# TOPPERS第3世代カーネル（ITRON系）統合仕様書

Release 3.6.0, 2023年3月11日

# 目次


仕様書で用いる記述項目と記号 . . . 2
タグの付与方法 . . . 4
第１章　TOPPERS第3世代カーネル（ITRON系）の概要 . . . 5
1.1 TOPPERS第3世代カーネル（ITRON系）仕様の位置付け . . . 5
1.2 TOPPERS第3世代カーネル（ITRON系）仕様の設計方針 . . . 5
1.3 TOPPERS/ASP3カーネルの適用対象領域と仕様設計方針 . . . 6
1.4 TOPPERS/FMP3カーネルの適用対象領域と仕様設計方針 . . . 7
1.5 TOPPERS/HRP3カーネルの適用対象領域と仕様設計方針 . . . 7
1.6 TOPPERS/HRMP3カーネルの適用対象領域と仕様設計方針 . . . 8
1.7 TOPPERS/SSP3カーネルの適用対象領域と仕様設計方針 . . . 8
1.8 その他のカーネル . . . 8
第２章　主要な概念と共通定義 . . . 9
2.1 仕様の位置付け . . . 9
2.1.1 カーネルの機能セット . . . 9
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 . . . 10
2.1.3 想定するソフトウェア構成 . . . 10
2.1.4 想定するハードウェア構成 . . . 11
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 . . . 13
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 . . . 16
2.3.3 保護機能 . . . 20
2.3.4 時間パーティショニング . . . 23
2.3.5 マルチプロセッサ対応 . . . 24
2.3.6 その他 . . . 26
2.4 処理単位の種類と実行順序 . . . 27
2.4.1 処理単位の種類 . . . 27
2.4.2 処理単位の実行順序 . . . 28
2.4.3 カーネル処理の不可分性 . . . 28
2.4.4 処理単位を実行するプロセッサ . . . 29
2.5 システム状態とコンテキスト . . . 30
2.5.1 カーネル動作状態と非動作状態 . . . 30
2.5.2 タスクコンテキストと非タスクコンテキスト . . . 30
2.5.3 カーネルの振舞いに影響を与える状態 . . . 31
2.5.4 全割込みロック状態と全割込みロック解除状態 . . . 31
2.5.5 CPUロック状態とCPUロック解除状態 . . . 31
2.5.6 割込み優先度マスク . . . 32
2.5.7 ディスパッチ禁止状態とディスパッチ許可状態 . . . 32
2.5.8 ディスパッチ保留状態 . . . 33
2.5.9 カーネル管理外の状態 . . . 33
2.5.10 処理単位の開始・終了とシステム状態 . . . 33
2.6 タスクの状態遷移とスケジューリング規則 . . . 35

2.6.1 基本的なタスク状態 . . . 35
2.6.2 タスクの状態遷移 . . . 37
2.6.3 タスクのスケジューリング規則 . . . 38
2.6.4 待ち行列と待ち解除の順序 . . . 39
2.6.5 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 . . . 39
2.6.6 制約タスク . . . 41
2.6.7 時間パーティショニング使用時のスケジューリング規則 . . . 42
2.6.8 スケジューリング単位 . . . 44
2.7 割込み処理モデル . . . 45
2.7.1 割込み処理の流れ . . . 46
2.7.2 割込み優先度 . . . 47
2.7.3 割込み要求ラインの属性 . . . 48
2.7.4 割込みを受け付ける条件 . . . 49
2.7.5 割込み番号と割込みハンドラ番号 . . . 49
2.7.6 マルチプロセッサにおける割込み処理 . . . 49
2.7.7 カーネル管理外の割込み . . . 51
2.7.8 カーネル管理外の割込みの設定方法 . . . 51
2.8 CPU例外処理モデル . . . 52
2.8.1 CPU例外処理の流れ . . . 52
2.8.2 CPU例外ハンドラから呼び出せるサービスコール . . . 53
2.8.3 エミュレートされたCPU例外ハンドラ . . . 54
2.8.4 カーネル管理外のCPU例外 . . . 54
2.9 システムの初期化と終了 . . . 55
2.9.1 システム初期化手順 . . . 55
2.9.2 システム終了手順 . . . 56
2.10 オブジェクトの登録とその解除 . . . 58
2.10.1 ID番号で識別するオブジェクト . . . 58
2.10.2 オブジェクト番号で識別するオブジェクト . . . 59
2.10.3 識別番号を持たないオブジェクト . . . 60
2.10.4 オブジェクト生成に必要なメモリ領域 . . . 60
2.10.5 オブジェクトが属する保護ドメインの設定 . . . 60
2.10.6 オブジェクトが属するクラスの設定 . . . 61
2.10.7 オブジェクトの状態参照 . . . 62
2.11 オブジェクトのアクセス保護 . . . 62
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 . . . 62
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 . . . 64
2.11.3 デフォルトのアクセス許可ベクタ . . . 64
2.11.4 アクセス許可ベクタの設定 . . . 65
2.11.5 カーネルの管理領域のアクセス保護 . . . 66
2.11.6 ユーザタスクのユーザスタック領域 . . . 66
2.12 システムコンフィギュレーション手順 . . . 67
2.12.1 システムコンフィギュレーションファイル . . . 67
2.12.2 静的APIの文法とパラメータ . . . 68
2.12.3 保護ドメインの指定 . . . 69
2.12.4 クラスの指定 . . . 70
2.12.5 コンフィギュレータの処理モデル . . . 71
2.12.6 静的APIのパラメータに関するエラー検出 . . . 74
2.12.7 オブジェクトのID番号の指定 . . . 75
2.13 TOPPERSネーミングコンベンション . . . 75
2.14 TOPPERS共通定義 . . . 79
2.14.1 TOPPERS共通ヘッダファイル . . . 79

2.14.2 TOPPERS共通データ型 . . . 79
2.14.3 TOPPERS共通定数 . . . 81
2.14.4 TOPPERS共通エラーコード . . . 83
2.14.5 TOPPERS共通マクロ . . . 84
2.14.6 TOPPERS共通構成マクロ . . . 86
2.15 カーネル共通定義 . . . 86
2.15.1 カーネルヘッダファイル . . . 86
2.15.2 カーネル共通定数 . . . 87
2.15.3 カーネル共通マクロ . . . 88
2.15.4 カーネル共通構成マクロ . . . 89
第3章 システムインタフェースレイヤAPI仕様 . . . 92
3.1 システムインタフェースレイヤの概要 . . . 92
3.2 SILヘッダファイル . . . 92
3.3 全割込みロック状態の制御 . . . 92
3.4 SILスピンロック . . . 93
3.5 微少時間待ち . . . 94
3.6 エンディアンの取得 . . . 94
3.7 メモリ空間アクセス関数 . . . 95
3.8 I/O空間アクセス関数 . . . 97
3.9 プロセッサIDの参照 . . . 98
第4章 カーネルAPI仕様 . . . 99
4.1 タスク管理機能 . . . 99
4.2 タスク付属同期機能 . . . 132
4.3 タスク終了機能 . . . 141
4.4 同期・通信機能 . . . 149
4.4.1 セマフォ . . . 149
4.4.2 イベントフラグ . . . 163
4.4.3 データキュー . . . 178
4.4.4 優先度データキュー . . . 195
4.4.5 ミューテックス . . . 212
4.4.6 メッセージバッファ . . . 231
4.4.7 スピンロック . . . 247
4.5 メモリプール管理機能 . . . 260
4.5.1 固定長メモリプール . . . 260
4.6 時間管理機能 . . . 277
4.6.1 システム時刻管理 . . . 277
4.6.2 タイムイベントの通知処理 . . . 287
4.6.3 周期通知 . . . 292
4.6.4 アラーム通知 . . . 307
4.6.5 オーバランハンドラ . . . 321
4.7 システム状態管理機能 . . . 329
4.8 メモリオブジェクト管理機能 . . . 357
4.9 割込み管理機能 . . . 383
4.10 CPU例外管理機能 . . . 412
4.11 拡張サービスコール管理機能 . . . 416
4.12 時間パーティショニング機能 . . . 421
4.13 システム構成管理機能 . . . 429
第5章 リファレンス . . . 443
5.1 サービスコール一覧 . . . 443
5.2 静的API一覧 . . . 448
5.3 データ型 . . . 451

5.3.1 TOPPERS共通データ型 . . . 451
5.3.2 カーネルの使用するデータ型 . . . 453
5.3.3 カーネルの使用するパケット形式 . . . 453
5.4 定数とマクロ . . . 461
5.4.1 TOPPERS共通定数 . . . 461
5.4.2 TOPPERS共通マクロ . . . 463
5.4.3 カーネル共通定数 . . . 464
5.4.4 カーネル共通マクロ . . . 465
5.4.5 カーネルの機能毎の定数 . . . 465
5.4.6 カーネルの機能毎のマクロ . . . 469
5.5 構成マクロ . . . 471
5.5.1 TOPPERS共通構成マクロ . . . 471
5.5.2 カーネル共通構成マクロ . . . 471
5.5.3 カーネルの機能毎の構成マクロ . . . 472
5.6 エラーコード一覧 . . . 475
5.7 機能コード一覧【NGKI4040】 . . . 475
5.8 カーネルオブジェクトに対するアクセスの種別 . . . 478
5.9 ターゲット定義事項一覧 . . . 480
5.10 省略名の元になった英語 . . . 484
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 . . . 484
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 . . . 486
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 . . . 487
5.11 バージョン履歴 . . . 488
付録A　バージョン間の差分 . . . 489
A.1 Release 3.2.0とRelease 3.2.1の差分 . . . 489
A.2 Release 3.2.1とRelease 3.3.0の差分 . . . 489
A.3 Release 3.3.0とRelease 3.4.0の差分 . . . 502
A.4 Release 3.4.0とRelease 3.4.1の差分 . . . 506
A.5 Release 3.4.1とRelease 3.4.2の差分 . . . 509
A.6 Release 3.4.2とRelease 3.5.0の差分 . . . 510
A.7 Release 3.5.0とRelease 3.6.0の差分 . . . 511
API索引 . . . 519

このドキュメントは，TOPPERS第3世代カーネル（ITRON系）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．

現時点で，基本仕様（ASP3カーネルの仕様），保護機能対応（HRP3カーネルの仕様），マルチプロセッサ対応（FMP3カーネルの仕様），保護機能とマルチプロセッサの両方に対応する場合（HRMP3カーネルの仕様）に関しては記述が完成している．それに対して，メニーコアプロセッサ対応や動的生成対応については，十分な検討ができておらず，今後変更が加わる可能性がある．


```
 TOPPERS Third Generation Kernel (ITRON-based) Specification

 Copyright (C) 2006-2023 by Embedded and Real-Time Systems Laboratory
             Graduate School of Information Science, Nagoya Univ., JAPAN
 Copyright (C) 2006-2023 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】の項では，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】の項では，TOPPERS第3世代カーネル（ITRON系）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】の項では，この仕様と，μITRON4.0仕様，μITRON4.0/PX仕様，TOPPERS新世代カーネル統合仕様との主な違いについて説明する．

【未決定事項】の項では，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】の項では，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．

【静的API】の項では，システムコンフィギュレーションファイル中で静的APIを記述する形式を規定する．また，【C言語API】の項では，C言語からサービスコールを呼び出す形式を規定する．

【パラメータ】の項では，サービスコールおよび静的APIに渡すパラメータの名称とデータ型を規定し，簡単な説明を行う．また，【リターンパラメータ】の項では，サービスコールが返すリターンパラメータの名称とデータ型を規定し，簡単な説明を行う．【エラーコード】の項では，サービスコールおよび静的APIが返す可能性のあるメインエラーコードと，その検出条件を規定する．

【機能】の項では，サービスコールおよび静的APIの機能を規定する．

TOPPERS第3世代カーネルに属する特定のカーネルにおいてのみサポートするAPIについては，【サポートするカーネル】の項で，そのことを記述する．

また，「第４章　カーネルAPI仕様」の章では，カーネルのAPIの種別とAPIをサポートするカーネルの種類を表すために，次の記号を用いる．

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

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

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

〔S〕は静的APIを示す．

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

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

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

〔x｜y〕は，xまたはyに該当するAPIを示す．例えば，〔TP｜TM〕は，保護機能対応カーネルまたはマルチプロセッサ対応カーネルでサポートされているタスクコンテキスト専用のサービスコールを示す．

また，エラーが発生する条件を表すために，次の記号を用いる．

〔s〕はサービスコールのみで発生するエラーを示す．静的APIでは，このエラーは発生しない．

〔S〕は静的APIのみで発生するエラーを示す．サービスコールでは，このエラーは発生しない．

〔P〕は保護機能対応カーネルのみで発生するエラーを示す．保護機能対応でないカーネルでは，このエラーは発生しない．

〔M〕はマルチプロセッサ対応カーネルのみで発生するエラーを示す．マルチプロセッサ対応でないカーネルでは，このエラーは発生しない．

〔D〕は動的生成対応カーネルのみで発生するエラーを示す．動的生成対応でないカーネルでは，このエラーは発生しない．

# タグの付与方法

この仕様書では，トレーサビリティの確保のために，記述事項に対してタグを付与する．具体的には，以下に該当する記述事項を，タグを付与する対象とする．

- 対象ソフトウェアの実装に対する要求事項や制限事項
- 対象ソフトウェアの仕様に対する一般要求事項
- 対象ソフトウェアの動作環境に対する要求事項
- ターゲット定義の規定

それに対して，用語の定義や補足説明，対象ソフトウェアを使用する上での推奨事項や注意事項，仕様決定の理由，他の仕様との関係に対しては，タグを付与しない．

タグの形式と意味は次の通りである（xxxxは4桁の数字を表す）．

| | |
|---|---|
| NGKIxxxx | TOPPERS第3世代カーネル（ITRON系）全体を対象とした記述 |
| ASPSxxxx | TOPPERS/ASP3カーネルを対象とした記述 |
| FMPSxxxx | TOPPERS/FMP3カーネルを対象とした記述 |
| HRPSxxxx | TOPPERS/HRP3カーネルを対象とした記述 |
| HRMPSxxxx | TOPPERS/HRMP3カーネルを対象とした記述 |
| SSPSxxxx | TOPPERS/SSP3カーネルを対象とした記述 |

仕様書中では，ある記述事項にタグYYYYxxxx（YYYYは4〜5文字の英文字，xxxxは4桁の数字を表す）が付与されていることを，【YYYYxxxx】で表現する．それに対して，タグYYYYxxxxを参照する場合には，［YYYYxxxx］と表記する．

# 第１章　TOPPERS第3世代カーネル（ITRON系）の概要

TOPPERS第3世代カーネル（ITRON系）とは，TOPPERSプロジェクトにおいて，ITRON仕様をベースとして開発している一連のリアルタイムカーネルの総称である．この章では，TOPPERS第3世代カーネル仕様（ITRON系）の位置付けと設計方針，それに属する各カーネルの適用対象領域と設計方針について述べる．

## 1.1 TOPPERS第3世代カーネル（ITRON系）仕様の位置付け

TOPPERSプロジェクトでは，プロジェクトの第1フェーズにおいて，2000年に公開したTOPPERS/JSPカーネルを始めとして，μITRON4.0仕様およびその保護機能拡張（μITRON4.0/PX仕様）に準拠したリアルタイムカーネルを開発してきた．

μITRON4.0仕様は1999年に，μITRON4.0/PX仕様は2002年に公表されたが，それ以降，大きな仕様改訂は実施されていない．その間に，組込みシステムおよびソフトウェアのますますの大規模化・複雑化，これまで以上に高い信頼性・安全性に対する要求，小さい消費エネルギー下での高い性能要求など，組込みシステム開発を取り巻く状況は刻々変化してきた．リアルタイムカーネルに対しても，マルチプロセッサへの対応，発展的な保護機能のサポート，機能安全対応，省エネルギー制御機能のサポートなど，新しい要求が生じた．

TOPPERSプロジェクトでは，リアルタイムカーネルに対するこのような新しい要求に対応するために，μITRON4.0仕様を発展させる形で，TOPPERS新世代カーネル仕様を策定することになった．それをさらに拡張・改良したのが，TOPPERS第3世代カーネル（ITRON系）仕様である。

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第3世代カーネル（ITRON系）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第3世代カーネル（ITRON系）仕様の設計方針

TOPPERS第3世代カーネル（ITRON系）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第3世代カーネル（ITRON系）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，およびそれを拡張・改良したTOPPERS新世代カーネル仕様をベースとする．ただし，μITRON4.0仕様およびTOPPERS新世代カーネル仕様の策定時以降の状況の変化を考慮し，それら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仕様への準拠性やTOPPERS新世代カーネル仕様との互換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仕様およびTOPPERS新世代カーネル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第3世代カーネル（ITRON系）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第3世代カーネル（ITRON系）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，保護機能対応において，機能安全規格の要求を満たすことができるパーティショニング機能を実現する．また，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．

#### (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/ASP3カーネルの適用対象領域と仕様設計方針

TOPPERS/ASP3カーネル（ASPは，Advanced Standard Profileの略．3はバージョン番号を示す．以下，ASP3カーネル）は，TOPPERS第3世代カーネル（ITRON系）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，ASP3カーネルを拡張する形で開発する．

ASP3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．

ASP3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．

ASP3カーネルの機能は，カーネル内で動的なメモリ管理が不要な範囲に留める．これは，高い信頼性・安全性・リアルタイム性を要求される組込みシステムでは，システム稼働中に発生するメモリ不足への対処が難しいためである．この方針から，カーネルオブジェクトは静的に生成することとし，動的なオブジェクト生成機能は設けない．ただし，アプリケーションプログラムが動的なメモリ管理をするためのカーネル機能である固定長メモリプール機能はサポートする．

ASP3カーネルがサポートしていない機能の中で，ASP3カーネルに対して小規模な修正を行うことで対応できるものに関しては，拡張パッケージによりサポートしている．現時点で，ASP3カーネルがサポートしている拡張パッケージは次の通りである．

- ドリフト調整機能拡張パッケージ
- メッセージバッファ機能拡張パッケージ
- オーバランハンドラ機能拡張パッケージ
- タスク優先度拡張パッケージ
- 制約タスク拡張パッケージ
- サブ優先度機能拡張パッケージ
- 優先度継承拡張パッケージ
- 動的生成機能拡張パッケージ

- モノトニックタイマ機能拡張パッケージ

## 1.4 TOPPERS/FMP3カーネルの適用対象領域と仕様設計方針

TOPPERS/FMP3カーネル（FMPは，Flexible　Multiprocessor　Profileの略．3はバージョン番号を示す．以下，FMP3カーネル）は，ASP3カーネルを，マルチプロセッサ向けに拡張したリアルタイムカーネルである．

FMP3カーネルの適用対象となるターゲットハードウェアは，ホモジニアスなマルチプロセッサシステムで，プロセッサ数が4〜8個程度以下のものである．各プロセッサが全く同一のものである必要はないが，すべてのプロセッサでバイナリコードを共有することから，同じバイナリコードを実行できることが必要である．プロセッサ間の排他制御をジャイアントロックによって実現しているため，プロセッサ数が多くなると性能が劣化する．

FMP3カーネルでは，タスクを実行するプロセッサを静的に決定するのが基本であり，カーネルは自動的に負荷分散する機能を持たないが，タスクをマイグレーションさせるサービスコールを備えている．これを用いて，アプリケーションまたはシステムサービスで動的な負荷分散を実現することが可能である．

FMP3カーネルの機能は，ASP3カーネルと同様に，カーネル内で動的なメモリ管理が不要な範囲に留める．

【TOPPERS新世代カーネル統合仕様との関係】

TOPPERS新世代カーネル統合仕様に基づくTOPPERS/FMPカーネルは，ジャイアントロックと細粒度ロックの両方に対応しており，プロセッサ数が多いマルチ/メニーコアプロセッサにも適用可能であった．しかし，細粒度ロックに対応するためにコードが複雑になり，ジャイアントロックの場合に不要なオーバヘッドが生じるため，FMP3カーネルでは，ジャイアントロックのみに対応することとした．細粒度ロックに対応するカーネルは，別に開発する計画である．

## 1.5 TOPPERS/HRP3カーネルの適用対象領域と仕様設計方針

TOPPERS/HRP3カーネル（HRPは，High Reliable system Profileの略．3はバージョン番号を示す．以下，HRP3カーネル）は，さらに高い信頼性・安全性を要求される組込みシステムや，より大規模な組込みシステム向けに適用できるように，ASP3カーネルを拡張したリアルタイムカーネルである．

HRP3カーネルの適用対象となるターゲットハードウェアは，特権モードと非特権モードを備え，メモリ保護のためにMMU（Memory Management Unit）またはMPU（Memory Protection Unit）を持つプロセッサを用いたシステムである．HRP3カーネルの主な適用対象は，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）が数百KB以上のシステムである．

HRP3カーネルの機能は，ASP3カーネルと同様に，カーネル内で動的なメモリ管理が不要な範囲に留める．具体的には，ASP3カーネルに対して，メモリ保護機能，時間パーティショニング機能，オブジェクトアクセス保護機能，拡張サービスコール機能，メッセージバッファ機能（ASP3カーネルでは拡張パッケージでサポート）を追加している．

HRP3カーネルがサポートしていない機能の中で，HRP3カーネルに対して小規模な修正を行うことで対応できるものに関しては，拡張パッケージによりサポートを予定している．現時点で，HRP3カーネルがサポートしている拡張パッケージは次の通りである．

- オーバランハンドラ機能拡張パッケージ
- 動的生成機能拡張パッケージ
- モノトニックタイマ機能拡張パッケージ

## 1.6 TOPPERS/HRMP3カーネルの適用対象領域と仕様設計方針

TOPPERS/HRMP3カーネル（HRMPは，High Reliable Multiprocessor Profileの略．3はバージョン番号を示す．以下，HRMP3カーネル）は，ASP3カーネルを，マルチプロセッサ向けかつ保護機能対応に拡張したリアルタイムカーネルである．

HRMP3カーネルの適用対象となるターゲットハードウェアは，FMP3カーネルの適用対象とHRP3の適用対象の両方の特徴を兼ね備えたシステムである．

【TOPPERS新世代カーネル統合仕様との関係】

マルチプロセッサと保護機能の両方に対応したカーネルは，TOPPERS新世代カーネル統合仕様に基づいては開発しなかった．

## 1.7 TOPPERS/SSP3カーネルの適用対象領域と仕様設計方針

TOPPERS/SSP3カーネル（SSPは，Smallest Set Profileの略．3はバージョン番号を示す．以下，SSP3カーネル）は，小規模な組込みシステムに用いるために，ASP3カーネルをベースに機能を絞り込んだリアルタイムカーネルである．

SSP3カーネルの機能は，μITRON4.0仕様の「仕様準拠の最低条件」の考え方を踏襲し，メモリ使用量を最小化するように定めている．具体的には，SSP3カーネルにおいては，タスクは待ち状態を持たない（言い換えると，制約タスクのみをサポートする）のが最大の特徴である．また，複数のタスクに同じ初期優先度を設定することができない．ASP3カーネルに対して下位互換性を持つように配慮しているが，システム全体のメモリ使用量を最小化するために有用な機能は，ASP3カーネルに対して追加している．

TOPPERS/SSP3カーネルの主な適用対象は，プログラムサイズ（バイナリコード）が数KB〜数十KB程度の極めて小規模な組込みシステムである．

## 1.8 その他のカーネル

以上で紹介したカーネルに加えて，メニーコアプロセッサに対応するカーネルを開発することを計画している．

# 第２章　主要な概念と共通定義

## 2.1 仕様の位置付け

この仕様は，TOPPERS第3世代カーネル（ITRON系）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第3世代カーネル（ITRON系）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第3世代カーネル（ITRON系）は，TOPPERS/ASP3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第3世代カーネル（ITRON系）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/ASP3カーネルにおける規定】

ASP3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/FMP3カーネルにおける規定】

FMP3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/HRP3カーネルにおける規定】

HRP3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/HRMP3カーネルにおける規定】

HRMP3カーネルは，保護機能とマルチプロセッサの両方に対応するカーネルであり，動的生成対応カーネルではない【HRMPS0001】．

【TOPPERS/SSP3カーネルにおける規定】

SSP3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第3世代カーネル（ITRON系）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第3世代カーネル（ITRON系）の仕様は，ターゲットシステムによらずに定めるターゲット非依存（target-independent）の規定と，ターゲットシステム毎に定めるターゲット定義（target-defined）の規定に分けて記述する．この仕様書は，ターゲット非依存の規定について記述するものであり，この仕様書で「ターゲット定義」とした事項は，ターゲットシステム毎に用意するドキュメントにおいて規定する．

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

### 2.1.3 想定するソフトウェア構成

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

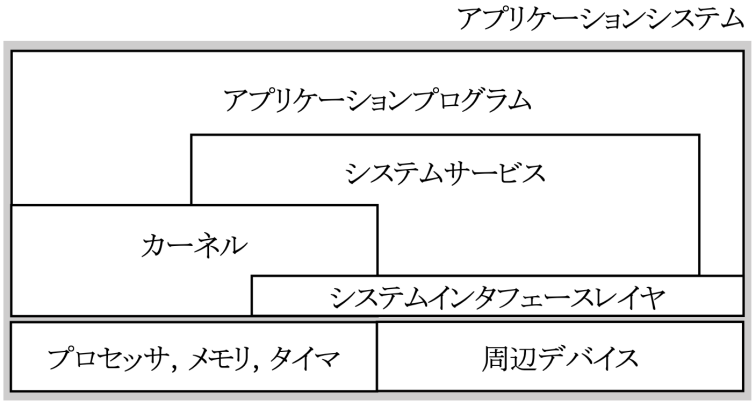


**図2−1. 想定するソフトウェア構成**

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

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

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

システムインタフェースレイヤ，カーネル，各種のシステムサービスを，上位階層のソフトウェアから使

うためのインタフェースを，API（Application Programming Interface）と呼ぶ．

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

【μITRON4.0仕様との関係】

μITRON4.0仕様では，カーネルとアプリケーションの中間にあるソフトウェアをソフトウェア部品と呼んでいたが，TOPPERS組込みコンポーネントシステム（TECS）においてはカーネルもソフトウェア部品の1つと捉えることから，この仕様ではシステムサービスと呼ぶことにした．

### 2.1.4 想定するハードウェア構成

この仕様では，カーネルがサポートするターゲットハードウェアの構成として，以下のことを想定している．

ここに挙げるのは，この仕様を策定するにあたっての想定であり，これらをすべて満たせばカーネルを動作させることができるという網羅的な要件リストではない．逆に，これらに合致しないターゲットハードウェアであっても，ターゲット依存の仕様や制約を追加することで，カーネルを動作させることが可能な場合もある．

【TOPPERS新世代カーネル統合仕様との関係】

ハードウェア構成に対する想定を追加した．TOPPERS新世代カーネル統合仕様では，(1)と(6)の2項目のみを記述していた．

(1) メモリ構成

あるメモリ番地は，常に同一のメモリを指すこと（オーバレイのように，異なるメモリを同一のメモリ番地でアクセスすることがないこと）【NGKI0001】．マルチプロセッサ対応カーネルにおいては，同一のメモリに対しては，各プロセッサから同一の番地でアクセスできること【NGKI0002】．

【補足説明】

これらの想定に合致しないメモリを持つターゲットハードウェアであっても，カーネルを動作させることは可能であるが，想定に合致しないメモリは，アプリケーションの責任で使用する必要がある．特に保護機能対応するカーネルを使用する場合には，必要なパーティショニングが達成できているか，注意して使用することが必要である．

【仕様決定の理由】

アクセスできるプロセッサが限定されたメモリを想定していないのは，そのようなメモリ構成のターゲットハードウェアが少なくなっているためである．今後，必要性が高まった場合には，アクセスできるプロセッサが限定されたメモリを想定するように仕様拡張する可能性がある．

(2) 高分解能タイマ

一定時間毎にカウントアップしながら，指定した回数カウントアップしたら割込みを発生させる機能を備えた高分解能タイマを持つこと【NGKI0556】．マルチプロセッサ対応カーネルにおいては，高分解能タイマはすべてのプロセッサから読み出すことができ，指定した回数カウントアップしたら割込みを発生させる機能は，タイムイベントを処理するプロセッサ（少なくとも1つは必要）毎に持つこと【NGKI0621】．

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルでは，指定した回数カウントアップしたら割込みを発生させる機能は，すべてのプロセッサに対して持つこと【HRMPS0013】.

【補足説明】

高分解能タイマは，一定時間毎にカウントアップするタイマと，指定した回数カウントアップしたら割込みを発生させるタイマの2つのタイマで実現することもできる．この場合，マルチプロセッサ対応カーネルにおいては，一定時間毎にカウントアップするタイマは各プロセッサで共有し，指定した回数カウントアップしたら割込みを発生させるタイマは，タイムイベントを処理するプロセッサ毎に持つことが必要である.

(3) タイムウィンドウタイマ

保護機能対応カーネルにおいては，時間パーティショニングをサポートしない場合を除き，指定した回数カウントアップしたら割込みを発生させる機能を備えたタイムウィンドウタイマを持つこと【NGKI0575】．マルチプロセッサ対応カーネルにおいては，タイムウィンドウタイマはプロセッサ毎に持つこと【NGKI0576】.

(4) オーバランタイマ

オーバランハンドラ機能をサポートする場合には，指定した回数カウントアップしたら割込みを発生させる機能を備えたオーバランタイマを持つこと【NGKI0564】．マルチプロセッサ対応カーネルにおいては，オーバランタイマはプロセッサ毎に持つこと【NGKI0565】.

(5) 実行モードとメモリ保護機構

保護機能対応カーネルにおいては，プロセッサが特権モードと非特権モードを備えていること【NGKI0646】．また，メモリ保護のために，MMU（Memory Management Unit）またはMPU（Memory Protection Unit）を持つこと【NGKI0647】.

(6) プロセッサ構成

マルチプロセッサ対応カーネルにおいては，各プロセッサが同一の機械語命令を実行できること【NGKI0003】.

(7) プロセッサ間の排他制御機構

マルチプロセッサ対応カーネルにおいては，プロセッサ間の排他制御（スピンロック）を実現するための機構を持つこと【NGKI0639】．システムインタフェースレイヤ用とカーネル内部用に，少なくとも2つのスピンロックを実現できることが望ましい【NGKI0640】.

### 2.1.5 想定するプログラミング言語

このAPI仕様における仕様は，ISO/IEC 9899:1990（以下，C90と呼ぶ）またはISO/IEC 9899:1999（以下，C99と呼ぶ）に準拠したC言語を，フリースタンディング環境で用いることを想定して規定している【NGKI0004】.

ただし，C90の規定に加えて，以下のことを仮定している.

- 16ビットおよび32ビットの整数型があること【NGKI0005】
- ポインタが格納できるサイズの整数型があること【NGKI0006】

## 2.2 APIの構成要素とコンベンション

### 2.2.1 APIの構成要素

#### (1) サービスコール

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

#### (2) コールバック

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

#### (3) 静的API

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

#### (4) 構成マクロ

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

### 2.2.2 パラメータとリターンパラメータ

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

オブジェクトを生成するサービスコールなど，パラメータの数が多い場合やオプションのパラメータがある場合，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（「2.14.4 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．

サブエラーコードは，エラーの原因をより詳細に表すために用いる．カーネルはサブエラーコードを使用せず，サブエラーコードとして常に-1が返る【NGKI0017】．サブエラーコードの名称・意味・値は，サブエラーコードを使用するシステムサービスのAPI仕様において規定する【NGKI0018】．

サービスコールが負の値のエラーコード（警告と通信エラーを表すものを除く）を返した場合には，サービスコールによる副作用がないのが原則である【NGKI0019】．ただし，そのような実装ができない場合にはこの原則の例外とし，サービスコールの機能説明にその旨を記述する【NGKI0020】．

サービスコールが複数のエラーを検出するべき状況では，その内のいずれか1つのエラーを示すエラーコードが返る【NGKI0021】．

コールバックが複数のエラーを検出するべき状況では，その内のいずれか1つのエラーを示すエラーコードを返せばよい【NGKI0022】．

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

### 2.2.4 機能コード

ソフトウェア割込みによりサービスコールを呼び出す場合などに用いるためのサービスコールを識別するための番号を，機能コード（function　code）と呼ぶ．機能コードは符号付きの整数値とし，カーネルのサービスコールには負の値を割り付け，拡張サービスコールには正の値を用いる【NGKI0023】．

### 2.2.5 ヘッダファイル

カーネルやシステムサービスを用いるために必要な定義を含むファイル．

ヘッダファイルは，原則として，複数回インクルードしてもエラーにならないように対処されている．具体的には，ヘッダファイルの先頭で特定の識別子（例えば，kernel.hなら"TOPPERS_KERNEL_H"）がマクロ定義され，ヘッダファイルの内容全体をその識別子が定義されていない場合のみ有効とする条件ディレクティブが付加されている【NGKI0024】．

## 2.3 主な概念

この仕様書では，メモリ領域（memory area）という用語は，連続したメモリの範囲という一般的な意味で使う．

### 2.3.1 オブジェクトと処理単位

(1) オブジェクト

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

オブジェクトは，種類毎に，番号によって識別する【NGKI0025】．カーネルまたはシステムサービスで，オブジェクトに対して任意に識別番号を付与できる場合には，1から連続する正の整数値でオブジェクトを識別するのを原則とする【NGKI0026】．この場合に，オブジェクトの識別番号を，オブジェクトのID番号（ID number）と呼ぶ．そうでない場合，すなわちカーネルまたはシステムサービスの内部または外部からの条件によって識別番号が決まる場合には，オブジェクトの識別番号を，オブジェクト番号（object　number）と呼ぶ．識別する必要のないオブジェクトには，識別番号を付与しない場合がある【NGKI0027】．

オブジェクト属性（object　attribute）は，オブジェクトの動作モードや初期状態を定めるもので，オブジェクトの登録時に指定する【NGKI0028】．オブジェクト属性にTA_XXXXが指定されている場合，そのオブジェクトを，TA_XXXX属性のオブジェクトと呼ぶ．複数の属性を指定する場合には，オブジェクト属性を渡すパラメータに，指定する属性値のビット毎論理和（C言語の"|"）を渡す【NGKI0029】．また，指定すべきオブジェクト属性がない場合には，TA_NULLを指定する【NGKI0030】．

### (2) 処理単位

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

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

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

例外処理を起こすための命令（ソフトウェア割込み命令）によって発生する例外処理を，ソフトウェア割込みと呼ぶ．ソフトウェア割込みは，特権モードで実行すべきサービスコールを呼び出すなどの目的で使用される．

#### 【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) タイムイベント通知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．タイムイベントをアプリケーションに通知する機能を，タイムイベント通知（time event notification），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．

#### 【μITRON4.0仕様，TOPPERS新世代カーネル統合仕様との関係】

タイムイベント通知の概念を追加した．

### 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，データ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データ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】．

サブ優先度（sub-priority）は，優先度が同一のタスクの間の優先順位を決定するために，アプリケーションがタスクに与える値である．サブ優先度機能をサポートするカーネルにおいては，サブ優先度を使用して優先順位を決定するか否かを，優先度毎に設定することができる【NGKI0558】．サブ優先度はuint_t型で表し，値が小さいほど優先度が高い【NGKI0559】．

【TOPPERS/ASP3カーネルにおける規定】

ASP3カーネルは，サブ優先度機能をサポートしない【ASPS0016】．ただし，サブ優先度機能拡張パッケージを用いると，サブ優先度機能を追加することができる【ASPS0017】．

【TOPPERS/FMP3カーネルにおける規定】

FMP3カーネルは，サブ優先度機能をサポートする【FMPS0010】．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルは，サブ優先度機能をサポートしない【HRPS0012】．

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルは，サブ優先度機能をサポートしない【HRMPS0002】．

【TOPPERS/SSP3カーネルにおける規定】

SSP3カーネルは，サブ優先度機能をサポートしない【SSPS0011】．

【μITRON4.0仕様，TOPPERS新世代カーネル統合仕様との関係】

サブ優先度の概念を追加した．

(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，64ビット（ただし，64ビットの整数型がサポートされていないターゲットでは，32ビット）の符号無しの整数型であるSYSTIM型で表し，単位はマイクロ秒とする【NGKI0548】．

タイム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，32ビットの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，すなわちマイクロ秒とする【NGKI0549】．相対時間に指定できる最大値は，4,000,000,000（66分40秒を表す）である【NGKI0550】．この値は，構成マクロTMAX_RELTIMに定義されている【NGKI0551】．

タイムイベントを発生させる時刻を相対時間で指定した場合，タイムイベントが処理されるのは，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】．

タイムイベントが発生するまでの時間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】．

タイムイベントが発生する時刻が相対時間で返された場合，タイムイベントが処理されるのは，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】．何らかの理由でタイムイベントの処理が遅れ，タイムイベントが発生する時刻をすでに過ぎている場合には，相対時間として0が返される【NGKI0552】．

【補足説明】

サービスコールで相対時間に0を指定した場合には，高分解能タイマが基準時刻後に最初にカウントアップするのをきっかけに，タイムイベントが処理される．また，1を指定した場合には，基準時刻後に2回目にカウントアップするのをきっかけに，タイムイベントが処理される．これは，基準時刻後の最初のカウントアップは，基準時刻の直後に発生する可能性があるため，ここでタイムイベントを処理すると，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が返された場合には，それ以降の可能な限り早いタイミングでタイムイベントが処理される．1が返された場合には，基準時刻後の2回目のカウントアップをきっかけにタイムイベントが処理される．

ただし，保護機能対応カーネルで時間パーティショニングを使用する場合には，ユーザドメインに属するタイムイベントの処理は，そのユーザドメインが実行されるタイムウィンドウに切り換わるまで実行されない．詳しくは，「2.6.7　　時間パーティショニング使用時のスケジューリング規則」の節を参照すること．

【μITRON4.0仕様との関係】

システム時刻（SYSTIM型）と相対時間（RELTIM型）の時間単位は，μITRON4.0仕様では実装定義としていたが，この仕様ではマイクロ秒と規定した．また，システム時刻と相対時間のビット長を定め，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．

【TOPPERS新世代カーネル統合仕様との関係】

システム時刻（SYSTIM型）と相対時間（RELTIM型）の時間単位は，TOPPERS新世代カーネル統合仕様ではミリ秒としていたが，この仕様ではマイクロ秒に変更した．また，システム時刻と相対時間のビット長を定め，相対時間に指定できる最大値を規定した．相対時間の解釈について，タイムティックを使わない実装を想定した規定に変更した．

【仕様決定の理由】

相対時間に指定できる最大値を4,000,000,000に制限したのは，タイムイベントを発生させる時刻を，カーネル内部では32ビットの整数型で扱えるようにするためである．

(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，32ビットの符号無しの整数型であるTMO型で表し，単位はシステム時刻と同一，すなわちマイクロ秒とする【NGKI0553】．タイムアウト時間に，0より大きく，TMAX_RELTIM以下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【NGKI0554】．すなわち，タイムアウトの処理が行われるのは，サービスコールを呼び出してから指定した以上の時間が経過した後となる．

ポーリング（polling）を行うサービスコールとは，サービスコールの中で待ち状態に遷移すべき状況になった場合に，サービスコールの処理を取りやめてリターンするサービスコールのことをいう．ここで，サービスコールの処理を取りやめてリターンすることを，ポーリングに失敗したという．ポーリングに失敗したサービスコールからは，E_TMOUTエラーが返る【NGKI0055】．

ポーリングを行うサービスコールでは，待ち状態に遷移することはないのが原則である【NGKI0056】．そのため，ポーリングを行うサービスコールは，ディスパッチ保留状態であっても呼び出せる【NGKI0057】．ただし，サービスコールの中で待ち状態に遷移する状況が複数ある場合，ある状況でポーリング動作をしても，他の状況では待ち状態に遷移する場合がある．このような場合の振舞いは，該当するサービスコール毎に規定する【NGKI0058】．

タイムアウト付きのサービスコールは，別に規定がない限りは，タイムアウト時間にTMO_POL（＝0）

を指定した場合にはポーリングを行い，TMO_FEVR（＝UINT32_MAX）を指定した場合にはタイムアウトを起こさないものとする【NGKI0059】．

【補足説明】

相対時間の0（基準時刻後の最初のカウントアップをきっかけにタイムイベントを処理）とタイムアウト時間のTMO_POL（＝0，ポーリング）は意味が異なる．具体的な例として，dly_tsk(0U)を呼び出したタスクは待ち状態に遷移するのに対して，tslp_tsk(TMO_POL)を呼び出したタスクは待ち状態には遷移しない．

［NGKI0019］の原則より，サービスコールがタイムアウトした場合やポーリングに失敗した場合には，サービスコールによる副作用がないのが原則である．ただし，そのような実装ができない場合にはこの原則の例外とし，どのような副作用があるかをサービスコール毎に規定する．

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

【μITRON4.0仕様との関係】

タイムアウト時間（TMO型）の時間単位は，μITRON4.0仕様では実装定義としていたが，この仕様ではマイクロ秒と規定した．また，TMO型を符号無し整数に変更するとともに，ビット長を定め，指定できる最大値を規定した．

【TOPPERS新世代カーネル統合仕様との関係】

タイムアウト時間（TMO型）の時間単位は，TOPPERS新世代カーネル統合仕様ではミリ秒単位としていたが，この仕様ではマイクロ秒に変更した．また，TMO型を符号無し整数に変更するとともに，ビット長を定め，指定できる最大値を規定した．

【仕様決定の理由】

ディスパッチ保留状態において，ポーリングを行うサービスコールを呼び出せる場合があるのに対して，タイムアウト付きのサービスコールをタイムアウト時間をTMO_POLとして呼び出すとエラーになるのは，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（タイムアウト付きのサービスコールはこれに該当）を呼び出すことはできないという原則［NGKI0175］と［NGKI0179］があるためである．

(4) ノンブロッキング

サービスコールの中で待ち状態に遷移すべき状況になった時，サービスコールの処理を継続したままサービスコールからリターンする場合，そのサービスコールをノンブロッキング（non-blocking）という．処理を継続したままリターンする場合，サービスコールからはE_WBLKエラーが返る【NGKI0060】．E_WBLKは警告を表すエラーコードであり，サービスコールによる副作用がないという原則は適用されない【NGKI0061】．

サービスコールからE_WBLKエラーが返った場合には，サービスコールの処理は継続しているため，サービスコールに渡したパラメータまたはリターンパラメータを入れる領域はまだアクセスされる可能性があり，別の目的に使用することはできない【NGKI0062】．継続している処理が完了した場合や，何らかの理由で処理が取りやめられた場合には，コールバックを呼び出すなどの方法で，サービスコールを呼び出したソフトウェアに通知するものとする【NGKI0063】．

ノンブロッキングの指定は，タイムアウト時間にTMO_NBLK（＝UINT32_MAX-1）を指定することに

よって行う【NGKI0064】．ノンブロッキングの指定を行えるサービスコールは，指定した場合の振舞いをサービスコール毎に規定する【NGKI0065】．

【補足説明】

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

(5) プロセッサ時間

プロセッサが処理単位の実行に要した時間をプロセッサ時間（processor time）と呼ぶ．プロセッサ時間は，32ビットの符号無しの整数型であるPRCTIM型で表し，単位はマイクロ秒とする【NGKI0573】．プロセッサ時間の計測精度は，ターゲットに依存する【NGKI0574】．

【補足説明】

プロセッサ時間は，処理単位の実行に要した時間であり，システム時刻の経過とは独立である．そのため，システム時刻の調整やドリフトの調整によって，プロセッサ時間の進み方が変わることはない．

【μITRON4.0仕様，TOPPERS新世代カーネル統合仕様との関係】

μITRON4.0仕様およびTOPPERS新世代カーネル統合仕様では，プロセッサ時間の概念はオーバランハンドラ機能のみで使用され，データ型の名称もOVRTIMであったが，この仕様では，時間パーティショニング機能でも使用するため，概念を一般化し，PRCTIM型に改名した．また，PRCTIM型のビット長を定めた．

### 2.3.3 保護機能

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

【TOPPERS新世代カーネル統合仕様との関係】

ユーザドメインに属する処理単位がタスクの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機能を廃止した．

(1) アクセス保護

保護機能対応カーネルは，処理単位が，許可されたカーネルオブジェクトに対して，許可された種別のアクセスを行うことのみを許し，それ以外のアクセスを防ぐアクセス保護機能を提供する【NGKI0066】．

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

(2) メモリオブジェクト

保護機能対応カーネルにおいては，メモリ領域をカーネルオブジェクトとして扱い，アクセス保護の対象とする【NGKI0067】．カーネルがアクセス保護の対象とする連続したメモリ領域を，メモリオブジェクト（memory object）と呼ぶ．メモリオブジェクトは，互いに重なりあうことはない【NGKI0068】．

メモリオブジェクトは，その先頭番地によって識別する【NGKI0069】．言い換えると，先頭番地がオブジェクト番号となる．

メモリオブジェクトの先頭番地とサイズには，ターゲットハードウェアでメモリ保護が実現できるように，ターゲット定義の制約が課せられる【NGKI0070】．

### (3) 保護ドメイン

保護機能を提供するために用いるカーネルオブジェクトの集合を，保護ドメイン（protection domain）と呼ぶ．保護ドメインは，保護ドメインIDと呼ぶID番号によって識別する【NGKI0071】．

カーネルオブジェクトは，たかだか1つの保護ドメインに属する．処理単位は，いずれか1つの保護ドメインに属さなければならないのに対して，それ以外のカーネルオブジェクトは，いずれの保護ドメインにも属さないことができる【NGKI0072】．いずれの保護ドメインにも属さないカーネルオブジェクトを，無所属のカーネルオブジェクト（independent kernel object）と呼ぶ．

処理単位がカーネルオブジェクトにアクセスできるかどうかは，処理単位が属する保護ドメインにより決まるのが原則である【NGKI0073】．すなわち，カーネルオブジェクトに対するアクセス権は，処理単位ではなく，保護ドメイン単位で管理される．このことから，ある保護ドメインに属する処理単位がアクセスできることを，単に，その保護ドメインからアクセスできるという．

ただし，タスクのユーザスタック領域は，ターゲット定義での変更がない限りは，そのタスク（とカーネルドメインに属する処理単位）のみがアクセスできる（「2.11.6 ユーザタスクのユーザスタック領域」の節を参照）．これは，［NGKI0073］の原則の例外となっている．

デフォルトでは，保護ドメインに属するカーネルオブジェクトは，同じ保護ドメイン（とカーネルドメイン）のみからアクセスできる【NGKI0075】．また，無所属のカーネルオブジェクトは，すべての保護ドメインからアクセスできる【NGKI0076】．詳しくは，「2.11.3 デフォルトのアクセス許可ベクタ」の節を参照すること．

### (4) カーネルドメインとユーザドメイン

システムには，カーネルドメイン（kernel domain）と呼ばれる保護ドメインが1つ存在する【NGKI0077】．カーネルドメインに属する処理単位は，プロセッサの特権モードで実行される【NGKI0078】．また，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【NGKI0079】．この仕様で，「ある保護ドメイン（またはタスク）のみからアクセスできる」といった場合でも，カーネルドメインからはアクセスすることができる．

カーネルドメイン以外の保護ドメインを，ユーザドメイン（user domain）と呼ぶ．ユーザドメインに属する処理単位は，プロセッサの非特権モードで実行される【NGKI0080】．また，どのカーネルオブジェクトに対してどの種別のアクセスを行えるかを制限することができる【NGKI0081】．

ユーザドメインには，1から連続する正の整数値の保護ドメインIDが付与される【NGKI0082】．カーネルドメインの保護ドメインIDは，TDOM_KERNEL（＝-1）である【NGKI0083】．

この仕様では，システムに登録できるユーザドメインの数は，32個以下に制限する【NGKI0084】．これを超える数のユーザドメインを登録した場合には，コ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ができる．

#### 【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，曖昧性を避けるために上記の定義とした．

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，サービスコールはソフトウェア割込みによって呼び出される【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，サービスコールは関数呼出しによって呼び出される【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，サービスコールは関数呼出しによって呼び出される【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，呼び出したのは単なる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．

## 2.3.4 時間パーティショニング

この節では，保護機能対応カーネルにおける時間パーティショニングに関連する主な概念について説明する．この節の内容は，保護機能対応カーネルにのみ適用される．

【μITRON4.0/PX仕様，TOPPERS新世代カーネル統合仕様との関係】

時間パーティショニングの機能を新たに導入した．

(1) システム周期

保護ドメインを繰り返し実行する基本的な周期を，システム周期（system cycle）と呼ぶ．

(2) システム動作モード

保護ドメインをどのように繰り返し実行するかは，システム動作モード（system operating mode）毎に設定することができる．システム動作モードは，システム動作モードIDと呼ぶID番号によって識別する【NGKI0577】．

保護ドメインを繰り返し実行するのを停止するシステム動作モードとして，システム周期停止モードが用意されている【NGKI0578】．システム周期停止モードのID番号は，TSOM_STP（＝-1）である【NGKI0579】．

(3) タイムウィンドウとアイドルウィンドウ

システム周期内の連続した時間区間をタイムウィンドウ（time window）と呼ぶ．タイムウィンドウは，システム動作モード毎に登録することができる【NGKI0580】．

システム周期内で，タイムウィンドウに含まれない時間区間を，アイドルウィンドウ（idle window）と呼ぶ．

タイムウィンドウは，1つのユーザドメインに割り当てる【NGKI0581】．1つのユーザドメインに，任意の数のタイムウィンドウを割り当てることができる【NGKI0582】．すなわち，1つのユーザドメインに複数のタイムウィンドウを割り当てることができるし，タイムウィンドウを割り当てないユーザドメインがあっても良い．

どのシステム動作モードにおいてもタイムウィンドウを割り当てられていないユーザドメインが1つ以上ある場合，スケジューリング上は，それらのユーザドメインをまとめて1つのユーザドメインであるかのように扱う【NGKI0583】．タイムウィンドウを割り当てられていないユーザドメインを1つにまとめたものを，アイドルドメイン（idle domain）と呼ぶ．あるオブジェクトが「アイドルドメインに属する」といった場合には，タイムウィンドウを割り当てられていないユーザドメインのいずれかに属することを意味する．

カーネルドメインにタイムウィンドウを割り当てることはできない．カーネルドメインに属する処理単位（ユーザタスクから呼び出された拡張サービスコールを除く）は，常に（すなわち，すべてのタイムウィンドウおよびアイドルウィンドウ内で）実行することができる【NGKI0623】．

マルチプロセッサ対応カーネルでは，タイムウィンドウは，プロセッサ毎に割り当てることができる【NGKI0624】．アイドルドメインにまとめられるのは，すべてのシステム動作モードにおいて，どのプロセッサに対してもタイムウィンドウを割り当てられていないユーザドメインである【NGKI0627】．

【使用上の注意】

あるユーザドメインに対して，システム動作モードAにおいてはタイムウィンドウが割り当てられており，システム動作モードBにおいては割り当てられていない場合を考える．この場合，そのユーザドメインは，システム動作モードAにおいてタイムウィンドウを割り当てられているため，アイドルドメインにはまとめられない．そのため，システム動作モードBでは，そのユーザドメインは全く実行されない．

## 2.3.5 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，たかだか1つのクラスに属する【NGKI0096】．

カーネルがどのようなクラスをサポートするかと，それらの識別名とID番号は，ターゲット定義である【NGKI0644】．カーネルヘッダファイル（kernel.h）のターゲット依存部において，各クラスの識別名が，クラスIDにマクロ定義される【NGKI0645】．

【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．

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）と呼ぶ．マルチプロセッサ対応カーネルは，同時に複数の処理単位を実行することができる【NGKI0099】．

カーネルがサポートするプロセッサの数と各プロセッサの識別名は，ターゲット定義である【NGKI0642】．カーネルヘッダファイル（kernel.h）のターゲット依存部において，TNUM_PRCIDがサポートしているプロセッサ数に，各プロセッサの識別名がプロセッサIDに，それぞれマクロ定義される【NGKI0643】．

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】．

### (3) 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】．

### (4) クラスの持つ属性とカーネルオブジェクト

タスクの初期割付けプロセッサや割付け可能プロセッサなど，カーネルオブジェクトをマルチプロセッサ上で実現する際に設定すべき属性は，そのカーネルオブジェクトが属するクラスによって定まる．

各クラスは，ターゲット定義の属性として，初期割付けプロセッサと割付け可能プロセッサ（初期割付けプロセッサを含む1つまたは複数のプロセッサ）の情報を持つ【NGKI0107】．また，保護機能に対応しないカーネルにおいては，オブジェクトを生成する際にコンフィギュレータが確保するメモリ領域（オブジェクトの管理ブロックなど）の配置場所も，ターゲット定義である【NGKI0664】．

【仕様決定の理由】

初期割付けプロセッサと割付け可能プロセッサを，カーネルオブジェクト毎に設定できるようにしなかっ

たのは，これらの属性をアプリケーション設計者が個別に設定するよりも，ターゲット依存部の実装者が有益な組み合わせをあらかじめクラスとして用意しておく方が良いと考えたためである．

#### (5) タイムイベント処理プロセッサ

タイムイベントの処理は，ターゲット定義の1つまたは複数のプロセッサで行う【NGKI0622】．タイムイベントを処理するプロセッサを，タイムイベント処理プロセッサと呼ぶ．

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルでは，すべてのプロセッサがタイムイベント処理プロセッサである【HRMPS0011】．すなわち，すべてのプロセッサがタイムイベントの処理を行う．

【TOPPERS新世代カーネル統合仕様との関係】

TOPPERS新世代カーネル統合仕様では，プロセッサ毎にシステム時刻を持つローカルタイマ方式と，システム全体で1つのシステム時刻を持つグローバルタイマ方式のどちらかを用いることとしていたが，この仕様では，システム全体で1つのシステム時刻を持つものと決めた．ただし，タイムイベント処理プロセッサを複数設けられることから，TOPPERS新世代カーネル統合仕様のグローバルタイマ方式とも異なる．これに伴い，ローカルタイマ方式とグローバルタイマ方式という用語を廃止した．

### 2.3.6 その他

【TOPPERS新世代カーネル統合仕様との関係】

保護ドメイン毎の標準セクションを廃止した．

#### (1) オブジェクトモジュール

プログラムのオブジェクトコードとデータを含むファイルを，オブジェクトモジュール（object module）と呼ぶ．オブジェクトファイルとライブラリは，オブジェクトモジュールである．

#### (2) メモリリージョン

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をメモリリージョン（memory region）と呼ぶ．

メモリリージョンは，文字列によって識別する【NGKI0112】．メモリリージョンを識別する文字列を，メモリリージョン名と呼ぶ．

【μITRON4.0/PX仕様との関係】

メモリリージョンは，μITRON4.0/PX仕様にはない概念である．

#### (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．ターゲット定義で，それ以外のセクションを標準のセクションと扱う場合もある【NGKI0113】．

#### (4) 自動メモリ配置と手動メモリ配置

保護機能対応カーネルにおいては，カーネルに登録されたオブジェクトモジュールやセクションをどの番地に配置するかはコンフィギュレータにより決定され，リンカスクリプトもコンフィギュレータにより生成されるのが標準である．これを，自動メモリ配置と呼ぶ．

それに対して，オブジェクトモジュールやセクションをどの番地に配置するかを，アプリケーションで用意するリンカスクリプトで決定する方法を，手動メモリ配置と呼ぶ．手動メモリ配置をサポートするかどうかは，ターゲット定義である【NGKI0608】．

【TOPPERS新世代カーネル統合仕様との関係】

手動メモリ配置の機能を新たに導入した．

## 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【NGKI0533】．

(a) タスク
(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/ASP3カーネルにおける規定】

ASP3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/FMP3カーネルにおける規定】

FMP3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/HRP3カーネルにおける規定】

HRP3カーネルでは，オーバランハンドラをサポートしていない【HRPS0013】．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【HRPS0014】．

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルでは，オーバランハンドラをサポートしていない【HRMPS0003】．

【TOPPERS/SSP3カーネルにおける規定】

SSP3カーネルでは，タイムイベントハンドラと拡張サービスコールをサポートしていない【SSPS0002】．

【μITRON4.0仕様，TOPPERS新世代カーネル統合仕様との関係】

タスク例外処理ルーチンを廃止した．

### 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，別に規定がない限りは，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．

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，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】．

初期化ルーチンの実行順序については「2.9.1　システム初期化手順」の節で，終了処理ルーチンの実行順序については「2.9.2 システム終了手順」の節で説明する．

**【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，mig_tsk，msta_cyc，msta_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，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で実行される【NGKI0138】．この場合，クラスの初期割付けプロセッサの情報は用いられない．ただし，割込みを複数のプロセッサで受け付けることができないターゲットシステムでは，ターゲット定義で，割込みサービスルーチンは，属するクラスの初期割付けプロセッサのみで実行されるものとする【NGKI0656】．

CPU例外ハンドラは，CPU例外が発生したプロセッサで実行される【NGKI0660】．属するクラスの割付け可能プロセッサは，そのCPU例外が発生する可能性があるプロセッサと一致していなければならない【NGKI0308】．クラスの初期割付け可能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，「△」はターゲットによってはその情報が使用される場合があることを，「–」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 | ○ | ○ |
| 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】．

【TOPPERS新世代カーネル統合仕様との関係】

割込みサービスルーチンを実行するプロセッサを，ターゲット定義で，属するクラスの初期割付けプロセッサのみとしても良いこととした．

複数のプロセッサで共通のCPU例外を許した（「2.8.1　CPU例外処理の流れ」の節を参照）ことに伴い，CPU例外が発生する可能性があるプロセッサを，CPU例外ハンドラが属するクラスの割付け可能プロセッサと一致させることとし，初期割付け可能プロセッサの情報は用いないこととした．また，CPU例外ハンドラを実行するプロセッサを，CPU例外が発生したプロセッサとした．

## 2.5 システム状態とコンテキスト

### 2.5.1 カーネル動作状態と非動作状態

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

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．呼び出した場合にはE_CTXエラーとなる【NGKI0157】．

【μITRON4.0仕様，TOPPERS新世代カーネル統合仕様との関係】

非タスクコンテキスト専用のサービスコールの概念を廃止し，非タスクコンテキストからも，タスクコンテキストと同じ名称のサービスコールを呼び出すこととした．

### 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】．

保護機能対応カーネルでは，非特権モードで実行中に全割込みロック状態になることはない【NGKI0655】．

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，unl_cpu
- unl_spn（マルチプロセッサ対応カーネルのみ）
- dis_int，ena_int，clr_int，ras_int，prb_int
- sns_ter，sns_ctx，sns_loc，sns_dsp，sns_dpn，sns_ker
- xsns_dpn（CPU例外ハンドラからのみ）
- fch_hrt

- 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】

マルチプロセッサ対応カーネルにおいて，あるプロセッサがディスパッチ禁止状態にある間は，そのプロセッサにおいてのみ，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，ディスパッチが起こるため，ディスパッチ禁止状態を使って他のプロセッサで実行されるタスクとの排他制御を実現することはできない．

### 2.5.8 ディスパッチ保留状態

非タスクコンテキストの実行中，CPUロック状態，割込み優先度マスクが全解除でない状態，ディスパッチ禁止状態では，ディスパッチが保留される【NGKI0181】．これらの状態を総称して，ディスパッチ保留状態と呼ぶ．

マルチプロセッサ対応カーネルでは，プロセッサ毎に，ディスパッチ保留状態かそうでない状態のいずれかの状態を取る【NGKI0182】．

【補足説明】

全割込みロック状態はカーネルが管理しておらず，ディスパッチが保留されることをカーネルが保証できないため，ディスパッチ保留状態に含めていない．

### 2.5.9 カーネル管理外の状態

全割込みロック状態，カーネル管理外の割込みハンドラ実行中（「2.7.7　カーネル管理外の割込み」の節を参照），カーネル管理外のCPU例外ハンドラ実行中（「2.8.4　カーネル管理外のCPU例外」の節を参照）を総称して，カーネル管理外の状態と呼ぶ．

カーネル管理外の状態では，システムインタフェースレイヤのAPIとsns_ker，ext_kerのみ（カーネル管理外のCPU例外ハンドラからは，それに加えてxsns_dpn）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【NGKI0543】．カーネル管理外の状態から，その他のサービスコールを呼び出した場合の動作は，保証されない【NGKI0544】．

カーネル管理外の状態では，少なくとも，カーネル管理の割込みはマスクされている【NGKI0545】．カーネル管理外の割込み（の一部）もマスクされている場合もある【NGKI0546】．

【補足説明】

保護機能対応カーネルでは，非特権モードで実行中にカーネル管理外の状態になることはない．

### 2.5.10 処理単位の開始・終了とシステム状態

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

| | CPUロックフラグ | 割込み優先度マスク | ディスパッチ禁止フラグ |
|---|---|---|---|
| 【タスク】【NGKI0183】 | | | |
| 実行開始条件 | 解除 | 全解除 | 許可 |
| 実行開始時処理 | そのまま | そのまま | そのまま |
| 終了前 | 原則解除(*1) | 原則全解除(*1) | 原則許可(*1) |
| 終了時処理 | 解除する | 全解除する | 許可する |
| 【カーネル管理の割込みハンドラ】【NGKI0185】 | | | |
| 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 | そのまま |
| 【割込みサービスルーチン】【NGKI0566】 | | | |
| 【タイムイベントハンドラ】【NGKI0567】 | | | |
| 実行開始条件 | 解除 | 任意(*4) | 任意 |
| 実行開始時処理 | そのまま | そのまま(*4) | 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 | そのまま(*4) | 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，割込みハンドラを起動した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．

【補足説明】

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から呼び出される．また，タイムイベントハンドラの内，周期ハンドラとアラームハンドラは高分解能タイマ割込みハンドラから，オーバランハンドラはオーバランタイマ割込みハンドラから，それぞれ呼び出される．割込み優先度マスクは，それらを呼び出す割込みハンドラでの状態のまま呼び出され，リターン時にも変更されない(*4)．

【TOPPERS新世代カーネル統合仕様との関係】

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からのリターン時に，TOPPERS新世代カーネル統合仕様では割込み優先度マスクを元に戻すものとしていたが，この仕様では元に戻さない（そのままとする）ものとした．これは，ターゲットによっては，割込み優先度マスクを元に戻すためのオーバヘッドが大きいためである．

【仕様決定の理由】

CPU例外ハンドラ中で割込み優先度マスクを変更するAPIが用意されていないにもかかわらず，CPU例外ハンドラからのリターン時に元の状態に戻す(*5)理由は次の通りである．プロセッサによっては，割込み優先度マスクがステータスレジスタ等に含まれており，CPU例外ハンドラからのリターンで自然に元の状態に戻ってしまう．ターゲットによって振舞いが異なるのは望ましくないため，ターゲットによらず，元の状態に戻すこととしている．

CPU例外ハンドラの実行開始時には，CPUロックフラグは変更されない(*6)ことから，CPUロック状態でCPU例外が発生した場合，CPU例外ハンドラの実行開始直後はCPUロック状態となっている．CPUロック状態で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はtrueを返す），CPU例外ハンドラ中でunl_cpuを呼び出してCPUロック状態を解除しようとした場合の動作は保証されない．ただし，保証されないにも関わらずunl_cpuを呼び出した場合も考えられるため，リターン時には元に戻すこととしている．

## 2.6 タスクの状態遷移とスケジューリング規則

【μITRON4.0/PX仕様，TOPPERS新世代カーネル統合仕様との関係】

時間パーティショニングの機能とスケジューリング単位の概念を新たに導入した．

μITRON4.0/PX仕様で導入された待ち禁止状態，TOPPERS新世代カーネル統合仕様で導入されたタスク例外処理マスク状態の概念は廃止した．新たに導入したタスク終了要求フラグがセットされた状態が，待ち禁止状態の役割を兼ねている．

### 2.6.1 基本的なタスク状態

カーネルに登録したタスクは，実行できる状態，休止状態，広義の待ち状態のいずれかの状態を取る【NGKI0193】．また，実行できる状態と広義の待ち状態を総称して，起動された状態と呼ぶ．さらに，タスクをカーネルに登録していない仮想的な状態を，未登録状態と呼ぶ．

(a) 実行できる状態（runnable）

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

(a.1) 実行状態（running）

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

(a.2) 実行可能状態（ready）

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

(b) 休止状態（dormant）

タスクが実行すべき処理がない状態．タスクの実行を終了した後，次に起動するまでの間は，タスクは休止状態となっている．タスクが休止状態にある時には，タスクの実行を再開するための情報（実行再開番地やレジスタの内容など）は保存されていない【NGKI0194】．

(c) 広義の待ち状態（blocked）

タスクが，処理の途中で実行を止められている状態．タスクが広義の待ち状態にある時には，タスクの実行を再開するための情報（実行再開番地やレジスタの内容など）は保存されており，タスクが実行を再開

する時には，広義の待ち状態に遷移する前の状態に戻される【NGKI0195】．広義の待ち状態は，さらに，（狭義の）待ち状態，強制待ち状態，二重待ち状態に分類される．

(c.1)（狭義の）待ち状態（waiting）

タスクが何らかの条件が揃うのを待つために，自ら実行を止めている状態．

(c.2) 強制待ち状態（suspended）

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

(c.3) 二重待ち状態（waiting-suspended）

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

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

(d) 未登録状態（non-existent）

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

カーネルによっては，これらのタスク状態以外に，過渡的な状態が存在する場合がある【NGKI0196】．過渡的な状態については，「2.6.5　ディスパッチ保留状態で実行中のタスクに対する強制待ち」の節を参照すること．

【TOPPERS/ASP3カーネルにおける規定】

ASP3カーネルでは，タスクが未登録状態になることはない【ASPS0005】．また，上記のタスク状態以外の過渡的な状態になることもない【ASPS0006】．ただし，動的生成機能拡張パッケージでは，タスクが未登録状態になる【ASPS0007】．

【TOPPERS/FMP3カーネルにおける規定】

FMP3カーネルでは，タスクが未登録状態になることはない【FMPS0003】．上記のタスク状態以外の過渡的な状態として，タスクが強制待ち状態［実行継続中］になることがある【FMPS0004】．詳しくは，「2.6.5 ディスパッチ保留状態で実行中のタスクに対する強制待ち」の節を参照すること．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルでは，タスクが未登録状態になることはない【HRPS0002】．また，上記のタスク状態以外の過渡的な状態になることもない【HRPS0003】．ただし，動的生成機能拡張パッケージでは，タスクが未登録状態になる【HRPS0010】．

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルでは，タスクが未登録状態になることはない【HRMPS0004】．上記のタスク状態以外の過渡的な状態として，タスクが強制待ち状態［実行継続中］になることがある【HRMPS0005】．詳しくは，「2.6.5 ディスパッチ保留状態で実行中のタスクに対する強制待ち」の節を参照すること．

【TOPPERS/SSP3カーネルにおける規定】

SSP3カーネルでは，タスクが広義の待ち状態と未登録状態になることはない【SSPS0003】．また，上記のタスク状態以外の過渡的な状態になることもない【SSPS0004】．

### 2.6.2 タスクの状態遷移

タスクの状態遷移を図2-2に示す【NGKI0197】．

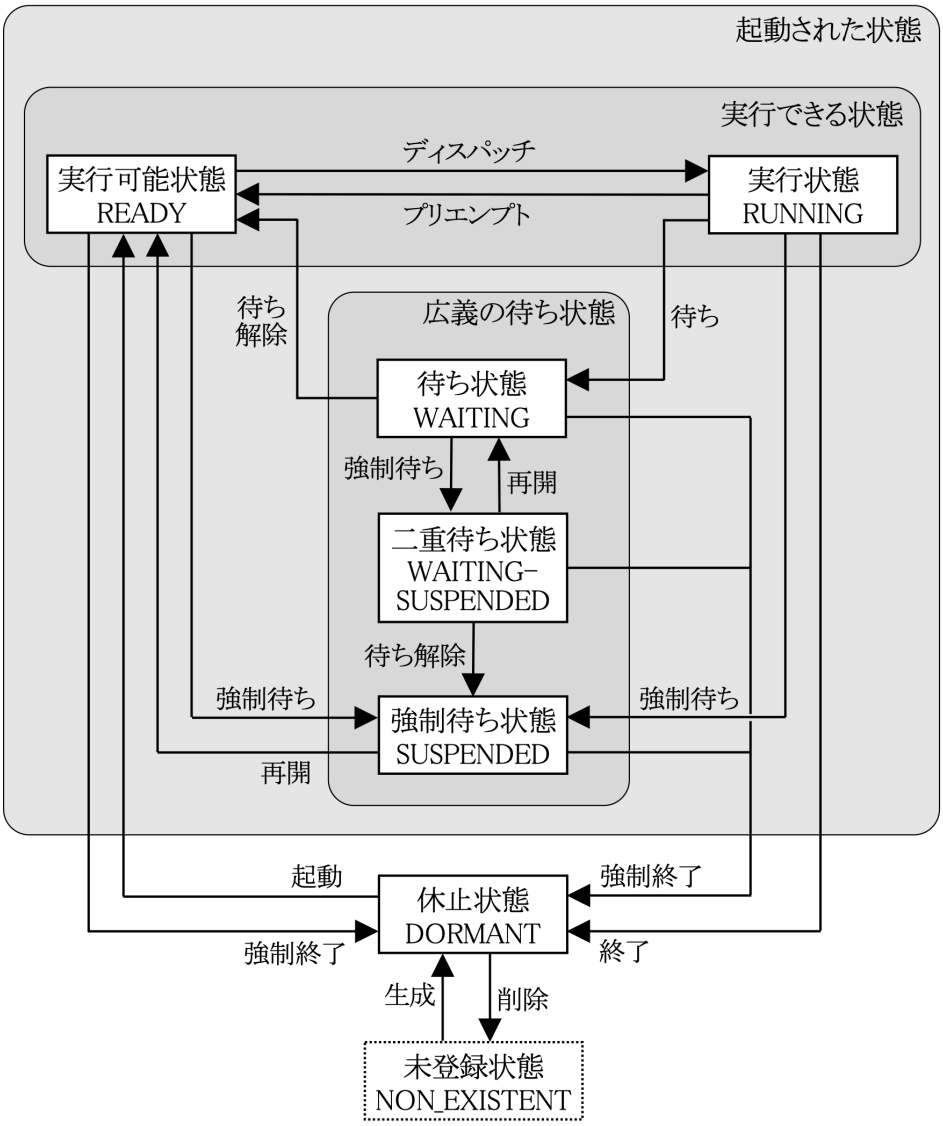


**図2-2. タスクの状態遷移**

未登録状態のタスクをカーネルに登録することを，タスクを生成する（create）という．生成されたタスクは，休止状態に遷移する【NGKI0198】．また，タスク生成時の属性指定により，生成と同時にタスクを起動し，実行できる状態にすることもできる【NGKI0199】．逆に，登録されたタスクを未登録状態に遷移させることを，タスクを削除する（delete）という．

休止状態のタスクを，実行できる状態にすることを，タスクを起動する（activate）という．起動されたタスクは，実行できる状態になる【NGKI0200】．逆に，起動された状態のタスクを，休止状態（または未登録状態）に遷移させることを，タスクを終了する（terminate）という．

実行できる状態になったタスクは，まずは実行可能状態に遷移するが，そのタスクの優先順位が実行状態のタスクよりも高い場合には，ディスパッチ保留状態でない限りはただちにディスパッチが起こり，実行状態へ遷移する【NGKI0201】．この時，それまで実行状態であったタスクは実行可能状態に遷移する【NGKI0202】．この時，実行状態に遷移したタスクは，実行可能状態に遷移したタスクをプリエンプトしたという．逆に，実行可能状態に遷移したタスクは，プリエンプトされたという．

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

【補足説明】

タスクの実行開始とは，タスクが起動された後に最初に実行される（実行状態に遷移する）時のことをいう．

### 2.6.3 タスクのスケジューリング規則

実行できるタスクは，優先順位の高いものから順に実行される【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，別に規定がない限りは，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】．

保護機能対応カーネルにおいて，カーネルドメインに属するタスクとユーザドメインに属するタスクが同一の優先度を持つ場合には，カーネルドメインに属するタスクが高い優先順位を持つ【NGKI0588】．カーネルドメイン内とユーザドメイン内では，別に規定がない限りは，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【NGKI0589】．なお，時間パーティショニングを使用した場合のスケジューリング規則については，「2.6.7　時間パーティショニング使用時のスケジューリング規則」の節を参照すること．

サブ優先度機能をサポートするカーネルにおいては，サブ優先度を使用して優先順位を決定すると設定した同一の優先度を持つタスク（保護機能対応カーネルにおいては，さらに，同じ保護ドメインに属するもの）の間では，別に規定がない限りは，サブ優先度の高いタスクが高い優先順位を持つ【NGKI0560】．サブ優先度も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【NGKI0561】．

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．

【μITRON4.0仕様，TOPPERS新世代カーネル統合仕様との関係】

サブ優先度機能を追加した．

【μITRON4.0/PX仕様，TOPPERS新世代カーネル統合仕様との関係】

保護機能対応カーネルにおいて，カーネルドメインに属するタスクは，ユーザドメインに属する同じ優先度のタスクよりも，高い優先順位を持つものとした．

### 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 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】．

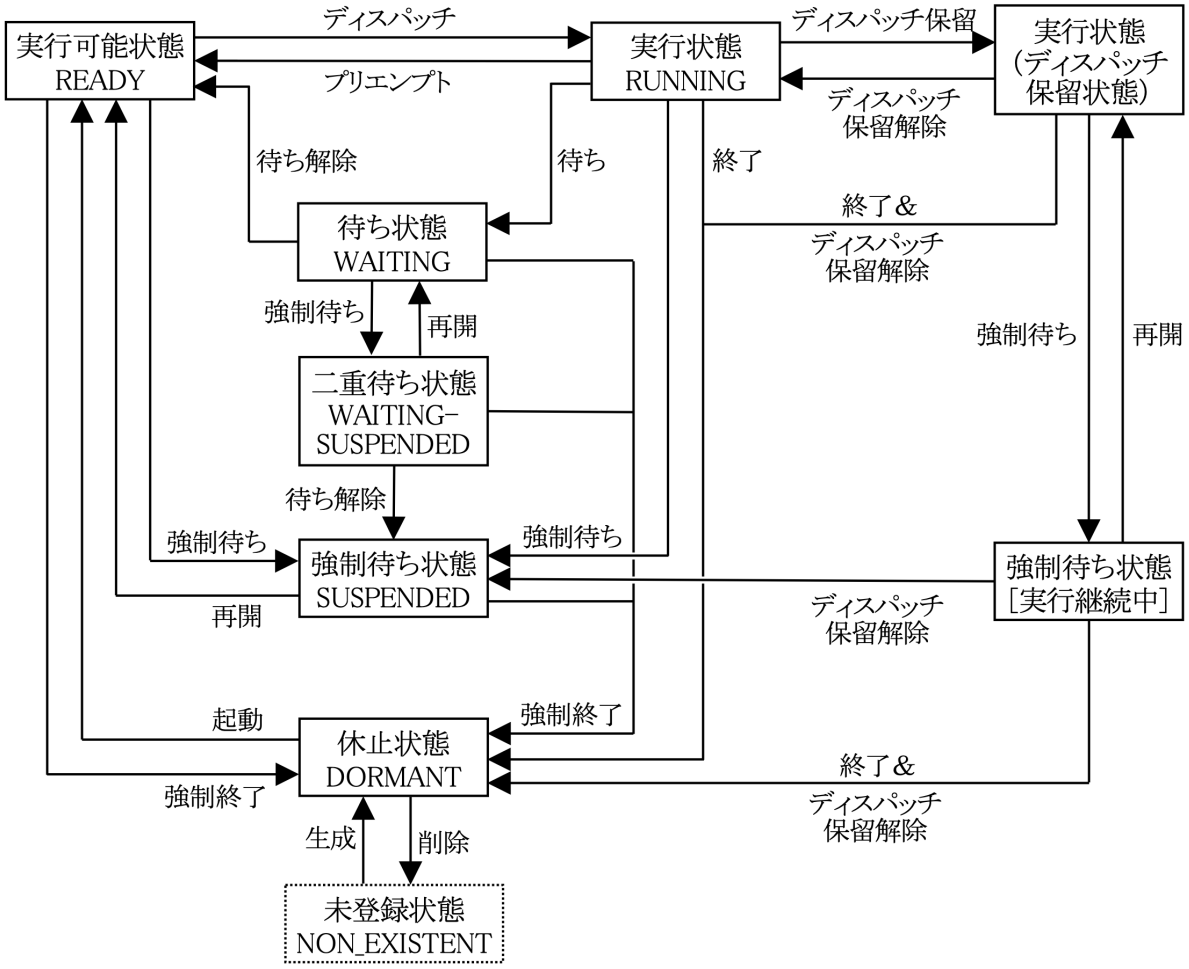


**図2-3. 過渡的な状態も含めたタスクの状態遷移**

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．

(a) 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】．

強制待ち状態［実行継続中］のタスクが，自タスクを終了させるサービスコールを発行すると，休止状態に遷移する【NGKI0629】．ただし，マルチプロセッサ対応カーネルでは，CPUロック状態で強制待ち状態［実行継続中］のタスクがext_tskを発行した場合には，強制待ち状態に遷移する場合もある【NGKI0648】．

(b) 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，get_did，sns_ter）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_terを発行するとそのタスクのタスク終了禁止フラグを参照する【NGKI0534】．

(c) 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/ASP3カーネルにおける規定】

ASP3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/FMP3カーネルにおける規定】

FMP3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/HRP3カーネルにおける規定】

HRP3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/HRMP3カーネルにおける規定】

HRMP3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【HRMPS0006】．

【TOPPERS/SSP3カーネルにおける規定】

SSP3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．

【μITRON4.0仕様との関係】

μITRON4.0仕様では，過渡的な状態におけるタスクの扱いは，実装依存とされている．

【TOPPERS新世代カーネル統合仕様との関係】

強制待ち状態［実行継続中］のタスクが，自タスクを終了させるサービスコールを発行した場合の扱いを規定した．

### 2.6.6 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．

(a) 広義の待ち状態に入ることができない【NGKI0235】．

(b) サービスコールによりベース優先度やサブ優先度を変更することができない【NGKI0236】．

(c)　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，mrot_rdq）を行うことができない【NGKI0237】．

(d) 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】．

【TOPPERS/ASP3カーネルにおける規定】

ASP3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/FMP3カーネルにおける規定】

FMP3カーネルでは，制約タスクをサポートしていない【FMPS0006】．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルでは，制約タスクをサポートしていない【HRPS0005】．

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルでは，制約タスクをサポートしていない【HRMPS0007】．

【TOPPERS/SSP3カーネルにおける規定】

SSP3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．

## 2.6.7 時間パーティショニング使用時のスケジューリング規則

保護機能対応カーネルで，時間パーティショニングを使用する場合には，以下の規則に従ってタスクがスケジューリングされる．

プロセッサは，システム周期毎に，現在のシステム動作モードに対して登録されたタイムウィンドウを順に実行する（図2-4）【NGKI0590】．あるタイムウィンドウの実行中は，カーネルドメインに属する実行できる処理単位と，そのタイムウィンドウを割り当てられたユーザドメインに属する実行できるタスクの中で，優先順位の高いものから順に実行される【NGKI0591】．これらの保護ドメインに属する実行できる処理単位がない場合には，アイドルドメインに属する実行できるタスクの中で，優先順位の高いものから順に実行される【NGKI0592】．

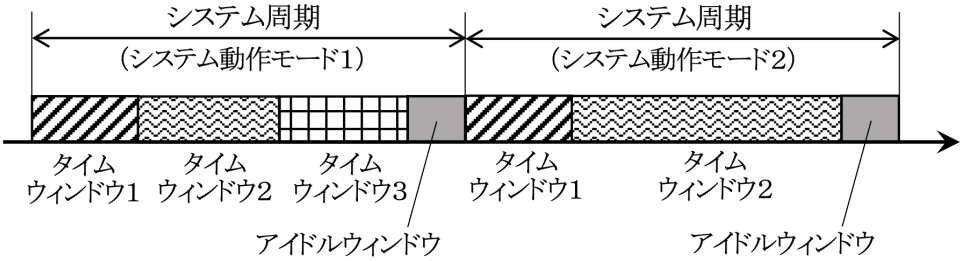


**図2-4. システム周期毎のタイムウィンドウの実行**

実行中のタイムウィンドウが使用したプロセッサ時間（いずれの処理単位も実行していなかった時間も含

む）から，カーネルドメインに属する処理単位（ユーザタスクから呼び出された拡張サービスコールを除く）が使用したプロセッサ時間を減じたものが，タイムウィンドウの長さに達すると，次のタイムウィンドウに切り換えられる（図2-5）【NGKI0593】．次のタイムウィンドウが設定されていない場合（言い換えると，実行中のタイムウィンドウが，現在のシステム動作モードに対して設定された最後のタイムウィンドウである場合）には，アイドルウィンドウに切り換えられる【NGKI0594】．アイドルウィンドウの実行中は，カーネルドメインに属する実行できる処理単位と，アイドルドメインに属する実行できるタスクの中で，優先順位の高いものから順に実行される【NGKI0595】．

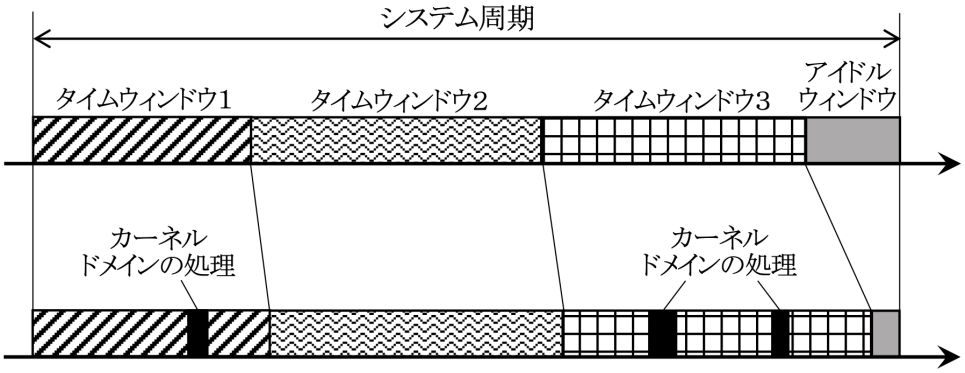


**図2−5. タイムウィンドウの切換えタイミング**

システム周期の終了時刻になると，システム周期の切換え処理が行われる．具体的には，まず，アイドルウィンドウが実行中であるか（言い換えると，現在のシステム動作モードに対して設定されたタイムウィンドウの実行がすべて終わっているか）をチェックし，そうでない場合，カーネルはエミュレートされたCPU例外を発生させる【NGKI0596】．これを，システム周期オーバラン例外と呼ぶ．次に，システム動作モードを，次のシステム周期での遷移先システム動作モードに切り換える【NGKI0597】．その後に，新しいシステム周期を開始する．すなわち，新しいシステム動作モードに対して設定された最初のタイムウィンドウに切り換える【NGKI0598】．

システム動作モードがシステム周期停止モードである場合には，システム周期の切換えも，タイムウィンドウの実行も行われず，カーネルドメインに属する処理単位（ユーザタスクから呼び出された拡張サービスコールを除く）の中で，優先順位が高いものから順に実行される【NGKI0599】．

ユーザドメインに属するタイムイベントの処理（ユーザドメインに属する周期通知とアラーム通知の処理，ユーザドメインに属するタスクに対するタイムアウト処理と時間経過待ち状態からの待ち解除処理）は，そのユーザドメインに割り当てられたタイムウィンドウへの切換え時（厳密には，タイムウィンドウを切り換え，タイムウィンドウ通知を処理した後，タスクの実行を開始する前）に実行される【NGKI0600】．アイドルドメインに属するタイムイベントの処理は，アイドルウィンドウへの切換え時に実行される【NGKI0601】．

ディスパッチ保留状態では，システム周期の切換えとタイムウィンドウの切換えは保留される【NGKI0602】．

システム周期の切換えは，高分解能タイマ割込みをきっかけに実行される．そのため，高分解能タイマ割込みのマスクや，ディスパッチの保留によって，システム周期の切換えは遅れることになる．この場合，その遅れ時間の分，次のシステム周期（結果的には，その最後のアイドルウィンドウ）が短くなる【NGKI0650】．システム周期の切換えが，次のシステム周期末を超えて遅れる状況では，カーネルの動作は保証されない【NGKI0651】．

また，タイムウィンドウの切換えは，タイムウィンドウタイマ割込みをきっかけに実行される．そのため，タイムウィンドウタイマ割込みのマスクや，ディスパッチの保留によって，タイムウィンドウの切換えは遅れることになる．この場合，その遅れ時間の分，そのシステム周期の最後のアイドルウィンドウが短くなる【NGKI0652】．タイムウィンドウの切換えが，そのシステム周期末を超えて遅れる状況では，カーネルの動作は保証されない【NGKI0653】．

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がスケジューリングされる【NGKI0625】．システム周期とシステム動作モードは，すべてのプロセッサで同一である【NGKI0626】．ただし，システム周期の切換え処理は，各プロセッサで同期して行われるわけではない．現在のシステム動作モードは，システム周期の切換え処理を最初に行ったプロセッサによって切り換えられる【NGKI0637】．そのため，他のプロセッサでは，一時的に，切換え前のシステム動作モードで実行している状況が生じる【NGKI0636】．なお，タイムウィンドウは，プロセッサ毎に登録することができる［NGKI0624］．

マルチプロセッサ対応カーネルでは，また，アイドルドメインにまとめられていないユーザドメインに属するタスク，周期通知，アラーム通知を，そのユーザドメインに対してどのシステム動作モードにおいてもタイムウィンドウを割り当てられていないプロセッサに割り付けることはできない【NGKI0628】．そのため例えば，プロセッサAでタイムウィンドウを割り当てられており，プロセッサBでタイムウィンドウを割り当てられていないユーザドメインに属するタスクを，プロセッサBに割り付けることはできない．

なお，システム周期オーバラン例外は，カーネル管理外のCPU例外であり，その例外ハンドラ番号はターゲット定義である【NGKI0603】．システム周期オーバラン例外に対して登録したCPU例外ハンドラからリターンすると，実行中のタイムウィンドウは打ち切られ，システム周期の切換え処理が継続される【NGKI0635】．

【使用上の注意】

システム周期内で，カーネルとカーネルドメインに属する処理単位（ユーザタスクから呼び出された拡張サービスコールを除く）が使用するプロセッサ時間や，システム時刻の調整やドリフト量の設定によるシステム周期の短縮，タイマ割込みのマスクやディスパッチの保留によるシステム周期の切換えとタイムウィンドウの切換えの遅延を考慮し，十分な長さのアイドルウィンドウを確保するのは，アプリケーションの責任である．

ユーザドメインに割り当てられたタイムウィンドウの実行途中に，そのユーザドメインに属するタイムイベントの発生時刻になっても，タイムイベントはすぐには処理されない．タイムイベントが処理されるのは，そのユーザドメインに割り当てられた次のタイムウィンドウへの切換え時である．

### 2.6.8 スケジューリング単位

タスクのスケジューリングに関する以上の仕様より，タスクを，FCFS方式に基づく優先順位が管理される単位に分類することができる．この単位（タスクの集合）をスケジューリング単位と呼ぶ．カーネルの実装上は，スケジューリング単位毎にレディキューを持つ．

保護機能とマルチプロセッサのいずれにも対応しないカーネルは，唯一のスケジューリング単位を持つ．すなわち，すべてのタスクが1つのスケジューリング単位に属する．

保護機能とマルチプロセッサのいずれかまたは両方に対応するカーネルは，複数のスケジューリング単位を持つ．また，スケジューリング単位を指定して操作を行うサービスコールをサポートする．それらのサービスコールにおいて，スケジューリング単位を指定する番号を，スケジューリング単位番号と呼ぶ．スケジューリング単位番号は，符号付きの整数型であるID型で表す．

保護機能対応カーネルにおいては，カーネルドメイン，アイドルドメイン，アイドルドメインにまとめられていない各ユーザドメインがスケジューリング単位となる．スケジューリング単位番号には，保護ドメインIDまたは呼び出した処理単位が属する保護ドメインを表すTDOM_SELF（＝0）を指定することができる【NGKI0630】．保護ドメインIDで指定したユーザドメインがアイドルドメインにまとめられている場合には，アイドルドメインを指定したものとする【NGKI0631】．

マルチプロセッサ対応カーネルにおいては，各プロセッサがスケジューリング単位となる．スケジューリング単位番号には，プロセッサIDを指定することができる【NGKI0632】．

保護機能とマルチプロセッサの両方に対応するカーネルにおいては，各プロセッサに対して，カーネルドメイン，アイドルドメイン，アイドルドメインにまとめられていない各ユーザドメインがスケジューリング単位となる．スケジューリング単位番号には，SCHEDNO(prcid，domid)を指定する【NGKI0633】．ここで，prcidはマルチプロセッサ対応カーネルにおけるスケジューリング単位番号，domidは保護機能対応カーネルにおけるスケジューリング単位番号，SCHEDNOは保護機能とマルチプロセッサの両方に対応するカーネルにおけるスケジューリング単位番号を構成するためのマクロである．

## 2.7 割込み処理モデル

TOPPERS第3世代カーネルにおける割込み処理モデルの概念図を図2-6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．

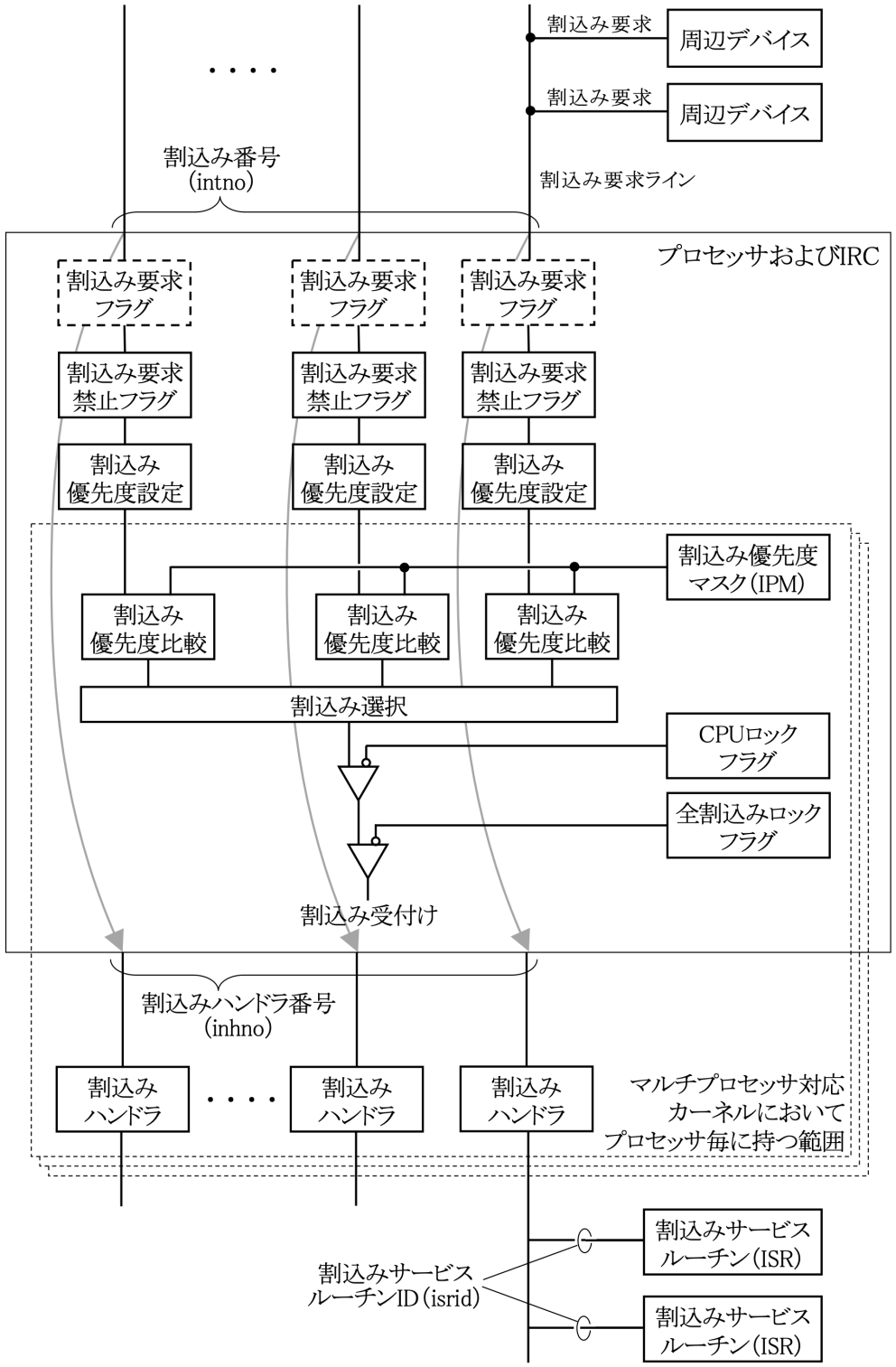


**図2-6. TOPPERS標準割込み処理モデルの概念図**

【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】．

カーネルが用いる高分解能タイマ，タイムウィンドウタイマ，オーバランタイマ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，タイムアウト処理やタイムイベントの通知処理（タイムイベントハンドラの呼出しを含む），タイムウィンドウの切換え処理，オーバランハンドラの呼出しなどが行われる【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】．

プロセッサが割込み要求禁止フラグを実現するための機能を持たないか，実現するために大きいオーバヘッドを生じる場合には，ターゲット定義で，割込み要求禁止フラグをサポートしない場合がある【NGKI0260】．また，プロセッサの持つ割込み要求禁止フラグの機能がこの仕様に合致しない場合には，ターゲット定義で，割込み要求禁止フラグをサポートしないか，振舞いが異なるものとする場合がある【NGKI0261】．

#### (2) 割込み優先度

割込み要求ライン毎に，割込み優先度を設定することができる【NGKI0262】．割込み要求ラインに対して設定された割込み優先度が，それが伝える割込み要求の割込み優先度となる【NGKI0263】．

#### (3) トリガモードと割込み要求フラグ

割込み要求ラインに対する割込み要求が，レベルトリガであるかエッジトリガであるかを設定することができる【NGKI0264】．エッジトリガの場合には，さらに，ターゲット定義で，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場合もある【NGKI0265】．また，レベルトリガの場合には，ターゲット定義で，ローレベルトリガかハイレベルトリガかを設定できる場合もある【NGKI0266】．

エッジトリガに設定された割込み要求ラインに対しては，割込み要求の有無を示す割込み要求フラグを持つ【NGKI0657】．割込み要求フラグは，割込み要求を示すエッジが検出されるとセットされ，割込みが受け付けられるとクリアされる【NGKI0658】．

プロセッサがトリガモードを設定するための機能を持たないか，設定するために大きいオーバヘッドを生じる場合には，ターゲット定義で，トリガモードの設定をサポートしない場合がある【NGKI0267】．

属性が設定されていない割込み要求ラインに対しては，割込み要求禁止フラグがセットされ，割込み要求はマスクされる【NGKI0268】．また，割込み要求禁止フラグをクリアすることもできない【NGKI0269】．

#### 【使用上の注意】

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

複数のデバイスからの割込み要求が接続されている割込み要求ラインを，エッジトリガに設定することは推奨されない．これは，次のような状況において，割込み要求を取りこぼす可能性があるためである．ある割込み要求ラインに，デバイスAとデバイスBからの割込み要求が接続されており，デバイスAの割込み

処理を先に行う場合を考える．この時，デバイスBからの割込み要求によって割込みハンドラが実行され，デバイスAの割込み処理を行った後，デバイスBの割込み処理を行う前に，デバイスAからの割込み要求が発生した場合に，デバイスAからの割込み要求を取りこぼしてしまう．

【TOPPERS新世代カーネル統合仕様との関係】

clr_int，ras_intの振舞いを明確化するために，割込み要求フラグの概念を導入した．

### 2.7.4 割込みを受け付ける条件

NMI以外の割込み要求は，次の4つの条件が揃った場合に受け付けられる【NGKI0270】．

(a) 割込み要求ラインに対する割込み要求禁止フラグがクリアされていること

(b)　割込み要求ラインに設定された割込み優先度が，割込み優先度マスクの現在値よりも高い（優先度の値としては小さい）こと

(c) 全割込みロックフラグがクリアされていること

(d) 割込み要求がカーネル管理の割込みである場合には，CPUロックフラグがクリアされていること

これらの条件が揃った割込み要求が複数ある場合に，どの割込み要求が最初に受け付けられるかは，この仕様では規定しない【NGKI0271】．すなわち，割込み優先度の高い割込み要求が先に受け付けられるとは限らない．

### 2.7.5 割込み番号と割込みハンドラ番号

割込み要求ラインを識別するための番号を，割込み番号と呼ぶ．割込み番号は，符号無しの整数型であるINTNO型で表し，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【NGKI0272】．そのため，1から連続した正の値であるとは限らない．

それに対して，アプリケーションが用意した割込みハンドラをカーネルに登録する場合に，割込みハンドラの登録対象となる割込みを識別するための番号を，割込みハンドラ番号と呼ぶ．割込みハンドラ番号は，符号無しの整数型であるINHNO型で表し，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【NGKI0273】．そのため，1から連続した正の値であるとは限らない．

割込みハンドラ番号は，割込み番号と1対1に対応するのが基本である（両者が一致する場合が多い）【NGKI0274】．

ただし，割込みを要求したデバイスが割込みベクタを生成してプロセッサに渡すアーキテクチャなどでは，割込み番号と割込みハンドラ番号の対応を，カーネルが管理していない場合がある【NGKI0275】．そこで，ターゲット定義で，割込み番号に対応しない割込みハンドラ番号や，割込みハンドラ番号に対応しない割込み番号を設ける場合もある【NGKI0276】．ただし，割込みサービスルーチンの登録対象にできる割込み番号は，割込みハンドラ番号との1対1の対応関係をカーネルが管理しているもののみである【NGKI0277】．

### 2.7.6 マルチプロセッサにおける割込み処理

この節では，マルチプロセッサにおける割込み処理について説明する．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．

マルチプロセッサ対応カーネルでは，割込み処理モデルの概念図（図2-6）の中で，破線に囲まれた部分はプロセッサ毎に持ち，それ以外の部分はシステム全体で1つのみ持つ【NGKI0278】．すなわち，全割込みロックフラグ，CPUロックフラグ，割込み優先度マスクはプロセッサ毎に持つのに対して，割込み要求ラ

インおよびその属性（割込み要求禁止フラグ，割込み優先度，トリガモード，割込み要求フラグ）はシステム全体で共通に持つ．

割込み番号は，割込み要求ラインを識別するための番号であることから，割込み要求ラインが複数のプロセッサに接続されている場合でも，1つの割込み要求ラインには1つの割込み番号を付与する【NGKI0279】．逆に，複数のプロセッサが同じ種類のデバイスを個別に持っている場合でも，別のデバイスからの割込み要求ラインには異なる割込み番号を付与する（図2-7）【NGKI0280】．図2-7において，ローカルIRCは個々のプロセッサに対する割込みを制御するための回路であり，グローバルIRCはデバイスからの割込みをプロセッサに分配するための回路である．グローバルIRCは，必ず備わっているとは限らない．

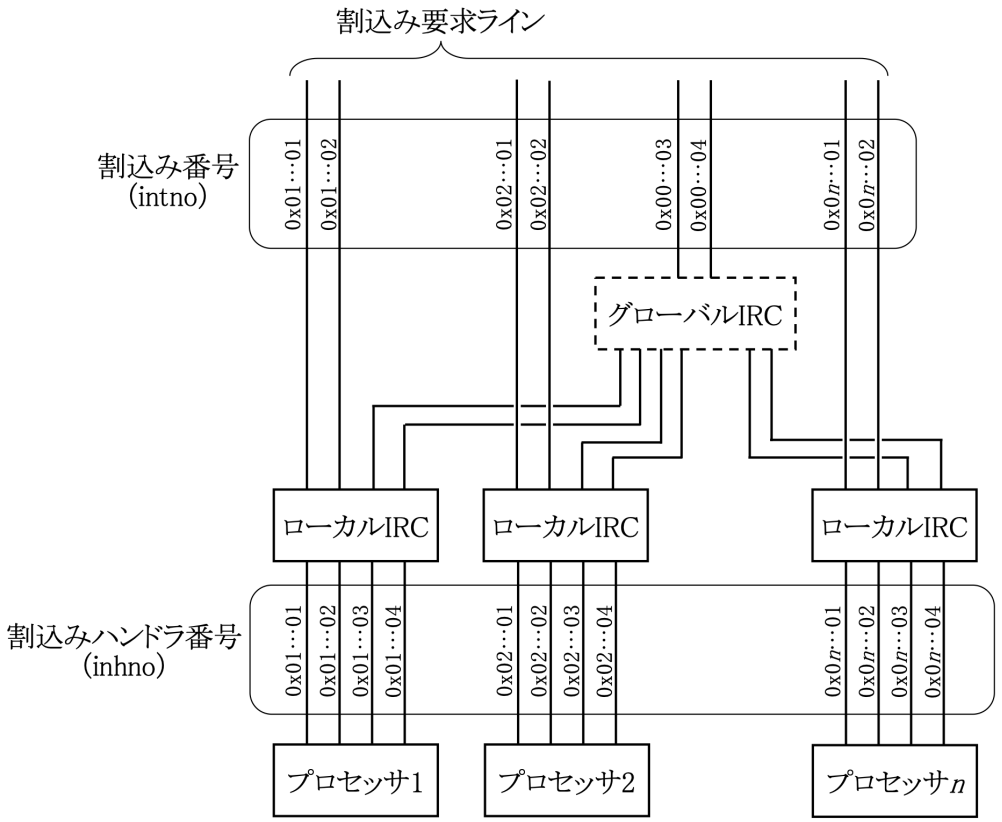


**図2–7. マルチプロセッサ対応カーネルにおける割込み番号と割込みハンドラ番号**

割込み要求禁止フラグは，この仕様上はシステム全体で共通に持つこととしているが，実際のターゲットハードウェア（特に，グローバルIRCを備えていないもの）では，プロセッサ毎に持っている場合がある．そのため，ターゲット定義で，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場合があるものとする【NGKI0281】．

複数のプロセッサに接続された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は，それらのプロセッサのいずれによっても実行することができる【NGKI0282】．ただし，その内のどのプロセッサで割込みサービスルーチンを実行するかは，割込みサービスルーチンが属するクラスの割付け可能プロセッサにより決定される（「2.4.4 処理単位を実行するプロセッサ」の節を参照）．

割込みサービスルーチンが複数のプロセッサで実行できる場合に，連続して複数回の割込み要求があっても，割込みサービスルーチンが異なるプロセッサで同時に実行されることはない【NGKI0659】．

割込みサービスルーチンが属するクラスの割付け可能プロセッサは，登録対象の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ければならない【NGKI0283】．また，同一の割込み要求ラインに対して登録する割込みサービスルーチンは，同一のクラスに属していなければならない【NGKI0284】．

それに対して，割込みハンドラはプロセッサ毎に登録する．そのため，同じ割込み要求に対応する割込みハンドラであっても，プロセッサ毎に異なる割込みハンドラ番号を付与する（図2-7）【NGKI0285】．割込みハンドラが属するクラスの初期割付けプロセッサは，割込みが要求されるプロセッサと一致していなければならない【NGKI0286】．

【補足説明】

マルチプロセッサ対応カーネルにおける割込み番号の付与方法は，すべてのプロセッサに接続された割込み要求ラインに対しては，割込み番号の上位ビットを0とし，1つのプロセッサのみに接続された割込み要求ラインに対しては，割込み番号の上位ビットに，接続されたプロセッサのID番号を含める方法を基本とする．また，割込みハンドラ番号の付与方法は，割込みハンドラ番号の上位ビットに，その割込みハンドラを実行するプロセッサのID番号を含める方法を基本とする（図2-7）．

1つのプロセッサのみに接続された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は，そのプロセッサのみを割付け可能プロセッサとするクラスに属していなければならない．

【TOPPERS新世代カーネル統合仕様との関係】

割込みサービスルーチンが異なるプロセッサで同時に実行されることはないこととした．

## 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】．

(a-1) NMI以外にカーネル管理外の割込みを設けない

(a-2) カーネル構築時に特定の割込みをカーネル管理外にすると決める

これら場合には，カーネル管理外とする割込みはカーネル構築時（ターゲット依存部の実装時やカーネルのコンパイル時）に決まるため，カーネル管理外とする割込みをアプリケーション側で設定する必要はない【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】．

(b) カーネル管理外とする割込みをアプリケーションで設定できるようにする

この場合には，カーネル管理外とする割込みの設定は，次の方法で行う．まず，カーネル管理外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ハンドラを定義するAPIにおいて，TA_NONKERNEL属性を指定してはならない【NGKI0654】．また，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】．

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例外を異なるCPU例外であると扱い，CPU例外ハンドラをプロセッサ毎に登録する方法と，CPU例外を複数のプロセッサ（すべてのプロセッサでも良い）で共通のものと扱い，CPU例外ハンドラを複数のプロセッサに対して登録する方法がある【NGKI0307】．どちらの方法を採用するか，または両方の方法を併用するかは，ターゲット定義である【NGKI0661】．

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，アプリケーション要求と，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3つの方法に分類できる【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優先して実行される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，リカバリ処理を行うタスクを，CPU例外を起こしたタスクより優先して実行することができない場合には，この方法でリカバリ処理を行うことはできない【NGKI0311】．

(c) 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(c)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)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，プロセッサ毎のCPU例外に対しては，CPU例外ハンドラ番号の上位ビットにプロセッサのID番号を含める方法を，すべてのプロセッサで共通のCPU例外に対しては，CPU例外ハンドラ番号の上位ビットを0とする方法を基本とする．

リカバリ処理を行うタスクを優先して実行することができない状況［NGKI0311］は，保護機能対応でないカーネルで最高優先度のタスクがCPU例外を起こした場合と，保護機能対応のカーネルでカーネルドメインに属する最高優先度のタスクがCPU例外を起こした場合に限られる．これらの場合にも，リカバリ処理を行うタスクを最高優先度とし，その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場合もあるが，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．

【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．

【TOPPERS新世代カーネル統合仕様との関係】

タスク例外処理ルーチンを廃止したことで，タスク例外処理ルーチンでリカバリ処理を行う方法が使えなくなったため，それに関する記述を削除した．

マルチプロセッサ対応カーネルで，CPU例外を複数のプロセッサで共通のものと扱うことを許すこととした．

### 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）を呼び出すことができる【NGKI0535】．

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，［NGKI0311］の状況を除いては，(b)の方法によるリカバリ処理が可能である【NGKI0317】．

xsns_dpnがtrueを返した場合，そのCPU例外ハンドラからは，xsns_dpn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】．xsns_dpnが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)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(c)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．

【TOPPERS新世代カーネル統合仕様との関係】

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）を廃止した．

### 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エミュレートされたCPU例外ハンドラと呼び，CPU例外ハンドラと同等に扱うものとする【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は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，ゼロ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（ゼロ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，最初の通知時刻が同じである複数の周期通知の通知順序は，同じシステム時刻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，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，マスタプロセッサによって実行される【NGKI0340】．ローカル初期化ルーチンは，グローバル初期化ルーチンの実行が完了した後に，それが属するクラスの初期割付けプロセッサによって実行される【NGKI0341】．

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，ディスパッチ保留状態が解除され），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2-8に示す【NGKI0344】．

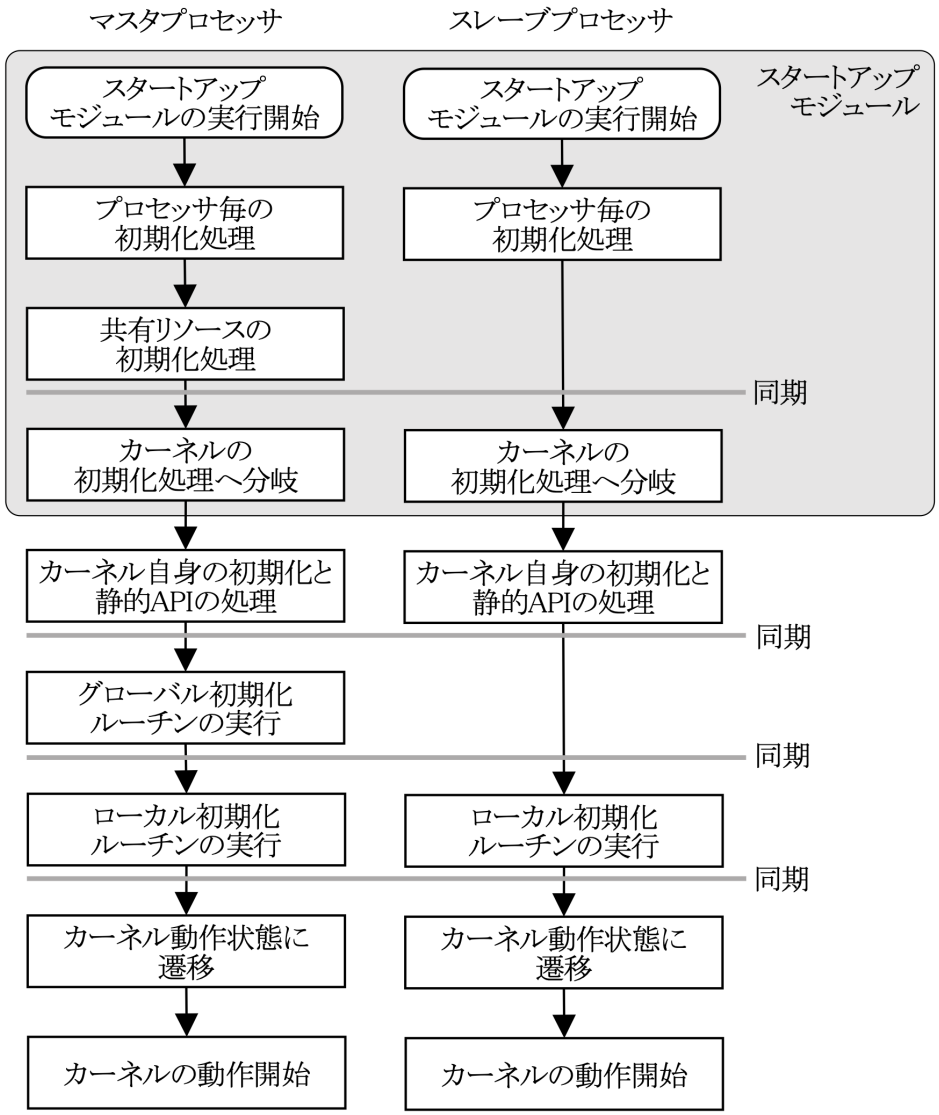


**図2-8. マルチプロセッサ対応カーネルにおけるシステム初期化の流れ**

【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．

マルチプロセッサ対応カーネルでは，カーネルを終了させるサービスコール（ext_ker）は，どのプロセッサからでも呼び出すことができる【NGKI0346】．1つのプロセッサでカーネルを終了させるサービスコールを呼び出すと，そのプロセッサがカーネル動作状態から非動作状態に遷移した後，他のプロセッサ

に対してカーネル終了処理の開始を要求する【NGKI0347】．複数のプロセッサから，カーネルを終了させるサービスコール（ext_ker）を呼び出してもよい【NGKI0348】．

次に，静的API（ATT_TER）により登録した終了処理ルーチンが，システムコンフィギュレーションファイルにおける静的APIの記述順と逆の順序で実行される【NGKI0349】．

マルチプロセッサ対応カーネルでは，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は，すべてのプロセッサがカーネル非動作状態に遷移した後に，それが属するクラスの初期割付けプロセッサによって実行される【NGKI0350】．グローバル終了処理ルーチンは，すべてのプロセッサでローカル終了処理ルーチンの実行が完了した後に，マスタプロセッサによって実行される【NGKI0351】．

以上が終了すると，ターゲットシステム依存の終了処理が呼び出される【NGKI0352】．ターゲットシステム依存の終了処理は，カーネルの管理外であり，アプリケーションで用意するのが基本であるが，カーネルの配布パッケージの中に，ターゲットシステム毎に標準的なルーチンが用意されている【NGKI0353】．標準のターゲットシステム依存の終了処理では，ソフトウェア環境（ライブラリなど）依存の終了処理フックを呼び出す【NGKI0354】．

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で，ターゲットシステム依存の終了処理が呼び出される【NGKI0355】．マルチプロセッサ対応カーネルにおけるシステム終了処理の流れと，各プロセッサが同期を取るタイミングを，図2-9に示す【NGKI0356】．

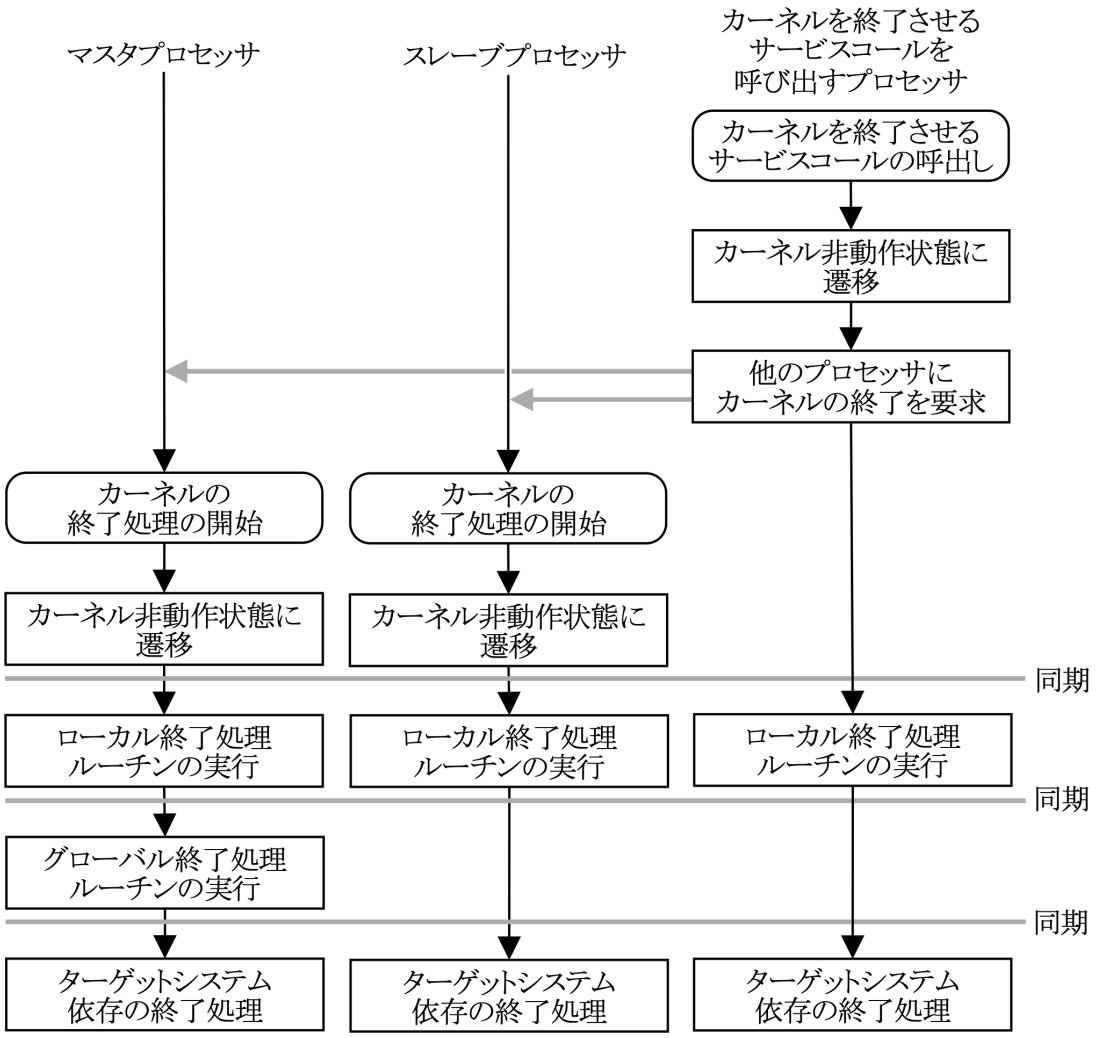


**図2-9. マルチプロセッサ対応カーネルにおけるシステム終了処理の流れ**

【使用上の注意】

マルチプロセッサ対応カーネルで，あるプロセッサからカーネルを終了させるサービスコール（ext_ker）を呼び出しても，他のプロセッサがカーネル動作状態で割込みをマスクしたまま実行し続けると，カーネルが終了しない．

プロセッサが割込みをマスクしたまま実行し続けないようにするのは，アプリケーションの責任である．例えば，ある時間を超えて割込みをマスクしたまま実行し続けていないかを，ウォッチドッグタイマを用いて監視する方法が考えられる．割込みをマスクしたまま実行し続けていた場合には，そのプロセッサからもカーネルを終了させるサービスコール（ext_ker）を呼び出すことで，カーネルを終了させることができる．

【μITRON4.0仕様との関係】

μITRON4.0仕様には，システム終了に関する規定はない．

## 2.10 オブジェクトの登録とその解除

### 2.10.1 ID番号で識別するオブジェクト

ID番号で識別するオブジェクトは，オブジェクトを生成する静的API（CRE_YYY）またはサービスコール（acre_yyy）によってカーネルに登録する【NGKI0357】．

オブジェクトを生成する静的API（CRE_YYY）は，生成するオブジェクトにID番号を割り付け，オブジェクトの識別名を割り付けたID番号にマクロ定義する【NGKI0359】．同じ識別名のオブジェクトが生成済みの場合には，E_OBJエラーとなる【NGKI0360】．

オブジェクトを生成するサービスコール（acre_yyy）は，割付け可能なID番号の数を指定する静的API（AID_YYY）によって確保されたID番号の中から，使用されていないID番号を1つ選び，生成するオブジェクトに割り付ける【NGKI0361】．割り付けたID番号は，サービスコールの返値としてアプリケーションに通知する【NGKI0362】．使用されていないID番号が残っていない場合には，E_NOIDエラーとなる【NGKI0363】．

割付け可能なID番号の数を指定する静的API（AID_YYY）は，システムコンフィギュレーションファイル中に複数記述することができる【NGKI0364】．その場合，各静的APIで指定した数の合計の数のID番号が確保される【NGKI0365】．

オブジェクトを生成するサービスコール（acre_yyy）によって登録したオブジェクトは，オブジェクトを削除するサービスコール（del_yyy）によって登録を解除することができる【NGKI0366】．登録解除したオブジェクトのID番号は，未使用の状態に戻され，新たに生成するオブジェクトに割り付けられる【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，新たに生成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の実行が途中で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，動的生成対応でないカーネルでは，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，ID番号で識別するオブジェクトを，オブジェクトを追加する静的API（ATT_YYY）によってカーネルに登録する機能を廃止した．オブジェクトを生成する静的APIによって登録したオブジェクトは，登録解除できないこととした．

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

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

【μITRON4.0/PX仕様との関係】

アクセス許可ベクタを指定してオブジェクトを生成する静的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．

【TOPPERS新世代カーネル統合仕様との関係】

ID番号で識別するオブジェクトを，オブジェクトを追加する静的API（ATT_YYY）によってカーネルに登録する機能を廃止した．

【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，動的生成対応でないカーネルでは，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】

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

### 2.10.3 識別番号を持たないオブジェクト

識別する必要がないために，識別番号を持たないオブジェクトは，オブジェクトを追加する静的API（ATT_YYY）によってカーネルに登録する【NGKI0604】．

### 2.10.4 オブジェクト生成に必要なメモリ領域

オブジェクトを生成する際に必要なメモリ領域の内，サイズが一定のものについては，コンフィギュレータによって，オブジェクトの管理ブロックなどの形で確保される【NGKI0665】．

それに対して，オブジェクトを生成する際に，サイズが一定でないメモリ領域を必要とする場合には，オブジェクトを生成する静的APIおよびサービスコールに，使用するメモリ領域の先頭番地を渡すパラメータを設ける【NGKI0385】．

静的APIにおいて，使用するメモリ領域の先頭番地を渡すパラメータをNULLとした場合，必要なメモリ領域は，コンフィギュレータによって確保される【NGKI0386】．

保護機能対応カーネルでは，コンフィギュレータによって確保されるメモリ領域は，別に規定がない限りは，カーネルドメイン向け（マルチプロセッサ対応カーネルの場合には，さらに，オブジェクトが属するクラス向け）の標準RAMリージョン内に確保される【NGKI0649】．

サービスコールにおいて，使用するメモリ領域の先頭番地を渡すパラメータをNULLとした場合，必要なメモリ領域は，カーネルによって，別に規定がない限りは，カーネルメモリプール領域から確保される【NGKI0618】．保護機能対応カーネルまたはマルチプロセッサ対応カーネルで，どのカーネルメモリプール領域が使用されるかについては，「4.13 システム構成管理機能」の節を参照すること．

【TOPPERS新世代カーネル統合仕様との関係】

コンフィギュレータとカーネルが，オブジェクト生成に必要なメモリ領域をどこに確保するかを規定した．

### 2.10.5 オブジェクトが属する保護ドメインの設定

保護機能対応カーネルにおいて，オブジェクトが属する保護ドメインは，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【NGKI0387】．

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保護ドメインの囲みの中に記述する［NGKI0459］．無所属のオブジェクトを登録する静的APIは，保護ドメインの囲みの外に記述する［NGKI0460］（「2.12.3　保護ドメインの指定」の節を参照）．

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_DOM(domid)を指定することにより，オブジェクトを属させる保護ドメインを設定する【NGKI0390】．ここでdomidは，そのオブジェクトを属させる保護ドメインのID番号であり，TDOM_KERNEL（＝-1）を指定することでカーネルドメインに属させることができる．また，domidにTDOM_SELF（＝0）を指定するか，オブジェクト属性にTA_DOM(domid)を指定しないことで，自タスクが属する保護ドメインに属させることが

できる．さらに，無所属のオブジェクトを登録する場合には，domidにTDOM_NONE（＝-2）を指定する．

ただし，特定の保護ドメインのみに属することができるオブジェクトを登録するサービスコールの中には，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設定する必要がないものもある【NGKI0391】．

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，AID_YYYを保護ドメインの囲みの中に記述した場合には，その保護ドメインに属するオブジェクトに，保護ドメインの囲みの外に記述した場合には，無所属のオブジェクトに割り付けられる【NGKI0610】．

【補足説明】

この仕様では，カーネルオブジェクトが属する保護ドメインを参照する機能は用意していない．

【TOPPERS新世代カーネル統合仕様との関係】

割付け可能なID番号は保護ドメイン毎に確保することにし，AID_YYYを保護ドメインの囲みの中にも記述できることとした．

【仕様決定の理由】

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，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】．

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［NGKI0470］．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［NGKI0397］（「2.12.4 クラスの指定」の節を参照）．

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，AID_YYY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】．AID_YYY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が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】

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を参照するサービスコール（get_inf）は，この原則に対する例外の一つである．

## 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．

【μITRON4.0/PX仕様との関係】

システム時刻に対するアクセス許可ベクタと，それを設定する静的APIおよびサービスコールを廃止し，保護ドメインに対するアクセス許可ベクタと，それを設定する静的APIおよびサービスコールを追加した．また，システム状態に対して，アクセス許可ベクタを2つ持つこととした．

【TOPPERS新世代カーネル統合仕様との関係】

保護ドメインに対するアクセス許可ベクタと，それを設定する静的APIおよびサービスコールを追加した．また，システム状態に対して，アクセス許可ベクタを2つ持つこととした．

### 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や，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操作，オブジェクトを定義するサービスコールについては，スケジューリングに関係するものは保護ドメインに対するアクセス許可ベクタによって，その他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【NGKI0568】．また，オブジェクトを生成するサービスコールは，生成するオブジェクトが属する保護ドメインに対するアクセス許可ベクタによって保護される【NGKI0612】．

アクセス許可ベクタ等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，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，アクセス許可ベクタによっ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，アクセス許可ベクタまたはメモリオブジェクト属性によっ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（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．

メモリアクセス違反ハンドラで，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）を参照する方法が，ターゲット定義で用意されている【NGKI0414】．

メモリオブジェクトに対して，サービスコールを通じて，アクセス許可ベクタまたはメモリオブジェクト属性によっ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【NGKI0412】．ただし，ターゲット定義で，E_MACVエラーが返るのに代えて，メモリアクセス違反ハンドラが起動される場合がある【NGKI0413】．

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】．

ユーザタスクから，サービスコールを通じて，メモリオブジェクトの境界を越えるメモリ領域に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【NGKI0673】．ただし，ターゲット定義で，このエラーチェックが省略される場合がある【NGKI0674】．

ユーザタスクから，サービスコールを通じて，アラインされていないメモリ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【NGKI0670】．ただし，ターゲット定義で，E_MACVエラーが返るのに代えて，ターゲット定義のCPU例外ハンドラが起動される場合がある【NGKI0671】

【補足説明】

以下の状況における振舞いは，この仕様では規定されず，ターゲットに依存する．

- 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，カーネルドメインからアクセスしようとした場合
- カーネルドメインに属する処理単位から，サービスコールを通じて，アラインされていないメモリアクセスを行おうとした場合

【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．

オブジェクトを生成するサービスコールの保護は，生成するオブジェクトが属する保護ドメインに対するアクセス許可ベクタによって行うものとした．

サービスコールを通じて，メモリオブジェクトの境界を越えるメモリ領域にアクセスを行おうとした場合と，アラインされていないメモリアクセスを行おうとした場合の振舞い明記した．

【TOPPERS新世代カーネル統合仕様との関係】

オブジェクトを生成するサービスコールの保護は，生成するオブジェクトが属する保護ドメインに対するアクセス許可ベクタによって行うものとした．これに伴って，オブジェクトを生成する際に使用するリソース（ID番号，カーネルメモリプール領域）の確保量についても，保護ドメイン毎に設定することとした．

サービスコールを通じて，メモリオブジェクトの境界を越えるメモリ領域にアクセスを行おうとした場合と，アラインされていないメモリアクセスを行おうとした場合の振舞い明記した．

【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．

(a)　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．

(b)　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．

(c) 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．

(d)　　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．

(e)　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，通常操作1，通常操作2，参照操作の3つの種別のアクセスが，その保護ドメインのみに許可される【NGKI0613】．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，上記の3つのアクセス種別のアクセス許可パターンが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，上記の3つのアクセス種別のアクセス許可パターンが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．管理操作のアクセス許可パターンは，そのカーネルオブジェクトが属する保護ドメインに対する通常操作1のアクセス許可パターンと同じ値に設定される【NGKI0614】．

無所属のカーネルオブジェクトに対しては，通常操作1，通常操作2，参照操作の3つの種別のアクセスが，すべての保護ドメインに許可される【NGKI0615】．すなわち，上記の3つのアクセス種別のアクセス許可パターンが，TACP_SHAREDに設定される．管理操作のアクセス許可パターンは，無所属に対する通常操作1のアクセス許可パターンと同じ値に設定される【NGKI0616】．

保護ドメインのアクセス許可ベクタは，通常操作1，通常操作2，参照操作はその保護ドメインのみに許可され，管理操作はカーネルドメインのみに許可される【NGKI0569】．すなわち，カーネルドメインに対しては，4つのアクセス許可パターンがいずれもTACP_KERNELに，ユーザドメインに対しては，通常操作1，通常操作2，参照操作のアクセス許可パターンがTACP(domid)に，管理操作のアクセス許可パターンがTACP_KERNELに設定される．

無所属に対するアクセス許可ベクタは，通常操作1，通常操作2，参照操作はすべての保護ドメインに許可され，管理操作はカーネルドメインのみに許可される【NGKI0617】．すなわち，通常操作1，通常操作2，参照操作のアクセス許可パターンがTACP_SHAREDに，管理操作のアクセス許可パターンがTACP_KERNELに設定される．

システム状態1とシステム状態2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，8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．

【μITRON4.0/PX仕様，TOPPERS新世代カーネル統合仕様との関係】

保護ドメイン以外のカーネルオブジェクトに対する管理操作のアクセス許可パターンを，そのカーネルオブジェクトが属する保護ドメインに対する通常操作1（無所属のカーネルドメイン場合は，無所属に対する通常操作1）のアクセス許可パターンと同じ値に設定されるものとした．

### 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，保護ドメインのアクセス許可ベクタを設定する静的API（ACV_DOM），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【NGKI0570】．

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，保護ドメインのアクセス許可ベクタを設定するサービスコール（sac_dom），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【NGKI0571】．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，オブジェクトを追加する静的API（ATT_YYY）によって，登録と同時にアクセス許可ベクタを設定することができる【NGKI0605】．アクセス許可ベクタの記述を省略した場合には，デフォルトのアクセス許可ベクタが設定される【NGKI0606】．

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．

【TOPPERS新世代カーネル統合仕様との関係】

メモリオブジェクトに対しては，オブジェクトを追加する静的API（ATT_YYY）によって，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ことができるようにし，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）は廃止した．

## 2.11.5 カーネルの管理領域のアクセス保護

カーネルオブジェクトの生成に必要なメモリ領域（「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）の中で，カーネルの内部およびカーネルドメインに属する処理単位のみからアクセスされるものを，カーネルの管理領域と呼ぶ．

ユーザドメインに属する処理単位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】．

カーネルの管理領域として，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】．

【補足説明】

以下のメモリ領域が，カーネルの管理領域に該当する．

- 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
- データキュー管理領域
- 優先度データキュー管理領域
- メッセージバッファ管理領域
- 固定長メモリプール管理領域
- 非タスクコンテキスト用のスタック領域
- カーネルメモリプール領域

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，ユーザドメインに属する処理単位からもアクセスされるため，カーネルの管理領域には該当しない．

## 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，通常操作1（書込みアクセス）と通常操作2（読出し・実行アクセス）に対するアクセス許可パターンは意味を持たない【NGKI0439】．管理操作と参照操作に対するアクセス許可パターンは，タスクと同じ値に設定される【NGKI0609】．また，ユーザスタック領域に対する実行アクセスは，禁止する【NGKI0663】．ただし，ターゲットハードウェアの制約によって実行アクセスを禁止できない場合には，ターゲット定義で，実行アクセスを禁止しないこととする

【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，その文法は次の通りである【NGKI0619】.

```
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，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】．コンフィギュレータが生成するファイルでは，インクルードディレクティブは先頭に集められる【NGKI0447】.

C言語プリプロセッサの条件ディレクティブは，有効とする静的APIとC言語プリプロセッサのディレクティブを選択するために用いることができる【NGKI0620】．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，そのタスクにアクセス許可ベクタを設定する静的APIが記述されていた場合，アクセス許可ベクタを設定する静的APIがE_NOEXSエラーとなる.

【使用上の注意】

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，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ため，インクルードするC言語のヘッダファイル中でマクロの定義を変更（定義を解除した後に，定義し直す）した場合，意図しない動作をする可能性がある.

C言語プリプロセッサの条件ディレクティブは，保護ドメインの囲みを無効にしない．そのため，条件ディレクティブで囲みが選択されなかった保護ドメインに対しても，保護ドメインIDが割り付けられる．

【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．

【TOPPERS新世代カーネル統合仕様との関係】

C言語プリプロセッサの条件ディレクティブは，有効とするC言語プリプロセッサのディレクティブの選択にも用いることができるものとした．

### 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．

(a) オブジェクト識別名

オブジェクトのID番号を表す識別名を指定するパラメータ．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】．

(b) 整数定数式パラメータ

オブジェクト番号や機能コード，オブジェクト属性，サイズや数，優先度など，整数値を指定するパラメータ．プログラムが配置される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】．

(c) 一般定数式パラメータ

処理単位のエントリ番地，メモリ領域の先頭番地，拡張情報など，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．

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【NGKI0458】．

NULLを指定することが特別な意味を持つ一般定数式パラメータにおいて，NULLの意味を持たせたい場合には，静的APIのパラメータに「NULL」と記述しなければならない【NGKI0638】．NULLに展開されるマクロを記述しても，NULLの意味にはならないので注意すること．

(d) 文字列パラメータ

オブジェクトモジュール名やセクション名など，文字列を指定するパラメータ．任意の文字列を，C言語の文字列の記法で記述することができる．

【μITRON4.0仕様との関係】

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

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

この仕様の(a)が，おおよそμITRON4.0仕様の(A)に相当するが，(a)には整数値を記述できない点が異なる．(b)～(c)と(B)～(D)の間には単純な対応関係がないが，記述できる定数式の範囲には，(B)⊂(C)⊂(b)⊂(c)＝(D)の関係がある．

μITRON4.0仕様では，静的APIのパラメータは基本的には(D)とし，コンフィギュレータが値を知る必要があるパラメータを(B)，構成・初期化ファイルに生成するC言語プリプロセッサの条件ディレクティブ（#if）中に含めたい可能性のあるパラメータを(C)としていた．

それに対して，この仕様におけるコンフィギュレータの処理モデル（「2.12.5 コンフィギュレータの処理モデル」の節を参照）では，コンフィギュレータのパス2において定数式パラメータの値を知ることができるため，(B)～(D)の区別をする必要がない．そのため，静的APIのパラメータは基本的には(b)とし，パス2で値を知ることのできない定数式パラメータのみを(c)としている．

## 2.12.3 保護ドメインの指定

保護機能対応カーネルでは，オブジェクトを登録する静的API等を，そのオブジェクトを属させる保護ドメインの囲みの中に記述する【NGKI0459】．無所属のオブジェクトを登録する静的APIは，保護ドメインの囲みの外に記述する【NGKI0460】．保護ドメインに属すべきオブジェクトを登録する静的API等を，保護ドメインの囲みの外に記述した場合には，コンフィギュレータがE_RSATRエラーを報告する【NGKI0461】．

ユーザドメインの囲みの文法は次の通り【NGKI0462】．

```
DOMAIN(保護ドメイン名) {
    ユーザドメインに属するオブジェクトを登録する静的API等
}
```

保護ドメイン名には，ユーザドメイン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

【NGKI0463】.

コンフィギュレータは，ユーザドメインの囲みを処理する際に，ユーザドメインに保護ドメインIDを割り付け，構成・初期化ヘッダファイルに，指定された保護ドメイン名を割り付けた保護ドメインID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【NGKI0464】．また，ユーザドメインの囲みの中およびそれ以降に記述する静的APIの整数定数式パラメータの記述に保護ドメイン名を記述すると，割り付けた保護ドメインIDの値に評価される【NGKI0465】.

ユーザドメインの囲みの中を空にすることで，ユーザドメインへの保護ドメインIDの割付けのみを行うことができる【NGKI0466】.

カーネルドメインの囲みの文法は次の通り【NGKI0467】.

```
KERNEL_DOMAIN {
    カーネルドメインに属するオブジェクトを登録する静的API等
}
```

同じ保護ドメイン名を指定したユーザドメインの囲みや，カーネルドメインの囲みを，複数回記述してもよい【NGKI0468】．保護機能対応でないカーネルで保護ドメインの囲みを記述した場合や，保護ドメインの囲みの中に保護ドメインの囲みを記述した場合には，コンフィギュレータがエラーを報告する【NGKI0469】.

【μITRON4.0/PX仕様との関係】

保護ドメインの囲みの文法を変更した.

【仕様決定の理由】

保護ドメインに属すべきオブジェクトを登録する静的API等を保護ドメインの囲みの外に記述した場合のエラーコードをE_RSATRとしたのは，オブジェクトを動的に登録するAPIにおいては，オブジェクトの属する保護ドメインを，オブジェクト属性によって指定するためである.

### 2.12.4 クラスの指定

マルチプロセッサ対応カーネルでは，オブジェクトを登録する静的API等を，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【NGKI0470】．クラスに属さないオブジェクトを登録する静的API等は，クラスの囲みの外に記述する【NGKI0397】．クラスに属すべきオブジェクトを登録する静的API等を，クラスの囲みの外に記述した場合や，クラスに属さないオブジェクトを登録する静的API等を，クラスの囲みの中に記述した場合には，コンフィギュレータがE_RSATRエラーを報告する【NGKI0471】.

クラスの囲みの文法は次の通り【NGKI0472】.

```
CLASS(クラスID) {
    クラスに属するオブジェクトを登録する静的API等
}
```

クラスIDには，静的APIの整数定数式パラメータと同等の定数式を記述することができる【NGKI0473】．使用できないクラスIDを指定した場合には，コンフィギュレータがE_IDエラーを報告する【NGKI0474】.

同じクラスIDを指定したクラスの囲みを複数回記述してもよい【NGKI0475】．マルチプロセッサ対応でないカーネルでクラスの囲みを記述した場合や，クラスの囲みの中にクラスの囲みを記述した場合には，コンフィギュレータがエラーを報告する【NGKI0476】.

なお，保護機能とマルチプロセッサの両方に対応するカーネルでは，保護ドメインの囲みとクラスの囲みはどちらが外側になっていてもよい【NGKI0477】．

【仕様決定の理由】

クラスに属すべきオブジェクトを登録する静的API等をクラスの囲みの外に記述した場合のエラーコードをE_RSATRとしたのは，オブジェクトを動的に登録するAPIにおいては，オブジェクトの属するクラスを，オブジェクト属性によって指定するためである．

## 2.12.5 コンフィギュレータの処理モデル

コンフィギュレータは，次の3つないしは4つのパスにより，システムコンフィギュレーションファイルを解釈し，構成・初期化情報を含むファイルなどを生成する（図2-10）．

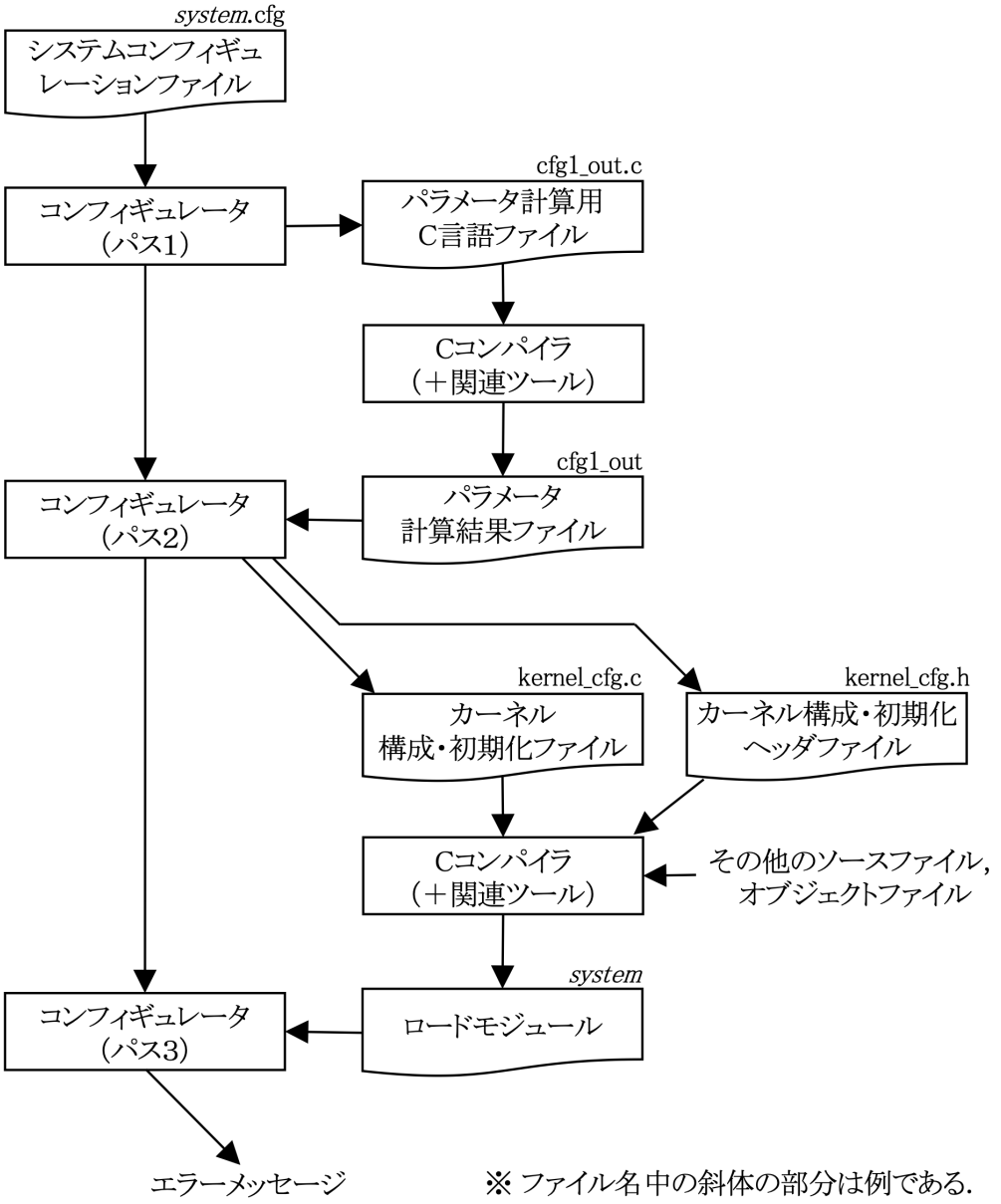


**図2-10. コンフィギュレータの処理モデル**

最初のパス1では，システムコンフィギュレーションファイルを解釈し，そこに含まれる静的APIの整数定数式パラメータの値をCコンパイラを用いて求めるために，パラメータ計算用C言語ファイル（cfg1_out.c）を生成する．この時，システムコンフィギュレーションファイルに含まれるC言語プリプロセッサのインクルードディレクティブは，パラメータ計算用C言語ファイルの先頭に集めて生成する．また，条件ディレクティブは，順序も含めて，そのままの形でパラメータ計算用C言語ファイルに出力す

る．システムコンフィギュレーションファイルに文法エラーや未サポートの記述があった場合，オブジェクトの識別名に重複があった場合，ユーザドメインの数が制限を超えた場合には，この段階でエラーが検出される．

次に，Cコンパイラおよび関連ツールを用いて，パラメータ計算用C言語ファイルをコンパイルし，パラメータの計算結果を含むロードモジュール（パラメータ計算結果ファイル，cfg1_out）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）と構成・初期化ヘッダファイル（kernel_cfg.h）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（動的生成対応でないカーネルでは，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やオブジェクト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．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2で生成されたロードモジュールから，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で自動メモリ配置の場合には，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2-11）．

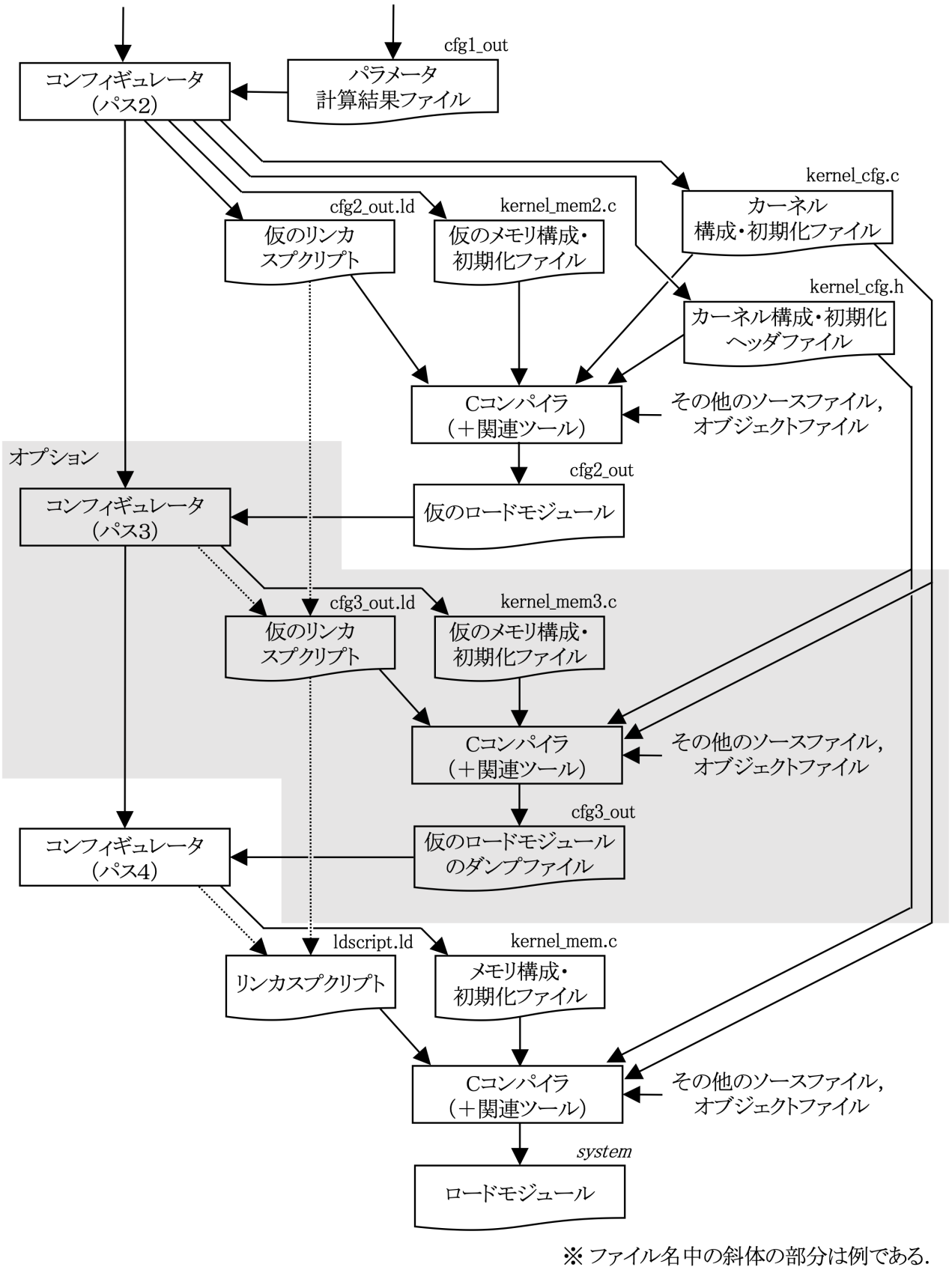


**図2-11. 保護機能対応カーネルにおけるコンフィギュレータの処理モデル**

コンフィギュレータは，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，リンカスクリプト（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カスクリプト（cfg2_out.ld）を生成し，これらを用いて仮のロードモジュール（cfg2_out）を生成する．

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）と仮のリンカスクリプト（cfg3_out.ld）を生成する．また，これらを用いて仮のロードモジュール（cfg3_out）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）とリンカスクリプト（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．

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．

なお，コンフィギュレータがロードモジュールから情報を取り出す際には，ロードモジュールからダンプファイル（Sレコードフォーマットなどファイル）とシンボルファイル（ロードモジュール中の各シンボルとアドレスの対応表を含むファイル）を生成し，それらのファイルを読み込む．

【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」という記述は，コンフィギュレータがそのエラーを検出することを意味する．

ただし，エラーの種類によっては，サービスコールと同等のエラー検出を行うことが難しいため，そのようなものについては例外とする【NGKI0479】．例えば，メモリ不足をコンフィギュレータによって検出するのは容易ではない．

逆に，オブジェクト属性については，サービスコールより強力なエラーチェックを行える可能性がある．例えば，タスク属性にTA_STAと記述されている場合，サービスコールではエラーを検出できないが，コンフィギュレータでは検出できる可能性がある．ただし，このようなエラー検出を完全に行おうとするとコンフィギュレータが複雑になるため，このようなエラーを検出することは必須とせず，検出できた場合には警告メッセージを出力する【NGKI0480】．

なお，コンフィギュレータは，この仕様で警告メッセージを出力すると規定されている以外の状況でも，警告メッセージを出力する場合がある【NGKI0641】．

【μITRON4.0仕様との関係】

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

### 2.12.7 オブジェクトのID番号の指定

アプリケーション設計者がオブジェクトのID番号を指定するために，コンフィギュレータのオプション機能として，以下の機能が用意されている．

コンフィギュレータのオプション指定により，オブジェクト識別名とID番号の対応表を含むファイルを渡すと，コンフィギュレータはそれに従ってオブジェクトにID番号を割り付ける【NGKI0481】．それに従ったID番号割付けができない場合（ID番号に抜けができる場合など）には，コンフィギュレータはエラーを報告する【NGKI0482】．

またコンフィギュレータは，オプション指定により，オブジェクト識別名とコンフィギュレータが割り付けたID番号の対応表を含むファイルを，コンフィギュレータに渡すファイルと同じフォーマットで生成する【NGKI0483】．

【μITRON4.0仕様との関係】

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

## 2.13 TOPPERSネーミングコンベンション

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

(1) モジュール識別名

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

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

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

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

(2) データ型名

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

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

データ型の種類毎に，次のネーミングコンベンションを定める．

(A) パケットのデータ型

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

(3) 関数名

関数の名称は，英小文字，数字，"_"で構成する．

関数の種類毎に，次のネーミングコンベンションを定める．

(A) サービスコール

サービスコールは，xxx_yyyまたはwww_xxx_yyyの名称とする．ここで，xxxは操作の方法，yyyは操作の対象を表す．xxx_yyyまたはwww_xxx_yyyから派生したサービスコールは，それぞれzxxx_yyyまたはwww_zxxx_yyyの名称とする．ここでzは，派生したことを表す文字である．派生したことを表す文字を2つ付加する場合には，zzxxx_yyyまたはwww_zzxxx_yyyの名称となる．

非タスクコンテキストから呼び出せるサービスコールは，サービスコールの名称に"i"を付加した，ixxx_yyy，izxxx_yyy，www_ixxx_yyy，www_izxxx_yyyといった名称で呼び出すこともできる【NGKI0562】．

【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．

【μITRON4.0仕様，μITRON4.0/PX仕様との関係】

「2.5.2　タスクコンテキストと非タスクコンテキスト」の節で述べた通り，非タスクコンテキストからもタスクコンテキストと同じ名称のサービスコールを呼び出すこととしたが，過去との互換性のために，"i"を付加した名称で呼び出すこともできるものとした．

(B) コールバック

コールバックの名称は，サービスコールのネーミングコンベンションに従う．

(4) 変数名

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

変数の名称に関して，次のガイドラインを設ける．

| | |
|---|---|
| ～id | ～ID（オブジェクトのID番号，ID型） |
| ～no | ～番号（オブジェクト番号） |
| ～atr | ～属性（オブジェクト属性，ATR型） |
| ～stat | ～状態（オブジェクト状態，STAT型） |

| | |
|---|---|
| ～mode | ～モード（サービスコールの動作モード，MODE型） |
| ～pri | ～優先度（優先度，PRI型） |
| ～sz | ～サイズ（単位はバイト数，size_t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 |

### (5) 定数名

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

定数の種類毎に，次のネーミングコンベンションを定める．

(A) メインエラーコード

メインエラーコードは，先頭が"E_"である名称とする．

(B) 機能コード

| | |
|---|---|
| TFN_XXX_YYY | xxx_yyyの機能コード |
| TFN_WWW_XXX_YYY | www_xxx_yyyの機能コード |

(C) その他の定数

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

また，定数の名称に関して，次のガイドラインを設ける．

| | |
|---|---|
| TA_〜 | オブジェクトの属性値 |
| TSZ_〜 | 〜のサイズ |
| TBIT_〜 | 〜のビット数 |
| TMAX_〜 | 〜の最大値 |
| TMIN_〜 | 〜の最小値 |

### (6) マクロ名

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

マクロの種類毎に，次のネーミングコンベンションを定める．

(A) 構成マクロ

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

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

### (7) 静的API名

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

### (8) ファイル名

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

ファイルの種類毎に，次のネーミングコンベンションを定める．

(A) ヘッダファイル

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

### (9) モジュール内部の名称の衝突回避

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

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

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

## 2.14 TOPPERS共通定義

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

### 2.14.1 TOPPERS共通ヘッダファイル

TOPPERS共通定義（共通データ型，共通定数，共通マクロ）は，TOPPERS共通ヘッダファイル（t_stddef.h）およびそこからインクルードされるファイルに含まれている【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】．

### 2.14.2 TOPPERS共通データ型

C90に規定されているデータ型以外で，TOPPERSソフトウェアで共通に用いるデータ型は次の通りである（一部，C90に規定されているものも含む）【NGKI0542】．

| | |
|---|---|
| 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 |
| size_t | メモリ領域のサイズを表す符号無し整数（C90準拠） |
| 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） |
| TMO | タイムアウト指定（符号無し32ビット整数，単位はマイクロ秒，uint32_tに定義） |
| EXINF | 拡張情報（ポインタまたは32ビット以上のサイズの整数を格納できるサイズの型） |
| RELTIM | 相対時間（符号無し32ビット整数，単位はマイクロ秒，uint32_tに定義） |
| SYSTIM | システム時刻（符号無し整数，単位はマイクロ秒，uint64_tまたはuint32_tに定義） |
| PRCTIM | プロセッサ時間（符号無し整数，単位はマイクロ秒，uint32_tに定義） |
| HRTCNT | 高分解能タイマのカウント値（符号無し整数，uint32_tまたはuint64_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．

メモリ領域のサイズを表すデータ型として，SIZEに代えて，C90準拠のsize_tを用いることにした．

データタイプが定まらないものへのポインタを表すデータ型である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準拠の倍精度浮動小数点数を表す型としてdouble64_tを導入した．

高分解能タイマのカウント値のデータ型としてHRTCNT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．

【TOPPERS新世代カーネル統合仕様との関係】

メモリ領域のサイズを表すデータ型として，SIZEに代えて，C90準拠のsize_tを用いることにした．

性能評価用システム時刻のためのデータ型であるSYSUTMを廃止し，高分解能タイマのカウント値のデータ型であるHRTCNTを追加した．

## 2.14.3 TOPPERS共通定数

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

(1) 一般定数【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） |
| SIZE_MAX | size_tに格納できる最大値（C99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【NGKI0555】

| | | |
|---|---|---|
| TMO_POL | 0U | ポーリング |
| TMO_FEVR | UINT32_MAX | 永久待ち |
| TMO_NBLK | UINT32_MAX-1 | ノンブロッキング |

(6) アクセス許可パターン［NGKI0089］

| | | |
|---|---|---|
| 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 |

### 2.14.4 TOPPERS共通エラーコード

TOPPERSソフトウェアで共通に用いるメインエラーコードは次の通りである【NGKI0540】.

(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_RASTER | -53 | タスクの終了要求 |

(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．

(I) 通信エラークラス（EC_COMM，-65〜-72）

| | | |
|---|---|---|
| E_COMM | -65 | 通信エラー |

このエラークラスに属するエラーコードは，通信エラー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】**

通信エラークラスは，μITRON4.0仕様に規定されていないエラークラスである．E_NORES, E_RASTER，E_COMMは，μITRON4.0仕様に規定されていないエラーコードである．

**【TOPPERS新世代カーネル統合仕様との関係】**

通信エラークラスは，TOPPERS新世代カーネル統合仕様に規定されていないエラークラスである．
E_RASTER，E_COMMは，TOPPERS新世代カーネル統合仕様に規定されていないエラーコードである．

## 2.14.5 TOPPERS共通マクロ

**【μITRON4.0仕様，μITRON4.0/PX仕様との関係】**

μITRON4.0仕様では，エラーコード構成・分解マクロ（ERCD，MERCD，SERCD）のみが定義されてお

り，μITRON4.0/PX仕様で，アクセス許可パターン構成マクロ（TACP）が追加定義されている．他は追加定義したものである．

### (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［NGKI0015］

| | |
|---|---|
| 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［NGKI0088］

| | |
|---|---|
| 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を指定した場合，どのようなアクセス許可パターンが構成されるかは保証されない【NGKI0505】．

### 2.14.6 TOPPERS共通構成マクロ

(1) 相対時間の範囲【NGKI0557】

| | |
|---|---|
| TMAX_RELTIM | 相対時間に指定できる最大値（＝4,000,000,000） |

## 2.15 カーネル共通定義

カーネルの複数の機能で共通に用いる定義を，カーネル共通定義と呼ぶ．

### 2.15.1 カーネルヘッダファイル

カーネルを用いるために必要な定義は，カーネルヘッダファイル（kernel.h）およびそこからインクルードされるファイルに含まれている【NGKI0507】．カーネルを用いる場合には，カーネルヘッダファイルをインクルードする【NGKI0508】．

ただし，カーネルを用いるために必要な定義の中で，コンフィギュレータによって生成されるものは，カーネル構成・初期化ヘッダファイル（kernel_cfg.h）に含まれる【NGKI0509】．具体的には，登録できるオブジェクトの数（TNUM_YYY）やオブジェクトのID番号などの定義が，これに該当する．これらの定義を用いる場合には，カーネル構成・初期化ヘッダファイルをインクルードする【NGKI0510】．

μITRON4.0仕様で規定されており，この仕様で廃止されたデータ型および定数を用いる場合には，ITRON仕様互換ヘッダファイル（itron.h）をインクルードする【NGKI0511】．

【μITRON4.0仕様との関係】

この仕様では，コンフィギュレータが生成するヘッダファイルに，オブジェクトのID番号の定義に加えて，登録できるオブジェクトの数（TNUM_YYY）の定義が含まれることとした．これに伴い，ヘッダファイルの名称を，μITRON4.0仕様の自動割付け結果ヘッダファイル（kernel_id.h）から，カーネル構成・初期化ヘッダファイル（kernel_cfg.h）に変更した．

## 2.15.2 カーネル共通定数

### (1) オブジェクト属性【NGKI0512】

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

【μITRON4.0仕様との関係】

値が0のオブジェクト属性（TA_HLNG，TA_TFIFO，TA_WSGL）は，デフォルトの扱いにして廃止した．これは，「(tskatr & TA_HLNG) != 0U」のような間違いを防ぐためである．TA_ASMは，有効な使途がないために廃止した．メールボックス機能を廃止したため，TA_MPRIとTA_MFIFOは廃止した．

### (2) 保護ドメインID【NGKI0513】

| | | |
|---|---|---|
| TDOM_SELF | 0 | 呼び出した処理単位が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【NGKI0514】

| | | |
|---|---|---|
| TSOM_STP | -1 | システム周期の停止 |
| 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

「2.14.4 TOPPERS共通エラーコード」の節で定義したメインエラーコードの中で，E_CLS，E_WBLK，E_BOVR，E_COMMの4つは，カーネルでは使用しない【NGKI0541】．

【TOPPERS/ASP3カーネルにおける規定】

ASP3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/FMP3カーネルにおける規定】

FMP3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/HRP3カーネルにおける規定】

HRP3カーネルでは，サービスコールから，E_RSATR，E_NOID，E_NORES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_NOIDが返る状況が起こる【HRPS0011】．

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルでは，サービスコールから，E_RSATR，E_NOID，E_NORESが返る状況は起こらない【HRMPS0008】．E_RSATRとE_NORESは，コンフィギュレータによって検出される【HRMPS0009】．

【TOPPERS/SSP3カーネルにおける規定】

SSP3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】．

## 2.15.3 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，サイズszの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(sz)];
```

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，スタック領域のサイズにはROUND_STK_T(sz)を，スタック領域の先頭番地には<スタック領域の変数名>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 |

#### (4) スケジューリング単位番号を構成するマクロ

保護機能とマルチプロセッサのいずれかまたは両方に対応するカーネルでは，スケジューリング単位番号を構成するためのマクロとして，次のマクロを用意している【NGKI0634】.

| | |
|---|---|
| SCHEDNO(prcid, domid) | マルチプロセッサ対応カーネルにおけるスケジューリング単位番号prcidと，保護機能対応カーネルにおけるスケジューリング単位番号domidから，保護機能とマルチプロセッサの両方に対応するカーネルにおけるスケジューリング単位番号を構成するためのマクロ |

#### 【TOPPERS新世代カーネル統合仕様との関係】

スケジューリング単位番号を構成するマクロ（SCHEDNO）を追加した.

### 2.15.4 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 |

#### (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/ASP3カーネルにおける規定】

ASP3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/FMP3カーネルにおける規定】

FMP3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/HRP3カーネルにおける規定】

HRP3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/HRMP3カーネルにおける規定】

HRMP3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【HRMPS0010】.

【TOPPERS/SSP3カーネルにおける規定】

SSP3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，TOPPERS新世代カーネル統合仕様との関係】

メールボックス機能を廃止したため，メッセージ優先度の最小値（TMIN_MPRI）と最大値（TMAX_MPRI）は廃止した.

### (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［NGKI0643］.

| | |
|---|---|
| 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_TEPP_PRC | タイムイベント処理プロセッサの集合 |

ここでTOPPERS_TEPP_PRCは，タイムイベント処理プロセッサの集合をビットマップとして表す符号無し整数型の数値で，プロセッサIDがNのプロセッサがタイムイベント処理プロセッサである場合に下位からN番目のビットが1，そうでない場合には下位からN番目のビットが0となっている.

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルでは，すべてのプロセッサがタイムイベント処理プロセッサであるため，TOPPERS_TEPP_PRCは用意していない【HRMPS0012】.

【TOPPERS新世代カーネル統合仕様との関係】

TOPPERS_SYSTIM_PRCID（システム時刻管理プロセッサのID番号），TOPPERS_SYSTIM_LOCAL（ローカルタイマ方式），TOPPERS_SYSTIM_GLOBAL（グローバルタイマ方式）を廃止した.
TOPPERS_TEPP_PRCを追加した.

### (5) メモリ配置の方法

保護機能対応カーネルでは，自動メモリ配置と手動メモリ配置のどちらが使われているか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【NGKI0607】.

| | |
|---|---|
| TOPPERS_ML_AUTO | 自動メモリ配置の場合にマクロ定義 |
| TOPPERS_ML_MANUAL | 手動メモリ配置の場合にマクロ定義 |

### (6) バージョン情報【NGKI0528】

| | |
|---|---|
| TKERNEL_MAKER | カーネルのメーカコード（＝0x0118） |
| TKERNEL_PRID | カーネルの識別番号 |
| TKERNEL_SPVER | カーネル仕様のバージョン番号 |
| TKERNEL_PRVER | カーネルのバージョン番号 |

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

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

| | |
|---|---|
| 0x0001 | TOPPERS/JSPカーネル |
| 0x0002 | 予約（IIMPカーネル） |
| 0x0003 | 予約（IDLカーネル） |
| 0x0004 | TOPPERS/FI4カーネル |
| 0x0005 | TOPPERS/FDMPカーネル |
| 0x0006 | TOPPERS/HRPカーネル（TOPPERS/HRP2カーネル，TOPPERS/HRP3カーネルを含む） |
| 0x0007 | TOPPERS/ASPカーネル（TOPPERS/ASP3カーネルを含む） |
| 0x0008 | TOPPERS/FMPカーネル（TOPPERS/FMP3カーネルを含む） |
| 0x0009 | TOPPERS/SSPカーネル（TOPPERS/SSP3カーネルを含む） |
| 0x000a | TOPPERS/ASP Safetyカーネル |
| 0x000b | TOPPRES/HRMPカーネル（TOPPERS/HRMP3カーネル） |

カーネル仕様のバージョン番号（TKERNEL_SPVER）は，上位8ビット（0xf6）がTOPPERS第3世代カーネル（ITRON系）仕様であることを，中位4ビットがメジャーバージョン番号，下位4ビットがマイナーバージョン番号を表す．

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

# 第３章　システムインタフェースレイヤAPI仕様

## 3.1 システムインタフェースレイヤの概要

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

SILの機能は，プロセッサの特権モードで実行されているプログラムが使用することを想定している【NGKI0801】．非特権モードで実行されているプログラムからSILの機能を呼び出した場合の動作は，次の例外を除いては保証されない【NGKI0802】．

- 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

【μITRON4.0仕様との関係】

μITRON4.0仕様では，システムインタフェースレイヤAPI仕様は規定されていなかった．

## 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．

SILスピンロックで用いるプロセッサ間ロックは，システムに唯一存在する【NGKI0815】．この節でプロセッサ間ロックと言った場合には，SILスピンロックで用いるプロセッサ間ロックのことを指す．

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（ただし，ロックの取得を試みる前からマスクしていた割込みは，マスクしたままとなる）【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．

### (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()

プロセッサ間ロックが取得されていない状態である場合には，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】.

プロセッサ間ロックが，SIL_LOC_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，全割込みロック状態への遷移のみを行う【NGKI0852】.

(3) SIL_UNL_SPN()

プロセッサ間ロックの取得状態と，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_LOC_SPNを実行せずにSIL_UNL_SPNを呼び出した場合の動作は保証されない【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．

また，デバイスレジスタやメモリへの書込みが完了するまで待つために，以下のメモリ空間同期書込み関数を用意している．

(15) void sil_swrb_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み，書込みが完了するまで待つ【NGKI0845】．

(16) void sil_swrh_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み，書込みが完了するまで待つ【NGKI0846】．

(17) void sil_swrh_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み，書込みが完了するまで待つ【NGKI0847】．

(18) void sil_swrh_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み，書込みが完了するまで待つ【NGKI0848】．

(19) void sil_swrw_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み，書込みが完了するまで待つ【NGKI0849】．

(20) void sil_swrw_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み，書

込みが完了するまで待つ【NGKI0850】.

(21) void sil_swrw_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み，書込みが完了するまで待つ【NGKI0851】.

【TOPPERS新世代カーネル統合仕様との関係】

メモリ空間同期書込み関数群を追加した.

## 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)

(15) void sil_swrb_iop(uint8_t *iop, uint8_t data)

(16) void sil_swrh_iop(uint16_t *iop, uint16_t data)

(17) void sil_swrh_lep(uint16_t *iop, uint16_t data)

(18) void sil_swrh_bep(uint16_t *iop, uint16_t data)

(19) void sil_swrw_iop(uint32_t *iop, uint32_t data)

(20) void sil_swrw_lep(uint32_t *iop, uint32_t data)

(21) void sil_swrw_bep(uint32_t *iop, uint32_t data)

【TOPPERS新世代カーネル統合仕様との関係】

I/O空間同期書込み関数群を追加した.

## 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.

# 第４章　カーネルAPI仕様

この章では，カーネルのAPI仕様について規定する．

【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．

INT → int_t

UINT → uint_t

VP → void *

VP_INT → intptr_t

SIZE → size_t

ただし，拡張情報の型（μITRON4.0仕様ではVP_INT）については，EXINFに変更した．

タスク例外処理機能を廃止し，それに代えて，タスク終了要求機能を追加した．これに関連して，待ち状態に入るすべてのサービスコールから，E_RASTERエラーが返る場合ある．

非タスクコンテキスト専用のサービスコールの概念を廃止し，非タスクコンテキストからも，タスクコンテキストと同じ名称のサービスコールを呼び出すこととした．

以上の変更については，個別のAPI仕様では記述しない．

【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．

【TOPPERS新世代カーネル統合仕様との関係】

TOPPERS共通データ型の変更に従い，パラメータのデータ型を次の通り変更した．

SIZE → size_t

また，拡張情報の型を，EXINFに変更した．

タスク例外処理機能を廃止し，それに代えて，タスク終了要求機能を追加した．これに関連して，待ち状態に入るすべてのサービスコールから，E_RASTERエラーが返る場合ある．

非タスクコンテキスト専用のサービスコールの概念を廃止し，非タスクコンテキストからも，タスクコンテキストと同じ名称のサービスコールを呼び出すこととした．

以上の変更については，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
- サブ優先度（サブ優先度機能をサポートするカーネルの場合）
- 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/SSP3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，タスクの優先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，「4.4.5 ミューテックス」の節を参照すること．

タスクのサブ優先度は，優先度が同一のタスクの間の優先順位を決定するために使われる値であり，タスクの生成時に最低値（＝UINT_MAX）に初期化される【NGKI3681】．

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の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，タスクの起動要求がmact_tskによりキューイングされている状況で，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ある．タスクの生成時に未設定の状態に初期化され，タスクの起動要求キューイング数が0の間は，未設定の状態になっている【NGKI1007】．

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，ユーザスタック領域と呼び，ユーザタスクが非特権モードで実行する間に用いる【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，カーネルの管理領域として扱われる【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持つもので，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【NGKI1012】．システムスタック領域は，カーネルの管理領域として扱われる【NGKI1013】．

タスク属性には，次の属性を指定することができる【NGKI3526】．

| | | |
|---|---|---|
| TA_ACT | 0x01U | タスクの生成時にタスクを起動する |
| TA_NOACTQUE | 0x02U | タスクに対する起動要求をキューイングしない |
| 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 |

C言語によるタスクの記述形式は次の通り【NGKI1018】．

```
void task(EXINF 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/ASP3カーネルにおける規定】

ASP3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/FMP3カーネルにおける規定】

FMP3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ない［FMPS0006］．

【TOPPERS/HRP3カーネルにおける規定】

HRP3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ない［HRPS0005］．

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルでは，TMAX_ACTCNTは1に固定されている【HRMPS0101】．また，制約タスクはサポートしない［HRMPS0007］．

【TOPPERS/SSP3カーネルにおける規定】

SSP3カーネルでは，TMAX_ACTCNTは1に固定されている【SSPS0101】．

SSP3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】．

SSP3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】．SSP3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．そのため，共有スタック領域に必要なサイズは，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】

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）を廃止し，自タスクの拡張情報の参照するサービスコール（get_inf），タスク状態を参照するサービスコール（get_tst），タスクのサブ優先度を変更するサービスコール（chg_spr）を追加した．

タスク属性に，TA_NOACTQUEを追加した．

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．

【TOPPERS新世代カーネル統合仕様との関係】

タスク状態を参照するサービスコール（get_tst）とタスクのサブ優先度を変更するサービスコール（chg_spr）を追加した．

タスク属性に，TA_NOACTQUEを追加した．TA_ACTの値を変更した．

| | |
|---|---|
| **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, EXINF exinf, TASK task,
                                PRI itskpri, size_t stksz, STK_T *stk })
```

※ stkの記述は省略することができる【NGKI3900】.

＊保護機能対応カーネルの場合

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

※ stk，sstksz，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 | 生成するタスクの識別名（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（acre_tskの場合） |

＊タスクの生成情報（パケットの内容）

| | | |
|---|---|---|
| ATR | tskatr | タスク属性 |
| EXINF | 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_t | 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_t | sstksz | タスクのシステムスタック領域のサイズ（バイト数，保護機能対応カーネルの場合） |
| STK_T * | sstk | タスクのシステムスタック領域の先頭番地（保護機能対応カーネルの場合） |

【リターンパラメータ】

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

【エラーコード】

| | |
|---|---|
| E_CTX | コンテキストエラー<br>・非タスクコンテキストからの呼出し〔s〕【NGKI1026】<br>・CPUロック状態からの呼出し〔s〕【NGKI1027】 |
| E_RSATR | 予約属性<br>・tskatrが無効【NGKI1028】<br>・属する保護ドメインの指定が有効範囲外または無所属〔sP〕【NGKI1029】<br>・保護ドメインの囲みの中に記述されていない〔SP〕【NGKI1030】<br>・属するクラスの指定が有効範囲外〔sM〕【NGKI1031】<br>・クラスの囲みの中に記述されていない〔SM〕【NGKI1032】 |
| E_PAR | パラメータエラー<br>・taskがプログラムの先頭番地として正しくない【NGKI1033】<br>・itskpriが有効範囲外【NGKI1034】<br>・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<br>・属する保護ドメインに対する通常操作1が許可されていない〔sP〕【NGKI3966】 |
| E_MACV | メモリアクセス違反<br>・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〔sP〕【NGKI1036】 |
| E_NOID | ID番号不足<br>・割り付けられるタスクIDがない〔sD〕【NGKI1037】 |
| E_NOMEM | メモリ不足<br>・スタック領域が確保できない【NGKI1038】<br>・システムスタック領域が確保できない〔P〕【NGKI1039】 |
| E_OBJ | オブジェクト状態エラー<br>・tskidで指定した識別名のオブジェクトが登録済み〔S〕【NGKI1040】<br>・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の生成情報に従って，タスクを生成する【NGKI5188】．タスクの生成時には，以下の処理を行う．

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，スタック領域の設定にsstkszも用いられる（具体的な仕様は後述する）．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にターゲット定義の最小値よりも小さい値を指定した時は，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，sstkszは整数定数式パラメータ，exinf，task，stk，sstkは一般定数式パラメータである【NGKI1047】．ただし，保護機能対応カー

ネルで手動メモリ配置の場合には，stkは整数定数式パラメータである【NGKI3906】．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】．

保護機能とマルチプロセッサの両方に対応するカーネルでは，生成するタスクがユーザタスクの場合に，［NGKI0628］の制約が課される．具体的には，生成するタスクが属するユーザドメインが，アイドルドメインにまとめられておらず，属するクラスの初期割付けプロセッサでタイムウィンドウを割り当てられていない場合，E_PARエラーとなる【NGKI5108】．

〔stkにNULLを指定した場合〕

stkにNULLを指定するか，静的APIにおいてstkの記述を省略した場合，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】．

保護機能対応カーネルにおいて，生成するタスクがユーザタスクの場合，コンフィギュレータにより確保されるスタック領域（ユーザスタック領域）は，タスクが属する保護ドメイン向け（マルチプロセッサ対応カーネルの場合には，さらに，タスクが属するクラス向け）の標準スタックリージョン内に配置され，「2.11.6 ユーザタスクのユーザスタック領域」の節の規定に従って，メモリオブジェクトとしてカーネルに登録される【NGKI1051】．

ただし，保護機能対応カーネルで手動メモリ配置の場合で，生成するタスクがユーザタスクの場合には，stkがNULLであってはならず，静的APIにおいてstkの記述を省略することができない．stkがNULLであった場合や，stkの記述を省略した場合には，E_PARエラーとなる【NGKI3907】．

保護機能対応カーネルにおいて，生成するタスクがシステムタスクの場合，コンフィギュレータにより確保されるスタック領域は，カーネルドメイン向け（マルチプロセッサ対応カーネルの場合には，さらに，タスクが属するクラス向け）の標準スタックリージョン内に配置される【NGKI5203】．

静的APIにより制約タスクを生成する場合（tskatrにTA_RSTRを指定して生成する場合），スタック領域は，制約タスクの起動時優先度毎に確保され，同じ起動時優先度を持つ制約タスクで共有される【NGKI1052】．確保されるスタック領域のサイズは，それを共有する制約タスクのスタック領域のサイズ（stksz）の最大値となる【NGKI1053】．マルチプロセッサ対応カーネルでは，以上のスタック領域の確保処理を，制約タスクの初期割付けプロセッサ毎に行う【NGKI1054】．

〔stkにNULL以外を指定した場合〕

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【NGKI1055】．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【NGKI1056】．

保護機能対応カーネルにおいて，生成するタスクがシステムタスクの場合に，stkとstkszで指定した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【NGKI1057】．

保護機能対応カーネルにおいて，生成するタスクがユーザタスクの場合，stkとstkszで指定したスタック領域（ユーザスタック領域）は，「2.11.6 ユーザタスクのユーザスタック領域」の節の規定に従って，メモリオブジェクトとしてカーネルに登録される【NGKI1058】．そのため，上の方法を用いてスタック領域を確保しても，ターゲット定義の制約に合致する先頭番地とサイズとなるとは限らず，スタック領域をアプリケーションで確保する方法は，ターゲット定義である【NGKI1059】．また，stkとstkszで指定したスタック領域が，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【NGKI1060】．

〔sstkとsstkszの扱い〕

保護機能対応カーネルにおけるsstkとsstkszの扱いは，生成するタスクがユーザタスクの場合とシステムタスクの場合で異なる．

生成するタスクがユーザタスクの場合の扱いは次の通り．

sstkにNULLを指定するか，静的APIにおいてsstkの記述を省略した場合，sstkszで指定したサイズのシステムスタック領域が，コンフィギュレータまたはカーネルにより確保される【NGKI1061】．sstksz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【NGKI1062】．sstkszの記述も省略した場合には，ターゲット定義のデフォルトのサイズで確保される【NGKI1063】．

ここで，コンフィギュレータにより確保されるシステムスタック領域は，カーネルドメイン向け（マルチプロセッサ対応カーネルの場合には，さらに，タスクが属するクラス向け）の標準スタックリージョン内に配置される【NGKI5204】．

sstkにNULL以外を指定した場合，sstkとsstkszで指定したシステムスタック領域は，アプリケーションで確保しておく必要がある【NGKI1064】．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，sstkやsstksz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【NGKI1065】．また，sstkとsstkszで指定したシステム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【NGKI1066】．

生成するタスクがシステムタスクの場合の扱いは次の通り．

sstkに指定することができるのは，NULLのみである．sstkにNULL以外を指定した場合には，E_PARエラーとなる【NGKI1068】．

sstkszに0以外の値を指定した場合で，stkがNULLの場合には，コンフィギュレータまたはカーネルにより確保されるスタック領域のサイズに，sstkszが加えられる【NGKI1069】．stkszにsstkszを加えた値が，ターゲット定義の制約に合致しないサイズになる時には，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【NGKI1070】．

sstkszに0以外の値を指定した場合で，stkがNULLでない場合には，E_PARエラーとなる【NGKI1071】．

静的APIでsstkszの記述を省略するか，sstkszに0を指定した場合，これらの処理は行わず，E_PARエラーにもならない【NGKI1072】．

〔**スタック領域への他のタスクからのアクセス**〕

この仕様では，タスクのスタック領域に，他のタスクからアクセスできるとは限らない【NGKI5166】．実際，保護機能対応カーネルでは，ユーザタスクのユーザスタック領域には，他のユーザタスクはアクセスすることができないのが基本である．また，マルチプロセッサ対応カーネルでは，スタック領域が他のプロセッサからアクセスできない領域に配置される場合もある．そのため，ローカル変数へのポインタを渡すなどして，他のタスクからローカル変数にアクセスさせることは推奨しない．

【TOPPERS/ASP3カーネルにおける規定】

ASP3カーネルの動的生成機能拡張パッケージでは，acre_tskをサポートする【ASPS0105】．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルの動的生成機能拡張パッケージでは，acre_tskをサポートする【HRPS0175】．ただし，生成するタスクがユーザタスクの場合，stkとstkszで指定するユーザスタック領域の扱いは，以下の点でCRE_TSKと異なる．

- 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】．
- ユーザスタック領域をアプリケーションで確保する方法は，ターゲット定義で，CRE_TSKとは異なる場合がある【HRPS0240】．その方法に従わず，指定したユーザスタック領域がターゲット定義の制約に合致しない場合には，E_OBJエラーとなる【HRPS0241】．
- 指定したユーザスタック領域は，カーネルにメモリオブジェクトとして登録されない【HRPS0242】．そのため，指定したユーザスタック領域が，動的に生成された他のタスクのユーザスタック領域と重なる場合のエラーチェックは行われない【HRPS0243】．同時に起動される他のタスクのユーザスタック領域と重ならないようにするのは，アプリケーションの責任である．

【TOPPERS/SSP3カーネルにおける規定】

SSP3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】．

SSP3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】．

SSP3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，SSP3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．

アクセス権を制御するためのアクセスの種別を，保護ドメインに対する通常操作1に変更した．

【TOPPERS新世代カーネル統合仕様との関係】

アクセス権を制御するためのアクセスの種別を，保護ドメインに対する通常操作1に変更した．

保護機能対応カーネルにおいて，コンフィギュレータが，スタック領域をどこに確保するかを規定した．

【未決定事項】

サービスコール（acre_tsk）で，stkにNULLを指定して制約タスクを生成する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．

保護機能対応カーネルにおいて，サービスコール（acre_tsk）で，tskにNULLを指定してユーザタスクを生成する場合のユーザ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<br>・保護ドメインの囲みの中に記述されていない〔P〕【NGKI3979】<br>・クラスの囲みの中に記述されていない〔M〕【NGKI1075】 |

【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/ASP3カーネルにおける規定】

ASP3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/HRP3カーネルにおける規定】

HRP3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 | 対象タスクの識別名（SAC_TSKの場合）またはID番号（sac_tskの場合） |
| 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（sac_tskの場合） |

＊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<br>・非タスクコンテキストからの呼出し〔s〕【NGKI1080】<br>・CPUロック状態からの呼出し〔s〕【NGKI1081】 |
| E_ID | 不正ID番号<br>・tskidが有効範囲外〔s〕【NGKI1082】 |
| E_RSATR | 予約属性<br>・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】<br>・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<br>・対象タスクが未登録【NGKI1085】 |
| E_OACV | オブジェクトアクセス違反<br>・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<br>・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<br>・対象タスクは静的APIで生成された〔s〕【NGKI1088】<br>・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/HRP3カーネルにおける規定】

HRP3カーネルの動的生成機能拡張パッケージでは，sac_tskをサポートする【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 | 対象タスクの識別名 |
| 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<br>・exepriが有効範囲外【NGKI1094】 |
| E_ILUSE | サービスコール不正使用<br>・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<br>・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/SSP3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<br>・非タスクコンテキストからの呼出し【NGKI1101】<br>・CPUロック状態からの呼出し【NGKI1102】 |
| E_ID | 不正ID番号<br>・tskidが有効範囲外【NGKI1103】 |
| E_NOEXS | オブジェクト未登録<br>・対象タスクが未登録【NGKI1104】 |
| E_OACV | オブジェクトアクセス違反<br>・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<br>・対象タスクが休止状態でない【NGKI1106】<br>・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/ASP3カーネルにおける規定】

ASP3カーネルの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/HRP3カーネルにおける規定】

HRP3カーネルの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】．

## act_tsk　　タスクの起動〔TI〕【NGKI3529】

【C言語API】

```
ER ercd = act_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<br>・CPUロック状態からの呼出し【NGKI1114】 |
| E_ID | 不正ID番号<br>・tskidが有効範囲外【NGKI1115】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・対象タスクに対する通常操作1が許可されていない〔P〕【NGKI1117】 |
| E_QOVR | キューイングオーバフロー<br>・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く，対象タスクがTA_NOACTQUE属性でない場合には，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【NGKI3527】．対象タスクがTA_NOACTQUE属性である場合や，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【NGKI3528】．

タスクコンテキストから呼び出した場合，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は，対象タスクの次回起動時の割付けプロセッサを設定しない．

## mact_tsk　割付けプロセッサ指定でのタスクの起動〔TIM〕【NGKI3530】

【C言語API】

```
ER ercd = mact_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<br>・CPUロック状態からの呼出し【NGKI1126】 |
| E_NOSPT | 未サポート機能<br>・対象タスクが制約タスク【NGKI1127】 |
| E_ID | 不正ID番号<br>・tskidが有効範囲外【NGKI1128】<br>・prcidが有効範囲外【NGKI1129】 |
| E_PAR | パラメータエラー<br>・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<br>・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<br>・対象タスクに対する通常操作1が許可されていない〔P〕【NGKI1131】 |
| E_QOVR | キューイングオーバフロー<br>・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【NGKI1132】．

対象タスクが休止状態でなく，対象タスクがTA_NOACTQUE属性でない場合には，対象タスクの起動要求キューイング数に1が加えられ，次回起動時の割付けプロセッサがprcidで指定したプロセッサに設定される【NGKI3557】．対象タスクがTA_NOACTQUE属性である場合や，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【NGKI3558】．

タスクコンテキストから呼び出した場合，tskidにTSK_SELF（＝0）を指定すると，自タスクが対象タス

クとなる【NGKI1135】.

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【NGKI1136】.

保護機能対応カーネルでは，対象タスクがユーザタスクの場合に，［NGKI0628］の制約が課される．具体的には，対象タスクが属するユーザドメインが，アイドルドメインにまとめられておらず，prcidで指定したプロセッサでタイムウィンドウを割り当てられていない場合，E_PARエラーとなる【NGKI5109】.

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とする【NGKI1137】.

【補足説明】

TMAX_ACTCNTが2以上の場合でも，対象タスクが次に起動される時の割付けプロセッサは，キューイングされない．すなわち，プロセッサAに割り付けられた休止状態でないタスクを対象として，プロセッサBを割付けプロセッサとしてmact_tskを呼び出し，さらにプロセッサCを割付けプロセッサとしてmact_tskを呼び出すと，対象タスクの次回起動時の割付けプロセッサがプロセッサCに変更され，対象タスクがプロセッサBで実行されることはない．なお，TMAX_ACTCNTが1の場合には，プロセッサCを割付けプロセッサとした2回目のmact_tskがE_QOVRエラーとなるため，次回起動時の割付けプロセッサはプロセッサBのまま変更されない.

【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.

## can_act　　タスク起動要求のキャンセル〔T〕【NGKI1138】

【C言語API】

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

【パラメータ】

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

【リターンパラメータ】

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

【エラーコード】

| | |
|---|---|
| E_CTX | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI1139】<br>・CPUロック状態からの呼出し【NGKI1140】 |
| E_ID | 不正ID番号<br>・tskidが有効範囲外【NGKI1141】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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の状態に戻される【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/SSP3カーネルにおける規定】

SSP3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<br>・非タスクコンテキストからの呼出し【NGKI1148】<br>・CPUロック状態からの呼出し【NGKI1149】<br>・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<br>・対象タスクが制約タスク【NGKI1150】 |
| E_ID | 不正ID番号<br>・tskidが有効範囲外【NGKI1151】<br>・prcidが有効範囲外【NGKI1152】 |
| E_PAR | パラメータエラー<br>・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<br>・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<br>・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<br>・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優先度が同じタスク（サブ優先度が使われる状況では，サブ優先度も同じタスク）の中で，優先順位が最も低くなる【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】．

保護機能対応カーネルでは，対象タスクがユーザタスクの場合に，［NGKI0628］の制約が課される．具体的には，対象タスクが属するユーザドメインが，アイドルドメインにまとめられておらず，prcidで指定したプロセッサでタイムウィンドウを割り当てられていない場合，E_PARエラーとなる【NGKI5110】．

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ト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ト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，優先度が同じタスク（サブ優先度が使われる状況では，サブ優先度も同じタスク）の中で，優先順位が最も低くなる．

【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．

## get_tst　　タスク状態の参照〔T〕【NGKI3613】

【C言語API】

```
ER ercd = get_tst(ID tskid, STAT *p_tskstat)
```

【パラメータ】

| | | |
|---|---|---|
| ID | tskid | 対象タスクのID番号 |
| STAT * | p_tskstat | タスク状態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 |
| STAT | tskstat | タスク状態 |

【エラーコード】

| | |
|---|---|
| E_CTX | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI3614】<br>・CPUロック状態からの呼出し【NGKI3615】 |
| E_ID | 不正ID番号<br>・tskidが有効範囲外【NGKI3616】 |
| E_NOEXS | オブジェクト未登録<br>・対象タスクが未登録〔D〕【NGKI3617】 |
| E_OACV | オブジェクトアクセス違反<br>・対象タスクに対する参照操作が許可されていない〔P〕【NGKI3618】 |
| E_MACV | メモリアクセス違反<br>・p_tskstat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〕【NGKI3619】 |

【機能】

tskidで指定したタスク（対象タスク）のタスク状態を参照する．具体的な振舞いは以下の通り．

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，p_tskstatが指すメモリ領域に返される【NGKI3620】．

| | | |
|---|---|---|
| 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，TTS_SUSが返される場合がある

【NGKI5209】．この状況は，自タスクに対してget_tst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【NGKI3621】．

【使用上の注意】

get_tst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，get_tst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．また，マルチプロセッサ対応カーネルでは，対象タスクの現在状態を参照した直後に，他のプロセッサで実行されている処理により対象タスクの状態が変化する場合もある．

【μITRON4.0仕様，TOPPERS新世代カーネル統合仕様との関係】

μITRON4.0仕様およびTOPPERS新世代カーネル統合仕様に定義されていないサービスコールである．

## 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<br>・非タスクコンテキストからの呼出し【NGKI1184】<br>・CPUロック状態からの呼出し【NGKI1185】 |
| E_NOSPT | 未サポート機能<br>・対象タスクが制約タスク【NGKI1186】 |
| E_ID | 不正ID番号<br>・tskidが有効範囲外【NGKI1187】 |
| E_PAR | パラメータエラー<br>・tskpriが有効範囲外【NGKI1188】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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<br>・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<br>・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】．

chg_priを発行した結果，対象タスクの現在優先度が変化する場合と，対象タスクが優先度上昇状態でない場合（正確な定義は「4.4.5　ミューテックス」の節を参照すること．ミューテックス機能を使わない場合には，この条件は常に成り立つ）には，以下の処理が行われる．

対象タスクが実行できる状態の場合には，現在優先度が同じ（サブ優先度が使われる状況では，サブ優先

度も同じ）タスクの中で，対象タスクの優先順位が最も低くなる【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】．待ち行列中に現在優先度が同じ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昇状態であり，対象タスクの現在優先度が変化しない場合には，対象タスクの優先順位や待ち行列中での順序は変更されない【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】．

【TOPPERS/SSP3カーネルにおける規定】

SSP3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．

【TOPPERS新世代カーネル統合仕様との関係】

ユーザドメインが指定できる最高のタスク優先度の機能を廃止した．

## 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<br>・非タスクコンテキストからの呼出し【NGKI1203】<br>・CPUロック状態からの呼出し【NGKI1204】 |
| E_ID | 不正ID番号<br>・tskidが有効範囲外【NGKI1205】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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<br>・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<br>・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/SSP3カーネルにおける規定】

SSP3カーネルでは，get_priをサポートしない【SSPS0122】．

## chg_spr　　タスクのサブ優先度の変更〔T〕【NGKI3665】

【C言語API】

```
ER ercd = chg_spr(ID tskid, uint_t subpri)
```

【パラメータ】

| | | |
|---|---|---|
| ID | tskid | 対象タスクのID番号 |
| uint_t | subpri | サブ優先度 |

【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<br>・非タスクコンテキストからの呼出し【NGKI3666】<br>・CPUロック状態からの呼出し【NGKI3667】 |
| E_NOSPT | 未サポート機能<br>・対象タスクが制約タスク【NGKI3668】 |
| E_ID | 不正ID番号<br>・tskidが有効範囲外【NGKI3669】 |
| E_NOEXS | オブジェクト未登録<br>・対象タスクが未登録〔D〕【NGKI3670】 |
| E_OACV | オブジェクトアクセス違反<br>・対象タスクに対する通常操作2が許可されていない〔P〕【NGKI3671】 |

【機能】

tskidで指定したタスク（対象タスク）のサブ優先度を，subpriで指定した値に変更する．具体的な振舞いは以下の通り．

対象タスクのサブ優先度が，subpriで指定した値に変更される【NGKI3672】．対象タスクが実行できる状態で，対象タスクの現在優先度がサブ優先度を使用すると設定されており，対象タスクが優先度上昇状態でない場合（正確な定義は「4.4.5　ミューテックス」の節を参照すること．ミューテックス機能を使わない場合には，この条件は常に成り立つ）には，優先度とサブ優先度が同じタスクの中で，対象タスクの優先順位が最も低くなる【NGKI3673】．

tskidにTSK_SELF（＝0）を指定すると，自タスクが対象タスクとなる【NGKI3674】．

【TOPPERS/ASP3カーネルにおける規定】

ASP3カーネルでは，chg_sprをサポートしない【ASPS0231】．ただし，サブ優先度機能拡張パッケージでは，chg_sprをサポートする【ASPS0232】．

【TOPPERS/FMP3カーネルにおける規定】

FMP3カーネルでは，chg_sprをサポートする【FMPS0169】．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルでは，chg_sprをサポートしない【HRPS0224】．

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルでは，chg_sprをサポートしない【HRMPS0103】．

【TOPPERS/SSP3カーネルにおける規定】

SSP3カーネルでは，chg_sprをサポートしない【SSPS0161】．

【μITRON4.0仕様，TOPPERS新世代カーネル統合仕様との関係】

μITRON4.0仕様およびTOPPERS新世代カーネル統合仕様に定義されていないサービスコールである．

## get_inf　　自タスクの拡張情報の参照〔T〕【NGKI1212】

【C言語API】

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

【パラメータ】

| | | |
|---|---|---|
| EXINF * | 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 |
| EXINF | exinf | 拡張情報 |

【エラーコード】

| | |
|---|---|
| E_CTX | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI1213】<br>・CPUロック状態からの呼出し【NGKI1214】 |
| E_MACV | メモリアクセス違反<br>・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/SSP3カーネルにおける規定】

SSP3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 |
| uint_t | subpri | タスクのサブ優先度（サブ優先度機能をサポートするカーネルの場合） |
| 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 | raster | タスク終了要求フラグ |
| bool_t | dister | タスク終了禁止状態 |
| 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<br>・非タスクコンテキストからの呼出し【NGKI1218】<br>・CPUロック状態からの呼出し【NGKI1219】 |
| E_ID | 不正ID番号<br>・tskidが有効範囲外【NGKI1220】 |

| | |
|---|---|
| E_NOEXS | オブジェクト未登録<br>・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<br>・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<br>・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】．

サブ優先度機能をサポートするカーネルでは，subpriには対象タスクのサブ優先度が返される【NGKI3662】．

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_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（＝UINT32_MAX）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】．

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】．また，rasterには，タスク終了要求フラグがセットされている場合にはtrue，クリアされている場合にはfalseが返される【NGKI3467】．disterには，タスク終了禁止状態の場合にはtrue，タスク終了許可状態の場合にはfalseが返される【NGKI3468】．対象タスクが休止状態である場合には，wupcnt，raster，dister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，svclevelに，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，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】．

【TOPPERS/ASP3カーネルにおける規定】

ASP3カーネルでは，tskwaitにTTW_SMBF，TTW_RMBFが返ることはない【ASPS0235】．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【ASPS0208】．

【TOPPERS/FMP3カーネルにおける規定】

FMP3カーネルでは，tskwaitにTTW_SMBF，TTW_RMBFが返ることはない【FMPS0173】．

【TOPPERS/SSP3カーネルにおける規定】

SSP3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．また，マルチプロセッサ対応カーネルでは，対象タスクの現在状態を参照した直後に，他のプロセッサで実行されている処理により対象タスクの状態が変化する場合もある．

【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．参照できる情報に，タスク終了要求フラグ（raster）とタスク終了禁止状態（dister）を追加した．逆に，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，拡張サービスコールのネストレベル（svclevel）を追加した．

【TOPPERS新世代カーネル統合仕様との関係】

参照できる情報に，タスク終了要求フラグ（raster）とタスク終了禁止状態（dister）を追加した．逆に，保護機能対応カーネルで参照できる情報から，タスク例外処理マスク状態（texmsk）と待ち禁止状態（waifbd）を削除した．

## 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/ASP3カーネルにおける規定】

ASP3カーネルでは，TMAX_WUPCNTは1に固定されている【ASPS0113】．

【TOPPERS/FMP3カーネルにおける規定】

FMP3カーネルでは，TMAX_WUPCNTは1に固定されている【FMPS0107】．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルでは，TMAX_WUPCNTは1に固定されている【HRPS0109】．

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルでは，TMAX_WUPCNTは1に固定されている【HRMPS0104】．

【TOPPERS/SSP3カーネルにおける規定】

SSP3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．

【TOPPERS新世代カーネル統合仕様との関係】

待ち禁止状態への遷移を行うサービスコール（dis_wai）とその解除を行うサービスコール（ena_wai）を廃止した．待ち禁止状態の役割は，新たに導入したタスク終了要求フラグがセットされた状態が兼ねている．

| | |
|---|---|
| **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<br>・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<br>・制約タスクからの呼出し【NGKI1255】 |
| E_PAR | パラメータエラー<br>・tmoutが無効（tslp_tskの場合）【NGKI1256】 |
| E_TMOUT | ポーリング失敗またはタイムアウト（tslp_tskの場合）【NGKI1257】 |
| E_RLWAI | 待ち状態の強制解除【NGKI1258】 |
| E_RASTER | タスクの終了要求【NGKI3455】 |

【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〔TI〕【NGKI3531】

【C言語API】

```
ER ercd = wup_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<br>・CPUロック状態からの呼出し【NGKI1265】 |
| E_NOSPT | 未サポート機能<br>・対象タスクが制約タスク【NGKI1266】 |
| E_ID | 不正ID番号<br>・tskidが有効範囲外【NGKI1267】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・対象タスクに対する通常操作1が許可されていない〔P〕【NGKI1269】 |
| E_OBJ | オブジェクト状態エラー<br>・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<br>・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】．

タスクコンテキストから呼び出した場合，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<br>・非タスクコンテキストからの呼出し【NGKI1277】<br>・CPUロック状態からの呼出し【NGKI1278】 |
| E_NOSPT | 未サポート機能<br>・対象タスクが制約タスク【NGKI1279】 |
| E_ID | 不正ID番号<br>・tskidが有効範囲外【NGKI1280】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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<br>・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〔TI〕【NGKI3532】

【C言語API】

```
ER ercd = rel_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<br>・CPUロック状態からの呼出し【NGKI1290】 |
| E_NOSPT | 未サポート機能<br>・対象タスクが制約タスク【NGKI1291】 |
| E_ID | 不正ID番号<br>・tskidが有効範囲外【NGKI1292】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・対象タスクに対する通常操作2が許可されていない〔P〕【NGKI1294】 |
| E_OBJ | オブジェクト状態エラー<br>・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<br>・非タスクコンテキストからの呼出し【NGKI1299】<br>・CPUロック状態からの呼出し【NGKI1300】<br>・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<br>・対象タスクが制約タスク【NGKI1301】 |
| E_ID | 不正ID番号<br>・tskidが有効範囲外【NGKI1302】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・対象タスクに対する通常操作2またはシステム状態2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<br>・対象タスクが休止状態【NGKI1305】 |
| E_RASTER | タスクの終了要求<br>・対象タスクのタスク終了要求フラグがセットされている【NGKI3605】 |
| E_QOVR | キューイングオーバフロー<br>・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．

【μITRON4.0/PX仕様，TOPPERS新世代カーネル統合仕様との関係】

システム状態に対するアクセス許可ベクタにより，呼出しを禁止できるようにした．

## 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<br>・非タスクコンテキストからの呼出し【NGKI1313】<br>・CPUロック状態からの呼出し【NGKI1314】 |
| E_NOSPT | 未サポート機能<br>・対象タスクが制約タスク【NGKI1315】 |
| E_ID | 不正ID番号<br>・tskidが有効範囲外【NGKI1316】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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<br>・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】．

## 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<br>・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<br>・制約タスクからの呼出し【NGKI1350】 |
| E_PAR | パラメータエラー<br>・dlytimがTMAX_RELTIMより大きい【NGKI1351】 |
| E_RLWAI | 待ち状態の強制解除【NGKI1352】 |
| E_RASTER | タスクの終了要求【NGKI3456】 |

【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 タスク終了機能

タスク終了機能には，自タスクを終了させる機能，タスクを安全に終了させるためのタスク終了要求機能，タスクを強制終了させる機能が含まれる．

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】．さらに，マルチプロセッサ対応カーネルにおいては，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】．

タスク終了要求機能に関連して，各タスクが持つ情報は次の通り【NGKI3684】．

- タスク終了要求フラグ
- タスク終了禁止フラグ

タスク終了要求フラグは，タスクに対する終了要求を記憶するためのフラグであり，タスクの起動時にクリアした状態に初期化される【NGKI3451】．

タスク終了禁止フラグは，タスクに対する終了要求を保留するためのフラグで，タスクの起動時にクリアした状態に初期化される【NGKI3454】．タスク終了禁止フラグがセットされた状態をタスク終了禁止状態，クリアされた状態をタスク終了許可状態と呼ぶ．

実行状態のタスクに対して，「タスク終了要求フラグがセットされる」「タスク終了許可状態である」「ディスパッチ許可状態である」「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5つの条件が揃った場合，そのタスクは終了する．すなわち，そのタスクに対して，タスク終了時に行うべき処理が行われる【NGKI3683】．

タスク終了要求フラグがセットされたタスクは，広義の待ち状態になることがない．具体的には，広義の待ち状態のタスクのタスク終了要求フラグがセットされた場合，タスクは実行できる状態に遷移する【NGKI3452】．タスクが待ち状態であった場合には，待ち状態となったサービスコールからE_RASTERが返る【NGKI5206】．また，タスク終了要求フラグがセットされたタスクが，自タスクを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した場合には，E_RASTERエラーとなる【NGKI3453】．さらに，タスク終了要求フラグがセットされたタスクを強制待ち状態に遷移させるサービスコールを呼び出した場合にも，E_RASTERエラーとなる【NGKI3607】．

**【μITRON4.0仕様，TOPPERS新世代カーネル統合仕様との関係】**

タスク終了要求機能を追加した．

タスク終了要求フラグがセットされた状態は，μITRON4.0/PX仕様で導入された待ち禁止状態を役割を兼ねている．ただし，この仕様でタスク終了要求フラグがセットされたタスクが自タスクを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した場合，無条件にエラーになる［NGKI3453］のに対して，TOPPERS新世代カーネル統合仕様で待ち禁止状態のタスクが同様のサービスコールを呼び出した場合には，待ち状態に遷移しようとした場合にのみエラーとなる［NGKI0222］．例えば，資源数が1のセマフォに対して，待ち禁止状態のタスクがwai_semを発行した場合，セマフォの獲得に成功するのに対して，タスク終了要求フラグがセットされたタスクが発行した場合には，E_RASTERエラーとなる．

## 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<br>・カーネルの誤動作【NGKI1163】 |
| E_CTX | コンテキストエラー<br>・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】．

マルチプロセッサ対応カーネルでは，CPUロック解除状態に遷移させた後，自タスクを終了させる前に，割込みを受け付ける場合があり，その結果，他のタスクに切り換わる場合もある【NGKI5183】．これは，カーネル処理の不可分性の原則の例外となっている．

ext_tskが正常に処理された場合，ext_tskからはリターンしない【NGKI1169】．

【補足説明】

保護機能対応カーネルにおいては，ext_tskにより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した場合，システム周期の切換えとタイムウィンドウの切換えの保留が解除され，システム周期の切換えとタイムウィンドウの切換えが起こる可能性がある．

【TOPPERS/SSP3カーネルにおける規定】

SSP3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．

## ras_ter　　タスクの終了要求〔T〕【NGKI3469】

【C言語API】

```
ER ercd = ras_ter(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<br>・非タスクコンテキストからの呼出し【NGKI3470】<br>・CPUロック状態からの呼出し【NGKI3471】 |
| E_ID | 不正ID番号<br>・tskidが有効範囲外【NGKI3472】 |
| E_NOEXS | オブジェクト未登録<br>・対象タスクが未登録〔D〕【NGKI3473】 |
| E_OACV | オブジェクトアクセス違反<br>・対象タスクに対する通常操作2が許可されていない〔P〕【NGKI3474】 |
| E_ILUSE | サービスコール不正使用<br>・対象タスクが自タスク【NGKI3475】 |
| E_OBJ | オブジェクト状態エラー<br>・対象タスクが休止状態【NGKI3476】<br>・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）に終了要求を行う．具体的な振舞いは以下の通り．

対象タスクが休止状態でなく，タスク終了許可状態である場合には，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【NGKI3477】．

対象タスクが休止状態でなく，タスク終了禁止状態である場合には，対象タスクのタスク終了要求フラグがセットされる【NGKI3478】．また，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【NGKI3479】．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_RASTERが返る【NGKI3480】．さらに，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【NGKI3606】．

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【NGKI3481】．

【TOPPERS/SSP3カーネルにおける規定】

SSP3カーネルでは，ras_terをサポートしない【SSPS0151】.

【μITRON4.0仕様，TOPPERS新世代カーネル統合仕様との関係】

μITRON4.0仕様およびTOPPERS新世代カーネル統合仕様に定義されていないサービスコールである.

## dis_ter　　タスク終了の禁止〔T〕【NGKI3482】

【C言語API】

```
ER ercd = dis_ter()
```

【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<br>・非タスクコンテキストからの呼出し【NGKI3483】<br>・CPUロック状態からの呼出し【NGKI3484】 |
| E_OACV | オブジェクトアクセス違反<br>・システム状態2に対する通常操作1が許可されていない〔P〕【NGKI3764】 |

【機能】

自タスクのタスク終了禁止フラグをセットする【NGKI3486】．すなわち，自タスクをタスク終了禁止状態に遷移させる．

【TOPPERS/SSP3カーネルにおける規定】

SSP3カーネルでは，dis_terをサポートしない【SSPS0152】．

【μITRON4.0仕様，TOPPERS新世代カーネル統合仕様との関係】

μITRON4.0仕様およびTOPPERS新世代カーネル統合仕様に定義されていないサービスコールである．

## ena_ter　　タスク終了の許可〔T〕【NGKI3487】

【C言語API】

```
ER ercd = ena_ter()
```

【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<br>・非タスクコンテキストからの呼出し【NGKI3488】<br>・CPUロック状態からの呼出し【NGKI3489】 |
| E_OACV | オブジェクトアクセス違反<br>・システム状態2に対する通常操作1が許可されていない〔P〕【NGKI3765】 |

【機能】

自タスクのタスク終了禁止フラグをクリアする【NGKI3491】．すなわち，自タスクをタスク終了許可状態に遷移させる．

【補足説明】

自タスクのタスク終了要求フラグがセットされていた場合には，タスク終了許可状態へ遷移した結果，自タスクが終了する可能性がある．この場合，ena_terからはリターンしない．

【TOPPERS/SSP3カーネルにおける規定】

SSP3カーネルでは，ena_terをサポートしない【SSPS0153】．

【μITRON4.0仕様，TOPPERS新世代カーネル統合仕様との関係】

μITRON4.0仕様およびTOPPERS新世代カーネル統合仕様に定義されていないサービスコールである．

## sns_ter　　タスク終了禁止状態の参照〔TI〕【NGKI3494】

【C言語API】

```
bool_t state = sns_ter()
```

【パラメータ】

なし

【リターンパラメータ】

| | | |
|---|---|---|
| bool_t | state | タスク終了禁止状態 |

【機能】

実行状態のタスク（タスクコンテキストから呼び出した場合には自タスク）のタスク終了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，タスク終了禁止状態の場合にtrue，タスク終了許可状態の場合にfalseが返る【NGKI3495】．sns_ter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【NGKI3496】．

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終了禁止フラグを参照する【NGKI3497】．

【補足説明】

sns_ter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．

【TOPPERS/SSP3カーネルにおける規定】

SSP3カーネルでは，sns_terをサポートしない【SSPS0154】．

【μITRON4.0仕様，TOPPERS新世代カーネル統合仕様との関係】

μITRON4.0仕様およびTOPPERS新世代カーネル統合仕様に定義されていないサービスコールである．

## 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<br>・非タスクコンテキストからの呼出し【NGKI1171】<br>・CPUロック状態からの呼出し【NGKI1172】 |
| E_ID | 不正ID番号<br>・tskidが有効範囲外【NGKI1173】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・対象タスクに対する通常操作2またはシステム状態2に対する管理操作が許可されていない〔P〕【NGKI3508】 |
| E_ILUSE | サービスコール不正使用<br>・対象タスクが自タスク【NGKI1176】 |
| E_OBJ | オブジェクト状態エラー<br>・対象タスクが休止状態【NGKI1177】<br>・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/SSP3カーネルにおける規定】

SSP3カーネルでは，ter_tskをサポートしない【SSPS0120】．

【μITRON4.0/PX仕様，TOPPERS新世代カーネル統合仕様との関係】

システム状態に対するアクセス許可ベクタにより，呼出しを禁止できるようにした．

## 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/SSP3カーネルにおける規定】

SSP3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】

メールボックス機能を廃止し，スピンロック機能を追加した．

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．

【TOPPERS新世代カーネル統合仕様との関係】

メールボックス機能を廃止した．

【仕様決定の理由】

メールボックス機能を廃止したのは，保護機能対応カーネルではサポートできないためである．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．

### 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識別名（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（acre_semの場合） |

＊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<br>・非タスクコンテキストからの呼出し〔s〕【NGKI1454】<br>・CPUロック状態からの呼出し〔s〕【NGKI1455】 |
| E_RSATR | 予約属性<br>・sematrが無効【NGKI1456】<br>・属する保護ドメインの指定が有効範囲外〔sP〕【NGKI1457】<br>・属するクラスの指定が有効範囲外〔sM〕【NGKI1458】<br>・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<br>・maxsemが有効範囲（1以上，TMAX_MAXSEM以下）外【NGKI1468】<br>・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<br>・属する保護ドメイン（または無所属）に対する通常操作1が許可されていない〔sP〕【NGKI3967】 |
| E_MACV | メモリアクセス違反<br>・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<br>・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<br>・semidで指定した識別名のオブジェクトが登録済み〔S〕【NGKI1463】 |

【機能】

各パラメータで指定したセマフォの生成情報に従って，セマフォを生成する【NGKI5189】．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/ASP3カーネルにおける規定】

ASP3カーネルの動的生成機能拡張パッケージでは，acre_semをサポートする【ASPS0120】．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルの動的生成機能拡張パッケージでは，acre_semをサポートする【HRPS0180】．

【μITRON4.0/PX仕様，TOPPERS新世代カーネル統合仕様との関係】

アクセス権を制御するためのアクセスの種別を，保護ドメインに対する通常操作1に変更した．

## 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<br>・クラスの囲みの中に記述されていない〔M〕【NGKI1470】 |

【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/ASP3カーネルにおける規定】

ASP3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/HRP3カーネルにおける規定】

HRP3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 | 対象セマフォの識別名（SAC_SEMの場合）またはID番号（sac_semの場合） |
| 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（sac_semの場合） |

＊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<br>・非タスクコンテキストからの呼出し〔s〕【NGKI1475】<br>・CPUロック状態からの呼出し〔s〕【NGKI1476】 |
| E_ID | 不正ID番号<br>・semidが有効範囲外〔s〕【NGKI1477】 |
| E_RSATR | 予約属性<br>・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】<br>・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<br>・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<br>・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<br>・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<br>・対象セマフォは静的APIで生成された〔s〕【NGKI1483】<br>・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/HRP3カーネルにおける規定】

HRP3カーネルの動的生成機能拡張パッケージでは，sac_semをサポートする【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<br>・非タスクコンテキストからの呼出し【NGKI1488】<br>・CPUロック状態からの呼出し【NGKI1489】 |
| E_ID | 不正ID番号<br>・semidが有効範囲外【NGKI1490】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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<br>・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】．

【TOPPERS/ASP3カーネルにおける規定】

ASP3カーネルの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/HRP3カーネルにおける規定】

HRP3カーネルの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〔TI〕【NGKI3533】

【C言語API】

```
ER ercd = sig_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<br>・CPUロック状態からの呼出し【NGKI1501】 |
| E_ID | 不正ID番号<br>・semidが有効範囲外【NGKI1502】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・対象セマフォに対する通常操作1が許可されていない〔P〕【NGKI1504】 |
| E_QOVR | キューイングオーバフロー<br>・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<br>・非タスクコンテキストからの呼出し【NGKI1513】<br>・CPUロック状態からの呼出し【NGKI1514】<br>・ディスパッチ保留状態（非タスクコンテキストの実行中，CPUロック状態を除く）からの呼出し（pol_semを除く）【NGKI1515】 |
| E_NOSPT | 未サポート機能<br>・制約タスクからの呼出し（pol_semを除く）【NGKI1516】 |
| E_ID | 不正ID番号<br>・semidが有効範囲外【NGKI1517】 |
| E_PAR | パラメータエラー<br>・tmoutが無効（twai_semの場合）【NGKI1518】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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 | 待ち状態の強制解除（pol_semを除く）【NGKI1522】 |
| E_RASTER | タスクの終了要求（pol_semを除く）【NGKI3457】 |
| 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<br>・非タスクコンテキストからの呼出し【NGKI1527】<br>・CPUロック状態からの呼出し【NGKI1528】 |
| E_ID | 不正ID番号<br>・semidが有効範囲外【NGKI1529】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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】

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<br>・非タスクコンテキストからの呼出し【NGKI1536】<br>・CPUロック状態からの呼出し【NGKI1537】 |
| E_ID | 不正ID番号<br>・semidが有効範囲外【NGKI1538】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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<br>・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．また，マルチプロセッサ対応カーネルでは，対象セマフォの現在状態を参照した直後に，他のプロセッサで実行されている処理により対象セ

マフォの状態が変化する場合もある.

### 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/ASP3カーネルにおける規定】

ASP3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/FMP3カーネルにおける規定】

FMP3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/HRP3カーネルにおける規定】

HRP3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】．

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3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【HRMPS0105】．

【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識別名（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（acre_flgの場合） |

＊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<br>・非タスクコンテキストからの呼出し〔s〕【NGKI1560】<br>・CPUロック状態からの呼出し〔s〕【NGKI1561】 |
| E_RSATR | 予約属性<br>・flgatrが無効【NGKI1562】<br>・属する保護ドメインの指定が有効範囲外〔sP〕【NGKI1563】<br>・属するクラスの指定が有効範囲外〔sM〕【NGKI1564】<br>・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<br>・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<br>・属する保護ドメイン（または無所属）に対する通常操作1が許可されていない〔sP〕【NGKI3968】 |
| E_MACV | メモリアクセス違反<br>・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<br>・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<br>・flgidで指定した識別名のオブジェクトが登録済み〔S〕【NGKI1569】 |

【機能】

各パラメータで指定したイベントフラグの生成情報に従って，イベントフラグを生成する【NGKI5190】．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/ASP3カーネルにおける規定】

ASP3カーネルの動的生成機能拡張パッケージでは，acre_flgをサポートする【ASPS0126】．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルの動的生成機能拡張パッケージでは，acre_flgをサポートする【HRPS0183】．

【μITRON4.0/PX仕様，TOPPERS新世代カーネル統合仕様との関係】

アクセス権を制御するためのアクセスの種別を，保護ドメインに対する通常操作1に変更した．

## 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<br>・クラスの囲みの中に記述されていない〔M〕【NGKI1573】 |

【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/ASP3カーネルにおける規定】

ASP3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/HRP3カーネルにおける規定】

HRP3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識別名（SAC_FLGの場合）またはID番号（sac_flgの場合） |
| 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（sac_flgの場合） |

＊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<br>・非タスクコンテキストからの呼出し〔s〕【NGKI1578】<br>・CPUロック状態からの呼出し〔s〕【NGKI1579】 |
| E_ID | 不正ID番号<br>・flgidが有効範囲外〔s〕【NGKI1580】 |
| E_RSATR | 予約属性<br>・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】<br>・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<br>・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<br>・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〕<br>【NGKI1584】 |
| E_MACV | メモリアクセス違反<br>・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〕<br>【NGKI1585】 |
| E_OBJ | オブジェクト状態エラー<br>・対象イベントフラグは静的APIで生成された〔s〕【NGKI1586】<br>・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〕<br>【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/HRP3カーネルにおける規定】

HRP3カーネルの動的生成機能拡張パッケージでは，sac_flgをサポートする【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<br>・非タスクコンテキストからの呼出し【NGKI1591】<br>・CPUロック状態からの呼出し【NGKI1592】 |
| E_ID | 不正ID番号<br>・flgidが有効範囲外【NGKI1593】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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<br>・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】．

【TOPPERS/ASP3カーネルにおける規定】

ASP3カーネルの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/HRP3カーネルにおける規定】

HRP3カーネルの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〔TI〕【NGKI3534】

【C言語API】

```
ER ercd = set_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<br>・CPUロック状態からの呼出し【NGKI1604】 |
| E_ID | 不正ID番号<br>・flgidが有効範囲外【NGKI1605】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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〔P〕<br>【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】．対象イベントフラグの待ち行列にタスクが存在する場合には，待ち解除の条件を満たしたタスクが，待ち行列の前方につながれたものから順に待ち解除される【NGKI1609】．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【NGKI1610】．

【補足説明】

対象イベントフラグがTA_CLR属性である場合には，待ち解除の条件を満たしたタスクを1つ待ち解除した時点で，対象イベントフラグのビットパターンが0にクリアされるため，他のタスクが待ち解除されることはない．

## clr_flg　　イベントフラグのクリア〔T〕【NGKI1611】

【C言語API】

```
ER ercd = clr_flg(ID flgid, FLGPTN clrptn)
```

【パラメータ】

| | | |
|---|---|---|
| ID | flgid | 対象イベントフラグのID番号 |
| FLGPTN | clrptn | クリアするビットパターン（クリアしないビットを1，クリアするビットを0とする） |

【リターンパラメータ】

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

【エラーコード】

| | |
|---|---|
| E_CTX | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI1612】<br>・CPUロック状態からの呼出し【NGKI1613】 |
| E_ID | 不正ID番号<br>・flgidが有効範囲外【NGKI1614】 |
| E_NOEXS | オブジェクト未登録<br>・対象イベントフラグが未登録〔D〕【NGKI1615】 |
| E_OACV | オブジェクトアクセス違反<br>・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〔P〕【NGKI1616】 |

【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clrptnで指定したビットをクリアする．対象イベントフラグのビットパターンは，それまでの値とclrptnで指定した値のビット毎論理積（C言語の"&"）に更新される【NGKI1617】．

| | |
|---|---|
| **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<br>・非タスクコンテキストからの呼出し【NGKI1621】<br>・CPUロック状態からの呼出し【NGKI1622】<br>・ディスパッチ保留状態（非タスクコンテキストの実行中，CPUロック状態を除く）からの呼出し（pol_flgを除く）【NGKI1623】 |
| E_NOSPT | 未サポート機能<br>・制約タスクからの呼出し（pol_flgを除く）【NGKI1624】 |
| E_ID | 不正ID番号<br>・flgidが有効範囲外【NGKI1625】 |
| E_PAR | パラメータエラー<br>・waiptnが0【NGKI1626】<br>・wfmodeが無効（TWF_ORWまたはTWF_ANDWでない）【NGKI1627】<br>・tmoutが無効（twai_flgの場合）【NGKI1628】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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<br>・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<br>・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 | 待ち状態の強制解除（pol_flgを除く）【NGKI1634】 |
| E_RASTER | タスクの終了要求（pol_flgを除く）【NGKI3458】 |
| 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<br>・非タスクコンテキストからの呼出し【NGKI1640】<br>・CPUロック状態からの呼出し【NGKI1641】 |
| E_ID | 不正ID番号<br>・flgidが有効範囲外【NGKI1642】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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】

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<br>・非タスクコンテキストからの呼出し【NGKI1649】<br>・CPUロック状態からの呼出し【NGKI1650】 |
| E_ID | 不正ID番号<br>・flgidが有効範囲外【NGKI1651】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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<br>・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．また，マルチプロセッサ対応

カーネルでは，対象イベントフラグの現在状態を参照した直後に，他のプロセッサで実行されている処理により対象イベントフラグの状態が変化する場合もある．

### 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 })
```

※ dtqmbの記述は省略することができる【NGKI3901】.

【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識別名（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（acre_dtqの場合） |

＊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<br>・非タスクコンテキストからの呼出し〔s〕【NGKI1667】<br>・CPUロック状態からの呼出し〔s〕【NGKI1668】 |
| E_RSATR | 予約属性<br>・dtqatrが無効【NGKI1669】<br>・属する保護ドメインの指定が有効範囲外〔sP〕【NGKI1670】<br>・属するクラスの指定が有効範囲外〔sM〕【NGKI1671】<br>・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<br>・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<br>・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<br>・属する保護ドメイン（または無所属）に対する通常操作1が許可されていない〔sP〕【NGKI3969】 |

| | |
|---|---|
| E_MACV | メモリアクセス違反<br>・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<br>・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<br>・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<br>・dtqidで指定した識別名のオブジェクトが登録済み〔S〕【NGKI1677】<br>・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の生成情報に従って，データキューを生成する【NGKI5191】．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にNULLを指定するか，静的APIにおいてdtqmbの記述を省略した場合，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，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】．

なお，dtqmbにNULL以外を指定できないカーネルでは，TSZ_DTQMBとTCNT_DTQMBは用意していない【NGKI5179】．

【TOPPERS/ASP3カーネルにおける規定】

ASP3カーネルでは，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】．

ASP3カーネルの動的生成機能拡張パッケージでは，acre_dtqをサポートする【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/FMP3カーネルにおける規定】

FMP3カーネルでは，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/HRP3カーネルにおける規定】

HRP3カーネルでは，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】．

HRP3カーネルの動的生成機能拡張パッケージでは，acre_dtqをサポートする【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】．

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルでは，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【HRMPS0106】．

【μITRON4.0仕様との関係】

μITRON4.0/PX仕様にあわせて，データキューの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．

【μITRON4.0/PX仕様，TOPPERS新世代カーネル統合仕様との関係】

アクセス権を制御するためのアクセスの種別を，保護ドメインに対する通常操作1に変更した．

## 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<br>・クラスの囲みの中に記述されていない〔M〕【NGKI1690】 |

【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/ASP3カーネルにおける規定】

ASP3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/HRP3カーネルにおける規定】

HRP3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識別名（SAC_DTQの場合）またはID番号（sac_dtqの場合） |
| 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（sac_dtqの場合） |

＊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<br>・非タスクコンテキストからの呼出し〔s〕【NGKI1695】<br>・CPUロック状態からの呼出し〔s〕【NGKI1696】 |
| E_ID | 不正ID番号<br>・dtqidが有効範囲外〔s〕【NGKI1697】 |
| E_RSATR | 予約属性<br>・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】<br>・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<br>・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<br>・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<br>・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<br>・対象データキューは静的APIで生成された〔s〕【NGKI1703】<br>・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/HRP3カーネルにおける規定】

HRP3カーネルの動的生成機能拡張パッケージでは，sac_dtqをサポートする【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<br>・非タスクコンテキストからの呼出し【NGKI1708】<br>・CPUロック状態からの呼出し【NGKI1709】 |
| E_ID | 不正ID番号<br>・dtqidが有効範囲外【NGKI1710】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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<br>・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．

【TOPPERS/ASP3カーネルにおける規定】

ASP3カーネルの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/HRP3カーネルにおける規定】

HRP3カーネルの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）〔TI〕【NGKI3535】 |
| **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 = 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<br>・CPUロック状態からの呼出し【NGKI1724】<br>・ディスパッチ保留状態（CPUロック状態を除く）からの呼出し（psnd_dtqを除く）【NGKI1725】 |
| E_NOSPT | 未サポート機能<br>・制約タスクからの呼出し（psnd_dtqを除く）【NGKI1726】 |
| E_ID | 不正ID番号<br>・dtqidが有効範囲外【NGKI1727】 |
| E_PAR | パラメータエラー<br>・tmoutが無効（tsnd_dtqの場合）【NGKI1728】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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〔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 | 待ち状態の強制解除（psnd_dtqを除く）【NGKI1732】 |
| E_RASTER | タスクの終了要求（psnd_dtqを除く）【NGKI3459】 |
| E_DLT | 待ちオブジェクトの削除または再初期化（psnd_dtqを除く）【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〔TI〕【NGKI3536】

【C言語API】

```
ER ercd = fsnd_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<br>・CPUロック状態からの呼出し【NGKI1742】 |
| E_ID | 不正ID番号<br>・dtqidが有効範囲外【NGKI1743】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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〔P〕【NGKI1745】 |
| E_ILUSE | サービスコール不正使用<br>・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<br>・非タスクコンテキストからの呼出し【NGKI1754】<br>・CPUロック状態からの呼出し【NGKI1755】<br>・ディスパッチ保留状態（非タスクコンテキストの実行中，CPUロック状態を除く）からの呼出し（prcv_dtqを除く）【NGKI1756】 |
| E_NOSPT | 未サポート機能<br>・制約タスクからの呼出し（prcv_dtqを除く）【NGKI1757】 |
| E_ID | 不正ID番号<br>・dtqidが有効範囲外【NGKI1758】 |
| E_PAR | パラメータエラー<br>・tmoutが無効（trcv_dtqの場合）【NGKI1759】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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<br>・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 | 待ち状態の強制解除（prcv_dtqを除く）【NGKI1764】 |

| | |
|---|---|
| E_RASTER | タスクの終了要求（prcv_dtqを除く）【NGKI3460】 |
| 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】．

対象データキューのデータキュー管理領域にデータが格納されておらず，送信待ち行列にタスクが存在する場合には，送信待ち行列の先頭のタスクの送信データを受信する【NGKI1769】．送信待ち行列の先頭のタスクは，待ち解除される【NGKI3422】．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【NGKI1770】．

対象データキューのデータキュー管理領域にデータが格納されておらず，送信待ち行列にタスクが存在しない場合には，自タスクはデータキューからの受信待ち状態となり，対象データキューの受信待ち行列につながれる【NGKI1771】．

## ini_dtq　　データキューの再初期化〔T〕【NGKI1772】

【C言語API】

```
ER ercd = ini_dtq(ID dtqid)
```

【パラメータ】

| | | |
|---|---|---|
| ID | dtqid | 対象データキューのID番号 |

【リターンパラメータ】

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

【エラーコード】

| | |
|---|---|
| E_CTX | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI1773】<br>・CPUロック状態からの呼出し【NGKI1774】 |
| E_ID | 不正ID番号<br>・dtqidが有効範囲外【NGKI1775】 |
| E_NOEXS | オブジェクト未登録<br>・対象データキューが未登録〔D〕【NGKI1776】 |
| E_OACV | オブジェクトアクセス違反<br>・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〕【NGKI1777】 |

【機能】

dtqidで指定したデータキュー（対象データキュー）を再初期化する．具体的な振舞いは以下の通り．

対象データキューのデータキュー管理領域は，格納されているデータがない状態に初期化される【NGKI1778】．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【NGKI1779】．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【NGKI1780】．

【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】

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<br>・非タスクコンテキストからの呼出し【NGKI1782】<br>・CPUロック状態からの呼出し【NGKI1783】 |
| E_ID | 不正ID番号<br>・dtqidが有効範囲外【NGKI1784】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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<br>・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．また，マルチプロセッサ対応カーネルでは，対象データキューの現在状態を参照した直後に，他のプロセッサで実行されている処理により対象データキューの状態が変化する場合もある．

### 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，カーネルの管理領域として扱われる【NGKI1794】．

送信待ち行列は，優先度データキューに対してデータが送信できるまで待っている状態（優先度データキューへの送信待ち状態）のタスクが，データを送信できる順序でつながれているキューである．また，受信待ち行列は，優先度データキューからデータが受信できるまで待っている状態（優先度データキューからの受信待ち状態）のタスクが，データを受信できる順序でつながれているキューである．

優先度データキュー属性には，次の属性を指定することができる【NGKI1795】．

| | | |
|---|---|---|
| | 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【NGKI1796】．受信待ち行列は，FIFO順に固定されている【NGKI1797】．

優先度データキュー機能に関連するカーネル構成マクロは次の通り．

| | | |
|---|---|---|
| | TMIN_DPRI | データ優先度の最小値（＝1）【NGKI1798】 |
| | 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）【NGKI1799】 |

【TOPPERS/ASP3カーネルにおける規定】

ASP3カーネルでは，データ優先度の最大値（TMAX_DPRI）は16に固定されている【ASPS0138】．ただし，タスク優先度拡張パッケージでは，TMAX_DPRIを256に拡張する【ASPS0139】．

【TOPPERS/FMP3カーネルにおける規定】

FMP3カーネルでは，データ優先度の最大値（TMAX_DPRI）は16に固定されている【FMPS0124】．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルでは，データ優先度の最大値（TMAX_DPRI）は16に固定されている【HRPS0124】．

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルでは，データ優先度の最大値（TMAX_DPRI）は16に固定されている【HRMPS0107】．

【使用上の注意】

データの優先度が使われるのは，データが優先度データキュー管理領域に格納される場合のみであり，データを送信するタスクが送信待ち行列につながれている間には使われない．そのため，送信待ち行列につながれているタスクが，優先度データキュー管理領域に格納されているデータよりも高い優先度のデータを送信しようとしている場合でも，最初に送信されるのは，優先度データキュー管理領域に格納されているデータである．また，TA_TPRI属性の優先度データキューにおいても，送信待ち行列はタスクの優先度順となり，タスクが送信しようとしているデータの優先度順となるわけではない．

【μITRON4.0仕様との関係】

μITRON4.0仕様に規定されていない機能である．

| | |
|---|---|
| **CRE_PDQ** | 優先度データキューの生成〔S〕【NGKI1800】 |
| **acre_pdq** | 優先度データキューの生成〔TD〕【NGKI1801】 |

【静的API】

```
CRE_PDQ(ID pdqid, { ATR pdqatr, uint_t pdqcnt, PRI maxdpri, void *pdqmb })
```

※ pdqmbの記述は省略することができる【NGKI3902】.

【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識別名（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（acre_pdqの場合） |

＊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<br>・非タスクコンテキストからの呼出し〔s〕【NGKI1802】<br>・CPUロック状態からの呼出し〔s〕【NGKI1803】 |
| E_RSATR | 予約属性<br>・pdqatrが無効【NGKI1804】<br>・属する保護ドメインの指定が有効範囲外〔sP〕【NGKI1805】<br>・属するクラスの指定が有効範囲外〔sM〕【NGKI1806】<br>・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<br>・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<br>・maxdpriが有効範囲（TMIN_DPRI以上，TMAX_DPRI以下）外【NGKI1819】<br>・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<br>・属する保護ドメイン（または無所属）に対する通常操作1が許可されていない〔sP〕【NGKI3970】 |
| E_MACV | メモリアクセス違反<br>・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<br>・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<br>・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<br>・pdqidで指定した識別名のオブジェクトが登録済み〔S〕【NGKI1812】<br>・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の生成情報に従って，優先度データキューを生成する【NGKI5192】．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にNULLを指定するか，静的APIにおいてpdqmbの記述を省略した場合，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，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おいて，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】．

なお，pdqmbにNULL以外を指定できないカーネルでは，TSZ_PDQMBとTCNT_PDQMBは用意していない【NGKI5180】．

【TOPPERS/ASP3カーネルにおける規定】

ASP3カーネルでは，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】．

ASP3カーネルの動的生成機能拡張パッケージでは，acre_pdqをサポートする【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/FMP3カーネルにおける規定】

FMP3カーネルでは，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/HRP3カーネルにおける規定】

HRP3カーネルでは，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】．

HRP3カーネルの動的生成機能拡張パッケージでは，acre_pdqをサポートする【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】．

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルでは，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【HRMPS0108】．

【TOPPERS新世代カーネル統合仕様との関係】

アクセス権を制御するためのアクセスの種別を，保護ドメインに対する通常操作1に変更した．

## 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<br>・クラスの囲みの中に記述されていない〔M〕【NGKI1827】 |

【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/ASP3カーネルにおける規定】

ASP3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/HRP3カーネルにおける規定】

HRP3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識別名（SAC_PDQの場合）またはID番号（sac_pdqの場合） |
| 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（sac_pdqの場合） |

＊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<br>・非タスクコンテキストからの呼出し〔s〕【NGKI1832】<br>・CPUロック状態からの呼出し〔s〕【NGKI1833】 |
| E_ID | 不正ID番号<br>・pdqidが有効範囲外〔s〕【NGKI1834】 |
| E_RSATR | 予約属性<br>・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】<br>・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<br>・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<br>・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<br>・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<br>・対象優先度データキューは静的APIで生成された〔s〕【NGKI1840】<br>・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/HRP3カーネルにおける規定】

HRP3カーネルの動的生成機能拡張パッケージでは，sac_pdqをサポートする【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<br>・非タスクコンテキストからの呼出し【NGKI1845】<br>・CPUロック状態からの呼出し【NGKI1846】 |
| E_ID | 不正ID番号<br>・pdqidが有効範囲外【NGKI1847】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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<br>・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．

【TOPPERS/ASP3カーネルにおける規定】

ASP3カーネルの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/HRP3カーネルにおける規定】

HRP3カーネルの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）〔TI〕【NGKI3537】 |
| **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 = 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<br>・CPUロック状態からの呼出し【NGKI1861】<br>・ディスパッチ保留状態（CPUロック状態を除く）からの呼出し（psnd_pdqを除く）【NGKI1862】 |
| E_NOSPT | 未サポート機能<br>・制約タスクからの呼出し（psnd_pdqを除く）【NGKI1863】 |
| E_ID | 不正ID番号<br>・pdqidが有効範囲外【NGKI1864】 |
| E_PAR | パラメータエラー<br>・tmoutが無効（tsnd_pdqの場合）【NGKI1865】<br>・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<br>・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<br>・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〔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 | 待ち状態の強制解除（psnd_pdqを除く）【NGKI1869】 |
| E_RASTER | タスクの終了要求（psnd_pdqを除く）【NGKI3461】 |

| | |
|---|---|
| E_DLT | 待ちオブジェクトの削除または再初期化（psnd_pdqを除く）【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<br>・非タスクコンテキストからの呼出し【NGKI1880】<br>・CPUロック状態からの呼出し【NGKI1881】<br>・ディスパッチ保留状態（非タスクコンテキストの実行中，CPUロック状態を除く）からの呼出し（prcv_pdqを除く）【NGKI1882】 |
| E_NOSPT | 未サポート機能<br>・制約タスクからの呼出し（prcv_pdqを除く）【NGKI1883】 |
| E_ID | 不正ID番号<br>・pdqidが有効範囲外【NGKI1884】 |
| E_PAR | パラメータエラー<br>・tmoutが無効（trcv_pdqの場合）【NGKI1885】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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<br>・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】<br>・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 | 待ち状態の強制解除（prcv_pdqを除く）【NGKI1891】 |
| E_RASTER | タスクの終了要求（prcv_pdqを除く）【NGKI3462】 |
| 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】 |

【機能】

pdqidで指定した優先度データキュー（対象優先度データキュー）からデータを受信する．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，その優先度はp_datapriが指すメモリ領域に返される【NGKI1894】．具体的な振舞いは以下の通り．

対象優先度データキューの優先度データキュー管理領域にデータが格納されている場合には，優先度データキュー管理領域の先頭に格納されたデータを受信する【NGKI1893】．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，データの優先度順で優先度データキュー管理領域に格納され，そのタスクは待ち解除される【NGKI1895】．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【NGKI1896】．

対象優先度データキューの優先度データキュー管理領域にデータが格納されておらず，送信待ち行列にタスクが存在する場合には，送信待ち行列の先頭のタスクの送信データを受信する【NGKI1897】．送信待ち行列の先頭のタスクは，待ち解除される【NGKI1899】．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【NGKI1900】．

対象優先度データキューの優先度データキュー管理領域にデータが格納されておらず，送信待ち行列にタスクが存在しない場合には，自タスクは優先度データキューからの受信待ち状態となり，対象優先度データキューの受信待ち行列につながれる【NGKI1901】．

## ini_pdq　　優先度データキューの再初期化〔T〕【NGKI1902】

【C言語API】

```
ER ercd = ini_pdq(ID pdqid)
```

【パラメータ】

| | | |
|---|---|---|
| ID | pdqid | 対象優先度データキューのID番号 |

【リターンパラメータ】

| | | |
|---|---|---|
| ER | 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<br>・非タスクコンテキストからの呼出し【NGKI1903】<br>・CPUロック状態からの呼出し【NGKI1904】 |
| E_ID | 不正ID番号<br>・pdqidが有効範囲外【NGKI1905】 |
| E_NOEXS | オブジェクト未登録<br>・対象優先度データキューが未登録〔D〕【NGKI1906】 |
| E_OACV | オブジェクトアクセス違反<br>・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〕【NGKI1907】 |

【機能】

pdqidで指定した優先度データキュー（対象優先度データキュー）を再初期化する．具体的な振舞いは以下の通り．

対象優先度データキューの優先度データキュー管理領域は，格納されているデータがない状態に初期化される【NGKI1908】．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【NGKI1909】．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【NGKI1910】．

【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】

優先度データキューを再初期化した場合に，アプリケーションとの整合性を保つのは，アプリケーションの責任である．

## ref_pdq　　優先度データキューの状態参照〔T〕【NGKI1911】

【C言語API】

```
ER ercd = ref_pdq(ID pdqid, T_RPDQ *pk_rpdq)
```

【パラメータ】

| | | |
|---|---|---|
| ID | pdqid | 対象優先度データキューのID番号 |
| T_RPDQ * | pk_rpdq | 優先度データキューの現在状態を入れるパケットへのポインタ |

【リターンパラメータ】

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

＊優先度データキューの現在状態（パケットの内容）

| | | |
|---|---|---|
| ID | stskid | 優先度データキューの送信待ち行列の先頭のタスクのID番号 |
| ID | rtskid | 優先度データキューの受信待ち行列の先頭のタスクのID番号 |
| uint_t | spdqcnt | 優先度データキュー管理領域に格納されているデータの数 |

【エラーコード】

| | |
|---|---|
| E_CTX | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI1912】<br>・CPUロック状態からの呼出し【NGKI1913】 |
| E_ID | 不正ID番号<br>・pdqidが有効範囲外【NGKI1914】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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<br>・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．また，マルチプロセッサ対応カーネルでは，対象優先度データキューの現在状態を参照した直後に，他のプロセッサで実行されている処理により対象優先度データキューの状態が変化する場合もある．

### 4.4.5 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抑えるために，優先度上限プロトコル（priority ceiling protocol）と優先度継承プロトコル（priority inheritance 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／優先度継承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／優先度継承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_INHERIT | 0x02U | 優先度継承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_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_INHERIT，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，常に，以下の優先度の中で最も高い優先度に設定される【NGKI2014】．

- そのタスクのベース優先度
- 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
- そのタスクがロックしている優先度継承ミューテックスに対してロック待ち状態となっているタスクの現在優先度

タスクが優先度上昇状態であるとは，タスクが以下のいずれかを満たしている状態をいう．

- そのタスクが優先度上限ミューテックスをロックしている．
- そのタスクがロックしている優先度継承ミューテックスに対してロック待ち状態となっているタスクの中で，そのタスクのベース優先度と同じかそれより高い現在優先度を持つものがある．

サブ優先度機能をサポートするカーネルでは，タスクが優先度上昇状態である間は，サブ優先度は一時的に最高値（＝0）になったものとみなしてスケジューリングされる【NGKI5219】．

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と，サブ優先度が使われる状況（サブ優先度機能をサポートするカーネルで，タスクの現在優先度がサブ優先度を使用すると設定されている場合）で優先度上昇状態であるか否かが変化する場合には，以下の処理が行われる．

現在優先度等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，優先度が同じタスク（サブ優先度が使われる状況では，サブ優先度も同じタスク）の中で優先順位が最も高くなる【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，優先度が同じタスク（サブ優先度が使われる状況では，サブ優先度も同じタスク）の中で優先順位が最も低くなる【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】．ロック解除の具体的な処理内容については，unl_mtxの機能の項を参照すること．

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．

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場合がある．

【補足説明】

優先度継承ミューテックスのロックを待っているタスクの現在優先度が，ミューテックスの操作により変更された場合，そのミューテックスをロックしているタスクの現在優先度の変更が必要になる場合がある．これを推移的な優先度継承と呼ぶ．さらにそのタスクが，別の優先度継承ミューテックスのロックを待っていた場合には，そのミューテックスをロックしているタスクに対して推移的な優先度継承が起こる場合がある．

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，AUTOSAR 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/ASP3カーネルにおける規定】

ASP3カーネルでは，優先度継承ミューテックスはサポートしない【ASPS0237】．ただし，優先度継承拡張パッケージを用いると，優先度継承ミューテックスの機能を追加することができる【ASPS0238】．

【TOPPERS/FMP3カーネルにおける規定】

FMP3カーネルでは，優先度継承ミューテックスはサポートしない【FMPS0174】．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルでは，優先度継承ミューテックスはサポートしない【HRPS0245】．

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルでは，優先度継承ミューテックスはサポートしない【HRMPS0121】．

【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．

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．

ミューテックスのロック解除は，ロックしたのと逆順で行わなければならないものとした．

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．

【TOPPERS新世代カーネル統合仕様との関係】

ミューテックス（優先度継承ミューテックスは除く）は標準機能の位置付けとした．そのため，ASP3カーネルとFMP3カーネルにおいても，ミューテックスをサポートしている．

μITRON4.0仕様でサポートしており，TOPPERS新世代カーネル統合仕様でサポートしていなかった優先度継承プロトコルをサポートしている．

ミューテックスのロック解除は，ロックしたのと逆順で行わなければならないものとした．

| | |
|---|---|
| **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 })
```

※ 優先度上限ミューテックス以外の場合には，ceilpriの記述を省略することができる【NGKI2035】.

【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識別名（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（acre_mtxの場合） |

＊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<br>・非タスクコンテキストからの呼出し〔s〕【NGKI2023】<br>・CPUロック状態からの呼出し〔s〕【NGKI2024】 |
| E_RSATR | 予約属性<br>・mtxatrが無効【NGKI2025】<br>・属する保護ドメインの指定が有効範囲外〔sP〕【NGKI2026】<br>・属するクラスの指定が有効範囲外〔sM〕【NGKI2027】<br>・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<br>・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<br>・属する保護ドメイン（または無所属）に対する通常操作1が許可されていない〔sP〕【NGKI3971】 |
| E_MACV | メモリアクセス違反<br>・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<br>・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<br>・mtxidで指定した識別名のオブジェクトが登録済み〔S〕【NGKI2032】 |

【機能】

各パラメータで指定したミューテックスの生成情報に従って，ミューテックスを生成する【NGKI5193】．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を生成する場合で，ceilpriが有効範囲（TMIN_TPRI以上，TMAX_TPRI以下）外の場合には，E_PARエラーとなる【NGKI2037】．

【補足説明】

優先度上限ミューテックス以外の場合には，ceilpriは使用されない．

【TOPPERS/ASP3カーネルにおける規定】

ASP3カーネルの動的生成機能拡張パッケージでは，acre_mtxをサポートする【ASPS0228】．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルの動的生成機能拡張パッケージでは，acre_mtxをサポートする【HRPS0194】．

【μITRON4.0/PX仕様，TOPPERS新世代カーネル統合仕様との関係】

アクセス権を制御するためのアクセスの種別を，保護ドメインに対する通常操作1に変更した．

## 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<br>・クラスの囲みの中に記述されていない〔M〕【NGKI2039】 |

【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/ASP3カーネルにおける規定】

ASP3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【ASPS0222】．

【TOPPERS/HRP3カーネルにおける規定】

HRP3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識別名（SAC_MTXの場合）またはID番号（sac_mtxの場合） |
| 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（sac_mtxの場合） |

＊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<br>・非タスクコンテキストからの呼出し〔s〕【NGKI2044】<br>・CPUロック状態からの呼出し〔s〕【NGKI2045】 |
| E_ID | 不正ID番号<br>・mtxidが有効範囲外〔s〕【NGKI2046】 |
| E_RSATR | 予約属性<br>・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】<br>・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<br>・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<br>・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〕<br>【NGKI2050】 |
| E_MACV | メモリアクセス違反<br>・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〕<br>【NGKI2051】 |
| E_OBJ | オブジェクト状態エラー<br>・対象ミューテックスは静的APIで生成された〔s〕【NGKI2052】<br>・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〕<br>【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/HRP3カーネルにおける規定】

HRP3カーネルの動的生成機能拡張パッケージでは，sac_mtxをサポートする【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<br>・非タスクコンテキストからの呼出し【NGKI2057】<br>・CPUロック状態からの呼出し【NGKI2058】 |
| E_ID | 不正ID番号<br>・mtxidが有効範囲外【NGKI2059】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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<br>・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すべて待ち解除される【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．

【TOPPERS/ASP3カーネルにおける規定】

ASP3カーネルの動的生成機能拡張パッケージでは，del_mtxをサポートする【ASPS0223】．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルの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<br>・非タスクコンテキストからの呼出し【NGKI2070】<br>・CPUロック状態からの呼出し【NGKI2071】<br>・ディスパッチ保留状態（非タスクコンテキストの実行中，CPUロック状態を除く）からの呼出し（ploc_mtxを除く）【NGKI2072】 |
| E_NOSPT | 未サポート機能<br>・制約タスクからの呼出し（ploc_mtxを除く）【NGKI2073】 |
| E_ID | 不正ID番号<br>・mtxidが有効範囲外【NGKI2074】 |
| E_PAR | パラメータエラー<br>・tmoutが無効（tloc_mtxの場合）【NGKI2075】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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<br>・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<br>・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 | 待ち状態の強制解除（ploc_mtxを除く）【NGKI2079】 |
| E_RASTER | タスクの終了要求（ploc_mtxを除く）【NGKI3463】 |

| | |
|---|---|
| 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】．また，自タスクがロックしているミューテックスのリストに，対象ミューテックスが追加される【NGKI2082】．

対象ミューテックスが自タスク以外のタスクによってロックされている場合には，自タスクはミューテックスのロック待ち状態となり，対象ミューテックスの待ち行列につながれる【NGKI2083】．

対象ミューテックスが優先度上限ミューテックスで，その上限優先度より自タスクのベース優先度が高い場合には，E_ILUSEエラーとなる【NGKI2085】．

## unl_mtx　　ミューテックスのロック解除〔T〕【NGKI2086】

【C言語API】

```
ER ercd = unl_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<br>・非タスクコンテキストからの呼出し【NGKI2087】<br>・CPUロック状態からの呼出し【NGKI2088】 |
| E_ID | 不正ID番号<br>・mtxidが有効範囲外【NGKI2089】 |
| E_NOEXS | オブジェクト未登録<br>・対象ミューテックスが未登録〔D〕【NGKI2090】 |
| E_OACV | オブジェクトアクセス違反<br>・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〕【NGKI3273】 |
| E_OBJ | オブジェクト状態エラー<br>・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ミューテックスの中で，最後にロックされたものでない【NGKI3611】 |

【機能】

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

まず，自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る【NGKI2091】．

対象ミューテ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待ち解除される【NGKI2092】．対象ミューテックスは，待ち解除されたタスクによってロックされている状態になる【NGKI2093】．待ち解除されたタスクがロックしているミューテックスのリストに対象ミューテックスが追加される【NGKI2094】．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【NGKI2095】．

待ち行列にタスクが存在しない場合には，対象ミューテックスはロックされていない状態になる【NGKI2096】．

【TOPPERS新世代カーネル統合仕様との関係】

ミューテックスのロック解除を，ロックしたのと逆順で行わない場合にエラーになるように，エラーの条

件を変更した.

## ini_mtx　　ミューテックスの再初期化〔T〕【NGKI2098】

【C言語API】

```
ER ercd = ini_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<br>・非タスクコンテキストからの呼出し【NGKI2099】<br>・CPUロック状態からの呼出し【NGKI2100】 |
| E_ID | 不正ID番号<br>・mtxidが有効範囲外【NGKI2101】 |
| E_NOEXS | オブジェクト未登録<br>・対象ミューテックスが未登録〔D〕【NGKI2102】 |
| E_OACV | オブジェクトアクセス違反<br>・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〕【NGKI2103】 |

【機能】

mtxidで指定したミューテックス（対象ミューテックス）を再初期化する．具体的な振舞いは以下の通り．

対象ミューテックスのロック状態は，ロックされていない状態に初期化される【NGKI2104】．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る【NGKI2105】．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すべて待ち解除される【NGKI2106】．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【NGKI2107】．

【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，ミューテックスが再初期化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，ミューテックスを再初期化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，ミューテックスを再初期化すればよい．

ミューテ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_mtx　　ミューテックスの状態参照〔T〕【NGKI2108】

【C言語API】

```
ER ercd = ref_mtx(ID mtxid, T_RMTX *pk_rmtx)
```

【パラメータ】

| | | |
|---|---|---|
| ID | mtxid | 対象ミューテックスのID番号 |
| T_RMTX * | pk_rmtx | ミューテ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 |

＊ミューテックスの現在状態（パケットの内容）

| | | |
|---|---|---|
| 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 |

【エラーコード】

| | |
|---|---|
| E_CTX | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI2109】<br>・CPUロック状態からの呼出し【NGKI2110】 |
| E_ID | 不正ID番号<br>・mtxidが有効範囲外【NGKI2111】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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<br>・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【NGKI5207】．

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．また，マルチプロセッサ対応カーネルでは，対象ミューテックスの現在状態を参照した直後に，他のプロセッサで実行されている処理により対象ミューテックスの状態が変化する場合もある．

### 4.4.6 メッセージバッファ

メッセージバッファは，指定したサイズ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．メッセージの送受信において，メッセージの単位は保存される．すなわち，送信側が指定したメッセージサイズを，受信側はそのまま受け取る．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，カーネルの管理領域として扱われる【NGKI3294】．

送信待ち行列は，メッセージバッファに対してメッセージが送信できるまで待っている状態（メッセージバッファへの送信待ち状態）のタスクが，メッセージを送信できる順序でつながれているキューである．送信待ち行列は，FIFO順に固定されている【NGKI3523】．

受信待ち行列は，メッセージバッファからメッセージが受信できるまで待っている状態（メッセージバッファ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．受信待ち行列は，FIFO順に固定されている【NGKI3297】．

メッセージバッファ属性に指定できる属性はない【NGKI3521】．そのためメッセージバッファ属性には，TA_NULLを指定しなければならない【NGKI3522】．

メッセージバッファ機能に関連するカーネル構成マクロは次の通り．

| | |
|---|---|
| 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）【NGKI3298】 |

**【使用上の注意】**

マルチプロセッサ対応カーネルでは，送信／受信するメッセージを置くメモリ領域は，他のプロセッサからアクセスできるメモリに置かなければならない［NGKI5210］［NGKI5211］．カーネルは，アプリケーションが指定したメモリ領域がこの制約を満たしているかチェックすることができず，制約を満たさないメモリ領域を渡された場合には，カーネルが誤動作する可能性がある．保護機能対応カーネルにおいては，ユーザタスクがこの制約を満たさないメモリ領域を指定してメッセージバッファ機能のサービスコールを呼び出すと，ユーザタスクがカーネルの誤動作を引き起こせてしまう．この問題を回避するためには，例えば，ユーザタスクからメッセージバッファへのアクセスは許可せず，拡張サービスコール経由でメッセージバッファにアクセスすることとし，拡張サービスコールの入口で不正なメモリ領域を指定していないかチェックするといった方法が考えられる．

【TOPPERS/ASP3カーネルにおける規定】

ASP3カーネルでは，メッセージバッファ機能をサポートしない【ASPS0202】．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，メッセージバッファ機能を追加することができる【ASPS0203】．

【TOPPERS/FMP3カーネルにおける規定】

FMP3カーネルでは，メッセージバッファ機能をサポートしない【FMPS0167】．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルでは，メッセージバッファ機能をサポートする【HRPS0232】．

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルでは，メッセージバッファ機能をサポートする【HRMPS0109】．

【μITRON4.0仕様との関係】

TNUM_MBFID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_MBF**　メッセージバッファの生成〔S〕【NGKI3299】

**acre_mbf**　メッセージバッファの生成〔TD〕【NGKI3300】

【静的API】

```
CRE_MBF(ID mbfid, { ATR mbfatr, uint_t maxmsz, size_t mbfsz, void *mbfmb })
```

※ mbfmbの記述は省略することができる【NGKI3903】.

【C言語API】

```
ER_ID mbfid = acre_mbf(const T_CMBF *pk_cmbf)
```

【パラメータ】

| | | |
|---|---|---|
| ID | mbfid | 生成するメッセージバッファの識別名（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（acre_mbfの場合） |

＊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_t | 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<br>・非タスクコンテキストからの呼出し〔s〕【NGKI3301】<br>・CPUロック状態からの呼出し〔s〕【NGKI3302】 |
| E_RSATR | 予約属性<br>・mbfatrが無効【NGKI3303】<br>・属する保護ドメインの指定が有効範囲外〔sP〕【NGKI3304】<br>・属するクラスの指定が有効範囲外〔sM〕【NGKI3305】<br>・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<br>・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<br>・maxmszが0【NGKI3307】<br>・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<br>・属する保護ドメイン（または無所属）に対する通常操作1が許可されていない〔sP〕【NGKI3972】 |
| E_MACV | メモリアクセス違反<br>・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<br>・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<br>・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<br>・mbfidで指定した識別名のオブジェクトが登録済み〔S〕【NGKI3313】<br>・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ッセージバッファの生成情報に従って，メッセージバッファを生成する【NGKI5194】．mbfszとmbfmbからメッセージバッファ管理領域が設定され，格納されているメッセージがない状態に初期化される【NGKI3314】．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【NGKI3315】．

静的APIにおいては，mbfidはオブジェクト識別名，mbfatr，maxmsz，mbfszは整数定数式パラメータ，mbfmbは一般定数式パラメータである【NGKI3316】．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【NGKI3317】．

mbfmbにNULLを指定するか，静的APIにおいてmbfmbの記述を省略した場合，mbfszで指定したサイズのメッセージバッファ管理領域が，コンフィギュレータまたはカーネルにより確保される【NGKI3318】．mbfsz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【NGKI3319】．

〔**mbfmbにNULL以外を指定した場合**〕

mbfmbにNULL以外を指定した場合，mbfmbとmbfszで指定したメッセージバッファ管理領域は，アプリケーションで確保しておく必要がある【NGKI3320】．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ために，次のマクロを用意している【NGKI3321】．

| | |
|---|---|
| TSZ_MBFMB(msgcnt, msgsz) | msgszで指定したサイズのメッセージを，msgcntで指定した数だけ格納できるメッセージバッファ管理領域のサイズ（バイト数） |
| TCNT_MBFMB(msgcnt, msgsz) | msgszで指定したサイズのメッセージを，msgcntで指定した数だけ格納できるメッセージバッファ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，msgszで指定したサイズのメッセージを，msgcntで指定した数だけ格納できるメッセージバッファ管理領域を確保する方法は次の通り【NGKI3322】．

```
MB_T    <メッセージバッファ管理領域の変数名>[TCNT_MBFMB(msgcnt, msgsz)];
```

この時，mbfszにはTSZ_MBFMB(msgcnt，msgsz)を，mbfmbには<メッセージバッファ管理領域の変数名>を指定する【NGKI3323】．

この方法に従わず，mbfmbとmbfsz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【NGKI3324】．また，保護機能対応カーネルにおいて，mbfmbとmbfszで指定したメッセージバッファ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【NGKI3325】．

なお，mbfmbにNULL以外を指定できないカーネルでは，TCNT_MBFMBは用意していない【NGKI5181】．TSZ_MBFMBは，mbfmbにNULLを指定する場合にも，メッセージバッファ管理領域のサイズを決めるために用いることができる．

【TOPPERS/ASP3カーネルにおける規定】

ASP3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，mbfmb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【ASPS0205】．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルでは，mbfmb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【HRPS0171】．

HRP3カーネルの動的生成機能拡張パッケージでは，acre_mbfをサポートする【HRPS0235】．acre_mbfに対しては，mbfmbにNULL以外を指定できないという制限はない【HRPS0236】．

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルでは，mbfmb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【HRMPS0110】．

【μITRON4.0仕様との関係】

μITRON4.0/PX仕様にあわせて，メッセージバッファの生成情報の最後のパラメータを，mbf（メッセージバッファ領域の先頭番地）から，mbfmb（メッセージバッ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．

メッセージバッファの送信待ち行列をタスクの優先度順にする機能を廃止した．

【μITRON4.0/PX仕様との関係】

アクセス権を制御するためのアクセスの種別を，保護ドメインに対する通常操作1に変更した．

【TOPPERS新世代カーネル統合仕様との関係】

メッセージバッファの送信待ち行列をタスクの優先度順にする機能を廃止した．

アクセス権を制御するためのアクセスの種別を，保護ドメインに対する通常操作1に変更した．

## 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<br>・クラスの囲みの中に記述されていない〔M〕【NGKI3327】 |

【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】．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルの動的生成機能拡張パッケージでは，AID_MBFをサポートする【HRPS0237】．

| | |
|---|---|
| **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識別名（SAC_MBFの場合）またはID番号（sac_mbfの場合） |
| 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（sac_mbfの場合） |

＊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<br>・非タスクコンテキストからの呼出し〔s〕【NGKI3333】<br>・CPUロック状態からの呼出し〔s〕【NGKI3334】 |
| E_ID | 不正ID番号<br>・mbfidが有効範囲外〔s〕【NGKI3335】 |
| E_RSATR | 予約属性<br>・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】<br>・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<br>・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<br>・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<br>・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<br>・対象メッセージバッファは静的APIで生成された〔s〕【NGKI3341】<br>・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/HRP3カーネルにおける規定】

HRP3カーネルの動的生成機能拡張パッケージでは，sac_mbfをサポートする【HRPS0233】.

## 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<br>・非タスクコンテキストからの呼出し【NGKI3346】<br>・CPUロック状態からの呼出し【NGKI3347】 |
| E_ID | 不正ID番号<br>・mbfidが有効範囲外【NGKI3348】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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<br>・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．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルの動的生成機能拡張パッケージでは，del_mbfをサポートする【HRPS0234】．

| | |
|---|---|
| **snd_mbf** | メッセージバッファへの送信〔T〕【NGKI3356】 |
| **psnd_mbf** | メッセージバッファへの送信（ポーリング）〔T〕【NGKI3357】 |
| **tsnd_mbf** | メッセージバッファへの送信（タイムアウト付き）〔T〕【NGKI3358】 |

【C言語API】

```
ER ercd = snd_mbf(ID mbfid, const void *msg, uint_t msgsz)
ER ercd = psnd_mbf(ID mbfid, const void *msg, uint_t msgsz)
ER ercd = tsnd_mbf(ID mbfid, const void *msg, uint_t msgsz, TMO tmout)
```

【パラメータ】

| | | |
|---|---|---|
| ID | mbfid | 対象メッセージバッファのID番号 |
| void * | msg | 送信メッセージの先頭番地 |
| uint_t | msgsz | 送信メッセージのサイズ（バイト数） |
| TMO | tmout | タイムアウト時間（tsnd_mbf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<br>・非タスクコンテキストからの呼出し【NGKI3359】<br>・CPUロック状態からの呼出し【NGKI3360】<br>・ディスパッチ保留状態（非タスクコンテキストの実行中，CPUロック状態を除く）からの呼出し（psnd_mbfを除く）【NGKI3361】 |
| E_NOSPT | 未サポート機能<br>・制約タスクからの呼出し（psnd_mbfを除く）【NGKI3362】 |
| E_ID | 不正ID番号<br>・mbfidが有効範囲外【NGKI3363】 |
| E_PAR | パラメータエラー<br>・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】<br>・tmoutが無効（tsnd_mbfの場合）【NGKI3365】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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<br>・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 | 待ち状態の強制解除（psnd_mbfを除く）【NGKI3370】 |
| E_RASTER | タスクの終了要求（psnd_mbfを除く）【NGKI3464】 |
| 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【NGKI3524】．

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【NGKI3525】．

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，タスクの終了要求やter_tskにより終了した場合や，rel_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】．

マルチプロセッサ対応カーネルでは，psnd_mbfを除いて，msgとmsgszが指すメモリ領域は，メッセージを受信するタスクや，送信待ち行列につながれているタスクを終了または待ち解除するタスクを実行するプロセッサからアクセスできるメモリに置かなければならない【NGKI5210】．

【補足説明】

送信待ち行列の先頭につながれているタスクの強制終了や待ち解除により，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．

| | |
|---|---|
| **rcv_mbf** | メッセージバッファからの受信〔T〕【NGKI3376】 |
| **prcv_mbf** | メッセージバッファからの受信（ポーリング）〔T〕【NGKI3377】 |
| **trcv_mbf** | メッセージバッファからの受信（タイムアウト付き）〔T〕【NGKI3378】 |

【C言語API】

```
ER_UINT msgsz = rcv_mbf(ID mbfid, void *msg)
ER_UINT msgsz = prcv_mbf(ID mbfid, void *msg)
ER_UINT msgsz = trcv_mbf(ID mbfid, void *msg, TMO tmout)
```

【パラメータ】

| | | |
|---|---|---|
| ID | mbfid | 対象メッセージバッファのID番号 |
| void * | msg | 受信メッセージを入れるメモリ領域の先頭番地 |
| TMO | tmout | タイムアウト時間（trcv_mbfの場合） |

【リターンパラメータ】

| | | |
|---|---|---|
| ER_UINT | msgsz | 受信メッセージサイズ（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<br>・非タスクコンテキストからの呼出し【NGKI3379】<br>・CPUロック状態からの呼出し【NGKI3380】<br>・ディスパッチ保留状態（非タスクコンテキストの実行中，CPUロック状態を除く）からの呼出し（prcv_mbfを除く）【NGKI3381】 |
| E_NOSPT | 未サポート機能<br>・制約タスクからの呼出し（prcv_mbfを除く）【NGKI3382】 |
| E_ID | 不正ID番号<br>・mbfidが有効範囲外【NGKI3383】 |
| E_PAR | パラメータエラー<br>・tmoutが無効（trcv_mbfの場合）【NGKI3384】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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<br>・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 | 待ち状態の強制解除（prcv_mbfを除く）【NGKI3389】 |

| | |
|---|---|
| E_RASTER | タスクの終了要求（prcv_mbfを除く）【NGKI3465】 |
| 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順でメッセージバッファ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】．

マルチプロセッサ対応カーネルでは，prcv_mbfを除いて，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は，メッセージを送信するタスクを実行するプロセッサからアクセスできるメモリに置かなければならない【NGKI5211】．

【使用上の注意】

msgで先頭番地を指定する受信メッセージを入れるメモリ領域は，オーバフローを防ぐために，対象メッセージバッファの最大メッセージサイズ分以上を確保しておかなければならない．

## 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<br>・非タスクコンテキストからの呼出し【NGKI3401】<br>・CPUロック状態からの呼出し【NGKI3402】 |
| E_ID | 不正ID番号<br>・mbfidが有効範囲外【NGKI3403】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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】

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_mbf　　メッセージバッファの状態参照〔T〕【NGKI3409】

【C言語API】

```
ER ercd = ref_mbf(ID mbfid, T_RMBF *pk_rmbf)
```

【パラメータ】

| | | |
|---|---|---|
| ID | mbfid | 対象メッセージバッファのID番号 |
| T_RMBF * | pk_rmbf | メッセージバッファ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 |

＊メッセージバッファの現在状態（パケットの内容）

| | | |
|---|---|---|
| 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_t | fmbfsz | メッセージバッファ管理領域中の空き領域のサイズ |

【エラーコード】

| | |
|---|---|
| E_CTX | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI3410】<br>・CPUロック状態からの呼出し【NGKI3411】 |
| E_ID | 不正ID番号<br>・mbfidが有効範囲外【NGKI3412】 |
| E_NOEXS | オブジェクト未登録<br>・対象メッセージバッファが未登録〔D〕【NGKI3413】 |
| E_OACV | オブジェクトアクセス違反<br>・対象メッセージバッファに対する参照操作が許可されていない〔P〕【NGKI3414】 |
| E_MACV | メモリアクセス違反<br>・pk_rmbf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〕【NGKI3415】 |

【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）の現在状態を参照する．参照した現在状態は，pk_rmbfで指定したパケットに返される【NGKI3416】．

対象メッセージバッファ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【NGKI3417】．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【NGKI3418】．

【使用上の注意】

ref_mbf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_mbfを呼び出し，対象メッセージバッファの現在状態を参照した直後に割込みが発生した場合，ref_mbfから戻ってきた時には対象メッセージバッファの状態が変化している可能性があるためである．また，マルチプロセッサ対応カーネルでは，対象メッセージバッファの現在状態を参照した直後に，他のプロセッサで実行されている処理により対象メッセージバッファの状態が変化する場合もある．

### 4.4.7 スピンロック

スピンロックは，マルチプロセッサ対応カーネルにおいて，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同期・通信オブジェクトである．スピンロックは，スピンロックIDと呼ぶID番号によって識別する【NGKI2117】．マルチプロセッサ対応でないカーネルでは，スピンロック機能をサポートしない【NGKI3887】．

プロセッサ間ロックを取得している間は，CPUロック状態にすることで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【NGKI2118】．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【NGKI2119】．ロックの取得を待つ間は，CPUロック解除状態であり，割込みはマスクされない【NGKI2120】．プロセッサ間ロックを取得し，CPUロック状態に遷移することを，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，CPUロック状態を解除することを，スピンロックを返却するという．

タスクが取得したスピンロック（タスクが呼び出した拡張サービスコールが取得したスピンロックを含む）を返却せずに終了した場合や，カーネル管理の割込みハンドラ，割込みサービスルーチン，タイムイベントハンドラ，カーネル管理のCPU例外ハンドラが取得したスピンロック（これらの処理単位が呼び出した拡張サービスコールが取得したスピンロックを含む）を返却せずにリターンした場合には，カーネルによってスピンロックが返却される【NGKI5167】．一方，拡張サービスコールからのリターンでは，スピンロックは返却されない【NGKI2123】．

各スピンロックが持つ情報は次の通り【NGKI2124】．

- スピンロック属性
- ロック状態（取得されている状態と取得されていない状態）
- 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

スピンロック属性には，次の属性を指定することができる【NGKI5168】．

| | | |
|---|---|---|
| TA_NATIVE | 0x01U | エミュレーションによって実現してはならない |

ターゲットによっては，ターゲット定義のスピンロック属性を指定できる場合がある【NGKI5175】．

カーネルが提供するスピンロック（ここでは，スピンロックオブジェクトと呼ぶ）は，ハードウェアが持つスピンロックを実現するための機構を用いて実現される．ターゲットハードウェアが持つ機構で実現できるスピンロックオブジェクトの数にターゲット定義の上限がある場合には，カーネルに登録されたスピンロックオブジェクトのすべてを，ハードウェアが持つ機構で実現できるとは限らない【NGKI5169】．そのような場合には，スピンロックオブジェクトの一部を，カーネル内部用のスピンロックを用いてエミュレートする【NGKI5170】．

カーネルに登録されたスピンロックオブジェクトの内，どのスピンロックオブジェクトをエミュレーションによって実現するかは，コンフィギュレータまたはカーネルが決定する【NGKI5171】．その際に，TA_NATIVE属性のスピンロックオブジェクトは，エミュレーションによって実現してはならないものと扱う【NGKI5172】．TA_NATIVE属性のスピンロックオブジェクトを，ターゲットハードウェアが持つ機構で実現できない場合，コンフィギュレータまたはカーネルはエラーを報告する【NGKI5173】．

スピンロック機能に関連するカーネル構成マクロは次の通り．

| | |
|---|---|
| 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スピンロックの数に一致）【NGKI2127】 |

【補足説明】

CPUロック状態では，スピンロックを取得するサービスコールを呼び出すことができないため，スピンロックを取得しているプロセッサが，さらにスピンロックを取得することはできない．そのため，1つの処理単位が，複数のスピンロックを取得した状態になることはできない．

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はtrueを返す），CPU例外ハンドラ中でunl_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unl_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新世代カーネル統合仕様との関係】

エミュレーションによりスピンロックが実現される場合を規定し，TA_NATIVE属性を導入した．

| | |
|---|---|
| **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識別名（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（acre_spnの場合） |

＊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<br>・非タスクコンテキストからの呼出し〔s〕【NGKI2130】<br>・CPUロック状態からの呼出し〔s〕【NGKI2131】 |
| E_RSATR | 予約属性<br>・spnatrが無効【NGKI2132】<br>・属する保護ドメインの指定が有効範囲外〔sP〕【NGKI2133】<br>・属するクラスの指定が有効範囲外〔s〕【NGKI2134】<br>・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<br>・属する保護ドメイン（または無所属）に対する通常操作1が許可されていない〔sP〕【NGKI3973】 |
| E_MACV | メモリアクセス違反<br>・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<br>・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<br>・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<br>・spnidで指定した識別名のオブジェクトが登録済み〔S〕【NGKI2139】 |

【機能】

各パラメータで指定したスピンロックの生成情報に従って，スピンロックを生成する【NGKI5195】．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】．

TA_NATIVE属性のスピンロックを生成する場合で，ターゲットハードウェアが持つ機構が不足する場合には，E_NORESエラーとなる【NGKI5174】．

【TOPPERS新世代カーネル統合仕様との関係】

E_NORESエラーは，TA_NATIVE属性のスピンロックの生成時にのみ発生することとした．

アクセス権を制御するためのアクセスの種別を，保護ドメインに対する通常操作1に変更した．

## 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<br>・クラスの囲みの中に記述されていない【NGKI2145】 |

【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識別名（SAC_SPNの場合）またはID番号（sac_spnの場合） |
| 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（sac_spnの場合） |

＊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<br>・非タスクコンテキストからの呼出し〔s〕【NGKI2150】<br>・CPUロック状態からの呼出し〔s〕【NGKI2151】 |
| E_ID | 不正ID番号<br>・spnidが有効範囲外〔s〕【NGKI2152】 |
| E_RSATR | 予約属性<br>・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】<br>・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<br>・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<br>・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<br>・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<br>・対象スピンロックは静的APIで生成された〔s〕【NGKI2158】<br>・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<br>・非タスクコンテキストからの呼出し【NGKI2163】<br>・CPUロック状態からの呼出し【NGKI2164】 |
| E_ID | 不正ID番号<br>・spnidが有効範囲外【NGKI2165】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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<br>・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】．

【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〔TIM〕【NGKI3540】

【C言語API】

```
ER ercd = loc_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<br>・CPUロック状態からの呼出し【NGKI2174】 |
| E_ID | 不正ID番号<br>・spnidが有効範囲外【NGKI2175】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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〔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）〔TIM〕【NGKI3541】

【C言語API】

```
ER ercd = try_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<br>・CPUロック状態からの呼出し【NGKI2188】 |
| E_ID | 不正ID番号<br>・spnidが有効範囲外【NGKI2189】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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〔P〕【NGKI2191】 |
| E_OBJ | オブジェクト状態エラー<br>・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〔TIM〕【NGKI3542】

【C言語API】

```
ER ercd = unl_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_ID | 不正ID番号<br>・spnidが有効範囲外【NGKI2200】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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〔P〕【NGKI2202】 |
| E_ILUSE | サービスコール不正使用<br>・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】．

## 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<br>・非タスクコンテキストからの呼出し【NGKI2207】<br>・CPUロック状態からの呼出し【NGKI2208】 |
| E_ID | 不正ID番号<br>・spnidが有効範囲外【NGKI2209】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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<br>・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．また，マルチプロセッサ対応

カーネルでは，対象スピンロックの現在状態を参照した直後に，他のプロセッサで実行されている処理により対象スピンロックの状態が変化する場合もある．

## 4.5 メモリプール管理機能

【TOPPERS/SSP3カーネルにおける規定】

SSP3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】

可変長メモリプール機能を廃止した.

【仕様決定の理由】

可変長メモリプール機能を廃止したのは，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 })
```

※ mpfとmpfmbの記述は省略することができる【NGKI3904】.

【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識別名（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（acre_mpfの場合） |

＊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<br>・非タスクコンテキストからの呼出し〔s〕【NGKI2223】<br>・CPUロック状態からの呼出し〔s〕【NGKI2224】 |
| E_RSATR | 予約属性<br>・mpfatrが無効【NGKI2225】<br>・属する保護ドメインの指定が有効範囲外〔sP〕【NGKI2226】<br>・属するクラスの指定が有効範囲外〔sM〕【NGKI2227】<br>・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<br>・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<br>・blkcntが0【NGKI2229】<br>・blkszが0【NGKI2230】<br>・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<br>・属する保護ドメイン（または無所属）に対する通常操作1が許可されていない〔sP〕【NGKI3974】 |
| E_MACV | メモリアクセス違反<br>・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<br>・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<br>・固定長メモリプール領域が確保できない【NGKI2234】<br>・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<br>・mpfidで指定した識別名のオブジェクトが登録済み〔S〕【NGKI2236】<br>・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の生成情報に従って，固定長メモリプールを生成する【NGKI5196】．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にNULLを指定するか，静的APIにおいてmpfの記述を省略した場合，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が属する保護ドメイン向け（固定長メモリプールが無所属の場合は，保護ドメイン共通．マルチプロセッサ対応カーネルの場合には，さらに，固定長メモリプールが属するクラス向け）の標準RAMリージョン内に配置され，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】．

ただし，保護機能対応カーネルで手動メモリ配置の場合，mpfをNULLとしたり，静的APIにおいてmpfの記述を省略することができるのは，カーネルドメインに属し，アクセス許可ベクタがデフォルトの値の固定長メモリプールの場合のみである．そうでない場合には，E_PARエラーとなる【NGKI3908】．

mpfmbにNULLを指定するか，静的APIにおいてmpfmbの記述を省略した場合，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】．固定長メモリプール領域をアプリケーションで確保するため

に，次のデータ型とマクロを用意している【NGKI2245】．

| | |
|---|---|
| MPF_T | 固定長メモリプール領域を確保するためのデータ型 |

| | |
|---|---|
| COUNT_MPF_T(blksz) | 固定長メモリブロックのサイズがblkszの固定長メモリプール領域を確保するために，固定長メモリブロック1つあたりに必要なMPF_T型の配列の要素数 |
| ROUND_MPF_T(blksz) | 要素数COUNT_MPF_T(blksz)のMPF_T型の配列のサイズ（blkszを，MPF_T型のサイズの倍数になるように大きい方に丸めた値） |

これらを用いて，blkszで指定したサイズの固定長メモリブロックを，blkcntで指定した数だけ獲得できる固定長メモリプール領域を確保する方法は次の通り【NGKI2246】．

```
MPF_T    <固定長メモリプール領域の変数名>[(blkcnt) * COUNT_MPF_T(blksz)];
```

この時，mpfには<固定長メモリプール領域の変数名>を指定する【NGKI2247】．

これ以外の方法で固定長メモリプール領域を確保する場合には，上記の配列と同じサイズのメモリ領域を確保しなければならない【NGKI2248】．また，その先頭番地がターゲット定義の制約に合致していなければならない．mpf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【NGKI2249】．

保護機能対応カーネルでは，アプリケーションで確保する固定長メモリプール領域は，カーネルに登録されたメモリオブジェクトに含まれていなければならない．指定した固定長メモリプール領域が，カーネルに登録されたメモリオブジェクトに含まれていない場合，E_OBJエラーとなる【NGKI2251】．

**〔mpfmbにNULL以外を指定した場合〕**

mpfmbにNULL以外を指定した場合，mpfmbを先頭番地とする固定長メモリプール管理領域は，アプリケーションで確保しておく必要がある【NGKI2252】．固定長メモリプール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【NGKI2253】．

| | |
|---|---|
| TSZ_MPFMB(blkcnt) | blkcntで指定した数の固定長メモリブロックを管理することができる固定長メモリプール管理領域のサイズ（バイト数） |
| TCNT_MPFMB(blkcnt) | blkcntで指定した数の固定長メモリブロックを管理することができる固定長メモリプール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，blkcntで指定した数の固定長メモリブロックを管理することができる固定長メモリプール管理領域を確保する方法は次の通り【NGKI2254】．

```
MB_T     <固定長メモリプール管理領域の変数名>[TCNT_MPFMB(blkcnt)];
```

この時，mpfmbには<固定長メモリプール管理領域の変数名>を指定する【NGKI2255】．

この方法に従わず，mpfmb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【NGKI2256】．また，保護機能対応カーネルにおいて，mpfmbで指定した固定長メモリプール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【NGKI2257】．

なお，mpfmbにNULL以外を指定できないカーネルでは，TSZ_MPFMBとTCNT_MPFMBは用意していない【NGKI5182】．

【使用上の注意】

保護機能対応カーネルにおいて，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場合には，固定長メモリプール領域が属する保護ドメインとアクセス権の設定は変更されない．これらを適切に設定することは，アプリケーションの責任である．

マルチプロセッサ対応カーネルにおいては，固定長メモリプール領域は，そこから獲得した固定長メモリブロックを使用するタスクがアクセスできるメモリに置かなければならないが，そのようにすることはアプリケーションの責任である．特に，保護機能とマルチプロセッサの両方に対応するカーネルにおいて，固定長メモリプール領域をコンフィギュレータに確保させる場合には，固定長メモリプールが属する保護ドメイン・クラス向けの標準RAMリージョン内に配置されるが，このメモリリージョンが，固定長メモリブロックを使用するタスクがアクセスできるものとすることも，アプリケーションの責任である．

【TOPPERS/ASP3カーネルにおける規定】

ASP3カーネルでは，mpfmb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【ASPS0166】．

ASP3カーネルの動的生成機能拡張パッケージでは，acre_mpfをサポートする【ASPS0167】．acre_mpfに対しては，mpfmbにNULL以外を指定できないという制限はない【ASPS0168】．

【TOPPERS/FMP3カーネルにおける規定】

FMP3カーネルでは，mpfmb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【FMPS0144】．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルでは，mpfmb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【HRPS0138】．

HRP3カーネルの動的生成機能拡張パッケージでは，acre_mpfをサポートする【HRPS0197】．acre_mpfに対しては，mpfmbにNULL以外を指定できないという制限はない【HRPS0198】．ただし，mpfにNULLが指定されるとカーネルが固定長メモリプール領域を確保する機能はサポートしない．mpfにNULLを指定した場合には，E_NOSPTエラーとなる【HRPS0199】．

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルでは，mpfmb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【HRMPS0111】．

【μITRON4.0仕様との関係】

mpfのデータ型をMPF_T　　*に変更した．COUNT_MPF_TとROUND_MPF_Tを新設し，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規定した．また，μITRON4.0/PX仕様にあわせて，固定長メモリプールの生成情報に，mpfmbを追加した．

【μITRON4.0/PX仕様との関係】

TCNT_MPFMBを新設し，固定長メモリプール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．

アクセス権を制御するためのアクセスの種別を，保護ドメインに対する通常操作1に変更した．

【TOPPERS新世代カーネル統合仕様との関係】

アクセス権を制御するためのアクセスの種別を，保護ドメインに対する通常操作1に変更した．

【未決定事項】

保護機能対応カーネルにおいて，サービスコール（acre_mpf）で，mpfにNULLを指定して固定長メモリプールを生成する場合の固定長メモリプール領域の確保方法については，今後の課題である．

## AID_MPF　割付け可能な固定長メモリプールIDの数の指定〔SD〕【NGKI2258】

【静的API】

```
AID_MPF(uint_t nompf)
```

【パラメータ】

| | | |
|---|---|---|
| uint_t | nompf | 割付け可能な固定長メモリプールIDの数 |

【エラーコード】

| | |
|---|---|
| E_RSATR | 予約属性<br>・クラスの囲みの中に記述されていない〔M〕【NGKI2259】 |

【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/ASP3カーネルにおける規定】

ASP3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/HRP3カーネルにおける規定】

HRP3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識別名（SAC_MPFの場合）またはID番号（sac_mpfの場合） |
| 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（sac_mpfの場合） |

＊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<br>・非タスクコンテキストからの呼出し〔s〕【NGKI2264】<br>・CPUロック状態からの呼出し〔s〕【NGKI2265】 |
| E_ID | 不正ID番号<br>・mpfidが有効範囲外〔s〕【NGKI2266】 |
| E_RSATR | 予約属性<br>・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】<br>・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<br>・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<br>・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<br>・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<br>・対象固定長メモリプールは静的APIで生成された〔s〕【NGKI2272】<br>・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/HRP3カーネルにおける規定】

HRP3カーネルの動的生成機能拡張パッケージでは，sac_mpfをサポートする【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<br>・非タスクコンテキストからの呼出し【NGKI2278】<br>・CPUロック状態からの呼出し【NGKI2279】 |
| E_ID | 不正ID番号<br>・mpfidが有効範囲外【NGKI2280】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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<br>・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】．

【TOPPERS/ASP3カーネルにおける規定】

ASP3カーネルの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/HRP3カーネルにおける規定】

HRP3カーネルの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<br>・非タスクコンテキストからの呼出し【NGKI2290】<br>・CPUロック状態からの呼出し【NGKI2291】<br>・ディスパッチ保留状態（非タスクコンテキストの実行中，CPUロック状態を除く）からの呼出し（pget_mpfを除く）【NGKI2292】 |
| E_NOSPT | 未サポート機能<br>・制約タスクからの呼出し（pget_mpfを除く）【NGKI2293】 |
| E_ID | 不正ID番号<br>・mpfidが有効範囲外【NGKI2294】 |
| E_PAR | パラメータエラー<br>・tmoutが無効（tget_mpfの場合）【NGKI2295】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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<br>・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 | 待ち状態の強制解除（pget_mpfを除く）【NGKI2300】 |
| E_RASTER | タスクの終了要求（pget_mpfを除く）【NGKI3466】 |
| 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<br>・非タスクコンテキストからの呼出し【NGKI2305】<br>・CPUロック状態からの呼出し【NGKI2306】 |
| E_ID | 不正ID番号<br>・mpfidが有効範囲外【NGKI2307】 |
| E_PAR | パラメータエラー<br>・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<br>・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<br>・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<br>・非タスクコンテキストからの呼出し【NGKI2315】<br>・CPUロック状態からの呼出し【NGKI2316】 |
| E_ID | 不正ID番号<br>・mpfidが有効範囲外【NGKI2317】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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】

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<br>・非タスクコンテキストからの呼出し【NGKI2324】<br>・CPUロック状態からの呼出し【NGKI2325】 |
| E_ID | 不正ID番号<br>・mpfidが有効範囲外【NGKI2326】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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<br>・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．また，マルチプロセッサ対応カーネルでは，対象固定長メモリプールの現在状態を参照した直後に，他のプロセッサで実行されている処理により対象固定長メモリプールの状態が変化する場合もある．

## 4.6 時間管理機能

時間管理機能は，タイマを制御して時間を管理し，タイムイベントの処理を行うための機能である．

周期通知とアラーム通知は，タイムイベントをアプリケーションに通知するための機能（タイムイベント通知）である．タイムイベントが発生すると，それらの生成時に指定された通知処理を行う【NGKI3688】．

【TOPPERS/SSP3カーネルにおける規定】

SSP3カーネルでは，時間管理機能をサポートしない【SSPS0129】．

【μITRON4.0仕様，TOPPERS新世代カーネル統合仕様との関係】

タイムイベントの通知処理を導入し，周期ハンドラとアラームハンドラの機能を拡張した．機能の名称をそれぞれ周期通知とアラーム通知に変更した．

### 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，周期通知の通知，アラーム通知の通知，システム周期の切換えに使用される時刻を管理するカーネルオブジェクトで，システムに1つのみ存在する【NGKI3603】．システム時刻は，64ビット（ただし，64ビットの整数型がサポートされていないターゲットでは，32ビット）の符号無しの整数型であるSYSTIM型で表され，単位はマイクロ秒である［NGKI0548］．

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】．高分解能タイマを用いてカーネルにより管理・更新され，SYSTIM型で表せる最大値を超えると0に戻される【NGKI3559】．システム時刻の精度はターゲットに依存する【NGKI2336】．

システム時刻をきっかけに発生するタイムイベントの処理は，高分解能タイマ割込みによって実行される．そのため，高分解能タイマ割込みのマスクや，高分解能タイマ割込みより優先される処理によって，高分解能タイマ割込みの処理が遅れると，タイムイベントの処理は遅れることになる．また，同じシステム時刻または短い時間間隔で複数のタイムイベントが発生した場合にも，タイムイベントの処理は遅れることになる．高分解能タイマ割込みの処理が大幅に遅れたり，プロセッサが処理できる以上の頻度でタイムイベントが発生すると，タイムイベントの処理が大幅に遅れ，カーネルの正しい動作が保証できなくなる．タイムイベントの処理が大幅に遅れる原因はアプリケーションによって作られるものであり，このような状況が起こらないことを保証するか，起こった場合に適切に対処するのは，アプリケーションの責任である．具体的には，タイムイベントの処理が2秒以上遅れる状況では，カーネルの動作は保証されない【NGKI3608】．

同じシステム時刻に複数のイベントを処理すべき状況では，それらの処理の間の処理順序は規定されない【NGKI3560】．

〔**ドリフトの調整機能**〕

ドリフトの調整機能は，システム時刻を外部時刻に徐々に合わせることと，高分解能タイマを駆動するクロックの発振周波数の誤差を補正することを目的にした機能である．

ドリフト量は，高分解能タイマの進みに対して，システム時刻をどの程度早く（負の値の場合には，遅く）進めるかを指定するパラメータで，ppm単位でカーネルに設定する【NGKI3593】．例えば，ドリフト量を10ppmに設定すると，高分解能タイマが1,000,000カウントアップする間に，システム時刻は1,000,010マイクロ秒進む．逆に，ドリフト量を-10ppmに設定すると，高分解能タイマが1,000,000カウントアップする間に，システム時刻は999,990マイクロ秒進む．カーネルに設定できるドリフト量は，±10%以内の範囲に制限する【NGKI3594】．

〔高分解能タイマの参照〕

性能評価に用いることを目的に，高分解能タイマの状態を参照し，抽象化されたカウント値に変換して返す機能を用意している．具体的には，高分解能タイマのカウント値は，0から，1マイクロ秒に1のペースでカウントアップし（1ずつ連続してカウントアップするとは限らない），高分解能タイマのタイマ周期に達したら0に戻るものと抽象化される【NGKI3580】．高分解能タイマのタイマ周期とカウントアップ毎の進み幅はターゲット定義である【NGKI3574】．高分解能タイマのカウント値は，32ビットまたは64ビット（いずれであるかはターゲット定義）の符号無しの整数型であるHRTCNT型で表される【NGKI3573】．高分解能タイマの時間精度は，ターゲットに依存する【NGKI3575】．

〔カーネル構成マクロ〕

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．

| | |
|---|---|
| TMIN_ADJTIM | システム時刻の調整時間の最小値（＝-1,000,000）【NGKI3561】 |
| TMAX_ADJTIM | システム時刻の調整時間の最大値（＝1,000,000） |

| | |
|---|---|
| TMIN_DRIFT | ドリフト量の最小値（＝-100,000）【NGKI3562】 |
| TMAX_DRIFT | ドリフト量の最大値（＝100,000） |

| | |
|---|---|
| TCYC_HRTCNT | 高分解能タイマのタイマ周期【NGKI3578】 |
| TSTEP_HRTCNT | 高分解能タイマのカウント値の進み幅【NGKI3579】 |

TCYC_HRTCNTは，HRTCNT型が32ビット符号無しの整数型であり，高分解能タイマのタイマ周期が2^32でない場合にのみマクロ定義される【NGKI5187】．

【TOPPERS/ASP3カーネルにおける規定】

ASP3カーネルでは，ドリフトの調整機能はサポートしない【ASPS0224】．ただし，ドリフト調整機能拡張パッケージを用いると，ドリフトの調整機能を追加することができる【ASPS0225】．

【TOPPERS/FMP3カーネルにおける規定】

FMP3カーネルでは，ドリフトの調整機能はサポートしない【FMPS0171】．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルでは，ドリフトの調整機能はサポートしない【HRPS0228】．ただし，ドリフト調整機能拡張パッケージを用いると，ドリフトの調整機能を追加することができる【HRPS0229】．

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルでは，ドリフトの調整機能はサポートしない【HRMPS0112】．

【μITRON4.0仕様との関係】

システム時刻の更新はカーネル内で実現することとし，タイムティックを供給するサービスコール（isig_tim）は廃止した．システム時刻を設定するサービスコール（set_tim）のパラメータを変更した．

システム時刻を調整するサービスコール（adj_tim）を追加した．また，ドリフトの調整機能と，ドリフト量を設定するサービスコール（set_dft）を追加した．

性能評価に用いるために，高分解能タイマを参照するサービスコール（fch_hrt）を追加した．

【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．

【TOPPERS新世代カーネル統合仕様との関係】

TOPPERS新世代カーネル統合仕様で廃止していたシステム時刻を設定するサービスコール（set_tim）を復活させた．

システム時刻を調整するサービスコール（adj_tim）を追加した．また，ドリフトの調整機能と，ドリフト量を設定するサービスコール（set_dft）を追加した．

システム時刻がマイクロ秒単位となったことから，性能評価用システム時刻の概念を廃止し，それを参照するサービスコール（get_utm）を廃止した．性能評価に用いるために，高分解能タイマを参照するサービスコール（fch_hrt）を追加した．

ローカルタイマ方式を廃止したことに伴い，ローカルタイマ方式の場合の規定を削除した．

## set_tim　　システム時刻の設定〔T〕【NGKI3563】

【C言語API】

```
ER ercd = set_tim(SYSTIM systim)
```

【パラメータ】

| | | |
|---|---|---|
| SYSTIM | systim | システム時刻 |

【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<br>・非タスクコンテキストからの呼出し【NGKI3564】<br>・CPUロック状態からの呼出し【NGKI3565】 |
| E_OACV | オブジェクトアクセス違反<br>・システム状態1に対する管理操作が許可されていない〔P〕【NGKI3566】 |

【機能】

システム時刻の現在値を，systimで指定した時刻に設定する【NGKI3567】．

【補足説明】

set_timによってシステム時刻を変更しても，タイムイベントが発生する実時刻は変化しない．例えば，タイムイベントを5ミリ秒後に発生するように設定した後に，set_timによりシステム時刻を1ミリ秒進めても，タイムイベントは最初に設定した時刻の5ミリ秒後に発生する．

【使用上の注意】

［NGKI3592］の規定より，adj_timによってシステム時刻を戻し，get_timで参照するシステム時刻の進みが止まっている間に，set_timによりシステム時刻を設定すると，期待しない結果になる可能性があるので注意すること．

【μITRON4.0仕様との関係】

設定するシステム時刻を，そのままパラメータとして渡すように変更した．

【TOPPERS新世代カーネル統合仕様との関係】

TOPPERS新世代カーネル統合仕様に定義されていないサービスコールである．

## 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<br>・非タスクコンテキストからの呼出し【NGKI2350】<br>・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<br>・システム状態1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<br>・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】．

システム時刻を調整するサービスコール（adj_tim）によってシステム時刻を戻した場合でも，get_timで参照するシステム時刻が戻ることはない．具体的には，adj_timによってシステム時刻を戻した場合，get_timは，システム時刻が最も進んでいた時のシステム時刻を返す【NGKI3591】．すなわち，get_timで参照するシステム時刻は，システム時刻を戻した時間の間，進みが止まることになる．get_timで参照するシステム時刻の進みが止まっている間に，set_timによりシステム時刻を設定した場合でも，get_timで参照するシステム時刻の進みが止まっている時間は変化しない【NGKI3592】．

【補足説明】

システム時刻を設定するサービスコール（set_tim）によってシステム時刻を設定した場合には，get_timで参照するシステム時刻が戻る可能性がある．

## adj_tim　　システム時刻の調整〔TI〕【NGKI3581】

【C言語API】

```
ER ercd = adj_tim(int32_t adjtim)
```

【パラメータ】

| | | |
|---|---|---|
| int_t | adjtim | システム時刻の調整量 |

【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<br>・CPUロック状態からの呼出し【NGKI3583】 |
| E_PAR | パラメータエラー<br>・adjtimが有効範囲（TMIN_ADJTIM以上，TMAX_ADJTIM以下）外【NGKI3584】 |
| E_OACV | オブジェクトアクセス違反<br>・システム状態1に対する管理操作が許可されていない〔P〕【NGKI3585】 |
| E_OBJ | オブジェクト状態エラー<br>・条件については機能の項を参照 |

【機能】

システム時刻を，adjtimで指定した時間だけ調整する．具体的な振舞いは以下の通り．

まず，システム時刻の現在値に，adjtimで指定した時間を加える【NGKI3586】．すなわち，adjtimが正の値の時はシステム時刻は進み，adjtimが負の時はシステム時刻は戻ることになる．

システム時刻の調整に伴って，システム時刻の経過をきっかけに発生するタイムイベントが発生するまでの相対時間も調整する【NGKI3587】．例えば，タイムイベントを5ミリ秒後に発生するように設定した後に，adj_timによりシステム時刻を1ミリ秒進めると，タイムイベントは最初に設定した時刻の4ミリ秒後に発生する．

adj_timの1回の呼出しで調整できる時間範囲には，adjtimの有効範囲により±1秒以内という制限が設けられているが，adj_timを連続して呼び出した場合，未処理の調整時間の累積値に対しても，次の制限が設けられている．adj_timで時間を進める場合（adjtimが正の場合）には，1秒以上過去の発生時刻を持つタイムイベントが処理されずに残っている場合に，E_OBJエラーとなる【NGKI3588】．時間を戻す場合（adjtimが負の場合）には，現在のシステム時刻が，システム時刻が最も進んでいた時のシステム時刻より1秒以上戻っている場合に，E_OBJエラーとなる【NGKI3589】．

【使用上の注意】

adj_timによりシステム時刻を進めた場合，多くのタイムイベントが一斉に発生する可能性がある．例えば，1ミリ秒周期の周期ハンドラが動作している時に，adj_timによりシステム時刻を100ミリ秒進めると，周期ハンドラが連続して100回呼び出される．これにより発生する負荷が問題にならないように，注

意が必要である．

【μITRON4.0仕様，TOPPERS新世代カーネル統合仕様との関係】

μITRON4.0仕様およびTOPPERS新世代カーネル統合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】

adj_timの1回の呼出しで調整できる時間範囲と未処理の調整時間の累積値に対して制限を設けているのは，タイムイベントを発生させる時刻を，カーネル内部でそれを表現するデータ型の範囲を超えないようにするためである．

## set_dft　　ドリフト量の設定〔TI〕【NGKI3596】

【C言語API】

```
ER ercd = set_dft(int32_t drift)
```

【パラメータ】

| | | |
|---|---|---|
| int_t | drift | ドリフト量（単位はppm） |

【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<br>・CPUロック状態からの呼出し【NGKI3598】 |
| E_PAR | パラメータエラー<br>・driftが有効範囲（TMIN_DRIFT以上，TMAX_DRIFT以下）外【NGKI3599】 |
| E_OACV | オブジェクトアクセス違反<br>・システム状態1に対する管理操作が許可されていない〔P〕【NGKI3600】 |

【機能】

ドリフト量を，driftで指定した値に設定する【NGKI3601】．

【補足説明】

set_dftによって設定したドリフト量は，次にドリフト量を設定するまで有効である．そのため，ドリフトの調整機能を用いて，システム時刻を指定した時間だけ進めたい（または遅らせたい）場合，どれだけの時間をかけてシステム時刻を調整するかを決め，必要なドリフト量を算出してset_dftにより設定した後，調整が終わるタイミングで再度set_dftを呼び出し，ドリフト量を0（または元の値）に戻すことが必要である．

【TOPPERS/ASP3カーネルにおける規定】

ASP3カーネルでは，set_dftをサポートしない【ASPS0226】．ただし，ドリフト調整機能拡張パッケージでは，set_dftをサポートする【ASPS0227】．

【TOPPERS/FMP3カーネルにおける規定】

FMP3カーネルでは，set_dftをサポートしない【FMPS0172】．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルでは，set_dftをサポートしない【HRPS0230】．ただし，ドリフト調整機能拡張パッケージでは，set_dftをサポートする【HRPS0231】．

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルでは，set_dftをサポートしない【HRMPS0113】.

【μITRON4.0仕様，TOPPERS新世代カーネル統合仕様との関係】

μITRON4.0仕様およびTOPPERS新世代カーネル統合仕様に定義されていないサービスコールである.

## fch_hrt　　高分解能タイマの参照〔TI〕【NGKI3569】

【C言語API】

```
HRTCNT hrtcnt = fch_hrt()
```

【パラメータ】

なし

【リターンパラメータ】

| | | |
|---|---|---|
| HRTCNT | hrtcnt | 高分解能タイマの現在のカウント値 |

【機能】

高分解能タイマの現在のカウント値を読み出す．具体的には，高分解能タイマの現在の状態を参照し，抽象化されたカウント値に変換して，返値として返す【NGKI3571】．

fch_hrtは，任意の状態から呼び出すことができる【NGKI3572】．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．

【使用方法】

fch_hrt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，fch_hrtを用いて高分解能タイマの現在のカウント値を読み出す．高分解能タイマの周期を考慮してその差を求めることで，対象プログラムの処理時間に，fch_hrt自身の処理時間を加えたものが得られる．

【使用上の注意】

fch_hrt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．

fch_hrt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，fch_hrt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．

【μITRON4.0仕様，TOPPERS新世代カーネル統合仕様との関係】

μITRON4.0仕様およびTOPPERS新世代カーネル統合仕様に定義されていないサービスコールである．

### 4.6.2 タイムイベントの通知処理

タイムイベントの通知は，次のいずれかの方法で行うことができる【NGKI3689】．

- タイムイベントハンドラの呼出し
- 変数の設定
- 変数のインクリメント
- タスクの起動（act_tsk）
- タスクの起床（wup_tsk）
- セマフォの資源の返却（sig_sem）
- イベントフラグのセット（set_flg）
- データキューへの送信（psnd_dtq）

これらの通知方法の中で，タイムイベントハンドラの呼出し，変数の設定，変数のインクリメント以外の方法では，通知のためのサービスコールがエラーを返し，タイムイベントの通知に失敗する場合がある．タイムイベントの通知に失敗した場合には，次のいずれかの方法で，エラーを通知することができる【NGKI3690】．

- 変数の設定
- 変数のインクリメント
- タスクの起動（act_tsk）
- タスクの起床（wup_tsk）
- セマフォの資源の返却（sig_sem）
- イベントフラグのセット（set_flg）
- データキューへの送信（psnd_dtq）

ただし，これらのエラーの通知方法の中で，エラーコードを伝えることができるのは，変数の設定とデータキューへの送信のみである．

変数の設定と変数のインクリメント以外の方法では，通知のためのサービスコールがエラーを返し，エラーの通知に失敗する場合があるが，エラーの通知に失敗しても何も行われない【NGKI3691】．

#### (1) タイムイベントハンドラの呼出しによる通知

タイムイベントハンドラの呼出しによるタイムイベントの通知では，アプリケーションが用意したプログラムを呼び出すことにより通知を行う【NGKI3692】．この通知処理を登録する場合には，付随情報として，タイムイベントハンドラの拡張情報と先頭番地を指定する．タイムイベントハンドラの先頭番地がプログラムの先頭番地として正しくない場合には，E_PARエラーとなる【NGKI3693】．

周期通知によって呼び出されるタイムイベントハンドラを周期ハンドラ，アラーム通知によって呼び出されるタイムイベントハンドラをアラームハンドラと呼ぶ．

保護機能対応カーネルにおいては，カーネルドメインに属する周期通知またはアラーム通知のみが，タイムイベントハンドラの呼出しによる通知を用いることができる【NGKI3694】．すなわち，周期ハンドラとアラームハンドラは，カーネルドメインに属する．ユーザドメインに属する周期通知またはアラーム通知の生成時や，タイムウィンドウの登録時に，タイムイベントハンドラの呼出しによる通知を指定した場合は，E_OACVエラーとなる【NGKI3695】．

C言語によるタイムイベントハンドラの記述形式は次の通り【NGKI3696】．

```
void time_event_handler(EXINF exinf)
{
    タイムイベントハンドラ本体
}
```

exinfには，タイムイベントハンドラの拡張情報が渡される【NGKI3697】．

(2) 変数の設定による通知

変数の設定によるタイムイベントの通知では，アプリケーションが用意したintptr_t型の変数に，指定した値を設定することにより通知を行う【NGKI3698】．この通知処理を登録する場合には，付随情報として，変数の番地と設定する値を指定する．変数の番地がintptr_t型の変数の番地として正しくない場合には，E_PARエラーとなる【NGKI3699】．

また，変数の設定によるエラーの通知では，アプリケーションが用意したintptr_t型の変数に，エラーコードを設定することにより通知を行う【NGKI3700】．この通知処理を登録する場合には，付随情報として，変数の番地を指定する．変数の番地がintptr_t型の変数の番地として正しくない場合には，E_PARエラーとなる【NGKI3701】．

保護機能対応カーネルにおいては，周期通知またはアラーム通知の生成時にはそれが属する保護ドメインに対して，タイムウィンドウ通知の登録時にはタイムウィンドウを割り当てるユーザドメインに対して，指定した変数に対する書込みアクセスが許可されていなければならない．そうでない場合には，E_MACVエラーとなる【NGKI3702】．

(3) 変数のインクリメントによる通知

変数のインクリメントによるタイムイベントまたはエラーの通知では，アプリケーションが用意したintptr_t型の変数を，CPUロック状態でインクリメントすることにより通知を行う【NGKI3896】．この通知処理を登録する場合には，付随情報として，変数の番地を指定する．変数の番地がintptr_t型の変数の番地として正しくない場合には，E_PARエラーとなる【NGKI3897】．

保護機能対応カーネルにおいては，周期通知またはアラーム通知の生成時にはそれが属する保護ドメインに対して，タイムウィンドウ通知の登録時にはタイムウィンドウを割り当てるユーザドメインに対して，指定した変数に対する書込みアクセスおよび読出しアクセスが許可されていなければならない．そうでない場合には，E_MACVエラーとなる【NGKI3898】．

【使用上の注意】

変数のインクリメントはCPUロック状態で行うため，カーネル管理外の割込みハンドラや，マルチプロセッサ対応カーネルにおいて他のプロセッサで実行される処理単位から変数を書き換えると，一貫性が保証されなくなる．

(4) タスクの起動／起床による通知

タスクの起動／起床によるタイムイベントまたはエラーの通知では，指定したタスク（対象タスク）を起動／起床することにより通知を行う【NGKI3703】．この通知処理を登録する場合には，付随情報として，対象タスクのタスクIDを指定する．タスクIDが有効範囲外の場合は，E_IDエラーとなる【NGKI3704】．

保護機能対応カーネルにおいては，周期通知またはアラーム通知の生成時にはそれが属する保護ドメインに対して，タイムウィンドウ通知の登録時にはタイムウィンドウを割り当てるユーザドメインに対して，対象タスクに対する通常操作1が許可されていなければならない．そうでない場合には，E_OACVエラーとなる【NGKI3705】．

(5) セマフォの資源の返却による通知

セマフォの資源の返却によるタイムイベントまたはエラーの通知では，指定したセマフォ（対象セマフォ）に資源を返却することにより通知を行う【NGKI3706】．この通知処理を登録する場合には，付随情報として，対象セマフォのセマフォIDを指定する．セマフォIDが有効範囲外の場合は，E_IDエラーとなる【NGKI3707】．

保護機能対応カーネルにおいては，周期通知またはアラーム通知の生成時にはそれが属する保護ドメインに対して，タイムウィンドウ通知の登録時にはタイムウィンドウを割り当てるユーザドメインに対して，対象セマフォに対する通常操作1が許可されていなければならない．そうでない場合には，E_OACVエラーとなる【NGKI3708】．

(6) イベントフラグのセットによる通知

イベントフラグのセットによるタイムイベントまたはエラーの通知では，指定したイベントフラグ（対象イベントフラグ）の指定したビットパターンをセットすることにより通知を行う【NGKI3709】．この通知処理を登録する場合には，付随情報として，対象イベントフラグのイベントフラグIDとセットするビットパターンを指定する．イベントフラグIDが有効範囲外の場合は，E_IDエラーとなる【NGKI3710】．

保護機能対応カーネルにおいては，周期通知またはアラーム通知の生成時にはそれが属する保護ドメインに対して，タイムウィンドウ通知の登録時にはタイムウィンドウを割り当てるユーザドメインに対して，対象イベントフラグに対する通常操作1が許可されていなければならない．そうでない場合には，E_OACVエラーとなる【NGKI3711】．

(7) データキューへの送信による通知

データキューへの送信によるタイムイベントの通知では，指定したデータキュー（対象データキュー）に，指定した値を送信することにより通知を行う【NGKI3712】．この通知処理を登録する場合には，付随情報として，対象データキューのデータキューIDと送信する値を指定する．データキューIDが有効範囲外の場合は，E_IDエラーとなる【NGKI3713】．

また，データキューへの送信によるエラーの通知では，指定したデータキュー（対象データキュー）に，エラーコードを送信することにより通知を行う【NGKI3714】．この通知処理を登録する場合には，付随情報として，対象データキューのデータキューIDを指定する．データキューIDが有効範囲外の場合は，E_IDエラーとなる【NGKI3715】．

保護機能対応カーネルにおいては，周期通知またはアラーム通知の生成時にはそれが属する保護ドメインに対して，タイムウィンドウ通知の登録時にはタイムウィンドウを割り当てるユーザドメインに対して，対象データキューに対する通常操作1が許可されていなければならない．そうでない場合には，E_OACVエラーとなる【NGKI3716】．

(8) 通知方法の指定

タイムイベントとエラーの通知方法は，通知処理モードと，タイムイベントとエラーの通知に関する付随情報で指定する．

通知処理モードは，タイムイベントの通知の種類を表す定数と，エラーの通知の種類を表す定数のビット毎論理和で表す【NGKI3717】．エラーの通知を行わない場合には，タイムイベントの通知の種類を表す定数で表す【NGKI3718】．通知処理モードにそれ以外の値を指定した場合には，E_PARエラーとなる【NGKI3730】．

タイムイベントの通知の種類を表す定数は次の通り【NGKI3719】．

| | | |
|---|---|---|
| TNFY_HANDLER | 0x00U | タイムイベントハンドラの呼出しによる通知 |

| | | |
|---|---|---|
| TNFY_SETVAR | 0x01U | 変数の設定による通知 |
| TNFY_INCVAR | 0x02U | 変数のインクリメントによる通知 |
| TNFY_ACTTSK | 0x03U | タスクの起動による通知 |
| TNFY_WUPTSK | 0x04U | タスクの起床による通知 |
| TNFY_SIGSEM | 0x05U | セマフォの返却による通知 |
| TNFY_SETFLG | 0x06U | イベントフラグのセットによる通知 |
| TNFY_SNDDTQ | 0x07U | データキューへの送信による通知 |

エラーの通知の種類を表す定数は次の通り【NGKI3720】.

| | | |
|---|---|---|
| TENFY_SETVAR | 0x10U | 変数の設定による通知 |
| TENFY_INCVAR | 0x20U | 変数のインクリメントによる通知 |
| TENFY_ACTTSK | 0x30U | タスクの起動による通知 |
| TENFY_WUPTSK | 0x40U | タスクの起床による通知 |
| TENFY_SIGSEM | 0x50U | セマフォの返却による通知 |
| TENFY_SETFLG | 0x60U | イベントフラグのセットによる通知 |
| TENFY_SNDDTQ | 0x70U | データキューへの送信による通知 |

TNFY_HANDLER，TNFY_SETVAR，TNFY_INCVARを指定した場合には，エラーの通知の方法を指定してはならない．指定した場合には，E_PARエラーとなる【NGKI3721】.

周期通知とアラーム通知をサービスコールにより生成する場合には，通知処理モードと付随情報を，次のように定義されるパケット（T_NFYINFO）に入れて渡す【NGKI3722】.

```
typedef struct {
    EXINF       exinf;      /* タイムイベントハンドラの拡張情報 */
    TMEHDR      tmehdr;     /* タイムイベントハンドラの先頭番地 */
} T_NFY_HDR;

typedef struct {
    intptr_t    *p_var;     /* 変数の番地 */
    intptr_t    value;      /* 設定する値 */
} T_NFY_VAR;

typedef struct {
    intptr_t    *p_var;     /* 変数の番地 */
} T_NFY_IVAR;

typedef struct {
    ID          tskid;      /* タスクID */
} T_NFY_TSK;

typedef struct {
    ID          semid;      /* セマフォID */
} T_NFY_SEM;
```

```
typedef struct {
    ID          flgid;      /* イベントフラグID */
    FLGPTN      flgptn;     /* セットするビットパターン */
} T_NFY_FLG;

typedef struct {
    ID          dtqid;      /* データキューID */
    intptr_t    data;       /* 送信する値 */
} T_NFY_DTQ;

typedef struct {
    intptr_t    *p_var;     /* 変数の番地 */
} T_ENFY_VAR;

typedef struct {
    ID          dtqid;      /* データキューID */
} T_ENFY_DTQ;

typedef struct {
    MODE    nfymode;        /* 通知処理モード */
    union {                 /* タイムイベントの通知に関する付随情報 */
        T_NFY_HDR   handler;    /* タイムイベントハンドラの呼出しの場合 */
        T_NFY_VAR   setvar;     /* 変数の設定の場合 */
        T_NFY_IVAR  incvar;     /* 変数のインクリメントの場合 */
        T_NFY_TSK   acttsk;     /* タスクの起動の場合 */
        T_NFY_TSK   wuptsk;     /* タスクの起床の場合 */
        T_NFY_SEM   sigsem;     /* セマフォの資源の返却の場合 */
        T_NFY_FLG   setflg;     /* イベントフラグのセットの場合 */
        T_NFY_DTQ   snddtq;     /* データキューへの送信の場合 */
    } nfy;
    union {                 /* エラーの通知に関する付随情報 */
        T_ENFY_VAR  setvar;     /* 変数の設定の場合 */
        T_NFY_IVAR  incvar;     /* 変数のインクリメントの場合 */
        T_NFY_TSK   acttsk;     /* タスクの起動の場合 */
        T_NFY_TSK   wuptsk;     /* タスクの起床の場合 */
        T_NFY_SEM   sigsem;     /* セマフォの資源の返却の場合 */
        T_NFY_FLG   setflg;     /* イベントフラグのセットの場合 */
        T_ENFY_DTQ  snddtq;     /* データキューへの送信の場合 */
    } enfy;
} T_NFYINFO;
```

周期通知とアラーム通知を静的APIにより生成する場合と，タイムウィンドウを登録する場合には，通知処理モードと必要な付随情報を","で区切って列挙したものを"{"と"}"で囲んだ形で記述する【NGKI3723】．各静的APIのインタフェース記述においては，この"{"と"}"で囲んだ形を，＜通知方法の指定＞と記述する．

【補足説明】

静的APIのインタフェース記述における＜通知方法の指定＞の記述例を示す．

例1）タイムイベントハンドラの呼出しによる通知

```
        { TNFY_HANDLER, 0, tmevt_handler }
```

例2）変数の設定によるタイムイベントの通知

```
        { TNFY_SETVAR, &event_variable, true }
```

例3）タスクの起動によるタイムイベントの通知（エラーの通知はなし）

```
        { TNFY_ACTTSK, TASK_1 }
```

例4）タスクの起床によるタイムイベントの通知とイベントフラグのセットによるエラーの通知

```
        { TNFY_WUPTSK|TENFY_SETFLG, TASK_1, FLG_1, 0x0001U }
```

例5）セマフォの資源の返却によるタイムイベントの通知と変数の設定によるエラーの通知

```
        { TNFY_SIGSEM|TENFY_SETVAR, SEM_1, &error_variable }
```

### 4.6.3 周期通知

周期通知は，指定した周期で通知処理を行うタイムイベント通知である．周期通知は，周期通知IDと呼ぶID番号によって識別する【NGKI3724】．

各周期通知が持つ情報は次の通り【NGKI3725】．

- 周期通知属性
- 周期通知の動作状態
- 次回通知時刻
- 通知方法
- 通知周期
- 通知位相
- 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）

周期通知の通知時刻は，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】．

| | |
|---|---|
| 通知位相＋通知周期×(n−1) | n＝1,2,… |

周期通知の動作状態は，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】．周期通知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．

周期通知が動作している状態の場合には，周期通知の通知時刻になると，周期通知の通知処理が行われる【NGKI3726】．

周期通知属性には，次の属性を指定することができる【NGKI2370】．

| | | |
|---|---|---|
| TA_STA | 0x02U | 周期通知の生成時に周期通知を動作開始する |
| TA_PHS | 0x04U | 周期通知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，周期通知の生成直後には，周期通知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，周期通知を動作開始した時刻が，周期通知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，周期通知を生成した時刻（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】．

次回通知時刻は，周期通知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，周期通知の動作開始時，周期通知の通知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は，周期通知は，割付け可能プロセッサがすべてタイムイベント処理プロセッサであるクラスにのみ属することができる【NGKI5088】．

周期通知機能に関連するカーネル構成マクロは次の通り．

| | |
|---|---|
| TNUM_CYCID | 登録できる周期通知の数（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通知の数に一致）【NGKI2378】 |

【TOPPERS/ASP3カーネルにおける規定】

ASP3カーネルでは，TA_PHS属性の周期通知をサポートしない【ASPS0172】．

【TOPPERS/FMP3カーネルにおける規定】

FMP3カーネルでは，TA_PHS属性の周期通知をサポートしない【FMPS0147】．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルでは，TA_PHS属性の周期通知をサポートしない【HRPS0141】．

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルでは，TA_PHS属性の周期通知をサポートしない【HRMPS0114】．

【TOPPERS新世代カーネル統合仕様との関係】

周期ハンドラ機能を拡張し，周期通知機能とした．

マルチプロセッサ対応カーネルで，周期通知が属することができるクラスの条件を変更した．

【μITRON4.0仕様との関係】

周期ハンドラ機能を拡張し，周期通知機能とした．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** | 周期通知の生成〔S〕【NGKI3727】 |
| **acre_cyc** | 周期通知の生成〔TD〕【NGKI3728】 |

【静的API】

```
CRE_CYC(ID cycid, { ATR cycatr, <通知方法の指定>,
                                    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 | 生成する周期通知の識別名（CRE_CYCの場合） |
| T_CCYC * | pk_ccyc | 周期通知の生成情報を入れたパケットへのポインタ（acre_cycの場合） |

＊周期通知の生成情報（パケットの内容）

| | | |
|---|---|---|
| ATR | cycatr | 周期通知属性 |
| T_NFYINFO | nfyinfo | 周期通知の通知方法 |
| RELTIM | cyctim | 周期通知の通知周期 |
| RELTIM | cycphs | 周期通知の通知位相 |

【リターンパラメータ】

| | | |
|---|---|---|
| ER_ID | cycid | 生成された周期通知の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<br>・非タスクコンテキストからの呼出し〔s〕【NGKI2381】<br>・CPUロック状態からの呼出し〔s〕【NGKI2382】 |
| E_RSATR | 予約属性<br>・cycatrが無効【NGKI2383】<br>・属する保護ドメインの指定が有効範囲外または無所属〔sP〕【NGKI3729】<br>・保護ドメインの囲みの中に記述されていない〔SP〕【NGKI3741】<br>・属するクラスの指定が有効範囲外〔sM〕【NGKI2386】<br>・クラスの囲みの中に記述されていない〔SM〕【NGKI2387】<br>・その他の条件については機能の項を参照 |
| E_ID | 不正ID番号<br>・通知方法中のオブジェクトIDが有効範囲外［NGKI3704］［NGKI3707］［NGKI3710］［NGKI3713］［NGKI3715］ |

| | |
|---|---|
| E_PAR | パラメータエラー<br>・通知方法中の通知処理モードの値が正しくない［NGKI3730］［NGKI3721］<br>・通知方法中のタイムイベントハンドラの先頭番地または変数の番地が正しくない［NGKI3693］［NGKI3699］［NGKI3701］［NGKI3897］<br>・cyctimが有効範囲（0より大きく，TMAX_RELTIM以下）外【NGKI2397】<br>・cycphsが有効範囲（0以上，TMAX_RELTIM以下）外【NGKI2399】<br>・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<br>・属する保護ドメインに対する通常操作1が許可されていない〔sP〕【NGKI3975】<br>・通知方法中のオブジェクトに対する操作が，属する保護ドメインに許可されていない〔P〕［NGKI3705］［NGKI3708］［NGKI3711］［NGKI3716］<br>・属する保護ドメインがユーザドメインで，通知方法にタイムイベントハンドラの呼出しを指定〔P〕［NGKI3695］ |
| E_MACV | メモリアクセス違反<br>・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】<br>・通知方法中の変数に対する操作が，属する保護ドメインに許可されていない〔P〕［NGKI3702］［NGKI3898］ |
| E_NOID | ID番号不足<br>・割り付けられる周期通知IDがない〔sD〕【NGKI2391】 |
| E_OBJ | オブジェクト状態エラー<br>・cycidで指定した識別名のオブジェクトが登録済み〔S〕【NGKI2392】 |

【機能】

各パラメータで指定した周期通知の生成情報に従って，周期通知を生成する【NGKI5197】．周期通知の生成時には，以下の処理を行う．

cycatrにTA_STAを指定した場合，対象周期通知は動作している状態となる【NGKI2393】．次回通知時刻は，サービスコールを呼び出した時刻（静的APIの場合はカーネルの起動時刻）から，cycphsで指定した相対時間後に設定される【NGKI2394】．cycphsにcyctimより大きい値を指定してもよい【NGKI2400】．

cycatrにTA_STAを指定しない場合，対象周期通知は動作していない状態に初期化される【NGKI2395】．

静的APIにおいては，cycidはオブジェクト識別名，cycatr，通知方法中の通知処理モード，cyctim，cycphsは整数定数式パラメータ，通知方法中の付随情報は一般定数式パラメータである【NGKI3731】．

マルチプロセッサ対応カーネルで，生成する周期通知の属するクラスの割付け可能プロセッサが，タイムイベント処理プロセッサでないものを含む場合には，E_RSATRエラーとなる【NGKI5089】．

保護機能とマルチプロセッサの両方に対応するカーネルでは，生成する周期通知がユーザドメインに属する場合に，［NGKI0628］の制約が課される．具体的には，生成する周期通知が属するユーザドメインが，アイドルドメインにまとめられておらず，属するクラスの初期割付けプロセッサでタイムウィンドウを割り当てられていない場合，E_PARエラーとなる【NGKI5130】．

【使用上の注意】

通知方法中のオブジェクトに対する操作が，属する保護ドメインに許可されていないエラーのチェックは，周期通知の生成時に行われる．そのため，周期通知の生成後に操作対象のオブジェクトのアクセス許可ベクタを変更しても，生成済みの周期通知がエラーになることはない． 通知方法中の変数に対する操作が，属する保護ドメインに許可されていないエラーのチェックについても，同様である．

【TOPPERS/ASP3カーネルにおける規定】

ASP3カーネルでは，TA_PHS属性の周期通知はサポートしない［ASPS0172］．ASP3カーネルの動的生成機能拡張パッケージでは，acre_cycをサポートする【ASPS0175】．

【TOPPERS/FMP3カーネルにおける規定】

FMP3カーネルでは，TA_PHS属性の周期通知はサポートしない［FMPS0147］．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルでは，TA_PHS属性の周期通知はサポートしない［HRPS0141］．HRP3カーネルの動的生成機能拡張パッケージでは，acre_cycをサポートする【HRPS0202】．

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルでは，TA_PHS属性の周期通知はサポートしない［HRMPS0114］．

【μITRON4.0仕様との関係】

通知方法が指定できるようにパラメータを変更した．また，cycphsにcyctimより大きい値を指定した場合の振舞いを規定した．

【μITRON4.0/PX仕様との関係】

アクセス権を制御するためのアクセスの種別を，保護ドメインに対する通常操作1に変更した．

【TOPPERS新世代カーネル統合仕様との関係】

通知方法が指定できるようにパラメータを変更した．

アクセス権を制御するためのアクセスの種別を，保護ドメインに対する通常操作1に変更した．

## AID_CYC　割付け可能な周期通知IDの数の指定〔SD〕【NGKI2402】

【静的API】

```
AID_CYC(uint_t nocyc)
```

【パラメータ】

| | | |
|---|---|---|
| uint_t | nocyc | 割付け可能な周期通知IDの数 |

【エラーコード】

| | |
|---|---|
| E_RSATR | 予約属性<br>・保護ドメインの囲みの中に記述されていない〔P〕【NGKI3980】<br>・クラスの囲みの中に記述されていない〔M〕【NGKI2404】<br>・その他の条件については機能の項を参照 |

【機能】

nocycで指定した数の周期通知IDを，周期通知を生成するサービスコールによって割付け可能な周期通知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，AID_CYCが属するクラスの割付け可能プロセッサが，タイムイベント処理プロセッサでないものを含む場合には，E_RSATRエラーとなる【NGKI5090】．

【TOPPERS/ASP3カーネルにおける規定】

ASP3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/HRP3カーネルにおける規定】

HRP3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** | 周期通知のアクセス許可ベクタの設定〔SP〕【NGKI2408】 |
| **sac_cyc** | 周期通知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 | 対象周期通知の識別名（SAC_CYCの場合）またはID番号（sac_cycの場合） |
| 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（sac_cycの場合） |

＊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<br>・非タスクコンテキストからの呼出し〔s〕【NGKI2410】<br>・CPUロック状態からの呼出し〔s〕【NGKI2411】 |
| E_ID | 不正ID番号<br>・cycidが有効範囲外〔s〕【NGKI2412】 |
| E_RSATR | 予約属性<br>・対象周期通知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〕【NGKI3732】<br>・対象周期通知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<br>・対象周期通知が未登録【NGKI2415】 |
| E_OACV | オブジェクトアクセス違反<br>・対象周期通知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<br>・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<br>・対象周期通知は静的APIで生成された〔s〕【NGKI2418】<br>・対象周期通知に対してアクセス許可ベクタが設定済み〔S〕【NGKI2419】 |

【機能】

cycidで指定した周期通知（対象周期通知）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/HRP3カーネルにおける規定】

HRP3カーネルの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　　周期通知の削除〔TD〕【NGKI2422】

【C言語API】

```
ER ercd = del_cyc(ID cycid)
```

【パラメータ】

| | | |
|---|---|---|
| ID | cycid | 対象周期通知の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<br>・非タスクコンテキストからの呼出し【NGKI2423】<br>・CPUロック状態からの呼出し【NGKI2424】 |
| E_ID | 不正ID番号<br>・cycidが有効範囲外【NGKI2425】 |
| E_NOEXS | オブジェクト未登録<br>・対象周期通知が未登録【NGKI2426】 |
| E_OACV | オブジェクトアクセス違反<br>・対象周期通知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<br>・対象周期通知は静的APIで生成された【NGKI2428】 |

【機能】

cycidで指定した周期通知（対象周期通知）を削除する．具体的な振舞いは以下の通り．

対象周期通知の登録が解除され，その周期通知IDが未使用の状態に戻される【NGKI2429】．対象周期通知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/ASP3カーネルにおける規定】

ASP3カーネルの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/HRP3カーネルにおける規定】

HRP3カーネルの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　　周期通知の動作開始〔T〕【NGKI2431】

【C言語API】

```
ER ercd = sta_cyc(ID cycid)
```

【パラメータ】

| | | |
|---|---|---|
| ID | cycid | 対象周期通知の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<br>・非タスクコンテキストからの呼出し【NGKI2432】<br>・CPUロック状態からの呼出し【NGKI2433】 |
| E_ID | 不正ID番号<br>・cycidが有効範囲外【NGKI2434】 |
| E_NOEXS | オブジェクト未登録<br>・対象周期通知が未登録〔D〕【NGKI2435】 |
| E_OACV | オブジェクトアクセス違反<br>・対象周期通知に対する通常操作1が許可されていない〔P〕【NGKI2436】 |

【機能】

cycidで指定した周期通知（対象周期通知）を動作開始する．具体的な振舞いは以下の通り．

対象周期通知が動作していない状態であれば，対象周期通知は動作している状態となる【NGKI2437】．次回通知時刻は，sta_cycを呼び出して以降の最初の通知時刻に設定される【NGKI2438】．

対象周期通知が動作している状態であれば，次回通知時刻の再設定のみが行われる【NGKI2439】．

【補足説明】

TA_PHS属性でない周期通知の場合，次回通知時刻は，sta_cycを呼び出してから，対象周期通知の通知位相で指定した相対時間後に設定される．

対象周期通知がTA_PHS属性で，動作している状態であれば，次回通知時刻は変化しない．

【μITRON4.0仕様との関係】

TA_PHS属性でない周期通知において，sta_cycを呼び出した後，最初に周期通知が通知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，通知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通知の動作開始〔TM〕【NGKI2440】

【C言語API】

```
ER ercd = msta_cyc(ID cycid, ID prcid)
```

【パラメータ】

| | | |
|---|---|---|
| ID | cycid | 対象周期通知のID番号 |
| ID | prcid | 周期通知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<br>・非タスクコンテキストからの呼出し【NGKI2441】<br>・CPUロック状態からの呼出し【NGKI2442】 |
| E_ID | 不正ID番号<br>・cycidが有効範囲外【NGKI2443】<br>・prcidが有効範囲外【NGKI2444】 |
| E_PAR | パラメータエラー<br>・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<br>・対象周期通知が未登録〔D〕【NGKI2445】 |
| E_OACV | オブジェクトアクセス違反<br>・対象周期通知に対する通常操作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で指定した周期通知（対象周期通知）を動作開始する．具体的な振舞いは以下の通り．

対象周期通知が動作していない状態であれば，対象周期通知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，対象周期通知は動作している状態となる【NGKI2447】．次回通知時刻は，msta_cycを呼び出して以降の最初の通知時刻に設定される【NGKI2448】．

対象周期通知が動作している状態であれば，対象周期通知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，次回通知時刻の再設定が行われる【NGKI2449】．

対象周期通知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，実行中の周期通知を実行するプロセッサは変更されない【NGKI2450】．対象周期通知が変更後の割付けプロセッサで実行されるのは，次に通知される時からである【NGKI2451】．

対象周期通知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【NGKI2452】．

保護機能対応カーネルでは，対象周期通知がユーザドメインに属する場合に，［NGKI0628］の制約が課

される．具体的には，対象周期通知が属するユーザドメインが，アイドルドメインにまとめられておらず，prcidで指定したプロセッサでタイムウィンドウを割り当てられていない場合，E_PARエラーとなる【NGKI5131】．

prcidにTPRC_INI（＝0）を指定すると，対象周期通知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【NGKI2453】．

【補足説明】

TA_PHS属性でない周期通知の場合，次回通知時刻は，msta_cycを呼び出してから，対象周期通知の通知位相で指定した相対時間後に設定される．

【使用上の注意】

msta_cycで実行中の周期通知の割付けプロセッサを変更した場合，同じ周期通知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，対象周期通知の通知位相が0の場合に，注意が必要である．

【TOPPERS新世代カーネル統合仕様との関係】

msta_cycがE_NOSPTエラーとなることがなくなった．

【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_cyc　　周期通知の動作停止〔T〕【NGKI2455】

【C言語API】

```
ER ercd = stp_cyc(ID cycid)
```

【パラメータ】

| | | |
|---|---|---|
| ID | cycid | 対象周期通知の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<br>・非タスクコンテキストからの呼出し【NGKI2456】<br>・CPUロック状態からの呼出し【NGKI2457】 |
| E_ID | 不正ID番号<br>・cycidが有効範囲外【NGKI2458】 |
| E_NOEXS | オブジェクト未登録<br>・対象周期通知が未登録〔D〕【NGKI2459】 |
| E_OACV | オブジェクトアクセス違反<br>・対象周期通知に対する通常操作2が許可されていない〔P〕【NGKI2460】 |

【機能】

cycidで指定した周期通知（対象周期通知）を動作停止する．具体的な振舞いは以下の通り．

対象周期通知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】．対象周期通知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　　周期通知の状態参照〔T〕【NGKI2463】

【C言語API】

```
ER ercd = ref_cyc(ID cycid, T_RCYC *pk_rcyc)
```

【パラメータ】

| | | |
|---|---|---|
| ID | cycid | 対象周期通知のID番号 |
| T_RCYC * | pk_rcyc | 周期通知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 |

＊周期通知の現在状態（パケットの内容）

| | | |
|---|---|---|
| STAT | cycstat | 周期通知の動作状態 |
| RELTIM | lefttim | 次回通知時刻までの相対時間 |
| ID | prcid | 周期通知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<br>・非タスクコンテキストからの呼出し【NGKI2464】<br>・CPUロック状態からの呼出し【NGKI2465】 |
| E_ID | 不正ID番号<br>・cycidが有効範囲外【NGKI2466】 |
| E_NOEXS | オブジェクト未登録<br>・対象周期通知が未登録〔D〕【NGKI2467】 |
| E_OACV | オブジェクトアクセス違反<br>・対象周期通知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<br>・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で指定した周期通知（対象周期通知）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，対象周期通知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 | 周期通知が動作していない状態 |
| TCYC_STA | 0x02U | 周期通知が動作している状態 |

対象周期通知が動作している状態である場合には，lefttimに，次回通知時刻までの相対時間が返される

【NGKI2472】．対象周期通知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，対象周期通知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，対象周期通知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通知の状態が変化している可能性があるためである．また，マルチプロセッサ対応カーネルでは，対象周期通知の現在状態を参照した直後に，他のプロセッサで実行されている処理により対象周期通知の状態が変化する場合もある．

【μITRON4.0仕様との関係】

TCYC_STPとTCYC_STAを値を変更した．

### 4.6.4 アラーム通知

アラーム通知は，指定した相対時間後に通知処理を行うタイムイベント通知である．アラーム通知は，アラーム通知IDと呼ぶID番号によって識別する【NGKI3733】．

各アラーム通知が持つ情報は次の通り【NGKI3734】．

- アラーム通知属性
- アラーム通知の動作状態
- 通知時刻
- 通知方法
- 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）

アラーム通知の動作状態は，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】．アラーム通知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．

通知時刻は，アラーム通知が動作している状態でのみ有効で，アラーム通知を動作開始する時に設定される【NGKI2478】．

アラーム通知が動作している状態の場合には，通知時刻になると，アラーム通知が動作していない状態にされた後，アラーム通知の通知処理が行われる【NGKI3735】．

マルチプロセッサ対応カーネルでは，アラーム通知は，割付け可能プロセッサがすべてタイムイベント処理プロセッサであるクラスにのみ属することができる【NGKI5091】．

アラーム通知属性に指定できる属性はない【NGKI3423】．そのためアラーム通知属性には，TA_NULLを指定しなければならない【NGKI3424】．

アラーム通知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通知の数（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通知の数に一致）【NGKI2486】 |

【TOPPERS新世代カーネル統合仕様との関係】

アラームハンドラ機能を拡張し，アラーム通知機能とした．

マルチプロセッサ対応カーネルで，アラーム通知が属することができるクラスの条件を変更した．

【μITRON4.0仕様との関係】

アラームハンドラ機能を拡張し，アラーム通知機能とした．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** | アラーム通知の生成〔S〕【NGKI3736】 |
| **acre_alm** | アラーム通知の生成〔TD〕【NGKI3737】 |

【静的API】

```
CRE_ALM(ID almid, { ATR almatr, <通知方法の指定> })
```

【C言語API】

```
ER_ID almid = acre_alm(const T_CALM *pk_calm)
```

【パラメータ】

| | | |
|---|---|---|
| ID | almid | 生成するアラーム通知の識別名（CRE_ALMの場合） |
| T_CALM * | pk_calm | アラーム通知の生成情報を入れたパケットへのポインタ（acre_almの場合） |

＊アラーム通知の生成情報（パケットの内容）

| | | |
|---|---|---|
| ATR | almatr | アラーム通知属性 |
| T_NFYINFO | nfyinfo | アラーム通知の通知方法 |

【リターンパラメータ】

| | | |
|---|---|---|
| ER_ID | almid | 生成されたアラーム通知の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<br>・非タスクコンテキストからの呼出し〔s〕【NGKI2489】<br>・CPUロック状態からの呼出し〔s〕【NGKI2490】 |
| E_RSATR | 予約属性<br>・almatrが無効【NGKI2491】<br>・属する保護ドメインの指定が有効範囲外または無所属〔sP〕【NGKI3738】<br>・保護ドメインの囲みの中に記述されていない〔SP〕【NGKI3742】<br>・属するクラスの指定が有効範囲外〔sM〕【NGKI2494】<br>・クラスの囲みの中に記述されていない〔SM〕【NGKI2495】<br>・その他の条件については機能の項を参照 |
| E_ID | 不正ID番号<br>・通知方法中のオブジェクトIDが有効範囲外［NGKI3704］［NGKI3707］［NGKI3710］［NGKI3713］［NGKI3715］ |
| E_PAR | パラメータエラー<br>・通知方法中の通知処理モードの値が正しくない［NGKI3730］［NGKI3721］<br>・通知方法中のタイムイベントハンドラの先頭番地または変数の番地が正しくない［NGKI3693］［NGKI3699］［NGKI3701］［NGKI3897］<br>・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<br>・属する保護ドメインに対する通常操作1が許可されていない〔sP〕【NGKI3976】<br>・通知方法中のオブジェクトに対する操作が，属する保護ドメインに許可されていない〔P〕［NGKI3705］［NGKI3708］［NGKI3711］［NGKI3716］<br>・属する保護ドメインがユーザドメインで，通知方法にタイムイベントハンドラの呼出しを指定〔P〕［NGKI3695］ |
| E_MACV | メモリアクセス違反<br>・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】<br>・通知方法中の変数に対する操作が，属する保護ドメインに許可されていない〔P〕［NGKI3702］［NGKI3898］ |
| E_NOID | ID番号不足<br>・割り付けられるアラーム通知IDがない〔sD〕【NGKI2499】 |
| E_OBJ | オブジェクト状態エラー<br>・almidで指定した識別名のオブジェクトが登録済み〔S〕【NGKI2500】 |

【機能】

各パラメータで指定したアラーム通知の生成情報に従って，アラーム通知を生成する【NGKI5198】．対象アラーム通知は，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と通知方法中の通知処理モードは整数定数式パラメータ，通知方法中の付随情報は一般定数式パラメータである【NGKI3739】．

マルチプロセッサ対応カーネルで，生成するアラーム通知の属するクラスの割付け可能プロセッサが，タイムイベント処理プロセッサでないものを含む場合には，E_RSATRエラーとなる【NGKI5092】．

保護機能とマルチプロセッサの両方に対応するカーネルでは，生成するアラーム通知がユーザドメインに属する場合に，［NGKI0628］の制約が課される．具体的には，生成するアラーム通知が属するユーザドメインが，アイドルドメインにまとめられておらず，属するクラスの初期割付けプロセッサでタイムウィンドウを割り当てられていない場合，E_PARエラーとなる【NGKI5132】．

【使用上の注意】

通知方法中のオブジェクトに対する操作が，属する保護ドメインに許可されていないエラーのチェックは，アラーム通知の生成時に行われる．そのため，アラーム通知の生成後に操作対象のオブジェクトのアクセス許可ベクタを変更しても，生成済みのアラーム通知がエラーになることはない． 通知方法中の変数に対する操作が，属する保護ドメインに許可されていないエラーのチェックについても，同様である．

【TOPPERS/ASP3カーネルにおける規定】

ASP3カーネルの動的生成機能拡張パッケージでは，acre_almをサポートする【ASPS0180】．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルの動的生成機能拡張パッケージでは，acre_almをサポートする【HRPS0205】．

【μITRON4.0仕様，TOPPERS新世代カーネル統合仕様との関係】

通知方法が指定できるようにパラメータを変更した．

【μITRON4.0/PX仕様，TOPPERS新世代カーネル統合仕様との関係】

アクセス権を制御するためのアクセスの種別を，保護ドメインに対する通常操作1に変更した．

## AID_ALM　割付け可能なアラーム通知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通知IDの数 |

【エラーコード】

| | |
|---|---|
| E_RSATR | 予約属性<br>・保護ドメインの囲みの中に記述されていない〔P〕【NGKI3981】<br>・クラスの囲みの中に記述されていない〔M〕【NGKI2506】<br>・その他の条件については機能の項を参照 |

【機能】

noalmで指定した数のアラーム通知IDを，アラーム通知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通知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，AID_ALMが属するクラスの割付け可能プロセッサが，タイムイベント処理プロセッサでないものを含む場合には，E_RSATRエラーとなる【NGKI5093】．

【TOPPERS/ASP3カーネルにおける規定】

ASP3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/HRP3カーネルにおける規定】

HRP3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** | アラーム通知のアクセス許可ベクタの設定〔SP〕【NGKI2510】 |
| **sac_alm** | アラーム通知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 | 対象アラーム通知の識別名（SAC_ALMの場合）またはID番号（sac_almの場合） |
| 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（sac_almの場合） |

＊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<br>・非タスクコンテキストからの呼出し〔s〕【NGKI2512】<br>・CPUロック状態からの呼出し〔s〕【NGKI2513】 |
| E_ID | 不正ID番号<br>・almidが有効範囲外〔s〕【NGKI2514】 |
| E_RSATR | 予約属性<br>・対象アラーム通知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〕【NGKI3740】<br>・対象アラーム通知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<br>・対象アラーム通知が未登録【NGKI2517】 |
| E_OACV | オブジェクトアクセス違反<br>・対象アラーム通知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<br>・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<br>・対象アラーム通知は静的APIで生成された〔s〕【NGKI2520】<br>・対象アラーム通知に対してアクセス許可ベクタが設定済み〔S〕【NGKI2521】 |

【機能】

almidで指定したアラーム通知（対象アラーム通知）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【NGKI2522】.

静的APIにおいては，almid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【NGKI2523】.

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルの動的生成機能拡張パッケージでは，sac_almをサポートする【HRPS0206】.

## del_alm　　アラーム通知の削除〔TD〕【NGKI2524】

【C言語API】

```
ER ercd = del_alm(ID almid)
```

【パラメータ】

| | | |
|---|---|---|
| ID | almid | 対象アラーム通知の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<br>・非タスクコンテキストからの呼出し【NGKI2525】<br>・CPUロック状態からの呼出し【NGKI2526】 |
| E_ID | 不正ID番号<br>・almidが有効範囲外【NGKI2527】 |
| E_NOEXS | オブジェクト未登録<br>・対象アラーム通知が未登録【NGKI2528】 |
| E_OACV | オブジェクトアクセス違反<br>・対象アラーム通知に対する管理操作が許可されていない〔P〕【NGKI2529】 |
| E_OBJ | オブジェクト状態エラー<br>・対象アラーム通知は静的APIで生成された【NGKI2530】 |

【機能】

almidで指定したアラーム通知（対象アラーム通知）を削除する．具体的な振舞いは以下の通り．

対象アラーム通知の登録が解除され，そのアラーム通知IDが未使用の状態に戻される【NGKI2531】．対象アラーム通知が動作している状態であった場合には，登録解除の前に，アラーム通知が動作していない状態となる【NGKI2532】．

【TOPPERS/ASP3カーネルにおける規定】

ASP3カーネルの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/HRP3カーネルにおける規定】

HRP3カーネルの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　　アラーム通知の動作開始〔TI〕【NGKI3543】

【C言語API】

```
ER ercd = sta_alm(ID almid, RELTIM almtim)
```

【パラメータ】

| | | |
|---|---|---|
| ID | almid | 対象アラーム通知のID番号 |
| RELTIM | almtim | アラーム通知の通知時刻（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<br>・CPUロック状態からの呼出し【NGKI3899】 |
| E_ID | 不正ID番号<br>・almidが有効範囲外【NGKI2537】 |
| E_PAR | パラメータエラー<br>・almtimがTMAX_RELTIMより大きい【NGKI2538】 |
| E_NOEXS | オブジェクト未登録<br>・対象アラーム通知が未登録〔D〕【NGKI2539】 |
| E_OACV | オブジェクトアクセス違反<br>・対象アラーム通知に対する通常操作1が許可されていない〔P〕【NGKI2540】 |

【機能】

almidで指定したアラーム通知（対象アラーム通知）を動作開始する．具体的な振舞いは以下の通り．

対象アラーム通知が動作していない状態であれば，対象アラーム通知は動作している状態となる【NGKI2541】．通知時刻は，sta_almを呼び出してから，almtimで指定した相対時間後に設定される【NGKI2542】．

対象アラーム通知が動作している状態であれば，通知時刻の再設定のみが行われる【NGKI2543】．

## msta_alm　割付けプロセッサ指定でのアラーム通知の動作開始〔TIM〕【NGKI3544】

【C言語API】

```
ER ercd = msta_alm(ID almid, RELTIM almtim, ID prcid)
```

【パラメータ】

| | | |
|---|---|---|
| ID | almid | 対象アラーム通知のID番号 |
| RELTIM | almtim | アラーム通知の通知時刻（相対時間） |
| ID | prcid | アラーム通知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<br>・CPUロック状態からの呼出し【NGKI2548】 |
| E_ID | 不正ID番号<br>・almidが有効範囲外【NGKI2549】<br>・prcidが有効範囲外【NGKI2550】 |
| E_PAR | パラメータエラー<br>・almtimがTMAX_RELTIMより大きい【NGKI2551】<br>・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<br>・対象アラーム通知が未登録〔D〕【NGKI2552】 |
| E_OACV | オブジェクトアクセス違反<br>・対象アラーム通知に対する通常操作1が許可されていない〔P〕【NGKI2553】 |

【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通知（対象アラーム通知）を動作開始する．具体的な振舞いは以下の通り．

対象アラーム通知が動作していない状態であれば，対象アラーム通知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，対象アラーム通知は動作している状態となる【NGKI2554】．通知時刻は，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】．

対象アラーム通知が動作している状態であれば，対象アラーム通知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，通知時刻の再設定が行われる【NGKI2556】．

対象アラーム通知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，実行中のアラーム通知を実行するプロセッサは変更されない【NGKI2557】．対象アラーム通知が変更後の割付けプロセッサで実行されるのは，次に通知される時からである【NGKI2558】．

対象アラーム通知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，対象アラーム通知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】．

保護機能対応カーネルでは，対象アラーム通知がユーザドメインに属する場合に，［NGKI0628］の制約が課される．具体的には，対象アラーム通知が属するユーザドメインが，アイドルドメインにまとめられておらず，prcidで指定したプロセッサでタイムウィンドウを割り当てられていない場合，E_PARエラーとなる【NGKI5133】．

【使用上の注意】

msta_almで実行中のアラーム通知の割付けプロセッサを変更した場合，同じアラーム通知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．

【TOPPERS新世代カーネル統合仕様との関係】

msta_almがE_NOSPTエラーとなることがなくなった．

【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　　アラーム通知の動作停止〔TI〕【NGKI3545】

【C言語API】

```
ER ercd = stp_alm(ID almid)
```

【パラメータ】

| | | |
|---|---|---|
| ID | almid | 対象アラーム通知の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<br>・CPUロック状態からの呼出し【NGKI2566】 |
| E_ID | 不正ID番号<br>・almidが有効範囲外【NGKI2567】 |
| E_NOEXS | オブジェクト未登録<br>・対象アラーム通知が未登録〔D〕【NGKI2568】 |
| E_OACV | オブジェクトアクセス違反<br>・対象アラーム通知に対する通常操作2が許可されていない〔P〕【NGKI2569】 |

【機能】

almidで指定したアラーム通知（対象アラーム通知）を動作停止する．具体的な振舞いは以下の通り．

対象アラーム通知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】．対象アラーム通知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　　アラーム通知の状態参照〔T〕【NGKI2572】

【C言語API】

```
ER ercd = ref_alm(ID almid, T_RALM *pk_ralm)
```

【パラメータ】

| | | |
|---|---|---|
| ID | almid | 対象アラーム通知のID番号 |
| T_RALM * | pk_ralm | アラーム通知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 |

＊アラーム通知の現在状態（パケットの内容）

| | | |
|---|---|---|
| STAT | almstat | アラーム通知の動作状態 |
| RELTIM | lefttim | 通知時刻までの相対時間 |
| ID | prcid | アラーム通知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<br>・非タスクコンテキストからの呼出し【NGKI2573】<br>・CPUロック状態からの呼出し【NGKI2574】 |
| E_ID | 不正ID番号<br>・almidが有効範囲外【NGKI2575】 |
| E_NOEXS | オブジェクト未登録<br>・対象アラーム通知が未登録〔D〕【NGKI2576】 |
| E_OACV | オブジェクトアクセス違反<br>・対象アラーム通知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<br>・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通知（対象アラーム通知）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，対象アラーム通知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 | アラーム通知が動作していない状態 |
| TALM_STA | 0x02U | アラーム通知が動作している状態 |

対象アラーム通知が動作している状態である場合には，lefttimに，通知時刻までの相対時間が返される

【NGKI2581】．対象アラーム通知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，対象アラーム通知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，対象アラーム通知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通知の状態が変化している可能性があるためである．また，マルチプロセッサ対応カーネルでは，対象アラーム通知の現在状態を参照した直後に，他のプロセッサで実行されている処理により対象アラーム通知の状態が変化する場合もある．

【μITRON4.0仕様との関係】

TALM_STPとTALM_STAを値を変更した．

### 4.6.5 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の呼出し処理は，オーバランタイマ割込みによって実行される．そのため，オーバランタイマ割込みのマスクや，オーバランタイマ割込みより優先される処理によって，オーバランタイマ割込みの処理が遅れると，オーバランハンドラの呼出し処理は遅れることになる．

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．

オーバランハンドラの動作状態は，タスクの生成時と，タスクが休止状態に遷移する時に，動作していない状態に初期化される【NGKI2587】．

残りプロセッサ時間は，オーバランハンドラの動作開始時に，指定した値に初期化される【NGKI3771】．この時，PRCTIM型に格納できる任意の値を指定できるとは限らず，指定できる値にターゲット定義の上限がある場合がある【NGKI2594】．残りプロセッサ時間に指定できる最大値は，カーネル構成マクロTMAX_OVRTIMに定義されている【NGKI2595】．

オーバランハンドラが動作している状態でタスクが実行している間，残りプロセッサ時間は，タスクが使用したプロセッサ時間の分だけ減少する【NGKI2588】．残りプロセッサ時間が0になると（これをオーバランと呼ぶ），オーバランハンドラが起動される【NGKI2589】．この時，オーバランしたタスクのオーバランハンドラの動作状態は，動作していない状態になる【NGKI3993】．

タスクが使用したプロセッサ時間には，そのタスク自身とそこから呼び出したサービルコール（拡張サービスコールを含む）の実行時間を含む【NGKI2590】．一方，タスクの実行中に起動されたカーネル管理の割込みハンドラ（割込みサービスルーチン，周期通知，アラーム通知，オーバランハンドラの実行時間を含む）とカーネル管理のCPU例外ハンドラの実行時間は含まないが，割込みハンドラおよびCPU例外ハンドラの呼出し／復帰にかかる時間と，それらの入口処理と出口処理の一部の実行時間は含んでしまう【NGKI2591】．また，タスクの実行中に起動されたカーネル管理外の割込みハンドラとカーネル管理外のCPU例外ハンドラの実行時間も含む【NGKI2592】．

保護機能対応カーネルにおいて，オーバランハンドラは，カーネルドメインに属する【NGKI2597】．

ターゲット定義で，オーバランハンドラ機能がサポートされていない場合がある【NGKI2598】．オーバランハンドラ機能がサポートされている場合には，TOPPERS_SUPPORT_OVRHDRがマクロ定義される【NGKI2599】．サポートされていない場合にオーバランハンドラ機能のサービスコールを呼び出すと，E_NOSPTエラーが返るか，リンク時にエラーとなる【NGKI2600】．

オーバランハンドラ属性に指定できる属性はない【NGKI2602】．そのためオーバランハンドラ属性には，TA_NULLを指定しなければならない【NGKI2603】．

C言語によるオーバランハンドラの記述形式は次の通り【NGKI2604】．

```
void overrun_handler(ID tskid, EXINF exinf)
{
    オーバランハンドラ本体
}
```

tskidにはオーバランを起こしたタスクのID番号が，exinfにはそのタスクの拡張情報が，それぞれ渡される【NGKI2605】．

オーバラン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．

| | |
|---|---|
| TMAX_OVRTIM | オーバランハンドラの残りプロセッサ時間に指定できる最大値［NGKI2595］ |
| TOPPERS_SUPPORT_OVRHDR | オーバランハンドラ機能がサポートされている［NGKI2599］ |

【使用上の注意】

マルチプロセッサ対応カーネルでは，オーバラン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ので，注意が必要である．

【TOPPERS/ASP3カーネルにおける規定】

ASP3カーネルでは，オーバランハンドラをサポートしない［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/FMP3カーネルにおける規定】

FMP3カーネルでは，オーバランハンドラをサポートしない［FMPS0002］．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルでは，オーバランハンドラをサポートしない［HRPS0013］．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［HRPS0014］．

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルでは，オーバランハンドラをサポートしない［HRMPS0003］．

【μITRON4.0仕様との関係】

残りプロセッサ時間の単位は，μITRON4.0仕様では実装定義としていたが，この仕様ではマイクロ秒と規定した．

TMAX_OVRTIMは，μITRON4.0仕様に規定されていないカーネル構成マクロである．

【TOPPERS新世代カーネル統合仕様との関係】

残りプロセッサ時間のデータ型を，OVRIM型からPRCTIM型に変更した．

## DEF_OVR　オーバランハンドラの定義〔S〕【NGKI2608】

【静的API】

```
DEF_OVR({ ATR ovratr, OVRHDR ovrhdr })
```

【パラメータ】

＊オーバランハンドラの定義情報

| | | |
|---|---|---|
| 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 |

【エラーコード】

| | |
|---|---|
| E_RSATR | 予約属性<br>・ovratrが無効【NGKI2612】<br>・カーネルドメインの囲みの中に記述されていない〔P〕【NGKI2621】<br>・クラスの囲みの中に記述されている〔M〕【NGKI2625】 |
| E_PAR | パラメータエラー<br>・ovrhdrがプログラムの先頭番地として正しくない【NGKI2613】 |
| E_OBJ | オブジェクト状態エラー<br>・オーバランハンドラが定義済み【NGKI2619】 |

【機能】

各パラメータで指定したオーバランハンドラの定義情報に従って，オーバランハンドラを定義する【NGKI2616】.

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】.

【μITRON4.0仕様との関係】

ovrhdrのデータ型をOVRHDRに変更した．オーバランハンドラを定義するサービスコール（def_ovr）を廃止した.

【TOPPERS新世代カーネル統合仕様との関係】

オーバランハンドラを定義するサービスコール（def_ovr）を廃止した.

## sta_ovr　　オーバランハンドラの動作開始〔TI〕【NGKI3546】

【C言語API】

```
ER ercd = sta_ovr(ID tskid, PRCTIM ovrtim)
```

【パラメータ】

| | | |
|---|---|---|
| ID | tskid | 対象タスクのID番号 |
| PRCTIM | 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<br>・CPUロック状態からの呼出し【NGKI2634】 |
| E_ID | 不正ID番号<br>・tskidが有効範囲外【NGKI2635】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・対象タスクに対する通常操作2が許可されていない〔P〕【NGKI2637】 |
| E_PAR | パラメータエラー<br>・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<br>・オーバランハンドラが定義されていない【NGKI2638】<br>・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，オーバランハンドラの動作を開始する．具体的な振舞いは以下の通り．

対象タスクに対するオーバランハンドラの動作状態は，動作している状態となり，残りプロセッサ時間は，ovrtimに指定した時間に設定される【NGKI2639】．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であれば，残りプロセッサ時間の設定のみが行われる【NGKI2640】．

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，オーバランハンドラの動作を開始できるかは，ターゲット定義である【NGKI5145】．ターゲット定義で開始できない場合には，E_OBJエラーとなる【NGKI5146】．

タスクコンテキストから呼び出した場合，tskidにTSK_SELF（＝0）を指定すると，自タスクが対象タスクとなる【NGKI2641】．

【μITRON4.0仕様との関係】

ovrtimのデータ型をPRCTIMに変更した．

【TOPPERS新世代カーネル統合仕様との関係】

ovrtimのデータ型をPRCTIMに変更した．

マルチプロセッサ対応カーネルにおいて，ターゲット定義で，他プロセッサに割り付けられたタスクのオーバランハンドラの動作を開始できない場合があることを明記した．

## stp_ovr　　オーバランハンドラの動作停止〔TI〕【NGKI3547】

【C言語API】

```
ER ercd = stp_ovr(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<br>・CPUロック状態からの呼出し【NGKI2648】 |
| E_ID | 不正ID番号<br>・tskidが有効範囲外【NGKI2649】 |
| E_NOEXS | オブジェクト未登録<br>・対象タスクが未登録〔D〕【NGKI2650】 |
| E_OACV | オブジェクトアクセス違反<br>・対象タスクに対する通常操作2が許可されていない〔P〕【NGKI2651】 |
| E_OBJ | オブジェクト状態エラー<br>・オーバランハンドラが定義されていない【NGKI2652】<br>・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，オーバランハンドラの動作を停止する．具体的な振舞いは以下の通り．

対象タスクに対するオーバランハンドラの動作状態は，動作していない状態となる【NGKI2653】．対象タスクに対してオーバラン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【NGKI2654】．

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，オーバランハンドラの動作を停止できるかは，ターゲット定義である【NGKI5147】．ターゲット定義で停止できない場合には，E_OBJエラーとなる【NGKI5148】．

タスクコンテキストから呼び出した場合，tskidにTSK_SELF（＝0）を指定すると，自タスクが対象タスクとなる【NGKI2655】．

【TOPPERS新世代カーネル統合仕様との関係】

マルチプロセッサ対応カーネルにおいて，ターゲット定義で，他プロセッサに割り付けられたタスクのオーバランハンドラの動作を停止できない場合があることを明記した．

## ref_ovr　　オーバランハンドラの状態参照〔T〕【NGKI2656】

【C言語API】

```
ER ercd = ref_ovr(ID tskid, T_ROVR *pk_rovr)
```

【パラメータ】

| | | |
|---|---|---|
| ID | tskid | 対象タスクのID番号 |
| T_ROVR * | pk_rovr | オーバラン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 |

＊タスクの現在状態（パケットの内容）

| | | |
|---|---|---|
| STAT | ovrstat | オーバランハンドラの動作状態 |
| PRCTIM | leftotm | 残りプロセッサ時間 |

【エラーコード】

| | |
|---|---|
| E_CTX | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI2657】<br>・CPUロック状態からの呼出し【NGKI2658】 |
| E_ID | 不正ID番号<br>・tskidが有効範囲外【NGKI2659】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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<br>・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<br>・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】．

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を参照できるかは，ターゲット定義である【NGKI5139】．ターゲット定義で参照できない場合には，leftotmの値は保証されない【NGKI5140】．

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，対象タスクに対する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タスクに対する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．また，マルチプロセッサ対応カーネルでは，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した直後に，他のプロセッサで実行されている処理により対象タスクに対するオーバランハンドラの状態が変化する場合もある．

【μITRON4.0仕様との関係】

TOVR_STPとTOVR_STAを値を変更した．

【TOPPERS新世代カーネル統合仕様との関係】

マルチプロセッサ対応カーネルにおいて，ターゲット定義で，他プロセッサに割り付けられたタスクのオーバランハンドラの残りプロセッサ時間を参照できない場合があることを明記した．

## 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 },
        { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
```

※ 2つめのアクセス許可ベクタ（システム状態2に対するアクセス許可ベクタ）の記述は省略することができる【NGKI5216】.

【C言語API】

```
ER ercd = sac_sys(const ACVCT *p_acvct1, const ACVCT *p_acvct2)
```

【パラメータ】

| | | |
|---|---|---|
| ACVCT * | p_acvct1 | システム状態1に対するアクセス許可ベクタを入れたパケットへのポインタ（sac_sysの場合） |
| ACVCT * | p_acvct2 | システム状態2に対するアクセス許可ベクタを入れたパケットへのポインタ（sac_sysの場合） |

＊システム状態1に対する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 |

＊システム状態2に対する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<br>・非タスクコンテキストからの呼出し〔s〕【NGKI2672】<br>・CPUロック状態からの呼出し〔s〕【NGKI2673】 |

| | |
|---|---|
| E_RSATR | 予約属性<br>・カーネルドメインの囲みの中に記述されていない〔S〕【NGKI2674】 |
| E_OACV | オブジェクトアクセス違反<br>・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<br>・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1とシステム状態2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，第1パラメータと第2パラメータで指定した値にそれぞれ設定する【NGKI2678】.

静的APIにおいては，いずれのacptn1～acptn4も整数定数式パラメータである【NGKI2679】.

静的APIにおいてシステム状態2に対するアクセス許可ベクタの記述を省略した場合，システム状態2に対するアクセス許可ベクタにはデフォルトの値が設定される【NGKI5217】.

マルチプロセッサ対応カーネルにおいて，SAC_SYSをクラスの囲みの中に記述しても，クラスの情報は使用されない．SAC_SYSをクラスの囲みの中に記述した場合，コンフィギュレータは警告メッセージを出力する【NGKI2675】.

【μITRON4.0/PX仕様，TOPPERS新世代カーネル統合仕様との関係】

システム状態に対して，2つのアクセス許可ベクタを持つこととした.

## rot_rdq　　タスクの優先順位の回転〔TI〕【NGKI3548】

【C言語API】

```
ER ercd = rot_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<br>・CPUロック状態からの呼出し【NGKI2684】 |
| E_NOSPT | 未サポート機能<br>・条件については機能の項を参照 |
| E_PAR | パラメータエラー<br>・tskpriが有効範囲外【NGKI2685】 |
| E_OACV | オブジェクトアクセス違反<br>・呼び出した処理単位が属する保護ドメインに対する通常操作2が許可されていない〔P〕【NGKI3766】 |
| E_ILUSE | サービスコール不正使用<br>・条件については機能の項を参照 |

【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，優先順位が最も高いタスクを，同じ優先度のタスクの中で優先順位を最も低くする【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，何も行わずに正常終了する【NGKI2688】．

保護機能とマルチプロセッサのいずれかまたは両方に対応するカーネルでは，タスクコンテキストから呼び出した場合には，自タスクと同じスケジューリング単位に属するタスクのみを操作対象とする【NGKI5125】．非タスクコンテキストから呼び出した場合には，保護機能に対応するカーネルではカーネルドメインに，マルチプロセッサに対応するカーネルで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対応するスケジューリング単位に属するタスクのみを操作対象とする【NGKI5208】．

タスクコンテキストから呼び出した場合，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】．

サブ優先度機能をサポートするカーネルで，対象優先度がサブ優先度を使用すると設定されている場合には，E_ILUSEエラーとなる【NGKI3663】．

【TOPPERS/SSP3カーネルにおける規定】

SSP3カーネルでは，rot_rdqをサポートしない【SSPS0131】.

【μITRON4.0/PX仕様，TOPPERS新世代カーネル統合仕様との関係】

アクセス権を制御するためのアクセスの種別を，保護ドメインに対する通常操作2に変更した.

## mrot_rdq　対象指定でのタスクの優先順位の回転〔TIP｜TIM〕【NGKI3549】

【C言語API】

```
ER ercd = mrot_rdq(ID schedno, PRI tskpri)
```

【パラメータ】

| | | |
|---|---|---|
| ID | schedno | 優先順位の回転対象とするスケジューリング単位番号 |
| 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<br>・CPUロック状態からの呼出し【NGKI2695】 |
| E_NOSPT | 未サポート機能<br>・条件については機能の項を参照 |
| E_ID | 不正ID番号<br>・schednoが有効範囲外【NGKI2696】 |
| E_PAR | パラメータエラー<br>・tskpriが有効範囲外【NGKI2697】 |
| E_OACV | オブジェクトアクセス違反<br>・schednoで指定した保護ドメインに対する通常操作2が許可されていない〔P〕【NGKI3778】 |
| E_ILUSE | サービスコール不正使用<br>・条件については機能の項を参照 |

【機能】

schednoで指定したスケジューリング単位に属し，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，優先順位が最も高いタスクを，同じ優先度のタスクの中で優先順位を最も低くする【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】．

タスクコンテキストから呼び出した場合，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】．

schednoで指定したスケジューリング単位に属し，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】．

サブ優先度機能をサポートするカーネルで，対象優先度がサブ優先度を使用すると設定されている場合には，E_ILUSEエラーとなる【NGKI3664】．

【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．

【TOPPERS新世代カーネル統合仕様との関係】

mrot_rdqは，TOPPERS新世代カーネル統合仕様ではマルチプロセッサ対応カーネルのみでサポートされるサービスコールであったが，この仕様では保護機能対応カーネルでもサポートされるものとした．それに伴い，優先度の回転対象を指定するパラメータを，プロセッサID（prcid）からスケジューリング単位番号（schedno）に変更した．また，パラメータの順序を変更した．

アクセス権を制御するためのアクセスの種別を，保護ドメインに対する通常操作2に変更した．

## get_tid　　実行状態のタスクIDの参照〔TI〕【NGKI3550】

【C言語API】

```
ER ercd = get_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<br>・CPUロック状態からの呼出し【NGKI2707】 |
| E_MACV | メモリアクセス違反<br>・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〔P〕【NGKI2708】 |

【機能】

実行状態のタスク（タスクコンテキストから呼び出した場合には自タスク）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】．

get_tid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，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】．

| get_did | 実行状態のタスクが属する保護ドメインIDの参照〔TIP〕【NGKI3553】 |
|---|---|

【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<br>・CPUロック状態からの呼出し【NGKI2714】 |
| E_MACV | メモリアクセス違反<br>・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（タスクコンテキストから呼び出した場合には自タスク）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】．

get_did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，TDOM_NONE（＝-2）が返される【NGKI3554】．

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新世代カーネル統合仕様との関係】

get_didは，非タスクコンテキストからも呼び出せるものとした．

## get_pid　　割付けプロセッサのID番号の参照〔TIM〕【NGKI3551】

【C言語API】

```
ER ercd = get_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<br>・CPUロック状態からの呼出し【NGKI2722】 |
| E_MACV | メモリアクセス違反<br>・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〔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．

【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_lod　　実行できるタスクの数の参照〔T〕【NGKI3623】

【C言語API】

```
ER ercd = get_lod(PRI tskpri, uint_t *p_load)
```

【パラメータ】

| | | |
|---|---|---|
| PRI | tskpri | 参照対象の優先度（対象優先度） |
| uint_t * | p_load | タスクの数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 |
| uint_t | load | タスクの数 |

【エラーコード】

| | |
|---|---|
| E_CTX | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI3624】<br>・CPUロック状態からの呼出し【NGKI3625】 |
| E_PAR | パラメータエラー<br>・tskpriが有効範囲外【NGKI3626】 |
| E_MACV | メモリアクセス違反<br>・p_load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〕【NGKI3627】 |
| E_OACV | オブジェクトアクセス違反<br>・自タスクが属する保護ドメインに対する参照操作が許可されていない〔P〕【NGKI3767】 |

【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数を参照する．参照したタスクの数は，p_loadが指すメモリ領域に返される【NGKI3629】．

保護機能とマルチプロセッサのいずれかまたは両方に対応するカーネルでは，自タスクと同じスケジューリング単位に属するタスクのみを参照対象とする【NGKI5126】．

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【NGKI3631】．

【TOPPERS/SSP3カーネルにおける規定】

SSP3カーネルでは，get_lodをサポートしない【SSPS0157】．

【μITRON4.0仕様，TOPPERS新世代カーネル統合仕様との関係】

μITRON4.0仕様およびTOPPERS新世代カーネル統合仕様に定義されていないサービスコールである．

## mget_lod　対象指定での実行できるタスクの数の参照〔TP｜TM〕【NGKI3632】

【C言語API】

```
ER ercd = mget_lod(ID schedno, PRI tskpri, uint_t *p_load)
```

【パラメータ】

| | | |
|---|---|---|
| ID | schedno | 参照対象とするスケジューリング単位番号 |
| PRI | tskpri | 参照対象の優先度（対象優先度） |
| uint_t * | p_load | タスクの数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 |
| uint_t | load | タスクの数 |

【エラーコード】

| | |
|---|---|
| E_CTX | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI3633】<br>・CPUロック状態からの呼出し【NGKI3634】 |
| E_ID | 不正ID番号<br>・schednoが有効範囲外【NGKI3635】 |
| E_PAR | パラメータエラー<br>・tskpriが有効範囲外【NGKI3636】 |
| E_MACV | メモリアクセス違反<br>・p_load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〕【NGKI3637】 |
| E_OACV | オブジェクトアクセス違反<br>・schednoで指定した保護ドメインに対する参照操作が許可されていない〔P〕【NGKI3881】 |

【機能】

schednoで指定したスケジューリング単位に属し，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数を参照する．参照したタスクの数は，p_loadが指すメモリ領域に返される【NGKI3639】．

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【NGKI3640】．

【μITRON4.0仕様，TOPPERS新世代カーネル統合仕様との関係】

μITRON4.0仕様およびTOPPERS新世代カーネル統合仕様に定義されていないサービスコールである．

## get_nth　　指定した優先順位のタスクIDの参照〔T〕【NGKI3641】

【C言語API】

```
ER ercd = get_nth(PRI tskpri, uint_t nth, ID *p_tskid)
```

【パラメータ】

| | | |
|---|---|---|
| PRI | tskpri | 参照対象の優先度（対象優先度） |
| uint_n | nth | 参照対象の優先順位 |
| 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<br>・非タスクコンテキストからの呼出し【NGKI3642】<br>・CPUロック状態からの呼出し【NGKI3643】 |
| E_PAR | パラメータエラー<br>・tskpriが有効範囲外【NGKI3644】 |
| E_MACV | メモリアクセス違反<br>・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〔P〕【NGKI3645】 |
| E_OACV | オブジェクトアクセス違反<br>・自タスクが属する保護ドメインに対する参照操作が許可されていない〔P〕【NGKI3768】 |

【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，nthで指定した優先順位のタスクのID番号を参照する．具体的な振舞いは以下の通り．

対象優先度を持つ実行できる状態のタスクの中で，優先順位が(nth＋1)番目に高いタスクのID番号が，p_tskidが指すメモリ領域に返される【NGKI3647】．優先順位が(nth＋1)番目に高いタスクがない，言い換えると，対象優先度を持つ実行できる状態のタスクの数がnth以下の場合には，TSK_NONE（＝0）が返される【NGKI3648】．

保護機能とマルチプロセッサのいずれかまたは両方に対応するカーネルでは，自タスクと同じスケジューリング単位に属するタスクのみを参照対象とする【NGKI5127】．

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【NGKI3650】．

【TOPPERS/SSP3カーネルにおける規定】

SSP3カーネルでは，get_nthをサポートしない【SSPS0159】．

【μITRON4.0仕様，TOPPERS新世代カーネル統合仕様との関係】

μITRON4.0仕様およびTOPPERS新世代カーネル統合仕様に定義されていないサービスコールである．

## mget_nth　対象指定での指定した優先順位のタスクIDの参照〔TP｜TM〕【NGKI3651】

【C言語API】

```
ER ercd = mget_nth(ID schedno, PRI tskpri, uint_t nth, ID *p_tskid)
```

【パラメータ】

| | | |
|---|---|---|
| ID | schedno | 参照対象とするスケジューリング単位番号 |
| PRI | tskpri | 参照対象の優先度（対象優先度） |
| uint_n | nth | 参照対象の優先順位 |
| ID * | p_tskid | タスクの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<br>・非タスクコンテキストからの呼出し【NGKI3652】<br>・CPUロック状態からの呼出し【NGKI3653】 |
| E_ID | 不正ID番号<br>・schednoが有効範囲外【NGKI3654】 |
| E_PAR | パラメータエラー<br>・tskpriが有効範囲外【NGKI3655】 |
| E_MACV | メモリアクセス違反<br>・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〔P〕【NGKI3656】 |
| E_OACV | オブジェクトアクセス違反<br>・schednoで指定した保護ドメインに対する参照操作が許可されていない〔P〕【NGKI3884】 |

【機能】

schednoで指定したスケジューリング単位に属し，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，nthで指定した優先順位のタスクのID番号を参照する．具体的な振舞いは以下の通り．

schednoで指定したスケジューリング単位に属し，対象優先度を持つ実行できる状態のタスクの中で，優先順位が(nth＋1)番目に高いタスクのID番号が，p_tskidが指すメモリ領域に返される【NGKI3658】．優先順位が(nth＋1)番目に高いタスクがない，言い換えると，対象優先度を持つ実行できる状態のタスクの数がnth以下の場合には，TSK_NONE（＝0）が返される【NGKI3659】．

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【NGKI3660】．

【μITRON4.0仕様，TOPPERS新世代カーネル統合仕様との関係】

μITRON4.0仕様およびTOPPERS新世代カーネル統合仕様に定義されていないサービスコールである．

## ENA_SPR　サブ優先度を使用するタスク優先度の設定〔S〕【NGKI3675】

【静的API】

```
ENA_SPR(tskpri)
```

【パラメータ】

| | | |
|---|---|---|
| PRI | tskpri | サブ優先度を使うタスク優先度 |

【エラーコード】

| | |
|---|---|
| E_RSATR | 予約属性<br>・ドメインの囲みの中に記述されている〔P〕【NGKI3676】<br>・クラスの囲みの中に記述されている〔M〕【NGKI3677】 |
| E_PAR | パラメータエラー<br>・tskpriが有効範囲外【NGKI3678】 |

【機能】

tskpriで指定したタスク優先度を，サブ優先度を使用して優先順位を決定するように設定する【NGKI3679】.

tskpriは整数定数式パラメータである【NGKI3680】.

【TOPPERS/ASP3カーネルにおける規定】

ASP3カーネルでは，ENA_SPRをサポートしない【ASPS0233】．ただし，サブ優先度機能拡張パッケージでは，ENA_SPRをサポートする【ASPS0234】.

【TOPPERS/FMP3カーネルにおける規定】

FMP3カーネルでは，ENA_SPRをサポートする【FMPS0170】.

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルでは，ENA_SPRをサポートしない【HRPS0225】.

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルでは，ENA_SPRをサポートしない【HRMPS0117】.

【TOPPERS/SSP3カーネルにおける規定】

SSP3カーネルでは，ENA_SPRをサポートしない【SSPS0162】.

【μITRON4.0仕様，TOPPERS新世代カーネル統合仕様との関係】

μITRON4.0仕様およびTOPPERS新世代カーネル統合仕様に定義されていない静的APIである.

## loc_cpu　　CPUロック状態への遷移〔TI〕【NGKI3538】

【C言語API】

```
ER ercd = loc_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_OACV | オブジェクトアクセス違反<br>・システム状態1に対する通常操作2が許可されていない〔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〔TI〕【NGKI3539】

【C言語API】

```
ER ercd = unl_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_OACV | オブジェクトアクセス違反<br>・システム状態1に対する通常操作2が許可されていない〔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．

## 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<br>・非タスクコンテキストからの呼出し【NGKI2741】<br>・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<br>・システム状態1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<br>・非タスクコンテキストからの呼出し【NGKI2747】<br>・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<br>・システム状態1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．また，自タスクのタスク終了処理要求フラグがセットされていた場合には，自タスクが終了する可能性がある．この場合，ena_dspからはリターンしない．

保護機能対応カーネルにおいては，ディスパッチ許可状態へ遷移した結果，システム周期の切換えとタイムウィンドウの切換えの保留が解除され，システム周期の切換えとタイムウィンドウの切換え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<br>・カーネルの誤動作【NGKI2763】 |
| E_OACV | オブジェクトアクセス違反<br>・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/ASP3カーネルにおける規定】

ASP3カーネルでは，ref_sysをサポートしない．

【TOPPERS/FMP3カーネルにおける規定】

FMP3カーネルでは，ref_sysをサポートしない．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルでは，ref_sysをサポートしない．

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルでは，ref_sysをサポートしない．

【TOPPERS/SSP3カーネルにおける規定】

SSP3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【NGKI3888】．

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がサポートされている［NGKI2860］ |
| TOPPERS_SUPPORT_ATT_PMA | ATT_PMA／att_pmaがサポートされている［NGKI2904］ |

ただし，att_pmaは，動的生成対応カーネルのみでサポートされるAPIであるため，サポートされているかを判定するには，TOPPERS_SUPPORT_DYNAMIC_CREとTOPPERS_SUPPORT_ATT_PMAの両方が定義されていることをチェックする必要がある【NGKI2798】．

### (1) 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［NGKI0112］．

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】．

### (2) 標準メモリリージョン

標準メモリリージョンとは，ATT_MODによって登録するオブジェクトモジュールに含まれる標準のセクションや，ATT_SECによって配置先のメモリリージョン名を指定せずに登録するセクション，オブジェクト生成に必要なメモリ領域でコンフィギュレータが確保するものなどが配置されるメモリリージョンの組である．

標準メモリリージョンは，標準のセクションの中で書込みアクセスを行わないセクションが配置される標準ROMリージョン，書込みアクセスを行うセクションが配置される標準RAMリージョン，書込みアクセスを行わないショートデータセクションが配置される標準ショートROMリージョン，書込みアクセスを行うショートデータセクションが配置される標準ショートRAMリージョン，スタック領域が配置される標準スタックリージョンで構成される．

特定の保護ドメインに属さないセクションが配置される標準メモリリージョンを，保護ドメイン共通の標準メモリリージョンと呼ぶ．保護ドメイン共通の標準メモリリージョンに加えて，各保護ドメイン向けの標準メモリリージョンを定義することができる【NGKI3891】．

自動メモリ配置の場合，保護ドメイン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】．ある保護ドメイン向けの標準メモリリージョンが定義されていない場合には，それに代えて，保護ドメイン共通の標準メモリリージョンが使われる【NGKI3894】．

マルチプロセッサ対応カーネルでは，標準メモリリージョンの扱いは，以下のように拡張される．［NGKI3259］と［NGKI3894］は，以下の規定に置き換わる．

特定のクラスに属さないセクションが配置される標準メモリリージョンを，クラス共通の標準メモリリージョンと呼ぶ．クラス共通の標準メモリリージョンに加えて，各クラス向けの標準メモリリージョンを定義することができる【NGKI3257】．すなわち，［NGKI3891］との組み合わせにより，次の4種類の標準メモリリージョンを定義することができる．

(A) 保護ドメイン共通・クラス共通の標準メモリリージョン
(B) ある保護ドメイン向け・クラス共通の標準メモリリージョン
(C) 保護ドメイン共通・あるクラス向けの標準メモリリージョン
(D) ある保護ドメイン向け・あるクラス向けの標準メモリリージョン

自動メモリ配置の場合，(A)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【NGKI5098】．(B)が定義されていない場合には，それに代えて(A)が使われる【NGKI5099】．(C)が定義されていない場合には，それに代えて(A)が使われる【NGKI2854】．(D)が定義されていない場合には，それに代えて，(C)が定義されていれば(C)が，(C)が定義されておらず(B)が定義されていれば(B)が，(C)も(B)も定義されていない場合は(A)が使われる【NGKI3895】．

【μITRON4.0/PX仕様との関係】

標準メモリリージョンは，μITRON4.0/PX仕様にはない概念である．

【TOPPERS新世代カーネル統合仕様との関係】

標準メモリリージョンの構成要素に，標準ショートROMリージョン，標準ショートRAMリージョン，標準スタックリージョンを追加した．また，各保護ドメイン向けの標準メモリリージョンを定義できることにした．

### (3) 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［NGKI0069］．

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

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［NGKI0070］．

### (4) メモリオブジェクト属性

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_MEMZERO | 0x10U | メモリのクリアを行う |
| 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）を指定せず，メモリオブジェクトに対する通常操作1（書込みアクセス）を許可している場合である【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）を指定せず，メモリオブジェクトに対する通常操作2（読出し・実行アクセス）を許可している場合である【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）を指定し，メモリオブジェクトに対する通常操作2（読出し・実行アクセス）を許可している場合である【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】．

メモリオブジェクト属性にTA_NOWRITEとTA_NOREADを指定し，TA_EXECを指定していない場合，そのメモリ領域は，メモリオブジェクトとしてカーネルに登録されない【NGKI3962】．

TA_MEMINI属性は，システム初期化時に指定した値に初期化されるメモリオブジェクトであることを，TA_MEMZERO属性は，システム初期化時にクリア（言い換えると，0に初期化）されるメモリオブジェクトであることを示す【NGKI5054】．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初期化されない【NGKI5055】．

TA_MEMINI属性のメモリオブジェクトの初期化に用いる初期化データは，保護ドメイン共通の標準ROMリージョン（マルチプロセッサ対応カーネルでは，保護ドメイン共通・クラス共通の標準ROMリージョン．以下同じ）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】．保護ドメイン共通の標準ROMリージョンに，TA_MEMINI属性のメモリオブジェクトを配置した場合，コンフィギュレータは警告メッセージを出力し，そのメモリオブジェクトの初期化データの配置と初期化は行われない【NGKI3984】．

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】．

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 |

【μITRON4.0/PX仕様との関係】

メモリオブジェクト属性を全面的に見直した．

【TOPPERS新世代カーネル統合仕様との関係】

TA_MEMPRSV属性（メモリの初期化を行わない）を廃止し，TA_MEMZERO属性を導入した．

ATT_SECにより，メモリオブジェクト属性にTA_NOWRITEとTA_NOREADを指定し，TA_EXECを指定せずにセクションを登録することで，メモリオブジェクトとして登録されないセクションを配置できるようにした．これにより，セクションの配置のみを行う静的API（LNK_SEC）は廃止した．

TA_MEMINI属性のメモリオブジェクトが，保護ドメイン共通の標準ROMリージョンに配置された場合，メモリオブジェクトの初期化は行わないこととした．

【仕様決定の理由】

TA_IODEV属性を導入したのは，ターゲットプロセッサによっては，周辺デバイスの領域として扱うためには，キャッシュ禁止に加えて，メモリのアクセス順序を変更しないことを指定しなければならないためである．メモリのアクセス順序を変更しないことを指定するメモリオブジェクト属性を，ターゲット定義で用意してもよいが，それを使うとアプリケーションのポータビリティが下がるため，TA_IODEV属性を用意することにした．

(5) 標準のメモリオブジェクト属性

標準のセクション等に対して，標準的に設定すべきメモリオブジェクト属性の組み合わせを，標準のメモリオブジェクト属性と呼ぶ．標準のメモリオブジェクト属性と呼ぶは，以下のシンボルに定義されている【NGKI3963】．

| | |
|---|---|
| TA_TEXTSEC | テキストセクション（textセクション）に設定する標準的なメモリオブジェクト属性 |
| TA_RODATASEC | リードオンリーデータセクション（rodataセクション）に設定する標準的なメモリオブジェクト属性 |
| TA_DATASEC | 初期化データセクション（dataセクション）に設定する標準的なメモリオブジェクト属性 |
| TA_BSSSEC | ゼロ初期化データセクション（bssセクション）に設定する標準的なメモリオブジェクト属性 |
| TA_NOINITSEC | 非初期化データセクションに設定する標準的なメモリオブジェクト属性 |
| TA_LOSEC | 配置のみを行うセクションに設定する標準的なメモリオブジェクト属性 |

これらの標準のメモリオブジェクト属性は，ターゲット定義で別に規定がない限りは，次の通りに定義される【NGKI3964】．

| | |
|---|---|
| TA_TEXTSEC | (TA_NOWRITE\|TA_EXEC) |
| TA_RODATASEC | (TA_NOWRITE) |

| | |
|---|---|
| TA_DATASEC | (TA_MEMINI) |
| TA_BSSSEC | (TA_MEMZERO) |
| TA_NOINITSEC | (TA_NULL) |
| TA_LOSEC | (TA_NOWRITE\|TA_NOREAD) |

【μITRON4.0/PX仕様，TOPPERS新世代カーネル統合仕様との関係】

標準のメモリオブジェクト属性を定義するシンボルを導入した.

### (6) メモリオブジェクトアクセス属性

メモリオブジェクト属性の中で，メモリのアクセスや保護の方法を指定するものを総称して，メモリオブジェクトアクセス属性と呼ぶ.

具体的には，ターゲット非依存のメモリオブジェクト属性の中で，メモリ領域の配置を指定するTA_SDATA属性，メモリの初期化方法を指定するTA_MEMINI属性とTA_MEMZERO属性以外のものが，メモリオブジェクトアクセス属性である【NGKI3986】．ターゲット定義のメモリオブジェクト属性が，メモリオブジェクトアクセス属性であるかは，ターゲット定義である【NGKI3987】.

【μITRON4.0/PX仕様，TOPPERS新世代カーネル統合仕様との関係】

メモリオブジェクトアクセス属性の概念を導入した.

## ATT_REG　メモリリージョンの登録〔SP〕【NGKI2799】

【静的API】

```
ATT_REG(char *regname, { ATR regatr, void *base, size_t size })
```

【パラメータ】

＊メモリリージョンの登録情報

| | | |
|---|---|---|
| char * | regname | 登録するメモリリージョン名 |
| 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_t | 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<br>・regatrが無効【NGKI2800】<br>・保護ドメインの囲みの中に記述されている【NGKI2814】<br>・クラスの囲みの中に記述されている〔M〕【NGKI3260】 |
| E_NOSPT | 未サポート機能<br>・条件については機能の項を参照 |
| E_PAR | パラメータエラー<br>・sizeが0【NGKI2816】<br>・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<br>・登録済みのメモリリージョンの再登録【NGKI2801】<br>・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の登録情報に従って，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，regnameで指定した名称で，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】．

regname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】．

手動メモリ配置の場合には，メモリリージョンの情報は使用されない．ATT_REGを使用した場合，コンフィギュレータは警告メッセージを出力する【NGKI3909】．

【使用上の注意】

baseは整数定数式パラメータである．すなわち，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でなければならない．

【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(char *stdrom, char *stdram,
                        char *stdsrom, char *stdsram, char *stdstk)
```

※ stdsrom，stdsram，stdstkの記述は省略することができる【NGKI3991】．

【パラメータ】

| | | |
|---|---|---|
| char * | stdrom | 標準ROMリージョンのメモリリージョン名 |
| char * | stdram | 標準RAMリージョンのメモリリージョン名 |
| char * | stdsrom | 標準ショートROMリージョンのメモリリージョン名 |
| char * | stdsram | 標準ショートRAMリージョンのメモリリージョン名 |
| char * | stdstk | 標準スタックリージョンのメモリリージョン名 |

【エラーコード】

| | |
|---|---|
| E_OBJ | オブジェクト状態エラー<br>・標準メモリリージョンが定義済み【NGKI3263】<br>・stdrom，stdram，stdsrom，stdsram，stdstkに指定したメモリリージョンが未登録【NGKI3264】<br>・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，標準メモリリージョンを定義する【NGKI3265】．

DEF_SRGを保護ドメインの囲みの外に記述すると，保護ドメイン共通の標準メモリリージョンを定義し，保護ドメインの囲みの中に記述すると，その保護ドメイン向けの標準メモリリージョンを定義する【NGKI3892】．

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，クラス共通の標準メモリリージョンを定義し，クラスの囲みの中に記述すると，そのクラス向けの標準メモリリージョンを定義する【NGKI3266】．

stdsromの記述を省略した場合，標準ショートROMリージョンは標準ROMリージョンと同じメモリリージョンに定義される【NGKI3988】．また，stdsramの記述を省略した場合，標準ショートRAMリージョンは標準RAMリージョンと同じメモリリージョンに定義される【NGKI3989】．stdstkの記述を省略した場合，標準スタックリージョンは標準RAMリージョンと同じメモリリージョンに定義される【NGKI5205】．

stdrom，stdram，stdsrom，stdsram，stdstkは文字列パラメータである【NGKI5149】．

標準RAMリージョン，標準ショートRAMリージョン，標準スタックリージョンは，TA_NOWRITE属性でないメモリリージョンでなければならない．これらに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】．

手動メモリ配置の場合には，標準メモリリージョンの情報は使用されない．DEF_SRGを使用した場合，コンフィギュレータは警告メッセージを出力する【NGKI3910】．

【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．

【TOPPERS新世代カーネル統合仕様との関係】

標準ショートROMリージョン名，標準ショートRAMリージョン名，標準スタックリージョン名の3つのパラメータを追加した．

標準ROMリージョンはTA_NOWRITE属性でなければならないという制約を外した．

## ATT_SEC　セクションの登録〔SP〕【NGKI2818】

【静的API】

```
ATT_SEC(char *secname, { ATR mematr, char *regname },
            { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
```

※ regname，アクセス許可ベクタの記述は省略することができる【NGKI5151】.

【パラメータ】

| | | |
|---|---|---|
| char * | secname | 登録するセクション名 |
| ATR | mematr | メモリオブジェクト属性 |
| char * | regname | セクションを配置するメモリリージョン名 |

＊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<br>・mematrが無効【NGKI2820】<br>・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<br>・条件については機能の項を参照 |
| E_PAR | パラメータエラー<br>・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<br>・登録済みのセクションの再登録【NGKI2821】<br>・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secnameで指定したセクションが，regname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】．ATT_SECにアクセス許可ベクタを記述した場合には，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】.

regnameの記述を省略した場合，secnameで指定したセクションを配置するメモリリージョンは，次のように決定される．ATT_SECを保護ドメインの囲みの外に記述すると，保護ドメイン共通の標準メモリ

リージョンが使われ，保護ドメインの囲みの中に記述すると，その保護ドメイン向けの標準メモリリージョンが使われる【NGKI5199】．マルチプロセッサ対応カーネルでは，ATT_SECをクラスの囲みの外に記述すると，クラス共通の標準メモリリージョンが使われ，クラスの囲みの中に記述すると，そのクラス向けの標準メモリリージョンが使われる【NGKI5200】．mematrにTA_NOWRITEを指定しTA_SDATAを指定しない時は標準ROMリージョンに，TA_NOWRITEとTA_SDATAのいずれも指定しない時は標準RAMリージョンに，TA_NOWRITEを指定せずTA_SDATAを指定した時は標準ショートROMリージョンに，TA_NOWRITEとTA_SDATAの両方を指定した時は標準ショートRAMリージョンに配置される【NGKI5150】．

登録されるメモリオブジェクトと同じメモリリージョンに配置され，メモリオブジェクトアクセス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】．

secnameとregnameは文字列パラメータ，matr，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，ATT_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】．

セクション名として，ターゲット定義の予約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【NGKI5129】．

ATT_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】．

mematrに，TA_MEMINIとTA_MEMZERO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】．

regnameで指定したメモリリージョンがTA_NOWRITE属性である場合には，mematrにTA_NOWRITEを指定しなければならない．指定しなかった場合には，E_RSATRエラーとなる【NGKI5058】．またこの場合，TA_MEMINIとTA_MEMZEROは指定することができない．これらを指定した場合には，E_RSATRエラーとなる【NGKI5059】．

手動メモリ配置の場合には，ATT_SECはサポートされない．ATT_SECを使用した場合，コンフィギュレータは警告メッセージを出力する【NGKI3911】．

#### 【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．

#### 【TOPPERS新世代カーネル統合仕様との関係】

ATT_SECによってアクセス許可ベクタを設定できるようにし，そのための静的API（ATA_SEC）は廃止した．

regnameの記述を省略できることとした．

1つのメモリオブジェクトにまとめて登録する条件を変更した．

ターゲット定義の予約セクションに関して規定した．

regnameで指定したメモリリージョンがTA_NOWRITE属性である時に，mematrにTA_NOWRITEを指定しなかった場合，TA_MEMINIやTA_MEMZEROを指定した場合は，エラーとすることにした．

## ATT_MOD　オブジェクトモジュールの登録〔SP〕【NGKI2845】

【静的API】

```
ATT_MOD(char *modname,
            { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
```

※ アクセス許可ベクタの記述は省略することができる【NGKI5152】.

【パラメータ】

| | | |
|---|---|---|
| char * | modname | 登録するオブジェクトモジュール名 |

＊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_NOSPT | 未サポート機能<br>・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<br>・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に含まれる標準の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．

modname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を配置するメモリリージョンは，次のように決定される．ATT_MODを保護ドメインの囲みの外に記述すると，保護ドメイン共通の標準メモリリージョンが使われ，保護ドメインの囲みの中に記述すると，その保護ドメイン向けの標準メモリリージョンが使われる【NGKI3893】．マルチプロセッサ対応カーネルでは，ATT_MODをクラスの囲みの外に記述すると，クラス共通の標準メモリリージョンが使われ，クラスの囲みの中に記述すると，そのクラス向けの標準メモリリージョンが使われる【NGKI2853】．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，書込みアクセスを行わないショートデータセクションは標準ショートROMリージョンに，書込みアクセスを行うショートデータセクションは標準ショート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，セクション毎に標準のメモリオブジェクト属性が設定される【NGKI2850】．ただし，テキストセクションとリードオンリーデータセクションが，TA_NOWRITE属性でないメモリリージョンに配置され，保護ドメイン共通の標準ROM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_MEMINIが追加で設定される【NGKI3985】.

ATT_MODにアクセス許可ベクタを記述した場合には，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】.

登録されるメモリオブジェクトと同じメモリリージョンに配置され，メモリオブジェクトアクセス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【NGKI2855】.

modnameは文字列パラメータ，acptn1～acptn4は整数定数式パラメータである【NGKI2856】.

ターゲット定義で，ATT_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制限がある場合がある【NGKI2857】. この制限に違反した場合には，E_NOSPTエラーとなる【NGKI2858】.

ターゲット定義で，ATT_MODがサポートされていない場合がある【NGKI2859】. ATT_MODがサポートされている場合には，TOPPERS_SUPPORT_ATT_MODがマクロ定義される【NGKI2860】. サポートされていない場合にATT_MODを使用すると，コンフィギュレータがE_NOSPTエラーを報告する【NGKI2861】.

手動メモリ配置の場合には，ATT_MODはサポートされない. ATT_MODを使用した場合，コンフィギュレータは警告メッセージを出力する【NGKI3913】.

【補足説明】

ATT_MODでは，標準のセクション以外は配置・登録されない. 標準のセクション以外のセクションを配置・登録するためには，ATT_SECを用いる必要がある.

【μITRON4.0/PX仕様との関係】

ATT_MODによってアクセス許可ベクタを設定できるようにし，そのための静的API（ATA_MOD）は廃止した.

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どこに配置するかを規定した.

1つのメモリオブジェクトにまとめて登録できる条件を変更した.

【TOPPERS新世代カーネル統合仕様との関係】

ATT_MODによってアクセス許可ベクタを設定できるようにし，そのための静的API（ATA_MOD）は廃止した.

ATT_MODを保護ドメインの囲みの中に記述した場合，定義されていれば，その保護ドメインの標準メモリリージョンに配置するようにした.

標準ショートROMリージョンと標準ショートRAMリージョンの追加に対応した.

テキストセクションとリードオンリーデータセクションが，TA_NOWRITE属性でないメモリリージョンに配置された場合に，メモリオブジェクトの初期化を行う仕様を追加した.

1つのメモリオブジェクトにまとめて登録できる条件を変更した.

【仕様決定の理由】

テキストセクションとリードオンリーデータセクションがTA_NOWRITE属性でないメモリリージョンに配置された場合に，メモリオブジェクトの初期化を行うこととしたのは，リードオンリーショートデータセクションを標準RAMリージョンに配置した時に，データが初期化されるようにするためである.

| | |
|---|---|
| **ATT_MEM** | メモリオブジェクトの登録〔SP〕【NGKI2862】 |
| **att_mem** | メモリオブジェクトの登録〔TPD〕【NGKI2864】 |

【静的API】

```
ATT_MEM({ ATR accatr, void *base, size_t size },
          { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
```

※ アクセス許可ベクタの記述は省略することができる【NGKI5153】.

【C言語API】

```
ER ercd = att_mem(const T_AMEM *pk_amem)
```

【パラメータ】

| | | |
|---|---|---|
| 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（att_memの場合） |

＊メモリオブジェクトの登録情報（パケットの内容）

| | | |
|---|---|---|
| ATR | accatr | メモリオブジェクトアクセス属性 |
| void * | base | 登録するメモリ領域の先頭番地 |
| size_t | 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）（静的APIのみ）

| | | |
|---|---|---|
| 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<br>・非タスクコンテキストからの呼出し〔s〕【NGKI2865】<br>・CPUロック状態からの呼出し〔s〕【NGKI2866】 |
| E_RSATR | 予約属性<br>・accatrが無効【NGKI2867】<br>・属する保護ドメインの指定が有効範囲外〔s〕【NGKI2868】<br>・クラスの囲みの中に記述されている〔SM〕【NGKI5212】 |
| E_NOSPT | 未サポート機能<br>・条件については機能の項を参照 |

| | |
|---|---|
| E_PAR | パラメータエラー<br>・sizeが0【NGKI2881】<br>・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<br>・システム状態1に対する管理操作が許可されていない〔s〕【NGKI2870】 |
| E_MACV | メモリアクセス違反<br>・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〔s〕【NGKI2871】 |
| E_OBJ | オブジェクト状態エラー<br>・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の登録情報に従って，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，accatrで指定したメモリオブジェクトアクセス属性が設定される【NGKI2873】．ATT_MEMにアクセス許可ベクタを記述した場合には，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】．

静的APIにおいては，accatr，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】．ただし，手動メモリ配置の場合には，baseは整数定数式パラメータである【NGKI3914】．

ターゲット定義で，ATT_MEM／att_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】

自動メモリ配置の場合，ATT_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ている．メモリ領域に対しては，ATT_SECかATT_MODを使用することを推奨する．

ATT_MEM／att_memで登録する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に含まれていても，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／att_memで登録する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．

【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．また，ATT_MEMによってアクセス許可ベクタを設定できるようにし，そのための静的API（ATA_MEM）は廃止した．

メモリオブジェクトに関する属性を，メモリオブジェクト属性（mematr）から，メモリオブジェクトア

クセス属性（accatr）に変更した．

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り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．

登録するメモリオブジェクトの一部分のみがメモリリージョンと重なる場合をエラーとすることとした．

【TOPPERS新世代カーネル統合仕様との関係】

ATT_MEMによってアクセス許可ベクタを設定できるようにし，そのための静的API（ATA_MEM）は廃止した．

メモリオブジェクトに関する属性を，メモリオブジェクト属性（mematr）から，メモリオブジェクトアクセス属性（accatr）に変更した．

| | |
|---|---|
| **ATT_PMA** | 物理メモリ領域の登録〔SP〕【NGKI2883】 |
| **att_pma** | 物理メモリ領域の登録〔TPD〕【NGKI2885】 |

【静的API】

```
ATT_PMA({ ATR accatr, void *base, size_t size, void *paddr  },
          { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
```

※ アクセス許可ベクタの記述は省略することができる【NGKI5154】.

【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<br>（att_pmaの場合） |

＊物理メモリ領域の登録情報（パケットの内容）

| | | |
|---|---|---|
| ATR | accatr | メモリオブジェクトアクセス属性 |
| void * | base | 登録するメモリ領域の先頭番地 |
| size_t | 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）（静的APIのみ）

| | | |
|---|---|---|
| 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<br>・非タスクコンテキストからの呼出し〔s〕【NGKI2886】<br>・CPUロック状態からの呼出し〔s〕【NGKI2887】 |
| E_RSATR | 予約属性<br>・accatrが無効【NGKI3889】<br>・属する保護ドメインの指定が有効範囲外〔s〕【NGKI2888】<br>・クラスの囲みの中に記述されている〔SM〕【NGKI5213】 |

| | |
|---|---|
| E_NOSPT | 未サポート機能<br>・条件については機能の項を参照 |
| E_PAR | パラメータエラー<br>・sizeが0【NGKI2901】<br>・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<br>・システム状態1に対する管理操作が許可されていない〔s〕【NGKI2890】 |
| E_MACV | メモリアクセス違反<br>・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〔s〕【NGKI2891】 |
| E_OBJ | オブジェクト状態エラー<br>・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物理メモリ領域の登録情報に従って，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．

物理アドレス空間において先頭番地がpaddr，サイズがsizeのメモリ領域が，論理アドレス空間においてbaseで指定した番地からアクセスできるように，メモリオブジェクトとして登録される【NGKI2892】．登録されるメモリオブジェクトには，accatrで指定したメモリオブジェクトアクセス属性が設定される【NGKI2893】．ATT_PMAにアクセス許可ベクタを記述した場合には，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【NGKI2894】．

静的APIにおいては，accatr，size，paddr，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【NGKI2897】．ただし，手動メモリ配置の場合には，baseは整数定数式パラメータである【NGKI3961】．

ターゲット定義で，ATT_PMA／att_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制限がある場合がある【NGKI2898】．この制限に違反した場合には，E_NOSPTエラーとなる【NGKI2899】．

base，size，paddr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【NGKI2900】．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論理アドレス空間においてメモリ領域が重なる場合には，E_OBJエラーとなる【NGKI2902】．

ATT_PMA／att_pmaは，MMU（Memory Management Unit）を持つターゲットシステムにおいて，ターゲット定義でサポートされる機能である【NGKI2903】．ATT_PMA／att_pmaがサポートされている場合には，TOPPERS_SUPPORT_ATT_PMAがマクロ定義される【NGKI2904】．ATT_PMAがサポートされていない場合にこの静的APIを使用すると，コンフィギュレータがE_NOSPTエラーを報告する【NGKI2905】．また，att_pmaがサポートされていない場合にatt_pma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【NGKI2906】．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルでは，ターゲット定義で，ATT_PMAをサポートする【HRPS0156】．

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルでは，ターゲット定義で，ATT_PMAをサポートする【HRMPS0118】．

【使用上の注意】

paddrは整数定数式パラメータである．すなわち，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でなければならない．

【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およびサービスコールである．

【TOPPERS新世代カーネル統合仕様との関係】

ATT_PMAによってアクセス許可ベクタを設定できるようにし，そのための静的API（ATA_PMA）は廃止した．

メモリオブジェクトに関する属性を，メモリオブジェクト属性（matr）から，メモリオブジェクトアクセス属性（accatr）に変更した．

## sac_mem　メモリオブジェクトのアクセス許可ベクタの設定〔TPD〕【NGKI2907】

【C言語API】

```
ER ercd = sac_mem(const void *base, const ACVCT *p_acvct)
```

【パラメータ】

| | | |
|---|---|---|
| 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<br>・非タスクコンテキストからの呼出し【NGKI2908】<br>・CPUロック状態からの呼出し【NGKI2909】 |
| E_PAR | パラメータエラー<br>・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<br>・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<br>・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<br>・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<br>・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】.

【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．

base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<br>・非タスクコンテキストからの呼出し【NGKI2917】<br>・CPUロック状態からの呼出し【NGKI2918】 |
| E_PAR | パラメータエラー<br>・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<br>・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<br>・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<br>・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】.

【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.

base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_t size, ID tskid, MODE pmmode)
```

【パラメータ】

| | | |
|---|---|---|
| void * | base | メモリ領域の先頭番地 |
| size_t | 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<br>・非タスクコンテキストからの呼出し【NGKI2925】 |
| E_ID | 不正ID番号<br>・tskidが有効範囲外【NGKI2927】 |
| E_PAR | パラメータエラー<br>・sizeが0【NGKI2929】<br>・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<br>・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】<br>・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<br>・対象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】<br>・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<br>・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<br>・baseとsizeで指定した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を含むメモリオブジェクト（対象メモリオブジェクト）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が返る【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，書込みアクセス，読出しアクセス，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オブジェクト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】．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルの動的生成機能拡張パッケージでは，acre_tskで生成したユーザタスクのユーザスタック領域に対して，他のタスクからアクセスが許可されているかをprb_memによりチェックした場合に，ターゲット定義でE_NOEXSエラーとなる場合がある【HRPS0244】．

【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〕【NGKI3954】

【C言語API】

```
ER ercd = ref_mem(const void *addr, T_RMEM *pk_rmem)
```

【パラメータ】

| | | |
|---|---|---|
| void * | addr | 状態参照するメモリ番地 |
| T_RMEM * | pk_rmem | メモリオブジェクト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 |

＊メモリオブジェクトの現在状態（パケットの内容）

| | | |
|---|---|---|
| ATR | accatr | メモリオブジェクトアクセス属性 |
| void * | base | メモリオブジェクトの先頭番地 |
| size_t | size | メモリオブジェクトのサイズ（バイト数） |

【エラーコード】

| | |
|---|---|
| E_CTX | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI3955】 |
| E_NOEXS | オブジェクト未登録<br>・addr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【NGKI3956】 |
| E_OACV | オブジェクトアクセス違反<br>・対象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【NGKI3957】 |
| E_MACV | メモリアクセス違反<br>・pk_rmem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【NGKI3958】 |

【機能】

addrで指定したメモリ番地を含むメモリオブジェクト（対象メモリオブジェクト）の現在状態を参照する．参照した現在状態は，pk_rmemで指定したパケットに返される【NGKI3959】．

accatrには，対象メモリオブジェクトの登録時に指定したメモリオブジェクトアクセス属性と，対象メモリオブジェクトが次のいずれかに該当する場合には，それを表す属性値とビット毎論理和をとったものが返される【NGKI3960】．

| | | |
|---|---|---|
| TA_ATTMEM | 0x1000U | ATT_MEM／att_mem／ATT_PMA／att_pmaで登録されたメモリオブジェクト |
| TA_USTACK | 0x2000U | ユーザスタック領域 |

【μITRON4.0/PX仕様，TOPPERS新世代カーネル統合仕様との関係】

参照できる現在状態を変更／決定した．

【仕様決定の理由】

ref_memを，非タスクコンテキストからは呼び出すことができず，CPUロック状態からは呼び出すことができるとした理由は，prb_memと同様である．

## 4.9 割込み管理機能

割込み管理機能は，外部割込みとそれによって起動されるプログラムを管理するための機能である．

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】．

〔**割込み要求ライン属性**〕
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**〕
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，ターゲット定義の割込みサービスルーチン属性を指定できる場合がある【NGKI5176】．ターゲット定義の割込みサービスルーチン属性がない場合，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(EXINF 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**〕

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
}
```

〔**カーネル構成マクロ**〕

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 | カーネル管理の割込み優先度の最小値（最高値）【NGKI2961】 |
| TMAX_INTPRI | カーネル管理の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 |
| TNUM_ISRID | 登録できる割込みサービスルーチンの数（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割込みサービスルーチンの数に一致）【NGKI3612】 |

| | |
|---|---|
| TOPPERS_SUPPORT_DIS_INT | dis_intがサポートされている［NGKI3092］ |
| TOPPERS_SUPPORT_ENA_INT | ena_intがサポートされている［NGKI3105］ |
| TOPPERS_SUPPORT_CLR_INT | clr_intがサポートされている［NGKI3926］ |
| TOPPERS_SUPPORT_RAS_INT | ras_intがサポートされている［NGKI3938］ |
| TOPPERS_SUPPORT_PRB_INT | prb_intがサポートされている［NGKI3950］ |

**【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/ASP3カーネルにおける規定】

ASP3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/FMP3カーネルにおける規定】

FMP3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/HRP3カーネルにおける規定】

HRP3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/HRMP3カーネルにおける規定】

HRMP3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【HRMPS0119】．

【TOPPERS/SSP3カーネルにおける規定】

SSP3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，TNUM_ISRID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．

【TOPPERS新世代カーネル統合仕様との関係】

TNUM_ISRIDは，TOPPERS新世代カーネル統合仕様に定義のないカーネル構成マクロである．

| | |
|---|---|
| **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（cfg_intの場合） |

＊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<br>・非タスクコンテキストからの呼出し〔s〕【NGKI2967】<br>・CPUロック状態からの呼出し〔s〕【NGKI2968】 |
| E_RSATR | 予約属性<br>・intatrが無効【NGKI2969】<br>・カーネルドメインの囲みの中に記述されていない〔SP〕【NGKI2989】<br>・属するクラスの指定が有効範囲外〔sM〕【NGKI2970】<br>・クラスの囲みの中に記述されていない〔SM〕【NGKI2971】<br>・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<br>・intnoが有効範囲外【NGKI2972】<br>・intpriが有効範囲外【NGKI2973】<br>・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<br>・システム状態1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<br>・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<br>・対象割込み要求ラインに対して属性が設定済み〔S〕【NGKI2976】<br>・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）の属性を，各パラメータで指定した属性の設定情報に従って設定する【NGKI2977】．マルチプロセッサ対応カーネルでは，割込み要求ラインが属するクラスも設定する【NGKI5143】．

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，カーネル管理外に固定されている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，カーネル管理に固定されている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】．

ターゲット定義の制限で，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】．また，マルチプロセッサ対応カーネルにおいて，ターゲット定義の制限により，対象割込み要求ラインが属することができないクラスに属させようとした場合には，E_RSATRエラーとなる【NGKI5184】．

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には，E_PARエラーとなる【NGKI2987】．どのプロセッサからどの割込み要求ラインの属性を設定できないかは，ターゲット定義である【NGKI5177】．

保護機能対応カーネルにおいて，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】．

【補足説明】

ターゲット定義の制限で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．

【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】 |
| **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, EXINF 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識別名（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（acre_isrの場合） |

＊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 |
| EXINF | 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<br>・非タスクコンテキストからの呼出し〔s〕【NGKI2996】<br>・CPUロック状態からの呼出し〔s〕【NGKI2997】 |
| E_RSATR | 予約属性<br>・isratrが無効【NGKI2998】<br>・属する保護ドメインの指定がカーネルドメイン以外〔sP〕【NGKI2999】<br>・カーネルドメインの囲みの中に記述されていない〔SP〕【NGKI3000】<br>・属するクラスの指定が有効範囲外〔sM〕【NGKI3001】<br>・クラスの囲みの中に記述されていない〔SM〕【NGKI3002】<br>・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<br>・intnoが有効範囲外【NGKI3003】<br>・isrがプログラムの先頭番地として正しくない【NGKI3004】<br>・isrpriが有効範囲外【NGKI3005】 |
| E_OACV | オブジェクトアクセス違反<br>・属する保護ドメインに対する通常操作1が許可されていない〔sP〕【NGKI3977】 |
| E_MACV | メモリアクセス違反<br>・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<br>・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<br>・isridで指定した識別名のオブジェクトが登録済み〔S〕【NGKI3009】<br>・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の生成情報に従って，割込みサービスルーチンを生成する【NGKI3010】．

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）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が，生成する割込みサービスルーチンと異なるクラスに属する場合にも，E_RSATRエラーとなる【NGKI5142】．さらに，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/ASP3カーネルにおける規定】

ASP3カーネルの動的生成機能拡張パッケージでは，acre_isrをサポートする【ASPS0195】．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルの動的生成機能拡張パッケージでは，acre_isrをサポートする【HRPS0208】．

【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優先度）を追加した．割込み

サービスルーチンを追加する静的API（ATT_ISR）を廃止し，生成する静的API（CRE_ISR）を新設した．CRE_ISRは，μITRON4.0仕様に定義されていない静的APIである．

【μITRON4.0/PX仕様との関係】

アクセス権を制御するためのアクセスの種別を，保護ドメインに対する通常操作1に変更した．

【TOPPERS新世代カーネル統合仕様との関係】

割込みサービスルーチンを追加する静的API（ATT_ISR）を廃止した．

アクセス権を制御するためのアクセスの種別を，保護ドメインに対する通常操作1に変更した．

## 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<br>・カーネルドメインの囲みの中に記述されていない〔P〕【NGKI3982】<br>・クラスの囲みの中に記述されていない〔M〕【NGKI3022】 |

【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/ASP3カーネルにおける規定】

ASP3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/HRP3カーネルにおける規定】

HRP3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識別名（SAC_ISRの場合）またはID番号（sac_isrの場合） |
| 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（sac_isrの場合） |

＊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<br>・非タスクコンテキストからの呼出し〔s〕【NGKI3028】<br>・CPUロック状態からの呼出し〔s〕【NGKI3029】 |
| E_ID | 不正ID番号<br>・isridが有効範囲外〔s〕【NGKI3030】 |
| E_RSATR | 予約属性<br>・カーネルドメインの囲みの中に記述されていない〔S〕【NGKI3031】<br>・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<br>・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<br>・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<br>・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<br>・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】<br>・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/HRP3カーネルにおける規定】

HRP3カーネルでは，SAC_ISRをサポートしない【HRPS0162】．HRP3カーネルの動的生成機能拡張パッケージでは，SAC_ISRとsac_isrをサポートする【HRPS0209】.

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルでは，SAC_ISRをサポートしない【HRMPS0120】.

## 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<br>・非タスクコンテキストからの呼出し【NGKI3041】<br>・CPUロック状態からの呼出し【NGKI3042】 |
| E_ID | 不正ID番号<br>・isridが有効範囲外【NGKI3043】 |
| E_NOEXS | オブジェクト未登録<br>・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<br>・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<br>・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/ASP3カーネルにおける規定】

ASP3カーネルの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/HRP3カーネルにおける規定】

HRP3カーネルの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】．

## 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/ASP3カーネルにおける規定】

ASP3カーネルでは，ref_isrをサポートしない．

【TOPPERS/FMP3カーネルにおける規定】

FMP3カーネルでは，ref_isrをサポートしない．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルでは，ref_isrをサポートしない．

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルでは，ref_isrをサポートしない．

【TOPPERS/SSP3カーネルにおける規定】

SSP3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】

【静的API】

```
DEF_INH(INHNO inhno, { ATR inhatr, INTHDR inthdr })
```

【パラメータ】

| | | |
|---|---|---|
| INHNO | inhno | 割込みハンドラ番号 |

＊割込みハンドラの定義情報

| | | |
|---|---|---|
| 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 |

【エラーコード】

| | |
|---|---|
| E_RSATR | 予約属性<br>・inhatrが無効【NGKI3052】<br>・カーネルドメインの囲みの中に記述されていない〔P〕【NGKI3070】<br>・クラスの囲みの中に記述されていない〔M〕【NGKI3054】<br>・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<br>・inhnoが有効範囲外【NGKI3055】<br>・inthdrがプログラムの先頭番地として正しくない【NGKI3056】<br>・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<br>・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の定義情報に従って，割込みハンドラを定義する【NGKI3059】．

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】．

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】．カーネル管理外に固定されている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_RSATRエラーとなる【NGKI3067】．逆に，カーネル管理に固定されている割込みハンドラを対象として，inhatrにTA_NONKERNELを指定した場合には，E_RSATRエラーとなる【NGKI3068】．また，対象割込みハンドラ番号に対応する割込み要求ラインにTMIN_INTPRIより小さい値の割込み優先度が設定されており，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，対応する割込み要求ラインにTMIN_INTPRI以上の値の割込み優先度が設定されており，inhatrにTA_NONKERNELを指定した場合にも，E_OBJエラーとなる【NGKI3066】．

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，対象割込みハンドラ番号に対応する割込み要求ラインが，登録する割込みハンドラと異なるクラスに属する場合にも，E_RSATRエラーとなる【NGKI5141】．さらに，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】．

ターゲット定義で，対象割込みハンドラを定義できない場合には，E_PARエラーとなる【NGKI3078】．

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】．

【μITRON4.0仕様との関係】

inthdrのデータ型をINTHDRに変更した．割込みハンドラを定義するサービスコール（def_inh）を廃止した．

【TOPPERS新世代カーネル統合仕様との関係】

割込みハンドラを定義するサービスコール（def_inh）を廃止した．

## dis_int　　割込みの禁止〔TI〕【NGKI3555】

【C言語API】

```
ER ercd = dis_int(INTNO intno)
```

【パラメータ】

| | | |
|---|---|---|
| INTNO | intno | 割込み番号 |

【リターンパラメータ】

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

【エラーコード】

| | |
|---|---|
| E_NOSPT | 未サポートエラー<br>・条件については機能の項を参照 |
| E_PAR | パラメータエラー<br>・intnoが有効範囲外【NGKI3083】<br>・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<br>・システム状態1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<br>・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，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】.

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を呼び出した処理単位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などが，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，あるプロセッサに割り付けられた処理単位から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.

【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.

dis_intは，CPUロック状態でも呼び出せるものとした．また，非タスクコンテキストからも呼び出せるものとした．

【TOPPERS新世代カーネル統合仕様との関係】

dis_intは，非タスクコンテキストからも呼び出せるものとした．

## ena_int　　割込みの許可〔TI〕【NGKI3556】

【C言語API】

```
ER ercd = ena_int(INTNO intno)
```

【パラメータ】

| | | |
|---|---|---|
| INTNO | intno | 割込み番号 |

【リターンパラメータ】

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

【エラーコード】

| | |
|---|---|
| E_NOSPT | 未サポートエラー<br>・条件については機能の項を参照 |
| E_PAR | パラメータエラー<br>・intnoが有効範囲外【NGKI3096】<br>・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<br>・システム状態1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<br>・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，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】.

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を呼び出した処理単位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などが，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，あるプロセッサに割り付けられた処理単位から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.

【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．

ena_intは，CPUロック状態でも呼び出せるものとした．また，非タスクコンテキストからも呼び出せるものとした．

【TOPPERS新世代カーネル統合仕様との関係】

ena_intは，非タスクコンテキストからも呼び出せるものとした．

## clr_int　　割込み要求のクリア〔TI〕【NGKI3920】

【C言語API】

```
ER ercd = clr_int(INTNO intno)
```

【パラメータ】

| | | |
|---|---|---|
| INTNO | intno | 割込み番号 |

【リターンパラメータ】

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

【エラーコード】

| | |
|---|---|
| E_NOSPT | 未サポートエラー<br>・条件については機能の項を参照 |
| E_PAR | パラメータエラー<br>・intnoが有効範囲外【NGKI3921】<br>・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<br>・システム状態1に対する通常操作2が許可されていない〔P〕【NGKI3922】 |
| E_OBJ | オブジェクト状態エラー<br>・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【NGKI3923】<br>・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）に対する割込み要求をクリアする．具体的な振舞いは以下の通り．

対象割込み要求ラインがエッジトリガである場合，対象割込み要求ラインの割込み要求フラグがクリアされる【NGKI3924】

対象割込み要求ラインがレベルトリガである場合のclr_intの振舞いはターゲット定義である【NGKI3928】．ターゲット定義でエラーとなる場合には，E_OBJエラーとなる【NGKI3929】．

ターゲット定義で，clr_intがサポートされていない場合がある【NGKI3925】．clr_intがサポートされている場合には，TOPPERS_SUPPORT_CLR_INTがマクロ定義される【NGKI3926】．サポートされていない場合にclr_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【NGKI3927】．

ターゲット定義で，対象割込み要求ラインに対する割込み要求をクリアできない場合には，E_PARエラーとなる【NGKI3930】．具体的には，マルチプロセッサ対応カーネルにおいて，clr_intを呼び出した処理単位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に対する割込み要求がクリアできない場合などが，これに該当する．

ターゲット定義で，割込み要求のクリアの振舞いが，この仕様の規定と異なる場合がある【NGKI3931】．特にマルチプロセッサ対応カーネルでは，あるプロセッサに割り付けられた処理単位か

らclr_intを呼び出して割込み要求をクリアしても，他のプロセッサに対する割込み要求はクリアされない場合がある．

【μITRON4.0仕様，TOPPERS新世代カーネル統合仕様との関係】

μITRON4.0仕様およびTOPPERS新世代カーネル統合仕様に定義されていないサービスコールである．

## ras_int　　割込みの要求〔TI〕【NGKI3932】

【C言語API】

```
ER ercd = ras_int(INTNO intno)
```

【パラメータ】

| | | |
|---|---|---|
| INTNO | intno | 割込み番号 |

【リターンパラメータ】

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

【エラーコード】

| | |
|---|---|
| E_NOSPT | 未サポートエラー<br>・条件については機能の項を参照 |
| E_PAR | パラメータエラー<br>・intnoが有効範囲外【NGKI3933】<br>・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<br>・システム状態1に対する通常操作2が許可されていない〔P〕【NGKI3934】 |
| E_OBJ | オブジェクト状態エラー<br>・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【NGKI3935】<br>・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）に対して割込みを要求する．具体的な振舞いは以下の通り．

対象割込み要求ラインがエッジトリガである場合，対象割込み要求ラインの割込み要求フラグがセットされる【NGKI3936】．

対象割込み要求ラインがレベルトリガである場合のras_intの振舞いはターゲット定義である【NGKI3940】．ターゲット定義でエラーとなる場合には，E_OBJエラーとなる【NGKI3941】．

ターゲット定義で，ras_intがサポートされていない場合がある【NGKI3937】．ras_intがサポートされている場合には，TOPPERS_SUPPORT_RAS_INTがマクロ定義される【NGKI3938】．サポートされていない場合にras_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【NGKI3939】．

ターゲット定義で，対象割込み要求ラインに対して割込みを要求できない場合には，E_PARエラーとなる【NGKI3942】．具体的には，マルチプロセッサ対応カーネルにおいて，ras_intを呼び出した処理単位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に対して割込みを要求できない場合などが，これに該当する．

ターゲット定義で，割込みの要求の振舞いが，この仕様の規定と異なる場合がある【NGKI3943】．特に

マルチプロセッサ対応カーネルでは，あるプロセッサに割り付けられた処理単位からras_intを呼び出して，複数のプロセッサに接続された割込み要求ラインに対して割込みを要求しても，他のプロセッサに対しては割込みが要求されない場合がある．

【μITRON4.0仕様，TOPPERS新世代カーネル統合仕様との関係】

μITRON4.0仕様およびTOPPERS新世代カーネル統合仕様に定義されていないサービスコールである．

## prb_int　　割込み要求のチェック〔TI〕【NGKI3944】

【C言語API】

```
ER_BOOL state = prb_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_BOOL | state | 割込み要求の有無（trueまたはfalse）またはエラーコード |

【エラーコード】

| | |
|---|---|
| E_NOSPT | 未サポートエラー<br>・条件については機能の項を参照 |
| E_PAR | パラメータエラー<br>・intnoが有効範囲外【NGKI3945】<br>・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<br>・システム状態1に対する参照操作が許可されていない〔P〕【NGKI3946】 |
| E_OBJ | オブジェクト状態エラー<br>・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【NGKI3947】 |

【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）に対する割込み要求の有無をチェックする．具体的な振舞いは以下の通り．

対象割込み要求ラインがレベルトリガである場合，割込み要求ラインが割込みを要求する状態である場合にはtrue，そうでない場合にはfalseが返る【NGKI5214】．対象割込み要求ラインがエッジトリガである場合，対象割込み要求ラインの割込み要求フラグがセットされている場合にはtrue，そうでない場合にはfalseが返る【NGKI5215】．

ターゲット定義で，prb_intがサポートされていない場合がある【NGKI3949】．prb_intがサポートされている場合には，TOPPERS_SUPPORT_PRB_INTがマクロ定義される【NGKI3950】．サポートされていない場合にprb_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【NGKI3951】．

ターゲット定義で，対象割込み要求ラインに対する割込み要求をチェックできない場合には，E_PARエラーとなる【NGKI3952】．具体的には，マルチプロセッサ対応カーネルにおいて，prb_intを呼び出した処理単位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に対する割込み要求がチェックできない場合などが，これに該当する．

ターゲット定義で，割込み要求のチェックの振舞いが，この仕様の規定と異なる場合がある【NGKI3953】．特にマルチプロセッサ対応カーネルでは，割込み要求がプロセッサ毎に管理されており，あるプロセッサに割り付けられた処理単位からprb_intを呼び出して割込み要求をチェックしても，

他のプロセッサに対する割込み要求がそれと一致していない場合がある．

【μITRON4.0仕様，TOPPERS新世代カーネル統合仕様との関係】

μITRON4.0仕様およびTOPPERS新世代カーネル統合仕様に定義されていないサービスコールである．

## 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/ASP3カーネルにおける規定】

ASP3カーネルでは，ref_intをサポートしない．

【TOPPERS/FMP3カーネルにおける規定】

FMP3カーネルでは，ref_intをサポートしない．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルでは，ref_intをサポートしない．

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルでは，ref_intをサポートしない．

【TOPPERS/SSP3カーネルにおける規定】

SSP3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<br>・非タスクコンテキストからの呼出し【NGKI3108】<br>・CPUロック状態からの呼出し【NGKI3109】 |
| E_PAR | パラメータエラー<br>・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<br>・システム状態1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，自タスクのタスク終了処理要求フラグがセットされていた場合には，自タスクが終了する可能性がある．この場合，chg_ipmからはリターンしない．

保護機能対応カーネルにおいて，割込み優先度マスクをTIPM_ENAALLに変更した場合，システム周期の切換えとタイムウィンドウの切換えの保留が解除され，システム周期の切換えとタイムウィンドウの切換えが起こる可能性がある．

【TOPPERS/SSP3カーネルにおける規定】

SSP3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<br>・非タスクコンテキストからの呼出し【NGKI3116】<br>・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<br>・システム状態1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<br>・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/SSP3カーネルにおける規定】

SSP3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例外ハンドラ属性がない場合，CPU例外ハンドラ属性には，TA_NULLを指定しなければならない【NGKI5178】．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．

【TOPPERS新世代カーネル統合仕様との関係】

CPU例外発生時のタスク例外処理保留状態を参照するサービスコール（xsns_xpn）を廃止した．

## DEF_EXC　CPU例外ハンドラの定義〔S〕【NGKI3127】

【静的API】

```
DEF_EXC(EXCNO excno, { ATR excatr, EXCHDR exchdr })
```

【パラメータ】

| | | |
|---|---|---|
| EXCNO | excno | CPU例外ハンドラ番号 |

＊CPU例外ハンドラの定義情報

| | | |
|---|---|---|
| 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_RSATR | 予約属性<br>・excatrが無効【NGKI3131】<br>・カーネルドメインの囲みの中に記述されていない〔P〕【NGKI3143】<br>・クラスの囲みの中に記述されていない〔M〕【NGKI3133】<br>・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<br>・excnoが有効範囲外【NGKI3134】<br>・exchdrがプログラムの先頭番地として正しくない【NGKI3135】 |
| E_OBJ | オブジェクト状態エラー<br>・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済み【NGKI3141】 |

【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例外ハンドラの定義情報に従って，CPU例外ハンドラを定義する【NGKI3138】.

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】.

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割付け可能プロセッサが，CPU例外が発生する可能性があるプロセッサと一致していない場合には，E_RSATRエラーとなる【NGKI3146】.

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】.

【μITRON4.0仕様，TOPPERS新世代カーネル統合仕様との関係】

CPU例外ハンドラを定義するサービスコール（def_exc）を廃止した.

## 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 state = 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，CPU例外が発生した時の状態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．詳しくは，「2.8.2 CPU例外ハンドラから呼び出せるサービスコール」の節を参照すること．

【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/SSP3カーネルにおける規定】

SSP3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.

## 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呼び出されて，特権モードで実行されるルーチンで，カーネルが実行を制御する処理単位である【NGKI3159】．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】．拡張サービスコールは，機能コードによって識別する【NGKI5246】．

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．

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コードの最大値（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】 |

**【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_t 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（def_svcの場合） |

＊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_t | 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<br>・非タスクコンテキストからの呼出し〔s〕【NGKI3170】<br>・CPUロック状態からの呼出し〔s〕【NGKI3171】 |
| E_RSATR | 予約属性<br>・svcatrが無効【NGKI3172】<br>・カーネルドメインの囲みの中に記述されていない〔S〕【NGKI3183】<br>・クラスの囲みの中に記述されている〔SM〕【NGKI3187】<br>・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<br>・fncdが0以下【NGKI3173】<br>・fncdがTMAX_FNCDよりも大きい〔s〕【NGKI3174】<br>・extsvcがプログラムの先頭番地として正しくない【NGKI3175】 |
| E_OACV | オブジェクトアクセス違反<br>・システム状態1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<br>・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<br>・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の定義情報に従って，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にNULLを指定した場合には，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，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にNULLを指定した場合）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】．

【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<br>・条件については機能の項を参照 |
| E_RSFN | 予約機能コード<br>・fncdが0以下【NGKI3192】<br>・fncdがTMAX_FNCDよりも大きい【NGKI3193】<br>・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<br>・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】．

cal_svcを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 時間パーティショニング機能

時間パーティショニング機能は，保護機能対応カーネルにおいて，システム周期，システム動作モード，タイムウィンドウに関する設定を行うための機能である．保護機能対応でないカーネルでは，時間パーティショニングをサポートしない【NGKI5001】．

【μITRON4.0/PX仕様・TOPPERS新世代カーネル統合仕様との関係】

μITRON4.0/PX仕様およびTOPPERS新世代カーネル統合仕様に定義されていない機能である．

### (1) システム動作モード

システム動作モードは，システムの動作状態に応じてタイムウィンドウの設定を変更できるようにするために，システムの動作状態を識別するためのカーネルオブジェクトである．システム動作モードは，システム動作モードIDと呼ぶID番号によって識別する［NGKI0577］．

各システム動作モードが持つ情報は次の通り【NGKI5003】．

- システム動作モード属性
- 遷移先システム動作モード
- 登録されているタイムウィンドウのリスト

システム動作モード属性には，次の属性を指定することができる【NGKI5004】．

```
    TA_INISOM       0x01U       初期システム動作モード
```

TA_INISOMを指定したシステム動作モードを，初期システム動作モードと呼ぶ．TA_INISOMを指定したシステム動作モードがない場合，システム周期停止モードが，初期システム動作モードとなる【NGKI5005】．TA_INISOMを指定したシステム動作モードが複数ある場合，コンフィギュレータがE_OBJエラーを報告する【NGKI5006】．

遷移先システム動作モードは，次のシステム周期でどのシステム動作モードに遷移すべきであるかを示す【NGKI5007】．

また，システム動作モードに関連して，カーネルが持つ情報は次の通り【NGKI5008】．

- 現在のシステム動作モード
- 次のシステム周期での遷移先システム動作モード

現在のシステム動作モードは，カーネルの起動時に，初期システム周期動作モードに初期化される【NGKI5009】．

次のシステム周期での遷移先システム動作モードは，各システム周期の開始時に，現在のシステム動作モードに対して設定された遷移先システム動作モードに初期化される【NGKI5010】．なお，システム周期停止モードにおいては，次のシステム周期での遷移先システム動作モードは意味を持たない．

### (2) タイムウィンドウ

タイムウィンドウは，システム周期内の連続した時間区間である．タイムウィンドウは，それを識別するための番号を持たない【NGKI5094】．

タイムウィンドウが持つ情報は次の通り【NGKI5095】．

- 割当て対象のユーザドメイン

- 登録対象のシステム動作モード
- システム周期内での順序
- タイムウィンドウの長さ
- タイムウィンドウ通知の通知方法
- 登録対象のプロセッサ（マルチプロセッサ対応カーネルの場合）

タイムウィンドウ通知（time window notification）は，タイムウィンドウの実行開始をアプリケーションに通知するための機能である．タイムウィンドウ通知が登録されているタイムウィンドウの実行が開始されると，タイムウィンドウの登録時に指定された通知処理を行う【NGKI5096】．タイムウィンドウの通知処理は，「4.6.2　タイムイベントの通知処理」の節で説明したタイムイベントの通知処理と同じ方法で行うことができる【NGKI5202】．

タイムウィンドウに関連するカーネル構成マクロは次の通り．

| | |
|---|---|
| TMAX_TWDTIM | タイムウィンドウの長さに指定できる最大値【NGKI5051】 |

## DEF_SCY　　システム周期の設定〔SP〕【NGKI5012】

【静的API】

```
DEF_SCY({ RELTIM scyctim })
```

【パラメータ】

| | | |
|---|---|---|
| RELTIM | scyctim | システム周期 |

【エラーコード】

| | |
|---|---|
| E_RSATR | 予約属性<br>・保護ドメインの囲みの中に記述されている【NGKI5013】<br>・クラスの囲みの中に記述されている〔M〕【NGKI5014】 |
| E_PAR | パラメータエラー<br>・scyctimが有効範囲（0より大きく，TMAX_RELTIM以下）外【NGKI5015】 |
| E_OBJ | オブジェクト状態エラー<br>・システム周期が設定済み【NGKI5016】 |

【機能】

パラメータに従って，システム周期を設定する【NGKI5017】.

scyctimは整数定数式パラメータである【NGKI5018】.

システム周期を設定しない場合，時間パーティショニングを行わない【NGKI5019】.

## CRE_SOM　システム動作モードの生成〔SP〕【NGKI5020】

【静的API】

```
CRE_SOM(ID somid, { ATR somatr, ID nxtsom })
```

※ nxtsomの指定は省略することができる【NGKI5028】.

【パラメータ】

| | | |
|---|---|---|
| ID | somid | 生成するシステム動作モードの識別名 |

＊システム動作モードの生成情報

| | | |
|---|---|---|
| ATR | somatr | システム動作モード属性 |
| ID | nxtsom | 遷移先システム動作モード |

【エラーコード】

| | |
|---|---|
| E_RSATR | 予約属性<br>・somatrが無効【NGKI5021】<br>・保護ドメインの囲みの中に記述されている【NGKI5022】<br>・クラスの囲みの中に記述されている〔M〕【NGKI5023】 |
| E_ID | 不正ID番号<br>・nxtsomが不正【NGKI5024】 |
| E_OBJ | オブジェクト状態エラー<br>・somidで指定した識別名のオブジェクトが登録済み〔S〕【NGKI5218】<br>・TA_INISOM属性のシステム動作モードが設定済み【NGKI5025】 |

【機能】

各パラメータで指定したシステム動作モードの生成情報に従って，システム動作モードを生成する【NGKI5026】.

somidはオブジェクト識別名，somatrは整数定数式パラメータ，nxtsomは一般定数式パラメータである【NGKI5027】．nxtsomの指定を省略した場合，遷移先システム動作モードは，生成したシステム動作モード自身に設定される【NGKI5029】.

システム周期が設定されていない場合，コンフィギュレータは警告メッセージを出力し，この静的APIを無視する【NGKI5039】.

## chg_som　　システム動作モードの変更〔TP〕【NGKI5030】

【C言語API】

```
ER ercd = chg_som(ID somid)
```

【パラメータ】

| | | |
|---|---|---|
| ID | somid | 変更先のシステム動作モードの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<br>・非タスクコンテキストからの呼出し【NGKI5031】<br>・CPUロック状態からの呼出し【NGKI5032】 |
| E_ID | 不正ID番号<br>・somidが有効範囲外【NGKI5033】 |
| E_OACV | オブジェクトアクセス違反<br>・システム状態1に対する通常操作1が許可されていない【NGKI5034】 |
| E_OBJ | オブジェクト状態エラー<br>・システム周期が設定されていない【NGKI5035】 |

【機能】

somidで指定したシステム動作モードに切り換える．具体的な振舞いは以下の通り．

次のシステム周期での遷移先システム動作モードを，somidで指定したシステム動作モードに設定する【NGKI5036】．すなわち，再度chg_somを呼び出さない限りは，現在のシステム周期の終了後に，somidで指定したシステム動作モードに切り換わる．

ただし，現在のシステム動作モードがシステム周期停止モード（TSOM_STP）の場合には，somidで指定したシステム動作モードに即座に切り換わる【NGKI5037】．

somidにTSOM_STP（＝-1）を指定することもできる【NGKI5038】．指定した場合，現在のシステム周期の終了後（現在のシステム動作モードがシステム周期停止モードの場合は，即座）に，システム周期停止モードに切り換わる．

マルチプロセッサ対応カーネルにおいて，chg_somを実行しているプロセッサが，一時的に切換え前のシステム動作モードで実行している状況［NGKI0636］では，somidで指定したシステム動作モードに切り換わるのは，切換え後のシステム周期の終了後になる【NGKI5134】．

## get_som　　システム動作モードの参照〔TP〕【NGKI5060】

【C言語API】

```
ER ercd = get_som(ID *p_somid)
```

【パラメータ】

| | | |
|---|---|---|
| ID * | p_somid | 現在のシステム動作モードのID番号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 | somid | 現在のシステム動作モードのID番号 |

【エラーコード】

| | |
|---|---|
| E_CTX | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI5061】<br>・CPUロック状態からの呼出し【NGKI5062】 |
| E_OACV | オブジェクトアクセス違反<br>・システム状態1に対する参照操作が許可されていない【NGKI5063】 |
| E_MACV | メモリアクセス違反<br>・p_somid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【NGKI5064】 |
| E_OBJ | オブジェクト状態エラー<br>・システム周期が設定されていない【NGKI5065】 |

【機能】

現在のシステム動作モードを参照する．具体的には，現在のシステム動作モードがシステム周期停止モードである場合にはTSOM_STP（＝-1）が，そうでない場合には現在のシステム動作モードのID番号が，p_somidが指すメモリ領域に返される【NGKI5066】．

マルチプロセッサ対応カーネルにおいて，get_somを実行しているプロセッサが，一時的に切換え前のシステム動作モードで実行している状況［NGKI0636］では，p_somidで指すメモリ領域に返されるのは，切換え後のシステム動作モードになる【NGKI5135】．

## ATT_TWD　タイムウィンドウの登録〔SP〕【NGKI5040】

【静的API】

```
ATT_TWD({ ID domid, ID somid, int_t twdord, PRCTIM twdlen,
                                          < 通知方法の指定 > })
```

※ 通知方法の指定の記述は省略することができる【NGKI5155】.

【パラメータ】

＊タイムウィンドウの登録情報

| | | |
|---|---|---|
| ID | domid | 割り当てるユーザドメインのID番号 |
| ID | somid | 登録対象のシステム動作モードの識別名 |
| int_t | twdord | システム周期内での順序 |
| PRCTIM | twdlen | タイムウィンドウの長さ |
| T_NFYINFO | nfyinfo | タイムウィンドウ通知の通知方法 |

【エラーコード】

| | |
|---|---|
| E_RSATR | 予約属性<br>・保護ドメインの囲みの中に記述されている【NGKI5041】<br>・クラスの囲みの中に記述されていない〔M〕【NGKI5042】 |
| E_ID | 不正ID番号<br>・domidが有効範囲外【NGKI5043】<br>・somidが不正【NGKI5044】<br>・通知方法中のオブジェクトIDが有効範囲外［NGKI3704］［NGKI3707］［NGKI3710］［NGKI3713］［NGKI3715］ |
| E_PAR | パラメータエラー<br>・twdlenが0，またはTMAX_TWDTIMより大きい【NGKI5045】<br>・通知方法中の通知処理モードの値が正しくない［NGKI3730］［NGKI3721］<br>・通知方法中の変数の番地が正しくない［NGKI3699］［NGKI3701］［NGKI3897］<br>・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<br>・通知方法中のオブジェクトに対する操作が，割り当てるユーザドメインに許可されていない［NGKI3705］［NGKI3708］［NGKI3711］［NGKI3716］<br>・通知方法にタイムイベントハンドラの呼出しを指定［NGKI3695］ |
| E_MACV | メモリアクセス違反<br>・通知方法中の変数に対する操作が，割り当てるユーザドメインに許可されていない［NGKI3702］［NGKI3898］ |

【機能】

各パラメータで指定したタイムウィンドウの登録情報に従って，タイムウィンドウを登録する【NGKI5046】.

domid，twdodr，twdlen，通知方法中の通知処理モードは整数定数式パラメータ，somidはオブジェクト識別名，通知方法中の付随情報は一般定数式パラメータである【NGKI5047】．

twdordは，somidで指定したシステム動作モードにおいて，登録するタイムウィンドウのシステム周期内での順序を示す．同じシステム動作モードに対して登録されたタイムウィンドウは，twdordが小さい順に配置される【NGKI5048】．twdordの値が同じタイムウィンドウが複数ある場合，その間の順序関係は，この仕様では規定しない．

somidで指定したシステム動作モードに対して，登録したタイムウィンドウの合計の長さが，システム周期と同じかそれより長い場合には，E_PARエラーとなる【NGKI5049】．

twdlenには，PRCTIM型に格納できる任意の値を指定できるとは限らず，指定できる値にターゲット定義の上限がある場合がある【NGKI5050】．

通知方法の指定の記述を省略した場合，タイムウィンドウ通知は登録されない【NGKI5097】．

マルチプロセッサ対応カーネルでは，タイムウィンドウは，プロセッサ毎に登録することができる［NGKI0624］．ATT_TWDにより登録するタイムウィンドウは，ATT_TWDの記述を囲みの中に含むクラスのすべての割付け可能プロセッサに対して登録される【NGKI5100】．ただし，タイムウィンドウ通知は，そのクラスの初期割付けプロセッサに対してのみ登録される【NGKI5106】．

システム周期が設定されていない場合，コンフィギュレータは警告メッセージを出力し，この静的APIを無視する【NGKI5052】．

【使用上の注意】

somidには，タイムウィンドウの登録対象となるシステム動作モードの識別名を記述しなければならない．システム動作モードのID番号を表す式を記述することはできない．

【仕様決定の理由】

ATT_TWDのパラメータに割り当てるユーザドメインのID番号を持たせ，保護ドメインの囲みの外に記述する仕様としたのは，タイムウィンドウを登録する静的APIを連続して記述した方が，システムコンフィギュレーションファイルが読みやすいためである．

## 4.13 システム構成管理機能

システム構成管理機能には，保護ドメインのアクセス許可ベクタを設定する機能，非タスクコンテキスト用スタック領域や共有スタック領域を設定する機能，カーネルメモリプール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．

【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，共有スタック領域の設定，カーネルメモリプール領域の設定，終了処理ルーチンの各機能を追加した．

【TOPPERS新世代カーネル統合仕様との関係】

カーネルメモリプール領域の設定機能を追加した．保護ドメインに対する制限を設定する機能（LMT_DOM）は廃止した．

### (1) 保護ドメイン

各保護ドメインが持つ情報は次の通り【NGKI5086】．

- 割り当てられているタイムウィンドウのリスト（ユーザドメインのみ）
- アクセス許可ベクタ

カーネルは，無所属に対して，アクセス許可ベクタの情報を持つ【NGKI5087】．

【TOPPERS新世代カーネル統合仕様との関係】

保護ドメインと無所属に対して，アクセス許可ベクタを持つこととした．

### (2) 非タスクコンテキスト用スタック領域

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，カーネルの管理領域として扱われる【NGKI3199】．

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】．

### (3) カーネルメモリプール領域

カーネルメモリプール領域は，動的生成対応カーネルにおいて，カーネルオブジェクトを生成するサービスコールに対して，オブジェクト生成に必要なメモリ領域の先頭番地のパラメータにNULLが渡された場合に，必要なメモリ領域を獲得するためのメモリ領域である【NGKI5067】．

保護機能対応カーネルでは，各保護ドメイン向けおよび無所属向けのカーネルメモリプール領域を設定することができる【NGKI5156】．ある保護ドメインに属するカーネルオブジェクトの生成には，その保護ドメイン向けのカーネルメモリプール領域が使用され，無所属のカーネルオブジェクトの生成には，無所属向けのカーネルメモリプール領域が使用される【NGKI5157】．

マルチプロセッサ対応カーネルでは，各クラス向けのカーネルメモリプール領域を設定することができる【NGKI5158】．また，クラス共通のカーネルメモリプール領域を設定することができる【NGKI5159】．あるクラスに属するカーネルオブジェクトの生成には，そのクラス向けのカーネルメモ

リプール領域が設定されていればその領域が，設定されていなければ，クラス共通のカーネルメモリプール領域が使用される【NGKI5160】.

保護機能対応カーネルにおいては，各カーネルメモリプール領域は，カーネルの管理領域として扱われる【NGKI5068】.

#### (4) 初期化ルーチン

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ある．初期化ルーチンは，それを識別するための番号を持たない【NGKI5244】.

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】.

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］.

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(EXINF 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】.

#### (5) 終了処理ルーチン

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ある．終了処理ルーチンは，それを識別するための番号を持たない【NGKI5245】.

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】.

マルチプロセッサ対応カーネルでは，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］.

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(EXINF 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】.

| | |
|---|---|
| **ACV_DOM** | 保護ドメインのアクセス許可ベクタの設定〔SP〕【NGKI3752】 |
| **sac_dom** | 保護ドメインのアクセス許可ベクタの設定〔TPD〕【NGKI3753】 |

【静的API】

```
ACV_DOM({ ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
```

【C言語API】

```
ER ercd = sac_dom(ID domid, const ACVCT *p_acvct)
```

【パラメータ】

| | | |
|---|---|---|
| ID | domid | 対象保護ドメインのID番号（sac_domの場合） |
| 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（sac_domの場合） |

＊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<br>・非タスクコンテキストからの呼出し〔s〕【NGKI3754】<br>・CPUロック状態からの呼出し〔s〕【NGKI3755】 |
| E_ID | 不正ID番号<br>・domidが有効範囲外〔s〕【NGKI3756】 |
| E_OACV | オブジェクトアクセス違反<br>・対象保護ドメインに対する管理操作が許可されていない〔s〕【NGKI3759】 |
| E_MACV | メモリアクセス違反<br>・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〕【NGKI3760】 |
| E_OBJ | オブジェクト状態エラー<br>・保護ドメインのアクセス許可ベクタが設定済み〔S〕【NGKI3761】 |

【機能】

ACV_DOMにおいてはそれを囲む保護ドメイン，sac_domにおいてはdomidで指定した保護ドメイン

（対象保護ドメイン）のアクセス許可ベクタ（4つのアク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【NGKI3762】．ACV_DOMを保護ドメインの囲みの外に記述した場合には，無所属に対するアクセス許可ベクタを設定する【NGKI3978】．

静的APIにおいては，acptn1〜acptn4は整数定数式パラメータである【NGKI3763】．

マルチプロセッサ対応カーネルにおいて，ACV_DOMをクラスの囲みの中に記述しても，クラスの情報は使用されない．ACV_DOMをクラスの囲みの中に記述した場合，コンフィギュレータは警告メッセージを出力する【NGKI3758】．

【TOPPERS新世代カーネル統合仕様との関係】

TOPPERS新世代カーネル統合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．

## 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_t istksz, STK_T *istk })
```

※ istkの記述は省略することができる【NGKI3905】.

【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_t | 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<br>・カーネルドメインの囲みの中に記述されていない〔P〕【NGKI3213】<br>・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<br>・条件については機能の項を参照 |
| E_NOMEM | メモリ不足<br>・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<br>・非タスクコンテキスト用スタック領域が設定済み【NGKI3216】<br>・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にターゲット定義の最小値よりも小さい値を指定した時は，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【NGKI3916】．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】.

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】.

〔istkにNULLを指定した場合〕

istkにNULLを指定するか，istkの記述を省略した場合，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】.

保護機能対応カーネルにおいては，コンフィギュレータにより確保されるスタック領域は，カーネルドメイン向け（マルチプロセッサ対応カーネルの場合には，さらに，DEF_ICSの記述をその囲みの中に含むクラス向け）の標準スタックリージョン内に確保される【NGKI5137】.

〔istkにNULL以外を指定した場合〕

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〕

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】．

保護機能対応カーネルにおいては，コンフィギュレータにより確保されるスタック領域は，カーネルドメイン向け（マルチプロセッサ対応カーネルの場合には，さらに，ターゲット定義のクラス向け）の標準スタックリージョン内に確保される【NGKI5138】．

【TOPPERS/SSP3カーネルにおける規定】

SSP3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，SSP3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．istk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．

【TOPPERS新世代カーネル統合仕様との関係】

保護機能対応カーネルにおいて，コンフィギュレータが，スタック領域をどこに確保するかを規定した．

## DEF_STK　共有スタック領域の設定〔S〕【NGKI3228】

【静的API】

```
DEF_STK({ size_t stksz, STK_T *stk })
```

※ stkの記述は省略することができる【NGKI3965】.

【パラメータ】

＊共有スタック領域の設定情報

| | | |
|---|---|---|
| size_t | 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<br>・条件については機能の項を参照 |
| E_NOMEM | メモリ不足<br>・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<br>・共有スタック領域が設定済み【NGKI3992】 |

【サポートするカーネル】

DEF_STKは，TOPPERS/SSP3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にターゲット定義の最小値よりも小さい値を指定した時は，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にNULLを指定するか，stkの記述を省略した場合，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.

## DEF_MPK　カーネルメモリプール領域の設定〔SD〕【NGKI5069】

【静的API】

```
DEF_MPK({ size_t mpksz, MB_T *mpk })
```

※ mpkの記述は省略することができる【NGKI5070】.

【パラメータ】

＊カーネルメモリプール領域の設定情報

| | | |
|---|---|---|
| size_t | mpksz | カーネルメモリプール領域のサイズ（バイト数） |
| MB_T | mpk | カーネルメモリプール領域の先頭番地 |

【エラーコード】

| | |
|---|---|
| E_PAR | パラメータエラー<br>・mpkszが0【NGKI5071】<br>・その他の条件については機能の項を参照 |
| E_NOMEM | メモリ不足<br>・カーネルメモリプール領域が確保できない【NGKI5072】 |
| E_OBJ | オブジェクト状態エラー<br>・カーネルメモリプール領域が設定済み【NGKI5073】<br>・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カーネルメモリプール領域の設定情報に従って，カーネルメモリプール領域を設定する【NGKI5074】.

mpkszは整数定数式パラメータ，mpkは一般定数式パラメータである【NGKI5075】．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【NGKI5076】.

保護機能対応カーネルでは，DEF_MPKを保護ドメインの囲みの中に記述すると，その保護ドメイン向けのカーネルメモリプール領域を設定し，保護ドメインの囲みの外に記述すると，無所属向けのカーネルメモリプール領域を設定する【NGKI5162】.

マルチプロセッサ対応カーネルでは，DEF_MPKをクラスの囲みの中に記述すると，そのクラス向けのカーネルメモリプール領域を設定し，クラスの囲みの外に記述すると，クラス共通のカーネルメモリプール領域を設定する【NGKI5163】.

〔mpkにNULLを指定した場合〕

mpkにNULLを指定するか，mpkの記述を省略した場合，mpkszで指定したサイズのメモリ領域が，コンフィギュレータにより確保される【NGKI5077】．mpksz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【NGKI5078】.

保護機能対応カーネルにおいては，コンフィギュレータにより確保されるカーネルメモリプール領域は，カーネルドメイン向け（マルチプロセッサ対応カーネルの場合には，さらに，DEF_MPKの記述をその囲みの中に含むクラス向け）の標準RAMリージョン内に確保される【NGKI5201】.

〔mpkにNULL以外を指定した場合〕

mpkにNULL以外を指定した場合，mpkとmpkszで指定したカーネル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【NGKI5079】．カーネルメモリプール領域をアプリケーションで確保するために，次のマクロを用意している【NGKI5080】．

| | |
|---|---|
| COUNT_MB_T(sz) | サイズszのカーネルメモリプール領域を確保するために必要なMB_T型の配列の要素数 |
| ROUND_MB_T(sz) | 要素数COUNT_MB_T(sz)のMB_T型の配列のサイズ（szを，MB_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，サイズszのカーネルメモリプール領域を確保する方法は次の通り【NGKI5081】．

```
MB_T    <カーネルメモリプール領域の変数名>[COUNT_MB_T(sz)];
```

この時，mpkszにはROUND_MB_T(sz)，mpkには<カーネルメモリプール領域の変数名>を指定する【NGKI5082】．

この方法に従わず，mpkやmpksz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【NGKI5083】．また，保護機能対応カーネルでは，mpkとmpkszで指定したカーネルメモリプール領域が，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【NGKI5084】．

【μITRON4.0仕様，TOPPERS新世代カーネル統合仕様との関係】

μITRON4.0仕様およびTOPPERS新世代カーネル統合仕様に定義されていない静的APIである．

## ATT_INI　　初期化ルーチンの登録〔S〕【NGKI3240】

【静的API】

```
ATT_INI({ ATR iniatr, EXINF exinf, INIRTN inirtn })
```

【パラメータ】

＊初期化ルーチンの登録情報

| | | |
|---|---|---|
| ATR | iniatr | 初期化ルーチン属性 |
| EXINF | 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<br>・iniatrが無効【NGKI3241】<br>・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<br>・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の登録情報に従って，初期化ルーチンを登録する【NGKI3244】.

マルチプロセッサ対応カーネルでは，ATT_INIをクラスの囲みの外に記述すると，クラスに属さないグローバル初期化ルーチンを定義し，クラスの囲みの中に記述すると，そのクラスに属するローカル初期化ルーチンを定義する【NGKI5242】.

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】.

## ATT_TER　終了処理ルーチンの登録〔S〕【NGKI3247】

【静的API】

```
ATT_TER({ ATR teratr, EXINF exinf, TERRTN terrtn })
```

【パラメータ】

＊終了処理ルーチンの登録情報

| | | |
|---|---|---|
| ATR | teratr | 終了処理ルーチン属性 |
| EXINF | 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<br>・teratrが無効【NGKI3248】<br>・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<br>・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の登録情報に従って，終了処理ルーチンを登録する【NGKI3251】.

マルチプロセッサ対応カーネルでは，ATT_TERをクラスの囲みの外に記述すると，クラスに属さないグローバル終了処理ルーチンを定義し，クラスの囲みの中に記述すると，そのクラスに属するローカル終了処理ルーチンを定義する【NGKI5243】.

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】.

【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/ASP3カーネルにおける規定】

ASP3カーネルでは，ref_cfgをサポートしない．

【TOPPERS/FMP3カーネルにおける規定】

FMP3カーネルでは，ref_cfgをサポートしない．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルでは，ref_cfgをサポートしない．

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルでは，ref_cfgをサポートしない．

【TOPPERS/SSP3カーネルにおける規定】

SSP3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/ASP3カーネルにおける規定】

ASP3カーネルでは，ref_verをサポートしない．

【TOPPERS/FMP3カーネルにおける規定】

FMP3カーネルでは，ref_verをサポートしない．

【TOPPERS/HRP3カーネルにおける規定】

HRP3カーネルでは，ref_verをサポートしない．

【TOPPERS/HRMP3カーネルにおける規定】

HRMP3カーネルでは，ref_verをサポートしない．

【TOPPERS/SSP3カーネルにおける規定】

SSP3カーネルでは，ref_verをサポートしない．

# 第５章　リファレンス

## 5.1 サービスコール一覧

(1) タスク管理機能

```
ER_ID tskid = acre_tsk(const T_CTSK *pk_ctsk)                〔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)                                  〔TI〕
ER ercd = mact_tsk(ID tskid, ID prcid)                       〔TIM〕
ER_UINT actcnt = can_act(ID tskid)                           〔T〕
ER ercd = mig_tsk(ID tskid, ID prcid)                        〔TM〕
ER ercd = get_tst(ID tskid, STAT *p_tskstat)                 〔T〕
ER ercd = chg_pri(ID tskid, PRI tskpri)                      〔T〕
ER ercd = get_pri(ID tskid, PRI *p_tskpri)                   〔T〕
ER ercd = chg_spr(ID tskid, uint_t subpri)                   〔T〕
ER ercd = get_inf(EXINF *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)                                  〔TI〕
ER_UINT wupcnt = can_wup(ID tskid)                           〔T〕
ER ercd = rel_wai(ID tskid)                                  〔TI〕
ER ercd = sus_tsk(ID tskid)                                  〔T〕
ER ercd = rsm_tsk(ID tskid)                                  〔T〕
ER ercd = dly_tsk(RELTIM dlytim)                             〔T〕
```

(3) タスク終了機能

```
ER ercd = ext_tsk()                                          〔T〕
ER ercd = ras_ter(ID tskid)                                  〔T〕
ER ercd = dis_ter()                                          〔T〕
ER ercd = ena_ter()                                          〔T〕
bool_t state = sns_ter()                                     〔TI〕
ER ercd = ter_tsk(ID tskid)                                  〔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)                                      〔TI〕
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)                       〔TI〕
ER ercd = clr_flg(ID flgid, FLGPTN clrptn)                       〔T〕
ER ercd = wai_flg(ID flgid, FLGPTN waiptn,                       〔T〕
                                MODE wfmode, FLGPTN *p_flgptn)
ER ercd = pol_flg(ID flgid, FLGPTN waiptn,                       〔T〕
                                MODE wfmode, FLGPTN *p_flgptn)
ER ercd = twai_flg(ID flgid, FLGPTN waiptn,                      〔T〕
                    MODE wfmode, FLGPTN *p_flgptn, TMO tmout)
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)                      〔TI〕
ER ercd = tsnd_dtq(ID dtqid, intptr_t data, TMO tmout)           〔T〕
ER ercd = fsnd_dtq(ID dtqid, intptr_t data)                      〔TI〕
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)         〔TI〕
ER ercd = tsnd_pdq(ID pdqid, intptr_t data,                      〔T〕
                                      PRI datapri, TMO tmout)
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,                   〔T〕
                                    PRI *p_datapri, TMO tmout)
ER ercd = ini_pdq(ID pdqid)                                      〔T〕
ER ercd = ref_pdq(ID pdqid, T_RPDQ *pk_rpdq)                     〔T〕
```

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,                    〔T〕
                                    uint_t msgsz, TMO tmout)
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)                                      〔TIM〕
ER ercd = try_spn(ID spnid)                                      〔TIM〕
ER ercd = unl_spn(ID spnid)                                      〔TIM〕
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 = set_tim(SYSTIM systim)                                 〔T〕
ER ercd = get_tim(SYSTIM *p_systim)                              〔T〕
ER ercd = adj_tim(int32_t adjtim)                                〔TI〕
ER ercd = set_dft(int32_t drift)                                 〔T〕
HRTCNT hrtcnt = fch_hrt()                                        〔TI〕
```

周期通知

```
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〕
```

アラーム通知

```
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)                       〔TI〕
ER ercd = msta_alm(ID almid, RELTIM almtim, ID prcid)            〔TIM〕
ER ercd = stp_alm(ID almid)                                      〔TI〕
ER ercd = ref_alm(ID almid, T_RALM *pk_ralm)                     〔T〕
```

オーバランハンドラ

```
ER ercd = sta_ovr(ID tskid, PRCTIM ovrtim)                       〔TI〕
ER ercd = stp_ovr(ID tskid)                                      〔TI〕
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)                                    〔TI〕
ER ercd = mrot_rdq(ID schedno, PRI tskpri)                       〔TIP | TIM〕
ER ercd = get_tid(ID *p_tskid)                                   〔TI〕
ER ercd = get_did(ID *p_domid)                                   〔TIP〕
ER ercd = get_pid(ID *p_prcid)                                   〔TIM〕
ER ercd = get_lod(PRI tskpri, uint_t *p_load)                    〔T〕
ER ercd = mget_lod(ID schedno, PRI tskpri, uint_t *p_load)       〔TP | TM〕
ER ercd = get_nth(PRI tskpri, uint_t nth, ID *p_tskid)           〔T〕
ER ercd = mget_nth(ID schedno, PRI tskpri, uint_t nth, ID *p_tskid)
                                                                 〔TP | TM〕
ER ercd = loc_cpu()                                              〔TI〕
ER ercd = unl_cpu()                                              〔TI〕
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_t size,                 〔TP〕
                                    ID tskid, MODE pmmode)
ER ercd = ref_mem(const void *addr, 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 = dis_int(INTNO intno)                                   〔TI〕
ER ercd = ena_int(INTNO intno)                                   〔TI〕
ER ercd = clr_int(INTNO intno)                                   〔TI〕
ER ercd = ras_int(INTNO intno)                                   〔TI〕
ER_BOOL state = prb_int(INTNO intno)                             〔TI〕
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

```
bool_t state = xsns_dpn(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,    〔TIP〕
                    intptr_t par3, intptr_t par4, intptr_t par5)
```

(12) 時間パーティショニング機能

```
ER ercd = chg_som(ID somid)                                      〔TP〕
ER ercd = get_som(ID *p_somid)                                   〔TP〕
```

(13) システム構成管理機能

```
ER ercd = sac_dom(ID domid, const ACVCT *p_acvct)                〔TPD〕
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, EXINF exinf, TASK task,          〔S〕
                        PRI itskpri, size_t stksz, STK_T *stk })
```

＊保護機能対応カーネルの場合

```
CRE_TSK(ID tskid, { ATR tskatr, EXINF exinf, TASK task,          〔SP〕
                        PRI itskpri, size_t stksz, STK_T *stk,
                        size_t sstksz, STK_T *sstk })
```

```
AID_TSK(uint_t notsk)                                            〔SD〕
SAC_TSK(ID tskid, { ACPTN acptn1, ACPTN acptn2,                  〔SP〕
                                ACPTN acptn3, ACPTN acptn4 })
DEF_EPR(ID tskid, { PRI exepri })                                〔S〕
```

(2) タスク付属同期機能

```
なし
```

(3) タスク終了機能

```
なし
```

(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,                  〔SP〕
                                ACPTN acptn3, ACPTN acptn4 })
```

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,                  〔SP〕
                                ACPTN acptn3, ACPTN acptn4 })
```

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,                  〔SP〕
                                ACPTN acptn3, ACPTN acptn4 })
```

優先度データキュー

```
CRE_PDQ(ID pdqid, { ATR pdqatr, uint_t pdqcnt,                   〔S〕
                                    PRI maxdpri, void *pdqmb })
AID_PDQ(uint_t nopdq)                                            〔SD〕
SAC_PDQ(ID pdqid, { ACPTN acptn1, ACPTN acptn2,                  〔SP〕
                                ACPTN acptn3, ACPTN acptn4 })
```

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,                  〔SP〕
                                ACPTN acptn3, ACPTN acptn4 })
```

メッセージバッファ

```
CRE_MBF(ID mbfid, { ATR mbfatr, uint_t maxmsz,
                                size_t mbfsz, void *mbfmb })     〔S〕
AID_MBF(uint_t nombf)                                            〔SD〕
SAC_MBF(ID mbfid, { ACPTN acptn1, ACPTN acptn2,                  〔SP〕
                                ACPTN acptn3, ACPTN acptn4 })
```

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,                  〔SPM〕
                                ACPTN acptn3, ACPTN acptn4 })
```

(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,     〔S〕
                                    MPF_T *mpf, void *mpfmb })
AID_MPF(uint_t nompf)                                            〔SD〕
SAC_MPF(ID mpfid, { ACPTN acptn1, ACPTN acptn2,                  〔SP〕
                                ACPTN acptn3, ACPTN acptn4 })
```

(6) 時間管理機能

周期通知

```
CRE_CYC(ID cycid, { ATR cycatr, ＜通知方法の指定＞,               〔S〕
                                RELTIM cyctim, RELTIM cycphs })
AID_CYC(uint_t nocyc)                                            〔SD〕
SAC_CYC(ID cycid, { ACPTN acptn1, ACPTN acptn2,                  〔SP〕
                                ACPTN acptn3, ACPTN acptn4 })
```

アラーム通知

```
CRE_ALM(ID almid, { ATR almatr, ＜通知方法の指定＞ })            〔S〕
AID_ALM(uint_t noalm)                                            〔SD〕
SAC_ALM(ID almid, { ACPTN acptn1, ACPTN acptn2,                  〔SP〕
                                ACPTN acptn3, ACPTN acptn4 })
```

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,                            〔SP〕
                                ACPTN acptn3, ACPTN acptn4 })
ENA_SPR(PRI tskpri)                                              〔S〕
```

(8) メモリオブジェクト管理機能

```
ATT_REG(char *regname, { ATR regatr, void *base, size_t size })  〔SP〕
DEF_SRG(char *stdrom, char *stdram,                              〔SP〕
                    char *stdsrom, char *stdsram, char *stdstk)
ATT_SEC(char *secname, { ATR rematr, char *regname },            〔SP〕
    { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
ATT_MOD(char *modname,                                           〔SP〕
    { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
ATT_MEM({ ATR accatr, void *base, size_t size },                 〔SP〕
    { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
ATT_PMA({ ATR accatr, void *base, size_t size, void *paddr },    〔SP〕
    { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
```

(9) 割込み管理機能

```
CFG_INT(INTNO intno, { ATR intatr, PRI intpri })                〔S〕
CRE_ISR(ID isrid, { ATR isratr, EXINF exinf,                    〔S〕
                            INTNO intno, ISR isr, PRI isrpri })
AID_ISR(uint_t noisr)                                           〔SD〕
SAC_ISR(ID isrid, { ACPTN acptn1, ACPTN acptn2,                 〔SP〕
                                ACPTN acptn3, ACPTN acptn4 })
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 extsvc, size_t stksz })   〔SP〕
```

(12) 時間パーティショニング機能

```
DEF_SCY({ RELTIM scyctim })                                     〔SP〕
CRE_SOM(ID somid, { ATR somatr, ID nxtsom })                    〔SP〕
ATT_TWD({ ID domid, ID somid, int_t twdord, PRCTIM twdlen,      〔SP〕
                                        <通知方法の指定> })
```

(13) システム構成管理機能

```
ACV_DOM({ ACPTN acptn1, ACPTN acptn2,                           〔SP〕
                                ACPTN acptn3, ACPTN acptn4 })
DEF_ICS({ size_t istksz, STK_T *istk })                         〔S〕
DEF_STK({ size_t stksz, STK_T *stk })                           〔S〕
DEF_MPK({ size_t mpksz, MB_T *mpk })                            〔SD〕
ATT_INI({ ATR iniatr, EXINF exinf, INIRTN inirtn })             〔S〕
ATT_TER({ ATR teratr, EXINF 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 |
| size_t | メモリ領域のサイズ（符号無し整数）（C90準拠） |
| 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） |
| TMO | タイムアウト指定（符号無し32ビット整数，単位はマイクロ秒，uint32_tに定義） |
| EXINF | 拡張情報（ポインタまたは32ビット以上のサイズの整数を格納できるサイズの型） |
| RELTIM | 相対時間（符号無し32ビット整数，単位はマイクロ秒，uint32_tに定義） |
| SYSTIM | システム時刻（符号無し整数，単位はマイクロ秒，uint64_tまたはuint32_tに定義） |
| PRCTIM | プロセッサ時間（符号無し整数，単位はマイクロ秒，uint32_tに定義） |
| HRTCNT | 高分解能タイマのカウント値（符号無し32ビット整数，uint32_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

| | |
|---|---|
| FLGPTN | イベントフラグのビットパターン（符号無し整数，uint_tに定義） |
| INTNO | 割込み番号（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） |
| TMEHDR | タイムイベントハンドラ（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 |

### 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 */
    EXINF       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_t      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_t      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【NGKI4039】

```
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 */
    /* 次は，サブ優先度機能をサポートするカーネルの場合 */
    uint_t      subpri;     /* タスクのサブ優先度 */
    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 */
    bool_t      raster;     /* タスク終了要求状態 */
    bool_t      dister;     /* タスク終了禁止状態 */
    /* 次は，保護機能対応カーネルの場合 */
    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      /* 次回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) タスク終了機能

```
なし
```

(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;
```

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_t      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_t  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

タイムイベントの通知方法のパケット形式［NGKI3722］

```
typedef struct {
    EXINF       exinf;      /* タイムイベントハンドラの拡張情報 */
```

```
    TMEHDR      tmehdr;     /* タイムイベントハンドラの先頭番地 */
} T_NFY_HDR;

typedef struct {
    intptr_t    *p_var;     /* 変数の番地 */
    intptr_t    value;      /* 設定する値 */
} T_NFY_VAR;

typedef struct {
    intptr_t    *p_var;     /* 変数の番地 */
} T_NFY_IVAR;

typedef struct {
    ID          tskid;      /* タスクID */
} T_NFY_TSK;

typedef struct {
    ID          semid;      /* セマフォID */
} T_NFY_SEM;

typedef struct {
    ID          flgid;      /* イベントフラグID */
    FLGPTN      flgptn;     /* セットするビットパターン */
} T_NFY_FLG;

typedef struct {
    ID          dtqid;      /* データキューID */
    intptr_t    data;       /* 送信する値 */
} T_NFY_DTQ;

typedef struct {
    intptr_t    *p_var;     /* 変数の番地 */
} T_ENFY_VAR;

typedef struct {
    ID          dtqid;      /* データキューID */
} T_ENFY_DTQ;

typedef struct {
    MODE    nfymode;        /* 通知処理モード */
    union {                 /* タイムイベントの通知に関する付随情報 */
        T_NFY_HDR   handler;    /* タイムイベントハンドラの呼出しの場合 */
        T_NFY_VAR   setvar;     /* 変数の設定の場合 */
        T_NFY_IVAR  incvar;     /* 変数のインクリメントの場合 */
        T_NFY_TSK   acttsk;     /* タスクの起動の場合 */
        T_NFY_TSK   wuptsk;     /* タスクの起床の場合 */
        T_NFY_SEM   sigsem;     /* セマフォの資源の返却の場合 */
        T_NFY_FLG   setflg;     /* イベントフラグのセットの場合 */
        T_NFY_DTQ   snddtq;     /* データキューへの送信の場合 */
    } nfy;
    union {                 /* エラーの通知に関する付随情報 */
```

```
        T_ENFY_VAR  setvar;     /* 変数の設定の場合 */
        T_NFY_IVAR  incvar;     /* 変数のインクリメントの場合 */
        T_NFY_TSK   acttsk;     /* タスクの起動の場合 */
        T_NFY_TSK   wuptsk;     /* タスクの起床の場合 */
        T_NFY_SEM   sigsem;     /* セマフォの資源の返却の場合 */
        T_NFY_FLG   setflg;     /* イベントフラグのセットの場合 */
        T_ENFY_DTQ  snddtq;     /* データキューへの送信の場合 */
    } enfy;
} T_NFYINFO;
```

周期通知の生成情報のパケット形式【NGKI4041】

```
typedef struct t_ccyc {
    ATR         cycatr;     /* 周期通知属性 */
    T_NFYINFO   nfyinfo;    /* 周期通知の通知方法 */
    RELTIM      cyctim;     /* 周期通知の通知周期 */
    RELTIM      cycphs;     /* 周期通知の通知位相 */
} T_CCYC;
```

周期通知の現在状態のパケット形式【NGKI4024】

```
typedef struct t_rcyc {
    STAT        cycstat;    /* 周期通知の動作状態 */
    RELTIM      lefttim;    /* 次回通知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;
```

アラーム通知の生成情報のパケット形式【NGKI4042】

```
typedef struct t_calm {
    ATR         almatr;     /* アラーム通知属性 */
    T_NFYINFO   nfyinfo;    /* アラーム通知の通知方法 */
} T_CALM;
```

アラーム通知の現在状態のパケット形式【NGKI4026】

```
typedef struct t_ralm {
    STAT        almstat;    /* アラーム通知の動作状態 */
    RELTIM      lefttim;    /* 通知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 */
    PRCTIM  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         accatr;     /* メモリオブジェクトアクセス属性 */
    void *      base;       /* 登録するメモリ領域の先頭番地 */
    size_t      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         accatr;     /* メモリオブジェクトアクセス属性 */
    void *      base;       /* 登録するメモリ領域の先頭番地 */
    size_t      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【NGKI4044】

```
typedef struct t_rmem {
    ATR         accatr;     /* メモリオブジェクトアクセス属性 */
    void *      base;       /* メモリオブジェクトの先頭番地 */
    size_t      size;       /* メモリオブジェクトのサイズ（バイト数）*/
} T_RMEM;
```

### (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 */
    EXINF       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

```
    なし
```

#### (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      extsvc;     /* 拡張サービスコールの先頭番地 */
    size_t      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) 時間パーティショニング機能

```
    なし
```

#### (13) 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) 整数型と浮動小数点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） |
| SIZE_MAX | size_tに格納できる最大値（C99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 | 0U | ポーリング |
| TMO_FEVR | UINT32_MAX | 永久待ち |
| TMO_NBLK | UINT32_MAX-1 | 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構成マクロ

| | |
|---|---|
| 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 |

### 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 | 呼び出した処理単位が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

| | | |
|---|---|---|
| TSOM_STP | -1 | システム周期の停止 |

| | | |
|---|---|---|
| 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 |

#### (3) スケジューリング単位番号を構成するマクロ

| | |
|---|---|
| SCHEDNO(prcid, domid) | マルチプロセッサ対応カーネルにおけるスケジューリング単位番号prcidと，保護機能対応カーネルにおけるスケジューリング単位番号domidから，保護機能とマルチプロセッサの両方に対応するカーネルにおけるスケジューリング単位番号を構成するためのマクロ |

### 5.4.5 カーネルの機能毎の定数

#### (1) タスク管理機能

| | | |
|---|---|---|
| TA_ACT | 0x01U | タスクの生成時にタスクを起動する |
| TA_NOACTQUE | 0x02U | タスクに対する起動要求をキューイングしない |
| 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 |

TA_FPUの値は，ターゲット定義とする【NGKI4045】.

| | | |
|---|---|---|
| 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_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 |

### (2) タスク付属同期機能

なし

### (3) タスク終了機能

なし

### (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 |

ミューテックス

| | | |
|---|---|---|
| TA_INHERIT | 0x02U | 優先度継承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_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 |

スピンロック

| | | |
|---|---|---|
| TA_NATIVE | 0x01U | エミュレーションによって実現してはならない |

| | | |
|---|---|---|
| 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 |

(5) メモリプール管理機能

| |
|---|
| なし |

(6) 時間管理機能

| | | |
|---|---|---|
| TNFY_HANDLER | 0x00U | タイムイベントハンドラの呼出しによる通知 |
| TNFY_SETVAR | 0x01U | 変数の設定による通知 |
| TNFY_INCVAR | 0x02U | 変数のインクリメントによる通知 |
| TNFY_ACTTSK | 0x03U | タスクの起動による通知 |
| TNFY_WUPTSK | 0x04U | タスクの起床による通知 |
| TNFY_SIGSEM | 0x05U | セマフォの返却による通知 |
| TNFY_SETFLG | 0x06U | イベントフラグのセットによる通知 |
| TNFY_SNDDTQ | 0x07U | データキューへの送信による通知 |

| | | |
|---|---|---|
| TENFY_SETVAR | 0x10U | 変数の設定による通知 |
| TENFY_INCVAR | 0x20U | 変数のインクリメントによる通知 |
| TENFY_ACTTSK | 0x30U | タスクの起動による通知 |
| TENFY_WUPTSK | 0x40U | タスクの起床による通知 |
| TENFY_SIGSEM | 0x50U | セマフォの返却による通知 |
| TENFY_SETFLG | 0x60U | イベントフラグのセットによる通知 |
| TENFY_SNDDTQ | 0x70U | データキューへの送信による通知 |

周期通知

| | | |
|---|---|---|
| TA_STA | 0x02U | 周期通知の生成時に周期通知を動作開始する |
| TA_PHS | 0x04U | 周期通知を生成した時刻を基準時刻とする |

| | | |
|---|---|---|
| TCYC_STP | 0x01U | 周期通知が動作していない状態 |
| TCYC_STA | 0x02U | 周期通知が動作している状態 |

アラーム通知

| | | |
|---|---|---|
| TALM_STP | 0x01U | アラーム通知が動作していない状態 |
| TALM_STA | 0x02U | アラーム通知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 |

(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

| | | |
|---|---|---|
| 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_MEMZERO | 0x10U | メモリのクリアを行う |
| 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 |

TA_WTHROUGHの値は，ターゲット定義とする【NGKI4046】.

| | | |
|---|---|---|
| TA_ATTMEM | 0x1000U | ATT_MEM／att_mem／ATT_PMA／att_pmaで登録されたメモリオブジェクト |
| TA_USTACK | 0x2000U | ユーザスタック領域 |

| | | |
|---|---|---|
| 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 |

標準のメモリオブジェクト属性

| | |
|---|---|
| TA_TEXTSEC | テキストセクション（textセクション）に設定する標準的なメモリオブジェクト属性 |
| TA_RODATASEC | リードオンリーデータセクション（rodataセクション）に設定する標準的なメモリオブジェクト属性 |
| TA_DATASEC | 初期化データセクション（dataセクション）に設定する標準的なメモリオブジェクト属性 |
| TA_BSSSEC | ゼロ初期化データセクション（bssセクション）に設定する標準的なメモリオブジェクト属性 |
| TA_NOINITSEC | 非初期化データセクションに設定する標準的なメモリオブジェクト属性 |
| TA_LOSEC | 配置のみを行うセクションに設定する標準的なメモリオブジェクト属性 |

(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_POSEDGE，TA_NEGEDGE，TA_BOTHEDGE，TA_LOWLEVEL，TA_HIGHLEVELの値は，ターゲット定義とする【NGKI4047】.

| | | |
|---|---|---|
| TA_NONKERNEL | 0x02U | カーネル管理外の割込み |

(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【NGKI4048】.

(11) 拡張サービスコール管理機能

なし

(12) 時間パーティショニング機能

なし

(13) システム構成管理機能

なし

### 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） |

(2) タスク付属同期機能

なし

(3) タスク終了機能

なし

(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で指定したサイズのメッセージを，msgcntで指定した数だけ格納できるメッセージバッファ管理領域のサイズ（バイト数） |
| TCNT_MBFMB(msgcnt, msgsz) | msgszで指定したサイズのメッセージを，msgcntで指定した数だけ格納できるメッセージバッファ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 |

(5) メモリプール管理機能

| | |
|---|---|
| COUNT_MPF_T(blksz) | 固定長メモリブロックのサイズがblkszの固定長メモリプール領域を確保するために，固定長メモリブロック1つあたりに必要なMPF_T型の配列の要素数を求めるマクロ |
| ROUND_MPF_T(blksz) | 要素数COUNT_MPF_T(blksz)のMPF_T型の配列のサイズ（blkszを，MPF_T型のサイズの倍数になるように大きい方に丸めた値） |
| TSZ_MPFMB(blkcnt) | blkcntで指定した数の固定長メモリブロックを管理することができる固定長メモリプール管理領域のサイズ（バイト数） |
| TCNT_MPFMB(blkcnt) | blkcntで指定した数の固定長メモリブロックを管理することができる固定長メモリプール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 |

(6) 時間管理機能

なし

(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

なし

(9) 割込み管理機能

なし

(10) CPU例外管理機能

なし

(11) 拡張サービスコール管理機能

なし

(12) 時間パーティショニング機能

なし

(13) システム構成管理機能

| | |
|---|---|
| COUNT_MB_T(sz) | サイズszのカーネルメモリプール領域を確保するために必要なMB_T型の配列の要素数 |
| ROUND_MB_T(sz) | 要素数COUNT_MB_T(sz)のMB_T型の配列のサイズ（szを，MB_T型のサイズの倍数になるように大きい方に丸めた値） |

## 5.5 構成マクロ

### 5.5.1 TOPPERS共通構成マクロ

(1) 相対時間の範囲

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

### 5.5.2 カーネル共通構成マクロ

(1) サポートする機能

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

(2) 優先度の範囲

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

(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 |

(5) メモリ配置の方法

| | |
|---|---|
| TOPPERS_ML_AUTO | 自動メモリ配置の場合にマクロ定義 |
| TOPPERS_ML_MANUAL | 手動メモリ配置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) タスク終了機能

| |
|---|
| なし |

(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） |

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

| | |
|---|---|
| TMIN_ADJTIM | システム時刻の調整時間の最小値（＝-1,000,000） |
| TMAX_ADJTIM | システム時刻の調整時間の最大値（＝1,000,000） |

| | |
|---|---|
| TMIN_DRIFT | ドリフト量の最小値（＝-100,000） |
| TMAX_DRIFT | ドリフト量の最大値（＝100,000） |

| | |
|---|---|
| TCYC_HRTCNT | 高分解能タイマのタイマ周期 |
| TSTEP_HRTCNT | 高分解能タイマのカウント値の進み幅 |

周期通知

| | |
|---|---|
| TNUM_CYCID | 登録できる周期通知の数（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通知の数に一致） |

アラーム通知

| | |
|---|---|
| TNUM_ALMID | 登録できるアラーム通知の数（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通知の数に一致） |

オーバランハンドラ

| | |
|---|---|
| TMAX_OVRTIM | オーバランハンドラの残り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がサポートされている |
| TOPPERS_SUPPORT_ATT_PMA | ATT_PMA／att_pmaがサポートされている |

(9) 割込み管理機能

| | |
|---|---|
| TMIN_INTPRI | カーネル管理の割込み優先度の最小値（最高値） |
| TMAX_INTPRI | カーネル管理の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 |
| TOPPERS_SUPPORT_CLR_INT | clr_intがサポートされている |
| TOPPERS_SUPPORT_RAS_INT | ras_intがサポートされている |
| TOPPERS_SUPPORT_PRB_INT | prb_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

| | |
|---|---|
| TMAX_FNCD | 拡張サービスコールの機能コードの最大値（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） |

(12) 時間パーティショニング機能

| | |
|---|---|
| TMAX_TWDTIM | タイムウィンドウの長さに指定できる最大値 |

(13) 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_RASTER | -53 | タスクの終了要求 |
| | 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 |
| | E_COMM | -65 | 通信エラー |

## 5.7 機能コード一覧【NGKI4040】

| 機能コード | -0 | -1 | -2 | -3 |
|---|---|---|---|---|
| -0x01 | 予約 | 予約 | 予約 | 予約 |
| -0x05 | act_tsk | get_tst | can_act | ext_tsk |
| -0x09 | ter_tsk | chg_pri | get_pri | get_inf |
| -0x0d | slp_tsk | tslp_tsk | wup_tsk | 予約 |
| -0x11 | can_wup | rel_wai | 予約 | 予約 |
| -0x15 | 予約 | 予約 | 予約 | 予約 |

| | | | | |
|---|---|---|---|---|
| -0x19 | sus_tsk | rsm_tsk | dly_tsk | 予約 |
| -0x1d | ras_ter | 予約 | dis_ter | ena_ter |
| -0x21 | sns_ter | 予約 | 予約 | 予約 |
| -0x25 | sig_sem | 予約 | wai_sem | pol_sem |
| -0x29 | twai_sem | 予約 | 予約 | 予約 |
| -0x2d | set_flg | 予約 | clr_flg | wai_flg |
| -0x31 | pol_flg | twai_flg | 予約 | 予約 |
| -0x35 | snd_dtq | psnd_dtq | 予約 | tsnd_dtq |
| -0x39 | fsnd_dtq | 予約 | rcv_dtq | prcv_dtq |
| -0x3d | trcv_dtq | 予約 | 予約 | 予約 |
| -0x41 | snd_pdq | psnd_pdq | 予約 | tsnd_pdq |
| -0x45 | rcv_pdq | prcv_pdq | trcv_pdq | 予約 |
| -0x49 | set_tim | get_tim | adj_tim | set_dft |
| -0x4d | loc_mtx | ploc_mtx | tloc_mtx | unl_mtx |
| -0x51 | snd_mbf | psnd_mbf | tsnd_mbf | rcv_mbf |
| -0x55 | prcv_mbf | trcv_mbf | 予約 | 予約 |
| -0x59 | get_mpf | pget_mpf | tget_mpf | rel_mpf |
| -0x5d | 予約 | fch_hrt | 予約 | ref_ovr |
| -0x61 | sta_cyc | stp_cyc | 予約 | 予約 |
| -0x65 | sta_alm | 予約 | stp_alm | 予約 |
| -0x69 | sta_ovr | 予約 | stp_ovr | 予約 |
| -0x6d | sac_sys | ref_sys | rot_rdq | get_lod |
| -0x71 | get_did | 予約 | get_tid | get_nth |
| -0x75 | loc_cpu | 予約 | unl_cpu | 予約 |
| -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 | 予約 | att_pma | 予約 |
| -0x89 | cfg_int | dis_int | ena_int | ref_int |
| -0x8d | chg_ipm | get_ipm | 予約 | 予約 |
| -0x91 | xsns_dpn | clr_int | ras_int | prb_int |
| -0x95 | ref_cfg | ref_ver | 予約 | 予約 |
| -0x99 | sac_dom | 予約 | 予約 | 予約 |
| -0x9d | chg_som | 予約 | get_som | 予約 |
| -0xa1 | 予約 | ini_sem | ini_flg | ini_dtq |
| -0xa5 | ini_pdq | 予約 | ini_mtx | ini_mbf |

| | | | | |
|---|---|---|---|---|
| -0xa9 | ini_mpf | 予約 | 予約 | 予約 |
| -0xad | 予約 | 予約 | 予約 | 予約 |
| -0xb1 | ref_tsk | ref_sem | ref_flg | ref_dtq |
| -0xb5 | ref_pdq | 予約 | ref_mtx | ref_mbf |
| -0xb9 | ref_mpf | ref_cyc | ref_alm | ref_isr |
| -0xbd | ref_spn | 予約 | 予約 | 予約 |
| -0xc1 | acre_tsk | acre_sem | acre_flg | acre_dtq |
| -0xc5 | acre_pdq | 予約 | acre_mtx | acre_mbf |
| -0xc9 | acre_mpf | acre_cyc | acre_alm | acre_isr |
| -0xcd | acre_spn | 予約 | 予約 | 予約 |
| -0xd1 | del_tsk | del_sem | del_flg | del_dtq |
| -0xd5 | del_pdq | 予約 | del_mtx | del_mbf |
| -0xd9 | del_mpf | del_cyc | del_alm | del_isr |
| -0xdd | del_spn | 予約 | 予約 | 予約 |
| -0xe1 | sac_tsk | sac_sem | sac_flg | sac_dtq |
| -0xe5 | sac_pdq | 予約 | sac_mtx | sac_mbf |
| -0xe9 | sac_mpf | sac_cyc | sac_alm | sac_isr |
| -0xed | sac_spn | 予約 | 予約 | 予約 |
| -0xf1 | 予約 | 予約 | 予約 | 予約 |
| -0xf5 | def_svc | 予約 | 予約 | 予約 |
| -0xf9 | 予約 | 予約 | 予約 | 予約 |
| -0xfd | 予約 | 予約 | 予約 | 予約 |
| -0x101 | mact_tsk | 予約 | mig_tsk | chg_spr |
| -0x105 | msta_cyc | 予約 | msta_alm | 予約 |
| -0x109 | mrot_rdq | mget_lod | get_pid | mget_nth |
| -0x10d | 予約 | 予約 | 予約 | 予約 |
| -0x111 | loc_spn | 予約 | try_spn | 予約 |
| -0x115 | unl_spn | 予約 | 予約 | 予約 |
| -0x119 | 予約 | 予約 | 予約 | 予約 |
| -0x11d | 予約 | 予約 | 予約 | 予約 |

【μITRON4.0仕様との関係】

サービスコールの機能コードを割り当てなおした．

【TOPPERS新世代カーネル統合仕様との関係】

サービスコールの増減に対応した．一部，サービスコールの機能コードを割り当てなおした．

## 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<br>実行 | det_mem<br>sac_mem | ref_mem<br>prb_mem |
| タスク | act_tsk<br>mact_tsk<br>can_act<br>mig_tsk<br>wup_tsk<br>can_wup | chg_pri<br>chg_spr<br>rel_wai<br>sus_tsk<br>rsm_tsk<br>ras_ter<br>ter_tsk<br>sta_ovr<br>stp_ovr | del_tsk<br>sac_tsk | get_tst<br>get_pri<br>ref_tsk<br>ref_ovr<br>prb_mem |
| セマフォ | sig_sem | wai_sem<br>pol_sem<br>twai_sem | del_sem<br>ini_sem<br>sac_sem | ref_sem |
| イベントフラグ | set_flg<br>clr_flg | wai_flg<br>pol_flg<br>twai_flg | del_flg<br>ini_flg<br>sac_flg | ref_flg |
| データキュー | snd_dtq<br>psnd_dtq<br>tsnd_dtq<br>fsnd_dtq | rcv_dtq<br>prcv_dtq<br>trcv_dtq | del_dtq<br>ini_dtq<br>sac_dtq | ref_dtq |
| 優先度データキュー | snd_pdq<br>psnd_pdq<br>tsnd_pdq | rcv_pdq<br>prcv_pdq<br>trcv_pdq | del_pdq<br>ini_pdq<br>sac_pdq | ref_pdq |
| ミューテックス | loc_mtx<br>ploc_mtx<br>tloc_mtx<br>unl_mtx | - | del_mtx<br>ini_mtx<br>sac_mtx | ref_mtx |
| メッセージバッファ | snd_mbf<br>psnd_mbf<br>tsnd_mbf | rcv_mbf<br>prcv_mbf<br>trcv_mbf | del_mbf<br>ini_mbf<br>sac_mbf | ref_mbf |
| スピンロック | loc_spn<br>try_spn<br>unl_spn | - | del_spn<br>sac_spn | ref_spn |
| 固定長メモリプール | get_mpf<br>pget_mpf<br>tget_mpf | rel_mpf | del_mpf<br>ini_mpf<br>sac_mpf | ref_mpf |
| 周期通知 | sta_cyc<br>msta_cyc | stp_cyc | del_cyc<br>sac_cyc | ref_cyc |
| アラーム通知 | sta_alm<br>msta_alm | stp_alm | del_alm<br>sac_alm | ref_alm |

| | | | | |
|---|---|---|---|---|
| 割込みサービスルーチン | - | - | del_isr<br>sac_isr | ref_isr |
| 保護ドメイン | acre_yyy | rot_rdq<br>mrot_rdq | sac_dom | get_lod<br>mget_lod<br>get_nth<br>mget_nth |
| システム状態1 | dis_dsp<br>ena_dsp<br>chg_som | loc_cpu<br>unl_cpu<br>dis_int<br>ena_int<br>clr_int<br>ras_int<br>chg_ipm | set_tim<br>adj_tim<br>set_dft<br>att_mem<br>att_pma<br>cfg_int<br>def_svc | get_tim<br>ref_sys<br>prb_int<br>ref_int<br>get_ipm<br>get_som<br>ref_cfg<br>ref_ver |
| システム状態2 | dis_ter<br>ena_ter | sus_tsk | ter_tsk | - |

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，sns_ter）
- 高分解能タイマの参照（fch_hrt）
- 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）
- 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）

【補足説明】

sus_tskとter_tskは，タスクに対するアクセス許可ベクタとシステム状態に対するアクセス許可ベクタの両方で保護される．前者が個々のタスクに対してこれらのサービスコールが呼べるかどうかを制御するための機能であるのに対して，後者はシステム全体でタスクの強制待ち／強制終了を使用するかどうかを制御するための機能である．

xsns_dpn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】

システム時刻に対するアクセス許可ベクタは廃止し，システム時刻に対する操作は，システム状態に対する管理操作と参照操作で保護するように変更した．また，保護ドメインに対するアクセス許可ベクタを新設し，システム状態に対するアクセス許可ベクタで保護していたサービスコールの中で，スケジューリングに関するものとオブジェクトを生成するサービスコールを，保護ドメインに対するアクセス許可ベクタで保護するように変更した．

get_yyyは，対象オブジェクトに対する通常操作1としていたのを，参照操作に変更した．unl_mtxを，アクセス許可ベクタによる保護を行うように変更した．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．

システム状態に対するアクセス許可ベクタを2つ持つこととし，sus_tskとter_tskを，タスクに対するアクセス許可ベクタに加えて，システム状態に対するアクセス許可ベクタでも呼出しを禁止できるようにした．

【TOPPERS新世代カーネル統合仕様との関係】

保護ドメインに対するアクセス許可ベクタを新設し，システム状態に対するアクセス許可ベクタで保護していたサービスコールの中で，スケジューリングに関するものとオブジェクトを生成するサービスコールを，保護ドメインに対するアクセス許可ベクタで保護するように変更した．

システム状態に対するアクセス許可ベクタを2つ持つこととし，sus_tskとter_tskを，タスクに対するアクセス許可ベクタに加えて，システム状態に対するアクセス許可ベクタでも呼出しを禁止できるようにした．

## 5.9 ターゲット定義事項一覧

- メモリオブジェクトの先頭番地とサイズに対する制約〔P〕［NGKI0070］
- サポートするクラスとその識別名およびID番号〔M〕［NGKI0644］
- サポートするプロセッサの数と各プロセッサの識別名〔M〕［NGKI0642］
- どのプロセッサをマスタプロセッサとするか〔M〕［NGKI0101］
- 各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〔M〕［NGKI0107］
- オブジェクトを生成する際にコンフィギュレータが確保するメモリ領域の配置場所〔pM〕［NGKI0664］
- どのプロセッサをタイムイベント処理プロセッサとするか〔M〕［NGKI0622］
- 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以外で，標準のセクションと扱うセクション〔P〕［NGKI0113］
- 手動メモリ配置をサポートするか〔P〕［NGKI0608］
- 割込みサービスルーチンは，属するクラスの初期割付けプロセッサのみで実行されるものとするか〔M〕［NGKI0656］
- システム周期オーバラン例外の例外ハンドラ番号〔P〕［NGKI0603］
- 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］
- 割込み優先度の段階数［NGKI0256］
- 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］
- 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］

- 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］
- 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［NGKI0292］
- カーネル管理外の割込みの設定方法として，3つの方法のいずれを採用するか［NGKI0295］
- 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［NGKI0297］
- CPU例外ハンドラ番号の付与方法［NGKI0306］
- CPU例外ハンドラをプロセッサ毎に登録するか，複数のプロセッサに対して登録するか，両方の方法を併用するか〔M〕［NGKI0661］
- 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［NGKI0314］
- 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に，どのCPU例外ハンドラが起動されるか〔P〕［NGKI0411］
- 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に，メモリアクセス違反ハンドラが起動されるものとするか〔P〕［NGKI0413］
- メモリアクセス違反ハンドラで，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〔P〕［NGKI0414］
- ユーザタスクから，サービスコールを通じて，アラインされていないメモリアクセスを行おうとした場合に，CPU例外ハンドラが起動されるものとするかと，CPU例外ハンドラが起動される場合にどのCPU例外ハンドラが起動されるか〔P〕［NGKI0671］
- 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に対する制限〔P〕［NGKI0417］
- 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に対する制限〔P〕［NGKI0423］
- ユーザスタック領域に対する実行アクセスを禁止しないこととするか〔P〕［NGKI0440］
- 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か〔P〕［NGKI0441］
- int8_t，uint8_t，int64_t，uint64_t，int128_t，uint128_t，float32_t，double64_tが使用できるか［NGKI0488］［NGKI0490］
- ターゲット定義のタスク属性［NGKI1016］
- タスクが用いるスタック領域のサイズの最小値［NGKI1042］
- タスクのシステムスタック領域のサイズの最小値〔P〕［NGKI1044］
- タスクが用いるスタック領域の先頭番地とサイズに対する制約［NGKI1050］［NGKI1056］
- ユーザタスクのスタック領域（ユーザスタック領域）をアプリケーションで確保する方法〔P〕［NGKI1059］
- 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］
- タスクのシステムスタック領域のデフォルトのサイズ〔P〕［NGKI1063］
- HRP3カーネルの動的生成機能拡張パッケージにおいて，acre_tskでユーザタスクを生成する場合に，ユーザスタック領域をアプリケーションで確保する方法とユーザスタック領域に対する制約［HRPS0240］［HRPS0241］

- 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］
- ターゲット定義のスピンロック属性〔M〕［NGKI5175］
- ハードウェアが持つ機構で実現できるスピンロックの数〔M〕［NGKI5169］
- 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］
- 高分解能タイマの周期とカウントアップの進み幅［NGKI3574］
- HRTCNT型が32ビットと64ビットのいずれであるか［NGKI3573］
- オーバランハンドラの残りプロセッサ時間に指定できる値の上限［NGKI2594］
- オーバランハンドラ機能がサポートされているか［NGKI2598］
- 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，オーバランハンドラの動作を開始できるか〔M〕［NGKI5145］
- 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，オーバランハンドラの動作を停止できるか〔M〕［NGKI5147］
- 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，オーバランハンドラの残りプロセッサ時間を参照できるか〔M〕［NGKI5139］
- 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］［NGKI2815］
- ターゲット定義のメモリリージョン属性〔P〕［NGKI2771］
- 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属性が無視されるか〔P〕［NGKI2789］
- 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］
- 標準のメモリオブジェクト属性の定義を，標準的なものから変更するか〔P〕［NGKI3964］
- ターゲット定義のメモリオブジェクト属性が，メモリオブジェクトアクセス属性であるか〔P〕［NGKI3987］
- ATT_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］
- ターゲット定義の予約セクション〔P〕［NGKI5129］
- ATT_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_MODがサポートされているか〔P〕［NGKI2859］
- ATT_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／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／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］
- ATT_PMA／att_pmaがサポートされているか〔P〕［NGKI2903］［HRPS0156］［HRMPS0118］
- HRP3カーネルの動的生成機能拡張パッケージにおいて，acre_tskで生成したユーザタスクのユーザスタック領域に対して，他のタスクからアクセスが許可されているかをprb_memによりチェックした場合に，E_NOEXSエラーとなるか［HRPS0244］
- ターゲット定義の割込み要求ライン属性［NGKI2945］
- ターゲット定義の割込みサービスルーチン属性［NGKI5176］
- 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［NGKI2985］
- CFG_INT／cfg_intで，各割込み要求ラインが属することができるクラスに対する制限〔M〕［NGKI5184］
- どのプロセッサからどの割込み要求ラインの属性を設定できないか〔MD〕［NGKI5177］
- 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〕［NGKI3019］
- CRE_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〕［NGKI3075］
- DEF_INHで割込みハンドラを定義できない割込みハンドラ番号［NGKI3078］
- 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］
- 割込み要求ラインがレベルトリガである場合のclr_intの振舞い［NGKI3928］
- clr_intがサポートされているか［NGKI3925］
- clr_intにより，どのような場合に割込み要求ラインに対する割込み要求をクリアできないか［NGKI3930］
- clr_intにおいて，割込み要求のクリアの振舞いが，この仕様の規定と異なるか［NGKI3931］
- 割込み要求ラインがレベルトリガである場合のras_intの振舞い［NGKI3940］
- ras_intがサポートされているか［NGKI3937］
- ras_intにより，どのような場合に割込み要求ラインに対して割込みを要求できないか［NGKI3942］
- ras_intにおいて，割込みの要求の振舞いが，この仕様の規定と異なるか［NGKI3943］

- prb_intがサポートされているか［NGKI3949］
- prb_intにより，どのような場合に割込み要求ラインに対する割込み要求をチェックできないか［NGKI3952］
- prb_intにおいて，割込み要求のチェックの振舞いが，この仕様の規定と異なるか［NGKI3953］
- 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］
- タイムウィンドウの長さに指定できる値の上限〔P〕［NGKI5050］
- 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］
- 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に，コンフィギュレータにより確保されるスタック領域は，どのクラス向けの標準メモリリージョン内に確保されるか〔PM〕［NGKI5138］
- 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］
- カーネルメモリプール領域の先頭番地とサイズに対する制約［NGKI5078］［NGKI5083］
- TA_FPUの値［NGKI4045］
- TA_WTHROUGHの値［NGKI4046］
- TA_POSEDGE，TA_NEGEDGE，TA_BOTHEDGE，TA_LOWLEVEL，TA_HIGHLEVELの値［NGKI4047］
- TA_DIRECTの値［NGKI4048］

## 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 | 元になった英語 |
|---|---|
| act | activate |
| acv | access permission vector |
| adj | adjust |
| aid | automatically assigned ID |
| 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 |
| ext | exit |
| get | get |
| ini | initialize |
| 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 permission 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 notification |
| cfg | configuration |
| cpu | CPU |
| ctx | context |
| cyc | cyclic notification |
| dft | dritf |
| did | domain ID |
| dom | domain |
| dpn | dispatch pending |
| dsp | dispatch |
| dtq | data queue |
| epr | execution priority |
| exc | exception |
| fch | fetch |
| flg | eventflag |
| hrt | high-resolution timer |
| ics | interrupt context stack |
| inf | information |
| inh | interrupt handler |
| ini | initialization |
| int | interrupt |
| ipm | interrupt priority mask |
| isr | interrupt service routine |
| ker | kernel |
| loc | lock |
| mbf | message buffer |
| mpf | fixed-sized memory pool |
| mpk | kernel 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 |
| scy | system cycle |
| sec | section |
| sem | semaphore |
| som | system operating mode |
| srg | standard memory region |
| spn | spin lock |
| spr | sub-priority |
| stk | stack |
| sys | system |
| svc | service call |
| ter | termination |
| tid | task ID |
| tim | time |
| tsk | task |
| twd | time window |
| utm | time in micro second |
| ver | version |
| wai | wait |
| wup | wake up |

### 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 |
| s | synchronized |
| t | timeout |
| x | exception |

## 5.11 バージョン履歴

| | | |
|---|---|---|
| 2014年11月24日 | Release 3.A.0 | 最初の早期リリース（α版） |
| 2016年2月8日 | Release 3.0.0 | 最初の一般公開 |
| 2017年7月21日 | Release 3.1.0 | |
| 2018年4月19日 | Release 3.2.0 | |
| 2018年4月26日 | Release 3.2.1 | |
| 2019年3月20日 | Release 3.3.0 | |
| 2019年10月5日 | Release 3.4.0 | 長期サポートバージョンの起点 |
| 2020年3月31日 | Release 3.4.1 | |
| 2020年11月6日 | Release 3.4.2 | |
| 2020年12月23日 | Release 3.5.0 | |
| 2023年3月11日 | Release 3.6.0 | |

# 付録A　バージョン間の差分

この付録では，この仕様のバージョンアップにより，仕様の内容に変更があったものをリストアップしている．仕様の内容に影響のない記述の変更や，仕様を補足説明する説明文などの変更はリストアップしていない．また，単純な誤記の修正もリストアップしていない．

## A.1 Release 3.2.0とRelease 3.2.1の差分

- タグ番号の付与漏れの修正

| | |
|---|---|
| NGKI3991〔P〕 | タグ番号を付与 |

## A.2 Release 3.2.1とRelease 3.3.0の差分

### (1) マルチプロセッサ対応に関する仕様変更

- プロセッサ間の排他制御を実現するための機構が必要であることを記述

| | |
|---|---|
| NGKI0639〔M〕 | 追加 |
| NGKI0640〔M〕 | 追加 |

- ローカルタイマ方式とグローバルタイマ方式の考え方を廃止し，タイムイベント処理プロセッサの概念を導入

| | |
|---|---|
| NGKI0563〔M〕 | 削除 |
| NGKI0621〔M〕 | 追加 |
| NGKI0108〔M〕 | 削除 |
| NGKI0109〔M〕 | 削除 |
| NGKI0110〔M〕 | 削除 |
| NGKI0111〔M〕 | 削除 |
| NGKI0622〔M〕 | 追加 |
| NGKI0526〔M〕 | 記述の変更 |
| NGKI0527〔M〕 | 削除 |
| NGKI3603 | 記述の変更 |
| NGKI2337 | 削除．NGKI3603に統合 |
| NGKI2338〔M〕 | 削除 |
| NGKI2339〔M〕 | 削除 |
| NGKI2340〔M〕 | 削除 |
| NGKI2341〔M〕 | 削除 |
| NGKI2342〔M〕 | 削除 |
| NGKI3595〔M〕 | 削除 |
| NGKI3576〔M〕 | 削除 |
| NGKI3568〔M〕 | 削除 |

| | |
|---|---|
| NGKI2355〔M〕 | 削除 |
| NGKI3590〔M〕 | 削除 |
| NGKI3602〔M〕 | 削除 |
| NGKI3577〔M〕 | 削除 |
| NGKI2375〔M〕 | 削除 |
| NGKI5088〔M〕 | 追加 |
| NGKI2401〔M〕 | 削除 |
| NGKI5089〔M〕 | 追加 |
| NGKI2407〔MD〕 | 削除 |
| NGKI5090〔MD〕 | 追加 |
| NGKI2454〔M〕 | 削除 |
| NGKI2483〔M〕 | 削除 |
| NGKI5091〔M〕 | 追加 |
| NGKI2503〔M〕 | 削除 |
| NGKI5092〔M〕 | 追加 |
| NGKI2509〔MD〕 | 削除 |
| NGKI5093〔MD〕 | 追加 |
| NGKI2561〔M〕 | 削除 |

- クラスとプロセッサの扱いに関して漏れていた規定を追加

| | |
|---|---|
| NGKI0644〔M〕 | 追加 |
| NGKI0645〔M〕 | 追加 |
| NGKI0642〔M〕 | 追加 |
| NGKI0643〔M〕 | 追加 |

- 構成マクロTOPPERS_TEPP_PRCに関する規定を追加

| | |
|---|---|
| NGKI0526〔M〕 | 記述の追加 |

- CPUロック状態でext_tskが呼ばれた場合に，CPUロック状態の解除と自タスクの終了の間に，割込みとタスク切換えが発生する場合があることを規定．また，これに関連して，CPUロック状態で強制待ち状態［実行継続中］のタスクがext_tskを発行した場合には，強制待ち状態に遷移する場合もあることを規定

| | |
|---|---|
| NGKI0648〔M〕 | 記述の追加 |
| NGKI5183〔M〕 | 記述の追加 |

- スピンロックがカーネルによって返却される状況を厳密に規定

| | |
|---|---|
| NGKI2121〔M〕 | 削除．NGKI5167に変更 |
| NGKI2122〔M〕 | 削除．NGKI5167に変更 |

| | |
|---|---|
| NGKI5167〔M〕 | 追加 |

- エミュレーションによりスピンロックが実現される場合を規定し，スピンロック属性を規定．あわせて，E_NORESエラーの発生状況を変更

| | |
|---|---|
| NGKI2125〔M〕 | 削除．NGKI5168に変更 |
| NGKI2126〔M〕 | 削除 |
| NGKI5168〔M〕 | 追加 |
| NGKI5175〔M〕 | 追加 |
| NGKI5169〔M〕 | 追加 |
| NGKI5170〔M〕 | 追加 |
| NGKI5171〔M〕 | 追加 |
| NGKI5172〔M〕 | 追加 |
| NGKI5173〔M〕 | 追加 |
| NGKI2142〔M〕 | 削除．NGKI5169に変更 |
| NGKI2143〔M〕 | 削除．NGKI5174に変更 |
| NGKI5174〔M〕 | 追加 |

- マルチプロセッサ対応カーネルにおいて，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タスクに対してオーバランハンドラの操作をした場合の振る舞いを，ターゲット定義と規定

| | |
|---|---|
| NGKI5145〔M〕 | 追加 |
| NGKI5146〔M〕 | 追加 |
| NGKI5147〔M〕 | 追加 |
| NGKI5148〔M〕 | 追加 |
| NGKI5139〔M〕 | 追加 |
| NGKI5140〔M〕 | 追加 |

- CRE_ISR／DEF_INHと対応するCFG_INTが異なるクラスの囲み中にある場合のエラーの扱いを整理

| | |
|---|---|
| NGKI5143〔M〕 | 追加 |
| NGKI2992〔M〕 | 削除．NGKI5141，NGKI5142に変更 |
| NGKI3017〔M〕 | 削除．NGKI5142に統合 |
| NGKI5141〔M〕 | 追加 |
| NGKI5142〔M〕 | 追加 |

- 属するクラスの指定が有効範囲外のエラーは，静的APIでも発生することを規定

| | |
|---|---|
| NGKI1031〔M〕 | 記述の変更 |
| NGKI5111〔MD〕 | 追加 |

| | |
|---|---|
| NGKI1458〔M〕 | 記述の変更 |
| NGKI5112〔MD〕 | 追加 |
| NGKI1564〔M〕 | 記述の変更 |
| NGKI5113〔MD〕 | 追加 |
| NGKI1671〔M〕 | 記述の変更 |
| NGKI5114〔MD〕 | 追加 |
| NGKI1806〔M〕 | 記述の変更 |
| NGKI5115〔MD〕 | 追加 |
| NGKI2027〔M〕 | 記述の変更 |
| NGKI5116〔MD〕 | 追加 |
| NGKI3305〔M〕 | 記述の変更 |
| NGKI5117〔MD〕 | 追加 |
| NGKI2134〔M〕 | 記述の変更 |
| NGKI5118〔MD〕 | 追加 |
| NGKI2227〔M〕 | 記述の変更 |
| NGKI5119〔MD〕 | 追加 |
| NGKI2386〔M〕 | 記述の変更 |
| NGKI5120〔MD〕 | 追加 |
| NGKI2494〔M〕 | 記述の変更 |
| NGKI5121〔MD〕 | 追加 |
| NGKI5103〔PM〕 | 追加 |
| NGKI5104〔PM〕 | 追加 |
| NGKI5105〔PM〕 | 追加 |
| NGKI2869〔PM〕 | 記述の変更 |
| NGKI2889〔PM〕 | 記述の変更 |
| NGKI2970〔M〕 | 記述の変更 |
| NGKI3001〔M〕 | 記述の変更 |
| NGKI5122〔MD〕 | 追加 |
| NGKI3053〔M〕 | 記述の変更 |
| NGKI3132〔M〕 | 記述の変更 |
| NGKI5107〔PM〕 | 追加 |
| NGKI5128〔M〕 | 追加 |
| NGKI5164〔M〕 | 追加 |
| NGKI5165〔M〕 | 追加 |

(2) 保護機能対応に関する仕様変更

・特権モードと非特権モード，MMUまたはMPUが必要であることを記述

| | |
|---|---|
| NGKI0646〔P〕 | 追加 |
| NGKI0647〔P〕 | 追加 |

・スケジューリング単位番号とSCHEDNOマクロについて規定

| | |
|---|---|
| NGKI0630〔P〕 | 追加 |
| NGKI0631〔P〕 | 追加 |
| NGKI0632〔M〕 | 追加 |
| NGKI0633〔PM〕 | 追加 |
| NGKI0634〔P｜M〕 | 追加 |

・スケジューリング単位番号の概念を用いて仕様を記述

| | |
|---|---|
| NGKI3772〔P〕 | 削除．NGKI5125に統合 |
| NGKI3773〔P〕 | 削除．NGKI5125に統合 |
| NGKI3622〔M〕 | 削除．NGKI5125に統合 |
| NGKI5125〔P｜M〕 | 追加 |
| NGKI2699〔P｜M〕 | 記述の変更 |
| NGKI3779〔P〕 | 削除．NGKI0630,NGKI0631に統合 |
| NGKI3880〔M〕 | 削除．NGKI0632に統合 |
| NGKI2702〔P｜M〕 | 記述の変更 |
| NGKI3774〔P〕 | 削除．NGKI5126に統合 |
| NGKI3775〔P〕 | 削除．NGKI5126に統合 |
| NGKI3630〔M〕 | 削除．NGKI5126に統合 |
| NGKI5126〔P｜M〕 | 追加 |
| NGKI3639〔P｜M〕 | 記述の変更 |
| NGKI3882〔P〕 | 削除．NGKI0630,NGKI0631に統合 |
| NGKI3883〔M〕 | 削除．NGKI0632に統合 |
| NGKI3776〔P〕 | 削除．NGKI5127に統合 |
| NGKI3777〔P〕 | 削除．NGKI5127に統合 |
| NGKI3649〔M〕 | 削除．NGKI5127に統合 |
| NGKI5127〔P｜M〕 | 追加 |
| NGKI3658〔P｜M〕 | 記述の変更 |
| NGKI3885〔P〕 | 削除．NGKI0630,NGKI0631に統合 |
| NGKI3886〔M〕 | 削除．NGKI0632に統合 |

・スケジューリング単位番号のパラメータ名を変更

| | |
|---|---|
| NGKI3549〔P｜M〕 | 記述の変更 |
| NGKI2696〔P｜M〕 | 記述の変更 |
| NGKI3632〔P｜M〕 | 記述の変更 |
| NGKI3635〔P｜M〕 | 記述の変更 |
| NGKI3651〔P｜M〕 | 記述の変更 |
| NGKI3654〔P｜M〕 | 記述の変更 |

- 標準メモリリージョンの扱いを変更・明確化

| | |
|---|---|
| NGKI3259〔P〕 | 記述の変更 |
| NGKI3894〔P〕 | 記述の変更 |
| NGKI5098〔PM〕 | 追加 |
| NGKI5099〔PM〕 | 追加 |
| NGKI2854〔PM〕 | 記述の変更 |
| NGKI3895〔PM〕 | 記述の変更 |
| NGKI3893〔P〕 | 記述の変更 |
| NGKI2853〔PM〕 | 記述の変更 |
| NGKI3271〔PM〕 | 削除 |

- DEF_SRGのパラメータ種別に関する規定を追加

| | |
|---|---|
| NGKI5149〔P〕 | 追加 |

- スタック領域や固定長メモリプール領域がどのメモリリージョン内に配置されるかを規定

| | |
|---|---|
| NGKI5136〔P〕 | 追加 |
| NGKI1051〔P〕 | 記述の変更 |
| NGKI1061〔P〕 | 記述の変更 |
| NGKI2242〔P〕 | 記述の変更 |
| NGKI5137〔P〕 | 追加 |
| NGKI5138〔P〕 | 追加 |

- 保護ドメイン毎の標準セクションを廃止

| | |
|---|---|
| NGKI0114〔P〕 | 削除 |
| NGKI0115〔P〕 | 削除 |
| NGKI0116〔P〕 | 削除 |
| NGKI2836〔P〕 | 削除 |

- ATT_SECのregnameの記述を省略可能に

| | |
|---|---|
| NGKI5150〔P〕 | 追加 |

| | |
|---|---|
| NGKI5151〔P〕 | 追加 |

- ターゲット定義の予約セクションに関して規定

| | |
|---|---|
| NGKI5129〔P〕 | 追加 |

- 保護ドメインに対して設定できる最高のタスク優先度の適用範囲を広げた

| | |
|---|---|
| NGKI5123〔PD〕 | 追加 |
| NGKI5124〔P〕 | 追加 |

- slt_domではE_RSATRエラーが返らないことを規定

| | |
|---|---|
| NGKI3747〔P〕 | 記述の変更 |

- タイムウィンドウに関する規定を追加・明確化

| | |
|---|---|
| NGKI5003〔P〕 | 記述の変更 |
| NGKI5094〔P〕 | 追加 |
| NGKI5095〔P〕 | 追加 |
| NGKI5100〔PM〕 | 追加 |

- タイムウィンドウ通知の機能を追加

| | |
|---|---|
| NGKI5096〔P〕 | 追加 |
| NGKI5040〔P〕 | 記述の変更 |
| NGKI5155〔P〕 | 追加 |
| NGKI5047〔P〕 | 記述の変更 |
| NGKI5097〔P〕 | 追加 |
| NGKI5106〔PM〕 | 追加 |

- システム周期オーバラン例外からリターンした場合の振る舞いを規定

| | |
|---|---|
| NGKI0635〔P〕 | 追加 |

- ユーザスタック領域の特殊扱いに関する規定（acre_tskを対象に追加したもの）を，HRP3カーネルの個別の規定に変更し，規定を厳密化

| | |
|---|---|
| NGKI3990〔P〕 | 削除 |
| HRPS0240 | 追加 |
| HRPS0241 | 追加 |
| HRPS0242 | 追加 |
| HRPS0243 | 追加 |
| HRPS0244 | 追加 |

(3) 保護機能とマルチプロセッサの両方に対応するための仕様の追加と変更

- マルチプロセッサ対応カーネルにおける時間パーティショニング使用時のスケジューリング規則（タイムウィンドウの扱いを含む）と制限事項を規定

| | |
|---|---|
| NGKI0624〔PM〕 | 追加 |
| NGKI0627〔PM〕 | 追加 |
| NGKI0625〔PM〕 | 追加 |
| NGKI0626〔PM〕 | 追加 |
| NGKI0636〔PM〕 | 追加 |
| NGKI0637〔PM〕 | 追加 |

- アイドルドメインにまとめられていないユーザドメインに属するタスク等を，そのユーザドメインに対してタイムウィンドウを割り当てられていないプロセッサに割り付けることはできないことを規定

| | |
|---|---|
| NGKI0628〔PM〕 | 追加 |
| NGKI5108〔PM〕 | 追加 |
| NGKI5109〔PM〕 | 追加 |
| NGKI5110〔PM〕 | 追加 |
| NGKI5130〔PM〕 | 追加 |
| NGKI5131〔PM〕 | 追加 |
| NGKI5132〔PM〕 | 追加 |
| NGKI5133〔PM〕 | 追加 |

- SAC_SYS，ACV_DOM，LMT_DOMをクラスの囲みの中に記述した場合を，エラーから警告に変更

| | |
|---|---|
| NGKI2675〔PM〕 | 記述の変更 |
| NGKI3758〔PM〕 | 記述の変更 |
| NGKI3443〔PM〕 | 記述の変更 |

- プロセッサ間でシステム動作モードの切換えが同期しない状況に関する規定を追加

| | |
|---|---|
| NGKI5134〔PM〕 | 追加 |
| NGKI5135〔PM〕 | 追加 |

- DEF_SCYとCRE_SOMはクラスの囲みの外に記述することに変更

| | |
|---|---|
| NGKI5014〔PM〕 | 記述の変更 |
| NGKI5053〔PM〕 | 削除（タグ番号の付与ミスだった） |
| NGKI5023〔PM〕 | 追加 |

- TOPPRES/HRMPカーネルのカーネル識別番号を規定

| | |
|---|---|
| NGKI0528 | 記述の変更 |

- HRMP3カーネルにおける規定を追加

| | |
|---|---|
| HRMPS0001 | 追加 |
| HRMPS0002 | 追加 |
| HRMPS0003 | 追加 |
| HRMPS0004 | 追加 |
| HRMPS0005 | 追加 |
| HRMPS0006 | 追加 |
| HRMPS0007 | 追加 |
| HRMPS0008 | 追加 |
| HRMPS0009 | 追加 |
| HRMPS0010 | 追加 |
| HRMPS0011 | 追加 |
| HRMPS0012 | 追加 |
| HRMPS0013 | 追加 |
| HRMPS0101 | 追加 |
| HRMPS0103 | 追加 |
| HRMPS0104 | 追加 |
| HRMPS0105 | 追加 |
| HRMPS0106 | 追加 |
| HRMPS0107 | 追加 |
| HRMPS0108 | 追加 |
| HRMPS0109 | 追加 |
| HRMPS0110 | 追加 |
| HRMPS0111 | 追加 |
| HRMPS0112 | 追加 |
| HRMPS0113 | 追加 |
| HRMPS0114 | 追加 |
| HRMPS0117 | 追加 |
| HRMPS0118 | 追加 |
| HRMPS0119 | 追加 |
| HRMPS0120 | 追加 |

### (4) その他の仕様変更

- カーネルメモリプール領域に関する仕様の整理とマルチプロセッサ対応カーネル向けの拡張

| | |
|---|---|
| NGKI5085〔PD〕 | 削除．NGKI5157とNGKI5162に分割 |
| NGKI5156〔PD〕 | 追加 |
| NGKI5157〔PD〕 | 追加 |

| | |
|---|---|
| NGKI5158〔MD〕 | 追加 |
| NGKI5159〔MD〕 | 追加 |
| NGKI5160〔MD〕 | 追加 |
| NGKI5161〔MD〕 | 追加 |
| NGKI5162〔PD〕 | 追加 |
| NGKI5163〔MD〕 | 追加 |

- ターゲット定義の割込みサービスルーチン属性に関する規定を追加

| | |
|---|---|
| NGKI2951 | 削除．NGKI5176に変更 |
| NGKI2952 | 記述の変更 |
| NGKI5176 | 追加 |

- コンフィギュレータに対するINCLUDEディレクティブの文法の制限

| | |
|---|---|
| NGKI0443 | 削除．NGKI0619に変更 |
| NGKI0619 | 追加 |

- コンフィギュレータに対する条件ディレクティブの振る舞いの変更

| | |
|---|---|
| NGKI0446 | 削除．NGKI0620に変更 |
| NGKI0620 | 追加 |

- SILの微少時間待ちの機能は，非特権モードから呼び出した場合に保証されないものと変更

| | |
|---|---|
| NGKI0802 | 記述の変更 |

- SILのメモリ空間同期書込み関数群とI/O空間同期書込み関数群を追加

| | |
|---|---|
| NGKI0845 | 追加 |
| NGKI0846 | 追加 |
| NGKI0847 | 追加 |
| NGKI0848 | 追加 |
| NGKI0849 | 追加 |
| NGKI0850 | 追加 |
| NGKI0851 | 追加 |
| NGKI0843 | 記述の変更 |

- 初期化ルーチンと終了処理ルーチンの先頭番地のエラーチェックをターゲット定義とする規定を廃止

| | |
|---|---|
| NGKI3246 | 削除 |
| NGKI3253 | 削除 |

### (5) 個別のカーネル仕様の変更

- HRP3カーネルにおけるオーバランハンドラ機能に関する規定を追加

| | |
|---|---|
| HRPS0013 | 追加 |
| HRPS0014 | 追加 |
| HRPS0238 | 追加 |
| HRPS0239 | 追加 |

- SSP3カーネルで，act_tskのtskidにTSK_SELFを指定する機能をサポートしないことを規定

| | |
|---|---|
| SSPS0163 | 追加 |

- SSP3カーネルで，get_tstとget_tidをサポートするように変更

| | |
|---|---|
| SSPS0156 | 削除 |
| SSPS0133 | 削除 |

(6) 仕様の明確化と整理

- 静的APIの一般定数式パラメータにNULLと記述することの明確化

| | |
|---|---|
| NGKI0638 | 追加 |

- タスクの次回起動時の割付けプロセッサの扱いを明確化

| | |
|---|---|
| NGKI1007〔M〕 | 記述の変更 |
| NGKI1008〔M〕 | 記載場所の変更 |
| NGKI1009〔M〕 | 削除 |

- カーネルドメインに属する処理単位は常に実行できることを明記

| | |
|---|---|
| NGKI0623〔M〕 | 追加 |

- 強制待ち状態［実行継続中］から休止状態への状態遷移があることを明記

| | |
|---|---|
| NGKI0629 | 追加 |

- 制約タスクのサブ優先度をサービスコールで変更できないことを明記

| | |
|---|---|
| NGKI0236 | 記述の変更 |

- コンフィギュレータが仕様に規定されている以外の警告メッセージを出力する場合があることを明記

| | |
|---|---|
| NGKI0641 | 追加 |

- 割込み要求ラインの属性の設定に関するターゲット定義のエラーの扱いを整理

| | |
|---|---|
| NGKI2985 | 記述の変更 |
| NGKI2986 | 削除．NGKI2985に統合 |
| NGKI2987〔M〕 | 記述の変更 |

NGKI5177〔M〕　追加

NGKI5184〔M〕　追加

- ターゲット定義のCPU例外ハンドラ属性がない場合，CPU例外ハンドラ属性にはTA_NULLを指定しなければならないことを明記

NGKI5178　追加

- タイムウィンドウの登録時に，タイムイベントハンドラの呼出しによる通知を指定した場合は，E_OACVエラーとなることを明記

NGKI3695〔P〕　記述の変更

- オーバランハンドラが起動されると，オーバランハンドラは動作していない状態になることを明記

NGKI3993　追加

- TDOM_SELFの意味の変更

NGKI0513〔P〕　記述の変更

- TSOM_STPの追加

NGKI0514　記述の変更

- ATT_TWDによる割り当て対象が，ユーザドメインのみであることを明確化

NGKI5040〔P〕　記述の変更

- DEF_ICSのistkパラメータの型を修正（STK_T → STK_T *）

NGKI3212　記述の変更

- T_DSVC型のフィールド名を修正（svcrtn → extsvc）

NGKI4035〔D〕　記述の変更

- タスクのスタック領域に，他のタスクからアクセスできるとは限らないことを明記

NGKI5166　追加

- yyymbにNULL以外を指定できないカーネルでは，TSZ_YYYMB，TCNT_YYYMBを用意しないことを明記（TSZ_MBFMBは例外）

NGKI5179　追加

NGKI5180　追加

NGKI5181　追加

NGKI5182　追加

- 記述の明確化

| | |
|---|---|
| NGKI0599〔P〕 | 記述の明確化 |
| NGKI0600〔P〕 | 記述の明確化 |

- 不要な記述の削除

| | |
|---|---|
| ASPS0186 | 削除 |
| ASPS0204 | 削除 |
| ASPS0207 | 削除 |
| HRPS0239 | 削除 |

- 保護機能対応カーネルのみでサポートされている静的APIのエラー記述から，〔P〕の記号を削除

| | |
|---|---|
| NGKI2868〔PD〕 | 記述の変更 |
| NGKI2870〔PD〕 | 記述の変更 |
| NGKI2871〔PD〕 | 記述の変更 |
| NGKI2888〔PD〕 | 記述の変更 |
| NGKI2890〔PD〕 | 記述の変更 |
| NGKI2891〔PD〕 | 記述の変更 |

### (7) 用語の変更

- 「保護ドメインの標準メモリリージョン」を「保護ドメイン向けの標準メモリリージョン」に，「クラスの標準メモリリージョン」を「クラス向けの標準メモリリージョン」に改称

| | |
|---|---|
| NGKI3891〔P〕 | 記述の変更 |
| NGKI3257〔PM〕 | 記述の変更 |
| NGKI3892〔P〕 | 記述の変更 |
| NGKI3266〔PM〕 | 記述の変更 |

- 「指定できる最高のタスク優先度」を「設定できる最高のタスク優先度」に改称

| | |
|---|---|
| NGKI3751〔P〕 | 記述の変更 |
| NGKI5086〔P〕 | 記述の変更 |
| NGKI3441〔P〕 | 記述の変更 |
| NGKI3743〔PD〕 | 記述の変更 |

### (8) タグの付与に関する変更

- タグの付与漏れの修正

| | |
|---|---|
| NGKI5151〔P〕 | タグの追加 |
| NGKI5152〔P〕 | タグの追加 |
| NGKI5153〔P〕 | タグの追加 |
| NGKI5154〔P〕 | タグの追加 |

| | |
|---|---|
| NGKI5155〔P〕 | タグの追加 |
| NGKI3992 | タグの追加 |

- タグの番号の誤りの修正

| | |
|---|---|
| NGKI0537〔P〕 | 削除．NGKI0426に変更 |
| NGKI0426〔P〕 | 追加 |

- タグの統合

| | |
|---|---|
| NGKI2606 | 削除．NGKI2595に統合 |
| NGKI2607 | 削除．NGKI2599に統合 |
| NGKI2796〔P〕 | 削除．NGKI2860に統合 |
| NGKI2797〔P〕 | 削除．NGKI2904に統合 |
| NGKI2963 | 削除．NGKI3092に統合 |
| NGKI2964 | 削除．NGKI3105に統合 |
| NGKI3917 | 削除．NGKI3926に統合 |
| NGKI3918 | 削除．NGKI3938に統合 |
| NGKI3919 | 削除．NGKI3950に統合 |
| NGKI3018〔M〕 | 削除．NGKI3019に統合 |
| NGKI3074〔M〕 | 削除．NGKI3075に統合 |

- 同じ記述に対する重複したタグの削除

| | |
|---|---|
| ASPS0102 | 削除．ASPS0009と重複 |
| ASPS0103 | 削除．ASPS0010と重複 |
| ASPS0174 | 削除．ASPS0172と重複 |
| ASPS0184 | 削除．ASPS0003と重複 |
| ASPS0185 | 削除．ASPS0004と重複 |
| FMPS0102 | 削除．FMPS0006と重複 |
| FMPS0149 | 削除．FMPS0147と重複 |
| FMPS0155 | 削除．FMPS0002と重複 |
| HRPS0102 | 削除．HRPS0005と重複 |
| HRPS0143 | 削除．HRPS0141と重複 |
| HRPS0149 | 削除．HRPS0013と重複 |
| HRPS0238 | 削除．HRPS0014と重複 |

## A.3 Release 3.3.0とRelease 3.4.0の差分

- SILスピンロックのネストを許すように変更

| | |
|---|---|
| NGKI0823 | 記述の変更 |
| NGKI0824 | 削除．NGKI0852に変更 |
| NGKI0852 | 追加 |

・HRTCNT型をターゲット定義で64ビット整数型にできるように変更

| | |
|---|---|
| NGKI0542 | 記述の変更 |
| NGKI3573 | 記述の変更 |

・設定できる最高のタスク優先度の機能を廃止．LMT_DOM／slt_domを廃止

| | |
|---|---|
| NGKI5086〔P〕 | 記述の変更 |
| NGKI0572〔P〕 | 削除 |
| NGKI3441〔P〕 | 削除 |
| NGKI3743〔PD〕 | 削除 |
| NGKI3744〔PD〕 | 削除 |
| NGKI3745〔PD〕 | 削除 |
| NGKI3746〔PD〕 | 削除 |
| NGKI3747〔P〕 | 削除 |
| NGKI3748〔PD〕 | 削除 |
| NGKI3749〔PD〕 | 削除 |
| NGKI3444〔P〕 | 削除 |
| NGKI3445〔P〕 | 削除 |
| NGKI3750〔P〕 | 削除 |
| NGKI3447〔P〕 | 削除 |
| NGKI3448〔P〕 | 削除 |
| NGKI3443〔PM〕 | 削除 |
| NGKI5123〔PD〕 | 削除 |
| NGKI3751〔P〕 | 削除 |
| NGKI5124〔P〕 | 削除 |
| NGKI4043〔PD〕 | 削除 |
| NGKI4040 | 記述の変更 |

・def_ovr，def_inh，def_excを廃止

| | |
|---|---|
| NGKI2609〔D〕 | 削除 |
| NGKI2610〔D〕 | 削除 |
| NGKI2611〔D〕 | 削除 |
| NGKI2614〔PD〕 | 削除 |
| NGKI2615〔PD〕 | 削除 |

| | |
|---|---|
| NGKI2617〔D〕 | 削除 |
| NGKI2622〔PD〕 | 削除 |
| NGKI2623〔PD〕 | 削除 |
| NGKI2626〔MD〕 | 削除 |
| NGKI2627〔MD〕 | 削除 |
| NGKI2628〔D〕 | 削除 |
| NGKI2629〔D〕 | 削除 |
| NGKI3049〔D〕 | 削除 |
| NGKI3050〔D〕 | 削除 |
| NGKI3051〔D〕 | 削除 |
| NGKI3057〔PD〕 | 削除 |
| NGKI3058〔PD〕 | 削除 |
| NGKI3060〔D〕 | 削除 |
| NGKI3071〔PD〕 | 削除 |
| NGKI3072〔PD〕 | 削除 |
| NGKI3076〔D〕 | 削除 |
| NGKI3077〔D〕 | 削除 |
| NGKI3078 | 記述の変更 |
| NGKI3079〔MD〕 | 削除 |
| NGKI3128〔D〕 | 削除 |
| NGKI3129〔D〕 | 削除 |
| NGKI3130〔D〕 | 削除 |
| NGKI3136〔PD〕 | 削除 |
| NGKI3137〔PD〕 | 削除 |
| NGKI3139〔D〕 | 削除 |
| NGKI3144〔PD〕 | 削除 |
| NGKI3145〔PD〕 | 削除 |
| NGKI3147〔D〕 | 削除 |
| NGKI3148〔D〕 | 削除 |
| NGKI4027 | 削除 |
| NGKI4033 | 削除 |
| NGKI4034 | 削除 |
| NGKI4040 | 記述の変更 |

- 時間パーティショニング使用時に，システム周期の切換えとタイムウィンドウの切換えが遅れた場合の規定を追加

| | |
|---|---|
| NGKI0650 | 追加 |
| NGKI0651 | 追加 |
| NGKI0652 | 追加 |
| NGKI0653 | 追加 |

- 時間パーティショニング使用時のスケジューリングにおいて，ユーザタスクから呼び出された拡張サービスコールは，カーネルドメインに属する処理単位とは扱われないことを明記

| | |
|---|---|
| NGKI0623〔M〕 | 記述の変更 |
| NGKI0593〔M〕 | 記述の変更 |
| NGKI0599〔M〕 | 記述の変更 |

- TCYC_HRTCNTがマクロ定義される条件を明記

| | |
|---|---|
| NGKI5187 | 追加 |

- クラスの属性に関する規定の訂正・明確化

| | |
|---|---|
| NGKI0106〔M〕 | 記述の変更 |

- 保護機能対応カーネルにおけるオブジェクト生成に必要なメモリ領域の確保方法を一般的に規定し，記述を厳密化

| | |
|---|---|
| NGKI0649〔P〕 | 追加 |
| NGKI0618 | 記述の変更 |
| NGKI5136〔P〕 | 削除 |
| NGKI1061〔P〕 | 記述の変更 |

- 保護機能対応カーネルにおいて，acre_tskでtskにNULLを指定してユーザタスクを生成する場合のユーザスタック領域と，acre_mpfでmpfにNULLを指定して固定長メモリプールを生成する場合の固定長メモリプール領域の確保方法を，未決定事項に

| | |
|---|---|
| NGKI1051〔P〕 | 記述の変更 |
| NGKI2242〔P〕 | 記述の変更 |

- SSP3カーネルで，act_tskのtskidにTSK_SELFを指定する機能をサポートしない規定を削除

| | |
|---|---|
| SSPS0163 | 削除 |

- システム周期の切換えが，システム時刻によって行われることを明記

| | |
|---|---|
| NGKI3603 | 記述の変更 |

- DEF_ICSでistkにNULLを指定した場合に，コンフィギュレータにより確保されるスタック領域が，どのクラス向けの標準メモリリージョン内に確保されるかの規定を訂正

| | |
|---|---|
| NGKI5137〔P〕 | 記述の変更 |

- カーネルが返すエラーコードの訂正

| | |
|---|---|
| HRPS0006 | 記述の変更 |
| HRPS0011 | 記述の変更 |
| HRMPS0008 | 記述の変更 |

- CPU例外からのリカバリ処理の方法(b)に関する説明を厳密化．また，同じ記述に対する重複したタグの削除

| | |
|---|---|
| NGKI0311 | 記述の変更 |
| NGKI0318 | 削除．NGKI0311と重複 |

- クラスのID番号に関する規定の記述を整理

| | |
|---|---|
| NGKI0644〔M〕 | 記述の変更 |
| NGKI0107〔M〕 | 記述の変更 |

## A.4 Release 3.4.0とRelease 3.4.1の差分

- 非タスクコンテキストからrot_rdqを呼び出した場合に，どのスケジューリング単位に属するタスクが対象となるかの記述を修正

| | |
|---|---|
| NGKI5125〔P｜M〕 | 記述の変更 |
| NGKI5208〔P｜M〕 | 追加 |

- mget_nthがE_OACVを返す条件を「参照操作が許可されていない」に修正

| | |
|---|---|
| NGKI3884〔P〕 | 記述の変更 |

- ATT_SECでregnameの記述を省略した場合に，どの保護ドメイン・クラスの標準メモリリージョンを使うかを規定

| | |
|---|---|
| NGKI5199〔P〕 | 追加 |
| NGKI5200〔PM〕 | 追加 |

- メモリ領域を配置するのが，標準メモリリージョンの中でどのメモリリージョンであるかを明記

| | |
|---|---|
| NGKI0649〔P〕 | 記述の変更 |
| NGKI1051〔P〕 | 記述の変更 |
| NGKI2242〔P〕 | 記述の変更 |
| NGKI5137〔P〕 | 記述の変更 |
| NGKI5138〔P〕 | 記述の変更 |

- コンフィギュレータにより確保されるカーネルメモリプール領域が，どのメモリリージョン内に確保されるかを規定

| | |
|---|---|
| NGKI5201〔PD〕 | 追加 |

- CFG_INTがカーネルドメインの囲みの中に記述されていない場合のエラーコードを修正

| | |
|---|---|
| NGKI2989〔P〕 | 記述場所の変更 |

- ATT_TWDでE_PARエラーになる条件に，「通知方法中の変数の番地が正しくない」を追加

| | |
|---|---|
| NGKI3699 | 参照箇所の追加 |
| NGKI3701 | 参照箇所の追加 |
| NGKI3897 | 参照箇所の追加 |

- カーネルオブジェクトを生成する旨の記述にタグを付与

| | |
|---|---|
| NGKI5188 | タグの追加 |
| NGKI5189 | タグの追加 |
| NGKI5190 | タグの追加 |
| NGKI5191 | タグの追加 |
| NGKI5192 | タグの追加 |
| NGKI5193 | タグの追加 |
| NGKI5194 | タグの追加 |
| NGKI5195〔M〕 | タグの追加 |
| NGKI5196 | タグの追加 |
| NGKI5197 | タグの追加 |
| NGKI5198 | タグの追加 |

- ATT_REGのregnameパラメータの扱いを明記

| | |
|---|---|
| NGKI2802〔P〕 | 記述の変更 |

- タイムウィンドウの通知処理が，タイムイベントの通知処理と同じ方法で行えることを明記

| | |
|---|---|
| NGKI5202〔P〕 | 記述の追加 |

- タイムイベント通知に関するエラーチェックが，タイムウィンドウ通知の登録時にも適用されることを明記

| | |
|---|---|
| NGKI3702〔P〕 | 記述の変更 |
| NGKI3898〔P〕 | 記述の変更 |
| NGKI3705〔P〕 | 記述の変更 |
| NGKI3708〔P〕 | 記述の変更 |
| NGKI3711〔P〕 | 記述の変更 |
| NGKI3716〔P〕 | 記述の変更 |

- 待ち状態のタスクに終了要求が出された場合に，待ち状態となったサービスコールからE_RASTERが返ることを明記

| | |
|---|---|
| NGKI5206 | 追加 |

- カーネル管理の割込みハンドラを定義する際に，TA_NONKERNEL属性を指定してはならないことを明記

| | |
|---|---|
| NGKI0654 | 追加 |

- スピンロックの自動解放に関する仕様を厳密化

| | |
|---|---|
| NGKI5167〔M〕 | 記述の変更 |

- マルチプロセッサ対応カーネルで，自タスクに対するget_tstがTTS_SUSを返す場合があることを明記

| | |
|---|---|
| NGKI5209〔M〕 | 追加 |

- 保護機能対応カーネルで，非特権モードで実行中には全割込みロック状態にならないことを明記し，カーネル管理外の状態にならないことは補足説明に

| | |
|---|---|
| NGKI0547〔P〕 | 削除（同じ趣旨の文章を補足説明に入れる） |
| NGKI0655〔P〕 | 追加 |

- カーネル管理外の割込みに関する記述を明確化し表現を統一

| | |
|---|---|
| NGKI2983 | 記述の変更 |
| NGKI2984 | 記述の変更 |
| NGKI3067 | 記述の変更 |
| NGKI3068 | 記述の変更 |
| NGKI3065 | 記述の変更 |
| NGKI3066 | 記述の変更 |

- 重複するエラー条件を削除

| | |
|---|---|
| NGKI1722 | 削除．NGKI1725と重複 |
| NGKI1859 | 削除．NGKI1862と重複 |

- エラー条件の記述を他の条件と重ならないように変更

| | |
|---|---|
| NGKI1515 | 記述の変更 |
| NGKI1623 | 記述の変更 |
| NGKI1725 | 記述の変更 |
| NGKI1756 | 記述の変更 |
| NGKI1862 | 記述の変更 |
| NGKI1882 | 記述の変更 |
| NGKI2072 | 記述の変更 |
| NGKI3361 | 記述の変更 |

| | |
|---|---|
| NGKI3381 | 記述の変更 |
| NGKI2292 | 記述の変更 |

- TMIN_INTPRIをカーネル管理の割込み優先度の最小値と明記

| | |
|---|---|
| NGKI2961 | 記述の変更 |

- カーネル仕様の規定であることが明確な表現に変更

| | |
|---|---|
| NGKI0099〔M〕 | 記述の変更 |
| NGKI0263 | 記述の変更 |
| NGKI0325 | 記述の変更 |
| NGKI1010〔P〕 | 記述の変更 |
| NGKI1012〔P〕 | 記述の変更 |

- パラメータの記述方法を変更

| | |
|---|---|
| NGKI2823〔P〕 | 記述の変更 |
| NGKI5058〔P〕 | 記述の変更 |
| NGKI2849〔P〕 | 記述の変更 |

- タグ番号の付与漏れの修正

| | |
|---|---|
| NGKI5207 | タグ番号を付与 |

## A.5 Release 3.4.1とRelease 3.4.2の差分

- メモリオブジェクトが持つ情報から「属するクラス」を削除し，ATT_MEMをクラスの囲みの中に記述するとエラーとなるように変更

| | |
|---|---|
| NGKI2773〔M〕 | 記述の変更 |
| NGKI2869〔PM〕 | 削除 |
| NGKI2889〔PM〕 | 削除 |
| NGKI5212〔PM〕 | 追加 |
| NGKI5213〔PM〕 | 追加 |

- マルチプロセッサ対応カーネルにおけるメッセージバッファの制限事項を規定し，使用上の注意を記載

| | |
|---|---|
| NGKI5210〔M〕 | 追加 |
| NGKI5211〔M〕 | 追加 |

- 手動メモリ配置において，ATT_SEC，ATT_MODを使用した場合に，警告メッセージを出力するように変更

| | |
|---|---|
| NGKI3911〔M〕 | 記述の変更 |

| | |
|---|---|
| NGKI3913〔M〕 | 記述の変更 |

- カーネルオブジェクトが属するクラスに関する記述の厳密化

| | |
|---|---|
| NGKI0096〔M〕 | 記述の変更 |

- エラーを報告する条件の記述の厳密化

| | |
|---|---|
| NGKI0471〔M〕 | 記述の変更 |

- 重複する記述の削除

| | |
|---|---|
| NGKI0097〔M〕 | 削除．NGKI0395と重複 |

- 同じ記述に対する重複したタグの削除

| | |
|---|---|
| NGKI0388〔P〕 | 削除．NGKI0459と重複 |
| NGKI0389〔P〕 | 削除．NGKI0460と重複 |
| NGKI0396〔M〕 | 削除．NGKI0470と重複 |

## A.6 Release 3.4.2とRelease 3.5.0の差分

- 拡張情報の型をEXINFに変更

| | |
|---|---|
| NGKI0542 | 記述の追加 |
| NGKI1018 | 記述の変更 |
| NGKI1023 | 記述の変更 |
| NGKI1024 | 記述の変更 |
| NGKI1212 | 記述の変更 |
| NGKI3696 | 記述の変更 |
| NGKI3722 | 記述の変更 |
| NGKI2604 | 記述の変更 |
| NGKI2953 | 記述の変更 |
| NGKI2993 | 記述の変更 |
| NGKI2995 | 記述の変更 |
| NGKI3204 | 記述の変更 |
| NGKI3210 | 記述の変更 |
| NGKI3240 | 記述の変更 |
| NGKI3247 | 記述の変更 |

- 標準スタックリージョンの新設．DEF_SRGの仕様拡張

| | |
|---|---|
| NGKI3261〔P〕 | 記述の変更 |
| NGKI3991〔P〕 | 記述の変更 |

| | |
|---|---|
| NGKI3264〔P〕 | 記述の変更 |
| NGKI5205〔P〕 | 追加 |
| NGKI5149〔P〕 | 記述の変更 |
| NGKI3270〔P〕 | 記述の変更 |

- スタック領域は標準スタックリージョン内に配置するものとする

| | |
|---|---|
| NGKI1051〔P〕 | 記述の変更 |
| NGKI5203〔P〕 | 追加 |
| NGKI5204〔P〕 | 追加 |
| NGKI5137〔P〕 | 記述の変更 |
| NGKI5138〔P〕 | 記述の変更 |

- スタックサイズの指定でE_PARエラーになる条件から，「0を指定した時」を外す

| | |
|---|---|
| NGKI1042 | 記述の変更 |
| NGKI1044〔P〕 | 記述の変更 |
| NGKI3254 | 記述の変更 |
| NGKI3255 | 記述の変更 |

- 割込みサービスルーチンを実行するプロセッサに関する仕様を変更

| | |
|---|---|
| NGKI0138〔M〕 | 記述の変更 |
| NGKI0656〔M〕 | 追加 |

## A.7 Release 3.5.0とRelease 3.6.0の差分

### (1) 仕様書の構成の変更

- 保護ドメイン管理機能の節を，時間パーティショニング機能に変更．保護ドメインのアクセス許可ベクタを設定する機能に関する記述を，システム構成管理機能の節に移動

| | |
|---|---|
| NGKI5001 | 記述の変更 |

- タイムイベントの通知処理に関する記述を独立した節に

### (2) 仕様の変更

- sus_tskとter_tskの呼出しを，システム状態のアクセス許可ベクタでも禁止できるように変更．これに伴って，システム状態に対してアクセス許可ベクタを2つ持つこととし，各サービスコールを制御するアクセス許可パターンを整理

| | |
|---|---|
| NGKI0569〔P〕 | 記述の変更 |
| NGKI0617〔P〕 | 記述の変更 |
| NGKI0426〔P〕 | 記述の変更 |
| NGKI1304〔P〕 | 記述の変更 |

| | |
|---|---|
| NGKI3764〔P〕 | 記述の変更 |
| NGKI3765〔P〕 | 記述の変更 |
| NGKI3508〔P〕 | 記述の変更 |
| NGKI3566〔P〕 | 記述の変更 |
| NGKI2352〔P〕 | 記述の変更 |
| NGKI3585〔P〕 | 記述の変更 |
| NGKI3600〔P〕 | 記述の変更 |
| NGKI2670〔P〕 | 記述の変更 |
| NGKI2671〔PD〕 | 記述の変更 |
| NGKI5216〔P〕 | 追加 |
| NGKI2678〔P〕 | 記述の変更 |
| NGKI2679〔P〕 | 記述の変更 |
| NGKI5217〔P〕 | 追加 |
| NGKI3766〔P〕 | 記述の変更 |
| NGKI3778〔P〕 | 記述の変更 |
| NGKI2729〔P〕 | 記述の変更 |
| NGKI2736〔P〕 | 記述の変更 |
| NGKI2743〔P〕 | 記述の変更 |
| NGKI2749〔P〕 | 記述の変更 |
| NGKI2870〔PD〕 | 記述の変更 |
| NGKI2890〔PD〕 | 記述の変更 |
| NGKI2974〔PD〕 | 記述の変更 |
| NGKI3084〔P〕 | 記述の変更 |
| NGKI3097〔P〕 | 記述の変更 |
| NGKI3922〔P〕 | 記述の変更 |
| NGKI3934〔P〕 | 記述の変更 |
| NGKI3946〔P〕 | 記述の変更 |
| NGKI3110〔P〕 | 記述の変更 |
| NGKI3118〔P〕 | 記述の変更 |
| NGKI3176〔PD〕 | 記述の変更 |
| NGKI5034〔P〕 | 記述の変更 |
| NGKI5063〔P〕 | 記述の変更 |

- 優先度継承ミューテックスを導入

| | |
|---|---|
| NGKI0560 | 記述の変更 |
| NGKI1194 | 適用条件の記述の変更 |

| | |
|---|---|
| NGKI1195 | 適用条件の記述の変更 |
| NGKI1196 | 適用条件の記述の変更 |
| NGKI1197 | 記述の変更 |
| NGKI3673 | 記述の変更 |
| NGKI2010 | 記述の変更 |
| NGKI2011 | 記述の変更 |
| NGKI2014 | 記述の変更 |
| NGKI2015 | 適用条件の記述の変更 |
| NGKI2016 | 適用条件の記述の変更 |
| NGKI2017 | 適用条件の記述の変更 |
| NGKI2018 | 適用条件の記述の変更 |
| NGKI5219 | 追加 |
| NGKI2016 | 記述の変更 |
| ASPS0237 | 追加 |
| ASPS0238 | 追加 |
| FMPS0174 | 追加 |
| HRPS0245 | 追加 |
| HRMPS0121 | 追加 |

- 優先度上限ミューテックスの上限優先度を，サブ優先度を使用するタスク優先度に設定することを禁止する仕様を廃止

| | |
|---|---|
| NGKI3682 | 削除 |

- 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，カーネルドメインからアクセスしようとした場合の振舞いを未規定に（仕様本文から補足説明に移動）

| | |
|---|---|
| NGKI0416〔P〕 | 削除 |

- マルチプロセッサ対応カーネルにおけるCPU例外の扱いを変更

| | |
|---|---|
| NGKI0137〔M〕 | 記述の変更 |
| NGKI0660〔M〕 | 追加 |
| NGKI0308〔M〕 | 記述の変更 |
| NGKI0307〔M〕 | 記述の変更 |
| NGKI0661〔M〕 | 追加 |
| NGKI3146〔M〕 | 記述の変更 |

- 不正なクラスIDのエラーチェックを静的APIでは行わないように変更

| | |
|---|---|
| NGKI1031〔M〕 | エラーの発生条件の変更 |

| | |
|---|---|
| NGKI5111〔MD〕 | 削除 |
| NGKI1458〔M〕 | エラーの発生条件の変更 |
| NGKI5112〔MD〕 | 削除 |
| NGKI1564〔M〕 | エラーの発生条件の変更 |
| NGKI5113〔MD〕 | 削除 |
| NGKI1671〔M〕 | エラーの発生条件の変更 |
| NGKI5114〔MD〕 | 削除 |
| NGKI1806〔M〕 | エラーの発生条件の変更 |
| NGKI5115〔MD〕 | 削除 |
| NGKI2027〔M〕 | エラーの発生条件の変更 |
| NGKI5116〔MD〕 | 削除 |
| NGKI3305〔M〕 | エラーの発生条件の変更 |
| NGKI5117〔MD〕 | 削除 |
| NGKI2134〔M〕 | エラーの発生条件の変更 |
| NGKI5118〔MD〕 | 削除 |
| NGKI2227〔M〕 | エラーの発生条件の変更 |
| NGKI5119〔MD〕 | 削除 |
| NGKI2386〔M〕 | エラーの発生条件の変更 |
| NGKI5120〔MD〕 | 削除 |
| NGKI2494〔M〕 | エラーの発生条件の変更 |
| NGKI5121〔MD〕 | 削除 |
| NGKI5103〔PM〕 | 削除 |
| NGKI5104〔PM〕 | 削除 |
| NGKI5105〔PM〕 | 削除 |
| NGKI2970〔M〕 | エラーの発生条件の変更 |
| NGKI3001〔M〕 | エラーの発生条件の変更 |
| NGKI5122〔MD〕 | 削除 |
| NGKI3053〔M〕 | 削除 |
| NGKI3132〔M〕 | 削除 |
| NGKI5107〔PM〕 | 削除 |
| NGKI5128〔M〕 | 削除 |
| NGKI5161〔MD〕 | 削除 |
| NGKI5164〔M〕 | 削除 |
| NGKI5165〔M〕 | 削除 |

- 割込みサービスルーチンが異なるプロセッサで同時に実行されることはないことに変更

| | |
|---|---|
| NGKI0659〔M〕 | 追加 |

- TA_NOWRITE属性のメモリオブジェクトを，TA_NOWRITE属性でないメモリリージョンに配置した場合に，TA_MEMINI属性を付ける条件を変更

| | |
|---|---|
| NGKI3985〔P〕 | 記述の変更 |

- TA_MEMINI属性のメモリオブジェクトの初期化データの配置と初期化を行わない条件を変更

| | |
|---|---|
| NGKI3984〔P〕 | 記述の変更 |

- ユーザスタック領域に対しては，実行アクセスを禁止することを原則に

| | |
|---|---|
| NGKI0663〔P〕 | 追加 |

(3) 仕様の漏れの修正と明確化

- クラスの識別名がターゲット定義であることを明記

| | |
|---|---|
| NGKI0644〔M〕 | 記述の変更 |

- クラスが持つ属性に関する記述を厳密化

| | |
|---|---|
| NGKI0106〔M〕 | 削除（NGKI0107と統合） |
| NGKI0107〔M〕 | 記述の変更 |
| NGKI0664〔pM〕 | 追加 |

- メモリオブジェクトのアクセス保護に関する記述を厳密化

| | |
|---|---|
| NGKI0408〔P〕 | 記述の変更 |
| NGKI0409〔P〕 | 記述の変更 |
| NGKI0410〔P〕 | 記述の変更 |
| NGKI0412〔P〕 | 記述の変更 |
| NGKI0413〔P〕 | 記述の変更 |

- サービスコールを通じてメモリオブジェクトの境界をまたぐメモリ領域にアクセスを行おうとした場合の振舞いを明記

| | |
|---|---|
| NGKI0673〔P〕 | 追加 |
| NGKI0674〔P〕 | 追加 |

- サービスコールを通じてアラインされていないメモリアクセスを行おうとした場合の振舞いを明記

| | |
|---|---|
| NGKI0670〔P〕 | 追加 |
| NGKI0671〔P〕 | 追加 |

- オブジェクトの管理ブロックなどをコンフィギュレータが確保することを明記（補足説明から仕様本文に移動）

| | |
|---|---|
| NGKI0665 | 追加 |

- 割込み要求フラグの概念を導入し，clr_int，ras_intの振舞いを明確化

| | |
|---|---|
| NGKI0657 | 追加 |
| NGKI0658 | 追加 |
| NGKI3924 | 記述の変更 |
| NGKI3936 | 記述の変更 |

- prb_intの振舞いを明確化

| | |
|---|---|
| NGKI3948 | 削除 |
| NGKI5214 | 追加 |
| NGKI5215 | 追加 |

- prb_memの仕様の記述を厳密化

| | |
|---|---|
| NGKI2933 | 記述の変更 |
| NGKI2939 | 記述の変更 |

- CRE_SOMでのE_OBJエラーの条件を追加

| | |
|---|---|
| NGKI5218〔P〕 | 追加 |

- chg_priで，優先順位が変更されない場合の記述に，待ち行列中での順序が変更されないことを追記

| | |
|---|---|
| NGKI1197 | 記述の変更 |

- 割込みサービスルーチンを，属するクラスの初期割付けプロセッサのみで実行されるものとするのを，ターゲット定義と明記

| | |
|---|---|
| NGKI0656〔M〕 | 記述の変更 |

- カーネルが処理する割込みに，タイムウィンドウタイマ割込みを追加

| | |
|---|---|
| NGKI0248 | 記述の変更 |

- CRE_XXXにおいて，xxxidパラメータの説明を「識別名」とし，他の種類のオブジェクトの識別子との衝突についてもエラーとするように変更

| | |
|---|---|
| NGKI1040 | 記述の変更 |
| NGKI1463 | 記述の変更 |
| NGKI1569 | 記述の変更 |
| NGKI1677 | 記述の変更 |
| NGKI1812 | 記述の変更 |
| NGKI2032 | 記述の変更 |
| NGKI3313 | 記述の変更 |

| | |
|---|---|
| NGKI2139 | 記述の変更 |
| NGKI2236 | 記述の変更 |
| NGKI2392 | 記述の変更 |
| NGKI2500 | 記述の変更 |
| NGKI3009 | 記述の変更 |
| NGKI5218〔P〕 | 記述の変更 |

- グローバル初期化ルーチンとローカル初期化ルーチン，グローバル終了処理ルーチンとローカル終了処理ルーチンの定義方法を明記

| | |
|---|---|
| NGKI5242〔M〕 | 追加 |
| NGKI5243〔M〕 | 追加 |

- 初期化ルーチンと終了処理ルーチンが識別番号を持たないことを明記

| | |
|---|---|
| NGKI5244 | 追加 |
| NGKI5245 | 追加 |

- 拡張サービスコールが機能コードで識別されることを明記

| | |
|---|---|
| NGKI5246 | 追加 |

- コンフィギュレータが警告メッセージを出力することを明記

| | |
|---|---|
| NGKI5039〔P〕 | 記述の変更 |
| NGKI5052〔P〕 | 記述の変更 |

- コンフィギュレータの処理モデルの説明において，Sレコードファイルという用語をダンプファイルに変更し，ダンプファイルとシンボルファイルを生成することを最後に記述するに留める．それに合わせて図も変更
- 重複した記述の削除

| | |
|---|---|
| NGKI0127 | 削除．NGKI0339と重複 |
| NGKI0128 | 削除．NGKI0349と重複 |
| NGKI0130〔M〕 | 削除．NGKI0340，NGKI0341と重複 |
| NGKI0132〔M〕 | 削除．NGKI0350，NGKI0351と重複 |

- 2.10.4節～2.10.6節の記述中の「カーネルオブジェクト」を「オブジェクト」に変更

### (4) タグの付与に関する変更

- 定数値をターゲット定義とする記述にタグを付与

| | |
|---|---|
| NGKI4045 | タグの追加 |
| NGKI4046〔P〕 | タグの追加 |
| NGKI4047 | タグの追加 |

| | |
|---|---|
| NGKI4048 | タグの追加 |

- タグの付与対象として適切でない記述からのタグの削除

| | |
|---|---|
| NGKI0404 | タグの削除 |

- 重複したタグの削除

| | |
|---|---|
| NGKI0496 | タグの削除．NGKI0089と重複 |
| NGKI0502 | タグの削除．NGKI0015と重複 |
| NGKI0503 | タグの削除．NGKI0088と重複 |
| NGKI0525〔M〕 | タグの削除．NGKI0643と重複 |
| NGKI2766〔P〕 | タグの削除．NGKI0112と重複 |
| NGKI2772〔P〕 | タグの削除．NGKI0069と重複 |
| NGKI2774〔P〕 | タグの削除．NGKI0070と重複 |
| NGKI3200 | タグの削除．NGKI0533，NGKI0339と重複 |
| NGKI3206 | タグの削除．NGKI0533，NGKI0349と重複 |
| NGKI5002〔P〕 | タグの削除．NGKI0577と重複 |

# API索引


**A**

acre_alm, 308
acre_cyc, 294
acre_dtq, 179
acre_flg, 165
acre_isr, 389
acre_mbf, 233
acre_mpf, 262
acre_mtx, 216
acre_pdq, 197
acre_sem, 151
acre_spn, 249
acre_tsk, 103
act_tsk, 114
ACV_DOM, 431
adj_tim, 282
AID_ALM, 311
AID_CYC, 297
AID_DTQ, 182
AID_FLG, 167
AID_ISR, 392
AID_MBF, 236
AID_MPF, 267
AID_MTX, 218
AID_PDQ, 200
AID_SEM, 153
AID_SPN, 251
AID_TSK, 109
ATT_INI, 439
ATT_MEM, 370
att_mem, 370
ATT_MOD, 368
ATT_PMA, 373
att_pma, 373
ATT_REG, 362
ATT_SEC, 366
ATT_TER, 440
ATT_TWD, 427

**C**

cal_svc, 419
can_act, 117
can_wup, 135
CFG_INT, 386
cfg_int, 386
chg_ipm, 410
chg_pri, 122
chg_som, 425
chg_spr, 125
clr_flg, 172
clr_int, 403
CRE_ALM, 308
CRE_CYC, 294
CRE_DTQ, 179
CRE_FLG, 165
CRE_ISR, 389
CRE_MBF, 233
CRE_MPF, 262
CRE_MTX, 216
CRE_PDQ, 197
CRE_SEM, 151
CRE_SOM, 424
CRE_SPN, 249
CRE_TSK, 103

**D**

DEF_EPR, 112
DEF_EXC, 413
DEF_ICS, 433
DEF_INH, 397
DEF_MPK, 437
DEF_OVR, 323
DEF_SCY, 423
DEF_SRG, 364
DEF_STK, 435
DEF_SVC, 417
def_svc, 417
del_alm, 314
del_cyc, 300
del_dtq, 185
del_flg, 170
del_isr, 395
del_mbf, 239
del_mpf, 270
del_mtx, 221
del_pdq, 203
del_sem, 156
del_spn, 254
del_tsk, 113
det_mem, 378
dis_dsp, 348
dis_int, 399
dis_ter, 145
dly_tsk, 140

**E**

ena_dsp, 349
ena_int, 401
ENA_SPR, 345
ena_ter, 146
ext_ker, 355
ext_tsk, 142

**F**

fch_hrt, 286
fsnd_dtq, 189

**G**

get_did, 337
get_inf, 127
get_ipm, 411
get_lod, 339
get_mpf, 271
get_nth, 341
get_pid, 338
get_pri, 124
get_som, 426
get_tid, 336
get_tim, 281
get_tst, 120

**I**

ini_dtq, 192
ini_flg, 175
ini_mbf, 244
ini_mpf, 274
ini_mtx, 227
ini_pdq, 209
ini_sem, 160

**L**

loc_cpu, 346
loc_mtx, 223
loc_spn, 255

**M**

mact_tsk, 115
mget_lod, 340
mget_nth, 343
mig_tsk, 118
mrot_rdq, 334
msta_alm, 316
msta_cyc, 302

**P**

pget_mpf, 271
ploc_mtx, 223
pol_flg, 173
pol_sem, 158
prb_int, 407
prb_mem, 379
prcv_dtq, 190
prcv_mbf, 242
prcv_pdq, 207
psnd_dtq, 187
psnd_mbf, 240
psnd_pdq, 205

**R**

ras_int, 405
ras_ter, 143
rcv_dtq, 190
rcv_mbf, 242
rcv_pdq, 207
ref_alm, 319
ref_cfg, 441
ref_cyc, 305
ref_dtq, 193
ref_flg, 176
ref_int, 409
ref_isr, 396
ref_mbf, 245
ref_mem, 381
ref_mpf, 275
ref_mtx, 229
ref_ovr, 327
ref_pdq, 210
ref_sem, 161
ref_spn, 258
ref_sys, 356
ref_tsk, 128
ref_ver, 442
rel_mpf, 273
rel_wai, 136
rot_rdq, 332
rsm_tsk, 139

**S**

SAC_ALM, 312
sac_alm, 312
SAC_CYC, 298
sac_cyc, 298
sac_dom, 431
SAC_DTQ, 183

sac_dtq, 183
SAC_FLG, 168
sac_flg, 168
SAC_ISR, 393
sac_isr, 393
SAC_MBF, 237
sac_mbf, 237
sac_mem, 376
SAC_MPF, 268
sac_mpf, 268
SAC_MTX, 219
sac_mtx, 219
SAC_PDQ, 201
sac_pdq, 201
SAC_SEM, 154
sac_sem, 154
SAC_SPN, 252
sac_spn, 252
SAC_SYS, 330
sac_sys, 330
SAC_TSK, 110
sac_tsk, 110
set_dft, 284
set_flg, 171
set_tim, 280
sig_sem, 157
slp_tsk, 133
snd_dtq, 187
snd_mbf, 240
snd_pdq, 205
sns_ctx, 350
sns_dpn, 353
sns_dsp, 352
sns_ker, 354
sns_loc, 351
sns_ter, 147
sta_alm, 315
sta_cyc, 301
sta_ovr, 324
stp_alm, 318
stp_cyc, 304
stp_ovr, 326
sus_tsk, 137

**T**

ter_tsk, 148
tget_mpf, 271
tloc_mtx, 223
trcv_dtq, 190
trcv_mbf, 242
trcv_pdq, 207
try_spn, 256
tslp_tsk, 133
tsnd_dtq, 187
tsnd_mbf, 240
tsnd_pdq, 205
twai_flg, 173
twai_sem, 158

**U**

unl_cpu, 347
unl_mtx, 225
unl_spn, 257

**W**

wai_flg, 173
wai_sem, 158
wup_tsk, 134

**X**

xsns_dpn, 414