TOPPERS新世代カーネル統合仕様書

目次

仕様書で用いる記述項目と記号 . . . 6
タグの付与方法 . . . 8
1. TOPPERS新世代カーネルの概要 . . . 8
1.1. TOPPERS新世代カーネル仕様の位置付け . . . 8
1.2. TOPPERS新世代カーネル仕様の設計方針 . . . 9
1.3. TOPPERS/ASPカーネルの適用対象領域と仕様設計方針 . . . 9
1.4. TOPPERS/FMPカーネルの適用対象領域と仕様設計方針 . . . 10
1.5. TOPPERS/HRP2カーネルの適用対象領域と仕様設計方針 . . . 10
1.6. TOPPERS/SSPカーネルの適用対象領域と仕様設計方針 . . . 10
1.7. TOPPERS/ASP Safetyカーネルの適用対象領域と仕様設計方針 . . . 10
2. 主要な概念と共通定義 . . . 11
2.1. 仕様の位置付け . . . 11
2.1.1. カーネルの機能セット . . . 11
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 . . . 11
2.1.3. 想定するソフトウェア構成 . . . 12
2.1.4. 想定するハードウェア構成 . . . 12
2.1.5. 想定するプログラミング言語 . . . 12
2.2. APIの構成要素とコンベンション . . . 12
2.2.1. APIの構成要素 . . . 13
2.2.2. パラメータとリターンパラメータ . . . 13
2.2.3. 返値とエラーコード . . . 13
2.2.4. 機能コード . . . 14
2.2.5. ヘッダファイル . . . 14
2.3. 主な概念 . . . 14
2.3.1. オブジェクトと処理単位 . . . 14
2.3.2. サービスコールとパラメータ . . . 15
2.3.3. 保護機能 . . . 18
2.3.4. マルチプロセッサ対応 . . . 21
2.3.5. その他 . . . 22
2.4. 処理単位の種類と実行順序 . . . 23
2.4.1. 処理単位の種類 . . . 23
2.4.2. 処理単位の実行順序 . . . 23
2.4.3. カーネル処理の不可分性 . . . 24
2.4.4. 処理単位を実行するプロセッサ . . . 25
2.5. システム状態とコンテキスト . . . 25
2.5.1. カーネル動作状態と非動作状態 . . . 25

2.5.2. タスクコンテキストと非タスクコンテキスト . . . 26
2.5.3. カーネルの振舞いに影響を与える状態 . . . 26
2.5.4. 全割込みロック状態と全割込みロック解除状態 . . . 26
2.5.5. CPUロック状態とCPUロック解除状態 . . . 26
2.5.6. 割込み優先度マスク . . . 27
2.5.7. ディスパッチ禁止状態とディスパッチ許可状態 . . . 27
2.5.8. ディスパッチ保留状態 . . . 28
2.5.9. カーネル管理外の状態 . . . 28
2.5.10. 処理単位の開始・終了とシステム状態 . . . 28
2.6. タスクの状態遷移とスケジューリング規則 . . . 30
2.6.1. 基本的なタスク状態 . . . 30
2.6.2. タスクの状態遷移 . . . 31
2.6.3. タスクのスケジューリング規則 . . . 32
2.6.4. 待ち行列と待ち解除の順序 . . . 32
2.6.5. タスク例外処理マスク状態と待ち禁止状態 . . . 32
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 . . . 34
2.6.7. 制約タスク . . . 35
2.7. 割込み処理モデル . . . 35
2.7.1. 割込み処理の流れ . . . 36
2.7.2. 割込み優先度 . . . 36
2.7.3. 割込み要求ラインの属性 . . . 36
2.7.4. 割込みを受け付ける条件 . . . 37
2.7.5. 割込み番号と割込みハンドラ番号 . . . 37
2.7.6. マルチプロセッサにおける割込み処理 . . . 38
2.7.7. カーネル管理外の割込み . . . 39
2.7.8. カーネル管理外の割込みの設定方法 . . . 39
2.8. CPU例外処理モデル . . . 39
2.8.1. CPU例外処理の流れ . . . 40
2.8.2. CPU例外ハンドラから呼び出せるサービスコール . . . 40
2.8.3. エミュレートされたCPU例外ハンドラ . . . 41
2.8.4. カーネル管理外のCPU例外 . . . 41
2.9. システムの初期化と終了 . . . 42
2.9.1. システム初期化手順 . . . 42
2.9.2. システム終了手順 . . . 42
2.10. オブジェクトの登録とその解除 . . . 43
2.10.1. ID番号で識別するオブジェクト . . . 43
2.10.2. オブジェクト番号で識別するオブジェクト . . . 44
2.10.3. 識別番号を持たないオブジェクト . . . 45
2.10.4. オブジェクト生成に必要なメモリ領域 . . . 45
2.10.5. オブジェクトが属する保護ドメインの設定 . . . 45
2.10.6. オブジェクトが属するクラスの設定 . . . 46

2.10.7. オブジェクトの状態参照 . . . 46
2.11. オブジェクトのアクセス保護 . . . 46
2.11.1. オブジェクトのアクセス保護とアクセス違反の通知 . . . 46
2.11.2. メモリオブジェクトに対するアクセス許可ベクタの制限 . . . 47
2.11.3. デフォルトのアクセス許可ベクタ . . . 48
2.11.4. アクセス許可ベクタの設定 . . . 48
2.11.5. カーネルの管理領域のアクセス保護 . . . 49
2.11.6. ユーザタスクのユーザスタック領域 . . . 49
2.12. システムコンフィギュレーション手順 . . . 49
2.12.1. システムコンフィギュレーションファイル . . . 49
2.12.2. 静的APIの文法とパラメータ . . . 50
2.12.3. 保護ドメインの指定 . . . 51
2.12.4. クラスの指定 . . . 52
2.12.5. コンフィギュレータの処理モデル . . . 52
2.12.6. 静的APIのパラメータに関するエラー検出 . . . 53
2.12.7. オブジェクトのID番号の指定 . . . 54
2.13. TOPPERSネーミングコンベンション . . . 54
2.13.1. モジュール識別名 . . . 54
2.13.2. データ型名 . . . 54
2.14. 関数名 . . . 55
2.14.1. 変数名 . . . 55
2.14.2. 定数名 . . . 56
2.14.3. マクロ名 . . . 57
2.14.4. 静的API名 . . . 57
2.14.5. ファイル名 . . . 57
2.14.6. モジュール内部の名称の衝突回避 . . . 58
2.15. TOPPERS共通定義 . . . 58
2.15.1. TOPPERS共通ヘッダファイル . . . 58
2.15.2. TOPPERS共通データ型 . . . 58
2.15.3. TOPPERS共通定数 . . . 60
2.15.4. TOPPERS共通エラーコード . . . 62
2.15.5. TOPPERS共通マクロ . . . 63
2.15.6. TOPPERS共通構成マクロ . . . 65
2.15.7. カーネル共通定数 . . . 65
2.15.8. カーネル共通マクロ . . . 67
2.15.9. カーネル共通構成マクロ . . . 67
3. システムインタフェースレイヤAPI仕様 . . . 69
3.1. システムインタフェースレイヤの概要 . . . 69
3.2. SILヘッダファイル . . . 70
3.3. 全割込みロック状態の制御 . . . 70
3.4. SILスピンロック . . . 71

3.5. 微少時間待ち . . . 72
3.6. エンディアンの取得 . . . 72
3.7. メモリ空間アクセス関数 . . . 72
3.8. I/O空間アクセス関数 . . . 73
3.9. プロセッサIDの参照 . . . 74
4. カーネルAPI仕様 . . . 74
4.1. タスク管理機能 . . . 75
4.2. タスク付属同期機能 . . . 100
4.3. タスク例外処理機能 . . . 109
4.4. 同期・通信機能 . . . 117
4.4.1. セマフォ . . . 118
4.4.2. イベントフラグ . . . 128
4.4.3. データキュー . . . 140
4.4.4. 優先度データキュー . . . 154
4.4.5. メールボックス . . . 167
4.4.6. ミューテックス . . . 176
4.4.7. メッセージバッファ . . . 188
4.4.8. スピンロック . . . 201
4.5. メモリプール管理機能 . . . 210
4.5.1. 固定長メモリプール . . . 211
4.6. 時間管理機能 . . . 223
4.6.1. システム時刻管理 . . . 223
4.6.2. 周期ハンドラ . . . 226
4.6.3. アラームハンドラ . . . 238
4.6.4. オーバランハンドラ . . . 249
4.7. システム状態管理機能 . . . 256
4.8. メモリオブジェクト管理機能 . . . 270
4.9. 割込み管理機能 . . . 286
4.10. CPU例外管理機能 . . . 305
4.11. 拡張サービスコール管理機能 . . . 310
4.12. システム構成管理機能 . . . 314
5. リファレンス . . . 320
5.1. サービスコール一覧 . . . 320
5.2. 静的API一覧 . . . 326
5.3. データ型 . . . 329
5.3.1. TOPPERS共通データ型 . . . 329
5.3.2. カーネルの使用するデータ型 . . . 330
5.3.3. カーネルの使用するパケット形式 . . . 331
5.4. 定数とマクロ . . . 340
5.4.1. TOPPERS共通定数 . . . 340
5.4.2. TOPPERS共通マクロ . . . 342

5.4.3. カーネル共通定数 . . . 343
5.4.4. カーネル共通マクロ . . . 344
5.4.5. カーネルの機能毎の定数 . . . 344
5.4.6. カーネルの機能毎のマクロ . . . 347
5.5. 構成マクロ . . . 348
5.5.1. TOPPERS共通構成マクロ . . . 348
5.5.2. カーネル共通構成マクロ . . . 348
5.5.3. カーネルの機能毎の構成マクロ . . . 349
5.6. エラーコード一覧 . . . 353
5.7. 機能コード一覧 [NGKI4036] . . . 353
5.8. カーネルオブジェクトに対するアクセスの種別 . . . 356
5.9. ターゲット定義事項一覧 . . . 358
5.10. 省略名の元になった英語 . . . 362
5.10.1. サービスコールと静的APIの名称の中のxxxの元になった英語 . . . 362
5.10.2. サービスコールと静的APIの名称の中のyyy の元になった英語 . . . 364
5.10.3. サービスコールの名称の中のzの元になった英語 . . . 365
5.11. バージョン履歴 . . . 365

バージョン: Release 1.7.1
最終更新: 2015年5月30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，統合的に記述したものである．今後，この仕様に対して，大きい機能追加や仕様改変は行わず，これ以降は第3世代カーネル仕様として検討を行う計画である．ただし，仕様が未完成の部分（特に動的生成対応カーネルに関しては，仕様検討が不十分なところが多い）については，それを実装する時点で追加で決定していくこととする．

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) 2006-2015 by Embedded and Real-Time Systems Laboratory
Graduate School of Information Science, Nagoya Univ., JAPAN
Copyright (C) 2006-2015 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．

仕様書で用いる記述項目と記号

この仕様書では，以下の記述項目を用いる．

【補足説明】

仕様本体の記述に対する補足事項を説明する．

【～～カーネルにおける規定】

TOPPERS新世代カーネルに属する特定のカーネルにおける追加仕様を規定する．

【～～仕様との関係】

この仕様と，μITRON4.0仕様またはμITRON4.0/PX仕様との違いについて説明する．

【未決定事項】

この仕様書の現時点のバージョンでは，決定されずに残っている事項について記述する．

【仕様決定の理由】

仕様を決定するにあたって考慮した事項について説明する．

「第４章 カーネルAPI仕様」の章の各サービスコールおよび静的APIの仕様記述 においては，以下の記述項目を用いる．

□□□**API**□

□□□

□P□

□□□

ル仕様を策定することになった．

ただし，ITRON仕様が，各社が開発するリアルタイムカーネルを標準化すること を目的に，リアルタイムカーネルの「標準仕様」を規定することを目指してい るのに対して，TOPPERS新世代カーネル仕様は，TOPPERSプロジェクトにおいて 開発している一連のリアルタイムカーネルの「実装仕様」を記述するものであ り，ITRON仕様とは異なる目的・位置付けを持つものである．

1.2. 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. μITRON4.0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μITRON4.0仕様をベースとする．ただし，μITRON4.0仕様 の策定時以降の状況の変化を考慮し，μITRON4.0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．μITRON4.0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

   μITRON4.0仕様の策定時点と比べると，組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，μITRON4.0仕様においてオーバヘッ 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．

1.3. 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新世代カーネルの出発点となるリアルタイムカーネルである．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 う観点から，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，保護機能を持ったリアルタイムカーネルを適用す 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

□□□

2. 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．

【TOPPERS/ASPカーネルにおける規定】

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【ASPS0001】．ただし，動的生成機能拡張パッケージを用いると，動的生成対応カーネルの機能の一部がサポートされる【ASPS0002】．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルは，マルチプロセッサ対応カーネルであり，保護機能対応カーネル，動的生成対応カーネルではない【FMPS0001】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルは，保護機能対応カーネルであり，マルチプロセッサ対応カーネル，動的生成対応カーネルではない【HRPS0001】．ただし，動的生成機能拡張パッケージを用いると，動的生成対応カーネルの機能の一部がサポートされる【HRPS0009】．

【TOPPERS/SSPカーネルにおける規定】

SSP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【SSPS0001】．

【μITRON4.0仕様，μITRON4.0/PX仕様との関係】

μITRON4.0仕様は，カーネルオブジェクトの動的生成機能を持っているが，保護機能を持っておらず，マルチプロセッサにも対応していない．μITRON4.0/PX仕様は，μITRON4.0仕様に対して保護機能を追加するための仕様であり，カー ネルオブジェクトの動的生成機能と保護機能を持っているが，マルチプロセッ サには対応していない．

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．

□□□

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【NGKI0007】．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【NGKI0008】．

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【NGKI0009】．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【NGKI0010】．

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，それらの領域が参照されることはなく，別の目的に使用できる【NGKI0011】．

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【NGKI0012】．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【NGKI0013】．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【NGKI0014】．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（ERCD）と，エラーコードからメインエラーコードを取り出すマクロ（MERCD），サブエラーコードを取り出すマクロ（SERCD）が用意されている【NGKI0015】．

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（「TOPPERS共通エラーコード」の節を参照）【NGKI0016】．サービスコールおよびコールバックの機能説明中の「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，メインエラーコードとしてE_XXXXXが返ることを意味する．

□□□

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【NGKI0031】．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【NGKI0032】．

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．拡張サービスコールからサービスコールを呼び出した場合には，拡張サービスコールを呼び出したタスクが自タスクである．

カーネルには，静的APIにより，少なくとも1つのタスクを登録しなければならない．タスクが登録されていない場合には，コンフィギュレータがエラーを報告する【NGKI0033】．

【補足説明】

タスクが呼び出した拡張サービスコールが実行されている間は，「サービスコールを呼び出した処理単位」は拡張サービスコールであり，「自タスク」とは一致しない．そのため，保護機能対応カーネルにおいて，「サービスコールを呼び出した処理単位の属する保護ドメイン」と「自タスクの属する保護ドメイン」は，異なるものを指す．

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【NGKI0034】．

5. 割込みとCPU例外

プロセッサが実行中の処理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，外部割込みまたは単に割込み（interrupt）と呼ぶ．それに対して，プロセッサが実行中の処理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込み要求をプロセッサに伝える経路を遮断し，割込み要求が受け付けられるのを抑止することを，割込みのマスク（mask interrupt）または割込みの禁止（disable interrupt）という．マスクが解除された時点で，まだ割込み要求が保持されていれば，その時点で割込み要求を受け付ける【NGKI0035】．

マスクすることができない割込みを，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【NGKI0036】.

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，1から連続した正の値を用いるのを原則とする【NGKI0037】．優先度は，値が小さいほど優先度が高い（すなわち，先に実行または配送される）ものとする【NGKI0038】.

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【NGKI0039】．システム時刻は，タイムティック（time tick）を通知するための タイマ割込みが発生する毎に更新される【NGKI0040】.

イベントを発生させる時刻を指定する場合には，基準時刻（base time）からの 相対時間（relative time）によって指定する【NGKI0041】．基準時刻は，別に規定がない限りは，相対時間を指定するサービスコールを呼び出した時刻となる【NGKI0042】.

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【NGKI0043】．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【NGKI0044】．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【NGKI0045】.

イベントを発生させる時刻を相対時間で指定した場合，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【NGKI0046】．ただし，基準時刻を定めるサービスコールを呼び出した時に，タイムティックを通知するためのタイマ割込みがマスクされている場合（タイマ割込みより優先して実行される割込み処理が実行されている場合を含む）は，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ことは保証 されない【NGKI0047】.

イベントが発生する時刻を参照する場合には，基準時刻からの相対時間として返される【NGKI0048】．基準時刻は，相対時間を返すサービスコールを呼び出した時刻となる【NGKI0049】.

イベントが発生する時刻が相対時間で返された場合，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【NGKI0050】．ただし，相対時間を返すサービスコールを呼び出した時に，タイムティックを通知するためのタイマ割込みがマスクされている場合（タイマ割込みより優先して実行される割込み処理が実行されている場合を含む）は，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ことは保証されない【NGKI0051】.

【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回目以降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回目以降のタイムティックでイベントの処理が行われる.

【μITRON4.0仕様との関係】

相対時間（RELTIM型）とシステム時刻（SYSTIM型）の時間単位は，μITRON4.0仕様では実装定義としていたが，この仕様ではミリ秒と規定した．また，相対時間の解釈について，より厳密に規定した.

TMAX_RELTIM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.

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【NGKI0052】.

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【NGKI0053】．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【NGKI0054】．すなわち，タイムアウトの処理

□□□

2.3.3. □□□□

□□□

ンフィギュレータがエラーを報告する【NGKI0085】．

【補足説明】

ユーザドメインは，システムコンフィギュレーションファイル中にユーザドメインの囲みを記述することで，カーネルに登録する（「2.12.3 保護ドメインの指定」の節を参照）．ユーザドメインを動的に生成する機能は，現時点では用 意していない．

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 みなすこともできる．

【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，現時点ではサポートしない．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を制限することが できる保護ドメインである．

5. システムタスクとユーザタスク

カーネルドメインに属するタスクをシステムタスク（system task），ユーザド メインに属するタスクをユーザタスク（user task）と呼ぶ．

【補足説明】

特権モードで実行されるタスクをシステムタスク，非特権モードで実行されるタスクをユーザタスクと定義する方法もあるが，ユーザタスクであっても，サービスコールの実行中は特権モードで実行されるため，上記の定義とした．

μITRON4.0/PX仕様のシステムドメインに属するタスクは，システムタスクと呼 ぶことになる．

6. アクセス許可パターン

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つのユーザドメインに対応する【NGKI0086】．カーネルドメインには，すべてのアクセスが許可されているため，カーネルドメインに対応する ビットは用意されていない．

アクセス許可パターンは，符号無し32ビット整数に定義されるデータ型（ACPTN）で保持し，値が1のビットに対応するユーザドメインにアクセスが許可されていることを表す【NGKI0087】．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ユーザドメインの和集合（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ユーザドメインの積集合（intersection）を得ることができる．

アクセス許可パターンの指定に用いるために，指定したユーザドメインのみにアクセスを許可することを示すアクセス許可パターンを構成するマクロ（TACP）が用意されている【NGKI0088】．また，カーネルドメインのみにアクセスを許可することを示すアクセス許可パターンを表す定数（TACP_KERNEL）と，すべての保護ドメインにアクセスを許可することを示すアクセス許可パターンを表す 定数（TACP_SHARED）が用意されている【NGKI0089】．

7. アクセス許可ベクタ

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【NGKI0090】．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）と呼び，次のように定義されるデータ型（ACVCT）で保持する【NGKI0091】．

```
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;
```

【補足説明】

カーネルオブジェクトの種類毎のアクセスの種別の分類については，「5.8 カーネルオブジェクトに対するアクセスの種別」の節を参照すること．

【μITRON4.0/PX仕様との関係】

μITRON4.0/PX仕様では，アクセス許可ベクタを，1つまたは2つのアクセス許可パターンで構成することも許しているが，この仕様では4つで構成するものと決めている．

8. サービスコールの呼出し方法

保護機能対応カーネルでは，サービスコールは，ソフトウェア割込みによって呼び出すのが基本である．サービスコール呼出しを通常の方法で記述した場合，ソフトウェア割込みによって呼び出すコードが生成される【NGKI0092】．

一般に，ソフトウェア割込みによるサービスコール呼出しはオーバヘッドが大きい．そのため，カーネルドメインに属する処理単位からは，関数呼出しによってサービスコールを呼び出すことで，オーバヘッドを削減することができる．そこで，カーネルドメインに属する処理単位から関数呼出しによってサービスコールを呼び出せるように，以下の機能が用意されている．

カーネルドメインに属する処理単位が実行する関数のみを含んだソースファイルでは，カーネルヘッダファイル（kernel.h）をインクルードする前に，TOPPERS_SVC_CALLをマクロ定義することで，サービスコール呼出しを通常の方法で記述した場合に，関数呼出しによって呼び出すコードが生成される【NGKI0093】．

また，カーネルドメインに属する処理単位が実行する関数と，ユーザドメインに属する処理単位が実行する関数の両方を含んだソースファイルでは，関数呼出しによってサービスコールを呼び出すための名称を作るマクロ（SVC_CALL）を用いることで，関数呼出しによって呼び出すコードが生成される【NGKI0094】．例えば，act_tskを関数呼出しによって呼び出す場合には，次のように記述すればよい．

```
ercd = SVC_CALL(act_tsk)(tskid);
```

【補足説明】

拡張サービスコールを，関数呼出しによって呼び出す方法は用意されていない．カーネルドメインに属する処理単位が，関数呼出しによって，拡張サービスコールとして登録した関数を呼び出すことはできるが，その場合には，処理単位が呼び出した通常の関数であるとみなされ，拡張サービスコールであるとは扱われない．

9. ユーザドメインから行える処理に対する制限

ユーザドメインに属する処理単位が，システムの重要な処理に悪影響を及ぼすのを防ぐために，ユーザドメインから行える処理に対して制限を設ける機能が用意されている．具体的には，ユーザドメインに属する処理単位が，タスクのベース優先度を変更する際に，指定できるタスク優先度を制限することができる．

この機能を実現するために，各ユーザドメインは次の情報を持つ【NGKI0531】．

- 指定できる最高のタスク優先度

  なお，カーネルドメインに対しては，制限を設ける機能を用意していない．すなわち，カーネルドメインに属する処理単位は，すべてのタスク優先度を使う

ことができる【NGKI0532】．

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．クラスは，クラスIDと呼ぶID番号によって識別する【NGKI0095】．

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のが原則である【NGKI0096】．カーネルオブジェクトが属するクラスは，オブジェクトの登録時に決定し，登録後に変更することはできない【NGKI0097】．

【補足説明】

処理単位を実行するプロセッサを静的に決定する機能分散型のマルチプロセッサシステムでは，プロセッサ毎にクラスを設ける方法が典型的である．それに対して，対称型のマルチプロセッサシステムで，処理単位のマイグレーションを許す場合には，プロセッサ毎のクラスに加えて，どのプロセッサでも実行できるクラスを（システム中に1つまたは初期割付けプロセッサ毎に）設ける方法が典型的である．

［NGKI0096］の原則に関わらず，以下のオブジェクトはいずれのクラスにも属さない．

- オーバランハンドラ
- 拡張サービスコール
- グローバル初期化ルーチン
- グローバル終了処理ルーチン

マルチプロセッサ対応でないカーネルは，カーネルによって規定された1つのクラスのみをサポートしているとみなすこともできる．

2. プロセッサ

たかだか1つの処理単位のみを同時に実行できるハードウェアの単位を，プロセッサ（processor）と呼ぶ．プロセッサは，プロセッサIDと呼ぶID番号によって識別する【NGKI0098】．

＋ 複数のプロセッサを持つシステム構成をマルチプロセッサ（multiprocessor）と呼び，同時に複数の処理単位を実行することができる【NGKI0099】．

＋ システムの初期化時と終了時に特別な役割を果たすプロセッサを，マスタプロセッサ（master processor）と呼び，システムに1つ存在する【NGKI0100】．どのプロセッサをマスタプロセッサとするかは，ターゲット定義である【NGKI0101】．マスタプロセッサ以外のプロセッサを，スレーブプロセッサ（slave processor）と呼ぶ．なお，カーネル動作状態では，マスタプロセッサとスレーブプロセッサの振舞いに違いはない【NGKI0102】．

1. 処理単位の割付けとマイグレーション

処理単位は，後述のマイグレーションが発生しない限りは，いずれか1つのプロセッサに割り付けられて実行される【NGKI0103】．処理単位を実行するプロセッサを，割付けプロセッサと呼ぶ．また，処理単位が登録時に割り付けられるプロセッサを，初期割付けプロセッサと呼ぶ．

処理単位によっては，処理単位の登録後に，割付けプロセッサを変更することが可能である【NGKI0104】．処理単位の登録後に割付けプロセッサを変更することを，処理単位のマイグレーション（migration）と呼ぶ．

割付けプロセッサを変更できる処理単位に対しては，処理単位を割り付けることができるプロセッサ（これを，割付け可能プロセッサと呼ぶ）を制限することができる【NGKI0105】．

2. クラスの持つ属性とカーネルオブジェクト

□□□

3. 標準のセクション

   コンパイラに特別な指定をしない場合に出力するセクションを，標準のセクション（standard sections）と呼ぶ．コンパイラが出力しないセクションの中で，ターゲット定義のものを，標準のセクションと扱う場合もある【NGKI0113】．

4. 保護ドメイン毎の標準セクション

保護機能対応カーネルにおいては，保護ドメイン毎に，標準のセクションを配置するためのセクションが登録される【NGKI0114】．また，無所属の標準のセクションを配置するためのセクションが登録される【NGKI0115】．これらのセクションを，保護ドメイン毎の標準セクションと呼ぶ（standards for each protection domain）．保護ドメイン毎の標準セクションのセクション名は，ターゲット定義で別に規定がない限りは，標準のセクション名と保護ドメイン名（カーネルドメインの場合は"kernel"，無所属の場合は"shared"）を"_"でつないだものとする【NGKI0116】．例えば，カーネルドメインの".text"セクションのセクション名は，".text_kernel" とする．

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【NGKI0117】．

(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
(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をサポートしていない【ASPS0003】．ただし，オーバランハンドラ機能拡張パッケージを用いると，オーバランハンドラ機能を追加することができる【ASPS0004】．

【TOPPERS/FMPカーネルにおける規定】

FMP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をサポートしていない【FMPS0002】．

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは，タスク例外処理ルーチン，タイムイベントハンドラ，拡張サービスコールをサポートしていない【SSPS0002】．

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【NGKI0118】．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【NGKI0119】．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【NGKI0120】．

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【NGKI0121】．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【NGKI0122】．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【NGKI0123】．

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【NGKI0124】．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【NGKI0125】．

拡張サービスコールの優先順位は，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【NGKI0126】．

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【NGKI0127】．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【NGKI0128】．

マルチプロセッサ対応カーネルでは，初期化ルーチンには，クラスに属さないグローバル初期化ルーチンと，クラスに属するローカル初期化ルーチンがある【NGKI0129】．グローバル初期化ルーチンがマスタプロセッサで実行された後に，各プロセッサでローカル初期化ルーチンが実行される【NGKI0130】．また，終了処理ルーチンには，クラスに属さないグローバル終了処理ルーチンと，クラスに属するローカル終了処理ルーチンがある【NGKI0131】．ローカル終了処理ルーチンが各プロセッサで実行された後に，マスタプロセッサでグローバル終了処理ルーチンが実行される【NGKI0132】．

【仕様決定の理由】

終了処理ルーチンを，登録する静的APIを記述したのと逆順で実行するのは，終了処理は初期化の逆の順序で行うのがよいためである（システムコンフィギュレーションファイルを分割すると，終了処理ルーチンを登録する静的APIだけ逆順に記述するのは難しい）．

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例外ハンドラの入口処理と出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場合はあるが，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うのが原則である【NGKI0133】．これを，カーネル処理の不可分性という．

ただし，マルチプロセッサ対応カーネルにおいては，カーネル処理が実行されているプロセッサ以外のプロセッサから，カーネル処理の途中の状態が観測できる場合がある．具体的には，1つのサービスコールにより複数のオブジェクトの状態が変化する場合に，一部のオブジェクトの状態のみが変化し，残りのオブジェクトの状態が変化していない過渡的な状態が観測できる場合がある【NGKI0134】．

【補足説明】

マルチプロセッサ対応でないカーネルでは，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．

マルチプロセッサ対応カーネルでは，上のことは，1つのプロセッサ内では成り立つが，他のプロセッサに割り付けられたタスクに対しては成り立たない．例えば，プロセッサ1で低優先度のタスクAが実行されている時に，他のプロセッサ2で実行されているタスクが発行したサービスコールにより，プロセッサ1に割り付けられた中優先度のタスク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Cが待ち解除される前に，タスクBへディスパッチされる場合がある．

2.4.4. 処理単位を実行するプロセッサ

マルチプロセッサ対応カーネルでは，処理単位を実行するプロセッサ（割付け プロセッサ）は，その処理単位が属するクラスの初期割付けプロセッサと割付 け可能プロセッサから，次のように決まる．

タスク，周期ハンドラ，アラームハンドラは，登録時に，属するクラスの初期割付けプロセッサに割り付けられる【NGKI0135】．また，割付けプロセッサを変更するサービスコール（mact_tsk/imact_tsk，mig_tsk，msta_cyc，msta_alm/imsta_alm）によって，割付けプロセッサを，クラスの割付け可能プロセッサのいずれかに変更することができる【NGKI0136】．

割込みハンドラ，CPU例外ハンドラ，ローカル初期化ルーチン，ローカル終了処理ルーチンは，属するクラスの初期割付けプロセッサで実行される【NGKI0137】．クラスの割付け可能プロセッサの情報は用いられない．

割込みサービスルーチンは，属するクラスの割付け可能プロセッサのいずれか（オプション設定によりすべて）で実行される【NGKI0138】．クラスの初期割 付けプロセッサの情報は用いられない．

以上を整理すると，次の表の通りとなる．この表の中で，「○」はその情報が 使用されることを，「ー」はその情報が使用されないことを示す．

	初期割付けプロセッサ	割付け可能プロセッサ
タスク（タスク例外処理ルーチンを含む）	○	○
割込みハンドラ	○	-
割込みサービスルーチン	-	○
周期ハンドラ	○	○
アラームハンドラ	○	○
CPU例外ハンドラ	○	-
ローカル初期化ルーチン	○	-
ローカル終了処理ルーチン	○	-

オーバランハンドラ，拡張サービスコール，グローバル初期化ルーチン，グローバル終了処理ルーチンは，いずれのクラスにも属さない【NGKI0139】．オーバランハンドラは，オーバランを起こしたタスクの割付けプロセッサによって実行される【NGKI0140】．拡張サービスコールは，それを呼び出した処理単位の割付けプロセッサによって実行される【NGKI0141】．グローバル初期化ルーチンとグローバル終了処理ルーチンは，マスタプロセッサによって実行される【NGKI0142】．

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，（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態かカーネル非動作状態のいずれかの状態を取る【NGKI0143】．

カーネル非動作状態では，原則として，NMIを除くすべての割込みがマスクされる【NGKI0144】．

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とカーネル非動作状態を参照するサービスコール（sns_ker）のみを呼び出すことができる【NGKI0145】．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【NGKI0146】．

マルチプロセッサ対応カーネルでは，プロセッサ毎に，カーネル動作状態かカー ネル非動作状態のいずれかの状態を取る【NGKI0147】．

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【NGKI0148】．

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【NGKI0149】．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【NGKI0150】．

割込みハンドラ（割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【NGKI0151】．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【NGKI0152】．

タスクコンテキストで実行される処理単位は，別に規定がない限り，タスクのスタック領域を用いて実行される【NGKI0153】．非タスクコンテキストで実行される処理単位は，別に規定がない限り，非タスクコンテキスト用スタック領域を用いて実行される【NGKI0154】．

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【NGKI0155】．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【NGKI0156】．いずれも，呼び出した場合にはE_CTXエラーとなる【NGKI0157】．

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【NGKI0158】．

- 全割込みロックフラグ（全割込みロック状態と全割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）
- 割込み優先度マスク（割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【NGKI0159】．

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【NGKI0160】．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とカーネル非動作状態を参照するサービスコール（sns_ker），カーネルを終了するサービスコール（ext_ker）のみを呼び出すことができる【NGKI0161】．全割込みロック状態で，その他のサービスコール（拡張サービスコールを含む）を呼び出した場合の動作は，保証されない【NGKI0162】．また，全割込みロック状態で，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【NGKI0164】．

マルチプロセッサ対応カーネルでは，プロセッサ毎に，全割込みロックフラグを持つ【NGKI0165】．すなわち，プロセッサ毎に，全割込みロック状態か全割込みロック解除状態のいずれかの状態を取る．

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（「2.7.7 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【NGKI0166】．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．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，ディスパッチが保留される【NGKI0167】．

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【NGKI0168】．

- システムインタフェースレイヤのAPI

- loc_cpu/iloc_cpu，unl_cpu/iunl_cpu
- unl_spn/iunl_spn（マルチプロセッサ対応カーネルのみ）
- dis_int，ena_int
- sns_yyy
- xsns_yyy（CPU例外ハンドラからのみ）
- get_utm
- ext_tsk，ext_ker
- prb_mem（保護機能対応カーネルのみ）
- cal_svc（保護機能対応カーネルのみ）

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【NGKI0169】．

マルチプロセッサ対応カーネルでは，プロセッサ毎に，CPUロックフラグを持つ【NGKI0170】．すなわち，プロセッサ毎に，CPUロック状態かCPUロック解除状態のいずれかの状態を取る．

【補足説明】

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．

マルチプロセッサ対応カーネルにおいて，あるプロセッサがCPUロック状態にある間は，そのプロセッサにおいてのみ，すべてのカーネル管理の割込みがマスクされ，ディスパッチが保留される．それに対して他のプロセッサにおいては，割込みはマスクされず，ディスパッチも起こるため，CPUロック状態を使って他のプロセッサで実行される処理単位との排他制御を実現することはできない．

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【NGKI0171】．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【NGKI0172】．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【NGKI0173】．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【NGKI0175】．

マルチプロセッサ対応カーネルでは，プロセッサ毎に，割込み優先度マスクを持つ【NGKI0176】．

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【NGKI0177】．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【NGKI0179】．

マルチプロセッサ対応カーネルでは，プロセッサ毎に，ディスパッチ禁止フラグを持つ【NGKI0180】．すなわち，プロセッサ毎に，ディスパッチ禁止状態かディスパッチ許可状態のいずれかの状態を取る．

【補足説明】

□□□

	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
【割込みサービスルーチン】【NGKI0186】			
【タイムイベントハンドラ】【NGKI0187】			
実行開始条件	解除	自優先度より低い	任意
実行開始時処理	そのまま	自優先度に(*2)	そのまま
リターン前	原則解除(*1)	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
【CPU例外ハンドラ】【NGKI0188】			
実行開始条件	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
リターン前	原則元に(*1)	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
【拡張サービスコール】【NGKI0189】			
実行開始条件	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
リターン前	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

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．

「自優先度に(*2)」とは，割込みハンドラと割込みサービスルーチンの場合にはそれを要求した割込みの割込み優先度，周期ハンドラとアラームハンドラの場合にはタイマ割込みの割込み優先度，オーバランハンドラの場合にはオーバ ランタイマ割込みの割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ディスパッチ禁止フラグを元に戻す処理(*4)は，タスクにディスパッチ禁止フラグの変更を許可している場合にのみ行われる【NGKI0529】．カーネルは，ディスパッチ禁止フラグの元の状態をユーザスタック上に保存する【NGKI0530】．アプリケーションがユーザスタック上に保存されたディスパッチ禁止フラグの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は，書き換えた後のディスパッチ禁止フラグの状態に変更される（すなわち，元に戻され るとは限らない）【NGKI0190】．

また，保護機能対応カーネルでは，タスクにディスパッチ禁止フラグの変更を許可していない場合で，タスク例外処理ルーチン中で拡張サービスコールを用いてディスパッチ禁止フラグを変更した場合，カーネルは元の状態に戻さない【NGKI0191】．このことから，タスク例外処理ルーチンからの終了前に，ディスパッチ禁止フラグを元の状態に戻すのは，アプリケーションの責任とする【NGKI0192】．

【補足説明】

マルチプロセッサ対応カーネルにおいて，タスクがタスク例外処理ルーチンを実行中にマイグレーションされた場合，マイグレーション先のプロセッサにおいて，割込み優先度マスクとディスパッチ禁止フラグが元に戻される．

【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にディスパッチ禁止フラグを元に戻す処理(*4)が，タスクにディスパッチ禁止フ

□□□

□□□

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【NGKI0203】．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【NGKI0204】．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【NGKI0205】．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，サービスコ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【NGKI0206】．

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【NGKI0207】．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【NGKI0208】．

マルチプロセッサ対応カーネルでは，プロセッサ毎に，上記のスケジューリング規則を適用して，タスクスケジューリングを行う【NGKI0209】．すなわち，プロセッサがディスパッチ保留状態でない限りは，そのプロセッサに割り付けられた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．そのため，実行状態のタスクは，プロセッサ毎に存在する．

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．また，タスクが同期・通信オブジェクトの待ち行列につながれている場合に，そのオブジェクトを，タスクの待ちオブジェクトと呼ぶ．

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【NGKI0210】．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【NGKI0211】．

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【NGKI0212】．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【NGKI0213】．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【NGKI0214】．

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【NGKI0215】．

(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【NGKI0216】．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【NGKI0217】．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【NGKI0218】．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【NGKI0219】．

2.6.5. タスク例外処理マスク状態と待ち禁止状態

保護機能対応カーネルにおいて，ユーザタスクについては特権モードで実行している間（特権モードを実行している間に，実行可能状態や広義の待ち状態になっている場合を含む．また，サービスコールを呼び出して，実行可能状態や広義の待ち状態になっている場合も含む．タスクの実行開始前は含まない），システムタスクについては拡張サービスコールを実行している間（拡張サービスコールを実行している間に，実行可能状態や広義の待ち状態になっている場合を含む）は，タスク例外処理ルーチンの実行は開始されない【NGKI0220】．これらの状態を，タスク例外処理マスク状態と呼ぶ．

□□□

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【NGKI0226】．

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【NGKI0227】．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【NGKI0228】．

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【NGKI0229】．

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【NGKI0230】．

タスクが強制待ち状態［実行継続中］である時の扱いは次の通りである．

1. プロセッサを占有して実行を継続する．

   強制待ち状態［実行継続中］のタスクは，プロセッサを占有して，そのまま継続して実行される【NGKI0231】．

2. 実行状態のタスクに関する情報を参照するサービスコールでは，実行状態であるものと扱う．

   実行状態のタスクに関する情報を参照するサービスコール（get_tid／iget_tid，get_did，sns_tex）では，強制待ち状態［実行継続中］のタスクが，それを実行するプロセッサにおいて実行状態のタスクであるものと扱う．具体的には，強制待ち状態［実行継続中］のタスクが実行されている時にget_tid／iget_tidを発行すると，そのタスクのID番号を参照する【NGKI0232】．また，get_didを発行するとそのタスクが属する保護ドメインのID番号を，sns_texを発行するとそのタスクのタスク例外処理禁止フラグを参照する【NGKI0233】．

3. その他のサービスコールでは，強制待ち状態であるものと扱う．

   その他のサービスコールでは，強制待ち状態［実行継続中］のタスクは，強制待ち状態であるものと扱う【NGKI0234】．

   【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【ASPS0008】．

   【TOPPERS/FMPカーネルにおける規定】

   FMP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を，他のプロセッサから呼び出すことができるため，タスクが強制待ち状態［実行継続中］になる場合がある【FMPS0005】．

   【TOPPERS/HRP2カーネルにおける規定】

   HRP2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【HRPS0004】．

   【TOPPERS/SSPカーネルにおける規定】

   SSPカーネルでは，タスクが広義の待ち状態になることはないため，タスクが強制待ち状態［実行継続中］になることもない【SSPS0005】．

   【補足説明】

   この仕様では，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．そのため，実行状態と休止状態の間の過渡的な状態は存在しない．

2.6.7. 制約タスク

制約タスク（restricted task）は，複数のタスクでスタック領域を共有することによるメモリ使用量の削減を目的に，通常のタスクに対して，広義の待ち状態を持たないなどの機能制限を加えたものである．具体的には，制約タスクには以下の機能制限がある．

1. 広義の待ち状態に入ることができない【NGKI0235】．
2. サービスコールによりベース優先度を変更することができない【NGKI0236】．
3. 対象優先度の中の先頭のタスクが制約タスクである場合には，タスクの優先順位の回転（rot_rdq/irot_rdq）を行うことができない【NGKI0237】．
4. マルチプロセッサ対応カーネルでは，割付けプロセッサを変更することができない【NGKI0238】．

制約タスクに対して，機能制限により使用できなくなったサービスコールを呼び出した場合には，E_NOSPTエラーとなる【NGKI0239】．E_NOSPTエラーが返ることに依存している場合を除いては，制約タスクを通常のタスクに置き換えることができる【NGKI0240】．

【未決定事項】

現状では，制約タスクの優先度を変更するサービスコールは設けていないが，制約タスクが，自タスクの優先度を，起動時優先度（SSPカーネルにおいては，実行時優先度）と同じかそれよりも高い値に変更することは許してもよい．ただし，優先度の変更後は，同じ優先度内で最高優先順位としなければならないため，chg_priとは振舞いが異なることになる．自タスクの優先度を起動時優先度と同じかそれよりも高い値に変更するサービスコールを設けるかどうかは，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，制約タスクをサポートしていない【ASPS0009】．ただし，制約タスク拡張パッケージを用いると，制約タスクの機能を追加することができる【ASPS0010】．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，制約タスクをサポートしていない【FMPS0006】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，制約タスクをサポートしていない【HRPS0005】．

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは，制約タスクのみをサポートする【SSPS0006】．そのため，すべてのタスクと非タスクコンテキストがスタック領域を共有することができ，すべての処理単位で同一のスタック領域を使用している【SSPS0007】．このスタック領域を，共有スタック領域と呼ぶ．

【μITRON4.0仕様との関係】

制約タスクは，μITRON4.0仕様の自動車制御プロファイルで導入された機能である．この仕様における制約タスクは，μITRON4.0仕様の制約タスクよりも機能制限が少なくなっている．

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【NGKI0241】．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（プロセッサおよび割込みコントローラ）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．

【μITRON4.0仕様との関係】

割込み処理モデルは，μITRON4.0仕様から大幅に拡張している．

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．デバイスから割込みコントローラ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【NGKI0242】．受け付けた割込み要求が，カーネル管理の割込みである場合には，カーネル内の割込みハンドラの入口処理（割込み入口処理）を経由して，カーネル内の割込みハンドラを実行する【NGKI0243】．

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【NGKI0244】．割込みサービスルーチンは，プロセッサの割込みアーキテクチャや割込みコントローラに依存せず，割込みを要求したデバイスのみに依存して記述するのが原則である【NGKI0245】．1つの割込み要求ラインに対して複数の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【NGKI0246】．

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【NGKI0247】．

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により，タイムイベントの処理が行われる．具体的には，タイムアウト処理等が行われることに加えて，アプリケーションが登録したタイムイベントハンドラが呼び出される【NGKI0248】．

なお，受け付けた割込み要求に対して，割込みサービスルーチンも割込みハンドラも登録していない場合の振舞いは，ターゲット定義である【NGKI0249】．

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【NGKI0250】．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【NGKI0251】．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）【NGKI0252】．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，割込み要求を受け付ける前の値に戻す【NGKI0253】．

これらのことから，他の方法で割込みをマスクしていない限り，ある割込み要求の処理中は，それと同じかそれより低い割込み優先度を持つ割込み要求は受け付けられず，それより高い割込み優先度を持つ割込み要求は受け付けられることになる．つまり，割込み優先度は，多重割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込み要求の中で，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【NGKI0254】．

割込み優先度は，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，［NGKI0037］の原則には従わず，-1から連続した負の値を用いる【NGKI0255】．

割込み優先度の段階数は，ターゲット定義である【NGKI0256】．プロセッサが割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ヘッドを生じる場合には，ターゲット定義で，割込み優先度の段階数を1にする（すなわち，多重割込みを許さない）場合がある．

【仕様決定の理由】

割込み優先度に-1から連続した負の値を用いるのは，割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，いずれの割込みもマスクしない割込み優先度マスクの値を0にできるためである．

2.7.3. 割込み要求ラインの属性

各割込み要求ラインは，以下の属性を持つ．なお，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ている場合，それらの割込み要求は同一の属性を持つ【NGKI0257】．それらの割込み要求に別々の属性を設定することはできない．

1. 割込み要求禁止フラグ

   割込み要求ライン毎に，割込みをマスクするための割込み要求禁止フラグを持つ【NGKI0258】．割込み要求禁止フラグをセットすると，その割込み要求ラインによって伝えられる割込み要求はマスクされる【NGKI0259】．

□□□

□□□

2.7.7. 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（これを，TMIN_INTPRIと書く）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【NGKI0287】．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【NGKI0288】．

TMIN_INTPRIよりも高い割込み優先度を持ち，カーネル内でマスクしない割込みを，カーネル管理外の割込みと呼ぶ．また，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．NMIは，カーネル管理外の割込みとして扱う．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【NGKI0289】．

それに対して，TMIN_INTPRIと同じかそれよりも低い割込み優先度を持つ割込みをカーネル管理の割込み，カーネル管理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は，カーネル内の割込み入口処理を経由せずに実行するのが基本である【NGKI0290】．ただし，すべての割込みで同じ番地に分岐するプロセッサでは，カーネル内の割込み入口処理を全く経由せずにカーネル管理外の割込みハンドラを実行することができず，入口処理の一部分を経由してカーネル管理外の割込みハンドラが実行されることになる【NGKI0291】．

カーネル管理外の割込みハンドラが実行開始される時のシステム状態とコンテキスト，割込みハンドラの終了時に行われる処理，割込みハンドラの記述方法は，ターゲット定義である【NGKI0292】．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，ext_ker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【NGKI0293】．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【NGKI0294】．

2.7.8. カーネル管理外の割込みの設定方法

カーネル管理外の割込みの設定方法は，ターゲット定義で，次の3つの方法のいずれかが採用される【NGKI0295】．

1.

    1. NMI以外にカーネル管理外の割込みを設けない
    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【NGKI0296】．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と，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【NGKI0297】．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【NGKI0298】．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【NGKI0299】．

2. 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，カーネル管理外とする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【NGKI0300】．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【NGKI0301】．

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【NGKI0302】．また，カーネル管理外の割込みに対して，割込みサービスルーチンを登録することはできない【NGKI0303】．

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【NGKI0304】．プロセッサがCPU例外の発生を検出すると，カーネル内のCPU例外ハンドラの入口処理（CPU例外入口処理）を経由して，発生した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【NGKI0305】．

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番号付けを基本として，ターゲット定義で付与される【NGKI0306】．そのため，1から連続した正の値であるとは限らない．

マルチプロセッサ対応カーネルでは，異なるプロセッサで発生するCPU例外は，異なるCPU例外であると扱う【NGKI0307】．すなわち，同じ種類のCPU例外であっても，異なるプロセッサのCPU例外には異なるCPU例外ハンドラ番号を付与し，プロセッサ毎にCPU例外ハンドラを登録する．CPU例外ハンドラが属するクラスの初期割付けプロセッサは，CPU例外が発生するプロセッサと一致していなければならない【NGKI0308】．

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【NGKI0309】．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【NGKI0310】．

(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（リカバリ処理を行うタスクを最高優先度とし，タスクの起動または待ち解除後に優先順位を回転させることで，リカバリ処理を行える可能性があるが，CPU例外を起こしたタスクが制約タスクの場合には適用できないなど，推奨できる方法ではない）【NGKI0311】．

(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【NGKI0312】．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【NGKI0313】．

なお，発生したCPU例外に対して，CPU例外ハンドラを登録していない場合の振舞いは，ターゲット定義である【NGKI0314】．

【使用上の注意】

CPU例外入口処理でCPU例外が発生し，それを処理するためのCPU例外ハンドラの入口処理で同じ原因でCPU例外が発生すると，CPU例外が繰り返し発生し，アプリケーションが登録したCPU例外ハンドラまで処理が到達しない状況が考えられる．このような状況が発生するかどうかはターゲットによるが，これが許容できない場合には，CPU例外入口処理を経由せずに，アプリケーションが用意したCPU例外ハンドラを直接実行するようにしなければならない．

【補足説明】

マルチプロセッサ対応カーネルにおけるCPU例外ハンドラ番号の付与方法は，CPU例外ハンドラ番号の上位ビットに，そのCPU例外が発生するプロセッサのID番号を含める方法を基本とする．

【μITRON4.0仕様との関係】

μITRON4.0仕様では，CPU例外からのリカバリ処理の方法については，記述されていない．

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例外発生時にタスク例外処理ルーチンを実行開

始できない状態であったか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〔NGKI0315〕．

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った場合にfalseを返す〔NGKI0316〕．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〔NGKI0317〕．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〔NGKI0318〕．

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った場合にfalseを返す〔NGKI0319〕．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〔NGKI0320〕．

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，ext_ker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〔NGKI0321〕．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〔NGKI0322〕．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．

〔μITRON4.0仕様との関係〕

CPU例外ハンドラで行える操作に関しては，μITRON4.0仕様を見直し，全面的に修正した．

2.8.3. エミュレートされたCPU例外ハンドラ

エラーコードによってアプリケーションに通知できないエラーをカーネルが検出した場合に，アプリケーションが登録したエラー処理を，カーネルが呼び出す場合がある〔NGKI0323〕．この場合に，カーネルが検出するエラーをCPU例外と同等に扱うものとし，エミュレートされたCPU例外と呼ぶ〔NGKI0324〕．また，エラー処理のためのプログラムをCPU例外ハンドラと同等に扱うものとし，エミュレートされたCPU例外ハンドラと呼ぶ〔NGKI0325〕．

具体的には，エミュレートされたCPU例外ハンドラに対してもCPU例外ハンドラ番号が付与され，CPU例外ハンドラと同じ方法で登録できる〔NGKI0326〕．また，エミュレートされたCPU例外ハンドラからも，CPU例外ハンドラから呼び出せるサービスコールを呼び出すことができ，CPU例外ハンドラと同様のリカバリ処理を行うことができる〔NGKI0327〕．

〔μITRON4.0仕様との関係〕

エミュレートされたCPU例外およびCPU例外ハンドラは，μITRON4.0仕様に定義されていない概念である．

2.8.4. カーネル管理外のCPU例外

カーネル非動作状態，カーネル内のクリティカルセクションの実行中，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例外とする．

それに対して，カーネル管理外のCPU例外以外のCPU例外をカーネル管理のCPU例外，カーネル管理のCPU例外によって起動されるCPU例外ハンドラをカーネル管理のCPU例外ハンドラと呼ぶ．

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〔NGKI0330〕．そのため，「2.8.2 CPU例外ハンドラから呼び出せるサービスコール」の節で述べた制限［NGKI0321］［NGKI0322］が課される．

【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．

2.9. システムの初期化と終了

2.9.1. システム初期化手順

システムのリセット後，最初に実行するプログラムを，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【NGKI0331】．

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を除くすべての割込みのマスク（全割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，非初期化データセクション（bssセクション）のクリア，初期化データセクション（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【NGKI0332】．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．

マルチプロセッサ対応カーネルでは，すべてのプロセッサがスタートアップモジュールを実行し，カーネルの初期化処理へ分岐する【NGKI0333】．ただし，共有リソースの初期化処理（非初期化データセクションのクリア，初期化データセクションの初期化，ソフトウェア環境依存の初期化フックの呼出しなど）は，マスタプロセッサのみで実行する【NGKI0334】．各プロセッサがカーネルの初期化処理へ分岐するのは，共有リソースの初期化処理が完了した後でなければならないため，スレーブプロセッサは，カーネルの初期化処理へ分岐する前に，マスタプロセッサによる共有リソースの初期化処理の完了を待ち合わせる必要がある【NGKI0335】．

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【NGKI0336】．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【NGKI0337】．

静的APIの処理順序によりシステムの規定された振舞いが変化する場合には，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【NGKI0338】．例えば，静的APIによって同じ優先度のタスクを複数生成・起動した場合，静的APIの記述順が先のタスクが高い優先順位を持つ．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），静的APIの記述順となるとは限らない．

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【NGKI0339】．

マルチプロセッサ対応カーネルでは，すべてのプロセッサがカーネル自身の初期化処理と静的APIの処理を完了した後に，マスタプロセッサがグローバル初期化ルーチンを実行する【NGKI0340】．グローバル初期化ルーチンの実行が完了した後に，各プロセッサは，自プロセッサに割り付けられたローカル初期化ルーチンを実行する【NGKI0341】．すなわち，ローカル初期化ルーチンは，初期割付けプロセッサにより実行される．

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【NGKI0342】．具体的には，システム状態が，全割込みロック解除状態・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設定され（すなわち，割込みがマスク解除され），タスクの実行が開始される．

マルチプロセッサ対応カーネルでは，すべてのプロセッサがローカル初期化ルーチンの実行を完了した後に，カーネル非動作状態から動作状態に遷移し，カーネルの動作が開始される【NGKI0343】．マルチプロセッサ対応カーネルにおけるシステム初期化の流れと，各プロセッサが同期を取るタイミングを，図2-6に示す【NGKI0344】．

【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ないものとした．

2.9.2. 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）【NGKI0345】．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．

□□□

ができる【NGKI0367】．この場合に，登録解除前のオブジェクトに対して行う つもりの操作が，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【NGKI0369】．

タスク以外の処理単位は，その処理単位が実行されている間でも，登録解除す ることができる【NGKI0370】．この場合，登録解除された処理単位に実行が強制的に終了させられることはなく，処理単位が自ら実行を終了するまで，処理 単位の実行は継続される【NGKI0371】．

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【NGKI0372】．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【NGKI0373】．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【NGKI0374】．

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【NGKI0375】．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【NGKI0376】．

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），オブジェクトのアクセス許可ベクタを設定するサービスコール（sac_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【NGKI0377】．

【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生成する静的API（CRA_YYY）は 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．これにあわせて，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）も廃止した．

【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．

2.10.2. 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【NGKI0378】．

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を，オブジェクトの定義情報を入れたパケットへのポインタをNULLとして呼び出すことによって，登録を解除することができる【NGKI0379】．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【NGKI0380】．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【NGKI0381】．

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【NGKI0382】．登録を解除しようとした場合には，E_OBJエラーとなる【NGKI0383】．

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【NGKI0384】．

【μITRON4.0仕様との関係】

□□□

【仕様決定の理由】

カーネルオブジェクトをサービスコールによって登録する場合に，オブジェクトを属させる保護ドメインをオブジェクト属性で指定することにしたのは，保護機能対応でないカーネルとの互換性のためには，サービスコールのパラメータを増やさない方が望ましいためである．

2.10.6. オブジェクトが属するクラスの設定

マルチプロセッサ対応カーネルにおいて，カーネルオブジェクトが属するクラスは，オブジェクトの登録時に決定し，登録後に変更することはできない【NGKI0395】．

カーネルオブジェクトを静的APIによって登録する場合には，オブジェクトを登録する静的APIを，そのオブジェクトを属させるクラスの囲みの中に記述する【NGKI0396】．クラスに属さないオブジェクトを登録する静的APIは，クラスの囲みの外に記述する（「2.12.4 クラスの指定」の節を参照）【NGKI0397】．

カーネルオブジェクトをサービスコールによって登録する場合には，オブジェクト属性にTA_CLS(clsid)を指定することにより，オブジェクトを属させるクラスを設定する【NGKI0398】．ここでclsidは，そのオブジェクトを属させるクラスのID番号であり，clsidにTCLS_SELF（=0）を指定するか，オブジェクト属性にTA_CLS(clsid)を指定しないことで，自タスクが属するクラスに属させることができる．

割付け可能なID番号の数を指定する静的API（AID_YYY）で確保したID番号は，静的APIを囲むクラスに属するオブジェクトにのみ割り付けられる【NGKI0399】．これらの静的APIは，確保したID番号を割り付けるオブジェクトの属すべきクラスの囲みの中に記述しなければならない．クラスの囲みの外に記述した場合には，E_RSATRエラーとなる【NGKI0401】．

【補足説明】

この仕様では，カーネルオブジェクトの属するクラスを参照する機能は用意していない．

【仕様決定の理由】

カーネルオブジェクトをサービスコールによって登録する場合に，オブジェクトを属させるクラスをオブジェクト属性で指定することにしたのは，マルチプロセッサ対応でないカーネルとの互換性のためには，サービスコールのパラメータを増やさない方が望ましいためである．

2.10.7. オブジェクトの状態参照

ID番号で識別するオブジェクトのすべてと，オブジェクト番号で識別するオブジェクトの一部に対して，オブジェクトの状態を参照するサービスコール（ref_yyy，get_yyy）を用意する【NGKI0402】．

オブジェクトの状態を参照するサービスコールでは，オブジェクトの登録時に指定し，その後に変化しない情報（例えば，タスクのタスク属性や初期優先度）を参照するための機能は用意しないことを原則とする【NGKI0403】．自タスクの拡張情報の参照するサービスコール（get_inf）は，この原則に対する例外である【NGKI0404】．

2.11. オブジェクトのアクセス保護

この節では，カーネルオブジェクトのアクセス保護について述べる．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．

2.11.1. オブジェクトのアクセス保護とアクセス違反の通知

カーネルオブジェクトに対するアクセスは，そのオブジェクトに対して設定されたアクセス許可ベクタによって保護される【NGKI0405】．ただし，アクセス許可ベクタを持たないオブジェクトに対するアクセスは，システム状態に対するアクセス許可ベクタによって保護される【NGKI0406】．また，オブジェクトを登録するサービスコールと，特定のオブジェクトに関連しないシステムの状態に対するアクセスについては，システム状態のアクセス許可ベクタによって保護される【NGKI0407】．

アクセス許可ベクタによって許可されていないアクセス（アクセス違反）は，カーネルによって検出され，以下の方法によって通知される．

サービスコールにより，メモリオブジェクト以外のカーネルオブジェクトに対して，許可されていないアクセスを行おうとした場合，サービスコールから

E_OACVエラーが返る【NGKI0408】．また，メモリオブジェクトに対して，許可されていない管理操作または参照操作を行おうとした場合も，サービスコールからE_OACVエラーが返る【NGKI0409】．

メモリオブジェクトに対して，通常のメモリアクセスにより，許可されていない書込みアクセスまたは読出しアクセス（実行アクセスを含む）を行おうとした場合，CPU例外ハンドラが起動される【NGKI0410】．どのCPU例外ハンドラが起動されるかは，ターゲット定義である【NGKI0411】．ターゲットによっては，エミュレートされたCPU例外ハンドラの場合もある．また，ターゲット定義で，アクセス違反の状況に応じて異なるCPU例外ハンドラが起動される場合もある．この（これらの）CPU例外ハンドラを，メモリアクセス違反ハンドラと呼ぶ．

メモリオブジェクトに対して，サービスコールを通じて，許可されていない書込みアクセスまたは読出しアクセスを行おうとした場合，サービスコールからE_MACVエラーが返るか，メモリアクセス違反ハンドラが起動される【NGKI0412】．E_MACVエラーが返るかメモリアクセス違反ハンドラが起動されるかは，ターゲット定義である【NGKI0413】．

メモリアクセス違反ハンドラでは，アクセス違反を発生させたアクセスに関する情報（アクセスした番地，アクセスの種別，アクセスした命令の番地など）を参照する方法を，ターゲット定義で用意する【NGKI0414】．

メモリオブジェクトとしてカーネルに登録されていないメモリ領域に対して，ユーザドメインから書込みアクセスまたは読出しアクセス（実行アクセスを含む）を行おうとした場合には，メモリオブジェクトに対するアクセスが許可されていない場合と同様に扱われる【NGKI0415】．カーネルドメインから同様のアクセスを行おうとした場合の動作は保証されない【NGKI0416】．

【未決定事項】

マルチプロセッサ対応カーネルにおいて，システム状態のアクセス許可ベクタをシステム全体で1つ持つかプロセッサ毎に持つかは，今後の課題である．

【μITRON4.0/PX仕様との関係】

μITRON4.0/PX仕様では，アクセス保護の実装定義の制限について規定しているが，この仕様では，メモリオブジェクトに対するアクセス許可ベクタのターゲット定義の制限以外については規定していない．

【仕様決定の理由】

オブジェクトを登録するサービスコールを，そのオブジェクトのアクセス許可 ベクタによって保護しないのは，オブジェクトを登録する前には，アクセス許 可ベクタが設定されていないためである．

2.11.2. メモリオブジェクトに対するアクセス許可ベクタの制限

メモリオブジェクトの書込みアクセスと読出しアクセス（実行アクセスを含む）に対して設定できるアクセス許可パターンは，ターゲット定義で制限される場 合がある【NGKI0417】．

ただし，少なくとも，次の5つの組み合わせの設定は，行うことができる．

1. メモリオブジェクトが属する保護ドメインのみに，読出しアクセス（実行 アクセスを含む）のみを許可する【NGKI0418】．これを，専有リードオン リー（private read only）と呼ぶ．
2. メモリオブジェクトが属する保護ドメインのみに，書込みアクセスと読出 しアクセス（実行アクセスを含む）を許可する【NGKI0419】．これを，専 有リードライト（private read/write）と呼ぶ．
3. すべての保護ドメインに，読出しアクセス（実行アクセスを含む）のみを 許可する【NGKI0420】．これを，共有リードオンリー（shared read only）と呼ぶ．
4. すべての保護ドメインに，書込みアクセスと読出しアクセス（実行アクセ スを含む）を許可する【NGKI0421】．これを，共有リードライト（shared read/write）と呼ぶ．
5. メモリオブジェクトが属する保護ドメインに，書込みアクセスと読出しアクセス（実行アクセスを含む）を許可し，他の保護ドメインには，読出しアクセス（実行アクセスを含む）のみを許可する【NGKI0422】．これを，共有リード専有ライト（shared read private write）と呼ぶ．

また，ターゲット定義で，1つの保護ドメインに登録できるメモリオブジェクト の数が制限される場合がある【NGKI0423】．

2.11.3. デフォルトのアクセス許可ベクタ

カーネルオブジェクトを登録した直後は，次に規定されるデフォルトのアクセス許可ベクタが設定される．

保護ドメインに属するカーネルオブジェクトに対しては，4つの種別のアクセスがいずれも，その保護ドメインのみに許可される【NGKI0424】．すなわち，カーネルドメインに属するオブジェクトに対しては，4つのアクセス許可パターンがいずれもTACP_KERNELに，ユーザドメインに属するオブジェクトに対しては，4つのアクセス許可パターンがいずれもTACP(domid)（domidはオブジェクトが属する保護ドメインのID番号）に設定される．ただし，カーネルオブジェクトをサービスコールにより登録した場合には，管理操作に対するアクセスは，サービスコールを呼び出した処理単位が属する保護ドメインにも許可される【NGKI3427】．

無所属のカーネルオブジェクトに対しては，4つの種別のアクセスがいずれも，すべての保護ドメインに許可される【NGKI0425】．すなわち，4つのアクセス許可パターンがいずれも，TACP_SHAREDに設定される．

システム状態のアクセス許可ベクタは，4つの種別のアクセスがいずれも，カーネルドメインのみに許可される【NGKI0426】．すなわち，4つのアクセス許可パターンがいずれも，TACP_KERNELに設定される．

2.11.4. アクセス許可ベクタの設定

アクセス許可ベクタをデフォルト以外の値に設定するために，カーネルオブジェクトのアクセス許可ベクタを設定する静的API（SAC_YYY）と，システム状態のアクセス許可ベクタを設定する静的API（SAC_SYS）が用意されている【NGKI0427】．

また，動的生成対応カーネルにおいては，カーネルオブジェクトのアクセス許可ベクタを設定するサービスコール（sac_yyy）と，システム状態のアクセス許可ベクタを設定するサービスコール（sac_sys）が用意されている【NGKI0428】．ただし，静的APIによって登録したオブジェクトは，サービスコール（sac_yyy）によってアクセス許可ベクタを設定することができない．アクセス許可ベクタを設定しようとした場合には，E_OBJエラーとなる【NGKI0430】．

メモリオブジェクトに対しては，アクセス許可ベクタを設定する静的APIは用意されておらず，オブジェクトの登録と同時にアクセス許可ベクタを設定する静的API（ATA_YYY）が用意されている【NGKI0431】．

オブジェクトに対するアクセスが許可されているかは，そのオブジェクトにアクセスするサービスコールを呼び出した時点でチェックされる【NGKI0432】．そのため，アクセス許可ベクタを変更しても，変更以前に呼び出されたサービスコールの振舞いには影響しない．例えば，待ち行列を持つ同期・通信オブジェクトのアクセス許可ベクタを変更しても，呼び出した時点ですでに待ち行列につながれているタスクには影響しない．また，ミューテックスのアクセス許可ベクタを変更しても，呼び出した時点ですでにミューテックスをロックしていたタスクには影響しない．

この仕様では，カーネルオブジェクトに設定されたアクセス許可ベクタを参照する機能は用意していない．

【使用上の注意】

カーネルオブジェクトのアクセス許可ベクタをデフォルト以外の値に設定する際に，オブジェクトに対して同じ保護ドメインに属する処理単位からアクセスできるようにするには，その保護ドメインからアクセスできることを明示的に指定する必要がある．

【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生成する静的API（CRA_YYY）は廃止し，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．

静的APIによって登録したオブジェクトは，サービスコール（sac_yyy）によってアクセス許可ベクタを設定することができないこととした．

オブジェクトの状態参照するサービスコール（ref_yyy）により，オブジェクトに設定されたアクセス許可ベクタを参照する機能サポートしないこととした．これは，［NGKI0403］の原則に合わせるための修正である．

2.11.5. カーネルの管理領域のアクセス保護

カーネルが動作するために，カーネルの内部で用いるメモリ領域を，カーネルの管理領域と呼ぶ．ユーザタスクからカーネルを保護するためには，カーネルの管理領域にアクセスできるのは，カーネルドメインのみでなければならない．そのため，カーネルの管理領域は，書込みアクセスおよび読出しアクセスが可能で，4つの種別のアクセスがカーネルドメインのみに許可されたメモリオブジェクト（これを，カーネル専用のメモリオブジェクトと呼ぶ）の中に置かれる【NGKI0433】．

カーネルの用いるオブジェクト管理領域（カーネルの管理領域に該当する．「2.10.4 オブジェクト生成に必要なメモリ領域」の節を参照）として，カーネル専用のメモリオブジェクトに含まれないメモリ領域を指定した場合，E_OBJエラーとなる【NGKI0434】．また，カーネルの用いるオブジェクト管理領域の先頭番地にNULLを指定した場合，必要なメモリ領域が，カーネル専用のメモリオブジェクトの中に確保される【NGKI0435】．

システムタスクのスタック領域，ユーザタスクのシステムスタック領域，非タスクコンテキスト用のスタック領域は，カーネルの用いるオブジェクト管理領域には該当しないが，カーネルドメインの実行中にのみアクセスされるため，カーネルの用いるオブジェクト管理領域と同様の扱いとなる【NGKI0436】．一方，ユーザタスクのユーザスタック領域と固定長メモリプール領域は，ユーザドメインの実行中にもアクセスされるため，カーネルの用いるオブジェクト管理領域とは異なる扱いとなる．

2.11.6. ユーザタスクのユーザスタック領域

ユーザタスクが非特権モードで実行する間に用いるスタック領域を，システムスタック領域（「4.1 タスク管理機能」の節を参照）と対比させて，ユーザスタック領域と呼ぶ．ユーザスタック領域は，そのタスクと同じ保護ドメインに属する1つのメモリオブジェクトとしてカーネルに登録される【NGKI0437】．ただし，他のメモリオブジェクトとは異なり，次のように扱われる．

タスクのユーザスタック領域に対しては，そのタスクのみが書込みアクセスおよび読出しアクセスを行うことができる【NGKI0438】．そのため，書込みアクセスと読出しアクセス（実行アクセスを含む）に対するアクセス許可パターンは意味を持たない【NGKI0439】．ユーザスタック領域に対して実行アクセスを行えるかどうかは，ターゲット定義である【NGKI0440】．

ただし，上記の仕様を実現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，タスクのユーザスタック領域を，そのタスクが属する保護ドメイン全体からアクセスできるものとする場合がある【NGKI0441】．

【μITRON4.0/PX仕様との関係】

この仕様では，タスクのユーザスタック領域は，そのタスクのみがアクセスできるものとした．

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，保護ドメインの囲み，クラスの囲み，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【NGKI0442】．

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）【NGKI0443】．

```
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分として解釈する【NGKI0444】．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【NGKI0445】．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【NGKI0446】．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【NGKI0447】．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の記述の途中に，条件ディレクティブを記述することはできない【NGKI0448】．

コンフィギュレータは，システムコンフィギュレーションファイル中の静的APIを，その記述順に解釈する【NGKI0449】．そのため例えば，タスクを生成する静的APIの前に，そのタスクにタスク例外処理ルーチンを定義する静的APIが 記述されていた場合，タスク例外処理ルーチンを定義する静的APIがE_NOEXSエ ラーとなる．

【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【NGKI0450】．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【NGKI0451】．

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【NGKI0452】．

静的APIのパラメータは，次の4種類に分類される．

1. 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【NGKI0453】．

   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，SAC_TSKやDEF_TEXのtskidパラメータ等がこれに該当する）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【NGKI0455】．

   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【NGKI0456】．

2. 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番地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．

   整数定数式の解釈に必要な定義や宣言等は，システムコンフィギュレーションファイルからC言語プリプロセッサのインクルードディレクティブによってインクルードするファイルに含まれていなければならない【NGKI0457】．

3. 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，番地を指定 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．

□□□

```
KERNEL_DOMAIN {
    □□□□
```

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，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に変換したSレコードファイル（cfg1_out.srec）と，その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の整数定数式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で生成されたロードモジュールのSレコードファイル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，そのSレコードファイル（system.srec）とシンボルファイル（system.syms）を生成する．静的APIの一般定数式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3では，パス1およびパス2で生成されたロードモジュールのSレコードファイルとシンボルファイルから，静的APIの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．

保護機能対応カーネルにおいては，メモリ配置を決定し，メモリ保護のための設定情報を生成するために，さらに以下の処理を行う（図2-9）．

コンフィギュレータは，決定したメモリ配置に従ってロードモジュールを生成するために，リンクスクリプト（ldscript.ld）を生成する．また，メモリ保護のための設定情報を，メモリ構成・初期化ファイル（kernel_mem.c）に生成する．これらのファイルを生成するためには，パス3以降で初めて得られる情報が必要となるため，これらのファイルはパス3以降でしか生成できず，最終的なロードモジュールも，パス3以降で生成する．

そのため，パス2で生成されたロードモジュールは，仮のロードモジュールという位置付けになる．ここで，パス3以降で必要な情報を取り出し，最終的なロードモジュールのサイズを割り出せるように，パス3以降でメモリ構成・初期化ファイルに生成するのと同様のデータ構造を，パス2において仮のメモリ構成・初期化ファイル（kernel_mem2.c）に生成する．また，これをリンクするための仮のリンクスクリプト（cfg2_out.ld）を生成し，これらを用いて仮のロードモジュールを生成する．さらに，仮のロードモジュールのSレコードファイル（cfg2_out.srec）とシンボルファイル（cfg2_out.syms）も，最終的なものと混同しないように，異なるファイル名で生成する．

パス3は，ターゲット依存で用いるパスで，メモリ配置やメモリ保護のための設定情報のサイズを最適化するための処理を行う．パス2で生成された仮のロードモジュールのSレコードファイルとシンボルファイルから必要な情報を取り出し，再度，仮のメモリ構成・初期化ファイル（kernel_mem3.c）と仮のリンクスクリプト（cfg3_out.ld）を生成する．また，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て仮のロードモジュールを生成し，そのSレコードファイル（cfg3_out.srec）とシンボルファイル（cfg3_out.syms）を生成する．この段階で，メモリオブジェクトに重なりがあるなどのエラーが検出される場合もある．

パス4では，パス3（パス3を用いない場合はパス2）で生成された仮のロードモジュールのSレコードファイルとシンボルファイルから必要な情報を取り出し，最終的なメモリ構成・初期化ファイル（kernel_mem.c）とリンクスクリプト（ldscript.ld）を生成する．またパス4では，保護機能対応でないカーネルにおいてパス3で行っていた静的APIパラメータの値などの妥当性のチェックも行う．そのため，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．

パス4で生成されたファイルを，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最終的なロードモジュールを生成する．また，そのSレコードファイル（system.srec，必要な場合のみ）とシンボルファイル（system.syms）を生成する．

最後に，最終的なロードモジュールが，パス3（パス3を用いない場合はパス2）で生成された仮のロードモジュールと同じメモリ配置であることをチェックする．両者のメモリ配置が異なっていた場合には，ロードモジュールが正しく生成されていない可能性があるが，これは，コンフィギュレーション処理の不具合を示すものである．

【μITRON4.0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【NGKI0478】．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」

□□□

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．

1. 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
```

2.14. 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．

1. 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．

   【補足説明】

   サービスコールの名称を構成する省略名（xxx，yyy，z）の元になった英語については，「5.10 省略名の元になった英語」の節を参照すること．

2. 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．

2.14.1. 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

2.14.2. 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
```

2.14.3. 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
```

2.14.4. 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 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．

2.14.5. 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．

2.14.6. 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．

2.15. 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．

2.15.1. 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【NGKI0484】．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【NGKI0485】．

TOPPERS共通ヘッダファイルは，カーネルヘッダファイル（kernel.h）やシステ ムインタフェースレイヤヘッダファイル（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【NGKI0486】．

2.15.2. 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【NGKI0487】．

```
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）
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）

MB_T	オブジェクト管理領域を確保するためのデータ型

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，ターゲット定義である〔NGKI0488〕．これらが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〔NGKI0489〕．IEEE754準拠の浮動小数点数がサポートされていない場合には，ターゲット定義で，float32_tとdouble64_tは使用できないものとする〔NGKI0490〕．

〔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．

VP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，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，オブジェクト管理領域を確保するためのデータ型としてMB_Tを用意することにした

2.15.3. 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〔NGKI0491〕

```
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〔NGKI0492〕

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 [NGKI0493]

CHAR_BIT	char型のビット数（C90準拠）

(4) オブジェクト属性 [NGKI0494]

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 [NGKI0495]

```
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【NGKI0496】

```
TACP_KERNEL     0U      カーネルドメインのみにアクセスを許可
TACP_SHARED     ~0U     すべての保護ドメインにアクセスを許可
```

2.15.4. 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【NGKI0497】.

(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
E_NORES     -35     資源不足
```

(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付け
E_BOVR      -58     バッファオーバフロー
```

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，［NGKI0019］の原則では例外としている．

【μITRON4.0仕様との関係】

E_NORESは，μITRON4.0仕様に規定されていないエラーコードである．

2.15.5. TOPPERS共通マクロ

(1) 整数定数を作るマクロ【NGKI0498】

```
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 [NGKI0499]

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)	番地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 [NGKI0500]

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 [NGKI0501]

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) エラーコード構成・分解マクロ [NGKI0502]

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，エラーコードを構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構成マクロ [NGKI0503]

TACP(domid)	domidで指定されるユーザドメインのみにアクセスを許可するアクセス許可パターンを構成するためのマクロ

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 [NGKI0504]．TDOM_SELF，TDOM_KERNEL，TDOM_NONEを指定し

□□□

(3) その他のカーネル共通定数【NGKI0514】

```
TCLS_SELF       0       自タスクの属するクラス
```

```
TPRC_NONE       0       割付けプロセッサの指定がない
TPRC_INI        0       初期割付けプロセッサ
```

```
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

「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【NGKI0515】．

【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_RSFN，E_RSATR，E_MACV，E_OACV，E_NOMEM，E_NOID，E_NORES，E_NOEXSが返る状況は起こらない【ASPS0011】．E_RSATRは，コンフィギュレータによって検出される【ASPS0012】．ただし，動的生成機能拡張パッケージでは，サービスコールから，E_RSATR，E_NOMEM，E_NOID，E_NOEXSが返る状況が起こる【ASPS0013】．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，サービスコールから，E_RSFN，E_RSATR，E_MACV，E_OACV，E_NOMEM，E_NOID，E_NORES，E_NOEXSが返る状況は起こらない【FMPS0007】．E_RSATRとE_NORESは，コンフィギュレータによって検出される【FMPS0008】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，サービスコールから，E_RSATR，E_NOMEM，E_NOID，E_NORES，E_NOEXSが返る状況は起こらない【HRPS0006】．E_RSATRは，コンフィギュレータによって検出される【HRPS0007】．ただし，動的生成機能拡張パッケージでは，サービスコールから，E_RSATR，E_NOMEM，E_NOID，E_NOEXSが返る状況が起こる【HRPS0011】．

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは，サービスコールから，E_RSFN，E_RSATR，E_MACV，E_OACV，E_ILUSE，E_NOMEM，E_NOID，E_NORES，E_NOEXS，E_RLWAI，E_TMOUT，E_DLTが返る状況は起こらない【SSPS0008】．E_RSATRは，コンフィギュレータによって検出される【SSPS0009】．

2.15.8. カーネル共通マクロ

(1) スタック領域をアプリケーションで確保するためのデータ型とマクロ

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【NGKI0516】．

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
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【NGKI0517】．

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>)];

この方法で確保したスタック領域を，サービスコールまたは静的APIに渡す場合 には，スタック領域の先頭番地に<スタック領域の変数名>を，スタック領域の サイズにROUND_STK_T(<スタック領域のサイズ>)を指定する【NGKI0518】．

ただし，保護機能対応カーネルにおいては，上の方法によりタスクのユーザス タック領域を確保することはできない【NGKI0519】．詳しくは，「4.1 タスク管理機能」の節のCRE_TSKの機能の項を参照すること．

(2) オブジェクト属性を作るマクロ

保護機能対応カーネルでは，オブジェクトが属する保護ドメインを指定するた めのオブジェクト属性を作るマクロとして，次のマクロを用意している【NGKI0520】．

TA_DOM(domid)	domidで指定される保護ドメインに属する

マルチプロセッサ対応カーネルでは，オブジェクトが属するクラスを指定する ためのオブジェクト属性を作るマクロとして，次のマクロを用意している【NGKI0521】．

TA_CLS(clsid)	clsidで指定されるクラスに属する

(3) サービスコールの呼出し方法を指定するマクロ

保護機能対応カーネルでは，サービスコールの呼出し方法を指定するためのマ クロとして，次のマクロを用意している【NGKI0522】．

SVC_CALL(svc)	svcで指定されるサービスコールを関数呼出しによっ て呼び出すための名称

2.15.9. カーネル共通構成マクロ

(1) サポートする機能【NGKI0523】

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【NGKI0524】

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【ASPS0014】．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【ASPS0015】．

【TOPPERS/FMPカーネルにおける規定】

FMP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【FMPS0009】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2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【HRPS0008】．

【TOPPERS/SSPカーネルにおける規定】

SSP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【SSPS0010】．

【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) プロセッサの数

マルチプロセッサ対応カーネルでは，プロセッサの数を知るためのマクロとして，次の構成マクロを用意している【NGKI0525】．

TNUM_PRCID	プロセッサの数

(4) 特殊な役割を持ったプロセッサ

マルチプロセッサ対応カーネルでは，特殊な役割を持ったプロセッサを知るためのマクロとして，次の構成マクロを用意している【NGKI0526】．

TOPPERS_MASTER_PRCID	マスタプロセッサのID番号
TOPPERS_SYSTIM_PRCID	システム時刻管理プロセッサのID番号（グローバルタイマ方式の場合のみ）

(5) □□□□□

□□□□

- エンディアンの取得のためのマクロを参照すること
- メモリ空間アクセス関数により，アクセスを許可されたメモリ領域にアクセスすること
- I/O空間アクセス関数により，アクセスを許可されたI/O領域にアクセスすること

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【NGKI0803】．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【NGKI0804】．

3.3. 全割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込み（NMI以外にカーネル管理外の割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，すべての割込みをマスクする全割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

全割込みロック状態の制御に必要な変数を宣言するマクロ【NGKI0805】．通常は，型と変数名を並べたもので，最後に";"を含まない．

このマクロは，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【NGKI0806】．SIL_LOC_INT，SIL_UNL_INTを1つの関数内でネストして用いることは可能であるが，その場合には，ネストレベル毎にブロックを作り，そのブロックの先頭の変数宣言部にSIL_PRE_LOCを記述しなければならない【NGKI0807】．そのように記述しなかった場合の動作は保証されない【NGKI0808】．

(2) SIL_LOC_INT()

全割込みロックフラグをセットすることで，NMIを除くすべての割込みをマスクし，全割込みロック状態に遷移する【NGKI0809】．

(3) SIL_UNL_INT()

全割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【NGKI0810】．SIL_LOC_INTを実行せずにSIL_UNL_INTを呼び出した場合の動作は保証されない【NGKI0811】．

なお，全割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 全割込みロック状態と全割込みロック解除状態」の節を参照すること．

【補足説明】

全割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();
}
```

3.4. SILスピンロック

マルチプロセッサシステムにおいて，カーネルの機能を用いずに，他のプロセッサとの間でも排他制御を実現したい場合がある．そこでSILでは，割込みのマスクとプロセッサ間ロックの取得により排他制御を行うためのスピンロックの機 能を用意している．これを，カーネルのスピンロック機能と区別するために， SILスピンロックと呼ぶ．

プロセッサ間ロックを取得している間は，全割込みロック状態にすることですべての割込み（NMIを除く）がマスクされる【NGKI0812】．ロックが他のプロセッサに取得されている場合には，ロックが取得できるまでループによって待つ【NGKI0813】．ロックの取得を待つ間は，割込みはマスクされない（ロックの取得を試みる前にマスクしていた割込みは，マスク解除されない）【NGKI0814】．プロセッサ間ロックを取得し割込みをマスクすることを，SILスピンロックを取得するという．また，プロセッサ間ロックを返却し割込みをマ スク解除することを，SILスピンロックを返却するという．

SILで取得・返却するプロセッサ間ロックは，システムに唯一存在する【NGKI0815】．

(1) SIL_PRE_LOC

全割込みロック状態の制御に必要な変数を宣言するマクロであるが，SILスピン ロックの取得・解放にも兼用する【NGKI0816】．

このマクロは，SIL_LOC_SPN，SIL_UNL_SPN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【NGKI0817】．SIL_LOC_SPN，SIL_UNL_SPNを，同じ関数内のSIL_LOC_INT，SIL_UNL_INTとネストして用いることは可能であるが，その場合には，ネストレベル毎にブロックを作り，そのブロックの先頭の変数宣言部にSIL_PRE_LOCを記述しなければならない【NGKI0818】．そのように記述しなかった場合の動作は保証されない【NGKI0819】．

(2) SIL_LOC_SPN()

SILスピンロックが取得されていない状態である場合には，プロセッサ間ロックの取得を試みる【NGKI0820】．ロックが他のプロセッサに取得されている状態である場合や，他のプロセッサがロックの取得に成功した場合には，ロックが返却されるまでループによって待ち，返却されたらロックの取得を試みる【NGKI0821】．ロックの取得に成功した場合には，全割込みロックフラグをセッ トし，全割込みロック状態に遷移する【NGKI0822】．

(3) SIL_UNL_SPN()

プロセッサ間ロックを返却し，全割込みロックフラグを対応するSIL_LOC_SPNを 実行する前の状態に戻す【NGKI0823】．

SILスピンロックを取得している状態でSIL_LOC_SPNを呼び出した場合の動作は保証されない【NGKI0824】．逆に，SILスピンロックを取得していない状態でSIL_UNL_SPNを呼び出した場合の動作も保証されない【NGKI0825】．

なお，SILスピンロック取得中は全割込みロック状態となっているため，SILスピンロック取得中に呼び出せるサービスコールなどについては，「2.5.4 全割込みロック状態と全割込みロック解除状態」の節の制限事項が適用される．

なお，マルチプロセッサシステム以外では，SIL_LOC_SPNとSIL_UNL_SPNは用意 されていない【NGKI0826】．

【使用上の注意】

全割込ロック状態やCPUロック状態でSIL_LOC_SPNを呼び出すことはできるが，割込みがマスクされている時間が長くなるために，そのような使い方は避ける べきである．

【補足説明】

SILスピンロック機能の使用例は次の通り．

```
{
    SIL_PRE_LOC;

    SIL_LOC_SPN();
    // この間はSILスピンロックを取得している.
    // この間はNMIを除くすべての割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_SPN();
}
```

3.5. 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【NGKI0827】．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．

3.6. エンディアンの取得

プロセッサのバイトエンディアンを取得す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_ENDIAL_LITTLEを，マクロ定義している【NGKI0828】．

3.7. 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(const uint8_t *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【NGKI0829】．

(2) void sil_wrb_mem(uint8_t *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【NGKI0830】．

(3) uint16_t sil_reh_mem(const uint16_t *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【NGKI0831】．

(4) void sil_wrh_mem(uint16_t *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【NGKI0832】．

(5) uint16_t sil_reh_lem(const uint16_t *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【NGKI0833】．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(uint16_t *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【NGKI0834】．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(const uint16_t *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【NGKI0835】．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(uint16_t *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【NGKI0836】．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(const uint32_t *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【NGKI0837】．

(10) void sil_wrw_mem(uint32_t *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【NGKI0838】．

(11) uint32_t sil_rew_lem(const uint32_t *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【NGKI0839】．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_wrw_lem(uint32_t *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【NGKI0840】．リトルエンディアンプロセッサでは，sil_wrw_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_wrw_memで書き込むのと同じ結果となる．

(13) uint32_t sil_rew_bem(const uint32_t *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【NGKI0841】．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(uint32_t *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【NGKI0842】．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．

3.8. 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【NGKI0843】．

(1) uint8_t sil_reb_iop(const uint8_t *iop)
(2) void sil_wrb_iop(uint8_t *iop, uint8_t data)

(3) uint16_t sil_reh_iop(const uint16_t *iop)
(4) void sil_wrh_iop(uint16_t *iop, uint16_t data)
(5) uint16_t sil_reh_lep(const uint16_t *iop)
(6) void sil_wrh_lep(uint16_t *iop, uint16_t data)
(7) uint16_t sil_reh_bep(const uint16_t *iop)
(8) void sil_wrh_bep(uint16_t *iop, uint16_t data)
(9) uint32_t sil_rew_iop(const uint32_t *iop)
(10) void sil_wrw_iop(uint32_t *iop, uint32_t data)
(11) uint32_t sil_rew_lep(const uint32_t *iop)
(12) void sil_wrw_lep(uint32_t *iop, uint32_t data)
(13) uint32_t sil_rew_bep(const uint32_t *iop)
(14) void sil_wrw_bep(uint32_t *iop, uint32_t data)

3.9. プロセッサIDの参照

マルチプロセッサシステムにおいては，プログラムがどのプロセッサで実行さ れているかを参照するために，以下の関数を用意している．

(1) void sil_get_pid(ID *p_prcid)

この関数を呼び出したプログラムを実行しているプロセッサのID番号を参照し，p_prcidで指定したメモリ領域に返す【NGKI0844】．

【使用上の注意】

タスクは，sil_get_pidを用いて，自タスクを実行しているプロセッサを正しく参照できるとは限らない．これは，sil_get_pidを呼び出し，自タスクを実行しているプロセッサのID番号を参照した直後に割込みが発生した場合，sil_get_pidから戻ってきた時には自タスクを実行しているプロセッサが変化し ている可能性があるためである．

4. 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．

【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．これらの変更については，個別のAPI仕様では記述しない．

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【NGKI1001】．

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【NGKI1002】．

- 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
- 割付けプロセッサ（マルチプロセッサ対応カーネルの場合）
- 次回起動時の割付けプロセッサ（マルチプロセッサ対応カーネルの場合）
- 拡張情報
- メインルーチンの先頭番地
- 起動時優先度
- 実行時優先度（TOPPERS/SSPカーネルの場合）
- 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）
- アクセス許可ベクタ（保護機能対応カーネルの場合）
- 属する保護ドメイン（保護機能対応カーネルの場合）
- 属するクラス（マルチプロセッサ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【NGKI1003】．

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 ある．単にタスクの優先度と言った場合には，現在優先度のことを指す．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【NGKI1004】．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【NGKI1005】．

割付けプロセッサは，マルチプロセッサ対応カーネルにおいて，タスクを実行 するプロセッサで，タスクの生成時に，タスクが属するクラスによって定まる初期割付けプロセッサに初期化される【NGKI1006】．

次回起動時の割付けプロセッサは，マルチプロセッサ対応カーネルにおいて， タスクが次に起動される時に割り付けられるプロセッサで，タスクの生成時に未設定の状態に初期化される【NGKI1007】．タスクの起動時に，次回起動時の 割付けプロセッサが設定されていれば，タスクの割付けプロセッサがそのプロセッサに変更され，次回起動時の割付けプロセッサは未設定の状態に戻される【NGKI1008】．次回起動時の割付けプロセッサが未設定の場合には，タスクの割付けプロセッサは変更されない（つまり，タスクが前に実行されていたのと 同じプロセッサで実行される）【NGKI1009】．

保護機能対応カーネルにおいては，スタック領域の扱いは，ユーザタスクとシ ステムタスクで異なる．ユーザタスクのスタック領域は，ユーザタスクが非特権モードで実行する間に用いるスタック領域であり，ユーザスタック領域と呼 ぶ【NGKI1010】．その扱いについては，「2.11.6 ユーザタスクのユーザスタック領域」の節を参照すること．システムタスクのスタック領域は，カーネルの 用いるオブジェクト管理領域と同様に扱われる【NGKI1011】．

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，ユーザタスクがサー ビスコール（拡張サービスコールを含む）を呼び出し，特権モードで実行する

間に用いるスタック領域である【NGKI1012】．システムスタック領域は，カーネルの用いるオブジェクト管理領域と同様に扱われる【NGKI1013】．

タスク属性には，次の属性を指定することができる【NGKI1014】．

```
TA_ACT      0x02U   タスクの生成時にタスクを起動する
TA_RSTR     0x04U   生成するタスクを制約タスクとする
```

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【NGKI1015】．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【NGKI1016】．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【NGKI1017】．

```
TA_FPU              FPUレジスタをコンテキストに含める
```

タスク終了時には，次の処理が行われる．まず，終了するタスク（対象タスク）に対してタスク終了時に行うべきその他の処理が行われた後，対象タスクは休止状態になる【NGKI1178】．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【NGKI1179】．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【NGKI1180】．

C言語によるタスクの記述形式は次の通り【NGKI1018】．

```
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【NGKI1019】．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た場合，ext_tskを呼び出した場合と同じ動作をする【NGKI1020】．

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【NGKI1021】
```

```
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）
                【NGKI1022】
```

【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【ASPS0101】．また，制約タスクはサポートしていない【ASPS0102】．ただし，制約タスク拡張パッケージを用いると，制約タスクの機能を追加することができる【ASPS0103】．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，TMAX_ACTCNTは1に固定されている【FMPS0101】．また，制約タスクはサポートしていない【FMPS0102】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，TMAX_ACTCNTは1に固定されている【HRPS0101】．また，制約タスクはサポートしていない【HRPS0102】．

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは，TMAX_ACTCNTは1に固定されている【SSPS0101】．

SSPカーネルは，制約タスクのみをサポートすることから，すべてのタスクでスタック領域を共有しており，タスク毎にスタック領域の情報を持たない【SSPS0102】．

SSPカーネルにおける追加機能として，タスクに対して，実行時優先度の情報を持つ【SSPS0103】．SSPカーネルにおいては，タスクが起動された後，最初に実行状態になる時に，タスクのベース優先度が，タスクの実行時優先度に設定される【SSPS0104】．実行時優先度の機能は，起動時優先度よりも高い優先度でタスクを実行することで，同時期に共有スタック領域を使用している状態になるタスクの組み合わせを限定し，スタック領域を節約するための機能である．

タスクの実行時優先度は，実行時優先度を定義する静的API（DEF_EPR）によって設定する【SSPS0105】．実行時優先度を定義しない場合，タスクの実行時優先度は，起動時優先度と同じ値に設定される【SSPS0106】．

〔実行時優先度によるスタック領域の節約〕

いずれのタスクにも実行時優先度が設定されていない場合には，すべてのタスクが同時期に共有スタック領域を使用している状態になる可能性があるため，すべてのタスクのスタック領域のサイズの和に，非タスクコンテキスト用のスタック領域のサイズを加えたものが，共有スタック領域に必要なサイズとなる．

タスクAに対して実行時優先度が設定されており，タスクAの起動時優先度よりも高く，タスクAの実行時優先度と同じかそれよりも低い起動時優先度を持つタスクBがある場合，タスクAとタスクBは同時期に共有スタック領域を使用している状態にならない．そのため，タスクAとタスクBの内，サイズが小さい方のスタック領域のサイズは，共有スタック領域のサイズに加える必要がなくなり，スタック領域を節約できることになる．

【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〕【NGKI1023】
acre_tsk タスクの生成〔TD〕【NGKI1024】

【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 })
```

※ sstkszおよびsstkの記述は省略することができる【NGKI1025】．

【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（CRE_TSKの場合）
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，静的API においては省略可） 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，静的APIにおいては省略可）

【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（tskid）【NGKI1026】 ・CPUロック状態からの呼出し（tskid）【NGKI1027】
E_RSATR	予約属性 ・tskatrが無効【NGKI1028】 ・属する保護ドメインの指定が有効範囲外または無所属（tskid） 【NGKI1029】 ・保護ドメインの囲みの中に記述されていない（tskid）【NGKI1030】 ・属するクラスの指定が有効範囲外（tskid）【NGKI1031】 ・クラスの囲みの中に記述されていない（tskid）【NGKI1032】
E_PAR	パラメータエラー ・taskがプログラムの先頭番地として正しくない【NGKI1033】 ・itskpriが有効範囲外【NGKI1034】 ・その他の条件については機能の項を参照
E_OACV	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（tskid） 【NGKI1035】
E_MACV	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（tskid）【NGKI1036】
E_NOID	ID番号不足 ・割り付けられるタスクIDがない（tskid）【NGKI1037】
E_NOMEM	メモリ不足 ・スタック領域が確保できない【NGKI1038】 ・システムスタック領域が確保できない（tskid）【NGKI1039】
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（CRE_TSKの場合）【NGKI1040】 ・その他の条件については機能の項を参照

【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設定される【NGKI1041】．ただし，保護機能対応カーネルで，生成するタスクがシステムタスクの場合には，sstkszも用いられる．stkszに0以下の値を指定した時や，設定されるスタック領域のサイズがターゲット定義の最小値よりも小さくなる時には，E_PARエラーとなる【NGKI1042】．

また，保護機能対応カーネルで，生成するタスクがユーザタスクの場合には，sstkとsstkszからシステムスタック領域が設定される【NGKI1043】．この場合，sstkszに0以下の値を指定した時や，ターゲット定義の最小値よりも小さい値を 指定した時には，E_PARエラーとなる【NGKI1044】．

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【NGKI1045】．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【NGKI1046】．

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【NGKI1047】．コンフィギュレータは，静的APIのメモリ不足（E_NOMEM）エラー を検出することができない【NGKI1048】．

〔stkにNULLを指定した場合〕

stkをNULLとした場合，stkszで指定したサイズのスタック領域が，コンフィギュレータまたはカーネルにより確保される【NGKI1049】．stkszにターゲット定義の制約に合致しないサイズを指定した時には，ターゲット定義の制約に合致す るように大きい方に丸めたサイズで確保される【NGKI1050】．

□□□

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，CRE_TSKのみをサポートする【FMPS0103】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，CRE_TSKのみをサポートする【HRPS0103】．

動的生成機能拡張パッケージでは，acre_tskもサポートする【HRPS0175】．ただし，生成するタスクがユーザタスクの場合，stkにNULLが指定されるとカーネルがスタック領域を確保する機能はサポートしない．stkにNULLを指定した場合には，E_NOSPTエラーとなる【HRPS0176】．

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは，CRE_TSKのみをサポートする【SSPS0107】．

SSPカーネルでは，複数のタスクに対して，同じ起動時優先度を設定することはできない．設定した場合には，コンフィギュレータがE_PARエラーを報告する【SSPS0109】．

SSPカーネルでは，制約タスクのみをサポートするため，タスク属性にTA_RSTR を指定しない場合でも，生成されるタスクは制約タスクとなる【SSPS0110】．

SSPカーネルでは，stkにはNULLを指定しなくてはならず，その場合でも，コンフィギュレータはタスクのスタック領域を確保しない【SSPS0111】．これは，SSPカーネルでは，すべての処理単位が共有スタック領域を使用し，タスク毎にスタック領域を持たないためである．stkにNULL以外を指定した場合には，E_PARエラーとなる【SSPS0112】．

共有スタック領域の設定方法については，DEF_STKの項を参照すること．

【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．COUNT_STK_TとROUND_STK_Tを新設し，スタック領域をアプリケーションで確保する方法を規定した．

【μITRON4.0/PX仕様との関係】

sstkのデータ型をSTK_T *に変更した．システムスタック領域をアプリケーションで確保する方法を規定した．

【未決定事項】

サービスコール（acre_tsk）により，stkにNULLを指定して制約タスクを生成した場合のスタック領域の確保方法については，今後の課題である．

【仕様決定の理由】

保護機能対応カーネルにおいて，sstkszおよびsstkの記述は省略することができることとしたのは，保護機能対応でないカーネル用のシステムコンフィギュレーションファイルを，保護機能対応カーネルにも変更なしに使えるようにするためである．

```
AID_TSK        割付け可能なタスクIDの数の指定（SD）【NGKI1073】
```

【静的API】

```
    AID_TSK(uint_t notsk)
```

【パラメータ】

```
uint_t      notsk       割付け可能なタスクIDの数
```

【エラーコード】

```
E_RSATR     予約属性
            ・保護ドメインの囲みの中に記述されている（P）【NGKI3428】
            ・クラスの囲みの中に記述されていない（M）【NGKI1075】
E_PAR       パラメータエラー
            ・notskが負の値【NGKI3276】
```

【機能】

notskで指定した数のタスクIDを，タスクを生成するサービスコールによって割 付け可能なタスクIDとして確保する【NGKI1076】．

notskは整数定数式パラメータである【NGKI1077】．

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルの動的生成機能拡張パッケージでは，AID_TSKをサポートする【ASPS0210】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルの動的生成機能拡張パッケージでは，AID_TSKをサポートする【HRPS0211】．

```
SAC_TSK     タスクのアクセス許可ベクタの設定（SP）【NGKI1078】
sac_tsk     タスクのアクセス許可ベクタの設定（TPD）【NGKI1079】
```

【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（S）〔NGKI1080〕 ・CPUロック状態からの呼出し（S）〔NGKI1081〕
E_ID	不正ID番号 ・tskidが有効範囲外（S）〔NGKI1082〕
E_RSATR	予約属性 ・対象タスクが属する保護ドメインの囲みの中に記述されて□いない（S）〔NGKI1083〕 ・対象タスクが属するクラスの囲みの中に記述されていない□（SM）〔NGKI1084〕
E_NOEXS	オブジェクト未登録 ・対象タスクが未登録〔NGKI1085〕
E_OACV	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（S）〔NGKI1086〕
E_MACV	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（S）〔NGKI1087〕
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（S）〔NGKI1088〕 ・対象タスクに対してアクセス許可ベクタが設定済み（S）□〔NGKI1089〕

【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〔NGKI1090〕．

静的APIにおいては，tskidはオブジェクト識別名，acptn1~acptn4は整数定数式パラメータである〔NGKI1091〕．

sac_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〔NGKI1092〕．

〔TOPPERS/HRP2カーネルにおける規定〕

HRP2カーネルでは，SAC_TSKのみをサポートする〔HRPS0104〕．ただし，動的生成機能拡張パッケージでは，sac_tskもサポートする〔HRPS0177〕．

```
DEF_EPR      タスクの実行時優先度の定義（S）〔NGKI1093〕
```

〔静的API〕

```
    DEF_EPR(ID tskid, { PRI exepri })
```

〔パラメータ〕

```
    ID          tskid       対象タスクのID番号
    PRI         exepri      タスクの実行時優先度
```

〔エラーコード〕

```
    E_PAR       パラメータエラー
                ・exepriが有効範囲外〔NGKI1094〕
    E_ILUSE     サービスコール不正使用
                ・条件については機能の項を参照
    E_OBJ       オブジェクト状態エラー
                ・対象タスクに対して実行優先度が設定済み〔NGKI1095〕
```

〔サポートするカーネル〕

DEF_EPRは，TOPPERS/SSPカーネルのみがサポートする静的APIである．他のカーネルは，DEF_EPRをサポートしない〔NGKI1096〕．

〔機能〕

tskidで指定したタスク（対象タスク）の実行時優先度を，exepriで指定した優先度に設定する〔NGKI1097〕．

tskidはオブジェクト識別名，exepriは整数定数式パラメータである〔NGKI1098〕．

exepriが，対象タスクの起動時優先度よりも低い場合には，E_ILUSEエラーとなる〔NGKI1099〕．

〔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．

```
del_tsk      タスクの削除（TD）〔NGKI1100〕
```

〔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【NGKI1101】 ・CPUロック状態からの呼出し【NGKI1102】
E_ID	不正ID番号 ・tskidが有効範囲外【NGKI1103】
E_NOEXS	オブジェクト未登録 ・対象タスクが未登録【NGKI1104】
E_OACV	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（P）【NGKI1105】
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【NGKI1106】 ・対象タスクは静的APIで生成された【NGKI1107】

【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【NGKI1108】．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【NGKI1109】．

【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【ASPS0107】．ただし，動的生成 機能拡張パッケージでは，del_tskをサポートする【ASPS0108】．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，del_tskをサポートしない【FMPS0105】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，del_tskをサポートしない【HRPS0105】．ただし，動的生成 機能拡張パッケージでは，del_tskをサポートする【HRPS0178】．

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは，del_tskをサポートしない【SSPS0114】．

act_tsk	タスクの起動（T）【NGKI1110】
iact_tsk	タスクの起動（I）【NGKI1111】

【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）〔NGKI1112〕 ・タスクコンテキストからの呼出し（iact_tskの場合）〔NGKI1113〕 ・CPUロック状態からの呼出し〔NGKI1114〕
E_ID	不正ID番号 ・tskidが有効範囲外〔NGKI1115〕
E_NOEXS	オブジェクト未登録 ・対象タスクが未登録（D）〔NGKI1116〕
E_OACV	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）（P）〔NGKI1117〕
E_QOVR	キューイングオーバフロー ・条件については機能の項を参照

【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〔NGKI1118〕．

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〔NGKI1119〕．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〔NGKI1120〕．

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〔NGKI1121〕．

【補足説明】

マルチプロセッサ対応カーネルでは，act_tsk／iact_tskは，対象タスクの次回起動時の割付けプロセッサを変更しない．

mact_tsk	割付けプロセッサ指定でのタスクの起動（TM）〔NGKI1122〕
imact_tsk	割付けプロセッサ指定でのタスクの起動（IM）〔NGKI1123〕

〔C言語API〕

```
ER ercd = mact_tsk(ID tskid, ID prcid)
ER ercd = imact_tsk(ID tskid, ID prcid)
```

【パラメータ】

ID	tskid	対象タスクのID番号
ID	prcid	タスクの割付け対象のプロセッサの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（mact_tskの場合） 〔NGKI1124〕 ・タスクコンテキストからの呼出し（imact_tskの場合） 〔NGKI1125〕 ・CPUロック状態からの呼出し〔NGKI1126〕
E_NOSPT	未サポート機能 ・対象タスクが制約タスク〔NGKI1127〕
E_ID	不正ID番号 ・tskidが有効範囲外〔NGKI1128〕 ・prcidが有効範囲外〔NGKI1129〕
E_PAR	パラメータエラー ・条件については機能の項を参照
E_NOEXS	オブジェクト未登録 ・対象タスクが未登録（D）〔NGKI1130〕
E_OACV	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（mact_tskの場合）（P）〔NGKI1131〕
E_QOVR	キューイングオーバフロー ・条件については機能の項を参照

【機能】

prcidで指定したプロセッサを割付けプロセッサとして，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，対象タスクの割付けプロセッサが prcidで指定したプロセッサに変更された後，対象タスクに対してタスク起動時

□□□

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〔NGKI1139〕 ・CPUロック状態からの呼出し〔NGKI1140〕
E_ID	不正ID番号 ・tskidが有効範囲外〔NGKI1141〕
E_NOEXS	オブジェクト未登録 ・対象タスクが未登録（D）〔NGKI1142〕
E_OACV	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（P） 〔NGKI1143〕

【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値として返される〔NGKI1144〕．また，マルチプロセッサ対応カーネルにおいては，対象タスクの次回起動時の割付けプロセッサが未設定状態に戻される〔NGKI1145〕．

tskidにTSK_SELF（=0）を指定すると，自タスクが対象タスクとなる〔NGKI1146〕．

〔TOPPERS/SSPカーネルにおける規定〕

SSPカーネルでは，can_actをサポートしない〔SSPS0116〕．

mig_tsk		タスクの割付けプロセッサの変更（TM）〔NGKI1147〕

〔C言語API〕

```
ER ercd = mig_tsk(ID tskid, ID prcid)
```

【パラメータ】

ID	tskid	対象タスクのID番号
ID	prcid	タスクの割付けプロセッサの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【NGKI1148】 ・CPUロック状態からの呼出し【NGKI1149】 ・その他の条件については機能の項を参照
E_NOSPT	未サポート機能 ・対象タスクが制約タスク【NGKI1150】
E_ID	不正ID番号 ・tskidが有効範囲外【NGKI1151】 ・prcidが有効範囲外【NGKI1152】
E_PAR	パラメータエラー ・条件については機能の項を参照
E_NOEXS	オブジェクト未登録 ・対象タスクが未登録（D）【NGKI1153】
E_OACV	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（P） 【NGKI1154】
E_OBJ	オブジェクト状態エラー ・条件については機能の項を参照

【機能】

tskidで指定したタスクの割付けプロセッサを，prcidで指定したプロセッサに 変更する．具体的な振舞いは以下の通り．

対象タスクが，自タスクが割り付けられたプロセッサに割り付けられている場 合には，対象タスクをprcidで指定したプロセッサに割り付ける【NGKI1155】．
対象タスクが実行できる状態の場合には，prcidで指定したプロセッサに割り付 けられた同じ優先度のタスクの中で，最も優先順位が低い状態となる【NGKI1156】．

対象タスクが，自タスクが割付けられたプロセッサと異なるプロセッサに割り 付けられている場合には，E_OBJエラーとなる【NGKI1157】．

tskidにTSK_SELF（=0）を指定すると，自タスクが対象タスクとなる【NGKI1158】．

ディスパッチ保留状態で，対象タスクを自タスクとしてmig_tskを呼び出すと， E_CTXエラーとなる【NGKI1159】．

対象タスクの属するクラスの割付け可能プロセッサが，prcidで指定したプロセッ サを含んでいない場合には，E_PARエラーとなる【NGKI1160】．

prcidにTPRC_INI（=0）を指定すると，対象タスクの割付けプロセッサを，そ れが属するクラスの初期割付けプロセッサに変更する【NGKI1161】．

【補足説明】

この仕様では，タスクをマイグレーションさせることができるのは，そのタス クと同じプロセッサに割り付けられたタスクのみである．そのため，CPUロック状態やディスパッチ禁止状態を用いて，他のタスクへのディスパッチが起こら ないようにすることで，自タスクが他のプロセッサへマイグレーションされる のを防ぐことができる．

対象タスクが，最初からprcidで指定したプロセッサに割り付けられている場合 には，割付けプロセッサの変更は起こらないが，優先順位が同一優先度のタス クの中で最低となる．

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは，mig_tskをサポートしない【ASPS0110】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，mig_tskをサポートしない〔HRPS0107〕．

〔TOPPERS/SSPカーネルにおける規定〕

SSPカーネルでは，mig_tskをサポートしない〔SSPS0117〕．

〔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_tsk      自タスクの終了〔T〕〔NGKI1162〕
```

〔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_SYS       システムエラー
                ・カーネルの誤動作〔NGKI1163〕
    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〔NGKI1164〕
```

【機能】

自タスクを終了させる．具体的には，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る〔NGKI3449〕．

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で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〔NGKI1168〕．

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〔NGKI1169〕．

〔TOPPERS/SSPカーネルにおける規定〕

SSPカーネルでは，ext_tskをサポートしない〔SSPS0118〕．自タスクを終了させる場合には，タスクのメインルーチンからリターンする〔SSPS0119〕．

〔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）〔NGKI1170〕

〔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〔NGKI1171〕 ・CPUロック状態からの呼出し〔NGKI1172〕
E_ID	不正ID番号 ・tskidが有効範囲外〔NGKI1173〕
E_NOEXS	オブジェクト未登録 ・対象タスクが未登録（D）〔NGKI1174〕
E_OACV	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（P） 〔NGKI1175〕
E_ILUSE	サービスコール不正使用 ・対象タスクが自タスク〔NGKI1176〕
E_OBJ	オブジェクト状態エラー ・対象タスクが休止状態〔NGKI1177〕 ・その他の条件については機能の項を参照

【機能】

tskidで指定したタスク（対象タスク）を終了させる．具体的には，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る〔NGKI3450〕．

マルチプロセッサ対応カーネルでは，対象タスクは，自タスクと同じプロセッサに割り付けられているタスクに限られる．対象タスクが自タスクと異なるプロセッサに割り付けられている場合には，E_OBJエラーとなる〔NGKI1182〕．

〔TOPPERS/FMPカーネルにおける使用上の注意〕

現時点のFMPカーネルの実装では，デッドロック回避のためのリトライ処理により，サービスコールの処理時間に上限がないため，注意が必要である（ロック方式にも依存する）．

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは，ter_tskをサポートしない【SSPS0120】．

chg_pri	タスクのベース優先度の変更（T）【NGKI1183】

【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【NGKI1184】 ・CPUロック状態からの呼出し【NGKI1185】
E_NOSPT	未サポート機能 ・対象タスクが制約タスク【NGKI1186】
E_ID	不正ID番号 ・tskidが有効範囲外【NGKI1187】
E_PAR	パラメータエラー ・tskpriが有効範囲外【NGKI1188】
E_NOEXS	オブジェクト未登録 ・対象タスクが未登録（D）【NGKI1189】
E_OACV	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（P） 【NGKI1190】
E_ILUSE	サービスコール不正使用 ・条件については機能の項を参照
E_OBJ	オブジェクト状態エラー ・対象タスクが休止状態【NGKI1191】

【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【NGKI1192】．それに伴って，対象タスクの現

在優先度も変更される〔NGKI1193〕．

対象タスクが，優先度上限ミューテックスをロックしていない場合には，次の処理が行われる．対象タスクが実行できる状態の場合には，同じ優先度のタスクの中で最低優先順位となる〔NGKI1194〕．対象タスクが待ち状態で，タスクの優先度順の待ち行列につながれている場合には，対象タスクの変更後の現在優先度に従って，その待ち行列中での順序が変更される〔NGKI1195〕．待ち行列中に同じ現在優先度のタスクがある場合には，対象タスクの順序はそれらの中で最後になる〔NGKI1196〕．

対象タスクが，優先度上限ミューテックスをロックしている場合には，対象タスクの現在優先度が変更されることはなく，優先順位も変更されない〔NGKI1197〕．

tskidにTSK_SELF（=0）を指定すると，自タスクが対象タスクとなる〔NGKI1198〕．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〔NGKI1199〕．

対象タスクが優先度上限ミューテックスをロックしているかロックを待っている場合，tskpriは，それらのミューテックスの上限優先度と同じかそれより低くなければならない．そうでない場合には，E_ILUSEエラーとなる〔NGKI1201〕．

保護機能対応カーネルで，chg_priを呼び出した処理単位がユーザドメインに属する場合，tskpriは，そのユーザドメインが指定できる最高のタスク優先度と同じかそれより低くなければならない．そうでない場合には，E_ILUSEエラーとなる〔NGKI3440〕．

〔TOPPERS/SSPカーネルにおける規定〕

SSPカーネルでは，chg_priをサポートしない〔SSPS0121〕．

〔μITRON4.0仕様との関係〕

対象タスクが，同じ優先度のタスクの中で最低の優先順位となる（対象タスクが待ち状態で，タスクの優先度順の待ち行列につながれている場合には，同じ優先度のタスクの中での順序が最後になる）条件を変更した．

get_pri	タスク優先度の参照（T）〔NGKI1202〕

〔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〔NGKI1203〕 ・CPUロック状態からの呼出し〔NGKI1204〕
E_ID	不正ID番号 ・tskidが有効範囲外〔NGKI1205〕
E_NOEXS	オブジェクト未登録 ・対象タスクが未登録（D）〔NGKI1206〕
E_OACV	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（P）〔NGKI1207〕
E_MACV	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（P）〔NGKI1208〕
E_OBJ	オブジェクト状態エラー ・対象タスクが休止状態〔NGKI1209〕

【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 が指すメモリ領域に返される〔NGKI1210〕．

tskidにTSK_SELF（=0）を指定すると，自タスクが対象タスクとなる〔NGKI1211〕．

〔TOPPERS/SSPカーネルにおける規定〕

SSPカーネルでは，get_priをサポートしない〔SSPS0122〕．

get_inf	自タスクの拡張情報の参照（T）〔NGKI1212〕

〔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〔NGKI1213〕 ・CPUロック状態からの呼出し〔NGKI1214〕
E_MACV		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〔P〕〔NGKI1215〕

【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が指すメモリ領域に返される〔NGKI1216〕．

〔TOPPERS/SSPカーネルにおける規定〕

SSPカーネルでは，get_infをサポートしない〔SSPS0123〕．

〔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〕〔NGKI1217〕

〔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
bool_t	texmsk	タスクがタスク例外処理マスク状態か否か（保護機能対応カーネルの場合）
bool_t	waifbd	タスクが待ち禁止状態か否か（保護機能対応カーネルの場合）
uint_t	svclevel	タスクの拡張サービスコールのネストレベル（保護機能対応カーネルの場合）
ID	prcid	タスクの割付けプロセッサのID（マルチプロセッサ対応カーネルの場合）
ID	actprc	タスクの次回起動時の割付けプロセッサのID（マルチプロセッサ対応カーネルの場合）

【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【NGKI1218】 ・CPUロック状態からの呼出し【NGKI1219】
E_ID	不正ID番号 ・tskidが有効範囲外【NGKI1220】
E_NOEXS	オブジェクト未登録 ・対象タスクが未登録（D）【NGKI1221】
E_OACV	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（P）【NGKI1222】
E_MACV	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（P）【NGKI1223】

【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【NGKI1224】．

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【NGKI1225】．

```
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
```

マルチプロセッサ対応カーネルでは，対象タスクが自タスクの場合にも，tskstatがTTS_SUSとなる場合がある【NGKI1226】．この状況は，自タスクに対してref_tskを発行するのと同じタイミングで，他のプロセッサで実行されている タスクから同じタスクに対してsus_tskが発行された場合に発生する可能性が ある．

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【NGKI1227】．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【NGKI1228】．

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【NGKI1229】．

```
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_MBX     0x0040U     メールボックスからの受信待ち
TTW_MTX     0x0080U     ミューテックスのロック待ち状態
TTW_SMBF    0x0400U     メッセージバッファへの送信待ち
TTW_RMBF    0x0800U     メッセージバッファからの受信待ち
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【NGKI1230】．

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【NGKI1231】．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【NGKI1232】．

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【NGKI1233】．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【NGKI1234】．

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【NGKI1235】．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【NGKI1236】．

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【NGKI1237】．

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【NGKI1238】．

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【NGKI1239】．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【NGKI1240】．

保護機能対応カーネルで，対象タスクが休止状態でない場合には，texmskに，対象タスクがタスク例外処理マスク状態の場合にtrue，そうでない場合に

falseが返される【NGKI1241】．waifbdには，対象タスクが待ち禁止状態の場合にtrue，そうでない場合にfalseが返される【NGKI1242】．またsvclevelには，対象タスクが拡張サービスコールを呼び出していない場合には0，呼び出している場合には，実行中の拡張サービスコールがネスト段数が返される【NGKI1243】．対象タスクが休止状態である場合には，texmsk，waifbd，svclevelの値は保証されない【NGKI1244】．

マルチプロセッサ対応カーネルでは，prcidに，対象タスクの割付けプロセッサのID番号が返される【NGKI1245】．またactprcには，対象タスクの次回起動時の割付けプロセッサのID番号が返される【NGKI1246】．次回起動時の割付けプロセッサが未設定の場合には，actprcにTPRC_NONE（=0）が返される【NGKI1247】．

tskidにTSK_SELF（=0）を指定すると，自タスクが対象タスクとなる【NGKI1248】．

【補足説明】

対象タスクが時間経過待ち状態である場合に，lefttmo（TMO型）に返される値をRELTIM型に型キャストすることで，タスクが待ち解除されるまでの相対時間 を正しく得ることができる．

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは，tskwaitにTTW_MTX，TTW_SMBF，TTW_RMBFが返ることはない【ASPS0111】．ただし，ミューテックス機能拡張パッケージを用いると，tskwaitにTTW_MTXが返る場合がある【ASPS0112】．また，メッセージバッファ機能拡張パッケージを用いると，tskwaitにTTW_SMBFとTTW_RMBFが返る場合がある【ASPS0208】．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，tskwaitにTTW_MTX，TTW_SMBF，TTW_RMBFが返ることはない【FMPS0106】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，tskwaitにTTW_MBX，TTW_SMBF，TTW_RMBFが返ることはない【HRPS0108】．ただし，メッセージバッファ機能拡張パッケージを用いると，tskwaitにTTW_SMBFとTTW_RMBFが返る場合がある【HRPS0174】．

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは，ref_tskをサポートしない【SSPS0124】．

【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．

マルチプロセッサ対応カーネルで参照できる情報として，割付けプロセッサの ID（prcid）と次回起動時の割付けプロセッサのID（actprc）を追加した．

【μITRON4.0/PX仕様との関係】

保護機能対応カーネルで参照できる情報として，タスク例外処理マスク状態か否か（texmsk），待ち禁止状態か否か（waifbd），拡張サービスコールのネストレベル（svclevel）を追加した．

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【NGKI1249】．

- 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【NGKI1250】．

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【NGKI1251】
```

【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【ASPS0113】．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，TMAX_WUPCNTは1に固定されている【FMPS0107】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，TMAX_WUPCNTは1に固定されている【HRPS0109】．

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは，タスク付属同期機能をサポートしない【SSPS0125】．

【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）とタスクの強制待ち 要求ネスト数の最大値を表すカーネル構成マクロ（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）【NGKI1252】
tslp_tsk     起床待ち（タイムアウト付き）（T）【NGKI1253】
```

【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 ・ディスパッチ保留状態からの呼出し［NGKI1254］
E_NOSPT	未サポート機能 ・制約タスクからの呼出し［NGKI1255］
E_PAR	パラメータエラー ・tmoutが無効（tslp_taskの場合）［NGKI1256］
E_TMOUT	ポーリング失敗またはタイムアウト（slp_taskを除く）［NGKI1257］
E_RLWAI	待ち禁止状態または待ち状態の強制解除［NGKI1258］

【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［NGKI1259］．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［NGKI1260］．

【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）［NGKI1261］
iwup_tsk	タスクの起床（I）［NGKI1262］

［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_taskの場合）〔NGKI1263〕 ・タスクコンテキストからの呼出し（iwup_taskの場合）〔NGKI1264〕 ・CPUロック状態からの呼出し〔NGKI1265〕
E_NOSPT	未サポート機能 ・対象タスクが制約タスク〔NGKI1266〕
E_ID	不正ID番号 ・tskidが有効範囲外〔NGKI1267〕
E_NOEXS	オブジェクト未登録 ・対象タスクが未登録（D）〔NGKI1268〕
E_OACV	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_task の場合）（P）〔NGKI1269〕
E_OBJ	オブジェクト状態エラー ・対象タスクが休止状態〔NGKI1270〕
E_QOVR	キューイングオーバフロー ・条件については機能の項を参照

【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〔NGKI1271〕．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〔NGKI1272〕．

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〔NGKI1273〕．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〔NGKI1274〕．

wup_task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〔NGKI1275〕．

can_wup	タスク起床要求のキャンセル（T）〔NGKI1276〕

〔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【NGKI1277】 ・CPUロック状態からの呼出し【NGKI1278】
E_NOSPT	未サポート機能 ・対象タスクが制約タスク【NGKI1279】
E_ID	不正ID番号 ・tskidが有効範囲外【NGKI1280】
E_NOEXS	オブジェクト未登録 ・対象タスクが未登録（D）【NGKI1281】
E_OACV	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（P） 【NGKI1282】
E_OBJ	オブジェクト状態エラー ・対象タスクが休止状態【NGKI1283】

【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【NGKI1284】．

tskidにTSK_SELF（=0）を指定すると，自タスクが対象タスクとなる【NGKI1285】．

rel_wai	強制的な待ち解除（T）【NGKI1286】
irel_wai	強制的な待ち解除（I）【NGKI1287】

【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）【NGKI1288】 ・タスクコンテキストからの呼出し（irel_waiの場合）【NGKI1289】 ・CPUロック状態からの呼出し【NGKI1290】
E_NOSPT	未サポート機能 ・対象タスクが制約タスク【NGKI1291】
E_ID	不正ID番号 ・tskidが有効範囲外【NGKI1292】
E_NOEXS	オブジェクト未登録 ・対象タスクが未登録（D）【NGKI1293】
E_OACV	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）（P）【NGKI1294】
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【NGKI1295】

【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【NGKI1296】．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【NGKI1297】．

sus_tsk	強制待ち状態への遷移（T）【NGKI1298】

【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［NGKI1299］ ・CPUロック状態からの呼出し［NGKI1300］ ・その他の条件については機能の項を参照
E_NOSPT	未サポート機能 ・対象タスクが制約タスク［NGKI1301］
E_ID	不正ID番号 ・tskidが有効範囲外［NGKI1302］
E_NOEXS	オブジェクト未登録 ・対象タスクが未登録（D）［NGKI1303］
E_OACV	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（P） ［NGKI1304］
E_OBJ	オブジェクト状態エラー ・対象タスクが休止状態［NGKI1305］
E_QOVR	キューイングオーバフロー ・対象タスクが強制待ち状態（二重待ち状態を含む）［NGKI1306］

【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［NGKI1307］．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［NGKI1308］．

マルチプロセッサ対応カーネルでは，対象タスクが自タスクの場合にも，E_QOVRエラーとなる場合がある［NGKI1309］．この状況は，自タスクに対してsus_tskを発行するのと同じタイミングで，他のプロセッサで実行されているタスクから同じタスクに対してsus_tskが発行された場合に発生する可能性がある．

tskidにTSK_SELF（=0）を指定すると，自タスクが対象タスクとなる［NGKI1310］．

ディスパッチ保留状態で，対象タスクを自タスクとしてsus_tskを呼び出すと，E_CTXエラーとなる［NGKI1311］．なお，sus_tsk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であるが，対象タスクが自タスクでない場合には，割込み優先度マスクが全解除でない状態やディスパッチ禁止状態で呼び出しても，E_CTXエラーにはならない［NGKI3604］．これは，［NGKI0175］と［NGKI0179］の原則の例外となっている．

rsm_tsk		強制待ち状態からの再開（T）［NGKI1312］

［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【NGKI1313】
            ・CPUロック状態からの呼出し【NGKI1314】
E_NOSPT     未サポート機能
            ・対象タスクが制約タスク【NGKI1315】
E_ID        不正ID番号
            ・tskidが有効範囲外【NGKI1316】
E_NOEXS     オブジェクト未登録
            ・対象タスクが未登録（D）【NGKI1317】
E_OACV      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（P）
            【NGKI1318】
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（二重待ち状態を含む）でない
            【NGKI1319】
```

【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【NGKI1320】．

```
dis_wai     待ち禁止状態への遷移（TP）【NGKI1321】
idis_wai    待ち禁止状態への遷移（IP）【NGKI1322】
```

【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）【NGKI1323】 ・タスクコンテキストからの呼出し（idis_waiの場合）【NGKI1324】 ・CPUロック状態からの呼出し【NGKI1325】
E_NOSPT	未サポート機能 ・対象タスクが制約タスク【NGKI1326】
E_ID	不正ID番号 ・tskidが有効範囲外【NGKI1327】
E_NOEXS	オブジェクト未登録 ・対象タスクが未登録（D）【NGKI1328】
E_OACV	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）【NGKI1329】
E_OBJ	オブジェクト状態エラー ・対象タスクが休止状態【NGKI1330】 ・対象タスクがタスク例外処理マスク状態でない【NGKI1331】
E_QOVR	キューイングオーバフロー ・対象タスクが待ち禁止状態【NGKI1332】

【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．

対象タスクがタスク例外処理マスク状態であり，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【NGKI1333】．

dis_wai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【NGKI1334】．

【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【ASPS0114】．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，dis_waiをサポートしない【FMPS0108】．

【補足説明】

dis_waiは，対象タスクの待ち解除は行わない．対象タスクを待ち禁止状態にす ることに加えて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．

【未決定事項】

マルチプロセッサ対応カーネルでは，対象タスクを，自タスクと同じプロセッ サに割り付けられているタスクに限るなどの制限を導入する可能性があるが，現時点では未決定である．

【μITRON4.0/PX仕様との関係】

μITRON4.0/PX仕様に定義されていないサービスコールである．

ena_wai	待ち禁止状態の解除（TP）【NGKI1335】
iena_wai	待ち禁止状態の解除（IP）【NGKI1336】

【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）【NGKI1337】 ・タスクコンテキストからの呼出し（iena_waiの場合）【NGKI1338】 ・CPUロック状態からの呼出し【NGKI1339】
E_NOSPT	未サポート機能 ・対象タスクが制約タスク【NGKI1340】
E_ID	不正ID番号 ・tskidが有効範囲外【NGKI1341】
E_NOEXS	オブジェクト未登録 ・対象タスクが未登録（D）【NGKI1342】
E_OACV	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）【NGKI1343】
E_OBJ	オブジェクト状態エラー ・対象タスクが休止状態【NGKI1344】 ・対象タスクが待ち禁止状態でない【NGKI1345】

【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【NGKI1346】．

ena_wai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【NGKI1347】．

【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【ASPS0115】．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，ena_waiをサポートしない【FMPS0109】．

【未決定事項】

マルチプロセッサ対応カーネルでは，対象タスクを，自タスクと同じプロセッサに割り付けられているタスクに限るなどの制限を導入する可能性があるが，現時点では未決定である．

【μITRON4.0/PX仕様との関係】

μITRON4.0/PX仕様に定義されていないサービスコールである．

dly_tsk	自タスクの遅延（T）【NGKI1348】	

【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 ・ディスパッチ保留状態からの呼出し【NGKI1349】
E_NOSPT	未サポート機能 ・制約タスクからの呼出し【NGKI1350】
E_PAR	パラメータエラー ・dlytimがTMAX_RELTIMより大きい【NGKI1351】
E_RLWAI	待ち禁止状態または待ち状態の強制解除【NGKI1352】

【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【NGKI1353】．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【NGKI1354】．

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【NGKI1355】．

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【NGKI1356】．

- 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【NGKI1357】．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【NGKI1358】．

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【NGKI1359】．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【NGKI1361】．

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【NGKI1362】．

タスク例外処理ルーチンは，「タスク例外処理許可状態である」「保留例外要因が0でない」「タスクが実行状態である」「タスクコンテキストが実行されている」「割込み優先度マスク全解除状態である」「CPUロック状態でない」の6つの条件が揃った場合に実行が開始される【NGKI1363】．保護機能対応カーネルにおいては，さらに，「タスク例外処理マスク状態でない」という条件が追加される【NGKI1364】．タスク例外処理マスク状態については，「2.6.5 タスク例外処理マスク状態と待ち禁止状態」の節を参照すること．

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【NGKI1365】．また，タスク例外処理ルーチンからのリターン時には，タスク例外処理禁止フラグはクリアされる【NGKI1366】．

保護機能対応カーネルでは，ユーザタスクのタスク例外処理ルーチンの実行開始時に，リターン先の番地やシステム状態等が，ユーザスタック上に保存される【NGKI1367】．ここで，ユーザスタック領域に十分な空きがない場合や，ユーザスタックポインタがユーザスタック領域以外を指している場合，カーネルは，エミュレートされたCPU例外を発生させる【NGKI1368】．これを，タスク例外実行開始時スタック不正例外と呼ぶ．

逆に，タスク例外処理ルーチンからのリターン時には，リターン先の番地やシステム状態等が，ユーザスタック上から取り出される【NGKI1369】．ここで，ユーザスタック領域に積まれている情報が足りない場合や，ユーザスタックポインタがユーザスタック領域以外を指している場合，カーネルは，エミュレートされたCPU例外を発生させる【NGKI1370】．これを，タスク例外リターン時スタック不正例外と呼ぶ．

タスク例外実行開始時スタック不正例外またはタスク例外リターン時スタック不正例外を起こしたタスクの実行を継続した場合の動作は保証されないため，アプリケーションは，これらのCPU例外を処理するCPU例外ハンドラで，「2.8.1 CPU例外処理の流れ」の節の(b)または(d)の方法でリカバリ処理を行う必要がある【NGKI1371】．この方法に従わなかった場合の動作は，保証されない【NGKI1372】．

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は，タスクと同じ保護ドメインに属する【NGKI1373】．

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）【NGKI1374】

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【NGKI1375】．

```
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【NGKI1376】．

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）
                〔NGKI1377〕
```

【補足説明】

保護機能対応でないカーネルでは，タスク例外処理ルーチンの実行開始条件の内，「CPUロック状態でない」は省いても同じ結果になる．これは，CPUロック状態で他の条件が揃うことはないためである．一方，保護機能対応カーネルでは，CPUロック状態で拡張サービスコールからリターンした場合（言い換えると，タスク例外処理マスク状態が解除された場合）に，CPUロック状態で他の条件が揃うことになる．

〔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〔ASPS0116〕．

〔TOPPERS/FMPカーネルにおける規定〕

FMP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〔FMPS0110〕．

〔TOPPERS/HRP2カーネルにおける規定〕

HRP2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〔HRPS0110〕．

〔TOPPERS/SSPカーネルにおける規定〕

SSPカーネルでは，タスク例外処理機能をサポートしない〔SSPS0126〕．

〔μITRON4.0仕様との関係〕

割込み優先度マスク全解除状態でない場合には，タスク例外処理ルーチンの実行が開始されないという仕様に変更した．

〔μITRON4.0/PX仕様との関係〕

ユーザタスクのタスク例外処理ルーチンの実行開始時とリターン時にユーザスタックが不正となる問題に関して，μITRON4.0/PX仕様では考慮されていない．

【仕様変更の経緯】

この仕様のRelease 1.2以前では，タスク例外処理ルーチンの実行開始条件に「割込み優先度マスク全解除状態である」の条件がなかったが，Release1.3以降で追加した．これは，マルチプロセッサ対応カーネルにおいて，他プロセッサで実行中のタスクに対してタスク例外処理を要求した場合に，割込み優先度マスクが全解除でないと，タスク例外処理ルーチンをただちに実行開始することができないためである．なお，ASPカーネル Release 1.6以前と，FMPカーネル Release 1.1.1以前のバージョンは，古い仕様に従って実装されている．

```
DEF_TEX     タスク例外処理ルーチンの定義（S）〔NGKI1378〕
def_tex     タスク例外処理ルーチンの定義（TD）〔NGKI1379〕
```

【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〔S〕【NGKI1380】 ・CPUロック状態からの呼出し〔S〕【NGKI1381】
E_ID	不正ID番号 ・tskidが有効範囲外〔S〕【NGKI1382】
E_RSATR	予約属性 ・texatrが無効【NGKI1383】 ・その他の条件については機能の項を参照
E_PAR	パラメータエラー ・texrtnがプログラムの先頭番地として正しくない【NGKI1384】
E_NOEXS	オブジェクト未登録 ・対象タスクが未登録【NGKI1385】
E_OACV	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〔SP〕□【NGKI1386】
E_MACV	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〔SP〕【NGKI1387】
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〔S〕【NGKI1388】 ・その他の条件については機能の項を参照

【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【NGKI1389】.

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【NGKI1390】．また，対象タスクのタスク例外処理禁止フラグをセットし，保留例外要因を0に初期化する【NGKI1391】．

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【NGKI1392】．

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【NGKI1393】．

保護機能対応カーネルにおいて，DEF_TEXは，対象タスクが属する保護ドメインの囲みの中に記述しなければならない．そうでない場合には，E_RSATRエラーとなる【NGKI1395】．また，def_texでタスク例外処理ルーチンを定義する場合には，タスク例外処理ルーチンの属する保護ドメインを設定する必要はなく，タスク例外処理ルーチン属性にTA_DOM(domid)を指定した場合にはE_RSATRエラーとなる【NGKI1396】．ただし，TA_DOM(TDOM_SELF)を指定した場合には，指定が無視され，E_RSATRエラーは検出されない【NGKI1397】．

マルチプロセッサ対応カーネルにおいて，DEF_TEXは，対象タスクが属するクラスの囲みの中に記述しなければならない．そうでない場合には，E_RSATRエラーとなる【NGKI1399】．また，def_texでタスク例外処理ルーチンを定義する場合には，タスク例外処理ルーチンの属するクラスを設定する必要はなく，タスク例外処理ルーチン属性にTA_CLS(clsid)を指定した場合にはE_RSATRエラーとなる【NGKI1400】．ただし，TA_CLS(CLS_SELF)を指定した場合には，指定が無視され，E_RSATRエラーは検出されない【NGKI1401】．

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【NGKI1402】．

def_tex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【NGKI1403】．

【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【ASPS0117】．ただし，動的生成機能拡張パッケージでは，def_texもサポートする【ASPS0118】．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，DEF_TEXのみをサポートする【FMPS0111】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，DEF_TEXのみをサポートする【HRPS0111】．ただし，動的生成機能拡張パッケージでは，def_texもサポートする【HRPS0179】．

【μITRON4.0仕様との関係】

texrtnのデータ型をTEXRTN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）【NGKI1404】
iras_tex     タスク例外処理の要求（I）【NGKI1405】
```

【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）【NGKI1406】
            ・タスクコンテキストからの呼出し（iras_texの場合）【NGKI1407】
            ・CPUロック状態からの呼出し【NGKI1408】
E_ID        不正ID番号
            ・tskidが有効範囲外【NGKI1409】
E_PAR       パラメータエラー
            ・rasptnが0【NGKI1410】
E_NOEXS     オブジェクト未登録
            ・対象タスクが未登録（D）【NGKI1411】
E_OACV      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）（P）【NGKI1412】
E_OBJ       オブジェクト状態エラー
            ・対象タスクが休止状態【NGKI1413】
            ・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【NGKI1414】
```

【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【NGKI1415】．

ras_tex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【NGKI1416】．

```
dis_tex     タスク例外処理の禁止（T）【NGKI1417】
```

【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［NGKI1419］ ・CPUロック状態からの呼出し［NGKI1420］
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［NGKI1421］

【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［NGKI1422］．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）［NGKI1423］

［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［NGKI1424］ ・CPUロック状態からの呼出し［NGKI1425］
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［NGKI1426］

【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［NGKI1427］．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．

【補足説明】

タスク例外処理ルーチン中でena_texを呼び出すことにより，タスク例外処理ルーチンの多重起動を行うことができる．ただし，多重起動の最大段数を制限するのは，アプリケーションの責任である．

sns_tex	タスク例外処理禁止状態の参照〔TI〕[NGKI1428]	

[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[NGKI1429]．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[NGKI1430]．

マルチプロセッサ対応カーネルにおいては，サービスコールを呼び出した処理単位を実行しているプロセッサにおいて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[NGKI1431]．

【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〕[NGKI1432]	

[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【NGKI1433】
                ・CPUロック状態からの呼出し【NGKI1434】
    E_ID        不正ID番号
                ・tskidが有効範囲外【NGKI1435】
    E_NOEXS     オブジェクト未登録
                ・対象タスクが未登録（D）【NGKI1436】
    E_OACV      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（P）【NGKI1437】
    E_MACV      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（P）【NGKI1438】
    E_OBJ       オブジェクト状態エラー
                ・対象タスクが休止状態【NGKI1439】
                ・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【NGKI1440】
```

【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【NGKI1441】．

texstat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【NGKI1442】．

```
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
```

pndptnには，対象タスクの現在の保留例外要因が返される【NGKI1443】．

tskidにTSK_SELF（=0）を指定すると，自タスクが対象タスクとなる【NGKI1444】．

4.4. 同期・通信機能

同期・通信機能は，タスクとは独立したオブジェクトにより，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・通信を行うための機能である．

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは，同期・通信機能をサポートしない【SSPS0127】．

【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同期・通信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【NGKI1445】．

各セマフォが持つ情報は次の通り【NGKI1446】．

- 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
- 最大資源数
- アクセス許可ベクタ（保護機能対応カーネルの場合）
- 属する保護ドメイン（保護機能対応カーネルの場合）
- 属するクラス（マルチプロセッサ対応カーネルの場合）

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と，E_QOVRエラーとなる【NGKI1447】．

セマフォ属性には，次の属性を指定することができる【NGKI1448】．

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【NGKI1449】．

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（=UINT_MAX）【NGKI1450】

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）【NGKI1451】

【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〕〔NGKI1452〕
acre_sem     セマフォの生成〔TD〕〔NGKI1453〕
```

【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（CRE_SEMの場合）
    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（s）【NGKI1454】 ・CPUロック状態からの呼出し（s）【NGKI1455】
E_RSATR	予約属性 ・sematrが無効【NGKI1456】 ・属する保護ドメインの指定が有効範囲外（sP）【NGKI1457】 ・属するクラスの指定が有効範囲外（sM）【NGKI1458】 ・クラスの囲みの中に記述されていない（SM）【NGKI1459】
E_PAR	パラメータエラー ・maxsemが有効範囲（1以上TMAX_MAXSEM以下）外【NGKI1468】 ・isemcntが有効範囲（0以上maxsem以下）外【NGKI1466】
E_OACV	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（sP） 【NGKI1460】
E_MACV	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（sP）【NGKI1461】
E_NOID	ID番号不足 ・割り付けられるセマフォIDがない（sD）【NGKI1462】
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（CRE_SEMの場合） 【NGKI1463】

【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【NGKI1464】．

静的APIにおいては，semidはオブジェクト識別名，sematr，isemcnt，maxsemは整数定数式パラメータである【NGKI1465】．

【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【ASPS0119】．ただし，動的生成機能拡張パッケージでは，acre_semもサポートする【ASPS0120】．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，CRE_SEMのみをサポートする【FMPS0112】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，CRE_SEMのみをサポートする【HRPS0112】．ただし，動的生成機能拡張パッケージでは，acre_semもサポートする【HRPS0180】．

AID_SEM	割付け可能なセマフォIDの数の指定（SD）【NGKI1469】

【静的API】

```
AID_SEM(uint_t nosem)
```

【パラメータ】

```
uint_t      nosem       割付け可能なセマフォIDの数
```

【エラーコード】

```
E_RSATR     予約属性
            ・保護ドメインの囲みの中に記述されている（P）〔NGKI3429〕
            ・クラスの囲みの中に記述されていない（M）〔NGKI1470〕
E_PAR       パラメータエラー
            ・nosemが負の値〔NGKI3277〕
```

【機能】

nosemで指定した数のセマフォIDを，セマフォを生成するサービスコールによって割付け可能なセマフォIDとして確保する〔NGKI1471〕．

nosemは整数定数式パラメータである〔NGKI1472〕．

〔TOPPERS/ASPカーネルにおける規定〕

ASPカーネルの動的生成機能拡張パッケージでは，AID_SEMをサポートする〔ASPS0211〕．

〔TOPPERS/HRP2カーネルにおける規定〕

HRP2カーネルの動的生成機能拡張パッケージでは，AID_SEMをサポートする〔HRPS0212〕．

```
SAC_SEM     セマフォのアクセス許可ベクタの設定（SP）〔NGKI1473〕
sac_sem     セマフォのアクセス許可ベクタの設定（TPD）〔NGKI1474〕
```

【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（S）〔NGKI1475〕 ・CPUロック状態からの呼出し（S）〔NGKI1476〕
E_ID	不正ID番号 ・semidが有効範囲外（S）〔NGKI1477〕
E_RSATR	予約属性 ・対象セマフォが属する保護ドメインの囲みの中（対象セマフォが無所属の場合は，保護ドメインの囲みの外）に記述されていない（S）〔NGKI1478〕 ・対象セマフォが属するクラスの囲みの中に記述されていない（SM）〔NGKI1479〕
E_NOEXS	オブジェクト未登録 ・対象セマフォが未登録〔NGKI1480〕
E_OACV	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（S）〔NGKI1481〕
E_MACV	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（S）〔NGKI1482〕
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（S）〔NGKI1483〕 ・対象セマフォに対してアクセス許可ベクタが設定済み（S）〔NGKI1484〕

【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〔NGKI1485〕．

静的APIにおいては，semidはオブジェクト識別名，acptn1~acptn4は整数定数式パラメータである〔NGKI1486〕．

〔TOPPERS/HRP2カーネルにおける規定〕

HRP2カーネルでは，SAC_SEMのみをサポートする〔HRPS0113〕．ただし，動的生成機能拡張パッケージでは，sac_semもサポートする〔HRPS0181〕．

del_sem	セマフォの削除〔TD〕〔NGKI1487〕	

〔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〔NGKI1488〕 ・CPUロック状態からの呼出し〔NGKI1489〕
E_ID	不正ID番号 ・semidが有効範囲外〔NGKI1490〕
E_NOEXS	オブジェクト未登録 ・対象セマフォが未登録〔NGKI1491〕
E_OACV	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（P） 〔NGKI1492〕
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〔NGKI1493〕

【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〔NGKI1494〕．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〔NGKI1495〕．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〔NGKI1496〕．

【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_semをサポートしない〔ASPS0122〕．ただし，動的生成機能拡張パッケージでは，del_semをサポートする〔ASPS0123〕．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，del_semをサポートしない【FMPS0114】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，del_semをサポートしない【HRPS0114】．ただし，動的生成機能拡張パッケージでは，del_semをサポートする【HRPS0182】．

```
sig_sem      セマフォの資源の返却（T）【NGKI1497】
isig_sem     セマフォの資源の返却（I）【NGKI1498】
```

【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）【NGKI1499】
                ・タスクコンテキストからの呼出し（isig_semの場合）【NGKI1500】
                ・CPUロック状態からの呼出し【NGKI1501】
    E_ID        不正ID番号
                ・semidが有効範囲外【NGKI1502】
    E_NOEXS     オブジェクト未登録
                ・対象セマフォが未登録（D）【NGKI1503】
    E_OACV      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）（P）【NGKI1504】
    E_QOVR      キューイングオーバフロー
                ・条件については機能の項を参照
```

【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【NGKI1505】．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【NGKI1506】．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【NGKI1507】．

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【NGKI1508】．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【NGKI1509】．

wai_sem	セマフォの資源の獲得（T）【NGKI1510】
pol_sem	セマフォの資源の獲得（ポーリング）（T）【NGKI1511】
twai_sem	セマフォの資源の獲得（タイムアウト付き）（T）【NGKI1512】

【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〔NGKI1513〕 ・CPUロック状態からの呼出し〔NGKI1514〕 ・ディスパッチ保留状態からの呼出し（pol_semを除く）〔NGKI1515〕
E_NOSPT	未サポート機能 ・制約タスクからの呼出し（pol_semを除く）〔NGKI1516〕
E_ID	不正ID番号 ・semidが有効範囲外〔NGKI1517〕
E_PAR	パラメータエラー ・tmoutが無効（twai_semの場合）〔NGKI1518〕
E_NOEXS	オブジェクト未登録 ・対象セマフォが未登録（D）〔NGKI1519〕
E_OACV	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（P） 〔NGKI1520〕
E_TMOUT	ポーリング失敗またはタイムアウト（wai_semを除く）〔NGKI1521〕
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） 〔NGKI1522〕
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） 〔NGKI1523〕

【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〔NGKI1524〕．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れる〔NGKI1525〕．

ini_sem	セマフォの再初期化（T）〔NGKI1526〕

〔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〔NGKI1527〕 ・CPUロック状態からの呼出し〔NGKI1528〕
E_ID	不正ID番号 ・semidが有効範囲外〔NGKI1529〕
E_NOEXS	オブジェクト未登録 ・対象セマフォが未登録（D）〔NGKI1530〕
E_OACV	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（P） 〔NGKI1531〕

【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〔NGKI1532〕．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〔NGKI1533〕．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〔NGKI1534〕．

【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）〔NGKI1535〕

〔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【NGKI1536】
                ・CPUロック状態からの呼出し【NGKI1537】
    E_ID        不正ID番号
                ・semidが有効範囲外【NGKI1538】
    E_NOEXS     オブジェクト未登録
                ・対象セマフォが未登録（D）【NGKI1539】
    E_OACV      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（P）
                □【NGKI1540】
    E_MACV      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（P）【NGKI1541】
```

【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【NGKI1542】．

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【NGKI1543】．

【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同期・通信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【NGKI1544】．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【NGKI1545】．

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い，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（=0x01U）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【NGKI1546】．待ちモードにTWF_ANDW（=0x02U）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【NGKI1547】．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【NGKI1548】．

- 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
- アクセス許可ベクタ（保護機能対応カーネルの場合）
- 属する保護ドメイン（保護機能対応カーネルの場合）
- 属するクラス（マルチプロセッサ対応カーネルの場合）

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）のタスクがつな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（〔NGKI0216〕に該当）〔NGKI1549〕．

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〔NGKI1550〕．

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〔NGKI1551〕．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〔NGKI1552〕．

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〔NGKI1553〕．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〔NGKI1554〕．

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）〔NGKI1555〕

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）〔NGKI1556〕

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）〔NGKI1557〕

〔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〔ASPS0124〕．

〔TOPPERS/FMPカーネルにおける規定〕

FMP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【FMPS0115】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2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【HRPS0115】．

【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）【NGKI1558】
acre_flg     イベントフラグの生成（TD）【NGKI1559】
```

【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（CRE_FLGの
                            場合）
    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（S）【NGKI1560】 ・CPUロック状態からの呼出し（S）【NGKI1561】
E_RSATR	予約属性 ・flgatrが無効【NGKI1562】 ・属する保護ドメインの指定が有効範囲外（SP）【NGKI1563】 ・属するクラスの指定が有効範囲外（SM）【NGKI1564】 ・クラスの囲みの中に記述されていない（SM）【NGKI1565】
E_PAR	パラメータエラー ・iflgptnがFLGPTNに格納できない（S）【NGKI3275】
E_OACV	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（SP） 【NGKI1566】
E_MACV	メモリアクセス違反 ・pk_cflg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（SP）【NGKI1567】
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（SD）【NGKI1568】
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（CRE_FLGの場合） 【NGKI1569】

【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【NGKI1570】．

静的APIにおいては，flgidはオブジェクト識別名，flgatrとiflgptnは整数定数式パラメータである【NGKI1571】．

【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【ASPS0125】．ただし，動的生成機能拡張パッケージでは，acre_flgもサポートする【ASPS0126】．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，CRE_FLGのみをサポートする【FMPS0116】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，CRE_FLGのみをサポートする【HRPS0116】．ただし，動的生成機能拡張パッケージでは，acre_flgもサポートする【HRPS0183】．

AID_FLG	割付け可能なイベントフラグIDの数の指定（SD）【NGKI1572】

【静的API】

```
AID_FLG(uint_t noflg)
```

【パラメータ】

```
uint_t      noflg       割付け可能なイベントフラグIDの数
```

【エラーコード】

```
E_RSATR     予約属性
            ・保護ドメインの囲みの中に記述されている（P）【NGKI3430】
            ・クラスの囲みの中に記述されていない（M）【NGKI1573】
E_PAR       パラメータエラー
            ・noflgが負の値【NGKI3278】
```

【機能】

noflgで指定した数のイベントフラグIDを，イベントフラグを生成するサービス コールによって割付け可能なイベントフラグIDとして確保する【NGKI1574】．

noflgは整数定数式パラメータである【NGKI1575】．

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルの動的生成機能拡張パッケージでは，AID_FLGをサポートする【ASPS0212】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルの動的生成機能拡張パッケージでは，AID_FLGをサポートする【HRPS0213】．

```
SAC_FLG     イベントフラグのアクセス許可ベクタの設定（SP）【NGKI1576】
sac_flg     イベントフラグのアクセス許可ベクタの設定（TPD）【NGKI1577】
```

【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（S）【NGKI1578】 ・CPUロック状態からの呼出し（S）【NGKI1579】
E_ID	不正ID番号 ・flgidが有効範囲外（S）【NGKI1580】
E_RSATR	予約属性 ・対象イベントフラグが属する保護ドメインの囲みの中（対象イベントフラグが無所属の場合は，保護ドメインの囲みの外）に記述されていない（S）【NGKI1581】 ・対象イベントフラグが属するクラスの囲みの中に記述されていない（SM）【NGKI1582】
E_NOEXS	オブジェクト未登録 ・対象イベントフラグが未登録【NGKI1583】
E_OACV	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（S）【NGKI1584】
E_MACV	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（S）【NGKI1585】
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（S）【NGKI1586】 ・対象イベントフラグに対してアクセス許可ベクタが設定済み（S）【NGKI1587】

【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【NGKI1588】．

静的APIにおいては，flgidはオブジェクト識別名，acptn1~acptn4は整数定数式パラメータである【NGKI1589】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，SAC_FLGのみをサポートする【HRPS0117】．ただし，動的生成機能拡張パッケージでは，sac_flgもサポートする【HRPS0184】．

del_flg	イベントフラグの削除（TD）【NGKI1590】

【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【NGKI1591】 ・CPUロック状態からの呼出し【NGKI1592】
E_ID	不正ID番号 ・flgidが有効範囲外【NGKI1593】
E_NOEXS	オブジェクト未登録 ・対象イベントフラグが未登録【NGKI1594】
E_OACV	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（P） 【NGKI1595】
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【NGKI1596】

【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【NGKI1597】．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【NGKI1598】．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【NGKI1599】．

【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_flgをサポートしない〔ASPS0128〕．ただし，動的生成 機能拡張パッケージでは，del_flgをサポートする〔ASPS0129〕．

〔TOPPERS/FMPカーネルにおける規定〕

FMPカーネルでは，del_flgをサポートしない〔FMPS0118〕．

〔TOPPERS/HRP2カーネルにおける規定〕

HRP2カーネルでは，del_flgをサポートしない〔HRPS0118〕．ただし，動的生成 機能拡張パッケージでは，del_flgをサポートする〔HRPS0185〕．

```
set_flg      イベントフラグのセット（T）〔NGKI1600〕
iset_flg     イベントフラグのセット（I）〔NGKI1601〕
```

〔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）〔NGKI1602〕
                ・タスクコンテキストからの呼出し（iset_flgの場合）〔NGKI1603〕
                ・CPUロック状態からの呼出し〔NGKI1604〕
    E_ID        不正ID番号
                ・flgidが有効範囲外〔NGKI1605〕
    E_NOEXS     オブジェクト未登録
                ・対象イベントフラグが未登録（D）〔NGKI1606〕
    E_OACV      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）（P）〔NGKI1607〕
```

【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〔NGKI1608〕．対象イベントフ

□□□

```
wai_flg     イベントフラグ待ち〔T〕〔NGKI1618〕
pol_flg     イベントフラグ待ち（ポーリング）〔T〕〔NGKI1619〕
twai_flg    イベントフラグ待ち（タイムアウト付き）〔T〕〔NGKI1620〕
```

〔C言語API〕

```
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)
```

【パラメータ】

```
ID          flgid       対象イベントフラグのID番号
FLGPTN      waiptn      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【NGKI1621】 ・CPUロック状態からの呼出し【NGKI1622】 ・ディスパッチ保留状態からの呼出し（pol_flgを除く）【NGKI1623】
E_NOSPT	未サポート機能 ・制約タスクからの呼出し（pol_flgを除く）【NGKI1624】
E_ID	不正ID番号 ・flgidが有効範囲外【NGKI1625】
E_PAR	パラメータエラー ・waiptnが0【NGKI1626】 ・wfmodeが無効（TWF_ORWまたはTWF_ANDWでない）【NGKI1627】 ・tmoutが無効（twai_flgの場合）【NGKI1628】
E_NOEXS	オブジェクト未登録 ・対象イベントフラグが未登録（D）【NGKI1629】
E_OACV	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（P） 【NGKI1630】
E_MACV	メモリアクセス違反 ・p_flgptn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（P）【NGKI1631】
E_ILUSE	サービスコール不正使用 ・TA_WMUL属性でないイベントフラグで待ちタスクあり【NGKI1632】
E_TMOUT	ポーリング失敗またはタイムアウト（wai_flgを除く）【NGKI1633】
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） 【NGKI1634】
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） 【NGKI1635】

【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p_flgptnが指すメモリ領域に返される【NGKI1636】．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【NGKI1637】．

待ち解除の条件を満たしていない場合には，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れる【NGKI1638】．

ini_flg	イベントフラグの再初期化（T）【NGKI1639】

【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【NGKI1640】 ・CPUロック状態からの呼出し【NGKI1641】
E_ID	不正ID番号 ・flgidが有効範囲外【NGKI1642】
E_NOEXS	オブジェクト未登録 ・対象イベントフラグが未登録（D）【NGKI1643】
E_OACV	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（P） 【NGKI1644】

【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【NGKI1645】．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【NGKI1646】．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【NGKI1647】．

【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．

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）【NGKI1648】

【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	flgptn	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【NGKI1649】 ・CPUロック状態からの呼出し【NGKI1650】
E_ID	不正ID番号 ・flgidが有効範囲外【NGKI1651】
E_NOEXS	オブジェクト未登録 ・対象イベントフラグが未登録（D）【NGKI1652】
E_OACV	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□（P）【NGKI1653】
E_MACV	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（P）【NGKI1654】

【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【NGKI1655】．

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【NGKI1656】．

【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の同期・通信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【NGKI1657】．

各データキューが持つ情報は次の通り【NGKI1658】．

- 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）
- アクセス許可ベクタ（保護機能対応カーネルの場合）
- 属する保護ドメイン（保護機能対応カーネルの場合）
- 属するクラス（マルチプロセッサ対応カーネルの場合）

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【NGKI1659】．

保護機能対応カーネルにおいて，データキュー管理領域は，カーネルの用いるオブジェクト管理領域として扱われる【NGKI1660】．

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【NGKI1661】．

```
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【NGKI1662】．受信待ち行列は，FIFO順に固定されている【NGKI1663】．

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）【NGKI1664】
```

【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）【NGKI1665】
acre_dtq     データキューの生成（TD）【NGKI1666】
```

【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（CRE_DTQの場合）
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（sS）【NGKI1667】 ・CPUロック状態からの呼出し（sS）【NGKI1668】
E_RSATR	予約属性 ・dtqatrが無効【NGKI1669】 ・属する保護ドメインの指定が有効範囲外（sP）【NGKI1670】 ・属するクラスの指定が有効範囲外（sM）【NGKI1671】 ・クラスの囲みの中に記述されていない（SM）【NGKI1672】
E_NOSPT	未サポート機能 ・条件については各カーネルにおける規定の項を参照
E_PAR	パラメータエラー ・dtqcntが負の値（S）【NGKI3288】 ・その他の条件については機能の項を参照
E_OACV	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（sP） 【NGKI1673】
E_MACV	メモリアクセス違反 ・pk_cdtq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（sP）【NGKI1674】
E_NOID	ID番号不足 ・割り付けられるデータキューIDがない（sD）【NGKI1675】
E_NOMEM	メモリ不足 ・データキュー管理領域が確保できない【NGKI1676】
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（CRE_DTQの場合） 【NGKI1677】 ・その他の条件については機能の項を参照

【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【NGKI1678】．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【NGKI1679】．

静的APIにおいては，dtqidはオブジェクト識別名，dtqatrと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【NGKI1680】．コンフィギュレータは，静的APIのメモリ不足（E_NOMEM）エラーを検出することができない【NGKI1681】．

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が，コンフィギュレータまたはカーネルにより確保される【NGKI1682】．

〔dtqmbにNULL以外を指定した場合〕

dtqmbにNULL以外を指定した場合，dtqmbを先頭番地とするデータキュー管理領域は，アプリケーションで確保しておく必要がある【NGKI1683】．データキュー管理領域をアプリケーションで確保するために，次のマクロを用意している【NGKI1684】．

TSZ_DTQMB(dtqcnt)	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のサイズ（バイト数）
TCNT_DTQMB(dtqcnt)	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確保するために必要なMB_T型の配列の要素数

これらを用いてデータキュー管理領域を確保する方法は次の通り【NGKI1685】.

MB_T <データキュー管理領域の変数名>[TCNT_DTQMB(dtqcnt)];

この時，dtqmbには<データキュー管理領域の変数名>を指定する【NGKI1686】.

この方法に従わず，dtqmbにターゲット定義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【NGKI1687】．また，保護機能対応カーネルにおいて，dtqmbで指定したデータキュー管理領域がカーネル専用のメモリオブジェクトに含まれない場合，E_OBJエラーとなる【NGKI1688】.

【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【ASPS0130】．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【ASPS0132】．ただし，動的生成機能拡張パッケージでは，acre_dtqもサポートする【ASPS0133】．acre_dtqに対しては，dtqmbにNULL以外を指定できないという制限はない【ASPS0134】.

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，CRE_DTQのみをサポートする【FMPS0119】．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【FMPS0121】.

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，CRE_DTQのみをサポートする【HRPS0119】．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【HRPS0121】．ただし，動的生成機能拡張パッケージでは，acre_dtqもサポートする【HRPS0186】．acre_dtqに対しては，dtqmbにNULL以外を指定できないという制限はない【HRPS0187】.

【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．また，TSZ_DTQをTSZ_DTQMBに改名した.

TCNT_DTQMBを新設し，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を規定した.

```
AID_DTQ      割付け可能なデータキューIDの数の指定（SD）【NGKI1689】
```

【静的API】

```
    AID_DTQ(uint_t nodtq)
```

【パラメータ】

```
    uint_t      nodtq        割付け可能なデータキューIDの数
```

【エラーコード】

E_RSATR		予約属性 ・保護ドメインの囲みの中に記述されている（P）【NGKI3431】 ・クラスの囲みの中に記述されていない（M）【NGKI1690】
E_PAR		パラメータエラー ・nodtqが負の値【NGKI3279】

【機能】

nodtqで指定した数のデータキューIDを，データキューを生成するサービスコールによって割付け可能なデータキューIDとして確保する【NGKI1691】．

nodtqは整数定数式パラメータである【NGKI1692】．

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルの動的生成機能拡張パッケージでは，AID_DTQをサポートする【ASPS0213】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルの動的生成機能拡張パッケージでは，AID_DTQをサポートする【HRPS0214】．

SAC_DTQ	データキューのアクセス許可ベクタの設定（SP）【NGKI1693】
sac_dtq	データキューのアクセス許可ベクタの設定（TPD）【NGKI1694】

【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（S）【NGKI1695】
            ・CPUロック状態からの呼出し（S）【NGKI1696】
E_ID        不正ID番号
            ・dtqidが有効範囲外（S）【NGKI1697】
E_RSATR     予約属性
            ・対象データキューが属する保護ドメインの囲みの中（対象
            データキューが無所属の場合は，保護ドメインの囲みの外）
            に記述されていない（S）【NGKI1698】
            ・対象データキューが属するクラスの囲みの中に記述されて
            いない（SM）【NGKI1699】
E_NOEXS     オブジェクト未登録
            ・対象データキューが未登録【NGKI1700】
E_OACV      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（S）
            【NGKI1701】
E_MACV      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（S）【NGKI1702】
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（S）【NGKI1703】
            ・対象データキューに対してアクセス許可ベクタが設定済み（S）
            【NGKI1704】
```

【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【NGKI1705】．

静的APIにおいては，dtqidはオブジェクト識別名，acptn1~acptn4は整数定数 式パラメータである【NGKI1706】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，SAC_DTQのみをサポートする【HRPS0122】．ただし，動的生 成機能拡張パッケージでは，sac_dtqもサポートする【HRPS0188】．

```
del_dtq     データキューの削除（TD）【NGKI1707】
```

【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［NGKI1708］ ・CPUロック状態からの呼出し［NGKI1709］
E_ID	不正ID番号 ・dtqidが有効範囲外［NGKI1710］
E_NOEXS	オブジェクト未登録 ・対象データキューが未登録［NGKI1711］
E_OACV	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（P ）［NGKI1712］
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［NGKI1713］

【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される［NGKI1714］．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［NGKI1715］．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［NGKI1716］．

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［NGKI1717］．

【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［ASPS0136］．ただし，動的生成機能拡張パッケージでは，del_dtqをサポートする［ASPS0137］．

［TOPPERS/FMPカーネルにおける規定］

FMPカーネルでは，del_dtqをサポートしない［FMPS0123］．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，del_dtqをサポートしない【HRPS0123】．ただし，動的生成機能拡張パッケージでは，del_dtqをサポートする【HRPS0189】．

```
snd_dtq      データキューへの送信（T）【NGKI1718】
psnd_dtq     データキューへの送信（ポーリング）（T）【NGKI1719】
ipsnd_dtq    データキューへの送信（ポーリング）（I）【NGKI1720】
tsnd_dtq     データキューへの送信（タイムアウト付き）（T）【NGKI1721】
```

【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） 【NGKI1722】 ・タスクコンテキストからの呼出し（ipsnd_dtqの場合） 【NGKI1723】 ・CPUロック状態からの呼出し【NGKI1724】 ・ディスパッチ保留状態からの呼出し（snd_dtqとtsnd_dtqの場合）【NGKI1725】
E_NOSPT	未サポート機能 ・制約タスクからの呼出し（snd_dtqとtsnd_dtqの場合）【NGKI1726】
E_ID	不正ID番号 ・dtqidが有効範囲外【NGKI1727】
E_PAR	パラメータエラー ・tmoutが無効（tsnd_dtqの場合）【NGKI1728】
E_NOEXS	オブジェクト未登録 ・対象データキューが未登録（D）【NGKI1729】
E_OACV	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）（P）【NGKI1730】
E_TMOUT	ポーリング失敗またはタイムアウト（snd_dtqを除く）【NGKI1731】
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）【NGKI1732】
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）【NGKI1733】

【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【NGKI1734】．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【NGKI1735】．

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【NGKI1736】．

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れる【NGKI1737】．

fsnd_dtq	データキューへの強制送信（T）【NGKI1738】
ifsnd_dtq	データキューへの強制送信（I）【NGKI1739】

【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）〔NGKI1740〕
            ・タスクコンテキストからの呼出し（ifsnd_dtqの場合）〔NGKI1741〕
            ・CPUロック状態からの呼出し〔NGKI1742〕
E_ID        不正ID番号
            ・dtqidが有効範囲外〔NGKI1743〕
E_NOEXS     オブジェクト未登録
            ・対象データキューが未登録（D）〔NGKI1744〕
E_OACV      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）（P）〔NGKI1745〕
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〔NGKI1746〕
```

【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〔NGKI1747〕．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〔NGKI1748〕．

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〔NGKI1749〕．

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〔NGKI1750〕．

```
rcv_dtq     データキューからの受信（T）〔NGKI1751〕
prcv_dtq    データキューからの受信（ポーリング）（T）〔NGKI1752〕
trcv_dtq    データキューからの受信（タイムアウト付き）（T）〔NGKI1753〕
```

〔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【NGKI1754】 ・CPUロック状態からの呼出し【NGKI1755】 ・ディスパッチ保留状態からの呼出し（prcv_dtqを除く） 【NGKI1756】
E_NOSPT	未サポート機能 ・制約タスクからの呼出し（prcv_dtqを除く）【NGKI1757】
E_ID	不正ID番号 ・dtqidが有効範囲外【NGKI1758】
E_PAR	パラメータエラー ・tmoutが無効（trcv_dtqの場合）【NGKI1759】
E_NOEXS	オブジェクト未登録 ・対象データキューが未登録（D）【NGKI1760】
E_OACV	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（P） 【NGKI1761】
E_MACV	メモリアクセス違反 ・p_data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（P）【NGKI1762】
E_TMOUT	ポーリング失敗またはタイムアウト（rcv_dtqを除く）【NGKI1763】
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） 【NGKI1764】
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） 【NGKI1765】

【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．データの受信に成功した場合，受信したデータはp_dataが指すメモリ領域に返される【NGKI3421】．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を受信する【NGKI1766】．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【NGKI1767】．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【NGKI1768】．

□□□

【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）〔NGKI1781〕
```

〔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〔NGKI1782〕 ・CPUロック状態からの呼出し〔NGKI1783〕
E_ID	不正ID番号 ・dtqidが有効範囲外〔NGKI1784〕
E_NOEXS	オブジェクト未登録 ・対象データキューが未登録（D）〔NGKI1785〕
E_OACV	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（P） 〔NGKI1786〕
E_MACV	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（P）〔NGKI1787〕

【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〔NGKI1788〕．

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〔NGKI1789〕．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〔NGKI1790〕．

【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の同期・通信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〔NGKI1791〕．

各優先度データキューが持つ情報は次の通り〔NGKI1792〕．

- 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
- アクセス許可ベクタ（保護機能対応カーネルの場合）
- 属する保護ドメイン（保護機能対応カーネルの場合）
- 属するクラス（マルチプロセッサ対応カーネルの場合）

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〔NGKI1793〕．

保護機能対応カーネルにおいて，優先度データキュー管理領域は，カーネルの 用いるオブジェクト管理領域として扱われる〔NGKI1794〕．

□□□

```
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（CRE_PDQの場合）
T_CPDQ *	pk_cpdq	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（sS）【NGKI1802】 ・CPUロック状態からの呼出し（sS）【NGKI1803】
E_RSATR	予約属性 ・pdqatrが無効【NGKI1804】 ・属する保護ドメインの指定が有効範囲外（sP）【NGKI1805】 ・属するクラスの指定が有効範囲外（sM）【NGKI1806】 ・クラスの囲みの中に記述されていない（SM）【NGKI1807】
E_NOSPT	未サポート機能 ・条件については各カーネルにおける規定の項を参照
E_PAR	パラメータエラー ・pdqcntが負の値（S）【NGKI3289】 ・maxdpriがTMIN_DPRIより小さい，またはTMAX_DPRIより大きい【NGKI1819】 ・その他の条件については機能の項を参照
E_OACV	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（sP）【NGKI1808】
E_MACV	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（sP）【NGKI1809】
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（sD）【NGKI1810】
E_NOMEM	メモリ不足 ・優先度データキュー管理領域が確保できない【NGKI1811】
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（CRE_PDQの場合）【NGKI1812】 ・その他の条件については機能の項を参照

【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【NGKI1813】．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【NGKI1814】．

静的APIにおいては，pdqidはオブジェクト識別名，pdqatr，pdqcnt，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【NGKI1815】．コンフィギュレータは，静的APIのメモリ不足（E_NOMEM）エラーを検出することができない【NGKI1816】．

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が，コンフィギュレータまたはカーネルにより確保される【NGKI1817】．

〔pdqmbにNULL以外を指定した場合〕

pdqmbにNULL以外を指定した場合，pdqmbを先頭番地とする優先度データキュー管理領域は，アプリケーションで確保しておく必要がある【NGKI1820】．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ために，次のマクロを用意している【NGKI1821】．

TSZ_PDQMB(pdqcnt)	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のサイズ（バイト数）
TCNT_PDQMB(pdqcnt)	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確保するために必要なMB_T型の配列の要素数

これらを用いて優先度データキュー管理領域を確保する方法は次の通り【NGKI1822】.

MB_T <優先度データキュー管理領域の変数名>[TCNT_PDQMB(pdqcnt)];

この時，pdqmbには<優先度データキュー管理領域の変数名>を指定する【NGKI1823】.

この方法に従わず，pdqmbにターゲット定義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【NGKI1824】．また，保護機能対応カーネルにいて，pdqmbで指定した優先度データキュー管理領域がカーネル専用のメモリオブジェクトに含まれない場合，E_OBJエラーとなる【NGKI1825】.

【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【ASPS0140】．また，pdqmb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【ASPS0142】．ただし，動的生成機能拡張パッケージでは，acre_pdqもサポートする【ASPS0143】．acre_pdqに対しては，pdqmbにNULL以外を指定できないという制限はない【ASPS0144】.

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，CRE_PDQのみをサポートする【FMPS0125】．また，pdqmb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【FMPS0127】.

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，CRE_PDQのみをサポートする【HRPS0125】．また，pdqmb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【HRPS0127】．ただし，動的生成機能拡張パッケージでは，acre_pdqもサポートする【HRPS0190】．acre_pdqに対しては，pdqmbにNULL以外を指定できないという制限はない【HRPS0191】.

AID_PDQ	割付け可能な優先度データキューIDの数の指定〔SD〕【NGKI1826】	

【静的API】

```
AID_PDQ(uint_t nopdq)
```

【パラメータ】

uint_t	nopdq	割付け可能な優先度データキューIDの数

【エラーコード】

E_RSATR	予約属性 ・保護ドメインの囲みの中に記述されている（P）【NGKI3432】 ・クラスの囲みの中に記述されていない（M）【NGKI1827】	
E_PAR	パラメータエラー ・nopdqが負の値【NGKI3280】	

【機能】

nopdqで指定した数の優先度データキューIDを，優先度データキューを生成するサービスコールによって割付け可能な優先度データキューIDとして確保する【NGKI1828】．

nopdqは整数定数式パラメータである【NGKI1829】．

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルの動的生成機能拡張パッケージでは，AID_PDQをサポートする【ASPS0214】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルの動的生成機能拡張パッケージでは，AID_PDQをサポートする【HRPS0215】．

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）【NGKI1830】
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）【NGKI1831】

【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（S）【NGKI1832】
            ・CPUロック状態からの呼出し（S）【NGKI1833】
E_ID        不正ID番号
            ・pdqidが有効範囲外（S）【NGKI1834】
E_RSATR     予約属性
            ・対象優先度データキューが属する保護ドメインの囲みの中
            （対象優先度データキューが無所属の場合は，保護ドメイ
            ンの囲みの外）に記述されていない（S）【NGKI1835】
            ・対象優先度データキューが属するクラスの囲みの中に記述
            されていない（SM）【NGKI1836】
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【NGKI1837】
E_OACV      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（S）【NGKI1838】
E_MACV      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（S）【NGKI1839】
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（S）【NGKI1840】
            ・対象優先度データキューに対してアクセス許可ベクタが設
            定済み（S）【NGKI1841】
```

【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【NGKI1842】．

静的APIにおいては，pdqidはオブジェクト識別名，acptn1~acptn4は整数定数 式パラメータである【NGKI1843】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，SAC_PDQのみをサポートする【HRPS0128】．ただし，動的生 成機能拡張パッケージでは，sac_pdqもサポートする【HRPS0192】．

```
del_pdq     優先度データキューの削除（TD）【NGKI1844】
```

【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【NGKI1845】 ・CPUロック状態からの呼出し【NGKI1846】
E_ID	不正ID番号 ・pdqidが有効範囲外【NGKI1847】
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【NGKI1848】
E_OACV	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（P）【NGKI1849】
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【NGKI1850】

【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【NGKI1851】．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【NGKI1852】．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【NGKI1853】．

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【NGKI1854】．

【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_pdqをサポートしない【ASPS0146】．ただし，動的生成機能拡張パッケージでは，del_pdqをサポートする【ASPS0147】．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，del_pdqをサポートしない【FMPS0129】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，del_pdqをサポートしない【HRPS0129】．ただし，動的生成機能拡張パッケージでは，del_pdqをサポートする【HRPS0193】．

```
snd_pdq      優先度データキューへの送信（T）【NGKI1855】
psnd_pdq     優先度データキューへの送信（ポーリング）（T）【NGKI1856】
ipsnd_pdq    優先度データキューへの送信（ポーリング）（I）【NGKI1857】
tsnd_pdq     優先度データキューへの送信（タイムアウト付き）（T）【NGKI1858】
```

【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） 【NGKI1859】 ・タスクコンテキストからの呼出し（ipsnd_pdqの場合）【NGKI1860】 ・CPUロック状態からの呼出し【NGKI1861】 ・ディスパッチ保留状態からの呼出し（snd_pdqとtsnd_pdqの 場合）【NGKI1862】
E_NOSPT	未サポート機能 ・制約タスクからの呼出し（snd_pdqとtsnd_pdqの場合）【NGKI1863】
E_ID	不正ID番号 ・pdqidが有効範囲外【NGKI1864】
E_PAR	パラメータエラー ・tmoutが無効（tsnd_pdqの場合）【NGKI1865】 ・その他の条件については機能の項を参照
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（D）【NGKI1866】
E_OACV	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）（P）【NGKI1867】
E_TMOUT	ポーリング失敗またはタイムアウト（snd_pdqを除く）【NGKI1868】
E_RLWAI	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）【NGKI1869】
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）【NGKI1870】

【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【NGKI1871】．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【NGKI1872】．

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【NGKI1873】．

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れる【NGKI1874】．

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．そうでない場合には，E_PARエラーとなる【NGKI1876】．

rcv_pdq	優先度データキューからの受信（T）【NGKI1877】
prcv_pdq	優先度データキューからの受信（ポーリング）（T）【NGKI1878】
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）【NGKI1879】

【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	ercd	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〔NGKI1880〕 ・CPUロック状態からの呼出し〔NGKI1881〕 ・ディスパッチ保留状態からの呼出し（prcv_pdqを除く）〔NGKI1882〕
E_NOSPT	未サポート機能 ・制約タスクからの呼出し（prcv_pdqを除く）〔NGKI1883〕
E_ID	不正ID番号 ・pdqidが有効範囲外〔NGKI1884〕
E_PAR	パラメータエラー ・tmoutが無効（trcv_pdqの場合）〔NGKI1885〕
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（D）〔NGKI1886〕
E_OACV	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（P）〔NGKI1887〕
E_MACV	メモリアクセス違反 ・p_data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（P）〔NGKI1888〕 ・p_datapri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（P）〔NGKI1889〕
E_TMOUT	ポーリング失敗またはタイムアウト（rcv_pdqを除く）〔NGKI1890〕
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）〔NGKI1891〕
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）〔NGKI1892〕

□□□□

pdqid□□□□

pdqid□□□

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〔NGKI1912〕 ・CPUロック状態からの呼出し〔NGKI1913〕
E_ID	不正ID番号 ・pdqidが有効範囲外〔NGKI1914〕
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（D）〔NGKI1915〕
E_OACV	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（P）〔NGKI1916〕
E_MACV	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（P）〔NGKI1917〕

【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で指定したパケットに返される〔NGKI1918〕．

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〔NGKI1919〕．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〔NGKI1920〕．

【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の同期・通信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〔NGKI1921〕．

各メールボックスが持つ情報は次の通り〔NGKI1922〕．

- 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
- 属するクラス（マルチプロセッサ対応カーネルの場合）

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〔NGKI1923〕．

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【NGKI1924】．TA_MPRIを 指定しない場合，メッセージキューはFIFO順になる【NGKI1925】．

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 なぐために，メッセージの先頭のメモリ領域を使用する【NGKI1926】．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【NGKI1927】．
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【NGKI1928】．

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

メッセージヘッダの領域は，メッセージがメッセージキューにつながれている 間（すなわち，メールボックスに送信してから受信するまでの間），カーネル によって使用される【NGKI1929】．そのため，メッセージキューにつながれて いるメッセージのメッセージヘッダの領域をアプリケーションが書き換えた場 合や，メッセージキューにつながれているメッセージを再度メールボックスに 送信した場合の動作は保証されない【NGKI1930】．

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【NGKI1931】．

保護機能対応カーネルでは，メールボックス機能はサポートしない【NGKI1932】．

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）	【NGKI1933】
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）【NGKI1934】

【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，μITRON4.0仕様との互換性のために残した機能であり， 保護機能対応カーネルではサポートしないため，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【ASPS0147】．メッセー ジ優先度の最大値（TMAX_MPRI）は16に固定されている【ASPS0148】．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【ASPS0149】．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，メールボックス機能をサポートする【FMPS0130】．メッセージ優先度の最大値（TMAX_MPRI）は16に固定されている【FMPS0131】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，メールボックス機能をサポートしない【HRPS0130】．

【μITRON4.0仕様との関係】

TNUM_MBXID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_MBX	メールボックスの生成（Sp）【NGKI1935】	
acre_mbx	メールボックスの生成（TpD）【NGKI1936】	

【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（CRE_MBXの場合）
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（sS）【NGKI1937】 ・CPUロック状態からの呼出し（sS）【NGKI1938】
E_RSATR	予約属性 ・mbxatrが無効【NGKI1939】 ・属するクラスの指定が有効範囲外（sM）【NGKI1940】 ・クラスの囲みの中に記述されていない（SM）【NGKI1941】
E_NOSPT	未サポート機能 ・条件については各カーネルにおける規定の項を参照
E_PAR	パラメータエラー ・maxmpriがTMIN_MPRIより小さい，またはTMAX_MPRIより大きい【NGKI1951】
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（sD）【NGKI1942】
E_NOMEM	メモリ不足 ・優先度別のメッセージキューヘッダ領域が確保できない【NGKI1943】
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（CRE_MBXの場合）【NGKI1944】

【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【NGKI1945】．また，待ち行列は空の状態に初期化される【NGKI1946】．

静的APIにおいては，mbxidはオブジェクト識別名，mbxatrと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【NGKI1947】．コンフィギュレータは，静的APIのメモリ不足（E_NOMEM）エラーを検出することができない【NGKI1948】．

mprihdをNULLとした場合，maxmpri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が，コンフィギュレータまたはカーネルにより確保される【NGKI1949】．

【未決定事項】

mprihdにNULL以外を指定した場合の扱いについては，この仕様では規定していない．

【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【ASPS0150】．また，優先度別のメッセージキューヘッダ領域は使用しておらず，mprihd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【ASPS0152】．ただし，動的生成機能拡張パッケージでは，acre_mbxもサポートする【ASPS0153】．acre_mbxに対しても，mprihdにはNULLのみを指定することができる【ASPS0154】．優先度別のメッセージキューヘッダ領域を使用しないため，E_NOMEMが返ることはない【ASPS0155】．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，CRE_MBXのみをサポートする【FMPS0132】．また，優先度別のメッセージキューヘッダ領域は使用しておらず，mprihd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【FMPS0134】．優先度別のメッセージキューヘッダ領域を使用しないため，E_NOMEMが返ることはない【FMPS0135】．

AID_MBX	割付け可能なメールボックスIDの数の指定（SpD）〔NGKI1952〕

〔静的API〕

```
AID_MBX(uint_t nombx)
```

〔パラメータ〕

uint_t	nombx	割付け可能なメールボックスIDの数

〔エラーコード〕

E_RSATR	予約属性 ・クラスの囲みの中に記述されていない（M）〔NGKI1953〕
E_PAR	パラメータエラー ・nombxが負の値〔NGKI3281〕

〔機能〕

nombxで指定した数のメールボックスIDを，メールボックスを生成するサービス コールによって割付け可能なメールボックスIDとして確保する〔NGKI1954〕．

nombxは整数定数式パラメータである〔NGKI1955〕．

〔TOPPERS/ASPカーネルにおける規定〕

ASPカーネルの動的生成機能拡張パッケージでは，AID_MBXをサポートする〔ASPS0215〕．

del_mbx	メールボックスの削除（TpD）〔NGKI1956〕

〔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〔NGKI1957〕 ・CPUロック状態からの呼出し〔NGKI1958〕
E_ID	不正ID番号 ・mbxidが有効範囲外〔NGKI1959〕
E_NOEXS	オブジェクト未登録 ・対象メールボックスが未登録〔NGKI1960〕
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〔NGKI1961〕

【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〔NGKI1962〕．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〔NGKI1963〕．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〔NGKI1964〕．

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〔NGKI1965〕．

【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〔ASPS0156〕．ただし，動的生成 機能拡張パッケージでは，del_mbxをサポートする〔ASPS0157〕．

〔TOPPERS/FMPカーネルにおける規定〕

FMPカーネルでは，del_mbxをサポートしない〔FMPS0136〕．

snd_mbx	メールボックスへの送信〔Tp〕〔NGKI1966〕	

〔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〔NGKI1967〕 ・CPUロック状態からの呼出し〔NGKI1968〕
E_ID	不正ID番号 ・mbxidが有効範囲外〔NGKI1969〕
E_PAR	パラメータエラー ・条件については機能の項を参照
E_NOEXS	オブジェクト未登録 ・対象メールボックスが未登録（D）〔NGKI1970〕

【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〔NGKI1971〕．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〔NGKI1972〕．

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がれる〔NGKI1973〕．

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．そうでない場合には，E_PARエラーとなる〔NGKI1975〕．

rcv_mbx	メールボックスからの受信（Tp）〔NGKI1976〕
prcv_mbx	メールボックスからの受信（ポーリング）（Tp）〔NGKI1977〕
trcv_mbx	メールボックスからの受信（タイムアウト付き）（Tp）〔NGKI1978〕

〔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〔NGKI1979〕 ・CPUロック状態からの呼出し〔NGKI1980〕 ・ディスパッチ保留状態からの呼出し（prcv_mbxを除く）〔NGKI1981〕
E_NOSPT	未サポート機能 ・制約タスクからの呼出し（prcv_mbxを除く）〔NGKI1982〕
E_ID	不正ID番号 ・mbxidが有効範囲外〔NGKI1983〕
E_PAR	パラメータエラー ・tmoutが無効（trcv_mbxの場合）〔NGKI1984〕
E_NOEXS	オブジェクト未登録 ・対象メールボックスが未登録〔D〕〔NGKI1985〕
E_TMOUT	ポーリング失敗またはタイムアウト（rcv_mbxを除く）〔NGKI1986〕
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） 〔NGKI1987〕
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） 〔NGKI1988〕

【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〔NGKI1989〕．

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〔NGKI1990〕．

ini_mbx	メールボックスの再初期化〔Tp〕〔NGKI1991〕	

〔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〔NGKI1992〕 ・CPUロック状態からの呼出し〔NGKI1993〕
E_ID	不正ID番号 ・mbxidが有効範囲外〔NGKI1994〕
E_NOEXS	オブジェクト未登録 ・対象メールボックスが未登録（D）〔NGKI1995〕

【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〔NGKI1996〕．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〔NGKI1997〕．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〔NGKI1998〕．

【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（Tp）〔NGKI1999〕

〔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〔NGKI2000〕 ・CPUロック状態からの呼出し〔NGKI2001〕
E_ID	不正ID番号 ・mbxidが有効範囲外〔NGKI2002〕
E_NOEXS	オブジェクト未登録 ・対象メールボックスが未登録（D）〔NGKI2003〕

【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〔NGKI2004〕．

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〔NGKI2005〕．また，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，pk_msgにはNULLが返る〔NGKI2006〕．

【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

ミューテックスは，タスク間の排他制御を行うための同期・通信オブジェクトである．タスクは，排他制御区間に入る時にミューテックスをロックし，排他制御区間を出る時にロック解除する．ミューテックスは，ミューテックスIDと呼ぶID番号によって識別する〔NGKI2007〕．

ミューテックスは，排他制御に伴う優先度逆転の時間を最小限に抑えるための優先度上限プロトコル（priority ceiling protocol）をサポートする．ミューテックス属性により優先度上限ミューテックスであると指定することで，そのミューテックスの操作時に，優先度上限プロトコルに従った現在優先度の制御が行われる．

各ミューテックスが持つ情報は次の通り〔NGKI2008〕．

- ミューテックス属性
- ロック状態（ロックされている状態とロック解除されている状態）
- ミューテックスをロックしているタスク
- 待ち行列（ミューテックスのロック待ち状態のタスクのキュー）
- 上限優先度（優先度上限ミューテックスの場合）
- アクセス許可ベクタ（保護機能対応カーネルの場合）

- 属する保護ドメイン（保護機能対応カーネルの場合）
- 属するクラス（マルチプロセッサ対応カーネルの場合）

待ち行列は，ミューテックスをロックできるまで待っている状態（ミューテッ クスのロック待ち状態）のタスクが，ミューテックスをロックできる順序でつ ながれているキューである．

上限優先度は，優先度上限ミューテックスに対してのみ有効で，ミューテック スの生成時に，そのミューテックスをロックする可能性のあるタスクのベース優先度の中で最も高い優先度（または，それより高い優先度）に設定する【NGKI2009】．

ミューテックス属性には，次の属性を指定することができる【NGKI2010】．

TA_TPRI	0x01U	待ち行列をタスクの優先度順にする
TA_CEILING	0x03U	優先度上限ミューテックスとする．待ち行列をタスクの優先度順にする

TA_TPRI，TA_CEILINGのいずれも指定しない場合，待ち行列はFIFO順になる【NGKI2011】．

ミューテックス機能に関連して，各タスクが持つ情報は次の通り【NGKI2012】．

- ロックしているミューテックスのリスト

ロックしているミューテックスのリストは，タスクの起動時に空に初期化される【NGKI2013】．

タスクの現在優先度は，そのタスクのベース優先度と，そのタスクがロックし ている優先度上限ミューテックスの優先度上限の中で，最も高い優先度に設定 される【NGKI2014】．

ミューテックス機能によりタスクの現在優先度が変化する場合には，次の処理が行われる．現在優先度を変化させるサービスコールの前後とも，当該タスクが実行できる状態である場合には，同じ優先度のタスクの中で最高優先順位となる【NGKI2015】．そのサービスコールにより，当該タスクが実行できる状態に遷移する場合には，同じ優先度のタスクの中で最低優先順位となる【NGKI2016】．そのサービスコールの後で，当該タスクが待ち状態で，タスクの優先度順の待ち行列につながれている場合には，当該タスクの変更後の現在優先度に従って，その待ち行列中での順序が変更される【NGKI2017】．待ち行列中に同じ現在優先度のタスクがある場合には，当該タスクの順序はそれらの 中で最後になる【NGKI2018】．

ミューテックス機能に関連して，タスクの終了時に行うべき処理として，タスクがロックしているミューテックスのロック解除がある．タスクの終了時にロックしているミューテックスが残っている場合，それらのミューテックスは，ロッ クしたのと逆の順序でロック解除される【NGKI2019】．

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）【NGKI2020】

【使用上の注意】

優先度上限プロトコルには，(a) 優先度の低いタスクの排他制御区間に最大1回 しかブロックされない，(b) タスクの実行が開始された以降は優先度の低いタスクにブロックされないという利点があるが，これは，タスク間の同期に優先 度上限ミューテックスのみを用い，他の方法でタスクのスケジューリングに関 与しない場合に得られる利点である．

これらの利点を得るためには，タスクの優先順位の回転やディスパッチの禁止を行ってはならないことに加えて，優先度上限ミューテックスをロックしたタスクを待ち状態にしてはならない．特に，優先度上限ミューテックスに対して，タスクがロック待ち状態になる状況に注意が必要である（優先度上限プロトコルでは，タスクがミューテックスのロック待ち状態になることはない）．

例えば，着目するタスクAと，タスクAよりベース優先度の低いタスクBとタスクC，タスクAよりも高い上限優先度を持った優先度上限ミューテックスがある場

合を考える．タスクAがミューテックスをロックし，タスクBとタスクCがミューテックスを待っている状況で，タスクAがミューテックスをロック解除すると，タスクBがミューテックスをロックして優先度が上がり，タスクBに切り換わる．さらにタスクBがミューテックスをロック解除すると，タスクCがミューテックスをロックして優先度が上がり，タスクCに切り換わる．タスクAが実行されるのは，タスクCがミューテックスをロック解除した後である．この例では，タスクAが実行開始後に，タスクBとタスクCの排他制御区間にブロックされることにな る．

優先度上限ミューテックスに対してタスクがロック待ち状態になる状況を回避 するためには，優先度上限ミューテックスをロックする場合に，待ち状態にな らないploc_mtxを用いるのが安全である．

【補足説明】

この仕様で優先度上限プロトコルと呼んでいる方式は，オリジナルのpriority ceiling protocolとは異なるものである．この仕様の方式は，OSEK/VDX OS仕様でもpriority ceiling protocolと呼ばれているが，学術論文や他のOSでは， immediate ceiling priority protocol，priority protection protocol， priority ceiling emulation，highest locker protocolなどと呼ばれている．

【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をサポートしない【ASPS0158】．ただし，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【ASPS0159】．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，ミューテックス機能をサポートしない【FMPS0137】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，ミューテックス機能をサポートする【HRPS0131】．

【未決定事項】

マルチプロセッサにおいては，タスク間の同期に優先度上限ミューテックスのみを用い，他の方法でタスクのスケジューリングに関与しない場合でも，優先度上限ミューテックスに対してタスクがロック待ち状態になる．マルチプロセッ サ対応カーネルにおける優先度上限ミューテックスの扱いについては，今後の 課題である．

【μITRON4.0仕様との関係】

μITRON4.0仕様の厳密な優先度制御規則を採用し，簡略化した優先度制御規則 はサポートしていない．また，μITRON4.0仕様でサポートしている優先度継承 プロトコル（priority inheritance protocol）は，現時点ではサポートしてい ない．

ミューテックス機能によりタスクの現在優先度が変化する場合の振舞いは， μITRON4.0仕様では実装依存となっているが，この仕様では規定している．

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）【NGKI2021】
acre_mtx     ミューテックスの生成（TD）【NGKI2022】
```

【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)
```

【パラメータ】

ID	mtxid	生成するミューテックスのID番号（CRE_MTXの場合）
T_CMTX *	pk_cmtx	ミューテ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）

□*ミューテックスの生成情報（パケットの内容）

ATR	mtxatr	ミューテックス属性
PRI	ceilpri	ミューテックスの上限優先度

【リターンパラメータ】

ER_ID	mtxid	生成されたミューテックスの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〔S〕【NGKI2023】 ・CPUロック状態からの呼出し〔S〕【NGKI2024】
E_RSATR	予約属性 ・mtxatrが無効【NGKI2025】 ・属する保護ドメインの指定が有効範囲外〔SP〕【NGKI2026】 ・属するクラスの指定が有効範囲外〔SM〕【NGKI2027】 ・クラスの囲みの中に記述されていない〔SM〕【NGKI2028】
E_PAR	パラメータエラー ・条件については機能の項を参照
E_OACV	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〔SP〕□【NGKI2029】
E_MACV	メモリアクセス違反 ・pk_cmtx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〔SP〕【NGKI2030】
E_NOID	ID番号不足 ・割り付けられるミューテックスIDがない〔SD〕【NGKI2031】
E_OBJ	オブジェクト状態エラー ・mtxidで指定したセマフォが登録済み（CRE_MTXの場合）【NGKI2032】

【機能】

各パラメータで指定したミューテックス生成情報に従って，ミューテックスを生成する．生成されたミューテックスのロック状態はロックされていない状態に，待ち行列は空の状態に初期化される【NGKI2033】．

静的APIにおいては，mtxidはオブジェクト識別名，mtxatrとceilpriは整数定数式パラメータである〔NGKI2034〕．優先度上限ミューテックス以外の場合には，ceilpriの指定を省略することができる〔NGKI2035〕．

優先度上限ミューテックスを生成する場合，ceilpriは，TMIN_TPRI以上，TMAX_TPRI以下でなければならない．そうでない場合には，E_PARエラーとなる〔NGKI2037〕．

〔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〔ASPS0160〕．

〔TOPPERS/HRP2カーネルにおける規定〕

HRP2カーネルでは，CRE_MTXのみをサポートする〔HRPS0132〕．ただし，動的生成機能拡張パッケージでは，acre_mtxもサポートする〔HRPS0194〕．

AID_MTX	割付け可能なミューテックスIDの数の指定〔SD〕〔NGKI2038〕

【静的API】

```
AID_MTX(uint_t nomtx)
```

【パラメータ】

uint_t	nomtx	割付け可能なミューテックスIDの数

【エラーコード】

E_RSATR	予約属性 ・保護ドメインの囲みの中に記述されている〔P〕〔NGKI3433〕 ・クラスの囲みの中に記述されていない〔M〕〔NGKI2039〕
E_PAR	パラメータエラー ・nomtxが負の値〔NGKI3282〕

【機能】

nomtxで指定した数のミューテックスIDを，ミューテックスを生成するサービスコールによって割付け可能なミューテックスIDとして確保する〔NGKI2040〕．

nomtxは整数定数式パラメータである〔NGKI2041〕．

〔TOPPERS/HRP2カーネルにおける規定〕

HRP2カーネルの動的生成機能拡張パッケージでは，AID_MTXをサポートする〔HRPS0216〕．

SAC_MTX	ミューテックスのアクセス許可ベクタの設定〔SP〕〔NGKI2042〕
sac_mtx	ミューテックスのアクセス許可ベクタの設定〔TPD〕〔NGKI2043〕

【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)
```

【パラメータ】

ID	mtxid	対象ミューテックスの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〔S〕【NGKI2044】 ・CPUロック状態からの呼出し〔S〕【NGKI2045】
E_ID	不正ID番号 ・mtxidが有効範囲外〔S〕【NGKI2046】
E_RSATR	予約属性 ・対象ミューテックスが属する保護ドメインの囲みの中（対象ミューテックスが無所属の場合は，保護ドメインの囲みの外）に記述されていない〔S〕【NGKI2047】 ・対象ミューテックスが属するクラスの囲みの中に記述されていない〔SM〕【NGKI2048】
E_NOEXS	オブジェクト未登録 ・対象ミューテックスが未登録【NGKI2049】
E_OACV	オブジェクトアクセス違反 ・対象ミューテックスに対する管理操作が許可されていない〔S〕【NGKI2050】
E_MACV	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〔S〕【NGKI2051】
E_OBJ	オブジェクト状態エラー ・対象ミューテックスは静的APIで生成された〔S〕【NGKI2052】 ・対象ミューテックスに対してアクセス許可ベクタが設定済み〔S〕【NGKI2053】

【機能】

mtxidで指定したミューテックス（対象ミューテックス）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【NGKI2054】．

静的APIにおいては，mtxidはオブジェクト識別名，acptn1~acptn4は整数定数 式パラメータである【NGKI2055】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，SAC_MTXのみをサポートする【HRPS0133】．ただし，動的生 成機能拡張パッケージでは，sac_mtxもサポートする【HRPS0195】．

del_mtx	ミューテックスの削除〔TD〕【NGKI2056】

【C言語API】

```
ER ercd = del_mtx(ID mtxid)
```

【パラメータ】

ID	mtxid	対象ミューテックスの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【NGKI2057】 ・CPUロック状態からの呼出し【NGKI2058】
E_ID	不正ID番号 ・mtxidが有効範囲外【NGKI2059】
E_NOEXS	オブジェクト未登録 ・対象ミューテックスが未登録【NGKI2060】
E_OACV	オブジェクトアクセス違反 ・対象ミューテックスに対する管理操作が許可されていない（P） 【NGKI2061】
E_OBJ	オブジェクト状態エラー ・対象ミューテックスは静的APIで生成された【NGKI2062】

【機能】

mtxidで指定したミューテックス（対象ミューテックス）を削除する．具体的な 振舞いは以下の通り．

対象ミューテックスの登録が解除され，そのミューテックスIDが未使用の状態に戻される【NGKI2063】．対象ミューテックスをロックしているタスクがある場合には，そのタスクがロックしているミューテックスのリストから対象ミューテックスが削除され，必要な場合にはそのタスクの現在優先度が変更される【NGKI2064】．また，対象ミューテ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【NGKI2065】．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【NGKI2066】．

【使用上の注意】

対象ミューテックスをロックしているタスクには，ミューテックスが削除されたことが通知されず，そのミューテックスをロック解除する時点でエラーとなる．これが不都合な場合には，ミューテックスを削除しようとするタスクが ミューテックスをロックした状態で，ミューテックスを削除すればよい．

del_mtx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【ASPS0162】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，del_mtxをサポートしない【HRPS0134】．ただし，動的生成 機能拡張パッケージでは，del_mtxをサポートする【HRPS0196】．

loc_mtx	ミューテックスのロック（T）【NGKI2067】
ploc_mtx	ミューテックスのロック（ポーリング）（T）【NGKI2068】
tloc_mtx	ミューテックスのロック（タイムアウト付き）（T）【NGKI2069】

【C言語API】

```
ER ercd = loc_mtx(ID mtxid)
ER ercd = ploc_mtx(ID mtxid)
ER ercd = tloc_mtx(ID mtxid, TMO tmout)
```

【パラメータ】

ID	mtxid	対象ミューテックスのID番号
TMO	tmout	タイムアウト時間（tloc_mtx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【NGKI2070】 ・CPUロック状態からの呼出し【NGKI2071】 ・ディスパッチ保留状態からの呼出し（ploc_mtxを除く）【NGKI2072】
E_NOSPT	未サポート機能 ・制約タスクからの呼出し（ploc_mtxを除く）【NGKI2073】
E_ID	不正ID番号 ・mtxidが有効範囲外【NGKI2074】
E_PAR	パラメータエラー ・tmoutが無効（tloc_mtxの場合）【NGKI2075】
E_NOEXS	オブジェクト未登録 ・対象ミューテックスが未登録（D）【NGKI2076】
E_OACV	オブジェクトアクセス違反 ・対象ミューテックスに対する通常操作1が許可されていない（P） 【NGKI2077】
E_ILUSE	サービスコール不正使用 ・条件については機能の項を参照
E_OBJ	オブジェクト状態エラー ・対象ミューテックスが自タスクによってロックされている 【NGKI3609】
E_TMOUT	ポーリング失敗またはタイムアウト（loc_mtxを除く）【NGKI2078】
E_RLWAI	待ち禁止状態または待ち状態の強制解除（ploc_mtxを除く） 【NGKI2079】
E_DLT	待ちオブジェクトの削除または再初期化（ploc_mtxを除く） 【NGKI2080】

【機能】

mtxidで指定したミューテックス（対象ミューテックス）をロックする．具体的 な振舞いは以下の通り．

対象ミューテックスがロックされていない場合には，自タスクによってロック されている状態になる【NGKI2081】．自タスクがロックしているミューテック

□□□

□□□

□□□

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〔NGKI2109〕 ・CPUロック状態からの呼出し〔NGKI2110〕
E_ID	不正ID番号 ・mtxidが有効範囲外〔NGKI2111〕
E_NOEXS	オブジェクト未登録 ・対象ミューテックスが未登録（D）〔NGKI2112〕
E_OACV	オブジェクトアクセス違反 ・対象ミューテックスに対する参照操作が許可されていない（P） 〔NGKI2113〕
E_MACV	メモリアクセス違反 ・pk_rmtx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（P）〔NGKI2114〕

【機能】

mtxidで指定したミューテックス（対象ミューテックス）の現在状態を参照する．参照した現在状態は，pk_rmtxで指定したパケットに返される．

対象ミューテックスがロックされていない場合，htskidにはTSK_NONE（=0）が返る〔NGKI2115〕．

対象ミューテ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〔NGKI2116〕．

【使用上の注意】

ref_mtx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_mtxを呼び出し，対象ミューテ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_mtxから戻ってきた時には対象ミューテックスの状態が変化している可能性があるためである．

4.4.7. メッセージバッファ

メッセージバッファは，指定した長さのバイト列をメッセージとして，FIFO順で送受信するための同期・通信オブジェクトである．メッセージバッファは，メッセージバッファIDと呼ぶID番号によって識別する〔NGKI3291〕．

各メッセージバッファが持つ情報は次の通り〔NGKI3292〕．

- メッセージバッファ属性
- 最大メッセージサイズ
- メッセージバッファ管理領域
- 送信待ち行列（メッセージバッファへの送信待ち状態のタスクのキュー）
- 受信待ち行列（メッセージバッファからの受信待ち状態のタスクのキュー）
- アクセス許可ベクタ（保護機能対応カーネルの場合）
- 属する保護ドメイン（保護機能対応カーネルの場合）
- 属するクラス（マルチプロセッサ対応カーネルの場合）

メッセージバッファ管理領域は，メッセージバッファに送信されたメッセージを，送信された順に格納しておくためのメモリ領域である．メッセージバッファ生成時の指定により，メッセージバッファ管理領域のサイズを0とすることができる〔NGKI3293〕．

保護機能対応カーネルにおいて，メッセージバッファ管理領域は，カーネルの 用いるオブジェクト管理領域として扱われる〔NGKI3294〕．

□□□

ID	mbfid	生成するメッセージバッファのID番号（CRE_MBFの場合）
T_CMBF *	pk_cmbf	メッセージバッファ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）

□*メッセージバッファの生成情報（パケットの内容）

ATR	mbfatr	メッセージバッファ属性
uint_t	maxmsz	メッセージバッファの最大メッセージサイズ（バイト数）
SIZE	mbfsz	メッセージバッファ管理領域のサイズ（バイト数）
void *	mbfmb	メッセージバッファ管理領域の先頭番地

【リターンパラメータ】

ER_ID	mbfid	生成されたメッセージバッファのID番号（正の値）またはエラーコード

【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〔S〕【NGKI3301】 ・CPUロック状態からの呼出し〔S〕【NGKI3302】
E_RSATR	予約属性 ・mbfatrが無効【NGKI3303】 ・属するクラスの指定が有効範囲外〔SP〕【NGKI3304】 ・属するクラスの指定が有効範囲外〔SM〕【NGKI3305】 ・クラスの囲みの中に記述されていない〔SM〕【NGKI3306】
E_NOSPT	未サポート機能 ・条件については各カーネルにおける規定の項を参照
E_PAR	パラメータエラー ・maxmszが0以下【NGKI3307】 ・mbfszが負の値〔S〕【NGKI3308】 ・その他の条件については機能の項を参照
E_OACV	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〔SP□】【NGKI3309】
E_MACV	メモリアクセス違反 ・pk_cmbf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〔SP〕【NGKI3310】
E_NOID	ID番号不足 ・割り付けられるメッセージバッファIDがない〔SD〕【NGKI3311】
E_NOMEM	メモリ不足 ・メッセージバッファ管理領域が確保できない【NGKI3312】
E_OBJ	オブジェクト状態エラー ・mbfidで指定したメッセージバッファが登録済み（CRE_MBF□の場合）【NGKI3313】 ・その他の条件については機能の項を参照

□□□□

□□□□

ファ管理領域の先頭番地）に改名した．また，TSZ_MBFをTSZ_MBFMBに改名した．

TCNT_MBFMBを新設し，メッセージバッファ管理領域をアプリケーションで確保する方法を規定した．

```
AID_MBF      割付け可能なメッセージバッファIDの数の指定〔SD〕【NGKI3326】
```

【静的API】

```
    AID_MBF(uint_t nombf)
```

【パラメータ】

```
    uint_t      nombf          割付け可能なメッセージバッファIDの数
```

【エラーコード】

```
    E_RSATR     予約属性
                ・保護ドメインの囲みの中に記述されている〔P〕【NGKI3434】
                ・クラスの囲みの中に記述されていない〔M〕【NGKI3327】
    E_PAR       パラメータエラー
                ・nombfが負の値【NGKI3328】
```

【機能】

nombfで指定した数のメッセージバッファIDを，メッセージバッファを生成するサービスコールによって割付け可能なメッセージバッファIDとして確保する【NGKI3329】．

nombfは整数定数式パラメータである【NGKI3330】．

```
SAC_MBF      メッセージバッファのアクセス許可ベクタの設定〔SP〕【NGKI3331】
sac_mbf      メッセージバッファのアクセス許可ベクタの設定〔TPD〕【NGKI3332】
```

【静的API】

```
    SAC_MBF(ID mbfid, { ACPTN acptn1, ACPTN acptn2,
                                      ACPTN acptn3, ACPTN acptn4 })
```

【C言語API】

```
    ER ercd = sac_mbf(ID mbfid, const ACVCT *p_acvct)
```

【パラメータ】

ID	mbfid	対象メッセージバッファの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（S）【NGKI3333】 ・CPUロック状態からの呼出し（S）【NGKI3334】
E_ID	不正ID番号 ・mbfidが有効範囲外（S）【NGKI3335】
E_RSATR	予約属性 ・対象メッセージバッファが属する保護ドメインの囲みの中（対象メッセージバッファが無所属の場合は，保護ドメインの囲みの外）に記述されていない（S）【NGKI3336】 ・対象メッセージバッファが属するクラスの囲みの中に記述されていない（SM）【NGKI3337】
E_NOEXS	オブジェクト未登録 ・対象メッセージバッファが未登録【NGKI3338】
E_OACV	オブジェクトアクセス違反 ・対象メッセージバッファに対する管理操作が許可されていない（S）【NGKI3339】
E_MACV	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（S）【NGKI3340】
E_OBJ	オブジェクト状態エラー ・対象メッセージバッファは静的APIで生成された（S）【NGKI3341】 ・対象メッセージバッファに対してアクセス許可ベクタが設定済み（S）【NGKI3342】

【機能】

mbfidで指定したメッセージバッファ（対象メッセージバッファ）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【NGKI3343】．

静的APIにおいては，mbfidはオブジェクト識別名，acptn1~acptn4は整数定数 式パラメータである【NGKI3344】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルのメッセージバッファ機能拡張パッケージでは，SAC_MBFのみをサポートする〔HRPS0172〕．

del_mbf	メッセージバッファの削除〔TD〕〔NGKI3345〕

〔C言語API〕

```
ER ercd = del_mbf(ID mbfid)
```

【パラメータ】

ID	mbfid	対象メッセージバッファの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〔NGKI3346〕 ・CPUロック状態からの呼出し〔NGKI3347〕
E_ID	不正ID番号 ・mbfidが有効範囲外〔NGKI3348〕
E_NOEXS	オブジェクト未登録 ・対象メッセージバッファが未登録〔NGKI3349〕
E_OACV	オブジェクトアクセス違反 ・対象メッセージバッファに対する管理操作が許可されていない（P）〔NGKI3350〕
E_OBJ	オブジェクト状態エラー ・対象メッセージバッファは静的APIで生成された〔NGKI3351〕

【機能】

mbfidで指定したメッセージバッファ（対象メッセージバッファ）を削除する．具体的な振舞いは以下の通り．

対象メッセージバッファの登録が解除され，そのメッセージバッファIDが未使用の状態に戻される〔NGKI3352〕．また，対象メッセージバッファ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〔NGKI3353〕．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〔NGKI3354〕．

メッセージバッファの生成時に，メッセージバッファ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〔NGKI3355〕．

【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_mbf□□□

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［NGKI3359］ ・CPUロック状態からの呼出し［NGKI3360］ ・ディスパッチ保留状態からの呼出し（psnd_mbfを除く） ［NGKI3361］
E_NOSPT	未サポート機能 ・制約タスクからの呼出し（psnd_mbfを除く）［NGKI3362］
E_ID	不正ID番号 ・mbfidが有効範囲外［NGKI3363］
E_PAR	パラメータエラー ・msgszが有効範囲（0より大きく対象メッセージバッファの 最大メッセージサイズ以下）外［NGKI3364］ ・tmoutが無効（tsnd_mbfの場合）［NGKI3365］
E_NOEXS	オブジェクト未登録 ・対象メッセージバッファが未登録（D）［NGKI3366］
E_OACV	オブジェクトアクセス違反 ・対象メッセージバッファに対する通常操作1が許可されて いない（P）［NGKI3367］
E_MACV	メモリアクセス違反 ・msgとmsgsz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（P）［NGKI3368］
E_TMOUT	ポーリング失敗またはタイムアウト（snd_mbfを除く）［NGKI3369］
E_RLWAI	待ち禁止状態または待ち状態の強制解除（psnd_mbfを除く） ［NGKI3370］
E_DLT	待ちオブジェクトの削除または再初期化（psnd_mbfを除く） ［NGKI3371］

【機能】

mbfidで指定したメッセージバッファ（対象メッセージバッファ）に，msgと msgszで指定したメッセージ（送信メッセージ）を送信する．具体的な振舞いは 以下の通り．

対象メッセージバッファ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，送信メッセージを受信し，待ち解除される［NGKI3372］．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，受信したメッセージのサイズが返る［NGKI3373］．

対象メッセージバッファの受信待ち行列にタスクが存在しない場合で，送信待ち行列に自タスクより優先してメッセージを送信できるタスクが存在せず，メッセージバッファ管理領域に送信メッセージを格納するスペースがある場合には，送信メッセージが，FIFO順でメッセージバッファ管理領域に格納される［NGKI3374］．ここで，送信待ち行列に自タスクより優先してメッセージを送信できるタスクが存在するとは，送信待ち行列がFIFO順の場合には送信待ち行列に何らかのタスクが存在すること，タスクの優先度順の場合には自タスクと 優先度が同じかより高いタスクが存在することを意味する．

対象メッセージバッファの受信待ち行列にタスクが存在しない場合で，送信待ち行列に自タスクより優先してメッセージを送信できるタスクが存在するか，メッセージバッファ管理領域に送信メッセージを格納するスペースがない場合には，自タスクはメッセージバッファへの送信待ち状態となり，対象メッセージバッファの送信待ち行列につながれる［NGKI3375］．

メッセージバッファの送信待ち行列の先頭につながれているタスクが，ter_tskにより強制終了した場合や，rel_wai／irel_waiやタイムアウトにより待ち解除された場合，新たに送信待ち行列の先頭になったタスクの送信メッセージを，メッセージバッファ管理領域に格納できる可能性がある．そのため，これらの場合には，メッセージバッファからの受信によりメッセージバッファ管理領域に空きができた時の処理［NGKI3393］［NGKI3394］［NGKI3395］と同じ処理が行われる［NGKI3419］．さらに，送信待ち行列がタスクの優先度順の時には，chg_priやミューテックスの操作によりタスクの優先度が変化し，送信待ち行列の先頭につながれているタスクが変わった場合にも，同じ処理が行われる［NGKI3420］．

【使用上の注意】

□□□

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【NGKI3379】 ・CPUロック状態からの呼出し【NGKI3380】 ・ディスパッチ保留状態からの呼出し（prcv_mbfを除く） 【NGKI3381】
E_NOSPT	未サポート機能 ・制約タスクからの呼出し（prcv_mbfを除く）【NGKI3382】
E_ID	不正ID番号 ・mbfidが有効範囲外【NGKI3383】
E_PAR	パラメータエラー ・tmoutが無効（trcv_mbfの場合）【NGKI3384】
E_NOEXS	オブジェクト未登録 ・対象メッセージバッファが未登録（D）【NGKI3385】
E_OACV	オブジェクトアクセス違反 ・対象メッセージバッファに対する通常操作2が許可されてい ない（P）【NGKI3386】
E_MACV	メモリアクセス違反 ・msgを先頭番地とし，対象メッセージバッファの最大メッセー ジサイズ分のメモリ領域への書込みアクセスが許可されて いない（P）【NGKI3387】
E_TMOUT	ポーリング失敗またはタイムアウト（rcv_mbfを除く）【NGKI3388】
E_RLWAI	待ち禁止状態または待ち状態の強制解除（prcv_mbfを除く） 【NGKI3389】
E_DLT	待ちオブジェクトの削除または再初期化（prcv_mbfを除く） 【NGKI3390】

【機能】

mbfidで指定したメッセージバッファ（対象メッセージバッファ）からメッセージを受信する．メッセージの受信に成功した場合，受信したメッセージはmsgを先頭番地とするメモリ領域に格納され，そのサイズはサービスコールの返値として返される【NGKI3391】．具体的な振舞いは以下の通り．

対象メッセージバッファのメッセージバッファ管理領域にメッセージが格納されている場合には，メッセージバッファ管理領域の先頭に格納されたメッセージを受信する【NGKI3392】．また，送信待ち行列にタスクが存在し，メッセージバッファ管理領域に送信待ち行列の先頭のタスクの送信メッセージを格納するスペースがある場合には，送信メッセージが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【NGKI3393】．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【NGKI3394】．この処理を，送信待ち行列にタスクが存在しなくなるか，メッセージバッファ管理領域に送信待ち行列の先頭のタスクの送信メッセージを格納するスペースがなくなるまで繰り返す【NGKI3395】．

対象メッセージバッファのメッセージバッファ管理領域にメッセージ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メッセージを受信する【NGKI3396】．送信待ち行列の先頭のタスクは，待ち解除される【NGKI3397】．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【NGKI3398】．

対象メッセージバッファのメッセージバッファ管理領域にメッセージ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，自タスクはメッセージバッファからの受信待ち状態となり，対象メッセージバッファの受信待ち行列につながれる【NGKI3399】．

【使用上の注意】

メッセージバッファ管理領域に格納されたメッセージを受信した結果，送信待ち行列につながれていたタスクが複数待ち解除される場合がある．この時，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．

ini_mbf	メッセージバッファの再初期化（T）【NGKI3400】

【C言語API】

```
ER ercd = ini_mbf(ID mbfid)
```

【パラメータ】

ID	mbfid	対象メッセージバッファの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【NGKI3401】 ・CPUロック状態からの呼出し【NGKI3402】
E_ID	不正ID番号 ・mbfidが有効範囲外【NGKI3403】
E_NOEXS	オブジェクト未登録 ・対象メッセージバッファが未登録（D）【NGKI3404】
E_OACV	オブジェクトアクセス違反 ・対象メッセージバッファに対する管理操作が許可されていない（P）【NGKI3405】

【機能】

mbfidで指定したメッセージバッファ（対象メッセージバッファ）を再初期化する．具体的な振舞いは以下の通り．

対象メッセージバッファのメッセージバッファ管理領域は，格納されているメッセージがない状態に初期化される【NGKI3406】．また，対象メッセージバッファ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【NGKI3407】．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【NGKI3408】．

【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_mbf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．

メッセージバッファ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】【】【】【】【】

μITRON4.0【】【】【】【】

□□□□

mbfid□□□□

スピンロックを取得した状態でCPU例外が発生した場合，起動されるCPU例外ハンドラはカーネル管理外のCPU例外ハンドラであり（xsns_dpn，xsns_xpnともtrueを返す），CPU例外ハンドラ中でiunl_spnを呼び出してスピンロックを返却しようとした場合の動作は保証されない．保証されないにも関わらずiunl_spnを呼び出した場合には，CPU例外ハンドラからのリターン時に元の状態に戻らない．これは，CPUロック状態の扱いと一貫していないため，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，スピンロック機能をサポートしない【ASPS0163】．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，スピンロック機能をサポートする【FMPS0138】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，スピンロック機能をサポートしない【HRPS0135】．

【μITRON4.0仕様との関係】

スピンロック機能は，μITRON4.0仕様に定義されていない機能である．

```
CRE_SPN      スピンロックの生成（SM）【NGKI2128】
acre_spn     スピンロックの生成（TMD）【NGKI2129】
```

【静的API】

```
    CRE_SPN(ID spnid, { ATR spnatr })
```

【C言語API】

```
    ER_ID spnid = acre_spn(const T_CSPN *pk_cspn)
```

【パラメータ】

```
    ID          spnid       生成するスピンロックのID番号（CRE_SPNの場合）
    T_CSPN *    pk_cspn     スピンロッ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）

 ＊スピンロックの生成情報（パケットの内容）
    ATR         spnatr      スピンロック属性
```

【リターンパラメータ】

```
    ER_ID       spnid       生成されたスピンロックの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（S）【NGKI2130】 ・CPUロック状態からの呼出し（S）【NGKI2131】
E_RSATR	予約属性 ・spnatrが無効【NGKI2132】 ・属する保護ドメインの指定が有効範囲外（SP）【NGKI2133】 ・属するクラスの指定が有効範囲外（S）【NGKI2134】 ・クラスの囲みの中に記述されていない（S）【NGKI2135】
E_OACV	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（SP） 【NGKI2136】
E_MACV	メモリアクセス違反 ・pk_cspn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（SP）【NGKI2137】
E_NOID	ID番号不足 ・割り付けられるスピンロックIDがない（SD）【NGKI2138】
E_NORES	資源不足 ・条件については機能の項を参照
E_OBJ	オブジェクト状態エラー ・spnidで指定したスピンロックが登録済み（CRE_SPNの場合） 【NGKI2139】

【機能】

各パラメータで指定したスピンロック生成情報に従って，スピンロックを生成 する．生成されたスピンロックのロック状態は，取得されていない状態に初期 化される【NGKI2140】．

静的APIにおいては，spnidはオブジェクト識別名，spnatrは整数定数式パラメー タである【NGKI2141】．

スピンロックをハードウェアによって実現している場合には，ターゲット定義 で，生成できるスピンロックの数に上限がある【NGKI2142】．この上限を超えてスピンロックを生成しようとした場合には，E_NORESエラーとなる【NGKI2143】．

【補足説明】

スピンロックを動的に生成する場合に，生成できるスピンロックの数の上限は AID_SPNによってチェックされるため，acre_spnでE_NORESエラーが返ることは ない．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，CRE_SPNのみをサポートする【FMPS0139】．

AID_SPN	割付け可能なスピンロックIDの数の指定（SMD）【NGKI2144】

【静的API】

```
AID_SPN(uint_t nospn)
```

【パラメータ】

```
uint_t      nospn       割付け可能なスピンロックIDの数
```

【エラーコード】

```
E_RSATR     予約属性
            ・保護ドメインの囲みの中に記述されている（P）【NGKI3435】
            ・クラスの囲みの中に記述されていない【NGKI2145】
E_PAR       パラメータエラー
            ・nospnが負の値【NGKI3283】
```

【機能】

nospnで指定した数のスピンロックIDを，スピンロックを生成するサービスコールによって割付け可能なスピンロックIDとして確保する【NGKI2146】．

nospnは整数定数式パラメータである【NGKI2147】．

```
SAC_SPN     スピンロックのアクセス許可ベクタの設定（SPM）【NGKI2148】
sac_spn     スピンロックのアクセス許可ベクタの設定（TPMD）【NGKI2149】
```

【静的API】

```
SAC_SPN(ID spnid, { ACPTN acptn1, ACPTN acptn2,
                                    ACPTN acptn3, ACPTN acptn4 })
```

【C言語API】

```
ER ercd = sac_spn(ID spnid, const ACVCT *p_acvct)
```

【パラメータ】

```
ID          spnid       対象スピンロックの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（S）〔NGKI2150〕 ・CPUロック状態からの呼出し（S）〔NGKI2151〕
E_ID	不正ID番号 ・spnidが有効範囲外（S）〔NGKI2152〕
E_RSATR	予約属性 ・対象スピンロックが属する保護ドメインの囲みの中（対象 スピンロックが無所属の場合は，保護ドメインの囲みの外） に記述されていない（S）〔NGKI2153〕 ・対象スピンロックが属するクラスの囲みの中に記述されて いない（S）〔NGKI2154〕
E_NOEXS	オブジェクト未登録 ・対象スピンロックが未登録〔NGKI2155〕
E_OACV	オブジェクトアクセス違反 ・対象スピンロックに対する管理操作が許可されていない（S） 〔NGKI2156〕
E_MACV	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（S）〔NGKI2157〕
E_OBJ	オブジェクト状態エラー ・対象スピンロックは静的APIで生成された（S）〔NGKI2158〕 ・対象スピンロックに対してアクセス許可ベクタが設定済み（S） 〔NGKI2159〕

【機能】

spnidで指定したスピンロック（対象スピンロック）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〔NGKI2160〕．

静的APIにおいては，spnidはオブジェクト識別名，acptn1~acptn4は整数定数式パラメータである〔NGKI2161〕．

del_spn		スピンロックの削除（TMD）〔NGKI2162〕

〔C言語API〕

```
ER ercd = del_spn(ID spnid)
```

【パラメータ】

ID	spnid	対象スピンロックの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［NGKI2163］ ・CPUロック状態からの呼出し［NGKI2164］
E_ID	不正ID番号 ・spnidが有効範囲外［NGKI2165］
E_NOEXS	オブジェクト未登録 ・対象スピンロックが未登録［NGKI2166］
E_OACV	オブジェクトアクセス違反 ・対象スピンロックに対する管理操作が許可されていない（P ）［NGKI2167］
E_OBJ	オブジェクト状態エラー ・対象スピンロックは静的APIで生成された［NGKI2168］

【機能】

spnidで指定したスピンロック（対象スピンロック）を削除する．具体的な振舞いは以下の通り．

対象スピンロックの登録が解除され，そのスピンロックIDが未使用の状態に戻される［NGKI2169］．

［TOPPERS/FMPカーネルにおける規定］

FMPカーネルでは，del_spnをサポートしない［FMPS0141］．

【未決定事項】

対象スピンロックが取得されている状態の場合の振舞いは，今後の課題である．

loc_spn	スピンロックの取得（TM）	［NGKI2170］
iloc_spn	スピンロックの取得（IM）	［NGKI2171］

［C言語API］

```
ER ercd = loc_spn(ID spnid)
ER ercd = iloc_spn(ID spnid)
```

【パラメータ】

ID	spnid	対象スピンロックの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（loc_spnの場合）〔NGKI2172〕 ・タスクコンテキストからの呼出し（iloc_spnの場合）〔NGKI2173〕 ・CPUロック状態からの呼出し〔NGKI2174〕
E_ID	不正ID番号 ・spnidが有効範囲外〔NGKI2175〕
E_NOEXS	オブジェクト未登録 ・対象スピンロックが未登録（D）〔NGKI2176〕
E_OACV	オブジェクトアクセス違反 ・対象スピンロックに対する通常操作1が許可されていない （loc_spnの場合）（P）〔NGKI2177〕

【機能】

spnidで指定したスピンロック（対象スピンロック）を取得する．具体的な振舞いは以下の通り．

対象スピンロックが取得されていない状態である場合には，プロセッサ間ロックの取得を試みる〔NGKI2178〕．ロックが他のプロセッサによって取得されている状態である場合や，他のプロセッサがロックの取得に成功した場合には，ロックが返却されるまでループによって待ち，返却されたらロックの取得を試みる〔NGKI2179〕．これを，ロックの取得に成功するまで繰り返す〔NGKI2180〕．

ロックの取得に成功した場合には，スピンロックは取得されている状態になる〔NGKI2181〕．また，CPUロックフラグをセットしてCPUロック状態へ遷移し，サービスコールからリターンする〔NGKI2182〕．

なお，複数のプロセッサがロックの取得を待っている時に，どのプロセッサが最初にロックを取得できるかは，現時点ではターゲット定義とする〔NGKI2183〕．

【補足説明】

対象スピンロックが，loc_spn/iloc_spnを呼び出したプロセッサによって取得されている状態である場合には，スピンロックの取得によりCPUロック状態になっているため，loc_spn/iloc_spnはE_CTXエラーとなる．

プロセッサがロックを取得できる順序を，現時点ではターゲット定義としたが，リアルタイム性保証のためには，（ロックの取得待ちの間に割込みが発生しない限りは）loc_spn/iloc_spnを呼び出した順序でロックを取得できるとするのが望ましい．ただし，ターゲットハードウェアの制限で，そのような実装ができるとは限らないため，現時点ではターゲット定義としている．

try_spn	スピンロックの取得（ポーリング）（TM）〔NGKI2184〕	
itry_spn	スピンロックの取得（ポーリング）（IM）〔NGKI2185〕	

〔C言語API〕

```
ER ercd = try_spn(ID spnid)
ER ercd = itry_spn(ID spnid)
```

【パラメータ】

ID	spnid	対象スピンロックの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（try_spnの場合）〔NGKI2186〕 ・タスクコンテキストからの呼出し（itry_spnの場合）〔NGKI2187〕 ・CPUロック状態からの呼出し〔NGKI2188〕
E_ID	不正ID番号 ・spnidが有効範囲外〔NGKI2189〕
E_NOEXS	オブジェクト未登録 ・対象スピンロックが未登録（D）〔NGKI2190〕
E_OACV	オブジェクトアクセス違反 ・対象スピンロックに対する通常操作1が許可されていない （try_spnの場合）（P）〔NGKI2191〕
E_OBJ	オブジェクト状態エラー ・条件については機能の項を参照

【機能】

spnidで指定したスピンロック（対象スピンロック）の取得を試みる．具体的な 振舞いは以下の通り．

対象スピンロックが取得されていない状態である場合には，プロセッサ間ロックの取得を試みる〔NGKI2192〕．ロックの取得に成功した場合には，スピンロックは取得されている状態になる〔NGKI2193〕．また，CPUロックフラグをセット してCPUロック状態へ遷移し，サービスコールからリターンする〔NGKI2194〕．

対象スピンロックが他のプロセッサによって取得されている状態である場合や，ロックの取得に失敗した場合（他のプロセッサがロックの取得に成功した場合）には，E_OBJエラーとする〔NGKI2195〕．

【使用上の注意】

try_spn/itry_spnを，ロックの取得に成功するまで繰り返し呼び出すことによりスピンロックを取得する方法は，loc_spn/iloc_spnによりスピンロックを取得する方法と，プロセッサがロックを取得できる順序が異なる可能性ある．

unl_spn	スピンロックの返却（TM）〔NGKI2196〕
iunl_spn	スピンロックの返却（IM）〔NGKI2197〕

〔C言語API〕

```
ER ercd = unl_spn(ID spnid)
ER ercd = iunl_spn(ID spnid)
```

【パラメータ】

ID	spnid	対象スピンロックの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（unl_spnの場合）〔NGKI2198〕 ・タスクコンテキストからの呼出し（iunl_spnの場合）〔NGKI2199〕
E_ID	不正ID番号 ・spnidが有効範囲外〔NGKI2200〕
E_NOEXS	オブジェクト未登録 ・対象スピンロックが未登録（D）〔NGKI2201〕
E_OACV	オブジェクトアクセス違反 ・対象スピンロックに対する通常操作1が許可されていない （unl_spnの場合）（P）〔NGKI2202〕
E_ILUSE	サービスコール不正使用 ・条件については機能の項を参照

【機能】

spnidで指定したスピンロック（対象スピンロック）を返却する．具体的な振舞いは以下の通り．

対象スピンロックが，unl_spn／iunl_spnを呼び出したプロセッサによって取得されている状態である場合には，ロックを返却し，スピンロックを取得されていない状態とする〔NGKI2203〕．また，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〔NGKI2204〕．

対象スピンロックが，取得されていない状態である場合や，他のプロセッサによって取得されている状態である場合には，E_ILUSEエラーとなる〔NGKI2205〕．

[[API_ref_spn]

ref_spn	スピンロックの状態参照（TM）〔NGKI2206〕	

〔C言語API〕

```
ER ercd = ref_spn(ID spnid, T_RSPN *pk_rspn)
```

【パラメータ】

```
    ID          spnid       対象スピンロックのID番号
    T_RSPN *    pk_rspn     スピンロッ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

□*スピンロックの現在状態（パケットの内容）
    STAT        spnstat     ロック状態
```

【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〔NGKI2207〕
                ・CPUロック状態からの呼出し〔NGKI2208〕
    E_ID        不正ID番号
                ・spnidが有効範囲外〔NGKI2209〕
    E_NOEXS     オブジェクト未登録
                ・対象スピンロックが未登録（D）〔NGKI2210〕
    E_OACV      オブジェクトアクセス違反
                ・対象スピンロックに対する参照操作が許可されていない（P）
                □〔NGKI2211〕
    E_MACV      メモリアクセス違反
                ・pk_rspn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）（P）〔NGKI2212〕
```

【機能】

spnidで指定したスピンロック（対象スピンロック）の現在状態を参照する．参照した現在状態は，pk_rspnで指定したパケットに返される〔NGKI2213〕．

spnstatには，対象スピンロックの現在のロック状態を表す次のいずれかの値が返される〔NGKI2214〕．

```
TSPN_UNL    0x01U       取得されていない状態
TSPN_LOC    0x02U       取得されている状態
```

【使用上の注意】

ref_spn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_spnを呼び出し，対象スピンロッ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_spnから戻ってきた時には対象スピンロックの状態が変化している可能性があるためである．

4.5. メモリプール管理機能

〔TOPPERS/SSPカーネルにおける規定〕

SSPカーネルでは，メモリプール管理機能をサポートしない〔SSPS0128〕．

【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・返却するための同期・通信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【NGKI2215】．

各固定長メモリプールが持つ情報は次の通り【NGKI2216】．

- 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
- アクセス許可ベクタ（保護機能対応カーネルの場合）
- 属する保護ドメイン（保護機能対応カーネルの場合）
- 属するクラス（マルチプロセッサ対応カーネルの場合）

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．

保護機能対応カーネルにおいて，固定長メモリプール管理領域は，カーネルの用いるオブジェクト管理領域として扱われる【NGKI2217】．

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【NGKI2218】．

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【NGKI2219】．

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）【NGKI2220】

【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アプリケーショ

ンで確保する方法を定めた結果，そのサイズは(blkcnt * ROUND_MPF_T(blksz)) で求めることができるようになったためである．

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〕【NGKI2221】
acre_mpf     固定長メモリプールの生成〔TD〕【NGKI2222】
```

【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（CRE_MPF
                            の場合）
    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（s）【NGKI2223】 ・CPUロック状態からの呼出し（s）【NGKI2224】
E_RSATR	予約属性 ・mpfatrが無効【NGKI2225】 ・属する保護ドメインの指定が有効範囲外（sP）【NGKI2226】 ・属するクラスの指定が有効範囲外（sM）【NGKI2227】 ・クラスの囲みの中に記述されていない（SM）【NGKI2228】
E_NOSPT	未サポート機能 ・条件については各カーネルにおける規定の項を参照
E_PAR	パラメータエラー ・blkcntが0以下【NGKI2229】 ・blkszが0以下【NGKI2230】 ・その他の条件については機能の項を参照
E_OACV	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（sP）□【NGKI2231】
E_MACV	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（sP）【NGKI2232】
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（sD）【NGKI2233】
E_NOMEM	メモリ不足 ・固定長メモリプール領域が確保できない【NGKI2234】 ・固定長メモリプール管理領域が確保できない【NGKI2235】
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（CRE_MPF□の場合）【NGKI2236】 ・その他の条件については機能の項を参照

【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，blkszから固定長メモリプール領域が，mpfmbとblkcnt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【NGKI2237】．また，待ち行列は空の状態に初期化される【NGKI2238】．

静的APIにおいては，mpfidはオブジェクト識別名，mpfatr，blkcnt，blksz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【NGKI2239】．コンフィギュレータは，静的APIのメモリ不足（E_NOMEM）エラーを検出することができない【NGKI2240】．

mpfをNULLとした場合，blkcntとblkszから決まるサイズの固定長メモリプール 領域が，コンフィギュレータまたはカーネルにより確保される【NGKI2241】．

保護機能対応カーネルでは，コンフィギュレータまたはカーネルにより確保される固定長メモリプール領域は，固定長メモリプールと同じ保護ドメインに属し，固定長メモリプールと同じアクセス許可ベクタを持ったメモリオブジェクト中に確保される【NGKI2242】．

mpfmbをNULLとした場合，blkcntから決まるサイズの固定長メモリプール管理領 域が，コンフィギュレータまたはカーネルにより確保される【NGKI2243】．

〔mpfにNULL以外を指定した場合〕

mpfにNULL以外を指定した場合，mpfを先頭番地とする固定長メモリプール領域は，アプリケーションで確保しておく必要がある【NGKI2244】．固定長メモリ

□□□

□□□

E_RSATR	予約属性 ・保護ドメインの囲みの中に記述されている（P）【NGKI3436】 ・クラスの囲みの中に記述されていない（M）【NGKI2259】
E_PAR	パラメータエラー ・nompfが負の値【NGKI3284】

【機能】

nompfで指定した数の固定長メモリプールIDを，固定長メモリプールを生成するサービスコールによって割付け可能な固定長メモリプールIDとして確保する【NGKI2260】．

nompfは整数定数式パラメータである【NGKI2261】．

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルの動的生成機能拡張パッケージでは，AID_MPFをサポートする【ASPS0216】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルの動的生成機能拡張パッケージでは，AID_MPFをサポートする【HRPS0217】．

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）【NGKI2262】
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）【NGKI2263】

【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（S）【NGKI2264】
            ・CPUロック状態からの呼出し（S）【NGKI2265】
E_ID        不正ID番号
            ・mpfidが有効範囲外（S）【NGKI2266】
E_RSATR     予約属性
            ・対象固定長メモリプールが属する保護ドメインの囲みの中
            （対象固定長メモリプールが無所属の場合は，保護ドメイ
            ンの囲みの外）に記述されていない（S）【NGKI2267】
            ・対象固定長メモリプールが属するクラスの囲みの中に記述
            されていない（SM）【NGKI2268】
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【NGKI2269】
E_OACV      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（S）【NGKI2270】
E_MACV      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（S）【NGKI2271】
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（S）【NGKI2272】
            ・対象固定長メモリプールに対してアクセス許可ベクタが設
            定済み（S）【NGKI2273】
```

【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【NGKI2274】．対象固定長メモリプールの固定長メモリプール領域がコンフィギュレータまたはカーネルにより確保されたものである場合には，固定長メモリプール領域のアクセス許可ベクタも，各パラメータで指定した値に 設定する【NGKI2275】．

静的APIにおいては，mpfidはオブジェクト識別名，acptn1~acptn4は整数定数 式パラメータである【NGKI2276】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，SAC_MPFのみをサポートする【HRPS0139】．ただし，動的生 成機能拡張パッケージでは，sac_mpfもサポートする【HRPS0200】．

```
del_mpf     固定長メモリプールの削除（TD）【NGKI2277】
```

【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【NGKI2278】 ・CPUロック状態からの呼出し【NGKI2279】
E_ID	不正ID番号 ・mpfidが有効範囲外【NGKI2280】
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【NGKI2281】
E_OACV	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（P）【NGKI2282】
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【NGKI2283】

【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【NGKI2284】．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【NGKI2285】．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【NGKI2286】．

【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_mpfをサポートしない【ASPS0170】．ただし，動的生成機能拡張パッケージでは，del_mpfをサポートする【ASPS0171】．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，del_mpfをサポートしない【FMPS0146】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，del_mpfをサポートしない【HRPS0140】．ただし，動的生成機能拡張パッケージでは，del_mpfをサポートする【HRPS0201】．

get_mpf	固定長メモリブロックの獲得（T）［NGKI2287］	
pget_mpf	固定長メモリブロックの獲得（ポーリング）（T）［NGKI2288］	
tget_mpf	固定長メモリブロックの獲得（タイムアウト付き）（T）［NGKI2289］	

［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〔NGKI2290〕 ・CPUロック状態からの呼出し〔NGKI2291〕 ・ディスパッチ保留状態からの呼出し（pget_mpfを除く） 〔NGKI2292〕
E_NOSPT	未サポート機能 ・制約タスクからの呼出し（pget_mpfを除く）〔NGKI2293〕
E_ID	不正ID番号 ・mpfidが有効範囲外〔NGKI2294〕
E_PAR	パラメータエラー ・tmoutが無効（tget_mpfの場合）〔NGKI2295〕
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〔D〕〔NGKI2296〕
E_OACV	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〔P〕〔NGKI2297〕
E_MACV	メモリアクセス違反 ・p_blk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〕〔P〕〔NGKI2298〕
E_TMOUT	ポーリング失敗またはタイムアウト（get_mpfを除く）〔NGKI2299〕
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） 〔NGKI2300〕
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） 〔NGKI2301〕

【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p_blkが指すメモリ領域に返す．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〔NGKI2302〕．

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れる〔NGKI2303〕．

rel_mpf	固定長メモリブロックの返却〔T〕〔NGKI2304〕

〔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【NGKI2305】 ・CPUロック状態からの呼出し【NGKI2306】
E_ID	不正ID番号 ・mpfidが有効範囲外【NGKI2307】
E_PAR	パラメータエラー ・条件については機能の項を参照
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（D）【NGKI2308】
E_OACV	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（P）【NGKI2309】

【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【NGKI2310】．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【NGKI2311】．

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【NGKI2312】．

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【NGKI2313】．

ini_mpf	固定長メモリプールの再初期化（T）【NGKI2314】	

【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〔NGKI2315〕 ・CPUロック状態からの呼出し〔NGKI2316〕
E_ID	不正ID番号 ・mpfidが有効範囲外〔NGKI2317〕
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（D）〔NGKI2318〕
E_OACV	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（P）〔NGKI2319〕

【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〔NGKI2320〕．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〔NGKI2321〕．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〔NGKI2322〕．

【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）〔NGKI2323〕

〔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【NGKI2324】 ・CPUロック状態からの呼出し【NGKI2325】
E_ID	不正ID番号 ・mpfidが有効範囲外【NGKI2326】
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（D）【NGKI2327】
E_OACV	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（P）【NGKI2328】
E_MACV	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（P）【NGKI2329】

【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【NGKI2330】．

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【NGKI2331】．

【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【NGKI3603】．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【NGKI2332】．

システム時刻は，カーネルの初期化時に0に初期化される【NGKI2333】．タイムティックを通知するためのタイマ割込みが発生する毎にカーネルによって更新され，SYSTIM型で表せる最大値（ULONG_MAX）を超えると0に戻される【NGKI2334】．タイムティックの周期は，ターゲット定義である【NGKI2335】．また，システム時刻の精度はターゲットに依存する【NGKI2336】．

マルチプロセッサ対応でないカーネルと，マルチプロセッサ対応カーネルでグローバルタイマ方式を用いている場合には，システム時刻は，システムに1つの

み存在する【NGKI2337】．マルチプロセッサ対応カーネルでローカルタイマ方式を用いている場合には，システム時刻は，プロセッサ毎に存在する【NGKI2338】．ローカルタイマ方式とグローバルタイマ方式については，「2.3.4 マルチプロセッサ対応」の節を参照すること．

マルチプロセッサ対応カーネルでローカルタイマ方式を用いている場合には，タイムアウト処理とタスクの遅延処理には，待ち解除されるタスクが割り付けられているプロセッサのシステム時刻が用いられる【NGKI2339】．また，周期ハンドラとアラームハンドラの起動には，それが割り付けられているプロセッサのシステム時刻が用いられる【NGKI2340】．これらの処理単位がマイグレーションする場合には，用いられるシステム時刻も変更される【NGKI2341】．この場合にも，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という規則は維持される【NGKI2342】．

1回のタイムティックの発生により，複数のイベント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【NGKI2343】．

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【NGKI2344】．ただし，実際の精度はターゲットに依存する【NGKI2345】．

マルチプロセッサ対応カーネルにおける性能評価用システム時刻の扱いは，ターゲット定義とする【NGKI2346】．

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	【NGKI2347】
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【NGKI2348】

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは，時間管理機能をサポートしない【SSPS0129】．

【使用上の注意】

タイムティックを通知するためのタイマ割込みが長時間マスクされた場合（タイマ割込みより優先して実行される割込み処理が長時間続けて実行された場合を含む）や，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 スケジュールされなかった場合には，システム時刻が正しく更新されない可能 性があるため，注意が必要である．

【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．

【μITRON4.0/PX仕様との関係】

システム時刻のアクセス許可ベクタは廃止し，システム状態のアクセス許可ベクタで代替することとした．そのため，システム時刻のアクセス許可ベクタを設定する静的API（SAC_TIM）とサービスコール（sac_tim）は廃止した．

get_tim	システム時刻の参照（T）	【NGKI2349】

【C言語API】

```
ER ercd = get_tim(SYSTIM *p_systim)
```

【パラメータ】

SYSTIM *	p_systim	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	systim	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【NGKI2350】 ・CPUロック状態からの呼出し【NGKI2351】
E_OACV	オブジェクトアクセス違反 ・システム状態に対する参照操作が許可されていない（P） 【NGKI2352】
E_MACV	メモリアクセス違反 ・p_systim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（P）【NGKI2353】

【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_systimが指すメモリ領域に返される【NGKI2354】．

マルチプロセッサ対応カーネルでローカルタイマ方式を用いている場合には，自タスクが割り付けられているプロセッサのシステム時刻の現在値を参照する【NGKI2355】．

【補足説明】

マルチプロセッサ対応カーネルでローカルタイマ方式を用いている場合に，他のプロセッサのシステム時刻の現在値を参照する機能は用意していない．

get_utm	性能評価用システム時刻の参照（TI）【NGKI2356】	

【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 ・条件については機能の項を参照
E_MACV	メモリアクセス違反 ・p_sysutmが指すメモリ領域へ書込みアクセスが許可されて □いない）（P）〔NGKI2357〕

【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が指すメモリ領域に返される〔NGKI2358〕．

get_utmは，任意の状態から呼び出すことができる〔NGKI2359〕．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〔NGKI2360〕．
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〔NGKI2361〕．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〔NGKI2362〕．

【使用方法】

get_utmを使用してプログラムの処理時間を計測する場合には，次の手順を取る． 処理時間を計測したいプログラムの実行直前と実行直後に，get_utm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 処理時間に，get_utm自身の処理時間を加えたものが得られる．

マルチプロセッサ対応カーネルにおいては，異なるプロセッサで読み出した性 能評価用システム時刻の差を求めることで，処理時間が正しく計測できるとは 限らない．

【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．

get_utmは，任意の状態から呼び出すことができるように，全割込みロック状態 を用いて実装されている．そのため，get_utmを用いると，カーネル管理外の割 込みの応答性が低下する．

システム時刻が正しく更新されない状況では，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行わ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〔NGKI2363〕．

各周期ハンドラが持つ情報は次の通り〔NGKI2364〕．

- 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時刻
- 拡張情報
- 周期ハンドラの先頭番地
- 起動周期
- 起動位相
- アクセス許可ベクタ（保護機能対応カーネルの場合）
- 属する保護ドメイン（保護機能対応カーネルの場合）
- 属するクラス（マルチプロセッサ対応カーネルの場合）

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【NGKI2365】．

```
起動位相+起動周期x(n-1)      n=1,2,…
```

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【NGKI2366】．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，周期ハンドラを起動する時刻に なると，周期ハンドラの起動処理が行われる【NGKI2367】．具体的には，拡張情報をパラメータとして，周期ハンドラが呼び出される【NGKI2368】．

保護機能対応カーネルにおいて，周期ハンドラが属することのできる保護ドメインは，カーネルドメインに限られる【NGKI2369】．

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【NGKI2370】．

```
TA_STA      0x02U   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      0x04U   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【NGKI2371】．

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【NGKI2372】．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）が，基準時刻となる【NGKI2373】．

次に周期ハンドラを起動する時刻は，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【NGKI2374】．

マルチプロセッサ対応カーネルでグローバルタイマ方式を用いている場合には， 周期ハンドラは，システム時刻管理プロセッサのみが割付け可能プロセッサであるクラスにのみ属することができる【NGKI2375】．すなわち，周期ハンドラ は，システム時刻管理プロセッサによって実行される．

C言語による周期ハンドラの記述形式は次の通り【NGKI2376】．

```
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【NGKI2377】．

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）【NGKI2378】

【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【ASPS0172】．

【TOPPERS/FMPカーネルにおける規定】

FMP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【FMPS0147】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2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【HRPS0141】．

【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）【NGKI2379】
acre_cyc	周期ハンドラの生成（TD）【NGKI2380】

【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（CRE_CYCの場合）
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	cychdr	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（sS）［NGKI2381］
	・CPUロック状態からの呼出し（sS）［NGKI2382］
E_RSATR	予約属性
	・cycatrが無効［NGKI2383］
	・属する保護ドメインの指定が有効範囲外またはカーネルド □メイン以外（sP）［NGKI2384］
	・カーネルドメインの囲みの中に記述されていない（SP） □［NGKI2385］
	・属するクラスの指定が有効範囲外（sM）［NGKI2386］
	・クラスの囲みの中に記述されていない（SM）［NGKI2387］
	・その他の条件については機能の項を参照
E_PAR	パラメータエラー
	・cychdrがプログラムの先頭番地として正しくない［NGKI2388］
	・cyctimが有効範囲（0より大きくTMAX_RELTIM以下）外［NGKI2397］
	・cycphsが有効範囲（0以上TMAX_RELTIM以下）外［NGKI2399］
E_OACV	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（sP） □［NGKI2389］
E_MACV	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（sP）［NGKI2390］
E_NOID	ID番号不足
	・割り付けられる周期ハンドラIDがない（sD）［NGKI2391］
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（CRE_CYCの場合） □［NGKI2392］

【機能】

□□□

【エラーコード】

E_RSATR	予約属性 ・保護ドメインの囲みの中に記述されている（P）【NGKI3437】 ・クラスの囲みの中に記述されていない（M）【NGKI2404】 ・その他の条件については機能の項を参照
E_PAR	パラメータエラー ・nocycが負の値【NGKI3285】

【機能】

nocycで指定した数の周期ハンドラIDを，周期ハンドラを生成するサービスコールによって割付け可能な周期ハンドラIDとして確保する【NGKI2405】．

nocycは整数定数式パラメータである【NGKI2406】．

マルチプロセッサ対応カーネルでグローバルタイマ方式を用いている場合で，AID_CYCが属するクラスの割付け可能プロセッサが，システム時刻管理プロセッサのみでない場合には，E_RSATRエラーとなる【NGKI2407】．

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルの動的生成機能拡張パッケージでは，AID_CYCをサポートする【ASPS0217】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルの動的生成機能拡張パッケージでは，AID_CYCをサポートする【HRPS0218】．

SAC_CYC	周期ハンドラのアクセス許可ベクタの設定（SP）【NGKI2408】
sac_cyc	周期ハンドラのアクセス許可ベクタの設定（TPD）【NGKI2409】

【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（S）【NGKI2410】 ・CPUロック状態からの呼出し（S）【NGKI2411】
E_ID	不正ID番号 ・cycidが有効範囲外（S）【NGKI2412】
E_RSATR	予約属性 ・カーネルドメインの囲みの中に記述されていない（S）【NGKI2413】 ・対象周期ハンドラが属するクラスの囲みの中に記述されていない（SM）【NGKI2414】
E_NOEXS	オブジェクト未登録 ・対象周期ハンドラが未登録【NGKI2415】
E_OACV	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（S）【NGKI2416】
E_MACV	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（S）【NGKI2417】
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（S）【NGKI2418】 ・対象周期ハンドラに対してアクセス許可ベクタが設定済み（S）【NGKI2419】

【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【NGKI2420】．

静的APIにおいては，cycidはオブジェクト識別名，acptn1~acptn4は整数定数 式パラメータである【NGKI2421】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，SAC_CYCのみをサポートする【HRPS0144】．ただし，動的生 成機能拡張パッケージでは，sac_cycもサポートする【HRPS0203】．

del_cyc	周期ハンドラの削除〔TD〕〔NGKI2422〕

〔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〔NGKI2423〕 ・CPUロック状態からの呼出し〔NGKI2424〕
E_ID	不正ID番号 ・cycidが有効範囲外〔NGKI2425〕
E_NOEXS	オブジェクト未登録 ・対象周期ハンドラが未登録〔NGKI2426〕
E_OACV	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（P） 〔NGKI2427〕
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〔NGKI2428〕

【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〔NGKI2429〕．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〔NGKI2430〕．

〔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〔ASPS0177〕．ただし，動的生成機能拡張パッケージでは，del_cycをサポートする〔ASPS0178〕．

〔TOPPERS/FMPカーネルにおける規定〕

FMPカーネルでは，del_cycをサポートしない〔FMPS0151〕．

〔TOPPERS/HRP2カーネルにおける規定〕

HRP2カーネルでは，del_cycをサポートしない〔HRPS0145〕．ただし，動的生成機能拡張パッケージでは，del_cycをサポートする〔HRPS0204〕．

sta_cyc	周期ハンドラの動作開始〔T〕〔NGKI2431〕

〔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〔NGKI2432〕 ・CPUロック状態からの呼出し〔NGKI2433〕
E_ID	不正ID番号 ・cycidが有効範囲外〔NGKI2434〕
E_NOEXS	オブジェクト未登録 ・対象周期ハンドラが未登録（D）〔NGKI2435〕
E_OACV	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（P） 〔NGKI2436〕

【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る〔NGKI2437〕．次に周期ハンドラを起動する時刻は，sta_cycを呼び出して以降の最初の起動時刻に設定される〔NGKI2438〕．

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時刻の再設定のみが行われる〔NGKI2439〕．

【補足説明】

TA_PHS属性でない周期ハンドラの場合，次に周期ハンドラを起動する時刻は，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．

対象周期ハンドラがTA_PHS属性で，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．

msta_cyc	割付けプロセッサ指定での周期ハンドラの動作開始〔TM〕〔NGKI2440〕

〔C言語API〕

```
ER ercd = msta_cyc(ID cycid, ID prcid)
```

【パラメータ】

ID	cycid	対象周期ハンドラのID番号
ID	prcid	周期ハンドラの割付け対象のプロセッサの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〔NGKI2441〕 ・CPUロック状態からの呼出し〔NGKI2442〕
E_NOSPT	未サポート機能 ・条件については機能の項を参照
E_ID	不正ID番号 ・cycidが有効範囲外〔NGKI2443〕 ・prcidが有効範囲外〔NGKI2444〕
E_PAR	パラメータエラー ・条件については機能の項を参照
E_NOEXS	オブジェクト未登録 ・対象周期ハンドラが未登録〔D〕〔NGKI2445〕
E_OACV	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〔P〕 〔NGKI2446〕

【機能】

prcidで指定したプロセッサを割付けプロセッサとして，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，対象周期ハンドラの割付け プロセッサがprcidで指定したプロセッサに変更された後，対象周期ハンドラは動作している状態となる〔NGKI2447〕．次に周期ハンドラを起動する時刻は， msta_cycを呼び出して以降の最初の起動時刻に設定される〔NGKI2448〕．

対象周期ハンドラが動作している状態であれば，対象周期ハンドラの割付けプ ロセッサがprcidで指定したプロセッサに変更された後，次に周期ハンドラを起動する時刻の再設定が行われる〔NGKI2449〕．

□□□

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〔NGKI2456〕 ・CPUロック状態からの呼出し〔NGKI2457〕
E_ID	不正ID番号 ・cycidが有効範囲外〔NGKI2458〕
E_NOEXS	オブジェクト未登録 ・対象周期ハンドラが未登録〔D〕〔NGKI2459〕
E_OACV	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〔P〕 〔NGKI2460〕

【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〔NGKI2461〕．対象周期ハンドラが動作していない状態であれば，何も行われずに正常終了する〔NGKI2462〕．

ref_cyc	周期ハンドラの状態参照〔T〕〔NGKI2463〕	

〔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
ID	prcid	周期ハンドラの割付けプロセッサのID（マルチプロセッサ対応カーネルの場合）

【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〔NGKI2464〕 ・CPUロック状態からの呼出し〔NGKI2465〕
E_ID	不正ID番号 ・cycidが有効範囲外〔NGKI2466〕
E_NOEXS	オブジェクト未登録 ・対象周期ハンドラが未登録（D）〔NGKI2467〕
E_OACV	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（P） 〔NGKI2468〕
E_MACV	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）（P）〔NGKI2469〕

【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〔NGKI2470〕．

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〔NGKI2471〕．

TCYC_STP	0x01U	周期ハンドラが動作していない状態
TCYC_STA	0x02U	周期ハンドラが動作している状態

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〔NGKI2472〕．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〔NGKI2473〕．

マルチプロセッサ対応カーネルでは，prcidに，対象周期ハンドラの割付けプロセッサのID番号が返される〔NGKI2474〕．

【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直後に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〔NGKI2475〕．

各アラームハンドラが持つ情報は次の通り〔NGKI2476〕．

- 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時刻
- 拡張情報

- アラームハンドラの先頭番地
- アクセス許可ベクタ（保護機能対応カーネルの場合）
- 属する保護ドメイン（保護機能対応カーネルの場合）
- 属するクラス（マルチプロセッサ対応カーネルの場合）

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【NGKI2477】．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時刻は，アラームハンドラを動作開始する時に設定される【NGKI2478】．

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時刻になると，アラームハンドラの起動処理が行われる【NGKI2479】．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【NGKI2480】．その後に，拡張情報をパラメータとして，アラームハンドラが呼び出される【NGKI2481】．

保護機能対応カーネルにおいて，アラームハンドラが属することのできる保護ドメインは，カーネルドメインに限られる【NGKI2482】．

マルチプロセッサ対応カーネルでグローバルタイマ方式を用いている場合には，アラームハンドラは，割付け可能プロセッサがシステム時刻管理プロセッサのみであるクラスにのみ属することができる【NGKI2483】．すなわち，アラーム ハンドラは，システム時刻管理プロセッサによって実行される．

アラームハンドラ属性に指定できる属性はない【NGKI3423】．そのためアラー ムハンドラ属性には，TA_NULLを指定しなければならない【NGKI3424】．

C言語によるアラームハンドラの記述形式は次の通り【NGKI2484】．

```
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【NGKI2485】．

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）【NGKI2486】

【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〕【NGKI2487】
acre_alm	アラームハンドラの生成〔TD〕【NGKI2488】

【静的API】

```
CRE_ALM(ID almid, { ATR almatr, intptr_t exinf, ALMHDR almhdr })
```

【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（CRE_ALMの場合）
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	アラームハンドラの先頭番地

【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（sP）【NGKI2489】 ・CPUロック状態からの呼出し（sP）【NGKI2490】
E_RSATR	予約属性 ・almatrが無効【NGKI2491】 ・属する保護ドメインの指定が有効範囲外またはカーネルドメイン以外（sP）【NGKI2492】 ・カーネルドメインの囲みの中に記述されていない（SP）【NGKI2493】 ・属するクラスの指定が有効範囲外（sM）【NGKI2494】 ・クラスの囲みの中に記述されていない（SM）【NGKI2495】 ・その他の条件については機能の項を参照
E_PAR	パラメータエラー ・almhdrがプログラムの先頭番地として正しくない【NGKI2496】
E_OACV	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（sP）【NGKI2497】
E_MACV	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（sP）【NGKI2498】
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（sD）【NGKI2499】
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（CRE_ALMの場合）【NGKI2500】

【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【NGKI2501】．

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【NGKI2502】．

マルチプロセッサ対応カーネルでグローバルタイマ方式を用いている場合で，生成するアラームハンドラの属するクラスの割付け可能プロセッサが，システム時刻管理プロセッサのみでない場合には，E_RSATRエラーとなる【NGKI2503】．

【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【ASPS0179】．ただし，動的生 成機能拡張パッケージでは，acre_almもサポートする【ASPS0180】．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，CRE_ALMのみをサポートする【FMPS0152】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，CRE_ALMのみをサポートする【HRPS0146】．ただし，動的生 成機能拡張パッケージでは，acre_almもサポートする【HRPS0205】．

【μITRON4.0仕様との関係】

almhdrのデータ型をALMHDRに変更した．

```
AID_ALM      割付け可能なアラームハンドラID の数の指定〔SD〕〔NGKI2504〕
```

【静的API】

```
    AID_ALM(uint_t noalm)
```

【パラメータ】

```
    uint_t      noalm        割付け可能なアラームハンドラID の数
```

【エラーコード】

```
    E_RSATR     予約属性
                ・保護ドメインの囲みの中に記述されている〔P〕〔NGKI3438〕
                ・クラスの囲みの中に記述されていない〔M〕〔NGKI2506〕
                ・その他の条件については機能の項を参照
    E_PAR       パラメータエラー
                ・noalm が負の値〔NGKI3286〕
```

【機能】

noalmで指定した数のアラームハンドラIDを，アラームハンドラを生成するサービスコールによって割付け可能なアラームハンドラIDとして確保する〔NGKI2507〕．

noalmは整数定数式パラメータである〔NGKI2508〕．

マルチプロセッサ対応カーネルでグローバルタイマ方式を用いている場合で，AID_ALMが属するクラスの割付け可能プロセッサが，システム時刻管理プロセッサのみでない場合には，E_RSATRエラーとなる〔NGKI2509〕．

〔TOPPERS/ASPカーネルにおける規定〕

ASPカーネルの動的生成機能拡張パッケージでは，AID_ALMをサポートする〔ASPS0218〕．

〔TOPPERS/HRP2カーネルにおける規定〕

HRP2カーネルの動的生成機能拡張パッケージでは，AID_ALMをサポートする〔HRPS0219〕．

```
SAC_ALM      アラームハンドラのアクセス許可ベクタの設定〔SP〕〔NGKI2510〕
sac_alm      アラームハンドラのアクセス許可ベクタの設定〔TPD〕〔NGKI2511〕
```

【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（S）【NGKI2512】 ・CPUロック状態からの呼出し（S）【NGKI2513】
E_ID	不正ID番号 ・almidが有効範囲外（S）【NGKI2514】
E_RSATR	予約属性 ・カーネルドメインの囲みの中に記述されていない（S）【NGKI2515】 ・対象アラームハンドラが属するクラスの囲みの中に記述されていない（SM）【NGKI2516】
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【NGKI2517】
E_OACV	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（S）【NGKI2518】
E_MACV	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（S）【NGKI2519】
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（S）【NGKI2520】 ・対象アラームハンドラに対してアクセス許可ベクタが設定済み（S）【NGKI2521】

【機能】

almid□□□

ASPカーネルでは，del_almをサポートしない〔ASPS0182〕．ただし，動的生成機能拡張パッケージでは，del_almをサポートする〔ASPS0183〕．

〔TOPPERS/FMPカーネルにおける規定〕

FMPカーネルでは，del_almをサポートしない〔FMPS0154〕．

〔TOPPERS/HRP2カーネルにおける規定〕

HRP2カーネルでは，del_almをサポートしない〔HRPS0148〕．ただし，動的生成機能拡張パッケージでは，del_almをサポートする〔HRPS0207〕．

```
sta_alm     アラームハンドラの動作開始（T）〔NGKI2533〕
ista_alm    アラームハンドラの動作開始（I）〔NGKI2534〕
```

〔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）〔NGKI2535〕
                ・タスクコンテキストからの呼出し（ista_almの場合）〔NGKI2536〕
                ・CPUロック状態からの呼出し
    E_ID        不正ID番号
                ・almidが有効範囲外〔NGKI2537〕
    E_PAR       パラメータエラー
                ・almtimがTMAX_RELTIMより大きい〔NGKI2538〕
    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（D）〔NGKI2539〕
    E_OACV      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）（P）〔NGKI2540〕
```

【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．

□□□

【機能】

prcidで指定したプロセッサを割付けプロセッサとして，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 の割付けプロセッサがprcidで指定したプロセッサに変更された後，対象アラームハンドラは動作している状態となる〔NGKI2554〕．アラームハンドラを起動する時刻は，msta_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〔NGKI2555〕．

対象アラームハンドラが動作している状態であれば，対象アラームハンドラの 割付けプロセッサがprcidで指定したプロセッサに変更された後，アラームハンドラを起動する時刻の再設定が行われる〔NGKI2556〕．

対象アラームハンドラが実行中である場合には，割付けプロセッサを変更して も，実行中のアラームハンドラを実行するプロセッサは変更されない〔NGKI2557〕．対象アラームハンドラが変更後の割付けプロセッサで実行されるのは，次に起動される時からである〔NGKI2558〕．

対象アラームハンドラの属するクラスの割付け可能プロセッサが，prcidで指定 したプロセッサを含んでいない場合には，E_PARエラーとなる〔NGKI2559〕．

prcidにTPRC_INI（=0）を指定すると，対象アラームハンドラの割付けプロセッ サを，それが属するクラスの初期割付けプロセッサとする〔NGKI2560〕．

グローバルタイマ方式を用いている場合，msta_alm/imsta_almはE_NOSPTを返す〔NGKI2561〕．

【使用上の注意】

msta_alm/imsta_almで実行中のアラームハンドラの割付けプロセッサを変更した場合，同じアラームハンドラが異なるプロセッサで同時に実行される可能性がある．特に，almtimに0を指定する場合に，注意が必要である．

〔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．

```
stp_alm     アラームハンドラの動作停止（T）〔NGKI2562〕
istp_alm    アラームハンドラの動作停止（I）〔NGKI2563〕
```

〔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）〔NGKI2564〕 ・タスクコンテキストからの呼出し（istp_almの場合）〔NGKI2565〕 ・CPUロック状態からの呼出し〔NGKI2566〕
E_ID	不正ID番号 ・almidが有効範囲外〔NGKI2567〕
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（D）〔NGKI2568〕
E_OACV	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）（P）〔NGKI2569〕

【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，動作していない状態とな る〔NGKI2570〕．対象アラームハンドラが動作していない状態であれば，何も行われずに正常終了する〔NGKI2571〕．

ref_alm	アラームハンドラの状態参照（T）〔NGKI2572〕

〔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
ID	prcid	アラームハンドラの割付けプロセッサのID（マルチプロセッサ対応カーネルの場合）

【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〔NGKI2573〕 ・CPUロック状態からの呼出し〔NGKI2574〕
E_ID	不正ID番号 ・almidが有効範囲外〔NGKI2575〕
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（D）〔NGKI2576〕
E_OACV	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（P）〔NGKI2577〕
E_MACV	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（P）〔NGKI2578〕

【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〔NGKI2579〕．

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〔NGKI2580〕．

TALM_STP	0x01U	アラームハンドラが動作していない状態
TALM_STA	0x02U	アラームハンドラが動作している状態

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〔NGKI2581〕．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〔NGKI2582〕．

マルチプロセッサ対応カーネルでは，prcidに，対象アラームハンドラの割付けプロセッサのID番号が返される〔NGKI2583〕．

【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

オーバランハンドラは，タスクが使用したプロセッサ時間が，指定した時間を超えた場合に起動されるタイムイベントハンドラである．オーバランハンドラは，システムで1つのみ登録することができる〔NGKI2584〕．

オーバランハンドラ機能に関連して，各タスクが持つ情報は次の通り〔NGKI2585〕．

- オーバランハンドラの動作状態
- 残りプロセッサ時間

オーバランハンドラの動作状態は，タスク毎に，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〔NGKI2586〕．残りプロセッサ時間は，オーバランハンドラが動作している状態の時に，タスクが使用できる残りのプロセッサ時間を表す．

□□□

□□□

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（sP）【NGKI2610】 ・CPUロック状態からの呼出し（sP）【NGKI2611】
E_RSATR	予約属性 ・ovratrが無効【NGKI2612】 ・その他の条件については機能の項を参照
E_PAR	パラメータエラー ・ovrhdrがプログラムの先頭番地として正しくない【NGKI2613】
E_OACV	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（sP） 【NGKI2614】
E_MACV	メモリアクセス違反 ・pk_dovr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（sP）【NGKI2615】
E_OBJ	オブジェクト状態エラー ・条件については機能の項を参照

【機能】

各パラメータで指定したオーバランハンドラ定義情報に従って，オーバランハンドラを定義する【NGKI2616】．ただし，def_ovrにおいてpk_dovrをNULLにした場合には，オーバランハンドラの定義を解除する【NGKI2617】．

静的APIにおいては，ovratrは整数定数式パラメータ，ovrhdrは一般定数式パラメータである【NGKI2618】．

オーバランハンドラを定義する場合（DEF_OVRの場合およびdef_ovrにおいてpk_dovrをNULL以外にした場合）で，すでにオーバランハンドラが定義されている場合には，E_OBJエラーとなる【NGKI2619】．

保護機能対応カーネルにおいて，DEF_OVRは，カーネルドメインの囲みの中に記述しなければならない．そうでない場合には，E_RSATRエラーとなる【NGKI2621】．また，def_ovrでオーバランハンドラを定義する場合には，オーバランハンドラの属する保護ドメインを設定する必要はなく，オーバランハンドラ属性にTA_DOM(domid)を指定した場合にはE_RSATRエラーとなる【NGKI2622】．ただし，TA_DOM(TDOM_SELF)を指定した場合には，指定が無視され，E_RSATRエラーは検出されない【NGKI2623】．

マルチプロセッサ対応カーネルでは，DEF_OVRは，クラスの囲みの外に記述しなければならない．そうでない場合には，E_RSATRエラーとなる【NGKI2625】．また，def_ovrオーバランハンドラを定義する場合には，オーバランハンドラの属するクラスを設定する必要はなく，オーバランハンドラ属性にTA_CLS(clsid)を指定した場合にはE_RSATRエラーとなる【NGKI2626】．ただし，TA_CLS(TCLS_SELF)を指定した場合には，指定が無視され，E_RSATRエラーは検出されない【NGKI2627】．

オーバランハンドラの定義を解除する場合（def_ovrにおいてpk_dovrをNULLにした場合）で，オーバラン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【NGKI2628】．

オーバランハンドラの定義を解除すると，オーバランハンドラの動作状態は，すべてのタスクに対して動作していない状態となる【NGKI2629】．

【使用上の注意】

def_ovrによりオーバランハンドラの定義を解除する場合，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，タスクの総数に比例して長くなる．特に，タスクの総数が多い場合，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カーネルのオーバランハンドラ機能拡張パッケージでは，DEF_OVRのみをサポートする【ASPS0186】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，DEF_OVRのみをサポートする【HRPS0150】．

【μITRON4.0仕様との関係】

ovrhdrのデータ型をOVRHDRに変更した．

def_ovrによって定義済みのオーバラン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．オーバラン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．

sta_ovr	オーバランハンドラの動作開始（T）【NGKI2630】
ista_ovr	オーバランハンドラの動作開始（I）【NGKI2631】

【C言語API】

```
ER ercd = sta_ovr(ID tskid, OVRTIM ovrtim)
ER ercd = ista_ovr(ID tskid, OVRTIM ovrtim)
```

【パラメータ】

ID	tskid	対象タスクのID番号
OVRTIM	ovrtim	対象タスクの残りプロセッサ時間

【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_ovrの場合）【NGKI2632】 ・タスクコンテキストからの呼出し（ista_ovrの場合）【NGKI2633】 ・CPUロック状態からの呼出し【NGKI2634】
E_ID	不正ID番号 ・tskidが有効範囲外【NGKI2635】
E_NOEXS	オブジェクト未登録 ・対象タスクが未登録（D）【NGKI2636】
E_OACV	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（sta_ovr□の場合）（P）【NGKI2637】
E_PAR	パラメータエラー ・ovrtimが0，またはTMAX_OVRTIMより大きい【NGKI2643】
E_OBJ	オブジェクト状態エラー ・オーバランハンドラが定義されていない【NGKI2638】

□□□□

tskid□□□□

□□□□

tskid□□□□

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〔NGKI2657〕 ・CPUロック状態からの呼出し〔NGKI2658〕
E_ID	不正ID番号 ・tskidが有効範囲外〔NGKI2659〕
E_NOEXS	オブジェクト未登録 ・対象タスクが未登録（D）〔NGKI2660〕
E_OACV	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（P）〔NGKI2661〕
E_MACV	メモリアクセス違反 ・pk_rovr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（P）〔NGKI2662〕
E_OBJ	オブジェクト状態エラー ・オーバランハンドラが定義されていない〔NGKI2663〕

【機能】

tskidで指定したタスク（対象タスク）に対するオーバランハンドラの現在状態を参照する．参照した現在状態は，pk_rovrで指定したメモリ領域に返される〔NGKI2664〕．

ovrstatには，対象タスクに対するオーバランハンドラの動作状態を表す次のいずれかの値が返される〔NGKI2665〕．

TOVR_STP	0x01U	オーバランハンドラが動作していない状態
TOVR_STA	0x02U	オーバランハンドラが動作している状態

対象タスクに対してオーバランハンドラが動作している状態の場合には，leftotmに，オーバランハンドラが起動されるまでの残りプロセッサ時間が返される〔NGKI2666〕．オーバランハンドラが起動される直前には，leftotmに0が返される可能性がある〔NGKI2667〕．オーバランハンドラが動作していない状態の場合には，leftotmの値は保証されない〔NGKI2668〕．

tskidにTSK_SELF（=0）を指定すると，自タスクが対象タスクとなる〔NGKI2669〕．

【使用上の注意】

ref_ovr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_ovrを呼び出し，対象オーバラン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_ovrから戻ってきた時には対象オーバラ ン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．

【未決定事項】

マルチプロセッサ対応カーネルにおいて，対象タスクが，自タスクが割付けられたプロセッサと異なるプロセッサに割り付けられている場合に，leftotmを参照できるとするかどうかは，今後の課題である．

〔μITRON4.0仕様との関係〕

TOVR_STPとTOVR_STAを値を変更した．

4.7. システム状態管理機能

システム状態管理機能は，特定のオブジェクトに関連しないシステムの状態を 変更／参照するための機能である．

```
SAC_SYS      システム状態のアクセス許可ベクタの設定（SP）【NGKI2670】
sac_sys      システム状態のアクセス許可ベクタの設定（TPD）【NGKI2671】
```

【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（S）【NGKI2672】
                ・CPUロック状態からの呼出し（S）【NGKI2673】
    E_RSATR     予約属性
                ・カーネルドメインの囲みの中に記述されていない（S）【NGKI2674】
                ・クラスの囲みの中に記述されている（SM）【NGKI2675】
    E_OACV      オブジェクトアクセス違反
                ・カーネルドメイン以外からの呼出し（S）【NGKI2676】
    E_OBJ       オブジェクト状態エラー
                ・システム状態のアクセス許可ベクタが設定済み（S）【NGKI2677】
```

【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【NGKI2678】．

静的APIにおいては，acptn1~acptn4は整数定数式パラメータである【NGKI2679】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，SAC_SYSのみをサポートする【HRPS0151】．

```
rot_rdq      タスクの優先順位の回転（T）【NGKI2680】
irot_rdq     タスクの優先順位の回転（I）【NGKI2681】
```

【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）【NGKI2682】
                ・タスクコンテキストからの呼出し（irot_rdqの場合）【NGKI2683】
                ・CPUロック状態からの呼出し【NGKI2684】
    E_NOSPT     未サポート機能
                ・条件については機能の項を参照
    E_PAR       パラメータエラー
                ・tskpriが有効範囲外【NGKI2685】
    E_OACV      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（rot_rdq
                □の場合）（P）【NGKI2686】
```

【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【NGKI2687】．対象優先度を持つ実行できる状態のタスクが無いか 1つのみの場合には，何も行われずに正常終了する【NGKI2688】．

マルチプロセッサ対応カーネルにおいては，自タスクと同じプロセッサに割り 付けられているタスクのみを操作対象とする【NGKI3622】．

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【NGKI2689】．

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 が制約タスクの場合には，E_NOSPTエラーとなる【NGKI2690】．

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは，rot_rdq，irot_rdqをサポートしない〔SSPS0131〕．

```
mrot_rdq     プロセッサ指定でのタスクの優先順位の回転（TM）〔NGKI2691〕
imrot_rdq    プロセッサ指定でのタスクの優先順位の回転（IM）〔NGKI2692〕
```

〔C言語API〕

```
    ER ercd = mrot_rdq(PRI tskpri, ID prcid)
    ER ercd = imrot_rdq(PRI tskpri, ID prcid)
```

【パラメータ】

```
    PRI         tskpri      回転対象の優先度（対象優先度）
    ID          prcid       優先順位の回転対象とするプロセッサの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（mrot_rdqの場合）
                □〔NGKI2693〕
                ・タスクコンテキストからの呼出し（imrot_rdqの場合）〔NGKI2694〕
                ・CPUロック状態からの呼出し〔NGKI2695〕
    E_NOSPT     未サポート機能
                ・条件については機能の項を参照
    E_ID        不正ID番号
                ・prcidが有効範囲外〔NGKI2696〕
    E_PAR       パラメータエラー
                ・tskpriが有効範囲外〔NGKI2697〕
    E_OACV      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（mrot_rdq
                □の場合）（P）〔NGKI2698〕
```

【機能】

prcidで指定したプロセッサに割り付けられており，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〔NGKI2699〕．対象優先度を持つ実行できる状態のタスクが無いか1つのみの場合には，何も行わずに正常終了する〔NGKI2700〕．

mrot_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〔NGKI2701〕．

prcidで指定したプロセッサに割り付けられており，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が制約タスクの場合には，

E_NOSPTエラーとなる〔NGKI2702〕.

〔TOPPERS/ASPカーネルにおける規定〕

ASPカーネルでは，mrot_rdq，imrot_rdqをサポートしない〔ASPS0188〕.

〔TOPPERS/HRP2カーネルにおける規定〕

HRP2カーネルでは，mrot_rdq，imrot_rdqをサポートしない〔HRPS0152〕.

〔TOPPERS/SSPカーネルにおける規定〕

SSPカーネルでは，mrot_rdq，imrot_rdqをサポートしない〔SSPS0132〕.

〔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.

```
get_tid      実行状態のタスクIDの参照（T）〔NGKI2703〕
iget_tid     実行状態のタスクIDの参照（I）〔NGKI2704〕
```

〔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）〔NGKI2705〕
                ・タスクコンテキストからの呼出し（iget_tidの場合）〔NGKI2706〕
                ・CPUロック状態からの呼出し〔NGKI2707〕
    E_MACV      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（get_tidの場合）（P）〔NGKI2708〕
```

【機能】

実行状態のタスク（get_tidの場合には自タスク）のID番号を参照する．参照したタスクIDは，p_tskidが指すメモリ領域に返される〔NGKI2709〕．

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〔NGKI2710〕．

マルチプロセッサ対応カーネルにおいては，サービスコールを呼び出した処理単位を実行しているプロセッサにおいて実行状態のタスクのID番号を参照する〔NGKI2711〕．

〔TOPPERS/SSPカーネルにおける規定〕

SSPカーネルでは，get_tidをサポートしない〔SSPS0133〕．

```
get_did      実行状態のタスクが属する保護ドメインIDの参照（TP）〔NGKI2712〕
```

〔C言語API〕

```
    ER ercd = get_did(ID *p_domid)
```

【パラメータ】

```
    ID *        p_domid      保護ドメイン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          domid        保護ドメイン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〔NGKI2713〕
                ・CPUロック状態からの呼出し〔NGKI2714〕
    E_MACV      メモリアクセス違反
                ・p_domid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〔NGKI2715〕
```

【機能】

実行状態のタスク（自タスク）が属する保護ドメインのID番号を参照する．参照した保護ドメインIDは，p_domidが指すメモリ領域に返される〔NGKI2716〕．

マルチプロセッサ対応カーネルにおいては，サービスコールを呼び出した処理単位を実行しているプロセッサにおいて実行状態のタスクが属する保護ドメインのID番号を参照する〔NGKI2717〕．

〔TOPPERS/ASPカーネルにおける規定〕

ASPカーネルでは，get_didをサポートしない〔ASPS0189〕．

〔TOPPERS/FMPカーネルにおける規定〕

FMPカーネルでは，get_didをサポートしない〔FMPS0157〕．

〔TOPPERS/SSPカーネルにおける規定〕

SSPカーネルでは，get_didをサポートしない〔SSPS0134〕．

```
get_pid      割付けプロセッサのID番号の参照（TM）〔NGKI2718〕
iget_pid     割付けプロセッサのID番号の参照（IM）〔NGKI2719〕
```

〔C言語API〕

```
    ER ercd = get_pid(ID *p_prcid)
    ER ercd = iget_pid(ID *p_prcid)
```

【パラメータ】

```
    ID *        p_prcid      プロセッサ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          prcid        プロセッサ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_pidの場合）〔NGKI2720〕
                ・タスクコンテキストからの呼出し（iget_pidの場合）〔NGKI2721〕
                ・CPUロック状態からの呼出し〔NGKI2722〕
    E_MACV      メモリアクセス違反
                ・p_prcid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（get_pidの場合）（P）〔NGKI2723〕
```

【機能】

サービスコールを呼び出した処理単位の割付けプロセッサのID番号を参照する．参照したプロセッサIDは，p_prcidが指すメモリ領域に返される〔NGKI2724〕．

【使用上の注意】

タスクは，get_pidを用いて，自タスクの割付けプロセッサを正しく参照できる　　とは限らない．これは，get_pidを呼び出し，自タスクの割付けプロセッサのID番号を参照した直後に割込みが発生した場合，get_pidから戻ってきた時には 割付けプロセッサが変化している可能性があるためである．

〔TOPPERS/ASPカーネルにおける規定〕

ASPカーネルでは，get_pid，iget_pidをサポートしない〔ASPS0190〕．

〔TOPPERS/HRP2カーネルにおける規定〕

HRP2カーネルでは，get_pid，iget_pidをサポートしない〔HRPS0153〕．

〔TOPPERS/SSPカーネルにおける規定〕

SSPカーネルでは，get_pid，iget_pidをサポートしない〔SSPS0135〕．

〔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．

```
loc_cpu      CPUロック状態への遷移〔T〕〔NGKI2725〕
iloc_cpu     CPUロック状態への遷移〔I〕〔NGKI2726〕
```

〔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）〔NGKI2727〕
                ・タスクコンテキストからの呼出し（iloc_cpuの場合）〔NGKI2728〕
    E_OACV      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
                □（loc_cpuの場合）〔P〕〔NGKI2729〕
```

【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〔NGKI2730〕．CPUロック状態で呼び出した場合には，何も行われずに正常終了する〔NGKI2731〕．

```
unl_cpu      CPUロック状態の解除〔T〕〔NGKI2732〕
iunl_cpu     CPUロック状態の解除〔I〕〔NGKI2733〕
```

【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（unl_cpuの場合）〔NGKI2734〕 ・タスクコンテキストからの呼出し（iunl_cpuの場合）〔NGKI2735〕
E_OACV	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 （unl_cpuの場合）（P）〔NGKI2736〕

【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〔NGKI2737〕．CPUロック解除状態で呼び出した場合には，何も行われずに正常終了する〔NGKI2738〕．

マルチプロセッサ対応カーネルにおいて，unl_cpu/iunl_cpuを呼び出したプロセッサによって取得されている状態となっているスピンロックがある場合には，unl_cpu/iunl_cpuによってCPUロック解除状態に遷移しない（何も行われずに正常終了する）〔NGKI2739〕．

【補足説明】

マルチプロセッサ対応カーネルでは，CPUロック解除状態へ遷移した結果，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．また，保護機能対応カーネルとマルチプロセッサ対応カーネルでは，タスク例外処理ルーチンの実行が開始される可能性がある．

dis_dsp	ディスパッチの禁止（T）〔NGKI2740〕	

【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［NGKI2741］ ・CPUロック状態からの呼出し［NGKI2742］
E_OACV	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（P） ［NGKI2743］

【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［NGKI2744］．ディスパッチ禁止状態で呼び出した場合には，何も行われずに 正常終了する［NGKI2745］．

ena_dsp	ディスパッチの許可（T）［NGKI2746］

［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［NGKI2747］ ・CPUロック状態からの呼出し［NGKI2748］
E_OACV	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（P） ［NGKI2749］

【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［NGKI2750］．ディスパッチ許可状態で呼び出した場合には，何も行われずに 正常終了する［NGKI2751］．

【補足説明】

ディスパッチ許可状態へ遷移した結果，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．

```
sns_ctx      コンテキストの参照〔TI〕〔NGKI2752〕
```

〔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〔NGKI2753〕．

```
sns_loc     CPUロック状態の参照〔TI〕〔NGKI2754〕
```

〔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〔NGKI2755〕．

```
sns_dsp       ディスパッチ禁止状態の参照〔TI〕〔NGKI2756〕
```

〔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〔NGKI2757〕．

```
sns_dpn       ディスパッチ保留状態の参照〔TI〕〔NGKI2758〕
```

〔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〔NGKI2759〕．

```
sns_ker        カーネル非動作状態の参照〔TI〕〔NGKI2760〕
```

〔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〔NGKI2761〕．

【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〔TI〕〔NGKI2762〕
```

〔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_SYS       システムエラー
                ・カーネルの誤動作〔NGKI2763〕
    E_OACV      オブジェクトアクセス違反
                ・カーネルドメイン以外からの呼出し〔P〕〔NGKI2764〕
```

【機能】

カーネルを終了する．具体的な振舞いについては，「2.9.2 システム終了手順」の節を参照すること．

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〔NGKI2765〕．

〔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をサポートしない．

〔TOPPERS/FMPカーネルにおける規定〕

FMPカーネルでは，ref_sysをサポートしない．

〔TOPPERS/HRP2カーネルにおける規定〕

HRP2カーネルでは，ref_sysをサポートしない．

〔TOPPERS/SSPカーネルにおける規定〕

SSPカーネルでは，ref_sysをサポートしない．

4.8. メモリオブジェクト管理機能

メモリオブジェクト管理機能は，保護機能対応カーネルでのみサポートされる 機能である．保護機能対応でないカーネルでは，メモリオブジェクト管理機能 をサポートしない．

〔メモリリージョン〕

メモリリージョンは，オブジェクトモジュールに含まれるセクションの配置対象となる同じ性質を持った連続したメモリ領域である．メモリリージョンは，メモリリージョン名によって識別する【NGKI2766】．

各メモリリージョンが持つ情報は次の通り【NGKI2767】．

- 先頭番地
- サイズ
- メモリリージョン属性

メモリリージョンの先頭番地とサイズには，ターゲット定義の制約が課せられる場合がある【NGKI2768】．

メモリリージョン属性には，次の属性を指定することができる【NGKI3256】．

```
TA_NOWRITE   0x01U    書込みアクセス禁止
```

ターゲットによっては，ターゲット定義のメモリリージョン属性を指定できる 場合がある【NGKI2771】．

標準メモリリージョンとは，ATT_MOD／ATA_MODによって，オブジェクトモジュールに含まれる標準のセクションが配置されるメモリリージョンである．標準メモリリージョンには，標準のセクションの中で，書込みアクセスを行わないものが配置される標準ROMリージョンと，書込みアクセスを行うものが配置される 標準RAMリージョンが含まれる．

マルチプロセッサ対応カーネルでは，ATT_MOD／ATA_MODがクラスの囲みの外に記述された場合に適用される共通の標準メモリリージョンに加えて，クラス毎の標準メモリリージョンを定義することができる【NGKI3257】．

標準メモリリージョン（マルチプロセッサ対応カーネルでは，共通の標準メモリリージョン）は，必ず定義しなければならない．定義しない場合には，コンフィギュレータがエラーを報告する【NGKI3259】．

〔メモリオブジェクト〕

メモリオブジェクトは，保護機能対応カーネルにおいてアクセス保護の対象とする連続したメモリ領域である．メモリオブジェクトは，その先頭番地によって識別する【NGKI2772】．

各メモリオブジェクトが持つ情報は次の通り【NGKI2773】．

- 先頭番地
- サイズ
- メモリオブジェクト属性
- アクセス許可ベクタ
- 属する保護ドメイン
- 属するクラス（マルチプロセッサ対応カーネルの場合）

メモリオブジェクトの先頭番地とサイズには，ターゲット定義の制約が課せられる【NGKI2774】．

メモリオブジェクト属性には，次の属性を指定することができる【NGKI2775】．

```
TA_NOWRITE   0x01U   書込みアクセス禁止
TA_NOREAD    0x02U   読出しアクセス禁止
TA_EXEC      0x04U   実行アクセス許可
TA_MEMINI    0x08U   メモリの初期化を行う
TA_MEMPRSV   0x10U   メモリの初期化を行わない
TA_SDATA     0x20U   ショートデータ領域に配置
TA_UNCACHE   0x40U   キャッシュ禁止
TA_IODEV     0x80U   周辺デバイスの領域
```

メモリオブジェクトに対して書込みアクセスできるのは，メモリオブジェクト属性に書込みアクセス禁止（TA_NOWRITE属性）が指定されておらず，アクセス許可ベクタにより書込みアクセスが許可されている場合である【NGKI2776】．また，読出しアクセスできるのは，メモリオブジェクト属性に読出しアクセス禁止（TA_NOREAD属性）が指定されておらず，アクセス許可ベクタにより読出し・実行アクセスが許可されている場合である【NGKI2777】．実行アクセスできるのは，メモリオブジェクト属性に実行アクセス許可（TA_EXEC属性）が指定されており，アクセス許可ベクタにより読出し・実行アクセスが許可されている場合である【NGKI2778】．

ただし，ターゲットハードウェアの制約によってこれらの属性を実現できない場合には，次のように扱われる．書込みアクセス禁止が実現できない場合には，TA_NOWRITEを指定しても無視される【NGKI2779】．また，読出しアクセス禁止が実現できない場合には，TA_NOREADを指定しても無視される【NGKI2780】．実行アクセス禁止が実現できない場合には，TA_EXECを指定しなくても実行アクセス許可となり，TA_EXECは無視される【NGKI2781】．どのような場合にどの属性の指定が無視されるかは，ターゲット定義である【NGKI2782】．

TA_MEMINI属性は，システム初期化時に初期化するメモリオブジェクトであることを，TA_MEMPRSV属性は，システム初期化時に初期化を行わないメモリオブジェクトであることを示す【NGKI2783】．いずれの属性も指定しない場合，そのメモリオブジェクトは，システム初期化時にクリア（言い換えると，0に初期化）される【NGKI2784】．

TA_MEMINI属性を設定したメモリオブジェクトを初期化に用いる初期化データは，標準ROMリージョン（マルチプロセッサ対応カーネルでは，共通の標準ROMリージョン）に配置され，メモリオブジェクトとしては登録されない【NGKI2787】．

TA_SDATA属性は，メモリオブジェクトをショートデータ領域に配置することを示す【NGKI2788】．具体的な扱いはターゲット定義であるが，ショートデータ領域がサポートされていないターゲットでは，この属性は無視される【NGKI2789】．また，ターゲットによっては，TA_NOWRITEを指定した場合に，TA_SDATAが無視される場合がある【NGKI2790】．

TA_UNCACHE属性は，メモリオブジェクトをキャッシュ禁止に設定することを，TA_IODEV属性は，メモリオブジェクトを周辺デバイスの領域として扱うことを示す【NGKI2791】．具体的な扱いはターゲット定義であるが，これらの属性を指定しても意味がないターゲット（例えば，キャッシュを持たないターゲットプロセッサでのTA_UNCACHE）では，これらの属性は無視される【NGKI2792】．逆に，キャッシュ禁止にできないメモリオブジェクトに対してTA_UNCACHEを指定した場合や，周辺デバイスの領域として扱うことができないメモリオブジェクトに対してTA_IODEVを指定した場合には，E_RSATRエラーとなる【NGKI2793】．

ターゲットによっては，ターゲット定義のメモリオブジェクト属性を指定できる場合がある【NGKI2794】．ターゲット定義のメモリオブジェクト属性として，次の属性を予約している【NGKI2795】．

```
TA_WTHROUGH   ライトスルーキャッシュを用いる
```

〔カーネル構成マクロ〕

メモリオブジェクト管理機能に関連するカーネル構成マクロは次の通り．

```
TOPPERS_SUPPORT_ATT_MOD      ATT_MOD、ATA_MODが使用可能
                             （NGKI2796）
TOPPERS_SUPPORT_ATT_PMA      ATT_PMA、ATA_PMA、att_pmaが使用可
                             能（NGKI2797）
```

なお、att_pmaは、静的APIのみで
動的生成オブジェクトを対象とするTOPPERS_SUPPORT_DYNAMIC_CRE、
TOPPERS_SUPPORT_ATT_PMAを定義しない場合は、使用不可 （NGKI2798）。

（例）

静的生成オブジェクトを対象とするATT_MOD、ATA_MODは使用可能 （動的生成オブジェクトを対象とするものは使用不可）。

（TOPPERS/ASPの場合）

ASPでは、静的生成オブジェクトを対象とするATT_MODはASPS0191。

（TOPPERS/FMPの場合）

FMPでは、静的生成オブジェクトを対象とするATT_MODはFMPS0158。

（TOPPERS/HRP2の場合）

HRP2では、静的生成オブジェクトを対象とするATT_MODはHRPS0154。

（TOPPERS/SSPの場合）

SSPでは、静的生成オブジェクトを対象とするATT_MODはSSPS0136。

（μITRON4.0/PXの場合）

（0）属性値として、TA_RW、TA_CACHEを指定可能。
属性値として、TA_RO、TA_NOWRITE、TA_NOREAD、TA_EXEC、TA_MEMINI、
TA_MEMPRSV、TA_IODEVを指定可能。TA_UNCACHEを指定可能。
属性値として、TA_WTHROUGHを指定。

（μITRON4.0/PXの場合）

（例）

TA_IODEVを指定した場合は、ATT_MOD、ATA_MODを使用可能。
ATT_MOD、ATA_MODを使用した場合は、TA_IODEVを指定可能。
ATT_MOD、ATA_MODを使用した場合は、TA_IODEVを指定可能。

```
ATT_REG      静的生成オブジェクトのSP（NGKI2799）
```

（）API

```
ATT_REG("メモリリージョン名", { ATR regatr, void *base, SIZE size })
```

【パラメータ】

```
"メモリリージョン名"  登録するメモリリージョンを指定する文字列
ATR          regatr       メモリリージョン属性
void *       base         登録するメモリリージョンの先頭番地
SIZE         size         登録するメモリリージョンのサイズ（バイト数）
```

【エラーコード】

```
E_RSATR      予約属性
             ・regatrが無効〔NGKI2800〕
             ・保護ドメインの囲みの中に記述されている〔NGKI2814〕
             ・クラスの囲みの中に記述されている（M）〔NGKI3260〕
E_PAR        パラメータエラー
             ・sizeが0以下〔NGKI2816〕
             ・その他の条件については機能の項を参照
E_OBJ        オブジェクト状態エラー
             ・登録済みのメモリリージョンの再登録〔NGKI2801〕
             ・その他の条件については機能の項を参照
```

【機能】

各パラメータで指定したメモリリージョン登録情報に従って，指定したメモリ リージョンを登録する．具体的な振舞いは以下の通り．

baseとsizeで指定したメモリ領域が，メモリリージョンとして登録される〔NGKI2802〕．登録されるメモリリージョンには，regatrで指定したメモリリージョン属性が設定される〔NGKI2803〕．

メモリリージョン名は文字列パラメータ，regatr，base，sizeは整数定数式パ ラメータである〔NGKI2804〕．

baseやsizeに，ターゲット定義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〔NGKI2815〕．登録しようとしたメモリリージョンが，登録済みのメモリリージョンとメモリ領域が重なる場合には，E_OBJエラー となる〔NGKI2817〕．

〔μITRON4.0/PX仕様との関係〕

μITRON4.0/PX仕様に定義されていない静的APIである．

```
DEF_SRG      標準メモリリージョンの定義（SP）〔NGKI3261〕
```

【静的API】

```
DEF_SRG("標準ROMリージョン名", "標準RAMリージョン名")
```

【パラメータ】

"標準ROMリージョン名"	標準ROMリージョンとするメモリリージョンを指定する文字列
"標準RAMリージョン名"	標準RAMリージョンとするメモリリージョンを指定する文字列

【エラーコード】

E_RSATR	予約属性 ・保護ドメインの囲みの中に記述されている〔NGKI3262〕
E_OBJ	オブジェクト状態エラー ・標準メモリリージョンが定義済み〔NGKI3263〕 ・標準ROMリージョンに指定したメモリリージョンが未登録 〔NGKI3264〕 ・標準RAMリージョンに指定したメモリリージョンが未登録 〔NGKI3272〕 ・その他の条件については機能の項を参照

【機能】

各パラメータに従って，標準ROMリージョンと標準RAMリージョンを定義する〔NGKI3265〕．

マルチプロセッサ対応カーネルでは，DEF_SRGをクラスの囲みの外に記述すると，共通の標準ROMリージョンと標準RAMリージョンを定義し，クラスの囲みの中に記述すると，そのクラスの標準ROMリージョンと標準RAMリージョンを定義する〔NGKI3266〕．

標準ROMリージョンは，TA_NOWRITE属性のメモリリージョンでなければならない．標準ROMリージョンとして指定したメモリリージョンが，TA_NOWRITE属性でない場合には，E_OBJエラーとなる〔NGKI3268〕．また，標準RAMリージョンは，TA_NOWRITE属性でないメモリリージョンでなければならない．標準RAMリージョンとして指定したメモリリージョンが，TA_NOWRITE属性である場合には，E_OBJエラーとなる〔NGKI3270〕．

〔μITRON4.0/PX仕様との関係〕

μITRON4.0/PX仕様に定義されていない静的APIである．

ATT_SEC	セクションの登録（SP）〔NGKI2818〕
ATA_SEC	セクションの登録（アクセス許可ベクタ付き）（SP）〔NGKI2819〕

【静的API】

```
ATT_SEC("セクション名", { ATR mematr, "メモリリージョン名" })
ATA_SEC("セクション名", { ATR mematr, "メモリリージョン名" },
            { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
```

【パラメータ】

```
"セクション名"                    登録するセクションを指定する文字列
ATR          mematr               メモリオブジェクト属性
"メモリリージョン名"  セクションを配置するメモリリージョンを指定
                                  する文字列

□*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
```

【エラーコード】

```
E_RSATR      予約属性
             ・mematrが無効〔NGKI2820〕
             ・その他の条件については機能の項を参照
E_NOSPT      未サポート機能
             ・条件については機能の項を参照
E_PAR        パラメータエラー
             ・条件については機能の項を参照
E_OBJ        オブジェクト状態エラー
             ・登録済みのセクションの再登録〔NGKI2821〕
             ・指定したメモリリージョンが未登録〔NGKI2822〕
```

【機能】

各パラメータで指定した情報に従って，指定したセクションをカーネルに登録 する．具体的な振舞いは以下の通り．

各オブジェクトモジュールに含まれるセクション名で指定したセクションが，メモリリージョン名で指定したメモリリージョンに配置され，メモリオブジェクトとして登録される〔NGKI2823〕．登録されるメモリオブジェクトには，mematrで指定したメモリオブジェクト属性が設定される〔NGKI2824〕．ATA_SECの場合には，登録されるメモリオブジェクト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）が，acptn1~acptn4で指定した値に設定される〔NGKI2825〕．

指定したメモリリージョンがTA_NOWRITE属性である場合には，メモリオブジェクト属性にTA_NOWRITE属性を指定したことになる（TA_NOWRITE属性を指定しても指定しなくても，同じ振舞いとなる）〔NGKI2826〕．また，メモリオブジェクト属性のTA_MEMINIとTA_MEMPRSVは無視される（指定しても指定しなくても，同じ振舞いとなる）〔NGKI2786〕．

mematrに，TA_MEMINIとTA_MEMPRSVを同時に指定することはできない．指定した場合には，E_RSATRエラーとなる〔NGKI2828〕．

登録されるメモリオブジェクトと同じ保護ドメインに属し，メモリオブジェクト属性とアクセス許可ベクタがすべて一致するメモリオブジェクトがある場合には，1つのメモリオブジェクトにまとめて登録される場合がある〔NGKI2829〕．

セクション名とメモリリージョン名は文字列パラメータ，mematr，acptn1~ acptn4は整数定数式パラメータである〔NGKI2830〕．

ターゲット定義で，ATA_SECにより登録できるセクションが属する保護ドメインや登録できる数に制限がある場合がある〔NGKI2831〕．この制限に違反した場合には，E_NOSPTエラーとなる〔NGKI2832〕．

ATT_MOD／ATA_MODがサポートされているターゲットでは，セクション名として，標準のセクションを指定することはできない．指定した場合には，E_PARエラーとなる〔NGKI2834〕．

□□□

μITRON4.0/PX仕様に定義されていない静的APIである．

ATT_MOD	オブジェクトモジュールの登録（SP）【NGKI2845】
ATA_MOD	オブジェクトモジュールの登録（アクセス許可ベクタ付き）（SP）【NGKI2846】

【静的API】

```
ATT_MOD("オブジェクトモジュール名")
ATA_MOD("オブジェクトモジュール名",
            { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
```

【パラメータ】

"オブジェクトモジュール名"		登録するオブジェクトモジュールを指定する文字列
*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

【エラーコード】

E_RSATR	予約属性 ・mematrが無効【NGKI2847】
E_NOSPT	未サポート機能 ・条件については機能の項を参照
E_OBJ	オブジェクト状態エラー ・登録済みのオブジェクトモジュールの再登録【NGKI2848】

【機能】

各パラメータで指定した情報に従って，指定したオブジェクトモジュールをカーネルに登録する．具体的な振舞いは以下の通り．

オブジェクトモジュール名で指定したオブジェクトモジュールに含まれる標準のセクションの内，書込みアクセスを行わないセクションは標準ROMリージョンに，書込みアクセスを行うセクションは標準RAMリージョンに配置され，メモリオブジェクトとして登録される【NGKI2849】．登録されるメモリオブジェクトには，ターゲット定義でセクション毎に定まるメモリオブジェクト属性が設定される【NGKI2850】．ATA_MODの場合には，登録されるメモリオブジェクト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）が，acptn1~acptn4で指定した値に設定される【NGKI2851】．

マルチプロセッサ対応カーネルでは，ATT_MOD／ATA_MODを，クラスの囲みの外に記述することも，クラスの囲みの中に記述することもできる【NGKI2852】．ATT_MOD／ATA_MODをクラスの囲みの外に記述した場合，標準のセクションは，共通の標準メモリリージョンに配置される【NGKI2853】．クラスの囲みの中に記述した場合，そのクラスの標準メモリリージョンが定義されていればそれらのメモリリージョン，定義されていなければ共通の標準メモリリージョンに配置される【NGKI2854】．ただし，セクションによっては，ターゲット定義で，クラスの標準メモリリージョンが定義されている場合でも，共通の標準メモリリージョンに配置される場合がある【NGKI3271】．

□□□

T_AMEM *	pk_amem	メモリオブジェクトの登録情報を入れたパケットへのポインタ（静的APIを除く）

□*メモリオブジェクトの登録情報（パケットの内容）

ATR	mematr	メモリオブジェクト属性
void *	base	登録するメモリ領域の先頭番地
SIZE	size	登録するメモリ領域のサイズ（バイト数）

□*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（s）【NGKI2865】 ・CPUロック状態からの呼出し（s）【NGKI2866】
E_RSATR	予約属性 ・mematrが無効【NGKI2867】 ・属する保護ドメインの指定が有効範囲外（sP）【NGKI2868】 ・属するクラスの指定が有効範囲外（sM）【NGKI2869】 ・その他の条件については機能の項を参照
E_NOSPT	未サポート機能 ・条件については機能の項を参照
E_PAR	パラメータエラー ・sizeが0以下【NGKI2881】 ・その他の条件については機能の項を参照
E_OACV	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（sP）□【NGKI2870】
E_MACV	メモリアクセス違反 ・pk_amem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（sP）【NGKI2871】
E_OBJ	オブジェクト状態エラー ・条件については機能の項を参照

【機能】

各パラメータで指定したメモリオブジェクト登録情報に従って，メモリオブジェ クトを登録する．具体的な振舞いは以下の通り．

baseとsizeで指定したメモリ領域が，メモリオブジェクトとして登録される【NGKI2872】．登録されるメモリオブジェクトには，mematrで指定したメモリ

オブジェクト属性が設定される【NGKI2873】．ATA_MEMの場合には，登録されるメモリオブジェクト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）が，acptn1~acptn4で指定した値に設定される【NGKI2874】．

mematrには，TA_MEMPRSVを指定しなければならず，TA_MEMINIを指定することはできない．TA_MEMPRSVを指定しない場合や，TA_MEMINIを指定した場合には，E_RSATRエラーとなる【NGKI2876】．また，mematrにTA_SDATAを指定することはできない．TA_SDATAを指定した場合には，E_RSATRエラーとなる【NGKI3274】．

静的APIにおいては，matr，size，acptn1~acptn4は整数定数式パラメータ，baseは一般定数式パラメータである【NGKI2877】．

ターゲット定義で，ATT_MEM/ATA_MEMにより登録できるメモリオブジェクトが属する保護ドメインや登録できる数に制限がある場合がある【NGKI2878】．この制限に違反した場合には，E_NOSPTエラーとなる【NGKI2879】．

baseやsizeに，ターゲット定義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【NGKI2880】．登録しようとしたメモリオブジェクトが，登録済みのメモリオブジェクトとメモリ領域が重なる場合には，E_OBJエラーとなる【NGKI2882】．

【使用上の注意】

ATT_MEM/ATA_MEMは，メモリ空間にマッピングされたI/O領域にアクセスできるようにするために使用することを想定した静的APIである．メモリ領域に対しては，ATT_SEC/ATA_SECかATT_MOD/ATA_MODを使用することを推奨する．

ATT_MEM/ATA_MEMで登録したメモリオブジェクトのメモリ領域が，ATT_REGで登録したメモリリージョンと重なっても，直ちにエラーとはならない．ただし，メモリリージョン内に配置されたメモリオブジェクトと，ATT_MEM/ATA_MEMで登録したメモリオブジェクトのメモリ領域が重なった場合には，E_OBJエラーとなる．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，ATT_MEMとATA_MEMのみをサポートする【HRPS0155】．

【μITRON4.0/PX仕様との関係】

アクセス許可ベクタを指定してメモリオブジェクトを登録するサービスコール（ata_mem）は廃止した．

baseやsizeがターゲット定義の制約に合致しない場合，μITRON4.0/PX仕様ではターゲット定義の制約に合致するようにメモリ領域を広げることとしていたが，この仕様ではE_PARエラーとなることとした．

ATT_PMA	物理メモリ領域の登録	〔SP〕【NGKI2883】
ATA_PMA	物理メモリ領域の登録（アクセス許可ベクタ付き）	〔SP〕【NGKI2884】
att_pma	物理メモリ領域の登録	〔TPD〕【NGKI2885】

【静的API】

```
ATT_PMA({ ATR mematr, void *base, SIZE size, void *paddr })
ATA_PMA({ ATR mematr, void *base, SIZE size, void *paddr  },
            { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
```

【C言語API】

```
ER ercd = att_pma(const T_APMA *pk_apma)
```

【パラメータ】

T_APMA *	pk_apma	物理メモリ領域の登録情報を入れたパケットへのポインタ（静的APIを除く）

□*物理メモリ領域の登録情報（パケットの内容）

ATR	mematr	メモリオブジェクト属性
void *	base	登録するメモリ領域の先頭番地
SIZE	size	登録するメモリ領域のサイズ（バイト数）
void *	paddr	登録するメモリ領域の物理アドレス空間における先頭番地

□*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（sS）〔NGKI2886〕 ・CPUロック状態からの呼出し（sS）〔NGKI2887〕
E_RSATR	予約属性 ・mematrが無効 ・属する保護ドメインの指定が有効範囲外（sP）〔NGKI2888〕 ・属するクラスの指定が有効範囲外（sM）〔NGKI2889〕 ・その他の条件については機能の項を参照
E_NOSPT	未サポート機能 ・条件については機能の項を参照
E_PAR	パラメータエラー ・sizeが0以下〔NGKI2901〕 ・その他の条件については機能の項を参照
E_OACV	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（sP）□〔NGKI2890〕
E_MACV	メモリアクセス違反 ・pk_apma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（sP）〔NGKI2891〕
E_OBJ	オブジェクト状態エラー ・条件については機能の項を参照

【機能】

□□□

void *	base	メモリオブジェクトの先頭番地
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

□*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〔NGKI2908〕 ・CPUロック状態からの呼出し〔NGKI2909〕
E_PAR	パラメータエラー ・baseがメモリオブジェクトの先頭番地でない〔NGKI2910〕
E_NOEXS	オブジェクト未登録 ・baseで指定した番地を含むメモリオブジェクトが登録されていない〔NGKI2911〕
E_OACV	オブジェクトアクセス違反 ・対象メモリオブジェクトに対する管理操作が許可されていない〔NGKI2912〕
E_MACV	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〔NGKI2913〕
E_OBJ	オブジェクト状態エラー ・対象メモリオブジェクトは静的APIで登録された〔NGKI2914〕

【機能】

baseで指定したメモリオブジェクト（対象メモリオブジェクト）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〔NGKI2915〕.

〔TOPPERS/HRP2カーネルにおける規定〕

HRP2カーネルでは，sac_memをサポートしない〔HRPS0157〕.

〔μITRON4.0/PX仕様との関係〕

静的APIによって登録したメモリオブジェクトは，アクセス許可ベクタを設定す ることができないこととした.

μITRON4.0/PX仕様では，baseはメモリオブジェクトに含まれる番地を指定する ものとしていたが，この仕様では，メモリオブジェクトの先頭番地でなければ ならないものとした.

det_mem	メモリオブジェクトの登録解除（TPD）【NGKI2916】

【C言語API】

```
ER ercd = det_mem(const void *base)
```

【パラメータ】

void *	base	メモリオブジェクトの先頭番地

【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【NGKI2917】 ・CPUロック状態からの呼出し【NGKI2918】
E_PAR	パラメータエラー ・baseがメモリオブジェクトの先頭番地でない【NGKI2919】
E_NOEXS	オブジェクト未登録 ・baseで指定した番地を含むメモリオブジェクトが登録され □ていない【NGKI2920】
E_OACV	オブジェクトアクセス違反 ・対象メモリオブジェクトに対する管理操作が許可されてい □ない【NGKI2921】
E_OBJ	オブジェクト状態エラー ・対象メモリオブジェクトは静的APIで登録された【NGKI2922】

【機能】

baseで指定したメモリオブジェクト（対象メモリオブジェクト）を登録解除する【NGKI2923】.

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，det_memをサポートしない【HRPS0158】.

【μITRON4.0/PX仕様との関係】

静的APIによって登録したメモリオブジェクトは，登録を解除することができないこととした.

μITRON4.0/PX仕様では，baseはメモリオブジェクトに含まれる番地を指定するものとしていたが，この仕様では，メモリオブジェクトの先頭番地でなければ ならないものとした.

prb_mem	メモリ領域に対するアクセス権のチェック（TP）〔NGKI2924〕

〔C言語API〕

```
ER ercd = prb_mem(const void *base, SIZE size, ID tskid, MODE pmmode)
```

【パラメータ】

void *	base	メモリ領域の先頭番地
SIZE	size	メモリ領域のサイズ（バイト数）
ID	tskid	アクセス元のタスクのID番号
MODE	pmmode	アクセスモード

【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〔NGKI2925〕
E_ID	不正ID番号 ・tskidが有効範囲外〔NGKI2927〕
E_PAR	パラメータエラー ・sizeが0〔NGKI2929〕 ・その他の条件については機能の項を参照
E_NOEXS	オブジェクト未登録 ・baseで指定した番地を含むメモリオブジェクトが登録され □ていない〔NGKI2930〕 ・tskidで指定したタスクが未登録（D）〔NGKI3425〕
E_OACV	オブジェクトアクセス違反 ・対象メモリ領域を含むメモリオブジェクトに対する参照操 □作が許可されていない〔NGKI2931〕 ・tskidで指定したタスクに対する参照操作が許可されていな □い〔NGKI3426〕
E_MACV	メモリアクセス違反 ・条件については機能の項を参照
E_OBJ	オブジェクト状態エラー ・対象メモリ領域がメモリオブジェクトの境界を越えている □〔NGKI2932〕

【機能】

tskidで指定したタスクから，baseとsizeで指定したメモリ領域（対象メモリ領域）に対して，pmmodeで指定した種別のアクセスが許可されているかをチェックする．アクセスが許可されている場合にE_OK，そうでない場合にE_MACVが返〔NGKI2933〕．tskidで指定したタスクがカーネルドメインに属する場合，

E_MACVが返ることはない【NGKI2934】．

pmmodeには，TPM_WRITE（=0x01U），TPM_READ（=0x02U），TPM_EXEC（=0x04U）のいずれか，またはそれらの内のいくつかのビット毎論理和（C言語の"|"）を指定することができる【NGKI2935】．TPM_WRITE，TPM_READ，TPM_EXECを指定した場合には，それぞれ，読出しアクセス，書込みアクセス，実行アクセスが許可されているかをチェックする【NGKI2936】．また，いくつかのビット毎論理和を指定した場合には，それらに対応した種別のアクセスがすべて許可されているかをチェックする【NGKI2937】．pmmodeにそれ以外の値を指定した場合には，E_PARエラーとなる【NGKI2938】．

tskidにTSK_SELF（=0）を指定すると，自タスクから対象メモリ領域に対して アクセスが許可されているかをチェックする【NGKI2939】．

【μITRON4.0/PX仕様との関係】

アクセスする主体の指定方法を，保護ドメインによる指定（domid）から，タスクによる指定（tskid）に変更した．また，pmmodeに指定できるアクセス種別にTPM_EXECを追加し，TPM_WRITEとTPM_READの値を入れ換えた．CPUロック状態からも呼び出せるものとした．

【仕様決定の理由】

prb_memを，CPUロック状態からも呼び出せるものとしたのは，次の理由による．prb_memは，拡張サービスコールの中で，タスクから渡されたポインタが，そのタスクからアクセスできる領域であるかを調べるために用いることを想定している．拡張サービスコールの中には，CPUロック状態でも呼び出せるものがあり，そのような拡張サービスコールを実現するには，prb_memがCPUロック状態から呼び出せることが必要である．

なお，prb_memを非タスクコンテキストから呼び出すことはできないが，非タスクコンテキストで実行される処理単位は必ずカーネルドメインに属するために，prb_memを使ってアクセス権を調べる必要がないことから，支障がない．

ref_mem	メモリオブジェクトの状態参照（TP）

【C言語API】

```
ER ercd = ref_mem(const void *base, T_RMEM *pk_rmem)
```

☆未完成

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，ref_memをサポートしない．

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の属性を設定しておく必要がある【NGKI2940】．割込みサービスルーチンは，カーネル内の割込みハンドラを経由して呼び出される【NGKI2941】．

ただし，カーネルが用意する割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込みハンドラを用意することも可能である【NGKI2942】．この場合にも，割込みハンドラをカーネルに登録する前に，割込みハンドラの登録対象となる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を設定しておく必要がある【NGKI2943】．

割込み要求ラインの属性を設定する際に指定する割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【NGKI2944】．

```
TA_ENAINT    0x01U    割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込み要求ライン属性を指定できる場合がある【NGKI2945】．ターゲット定義の割込み要求ライン属性として，次の属性を予約している【NGKI2946】．

```
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_HIGHLEVEL    ハイレベルトリガ
TA_BROADCAST    すべてのプロセッサで割込みを処理（マルチプロセッサ対応カーネルの場合）
```

割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．割込みサービスルーチンは，割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【NGKI2947】．

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した場合，それらの割込みサービスルーチンは，割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【NGKI2948】．割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【NGKI2949】．

保護機能対応カーネルにおいて，割込みサービスルーチンが属することのできる保護ドメインは，カーネルドメインに限られる【NGKI2950】．

割込みサービスルーチン属性に指定できる属性はない【NGKI2951】．そのため割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【NGKI2952】．

C言語による割込みサービスルーチンの記述形式は次の通り【NGKI2953】．

```
void interrupt_service_routine(intptr_t exinf)
{
    割込みサービスルーチン本体
}
```

exinfには，割込みサービスルーチンの拡張情報が渡される【NGKI2954】．

割込みハンドラは，カーネルが実行を制御する処理単位である．割込みハンドラは，割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【NGKI2955】．

保護機能対応カーネルにおいて，割込みハンドラは，カーネルドメインに属する【NGKI2956】．

割込みハンドラを登録する際に指定する割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【NGKI2957】．

```
TA_NONKERNEL    0x02U    カーネル管理外の割込み
```

TA_NONKERNELを指定しない場合，カーネル管理の割込みとなる【NGKI2958】．また，ターゲットによっては，その他のターゲット定義の割込みハンドラ属性を指定できる場合がある【NGKI2959】．

C言語による割込みハンドラの記述形式は次の通り【NGKI2960】．

```
void interrupt_handler(void)
{
    割込みハンドラ本体
}
```

割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     割込み優先度の最小値（最高値）【NGKI2961】
TMAX_INTPRI     割込み優先度の最大値（最低値，=-1）
```

```
TMIN_ISRPRI     割込みサービスルーチン優先度の最小値（=1）【NGKI2962】
TMAX_ISRPRI     割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【NGKI2963】
TOPPERS_SUPPORT_ENA_INT     ena_intがサポートされている【NGKI2964】
```

【使用上の注意】

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ている場合に対応するために，割込みサービスルーチンは，それが処理する割込み要求が発生しているかをチェックし，割込み要求が発生していない場合には何もせ ずにリターンするように実装すべきである．

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは，割込みサービスルーチン優先度の最大値（=TMAX_ISRPRI）は16に固定されている【ASPS0192】．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【ASPS0193】．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，割込みサービスルーチン優先度の最大値（=TMAX_ISRPRI）は16に固定されている【FMPS0159】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，割込みサービスルーチン優先度の最大値（=TMAX_ISRPRI）は16に固定されている【HRPS0159】．

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは，割込みサービスルーチン優先度の最大値（=TMAX_ISRPRI）は16に固定されている【SSPS0137】．

【μITRON4.0仕様との関係】

割込み要求ラインの属性，割込み優先度，割込みサービスルーチン優先度は，
μITRON4.0仕様にない概念であり，TMIN_INTPRI，TMAX_INTPRI，TMIN_ISRPRI，
TMAX_ISRPRIは，μITRON4.0仕様に定義のないカーネル構成マクロである．また，
TA_NONKERNELは，μITRON4.0仕様に定義のない割込みハンドラ属性である．

```
CFG_INT      割込み要求ラインの属性の設定（S）〔NGKI2965〕
cfg_int      割込み要求ラインの属性の設定（TD）〔NGKI2966〕
```

〔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        割込み番号
    T_CINT *     pk_cint      割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       割込み要求ライン属性
    PRI          intpri       割込み優先度
```

【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（sS）【NGKI2967】 ・CPUロック状態からの呼出し（sS）【NGKI2968】
E_RSATR	予約属性 ・intatrが無効【NGKI2969】 ・属するクラスの指定が有効範囲外（sM）【NGKI2970】 ・クラスの囲みの中に記述されていない（SM）【NGKI2971】 ・その他の条件については機能の項を参照
E_PAR	パラメータエラー ・intnoが有効範囲外【NGKI2972】 ・intpriが有効範囲外【NGKI2973】 ・その他の条件については機能の項を参照
E_OACV	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（sP） 【NGKI2974】
E_MACV	メモリアクセス違反 ・pk_cint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（sP）【NGKI2975】
E_OBJ	オブジェクト状態エラー ・対象割込み要求ラインに対して属性が設定済み（S）【NGKI2976】 ・その他の条件については機能の項を参照

【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）に対して，各パラ メータで指定した属性を設定する【NGKI2977】．

対象割込み要求ラインの割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【NGKI2978】．

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【NGKI2979】．

cfg_intにおいて，ターゲット定義で，複数の割込み要求ラインの割込み優先度 が連動して設定される場合がある【NGKI2980】．

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【NGKI2981】．ターゲット定義の拡張で，カーネル管理外の割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【NGKI2982】．このように拡張されている場合，カーネル管理外の割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【NGKI2983】．逆に，カーネル管理の割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【NGKI2984】．

対象割込み要求ラインに対して，設定できない割込み要求ライン属性をintatr に指定した場合にはE_RSATRエラー，設定できない割込み優先度をintpriに指定した場合にはE_PARエラーとなる【NGKI2985】．ここで，設定できない割込み要求ライン属性／割込み優先度には，ターゲット定義の制限によって設定できない値も含む【NGKI2986】．また，マルチプロセッサ対応カーネルにおいて，cfg_intを呼び出したタスクが割り付けられているプロセッサから，対象割込み要求ラインの属性を設定できない場合も，これに該当する【NGKI2987】．

保護機能対応カーネルにおいて，CFG_INTは，カーネルドメインの囲みの中に記 述しなければならない．そうでない場合には，E_RSATRエラーとなる【NGKI2989】．また，cfg_intはカーネルオブジェクトを登録するサービスコー ルではないため，割込み要求ライン属性にTA_DOM(domid)を指定した場合にはE_RSATRエラーとなる【NGKI2990】．ただし，TA_DOM(TDOM_SELF)を指定した場合には，指定が無視され，E_RSATRエラーは検出されない【NGKI2991】．

マルチプロセッサ対応カーネルで，CFG_INTの記述が，対象割込み要求ラインに 対して登録された割込みサービスルーチン（または対象割込み番号に対応する割込みハンドラ番号に対して登録された割込みハンドラ）と異なるクラスの囲 み中にある場合には，E_RSATRエラーとなる【NGKI2992】．

【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込み要求ライン属性／割込み優先度は，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，対象割込み要求ラインに対して，トリガモードや割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，複数の割込み要求ラインの割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，異なる割込み要求ラインに対して，同一の割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，同一の割込み優先度しか設定できない割込み要求ラインに対して異なる割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【ASPS0194】．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，CFG_INTのみをサポートする【FMPS0160】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，CFG_INTのみをサポートする【HRPS0160】．

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは，CFG_INTのみをサポートする【SSPS0138】．

【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      割込みサービスルーチンの生成（S）【NGKI2993】
ATT_ISR      割込みサービスルーチンの追加（S）【NGKI2994】
acre_isr     割込みサービスルーチンの生成（TD）【NGKI2995】
```

【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（CRE_ISRの場合）
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（sS）【NGKI2996】 ・CPUロック状態からの呼出し（sS）【NGKI2997】
E_RSATR	予約属性 ・isratrが無効【NGKI2998】 ・属する保護ドメインの指定が有効範囲外またはカーネルドメイン以外（sP）【NGKI2999】 ・カーネルドメインの囲みの中に記述されていない（SP）【NGKI3000】 ・属するクラスの指定が有効範囲外（sM）【NGKI3001】 ・クラスの囲みの中に記述されていない（SM）【NGKI3002】 ・その他の条件については機能の項を参照
E_PAR	パラメータエラー ・intnoが有効範囲外【NGKI3003】 ・isrがプログラムの先頭番地として正しくない【NGKI3004】 ・isrpriが有効範囲外【NGKI3005】
E_OACV	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（sP）【NGKI3006】
E_MACV	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（sP）【NGKI3007】
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（sD）【NGKI3008】
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（CRE_ISRの場合）【NGKI3009】 ・その他の条件については機能の項を参照

【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【NGKI3010】．

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【NGKI3011】．

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【NGKI3012】．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【NGKI3013】．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【NGKI3014】．

静的APIにおいては，isridはオブジェクト識別名，isratr，intno，isrpriは整数定数式パラメータ，exinfとisrは一般定数式パラメータである【NGKI3015】．

マルチプロセッサ対応カーネルで，生成する割込みサービスルーチンの属するクラスの割付け可能プロセッサが，intnoで指定した割込み要求ラインが接続されたプロセッサの集合に含まれていない場合には，E_RSATRエラーとなる【NGKI3016】．また，intnoで指定した割込み要求ラインに対して登録済みの割込みサービスルーチンがある場合に，生成する割込みサービスルーチンがそれと異なるクラスに属する場合にも，E_RSATRエラーとなる【NGKI3017】．さらに，ターゲット定義で，割込みサービスルーチンが属することができるクラスに制限がある場合がある【NGKI3018】．生成する割込みサービスルーチンの属するクラスが，ターゲット定義の制限に合致しない場合にも，E_RSATRエラーとなる【NGKI3019】．

静的APIにおいて，isrが不正である場合にE_PARエラーが検出されるか否かは，ターゲット定義である【NGKI3020】．

【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【ASPS0209】．ただし，動的生成機能拡張パッケージでは，acre_isrもサポートする【ASPS0195】．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，ATT_ISRのみをサポートする【FMPS0161】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，ATT_ISRのみをサポートする【HRPS0161】．ただし，動的生成機能拡張パッケージでは，acre_isrもサポートする【HRPS0208】．

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは，ATT_ISRのみをサポートする【SSPS0139】．

【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．

AID_ISR	割付け可能な割込みサービスルーチンIDの数の指定（SD）【NGKI3021】

【静的API】

```
AID_ISR(uint_t noisr)
```

【パラメータ】

uint_t	noisr	割付け可能な割込みサービスルーチンIDの数

【エラーコード】

E_RSATR	予約属性 ・保護ドメインの囲みの中に記述されている（P）【NGKI3439】 ・クラスの囲みの中に記述されていない（M）【NGKI3022】
E_PAR	パラメータエラー ・noisrが負の値【NGKI3287】

【機能】

noisrで指定した数の割込みサービスルーチンIDを，割込みサービスルーチンを生成するサービスコールによって割付け可能な割込みサービスルーチンIDとして確保する【NGKI3024】．

noisrは整数定数式パラメータである【NGKI3025】．

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルの動的生成機能拡張パッケージでは，AID_ISRをサポートする【ASPS0219】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルの動的生成機能拡張パッケージでは，AID_ISRをサポートする【HRPS0220】．

```
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〕【NGKI3026】
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〕【NGKI3027】
```

【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（S）【NGKI3028】
            ・CPUロック状態からの呼出し（S）【NGKI3029】
E_ID        不正ID番号
            ・isridが有効範囲外（S）【NGKI3030】
E_RSATR     予約属性
            ・カーネルドメインの囲みの中に記述されていない（S）【NGKI3031】
            ・対象割込みサービスルーチンが属するクラスの囲みの中に
            記述されていない（SM）【NGKI3032】
E_NOEXS     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【NGKI3033】
E_OACV      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）（S）【NGKI3034】
E_MACV      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）（S）【NGKI3035】
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（S）
            【NGKI3036】
            ・対象割込みサービスルーチンに対してアクセス許可ベクタ
            が設定済み（S）【NGKI3037】
```

【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【NGKI3038】．

静的APIにおいては，isridはオブジェクト識別名，acptn1~acptn4は整数定数式パラメータである【NGKI3039】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，SAC_ISR，sac_isrをサポートしない【HRPS0162】．ただし，動的生成機能拡張パッケージでは，sac_isrをサポートする【HRPS0209】．

【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）【NGKI3040】
```

【C言語API】

```
ER ercd = del_isr(ID isrid)
```

【パラメータ】

ID	isrid	対象割込みサービスルーチンの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【NGKI3041】 ・CPUロック状態からの呼出し【NGKI3042】
E_ID	不正ID番号 ・isridが有効範囲外【NGKI3043】
E_NOEXS	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【NGKI3044】
E_OACV	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（P）【NGKI3045】
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【NGKI3046】

【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）を削 除する．具体的な振舞いは以下の通り．

対象割込みサービスルーチンの登録が解除され，その割込みサービスルーチン IDが未使用の状態に戻される【NGKI3047】．

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは，del_isrをサポートしない【ASPS0197】．ただし，動的生成 機能拡張パッケージでは，del_isrをサポートする【ASPS0198】．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，del_isrをサポートしない【FMPS0163】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，del_isrをサポートしない【HRPS0163】．ただし，動的生成 機能拡張パッケージでは，del_isrをサポートする【HRPS0210】．

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは，del_isrをサポートしない【SSPS0141】．

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をサポートしない．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，ref_isrをサポートしない．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，ref_isrをサポートしない．

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは，ref_isrをサポートしない．

```
DEF_INH      割込みハンドラの定義（S）【NGKI3048】
def_inh      割込みハンドラの定義（TD）【NGKI3049】
```

【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パケットへ
                         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〔S〕〔NGKI3050〕 ・CPUロック状態からの呼出し〔S〕〔NGKI3051〕
E_RSATR	予約属性 ・inhatrが無効〔NGKI3052〕 ・属するクラスの指定が有効範囲外〔SM〕〔NGKI3053〕 ・クラスの囲みの中に記述されていない〔SM〕〔NGKI3054〕 ・その他の条件については機能の項を参照
E_PAR	パラメータエラー ・inhnoが有効範囲外〔NGKI3055〕 ・inthdrがプログラムの先頭番地として正しくない〔NGKI3056〕 ・その他の条件については機能の項を参照
E_OACV	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〔SP〕 〔NGKI3057〕
E_MACV	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〔SP〕〔NGKI3058〕
E_OBJ	オブジェクト状態エラー ・条件については機能の項を参照

【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〔NGKI3059〕．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〔NGKI3060〕．

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〔NGKI3061〕．

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〔NGKI3062〕．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〔NGKI3063〕．

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〔NGKI3064〕．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〔NGKI3065〕．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〔NGKI3066〕．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〔NGKI3067〕．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〔NGKI3068〕．

保護機能対応カーネルにおいて，DEF_INHは，カーネルドメインの囲みの中に記述しなければならない．そうでない場合には，E_RSATRエラーとなる〔NGKI3070〕．また，def_inhで割込みハンドラを定義する場合には，割込みハンドラの属する保護ドメインを設定する必要はなく，割込みハンドラ属性にTA_DOM(domid)を指定した場合にはE_RSATRエラーとなる〔NGKI3071〕．ただし，TA_DOM(TDOM_SELF)を指定した場合には，指定が無視され，E_RSATRエラーは検 出されない〔NGKI3072〕．

マルチプロセッサ対応カーネルで，登録する割込みハンドラの属するクラスの初期割付けプロセッサが，その割込みが要求されるプロセッサでない場合には，

E_RSATRエラーとなる【NGKI3073】．また，ターゲット定義で，割込みハンドラ が属することができるクラスに制限がある場合がある【NGKI3074】．登録する割込みハンドラの属するクラスが，ターゲット定義の制限に合致しない場合にも，E_RSATRエラーとなる【NGKI3075】．

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【NGKI3076】．また，対象割込みハンドラ番号 に対して定義された割込みハンドラが，静的APIで定義されたものである場合には，ターゲット定義でE_OBJエラーとなる場合がある【NGKI3077】．

ターゲット定義で，対象割込みハンドラを定義（または定義解除）できない場 合には，E_PARエラーとなる【NGKI3078】．具体的には，マルチプロセッサ対応カーネルにおいて，def_inhを呼び出したタスクが割り付けられているプロセッサから，対象割込みハンドラを定義（または定義解除）できない場合が，これに該当する【NGKI3079】．

静的APIにおいて，inthdrが不正である場合にE_PARエラーが検出されるか否かは，ターゲット定義である【NGKI3080】．

【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【ASPS0199】．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，DEF_INHのみをサポートする【FMPS0164】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，DEF_INHのみをサポートする【HRPS0164】．

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは，DEF_INHのみをサポートする【SSPS0142】．

【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）【NGKI3081】
```

【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_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【NGKI3082】
E_NOSPT     未サポートエラー
            ・条件については機能の項を参照
E_PAR       パラメータエラー
            ・intnoが有効範囲外【NGKI3083】
            ・その他の条件については機能の項を参照
E_OACV      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（P）
            □【NGKI3084】
E_OBJ       オブジェクト状態エラー
            ・対象割込み要求ラインに対して割込み要求ライン属性が設
            □定されていない【NGKI3085】
```

【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【NGKI3086】.

ターゲット定義で，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で きない場合には，E_PARエラーとなる【NGKI3087】．具体的には，対象割込み要求ラインに対して割込み要求禁止フラグがサポートされていない場合や，マルチプロセッサ対応カーネルにおいて，dis_intを呼び出したタスクが割り付けられているプロセッサから，対象割込み要求ラインの割込み要求禁止フラグが操 作できない場合が，これに該当する．

ターゲット定義で，割込み要求禁止フラグの振舞いが，この仕様の規定と異な る場合がある【NGKI3089】．特にマルチプロセッサ対応カーネルでは，あるプロセッサからdis_intを呼び出して割込み要求禁止フラグをセットしても，他の プロセッサに対しては割込みがマスクされない場合がある．

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【NGKI3091】．
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【NGKI3092】．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【NGKI3093】．

【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）【NGKI3094】
```

【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_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【NGKI3095】
E_NOSPT	未サポートエラー ・条件については機能の項を参照
E_PAR	パラメータエラー ・intnoが有効範囲外【NGKI3096】 ・その他の条件については機能の項を参照
E_OACV	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（P） 【NGKI3097】
E_OBJ	オブジェクト状態エラー ・対象割込み要求ラインに対して割込み要求ライン属性が設 定されていない【NGKI3098】

【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【NGKI3099】．

ターゲット定義で，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で きない場合には，E_PARエラーとなる【NGKI3100】．具体的には，対象割込み要求ラインに対して割込み要求禁止フラグがサポートされていない場合や，マルチプロセッサ対応カーネルにおいて，ena_intを呼び出したタスクが割り付けられているプロセッサから，対象割込み要求ラインの割込み要求禁止フラグが操 作できない場合が，これに該当する．

ターゲット定義で，割込み要求禁止フラグの振舞いが，この仕様の規定と異な る場合がある【NGKI3102】．特にマルチプロセッサ対応カーネルでは，あるプロセッサからena_intを呼び出して割込み要求禁止フラグをクリアしても，他の プロセッサに対しては割込みがマスク解除されない場合がある．

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【NGKI3104】．
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【NGKI3105】．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【NGKI3106】．

【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(INTNO intno, 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をサポートしない．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，ref_intをサポートしない．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，ref_intをサポートしない．

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは，ref_intをサポートしない．

【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．

```
chg_ipm      割込み優先度マスクの変更〔T〕【NGKI3107】
```

【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〔NGKI3108〕 ・CPUロック状態からの呼出し〔NGKI3109〕	
E_PAR	パラメータエラー ・条件については機能の項を参照	
E_OACV	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〔P〕 〔NGKI3110〕	

【機能】

割込み優先度マスクを，intpriで指定した値に変更する〔NGKI3111〕．

intpriは，TMIN_INTPRI以上，TIPM_ENAALL以下でなければならない．そうでない場合には，E_PARエラーとなる〔NGKI3113〕．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〔NGKI3114〕．

【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．また，タスク例外処理ルーチンの実行が開始される可能性がある．

〔TOPPERS/SSPカーネルにおける規定〕

SSPカーネルでは，chg_ipmをサポートしない〔SSPS0143〕．

〔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．

get_ipm	割込み優先度マスクの参照〔T〕〔NGKI3115〕

〔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〔NGKI3116〕
            ・CPUロック状態からの呼出し〔NGKI3117〕
E_OACV      オブジェクトアクセス違反
            ・システム状態に対する参照操作が許可されていない（P）
            □〔NGKI3118〕
E_MACV      メモリアクセス違反
            ・p_intpri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（P）〔NGKI3119〕
```

【機能】

割込み優先度マスクの現在値を参照する．参照した割込み優先度マスクは，p_intpriが指すメモリ領域に返される〔NGKI3120〕．

〔TOPPERS/SSPカーネルにおける規定〕

SSPカーネルでは，get_ipmをサポートしない〔SSPS0144〕．

〔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．

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〔NGKI3121〕．

保護機能対応カーネルにおいて，CPU例外ハンドラは，カーネルドメインに属する〔NGKI3122〕．

CPU例外ハンドラ属性に標準で指定できる属性はないが，ターゲットによっては，ターゲット定義のCPU例外ハンドラ属性を指定できる場合がある〔NGKI3123〕．ターゲット定義のCPU例外ハンドラ属性として，次の属性を予約している〔NGKI3124〕．

```
TA_DIRECT           CPU例外ハンドラを直接呼び出す
```

C言語によるCPU例外ハンドラの記述形式は次の通り〔NGKI3125〕．

```
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〔NGKI3126〕．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）〔NGKI3127〕
def_exc     CPU例外ハンドラの定義（TD）〔NGKI3128〕
```

【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（S）［NGKI3129］ ・CPUロック状態からの呼出し（S）［NGKI3130］
E_RSATR	予約属性 ・excatrが無効［NGKI3131］ ・属するクラスの指定が有効範囲外（SM）［NGKI3132］ ・クラスの囲みの中に記述されていない（SM）［NGKI3133］ ・その他の条件については機能の項を参照
E_PAR	パラメータエラー ・excnoが有効範囲外［NGKI3134］ ・exchdrがプログラムの先頭番地として正しくない［NGKI3135］
E_OACV	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（SP）□［NGKI3136］
E_MACV	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（SP）［NGKI3137］
E_OBJ	オブジェクト状態エラー ・条件については機能の項を参照

【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〔NGKI3138〕．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〔NGKI3139〕．

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〔NGKI3140〕．

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〔NGKI3141〕．

保護機能対応カーネルにおいて，DEF_EXCは，カーネルドメインの囲みの中に記述しなければならない．そうでない場合には，E_RSATRエラーとなる〔NGKI3143〕．また，def_excでCPU例外ハンドラを定義する場合には，CPU例外ハンドラの属する保護ドメインを設定する必要はなく，CPU例外ハンドラ属性にTA_DOM(domid)を指定した場合にはE_RSATRエラーとなる〔NGKI3144〕．ただし，TA_DOM(TDOM_SELF)を指定した場合には，指定が無視され，E_RSATRエラーは検出されない〔NGKI3145〕．

マルチプロセッサ対応カーネルで，登録するCPU例外ハンドラの属するクラスの初期割付けプロセッサが，そのCPU例外が発生するプロセッサでない場合には，E_RSATRエラーとなる〔NGKI3146〕．

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〔NGKI3147〕．また，対象CPU例外ハンドラ番号に対して定義されたCPU例外ハンドラが，静的APIで定義されたものである場合には，ターゲット定義でE_OBJエラーとなる場合がある〔NGKI3148〕．

静的APIにおいて，exchdrが不正である場合にE_PARエラーが検出されるか否かは，ターゲット定義である〔NGKI3149〕．

〔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〔ASPS0200〕．

〔TOPPERS/FMPカーネルにおける規定〕

FMPカーネルでは，DEF_EXCのみをサポートする〔FMPS0165〕．

〔TOPPERS/HRP2カーネルにおける規定〕

HRP2カーネルでは，DEF_EXCのみをサポートする〔HRPS0165〕．

〔TOPPERS/SSPカーネルにおける規定〕

SSPカーネルでは，DEF_EXCのみをサポートする〔SSPS0145〕．

〔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）〔NGKI3150〕
```

〔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，カーネル管理外のCPU例外でなく，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保留状態でなかった場合にfalse，そうでない場合にtrueが返る〔NGKI3151〕．

保護機能対応のカーネルにおいて，xsns_dpnをタスクコンテキストから呼び出した場合には，trueが返る〔NGKI3152〕．

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〔NGKI3153〕．それ以外の値を渡した場合の動作は保証されない〔NGKI3552〕．

【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，xsns_dpn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．

〔TOPPERS/SSPカーネルにおける規定〕

SSPカーネルでは，xsns_dpnをサポートしない〔SSPS0146〕．

〔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．

【仕様決定の理由】

保護機能対応のカーネルにおいては，xsns_dpnをユーザドメインから呼び出すことは禁止すべきである．ユーザドメインの実行中は，必ずタスクコンテキストであるため，xsns_dpnをタスクコンテキストから呼び出した場合に必ずtrueを返す仕様とすることで，xsns_dpnをユーザドメインから呼び出すことを実質的に禁止している．

xsns_xpn	CPU例外発生時のタスク例外処理保留状態の参照〔TI〕〔NGKI3154〕

〔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例外発生時にタスク例外処理ルーチンを実行開始できない状態であったかを 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，カーネル管理外のCPU例外でなく，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開始できる状態であった場合にfalse，そうでない場合にtrueが返る【NGKI3155】．

保護機能対応カーネルにおいて，CPU例外が発生したタスクがユーザタスクの場合には，ユーザスタック領域の残りが少なく，タスク例外処理ルーチンを実行開始できない（タスク例外処理ルーチンを実行開始しようとすると，タスク例 外実行開始時スタック不正例外が発生する）場合にも，trueを返す【NGKI3156】．

保護機能対応のカーネルにおいて，xsns_xpnをタスクコンテキストから呼び出 した場合には，trueが返る【NGKI3157】．

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【NGKI3158】．

【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，xsns_xpn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．

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは，xsns_xpnをサポートしない【SSPS0147】．

【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．

【仕様決定の理由】

保護機能対応のカーネルにおいては，xsns_xpnをユーザドメインから呼び出すことは禁止すべきである．ユーザドメインの実行中は，必ずタスクコンテキストであるため，xsns_xpnをタスクコンテキストから呼び出した場合に必ずtrue を返す仕様とすることで，xsns_xpnをユーザドメインから呼び出すことを実質 的に禁止している．

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【NGKI3159】．特権モードで実行するルーチンを，拡張サービスコールと呼ぶ．拡張サービスコールは，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【NGKI3160】．

保護機能対応カーネルにおいて，拡張サービスコールは，カーネルドメインに属する【NGKI3161】．拡張サービスコールは，それを呼び出す処理単位とは別の処理単位であり，拡張サービスコールからカーネルオブジェクトをアクセスする場合には，拡張サービスコールがアクセスの主体となる【NGKI3162】．そのため，拡張サービスコールからは，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．

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【NGKI3163】．

拡張サービスコール属性に指定できる属性はない【NGKI3686】．そのため拡張サービスコール属性には，TA_NULLを指定しなければならない【NGKI3687】．

C言語による拡張サービスコールの記述形式は次の通り【NGKI3164】．

```
ER_UINT extended_svc(intptr_t par1, intptr_t par2, intptr_t par3,
                             intptr_t par4, intptr_t par5, ID cdmid)
{
      拡張サービスコール本体
}
```

cdmidには，拡張サービスコールを呼び出した処理単位が属する保護ドメインのID番号が渡される【NGKI3165】．すなわち，拡張サービスコールから呼び出した場合にはTDOM_KERNEL（=-1）が，タスク本体（拡張サービスコールを除く）から呼び出した場合にはそのタスク（自タスク）の属する保護ドメインIDが渡される．

par1~par5には，拡張サービスコールに対するパラメータが渡される【NGKI3166】．

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．

TMAX_FNCD	拡張サービスコールの機能番号の最大値（動的生成対応カーネルでは，登録できる拡張サービスコールの数に一致）【NGKI3167】

【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ない【ASPS0201】．

【TOPPERS/FMPカーネルにおける規定】

FMP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ない【FMPS0166】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，拡張サービスコール管理機能をサポートする【HRPS0166】．

【TOPPERS/SSPカーネルにおける規定】

SSP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ない【SSPS0148】．

【未決定事項】

動的生成対応カーネルにおいてTMAX_FNCDを設定する方法については，現時点で は未決定である．

【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に，それを呼び出した処理単位が属する保護ドメインのID 番号を渡す機能を追加した．

TMAX_FNCD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．

```
DEF_SVC     拡張サービスコールの定義〔SP〕〔NGKI3168〕
def_svc     拡張サービスコールの定義〔TPD〕〔NGKI3169〕
```

【静的API】

```
    DEF_SVC(FN fncd, { ATR svcatr, EXTSVC extsvc, SIZE stksz })
```

【C言語API】

```
    ER ercd = def_svc(FN fncd, const T_DSVC *pk_dsvc)
```

【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      extsvc      拡張サービスコールの先頭番地
    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（S）【NGKI3170】 ・CPUロック状態からの呼出し（S）【NGKI3171】
E_RSATR	予約属性 ・svcatrが無効【NGKI3172】 ・その他の条件については機能の項を参照
E_PAR	パラメータエラー ・fncdが0または負の値【NGKI3173】 ・fncdがTMAX_FNCDよりも大きい（S）【NGKI3174】 ・extsvcがプログラムの先頭番地として正しくない【NGKI3175】 ・stkszが負の値（S）【NGKI3290】
E_OACV	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（S） 【NGKI3176】
E_MACV	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（S）【NGKI3177】
E_OBJ	オブジェクト状態エラー ・条件については機能の項を参照

【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【NGKI3178】．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【NGKI3179】．

静的APIにおいては，fncd，svcatr，stkszは整数定数式パラメータ，svchdrは一般定数式パラメータである【NGKI3180】．

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【NGKI3181】．

DEF_SVCは，カーネルドメインの囲みの中に記述しなければならない．そうでない場合には，E_RSATRエラーとなる【NGKI3183】．また，def_svcで拡張サービスコールを定義する場合には，拡張サービスコールの属する保護ドメインを設定する必要はなく，拡張サービスコール属性にTA_DOM(domid)を指定した場合にはE_RSATRエラーとなる【NGKI3184】．ただし，TA_DOM(TDOM_SELF)を指定した場合には，指定が無視され，E_RSATRエラーは検出されない【NGKI3185】．

マルチプロセッサ対応カーネルでは，DEF_SVCは，クラスの囲みの外に記述しなければならない．そうでない場合には，E_RSATRエラーとなる【NGKI3187】．また，def_svcで拡張サービスコールを定義する場合には，拡張サービスコールの属するクラスを設定する必要はなく，拡張サービスコール属性にTA_CLS(clsid)を指定した場合にはE_RSATRエラーとなる【NGKI3188】．ただし，TA_CLS(TCLS_SELF)を指定した場合には，指定が無視され，E_RSATRエラーは検出されない【NGKI3189】．

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【NGKI3190】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，DEF_SVCのみをサポートする【HRPS0167】．

【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．

extsvc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〕【NGKI3191】	

【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 ・条件については機能の項を参照
E_RSFN	予約機能コード ・fncdが0または負の値【NGKI3192】 ・fncdがTMAX_FNCDよりも大きい【NGKI3193】 ・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【NGKI3194】
E_NOMEM	メモリ不足 ・条件については機能の項を参照

＊その他，拡張サービスコール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【NGKI3195】．

また，タスクコンテキストから呼び出した場合には，次のエラーが検出される【NGKI3196】．スタック（ユーザタスクの場合はシステムスタック）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【NGKI3197】．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【NGKI3198】．

【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．

拡張サービスコール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．

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，cal_svcから返るエラー（E_SYS，E_RSFN，E_NOMEM）について規定した．

【仕様決定の理由】

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．

4.12. システム構成管理機能

システム構成管理機能には，非タスクコンテキスト用スタック領域を設定する機能，初期化ルーチンと終了処理ルーチンを登録する機能，カーネルのコンフィギュレーション情報やバージョン情報を参照する機能が含まれる．

非タスクコンテキスト用スタック領域は，非タスクコンテキストで実行される 処理単位が用いるスタック領域である．

保護機能対応カーネルにおいて，非タスクコンテキスト用のスタック領域は，カーネルの用いるオブジェクト管理領域と同様に扱われる【NGKI3199】．

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 始の直前に，カーネル非動作状態で実行される【NGKI3200】．

保護機能対応カーネルにおいて，初期化ルーチンは，カーネルドメインに属する【NGKI3201】．

初期化ルーチン属性に指定できる属性はない【NGKI3202】．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【NGKI3203】．

C言語による初期化ルーチンの記述形式は次の通り【NGKI3204】．

```
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【NGKI3205】．

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 終了の直後に，カーネル非動作状態で実行される【NGKI3206】．

保護機能対応カーネルにおいて，終了処理ルーチンは，カーネルドメインに属 する【NGKI3207】．

終了処理ルーチン属性に指定できる属性はない【NGKI3208】．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【NGKI3209】．

C言語による終了処理ルーチンの記述形式は次の通り【NGKI3210】．

```
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【NGKI3211】．

【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．

```
LMT_DOM        保護ドメインに対する制限の設定〔SP〕〔NGKI3441〕
```

【静的API】

```
    LMT_DOM({ PRI mintpri })
```

【パラメータ】

```
*保護ドメインに対する制限の設定情報
    PRI          mintpri        指定できる最高のタスク優先度
```

【エラーコード】

```
    E_RSATR      予約属性
                 ・ユーザドメインの囲みの中に記述されていない〔NGKI3442〕
                 ・クラスの囲みの中に記述されている〔M〕〔NGKI3443〕
    E_OBJ        オブジェクト状態エラー
                 ・保護ドメインに対する制限が設定済み〔NGKI3444〕
    E_PAR        パラメータエラー
                 ・mintpriが有効範囲外〔NGKI3445〕
```

【機能】

パラメータで指定した保護ドメインに対する制限の設定情報に従って，ユーザ ドメインに対する制限を設定する〔NGKI3446〕．

mintpriは整数定数式パラメータである〔NGKI3447〕．

LMT_DOMにより保護ドメインに対する制限を設定しないユーザドメインに対して は，指定できる最高のタスク優先度はTMIN_TPRI+1に設定される〔NGKI3448〕．

〔μITRON4.0/PX仕様との関係〕

μITRON4.0/PX仕様に定義されていない静的API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〕〔NGKI3212〕
```

【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（バイト数）
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_RSATR	予約属性 ・カーネルドメインの囲みの中に記述されていない（P）【NGKI3213】 ・クラスの囲みの中に記述されていない（M）【NGKI3214】
E_PAR	パラメータエラー ・条件については機能の項を参照
E_NOMEM	メモリ不足 ・非タスクコンテキスト用スタック領域が確保できない【NGKI3215】
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設定済み【NGKI3216】 ・その他の条件については機能の項を参照

【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【NGKI3217】．istkszに0以下の値を指定した時や，ターゲット定義の最小値よりも小さい値を指定した時には，E_PARエラーとなる【NGKI3254】．

istkszは整数定数式パラメータ，istkは一般定数式パラメータである．コンフィギュレータは，静的APIのメモリ不足（E_NOMEM）エラーを検出することができない【NGKI3218】．

istkをNULLとした場合，istkszで指定したサイズのスタック領域が，コンフィギュレータにより確保される【NGKI3219】．istkszにターゲット定義の制約に合致しないサイズを指定した時には，ターゲット定義の制約に合致するように大きい方に丸めたサイズで確保される【NGKI3220】．

istkにNULL以外を指定した場合，istkとistkszで指定したスタック領域は，アプリケーションで確保しておく必要がある【NGKI3221】．スタック領域をアプリケーションで確保する方法については，「2.15.3 カーネル共通マクロ」の節を参照すること．その方法に従わず，istkやistkszにターゲット定義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【NGKI3222】．

保護機能対応カーネルでは，istkとistkszで指定した非タスクコンテキスト用のスタック領域がカーネル専用のメモリオブジェクトに含まれない場合，E_OBJエラーとなる【NGKI3223】．

DEF_ICSにより非タスクコンテキスト用スタック領域を設定しない場合，ターゲット定義のデフォルトのサイズのスタック領域が，コンフィギュレータにより確保される【NGKI3224】．

マルチプロセッサ対応カーネルでは，非タスクコンテキスト用スタック領域はプロセッサ毎に確保する必要がある【NGKI3225】．DEF_ICSにより設定する非タスクコンテキスト用スタック領域は，DEF_ICSの記述をその囲みの中に含むクラスの初期割付けプロセッサが使用する【NGKI3226】．そのプロセッサに対してすでに非タスクコンテキスト用スタック領域が設定されている場合には，E_OBJエラーとなる【NGKI3227】．

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは，istkにはNULLを指定しなくてはならず，その場合でも，コンフィギュレータは非タスクコンテキスト用のスタック領域を確保しない【SSPS0149】．これは，SSPカーネルでは，すべての処理単位が共有スタック領域を使用し，非タスクコンテキストのみが用いるスタック領域を持たないためである．そのため，DEF_ICSの役割は，非タスクコンテキストが用いるスタック領域のサイズを指定することのみとなる．itskにNULL以外を指定した場合には，E_PARエラーとなる【SSPS0150】．

共有スタック領域の設定方法については，DEF_STKの項を参照すること．

【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていない静的APIである．

DEF_STK	共有スタック領域の設定（S）【NGKI3228】	

【静的API】

```
DEF_STK({ SIZE stksz, STK_T *stk })
```

【パラメータ】

*共有スタック領域の設定情報		
SIZE	stksz	共有スタック領域のサイズ（バイト数）
STK_T	stk	共有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 ・条件については機能の項を参照
E_NOMEM	メモリ不足 ・共有スタック領域が確保できない【NGKI3229】
E_OBJ	オブジェクト状態エラー ・共有スタック領域が設定済み

【サポートするカーネル】

DEF_STKは，TOPPERS/SSPカーネルのみがサポートする静的APIである．他のカーネルは，DEF_STKをサポートしない【NGKI3230】．

【機能】

各パラメータで指定した共有スタック領域の設定情報に従って，共有スタック領域を設定する【NGKI3231】．stkszに0以下の値を指定した時や，ターゲット定義の最小値よりも小さい値を指定した時には，E_PARエラーとなる【NGKI3255】．

stkszは整数定数式パラメータ，stkは一般定数式パラメータである．コンフィギュレータは，静的APIのメモリ不足（E_NOMEM）エラーを検出することができない【NGKI3232】．

stkをNULLとした場合，stkszで指定したサイズのスタック領域が，コンフィギュレータにより確保される【NGKI3233】．stkszにターゲット定義の制約に合致しないサイズを指定した時には，ターゲット定義の制約に合致するように大きい方に丸めたサイズで確保される【NGKI3234】．

stkにNULL以外を指定した場合，stkとstkszで指定したスタック領域は，アプリケーションで確保しておく必要がある【NGKI3235】．スタック領域をアプリケーションで確保する方法については，「2.15.3 カーネル共通マクロ」の節を参照すること．その方法に従わず，stkやstkszにターゲット定義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【NGKI3236】．

コンフィギュレータは，各タスクのスタック領域のサイズと，非タスクコンテキスト用のスタック領域のサイズから，共有スタック領域に必要なサイズを計

算する〔NGKI3237〕．DEF_STKにより共有スタック領域を設定しない場合，必要なサイズの共有スタック領域が，コンフィギュレータにより確保される〔NGKI3238〕．

stkszに指定したスタック領域のサイズが，共有スタック領域に必要なサイズよりも小さい場合，コンフィギュレータは警告メッセージを出力する〔NGKI3239〕．

〔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〕〔NGKI3240〕

【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が無効〔NGKI3241〕 ・カーネルドメインの囲みの中に記述されていない〔P〕〔NGKI3242〕
E_PAR	パラメータエラー ・inirtnがプログラムの先頭番地として正しくない〔NGKI3243〕

【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〔NGKI3244〕．

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〔NGKI3245〕．

inirtnが不正である場合にE_PARエラーが検出されるか否かは，ターゲット定義 である〔NGKI3246〕．

【補足説明】

マルチプロセッサ対応カーネルでは，クラスに属さないグローバル初期化ルーチンはマスタプロセッサで実行され，クラスに属するローカル初期化ルーチンはそのクラスの初期割付けプロセッサにより実行される．

ATT_TER	終了処理ルーチンの追加〔S〕〔NGKI3247〕

【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が無効【NGKI3248】
                ・カーネルドメインの囲みの中に記述されていない〔P〕【NGKI3249】
    E_PAR       パラメータエラー
                ・terrtnがプログラムの先頭番地として正しくない【NGKI3250】
```

【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【NGKI3251】．

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【NGKI3252】．

terrtnが不正である場合にE_PARエラーが検出されるか否かは，ターゲット定義である【NGKI3253】．

【補足説明】

マルチプロセッサ対応カーネルでは，クラスに属さないグローバル終了処理ルーチンはマスタプロセッサで実行され，クラスに属するローカル終了処理ルーチンはそのクラスの初期割付けプロセッサにより実行される．

【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．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，ref_cfgをサポートしない．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，ref_cfgをサポートしない．

【TOPPERS/SSPカーネルにおける規定】

SSP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．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，ref_verをサポートしない．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは，ref_verをサポートしない．

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは，ref_verをサポートしない．

5. 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)〔TD〕
ER ercd = sac_tsk(ID tskid, const ACVCT *p_acvct)〔TPD〕
ER ercd = del_tsk(ID tskid)〔TD〕
ER ercd = act_tsk(ID tskid)〔T〕
ER ercd = iact_tsk(ID tskid)〔I〕
ER ercd = mact_tsk(ID tskid, ID prcid)〔TM〕

ER ercd = imact_tsk(ID tskid, ID prcid)〔IM〕
ER_UINT actcnt = can_act(ID tskid)〔T〕
ER ercd = mig_tsk(ID tskid, ID prcid)〔TM〕
ER ercd = ext_tsk()〔T〕
ER ercd = ter_tsk(ID tskid)〔T〕
ER ercd = chg_pri(ID tskid, PRI tskpri)〔T〕
ER ercd = get_pri(ID tskid, PRI *p_tskpri)〔T〕
ER ercd = get_inf(intptr_t *p_exinf)〔T〕
ER ercd = ref_tsk(ID tskid, T_RTSK *pk_rtsk)〔T〕

(2) タスク付属同期機能

ER ercd = slp_tsk()〔T〕
ER ercd = tslp_tsk(TMO tmout)〔T〕
ER ercd = wup_tsk(ID tskid)〔T〕
ER ercd = iwup_tsk(ID tskid)〔I〕
ER_UINT wupcnt = can_wup(ID tskid)〔T〕
ER ercd = rel_wai(ID tskid)〔T〕
ER ercd = irel_wai(ID tskid)〔I〕
ER ercd = sus_tsk(ID tskid)〔T〕
ER ercd = rsm_tsk(ID tskid)〔T〕
ER ercd = dis_wai(ID tskid)〔TP〕
ER ercd = idis_wai(ID tskid)〔IP〕
ER ercd = ena_wai(ID tskid)〔TP〕
ER ercd = iena_wai(ID tskid)〔IP〕
ER ercd = dly_tsk(RELTIM dlytim)〔T〕

(3) タスク例外処理機能

ER ercd = def_tex(ID tskid, const T_DTEX *pk_dtex)〔TD〕
ER ercd = ras_tex(ID tskid, TEXPTN rasptn)〔T〕
ER ercd = iras_tex(ID tskid, TEXPTN rasptn)〔I〕
ER ercd = dis_tex()〔T〕
ER ercd = ena_tex()〔T〕
bool_t state = sns_tex()〔TI〕
ER ercd = ref_tex(ID tskid, T_RTEX *pk_rtex)〔T〕

(4) 同期・通信機能

セマフォ

ER_ID semid = acre_sem(const T_CSEM *pk_csem)〔TD〕
ER ercd = sac_sem(ID semid, const ACVCT *p_acvct)〔TPD〕
ER ercd = del_sem(ID semid)〔TD〕
ER ercd = sig_sem(ID semid)〔T〕
ER ercd = isig_sem(ID semid)〔I〕
ER ercd = wai_sem(ID semid)〔T〕
ER ercd = pol_sem(ID semid)〔T〕
ER ercd = twai_sem(ID semid, TMO tmout)〔T〕

ER ercd = ini_sem(ID semid)〔T〕
ER ercd = ref_sem(ID semid, T_RSEM *pk_rsem)〔T〕

イベントフラグ

ER_ID flgid = acre_flg(const T_CFLG *pk_cflg)〔TD〕
ER ercd = sac_flg(ID flgid, const ACVCT *p_acvct)〔TPD〕
ER ercd = del_flg(ID flgid)〔TD〕
ER ercd = set_flg(ID flgid, FLGPTN setptn)〔T〕
ER ercd = iset_flg(ID flgid, FLGPTN setptn)〔I〕
ER ercd = clr_flg(ID flgid, FLGPTN clrptn)〔T〕
ER ercd = wai_flg(ID flgid, FLGPTN waiptn, MODE wfmode, FLGPTN *p_flgptn)〔T〕
ER ercd = pol_flg(ID flgid, FLGPTN waiptn, MODE wfmode, FLGPTN *p_flgptn)〔T〕
ER ercd = twai_flg(ID flgid, FLGPTN waiptn, MODE wfmode, FLGPTN *p_flgptn, TMO tmout)〔T〕
ER ercd = ini_flg(ID flgid)〔T〕
ER ercd = ref_flg(ID flgid, T_RFLG *pk_rflg)〔T〕

データキュー

ER_ID dtqid = acre_dtq(const T_CDTQ *pk_cdtq)〔TD〕
ER ercd = sac_dtq(ID dtqid, const ACVCT *p_acvct)〔TPD〕
ER ercd = del_dtq(ID dtqid)〔TD〕
ER ercd = snd_dtq(ID dtqid, intptr_t data)〔T〕
ER ercd = psnd_dtq(ID dtqid, intptr_t data)〔T〕
ER ercd = ipsnd_dtq(ID dtqid, intptr_t data)〔I〕
ER ercd = tsnd_dtq(ID dtqid, intptr_t data, TMO tmout)〔T〕
ER ercd = fsnd_dtq(ID dtqid, intptr_t data)〔T〕
ER ercd = ifsnd_dtq(ID dtqid, intptr_t data)〔I〕
ER ercd = rcv_dtq(ID dtqid, intptr_t *p_data)〔T〕
ER ercd = prcv_dtq(ID dtqid, intptr_t *p_data)〔T〕
ER ercd = trcv_dtq(ID dtqid, intptr_t *p_data, TMO tmout)〔T〕
ER ercd = ini_dtq(ID dtqid)〔T〕
ER ercd = ref_dtq(ID dtqid, T_RDTQ *pk_rdtq)〔T〕

優先度データキュー

ER_ID pdqid = acre_pdq(const T_CPDQ *pk_cpdq)〔TD〕
ER ercd = sac_pdq(ID pdqid, const ACVCT *p_acvct)〔TPD〕
ER ercd = del_pdq(ID pdqid)〔TD〕
ER ercd = snd_pdq(ID pdqid, intptr_t data, PRI datapri)〔T〕
ER ercd = psnd_pdq(ID pdqid, intptr_t data, PRI datapri)〔T〕
ER ercd = ipsnd_pdq(ID pdqid, intptr_t data, PRI datapri)〔I〕
ER ercd = tsnd_pdq(ID pdqid, intptr_t data, PRI datapri, TMO tmout)〔T〕
ER ercd = rcv_pdq(ID pdqid, intptr_t *p_data, PRI *p_datapri)〔T〕
ER ercd = prcv_pdq(ID pdqid, intptr_t *p_data, PRI *p_datapri)〔T〕
ER ercd = trcv_pdq(ID pdqid, intptr_t *p_data, PRI *p_datapri, TMO tmout)〔T〕
ER ercd = ini_pdq(ID pdqid)〔T〕
ER ercd = ref_pdq(ID pdqid, T_RPDQ *pk_rpdq)〔T〕

メールボックス

ER_ID mbxid = acre_mbx(const T_CMBX *pk_cmbx)〔TDp〕
ER ercd = del_mbx(ID mbxid)〔TDp〕
ER ercd = snd_mbx(ID mbxid, T_MSG *pk_msg)〔Tp〕
ER ercd = rcv_mbx(ID mbxid, T_MSG **ppk_msg)〔Tp〕
ER ercd = prcv_mbx(ID mbxid, T_MSG **ppk_msg)〔Tp〕
ER ercd = trcv_mbx(ID mbxid, T_MSG **ppk_msg, TMO tmout)〔Tp〕
ER ercd = ini_mbx(ID mbxid)〔Tp〕
ER ercd = ref_mbx(ID mbxid, T_RMBX *pk_rmbx)〔Tp〕

ミューテックス

ER_ID mtxid = acre_mtx(const T_CMTX *pk_cmtx)〔TD〕
ER ercd = sac_mtx(ID mtxid, const ACVCT *p_acvct)〔TPD〕
ER ercd = del_mtx(ID mtxid)〔TD〕
ER ercd = loc_mtx(ID mtxid)〔T〕
ER ercd = ploc_mtx(ID mtxid)〔T〕
ER ercd = tloc_mtx(ID mtxid, TMO tmout)〔T〕
ER ercd = unl_mtx(ID mtxid)〔T〕
ER ercd = ini_mtx(ID mtxid)〔T〕
ER ercd = ref_mtx(ID mtxid, T_RMTX *pk_rmtx)〔T〕

メッセージバッファ

ER_ID mbfid = acre_mbf(const T_CMBF *pk_cmbf)〔TD〕
ER ercd = sac_mbf(ID mbfid, const ACVCT *p_acvct)〔TPD〕
ER ercd = del_mbf(ID mbfid)〔TD〕
ER ercd = snd_mbf(ID mbfid, const void *msg, uint_t msgsz)〔T〕
ER ercd = psnd_mbf(ID mbfid, const void *msg, uint_t msgsz)〔T〕
ER ercd = tsnd_mbf(ID mbfid, const void *msg, uint_t msgsz, TMO tmout)〔T〕
ER_UINT msgsz = rcv_mbf(ID mbfid, void *msg)〔T〕
ER_UINT msgsz = prcv_mbf(ID mbfid, void *msg)〔T〕
ER_UINT msgsz = trcv_mbf(ID mbfid, void *msg, TMO tmout)〔T〕
ER ercd = ini_mbf(ID mbfid)〔T〕
ER ercd = ref_mbf(ID mbfid, T_RMBF *pk_rmbf)〔T〕

スピンロック

ER_ID spnid = acre_spn(const T_CSPN *pk_cspn)〔TMD〕
ER ercd = sac_spn(ID spnid, const ACVCT *p_acvct)〔TPMD〕
ER ercd = del_spn(ID spnid)〔TMD〕
ER ercd = loc_spn(ID spnid)〔TM〕
ER ercd = iloc_spn(ID spnid)〔IM〕
ER ercd = try_spn(ID spnid)〔TM〕
ER ercd = itry_spn(ID spnid)〔IM〕
ER ercd = unl_spn(ID spnid)〔TM〕
ER ercd = iunl_spn(ID spnid)〔IM〕
ER ercd = ref_spn(ID spnid, T_RSPN *pk_rspn)〔TM〕

(5) メモリプール管理機能

固定長メモリプール

ER_ID mpfid = acre_mpf(const T_CMPF *pk_cmpf)〔TD〕
ER ercd = sac_mpf(ID mpfid, const ACVCT *p_acvct)〔TPD〕
ER ercd = del_mpf(ID mpfid)〔TD〕
ER ercd = get_mpf(ID mpfid, void **p_blk)〔T〕
ER ercd = pget_mpf(ID mpfid, void p_blk)〔T〕
ER ercd = tget_mpf(ID mpfid, void **p_blk, TMO tmout)〔T〕
ER ercd = rel_mpf(ID mpfid, void *blk)〔T〕
ER ercd = ini_mpf(ID mpfid)〔T〕
ER ercd = ref_mpf(ID mpfid, T_RMPF *pk_rmpf)〔T〕

(6) 時間管理機能

システム時刻管理

ER ercd = get_tim(SYSTIM *p_systim)〔T〕
ER ercd = get_utm(SYSUTM *p_sysutm)〔TI〕

周期ハンドラ

ER_ID cycid = acre_cyc(const T_CCYC *pk_ccyc)〔TD〕
ER ercd = sac_cyc(ID cycid, const ACVCT *p_acvct)〔TPD〕
ER ercd = del_cyc(ID cycid)〔TD〕
ER ercd = sta_cyc(ID cycid)〔T〕
ER ercd = msta_cyc(ID cycid, ID prcid)〔TM〕
ER ercd = stp_cyc(ID cycid)〔T〕
ER ercd = ref_cyc(ID cycid, T_RCYC *pk_rcyc)〔T〕

アラームハンドラ

ER_ID almid = acre_alm(const T_CALM *pk_calm)〔TD〕
ER ercd = sac_alm(ID almid, const ACVCT *p_acvct)〔TPD〕
ER ercd = del_alm(ID almid)〔TD〕
ER ercd = sta_alm(ID almid, RELTIM almtim)〔T〕
ER ercd = ista_alm(ID almid, RELTIM almtim)〔I〕
ER ercd = msta_alm(ID almid, RELTIM almtim, ID prcid)〔TM〕
ER ercd = imsta_alm(ID almid, RELTIM almtim, ID prcid)〔IM〕
ER ercd = stp_alm(ID almid)〔T〕
ER ercd = istp_alm(ID almid)〔I〕
ER ercd = ref_alm(ID almid, T_RALM *pk_ralm)〔T〕

オーバランハンドラ

ER ercd = def_ovr,(const T_DOVR *pk_dovr)〔TD〕
ER ercd = sta_ovr(ID tskid, OVRTIM ovrtim)〔T〕
ER ercd = ista_ovr(ID tskid, OVRTIM ovrtim)〔I〕
ER ercd = stp_ovr(ID tskid)〔T〕

ER ercd = istp_ovr(ID tskid) 〔I〕
ER ercd = ref_ovr(ID tskid, T_ROVR *pk_rovr) 〔T〕

(7) システム状態管理機能

ER ercd = sac_sys(const ACVCT *p_acvct) 〔TPD〕
ER ercd = rot_rdq(PRI tskpri) 〔T〕
ER ercd = irot_rdq(PRI tskpri) 〔I〕
ER ercd = mrot_rdq(PRI tskpri, ID prcid) 〔TM〕
ER ercd = imrot_rdq(PRI tskpri, ID prcid) 〔IM〕
ER ercd = get_tid(ID *p_tskid) 〔T〕
ER ercd = iget_tid(ID *p_tskid) 〔I〕
ER ercd = get_did(ID *p_domid) 〔TP〕
ER ercd = get_pid(ID *p_prcid) 〔TM〕
ER ercd = iget_pid(ID *p_prcid) 〔IM〕
ER ercd = loc_cpu() 〔T〕
ER ercd = iloc_cpu() 〔I〕
ER ercd = unl_cpu() 〔T〕
ER ercd = iunl_cpu() 〔I〕
ER ercd = dis_dsp() 〔T〕
ER ercd = ena_dsp() 〔T〕
bool_t state = sns_ctx() 〔TI〕
bool_t state = sns_loc() 〔TI〕
bool_t state = sns_dsp() 〔TI〕
bool_t state = sns_dpn() 〔TI〕
bool_t state = sns_ker() 〔TI〕
ER ercd = ext_ker() 〔TI〕
ER ercd = ref_sys(T_RSYS *pk_rsys) 〔T〕

(8) メモリオブジェクト管理機能

ER ercd = att_mem(const T_AMEM *pk_amem) 〔TPD〕
ER ercd = att_pma(const T_AMEM *pk_apma) 〔TPD〕
ER ercd = sac_mem(const void *base, const ACVCT *p_acvct) 〔TPD〕
ER ercd = det_mem(const void *base) 〔TPD〕
ER ercd = prb_mem(const void *base, SIZE size, ID tskid, MODE pmmode) 〔TP〕
ER ercd = ref_mem(const void *base, T_RMEM *pk_rmem) 〔TP〕

(9) 割込み管理機能

ER ercd = cfg_int(INTNO intno, const T_CINT *pk_cint) 〔TD〕
ER_ID isrid = acre_isr(const T_CISR *pk_cisr) 〔TD〕
ER ercd = sac_isr(ID isrid, const ACVCT *p_acvct) 〔TPD〕
ER ercd = del_isr(ID isrid) 〔TD〕
ER ercd = ref_isr(ID isrid, T_RISR *pk_risr) 〔T〕
ER ercd = def_inh(INHNO inhno, const T_DINH *pk_dinh) 〔TD〕
ER ercd = dis_int(INTNO intno) 〔T〕
ER ercd = ena_int(INTNO intno) 〔T〕
ER ercd = ref_int(INTNO intno, T_RINT *pk_rint) 〔T〕

ER ercd = chg_ipm(PRI intpri)〔T〕
ER ercd = get_ipm(PRI *p_intpri)〔T〕

(10) CPU例外管理機能

ER ercd = def_exc(EXCNO excno, const T_DEXC *pk_dexc)〔TD〕
bool_t stat = xsns_dpn(void *p_excinf)〔TI〕
bool_t stat = xsns_xpn(void *p_excinf)〔TI〕

(11) 拡張サービスコール管理機能

ER ercd = def_svc(FN fncd, const T_DSVC *pk_dsvc)〔TPD〕
ER_UINT ercd = cal_svc(FN fncd, intptr_t par1, intptr_t par2,intptr_t par3, intptr_t par4, intptr_t par5)〔TIP〕

(12) システム構成管理機能

ER ercd = ref_cfg(T_RCFG *pk_rcfg)〔T〕
ER ercd = ref_ver(T_RVER *pk_rver)〔T〕

5.2. 静的API一覧

(1) タスク管理機能

＊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 })〔S〕

＊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 })〔SP〕※ sstkszおよびsstkの記述は省略することができる.

AID_TSK(uint_t notsk)〔SD〕
SAC_TSK(ID tskid, { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })〔SP〕
DEF_EPR(ID tskid, { PRI exepri })〔S〕

(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 })〔S〕

(4) 同期・通信機能

セマフォ

CRE_SEM(ID semid, { ATR sematr, uint_t isemcnt, uint_t maxsem })〔S〕
AID_SEM(uint_t nosem)〔SD〕
SAC_SEM(ID semid, { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })〔SP〕

イベントフラグ

CRE_FLG(ID flgid, { ATR flgatr, FLGPTN iflgptn }) 〔S〕
AID_FLG(uint_t noflg) 〔SD〕
SAC_FLG(ID flgid, { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 }) 〔SP〕

データキュー

CRE_DTQ(ID dtqid, { ATR dtqatr, uint_t dtqcnt, void *dtqmb }) 〔S〕
AID_DTQ(uint_t nodtq) 〔SD〕
SAC_DTQ(ID dtqid, { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 }) 〔SP〕

優先度データキュー

CRE_PDQ(ID pdqid, { ATR pdqatr, uint_t pdqcnt, PRI maxdpri, void *pdqmb }) 〔S〕
AID_PDQ(uint_t nopdq) 〔SD〕
SAC_PDQ(ID pdqid, { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 }) 〔SP〕

メールボックス

CRE_MBX(ID mbxid, { ATR mbxatr, PRI maxmpri, void *mprihd }) 〔Sp〕
AID_MBX(uint_t nombx) 〔SpD〕

ミューテックス

CRE_MTX(ID mtxid, { ATR mtxatr, PRI ceilpri }) 〔S〕
AID_MTX(uint_t nomtx) 〔SD〕
SAC_MTX(ID mtxid, { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 }) 〔SP〕

メッセージバッファ

CRE_MBF(ID mbfid, { ATR mbfatr, uint_t maxmsz, SIZE mbfsz, void *mbfmb }) 〔S〕
AID_MBF(uint_t nombf) 〔SD〕
SAC_MBF(ID mbfid, { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 }) 〔SP〕

スピンロック

CRE_SPN(ID spnid, { ATR spnatr }) 〔SM〕
AID_SPN(uint_t nospn) 〔SMD〕
SAC_SPN(ID spnid, { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 }) 〔SPM〕

(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 }) 〔S〕
AID_MPF(uint_t nompf) 〔SD〕
SAC_MPF(ID mpfid, { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 }) 〔SP〕

(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 }) 〔S〕
AID_CYC(uint_t nocyc) 〔SD〕
SAC_CYC(ID cycid, { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 }) 〔SP〕

アラームハンドラ

CRE_ALM(ID almid, { ATR almatr, intptr_t exinf, ALMHDR almhdr }) 〔S〕
AID_ALM(uint_t noalm) 〔SD〕
SAC_ALM(ID almid, { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 }) 〔SP〕

オーバランハンドラ

DEF_OVR({ ATR ovratr, OVRHDR ovrhdr }) 〔S〕

(7) システム状態管理機能

SAC_SYS({ ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 }) 〔SP〕

(8) メモリオブジェクト管理機能

ATT_REG("メモリリージョン名", { ATR regatr, void *base, SIZE size }) 〔SP〕
DEF_SRG("標準ROMリージョン名", "標準RAMリージョン名") 〔SP〕
ATT_SEC("セクション名", { ATR rematr, "メモリリージョン名" }) 〔SP〕
ATA_SEC("セクション名", { ATR rematr, "メモリリージョン名" }, { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 }) 〔SP〕
LNK_SEC("セクション名", { "メモリリージョン名" }) 〔SP〕
ATT_MOD("オブジェクトモジュール名") 〔SP〕
ATA_MOD("オブジェクトモジュール名", { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 }) 〔SP〕
ATT_MEM({ ATR rematr, void *base, SIZE size }) 〔SP〕
ATA_MEM({ ATR rematr, void *base, SIZE size }, { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 }) 〔SP〕
ATA_PMA({ ATR rematr, void *base, SIZE size, void *paddr }) 〔SP〕
ATA_PMA({ ATR rematr, void *base, SIZE size, void *paddr }, { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 }) 〔SP〕

(9) 割込み管理機能

CFG_INT(INTNO intno, { ATR intatr, PRI intpri }) 〔S〕
CRE_ISR(ID isrid, { ATR isratr, intptr_t exinf, INTNO intno, ISR isr, PRI isrpri }) 〔S〕
ATT_ISR({ ATR isratr, intptr_t exinf, INTNO intno, ISR isr, PRI isrpri }) 〔S〕
AID_ISR(uint_t noisr) 〔SD〕
SAC_ISR(ID isrid, { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 }) 〔SP〕
DEF_INH(INHNO inhno, { ATR inhatr, INTHDR inthdr }) 〔S〕

(10) CPU例外管理機能

DEF_EXC(EXCNO excno, { ATR excatr, EXCHDR exchdr }) 〔S〕

(11) 拡張サービスコール管理機能

DEF_SVC(FN fncd, { ATR svcatr, EXTSVC svcrtn, SIZE stksz }) 〔SP〕

(12) システム構成管理機能

LMT_DOM({ PRI mintpri }) 〔SP〕 DEF_ICS({ SIZE istksz, STK_T *istk }) 〔S〕 DEF_STK({ SIZE stksz, STK_T *stk }) 〔S〕 ATT_INI({ ATR iniatr, intptr_t exinf, INIRTN inirtn }) 〔S〕 ATT_TER({ ATR teratr, intptr_t exinf, TERRTN terrtn }) 〔S〕

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）
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）

MB_T	オブジェクト管理領域を確保するためのデータ型

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）
OVRTIM	プロセッサ時間（符号無し整数，単位はマイクロ秒，ulong_tに定義）
INTNO	割込み番号（符号無し整数，uint_tに定義）
INHNO	割込みハンドラ番号（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）
OVRHDR	オーバランハンドラ（関数ポインタ）
ISR	割込みサービスルーチン（関数ポインタ）
INTHDR	割込みハンドラ（関数ポインタ）
EXCHDR	CPU例外ハンドラ（関数ポインタ）
EXTSVC	拡張サービスコール（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

メールボックスのメッセージヘッダ〔NGKI4001〕

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

メールボックスの優先度付きメッセージヘッダ〔NGKI4002〕

```
typedef struct t_msg_pri {
    T_MSG       msgque;         /* メールボックスのメッセージヘッダ */
    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〔NGKI4003〕

```
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 [NGKI4004]

```
typedef struct t_rtsk {
    STAT        tskstat;    /* タスク状態 */
    PRI         tskpri;     /* タスクの現在優先度 */
    PRI         tskbpri;    /* タスクのベース優先度 */
    STAT        tskwait;    /* 待ち要因 */
    ID          wobjid;     /* 待ち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 */
    /* 以下は，保護機能対応カーネルの場合 */
    bool_t      texmsk;     /* タスク例外マスク状態か否か */
    bool_t      waifbd;     /* 待ち禁止状態か否か */
    uint_t      svclevel;   /* 拡張サービスコールのネストレベル */
    /* 以下は，マルチプロセッサ対応カーネルの場合 */
    ID          prcid;      /* 割付けプロセッサのID */
    ID          actprc      /* 次の起動時の割付けプロセッサのID */
} 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ケット形式 [NGKI4005]

```
typedef struct t_dtex {
    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ケット形式 [NGKI4006]

```
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ケット形式 [NGKI4007]

```
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ケット形式 [NGKI4008]

```
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ケット形式 [NGKI4009]

```
typedef struct t_cflg {
    ATR         flgatr;     /* イベントフラグ属性 */
    FLGPTN      iflgptn;    /* イベントフラグの初期ビットパターン */
} T_CFLG;
```

イベントフラグの現在状態のパケット形式 [NGKI4010]

```
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ケット形式 [NGKI4011]

```
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 [NGKI4012]

```
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 [NGKI4013]

```
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 [NGKI4014]

```
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 [NGKI4015]

```
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 [NGKI4016]

```
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 [NGKI4017]

```
typedef struct t_cmtx {
    ATR         mtxatr;     /* ミューテックス属性 */
    PRI         ceilpri;    /* ミューテックスの上限優先度 */
} T_CMTX;
```

ミューテックスの現在状態のパケット形式 [NGKI4018]

```
typedef struct t_rmtx {
    ID          htskid;     /* ミューテックスをロックしているタス
                               クのID番号 */
    ID          wtskid;     /* ミューテックスの待ち行列の先頭のタ
                               スクのID番号 */
} T_RMTX;
```

メッセージバッファの生成情報のパケット形式 [NGKI4037]

```
typedef struct t_cmbf {
    ATR         mbfatr;     /* メッセージバッファ属性 */
    uint_t      maxmsz;     /* メッセージバッファの最大メッセージ
                               サイズ（バイト数）*/
    SIZE        mbfsz;      /* メッセージバッファ管理領域のサイズ
                               （バイト数）*/
    void *      mbfmb;      /* メッセージバッファ管理領域の先頭番地 */
} T_CMBF;
```

メッセージバッファの現在状態のパケット形式 [NGKI4038]

```
typedef struct t_rmbf {
    ID      stskid;     /* メッセージバッファの送信待ち行列の先頭の
                           タスクのID番号 */
    ID      rtskid;     /* メッセージバッファの受信待ち行列の先頭の
                           タスクのID番号 */
    uint_t  smbfcnt;    /* メッセージバッファ管理領域に格納されてい
                           るメッセージの数 */
    SIZE    fmbfsz;     /* メッセージバッファ管理領域中の空き領域の
                           サイズ */
} T_RMBF;
```

スピンロックの生成情報のパケット形式 [NGKI4019]

```
typedef struct t_cspn {
    ATR         spnatr;     /* スピンロック属性 */
} T_CSPN;
```

スピンロックの現在状態のパケット形式 [NGKI4020]

```
typedef struct t_rspn {
    STAT        spnstat     /* スピンロックのロック状態 */
} T_RSPN;
```

(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 [NGKI4021]

```
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      blksz;      /* 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 [NGKI4022]

```
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 [NGKI4023]

```
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      cychdr;     /* 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 */
} T_CCYC;
```

周期ハンドラの現在状態のパケット形式 [NGKI4024]

```
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 */
    /* 以下は，マルチプロセッサ対応カーネルの場合 */
    ID          prcid;      /* 割付けプロセッサのID */
} T_RCYC;
```

アラームハンドラの生成情報のパケット形式 [NGKI4025]

```
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;     /* アラームハンドラの先頭番地 */
} T_CALM;
```

アラームハンドラの現在状態のパケット形式 [NGKI4026]

```
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 */
    /* 以下は，マルチプロセッサ対応カーネルの場合 */
    ID          prcid;      /* 割付けプロセッサのID */
} T_RALM;
```

オーバランハンドラの定義情報のパケット形式 [NGKI4027]

```
typedef struct t_dovr {
    ATR         ovratr;     /* オーバランハンドラ属性 */
    OVRHDR      ovrhdr;     /* オーバランハンドラの先頭番地 */
} T_DOVR;
```

オーバランハンドラの現在状態のパケット形式 [NGKI4028]

```
typedef struct t_rovr {
    STAT    ovrstat;        /* オーバランハンドラの動作状態 */
    OVRTIM  leftotm;        /* 残りプロセッサ時間 */
} T_ROVR;
```

(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

(8) メモリオブジェクト管理機能

メモリオブジェクトの登録情報のパケット形式 [NGKI4029]

```
typedef struct t_amem {
    ATR         mematr      /* メモリオブジェクト属性 */
    void *      base        /* 登録するメモリ領域の先頭番地 */
    SIZE        size        /* 登録するメモリ領域のサイズ（バイト数）*/
} T_AMEM;
```

物理メモリ領域の登録情報のパケット形式 [NGKI4030]

```
typedef struct t_apma {
    ATR         mematr      /* メモリオブジェクト属性 */
    void *      base        /* 登録するメモリ領域の先頭番地 */
    SIZE        size        /* 登録するメモリ領域のサイズ（バイト数）*/
    void *      paddr       /* 登録するメモリ領域の物理アドレスの先頭
                               番地 */
} T_APMA;
```

メモリオブジェクト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 [NGKI4031]

```
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 [NGKI4032]

```
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 [NGKI4033]

```
typedef struct t_dinh {
    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 */
} T_DINH;
```

割込み要求ラインの現在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 [NGKI4034]

```
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 [NGKI4035]

```
typedef struct t_dsvc {
    ATR         svcatr      /* 拡張サービスコール属性 */
    EXTSVC      svcrtn      /* 拡張サービスコールの先頭番地 */
    SIZE        stksz       /* 拡張サービスコールで使用するスタック
                               サイズ */
} T_DSVC;
```

(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	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      カーネルドメインのみにアクセスを許可
TACP_SHARED     ~0U     すべての保護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)      番地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のみ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

TCLS_SELF	0	自タスクの属するクラス

TPRC_NONE	0	割付けプロセッサの指定がない
TPRC_INI	0	初期割付けプロセッサ

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. カーネル共通マクロ

(1) オブジェクト属性を作るマクロ

TA_DOM(domid)	domidで指定される保護ドメインに属する
TA_CLS(clsid)	clsidで指定されるクラスに属する

(2) サービスコールの呼出し方法を指定するマクロ

SVC_CALL(svc)	svcで指定されるサービスコールを関数呼出しによって呼び出すための名称

5.4.5. カーネルの機能毎の定数

(1) タスク管理機能

TA_ACT	0x02U	タスクの生成時にタスクを起動する
TA_RSTR	0x04U	生成するタスクを制約タスクとする
TA_FPU		FPUレジスタをコンテキストに含める

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

```
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_MBX         0x0040U メールボックスからの受信待ち
TTW_MTX         0x0080U ミューテックスのロック待ち状態
TTW_SMBF        0x0400U メッセージバッファへの送信待ち
TTW_RMBF        0x0800U メッセージバッファからの受信待ち
TTW_MPF         0x2000U 固定長メモリブロックの獲得待ち
```

TA_FPUの値は，ターゲット定義とする．

(3) タスク例外処理機能

```
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
```

(4) 同期・通信機能

イベントフラグ

```
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
```

```
TWF_ORW         0x01U   イベントフラグのOR待ちモード
TWF_ANDW        0x02U   イベントフラグのAND待ちモード
```

メールボックス

```
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
```

スピンロック

```
TSPN_UNL        0x01U   取得されていない状態
TSPN_LOC        0x02U   取得されている状態
```

(6) 時間管理機能

周期ハンドラ

TA_STA	0x02U	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	0x04U	周期ハンドラを生成した時刻を基準時刻とする

TCYC_STP	0x01U	周期ハンドラが動作していない状態
TCYC_STA	0x02U	周期ハンドラが動作している状態

アラームハンドラ

TALM_STP	0x01U	アラームハンドラが動作していない状態
TALM_STA	0x02U	アラームハンドラが動作している状態

オーバランハンドラ

TOVR_STP	0x01U	オーバランハンドラが動作していない状態
TOVR_STA	0x02U	オーバランハンドラが動作している状態

(8) メモリオブジェクト管理機能

TA_NOWRITE	0x01U	書込みアクセス禁止
TA_NOREAD	0x02U	読出しアクセス禁止
TA_EXEC	0x04U	実行アクセス許可
TA_MEMINI	0x08U	メモリの初期化を行う
TA_MEMPRSV	0x10U	メモリの初期化を行わない
TA_SDATA	0x20U	ショートデータ領域に配置
TA_UNCACHE	0x40U	キャッシュ禁止
TA_IODEV	0x80U	周辺デバイスの領域
TA_WTHROUGH		ライトスルーキャッシュを用いる

TPM_WRITE	0x01U	書込みアクセス権のチェック
TPM_READ	0x02U	読出しアクセス権のチェック
TPM_EXEC	0x04U	実行アクセス権のチェック

TA_WTHROUGHの値は，ターゲット定義とする．

(9) 割込み管理機能

```
TA_ENAINT       0x01U   割込み要求禁止フラグをクリア
TA_EDGE         0x02U   エッジトリガ
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_HIGHLEVEL            ハイレベルトリガ
```

```
TA_NONKERNEL    0x02U   カーネル管理外の割込み
```

TA_POSEDGE，TA_NEGEDGE，TA_BOTHEDGE，TA_LOWLEVEL，TA_HIGHLEVELの値は，ターゲット定義とする．

(10) CPU例外管理機能

```
TA_DIRECT               CPU例外ハンドラを直接呼び出す
```

TA_DIRECTの値は，ターゲット定義とする．

5.4.6. 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
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）
```

(4) 同期・通信機能

```
TSZ_DTQMB(dtqcnt)   dtqcntで指定した数のデータを格納できるデータ
                    キュー管理領域のサイズ（バイト数）
TCNT_DTQMB(dtqcnt)  dtqcntで指定した数のデータを格納できるデータ
                    キュー管理領域を確保するために必要なMB_T型の配
                    列の要素数
```

```
TSZ_PDQMB(pdqcnt)   pdqcntで指定した数のデータを格納できる優先度デー
                    タキュー管理領域のサイズ（バイト数）
TCNT_PDQMB(pdqcnt)  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確保するために必要なMB_T型の
                    配列の要素数
```

```
TSZ_MBFMB(msgcnt, msgsz)    msgszのメッセージ領域サイズ
                            msgcntのメッセージ数を格納
                            するのに必要なサイズ
TCNT_MBFMB(msgcnt, msgsz)   msgszのメッセージ領域サイズ
                            msgcntのメッセージ数を格納
                            するのに必要な
                            MB_Tの個数
```

(5) メモリプール

```
COUNT_MPF_T(blksz)  メモリプールの領域にblkszのブロック
                    を格納するのに必要な
                    1ブロックのサイズMPF_Tの個数
                    を返す
ROUND_MPF_T(blksz)  COUNT_MPF_T(blksz)のMPF_Tの個数を
                    blkszにMPF_Tの個数を加えた
                    値を返す
```

```
TSZ_MPFMB(blkcnt)   blkcntのブロックを格納するのに必要な
                    メモリプール領域サイズ
                    を返す
TCNT_MPFMB(blkcnt)  blkcntのブロックを格納するのに必要な
                    メモリプール領域サイズを
                    MB_Tの個数で
```

5.5. その他

5.5.1. TOPPERSのバージョン

(1) バージョン

```
TMAX_RELTIM      相対時間の最大値
```

5.5.2. サポートされるオプション

(1) オプション

```
TOPPERS_SUPPORT_PROTECT          保護機能
TOPPERS_SUPPORT_MULTI_PRC        マルチプロセッサ
TOPPERS_SUPPORT_DYNAMIC_CRE      動的生成
```

(2) 拡張機能

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

(3) プロセッサの数

TNUM_PRCID	プロセッサの数

(4) 特殊な役割を持ったプロセッサ

TOPPERS_MASTER_PRCID	マスタプロセッサのID番号
TOPPERS_SYSTIM_PRCID	システム時刻管理プロセッサのID番号

(5) タイマ方式

TOPPERS_SYSTIM_LOCAL	ローカルタイマ方式の場合にマクロ定義
TOPPERS_SYSTIM_GLOBAL	グローバルタイマ方式の場合にマクロ定義

(6) 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

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）

メッセージバッファ

TNUM_MBFID	登録できるメッセージバッファの数（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ッセージバッファの数に一致）

スピンロック

TNUM_SPNID	登録できるスピンロックの数（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）

オーバランハンドラ

TMAX_OVRTIM	プロセッサ時間に指定できる最大値

TOPPERS_SUPPORT_OVRHDR	オーバランハンドラ機能がサポートされて いる

(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

TOPPERS_SUPPORT_ATT_MOD	ATT_MOD/ATA_MODがサポートされている
TOPPERS_SUPPORT_ATT_PMA	ATT_PMA/ATA_PMA/att_pmaがサポートさ れている

(9) 割込み管理機能

TMIN_INTPRI	割込み優先度の最小値（最高値）
TMAX_INTPRI	割込み優先度の最大値（最低値，=-1）

TMIN_ISRPRI	割込みサービスルーチン優先度の最小値（=1）
TMAX_ISRPRI	割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_NORES	-35	資源不足
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	バッファオーバフロー

5.7. 機能コード一覧【NGKI4036】

	-0	-1	-2	-3
-0x01	予約	予約	予約	予約
-0x05	act_tsk	iact_tsk	can_act	ext_tsk
-0x09	ter_tsk	chg_pri	get_pri	get_inf
-0x0d	slp_tsk	tslp_tsk	wup_tsk	iwup_tsk
-0x11	can_wup	rel_wai	irel_wai	予約

-0x15	dis_wai	idis_wai	ena_wai	iena_wai
-0x19	sus_tsk	rsm_tsk	dly_tsk	□□
-0x1d	ras_tex	iras_tex	dis_tex	ena_tex
-0x21	sns_tex	ref_tex	□□	□□
-0x25	sig_sem	isig_sem	wai_sem	pol_sem
-0x29	twai_sem	□□	□□	□□
-0x2d	set_flg	iset_flg	clr_flg	wai_flg
-0x31	pol_flg	twai_flg	□□	□□
-0x35	snd_dtq	psnd_dtq	ipsnd_dtq	tsnd_dtq
-0x39	fsnd_dtq	ifsnd_dtq	rcv_dtq	prcv_dtq
-0x3d	trcv_dtq	□□	□□	□□
-0x41	snd_pdq	psnd_pdq	ipsnd_pdq	tsnd_pdq
-0x45	rcv_pdq	prcv_pdq	trcv_pdq	□□
-0x49	snd_mbx	rcv_mbx	prcv_mbx	trcv_mbx
-0x4d	loc_mtx	ploc_mtx	tloc_mtx	unl_mtx
-0x51	snd_mbf	psnd_mbf	tsnd_mbf	rcv_mbf
-0x55	prcv_mbf	trcv_mbf	□□	□□
-0x59	get_mpf	pget_mpf	tget_mpf	rel_mpf
-0x5d	get_tim	get_utm	□□	ref_ovr
-0x61	sta_cyc	stp_cyc	□□	□□
-0x65	sta_alm	ista_alm	stp_alm	istp_alm
-0x69	sta_ovr	ista_ovr	stp_ovr	istp_ovr
-0x6d	sac_sys	ref_sys	rot_rdq	irot_rdq
-0x71	get_did	□□	get_tid	iget_tid
-0x75	loc_cpu	iloc_cpu	unl_cpu	iunl_cpu
-0x79	dis_dsp	ena_dsp	sns_ctx	sns_loc
-0x7d	sns_dsp	sns_dpn	sns_ker	ext_ker
-0x81	att_mem	det_mem	sac_mem	prb_mem
-0x85	ref_mem	□□	att_pma	□□
-0x89	cfg_int	dis_int	ena_int	ref_int
-0x8d	chg_ipm	get_ipm	□□	□□
-0x91	xsns_dpn	xsns_xpn	□□	□□
-0x95	ref_cfg	ref_ver	□□	□□
-0x99	□□	□□	□□	□□

-0x9d	□□	□□	□□	□□
-0xa1	□□	ini_sem	ini_flg	ini_dtq
-0xa5	ini_pdq	ini_mbx	ini_mtx	ini_mbf
-0xa9	ini_mpf	□□	□□	□□
-0xad	□□	□□	□□	□□
-0xb1	ref_tsk	ref_sem	ref_flg	ref_dtq
-0xb5	ref_pdq	ref_mbx	ref_mtx	ref_mbf
-0xb9	ref_mpf	ref_cyc	ref_alm	ref_isr
-0xbd	ref_spn	□□	□□	□□
-0xc1	acre_tsk	acre_sem	acre_flg	acre_dtq
-0xc5	acre_pdq	acre_mbx	acre_mtx	acre_mbf
-0xc9	acre_mpf	acre_cyc	acre_alm	acre_isr
-0xcd	acre_spn	□□	□□	□□
-0xd1	del_tsk	del_sem	del_flg	del_dtq
-0xd5	del_pdq	del_mbx	del_mtx	del_mbf
-0xd9	del_mpf	del_cyc	del_alm	del_isr
-0xdd	del_spn	□□	□□	□□
-0xe1	sac_tsk	sac_sem	sac_flg	sac_dtq
-0xe5	sac_pdq	□□	sac_mtx	sac_mbf
-0xe9	sac_mpf	sac_cyc	sac_alm	sac_isr
-0xed	sac_spn	□□	□□	□□
-0xf1	def_tex	def_ovr	def_inh	def_exc
-0xf5	def_svc	□□	□□	□□
-0xf9	□□	□□	□□	□□
-0xfd	□□	□□	□□	□□
-0x101	mact_tsk	imact_tsk	mig_tsk	□□
-0x105	msta_cyc	□□	msta_alm	imsta_alm
-0x109	mrot_rdq	imrot_rdq	get_pid	iget_pid
-0x10d	□□	□□	□□	□□
-0x111	loc_spn	iloc_spn	try_spn	itry_spn
-0x115	unl_spn	iunl_spn	□□	□□
-0x119	□□	□□	□□	□□
-0x11d	□□	□□	□□	□□

□μITRON4.0□□□□□□□

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	ref_mem
		実行	sac_mem	prb_mem
タスク	act_tsk	ter_tsk	del_tsk	get_pri
	mact_tsk	chg_pri	sac_tsk	ref_tsk
	can_act	rel_wai	def_tex	ref_tex
	mig_tsk	sus_tsk		ref_ovr
	wup_tsk	rsm_tsk		prb_mem
	can_wup	dis_wai		
		ena_wai		
		ras_tex		
		sta_ovr		
		stp_ovr		
セマフォ	sig_sem	wai_sem	del_sem	ref_sem
		pol_sem	ini_sem	
		twai_sem	sac_sem	
イベントフラグ	set_flg	wai_flg	del_flg	ref_flg
	clr_flg	pol_flg	ini_flg	
		twai_flg	sac_flg	
データキュー	snd_dtq	rcv_dtq	del_dtq	ref_dtq
	psnd_dtq	prcv_dtq	ini_dtq	
	tsnd_dtq	trcv_dtq	sac_dtq	
	fsnd_dtq			
優先度データキュー	snd_pdq	rcv_pdq	del_pdq	ref_pdq
	psnd_pdq	prcv_pdq	ini_pdq	
	tsnd_pdq	trcv_pdq	sac_pdq	
メッセージバッファ	snd_mbf	rcv_mbf	del_mbf	ref_mbf
	psnd_mbf	prcv_mbf	ini_mbf	
	tsnd_mbf	trcv_mbf	sac_mbf	

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
ミューテックス	loc_mtx		del_mtx	ref_mtx
	ploc_mtx		ini_mtx	
	tloc_mtx		sac_mtx	
	unl_mtx			
スピンロック	loc_spn		del_spn	ref_spn
	try_spn		sac_spn	
	unl_spn			
固定長メモリプール	get_mpf	rel_mpf	del_mpf	ref_mpf
	pget_mpf		ini_mpf	
	tget_mpf		sac_mpf	
周期ハンドラ	sta_cyc	stp_cyc	del_cyc	ref_cyc
	msta_cyc		sac_cyc	
アラームハンドラ	sta_alm	stp_alm	del_alm	ref_alm
	msta_alm		sac_alm	
割込みサービスルーチン			del_isr	ref_isr
			sac_isr	
システム状態	rot_rdq	loc_cpu	acre_yyy	get_tim
	mrot_rdq	unl_cpu	att_mem	get_ipm
	dis_dsp	dis_int	att_pma	ref_sys
	ena_dsp	ena_int	cfg_int	ref_int
		chg_ipm	def_inh	ref_cfg
			def_exc	ref_ver
	def_svc		def_ovr	

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)
- 性能評価用システム時刻の参照 (get_utm)
- システム状態参照 (get_tid, get_did, get_pid, sns_ctx, sns_loc, sns_dsp, sns_dpn, sns_ker)
- CPU例外発生時の状態参照 (xsns_dpn, xsns_xpn)
- 拡張サービスコールの呼出し (cal_svc)

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:

- システム状態のアクセス許可ベクタの設定 (sac_sys)

- カーネルの終了（ext_ker）
- 非タスクコンテキスト専用のサービスコール

【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，タスクコンテキスト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．

【μITRON4.0/PX仕様との関係】

get_priは，μITRON4.0/PX仕様ではタスクに対する通常操作1としていたのを，タスクに対する参照操作に変更した．また，get_ipm（μITRON4.0/PX仕様ではget_ixx）をシステム状態に対する通常操作2から参照操作に，sac_sysをシステム状態に対する管理操作から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に変更した．システム時刻に対するアクセス許可ベクタは廃止し，get_timはシステム状態に対する参照操作とした．

【仕様変更の経緯】

この仕様のRelease 1.5以前では，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た．これは，ミューテックスをロックしたタスク以外がunl_mtxを呼び出すと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と考えたためである．しかし，タスクが拡張サービスコールの中でミューテックスをロックした場合，アクセス許可ベクタではアクセスが許可されていないミューテックスをロックすることができる．このようなミューテックスのロック解除は，タスクから直接unl_mtxを呼んで行うのではなく，拡張サービスコールの中で行うべきと考えられる．そこで，unl_mtxを，対象ミューテックスの通常操作1としてアクセス保護する仕様に変更した．なお，HRP2カーネルRelease 2.1以前のバージョンは，古い仕様に従って実装されている．

5.9. ターゲット定義事項一覧

・割込み優先度の段階数［NGKI0256］

・割込み番号の付与方法［NGKI0272］

・割込みハンドラ番号の付与方法［NGKI0273］

・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か［NGKI0276］

・受け付けた割込み要求に対して，割込みサービスルーチンも割込みハンドラも登録していない場合の振舞い［NGKI0249］

・割込み要求禁止フラグがサポートされているか［NGKI0260］［NGKI0261］

・割込み要求禁止フラグの振舞いを仕様と異なるものとするか［NGKI0261］

・割込み要求ラインのトリガモードの設定がサポートされているか［NGKI0267］

・割込み要求ラインをエッジトリガに設定する場合に，ポジティブエッジトリガかネガティブエッジトリガか両エッジトリガかを設定できるか［NGKI0265］

・割込み要求ラインをレベルトリガに設定する場合に，ローレベルトリガかハイレベルトリガかを設定できるか［NGKI0266］

・あるプロセッサで割込み要求禁止フラグを動的にセット／クリアしても，他のプロセッサに対しては割込みがマスク／マスク解除されないものとするか（M）［NGKI0281］

・TMIN_INTPRIを固定するか設定できるようにするかと，設定できるようにする場合の設定方法［NGKI0288］

・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）［NGKI0289］

□□□

・タスクのシステムスタック領域の先頭番地とサイズに対する制約〔P〕□［NGKI1062］［NGKI1065］［NGKI1070］

・データキュー管理領域の先頭番地に対する制約［NGKI1687］

・優先度データキュー管理領域の先頭番地に対する制約［NGKI1824］

・メッセージバッファ管理領域の先頭番地とサイズに対する制約［NGKI3319］□［NGKI3324］

・生成できるスピンロックの数の上限〔M〕［NGKI2142］

・スピンロックに対して，複数のプロセッサがロックの取得を待っている時に，□どのプロセッサが最初にロックを取得できるか〔M〕［NGKI2183］

・固定長メモリプール領域の先頭番地に対する制約［NGKI2249］

・固定長メモリプール管理領域の先頭番地に対する制約［NGKI2256］

・タイムティックの周期［NGKI2335］

・マルチプロセッサ対応カーネルにおける性能評価用システム時刻の扱い〔M〕□［NGKI2346］

・get_utmがサポートされているか［NGKI2360］

・オーバランハンドラ機能がサポートされているか［NGKI2598］

・オーバランハンドラ機能のプロセッサ時間に指定できる値の上限［NGKI2594］

・ターゲット定義のメモリリージョン属性〔P〕

・メモリリージョンの先頭番地とサイズに対する制約〔P〕［NGKI2768］

・メモリオブジェクトに対するTA_NOWRITE属性，TA_NOREAD属性，TA_EXEC属性
□の内，どのような場合にどの属性の指定が無視されるか〔P〕［NGKI2782］

・ショートデータ領域がサポートされておらず，TA_SDATA属性が無視される□〔P〕［NGKI2789］

・TA_NOWRITEを指定した場合に，TA_SDATAが無視されるか〔P〕［NGKI2790］

・TA_UNCACHE属性やTA_IODEV属性を指定しても意味がなく，これらの属性が無□視されるか〔P〕［NGKI2792］

・キャッシュ禁止にできないメモリオブジェクトと周辺デバイスの領域として□扱うことができないメモリオブジェクト〔P〕［NGKI2793］

・ターゲット定義のメモリオブジェクト属性〔P〕［NGKI2794］

・ATA_SECにより登録できるセクションが属する保護ドメインや登録できる数□に対する制限〔P〕［NGKI2831］

・ATT_MOD／ATA_MODがサポートされているか〔P〕［NGKI2859］

・ATT_MOD／ATA_MODにより登録されるセクション毎のメモリオブジェクトに設□定されるメモリオブジェクト属性〔P〕［NGKI2850］

・クラスの囲みの中に記述されたATT_MOD／ATA_MODにおいて，クラスの標準メ□モリリージョンが定義されている場合でも，共通の標準メモリリージョンに□配置されるセクション〔PM〕［NGKI3271］

・ATA_MODにより登録できるオブジェクトモジュールが属する保護ドメインや登□録できる数に対する制限〔P〕［NGKI2857］

・ATT_MEM／ATA_MEMにより登録できるメモリオブジェクトが属する保護ドメインや登録できる数に対する制限〔P〕［NGKI2878］

・ATT_MEM／ATA_MEM／att_memにより登録するメモリ領域の先頭番地とサイズに対する制約〔P〕［NGKI2880］

・ATT_PMA／ATA_PMA／att_pmaがサポートされているか〔P〕［NGKI2903］［HRPS0156］

・ATT_PMA／ATA_PMAにより登録できるメモリオブジェクトが属する保護ドメインや登録できる数に対する制限〔P〕［NGKI2898］

・ATT_PMA／ATA_PMA／att_pmaにより登録するメモリ領域の先頭番地とサイズ，物理アドレス空間における先頭番地に対する制約〔P〕［NGKI2900］

・ターゲット定義の割込み要求ライン属性［NGKI2945］

・割込みハンドラ属性にTA_NONKERNELを指定できるか［NGKI2957］

・その他のターゲット定義の割込みハンドラ属性［NGKI2959］

・cfg_intにおいて，複数の割込み要求ラインの割込み優先度が連動して設定されるか〔D〕［NGKI2980］

・CFG_INT／cfg_intで，カーネル管理外の割込み要求ラインに対しても属性を設定できるか［NGKI2982］

・CFG_INT／cfg_intで，各割込み要求ラインに対して設定できる割込み要求ライン属性／割込み優先度に対する制限［NGKI2986］

・割込みサービスルーチンが属することができるクラスに対する制限〔M〕［NGKI3018］

・CRE_ISR／ATT_ISRにおいて，isrが不正である場合にE_PARエラーが検出されるか［NGKI3020］

・DEF_INH／def_inh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か［NGKI3064］

・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か［NGKI3067］

・カーネル管理に固定されている割込みハンドラがあるか［NGKI3068］

・割込みハンドラが属することができるクラスに対する制限〔M〕［NGKI3074］

・def_inhで，静的APIで定義された割込みハンドラの定義を解除できるか〔D〕［NGKI3077］

・DEF_INH／def_inhで割込みハンドラを定義（または定義解除）できない割込みハンドラ番号［NGKI3078］

・def_inhを呼び出したタスクが割り付けられているプロセッサから定義（または定義解除）できない割込みハンドラ〔M〕［NGKI3079］

・DEF_INHにおいて，inthdrが不正である場合にE_PARエラーが検出されるか［NGKI3080］

・dis_intがサポートされているか［NGKI3091］

・dis_intにより，どのような場合に割込み要求ラインの割込み要求禁止フラグをセットできないか［NGKI3087］

・dis_intにおいて，割込み要求禁止フラグの振舞いが，この仕様の規定と異なるか［NGKI3089］

・ena_intがサポートされているか［NGKI3104］

・ena_intにより，どのような場合に割込み要求ラインの割込み要求禁止フラグをクリアできないか［NGKI3100］

・ena_intにおいて，割込み要求禁止フラグの振舞いが，この仕様の規定と異なるか［NGKI3102］

・chg_ipmにより，割込み優先度マスクをTMIN_INTPRIよりも小さい値に変更できるか［NGKI3114］

・ターゲット定義のCPU例外ハンドラ属性［NGKI3123］

・def_excで，静的APIで定義されたCPU例外ハンドラの定義を解除できるか〔D〕［NGKI3148］

・DEF_EXCにおいて，exchdrが不正である場合にE_PARエラーが検出されるか［NGKI3149］

・非タスクコンテキスト用スタック領域のサイズの最小値［NGKI3254］

・非タスクコンテキスト用スタック領域の先頭番地とサイズに対する制約［NGKI3220］［NGKI3222］

・DEF_ICSにより非タスクコンテキスト用スタック領域を設定しない場合の，非タスクコンテキスト用スタック領域のデフォルトのサイズ［NGKI3224］

・共有スタック領域のサイズの最小値［NGKI3255］

・共有スタック領域の先頭番地とサイズに対する制約［NGKI3234］［NGKI3236］

・ATT_INIにおいて，inirtnが不正である場合にE_PARエラーが検出されるか［NGKI3246］

・ATT_TERにおいて，terrtnが不正である場合にE_PARエラーが検出されるか［NGKI3253］

5.10. 省略名の元になった英語

5.10.1. サービスコールと静的APIの名称の中のxxxの元になった英語

```
xxx     □□□□□□□
-----------------------
act     activate
aid     automatically assigned ID
ata     attach with access control vector
att     attach
cal     call
can     cancel
cfg     configure
chg     change
clr     clear
cre     create
def     define
del     delete
det     detach
dis     disable
dly     delay
ena     enable
epr     execution priority
ext     exit
get     get
ini     initialize
lmt     limit
lnk     link
loc     lock
mig     migrate
pol     poll
prb     probe
ras     raise
rcv     receive
ref     reference
rel     release
rot     rotate
rsm     resume
sac     set access control vector
set     set
sig     signal
slp     sleep
snd     send
sns     sense
sta     start
stp     stop
sus     suspend
ter     terminate
try     try
unl     unlock
wai     wait
wup     wake up
```

5.10.2. サービスコールと静的APIの名称の中のyyy の元になった英語

yyy	元になった英語
act	activation
alm	alarm handler
cfg	configuration
cpu	CPU
ctx	context
cyc	cyclic handler
did	domain ID
dom	domain
dpn	dispatch pending
dsp	dispatch
dtq	data queue
exc	exception
flg	eventflag
ics	interrupt context stack
inf	information
inh	interrupt handler
ini	initilization
int	interrupt
ipm	interrupt priority mask
isr	interrupt service routine
ker	kernel
loc	lock
mbf	message buffer
mbx	mailbox
mpf	fixed-sized memory pool
mem	memory
mod	module
mtx	mutex
ovr	overrun handler
pdq	priority data queue
pid	processor ID
pma	physical memory area
pri	priority
rdq	ready queue
reg	region
sec	section
sem	semaphore
srg	standard memory region
spn	spin lock
stk	stack
sys	system
svc	service call
ter	termination
tex	task exception
tid	task ID
tim	time

```
tsk     task
utm     time in micro second
ver     version
wai     wait
wup     wake up
xpn     exception pending
```

5.10.3. サービスコールの名称の中のzの元になった英語

```
z       元になった英語
----------------------
a       automatic ID assignment
f       force
i       interrupt
m       multiprocessor
p       poll
t       timeout
x       exception
```

5.11. バージョン履歴

```
2008年11月19日 Release 1.0.0       最初のリリース
2009年5月8日   Release 1.1.0       FMPカーネルに関する記述が完成
2010年5月10日  Release 1.2.0
2011年5月5日   Release 1.3.0       HRP2カーネルに関する記述が完成
2012年5月16日  Release 1.4.0       SSPカーネルに関する記述が完成
2012年12月19日 Release 1.5.0       HRP2カーネルの仕様変更を反映
2014年1月16日  Release 1.6.0
2014年11月17日 Release 1.7.0
2015年5月30日  Release 1.7.1
```

以上