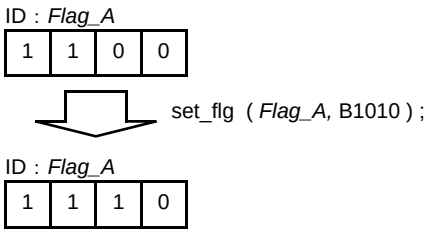
I/O	引数	説明
I	ID *flgid*;	イベントフラグの ID
I	FLGPTN *setptn*;	設定するビット・パターン

［機能］

flgid で指定されたイベントフラグの現在ビット・パターンと *setptn* で指定されたビット・パターンの論理和 OR をとり，該当結果を対象イベントフラグの現在ビット・パターンとして設定します。

ただし，本サービス・コールを発行した際，対象イベントフラグの待ちキューにキューイングされているタスクの要求条件を満足した場合には，ビット・パターンの設定操作を行ったのち，該当タスクを待ちキューから外します。これにより，該当タスクは，イベントフラグ待ち状態から実行可能状態へ，または二重待ち状態から強制待ち状態へと遷移します。

備考 1. 対象イベントフラグの現在ビット・ターンが B1100，*setptn* で指定されたビット・パターンが B1010 の場合には，B1110 が現在ビット・パターンとして設定されます。



2. 対象イベントフラグに TA_WMUL 属性（キューイング可能な最大タスク数：複数のタスク）が付与されていた場合，“本サービス・コールの発行に伴い要求条件を満足したか否か”の判定対象となるタスクは，

TA_CLR 属性（クリア方法：要求条件を満足した際，ビット・パターンをクリア）が付与されているか否かにより異なります。

- TA_CLR 属性が付与されている場合
  待ちキューにキューイングされている先頭タスクから要求条件を満足したタスクまでが判定対象となります。
- TA_CLR 属性が付与されていない場合
  待ちキューにキューイングされている全タスクが判定対象となります。

［戻り値］

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_ID	-18	*flgid* が不正 - イベントフラグ情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行

clr_flg
iclr_flg

[概要]

ビット・パターンのクリア

[記述形式]

```
ER  clr_flg(ID flgid, FLGPTN clrptn);
ER  iclr_flg(ID flgid, FLGPTN clrptn);
```

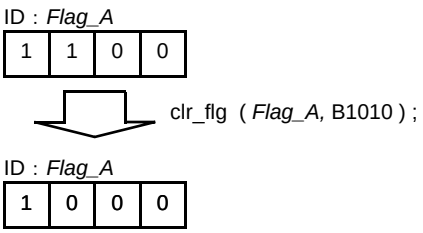
[引数]

I/O	引数	説明
I	ID *flgid*;	イベントフラグの ID
I	FLGPTN *clrptn*;	クリアするビット・パターン

[機能]

flgid で指定されたイベントフラグの現在ビット・パターンと *clrptn* で指定されたクリア・パターンの論理積 AND をとり，該当結果を対象イベントフラグの現在ビット・パターンとして設定します。

備考 対象イベントフラグの現在ビット・ターンが B1100，*clrptn* で指定されたクリア・パターンが B1010 の場合には，B1000 が現在ビット・パターンとして設定されます。



[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない

マクロ	数値	説明
E_ID	-18	*flgid* が不正 - イベントフラグ情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行

wai_flg

［概要］

ビット・パターンの判定

［記述形式］

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

［引数］

I/O	引数	説明
I	ID　　*flgid*;	イベントフラグの ID
I	FLGPTN　*waiptn*;	要求ビット・パターン
I	MODE　　*wfmode*;	要求条件 TWF_ANDW： AND 待ち TWF_ORW：　OR 待ち
O	FLGPTN　**p_flgptn*;	ビット・パターンを格納する領域へのポインタ

［機能］

flgid で指定されたイベントフラグの現在ビット・パターンが *waiptn* で指定された要求ビット・パターンと *wfmode* で指定された要求条件を満足するビット・パターンであるのか否かを判定し，満足するビット・パターンであった場合には，該当現在ビット・パターンを *p_flgptn* で指定された領域に格納します。

ただし，本サービス・コールを発行した際，対象イベントフラグの現在ビット・パターンが満足するビット・パターンでなかった場合には，自タスクをイベントフラグの待ちキューにキューイングしたのち，実行状態からイベントフラグ待ち状態へと遷移させます。

なお，イベントフラグ待ち状態の解除は，以下の場合に行われ，イベントフラグ待ち状態から実行可能状態へと遷移します。

イベントフラグ待ち状態の解除	戻り値
set_flg ／ iset_flg の発行により，対象イベントフラグに要求条件を満足するビット・パターンが設定された	E_OK
rel_wai ／ irel_wai の発行により，イベントフラグ待ち状態を強制的に解除された	E_RLWAI

以下に，ビット・パターンの判定基準である要求条件 *wfmode* について示します。

- TWF_ANDW

waiptn で“1”を設定している全ビットが対象イベントフラグに設定されているか否かを判定します。

- TWF_ORW

waiptn で“1”を設定しているビットのうち，いずれかのビットが対象イベントフラグに設定されているか否かを判定します。

備考 1. RI850MP では，既に待ちキューにタスクがキューイングされている TA_WSGL 属性（キューイング可能な最大タスク数：1 個のタスク）のイベントフラグに対して本サービス・コールが発行された場合，要求条件の即時成立／不成立を問わず，戻り値として“E_ILUSE（= -28）”を返却します。

2. タスクを TA_WMUL 属性（キューイング可能な最大タスク数：複数タスク）のイベントフラグの待ちキューにキューイングする順序は，属性 flgatr で指定したキューイング方法（ビット・パターンの判定要求を行った順，またはタスクの優先度順）となります。

3. rel_wai ／ irel_wai の発行により，イベントフラグ待ち状態を強制的に解除された場合，*p_flgptn* で指定された領域の内容は不定値となります。

［戻り値］

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_PAR	-17	*waiptn*，*wfmode*，*p_flgptn* が不正 - *waiptn* = 0 - *wfmode* が不正 - *p_flgptn* = 0
E_ID	-18	*flgid* が不正 - イベントフラグ情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - 非タスクから発行 - CPU ロック状態から発行 - ディスパッチ禁止状態から発行
E_ILUSE	-28	サービス・コールの使用方法が不正 - 既にタスクがキューイングされている TA_WSGL 属性のイベントフラグに対する発行
E_RLWAI	-49	イベントフラグ待ち状態の強制解除 - rel_wai ／ irel_wai の発行

pol_flg
ipol_flg

［概要］

ビット・パターンの判定（ポーリング）

［記述形式］

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

［引数］

I/O	引数	説明
I	ID　　*flgid*;	イベントフラグの ID
I	FLGPTN　*waiptn*;	要求ビット・パターン
I	MODE　　*wfmode*;	要求条件 TWF_ANDW： AND 待ち TWF_ORW：　OR 待ち
O	FLGPTN　**p_flgptn*;	ビット・パターンを格納する領域へのポインタ

［機能］

flgid で指定されたイベントフラグの現在ビット・パターンが *waiptn* で指定された要求ビット・パターンと *wfmode* で指定された要求条件を満足するビット・パターンであるのか否かを判定し，満足するビット・パターンであった場合には，該当現在ビット・パターンを *p_flgptn* で指定された領域に格納します。

ただし，本サービス・コールを発行した際，対象イベントフラグの現在ビット・パターンが満足するビット・パターンでなかった場合には，戻り値として“E_TMOUT（= -50）”を返却します。

以下に，ビット・パターンの判定基準である要求条件 *wfmode* について示します。

- TWF_ANDW

  waiptn で“1”を設定している全ビットが対象イベントフラグに設定されているか否かを判定します。

- TWF_ORW

  waiptn で“1”を設定しているビットのうち，いずれかのビットが対象イベントフラグに設定されているか否かを判定します。

備考　RI850MP では，既に待ちキューにタスクがキューイングされている TA_WSGL 属性（キューイング可能な最大タスク数：1 個のタスク）のイベントフラグに対して本サービス・コールが発行された場合，要求条件の即時成立／不成立を問わず，戻り値として“E_ILUSE（= -28）”を返却します。

［戻り値］

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_PAR	-17	*waiptn*，*wfmode*，*p_flgptn* が不正 - *waiptn* = 0 - *wfmode* が不正 - *p_flgptn* = 0
E_ID	-18	*flgid* が不正 - イベントフラグ情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行
E_ILUSE	-28	サービス・コールの使用方法が不正 - 既にタスクがキューイングされている TA_WSGL 属性のイベントフラグに対する発行
E_TMOUT	-50	ポーリング失敗 - 対象イベントフラグのビット・パターンが要求条件を満足していない

twai_flg

[概要]

ビット・パターンの判定（タイムアウトあり）

[記述形式]

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

[引数]

I/O	引数	説明
I	ID *flgid*;	イベントフラグの ID
I	FLGPTN *waiptn*;	要求ビット・パターン
I	MODE *wfmode*;	要求条件 TWF_ANDW： AND 待ち TWF_ORW： OR 待ち
O	FLGPTN **p_flgptn*;	ビット・パターンを格納する領域へのポインタ
I	TMO *tmout*;	待ち時間（単位：ミリ秒） TMO_FEVR： 永久待ち TMO_POL： ポーリング 数値： 待ち時間

[機能]

flgid で指定されたイベントフラグの現在ビット・パターンが *waiptn* で指定された要求ビット・パターンと *wfmode* で指定された要求条件を満足するビット・パターンであるのか否かを判定し，満足するビット・パターンであった場合には，該当現在ビット・パターンを *p_flgptn* で指定された領域に格納します。

ただし，本サービス・コールを発行した際，対象イベントフラグの現在ビット・パターンが満足するビット・パターンでなかった場合には，自タスクをイベントフラグの待ちキューにキューイングしたのち，実行状態からイベントフラグ待ち状態へと遷移させます。

なお，イベントフラグ待ち状態の解除は，以下の場合に行われ，イベントフラグ待ち状態から実行可能状態へと遷移します。

イベントフラグ待ち状態の解除	戻り値
set_flg ／ iset_flg の発行により，対象イベントフラグに要求条件を満足するビット・パターンが設定された	E_OK
rel_wai ／ irel_wai の発行により，イベントフラグ待ち状態を強制的に解除された	E_RLWAI
tmout で指定された待ち時間が経過した	E_TMOUT

以下に，ビット・パターンの判定基準である要求条件 *wfmode* について示します。

- TWF_ANDW

  waiptn で“1”を設定している全ビットが対象イベントフラグに設定されているか否かを判定します。

- TWF_ORW

  waiptn で“1”を設定しているビットのうち，いずれかのビットが対象イベントフラグに設定されているか否かを判定します。

備考 1. RI850MP では，既に待ちキューにタスクがキューイングされている TA_WSGL 属性（キューイング可能な最大タスク数：1 個のタスク）のイベントフラグに対して本サービス・コールが発行された場合，要求条件の即時成立／不成立を問わず，戻り値として“E_ILUSE（= -28）”を返却します。

2. タスクを TA_WMUL 属性（キューイング可能な最大タスク数：複数タスク）のイベントフラグの待ちキューにキューイングする順序は，属性 flgatr で指定したキューイング方法（ビット・パターンの判定要求を行った順，またはタスクの優先度順）となります。

3. rel_wai ／ irel_wai の発行により，イベントフラグ待ち状態を強制的に解除された場合，*p_flgptn* で指定された領域の内容は不定値となります。

4. *tmout* に TMO_FEVR を指定した場合は wai_flg と，TMO_POL を指定した場合は pol_flg ／ ipol_flg と同等の処理が行われます。

［戻り値］

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_PAR	-17	*waiptn*，*wfmode*，*p_flgptn*，*tmout* が不正 - *waiptn* = 0 - *wfmode* が不正 - *p_flgptn* = 0 - *tmout* ＜ TMO_FEVR
E_ID	-18	*flgid* が不正 - イベントフラグ情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - 非タスクから発行 - CPU ロック状態から発行 - ディスパッチ禁止状態から発行
E_ILUSE	-28	サービス・コールの使用方法が不正 - 既にタスクがキューイングされている TA_WSGL 属性のイベントフラグに対する発行
E_RLWAI	-49	イベントフラグ待ち状態の強制解除 - rel_wai ／ irel_wai の発行

マクロ	数値	説明
E_TMOUT	-50	タイムアウト - *tmout* で指定された待ち時間が経過

ref_flg
iref_flg

[概要]

イベントフラグ情報の参照

[記述形式]

```
ER  ref_flg(ID flgid, T_RFLG *pk_rflg);
ER  iref_flg(ID flgid, T_RFLG *pk_rflg);
```

[引数]

I/O	引数	説明
I	ID flgid;	イベントフラグの ID
O	T_RFLG *pk_rflg;	イベントフラグ情報を格納する領域へのポインタ

【イベントフラグ情報 T_RFLG】

```
typedef struct  t_rflg {
        ID      wtskid;   /* 待ちタスクの有無 */
        FLGPTN  flgptn;   /* 現在ビット・パターン */
        ATR     flgatr;   /* 属性 */
} T_RFLG;
```

[機能]

flgid で指定されたイベントフラグのイベントフラグ情報（待ちタスクの有無など）を *pk_rtsk* で指定された領域に格納します。

備考 イベントフラグ情報についての詳細は，「13. 3. 3 イベントフラグ情報 T_RFLG」を参照してください。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない

マクロ	数値	説明
E_PAR	-17	*pk_rflg* が不正 - *pk_rflg* = 0
E_ID	-18	*flgid* が不正 - イベントフラグ情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行

13. 4. 5　同期・通信機能（データ・キュー）

以下に，RI850MP が同期・通信機能（データ・キュー）として提供しているサービス・コールを示します。

表 13—15　同期・通信機能（データ・キュー）

サービス・コール名	機能概要
snd_dtq	データ・キューへの送信
psnd_dtq ／ ipsnd_dtq	データ・キューへの送信（ポーリング）
tsnd_dtq	データ・キューへの送信（タイムアウトあり）
fsnd_dtq ／ 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 ／ iprcv_dtq	データ・キューからの受信（ポーリング）
trcv_dtq	データ・キューからの受信（タイムアウトあり）
ref_dtq ／ iref_dtq	データ・キュー情報の参照

snd_dtq

[概要]

データ・キューへの送信

[記述形式]

```
ER  snd_dtq(ID dtqid, VP_INT data);
```

[引数]

I/O	引数	説明
I	ID *dtqid*;	データ・キューの ID
I	VP_INT *data*;	送信するデータ

[機能]

dtqid で指定されたデータ・キューに *data* で指定されたデータを送信します。

ただし，本サービス・コールを発行した際，対象データ・キューのバッファ領域に最大データ数 dtqcnt で指定した値と同数のデータが書き込まれていた場合には，データの送信操作は行わず，自タスクをデータ・キューの待ちキューにキューイングしたのち，実行状態からデータ送信待ち状態へと遷移させます。

なお，データ送信待ち状態の解除は，以下の場合に行われ，データ送信待ち状態から実行可能状態へと遷移します。

データ送信待ち状態の解除	戻り値
rcv_dtq の発行により，対象データ・キューのバッファ領域に空き領域が確保された	E_OK
prcv_dtq ／ iprcv_dtq の発行により，対象データ・キューのバッファ領域に空き領域が確保された	E_OK
trcv_dtq の発行により，対象データ・キューのバッファ領域に空き領域が確保された	E_OK
rel_wai ／ irel_wai の発行により，データ送信待ち状態を強制的に解除された	E_RLWAI

また，本サービス・コールを発行した際，対象データ・キューの待ちキューにタスクがキューイングされていた場合には，データの送信操作は行わず，該当タスク（待ちキューの先頭タスク）にデータを渡します。これにより，該当タスクは待ちキューから外れ，データ受信待ち状態から実行可能状態へ，または二重待ち状態から強制待ち状態へと遷移します。

備考 1. RI850MP における同期・通信機能（データ・キュー）では，データの送信処理として，データ・キューのバッファ領域にデータの書き込みを，データの受信処理として，データ・キューのバッファ領域からデータの読み出しを行います。

2. タスクをデータ・キューの待ちキューにキューイングする順序は，属性 dtqatr で指定したキューイング方法（データの送信要求を行った順，またはタスクの優先度順）となります。

［戻り値］

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_ID	-18	*dtqid* が不正 - データ・キュー情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - 非タスクから発行 - CPU ロック状態から発行 - ディスパッチ禁止状態から発行
E_RLWAI	-49	データ送信待ち状態の強制解除 - rel_wai ／ irel_wai の発行

psnd_dtq
ipsnd_dtq

[概要]

データ・キューへの送信（ポーリング）

[記述形式]

```
ER  psnd_dtq(ID dtqid, VP_INT data);
ER  ipsnd_dtq(ID dtqid, VP_INT data);
```

[引数]

I/O	引数	説明
I	ID *dtqid*;	データ・キューの ID
I	VP_INT *data*;	送信するデータ

[機能]

dtqid で指定されたデータ・キューに *data* で指定されたデータを送信します。

ただし，本サービス・コールを発行した際，対象データ・キューのバッファ領域に最大データ数 dtqcnt で指定した値と同数のデータが書き込まれていた場合には，戻り値として“E_TMOUT（= -50）”を返却します。

また，本サービス・コールを発行した際，対象データ・キューの待ちキューにタスクがキューイングされていた場合には，データの送信操作は行わず，該当タスク（待ちキューの先頭タスク）にデータを渡します。これにより，該当タスクは待ちキューから外れ，データ受信待ち状態から実行可能状態へ，または二重待ち状態から強制待ち状態へと遷移します。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_ID	-18	*dtqid* が不正 - データ・キュー情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行

マクロ	数値	説明
E_TMOUT	-50	ポーリング失敗 - データ数が最大データ数 dtqcnt を越える

tsnd_dtq

［概要］

データ・キューへの送信（タイムアウトあり）

［記述形式］

```
ER  tsnd_dtq(ID dtqid, VP_INT data, TMO tmout);
```

［引数］

I/O	引数	説明
I	ID *dtqid*;	データ・キューの ID
I	VP_INT *data*;	送信するデータ
I	TMO *tmout*;	待ち時間（単位：ミリ秒） TMO_FEVR：永久待ち TMO_POL：ポーリング 数値：待ち時間

［機能］

dtqid で指定されたデータ・キューに *data* で指定されたデータを送信します。

ただし，本サービス・コールを発行した際，対象データ・キューのバッファ領域に最大データ数 dtqcnt で指定した値と同数のデータが書き込まれていた場合には，データの送信操作は行わず，自タスクをデータ・キューの待ちキューにキューイングしたのち，実行状態からデータ送信待ち状態へと遷移させます。

なお，データ送信待ち状態の解除は，以下の場合に行われ，データ送信待ち状態から実行可能状態へと遷移します。

データ送信待ち状態の解除	戻り値
rcv_dtq の発行により，対象データ・キューのバッファ領域に空き領域が確保された	E_OK
prcv_dtq ／ iprcv_dtq の発行により，対象データ・キューのバッファ領域に空き領域が確保された	E_OK
trcv_dtq の発行により，対象データ・キューのバッファ領域に空き領域が確保された	E_OK
rel_wai ／ irel_wai の発行により，データ送信待ち状態を強制的に解除された	E_RLWAI
tmout で指定された待ち時間が経過した	E_TMOUT

また，本サービス・コールを発行した際，対象データ・キューの待ちキューにタスクがキューイングされていた場合には，データの送信操作は行わず，該当タスク（待ちキューの先頭タスク）にデータを渡します。これにより，該当タスクは待ちキューから外れ，データ受信待ち状態から実行可能状態へ，または二重待ち状態から強制待ち状態へと遷移します。

備考 1. タスクをデータ・キューの待ちキューにキューイングする順序は，属性 dtqatr で指定したキューイング方法（データの送信要求を行った順，またはタスクの優先度順）となります。

2. *tmout* に TMO_FEVR を指定した場合は snd_dtq と，TMO_POL を指定した場合は psnd_dtq ／ ipsnd_dtq と同等の処理が行われます。

［戻り値］

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_PAR	-17	*tmout* が不正 - *tmout* ＜ TMO_FEVR
E_ID	-18	*dtqid* が不正 - データ・キュー情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - 非タスクから発行 - CPU ロック状態から発行 - ディスパッチ禁止状態から発行
E_RLWAI	-49	データ送信待ち状態の強制解除 - rel_wai ／ irel_wai の発行
E_TMOUT	-50	タイムアウト - *tmout* で指定された待ち時間が経過

fsnd_dtq
ifsnd_dtq

[概要]

データ・キューへの強制送信

[記述形式]

```
ER  fsnd_dtq(ID dtqid, VP_INT data);
ER  ifsnd_dtq(ID dtqid, VP_INT data);
```

[引数]

I/O	引数	説明
I	ID *dtqid*;	データ・キューの ID
I	VP_INT *data*;	送信するデータ

[機能]

dtqid で指定されたデータ・キューに *data* で指定されたデータを送信します。

ただし，本サービス・コールを発行した際，対象データ・キューのバッファ領域に最大データ数 dtqcnt で指定した値と同数のデータが書き込まれていた場合には，バッファ領域に書き込まれてから最も時間が経過しているデータを削除後，*data* で指定されたデータを送信します。

また，本サービス・コールを発行した際，対象データ・キューの待ちキューにタスクがキューイングされていた場合には，データの送信操作は行わず，該当タスク（待ちキューの先頭タスク）にデータを渡します。これにより，該当タスクは待ちキューから外れ，データ受信待ち状態から実行可能状態へ，または二重待ち状態から強制待ち状態へと遷移します。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_ID	-18	*dtqid* が不正 - データ・キュー情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行

マクロ	数値	説明
E_ILUSE	-28	サービス・コールの使用方法が不正 - 対象データ・キューの最大データ数 dtqcnt が 0

rcv_dtq

[概要]

データ・キューからの受信

[記述形式]

```
ER  rcv_dtq(ID dtqid, VP_INT *p_data);
```

[引数]

I/O	引数	説明
I	ID *dtqid*;	データ・キューの ID
O	VP_INT *p_data;	データを格納する領域へのポインタ

[機能]

dtqid で指定されたデータ・キューからデータを受信し *p_data* で指定された領域に格納します。

ただし，本サービス・コールを発行した際，対象データ・キューからデータを受信することができなかった（バッファ領域にデータが書き込まれていなかった）場合には，自タスクをデータ・キューの待ちキューにキューイングしたのち，実行状態からデータ受信待ち状態へと遷移させます。

なお，データ受信待ち状態の解除は，以下の場合に行われ，データ受信待ち状態から実行可能状態へと遷移します。

データ受信待ち状態の解除	戻り値
snd_dtq の発行により，対象データ・キューにデータが送信された	E_OK
prcv_dtq ／ iprcv_dtq の発行により，対象データ・キューにデータが送信された	E_OK
tsnd_dtq の発行により，対象データ・キューにデータが送信された	E_OK
fsnd_dtq ／ ifsnd_dtq の発行により，対象データ・キューにデータが送信された	E_OK
rel_wai ／ irel_wai の発行により，データ受信待ち状態を強制的に解除された	E_RLWAI

備考 1. タスクをデータ・キューの待ちキューにキューイングする順序は，データの受信要求を行った順となります。

2. rel_wai ／ irel_wai の発行により，データ受信待ち状態を強制的に解除された場合，*p_data* で指定された領域の内容は不定値となります。

［戻り値］

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_PAR	-17	*p_data* が不正 - *p_data* = 0
E_ID	-18	*dtqid* が不正 - データ・キュー情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - 非タスクから発行 - CPU ロック状態から発行 - ディスパッチ禁止状態から発行
E_RLWAI	-49	データ受信待ち状態の強制解除 - rel_wai ／ irel_wai の発行

prcv_dtq
iprcv_dtq

[概要]

データ・キューからの受信（ポーリング）

[記述形式]

```
ER  prcv_dtq(ID dtqid, VP_INT *p_data);
ER  iprcv_dtq(ID dtqid, VP_INT *p_data);
```

[引数]

I/O	引数	説明
I	ID dtqid;	データ・キューの ID
O	VP_INT *p_data;	データを格納する領域へのポインタ

[機能]

dtqid で指定されたデータ・キューからデータを受信し *p_data* で指定された領域に格納します。

ただし，本サービス・コールを発行した際，対象データ・キューからデータを受信することができなかった（バッファ領域にデータが書き込まれていなかった）場合には，戻り値として“E_TMOUT（= -50）”を返却します。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_PAR	-17	*p_data* が不正 - *p_data* = 0
E_ID	-18	*dtqid* が不正 - データ・キュー情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行
E_TMOUT	-50	ポーリング失敗 - 対象データ・キューのバッファ領域にデータが書き込まれていない

trcv_dtq

[概要]

データ・キューからの受信（タイムアウトあり）

[記述形式]

```
ER  trcv_dtq(ID dtqid, VP_INT *p_data, TMO tmout);
```

[引数]

I/O	引数	説明
I	ID *dtqid*;	データ・キューの ID
O	VP_INT **p_data*;	データを格納する領域へのポインタ
I	TMO *tmout*;	待ち時間（単位：ミリ秒） TMO_FEVR： 永久待ち TMO_POL： ポーリング 数値： 待ち時間

[機能]

dtqid で指定されたデータ・キューからデータを受信し *p_data* で指定された領域に格納します。

ただし，本サービス・コールを発行した際，対象データ・キューからデータを受信することができなかった（バッファ領域にデータが書き込まれていなかった）場合には，自タスクをデータ・キューの待ちキューにキューイングしたのち，実行状態からデータ受信待ち状態へと遷移させます。

なお，データ受信待ち状態の解除は，以下の場合に行われ，データ受信待ち状態から実行可能状態へと遷移します。

データ受信待ち状態の解除	戻り値
snd_dtq の発行により，対象データ・キューにデータが送信された	E_OK
psnd_dtq ／ ipsnd_dtq の発行により，対象データ・キューにデータが送信された	E_OK
tsnd_dtq の発行により，対象データ・キューにデータが送信された	E_OK
fsnd_dtq ／ ifsnd_dtq の発行により，対象データ・キューにデータが送信された	E_OK
rel_wai ／ irel_wai の発行により，データ受信待ち状態を強制的に解除された	E_RLWAI
tmout で指定された待ち時間が経過した	E_TMOUT

備考 1. タスクをデータ・キューの待ちキューにキューイングする順序は，データの受信要求を行った順となります。

2. rel_wai ／ irel_wai の発行，および *tmout* で指定された待ち時間の経過により，データ受信待ち状態を強制的に解除された場合，*p_data* で指定された領域の内容は不定値となります。

3. *tmout* に TMO_FEVR を指定した場合は rcv_dtq と，TMO_POL を指定した場合は prcv_dtq ／ iprcv_dtq と同等の処理が行われます。

［戻り値］

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_PAR	-17	*p_data*，*tmout* が不正 - *p_data* = 0 - *tmout* ＜ TMO_FEVR
E_ID	-18	*dtqid* が不正 - データ・キュー情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - 非タスクから発行 - CPU ロック状態から発行 - ディスパッチ禁止状態から発行
E_RLWAI	-49	データ受信待ち状態の強制解除 - rel_wai ／ irel_wai の発行
E_TMOUT	-50	タイムアウト - *tmout* で指定された待ち時間が経過

ref_dtq
iref_dtq

［概要］

データ・キュー情報の参照

［記述形式］

```
ER  ref_dtq(ID dtqid, T_RDTQ *pk_rdtq);
ER  iref_dtq(ID dtqid, T_RDTQ *pk_rdtq);
```

［引数］

I/O	引数	説明
I	ID　　dtqid;	データ・キューの ID
O	T_RDTQ　*pk_rdtq;	データ・キュー情報を格納する領域へのポインタ

【データ・キュー情報 T_RDTQ】

```
typedef struct  t_rdtq {
        ID      stskid;     /* 送信待ちタスクの有無 */
        ID      rtskid;     /* 受信待ちタスクの有無 */
        UINT    sdtqcnt;    /* データ数 */
        ATR     dtqatr;     /* 属性 */
        UINT    dtqcnt;     /* 最大データ数 */
} T_RDTQ;
```

［機能］

dtqid で指定されたデータ・キューのデータ・キュー情報（送信待ちタスクの有無など）を *pk_rdtq* で指定された領域に格納します。

備考　データ・キュー情報についての詳細は，「13. 3. 4　データ・キュー情報 T_RDTQ」を参照してください。

［戻り値］

マクロ	数値	説明
E_OK	0	正常終了

マクロ	数値	説明
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_PAR	-17	*pk_rdtq* が不正 - *pk_rdtq* = 0
E_ID	-18	*dtqid* が不正 - データ・キュー情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行

13. 4. 6　同期・通信機能（メールボックス）

以下に，RI850MP が同期・通信機能（メールボックス）として提供しているサービス・コールを示します。

表 13—16　同期・通信機能（メールボックス）

サービス・コール名	機能概要
snd_mbx ／ isnd_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 ／ iprcv_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）
ref_mbx ／ iref_mbx	メールボックス情報の参照

snd_mbx
isnd_mbx

[概要]

メールボックスへの送信

[記述形式]

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

[引数]

I/O	引数	説明
I	ID mbxid;	メールボックスの ID
I	T_MSG *pk_msg;	メッセージを格納した領域へのポインタ

【メッセージ（優先度なし）T_MSG】

```
typedef struct  t_msg {
        struct  t_msg   *msgque;    /* システム予約領域 */
        ·····                       /* メッセージの本体 */
        ·····
} T_MSG;
```

【メッセージ（優先度あり）T_MSG_PRI】

```
typedef struct  t_msg_pri {
        T_MSG   msgque;             /* システム予約領域 */
        PRI     msgpri;             /* 優先度 */
        ·····                       /* メッセージの本体 */
        ·····
} T_MSG_PRI;
```

[機能]

mbxid で指定されたメールボックスに *pk_msg* で指定されたメッセージを送信します。

ただし，本サービス・コールを発行した際，対象メールボックスの待ちキューにタスクがキューイングされていた場合には，メッセージの送信操作は行わず，該当タスク（待ちキューの先頭タスク）にメッセージを渡します。これによ

り，該当タスクは待ちキューから外れ，メッセージ受信待ち状態から実行可能状態へ，または二重待ち状態から強制待ち状態へと遷移します。

備考 1. RI850MP における同期・通信機能（メールボックス）では，メッセージの送受信処理として，該当メッセージの先頭アドレスの受け渡しのみを行い，該当メッセージの内容が他の領域にコピーされることはありません。

2. メッセージをメールボックスの待ちキューにキューイングする順序は，属性 mbxatr で指定したキューイング方法（メッセージの送信要求を行った順，またはメッセージの優先度順）となります。

3. メッセージ（優先度なし），メッセージ（優先度あり）についての詳細は，「13. 3. 9　メッセージ（優先度なし）T_MSG」，「13. 3. 10　メッセージ（優先度あり）T_MSG_PRI」を参照してください。

［戻り値］

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_PAR	-17	*pk_msg*，msgpri が不正 - *pk_msg* = 0 - msgpri ≦ 0 - msgpri ＞ 最大優先度 maxmpri
E_ID	-18	*mbxid* が不正 - メールボックス情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行

rcv_mbx

[概要]

メールボックスからの受信

[記述形式]

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

[引数]

I/O	引数	説明
I	ID mbxid;	メールボックスの ID
O	T_MSG **ppk_msg;	メッセージの先頭アドレスを格納する領域へのポインタ

【メッセージ（優先度なし）T_MSG】

```
typedef struct  t_msg {
        struct  t_msg   *msgque;    /* システム予約領域 */
        ……                          /* メッセージの本体 */
        ……
} T_MSG;
```

【メッセージ（優先度あり）T_MSG_PRI】

```
typedef struct  t_msg_pri {
        T_MSG   msgque;             /* システム予約領域 */
        PRI     msgpri;             /* 優先度 */
        ……                          /* メッセージの本体 */
        ……
} T_MSG_PRI;
```

[機能]

mbxid で指定されたメールボックスからメッセージを受信し，該当メッセージの先頭アドレスを *ppk_msg* で指定された領域に格納します。

ただし，本サービス・コールを発行した際，対象メールボックスからメッセージを受信することができなかった（待ちキューにメッセージがキューイングされていなかった）場合には，自タスクをメールボックスの待ちキューにキューイングしたのち，実行状態からメッセージ受信待ち状態へと遷移させます。

なお，メッセージ受信待ち状態の解除は，以下の場合に行われ，メッセージ受信待ち状態から実行可能状態へと遷移します。

メッセージ受信待ち状態の解除	戻り値
snd_mbx ／ isnd_mbx の発行により，対象メールボックスにメッセージが送信された	E_OK
rel_wai ／ irel_wai の発行により，メッセージ受信待ち状態を強制的に解除された	E_RLWAI

備考 1. タスクをメールボックスの待ちキューにキューイングする順序は，属性 mbxatr で指定したキューイング方法（メッセージの受信要求を行った順，またはタスクの優先度順）となります。

2. rel_wai ／ irel_wai の発行により，メッセージ受信待ち状態を強制的に解除された場合，*ppk_msg* で指定された領域の内容は不定値となります。

3. メッセージ（優先度なし），メッセージ（優先度あり）についての詳細は，「13. 3. 9　メッセージ（優先度なし）T_MSG」，「13. 3. 10　メッセージ（優先度あり）T_MSG_PRI」を参照してください。

［戻り値］

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_PAR	-17	*ppk_msg* が不正 - *ppk_msg* = 0
E_ID	-18	*mbxid* が不正 - メールボックス情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - 非タスクから発行 - CPU ロック状態から発行 - ディスパッチ禁止状態から発行
E_RLWAI	-49	メッセージ受信待ち状態の強制解除 - rel_wai ／ irel_wai の発行

prcv_mbx
iprcv_mbx

［概要］

メールボックスからの受信（ポーリング）

［記述形式］

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

［引数］

I/O	引数	説明
I	ID　　*mbxid*;	メールボックスの ID
O	T_MSG ***ppk_msg*;	メッセージの先頭アドレスを格納する領域へのポインタ

【メッセージ（優先度なし）T_MSG】

```
typedef struct  t_msg {
        struct  t_msg   *msgque;    /* システム予約領域 */
        ……                          /* メッセージの本体 */
        ……
} T_MSG;
```

【メッセージ（優先度あり）T_MSG_PRI】

```
typedef struct  t_msg_pri {
        T_MSG   msgque;             /* システム予約領域 */
        PRI     msgpri;             /* 優先度 */
        ……                          /* メッセージの本体 */
        ……
} T_MSG_PRI;
```

［機能］

mbxid で指定されたメールボックスからメッセージを受信し，該当メッセージの先頭アドレスを *ppk_msg* で指定された領域に格納します。

ただし，本サービス・コールを発行した際，対象メールボックスからメッセージを受信することができなかった（待ちキューにメッセージがキューイングされていなかった）場合には，戻り値として“E_TMOUT（= -50）”を返却します。

備考　メッセージ（優先度なし），メッセージ（優先度あり）についての詳細は，「13. 3. 9　メッセージ（優先度なし）T_MSG」，「13. 3. 10　メッセージ（優先度あり）T_MSG_PRI」を参照してください。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_PAR	-17	*ppk_msg* が不正 - *ppk_msg* = 0
E_ID	-18	*mbxid* が不正 - メールボックス情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行
E_TMOUT	-50	ポーリング失敗 - 対象メールボックスの待ちキューにメッセージがキューイングされていない

trcv_mbx

［概要］

メールボックスからの受信（タイムアウトあり）

［記述形式］

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

［引数］

I/O	引数	説明
I	ID　*mbxid*;	メールボックスの ID
O	T_MSG **ppk_msg;	メッセージの先頭アドレスを格納する領域へのポインタ
I	TMO　*tmout*;	待ち時間（単位：ミリ秒） TMO_FEVR：　永久待ち TMO_POL：　ポーリング 数値：　待ち時間

【メッセージ（優先度なし）T_MSG】

```
typedef struct  t_msg {
        struct  t_msg   *msgque;    /* システム予約領域 */
        ……                          /* メッセージの本体 */
        ……
} T_MSG;
```

【メッセージ（優先度あり）T_MSG_PRI】

```
typedef struct  t_msg_pri {
        T_MSG   msgque;             /* システム予約領域 */
        PRI     msgpri;             /* 優先度 */
        ……                          /* メッセージの本体 */
        ……
} T_MSG_PRI;
```

［機能］

mbxid で指定されたメールボックスからメッセージを受信し，該当メッセージの先頭アドレスを *ppk_msg* で指定された領域に格納します。

ただし，本サービス・コールを発行した際，対象メールボックスからメッセージを受信することができなかった（待ちキューにメッセージがキューイングされていなかった）場合には，自タスクをメールボックスの待ちキューにキューイングしたのち，実行状態からメッセージ受信待ち状態へと遷移させます。

なお，メッセージ受信待ち状態の解除は，以下の場合に行われ，メッセージ受信待ち状態から実行可能状態へと遷移します。

メッセージ受信待ち状態の解除	戻り値
snd_mbx ／ isnd_mbx の発行により，対象メールボックスにメッセージが送信された	E_OK
rel_wai ／ irel_wai の発行により，メッセージ受信待ち状態を強制的に解除された	E_RLWAI
tmout で指定された待ち時間が経過した	E_TMOUT

備考 1. タスクをメールボックスの待ちキューにキューイングする順序は，属性 mbxatr で指定したキューイング方法（メッセージの受信要求を行った順，またはタスクの優先度順）となります。

2. rel_wai ／ irel_wai の発行，および *tmout* で指定された待ち時間の経過により，メッセージ受信待ち状態を強制的に解除された場合，*ppk_msg* で指定された領域の内容は不定値となります。

3. *tmout* に TMO_FEVR を指定した場合は rcv_mbx と，TMO_POL を指定した場合は prcv_mbx ／ iprcv_mbx と同等の処理が行われます。

4. メッセージ（優先度なし），メッセージ（優先度あり）についての詳細は，「13. 3. 9　メッセージ（優先度なし）T_MSG」，「13. 3. 10　メッセージ（優先度あり）T_MSG_PRI」を参照してください。

［戻り値］

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_PAR	-17	*ppk_msg*，*tmout* が不正 - *ppk_msg* = 0 - *tmout* ＜ TMO_FEVR
E_ID	-18	*mbxid* が不正 - メールボックス情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - 非タスクから発行 - CPU ロック状態から発行 - ディスパッチ禁止状態から発行
E_RLWAI	-49	メッセージ受信待ち状態の強制解除 - rel_wai ／ irel_wai の発行
E_TMOUT	-50	タイムアウト - *tmout* で指定された待ち時間が経過

ref_mbx
iref_mbx

[概要]

メールボックス情報の参照

[記述形式]

```
ER  ref_mbx(ID mbxid, T_RMBX *pk_rmbx);
ER  iref_mbx(ID mbxid, T_RMBX *pk_rmbx);
```

[引数]

I/O	引数	説明
I	ID mbxid;	メールボックスの ID
O	T_RMBX *pk_rmbx;	メールボックス情報を格納する領域へのポインタ

【メールボックス情報 T_RMBX】

```
typedef struct  t_rmbx {
        ID      wtskid;     /* 待ちタスクの有無 */
        T_MSG   *pk_msg;    /* 待ちメッセージの有無 */
        ATR     mbxatr;     /* 属性 */
} T_RMBX;
```

[機能]

mbxid で指定されたメールボックスのメールボックス情報（待ちタスクの有無など）を *pk_rmbx* で指定された領域に格納します。

備考 メールボックス情報についての詳細は，「13. 3. 5 メールボックス情報 T_RMBX」を参照してください。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない

マクロ	数値	説明
E_PAR	-17	*pk_rmbx* が不正 - *pk_rmbx* = 0
E_ID	-18	*mbxid* が不正 - メールボックス情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行

13. 4. 7　拡張同期・通信機能

以下に，RI850MP が拡張同期・通信機能として提供しているサービス・コールを示します。

表 13—17　拡張同期・通信機能

サービス・コール名	機能概要
loc_mtx	ミューテックスの獲得
ploc_mtx	ミューテックスの獲得（ポーリング）
tloc_mtx	ミューテックスの獲得（タイムアウトあり）
unl_mtx	ミューテックスの返却
ref_mtx ／ iref_mtx	ミューテックス情報の参照

loc_mtx

［概要］

ミューテックスの獲得

［記述形式］

```
ER  loc_mtx(ID mtxid);
```

［引数］

I/O	引数	説明
I	ID *mtxid*;	ミューテックスの ID

［機能］

mtxid で指定されたミューテックスを獲得します。

ただし，本サービス・コールを発行した際，対象ミューテックスを獲得することができなかった（既に他タスクが獲得していた）場合には，自タスクをミューテックスの待ちキューにキューイングしたのち，実行状態からミューテックス獲得待ち状態へと遷移させます。

なお，ミューテックス獲得待ち状態の解除は，以下の場合に行われ，ミューテックス獲得待ち状態から実行可能状態へと遷移します。

ミューテックス獲得待ち状態の解除	戻り値
unl_mtx の発行により，対象ミューテックスが返却された	E_OK
ext_tsk の発行により，対象ミューテックスが返却された	E_OK
ter_tsk の発行により，対象ミューテックスが返却された	E_OK
rel_wai ／ irel_wai の発行により，ミューテックス獲得待ち状態を強制的に解除された	E_RLWAI

備考　タスクをミューテックスの待ちキューにキューイングする順序は，属性 mtxatr で指定したキューイング方法（ミューテックスの獲得要求を行った順，またはタスクの優先度順）となります。

［戻り値］

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない

マクロ	数値	説明
E_ID	-18	*mtxid* が不正 - ミューテックス情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - 非タスクから発行 - CPU ロック状態から発行 - ディスパッチ禁止状態から発行
E_ILUSE	-28	サービス・コールの使用方法が不正 - 既に自タスクが獲得しているミューテックスに対する発行
E_RLWAI	-49	ミューテックス獲得待ち状態の強制解除 - rel_wai ／ irel_wai の発行

ploc_mtx

［概要］

ミューテックスの獲得（ポーリング）

［記述形式］

```
ER  ploc_mtx(ID mtxid);
```

［引数］

I/O	引数	説明
I	ID *mtxid*;	ミューテックスの ID

［機能］

mtxid で指定されたミューテックスを獲得します。

ただし，本サービス・コールを発行した際，対象ミューテックスを獲得することができなかった（既に他タスクが獲得していた）場合には，戻り値として“E_TMOUT（= -50）”を返却します。

［戻り値］

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_ID	-18	*mtxid* が不正 - ミューテックス情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - 非タスクから発行 - CPU ロック状態から発行
E_ILUSE	-28	サービス・コールの使用方法が不正 - 既に自タスクが獲得しているミューテックスに対する発行
E_TMOUT	-50	ポーリング失敗 - 対象ミューテックスを他タスクが獲得している

tloc_mtx

［概要］

ミューテックスの獲得（タイムアウトあり）

［記述形式］

```
ER  tloc_mtx(ID mtxid, TMO tmout);
```

［引数］

I/O	引数	説明
I	ID *mtxid*;	ミューテックスの ID
I	TMO *tmout*;	待ち時間（単位：ミリ秒） TMO_FEVR： 永久待ち TMO_POL： ポーリング 数値： 待ち時間

［機能］

mtxid で指定されたミューテックスを獲得します。

ただし，本サービス・コールを発行した際，対象ミューテックスを獲得することができなかった（既に他タスクが獲得していた）場合には，自タスクをミューテックスの待ちキューにキューイングしたのち，実行状態からミューテックス獲得待ち状態へと遷移させます。

なお，ミューテックス獲得待ち状態の解除は，以下の場合に行われ，ミューテックス獲得待ち状態から実行可能状態へと遷移します。

ミューテックス獲得待ち状態の解除	戻り値
unl_mtx の発行により，対象ミューテックスが返却された	E_OK
ext_tsk の発行により，対象ミューテックスが返却された	E_OK
ter_tsk の発行により，対象ミューテックスが返却された	E_OK
rel_wai ／ irel_wai の発行により，ミューテックス獲得待ち状態を強制的に解除された	E_RLWAI
tmout で指定された待ち時間が経過した	E_TMOUT

備考 1. タスクをミューテックスの待ちキューにキューイングする順序は，属性 mtxatr で指定したキューイング方法（ミューテックスの獲得要求を行った順，またはタスクの優先度順）となります。

2. *tmout* に TMO_FEVR を指定した場合は loc_mtx と，TMO_POL を指定した場合は ploc_mtx と同等の処理が行われます。

［戻り値］

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_PAR	-17	*tmout* が不正 - *tmout* ＜ TMO_FEVR
E_ID	-18	*mtxid* が不正 - ミューテックス情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - 非タスクから発行 - CPU ロック状態から発行 - ディスパッチ禁止状態から発行
E_ILUSE	-28	サービス・コールの使用方法が不正 - 既に自タスクが獲得しているミューテックスに対する発行
E_RLWAI	-49	ミューテックス獲得待ち状態の強制解除 - rel_wai ／ irel_wai の発行
E_TMOUT	-50	タイムアウト - *tmout* で指定された待ち時間が経過

unl_mtx

［概要］

ミューテックスの返却

［記述形式］

```
ER  unl_mtx(ID mtxid);
```

［引数］

I/O	引数	説明
I	ID *mtxid*;	ミューテックスの ID

［機能］

mtxid で指定されたミューテックスを返却します。

ただし，本サービス・コールを発行した際，対象ミューテックスの待ちキューにタスクがキューイングされていた場合には，ミューテックスの返却操作は行わず，該当タスク（待ちキューの先頭タスク）にミューテックスを渡します。これにより，該当タスクは待ちキューから外れ，ミューテックス獲得待ち状態から実行可能状態へ，または二重待ち状態から強制待ち状態へと遷移します。

［戻り値］

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_ID	-18	*mtxid* が不正 - ミューテックス情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - 非タスクから発行 - CPU ロック状態から発行
E_ILUSE	-28	サービス・コールの使用方法が不正 - 既に自タスクが返却しているミューテックスに対する発行 - 他タスクが獲得しているミューテックスに対する発行

ref_mtx
iref_mtx

[概要]

ミューテックス情報の参照

[記述形式]

```
ER  ref_mtx(ID mtxid, T_RMTX *pk_rmtx);
ER  iref_mtx(ID mtxid, T_RMTX *pk_rmtx);
```

[引数]

I/O	引数	説明
I	ID mtxid;	ミューテックスの ID
O	T_RMTX *pk_rmtx;	ミューテックス情報を格納する領域へのポインタ

【ミューテックス情報 T_RMTX】

```
typedef struct  t_rmtx {
        ID      htskid;     /* 獲得タスクの有無 */
        ID      wtskid;     /* 待ちタスクの有無 */
        ATR     mtxatr;     /* 属性 */
        PRI     ceilpri;    /* システム予約領域 */
} T_RMTX;
```

[機能]

mtxid で指定されたミューテックスのミューテックス情報（獲得タスクの有無など）を *pk_rmtx* で指定された領域に格納します。

備考 ミューテックス情報についての詳細は，「13. 3. 6 ミューテックス情報 T_RMTX」を参照してください。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない

マクロ	数値	説明
E_PAR	-17	*pk_rmtx* が不正 - *pk_rmtx* = 0
E_ID	-18	*mtxid* が不正 - ミューテックス情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行

13. 4. 8　メモリ・プール管理機能

以下に，RI850MP がメモリ・プール管理機能として提供しているサービス・コールを示します。

表 13—18　メモリ・プール管理機能

サービス・コール名	機能概要
get_mpf	固定長メモリ・ブロックの獲得
pget_mpf ／ ipget_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 ／ irel_mpf	固定長メモリ・ブロックの返却
ref_mpf ／ iref_mpf	固定長メモリ・プール情報の参照

get_mpf

[概要]

固定長メモリ・ブロックの獲得

[記述形式]

```
ER  get_mpf(ID mpfid, VP *p_blk);
```

[引数]

I/O	引数	説明
I	ID *mpfid*;	固定長メモリ・プールの ID
O	VP *p_blk;	固定長メモリ・ブロックの先頭アドレスを格納する領域へのポインタ

[機能]

mpfid で指定された固定長メモリ・プールから固定長メモリ・ブロックを獲得し，該当ブロックの先頭アドレスを *p_blk* で指定された領域に格納します。

ただし，本サービス・コールを発行した際，対象固定長メモリ・プールから固定長メモリ・ブロックを獲得することができなかった（既に残りブロック数が 0 であった）場合には，自タスクを固定長メモリ・プールの待ちキューにキューイングしたのち，実行状態から固定長メモリ・ブロック獲得待ち状態へと遷移させます。

なお，固定長メモリ・ブロック獲得待ち状態の解除は，以下の場合に行われ，固定長メモリ・ブロック獲得待ち状態から実行可能状態へと遷移します。

固定長メモリ・ブロック獲得待ち状態の解除	戻り値
rel_mpf ／ irel_mpf の発行により，対象固定長メモリ・プールに固定長メモリ・ブロックが返却された	E_OK
rel_wai ／ irel_wai の発行により，固定長メモリ・ブロック獲得待ち状態を強制的に解除された	E_RLWAI

備考 1. タスクを固定長メモリ・プールの待ちキューにキューイングする順序は，属性 mpfatr で指定したキューイング方法（固定長メモリ・ブロックの獲得要求を行った順，またはタスクの優先度順）となります。

2. rel_wai ／ irel_wai の発行により，固定長メモリ・ブロック獲得待ち状態を強制的に解除された場合，*p_blk* で指定された領域の内容は不定値となります。

［戻り値］

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_PAR	-17	*p_blk* が不正 - *p_blk* = 0
E_ID	-18	*mpfid* が不正 - 固定長メモリ・プール情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - 非タスクから発行 - CPU ロック状態から発行 - ディスパッチ禁止状態から発行
E_RLWAI	-49	固定長メモリ・ブロック獲得待ち状態の強制解除 - rel_wai ／ irel_wai の発行

pget_mpf
ipget_mpf

[概要]

固定長メモリ・ブロックの獲得（ポーリング）

[記述形式]

```
ER  pget_mpf(ID mpfid, VP *p_blk);
ER  ipget_mpf(ID mpfid, VP *p_blk);
```

[引数]

I/O	引数	説明
I	ID *mpfid*;	固定長メモリ・プールの ID
O	VP *p_blk;	固定長メモリ・ブロックの先頭アドレスを格納する領域へのポインタ

[機能]

mpfid で指定された固定長メモリ・プールから固定長メモリ・ブロックを獲得し，該当ブロックの先頭アドレスを *p_blk* で指定された領域に格納します。

ただし，本サービス・コールを発行した際，対象固定長メモリ・プールから固定長メモリ・ブロックを獲得することができなかった（既に残りブロック数が 0 であった）場合には，戻り値として“E_TMOUT（= -50）”を返却します。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_PAR	-17	*p_blk* が不正 - *p_blk* = 0
E_ID	-18	*mpfid* が不正 - 固定長メモリ・プール情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行
E_TMOUT	-50	ポーリング失敗 - 対象固定長メモリ・プールから獲得可能な残りブロック数が 0

tget_mpf

[概要]

固定長メモリ・ブロックの獲得（タイムアウトあり）

[記述形式]

```
ER  tget_mpf(ID mpfid, VP *p_blk, TMO tmout);
```

[引数]

I/O	引数	説明
I	ID *mpfid*;	固定長メモリ・プールの ID
O	VP *p_blk;	固定長メモリ・ブロックの先頭アドレスを格納する領域へのポインタ
I	TMO *tmout*;	待ち時間（単位：ミリ秒） TMO_FEVR： 永久待ち TMO_POL： ポーリング 数値： 待ち時間

[機能]

mpfid で指定された固定長メモリ・プールから固定長メモリ・ブロックを獲得し，該当ブロックの先頭アドレスを *p_blk* で指定された領域に格納します。

ただし，本サービス・コールを発行した際，対象固定長メモリ・プールから固定長メモリ・ブロックを獲得することができなかった（既に残りブロック数が 0 であった）場合には，自タスクを固定長メモリ・プールの待ちキューにキューイングしたのち，実行状態から固定長メモリ・ブロック獲得待ち状態へと遷移させます。

なお，固定長メモリ・ブロック獲得待ち状態の解除は，以下の場合に行われ，固定長メモリ・ブロック獲得待ち状態から実行可能状態へと遷移します。

固定長メモリ・ブロック獲得待ち状態の解除	戻り値
rel_mpf ／ irel_mpf の発行により，対象固定長メモリ・プールに固定長メモリ・ブロックが返却された	E_OK
rel_wai ／ irel_wai の発行により，固定長メモリ・ブロック獲得待ち状態を強制的に解除された	E_RLWAI
tmout で指定された待ち時間が経過した	E_TMOUT

備考 1. タスクを固定長メモリ・プールの待ちキューにキューイングする順序は，属性 mpfatr で指定したキューイング方法（固定長メモリ・ブロックの獲得要求を行った順，またはタスクの優先度順）となります。

2. rel_wai ／ irel_wai の発行，および *tmout* で指定された待ち時間の経過により，固定長メモリ・ブロック獲得待ち状態を強制的に解除された場合，*p_blk* で指定された領域の内容は不定値となります。

3. *tmout* に TMO_FEVR を指定した場合は get_mpf と，TMO_POL を指定した場合は pget_mpf ／ ipget_mpf と同等の処理が行われます。

［戻り値］

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_PAR	-17	*p_blk*，*tmout* が不正 - *p_blk* = 0 - tmout ＜ TMO_FEVR
E_ID	-18	*mpfid* が不正 - 固定長メモリ・プール情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - 非タスクから発行 - CPU ロック状態から発行 - ディスパッチ禁止状態から発行
E_RLWAI	-49	固定長メモリ・ブロック獲得待ち状態の強制解除 - rel_wai ／ irel_wai の発行
E_TMOUT	-50	タイムアウト - *tmout* で指定された待ち時間が経過

rel_mpf
irel_mpf

[概要]

固定長メモリ・ブロックの返却

[記述形式]

```
ER  rel_mpf(ID mpfid, VP blk);
ER  irel_mpf(ID mpfid, VP blk);
```

[引数]

I/O	引数	説明
I	ID *mpfid*;	固定長メモリ・プールの ID
I	VP *blk*;	固定長メモリ・ブロックの先頭アドレス

[機能]

mpfid で指定された固定長メモリ・プールに *blk* で指定された固定長メモリ・ブロックを返却します。

ただし，本サービス・コールを発行した際，対象固定長メモリ・プールの待ちキューにタスクがキューイングされていた場合には，ブロックの返却操作は行わず，該当タスク（待ちキューの先頭タスク）に固定長メモリ・ブロックを渡します。これにより，該当タスクは待ちキューから外れ，固定長メモリ・ブロック獲得待ち状態から実行可能状態へ，または二重待ち状態から強制待ち状態へと遷移します。

備考 1. 本サービス・コールでは，固定長メモリ・ブロックを返却する際，クリア操作を行っていません。したがって，返却された固定長メモリ・ブロックの内容は不定値となります。

2. 固定長メモリ・ブロックを獲得した固定長メモリ・プールとは異なる固定長メモリ・プールに返却操作を行った場合，以後の動作が保証されません。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_PAR	-17	*blk* が不正 - *blk* = 0

マクロ	数値	説明
E_ID	-18	*mpfid* が不正 - 固定長メモリ・プール情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行

ref_mpf
iref_mpf

[概要]

固定長メモリ・プール情報の参照

[記述形式]

```
ER  ref_mpf(ID mpfid, T_RMFP *pk_rmpf);
ER  iref_mpf(ID mpfid, T_RMPF *pk_rmpf);
```

[引数]

I/O	引数	説明
I	ID mpfid;	固定長メモリ・プールの ID
O	T_RMPF *pk_rmpf;	固定長メモリ・プール情報を格納する領域へのポインタ

【固定長メモリ・プール情報 T_RMPF】

```
typedef struct  t_rmpf {
        ID      wtskid;     /* 待ちタスクの有無 */
        UINT    fblkcnt;    /* 獲得可能な残りブロック数 */
        ATR     mpfatr;     /* 属性 */
} T_RMPF;
```

[機能]

mpfid で指定された固定長メモリ・プールの固定長メモリ・プール情報（待ちタスクの有無など）を *pk_rmpf* で指定された領域に格納します。

備考 固定長メモリ・プール情報についての詳細は，「13. 3. 7 固定長メモリ・プール情報 T_RMPF」を参照してください。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない

マクロ	数値	説明
E_PAR	-17	*pk_rmpf* が不正 - *pk_rmpf* = 0
E_ID	-18	*mpfid* が不正 - 固定長メモリ・プール情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行

13. 4. 9　時間管理機能

以下に，RI850MP が時間管理機能として提供しているサービス・コールを示します。

表 13—19　時間管理機能

サービス・コール名	機能概要
set_tim ／ iset_tim	システム時刻の変更
get_tim ／ iget_tim	システム時刻の参照
sta_cyc ／ ista_cyc	周期ハンドラの起動
stp_cyc ／ istp_cyc	周期ハンドラの終了
ref_cyc ／ iref_cyc	周期ハンドラ情報の参照

set_tim
iset_tim

[概要]

システム時刻の変更

[記述形式]

```
ER  set_tim(SYSTEM *p_systim);
ER  iset_tim(SYSTEM *p_systim);
```

[引数]

I/O	引数	説明
I	SYSTIM *p_systim;	システム時刻（単位：ミリ秒）を格納した領域へのポインタ

【システム時刻 SYSTIM】

```
typedef struct  t_systim {
        UW      ltime;    /* システム時刻（下位 32 ビット）*/
        UH      utime;    /* システム時刻（上位 16 ビット）*/
} SYSTIM;
```

[機能]

システム時刻（単位：ミリ秒）を *p_systim* で指定された時刻に変更します。

備考 1. 本サービス・コールの発行に伴い，tslp_tsk，dly_tsk，twai_sem などの待ち時間，および周期ハンドラの起動位相，起動周期に影響を与えることはありません。

2. システム時刻についての詳細は，「13. 3. 11 システム時刻 SYSTIM」を参照してください。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_PAR	-17	*p_systim* が不正 - *p_systim* = 0

マクロ	数値	説明
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行

get_tim
iget_tim

[概要]

システム時刻の参照

[記述形式]

```
ER  get_tim(SYSTIM *p_systim);
ER  iget_tim(SYSTIM *p_systim);
```

[引数]

I/O	引数	説明
O	SYSTIM *p_systim;	システム時刻を格納する領域へのポインタ

【システム時刻 SYSTIM】

```
typedef struct  t_systim {
        UW      ltime;    /* システム時刻（下位 32 ビット）*/
        UH      utime;    /* システム時刻（上位 16 ビット）*/
} SYSTIM;
```

[機能]

システム時刻を獲得し，*p_systim* で指定された領域に格納します。

備考 システム時刻についての詳細は，「13. 3. 11 システム時刻 SYSTIM」を参照してください。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_PAR	-17	*p_systim* が不正 - *p_systim* = 0
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行

sta_cyc
ista_cyc

［概要］

周期ハンドラの起動

［記述形式］

```
ER  sta_cyc(ID cycid);
ER  ista_cyc(ID cycid);
```

［引数］

I/O	引数	説明
I	ID *cycid*;	周期ハンドラの ID

［機能］

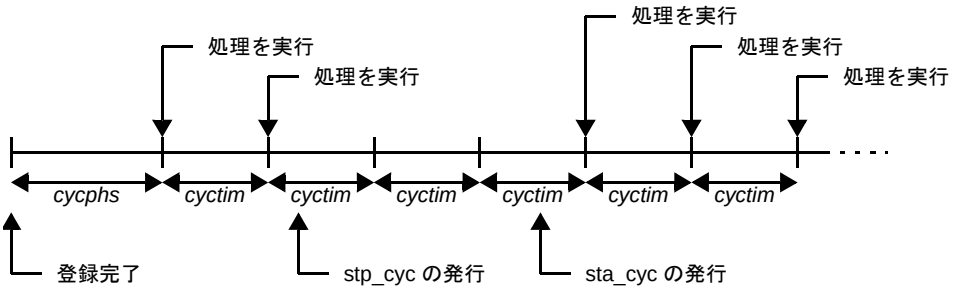
cycid で指定された周期ハンドラに起動要求を発行し，停止状態から動作状態へと遷移させます。

ただし，本サービス・コールの発行から対象周期ハンドラが 1 回目の処理を実行するまでの間隔は，対象周期ハンドラに TA_PHS 属性（保存有無：起動位相を保存）が付与されているか否かにより異なります。

- TA_PHS 属性が付与されている場合

  カーネル初期化部における該当周期ハンドラの登録完了を基準点とした初期起動位相 cycphs，起動周期 cyctim を遵守したタイミングで該当周期ハンドラの処理を実行します。

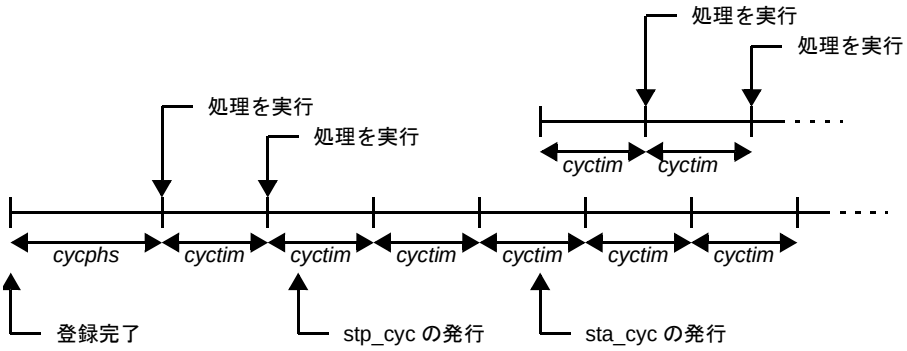
  以下に，該当周期ハンドラに TA_STA 属性（初期状態：動作状態）が付与されていた場合における処理の実行タイミングを示します。



- TA_PHS 属性が付与されていない場合

  本サービス・コールの発行を基準点とした起動周期 cyctim のタイミングで該当周期ハンドラの処理を実行します。

以下に，該当周期ハンドラに TA_STA 属性（初期状態：動作状態）が付与されていた場合における処理の実行タイミングを示します。



備考　本サービス・コールでは，起動要求のキューイングが行われません。このため，対象周期ハンドラが停止状態以外の場合には，何も操作は行わず，エラーとしても扱いません。

［戻り値］

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_ID	-18	*cycid* が不正 - 周期ハンドラ情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行

stp_cyc
istp_cyc

[概要]

周期ハンドラの終了

[記述形式]

```
ER  stp_cyc(ID cycid);
ER  istp_cyc(ID cycid);
```

[引数]

I/O	引数	説明
I	ID *cycid*;	周期ハンドラの ID

[機能]

cycid で指定された周期ハンドラに終了要求を発行し，動作状態から停止状態へと遷移させます。

備考 本サービス・コールでは，終了要求のキューイングが行われません。このため，対象周期ハンドラが動作状態以外の場合には，何も操作は行わず，エラーとしても扱いません。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_ID	-18	*cycid* が不正 - 周期ハンドラ情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行

ref_cyc
iref_cyc

[概要]

周期ハンドラ情報の参照

[記述形式]

```
ER  ref_cyc(ID cycid, T_RCYC *pk_rcyc);
ER  iref_cyc(ID cycid, T_RCYC *pk_rcyc);
```

[引数]

I/O	引数	説明
I	ID cycid;	周期ハンドラの ID
O	T_RCYC *pk_rcyc;	周期ハンドラ情報を格納する領域へのポインタ

【周期ハンドラ情報 T_RCYC】

```
typedef struct  t_rcyc {
        STAT    cycstat;    /* 現在状態 */
        RELTIM  lefttim;    /* 残り時間 */
        ATR     cycatr;     /* 属性 */
        RELTIM  cyctim;     /* 起動周期 */
        RELTIM  cycphs;     /* 起動位相 */
        PE_ID   peid;       /*PE 番号 */
} T_RCYC;
```

[機能]

cycid で指定された周期ハンドラの周期ハンドラ情報（現在状態など）を *pk_rcyc* で指定された領域に格納します。

備考 周期ハンドラ情報についての詳細は，「13. 3. 8 周期ハンドラ情報 T_RCYC」を参照してください。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了

マクロ	数値	説明
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_PAR	-17	*pk_rcyc* が不正 - *pk_rcyc* = 0
E_ID	-18	*cycid* が不正 - 周期ハンドラ情報で未定義の ID
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行

13. 4. 10　システム状態管理機能

以下に，RI850MP がシステム状態管理機能として提供しているサービス・コールを示します。

表 13—20　システム状態管理機能

サービス・コール名	機能概要
rot_rdq ／ irot_rdq	優先順位の回転
get_tid ／ iget_tid	ID の参照
loc_cpu ／ iloc_cpu	CPU ロック状態への移行
unl_cpu ／ iunl_cpu	CPU ロック解除状態への移行
dis_dsp	ディスパッチ禁止状態への移行
ena_dsp	ディスパッチ許可状態への移行
sns_ctx	コンテキスト種別（非タスク・コンテキスト，タスク・コンテキスト）の参照
sns_loc	システム状態種別（CPU ロック状態，CPU ロック解除状態）の参照
sns_dsp	システム状態種別（ディスパッチ禁止状態，ディスパッチ許可状態）の参照
sns_dpn	システム状態種別（ディスパッチ保留状態，非ディスパッチ保留状態）の参照

rot_rdq
irot_rdq

［概要］

優先順位の回転

［記述形式］

```
ER  rot_rdq(PRI tskpri);
ER  irot_rdq(PRI tskpri);
```

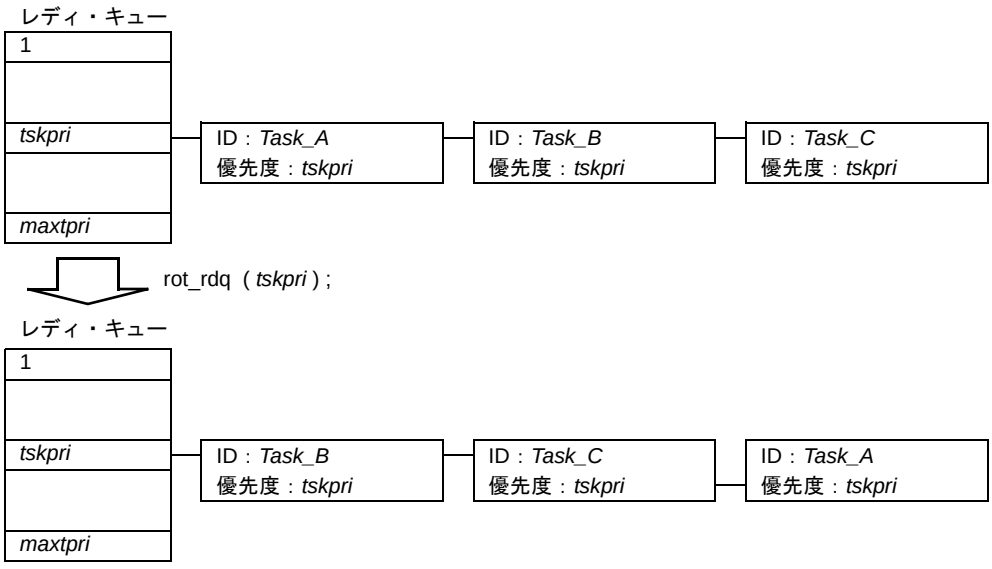
［引数］

I/O	引数	説明
I	PRI *tskpri*;	タスクの優先度 TPRI_SELF：　自タスクの現在優先度 数値：　　タスクの優先度

［機能］

レディ・キューに回転要求を発行し，タスクの実行順序を明示的に変更します。これにより，*tskpri* で指定された優先度に対応したレディ・キューの先頭にキューイングされているタスクは，該当優先度の末尾へとつなぎ替えられます。

備考 1.　以下に，本サービス・コールの発行に伴う，レディ・キューの状態変更イメージを示します。



2. 本サービス・コールでは，回転要求のキューイングが行われません。このため，対象優先度に対応したレディ・キューにタスクが 1 個もキューイングされていなかった場合には，何も操作は行わず，エラーとしても扱いません。

［戻り値］

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_PAR	-17	*tskpri* が不正 - *tskpri* ＜ 0 - *tskpri* ＞ 最大優先度 maxtpri - 非タスクからの発行時に TPRI_SELF を指定
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行

get_tid
iget_tid

[概要]

ID の参照

[記述形式]

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

[引数]

I/O	引数	説明
O	ID *p_tskid;	ID を格納する領域へのポインタ

[機能]

実行状態へと遷移しているタスクの ID を獲得し，*p_tskid* で指定された領域に格納します。

備考 実行状態へと遷移しているタスクが存在しなかった場合には，*p_tskid* で指定された領域に TSK_NONE（= 0）が格納されます。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_PAR	-17	*p_tskid* が不正 - *p_tskid* = 0
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行

loc_cpu
iloc_cpu

[概要]

CPU ロック状態への移行

[記述形式]

```
ER  loc_cpu(void);
ER  iloc_cpu(void);
```

[引数]

なし

[機能]

システム状態種別を CPU ロック解除状態から CPU ロック状態へと移行します。

これにより，本サービス・コールの発行から unl_cpu ／ iunl_cpu が発行されるまでの間，ディスパッチャ“タスクの切り替え処理”の実行が抑制された状態，および全マスカブル割り込みの受け付けが禁止された状態となります。

備考 1. CPU ロック状態（本サービス・コールの発行により移行）とは，以下の状態を意味しています。

- ディスパッチャの実行が抑制された状態
- マスカブル割り込みの受け付けが禁止された状態

2. 本サービス・コールでは，ロック要求のキューイングが行われません。このため，本サービス・コールを発行した際，システム状態種別を変更することができなかった（既にシステム状態種別が CPU ロック状態であった）場合には，何も操作は行わず，エラーとしても扱いません。

3. CPU ロック状態でマスカブル割り込みが発生した場合，対応する割り込みハンドラへの移行は unl_cpu ／ iunl_cpu が発行されるまで遅延されます。

4. RI850MP では，マスカブル割り込みの 1 種であるタイマ割り込みを利用して時間管理機能（システム時刻の更新，タスクのタイムアウト，周期ハンドラの起動など）を実現しています。このため，CPU ロック状態が基本クロック周期 tbase で指定された間隔よりも長い時間持続する場合には，時間管理機能が正常に動作しなくなる場合があります。

5. RI850MP では，CPU ロック状態で発行可能なサービス・コールを以下の 8 種類に限定しています。

loc_cpu，iloc_cpu，unl_cpu，iunl_cpu，sns_ctx，sns_loc，sns_dsp，sns_dpn

6. 本サービス・コールの内部処理（割り込みマスクの論理和：_kernel_usr_msk_intmsk，割り込みマスクの参照：_kernel_usr_get_intmsk）は，ユーザの実行環境に依存しています。このため，RI850MP では，該当処理をユーザ・オウン・コーディング部として切り出し，サンプル・ソース・ファイルを提供しています。な

お，“_kernel_usr_msk_intmsk”，および“_kernel_usr_get_intmsk”についての詳細は，「9. 2. 1　割り込みマスクの論理和」，「9. 2. 2　割り込みマスクの参照」を参照してください。

［戻り値］

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない

unl_cpu
iunl_cpu

[概要]

CPU ロック解除状態への移行

[記述形式]

```
ER  unl_cpu(void);
ER  iunl_cpu(void);
```

[引数]

なし

[機能]

システム状態種別を CPU ロック状態から CPU ロック解除状態へと移行します。

これにより，loc_cpu ／ iloc_cpu の発行により抑制されていたディスパッチャ“タスクの切り替え処理”の実行，および禁止されていたマスカブル割り込みの受け付けが許可されます。

備考 1. 本サービス・コールでは，ディスパッチ禁止状態の解除は行いません。

2. 本サービス・コールでは，解除要求のキューイングが行われません。このため，本サービス・コールを発行した際，システム状態種別を変更することができなかった（既にシステム状態種別が CPU ロック解除状態であった）場合には，何も操作は行わず，エラーとしても扱いません。

3. 本サービス・コールの内部処理（割り込みマスクの上書き：_kernel_usr_set_intmsk）は，ユーザの実行環境に依存しています。このため，RI850MP では，該当処理をユーザ・オウン・コーディング部として切り出し，サンプル・ソース・ファイルを提供しています。なお，“_kernel_usr_set_intmsk”についての詳細は，「9. 2. 3　割り込みマスクの上書き」を参照してください。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない

dis_dsp

[概要]

ディスパッチ禁止状態への移行

[記述形式]

```
ER  dis_dsp(void);
```

[引数]

なし

[機能]

システム状態種別をディスパッチ許可状態からディスパッチ禁止状態へと移行します。

これにより，本サービス・コールの発行から ena_dsp が発行されるまでの間，ディスパッチャ“タスクの切り替え処理”の実行が抑制された状態となります。

備考 1. ディスパッチ禁止状態（本サービス・コールの発行により移行）とは，以下の状態を意味しています。

- ディスパッチャの実行が抑制された状態

2. 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，本サービス・コールを発行した際，システム状態種別を変更することができなかった（既にシステム状態種別がディスパッチ禁止状態であった）場合には，何も操作は行わず，エラーとしても扱いません。

3. ディスパッチ禁止状態で発行されたサービス・コール（act_tsk，chg_pri，wup_tsk など）では，キュー操作，カウンタ操作などといった処理のみが行われ，実際の“タスクの切り替え処理”は，ena_dsp が発行されるまで遅延され，一括処理されます。

4. RI850MP では，ディスパッチ禁止状態において，“自タスクの状態を遷移させる可能性があるサービス・コール（slp_tsk，wai_sem，wai_flg など）”が発行された場合，要求条件の即時成立／不成立を問わず，戻り値として“E_CTX（= -25）”を返却します。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない

マクロ	数値	説明
E_CTX	-25	コンテキスト・エラー - 非タスクから発行 - CPU ロック状態から発行

ena_dsp

［概要］

ディスパッチ許可状態への移行

［記述形式］

```
ER  ena_dsp(void);
```

［引数］

なし

［機能］

システム状態種別をディスパッチ禁止状態からディスパッチ許可状態へと移行します。
これにより，dis_dsp の発行により抑制されていたディスパッチャ“タスクの切り替え処理”の実行が許可されます。

備考　本サービス・コールでは，許可要求のキューイングが行われません。このため，本サービス・コールを発行した際，システム状態種別を変更することができなかった（既にシステム状態種別がディスパッチ許可状態であった）場合には，何も操作は行わず，エラーとしても扱いません。

［戻り値］

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_CTX	-25	コンテキスト・エラー - 非タスクから発行 - CPU ロック状態から発行

sns_ctx

［概要］

コンテキスト種別（非タスク・コンテキスト，タスク・コンテキスト）の参照

［記述形式］

```
BOOL  sns_ctx(void);
```

［引数］

なし

［機能］

本サービス・コールを発行した処理プログラムのコンテキスト種別（非タスク・コンテキスト，タスク・コンテキスト）を獲得し，戻り値として返却します。

備考　本サービス・コールにおける非タスク・コンテキストの処理プログラムとは，以下の処理プログラムを意味しています。

- 周期ハンドラ
- 割り込みハンドラ
- CPU 例外ハンドラ
- 初期化ルーチン

［戻り値］

マクロ	数値	説明
TRUE	1	正常終了 - 非タスク・コンテキスト
FALSE	0	正常終了 - タスク・コンテキスト
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない

sns_loc

［概要］

システム状態種別（CPU ロック状態，CPU ロック解除状態）の参照

［記述形式］

```
BOOL  sns_loc(void);
```

［引数］

なし

［機能］

本サービス・コールを発行した時点におけるシステム状態種別（CPU ロック状態，CPU ロック解除状態）を獲得し，戻り値として返却します。

備考　CPU ロック状態（loc_cpu ／ iloc_cpu の発行により移行）とは，以下の状態を意味しています。

- ディスパッチャの実行が抑制された状態
- マスカブル割り込みの受け付けが禁止された状態

［戻り値］

マクロ	数値	説明
TRUE	1	正常終了 - CPU ロック状態
FALSE	0	正常終了 - CPU ロック解除状態
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない

sns_dsp

［概要］

システム状態種別（ディスパッチ禁止状態，ディスパッチ許可状態）の参照

［記述形式］

```
BOOL  sns_dsp(void);
```

［引数］

なし

［機能］

本サービス・コールを発行した時点におけるシステム状態種別（ディスパッチ禁止状態，ディスパッチ許可状態）を獲得し，戻り値として返却します。

備考　ディスパッチ禁止状態（dis_dsp の発行により移行）とは，以下の状態を意味しています。

- ディスパッチャの実行が抑制された状態

［戻り値］

マクロ	数値	説明
TRUE	1	正常終了 - ディスパッチ禁止状態
FALSE	0	正常終了 - ディスパッチ許可状態
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない

sns_dpn

［概要］

システム状態種別（ディスパッチ保留状態，非ディスパッチ保留状態）の参照

［記述形式］

```
BOOL  sns_dpn(void);
```

［引数］

なし

［機能］

本サービス・コールを発行した時点におけるシステム状態種別（ディスパッチ保留状態，非ディスパッチ保留状態）を獲得し，戻り値として返却します。

備考　ディスパッチ保留状態とは，以下の状態を意味しています。

- ディスパッチャよりも優先度の高い処理が実行している状態
- ディスパッチャの実行が抑制された状態
- マスカブル割り込みの受け付けが禁止された状態

［戻り値］

マクロ	数値	説明
TRUE	1	正常終了 - ディスパッチ保留状態
FALSE	0	正常終了 - 非ディスパッチ保留状態
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない

13. 4. 11　割り込み管理機能

以下に，RI850MP が割り込み管理機能として提供しているサービス・コールを示します。

表 13—21　割り込み管理機能

サービス・コール名	機能概要
dis_int	マスカブル割り込みの受け付け禁止
ena_int	マスカブル割り込みの受け付け許可
chg_ipm ／ ichg_ipm	プライオリティ・マスク・レジスタの変更
get_ipm ／ iget_ipm	プライオリティ・マスク・レジスタの参照

dis_int

［概要］

マスカブル割り込みの受け付け禁止

［記述形式］

```
ER  dis_int(INTNO intno);
```

［引数］

I/O	引数	説明
I	INTNO *intno*;	例外要因コード

［機能］

intno で指定された例外要因コードに対応したマスカブル割り込みの受け付けを禁止します。

これにより，本サービス・コールの発行から ena_int が発行されるまでの間，該当マスカブル割り込みの受け付けが禁止された状態となります。

備考 1. 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，本サービス・コールを発行した際，マスカブル割り込みの受け付け状態を変更することができなかった（既にマスカブル割り込みの受け付けが禁止されていた）場合には，何も操作は行わず，エラーとしても扱いません。

2. マスカブル割り込みの受け付けが禁止された状態でマスカブル割り込みが発生した場合，対応する割り込みハンドラへの移行は ena_int が発行されるまで遅延されます。

3. RI850MP では，マスカブル割り込みの 1 種であるタイマ割り込みを利用して時間管理機能（システム時刻の更新，タスクのタイムアウト，周期ハンドラの起動など）を実現しています。このため，タイマ割り込みの受け付けが禁止された状態が基本クロック周期 tbase で指定された間隔よりも長い時間持続する場合には，時間管理機能が正常に動作しなくなる場合があります。

4. 本サービス・コールの内部処理（マスカブル割り込みの受け付け禁止：_kernel_usr_dis_int）は，ユーザの実行環境に依存しています。このため，RI850MP では，該当処理をユーザ・オウン・コーディング部として切り出し，サンプル・ソース・ファイルを提供しています。なお，“_kernel_usr_dis_int”についての詳細は，「9. 2. 4　マスカブル割り込みの受け付け禁止」を参照してください。

［戻り値］

マクロ	数値	説明
E_OK	0	正常終了

マクロ	数値	説明
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_PAR	-17	*intno* が不正 - 0x0 ≦ *intno* ≦ 0x70 - 対象デバイスが未サポートの例外要因コード
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行

ena_int

［概要］

マスカブル割り込みの受け付け許可

［記述形式］

```
ER  ena_int(INTNO intno);
```

［引数］

I/O	引数	説明
I	INTNO *intno*;	例外要因コード

［機能］

intno で指定された例外要因コードに対応したマスカブル割り込みの受け付けを許可します。

備考 1. 本サービス・コールでは，許可要求のキューイングが行われません。このため，本サービス・コールを発行した際，マスカブル割り込みの受け付け状態を変更することができなかった（既にマスカブル割り込みの受け付けが許可されていた）場合には，何も操作は行わず，エラーとしても扱いません。

2. 本サービス・コールの内部処理（マスカブル割り込みの受け付け許可：_kernel_usr_ena_int）は，ユーザの実行環境に依存しています。このため，RI850MP では，該当処理をユーザ・オウン・コーディング部として切り出し，サンプル・ソース・ファイルを提供しています。なお，“_kernel_usr_ena_int”についての詳細は，「9. 2. 5　マスカブル割り込みの受け付け許可」を参照してください。

［戻り値］

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_PAR	-17	*intno* が不正 - 0x0 ≦ *intno* ≦ 0x70 - 対象デバイスが未サポートの例外要因コード
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行

chg_ipm
ichg_ipm

[概要]

プライオリティ・マスク・レジスタの変更

[記述形式]

```
ER  chg_ipm(INTPMR ipmptn);
ER  ichg_ipm(INTPMR ipmptn);
```

[引数]

I/O	引数	説明
I	INTPMT *ipmptn*;	変更後のレジスタ値

[機能]

プライオリティ・マスク・レジスタ（PMR：Priority Mask Register）のレジスタ値を *ipmptn* で指定された値に変更します。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行

get_ipm
iget_ipm

[概要]

プライオリティ・マスク・レジスタの参照

[記述形式]

```
ER  get_ipm(INTPMR *p_ipmptn);
ER  iget_ipm(INTPMR *p_ipmptn);
```

[引数]

I/O	引数	説明
O	INTPMR *p_ipmptn;	レジスタ値を格納する領域へのポインタ

[機能]

プライオリティ・マスク・レジスタ（PMR：Priority Mask Register）のレジスタ値を獲得し，*p_ipmptn* で指定された領域に格納します。

[戻り値]

マクロ	数値	説明
E_OK	0	正常終了
E_NOSPT	-9	未サポートの機能 - SCT 情報で本サービス・コールを使用する旨の定義が行われていない
E_PAR	-17	*p_ipmptn* が不正 - *p_ipmptn* = 0
E_CTX	-25	コンテキスト・エラー - CPU ロック状態から発行

付録 A　コンフィギュレータ

本付録では，コンフィギュレータについて解説しています。

A. 1　概　　要

コンフィギュレータは，コンフィギュレーション・ファイルを入力ファイルとして読み込んだのち，RI850MP に提供するコンフィギュレーション情報を保持したファイル（システム情報テーブル・ファイル，エントリ・ファイル，システム情報ヘッダ・ファイル，サービス・コール・テーブル・ファイル）を出力するユーティリティ・ツールです。

以下に，コンフィギュレータが出力する情報ファイルについて示します。

- システム情報テーブル・ファイル

  RI850MP が動作する上で必要となるデータを保持した情報ファイルです。

- エントリ・ファイル

  タイマ割り込み，割り込み，CPU 例外が発生した際，対象デバイスが強制的に制御を移すハンドラ・アドレスに対して該当処理（時間管理機能，割り込みハンドラ，CPU 例外ハンドラ）への分岐処理が割り付けられた情報ファイルです。

- システム情報ヘッダ・ファイル

  オブジェクト（タスク，セマフォ，イベントフラグなど）の名前と ID の対応付けを保持した情報ファイルです。

- サービス・コール・テーブル・ファイル

  RI850MP が提供しているサービス・コールの使用有無に関する情報を保持した情報ファイルです。

備考　コンフィギュレータの起動には，“.NET Framework 2.0”が必要となります。

A. 2　起動方法

A. 2. 1　コマンド・ラインからの起動

以下に，コンフィギュレータをコマンド・ラインから起動する際の方法を示します。

なお，“C:\>”はコマンド・プロンプトを，“△”は Space キーの入力を，“<Enter>”は Enter キーの入力を，“[　]”で囲まれた部分は省略可能な起動オプションである旨を示しています。

```
C:\> cf850mp.exe [△@<cmd_file>] △-cpu△<name> [△-devpath=<path>] [△-i△<sitfile>] [△-e△<entryfile>] [△-d△<includefile>] [△-c△<sctfile>] [△-ni] [△-ne] [△-nd] [△-nc] [△-t△<tool>] [△-T△<compiler_path>] [△-I△<include_path>] [△-np] [△-cnv△<cnvfile>] [△-V] [△-help] [△<cffile>] <Enter>
```

以下に，各起動オプションの詳細を示します。

起動オプション	意味
@<*cmd_file*>	コンフィギュレータへの入力ファイル名（コマンド・ファイル名）を指定します。 - 本起動オプションの指定を省略した場合，コマンド・ファイルの読み込みは行われません。 - コマンド・ファイルについての詳細は，「A. 3 コマンド・ファイル」を参照してください。
-cpu △ <*name*>	対象デバイスの品種指定名を指定します。 なお，<*name*> として指定可能なキーワードは，デバイス・ファイル名から先頭文字“D”，および拡張子“.800”を除いた文字列となります。 - デバイス・ファイル名が DF3507.800 の場合，<*name*> に指定するキーワードは F3507 となります。
-i △ <*sitfile*>	コンフィギュレータからの出力ファイル名（システム情報テーブル・ファイル名）を指定します。 - 本起動オプションの指定を省略した場合，-i △ sit.c が指定されていたものとして処理が行われます。
-e △ <*entryfile*>	コンフィギュレータからの出力ファイル名（エントリ・ファイル名）を指定します。 - 本起動オプションの指定を省略した場合，CX 対応版の場合は -e △ entry.s が，CCV850E 対応版の場合は -e △ entry.850 が指定されていたものとして処理が行われます。
-d △ <*includefile*>	コンフィギュレータからの出力ファイル名（システム情報ヘッダ・ファイル名）を指定します。 - 本起動オプションの指定を省略した場合，-d △ kernel_id.h が指定されていたものとして処理が行われます。
-c △ <*sctfile*>	コンフィギュレータからの出力ファイル名（サービス・コール・テーブル・ファイル名）を指定します。 - 本起動オプションの指定を省略した場合，-c △ sct.c が指定されていたものとして処理が行われます。
-ni	システム情報テーブルファイルの出力を抑制します。 - 本起動オプションと -i △ <*sitfile*> が同時に指定された場合，コンフィギュレータは本起動オプションを有効オプションとして扱います。
-ne	エントリ・ファイルの出力を抑制します - 本起動オプションと -e △ <*entryfile*> が同時に指定された場合，コンフィギュレータは本起動オプションを有効オプションとして扱います。
-nd	システム情報ヘッダ・・ファイルの出力を抑制します。 - 本起動オプションと -d △ <*includefile*> が同時に指定された場合，コンフィギュレータは本起動オプションを有効オプションとして扱います。
-nc	サービス・コール・テーブル・ファイルの出力を抑制します。 - 本起動オプションと -c △ <*sctfile*> が同時に指定された場合，コンフィギュレータは本起動オプションを有効オプションとして扱います。
-t △ <*tool*>	ユーザが使用する C コンパイラ・パッケージの種類をします。 なお，<*tool*> として指定可能なキーワードは，CX，または CCV850E に限られます。 - 本起動オプションの指定を省略した場合，-t △ CX が指定されていたものとして処理が行われます。

起動オプション	意味
-T △ <*compiler_path*>	-t △ <*tool*> で指定された C コンパイラ・パッケージの C プリプロセッサを <*compiler_path*> フォルダから検索します。 - 本起動オプションの指定を省略した場合，カレント・フォルダ，および Windows の環境変数（PATH など）で指定されているフォルダに対する検索のみが行われます。
-I △ <*include_path*>	ヘッダ・ファイル情報で定義されているヘッダ・ファイルを <*include_path*> フォルダから検索します。 - 本起動オプションの指定を省略した場合，-t △ <*tool*> で指定された C コンパイラ・パッケージのデフォルト検索フォルダに対する検索のみが行われます。
-np	C プリプロセッサの起動を抑制します。 - 本起動オプションが指定された場合，コンフィギュレーション・ファイルの行頭に“#”が記述されている行については，行末までがコメントとして扱われます。
-cnv △ <*cnvfile*>	<*cffile*> で指定されたコンフィギュレーションを RI850MP 用コンフィギュレーション・ファイルとして出力します。
-V	コンフィギュレータのバージョン情報を標準出力に出力します。
-help	コンフィギュレータの起動オプションに関する情報（種類，用途など）を標準出力に出力します。
<*cffile*>	コンフィギュレータへの入力ファイル名（コンフィギュレーション・ファイル名）を指定します。 - 本起動オプションの指定を省略した場合，コンフィギュレーション・ファイルの読み込みは行われません。 なお，本起動オプションの省略は，-V，または -help の指定時に限定されています。 - コンフィギュレーション・ファイルについての詳細は，「付録 B　コンフィギュレーション・ファイル」を参照してください。

A. 2. 2　CubeSuite+ からの起動

プロパティ パネルの［システム・コンフィギュレーション・ファイル関連情報］タブで設定した内容に基づき，CubeSuite+ のビルド時に起動されます。

A. 3　コマンド・ファイル

コンフィギュレータでは，コマンド・ライン上で指定可能な起動オプションの文字数制限を解消する目的からコマンド・ファイル対応を行っています。

以下に，コマンド・ファイルの記述形式を示します。

- 文字コード

  ASCII コードでの記述となります。

  なお，日本語については，Shift-JIS コード，EUC-JP コード，UTF-8 コードの文字列をコメント内でのみ記述することができます。

- コメント

  行頭に“#”を記述した場合，行末までがコメントとして扱われます。

付録 B　コンフィギュレーション・ファイル

本付録では，コンフィギュレーション・ファイルの記述方法について解説しています。

B. 1　概　　要

コンフィギュレーション・ファイルは，RI850MP に提供するコンフィギュレーション情報を保持したファイル（システム情報テーブル・ファイル，エントリ・ファイル，システム情報ヘッダ・ファイル，サービス・コール・テーブル・ファイル）を生成する際に必要となるファイルであり，ユーザがテキスト・エディタを利用して記述するものです。

以下に，コンフィギュレーション・ファイルを記述する際の表記方法を示します。

- 文字コード

ASCII コードでの記述となります。

なお，日本語については，Shift-JIS コード，EUC-JP コード，UTF-8 コードの文字列をコメント内でのみ記述することができます。

- コメント

コメントの開始宣言“/*”からコメントの終了宣言“*/”で囲まれた部分，および，コメントの開始宣言“//”から行末までがコメントとして扱われます。

備考　コンフィギュレータの起動オプションに“-np”が指定された場合，行頭の“#”から行末までもコメントとして扱われます。

- 数値

数字“0 ～ 9”で始まる単語が数値として扱われます。

なお，RI850MP では，数値を以下のように区別しています。

10 進数：　数字“1 ～ 9”で始まる単語

16 進数：　0x，または 0X で始まる単語

- 名前

英字“a ～ z，A ～ Z”，またはアンダーバー“_”で始まる単語が名前として扱われます。

なお，名前として指定可能な文字数は，255 文字以内に限られます。

備考 1.　“名前”と“シンボル名”の区別は，コンフィギュレーション・ファイルの文脈から判断されます。

2.　コンフィギュレーション・ファイルでは，英小文字“a ～ z”と英大文字“A ～ Z”の区別が行われます。

- シンボル名

英字“a ～ z，A ～ Z”，またはアンダーバー“_”で始まる単語がシンボル名として扱われます。

なお，シンボル名として指定可能な文字数は，4095 文字以内に限られます。

備考 1. “名前”と“シンボル名”の区別は，コンフィギュレーション・ファイルの文脈から判断されます。

2. コンフィギュレーション・ファイルでは，英小文字“a ～ z”と英大文字“A ～ Z”の区別が行われます。

3. シンボル名は，“シンボル名 + オフセット”といった形式での記述も可能ですが，オフセットに指定可能な値は，“定数式”に限られます。

- キーワード

以下に示した単語は，コンフィギュレーション・ファイル用のキーワードとして予約されているため，指定された用途以外で使用することは禁止されています。

ATT_INI，CLK_INTNO，CRE_CYC，CRE_DTQ，CRE_FLG，CRE_MBX，CRE_MPF，CRE_MTX，CRE_SEM，CRE_TSK，DEF_EXC，DEF_FPSR，DEF_INH，DEF_SCT，DEF_TIM，DOMAIN，DOMAIN_ALLOCATION，INCLUDE，MAX_PRI，MEM_AREA，NULL，RI850MP，RI_SERIES，SIZE_AUTO，SYS_STK，TA_ACT，TA_ASM，TA_CLR，TA_DISINT，TA_ENAINT，TA_HLNG，TA_MFIFO，TA_MPRI，TA_PHS，TA_STA，TA_TFIFO，TA_TPRI，TA_WMUL，TA_WSGL，V100 ～ V199，VATT_IDL，サービス・コール名

B. 1. 1　コンフィギュレーション情報

RI850MP に記述するコンフィギュレーション情報は，以下に示した 5 種類に大別されます。

- 宣言情報
- システム情報
- ドメイン情報
- 静的 API 情報
- SCT 情報

以下に，コンフィギュレーション・ファイルの記述イメージを示します。

図 B—1　コンフィギュレーション・ファイルの記述イメージ

```
// 宣言情報
INCLUDE("h_file");                  // ヘッダ・ファイル情報

// システム情報
RI_SERIES(osnam, osver);            // RI シリーズ情報
DEF_TIM(tbase);                     // 基本クロック周期情報
CLK_INTNO(tintno);                  // タイマ割り込み情報
SYS_STK(sysstksz, peno);            // システム・スタック情報
MAX_PRI(maxtpri);                   // 最大優先度情報
DEF_FPSR(fpsr);                     // 浮動小数点設定／状態レジスタ情報
MEM_AREA(secnam, secsz);            // セクション情報
DOMAIN_ALLOCATION(domnam, peno);    // プロセッサ・エレメント情報

// ドメイン情報
DOMAIN (domnam) {
    // 静的 API 情報
    CRE_TSK(tskid, {tskatr, exinf, task, itskpri, stksz[:secnam], stk});    // タスク情報
    CRE_SEM(semid, {sematr, isemcnt, maxsem});                  // セマフォ情報
    CRE_FLG(flgid, {flgatr, iflgptn});                          // イベントフラグ情報
    CRE_DTQ(dtqid, {dtqatr, dtqcnt[:secnam], dtq});             // データ・キュー情報
    CRE_MBX(mbxid, {mbxatr, maxmpri, mprihd});                  // メールボックス情報
    CRE_MTX(mtxid, {mtxatr, ceilpri});                          // ミューテックス情報
    CRE_MPF(mpfid, {mpfatr, blkcnt, blksz[:secnam], mpf});      // 固定長メモリ・プール情報
    CRE_CYC(cycid, {cycatr, exinf, cychdr, cyctim, cycphs});    // 周期ハンドラ情報
    DEF_INH(inhno, {inhatr, inthdr});                           // 割り込みハンドラ情報
    DEF_EXC(excno, {excatr, exchdr});                           // CPU 例外ハンドラ情報
    ATT_INI({iniatr, exinf, inirtn});                           // 初期化ルーチン情報
    VATT_IDL({idlatr, idlrtn});                                 // アイドル・ルーチン情報
}

// SCT 情報
DEF_SCT(svc_nam);
```

B. 2　宣言情報

宣言情報として，以下の情報を定義します。

- ヘッダ・ファイル情報

以下に，宣言情報の記述イメージを示します。

```
INCLUDE("h_file");                    // ヘッダ・ファイル情報
```

B. 2. 1　ヘッダ・ファイル情報

ヘッダ・ファイルに関する情報として，以下の項目を定義します。

- ヘッダ・ファイル名 h_file

なお，ヘッダ・ファイル情報の定義数に制限はありません。

以下に，ヘッダ・ファイル情報の記述形式を示します。

```
INCLUDE("h_file");
```

（1）ヘッダ・ファイル名 *h_file*

システム情報ヘッダ・ファイルに出力するヘッダ・ファイル名を指定します。

なお，*h_file* として指定可能な値は，名前に限られます。

備考 1. “INCLUDE (" <itron.h> ");”と指定した場合，システム情報ヘッダ・ファイルには，以下に示したヘッダ・ファイルの定義（インクルード処理）が出力されます。

```
#include    <itron.h>
```

2. “INCLUDE (" \"itron.h\" ");”と指定した場合，システム情報ヘッダ・ファイルには，以下に示したヘッダ・ファイルの定義（インクルード処理）が出力されます。

```
#include    "itron.h"
```

3. *h_file* の検索対象フォルダは，コンフィギュレータの起動オプション“-I △ <*include_path*>”で指定したフォルダとなります。

B. 3 システム情報

システム情報として，以下の情報を定義します。

- RI シリーズ情報
- 基本クロック周期情報
- タイマ割り込み情報
- システム・スタック情報
- 最大優先度情報
- 浮動小数点設定／状態レジスタ情報
- セクション情報
- プロセッサ・エレメント情報

以下に，システム情報の記述イメージを示します。

```
RI_SERIES(osnam, osver);              // RI シリーズ情報
DEF_TIM(tbase);                       // 基本クロック周期情報
CLK_INTNO(tintno);                    // タイマ割り込み情報
SYS_STK(sysstksz, peno);              // システム・スタック情報
MAX_PRI(maxtpri);                     // 最大優先度情報
DEF_FPSR(fpsr);                       // 浮動小数点設定／状態レジスタ情報
MEM_AREA(secnam, secsz);              //セクション情報
DOMAIN_ALLOCATION(domnam, peno);      // プロセッサ・エレメント情報
```

B. 3. 1 RI シリーズ情報

リアルタイム OS に関する情報として，以下の項目を定義します。

- リアルタイム OS 名 osnam
- バージョン番号 osver

なお，RI シリーズ情報として定義可能な数は，1 個に限られます。

以下に，RI シリーズ情報の記述形式を示します。

```
RI_SERIES(osnam, osver);
```

(1) リアルタイム OS 名 *osnam*

リアルタイム OS の名前を指定します。

なお，*osnam* として指定可能な名前は，RI850MP に限られます。

(2) バージョン番号 *osver*

リアルタイム OS のバージョン番号を指定します。

なお，*osver* として指定可能な値は，V100 ～ V199 に限られます。

B. 3. 2　基本クロック周期情報

RI850MP が提供している時間管理機能（システム時刻の更新，タスクのタイムアウト，周期ハンドラの起動など）を実現するうえで必要となるタイマ割り込みに関する情報として，以下の項目を定義します。

- 基本クロック周期 tbase

なお，基本クロック周期情報として定義可能な数は，0 ～ 1 個に限られます。

以下に，基本クロック周期情報の記述形式を示します。

```
DEF_TIM(tbase);
```

（1）基本クロック周期 *tbase*

タイマ割り込みの発生周期（基本クロック周期，単位：ミリ秒）を指定します。

なお，*tbase* として指定可能な値は，1 ～ 65535 に限られます。

備考　本項目の定義を省略した場合，以下の記述が行われていたものとして扱われます。

```
DEF_TIM(1);
```

B. 3. 3　タイマ割り込み情報

RI850MP が提供している時間管理機能（システム時刻の更新，タスクのタイムアウト，周期ハンドラの起動など）を実現するうえで必要となるタイマ割り込みに関する情報として，以下の項目を定義します。

- 例外要因コード tintno

なお，タイマ割り込み情報として定義可能な数は，1 個に限られます。

以下に，タイマ割り込み情報の記述形式を示します。

```
CLK_INTNO(tintno);
```

（1）例外要因コード *tintno*

タイマ割り込みに対応した例外要因コードを指定します。

なお，*tintno* として指定可能な値は，タイマ割り込みに対応した 0x90 以降の 16 バイト境界値，または例外要因名に限られます。

備考 1. 例外要因コード inhno，または例外要因コード excno で指定した例外要因コードを割り当てることはできません。

2. 本項目で指定可能な例外要因名は，デバイス・ファイルで規定されている例外要因名に限られます。

B. 3. 4　システム・スタック情報

RI850MP がプロセッサ・エレメント単位に割り付けるシステム・スタックに関する情報として，以下の項目を定義します。

- スタック・サイズ sysstksz
- PE 番号 peno

なお，システム・スタック情報として定義可能な数は，2 個に限られます。

以下に，システム・スタック情報の記述形式を示します。

```
SYS_STK(sysstksz, peno);
```

（1）スタック・サイズ *sysstksz*

システム・スタックのサイズ（単位：バイト）を指定します。

なお，*sysstksz* として指定可能な値は，0x0 ～ 0x10000000 の 4 バイト境界値に限られます。

（2）PE 番号 *peno*

システム・スタックを割り付けるプロセッサ・エレメントの PE 番号を指定します。

なお，*peno* として指定可能な値は，1 ～ 2 に限られます。

備考　システム・スタックは，.kernel_stack_pe*peno* セクションに割り付けられます。

B. 3. 5　最大優先度情報

タスクの優先度に関する情報として，以下の項目を定義します。

- 最大優先度 maxtpri

なお，最大優先度情報として定義可能な数は，0 ～ 1 個に限られます。

以下に，最大優先度情報の記述形式を示します。

```
MAX_PRI(maxtpri);
```

（1）最大優先度 *maxtpri*

初期優先度 itskpri，および chg_pri ／ ichg_pri で指定するタスクの優先度範囲を指定します。

なお，*maxtpri* として指定可能な値は，1 ～ 32 に限られます。

備考　本項目の定義を省略した場合，以下の記述が行われていたものとして扱われます。

```
MAX_PRI(32);
```

B. 3. 6　浮動小数点設定／状態レジスタ情報

浮動小数点設定／状態レジスタ（FPSR：Floating-point setting/status register）に関する情報として，以下の項目を定義します。

- 浮動小数点設定／状態レジスタ fpsr

なお，浮動小数点設定／状態レジスタ情報として定義可能な数は，0 ～ 1 個に限られます。

以下に，浮動小数点設定／状態レジスタ情報の記述形式を示します。

```
DEF_FPSR(fpsr);
```

（1）浮動小数点設定／状態レジスタ *fpsr*

浮動小数点設定／状態レジスタの初期値を指定します。

なお，*fpsr* として指定可能な値は，0x0 ～ 0xFFFFFFFF に限られます。

備考　本項目の定義を省略した場合，以下の記述が行われていたものとして扱われます。

```
DEF_FPSR(0x20000);
```

B. 3. 7 セクション情報

システム・スタック，タスク・スタック，データ・キューのバッファ領域，固定長メモリ・プールを割り付けるセクションに関する情報として，以下の項目を定義します。

- セクション名 secnam
- セクション・サイズ secsz

なお，セクション情報として定義可能な数は，0 ～ 255 個に限られます。

以下に，セクション情報の記述形式を示します。

```
MEM_AREA(secnam, secsz);
```

(1) セクション名 *secnam*

システム・スタック，タスク・スタック，データ・キューのバッファ領域，固定長メモリ・プールを割り付けるセクション名を指定します。

なお，*secnam* として指定可能な値は，リンク・ディレクティブ・ファイルで指定したセクション名に限られます。

(2) セクション・サイズ *secsz*

セクションのサイズ（単位：バイト）を指定します。

なお，*secsz* として指定可能な値は，0x0 ～ 0x10000000 の 4 バイト境界値，または SIZE_AUTO に限られます。

備考 本項目に SIZE_AUTO が指定された場合，システム・スタック情報，タスク情報，データ・キュー情報，固定長メモリ・プール情報で指定されたサイズから適切な値が算出され，該当値が指定されていたものとして扱われます。

備考 本項目の定義を省略した場合，以下の記述が行われていたものとして扱われます。

```
MEM_AREA(.kernel_work_pe1, SIZE_AUTO);
MEM_AREA(.kernel_work_pe2, SIZE_AUTO);
```

B. 3. 8　プロセッサ・エレメント情報

プロセッサ・エレメント（PE：Processor Element）に関する情報として，以下の項目を定義します。

- ドメイン名 domnam
- PE 番号 peno

なお，プロセッサ・エレメント情報として定義可能な数は，1 ～ 31 個に限られます。

以下に，プロセッサ・エレメント情報の記述形式を示します。

```
DOMAIN_ALLOCATION(domnam, peno);
```

（1）ドメイン名 *domnam*

ドメインの名前を指定します。

なお，*domnam* として指定可能な値は，ドメイン名 domnam で指定したドメイン名に限られます。

（2）PE 番号 *peno*

ドメインを割り付けるプロセッサ・エレメントの PE 番号を指定します。

なお，*peno* として指定可能な値は，1 ～ 2 に限られます。

B. 4　ドメイン情報

ドメインに関する情報として，以下の項目を定義します。

- ドメイン名 domnam

なお，ドメイン情報として定義可能な数は，1 ～ 31 個に限られます。

以下に，ドメイン情報の記述形式を示します。

```
DOMAIN (domnam) {
    // 静的 API 情報
}
```

（1）ドメイン名 *domnam*

ドメインの名前を指定します。

なお，*domnam* として指定可能な値は，名前に限られます。

備考　静的 API 情報についての詳細は，「B. 5　静的 API 情報」を参照してください。

B. 5　静的 API 情報

静的 API 情報として，以下の情報を定義します。

- タスク情報
- セマフォ情報
- イベントフラグ情報
- データ・キュー情報
- メールボックス情報
- ミューテックス情報
- 固定長メモリ・プール情報
- 周期ハンドラ情報
- 割り込みハンドラ情報
- CPU 例外ハンドラ情報
- 初期化ルーチン情報
- アイドル・ルーチン情報

以下に，静的 API 情報の記述イメージを示します。

```
CRE_TSK(tskid, {tskatr, exinf, task, itskpri, stksz[:secnam], stk});    // タスク情報
CRE_SEM(semid, {sematr, isemcnt, maxsem});                              // セマフォ情報
CRE_FLG(flgid, {flgatr, iflgptn});                                      // イベントフラグ情報
CRE_DTQ(dtqid, {dtqatr, dtqcnt[:secnam], dtq});                         // データ・キュー情報
CRE_MBX(mbxid, {mbxatr, maxmpri, mprihd});                              // メールボックス情報
CRE_MTX(mtxid, {mtxatr, ceilpri});                                      // ミューテックス情報
CRE_MPF(mpfid, {mpfatr, blkcnt, blksz[:secnam], mpf});                  // 固定長メモリ・プール情報
CRE_CYC(cycid, {cycatr, exinf, cychdr, cyctim, cycphs});                // 周期ハンドラ情報
DEF_INH(inhno, {inhatr, inthdr});                                       // 割り込みハンドラ情報
DEF_EXC(excno, {excatr, exchdr});                                       // CPU 例外ハンドラ情報
ATT_INI({iniatr, exinf, inirtn});                                       // 初期化ルーチン情報
VATT_IDL({idlatr, idlrtn});                                             // アイドル・ルーチン情報
```

B. 5. 1　タスク情報

タスクに関する情報として，以下の項目を定義します。

- ID tskid
- 属性 tskatr
- 拡張情報 exinf
- 起動アドレス task
- 初期優先度 itskpri
- スタック・サイズ stksz
- セクション名 secnam
- システム予約領域 stk

なお，タスク情報として定義可能な数は，1 ～ 1023 個に限られます。

以下に，タスク情報の記述形式を示します。

```
CRE_TSK(tskid, {tskatr, exinf, task, itskpri, stksz[:secnam], stk});
```

（1）ID *tskid*

タスクの ID を指定します。

なお，*tskid* として指定可能な値は，名前に限られます。

備考　*tskid* と数値の対応は，以下に示した形式でシステム情報ヘッダ・ファイルに出力されます。

```
#define tskid    数値
```

（2）属性 *tskatr*

タスクの属性（記述言語，初期状態，初期割り込み状態）を指定します。

なお，*tskatr* として指定可能な値は，以下のキーワードに限られます。

- タスクの記述言語

  TA_HLNG：　C 言語

  TA_ASM：　アセンブリ言語

- タスクの初期状態（省略可）

  TA_ACT：　実行可能状態

- タスクの初期割り込み状態（省略可）

  TA_ENAINT：　割り込みを許可

  TA_DISINT：　割り込みを禁止

備考 1.　TA_ACT の指定を省略した場合，タスクの初期状態は“休止状態”となります。

2.　TA_ENAINT，および TA_DISINT の指定を省略した場合，タスクの初期割り込み状態は“割り込みを許可”となります。

(3) 拡張情報 *exinf*

タスクの拡張情報を指定します。

なお，*exinf* として指定可能な値は，0x0 ～ 0xFFFFFFFF，またはシンボル名に限られます。

備考　拡張情報は，タスクが休止状態から実行可能状態へと遷移する際，該当タスクの引数として設定されます。

(4) 起動アドレス *task*

タスクの起動アドレスを指定します。

なお，*task* として指定可能な値は，0x0 ～ 0xFFFFFFFE の 2 バイト境界値，またはシンボル名に限られます。

(5) 初期優先度 *itskpri*

タスクの初期優先度を指定します。

なお，*itskpri* として指定可能な値は，1 ～ 最大優先度 maxtpri に限られます。

備考　RI850MP における“タスクの優先度”は，その値が小さいほど，高い優先度であることを意味します。

(6) スタック・サイズ *stksz*

タスク・スタックのサイズ（単位：バイト）を指定します。

なお，*stksz* として指定可能な値は，0x0 ～ 0x10000000 の 4 バイト境界値に限られます。

(7) セクション名 *secnam*

タスク・スタックを割り付けるセクション名を指定します。

なお，*secnam* として指定可能な値は，セクション情報で指定したセクション名に限られます。

備考　本項目の定義を省略した場合，“.kernel_stack_pe*peno*”が指定されていたものとして扱われます。なお，“.kernel_stack_pe*peno*”における *peno* は，本タスク情報が所属しているドメインに割り付けられたプロセッサ・エレメントの PE 番号（プロセッサ・エレメント情報で指定した PE 番号）となります。

(8) システム予約領域 *stk*

システム予約領域です。

なお，*stk* として指定可能な値は，0，または NULL に限られます。

B. 5. 2　セマフォ情報

セマフォに関する情報として，以下の項目を定義します。

- ID semid
- 属性 sematr
- 初期資源数 isemcnt
- 最大資源数 maxsem

なお，セマフォ情報として定義可能な数は，0 ～ 1023 個に限られます。

以下に，セマフォ情報の記述形式を示します。

```
CRE_SEM(semid, {sematr, isemcnt, maxsem});
```

（1）ID *semid*

セマフォの ID を指定します。

なお，*semid* として指定可能な値は，名前に限られます。

備考　*semid* と数値の対応は，以下に示した形式でシステム情報ヘッダ・ファイルに出力されます。

```
#define semid    数値
```

（2）属性 *sematr*

セマフォの属性（キューイング方法）を指定します。

なお，*sematr* として指定可能な値は，以下のキーワードに限られます。

- タスクのキューイング方法

TA_TFIFO :　資源の獲得要求を行った順

TA_TPRI :　タスクの優先度順

（3）初期資源数 *isemcnt*

セマフォの初期資源数を指定します。

なお，*isemcnt* として指定可能な値は，0 ～ 最大資源数 maxsem に限られます。

（4）最大資源数 *maxsem*

セマフォの最大資源数を指定します。

なお，*maxsem* として指定可能な値は，1 ～ 65535 に限られます。

B. 5. 3　イベントフラグ情報

イベントフラグに関する情報として，以下の項目を定義します。

- ID flgid
- 属性 flgatr
- 初期ビット・パターン iflgptn

なお，イベントフラグ情報として定義可能な数は，0 ～ 1023 個に限られます。

以下に，イベントフラグ情報の記述形式を示します。

```
CRE_FLG(flgid, {flgatr, iflgptn});
```

（1）ID *flgid*

イベントフラグの ID を指定します。

なお，*flgid* として指定可能な値は，名前に限られます。

備考　*flgid* と数値の対応は，以下に示した形式でシステム情報ヘッダ・ファイルに出力されます。

```
#define flgid    数値
```

（2）属性 *flgatr*

イベントフラグの属性（キューイング方法，最大タスク数，クリア有無）を指定します。

なお，*flgatr* として指定可能な値は，以下のキーワードに限られます。

- タスクのキューイング方法
  - TA_TFIFO：　ビット・パターンの判定要求を行った順
  - TA_TPRI：　タスクの優先度順
- キューイング可能な最大タスク数
  - TA_WSGL：　1 個のタスク
  - TA_WMUL：　複数のタスク
- ビット・パターンのクリア有無（省略可）
  - TA_CLR：　要求条件を満足した際，ビット・パターンをクリア

備考　TA_CLR の指定を省略した場合，ビット・パターンのクリア有無は“要求条件を満足した際，ビット・パターンをクリアしない”となります。

（3）初期ビット・パターン *iflgptn*

イベントフラグの初期ビット・パターンを指定します。

なお，*iflgptn* として指定可能な値は，0x0 ～ 0xFFFFFFFF に限られます。

B. 5. 4　データ・キュー情報

データ・キューに関する情報として，以下の項目を定義します。

- ID dtqid
- 属性 dtqatr
- 最大データ数 dtqcnt
- セクション名 secnam
- システム予約領域 dtq

なお，データ・キュー情報として定義可能な数は，0 ～ 1023 個に限られます。

以下に，データ・キュー情報の記述形式を示します。

```
CRE_DTQ(dtqid, {dtqatr, dtqcnt[:secnam], dtq});
```

（1）ID *dtqid*

データ・キューの ID を指定します。

なお，*dtqid* として指定可能な値は，名前に限られます。

備考　*dtqid* と数値の対応は，以下に示した形式でシステム情報ヘッダ・ファイルに出力されます。

```
#define dtqid    数値
```

（2）属性 *dtqatr*

データ・キューの属性（キューイング方法）を指定します。

なお，*dtqatr* として指定可能な値は，以下のキーワードに限られます。

- タスクのキューイング方法

TA_TFIFO :　データの送信要求を行った順

TA_TPRI :　タスクの優先度順

備考　タスクがデータの受信要求を行った際，データを即時受信することができなかった場合，該当タスクはデータ・キューの待ちキューに，データの受信要求を行った順にキューイングされます。

（3）最大データ数 *dtqcnt*

データ・キューのバッファ領域に書き込み可能なデータの最大数を指定します。

なお，*dtqcnt* として指定可能な値は，0 ～ 1023 に限られます。

（4）セクション名 *secnam*

データ・キューのバッファ領域を割り付けるセクション名を指定します。

なお，*secnam* として指定可能な値は，セクション情報で指定したセクション名に限られます。

備考　本項目の定義を省略した場合，“.kernel_work_pe*peno*”が指定されていたものとして扱われます。なお，“.kernel_work_pe*peno*”における *peno* は，本データ・キュー情報が所属しているドメインに

割り付けられたプロセッサ・エレメントの PE 番号（プロセッサ・エレメント情報で指定した PE 番号）となります。

（5）システム予約領域 *dtq*

システム予約領域です。

なお，*dtq* として指定可能な値は，0，または NULL に限られます。

B. 5. 5　メールボックス情報

メールボックスに関する情報として，以下の項目を定義します。

- ID mbxid
- 属性 mbxatr
- 最大優先度 maxmpri
- システム予約領域 mprihd

なお，メールボックス情報として定義可能な数は，0 ～ 1023 個に限られます。

以下に，メールボックス情報の記述形式を示します。

```
CRE_MBX(mbxid, {mbxatr, maxmpri, mprihd});
```

（1）ID *mbxid*

メールボックスの ID を指定します。

なお，*mbxid* として指定可能な値は，名前に限られます。

備考　*mbxid* と数値の対応は，以下に示した形式でシステム情報ヘッダ・ファイルに出力されます。

```
#define mbxid    数値
```

（2）属性 *mbxatr*

メールボックスの属性（キューイング方法）を指定します。

なお，*mbxatr* として指定可能な値は，以下のキーワードに限られます。

- タスクのキューイング方法

  TA_TFIFO：　メッセージの受信要求を行った順

  TA_TPRI：　タスクの優先度順

- メッセージのキューイング方法

  TA_MFIFO：　メッセージの送信要求を行った順

  TA_MPRI：　メッセージの優先度順

（3）最大優先度 *maxmpri*

メールボックスに送信可能なメッセージの最大優先度を指定します。

なお，*maxmpri* として指定可能な値は，1 ～ 32767 に限られます。

備考　RI850MP における“メッセージの優先度”は，その値が小さいほど，高い優先度であることを意味します。

（4）システム予約領域 *mprihd*

システム予約領域です。

なお，*mprihd* として指定可能な値は，0，または NULL に限られます。

B. 5. 6　ミューテックス情報

ミューテックスに関する情報として，以下の項目を定義します。

- ID mtxid
- 属性 mtxatr
- システム予約領域 ceilpri

なお，ミューテックス情報として定義可能な数は，0 ～ 1023 個に限られます。

以下に，ミューテックス情報の記述形式を示します。

```
CRE_MTX(mtxid, {mtxatr, ceilpri});
```

（1）ID *mtxid*

ミューテックスの ID を指定します。

なお，*mtxid* として指定可能な値は，名前に限られます。

備考　*mtxid* と数値の対応は，以下に示した形式でシステム情報ヘッダ・ファイルに出力されます。

```
#define mtxid    数値
```

（2）属性 *mtxatr*

ミューテックスの属性（キューイング方法）を指定します。

なお，*mtxatr* として指定可能な値は，以下のキーワードに限られます。

- タスクのキューイング方法

TA_TFIFO：　ミューテックスの獲得要求を行った順

TA_TPRI：　タスクの優先度順

（3）システム予約領域 *ceilpri*

システム予約領域です。

なお，*ceilpri* として指定可能な値は，0，または NULL に限られます。

B. 5. 7　固定長メモリ・プール情報

固定長メモリ・プールに関する情報として，以下の項目を定義します。

- ID mpfid
- 属性 mpfatr
- ブロック数 blkcnt
- ブロック・サイズ blksz
- セクション名 secnam
- システム予約領域 mpf

なお，固定長メモリ・プール情報として定義可能な数は，0 ～ 1023 個に限られます。

以下に，固定長メモリ・プール情報の記述形式を示します。

```
CRE_MPF(mpfid, {mpfatr, blkcnt, blksz[:secnam], mpf});
```

（1）ID *mpfid*

固定長メモリ・プールの ID を指定します。

なお，*mpfid* として指定可能な値は，名前に限られます。

備考　*mpfid* と数値の対応は，以下に示した形式でシステム情報ヘッダ・ファイルに出力されます。

```
#define mpfid    数値
```

（2）属性 *mpfatr*

固定長メモリ・プールの属性（キューイング方法）を指定します。

なお，*mpfatr* として指定可能な値は，以下のキーワードに限られます。

- タスクのキューイング方法

TA_TFIFO :　固定長メモリ・ブロックの獲得要求を行った順

TA_TPRI :　タスクの優先度順

（3）ブロック数 *blkcnt*

固定長メモリ・ブロックの総ブロック数を指定します。

なお，*blkcnt* として指定可能な値は，1 ～ 32767 に限られます。

（4）ブロック・サイズ *blksz*

1 ブロック当りのサイズ（単位：バイト）を指定します。

なお，*blksz* として指定可能な値は，0x1 ～ 0x100000000 の 4 バイト境界値に限られます。

（5）セクション名 *secnam*

固定長メモリ・プールを割り付けるセクション名を指定します。

なお，*secnam* として指定可能な値は，セクション情報で指定したセクション名に限られます。

備考　本項目の定義を省略した場合，“.kernel_work_pe*peno*”が指定されていたものとして扱われます。
なお，“.kernel_work_pe*peno*”における *peno* は，本固定長メモリ・プール情報が所属しているドメインに割り付けられたプロセッサ・エレメントの PE 番号（プロセッサ・エレメント情報で指定した PE 番号）となります。

（6）システム予約領域 *mpf*

システム予約領域です。
なお，*mpf* として指定可能な値は，0，または NULL に限られます。

B. 5. 8　周期ハンドラ情報

周期ハンドラに関する情報として，以下の項目を定義します。

- ID cycid
- 属性 cycatr
- 拡張情報 exinf
- 起動アドレス cychdr
- 起動周期 cyctim
- 初期起動位相 cycphs

なお，周期ハンドラ情報として定義可能な数は，0 ～ 1023 個に限られます。

以下に，周期ハンドラ情報の記述形式を示します。

```
CRE_CYC(cycid, {cycatr, exinf, cychdr, cyctim, cycphs});
```

（1）ID *cycid*

周期ハンドラの ID を指定します。

なお，*cycid* として指定可能な値は，名前に限られます。

備考　*cycid* と数値の対応は，以下に示した形式でシステム情報ヘッダ・ファイルに出力されます。

```
#define cycid    数値
```

（2）属性 *cycatr*

周期ハンドラの属性（記述言語，初期状態，保存有無）を指定します。

なお，*cycatr* として指定可能な値は，以下のキーワードに限られます。

- 周期ハンドラの記述言語

  TA_HLNG :　C 言語

  TA_ASM :　アセンブリ言語

- 周期ハンドラの初期状態（省略可）

  TA_STA :　動作状態

- 起動位相の保存有無（省略可）

  TA_PHS :　起動位相を保存

備考 1.　TA_STA の指定を省略した場合，周期ハンドラの初期状態は“停止状態”となります。

2.　TA_PHS の指定を省略した場合，起動位相の保存有無は“起動位相を保存しない”となります。

（3）拡張情報 *exinf*

周期ハンドラの拡張情報を指定します。

なお，*exinf* として指定可能な値は，0x0 ～ 0xFFFFFFFF，またはシンボル名に限られます。

備考 拡張情報は，周期ハンドラが停止状態から動作状態へと遷移する際，該当周期ハンドラの引数として設定されます。

(4) 起動アドレス *cychdr*

周期ハンドラの起動アドレスを指定します。

なお，*cychdr* として指定可能な値は，0x0 ～ 0xFFFFFFFE の 2 バイト境界値，またはシンボル名に限られます。

(5) 起動周期 *cyctim*

周期ハンドラの起動周期（単位：ミリ秒）を指定します。

なお，*cyctim* として指定可能な値は，0x1 ～ 0x7FFFFFFF に限られます。

(6) 初期起動位相 *cycphs*

周期ハンドラの初期起動位相（単位：ミリ秒）を指定します。

なお，*cycphs* として指定可能な値は，0x1 ～ 0x7FFFFFFF に限られます。

B. 5. 9　割り込みハンドラ情報

割り込みハンドラに関する情報として，以下の項目を定義します。

- 例外要因コード inhno
- 属性 inhatr
- 起動アドレス inthdr

なお，割り込みハンドラ情報として定義可能な数は，1 例外要因に対して 1PE 当り 0 ～ 1 個に限られます。

以下に，割り込みハンドラ情報の記述形式を示します。

```
DEF_INH(inhno, {inhatr, inthdr});
```

（1）例外要因コード *inhno*

割り込みハンドラの起動要因となる割り込みに対応した例外要因コードを指定します。

なお，*inhno* として指定可能な値は，割り込みに対応した 0x90 以降の 16 バイト境界値，または例外要因名に限られます。

備考 1. 例外要因コード tintno，または例外要因コード excno で指定した例外要因コードを割り当てることはできません。

2. 本項目で指定可能な例外要因名は，デバイス・ファイルで規定されている例外要因名に限られます。

（2）属性 *inhatr*

割り込みハンドラの属性（記述言語）を指定します。

なお，*inhatr* として指定可能な値は，以下のキーワードに限られます。

- 割り込みハンドラの記述言語

TA_HLNG：　C 言語

TA_ASM：　アセンブリ言語

（3）起動アドレス *inthdr*

割り込みハンドラの起動アドレスを指定します。

なお，*inthdr* として指定可能な値は，0x0 ～ 0xFFFFFFFE の 2 バイト境界値，またはシンボル名に限られます。

B. 5. 10　CPU 例外ハンドラ情報

CPU 例外ハンドラに関する情報として，以下の項目を定義します。

- 例外要因コード excno
- 属性 excatr
- 起動アドレス exchdr

なお，CPU 例外ハンドラ情報として定義可能な数は，1 例外要因に対して 1PE 当り 0 ～ 1 個に限られます。

以下に，CPU 例外ハンドラ情報の記述形式を示します。

```
DEF_EXC(excno, {excatr, exchdr});
```

（1）例外要因コード *excno*

CPU 例外ハンドラの起動要因となる CPU 例外（EI レベル・ソフトウエア例外，浮動小数点演算割り込み）に対応した例外要因コードを指定します。

なお，*excno* として指定可能な値は，CPU 例外に対応した 16 バイト境界値，または例外要因名に限られます。

備考 1. 例外要因コード tintno，または例外要因コード inhno で指定した例外要因コードを割り当てることはできません。

2. 本項目で指定可能な例外要因名は，デバイス・ファイルで規定されている例外要因名に限られます。

（2）属性 *excatr*

CPU 例外ハンドラの属性（記述言語）を指定します。

なお，*excatr* として指定可能な値は，以下のキーワードに限られます。

- CPU 例外ハンドラの記述言語

TA_HLNG :　C 言語

TA_ASM :　アセンブリ言語

（3）起動アドレス *exchdr*

CPU 例外ハンドラの起動アドレスを指定します。

なお，*exchdr* として指定可能な値は，0x0 ～ 0xFFFFFFFE の 2 バイト境界値，またはシンボル名に限られます。

B. 5. 11　初期化ルーチン情報

初期化ルーチンに関する情報として，以下の項目を定義します。

- 属性 iniatr
- 拡張情報 exinf
- 起動アドレス inirtn

なお，初期化ルーチン情報として定義可能な数は，0 ～ 1023 個に限られます。

以下に，初期化ルーチン情報の記述形式を示します。

```
ATT_INI({iniatr, exinf, inirtn});
```

（1）属性 *iniatr*

初期化ルーチンの属性（記述言語）を指定します。

なお，*iniatr* として指定可能な値は，以下のキーワードに限られます。

- 初期化ルーチンの記述言語

TA_HLNG：　C 言語

TA_ASM：　アセンブリ言語

（2）拡張情報 *exinf*

初期化ルーチンの拡張情報を指定します。

なお，*exinf* として指定可能な値は，0x0 ～ 0xFFFFFFFF，またはシンボル名に限られます。

備考　拡張情報は，初期化ルーチンがカーネル初期化部から呼び出された際，該当初期化ルーチンの引数として設定されます。

（3）起動アドレス *inirtn*

初期化ルーチンの起動アドレスを指定します。

なお，*inirtn* として指定可能な値は，0x0 ～ 0xFFFFFFFE の 2 バイト境界値，またはシンボル名に限られます。

B. 5. 12 アイドル・ルーチン情報

アイドル・ルーチンに関する情報として，以下の項目を定義します。

- 属性 idlatr
- 起動アドレス idlrtn

なお，アイドル・ルーチン情報として定義可能な数は，1PE 当り 0 ～ 1 個に限られます。

以下に，アイドル・ルーチン情報の記述形式を示します。

```
VATT_IDL({idlatr, idlrtn});
```

（1）属性 *idlatr*

アイドル・ルーチンの属性（記述言語）を指定します。

なお，*idlatr* として指定可能な値は，以下のキーワードに限られます。

- アイドル・ルーチンの記述言語

TA_HLNG： C 言語

TA_ASM： アセンブリ言語

（2）起動アドレス *idlrtn*

アイドル・ルーチンの起動アドレスを指定します。

なお，*idlrtn* として指定可能な値は，0x0 ～ 0xFFFFFFFE の 2 バイト境界値，またはシンボル名に限られます。

備考 本項目の定義を省略した場合，以下の記述が行われていたものとして扱われます。

```
VATT_IDL(TA_HLNG, _default_idlrtn);
```

B. 6　SCT 情報

RI850MP が提供しているサービス・コールの使用有無に関する情報として，以下の項目を定義します。

- サービス・コール名 svc_nam

なお，SCT 情報として定義可能な数は，0 ～ 69 個に限られます。

以下に，SCT 情報の記述形式を示します。

```
DEF_SCT(svc_nam);
```

（1）サービス・コール名 *svc_nam*

処理プログラム内で使用するサービス・コールの名称を指定します。

なお，*svc_nam* として指定可能な値は，以下のキーワードに限られます。

- タスク管理機能

  act_tsk，iact_tsk，can_act，ican_act，ext_tsk，ter_tsk，chg_pri，ichg_pri，get_pri，iget_pri，ref_tsk，iref_tsk

- タスク付属同期機能

  slp_tsk，tslp_tsk，wup_tsk，iwup_tsk，can_wup，ican_wup，rel_wai，irel_wai，sus_tsk，isus_tsk，rsm_tsk，irsm_tsk，frsm_tsk，ifrsm_tsk，dly_tsk

- 同期・通信機能（セマフォ）

  sig_sem，isig_sem，wai_sem，pol_sem，ipol_sem，twai_sem，ref_sem，iref_sem

- 同期・通信機能（イベントフラグ）

  set_flg，iset_flg，clr_flg，iclr_flg，wai_flg，pol_flg，ipol_flg，twai_flg，ref_flg，iref_flg

- 同期・通信機能（データ・キュー）

  snd_dtq，psnd_dtq，ipsnd_dtq，tsnd_dtq，fsnd_dtq，ifsnd_dtq，rcv_dtq，prcv_dtq，iprcv_dtq，trcv_dtq，ref_dtq，iref_dtq

- 同期・通信機能（メールボックス）

  snd_mbx，isnd_mbx，rcv_mbx，prcv_mbx，iprcv_mbx，trcv_mbx，ref_mbx，iref_mbx

- 拡張同期・通信機能

  loc_mtx，ploc_mtx，tloc_mtx，unl_mtx，ref_mtx，iref_mtx

- メモリ・プール管理機能

  get_mpf，pget_mpf，ipget_mpf，tget_mpf，rel_mpf，irel_mpf，ref_mpf，iref_mpf

- 時間管理機能

  set_tim，iset_tim，get_tim，iget_tim，sta_cyc，ista_cyc，stp_cyc，istp_cyc，ref_cyc，iref_cyc

- システム状態管理機能

  rot_rdq，irot_rdq，get_tid，iget_tid，loc_cpu，iloc_cpu，unl_cpu，iunl_cpu，dis_dsp，ena_dsp，sns_ctx，sns_loc，sns_dsp，sns_dpn

- 割り込み管理機能

  dis_int，ena_int，chg_ipm，ichg_ipm，get_ipm，iget_ipm

付録C 索 引

【さ行】

【た行】

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2011.04.01	―	初版発行

RI850MP ユーザーズマニュアル
コーディング編

発行年月日　2011 年 4 月 1 日　Rev.1.00
発行　ルネサス エレクトロニクス株式会社
〒211-8668 神奈川県川崎市中原区下沼部 1753

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

http://www.renesas.com

※営業お問合せ窓口の住所・電話番号は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス販売株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル）　(03)5201-5307

■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：http://japan.renesas.com/inquiry

RI850MP

RENESAS

ルネサス エレクトロニクス株式会社

R20UT0519JJ0100