

## TOPPERS新世代カーネル統合仕様書

バージョン: Release 1.0.0

最終更新: 2008年11月19日

このドキュメントは、TOPPERS新世代カーネルに属する一連のリアルタイムカーネルの仕様を、統合的に記述するものである。現時点で、TOPPERS/ASPカーネルの仕様に関しては記述が完成しているが、未完成部分も多い。保護機能対応カーネル、マルチプロセッサ対応カーネル、動的生成対応カーネルについては、今後仕様検討と記述を進める。なお、本文中から参照している図は、ファイルの最後にまとめて掲載してある。

---

### TOPPERS New Generation Kernel Specification

Copyright (C) 2006-2008 by Embedded and Real-Time Systems Laboratory  
Graduate School of Information Science, Nagoya Univ., JAPAN  
Copyright (C) 2006-2008 by TOPPERS Project, Inc., JAPAN

上記著作権者は、以下の (1) ~ (3) の条件を満たす場合に限り、本ドキュメント (本ドキュメントを改変したものを含む。以下同じ) を使用・複製・改変・再配布 (以下、利用と呼ぶ) することを無償で許諾する。

- (1) 本ドキュメントを利用する場合には、上記の著作権表示、この利用条件および下記の無保証規定が、そのままの形でドキュメント中に含まれていること。
- (2) 本ドキュメントを改変する場合には、ドキュメントを改変した旨の記述を、改変後のドキュメント中に含めること。ただし、改変後のドキュメントが、TOPPERSプロジェクト指定の開発成果物である場合には、この限りではない。
- (3) 本ドキュメントの利用により直接的または間接的に生じるいかなる損害からも、上記著作権者およびTOPPERSプロジェクトを免責すること。また、本ドキュメントのユーザまたはエンドユーザからのいかなる理由に基づく請求からも、上記著作権者およびTOPPERSプロジェクトを免責すること。

本ドキュメントは、無保証で提供されているものである。上記著作権者およびTOPPERSプロジェクトは、本ドキュメントに関して、特定の使用目的に対する適合性も含めて、いかなる保証も行わない。また、本ドキュメントの利用により直接的または間接的に生じたいかなる損害に関しても、その責任を負わない。

---

## 目次

### 第1章 TOPPERS新世代カーネルの概要

- 1.1 TOPPERS新世代カーネル仕様の設計方針
- 1.2 TOPPERS/ASPカーネルの適用対象領域と仕様設計方針

### 第2章 主要な概念と共通定義

- 2.1 仕様の位置付け
  - 2.1.1 カーネルの機能セット
  - 2.1.2 ターゲット非依存の規定とターゲット定義の規定
  - 2.1.3 想定するソフトウェア構成
  - 2.1.4 想定するプログラミング言語
- 2.2 APIの構成要素とコンベンション
  - 2.2.1 APIの構成要素

- 2.2.2 パラメータとリターンパラメータ
- 2.2.3 返値とエラーコード
- 2.2.4 機能コード
- 2.2.5 ヘッダファイル
- 2.3 主な概念
  - 2.3.1 オブジェクトと処理単位
  - 2.3.2 サービスコールとパラメータ
  - 2.3.3 保護機能
  - 2.3.4 マルチプロセッサ対応
- 2.4 処理単位の種類と実行順序
  - 2.4.1 処理単位の種類
  - 2.4.2 処理単位の実行順序
  - 2.4.3 カーネル処理の不可分性
- 2.5 システム状態とコンテキスト
  - 2.5.1 カーネル動作状態と非動作状態
  - 2.5.2 タスクコンテキストと非タスクコンテキスト
  - 2.5.3 カーネルの振舞いに影響を与える状態
  - 2.5.4 全割込みロック状態と全割込みロック解除状態
  - 2.5.5 CPUロック状態とCPUロック解除状態
  - 2.5.6 割込み優先度マスク
  - 2.5.7 ディスパッチ禁止状態とディスパッチ許可状態
  - 2.5.8 ディスパッチ保留状態
  - 2.5.9 カーネル管理外の割込み
  - 2.5.10 カーネル管理外のCPU例外
  - 2.5.11 カーネル管理外の状態
  - 2.5.12 処理単位とシステム状態
- 2.6 タスクの状態遷移とスケジューリング規則
  - 2.6.1 基本的なタスク状態
  - 2.6.2 タスクの状態遷移
  - 2.6.3 タスクのスケジューリング規則
  - 2.6.4 待ち行列と待ち解除の順序
  - 2.6.5 待ち禁止状態
  - 2.6.6 ディスパッチ保留状態で実行中のタスクに対する強制待ち
- 2.7 割込み処理モデル
  - 2.7.1 割込み処理の流れ
  - 2.7.2 割込み優先度
  - 2.7.3 割込み要求ラインの属性
  - 2.7.4 割込みを受け付ける条件
  - 2.7.5 割込み番号と割込みハンドラ番号
  - 2.7.6 カーネル管理外の割込みの設定方法
- 2.8 CPU例外処理モデル
  - 2.8.1 CPU例外処理の流れ
  - 2.8.2 CPU例外ハンドラから呼び出せるサービスコール
- 2.9 システム初期化手順
- 2.10 システム終了手順
- 2.11 オブジェクトの登録とその解除
  - 2.11.1 ID番号で識別するオブジェクト
  - 2.11.2 オブジェクト番号で識別するオブジェクト
  - 2.11.3 オブジェクト生成に必要なメモリ領域
- 2.12 システムコンフィギュレーション手順
  - 2.12.1 システムコンフィギュレーションファイル
  - 2.12.2 静的APIの文法とパラメータ
  - 2.12.3 保護ドメインの指定 ( 未完成 )
  - 2.12.4 クラスの指定 ( 未完成 )
  - 2.12.5 コンフィギュレータの処理モデル
  - 2.12.6 静的APIのパラメータに関するエラー検出
  - 2.12.7 オブジェクトのID番号の指定
- 2.13 TOPPERSネーミングコンベンション

- 2.13.1 モジュール識別名
- 2.13.2 データ型名
- 2.13.3 関数名
- 2.13.4 変数名
- 2.13.5 定数名
- 2.13.6 マクロ名
- 2.13.7 静的API名
- 2.13.8 ファイル名
- 2.13.9 モジュール内部の名称の衝突回避
- 2.14 TOPPERS共通定義
  - 2.14.1 TOPPERS共通ヘッダファイル
  - 2.14.2 TOPPERS共通データ型
  - 2.14.3 TOPPERS共通定数
  - 2.14.4 TOPPERS共通エラーコード
  - 2.14.5 TOPPERS共通マクロ
  - 2.14.6 TOPPERS共通構成マクロ
- 2.15 カーネル共通定義
  - 2.15.1 カーネルヘッダファイル
  - 2.15.2 カーネル共通定数
  - 2.15.3 カーネル共通構成マクロ

### 第3章 システムインタフェースレイヤAPI仕様

- 3.1 システムインタフェースレイヤの概要
- 3.2 SILヘッダファイル
- 3.3 全割込みロック状態の制御
- 3.4 微少時間待ち
- 3.5 エンディアン
- 3.6 メモリ空間アクセス関数
- 3.7 I/O空間アクセス関数

### 第4章 カーネルAPI仕様

- 4.1 タスク管理機能
- 4.2 タスク付属同期機能
- 4.3 タスク例外処理機能
- 4.4 同期・通信機能
  - 4.4.1 セマフォ
  - 4.4.2 イベントフラグ
  - 4.4.3 データキュー
  - 4.4.4 優先度データキュー
  - 4.4.5 メールボックス
  - 4.4.6 ミューテックス
  - 4.4.7 メッセージバッファ
- 4.5 メモリプール管理機能
  - 4.5.1 固定長メモリプール
- 4.6 時間管理機能
  - 4.6.1 システム時刻管理
  - 4.6.2 周期ハンドラ
  - 4.6.3 アラームハンドラ
  - 4.6.4 オーバランハンドラ
- 4.7 システム状態管理機能
- 4.8 メモリオブジェクト管理機能
- 4.9 割込み管理機能
- 4.10 CPU例外管理機能
- 4.11 拡張サービスコール管理機能
- 4.12 システム構成管理機能

## 第5章 リファレンス

- 5.1 サービスコール一覧
- 5.2 静的API一覧
- 5.3 データ型
  - 5.3.1 TOPPERS共通データ型
  - 5.3.2 カーネルの使用データ型
  - 5.3.3 カーネルの使用するバケット形式
- 5.4 定数とマクロ
  - 5.4.1 TOPPERS共通定数
  - 5.4.2 TOPPERS共通マクロ
  - 5.4.3 カーネル共通定数
  - 5.4.4 カーネルの機能毎の定数
  - 5.4.5 カーネルの機能毎のマクロ
- 5.5 構成マクロ
  - 5.5.1 TOPPERS共通構成マクロ
  - 5.5.2 カーネル共通構成マクロ
  - 5.5.3 カーネルの機能毎の構成マクロ
- 5.6 エラーコード一覧
- 5.7 機能コード一覧
- 5.8 カーネルオブジェクトに対するアクセスの種別

## 第1章 TOPPERS新世代カーネルの概要

TOPPERS新世代カーネルとは、TOPPERSプロジェクトにおいてITRON仕様をベースとして開発している一連のリアルタイムカーネルの総称である。この章では、TOPPERS新世代カーネル仕様の設計方針と、それに属する各カーネルの適用対象領域と設計方針について述べる。

### 1.1 TOPPERS新世代カーネル仕様の設計方針

TOPPERS新世代カーネル仕様を設計するにあたり、次の方針を設定する。

#### (1) $\mu$ ITRON4.0仕様をベースに拡張・改良を加える

TOPPERS新世代カーネル仕様は、多くの技術者の尽力により作成され、多くの実装・使用実績がある $\mu$ ITRON4.0仕様をベースとする。ただし、 $\mu$ ITRON4.0仕様の策定時以降の状況の変化を考慮し、 $\mu$ ITRON4.0仕様で不十分と考えられる点については積極的に拡張・改良する。 $\mu$ ITRON4.0仕様への準拠性にはこだわらない。

#### (2) ソフトウェアの再利用性を重視する

$\mu$ ITRON4.0仕様の策定時点と比べると、組み込みソフトウェアの大規模化が進展している一方で、ハードウェアの性能向上も著しい。そのため、ソフトウェアの再利用性を向上させるためには、少々のオーバーヘッドは許容される状況にある。

そこで、TOPPERS新世代カーネル仕様では、 $\mu$ ITRON4.0仕様においてオーバーヘッド削減のために実装定義または実装依存としていたような項目についても、ターゲットシステムに依存する項目とするのではなく、強く規定する方針とする。

#### (3) 高信頼・安全なシステム構築を支援する

TOPPERS新世代カーネル仕様は、高信頼・安全な組み込みシステム構築を支援するものとする。

安全性の面では、アプリケーションプログラムに問題がある場合でも、リーゾナブルなオーバヘッドでそれを救済できるなら、救済するような仕様とする。また、アプリケーションプログラムの誤動作を検出する機能や、システムの自己診断のための機能についても、順次取り込んでいく。

#### (4) アプリケーションシステム構築に必要な機能は積極的に取り込む

上記の方針を満たした上で、多くのアプリケーションシステムに共通に必要なとなる機能については、積極的にカーネルに取り込む。

カーネル単体の信頼性を向上させるためには、カーネルの機能は少なくした方が楽である。しかし、アプリケーションシステム構築に必要なとなる機能は、カーネルがサポートしていなければアプリケーションプログラムで実現しなければならず、システム全体の信頼性を考えると、多くのアプリケーションシステムに共通に必要なとなる機能については、カーネルに取り込んだ方が有利である。

### 1.2 TOPPERS/ASPカーネルの適用対象領域と仕様設計方針

TOPPERS/ASPカーネル（ASPは、Advanced Standard Profileの略。以下、ASPカーネル）は、TOPPERS新世代カーネルの基盤となるリアルタイムカーネルである。TOPPERS新世代カーネルの中で今後開発していく保護機能を持ったカーネルやマルチプロセッサ対応のカーネルは、ASPカーネルを拡張する形で開発する。

ASPカーネルは、20年以上に渡るITRON仕様の技術開発成果をベースとして、完成度の高いリアルタイムカーネルを実現するものである。完成度を高めるといふ観点から、カーネル本体の仕様については、枯れた技術で実装できる範囲に留める。

ASPカーネルの主な適用対象は、高い信頼性・安全性・リアルタイム性を要求される組込みシステムとする。ソフトウェア規模の面では、プログラムサイズ（バイナリコード）が数十KB～1MB程度のシステムを主な適用対象とする。それより大規模なシステムには、保護機能を持ったカーネルを適用すべきと考えられる。

ASPカーネルの機能は、カーネル内で動的なメモリ管理が不要な範囲に留める。これは、高い信頼性・安全性・リアルタイム性を要求される組込みシステムでは、システム稼働中に発生するメモリ不足への対処が難しいためである。この方針から、カーネルオブジェクトは静的に生成することとし、動的なオブジェクト生成機能は設けない。ただし、アプリケーションプログラムが動的なメモリ管理をするためのカーネル機能である固定長メモリプール機能はサポートする。

## 第2章 主要な概念と共通定義

### 2.1 仕様の位置付け

この仕様は、TOPPERS新世代カーネルに属する各カーネルの仕様を、統合的に記述することを目標としている。また、TOPPERS新世代カーネル上で動作する各種のシステムサービスに共通に適用される事項についても規定する。

#### 2.1.1 カーネルの機能セット

TOPPERS新世代カーネルは、ASPカーネルをベースとして、保護機能、マルチプロセッサ、カーネルオブジェクトの動的生成、機能安全などに対応した一連のカーネルで構成される。

この仕様では、TOPPERS新世代カーネルを構成する一連のカーネルの仕様を統合

的に記述するが、言うまでもなく、カーネルの種類によってサポートする機能は異なる。サポートする機能をカーネルの種類毎に記述する方法もあるが、カーネルの種類はユーザ要求に対応して増える可能性もあり、その度に仕様書を修正するのは得策ではない。

そこでこの仕様では、サポートする機能を、カーネルの種類毎ではなく、カーネルの対応する機能セット毎に記述する。具体的には、保護機能を持ったカーネルを保護機能対応カーネル、マルチプロセッサに対応したカーネルをマルチプロセッサ対応カーネル、カーネルオブジェクトの動的生成機能を持ったカーネルを動的生成対応カーネルと呼ぶことにする。

なお、ASPカーネルは、保護機能対応カーネル、マルチプロセッサ対応カーネル、動的生成対応カーネルのいずれでもない。

### 2.1.2 ターゲット非依存の規定とターゲット定義の規定

TOPPERS新世代カーネルは、アプリケーションプログラムの再利用性を向上させるために、ターゲットハードウェアや開発環境の違いをできる限り隠蔽することを目指している。ただし、ターゲットハードウェアや開発環境の制限によって実現できない機能が生じたり、逆にターゲットハードウェアの特徴を活かすためには機能拡張が不可欠になる場合がある。また、同一のターゲットハードウェアであっても、アプリケーションシステムによって使用方法が異なる場合があり、ターゲットシステム毎に仕様の細部に違いが生じることは避けられない。

そこで、TOPPERS新世代カーネルの仕様は、ターゲットシステムによらずに定めるターゲット非依存 (target-independent) の規定と、ターゲットシステム毎に定めるターゲット定義 (target-defined) の規定に分けて記述する。この仕様書は、ターゲット非依存の規定について記述するものであり、この仕様書で「ターゲット定義」とした事項は、ターゲットシステム毎に用意するドキュメントにおいて規定する。

また、この仕様書でターゲット非依存に規定した事項であっても、ターゲットハードウェアや開発環境の制限によって実現できない場合や、実現するためのオーバーヘッドが大きくなる場合には、この仕様書の規定を逸脱する場合がある。このような場合には、ターゲットシステム毎に用意するドキュメントでその旨を明記する。

### 2.1.3 想定するソフトウェア構成

この仕様では、アプリケーションシステムを構成するソフトウェアを、アプリケーションプログラム (以下、単にアプリケーションと呼ぶ)、システムサービス、カーネルの3階層に分けて考える (図2-1)。カーネルとシステムサービスをあわせて、ソフトウェアプラットフォームと呼ぶ。

カーネルは、コンピュータの持つ最も基本的なハードウェア資源であるプロセッサ、メモリ、タイマを抽象化し、上位階層のソフトウェア (アプリケーションおよびシステムサービス) に論理的なプログラム実行環境を提供するソフトウェアである。

システムサービスは、各種の周辺デバイスを抽象化するソフトウェアで、ファイルシステムやネットワークプロトコルスタック、各種のデバイスドライバなどが含まれる。

また、この仕様では、プロセッサと各種の周辺デバイスの接続方法を隠蔽するためのソフトウェア階層として、システムインタフェースレイヤ (SIL) を規定する。

システムインタフェースレイヤ，カーネル，各種のシステムサービス（これらをモジュールと呼ぶ）を，上位階層のソフトウェアから使うためのインタフェースを，API（Application Programming Interface）と呼ぶ．

この仕様書では，第3章においてシステムインタフェースレイヤのAPI仕様を，第4章においてカーネルのAPI仕様を規定する．システムサービスのAPI仕様は，システムサービス毎の仕様書で規定される．

#### 【 $\mu$ ITRON4.0仕様との関係】

$\mu$ ITRON4.0仕様では，カーネルとアプリケーションの間にあるソフトウェアをソフトウェア部品と呼んでいたが，TOPPERS組込みコンポーネントシステム（TECS）においてはカーネルもソフトウェア部品の1つと捉えることから，この仕様ではシステムサービスと呼ぶことにした．

#### 2.1.4 想定するプログラミング言語

この仕様書におけるAPI仕様は，ISO/IEC 9899:1990（以下，C90と呼ぶ）またはISO/IEC 9899:1999（以下，C99と呼ぶ）に準拠したC言語を，フリースタANDING環境で用いることを想定して規定している．

ただし，C90の規定に加えて，以下のことを仮定している．

- ・16ビットおよび32ビットの整数型があること
- ・ポインタが格納できるサイズの整数型があること

#### 2.2 APIの構成要素とコンベンション

##### 2.2.1 APIの構成要素

###### (1) サービスコール

上位階層のソフトウェアから，下位階層のソフトウェアを呼び出すインタフェースをサービスコール（service call）と呼ぶ．カーネルのサービスコールを，システムコール（system call）と呼ぶ場合もある．

###### (2) コールバック

下位階層のソフトウェアから，上位階層のソフトウェアを呼び出すインタフェースをコールバック（callback）と呼ぶ．

###### (3) 静的API

オブジェクトの生成情報や初期状態などを定義するために，システムコンフィギュレーションファイル中に記述するインタフェースを，静的API（static API）と呼ぶ．

###### (4) 構成マクロ

下位階層のソフトウェアに関する各種の情報を取り出すために，上位階層のソフトウェアが用いるマクロを，構成マクロ（configuration macro）と呼ぶ．

##### 2.2.2 パラメータとリターンパラメータ

サービスコールやコールバックに渡すデータをパラメータ（parameter），それらが返すデータをリターンパラメータ（return parameter）と呼ぶ．また，静的APIに渡すデータもパラメータと呼ぶ．

オブジェクトを生成するサービスコールなど、パラメータの数が多い場合やターゲット定義のパラメータを追加する可能性がある場合には、複数のパラメータを1つの構造体に入れ、その領域へのポインタをパラメータとして渡す。また、パラメータのサイズが大きい場合にも、パラメータを入れた領域へのポインタをパラメータとして渡す場合がある。

C言語APIでは、リターンパラメータは、関数の返値とするか、リターンパラメータを入れる領域へのポインタをパラメータとして渡すことで実現する。オブジェクトの状態を参照するサービスコールなど、リターンパラメータの数が多い場合やターゲット定義のリターンパラメータを追加する可能性がある場合には、複数のリターンパラメータを1つの構造体に入れて返すこととし、その領域へのポインタをパラメータとして渡す。

複数のパラメータまたはリターンパラメータを入れるための構造体を、パケット (packet) と呼ぶ。

サービスコールやコールバックに、パケットを置く領域へのポインタやリターンパラメータを入れる領域へのポインタを渡す場合、別に規定がない限りは、サービスコールやコールバックの処理が完了した後は、それらの領域を参照することはなく、別の目的に使用できる。

### 2.2.3 返値とエラーコード

一部の例外を除いて、サービスコールおよびコールバックの返値は、処理が正常終了したかを表す符号付き整数とする。処理が正常終了した場合には、E\_OK (=0) または正の値が返るものとし、値の意味はサービスコールまたはコールバック毎に定める。処理が正常終了しなかった場合には、その原因を表す負の値が返る。処理が正常終了しなかった原因を表す値を、エラーコード (error code) と呼ぶ。

エラーコードは、いずれも負の値のメインエラーコードとサブエラーコードで構成される。メインエラーコードとサブエラーコードからエラーコードを構成するマクロと、エラーコードからメインエラーコードとサブエラーコードを抽出するマクロが用意されている (「2.14.5 TOPPERS共通マクロ」の節を参照)。

メインエラーコードの名称・意味・値は、カーネルとシステムサービスで共通に定める (「2.14.4 TOPPERS共通エラーコード」の節を参照)。サービスコールおよびコールバックの機能説明中の「E\_XXXXエラーとなる」または「E\_XXXXエラーが返る」という記述は、メインエラーコードとしてE\_XXXXが返ることを意味する。

サブエラーコードは、エラーの原因をより詳細に表すために用いる。カーネルはサブエラーコードを使用せず、サブエラーコードとして常に-1が返る。サブエラーコードの名称・意味・値は、サブエラーコードを使用するシステムサービスのAPI仕様において規定する。

サービスコールが負の値のエラーコード (警告を表すものを除く) を返した場合には、サービスコールによる副作用がないのが原則である。ただし、そのような実装ができない場合にはこの原則の例外とし、サービスコールの機能説明にその旨を記述する。

サービスコールが複数のエラーを検出するべき状況では、その内のいずれか1つのエラーを示すエラーコードが返る。コールバックが複数のエラーを検出するべき状況では、その内のいずれか1つのエラーを示すエラーコードを返せばよい。

なお、静的APIは返値を持たない。静的APIの処理でエラーが検出された場合の扱いについては、「2.12.5 コンフィギュレータの処理モデル」の節および「2.12.6 静的APIのパラメータに関するエラー検出」の節を参照すること。



## 2.2.4 機能コード

ソフトウェア割込みによりサービスコールを呼び出す場合などに用いるためのサービスコールを識別するための番号を、機能コード (function code) と呼ぶ。機能コードは符号付きの整数値とし、カーネルのサービスコールには負の値を割り付け、拡張サービスコールには正の値を用いる。

## 2.2.5 ヘッダファイル

カーネルやシステムサービスを用いるために必要な定義を含むファイル。

### 【補足説明】

ヘッダファイルは、複数回インクルードしてもエラーにならないように対処するのが原則である。具体的には、ヘッダファイルの先頭で特定の識別子 (例えば、kernel.hなら"TOPPERS\_KERNEL\_H") をマクロ定義し、ヘッダファイルの内容全体をその識別子が定義されていない場合のみ有効とする条件ディレクティブを付加する。

## 2.3 主な概念

### 2.3.1 オブジェクトと処理単位

#### (1) オブジェクト

カーネルまたはシステムサービスが管理対象とするソフトウェア資源を、オブジェクト (object) と呼ぶ。特に、カーネルが管理対象とするソフトウェア資源を、カーネルオブジェクト (kernel object) と呼ぶ。

オブジェクトは、種類毎に、番号によって識別する。カーネルまたはシステムサービスで、オブジェクトに対して任意に識別番号を付与できる場合には、1から連続する正の整数値でオブジェクトを識別する。この場合に、オブジェクトの識別番号を、オブジェクトのID番号 (ID number) と呼ぶ。そうでない場合、すなわちカーネルまたはシステムサービスの内部または外部からの条件によって識別番号が決まる場合には、オブジェクトの識別番号を、オブジェクト番号 (object number) と呼ぶ。識別する必要のないオブジェクトには、識別番号を付与しない場合がある。

オブジェクト属性 (object attribute) は、オブジェクトの動作モードや初期状態を定めるもので、オブジェクトの登録時に指定する。オブジェクト属性にTA\_XXXXが指定されている場合、そのオブジェクトを、TA\_XXXX属性のオブジェクトと呼ぶ。複数の属性を指定する場合には、オブジェクト属性を渡すパラメータに、指定する属性値のビット毎論理和 (C言語の"|") を渡す。また、指定すべきオブジェクト属性がない場合には、TA\_NULLを指定する。

#### (2) 処理単位

オブジェクトの中には、プログラムが対応付けられるものがある。プログラムが対応付けられるオブジェクト (または、対応付けられるプログラム) を、処理単位 (processing unit) と呼ぶ。処理単位に対応付けられるプログラムは、アプリケーションまたはシステムサービスで用意し、カーネルが実行制御する。

処理単位の実行を要求することを起動 (activate)、処理単位の実行を開始することを実行開始 (start) と呼ぶ。

拡張情報 (extended information) は、処理単位が呼び出される時にパラメータとして渡される情報で、処理単位の登録時に指定する。拡張情報は、カーネ

ルやシステムサービスの動作には影響しない。

### (3) タスク

カーネルが実行順序を制御するプログラムの並行実行の単位をタスク (task) と呼ぶ。タスクは、処理単位の1つである。

サービスコールの機能説明において、サービスコールを呼び出したタスクを、自タスク (invoking task) と呼ぶ。

### (4) ディスパッチとスケジューリング

プロセッサが実行するタスクを切り換えることを、タスクディスパッチまたは単にディスパッチ (dispatching) と呼ぶ。それに対して、次に実行すべきタスクを決定する処理を、タスクスケジューリングまたは単にスケジューリング (scheduling) と呼ぶ。

ディスパッチが起こるべき状態 (すなわち、スケジューリングによって、現在実行しているタスクとは異なるタスクが、実行すべきタスクに決定されている状態) となっても、何らかの理由でディスパッチを行わないことを、ディスパッチの保留 (pend dispatching) という。ディスパッチを行わない理由が解除された時点で、ディスパッチが起こる。

### (5) 割り込みとCPU例外

プロセッサが実行中の命令とは独立に発生するイベントによって起動される例外処理のことを、外部割り込みまたは単に割り込み (interrupt) と呼ぶ。それに対して、プロセッサが実行中の命令に依存して起動される例外処理を、CPU例外 (CPU exception) と呼ぶ。

周辺デバイスからの割り込み要求をプロセッサに伝える経路を遮断し、割り込み要求が受け付けられるのを抑止することを、割り込みのマスク (mask interrupt) または割り込みの禁止 (disable interrupt) という。マスクが解除された時点で、まだ割り込み要求が保持されていれば、その時点で割り込み要求を受け付ける。

マスクすることができない割り込みを、NMI (non-maskable interrupt) と呼ぶ。

#### 【μITRON4.0仕様との関係】

μITRON4.0仕様において、未定義のまま使われていた割り込みとCPU例外という用語を定義した。

### (6) タイムイベントとタイムイベントハンドラ

時間の経過をきっかけに発生するイベントをタイムイベント (time event) と呼ぶ。タイムイベントにより起動され、カーネルが実行制御する処理単位を、タイムイベントハンドラ (time event handler) と呼ぶ。

## 2.3.2 サービスコールとパラメータ

### (1) 優先順位と優先度

優先順位 (precedence) とは、処理単位の実行順序を説明するための仕様上の概念である。複数の処理単位が実行できる場合には、その中で最も優先順位の高い処理単位が実行される。同じ優先順位を持つ処理単位の間では、後に起動された処理単位が先に実行される。

優先度 (priority) は、タスクなどの処理単位の優先順位や、メッセージなど

の配送順序を決定するために、アプリケーションが処理単位やメッセージなどに与える値である。優先度は、符号付きの整数型であるPRI型で表し、値が小さいほど優先度が高い（すなわち、先に処理または配送される）ものとする。優先度には、1から連続した正の値を用いるのを原則とする。

## (2) システム時刻と相対時間

カーネルが管理する時刻を、システム時刻 (system time) と呼ぶ。システム時刻は、符号無しの整数型であるSYSTIM型で表し、単位はミリ秒とする。

イベントを発生させるシステム時刻を絶対値で指定する方法は用意されておらず、一般には、基準となるシステム時刻（典型的には、サービスコールを呼び出したシステム時刻）からの相対時間 (relative time) によって指定する。相対時間は、符号無しの整数型であるRELTIM型で表し、単位はシステム時刻と同一、すなわちミリ秒とする。

相対時間には、少なくとも、16ビットの符号無しの整数型 (uint16\_t型) に格納できる任意の値を指定することができるが、RELTIM型 (uint\_t型に定義される) に格納できる任意の値を指定できるとは限らない。相対時間に指定できる最大値は、構成マクロTMAX\_RELTIMに定義されている。

イベントを発生させるシステム時刻を相対時間によって指定した場合、イベントの処理が行われるのは、基準時刻から相対時間によって指定した以上の時間が経過した後となる。

逆に、イベントの発生するシステム時刻を絶対値で参照する方法は用意されておらず、一般には、基準となるシステム時刻（典型的には、サービスコールを呼び出したシステム時刻）からの相対時間によって参照する。

イベントを発生させるシステム時刻が相対時間によって返された場合、イベントの処理が行われるのは、基準時刻から相対時間として返された以上の時間が経過した後となる。

### 【補足説明】

相対時間に0を指定した場合、基準時刻後の最初のタイムティックでイベントの処理が行われる。また、1を指定した場合、基準時刻後の2回目のタイムティックでイベントの処理が行われる。これは、基準時刻後の最初のタイムティックは、基準時刻の直後に発生する可能性があるため、ここでイベントの処理を行うと、基準時刻からの経過時間が1以上という仕様を満たせないためである。

同様に、相対時間として0が返された場合、基準時刻後の最初のタイムティックでイベントの処理が行われる。また、1が返された場合、基準時刻後の2回目のタイムティックでイベントの処理が行われる。

## (3) タイムアウトとポーリング

サービスコールの中で待ち状態が指定した時間以上継続した場合に、サービスコールの処理を取りやめて、サービスコールからリターンすることを、タイムアウト (timeout) という。タイムアウトしたサービスコールからは、E\_TMOUTエラーが返る。

タイムアウトを起こすまでの時間 (タイムアウト時間) は、符号付きの整数型であるTMO型で表し、単位はシステム時刻と同一、すなわちミリ秒とする。タイムアウト時間に正の値を指定した場合には、タイムアウトを起こすまでの相対時間を表す。すなわち、タイムアウトの処理が行われるのは、サービスコールを呼び出してから指定した以上の時間が経過した後となる。

ポーリング (polling) を行うサービスコールとは、サービスコールの中で待ち状態に遷移すべき状況になった場合に、サービスコールの処理を取りやめてリターンするサービスコールのことをいう。ここで、サービスコールの処理を取りやめてリターンすることを、ポーリングに失敗したという。ポーリングに失敗したサービスコールからは、E\_TMOUTエラーが返る。

ポーリングを行うサービスコールでは、待ち状態に遷移することはないのが原則である。そのため、ポーリングを行うサービスコールは、ディスパッチ保留状態であっても呼び出すことができる。ただし、サービスコールの中で待ち状態に遷移する状況が複数ある場合、ある状況でポーリング動作をしても、他の状況では待ち状態に遷移する場合がある。このような場合の振舞いは、該当するサービスコール毎に規定する。

タイムアウト付きのサービスコールは、別に規定がない限りは、タイムアウト時間にTMO\_POL (= 0) を指定した場合にはポーリングを行い、TMO\_FEVR (= -1) を指定した場合にはタイムアウトを起こさないものとする。

エラーコードに関する原則により、サービスコールがタイムアウトした場合やポーリングに失敗した場合には、サービスコールによる副作用がないのが原則である。ただし、そのような実装ができない場合にはこの原則の例外とし、どのような副作用があるかをサービスコール毎に規定する。

#### 【補足説明】

タイムアウト付きのサービスコールを、タイムアウト時間をTMO\_POLとして呼び出した場合には、ディスパッチ保留状態で呼び出すとE\_CTXエラーとなることを除いては、ポーリングを行うサービスコールと同じ振舞いをする。また、タイムアウト時間をTMO\_FEVRとして呼び出した場合には、タイムアウトなしのサービスコールと全く同じ振舞いをする。

#### 【仕様決定の理由】

ディスパッチ保留状態において、ポーリングを行うサービスコールを呼び出すことができるのに対して、タイムアウト付きのサービスコールをタイムアウト時間をTMO\_POLとして呼び出すとエラーになるのは、ディスパッチ保留状態では、別に規定がない限り、自タスクを広義の待ち状態に遷移させる可能性のあるサービスコール (タイムアウト付きのサービスコールはこれに該当) を呼び出すことはできないと規定されているためである。

#### (4) ノンブロッキング

サービスコールの中で待ち状態に遷移すべき状況になった時、サービスコールの処理を継続したままサービスコールからリターンする場合、そのサービスコールをノンブロッキング (non-blocking) という。処理を継続したままリターンする場合、サービスコールからはE\_WBLKエラーが返る。E\_WBLKは警告を表すエラーコードであり、サービスコールによる副作用がないという原則は適用されない。

サービスコールからE\_WBLKエラーが返った場合には、サービスコールの処理は継続しているため、サービスコールに渡したパラメータまたはリターンパラメータを入れる領域はまだ参照される可能性があり、別の目的に使用することはできない。継続している処理が完了した場合や、何らかの理由で処理が取りやめられた場合には、コールバックを呼び出すなどの方法で、サービスコールを呼び出したソフトウェアに通知するものとする。

ノンブロッキングの指定は、タイムアウト時間にTMO\_NBLK (= -2) を指定することによって行う。ノンブロッキングの指定を行えるサービスコールは、指定した場合の振舞いをサービスコール毎に規定する。

## 【補足説明】

ノンブロッキングは、システムサービスでサポートすることを想定した機能である。カーネルは、ノンブロッキングの指定を行えるサービスコールをサポートしていない。

### 2.3.3 保護機能

この節では、保護機能に関連する主な概念について説明する。この節の内容は、保護機能対応カーネルにのみ適用される。

#### (1) アクセス保護

保護機能対応カーネルは、処理単位が、許可されたカーネルオブジェクトに対して、許可された種別のアクセスを行うことのみを許し、それ以外のアクセスを防ぐアクセス保護機能を提供する。

アクセス制御の用語では、処理単位が主体 (subject)、カーネルオブジェクトが対象 (object) ということになる。

#### (2) メモリオブジェクト

保護機能対応カーネルにおいては、メモリ領域をカーネルオブジェクトとして扱い、アクセス保護の対象とする。カーネルがアクセス保護の対象とする一連のメモリ領域を、メモリオブジェクト (memory object) と呼ぶ。

メモリオブジェクトは、その先頭番地により識別する。言い換えると、先頭番地がオブジェクト番号となる。

#### (3) 保護ドメイン

保護機能を提供するために用いるカーネルオブジェクトの集合を、保護ドメイン (protection domain) を呼ぶ。

カーネルオブジェクトは、ただか1つの保護ドメインに属する。処理単位は、いずれか1つの保護ドメインに属さなければならないのに対して、それ以外のカーネルオブジェクトは、いずれの保護ドメインにも属さないことができる。

いずれの保護ドメインにも属さないカーネルオブジェクトを、無所属のカーネルオブジェクト (independent kernel object) と呼ぶ。

#### (4) カーネルドメインとユーザドメイン

システムには、カーネルドメイン (kernel domain) と呼ばれる保護ドメインが1つ存在する。カーネルドメインに属する処理単位は、プロセッサの特権モードで実行され、すべてのカーネルオブジェクトに対して、すべての種別のアクセスを行うことが許可される。

カーネルドメイン以外の保護ドメインを、ユーザドメイン (user domain) と呼ぶ。ユーザドメインに属する処理単位は、プロセッサの非特権モードで実行され、カーネルオブジェクトに対するアクセスに制限がある。

保護機能対応でないカーネルは、カーネルドメインのみをサポートしていると見なすこともできる。

## 【μITRON4.0/PX仕様との関係】

μITRON4.0/PX仕様のシステムドメイン (system domain) は、サポートしないこととした。システムドメインは、それに属する処理単位が、プロセッサの非特権モードで実行され、カーネルオブジェクトに対するアクセスに制限がある保護ドメインである。

#### (5) アクセス許可パターンとアクセス許可ベクタ

あるカーネルオブジェクトに対するある種別のアクセスが、どの保護ドメインに属する処理単位に許可されているかを表現するビットパターンを、アクセス許可パターン (access permission pattern) と呼ぶ。アクセス許可パターンの各ビットは、1つの保護ドメインに対応する。

アクセス許可パターンは、符号無し整数に定義されるデータ型 (ACPTN) で保持する。そのため、2つのアクセス許可パターンのビット毎論理和 (C言語の "|") を求めることで、アクセスを許可されている保護ドメインの和集合 (union) を得ることができる。また、2つのアクセス許可パターンのビット毎論理積 (C言語の "&") を求めることで、アクセスを許可されている保護ドメインの積集合 (intersection) を得ることができる。

カーネルオブジェクトに対するアクセスは、カーネルオブジェクトの種類毎に、通常操作1、通常操作2、管理操作、参照操作の4つの種別に分類されている。あるカーネルオブジェクトに対する4つの種別のアクセスに関するアクセス許可パターンをひとまとめにしたものを、アクセス許可ベクタ (access permission vector) と呼ぶ。

#### 2.3.4 マルチプロセッサ対応

この節では、マルチプロセッサ対応に関連する主な概念について説明する。この節の内容は、マルチプロセッサ対応カーネルにのみ適用される。

##### (1) クラス

マルチプロセッサに対応するために用いるカーネルオブジェクトの集合を、クラス (class) と呼ぶ。

カーネルオブジェクトは、いずれか1つのクラスに属する。カーネルオブジェクトの属するクラスは、オブジェクトの登録時に決定し、動的に変更することはできない。

マルチプロセッサ対応でないカーネルは、1つのクラスのみをサポートしていると見なすこともできる。

#### 2.4 処理単位の種類と実行順序

##### 2.4.1 処理単位の種類

カーネルが実行を制御する処理単位の種類は次の通りである。

- (a) タスク
  - (a.1) タスク例外処理ルーチン
- (b) 割込みハンドラ
  - (b.1) 割込みサービスルーチン
  - (b.2) タイムイベントハンドラ
- (c) CPU例外ハンドラ
- (d) 拡張サービスコールルーチン
- (e) 初期化ルーチン
- (f) 終了処理ルーチン

ここで、タイムイベントハンドラとは、時間の経過をきっかけに起動される処理単位である周期ハンドラ、アラームハンドラ、オーバランハンドラの総称である。

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、オーバランハンドラと拡張サービスコールルーチンをサポートしていない。

#### 2.4.2 処理単位の実行順序

処理単位の実行順序を規定するために、ここでは、処理単位の優先順位を規定する。また、ディスパッチが起こるタイミングを規定するために、ディスパッチを行うカーネル内の処理であるディスパッチャの優先順位についても規定する。

タスクの優先順位は、ディスパッチャの優先順位よりも低い。タスク間では、高い優先度を持つ方が優先順位が高く、同じ優先度を持つタスク間では、先に実行できる状態となった方が優先順位が高い。詳しくは、「2.6.3 タスクのスケジューリング規則」の節を参照すること。

タスク例外処理ルーチンの優先順位は、例外が要求されたタスクと同じであるが、タスクよりも先に実行される。

割込みハンドラの優先順位は、ディスパッチャの優先順位よりも高い。割込みハンドラ間では、高い割込み優先度を持つ方が優先順位が高く、同じ割込み優先度を持つ割込みハンドラ間では、先に実行開始された方が優先順位が高い。同じ割込み優先度を持つ割込みハンドラ間での実行開始順序は、この仕様では規定しない。詳しくは、「2.7.2 割込み優先度」の節を参照すること。

割込みサービスルーチンとタイムイベントハンドラの優先順位は、それを呼び出す割込みハンドラと同じである。

CPU例外ハンドラの優先順位は、CPU例外がタスクまたはタスク例外処理ルーチンで発生した場合には、ディスパッチャの優先順位と同じであるが、ディスパッチャよりも先に実行される。CPU例外がその他の処理単位で発生した場合には、CPU例外ハンドラの優先順位は、その処理単位の優先順位と同じであるが、その処理単位よりも先に実行される。

拡張サービスコールルーチンの優先順位は、それを呼び出した処理単位と同じであるが、それを呼び出した処理単位よりも先に実行される。

初期化ルーチンは、カーネルの動作開始前に、システムコンフィギュレーションファイル中に初期化ルーチンを登録する静的APIを記述したのと同じ順序で実行される。終了処理ルーチンは、カーネルの動作終了後に、終了処理ルーチンを登録する静的APIを記述したのと逆の順序で実行される。

#### 2.4.3 カーネル処理の不可分性

カーネルのサービスコール処理やディスパッチャ、割込みハンドラとCPU例外ハンドラの出入口処理などのカーネル処理は不可分に実行されるのが基本である。実際には、カーネル処理の途中でアプリケーションが実行されることは許されており、アプリケーションがサービスコールを用いて観測できる範囲で、カーネル処理が不可分に実行された場合と同様に振る舞えばよい。これを、カーネル処理の不可分性という。

#### 【補足説明】

1つのサービスコールにより複数のタスクが実行できる状態になる場合、新しく実行状態となるべきタスクへのディスパッチは、すべてのタスクの状態遷移が完了した後に行われる。例えば、低優先度のタスクAが発行したサービスコールにより、中優先度のタスクBと高優先度のタスクCがこの順で待ち解除される場合、タスクBとタスクCが待ち解除された後に、タスクCへのディスパッチが行われる。

## 2.5 システム状態とコンテキスト

### 2.5.1 カーネル動作状態と非動作状態

カーネルの初期化が完了した後、カーネルの終了処理が開始されるまでの間を、カーネル動作状態と呼ぶ。それ以外の状態、すなわちカーネルの初期化完了前（初期化ルーチンの実行中を含む）と終了処理開始後（終了処理ルーチンの実行中を含む）を、カーネル非動作状態と呼ぶ。

カーネル非動作状態では、原則として、NMIを除くすべての割り込みがマスクされる。

カーネル非動作状態では、システムインタフェースレイヤのAPIとsns\_ker（カーネル非動作状態の参照）のみを呼び出すことができ、その他のサービスコールを呼び出すことはできない。カーネル非動作状態で、その他のサービスコールを呼び出した場合の動作は、保証されない。

### 2.5.2 タスクコンテキストと非タスクコンテキスト

処理単位が実行される環境（用いるスタック領域やプロセッサの動作モードなど）をコンテキストと呼ぶ。

カーネル動作状態において、処理単位が実行されるコンテキストは、タスクコンテキストと非タスクコンテキストに分類される。

タスク（タスク例外処理ルーチンを含む）が実行されるコンテキストは、タスクコンテキストに分類される。また、タスクコンテキストから呼び出した拡張サービスコールルーチンが実行されるコンテキストは、タスクコンテキストに分類される。

割り込みハンドラ（割り込みサービスルーチンおよびタイムイベントハンドラを含む）とCPU例外ハンドラが実行されるコンテキストは、非タスクコンテキストに分類される。また、非タスクコンテキストから呼び出した拡張サービスコールルーチンが実行されるコンテキストは、非タスクコンテキストに分類される。

タスクコンテキストからは、非タスクコンテキスト専用のサービスコールを呼び出すことはできない。逆に、非タスクコンテキストからは、タスクコンテキスト専用のサービスコールを呼び出すことはできない。いずれも、呼び出した場合にはE\_CTXエラーとなる。

### 2.5.3 カーネルの振舞いに影響を与える状態

カーネル動作状態において、プロセッサは、カーネルの振舞いに影響を与える状態として、次の状態を持つ。

- ・全割り込みロックフラグ（全割り込みロック状態と全割り込みロック解除状態）
- ・CPUロックフラグ（CPUロック状態とCPUロック解除状態）
- ・割り込み優先度マスク（割り込み優先度マスク全解除状態と全解除でない状態）
- ・ディスパッチ禁止フラグ（ディスパッチ禁止状態とディスパッチ許可状態）

これらの状態は、それぞれ独立な状態である。すなわち、プロセッサは上記の



状態の任意の組合せを取ることができ、それぞれの状態を独立に変化させることができる。

#### 2.5.4 全割込みロック状態と全割込みロック解除状態

プロセッサは、NMIを除くすべての割込みをマスクするための全割込みロックフラグを持つ。全割込みロックフラグがセットされた状態を全割込みロック状態、クリアされた状態を全割込みロック解除状態と呼ぶ。すなわち、全割込みロック状態では、NMIを除くすべての割込みがマスクされる。

全割込みロック状態では、システムインタフェースレイヤのAPIとsns\_ker（カーネル非動作状態の参照）、ext\_ker（カーネルの終了）のみを呼び出すことができ、その他のサービスコールを呼び出すことはできない。全割込みロック状態で、その他のサービスコールを呼び出した場合の動作は、保証されない。また、全割込みロック状態では、実行中の処理単位からリターンしてはならない。リターンした場合の動作は保証されない。

#### 2.5.5 CPUロック状態とCPUロック解除状態

プロセッサは、カーネル管理の割込み（「2.5.9 カーネル管理外の割込み」の節を参照）をすべてマスクするためのCPUロックフラグを持つ。CPUロックフラグがセットされた状態をCPUロック状態、クリアされた状態をCPUロック解除状態と呼ぶ。すなわち、CPUロック状態では、すべてのカーネル管理の割込みがマスクされる。

NMI以外にカーネル管理外の割込みを設けない場合には、全割込みロックフラグとCPUロックフラグの機能は同一となるが、両フラグは独立に存在する。

CPUロック状態で呼び出すことができるサービスコールは次の通り。

- ・システムインタフェースレイヤのAPI
- ・loc\_cpu / iloc\_cpu, unl\_cpu / iunl\_cpu
- ・dis\_int, ena\_int, chg\_ipm, get\_ipm
- ・sns\_yyy, xsns\_yyy（CPU例外ハンドラからのみ）、get\_utm
- ・ext\_tsk, ext\_ker

CPUロック状態で、その他のサービスコールを呼び出した場合には、E\_CTXエラーとなる。

#### 2.5.6 割込み優先度マスク

プロセッサは、割込み優先度を基準に割込みをマスクするための割込み優先度マスクを持つ。割込み優先度マスクがTIPM\_ENAALL（=0）の時は、いずれの割込み要求もマスクされない。この状態を割込み優先度マスク全解除状態と呼ぶ。割込み優先度マスクがTIPM\_ENAALL（=0）以外の時は、割込み優先度マスクと同じかそれより低い割込み優先度を持つ割込みはマスクされ、ディスパッチは保留される。この状態を割込み優先度マスクが全解除でない状態と呼ぶ。

割込み優先度マスクが全解除でない状態では、別に規定がない限りは、自タスクを広義の待ち状態に遷移させる可能性のあるサービスコールを呼び出すことはできない。呼び出した場合には、E\_CTXエラーとなる。

#### 2.5.7 ディスパッチ禁止状態とディスパッチ許可状態

プロセッサは、ディスパッチを保留するためのディスパッチ禁止フラグを持つ。ディスパッチ禁止フラグがセットされた状態をディスパッチ禁止状態、クリアされた状態をディスパッチ許可状態と呼ぶ。すなわち、ディスパッチ禁止状態では、ディスパッチは保留される。

ディスパッチ禁止状態では、別に規定がない限りは、自タスクを広義の待ち状態に遷移させる可能性のあるサービスコールを呼び出すことはできない。呼び出した場合には、E\_CTXエラーとなる。

#### 2.5.8 ディスパッチ保留状態

非タスクコンテキストの実行中、全割込みロック状態、CPUロック状態、割込み優先度マスクが全解除でない状態、ディスパッチ禁止状態では、ディスパッチが保留される。これらの状態を総称して、ディスパッチ保留状態と呼ぶ。

#### 2.5.9 カーネル管理外の割込み

高い割込み応答性を求められるアプリケーションでは、カーネル内で割込みをマスクすることにより、割込み応答性の要求を満たせなくなる場合がある。このような要求に対応するために、カーネル内では、ある割込み優先度よりも高い割込み優先度を持つ割込みをマスクしないこととしている。言い換えると、カーネルは、ある割込み優先度以下の（それと同じかより低い）割込み優先度を持つ割込みのみを管理し、それよりも高い割込み優先度を持つ割込みは管理しない。

カーネルが管理する割込みをカーネル管理の割込み、カーネルが管理しない割込みをカーネル管理外の割込みと呼ぶ。NMIは、カーネル管理外の割込みとして扱う。カーネル管理外の割込みの設定方法については、「2.7.6 カーネル管理外の割込みの設定方法」の節を参照すること。

カーネル管理外の割込みによって起動される割込みハンドラを、カーネル管理外の割込みハンドラと呼ぶ。カーネル管理外の割込みハンドラからは、システムインタフェースレイヤのAPIとsns\_ker（カーネル非動作状態の参照）、ext\_ker（カーネルの終了）のみを呼び出すことができ、その他のサービスコールを呼び出すことはできない。カーネル管理外の割込みハンドラから、その他のサービスコールを呼び出した場合の動作は、保証されない。

#### 【補足説明】

カーネル管理外の割込みには、カーネル内の割込み出入口処理を経由せずに割込みハンドラを実行できるという意義も考えられるが、この仕様では、カーネル内でマスクされないという意義のみを実現するものと位置づけている。これは、すべての割込みが同じアドレスに分岐するプロセッサでは、カーネル内の割込み出入口処理を経由せずに割込みハンドラを実行することができるとは限らないためである。

カーネル内の割込み出入口処理を経由せずに割込みハンドラを実行する仕組みについては、割込みハンドラの直接呼出しと呼び、ターゲット依存でサポートしてもよいこととする。

#### 2.5.10 カーネル管理外のCPU例外

カーネル内のクリティカルセクションの実行中（これを、カーネル実行中と呼ぶ）、全割込みロック状態、CPUロック状態、カーネル管理外の割込みハンドラ実行中のいずれかで発生したCPU例外を、カーネル管理外のCPU例外と呼ぶ。また、それによって起動されるCPU例外ハンドラを、カーネル管理外のCPU例外ハンドラと呼ぶ。さらに、カーネル管理外のCPU例外ハンドラ実行中に発生したCPU例外も、カーネル管理外のCPU例外に含める。

カーネル管理外のCPU例外ハンドラからは、システムインタフェースレイヤのAPIとsns\_ker（カーネル非動作状態の参照）、ext\_ker（カーネルの終了）、xsns\_dpn（CPU例外発生時のディスパッチ保留状態の参照）、xsns\_xpn（CPU例

外発生時のタスク例外処理保留状態の参照)のみを呼び出すことができ、その他のサービスコールを呼び出すことはできない。カーネル管理外のCPU例外ハンドラから、その他のサービスコールを呼び出した場合の動作は、保証されない。

#### 【補足説明】

カーネル管理外のCPU例外は、カーネル管理外の割込みと異なり、特定のCPU例外をカーネル外とするわけではない。同じCPU例外であっても、CPU例外が起こる状況によって、カーネル管理となる場合とカーネル管理外となる場合がある。

#### 2.5.11 カーネル管理外の状態

全割込みロック状態、カーネル管理外の割込みハンドラ実行中、カーネル管理外のCPU例外ハンドラ実行中を総称して、カーネル管理外の状態と呼ぶ。

すでに述べた通り、カーネル管理外の状態では、いくつかの例外的なサービスコールを除いては、システムインタフェースレイヤのAPIのみを呼び出すことができる。カーネル管理外の状態から、その他のサービスコールを呼び出した場合の動作は、保証されない。

カーネル管理外の状態では、少なくとも、カーネル管理の割込みはマスクされている。カーネル管理外の割込み(の一部)もマスクされている場合もある。保護機能を持ったカーネルでは、カーネル管理外の状態になるのは、特権モードで実行している間に限られる。

#### 2.5.12 処理単位とシステム状態

各処理単位が実行開始されるシステム状態の条件(実行開始条件)、各処理単位の実行開始時にカーネルによって行われるシステム状態の変更処理(実行開始時処理)、各処理単位からのリターン前(または終了前)にアプリケーションが設定しておくべきシステム状態(リターン前または終了前)、各処理単位からのリターン時(または終了時)にカーネルによって行われるシステム状態の変更処理(リターン時処理または終了時処理)は、次の表の通りである。

	CPUロック フラグ	割込み優先度 マスク	ディスパッチ 禁止フラグ	タスク例外 禁止フラグ
<b>【タスク】</b>				
実行開始条件	解除	全解除	許可	-
実行開始時処理	そのまま	そのまま	そのまま	禁止する(*7)
終了前	原則解除(*1)	原則全解除(*1)	原則許可(*1)	任意
終了時処理	解除する	全解除する	許可する	-
<b>【タスク例外処理ルーチン】</b>				
実行開始条件	解除	任意	任意	許可
実行開始時処理	そのまま	そのまま	そのまま	禁止する
リターン前	原則解除(*1)	原則元に(*1)	原則元に(*1)	原則禁止(*1)
リターン時処理	解除する(*4)	元に戻す(*4)	元に戻す(*4)	許可する
<b>【割込みハンドラ、割込みサービスルーチン、タイムイベントハンドラ】</b>				
実行開始条件	解除	自優先度より低い	任意	任意
実行開始時処理	そのまま	自優先度に(*2)	そのまま	そのまま
リターン前	原則解除(*1)	変更不可(*3)	変更不可(*3)	変更不可(*3)
リターン時処理	解除する	元に戻す(*5)	そのまま	そのまま
<b>【CPU例外ハンドラ】</b>				
実行開始条件	任意	任意	任意	任意
実行開始時処理	そのまま(*6)	そのまま	そのまま	そのまま

リターン前	原則元>(*1)	変更不可(*3)	変更不可(*3)	変更不可(*3)
リターン時処理	元に戻す	元に戻す(*5)	そのまま	そのまま

---

【拡張サービスコールルーチン】

実行開始条件	任意	任意	任意	任意
実行開始時処理	そのまま	そのまま	そのまま	そのまま
リターン前	任意	任意	任意	任意
リターン時処理	そのまま	そのまま	そのまま	そのまま

---

この表の中で「原則(\*1)」とは、処理単位からのリターン前(または終了前)に、アプリケーションが指定された状態に設定しておくことが原則であるが、この原則に従わなくても、リターン時(または終了時)にカーネルによって状態が設定されるため、支障がないことを意味する。

「自優先度(\*2)」とは、割り込みハンドラと割り込みサービスルーチンの場合にはそれを要求した割り込みの割り込み優先度、周期ハンドラとアラームハンドラの場合にはタイマ割り込みの割り込み優先度、オーバーランハンドラの場合にはオーバーランタイマ割り込みの割り込み優先度に変更することを意味する。

「変更不可(\*3)」とは、その処理単位中で、そのシステム状態を変更するAPIが用意されていないことを示す。

保護機能対応カーネルでは、タスク例外処理ルーチンからのリターン時にカーネルによって行われるシステム状態の変更処理(\*4)は、タスクにそれぞれの状態変更を許可している場合にのみ行われる。ここでカーネルは、元の状態に戻す処理を実現するために、元の状態をユーザスタック上に保存する。アプリケーションがユーザスタック上に保存された状態を書き換えた場合、タスク例外処理ルーチンからのリターン時に、書き換えた状態に変更される(すなわち、元に戻されるとは限らない)。

タスク実行開始時のタスク例外禁止フラグの変更処理(\*7)は、正確には、タスク起動時に行われる処理である。タスク例外禁止フラグは、タスクが休止状態の時(すなわち、タスクの起動前と終了後)は無効である。

【仕様決定の理由】

保護機能対応カーネルにおいて、タスク例外処理ルーチンからのリターン時のシステム状態の変更処理(\*4)が、タスクに状態変更を許可している場合にのみ行われるのは、タスクがユーザスタック上の状態を書き換えることで、許可していない状態変更を起こしてしまうことを防止するためである。タスクに状態変更を許可していない場合には、タスク例外処理ルーチン中で状態を変更できないため、カーネルによって元の状態に戻す必要がない。

割り込みハンドラやCPU例外ハンドラで、その処理単位中で割り込み優先度マスクを変更するAPIが用意されていないにもかかわらず、処理単位からのリターン時に元の状態に戻す(\*5)のは、プロセッサによっては、割り込み優先度マスクがステータスレジスタ等に含まれており、APIを用いずに変更できてしまう場合があるためである。

CPU例外ハンドラの実行開始時には、CPUロックフラグは変更されない(\*6)ことから、CPUロック状態でCPU例外が発生した場合、CPU例外ハンドラの実行開始直後はCPUロック状態となっている。この場合、xsns\_dpn, xsns\_xpnともtrueを返すために、CPU例外ハンドラ中でCPUロック状態を解除することができない。ただし、規則を無視してCPUロック状態を解除しようとした場合も考えられるため、リターン時には元に戻すこととしている。

## 2.6 タスクの状態遷移とスケジューリング規則

### 2.6.1 基本的なタスク状態

カーネルに登録したタスクは、実行できる状態、休止状態、広義の待ち状態のいずれかの状態を取る。また、実行できる状態と広義の待ち状態を総称して、起動された状態と呼ぶ。さらに、タスクをカーネルに登録していない仮想的な状態を、未登録状態と呼ぶ。

#### (a) 実行できる状態 (runnable)

タスクを実行できる条件が、プロセッサが使用できるかどうかを除いて、揃っている状態。実行できる状態は、さらに、実行状態と実行可能状態に分類される。

##### (a.1) 実行状態 (running)

タスクが実行されている状態。または、そのタスクの実行中に、割り込みまたはCPU例外により非タスクコンテキストの実行が開始され、かつ、タスクコンテキストに戻った後に、そのタスクの実行を再開するという状態。

##### (a.2) 実行可能状態 (ready)

タスク自身は実行できる状態にあるが、それよりも優先順位の高いタスクが実行状態にあるために、そのタスクが実行されない状態。

#### (b) 休止状態 (dormant)

タスクが実行すべき処理がない状態。タスクの実行を終了した後、次に起動するまでの間は、タスクは休止状態となっている。タスクが休止状態にある時には、タスクの実行を再開するための情報（実行再開番地やレジスタの内容など）は保存されていない。

#### (c) 広義の待ち状態 (blocked)

タスクが、処理の途中で実行を止められている状態。タスクが広義の待ち状態にある時には、タスクの実行を再開するための情報（実行再開番地やレジスタの内容など）は保存されており、タスクが実行を再開する時には、広義の待ち状態に遷移する前の状態に戻される。広義の待ち状態は、さらに、（狭義の）待ち状態、強制待ち状態、二重待ち状態に分類される。

##### (c.1) （狭義の）待ち状態 (waiting)

タスクが何らかの条件が揃うのを待つために、自ら実行を止めている状態。

##### (c.2) 強制待ち状態 (suspended)

他のタスクによって、強制的に実行を止められている状態。ただし、自タスクを強制待ち状態にすることも可能である。

##### (c.3) 二重待ち状態 (waiting-suspended)

待ち状態と強制待ち状態が重なった状態。すなわち、タスクが何らかの条件が揃うのを待つために自ら実行を止めている時に、他のタスクによって強制的に実行を止められている状態。

単にタスクが「待ち状態である」といった場合には、二重待ち状態である場合を含み、「待ち状態でない」といった場合には、二重待ち状態でもないことを意味する。また、単にタスクが「強制待ち状態である」といった場合には、二

重待ち状態である場合を含み、「強制待ち状態でない」といった場合には、二重待ち状態でもないことを意味する。

(d) 未登録状態 (non-existent)

タスクをカーネルに登録していない仮想的な状態。タスクの生成前と削除後は、タスクは未登録状態にあるとみなす。

カーネルによっては、これらのタスク状態以外に、過渡的な状態が存在する場合がある。過渡的な状態については、「2.6.6 ディスパッチ保留状態で実行中のタスクに対する強制待ち」の節を参照すること。

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、タスクが未登録状態になることはない。また、上記のタスク状態以外の過渡的な状態になることもない。

### 2.6.2 タスクの状態遷移

タスクの状態遷移を図2-2に示す。

未登録状態のタスクをカーネルに登録することを、タスクを生成する (create) という。生成されたタスクは、休止状態に遷移する。また、タスク生成時の属性指定により、生成と同時にタスクを起動し、実行できる状態にすることもできる。逆に、登録されたタスクを未登録状態に遷移させることを、タスクを削除する (delete) という。

休止状態のタスクを、実行できる状態にすることを、タスクを起動する (activate) という。起動されたタスクは、実行できる状態になる。逆に、起動された状態のタスクを、休止状態 (または未登録状態) に遷移させることを、タスクを終了する (terminate) という。

実行できる状態になったタスクは、まずは実行可能状態に遷移するが、そのタスクの優先順位が実行状態のタスクよりも高い場合には、ディスパッチ保留状態でない限りはただちにディスパッチが起こり、実行状態へ遷移する。この時、それまで実行状態であったタスクは実行可能状態に遷移する。この時、実行状態に遷移したタスクは、実行可能状態に遷移したタスクをプリエンプトしたという。逆に、実行可能状態に遷移したタスクは、プリエンプトされたという。

タスクを待ち解除するとは、タスクが待ち状態 (二重待ち状態を除く) であれば実行できる状態に、二重待ち状態であれば強制待ち状態に遷移させることをいう。また、タスクを強制待ちから再開するとは、タスクが強制待ち状態 (二重待ち状態を除く) であれば実行できる状態に、二重待ち状態であれば待ち状態に遷移させることをいう。

### 2.6.3 タスクのスケジューリング規則

実行できるタスクは、優先順位の高いものから順に実行される。すなわち、ディスパッチ保留状態でない限りは、実行できるタスクの中で最も高い優先順位を持つタスクが実行状態となり、他は実行可能状態となる。

タスクの優先順位は、タスクの優先度とタスクが実行できる状態になった順序から、次のように定まる。優先度の異なるタスクの間では、優先度の高いタスクが高い優先順位を持つ。優先度が同一のタスクの間では、先に実行できる状態になったタスクが高い優先順位を持つ。すなわち、同じ優先度を持つタスクは、FCFS (First Come First Served) 方式でスケジューリングされる。ただし、サービスクールの呼出しにより、同じ優先度を持つタスク間の優先順位を変更することも可能である。

最も高い優先順位を持つタスクが変化した場合には、ディスパッチ保留状態でない限りはただちにディスパッチが起こり、最も高い優先順位を持つタスクが実行状態となる。ディスパッチ保留状態においては、実行状態のタスクは切り換わらず、最も高い優先順位を持つタスクは実行可能状態にとどまる。

#### 2.6.4 待ち行列と待ち解除の順序

タスクが待ち解除される順序の管理のために、待ち状態のタスクがつながれているキューを、待ち行列と呼ぶ。

待ち行列にタスクをつなぐ順序には、FIFO順とタスクの優先度順がある。どちらの順序でつなぐかは、待ち行列毎に規定される。多くの待ち行列において、どちらの順序でつなぐかを、オブジェクト属性により指定できる。

FIFO順の待ち行列においては、新たに待ち状態に遷移したタスクは待ち行列の最後につながれる。それに対してタスクの優先度順の待ち行列においては、新たに待ち状態に遷移したタスクは、優先度の高い順に待ち行列につながれる。同じ優先度のタスクが待ち行列につながれている場合には、新たに待ち状態に遷移したタスクが、同じ優先度のタスクの中で最後につながれる。

待ち解除の条件がタスクによって異なる場合には、待ち行列の先頭のタスクは待ち解除の条件を満たさないが、後方のタスクが待ち解除の条件を満たす場合がある。このような場合の振舞いとして、次の2つのケースがある。どちらの振舞いをするかは、待ち行列毎に規定される。

(a) 待ち解除の条件を満たしたタスクの中で、待ち行列の前方につながれたものから順に待ち解除される。すなわち、待ち行列の前方に待ち解除の条件を満たさないタスクがあっても、後方のタスクが待ち解除の条件を満たしていれば、先に待ち解除される。

(b) タスクの待ち解除は、待ち行列につながれている順序で行われる。すなわち、待ち行列の前方に待ち解除の条件を満たさないタスクがあると、後方のタスクが待ち解除の条件を満たしても、待ち解除されない。

ここで、(b)の振舞いをする待ち行列においては、待ち行列につながれたタスクの強制終了、タスク優先度の変更（待ち行列がタスクの優先度順の場合のみ）、待ち状態の強制解除が行われた場合に、タスクの待ち解除が起こることがある。具体的には、これらの操作により新たに待ち行列の先頭になったタスクが、待ち解除の条件を満たしていれば、ただちに待ち解除される。さらに、この待ち解除により新たに待ち行列の先頭になったタスクに対しても、同じ処理が繰り返される。

#### 2.6.5 待ち禁止状態

拡張サービスコールを実行中のタスクは、基本的なタスク状態と重複して、待ち禁止状態になることができる。

待ち禁止状態とは、タスクが待ち状態に入ることが一時的に禁止された状態である。待ち禁止状態にあるタスクが、サービスコールを呼び出して待ち状態に遷移しようとした場合、サービスコールはE\_RLWAIエラーとなる。

タスクを待ち禁止状態に遷移させるサービスコールは、対象タスクが拡張サービスコールを実行中の場合に、対象タスクを待ち禁止状態に遷移させる。一方、そのタスクが次にタスク自身のコンテキスト（拡張サービスコールルーチンのコンテキストを除く）を実行する時点で、待ち禁止状態が解除される。また、タスクの待ち禁止状態を解除するサービスコールによっても、待ち禁止状態を解除することができる。

## 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルは、拡張サービスコールをサポートしておらず、タスクを待ち禁止状態に遷移させるサービスコールもサポートしていないため、タスクが待ち禁止状態になることはない。

## 【仕様決定の理由】

対象タスクが待ち状態にある場合でも、タスク例外処理ルーチンをすみやかに実行させたい状況を考える。拡張サービスコールを用いない場合には、タスク例外処理の要求に加えて、待ち状態の強制解除を行うことで、これを実現することができる。それに対して拡張サービスコールを用いる場合には、対象タスクが拡張サービスコールルーチンを実行している途中で、待ち状態の強制解除を発行し、その直後に対象タスクが待ち状態に入ると、タスク例外処理ルーチンがすみやかに実行されない（拡張サービスコールルーチンの実行中はタスク例外処理ルーチンが起動されないため）。

待ち禁止状態は、この問題を解決するために導入したものである。タスク例外処理の要求に加えて、待ち状態の強制解除と待ち禁止状態への遷移を行うことで、対象タスクが待ち状態にある場合でも、タスク例外処理ルーチンをすみやかに実行させることができる。

【 $\mu$ ITRON4.0仕様、 $\mu$ ITRON4.0/PX仕様との関係】

待ち禁止状態は、 $\mu$ ITRON4.0仕様にはない概念であり、 $\mu$ ITRON4.0/PX仕様で導入された。ただし、 $\mu$ ITRON4.0/PX仕様では、タスクの待ち状態を強制解除するサービスコールが、タスクを待ち禁止状態へ遷移させる機能も持つこととしている。その結果 $\mu$ ITRON4.0/PX仕様は、待ち状態を強制解除するサービスコールの仕様において、 $\mu$ ITRON4.0仕様との互換性がなくなっている。

この仕様では、待ち状態の強制解除と待ち禁止状態への遷移を別々のサービスコールで行うこととした。これにより、待ち状態を強制解除するサービスコールの仕様が、 $\mu$ ITRON4.0仕様と互換になっている。一方、 $\mu$ ITRON4.0/PX仕様とは互換性がない。

## 2.6.6 ディスパッチ保留状態で実行中のタスクに対する強制待ち

ディスパッチ保留状態において、実行状態のタスクを強制待ち状態へ遷移させるサービスコールを呼び出した場合、実行状態のタスクの切換えは、ディスパッチ保留状態が解除されるまで保留される。

この間、それまで実行状態であったタスクは、実行状態と強制待ち状態の間の過渡的な状態にあると考える。この状態を、強制待ち状態 [ 実行継続中 ] と呼ぶ。一方、ディスパッチ保留状態が解除された後に実行すべきタスクは、実行可能状態にとどまる。

タスクが強制待ち状態 [ 実行継続中 ] にある時に、ディスパッチ保留状態が解除されると、ただちにディスパッチが起こり、タスクは強制待ち状態に遷移する。

過渡的な状態も含めたタスクの状態遷移を図2-3に示す。

タスクが強制待ち状態 [ 実行継続中 ] である時の具体的な扱いは、必要になった時に規定するものとする。

なお、TOPPERS新世代カーネルでは、ディスパッチ保留状態において、実行状態のタスクを強制終了させるサービスコールはサポートしていない。



## 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、ディスパッチ保留状態において実行状態のタスクを強制待ち状態へ遷移させるサービスコールはサポートしていないため、タスクが強制待ち状態 [ 実行継続中 ] になることはない。

### 2.7 割込み処理モデル

TOPPERS新世代カーネルにおける割込み処理のモデルは、TOPPERS標準割込み処理モデルに準拠している。

TOPPERS標準割込み処理モデルの概念図を図2-4に示す。この図は、割込み処理モデルの持つすべての機能が、ハードウェア（プロセッサおよびIRC）で実現されているとして描いた概念図である。実際のハードウェアで不足している機能については、カーネル内の割込み処理のソフトウェアで実現される。

#### 2.7.1 割込み処理の流れ

周辺デバイス（以下、デバイスと呼ぶ）からの割込み要求は、割込みコントローラ（IRC）を経由して、プロセッサに伝えられる。デバイスからIRCに割込み要求を伝えるための信号線を、割込み要求ラインと呼ぶ。一般には、1つの割込み要求ラインに、複数のデバイスからの割込み要求が接続される。

プロセッサは、デバイスからの割込み要求を受け付ける条件が満たされた場合、割込み要求を受け付ける。割込み要求を受け付けると、カーネル内の割込み出入口処理を経由して、カーネル内の割込みハンドラを実行する。

カーネル内の割込みハンドラは、アプリケーションが割込み要求ラインに対して登録した割込みサービスルーチン（ISR）を呼び出す。割込みサービスルーチンは、プロセッサ（およびIRC）の割込みアーキテクチャに依存せず、割込みを要求したデバイスのみ依存して記述するのが原則である。1つの割込み要求ラインに対して複数の周辺デバイスが接続されることから、1つの割込み要求ラインに対して複数の割込みサービスルーチンを登録することができる。

ただし、カーネルが標準的に用意している割込みハンドラで対応できない特殊なケースも考えられる。このような場合に対応するために、アプリケーションが用意した割込みハンドラをカーネルに登録することもできる。

カーネルが用いるタイマデバイスからの割込み要求の場合、カーネル内の割込みハンドラが、タイムイベントの処理を行う。具体的には、タイムアウト処理等を行うことに加えて、アプリケーションが登録したタイムイベントハンドラを呼び出す。

カーネル管理外の割込みを受け付けた場合の振舞いは、ターゲット定義である。カーネル管理外の割込みに対して、割込みハンドラを登録できるかは、ターゲット定義である。割込みサービスルーチンを登録することはできない。

#### 2.7.2 割込み優先度

割込み要求は、割込み処理の優先順位を指定するための割込み優先度を持つ。プロセッサは、割込み優先度マスクの現在値よりも高い割込み優先度を持つ割込み要求のみを受け付ける。逆に言うと、割込み優先度マスクの現在値と同じか、それより低い割込み優先度を持つ割込みは、マスクされる。

プロセッサは、割込み要求を受け付けると、割込み優先度マスクを、受け付けた割込み要求の割込み優先度に設定する（ただし、受け付けた割込みがNMIである場合には例外とする）。また、割込み処理からのリターンにより、割込み優

先度マスクを、割り込み要求を受け付ける前の値に戻す。

これらのことから、基本的には、ある割り込み要求の処理中は、それ以下の（それと同じかより低い）割り込み優先度を持つ割り込み要求は受け付けられず、それより高い割り込み優先度を持つ割り込み要求は受け付けられることになる。つまり、割り込み優先度は、多重割り込みを制御するためのものと位置付けることができる。それに対して、同時に発生している割り込み要求の中で、割り込み優先度の高い割り込み要求が先に受け付けられるとは限らない。

割り込み優先度は、PRI型で表現し、値が小さいほど優先度が高いものとするが、優先度に関する原則には従わず、-1から連続した負の値を用いる。これは、割り込み優先度とタスク優先度を比較できるようになることと、いずれの割り込みもマスクしない割り込み優先度マスクの値を0にできるためである。

割り込み優先度の段階数は、ターゲット定義である。プロセッサが割り込み優先度マスクを実現するための機能を持たないか、実現するために大きいオーバーヘッドを生じる場合には、ターゲット定義で、割り込み優先度の段階数を1にする（すなわち、多重割り込みを許さない）場合がある。

カーネルが管理する割り込みの中の割り込み優先度の最高値（数としては最小値）を、TMIN\_INTPRIと書く。すなわち、カーネル管理外の割り込みは、TMIN\_INTPRIよりも高い割り込み優先度を持つ。TMIN\_INTPRIを固定するか設定できるようにするか、設定できるようにする場合の設定方法は、ターゲット定義である。

### 2.7.3 割り込み要求ラインの属性

各割り込み要求ラインは、以下の属性を持つ。なお、1つの割り込み要求ラインに複数のデバイスからの割り込み要求が接続されている場合、それらの割り込み要求は同一の属性を持つ。それらの割り込み要求に別々の属性を設定することはできない。

#### (1) 割り込み要求禁止フラグ

割り込み要求ライン毎に、割り込みをマスクするための割り込み要求禁止フラグを持つ。割り込み要求禁止フラグをセットすると、その割り込み要求処理ラインにより伝えられる割り込み要求はマスクされる。

プロセッサが割り込み要求禁止フラグを実現するための機能を持たないか、実現するために大きいオーバーヘッドを生じる場合には、ターゲット定義で、割り込み禁止フラグをサポートしない場合がある。また、プロセッサの持つ割り込み要求禁止フラグの機能がこの仕様に合致しない場合には、ターゲット定義で、割り込み禁止フラグをサポートしないか、振舞いが異なるものとする場合がある。

アプリケーションが、割り込み要求禁止フラグを動的にセット/クリアする機能を用いると、次の理由でソフトウェアの再利用性が下がる可能性があるため、注意が必要である。プロセッサによっては（この割り込み処理モデルに合致した）割り込み要求禁止フラグの機能を実現できない場合がある。また、割り込み要求禁止フラグを設定することで複数のデバイスからの割り込みがマスクされる場合がある。ソフトウェアの再利用性を上げるためには、あるデバイスからの割り込みのみをマスクしたい場合には、できる限り、そのデバイス自身の機能を使ってマスクを実現すべきである。

#### (2) 割り込み優先度

割り込み要求ライン毎に、割り込み優先度を設定することができる。割り込み要求の割り込み優先度とは、その割り込み要求を伝える割り込み要求ラインに対して設定された割り込み優先度のことである。

### (3) トリガモード

割込み要求ラインに対する割込み要求が、レベルトリガであるかエッジトリガであるかを設定することができる。エッジトリガの場合には、さらに、ポジティブエッジかネガティブエッジか両エッジトリガかを設定できる場合もある。また、レベルトリガの場合には、ローレベルトリガかハイレベルトリガかを設定できる場合もある。

プロセッサがトリガモードを設定するための機能を持たないか、設定するために大きいオーバヘッドを生じる場合には、ターゲット定義で、トリガモードの設定をサポートしない場合がある。

#### 2.7.4 割込みを受け付ける条件

NMI以外の割込み要求は、次の4つの条件が揃った場合に受け付けられる。

- (a) 割込み要求ラインに対する割込み要求禁止フラグがクリアされていること
- (b) 割込み要求ラインに設定された割込み優先度が、割込み優先度マスクの現在値よりも高い（優先度の値としては小さい）こと
- (c) 全割込みロックフラグがクリアされていること
- (d) 割込み要求がカーネル管理外の割込みでない場合には、CPUロックフラグがクリアされていること

これらの条件が揃った割込み要求が複数ある場合に、どの割込み要求が最初に受け付けられるかは、この仕様では規定しない。すなわち、割込み優先度の高い割込み要求が先に受け付けられるとは限らない。

#### 2.7.5 割込み番号と割込みハンドラ番号

割込みサービスルーチンの登録対象となる割込み要求ラインを識別するための番号を、割込み番号と呼ぶ。割込み番号は、符号無しの整数型であるINTNO型で表し、プロセッサの仕様から決まる自然な方法を取ることを基本とする。そのため、1から連続した正の値であるとは限らない。

それに対して、アプリケーションが用意した割込みハンドラをカーネルに登録する場合に、割込みハンドラの登録対象となる割込みを識別するための番号を、割込みハンドラ番号と呼ぶ。割込みハンドラ番号は、符号無しの整数型であるINHNO型で表し、割込み番号と1対1に対応するのが基本である（両者が一致する場合が多い）。

ただし、割込みを要求したデバイスが割込みベクタを生成してプロセッサに渡すアーキテクチャなどでは、割込み番号と割込みハンドラ番号が1対1を対応せさない方が都合が良い場合がある。そこで、ターゲット定義で、割込み番号に対応しない割込みハンドラ番号や、割込みハンドラ番号に対応しない割込み番号を設ける場合もある。

#### 2.7.6 カーネル管理外の割込みの設定方法

NMI以外にカーネル管理外の割込みを設けるか（設けられるようにするか）どうかは、ターゲット定義である。具体的には、次の3つの方法のいずれかが採用される。

- (a-1) NMI以外にカーネル管理外の割込みを設けない
- (a-2) カーネル構築時に特定の割込みをカーネル管理外にすると決める

これら場合には、カーネル管理外とする割込みはカーネル構築時（ターゲット依存部の実装時やカーネルのコンパイル時）に決まるため、カーネル管理外とする割込みをアプリケーション側で設定する必要はない。ここで、カーネル管理外とされた割込みに対して、カーネルのAPIにより割込みハンドラを登録できるかや、割込み要求ラインの属性を設定できるかは、ターゲット定義である。割込みハンドラを登録できる場合には、それを定義するAPIにおいて、カーネル管理外であることを示す割込みハンドラ属性（TA\_NONKERNEL）を指定する。また、割込み要求ラインの属性を設定できる場合には、設定する割込み優先度をTMIN\_INTPRIよりも高い値とする。

(b) カーネル管理外とする割込みをアプリケーションで設定できるようにする

この場合には、カーネル管理外とする割込みの設定は、次の方法で行う。まず、カーネル管理外の割込みハンドラを定義するAPIにおいて、カーネル管理外であることを示す割込みハンドラ属性（TA\_NONKERNEL）を指定する。また、カーネル管理外とする割込みの割込み要求ラインに対して設定する割込み優先度を、TMIN\_INTPRIよりも高い値とする。

いずれの場合にも、カーネル管理の割込みの割込み要求ラインに対して設定する割込み優先度は、TMIN\_INTPRIより高い値であってはならない。

## 2.8 CPU例外処理モデル

プロセッサが検出するCPU例外の種類や、CPU例外検出時のプロセッサの振舞いは、プロセッサによって大きく異なる。そのため、CPU例外ハンドラをターゲットハードウェアに依存せずに記述することは、少なくとも現時点では困難である。そこでこの仕様では、CPU例外の処理モデルを厳密に標準化するのではなく、ターゲットハードウェアに依存せずに決められる範囲で規定する。

### 2.8.1 CPU例外処理の流れ

アプリケーションは、プロセッサが検出するCPU例外の種類毎に、CPU例外ハンドラを登録することができる。プロセッサがCPU例外を検出すると、カーネル内のCPU例外出入口処理を経由して、検出したCPU例外に対して登録したCPU例外ハンドラが呼び出される。

CPU例外ハンドラの登録対象となるCPU例外を識別するための番号を、CPU例外ハンドラ番号と呼ぶ。CPU例外ハンドラ番号は、符号無しの整数型であるEXCNO型で表し、ターゲットハードウェアの仕様から決まる自然な方法を取ることを基本とする。そのため、1から連続した正の値であるとは限らない。

CPU例外ハンドラにおいては、CPU例外が発生した状態からのリカバリ処理を行う。どのようなリカバリ処理を行うかは、一般にはCPU例外の種類やそれが発生したコンテキストおよび状態に依存するが、大きく次の4つの方法が考えられる。

(a) カーネルに依存しない形でCPU例外の原因を取り除き、実行を継続する。

(b) CPU例外を起こしたタスクよりも優先度の高いタスクを起動または待ち解除し、そのタスクでリカバリ処理を行う（例えば、CPU例外を起こしたタスクを強制終了し、再度起動する）。ただし、CPU例外を起こしたタスクが最高優先度の場合には、この方法でリカバリ処理を行うことはできない。

(c) CPU例外を起こしたタスクにタスク例外処理を要求し、タスク例外処理ルーチンでリカバリ処理を行う（例えば、CPU例外を起こしたタスクを終了する）。

(d) システム全体に対してリカバリ処理を行う（例えば、システムを再起動する）。

この中で(a)と(d)の方法は、カーネルの機能を必要としないため、CPU例外が発生したコンテキストおよび状態に依存せずに常に行える。それに対して(b)と(c)の方法は、CPU例外ハンドラからそのためのサービスコールを呼び出せることが必要であり、それが行えるかどうかは、CPU例外が発生したコンテキストおよび状態に依存する。

### 2.8.2 CPU例外ハンドラから呼び出せるサービスコール

CPU例外ハンドラからは、CPU例外発生時のディスパッチ保留状態を参照するサービスコール(xsns\_dpn)と、CPU例外発生時のタスク例外処理保留状態を参照するサービスコール(xsns\_xpn)を呼び出すことができる。

xsns\_dpnは、CPU例外がタスクコンテキストで発生し、そのタスクがディスパッチできる状態である場合にfalseを返す。xsns\_dpnがfalseを返した場合、そのCPU例外ハンドラから、非タスクコンテキストから呼び出せるすべてのサービスコールを呼び出すことができ、(b)の方法によるリカバリ処理が可能である。ただし、CPU例外を起こしたタスクが最高優先度の場合には、この方法でリカバリ処理を行うことはできない。

xsns\_xpnは、CPU例外がタスクコンテキストで発生し、そのタスクがタスク例外処理ルーチンを実行できる状態である場合にfalseを返す。xsns\_xpnがfalseを返した場合、そのCPU例外ハンドラから、非タスクコンテキストから呼び出せるすべてのサービスコールを呼び出すことができ、(c)の方法によるリカバリ処理が可能である。

xsns\_dpnとxsns\_xpnのいずれのサービスコールもtrueを返した場合、そのCPU例外ハンドラからは、xsns\_dpnとxsns\_xpnに加えて、システムインタフェースレイヤのAPIとsns\_ker(カーネル非動作状態の参照)、ext\_ker(カーネルの終了)のみを呼び出すことができ、その他のサービスコールを呼び出すことはできない。いずれのサービスコールもtrueを返したにもかかわらず、その他のサービスコールを呼び出した場合の動作は、保証されない。この場合には、(b)と(c)の方法によるリカバリ処理は行うことはできず、(a)または(d)の方法によるリカバリ処理を行うしかないことになる。

カーネル管理外のCPU例外ハンドラにおいては、xsns\_dpnとxsns\_xpnのいずれのサービスコールもtrueを返す。

ターゲットシステムによっては、xsns\_xpnがfalseを返すべき状態であっても、それを正確に判断するために大きいオーバーヘッドがかかる場合には、安全側に倒してtrueを返す場合がある。具体的には、CPU例外がタスクコンテキストで発生し、そのタスクがタスク例外処理ルーチンを実行できる状態であっても、そのタスクが割り込み優先度マスクをTMIN\_INTPRIまたはそれよりも高い値に設定している場合には、trueを返す場合がある。

## 2.9 システム初期化手順

システムのリセット後、最初に行うプログラムを、スタートアップモジュールと呼ぶ。スタートアップモジュールはカーネルの管理外であり、アプリケーションで用意するのが基本であるが、スタートアップモジュールで行うべき処理を明確にするために、カーネルの配布パッケージの中に、標準のスタートアップモジュールが用意されている。

標準のスタートアップモジュールは、プロセッサのモードとスタックポインタ等の初期化、NMIを除くすべての割り込みのマスク(全割り込みロック状態と同等の状態にする)、ターゲットシステム依存の初期化フックの呼出し、BSSセクションのクリア、DATAセクションの初期化、ソフトウェア環境(ライブラリなど)依存の初期化フックの呼出しを行った後、カーネルの初期化処理へ分岐する。ここで呼び出すターゲットシステム依存の初期化フックでは、リセット後に速

やかに行うべき初期化処理を行うことが想定されている。

カーネルの初期化処理においては、まず、カーネル自身の初期化処理（カーネル内のデータ構造の初期化、カーネルが用いるデバイスの初期化など）と静的APIの処理（オブジェクトの登録など）が行われる。静的APIのパラメータに関するエラーは、コンフィギュレータによって検出されるのが原則であるが、コンフィギュレータで検出できないエラーが、この処理中に検出される場合もある。

タスクの起動順序など、静的APIの処理順序によりシステムの規定された振舞いに変化する場合には、システムコンフィギュレーションファイルにおける静的APIの記述順と同じ順序で静的APIが処理された場合と、同じ振舞いとなる。それに対して、周期ハンドラの動作開始順序は、同じタイムティックで行うべき処理が複数ある場合の処理順序が規定されないことから（「4.6.1 システム時刻管理」の節を参照）、このことが適用されない。

次に、静的API（ATT\_INI）により登録した初期化ルーチンが、システムコンフィギュレーションファイルにおける静的APIの記述順と同じ順序で実行される。

以上が終了すると、カーネル非動作状態から動作状態に遷移し（「2.5.1 カーネル動作状態と非動作状態」の節を参照）、カーネルの動作が開始される。具体的には、割込みがマスク解除され、タスクの実行が開始される。

#### 【μITRON4.0仕様との関係】

μITRON4.0仕様においては、初期化ルーチンの実行は静的APIの処理に含まれるものとしていたが、この仕様では、初期化ルーチンを登録する静的APIの処理には、初期化ルーチンを登録することのみを意味し、初期化ルーチンの実行は含まれないものとした。

## 2.10 システム終了手順

カーネルを終了させるサービスコール（ext\_ker）を呼び出すと、カーネル動作状態から非動作状態に遷移する（「2.5.1 カーネル動作状態と非動作状態」の節を参照）。具体的には、NMIを除くすべての割込みがマスクされ、タスクの実行が停止される。

次に、静的API（ATT\_TER）により登録した終了処理ルーチンが、システムコンフィギュレーションファイルにおける静的APIの記述順と逆の順序で実行される。また、ソフトウェア環境（ライブラリなど）依存の終了処理を行うために、atexitで登録された関数が呼び出される。

以上が終了すると、ターゲットシステム依存の終了処理が呼び出される。ターゲットシステム依存の終了処理は、カーネルの管理外であり、アプリケーションで用意するのが基本であるが、カーネルの配布パッケージの中に、ターゲットシステム毎に標準的なルーチンが用意されている。

#### 【μITRON4.0仕様との関係】

μITRON4.0仕様には、システム終了に関する規定はない。

## 2.11 オブジェクトの登録とその解除

### 2.11.1 ID番号で識別するオブジェクト

ID番号で識別するオブジェクトは、オブジェクトを生成する静的API（CRE\_YYY）、サービスコール（acre\_yyy）、またはオブジェクトを追加する静的API（ATT\_YYY）によってカーネルに登録する。オブジェクトを追加する

静的APIによって登録されたオブジェクトはID番号を持たないため、ID番号を指定して操作することができない。

オブジェクトを生成する静的API (CRE\_YYY) は、生成するオブジェクトにID番号を割り付け、ID番号を指定するパラメータとして記述した識別名を、割り付けたID番号にマクロ定義する。同じ識別名のオブジェクトが生成済みの場合には、E\_OBJエラーとなる。

オブジェクトを生成するサービスコール (acre\_yyy) は、割付け可能なID番号の数を指定する静的API (AID\_YYY) によって確保されたID番号の中から、使用されていないID番号を1つ選び、生成するオブジェクトに割り付ける。割り付けたID番号は、サービスコールの返値としてアプリケーションに通知する。使用されていないID番号が残っていない場合には、E\_NOIDエラーとなる。

オブジェクトを生成するサービスコールによって登録したオブジェクトは、オブジェクトを削除するサービスコール (del\_yyy) によって登録を解除することができる。登録解除したオブジェクトのID番号は、未使用の状態に戻され、そのID番号を用いて新しいオブジェクトを登録することができる。この場合に、登録解除前のオブジェクトに対して行うつもりが、新たに登録したオブジェクトに対して行われないように、注意が必要である。

オブジェクトを生成または追加する静的APIによって登録したオブジェクトは、登録を解除することができない。登録を解除しようとした場合には、E\_OBJエラーとなる。

同期・通信オブジェクトを削除した時に、そのオブジェクトを待っているタスクがあった場合、それらのタスクは待ち解除され、待ち状態に遷移させたサービスコールはE\_DLTエラーとなる。複数のタスクが待ち解除される場合には、待ち行列につながれていた順序で待ち解除される。削除した同期・通信オブジェクトが複数の待ち行列を持つ場合には、別の待ち行列で待っていたタスクの間の待ち解除の順序は、該当するサービスコール毎に規定する。

オブジェクトを再初期化するサービスコール (ini\_yyy) は、指定したオブジェクトを削除した後に、同じパラメータで再度生成したのと等価の振舞いをする。ただし、オブジェクトを生成または追加する静的APIによって登録したオブジェクトも、再初期化することができる。

なお、動的生成対応カーネル以外では、オブジェクトを生成するサービスコール (acre\_yyy)、割付け可能なID番号の数を指定する静的API (AID\_YYY)、オブジェクトを削除するサービスコール (del\_yyy) は、サポートされない。

#### 【μITRON4.0仕様との関係】

ID番号を指定してオブジェクトを生成するサービスコール (cre\_yyy) を廃止した。また、オブジェクトを生成する静的APIによって登録したオブジェクトは、登録解除できないこととした。

μITRON4.0仕様では、割付け可能なID番号の数を指定する静的API (AID\_YYY) は規定されていない。

複数の待ち行列を持つ同期・通信オブジェクトを削除した時に、別の待ち行列で待っていたタスクの間の待ち解除の順序は、μITRON4.0仕様では実装依存とされている。

#### 【μITRON4.0/PX仕様との関係】

アクセス許可ベクタを指定してオブジェクトを登録する機能 (CRA\_YYY, cra\_yyy, acra\_yyy, ATA\_YYY, ata\_yyy) を廃止し、オブジェクトの登録後にア

クセス許可ベクタを設定する静的API (SAC\_YYY) をサポートすることとした。

#### 【仕様決定の理由】

ID番号を指定してオブジェクトを生成するサービスコール (cre\_yyy) とアクセス許可ベクタを指定してオブジェクトを登録するサービスコール (cra\_yyy, acra\_yyy, ata\_yyy) を廃止したのは、必要性が低いと考えたためである。静的APIについても、サービスコールに整合するよう変更した。

#### 2.11.2 オブジェクト番号で識別するオブジェクト

オブジェクト番号で識別するオブジェクトは、オブジェクトを定義する静的API (DEF\_YYY) またはサービスコール (def\_yyy) によってカーネルに登録する。

オブジェクトを定義するサービスコール (def\_yyy) によって登録したオブジェクトは、同じサービスコールによって登録を解除することができる。登録解除したオブジェクト番号は、オブジェクト登録前の状態に戻され、同じオブジェクト番号に対して新たにオブジェクトを定義することができる。登録解除されていないオブジェクト番号に対して再度オブジェクトを登録しようとした場合には、E\_OBJエラーとなる。

オブジェクトを定義する静的APIによって登録したオブジェクトは、登録を解除することができない。登録を解除しようとした場合には、E\_OBJエラーとなる。

なお、動的生成対応カーネル以外では、オブジェクトを定義するサービスコール (def\_yyy) はサポートされない。

#### 【μITRON4.0仕様との関係】

この仕様では、オブジェクトの定義を変更したい場合には、一度登録解除した後に、新たにオブジェクトを定義する必要がある。また、オブジェクトを定義する静的APIによって登録したオブジェクトは、この仕様では、登録解除できないこととした。

#### 2.11.3 オブジェクト生成に必要なメモリ領域

カーネルオブジェクトを生成する際に必要なメモリ領域の中で、サイズが変化するものについては、カーネルオブジェクトを生成する静的APIおよびサービスコールに、使用するメモリ領域の先頭番地を渡すパラメータを設けている。このパラメータをNULLとすることは、必要なメモリ領域をコンフィギュレータまたはカーネルが確保することを意味する。

未完成

#### 【未決定事項】

動的生成対応カーネルにおけるメモリ領域の確保方法、特に保護機能との関連については、今後の課題である。

### 2.12 システムコンフィギュレーション手順

#### 2.12.1 システムコンフィギュレーションファイル

カーネルやシステムサービスが管理するオブジェクトの生成情報や初期状態などを記述するファイルを、システムコンフィギュレーションファイル (system configuration file) と呼ぶ。また、システムコンフィギュレーションファイルを解釈して、カーネルやシステムサービスの構成・初期化情報を含むファイルなどを生成するツールを、コンフィギュレータ (configurator) と呼ぶ。



システムコンフィギュレーションファイルには、カーネルの静的API、システムサービスの静的API、コンフィギュレータに対するINCLUDEディレクティブ、C言語プリプロセッサのインクルードディレクティブ(#include)と条件ディレクティブ(#if, #ifdefなど)のみを記述することができる。

コンフィギュレータに対するINCLUDEディレクティブは、システムコンフィギュレーションファイルを複数のファイルに分割して記述するために用いるもので、その文法は次のいずれかである(両者の違いは、指定されたファイルを探すディレクトリの違いのみ)。

```
INCLUDE("ファイル名");
INCLUDE(<ファイル名>);
```

コンフィギュレータは、INCLUDEディレクティブによって指定されたファイル中の記述を、システムコンフィギュレーションファイルの一部として解釈する。すなわち、INCLUDEディレクティブによって指定されたファイル中には、カーネルの静的API、システムサービスの静的API、コンフィギュレータに対するINCLUDEディレクティブ、C言語プリプロセッサのインクルードディレクティブと条件ディレクティブのみを記述することができる。

C言語プリプロセッサのインクルードディレクティブは、静的APIのパラメータを解釈するために必要なC言語のヘッダファイルを指定するために用いる。また、条件ディレクティブは、有効とする静的APIを選択するために用いることができる。ただし、インクルードディレクティブは、コンフィギュレータが生成するファイルでは先頭に集められる。そのため、条件ディレクティブの中にインクルードディレクティブを記述しても、インクルードディレクティブは常に有効となる。また、1つの静的APIの記述の途中で、条件ディレクティブを記述することはできない。

#### 【μITRON4.0仕様との関係】

システムコンフィギュレーションファイルにおけるC言語プリプロセッサのディレクティブの扱いを全面的に見直し、コンフィギュレータに対するINCLUDEディレクティブを設けた。また、共通静的APIを廃止した。μITRON4.0仕様における#includeディレクティブの役割は、この仕様ではINCLUDEディレクティブに置き換わる。逆に、μITRON4.0仕様におけるINCLUDE静的APIの役割は、この仕様では#includeディレクティブに置き換わる。

#### 2.12.2 静的APIの文法とパラメータ

静的APIは、次に述べる例外を除いては、C言語の関数呼出しと同様の文法で記述する。すなわち、静的APIの名称に続けて、静的APIの各パラメータを","で区切って列挙したものを("("と")"で囲んで記述し、最後に";"を記述する。ただし、静的APIのパラメータに構造体(または構造体へのポインタ)を記述する場合には、構造体の各フィールドを","で区切って列挙したものを "{"と"}"で囲んだ形で記述する。

サービスコールに対応する静的APIの場合、静的APIのパラメータは、対応するサービスコールのパラメータと同一とすることを原則とする。

静的APIのパラメータは、次の3種類に分類される。

##### (a) オブジェクト識別名

オブジェクトのID番号を指定するパラメータ。オブジェクトの名称を表す単一の識別名のみを記述することができる。

コンフィギュレータは、オブジェクト生成のための静的API (CRE\_YYY) を処理する際に、オブジェクトにID番号を割り付け、構成・初期化ヘッダファイルに、指定された識別名を割り付けたID番号にマクロ定義するC言語プリプロセッサのディレクティブ (#define) を生成する。

オブジェクト生成以外の静的APIが、オブジェクトのID番号をパラメータに取る場合 (カーネルの静的APIでは、DEF\_TEXのtskidパラメータのみがこれに該当) には、パラメータとして記述する識別名は、生成済みのオブジェクトの名称を表す識別名でなければならない。そうでない場合には、コンフィギュレータがエラーを報告する。

静的APIの整数定数式パラメータの記述に、オブジェクト識別名を使用することはできない。

#### (b) 整数定数式パラメータ

オブジェクト番号や機能コード、オブジェクト属性、サイズや数、優先度など、整数値を指定するパラメータ。プログラムが配置されるアドレスに依存せずに値の決まる整数定数式を記述することができる。

#### (c) 一般定数式パラメータ

処理単位のエントリ番地、メモリ領域の先頭番地、拡張情報など、アドレスを指定する可能性のあるパラメータ。任意の定数式を記述することができる。

#### 【μITRON4.0仕様との関係】

μITRON4.0仕様においては、静的APIのパラメータを次の4種類に分類していたが、コンフィギュレータの仕組みを見直したことに伴い全面的に見直した。

- (A) 自動割付け対応整数値パラメータ
- (B) 自動割付け非対応整数値パラメータ
- (C) プリプロセッサ定数式パラメータ
- (D) 一般定数式パラメータ

TOPPERS新世代カーネル仕様の(a)が、おおよそμITRON4.0仕様の(A)に相当するが、(a)には整数値を記述できない点が異なる。(b)~(c)と(B)~(D)には、対応関係がない。

#### 2.12.3 保護ドメインの指定

保護機能対応カーネルでは、オブジェクトを登録する静的APIを、そのオブジェクトが属する保護ドメインの囲みの中に記述する。無所属のオブジェクトを登録する静的APIは、保護ドメインの囲みの外側に記述する。

#### 2.12.4 クラスの指定

マルチプロセッサ対応カーネルでは、オブジェクトを登録する静的APIを、そのオブジェクトが属するクラスの囲みの中に記述する。

なお、保護機能とマルチプロセッサの両方に対応するカーネルでは、保護ドメインの囲みとクラスの囲みはどちらが外側になってもよい。

#### 2.12.5 コンフィギュレータの処理モデル

コンフィギュレータは、次の3つのパスにより、システムコンフィギュレーションファイルを解釈し、構成・初期化情報を含むファイルなどを生成する (図 2-5)。

最初のパス1では、システムコンフィギュレーションファイルを解釈し、そこに含まれる静的APIの整数定数式パラメータの値をCコンパイラを用いて求めるために、パラメータ計算用C言語ファイル (cfg1\_out.c) を生成する。この時、システムコンフィギュレーションファイルに含まれるC言語プリプロセッサのインクルードディレクティブは、パラメータ計算用C言語ファイルの先頭に集めて生成する。また、条件ディレクティブは、順序も含めて、そのままの形でパラメータ計算用C言語ファイルに出力する。システムコンフィギュレーションファイルに文法エラーがあった場合には、この段階で検出される。

次に、Cコンパイラおよび関連ツールを用いて、パラメータ計算用C言語ファイルをコンパイルし、オブジェクトファイルを生成する。また、それをSレコードフォーマットの形 (cfg1\_out.srec) に変換し、オブジェクトファイル中の各シンボルとアドレスの対応表を含むシンボルファイル (cfg1\_out.syms) を生成する。静的APIのパラメータに解釈できない式が記述された場合には、この段階でエラーが検出される。

コンフィギュレータのパス2では、パス1で生成されたオブジェクトファイルをSレコードフォーマットの形に変換したものとシンボルファイルから、C言語プリプロセッサの条件ディレクティブによりどの静的APIが有効となったかと、それらの静的APIの整数定数式パラメータの値を取り出し、カーネルおよびシステムサービスの構成・初期化ファイル (kernel\_cfg.cなど) と構成・初期化ヘッダファイル (kernel\_cfg.hなど) を生成する。構成・初期化ヘッダファイルには、登録できるオブジェクトの数 (動的生成対応カーネル以外では、静的APIによって登録されたオブジェクトの数に一致) やオブジェクトのID番号などの定義を出力する。静的APIの整数定数式パラメータに不正がある場合には、この段階でエラーが検出される。

パス2で生成されたこれらのファイルを、他のソースファイルとあわせてコンパイルし、アプリケーションのロードモジュールを生成する。また、それをSレコードフォーマットの形 (system.srec) に変換し、ロードモジュール中の各シンボルとアドレスの対応表を含むシンボルファイル (system.syms) を生成する。

コンフィギュレータのパス3では、パス1で生成されたオブジェクトファイルをSレコードフォーマットの形に変換したものとシンボルファイル、パス2で生成されたロードモジュールをSレコードフォーマットの形に変換したものとシンボルファイルから、静的APIパラメータの値などを取り出し、妥当性のチェックを行う。静的APIの一般定数式パラメータに不正がある場合には、この段階でエラーが検出される。

#### 【 $\mu$ ITRON4.0仕様との関係】

コンフィギュレータの処理モデルは全面的に変更した。

#### 2.12.6 静的APIのパラメータに関するエラー検出

静的APIのパラメータに関するエラー検出は、同じものがサービスコールとして呼ばれた場合と同等とすることを原則とする。言い換えると、サービスコールによっても検出できないエラーは、静的APIにおいても検出しない。静的APIの機能説明中の「E\_XXXXXエラーとなる」または「E\_XXXXXエラーが返る」という記述は、コンフィギュレータがそのエラーを検出することを意味する。

ただし、エラーの種類によっては、サービスコールと同等のエラー検出を行うことが難しいため、そのようなものについては例外とする。例えば、メモリ不足をコンフィギュレータによって検出するのは容易ではない。

逆に、オブジェクト属性については、サービスコールより強力なエラーチェックを行える可能性がある。例えば、タスク属性にTA\_STAと記述されている場合、

サービスコールではエラーを検出できないが、コンフィギュレータでは検出できる可能性がある。ただし、このようなエラー検出を完全に行おうとするとコンフィギュレータが複雑になるため、このようなエラーを検出することは必須とせず、検出できた場合には警告として報告する。

#### 【μITRON4.0仕様との関係】

μITRON4.0仕様では、静的APIのパラメータに関するエラー検出について規定されていない。

### 2.12.7 オブジェクトのID番号の指定

コンフィギュレータのオプション機能として、アプリケーション設計者がオブジェクトのID番号を指定するための次の機能を用意する。

コンフィギュレータのオプション指定により、オブジェクト識別名とID番号の対応表を含むファイルを渡すと、コンフィギュレータはそれに従ってオブジェクトにID番号を割り付ける。それに従ったID番号割付けができない場合（ID番号に抜けができる場合など）には、コンフィギュレータはエラーを報告する。

またコンフィギュレータは、オプション指定により、オブジェクト識別名とコンフィギュレータが割り付けたID番号の対応表を含むファイルを、コンフィギュレータに渡すファイルと同じフォーマットで生成する。

#### 【μITRON4.0仕様との関係】

μITRON4.0仕様では、オブジェクト生成のための静的APIのID番号を指定するパラメータに整数値を記述できるため、このような機能は用意されていない。

### 2.13 TOPPERSネーミングコンベンション

この節では、TOPPERSソフトウェアのAPIの構成要素の名称に関するネーミングコンベンションについて述べる。このネーミングコンベンションは、モジュール間のインタフェースに関わる名称に適用することを想定しているが、モジュール内部の名称に適用してもよい。

#### 2.13.1 モジュール識別名

異なるモジュールのAPIの構成要素の名称が衝突することを避けるために、各モジュールに対して、それを識別するためのモジュール識別名を定める。モジュール識別名は、英文字と数字で構成し、2～8文字程度の長さとする。

カーネルのモジュール識別名は"kernel"、システムインタフェースレイヤのモジュール識別名は"sil"とする。

APIの構成要素の名称には、モジュール識別名を含めることを原則とするが、カーネルのAPIなど、頻繁に使用されて衝突のおそれが少ない場合には、モジュール識別名を含めない名称を使用する。

以下では、モジュール識別名の英文字を英小文字としたものをwww、英大文字としたものをWWWと表記する。

#### 2.13.2 データ型名

各サイズの整数型など、データの意味を定めない基本データ型の名称は、英小文字、数字、"\_"で構成する。データ型であることを明示するために、末尾が"\_t"である名称とする。

複合データ型やデータの意味を定めるデータ型の名称は、英大文字、数字、"\_"で構成する。データ型であることを明示するために、先頭が"T\_"または末尾が"\_T"である名称とする場合もある。

データ型の種類毎に、次のネーミングコンベンションを定める。

#### (A) パケットのデータ型

T_CYYY	acre_yyyに渡すパケットのデータ型
T_DYYY	def_yyyに渡すパケットのデータ型
T_RYYY	ref_yyyに渡すパケットのデータ型
T_WWW_CYYY	www_acre_yyyに渡すパケットのデータ型
T_WWW_DYYY	www_def_yyyに渡すパケットのデータ型
T_WWW_RYYY	www_ref_yyyに渡すパケットのデータ型

### 2.13.3 関数名

関数の名称は、英小文字、数字、"\_"で構成する。

関数の種類毎に、次のネーミングコンベンションを定める。

#### (A) サービスコール

サービスコールは、xxx\_yyyまたはwww\_xxx\_yyyの名称とする。ここで、xxxは操作の方法、yyyは操作の対象を表す。xxx\_yyyまたはwww\_xxx\_yyyから派生したサービスコールは、それぞれzxxx\_yyyまたはwww\_zxxx\_yyyの名称とする。ここでzは、派生したことを表す文字である。派生したことを表す文字を2つ付加する場合には、zzxxx\_yyyまたはwww\_zzxxx\_yyyの名称となる。

非タスクコンテキスト専用のサービスコールの名称は、派生したことを表す文字として"i"を付加し、ixxx\_yyy、izxxx\_yyy、www\_ixxx\_yyy、www\_izxxx\_yyyといった名称とする。

#### (B) コールバック

コールバックの名称は、サービスコールのネーミングコンベンションに従う。

### 2.13.4 変数名

変数（const修飾子のついたものを含む）の名称は、英小文字、数字、"\_"で構成する。データ型が異なる変数には、異なる名称を付けることを原則とする。

変数の名称に関して、次のガイドラインを設ける。

~ id	~ ID (オブジェクトのID番号, ID型)
~ no	~ 番号 (オブジェクト番号)
~ atr	~ 属性 (オブジェクト属性, ATR型)
~ stat	~ 状態 (オブジェクト状態, STAT型)
~ mode	~ モード (サービスコールの動作モード, MODE型)
~ pri	~ 優先度 (優先度, PRI型)
~ sz	~ サイズ (単位はバイト数, SIZE型またはuint_t型)
~ cnt	~ の個数 (単位は個数, uint_t型)
~ ptn	~ パターン
~ tim	~ 時刻, ~ 時間
~ cd	~ コード
i ~	~ の初期値
max ~	~ の最大値
min ~	~ の最小値

left ~        ~の残り

また、ポインタ変数（関数ポインタを除く）の名称に関して、次のガイドラインを設ける。

p\_~            ポインタ  
 pp\_~          ポインタを入れる領域へのポインタ  
 pk\_~          パケットへのポインタ  
 ppk\_~        パケットへのポインタを入れる領域へのポインタ

変数の種類毎に、次のネーミングコンベンションを定める。

#### (A) パケットへのポインタ

pk\_cyyy        acre\_yyyに渡すパケットへのポインタ  
 pk\_dyyy        def\_yyyに渡すパケットへのポインタ  
 pk\_ryyy        ref\_yyyに渡すパケットへのポインタ  
 pk\_www\_cyyy   www\_acre\_yyyに渡すパケットへのポインタ  
 pk\_www\_dyyy   www\_def\_yyyに渡すパケットへのポインタ  
 pk\_www\_ryyy   www\_ref\_yyyに渡すパケットへのポインタ

#### 2.13.5 定数名

定数（C言語プリプロセッサのマクロ定義によるもの）の名称は、英大文字、数字、"\_"で構成する。

定数の種類毎に、次のネーミングコンベンションを定める。

#### (A) メインエラーコード

メインエラーコードは、先頭が"E\_"である名称とする。

#### (B) 機能コード

TFN\_XXX\_YYY    xxx\_yyyの機能コード  
 TFN\_WWW\_XXX\_YYY   www\_xxx\_yyyの機能コード

#### (C) その他の定数

その他の定数は、先頭がTUU\_またはTUU\_www\_である名称とする。ここでUUは、定数の種類またはデータ型を表す。同じパラメータまたはリターンパラメータに用いられる定数の名称については、UUを同一にすることを原則とする。

また、定数の名称に関して、次のガイドラインを設ける。

TA\_~           オブジェクトの属性値  
 TSZ\_~          ~のサイズ  
 TBIT\_~        ~のビット数  
 TMAX\_~        ~の最大値  
 TMIN\_~        ~の最小値

#### 2.13.6 マクロ名

マクロ（C言語プリプロセッサのマクロ定義によるもの）の名称は、それが表す構成要素のネーミングコンベンションに従う。すなわち、関数を表すマクロは関数のネーミングコンベンションに、定数を表すマクロは定数のネーミングコンベンションに従う。ただし、簡単な関数を表すマクロや、副作用があるなどの理由でマクロであることを明示したい場合には、英大文字、数字、"\_"で構成

する場合もある。

マクロの種類毎に、次のネーミングコンベンションを定める。

#### (A) 構成マクロ

構成マクロの名称は、英大文字、数字、"\_"で構成し、次のガイドラインを設ける。

TSZ_~	~のサイズ
TBIT_~	~のビット数
TMAX_~	~の最大値
TMIN_~	~の最小値

#### 2.13.7 静的API名

静的APIの名称は、英大文字、数字、"\_"で構成し、対応するサービスコールの名称中の英小文字を英大文字で置き換えたものとする。対応するサービスコールがない場合には、サービスコールのネーミングコンベンションに従って定められた名称中の英小文字を英大文字で置き換えたものとする。

#### 2.13.8 ファイル名

ファイルの名称は、英小文字、数字、"\_"、"."で構成する。英大文字と英小文字を区別しないファイルシステムに対応するために、英大文字は使用しない。また、"- "も使用しない。

ファイルの種類毎に、次のネーミングコンベンションを定める。

#### (A) ヘッダファイル

モジュールを用いるために必要な定義を含むヘッダファイルは、そのモジュールのモジュール識別名の末尾に".h"を付加した名前（すなわち、www.h）とする。

#### 2.13.9 モジュール内部の名称の衝突回避

モジュール内部の名称が、他のモジュール内部の名称と衝突することを避けるために、次のガイドラインを設ける。

モジュール内部に閉じて使われる関数や変数などの名称で、オブジェクトファイルのシンボル表に登録されて外部から参照できる名称は、C言語レベルで、先頭が\_www\_または\_www\_である名称とする。例えば、カーネルの内部シンボルは、C言語レベルで、先頭が"\_kernel\_"または"\_KERNEL\_"である名称とする。

また、モジュールを用いるために必要な定義を含むヘッダファイル中に用いる名称で、それをインクルードする他のモジュールで使用する名称と衝突する可能性のある名称は、"TOPPERS\_"で始まる名称とする。

#### 2.14 TOPPERS共通定義

TOPPERSソフトウェアに共通に用いる定義を、TOPPERS共通定義と呼ぶ。

##### 2.14.1 TOPPERS共通ヘッダファイル

TOPPERS共通定義（共通データ型、共通定数、共通マクロ）は、TOPPERS共通ヘッダファイル（t\_stddef.h）およびそこからインクルードされるファイルに含まれている。TOPPERS共通定義を用いる場合には、TOPPERS共通ヘッダファイルをインクルードする。

TOPPERS共通ヘッダファイルは、カーネルヘッダファイル (kernel.h) やSILヘッダファイル (sil.h) からインクルードされるため、これらのファイルをインクルードする場合には、TOPPERS共通ヘッダファイルを直接インクルードする必要はない。

## 2.14.2 TOPPERS共通データ型

C90に規定されているデータ型以外で、TOPPERSソフトウェアで共通に用いるデータ型は次の通りである。

int8_t	符号付き8ビット整数 (オプション, C99準拠)
uint8_t	符号無し8ビット整数 (オプション, C99準拠)
int16_t	符号付き16ビット整数 (C99準拠)
uint16_t	符号無し16ビット整数 (C99準拠)
int32_t	符号付き32ビット整数 (C99準拠)
uint32_t	符号無し32ビット整数 (C99準拠)
int64_t	符号付き64ビット整数 (オプション, C99準拠)
uint64_t	符号無し64ビット整数 (オプション, C99準拠)
int128_t	符号付き128ビット整数 (オプション, C99準拠)
uint128_t	符号無し128ビット整数 (オプション, C99準拠)
int_least8_t	8ビット以上の符号付き整数 (C99準拠)
uint_least8_t	int_least8_t型と同じサイズの符号無し整数 (C99準拠)
float32_t	IEEE754準拠の32ビット単精度浮動小数点数 (オプション)
double64_t	IEEE754準拠の64ビット倍精度浮動小数点数 (オプション)
bool_t	真偽値 (trueまたはfalse)
char_t	符号無しの文字型 (unsigned charと一致)
int_t	16ビット以上の符号付き整数
uint_t	int_t型と同じサイズの符号無し整数
long_t	32ビット以上かつint_t型以上のサイズの符号付き整数
ulong_t	long_t型と同じサイズの符号無し整数
intptr_t	ポインタを格納できるサイズの符号付き整数 (C99準拠)
uintptr_t	intptr_t型と同じサイズの符号無し整数 (C99準拠)
FN	機能コード (符号付き整数, int_tに定義)
ER	正常終了 (E_OK) またはエラーコード (符号付き整数, int_tに定義)
ID	オブジェクトのID番号 (符号付き整数, int_tに定義)
ATR	オブジェクト属性 (符号無し整数, uint_tに定義)
STAT	オブジェクトの状態 (符号無し整数, uint_tに定義)
MODE	サービスコールの動作モード (符号無し整数, uint_tに定義)
PRI	優先度 (符号付き整数, int_tに定義)
SIZE	メモリ領域のサイズ (符号無し整数, ポインタを格納できるサイズの符号無し整数型に定義)
TMO	タイムアウト指定 (符号付き整数, 単位はミリ秒, int_tに定義)
RELTIM	相対時間 (符号無し整数, 単位はミリ秒, uint_tに定義)
SYSTIM	システム時刻 (符号無し整数, 単位はミリ秒, ulong_tに定義)
SYSUTM	性能評価用システム時刻 (符号無し整数, 単位はマイクロ秒, ulong_tに定義)
FP	プログラムの起動番地 (型の定まらない関数ポインタ)
ER_BOOL	エラーコードまたは真偽値 (符号付き整数, int_tに定義)



ER_ID	エラーコードまたはID番号（符号付き整数，int_tに定義，負のID番号は格納できない）
ER_UINT	エラーコードまたは符号無し整数（符号付き整数，int_tに定義，符号無し整数を格納する場合の有効ビット数はuint_tより1ビット短い）
ACPTN	アクセス許可パターン（符号無し32ビット整数，uint32_tに定義）
ACVCT	アクセス許可ベクタ

ここで，データ型が「AまたはB」とは，AかBのいずれかの値を取ることを示す．例えばER\_BOOLは，エラーコードまたは真偽値のいずれかの値を取る．

int8\_t, uint8\_t, int64\_t, uint64\_t, int128\_t, uint128\_t, float32\_t, double64\_tが使用できるかどうかは，ターゲットシステムに依存する．これらを使用できるかどうかは，それぞれ，INT8\_MAX, UINT8\_MAX, INT64\_MAX, UINT64\_MAX, INT128\_MAX, UINT128\_MAX, FLOAT32\_MAX, DOUBLE64\_MAXがマクロ定義されているかどうかで判別することができる．IEEE754準拠の浮動小数点数がサポートされていないターゲットシステムでは，float32\_tとdouble64\_tは使用できないものとする．

ACVCTは，ACPTNを用いて次のように定義される．

```
typedef struct acvct {
    ACPTN  acptn1; /* 通常操作1のアクセス許可パターン */
    ACPTN  acptn2; /* 通常操作2のアクセス許可パターン */
    ACPTN  acptn3; /* 管理操作のアクセス許可パターン */
    ACPTN  acptn4; /* 参照操作のアクセス許可パターン */
} ACVCT;
```

#### 【μITRON4.0仕様との関係】

B, UB, H, UH, W, UW, D, UD, VP\_INTに代えて，C99準拠のint8\_t, uint8\_t, int16\_t, uint16\_t, int32\_t, uint32\_t, int64\_t, uint64\_t, intptr\_tを用いることにした．また，uintptr\_t, int128\_t, uint128\_tを用意することにした．

VPIは，void \*と等価であるため，用意しないことにした．また，ターゲットシステムにより振舞いが一定しないことから，VB, VH, VW, VDに代わるデータ型は用意しないことにした．

INT, UINTに代えて，C99の型名と相性が良いint\_t, uint\_tを用いることにした．また，32ビット以上かつint\_t型（またはuint\_t型）以上のサイズが保証される整数型として，long\_t, ulong\_tを用意し，8ビット以上のサイズで必ず存在する整数型として，C99準拠のint\_least8\_t, uint\_least8\_tを導入することにした．int\_least16\_t, uint\_least16\_t, int\_least32\_t, uint\_least32\_tを導入しなかったのは，16ビットおよび32ビットの整数型があることを仮定しており，それぞれint16\_t, uint16\_t, int32\_t, uint32\_tで代用できるためである．

TECSとの整合性を取るために，BOOLに代えて，bool\_tを用いることにした．また，符号無し整数で文字を表す型としてchar\_t，IEEE754準拠の単精度浮動小数点数を表す型としてfloat32\_t，IEEE754準拠の64ビットを表す型としてdouble64\_tを導入した．

性能評価用システム時刻のためのデータ型として，SYSUTMを用意することにした．

#### 2.14.3 TOPPERS共通定数

C90に規定されている定数以外で，TOPPERSソフトウェアで共通に用いる定数は次の通りである（一部，C90に規定されているものも含む）。

### (1) 一般定数

NULL		無効ポインタ
true	1	真
false	0	偽
E_OK	0	正常終了

### 【μITRON4.0仕様との関係】

BOOLをbool\_tに代えたことから，TRUEおよびFALSEに代えて，trueおよびfalseを用いることにした。

### (2) 整数型に格納できる最大値と最小値

INT8_MAX	int8_tに格納できる最大値（オプション，C99準拠）
INT8_MIN	int8_tに格納できる最小値（オプション，C99準拠）
UINT8_MAX	uint8_tに格納できる最大値（オプション，C99準拠）
INT16_MAX	int16_tに格納できる最大値（C99準拠）
INT16_MIN	int16_tに格納できる最小値（C99準拠）
UINT16_MAX	uint16_tに格納できる最大値（C99準拠）
INT32_MAX	int32_tに格納できる最大値（C99準拠）
INT32_MIN	int32_tに格納できる最小値（C99準拠）
UINT32_MAX	uint32_tに格納できる最大値（C99準拠）
INT64_MAX	int64_tに格納できる最大値（オプション，C99準拠）
INT64_MIN	int64_tに格納できる最小値（オプション，C99準拠）
UINT64_MAX	uint64_tに格納できる最大値（オプション，C99準拠）
INT128_MAX	int128_tに格納できる最大値（オプション，C99準拠）
INT128_MIN	int128_tに格納できる最小値（オプション，C99準拠）
UINT128_MAX	uint128_tに格納できる最大値（オプション，C99準拠）
INT_LEAST8_MAX	int_least8_tに格納できる最大値（C99準拠）
INT_LEAST8_MIN	int_least8_tに格納できる最小値（C99準拠）
UINT_LEAST8_MAX	uint_least8_tに格納できる最大値（C99準拠）
INT_MAX	int_tに格納できる最大値（C90準拠）
INT_MIN	int_tに格納できる最小値（C90準拠）
UINT_MAX	uint_tに格納できる最大値（C90準拠）
LONG_MAX	long_tに格納できる最大値（C90準拠）
LONG_MIN	long_tに格納できる最小値（C90準拠）
ULONG_MAX	ulong_tに格納できる最大値（C90準拠）
FLOAT32_MIN	float32_tに格納できる最小の正規化された正の浮動小数点数（オプション）
FLOAT32_MAX	float32_tに格納できる表現可能な最大の有限浮動小数点数（オプション）
DOUBLE64_MIN	double64_tに格納できる最小の正規化された正の浮動小数点数（オプション）
DOUBLE64_MAX	double64_tに格納できる表現可能な最大の有限浮動小数点数（オプション）

### (3) 整数型のビット数

CHAR_BIT	char型のビット数（C90準拠）
----------	-------------------

## (4) オブジェクト属性

TA_NULL	0U	オブジェクト属性を指定しない
---------	----	----------------

## (5) タイムアウト指定

TMO_POL	0	ポーリング
TMO_FEVR	-1	永久待ち
TMO_NBLK	-2	ノンブロッキング

## (6) アクセス許可パターン

TACP_KERNEL	0U	カーネルドメインだけにアクセスを許可
TACP_SHARED	0U	すべてのドメインにアクセスを許可

## 2.14.4 TOPPERS共通エラーコード

TOPPERSソフトウェアで共通に用いるメインエラーコードは次の通りである。

## (A) 内部エラークラス (EC\_SYS, -5 ~ -8)

E_SYS	-5	システムエラー
-------	----	---------

## (B) 未サポートエラークラス (EC\_NOSPT, -9 ~ -16)

E_NOSPT	-9	未サポート機能
E_RSFN	-10	予約機能コード
E_RSATR	-11	予約属性

## (C) パラメータエラークラス (EC\_PAR, -17 ~ -24)

E_PAR	-17	パラメータエラー
E_ID	-18	不正ID番号

## (D) 呼出しコンテキストエラークラス (EC\_CTX, -25 ~ -32)

E_CTX	-25	コンテキストエラー
E_MACV	-26	メモリアクセス違反
E_OACV	-27	オブジェクトアクセス違反
E_ILUSE	-28	サービスコール不正使用

## (E) 資源不足エラークラス (EC\_NOMEM, -33 ~ -40)

E_NOMEM	-33	メモリ不足
E_NOID	-34	ID番号不足

## (F) オブジェクト状態エラークラス (EC\_OBJ, -41 ~ -48)

E_OBJ	-41	オブジェクト状態エラー
E_NOEXS	-42	オブジェクト未登録
E_QOVR	-43	キューイングオーバーフロー

## (G) 待ち解除エラークラス (EC\_RLWAI, -49 ~ -56)

E_RLWAI	-49	待ち禁止状態または待ち状態の強制解除
E_TMOUT	-50	ポーリング失敗またはタイムアウト
E_DLT	-51	待ちオブジェクトの削除または再初期化
E_CLS	-52	待ちオブジェクトの状態変化

## (H) 警告クラス (EC\_WARN, -57 ~ -64)

E_WBLK	-57	ノンブロッキング受け付け
E_BOVR	-58	バッファオーバーフロー

このエラークラスに属するエラーコードは、警告を表すエラーコードであり、サービスコールがエラーコードを返した場合には副作用がないという原則の例外となる。

## 2.14.5 TOPPERS共通マクロ

## (1) 整数定数を作るマクロ

INT8_C(val)	int_least8_t型の定数を作るマクロ (C99準拠)
UINT8_C(val)	uint_least8_t型の定数を作るマクロ (C99準拠)
INT16_C(val)	int16_t型の定数を作るマクロ (C99準拠)
UINT16_C(val)	uint16_t型の定数を作るマクロ (C99準拠)
INT32_C(val)	int32_t型の定数を作るマクロ (C99準拠)
UINT32_C(val)	uint32_t型の定数を作るマクロ (C99準拠)
INT64_C(val)	int64_t型の定数を作るマクロ (オプション, C99準拠)
UINT64_C(val)	uint64_t型の定数を作るマクロ (オプション, C99準拠)
INT128_C(val)	int128_t型の定数を作るマクロ (オプション, C99準拠)
UINT128_C(val)	uint128_t型の定数を作るマクロ (オプション, C99準拠)
UINT_C(val)	uint_t型の定数を作るマクロ
ULONG_C(val)	ulong_t型の定数を作るマクロ

## 【仕様決定の理由】

C99に用意されていないUINT\_CとULONG\_Cを導入したのは、アセンブリ言語からも参照する定数を記述するためである。C言語のみで用いる定数をこれらのマクロを使って記述する必要はない。

## (2) 型に関する情報を取り出すためのマクロ

offsetof(structure, field)	構造体structure中のフィールドfieldのバイト位置を返すマクロ (C90準拠)
alignof(type)	型typeのアラインメント単位を返すマクロ
ALIGN_TYPE(addr, type)	アドレスaddrが型typeに対してアラインしているかどうかを返すマクロ

## (3) assertマクロ

assert(exp)	expが成立しているかを検査するマクロ (C90準拠)
-------------	-----------------------------

## (4) コンパイラの拡張機能のためのマクロ

inline	インライン関数
Inline	ファイルローカルなインライン関数
asm	インラインアセンブラ
Asm	インラインアセンブラ (最適化抑止)
throw()	例外を発生しない関数
NoReturn	リターンしない関数

## (5) エラーコード生成・分解マクロ

ERCD(mercd, sercd)	メインエラーコードmercdとサブエラーコードsercdから、エラーコードを生成するためのマクロ
MERCD(ercd)	エラーコードercdからメインエラーコードを抽出するためのマクロ
SERCD(ercd)	エラーコードercdからサブエラーコードを抽出するためのマクロ

#### (6) アクセス許可パターン生成マクロ

TACP(domid)	domidで指定される保護ドメインに属する処理単位だけにアクセスを許可するアクセス許可パターン
-------------	---

ここで、TACPのパラメータ ( domid ) には、ユーザドメインのID番号のみを指定することができる。TDOM\_SELF, TDOM\_KERNEL, TDOM\_NONEを指定した場合の動作は、保証されない。

#### 2.14.6 TOPPERS共通構成マクロ

##### (1) 相対時間の範囲

TMAX_RELTIM	相対時間に指定できる最大値
-------------	---------------

#### 2.15 カーネル共通定義

カーネルの複数の機能で共通に用いる定義を、カーネル共通定義と呼ぶ。

##### 2.15.1 カーネルヘッダファイル

カーネルを用いるために必要な定義は、カーネルヘッダファイル ( kernel.h ) およびそこからインクルードされるファイルに含まれている。カーネルを用いる場合には、カーネルヘッダファイルをインクルードする。

ただし、カーネルを用いるために必要な定義の中で、コンフィギュレータによって生成されるものは、カーネル構成・初期化ヘッダファイル ( kernel\_cfg.h ) に含まれる。具体的には、登録できるオブジェクトの数 ( TNUM\_YYY ) やオブジェクトのID番号などの定義が、これに該当する。これらの定義を用いる場合には、カーネル構成・初期化ヘッダファイルをインクルードする。

μITRON4.0仕様で規定されており、この仕様で廃止されたデータ型および定数を用いる場合には、ITRON仕様互換ヘッダファイル ( itron.h ) をインクルードする。

##### 2.15.2 カーネル共通定数

###### (1) オブジェクト属性

TA_TPRI	0x01U	タスクの待ち行列をタスクの優先度順に
---------	-------	--------------------

###### 【 μITRON4.0仕様との関係】

値が0のオブジェクト属性 ( TA\_HLNG, TA\_TFIFO, TA\_MFIFO ) は、デフォルトの扱いにして廃止した。これは、「 ( tskatr & TA\_HLNG ) != 0U 」のような間違いを防ぐためである。TA\_ASMは、有効な用途がないために廃止した。TA\_MPRIは、メールボックス機能でのみ使用するため、共通定義から外した。

###### (2) 保護ドメインID

TDOM_SELF	0	自タスクの属する保護ドメイン
TDOM_KERNEL	-1	カーネルドメイン
TDOM_NONE	-2	無所属（保護ドメインに属さない）

## (3) その他のカーネル共通定数

TSK_SELF	0	自タスク指定
TSK_NONE	0	該当するタスクがない
TPRI_SELF	0	自タスクのベース優先度の指定
TPRI_INI	0	タスクの起動時優先度の指定
TIPM_ENAALL	0	割込み優先度マスク全解除

## (4) カーネルで用いるメインエラーコード

「2.14.4 TOPPERS共通エラーコード」の節で定義したメインエラーコードの中で、E\_CLS、E\_WBLK、E\_BOVRの3つは、カーネルでは使用しない。

## 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、サービスコールから、E\_SYS、E\_RSFN、E\_RSATR、E\_MACV、E\_OACV、E\_NOMEM、E\_NOID、E\_NOEXSが返る状況は起こらない。E\_RSATRは、コンフィギュレータによって検出される。

## 2.15.3 カーネル共通構成マクロ

## (1) サポートする機能

TOPPERS_SUPPORT_PROTECT	保護機能対応のカーネル
TOPPERS_SUPPORT_MULTI_PRC	マルチプロセッサ対応のカーネル
TOPPERS_SUPPORT_DYNAMIC_CRE	動的生成対応のカーネル

## 【未決定事項】

マクロ名は、今後変更する可能性がある。

## (2) 優先度の範囲

TMIN_TPRI	タスク優先度の最小値（=1）
TMAX_TPRI	タスク優先度の最大値

## 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、タスク優先度の最大値（TMAX\_TPRI）は16に固定されている。ただし、タスク優先度拡張パッケージを用いると、TMAX\_TPRIを256に拡張することができる。

## 【μITRON4.0仕様との関係】

メッセージ優先度の最小値（TMIN\_MPRI）と最大値（TMAX\_MPRI）は、メールボックス機能でのみ使用するため、カーネル共通定義から外した。

## (3) バージョン情報

TKERNEL_MAKER	カーネルのメーカコード（=0x0118）
TKERNEL_PRID	カーネルの識別番号

TKERNEL_SPVER	カーネル仕様のバージョン番号
TKERNEL_PRVER	カーネルのバージョン番号

カーネルのメーカーコード (TKERNEL\_MAKER) は, TOPPERSプロジェクトから配布するカーネルでは, TOPPERSプロジェクトを表す値 (0x0118) に設定されている.

カーネルの識別番号 (TKERNEL\_PRID) は, TOPPERSカーネルの種類を表す.

0x0001	TOPPERS/JSPカーネル
0x0002	予約 (IIMPカーネル)
0x0003	予約 (IDLカーネル)
0x0004	TOPPERS/FI4カーネル
0x0005	TOPPERS/FDMPカーネル
0x0006	TOPPERS/HRPカーネル
0x0007	TOPPERS/ASPカーネル
0x0008	TOPPERS/FMPカーネル (仮)

カーネル仕様のバージョン番号 (TKERNEL\_SPVER) は, 上位8ビット (0xf5) が TOPPERS新世代カーネル仕様であることを, 中位4ビットがメジャーバージョン番号, 下位4ビットがマイナーバージョン番号を表す.

カーネルのバージョン番号 (TKERNEL\_PRVER) は, 上位4ビットがメジャーバージョン番号, 中位8ビットがマイナーバージョン番号, 下位4ビットがパッチレベルを表す.

### 第3章 システムインタフェースレイヤAPI仕様

#### 3.1 システムインタフェースレイヤの概要

システムインタフェースレイヤ (SIL) は, デバイスを直接操作するプログラムが用いるための機能である. ITRONデバイスドライバ設計ガイドラインの一部として検討されたものをベースに, TOPPERSプロジェクトにおいて修正を加えて用いている.

#### 3.2 SILヘッダファイル

SILを用いるために必要な定義は, SILヘッダファイル (sil.h) およびそこからインクルードされるファイルに含まれている. SILを用いる場合には, SILヘッダファイルをインクルードする.

#### 3.3 全割り込みロック状態の制御

デバイスを扱うプログラムの中では, すべての割り込み (NMIを除く, 以下同じ) をマスクしたい場合がある. カーネルで制御できるCPUロック状態は, カーネル管理外の割り込み (NMI以外にカーネル管理外の割り込みがあるかはターゲット定義) をマスクしないため, このような場合に用いることはできない.

そこで, SILでは, すべての割り込みをマスクする全割り込みロック状態を制御するための以下の機能を用意している.

##### (1) SIL\_PRE\_LOC

全割り込みロック状態の制御に必要な変数を宣言するマクロ. 通常は, 型と変数名を並べたもので, 最後に ";" を含まない. SIL\_LOC\_INT, SIL\_UNL\_INTを用いる関数またはブロックの先頭の変数宣言部に記述しなければならない.

##### (2) SIL\_LOC\_INT()

全割り込みロックフラグをセットすることで、NMIを除くすべての割り込みをマスクし、全割り込みロック状態に遷移する。

### (3) SIL\_UNL\_INT()

全割り込みロックフラグを、対応するSIL\_LOC\_INTを実行する前の状態に戻す。

全割り込みロック状態の制御機能の使用例は次の通り。

```
{
    SIL_PRE_LOC;

    SIL_LOC_INT();
    // この間はNMIを除くすべての割り込みがマスクされる。
    // この間にサービスコールを呼び出してはならない(一部例外あり)。
    SIL_UNL_INT();
}
```

なお、全割り込みロック状態で呼び出せるサービスコールなどの制限事項については、「2.5.4 全割り込みロック状態と全割り込みロック解除状態」の節を参照すること。

## 3.4 微少時間待ち

デバイスをアクセスする際に、微少な時間待ちを入れなければならない場合がある。そのような場合に、NOP命令をいくつか入れるなどの方法で対応すると、ポータビリティを損なうことになる。そこで、SILでは、微少な時間待ちを行うための以下の機能を用意している。

### (1) void sil\_dly\_nse(ulong\_t dlytim)

dlytimで指定された以上の時間(単位はナノ秒)、ループなどによって待つ。指定した値によっては、指定した時間よりもかなり長く待つ場合があるので注意すること。

## 3.5 エンディアン

プロセッサのバイトエンディアンを知るためのマクロとして、SILでは、以下のマクロを定義している。

### (1) SIL\_ENDIAN\_BIG, SIL\_ENDIAN\_LITTLE

ビッグエンディアンプロセッサではSIL\_ENDIAN\_BIGを、リトルエンディアンプロセッサではSIL\_ENDIAN\_LITTLEを、マクロ定義している。

## 3.6 メモリ空間アクセス関数

メモリ空間にマッピングされたデバイスレジスタや、デバイスとの共有メモリをアクセスするために、SILでは、以下の関数を用意している。

### (1) uint8\_t sil\_reb\_mem(void \*mem)

memで指定されるアドレスから8ビット単位で読み出した値を返す。

### (2) void sil\_wrb\_mem(void \*mem, uint8\_t data)

memで指定されるアドレスにdataで指定される値を8ビット単位で書き込む。



(3) uint16\_t sil\_reh\_mem(void \*mem)

memで指定されるアドレスから16ビット単位で読み出した値を返す。

(4) void sil\_wrh\_mem(void \*mem, uint16\_t data)

memで指定されるアドレスにdataで指定される値を16ビット単位で書き込む。

(5) uint16\_t sil\_reh\_lem(void \*mem)

memで指定されるアドレスから16ビット単位でリトルエンディアンで読み出した値を返す。リトルエンディアンプロセッサでは、sil\_reh\_memと一致する。ビッグエンディアンプロセッサでは、sil\_reh\_memが返す値を、エンディアン変換した値を返す。

(6) void sil\_wrh\_lem(void \*mem, uint16\_t data)

memで指定されるアドレスにdataで指定される値を16ビット単位でリトルエンディアンで書き込む。リトルエンディアンプロセッサでは、sil\_wrh\_memと一致する。ビッグエンディアンプロセッサでは、dataをエンディアン変換した値を、sil\_wrh\_memで書き込むのと同じ結果となる。

(7) uint16\_t sil\_reh\_bem(void \*mem)

memで指定されるアドレスから16ビット単位でビッグエンディアンで読み出した値を返す。ビッグエンディアンプロセッサでは、sil\_reh\_memと一致する。リトルエンディアンプロセッサでは、sil\_reh\_memが返す値を、エンディアン変換した値を返す。

(8) void sil\_wrh\_bem(void \*mem, uint16\_t data)

memで指定されるアドレスにdataで指定される値を16ビット単位でビッグエンディアンで書き込む。ビッグエンディアンプロセッサでは、sil\_wrh\_memと一致する。リトルエンディアンプロセッサでは、dataをエンディアン変換した値を、sil\_wrh\_memで書き込むのと同じ結果となる。

(9) uint32\_t sil\_rew\_mem(void \*mem)

memで指定されるアドレスから32ビット単位で読み出した値を返す。

(10) void sil\_rwr\_mem(void \*mem, uint32\_t data)

memで指定されるアドレスにdataで指定される値を32ビット単位で書き込む。

(11) uint32\_t sil\_rew\_lem(void \*mem)

memで指定されるアドレスから32ビット単位でリトルエンディアンで読み出した値を返す。リトルエンディアンプロセッサでは、sil\_rew\_memと一致する。ビッグエンディアンプロセッサでは、sil\_rew\_memが返す値を、エンディアン変換した値を返す。

(12) void sil\_rwr\_lem(void \*mem, uint32\_t data)

memで指定されるアドレスにdataで指定される値を32ビット単位でリトルエンディアンで書き込む。リトルエンディアンプロセッサでは、sil\_rwr\_memと一致する。ビッグエンディアンプロセッサでは、dataをエンディアン変換した値を、sil\_rwr\_memで書き込むのと同じ結果となる。

(13) uint32\_t sil\_rew\_bem(void \*mem)

memで指定されるアドレスから32ビット単位でビッグエンディアンで読み出した値を返す。ビッグエンディアンプロセッサでは、sil\_rew\_memと一致する。リトルエンディアンプロセッサでは、sil\_rew\_memが返す値を、エンディアン変換した値を返す。

(14) void sil\_wrw\_bem(void \*mem, uint32\_t data)

memで指定されるアドレスにdataで指定される値を32ビット単位でビッグエンディアンで書き込む。ビッグエンディアンプロセッサでは、sil\_wrw\_memと一致する。リトルエンディアンプロセッサでは、dataをエンディアン変換した値を、sil\_wrw\_memで書き込むのと同じ結果となる。

### 3.7 I/O空間アクセス関数

メモリ空間とは別にI/O空間を持つプロセッサでは、I/O空間にあるデバイスレジスタをアクセスするために、メモリ空間アクセス関数と同等の以下の関数を用意している。

- (1) uint8\_t sil\_reb\_iop(void \*iop)
- (2) void sil\_wrb\_iop(void \*iop, uint8\_t data)
- (3) uint16\_t sil\_reh\_iop(void \*iop)
- (4) void sil\_wrh\_iop(void \*iop, uint16\_t data)
- (5) uint16\_t sil\_reh\_lep(void \*iop)
- (6) void sil\_wrh\_lep(void \*iop, uint16\_t data)
- (7) uint16\_t sil\_reh\_bep(void \*iop)
- (8) void sil\_wrh\_bep(void \*iop, uint16\_t data)
- (9) uint32\_t sil\_rew\_iop(void \*iop)
- (10) void sil\_wrw\_iop(void \*iop, uint32\_t data)
- (11) uint32\_t sil\_rew\_lep(void \*iop)
- (12) void sil\_wrw\_lep(void \*iop, uint32\_t data)
- (13) uint32\_t sil\_rew\_bep(void \*iop)
- (14) void sil\_wrw\_bep(void \*iop, uint32\_t data)

## 第4章 カーネルAPI仕様

この章では、カーネルのAPI仕様について規定する。

カーネルのAPIの種別とAPIをサポートするカーネルの種類を表すために、次の記号を用いる。

〔T〕はタスクコンテキスト専用のサービスコールを示す。非タスクコンテキストから呼び出すと、E\_CTXエラーとなる。

〔I〕は非タスクコンテキスト専用のサービスコールを示す。タスクコンテキストから呼び出すと、E\_CTXエラーとなる。

〔TI〕はタスクコンテキストからも非タスクコンテキストからも呼び出すことのできるサービスコールを示す。

〔S〕は静的APIを示す。

〔P〕は保護機能対応カーネルのみでサポートされているAPIを示す。保護機能対応でないカーネルでは、このAPIはサポートされない。

〔M〕はマルチプロセッサ対応カーネルのみでサポートされているAPIを示す。マルチプロセッサ対応でないカーネルでは、このAPIはサポートされない。

〔D〕は動的生成対応カーネルのみでサポートされているAPIを示す。動的生成対応でないカーネルでは、このAPIはサポートされない。

また、エラーコードが返るカーネルの種類を表すために、次の記号を用いる。

〔P〕は保護機能対応カーネルのみで返るエラーコードを示す。保護機能対応でないカーネルでは、このエラーコードは返らない。

〔D〕は動的生成対応カーネルのみで返るエラーコードを示す。動的生成対応でないカーネルでは、このエラーコードは返らない。

#### 【μITRON4.0仕様との関係】

TOPPERS共通データ型に従い、パラメータのデータ型を次の通り変更した。これらの変更については、個別のAPI仕様では記述しない。

```
INT    int_t
UINT   uint_t
VP     void *
VP_INT intptr_t
```

#### 【μITRON4.0/PX仕様との関係】

ID番号で識別するオブジェクトのアクセス許可ベクタをデフォルト以外に設定する場合には、オブジェクトを生成した後に設定することとし、アクセス許可ベクタを設定する静的API (SAC\_YYY) を新設した。逆に、アクセス許可ベクタを指定してオブジェクトを生成する機能 (CRA\_YYY, cra\_yyy, acra\_yyy) は廃止したいサポートしないことにしたらの変更については、個別のAPI仕様では記述しない。

#### 【未決定事項】

保護機能対応・動的生成対応カーネルにおいて、オブジェクトを生成するサービスコールに対してオブジェクトの属するドメインの指定する方法、マルチプロセッサ対応・動的生成対応カーネルにおいて、オブジェクトを生成するサービスコールに対してオブジェクトの属するクラスを指定する方法は、今後の課題である。

### 4.1 タスク管理機能

タスクは、プログラムの並行実行の単位で、カーネルが実行を制御する処理単位である。タスクは、タスクIDと呼ぶID番号によって識別する。

タスク管理機能に関連して、各タスクが持つ情報は次の通り。

- ・タスク属性
- ・タスク状態
- ・ベース優先度
- ・現在優先度
- ・起動要求キューイング数
- ・拡張情報
- ・メインルーチンの先頭番地
- ・起動時優先度
- ・スタック領域
- ・システムスタック領域 (保護機能対応カーネルの場合)

タスクのベース優先度は、タスクの現在優先度を決定するために使われる優先度であり、タスクの起動時に起動時優先度に初期化される。

タスクの現在優先度は、タスクの実行順位を決定するために使われる優先度であり、単にタスクの優先度ともいう。タスクがミューテックスをロックしていない間は、タスクの現在優先度はベース優先度に一致する。ミューテックスをロックしている間のタスクの現在優先度については、「4.4.6 ミューテックス」の節を参照すること。

タスクの起動要求キューイング数は、処理されていないタスクの起動要求の数であり、タスクの生成時に0に初期化される。

システムスタック領域は、保護機能対応カーネルにおいて、タスクが呼び出したサービスコール（拡張サービスコールを含む）を実行するために用いるスタック領域である。

タスク属性には、次の属性を指定することができる。

TA\_ACT      0x01U      タスクの生成時にタスクを起動する

TA\_ACTを指定しない場合、タスクの生成直後には、タスクは休止状態となる。また、ターゲットによっては、ターゲット定義のタスク属性を指定できる場合がある。ターゲット定義のタスク属性として、次の属性を予約している。

TA\_FPU                  FPUレジスタをコンテキストに含める

C言語によるタスクの記述形式は次の通り。

```
void task(intptr_t exinf)
{
    タスク本体
    ext_tsk();
}
```

exinfには、タスクの拡張情報が渡される。ext\_tskを呼び出さず、タスクのメインルーチンからリターンしてもよい（ext\_tskを呼び出した場合と、同じ動作をする）。

タスク管理機能に関連するカーネル構成マクロは次の通り。

TMAX\_ACTCNT      タスクの起動要求キューイング数の最大値

TNUM\_TSKID      登録できるタスクの数（動的生成対応でないカーネルでは、静的APIによって登録されたタスクの数に一致）

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、TMAX\_ACTCNTは1に固定されている。

#### 【μITRON4.0仕様との関係】

この仕様では、自タスクの拡張情報の参照するサービスコール（get\_inf）をサポートし、起動コードを指定してタスクを起動するサービスコール（sta\_tsk）、タスクを終了と同時に削除するサービスコール（exd\_tsk）、タスクの状態を参照するサービスコールの簡易版（ref\_tst）はサポートしないこととした。

TNUM\_TSKIDは、μITRON4.0仕様に規定されていないカーネル構成マクロである。

-----  
 CRE\_TSK タスクの生成〔S〕  
 acre\_tsk タスクの生成〔TD〕

#### 【静的API】

\* 保護機能対応でないカーネルの場合

```
CRE_TSK(ID tskid, { ATR tskatr, intptr_t exinf, TASK task,
                  PRI itskpri, SIZE stksz, STK_T *stk })
```

\* 保護機能対応カーネルの場合

```
CRE_TSK(ID tskid, { ATR tskatr, intptr_t exinf, TASK task,
                  PRI itskpri, SIZE stksz, STK_T *stk, SIZE sstksz, STK_T *sstk })
```

#### 【C言語API】

```
ER_ID tskid = acre_tsk(const T_CTSK *pk_ctsk)
```

#### 【パラメータ】

ID	tskid	生成するタスクのID番号 (acre_tskを除く)
T_CTSK *	pk_ctsk	タスクの生成情報を入れたパケットへのポインタ (静的APIを除く)

\* タスクの生成情報 (パケットの内容)

ATR	tskatr	タスク属性
intptr_t	exinf	タスクの拡張情報
TASK	task	タスクのメインルーチンの先頭番地
PRI	itskpri	タスクの起動時優先度
SIZE	stksz	タスクのスタック領域のサイズ (バイト数)
STK_T *	stk	タスクのスタック領域の先頭番地
SIZE	sstksz	タスクのシステムスタック領域のサイズ (バイト数, 保護機能対応カーネルの場合)
STK_T *	sstk	タスクのシステムスタック領域の先頭番地 (保護機能対応カーネルの場合)

#### 【リターンパラメータ】

ER_ID	tskid	生成されたタスクのID番号 (正の値) またはエラーコード
-------	-------	-------------------------------

#### 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
E_RSATR	予約属性 (tskatrが不正または使用できない)
E_PAR	パラメータエラー (task, itskpri, stksz, stk, sstksz, sstkが不正)
E_OACV〔P〕	オブジェクトアクセス違反 (システム状態に対する管理操作が許可されていない)
E_MACV〔P〕	メモリアクセス違反 (pk_ctskが指すメモリ領域への読出しアクセスが許可されていない)
E_NOID	ID番号不足 (割り付けられるタスクIDがない: acre_tskの場合)
E_OBJ	オブジェクト状態エラー (tskidで指定したタスクが登録済み: acre_tskを除く)

#### 【機能】

各パラメータで指定したタスク生成情報に従って, タスクを生成する. 具体的な振舞いは以下の通り.

まず, stkとstkszからタスクが用いるスタック領域が, 保護機能対応のカーネ

ルにおいてはsstkとsstkszからシステムスタック領域が、それぞれ設定される。

次に、生成されたタスクに対してタスク生成時に行うべき初期化処理が行われ、生成されたタスクは休止状態になる。さらに、tskatrにTA\_ACTを指定した場合には、タスク起動時に行うべき初期化処理が行われ、生成されたタスクは実行できる状態になる。

静的APIにおいては、tskidはオブジェクト識別名、tskatr, itskpri, stkszは整数定数式パラメータ、exinf, task, stkは一般定数式パラメータである。

stkをNULLとした場合、stkszで指定したサイズのスタック領域を、コンフィギュレータまたはカーネルが確保する。stkszにターゲットシステムの制約に合致しないサイズを指定した時には、ターゲットシステムの制約に合致するようにサイズを大きい方に丸めて確保する。

stkがNULLでない場合、stkszで指定したサイズのスタック領域を、アプリケーションで確保する。スタック領域をアプリケーションで確保するために、次のデータ型とマクロを用意している。

STK_T	スタック領域を確保するためのデータ型
COUNT_STK_T(sz)	サイズszのスタック領域を確保するために必要なSTK_T型の配列の要素数を求めるマクロ
ROUND_STK_T(sz)	要素数COUNT_STK_T(sz)のSTK_T型の配列のサイズ (szを、STK_T型のサイズの倍数になるように大きい方に丸めた値)

これらを用いてスタック領域を確保する方法は次の通り。

```
STK_T <スタック領域の変数名>[COUNT_STK_T(<スタック領域のサイズ>)];
```

この時、stkには<スタック領域の変数名>を、stkszにはROUND\_STK\_T(<スタック領域のサイズ>)を指定する。

この方法に従わず、stkやstkszにターゲットシステムの制約に合致しない先頭番地とサイズを指定した時には、E\_PARエラーとなる。

itskpriは、TMIN\_TPRI以上、TMAX\_TPRI以下でなければならない。

#### 【未決定事項】

保護機能対応カーネルにおけるスタック領域およびシステムスタック領域の指定方法と確保方法については、今後の課題である。

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、CRE\_TSKのみをサポートする。

#### 【μITRON4.0仕様との関係】

taskのデータ型をTASKに、stkのデータ型をSTK\_T \*に変更した。

#### 【μITRON4.0/PX仕様との関係】

sstkのデータ型をSTK\_T \*に変更した。

---

SAC_TSK	タスクのアクセス許可ベクタの設定 [SP]
sac_tsk	タスクのアクセス許可ベクタの設定 [TPD]

**【静的API】**

```
SAC_TSK(ID tskid, { ACPTN acptn1, ACPTN acptn2,
                    ACPTN acptn3, ACPTN acptn4 })
```

**【C言語API】**

```
ER ercd = sac_tsk(ID tskid, const ACVCT *p_acvct)
```

**【パラメータ】**

ID	tskid	対象タスクのID番号
ACVCT *	p_acvct	アクセス許可ベクタを入れたパケットへのポインタ (静的APIを除く)

**\* アクセス許可ベクタ (パケットの内容)**

ACPTN	acptn1	通常操作1のアクセス許可パターン
ACPTN	acptn2	通常操作2のアクセス許可パターン
ACPTN	acptn3	管理操作のアクセス許可パターン
ACPTN	acptn4	参照操作のアクセス許可パターン

**【リターンパラメータ】**

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

**【エラーコード】**

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
E_ID	不正ID番号 (tskidが不正)
E_NOEXS [D]	オブジェクト未登録 (対象タスクが未登録)
E_OACV [P]	オブジェクトアクセス違反 (対象タスクに対する管理操作が許可されていない)
E_MACV [P]	メモリアクセス違反 (p_acvctが指すメモリ領域への読出しアクセスが許可されていない)
E_OBJ	オブジェクト状態エラー (対象タスクは静的APIで生成された: 静的APIを除く)

**【機能】**

tskidで指定したタスク (対象タスク) のアクセス許可ベクタ (4つのアクセス許可パターンの組) を, 各パラメータで指定した値に設定する.

tskidにTSK\_SELF (= 0) を指定すると, 自タスクが対象タスクとなる.

**【TOPPERS/ASPカーネルにおける規定】**

ASPカーネルでは, SAC\_TSK, sac\_tskをサポートしない.

-----  
del\_tsk      タスクの削除 [TD]

**【C言語API】**

```
ER ercd = del_tsk(ID tskid)
```

**【パラメータ】**

ID	tskid	対象タスクのID番号
----	-------	------------

**【リターンパラメータ】**

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

**【エラーコード】**

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し)
-------	------------------------------

	し, CPUロック状態からの呼出し)
E_ID	不正ID番号 (tskidが不正)
E_NOEXS	オブジェクト未登録 (対象タスクが未登録)
E_OACV [P]	オブジェクトアクセス違反 (対象タスクに対する管理操作が許可されていない)
E_OBJ	オブジェクト状態エラー (対象タスクが休止状態でない, 対象タスクは静的APIで生成された)

### 【機能】

tskidで指定したタスク (対象タスク) を削除する。具体的な振舞いは以下の通り。

対象タスクが休止状態である場合には, 対象タスクの登録が解除され, そのタスクIDが未使用の状態に戻される。また, タスクの生成時にタスクのスタック領域およびシステムスタック領域がカーネルによって確保された場合は, それらのメモリ領域が解放される。

対象タスクが休止状態でない場合には, E\_OBJエラーとなる。

### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは, del\_tskをサポートしない。

-----  
act\_tsk     タスクの起動 [T]  
iact\_tsk    タスクの起動 [I]

### 【C言語API】

```
ER ercd = act_tsk(ID tskid)
ER ercd = iact_tsk(ID tskid)
```

### 【パラメータ】

ID	tskid	対象タスクのID番号
----	-------	------------

### 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

### 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し: act_tskの場合, タスクコンテキストからの呼出し: iact_tskの場合, CPUロック状態からの呼出し)
E_ID	不正ID番号 (tskidが不正)
E_NOEXS [D]	オブジェクト未登録 (対象タスクが未登録)
E_OACV [P]	オブジェクトアクセス違反 (対象タスクに対する通常操作1が許可されていない: act_tskの場合)
E_QOVR	キューイングオーバーフロー (起動要求キューイング数がTMAX_ACTCNTに一致)

### 【機能】

tskidで指定したタスク (対象タスク) に対して起動要求を行う。具体的な振舞いは以下の通り。

対象タスクが休止状態である場合には, 対象タスクに対してタスク起動時に行うべき初期化処理が行われ, 対象タスクは実行できる状態になる。

対象タスクが休止状態でない場合には, 対象タスクの起動要求キューイング数に1が加えられる。起動要求キューイング数に1を加えるとTMAX\_ACTCNTを超える



場合には、E\_QOVRエラーとなる。

act\_tskにおいてtskidにTSK\_SELF (= 0) を指定すると、自タスクが対象タスクとなる。

-----  
can\_act      タスク起動要求のキャンセル [T]

**【C言語API】**

```
ER_UINT actcnt = can_act(ID tskid)
```

**【パラメータ】**

ID            tskid            対象タスクのID番号

**【リターンパラメータ】**

ER\_UINT      actcnt            キューイングされていた起動要求の数 ( 正の値  
または0 ) またはエラーコード

**【エラーコード】**

E\_CTX                    コンテキストエラー ( 非タスクコンテキストからの呼出し、CPUロック状態からの呼出し )  
E\_ID                      不正ID番号 ( tskidが不正 )  
E\_NOEXS [D]              オブジェクト未登録 ( 対象タスクが未登録 )  
E\_OACV [P]                オブジェクトアクセス違反 ( 対象タスクに対する通常操作1が許可されていない )

**【機能】**

tskidで指定したタスク ( 対象タスク ) に対する処理されていない起動要求をすべてキャンセルし、キャンセルした起動要求の数を返す。具体的な振舞いは以下の通り。

対象タスクの起動要求キューイング数が0に設定され、0に設定する前の起動要求キューイング数が、サービスコールの返回值として返される。

tskidにTSK\_SELF (= 0) を指定すると、自タスクが対象タスクとなる。

-----  
ext\_tsk      自タスクの終了 [T]

**【C言語API】**

```
ER ercd = ext_tsk()
```

**【パラメータ】**

なし

**【リターンパラメータ】**

ER            ercd            エラーコード

**【エラーコード】**

E\_CTX                    コンテキストエラー ( 非タスクコンテキストからの呼出し )

**【機能】**

自タスクを終了させる。具体的な振舞いは以下の通り。

自タスクに対してタスク終了時に行うべき処理が行われ、自タスクは休止状態になる。さらに、自タスクの起動要求キューイング数が0でない場合には、自タスクに対してタスク起動時に行うべき処理が行われ、自タスクは実行できる状

態になる。またこの時、起動要求キューイング数から1が減ぜられる。

ext\_tskは、CPUロック解除状態、割込み優先度マスク全解除状態、ディスパッチ許可状態と呼び出すのが原則であるが、そうでない状態で呼び出された場合には、CPUロック解除状態、割込み優先度マスク全解除状態、ディスパッチ許可状態に遷移させた後、自タスクを終了させる。

ext\_tskが正常に処理された場合、ext\_tskからはリターンしない。

#### 【μITRON4.0仕様との関係】

ext\_tskを非タスクコンテキストから呼び出した場合に、E\_CTXエラーが返ることとした。μITRON4.0仕様においては、ext\_tskからはリターンしないと規定されている。

-----  
 ter\_tsk      タスクの強制終了〔T〕

#### 【C言語API】

```
ER ercd = ter_tsk(ID tskid)
```

#### 【パラメータ】

ID	tskid	対象タスクのID番号
----	-------	------------

#### 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

#### 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
E_ID	不正ID番号 (tskidが不正)
E_NOEXS〔D〕	オブジェクト未登録 (対象タスクが未登録)
E_OACV〔P〕	オブジェクトアクセス違反 (対象タスクに対する通常操作2が許可されていない)
E_ILUSE	サービスコール不正使用 (対象タスクが自タスク, 対象タスクが自タスクと異なるプロセッサに属している: マルチプロセッサ対応カーネルの場合)
E_OBJ	オブジェクト状態エラー (対象タスクが休止状態)

#### 【機能】

tskidで指定したタスク (対象タスク) を終了させる。具体的な振舞いは以下の通り。

対象タスクが休止状態でない場合には、対象タスクに対してタスク終了時に行うべき処理が行われ、対象タスクは休止状態になる。さらに、対象タスクの起動要求キューイング数が0でない場合には、対象タスクに対してタスク起動時に行うべき処理が行われ、対象タスクは実行できる状態になる。またこの時、起動要求キューイング数から1が減ぜられる。

対象タスクが休止状態である場合には、E\_OBJエラーとなる。

また、対象タスクが自タスクの場合と、マルチプロセッサ対応カーネルにおいて対象タスクが自タスクと異なるプロセッサに属している場合には、E\_ILUSEエラーとなる。

-----  
 chg\_pri      タスク優先度の変更〔T〕

#### 【C言語API】

```
ER ercd = chg_pri(ID tskid, PRI tskpri)
```

#### 【パラメータ】

ID	tskid	対象タスクのID番号
PRI	tskpri	ベース優先度

#### 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

#### 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
E_ID	不正ID番号 (tskidが不正)
E_PAR	パラメータエラー (tskpriが不正)
E_NOEXS [D]	オブジェクト未登録 (対象タスクが未登録)
E_OACV [P]	オブジェクトアクセス違反 (対象タスクに対する通常操作2が許可されていない)
E_ILUSE	サービスコール不正使用 ( 未完成)
E_OBJ	オブジェクト状態エラー (対象タスクが休止状態)

#### 【機能】

tskidで指定したタスク (対象タスク) のベース優先度を, tskpriで指定した優先度に変更する. 具体的な振舞いは以下の通り.

対象タスクが休止状態でない場合には, 対象タスクのベース優先度が, tskpriで指定した優先度に変更される.

未完成

対象タスクが休止状態である場合には, E\_OBJエラーとなる.

tskidにTSK\_SELF (=0) を指定すると, 自タスクが対象タスクとなる. また, tskpriにTPRI\_INI (=0) を指定すると, 対象タスクのベース優先度が, 起動時優先度に変更される.

未完成 (E\_ILUSEの条件)

tskpriは, TPRI\_INIであるか, TMIN\_TPRI以上, TMAX\_TPRI以下でなければならない.

-----  
get\_pri      タスク優先度の参照 [T]

#### 【C言語API】

```
ER ercd = get_pri(ID tskid, PRI *p_tskpri)
```

#### 【パラメータ】

ID	tskid	対象タスクのID番号
PRI *	p_tskpri	現在優先度を入れるメモリ領域へのポインタ

#### 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
PRI	tskpri	現在優先度

#### 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
E_ID	不正ID番号 (tskidが不正)

E_NOEXS [D]	オブジェクト未登録 (対象タスクが未登録)
E_OACV [P]	オブジェクトアクセス違反 (対象タスクに対する通常操作1が許可されていない)
E_MACV [P]	メモリアクセス違反 (p_tskpriが指すメモリ領域への書込みアクセスが許可されていない)
E_OBJ	オブジェクト状態エラー (対象タスクが休止状態)

### 【機能】

tskidで指定したタスク (対象タスク) の現在優先度を参照する。具体的な振舞いは以下の通り。

対象タスクが休止状態でない場合には、対象タスクの現在優先度が、p\_tskpriで指定したメモリ領域に返される。対象タスクが休止状態である場合には、E\_OBJエラーとなる。

tskidにTSK\_SELF (= 0) を指定すると、自タスクが対象タスクとなる。

-----  
get\_inf 自タスクの拡張情報の参照 [T]

### 【C言語API】

```
ER ercd = get_inf(intptr_t *p_exinf)
```

### 【パラメータ】

intptr\_t \* p\_exinf 拡張情報を入れるメモリ領域へのポインタ

### 【リターンパラメータ】

ER ercd 正常終了 (E\_OK) またはエラーコード  
intptr\_t exinf 拡張情報

### 【エラーコード】

E\_CTX コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)  
E\_MACV [P] メモリアクセス違反 (p\_exinfが指すメモリ領域への書込みアクセスが許可されていない)

### 【機能】

自タスクの拡張情報を参照する。参照した拡張情報は、p\_exinfで指定したメモリ領域に返される。

### 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである。

-----  
ref\_tsk タスクの状態参照 [T]

### 【C言語API】

```
ER ercd = ref_tsk(ID tskid, T_RTsk *pk_rtsk)
```

### 【パラメータ】

ID tskid 対象タスクのID番号  
T\_RTsk \* pk\_rtsk タスクの現在状態を入れるパケットへのポインタ

### 【リターンパラメータ】

ER ercd 正常終了 (E\_OK) またはエラーコード

\* タスクの現在状態 (パケットの内容)

STAT	tskstat	タスク状態
PRI	tskpri	タスクの現在優先度
PRI	tskbpri	タスクのベース優先度
STAT	tskwait	タスクの待ち要因
ID	wobjid	タスクの待ち対象のオブジェクトのID
TMO	lefttmo	タスクがタイムアウトするまでの時間
uint_t	actcnt	タスクの起動要求キューイング数
uint_t	wupcnt	タスクの起床要求キューイング数

## 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し，CPUロック状態からの呼出し）
E_ID	不正ID番号（tskidが不正）
E_NOEXS [D]	オブジェクト未登録（対象タスクが未登録）
E_OACV [P]	オブジェクトアクセス違反（対象タスクに対する参照操作が許可されていない）
E_MACV [P]	メモリアクセス違反（pk_rtskが指すメモリ領域への書込みアクセスが許可されていない）

## 【機能】

tskidで指定したタスク（対象タスク）の現在状態を参照する．参照した現在状態は，pk\_rtskで指定したメモリ領域に返される．

tskstatには，対象タスクの現在のタスク状態を表す次のいずれかの値が返される．

TTS_RUN	0x01U	実行状態
TTS_RDY	0x02U	実行可能状態
TTS_WAI	0x04U	待ち状態
TTS_SUS	0x08U	強制待ち状態
TTS_WAS	0x0cU	二重待ち状態
TTS_DMT	0x10U	休止状態

対象タスクが休止状態でない場合には，tskpriには対象タスクの現在優先度が，tskbpriには対象タスクのベース優先度が返される．対象タスクが休止状態である場合には，tskpriとtskbpriの値は保証されない．

対象タスクが待ち状態である場合には，tskwaitには，対象タスクが何を待っている状態であるかを表す次のいずれかの値が返される．

TTW_SLP	0x0001U	起床待ち
TTW_DLY	0x0002U	時間経過待ち
TTW_SEM	0x0004U	セマフォの資源獲得待ち
TTW_FLG	0x0008U	イベントフラグ待ち
TTW_SDTQ	0x0010U	データキューへの送信待ち
TTW_RDTQ	0x0020U	データキューからの受信待ち
TTW_SPDQ	0x0100U	優先度データキューへの送信待ち
TTW_RPDQ	0x0200U	優先度データキューからの受信待ち
TTW_MTX	0x0080U	ミューテックスのロック待ち状態
TTW_MBX	0x0040U	メールボックスからの受信待ち
TTW_MPF	0x2000U	固定長メモリブロックの獲得待ち

対象タスクが待ち状態でない場合には，tskwaitの値は保証されない．

対象タスクが起床待ち状態および時間経過待ち状態以外の待ち状態である場合には，wobjidに，対象タスクが待っているオブジェクトのID番号が返される．対象タスクが待ち状態でない場合や，起床待ち状態または時間経過待ち状態で

ある場合には、wobjidの値は保証されない。

対象タスクが時間経過待ち状態以外の待ち状態である場合には、lefttmoに、タスクがタイムアウトを起こすまでの相対時間が返される。タスクがタイムアウトを起こさない場合には、TMO\_FEVR (= -1) が返される。

対象タスクが時間経過待ち状態である場合には、lefttmoに、タスクの遅延時間が経過して待ち解除されるまでの相対時間が返される。ただし、返されるべき相対時間がTMO型に格納することができない場合がありうる。この場合には、相対時間 (RELTIM型, uint\_t型に定義される) をTMO型 (int\_t型に定義される) に型キャストした値が返される。

対象タスクが待ち状態でない場合には、lefttmoの値は保証されない。

actcntには、対象タスクの起動要求キューイング数が返される。

対象タスクが休止状態でない場合には、wupcntに、タスクの起床要求キューイング数が返される。対象タスクが休止状態である場合には、wupcntの値は保証されない。

tskidにTSK\_SELF (= 0) を指定すると、自タスクが対象タスクとなる。

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、tskwaitにTTW\_MTXが返ることはない。

#### 【使用上の注意】

ref\_tskはデバッグ時向けの機能であり、その他の目的に使用することは推奨しない。これは、ref\_tskを呼び出し、対象タスクの現在状態を参照した直後に割込みが発生した場合、ref\_tskから戻ってきた時には対象タスクの状態が変化している可能性があるためである。

#### 【μITRON4.0仕様との関係】

対象タスクが時間経過待ち状態の時にlefttmoに返される値について規定した。また、参照できるタスクの状態から、強制待ち要求ネスト数 (suscnt) を除外した。

## 4.2 タスク付属同期機能

タスク付属同期機能は、タスクとタスクの間、または非タスクコンテキストの処理とタスクの間で同期を取るために、タスク単独で持っている機能である。

タスク付属同期機能に関連して、各タスクが持つ情報は次の通り。

- ・起床要求キューイング数

タスクの起床要求キューイング数は、処理されていないタスクの起床要求の数であり、タスクの起動時に0に初期化される。

タスク付属同期機能に関連するカーネル構成マクロは次の通り。

TMAX\_WUPCNT      タスクの起床要求キューイング数の最大値

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、TMAX\_WUPCNTは1に固定されている。

#### 【μITRON4.0仕様との関係】

この仕様では、強制待ち要求をネストする機能をサポートしないこととした。言い換えると、強制待ち要求ネスト数の最大値を1に固定する。これに伴い、強制待ち状態から強制再開するサービスコール ( frsm\_tsk ) と TMAX\_SUSCNT は廃止した。また、ref\_tsk で参照できる情報 ( T\_RTsk のフィールド ) から、強制待ち要求ネスト数 ( suscnt ) を除外した。

-----  
 slp\_tsk 起床待ち [ T ]  
 tslp\_tsk 起床待ち ( タイムアウト付き ) [ T ]

#### 【C言語API】

```
ER ercd = slp_tsk()
ER ercd = tslp_tsk(TMO tmout)
```

#### 【パラメータ】

TMO tmout タイムアウト時間 ( tslp\_tsk の場合 )

#### 【リターンパラメータ】

ER ercd 正常終了 ( E\_OK ) またはエラーコード

#### 【エラーコード】

E\_CTX コンテキストエラー ( 非タスクコンテキストからの呼出し、CPUロック状態からの呼出し、ディスパッチ保留状態からの呼出し )  
 E\_PAR パラメータエラー ( tmout が不正 : tslp\_tsk の場合 )  
 E\_TMOUT ポーリング失敗またはタイムアウト ( slp\_tsk を除く )  
 E\_RLWAI 待ち禁止状態または待ち状態の強制解除

#### 【機能】

自タスクを起床待ちさせる。具体的な振舞いは以下の通り。

自タスクの起床要求キューイング数が0でない場合には、起床要求キューイング数から1が減ぜられる。起床要求キューイング数が0の場合には、自タスクは起床待ち状態となる。

#### 【補足説明】

自タスクの起床要求キューイング数が0でない場合には、自タスクは実行できる状態を維持し、自タスクの優先順位は変化しない。

-----  
 wup\_tsk タスクの起床 [ T ]  
 iwup\_tsk タスクの起床 [ I ]

#### 【C言語API】

```
ER ercd = wup_tsk(ID tskid)
ER ercd = iwup_tsk(ID tskid)
```

#### 【パラメータ】

ID tskid 対象タスクのID番号

#### 【リターンパラメータ】

ER ercd 正常終了 ( E\_OK ) またはエラーコード

#### 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し：wup_tskの場合、タスクコンテキストからの呼出し：iwup_tskの場合、CPUロック状態からの呼出し）
E_ID	不正ID番号（tskidが不正）
E_NOEXS [D]	オブジェクト未登録（対象タスクが未登録）
E_OACV [P]	オブジェクトアクセス違反（対象タスクに対する通常操作1が許可されていない：wup_tskの場合）
E_OBJ	オブジェクト状態エラー（対象タスクが休止状態）
E_QOVR	キューイングオーバーフロー（起床要求キューイング数がTMAX_WUPCNTに一致）

### 【機能】

tskidで指定したタスク（対象タスク）を起床する．具体的な振舞いは以下の通り．

対象タスクが起床待ち状態である場合には，対象タスクが待ち解除される．待ち解除されたタスクには，待ち状態となったサービスコールからE\_OKが返る．

対象タスクが起床待ち状態でなく，休止状態でもない場合には，対象タスクの起床要求キューイング数に1が加えられる．起床要求キューイング数に1を加えるとTMAX\_WUPCNTを超える場合には，E\_QOVRエラーとなる．

対象タスクが休止状態である場合には，E\_OBJエラーとなる．

tskidにTSK\_SELF（=0）を指定すると，自タスクが対象タスクとなる．

-----  
can\_wup      タスク起床要求のキャンセル [T]

### 【C言語API】

```
ER_UINT wupcnt = can_wup(ID tskid)
```

### 【パラメータ】

ID	tskid	対象タスクのID番号
----	-------	------------

### 【リターンパラメータ】

ER_UINT	wupcnt	キューイングされていた起床要求の数（正の値または0）またはエラーコード
---------	--------	-------------------------------------

### 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し，CPUロック状態からの呼出し）
E_ID	不正ID番号（tskidが不正）
E_NOEXS [D]	オブジェクト未登録（対象タスクが未登録）
E_OACV [P]	オブジェクトアクセス違反（対象タスクに対する通常操作1が許可されていない）
E_OBJ	オブジェクト状態エラー（対象タスクが休止状態）

### 【機能】

tskidで指定したタスク（対象タスク）に対する処理されていない起床要求をすべてキャンセルし，キャンセルした起床要求の数を返す．具体的な振舞いは以下の通り．

対象タスクが休止状態でない場合には，対象タスクの起床要求キューイング数が0に設定され，0に設定する前の起床要求キューイング数が，サービスコールの返値として返される．



対象タスクが休止状態である場合には、E\_OBJエラーとなる。

tskidにTSK\_SELF (=0) を指定すると、自タスクが対象タスクとなる。

-----  
rel\_wai 強制的な待ち解除 [T]  
irel\_wai 強制的な待ち解除 [I]

#### 【C言語API】

```
ER ercd = rel_wai(ID tskid)
ER ercd = irel_wai(ID tskid)
```

#### 【パラメータ】

ID tskid 対象タスクのID番号

#### 【リターンパラメータ】

ER ercd 正常終了 (E\_OK) またはエラーコード

#### 【エラーコード】

E\_CTX コンテキストエラー (非タスクコンテキストからの呼出し: rel\_waiの場合、タスクコンテキストからの呼出し: irel\_waiの場合、CPUロック状態からの呼出し)  
E\_ID 不正ID番号 (tskidが不正)  
E\_NOEXS [D] オブジェクト未登録 (対象タスクが未登録)  
E\_OACV [P] オブジェクトアクセス違反 (対象タスクに対する通常操作2が許可されていない: rel\_waiの場合)  
E\_OBJ オブジェクト状態エラー (対象タスクが待ち状態でない)

#### 【機能】

tskidで指定したタスク (対象タスク) を、強制的に待ち解除する。具体的な振舞いは以下の通り。

対象タスクが待ち状態である場合には、対象タスクが待ち解除される。待ち解除されたタスクには、待ち状態となったサービスコールからE\_RLWAIが返る。

対象タスクが待ち状態でない場合には、E\_OBJエラーとなる。

-----  
sus\_tsk 強制待ち状態への遷移 [T]

#### 【C言語API】

```
ER ercd = sus_tsk(ID tskid)
```

#### 【パラメータ】

ID tskid 対象タスクのID番号

#### 【リターンパラメータ】

ER ercd 正常終了 (E\_OK) またはエラーコード

#### 【エラーコード】

E\_CTX コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し, 対象タスクが自タスクでディスパッチ保留状態からの呼出し)  
E\_ID 不正ID番号 (tskidが不正)  
E\_NOEXS [D] オブジェクト未登録 (対象タスクが未登録)  
E\_OACV [P] オブジェクトアクセス違反 (対象タスクに対する通常操作2が許可されていない)  
E\_OBJ オブジェクト状態エラー (対象タスクが休止状態)  
E\_QOVR キューイングオーバーフロー (対象タスクが強制待ち状態)

## 【機能】

tskidで指定したタスク（対象タスク）を強制待ちにする．具体的な振舞いは以下の通り．

対象タスクが実行できる状態である場合には，対象タスクは強制待ち状態となる．また，待ち状態（二重待ち状態を除く）である場合には，二重待ち状態となる．

対象タスクが強制待ち状態または二重待ち状態である場合はE\_QOVRエラー，休止状態である場合にはE\_OBJエラーとなる．

tskidにTSK\_SELF (= 0) を指定すると，自タスクが対象タスクとなる．

-----  
rsm\_tsk 強制待ち状態からの再開〔T〕

## 【C言語API】

```
ER ercd = rsm_tsk(ID tskid)
```

## 【パラメータ】

ID	tskid	対象タスクのID番号
----	-------	------------

## 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

## 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し，CPUロック状態からの呼出し）
E_ID	不正ID番号（tskidが不正）
E_NOEXS〔D〕	オブジェクト未登録（対象タスクが未登録）
E_OACV〔P〕	オブジェクトアクセス違反（対象タスクに対する通常操作2が許可されていない）
E_OBJ	オブジェクト状態エラー（対象タスクが強制待ち状態でない）

## 【機能】

tskidで指定したタスク（対象タスク）を，強制待ちから再開する．具体的な振舞いは以下の通り．

対象タスクが強制待ち状態である場合には，対象タスクは強制待ちから再開される．強制待ち状態でない場合には，E\_OBJエラーとなる．

-----  
dis\_wai 待ち禁止状態への遷移〔TP〕

idis\_wai 待ち禁止状態への遷移〔IP〕

## 【C言語API】

```
ER ercd = dis_wai(ID tskid)
```

```
ER ercd = idis_wai(ID tskid)
```

## 【パラメータ】

ID	tskid	対象タスクのID番号
----	-------	------------

## 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

## 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し：dis_waiの場合、タスクコンテキストからの呼出し：idis_waiの場合、CPUロック状態からの呼出し）
E_ID	不正ID番号（tskidが不正）
E_NOEXS [D]	オブジェクト未登録（対象タスクが未登録）
E_OACV [P]	オブジェクトアクセス違反（対象タスクに対する通常操作2が許可されていない：dis_waiの場合）
E_OBJ	オブジェクト状態エラー（対象タスクが休止状態、対象タスクが拡張サービスコールを実行中でない）
E_QOVR	キューイングオーバーフロー（対象タスクが待ち禁止状態）

### 【機能】

tskidで指定したタスク（対象タスク）を待ち禁止状態にする．具体的な振舞いは以下の通り．

対象タスクが拡張サービスコールを実行中であり、待ち禁止状態でない場合には、対象タスクは待ち禁止状態になる．

対象タスクが休止状態である場合には、E\_OBJエラーとなる．また、対象タスクが拡張サービスコールを実行中でない場合にはE\_OBJエラー、待ち禁止状態の場合にはE\_QOVRエラーとなる．

tskidにTSK\_SELF（=0）を指定すると、自タスクが対象タスクとなる．

### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、dis\_waiをサポートしない．

### 【補足説明】

dis\_waiは、対象タスクの待ち解除は行わない．対象タスクを待ち解除したい場合には、dis\_waiを呼び出した後に、rel\_waiを呼び出せばよい．

-----  
ena\_wai 待ち禁止状態の解除〔TP〕  
iena\_wai 待ち禁止状態の解除〔IP〕

### 【C言語API】

```
ER ercd = ena_wai(ID tskid)
ER ercd = iena_wai(ID tskid)
```

### 【パラメータ】

ID	tskid	対象タスクのID番号
----	-------	------------

### 【リターンパラメータ】

ER	ercd	正常終了（E_OK）またはエラーコード
----	------	---------------------

### 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し：ena_waiの場合、タスクコンテキストからの呼出し：iena_waiの場合、CPUロック状態からの呼出し）
E_ID	不正ID番号（tskidが不正）
E_NOEXS [D]	オブジェクト未登録（対象タスクが未登録）
E_OACV [P]	オブジェクトアクセス違反（対象タスクに対する通常操作2が許可されていない：ena_waiの場合）
E_OBJ	オブジェクト状態エラー（対象タスクが待ち禁止状態でない）

## 【機能】

tskidで指定したタスク（対象タスク）の待ち禁止状態を解除する．具体的な振舞いは以下の通り．

対象タスクが待ち禁止状態である場合には，待ち禁止状態は解除される．対象タスクが待ち禁止状態でない場合には，E\_OBJエラーとなる．

tskidにTSK\_SELF（=0）を指定すると，自タスクが対象タスクとなる．

## 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは，ena\_waiをサポートしない．

-----  
dly\_tsk 自タスクの遅延 [T]

## 【C言語API】

```
ER ercd = dly_tsk(RELTIM dlytim)
```

## 【パラメータ】

RELTIM	dlytim	遅延時間
--------	--------	------

## 【リターンパラメータ】

ER	ercd	正常終了（E_OK）またはエラーコード
----	------	---------------------

## 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し，CPUロック状態からの呼出し，ディスパッチ保留状態からの呼出し）
E_PAR	パラメータエラー（dlytimが不正）
E_RLWAI	待ち禁止状態または待ち状態の強制解除

## 【機能】

dlytimで指定した時間，自タスクを遅延させる．具体的な振舞いは以下の通り．

自タスクは，dlytimで指定した時間が経過するまでの間，時間経過待ち状態となる．dly\_tskを呼び出してからdlytimで指定した相対時間後に，自タスクは待ち解除され，dly\_tskからE\_OKが返る．

dlytimは，TMAX\_RELTIM以下でなければならない．

## 4.3 タスク例外処理機能

タスク例外処理ルーチンは，カーネルが実行を制御する処理単位で，タスクと同一のコンテキスト内で実行される．タスク例外処理ルーチンは，各タスクに1つのみ登録できるため，タスクIDによって識別する．

タスク例外処理機能に関連して，各タスクが持つ情報は次の通り．

- ・タスク例外処理ルーチン属性
- ・タスク例外処理禁止フラグ
- ・保留例外要因
- ・タスク例外処理ルーチンの先頭番地

タスク例外処理ルーチン属性に指定できる属性はない．そのため，タスク例外処理ルーチン属性には，TA\_NULLを指定しなければならない．

タスクは、タスク例外処理ルーチンの実行を保留するためのタスク例外処理禁止フラグを持つ。タスク例外処理禁止フラグがセットされた状態をタスク例外処理禁止状態、クリアされた状態をタスク例外処理許可状態と呼ぶ。タスク例外処理禁止フラグは、タスクの起動時に、セットした状態に初期化される。

タスクの保留例外要因は、タスクに対して要求された例外要因を蓄積するためのビットマップであり、タスクの起動時に0に初期化される。

タスク例外処理ルーチンは、「タスク例外処理許可状態」「保留例外要因が0でない」「CPUロック状態でない」「タスク自身のコンテキストが実行されている」の4つの条件が揃った場合に実行が開始される。タスク例外処理ルーチンの実行が開始される時、タスク例外処理禁止フラグはセットされ、保留例外要因は0にクリアされる。

タスク例外処理機能に用いるデータ型は次の通り。

TEXPTN      タスク例外要因のビットパターン（符号無し整数，uint\_tに定義）

C言語によるタスク例外処理ルーチンの記述形式は次の通り。

```
void task_exception_routine(TEXPTN texptn, intptr_t exinf)
{
    タスク例外処理ルーチン本体
}
```

texptnにはタスク例外処理ルーチン起動時の保留例外要因が、exinfにはタスクの拡張情報が、それぞれ渡される。

タスク例外処理機能に関連するカーネル構成マクロは次の通り。

TBIT\_TEXPTN      タスク例外要因のビット数（TEXPTNの有効ビット数）

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、タスク例外要因のビット数（TBIT\_TEXPTN）は16以上である。

DEF\_TEX      タスク例外処理ルーチンの定義〔S〕  
def\_tex      タスク例外処理ルーチンの定義〔TD〕

【静的API】

DEF\_TEX(ID tskid, { ATR texatr, TEXRTN texrtn })

【C言語API】

ER ercd = def\_tex(ID tskid, const T\_DTEX \*pk\_dtex)

【パラメータ】

ID	tskid	対象タスクのID番号
T_DTEX *	pk_dtex	タスク例外処理ルーチンの定義情報を入れたパケットへのポインタ（静的APIを除く）

\* タスク例外処理ルーチンの定義情報（パケットの内容）

ATR	texatr	タスク例外処理ルーチン属性
TEXRTN	texrtn	タスク例外処理ルーチンの先頭番地

【リターンパラメータ】

ER	ercd	正常終了（E_OK）またはエラーコード
----	------	---------------------

## 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し，CPUロック状態からの呼出し）
E_ID	不正ID番号（tskidが不正）
E_NOEXS [D]	オブジェクト未登録（対象タスクが未登録）
E_OACV [P]	オブジェクトアクセス違反（対象タスクに対する管理操作が許可されていない）
E_MACV [P]	メモリアクセス違反（pk_dtexが指すメモリ領域への読出しアクセスが許可されていない）
E_RSATR	予約属性（texatrが不正または使用できない）
E_PAR	パラメータエラー（texrtnが不正）
E_OBJ	オブジェクト状態エラー（タスク例外処理ルーチンを定義済みのタスクに対する定義，タスク例外処理ルーチンを未定義のタスクに対する解除，対象タスクは静的APIで生成された）

## 【機能】

tskidで指定したタスク（対象タスク）に対して，各パラメータで指定したタスク例外処理ルーチン定義情報に従って，タスク例外処理ルーチンを定義する．ただし，def\_texにおいてpk\_dtexをNULLにした場合には，対象タスクに対するタスク例外処理ルーチンの定義を解除する．

静的APIにおいては，tskidはオブジェクト識別名，texatrは整数定数式パラメータ，texrtnは一般定数式パラメータである．

静的APIによって生成したタスクに対しては，タスク例外処理ルーチンの登録はDEF\_TEXによって行わねばならず，def\_texによってタスク例外処理ルーチンを登録／登録解除することはできない．def\_texにおいて，対象タスクが静的APIで生成したタスクである場合には，E\_OBJエラーとなる．

タスク例外処理ルーチンを定義する場合（DEF\_TEXの場合およびdef\_texにおいてpk\_dtexをNULL以外にした場合）で，対象タスクに対してすでにタスク例外処理ルーチンが定義されている場合には，E\_OBJエラーとなる．

タスク例外処理ルーチンの定義を解除する場合（def\_texにおいてpk\_dtexをNULLにした場合）で，対象タスクに対してタスク例外処理ルーチンが定義されていない場合には，E\_OBJエラーとなる．

## 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは，DEF\_TEXのみをサポートする．

## 【μITRON4.0仕様との関係】

texrtnのデータ型をTEXRTNIに変更した．

def\_texによって，定義済みのタスク例外処理ルーチンを再定義しようとした場合に，E\_OBJエラーとすることにした．

-----  
 ras\_tex     タスク例外処理の要求 [T]  
 iras\_tex    タスク例外処理の要求 [I]

## 【C言語API】

ER ercd = ras\_tex(ID tskid, TEXPTN rasptn)  
 ER ercd = iras\_tex(ID tskid, TEXPTN rasptn)

## 【パラメータ】

ID	tskid	対象タスクのID番号
TEXPTN	rasptn	要求するタスク例外処理のタスク例外要因

## 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

## 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し: ras_texの場合, タスクコンテキストからの呼出し: iras_texの場合, CPUロック状態からの呼出し)
E_ID	不正ID番号 (tskidが不正)
E_NOEXS [D]	オブジェクト未登録 (対象タスクが未登録)
E_OACV [P]	オブジェクトアクセス違反 (対象タスクに対する通常操作2が許可されていない: ras_texの場合)
E_PAR	パラメータエラー (rasptnが不正)
E_OBJ	オブジェクト状態エラー (対象タスクが休止状態, 対象タスクに対してタスク例外処理ルーチンが定義されていない)

## 【機能】

tskidで指定したタスク (対象タスク) に対して, rasptnで指定したタスク例外要因のタスク例外処理を要求する. 対象タスクの保留例外要因が, それまでの値とrasptnで指定した値のビット毎論理和 (C言語の"|") に更新される.

tskidにTSK\_SELF (= 0) を指定すると, 自タスクが対象タスクとなる.

rasptnが0の場合には, E\_PARエラーとなる.

-----  
dis\_tex      タスク例外処理の禁止 [T]

## 【C言語API】

```
ER ercd = dis_tex()
```

## 【パラメータ】

なし

## 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

## 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
E_OBJ	オブジェクト状態エラー (自タスクに対してタスク例外処理ルーチンが定義されていない)

## 【機能】

自タスクのタスク例外処理禁止フラグをセットする. すなわち, 自タスクをタスク例外処理禁止状態に遷移させる.

-----  
ena\_tex      タスク例外処理の許可 [T]

## 【C言語API】

```
ER ercd = ena_tex()
```

## 【パラメータ】

なし

## 【リターンパラメータ】

ER ercd 正常終了 (E\_OK) またはエラーコード

## 【エラーコード】

E\_CTX コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)  
 E\_OBJ オブジェクト状態エラー (自タスクに対してタスク例外処理ルーチンが定義されていない)

## 【機能】

自タスクのタスク例外処理禁止フラグをクリアする。すなわち, 自タスクをタスク例外処理許可状態に遷移させる。

-----  
 sns\_tex タスク例外処理禁止状態の参照 [TI]

## 【C言語API】

```
bool_t state = sns_tex()
```

## 【パラメータ】

なし

## 【リターンパラメータ】

bool\_t state タスク例外処理禁止状態

## 【機能】

実行状態のタスクのタスク例外処理禁止フラグを参照する。具体的な振舞いは以下の通り。

実行状態のタスクが, タスク例外処理禁止状態の場合にtrue, タスク例外処理許可状態の場合にfalseが返る。sns\_texを非タスクコンテキストから呼び出した場合で, 実行状態のタスクがない場合には, trueが返る。

## 【補足説明】

sns\_texをタスクコンテキストから呼び出した場合, 実行状態のタスクは自タスクに一致する。

-----  
 ref\_tex タスク例外処理の状態参照 [T]

## 【C言語API】

```
ER ercd = ref_tex(ID tskid, T_RTEX *pk_rtex)
```

## 【パラメータ】

ID tskid 対象タスクのID番号  
 T\_RTEX \* pk\_rtex タスク例外処理の現在状態を入れるパケットへのポインタ

## 【リターンパラメータ】

ER ercd 正常終了 (E\_OK) またはエラーコード

\* タスク例外処理の現在状態 (パケットの内容)

STAT texstat タスク例外処理の状態  
 TEXPTN pndptn タスクの保留例外要因



## 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し，CPUロック状態からの呼出し）
E_ID	不正ID番号（tskidが不正）
E_NOEXS [D]	オブジェクト未登録（対象タスクが未登録）
E_OACV [P]	オブジェクトアクセス違反（対象タスクに対する参照操作が許可されていない）
E_MACV [P]	メモリアクセス違反（pk_rtexが指すメモリ領域への書込みアクセスが許可されていない）
E_OBJ	オブジェクト状態エラー（対象タスクが休止状態，対象タスクに対してタスク例外処理ルーチンが定義されていない）

## 【機能】

tskidで指定したタスク（対象タスク）のタスク例外処理に関する現在状態を参照する．参照した現在状態は，pk\_rtexで指定したパケットに返される．

tskstatには，対象タスクの現在のタスク例外処理禁止フラグを表す次のいずれかの値が返される．

TTEX_ENA	0x01U	タスク例外処理許可状態
TTEX_DIS	0x02U	タスク例外処理禁止状態

pndpntnには，対象タスクの現在の保留例外要因が返される．

## 4.4 同期・通信機能

## 【μITRON4.0仕様との関係】

この仕様では，ランデブ機能はサポートしていない．今後の検討により，ランデブ機能をサポートすることに変更する可能性もある．

## 4.4.1 セマフォ

セマフォは，資源の数を表す0以上の整数値を取るカウンタ（資源数）を介して，排他制御やイベント通知を行うためのカーネルオブジェクトである．セマフォの資源数から1を減ずることを資源の獲得，資源数に1を加えることを資源の返却と呼ぶ．セマフォは，セマフォIDと呼ぶID番号によって識別する．

各セマフォが持つ情報は次の通り．

- ・セマフォ属性
- ・資源数（の現在値）
- ・待ち行列（セマフォの資源獲得待ち状態のタスクのキュー）
- ・初期資源数（資源数の初期値）
- ・最大資源数（資源数が取りうる最大値）

待ち行列は，セマフォの資源が獲得できるまで待っている状態（セマフォの資源獲得待ち状態）のタスクが，資源を獲得できる順序でつながれているキューである．

セマフォの初期資源数は，セマフォを生成または再初期化した際の，資源数の初期値である．また，セマフォの最大資源数は，資源数が取りうる最大値である．資源数が最大資源数に一致している時に資源を返却しようとする時，E\_QOVRエラーとなる．

セマフォ属性には、次の属性を指定することができる。

TA\_TPRI 0x01U 待ち行列をタスクの優先度順にする

TA\_TPRIを指定しない場合、待ち行列はFIFO順になる。

セマフォ機能に関連するカーネル構成マクロは次の通り。

TMAX\_MAXSEM セマフォの最大資源数の最大値

TNUM\_SEMID 登録できるセマフォの数（動的生成対応でないカーネルでは、静的APIによって登録されたセマフォの数に一致）

この仕様では、TMAX\_MAXSEMは、UINT\_MAX (uint\_tに格納できる最大値) に固定されている。

#### 【μITRON4.0仕様との関係】

TNUM\_SEMIDは、μITRON4.0仕様に規定されていないカーネル構成マクロである。

-----  
 CRE\_SEM セマフォの生成〔S〕  
 acre\_sem セマフォの生成〔TD〕

#### 【静的API】

CRE\_SEM(ID semid, { ATR sematr, uint\_t isemcnt, uint\_t maxsem })

#### 【C言語API】

ER\_ID semid = acre\_sem(const T\_CSEM \*pk\_csem)

#### 【パラメータ】

ID	semid	生成するセマフォのID番号 (acre_semを除く)
T_CSEM *	pk_csem	セマフォの生成情報を入れたパケットへのポインタ (静的APIを除く)

\*セマフォの生成情報 (パケットの内容)

ATR	sematr	セマフォ属性
uint_t	isemcnt	セマフォの初期資源数
uint_t	maxsem	セマフォの最大資源数

#### 【リターンパラメータ】

ER_ID	semid	生成されたセマフォのID番号 (正の値) またはエラーコード
-------	-------	--------------------------------

#### 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
E_RSATR	予約属性 (sematrが不正または使用できない)
E_PAR	パラメータエラー (isemcnt, maxsemが不正)
E_OACV〔P〕	オブジェクトアクセス違反 (システム状態に対する管理操作が許可されていない)
E_MACV〔P〕	メモリアクセス違反 (pk_csemが指すメモリ領域への読出しアクセスが許可されていない)
E_NOID	ID番号不足 (割り付けられるセマフォIDがない: acre_semの場合)
E_OBJ	オブジェクト状態エラー (semidで指定したセマフォが登録済み: acre_semを除く)

#### 【機能】

各パラメータで指定したセマフォ生成情報に従って、セマフォを生成する。生成されたセマフォの資源数は初期資源数に、待ち行列は空の状態に初期化される。

静的APIにおいては、semidはオブジェクト識別名、isemcntとmaxsemは整数定数式パラメータである。

isemcntは、0以上で、maxsem以下でなければならない。また、maxsemは、1以上で、TMAX\_MAXSEM以下でなければならない。

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、CRE\_SEMのみをサポートする。

-----  
 SAC\_SEM      セマフォのアクセス許可ベクタの設定〔SP〕  
 sac\_sem      セマフォのアクセス許可ベクタの設定〔TPD〕

#### 【静的API】

```
SAC_SEM(ID semid, { ACPTN acptn1, ACPTN acptn2,
                  ACPTN acptn3, ACPTN acptn4 })
```

#### 【C言語API】

```
ER ercd = sac_sem(ID semid, const ACVCT *p_acvct)
```

#### 【パラメータ】

ID	semid	対象セマフォのID番号
ACVCT *	p_acvct	アクセス許可ベクタを入れたパケットへのポインタ（静的APIを除く）

#### \* アクセス許可ベクタ（パケットの内容）

ACPTN	acptn1	通常操作1のアクセス許可パターン
ACPTN	acptn2	通常操作2のアクセス許可パターン
ACPTN	acptn3	管理操作のアクセス許可パターン
ACPTN	acptn4	参照操作のアクセス許可パターン

#### 【リターンパラメータ】

ER	ercd	正常終了（E_OK）またはエラーコード
----	------	---------------------

#### 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し、CPUロック状態からの呼出し）
E_ID	不正ID番号（semidが不正）
E_NOEXS〔D〕	オブジェクト未登録（対象セマフォが未登録）
E_OACV〔P〕	オブジェクトアクセス違反（対象セマフォに対する管理操作が許可されていない）
E_MACV〔P〕	メモリアクセス違反（p_acvctが指すメモリ領域への読出しアクセスが許可されていない）
E_OBJ	オブジェクト状態エラー（対象セマフォは静的APIで生成された：静的APIを除く）

#### 【機能】

semidで指定したセマフォ（対象セマフォ）のアクセス許可ベクタ（4つのアクセス許可パターンの組）を、各パラメータで指定した値に設定する。

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、SAC\_SEM, sac\_semをサポートしない。

-----  
del\_sem セマフォの削除 [TD]

**【C言語API】**

```
ER ercd = del_sem(ID semid)
```

**【パラメータ】**

ID semid 対象セマフォのID番号

**【リターンパラメータ】**

ER ercd 正常終了 (E\_OK) またはエラーコード

**【エラーコード】**

E\_CTX コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)  
 E\_ID 不正ID番号 (semidが不正)  
 E\_NOEXS オブジェクト未登録 (対象セマフォが未登録)  
 E\_OACV [P] オブジェクトアクセス違反 (対象セマフォに対する管理操作が許可されていない)  
 E\_OBJ オブジェクト状態エラー (対象セマフォは静的APIで生成された)

**【機能】**

semidで指定したセマフォ (対象セマフォ) を削除する。具体的な振舞いは以下の通り。

対象セマフォの登録が解除され、そのセマフォIDが未使用の状態に戻される。また、対象セマフォの待ち行列につながれたタスクは、待ち行列の先頭のタスクから順に待ち解除される。待ち解除されたタスクには、待ち状態となったサービスコールからE\_DLTエラーが返る。

**【使用上の注意】**

del\_semにより複数のタスクが待ち解除される場合、サービスコールの処理時間およびカーネル内での割り込み禁止時間が、待ち解除されるタスクの数に比例して長くなる。特に、多くのタスクが待ち解除される場合、カーネル内での割り込み禁止時間が長くなるため、注意が必要である。

**【TOPPERS/ASPカーネルにおける規定】**

ASPカーネルでは、del\_semをサポートしない。

-----  
sig\_sem セマフォの資源の返却 [T]

isig\_sem セマフォの資源の返却 [I]

**【C言語API】**

```
ER ercd = sig_sem(ID semid)
```

```
ER ercd = isig_sem(ID semid)
```

**【パラメータ】**

ID semid 対象セマフォのID番号

**【リターンパラメータ】**

ER ercd 正常終了 (E\_OK) またはエラーコード

**【エラーコード】**

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し：sig_semの場合、タスクコンテキストからの呼出し：isig_semの場合、CPUロック状態からの呼出し）
E_ID	不正ID番号（semidが不正）
E_NOEXS [D]	オブジェクト未登録（対象セマフォが未登録）
E_OACV [P]	オブジェクトアクセス違反（対象セマフォに対する通常操作1が許可されていない：sig_semの場合）
E_QOVR	キューイングオーバーフロー（資源数が最大資源数に一致）

### 【機能】

semidで指定したセマフォ（対象セマフォ）に資源を返却する．具体的な振舞いは以下の通り．

対象セマフォの待ち行列にタスクが存在する場合には、待ち行列の先頭のタスクが待ち解除される．この時、待ち解除されたタスクが資源を獲得したことになるため、対象セマフォの資源数は変化しない．待ち解除されたタスクには、待ち状態となったサービスコールからE\_OKが返る．

待ち行列にタスクが存在しない場合には、対象セマフォの資源数に1が加えられる．資源数に1を加えるとそのセマフォの最大資源数を越える場合には、E\_QOVRエラーとなる．

-----

wai_sem	セマフォの資源の獲得 [T]
pol_sem	セマフォの資源の獲得（ポーリング） [T]
twai_sem	セマフォの資源の獲得（タイムアウト付き） [T]

### 【C言語API】

```
ER ercd = wai_sem(ID semid)
ER ercd = pol_sem(ID semid)
ER ercd = twai_sem(ID semid, TMO tmout)
```

### 【パラメータ】

ID	semid	対象セマフォのID番号
TMO	tmout	タイムアウト時間（twai_semの場合）

### 【リターンパラメータ】

ER	ercd	正常終了（E_OK）またはエラーコード
----	------	---------------------

### 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し、CPUロック状態からの呼出し、ディスパッチ保留状態からの呼出し：pol_semを除く）
E_ID	不正ID番号（semidが不正）
E_PAR	パラメータエラー（tmoutが不正：twai_semの場合）
E_NOEXS [D]	オブジェクト未登録（対象セマフォが未登録）
E_OACV [P]	オブジェクトアクセス違反（対象セマフォに対する通常操作2が許可されていない）
E_TMOUT	ポーリング失敗またはタイムアウト（wai_semを除く）
E_RLWAI	待ち禁止状態または待ち状態の強制解除（pol_semを除く）
E_DLT	待ちオブジェクトの削除または再初期化（pol_semを除く）

### 【機能】

semidで指定したセマフォ（対象セマフォ）から資源を獲得する．具体的な振舞いは以下の通り．

対象セマフォの資源数が1以上の場合には、資源数から1が減ぜられる．資源数

が0の場合には、自タスクはセマフォの資源獲得待ち状態となり、対象セマフォの待ち行列につながる。

-----  
ini\_sem      セマフォの再初期化〔T〕

**【C言語API】**

ER ercd = ini\_sem(ID semid)

**【パラメータ】**

ID            semid          対象セマフォのID番号

**【リターンパラメータ】**

ER            ercd            正常終了 (E\_OK) またはエラーコード

**【エラーコード】**

E\_CTX            コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)  
E\_ID              不正ID番号 (semidが不正)  
E\_NOEXS〔D〕    オブジェクト未登録 (対象セマフォが未登録)  
E\_OACV〔P〕     オブジェクトアクセス違反 (対象セマフォに対する管理操作が許可されていない)

**【機能】**

semidで指定したセマフォ (対象セマフォ) を再初期化する。具体的な振舞いは以下の通り。

対象セマフォの資源数は、初期資源数に初期化される。また、対象セマフォの待ち行列につながれたタスクは、待ち行列の先頭のタスクから順に待ち解除される。待ち解除されたタスクには、待ち状態となったサービスコールから E\_DLTエラーが返る。

**【使用上の注意】**

ini\_semにより複数のタスクが待ち解除される場合、サービスコールの処理時間およびカーネル内での割込み禁止時間が、待ち解除されるタスクの数に比例して長くなる。特に、多くのタスクが待ち解除される場合、カーネル内での割込み禁止時間が長くなるため、注意が必要である。

セマフォを再初期化した場合に、アプリケーションとの整合性を保つのは、アプリケーションの責任である。

**【μITRON4.0仕様との関係】**

μITRON4.0仕様に定義されていないサービスコールである。

-----  
ref\_sem      セマフォの状態参照〔T〕

**【C言語API】**

ER ercd = ref\_sem(ID semid, T\_RSEM \*pk\_rsem)

**【パラメータ】**

ID            semid          対象セマフォのID番号  
T\_RSEM \*    pk\_rsem       セマフォの現在状態を入れるパケットへのポインタ

**【リターンパラメータ】**

ER            ercd            正常終了 (E\_OK) またはエラーコード

\* セマフォの現在状態 (パケットの内容)

ID	wtskid	セマフォの待ち行列の先頭のタスクのID番号
uint_t	semcnt	セマフォの資源数

#### 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
E_ID	不正ID番号 (semidが不正)
E_NOEXS [D]	オブジェクト未登録 (対象セマフォが未登録)
E_OACV [P]	オブジェクトアクセス違反 (対象セマフォに対する参照操作が許可されていない)
E_MACV [P]	メモリアクセス違反 (pk_rsemが指すメモリ領域への書込みアクセスが許可されていない)

#### 【機能】

semidで指定したセマフォ (対象セマフォ) の現在状態を参照する。参照した現在状態は, pk\_rsemで指定したパケットに返される。

対象セマフォの待ち行列にタスクが存在しない場合, wtskidにはTSK\_NONE (= 0) が返る。

#### 【使用上の注意】

ref\_semはデバッグ時向けの機能であり, その他の目的に使用することは推奨しない。これは, ref\_semを呼び出し, 対象セマフォの現在状態を参照した直後に割込みが発生した場合, ref\_semから戻ってきた時には対象セマフォの状態が変化している可能性があるためである。

#### 4.4.2 イベントフラグ

イベントフラグは, イベントの発生の有無を表すビットの集合 (ビットパターン) を介して, イベント通知を行うためのカーネルオブジェクトである。イベントが発生している状態を1, 発生していない状態を0とし, ビットパターンにより複数のイベントの発生の有無を表す。イベントフラグは, イベントフラグIDと呼ぶID番号によって識別する。

1つまたは複数のビットをセットする1にする (セットする) ことを, イベントフラグをセットするといひ, 0にする (クリアする) ことを, イベントフラグをクリアするという。イベントフラグによりイベントを通知する側のタスクは, イベントフラグをセットまたはクリアすることで, イベントの発生を通知する。

イベントフラグによりイベントの通知を受ける側のタスクは, 待ちビットパターンと待ちモードにより, どのビットがセットされるのを待つかを指定する。待ちモードにTWF\_ORWを指定した場合, 待ちビットパターンに含まれるいずれかのビットがセットされるのを待つ。待ちモードにTWF\_ANDWを指定した場合, 待ちビットパターンに含まれるすべてのビットがセットされるのを待つ。この条件を, イベントフラグの待ち解除の条件と呼ぶ。

各イベントフラグが持つ情報は次の通り。

- ・ イベントフラグ属性
- ・ ビットパターン (の現在値)
- ・ 待ち行列 (イベントフラグ待ち状態のタスクのキュー)
- ・ 初期ビットパターン (ビットパターンの初期値)

待ち行列は、イベントフラグが指定した待ち解除の条件を満たすまで待っている状態（イベントフラグ待ち状態）のタスクが繋がれているキューである。待ち行列につながれたタスクの待ち解除は、待ち解除の条件を満たした中で、待ち行列の前方につながれたものから順に行われる（「2.6.4 待ち行列と待ち解除の順序」の節の(a)に該当）。

イベントフラグの初期ビットパターンは、イベントフラグを生成または再初期化した際の、ビットパターンの初期値である。

イベントフラグ属性には、次の属性を指定することができる。

TA_TPRI	0x01U	待ち行列をタスクの優先度順にする
TA_WMUL	0x02U	複数のタスクが待つのを許す
TA_CLR	0x04U	タスクの待ち解除時にイベントフラグをクリアする

TA\_TPRIを指定しない場合、待ち行列はFIFO順になる。TA\_WMULを指定しない場合、1つのイベントフラグに複数のタスクが待つことを禁止する。

TA\_CLRを指定した場合、タスクの待ち解除時に、イベントフラグのビットパターンを0にクリアする。TA\_CLRを指定しない場合、タスクの待ち解除時にイベントフラグをクリアしない。

イベントフラグ機能に用いるデータ型は次の通り。

FLGPTN	イベントフラグのビットパターン（符号無し整数，uint_tに定義）
--------	-----------------------------------

イベントフラグ機能に関連するカーネル構成マクロは次の通り。

TBIT_FLGPTN	イベントフラグのビット数（FLGPTNの有効ビット数）
TNUM_FLGID	登録できるイベントフラグの数（動的生成対応でないカーネルでは、静的APIによって登録されたイベントフラグの数に一致）

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、イベントフラグのビット数（TBIT\_FLGPTN）は16以上である。

#### 【μITRON4.0仕様との関係】

TNUM\_FLGIDは、μITRON4.0仕様に規定されていないカーネル構成マクロである。

---

CRE_FLG	イベントフラグの生成〔S〕
acre_flg	イベントフラグの生成〔TD〕

#### 【静的API】

```
CRE_FLG(ID flgid, { ATR flgatr, FLGPTN iflgptn })
```

#### 【C言語API】

```
ER_ID flgid = acre_flg(const T_CFLG *pk_cflg)
```

#### 【パラメータ】

ID	flgid	生成するイベントフラグのID番号（acre_flgを除く）
T_CFLG *	pk_cflg	イベントフラグの生成情報を入れたパケットへのポインタ（静的APIを除く）



\* イベントフラグの生成情報 (パケットの内容)

ATR	flgatr	イベントフラグ属性
FLGPTN	iflgptn	イベントフラグの初期ビットパターン

#### 【リターンパラメータ】

ER_ID	flgid	生成されたイベントフラグのID番号 (正の値) またはエラーコード
-------	-------	--------------------------------------

#### 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
E_RSATR	予約属性 (flgatrが不正または使用できない)
E_OACV [P]	オブジェクトアクセス違反 (システム状態に対する管理操作が許可されていない)
E_MACV [P]	メモリアクセス違反 (pk_flgが指すメモリ領域への読出しアクセスが許可されていない)
E_NOID	ID番号不足 (割り付けられるイベントフラグIDがない: acre_flgの場合)
E_OBJ	オブジェクト状態エラー (flgidで指定したイベントフラグが登録済み: acre_flgを除く)

#### 【機能】

各パラメータで指定したイベントフラグ生成情報に従って, イベントフラグを生成する. 生成されたイベントフラグのビットパターンは初期ビットパターンに, 待ち行列は空の状態に初期化される.

静的APIにおいては, flgidはオブジェクト識別名, iflgptnは整数定数式パラメータである.

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは, CRE\_FLGのみをサポートする.

---

SAC_FLG	イベントフラグのアクセス許可ベクタの設定 [SP]
sac_flg	イベントフラグのアクセス許可ベクタの設定 [TPD]

#### 【静的API】

```
SAC_FLG(ID flgid, { ACPTN acptn1, ACPTN acptn2,
                  ACPTN acptn3, ACPTN acptn4 })
```

#### 【C言語API】

```
ER ercd = sac_flg(ID flgid, const ACVCT *p_acvct)
```

#### 【パラメータ】

ID	flgid	対象イベントフラグのID番号
ACVCT *	p_acvct	アクセス許可ベクタを入れたパケットへのポインタ (静的APIを除く)

\* アクセス許可ベクタ (パケットの内容)

ACPTN	acptn1	通常操作1のアクセス許可パターン
ACPTN	acptn2	通常操作2のアクセス許可パターン
ACPTN	acptn3	管理操作のアクセス許可パターン
ACPTN	acptn4	参照操作のアクセス許可パターン

#### 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

## 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し，CPUロック状態からの呼出し）
E_ID	不正ID番号（flgidが不正）
E_NOEXS [D]	オブジェクト未登録（対象イベントフラグが未登録）
E_OACV [P]	オブジェクトアクセス違反（対象イベントフラグに対する管理操作が許可されていない）
E_MACV [P]	メモリアクセス違反（p_acvctが指すメモリ領域への読出しアクセスが許可されていない）
E_OBJ	オブジェクト状態エラー（対象イベントフラグは静的APIで生成された：静的APIを除く）

## 【機能】

flgidで指定したイベントフラグ（対象イベントフラグ）のアクセス許可ベクタ（4つのアクセス許可パターンの組）を，各パラメータで指定した値に設定する．

## 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは，SAC\_FLG，sac\_flgをサポートしない．

-----  
del\_flg イベントフラグの削除 [TD]

## 【C言語API】

```
ER ercd = del_flg(ID flgid)
```

## 【パラメータ】

ID	flgid	対象イベントフラグのID番号
----	-------	----------------

## 【リターンパラメータ】

ER	ercd	正常終了（E_OK）またはエラーコード
----	------	---------------------

## 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し，CPUロック状態からの呼出し）
E_ID	不正ID番号（flgidが不正）
E_NOEXS	オブジェクト未登録（対象イベントフラグが未登録）
E_OACV [P]	オブジェクトアクセス違反（対象イベントフラグに対する管理操作が許可されていない）
E_OBJ	オブジェクト状態エラー（対象イベントフラグは静的APIで生成された）

## 【機能】

flgidで指定したイベントフラグ（対象イベントフラグ）を削除する．具体的な振舞いは以下の通り．

対象イベントフラグの登録が解除され，そのイベントフラグIDが未使用の状態に戻される．また，対象イベントフラグの待ち行列につながれたタスクは，待ち行列の先頭のタスクから順に待ち解除される．待ち解除されたタスクには，待ち状態となったサービスコールからE\_DLTエラーが返る．

## 【使用上の注意】

del\_flgにより複数のタスクが待ち解除される場合，サービスコールの処理時間およびカーネル内での割り込み禁止時間が，待ち解除されるタスクの数に比例して長くなる．特に，多くのタスクが待ち解除される場合，カーネル内での割り込み禁止時間が長くなるため，注意が必要である．

## 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、del\_flgをサポートしない。

-----  
 set\_flg イベントフラグのセット [T]  
 iset\_flg イベントフラグのセット [I]

## 【C言語API】

ER ercd = set\_flg(ID flgid, FLGPTN setptn)  
 ER ercd = iset\_flg(ID flgid, FLGPTN setptn)

## 【パラメータ】

ID	flgid	対象イベントフラグのID番号
FLGPTN	setptn	セットするビットパターン

## 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

## 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し: set_flgの場合, タスクコンテキストからの呼出し: iset_flgの場合, CPUロック状態からの呼出し)
E_ID	不正ID番号 (flgidが不正)
E_NOEXS [D]	オブジェクト未登録 (対象イベントフラグが未登録)
E_OACV [P]	オブジェクトアクセス違反 (対象イベントフラグに対する通常操作1が許可されていない: set_flgの場合)

## 【機能】

flgidで指定したイベントフラグ (対象イベントフラグ) のsetptnで指定したビットをセットする。具体的な振舞いは以下の通り。

対象イベントフラグのビットパターンは、それまでの値とsetptnで指定した値のビット毎論理和 (C言語の"|") に更新される。対象イベントフラグの待ち行列にタスクが存在する場合には、待ち解除の条件を満たしたタスクが、待ち行列の前方につながれたものから順に待ち解除される。待ち解除されたタスクには、待ち状態となったサービスコールからE\_OKが返る。

ただし、対象イベントフラグがTA\_CLR属性である場合には、待ち解除の条件を満たしたタスクを1つ待ち解除した時点で、対象イベントフラグのビットパターンが0にクリアされるため、他のタスクが待ち解除されることはない。

## 【使用上の注意】

対象イベントフラグが、WA\_WMUL属性であり、TA\_CLR属性でない場合、set\_flgまたはiset\_flgにより複数のタスクが待ち解除される場合がある。この場合、サービスコールの処理時間およびカーネル内での割込み禁止時間が、待ち解除されるタスクの数に比例して長くなる。特に、多くのタスクが待ち解除される場合、カーネル内での割込み禁止時間が長くなるため、注意が必要である。

-----  
 clr\_flg イベントフラグのクリア [T]

## 【C言語API】

ER ercd = clr\_flg(ID flgid, FLGPTN clrptn)

## 【パラメータ】

ID	flgid	対象イベントフラグのID番号
----	-------	----------------

FLGPTN	clrptn	クリアするビットパターン（クリアしないビットを1, クリアするビットを0とする）
--------	--------	--

## 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

## 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し, CPUロック状態からの呼出し）
E_ID	不正ID番号 (flgidが不正)
E_NOEXS [D]	オブジェクト未登録（対象イベントフラグが未登録）
E_OACV [P]	オブジェクトアクセス違反（対象イベントフラグに対する通常操作1が許可されていない: clr_flgの場合）

## 【機能】

flgidで指定したイベントフラグ（対象イベントフラグ）のsetptnで指定したビットをクリアする。対象イベントフラグのビットパターンは、それまでの値とclrptnで指定した値のビット毎論理積（C言語の"&"）に更新される。

```
-----
wai_flg   イベントフラグ待ち [T]
pol_flg   イベントフラグ待ち（ポーリング） [T]
twai_flg  イベントフラグ待ち（タイムアウト付き） [T]
```

## 【C言語API】

```
ER ercd = wai_flg(ID flgid, FLGPTN waipn, MODE wfmode, FLGPTN *p_flgptn)
ER ercd = pol_flg(ID flgid, FLGPTN waipn, MODE wfmode, FLGPTN *p_flgptn)
ER ercd = twai_flg(ID flgid, FLGPTN waipn,
                   MODE wfmode, FLGPTN *p_flgptn, TMO tmout)
```

## 【パラメータ】

ID	flgid	対象イベントフラグのID番号
FLGPTN	waipn	待ちビットパターン
MODE	wfmode	待ちモード
FLGPTN *	p_flgptn	待ち解除時のビットパターンを入れるメモリ領域へのポインタ
TMO	tmout	タイムアウト時間 (twai_flgの場合)

## 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
FLGPTN	flgptn	待ち解除時のビットパターン

## 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し, CPUロック状態からの呼出し, ディスパッチ保留状態からの呼出し: pol_flgを除く）
E_ID	不正ID番号 (flgidが不正)
E_PAR	パラメータエラー (tmoutが不正: twai_flgの場合)
E_NOEXS [D]	オブジェクト未登録（対象イベントフラグが未登録）
E_OACV [P]	オブジェクトアクセス違反（対象イベントフラグに対する通常操作2が許可されていない）
E_MACV [P]	メモリアクセス違反 (p_flgptnが指すメモリ領域への書き込みアクセスが許可されていない)
E_ILUSE	サービスコール不正使用 (TA_WSGL属性のイベントフラグで待ちタスクあり)
E_TMOUT	ポーリング失敗またはタイムアウト (wai_flgを除く)
E_RLWAI	待ち禁止状態または待ち状態の強制解除 (pol_flgを除く)
E_DLT	待ちオブジェクトの削除または再初期化 (pol_flgを除く)

## 【機能】

flgidで指定したイベントフラグ（対象イベントフラグ）が，waiptnとwfmodeで指定した待ち解除の条件を満たすのを待つ．具体的な振舞いは以下の通り．

対象イベントフラグが，waiptnとwfmodeで指定した待ち解除の条件を満たしている場合には，対象イベントフラグのビットパターンの現在値がflgptnに返される．対象イベントフラグがTA\_CLR属性である場合には，対象イベントフラグのビットパターンが0にクリアされる．

待ち解除の条件を満たさない場合には，自タスクはイベントフラグ待ち状態となり，対象イベントフラグの待ち行列につながる．

-----  
ini\_flg イベントフラグの再初期化〔T〕

## 【C言語API】

```
ER ercd = ini_flg(ID flgid)
```

## 【パラメータ】

ID	flgid	対象イベントフラグのID番号
----	-------	----------------

## 【リターンパラメータ】

ER	ercd	正常終了（E_OK）またはエラーコード
----	------	---------------------

## 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し，CPUロック状態からの呼出し）
E_ID	不正ID番号（flgidが不正）
E_NOEXS〔D〕	オブジェクト未登録（対象イベントフラグが未登録）
E_OACV〔P〕	オブジェクトアクセス違反（対象イベントフラグに対する管理操作が許可されていない）

## 【機能】

flgidで指定したイベントフラグ（対象イベントフラグ）を再初期化する．具体的な振舞いは以下の通り．

対象イベントフラグのビットパターンは，初期ビットパターンに初期化される．また，対象イベントフラグの待ち行列につながれたタスクは，待ち行列の先頭のタスクから順に待ち解除される．待ち解除されたタスクには，待ち状態となったサービスコールからE\_DLTエラーが返る．

## 【使用上の注意】

ini\_flgにより複数のタスクが待ち解除される場合，サービスコールの処理時間およびカーネル内での割り込み禁止時間が，待ち解除されるタスクの数に比例して長くなる．特に，多くのタスクが待ち解除される場合，カーネル内での割り込み禁止時間が長くなるため，注意が必要である．

イベントフラグを再初期化した場合に，アプリケーションとの整合性を保つのは，アプリケーションの責任である．

## 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである．

-----  
ref\_flg イベントフラグの状態参照〔T〕

## 【C言語API】

```
ER ercd = ref_flg(ID flgid, T_RFLG *pk_rflg)
```

## 【パラメータ】

ID	flgid	対象イベントフラグのID番号
T_RFLG *	pk_rflg	イベントフラグの現在状態を入れるパケットへのポインタ

## 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

\* イベントフラグの現在状態 (パケットの内容)

ID	wtskid	イベントフラグの待ち行列の先頭のタスクのID番号
uint_t	flgcnt	イベントフラグのビットパターン

## 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
E_ID	不正ID番号 (flgidが不正)
E_NOEXS [D]	オブジェクト未登録 (対象イベントフラグが未登録)
E_OACV [P]	オブジェクトアクセス違反 (対象イベントフラグに対する参照操作が許可されていない)
E_MACV [P]	メモリアクセス違反 (pk_rflgが指すメモリ領域への書込みアクセスが許可されていない)

## 【機能】

flgidで指定したイベントフラグ (対象イベントフラグ) の現在状態を参照する。参照した現在状態は, pk\_rflgで指定したパケットに返される。

対象イベントフラグの待ち行列にタスクが存在しない場合, wtskidには TSK\_NONE (= 0) が返る。

## 【使用上の注意】

ref\_flgはデバッグ時向けの機能であり, その他の目的に使用することは推奨しない。これは, ref\_flgを呼び出し, 対象イベントフラグの現在状態を参照した直後に割込みが発生した場合, ref\_flgから戻ってきた時には対象イベントフラグの状態が変化している可能性があるためである。

## 4.4.3 データキュー

データキューは, 1ワードのデータをメッセージとして, FIFO順で送受信するためカーネルオブジェクトである。より大きいサイズのメッセージを送受信したい場合には, メッセージを置いたメモリ領域へのポインタを1ワードのデータとして送受信する方法がある。データキューは, データキューIDと呼ぶID番号によって識別する。

各データキューが持つ情報は次の通り。

- ・データキュー属性
- ・データキュー管理領域
- ・送信待ち行列 (データキューへの送信待ち状態のタスクのキュー)
- ・受信待ち行列 (データキューからの受信待ち状態のタスクのキュー)

データキュー管理領域は、データキューに送信されたデータを、送信された順に格納しておくためのメモリ領域である。データキュー生成時に、データキュー管理領域に格納できるデータ数を0とすることで、データキュー管理領域のサイズを0とすることができる。

送信待ち行列は、データキューに対してデータが送信できるまで待っている状態（データキューへの送信待ち状態）のタスクが、データを送信できる順序でつながれているキューである。また、受信待ち行列は、データキューからデータが受信できるまで待っている状態（データキューからの受信待ち状態）のタスクが、データを受信できる順序でつながれているキューである。

データキュー属性には、次の属性を指定することができる。

TA\_TPRI 0x01U 送信待ち行列をタスクの優先度順にする

TA\_TPRIを指定しない場合、送信待ち行列はFIFO順になる。受信待ち行列は、FIFO順に固定されている。

データキュー機能に関連するカーネル構成マクロは次の通り。

TNUM\_DTQID 登録できるデータキューの数（動的生成対応でないカーネルでは、静的APIによって登録されたデータキューの数に一致）

#### 【μITRON4.0仕様との関係】

TNUM\_DTQIDは、μITRON4.0仕様に規定されていないカーネル構成マクロである。

-----  
 CRE\_DTQ データキューの生成〔S〕  
 acre\_dtq データキューの生成〔TD〕

#### 【静的API】

CRE\_DTQ(ID dtqid, { ATR dtqatr, uint\_t dtqcnt, void \*dtqmb })

#### 【C言語API】

ER\_ID dtqid = acre\_dtq(const T\_CDTQ \*pk\_cdtq)

#### 【パラメータ】

ID	dtqid	生成するデータキューのID番号（acre_dtqを除く）
T_CDTQ *	pk_cdtq	データキューの生成情報を入れたパケットへのポインタ（静的APIを除く）

\* データキューの生成情報（パケットの内容）

ATR	dtqatr	データキュー属性
uint_t	dtqcnt	データキュー管理領域に格納できるデータ数
void *	dtqmb	データキュー管理領域の先頭番地

#### 【リターンパラメータ】

ER_ID	dtqid	生成されたデータキューのID番号（正の値）またはエラーコード
-------	-------	--------------------------------

#### 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し、CPUロック状態からの呼出し）
E_RSATR	予約属性（dtqatrが不正または使用できない）
E_NOSPT	未サポート機能（dtqmbがサポートされていない値）
E_PAR	パラメータエラー（dtqmbが不正）

E_OACV [P]	オブジェクトアクセス違反 (システム状態に対する管理操作が許可されていない)
E_MACV [P]	メモリアクセス違反 (pk_cdtqが指すメモリ領域への読みアクセスが許可されていない)
E_NOID	ID番号不足 (割り付けられるデータキューIDがない: acre_dtqの場合)
E_OBJ	オブジェクト状態エラー (dtqidで指定したデータキューが登録済み: acre_dtqを除く)

### 【機能】

各パラメータで指定したデータキュー生成情報に従って、データキューを生成する。dtqcntとdtqmbからデータキュー管理領域が設定され、格納されているデータがない状態に初期化される。また、送信待ち行列と受信待ち行列は、空の状態に初期化される。

静的APIにおいては、dtqidはオブジェクト識別名、dtqcntは整数定数式パラメータ、dtqmbは一般定数式パラメータである。

dtqmbをNULLとした場合、dtqcntで指定した数のデータを格納できるデータキュー管理領域を、コンフィギュレータまたはカーネルが確保する。

### 【未決定事項】

dtqmbがNULLでない場合に、データキュー管理領域をアプリケーションで確保する方法については、今後の課題である。

保護機能対応カーネルにおけるデータキュー管理領域の指定方法と確保方法については、今後の課題である。

### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、CRE\_DTQのみをサポートする。また、dtqmbにはNULLのみを指定することができる。NULL以外を指定した場合には、E\_NOSPTエラーとなる。

### 【μITRON4.0仕様との関係】

μITRON4.0/PX仕様にあわせて、データキュー生成情報の最後のパラメータを、dtq (データキュー領域の先頭番地) から、dtqmb (データキュー管理領域の先頭番地) に改名した。

-----  
SAC\_DTQ      データキューのアクセス許可ベクタの設定 [SP]  
sac\_dtq      データキューのアクセス許可ベクタの設定 [TPD]

### 【静的API】

```
SAC_DTQ(ID dtqid, { ACPTN acptn1, ACPTN acptn2,
                    ACPTN acptn3, ACPTN acptn4 })
```

### 【C言語API】

```
ER ercd = sac_dtq(ID dtqid, const ACVCT *p_acvct)
```

### 【パラメータ】

ID	dtqid	対象データキューのID番号
ACVCT *	p_acvct	アクセス許可ベクタを入れたパケットへのポインタ (静的APIを除く)

\* アクセス許可ベクタ (パケットの内容)

ACPTN	acptn1	通常操作1のアクセス許可パターン
-------	--------	------------------



ACPTN	acptn2	通常操作2のアクセス許可パターン
ACPTN	acptn3	管理操作のアクセス許可パターン
ACPTN	acptn4	参照操作のアクセス許可パターン

## 【リターンパラメータ】

ER ercd 正常終了 (E\_OK) またはエラーコード

## 【エラーコード】

E\_CTX コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)  
 E\_ID 不正ID番号 (dtqidが不正)  
 E\_NOEXS [D] オブジェクト未登録 (対象データキューが未登録)  
 E\_OACV [P] オブジェクトアクセス違反 (対象データキューに対する管理操作が許可されていない)  
 E\_MACV [P] メモリアクセス違反 (p\_acvctが指すメモリ領域への読出しアクセスが許可されていない)  
 E\_OBJ オブジェクト状態エラー (対象データキューは静的APIで生成された: 静的APIを除く)

## 【機能】

dtqidで指定したデータキュー (対象データキュー) のアクセス許可ベクタ (4つのアクセス許可パターンの組) を, 各パラメータで指定した値に設定する.

## 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは, SAC\_DTQ, sac\_dtqをサポートしない.

-----  
 del\_dtq データキューの削除 [TD]

## 【C言語API】

ER ercd = del\_dtq(ID dtqid)

## 【パラメータ】

ID dtqid 対象データキューのID番号

## 【リターンパラメータ】

ER ercd 正常終了 (E\_OK) またはエラーコード

## 【エラーコード】

E\_CTX コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)  
 E\_ID 不正ID番号 (dtqidが不正)  
 E\_NOEXS オブジェクト未登録 (対象データキューが未登録)  
 E\_OACV [P] オブジェクトアクセス違反 (対象データキューに対する管理操作が許可されていない)  
 E\_OBJ オブジェクト状態エラー (対象データキューは静的APIで生成された)

## 【機能】

dtqidで指定したデータキュー (対象データキュー) を削除する. 具体的な振舞いは以下の通り.

対象データキューの登録が解除され, そのデータキューIDが未使用の状態に戻る. また, 対象データキューの送信待ち行列と受信待ち行列につながれたタスクは, それぞれの待ち行列の先頭のタスクから順に待ち解除される. 待ち解除されたタスクには, 待ち状態となったサービスコールからE\_DLTエラーが返

る。

データキューの生成時に、データキュー管理領域がカーネルによって確保された場合は、そのメモリ領域が解放される。

#### 【補足説明】

送信待ち行列と受信待ち行列の両方にタスクがつながれていることはないため、別の待ち行列で待っていたタスクの間の待ち解除の順序は、規定する必要がない。

#### 【使用上の注意】

del\_dtqにより複数のタスクが待ち解除される場合、サービスコールの処理時間およびカーネル内での割込み禁止時間が、待ち解除されるタスクの数に比例して長くなる。特に、多くのタスクが待ち解除される場合、カーネル内での割込み禁止時間が長くなるため、注意が必要である。

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、del\_dtqをサポートしない。

```
-----
snd_dtq      データキューへの送信〔T〕
psnd_dtq     データキューへの送信（ポーリング）〔T〕
ipsnd_dtq    データキューへの送信（ポーリング）〔I〕
tsnd_dtq     データキューへの送信（タイムアウト付き）〔T〕
```

#### 【C言語API】

```
ER ercd = snd_dtq(ID dtqid, intptr_t data)
ER ercd = psnd_dtq(ID dtqid, intptr_t data)
ER ercd = ipsnd_dtq(ID dtqid, intptr_t data)
ER ercd = tsnd_dtq(ID dtqid, intptr_t data, TMO tmout)
```

#### 【パラメータ】

ID	dtqid	対象データキューのID番号
intptr_t	data	送信データ
TMO	tmout	タイムアウト時間（tsnd_dtqの場合）

#### 【リターンパラメータ】

ER	ercd	正常終了（E_OK）またはエラーコード
----	------	---------------------

#### 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し：ipsnd_dtqを除く、タスクコンテキストからの呼出し：ipsnd_dtqの場合、CPUロック状態からの呼出し）
E_ID	不正ID番号（dtqidが不正）
E_PAR	パラメータエラー（tmoutが不正：tsnd_dtqの場合）
E_NOEXS〔D〕	オブジェクト未登録（対象データキューが未登録）
E_OACV〔P〕	オブジェクトアクセス違反（対象データキューに対する通常操作1が許可されていない：ipsnd_dtqを除く）
E_TMOUT	ポーリング失敗またはタイムアウト（snd_dtqを除く）
E_RLWAI	待ち禁止状態または待ち状態の強制解除（snd_dtqとtsnd_dtqの場合）
E_DLT	待ちオブジェクトの削除または再初期化（snd_dtqとtsnd_dtqの場合）

#### 【機能】

dtqidで指定したデータキュー（対象データキュー）に，dataで指定したデータを送信する．具体的な振舞いは以下の通り．

対象データキューの受信待ち行列にタスクが存在する場合には，受信待ち行列の先頭のタスクが，dataで指定したデータを受信し，待ち解除される．待ち解除されたタスクには，待ち状態となったサービスコールからE\_OKが返る．

対象データキューの受信待ち行列にタスクが存在せず，データキュー管理領域にデータを格納するスペースがある場合には，dataで指定したデータが，FIFO順でデータキュー管理領域に格納される．

対象データキューの受信待ち行列にタスクが存在せず，データキュー管理領域にデータを格納するスペースがない場合には，自タスクはデータキューへの送信待ち状態となり，対象データキューの送信待ち行列につながる．

-----  
fsnd\_dtq データキューへの強制送信〔T〕  
ifsnd\_dtq データキューへの強制送信〔I〕

#### 【C言語API】

```
ER ercd = fsnd_dtq(ID dtqid, intptr_t data)
ER ercd = ifsnd_dtq(ID dtqid, intptr_t data)
```

#### 【パラメータ】

ID	dtqid	対象データキューのID番号
intptr_t	data	送信データ

#### 【リターンパラメータ】

ER	ercd	正常終了（E_OK）またはエラーコード
----	------	---------------------

#### 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し：fsnd_dtqの場合，タスクコンテキストからの呼出し：ifsnd_dtqの場合，CPUロック状態からの呼出し）
E_ID	不正ID番号（dtqidが不正）
E_NOEXS〔D〕	オブジェクト未登録（対象データキューが未登録）
E_OACV〔P〕	オブジェクトアクセス違反（対象データキューに対する通常操作1が許可されていない：fsnd_dtqの場合）
E_ILUSE	サービスコール不正使用（対象データキューのデータキュー管理領域のサイズが0）

#### 【機能】

dtqidで指定したデータキュー（対象データキュー）に，dataで指定したデータを強制送信する．具体的な振舞いは以下の通り．

対象データキューの受信待ち行列にタスクが存在する場合には，受信待ち行列の先頭のタスクが，dataで指定したデータを受信し，待ち解除される．待ち解除されたタスクには，待ち状態となったサービスコールからE\_OKが返る．

対象データキューの受信待ち行列にタスクが存在せず，データキュー管理領域にデータを格納するスペースがある場合には，dataで指定したデータが，FIFO順でデータキュー管理領域に格納される．

対象データキューの受信待ち行列にタスクが存在せず，データキュー管理領域にデータを格納するスペースがない場合には，データキュー管理領域の先頭に格納されたデータを削除し，空いたスペースを用いて，dataで指定したデータが，FIFO順でデータキュー管理領域に格納される．

対象データキューのデータキュー管理領域のサイズが0の場合には、E\_ILUSEエラーとなる。

```
-----
rcv_dtq    データキューからの受信 [T]
prcv_dtq   データキューからの受信 (ポーリング) [T]
trcv_dtq   データキューからの受信 (タイムアウト付き) [T]
```

#### 【C言語API】

```
ER ercd = rcv_dtq(ID dtqid, intptr_t *p_data)
ER ercd = prcv_dtq(ID dtqid, intptr_t *p_data)
ER ercd = trcv_dtq(ID dtqid, intptr_t *p_data, TMO tmout)
```

#### 【パラメータ】

ID	dtqid	対象データキューのID番号
intptr_t *	p_data	受信データを入れるメモリ領域へのポインタ
TMO	tmout	タイムアウト時間 (trcv_dtqの場合)

#### 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
intptr_t	data	受信データ

#### 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し、CPUロック状態からの呼出し、ディスパッチ保留状態からの呼出し: prcv_dtqを除く)
E_ID	不正ID番号 (dtqidが不正)
E_PAR	パラメータエラー (tmoutが不正: trcv_dtqの場合)
E_NOEXS [D]	オブジェクト未登録 (対象データキューが未登録)
E_OACV [P]	オブジェクトアクセス違反 (対象データキューに対する通常操作2が許可されていない)
E_MACV [P]	メモリアクセス違反 (p_dataが指すメモリ領域への書き込みアクセスが許可されていない)
E_TMOUT	ポーリング失敗またはタイムアウト (rcv_dtqを除く)
E_RLWAI	待ち禁止状態または待ち状態の強制解除 (prcv_dtqを除く)
E_DLT	待ちオブジェクトの削除または再初期化 (prcv_dtqを除く)

#### 【機能】

dtqidで指定したデータキュー (対象データキュー) からデータを受信する。受信したデータは、p\_dataで指定したメモリ領域に返される。具体的な振舞いは以下の通り。

対象データキューのデータキュー管理領域にデータが格納されている場合には、データキュー管理領域の先頭に格納されたデータが取り出され、p\_dataで指定したメモリ領域に返される。また、送信待ち行列にタスクが存在する場合には、送信待ち行列の先頭のタスクの送信データが、FIFO順でデータキュー管理領域に格納され、そのタスクは待ち解除される。待ち解除されたタスクには、待ち状態となったサービスコールからE\_OKが返る。

対象データキューのデータキュー管理領域にデータが格納されておらず、送信待ち行列にタスクが存在する場合には、送信待ち行列の先頭のタスクの送信データが、p\_dataで指定したメモリ領域に返される。送信待ち行列の先頭のタスクは、待ち解除される。待ち解除されたタスクには、待ち状態となったサービスコールからE\_OKが返る。

対象データキューのデータキュー管理領域にデータが格納されておらず、送信待ち行列にタスクが存在しない場合には、自タスクはデータキューからの受信待ち状態となり、対象データキューの受信待ち行列につながる。

-----  
 ini\_dtq データキューの再初期化〔T〕

【C言語API】

ER ercd = ini\_dtq(ID dtqid)

【パラメータ】

ID dtqid 対象データキューのID番号

【リターンパラメータ】

ER ercd 正常終了 (E\_OK) またはエラーコード

【エラーコード】

E\_CTX コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)  
 E\_ID 不正ID番号 (dtqidが不正)  
 E\_NOEXS〔D〕 オブジェクト未登録 (対象データキューが未登録)  
 E\_OACV〔P〕 オブジェクトアクセス違反 (対象データキューに対する管理操作が許可されていない)

【機能】

dtqidで指定したデータキュー (対象データキュー) を再初期化する。具体的な振舞いは以下の通り。

対象データキューのデータキュー管理領域は、格納されているデータがない状態に初期化される。また、対象データキューの送信待ち行列と受信待ち行列につながれたタスクは、それぞれの待ち行列の先頭のタスクから順に待ち解除される。待ち解除されたタスクには、待ち状態となったサービスコールからE\_DLTエラーが返る。

【補足説明】

送信待ち行列と受信待ち行列の両方にタスクがつながれていることはないため、別の待ち行列で待っていたタスクの間の待ち解除の順序は、規定する必要がない。

【使用上の注意】

ini\_dtqにより複数のタスクが待ち解除される場合、サービスコールの処理時間およびカーネル内での割込み禁止時間が、待ち解除されるタスクの数に比例して長くなる。特に、多くのタスクが待ち解除される場合、カーネル内での割込み禁止時間が長くなるため、注意が必要である。

データキューを再初期化した場合に、アプリケーションとの整合性を保つのは、アプリケーションの責任である。

【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである。

-----  
 ref\_dtq データキューの状態参照〔T〕

【C言語API】

ER ercd = ref\_dtq(ID dtqid, T\_RDTQ \*pk\_rdtq)

【パラメータ】

ID dtqid 対象データキューのID番号

T_RDTQ *	pk_rdtq	データキューの現在状態を入れるパケットへのポインタ
----------	---------	---------------------------

## 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

\* データキューの現在状態 (パケットの内容)

ID	stskid	データキューの送信待ち行列の先頭のタスクのID番号
ID	rtskid	データキューの受信待ち行列の先頭のタスクのID番号
uint_t	sdtqcnt	データキュー管理領域に格納されているデータの数

## 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
E_ID	不正ID番号 (dtqidが不正)
E_NOEXS [D]	オブジェクト未登録 (対象データキューが未登録)
E_OACV [P]	オブジェクトアクセス違反 (対象データキューに対する参照操作が許可されていない)
E_MACV [P]	メモリアクセス違反 (pk_rdtqが指すメモリ領域への書込みアクセスが許可されていない)

## 【機能】

dtqidで指定したデータキュー (対象データキュー) の現在状態を参照する。参照した現在状態は、pk\_rdtqで指定したパケットに返される。

対象データキューの送信待ち行列にタスクが存在しない場合、stskidにはTSK\_NONE (= 0) が返る。また、受信待ち行列にタスクが存在しない場合、rtskidにはTSK\_NONE (= 0) が返る。

## 【使用上の注意】

ref\_dtqはデバッグ時向けの機能であり、その他の目的に使用することは推奨しない。これは、ref\_dtqを呼び出し、対象データキューの現在状態を参照した直後に割込みが発生した場合、ref\_dtqから戻ってきた時には対象データキューの状態が変化している可能性があるためである。

## 4.4.4 優先度データキュー

優先度データキューは、1ワードのデータをメッセージとして、データの優先度順で送受信するためカーネルオブジェクトである。より大きいサイズのメッセージを送受信したい場合には、メッセージを置いたメモリ領域へのポインタを1ワードのデータとして送受信する方法がある。優先度データキューは、優先度データキューIDと呼ぶID番号によって識別する。

各優先度データキューが持つ情報は次の通り。

- ・優先度データキュー属性
- ・優先度データキュー管理領域
- ・送信待ち行列 (優先度データキューへの送信待ち状態のタスクのキュー)
- ・受信待ち行列 (優先度データキューからの受信待ち状態のタスクのキュー)
- ・送信できるデータ優先度の最大値

優先度データキュー管理領域は、優先度データキューに送信されたデータを、

データの優先度順に格納しておくためのメモリ領域である。優先度データキュー生成時に、優先度データキュー管理領域に格納できるデータ数を0とすることで、優先度データキュー管理領域のサイズを0とすることができる。

送信待ち行列は、優先度データキューに対してデータが送信できるまで待っている状態（優先度データキューへの送信待ち状態）のタスクが、データを送信できる順序でつながれているキューである。また、受信待ち行列は、優先度データキューからデータが受信できるまで待っている状態（優先度データキューからの受信待ち状態）のタスクが、データを受信できる順序でつながれているキューである。

優先度データキュー属性には、次の属性を指定することができる。

TA\_TPRI 0x01U 送信待ち行列をタスクの優先度順にする

TA\_TPRIを指定しない場合、送信待ち行列はFIFO順になる。受信待ち行列は、FIFO順に固定されている。

優先度データキュー機能に関連するカーネル構成マクロは次の通り。

TMIN\_DPRI データ優先度の最小値 (= 1)  
 TMAX\_DPRI データ優先度の最大値  
 TNUM\_PDQID 登録できる優先度データキューの数（動的生成対応でないカーネルでは、静的APIによって登録された優先度データキューの数に一致）

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、TMAX\_DPRIは16に固定されている。ただし、タスク優先度拡張パッケージでは、TMAX\_DPRIを256に拡張する。

#### 【使用上の注意】

データの優先度が使われるのは、データが優先度データキュー管理領域に格納される場合のみであり、データを送信するタスクが送信待ち行列につながれている間には使われない。そのため、送信待ち行列につながれているタスクが、優先度データキュー管理領域に格納されているデータよりも高い優先度のデータを送信しようとしている場合でも、最初に送信されるのは、優先度データキュー管理領域に格納されているデータである。また、TA\_TPRI属性の優先度データキューにおいても、送信待ち行列はタスクの優先度順となり、タスクが送信しようとしているデータの優先度順となるわけではない。

#### 【μITRON4.0仕様との関係】

μITRON4.0仕様に規定されていない機能である。

-----  
 CRE\_PDQ 優先度データキューの生成〔S〕  
 acre\_pdq 優先度データキューの生成〔TD〕

#### 【静的API】

CRE\_PDQ(ID pdqid, { ATR pdqatr, uint\_t pdqcnt, PRI maxdpri, void \*pdqmb })

#### 【C言語API】

ER\_ID pdqid = acre\_pdq(const T\_CPDQ \*pk\_cpdq)

#### 【パラメータ】

ID pdqid 生成する優先度データキューのID番号 (acre\_pdq

T\_CPDQ \* pk\_cpdq を除く)  
優先度データキューの生成情報を入れたパケットへのポインタ (静的APIを除く)

\* 優先度データキューの生成情報 (パケットの内容)

ATR pdqatr 優先度データキュー属性  
uint\_t pdqcnt 優先度データキュー管理領域に格納できるデータ数  
PRI maxdpri 優先度データキューに送信できるデータ優先度の最大値  
void \* pdqmb 優先度データキュー管理領域の先頭番地

【リターンパラメータ】

ER\_ID pdqid 生成された優先度データキューのID番号 (正の値) またはエラーコード

【エラーコード】

E\_CTX コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)  
E\_RSATR 予約属性 (pdqatrが不正または使用できない)  
E\_NOSPT 未サポート機能 (pdqmbがサポートされていない値)  
E\_PAR パラメータエラー (pdqmb, maxdpriが不正)  
E\_OACV [P] オブジェクトアクセス違反 (システム状態に対する管理操作が許可されていない)  
E\_MACV [P] メモリアクセス違反 (pk\_cpdqが指すメモリ領域への読出しアクセスが許可されていない)  
E\_NOID ID番号不足 (割り付けられる優先度データキューIDがない: acre\_pdqの場合)  
E\_OBJ オブジェクト状態エラー (pdqidで指定した優先度データキューが登録済み: acre\_pdqを除く)

【機能】

各パラメータで指定した優先度データキュー生成情報に従って, 優先度データキューを生成する. pdqcntとpdqmbから優先度データキュー管理領域が設定され, 格納されているデータがない状態に初期化される. また, 送信待ち行列と受信待ち行列は, 空の状態に初期化される.

静的APIにおいては, pdqidはオブジェクト識別名, pdqcntとmaxdpriは整数定数式パラメータ, pdqmbは一般定数式パラメータである.

pdqmbをNULLとした場合, pdqcntで指定した数のデータを格納できる優先度データキュー管理領域を, コンフィギュレータまたはカーネルが確保する.

maxdpriは, TMIN\_DPRI以上, TMAX\_DPRI以下でなければならない.

【未決定事項】

pdqmbがNULLでない場合に, 優先度データキュー管理領域をアプリケーションで確保する方法については, 今後の課題である.

保護機能対応カーネルにおける優先度データキュー管理領域の指定方法と確保方法については, 今後の課題である.

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは, CRE\_PDQのみをサポートする. また, pdqmbにはNULLのみを渡すことができる. NULL以外を指定した場合には, E\_NOSPTエラーとなる.



-----  
 SAC\_PDQ 優先度データキューのアクセス許可ベクタの設定 [ SP ]  
 sac\_pdq 優先度データキューのアクセス許可ベクタの設定 [ TPD ]

**【静的API】**

SAC\_PDQ(ID pdqid, { ACPTN acptn1, ACPTN acptn2,  
 ACPTN acptn3, ACPTN acptn4 })

**【C言語API】**

ER ercd = sac\_pdq(ID pdqid, const ACVCT \*p\_acvct)

**【パラメータ】**

ID	pdqid	対象優先度データキューのID番号
ACVCT *	p_acvct	アクセス許可ベクタを入れたパケットへのポインタ (静的APIを除く)

\* アクセス許可ベクタ (パケットの内容)

ACPTN	acptn1	通常操作1のアクセス許可パターン
ACPTN	acptn2	通常操作2のアクセス許可パターン
ACPTN	acptn3	管理操作のアクセス許可パターン
ACPTN	acptn4	参照操作のアクセス許可パターン

**【リターンパラメータ】**

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

**【エラーコード】**

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
E_ID	不正ID番号 (pdqidが不正)
E_NOEXS [ D ]	オブジェクト未登録 (対象優先度データキューが未登録)
E_OACV [ P ]	オブジェクトアクセス違反 (対象優先度データキューに対する管理操作が許可されていない)
E_MACV [ P ]	メモリアクセス違反 (p_acvctが指すメモリ領域への読出しアクセスが許可されていない)
E_OBJ	オブジェクト状態エラー (対象優先度データキューは静的APIで生成された: 静的APIを除く)

**【機能】**

pdqidで指定した優先度データキュー (対象優先度データキュー) のアクセス許可ベクタ (4つのアクセス許可パターンの組) を, 各パラメータで指定した値に設定する.

**【TOPPERS/ASPカーネルにおける規定】**

ASPカーネルでは, SAC\_PDQ, sac\_pdqをサポートしない.

-----  
 del\_pdq 優先度データキューの削除 [ TD ]

**【C言語API】**

ER ercd = del\_pdq(ID pdqid)

**【パラメータ】**

ID	pdqid	対象優先度データキューのID番号
----	-------	------------------

**【リターンパラメータ】**

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

## 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し，CPUロック状態からの呼出し）
E_ID	不正ID番号（pdqidが不正）
E_NOEXS	オブジェクト未登録（対象優先度データキューが未登録）
E_OACV [ P ]	オブジェクトアクセス違反（対象優先度データキューに対する管理操作が許可されていない）
E_OBJ	オブジェクト状態エラー（対象優先度データキューは静的APIで生成された）

## 【機能】

pdqidで指定した優先度データキュー（対象優先度データキュー）を削除する．具体的な振舞いは以下の通り．

対象優先度データキューの登録が解除され，その優先度データキューIDが未使用の状態に戻される．また，対象優先度データキューの送信待ち行列と受信待ち行列につながれたタスクは，それぞれの待ち行列の先頭のタスクから順に待ち解除される．待ち解除されたタスクには，待ち状態となったサービスコールからE\_DLTエラーが返る．

優先度データキューの生成時に，優先度データキュー管理領域がカーネルによって確保された場合は，そのメモリ領域が解放される．

## 【補足説明】

送信待ち行列と受信待ち行列の両方にタスクがつながれていることはないため，別の待ち行列で待っていたタスクの間の待ち解除の順序は，規定する必要がない．

## 【使用上の注意】

del\_pdqにより複数のタスクが待ち解除される場合，サービスコールの処理時間およびカーネル内での割り込み禁止時間が，待ち解除されるタスクの数に比例して長くなる．特に，多くのタスクが待ち解除される場合，カーネル内での割り込み禁止時間が長くなるため，注意が必要である．

## 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは，del\_pdqをサポートしない．

---

snd_pdq	優先度データキューへの送信 [ T ]
psnd_pdq	優先度データキューへの送信（ポーリング） [ T ]
ipsnd_pdq	優先度データキューへの送信（ポーリング） [ I ]
tsnd_pdq	優先度データキューへの送信（タイムアウト付き） [ T ]

## 【C言語API】

```
ER ercd = snd_pdq(ID pdqid, intptr_t data, PRI datapri)
ER ercd = psnd_pdq(ID pdqid, intptr_t data, PRI datapri)
ER ercd = ipsnd_pdq(ID pdqid, intptr_t data, PRI datapri)
ER ercd = tsnd_pdq(ID pdqid, intptr_t data, PRI datapri, TMO tmout)
```

## 【パラメータ】

ID	pdqid	対象優先度データキューのID番号
intptr_t	data	送信データ
PRI	datapri	送信データの優先度
TMO	tmout	タイムアウト時間（tsnd_pdqの場合）

## 【リターンパラメータ】

ER ercd 正常終了 (E\_OK) またはエラーコード

## 【エラーコード】

E\_CTX コンテキストエラー (非タスクコンテキストからの呼出し: ipsnd\_pdqを除く, タスクコンテキストからの呼出し: ipsnd\_pdqの場合, CPUロック状態からの呼出し)

E\_ID 不正ID番号 (pdqidが不正)

E\_PAR パラメータエラー (datapriが不正, tmoutが不正: tsnd\_pdqのみ)

E\_NOEXS [D] オブジェクト未登録 (対象優先度データキューが未登録)

E\_OACV [P] オブジェクトアクセス違反 (対象優先度データキューに対する通常操作1が許可されていない: ipsnd\_pdqを除く)

E\_TMOUT ポーリング失敗またはタイムアウト (snd\_pdqを除く)

E\_RLWA1 待ち禁止状態または待ち状態の強制解除 (snd\_pdqとtsnd\_pdqの場合)

E\_DLT 待ちオブジェクトの削除または再初期化 (snd\_pdqとtsnd\_pdqの場合)

## 【機能】

pdqidで指定した優先度データキュー (対象優先度データキュー) に, dataで指定したデータを, datapriで指定した優先度で送信する. 具体的な振舞いは以下の通り.

対象優先度データキューの受信待ち行列にタスクが存在する場合には, 受信待ち行列の先頭のタスクが, dataで指定したデータを受信し, 待ち解除される. 待ち解除されたタスクには, 待ち状態となったサービスコールからE\_OKが返る.

対象優先度データキューの受信待ち行列にタスクが存在せず, 優先度データキュー管理領域にデータを格納するスペースがある場合には, dataで指定したデータが, datapriで指定したデータの優先度順で優先度データキュー管理領域に格納される.

対象優先度データキューの受信待ち行列にタスクが存在せず, 優先度データキュー管理領域にデータを格納するスペースがない場合には, 自タスクは優先度データキューへの送信待ち状態となり, 対象優先度データキューの送信待ち行列につながる.

datapriは, TMIN\_DPRI以上で, 対象データキューに送信できるデータ優先度の最大値以下でなければならない.

-----

rcv\_pdq 優先度データキューからの受信 [T]

prcv\_pdq 優先度データキューからの受信 (ポーリング) [T]

trcv\_pdq 優先度データキューからの受信 (タイムアウト付き) [T]

## 【C言語API】

ER ercd = rcv\_pdq(ID pdqid, intptr\_t \*p\_data, PRI \*p\_datapri)

ER ercd = prcv\_pdq(ID pdqid, intptr\_t \*p\_data, PRI \*p\_datapri)

ER ercd = trcv\_pdq(ID pdqid, intptr\_t \*p\_data, PRI \*p\_datapri, TMO tmout)

## 【パラメータ】

ID pdqid 対象優先度データキューのID番号

intptr\_t \* p\_data 受信データを入れるメモリ領域へのポインタ

PRI \* p\_datapri 受信データの優先度を入れるメモリ領域へのポインタ

TMO tmout タイムアウト時間 (trcv\_pdqの場合)

## 【リターンパラメータ】

ER	erccd	正常終了 (E_OK) またはエラーコード
intptr_t	data	受信データ
PRI	datapri	受信データの優先度

## 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し, ディスパッチ保留状態からの呼出し: prcv_pdqを除く)
E_ID	不正ID番号 (pdqidが不正)
E_PAR	パラメータエラー (tmoutが不正: trcv_pdqの場合)
E_NOEXS [D]	オブジェクト未登録 (対象優先度データキューが未登録)
E_OACV [P]	オブジェクトアクセス違反 (対象優先度データキューに対する通常操作2が許可されていない)
E_MACV [P]	メモリアクセス違反 (p_dataまたはp_datapriが指すメモリ領域への書き込みアクセスが許可されていない)
E_TMOUT	ポーリング失敗またはタイムアウト (rcv_pdqを除く)
E_RLWAI	待ち禁止状態または待ち状態の強制解除 (prcv_pdqを除く)
E_DLT	待ちオブジェクトの削除または再初期化 (prcv_pdqを除く)

## 【機能】

pdqidで指定した優先度データキュー (対象優先度データキュー) からデータを受信する。受信したデータはp\_dataで指定したメモリ領域に、その優先度はp\_datapriで指定したメモリ領域に返される。具体的な振舞いは以下の通り。

対象優先度データキューの優先度データキュー管理領域にデータが格納されている場合には、優先度データキュー管理領域の先頭に格納されたデータが取り出され、p\_dataで指定したメモリ領域に返される。また、その優先度がp\_datapriで指定したメモリ領域に返される。さらに、送信待ち行列にタスクが存在する場合には、送信待ち行列の先頭のタスクの送信データが、データの優先度順で優先度データキュー管理領域に格納され、そのタスクは待ち解除される。待ち解除されたタスクには、待ち状態となったサービスコールからE\_OKが返る。

対象優先度データキューの優先度データキュー管理領域にデータが格納されておらず、送信待ち行列にタスクが存在する場合には、送信待ち行列の先頭のタスクの送信データが、p\_dataで指定したメモリ領域に返される。また、その優先度がp\_datapriで指定したメモリ領域に返される。送信待ち行列の先頭のタスクは、待ち解除される。待ち解除されたタスクには、待ち状態となったサービスコールからE\_OKが返る。

対象優先度データキューの優先度データキュー管理領域にデータが格納されておらず、送信待ち行列にタスクが存在しない場合には、自タスクは優先度データキューからの受信待ち状態となり、対象優先度データキューの受信待ち行列につながる。

-----  
ini\_pdq 優先度データキューの再初期化 [T]

## 【C言語API】

```
ER erccd = ini_pdq(ID pdqid)
```

## 【パラメータ】

ID	pdqid	対象優先度データキューのID番号
----	-------	------------------

## 【リターンパラメータ】

ER	erccd	正常終了 (E_OK) またはエラーコード
----	-------	-----------------------

## 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し，CPUロック状態からの呼出し）
E_ID	不正ID番号（pdqidが不正）
E_NOEXS [D]	オブジェクト未登録（対象優先度データキューが未登録）
E_OACV [P]	オブジェクトアクセス違反（対象優先度データキューに対する管理操作が許可されていない）

## 【機能】

pdqidで指定した優先度データキュー（対象優先度データキュー）を再初期化する．具体的な振舞いは以下の通り．

対象優先度データキューの優先度データキュー管理領域は，格納されているデータがない状態に初期化される．また，対象優先度データキューの送信待ち行列と受信待ち行列につながれたタスクは，それぞれの待ち行列の先頭のタスクから順に待ち解除される．待ち解除されたタスクには，待ち状態となったサービスコールからE\_DLTエラーが返る．

## 【補足説明】

送信待ち行列と受信待ち行列の両方にタスクがつながれていることはないため，別の待ち行列で待っていたタスクの間の待ち解除の順序は，規定する必要がない．

## 【使用上の注意】

ini\_pdqにより複数のタスクが待ち解除される場合，サービスコールの処理時間およびカーネル内での割込み禁止時間が，待ち解除されるタスクの数に比例して長くなる．特に，多くのタスクが待ち解除される場合，カーネル内での割込み禁止時間が長くなるため，注意が必要である．

優先度データキューを再初期化した場合に，アプリケーションとの整合性を保つのは，アプリケーションの責任である．

-----  
ref\_pdq 優先度データキューの状態参照 [T]

## 【C言語API】

```
ER ercd = ref_pdq(ID pdqid, T_RPDQ *pk_rpdq)
```

## 【パラメータ】

ID	pdqid	対象優先度データキューのID番号
T_RPDQ *	pk_rpdq	優先度データキューの現在状態を入れるパケットへのポインタ

## 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

\* 優先度データキューの現在状態（パケットの内容）

ID	stskid	優先度データキューの送信待ち行列の先頭のタスクのID番号
ID	rtskid	優先度データキューの受信待ち行列の先頭のタスクのID番号
uint_t	spdqcnt	優先度データキュー管理領域に格納されているデータの数

## 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出
-------	---------------------------

	し, CPUロック状態からの呼出し)
E_ID	不正ID番号 (pdqidが不正)
E_NOEXS [D]	オブジェクト未登録 (対象優先度データキューが未登録)
E_OACV [P]	オブジェクトアクセス違反 (対象優先度データキューに対する参照操作が許可されていない)
E_MACV [P]	メモリアクセス違反 (pk_rpdqが指すメモリ領域への書込みアクセスが許可されていない)

### 【機能】

pdqidで指定した優先度データキュー (対象優先度データキュー) の現在状態を参照する。参照した現在状態は, pk\_rpdqで指定したバケットに返される。

対象優先度データキューの送信待ち行列にタスクが存在しない場合, stskidにはTSK\_NONE (=0) が返る。また, 受信待ち行列にタスクが存在しない場合, rtskidにはTSK\_NONE (=0) が返る。

### 【使用上の注意】

ref\_pdqはデバッグ時向けの機能であり, その他の目的に使用することは推奨しない。これは, ref\_pdqを呼び出し, 対象優先度データキューの現在状態を参照した直後に割り込みが発生した場合, ref\_pdqから戻ってきた時には対象優先度データキューの状態が変化している可能性があるためである。

## 4.4.5 メールボックス

メールボックスは, 共有メモリ上に置いたメッセージを, FIFO順またはメッセージの優先度順で送受信するためカーネルオブジェクトである。メールボックスは, メールボックスIDと呼ぶID番号によって識別する。

各メールボックスが持つ情報は次の通り。

- ・メールボックス属性
- ・メッセージキュー
- ・待ち行列 (メールボックスからの受信待ち状態のタスクのキュー)
- ・送信できるメッセージ優先度の最大値
- ・優先度別のメッセージキューヘッダ領域

メッセージキューは, メールボックスに送信されたメッセージを, FIFO順またはメッセージの優先度順につないでおくためのキューである。

待ち行列は, メールボックスからメッセージが受信できるまで待っている状態 (メールボックスからの受信待ち状態) のタスクが, メッセージを受信できる順序でつながれているキューである。

メールボックス属性には, 次の属性を指定することができる。

TA_TPRI	0x01U	待ち行列をタスクの優先度順にする
TA_MPRI	0x02U	メッセージキューをメッセージの優先度順にする

TA\_TPRIを指定しない場合, 待ち行列はFIFO順になる。TA\_MPRIを指定しない場合, メッセージキューはFIFO順になる。

優先度別のメッセージキューヘッダ領域は, TA\_MPRI属性のメールボックスに対して, メッセージキューを優先度別に設ける場合に使用する領域である。

カーネルは, メールボックスに送信されたメッセージをメッセージキューにつ

なくために、メッセージの先頭のメモリ領域を使用する。そのためアプリケーションは、メールボックスに送信するメッセージの先頭に、カーネルが利用するためのメッセージヘッダを置かなければならない。メッセージヘッダのデータ型として、メールボックス属性にTA\_MPRIが指定されているか否かにより、以下のいずれかを用いる。

T_MSG	TA_MPRI属性でないメールボックス用のメッセージヘッダ
T_MSG_PRI	TA_MPRI属性のメールボックス用のメッセージヘッダ

TA\_MPRI属性のメールボックスにメッセージを送信する場合、アプリケーションは、メッセージの優先度を、T\_MSG\_PRI型のメッセージヘッダ中のmsgpriフィールドに設定する。

メールボックス機能に関連するカーネル構成マクロは次の通り。

TMIN_MPRI	メッセージ優先度の最小値 (=1)
TMAX_MPRI	メッセージ優先度の最大値
TNUM_MBXID	登録できるメールボックスの数 (動的生成対応でないカーネルでは、静的APIによって登録されたメールボックスの数に一致)

#### 【補足説明】

TOPPERS新世代カーネルの現時点の実装では、優先度別のメッセージキューヘッダ領域は用いていない。

#### 【使用上の注意】

メールボックス機能は、 $\mu$ ITRON4.0仕様との互換性のために残した機能であり、保護機能を持ったカーネルではサポートされないため、使用することは推奨しない。メールボックス機能は、ほとんどの場合に、データキュー機能または優先度データキュー機能を用いて、メッセージを置いたメモリ領域へのポインタを送受信する方法で置き換えることができる。

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、メールボックス機能をサポートする。メッセージ優先度の最大値 (TMAX\_MPRI) は16に固定されている。ただし、タスク優先度拡張パッケージでは、TMAX\_MPRIを256に拡張する。

#### 【 $\mu$ ITRON4.0仕様との関係】

TNUM\_MBXIDは、 $\mu$ ITRON4.0仕様に規定されていないカーネル構成マクロである。

---

CRE_MBX	メールボックスの生成 [S]
acre_mbx	メールボックスの生成 [TD]

#### 【静的API】

```
CRE_MBX(ID mbxid, { ATR mbxatr, PRI maxmpri, void *mprihd })
```

#### 【C言語API】

```
ER_ID mbxid = acre_mbx(const T_CMBX *pk_cmbx)
```

#### 【パラメータ】

ID	mbxid	生成するメールボックスのID番号 (acre_mbxを除く)
T_CMBX *	pk_cmbx	メールボックスの生成情報を入れたパケットへ

## のポインタ（静的APIを除く）

## \* メールボックスの生成情報（パケットの内容）

ATR	mbxatr	メールボックス属性
PRI	maxmpri	優先度メールボックスに送信できるメッセージ優先度の最大値
void *	mprihd	優先度別のメッセージキューヘッダ領域の先頭番地

## 【リターンパラメータ】

ER_ID	mbxid	生成されたメールボックスのID番号（正の値）またはエラーコード
-------	-------	---------------------------------

## 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し，CPUロック状態からの呼出し）
E_RSATR	予約属性（mbxatrが不正または使用できない）
E_NOSPT	未サポート機能（mprihdがサポートされていない値）
E_PAR	パラメータエラー（mprihdが不正）
E_OACV [P]	オブジェクトアクセス違反（システム状態に対する管理操作が許可されていない）
E_MACV [P]	メモリアクセス違反（pk_cmbxが指すメモリ領域への読出しアクセスが許可されていない）
E_NOID	ID番号不足（割り付けられるメールボックスIDがない：acre_mbxの場合）
E_OBJ	オブジェクト状態エラー（mbxidで指定したメールボックスが登録済み：acre_mbxを除く）

## 【機能】

各パラメータで指定したメールボックス生成情報に従って，メールボックスを生成する．メッセージキューはつながれているメッセージがない状態に初期化され，mprihdとmaxmpriから優先度別のメッセージキューヘッダ領域が設定される．また，待ち行列は空の状態に初期化される．

静的APIにおいては，mbxidはオブジェクト識別名，maxmpriは整数定数式パラメータ，mprihdは一般定数式パラメータである．

mprihdをNULLとした場合，mprihdの指定に合致したサイズの優先度別のメッセージキューヘッダ領域を，コンフィギュレータまたはカーネルが確保する．

maxmpriは，TMIN\_MPRI以上，TMAX\_MPRI以下でなければならない．

## 【未決定事項】

mprihdがNULLでない場合に，優先度別のメッセージキューヘッダ領域をアプリケーションで確保する方法については，今後の課題である．

## 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは，CRE\_MBXのみをサポートする．また，mprihdにはNULLのみを渡すことができる．NULL以外を指定した場合には，E\_NOSPTエラーとなる．

-----  
del\_mbx      メールボックスの削除 [TD]

## 【C言語API】

```
ER ercd = del_mbx(ID mbxid)
```



## 【パラメータ】

ID	mbxid	対象メールボックスのID番号
----	-------	----------------

## 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

## 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
E_ID	不正ID番号 (mbxidが不正)
E_NOEXS	オブジェクト未登録 (対象メールボックスが未登録)
E_OACV [P]	オブジェクトアクセス違反 (対象メールボックスに対する管理操作が許可されていない)
E_OBJ	オブジェクト状態エラー (対象メールボックスは静的APIで生成された)

## 【機能】

mbxidで指定したメールボックス (対象メールボックス) を削除する。具体的な振舞いは以下の通り。

対象メールボックスの登録が解除され、そのメールボックスIDが未使用の状態に戻される。また、対象メールボックスの待ち行列につながれたタスクは、待ち行列の先頭のタスクから順に待ち解除される。待ち解除されたタスクには、待ち状態となったサービスコールからE\_DLTエラーが返る。

メールボックスの生成時に、優先度別のメッセージキューヘッダ領域がカーネルによって確保された場合は、そのメモリ領域が解放される。

## 【使用上の注意】

del\_mbxにより複数のタスクが待ち解除される場合、サービスコールの処理時間およびカーネル内での割込み禁止時間が、待ち解除されるタスクの数に比例して長くなる。特に、多くのタスクが待ち解除される場合、カーネル内での割込み禁止時間が長くなるため、注意が必要である。

## 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、del\_mbxをサポートしない。

-----  
snd\_mbx      メールボックスへの送信 [T]

## 【C言語API】

```
ER ercd = snd_mbx(ID mbxid, T_MSG *pk_msg)
```

## 【パラメータ】

ID	mbxid	対象メールボックスのID番号
T_MSG	*pk_msg	送信メッセージの先頭番地

## 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

## 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
E_ID	不正ID番号 (mbxidが不正)
E_PAR	パラメータエラー (メッセージヘッダ中のmsgpriが不正)
E_NOEXS [D]	オブジェクト未登録 (対象メールボックスが未登録)

E_OACV [P]	オブジェクトアクセス違反 (対象メールボックスに対する通常操作1が許可されていない)
E_MACV [P]	メモリアクセス違反 (pk_msgが指すメモリ領域への読出しアクセスが許可されていない)

### 【機能】

mbxidで指定したメールボックス (対象メールボックス) に, pk\_msgで指定したメッセージを送信する. 具体的な振舞いは以下の通り.

対象メールボックスの待ち行列にタスクが存在する場合には, 待ち行列の先頭のタスクが, pk\_msgで指定したメッセージを受信し, 待ち解除される. 待ち解除されたタスクには, 待ち状態となったサービスコールからE\_OKが返る.

対象メールボックスの待ち行列にタスクが存在しない場合には, pk\_msgで指定したメッセージが, メールボックス属性のTA\_MPRI指定の有無によって指定される順序で, メッセージキューにつなく.

対象メールボックスがTA\_MPRI属性である場合には, pk\_msgで指定したメッセージの先頭のメッセージヘッダ中のmsgpriフィールドの値が, TMIN\_MPRI以上で, 対象メールボックスに送信できるメッセージ優先度の最大値以下でなければならない.

-----  
rcv\_mbx      メールボックスからの受信 [T]  
prcv\_mbx     メールボックスからの受信 (ポーリング) [T]  
trcv\_mbx     メールボックスからの受信 (タイムアウト付き) [T]

### 【C言語API】

```
ER ercd = rcv_mbx(ID mbxid, T_MSG **ppk_msg)
ER ercd = prcv_mbx(ID mbxid, T_MSG **ppk_msg)
ER ercd = trcv_mbx(ID mbxid, T_MSG **ppk_msg, TMO tmout)
```

### 【パラメータ】

ID	mbxid	対象メールボックスのID番号
T_MSG **	ppk_msg	受信メッセージの先頭番地を入れるメモリ領域へのポインタ
TMO	tmout	タイムアウト時間 (trcv_mbxの場合)

### 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
T_MSG *	ppk_msg	受信メッセージの先頭番地

### 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し, ディスパッチ保留状態からの呼出し: prcv_mbxを除く)
E_ID	不正ID番号 (mbxidが不正)
E_PAR	パラメータエラー (tmoutが不正: trcv_mbxの場合)
E_NOEXS [D]	オブジェクト未登録 (対象メールボックスが未登録)
E_OACV [P]	オブジェクトアクセス違反 (対象メールボックスに対する通常操作2が許可されていない)
E_MACV [P]	メモリアクセス違反 (ppk_msgが指すメモリ領域への書込みアクセスが許可されていない)
E_TMOUT	ポーリング失敗またはタイムアウト (rcv_mbxを除く)
E_RLWAI	待ち禁止状態または待ち状態の強制解除 (prcv_mbxを除く)
E_DLT	待ちオブジェクトの削除または再初期化 (prcv_mbxを除く)

### 【機能】

mbxidで指定したメールボックス（対象メールボックス）からメッセージを受信する。受信したメッセージの先頭番地は、ppk\_msgで指定したメモリ領域に返される。具体的な振舞いは以下の通り。

対象メールボックスのメッセージキューにメッセージがつながれている場合には、メッセージキューの先頭につながれたメッセージが取り出され、ppk\_msgで指定したメモリ領域に返される。

対象メールボックスのメッセージキューにメッセージがつながれていない場合には、自タスクはメールボックスからの受信待ち状態となり、対象メールボックスの待ち行列につながれる。

-----  
ini\_mbx      メールボックスの再初期化 [T]

**【C言語API】**

ER ercd = ini\_mbx(ID mbxid)

**【パラメータ】**

ID            mbxid            対象メールボックスのID番号

**【リターンパラメータ】**

ER            ercd            正常終了 (E\_OK) またはエラーコード

**【エラーコード】**

E\_CTX            コンテキストエラー（非タスクコンテキストからの呼出し、CPUロック状態からの呼出し）  
E\_ID            不正ID番号（mbxidが不正）  
E\_NOEXS [D]    オブジェクト未登録（対象メールボックスが未登録）  
E\_OACV [P]    オブジェクトアクセス違反（対象メールボックスに対する管理操作が許可されていない）

**【機能】**

mbxidで指定したメールボックス（対象メールボックス）を再初期化する。具体的な振舞いは以下の通り。

対象メールボックスのメールボックス管理領域は、メッセージキューはつながれているメッセージがない状態に初期化される。また、対象メールボックスの待ち行列につながれたタスクは、待ち行列の先頭のタスクから順に待ち解除される。待ち解除されたタスクには、待ち状態となったサービスコールからE\_DLTエラーが返る。

**【使用上の注意】**

ini\_mbxにより複数のタスクが待ち解除される場合、サービスコールの処理時間およびカーネル内での割込み禁止時間が、待ち解除されるタスクの数に比例して長くなる。特に、多くのタスクが待ち解除される場合、カーネル内での割込み禁止時間が長くなるため、注意が必要である。

メールボックスを再初期化した場合に、アプリケーションとの整合性を保つのは、アプリケーションの責任である。

**【μITRON4.0仕様との関係】**

μITRON4.0仕様に定義されていないサービスコールである。

-----  
ref\_mbx      メールボックスの状態参照 [T]

## 【C言語API】

```
ER ercd = ref_mbx(ID mbxid, T_RMBX *pk_rmbx)
```

## 【パラメータ】

ID	mbxid	対象メールボックスのID番号
T_RMBX *	pk_rmbx	メールボックスの現在状態を入れるパケットへのポインタ

## 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

\* メールボックスの現在状態 (パケットの内容)

ID	wtskid	メールボックスの待ち行列の先頭のタスクのID番号
T_MSG *	pk_msg	メッセージキューの先頭につながれたメッセージの先頭番地

## 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
E_ID	不正ID番号 (mbxidが不正)
E_NOEXS [D]	オブジェクト未登録 (対象メールボックスが未登録)
E_OACV [P]	オブジェクトアクセス違反 (対象メールボックスに対する参照操作が許可されていない)
E_MACV [P]	メモリアクセス違反 (pk_rmbxが指すメモリ領域への書込みアクセスが許可されていない)

## 【機能】

mbxidで指定したメールボックス (対象メールボックス) の現在状態を参照する。参照した現在状態は, pk\_rmbxで指定したパケットに返される。

対象メールボックスの待ち行列にタスクが存在しない場合, wtskidには TSK\_NONE (=0) が返る。また, メッセージキューにメッセージが繋がっていない場合, pk\_msgにはNULLが返る。

## 【使用上の注意】

ref\_mbxはデバッグ時向けの機能であり, その他の目的に使用することは推奨しない。これは, ref\_mbxを呼び出し, 対象メールボックスの現在状態を参照した直後に割込みが発生した場合, ref\_mbxから戻ってきた時には対象メールボックスの状態が変化している可能性があるためである。

## 4.4.6 ミューテックス

未完成

ミューテックス機能に関連するカーネル構成マクロは次の通り。

TNUM_MTXID	登録できるミューテックスの数 (動的生成対応でないカーネルでは, 静的APIによって登録されたミューテックスの数に一致)
------------	--

## 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは, ミューテックス機能拡張パッケージを用いると, ミューテッ

クス機能を追加することができる。ただし、TA\_INHERIT属性のミューテックスはサポートしていない。

【μITRON4.0仕様との関係】

TNUM\_MTXIDは、μITRON4.0仕様に規定されていないカーネル構成マクロである。

-----  
 CRE\_MTX ミューテックスの生成〔S〕  
 acre\_mtx ミューテックスの生成〔TD〕

【静的API】

CRE\_MTX(ID mtxid, { ATR mtxatr, PRI ceilpri })

【C言語API】

ER\_ID mtxid = acre\_mtx(const T\_CMTX \*pk\_cmtx)

未完成

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルのミューテックス機能拡張パッケージでは、CRE\_MTXのみをサポートする。

-----  
 SAC\_MTX ミューテックスのアクセス許可ベクタの設定〔SP〕  
 sac\_mtx ミューテックスのアクセス許可ベクタの設定〔TPD〕

【静的API】

SAC\_MTX(ID mtxid, { ACPTN acptn1, ACPTN acptn2,  
 ACPTN acptn3, ACPTN acptn4 })

【C言語API】

ER ercd = sac\_mtx(ID mtxid, const ACVCT \*p\_acvct)

未完成

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルのミューテックス機能拡張パッケージでは、SAC\_MTX、sac\_mtxをサポートしない。

-----  
 del\_mtx ミューテックスの削除〔TD〕

【C言語API】

ER ercd = del\_mtx(ID mtxid)

未完成

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルのミューテックス機能拡張パッケージでは、del\_mtxをサポートしない。

-----  
 loc\_mtx ミューテックスのロック〔T〕  
 ploc\_mtx ミューテックスのロック（ポーリング）〔T〕  
 tloc\_mtx ミューテックスのロック（タイムアウト付き）〔T〕

【C言語API】

ER ercd = loc\_mtx(ID mtxid)

```
ER ercd = ploc_mtx(ID mtxid)
ER ercd = tloc_mtx(ID mtxid, TMO tmout)
```

-----  
unl\_mtx ミューテックスのロック解除 [T]

未完成

【C言語API】

```
ER ercd = unl_mtx(ID mtxid)
```

-----  
ini\_mtx ミューテックスの初期化 [T]

【C言語API】

```
ER ercd = ini_mtx(ID mtxid)
```

未完成

【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである。

-----  
ref\_mtx ミューテックスの状態参照 [T]

【C言語API】

```
ER ercd = ref_mtx(ID mtxid, T_RMTX *pk_rmtx)
```

未完成  
-----

#### 4.4.7 メッセージバッファ

未完成

#### 4.5 メモリプール管理機能

【μITRON4.0仕様との関係】

この仕様では、可変長メモリプール機能はサポートしないこととした。

【仕様決定の理由】

可変長メモリプール機能をサポートしないこととしたのは、メモリ割付けの処理時間とフラグメンテーションの発生を考えると、最適なメモリ管理アルゴリズムはアプリケーション依存となるため、カーネル内で実現するより、ライブラリとして実現する方が適切と考えたためである。

##### 4.5.1 固定長メモリプール

固定長メモリプールは、生成時に決めたサイズのメモリブロック（固定長メモリブロック）を動的に獲得・解放するためのカーネルオブジェクトである。固定長メモリプールは、固定長メモリプールIDと呼ぶID番号で識別する。

各固定長メモリプールが持つ情報は次の通り。

- ・固定長メモリプール属性
- ・待ち行列（固定長メモリブロックの獲得待ち状態のタスクのキュー）
- ・固定長メモリプール領域
- ・固定長メモリプール管理領域

待ち行列は、固定長メモリブロックが獲得できるまで待っている状態（固定長メモリブロックの獲得待ち状態）のタスクが、固定長メモリブロックを獲得できる順序でつながれているキューである。

固定長メモリプール領域は、その中から固定長メモリブロックを割り付けるためのメモリ領域である。

固定長メモリプール管理領域は、固定長メモリプール領域中の割当て済みの固定長メモリブロックと未割当てのメモリ領域に関する情報を格納しておくためのメモリ領域である。

固定長メモリプール属性には、次の属性を指定することができる。

TA\_TPRI 0x01U 待ち行列をタスクの優先度順にする

TA\_TPRIを指定しない場合、待ち行列はFIFO順になる。

固定長メモリプール機能に関連するカーネル構成マクロは次の通り。

TNUM\_MPFID 登録できる固定長メモリプールの数（動的生成対応でないカーネルでは、静的APIによって登録された固定長メモリプールの数に一致）

#### 【μITRON4.0仕様との関係】

固定長メモリプール領域として確保すべき領域のサイズを返すカーネル構成マクロ（TSZ\_MPF）は廃止した。これは、固定長メモリプール領域を確保する方法を定めたことに加えて、サイズが必要な場合には、 $((blkcnt) * ROUND\_MPF\_T(blksz))$ で求めることができるためである。

TNUM\_MPFIDは、μITRON4.0仕様に規定されていないカーネル構成マクロである。

-----  
 CRE\_MPF 固定長メモリプールの生成〔S〕  
 acre\_mpf 固定長メモリプールの生成〔TD〕

#### 【静的API】

CRE\_MPF(ID mpfid, { ATR mpfatr, uint\_t blkcnt, uint\_t blksz,  
 MPF\_T \*mpf, void \*mpfmb })

#### 【C言語API】

ER\_ID mpfid = acre\_mpf(const T\_CMPF \*pk\_cmpf)

#### 【パラメータ】

ID	mpfid	生成する固定長メモリプールのID番号（acre_mpfを除く）
T_CMPF *	pk_cmpf	固定長メモリプールの生成情報を入れたパケットへのポインタ（静的APIを除く）

#### \* 固定長メモリプールの生成情報（パケットの内容）

ATR	mpfatr	固定長メモリプール属性
uint_t	blkcnt	獲得できる固定長メモリブロックの数
uint_t	blksz	固定長メモリブロックのサイズ（バイト数）
MPF_T *	mpf	固定長メモリプール領域の先頭番地
void *	mpfmb	固定長メモリプール管理領域の先頭番地

#### 【リターンパラメータ】

ER_ID	mpfid	生成された固定長メモリプールのID番号（正の
-------	-------	------------------------

値)またはエラーコード

### 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
E_RSATR	予約属性 (mpfatrが不正または使用できない)
E_NOSPT	未サポート機能 (mpfmbがサポートされていない値)
E_PAR	パラメータエラー (blkcnt, blkksz, mpf, mpfmbが不正)
E_OACV [P]	オブジェクトアクセス違反 (システム状態に対する管理操作が許可されていない)
E_MACV [P]	メモリアクセス違反 (pk_cmpfが指すメモリ領域への読出しアクセスが許可されていない)
E_NOID	ID番号不足 (割り付けられる固定長メモリプールIDがない: acre_mpfの場合)
E_OBJ	オブジェクト状態エラー (mpfidで指定した固定長メモリプールが登録済み: acre_mpfを除く)

### 【機能】

各パラメータで指定した固定長メモリプール生成情報に従って, 固定長メモリプールを生成する. mpf, blkcnt, blkkszから固定長メモリプール領域が, mpfmb, blkcnt, blkkszから固定長メモリプール管理領域がそれぞれ設定され, メモリプール領域全体が未割当ての状態に初期化される. また, 待ち行列は空の状態に初期化される.

静的APIにおいては, mpfidはオブジェクト識別名, blkcntとblkkszは整数定数式パラメータ, mpfとmpfmbは一般定数式パラメータである.

mpfをNULLとした場合, blkcntとblkkszから決まるサイズの固定長メモリプール領域を, コンフィギュレータまたはカーネルが確保する.

mpfがNULLでない場合, blkcntとblkkszから決まるサイズの固定長メモリプール領域を, アプリケーションで確保する. 固定長メモリプール領域をアプリケーションで確保するために, 次のデータ型とマクロを用意している.

MPF_T	固定長メモリプール領域を確保するためのデータ型
COUNT_MPF_T(blkksz)	固定長メモリブロックのサイズがblkkszの固定長メモリプール領域を確保するために, 固定長メモリブロック1つあたりに必要なMPF_T型の配列の要素数を求めるマクロ
ROUND_MPF_T(blkksz)	要素数COUNT_MPF_T(blkksz)のMPF_T型の配列のサイズ (blkkszを, MPF_T型のサイズの倍数になるように大きい方に丸めた値)

これらを用いて固定長メモリプール領域を確保する方法は次の通り.

```
MPF_T <固定長メモリプール領域の変数名>[(blkcnt) * COUNT_MPF_T(blkksz)];
```

この時, mpfには<固定長メモリプール領域の変数名>を指定する.

これ以外の方法で固定長メモリプール領域を確保する場合には, 先頭番地がターゲットシステムの制約に合致しており, 上記の配列と同じサイズのメモリ領域を確保しなければならない. mpfにターゲットシステムの制約に合致しない先頭番地を指定した時には, E\_PARエラーとなる.

mpfmbをNULLとした場合, blkcntとblkkszの指定に合致したサイズのデータキュー管理領域を, コンフィギュレータまたはカーネルが確保する.



blkcntとblkkszは、0より大きい値でなければならない。

#### 【未決定事項】

mpfmbがNULLでない場合に、固定長メモリプール領域をアプリケーションで確保する方法については、今後の課題である。

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、CRE\_MPFのみをサポートする。また、mpfmbにはNULLのみを渡すことができる。NULL以外を指定した場合には、E\_NOSPTエラーとなる。

#### 【μITRON4.0仕様との関係】

μITRON4.0/PX仕様にあわせて、固定長メモリプール生成情報に、mpfmb（固定長メモリプール管理領域の先頭番地）を追加した。また、mpfのデータ型をMPF\_T \*に変更した。

-----  
 SAC\_MPF 固定長メモリプールのアクセス許可ベクタの設定〔SP〕  
 sac\_mpf 固定長メモリプールのアクセス許可ベクタの設定〔TPD〕

#### 【静的API】

```
SAC_MPF(ID mpfid, { ACPTN acptn1, ACPTN acptn2,
                  ACPTN acptn3, ACPTN acptn4 })
```

#### 【C言語API】

```
ER ercd = sac_mpf(ID mpfid, const ACVCT *p_acvct)
```

#### 【パラメータ】

ID	mpfid	対象固定長メモリプールのID番号
ACVCT *	p_acvct	アクセス許可ベクタを入れたパケットへのポインタ（静的APIを除く）

#### \* アクセス許可ベクタ（パケットの内容）

ACPTN	acptn1	通常操作1のアクセス許可パターン
ACPTN	acptn2	通常操作2のアクセス許可パターン
ACPTN	acptn3	管理操作のアクセス許可パターン
ACPTN	acptn4	参照操作のアクセス許可パターン

#### 【リターンパラメータ】

ER	ercd	正常終了（E_OK）またはエラーコード
----	------	---------------------

#### 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し、CPUロック状態からの呼出し）
E_ID	不正ID番号（mpfidが不正）
E_NOEXS〔D〕	オブジェクト未登録（対象固定長メモリプールが未登録）
E_OACV〔P〕	オブジェクトアクセス違反（対象固定長メモリプールに対する管理操作が許可されていない）
E_MACV〔P〕	メモリアクセス違反（p_acvctが指すメモリ領域への読出しアクセスが許可されていない）
E_OBJ	オブジェクト状態エラー（対象固定長メモリプールは静的APIで生成された：静的APIを除く）

#### 【機能】

mpfidで指定した固定長メモリプール（対象固定長メモリプール）のアクセス許

可ベクタ（4つのアクセス許可パターンの組）を、各パラメータで指定した値に設定する。

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、SAC\_MPF, sac\_mpfをサポートしない。

-----  
del\_mpf 固定長メモリプールの削除〔TD〕

【C言語API】

```
ER ercd = del_mpf(ID mpfid)
```

【パラメータ】

ID mpfid 対象固定長メモリプールのID番号

【リターンパラメータ】

ER ercd 正常終了（E\_OK）またはエラーコード

【エラーコード】

E\_CTX コンテキストエラー（非タスクコンテキストからの呼出し、CPUロック状態からの呼出し）  
 E\_ID 不正ID番号（mpfidが不正）  
 E\_NOEXS オブジェクト未登録（対象固定長メモリプールが未登録）  
 E\_OACV〔P〕 オブジェクトアクセス違反（対象固定長メモリプールに対する管理操作が許可されていない）  
 E\_OBJ オブジェクト状態エラー（対象固定長メモリプールは静的APIで生成された）

【機能】

mpfidで指定した固定長メモリプール（対象固定長メモリプール）を削除する。具体的な振舞いは以下の通り。

対象固定長メモリプールの登録が解除され、その固定長メモリプールIDが未使用の状態に戻される。また、対象固定長メモリプールの待ち行列につながれたタスクは、待ち行列の先頭のタスクから順に待ち解除される。待ち解除されたタスクには、待ち状態となったサービスコールからE\_DLTエラーが返る。

【使用上の注意】

del\_mpfにより複数のタスクが待ち解除される場合、サービスコールの処理時間およびカーネル内での割り込み禁止時間が、待ち解除されるタスクの数に比例して長くなる。特に、多くのタスクが待ち解除される場合、カーネル内での割り込み禁止時間が長くなるため、注意が必要である。

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、del\_mpfをサポートしない。

-----  
get\_mpf 固定長メモリブロックの獲得〔T〕  
 pget\_mpf 固定長メモリブロックの獲得（ポーリング）〔T〕  
 tget\_mpf 固定長メモリブロックの獲得（タイムアウト付き）〔T〕

【C言語API】

```
ER ercd = get_mpf(ID mpfid, void **p_blk)
ER ercd = pget_mpf(ID mpfid, void **p_blk)
ER ercd = tget_mpf(ID mpfid, void **p_blk, TMO tmout)
```

## 【パラメータ】

ID	mpfid	対象固定長メモリプールのID番号
void **	p_blk	獲得した固定長メモリブロックの先頭番地を入れるメモリ領域へのポインタ
TMO	tmout	タイムアウト時間 (twai_mpfの場合)

## 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
void *	blk	獲得した固定長メモリブロックの先頭番地

## 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し, ディスパッチ保留状態からの呼出し: pget_mpfを除く)
E_ID	不正ID番号 (mpfidが不正)
E_PAR	パラメータエラー (tmoutが不正: tget_mpfの場合)
E_NOEXS [D]	オブジェクト未登録 (対象固定長メモリプールが未登録)
E_OACV [P]	オブジェクトアクセス違反 (対象固定長メモリプールに対する通常操作2が許可されていない)
E_MACV [P]	メモリアクセス違反 (p_blkが指すメモリ領域への読出しアクセスが許可されていない)
E_TMOUT	ポーリング失敗またはタイムアウト (get_mpfを除く)
E_RLWAI	待ち禁止状態または待ち状態の強制解除 (pget_mpfを除く)
E_DLT	待ちオブジェクトの削除または再初期化 (pget_mpfを除く)

## 【機能】

mpfidで指定した固定長メモリプール (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し, その先頭番地をblkに返す. 具体的な振舞いは以下の通り.

対象固定長メモリプールの固定長メモリプール領域の中に, 固定長メモリブロックを割り付けることのできる未割当てのメモリ領域がある場合には, 固定長メモリブロックが1つ割り付けられ, その先頭番地がblkに返される.

未割当てのメモリ領域がない場合には, 自タスクは固定長メモリプールの獲得待ち状態となり, 対象固定長メモリプールの待ち行列につながる.

-----  
rel\_mpf 固定長メモリブロックの返却 [T]

## 【C言語API】

```
ER ercd = rel_mpf(ID mpfid, void *blk)
```

## 【パラメータ】

ID	mpfid	対象固定長メモリプールのID番号
void *	blk	返却する固定長メモリブロックの先頭番地

## 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

## 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
E_ID	不正ID番号 (mpfidが不正)
E_PAR	パラメータエラー (blkが不正)
E_NOEXS [D]	オブジェクト未登録 (対象固定長メモリプールが未登録)

E\_OACV [P] オブジェクトアクセス違反 (対象固定長メモリプールに対する通常操作1が許可されていない)

#### 【機能】

mpfidで指定した固定長メモリプール (対象固定長メモリプール) に、blkで指定した固定長メモリブロックを返却する。具体的な振舞いは以下の通り。

対象固定長メモリプールの待ち行列にタスクが存在する場合には、待ち行列の先頭のタスクが、blkで指定した固定長メモリブロックを獲得し、待ち解除される。待ち解除されたタスクには、待ち状態となったサービスコールからE\_OKが返る。

待ち行列にタスクが存在しない場合には、blkで指定した固定長メモリブロックは、対象固定長メモリプールのメモリプール領域に返却される。

blkが、対象固定長メモリプールから獲得した固定長メモリブロックの先頭番地でない場合には、E\_PARエラーとなる。

-----

ini\_mpf 固定長メモリプールの再初期化 [T]

#### 【C言語API】

```
ER ercd = ini_mpf(ID mpfid)
```

#### 【パラメータ】

ID mpfid 対象固定長メモリプールのID番号

#### 【リターンパラメータ】

ER ercd 正常終了 (E\_OK) またはエラーコード

#### 【エラーコード】

E\_CTX コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)  
 E\_ID 不正ID番号 (mpfidが不正)  
 E\_NOEXS [D] オブジェクト未登録 (対象固定長メモリプールが未登録)  
 E\_OACV [P] オブジェクトアクセス違反 (対象固定長メモリプールに対する管理操作が許可されていない)

#### 【機能】

mpfidで指定した固定長メモリプール (対象固定長メモリプール) を再初期化する。具体的な振舞いは以下の通り。

対象固定長メモリプールのメモリプール領域全体が未割当ての状態に初期化される。また、対象固定長メモリプールの待ち行列につながれたタスクは、待ち行列の先頭のタスクから順に待ち解除される。待ち解除されたタスクには、待ち状態となったサービスコールからE\_DLTエラーが返る。

#### 【使用上の注意】

ini\_mpfにより複数のタスクが待ち解除される場合、サービスコールの処理時間およびカーネル内での割込み禁止時間が、待ち解除されるタスクの数に比例して長くなる。特に、多くのタスクが待ち解除される場合、カーネル内での割込み禁止時間が長くなるため、注意が必要である。

固定長メモリプールを再初期化した場合に、アプリケーションとの整合性を保つのは、アプリケーションの責任である。

## 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである。

-----  
ref\_mpf 固定長メモリプールの状態参照〔T〕

## 【C言語API】

```
ER ercd = ref_mpf(ID mpfid, T_RMPF *pk_rmpf)
```

## 【パラメータ】

ID	mpfid	対象固定長メモリプールのID番号
T_RMPF *	pk_rmpf	固定長メモリプールの現在状態を入れるパケットへのポインタ

## 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

\* 固定長メモリプールの現在状態 (パケットの内容)

ID	wtskid	固定長メモリプールの待ち行列の先頭のタスクのID番号
uint_t	fblkcnt	固定長メモリプール領域の空きメモリ領域に割り付けることができる固定長メモリブロックの数

## 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
E_ID	不正ID番号 (mpfidが不正)
E_NOEXS〔D〕	オブジェクト未登録 (対象固定長メモリプールが未登録)
E_OACV〔P〕	オブジェクトアクセス違反 (対象固定長メモリプールに対する参照操作が許可されていない)
E_MACV〔P〕	メモリアクセス違反 (pk_rmpfが指すメモリ領域への書込みアクセスが許可されていない)

## 【機能】

mpfidで指定した固定長メモリプール (対象固定長メモリプール) の現在状態を参照する。参照した現在状態は, pk\_rmpfで指定したパケットに返される。

対象固定長メモリプールの待ち行列にタスクが存在しない場合, wtskidには TSK\_NONE (= 0) が返る。

## 【使用上の注意】

ref\_mpfはデバッグ時向けの機能であり, その他の目的に使用することは推奨しない。これは, ref\_mpfを呼び出し, 対象固定長メモリプールの現在状態を参照した直後に割込みが発生した場合, ref\_mpfから戻ってきた時には対象固定長メモリプールの状態が変化している可能性があるためである。

-----

## 4.6 時間管理機能

## 4.6.1 システム時刻管理

システム時刻は, カーネルによって管理され, タイムアウト処理, タスクの遅延, 周期ハンドラの起動, アラームハンドラの起動に使用される時刻を管理するカーネルオブジェクトである。システム時刻は, 符号無しの整数型である SYSTIM型で表され, 単位はミリ秒である。ただし, 実際の精度はターゲット依

存である。

システム時刻は、カーネル内に1つのみ存在し、カーネルの起動時に0に初期化される。タイムティックの発生毎にカーネルによって更新され、SYSTIM型で表現できる最大値 (ULONG\_MAX) を超えると0に戻される。タイムティックの周期は、ターゲット定義である。

マルチプロセッサ対応カーネルでは、プロセッサ毎にシステム時刻が複数存在する場合がある。

1回のタイムティックが発生により、複数の処理を行うべき状況になった場合、それらの処理の間の処理順序は規定されない。

性能評価用システム時刻は、性能評価に使用することを目的とした、システム時刻よりも精度の高い時刻である。性能評価用システム時刻は、符号無しの整数型であるSYSUTM型で表され、単位はマイクロ秒である。ただし、実際の精度はターゲット依存である。

システム時刻管理機能に関連するカーネル構成マクロは次の通り。

TIC_NUME	タイムティックの周期 (単位はミリ秒) の分子
TIC_DENO	タイムティックの周期 (単位はミリ秒) の分母
TOPPERS_SUPPORT_GET_UTM	get_utmがサポートされている

#### 【μITRON4.0仕様との関係】

システム時刻を設定するサービスコール (set\_tim) を廃止した。また、タイムティックを供給する機能は、カーネル内に実現することとし、そのためのサービスコール (isig\_tim) は廃止した。

-----  
get\_tim      システム時刻の参照 [T]

#### 【C言語API】

```
ER ercd = get_tim(SYSTIM *p_system)
```

#### 【パラメータ】

SYSTIM *	p_system	システム時刻を入れるメモリ領域へのポインタ
----------	----------	-----------------------

#### 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
SYSTIM	system	システム時刻の現在値

#### 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
E_OACV [P]	オブジェクトアクセス違反 (システム状態に対する通常操作1が許可されていない)
E_MACV [P]	メモリアクセス違反 (p_systemが指すメモリ領域への書き込みアクセスが許可されていない)

#### 【機能】

システム時刻の現在値を参照する。参照したシステム時刻は、p\_systemで指定したメモリ領域に返される。

-----  
get\_utm      性能評価用システム時刻の参照 [TI]

## 【C言語API】

```
ER ercd = get_utm(SYSUTM *p_sysutm)
```

## 【パラメータ】

SYSUTM *	p_sysutm	性能評価用システム時刻を入れるメモリ領域へのポインタ
----------	----------	----------------------------

## 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
SYSUTM	sysutm	性能評価用システム時刻の現在値

## 【エラーコード】

E_NOSPT	未サポート機能 (get_utmがサポートされていない)
E_OACV [P]	オブジェクトアクセス違反 (システム状態に対する通常操作1が許可されていない)
E_MACV [P]	メモリアクセス違反 (p_sysutmが指すメモリ領域へ書き込みアクセスが許可されていない)

## 【機能】

性能評価用システム時刻の現在値を参照する。参照した性能評価用システム時刻は、p\_sysutmで指定したメモリ領域に返される。

get\_utmは、任意の状態から呼び出すことができる。タスクコンテキストからも非タスクコンテキストからも呼び出すことができるし、CPUロック状態であっても呼び出すことができる。

ターゲット定義で、get\_utmがサポートされていない場合がある。get\_utmがサポートされている場合には、TOPPERS\_SUPPORT\_GET\_UTMがマクロ定義される。サポートされていない場合にget\_utmを呼び出すと、E\_NOSPTエラーが返るか、リンク時にエラーとなる。

## 【使用方法】

get\_utmを使用してプログラムの実行時間を計測する場合には、次の手順を取る。実行時間を計測したいプログラムの実行直前と実行直後に、このサービスコールを用いて性能評価用システム時刻を読み出す。その差を求めることで、対象プログラムの実行時間に、get\_utm自身の実行時間を加えたものが得られる。

## 【使用上の注意】

get\_utmは性能評価のための機能であり、その他の目的に使用することは推奨しない。

タイマ割込みが長時間禁止されている、タイマ割込みよりも優先度の高い割込み処理が長時間続けて実行された、シミュレーション環境においてシミュレータのプロセスが長時間スケジューラされなかったなどの理由で、システム時刻の更新が正しく行えない状況では、get\_utmは誤った性能評価用システム時刻を返す可能性がある。システム時刻の更新が確実に実行されることを保証できない場合には、get\_utmが誤った性能評価用システム時刻を返す可能性を考慮に入れて使用しなければならない。

## 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである。

## 4.6.2 周期ハンドラ

周期ハンドラは、指定した周期で起動されるタイムイベントハンドラである。  
周期ハンドラは、周期ハンドラIDと呼ぶID番号によって識別する。

各周期ハンドラが持つ情報は次の通り。

- ・周期ハンドラ属性
- ・周期ハンドラの動作状態
- ・次に周期ハンドラを起動するシステム時刻
- ・拡張情報
- ・周期ハンドラ先頭番地
- ・起動周期
- ・起動位相

周期ハンドラの起動時刻は、後述する基準時刻から、以下の式で求められる相対時間後である。

$$\text{起動位相} + \text{起動周期} \times (n - 1) \quad n = 1, 2, \dots$$

周期ハンドラの動作状態は、動作している状態と動作していない状態のいずれかをとる。周期ハンドラを動作している状態にすることを動作開始、動作していない状態にすることを動作停止という。

周期ハンドラが動作している状態の場合には、周期ハンドラの起動時刻になると、周期ハンドラの起動処理が行われる。具体的には、拡張情報をパラメータとして、周期ハンドラ先頭番地が呼び出される。

周期ハンドラ属性には、次の属性を指定することができる。

TA_STA	0x01U	周期ハンドラの生成時に周期ハンドラを動作開始する
TA_PHS	0x02U	周期ハンドラを生成した時刻を基準時刻とする

TA\_STAを指定しない場合、周期ハンドラの生成直後には、周期ハンドラは動作していない状態となる。

TA\_PHSを指定しない場合には、周期ハンドラを動作開始した時刻が、周期ハンドラを起動する時刻の基準時刻となる。TA\_PHSを指定した場合には、周期ハンドラを生成した時刻（静的APIで生成した場合にはカーネルの起動時刻、すなわちシステム時刻が0）が、基準時刻となる。

次に周期ハンドラを起動するシステム時刻は、周期ハンドラが動作している状態でのみ有効で、必要に応じて、カーネルの起動時、周期ハンドラの動作開始時、周期ハンドラの起動処理時に設定される。

C言語による周期ハンドラの記述形式は次の通り。

```
void cyclic_handler(intptr_t exinf)
{
    周期ハンドラ本体
}
```

exinfには、周期ハンドラの拡張情報が渡される。

周期ハンドラ機能に関連するカーネル構成マクロは次の通り。

TNUM_CYCID	登録できる周期ハンドラの数（動的生成対応でないカーネルでは、静的APIによって登録された周期ハンドラの数に一致）
------------	--



## 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、TA\_PHS属性の周期ハンドラをサポートしない。

## 【μITRON4.0仕様との関係】

TNUM\_CYCIDは、μITRON4.0仕様に規定されていないカーネル構成マクロである。

-----  
 CRE\_CYC 周期ハンドラの生成〔S〕  
 acre\_cyc 周期ハンドラの生成〔TD〕

## 【静的API】

CRE\_CYC(ID cycid, { ATR cycatr, intptr\_t exinf, CYCHDR cychdr,  
 RELTIM cyctim, RELTIM cycphs })

## 【C言語API】

ER\_ID cycid = acre\_cyc(const T\_CCYC \*pk\_ccyc)

## 【パラメータ】

ID	cycid	生成する周期ハンドラのID番号 (acre_cycを除く)
T_CCYC *	pk_ccyc	周期ハンドラの生成情報を入れたパケットへのポインタ (静的APIを除く)

## \* 周期ハンドラの生成情報 (パケットの内容)

ATR	cycatr	周期ハンドラ属性
intptr_t	exinf	周期ハンドラの拡張情報
CYCHDR	cyhdr	周期ハンドラの先頭番地
RELTIM	cyctim	周期ハンドラの起動周期
RELTIM	cycphs	周期ハンドラの起動位相

## 【リターンパラメータ】

ER_ID	cycid	生成された周期ハンドラのID番号 (正の値) またはエラーコード
-------	-------	----------------------------------

## 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
E_RSATR	予約属性 (cycatrが不正または使用できない)
E_PAR	パラメータエラー (cyhdr, cyctim, cycphsが不正)
E_OACV〔P〕	オブジェクトアクセス違反 (システム状態に対する管理操作が許可されていない)
E_MACV〔P〕	メモリアクセス違反 (pk_ccycが指すメモリ領域への読出しアクセスが許可されていない)
E_NOID	ID番号不足 (割り付けられる周期ハンドラIDがない: acre_cycの場合)
E_OBJ	オブジェクト状態エラー (cycidで指定した周期ハンドラが登録済み: acre_cycを除く)

## 【機能】

各パラメータで指定した周期ハンドラ生成情報に従って、周期ハンドラを生成する。具体的な振舞いは以下の通り。

cycatrにTA\_STAを指定した場合、対象周期ハンドラは動作している状態となる。次に周期ハンドラを起動するシステム時刻は、静的APIの場合はcycphsで指定したシステム時刻に、サービスコールの場合は呼び出してからcycphsで指定した相対時間後に設定される。

cycatrにTA\_STAを指定しない場合，対象周期ハンドラは動作していない状態に初期化される．

静的APIにおいては，cycidはオブジェクト識別名，cycatr，cyctim，cycphsは整数定数式パラメータ，exinfとcychdrは一般定数式パラメータである．

cyctimは，0より大きく，TMAX\_RELTIM以下の値でなければならない．また，cycphsは，TMAX\_RELTIM以下でなければならない．cycphsにcyctimより大きい値を指定してもよい．

#### 【補足説明】

静的APIにおいて，cycatrにTA\_STAを，cycphsに0を指定した場合，周期ハンドラが最初に呼び出されるのは，カーネル起動後最初のタイムティックになる．cycphsに1を指定した場合も同じ振舞いとなるため，静的APIでcycphsに0を指定することは推奨しないこととし，コンフィギュレータが警告メッセージを出力する．

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは，GRE\_CYCのみをサポートする．ただし，TA\_PHS属性の周期ハンドラはサポートしない．

#### 【μITRON4.0仕様との関係】

cychdrのデータ型をCYCHDRに変更した．また，cycphsにcyctimより大きい値を指定した場合の振舞いと，静的APIでcycphsに0を指定した場合の振舞いを規定した．

-----  
SAC\_CYC 周期ハンドラのアクセス許可ベクタの設定〔SP〕  
sac\_cyc 周期ハンドラのアクセス許可ベクタの設定〔TPD〕

#### 【静的API】

```
SAC_CYC(ID cycid, { ACPTN acptn1, ACPTN acptn2,
                    ACPTN acptn3, ACPTN acptn4 })
```

#### 【C言語API】

```
ER ercd = sac_cyc(ID cycid, const ACVCT *p_acvct)
```

#### 【パラメータ】

ID	cycid	対象周期ハンドラのID番号
ACVCT *	p_acvct	アクセス許可ベクタを入れたパケットへのポインタ（静的APIを除く）

#### \* アクセス許可ベクタ（パケットの内容）

ACPTN	acptn1	通常操作1のアクセス許可パターン
ACPTN	acptn2	通常操作2のアクセス許可パターン
ACPTN	acptn3	管理操作のアクセス許可パターン
ACPTN	acptn4	参照操作のアクセス許可パターン

#### 【リターンパラメータ】

ER	ercd	正常終了（E_OK）またはエラーコード
----	------	---------------------

#### 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し，CPUロック状態からの呼出し）
E_ID	不正ID番号（cycidが不正）

E_NOEXS [D]	オブジェクト未登録 (対象周期ハンドラが未登録)
E_OACV [P]	オブジェクトアクセス違反 (対象周期ハンドラに対する管理操作が許可されていない)
E_MACV [P]	メモリアクセス違反 (p_acvctが指すメモリ領域への読みアクセスが許可されていない)
E_OBJ	オブジェクト状態エラー (対象周期ハンドラは静的APIで生成された: 静的APIを除く)

### 【機能】

cycidで指定した周期ハンドラ (対象周期ハンドラ) のアクセス許可ベクタ (4つのアクセス許可パターンの組) を, 各パラメータで指定した値に設定する.

### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは, SAC\_CYC, sac\_cycをサポートしない.

-----  
del\_cyc 周期ハンドラの削除 [TD]

### 【C言語API】

```
ER ercd = del_cyc(ID cycid)
```

### 【パラメータ】

ID	cycid	対象周期ハンドラのID番号
----	-------	---------------

### 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

### 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
E_ID	不正ID番号 (cycidが不正)
E_NOEXS	オブジェクト未登録 (対象周期ハンドラが未登録)
E_OACV [P]	オブジェクトアクセス違反 (対象周期ハンドラに対する管理操作が許可されていない)
E_OBJ	オブジェクト状態エラー (対象周期ハンドラは静的APIで生成された)

### 【機能】

cycidで指定した周期ハンドラ (対象周期ハンドラ) を削除する. 具体的な振舞いは以下の通り.

対象周期ハンドラの登録が解除され, その周期ハンドラIDが未使用の状態に戻される. 対象周期ハンドラが動作している状態であった場合には, 動作していない状態にされた後に, 登録が解除される.

### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは, del\_cycをサポートしない.

-----  
sta\_cyc 周期ハンドラの動作開始 [T]

### 【C言語API】

```
ER ercd = sta_cyc(ID cycid)
```

### 【パラメータ】

ID	cycid	対象周期ハンドラのID番号
----	-------	---------------

## 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

## 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
E_ID	不正ID番号 (cycidが不正)
E_NOEXS [D]	オブジェクト未登録 (対象周期ハンドラが未登録)
E_OACV [P]	オブジェクトアクセス違反 (対象周期ハンドラに対する通常操作1が許可されていない)

## 【機能】

cycidで指定した周期ハンドラ (対象周期ハンドラ) を動作開始する。具体的な振舞いは以下の通り。

対象周期ハンドラが動作していない状態であれば、対象周期ハンドラは動作している状態となり、次に周期ハンドラを起動するシステム時刻は、sta\_cycを呼び出して以降の最初の起動時刻に設定される。TA\_PHS属性でない周期ハンドラの場合には、sta\_cycを呼び出してから、対象周期ハンドラの起動位相で指定した相対時間後に設定されることになる。

対象周期ハンドラが動作している状態であれば、次に周期ハンドラを起動するシステム時刻の再設定のみが行われる (TA\_PHS属性の周期ハンドラの場合には、次に周期ハンドラを起動するシステム時刻は変化しない)。

## 【μITRON4.0仕様との関係】

TA\_PHS属性でない周期ハンドラにおいて、sta\_cycを呼び出した後、最初に周期ハンドラが起動される時刻を変更した。μITRON4.0仕様では、sta\_cycを呼び出してから周期ハンドラの起動周期で指定した相対時間後となっているが、この仕様では、起動位相で指定した相対時間後とした。

-----  
stp\_cyc 周期ハンドラの動作停止 [T]

## 【C言語API】

```
ER ercd = stp_cyc(ID cycid)
```

## 【パラメータ】

ID	cycid	対象周期ハンドラのID番号
----	-------	---------------

## 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

## 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
E_ID	不正ID番号 (cycidが不正)
E_NOEXS [D]	オブジェクト未登録 (対象周期ハンドラが未登録)
E_OACV [P]	オブジェクトアクセス違反 (対象周期ハンドラに対する通常操作2が許可されていない)

## 【機能】

cycidで指定した周期ハンドラ (対象周期ハンドラ) を動作停止する。具体的な振舞いは以下の通り。

対象周期ハンドラが動作している状態であれば、動作していない状態になる。  
対象周期ハンドラが動作していない状態であれば、何も行われない。

-----  
ref\_cyc 周期ハンドラの状態参照 [T]

**【C言語API】**

```
ER ercd = ref_cyc(ID cycid, T_RCYC *pk_rcyc)
```

**【パラメータ】**

ID	cycid	対象周期ハンドラのID番号
T_RCYC *	pk_rcyc	周期ハンドラの現在状態を入れるパケットへのポインタ

**【リターンパラメータ】**

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

\* 周期ハンドラの現在状態 (パケットの内容)

STAT	cycstat	周期ハンドラの動作状態
RELTIM	lefttim	次に周期ハンドラを起動する時刻までの相対時間

**【エラーコード】**

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
E_ID	不正ID番号 (cycidが不正)
E_NOEXS [D]	オブジェクト未登録 (対象周期ハンドラが未登録)
E_OACV [P]	オブジェクトアクセス違反 (対象周期ハンドラに対する参照操作が許可されていない)
E_MACV [P]	メモリアクセス違反 (pk_rcycが指すメモリ領域への書込みアクセスが許可されていない)

**【機能】**

cycidで指定した周期ハンドラ (対象周期ハンドラ) の現在状態を参照する。参照した現在状態は、pk\_rcycで指定したパケットに返される。

cycstatには、対象周期ハンドラの現在の動作状態を表す次のいずれかの値が返される。

TCYC_STP	0x01U	動作していない状態
TCYC_STA	0x02U	動作している状態

対象周期ハンドラが動作している状態である場合には、lefttimに、次に周期ハンドラ起動する時刻までの相対時間が返される。対象周期ハンドラが動作していない状態である場合には、lefttimの値は保証されない。

**【使用上の注意】**

ref\_cycはデバッグ時向けの機能であり、その他の目的に使用することは推奨しない。これは、ref\_cycを呼び出し、対象周期ハンドラの現在状態を参照した後、割込みが発生した場合、ref\_cycから戻ってきた時には対象周期ハンドラの状態が変化している可能性があるためである。

**【μITRON4.0仕様との関係】**

TCYC\_STPとTCYC\_STAを値を変更した。  
-----

#### 4.6.3 アラームハンドラ

アラームハンドラは、指定した相対時間後に起動されるタイムイベントハンドラである。アラームハンドラは、アラームハンドラIDと呼ぶID番号によって識別する。

各アラームハンドラが持つ情報は次の通り。

- ・アラームハンドラ属性
- ・アラームハンドラの動作状態
- ・アラームハンドラを起動するシステム時刻
- ・拡張情報
- ・アラームハンドラ先頭番地

アラームハンドラの動作状態は、動作している状態と動作していない状態のいずれかをとる。アラームハンドラを動作している状態にすることを動作開始、動作していない状態にすることを動作停止という。

アラームハンドラを起動するシステム時刻は、アラームハンドラを動作開始する時に設定され、動作停止する時に設定解除される。

アラームハンドラが動作している状態の場合には、アラームハンドラを起動するシステム時刻になると、アラームハンドラの起動処理が行われる。具体的には、まず、アラームハンドラが動作していない状態にされる。その後に、拡張情報をパラメータとして、アラームハンドラ先頭番地が呼び出される。

C言語によるアラームハンドラの記述形式は次の通り。

```
void alarm_handler(intptr_t exinf)
{
    アラームハンドラ本体
}
```

exinfには、アラームハンドラの拡張情報が渡される。

アラームハンドラ機能に関連するカーネル構成マクロは次の通り。

```
TNUM_ALMID    登録できるアラームハンドラの数（動的生成対応でない
               カーネルでは、静的APIによって登録されたアラームハン
               ドラの数に一致）
```

#### 【μITRON4.0仕様との関係】

TNUM\_ALMIDは、μITRON4.0仕様に規定されていないカーネル構成マクロである。

```
-----
CRE_ALM      アラームハンドラの生成〔S〕
acre_alm     アラームハンドラの生成〔TD〕
```

#### 【静的API】

```
CRE_ALM(ID almid, { ATR almatr, intptr_t exinf, ALMHDR alhdr })
```

#### 【C言語API】

```
ER_ID almid = acre_alm(const T_CALM *pk_calm)
```

#### 【パラメータ】

```
ID          almid      生成するアラームハンドラのID番号（acre_alm
                       を除く）
T_CALM *    pk_calm    アラームハンドラの生成情報を入れたパケット
                       へのポインタ（静的APIを除く）
```

\* アラームハンドラの生成情報 (パケットの内容)

ATR	almatr	アラームハンドラ属性
intptr_t	exinf	アラームハンドラの拡張情報
ALMHDR	almhdr	アラームハンドラ先頭番地

#### 【リターンパラメータ】

ER_ID	almid	生成されたアラームハンドラのID番号 (正の値) またはエラーコード
-------	-------	---------------------------------------

#### 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
E_RSATR	予約属性 (almatrが不正または使用できない)
E_PAR	パラメータエラー (almhdrが不正)
E_OACV [P]	オブジェクトアクセス違反 (システム状態に対する管理操作が許可されていない)
E_MACV [P]	メモリアクセス違反 (pk_calmが指すメモリ領域への読出しアクセスが許可されていない)
E_NOID	ID番号不足 (割り付けられるアラームハンドラIDがない : acre_almの場合)
E_OBJ	オブジェクト状態エラー (almidで指定したアラームハンドラが登録済み : acre_almを除く)

#### 【機能】

各パラメータで指定したアラームハンドラ生成情報に従って, アラームハンドラを生成する. 対象アラームハンドラは, 動作していない状態に初期化される.

静的APIにおいては, almidはオブジェクト識別名, almatrは整数定数式パラメータ, exinfとalmhdrは一般定数式パラメータである.

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは, CRE\_ALMのみをサポートする.

#### 【μITRON4.0仕様との関係】

almhdrのデータ型をALMHDRに変更した.

---

SAC_ALM	アラームハンドラのアクセス許可ベクタの設定 [SP]
sac_alm	アラームハンドラのアクセス許可ベクタの設定 [TPD]

#### 【静的API】

```
SAC_ALM(ID almid, { ACPTN acptn1, ACPTN acptn2,
                    ACPTN acptn3, ACPTN acptn4 })
```

#### 【C言語API】

```
ER ercd = sac_alm(ID almid, const ACVCT *p_acvct)
```

#### 【パラメータ】

ID	almid	対象アラームハンドラのID番号
ACVCT *	p_acvct	アクセス許可ベクタを入れたパケットへのポインタ (静的APIを除く)

\* アクセス許可ベクタ (パケットの内容)

ACPTN	acptn1	通常操作1のアクセス許可パターン
ACPTN	acptn2	通常操作2のアクセス許可パターン

ACPTN	acptn3	管理操作のアクセス許可パターン
ACPTN	acptn4	参照操作のアクセス許可パターン

## 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

## 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
E_ID	不正ID番号 (almidが不正)
E_NOEXS [D]	オブジェクト未登録 (対象アラームハンドラが未登録)
E_OACV [P]	オブジェクトアクセス違反 (対象アラームハンドラに対する管理操作が許可されていない)
E_MACV [P]	メモリアクセス違反 (p_acvctが指すメモリ領域への読出しアクセスが許可されていない)
E_OBJ	オブジェクト状態エラー (対象アラームハンドラは静的APIで生成された: 静的APIを除く)

## 【機能】

almidで指定したアラームハンドラ (対象アラームハンドラ) のアクセス許可ベクタ (4つのアクセス許可パターンの組) を, 各パラメータで指定した値に設定する.

## 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは, SAC\_ALM, sac\_almをサポートしない.

-----  
del\_alm アラームハンドラの削除 [TD]

## 【C言語API】

```
ER ercd = del_alm(ID almid)
```

## 【パラメータ】

ID	almid	対象アラームハンドラのID番号
----	-------	-----------------

## 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

## 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
E_ID	不正ID番号 (almidが不正)
E_NOEXS	オブジェクト未登録 (対象アラームハンドラが未登録)
E_OACV [P]	オブジェクトアクセス違反 (対象アラームハンドラに対する管理操作が許可されていない)
E_OBJ	オブジェクト状態エラー (対象アラームハンドラは静的APIで生成された)

## 【機能】

almidで指定したアラームハンドラ (対象アラームハンドラ) を削除する. 具体的な振舞いは以下の通り.

対象アラームハンドラの登録が解除され, そのアラームハンドラIDが未使用の状態に戻される. 対象アラームハンドラが動作している状態であった場合には, 登録解除の前に, アラームハンドラを起動するシステム時刻が設定解除され, 動作していない状態となる.



## 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、del\_almをサポートしない。

-----  
 sta\_alm アラームハンドラの動作開始 [T]  
 ista\_alm アラームハンドラの動作開始 [I]

## 【C言語API】

```
ER ercd = sta_alm(ID almid, RELTIM almtim)
ER ercd = ista_alm(ID almid, RELTIM almtim)
```

## 【パラメータ】

ID	almid	対象アラームハンドラのID番号
RELTIM	almtim	アラームハンドラの起動時刻（相対時間）

## 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

## 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し：sta_almの場合、タスクコンテキストからの呼出し：ista_almの場合、CPUロック状態からの呼出し）
E_ID	不正ID番号（almidが不正）
E_PAR	パラメータエラー（almtimが不正）
E_NOEXS [D]	オブジェクト未登録（対象アラームハンドラが未登録）
E_OACV [P]	オブジェクトアクセス違反（対象アラームハンドラに対する通常操作1が許可されていない：sta_almの場合）

## 【機能】

almidで指定したアラームハンドラ（対象アラームハンドラ）を動作開始する。具体的な振舞いは以下の通り。

対象アラームハンドラが動作していない状態であれば、対象アラームハンドラは動作している状態となる。アラームハンドラを起動するシステム時刻は、sta\_almを呼び出してから、almtimで指定した相対時間後に設定される。

対象アラームハンドラが動作している状態であれば、アラームハンドラを起動するシステム時刻の再設定のみが行われる。

almtimは、TMAX\_RELTIM以下でなければならない。

-----  
 stp\_alm アラームハンドラの動作停止 [T]  
 istp\_alm アラームハンドラの動作停止 [I]

## 【C言語API】

```
ER ercd = stp_alm(ID almid)
ER ercd = istp_alm(ID almid)
```

## 【パラメータ】

ID	almid	対象アラームハンドラのID番号
----	-------	-----------------

## 【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

## 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出
-------	---------------------------

し：stp\_almの場合，タスクコンテキストからの呼出し：  
 istp\_almの場合，CPUロック状態からの呼出し）  
 E\_ID 不正ID番号（almidが不正）  
 E\_NOEXS [D] オブジェクト未登録（対象アラームハンドラが未登録）  
 E\_OACV [P] オブジェクトアクセス違反（対象アラームハンドラに対  
 する通常操作2が許可されていない：stp\_almの場合）

### 【機能】

almidで指定したアラームハンドラ（対象アラームハンドラ）を動作停止する．  
 具体的な振舞いは以下の通り．

対象アラームハンドラが動作している状態であれば，アラームハンドラを起動  
 するシステム時刻が設定解除され，動作していない状態となる．対象アラーム  
 ハンドラが動作していない状態であれば，何もしない．

-----  
 ref\_alm アラームハンドラの状態参照 [T]

### 【C言語API】

```
ER ercd = ref_alm(ID almid, T_RALM *pk_ralm)
```

### 【パラメータ】

ID	almid	対象アラームハンドラのID番号
T_RALM *	pk_ralm	アラームハンドラの現在状態を入れるパケット へのポインタ

### 【リターンパラメータ】

ER	ercd	正常終了（E_OK）またはエラーコード
----	------	---------------------

\* アラームハンドラの現在状態（パケットの内容）

STAT	almstat	アラームハンドラの動作状態
RELTIM	lefttim	アラームハンドラを起動する時刻までの相対時間

### 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出 し，CPUロック状態からの呼出し）
E_ID	不正ID番号（almidが不正）
E_NOEXS [D]	オブジェクト未登録（対象アラームハンドラが未登録）
E_OACV [P]	オブジェクトアクセス違反（対象アラームハンドラに対 する参照操作が許可されていない）
E_MACV [P]	メモリアクセス違反（pk_ralmが指すメモリ領域への書込 みアクセスが許可されていない）

### 【機能】

almidで指定したアラームハンドラ（対象アラームハンドラ）の現在状態を参照  
 する．参照した現在状態は，pk\_ralmで指定したパケットに返される．

almstatには，対象アラームハンドラの現在の動作状態を表す次のいずれかの値  
 が返される．

TALM_STP	0x01U	動作していない状態
TALM_STA	0x02U	動作している状態

対象アラームハンドラが動作している状態である場合には，lefttimに，アラーム  
 ハンドラ起動する時刻までの相対時間が返される．対象アラームハンドラが  
 動作していない状態である場合には，lefttimの値は保証されない．

## 【使用上の注意】

ref\_almはデバッグ時向けの機能であり，その他の目的に使用することは推奨しない．これは，ref\_almを呼び出し，対象アラームハンドラの現在状態を参照した直後に割込みが発生した場合，ref\_almから戻ってきた時には対象アラームハンドラの状態が変化している可能性があるためである．

## 【μITRON4.0仕様との関係】

TALM\_STPとTALM\_STAを値を変更した．

---

## 4.6.4 オーバランハンドラ

未完成

C言語によるオーバランハンドラの記述形式は次の通り．

```
void overrun_handler(ID tskid, intptr_t exinf)
{
    オーバランハンドラ本体
}
```

tskidにはオーバランを起こしたタスクのID番号が，exinfにはそのタスクの拡張情報が，それぞれ渡される．

## 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは，オーバランハンドラをサポートしていない．

---

```
DEF_OVR    オーバランハンドラの定義〔S〕
def_ovr    オーバランハンドラの定義〔TD〕
```

## 【静的API】

```
DEF_OVR({ ATR ovratr, OVRHDR ovrhdr })
```

## 【C言語API】

```
ER ercd = def_ovr(const T_DOVR *pk_dovr)
```

未完成

---

```
sta_ovr    オーバランハンドラの動作開始〔T〕
```

## 【C言語API】

```
ER ercd = sta_ovr(ID tskid, OVRTIM ovrtime)
```

未完成

---

```
stp_ovr    オーバランハンドラの動作停止〔T〕
```

## 【C言語API】

```
ER ercd = stp_ovr(ID tskid)
```

未完成

---

```
ref_ovr    オーバランハンドラの状態参照〔T〕
```

## 【C言語API】

```
ER ercd = ref_ovr(ID tskid, T_ROVR *pk_rovr)
```

未完成

---

#### 4.7 システム状態管理機能

未完成

---

SAC\_SYS システム状態のアクセス許可ベクタの設定 [ SP ]  
 sac\_sys システム状態のアクセス許可ベクタの設定 [ TPD ]

##### 【静的API】

```
SAC_SYS({ ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
```

##### 【C言語API】

```
ER ercd = sac_sys(const ACVCT *p_acvct)
```

##### 【パラメータ】

ACVCT \* p\_acvct アクセス許可ベクタを入れたパケットへのポインタ (静的APIを除く)

\* アクセス許可ベクタ (パケットの内容)

ACPTN	acptn1	通常操作1のアクセス許可パターン
ACPTN	acptn2	通常操作2のアクセス許可パターン
ACPTN	acptn3	管理操作のアクセス許可パターン
ACPTN	acptn4	参照操作のアクセス許可パターン

##### 【リターンパラメータ】

ER ercd 正常終了 (E\_OK) またはエラーコード

##### 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
E_OACV [ P ]	オブジェクトアクセス違反 (システム状態に対する管理操作が許可されていない)
E_MACV [ P ]	メモリアクセス違反 (p_acvctが指すメモリ領域への読出しアクセスが許可されていない)

##### 【機能】

システム状態のアクセス許可ベクタ (4つのアクセス許可パターンの組) を, 各パラメータで指定した値に設定する.

##### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは, SAC\_SYS, sac\_sysをサポートしない.

---

rot\_rdq タスクの優先順位の回転 [ T ]  
 irot\_rdq タスクの優先順位の回転 [ I ]

##### 【C言語API】

```
ER ercd = rot_rdq(PRI tskpri)
ER ercd = irot_rdq(PRI tskpri)
```

##### 【パラメータ】

PRI tskpri 回転対象の優先度 (対象優先度)

## 【リターンパラメータ】

ER ercd 正常終了 (E\_OK) またはエラーコード

## 【エラーコード】

E\_CTX コンテキストエラー (非タスクコンテキストからの呼出し: rot\_rdqの場合, タスクコンテキストからの呼出し: irot\_rdqの場合, CPUロック状態からの呼出し)  
 E\_PAR パラメータエラー (tskpriが不正)  
 E\_OACV [P] オブジェクトアクセス違反 (システム状態に対する通常操作1が許可されていない)

## 【機能】

tskpriで指定した優先度 (対象優先度) を持つ実行できる状態のタスクの中で, 最も優先順位が高いタスクを, 同じ優先度のタスクの中で最も優先順位が低い状態にする.

rot\_rdqにおいて, tskpriにTPRI\_SELF (=0) を指定すると, 自タスクのベース優先度が対象優先度となる.

tskpriは, TPRI\_SELFであるか (rot\_rdqの場合のみ), TMIN\_TPRI以上, TMAX\_TPRI以下でなければならない.

-----  
 get\_tid 実行状態のタスクIDの参照 [T]  
 iget\_tid 実行状態のタスクIDの参照 [I]

## 【C言語API】

ER ercd = get\_tid(ID \*p\_tskid)  
 ER ercd = iget\_tid(ID \*p\_tskid)

## 【パラメータ】

ID \* p\_tskid タスクIDを入れるメモリ領域へのポインタ

## 【リターンパラメータ】

ER ercd 正常終了 (E\_OK) またはエラーコード  
 ID tskid タスクID

## 【エラーコード】

E\_CTX コンテキストエラー (非タスクコンテキストからの呼出し: get\_tidの場合, タスクコンテキストからの呼出し: iget\_tidの場合, CPUロック状態からの呼出し)  
 E\_MACV [P] メモリアクセス違反 (p\_tskidが指すメモリ領域への書込みアクセスが許可されていない)

## 【機能】

実行状態のタスクのID番号を参照する. 参照したタスクIDは, p\_tskidで指定したメモリ領域に返される.

iget\_tidにおいて, 実行状態のタスクがない場合には, TSK\_NONE (=0) が返される.

-----  
 loc\_cpu CPUロック状態への遷移 [T]  
 iloc\_cpu CPUロック状態への遷移 [I]

## 【C言語API】

```
ER ercd = loc_cpu()
ER ercd = iloc_cpu()
```

## 【パラメータ】

なし

## 【リターンパラメータ】

ER ercd 正常終了 (E\_OK) またはエラーコード

## 【エラーコード】

E\_CTX コンテキストエラー (非タスクコンテキストからの呼出し: loc\_cpuの場合, タスクコンテキストからの呼出し: iloc\_cpuの場合)

## 【機能】

CPUロックフラグをセットし, CPUロック状態へ遷移する. CPUロック状態で呼び出した場合には何も起こらない.

```
unl_cpu CPUロック状態の解除 [T]
iunl_cpu CPUロック状態の解除 [I]
```

## 【C言語API】

```
ER ercd = unl_cpu()
ER ercd = iunl_cpu()
```

## 【パラメータ】

なし

## 【リターンパラメータ】

ER ercd 正常終了 (E\_OK) またはエラーコード

## 【エラーコード】

E\_CTX コンテキストエラー (非タスクコンテキストからの呼出し: loc\_cpuの場合, タスクコンテキストからの呼出し: iloc\_cpuの場合)

## 【機能】

CPUロックフラグをクリアし, CPUロック解除状態へ遷移する. CPUロック解除状態で呼び出した場合には何も起こらない.

```
dis_dsp ディスパッチの禁止 [T]
```

## 【C言語API】

```
ER ercd = dis_dsp()
```

## 【パラメータ】

なし

## 【リターンパラメータ】

ER ercd 正常終了 (E\_OK) またはエラーコード

## 【エラーコード】

E\_CTX コンテキストエラー (非タスクコンテキストからの呼出し)

## 【機能】

ディスパッチ禁止フラグをセットし，ディスパッチ禁止状態へ遷移する．ディスパッチ禁止状態で呼び出した場合には何も起こらない．

-----  
ena\_dsp      ディスパッチの許可〔T〕

**【C言語API】**

ER ercd = ena\_dsp()

**【パラメータ】**

なし

**【リターンパラメータ】**

ER            ercd            正常終了 (E\_OK) またはエラーコード

**【エラーコード】**

E\_CTX                    コンテキストエラー (非タスクコンテキストからの呼出し)

**【機能】**

ディスパッチ禁止フラグをクリアし，ディスパッチ許可状態へ遷移する．ディスパッチ許可状態で呼び出した場合には何も起こらない．

-----  
sns\_ctx      コンテキストの参照〔TI〕

**【C言語API】**

bool\_t state = sns\_ctx()

**【パラメータ】**

なし

**【リターンパラメータ】**

bool\_t state            コンテキスト

**【機能】**

実行中のコンテキストを参照する．具体的な振舞いは以下の通り．

sns\_ctxを非タスクコンテキストから呼び出した場合にはtrue，タスクコンテキストから呼び出した場合にはfalseが返る．

-----  
sns\_loc      CPUロック状態の参照〔TI〕

**【C言語API】**

bool\_t state = sns\_loc()

**【パラメータ】**

なし

**【リターンパラメータ】**

bool\_t state            CPUロックフラグ

**【機能】**

CPUロックフラグを参照する．具体的な振舞いは以下の通り．

sns\_locをCPUロック状態で呼び出した場合にはtrue，CPUロック解除状態で呼び

出した場合にはfalseが返る .

-----  
sns\_dsp      ディスパッチ禁止状態の参照 [ TI ]

**【C言語API】**

bool\_t state = sns\_dsp()

**【パラメータ】**

なし

**【リターンパラメータ】**

bool\_t state      ディスパッチ禁止フラグ

**【機能】**

ディスパッチ禁止フラグを参照する . 具体的な振舞いは以下の通り .

sns\_dspをディスパッチ禁止状態で呼び出した場合にはtrue , ディスパッチ許可状態で呼び出した場合にはfalseが返る .

-----  
sns\_dpn      ディスパッチ保留状態の参照 [ TI ]

**【C言語API】**

bool\_t state = sns\_dpn()

**【パラメータ】**

なし

**【リターンパラメータ】**

bool\_t state      ディスパッチ保留状態

**【機能】**

ディスパッチ保留状態であるか否かを参照する . 具体的な振舞いは以下の通り .

sns\_dpnをディスパッチ保留状態で呼び出した場合にはtrue , ディスパッチ保留状態でない状態で呼び出した場合にはfalseが返る .

-----  
sns\_ker      カーネル非動作状態の参照 [ TI ]

**【C言語API】**

bool\_t state = sns\_ker()

**【パラメータ】**

なし

**【リターンパラメータ】**

bool\_t state      カーネル非動作状態

**【機能】**

カーネルが動作中であるか否かを参照する . 具体的な振舞いは以下の通り .

sns\_kerをカーネルの初期化完了前 ( 初期化ルーチン実行中を含む ) または終了処理開始後 ( 終了処理ルーチン実行中を含む ) に呼び出した場合にはtrue , カーネルの動作中に呼び出した場合にはfalseが返る .

**【使用方法】**



sns\_kerは、カーネルが動作している時とそうでない時で、処理内容を変えたい場合に使用する。sns\_kerがtrueを返した場合、他のサービスコールを呼び出すことはできない。sns\_kerがtrueを返す時に他のサービスコールを呼び出した場合の動作は保証されない。

#### 【使用上の注意】

どちらの条件でtrueが返るか間違いやすいので注意すること。

#### 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである。

-----  
ext\_ker      カーネルの終了〔T〕

#### 【C言語API】

ER ercd = ext\_ker()

#### 【パラメータ】

なし

#### 【リターンパラメータ】

ER            ercd            エラーコード

#### 【エラーコード】

E\_OACV〔P〕      オブジェクトアクセス違反（システム状態に対する通常操作2が許可されていない）

#### 【機能】

カーネルを終了する。

ext\_kerが正常に処理された場合、ext\_kerからはリターンしない。

#### 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである。

-----  
ref\_sys      システムの状態参照〔T〕

#### 【C言語API】

ER ercd = ref\_sys(T\_RSYS \*pk\_rsys)

未完成

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、ref\_sysをサポートしていない。

### 4.8 メモリオブジェクト管理機能

未完成

### 4.9 割込み管理機能

割込み処理のプログラムは、割込みサービスルーチン（ISR）として実現するこ

とを推奨する。割り込みサービスルーチンをカーネルに登録する場合には、まず、割り込みサービスルーチンの登録対象となる割り込み要求ラインの属性を設定しておく必要がある。割り込みサービスルーチンは、カーネル内の割り込みハンドラを經由して呼び出される。

ただし、カーネルが用意する割り込みハンドラで対応できないケースに対応するために、アプリケーションで割り込みハンドラを用意することも可能である。この場合にも、割り込みハンドラをカーネルに登録する前に、割り込みハンドラの登録対象となる割り込みハンドラ番号に対応する割り込み要求ラインの属性を設定しておく必要がある。

割り込み要求ラインの属性を設定する際に指定する割り込み要求ライン属性には、次の属性を指定することができる。

TA_ENAINT	0x01U	割り込み要求禁止フラグをクリア
TA_EDGE	0x02U	エッジトリガ

ターゲットによっては、ターゲット定義の割り込み要求ライン属性を指定できる場合がある。ターゲット定義の割り込み要求ライン属性として、次の属性を予約している。

TA_POSEDGE	ポジティブエッジトリガ
TA_NEGEDGE	ネガティブエッジトリガ
TA_BOTHEDGE	両エッジトリガ
TA_LOWLEVEL	ローレベルトリガ
TA_HIGLEVEL	ハイレベルトリガ

割り込みサービスルーチンは、カーネルが実行を制御する処理単位である。割り込みサービスルーチンは、割り込みサービスルーチンIDと呼ぶID番号によって識別する。

1つの割り込み要求ラインに対して複数の割り込みサービスルーチンを登録した場合、それらの割り込みサービスルーチンは、割り込みサービスルーチン優先度の高い順にすべて呼び出される。割り込みサービスルーチン優先度が同じ場合には、登録した順（静的APIにより登録した場合には、割り込みサービスルーチンを生成するAPIをコンフィギュレーションファイル中に記述した順）で呼び出される。

割り込みサービスルーチン属性に指定できる属性はない。そのため割り込みサービスルーチン属性には、TA\_NULLを指定しなければならない。

C言語による割り込みサービスルーチンの記述形式は次の通り。

```
void interrupt_service_routine(intptr_t exinf)
{
    割り込みサービスルーチン本体
}
```

exinfには、割り込みサービスルーチンの拡張情報が渡される。

割り込みハンドラは、カーネルが実行を制御する処理単位である。割り込みハンドラは、割り込みハンドラ番号と呼ぶオブジェクト番号によって識別する。

割り込みハンドラを登録する際に指定する割り込みハンドラ属性には、ターゲット定義で、次の属性を指定することができる。

TA_NONKERNEL	0x02U	カーネル管理外の割り込み
--------------	-------	--------------

TA\_NONKERNELを指定しない場合、カーネル管理の割り込みとなる。

C言語による割り込みハンドラの記述形式は次の通り。

```
void interrupt_handler(void)
{
    割り込みハンドラ本体
}
```

割り込み管理機能に関連するカーネル構成マクロは次の通り。

TMIN_INTPRI	割り込み優先度の最小値（最高値）
TMAX_INTPRI	割り込み優先度の最大値（最低値， = -1）
TMIN_ISRPRI	割り込みサービスルーチン優先度の最小値（ = 1）
TMAX_ISRPRI	割り込みサービスルーチン優先度の最大値
TOPPERS_SUPPORT_DIS_INT	dis_intがサポートされている
TOPPERS_SUPPORT_ENA_INT	ena_intがサポートされている

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは，TMAX\_ISRPRIは16に固定されている。ただし，タスク優先度拡張パッケージでは，TMAX\_ISRPRIを256に拡張する。

---

CFG_INT	割り込み要求ラインの属性の設定〔S〕
cfg_int	割り込み要求ラインの属性の設定〔TD〕

#### 【静的API】

```
CFG_INT(INTNO intno, { ATR intatr, PRI intpri })
```

#### 【C言語API】

```
ER ercd = cfg_int(INTNO intno, const T_CINT *pk_cint)
```

#### 【パラメータ】

INTNO	intno	割り込み番号
T_CINT *	pk_cint	割り込み要求ラインの属性の設定情報を入れたパケットへのポインタ（静的APIを除く）

\* 割り込み要求ラインの属性の設定情報（パケットの内容）

ATR	intatr	割り込み要求ライン属性
PRI	intpri	割り込み優先度

#### 【リターンパラメータ】

ER	ercd	正常終了（E_OK）またはエラーコード
----	------	---------------------

#### 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し，CPUロック状態からの呼出し）
E_RSATR	予約属性（intatrが不正または使用できない）
E_PAR	パラメータエラー（intnoが不正，intpriが不正）
E_OBJJ	オブジェクト状態エラー（対象割り込み要求ラインに対してすでに属性が設定されている：CFG_INTの場合，カーネル管理の割り込みか否かとintpriの値が整合していない）

#### 【機能】

intnoで指定した割り込み要求ライン（対象割り込み要求ライン）に対して，各パラメータで指定した属性を設定する。

対象割り込み要求ラインの割り込み要求禁止フラグは，intatrにTA\_ENAINTを指定した場合にクリアされ，指定しない場合にセットされる。

静的APIにおいては，intno，intatr，intpriは整数定数式パラメータである。

cfg\_intにおいて，ターゲット定義で，複数の割り込み要求ラインの割り込み優先度が連動して設定される場合がある。

CFG\_INTにおいて，対象割り込み要求ラインに対してすでに属性が設定されている場合（言い換えると，同じ割り込み番号に対するCFG\_INTが複数ある場合）には，E\_OBJエラーとなる。

intpriに指定できる値は，基本的には，TMIN\_INTPRI以上，TMAX\_INTPRI以下の値である。ターゲット定義の拡張で，カーネル管理外の割り込み要求ラインに対しても属性を設定できる場合には，TMIN\_INTPRIよりも小さい値を指定することができる。このように拡張されている場合，カーネル管理外の割り込み要求ラインを対象として，intpriにTMIN\_INTPRI以上の値を指定した場合には，E\_OBJエラーとなる。逆に，カーネル管理の割り込み要求ラインを対象として，intpriがTMIN\_INTPRIよりも小さい値である場合にも，E\_OBJエラーとなる。

対象割り込み要求ラインに対して，設定できない割り込み属性をintatrに指定した場合にはE\_RSATRエラー，設定できない割り込み優先度をintpriに指定した場合にはE\_PARエラーとなる。ここで，設定できない割り込み属性 / 割り込み優先度には，ターゲット定義の制限によって設定できない値も含む。

#### 【補足説明】

ターゲット定義の制限によって設定できない割り込み属性 / 割り込み優先度は，主にターゲットハードウェアの制限から来るものである。例えば，対象割り込み要求ラインに対して，トリガモードや割り込み優先度が固定されていて，変更できないケースが考えられる。

cfg\_intにおいて，ターゲット定義で，複数の割り込み要求ラインの割り込み優先度が連動して設定されるのは，ターゲットハードウェアの制限により，異なる割り込み要求ラインに対して，同一の割り込み優先度しか設定できないケースに対応するための仕様である。この場合，CFG\_INTにおいては，同一の割り込み優先度しか設定できない割り込み要求ラインに対して異なる割り込み優先度を設定した場合には，E\_PARエラーとなる。

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは，CFG\_INTのみをサポートする。

#### 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていない静的APIおよびサービスコールである。

```

CRE_ISR    割り込みサービスルーチンの生成〔S〕
ATT_ISR    割り込みサービスルーチンの追加〔S〕
acre_isr   割り込みサービスルーチンの生成〔TD〕

```

#### 【静的API】

```

CRE_ISR(ID isrid, { ATR isratr, intptr_t exinf,
                  INTNO intno, ISR isr, PRI isrpri })
ATT_ISR({ ATR isratr, intptr_t exinf, INTNO intno, ISR isr, PRI isrpri })

```

#### 【C言語API】

```
ER_ID isrid = acre_isr(const T_CISR *pk_cisr)
```

### 【パラメータ】

ID	isrid	対象割込みサービスルーチンのID番号 (acre_isrを除く)
T_CISR *	pk_cisr	割込みサービスルーチンの生成情報を入れたパケットへのポインタ (静的APIを除く)

\* 割込みサービスルーチンの生成情報 (パケットの内容)

ATR	isratr	割込みサービスルーチン属性
intptr_t	exinf	割込みサービスルーチンの拡張情報
INTNO	intno	割込みサービスルーチンを登録する割込み番号
ISR	isr	割込みサービスルーチンの先頭番地
PRI	isrpri	割込みサービスルーチン優先度

### 【リターンパラメータ】

ER_ID	isrid	生成された割込みサービスルーチンのID番号 (正の値) またはエラーコード
-------	-------	---------------------------------------

### 【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
E_RSATR	予約属性 (isratrが不正または使用できない)
E_PAR	パラメータエラー (intno, isr, isrpriが不正)
E_OACV [P]	オブジェクトアクセス違反 (システム状態に対する管理操作が許可されていない)
E_MACV [P]	メモリアクセス違反 (pk_cisrが指すメモリ領域への読出しアクセスが許可されていない)
E_NOID	ID番号不足 (割り付けられる割込みサービスルーチンIDがない: acre_isrの場合)
E_OBJ	オブジェクト状態エラー (isridで指定した割込みサービスルーチンが登録済み: acre_isrを除く, その他の条件については機能説明を参照すること)

### 【機能】

各パラメータで指定した割込みサービスルーチン生成情報に従って, 割込みサービスルーチンを生成する.

ATT\_ISRによって生成された割込みサービスルーチンは, ID番号を持たない.

intnoで指定した割込み要求ラインの属性が設定されていない場合には, E\_OBJエラーとなる. また, intnoで指定した割込み番号に対応する割込みハンドラ番号に対して, 割込みハンドラを定義する機能 (DEF\_INH, def\_inh) によって割込みハンドラが定義されている場合にも, E\_OBJエラーとなる. さらに, intnoでカーネル管理外の割込みを指定した場合にも, E\_OBJエラーとなる.

isrpriは, TMIN\_ISRPRI以上, TMAX\_ISRPRI以下でなければならない.

### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは, ATT\_ISRのみをサポートする.

### 【μITRON4.0仕様との関係】

割込みサービスルーチンの生成情報に, isrpri (割込みサービスルーチンの割込み優先度) を追加した. CRE\_ISRは, μITRON4.0仕様に定義されていない静的APIである.

-----  
 SAC\_ISR 割込みサービスルーチンのアクセス許可ベクタの設定〔SP〕  
 sac\_isr 割込みサービスルーチンのアクセス許可ベクタの設定〔TPD〕

【静的API】

SAC\_ISR(ID isrid, { ACPTN acptn1, ACPTN acptn2,  
 ACPTN acptn3, ACPTN acptn4 })

【C言語API】

ER ercd = sac\_isr(ID isrid, const ACVCT \*p\_acvct)

【パラメータ】

ID	isrid	対象割込みサービスルーチンのID番号
ACVCT *	p_acvct	アクセス許可ベクタを入れたパケットへのポインタ（静的APIを除く）

\* アクセス許可ベクタ（パケットの内容）

ACPTN	acptn1	通常操作1のアクセス許可パターン
ACPTN	acptn2	通常操作2のアクセス許可パターン
ACPTN	acptn3	管理操作のアクセス許可パターン
ACPTN	acptn4	参照操作のアクセス許可パターン

【リターンパラメータ】

ER	ercd	正常終了（E_OK）またはエラーコード
----	------	---------------------

【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し，CPUロック状態からの呼出し）
E_ID	不正ID番号（isridが不正）
E_NOEXS〔D〕	オブジェクト未登録（対象割込みサービスルーチンが未登録）
E_OACV〔P〕	オブジェクトアクセス違反（対象割込みサービスルーチンに対する管理操作が許可されていない）
E_MACV〔P〕	メモリアクセス違反（p_acvctが指すメモリ領域への読出しアクセスが許可されていない）
E_OBJ	オブジェクト状態エラー（対象割込みサービスルーチンは静的APIで生成された：静的APIを除く）

【機能】

isridで指定した割込みサービスルーチン（対象割込みサービスルーチン）のアクセス許可ベクタ（4つのアクセス許可パターンの組）を，各パラメータで指定した値に設定する．

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは，SAC\_ISR，sac\_isrをサポートしない．

【未決定事項】

割込みサービスルーチンのアクセス許可ベクタを設けず，システム状態のアクセス許可ベクタでアクセス保護する方法も考えられる．

-----  
 del\_isr 割込みサービスルーチンの削除〔TD〕

【C言語API】

ER ercd = del\_isr(ID isrid)

未完成

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、del\_isrをサポートしていない。

-----  
ref\_isr 割込みサービスルーチンの状態参照 [T]

【C言語API】

```
ER ercd = ref_isr(ID isrid, T_RISR *pk_risr)
```

未完成

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、ref\_isrをサポートしていない。

-----  
DEF\_INH 割込みハンドラの定義 [S]  
def\_inh 割込みハンドラの定義 [TD]

【静的API】

```
DEF_INH(INHNO inhno, { ATR inhatr, INTHDR inthdr })
```

【C言語API】

```
ER ercd = def_inh(INHNO inhno, const T_DINH *pk_dinh)
```

【パラメータ】

INHNO	inhno	割込みハンドラ番号
T_DINH *	pk_dinh	割込みハンドラの定義情報を入れたバケットへのポインタ (静的APIを除く)

\* 割込みハンドラの定義情報 (パケットの内容)

ATR	inhatr	割込みハンドラ属性
INTHDR	inthdr	割込みハンドラの先頭番地

【リターンパラメータ】

ER	ercd	正常終了 (E_OK) またはエラーコード
----	------	-----------------------

【エラーコード】

E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
E_OACV [P]	オブジェクトアクセス違反 (システム状態に対する管理操作が許可されていない)
E_MACV [P]	メモリアクセス違反 (pk_dinhが指すメモリ領域への読出しアクセスが許可されていない)
E_RSATR	予約属性 (inhatrが不正または使用できない)
E_PAR	パラメータエラー (inhnoが不正, inthdrが不正)
E_OBJ	オブジェクト状態エラー (条件については機能説明を参照すること)

【機能】

inhnoで指定した割込みハンドラ番号 (対象割込みハンドラ番号) に対して, 各パラメータで指定した割込みハンドラ定義情報に従って, 割込みハンドラを定義する. ただし, def\_inhにおいてpk\_dinhをNULLにした場合には, 対象割込みハンドラ番号に対する割込みハンドラの定義を解除する.

静的APIにおいては、`inhno`と`inhatr`は整数定数式パラメータ、`inthdr`は一般定数式パラメータである。

割込みハンドラを定義する場合（`DEF_INH`の場合および`def_inh`において`pk_dinh`をNULL以外にした場合）には、次のエラーが検出される。

対象割込みハンドラ番号に対応する割込み要求ラインの属性が設定されていない場合には、`E_OBJ`エラーとなる。また、対象割込みハンドラ番号に対してすでに割込みハンドラが定義されている場合と、対象割込みハンドラ番号に対応する割込み番号を対象に割込みサービスルーチンが登録されている場合にも、`E_OBJ`エラーとなる。

ターゲット定義の拡張で、カーネル管理外の割込みに対しても割込みハンドラを定義できる場合には、次のエラーが検出される。カーネル管理外の割込みハンドラを対象として、`inhatr`に`TA_NONKERNEL`を指定しない場合には、`E_OBJ`エラーとなる。逆に、カーネル管理の割込みハンドラを対象として、`inhatr`に`TA_NONKERNEL`を指定した場合にも、`E_OBJ`エラーとなる。また、ターゲット定義でカーネル管理外に固定されている割込みハンドラがある場合には、それを対象割込みハンドラに指定して、`inhatr`に`TA_NONKERNEL`を指定しない場合には、`E_RSATR`エラーとなる。逆に、ターゲット定義でカーネル管理に固定されている割込みハンドラがある場合には、それを対象割込みハンドラに指定して、`inhatr`に`TA_NONKERNEL`を指定した場合には、`E_RSATR`エラーとなる。

割込みハンドラの定義を解除する場合（`def_inh`において`pk_dinh`をNULLにした場合）で、対象割込みハンドラ番号に対して割込みハンドラが定義されていない場合には、`E_OBJ`エラーとなる。

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、`DEF_INH`のみをサポートする。

#### 【μITRON4.0仕様との関係】

`inthdr`のデータ型を`INTHDR`に変更した。

`def_inh`によって定義済みの割込みハンドラを再定義しようとした場合に、`E_OBJ`エラーとすることにした。割込みハンドラの定義を変更するには、一度定義を解除してから、再度定義する必要がある。

-----  
`dis_int` 割込みの禁止〔T〕

#### 【C言語API】

```
ER ercd = dis_int(INTNO intno)
```

#### 【パラメータ】

```
INTNO      intno      割込み番号
```

#### 【リターンパラメータ】

```
ER          ercd      正常終了（E_OK）またはエラーコード
```

#### 【エラーコード】

```
E_NOSPT    未サポートエラー（dis_intがサポートされていない）
E_CTX      コンテキストエラー（非タスクコンテキストからの呼出し）
E_PAR      パラメータエラー（intnoが不正、intnoで指定した割込み要求ラインに対して割込み要求禁止フラグをセットすることはできない）
E_OBJ      オブジェクト状態エラー（intnoで指定した割込み要求ラ
```



インに対して割込み属性が設定されていない)

#### 【機能】

intnoで指定した割込み要求ライン(対象割込み要求ライン)の割込み要求禁止フラグをセットする。

ターゲット定義で、dis\_intがサポートされていない場合がある。dis\_intがサポートされている場合には、TOPPERS\_SUPPORT\_DIS\_INTがマクロ定義される。サポートされていない場合にdis\_intを呼び出すと、E\_NOSPTエラーが返るか、リンク時にエラーとなる。

#### 【μITRON4.0仕様との関係】

μITRON4.0仕様で実装定義としていたintnoの意味を標準化した。

CPUロック状態でも呼び出せるものとした。

-----  
ena\_int 割込みの許可〔T〕

#### 【C言語API】

```
ER ercd = ena_int(INTNO intno)
```

#### 【パラメータ】

INTNO	intno	割込み番号
-------	-------	-------

#### 【リターンパラメータ】

ER	ercd	正常終了(E_OK)またはエラーコード
----	------	---------------------

#### 【エラーコード】

E_NOSPT	未サポートエラー(ena_intがサポートしていない)
E_CTX	コンテキストエラー(非タスクコンテキストからの呼出し)
E_PAR	パラメータエラー(intnoが不正、intnoで指定した割込み要求ラインに対して割込み要求禁止フラグをクリアすることはできない)
E_OBJ	オブジェクト状態エラー(intnoで指定した割込み要求ラインに対して割込み属性が設定されていない)

#### 【機能】

intnoで指定した割込み要求ライン(対象割込み要求ライン)の割込み要求禁止フラグをクリアする。

ターゲット定義で、ena\_intがサポートされていない場合がある。ena\_intがサポートされている場合には、TOPPERS\_SUPPORT\_ENA\_INTがマクロ定義される。サポートされていない場合にena\_intを呼び出すと、E\_NOSPTエラーが返るか、リンク時にエラーとなる。

#### 【μITRON4.0仕様との関係】

μITRON4.0仕様で実装定義としていたintnoの意味を標準化した。

CPUロック状態でも呼び出せるものとした。

-----  
ref\_int 割込み要求ラインの参照〔T〕

#### 【C言語API】

```
ER ercd = ref_int(ID intid, T_RINT *pk_rint)
```

未完成

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、ref\_intをサポートしていない。

【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである。

chg\_ipm 割込み優先度マスクの変更〔T〕

【C言語API】

```
ER ercd = chg_ipm(PRI intpri)
```

【パラメータ】

```
PRI intpri 割込み優先度マスク
```

【リターンパラメータ】

```
ER ercd 正常終了 (E_OK) またはエラーコード
```

【エラーコード】

```
E_CTX コンテキストエラー (非タスクコンテキストからの呼出し)
```

```
E_PAR パラメータエラー (intpriが不正)
```

【機能】

割込み優先度マスクを、intpriで指定した値に変更する。

intpriは、TMIN\_INTPRI以上、TIPM\_ENAALL以下でなければならない。ただし、ターゲット定義の拡張として、TMIN\_INTPRIよりも小さい値を指定できる場合がある。

【補足説明】

割込み優先度マスクをTIPM\_ENAALLに変更した場合、ディスパッチ保留状態が解除され、ディスパッチが起こる可能性がある。

【μITRON4.0仕様との関係】

μITRON4.0仕様では、サービスコールの名称およびパラメータの名称が実装定義となっているサービスコールである。

CPUロック状態でも呼び出せるものとした。

get\_ipm 割込み優先度マスクの参照〔T〕

【C言語API】

```
ER ercd = get_ipm(PRI *p_intpri)
```

【パラメータ】

```
PRI * p_intpri 割込み優先度マスクを入れるメモリ領域へのポインタ
```

【リターンパラメータ】

ER	ercd	エラーコード
PRI	intpri	割込み優先度マスク

## 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し）
-------	-----------------------------

## 【機能】

割込み優先度マスクの現在値を参照する。

## 【μITRON4.0仕様との関係】

μITRON4.0仕様では、サービスコールの名称およびパラメータの名称が実装定義となっているサービスコールである。

CPUロック状態でも呼び出せるものとした。

## 4.10 CPU例外管理機能

CPU例外ハンドラは、カーネルが実行を制御する処理単位である。CPU例外ハンドラは、CPU例外ハンドラ番号と呼ぶオブジェクト番号によって識別する。

CPU例外ハンドラ属性に指定できる属性はない。そのためCPU例外ハンドラ属性には、TA\_NULLを指定しなければならない。

C言語によるCPU例外ハンドラの記述形式は次の通り。

```
void cpu_exception_handler(void *p_excinf)
{
    CPU例外ハンドラ本体
}
```

p\_excinfには、CPU例外の情報を記憶しているメモリ領域の先頭番地が渡される。これは、CPU例外ハンドラ内で、CPU例外発生時の状態を参照する際に必要となる。

DEF_EXC	CPU例外ハンドラの定義〔S〕
def_exc	CPU例外ハンドラの定義〔TD〕

## 【静的API】

```
DEF_EXC(EXCNO excno, { ATR excatr, EXCHDR exchdr })
```

## 【C言語API】

```
ER ercd = def_exc(EXCNO excno, const T_DEXC *pk_dexc)
```

## 【パラメータ】

EXCNO	excno	CPU例外ハンドラ番号
T_DEXC *	pk_dexc	CPU例外ハンドラの定義情報を入れたパケットへのポインタ（静的APIを除く）

\*CPU例外ハンドラの定義情報（パケットの内容）

ATR	excatr	CPU例外ハンドラ属性
EXCHDR	exchdr	CPU例外ハンドラ先頭番地

## 【リターンパラメータ】

ER	ercd	正常終了（E_OK）またはエラーコード
----	------	---------------------

## 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し，CPUロック状態からの呼出し）
E_OACV [P]	オブジェクトアクセス違反（システム状態に対する管理操作が許可されていない）
E_MACV [P]	メモリアクセス違反（pk_dexcが指すメモリ領域への読出しアクセスが許可されていない）
E_RSATR	予約属性（excatrが不正または使用できない）
E_PAR	パラメータエラー（excnoが不正，exchdrが不正）
E_OBJ	オブジェクト状態エラー（定義済みのCPU例外ハンドラ番号に対する再定義，未定義のCPU例外ハンドラ番号に対する定義解除）

## 【機能】

excnoで指定したCPU例外ハンドラ番号（対象CPU例外ハンドラ番号）に対して，各パラメータで指定したCPU例外ハンドラ定義情報に従って，CPU例外ハンドラを定義する．ただし，def\_excにおいてpk\_dexcをNULLにした場合には，対象CPU例外ハンドラ番号に対するCPU例外ハンドラの定義を解除する．

静的APIにおいては，excnoとexcatrは整数定数式パラメータ，exchdrは一般定数式パラメータである．

CPU例外ハンドラを定義する場合（DEF\_EXCの場合およびdef\_excにおいてpk\_dexcをNULL以外にした場合）で，対象CPU例外ハンドラ番号に対してすでにCPU例外ハンドラが定義されている場合には，E\_OBJエラーとなる．

CPU例外ハンドラの定義を解除する場合（def\_excにおいてpk\_dexcをNULLにした場合）で，対象CPU例外ハンドラ番号に対してCPU例外ハンドラが定義されていない場合には，E\_OBJエラーとなる．

## 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは，DEF\_EXCのみをサポートする．

## 【μITRON4.0仕様との関係】

def\_excによって，定義済みのCPU例外ハンドラを再定義しようとした場合に，E\_OBJエラーとすることにした．

-----  
xsns\_dpn CPU例外発生時のディスパッチ保留状態の参照 [TI]

## 【C言語API】

```
bool_t stat = xsns_dpn(void *p_excinf)
```

## 【パラメータ】

void *	p_excinf	CPU例外の情報を記憶しているメモリ領域の先頭番地
--------	----------	---------------------------

## 【リターンパラメータ】

bool_t	state	ディスパッチ保留状態
--------	-------	------------

## 【機能】

CPU例外発生時のディスパッチ保留状態を参照する．具体的な振舞いは以下の通り．

実行中のCPU例外ハンドラの起動原因となったCPU例外が、タスクコンテキストで発生し、そのタスクがディスパッチできる状態である場合にfalse、そうでない場合にtrueが返る。

保護機能対応のカーネルにおいて、xsns\_dpnをカーネルドメイン以外から呼び出した場合には、trueが返る。

p\_excinfには、CPU例外ハンドラに渡されるp\_excinfパラメータをそのまま渡す。

#### 【使用方法】

xsns\_dpnは、CPU例外ハンドラの中で、どのようなリカバリ処理が可能かを判別したい場合に使用する。xsns\_dpnがfalseを返した場合（trueを返した場合ではないので注意すること）、非タスクコンテキスト用のサービスコールを用いてCPU例外を起こしたタスクよりも優先度の高いタスクを起動または待ち解除し、そのタスクでリカバリ処理を行うことができる。ただし、CPU例外を起こしたタスクが最高優先度の場合には、この方法でリカバリ処理を行うことはできない。

#### 【使用上の注意】

xsns\_dpnは、E\_CTXエラーを返すことがないために〔TI〕となっているが、CPU例外ハンドラから呼び出すためのものである。CPU例外ハンドラ以外から呼び出した場合や、p\_excinfに正しい値を渡さなかった場合、このサービスコールが返す値は意味を持たない。

どちらの条件でtrueが返るか間違いやすいので注意すること。

#### 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである。

-----  
xsns\_xpn CPU例外発生時のタスク例外処理保留状態の参照〔TI〕

#### 【C言語API】

```
bool_t stat = xsns_xpn(void *p_excinf)
```

#### 【パラメータ】

```
void *      p_excinf      CPU例外の情報を記憶しているメモリ領域の先頭番地
```

#### 【リターンパラメータ】

```
bool_t      state        タスク例外処理保留状態
```

#### 【機能】

CPU例外発生時のタスク例外処理保留状態を参照する。具体的な振舞いは以下の通り。

実行中のCPU例外ハンドラの起動原因となったCPU例外が、タスク自身のコンテキスト（拡張サービスコールルーチンのコンテキストを除く）で発生し、そのタスクがタスク例外処理ルーチンを実行できる状態である場合にfalse、そうでない場合にtrueが返る。

保護機能対応のカーネルにおいて、xsns\_xpnをカーネルドメイン以外から呼び出した場合には、trueが返る。

p\_excinfには、CPU例外ハンドラに渡されるp\_excinfパラメータをそのまま渡す。

ターゲット定義の制限として、CPU例外がタスクコンテキストで発生し、そのタスクがタスク例外処理ルーチンを実行できる状態であっても、そのタスクが割り込み優先度マスクをTMIN\_INTPRI（またはそれよりも高い値）に設定している場合には、trueが返る場合がある。

#### 【使用方法】

xsns\_xpnは、CPU例外ハンドラの中で、どのようなリカバリ処理が可能かを判別したい場合に使用する。xsns\_xpnがfalseを返した場合（trueを返した場合ではないので注意すること）、非タスクコンテキスト用のサービスコールを用いてCPU例外を起こしたタスクにタスク例外を要求し、タスク例外処理ルーチンでリカバリ処理を行うことができる。

#### 【使用上の注意】

xsns\_xpnは、E\_CTXエラーを返すことがないために〔TI〕となっているが、CPU例外ハンドラから呼び出すためのものである。CPU例外ハンドラ以外から呼び出した場合や、p\_excinfに正しい値を渡さなかった場合、このサービスコールが返す値は意味を持たない。

どちらの条件でtrueが返るか間違いやすいので注意すること。

#### 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである。

### 4.11 拡張サービスコール管理機能

拡張サービスコールは、非特権モードで実行される処理単位から、特権モードで実行すべきルーチンを呼び出すための機能である。特権モードで実行するルーチンを、拡張サービスコールルーチンと呼ぶ。拡張サービスコールルーチンは、特権モードで実行される処理単位からも呼び出すことができる。

保護機能対応でないカーネルでは、非特権モードと特権モードの区別がないため、拡張サービスコール管理機能をサポートしない。

C言語による拡張サービスコールルーチンの記述形式は次の通り。

```
ER_UINT extended_svc_routine(intptr_t par1, intptr_t par2,
                              intptr_t par3, intptr_t par4, intptr_t par5)
{
    拡張サービスコールルーチン本体
}
```

par1～par5には、拡張サービスコールルーチンに対するパラメータが渡される。

拡張サービスコール管理機能に関連するカーネル構成マクロは次の通り。

```
TNUM_FNCD    登録できる拡張サービスコールの数（動的生成対応でない
              カーネルでは、静的APIによって登録された拡張サービス
              コールの数に一致）
```

#### 【μITRON4.0仕様との関係】

この仕様では、拡張サービスコールに対するパラメータを、intptr\_t型のパラメータ5個に固定した。

拡張サービスコールルーチンが呼び出される時に、スタックの残り領域のサイズをチェックする機能を追加した。

TNUM\_FNCDDは、μITRON4.0仕様に規定されていないカーネル構成マクロである。

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、拡張サービスコール管理機能をサポートしていない。

```
DEF_SVC      拡張サービスコールの定義〔SP〕
def_svc      拡張サービスコールの定義〔TPD〕
```

#### 【静的API】

```
DEF_SVC(FN fncd, { ATR svcatr, EXTSVC svcrtn, SIZE stksz })
```

#### 【C言語API】

```
ER ercd = def_svc(FN fncd, const T_DSVC *pk_dsvc, SIZE stksz)
```

#### 【パラメータ】

FN	fncd	拡張サービスコールの機能コード
T_DSVC *	pk_dsvc	拡張サービスコールの定義情報を入れたパケットへのポインタ（静的APIを除く）

\* 拡張サービスコールの定義情報（パケットの内容）

ATR	svcatr	拡張サービスコール属性
EXTSVC	svcrtn	拡張サービスコールルーチンの先頭番地
SIZE	stksz	拡張サービスコールで使用するスタックサイズ

#### 【リターンパラメータ】

ER	ercd	正常終了（E_OK）またはエラーコード
----	------	---------------------

#### 【エラーコード】

E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し、CPUロック状態からの呼出し）
E_OACV〔P〕	オブジェクトアクセス違反（システム状態に対する管理操作が許可されていない）
E_MACV〔P〕	メモリアクセス違反（pk_dsvcが指すメモリ領域への読出しアクセスが許可されていない）
E_RSATR	予約属性（svcatrが不正または使用できない）
E_PAR	パラメータエラー（fncdが不正、svcrtnが不正）
E_OBJ	オブジェクト状態エラー（定義済みの機能コードに対する再定義、未定義の機能コードに対する定義解除）

#### 【機能】

fncdで指定した機能コード（対象機能コード）に対して、各パラメータで指定した拡張サービスコール定義情報に従って、拡張サービスコールを定義する。ただし、def\_svcにおいてpk\_dsvcをNULLにした場合には、対象機能コードに対する拡張サービスコールの定義を解除する。

静的APIにおいては、fncd、svcatr、stkszは整数定数式パラメータ、svchrは一般定数式パラメータである。

拡張サービスコールを定義する場合（DEF\_SVCの場合およびdef\_svcにおいてpk\_dsvcをNULL以外にした場合）で、対象機能コードに対してすでに拡張サービスコールが定義されている場合には、E\_OBJエラーとなる。

拡張サービスコールの定義を解除する場合（def\_svcにおいてpk\_dsvcをNULLに

した場合)で、対象機能コードに対して拡張サービスコールが定義されていない場合には、E\_OBJエラーとなる。

拡張サービスコールの機能コードには、正の値を用いる。fncdが0または負の値の場合には、E\_PARエラーとなる。また、fncdがTNUM\_FNCNDよりも大きい場合にも、E\_PARエラーとなる。

#### 【μITRON4.0仕様との関係】

拡張サービスコールの定義情報に、stksz (拡張サービスコールで使用するスタックサイズ)を追加した。

svcrtnのデータ型を、EXTSVCに変更した。

-----  
cal\_svc 拡張サービスコールの呼出し [TIP]

#### 【C言語API】

```
ER_UINT ercd = cal_svc(FN fncd, intptr_t par1, intptr_t par2,
                      intptr_t par3, intptr_t par4, intptr_t par5)
```

#### 【パラメータ】

FN	fncd	呼び出す拡張サービスコールの機能コード
intptr_t	par1	拡張サービスコールへの第1パラメータ
intptr_t	par2	拡張サービスコールへの第2パラメータ
intptr_t	par3	拡張サービスコールへの第3パラメータ
intptr_t	par4	拡張サービスコールへの第4パラメータ
intptr_t	par5	拡張サービスコールへの第5パラメータ

#### 【リターンパラメータ】

ER_UINT	ercd	正常終了 (正の値または0) またはエラーコード
---------	------	--------------------------

#### 【エラーコード】

E_SYS	システムエラー (拡張サービスコールのネストレベルが上限を超える)
E_RSFN	予約機能コード (fncdが不正。fncdに対して拡張サービスコールが定義されていない)
E_NOMEM	メモリ不足 (スタックの残り領域が不足)

\* その他、拡張サービスコールルーチンが返すエラーコードがそのまま返る。

#### 【機能】

fncdで指定した機能コードの拡張サービスコールルーチンを、par1, par2, ..., par5をパラメータとして呼び出し、拡張サービスコールの返値を返す。

fncdが不正な値である場合や、fncdで指定した機能コードに対して拡張サービスコールが定義されていない場合には、E\_RSFNエラーとなる。スタックの残り領域が、拡張サービスコールで使用するスタックサイズよりも小さい場合には、E\_NOMEMエラーとなる。また、拡張サービスコールのネストレベルが上限 (255) を超える場合には、E\_SYSエラーが返る。

#### 【μITRON4.0仕様との関係】

μITRON4.0仕様では、cal\_svcでカーネルのサービスコールを呼び出せるかどうかは実装定義としているが、この仕様では、カーネルのサービスコールを呼び出せないこととした。また、拡張サービスコールに対するパラメータを、intptr\_t型のパラメータ5個に固定した。

パラメータの型と数を固定したのは、型チェックを厳密にできるようにし、パ



ラメータをコンパイラやコーリングコンベンションによらずに正しく渡せるようにするためである。

また、cal\_svcから返るエラー（E\_SYS, E\_RSFN, E\_NOMEM）について規定した。

---

#### 4.12 システム構成管理機能

初期化ルーチンは、カーネルが実行を制御する処理単位で、カーネルの動作開始前に、カーネル非動作状態で実行される。

初期化ルーチン属性に指定できる属性はない。そのため初期化ルーチン属性には、TA\_NULLを指定しなければならない。

C言語による初期化ルーチンの記述形式は次の通り。

```
void initialization_routine(intptr_t exinf)
{
    初期化ルーチン本体
}
```

exinfには、初期化ルーチンの拡張情報が渡される。

終了処理ルーチンは、カーネルが実行を制御する処理単位で、カーネルの動作終了後に、カーネル非動作状態で実行される。

終了処理ルーチン属性に指定できる属性はない。そのため終了処理ルーチン属性には、TA\_NULLを指定しなければならない。

C言語による終了処理ルーチンの記述形式は次の通り。

```
void termination_routine(intptr_t exinf)
{
    終了処理ルーチン本体
}
```

exinfには、終了処理ルーチンの拡張情報が渡される。

#### 【μITRON4.0仕様との関係】

非タスクコンテキスト用スタック領域の設定と、終了処理ルーチンは、μITRON4.0仕様に規定されていない機能である。

---

DEF\_ICS 非タスクコンテキスト用スタック領域の設定〔S〕

#### 【静的API】

```
DEF_ICS({ SIZE istksz, STK_T *istk })
```

#### 【パラメータ】

\* 非タスクコンテキスト用スタック領域の設定情報

SIZE	istksz	非タスクコンテキスト用スタック領域のサイズ
STK_T	istk	非タスクコンテキスト用スタック領域の先頭番地

#### 【エラーコード】

E_PAR	パラメータエラー（istkszが不正, istkが不正）
E_OBJ	オブジェクト状態エラー（非タスクコンテキスト用スタック領域がすでに設定されている）

## 【機能】

各パラメータで指定した非タスクコンテキスト用スタック領域の設定情報に従って、非タスクコンテキスト用スタック領域を設定する。

istkszは整数定数式パラメータ、istklは一般定数式パラメータである。

istkをNULLとした場合、istkszで指定したサイズのスタック領域を、コンフィギュレータが確保する。istkszにターゲットシステムの制約に合致しないサイズを指定した時には、ターゲットシステムの制約に合致するようにサイズを大きい方に丸めて確保する。

istkがNULLでない場合、istkszで指定したサイズのスタック領域を、アプリケーションで確保する。スタック領域をアプリケーションで確保するために用意しているデータ型とマクロについては、CRE\_TSKの機能説明を参照すること。

istkやistkszにターゲットシステムの制約に合致しない先頭番地やサイズを指定した時には、E\_PARエラーとなる。

## 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていない静的APIである。

-----  
ATT\_INI 初期化ルーチンの追加〔S〕

## 【静的API】

ATT\_INI({ ATR iniatr, intptr\_t exinf, INIRTN inirtn })

## 【パラメータ】

\* 初期化ルーチンの追加情報

ATR	iniatr	初期化ルーチン属性
intptr_t	exinf	初期化ルーチンの拡張情報
INIRTN	inirtn	初期化ルーチンの先頭番地

## 【エラーコード】

E_RSATR	予約属性 ( iniatrが不正または使用できない )
E_PAR	パラメータエラー ( inirtnが不正 )

## 【機能】

各パラメータで指定した初期化ルーチン追加情報に従って、初期化ルーチンを追加する。

iniatrは整数定数式パラメータ、exinfとinirtnは一般定数式パラメータである。

-----  
ATT\_TER 終了処理ルーチンの追加〔S〕

## 【静的API】

ATT\_TER({ ATR teratr, intptr\_t exinf, TERRTN terrtn })

## 【パラメータ】

\* 終了処理ルーチンの追加情報

ATR	teratr	終了処理ルーチン属性
intptr_t	exinf	終了処理ルーチンの拡張情報
TERRTN	terrtn	終了処理ルーチンの先頭番地

## 【エラーコード】

E_RSATR	予約属性 ( teratrが不正または使用できない )
---------	-----------------------------

E\_PAR パラメータエラー ( terrtnが不正 )

**【機能】**

各パラメータで指定した終了処理ルーチン追加情報に従って、終了処理ルーチンを追加する。

teratrは整数定数式パラメータ，exinfとterrtnは一般定数式パラメータである。

**【μITRON4.0仕様との関係】**

μITRON4.0仕様に定義されていない静的APIである。

-----  
ref\_cfg コンフィギュレーション情報の参照 [ T ]

**【C言語API】**

```
ER ercd = ref_cfg(T_RCFG *pk_rcfg)
```

未完成

**【TOPPERS/ASPカーネルにおける規定】**

ASPカーネルでは、ref\_cfgをサポートしない。

-----  
ref\_ver バージョン情報の参照 [ T ]

**【C言語API】**

```
ER ercd = ref_ver(T_RVER *pk_rver)
```

未完成

**【TOPPERS/ASPカーネルにおける規定】**

ASPカーネルでは、ref\_verをサポートしない。  
-----

## 第5章 リファレンス

### 5.1 サービスコール一覧

#### (1) タスク管理機能

```
ER_ID tskid = acre_tsk(const T_CTSK *pk_ctsk)
ER ercd = sac_tsk(ID tskid, const ACVCT *p_acvct)
ER ercd = del_tsk(ID tskid)
ER ercd = act_tsk(ID tskid)
ER ercd = iact_tsk(ID tskid)
ER_UINT actcnt = can_act(ID tskid)
ER ercd = ext_tsk()
ER ercd = ter_tsk(ID tskid)
ER ercd = chg_pri(ID tskid, PRI tskpri)
ER ercd = get_pri(ID tskid, PRI *p_tskpri)
ER ercd = get_inf(intptr_t *p_exinf)
ER ercd = ref_tsk(ID tskid, T_RTsk *pk_rtsk)
```

#### (2) タスク付属同期機能

```

ER ercd = slp_tsk()
ER ercd = tslp_tsk(TMO tmout)
ER ercd = wup_tsk(ID tskid)
ER ercd = iwup_tsk(ID tskid)
ER_UINT wupcnt = can_wup(ID tskid)
ER ercd = rel_wai(ID tskid)
ER ercd = irel_wai(ID tskid)
ER ercd = sus_tsk(ID tskid)
ER ercd = rsm_tsk(ID tskid)
ER ercd = dis_wai(ID tskid)
ER ercd = idis_wai(ID tskid)
ER ercd = ena_wai(ID tskid)
ER ercd = iena_wai(ID tskid)
ER ercd = dly_tsk(RELTIM dlytim)

```

### (3) タスク例外処理機能

```

ER ercd = def_tex(ID tskid, const T_DTEX *pk_dtex)
ER ercd = ras_tex(ID tskid, TEXPTN rasptn)
ER ercd = iras_tex(ID tskid, TEXPTN rasptn)
ER ercd = dis_tex()
ER ercd = ena_tex()
bool_t state = sns_tex()
ER ercd = ref_tex(ID tskid, T_RTEX *pk_rtex)

```

### (4) 同期・通信機能

#### セマフォ

```

ER_ID semid = acre_sem(const T_CSEM *pk_csem)
ER ercd = sac_sem(ID semid, const ACVCT *p_acvct)
ER ercd = del_sem(ID semid)
ER ercd = sig_sem(ID semid)
ER ercd = isig_sem(ID semid)
ER ercd = wai_sem(ID semid)
ER ercd = pol_sem(ID semid)
ER ercd = twai_sem(ID semid, TMO tmout)
ER ercd = ini_sem(ID semid)
ER ercd = ref_sem(ID semid, T_RSEM *pk_rsem)

```

#### イベントフラグ

```

ER_ID flgid = acre_flg(const T_CFLG *pk_cflg)
ER ercd = sac_flg(ID flgid, const ACVCT *p_acvct)
ER ercd = del_flg(ID flgid)
ER ercd = set_flg(ID flgid, FLGPTN setptn)
ER ercd = iset_flg(ID flgid, FLGPTN setptn)
ER ercd = clr_flg(ID flgid, FLGPTN clrptn)
ER ercd = wai_flg(ID flgid, FLGPTN waiptn, MODE wfmode, FLGPTN *p_flgptn)
ER ercd = pol_flg(ID flgid, FLGPTN waiptn, MODE wfmode, FLGPTN *p_flgptn)
ER ercd = twai_flg(ID flgid, FLGPTN waiptn,
                   MODE wfmode, FLGPTN *p_flgptn, TMO tmout)
ER ercd = ini_flg(ID flgid)
ER ercd = ref_flg(ID flgid, T_RFLG *pk_rflg)

```

#### データキュー

```

ER_ID dtqid = acre_dtq(const T_CDTQ *pk_cdtq)

```

```

ER ercd = sac_dtq(ID dtqid, const ACVCT *p_acvct)
ER ercd = del_dtq(ID dtqid)
ER ercd = snd_dtq(ID dtqid, intptr_t data)
ER ercd = psnd_dtq(ID dtqid, intptr_t data)
ER ercd = ipsnd_dtq(ID dtqid, intptr_t data)
ER ercd = tsnd_dtq(ID dtqid, intptr_t data, TMO tmout)
ER ercd = fsnd_dtq(ID dtqid, intptr_t data)
ER ercd = ifsnd_dtq(ID dtqid, intptr_t data)
ER ercd = rcv_dtq(ID dtqid, intptr_t *p_data)
ER ercd = prcv_dtq(ID dtqid, intptr_t *p_data)
ER ercd = trcv_dtq(ID dtqid, intptr_t *p_data, TMO tmout)
ER ercd = ini_dtq(ID dtqid)
ER ercd = ref_dtq(ID dtqid, T_RDTQ *pk_rdtq)

```

#### 優先度データキュー

```

ER_ID pdqid = acre_pdq(const T_CPDQ *pk_cpdq)
ER ercd = sac_pdq(ID pdqid, const ACVCT *p_acvct)
ER ercd = del_pdq(ID pdqid)
ER ercd = snd_pdq(ID pdqid, intptr_t data, PRI datapri)
ER ercd = psnd_pdq(ID pdqid, intptr_t data, PRI datapri)
ER ercd = ipsnd_pdq(ID pdqid, intptr_t data, PRI datapri)
ER ercd = tsnd_pdq(ID pdqid, intptr_t data, PRI datapri, TMO tmout)
ER ercd = rcv_pdq(ID pdqid, intptr_t *p_data, PRI *p_datapri)
ER ercd = prcv_pdq(ID pdqid, intptr_t *p_data, PRI *p_datapri)
ER ercd = trcv_pdq(ID pdqid, intptr_t *p_data, PRI *p_datapri, TMO tmout)
ER ercd = ini_pdq(ID pdqid)
ER ercd = ref_pdq(ID pdqid, T_RPDQ *pk_rpdq)

```

#### メールボックス

```

ER_ID mbxid = acre_mbx(const T_CMBX *pk_cmbx)
ER ercd = del_mbx(ID mbxid)
ER ercd = snd_mbx(ID mbxid, T_MSG *pk_msg)
ER ercd = rcv_mbx(ID mbxid, T_MSG **ppk_msg)
ER ercd = prcv_mbx(ID mbxid, T_MSG **ppk_msg)
ER ercd = trcv_mbx(ID mbxid, T_MSG **ppk_msg, TMO tmout)
ER ercd = ini_mbx(ID mbxid)
ER ercd = ref_mbx(ID mbxid, T_RMBX *pk_rmbx)

```

#### ミューテックス

```

ER_ID mtxid = acre_mtx(const T_CMTX *pk_cmtx)
ER ercd = sac_mtx(ID mtxid, const ACVCT *p_acvct)
ER ercd = del_mtx(ID mtxid)
ER ercd = loc_mtx(ID mtxid)
ER ercd = ploc_mtx(ID mtxid)
ER ercd = tloc_mtx(ID mtxid, TMO tmout)
ER ercd = unl_mtx(ID mtxid)
ER ercd = ini_mtx(ID mtxid)
ER ercd = ref_mtx(ID mtxid, T_RMTX *pk_rmtx)

```

#### メッセージバッファ

未完成

#### (5) メモリプール管理機能

## 固定長メモリーブール

```

ER_ID mpfid = acre_mpf(const T_CMPF *pk_cmpf)
ER ercd = sac_mpf(ID mpfid, const ACVCT *p_acvct)
ER ercd = del_mpf(ID mpfid)
ER ercd = get_mpf(ID mpfid, void **p_blk)
ER ercd = pget_mpf(ID mpfid, void **p_blk)
ER ercd = tget_mpf(ID mpfid, void **p_blk, TMO tmout)
ER ercd = rel_mpf(ID mpfid, void *blk)
ER ercd = ini_mpf(ID mpfid)
ER ercd = ref_mpf(ID mpfid, T_RMPF *pk_rmpf)

```

## (6) 時間管理機能

## システム時刻管理

```

ER ercd = get_tim(SYSTIM *p_systim)
ER ercd = get_utm(SYSUTM *p_sysutm)

```

## 周期ハンドラ

```

ER_ID cycid = acre_cyc(const T_CCYC *pk_ccyc)
ER ercd = sac_cyc(ID cycid, const ACVCT *p_acvct)
ER ercd = del_cyc(ID cycid)
ER ercd = sta_cyc(ID cycid)
ER ercd = stp_cyc(ID cycid)
ER ercd = ref_cyc(ID cycid, T_RCYC *pk_rcyc)

```

## アラームハンドラ

```

ER_ID almid = acre_alm(const T_CALM *pk_calm)
ER ercd = sac_alm(ID almid, const ACVCT *p_acvct)
ER ercd = del_alm(ID almid)
ER ercd = sta_alm(ID almid, RELTIM almtim)
ER ercd = ista_alm(ID almid, RELTIM almtim)
ER ercd = stp_alm(ID almid)
ER ercd = istp_alm(ID almid)
ER ercd = ref_alm(ID almid, T_RALM *pk_ralm)

```

## オーバランハンドラ

```

ER ercd = def_ovr(const T_DOVR *pk_dovr)
ER ercd = sta_ovr(ID tskid, OVRTIM ovrtime)
ER ercd = stp_ovr(ID tskid)
ER ercd = ref_ovr(ID tskid, T_ROVR *pk_rovr)

```

## (7) システム状態管理機能

```

ER ercd = sac_sys(const ACVCT *p_acvct)
ER ercd = rot_rdq(PRI tskpri)
ER ercd = irot_rdq(PRI tskpri)
ER ercd = get_tid(ID *p_tskid)
ER ercd = iget_tid(ID *p_tskid)
ER ercd = loc_cpu()
ER ercd = iloc_cpu()
ER ercd = unl_cpu()
ER ercd = iunl_cpu()
ER ercd = dis_dsp()

```

```

ER ercd = ena_dsp()
bool_t state = sns_ctx()
bool_t state = sns_loc()
bool_t state = sns_dsp()
bool_t state = sns_dpn()
bool_t state = sns_ker()
ER ercd = ext_ker()
ER ercd = ref_sys(T_RSYS *pk_rsys)

```

#### (8) メモリオブジェクト管理機能

未完成

#### (9) 割込み管理機能

```

ER ercd = cfg_int(INTNO intno, const T_CINT *pk_cint)
ER_ID isrid = acre_isr(const T_CISR *pk_cisr)
ER ercd = sac_isr(ID isrid, const ACVCT *p_acvct)
ER ercd = del_isr(ID isrid)
ER ercd = ref_isr(ID isrid, T_RISR *pk_risr)
ER ercd = def_inh(INHNO inhno, const T_DINH *pk_dinh)
ER ercd = dis_int(INTNO intno)
ER ercd = ena_int(INTNO intno)
ER ercd = ref_int(ID intid, T_RINT *pk_rint)
ER ercd = chg_ipm(PRI intpri)
ER ercd = get_ipm(PRI *p_intpri)

```

#### (10) CPU例外管理機能

```

ER ercd = def_exc(EXCNO excno, const T_DEXC *pk_dexc)
bool_t stat = xsns_dpn(void *p_excinf)
bool_t stat = xsns_xpn(void *p_excinf)

```

#### (11) 拡張サービスコール管理機能

```

ER ercd = def_svc(FN fcnd, const T_DSVC *pk_dsvc, SIZE stksz)
ER_UINT ercd = cal_svc(FN fcnd, intptr_t par1, intptr_t par2,
                      intptr_t par3, intptr_t par4, intptr_t par5)

```

#### (12) システム構成管理機能

```

ER ercd = ref_cfg(T_RCFG *pk_rcfg)
ER ercd = ref_ver(T_RVER *pk_rver)

```

### 5.2 静的API一覧

#### (1) タスク管理機能

```

CRE_TSK(ID tskid, { ATR tskatr, intptr_t exinf, TASK task,
                  PRI itskpri, SIZE stksz, STK_T *stk })
/* 保護機能対応カーネルの場合 */
CRE_TSK(ID tskid, { ATR tskatr, intptr_t exinf, TASK task,
                  PRI itskpri, SIZE stksz, STK_T *stk, SIZE sstksz, STK_T *sstk })
SAC_TSK(ID tskid, { ACPTN acptn1, ACPTN acptn2,
                  ACPTN acptn3, ACPTN acptn4 })

```

#### (2) タスク付属同期機能

なし

(3) タスク例外処理機能

DEF\_TEX(ID tskid, { ATR texatr, TEXRTN texrtn })

(4) 同期・通信機能

セマフォ

CRE\_SEM(ID semid, { ATR sematr, uint\_t isemcnt, uint\_t maxsem })  
SAC\_SEM(ID semid, { ACPTN acptn1, ACPTN acptn2,  
ACPTN acptn3, ACPTN acptn4 })

イベントフラグ

CRE\_FLG(ID flgid, { ATR flgatr, FLGPTN iflgptn })  
SAC\_FLG(ID flgid, { ACPTN acptn1, ACPTN acptn2,  
ACPTN acptn3, ACPTN acptn4 })

データキュー

CRE\_DTQ(ID dtqid, { ATR dtqatr, uint\_t dtqcnt, void \*dtqmb })  
SAC\_DTQ(ID dtqid, { ACPTN acptn1, ACPTN acptn2,  
ACPTN acptn3, ACPTN acptn4 })

優先度データキュー

CRE\_PDQ(ID pdqid, { ATR pdqatr, uint\_t pdqcnt, PRI maxdpri, void \*pdqmb })  
SAC\_PDQ(ID pdqid, { ACPTN acptn1, ACPTN acptn2,  
ACPTN acptn3, ACPTN acptn4 })

メールボックス

CRE\_MBX(ID mbxid, { ATR mbxatr, PRI maxmpri, void \*mprihd })

ミューテックス

CRE\_MTX(ID mtxid, { ATR mtxatr, PRI ceilpri })  
SAC\_MTX(ID mtxid, { ACPTN acptn1, ACPTN acptn2,  
ACPTN acptn3, ACPTN acptn4 })

メッセージバッファ

未完成

(5) メモリプール管理機能

固定長メモリプール

CRE\_MPF(ID mpfid, { ATR mpfatr, uint\_t blkcnt, uint\_t blksz,  
MPF\_T \*mpf, void \*mpfmb })  
SAC\_MPF(ID mpfid, { ACPTN acptn1, ACPTN acptn2,  
ACPTN acptn3, ACPTN acptn4 })

(6) 時間管理機能

周期ハンドラ



```

CRE_CYC(ID cycid, { ATR cycatr, intptr_t exinf, CYCHDR cychdr,
                  RELTIM cyctim, RELTIM cycphs })
SAC_CYC(ID cycid, { ACPTN acptn1, ACPTN acptn2,
                  ACPTN acptn3, ACPTN acptn4 })

```

#### アラームハンドラ

```

CRE_ALM(ID almid, { ATR almatr, intptr_t exinf, ALMHDR almhdr })
SAC_ALM(ID almid, { ACPTN acptn1, ACPTN acptn2,
                  ACPTN acptn3, ACPTN acptn4 })

```

#### オーバランハンドラ

```
DEF_OVR({ ATR ovratr, OVRHDR ovrhdr })
```

#### (7) システム状態管理機能

```
SAC_SYS({ ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
```

#### (8) メモリオブジェクト管理機能

未完成

#### (9) 割込み管理機能

```

CFG_INT(INTNO intno, { ATR intatr, PRI intpri })
CRE_ISR(ID isrid, { ATR isratr, intptr_t exinf,
                  INTNO intno, ISR isr, PRI isrpri })
ATT_ISR({ ATR isratr, intptr_t exinf, INTNO intno, ISR isr, PRI isrpri })
SAC_ISR(ID isrid, { ACPTN acptn1, ACPTN acptn2,
                  ACPTN acptn3, ACPTN acptn4 })
DEF_INH(INHNO inhno, { ATR inhatr, INTHDR inthdr })

```

#### (10) CPU例外管理機能

```
DEF_EXC(EXCNO excno, { ATR excatr, EXCHDR exchr })
```

#### (11) 拡張サービスコール管理機能

```
DEF_SVC(ID tskid, { ATR svcatr, EXTSVC svcrtm })
```

#### (12) システム構成管理機能

```

DEF_ICS({ SIZE istksz, STK_T *istk })
ATT_INI({ ATR iniatr, intptr_t exinf, INIRTN inirtn })
ATT_TER({ ATR teratr, intptr_t exinf, TERRTN terrtn })

```

### 5.3 データ型

#### 5.3.1 TOPPERS共通データ型

```

int8_t      符号付き8ビット整数 (オプション, C99準拠)
uint8_t     符号無し8ビット整数 (オプション, C99準拠)
int16_t     符号付き16ビット整数 (C99準拠)
uint16_t    符号無し16ビット整数 (C99準拠)
int32_t     符号付き32ビット整数 (C99準拠)
uint32_t    符号無し32ビット整数 (C99準拠)

```

```

int64_t      符号付き64ビット整数 (オプション, C99準拠)
uint64_t     符号無し64ビット整数 (オプション, C99準拠)
int128_t     符号付き128ビット整数 (オプション, C99準拠)
uint128_t    符号無し128ビット整数 (オプション, C99準拠)

int_least8_t  8ビット以上の符号付き整数 (C99準拠)
uint_least8_t int_least8_t型と同じサイズの符号無し整数 (C99準拠)

float32_t    IEEE754準拠の32ビット単精度浮動小数点数 (オプション)
double64_t   IEEE754準拠の64ビット倍精度浮動小数点数 (オプション)

bool_t       真偽値 (trueまたはfalse)
char_t       符号無しの文字型 (unsigned charと一致)
int_t        16ビット以上の符号付き整数
uint_t       int_t型と同じサイズの符号無し整数
long_t       32ビット以上かつint_t型以上のサイズの符号付き整数
ulong_t      long_t型と同じサイズの符号無し整数

intptr_t     ポインタを格納できるサイズの符号付き整数 (C99準拠)
uintptr_t    intptr_t型と同じサイズの符号無し整数 (C99準拠)

FN           機能コード (符号付き整数, int_tに定義)
ER           エラーコード (符号付き整数, int_tに定義)
ID           オブジェクトのID番号 (符号付き整数, int_tに定義)
ATR         オブジェクト属性 (符号無し整数, uint_tに定義)
STAT        オブジェクトの状態 (符号無し整数, uint_tに定義)
MODE        サービスコールの動作モード (符号無し整数, uint_tに定義)
PRI         優先度 (符号付き整数, int_tに定義)
SIZE        メモリ領域のサイズ (符号無し整数, ポインタを格納できる
            サイズの符号無し整数型に定義)

TMO         タイムアウト指定 (符号付き整数, 単位はミリ秒, int_tに定義)
RELTIM      相対時間 (符号無し整数, 単位はミリ秒, uint_tに定義)
SYSTIM      システム時刻 (符号無し整数, 単位はミリ秒, ulong_tに定義)
SYSUTM      性能評価用システム時刻 (符号無し整数, 単位はマイクロ秒,
            ulong_tに定義)

FP          プログラムの起動番地 (型の定まらない関数ポインタ)

ER_BOOL     エラーコードまたは真偽値 (符号付き整数, int_tに定義)
ER_ID       エラーコードまたはID番号 (符号付き整数, int_tに定義,
            負のID番号は格納できない)
ER_UINT     エラーコードまたは符号無し整数 (符号付き整数, int_tに
            定義, 符号無し整数を格納する場合の有効ビット数はuint_t
            より1ビット短い)

ACPTN       アクセス許可パターン (符号無し32ビット整数, uint32_tに
            定義)

typedef struct acvct {          /* アクセス許可ベクタ */
    ACPTN  acptn1;             /* 通常操作1のアクセス許可パターン */
    ACPTN  acptn2;             /* 通常操作2のアクセス許可パターン */
    ACPTN  acptn3;             /* 管理操作のアクセス許可パターン */
    ACPTN  acptn4;             /* 参照操作のアクセス許可パターン */
} ACVCT;

```

### 5.3.2 カーネルの使用するデータ型

TEXPTN      タスク例外要因のビットパターン (符号無し整数, uint\_tに定義)  
 FLGPTN      イベントフラグのビットパターン (符号無し整数, uint\_tに定義)  
 INTNO        割り込み番号 (符号無し整数, uint\_tに定義)  
 INHNO        割り込みハンドラ番号 (符号無し整数, uint\_tに定義)  
 EXCNO        CPU例外ハンドラ番号 (符号無し整数, uint\_tに定義)

TASK         タスクのメインルーチン (関数ポインタ)  
 TEXRTN       タスク例外処理ルーチン (関数ポインタ)  
 CYCHDR       周期ハンドラ (関数ポインタ)  
 ALMHDR       アラームハンドラ (関数ポインタ)  
 ISR          割り込みサービ斯拉ーチン (関数ポインタ)  
 INTHDR       割り込みハンドラ (関数ポインタ)  
 EXCHDR       CPU例外ハンドラ (関数ポインタ)  
 INIRTN       初期化ルーチン (関数ポインタ)  
 TERRTN       終了処理ルーチン (関数ポインタ)

STK\_T        スタック領域を確保するためのデータ型  
 MPF\_T        固定長メモリプール領域を確保するためのデータ型

```

typedef struct t_msg {          /* メールボックスのメッセージヘッダ */
    struct t_msg  *pk_next;
} T_MSG;

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

```

### 5.3.3 カーネルの使用するパケット形式

#### (1) タスク管理機能

##### タスクの生成情報のパケット形式

```

typedef struct t_ctsk {
    ATR          tskatr;        /* タスク属性 */
    intptr_t     exinf;        /* タスクの拡張情報 */
    TASK         task;         /* タスクのメインルーチンの先頭番地 */
    PRI          itskpri;       /* タスクの起動時優先度 */
    SIZE         stksz;        /* タスクのスタック領域のサイズ */
    STK_T *      stk;          /* タスクのスタック領域の先頭番地 */
    /* 以下は、保護機能対応カーネルの場合 */
    SIZE         sstksz;       /* タスクのシステムスタック領域のサイズ */
    STK_T *      sstk;        /* タスクのシステムスタック領域の先頭番地 */
} T_CTSK;

```

##### タスクの現在状態のパケット形式

```

typedef struct t_rtsk {
    STAT         tskstat;      /* タスク状態 */
    PRI          tskpri;       /* タスクの現在優先度 */
    PRI          tsbpri;       /* タスクのベース優先度 */
    STAT         tskswait;     /* 待ち要因 */
    ID           wobjid;       /* 待ち対象のオブジェクトのID */
    TMO         lefttmo;      /* タイムアウトするまでの時間 */
    uint_t       actcnt;       /* 起動要求キューイング数 */
    uint_t       wupcnt;       /* 起床要求キューイング数 */
} T_RTSK;

```

## (2) タスク付属同期機能

なし

## (3) タスク例外処理機能

タスク例外処理ルーチンの定義情報のパッケージ形式

```
typedef struct t_dtsk {
    ATR          texatr;    /* タスク例外処理ルーチン属性 */
    TEXRTN      texrtn;    /* タスク例外処理ルーチンの先頭番地 */
} T_DTEX;
```

タスク例外処理の現在状態のパッケージ形式

```
typedef struct t_rtex {
    STAT        texstat;   /* タスク例外処理の状態 */
    TEXPTN      pndptn;   /* 保留例外要因 */
} T_RTEX;
```

## (4) 同期・通信機能

セマフォの生成情報のパッケージ形式

```
typedef struct t_csem {
    ATR          sematr;   /* セマフォ属性 */
    uint_t      isemcnt;  /* セマフォの初期資源数 */
    uint_t      maxsem;   /* セマフォの最大資源数 */
} T_CSEM;
```

セマフォの現在状態のパッケージ形式

```
typedef struct t_rsem {
    ID          wtskid;   /* セマフォの待ち行列の先頭のタスクのID番号 */
    uint_t      semcnt;   /* セマフォの資源数 */
} T_RSEM;
```

イベントフラグの生成情報のパッケージ形式

```
typedef struct t_cflg {
    ATR          sematr;   /* イベントフラグ属性 */
    FLGPTN      iflgptn;  /* イベントフラグの初期ビットパターン */
} T_CFLG;
```

イベントフラグの現在状態のパッケージ形式

```
typedef struct t_rflg {
    ID          wtskid;   /* イベントフラグの待ち行列の先頭のタスクのID番号 */
    FLGPTN      flgptn;  /* イベントフラグのビットパターン */
} T_RFLG;
```

データキューの生成情報のパッケージ形式

```
typedef struct t_cdtq {
    ATR          dtqatr;   /* データキュー属性 */
    uint_t      dtqcnt;   /* データキュー管理領域に格納できるデータ数 */
}
```

```

    void *    dtqmb;    /* データキュー管理領域の先頭番地 */
} T_CDTQ;

```

#### データキューの現在状態のパケット形式

```

typedef struct t_rdtq {
    ID        stskid;    /* データキューの送信待ち行列の先頭のタ
                        スクのID番号 */
    ID        rtskid;    /* データキューの受信待ち行列の先頭のタ
                        スクのID番号 */
    uint_t    sdtqcnt;    /* データキュー管理領域に格納されている
                        データの数 */
} T_RDTQ;

```

#### 優先度データキューの生成情報のパケット形式

```

typedef struct t_cpdq {
    ATR        pdqatr;    /* 優先度データキュー属性 */
    uint_t     pdqcnt;    /* 優先度データキュー管理領域に格納でき
                        るデータ数 */
    PRI        maxdpri;    /* 優先度データキューに送信できるデー
                        タ優先度の最大値 */
    void *     pdqmb;    /* 優先度データキュー管理領域の先頭番地 */
} T_CPDQ;

```

#### 優先度データキューの現在状態のパケット形式

```

typedef struct t_rpdq {
    ID        stskid;    /* 優先度データキューの送信待ち行列の先
                        頭のタスクのID番号 */
    ID        rtskid;    /* 優先度データキューの受信待ち行列の先
                        頭のタスクのID番号 */
    uint_t    spdqcnt;    /* 優先度データキュー管理領域に格納され
                        ているデータの数 */
} T_RPDQ;

```

#### メールボックスの生成情報のパケット形式

```

typedef struct t_cmbx {
    ATR        mbxatr;    /* メールボックス属性 */
    PRI        maxmpri;    /* 優先度メールボックスに送信できるメッ
                        セージ優先度の最大値 */
    void *     mprihd;    /* 優先度別のメッセージキューヘッダ領域
                        の先頭番地 */
} T_CMBX;

```

#### メールボックスの現在状態のパケット形式

```

typedef struct t_rmbx {
    ID        wtskid;    /* メールボックスの待ち行列の先頭のタスク
                        のID番号 */
    T_MSG     *pk_msg;    /* メッセージキューの先頭につながれたメッ
                        セージの先頭番地 */
} T_RMBX;

```

#### ミューテックスの生成情報のパケット形式

未完成

ミューテックスの現在状態のパケット形式

未完成

メッセージバッファの生成情報のパケット形式

未完成

メッセージバッファの現在状態のパケット形式

未完成

#### (5) メモリプール管理機能

固定長メモリプールの生成情報のパケット形式

```
typedef struct t_cmpf {
    ATR      mpfatr;      /* 固定長メモリプール属性 */
    uint_t   blkcnt;     /* 獲得できる固定長メモリブロックの数 */
    uint_t   blkksz;     /* 固定長メモリブロックのサイズ */
    MPF_T *  mpf;        /* 固定長メモリプール領域の先頭番地 */
    void *   mpfmb;      /* 固定長メモリプール管理領域の先頭番地 */
} T_CMPF;
```

固定長メモリプールの現在状態のパケット形式

```
typedef struct t_rmpf {
    ID       wtskid;     /* 固定長メモリプールの待ち行列の先頭の
                        タスクのID番号 */
    uint_t   fblkcnt;   /* 固定長メモリプール領域の空きメモリ領
                        域に割り付けることができる固定長メモ
                        リブロックの数 */
} T_RMPF;
```

#### (6) 時間管理機能

周期ハンドラの生成情報のパケット形式

```
typedef struct t_ccyc {
    ATR      cycatr;     /* 周期ハンドラ属性 */
    intptr_t exinf;     /* 周期ハンドラの拡張情報 */
    CYCHDR   cychr;     /* 周期ハンドラ先頭番地 */
    RELTIM   cyctim;    /* 周期ハンドラ起動周期 */
    RELTIM   cycphs;    /* 周期ハンドラ起動位相 */
} T_CCYC;
```

周期ハンドラの現在状態のパケット形式

```
typedef struct t_rcyc {
    STAT     cycstat;    /* 周期ハンドラの動作状態 */
    RELTIM   lefttim;   /* 次に周期ハンドラを起動する時刻までの
                        相対時間 */
} T_RCYC;
```

アラームハンドラの生成情報のパケット形式

```
typedef struct t_calm {
```

```

    ATR      almatr;    /* アラームハンドラ属性 */
    intptr_t exinf;    /* アラームハンドラの拡張情報 */
    ALMHDR   almhdr;   /* アラームハンドラ先頭番地 */
} T_CALM;

```

アラームハンドラの現在状態のパケット形式

```

typedef struct t_ralm {
    STAT      almstat; /* アラームハンドラの動作状態 */
    RELTIM    lefttim; /* アラームハンドラを起動する時刻までの
                        相対時間 */
} T_RALM;

```

オーバランハンドラの定義情報のパケット形式

未完成

オーバランハンドラの現在状態のパケット形式

未完成

(7) システム状態管理機能

システムの現在状態のパケット形式

未完成

(9) 割込み管理機能

割込み要求ラインの属性の設定情報のパケット形式

```

typedef struct t_cint {
    ATR      intatr;    /* 割込み要求ライン属性 */
    PRI      intpri;   /* 割込み優先度 */
} T_CINT;

```

割込みサービスルーチンの生成情報のパケット形式

```

typedef struct t_cisr {
    ATR      isratr;   /* 割込みサービスルーチン属性 */
    intptr_t exinf;   /* 割込みサービスルーチンの拡張情報 */
    INTNO    intno;   /* 割込みサービスルーチンを登録する割込
                        み番号 */
    ISR      isr;     /* 割込みサービスルーチンの先頭番地 */
    PRI      isrpri;  /* 割込みサービスルーチン優先度 */
} T_CISR;

```

割込みサービスルーチンの現在状態のパケット形式

未完成

割込みハンドラの定義情報のパケット形式

```

typedef struct t_dinh {
    ATR      inhatr;   /* 割込みハンドラ属性 */
    INTHDR   inthdr;  /* 割込みハンドラ先頭番地 */
} T_DINH;

```

割込み要求ラインの現在の状態のパケット形式

未完成

#### (10) CPU例外管理機能

CPU例外ハンドラの定義情報のパケット形式

```
typedef struct t_dexc {
    ATR          excatr;    /* CPU例外ハンドラ属性 */
    EXCHDR      exchdr;    /* CPU例外ハンドラの先頭番地 */
} T_DEXC;
```

#### (11) 拡張サービスコール管理機能

拡張サービスコールの定義情報のパケット形式

未完成

#### (12) システム構成管理機能

コンフィギュレーション情報のパケット形式

未完成

バージョン情報のパケット形式

未完成

### 5.4 定数とマクロ

#### 5.4.1 TOPPERS共通定数

##### (1) 一般定数

NULL		無効ポインタ
true	1	真
false	0	偽
E_OK	0	正常終了

##### (2) 整数型に格納できる最大値と最小値

INT8_MAX	int8_tに格納できる最大値 (オプション, C99準拠)
INT8_MIN	int8_tに格納できる最小値 (オプション, C99準拠)
UINT8_MAX	uint8_tに格納できる最大値 (オプション, C99準拠)
INT16_MAX	int16_tに格納できる最大値 (C99準拠)
INT16_MIN	int16_tに格納できる最小値 (C99準拠)
UINT16_MAX	uint16_tに格納できる最大値 (C99準拠)
INT32_MAX	int32_tに格納できる最大値 (C99準拠)
INT32_MIN	int32_tに格納できる最小値 (C99準拠)
UINT32_MAX	uint32_tに格納できる最大値 (C99準拠)
INT64_MAX	int64_tに格納できる最大値 (オプション, C99準拠)
INT64_MIN	int64_tに格納できる最小値 (オプション, C99準拠)
UINT64_MAX	uint64_tに格納できる最大値 (オプション, C99準拠)
INT128_MAX	int128_tに格納できる最大値 (オプション, C99準拠)
INT128_MIN	int128_tに格納できる最小値 (オプション, C99準拠)



UINT128_MAX	uint128_tに格納できる最大値 (オプション, C99準拠)
INT_LEAST8_MAX	int_least8_tに格納できる最大値 (C99準拠)
INT_LEAST8_MIN	int_least8_tに格納できる最小値 (C99準拠)
UINT_LEAST8_MAX	uint_least8_tに格納できる最大値 (C99準拠)
INT_MAX	int_tに格納できる最大値 (C90準拠)
INT_MIN	int_tに格納できる最小値 (C90準拠)
UINT_MAX	uint_tに格納できる最大値 (C90準拠)
LONG_MAX	long_tに格納できる最大値 (C90準拠)
LONG_MIN	long_tに格納できる最小値 (C90準拠)
ULONG_MAX	ulong_tに格納できる最大値 (C90準拠)
FLOAT32_MIN	float32_tに格納できる最小の正規化された正の浮動小数点数 (オプション)
FLOAT32_MAX	float32_tに格納できる表現可能な最大の有限浮動小数点数 (オプション)
DOUBLE64_MIN	double64_tに格納できる最小の正規化された正の浮動小数点数 (オプション)
DOUBLE64_MAX	double64_tに格納できる表現可能な最大の有限浮動小数点数 (オプション)

## (3) 整数型のビット数

CHAR_BIT	char型のビット数 (C90準拠)
----------	--------------------

## (4) オブジェクト属性

TA_NULL	OU	オブジェクト属性を指定しない
---------	----	----------------

## (5) タイムアウト指定

TMO_POL	0	ポーリング
TMO_FEVR	-1	永久待ち
TMO_NBLK	-2	ノンブロッキング

## (6) アクセス許可パターン

TACP_KERNEL	OU	カーネルドメインだけにアクセスを許可
TACP_SHARED	OU	すべてのドメインにアクセスを許可

## 5.4.2 TOPPERS共通マクロ

## (1) 整数定数を作るマクロ

INT8_C(val)	int_least8_t型の定数を作るマクロ (C99準拠)
UINT8_C(val)	uint_least8_t型の定数を作るマクロ (C99準拠)
INT16_C(val)	int16_t型の定数を作るマクロ (C99準拠)
UINT16_C(val)	uint16_t型の定数を作るマクロ (C99準拠)
INT32_C(val)	int32_t型の定数を作るマクロ (C99準拠)
UINT32_C(val)	uint32_t型の定数を作るマクロ (C99準拠)
INT64_C(val)	int64_t型の定数を作るマクロ (オプション, C99準拠)
UINT64_C(val)	uint64_t型の定数を作るマクロ (オプション, C99準拠)
INT128_C(val)	int128_t型の定数を作るマクロ (オプション, C99準拠)
UINT128_C(val)	uint128_t型の定数を作るマクロ (オプション, C99準拠)
UINT_C(val)	uint_t型の定数を作るマクロ
ULONG_C(val)	ulong_t型の定数を作るマクロ

## (2) 型に関する情報を取り出すためのマクロ

offsetof(structure, field)	構造体structure中のフィールドfieldのバイト位置を返すマクロ (C90準拠)
alignof(type)	型typeのアラインメント単位を返すマクロ
ALIGN_TYPE(addr, type)	アドレスaddrが型typeに対してアラインしているかどうかを返すマクロ

## (3) assertマクロ

assert(exp)	expが成立しているかを検査するマクロ (C90準拠)
-------------	-----------------------------

## (4) コンパイラの拡張機能のためのマクロ

inline	インライン関数
Inline	ファイルローカルなインライン関数
asm	インラインアセンブラ
Asm	インラインアセンブラ (最適化抑止)
throw()	例外を発生しない関数
NoReturn	リターンしない関数

## (5) エラーコード生成・分解マクロ

ERCD(mercd, sercd)	メインエラーコードmercdとサブエラーコードsercdから、エラーコードを生成するためのマクロ
MERCD(ercd)	エラーコードercdからメインエラーコードを抽出するためのマクロ
SERCD(ercd)	エラーコードercdからサブエラーコードを抽出するためのマクロ

## (6) アクセス許可パターン生成マクロ

TACP(domid)	domidで指定される保護ドメインに属する処理単位だけにアクセスを許可するアクセス許可パターン
-------------	---

## 5.4.3 カーネル共通定数

## (1) オブジェクト属性

TA_TPRI	0x01U	タスクの待ち行列をタスクの優先度順に
---------	-------	--------------------

## (2) 保護ドメインID

TDOM_SELF	0	自タスクの属する保護ドメイン
TDOM_KERNEL	-1	カーネルドメイン
TDOM_NONE	-2	無所属 (保護ドメインに属さない)

## (3) その他のカーネル共通定数

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

TIPM\_ENAALL 0 割込み優先度マスク全解除

#### 5.4.4 カーネルの機能毎の定数

#### 5.4.5 カーネルの機能毎のマクロ

##### (1) タスク管理機能

COUNT\_STK\_T(sz) サイズszのスタック領域を確保するために必要なSTK\_T型の配列の要素数を求めるマクロ  
 ROUND\_STK\_T(sz) 要素数COUNT\_STK\_T(sz)のSTK\_T型の配列のサイズ (szを, STK\_T型のサイズの倍数になるように大きい方に丸めた値)

##### (5) メモリプール管理機能

COUNT\_MPF\_T(blksz) 固定長メモリブロックのサイズがblkszの固定長メモリプール領域を確保するために, 固定長メモリブロック1つあたりに必要なMPF\_T型の配列の要素数を求めるマクロ  
 ROUND\_MPF\_T(blksz) 要素数COUNT\_MPF\_T(blksz)のMPF\_T型の配列のサイズ (blkszを, MPF\_T型のサイズの倍数になるように大きい方に丸めた値)

#### 5.5 構成マクロ

##### 5.5.1 TOPPERS共通構成マクロ

##### (1) 相対時間の範囲

TMAX\_RELTIM 相対時間に指定できる最大値

##### 5.5.2 カーネル共通構成マクロ

##### (1) サポートする機能

TOPPERS\_SUPPORT\_PROTECT 保護機能対応のカーネル  
 TOPPERS\_SUPPORT\_MULTI\_PRC マルチプロセッサ対応のカーネル  
 TOPPERS\_SUPPORT\_DYNAMIC\_CRE 動的生成対応のカーネル

##### (2) 優先度の範囲

TMIN\_TPRI タスク優先度の最小値 (= 1)  
 TMAX\_TPRI タスク優先度の最大値

##### 5.5.3 カーネルの機能毎の構成マクロ

##### (1) タスク管理機能

TMAX\_ACTCNT タスクの起動要求キューイング数の最大値  
 TNUM\_TSKID 登録できるタスクの数 (動的生成対応でないカーネルでは, 静的APIによって登録されたタスクの数に一致)

##### (2) タスク付属同期機能

TMAX\_WUPCNT タスクの起床要求キューイング数の最大値

## (3) タスク例外処理機能

TBIT\_TEXPTN      タスク例外要因のビット数 (TEXPTNの有効ビット数)

## (4) 同期・通信機能

## セマフォ

TMAX\_MAXSEM      セマフォの最大資源数の最大値

TNUM\_SEMID      登録できるセマフォの数 (動的生成対応でないカーネルでは, 静的APIによって登録されたセマフォの数に一致)

## イベントフラグ

TBIT\_FLGPTN      イベントフラグのビット数 (FLGPTNの有効ビット数)

TNUM\_FLGID      登録できるイベントフラグの数 (動的生成対応でないカーネルでは, 静的APIによって登録されたイベントフラグの数に一致)

## データキュー

TNUM\_DTQID      登録できるデータキューの数 (動的生成対応でないカーネルでは, 静的APIによって登録されたデータキューの数に一致)

## 優先度データキュー

TMIN\_DPRI      データ優先度の最小値 (= 1)

TMAX\_DPRI      データ優先度の最大値

TNUM\_PDQID      登録できる優先度データキューの数 (動的生成対応でないカーネルでは, 静的APIによって登録された優先度データキューの数に一致)

## メールボックス

TMIN\_MPRI      メッセージ優先度の最小値 (= 1)

TMAX\_MPRI      メッセージ優先度の最大値

TNUM\_MBXID      登録できるメールボックスの数 (動的生成対応でないカーネルでは, 静的APIによって登録されたメールボックスの数に一致)

## ミューテックス

TNUM\_MTXID      登録できるミューテックスの数 (動的生成対応でないカーネルでは, 静的APIによって登録されたミューテックスの数に一致)

## (5) メモリプール管理機能

## 固定長メモリプール

TNUM\_MPFID      登録できる固定長メモリプールの数 (動的生成対応でないカーネルでは, 静的APIによって登録された固定長メモリプールの数に一致)

(6) 時間管理機能

システム時刻管理

TIC\_NUME タイムティックの周期 (単位はミリ秒) の分子  
TIC\_DENO タイムティックの周期 (単位はミリ秒) の分母  
  
TOPPERS\_SUPPORT\_GET\_UTM get\_utmがサポートされている

周期ハンドラ

TNUM\_CYCID 登録できる周期ハンドラの数 (動的生成対応でないカーネルでは, 静的APIによって登録された周期ハンドラの数に一致)

アラームハンドラ

TNUM\_ALMID 登録できるアラームハンドラの数 (動的生成対応でないカーネルでは, 静的APIによって登録されたアラームハンドラの数に一致)

(7) システム状態管理機能

なし

(8) メモリオブジェクト管理機能

未完成

(9) 割り込み管理機能

TMIN\_INTPRI 割り込み優先度の最小値 (最高値)  
TMAX\_INTPRI 割り込み優先度の最大値 (最低値, = -1)  
  
TMIN\_ISRPRI 割り込みサービスルーチン優先度の最小値 (= 1)  
TMAX\_ISRPRI 割り込みサービスルーチン優先度の最大値  
  
TOPPERS\_SUPPORT\_DIS\_INT dis\_intがサポートされている  
TOPPERS\_SUPPORT\_ENA\_INT ena\_intがサポートされている

(10) CPU例外管理機能

なし

(11) 拡張サービスコール管理機能

TNUM\_FNCD 登録できる拡張サービスコールの数 (動的生成対応でないカーネルでは, 静的APIによって登録された拡張サービスコールの数に一致)

(12) システム構成管理機能

なし

5.6 エラーコード一覧

(1) メインエラーコード

E_SYS	-5	システムエラー
E_NOSPT	-9	未サポート機能
E_RSFN	-10	予約機能コード
E_RSATR	-11	予約属性
E_PAR	-17	パラメータエラー
E_ID	-18	不正ID番号
E_CTX	-25	コンテキストエラー
E_MACV	-26	メモリアクセス違反
E_OACV	-27	オブジェクトアクセス違反
E_ILUSE	-28	サービスコール不正使用
E_NOMEM	-33	メモリ不足
E_NOID	-34	ID番号不足
E_OBJ	-41	オブジェクト状態エラー
E_NOEXS	-42	オブジェクト未登録
E_QOVR	-43	キューイングオーバーフロー
E_RLWAI	-49	待ち禁止状態または待ち状態の強制解除
E_TMOUT	-50	ポーリング失敗またはタイムアウト
E_DLT	-51	待ちオブジェクトの削除または再初期化
E_CLS	-52	待ちオブジェクトの状態変化
E_WBLK	-57	ノンブロッキング受付け
E_BOVR	-58	バッファオーバーフロー

### 5.7 機能コード一覧

未完成（機能コードを割り当て直す）

	-0	-1	-2	-3
-0x01	予約	予約	予約	予約
-0x05				
-0x09				
-0x0d				
-0x11				
-0x15				
-0x19				
-0x1d				
-0x21				
-0x25				
-0x29				
-0x2d				
-0x31				
-0x35				
-0x39				
-0x3d				
-0x41				
-0x45				
-0x49				
-0x4d				
-0x51				
-0x55				
-0x59				
-0x5d				
-0x61				
-0x65				
-0x69				
-0x6d				

-0x71  
 -0x75  
 -0x79  
 -0x7d  
 -0x81  
 -0x85  
 -0x89  
 -0x8d  
 -0x91  
 -0x95  
 -0x99  
 -0x9d  
 -0xa1  
 -0xa5  
 -0xa9  
 -0xad  
 -0xb1  
 -0xb5  
 -0xb9  
 -0xbd  
 -0xc1  
 -0xc5  
 -0xc9  
 -0xcd  
 -0xd1  
 -0xd5  
 -0xd9  
 -0xdd  
 -0xe1  
 -0xe5  
 -0xe9  
 -0xed  
 -0xf1  
 -0xf5  
 -0xf9  
 -0xfd

5.8 カーネルオブジェクトに対するアクセスの種別

オブジェクトの種類	通常操作1	通常操作2	管理操作	参照操作
メモリオブジェクト	書込み	読出し	det_mem sac_mem	ref_mem prb_mem
タスク	act_tsk can_act get_pri wup_tsk can_wup	ter_tsk chg_pri rel_wai sus_tsk rsm_tsk dis_wai ena_wai ras_tex sta_ovr stp_ovr	del_tsk sac_tsk def_tex	ref_tsk ref_tex ref_ovr
セマフォ	sig_sem	wai_sem	del_sem	ref_sem

		pol_sem twai_sem	ini_sem sac_sem	
イベントフラグ	set_flg clr_flg	wai_flg pol_flg twai_flg	del_flg ini_flg sac_flg	ref_flg
データキュー	snd_dtq psnd_dtq tsnd_dtq fsnd_dtq	rcv_dtq prcv_dtq trcv_dtq	del_dtq ini_dtq sac_dtq	ref_dtq
優先度データキュー	snd_pdq psnd_pdq tsnd_pdq	rcv_pdq prcv_pdq trcv_pdq	del_pdq ini_pdq sac_pdq	ref_pdq
メールボックス	snd_mbx	rcv_mbx prcv_mbx trcv_mbx	del_mbx ini_mbx	ref_mbx
ミューテックス	loc_mtx ploc_mtx tloc_mtx	-	del_mtx ini_mtx sac_mtx	ref_mtx
固定長メモリプール	get_mpf pget_mpf tget_mpf	rel_mpf	del_mpf ini_mpf sac_mpf	ref_mpf
周期ハンドラ	sta_cyc	stp_cyc	del_cyc sac_cyc	ref_cyc
アラームハンドラ	sta_alm	stp_alm	del_alm sac_alm	ref_alm
割込みサービスルーチン	-	-	del_isr sac_isr	ref_isr
システム状態	rot_rdq dis_dsp ena_dsp get_tim get_utm	loc_cpu unl_cpu ena_int dis_int chg_ipm get_ipm ext_ker	acre_yyy def_inh def_exc def_svc def_ovr sac_sys	ref_sys ref_int ref_cfg ref_ver

すべての保護ドメインから呼び出すことができるサービスコール：

- ・自タスクへの操作 ( ext\_tsk , get\_inf , slp\_tsk , tslp\_tsk , dly\_tsk , dis\_tex , ena\_tex )
- ・タスク例外状態参照 ( sns\_tex )
- ・システム状態参照 ( get\_tid , get\_did , sns\_ctx , sns\_loc , sns\_dsp , sns\_dpn , sns\_ker )
- ・CPU例外状態参照 ( xsns\_dpn , xsns\_xpn )

カーネルドメインのみから呼び出すことができるサービスコール：

- ・非タスクコンテキスト専用のサービスコール



アクセス許可ベクタによるアクセス保護を行わないサービスコールは：

- ・ ミューテックスのロック解除 (unl\_mtx)

【補足説明】

xsns\_dpnとxsns\_xpnは、エラーコードを返さないために、すべての保護ドメインから呼び出すことができるサービスコールとしているが、カーネルドメイン以外から呼び出した場合には必ずtrueが返ることとしており、実質的にはカーネルドメインのみから呼び出すことができる。

unl\_mtxは、アクセス許可ベクタによるアクセス保護を行わないサービスコールとしているが、ミューテックスをロックしたタスク以外が呼び出すとE\_ILUSEエラーとなるため、実質的には対象ミューテックスの通常操作1としてアクセス保護されているとみなすことができる（ミューテックスのロック中にアクセス許可ベクタを変更した場合の振舞いは異なる）。

以上

## アプリケーションシステム

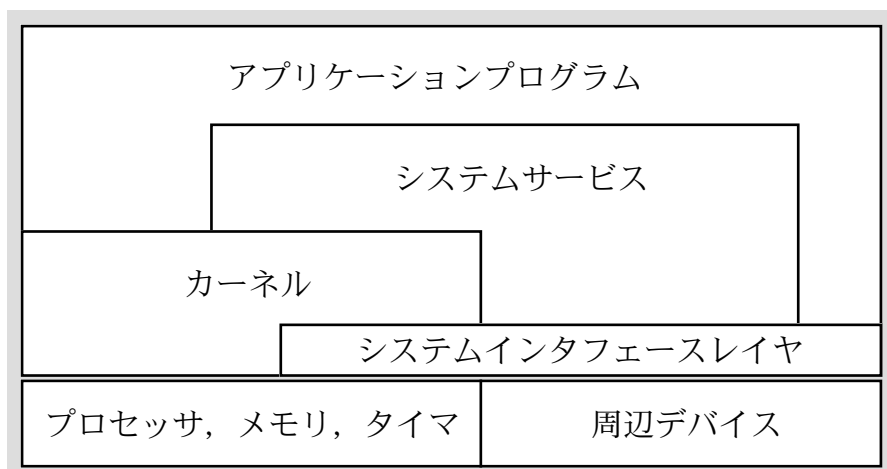


図2-1. 想定するソフトウェア構成

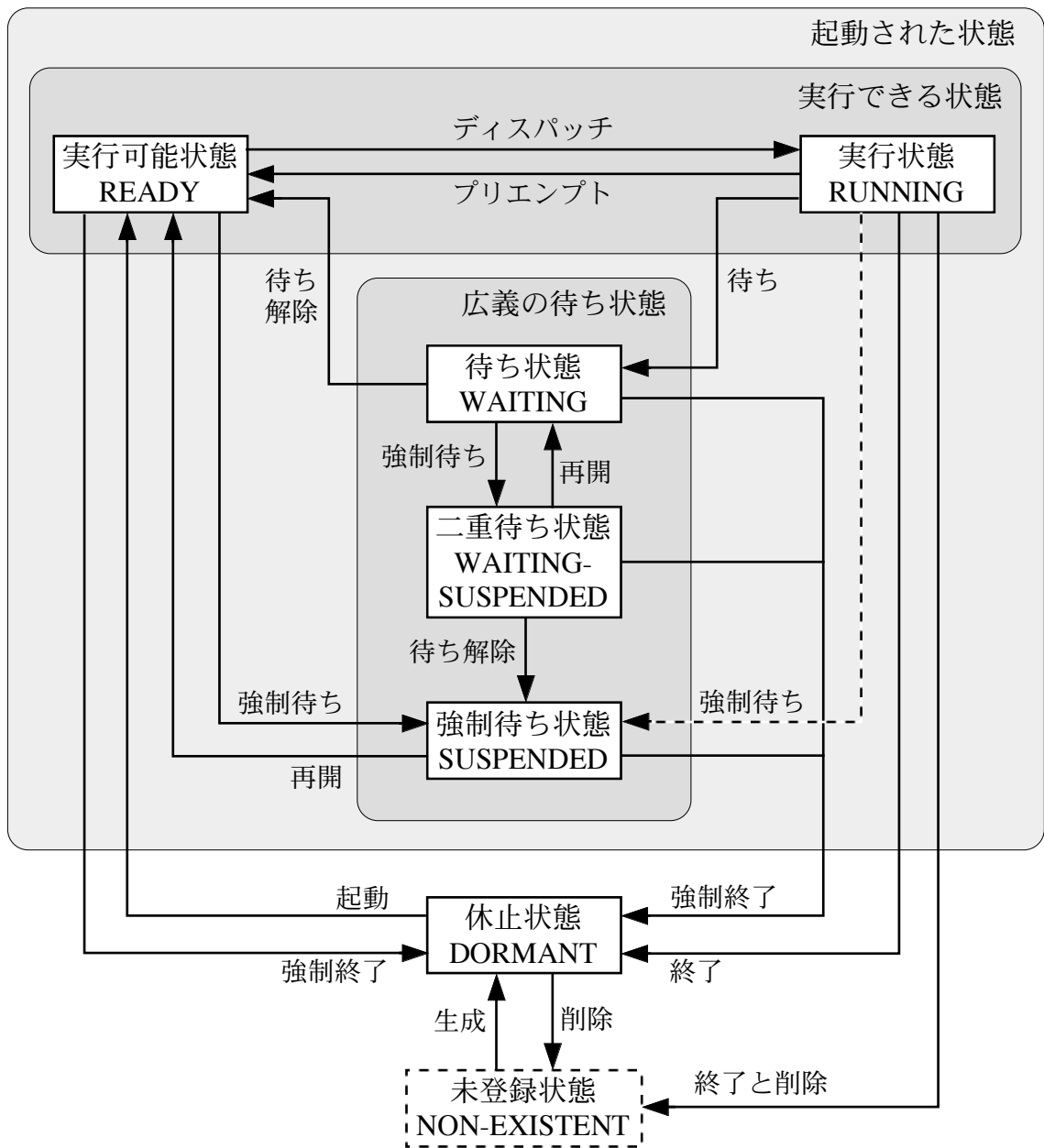


図2-2. タスクの状態遷移

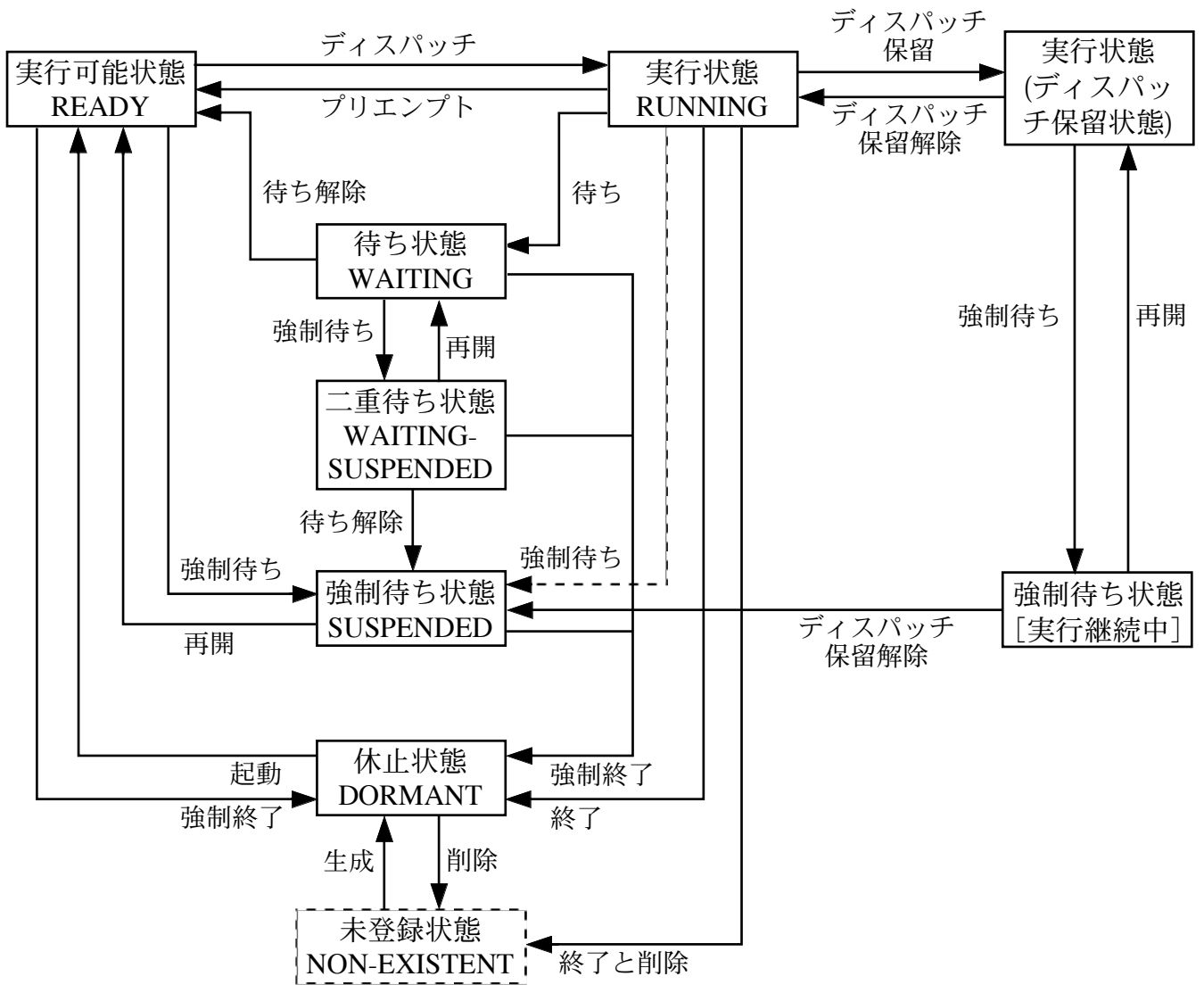


図2-3. 過渡的な状態も含めたタスクの状態遷移

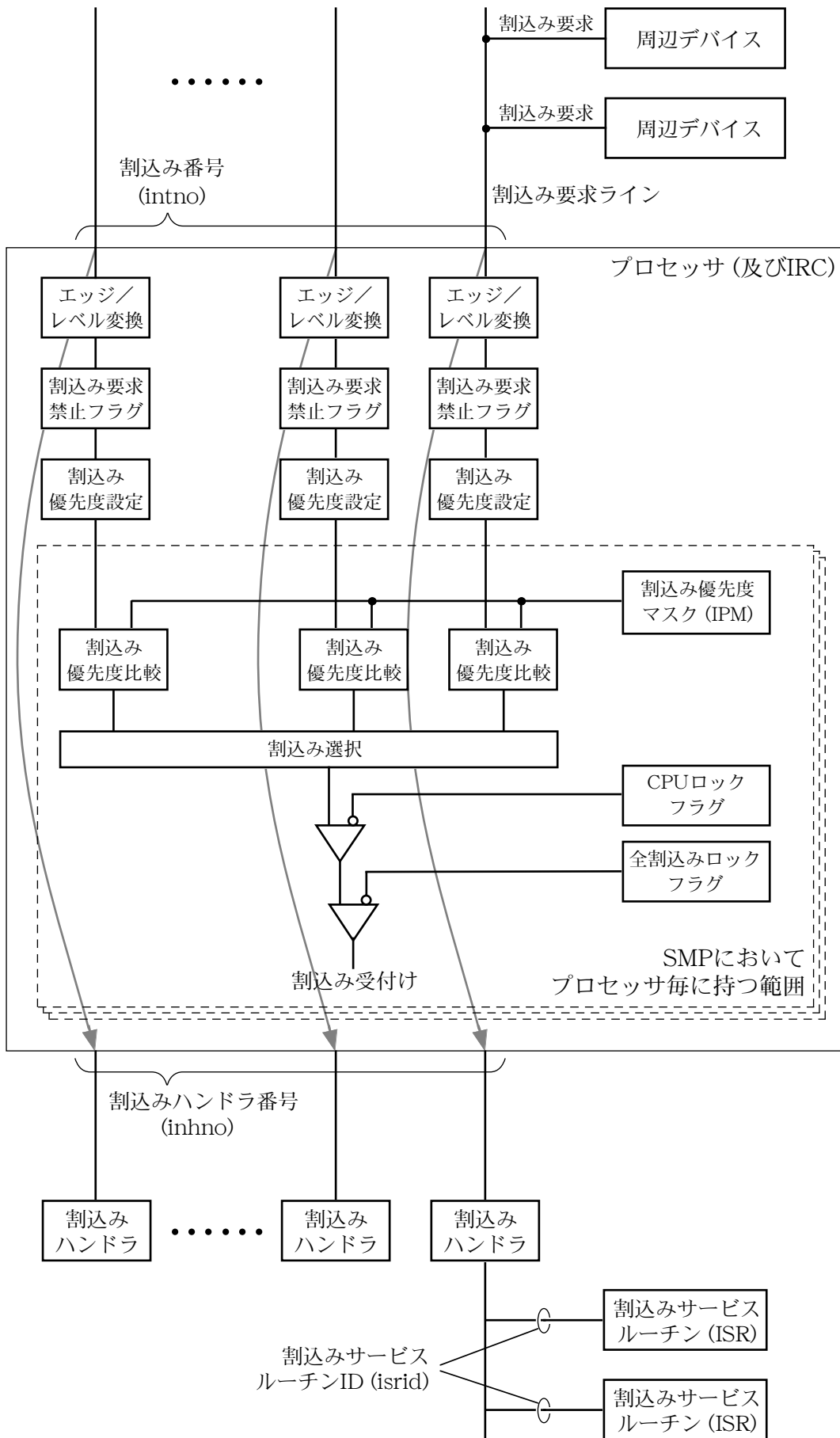


図2-4. TOPPERS標準割り込み処理モデルの概念図

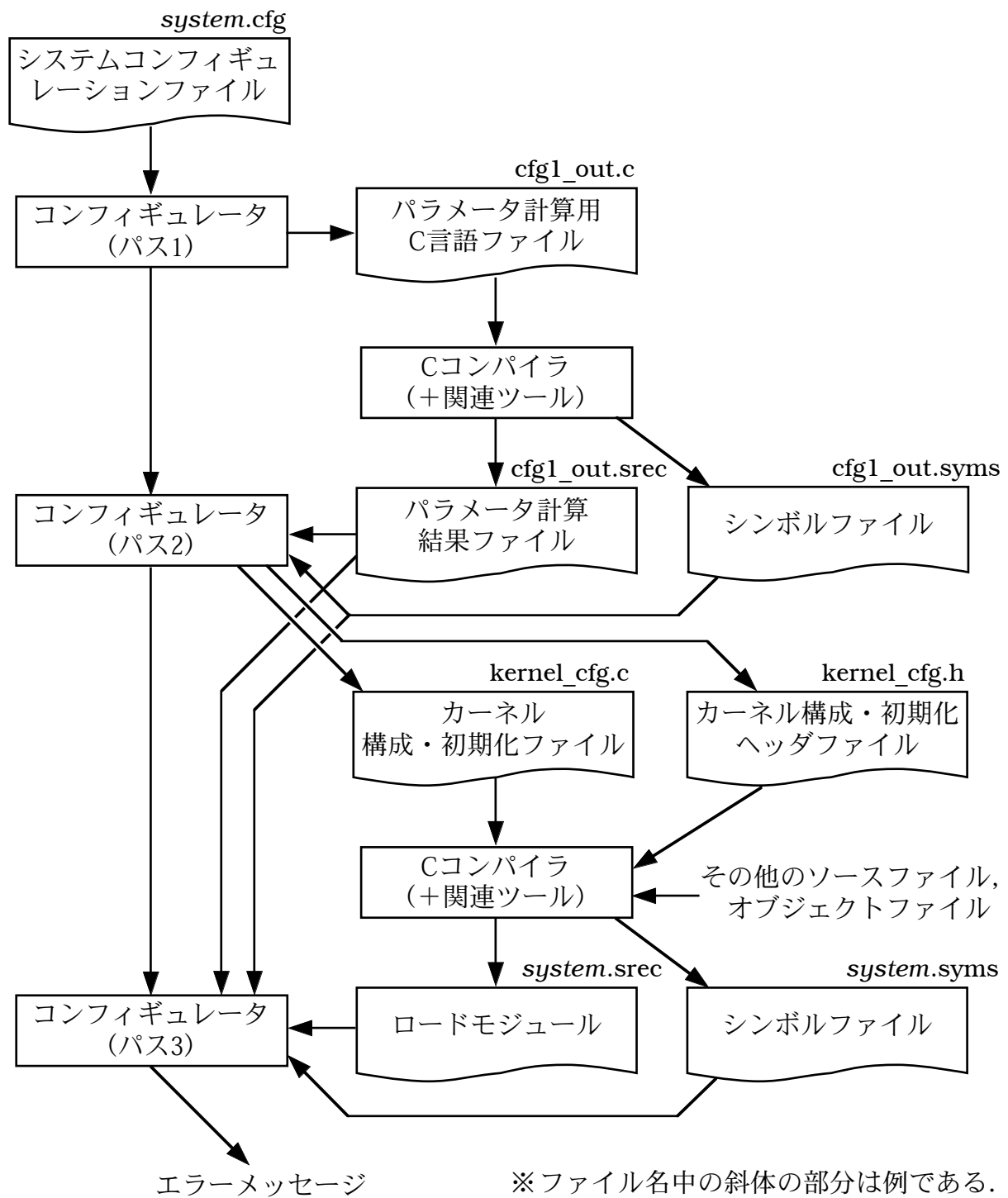


図2-5. コンフィギュレータの処理モデル