

TOPPERS新世代カーネル統合仕様書

バージョン: Release 1.4.0

最終更新: 2012年5月16日

このドキュメントは、TOPPERS新世代カーネルに属する一連のリアルタイムカーネルの仕様を、統合的に記述するものである。現時点で、TOPPERS/ASPカーネル、TOPPERS/FMPカーネル、TOPPERS/HRP2カーネル、TOPPERS/SSPカーネルの仕様に関しては記述が完成しているが、未完成部分も残っている。特に動的生成対応カーネルについては、仕様検討が不十分なところが多い。なお、本文中から参照している図は、ファイルの最後にまとめて掲載してある。

なお、TOPPERS/SSPカーネルは、Release 1.1.0の時点では、この仕様に準拠していない。今後のリリースでは、この仕様に準拠するよう修正される予定である。

TOPPERS New Generation Kernel Specification

Copyright (C) 2006-2012 by Embedded and Real-Time Systems Laboratory
Graduate School of Information Science, Nagoya Univ., JAPAN
Copyright (C) 2006-2012 by TOPPERS Project, Inc., JAPAN

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

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

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

○目次

第1章 TOPPERS新世代カーネルの概要

51	1.1 TOPPERS新世代カーネル仕様の位置付け
52	1.2 TOPPERS新世代カーネル仕様の設計方針
53	1.3 TOPPERS/ASPカーネルの適用対象領域と仕様設計方針
54	1.4 TOPPERS/FMPカーネルの適用対象領域と仕様設計方針
55	1.5 TOPPERS/HRP2カーネルの適用対象領域と仕様設計方針
56	1.6 TOPPERS/SSPカーネルの適用対象領域と仕様設計方針
57	1.7 TOPPERS/ASP Safetyカーネルの適用対象領域と仕様設計方針
58	
59	第2章 主要な概念と共通定義
60	
61	2.1 仕様の位置付け
62	2.1.1 カーネルの機能セット
63	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
64	2.1.3 想定するソフトウェア構成
65	2.1.4 想定するハードウェア構成
66	2.1.5 想定するプログラミング言語
67	2.2 APIの構成要素とコンベンション
68	2.2.1 APIの構成要素
69	2.2.2 パラメータとリターンパラメータ
70	2.2.3 返値とエラーコード
71	2.2.4 機能コード
72	2.2.5 ヘッダファイル
73	2.3 主な概念
74	2.3.1 オブジェクトと処理単位
75	2.3.2 サービスコールとパラメータ
76	2.3.3 保護機能
77	2.3.4 マルチプロセッサ対応
78	2.3.5 その他
79	2.4 処理単位の種類と実行
80	2.4.1 処理単位の種類
81	2.4.2 処理単位の実行順序
82	2.4.3 カーネル処理の不可分性
83	2.4.4 処理単位を実行するプロセッサ
84	2.5 システム状態とコンテキスト
85	2.5.1 カーネル動作状態と非動作状態
86	2.5.2 タスクコンテキストと非タスクコンテキスト
87	2.5.3 カーネルの振舞いに影響を与える状態
88	2.5.4 全割込みロック状態と全割込みロック解除状態
89	2.5.5 CPUロック状態とCPUロック解除状態
90	2.5.6 割込み優先度マスク
91	2.5.7 ディスパッチ禁止状態とディスパッチ許可状態
92	2.5.8 ディスパッチ保留状態
93	2.5.9 カーネル管理外の状態
94	2.5.10 処理単位の開始・終了とシステム状態
95	2.6 タスクの状態遷移とスケジューリング規則
96	2.6.1 基本的なタスク状態
97	2.6.2 タスクの状態遷移
98	2.6.3 タスクのスケジューリング規則
99	2.6.4 待ち行列と待ち解除の順序
100	2.6.5 タスク例外処理マスク状態と待ち禁止状態

101	2.6.6 ディスパッチ保留状態で実行中のタスクに対する強制待ち
102	2.6.7 制約タスク
103	2.7 割込み処理モデル
104	2.7.1 割込み処理の流れ
105	2.7.2 割込み優先度
106	2.7.3 割込み要求ラインの属性
107	2.7.4 割込みを受け付ける条件
108	2.7.5 割込み番号と割込みハンドラ番号
109	2.7.6 マルチプロセッサにおける割込み処理
110	2.7.7 カーネル管理外の割込み
111	2.7.8 カーネル管理外の割込みの設定方法
112	2.8 CPU例外処理モデル
113	2.8.1 CPU例外処理の流れ
114	2.8.2 CPU例外ハンドラから呼び出せるサービスコール
115	2.8.3 エミュレートされたCPU例外ハンドラ
116	2.8.4 カーネル管理外のCPU例外
117	2.9 システムの初期化と終了
118	2.9.1 システム初期化手順
119	2.9.2 システム終了手順
120	2.10 オブジェクトの登録とその解除
121	2.10.1 ID番号で識別するオブジェクト
122	2.10.2 オブジェクト番号で識別するオブジェクト
123	2.10.3 識別番号を持たないオブジェクト
124	2.10.4 オブジェクト生成に必要なメモリ領域
125	2.10.5 オブジェクトが属する保護ドメインの設定
126	2.10.6 オブジェクトが属するクラスの設定
127	2.10.7 オブジェクトの状態参照
128	2.11 オブジェクトのアクセス保護
129	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
130	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
131	2.11.3 デフォルトのアクセス許可ベクタ
132	2.11.4 アクセス許可ベクタの設定
133	2.11.5 カーネルの管理領域のアクセス保護
134	2.11.6 ユーザタスクのユーザスタック領域
135	2.12 システムコンフィギュレーション手順
136	2.12.1 システムコンフィギュレーションファイル
137	2.12.2 静的APIの文法とパラメータ
138	2.12.3 保護ドメインの指定
139	2.12.4 クラスの指定
140	2.12.5 コンフィギュレータの処理モデル
141	2.12.6 静的APIのパラメータに関するエラー検出
142	2.12.7 オブジェクトのID番号の指定
143	2.13 TOPPERSネーミングコンベンション
144	2.13.1 モジュール識別名
145	2.13.2 データ型名
146	2.13.3 関数名
147	2.13.4 変数名
148	2.13.5 定数名
149	2.13.6 マクロ名
150	2.13.7 静的API名

151	2.13.8 ファイル名
152	2.13.9 モジュール内部の名称の衝突回避
153	2.14 TOPPERS共通定義
154	2.14.1 TOPPERS共通ヘッダファイル
155	2.14.2 TOPPERS共通データ型
156	2.14.3 TOPPERS共通定数
157	2.14.4 TOPPERS共通エラーコード
158	2.14.5 TOPPERS共通マクロ
159	2.14.6 TOPPERS共通構成マクロ
160	2.15 カーネル共通定義
161	2.15.1 カーネルヘッダファイル
162	2.15.2 カーネル共通定数
163	2.15.3 カーネル共通マクロ
164	2.15.4 カーネル共通構成マクロ
165	
166	第3章 システムインタフェースレイヤAPI仕様
167	
168	3.1 システムインタフェースレイヤの概要
169	3.2 SILヘッダファイル
170	3.3 全割込みロック状態の制御
171	3.4 SILスピンロック
172	3.5 微少時間待ち
173	3.6 エンディアンの取得
174	3.7 メモリ空間アクセス関数
175	3.8 I/O空間アクセス関数
176	3.9 プロセッサIDの参照
177	
178	第4章 カーネルAPI仕様
179	
180	4.1 タスク管理機能
181	4.2 タスク付属同期機能
182	4.3 タスク例外処理機能
183	4.4 同期・通信機能
184	4.4.1 セマフォ
185	4.4.2 イベントフラグ
186	4.4.3 データキュー
187	4.4.4 優先度データキュー
188	4.4.5 メールボックス
189	4.4.6 ミューテックス
190	4.4.7 メッセージバッファ (☆未完成)
191	4.4.8 スピンロック
192	4.5 メモリプール管理機能
193	4.5.1 固定長メモリプール
194	4.6 時間管理機能
195	4.6.1 システム時刻管理
196	4.6.2 周期ハンドラ
197	4.6.3 アラームハンドラ
198	4.6.4 オーバランハンドラ
199	4.7 システム状態管理機能
200	4.8 メモリオブジェクト管理機能

201	4.9 割込み管理機能
202	4.10 CPU例外管理機能
203	4.11 拡張サービスコール管理機能
204	4.12 システム構成管理機能
205	
206	第5章 リファレンス
207	
208	5.1 サービスコール一覧
209	5.2 静的API一覧
210	5.3 データ型
211	5.3.1 TOPPERS共通データ型
212	5.3.2 カーネルの使用データ型
213	5.3.3 カーネルの使用データ形式
214	5.4 定数とマクロ
215	5.4.1 TOPPERS共通定数
216	5.4.2 TOPPERS共通マクロ
217	5.4.3 カーネル共通定数
218	5.4.4 カーネル共通マクロ
219	5.4.5 カーネルの機能毎の定数
220	5.4.6 カーネルの機能毎のマクロ
221	5.5 構成マクロ
222	5.5.1 TOPPERS共通構成マクロ
223	5.5.2 カーネル共通構成マクロ
224	5.5.3 カーネルの機能毎の構成マクロ
225	5.6 エラーコード一覧
226	5.7 機能コード一覧
227	5.8 カーネルオブジェクトに対するアクセスの種別
228	5.9 省略名の元になった英語
229	5.9.1 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
230	5.9.2 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
231	5.9.3 サービスコールの名称の中のzの元になった英語
232	5.10 バージョン履歴
233	
234	
235	第1章 TOPPERS新世代カーネルの概要
236	
237	TOPPERS新世代カーネルとは、TOPPERSプロジェクトにおいてITRON仕様をベース
238	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、
239	TOPPERS新世代カーネル仕様の位置付けと設計方針、それに属する各カーネルの
240	適用対象領域と設計方針について述べる。
241	
242	1.1 TOPPERS新世代カーネル仕様の位置付け
243	
244	TOPPERSプロジェクトでは、2000年に公開したTOPPERS/JSPカーネルを始めとし
245	て、 μ ITRON4.0仕様およびその保護機能拡張 (μ ITRON4.0/PX仕様) に準拠した
246	リアルタイムカーネルを開発してきた。
247	
248	μ ITRON4.0仕様は1999年に、 μ ITRON4.0/PX仕様は2002年に公表されたが、それ
249	以降現在までの間に、大きな仕様改訂は実施されていない。その間に、組込み
250	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いる。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、新しい要求が生じている。

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、 μ ITRON4.0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。

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

1.2 TOPPERS新世代カーネル仕様の設計方針

TOPPERS新世代カーネル仕様を設計するにあたり、次の方針を設定する。

(1) μ ITRON4.0仕様をベースに拡張・改良を加える

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

(2) ソフトウェアの再利用性を重視する

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

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

(3) 高信頼・安全なシステム構築を支援する

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

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

(4) アプリケーションシステム構築に必要な機能は積極的に取り込む

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

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

1.3 TOPPERS/ASPカーネルの適用対象領域と仕様設計方針

TOPPERS/ASPカーネル（ASPは、Advanced Standard Profileの略。以下、ASPカーネル）は、TOPPERS新世代カーネルの出発点となるリアルタイムカーネルである。保護機能を持ったカーネルやマルチプロセッサ対応のカーネルは、ASPカーネルを拡張する形で開発する。

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

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

ASPカーネルの機能は、カーネル内で動的なメモリ管理が不要な範囲に留める。これは、高い信頼性・安全性・リアルタイム性を要求される組み込みシステムでは、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。

1.4 TOPPERS/FMPカーネルの適用対象領域と仕様設計方針

TOPPERS/FMPカーネル（FMPは、Flexible Multiprocessor Profileの略。以下、FMPカーネル）は、ASPカーネルを、マルチプロセッサ対応に拡張したリアルタイムカーネルである。

FMP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。

FMP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

351 で動的な負荷分散を実現することが可能である。

352

353 FMPカーネルの機能は、ASPカーネルと同様に、カーネル内で動的なメモリ管理
354 が不要な範囲に留める。

355

356 1.5 TOPPERS/HRP2カーネルの適用対象領域と仕様設計方針

357

358 TOPPERS/HRP2カーネル（HRPは、High Reliable system Profileの略。2はバー
359 ジョン番号を示す。以下、HRP2カーネル）は、さらに高い信頼性・安全性を要
360 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
361 うに、ASPカーネルを拡張したリアルタイムカーネルである。

362

363 HRP2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
364 権モードを備え、メモリ保護のためにMMU（Memory Management Unit）または
365 MPU（Memory Protection Unit）を持つプロセッサを用いたシステムである。
366 HRP2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
367 ズ（バイナリコード）が数百KB以上のシステムである。

368

369 HRP2カーネルの機能は、ASPカーネルと同様に、カーネル内で動的なメモリ管理
370 が不要な範囲に留める。具体的には、ASPカーネルに対して、メモリ保護機能と
371 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、ミューテックス機
372 能、オーバランハンドラ機能を追加し、メールボックス機能を削除している。

373

374 1.6 TOPPERS/SSPカーネルの適用対象領域と仕様設計方針

375

376 TOPPERS/SSPカーネル（SSPは、Smallest Set Profileの略。以下、SSPカーネル）
377 は、小規模システムに用いるために、ASPカーネルをベースに可能な限り機能を
378 絞り込んだリアルタイムカーネルである。

379

380 SSPカーネルの機能は、 μ ITRON4.0仕様の「仕様準拠の最低条件」の考え方を踏
381 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、SSPカーネル
382 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
383 サポートする）のが最大の特徴である。また、ASPカーネルに対して下位互換性
384 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
385 に有用な機能は、ASPカーネルに対して追加している。

386

387 TOPPERS/SSP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）
388 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。

389

390 1.7 TOPPERS/ASP Safetyカーネルの適用対象領域と仕様設計方針

391

392 TOPPERS/ASP Safetyカーネル（以下、ASP Safetyカーネル）は、小規模な安全
393 関連システムに用いるために、ASPカーネルの機能を徹底的な検証が可能な範囲
394 にサブセット化したものである。メールボックスのように安全性の観点から問
395 題のある機能や、タスク例外処理機能のように使用頻度に比べて検証にコスト
396 のかかる機能はサポートしない。

397

398 ASP Safetyカーネルの主な適用対象は、特に高い安全性を要求される組込みシ
399 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
400 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

401 システムには、保護機能を持ったカーネルを適用すべきと考えられる。
402
403

404 第2章 主要な概念と共通定義

406 2.1 仕様の位置付け

407
408 この仕様は、TOPPERS新世代カーネルに属する各カーネルの仕様を、統合的に記
409 述することを目指している。また、TOPPERS新世代カーネル上で動作する各種
410 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。
411

412 2.1.1 カーネルの機能セット

413
414 TOPPERS新世代カーネルは、ASPカーネルをベースとして、保護機能、マルチプ
415 ロセッサ、カーネルオブジェクトの動的生成、機能安全などに対応した一連の
416 カーネルで構成される。
417

418 この仕様では、TOPPERS新世代カーネルを構成する一連のカーネルの仕様を統合
419 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
420 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
421 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
422 正するのは得策ではない。
423

424 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
425 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
426 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
427 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
428 ネルを動的生成対応カーネルと呼ぶことにする。
429

430 【TOPPERS/ASPカーネルにおける規定】

431
432 ASPカーネルは、保護機能対応カーネル、マルチプロセッサ対応カーネル、動的
433 生成対応カーネルのいずれでもない。ただし、動的生成機能拡張パッケージを
434 用いると、動的生成対応カーネルとなる。
435

436 【TOPPERS/FMPカーネルにおける規定】

437
438 FMPカーネルは、マルチプロセッサ対応カーネルであり、保護機能対応カーネル、
439 動的生成対応カーネルではない。
440

441 【TOPPERS/HRP2カーネルにおける規定】

442
443 HRP2カーネルは、保護機能対応カーネルであり、マルチプロセッサ対応カー
444 ネル、動的生成対応カーネルではない。
445

446 【TOPPERS/SSPカーネルにおける規定】

447
448 SSPカーネルは、保護機能対応カーネル、マルチプロセッサ対応カーネル、動的
449 生成対応カーネルのいずれでもない。
450

【 μ ITRON4.0仕様, μ ITRON4.0/PX仕様との関係】

μ ITRON4.0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。 μ ITRON4.0/PX仕様は、 μ ITRON4.0仕様に対して保護機能を追加するための仕様であり、カーネルオブジェクトの動的生成機能と保護機能を持っているが、マルチプロセッサには対応していない。

2.1.2 ターゲット非依存の規定とターゲット定義の規定

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

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

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

2.1.3 想定するソフトウェア構成

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

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

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

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

501 する.

502

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

506

507 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、
508 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、
509 システムサービス毎の仕様書で規定される.

510

511 【 μ ITRON4.0仕様との関係】

512

513 μ ITRON4.0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
514 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
515 (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
516 仕様ではシステムサービスと呼ぶことにした.

517

518 2.1.4 想定するハードウェア構成

519

520 この仕様では、カーネルがサポートするハードウェア構成として、以下のこと
521 を想定している．これらに合致しないターゲットハードウェアでカーネルを動
522 作させることは可能であるが、合致しない部分への適応はアプリケーションの
523 責任になる.

524

525 (a) 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
526 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）．マルチプロセッ
527 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
528 の番地でアクセスできること.

529

530 (b) 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
531 命令を実行できること.

532

533 2.1.5 想定するプログラミング言語

534

535 この仕様におけるAPI仕様は、ISO/IEC 9899:1990（以下、C90と呼ぶ）または
536 ISO/IEC 9899:1999（以下、C99と呼ぶ）に準拠したC言語を、フリースタンドイ
537 ング環境で用いることを想定して規定している.

538

539 ただし、C90の規定に加えて、以下のことを仮定している.

540

- 541 ・16ビットおよび32ビットの整数型があること
- 542 ・ポインタが格納できるサイズの整数型があること

543

544 2.2 APIの構成要素とコンベンション

545

546 2.2.1 APIの構成要素

547

548 (1) サービスコール

549

550 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

551 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、
552 システムコール (system call) と呼ぶ場合もある。

553

554 (2) コールバック

555

556 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
557 スをコールバック (callback) と呼ぶ。

558

559 (3) 静的API

560

561 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
562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
563 API) と呼ぶ。

564

565 (4) 構成マクロ

566

567 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
568フトウェアが用いるマクロを、構成マクロ (configuration macro) と呼ぶ。

569

570 2.2.2 パラメータとリターンパラメータ

571

572 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
573 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
574 的APIに渡すデータもパラメータと呼ぶ。

575

576 オブジェクトを生成するサービスコールなど、パラメータの数が多い場合やター
577 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
578 を1つの構造体に入れ、その領域へのポインタをパラメータとして渡す。また、
579 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
580 をパラメータとして渡す場合がある。

581

582 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
583 タを入れる領域へのポインタをパラメータとして渡すことで実現する。オブジェ
584 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
585 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、
586 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
587 ポインタをパラメータとして渡す。

588

589 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
590 ト (packet) と呼ぶ。

591

592 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
593 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、
594 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
595 れることはなく、別の目的に使用できる。

596

597 2.2.3 返値とエラーコード

598

599 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
600 常終了したかを表す符号付き整数とする。処理が正常終了した場合には、E_OK

(=0) または正の値が返るものとし、値の意味はサービスコールまたはコールバック毎に定める。処理が正常終了しなかった場合には、その原因を表す負の値が返る。処理が正常終了しなかった原因を表す値を、エラーコード (error code) と呼ぶ。

エラーコードは、いずれも負の値のメインエラーコードとサブエラーコードで構成される。メインエラーコードとサブエラーコードからエラーコードを構成するマクロ (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。

メインエラーコードの名称・意味・値は、カーネルとシステムサービスで共通に定める (「2.14.4 TOPPERS共通エラーコード」の節を参照)。サービスコールおよびコールバックの機能説明中の「E_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。サブエラーコードの名称・意味・値は、サブエラーコードを使用するシステムサービスのAPI仕様において規定する。

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

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

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

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

2.2.4 機能コード

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

2.2.5 ヘッダファイル

カーネルやシステムサービスを用いるために必要な定義を含むファイル。

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。

2.3 主な概念

2.3.1 オブジェクトと処理単位

(1) オブジェクト

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

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

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

(2) 処理単位

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

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

拡張情報 (extended information) は、処理単位が呼び出される時にパラメー
タとして渡される情報で、処理単位の登録時に指定する。拡張情報は、カーネ
ルやシステムサービスの動作には影響しない。

(3) タスク

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

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

【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。

(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。

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

【 μ ITRON4.0仕様との関係】

μ ITRON4.0仕様において、未定義のまま使われていた割込みとCPU例外という用語を定義した。

(6) タイムイベントとタイムイベントハンドラ

時間の経過をきっかけに発生するイベントをタイムイベント (time event) と

751 呼ぶ。タイムイベントにより起動され、カーネルが実行制御する処理単位を、
752 タイムイベントハンドラ (time event handler) と呼ぶ。

753

754 2.3.2 サービスコールとパラメータ

755

756 (1) 優先順位と優先度

757

758 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
759 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
760 高い処理単位が実行される。

761

762 優先度 (priority) は、タスクなどの処理単位の優先順位や、メッセージなど
763 の配送順序を決定するために、アプリケーションが処理単位やメッセージなど
764 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
765 続した正の値を用いるのを原則とする。優先度は、値が小さいほど優先度が高
766 い (すなわち、先に実行または配送される) ものとする。

767

768 (2) システム時刻と相対時間

769

770 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
771 刻は、符号無しの整数型であるSYSTIM型で表し、単位はミリ秒とする。システ
772 ム時刻は、タイムティック (time tick) を通知するためのタイマ割込みが発生
773 する毎に更新される。

774

775 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
776 相対時間 (relative time) によって指定する。基準時刻は、別に規定がない限
777 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。

778

779 相対時間は、符号無しの整数型であるRELTIM型で表し、単位はシステム時刻と
780 同一、すなわちミリ秒とする。相対時間には、少なくとも、16ビットの符号無
781 しの整数型 (uint16_t型) に格納できる任意の値を指定することができるが、
782 RELTIM型 (uint_t型に定義される) に格納できる任意の値を指定できるとは限
783 らない。相対時間に指定できる最大値は、構成マクロTMAX_RELTIMに定義されて
784 いる。

785

786 イベントを発生させる時刻を相対時間で指定した場合、イベントの処理が行わ
787 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
788 なる。ただし、基準時刻を定めるサービスコールを呼び出した時に、タイム
789 ティックを通知するためのタイマ割込みがマスクされている場合 (タイマ割込
790 みより優先して実行される割込み処理が実行されている場合を含む) は、相対
791 時間によって指定した以上の時間が経過した後となることは保証されない。

792

793 イベントが発生する時刻を参照する場合には、基準時刻からの相対時間として
794 返される。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
795 る。

796

797 イベントが発生する時刻が相対時間で返された場合、イベントの処理が行われ
798 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。
799 ただし、相対時間を返すサービスコールを呼び出した時に、タイムティックを
800 通知するためのタイマ割込みがマスクされている場合 (タイマ割込みより優先

して実行される割込み処理が実行されている場合を含む) は、相対時間として返された以上の時間が経過した後となることは保証されない。

【補足説明】

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

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

【 μ ITRON4.0仕様との関係】

相対時間 (RELTIM型) とシステム時刻 (SYSTIM型) の時間単位は、 μ ITRON4.0仕様では実装定義としていたが、この仕様ではミリ秒と規定した。また、相対時間の解釈について、より厳密に規定した。

TMAX_RELTIMは、 μ ITRON4.0仕様に規定されていないカーネル構成マクロである。

(3) タイムアウトとポーリング

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

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

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

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

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

【補足説明】

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

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、この仕様ではミリ秒と規定した。

【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、ディスパッチ保留状態では、別に規定がない限り、自タスクを広義の待ち状態に遷移させる可能性のあるサービスコール（タイムアウト付きのサービスコールはこれに該当）を呼び出すことはできないと規定されているためである。

(4) ノンブロッキング

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

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

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

【補足説明】

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

2.3.3 保護機能

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

(1) アクセス保護

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

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

(2) メモリオブジェクト

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

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

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。

(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。

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。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。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、「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」という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。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。

(4) カーネルドメインとユーザドメイン

システムには、カーネルドメイン (kernel domain) と呼ばれる保護ドメインが1つ存在する。カーネルドメインに属する処理単位は、プロセッサの特権モードで実行される。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。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。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。

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。カーネルドメインの保護ドメインIDは、TDOM_KERNEL (=-1) である。

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。これを超える数のユーザドメインに登録した場合には、コンフィギュレータがエラーを報告する。

【補足説明】

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

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

【μ ITRON4.0/PX仕様との関係】

μ ITRON4.0/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。

1001 (5) システムタスクとユーザタスク

1002

1003 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
1004 メインに属するタスクをユーザタスク (user task) と呼ぶ.

1005

1006 【補足説明】

1007

1008 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
1009 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
1010 ビスコールの実行中は特権モードで実行されるため, 上記の定義とした.

1011

1012 μ ITRON4.0/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
1013 ぶことになる.

1014

1015 (6) アクセス許可パターン

1016

1017 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
1018 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
1019 許可パターン (access permission pattern) と呼ぶ. アクセス許可パターンの
1020 各ビットは, 1つのユーザドメインに対応する. カーネルドメインには, すべて
1021 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
1022 されていない.

1023

1024 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

1025 (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
1026 可されていることを表す. そのため, 2つのアクセス許可パターンのビット毎論
1027 理和 (C言語の" \mid ") 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
1028 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
1029 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
1030 ユーザドメインの積集合 (intersection) を得ることができる.

1031

1032 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
1033 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)
1034 が用意されている. 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
1035 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
1036 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
1037 (TACP_SHARED) が用意されている.

1038

1039 (7) アクセス許可ベクタ

1040

1041 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,
1042 通常操作1, 通常操作2, 管理操作, 参照操作の4つの種別に分類されている. あ
1043 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
1044 ターンをひとまとめたものを, アクセス許可ベクタ (access permission
1045 vector) と呼び, 次のように定義されるデータ型 (ACVCT) で保持する.

1046

```
1047 typedef struct acvct {
```

```
1048     ACPTN  acptn1;    /* 通常操作1のアクセス許可パターン */
```

```
1049     ACPTN  acptn2;    /* 通常操作2のアクセス許可パターン */
```

```
1050     ACPTN  acptn3;    /* 管理操作のアクセス許可パターン */
```

```
1051          ACPTN    acptn4;    /* 参照操作のアクセス許可パターン */
1052      } ACVCT;
```

1054 【補足説明】

1055
1056 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
1057 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。

1059 【 μ ITRON4.0/PX仕様との関係】

1060
1061 μ ITRON4.0/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
1062 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
1063 めている。

1065 (8) サービスコールの呼出し方法

1066
1067 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
1068 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、
1069 ソフトウェア割込みによって呼び出すコードが生成される。

1070
1071 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ーヘッドが大
1072 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
1073 てサービスコールを呼び出すことで、オーバーヘッドを削減することができる。
1074 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
1075 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。

1076
1077 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
1078 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、
1079 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
1080 法で記述した場合に、関数呼出しによって呼び出すコードが生成される。

1081
1082 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
1083 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
1084 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)
1085 を用いることで、関数呼出しによって呼び出すコードが生成される。例えば、
1086 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。

```
1088     ercd = SVC_CALL(act_tsk)(tskid);
```

1090 【補足説明】

1091
1092 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。
1093 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
1094 ルとして登録した関数を呼び出すことはできるが、その場合には、処理単位が
1095 呼び出した通常の間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
1096 れない。

1098 2.3.4 マルチプロセッサ対応

1099
1100 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

1101 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。

1102

1103 (1) クラス

1104

1105 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
1106 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。

1107

1108 カーネルオブジェクトは、いずれか1つのクラスに属するのが原則である。カー
1109 ネルオブジェクトが属するクラスは、オブジェクトの登録時に決定し、登録後
1110 に変更することはできない。

1111

1112 【補足説明】

1113

1114 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
1115 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
1116 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
1117 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
1118 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
1119 が典型的である。

1120

1121 カーネルオブジェクトはいずれか1つのクラスに属するという原則に関わらず、
1122 以下のオブジェクトはいずれのクラスにも属さない。

1123

- 1124 ・ オーバランハンドラ
- 1125 ・ 拡張サービスコール
- 1126 ・ グローバル初期化ルーチン
- 1127 ・ グローバル終了処理ルーチン

1128

1129 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
1130 ラスのみをサポートしているとみなすこともできる。

1131

1132 (2) プロセッサ

1133

1134 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
1135 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
1136 別する。

1137

1138 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)
1139 と呼び、同時に複数の処理単位を実行することができる。

1140

1141 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
1142 セッサ (master processor) と呼び、システムに1つ存在する。どのプロセッサ
1143 をマスタプロセッサとするかは、ターゲット定義である。マスタプロセッサ以
1144 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
1145 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
1146 ない。

1147

1148 (3) 処理単位の割付けとマイグレーション

1149

1150 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

1151 セッサに割り付けられて実行される。処理単位を実行するプロセッサを、割付
1152 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、
1153 初期割付けプロセッサと呼ぶ。

1154

1155 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
1156 が可能である。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
1157 単位のマイグレーション (migration) と呼ぶ。

1158

1159 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
1160 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
1161 とができる。

1162

1163 (4) クラスの持つ属性とカーネルオブジェクト

1164

1165 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
1166 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
1167 ブジェクトが属するクラスによって定まる。

1168

1169 各クラスが持ち、それに属するカーネルオブジェクトに適用される属性は、次
1170 の通りである。

1171

- 1172 ・初期割付けプロセッサ
- 1173 ・割付け可能プロセッサ (複数のプロセッサを指定可能、初期割付けプロセッ
1174 サを含む)
- 1175 ・ATT_MOD/ATA_MODにおいて、オブジェクトモジュール中に含まれる標準の
1176 セクションが配置されるメモリリージョン
- 1177 ・オブジェクト生成に必要なメモリ領域 (オブジェクトの管理ブロック、タ
1178 スクのスタック領域やデータキューのデータキュー管理領域など) の配置
1179 場所
- 1180 ・その他の管理情報 (ロック単位など)

1181

1182 使用できるクラスのID番号とその属性は、ターゲット定義である。

1183

1184 【仕様決定の理由】

1185

1186 クラスを導入することで、カーネルオブジェクト毎に上記の属性を設定できる
1187 ようにできなかったのは、これらの属性をアプリケーション設計者が個別に設定
1188 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用意
1189 しておく方が良いと考えたためである。

1190

1191 (5) ローカルタイマ方式とグローバルタイマ方式

1192

1193 システム時刻の管理方式として、プロセッサ毎にシステム時刻を持つローカル
1194 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
1195 の2つの方式がある。どちらの方式を用いることができるかは、ターゲット定義
1196 である。

1197

1198 ローカルタイマ方式では、プロセッサ毎のシステム時刻は、それぞれのプロセッ
1199 サが更新する。異なるプロセッサのシステム時刻を同期させる機能は、カーネ
1200 ルでは用意しない。

1201
1202 グローバルタイマ方式では、システム中の1つのプロセッサがシステム時刻を更
1203 新する。これを、システム時刻管理プロセッサと呼ぶ。どのプロセッサをシス
1204 テム時刻管理プロセッサとするかは、ターゲット定義である。

1205

1206 **【補足説明】**

1207

1208 システム時刻管理プロセッサが、マスタプロセッサと一致している必要はない。

1209

1210 **【未決定事項】**

1211

1212 ローカルタイマ方式の場合に、プロセッサ毎に異なるタイムティックの周期を
1213 設定したい場合が考えられるが、現時点の実装ではサポートしておらず、
1214 TIC_NUMEとTIC_DEN0の扱いも未決定であるため、今後の課題とする。

1215

1216 2.3.5 その他

1217

1218 (1) オブジェクトモジュール

1219

1220 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
1221 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
1222 ブジェクトモジュールである。

1223

1224 (2) メモリリージョン

1225

1226 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
1227 た連続したメモリ領域をメモリリージョン (memory region) と呼ぶ。

1228

1229 メモリリージョンは、文字列によって識別する。メモリリージョンを識別する
1230 文字列を、メモリリージョン名と呼ぶ。

1231

1232 **【補足説明】**

1233

1234 この仕様では、メモリ領域 (memory area) という用語は、連続したメモリの範
1235 囲という一般的な意味で使っている。

1236

1237 (3) 標準のセクション

1238

1239 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
1240 ン (standard sections) と呼ぶ。また、ターゲット定義で、コンパイラが出力
1241 しないセクションを、標準のセクションと扱う場合もある。

1242

1243 (4) 保護ドメイン毎の標準セクション

1244

1245 保護機能対応カーネルにおいては、保護ドメイン毎に、標準のセクションを配
1246 置するためのセクションが登録される。また、無所属の標準のセクションを配
1247 置するためのセクションが登録される。これらのセクションを、保護ドメイン
1248 毎の標準セクションと呼ぶ (standard sections for each protection domain)。
1249 保護ドメイン毎の標準セクションのセクション名は、ターゲット定義で別に規
1250 定がない限りは、標準のセクション名と保護ドメイン名 (カーネルドメインの

1251 場合は“kernel”，無所属の場合は“shared”）を“_”でつないだものとする．例え
1252 ば，カーネルドメインの“.text”セクションのセクション名は，“.text_kernel”
1253 とする．

1254

1255 2.4 処理単位の種類と実行順序

1256

1257 2.4.1 処理単位の種類

1258

1259 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．

1260

1261 (a) タスク

1262 (a.1) タスク例外処理ルーチン

1263 (b) 割込みハンドラ

1264 (b.1) 割込みサービスルーチン

1265 (b.2) タイムイベントハンドラ

1266 (c) CPU例外ハンドラ

1267 (d) 拡張サービスコール

1268 (e) 初期化ルーチン

1269 (f) 終了処理ルーチン

1270

1271 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
1272 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
1273 ある．

1274

1275 【TOPPERS/ASPカーネルにおける規定】

1276

1277 ASPカーネルでは，オーバランハンドラと拡張サービスコールをサポートしてい
1278 ない．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
1279 ンハンドラ機能を追加することができる．

1280

1281 【TOPPERS/FMPカーネルにおける規定】

1282

1283 FMPカーネルでは，オーバランハンドラと拡張サービスコールをサポートしてい
1284 ない．

1285

1286 【TOPPERS/SSPカーネルにおける規定】

1287

1288 SSPカーネルでは，タスク例外処理ルーチン，タイムイベントハンドラ，拡張サー
1289 ビスコールをサポートしていない．

1290

1291 2.4.2 処理単位の実行順序

1292

1293 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
1294 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
1295 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
1296 る．

1297

1298 タスクの優先順位は，ディスパッチャの優先順位よりも低い．タスク間では，
1299 高い優先度を持つ方が優先順位が高く，同じ優先度を持つタスク間では，先に
1300 実行できる状態となった方が優先順位が高い．詳しくは，「2.6.3 タスクのス

1301 ケジューリング規則」の節を参照すること。
1302
1303 タスク例外処理ルーチンの優先順位は、例外が要求されたタスクと同じである
1304 が、タスクよりも先に実行される。
1305
1306 割込みハンドラの優先順位は、ディスパッチャの優先順位よりも高い。割込み
1307 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
1308 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。
1309 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
1310 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。
1311
1312 割込みサービ斯拉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
1313 出す割込みハンドラと同じである。
1314
1315 CPU例外ハンドラの優先順位は、CPU例外がタスクまたはタスク例外処理ルー
1316 チンで発生した場合には、ディスパッチャの優先順位と同じであるが、ディスパッ
1317 チャよりも先に実行される。CPU例外がその他の処理単位で発生した場合には、
1318 CPU例外ハンドラの優先順位は、その処理単位の優先順位と同じであるが、その
1319 処理単位よりも先に実行される。
1320
1321 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、
1322 それを呼び出した処理単位よりも先に実行される。
1323
1324 初期化ルーチンは、カーネルの動作開始前に、システムコンフィギュレーショ
1325 ンファイル中に初期化ルーチンを登録する静的APIを記述したのと同じ順序で実
1326 行される。終了処理ルーチンは、カーネルの動作終了後に、終了処理ルーチン
1327 を登録する静的APIを記述したのと逆の順序で実行される。
1328
1329 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
1330 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。
1331 グローバル初期化ルーチンがマスタプロセッサで実行された後に、各プロセッ
1332 サでローカル初期化ルーチンが実行される。また、終了処理ルーチンには、ク
1333 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
1334 処理ルーチンがある。ローカル終了処理ルーチンが各プロセッサで実行された
1335 後に、マスタプロセッサでグローバル終了処理ルーチンが実行される。
1336
1337 **【仕様決定の理由】**
1338
1339 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
1340 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
1341 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
1342 順に記述するのは難しい）。
1343
1344 2.4.3 カーネル処理の不可分性
1345
1346 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
1347 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
1348 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
1349 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、
1350 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。こ

1351 れを、カーネル処理の不可分性という。

1352

1353 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
1354 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
1355 ける場合がある。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
1356 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
1357 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。

1358

1359 【補足説明】

1360

1361 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
1362 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
1363 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
1364 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
1365 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
1366 れた後に、タスクCへのディスパッチが行われる。

1367

1368 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
1369 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
1370 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
1371 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
1372 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
1373 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
1374 がある。

1375

1376 2.4.4 処理単位を実行するプロセッサ

1377

1378 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
1379 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
1380 け可能プロセッサから、次のように決まる。

1381

1382 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
1383 割付けプロセッサに割り付けられる。また、割付けプロセッサを変更するサー
1384 ビスコール（mact_tsk/imact_tsk, mig_tsk, msta_cyc, msta_alm/
1385 imsta_alm）によって、割付けプロセッサを、クラスの割付け可能プロセッサの
1386 いずれかに変更することができる。

1387

1388 割込みハンドラ、CPU例外ハンドラ、ローカル初期化ルーチン、ローカル終了処
1389 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。クラスの
1390 割付け可能プロセッサの情報は用いられない。

1391

1392 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
1393 （オプション設定によりすべて）で実行される。クラスの初期割付けプロセッ
1394 サの情報は用いられない。

1395

1396 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
1397 使用されることを、「—」はその情報が使用されないことを示す。

1398

1399 初期割付けプロセッサ 割付け可能プロセッサ

1400

1401	タスク（タスク例外処理	○	○
1402	ルーチンを含む）		
1403	-----		
1404	割込みハンドラ	○	—
1405	割込みサービスルーチン	—	○
1406	周期ハンドラ	○	○
1407	アラームハンドラ	○	○
1408	-----		
1409	CPU例外ハンドラ	○	—
1410	-----		
1411	ローカル初期化ルーチン	○	—
1412	ローカル終了処理ルーチン	○	—
1413	-----		
1414			
1415	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		
1416	バル終了処理ルーチンは，いずれのクラスにも属さない．オーバランハンドラ		
1417	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．拡		
1418	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		
1419	実行される．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		
1420	スタプロセッサによって実行される．		
1421			
1422	2.5 システム状態とコンテキスト		
1423			
1424	2.5.1 カーネル動作状態と非動作状態		
1425			
1426	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，		
1427	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		
1428	（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		
1429	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		
1430	態かカーネル非動作状態のいずれかの状態を取る．		
1431			
1432	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		
1433	る．		
1434			
1435	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		
1436	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．カー		
1437	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		
1438	されない．		
1439			
1440	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		
1441	ネル非動作状態のいずれかの状態を取る．		
1442			
1443	2.5.2 タスクコンテキストと非タスクコンテキスト		
1444			
1445	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		
1446	ど）をコンテキストと呼ぶ．		
1447			
1448	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		
1449	ンテキストと非タスクコンテキストに分類される．		
1450			

1451 タスク（タスク例外処理ルーチンを含む）が実行されるコンテキストは、タスク
1452 コンテキストに分類される。また、タスクコンテキストから呼び出した拡張
1453 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
1454 る。

1455
1456 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
1457 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
1458 分類される。また、非タスクコンテキストから呼び出した拡張サービスコール
1459 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。

1460
1461 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
1462 スタック領域を用いて実行される。非タスクコンテキストで実行される処理単
1463 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
1464 行される。

1465
1466 タスクコンテキストからは、非タスクコンテキスト専用のサービスコールを呼
1467 び出すことはできない。逆に、非タスクコンテキストからは、タスクコンテキ
1468 スト専用のサービスコールを呼び出すことはできない。いずれも、呼び出した
1469 場合にはE_CTXエラーとなる。

1470

1471 2.5.3 カーネルの振舞いに影響を与える状態

1472

1473 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
1474 状態として、次の状態を持つ。

1475

- 1476 ・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）
- 1477 ・CPUロックフラグ（CPUロック状態とCPUロック解除状態）
- 1478 ・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）
- 1479 ・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）

1480

1481 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
1482 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
1483 とができる。

1484

1485 2.5.4 全割込みロック状態と全割込みロック解除状態

1486

1487 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
1488 ラグを持つ。全割込みロックフラグがセットされた状態を全割込みロック状態、
1489 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
1490 ク状態では、NMIを除くすべての割込みがマスクされる。

1491

1492 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
1493 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
1494 ル（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
1495 ことはできない。全割込みロック状態で、その他のサービスコールを呼び出し
1496 た場合の動作は、保証されない。また、全割込みロック状態では、実行中の処
1497 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。

1498

1499 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
1500 を持つ。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

1501 除状態のいずれかの状態を取る。

1502

1503 2.5.5 CPUロック状態とCPUロック解除状態

1504

1505 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
1506 節を参照）をすべてマスクするためのCPUロックフラグを持つ。CPUロックフラ
1507 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
1508 態と呼ぶ。すなわち、CPUロック状態では、すべてのカーネル管理の割込みがマ
1509 スクされ、ディスパッチが保留される。

1510

1511 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
1512 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。

1513

1514 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。

1515

- 1516 ・ システムインタフェースレイヤのAPI
- 1517 ・ loc_cpu/iloc_cpu, unl_cpu/iunl_cpu
- 1518 ・ unl_spn/iunl_spn（マルチプロセッサ対応カーネルのみ）
- 1519 ・ dis_int, ena_int
- 1520 ・ sns_yyy, xsns_yyy（CPU例外ハンドラからのみ）
- 1521 ・ get_utm
- 1522 ・ ext_tsk, ext_ker
- 1523 ・ cal_svc（保護機能対応カーネルのみ）

1524

1525 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
1526 となる。

1527

1528 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。
1529 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
1530 状態を取る。

1531

1532 【補足説明】

1533

1534 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
1535 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
1536 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、
1537 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
1538 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。

1539

1540 2.5.6 割込み優先度マスク

1541

1542 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
1543 マスクを持つ。割込み優先度マスクがTIPM_ENAALL（=0）の時は、いずれの割
1544 込み要求もマスクされない。この状態を割込み優先度マスク全解除状態と呼ぶ。
1545 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
1546 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
1547 保留される。この状態を割込み優先度マスクが全解除でない状態と呼ぶ。

1548

1549 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
1550 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

1551 はできない。呼び出した場合には、E_CTXエラーとなる。

1552

1553 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
1554 持つ。

1555

1556 2.5.7 ディスパッチ禁止状態とディスパッチ許可状態

1557

1558 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。
1559 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
1560 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
1561 では、ディスパッチは保留される。

1562

1563 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
1564 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
1565 出した場合には、E_CTXエラーとなる。

1566

1567 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
1568 グを持つ。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
1569 許可状態のいずれかの状態を取る。

1570

1571 【補足説明】

1572

1573 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
1574 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。
1575 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
1576 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
1577 現することはできない。

1578

1579 2.5.8 ディスパッチ保留状態

1580

1581 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
1582 でない状態、ディスパッチ禁止状態では、ディスパッチが保留される。これら
1583 の状態を総称して、ディスパッチ保留状態と呼ぶ。

1584

1585 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
1586 かそうでない状態のいずれかの状態を取る。

1587

1588 【補足説明】

1589

1590 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
1591 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。

1592

1593 2.5.9 カーネル管理外の状態

1594

1595 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
1596 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
1597 （「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
1598 外の状態と呼ぶ。

1599

1600 それぞれの節で規定する通り、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とxsns_xpn）を呼び出すことができ、その他のサービスコールを呼び出すことはできない。カーネル管理外の状態から、その他のサービスコールを呼び出した場合の動作は、保証されない。

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

2.5.10 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】			
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

【タスク例外処理ルーチン】			
実行開始条件	解除	全解除	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	原則解除(*1)	原則全解除(*1)	元に戻す
リターン時処理	解除する	全解除する	元に戻す(*4)

【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】			
実行開始条件	解除	自優先度より低い	任意
実行開始時処理	そのまま	自優先度に(*2)	そのまま
リターン前	原則解除(*1)	変更不可(*3)	変更不可(*3)
リターン時処理	解除する	元に戻す(*5)	そのまま

【CPU例外ハンドラ】			
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	原則元に戻す(*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】			
実行開始条件	任意	任意	任意

1651	実行開始時処理	そのまま	そのまま	そのまま
1652	リターン前	任意	任意	任意
1653	リターン時処理	そのまま	そのまま	そのまま

1654

1655

1656 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）

1657 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、

1658 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

1659 態が設定されるため、支障がないことを意味する。

1660

1661 「自優先度に(*2)」とは、割込みハンドラと割込みサービスルーチンの場合に

1662 はそれを要求した割込みの割込み優先度、周期ハンドラとアラームハンドラの

1663 場合にはタイマ割込みの割込み優先度、オーバランハンドラの場合にはオーバ

1664 ランタイマ割込みの割込み優先度に変更することを意味する。

1665

1666 「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

1667 が用意されていないことを示す。

1668

1669 保護機能対応カーネルでは、タスク例外処理ルーチンからのリターン時にディ

1670 スパッチ禁止フラグを元に戻す処理(*4)は、タスクにディスパッチ禁止フラグ

1671 の変更を許可している場合にのみ行われる。カーネルは、ディスパッチ禁止フ

1672 ラグの元の状態をユーザスタック上に保存する。アプリケーションがユーザス

1673 タック上に保存されたディスパッチ禁止フラグの状態を書き換えた場合、タス

1674 ク例外処理ルーチンからのリターン時には、書き換えた後のディスパッチ禁止

1675 フラグの状態に変更される（すなわち、元に戻されるとは限らない）。

1676

1677 また、タスクにディスパッチ禁止フラグの変更を許可していない場合で、タス

1678 ク例外処理ルーチン中で拡張サービスコールを用いてディスパッチ禁止フラグ

1679 を変更した場合、カーネルは元の状態に戻さない。このことから、タスク例外

1680 処理ルーチンからの終了前に、ディスパッチ禁止フラグを元の状態に戻すのは、

1681 アプリケーションの責任とする。

1682

1683 【補足説明】

1684

1685 マルチプロセッサ対応カーネルにおいて、タスクがタスク例外処理ルーチンを

1686 実行中にマイグレーションされた場合、マイグレーション先のプロセッサにお

1687 いて、割込み優先度マスクとディスパッチ禁止フラグが元に戻される。

1688

1689 【仕様決定の理由】

1690

1691 保護機能対応カーネルにおいて、タスク例外処理ルーチンからのリターン時に

1692 ディスパッチ禁止フラグを元に戻す処理(*4)が、タスクにディスパッチ禁止フ

1693 ラグの変更を許可している場合にのみ行われるのは、タスクがユーザスタック

1694 上の状態を書き換えることで、許可していない状態変更を起こしてしまうこと

1695 を防止するためである。

1696

1697 割込みハンドラやCPU例外ハンドラで、その処理単位中で割込み優先度マスクを

1698 変更するAPIが用意されていないにもかかわらず、処理単位からのリターン時に

1699 元の状態に戻す(*5)のは、プロセッサによっては、割込み優先度マスクがステー

1700 タスレジスタ等に含まれており、APIを用いずに変更できてしまう場合があるた

1701 めである。

1702

1703 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
1704 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
1705 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
1706 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,
1707 xsns_xpnともtrueを返す）、CPU例外ハンドラ中でiunl_cpuを呼び出してCPUロッ
1708 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
1709 にも関わらずiunl_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
1710 に戻すこととしている。

1711

1712 2.6 タスクの状態遷移とスケジューリング規則

1713

1714 2.6.1 基本的なタスク状態

1715

1716 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
1717 いずれかの状態を取る。また、実行できる状態と広義の待ち状態を総称して、
1718 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
1719 状態を、未登録状態と呼ぶ。

1720

1721 (a) 実行できる状態 (runnable)

1722

1723 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
1724 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
1725 る。

1726

1727 (a.1) 実行状態 (running)

1728

1729 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
1730 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
1731 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。

1732

1733 (a.2) 実行可能状態 (ready)

1734

1735 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
1736 行状態にあるために、そのタスクが実行されない状態。

1737

1738 (b) 休止状態 (dormant)

1739

1740 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
1741 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
1742 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）
1743 は保存されていない。

1744

1745 (c) 広義の待ち状態 (blocked)

1746

1747 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
1748 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
1749 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
1750 状態に遷移する前の状態に戻される。広義の待ち状態は、さらに、（狭義の）

1751 待ち状態、強制待ち状態、二重待ち状態に分類される。
1752
1753 (c.1) (狭義の) 待ち状態 (waiting)
1754
1755 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。
1756
1757 (c.2) 強制待ち状態 (suspended)
1758
1759 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
1760 を強制待ち状態にすることも可能である。
1761
1762 (c.3) 二重待ち状態 (waiting-suspended)
1763
1764 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
1765 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
1766 実行を止められている状態。
1767
1768 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
1769 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
1770 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
1771 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
1772 重待ち状態でもないことを意味する。
1773
1774 (d) 未登録状態 (non-existent)
1775
1776 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、
1777 タスクは未登録状態にあるとみなす。
1778
1779 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
1780 合がある。過渡的な状態については、「2.6.6 ディスパッチ保留状態で実行中
1781 のタスクに対する強制待ち」の節を参照すること。
1782
1783 **【TOPPERS/ASPカーネルにおける規定】**
1784
1785 ASPカーネルでは、タスクが未登録状態になることはない。また、上記のタスク
1786 状態以外の過渡的な状態になることもない。ただし、動的生成機能拡張パッケ
1787 ジでは、タスクが未登録状態になる。
1788
1789 **【TOPPERS/FMPカーネルにおける規定】**
1790
1791 FMPカーネルでは、タスクが未登録状態になることはない。上記のタスク状態以
1792 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
1793 ある。詳しくは、「2.6.6 ディスパッチ保留状態で実行中のタスクに対する強
1794 制待ち」の節を参照すること。
1795
1796 **【TOPPERS/HRP2カーネルにおける規定】**
1797
1798 HRP2カーネルでは、タスクが未登録状態になることはない。また、上記のタ
1799 スク状態以外の過渡的な状態になることもない。
1800

【TOPPERS/SSPカーネルにおける規定】

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

2.6.2 タスクの状態遷移

タスクの状態遷移を図2-2に示す。

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

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

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

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

【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。すなわち、ディスパッチ保留状態でない限りは、実行できるタスクの中で最も高い優先順位を持つタスクが実行状態となり、他は実行可能状態となる。

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

1851 することも可能である。

1852

1853 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
1854 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
1855 実行状態となる。ディスパッチ保留状態においては、実行状態のタスクは切り
1856 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。

1857

1858 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
1859 グ規則を適用して、タスクスケジューリングを行う。すなわち、プロセッサが
1860 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
1861 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
1862 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。

1863

1864 2.6.4 待ち行列と待ち解除の順序

1865

1866 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
1867 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
1868 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
1869 クトと呼ぶ。

1870

1871 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
1872 らの順序でつなぐかは、待ち行列毎に規定される。多くの待ち行列において、
1873 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。

1874

1875 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
1876 最後につながれる。それに対してタスクの優先度順の待ち行列においては、新
1877 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。
1878 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
1879 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。

1880

1881 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
1882 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
1883 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
1884 舞いをするかは、待ち行列毎に規定される。

1885

1886 (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
1887 のから順に待ち解除される。すなわち、待ち行列の前方に待ち解除の条件を満
1888 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、
1889 先に待ち解除される。

1890

1891 (b) タスクの待ち解除は、待ち行列につながれている順序で行われる。すなわ
1892 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
1893 スクが待ち解除の条件を満たしても、待ち解除されない。

1894

1895 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
1896 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）、
1897 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。
1898 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
1899 ち解除の条件を満たしていれば、ただちに待ち解除される。さらに、この待ち
1900 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

1901 返される。

1902

1903 2.6.5 タスク例外処理マスク状態と待ち禁止状態

1904

1905 保護機能対応カーネルにおいて、ユーザタスクについては特権モードで実行し
1906 ている間（特権モードを実行している間に、実行可能状態や広義の待ち状態に
1907 なっている場合を含む。また、サービスコールを呼び出して、実行可能状態や
1908 広義の待ち状態になっている場合も含む。タスクの実行開始前は含まない）、
1909 システムタスクについては拡張サービスコールを実行している間（拡張サービ
1910 スコールを実行している間に、実行可能状態や広義の待ち状態になっている場
1911 合を含む）は、タスク例外処理ルーチンの実行は開始されない。これらの状態
1912 を、タスク例外処理マスク状態と呼ぶ。

1913

1914 タスクは、タスク例外処理マスク状態である時に、基本的なタスク状態と重複
1915 して、待ち禁止状態になることができる。

1916

1917 待ち禁止状態とは、タスクが待ち状態に入ることが一時的に禁止された状態で
1918 ある。待ち禁止状態にあるタスクが、サービスコールを呼び出して待ち状態に
1919 遷移しようとした場合、サービスコールはE_RLWAIエラーとなる。

1920

1921 タスクを待ち禁止状態に遷移させるサービスコールは、対象タスクがタスク例
1922 外処理マスク状態である場合に、対象タスクを待ち禁止状態に遷移させる。そ
1923 の後、タスクがタスク例外処理マスク状態でなくなる時点（ユーザタスクにつ
1924 いては特権モードから戻る時点、システムタスクについて拡張サービスコール
1925 からリターンする時点）で、待ち禁止状態が解除される。また、タスクの待ち
1926 禁止状態を解除するサービスコールによっても、待ち禁止状態を解除するこ
1927 ができる。

1928

1929 【仕様決定の理由】

1930

1931 タスク例外処理ルーチンでは、タスクの本体のための例外処理（例えば、タス
1932 クに対して終了要求があった時の処理）を行うことを想定しており、タスクか
1933 ら呼び出した拡張サービスコールのための例外処理を行うことは想定していな
1934 い。そのため、拡張サービスコールを実行している間にタスク例外処理が要求
1935 された場合に、すぐにタスク例外処理ルーチンを実行すると、拡張サービスコ
1936 ルのための例外処理が行われないことになる。

1937

1938 また、ユーザタスクの場合には、特権モードを実行中にタスク例外処理ルーチ
1939 ンを実行すると、システムスタックに情報を残したまま非特権モードに戻るこ
1940 とになる。この状態で、タスク例外処理ルーチンから大域脱出すると、システ
1941 ムスタック上に不要な情報が残ってしまう。

1942

1943 これらの理由から、タスクが拡張サービスコールを実行している間は、タスク
1944 例外処理マスク状態とし、タスク例外処理ルーチンの実行を開始しないことと
1945 する。さらに、ユーザタスクについては、特権モードを実行している間（拡張
1946 サービスコールを実行している間を含む）を、タスク例外処理マスク状態とす
1947 る。

1948

1949 対象タスクに、タスク例外処理ルーチンをすみやかに実行させたい場合には、
1950 タスク例外処理の要求に加えて、待ち状態の強制解除を行う（必要に応じて、

1951 強制待ち状態からの再開も行う)。保護機能対応でないカーネルにおいては、
1952 この方法により、対象タスクが正常に待ち解除されるのを待たずに、タスク例
1953 外処理ルーチンを実行させることができる。

1954
1955 それに対して、保護機能対応カーネルにおいては、対象タスクがタスク例外処
1956 理マスク状態で実行している間は、タスク例外処理ルーチンの実行が開始され
1957 ない。そのため、対象タスクに対して待ち状態の強制解除を行っても、その後
1958 に対象タスクが待ち状態に入ると、タスク例外処理ルーチンがすみやかに実行
1959 されないことになる。

1960
1961 待ち禁止状態は、この問題を解決するために導入したものである。タスク例外
1962 処理の要求 (ras_tex/iras_tex) に加えて、待ち禁止状態への遷移 (dis_wai/
1963 idis_wai) と待ち状態の強制解除 (rel_wai/irel_wai) をこの順序で行うこと
1964 で、対象タスクが正常に待ち解除されるのを待たずに、タスク例外処理ルーチ
1965 ンを実行させることができる。

1966
1967 タスク例外処理マスク状態を、ユーザタスクについても拡張サービスコールを
1968 実行している間とせず、特権モードで実行している間とした理由は、拡張サー
1969 ビスコールを実行している間とした場合に次のような問題があるためである。

1970
1971 ユーザタスクが、ソフトウェア割込みにより自タスクを待ち状態に遷移させる
1972 サービスコールを呼び出した直後に割込みが発生し、その割込みハンドラの中
1973 でiras_tex, idis_wai, irel_waiが呼び出されると、この時点では待ち解除も
1974 されず待ち禁止状態にもならないために、割込みハンドラからのリターン後に
1975 待ち状態に入ってしまう。ソフトウェア割込みによりすべての割込みが禁止さ
1976 れないターゲットプロセッサでは、ソフトウェア割込みの発生とサービスコー
1977 ルの実行を不可分にできないため、このような状況を防ぐことができない。

1978
1979 なお、拡張サービスコールは、待ち状態に入るサービスコールからE_RLWAIが返
1980 された場合には、実行中の処理を取りやめて、E_RLWAIを返値としてリターンす
1981 るように実装すべきである。

1982
1983 【 μ ITRON4.0仕様、 μ ITRON4.0/PX仕様との関係】

1984
1985 待ち禁止状態は、 μ ITRON4.0仕様にはない概念であり、 μ ITRON4.0/PX仕様で導
1986 入された。ただし、 μ ITRON4.0/PX仕様では、タスクの待ち状態を強制解除する
1987 サービスコールが、タスクを待ち禁止状態へ遷移させる機能も持つこととして
1988 いる。その結果 μ ITRON4.0/PX仕様は、待ち状態を強制解除するサービスコール
1989 の仕様において、 μ ITRON4.0仕様との互換性がなくなっている。

1990
1991 この仕様では、待ち状態の強制解除と待ち禁止状態への遷移を別々のサービス
1992 コールで行うこととした。これにより、待ち状態を強制解除するサービスコー
1993 ルの仕様が、 μ ITRON4.0仕様と互換になっている。一方、 μ ITRON4.0/PX仕様と
1994 は互換性がない。

1995
1996 2.6.6 ディスパッチ保留状態で実行中のタスクに対する強制待ち

1997
1998 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
1999 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
2000 チ保留状態が解除されるまで保留される。

2001
2002 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
2003 過渡的な状態にあると考える。この状態を、強制待ち状態〔実行継続中〕と呼
2004 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
2005 可能状態にとどまる。

2006
2007 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
2008 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
2009 る。

2010
2011 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。

2012
2013 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。

2014
2015 (a) プロセッサを占有して実行を継続する。

2016
2017 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
2018 続して実行される。

2019
2020 (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
2021 であるものと扱う。

2022
2023 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/`
2024 `iget_tid`, `get_did`, `sns_tex`) では、強制待ち状態〔実行継続中〕のタスクが、
2025 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
2026 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/`
2027 `iget_tid`を発行すると、そのタスクのID番号を参照する。また、`get_did`を発行
2028 するとそのタスクが属する保護ドメインのID番号を、`sns_tex`を発行するとその
2029 タスクのタスク例外処理禁止フラグを参照する。

2030
2031 (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。

2032
2033 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
2034 待ち状態であるものと扱う。

2035
2036 なお、TOPPERS新世代カーネルでは、ディスパッチ保留状態において、実行状態
2037 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。

2038
2039 **【TOPPERS/ASPカーネルにおける規定】**

2040
2041 ASPカーネルでは、ディスパッチ保留状態において実行状態のタスクを強制待ち
2042 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
2043 ち状態〔実行継続中〕になることはない。

2044
2045 **【TOPPERS/FMPカーネルにおける規定】**

2046
2047 FMPカーネルでは、ディスパッチ保留状態において実行状態のタスクを強制待ち
2048 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
2049 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。

2050

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、ディスパッチ保留状態において実行状態のタスクを強制待ち状態へ遷移させるサービスコールはサポートしていないため、タスクが強制待ち状態〔実行継続中〕になることはない。

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、タスクが広義の待ち状態になることはないため、タスクが強制待ち状態〔実行継続中〕になることもない。

2.6.7 制約タスク

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

(a) 広義の待ち状態に入ることができない。

(b) サービスコールにより優先度を変更することができない。

(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/irot_rdq) を行うことができない。

(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。

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。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。

【未決定事項】

現状では、制約タスクの優先度を変更するサービスコールは設けていないが、制約タスクが、自タスクの優先度を、起動時優先度 (SSPカーネルにおいては、実行時優先度) と同じかそれよりも高い値に変更することは許してもよい。ただし、優先度の変更後は、同じ優先度内で最高優先順位としなければならないため、chg_priとは振舞いが異なることになる。自タスクの優先度を起動時優先度と同じかそれよりも高い値に変更するサービスコールを設けるかどうかは、今後の課題である。

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、制約タスクをサポートしていない。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。

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、制約タスクをサポートしていない。

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、制約タスクをサポートしていない。

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、制約タスクのみをサポートする。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。このスタック領域を、共有スタック領域と呼ぶ。

【 μ ITRON4.0仕様との関係】

制約タスクは、 μ ITRON4.0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、 μ ITRON4.0仕様の制約タスクよりも機能制限が少なくなっている。

2.7 割込み処理モデル

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

TOPPERS標準割込み処理モデルの概念図を図2-4に示す。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。

【 μ ITRON4.0仕様との関係】

割込み処理モデルは、 μ ITRON4.0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。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。

カーネル内の割込みハンドラは、アプリケーションが割込み要求ラインに対して登録した割込みサービスルーチン（ISR）を呼び出す。割込みサービスルーチンは、プロセッサ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だけに依存して記述するのが原則である。1つの割込み要求ラインに対して複数のデバイスが接続されることから、1つの割込み要求

2151 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。

2152

2153 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
2154 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
2155 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。

2156

2157 カーネルが用いるタイマデバイスからの割込み要求の場合、カーネル内の割込
2158 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
2159 ウト処理等が行われることに加えて、アプリケーションが登録したタイムイベ
2160 ントハンドラが呼び出される。

2161

2162 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
2163 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。

2164

2165 2.7.2 割込み優先度

2166

2167 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。
2168 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
2169 込み要求のみを受け付ける。逆に言うと、割込み優先度マスクの現在値と同じ
2170 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。

2171

2172 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
2173 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
2174 る場合には例外とする）。また、割込み処理からのリターンにより、割込み優
2175 先度マスクを、割込み要求を受け付ける前の値に戻す。

2176

2177 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
2178 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
2179 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
2180 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
2181 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、
2182 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。

2183

2184 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、
2185 優先度に関する原則には従わず、-1から連続した負の値を用いる。

2186

2187 割込み優先度の段階数は、ターゲット定義である。プロセッサが割込み優先度
2188 マスクを実現するための機能を持たないか、実現するために大きいオーバーヘッ
2189 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
2190 なわち、多重割込みを許さない）場合がある。

2191

2192 【仕様決定の理由】

2193

2194 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
2195 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
2196 先度マスクの値を0にできるためである。

2197

2198 2.7.3 割込み要求ラインの属性

2199

2200 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

2201 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
2202 同一の属性を持つ。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
2203 い。

2204

2205 (1) 割込み要求禁止フラグ

2206

2207 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
2208 つ。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
2209 えられる割込み要求はマスクされる。

2210

2211 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
2212 するために大きいオーバーヘッドを生じる場合には、ターゲット定義で、割込み
2213 要求禁止フラグをサポートしない場合がある。また、プロセッサの持つ割込み
2214 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、
2215 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
2216 ある。

2217

2218 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
2219 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、
2220 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
2221 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
2222 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
2223 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
2224 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
2225 実現すべきである。

2226

2227 (2) 割込み優先度

2228

2229 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。割込み要求の
2230 割込み優先度とは、その割込み要求を伝える割込み要求ラインに対して設定さ
2231 れた割込み優先度のことである。

2232

2233 (3) トリガモード

2234

2235 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
2236 であるかを設定することができる。エッジトリガの場合には、さらに、ポジティ
2237 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
2238 合もある。また、レベルトリガの場合には、ローレベルトリガかハイレベルト
2239 リガかを設定できる場合もある。

2240

2241 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
2242 に大きいオーバーヘッドを生じる場合には、ターゲット定義で、トリガモードの
2243 設定をサポートしない場合がある。

2244

2245 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
2246 がセットされ、割込み要求はマスクされる。また、割込み要求禁止フラグをク
2247 リアすることもできない。

2248

2249 2.7.4 割込みを受け付ける条件

2250

2251 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。

2252

2253 (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

2254

2255 (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
2256 在値よりも高い（優先度の値としては小さい）こと

2257

2258 (c) 全割込みロックフラグがクリアされていること

2259

2260 (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
2261 リアされていること

2262

2263 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
2264 受け付けられるかは、この仕様では規定しない。すなわち、割込み優先度の高
2265 い割込み要求が先に受け付けられるとは限らない。

2266

2267 2.7.5 割込み番号と割込みハンドラ番号

2268

2269 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、
2270 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
2271 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。そのため、
2272 1から連続した正の値であるとは限らない。

2273

2274 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
2275 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、
2276 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
2277 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
2278 本として、ターゲット定義で付与される。そのため、1から連続した正の値であ
2279 るとは限らない。

2280

2281 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
2282 一致する場合が多い）。

2283

2284 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
2285 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
2286 ネルが管理していない場合がある。そこで、ターゲット定義で、割込み番号に
2287 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
2288 号を設ける場合もある。ただし、割込みサービスルーチンの登録対象にできる
2289 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
2290 いるもののみである。

2291

2292 2.7.6 マルチプロセッサにおける割込み処理

2293

2294 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
2295 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。

2296

2297 マルチプロセッサ対応カーネルでは、TOPPERS標準割込み処理モデルの構成要素
2298 の中で、図2-4の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
2299 システム全体で1つのみ持つ。すなわち、全割込みロックフラグ、CPUロックフ
2300 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）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。逆に、複数のプロセッサが同じ種類のデバイスを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2-5）。図2-5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。

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。

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。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。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。

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2-5）。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。

【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、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2-5）。

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。

【使用上の注意】

複数のプロセッサで実行することができる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かで実行されるものと設定した場合でも、複数回の割込み要求により、異なるプロセッサで同時に実行される可能性がある。

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。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。

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。

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。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。

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。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。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。

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。

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

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

2401 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
2402 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。
2403 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
2404 管理外であることを示す割込みハンドラ属性 (TA_NONKERNEL) を指定する。また、
2405 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
2406 TMIN_INTPRIよりも高い値とする。

2407

2408 (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

2409

2410 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、
2411 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
2412 であることを示す割込みハンドラ属性 (TA_NONKERNEL) を指定する。また、カー
2413 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、
2414 TMIN_INTPRIよりも高い値とする。

2415

2416 いずれの場合にも、カーネル管理の割込みの割込み要求ラインに対して設定す
2417 る割込み優先度は、TMIN_INTPRIより高い値であってはならない。また、カー
2418 ネル管理外の割込みに対して、割込みサバスルーチンを登録することはできな
2419 い。

2420

2421 2.8 CPU例外処理モデル

2422

2423 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
2424 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
2425 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
2426 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、
2427 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。

2428

2429 2.8.1 CPU例外処理の流れ

2430

2431 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
2432 ドラを登録することができる。プロセッサがCPU例外の発生を検出すると、カー
2433 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
2434 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。

2435

2436 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
2437 ンドラ番号と呼ぶ。CPU例外ハンドラ番号は、符号無し of 整数型であるEXCNO型
2438 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
2439 て、ターゲット定義で付与される。そのため、1から連続した正の値であるとは
2440 限らない。

2441

2442 マルチプロセッサ対応カーネルでは、異なるプロセッサで発生するCPU例外は、
2443 異なるCPU例外であると扱う。すなわち、同じ種類のCPU例外であっても、異な
2444 るプロセッサのCPU例外には異なるCPU例外ハンドラ番号を付与し、プロセッサ
2445 毎にCPU例外ハンドラを登録する。CPU例外ハンドラが属するクラスの初期割付
2446 けプロセッサは、CPU例外が発生するプロセッサと一致していなければならない。

2447

2448 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
2449 う。どのようなリカバリ処理を行うかは、一般にはCPU例外の種類やそれが発生
2450 したコンテキストおよび状態に依存するが、大きく次の4つの方法が考えられる。

2451
2452 (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。
2453
2454 (b) CPU例外を起こしたタスクよりも優先度の高いタスクを起動または待ち解除
2455 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
2456 制終了し、再度起動する）。ただし、CPU例外を起こしたタスクが最高優先度の
2457 場合には、この方法でリカバリ処理を行うことはできない（リカバリ処理を行
2458 うタスクを最高優先度とし、タスクの起動または待ち解除後に優先順位を回転
2459 させることで、リカバリ処理を行える可能性があるが、CPU例外を起こしたタ
2460 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）。
2461
2462 (c) CPU例外を起こしたタスクにタスク例外処理を要求し、タスク例外処理ルー
2463 チンでリカバリ処理を行う（例えば、CPU例外を起こしたタスクを終了する）。
2464
2465 (d) システム全体に対してリカバリ処理を行う（例えば、システムを再起動す
2466 る）。
2467
2468 この中で(a)と(d)の方法は、カーネルの機能を必要としないため、CPU例外が発
2469 生したコンテキストおよび状態に依存せずに常に行える。それに対して(b)と
2470 (c)の方法は、CPU例外ハンドラからそのためのサービスコールを呼び出せるこ
2471 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
2472 よび状態に依存する。
2473
2474 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
2475 舞いは、ターゲット定義である。

2476 【使用上の注意】

2477
2478
2479 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
2480 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
2481 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
2482 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
2483 けない場合には、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
2484 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。

2485 【補足説明】

2486
2487
2488 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、
2489 CPU例外ハンドラ番号の上位ビットに、そのCPU例外が発生するプロセッサのID
2490 番号を含める方法を基本とする。

2491 【 μ ITRON4.0仕様との関係】

2492
2493
2494 μ ITRON4.0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
2495 ていない。

2496 2.8.2 CPU例外ハンドラから呼び出せるサービスコール

2497
2498
2499 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
2500 ビスコール（xsns_dpn）と、CPU例外発生時にタスク例外処理ルーチンを実行開

2501 始できない状態であったかを参照するサービスコール (xsns_xpn) を呼び出す
2502 ことができる。

2503

2504 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
2505 チできる状態であった場合にfalseを返す。xsns_dpnがfalseを返した場合、そ
2506 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
2507 スコールを呼び出すことができ、(b)の方法によるリカバリ処理が可能である。
2508 ただし、CPU例外を起こしたタスクが最高優先度の場合には、この方法でリカバ
2509 リ処理を行うことはできない。

2510

2511 xsns_xpnは、CPU例外がタスクコンテキストで発生し、そのタスクがタスク例外
2512 処理ルーチンを実行できる状態であった場合にfalseを返す。xsns_xpnがfalse
2513 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
2514 るすべてのサービスコールを呼び出すことができ、(c)の方法によるリカバリ処
2515 理が可能である。

2516

2517 xsns_dpnとxsns_xpnのいずれのサービスコールもtrueを返した場合、そのCPU例
2518 外ハンドラからは、xsns_dpnとxsns_xpnに加えて、システムインタフェースレ
2519 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
2520 ールを呼び出すことはできない。いずれのサービスコールもtrueを返したにもか
2521 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。
2522 この場合には、(b)と(c)の方法によるリカバリ処理は行うことはできず、(a)ま
2523 たは(d)の方法によるリカバリ処理を行うしかないことになる。

2524

2525 【μ ITRON4.0仕様との関係】

2526

2527 CPU例外ハンドラで行える操作に関しては、μ ITRON4.0仕様を見直し、全面的に
2528 修正した。

2529

2530 2.8.3 エミュレートされたCPU例外ハンドラ

2531

2532 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
2533 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
2534 す場合がある。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
2535 ものとし、エミュレートされたCPU例外と呼ぶ。また、エラー処理のためのプロ
2536 グラムをCPU例外ハンドラと同等に扱うものとし、エミュレートされたCPU例外
2537 ハンドラと呼ぶ。

2538

2539 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
2540 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。また、エミュレー
2541 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
2542 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
2543 ができる。

2544

2545 【μ ITRON4.0仕様との関係】

2546

2547 エミュレートされたCPU例外およびCPU例外ハンドラは、μ ITRON4.0仕様に定義
2548 されていない概念である。

2549

2550 2.8.4 カーネル管理外のCPU例外

2551
2552 カーネル内のクリティカルセクションの実行中（これを、カーネル実行中と呼
2553 ぶ）、全割込みロック状態、CPUロック状態、カーネル管理外の割込みハンドラ
2554 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
2555 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
2556 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
2557 CPU例外も、カーネル管理外のCPU例外とする。
2558
2559 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
2560 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
2561 理のCPU例外ハンドラと呼ぶ。
2562
2563 カーネル管理外のCPU例外ハンドラからは、システムインタフェースレイヤの
2564 APIとsns_ker, ext_ker, xsns_dpn, xsns_xpnのみを呼び出すことができ、その
2565 他のサービスコールを呼び出すことはできない。カーネル管理外のCPU例外ハン
2566 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。
2567
2568 カーネル管理外のCPU例外ハンドラにおいては、xsns_dpnとxsns_xpnのいずれの
2569 サービスコールもtrueを返す。そのため、カーネル管理外のCPU例外からは、
2570 (a)または(d)の方法によるリカバリ処理しか行えない。
2571
2572 【補足説明】
2573
2574 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
2575 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
2576 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。
2577
2578 2.9 システムの初期化と終了
2579
2580 2.9.1 システム初期化手順
2581
2582 システムのリセット後、最初に行うプログラムを、スタートアップモジュール
2583 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
2584 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
2585 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
2586 ップモジュールが用意されている。
2587
2588 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
2589 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
2590 状態にする）、ターゲットシステム依存の初期化フックの呼出し、非初期化デー
2591 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
2592 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
2593 の呼出しを行った後、カーネルの初期化処理へ分岐する。ここで呼び出すター
2594 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
2595 化処理を行うことが想定されている。
2596
2597 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
2598 ジュールを実行し、カーネルの初期化処理へ分岐する。ただし、共有リソース
2599 の初期化処理（非初期化データセクションのクリア、初期化データセクション
2600 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

2601 ロセッサのみで実行する。各プロセッサがカーネルの初期化処理へ分岐するの
2602 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
2603 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
2604 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。

2605
2606 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
2607 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
2608 APIの処理（オブジェクトの登録など）が行われる。静的APIのパラメータに関
2609 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
2610 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
2611 る。

2612
2613 静的APIの処理順序によりシステムの規定された振舞いに変化する場合には、シ
2614 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
2615 静的APIが処理された場合と、同じ振舞いとなる。例えば、静的APIによって同
2616 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
2617 高い優先順位を持つ。それに対して、周期ハンドラの動作開始順序は、同じタ
2618 イムティックで行うべき処理が複数ある場合の処理順序が規定されないことか
2619 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
2620 らない。

2621
2622 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
2623 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。

2624
2625 マルチプロセッサ対応カーネルでは、すべてのプロセッサがカーネル自身の初
2626 期化処理と静的APIの処理を完了した後に、マスタプロセッサがグローバル初期
2627 化ルーチンを実行する。グローバル初期化ルーチンの実行が完了した後に、各
2628 プロセッサは、自プロセッサに割り付けられたローカル初期化ルーチンを実行
2629 する。すなわち、ローカル初期化ルーチンは、初期割付けプロセッサにより実
2630 行される。

2631
2632 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
2633 ネル動作状態と非動作状態」の節を参照）、カーネルの動作が開始される。具
2634 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
2635 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、
2636 割込みがマスク解除され）、タスクの実行が開始される。

2637
2638 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
2639 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
2640 ネルの動作が開始される。マルチプロセッサ対応カーネルにおけるシステム初
2641 期化の流れと、各プロセッサが同期を取るタイミングを、図2-6に示す。

2642 【μ ITRON4.0仕様との関係】

2643
2644
2645 μ ITRON4.0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
2646 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、
2647 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
2648 いものとした。

2649 2.9.2 システム終了手順

2650

2651
2652 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
2653 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
2654 節を参照）。具体的には、NMIを除くすべての割込みがマスクされ、タスクの実
2655 行が停止される。

2656
2657 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
2658 (ext_ker) は、どのプロセッサからでも呼び出すことができる。1つのプロセッ
2659 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
2660 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
2661 終了処理の開始を要求する。複数のプロセッサから、カーネルを終了させるサー
2662 ビスコール (ext_ker) を呼び出してよい。

2663
2664 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
2665 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。

2666
2667 マルチプロセッサ対応カーネルでは、すべてのプロセッサがカーネル非動作状
2668 態に遷移した後に、各プロセッサが、自プロセッサに割り付けられたローカル
2669 終了処理ルーチンを実行する。すなわち、ローカル終了処理ルーチンは、初期
2670 割付けプロセッサにより実行される。すべてのプロセッサでローカル処理ルー
2671 チンの実行が完了した後に、マスタプロセッサがグローバル終了処理ルーチン
2672 を実行する。

2673
2674 以上が終了すると、ターゲットシステム依存の終了処理が呼び出される。ター
2675 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
2676 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
2677 トシステム毎に標準的なルーチンが用意されている。標準のターゲットシステ
2678 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
2679 フックを呼び出す。

2680
2681 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
2682 テム依存の終了処理が呼び出される。マルチプロセッサ対応カーネルにおける
2683 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2-7に
2684 示す。

2685
2686 **【使用上の注意】**

2687
2688 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
2689 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
2690 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。

2691
2692 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
2693 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
2694 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
2695 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
2696 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
2697 ネルを終了させることができる。

2698
2699 **【 μ ITRON4.0仕様との関係】**

2700

2701 μ ITRON4.0仕様には、システム終了に関する規定はない。

2702

2703 2.10 オブジェクトの登録とその解除

2704

2705 2.10.1 ID番号で識別するオブジェクト

2706

2707 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

2708 API (CRE_YYY) , サービスコール (acre_yyy) , またはオブジェクトを追加す

2709 る静的API (ATT_YYY, ATA_YYY) によってカーネルに登録する。オブジェクトを

2710 追加する静的APIによって登録されたオブジェクトはID番号を持たないため、

2711 ID番号を指定して操作することができない。

2712

2713 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

2714 号を割り付け、ID番号を指定するパラメータとして記述した識別名を、割り付

2715 けたID番号にマクロ定義する。同じ識別名のオブジェクトが生成済みの場合に

2716 は、E_OBJエラーとなる。

2717

2718 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

2719 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

2720 されていないID番号を1つ選び、生成するオブジェクトに割り付ける。割り付け

2721 たID番号は、サービスコールの返回值としてアプリケーションに通知する。使用

2722 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。

2723

2724 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

2725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。その場合、各静的

2726 APIで指定した数の合計の数のID番号が確保される。

2727

2728 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

2729 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

2730 除することができる。登録解除したオブジェクトのID番号は、未使用の状態に

2731 戻され、そのID番号を用いて新しいオブジェクトを登録することができる。こ

2732 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方が、新たに登

2733 録したオブジェクトに対して行われないように、注意が必要である。

2734

2735 オブジェクトを生成または追加する静的APIによって登録したオブジェクトは、

2736 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

2737 となる。

2738

2739 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

2740 ることができる。この場合、登録解除された処理単位に実行が強制的に終了さ

2741 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

2742 継続される。

2743

2744 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

2745 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

2746 ビスコールはE_DLTエラーとなる。複数のタスクが待ち解除される場合には、待

2747 ち行列につながれていた順序で待ち解除される。削除した同期・通信オブジェ

2748 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

2749 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。

2750

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

2755

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

2759

2760 【 μ ITRON4.0仕様との関係】

2761

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

2765

μ ITRON4.0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。

2768

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、 μ ITRON4.0仕様では実装依存とされている。

2772

2773 【 μ ITRON4.0/PX仕様との関係】

2774

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。

2779

2781 【仕様決定の理由】

2782

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。

2787

2788 2.10.2 オブジェクト番号で識別するオブジェクト

2789

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。

2792

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。登録解除したオブジェクト番号は、オブジェクト登録前の状態に戻され、同じオブジェクト番号に対して新たにオブジェクトを定義することができる。登録解除されていないオブジェクト番号に対して再度オブジェクトを登録しようとした場合には、E_OBJエラーとなる。

2799

2800

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

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

【 μ ITRON4.0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、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

識別する必要があるために、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。

2.10.4 オブジェクト生成に必要なメモリ領域

カーネルオブジェクトを生成する際に、サイズが一定でないメモリ領域を必要とする場合には、カーネルオブジェクトを生成する静的APIおよびサービスコールに、使用するメモリ領域の先頭番地を渡すパラメータを設けている。このパラメータをNULLとした場合、必要なメモリ領域は、コンフィギュレータまたはカーネルにより確保される。

オブジェクト生成に必要なメモリ領域の中で、カーネルの内部で用いるものを、カーネルの用いるオブジェクト管理領域と呼ぶ。この仕様では、以下のメモリ領域が、カーネルの用いるオブジェクト管理領域に該当する。

- ・データキュー管理領域
- ・優先度データキュー管理領域
- ・優先度別のメッセージキューヘッダ領域
- ・固定長メモリプール管理領域

【補足説明】

カーネルオブジェクトを生成する際には、管理ブロックなどを置くためのメモリ領域も必要になるが、サイズが一定のメモリ領域はコンフィギュレータにより確保されるため、カーネルオブジェクトを生成する静的APIおよびサービスコールにそれらのメモリ領域の先頭番地を渡すパラメータを設けていない。

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、カーネル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。

カーネルオブジェクトを静的APIによって登録する場合には、オブジェクトを登録する静的APIを、そのオブジェクトを属させる保護ドメインの囲みの中に記述する。無所属のオブジェクトを登録する静的APIは、保護ドメインの囲みの外に

2851 記述する（「2.12.3 保護ドメインの指定」の節を参照）。

2852

2853 カーネルオブジェクトをサービスコールによって登録する場合には、オブジェ
2854 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
2855 ドメインを設定する。ここでdomidは、そのオブジェクトを属させる保護ドメイ
2856 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
2857 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
2858 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
2859 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、
2860 domidにTDOM_NONE(=-2)を指定する。

2861

2862 ただし、特定の保護ドメインのみに属することができるカーネルオブジェクト
2863 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
2864 オブジェクト属性で設定する必要がないものもある。

2865

2866 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、
2867 どの保護ドメインに属するオブジェクトにも（また、無所属のオブジェクトに
2868 も）割り付けられる。これらの静的APIは、保護ドメインの囲みの外に記述しな
2869 ければならない。保護ドメインの囲みの中に記述した場合には、E_RSATRエラー
2870 となる。

2871

2872 **【補足説明】**

2873

2874 この仕様では、カーネルオブジェクトの属する保護ドメインを参照する機能は
2875 用意していない。

2876

2877 **【仕様決定の理由】**

2878

2879 カーネルオブジェクトをサービスコールによって登録する場合に、オブジェ
2880 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
2881 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
2882 タを増やさない方が望ましいためである。

2883

2884 2.10.6 オブジェクトが属するクラスの設定

2885

2886 マルチプロセッサ対応カーネルにおいて、カーネルオブジェクトが属するクラ
2887 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。

2888

2889 カーネルオブジェクトを静的APIによって登録する場合には、オブジェクトを登
2890 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。
2891 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
2892 する（「2.12.4 クラスの指定」の節を参照）。

2893

2894 カーネルオブジェクトをサービスコールによって登録する場合には、オブジェ
2895 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
2896 スを設定する。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
2897 あり、clsidにTCLS_SELF(=0)を指定するか、オブジェクト属性に
2898 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
2899 できる。

2900

2901 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、
2902 静的APIを囲むクラスに属するオブジェクトにのみ割り付けられる。これらの静
2903 的APIは、確保したID番号を割り付けるオブジェクトの属すべきクラスの囲み
2904 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、
2905 E_RSATRエラーとなる。

2906

2907 【補足説明】

2908

2909 この仕様では、カーネルオブジェクトの属するクラスを参照する機能は用意し
2910 ていない。

2911

2912 【仕様決定の理由】

2913

2914 カーネルオブジェクトをサービスコールによって登録する場合に、オブジェク
2915 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
2916 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
2917 タを増やさない方が望ましいためである。

2918

2919 2.10.7 オブジェクトの状態参照

2920

2921 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
2922 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
2923 (ref_yyy, get_yyy) を用意する。

2924

2925 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
2926 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）
2927 を参照するための機能は用意しないことを原則とする。自タスクの拡張情報の
2928 参照するサービスコール (get_inf) は、この原則に対する例外である。

2929

2930 2.11 オブジェクトのアクセス保護

2931

2932 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
2933 内容は、保護機能対応カーネルにのみ適用される。

2934

2935 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

2936

2937 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
2938 れたアクセス許可ベクタによって保護される。ただし、アクセス許可ベクタを
2939 持たないオブジェクトに対するアクセスは、システム状態に対するアクセス許
2940 可ベクタによって保護される。また、オブジェクトを登録するサービスコール
2941 と、特定のオブジェクトに関連しないシステムの状態に対するアクセスについ
2942 ては、システム状態のアクセス許可ベクタによって保護される。

2943

2944 アクセス許可ベクタによって許可されていないアクセス（アクセス違反）は、
2945 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。

2946

2947 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
2948 して、許可されていないアクセスを行おうとした場合、サービスコールから
2949 E_OACVエラーが返る。また、メモリオブジェクトに対して、許可されていない
2950 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

2951 ラーが返る.

2952

2953 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
2954 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
2955 た場合、CPU例外ハンドラが起動される。どのCPU例外ハンドラが起動されるか
2956 は、ターゲット定義である。ターゲットによっては、エミュレートされたCPU例
2957 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
2958 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
2959 外ハンドラを、メモリアクセス違反ハンドラと呼ぶ。

2960

2961 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
2962 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
2963 E_MACVエラーが返るか、メモリアクセス違反ハンドラが起動される。E_MACVエ
2964 ラーが返るかメモリアクセス違反ハンドラされるかは、ターゲット定義である。

2965

2966 メモリアクセス違反ハンドラでは、アクセス違反を発生させたアクセスに関す
2967 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）
2968 を参照する方法を、ターゲット定義で用意する。

2969

2970 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、
2971 カーネルドメイン以外の保護ドメインから、書込みアクセスまたは読出しアク
2972 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
2973 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。

2974

2975 【未決定事項】

2976

2977 マルチプロセッサ対応カーネルにおいて、システム状態のアクセス許可ベクタ
2978 をシステム全体で1つ持つかプロセッサ毎に持つかは、今後の課題である。

2979

2980 【μ ITRON4.0/PX仕様との関係】

2981

2982 μ ITRON4.0/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
2983 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
2984 ト定義の制限以外については規定していない。

2985

2986 【仕様決定の理由】

2987

2988 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
2989 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
2990 可ベクタが設定されていないためである。

2991

2992 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

2993

2994 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）
2995 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
2996 合がある。

2997

2998 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。

2999

3000 (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

3001 アクセスを含む)のみを許可する。これを、専有リードオンリー (private
3002 read only) と呼ぶ。

3003

3004 (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
3005 しアクセス (実行アクセスを含む) を許可する。これを、専有リードライ
3006 ト (private read/write) と呼ぶ。

3007

3008 (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
3009 許可する。これを、共有リードオンリー (shared read only) と呼ぶ。

3010

3011 (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
3012 セスを含む) を許可する。これを、共有リードライト (shared read/write)
3013 と呼ぶ。

3014

3015 (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
3016 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
3017 アクセス (実行アクセスを含む) のみを許可する。これを、共有リード専
3018 有ライト (shared read private write) と呼ぶ。

3019

3020 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
3021 の数が制限される場合がある。

3022

3023 2. 11. 3 デフォルトのアクセス許可ベクタ

3024

3025 静的APIによりカーネルオブジェクトに登録した直後は、次に規定されるデフォ
3026 ルトのアクセス許可ベクタが設定される。

3027

3028 保護ドメインに属するカーネルオブジェクトに対しては、4つの種別のアクセス
3029 がいずれも、その保護ドメインのみに許可される。すなわち、カーネルドメイ
3030 ンに属するオブジェクトに対しては、4つのアクセス許可パターンがいずれも
3031 TACP_KERNELに、ユーザドメインに属するオブジェクトに対しては、4つのアク
3032 セス許可パターンがいずれもTACP(domid) (domidはオブジェクトが属する保護
3033 ドメインのID番号) に設定される。

3034

3035 無所属のカーネルオブジェクトに対しては、4つの種別のアクセスがいずれも、
3036 すべての保護ドメインに許可される。すなわち、4つのアクセス許可パターンが
3037 いずれも、TACP_SHAREDに設定される。

3038

3039 システム状態のアクセス許可ベクタは、4つの種別のアクセスがいずれも、カー
3040 ネルドメインのみに許可される。すなわち、4つのアクセス許可パターンがいず
3041 れも、TACP_KERNELに設定される。

3042

3043 【未決定事項】

3044

3045 サービスコールによりカーネルオブジェクトに登録した直後のアクセス許可ベ
3046 クタについては、今後の課題である。

3047

3048 2. 11. 4 アクセス許可ベクタの設定

3049

3050 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

3051 クトのアクセス許可ベクタを設定する静的API (SAC_YYY) とサービスコール
3052 (sac_yyy) が用意されている。また、システム状態のアクセス許可ベクタを設定
3053 する静的API (SAC_SYS) とサービスコール (sac_sys) が用意されている。

3054

3055 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)
3056 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
3057 を設定しようとした場合には、E_OBJエラーとなる。

3058

3059 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
3060 されておらず、オブジェクトの登録と同時にアクセス許可ベクタを設定する静
3061 的API (ATA_YYY) が用意されている。

3062

3063 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
3064 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。そのため、ア
3065 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
3066 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
3067 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って
3068 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
3069 更しても、呼び出した時点ですでにミューテックをロックしていたタスクには
3070 影響しない。

3071

3072 なお、動的生成対応カーネル以外では、アクセス許可ベクタを設定するサービ
3073 スコール (sac_yyy) はサポートされない。

3074

3075 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
3076 する機能は用意していない。

3077

3078 【μITRON4.0/PX仕様との関係】

3079

3080 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
3081 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
3082 API (SAC_YYY) をサポートすることとした。

3083

3084 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
3085 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。

3086

3087 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
3088 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。
3089 これは、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を参照する
3090 ための機能は用意しないという原則に合わせるための修正である。

3091

3092 2.11.5 カーネルの管理領域のアクセス保護

3093

3094 カーネルが動作するために、カーネルの内部で用いるメモリ領域を、カーネル
3095 の管理領域と呼ぶ。ユーザタスクからカーネルを保護するためには、カーネル
3096 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。
3097 そのため、カーネルの管理領域は、4つの種別のアクセスがカーネルドメインの
3098 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
3099 クトと呼ぶ）の中に置かれる。

3100

3101 カーネルの用いるオブジェクト管理領域（カーネルの管理領域に該当する。
3102 「2.10.4 オブジェクト生成に必要なメモリ領域」の節を参照）として、カーネ
3103 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
3104 ラーとなる。また、カーネルの用いるオブジェクト管理領域の先頭番地にNULL
3105 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
3106 中に確保される。
3107
3108 システムタスクのスタック領域、ユーザタスクのシステムスタック領域、非タ
3109 スクコンテキスト用のスタック領域は、カーネルの用いるオブジェクト管理領
3110 域には該当しないが、カーネルドメインの実行中にのみアクセスされるため、
3111 カーネルの用いるオブジェクト管理領域と同様の扱いとなる。一方、ユーザタ
3112 スクのユーザスタック領域と固定長メモリプール領域は、ユーザドメインの実
3113 行中にもアクセスされるため、カーネルの用いるオブジェクト管理領域とは異
3114 なる扱いとなる。
3115
3116 2.11.6 ユーザタスクのユーザスタック領域
3117
3118 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
3119 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
3120 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
3121 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が、他のメモリオ
3122 ブジェクトとは異なり、次のように扱われる。
3123
3124 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
3125 よび読出しアクセスを行うことができる。そのため、書込みアクセスと読出し
3126 アクセス（実行アクセスを含む）に対するアクセス許可パターンは意味を持た
3127 ない。ユーザスタック領域に対して実行アクセスを行えるかどうかは、ターゲッ
3128 ト定義である。
3129
3130 ただし、上記の仕様を実現するために大きいオーバーヘッドを生じる場合には、
3131 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
3132 ドメインのみからアクセスできるものとする場合がある。
3133
3134 【 μ ITRON4.0/PX仕様との関係】
3135
3136 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
3137 きるものとした。
3138
3139 2.12 システムコンフィギュレーション手順
3140
3141 2.12.1 システムコンフィギュレーションファイル
3142
3143 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
3144 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
3145 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
3146 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
3147 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。
3148
3149 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
3150 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

3151 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
3152 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
3153 ることができる。

3154
3155 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
3156 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、
3157 その文法は次のいずれかである（両者の違いは、指定されたファイルを探すディ
3158 レクトリの違いのみ）。

3159
3160 INCLUDE("ファイル名");
3161 INCLUDE(<ファイル名>);
3162

3163 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
3164 の記述を、システムコンフィギュレーションファイルの一部として解釈する。
3165 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
3166 ルの静的API、システムサービスの静的API、コンフィギュレータに対する
3167 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
3168 と条件ディレクティブのみを記述することができる。

3169
3170 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
3171 を解釈するために必要なC言語のヘッダファイルを指定するために用いる。また、
3172 条件ディレクティブは、有効とする静的APIを選択するために用いることができ
3173 る。ただし、インクルードディレクティブは、コンフィギュレータが生成する
3174 ファイルでは先頭に集められる。そのため、条件ディレクティブの中にインク
3175 ルードディレクティブを記述しても、インクルードディレクティブは常に有効
3176 となる。また、1つの静的APIの記述の途中に、条件ディレクティブを記述する
3177 ことはできない。

3178
3179 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
3180 APIを、その記述順に解釈する。そのため例えば、タスクを生成する静的APIの
3181 前に、そのタスクにタスク例外処理ルーチンを定義する静的APIが記述されてい
3182 た場合、タスク例外処理ルーチンを定義する静的APIがE_NOEXSエラーとなる。

3183 3184 【μ ITRON4.0仕様との関係】 3185

3186 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
3187 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
3188 レクティブを設けた。また、共通静的APIを廃止した。μ ITRON4.0仕様における
3189 #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
3190 換わる。逆に、μ ITRON4.0仕様におけるINCLUDE静的APIの役割は、この仕様で
3191 は#includeディレクティブに置き換わる。

3192 3193 2.12.2 静的APIの文法とパラメータ 3194

3195 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
3196 述する。すなわち、静的APIの名称に続けて、静的APIの各パラメータを","で区
3197 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、
3198 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
3199 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
3200 で記述する。

3201
3202 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
3203 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。
3204
3205 静的APIのパラメータは、次の4種類に分類される。
3206
3207 (a) オブジェクト識別名
3208
3209 オブジェクトのID番号を指定するパラメータ。オブジェクトの名称を表す単一
3210 の識別名のみを記述することができる。
3211
3212 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
3213 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、
3214 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
3215 ディレクティブ (#define) を生成する。
3216
3217 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
3218 場合 (カーネルの静的APIでは、SAC_TSKやDEF_TEXのtskidパラメータ等がこれ
3219 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
3220 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
3221ギュレータがエラーを報告する。
3222
3223 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
3224 はできない。
3225
3226 (b) 整数定数式パラメータ
3227
3228 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、
3229 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
3230 決まる整数定数式を記述することができる。
3231
3232 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
3233 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
3234 クルードするファイルに含まれていなければならない。
3235
3236 (c) 一般定数式パラメータ
3237
3238 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
3239 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。
3240
3241 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
3242 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
3243ードするファイルに含まれていなければならない。
3244
3245 (d) 文字列パラメータ
3246
3247 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。
3248 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。
3249
3250 【μ ITRON4.0仕様との関係】

3251
3252 μ ITRON4.0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
3253 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。
3254
3255 (A) 自動割付け対応整数値パラメータ
3256 (B) 自動割付け非対応整数値パラメータ
3257 (C) プリプロセッサ定数式パラメータ
3258 (D) 一般定数式パラメータ
3259
3260 この仕様の(a)が、おおよそ μ ITRON4.0仕様の(A)に相当するが、(a)には整数値
3261 を記述できない点異なる。(b)～(c)と(B)～(D)の間には単純な対応関係がな
3262 いが、記述できる定数式の範囲には、(B) \subset (C) \subset (b) \subset (c) = (D)の関係がある。
3263
3264 μ ITRON4.0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
3265 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
3266 るC言語プリプロセッサの条件ディレクティブ(#if)の中に含めたい可能性のあ
3267 るパラメータを(C)としていた。
3268
3269 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
3270 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
3271 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
3272 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
3273 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。
3274
3275 2.12.3 保護ドメインの指定
3276
3277 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
3278 クトが属する保護ドメインの囲みの中に記述する。無所属のオブジェクトに登
3279 録する静的APIは、保護ドメインの囲みの外に記述する。保護ドメインに属すべ
3280 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
3281 合には、コンフィギュレータがE_RSATRエラーを報告する。
3282
3283 ユーザドメインの囲みの文法は次の通り。
3284
3285 DOMAIN(保護ドメイン名) {
3286 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
3287 }
3288
3289 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
3290 ることができる。
3291
3292 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
3293 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
3294 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
3295 サのディレクティブ(#define)を生成する。また、ユーザドメインの囲みの中
3296 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
3297 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。
3298
3299 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
3300 ンIDの割付けのみを行うことができる。

3301
3302 カーネルドメインの囲みの文法は次の通り.

3303
3304 KERNEL_DOMAIN {
3305 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
3306 }

3307
3308 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
3309 囲みを、複数回記述してもよい. 保護機能対応でないカーネルで保護ドメイン
3310 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
3311 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.

3312 【μ ITRON4.0/PX仕様との関係】

3313
3314 ユーザドメインの囲みの文法を変更した.

3315
3316 【仕様決定の理由】

3317
3318 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
3319 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
3320 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
3321 クト属性によって指定するためである.

3322
3323
3324 2.12.4 クラスの指定

3325
3326 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
3327 のオブジェクトが属するクラスの囲みの中に記述する. クラスに属すべきオブ
3328 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には、コン
3329 フィギュレータがE_RSATRエラーを報告する.

3330
3331 クラスの囲みの文法は次の通り.

3332
3333 CLASS(クラスID) {
3334 クラスに属するオブジェクトを登録する静的API等
3335 }

3336
3337 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
3338 ができる. 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
3339 E_IDエラーを報告する.

3340
3341 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. マルチプロセッ
3342 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
3343 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.

3344
3345 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
3346 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.

3347
3348 【仕様決定の理由】

3349
3350 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

3351 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
3352 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
3353 定するためである。

3354

3355 2.12.5 コンフィギュレータの処理モデル

3356

3357 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
3358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
3359成する（図2-8）。

3360

3361 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
3362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
3363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
3364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
3365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
3366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
3367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
3368に文法エラーや未サポートの記述があった場合には、この段階で検出される。

3369

3370 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
3371ルをコンパイルし、ロードモジュールを生成する。また、それをSレコードフォー
3372マットの形（cfg1_out.srec）に変換し、ロードモジュール中の各シンボルとア
3373ドレスの対応表を含むシンボルファイル（cfg1_out.syms）を生成する。静的
3374APIのパラメータに解釈できない式が記述された場合には、この段階でエラーが
3375検出される。

3376

3377 コンフィギュレータのパス2では、パス1で生成されたオブジェクトファイルを
3378Sレコードフォーマットの形に変換したものとシンボルファイルから、C言語プ
3379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
3380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
3381ムサービスの構成・初期化ファイル（kernel_cfg.cなど）と構成・初期化ヘッ
3382ダファイル（kernel_cfg.hなど）を生成する。構成・初期化ヘッダファイルに
3383は、登録できるオブジェクトの数（動的生成対応カーネル以外では、静的APIに
3384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
3385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
3386階でエラーが検出される。

3387

3388 パス2で生成されたこれらのファイルを、他のソースファイルとあわせてコンパ
3389イルし、アプリケーションのロードモジュールを生成する。また、それをSレコ
3390ードフォーマットの形（system.srec）に変換し、ロードモジュール中の各シンボ
3391ルと番地の対応表を含むシンボルファイル（system.syms）を生成する。

3392

3393 コンフィギュレータのパス3では、パス1で生成されたロードモジュールをSレコ
3394ードフォーマットの形に変換したものとシンボルファイル、パス2で生成されたロ
3395ードモジュールをSレコードフォーマットの形に変換したものとシンボルファイル
3396から、静的APIパラメータの値などを取り出し、妥当性のチェックを行う。静的
3397API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
3398れる。

3399

3400 保護機能対応カーネルにおいては、メモリ保護のための設定情報を生成するた

めに、パス3ではじめて得られる情報が必要となる。そこで、そのようなメモリ保護のための設定情報は、パス3においてメモリ構成・初期化ファイル(kernel_mem.c)に生成する。生成したメモリ構成・初期化ファイルは、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。

そのため、パス2で生成されたファイルから生成し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、仮のロードモジュールと最終的なロードモジュールでサイズが変化してはならないため、パス3でメモリ構成・初期化ファイルに生成するのと同じサイズのデータ構造を、パス2において仮のメモリ構成・初期化ファイル(kernel_mem2.c)に生成し、これも含めて仮のロードモジュールを生成しておく。また、仮のロードモジュールをSレコードフォーマットの形に変換したもの(cfg2_out.srec)、仮のロードモジュール中の各シンボルと番地の対応表を含むシンボルファイル(cfg2_out.syms)も、混乱を避けるためにファイル名を変更しておく(図2-9)。

パス3でメモリ保護のための設定情報を生成した場合には、パス4を実行する。コンフィギュレータのパス4では、パス1で生成されたロードモジュールをSレコードフォーマットの形に変換したものとシンボルファイル、パス3で生成されたロードモジュールをSレコードフォーマットの形に変換したものとシンボルファイルから、生成したロードモジュールの妥当性のチェックを行う。この段階で検出されるエラーは、コンフィギュレーション処理の不具合を示すものである。

【μITRON4.0仕様との関係】

コンフィギュレータの処理モデルは全面的に変更した。

2.12.6 静的APIのパラメータに関するエラー検出

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

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

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

【μITRON4.0仕様との関係】

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

3451
3452 2.12.7 オブジェクトのID番号の指定
3453
3454 コンフィギュレータのオプション機能として、アプリケーション設計者がオブ
3455 ジェクトのID番号を指定するための次の機能を用意する。
3456
3457 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
3458 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
3459 クトにID番号を割り付ける。それに従ったID番号割付けができない場合（ID番
3460 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。
3461
3462 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
3463 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
3464 レータに渡すファイルと同じフォーマットで生成する。
3465
3466 【 μ ITRON4.0仕様との関係】
3467
3468 μ ITRON4.0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
3469 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。
3470
3471 2.13 TOPPERSネーミングコンベンション
3472
3473 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
3474 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
3475 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
3476 ル内部の名称に適用してもよい。
3477
3478 2.13.1 モジュール識別名
3479
3480 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
3481 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
3482 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。
3483
3484 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
3485 ジュール識別名は“sil”とする。
3486
3487 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
3488 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
3489 識別名を含めない名称を使用する。
3490
3491 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
3492 したものをWWWと表記する。
3493
3494 2.13.2 データ型名
3495
3496 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
3497 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
3498 “_t”である名称とする。
3499
3500 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、

3501 "_"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
3502 が"_T"である名称とする場合もある。

3503
3504 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。

3505
3506 (A) パケットのデータ型

3507		
3508	T_CYYY	acre_yyyに渡すパケットのデータ型
3509	T_DYYY	def_yyyに渡すパケットのデータ型
3510	T_RYYY	ref_yyyに渡すパケットのデータ型
3511	T_WWW_CYYY	www_acre_yyyに渡すパケットのデータ型
3512	T_WWW_DYYY	www_def_yyyに渡すパケットのデータ型
3513	T_WWW_RYYY	www_ref_yyyに渡すパケットのデータ型

3514

3515 2.13.3 関数名

3516
3517 関数の名称は、英小文字、数字、"_"で構成する。

3518
3519 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。

3520
3521 (A) サービスコール

3522
3523 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
3524 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
3525 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、
3526 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
3527 は、zzxxx_yyyまたはwww_zzxxx_yyyの名称となる。

3528

3529 非タスクコンテキスト専用のサービスコールの名称は、派生したことを表す文
3530 字として"i"を付加し、ixxx_yyy, izxxx_yyy, www_ixxx_yyy, www_izxxx_yyyと
3531 いった名称とする。

3532

3533 【補足説明】

3534

3535 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
3536 いては、「5.9 省略名の元になった英語」の節を参照すること。

3537

3538 (B) コールバック

3539

3540 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。

3541

3542 2.13.4 変数名

3543

3544 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
3545 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。

3546

3547 変数の名称に関して、次のガイドラインを設ける。

3548

3549	~id	~ID (オブジェクトのID番号, ID型)
3550	~no	~番号 (オブジェクト番号)

3551	～atr	～属性（オブジェクト属性，ATR型）
3552	～stat	～状態（オブジェクト状態，STAT型）
3553	～mode	～モード（サービスコールの動作モード，MODE型）
3554	～pri	～優先度（優先度，PRI型）
3555	～sz	～サイズ（単位はバイト数，SIZE型またはuint_t型）
3556	～cnt	～の個数（単位は個数，uint_t型）
3557	～ptn	～パターン
3558	～tim	～時刻，～時間
3559	～cd	～コード
3560	i～	～の初期値
3561	max～	～の最大値
3562	min～	～の最小値
3563	left～	～の残り

3564

3565 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
3566 ンを設ける.

3567

3568	p_～	ポインタ
3569	pp_～	ポインタを入れる領域へのポインタ
3570	pk_～	パケットへのポインタ
3571	ppk_～	パケットへのポインタを入れる領域へのポインタ

3572

3573 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.

3574

3575 (A) パケットへのポインタ

3576

3577	pk_cyyy	acre_yyyに渡すパケットへのポインタ
3578	pk_dyyy	def_yyyに渡すパケットへのポインタ
3579	pk_ryyy	ref_yyyに渡すパケットへのポインタ
3580	pk_www_cyyy	www_acre_yyyに渡すパケットへのポインタ
3581	pk_www_dyyy	www_def_yyyに渡すパケットへのポインタ
3582	pk_www_ryyy	www_ref_yyyに渡すパケットへのポインタ

3583

3584 2.13.5 定数名

3585

3586 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
3587 字，“_”で構成する.

3588

3589 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.

3590

3591 (A) メインエラーコード

3592

3593 メインエラーコードは，先頭が“E_”である名称とする.

3594

3595 (B) 機能コード

3596

3597	TFN_XXX_YYY	xxx_yyyの機能コード
3598	TFN_WWW_XXX_YYY	www_xxx_yyyの機能コード

3599

3600 (C) その他の定数

3601
3602 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、
3603 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
3604 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。

3605
3606 また、定数の名称に関して、次のガイドラインを設ける。

3607
3608 TA_～ オブジェクトの属性値
3609 TSZ_～ ～のサイズ
3610 TBIT_～ ～のビット数
3611 TMAX_～ ～の最大値
3612 TMIN_～ ～の最小値

3613

3614 2.13.6 マクロ名

3615

3616 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
3617 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
3618 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
3619 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
3620 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
3621 する場合もある。

3622

3623 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。

3624

3625 (A) 構成マクロ

3626

3627 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
3628 る。

3629

3630 TSZ_～ ～のサイズ
3631 TBIT_～ ～のビット数
3632 TMAX_～ ～の最大値
3633 TMIN_～ ～の最小値

3634

3635 2.13.7 静的API名

3636

3637 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
3638 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
3639 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
3640 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。

3641

3642 2.13.8 ファイル名

3643

3644 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
3645 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。
3646 また、“-”も使用しない。

3647

3648 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。

3649

3650 (A) ヘッダファイル

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

2.13.9 モジュール内部の名称の衝突回避

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

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

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

2.14 TOPPERS共通定義

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

2.14.1 TOPPERS共通ヘッダファイル

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

TOPPERS共通ヘッダファイルは、カーネルヘッダファイル（kernel.h）や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。

2.14.2 TOPPERS共通データ型

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

int8_t	符号付き8ビット整数（オプション、C99準拠）
uint8_t	符号無し8ビット整数（オプション、C99準拠）
int16_t	符号付き16ビット整数（C99準拠）
uint16_t	符号無し16ビット整数（C99準拠）
int32_t	符号付き32ビット整数（C99準拠）
uint32_t	符号無し32ビット整数（C99準拠）
int64_t	符号付き64ビット整数（オプション、C99準拠）
uint64_t	符号無し64ビット整数（オプション、C99準拠）
int128_t	符号付き128ビット整数（オプション、C99準拠）
uint128_t	符号無し128ビット整数（オプション、C99準拠）

3701	int_least8_t	8ビット以上の符号付き整数 (C99準拠)
3702	uint_least8_t	int_least8_t型と同じサイズの符号無し整数 (C99準拠)
3703		
3704	float32_t	IEEE754準拠の32ビット単精度浮動小数点数 (オプション)
3705	double64_t	IEEE754準拠の64ビット倍精度浮動小数点数 (オプション)
3706		
3707	bool_t	真偽値 (trueまたはfalse)
3708	int_t	16ビット以上の符号付き整数
3709	uint_t	int_t型と同じサイズの符号無し整数
3710	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
3711	ulong_t	long_t型と同じサイズの符号無し整数
3712		
3713	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)
3714	uintptr_t	intptr_t型と同じサイズの符号無し整数 (C99準拠)
3715		
3716	FN	機能コード (符号付き整数, int_tに定義)
3717	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)
3718		
3719	ID	オブジェクトのID番号 (符号付き整数, int_tに定義)
3720	ATR	オブジェクト属性 (符号無し整数, uint_tに定義)
3721	STAT	オブジェクトの状態 (符号無し整数, uint_tに定義)
3722	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)
3723	PRI	優先度 (符号付き整数, int_tに定義)
3724	SIZE	メモリ領域のサイズ (符号無し整数, ポインタを格納できるサイズの符号無し整数型に定義)
3725		
3726		
3727	TMO	タイムアウト指定 (符号付き整数, 単位はミリ秒, int_tに定義)
3728	RELTIM	相対時間 (符号無し整数, 単位はミリ秒, uint_tに定義)
3729	SYSTIM	システム時刻 (符号無し整数, 単位はミリ秒, ulong_tに定義)
3730	SYSUTM	性能評価用システム時刻 (符号無し整数, 単位はマイクロ秒, ulong_tに定義)
3731		
3732		
3733	FP	プログラムの起動番地 (型の定まらない関数ポインタ)
3734		
3735	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)
3736	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)
3737		
3738	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)
3739		
3740		
3741		
3742	MB_T	オブジェクト管理領域を確保するためのデータ型
3743		
3744	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)
3745		
3746	ACVCT	アクセス許可ベクタ
3747		
3748	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。	
3749	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。	
3750		

int8_t, uint8_t, int64_t, uint64_t, int128_t, uint128_t, float32_t, double64_tが使用できるかどうかは、ターゲットシステムに依存する。これら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。IEEE754準拠の浮動小数点数がサポートされていないターゲットシステムでは、float32_tとdouble64_tは使用できないものとする。

3758

3759 【μITRON4.0仕様との関係】

3760

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。

3764

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。

3768

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。

3776

TECSとの整合性を取るために、BOOLに代えて、bool_tを用いることにした。また、IEEE754準拠の単精度浮動小数点数を表す型としてfloat32_t, IEEE754準拠の64ビットを表す型としてdouble64_tを導入した。

3780

性能評価用システム時刻のためのデータ型としてSYSUTMを、オブジェクト管理領域を確保するためのデータ型としてMB_Tを用意することにした

3783

3784 2.14.3 TOPPERS共通定数

3785

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）。

3788

3789 (1) 一般定数

3790

3791	NULL		無効ポインタ
------	------	--	--------

3792

3793	true	1	真
3794	false	0	偽

3795

3796	E_OK	0	正常終了
------	------	---	------

3797

3798 【μITRON4.0仕様との関係】

3799

3800 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

3801 を用いることにした.

3802

3803 (2) 整数型に格納できる最大値と最小値

3804

3805	INT8_MAX	int8_tに格納できる最大値 (オプション, C99準拠)
3806	INT8_MIN	int8_tに格納できる最小値 (オプション, C99準拠)
3807	UINT8_MAX	uint8_tに格納できる最大値 (オプション, C99準拠)
3808	INT16_MAX	int16_tに格納できる最大値 (C99準拠)
3809	INT16_MIN	int16_tに格納できる最小値 (C99準拠)
3810	UINT16_MAX	uint16_tに格納できる最大値 (C99準拠)
3811	INT32_MAX	int32_tに格納できる最大値 (C99準拠)
3812	INT32_MIN	int32_tに格納できる最小値 (C99準拠)
3813	UINT32_MAX	uint32_tに格納できる最大値 (C99準拠)
3814	INT64_MAX	int64_tに格納できる最大値 (オプション, C99準拠)
3815	INT64_MIN	int64_tに格納できる最小値 (オプション, C99準拠)
3816	UINT64_MAX	uint64_tに格納できる最大値 (オプション, C99準拠)
3817	INT128_MAX	int128_tに格納できる最大値 (オプション, C99準拠)
3818	INT128_MIN	int128_tに格納できる最小値 (オプション, C99準拠)
3819	UINT128_MAX	uint128_tに格納できる最大値 (オプション, C99準拠)

3820

3821	INT_LEAST8_MAX	int_least8_tに格納できる最大値 (C99準拠)
3822	INT_LEAST8_MIN	int_least8_tに格納できる最小値 (C99準拠)
3823	UINT_LEAST8_MAX	uint_least8_tに格納できる最大値 (C99準拠)
3824	INT_MAX	int_tに格納できる最大値 (C90準拠)
3825	INT_MIN	int_tに格納できる最小値 (C90準拠)
3826	UINT_MAX	uint_tに格納できる最大値 (C90準拠)
3827	LONG_MAX	long_tに格納できる最大値 (C90準拠)
3828	LONG_MIN	long_tに格納できる最小値 (C90準拠)
3829	ULONG_MAX	ulong_tに格納できる最大値 (C90準拠)
3830		
3831	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)
3832		
3833	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)
3834		
3835	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)
3836		
3837	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)
3838		

3839

3840 (3) 整数型のビット数

3841

3842	CHAR_BIT	char型のビット数 (C90準拠)
------	----------	--------------------

3843

3844 (4) オブジェクト属性

3845

3846	TA_NULL	OU	オブジェクト属性を指定しない
------	---------	----	----------------

3847

3848 (5) タイムアウト指定

3849

3850	TMO_POL	0	ポーリング
------	---------	---	-------

3851	TMO_FEVR	-1	永久待ち
3852	TMO_NBLK	-2	ノンブロッキング
3853			
3854	(6) アクセス許可パターン		
3855			
3856	TACP_KERNEL	0U	カーネルドメインのみにアクセスを許可
3857	TACP_SHARED	~0U	すべての保護ドメインにアクセスを許可
3858			
3859	2.14.4 TOPPERS共通エラーコード		
3860			
3861	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.		
3862			
3863	(A) 内部エラークラス (EC_SYS, -5~-8)		
3864			
3865	E_SYS	-5	システムエラー
3866			
3867	(B) 未サポートエラークラス (EC_NOSPT, -9~-16)		
3868			
3869	E_NOSPT	-9	未サポート機能
3870	E_RSFN	-10	予約機能コード
3871	E_RSATR	-11	予約属性
3872			
3873	(C) パラメータエラークラス (EC_PAR, -17~-24)		
3874			
3875	E_PAR	-17	パラメータエラー
3876	E_ID	-18	不正ID番号
3877			
3878	(D) 呼出しコンテキストエラークラス (EC_CTX, -25~-32)		
3879			
3880	E_CTX	-25	コンテキストエラー
3881	E_MACV	-26	メモリアクセス違反
3882	E_OACV	-27	オブジェクトアクセス違反
3883	E_ILUSE	-28	サービスコール不正使用
3884			
3885	(E) 資源不足エラークラス (EC_NOMEM, -33~-40)		
3886			
3887	E_NOMEM	-33	メモリ不足
3888	E_NOID	-34	ID番号不足
3889	E_NORES	-35	資源不足
3890			
3891	(F) オブジェクト状態エラークラス (EC_OBJ, -41~-48)		
3892			
3893	E_OBJ	-41	オブジェクト状態エラー
3894	E_NOEXS	-42	オブジェクト未登録
3895	E_QOVR	-43	キューイングオーバーフロー
3896			
3897	(G) 待ち解除エラークラス (EC_RLWAI, -49~-56)		
3898			
3899	E_RLWAI	-49	待ち禁止状態または待ち状態の強制解除
3900	E_TMOUT	-50	ポーリング失敗またはタイムアウト

3901 E_DLT -51 待ちオブジェクトの削除または再初期化
 3902 E_CLS -52 待ちオブジェクトの状態変化
 3903

3904 (H) 警告クラス (EC_WARN, -57~-64)
 3905

3906 E_WBLK -57 ノンブロッキング受け付け
 3907 E_BOVR -58 バッファオーバーフロー
 3908

3909 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、
 3910 サービスコールがエラーコードを返した場合には副作用がないという原則の例
 3911 外となる。
 3912

3913 【 μ ITRON4.0仕様との関係】
 3914

3915 E_NORESは、 μ ITRON4.0仕様に規定されていないエラーコードである。
 3916

3917 2.14.5 TOPPERS共通マクロ

3918 (1) 整数定数を作るマクロ
 3919

3920
 3921 INT8_C(val) int_least8_t型の定数を作るマクロ (C99準拠)
 3922 UINT8_C(val) uint_least8_t型の定数を作るマクロ (C99準拠)
 3923 INT16_C(val) int16_t型の定数を作るマクロ (C99準拠)
 3924 UINT16_C(val) uint16_t型の定数を作るマクロ (C99準拠)
 3925 INT32_C(val) int32_t型の定数を作るマクロ (C99準拠)
 3926 UINT32_C(val) uint32_t型の定数を作るマクロ (C99準拠)
 3927 INT64_C(val) int64_t型の定数を作るマクロ (オプション, C99準拠)
 3928 UINT64_C(val) uint64_t型の定数を作るマクロ (オプション, C99準拠)
 3929 INT128_C(val) int128_t型の定数を作るマクロ (オプション, C99準拠)
 3930 UINT128_C(val) uint128_t型の定数を作るマクロ (オプション, C99準拠)
 3931
 3932 UINT_C(val) uint_t型の定数を作るマクロ
 3933 ULONG_C(val) ulong_t型の定数を作るマクロ
 3934

3935 【仕様決定の理由】
 3936

3937 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
 3938 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
 3939 ロを使って記述する必要はない。
 3940

3941 (2) 型に関する情報を取り出すためのマクロ
 3942

3943 offsetof(structure, field) 構造体structure中のフィールドfieldの
 3944 バイト位置を返すマクロ (C90準拠)
 3945
 3946 alignof(type) 型typeのアラインメント単位を返すマクロ
 3947
 3948 ALIGN_TYPE(addr, type) 番地addrが型typeに対してアラインしてい
 3949 るかどうかを返すマクロ
 3950

3951 (3) assertマクロ
3952
3953 assert(exp) expが成立しているかを検査するマクロ (C90準拠)
3954
3955 (4) コンパイラの拡張機能のためのマクロ
3956
3957 inline インライン関数
3958 Inline ファイルローカルなインライン関数
3959 asm インラインアセンブラ
3960 Asm インラインアセンブラ (最適化抑止)
3961 throw() 例外を発生しない関数
3962 NoReturn リターンしない関数
3963
3964 (5) エラーコード構成・分解マクロ
3965
3966 ERCD(mercd, sercd) メインエラーコードmercdとサブエラーコードsercdか
3967 ら、エラーコードを構成するためのマクロ
3968
3969 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
3970 ためのマクロ
3971 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
3972 めのマクロ
3973
3974 (6) アクセス許可パターン構成マクロ
3975
3976 TACP(domid) domidで指定されるユーザドメインのみにアクセスを
3977 許可するアクセス許可パターンを構成するためのマ
3978 クロ
3979
3980 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
3981 することができる。TDOM_SELF, TDOM_KERNEL, TDOM_NONEを指定した場合の動作
3982 は、保証されない。
3983
3984 2.14.6 TOPPERS共通構成マクロ
3985
3986 (1) 相対時間の範囲
3987
3988 TMAX_RELTIM 相対時間に指定できる最大値
3989
3990 2.15 カーネル共通定義
3991
3992 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。
3993
3994 2.15.1 カーネルヘッダファイル
3995
3996 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)
3997 およびそこからインクルードされるファイルに含まれている。カーネルを用い
3998 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。
3999
4000 ただし、カーネルを用いるために必要な定義の中で、コンフィギュレータによっ

て生成されるものは、カーネル構成・初期化ヘッダファイル (kernel_cfg.h) に含まれる。具体的には、登録できるオブジェクトの数 (TNUM_YYY) やオブジェクトのID番号などの定義が、これに該当する。これらの定義を用いる場合には、カーネル構成・初期化ヘッダファイルをインクルードする。

μ ITRON4.0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。

【 μ ITRON4.0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、 μ ITRON4.0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

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

【 μ ITRON4.0仕様との関係】

値が0のオブジェクト属性 (TA_HLNG, TA_TFIFO, TA_MFIFO, 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。TA_MPRIは、メールボックス機能でのみ使用するため、カーネル共通定義から外した。

(2) 保護ドメインID

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

(3) その他のカーネル共通定数

TCLS_SELF	0	自タスクの属するクラス
TPRC_NONE	0	割付けプロセッサの指定がない
TPRC_INI	0	初期割付けプロセッサ
TSK_SELF	0	自タスク指定
TSK_NONE	0	該当するタスクがない
TPRI_SELF	0	自タスクのベース優先度の指定
TPRI_INI	0	タスクの起動時優先度の指定

4051 TIPM_ENAALL 0 割込み優先度マスク全解除

4052

4053 (4) カーネルで用いるメインエラーコード

4054

4055 「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
4056 で、E_CLS、E_WBLK、E_BOVRの3つは、カーネルでは使用しない。

4057

4058 【TOPPERS/ASPカーネルにおける規定】

4059

4060 ASPカーネルでは、サービスコールから、E_RSFN、E_RSATR、E_MACV、E_OACV、
4061 E_NOMEM、E_NOID、E_NORES、E_NOEXSが返る状況は起こらない。E_RSATRは、コ
4062 ンフィギュレータによって検出される。ただし、動的生成機能拡張パッケージ
4063 では、E_RSATR、E_NOMEM、E_NOID、E_NOEXSが返る状況が起こる。

4064

4065 【TOPPERS/FMPカーネルにおける規定】

4066

4067 FMPカーネルでは、サービスコールから、E_RSFN、E_RSATR、E_MACV、E_OACV、
4068 E_NOMEM、E_NOID、E_NORES、E_NOEXSが返る状況は起こらない。E_RSATRと
4069 E_NORESは、コンフィギュレータによって検出される。

4070

4071 【TOPPERS/HRP2カーネルにおける規定】

4072

4073 HRP2カーネルでは、サービスコールから、E_RSATR、E_NOID、E_NORES、
4074 E_NOEXSが返る状況は起こらない。E_RSATRは、コンフィギュレータによって検
4075 出される。

4076

4077 【TOPPERS/SSPカーネルにおける規定】

4078

4079 SSPカーネルでは、サービスコールから、E_RSFN、E_RSATR、E_MACV、E_OACV、
4080 E_ILUSE、E_NOMEM、E_NOID、E_NORES、E_NOEXS、E_RLWAI、E_TMOUT、E_DLTが返
4081 る状況は起こらない。E_RSATRは、コンフィギュレータによって検出される。

4082

4083 2.15.3 カーネル共通マクロ

4084

4085 (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

4086

4087 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
4088 用意している。

4089

4090 STK_T スタック領域を確保するためのデータ型

4091

4092 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
4093 STK_T型の配列の要素数

4094 ROUND_STK_T(sz) 要素数COUNT_STK_T(sz)のSTK_T型の配列のサイズ (sz
4095 を、STK_T型のサイズの倍数になるように大きい方に
4096 丸めた値)

4097

4098 これらを用いてスタック領域を確保する方法は次の通り。

4099

4100 STK_T <スタック領域の変数名>[COUNT_STK_T(<スタック領域のサイズ>)];

4101
4102 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
4103 には、スタック領域の先頭番地に<スタック領域の変数名>を、スタック領域の
4104 サイズにROUND_STK_T(<スタック領域のサイズ>)を指定する。
4105

4106 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
4107 タック領域を確保することはできない。詳しくは、「4.1 タスク管理機能」の
4108 節のCRE_TSKの機能の項を参照すること。
4109

4110 (2) オブジェクト属性を作るマクロ

4111
4112 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
4113 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。
4114

4115 TA_DOM(domid) domidで指定される保護ドメインに属する
4116

4117 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
4118 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。
4119

4120 TA_CLS(clsid) clsidで指定されるクラスに属する
4121

4122 (3) サービスコールの呼出し方法を指定するマクロ

4123
4124 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
4125 クロとして、次のマクロを用意している。
4126

4127 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
4128 て呼び出すための名称
4129

4130 2.15.4 カーネル共通構成マクロ

4131 (1) サポートする機能

4132
4133 TOPPERS_SUPPORT_PROTECT 保護機能対応のカーネル
4134 TOPPERS_SUPPORT_MULTI_PRC マルチプロセッサ対応のカーネル
4135 TOPPERS_SUPPORT_DYNAMIC_CRE 動的生成対応のカーネル
4136
4137

4138 【未決定事項】

4139
4140 マクロ名は、今後変更する可能性がある。
4141

4142 (2) 優先度の範囲

4143
4144 TMIN_TPRI タスク優先度の最小値 (=1)
4145 TMAX_TPRI タスク優先度の最大値
4146

4147 【TOPPERS/ASPカーネルにおける規定】

4148
4149 ASPカーネルでは、タスク優先度の最大値 (TMAX_TPRI) は16に固定されている。
4150 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

4151 ことができる.

4152

4153 **【TOPPERS/FMPカーネルにおける規定】**

4154

4155 FMPカーネルでは、タスク優先度の最大値 (TMAX_TPRI) は16に固定されている.

4156

4157 **【TOPPERS/HRP2カーネルにおける規定】**

4158

4159 HRP2カーネルでは、タスク優先度の最大値 (TMAX_TPRI) は16に固定されている.

4160

4161 **【TOPPERS/SSPカーネルにおける規定】**

4162

4163 SSPカーネルでは、タスク優先度の最大値 (TMAX_TPRI) は16に固定されている.

4164

4165 **【 μ ITRON4.0仕様との関係】**

4166

4167 メッセージ優先度の最小値 (TMIN_MPRI) と最大値 (TMAX_MPRI) は、メールボッ
4168 クス機能でのみ使用するため、カーネル共通定義から外した.

4169

4170 (3) プロセッサの数

4171

4172 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らるためのマクロとし
4173 て、次の構成マクロを用意している.

4174

4175 TNUM_PRCID プロセッサの数

4176

4177 (4) 特殊な役割を持ったプロセッサ

4178

4179 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
4180 めのマクロとして、次の構成マクロを用意している.

4181

4182 TOPPERS_MASTER_PRCID マスタプロセッサのID番号
4183 TOPPERS_SYSTIM_PRCID システム時刻管理プロセッサのID番号 (グ
4184 ローバルタイマ方式の場合のみ)

4185

4186 (5) タイマ方式

4187

4188 マルチプロセッサ対応カーネルでは、システム時刻の方式を知るためのマクロ
4189 として、次の構成マクロを用意している.

4190

4191 TOPPERS_SYSTIM_LOCAL ローカルタイマ方式の場合にマクロ定義
4192 TOPPERS_SYSTIM_GLOBAL グローバルタイマ方式の場合にマクロ定義

4193

4194 (6) バージョン情報

4195

4196 TKERNEL_MAKER カーネルのメーカコード (=0x0118)

4197 TKERNEL_PRID カーネルの識別番号

4198 TKERNEL_SPVER カーネル仕様のバージョン番号

4199 TKERNEL_PRVER カーネルのバージョン番号

4200

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

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

0x0001	TOPPERS/JSPカーネル
0x0002	予約 (IIMPカーネル)
0x0003	予約 (IDLカーネル)
0x0004	TOPPERS/FI4カーネル
0x0005	TOPPERS/FDMPカーネル
0x0006	TOPPERS/HRPカーネル
0x0007	TOPPERS/ASPカーネル
0x0008	TOPPERS/FMPカーネル
0x0009	TOPPERS/SSPカーネル
0x000a	TOPPERS/ASP Safety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 (0xf5) が TOPPERS新世代カーネル仕様であることを、中位4ビットがメジャーバージョン番号、下位4ビットがマイナーバージョン番号を表す。

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

第3章 システムインタフェースレイヤAPI仕様

3.1 システムインタフェースレイヤの概要

システムインタフェースレイヤ（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。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。

- ・微少時間待ちの機能を呼び出すこと
- ・エンディアン取得のためのマクロを参照すること
- ・メモリ空間アクセス関数により、アクセスを許可されたメモリ領域にアクセスすること
- ・I/O空間アクセス関数により、アクセスを許可されたI/O領域にアクセスすること

3.2 SILヘッダファイル

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

4251

4252 3.3 全割込みロック状態の制御

4253

4254 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）
4255 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
4256 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）
4257 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。

4258

4259 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
4260 ための以下の機能を用意している。

4261

4262 (1) SIL_PRE_LOC

4263

4264 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。通常は、型と変数
4265 名を並べたもので、最後に";"を含まない。

4266

4267 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
4268 の変数宣言部に記述しなければならない。SIL_LOC_INT、SIL_UNL_INTを1つの関
4269 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
4270 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
4271 なければならない。そのように記述しなかった場合の動作は保証されない。

4272

4273 (2) SIL_LOC_INT()

4274

4275 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
4276 し、全割込みロック状態に遷移する。

4277

4278 (3) SIL_UNL_INT()

4279

4280 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。
4281 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。

4282

4283 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
4284 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
4285 ること。

4286

4287 【補足説明】

4288

4289 全割込みロック状態の制御機能の使用例は次の通り。

4290

```
4291 {  
4292     SIL_PRE_LOC;  
4293  
4294     SIL_LOC_INT();  
4295     // この間はNMIを除くすべての割込みがマスクされる。  
4296     // 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）。  
4297     SIL_UNL_INT();  
4298 }
```

4299

4300 3.4 SILスピンロック

4301
4302 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
4303 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
4304 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
4305 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、
4306 SILスピンロックと呼ぶ。
4307
4308 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
4309 全ての割込み（NMIを除く）がマスクされる。ロックが他のプロセッサに取得さ
4310 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。ロックの取
4311 得を待つ間は、割込みはマスクされない（ロックの取得を試みる前にマスクし
4312 ていた割込みは、マスク解除されない）。プロセッサ間ロックを取得し割込み
4313 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
4314 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
4315 という。
4316
4317 SILで取得・返却するプロセッサ間ロックは、システムに唯一存在する。
4318
4319 (1) SIL_PRE_LOC
4320
4321 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
4322 ロックの取得・解放にも兼用する。
4323
4324 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
4325 の変数宣言部に記述しなければならない。SIL_LOC_SPN、SIL_UNL_SPNを、同じ
4326 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、
4327 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
4328 宣言部にSIL_PRE_LOCを記述しなければならない。そのように記述しなかった場
4329 合の動作は保証されない。
4330
4331 (2) SIL_LOC_SPN()
4332
4333 SILスピンロックが取得されていない状態である場合には、プロセッサ間ロック
4334 の取得を試みる。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、
4335 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
4336 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。ロックの取得に成
4337 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
4338 移する。
4339
4340 (3) SIL_UNL_SPN()
4341
4342 プロセッサ間ロックを返却し、全割込みロックフラグを対応するSIL_LOC_SPNを
4343 実行する前の状態に戻す。
4344
4345 SILスピンロックを取得している状態でSIL_LOC_SPNを呼び出した場合の動作は
4346 保証されない。逆に、SILスピンロックを取得していない状態でSIL_UNL_SPNを
4347 呼び出した場合の動作も保証されない。
4348
4349 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
4350 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

4351 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。

4352

4353 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
4354 されていない。

4355

4356 **【使用上の注意】**

4357

4358 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、
4359 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
4360 べきである。

4361

4362 **【補足説明】**

4363

4364 SILスピンロック機能の使用例は次の通り。

4365

```
4366 {  
4367     SIL_PRE_LOC;  
4368  
4369     SIL_LOC_SPN();  
4370     // この間はSILスピンロックを取得している。  
4371     // この間はNMIを除くすべての割込みがマスクされる。  
4372     // 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）。  
4373     SIL_UNL_SPN();  
4374 }
```

4375

4376 3.5 微少時間待ち

4377

4378 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
4379 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、
4380 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
4381 ための以下の機能を用意している。

4382

4383 (1) void sil_dly_nse(ulong_t dlytim)

4384

4385 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。
4386 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
4387 意すること。

4388

4389 3.6 エンディアンの取得

4390

4391 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
4392 下のマクロを定義している。

4393

4394 (1) SIL_ENDIAN_BIG, SIL_ENDIAN_LITTLE

4395

4396 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
4397 プロセッサではSIL_ENDIAN_LITTLEを、マクロ定義している。

4398

4399 3.7 メモリ空間アクセス関数

4400

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。

(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。

(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。

(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。

(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。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。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。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。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。

4451
4452 (10) void sil_wrw_mem(uint32_t *mem, uint32_t data)
4453
4454 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。
4455
4456 (11) uint32_t sil_rew_lem(const uint32_t *mem)
4457
4458 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
4459 値を返す。リトルエンディアンプロセッサでは、sil_rew_memと一致する。ビッグ
4460 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
4461 た値を返す。
4462
4463 (12) void sil_wrw_lem(uint32_t *mem, uint32_t data)
4464
4465 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
4466 アンで書き込む。リトルエンディアンプロセッサでは、sil_wrw_memと一致する。
4467 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、
4468 sil_wrw_memで書き込むのと同じ結果となる。
4469
4470 (13) uint32_t sil_rew_bem(const uint32_t *mem)
4471
4472 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
4473 値を返す。ビッグエンディアンプロセッサでは、sil_rew_memと一致する。リト
4474 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
4475 た値を返す。
4476
4477 (14) void sil_wrw_bem(uint32_t *mem, uint32_t data)
4478
4479 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
4480 アンで書き込む。ビッグエンディアンプロセッサでは、sil_wrw_memと一致する。
4481 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、
4482 sil_wrw_memで書き込むのと同じ結果となる。
4483
4484 3.8 I/O空間アクセス関数
4485
4486 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
4487 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
4488 用意している。
4489
4490 (1) uint8_t sil_reb_iop(const uint8_t *iop)
4491 (2) void sil_wrb_iop(uint8_t *iop, uint8_t data)
4492 (3) uint16_t sil_reh_iop(const uint16_t *iop)
4493 (4) void sil_wrh_iop(uint16_t *iop, uint16_t data)
4494 (5) uint16_t sil_reh_lep(const uint16_t *iop)
4495 (6) void sil_wrh_lep(uint16_t *iop, uint16_t data)
4496 (7) uint16_t sil_reh_bep(const uint16_t *iop)
4497 (8) void sil_wrh_bep(uint16_t *iop, uint16_t data)
4498 (9) uint32_t sil_rew_iop(const uint32_t *iop)
4499 (10) void sil_wrw_iop(uint32_t *iop, uint32_t data)
4500 (11) uint32_t sil_rew_lep(const uint32_t *iop)

4501 (12) void sil_wrw_lep(uint32_t *iop, uint32_t data)

4502 (13) uint32_t sil_rew_bep(const uint32_t *iop)

4503 (14) void sil_wrw_bep(uint32_t *iop, uint32_t data)

4504

4505 3.9 プロセッサIDの参照

4506

4507 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
4508 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。

4509

4510 (1) void sil_get_pid(ID *p_prcid)

4511

4512 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、
4513 p_prcidで指定したメモリ領域に返す。

4514

4515 【使用上の注意】

4516

4517 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
4518 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
4519 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、
4520 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
4521 ている可能性があるためである。

4522

4523

4524 第4章 カーネルAPI仕様

4525

4526 この章では、カーネルのAPI仕様について規定する。

4527

4528 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
4529 記号を用いる。

4530

4531 [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
4532 トから呼び出すと、E_CTXエラーとなる。

4533

4534 [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
4535 トから呼び出すと、E_CTXエラーとなる。

4536

4537 [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
4538 のできるサービスコールを示す。

4539

4540 [S] は静的APIを示す。

4541

4542 [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
4543 対応でないカーネルでは、このAPIはサポートされない。

4544

4545 [p] は保護機能対応でないカーネルのみでサポートされているAPIを示す。保
4546 護機能対応カーネルでは、このAPIはサポートされない。

4547

4548 [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。
4549 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。

4550

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

また、エラーコードが返る条件を表すために、次の記号を用いる。

[s] はサービスコールのみで返るエラーコードを示す。静的APIでは、このエラーコードは返らない。

[S] は静的APIのみで返るエラーコードを示す。サービスコールでは、このエラーコードは返らない。

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

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

【 μ ITRON4.0仕様との関係】

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

INT \rightarrow int_t

UINT \rightarrow uint_t

VP \rightarrow void *

VP_INT \rightarrow intptr_t

【 μ ITRON4.0/PX仕様との関係】

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

4.1 タスク管理機能

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

タスク管理機能に関連して、各タスクが持つ情報は次の通り。

- ・タスク属性
- ・タスク状態
- ・ベース優先度
- ・現在優先度
- ・起動要求キューイング数
- ・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

4601	・メインルーチンの先頭番地
4602	・起動時優先度
4603	・実行時優先度 (TOPPERS/SSPカーネルの場合)
4604	・スタック領域
4605	・システムスタック領域 (保護機能対応カーネルの場合)
4606	・アクセス許可ベクタ (保護機能対応カーネルの場合)
4607	・属する保護ドメイン (保護機能対応カーネルの場合)
4608	・属するクラス (マルチプロセッサ対応カーネルの場合)
4609	
4610	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
4611	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。
4612	
4613	タスクの現在優先度は、タスクの実行順位を決定するために使われる優先度で
4614	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
4615	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
4616	先度に一致する。ミューテックスをロックしている間のタスクの現在優先度につ
4617	いては、「4.4.6 ミューテックス」の節を参照すること。
4618	
4619	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
4620	であり、タスクの生成時に0に初期化される。
4621	
4622	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
4623	するプロセッサで、タスクの生成時に、タスクが属するクラスによって定まる
4624	初期割付けプロセッサに初期化される。
4625	
4626	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、
4627	タスクが次に起動される時に割り付けられるプロセッサで、タスクの生成時に
4628	未設定の状態に初期化される。タスクの起動時に、次回起動時の割付けプロセッ
4629	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
4630	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。次回起動時の
4631	割付けプロセッサが未設定の場合には、タスクの割付けプロセッサは変更され
4632	ない (つまり、タスクが前に実行されていたのと同じプロセッサで実行され
4633	る)。
4634	
4635	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
4636	ステムタスクで異なる。ユーザタスクのスタック領域は、ユーザタスクが非特
4637	権モードで実行する間に用いるスタック領域であり、ユーザスタック領域と呼
4638	ぶ。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
4639	を参照すること。システムタスクのスタック領域は、カーネルの用いるオブジェ
4640	クト管理領域と同様に扱われる。
4641	
4642	システムスタック領域は、保護機能対応カーネルにおいて、ユーザタスクがサー
4643	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
4644	間に用いるスタック領域である。システムスタック領域は、カーネルの用いる
4645	オブジェクト管理領域と同様に扱われる。
4646	
4647	タスク属性には、次の属性を指定することができる。
4648	
4649	TA_ACT 0x02U タスクの生成時にタスクを起動する
4650	TA_RSTR 0x04U 生成するタスクを制約タスクとする

4651
4652 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。
4653 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
4654 がある。ターゲット定義のタスク属性として、次の属性を予約している。

4655
4656 TA_FPU FPUレジスタをコンテキストに含める

4657
4658 C言語によるタスクの記述形式は次の通り。

```
4659 void task(intptr_t exinf)
4660 {
4661     タスク本体
4662     ext_tsk();
4663 }
4664
```

4665
4666 exinfには、タスクの拡張情報が渡される。ext_tskを呼び出さず、タスクのメ
4667 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
4668 る。

4669
4670 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。

4671
4672 TMAX_ACTCNT タスクの起動要求キューイング数の最大値

4673
4674 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
4675 は、静的APIによって登録されたタスクの数に一致）

4676
4677 **【TOPPERS/ASPカーネルにおける規定】**

4678
4679 ASPカーネルでは、TMAX_ACTCNTは1に固定されている。また、制約タスクはサポー
4680 トしていない。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
4681 機能を追加することができる。

4682
4683 **【TOPPERS/FMPカーネルにおける規定】**

4684
4685 FMPカーネルでは、TMAX_ACTCNTは1に固定されている。また、制約タスクはサポー
4686 トしていない。

4687
4688 **【TOPPERS/HRP2カーネルにおける規定】**

4689
4690 HRP2カーネルでは、TMAX_ACTCNTは1に固定されている。また、制約タスクはサ
4691 ポートしていない。

4692
4693 **【TOPPERS/SSPカーネルにおける規定】**

4694
4695 SSPカーネルでは、タスクの起動要求のキューイングはサポートしておらず、タ
4696 スクに対して起動要求キューイング数の情報を持たない。また、TMAX_ACTCNTを
4697 定義していない。

4698
4699 また、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
4700 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。

SSPカーネルにおける追加機能として、タスクに対して、実行時優先度の情報を持つ。SSPカーネルにおいては、タスクが起動された後、最初に実行状態になる時に、タスクのベース優先度が、タスクの実行時優先度に設定される。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。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。

【実行時優先度によるスタック領域の節約】

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

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

【 μ ITRON4.0仕様との関係】

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

TNUM_TSKIDは、 μ ITRON4.0仕様に規定されていないカーネル構成マクロである。

CRE_TSK タスクの生成 [S]
acre_tsk タスクの生成 [TD]

【静的API】

*保護機能対応でないカーネルの場合

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

*保護機能対応カーネルの場合

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

※ sstkszおよびssstkの記述は省略することができる。

【C言語API】

```
ER ID tskid = acre_tsk(const T CTSK *pk ctsk)
```

4751

4752 **【パラメータ】**

4753 ID tskid 生成するタスクのID番号 (CRE_TSKの場合)

4754 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
4755 タ (静的APIを除く)

4756

4757 *タスクの生成情報 (パケットの内容)

4758 ATR tskatr タスク属性

4759 intptr_t exinf タスクの拡張情報

4760 TASK task タスクのメインルーチンの先頭番地

4761 PRI itskpri タスクの起動時優先度

4762 SIZE stksz タスクのスタック領域のサイズ (バイト数)

4763 STK_T * stk タスクのスタック領域の先頭番地

4764 SIZE sstksz タスクのシステムスタック領域のサイズ (バイ
4765 ト数, 保護機能対応カーネルの場合, 静的API
4766 においては省略可)

4767 STK_T * sstk タスクのシステムスタック領域の先頭番地 (保
4768 護機能対応カーネルの場合, 静的APIにおいて
4769 は省略可)

4770

4771 **【リターンパラメータ】**

4772 ER_ID tskid 生成されたタスクのID番号 (正の値) またはエ
4773 ラーコード

4774

4775 **【エラーコード】**

4776 E_CTX [s] コンテキストエラー (非タスクコンテキストからの呼出
4777 し, CPUロック状態からの呼出し)

4778 E_RSATR 予約属性 (tskatrが不正または使用できない, 属する保
4779 護ドメインがクラスが不正)

4780 E_PAR パラメータエラー (task, itskpri, stksz, stk, sstksz,
4781 sstkが不正, その他の条件については機能の項を参照す
4782 ること)

4783 E_OACV [sP] オブジェクトアクセス違反 (システム状態に対する管理
4784 操作が許可されていない)

4785 E_MACV [sP] メモリアクセス違反 (pk_ctskが指すメモリ領域への読出
4786 しアクセスが許可されていない)

4787 E_NOID [sD] ID番号不足 (割り付けられるタスクIDがない)

4788 E_NOMEM メモリ不足 (スタック領域やシステムスタック領域が確
4789 保できない)

4790 E_OBJ オブジェクト状態エラー (tskidで指定したタスクが登録
4791 済み: CRE_TSKの場合, その他の条件については機能の項
4792 を参照すること)

4793

4794 **【機能】**

4795

4796 各パラメータで指定したタスク生成情報に従って, タスクを生成する. 具体的
4797 な振舞いは以下の通り.

4798

4799 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. stkszに0を
4800 指定した時や, ターゲット定義の最小値よりも小さい値を指定した時には,

4801 E_PARエラーとなる。また、保護機能対応カーネルで、生成するタスクがユーザ
4802 タスクの場合には、sstkとsstkszsからシステムスタック領域が設定される。こ
4803 の場合、sstkszsに0を指定した時や、ターゲット定義の最小値よりも小さい値を
4804 指定した時には、E_PARエラーとなる。

4805

4806 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、
4807 生成されたタスクは休止状態になる。さらに、tskatrにTA_ACTを指定した場合
4808 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
4809 できる状態になる。

4810

4811 静的APIにおいては、tskidはオブジェクト識別名、tskatr、itskpri、stkszsは
4812 整数定数式パラメータ、exinf、task、stkは一般定数式パラメータである。コ
4813 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
4814 ができない。

4815

4816 itskpriは、TMIN_TPRI以上、TMAX_TPRI以下でなければならない。

4817

4818 [stkにNULLを指定した場合]

4819

4820 stkをNULLとした場合、stkszsで指定したサイズのスタック領域が、コンフィギュ
4821 レータまたはカーネルにより確保される。stkszsにターゲット定義の制約に合致
4822 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
4823 い方に丸めたサイズで確保される。

4824

4825 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
4826 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（ユーザスタ
4827 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
4828 て、メモリオブジェクトとしてカーネルに登録される。

4829

4830 静的APIにおいて、生成するタスクが制約タスクの場合（tskatrにTA_RSTRを指
4831 定した場合）、コンフィギュレータは、生成する制約タスクの起動時優先度毎
4832 にスタック領域を確保し、同じ起動時優先度を持つ制約タスクにそのスタック
4833 領域を共有させる。確保するスタック領域のサイズは、コンフィギュレータが
4834 スタック領域を確保し（stkにNULLを指定して生成され）、同じ起動時優先度
4835 を持つ制約タスクのスタック領域のサイズ（stkszs）の最大値となる。マルチプロ
4836 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
4837 期割付けプロセッサ毎に行う。

4838

4839 [stkにNULL以外を指定した場合]

4840

4841 stkにNULL以外を指定した場合、stkとstkszsで指定したスタック領域は、アプリ
4842 ケーションで確保しておく必要がある。スタック領域をアプリケーションで確
4843 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。
4844 その方法に従わず、stkやstkszsにターゲット定義の制約に合致しない先頭番地
4845 やサイズを指定した時には、E_PARエラーとなる。

4846

4847 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、
4848 stkとstkszs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
4849 まれない場合、E_OBJエラーとなる。

4850

4851 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
4852 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
4853 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
4854 カーネルに登録される。そのため、上の方法を用いてスタック領域を確保して
4855 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
4856 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。ま
4857 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
4858 メモリ領域が重なる場合には、E_OBJエラーとなる。

4859
4860 [sstkとsstkszの扱い]

4861
4862 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
4863 ザタスクの場合とシステムタスクの場合で異なる。

4864
4865 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。
4866

4867 sstkの記述を省略するか、sstkをNULLとした場合、sstkszで指定したサイズの
4868 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
4869 る。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
4870 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。
4871 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
4872 保される。

4873
4874 sstkにNULL以外を指定した場合、sstkとsstkszで指定したスタック領域は、ア
4875 プリケーションで確保しておく必要がある。スタック領域をアプリケーション
4876 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。
4877 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
4878 頭番地やサイズを指定した時には、E_PARエラーとなる。また、stkとstkszで指
4879 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
4880 い場合、E_OBJエラーとなる。

4881
4882 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。

4883
4884 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
4885 た場合には、E_PARエラーとなる。

4886
4887 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
4888 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
4889 れる。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
4890 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
4891 ズで確保される。

4892
4893 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
4894 となる。

4895
4896 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。

4897
4898 **【TOPPERS/ASPカーネルにおける規定】**

4899
4900 ASPカーネルでは、CRE_TSKのみをサポートする。ただし、動的生成機能拡張パッ

4901 ケージでは、`acre_tsk`もサポートする。
4902
4903 **【TOPPERS/FMPカーネルにおける規定】**
4904
4905 FMPカーネルでは、`CRE_TSK`のみをサポートする。
4906
4907 **【TOPPERS/HRP2カーネルにおける規定】**
4908
4909 HRP2カーネルでは、`CRE_TSK`のみをサポートする。
4910
4911 **【TOPPERS/SSPカーネルにおける規定】**
4912
4913 SSPカーネルでは、`CRE_TSK`のみをサポートする。
4914
4915 SSPカーネルでは、複数のタスクに対して、同じ起動時優先度を設定することは
4916 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。
4917
4918 SSPカーネルでは、制約タスクのみをサポートするため、タスク属性に`TA_RSTR`
4919 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。
4920
4921 SSPカーネルでは、`stk`には`NULL`を指定しなくてはならず、その場合でも、コン
4922 フィギュレータはタスクのスタック領域を確保しない。これは、SSPカーネルで
4923 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
4924 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
4925 る。
4926
4927 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。
4928
4929 **【 μ ITRON4.0仕様との関係】**
4930
4931 `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
4932 `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
4933 した。
4934
4935 **【 μ ITRON4.0/PX仕様との関係】**
4936
4937 `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
4938 で確保する方法を規定した。
4939
4940 **【未決定事項】**
4941
4942 サービスコール (`acre_tsk`) により、`stk`に`NULL`を指定して制約タスクを生成し
4943 た場合のスタック領域の確保方法については、今後の課題である。
4944
4945 **【仕様決定の理由】**
4946
4947 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
4948 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
4949 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
4950 るためである。

```

4951 -----
4952 AID_TSK      割付け可能なタスクIDの数の指定 [SD]
4953
4954 【静的API】
4955     AID_TSK(uint_t notsk)
4956
4957 【パラメータ】
4958     uint_t      notsk      割付け可能なタスクIDの数
4959
4960 【エラーコード】
4961     E_RSATR      予約属性（属する保護ドメインまたはクラスが不正）
4962
4963 【機能】
4964
4965 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
4966 付け可能なタスクIDとして確保する。
4967
4968 notskは整数定数式パラメータである。
4969 -----
4970 SAC_TSK      タスクのアクセス許可ベクタの設定 [SP]
4971 sac_tsk      タスクのアクセス許可ベクタの設定 [TPD]
4972
4973 【静的API】
4974     SAC_TSK(ID tskid, { ACPTN acptn1, ACPTN acptn2,
4975                          ACPTN acptn3, ACPTN acptn4 })
4976
4977 【C言語API】
4978     ER ercd = sac_tsk(ID tskid, const ACVCT *p_acvct)
4979
4980 【パラメータ】
4981     ID          tskid      対象タスクのID番号
4982     ACVCT *      p_acvct   アクセス許可ベクタを入れたパケットへのポ
4983                             インタ（静的APIを除く）
4984
4985     *アクセス許可ベクタ（パケットの内容）
4986     ACPTN        acptn1    通常操作1のアクセス許可パターン
4987     ACPTN        acptn2    通常操作2のアクセス許可パターン
4988     ACPTN        acptn3    管理操作のアクセス許可パターン
4989     ACPTN        acptn4    参照操作のアクセス許可パターン
4990
4991 【リターンパラメータ】
4992     ER          ercd      正常終了（E_OK）またはエラーコード
4993
4994 【エラーコード】
4995     E_CTX [s]      コンテキストエラー（非タスクコンテキストからの呼出
4996                     し、CPUロック状態からの呼出し）
4997     E_ID           不正ID番号（tskidが不正）
4998     E_RSATR        予約属性（属する保護ドメインかクラスが不正：SAC_TSK
4999                     の場合）
5000     E_NOEXS [D]    オブジェクト未登録（対象タスクが未登録）

```

5001 E_OACV [sP] オブジェクトアクセス違反（対象タスクに対する管理操
5002 作が許可されていない）
5003 E_MACV [sP] メモリアクセス違反（p_acvctが指すメモリ領域への読出
5004 しアクセスが許可されていない）
5005 E_OBJ オブジェクト状態エラー（対象タスクは静的APIで生成さ
5006 れた：sac_tskの場合、対象タスクに対してアクセス許可
5007 ベクタが設定済み：SAC_TSKの場合）
5008

5009 【機能】

5010

5011 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
5012 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。

5013

5014 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
5015 式パラメータである。

5016

5017 SAC_TSKは、対象タスクが属する保護ドメインの囲みの中に記述しなければなら
5018 ない。そうでない場合には、E_RSATRエラーとなる。

5019

5020 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
5021 となる。

5022

5023 【TOPPERS/ASPカーネルにおける規定】

5024

5025 ASPカーネルでは、SAC_TSK、sac_tskをサポートしない。

5026

5027 【TOPPERS/FMPカーネルにおける規定】

5028

5029 FMPカーネルでは、SAC_TSK、sac_tskをサポートしない。

5030

5031 【TOPPERS/HRP2カーネルにおける規定】

5032

5033 HRP2カーネルでは、SAC_TSKのみをサポートする。

5034

5035 【TOPPERS/SSPカーネルにおける規定】

5036

5037 SSPカーネルでは、SAC_TSK、sac_tskをサポートしない。

5038

5039 DEF_EPR タスクの実行時優先度の定義 [S]

5040

5041 【静的API】

5042 DEF_EPR(ID tskid, { PRI exepr })

5043

5044 【パラメータ】

5045 ID tskid 対象タスクのID番号

5046 PRI exepr タスクの実行時優先度

5047

5048 【エラーコード】

5049 E_ID 不正ID番号（tskidが不正）

5050 E_PAR パラメータエラー（exeprが不正）

5051 E_ILUSE サービスコール不正使用 (exepriが、自タスクの起動時
5052 優先度よりも低い場合)
5053 E_OBJ オブジェクト状態エラー (対象タスクに対して実行優先
5054 度が設定済み)
5055

5056 **【サポートするカーネル】**

5057
5058 DEF_EPRは、TOPPERS/SSPカーネルのみがサポートする静的APIである。他のカー
5059 ネルは、DEF_EPRをサポートしない。
5060

5061 **【機能】**

5062
5063 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
5064 先度に設定する。
5065

5066 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。
5067

5068 exepriは、TMIN_TPRI以上、TMAX_TPRI以下でなければならない。また、exepri
5069 は、対象タスクの起動時優先度と同じかそれよりも高くなければならない。
5070

5071 **【 μ ITRON4.0仕様との関係】**

5072
5073 μ ITRON4.0仕様に定義されていない静的APIである。
5074

5075 del_tsk タスクの削除 [TD]

5076
5077 **【C言語API】**

5078 ER ercd = del_tsk(ID tskid)
5079

5080 **【パラメータ】**

5081 ID tskid 対象タスクのID番号
5082

5083 **【リターンパラメータ】**

5084 ER ercd 正常終了 (E_OK) またはエラーコード
5085

5086 **【エラーコード】**

5087 E_CTX コンテキストエラー (非タスクコンテキストからの呼出
5088 し、CPUロック状態からの呼出し)

5089 E_ID 不正ID番号 (tskidが不正)

5090 E_NOEXS [D] オブジェクト未登録 (対象タスクが未登録)

5091 E_OACV [P] オブジェクトアクセス違反 (対象タスクに対する管理操
5092 作が許可されていない)

5093 E_OBJ オブジェクト状態エラー (対象タスクが休止状態でない、
5094 対象タスクは静的APIで生成された)
5095

5096 **【機能】**

5097
5098 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
5099 り。
5100

5101 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
5102 スクIDが未使用の状態に戻される。また、タスクの生成時にタスクのスタック
5103 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
5104 らのメモリ領域が解放される。

5105
5106 対象タスクが休止状態でない場合には、E_OBJエラーとなる。

5107
5108 **【TOPPERS/ASPカーネルにおける規定】**

5109 ASPカーネルでは、del_tskをサポートしない。ただし、動的生成機能拡張パッ
5110 ケージでは、del_tskをサポートする。

5111
5112 **【TOPPERS/FMPカーネルにおける規定】**

5113 FMPカーネルでは、del_tskをサポートしない。

5114
5115 **【TOPPERS/HRP2カーネルにおける規定】**

5116 HRP2カーネルでは、del_tskをサポートしない。

5117
5118 **【TOPPERS/SSPカーネルにおける規定】**

5119 SSPカーネルでは、del_tskをサポートしない。

5120
5121 -----
5122 act_tsk タスクの起動 [T]
5123 iact_tsk タスクの起動 [I]

5124
5125 **【C言語API】**
5126 ER ercd = act_tsk(ID tskid)
5127 ER ercd = iact_tsk(ID tskid)

5128
5129 **【パラメータ】**
5130 ID tskid 対象タスクのID番号

5131
5132 **【リターンパラメータ】**
5133 ER ercd 正常終了 (E_OK) またはエラーコード

5134
5135 **【エラーコード】**
5136 E_CTX コンテキストエラー（非タスクコンテキストからの呼出
5137 し：act_tskの場合、タスクコンテキストからの呼出し：
5138 iact_tskの場合、CPUロック状態からの呼出し）
5139 E_ID 不正ID番号（tskidが不正）
5140 E_NOEXS [D] オブジェクト未登録（対象タスクが未登録）
5141 E_OACV [P] オブジェクトアクセス違反（対象タスクに対する通常操
5142 作1が許可されていない：act_tskの場合）
5143 E_QOVR キューイングオーバーフロー（起動要求キューイング数が
5144 TMAX_ACTCNTに一致）

5145
5146 **【機能】**

5147
5148
5149
5150

5151 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
5152 いは以下の通り．

5153

5154 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
5155 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．

5156

5157 対象タスクが休止状態でない場合には，対象タスクの起動要求キューイング数
5158 に1が加えられる．起動要求キューイング数に1を加えるとTMAX_ACTCNTを超える
5159 場合には，E_QOVRエラーとなる．

5160

5161 act_tskにおいてtskidにTSK_SELF（=0）を指定すると，自タスクが対象タスク
5162 となる．

5163

5164 【補足説明】

5165

5166 マルチプロセッサ対応カーネルでは，act_tsk/iact_tskは，対象タスクの次回
5167 起動時の割付けプロセッサを変更しない．

5168

5169 mact_tsk 割付けプロセッサ指定でのタスクの起動 [TM]

5170 imact_tsk 割付けプロセッサ指定でのタスクの起動 [IM]

5171

5172 【C言語API】

5173 ER ercd = mact_tsk(ID tskid, ID prcid)

5174 ER ercd = imact_tsk(ID tskid, ID prcid)

5175

5176 【パラメータ】

5177 ID tskid 対象タスクのID番号

5178 ID prcid タスクの割付け対象のプロセッサのID番号

5179

5180 【リターンパラメータ】

5181 ER ercd 正常終了 (E_OK) またはエラーコード

5182

5183 【エラーコード】

5184 E_CTX コンテキストエラー（非タスクコンテキストからの呼出
5185 し：mact_tskの場合，タスクコンテキストからの呼出し
5186 ：imact_tskの場合，CPUロック状態からの呼出し）

5187 E_NOSPT 未サポート機能（対象タスクが制約タスク）

5188 E_ID 不正ID番号（tskid, prcidが不正）

5189 E_PAR パラメータエラー（対象タスクはprcidで指定したプロ
5190 セッサに割り付けられない）

5191 E_NOEXS [D] オブジェクト未登録（対象タスクが未登録）

5192 E_OACV [P] オブジェクトアクセス違反（対象タスクに対する通常操
5193 作1が許可されていない：mact_tskの場合）

5194 E_QOVR キューイングオーバフロー（起動要求キューイング数が
5195 TMAX_ACTCNTに一致）

5196

5197 【機能】

5198

5199 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
5200 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．

5201
5202 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
5203 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
5204 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。
5205
5206 対象タスクが休止状態でない場合には、対象タスクの起動要求キューイング数
5207 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
5208 に変更される。起動要求キューイング数に1を加えるとTMAX_ACTCNTを超える場
5209 合には、E_QOVRエラーとなる。
5210
5211 mact_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
5212 クとなる。
5213
5214 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
5215 サを含んでいない場合には、E_PARエラーとなる。
5216
5217 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
5218 れが属するクラスの初期割付けプロセッサとする。
5219
5220 **【補足説明】**
5221
5222 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
5223 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
5224 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
5225 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
5226 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
5227 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
5228 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
5229 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
5230 変更されない。
5231
5232 **【TOPPERS/ASPカーネルにおける規定】**
5233
5234 ASPカーネルでは、mact_tsk、imact_tskをサポートしない。
5235
5236 **【TOPPERS/HRP2カーネルにおける規定】**
5237
5238 HRP2カーネルでは、mact_tsk、imact_tskをサポートしない。
5239
5240 **【TOPPERS/SSPカーネルにおける規定】**
5241
5242 SSPカーネルでは、mact_tsk、imact_tskをサポートしない。
5243
5244 **【μITRON4.0仕様との関係】**
5245
5246 μITRON4.0仕様に定義されていないサービスコールである。
5247 -----
5248 can_act タスク起動要求のキャンセル [T]
5249
5250 **【C言語API】**

5251 ER_UINT actcnt = can_act(ID tskid)

5252

5253 **【パラメータ】**

5254 ID tskid 対象タスクのID番号

5255

5256 **【リターンパラメータ】**

5257 ER_UINT actcnt キューイングされていた起動要求の数（正の値
5258 または0）またはエラーコード

5259

5260 **【エラーコード】**

5261 E_CTX コンテキストエラー（非タスクコンテキストからの呼出し、CPUロック状態からの呼出し）

5262

5263 E_ID 不正ID番号（tskidが不正）

5264 E_NOEXS [D] オブジェクト未登録（対象タスクが未登録）

5265 E_OACV [P] オブジェクトアクセス違反（対象タスクに対する通常操
5266 作1が許可されていない）

5267

5268 **【機能】**

5269

5270 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
5271 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
5272 下の通り。

5273

5274 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
5275 求キューイング数が、サービスコールの返回值として返される。また、マルチプ
5276 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
5277 サが未設定状態に戻される。

5278

5279 tskidにTSK_SELF（=0）を指定すると、自タスクが対象タスクとなる。

5280

5281 **【TOPPERS/SSPカーネルにおける規定】**

5282

5283 SSPカーネルでは、can_actをサポートしない。

5284

5285 mig_tsk タスクの割付けプロセッサの変更 [TM]

5286

5287 **【C言語API】**

5288 ER ercd = mig_tsk(ID tskid, ID prcid)

5289

5290 **【パラメータ】**

5291 ID tskid 対象タスクのID番号

5292 ID prcid タスクの割付けプロセッサのID番号

5293

5294 **【リターンパラメータ】**

5295 ER ercd 正常終了（E_OK）またはエラーコード

5296

5297 **【エラーコード】**

5298 E_CTX コンテキストエラー（非タスクコンテキストからの呼出し、CPUロック状態からの呼出し、対象タスクが自タスク
5299 でディスパッチ保留状態からの呼出し）

5300

5301	E_NOSPT	未サポート機能（対象タスクが制約タスク）
5302	E_ID	不正ID番号（tskid, prcidが不正）
5303	E_PAR	パラメータエラー（対象タスクはprcidで指定したプロセッサに割り付けられない）
5304		
5305	E_NOEXS [D]	オブジェクト未登録（対象タスクが未登録）
5306	E_OACV [P]	オブジェクトアクセス違反（対象タスクに対する通常操作1が許可されていない）
5307		
5308	E_OBJ	オブジェクト状態エラー（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）
5309		
5310		

【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。対象タスクが実行できる状態の場合には、prcidで指定したプロセッサに割り付けられた同じ優先度のタスクの中で、最も優先順位が低い状態となる。

対象タスクが、自タスクが割り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。

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

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。

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。

【補足説明】

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

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

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、mig_tskをサポートしない。

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、mig_tskをサポートしない。

5351

5352 【TOPPERS/SSPカーネルにおける規定】

5353

5354 SSPカーネルでは、mig_tskをサポートしない。

5355

5356 【μ ITRON4.0仕様との関係】

5357

5358 μ ITRON4.0仕様に定義されていないサービスコールである。

5359

5360 ext_tsk 自タスクの終了 [T]

5361

5362 【C言語API】

5363 ER ercd = ext_tsk()

5364

5365 【パラメータ】

5366 なし

5367

5368 【リターンパラメータ】

5369 ER ercd エラーコード

5370

5371 【エラーコード】

5372 E_SYS システムエラー（カーネルの誤動作）

5373 E_CTX コンテキストエラー（非タスクコンテキストからの呼出し）

5374

5375 【機能】

5376

5377 自タスクを終了させる。具体的な振舞いは以下の通り。

5378

5379 自タスクに対してタスク終了時に行うべき処理が行われ、自タスクは休止状態になる。さらに、自タスクの起動要求キューイング数が0でない場合には、自タスクに対してタスク起動時に行うべき処理が行われ、自タスクは実行できる状態になる。またこの時、起動要求キューイング数から1が減ぜられる。

5380

5381 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。

5382

5383 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。

5384

5385 【TOPPERS/SSPカーネルにおける規定】

5386

5387 SSPカーネルでは、ext_tskをサポートしない。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。

5388

5389 【μ ITRON4.0仕様との関係】

5390

5391 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。μ ITRON4.0仕様においては、ext_tskからはリターンしないと規定さ

5392

5401 れている.

5402 -----

5403 `ter_tsk` タスクの強制終了 [T]

5404

5405 **【C言語API】**

5406 `ER ercd = ter_tsk(ID tskid)`

5407

5408 **【パラメータ】**

5409 `ID` `tskid` 対象タスクのID番号

5410

5411 **【リターンパラメータ】**

5412 `ER` `ercd` 正常終了 (E_OK) またはエラーコード

5413

5414 **【エラーコード】**

5415 `E_CTX` コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)

5416 `E_ID` 不正ID番号 (tskidが不正)

5417 `E_NOEXS [D]` オブジェクト未登録 (対象タスクが未登録)

5418 `E_OACV [P]` オブジェクトアクセス違反 (対象タスクに対する通常操作2が許可されていない)

5419 `E_ILUSE` サービスコール不正使用 (対象タスクが自タスク)

5420 `E_OBJ` オブジェクト状態エラー (対象タスクが休止状態, 対象タスクが自タスクと異なるプロセッサに割り付けられている)

5421

5422 **【機能】**

5423

5424 `tskid`で指定したタスク (対象タスク) を終了させる. 具体的な振舞いは以下の通り.

5425

5426 対象タスクが休止状態でない場合には, 対象タスクに対してタスク終了時に行うべき処理が行われ, 対象タスクは休止状態になる. さらに, 対象タスクの起動要求キューイング数が0でない場合には, 対象タスクに対してタスク起動時に行うべき処理が行われ, 対象タスクは実行できる状態になる. またこの時, 起動要求キューイング数から1が減ぜられる.

5427

5428 対象タスクが休止状態である場合には, `E_OBJ`エラーとなる. また, 対象タスクが自タスクの場合には, `E_ILUSE`エラーとなる.

5429

5430 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.

5431

5432 **【TOPPERS/FMPカーネルにおける使用上の注意】**

5433

5434 現時点のFMPカーネルの実装では, デッドロック回避のためのリトライ処理により, サービスコールの処理時間に上限がないため, 注意が必要である (ロック方式にも依存する).

5435

5436 **【TOPPERS/SSPカーネルにおける規定】**

5437

5451
 5452 SSPカーネルでは、ter_tskをサポートしない。
 5453 -----
 5454 chg_pri タスクのベース優先度の変更 [T]
 5455
 5456 【C言語API】
 5457 ER ercd = chg_pri(ID tskid, PRI tskpri)
 5458
 5459 【パラメータ】
 5460 ID tskid 対象タスクのID番号
 5461 PRI tskpri ベース優先度
 5462
 5463 【リターンパラメータ】
 5464 ER ercd 正常終了 (E_OK) またはエラーコード
 5465
 5466 【エラーコード】
 5467 E_CTX コンテキストエラー (非タスクコンテキストからの呼出
 5468 し、CPUロック状態からの呼出し)
 5469 E_NOSPT 未サポート機能 (対象タスクが制約タスク)
 5470 E_ID 不正ID番号 (tskidが不正)
 5471 E_PAR パラメータエラー (tskpriが不正)
 5472 E_NOEXS [D] オブジェクト未登録 (対象タスクが未登録)
 5473 E_OACV [P] オブジェクトアクセス違反 (対象タスクに対する通常操
 5474 作2が許可されていない)
 5475 E_ILUSE サービスコール不正使用 (tskpriが、対象タスクがロッ
 5476 クしているかロックを待っている優先度上限ミューテッ
 5477 クスの上限優先度よりも高い場合)
 5478 E_OBJ オブジェクト状態エラー (対象タスクが休止状態)
 5479
 5480 【機能】
 5481
 5482 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
 5483 先度に変更する。具体的な振舞いは以下の通り。
 5484
 5485 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
 5486 で指定した優先度に変更される。それに伴って、対象タスクの現在優先度も変
 5487 更される。
 5488
 5489 対象タスクが、優先度上限ミューテックスをロックしていない場合には、次の
 5490 処理が行われる。対象タスクが実行できる状態の場合には、同じ優先度のタス
 5491 クの中で最低優先順位となる。対象タスクが待ち状態で、タスクの優先度順の
 5492 待ち行列につながれている場合には、対象タスクの変更後の現在優先度に従っ
 5493 て、その待ち行列中での順序が変更される。待ち行列中に同じ現在優先度のタ
 5494 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。
 5495
 5496 対象タスクが、優先度上限ミューテックスをロックしている場合には、対象タ
 5497 スクの現在優先度の変更されることはなく、優先順位も変更されない。
 5498
 5499 対象タスクが休止状態である場合には、E_OBJエラーとなる。
 5500

5501 tskidにTSK_SELF (=0) を指定すると、自タスクが対象タスクとなる。また、
 5502 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
 5503 優先度に変更される。

5504
 5505 tskpriは、TPRI_INIであるか、TMIN_TPRI以上、TMAX_TPRI以下でなければなら
 5506 ない。また、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
 5507 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
 5508 かそれより低くなければならない。

5509
 5510 **【TOPPERS/SSPカーネルにおける規定】**

5511
 5512 SSPカーネルでは、chg_priをサポートしない。

5513
 5514 **【 μ ITRON4.0仕様との関係】**

5515
 5516 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
 5517 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
 5518 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。

5519 -----
 5520 get_pri タスク優先度の参照 [T]

5521
 5522 **【C言語API】**

5523 ER ercd = get_pri(ID tskid, PRI *p_tskpri)

5524
 5525 **【パラメータ】**

5526 ID tskid 対象タスクのID番号
 5527 PRI * p_tskpri 現在優先度を入れるメモリ領域へのポインタ

5528
 5529 **【リターンパラメータ】**

5530 ER ercd 正常終了 (E_OK) またはエラーコード
 5531 PRI tskpri 現在優先度

5532
 5533 **【エラーコード】**

5534 E_CTX コンテキストエラー（非タスクコンテキストからの呼出
 5535 し、CPUロック状態からの呼出し）
 5536 E_ID 不正ID番号（tskidが不正）
 5537 E_NOEXS [D] オブジェクト未登録（対象タスクが未登録）
 5538 E_OACV [P] オブジェクトアクセス違反（対象タスクに対する参照操
 5539 作が許可されていない）
 5540 E_MACV [P] メモリアクセス違反（p_tskpriが指すメモリ領域への書
 5541 込みアクセスが許可されていない）
 5542 E_OBJ オブジェクト状態エラー（対象タスクが休止状態）

5543
 5544 **【機能】**

5545
 5546 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
 5547 いは以下の通り。

5548
 5549 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
 5550 で指定したメモリ領域に返される。対象タスクが休止状態である場合には、

5551 E_OBJエラーとなる.

5552

5553 tskidにTSK_SELF (=0) を指定すると, 自タスクが対象タスクとなる.

5554

5555 **【TOPPERS/SSPカーネルにおける規定】**

5556

5557 SSPカーネルでは, get_priをサポートしない.

5558

5559 get_inf 自タスクの拡張情報の参照 [T]

5560

5561 **【C言語API】**

5562 ER ercd = get_inf(intptr_t *p_exinf)

5563

5564 **【パラメータ】**

5565 intptr_t * p_exinf 拡張情報を入れるメモリ領域へのポインタ

5566

5567 **【リターンパラメータ】**

5568 ER ercd 正常終了 (E_OK) またはエラーコード

5569 intptr_t exinf 拡張情報

5570

5571 **【エラーコード】**

5572 E_CTX コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)

5573 E_MACV [P] メモリアクセス違反 (p_exinfが指すメモリ領域への書き込みアクセスが許可されていない)

5574

5575

5576 **【機能】**

5577

5578 自タスクの拡張情報を参照する. 参照した拡張情報は, p_exinfで指定したメモリ領域に返される.

5579

5580

5581 **【TOPPERS/SSPカーネルにおける規定】**

5582

5583 SSPカーネルでは, get_infをサポートしない.

5584

5585 **【μITRON4.0仕様との関係】**

5586

5587 μITRON4.0仕様に定義されていないサービスコールである.

5588

5589 ref_tsk タスクの状態参照 [T]

5590

5591 **【C言語API】**

5592 ER ercd = ref_tsk(ID tskid, T_RTsk *pk_rtsk)

5593

5594 **【パラメータ】**

5595 ID tskid 対象タスクのID番号

5596 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

5597

5598 **【リターンパラメータ】**

5599 ER ercd 正常終了 (E_OK) またはエラーコード

5600

5601

5602 *タスクの現在状態（パケットの内容）

5603	STAT	tskstat	タスク状態
5604	PRI	tskpri	タスクの現在優先度
5605	PRI	tskbpri	タスクのベース優先度
5606	STAT	tskwait	タスクの待ち要因
5607	ID	wobjid	タスクの待ち対象のオブジェクトのID
5608	TMO	lefttmo	タスクがタイムアウトするまでの時間
5609	uint_t	actcnt	タスクの起動要求キューイング数
5610	uint_t	wupcnt	タスクの起床要求キューイング数
5611	bool_t	texmsk	タスクがタスク例外処理マスク状態か否か（保護機能対応カーネルの場合）
5612			
5613	bool_t	waifbd	タスクが待ち禁止状態か否か（保護機能対応カーネルの場合）
5614			
5615	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）
5616			
5617	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）
5618			
5619	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）
5620			
5621			
5622	【エラーコード】		
5623	E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し、CPUロック状態からの呼出し）	
5624			
5625	E_ID	不正ID番号（tskidが不正）	
5626	E_NOEXS [D]	オブジェクト未登録（対象タスクが未登録）	
5627	E_OACV [P]	オブジェクトアクセス違反（対象タスクに対する参照操作が許可されていない）	
5628			
5629	E_MACV [P]	メモリアクセス違反（pk_rtskが指すメモリ領域への書き込みアクセスが許可されていない）	
5630			
5631			
5632	【機能】		
5633			
5634	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．		
5635			
5636			
5637	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．		
5638			
5639			
5640	TTS_RUN	0x01U	実行状態
5641	TTS_RDY	0x02U	実行可能状態
5642	TTS_WAI	0x04U	待ち状態
5643	TTS_SUS	0x08U	強制待ち状態
5644	TTS_WAS	0x0cU	二重待ち状態
5645	TTS_DMT	0x10U	休止状態
5646			
5647	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、		
5648	tskstatがTTS_SUSとなる場合がある．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		
5649	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．		
5650			

5651
 5652 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、
 5653 tskbpriには対象タスクのベース優先度が返される。対象タスクが休止状態であ
 5654 る場合には、tskpriとtskbpriの値は保証されない。
 5655
 5656 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つて
 5657 いる状態であるかを表す次のいずれかの値が返される。
 5658
 5659 TTW_SLP 0x0001U 起床待ち
 5660 TTW_DLY 0x0002U 時間経過待ち
 5661 TTW_SEM 0x0004U セマフォの資源獲得待ち
 5662 TTW_FLG 0x0008U イベントフラグ待ち
 5663 TTW_SDTQ 0x0010U データキューへの送信待ち
 5664 TTW_RDTQ 0x0020U データキューからの受信待ち
 5665 TTW_SPDQ 0x0100U 優先度データキューへの送信待ち
 5666 TTW_RPDQ 0x0200U 優先度データキューからの受信待ち
 5667 TTW_MBX 0x0040U メールボックスからの受信待ち
 5668 TTW_MTX 0x0080U ミューテックスのロック待ち状態
 5669 TTW_MPF 0x2000U 固定長メモリブロックの獲得待ち
 5670
 5671 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。
 5672
 5673 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
 5674 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。
 5675 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
 5676 る場合には、wobjidの値は保証されない。
 5677
 5678 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
 5679 スクがタイムアウトを起こすまでの相対時間が返される。タスクがタイムアウト
 5680 を起こさない場合には、TMO_FEVR (=-1) が返される。
 5681
 5682 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
 5683 が経過して待ち解除されるまでの相対時間が返される。ただし、返されるべき
 5684 相対時間がTMO型に格納することができない場合がありうる。この場合には、相
 5685 対時間 (RELTIM型、uint_t型に定義される) をTMO型 (int_t型に定義される)
 5686 に型キャストした値が返される。
 5687
 5688 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。
 5689
 5690 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。
 5691
 5692 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
 5693 ング数が返される。対象タスクが休止状態である場合には、wupcntの値は保証
 5694 されない。
 5695
 5696 保護機能対応カーネルで、対象タスクが休止状態でない場合には、texmskに、
 5697 対象タスクがタスク例外処理マスク状態の場合にtrue、そうでない場合に
 5698 falseが返される。waifbdには、対象タスクが待ち禁止状態の場合にtrue、そう
 5699 でない場合にfalseが返される。またsvclevelには、対象タスクが拡張サービ
 5700 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

5701 サービスコールがネスト段数が返される。対象タスクが休止状態である場合に
5702 は、texmsk, waifbd, svclevelの値は保証されない。

5703

5704 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
5705 のID番号が返される。またactprcには、対象タスクの次回起動時の割付けプロ
5706 セッサのID番号が返される。次回起動時の割付けプロセッサが未設定の場合に
5707 は、actprcにTPRC_NONE (=0) が返される。

5708

5709 tskidにTSK_SELF (=0) を指定すると、自タスクが対象タスクとなる。

5710

5711 **【補足説明】**

5712

5713 対象タスクが時間経過待ち状態である場合に、lefttmo (TMO型) に返される値
5714 をRELTIM型に型キャストすることで、タスクが待ち解除されるまでの相対時間
5715 を正しく得ることができる。

5716

5717 **【TOPPERS/ASPカーネルにおける規定】**

5718

5719 ASPカーネルでは、tskwaitにTTW_MTXが返ることはない。ただし、ミューテック
5720 ス機能拡張パッケージを用いると、tskwaitにTTW_MTXが返る場合がある。

5721

5722 **【TOPPERS/FMPカーネルにおける規定】**

5723

5724 FMPカーネルでは、tskwaitにTTW_MTXが返ることはない。

5725

5726 **【TOPPERS/HRP2カーネルにおける規定】**

5727

5728 HRP2カーネルでは、tskwaitにTTW_MBXが返ることはない。

5729

5730 **【TOPPERS/SSPカーネルにおける規定】**

5731

5732 SSPカーネルでは、ref_tskをサポートしない。

5733

5734 **【使用上の注意】**

5735

5736 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
5737 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
5738 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
5739 ている可能性があるためである。

5740

5741 **【 μ ITRON4.0仕様との関係】**

5742

5743 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。
5744 また、参照できるタスクの状態から、強制待ち要求ネスト数 (suscnt) を除外
5745 した。

5746

5747 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
5748 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。

5749

5750 **【 μ ITRON4.0/PX仕様との関係】**

5751
5752 保護機能対応カーネルで参照できる情報として、タスク例外処理マスク状態か
5753 否か (texmsk) , 待ち禁止状態か否か (waifbd) , 拡張サービスコールのネス
5754 トレベル (svclevel) を追加した。
5755 -----
5756
5757 4.2 タスク付属同期機能
5758
5759 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
5760 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。
5761
5762 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。
5763
5764 ・起床要求キューイング数
5765
5766 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
5767 であり、タスクの起動時に0に初期化される。
5768
5769 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。
5770
5771 TMAX_WUPCNT タスクの起床要求キューイング数の最大値
5772
5773 【TOPPERS/ASPカーネルにおける規定】
5774
5775 ASPカーネルでは、TMAX_WUPCNTは1に固定されている。
5776
5777 【TOPPERS/FMPカーネルにおける規定】
5778
5779 FMPカーネルでは、TMAX_WUPCNTは1に固定されている。
5780
5781 【TOPPERS/HRP2カーネルにおける規定】
5782
5783 HRP2カーネルでは、TMAX_WUPCNTは1に固定されている。
5784
5785 【TOPPERS/SSPカーネルにおける規定】
5786
5787 SSPカーネルでは、タスク付属同期機能をサポートしない。
5788
5789 【μITRON4.0仕様との関係】
5790
5791 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。
5792 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
5793 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
5794 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。
5795 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
5796 スト数 (suscnt) を除外した。
5797 -----
5798 slp_tsk 起床待ち [T]
5799 tslp_tsk 起床待ち (タイムアウト付き) [T]
5800

5801 **【C言語API】**

5802 ER ercd = slp_tsk()
5803 ER ercd = tslp_tsk(TMO tmout)

5804

5805 **【パラメータ】**

5806 TMO tmout タイムアウト時間 (tslp_tskの場合)

5807

5808 **【リターンパラメータ】**

5809 ER ercd 正常終了 (E_OK) またはエラーコード

5810

5811 **【エラーコード】**

5812 E_CTX コンテキストエラー (ディスパッチ保留状態からの呼出し)

5813

5814 E_NOSPT 未サポート機能 (制約タスクからの呼出し)

5815 E_PAR パラメータエラー (tmoutが不正: tslp_tskの場合)

5816 E_TMOUT ポーリング失敗またはタイムアウト (slp_tskを除く)

5817 E_RLWAI 待ち禁止状態または待ち状態の強制解除

5818

5819 **【機能】**

5820

5821 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。

5822

5823 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
5824 数から1が減ぜられる。起床要求キューイング数が0の場合には、自タスクは起
5825 床待ち状態となる。

5826

5827 **【補足説明】**

5828

5829 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
5830 状態を維持し、自タスクの優先順位は変化しない。

5831

5832 wup_tsk タスクの起床 [T]

5833 iwup_tsk タスクの起床 [I]

5834

5835 **【C言語API】**

5836 ER ercd = wup_tsk(ID tskid)

5837 ER ercd = iwup_tsk(ID tskid)

5838

5839 **【パラメータ】**

5840 ID tskid 対象タスクのID番号

5841

5842 **【リターンパラメータ】**

5843 ER ercd 正常終了 (E_OK) またはエラーコード

5844

5845 **【エラーコード】**

5846 E_CTX コンテキストエラー (非タスクコンテキストからの呼出し : wup_tskの場合、タスクコンテキストからの呼出し :
5847 iwup_tskの場合、CPUロック状態からの呼出し)

5848 E_NOSPT 未サポート機能 (対象タスクが制約タスク)

5849 E_ID 不正ID番号 (tskidが不正)

5850

5851 E_NOEXS [D] オブジェクト未登録 (対象タスクが未登録)
 5852 E_OACV [P] オブジェクトアクセス違反 (対象タスクに対する通常操
 5853 作1が許可されていない: wup_tskの場合)
 5854 E_OBJ オブジェクト状態エラー (対象タスクが休止状態)
 5855 E_QOVR キューイングオーバフロー (起床要求キューイング数が
 5856 TMAX_WUPCNTに一致)
 5857

5858 【機能】

5859
 5860 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
 5861 り.

5862
 5863 対象タスクが起床待ち状態である場合には, 対象タスクが待ち解除される. 待
 5864 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.
 5865

5866 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
 5867 起床要求キューイング数に1が加えられる. 起床要求キューイング数に1を加え
 5868 るとTMAX_WUPCNTを超える場合には, E_QOVRエラーとなる.
 5869

5870 対象タスクが休止状態である場合には, E_OBJエラーとなる.
 5871

5872 wup_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
 5873 となる.
 5874

5875 -----
 5876 can_wup タスク起床要求のキャンセル [T]

5877 【C言語API】

5878 ER_UINT wupcnt = can_wup(ID tskid)
 5879

5880 【パラメータ】

5881 ID tskid 対象タスクのID番号
 5882

5883 【リターンパラメータ】

5884 ER_UINT wupcnt キューイングされていた起床要求の数 (正の値
 5885 または0) またはエラーコード
 5886

5887 【エラーコード】

5888 E_CTX コンテキストエラー (非タスクコンテキストからの呼出
 5889 し, CPUロック状態からの呼出し)
 5890 E_NOSPT 未サポート機能 (対象タスクが制約タスク)
 5891 E_ID 不正ID番号 (tskidが不正)
 5892 E_NOEXS [D] オブジェクト未登録 (対象タスクが未登録)
 5893 E_OACV [P] オブジェクトアクセス違反 (対象タスクに対する通常操
 5894 作1が許可されていない)
 5895 E_OBJ オブジェクト状態エラー (対象タスクが休止状態)
 5896

5897 【機能】

5898
 5899 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
 5900 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

5901 下の通り.

5902

5903 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
5904 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
5905 の返回值として返される.

5906

5907 対象タスクが休止状態である場合には、E_OBJエラーとなる.

5908

5909 tskidにTSK_SELF (=0) を指定すると、自タスクが対象タスクとなる.

5910

5911 rel_wai 強制的な待ち解除 [T]

5912 irel_wai 強制的な待ち解除 [I]

5913

5914 【C言語API】

5915 ER ercd = rel_wai(ID tskid)

5916 ER ercd = irel_wai(ID tskid)

5917

5918 【パラメータ】

5919 ID tskid 対象タスクのID番号

5920

5921 【リターンパラメータ】

5922 ER ercd 正常終了 (E_OK) またはエラーコード

5923

5924 【エラーコード】

5925 E_CTX コンテキストエラー (非タスクコンテキストからの呼出
5926 し: rel_waiの場合、タスクコンテキストからの呼出し:
5927 irel_waiの場合、CPUロック状態からの呼出し)

5928 E_NOSPT 未サポート機能 (対象タスクが制約タスク)

5929 E_ID 不正ID番号 (tskidが不正)

5930 E_NOEXS [D] オブジェクト未登録 (対象タスクが未登録)

5931 E_OACV [P] オブジェクトアクセス違反 (対象タスクに対する通常操
5932 作2が許可されていない: rel_waiの場合)

5933 E_OBJ オブジェクト状態エラー (対象タスクが待ち状態でない)

5934

5935 【機能】

5936

5937 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
5938 舞いは以下の通り.

5939

5940 対象タスクが待ち状態である場合には、対象タスクが待ち解除される. 待ち解
5941 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.

5942

5943 対象タスクが待ち状態でない場合には、E_OBJエラーとなる.

5944

5945 sus_tsk 強制待ち状態への遷移 [T]

5946

5947 【C言語API】

5948 ER ercd = sus_tsk(ID tskid)

5949

5950 【パラメータ】

5951 ID tskid 対象タスクのID番号
5952
5953 **【リターンパラメータ】**
5954 ER ercd 正常終了 (E_OK) またはエラーコード
5955
5956 **【エラーコード】**
5957 E_CTX コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し, 対象タスクが自タスク
5958 し, ディスパッチ保留状態からの呼出し)
5959 E_NOSPT 未サポート機能 (対象タスクが制約タスク)
5960 E_ID 不正ID番号 (tskidが不正)
5961 E_NOEXS [D] オブジェクト未登録 (対象タスクが未登録)
5962 E_OACV [P] オブジェクトアクセス違反 (対象タスクに対する通常操
5963 作2が許可されていない)
5964 E_OBJ オブジェクト状態エラー (対象タスクが休止状態)
5965 E_QOVR キューイングオーバーフロー (対象タスクが強制待ち状態)
5966
5967
5968 **【機能】**
5969
5970 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
5971 下の通り.
5972
5973 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
5974 る. また, 待ち状態 (二重待ち状態を除く) である場合には, 二重待ち状態と
5975 なる.
5976
5977 対象タスクが強制待ち状態または二重待ち状態である場合はE_QOVRエラー, 休
5978 止状態である場合にはE_OBJエラーとなる.
5979
5980 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,
5981 E_QOVRエラーとなる場合がある. この状況は, 自タスクに対してsus_tskを発行
5982 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
5983 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.
5984
5985 tskidにTSK_SELF (=0) を指定すると, 自タスクが対象タスクとなる.
5986
5987 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,
5988 E_CTXエラーとなる.
5989 -----
5990 rsm_tsk 強制待ち状態からの再開 [T]
5991
5992 **【C言語API】**
5993 ER ercd = rsm_tsk(ID tskid)
5994
5995 **【パラメータ】**
5996 ID tskid 対象タスクのID番号
5997
5998 **【リターンパラメータ】**
5999 ER ercd 正常終了 (E_OK) またはエラーコード
6000

6001 **【エラーコード】**

6002 E_CTX コンテキストエラー（非タスクコンテキストからの呼出し、CPUロック状態からの呼出し）

6003

6004 E_NOSPT 未サポート機能（対象タスクが制約タスク）

6005 E_ID 不正ID番号（tskidが不正）

6006 E_NOEXS [D] オブジェクト未登録（対象タスクが未登録）

6007 E_OACV [P] オブジェクトアクセス違反（対象タスクに対する通常操作2が許可されていない）

6008

6009 E_OBJ オブジェクト状態エラー（対象タスクが強制待ち状態でない）

6010

6011

6012 **【機能】**

6013

6014 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。

6015

6016

6017 対象タスクが強制待ち状態である場合には、対象タスクは強制待ちから再開される。強制待ち状態でない場合には、E_OBJエラーとなる。

6018

6019

6020 dis_wai 待ち禁止状態への遷移 [TP]

6021 idis_wai 待ち禁止状態への遷移 [IP]

6022

6023 **【C言語API】**

6024 ER ercd = dis_wai(ID tskid)

6025 ER ercd = idis_wai(ID tskid)

6026

6027 **【パラメータ】**

6028 ID tskid 対象タスクのID番号

6029

6030 **【リターンパラメータ】**

6031 ER ercd 正常終了 (E_OK) またはエラーコード

6032

6033 **【エラーコード】**

6034 E_CTX コンテキストエラー（非タスクコンテキストからの呼出し：dis_waiの場合、タスクコンテキストからの呼出し：idis_waiの場合、CPUロック状態からの呼出し）

6035

6036 E_NOSPT 未サポート機能（対象タスクが制約タスク）

6037 E_ID 不正ID番号（tskidが不正）

6038 E_NOEXS [D] オブジェクト未登録（対象タスクが未登録）

6039 E_OACV [P] オブジェクトアクセス違反（対象タスクに対する通常操作2が許可されていない：dis_waiの場合）

6040 E_OBJ オブジェクト状態エラー（対象タスクが休止状態、対象タスクがタスク例外処理マスク状態でない）

6041

6042 E_QOVR キューイングオーバーフロー（対象タスクが待ち禁止状態）

6043

6044

6045

6046 **【機能】**

6047

6048 tskidで指定したタスク（対象タスク）を待ち禁止状態にする。具体的な振舞いは以下の通り。

6049

6050

6051 対象タスクがタスク例外処理マスク状態であり，待ち禁止状態でない場合には，
6052 対象タスクは待ち禁止状態になる。
6053
6054 対象タスクが休止状態である場合には，E_OBJエラーとなる．また，対象タスク
6055 がタスク例外処理マスク状態でない場合にはE_OBJエラー，待ち禁止状態の場合
6056 にはE_QOVRエラーとなる。
6057
6058 dis_waiにおいてtskidにTSK_SELF (=0) を指定すると，自タスクが対象タスク
6059 となる。
6060
6061 **【TOPPERS/ASPカーネルにおける規定】**
6062
6063 ASPカーネルでは，dis_waiをサポートしない。
6064
6065 **【TOPPERS/FMPカーネルにおける規定】**
6066
6067 FMPカーネルでは，dis_waiをサポートしない。
6068
6069 **【補足説明】**
6070
6071 dis_waiは，対象タスクの待ち解除は行わない．対象タスクを待ち禁止状態にす
6072 ることに加えて待ち解除したい場合には，dis_waiを呼び出した後に，rel_wai
6073 を呼び出せばよい。
6074
6075 **【未決定事項】**
6076
6077 マルチプロセッサ対応カーネルでは，対象タスクを，自タスクと同じプロセッ
6078 サに割り付けられているタスクに限るなどの制限を導入する可能性があるが，
6079 現時点では未決定である。
6080
6081 **【μITRON4.0/PX仕様との関係】**
6082
6083 μITRON4.0/PX仕様に定義されていないサービスコールである。
6084 -----
6085 ena_wai 待ち禁止状態の解除 [TP]
6086 iena_wai 待ち禁止状態の解除 [IP]
6087
6088 **【C言語API】**
6089 ER ercd = ena_wai(ID tskid)
6090 ER ercd = iena_wai(ID tskid)
6091
6092 **【パラメータ】**
6093 ID tskid 対象タスクのID番号
6094
6095 **【リターンパラメータ】**
6096 ER ercd 正常終了 (E_OK) またはエラーコード
6097
6098 **【エラーコード】**
6099 E_CTX コンテキストエラー（非タスクコンテキストからの呼出
6100 し：ena_waiの場合，タスクコンテキストからの呼出し：

6101 iena_waiの場合、CPUロック状態からの呼出し)
 6102 E_NOSPT 未サポート機能 (対象タスクが制約タスク)
 6103 E_ID 不正ID番号 (tskidが不正)
 6104 E_NOEXS [D] オブジェクト未登録 (対象タスクが未登録)
 6105 E_OACV [P] オブジェクトアクセス違反 (対象タスクに対する通常操
 6106 作2が許可されていない: ena_waiの場合)
 6107 E_OBJ オブジェクト状態エラー (対象タスクが休止状態、対象
 6108 タスクが待ち禁止状態でない)
 6109
 6110 **【機能】**
 6111
 6112 tskidで指定したタスク (対象タスク) の待ち禁止状態を解除する。具体的な振
 6113 舞いは以下の通り。
 6114
 6115 対象タスクが待ち禁止状態である場合には、待ち禁止状態は解除される。対象
 6116 タスクが休止状態である場合や、待ち禁止状態でない場合には、E_OBJエラーと
 6117 なる。
 6118
 6119 ena_waiにおいてtskidにTSK_SELF (=0) を指定すると、自タスクが対象タスク
 6120 となる。
 6121
 6122 **【TOPPERS/ASPカーネルにおける規定】**
 6123
 6124 ASPカーネルでは、ena_waiをサポートしない。
 6125
 6126 **【TOPPERS/FMPカーネルにおける規定】**
 6127
 6128 FMPカーネルでは、ena_waiをサポートしない。
 6129
 6130 **【未決定事項】**
 6131
 6132 マルチプロセッサ対応カーネルでは、対象タスクを、自タスクと同じプロセッ
 6133 サに割り付けられているタスクに限るなどの制限を導入する可能性があるが、
 6134 現時点では未決定である。
 6135
 6136 **【 μ ITRON4.0/PX仕様との関係】**
 6137
 6138 μ ITRON4.0/PX仕様に定義されていないサービスコールである。
 6139 -----
 6140 dly_tsk 自タスクの遅延 [T]
 6141
 6142 **【C言語API】**
 6143 ER ercd = dly_tsk(RELTIM dlytim)
 6144
 6145 **【パラメータ】**
 6146 RELTIM dlytim 遅延時間
 6147
 6148 **【リターンパラメータ】**
 6149 ER ercd 正常終了 (E_OK) またはエラーコード
 6150

【エラーコード】

E_CTX	コンテキストエラー（ディスパッチ保留状態からの呼出し）
E_NOSPT	未サポート機能（制約タスクからの呼出し）
E_PAR	パラメータエラー（dlytimが不正）
E_RLWAI	待ち禁止状態または待ち状態の強制解除

【機能】

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

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

dlytimは，TMAX_RELTIM以下でなければならない．

4.3 タスク例外処理機能

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

タスク例外処理機能に関連して，各タスクが持つ情報は次の通り．

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

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

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

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

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

タスク例外処理ルーチンの実行が開始される時、タスク例外処理禁止フラグはセットされ、保留例外要因は0にクリアされる。また、タスク例外処理ルーチンからのリターン時には、タスク例外処理禁止フラグはクリアされる。

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

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

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

保護機能対応カーネルにおいて、タスク例外処理ルーチンは、タスクと同じ保護ドメインに属する。

タスク例外処理機能に用いるデータ型は次の通り。

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

C言語によるタスク例外処理ルーチンの記述形式は次の通り。

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

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

タスク例外処理機能に関連するカーネル構成マクロは次の通り。

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

【補足説明】

保護機能対応でないカーネルでは、タスク例外処理ルーチンの実行開始条件の内、「CPUロック状態でない」は省いても同じ結果になる。これは、CPUロック状態で他の条件が揃うことはないためである。一方、保護機能対応カーネルで

6251 は、CPUロック状態で拡張サービスコールからリターンした場合（より厳密には、
6252 タスク例外処理マスク状態が解除された場合）に、CPUロック状態で他の条件が
6253 揃うことになる。

6254

6255 **【TOPPERS/ASPカーネルにおける規定】**

6256

6257 ASPカーネルでは、タスク例外要因のビット数 (TBIT_TEXPTN) は16以上である。

6258

6259 **【TOPPERS/FMPカーネルにおける規定】**

6260

6261 FMPカーネルでは、タスク例外要因のビット数 (TBIT_TEXPTN) は16以上である。

6262

6263 **【TOPPERS/HRP2カーネルにおける規定】**

6264

6265 HRP2カーネルでは、タスク例外要因のビット数 (TBIT_TEXPTN) は16以上である。

6266

6267 **【TOPPERS/SSPカーネルにおける規定】**

6268

6269 SSPカーネルでは、タスク例外処理機能をサポートしない。

6270

6271 **【 μ ITRON4.0仕様との関係】**

6272

6273 割込み優先度マスク全解除状態でない場合には、タスク例外処理ルーチンの実
6274 行が開始されないという仕様に変更した。

6275

6276 **【 μ ITRON4.0/PX仕様との関係】**

6277

6278 ユーザタスクのタスク例外処理ルーチンの実行開始時とリターン時にユーザス
6279 タックが不正となる問題に関して、 μ ITRON4.0/PX仕様では考慮されていない。

6280

6281 **【仕様変更の経緯】**

6282

6283 この仕様のRelease 1.2以前では、タスク例外処理ルーチンの実行開始条件に
6284 「割込み優先度マスク全解除状態である」の条件がなかったが、Release1.3以
6285 降で追加した。これは、マルチプロセッサ対応カーネルにおいて、他プロセッ
6286 サで実行中のタスクに対してタスク例外処理を要求した場合に、割込み優先度
6287 マスクが全解除でないと、タスク例外処理ルーチンをただちに実行開始するこ
6288 とができないためである。なお、ASPカーネル Release 1.6以前と、FMPカーネ
6289 ルRelease 1.1.1以前のバージョンは、古い仕様に従って実装されている。

6290

6291 DEF_TEX タスク例外処理ルーチンの定義 [S]

6292 def_tex タスク例外処理ルーチンの定義 [TD]

6293

6294 **【静的API】**

6295 DEF_TEX(ID tskid, { ATR texatr, TEXRTN texrtn })

6296

6297 **【C言語API】**

6298 ER ercd = def_tex(ID tskid, const T_DTEX *pk_dtex)

6299

6300 **【パラメータ】**

6301	ID	tskid	対象タスクのID番号
6302	T_DTEX *	pk_dtex	タスク例外処理ルーチンの定義情報を入れたパ
6303			ケットへのポインタ (静的APIを除く)
6304			
6305	*タスク例外処理ルーチンの定義情報 (パケットの内容)		
6306	ATR	texatr	タスク例外処理ルーチン属性
6307	TEXRTN	texrtn	タスク例外処理ルーチンの先頭番地
6308			
6309	【リターンパラメータ】		
6310	ER	ercd	正常終了 (E_OK) またはエラーコード
6311			
6312	【エラーコード】		
6313	E_CTX [s]		コンテキストエラー (非タスクコンテキストからの呼出
6314			し, CPUロック状態からの呼出し)
6315	E_ID		不正ID番号 (tskidが不正)
6316	E_RSATR		予約属性 (texatrが不正または使用できない, 属する保
6317			護ドメインかクラスが不正)
6318	E_NOEXS [D]		オブジェクト未登録 (対象タスクが未登録)
6319	E_OACV [sP]		オブジェクトアクセス違反 (対象タスクに対する管理操
6320			作が許可されていない)
6321	E_MACV [sP]		メモリアクセス違反 (pk_dtexが指すメモリ領域への読出
6322			しアクセスが許可されていない)
6323	E_PAR		パラメータエラー (texrtnが不正)
6324	E_OBJ		オブジェクト状態エラー (タスク例外処理ルーチンを定
6325			義済みのタスクに対する定義, タスク例外処理ルーチン
6326			を未定義のタスクに対する解除, 対象タスクは静的APIで
6327			生成された: def_texの場合)
6328			
6329	【機能】		
6330			
6331	tskidで指定したタスク (対象タスク) に対して, 各パラメータで指定したタス		
6332	ク例外処理ルーチン定義情報に従って, タスク例外処理ルーチンを定義する.		
6333			
6334	ただし, def_texにおいてpk_dtexをNULLにした場合には, 対象タスクに対する		
6335	タスク例外処理ルーチンの定義を解除する. また, 対象タスクのタスク例外処		
6336	理禁止フラグをセットし, 保留例外要因を0に初期化する.		
6337			
6338	静的APIにおいては, tskidはオブジェクト識別名, texatrは整数定数式パラメー		
6339	タ, texrtnは一般定数式パラメータである.		
6340			
6341	静的APIによって生成したタスクに対しては, タスク例外処理ルーチンの登録は		
6342	DEF_TEXによって行わねばならず, def_texによってタスク例外処理ルーチンを		
6343	登録/登録解除することはできない. def_texにおいて, 対象タスクが静的API		
6344	で生成したタスクである場合には, E_OBJエラーとなる.		
6345			
6346	タスク例外処理ルーチンを定義する場合 (DEF_TEXの場合およびdef_texにおい		
6347	てpk_dtexをNULL以外にした場合) で, 対象タスクに対してすでにタスク例外処		
6348	理ルーチンが定義されている場合には, E_OBJエラーとなる.		
6349			
6350	保護機能対応カーネルにおいて, DEF_TEXは, 対象タスクが属する保護ドメイン		

6351 の囲みの中に記述しなければならない。そうでない場合には、E_RSATRエラーと
 6352 なる。また、def_texでタスク例外処理ルーチンを定義する場合には、タスク例
 6353 外処理ルーチンの属する保護ドメインを設定する必要はなく、タスク例外処理
 6354 ルーチン属性にTA_DOM(domid)を指定した場合にはE_RSATRエラーとなる。ただ
 6355 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
 6356 は検出されない。

6357

6358 タスク例外処理ルーチンの定義を解除する場合（def_texにおいてpk_dtexを
 6359 NULLにした場合）で、対象タスクに対してタスク例外処理ルーチンが定義され
 6360 ていない場合には、E_OBJエラーとなる。

6361

6362 def_texにおいてtskidにTSK_SELF（=0）を指定すると、自タスクが対象タスク
 6363 となる。

6364

6365 **【TOPPERS/ASPカーネルにおける規定】**

6366

6367 ASPカーネルでは、DEF_TEXのみをサポートする。ただし、動的生成機能拡張パッ
 6368 ケージでは、def_texもサポートする。

6369

6370 **【TOPPERS/FMPカーネルにおける規定】**

6371

6372 FMPカーネルでは、DEF_TEXのみをサポートする。

6373

6374 **【TOPPERS/HRP2カーネルにおける規定】**

6375

6376 HRP2カーネルでは、DEF_TEXのみをサポートする。

6377

6378 **【 μ ITRON4.0仕様との関係】**

6379

6380 texrtnのデータ型をTEXRTNに変更した。

6381

6382 def_texによって、定義済みのタスク例外処理ルーチンを再定義しようとした場
 6383 合に、E_OBJエラーとすることにした。

6384 -----

6385 ras_tex タスク例外処理の要求 [T]
 6386 iras_tex タスク例外処理の要求 [I]

6387

6388 **【C言語API】**

6389 ER ercd = ras_tex(ID tskid, TEXPTN rasptn)
 6390 ER ercd = iras_tex(ID tskid, TEXPTN rasptn)

6391

6392 **【パラメータ】**

6393 ID tskid 対象タスクのID番号
 6394 TEXPTN rasptn 要求するタスク例外処理のタスク例外要因

6395

6396 **【リターンパラメータ】**

6397 ER ercd 正常終了 (E_OK) またはエラーコード

6398

6399 **【エラーコード】**

6400 E_CTX コンテキストエラー（非タスクコンテキストからの呼出

6401 し：ras_texの場合、タスクコンテキストからの呼出し：
 6402 iras_texの場合、CPUロック状態からの呼出し)
 6403 E_ID 不正ID番号 (tskidが不正)
 6404 E_NOEXS [D] オブジェクト未登録 (対象タスクが未登録)
 6405 E_OACV [P] オブジェクトアクセス違反 (対象タスクに対する通常操
 6406 作2が許可されていない：ras_texの場合)
 6407 E_PAR パラメータエラー (rasptnが不正)
 6408 E_OBJ オブジェクト状態エラー (対象タスクが休止状態、対象
 6409 タスクに対してタスク例外処理ルーチンが定義されてい
 6410 ない)

6411 【機能】

6412
 6413
 6414 tskidで指定したタスク (対象タスク) に対して、rasptnで指定したタスク例外
 6415 要因のタスク例外処理を要求する。対象タスクの保留例外要因が、それまでの
 6416 値とrasptnで指定した値のビット毎論理和 (C言語の" \mid ") に更新される。

6417
 6418 対象タスクが休止状態である場合と、対象タスクに対してタスク例外処理ルー
 6419 チンが定義されていない場合には、E_OBJエラーとなる。

6420
 6421 ras_texにおいてtskidにTSK_SELF (=0) を指定すると、自タスクが対象タスク
 6422 となる。

6423
 6424 rasptnが0の場合には、E_PARエラーとなる。

6425 -----
 6426 dis_tex タスク例外処理の禁止 [T]

6427 【C言語API】

6428 ER ercd = dis_tex()

6429 【パラメータ】

6430 なし

6431 【リターンパラメータ】

6432 ER ercd 正常終了 (E_OK) またはエラーコード

6433 【エラーコード】

6434 E_CTX コンテキストエラー (非タスクコンテキストからの呼出
 6435 し、CPUロック状態からの呼出し)
 6436 E_OBJ オブジェクト状態エラー (自タスクに対してタスク例外
 6437 処理ルーチンが定義されていない)

6438 【機能】

6439
 6440 自タスクのタスク例外処理禁止フラグをセットする。すなわち、自タスクをタ
 6441 スク例外処理禁止状態に遷移させる。

6442
 6443 -----
 6444 ena_tex タスク例外処理の許可 [T]

6445 【C言語API】

6451 ER ercd = ena_tex()
6452
6453 **【パラメータ】**
6454 なし
6455
6456 **【リターンパラメータ】**
6457 ER ercd 正常終了 (E_OK) またはエラーコード
6458
6459 **【エラーコード】**
6460 E_CTX コンテキストエラー (非タスクコンテキストからの呼出し,
6461 CPUロック状態からの呼出し)
6462 E_OBJ オブジェクト状態エラー (自タスクに対してタスク例外
6463 処理ルーチンが定義されていない)
6464
6465 **【機能】**
6466
6467 自タスクのタスク例外処理禁止フラグをクリアする. すなわち, 自タスクをタ
6468 スク例外処理許可状態に遷移させる.
6469
6470 **【補足説明】**
6471
6472 タスク例外処理ルーチン中でena_texを呼び出すことにより, タスク例外処理ルー
6473 チンの多重起動を行うことができる. ただし, 多重起動の最大段数を制限する
6474 のは, アプリケーションの責任である.
6475 -----
6476 sns_tex タスク例外処理禁止状態の参照 [TI]
6477
6478 **【C言語API】**
6479 bool_t state = sns_tex()
6480
6481 **【パラメータ】**
6482 なし
6483
6484 **【リターンパラメータ】**
6485 bool_t state タスク例外処理禁止状態
6486
6487 **【機能】**
6488
6489 実行状態のタスクのタスク例外処理禁止フラグを参照する. 具体的な振舞いは
6490 以下の通り.
6491
6492 実行状態のタスクが, タスク例外処理禁止状態の場合にtrue, タスク例外処理
6493 許可状態の場合にfalseが返る. sns_texを非タスクコンテキストから呼び出し
6494 た場合で, 実行状態のタスクがない場合には, trueが返る.
6495
6496 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
6497 単位を実行しているプロセッサにおいて実行状態のタスクのタスク例外処理禁
6498 止フラグを参照する.
6499
6500 **【補足説明】**

6501
 6502 sns_texをタスクコンテキストから呼び出した場合、実行状態のタスクは自タス
 6503 クに一致する。
 6504 -----
 6505 ref_tex タスク例外処理の状態参照 [T]
 6506
 6507 【C言語API】
 6508 ER ercd = ref_tex(ID tskid, T_RTEX *pk_rtex)
 6509
 6510 【パラメータ】
 6511 ID tskid 対象タスクのID番号
 6512 T_RTEX * pk_rtex タスク例外処理の現在状態を入れるパケットへ
 6513 のポインタ
 6514
 6515 【リターンパラメータ】
 6516 ER ercd 正常終了 (E_OK) またはエラーコード
 6517
 6518 *タスク例外処理の現在状態 (パケットの内容)
 6519 STAT texstat タスク例外処理の状態
 6520 TEXPTN pndptn タスクの保留例外要因
 6521
 6522 【エラーコード】
 6523 E_CTX コンテキストエラー (非タスクコンテキストからの呼出
 6524 し, CPUロック状態からの呼出し)
 6525 E_ID 不正ID番号 (tskidが不正)
 6526 E_NOEXS [D] オブジェクト未登録 (対象タスクが未登録)
 6527 E_OACV [P] オブジェクトアクセス違反 (対象タスクに対する参照操
 6528 作が許可されていない)
 6529 E_MACV [P] メモリアクセス違反 (pk_rtexが指すメモリ領域への書込
 6530 みアクセスが許可されていない)
 6531 E_OBJ オブジェクト状態エラー (対象タスクが休止状態, 対象
 6532 タスクに対してタスク例外処理ルーチンが定義されてい
 6533 ない)
 6534
 6535 【機能】
 6536
 6537 tskidで指定したタスク (対象タスク) のタスク例外処理に関する現在状態を参
 6538 照する。参照した現在状態は, pk_rtexで指定したパケットに返される。
 6539
 6540 texstatには, 対象タスクの現在のタスク例外処理禁止フラグを表す次のいずれ
 6541 かの値が返される。
 6542
 6543 TTEX_ENA 0x01U タスク例外処理許可状態
 6544 TTEX_DIS 0x02U タスク例外処理禁止状態
 6545
 6546 pndptnには, 対象タスクの現在の保留例外要因が返される。
 6547
 6548 tskidにTSK_SELF (=0) を指定すると, 自タスクが対象タスクとなる。
 6549 -----
 6550

6551 4.4 同期・通信機能

6552

6553 【TOPPERS/SSPカーネルにおける規定】

6554

6555 SSPカーネルでは、同期・通信機能をサポートしない．．

6556

6557 【μITRON4.0仕様との関係】

6558

6559 この仕様では、ランデブ機能はサポートしていない．今後の検討により、ラン
6560 デブ機能をサポートすることに変更する可能性もある．

6561

6562 4.4.1 セマフォ

6563

6564 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、
6565 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
6566 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
6567 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．

6568

6569 各セマフォが持つ情報は次の通り．

6570

- 6571 ・セマフォ属性
- 6572 ・資源数（の現在値）
- 6573 ・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）
- 6574 ・初期資源数
- 6575 ・最大資源数
- 6576 ・アクセス許可ベクタ（保護機能対応カーネルの場合）
- 6577 ・属する保護ドメイン（保護機能対応カーネルの場合）
- 6578 ・属するクラス（マルチプロセッサ対応カーネルの場合）

6579

6580 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
6581 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
6582 である．

6583

6584 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
6585 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
6586 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、
6587 E_QOVRエラーとなる．

6588

6589 セマフォ属性には、次の属性を指定することができる．

6590

6591 TA_TPRI 0x01U 待ち行列をタスクの優先度順にする

6592

6593 TA_TPRIを指定しない場合、待ち行列はFIFO順になる．

6594

6595 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．

6596

6597 TMAX_MAXSEM セマフォの最大資源数の最大値（=UINT_MAX）

6598

6599 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
6600 では、静的APIによって登録されたセマフォの数に一致）

6601

6602 **【 μ ITRON4.0仕様との関係】**

6603

6604 TNUM_SEMIDは、 μ ITRON4.0仕様に規定されていないカーネル構成マクロである。

6605

6606 CRE_SEM セマフォの生成 [S]

6607 acre_sem セマフォの生成 [TD]

6608

6609 **【静的API】**

6610 CRE_SEM(ID semid, { ATR sematr, uint_t isemcnt, uint_t maxsem })

6611

6612 **【C言語API】**

6613 ER_ID semid = acre_sem(const T_CSEM *pk_csem)

6614

6615 **【パラメータ】**

6616 ID semid 生成するセマフォのID番号 (CRE_SEMの場合)

6617 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
6618 ンタ (静的APIを除く)

6619

6620 *セマフォの生成情報 (パケットの内容)

6621 ATR sematr セマフォ属性

6622 uint_t isemcnt セマフォの初期資源数

6623 uint_t maxsem セマフォの最大資源数

6624

6625 **【リターンパラメータ】**6626 ER_ID semid 生成されたセマフォのID番号 (正の値) または
6627 エラーコード

6628

6629 **【エラーコード】**6630 E_CTX [s] コンテキストエラー (非タスクコンテキストからの呼出
6631 し, CPUロック状態からの呼出し)6632 E_RSATR 予約属性 (sematrが不正または使用できない, 属する保
6633 護ドメインがクラスが不正)

6634 E_PAR パラメータエラー (isemcnt, maxsemが不正)

6635 E_OACV [sP] オブジェクトアクセス違反 (システム状態に対する管理
6636 操作が許可されていない)6637 E_MACV [sP] メモリアクセス違反 (pk_csemが指すメモリ領域への読出
6638 しアクセスが許可されていない)

6639 E_NOID [sD] ID番号不足 (割り付けられるセマフォIDがない)

6640 E_OBJ オブジェクト状態エラー (semidで指定したセマフォが登
6641 録済み: CRE_SEMの場合)

6642

6643 **【機能】**

6644

6645 各パラメータで指定したセマフォ生成情報に従って, セマフォを生成する. 生
6646 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
6647 る.

6648

6649 静的APIにおいては, semidはオブジェクト識別名, isemcntとmaxsemは整数定数
6650 式パラメータである.

6651
6652 isemcntは、0以上で、maxsem以下でなければならない。また、maxsemは、1以上
6653 で、TMAX_MAXSEM以下でなければならない。
6654
6655 **【TOPPERS/ASPカーネルにおける規定】**
6656
6657 ASPカーネルでは、CRE_SEMのみをサポートする。ただし、動的生成機能拡張パッ
6658 ケージでは、acre_semもサポートする。
6659
6660 **【TOPPERS/FMPカーネルにおける規定】**
6661
6662 FMPカーネルでは、CRE_SEMのみをサポートする。
6663
6664 **【TOPPERS/HRP2カーネルにおける規定】**
6665
6666 HRP2カーネルでは、CRE_SEMのみをサポートする。
6667 -----
6668 AID_SEM 割付け可能なセマフォIDの数の指定 [SD]
6669
6670 **【静的API】**
6671 AID_SEM(uint_t nosem)
6672
6673 **【パラメータ】**
6674 uint_t nosem 割付け可能なセマフォIDの数
6675
6676 **【エラーコード】**
6677 E_RSATR 予約属性（属する保護ドメインまたはクラスが不正）
6678
6679 **【機能】**
6680
6681 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
6682 て割付け可能なセマフォIDとして確保する。
6683
6684 nosemは整数定数式パラメータである。
6685 -----
6686 SAC_SEM セマフォのアクセス許可ベクタの設定 [SP]
6687 sac_sem セマフォのアクセス許可ベクタの設定 [TPD]
6688
6689 **【静的API】**
6690 SAC_SEM(ID semid, { ACPTN acptn1, ACPTN acptn2,
6691 ACPTN acptn3, ACPTN acptn4 })
6692
6693 **【C言語API】**
6694 ER ercd = sac_sem(ID semid, const ACVCT *p_acvct)
6695
6696 **【パラメータ】**
6697 ID semid 対象セマフォのID番号
6698 ACVCT * p_acvct アクセス許可ベクタを入れたパケットへのポ
6699 インタ（静的APIを除く）
6700

6701 *アクセス許可ベクタ (パケットの内容)

6702 ACPTN acptn1 通常操作1のアクセス許可パターン

6703 ACPTN acptn2 通常操作2のアクセス許可パターン

6704 ACPTN acptn3 管理操作のアクセス許可パターン

6705 ACPTN acptn4 参照操作のアクセス許可パターン

6706

6707 **【リターンパラメータ】**

6708 ER ercd 正常終了 (E_OK) またはエラーコード

6709

6710 **【エラーコード】**

6711 E_CTX [s] コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)

6712

6713 E_ID 不正ID番号 (semidが不正)

6714 E_RSATR 予約属性 (属する保護ドメインかクラスが不正: SAC_SEMの場合)

6715

6716 E_NOEXS [D] オブジェクト未登録 (対象セマフォが未登録)

6717 E_OACV [sP] オブジェクトアクセス違反 (対象セマフォに対する管理操作が許可されていない)

6718

6719 E_MACV [sP] メモリアクセス違反 (p_acvctが指すメモリ領域への読出しアクセスが許可されていない)

6720

6721 E_OBJ オブジェクト状態エラー (対象セマフォは静的APIで生成された: sac_semの場合, 対象セマフォに対してアクセス許可ベクタが設定済み: SAC_SEMの場合)

6722

6723

6724

6725 **【機能】**

6726

6727 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.

6728

6729

6730 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.

6731

6732

6733 SAC_SEMは, 対象セマフォが属する保護ドメインの囲みの中に記述しなければならない. そうでない場合には, E_RSATRエラーとなる.

6734

6735

6736 **【TOPPERS/ASPカーネルにおける規定】**

6737

6738 ASPカーネルでは, SAC_SEM, sac_semをサポートしない.

6739

6740 **【TOPPERS/FMPカーネルにおける規定】**

6741

6742 FMPカーネルでは, SAC_SEM, sac_semをサポートしない.

6743

6744 **【TOPPERS/HRP2カーネルにおける規定】**

6745

6746 HRP2カーネルでは, SAC_SEMのみをサポートする.

6747 -----

6748 del_sem セマフォの削除 [TD]

6749

6750 **【C言語API】**

```

6751         ER ercd = del_sem(ID semid)
6752
6753     【パラメータ】
6754         ID          semid      対象セマフォのID番号
6755
6756     【リターンパラメータ】
6757         ER          ercd      正常終了 (E_OK) またはエラーコード
6758
6759     【エラーコード】
6760         E_CTX      コンテキストエラー (非タスクコンテキストからの呼出
6761                  し, CPUロック状態からの呼出し)
6762         E_ID       不正ID番号 (semidが不正)
6763         E_NOEXS [D] オブジェクト未登録 (対象セマフォが未登録)
6764         E_OACV [P] オブジェクトアクセス違反 (対象セマフォに対する管理
6765                  操作が許可されていない)
6766         E_OBJ      オブジェクト状態エラー (対象セマフォは静的APIで生成
6767                  された)
6768
6769     【機能】
6770
6771     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
6772     の通り.
6773
6774     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.
6775     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
6776     クから順に待ち解除される. 待ち解除されたタスクには, 待ち状態となったサー
6777     ビスコールからE_DLTエラーが返る.
6778
6779     【使用上の注意】
6780
6781     del_semにより複数のタスクが待ち解除される場合, サービスコールの処理時間
6782     およびカーネル内での割込み禁止時間が, 待ち解除されるタスクの数に比例し
6783     て長くなる. 特に, 多くのタスクが待ち解除される場合, カーネル内での割込
6784     み禁止時間が長くなるため, 注意が必要である.
6785
6786     【TOPPERS/ASPカーネルにおける規定】
6787
6788     ASPカーネルでは, del_semをサポートしない. ただし, 動的生成機能拡張パッ
6789     ケージでは, del_semをサポートする.
6790
6791     【TOPPERS/FMPカーネルにおける規定】
6792
6793     FMPカーネルでは, del_semをサポートしない.
6794
6795     【TOPPERS/HRP2カーネルにおける規定】
6796
6797     HRP2カーネルでは, del_semをサポートしない.
6798     -----
6799     sig_sem      セマフォの資源の返却 [T]
6800     isig_sem     セマフォの資源の返却 [I]

```



```

6801
6802  【C言語API】
6803      ER ercd = sig_sem(ID semid)
6804      ER ercd = isig_sem(ID semid)
6805
6806  【パラメータ】
6807      ID          semid      対象セマフォのID番号
6808
6809  【リターンパラメータ】
6810      ER          ercd      正常終了 (E_OK) またはエラーコード
6811
6812  【エラーコード】
6813      E_CTX      コンテキストエラー (非タスクコンテキストからの呼出
6814                  し: sig_semの場合, タスクコンテキストからの呼出し:
6815                  isig_semの場合, CPUロック状態からの呼出し)
6816      E_ID      不正ID番号 (semidが不正)
6817      E_NOEXS [D] オブジェクト未登録 (対象セマフォが未登録)
6818      E_OACV [P] オブジェクトアクセス違反 (対象セマフォに対する通常
6819                  操作1が許可されていない: sig_semの場合)
6820      E_QOVR      キューイングオーバーフロー (資源数が最大資源数に一致)
6821
6822  【機能】
6823
6824  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
6825  は以下の通り.
6826
6827  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
6828  クが待ち解除される. この時, 待ち解除されたタスクが資源を獲得したこと
6829  になるため, 対象セマフォの資源数は変化しない. 待ち解除されたタスクには,
6830  待ち状態となったサービスコールからE_OKが返る.
6831
6832  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
6833  る. 資源数に1を加えるとそのセマフォの最大資源数を越える場合には, E_QOVR
6834  エラーとなる.
6835  -----
6836  wai_sem      セマフォの資源の獲得 [T]
6837  pol_sem      セマフォの資源の獲得 (ポーリング) [T]
6838  twai_sem     セマフォの資源の獲得 (タイムアウト付き) [T]
6839
6840  【C言語API】
6841      ER ercd = wai_sem(ID semid)
6842      ER ercd = pol_sem(ID semid)
6843      ER ercd = twai_sem(ID semid, TMO tmout)
6844
6845  【パラメータ】
6846      ID          semid      対象セマフォのID番号
6847      TMO          tmout     タイムアウト時間 (twai_semの場合)
6848
6849  【リターンパラメータ】
6850      ER          ercd      正常終了 (E_OK) またはエラーコード

```

6851
6852 **【エラーコード】**
6853 E_CTX コンテキストエラー（非タスクコンテキストからの呼出
6854 し、CPUロック状態からの呼出し、ディスパッチ保留状態
6855 からの呼出し：pol_semを除く）
6856 E_NOSPT 未サポート機能（制約タスクからの呼出し：pol_semを除
6857 く）
6858 E_ID 不正ID番号（semidが不正）
6859 E_PAR パラメータエラー（tmoutが不正：twai_semの場合）
6860 E_NOEXS [D] オブジェクト未登録（対象セマフォが未登録）
6861 E_OACV [P] オブジェクトアクセス違反（対象セマフォに対する通常
6862 操作2が許可されていない）
6863 E_TMOUT ポーリング失敗またはタイムアウト（wai_semを除く）
6864 E_RLWAI 待ち禁止状態または待ち状態の強制解除（pol_semを除く）
6865 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）
6866
6867 **【機能】**
6868
6869 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
6870 いは以下の通り．
6871
6872 対象セマフォの資源数が1以上の場合には、資源数から1が減ぜられる．資源数
6873 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
6874 の待ち行列につながれる．
6875 -----
6876 ini_sem セマフォの再初期化 [T]
6877
6878 **【C言語API】**
6879 ER ercd = ini_sem(ID semid)
6880
6881 **【パラメータ】**
6882 ID semid 対象セマフォのID番号
6883
6884 **【リターンパラメータ】**
6885 ER ercd 正常終了（E_OK）またはエラーコード
6886
6887 **【エラーコード】**
6888 E_CTX コンテキストエラー（非タスクコンテキストからの呼出
6889 し、CPUロック状態からの呼出し）
6890 E_ID 不正ID番号（semidが不正）
6891 E_NOEXS [D] オブジェクト未登録（対象セマフォが未登録）
6892 E_OACV [P] オブジェクトアクセス違反（対象セマフォに対する管理
6893 操作が許可されていない）
6894
6895 **【機能】**
6896
6897 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
6898 以下の通り．
6899
6900 対象セマフォの資源数は、初期資源数に初期化される．また、対象セマフォの

6901 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
 6902 れる。待ち解除されたタスクには、待ち状態となったサービスコールから
 6903 E_DLTエラーが返る。

6904

6905 **【使用上の注意】**

6906

6907 ini_semにより複数のタスクが待ち解除される場合、サービスコールの処理時間
 6908 およびカーネル内での割込み禁止時間が、待ち解除されるタスクの数に比例し
 6909 て長くなる。特に、多くのタスクが待ち解除される場合、カーネル内での割込
 6910 み禁止時間が長くなるため、注意が必要である。

6911

6912 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
 6913 プリケーションの責任である。

6914

6915 **【 μ ITRON4.0仕様との関係】**

6916

6917 μ ITRON4.0仕様に定義されていないサービスコールである。

6918

6919 ref_sem セマフォの状態参照 [T]

6920

6921 **【C言語API】**

6922 ER ercd = ref_sem(ID semid, T_RSEM *pk_rsem)

6923

6924 **【パラメータ】**

6925	ID	semid	対象セマフォのID番号
6926	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

6927

6928

6929 **【リターンパラメータ】**

6930	ER	ercd	正常終了 (E_OK) またはエラーコード
------	----	------	-----------------------

6931

6932 *セマフォの現在状態 (パケットの内容)

6933	ID	wtskid	セマフォの待ち行列の先頭のタスクのID番号
------	----	--------	-----------------------

6934	uint_t	semcnt	セマフォの資源数
------	--------	--------	----------

6935

6936 **【エラーコード】**

6937	E_CTX	コンテキストエラー (非タスクコンテキストからの呼出
------	-------	----------------------------

6938		し, CPUロック状態からの呼出し)
------	--	--------------------

6939	E_ID	不正ID番号 (semidが不正)
------	------	-------------------

6940	E_NOEXS [D]	オブジェクト未登録 (対象セマフォが未登録)
------	-------------	------------------------

6941	E_OACV [P]	オブジェクトアクセス違反 (対象セマフォに対する参照 操作が許可されていない)
------	------------	--

6943	E_MACV [P]	メモリアクセス違反 (pk_rsemが指すメモリ領域への書込 みアクセスが許可されていない)
------	------------	---

6944

6945

6946 **【機能】**

6947

6948 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
 6949 在状態は、pk_rsemで指定したパケットに返される。

6950

6951 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.

6952

6953

6954

6955

6956

6957

6958

6959

6960

6961

6962

6963

6964

6965

6966

6967

6968

6969

6970

6971

6972

6973

6974

6975

6976

6977

6978

6979

6980

6981

6982

6983

6984

6985

6986

6987

6988

6989

6990

6991

6992

6993

6994

6995

6996

6997

6998

6999

7000

【使用上の注意】

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

4.4.2 イベントフラグ

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

1つまたは複数のビットをセットする1にする（セットする）ことを、イベントフラグをセットするといい、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。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。この条件を、イベントフラグの待ち解除の条件と呼ぶ。

各イベントフラグが持つ情報は次の通り。

- ・ イベントフラグ属性
- ・ ビットパターン（の現在値）
- ・ 待ち行列（イベントフラグ待ち状態のタスクのキュー）
- ・ 初期ビットパターン
- ・ 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（「2.6.4 待ち行列と待ち解除の順序」の節の(a)に該当）。

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

7001 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。
 7002
 7003 TA_TPRI 0x01U 待ち行列をタスクの優先度順にする
 7004 TA_WMUL 0x02U 複数のタスクが待つのを許す
 7005 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
 7006
 7007 TA_TPRIを指定しない場合、待ち行列はFIFO順になる。TA_WMULを指定しない場
 7008 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。
 7009
 7010 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
 7011 ンを0にクリアする。TA_CLRを指定しない場合、タスクの待ち解除時にイベント
 7012 フラグをクリアしない。
 7013
 7014 イベントフラグ機能に用いるデータ型は次の通り。
 7015
 7016 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
 7017 定義）
 7018
 7019 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。
 7020
 7021 TBIT_FLGPTN イベントフラグのビット数（FLGPTNの有効ビット数）
 7022
 7023 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
 7024 ネルでは、静的APIによって登録されたイベントフラグの
 7025 数に一致）
 7026
 7027 **【TOPPERS/ASPカーネルにおける規定】**
 7028
 7029 ASPカーネルでは、イベントフラグのビット数（TBIT_FLGPTN）は16以上である。
 7030
 7031 **【TOPPERS/FMPカーネルにおける規定】**
 7032
 7033 FMPカーネルでは、イベントフラグのビット数（TBIT_FLGPTN）は16以上である。
 7034
 7035 **【TOPPERS/HRP2カーネルにおける規定】**
 7036
 7037 HRP2カーネルでは、イベントフラグのビット数（TBIT_FLGPTN）は16以上である。
 7038
 7039 **【 μ ITRON4.0仕様との関係】**
 7040
 7041 TNUM_FLGIDは、 μ ITRON4.0仕様に規定されていないカーネル構成マクロである。
 7042 -----
 7043 CRE_FLG イベントフラグの生成 [S]
 7044 acre_flg イベントフラグの生成 [TD]
 7045
 7046 **【静的API】**
 7047 CRE_FLG(ID flgid, { ATR flgatr, FLGPTN iflgptn })
 7048
 7049 **【C言語API】**
 7050 ER_ID flgid = acre_flg(const T_CFLG *pk_cflg)

7051

7052 **【パラメータ】**

7053	ID	flgid	生成するイベントフラグのID番号 (CRE_FLGの
7054			場合)
7055	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
7056			のポインタ (静的APIを除く)

7057

7058 * イベントフラグの生成情報 (パケットの内容)

7059	ATR	flgatr	イベントフラグ属性
7060	FLGPTN	iflgptn	イベントフラグの初期ビットパターン

7061

7062 **【リターンパラメータ】**

7063	ER_ID	flgid	生成されたイベントフラグのID番号 (正の値)
7064			またはエラーコード

7065

7066 **【エラーコード】**

7067	E_CTX [s]	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
7068		
7069	E_RSATR	予約属性 (flgatrが不正または使用できない, 属する保護ドメインがクラスが不正)
7070		
7071	E_OACV [sP]	オブジェクトアクセス違反 (システム状態に対する管理操作が許可されていない)
7072		
7073	E_MACV [sP]	メモリアクセス違反 (pk_cflgが指すメモリ領域への読出しアクセスが許可されていない)
7074		
7075	E_NOID [sD]	ID番号不足 (割り付けられるイベントフラグIDがない)
7076	E_OBJ	オブジェクト状態エラー (flgidで指定したイベントフラグが登録済み: CRE_FLGの場合)
7077		

7078

7079 **【機能】**

7080

7081 各パラメータで指定したイベントフラグ生成情報に従って, イベントフラグを

7082 生成する. 生成されたイベントフラグのビットパターンは初期ビットパターン

7083 に, 待ち行列は空の状態に初期化される.

7084

7085 静的APIにおいては, flgidはオブジェクト識別名, iflgptnは整数定数式パラメータである.

7086

7087

7088 **【TOPPERS/ASPカーネルにおける規定】**

7089

7090 ASPカーネルでは, CRE_FLGのみをサポートする. ただし, 動的生成機能拡張パッケージでは, acre_flgもサポートする.

7091

7092

7093 **【TOPPERS/FMPカーネルにおける規定】**

7094

7095 FMPカーネルでは, CRE_FLGのみをサポートする.

7096

7097 **【TOPPERS/HRP2カーネルにおける規定】**

7098

7099 HRP2カーネルでは, CRE_FLGのみをサポートする.

7100 -----

7101 AID_FLG 割付け可能なイベントフラグIDの数の指定 [SD]

7102

7103 **【静的API】**

7104 AID_FLG(uint_t noflg)

7105

7106 **【パラメータ】**

7107 uint_t noflg 割付け可能なイベントフラグIDの数

7108

7109 **【エラーコード】**

7110 E_RSATR 予約属性（属する保護ドメインまたはクラスが不正）

7111

7112 **【機能】**

7113

7114 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
7115 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。

7116

7117 noflgは整数定数式パラメータである。

7118

7119 SAC_FLG イベントフラグのアクセス許可ベクタの設定 [SP]

7120 sac_flg イベントフラグのアクセス許可ベクタの設定 [TPD]

7121

7122 **【静的API】**

7123 SAC_FLG(ID flgid, { ACPTN acptn1, ACPTN acptn2,
7124 ACPTN acptn3, ACPTN acptn4 })

7125

7126 **【C言語API】**

7127 ER ercd = sac_flg(ID flgid, const ACVCT *p_acvct)

7128

7129 **【パラメータ】**

7130 ID flgid 対象イベントフラグのID番号
7131 ACVCT * p_acvct アクセス許可ベクタを入れたパケットへのポ
7132 インタ（静的APIを除く）

7133

7134 *アクセス許可ベクタ（パケットの内容）

7135 ACPTN acptn1 通常操作1のアクセス許可パターン

7136 ACPTN acptn2 通常操作2のアクセス許可パターン

7137 ACPTN acptn3 管理操作のアクセス許可パターン

7138 ACPTN acptn4 参照操作のアクセス許可パターン

7139

7140 **【リターンパラメータ】**

7141 ER ercd 正常終了 (E_OK) またはエラーコード

7142

7143 **【エラーコード】**

7144 E_CTX [s] コンテキストエラー（非タスクコンテキストからの呼出
7145 し、CPUロック状態からの呼出し）

7146 E_ID 不正ID番号 (flgidが不正)

7147 E_RSATR 予約属性（属する保護ドメインかクラスが不正：SAC_FLG
7148 の場合）

7149 E_NOEXS [D] オブジェクト未登録（対象イベントフラグが未登録）

7150 E_OACV [sP] オブジェクトアクセス違反（対象イベントフラグに対す

7151 る管理操作が許可されていない)
 7152 E_MACV [sP] メモリアクセス違反 (p_acvctが指すメモリ領域への読出
 7153 しアクセスが許可されていない)
 7154 E_OBJ オブジェクト状態エラー (対象イベントフラグは静的API
 7155 で生成された：sac_flgの場合、対象イベントフラグに対
 7156 してアクセス許可ベクタが設定済み：SAC_FLGの場合)
 7157
 7158 **【機能】**
 7159
 7160 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
 7161 (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。
 7162
 7163 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
 7164 式パラメータである。
 7165
 7166 SAC_FLGは、対象イベントフラグが属する保護ドメインの囲みの中に記述しなけ
 7167 ればならない。そうでない場合には、E_RSATRエラーとなる。
 7168
 7169 **【TOPPERS/ASPカーネルにおける規定】**
 7170
 7171 ASPカーネルでは、SAC_FLG、sac_flgをサポートしない。
 7172
 7173 **【TOPPERS/FMPカーネルにおける規定】**
 7174
 7175 FMPカーネルでは、SAC_FLG、sac_flgをサポートしない。
 7176
 7177 **【TOPPERS/HRP2カーネルにおける規定】**
 7178
 7179 HRP2カーネルでは、SAC_FLGのみをサポートする。
 7180 -----
 7181 del_flg イベントフラグの削除 [TD]
 7182
 7183 **【C言語API】**
 7184 ER ercd = del_flg(ID flgid)
 7185
 7186 **【パラメータ】**
 7187 ID flgid 対象イベントフラグのID番号
 7188
 7189 **【リターンパラメータ】**
 7190 ER ercd 正常終了 (E_OK) またはエラーコード
 7191
 7192 **【エラーコード】**
 7193 E_CTX コンテキストエラー (非タスクコンテキストからの呼出
 7194 し、CPUロック状態からの呼出し)
 7195 E_ID 不正ID番号 (flgidが不正)
 7196 E_NOEXS [D] オブジェクト未登録 (対象イベントフラグが未登録)
 7197 E_OACV [P] オブジェクトアクセス違反 (対象イベントフラグに対す
 7198 る管理操作が許可されていない)
 7199 E_OBJ オブジェクト状態エラー (対象イベントフラグは静的API
 7200 で生成された)

7201

7202 **【機能】**

7203

7204 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
7205 振舞いは以下の通り．

7206

7207 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
7208 に戻される．また，対象イベントフラグの待ち行列につながれたタスクは，待
7209 ち行列の先頭のタスクから順に待ち解除される．待ち解除されたタスクには，
7210 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．

7211

7212 **【使用上の注意】**

7213

7214 del_flgにより複数のタスクが待ち解除される場合，サービスコールの処理時間
7215 およびカーネル内での割込み禁止時間が，待ち解除されるタスクの数に比例し
7216 て長くなる．特に，多くのタスクが待ち解除される場合，カーネル内での割込
7217 み禁止時間が長くなるため，注意が必要である．

7218

7219 **【TOPPERS/ASPカーネルにおける規定】**

7220

7221 ASPカーネルでは，del_flgをサポートしない．ただし，動的生成機能拡張パッ
7222 ケージでは，del_flgをサポートする．

7223

7224 **【TOPPERS/FMPカーネルにおける規定】**

7225

7226 FMPカーネルでは，del_flgをサポートしない．

7227

7228 **【TOPPERS/HRP2カーネルにおける規定】**

7229

7230 HRP2カーネルでは，del_flgをサポートしない．

7231

7232 set_flg イベントフラグのセット [T]

7233 iset_flg イベントフラグのセット [I]

7234

7235 **【C言語API】**

7236 ER ercd = set_flg(ID flgid, FLGPTN setptn)

7237 ER ercd = iset_flg(ID flgid, FLGPTN setptn)

7238

7239 **【パラメータ】**

7240 ID flgid 対象イベントフラグのID番号

7241 FLGPTN setptn セットするビットパターン

7242

7243 **【リターンパラメータ】**

7244 ER ercd 正常終了 (E_OK) またはエラーコード

7245

7246 **【エラーコード】**

7247 E_CTX コンテキストエラー（非タスクコンテキストからの呼出
7248 し：set_flgの場合，タスクコンテキストからの呼出し：

7249 iset_flgの場合，CPUロック状態からの呼出し）

7250 E_ID 不正ID番号（flgidが不正）

7251 E_NOEXS [D] オブジェクト未登録（対象イベントフラグが未登録）
 7252 E_OACV [P] オブジェクトアクセス違反（対象イベントフラグに対す
 7253 る通常操作1が許可されていない：set_flgの場合）
 7254

7255 【機能】

7256
 7257 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
 7258 をセットする．具体的な振舞いは以下の通り．
 7259

7260 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
 7261 のビット毎論理和（C言語の" \mid "）に更新される．対象イベントフラグの待ち行
 7262 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
 7263 列の前方につながれたものから順に待ち解除される．待ち解除されたタスクに
 7264 は，待ち状態となったサービスコールからE_OKが返る．
 7265

7266 ただし，対象イベントフラグがTA_CLR属性である場合には，待ち解除の条件を
 7267 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
 7268 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．
 7269

7270 【使用上の注意】

7271
 7272 対象イベントフラグが，TA_WMUL属性であり，TA_CLR属性でない場合，set_flg
 7273 またはiset_flgにより複数のタスクが待ち解除される場合がある．この場合，
 7274 サービスコールの処理時間およびカーネル内での割込み禁止時間が，待ち解除
 7275 されるタスクの数に比例して長くなる．特に，多くのタスクが待ち解除される
 7276 場合，カーネル内での割込み禁止時間が長くなるため，注意が必要である．
 7277

7278 clr_flg イベントフラグのクリア [T]
 7279

7280 【C言語API】

7281 ER ercd = clr_flg(ID flgid, FLGPTN clrptn)
 7282

7283 【パラメータ】

7284 ID flgid 対象イベントフラグのID番号
 7285 FLGPTN clrptn クリアするビットパターン（クリアしないビッ
 7286 トを1，クリアするビットを0とする）
 7287

7288 【リターンパラメータ】

7289 ER ercd 正常終了（E_OK）またはエラーコード
 7290

7291 【エラーコード】

7292 E_CTX コンテキストエラー（非タスクコンテキストからの呼出
 7293 し，CPUロック状態からの呼出し）
 7294 E_ID 不正ID番号（flgidが不正）
 7295 E_NOEXS [D] オブジェクト未登録（対象イベントフラグが未登録）
 7296 E_OACV [P] オブジェクトアクセス違反（対象イベントフラグに対す
 7297 る通常操作1が許可されていない：clr_flgの場合）
 7298

7299 【機能】

7300

7301 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
 7302 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
 7303 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。
 7304 -----

7305 wai_flg イベントフラグ待ち [T]
 7306 pol_flg イベントフラグ待ち（ポーリング） [T]
 7307 twai_flg イベントフラグ待ち（タイムアウト付き） [T]
 7308

7309 【C言語API】
 7310 ER ercd = wai_flg(ID flgid, FLGPTN waiptn, MODE wfmode, FLGPTN *p_flgptn)
 7311 ER ercd = pol_flg(ID flgid, FLGPTN waiptn, MODE wfmode, FLGPTN *p_flgptn)
 7312 ER ercd = twai_flg(ID flgid, FLGPTN waiptn,
 7313 MODE wfmode, FLGPTN *p_flgptn, TMO tmout)
 7314

7315 【パラメータ】
 7316 ID flgid 対象イベントフラグのID番号
 7317 FLGPTN waiptn 待ちビットパターン
 7318 MODE wfmode 待ちモード
 7319 FLGPTN * p_flgptn 待ち解除時のビットパターンを入れるメモリ領
 7320 域へのポインタ
 7321 TMO tmout タイムアウト時間（twai_flgの場合）
 7322

7323 【リターンパラメータ】
 7324 ER ercd 正常終了（E_OK）またはエラーコード
 7325 FLGPTN flgptn 待ち解除時のビットパターン
 7326

7327 【エラーコード】
 7328 E_CTX コンテキストエラー（非タスクコンテキストからの呼出
 7329 し、CPUロック状態からの呼出し、ディスパッチ保留状態
 7330 からの呼出し：pol_flgを除く）
 7331 E_NOSPT 未サポート機能（制約タスクからの呼出し：pol_flgを除
 7332 く）
 7333 E_ID 不正ID番号（flgidが不正）
 7334 E_PAR パラメータエラー（waiptn, wfmodeが不正, tmoutが不正：
 7335 twai_flgの場合）
 7336 E_NOEXS [D] オブジェクト未登録（対象イベントフラグが未登録）
 7337 E_OACV [P] オブジェクトアクセス違反（対象イベントフラグに対す
 7338 る通常操作2が許可されていない）
 7339 E_MACV [P] メモリアクセス違反（p_flgptnが指すメモリ領域への書
 7340 込みアクセスが許可されていない）
 7341 E_ILUSE サービスコール不正使用（TA_WMUL属性でないイベントフ
 7342 ラグで待ちタスクあり）
 7343 E_TMOUT ポーリング失敗またはタイムアウト（wai_flgを除く）
 7344 E_RLWAI 待ち禁止状態または待ち状態の強制解除（pol_flgを除く）
 7345 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）
 7346

7347 【機能】
 7348
 7349 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
 7350 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。

7351
7352 対象イベントフラグが、waitpnとwfmodeで指定した待ち解除の条件を満たして
7353 いる場合には、対象イベントフラグのビットパターンの現在値がflgptnに返さ
7354 れる。対象イベントフラグがTA_CLR属性である場合には、対象イベントフラグ
7355 のビットパターンが0にクリアされる。

7356
7357 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
7358 態となり、対象イベントフラグの待ち行列につながる。

7359 -----

7360 ini_flg イベントフラグの再初期化 [T]

7361
7362 【C言語API】

7363 ER ercd = ini_flg(ID flgid)

7364
7365 【パラメータ】

7366 ID flgid 対象イベントフラグのID番号

7367
7368 【リターンパラメータ】

7369 ER ercd 正常終了 (E_OK) またはエラーコード

7370
7371 【エラーコード】

7372 E_CTX コンテキストエラー (非タスクコンテキストからの呼出
7373 し、CPUロック状態からの呼出し)

7374 E_ID 不正ID番号 (flgidが不正)

7375 E_NOEXS [D] オブジェクト未登録 (対象イベントフラグが未登録)

7376 E_OACV [P] オブジェクトアクセス違反 (対象イベントフラグに対す
7377 る管理操作が許可されていない)

7378
7379 【機能】

7380
7381 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
7382 的な振舞いは以下の通り。

7383
7384 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。
7385 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
7386 のタスクから順に待ち解除される。待ち解除されたタスクには、待ち状態となっ
7387 たサービスコールからE_DLTエラーが返る。

7388
7389 【使用上の注意】

7390
7391 ini_flgにより複数のタスクが待ち解除される場合、サービスコールの処理時間
7392 およびカーネル内での割込み禁止時間が、待ち解除されるタスクの数に比例し
7393 て長くなる。特に、多くのタスクが待ち解除される場合、カーネル内での割込
7394 み禁止時間が長くなるため、注意が必要である。

7395
7396 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
7397 は、アプリケーションの責任である。

7398
7399 【μITRON4.0仕様との関係】

7400

7401 μ ITRON4.0仕様に定義されていないサービスコールである.

7402 -----

7403 ref_flg イベントフラグの状態参照 [T]

7404

7405 **【C言語API】**

7406 ER ercd = ref_flg(ID flgid, T_RFLG *pk_rflg)

7407

7408 **【パラメータ】**

7409 ID flgid 対象イベントフラグのID番号

7410 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

7411

7412 **【リターンパラメータ】**

7413 ER ercd 正常終了 (E_OK) またはエラーコード

7414

7415 * イベントフラグの現在状態 (パケットの内容)

7416 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

7417 uint_t flgptn イベントフラグのビットパターン

7418

7419 **【エラーコード】**

7420 E_CTX コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)

7421 E_ID 不正ID番号 (flgidが不正)

7422 E_NOEXS [D] オブジェクト未登録 (対象イベントフラグが未登録)

7423 E_OACV [P] オブジェクトアクセス違反 (対象イベントフラグに対する参照操作が許可されていない)

7424 E_MACV [P] メモリアクセス違反 (pk_rflgが指すメモリ領域への書き込みアクセスが許可されていない)

7425

7426 **【機能】**

7427

7428 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.

7429

7430 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.

7431

7432 **【使用上の注意】**

7433

7434 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.

7435 -----

7436

7437 **4.4.3 データキュー**

7438

7439 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

7440

7451 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
7452 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
7453 号によって識別する。

7454
7455 各データキューが持つ情報は次の通り。

- 7456
- 7457 • データキュー属性
- 7458 • データキュー管理領域
- 7459 • 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）
- 7460 • 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）
- 7461 • アクセス許可ベクタ（保護機能対応カーネルの場合）
- 7462 • 属する保護ドメイン（保護機能対応カーネルの場合）
- 7463 • 属するクラス（マルチプロセッサ対応カーネルの場合）

7464
7465 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
7466 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
7467 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
7468 ズを0とすることができる。

7469
7470 保護機能対応カーネルにおいて、データキュー管理領域は、カーネルの用いる
7471 オブジェクト管理領域として扱われる。

7472
7473 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
7474 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
7475 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
7476 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
7477 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。

7478
7479 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。

7480
7481 TA_TPRI 0x01U 送信待ち行列をタスクの優先度順にする

7482
7483 TA_TPRIを指定しない場合、送信待ち行列はFIFO順になる。受信待ち行列は、
7484 FIFO順に固定されている。

7485
7486 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。

7487
7488 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
7489 ネルでは、静的APIによって登録されたデータキューの数
7490 に一致）

7491
7492 【 μ ITRON4.0仕様との関係】

7493
7494 TNUM_DTQIDは、 μ ITRON4.0仕様に規定されていないカーネル構成マクロである。

7495 -----

7496 CRE_DTQ データキューの生成 [S]
7497 acre_dtq データキューの生成 [TD]

7498
7499 【静的API】

7500 CRE_DTQ(ID dtqid, { ATR dtqatr, uint_t dtqcnt, void *dtqmb })

7501

7502 **【C言語API】**

7503 ER_ID dtqid = acre_dtq(const T_CDTQ *pk_cdtq)

7504

7505 **【パラメータ】**

7506 ID dtqid 生成するデータキューのID番号 (CRE_DTQの場合)

7507 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

7508 ポインタ (静的APIを除く)

7509

7510 *データキューの生成情報 (パケットの内容)

7511 ATR dtqatr データキュー属性

7512 uint_t dtqcnt データキュー管理領域に格納できるデータ数

7513 void * dtqmb データキュー管理領域の先頭番地

7514

7515 **【リターンパラメータ】**

7516 ER_ID dtqid 生成されたデータキューのID番号 (正の値) ま

7517 たはエラーコード

7518

7519 **【エラーコード】**

7520 E_CTX [s] コンテキストエラー (非タスクコンテキストからの呼出

7521 し, CPUロック状態からの呼出し)

7522 E_RSATR 予約属性 (dtqatrが不正または使用できない, 属する保

7523 護ドメインがクラスが不正)

7524 E_NOSPT 未サポート機能 (dtqmbがサポートされていない値)

7525 E_PAR パラメータエラー (dtqmbが不正)

7526 E_OACV [sP] オブジェクトアクセス違反 (システム状態に対する管理

7527 操作が許可されていない)

7528 E_MACV [sP] メモリアクセス違反 (pk_cdtqが指すメモリ領域への読出

7529 しアクセスが許可されていない)

7530 E_NOID [sD] ID番号不足 (割り付けられるデータキューIDがない)

7531 E_NOMEM メモリ不足 (データキュー管理領域が確保できない)

7532 E_OBJ オブジェクト状態エラー (dtqidで指定したデータキュー

7533 が登録済み: CRE_DTQの場合, その他の条件については機

7534 能の項を参照すること)

7535

7536 **【機能】**

7537

7538 各パラメータで指定したデータキュー生成情報に従って, データキューを生成

7539 する. dtqcntとdtqmbからデータキュー管理領域が設定され, 格納されているデー

7540 タがない状態に初期化される. また, 送信待ち行列と受信待ち行列は, 空の状

7541 態に初期化される.

7542

7543 静的APIにおいては, dtqidはオブジェクト識別名, dtqcntは整数定数式パラメー

7544 タ, dtqmb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

7545 メモリ不足 (E_NOMEM) エラーを検出することができない.

7546

7547 dtqmbをNULLとした場合, dtqcntで指定した数のデータを格納できるデータキュー

7548 管理領域を, コンフィギュレータまたはカーネルが確保する.

7549

7550 [dtqmbにNULL以外を指定した場合]

7551
7552 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
7553 域は、アプリケーションで確保しておく必要がある。データキュー管理領域を
7554 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。
7555
7556 TSZ_DTQMB(dtqcnt) dtqcntで指定した数のデータを格納できるデータ
7557 キュー管理領域のサイズ（バイト数）
7558 TCNT_DTQMB(dtqcnt) dtqcntで指定した数のデータを格納できるデータ
7559 キュー管理領域を確保するために必要なMB_T型の配
7560 列の要素数
7561
7562 これらを用いてデータキュー管理領域を確保する方法は次の通り。
7563
7564 MB_T <データキュー管理領域の変数名>[TCNT_DTQMB(dtqcnt)];
7565
7566 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。
7567
7568 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
7569 した時には、E_PARエラーとなる。また、保護機能対応カーネルにおいて、
7570 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
7571 含まれない場合、E_OBJエラーとなる。
7572
7573 【TOPPERS/ASPカーネルにおける規定】
7574
7575 ASPカーネルでは、CRE_DTQのみをサポートする。また、dtqmbにはNULLのみを指
7576 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。た
7577 だし、動的生成機能拡張パッケージでは、acre_dtqもサポートする。acre_dtq
7578 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。
7579
7580 【TOPPERS/FMPカーネルにおける規定】
7581
7582 FMPカーネルでは、CRE_DTQのみをサポートする。また、dtqmbにはNULLのみを指
7583 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。
7584
7585 【TOPPERS/HRP2カーネルにおける規定】
7586
7587 HRP2カーネルでは、CRE_DTQのみをサポートする。また、dtqmbにはNULLのみを
7588 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。
7589
7590 【μITRON4.0仕様との関係】
7591
7592 μITRON4.0/PX仕様にあわせて、データキュー生成情報の最後のパラメータを、
7593 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
7594 頭番地）に改名した。また、TSZ_DTQをTSZ_DTQMBに改名した。
7595
7596 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
7597 法を規定した。
7598 -----
7599 AID_DTQ 割付け可能なデータキューIDの数の指定 [SD]
7600

7601 **【静的API】**

7602 AID_DTQ(uint_t noddq)

7603

7604 **【パラメータ】**

7605 uint_t noddq 割付け可能なデータキューIDの数

7606

7607 **【エラーコード】**

7608 E_RSATR 予約属性（属する保護ドメインまたはクラスが不正）

7609

7610 **【機能】**

7611

7612 noddq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
7613 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。

7614

7615 noddqは整数定数式パラメータである。

7616

7617 SAC_DTQ データキューのアクセス許可ベクタの設定 [SP]

7618 sac_dtt データキューのアクセス許可ベクタの設定 [TPD]

7619

7620 **【静的API】**7621 SAC_DTQ(ID dtqid, { ACPTN acptn1, ACPTN acptn2,
7622 ACPTN acptn3, ACPTN acptn4 })

7623

7624 **【C言語API】**

7625 ER ercd = sac_dtt(ID dtqid, const ACVCT *p_acvct)

7626

7627 **【パラメータ】**

7628 ID dtqid 対象データキューのID番号

7629 ACVCT * p_acvct アクセス許可ベクタを入れたパケットへのポ
7630 インタ（静的APIを除く）

7631

7632 *アクセス許可ベクタ（パケットの内容）

7633 ACPTN acptn1 通常操作1のアクセス許可パターン

7634 ACPTN acptn2 通常操作2のアクセス許可パターン

7635 ACPTN acptn3 管理操作のアクセス許可パターン

7636 ACPTN acptn4 参照操作のアクセス許可パターン

7637

7638 **【リターンパラメータ】**

7639 ER ercd 正常終了（E_OK）またはエラーコード

7640

7641 **【エラーコード】**7642 E_CTX [s] コンテキストエラー（非タスクコンテキストからの呼出
7643 し、CPUロック状態からの呼出し）

7644 E_ID 不正ID番号（dtqidが不正）

7645 E_RSATR 予約属性（属する保護ドメインかクラスが不正：SAC_DTQ
7646 の場合）

7647 E_NOEXS [D] オブジェクト未登録（対象データキューが未登録）

7648 E_OACV [sP] オブジェクトアクセス違反（対象データキューに対する
7649 管理操作が許可されていない）

7650 E_MACV [sP] メモリアクセス違反（p_acvctが指すメモリ領域への読出

7651 しアクセスが許可されていない)
 7652 E_OBJ オブジェクト状態エラー (対象データキューは静的APIで
 7653 生成された: sac_dtqの場合, 対象データキューに対して
 7654 アクセス許可ベクタが設定済み: SAC_DTQの場合)
 7655
 7656 **【機能】**
 7657
 7658 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
 7659 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.
 7660
 7661 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
 7662 式パラメータである.
 7663
 7664 SAC_DTQは, 対象データキューが属する保護ドメインの囲みの中に記述しなけれ
 7665 ばならない. そうでない場合には, E_RSATRエラーとなる.
 7666
 7667 **【TOPPERS/ASPカーネルにおける規定】**
 7668
 7669 ASPカーネルでは, SAC_DTQ, sac_dtqをサポートしない.
 7670
 7671 **【TOPPERS/FMPカーネルにおける規定】**
 7672
 7673 FMPカーネルでは, SAC_DTQ, sac_dtqをサポートしない.
 7674
 7675 **【TOPPERS/HRP2カーネルにおける規定】**
 7676
 7677 HRP2カーネルでは, SAC_DTQのみをサポートする.
 7678
 7679 -----
 7680 del_dtq データキューの削除 [TD]
 7681
 7682 **【C言語API】**
 7683 ER ercd = del_dtq(ID dtqid)
 7684
 7685 **【パラメータ】**
 7686 ID dtqid 対象データキューのID番号
 7687
 7688 **【リターンパラメータ】**
 7689 ER ercd 正常終了 (E_OK) またはエラーコード
 7690
 7691 **【エラーコード】**
 7692 E_CTX コンテキストエラー (非タスクコンテキストからの呼出
 7693 E_ID 不正ID番号 (dtqidが不正)
 7694 E_NOEXS [D] オブジェクト未登録 (対象データキューが未登録)
 7695 E_OACV [P] オブジェクトアクセス違反 (対象データキューに対する
 7696 管理操作が許可されていない)
 7697 E_OBJ オブジェクト状態エラー (対象データキューは静的APIで
 7698 生成された)
 7699
 7700 **【機能】**

7701
7702 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
7703 いは以下の通り．
7704
7705 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
7706 される．また，対象データキューの送信待ち行列と受信待ち行列につながれた
7707 タスクは，それぞれの待ち行列の先頭のタスクから順に待ち解除される．待ち
7708 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
7709 る．
7710
7711 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
7712 た場合は，そのメモリ領域が解放される．
7713
7714 **【補足説明】**
7715
7716 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，
7717 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
7718 い．
7719
7720 **【使用上の注意】**
7721
7722 del_dtqにより複数のタスクが待ち解除される場合，サービスコールの処理時間
7723 およびカーネル内での割込み禁止時間が，待ち解除されるタスクの数に比例し
7724 て長くなる．特に，多くのタスクが待ち解除される場合，カーネル内での割込
7725 み禁止時間が長くなるため，注意が必要である．
7726
7727 **【TOPPERS/ASPカーネルにおける規定】**
7728
7729 ASPカーネルでは，del_dtqをサポートしない．ただし，動的生成機能拡張パッ
7730 ケージでは，del_dtqをサポートする．
7731
7732 **【TOPPERS/FMPカーネルにおける規定】**
7733
7734 FMPカーネルでは，del_dtqをサポートしない．
7735
7736 **【TOPPERS/HRP2カーネルにおける規定】**
7737
7738 HRP2カーネルでは，del_dtqをサポートしない．
7739 -----
7740 snd_dtq データキューへの送信 [T]
7741 psnd_dtq データキューへの送信（ポーリング） [T]
7742 ipsnd_dtq データキューへの送信（ポーリング） [I]
7743 tsnd_dtq データキューへの送信（タイムアウト付き） [T]
7744
7745 **【C言語API】**
7746 ER ercd = snd_dtq(ID dtqid, intptr_t data)
7747 ER ercd = psnd_dtq(ID dtqid, intptr_t data)
7748 ER ercd = ipsnd_dtq(ID dtqid, intptr_t data)
7749 ER ercd = tsnd_dtq(ID dtqid, intptr_t data, TMO tmout)
7750

7751 **【パラメータ】**

7752 ID dtqid 対象データキューのID番号

7753 intptr_t data 送信データ

7754 TMO tmout タイムアウト時間 (tsnd_dtqの場合)

7755

7756 **【リターンパラメータ】**

7757 ER ercd 正常終了 (E_OK) またはエラーコード

7758

7759 **【エラーコード】**

7760 E_CTX コンテキストエラー (非タスクコンテキストからの呼出し: ipsnd_dtqを除く, タスクコンテキストからの呼出し: ipsnd_dtqの場合, CPUロック状態からの呼出し, ディスパッチ保留状態からの呼出し: snd_dtqとtsnd_dtqの場合)

7761

7762

7763

7764 E_NOSPT 未サポート機能 (制約タスクからの呼出し: snd_dtqとtsnd_dtqの場合)

7765

7766 E_ID 不正ID番号 (dtqidが不正)

7767 E_PAR パラメータエラー (tmoutが不正: tsnd_dtqの場合)

7768 E_NOEXS [D] オブジェクト未登録 (対象データキューが未登録)

7769 E_OACV [P] オブジェクトアクセス違反 (対象データキューに対する通常操作1が許可されていない: ipsnd_dtqを除く)

7770

7771 E_TMOUT ポーリング失敗またはタイムアウト (snd_dtqを除く)

7772 E_RLWAI 待ち禁止状態または待ち状態の強制解除 (snd_dtqとtsnd_dtqの場合)

7773

7774 E_DLT 待ちオブジェクトの削除または再初期化 (snd_dtqとtsnd_dtqの場合)

7775

7776

7777 **【機能】**

7778

7779 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.

7780

7781

7782 対象データキューの受信待ち行列にタスクが存在する場合には, 受信待ち行列の先頭のタスクが, dataで指定したデータを受信し, 待ち解除される. 待ち解除されたタスクには, 待ち状態となったサービスコールからE_OKが返る.

7783

7784

7785

7786 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.

7787

7788

7789

7790 対象データキューの受信待ち行列にタスクが存在せず, データキュー管理領域にデータを格納するスペースがない場合には, 自タスクはデータキューへの送信待ち状態となり, 対象データキューの送信待ち行列につながる.

7791

7792

7793 -----

7794 fsnd_dtq データキューへの強制送信 [T]

7795 ifsnd_dtq データキューへの強制送信 [I]

7796

7797 **【C言語API】**

7798 ER ercd = fsnd_dtq(ID dtqid, intptr_t data)

7799 ER ercd = ifsnd_dtq(ID dtqid, intptr_t data)

7800

7801 **【パラメータ】**

7802 ID dtqid 対象データキューのID番号
 7803 intptr_t data 送信データ

7804

7805 **【リターンパラメータ】**

7806 ER ercd 正常終了 (E_OK) またはエラーコード

7807

7808 **【エラーコード】**

7809 E_CTX コンテキストエラー (非タスクコンテキストからの呼出し: fsnd_dtq の場合, タスクコンテキストからの呼出し: ifsnd_dtq の場合, CPU ロック状態からの呼出し)
 7810
 7811
 7812 E_ID 不正ID番号 (dtqid が不正)
 7813 E_NOEXS [D] オブジェクト未登録 (対象データキューが未登録)
 7814 E_OACV [P] オブジェクトアクセス違反 (対象データキューに対する
 7815 通常操作1が許可されていない: fsnd_dtq の場合)
 7816 E_ILUSE サービスコール不正使用 (対象データキューのデータキュー
 7817 管理領域のサイズが0)

7818

7819 **【機能】**

7820

7821 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.

7822

7823

7824 対象データキューの受信待ち行列にタスクが存在する場合には, 受信待ち行列の先頭のタスクが, data で指定したデータを受信し, 待ち解除される. 待ち解除されたタスクには, 待ち状態となったサービスコールから E_OK が返る.

7826

7827
 7828 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.

7830

7831
 7832 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.

7836

7837 対象データキューのデータキュー管理領域のサイズが0の場合には, E_ILUSE エラーとなる.

7838

7839 -----

7840 rcv_dtq データキューからの受信 [T]
 7841 prev_dtq データキューからの受信 (ポーリング) [T]
 7842 trcv_dtq データキューからの受信 (タイムアウト付き) [T]

7843

7844 **【C言語API】**

7845 ER ercd = rcv_dtq(ID dtqid, intptr_t *p_data)
 7846 ER ercd = prev_dtq(ID dtqid, intptr_t *p_data)
 7847 ER ercd = trcv_dtq(ID dtqid, intptr_t *p_data, TMO tmout)

7848

7849 **【パラメータ】**

7850 ID dtqid 対象データキューのID番号

7851 intptr_t * p_data 受信データを入れるメモリ領域へのポインタ
7852 TMO tmout タイムアウト時間 (trcv_dtqの場合)
7853
7854 **【リターンパラメータ】**
7855 ER ercd 正常終了 (E_OK) またはエラーコード
7856 intptr_t data 受信データ
7857
7858 **【エラーコード】**
7859 E_CTX コンテキストエラー (非タスクコンテキストからの呼出し、CPUロック状態からの呼出し、ディスパッチ保留状態からの呼出し：prcv_dtqを除く)
7860
7861 E_NOSPT 未サポート機能 (制約タスクからの呼出し：prcv_dtqを除く)
7862
7863 E_ID 不正ID番号 (dtqidが不正)
7864 E_PAR パラメータエラー (tmoutが不正：trcv_dtqの場合)
7865 E_NOEXS [D] オブジェクト未登録 (対象データキューが未登録)
7866 E_OACV [P] オブジェクトアクセス違反 (対象データキューに対する
7867 通常操作2が許可されていない)
7868 E_MACV [P] メモリアクセス違反 (p_dataが指すメモリ領域への書込
7869 みアクセスが許可されていない)
7870 E_TMOUT ポーリング失敗またはタイムアウト (rcv_dtqを除く)
7871 E_RLWAI 待ち禁止状態または待ち状態の強制解除 (prcv_dtqを除く)
7872 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)
7873
7874
7875 **【機能】**
7876
7877 dtqidで指定したデータキュー (対象データキュー) からデータを受信する。受信したデータは、p_dataで指定したメモリ領域に返される。具体的な振舞いは以下の通り。
7878
7879
7880
7881 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、
7882 データキュー管理領域の先頭に格納されたデータが取り出され、p_dataで指定したメモリ領域に返される。また、送信待ち行列にタスクが存在する場合には、
7883 送信待ち行列の先頭のタスクの送信データが、FIFO順でデータキュー管理領域に格納され、そのタスクは待ち解除される。待ち解除されたタスクには、待ち状態となったサービスコールからE_OKが返る。
7884
7885
7886
7887
7888 対象データキューのデータキュー管理領域にデータが格納されておらず、送信待ち行列にタスクが存在する場合には、送信待ち行列の先頭のタスクの送信データが、p_dataで指定したメモリ領域に返される。送信待ち行列の先頭のタスクは、待ち解除される。待ち解除されたタスクには、待ち状態となったサービスコールからE_OKが返る。
7889
7890
7891
7892
7893
7894 対象データキューのデータキュー管理領域にデータが格納されておらず、送信待ち行列にタスクが存在しない場合には、自タスクはデータキューからの受信待ち状態となり、対象データキューの受信待ち行列につながる。
7895
7896
7897 -----
7898 ini_dtq データキューの再初期化 [T]
7899
7900 **【C言語API】**

7901 ER ercd = ini_dtq(ID dtqid)

7902

7903 **【パラメータ】**

7904 ID dtqid 対象データキューのID番号

7905

7906 **【リターンパラメータ】**

7907 ER ercd 正常終了 (E_OK) またはエラーコード

7908

7909 **【エラーコード】**

7910 E_CTX コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)

7911 E_ID 不正ID番号 (dtqidが不正)

7912 E_NOEXS [D] オブジェクト未登録 (対象データキューが未登録)

7913 E_OACV [P] オブジェクトアクセス違反 (対象データキューに対する管理操作が許可されていない)

7914

7915

7916

7917 **【機能】**

7918

7919 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

7920 振舞いは以下の通り.

7921

7922 対象データキューのデータキュー管理領域は, 格納されているデータがない状態に初期化される. また, 対象データキューの送信待ち行列と受信待ち行列につながれたタスクは, それぞれの待ち行列の先頭のタスクから順に待ち解除される. 待ち解除されたタスクには, 待ち状態となったサービスコールからE_DLTエラーが返る.

7926

7927

7928 **【補足説明】**

7929

7930 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,

7931 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.

7932

7933

7934 **【使用上の注意】**

7935

7936 ini_dtqにより複数のタスクが待ち解除される場合, サービスコールの処理時間およびカーネル内での割込み禁止時間が, 待ち解除されるタスクの数に比例して長くなる. 特に, 多くのタスクが待ち解除される場合, カーネル内での割込み禁止時間が長くなるため, 注意が必要である.

7939

7940 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.

7941

7942

7943

7944 **【μ ITRON4.0仕様との関係】**

7945

7946 μ ITRON4.0仕様に定義されていないサービスコールである.

7947

7948 ref_dtq データキューの状態参照 [T]

7949

7950 **【C言語API】**

```

7951     ER ercd = ref_dtq(ID dtqid, T_RDTQ *pk_rdtq)
7952
7953     【パラメータ】
7954         ID          dtqid      対象データキューのID番号
7955         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
7956                                 ポインタ
7957
7958     【リターンパラメータ】
7959         ER          ercd      正常終了 (E_OK) またはエラーコード
7960
7961     * データキューの現在状態 (パケットの内容)
7962         ID          stskid     データキューの送信待ち行列の先頭のタスクの
7963                                 ID番号
7964         ID          rtskid     データキューの受信待ち行列の先頭のタスクの
7965                                 ID番号
7966         uint_t      sdtqcnt    データキュー管理領域に格納されているデータ
7967                                 の数
7968
7969     【エラーコード】
7970         E_CTX       コンテキストエラー (非タスクコンテキストからの呼出
7971                                 し, CPUロック状態からの呼出し)
7972         E_ID        不正ID番号 (dtqidが不正)
7973         E_NOEXS [D] オブジェクト未登録 (対象データキューが未登録)
7974         E_OACV [P]  オブジェクトアクセス違反 (対象データキューに対する
7975                                 参照操作が許可されていない)
7976         E_MACV [P]  メモリアクセス違反 (pk_rdtqが指すメモリ領域への書込
7977                                 みアクセスが許可されていない)
7978
7979     【機能】
7980
7981     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
7982     照した現在状態は, pk_rdtqで指定したパケットに返される.
7983
7984     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
7985     TSK_NONE (=0) が返る. また, 受信待ち行列にタスクが存在しない場合,
7986     rtskidにはTSK_NONE (=0) が返る.
7987
7988     【使用上の注意】
7989
7990     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
7991     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
7992     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
7993     状態が変化している可能性があるためである.
7994     -----
7995
7996     4.4.4 優先度データキュー
7997
7998     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
7999     順で送受信するための同期・通信カーネルオブジェクトである. より大きいサ
8000     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

```


8001 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
8002 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。

8003

8004 各優先度データキューが持つ情報は次の通り。

8005

8006 ・優先度データキュー属性

8007 ・優先度データキュー管理領域

8008 ・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）

8009 ・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）

8010 ・送信できるデータ優先度の最大値

8011 ・アクセス許可ベクタ（保護機能対応カーネルの場合）

8012 ・属する保護ドメイン（保護機能対応カーネルの場合）

8013 ・属するクラス（マルチプロセッサ対応カーネルの場合）

8014

8015 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、
8016 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
8017 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、
8018 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。

8019

8020 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
8021 用いるオブジェクト管理領域として扱われる。

8022

8023 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つて
8024 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
8025 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
8026 タキューからデータが受信できるまで待つている状態（優先度データキューか
8027 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
8028 キューである。

8029

8030 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。

8031

8032 TA_TPRI 0x01U 送信待ち行列をタスクの優先度順にする

8033

8034 TA_TPRIを指定しない場合、送信待ち行列はFIFO順になる。受信待ち行列は、
8035 FIFO順に固定されている。

8036

8037 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。

8038

8039 TMIN_DPRI データ優先度の最小値（=1）

8040 TMAX_DPRI データ優先度の最大値

8041

8042 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
8043 いカーネルでは、静的APIによって登録された優先度デー
8044 タキューの数に一致）

8045

8046 【TOPPERS/ASPカーネルにおける規定】

8047

8048 ASP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。
8049 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。

8050

8051 【TOPPERS/FMPカーネルにおける規定】

8052

8053 FMP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。

8054

8055 【TOPPERS/HRP2カーネルにおける規定】

8056

8057 HRP2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。

8058

8059 【使用上の注意】

8060

8061 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
8062 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
8063 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、
8064 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
8065 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
8066 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
8067 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
8068 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。

8069

8070 【μ ITRON4.0仕様との関係】

8071

8072 μ ITRON4.0仕様に規定されていない機能である。

8073

8074 CRE_PDQ 優先度データキューの生成 [S]

8075 acre_pdq 優先度データキューの生成 [TD]

8076

8077 【静的API】

8078 CRE_PDQ(ID pdqid, { ATR pdqatr, uint_t pdqcnt, PRI maxdpri, void *pdqmb })

8079

8080 【C言語API】

8081 ER_ID pdqid = acre_pdq(const T_CPDQ *pk_cpdq)

8082

8083 【パラメータ】

8084 ID pdqid 生成する優先度データキューのID番号（CRE_PDQ
8085 の場合）

8086 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
8087 トへのポインタ（静的APIを除く）

8088

8089 * 優先度データキューの生成情報（パケットの内容）

8090 ATR pdqatr 優先度データキュー属性

8091 uint_t pdqcnt 優先度データキュー管理領域に格納できるデー
8092 タ数

8093 PRI maxdpri 優先度データキューに送信できるデータ優先度
8094 の最大値

8095 void * pdqmb 優先度データキュー管理領域の先頭番地

8096

8097 【リターンパラメータ】

8098 ER_ID pdqid 生成された優先度データキューのID番号（正の
8099 値）またはエラーコード

8100

8101 **【エラーコード】**

8102 E_CTX [s] コンテキストエラー（非タスクコンテキストからの呼出し、CPUロック状態からの呼出し）

8103

8104 E_RSATR 予約属性（pdqatrが不正または使用できない、属する保護ドメインかクラスが不正）

8105

8106 E_NOSPT 未サポート機能（pdqmbがサポートされていない値）

8107 E_PAR パラメータエラー（pdqmb, maxdpriが不正）

8108 E_OACV [sP] オブジェクトアクセス違反（システム状態に対する管理操作が許可されていない）

8109

8110 E_MACV [sP] メモリアクセス違反（pk_cpdqが指すメモリ領域への読出しアクセスが許可されていない）

8111

8112 E_NOID [sD] ID番号不足（割り付けられる優先度データキューIDがない）

8113 E_NOMEM メモリ不足（優先度データキュー管理領域が確保できない）

8114 E_OBJ オブジェクト状態エラー（pdqidで指定した優先度データキューが登録済み：CRE_PDQの場合、その他の条件については機能の項を参照すること）

8115

8116

8117

8118 **【機能】**

8119

8120 各パラメータで指定した優先度データキュー生成情報に従って、優先度データキューを生成する。pdqcntとpdqmbから優先度データキュー管理領域が設定され、格納されているデータがない状態に初期化される。また、送信待ち行列と受信待ち行列は、空の状態に初期化される。

8121

8122

8123

8124

8125 静的APIにおいては、pdqidはオブジェクト識別名、pdqcntとmaxdpriは整数定数式パラメータ、pdqmbは一般定数式パラメータである。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。

8126

8127

8128

8129 pdqmbをNULLとした場合、pdqcntで指定した数のデータを格納できる優先度データキュー管理領域を、コンフィギュレータまたはカーネルが確保する。

8130

8131

8132 maxdpriは、TMIN_DPRI以上、TMAX_DPRI以下でなければならない。

8133

8134 〔pdqmbにNULL以外を指定した場合〕

8135

8136 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。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。

8137

8138

8139

8140 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）

8141

8142 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

8143

8144

8145

8146 これらを用いて優先度データキュー管理領域を確保する方法は次の通り。

8147

8148 MB_T <優先度データキュー管理領域の変数名>[TCNT_PDQMB(pdqcnt)];

8149

8150 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。

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。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。

【TOPPERS/ASPカーネルにおける規定】

8159 ASPカーネルでは、CRE_PDQのみをサポートする。また、pdqmbにはNULLのみを指
8160 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。た
8161 だし、動的生成機能拡張パッケージでは、acre_pdqもサポートする。acre_pdq
8162 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。

【TOPPERS/FMPカーネルにおける規定】

8166 FMPカーネルでは、CRE_PDQのみをサポートする。また、pdqmbにはNULLのみを渡
8167 すことができる。NULL以外を指定した場合には、E_NOSPTエラーとなる。

【TOPPERS/HRP2カーネルにおける規定】

8171 HRP2カーネルでは、CRE_PDQのみをサポートする。また、pdqmbにはNULLのみを
8172 渡すことができる。NULL以外を指定した場合には、E NOSPTエラーとなる。

8174 AID PDQ 割付け可能な優先度データキューIDの数の指定 [SD]

【静的API】

```
8177      AID_PDQ(uint_t nopdq)
```

【パラメータ】

```
8180      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】

```
8183      E RSATR      予約属性 (属する保護ドメインまたはクラスが不正)
```

【機能】

8187 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
8188 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。

8190 `nopdq`は整数定数式パラメータである.

8192	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]
8193	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]

【靜的API】

```

8196     SAC_PDQ(ID pdqid, { ACPTN acptn1, ACPTN acptn2,
8197                          ACPTN acptn3, ACPTN acptn4 })

```

【C言語API】

```
8200 ER ercd = sac pdq(ID pdqid, const ACVCT *p acvct)
```

8201

8202

8203

8204

8205

8206

8207

8208

8209

8210

8211

8212

8213

8214

8215

8216

8217

8218

8219

8220

8221

8222

8223

8224

8225

8226

8227

8228

8229

8230

8231

8232

8233

8234

8235

8236

8237

8238

8239

8240

8241

8242

8243

8244

8245

8246

8247

8248

8249

8250

【パラメータ】

ID

pdqid

対象優先度データキューのID番号

ACVCT *

p_acvct

アクセス許可ベクタを入れたパケットへのポインタ（静的APIを除く）

*アクセス許可ベクタ（パケットの内容）

ACPTN

acptn1

通常操作1のアクセス許可パターン

ACPTN

acptn2

通常操作2のアクセス許可パターン

ACPTN

acptn3

管理操作のアクセス許可パターン

ACPTN

acptn4

参照操作のアクセス許可パターン

【リターンパラメータ】

ER

ercd

正常終了（E_OK）またはエラーコード

【エラーコード】

E_CTX [s]

コンテキストエラー（非タスクコンテキストからの呼出し、CPUロック状態からの呼出し）

E_ID

不正ID番号（pdqidが不正）

E_RSATR

予約属性（属する保護ドメインかクラスが不正：SAC_PDQの場合）

E_NOEXS [D]

オブジェクト未登録（対象優先度データキューが未登録）

E_OACV [sP]

オブジェクトアクセス違反（対象優先度データキューに対する管理操作が許可されていない）

E_MACV [sP]

メモリアクセス違反（p_acvctが指すメモリ領域への読出しアクセスが許可されていない）

E_OBJ

オブジェクト状態エラー（対象優先度データキューは静的APIで生成された：sac_pdqの場合、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：SAC_PDQの場合）

【機能】

pdqidで指定した優先度データキュー（対象優先度データキュー）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。

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。

SAC_PDQは、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に記述しなければならない。そうでない場合には、E_RSATRエラーとなる。

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、SAC_PDQ、sac_pdqをサポートしない。

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、SAC_PDQ、sac_pdqをサポートしない。

8251 【TOPPERS/HRP2カーネルにおける規定】
8252
8253 HRP2カーネルでは、SAC_PDQのみをサポートする。
8254 -----
8255 del_pdq 優先度データキューの削除 [TD]
8256
8257 【C言語API】
8258 ER ercd = del_pdq(ID pdqid)
8259
8260 【パラメータ】
8261 ID pdqid 対象優先度データキューのID番号
8262
8263 【リターンパラメータ】
8264 ER ercd 正常終了 (E_OK) またはエラーコード
8265
8266 【エラーコード】
8267 E_CTX コンテキストエラー (非タスクコンテキストからの呼出し、CPUロック状態からの呼出し)
8268 E_ID 不正ID番号 (pdqidが不正)
8269 E_NOEXS [D] オブジェクト未登録 (対象優先度データキューが未登録)
8270 E_OACV [P] オブジェクトアクセス違反 (対象優先度データキューに対する管理操作が許可されていない)
8271 E_OBJ オブジェクト状態エラー (対象優先度データキューは静的APIで生成された)
8272
8273 E_OBJ オブジェクト状態エラー (対象優先度データキューは静的APIで生成された)
8274
8275
8276 【機能】
8277
8278 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。
8279 具体的な振舞いは以下の通り。
8280
8281 対象優先度データキューの登録が解除され、その優先度データキューIDが未使用の状態に戻される。また、対象優先度データキューの送信待ち行列と受信待ち行列につながれたタスクは、それぞれの待ち行列の先頭のタスクから順に待ち解除される。待ち解除されたタスクには、待ち状態となったサービスコールからE_DLTエラーが返る。
8282
8283 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。
8284
8285
8286
8287 【補足説明】
8288
8289 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。
8290
8291 【使用上の注意】
8292
8293 del_pdqにより複数のタスクが待ち解除される場合、サービスコールの処理時間およびカーネル内での割込み禁止時間が、待ち解除されるタスクの数に比例して長くなる。特に、多くのタスクが待ち解除される場合、カーネル内での割込

8301 み禁止時間が長くなるため、注意が必要である。

8302

8303 **【TOPPERS/ASPカーネルにおける規定】**

8304

8305 ASPカーネルでは、del_pdqをサポートしない。ただし、動的生成機能拡張パッ
8306 ケージでは、del_pdqをサポートする。

8307

8308 **【TOPPERS/FMPカーネルにおける規定】**

8309

8310 FMPカーネルでは、del_pdqをサポートしない。

8311

8312 **【TOPPERS/HRP2カーネルにおける規定】**

8313

8314 HRP2カーネルでは、del_pdqをサポートしない。

8315

8316 snd_pdq 優先度データキューへの送信 [T]
8317 psnd_pdq 優先度データキューへの送信（ポーリング） [T]
8318 ipsnd_pdq 優先度データキューへの送信（ポーリング） [I]
8319 tsnd_pdq 優先度データキューへの送信（タイムアウト付き） [T]

8320

8321 **【C言語API】**

8322 ER ercd = snd_pdq(ID pdqid, intptr_t data, PRI datapri)
8323 ER ercd = psnd_pdq(ID pdqid, intptr_t data, PRI datapri)
8324 ER ercd = ipsnd_pdq(ID pdqid, intptr_t data, PRI datapri)
8325 ER ercd = tsnd_pdq(ID pdqid, intptr_t data, PRI datapri, TMO tmout)

8326

8327 **【パラメータ】**

8328	ID	pdqid	対象優先度データキューのID番号
8329	intptr_t	data	送信データ
8330	PRI	datapri	送信データの優先度
8331	TMO	tmout	タイムアウト時間（tsnd_pdqの場合）

8332

8333 **【リターンパラメータ】**

8334	ER	ercd	正常終了（E_OK）またはエラーコード
------	----	------	---------------------

8335

8336 **【エラーコード】**

8337	E_CTX	コンテキストエラー（非タスクコンテキストからの呼出し：ipsnd_pdqを除く、タスクコンテキストからの呼出し：ipsnd_pdqの場合、CPUロック状態からの呼出し、ディスパッチ保留状態からの呼出し：snd_pdqとtsnd_pdqの場合）
8338		
8339		
8340		
8341	E_NOSPT	未サポート機能（制約タスクからの呼出し：snd_pdqとtsnd_pdqの場合）
8342		
8343	E_ID	不正ID番号（pdqidが不正）
8344	E_PAR	パラメータエラー（datapriが不正、tmoutが不正：tsnd_pdqのみ）
8345		
8346	E_NOEXS [D]	オブジェクト未登録（対象優先度データキューが未登録）
8347	E_OACV [P]	オブジェクトアクセス違反（対象優先度データキューに対する通常操作1が許可されていない：ipsnd_pdqを除く）
8348		
8349	E_TMOUT	ポーリング失敗またはタイムアウト（snd_pdqを除く）
8350	E_RLWAI	待ち禁止状態または待ち状態の強制解除（snd_pdqと

8351 tsnd_pdqの場合)
8352 E_DLT 待ちオブジェクトの削除または再初期化 (snd_pdqと
8353 tsnd_pdqの場合)
8354
8355 **【機能】**
8356
8357 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
8358 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
8359 の通り。
8360
8361 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
8362 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。
8363 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。
8364
8365 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
8366 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
8367 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
8368 に格納される。
8369
8370 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
8371 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
8372 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
8373 行列につながる。
8374
8375 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
8376 最大値以下でなければならない。
8377 -----
8378 rcv_pdq 優先度データキューからの受信 [T]
8379 prcv_pdq 優先度データキューからの受信 (ポーリング) [T]
8380 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]
8381
8382 **【C言語API】**
8383 ER ercd = rcv_pdq(ID pdqid, intptr_t *p_data, PRI *p_datapri)
8384 ER ercd = prcv_pdq(ID pdqid, intptr_t *p_data, PRI *p_datapri)
8385 ER ercd = trcv_pdq(ID pdqid, intptr_t *p_data, PRI *p_datapri, TMO tmout)
8386
8387 **【パラメータ】**
8388 ID pdqid 対象優先度データキューのID番号
8389 intptr_t * p_data 受信データを入れるメモリ領域へのポインタ
8390 PRI * p_datapri 受信データの優先度を入れるメモリ領域へのポ
8391 インタ
8392 TMO tmout タイムアウト時間 (trcv_pdqの場合)
8393
8394 **【リターンパラメータ】**
8395 ER ercd 正常終了 (E_OK) またはエラーコード
8396 intptr_t data 受信データ
8397 PRI datapri 受信データの優先度
8398
8399 **【エラーコード】**
8400 E_CTX コンテキストエラー (非タスクコンテキストからの呼出

8401 し、CPUロック状態からの呼出し、ディスパッチ保留状態
 8402 からの呼出し：prcv_pdqを除く)
 8403 E_NOSPT 未サポート機能（制約タスクからの呼出し：prcv_pdqを除
 8404 く）
 8405 E_ID 不正ID番号（pdqidが不正）
 8406 E_PAR パラメータエラー（tmoutが不正：trcv_pdqの場合）
 8407 E_NOEXS [D] オブジェクト未登録（対象優先度データキューが未登録）
 8408 E_OACV [P] オブジェクトアクセス違反（対象優先度データキューに
 8409 対する通常操作2が許可されていない）
 8410 E_MACV [P] メモリアクセス違反（p_dataまたはp_datapriが指すメモ
 8411 リ領域への書き込みアクセスが許可されていない）
 8412 E_TMOUT ポーリング失敗またはタイムアウト（rcv_pdqを除く）
 8413 E_RLWAI 待ち禁止状態または待ち状態の強制解除（prcv_pdqを除く）
 8414 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）
 8415

8416 【機能】

8417
 8418 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
 8419 受信する．受信したデータはp_dataで指定したメモリ領域に、その優先度は
 8420 p_datapriで指定したメモリ領域に返される．具体的な振舞いは以下の通り．
 8421

8422 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
 8423 いる場合には、優先度データキュー管理領域の先頭に格納されたデータが取り
 8424 出され、p_dataで指定したメモリ領域に返される．また、その優先度が
 8425 p_datapriで指定したメモリ領域に返される．さらに、送信待ち行列にタスクが
 8426 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
 8427 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
 8428 る．待ち解除されたタスクには、待ち状態となったサービスコールからE_OKが
 8429 返る．
 8430

8431 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
 8432 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
 8433 スクの送信データが、p_dataで指定したメモリ領域に返される．また、その優
 8434 先度がp_datapriで指定したメモリ領域に返される．送信待ち行列の先頭のタス
 8435 クは、待ち解除される．待ち解除されたタスクには、待ち状態となったサービ
 8436 スコールからE_OKが返る．
 8437

8438 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
 8439 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
 8440 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
 8441 につながる．
 8442

8443 ini_pdq 優先度データキューの再初期化 [T]
 8444

8445 【C言語API】

8446 ER ercd = ini_pdq(ID pdqid)
 8447

8448 【パラメータ】

8449 ID pdqid 対象優先度データキューのID番号
 8450

8451 **【リターンパラメータ】**

8452 ER ercd 正常終了 (E_OK) またはエラーコード

8453

8454 **【エラーコード】**

8455 E_CTX コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)

8456 E_ID 不正ID番号 (pdqidが不正)

8457 E_NOEXS [D] オブジェクト未登録 (対象優先度データキューが未登録)

8458 E_OACV [P] オブジェクトアクセス違反 (対象優先度データキューに対する管理操作が許可されていない)

8460

8461 **【機能】**

8462

8464 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.

8465

8466
8467 対象優先度データキューの優先度データキュー管理領域は, 格納されているデータがない状態に初期化される. また, 対象優先度データキューの送信待ち行列と受信待ち行列につながれたタスクは, それぞれの待ち行列の先頭のタスクから順に待ち解除される. 待ち解除されたタスクには, 待ち状態となったサービスコールからE_DLTエラーが返る.

8471

8472 **【補足説明】**

8473

8474
8475 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.

8477

8478 **【使用上の注意】**

8479

8480
8481 ini_pdqにより複数のタスクが待ち解除される場合, サービスコールの処理時間およびカーネル内での割込み禁止時間が, 待ち解除されるタスクの数に比例して長くなる. 特に, 多くのタスクが待ち解除される場合, カーネル内での割込み禁止時間が長くなるため, 注意が必要である.

8485

8486 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.

8487

8488 -----
8489 ref_pdq 優先度データキューの状態参照 [T]

8490

8491 **【C言語API】**

8492 ER ercd = ref_pdq(ID pdqid, T_RPDQ *pk_rpdq)

8493

8494 **【パラメータ】**

8495 ID pdqid 対象優先度データキューのID番号

8496 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

8497

8498 **【リターンパラメータ】**

8499 ER ercd 正常終了 (E_OK) またはエラーコード

8500

8501

8502 * 優先度データキューの現在状態 (パケットの内容)

8503 ID stskid 優先度データキューの送信待ち行列の先頭のタ
8504 スクのID番号8505 ID rtskid 優先度データキューの受信待ち行列の先頭のタ
8506 スクのID番号8507 uint_t spdqcnt 優先度データキュー管理領域に格納されている
8508 データの数

8509

8510 【エラーコード】

8511 E_CTX コンテキストエラー (非タスクコンテキストからの呼出
8512 し, CPUロック状態からの呼出し)

8513 E_ID 不正ID番号 (pdqidが不正)

8514 E_NOEXS [D] オブジェクト未登録 (対象優先度データキューが未登録)

8515 E_OACV [P] オブジェクトアクセス違反 (対象優先度データキューに
8516 対する参照操作が許可されていない)8517 E_MACV [P] メモリアクセス違反 (pk_rpdqが指すメモリ領域への書込
8518 みアクセスが許可されていない)

8519

8520 【機能】

8521

8522 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
8523 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.

8524

8525 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
8526 はTSK_NONE (=0) が返る. また, 受信待ち行列にタスクが存在しない場合,
8527 rtskidにはTSK_NONE (=0) が返る.

8528

8529 【使用上の注意】

8530

8531 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
8532 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
8533 した直後に割り込みが発生した場合, ref_pdqから戻ってきた時には対象優先度デー
8534 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.

8535

8536

8537 4.4.5 メールボックス

8538

8539 メールボックスは, 共有メモリ上に置いたメッセージを, FIFO順またはメッセー
8540 ジの優先度順で送受信するための同期・通信オブジェクトである. メールボッ
8541 クスは, メールボックスIDと呼ぶID番号によって識別する.

8542

8543 各メールボックスが持つ情報は次の通り.

8544

- 8545 • メールボックス属性
- 8546 • メッセージキュー
- 8547 • 待ち行列 (メールボックスからの受信待ち状態のタスクのキュー)
- 8548 • 送信できるメッセージ優先度の最大値
- 8549 • 優先度別のメッセージキューヘッダ領域
- 8550 • 属するクラス (マルチプロセッサ対応カーネルの場合)

8551
8552 メッセージキューは、メールボックスに送信されたメッセージを、FIFO順また
8553 はメッセージの優先度順につないでおくためのキューである。
8554
8555 待ち行列は、メールボックスからメッセージが受信できるまで待っている状態
8556 （メールボックスからの受信待ち状態）のタスクが、メッセージを受信できる
8557 順序でつながれているキューである。
8558
8559 メールボックス属性には、次の属性を指定することができる。
8560
8561 TA_TPRI 0x01U 待ち行列をタスクの優先度順にする
8562 TA_MPRI 0x02U メッセージキューをメッセージの優先度順にする
8563
8564 TA_TPRIを指定しない場合、待ち行列はFIFO順になる。TA_MPRIを指定しない場
8565 合、メッセージキューはFIFO順になる。
8566
8567 優先度別のメッセージキューヘッダ領域は、TA_MPRI属性のメールボックスに対
8568 して、メッセージキューを優先度別に設ける場合に使用する領域である。
8569
8570 カーネルは、メールボックスに送信されたメッセージをメッセージキューにつ
8571 なぐために、メッセージの先頭のメモリ領域を使用する。そのためアプリケー
8572 ションは、メールボックスに送信するメッセージの先頭に、カーネルが利用す
8573 るためのメッセージヘッダを置かなければならない。メッセージヘッダのデー
8574 タ型として、メールボックス属性にTA_MPRIが指定されているか否かにより、以
8575 下のいずれかを用いる。
8576
8577 T_MSG TA_MPRI属性でないメールボックス用のメッセージヘッダ
8578 T_MSG_PRI TA_MPRI属性のメールボックス用のメッセージヘッダ
8579
8580 メッセージヘッダの領域は、メッセージがメッセージキューにつながれている
8581 間（すなわち、メールボックスに送信してから受信するまでの間）、カーネル
8582 によって使用される。そのため、メッセージキューにつながれているメッセー
8583 ジのメッセージヘッダの領域をアプリケーションが書き換えた場合や、メッセー
8584 ジキューにつながれているメッセージを再度メールボックスに送信した場合の
8585 動作は保証されない。
8586
8587 TA_MPRI属性のメールボックスにメッセージを送信する場合、アプリケーション
8588 は、メッセージの優先度を、T_MSG_PRI型のメッセージヘッダ中のmsgpriフィー
8589 ルドに設定する。
8590
8591 保護機能対応カーネルでは、メールボックス機能はサポートしない。
8592
8593 メールボックス機能に関連するカーネル構成マクロは次の通り。
8594
8595 TMIN_MPRI メッセージ優先度の最小値（=1）
8596 TMAX_MPRI メッセージ優先度の最大値
8597
8598 TNUM_MBXID 登録できるメールボックスの数（動的生成対応でないカー
8599 ネルでは、静的APIによって登録されたメールボックスの
8600 数に一致）

8601
8602 **【補足説明】**
8603
8604 TOPPERS新世代カーネルの現時点の実装では、優先度別のメッセージキューヘッ
8605 ダ領域は用いていない。
8606
8607 **【使用上の注意】**
8608
8609 メールボックス機能は、 μ ITRON4.0仕様との互換性のために残した機能であり、
8610 保護機能対応カーネルではサポートしないため、使用することは推奨しない。
8611 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
8612 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
8613 する方法で置き換えることができる。
8614
8615 **【TOPPERS/ASPカーネルにおける規定】**
8616
8617 ASPカーネルでは、メールボックス機能をサポートする。メッセージ優先度の最
8618 大値（TMAX_MPRI）は16に固定されている。ただし、タスク優先度拡張パッケ
8619 ジでは、TMAX_MPRIを256に拡張する。
8620
8621 **【TOPPERS/FMPカーネルにおける規定】**
8622
8623 FMPカーネルでは、メールボックス機能をサポートする。メッセージ優先度の最
8624 大値（TMAX_MPRI）は16に固定されている。
8625
8626 **【TOPPERS/HRP2カーネルにおける規定】**
8627
8628 HRP2カーネルでは、メールボックス機能をサポートしない。
8629
8630 **【 μ ITRON4.0仕様との関係】**
8631
8632 TNUM_MBXIDは、 μ ITRON4.0仕様に規定されていないカーネル構成マクロである。
8633 -----
8634 CRE_MBX メールボックスの生成 [Sp]
8635 acre_mbx メールボックスの生成 [TpD]
8636
8637 **【静的API】**
8638 CRE_MBX(ID mbxid, { ATR mbxatr, PRI maxmpri, void *mprihd })
8639
8640 **【C言語API】**
8641 ER_ID mbxid = acre_mbx(const T_CMBX *pk_cmbx)
8642
8643 **【パラメータ】**
8644 ID mbxid 生成するメールボックスのID番号（CRE_MBXの場
8645 合）
8646 T_CMBX * pk_cmbx メールボックスの生成情報を入れたパケットへ
8647 のポインタ（静的APIを除く）
8648
8649 * メールボックスの生成情報（パケットの内容）
8650 ATR mbxatr メールボックス属性

8651	PRI	maxmpri	優先度メールボックスに送信できるメッセージ
8652			優先度の最大値
8653	void *	mprihd	優先度別のメッセージキューヘッダ領域の先頭
8654			番地

8655

8656 **【リターンパラメータ】**

8657	ER_ID	mbxid	生成されたメールボックスのID番号（正の値）
8658			またはエラーコード

8659

8660 **【エラーコード】**

8661	E_CTX [s]	コンテキストエラー（非タスクコンテキストからの呼出し、CPUロック状態からの呼出し）
8662		
8663	E_RSATR	予約属性（mbxatrが不正または使用できない、属するクラスが不正）
8664		
8665	E_NOSPT	未サポート機能（mprihdがサポートされていない値）
8666	E_PAR	パラメータエラー（mprihdが不正）
8667	E_NOID [sD]	ID番号不足（割り付けられるメールボックスIDがない）
8668	E_NOMEM	メモリ不足（優先度別のメッセージキューヘッダ領域が確保できない）
8669		
8670	E_OBJ	オブジェクト状態エラー（mbxidで指定したメールボックスが登録済み：CRE_MBXの場合）
8671		

8672

8673 **【機能】**

8674

8675 各パラメータで指定したメールボックス生成情報に従って、メールボックスを

8676 生成する。メッセージキューはつながれているメッセージがない状態に初期化

8677 され、mprihdとmaxmpriから優先度別のメッセージキューヘッダ領域が設定され

8678 る。また、待ち行列は空の状態に初期化される。

8679

8680 静的APIにおいては、mbxidはオブジェクト識別名、maxmpriは整数定数式パラメー

8681 タ、mprihd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

8682 メモリ不足（E_NOMEM）エラーを検出することができない。

8683

8684 mprihdをNULLとした場合、maxmpriの指定に合致したサイズの優先度別のメッセー

8685 ジキューヘッダ領域を、コンフィギュレータまたはカーネルが確保する。

8686

8687 maxmpriは、TMIN_MPRI以上、TMAX_MPRI以下でなければならない。

8688

8689 **【TOPPERS/ASPカーネルにおける規定】**

8690

8691 ASPカーネルでは、CRE_MBXのみをサポートする。また、優先度別のメッセージ

8692 キューヘッダ領域は使用しておらず、mprihdにはNULLのみを指定することがで

8693 きる。NULL以外を指定した場合には、E_NOSPTエラーとなる。ただし、動的生成

8694 機能拡張パッケージでは、acre_mbxもサポートする。acre_mbxに対しても、

8695 mprihdにはNULLのみを指定することができる。

8696

8697 **【TOPPERS/FMPカーネルにおける規定】**

8698

8699 FMPカーネルでは、CRE_MBXのみをサポートする。また、優先度別のメッセージ

8700 キューヘッダ領域は使用しておらず、mprihdにはNULLのみを渡すことができる。

8701 NULL以外を指定した場合には、E_NOSPTエラーとなる。
 8702 -----

8703 AID_MBX 割付け可能なメールボックスIDの数の指定 [SpD]
 8704

8705 **【静的API】**
 8706 AID_MBX(uint_t nombx)
 8707

8708 **【パラメータ】**
 8709 uint_t nombx 割付け可能なメールボックスIDの数
 8710

8711 **【エラーコード】**
 8712 E_RSATR 予約属性（属する保護ドメインまたはクラスが不正）
 8713

8714 **【機能】**
 8715

8716 nombxで指定した数のメールボックスIDを、メールボックスを生成するサービス
 8717 コールによって割付け可能なメールボックスIDとして確保する。
 8718

8719 nombxは整数定数式パラメータである。
 8720 -----

8721 del_mbx メールボックスの削除 [TDp]
 8722

8723 **【C言語API】**
 8724 ER ercd = del_mbx(ID mbxid)
 8725

8726 **【パラメータ】**
 8727 ID mbxid 対象メールボックスのID番号
 8728

8729 **【リターンパラメータ】**
 8730 ER ercd 正常終了 (E_OK) またはエラーコード
 8731

8732 **【エラーコード】**
 8733 E_CTX コンテキストエラー（非タスクコンテキストからの呼出
 8734 し、CPUロック状態からの呼出し）
 8735 E_ID 不正ID番号（mbxidが不正）
 8736 E_NOEXS [D] オブジェクト未登録（対象メールボックスが未登録）
 8737 E_OBJ オブジェクト状態エラー（対象メールボックスは静的API
 8738 で生成された）
 8739

8740 **【機能】**
 8741

8742 mbxidで指定したメールボックス（対象メールボックス）を削除する。具体的な
 8743 振舞いは以下の通り。
 8744

8745 対象メールボックスの登録が解除され、そのメールボックスIDが未使用の状態
 8746 に戻される。また、対象メールボックスの待ち行列につながれたタスクは、待
 8747 ち行列の先頭のタスクから順に待ち解除される。待ち解除されたタスクには、
 8748 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。
 8749

8750 メールボックスの生成時に、優先度別のメッセージキューヘッダ領域がカーネ

8751 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。

8752

8753 **【使用上の注意】**

8754

8755 del_mbxにより複数のタスクが待ち解除される場合、サービスコールの処理時間
8756 およびカーネル内での割込み禁止時間が、待ち解除されるタスクの数に比例し
8757 て長くなる。特に、多くのタスクが待ち解除される場合、カーネル内での割込
8758 み禁止時間が長くなるため、注意が必要である。

8759

8760 **【TOPPERS/ASPカーネルにおける規定】**

8761

8762 ASPカーネルでは、del_mbxをサポートしない。ただし、動的生成機能拡張パッ
8763 ケージでは、del_mbxをサポートする。

8764

8765 **【TOPPERS/FMPカーネルにおける規定】**

8766

8767 FMPカーネルでは、del_mbxをサポートしない。

8768

8769 snd_mbx メールボックスへの送信 [Tp]

8770

8771 **【C言語API】**

8772 ER ercd = snd_mbx(ID mbxid, T_MSG *pk_msg)

8773

8774 **【パラメータ】**

8775 ID mbxid 対象メールボックスのID番号

8776 T_MSG *pk_msg 送信メッセージの先頭番地

8777

8778 **【リターンパラメータ】**

8779 ER ercd 正常終了 (E_OK) またはエラーコード

8780

8781 **【エラーコード】**

8782 E_CTX コンテキストエラー (非タスクコンテキストからの呼出
8783 し、CPUロック状態からの呼出し)

8784 E_ID 不正ID番号 (mbxidが不正)

8785 E_PAR パラメータエラー (メッセージヘッダ中のmsgpriが不正)

8786 E_NOEXS [D] オブジェクト未登録 (対象メールボックスが未登録)

8787

8788 **【機能】**

8789

8790 mbxidで指定したメールボックス (対象メールボックス) に、pk_msgで指定した
8791 メッセージを送信する。具体的な振舞いは以下の通り。

8792

8793 対象メールボックスの待ち行列にタスクが存在する場合には、待ち行列の先頭
8794 のタスクが、pk_msgで指定したメッセージを受信し、待ち解除される。待ち解
8795 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。

8796

8797 対象メールボックスの待ち行列にタスクが存在しない場合には、pk_msgで指定
8798 したメッセージが、メールボックス属性のTA_MPRI指定の有無によって指定され
8799 る順序で、メッセージキューにつなぐ。

8800

8801 対象メールボックスがTA_MPRI属性である場合には、pk_msgで指定したメッセー
 8802 ジの先頭のメッセージヘッダ中のmsgpriフィールドの値が、TMIN_MPRI以上で、
 8803 対象メールボックスに送信できるメッセージ優先度の最大値以下でなければな
 8804 らない。

8805 -----

8806 rcv_mbx メールボックスからの受信 [Tp]
 8807 prcv_mbx メールボックスからの受信（ポーリング） [Tp]
 8808 trcv_mbx メールボックスからの受信（タイムアウト付き） [Tp]

8809

8810 【C言語API】

8811 ER ercd = rcv_mbx(ID mbxid, T_MSG **ppk_msg)
 8812 ER ercd = prcv_mbx(ID mbxid, T_MSG **ppk_msg)
 8813 ER ercd = trcv_mbx(ID mbxid, T_MSG **ppk_msg, TMO tmout)

8814

8815 【パラメータ】

8816 ID mbxid 対象メールボックスのID番号
 8817 T_MSG ** ppk_msg 受信メッセージの先頭番地を入れるメモリ領域
 8818 へのポインタ
 8819 TMO tmout タイムアウト時間（trcv_mbxの場合）

8820

8821 【リターンパラメータ】

8822 ER ercd 正常終了 (E_OK) またはエラーコード
 8823 T_MSG * ppk_msg 受信メッセージの先頭番地

8824

8825 【エラーコード】

8826 E_CTX コンテキストエラー（非タスクコンテキストからの呼出
 8827 し、CPUロック状態からの呼出し、ディスパッチ保留状態
 8828 からの呼出し：prcv_mbxを除く）
 8829 E_NOSPT 未サポート機能（制約タスクからの呼出し：prcv_mbxを除
 8830 く）
 8831 E_ID 不正ID番号（mbxidが不正）
 8832 E_PAR パラメータエラー（tmoutが不正：trcv_mbxの場合）
 8833 E_NOEXS [D] オブジェクト未登録（対象メールボックスが未登録）
 8834 E_TMOUT ポーリング失敗またはタイムアウト（rcv_mbxを除く）
 8835 E_RLWAI 待ち禁止状態または待ち状態の強制解除（prcv_mbxを除く）
 8836 E_DLT 待ちオブジェクトの削除または再初期化（prcv_mbxを除く）

8837

8838 【機能】

8839

8840 mbxidで指定したメールボックス（対象メールボックス）からメッセージを受信
 8841 する。受信したメッセージの先頭番地は、ppk_msgで指定したメモリ領域に返さ
 8842 れる。具体的な振舞いは以下の通り。

8843

8844 対象メールボックスのメッセージキューにメッセージがつながれている場合
 8845 には、メッセージキューの先頭につながれたメッセージが取り出され、ppk_msgで
 8846 指定したメモリ領域に返される。

8847

8848 対象メールボックスのメッセージキューにメッセージがつながれていない場合
 8849 には、自タスクはメールボックスからの受信待ち状態となり、対象メールボッ
 8850 クスの待ち行列につながれる。

8851 -----
8852 ini_mbx メールボックスの再初期化 [Tp]
8853
8854 【C言語API】
8855 ER ercd = ini_mbx(ID mbxid)
8856
8857 【パラメータ】
8858 ID mbxid 対象メールボックスのID番号
8859
8860 【リターンパラメータ】
8861 ER ercd 正常終了 (E_OK) またはエラーコード
8862
8863 【エラーコード】
8864 E_CTX コンテキストエラー (非タスクコンテキストからの呼出
8865 し, CPUロック状態からの呼出し)
8866 E_ID 不正ID番号 (mbxidが不正)
8867 E_NOEXS [D] オブジェクト未登録 (対象メールボックスが未登録)
8868
8869 【機能】
8870
8871 mbxidで指定したメールボックス (対象メールボックス) を再初期化する. 具体
8872 的な振舞いは以下の通り.
8873
8874 対象メールボックスのメールボックス管理領域は, メッセージキューはつなが
8875 れているメッセージがない状態に初期化される. また, 対象メールボックスの
8876 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
8877 れる. 待ち解除されたタスクには, 待ち状態となったサービスコールから
8878 E_DLTエラーが返る.
8879
8880 【使用上の注意】
8881
8882 ini_mbxにより複数のタスクが待ち解除される場合, サービスコールの処理時間
8883 およびカーネル内での割込み禁止時間が, 待ち解除されるタスクの数に比例し
8884 て長くなる. 特に, 多くのタスクが待ち解除される場合, カーネル内での割込
8885 み禁止時間が長くなるため, 注意が必要である.
8886
8887 メールボックスを再初期化した場合に, アプリケーションとの整合性を保つのは,
8888 アプリケーションの責任である.
8889
8890 【μITRON4.0仕様との関係】
8891
8892 μITRON4.0仕様に定義されていないサービスコールである.
8893 -----
8894 ref_mbx メールボックスの状態参照 [Tp]
8895
8896 【C言語API】
8897 ER ercd = ref_mbx(ID mbxid, T_RMBX *pk_rmbx)
8898
8899 【パラメータ】
8900 ID mbxid 対象メールボックスのID番号

8901 T_RMBX * pk_rmbx メールボックスの現在状態を入れるパケットへ
8902 のポインタ
8903

8904 **【リターンパラメータ】**

8905 ER ercd 正常終了 (E_OK) またはエラーコード
8906

8907 * メールボックスの現在状態 (パケットの内容)

8908 ID wtskid メールボックスの待ち行列の先頭のタスクのID
8909 番号
8910 T_MSG * pk_msg メッセージキューの先頭につながれたメッセー
8911 ジの先頭番地
8912

8913 **【エラーコード】**

8914 E_CTX コンテキストエラー (非タスクコンテキストからの呼出
8915 し, CPUロック状態からの呼出し)
8916 E_ID 不正ID番号 (mbxidが不正)
8917 E_NOEXS [D] オブジェクト未登録 (対象メールボックスが未登録)
8918

8919 **【機能】**

8920

8921 mbxidで指定したメールボックス (対象メールボックス) の現在状態を参照する.
8922 参照した現在状態は, pk_rmbxで指定したパケットに返される.
8923

8924

8925 対象メールボックスの待ち行列にタスクが存在しない場合, wtskidには
8926 TSK_NONE (=0) が返る. また, メッセージキューにメッセージがつながれてい
8927 ない場合, pk_msgにはNULLが返る.
8928

8929 **【使用上の注意】**

8930

8931 ref_mbxはデバッグ時向けの機能であり, その他の目的に使用することは推奨し
8932 ない. これは, ref_mbxを呼び出し, 対象メールボックスの現在状態を参照した
8933 直後に割込みが発生した場合, ref_mbxから戻ってきた時には対象メールボッ
8934 クスの状態が変化している可能性があるためである.
8935

8936

8937

8938 4.4.6 ミューテックス

8939

8940 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
8941 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
8942 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
8943 呼ぶID番号によって識別する.
8944

8945

8946 ミューテックスは, 排他制御に伴う優先度逆転の時間を最小限に抑えるための
8947 優先度上限プロトコル (priority ceiling protocol) をサポートする. ミュー
8948 テックス属性により優先度上限ミューテックスであると指定することで, その
8949 ミューテックスの操作時に, 優先度上限プロトコルに従った現在優先度の制御
8950 が行われる.

8951

8952 各ミューテックスが持つ情報は次の通り.
8953

8954

8951 ・ ミューテックス属性
8952 ・ 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）
8953 ・ ミューテックスをロックしているタスク
8954 ・ 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）
8955 ・ 上限優先度（優先度上限ミューテックスの場合）
8956 ・ アクセス許可ベクタ（保護機能対応カーネルの場合）
8957 ・ 属する保護ドメイン（保護機能対応カーネルの場合）
8958 ・ 属するクラス（マルチプロセッサ対応カーネルの場合）
8959
8960 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。
8961
8962
8963
8964 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。
8965
8966
8967
8968 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。
8969
8970 TA_TPRI 0x01U 待ち行列をタスクの優先度順にする
8971 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
8972
8973
8974 TA_TPRI, TA_CEILINGのいずれも指定しない場合、待ち行列はFIFO順になる。
8975
8976 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。
8977
8978 ・ ロックしているミューテックスのリスト
8979
8980 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。
8981
8982
8983 タスクの現在優先度は、そのタスクのベース優先度と、そのタスクがロックしている優先度上限ミューテックスの優先度上限の中で、最も高い優先度に設定される。
8984
8985
8986
8987 ミューテックス機能によりタスクの現在優先度が変化する場合には、次の処理が行われる。現在優先度を変化させるサービスコールの前後とも、当該タスクが実行できる状態である場合には、同じ優先度のタスクの中で最高優先順位となる。そのサービスコールにより、当該タスクが実行できる状態に遷移する場合には、同じ優先度のタスクの中で最低優先順位となる。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。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。
8988
8989
8990
8991
8992
8993
8994
8995
8996
8997 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。
8998
8999
9000

9001
9002 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。
9003
9004 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
9005 ネルでは、静的APIによって登録されたミューテックスの
9006 数に一致）
9007
9008 **【使用上の注意】**
9009
9010 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
9011 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
9012 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
9013 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
9014 与しない場合に得られる利点である。
9015
9016 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
9017 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
9018 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、
9019 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
9020 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）。
9021
9022 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
9023 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
9024 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
9025 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、
9026 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。
9027 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
9028 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
9029 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
9030 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
9031 る。
9032
9033 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
9034 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
9035 らないploc_mtxを用いるのが安全である。
9036
9037 **【補足説明】**
9038
9039 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
9040 ceiling protocolとは異なるものである。この仕様の方式は、OSEK/VDX OS仕様
9041 でもpriority ceiling protocolと呼ばれているが、学术论文や他のOSでは、
9042 immediate ceiling priority protocol, priority protection protocol,
9043 priority ceiling emulation, highest locker protocolなどと呼ばれている。
9044
9045 **【TOPPERS/ASPカーネルにおける規定】**
9046
9047 ASPカーネルでは、ミューテックス機能をサポートしない。ただし、ミューテッ
9048 クス機能拡張パッケージを用いると、ミューテックス機能を追加することがで
9049 きる。
9050

9051 【TOPPERS/FMPカーネルにおける規定】

9052

9053 FMPカーネルでは、ミューテックス機能をサポートしない。

9054

9055 【TOPPERS/HRP2カーネルにおける規定】

9056

9057 HRP2カーネルでは、ミューテックス機能をサポートする。

9058

9059 【未決定事項】

9060

9061 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
 9062 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
 9063 度上限ミューテックスに対してタスクがロック待ち状態になる。マルチプロセッ
 9064 サ対応カーネルにおける優先度上限ミューテックスの扱いについては、今後の
 9065 課題である。

9066

9067 【 μ ITRON4.0仕様との関係】

9068

9069 μ ITRON4.0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
 9070 はサポートしていない。また、 μ ITRON4.0仕様でサポートしている優先度継承
 9071 プロトコル (priority inheritance protocol) は、現時点ではサポートしてい
 9072 ない。

9073

9074 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、
 9075 μ ITRON4.0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。

9076

9077 TNUM_MTXIDは、 μ ITRON4.0仕様に規定されていないカーネル構成マクロであ
 9078 る。

9079

9080 CRE_MTX ミューテックスの生成 [S]

9081 acre_mtx ミューテックスの生成 [TD]

9082

9083 【静的API】

9084 CRE_MTX(ID mtxid, { ATR mtxatr, PRI ceilpri })

9085

9086 【C言語API】

9087 ER_ID mtxid = acre_mtx(const T_CMTX *pk_cmtx)

9088

9089 【パラメータ】

9090 ID mtxid 生成するミューテックスのID番号 (CRE_MTXの
 9091 場合)

9092 T_CMTX * pk_cmtx ミューテックスの生成情報を入れたパケット
 9093 へのポインタ (静的APIを除く)

9094

9095 * ミューテックスの生成情報 (パケットの内容)

9096 ATR mtxatr ミューテックス属性

9097 PRI ceilpri ミューテックスの上限優先度

9098

9099 【リターンパラメータ】

9100 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 [s]	コンテキストエラー（非タスクコンテキストからの呼出し、CPUロック状態からの呼出し）
E_RSATR	予約属性（mtxatrが不正または使用できない、属する保護ドメインかクラスが不正）
E_PAR	パラメータエラー（ceilpriが不正）
E_OACV [sP]	オブジェクトアクセス違反（システム状態に対する管理操作が許可されていない）
E_MACV [sP]	メモリアクセス違反（pk_cmtxが指すメモリ領域への読出しアクセスが許可されていない）
E_NOID [sD]	ID番号不足（割り付けられるミューテックスIDがない）
E_OBJ	オブジェクト状態エラー（mtxidで指定したミューテックスが登録済み：CRE_MTXの場合）

【機能】

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

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

優先度上限ミューテックスを生成する場合、ceilpriは、TMIN_TPRI以上、TMAX_TPRI以下でなければならない。

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルのミューテックス機能拡張パッケージでは、CRE_MTXのみをサポートする。

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、CRE_MTXのみをサポートする。

AID_MTX 割付け可能なミューテックスIDの数の指定 [SD]

【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（属する保護ドメインまたはクラスが不正）

【機能】

9151
 9152 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
 9153 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。
 9154
 9155 nomtxは整数定数式パラメータである。
 9156 -----
 9157 SAC_MTX ミューテックスのアクセス許可ベクタの設定 [SP]
 9158 sac_mtx ミューテックスのアクセス許可ベクタの設定 [TPD]
 9159
 9160 **【静的API】**
 9161 SAC_MTX(ID mtxid, { ACPTN acptn1, ACPTN acptn2,
 9162 ACPTN acptn3, ACPTN acptn4 })
 9163
 9164 **【C言語API】**
 9165 ER ercd = sac_mtx(ID mtxid, const ACVCT *p_acvct)
 9166
 9167 **【パラメータ】**
 9168 ID mtxid 対象ミューテックスのID番号
 9169 ACVCT * p_acvct アクセス許可ベクタを入れたパケットへのポ
 9170 インタ（静的APIを除く）
 9171
 9172 *アクセス許可ベクタ（パケットの内容）
 9173 ACPTN acptn1 通常操作1のアクセス許可パターン
 9174 ACPTN acptn2 通常操作2のアクセス許可パターン
 9175 ACPTN acptn3 管理操作のアクセス許可パターン
 9176 ACPTN acptn4 参照操作のアクセス許可パターン
 9177
 9178 **【リターンパラメータ】**
 9179 ER ercd 正常終了（E_OK）またはエラーコード
 9180
 9181 **【エラーコード】**
 9182 E_CTX [s] コンテキストエラー（非タスクコンテキストからの呼出
 9183 し、CPUロック状態からの呼出し）
 9184 E_RSATR 予約属性（属する保護ドメインかクラスが不正：SAC_MTX
 9185 の場合）
 9186 E_ID 不正ID番号（mtxidが不正）
 9187 E_NOEXS [D] オブジェクト未登録（対象ミューテックスが未登録）
 9188 E_OACV [sP] オブジェクトアクセス違反（対象ミューテックスに対す
 9189 る管理操作が許可されていない）
 9190 E_MACV [sP] メモリアクセス違反（p_acvctが指すメモリ領域への読出
 9191 しアクセスが許可されていない）
 9192 E_OBJ オブジェクト状態エラー（対象ミューテックスは静的API
 9193 で生成された：sac_mtxの場合、対象ミューテックスに対
 9194 してアクセス許可ベクタが設定済み：SAC_MTXの場合）
 9195
 9196 **【機能】**
 9197
 9198 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
 9199 （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。
 9200

9201 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
9202 式パラメータである。

9203

9204 SAC_MTXは、対象ミューテックスが属する保護ドメインの囲みの中に記述しなけ
9205 ればならない。そうでない場合には、E_RSATRエラーとなる。

9206

9207 **【TOPPERS/ASPカーネルにおける規定】**

9208

9209 ASPカーネルのミューテックス機能拡張パッケージでは、SAC_MTX、sac_mtxをサ
9210 ポートしない。

9211

9212 **【TOPPERS/HRP2カーネルにおける規定】**

9213

9214 HRP2カーネルでは、SAC_MTXのみをサポートする。

9215

9216 del_mtx ミューテックスの削除 [TD]

9217

9218 **【C言語API】**

9219 ER ercd = del_mtx(ID mtxid)

9220

9221 **【パラメータ】**

9222 ID mtxid 対象ミューテックスのID番号

9223

9224 **【リターンパラメータ】**

9225 ER ercd 正常終了 (E_OK) またはエラーコード

9226

9227 **【エラーコード】**

9228 E_CTX コンテキストエラー (非タスクコンテキストからの呼出
9229 し、CPUロック状態からの呼出し)

9230 E_ID 不正ID番号 (mtxidが不正)

9231 E_NOEXS [D] オブジェクト未登録 (対象ミューテックスが未登録)

9232 E_OACV [P] オブジェクトアクセス違反 (対象ミューテックスに対す
9233 る管理操作が許可されていない)

9234 E_OBJ オブジェクト状態エラー (対象ミューテックスは静的API
9235 で生成された)

9236

9237 **【機能】**

9238

9239 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
9240 振舞いは以下の通り。

9241

9242 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
9243 に戻される。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
9244 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
9245 削除され、必要な場合にはそのタスクの現在優先度に変更される。また、対象
9246 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
9247 ら順に待ち解除される。待ち解除されたタスクには、待ち状態となったサービ
9248 スコールからE_DLTエラーが返る。

9249

9250 **【使用上の注意】**

9251
9252 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
9253 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
9254 る。これが不都合な場合には、ミューテックスをロックした状態で、ミューテッ
9255 クスを削除すればよい。

9256
9257 del_mtxにより複数のタスクが待ち解除される場合、サービスコールの処理時間
9258 およびカーネル内での割込み禁止時間が、待ち解除されるタスクの数に比例し
9259 て長くなる。特に、多くのタスクが待ち解除される場合、カーネル内での割込
9260 み禁止時間が長くなるため、注意が必要である。

9261
9262 【TOPPERS/ASPカーネルにおける規定】

9263
9264 ASPカーネルのミューテックス機能拡張パッケージでは、del_mtxをサポートし
9265 ない。

9266
9267 【TOPPERS/HRP2カーネルにおける規定】

9268
9269 HRP2カーネルでは、del_mtxをサポートしない。

9270 -----

9271 loc_mtx ミューテックスのロック [T]
9272 ploc_mtx ミューテックスのロック (ポーリング) [T]
9273 tloc_mtx ミューテックスのロック (タイムアウト付き) [T]

9274
9275 【C言語API】

9276 ER ercd = loc_mtx(ID mtxid)
9277 ER ercd = ploc_mtx(ID mtxid)
9278 ER ercd = tloc_mtx(ID mtxid, TMO tmout)

9279
9280 【パラメータ】

9281 ID mtxid 対象ミューテックスのID番号
9282 TMO tmout タイムアウト時間 (twai_mtxの場合)

9283
9284 【リターンパラメータ】

9285 ER ercd 正常終了 (E_OK) またはエラーコード

9286
9287 【エラーコード】

9288 E_CTX コンテキストエラー (非タスクコンテキストからの呼出
9289 し、CPUロック状態からの呼出し、ディスパッチ保留状態
9290 からの呼出し：pol_mtxを除く)
9291 E_NOSPT 未サポート機能 (制約タスクからの呼出し：pol_mtxを除
9292 く)
9293 E_ID 不正ID番号 (mtxidが不正)
9294 E_PAR パラメータエラー (tmoutが不正：twai_mtxの場合)
9295 E_NOEXS [D] オブジェクト未登録 (対象ミューテックスが未登録)
9296 E_OACV [P] オブジェクトアクセス違反 (対象ミューテックスに対す
9297 る通常操作1が許可されていない)
9298 E_ILUSE サービスコール不正使用 (対象ミューテックスを自タ
9299 スクがロックしている、対象優先度上限ミューテックスの
9300 上限優先度より自タスクのベース優先度が高い)

9301 E_TMOUT ポーリング失敗またはタイムアウト (wai_mtxを除く)
 9302 E_RLWAI 待ち禁止状態または待ち状態の強制解除 (pol_mtxを除く)
 9303 E_DLT 待ちオブジェクトの削除または再初期化 (pol_mtxを除く)

9304

9305 **【機能】**

9306

9307 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
 9308 な振舞いは以下の通り.

9309

9310 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
 9311 されている状態になる. 自タスクがロックしているミューテックスのリストに
 9312 対象ミューテックスが追加され, 必要な場合には自タスクの現在優先度の変更
 9313 される.

9314

9315 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
 9316 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
 9317 の待ち行列につながる.

9318

9319 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場合には, E_ILUSEエ
 9320 ラーとなる. また, 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
 9321 上限優先度より自タスクのベース優先度が高い場合にも, E_ILUSEエラーとなる.

9322

9323 unl_mtx ミューテックスのロック解除 [T]

9324

9325 **【C言語API】**

9326 ER ercd = unl_mtx(ID mtxid)

9327

9328 **【パラメータ】**

9329 ID mtxid 対象ミューテックスのID番号

9330

9331 **【リターンパラメータ】**

9332 ER ercd 正常終了 (E_OK) またはエラーコード

9333

9334 **【エラーコード】**

9335 E_CTX コンテキストエラー (非タスクコンテキストからの呼出
 9336 し, CPUロック状態からの呼出し)

9337 E_ID 不正ID番号 (mtxidが不正)

9338 E_NOEXS [D] オブジェクト未登録 (対象ミューテックスが未登録)

9339 E_ILUSE サービスコール不正使用 (対象ミューテックスを自タス
 9340 クがロックしていない)

9341

9342 **【機能】**

9343

9344 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
 9345 体的な振舞いは以下の通り.

9346

9347 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
 9348 クスが削除され, 必要な場合には自タスクの現在優先度の変更される.

9349

9350 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

9351 のタスクが待ち解除される。対象ミューテックスは、待ち解除されたタスクに
9352 よってロックされている状態になる。待ち解除されたタスクがロックしている
9353 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、必要な場合にはそ
9354 のタスクの現在優先度の変更される。待ち解除されたタスクには、待ち状態と
9355 なったサービスコールからE_OKが返る。
9356
9357 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
9358 いない状態になる。
9359
9360 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ない場合には、
9361 E_ILUSEエラーとなる。
9362 -----
9363 ini_mtx ミューテックスの再初期化 [T]
9364
9365 【C言語API】
9366 ER ercd = ini_mtx(ID mtxid)
9367
9368 【パラメータ】
9369 ID mtxid 対象ミューテックスのID番号
9370
9371 【リターンパラメータ】
9372 ER ercd 正常終了 (E_OK) またはエラーコード
9373
9374 【エラーコード】
9375 E_CTX コンテキストエラー (非タスクコンテキストからの呼出
9376 し、CPUロック状態からの呼出し)
9377 E_ID 不正ID番号 (mtxidが不正)
9378 E_NOEXS [D] オブジェクト未登録 (対象ミューテックスが未登録)
9379 E_OACV [P] オブジェクトアクセス違反 (対象ミューテックスに対す
9380 る管理操作が許可されていない)
9381
9382 【機能】
9383
9384 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
9385 的な振舞いは以下の通り。
9386
9387 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。
9388 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
9389 クしているミューテックスのリストから対象ミューテックスが削除され、必要
9390 な場合にはそのタスクの現在優先度の変更される。また、対象ミューテックス
9391 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
9392 される。待ち解除されたタスクには、待ち状態となったサービスコールから
9393 E_DLTエラーが返る。
9394
9395 【使用上の注意】
9396
9397 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
9398 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
9399 となる。これが不都合な場合には、ミューテックスをロックした状態で、ミュー
9400 テックスを再初期化すればよい。

9401
 9402 ini_mtxにより複数のタスクが待ち解除される場合、サービスコールの処理時間
 9403 およびカーネル内での割込み禁止時間が、待ち解除されるタスクの数に比例し
 9404 て長くなる。特に、多くのタスクが待ち解除される場合、カーネル内での割込
 9405 み禁止時間が長くなるため、注意が必要である。

9406
 9407 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
 9408 は、アプリケーションの責任である。

9409
 9410 **【μITRON4.0仕様との関係】**

9411
 9412 μITRON4.0仕様に定義されていないサービスコールである。
 9413 -----

9414 ref_mtx ミューテックスの状態参照 [T]

9415
 9416 **【C言語API】**

9417 ER ercd = ref_mtx(ID mtxid, T_RMTX *pk_rmtx)

9418
 9419 **【パラメータ】**

9420	ID	mtxid	対象ミューテックスのID番号
9421	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

9422
 9423

9424 **【リターンパラメータ】**

9425	ER	ercd	正常終了 (E_OK) またはエラーコード
------	----	------	-----------------------

9426
 9427 * ミューテックスの現在状態 (パケットの内容)

9428	ID	htskid	ミューテックスをロックしているタスクのID番号
9429	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

9430
 9431

9432 **【エラーコード】**

9433	E_CTX	コンテキストエラー (非タスクコンテキストからの呼出 し, CPUロック状態からの呼出し)
9434	E_ID	不正ID番号 (mtxidが不正)
9435	E_NOEXS [D]	オブジェクト未登録 (対象ミューテックスが未登録)
9436	E_OACV [P]	オブジェクトアクセス違反 (対象ミューテックスに対す る参照操作が許可されていない)
9437	E_MACV [P]	メモリアクセス違反 (pk_rmtxが指すメモリ領域への書込 みアクセスが許可されていない)

9438
 9439
 9440
 9441

9442 **【機能】**

9443
 9444 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。
 9445 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。

9446
 9447 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
 9448 返る。

9449
 9450 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

9451 TSK_NONE (=0) が返る.

9452

9453 **【使用上の注意】**

9454

9455 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
9456 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
9457 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
9458 スの状態が変化している可能性があるためである。

9459 -----

9460

9461 4.4.7 メッセージバッファ

9462

9463 ☆未完成

9464

9465 4.4.8 スピンロック

9466

9467 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
9468 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
9469 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。

9470

9471 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
9472 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。ロックが他
9473 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
9474 て待つ。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
9475 されない。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
9476 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
9477 状態を解除することを、スピンロックを返却するという。

9478

9479 タスクが取得したスピンロックを返却せずに終了した場合や、タスク例外処理
9480 ルーチン、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
9481 ラが取得したスピンロックを返却せずにリターンした場合には、カーネルによっ
9482 てスピンロックが返却される。また、スピンロックを取得していない状態で発
9483 生したCPU例外によって呼び出されたCPU例外ハンドラが、取得したスピンロッ
9484 ク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
9485 される。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
9486 されない。

9487

9488 各スピンロックが持つ情報は次の通り。

9489

- 9490 ・ スピンロック属性
- 9491 ・ 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）
- 9492 ・ アクセス許可ベクタ（保護機能対応カーネルの場合）
- 9493 ・ 属する保護ドメイン（保護機能対応カーネルの場合）
- 9494 ・ 属するクラス

9495

9496 スピンロック属性に指定できる属性はない。そのためスピンロック属性には、
9497 TA_NULLを指定しなければならない。

9498

9499 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。

9500

9501 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
 9502 ネルでは、静的APIによって登録されたスピンロックの数
 9503 に一致）
 9504

9505 **【補足説明】**

9506
 9507 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
 9508 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
 9509 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
 9510 クを取得した状態になることはできない。

9511
 9512 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
 9513 ンドラはカーネル管理外のCPU例外ハンドラであり（xsns_dpn, xsns_xpnとも
 9514 trueを返す）、CPU例外ハンドラ中でiunl_spnを呼び出してスピンロックを返却
 9515 しようとした場合の動作は保証されない。保証されないにも関わらずiunl_spn
 9516 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
 9517 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。

9518
 9519 **【TOPPERS/ASPカーネルにおける規定】**

9520
 9521 ASPカーネルでは、スピンロック機能をサポートしない。

9522
 9523 **【TOPPERS/FMPカーネルにおける規定】**

9524
 9525 FMPカーネルでは、スピンロック機能をサポートする。

9526
 9527 **【TOPPERS/HRP2カーネルにおける規定】**

9528
 9529 HRP2カーネルでは、スピンロック機能をサポートしない。

9530
 9531 **【μ ITRON4.0仕様との関係】**

9532
 9533 スピンロック機能は、μ ITRON4.0仕様に定義されていない機能である。

9534 -----
 9535 CRE_SPN スピンロックの生成 [SM]
 9536 acre_spn スピンロックの生成 [TMD]

9537
 9538 **【静的API】**

9539 CRE_SPN(ID spnid, { ATR spnatr })

9540
 9541 **【C言語API】**

9542 ER_ID spnid = acre_spn(const T_CSPN *pk_cspn)

9543
 9544 **【パラメータ】**

9545 ID spnid 生成するスピンロックのID番号（CRE_SPNの場合）
 9546 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
 9547 ポインタ（静的APIを除く）
 9548

9549 * スピンロックの生成情報（パケットの内容）

9550 ATR spnatr スピンロック属性

9551

9552 **【リターンパラメータ】**

9553 ER_ID spnid 生成されたスピンロックのID番号（正の値）ま
9554 たはエラーコード

9555

9556 **【エラーコード】**

9557 E_CTX [s] コンテキストエラー（非タスクコンテキストからの呼出
9558 し、CPUロック状態からの呼出し）
9559 E_RSATR 予約属性（spnatrが不正または使用できない、属する保
9560 護ドメインかクラスが不正）
9561 E_OACV [sP] オブジェクトアクセス違反（システム状態に対する管理
9562 操作が許可されていない）
9563 E_MACV [sP] メモリアクセス違反（pk_cspnが指すメモリ領域への読出
9564 しアクセスが許可されていない）
9565 E_NOID [sD] ID番号不足（割り付けられるスピンロックIDがない）
9566 E_NORES 資源不足（スピンロックを実現するためのハードウェア
9567 資源がない：CRE_SPNの場合）
9568 E_OBJ オブジェクト状態エラー（spnidで指定したスピンロック
9569 が登録済み：CRE_SPNの場合）

9570

9571 **【機能】**

9572

9573 各パラメータで指定したスピンロック生成情報に従って、スピンロックを生成
9574 する。生成されたスピンロックのロック状態は、取得されていない状態に初期
9575 化される。

9576

9577 静的APIにおいては、spnidはオブジェクト識別名である。

9578

9579 スピンロックをハードウェアによって実現している場合には、ターゲット定義
9580 で、生成できるスピンロックの数に上限がある。この上限を超えてスピンロッ
9581 クを生成しようとした場合には、E_NORESエラーとなる。

9582

9583 **【補足説明】**

9584

9585 スピンロックを動的に生成する場合に、生成できるスピンロックの数の上限は
9586 AID_SPNによってチェックされるため、acre_spnでE_NORESエラーが返ることは
9587 ない。

9588

9589 **【TOPPERS/FMPカーネルにおける規定】**

9590

9591 FMPカーネルでは、CRE_SPNのみをサポートする。

9592

9593 AID_SPN 割付け可能なスピンロックIDの数の指定 [SMD]

9594

9595 **【静的API】**

9596 AID_SPN(uint_t nospn)

9597

9598 **【パラメータ】**

9599 uint_t nospn 割付け可能なスピンロックIDの数

9600

193

9651
 9652 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
 9653 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。
 9654
 9655 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
 9656 式パラメータである。
 9657
 9658 SAC_SPNは、対象スピンロックが属する保護ドメインの囲みの中に記述しなけれ
 9659 ばならない。そうでない場合には、E_RSATRエラーとなる。
 9660
 9661 **【TOPPERS/FMPカーネルにおける規定】**
 9662
 9663 FMPカーネルでは、SAC_SPN、sac_spnをサポートしない。
 9664 -----
 9665 del_spn スピンロックの削除 [TMD]
 9666
 9667 **【C言語API】**
 9668 ER ercd = del_spn(ID spnid)
 9669
 9670 **【パラメータ】**
 9671 ID spnid 対象スピンロックのID番号
 9672
 9673 **【リターンパラメータ】**
 9674 ER ercd 正常終了 (E_OK) またはエラーコード
 9675
 9676 **【エラーコード】**
 9677 E_CTX コンテキストエラー（非タスクコンテキストからの呼出
 9678 し、CPUロック状態からの呼出し）
 9679 E_ID 不正ID番号（spnidが不正）
 9680 E_NOEXS [D] オブジェクト未登録（対象スピンロックが未登録）
 9681 E_OACV [P] オブジェクトアクセス違反（対象スピンロックに対する
 9682 管理操作が許可されていない）
 9683 E_OBJ オブジェクト状態エラー（対象スピンロックは静的APIで
 9684 生成された）
 9685
 9686 **【機能】**
 9687
 9688 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
 9689 いは以下の通り。
 9690
 9691 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
 9692 される。
 9693
 9694 **【TOPPERS/FMPカーネルにおける規定】**
 9695
 9696 FMPカーネルでは、del_spnをサポートしない。
 9697
 9698 **【未決定事項】**
 9699
 9700 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。

```

9701 -----
9702 loc_spn      スピンロックの取得 [TM]
9703 iloc_spn     スピンロックの取得 [IM]
9704
9705 【C言語API】
9706     ER ercd = loc_spn(ID spnid)
9707     ER ercd = iloc_spn(ID spnid)
9708
9709 【パラメータ】
9710     ID          spnid      対象スピンロックのID番号
9711
9712 【リターンパラメータ】
9713     ER          ercd      正常終了 (E_OK) またはエラーコード
9714
9715 【エラーコード】
9716     E_CTX      コンテキストエラー (非タスクコンテキストからの呼出
9717                  し: loc_spnの場合, タスクコンテキストからの呼出し:
9718                  iloc_spnの場合, CPUロック状態からの呼出し)
9719     E_ID        不正ID番号 (spnidが不正)
9720     E_NOEXS [D] オブジェクト未登録 (対象スピンロックが未登録)
9721     E_OACV [P] オブジェクトアクセス違反 (対象スピンロックに対する
9722                  通常操作1が許可されていない: loc_spnの場合)
9723
9724 【機能】
9725
9726 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
9727 いは以下の通り.
9728
9729 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
9730 クの取得を試みる. 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
9731 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
9732 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. これを,
9733 ロックの取得に成功するまで繰り返す.
9734
9735 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.
9736 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
9737 からリターンする.
9738
9739 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
9740 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.
9741
9742 【補足説明】
9743
9744 対象スピンロックが, loc_spn/iloc_spnを呼び出したプロセッサによって取得
9745 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
9746 ているため, loc_spn/iloc_spnはE_CTXエラーとなる.
9747
9748 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,
9749 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
9750 い限りは) loc_spn/iloc_spnを呼び出した順序でロックを取得できるとするの

```

9751 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
 9752 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。

9753 -----

9754 try_spn スピンロックの取得（ポーリング） [TM]
 9755 itry_spn スピンロックの取得（ポーリング） [IM]

9756

9757 **【C言語API】**

9758 ER ercd = try_spn(ID spnid)
 9759 ER ercd = itry_spn(ID spnid)

9760

9761 **【パラメータ】**

9762 ID spnid 対象スピンロックのID番号

9763

9764 **【リターンパラメータ】**

9765 ER ercd 正常終了 (E_OK) またはエラーコード

9766

9767 **【エラーコード】**

9768 E_CTX コンテキストエラー（非タスクコンテキストからの呼出
 9769 し：try_spnの場合、タスクコンテキストからの呼出し：
 9770 itry_spnの場合、CPUロック状態からの呼出し）
 9771 E_ID 不正ID番号（spnidが不正）
 9772 E_NOEXS [D] オブジェクト未登録（対象スピンロックが未登録）
 9773 E_OACV [P] オブジェクトアクセス違反（対象スピンロックに対する
 9774 通常操作1が許可されていない：try_spnの場合）
 9775 E_OBJ オブジェクト状態エラー（対象スピンロックが取得され
 9776 ている状態）

9777

9778 **【機能】**

9779

9780 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
 9781 振舞いは以下の通り。

9782

9783 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
 9784 クの取得を試みる。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
 9785 れている状態になる。また、CPUロックフラグをセットしてCPUロック状態へ遷
 9786 移し、サービスコールからリターンする。

9787

9788 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、
 9789 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）
 9790 には、E_OBJエラーとする。

9791 -----

9792 unl_spn スピンロックの返却 [TM]
 9793 iunl_spn スピンロックの返却 [IM]

9794

9795 **【C言語API】**

9796 ER ercd = unl_spn(ID spnid)
 9797 ER ercd = iunl_spn(ID spnid)

9798

9799 **【パラメータ】**

9800 ID spnid 対象スピンロックのID番号

9801
 9802 **【リターンパラメータ】**
 9803 ER ercd 正常終了 (E_OK) またはエラーコード
 9804
 9805 **【エラーコード】**
 9806 E_CTX コンテキストエラー (非タスクコンテキストからの呼出し : unl_spnの場合, タスクコンテキストからの呼出し : iunl_spnの場合)
 9807
 9808 E_ID 不正ID番号 (spnidが不正)
 9809 E_NOEXS [D] オブジェクト未登録 (対象スピンロックが未登録)
 9810 E_OACV [P] オブジェクトアクセス違反 (対象スピンロックに対する通常操作1が許可されていない : unl_spnの場合)
 9811 E_ILUSE サービスコール不正使用 (対象スピンロックをロックしていない)
 9812
 9813
 9814
 9815
 9816 **【機能】**
 9817
 9818 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.
 9819
 9820
 9821 対象スピンロックが, unl_spn/iunl_spnを呼び出したプロセッサによって取得されている状態である場合には, ロックを返却し, スピンロックを取得されていない状態とする. 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.
 9822
 9823
 9824
 9825
 9826 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.
 9827
 9828 -----
 9829 ref_spn スピンロックの状態参照 [TM]
 9830
 9831 **【C言語API】**
 9832 ER ercd = ref_spn(ID spnid, T_RSPN *pk_rspn)
 9833
 9834 **【パラメータ】**
 9835 ID spnid 対象スピンロックのID番号
 9836 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
 9837
 9838
 9839 **【リターンパラメータ】**
 9840 ER ercd 正常終了 (E_OK) またはエラーコード
 9841
 9842 * スピンロックの現在状態 (パケットの内容)
 9843 STAT spnstat ロック状態
 9844
 9845 **【エラーコード】**
 9846 E_CTX コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
 9847
 9848 E_ID 不正ID番号 (spnidが不正)
 9849 E_NOEXS [D] オブジェクト未登録 (対象スピンロックが未登録)
 9850 E_OACV [P] オブジェクトアクセス違反 (対象スピンロックに対する

E_MACV [P]

9855
9856

9857
9858
9859

9860
9861

9863
9864
9865

9866
9867

9868
9869
9870
9871
9872

9874
9875

9876
9877

9878
9879

9880
9881

9882
9883

9884
9885

9886
9887
9888
9889
9890

9891
9892

9893
9894
9895
9896

9897
9898

- 9899
-
- 9900

9901 • 固定長メモリプール領域
 9902 • 固定長メモリプール管理領域
 9903 • アクセス許可ベクタ（保護機能対応カーネルの場合）
 9904 • 属する保護ドメイン（保護機能対応カーネルの場合）
 9905 • 属するクラス（マルチプロセッサ対応カーネルの場合）
 9906
 9907 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
 9908 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
 9909 きる順序でつながれているキューである。
 9910
 9911 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
 9912 めのメモリ領域である。
 9913
 9914 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
 9915 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
 9916 のメモリ領域である。
 9917
 9918 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
 9919 用いるオブジェクト管理領域として扱われる。
 9920
 9921 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。
 9922
 9923 TA_TPRI 0x01U 待ち行列をタスクの優先度順にする
 9924
 9925 TA_TPRIを指定しない場合、待ち行列はFIFO順になる。
 9926
 9927 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。
 9928
 9929 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
 9930 カーネルでは、静的APIによって登録された固定長メモリ
 9931 プールの数に一致）
 9932
 9933 【 μ ITRON4.0仕様との関係】
 9934
 9935 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
 9936 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
 9937 ンで確保する方法を定めた結果、そのサイズは($\text{blkcnt} * \text{ROUND_MPF_T}(\text{blkksz})$)
 9938 で求めることができるようになったためである。
 9939
 9940 TNUM_MPFIDは、 μ ITRON4.0仕様に規定されていないカーネル構成マクロである。
 9941 -----
 9942 CRE_MPF 固定長メモリプールの生成 [S]
 9943 acre_mpf 固定長メモリプールの生成 [TD]
 9944
 9945 【静的API】
 9946 CRE_MPF(ID mpfid, { ATR mpfatr, uint_t blkcnt, uint_t blkksz,
 9947 MPF_T *mpf, void *mpfmb })
 9948
 9949 【C言語API】
 9950 ER_ID mpfid = acre_mpf(const T_CMPF *pk_cmpf)

9951

9952 **【パラメータ】**

9953 ID mpfid 生成する固定長メモリプールのID番号 (CRE_MPF

9954 の場合)

9955 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

9956 トへのポインタ (静的APIを除く)

9957

9958 * 固定長メモリプールの生成情報 (パケットの内容)

9959 ATR mpfatr 固定長メモリプール属性

9960 uint_t blkcnt 獲得できる固定長メモリブロックの数

9961 uint_t blkksz 固定長メモリブロックのサイズ (バイト数)

9962 MPF_T * mpf 固定長メモリプール領域の先頭番地

9963 void * mpfmb 固定長メモリプール管理領域の先頭番地

9964

9965 **【リターンパラメータ】**

9966 ER_ID mpfid 生成された固定長メモリプールのID番号 (正の

9967 値) またはエラーコード

9968

9969 **【エラーコード】**

9970 E_CTX [s] コンテキストエラー (非タスクコンテキストからの呼出

9971 し, CPUロック状態からの呼出し)

9972 E_RSATR 予約属性 (mpfatrが不正または使用できない, 属する保

9973 護ドメインがクラスが不正)

9974 E_NOSPT 未サポート機能 (mpfmbがサポートされていない値)

9975 E_PAR パラメータエラー (blkcnt, blkksz, mpf, mpfmbが不正)

9976 E_OACV [sP] オブジェクトアクセス違反 (システム状態に対する管理

9977 操作が許可されていない)

9978 E_MACV [sP] メモリアクセス違反 (pk_cmpfが指すメモリ領域への読出

9979 しアクセスが許可されていない)

9980 E_NOID [sD] ID番号不足 (割り付けられる固定長メモリプールIDがな

9981 い)

9982 E_NOMEM メモリ不足 (固定長メモリプール領域や固定長メモリプー

9983 ル管理領域が確保できない)

9984 E_OBJ オブジェクト状態エラー (mpfidで指定した固定長メモリ

9985 プールが登録済み: CRE_MPFの場合, その他の条件につい

9986 ては機能の項を参照すること)

9987

9988 **【機能】**

9989

9990 各パラメータで指定した固定長メモリプール生成情報に従って, 固定長メモリ

9991 プールを生成する. mpf, blkcnt, blkkszから固定長メモリプール領域が,

9992 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

9993 ル領域全体が未割当ての状態に初期化される. また, 待ち行列は空の状態に初

9994 期化される.

9995

9996 静的APIにおいては, mpfidはオブジェクト識別名, blkcntとblkkszは整数定数式

9997 パラメータ, mpfとmpfmbは一般定数式パラメータである. コンフィギュレータ

9998 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.

9999

10000 mpfをNULLとした場合, blkcntとblkkszから決まるサイズの固定長メモリプール

領域が、コンフィギュレータまたはカーネルにより確保される。

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

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

blkcntとblkszは、0より大きい値でなければならない。

[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。固定長メモリプール領域をアプリケーションで確保するために、次のデータ型とマクロを用意している。

MPF_T	固定長メモリプール領域を確保するためのデータ型
COUNT_MPF_T(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)

これらを用いて固定長メモリプール領域を確保する方法は次の通り。

```
MPF_T <固定長メモリプール領域の変数名>[(blkcnt) * COUNT_MPF_T(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。

これ以外の方法で固定長メモリプール領域を確保する場合には、先頭番地がターゲット定義の制約に合致しており、上記の配列と同じサイズのメモリ領域を確保しなければならない。mpfにターゲット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。

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。

[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。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。

TSZ_MPFMB(blkcnt)	blkcntで指定した数の固定長メモリブロックを管理
-------------------	----------------------------

10053	TCNT_MPFMB(blkent)	blkcntで指定した数の固定長メモリブロックを管理
10054		することができる固定長メモリプール管理領域を確
10055		保するために必要なMB T型の配列の要素数

```
10058
10059     MB T    <固定長メモリプール管理領域の変数名>[TCNT MPFMB(blkcnt)];
```

10062
10063 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
10064 した時には、E_PARエラーとなる。また、保護機能対応カーネルにおいて、
10065 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
10066 クトに含まれない場合、E_OBJエラーとなる。

10069
10070 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
10071 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
10072 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
10073 ンの責任である。

```

10076
10077     ASPカーネルでは、CRE_MPFのみをサポートする。また、mpfmbにはNULLのみを指
10078     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。た
10079     だし、動的生成機能拡張パッケージでは、acre_mpfもサポートする。acre_mpf
10080     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。

```

10083
10084 FMPカーネルでは、CRE_MPFのみをサポートする。また、mpfmbにはNULLのみを渡
10085 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。

10088
10089 HRP2カーネルでは、CRE_MPFのみをサポートする。また、mpfmbにはNULLのみを
10090 渡すことができる。NULL以外を指定した場合には、E_NOSPTエラーとなる。

10093
10094 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
10095 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，
10096 μ ITRON4.0/PX仕様にあわせて，固定長メモリプール生成情報に，mpfmb を追加
10097 した．

10100

10101 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
 10102 する方法を規定した。

10103

10104 AID_MPF 割付け可能な固定長メモリプールIDの数の指定 [SD]

10105

10106 **【静的API】**

10107 AID_MPF(uint_t nompf)

10108

10109 **【パラメータ】**

10110 uint_t nompf 割付け可能な固定長メモリプールIDの数

10111

10112 **【エラーコード】**

10113 E_RSATR 予約属性（属する保護ドメインまたはクラスが不正）

10114

10115 **【機能】**

10116

10117 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
 10118 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。

10119

10120 nompfは整数定数式パラメータである。

10121

10122 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]

10123 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]

10124

10125 **【静的API】**

10126 SAC_MPF(ID mpfid, { ACPTN acptn1, ACPTN acptn2,
 10127 ACPTN acptn3, ACPTN acptn4 })

10128

10129 **【C言語API】**

10130 ER ercd = sac_mpf(ID mpfid, const ACVCT *p_acvct)

10131

10132 **【パラメータ】**

10133 ID mpfid 対象固定長メモリプールのID番号
 10134 ACVCT * p_acvct アクセス許可ベクタを入れたバケットへのポ
 10135 インタ（静的APIを除く）

10136

10137 *アクセス許可ベクタ（バケットの内容）

10138 ACPTN acptn1 通常操作1のアクセス許可パターン

10139 ACPTN acptn2 通常操作2のアクセス許可パターン

10140 ACPTN acptn3 管理操作のアクセス許可パターン

10141 ACPTN acptn4 参照操作のアクセス許可パターン

10142

10143 **【リターンパラメータ】**

10144 ER ercd 正常終了 (E_OK) またはエラーコード

10145

10146 **【エラーコード】**

10147 E_CTX [s] コンテキストエラー（非タスクコンテキストからの呼出
 10148 し、CPUロック状態からの呼出し）

10149 E_ID 不正ID番号（mpfidが不正）

10150 E_RSATR 予約属性（属する保護ドメインかクラスが不正：SAC_MPF

10151 の場合)

10152 E_NOEXS [D] オブジェクト未登録 (対象固定長メモリプールが未登録)

10153 E_OACV [sP] オブジェクトアクセス違反 (対象固定長メモリプールに

10154 対する管理操作が許可されていない)

10155 E_MACV [sP] メモリアクセス違反 (p_acvctが指すメモリ領域への読出

10156 しアクセスが許可されていない)

10157 E_OBJ オブジェクト状態エラー (対象固定長メモリプールは静

10158 的APIで生成された:sac_mpfの場合, 対象固定長メモリ

10159 プールに対してアクセス許可ベクタが設定済み:SAC_MPF

10160 の場合)

10161

10162 **【機能】**

10163

10164 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

10165 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

10166 設定する. 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

10167 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

10168 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.

10169

10170 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

10171 式パラメータである.

10172

10173 SAC_MPFは, 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に記述し

10174 なければならない. そうでない場合には, E_RSATRエラーとなる.

10175

10176 **【TOPPERS/ASPカーネルにおける規定】**

10177

10178 ASPカーネルでは, SAC_MPF, sac_mpfをサポートしない.

10179

10180 **【TOPPERS/FMPカーネルにおける規定】**

10181

10182 FMPカーネルでは, SAC_MPF, sac_mpfをサポートしない.

10183

10184 **【TOPPERS/HRP2カーネルにおける規定】**

10185

10186 HRP2カーネルでは, SAC_MPFのみをサポートする.

10187 -----

10188 del_mpf 固定長メモリプールの削除 [TD]

10189

10190 **【C言語API】**

10191 ER ercd = del_mpf(ID mpfid)

10192

10193 **【パラメータ】**

10194 ID mpfid 対象固定長メモリプールのID番号

10195

10196 **【リターンパラメータ】**

10197 ER ercd 正常終了 (E_OK) またはエラーコード

10198

10199 **【エラーコード】**

10200 E_CTX コンテキストエラー (非タスクコンテキストからの呼出

10201 し、CPUロック状態からの呼出し)
 10202 E_ID 不正ID番号 (mpfidが不正)
 10203 E_NOEXS [D] オブジェクト未登録 (対象固定長メモリプールが未登録)
 10204 E_OACV [P] オブジェクトアクセス違反 (対象固定長メモリプールに
 10205 対する管理操作が許可されていない)
 10206 E_OBJ オブジェクト状態エラー (対象固定長メモリプールは静
 10207 的APIで生成された)
 10208

10209 【機能】

10210
 10211 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。
 10212 具体的な振舞いは以下の通り。

10213
 10214 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
 10215 用の状態に戻される。また、対象固定長メモリプールの待ち行列につながれた
 10216 タスクは、待ち行列の先頭のタスクから順に待ち解除される。待ち解除された
 10217 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。

10219 【使用上の注意】

10220
 10221 del_mpfにより複数のタスクが待ち解除される場合、サービスコールの処理時間
 10222 およびカーネル内での割込み禁止時間が、待ち解除されるタスクの数に比例し
 10223 て長くなる。特に、多くのタスクが待ち解除される場合、カーネル内での割込
 10224 み禁止時間が長くなるため、注意が必要である。

10226 【TOPPERS/ASPカーネルにおける規定】

10227
 10228 ASPカーネルでは、del_mpfをサポートしない。ただし、動的生成機能拡張パッ
 10229 ケージでは、del_mpfをサポートする。

10231 【TOPPERS/FMPカーネルにおける規定】

10232
 10233 FMPカーネルでは、del_mpfをサポートしない。

10235 【TOPPERS/HRP2カーネルにおける規定】

10236
 10237 HRP2カーネルでは、del_mpfをサポートしない。

10238 -----
 10239 get_mpf 固定長メモリブロックの獲得 [T]
 10240 pget_mpf 固定長メモリブロックの獲得 (ポーリング) [T]
 10241 tget_mpf 固定長メモリブロックの獲得 (タイムアウト付き) [T]
 10242

10243 【C言語API】

10244 ER ercd = get_mpf(ID mpfid, void **p_blk)
 10245 ER ercd = pget_mpf(ID mpfid, void **p_blk)
 10246 ER ercd = tget_mpf(ID mpfid, void **p_blk, TMO tmout)
 10247

10248 【パラメータ】

10249 ID mpfid 対象固定長メモリプールのID番号
 10250 void ** p_blk 獲得した固定長メモリブロックの先頭番地を入

10251			れるメモリ領域へのポインタ
10252	TMO	tmout	タイムアウト時間 (twai_mpfの場合)
10253			
10254	【リターンパラメータ】		
10255	ER	ercd	正常終了 (E_OK) またはエラーコード
10256	void *	blk	獲得した固定長メモリブロックの先頭番地
10257			
10258	【エラーコード】		
10259	E_CTX		コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し, ディスパッチ保留状態からの呼出し: pget_mpfを除く)
10260			
10261			
10262	E_NOSPT		未サポート機能 (制約タスクからの呼出し: pget_mpfを除く)
10263			
10264	E_ID		不正ID番号 (mpfidが不正)
10265	E_PAR		パラメータエラー (tmoutが不正: tget_mpfの場合)
10266	E_NOEXS [D]		オブジェクト未登録 (対象固定長メモリプールが未登録)
10267	E_OACV [P]		オブジェクトアクセス違反 (対象固定長メモリプールに対する通常操作2が許可されていない)
10268			
10269	E_MACV [P]		メモリアクセス違反 (p_blkが指すメモリ領域への読出しアクセスが許可されていない)
10270			
10271	E_TMOUT		ポーリング失敗またはタイムアウト (get_mpfを除く)
10272	E_RLWAI		待ち禁止状態または待ち状態の強制解除 (pget_mpfを除く)
10273			
10274	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)
10275			
10276			
10277	【機能】		
10278			
10279	mpfidで指定した固定長メモリプール (対象固定長メモリプール) から固定長メモリブロックを獲得し, その先頭番地をblkに返す. 具体的な振舞いは以下の通り.		
10280			
10281			
10282			
10283	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.		
10284			
10285			
10286			
10287	未割当てのメモリ領域がない場合には, 自タスクは固定長メモリプールの獲得待ち状態となり, 対象固定長メモリプールの待ち行列につながる.		
10288			
10289	-----		
10290	rel_mpf	固定長メモリブロックの返却 [T]	
10291			
10292	【C言語API】		
10293	ER	ercd = rel_mpf(ID mpfid, void *blk)	
10294			
10295	【パラメータ】		
10296	ID	mpfid	対象固定長メモリプールのID番号
10297	void *	blk	返却する固定長メモリブロックの先頭番地
10298			
10299	【リターンパラメータ】		
10300	ER	ercd	正常終了 (E_OK) またはエラーコード

10301

10302 **【エラーコード】**

10303 E_CTX コンテキストエラー（非タスクコンテキストからの呼出し、CPUロック状態からの呼出し）

10304

10305 E_ID 不正ID番号（mpfidが不正）

10306 E_PAR パラメータエラー（blkが不正）

10307 E_NOEXS [D] オブジェクト未登録（対象固定長メモリプールが未登録）

10308 E_OACV [P] オブジェクトアクセス違反（対象固定長メモリプールに対する通常操作1が許可されていない）

10309

10310

10311 **【機能】**

10312

10313 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．

10314

10315

10316 対象固定長メモリプールの待ち行列にタスクが存在する場合には、待ち行列の先頭のタスクが、blkで指定した固定長メモリブロックを獲得し、待ち解除される．待ち解除されたタスクには、待ち状態となったサービスコールからE_OKが返る．

10317

10318

10319

10320

10321 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．

10322

10323

10324 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．

10325

10326 -----

10327 ini_mpf 固定長メモリプールの再初期化 [T]

10328

10329 **【C言語API】**

10330 ER ercd = ini_mpf(ID mpfid)

10331

10332 **【パラメータ】**

10333 ID mpfid 対象固定長メモリプールのID番号

10334

10335 **【リターンパラメータ】**

10336 ER ercd 正常終了（E_OK）またはエラーコード

10337

10338 **【エラーコード】**

10339 E_CTX コンテキストエラー（非タスクコンテキストからの呼出し、CPUロック状態からの呼出し）

10340

10341 E_ID 不正ID番号（mpfidが不正）

10342 E_NOEXS [D] オブジェクト未登録（対象固定長メモリプールが未登録）

10343 E_OACV [P] オブジェクトアクセス違反（対象固定長メモリプールに対する管理操作が許可されていない）

10344

10345

10346 **【機能】**

10347

10348 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．

10349

10350

10351 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
 10352 れる。また、対象固定長メモリプールの待ち行列につながれたタスクは、待ち
 10353 行列の先頭のタスクから順に待ち解除される。待ち解除されたタスクには、待
 10354 ち状態となったサービスコールからE_DLTエラーが返る。

10355

10356 **【使用上の注意】**

10357

10358 ini_mpfにより複数のタスクが待ち解除される場合、サービスコールの処理時間
 10359 およびカーネル内での割込み禁止時間が、待ち解除されるタスクの数に比例し
 10360 て長くなる。特に、多くのタスクが待ち解除される場合、カーネル内での割込
 10361 み禁止時間が長くなるため、注意が必要である。

10362

10363 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
 10364 つのは、アプリケーションの責任である。

10365

10366 **【 μ ITRON4.0仕様との関係】**

10367

10368 μ ITRON4.0仕様に定義されていないサービスコールである。

10369

10370 ref_mpf 固定長メモリプールの状態参照 [T]

10371

10372 **【C言語API】**

10373 ER ercd = ref_mpf(ID mpfid, T_RMPF *pk_rmpf)

10374

10375 **【パラメータ】**

10376	ID	mpfid	対象固定長メモリプールのID番号
10377	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

10378

10379

10380 **【リターンパラメータ】**

10381	ER	ercd	正常終了 (E_OK) またはエラーコード
-------	----	------	-----------------------

10382

10383 * 固定長メモリプールの現在状態 (パケットの内容)

10384	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
-------	----	--------	--------------------------------

10386	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
-------	--------	---------	---

10388

10389

10390 **【エラーコード】**

10391	E_CTX	コンテキストエラー (非タスクコンテキストからの呼出 し, CPUロック状態からの呼出し)
10392		
10393	E_ID	不正ID番号 (mpfidが不正)
10394	E_NOEXS [D]	オブジェクト未登録 (対象固定長メモリプールが未登録)
10395	E_OACV [P]	オブジェクトアクセス違反 (対象固定長メモリプールに 対する参照操作が許可されていない)
10396		
10397	E_MACV [P]	メモリアクセス違反 (pk_rmpfが指すメモリ領域への書込 みアクセスが許可されていない)
10398		
10399		

10399

10400 **【機能】**

10401
10402 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
10403 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．
10404
10405 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
10406 TSK_NONE（=0）が返る．
10407
10408 **【使用上の注意】**
10409
10410 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
10411 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
10412 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
10413 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．
10414 -----
10415
10416 4.6 時間管理機能
10417
10418 4.6.1 システム時刻管理
10419
10420 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
10421 延、周期ハンドラの起動、アラームハンドラの起動に使用される時刻を管理す
10422 るカーネルオブジェクトである．システム時刻は、符号無しの整数型である
10423 SYSTIM型で表され、単位はミリ秒である．
10424
10425 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．タイムティックを通
10426 知するためのタイマ割込みが発生する毎にカーネルによって更新され、SYSTIM
10427 型で表せる最大値（ULONG_MAX）を超えると0に戻される．タイムティックの周
10428 期は、ターゲット定義である．また、システム時刻の精度はターゲットに依存
10429 する．
10430
10431 マルチプロセッサ対応でないカーネルと、マルチプロセッサ対応カーネルでグ
10432 ローバルタイマ方式を用いている場合には、システム時刻は、システムに1つの
10433 み存在する．マルチプロセッサ対応カーネルでローカルタイマ方式を用いてい
10434 る場合には、システム時刻は、プロセッサ毎に存在する．ローカルタイマ方式
10435 とグローバルタイマ方式については、「2.3.4 マルチプロセッサ対応」の節を
10436 参照すること．
10437
10438 マルチプロセッサ対応カーネルでローカルタイマ方式を用いている場合には、
10439 タイムアウト処理とタスクの遅延処理には、待ち解除されるタスクが割り付け
10440 られているプロセッサのシステム時刻が用いられる．また、周期ハンドラとア
10441 ラームハンドラの起動には、それが割り付けられているプロセッサのシステム
10442 時刻が用いられる．これらの処理単位がマイグレーションする場合には、用い
10443 られるシステム時刻も変更される．この場合にも、イベントの処理が行われる
10444 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
10445 という原則は維持される．
10446
10447 1回のタイムティックの発生により、複数のイベントの処理を行うべき状況になっ
10448 た場合、それらの処理の間の処理順序は規定されない．
10449
10450 性能評価用システム時刻は、性能評価に使用することを目的とした、システム

時刻よりも精度の高い時刻である。性能評価用システム時刻は、符号無しの整数型であるSYSUTM型で表され、単位はマイクロ秒である。ただし、実際の精度はターゲットに依存する。

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

システム時刻管理機能に関連するカーネル構成マクロは次の通り。

TIC_NUME	タイムティックの周期（単位はミリ秒）の分子
TIC_DEN0	タイムティックの周期（単位はミリ秒）の分母
TOPPERS_SUPPORT_GET_UTM	get_utmがサポートされている

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、時間管理機能をサポートしない。

【使用上の注意】

タイムティックを通知するためのタイマ割込みが長時間マスクされた場合（タイマ割込みより優先して実行される割込み処理が長時間続けて実行された場合を含む）や、シミュレーション環境においてシミュレータのプロセスが長時間スケジュールされなかった場合には、システム時刻が正しく更新されない可能性があるため、注意が必要である。

【 μ ITRON4.0仕様との関係】

システム時刻を設定するサービスコール（set_tim）を廃止した。また、タイムティックを供給する機能は、カーネル内に実現することとし、そのためのサービスコール（isig_tim）は廃止した。

【 μ ITRON4.0/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）は廃止した。

get_tim システム時刻の参照 [T]

【C言語API】

```
ER ercd = get_tim(SYSTIM *p_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

10501 **【エラーコード】**

10502 E_CTX コンテキストエラー（非タスクコンテキストからの呼出し、CPUロック状態からの呼出し）

10503

10504 E_OACV [P] オブジェクトアクセス違反（システム状態に対する参照操作が許可されていない）

10505

10506 E_MACV [P] メモリアクセス違反（p_systimが指すメモリ領域への書き込みアクセスが許可されていない）

10507

10508

10509 **【機能】**

10510

10511 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で指定したメモリ領域に返される。

10512

10513

10514 マルチプロセッサ対応カーネルでローカルタイマ方式を用いている場合には、

10515 自タスクが割り付けられているプロセッサのシステム時刻の現在値を参照する。

10516

10517 **【補足説明】**

10518

10519 マルチプロセッサ対応カーネルでローカルタイマ方式を用いている場合に、他のプロセッサのシステム時刻の現在値を参照する機能は用意していない。

10520

10521 -----

10522 get_utm 性能評価用システム時刻の参照 [TI]

10523

10524 **【C言語API】**

10525 ER ercd = get_utm(SYSUTM *p_sysutm)

10526

10527 **【パラメータ】**

10528 SYSUTM * p_sysutm 性能評価用システム時刻を入れるメモリ領域へのポインタ

10529

10530

10531 **【リターンパラメータ】**

10532 ER ercd 正常終了（E_OK）またはエラーコード

10533 SYSUTM sysutm 性能評価用システム時刻の現在値

10534

10535 **【エラーコード】**

10536 E_NOSPT 未サポート機能（get_utmがサポートされていない）

10537 E_MACV [P] メモリアクセス違反（p_sysutmが指すメモリ領域へ書き込みアクセスが許可されていない）

10538

10539

10540 **【機能】**

10541

10542 性能評価用システム時刻の現在値を参照する。参照した性能評価用システム時刻は、p_sysutmで指定したメモリ領域に返される。

10543

10544

10545 get_utmは、任意の状態から呼び出すことができる。タスクコンテキストからも非タスクコンテキストからも呼び出すことができるし、CPUロック状態であっても呼び出すことができる。

10546

10547

10548

10549 ターゲット定義で、get_utmがサポートされていない場合がある。get_utmがサポートされている場合には、TOPPERS_SUPPORT_GET_UTMがマクロ定義される。サ

10550

10551 ポートされていない場合にget_utmを呼び出すと、E_NOSPTエラーが返るか、リ
10552 ンク時にエラーとなる。

10553

10554 **【使用方法】**

10555

10556 get_utmを使用してプログラムの処理時間を計測する場合には、次の手順を取る。
10557 処理時間を計測したいプログラムの実行直前と実行直後に、get_utmを用いて性
10558 能評価用システム時刻を読み出す。その差を求めることで、対象プログラムの
10559 処理時間に、get_utm自身の処理時間を加えたものが得られる。

10560

10561 マルチプロセッサ対応カーネルにおいては、異なるプロセッサで読み出した性
10562 能評価用システム時刻の差を求めることで、処理時間が正しく計測できるとは
10563 限らない。

10564

10565 **【使用上の注意】**

10566

10567 get_utmは性能評価のための機能であり、その他の目的に使用することは推奨し
10568 ない。

10569

10570 get_utmは、任意の状態から呼び出すことができるように、全割込みロック状態
10571 を用いて実装されている。そのため、get_utmを用いると、カーネル管理外の割
10572 込みの応答性が低下する。

10573

10574 システム時刻が正しく更新されない状況では、get_utmは誤った性能評価用シス
10575 テム時刻を返す可能性がある。システム時刻の更新が確実に行われることを保
10576 証できない場合には、get_utmが誤った性能評価用システム時刻を返す可能性を
10577 考慮に入れて使用しなければならない。

10578

10579 **【 μ ITRON4.0仕様との関係】**

10580

10581 μ ITRON4.0仕様に定義されていないサービスコールである。

10582 -----

10583

10584 4.6.2 周期ハンドラ

10585

10586 周期ハンドラは、指定した周期で起動されるタイムイベントハンドラである。

10587 周期ハンドラは、周期ハンドラIDと呼ぶID番号によって識別する。

10588

10589 各周期ハンドラが持つ情報は次の通り。

10590

- 10591 ・ 周期ハンドラ属性
- 10592 ・ 周期ハンドラの動作状態
- 10593 ・ 次に周期ハンドラを起動する時刻
- 10594 ・ 拡張情報
- 10595 ・ 周期ハンドラ先の頭番地
- 10596 ・ 起動周期
- 10597 ・ 起動位相
- 10598 ・ アクセス許可ベクタ（保護機能対応カーネルの場合）
- 10599 ・ 属する保護ドメイン（保護機能対応カーネルの場合）
- 10600 ・ 属するクラス（マルチプロセッサ対応カーネルの場合）

10601
10602 周期ハンドラの起動時刻は、後述する基準時刻から、以下の式で求められる相
10603 対時間後である。
10604
10605
$$\text{起動位相} + \text{起動周期} \times (n-1) \quad n=1, 2, \dots$$

10606
10607 周期ハンドラの動作状態は、動作している状態と動作していない状態のいずれ
10608 かをとる。周期ハンドラを動作している状態にすることを動作開始、動作して
10609 いない状態にすることを動作停止という。
10610
10611 周期ハンドラが動作している状態の場合には、周期ハンドラを起動する時刻に
10612 になると、周期ハンドラの起動処理が行われる。具体的には、拡張情報をパラメー
10613 タとして、周期ハンドラが呼び出される。
10614
10615 保護機能対応カーネルにおいて、周期ハンドラが属することのできる保護ドメ
10616 インは、カーネルドメインに限られる。
10617
10618 周期ハンドラ属性には、次の属性を指定することができる。
10619
10620 TA_STA 0x02U 周期ハンドラの生成時に周期ハンドラを動作開始する
10621 TA_PHS 0x04U 周期ハンドラを生成した時刻を基準時刻とする
10622
10623 TA_STAを指定しない場合、周期ハンドラの生成直後には、周期ハンドラは動作
10624 していない状態となる。
10625
10626 TA_PHSを指定しない場合には、周期ハンドラを動作開始した時刻が、周期ハン
10627 ドラを起動する時刻の基準時刻となる。TA_PHSを指定した場合には、周期ハン
10628 ドラを生成した時刻（静的APIで生成した場合にはカーネルの起動時刻）が、基
10629 準時刻となる。
10630
10631 次に周期ハンドラを起動する時刻は、周期ハンドラが動作している状態でのみ
10632 有効で、必要に応じて、カーネルの起動時、周期ハンドラの動作開始時、周期
10633 ハンドラの起動処理時に設定される。
10634
10635 マルチプロセッサ対応カーネルでグローバルタイマ方式を用いている場合には、
10636 周期ハンドラは、システム時刻管理プロセッサのみが割付け可能プロセッサで
10637 あるクラスにのみ属することができる。すなわち、周期ハンドラは、システム
10638 時刻管理プロセッサによって実行される。
10639
10640 C言語による周期ハンドラの記述形式は次の通り。
10641
10642

```
void cyclic_handler(intptr_t exinf)
{
    周期ハンドラ本体
}
```


10643
10644
10645
10646
10647 exinfには、周期ハンドラの拡張情報が渡される。
10648
10649 周期ハンドラ機能に関連するカーネル構成マクロは次の通り。
10650

10651 TNUM_CYCID 登録できる周期ハンドラの数（動的生成対応でないカー
 10652 ネルでは、静的APIによって登録された周期ハンドラの数
 10653 に一致）
 10654
 10655 **【TOPPERS/ASPカーネルにおける規定】**
 10656
 10657 ASPカーネルでは、TA_PHS属性の周期ハンドラをサポートしない。
 10658
 10659 **【TOPPERS/FMPカーネルにおける規定】**
 10660
 10661 FMPカーネルでは、TA_PHS属性の周期ハンドラをサポートしない。
 10662
 10663 **【TOPPERS/HRP2カーネルにおける規定】**
 10664
 10665 HRP2カーネルでは、TA_PHS属性の周期ハンドラをサポートしない。
 10666
 10667 **【 μ ITRON4.0仕様との関係】**
 10668
 10669 TNUM_CYCIDは、 μ ITRON4.0仕様に規定されていないカーネル構成マクロである。
 10670 -----
 10671 CRE_CYC 周期ハンドラの生成 [S]
 10672 acre_cyc 周期ハンドラの生成 [TD]
 10673
 10674 **【静的API】**
 10675 CRE_CYC(ID cycled, { ATR cycatr, intptr_t exinf, CYCHDR cychdr,
 10676 RELTIM cycetim, RELTIM cycphs })
 10677
 10678 **【C言語API】**
 10679 ER_ID cycled = acre_cyc(const T_CCYC *pk_ccyc)
 10680
 10681 **【パラメータ】**
 10682 ID cycled 生成する周期ハンドラのID番号（CRE_CYCの場合）
 10683 T_CCYC * pk_ccyc 周期ハンドラの生成情報を入れたパケットへの
 10684 ポインタ（静的APIを除く）
 10685
 10686 *周期ハンドラの生成情報（パケットの内容）
 10687 ATR cycatr 周期ハンドラ属性
 10688 intptr_t exinf 周期ハンドラの拡張情報
 10689 CYCHDR cychdr 周期ハンドラ先頭の番地
 10690 RELTIM cycetim 周期ハンドラの起動周期
 10691 RELTIM cycphs 周期ハンドラの起動位相
 10692
 10693 **【リターンパラメータ】**
 10694 ER_ID cycled 生成された周期ハンドラのID番号（正の値）また
 10695 はエラーコード
 10696
 10697 **【エラーコード】**
 10698 E_CTX [s] コンテキストエラー（非タスクコンテキストからの呼出
 10699 し、CPUロック状態からの呼出し）
 10700 E_RSATR 予約属性（cycatrが不正または使用できない、属する保

10701		護ドメインかクラスが不正)
10702	E_PAR	パラメータエラー (cychdr, cyctim, cycphsが不正)
10703	E_OACV [sP]	オブジェクトアクセス違反 (システム状態に対する管理
10704		操作が許可されていない)
10705	E_MACV [sP]	メモリアクセス違反 (pk_ccycが指すメモリ領域への読出
10706		しアクセスが許可されていない)
10707	E_NOID [sD]	ID番号不足 (割り付けられる周期ハンドラIDがない)
10708	E_OBJ	オブジェクト状態エラー (cycidで指定した周期ハンドラ
10709		が登録済み: CRE_CYCの場合)
10710		

【機能】

10711
10712
10713 各パラメータで指定した周期ハンドラ生成情報に従って、周期ハンドラを生成
10714 する。具体的な振舞いは以下の通り。
10715

10716 cycattrにTA_STAを指定した場合、対象周期ハンドラは動作している状態となる。
10717 次に周期ハンドラを起動する時刻は、サービスコールを呼び出した時刻 (静的
10718 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
10719 れる。

10720
10721 cycattrにTA_STAを指定しない場合、対象周期ハンドラは動作していない状態に
10722 初期化される。
10723

10724 静的APIにおいては、cycidはオブジェクト識別名、cycattr, cyctim, cycphsは
10725 整数定数式パラメータ、exinfとcychdrは一般定数式パラメータである。
10726

10727 保護機能対応カーネルにおいて、CRE_CYCは、カーネルドメインの囲みの中に記
10728 述しなければならない。そうでない場合には、E_RSATRエラーとなる。また、
10729 acre_cycで、生成する周期ハンドラが属する保護ドメインとしてカーネルドメ
10730 イン以外を指定した場合には、E_RSATRエラーとなる。
10731

10732 cyctimは、0より大きく、TMAX_RELTIM以下の値でなければならない。また、
10733 cycphsは、TMAX_RELTIM以下でなければならない。cycphsにcyctimより大きい値
10734 を指定してもよい。
10735

10736 マルチプロセッサ対応カーネルでグローバルタイマ方式を用いている場合で、
10737 生成する周期ハンドラの属するクラスの割付け可能プロセッサが、システム時
10738 刻管理プロセッサのみでない場合には、E_RSATRエラーとなる。
10739

【補足説明】

10740
10741
10742 静的APIにおいて、cycattrにTA_STAを、cycphsに0を指定した場合、周期ハンド
10743 ラが最初に呼び出されるのは、カーネル起動後最初のタイムティックになる。
10744 cycphsに1を指定した場合も同じ振舞いとなるため、静的APIでcycattrにTA_STA
10745 が指定されている場合には、cycphsに0を指定することは推奨されず、コンフィ
10746 ギュレータが警告メッセージを出力する。
10747

【TOPPERS/ASPカーネルにおける規定】

10748
10749
10750 ASPカーネルでは、CRE_CYCのみをサポートする。ただし、TA_PHS属性の周期ハ

10751 シンドラはサポートしない。動的生成機能拡張パッケージでは、acre_cycもサポー
10752 トする。

10753

10754 **【TOPPERS/FMPカーネルにおける規定】**

10755

10756 FMPカーネルでは、CRE_CYCのみをサポートする。ただし、TA_PHS属性の周期ハ
10757 シンドラはサポートしない。

10758

10759 **【TOPPERS/HRP2カーネルにおける規定】**

10760

10761 HRP2カーネルでは、CRE_CYCのみをサポートする。ただし、TA_PHS属性の周期ハ
10762 シンドラはサポートしない。

10763

10764 **【 μ ITRON4.0仕様との関係】**

10765

10766 cycchrのデータ型をCYCHDRに変更した。また、cycphsにcycctimより大きい値を
10767 指定した場合の振舞いと、静的APIでcycphsに0を指定した場合の振舞いを規定
10768 した。

10769

10770 AID_CYC 割付け可能な周期ハンドラIDの数の指定 [SD]

10771

10772 **【静的API】**

10773 AID_CYC(uint_t nocyc)

10774

10775 **【パラメータ】**

10776 uint_t nocyc 割付け可能な周期ハンドラIDの数

10777

10778 **【エラーコード】**

10779 E_RSATR 予約属性（属する保護ドメインまたはクラスが不正）

10780

10781 **【機能】**

10782

10783 nocycで指定した数の周期ハンドラIDを、周期ハンドラを生成するサービスコー
10784 ルによって割付け可能な周期ハンドラIDとして確保する。

10785

10786 nocycは整数定数式パラメータである。

10787

10788 SAC_CYC 周期ハンドラのアクセス許可ベクタの設定 [SP]

10789 sac_cyc 周期ハンドラのアクセス許可ベクタの設定 [TPD]

10790

10791 **【静的API】**

10792 SAC_CYC(ID cycid, { ACPTN acptn1, ACPTN acptn2,
10793 ACPTN acptn3, ACPTN acptn4 })

10794

10795 **【C言語API】**

10796 ER ercd = sac_cyc(ID cycid, const ACVCT *p_acvct)

10797

10798 **【パラメータ】**

10799 ID cycid 対象周期ハンドラのID番号

10800 ACVCT * p_acvct アクセス許可ベクタを入れたパケットへのポ

217

10851 del_cyc 周期ハンドラの削除 [TD]

10852

10853 **【C言語API】**

10854 ER ercd = del_cyc(ID cycid)

10855

10856 **【パラメータ】**

10857 ID cycid 対象周期ハンドラのID番号

10858

10859 **【リターンパラメータ】**

10860 ER ercd 正常終了 (E_OK) またはエラーコード

10861

10862 **【エラーコード】**

10863 E_CTX コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)

10864

10865 E_ID 不正ID番号 (cycidが不正)

10866 E_NOEXS [D] オブジェクト未登録 (対象周期ハンドラが未登録)

10867 E_OACV [P] オブジェクトアクセス違反 (対象周期ハンドラに対する管理操作が許可されていない)

10868

10869 E_OBJ オブジェクト状態エラー (対象周期ハンドラは静的APIで生成された)

10870

10871

10872 **【機能】**

10873

10874 cycidで指定した周期ハンドラ (対象周期ハンドラ) を削除する. 具体的な振舞いは以下の通り.

10875

10876

10877 対象周期ハンドラの登録が解除され, その周期ハンドラIDが未使用の状態に戻る. 対象周期ハンドラが動作している状態であった場合には, 動作していない状態にされた後に, 登録が解除される.

10878

10879

10880

10881 **【TOPPERS/ASPカーネルにおける規定】**

10882

10883 ASPカーネルでは, del_cycをサポートしない. ただし, 動的生成機能拡張パッケージでは, del_cycをサポートする.

10884

10885

10886 **【TOPPERS/FMPカーネルにおける規定】**

10887

10888 FMPカーネルでは, del_cycをサポートしない.

10889

10890 **【TOPPERS/HRP2カーネルにおける規定】**

10891

10892 HRP2カーネルでは, del_cycをサポートしない.

10893 -----

10894 sta_cyc 周期ハンドラの動作開始 [T]

10895

10896 **【C言語API】**

10897 ER ercd = sta_cyc(ID cycid)

10898

10899 **【パラメータ】**

10900 ID cycid 対象周期ハンドラのID番号

10901

10902 **【リターンパラメータ】**

10903 ER ercd 正常終了 (E_OK) またはエラーコード

10904

10905 **【エラーコード】**

10906 E_CTX コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)

10907

10908 E_ID 不正ID番号 (cycidが不正)

10909 E_NOEXS [D] オブジェクト未登録 (対象周期ハンドラが未登録)

10910 E_OACV [P] オブジェクトアクセス違反 (対象周期ハンドラに対する通常操作1が許可されていない)

10911

10912

10913 **【機能】**

10914

10915 cycidで指定した周期ハンドラ (対象周期ハンドラ) を動作開始する. 具体的な

10916 振舞いは以下の通り.

10917

10918 対象周期ハンドラが動作していない状態であれば, 対象周期ハンドラは動作し

10919 ている状態となる. 次に周期ハンドラを起動する時刻は, sta_cycを呼び出して

10920 以降の最初の起動時刻に設定される.

10921

10922 対象周期ハンドラが動作している状態であれば, 次に周期ハンドラを起動する

10923 時刻の再設定のみが行われる.

10924

10925 **【補足説明】**

10926

10927 TA_PHS属性でない周期ハンドラの場合, 次に周期ハンドラを起動する時刻は,

10928 sta_cycを呼び出してから, 対象周期ハンドラの起動位相で指定した相対時間後

10929 に設定される.

10930

10931 対象周期ハンドラがTA_PHS属性で, 動作している状態であれば, 次に周期ハン

10932 ドラを起動する時刻は変化しない.

10933

10934 **【μITRON4.0仕様との関係】**

10935

10936 TA_PHS属性でない周期ハンドラにおいて, sta_cycを呼び出した後, 最初に周期

10937 ハンドラが起動される時刻を変更した. μITRON4.0仕様では, sta_cycを呼び出

10938 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

10939 仕様では, 起動位相で指定した相対時間後とした.

10940 -----

10941 msta_cyc 割付けプロセッサ指定での周期ハンドラの動作開始 [TM]

10942

10943 **【C言語API】**

10944 ER ercd = msta_cyc(ID cycid, ID prcid)

10945

10946 **【パラメータ】**

10947 ID cycid 対象周期ハンドラのID番号

10948 ID prcid 周期ハンドラの割付け対象のプロセッサのID番号

10949

10950 **【リターンパラメータ】**

10951	ER	ercd	正常終了 (E_OK) またはエラーコード
10952			
10953	【エラーコード】		
10954	E_CTX		コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
10955			
10956	E_NOSPT		未サポート機能 (グローバルタイマ方式を用いている場合)
10957			
10958	E_ID		不正ID番号 (cycid, prcidが不正)
10959	E_PAR		パラメータエラー (対象周期ハンドラはprcidで指定したプロセッサに割り付けられない)
10960			
10961	E_NOEXS [D]		オブジェクト未登録 (対象周期ハンドラが未登録)
10962	E_OACV [P]		オブジェクトアクセス違反 (対象周期ハンドラに対する通常操作1が許可されていない)
10963			
10964			
10965	【機能】		
10966			
10967	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		
10968	ハンドラ (対象周期ハンドラ) を動作開始する. 具体的な振舞いは以下の通り.		
10969			
10970	対象周期ハンドラが動作していない状態であれば, 対象周期ハンドラの割付け		
10971	プロセッサがprcidで指定したプロセッサに変更された後, 対象周期ハンドラは		
10972	動作している状態となる. 次に周期ハンドラを起動する時刻は, msta_cycを呼		
10973	び出して以降の最初の起動時刻に設定される.		
10974			
10975	対象周期ハンドラが動作している状態であれば, 対象周期ハンドラの割付けプ		
10976	ロセッサがprcidで指定したプロセッサに変更された後, 次に周期ハンドラを起		
10977	動する時刻の再設定が行われる.		
10978			
10979	対象周期ハンドラが実行中である場合には, 割付けプロセッサを変更しても,		
10980	実行中の周期ハンドラを実行するプロセッサは変更されない. 対象周期ハンド		
10981	ラが変更後の割付けプロセッサで実行されるのは, 次に起動される時からであ		
10982	る.		
10983			
10984	対象周期ハンドラの属するクラスの割付け可能プロセッサが, prcidで指定した		
10985	プロセッサを含んでいない場合には, E_PARエラーとなる.		
10986			
10987	prcidにTPRC_INI (=0) を指定すると, 対象周期ハンドラの割付けプロセッサ		
10988	を, それが属するクラスの初期割付けプロセッサとする.		
10989			
10990	グローバルタイマ方式を用いている場合, msta_cycはE_NOSPTを返す.		
10991			
10992	【補足説明】		
10993			
10994	TA_PHS属性でない周期ハンドラの場合, 次に周期ハンドラを起動する時刻は,		
10995	msta_cycを呼び出してから, 対象周期ハンドラの起動位相で指定した相対時間		
10996	後に設定される.		
10997			
10998	【使用上の注意】		
10999			
11000	msta_cycで実行中の周期ハンドラの割付けプロセッサを変更した場合, 同じ周		

11001 期ハンドラが異なるプロセッサで同時に実行される可能性がある。特に、対象
 11002 周期ハンドラの起動位相が0の場合に、注意が必要である。

11003

11004 **【 μ ITRON4.0仕様との関係】**

11005

11006 μ ITRON4.0仕様に定義されていないサービスコールである。

11007 -----

11008 stp_cyc 周期ハンドラの動作停止 [T]

11009

11010 **【C言語API】**

11011 ER ercd = stp_cyc(ID cycid)

11012

11013 **【パラメータ】**

11014 ID cycid 対象周期ハンドラのID番号

11015

11016 **【リターンパラメータ】**

11017 ER ercd 正常終了 (E_OK) またはエラーコード

11018

11019 **【エラーコード】**

11020 E_CTX コンテキストエラー (非タスクコンテキストからの呼出し、CPUロック状態からの呼出し)

11021

11022 E_ID 不正ID番号 (cycidが不正)

11023 E_NOEXS [D] オブジェクト未登録 (対象周期ハンドラが未登録)

11024 E_OACV [P] オブジェクトアクセス違反 (対象周期ハンドラに対する

11025 通常操作2が許可されていない)

11026

11027 **【機能】**

11028

11029 cycidで指定した周期ハンドラ (対象周期ハンドラ) を動作停止する。具体的な
 11030 振舞いは以下の通り。

11031

11032 対象周期ハンドラが動作している状態であれば、動作していない状態になる。

11033 対象周期ハンドラが動作していない状態であれば、何も行われずに正常終了す

11034 る。

11035 -----

11036 ref_cyc 周期ハンドラの状態参照 [T]

11037

11038 **【C言語API】**

11039 ER ercd = ref_cyc(ID cycid, T_RCYC *pk_rcyc)

11040

11041 **【パラメータ】**

11042 ID cycid 対象周期ハンドラのID番号

11043 T_RCYC * pk_rcyc 周期ハンドラの現在状態を入れるパケットへの
 11044 ポインタ

11045

11046 **【リターンパラメータ】**

11047 ER ercd 正常終了 (E_OK) またはエラーコード

11048

11049 *周期ハンドラの現在状態 (パケットの内容)

11050 STAT cycstat 周期ハンドラの動作状態

11051 RELTIM lefttim 次に周期ハンドラを起動する時刻までの相対時間
 11052 ID prcid 周期ハンドラの割付けプロセッサのID (マルチプ
 11053 ロセッサ対応カーネルの場合)
 11054

【エラーコード】

11056 E_CTX コンテキストエラー (非タスクコンテキストからの呼出
 11057 し, CPUロック状態からの呼出し)
 11058 E_ID 不正ID番号 (cycidが不正)
 11059 E_NOEXS [D] オブジェクト未登録 (対象周期ハンドラが未登録)
 11060 E_OACV [P] オブジェクトアクセス違反 (対象周期ハンドラに対する
 11061 参照操作が許可されていない)
 11062 E_MACV [P] メモリアクセス違反 (pk_rcycが指すメモリ領域への書込
 11063 みアクセスが許可されていない)
 11064

【機能】

11065
 11066
 11067 cycidで指定した周期ハンドラ (対象周期ハンドラ) の現在状態を参照する. 参
 11068 照した現在状態は, pk_rcycで指定したパケットに返される.
 11069

11070 cycstatには, 対象周期ハンドラの現在の動作状態を表す次のいずれかの値が返
 11071 される.
 11072

11073 TCYC_STP 0x01U 周期ハンドラが動作していない状態
 11074 TCYC_STA 0x02U 周期ハンドラが動作している状態
 11075

11076 対象周期ハンドラが動作している状態である場合には, lefttimに, 次に周期ハ
 11077 ンドラ起動する時刻までの相対時間が返される. 対象周期ハンドラが動作して
 11078 いない状態である場合には, lefttimの値は保証されない.
 11079

11080 マルチプロセッサ対応カーネルでは, prcidに, 対象周期ハンドラの割付けプロ
 11081 セッサのID番号が返される.
 11082

【使用上の注意】

11083
 11084
 11085 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
 11086 ない. これは, ref_cycを呼び出し, 対象周期ハンドラの現在状態を参照した直
 11087 後に割込みが発生した場合, ref_cycから戻ってきた時には対象周期ハンドラの
 11088 状態が変化している可能性があるためである.
 11089

【μITRON4.0仕様との関係】

11090 TCYC_STPとTCYC_STAを値を変更した.
 11091
 11092
 11093 -----
 11094

4.6.3 アラームハンドラ

11095
 11096
 11097 アラームハンドラは, 指定した相対時間後に起動されるタイムイベントハンド
 11098 ラである. アラームハンドラは, アラームハンドラIDと呼ぶID番号によって識
 11099 別する.
 11100

11101 各アラームハンドラが持つ情報は次の通り.

11102

- 11103 • アラームハンドラ属性
- 11104 • アラームハンドラの動作状態
- 11105 • アラームハンドラを起動する時刻
- 11106 • 拡張情報
- 11107 • アラームハンドラ先頭番地
- 11108 • アクセス許可ベクタ (保護機能対応カーネルの場合)
- 11109 • 属する保護ドメイン (保護機能対応カーネルの場合)
- 11110 • 属するクラス (マルチプロセッサ対応カーネルの場合)

11111

11112 アラームハンドラの動作状態は、動作している状態と動作していない状態のい
11113 ずれかをとる. アラームハンドラを動作している状態にすることを動作開始、
11114 動作していない状態にすることを動作停止という.

11115

11116 アラームハンドラを起動する時刻は、アラームハンドラを動作開始する時に設
11117 定される.

11118

11119 アラームハンドラが動作している状態の場合には、アラームハンドラを起動す
11120 る時刻になると、アラームハンドラの起動処理が行われる. 具体的には、まず、
11121 アラームハンドラが動作していない状態にされる. その後、拡張情報をパラ
11122 メータとして、アラームハンドラが呼び出される.

11123

11124 保護機能対応カーネルにおいて、アラームハンドラが属することのできる保護
11125 ドメインは、カーネルドメインに限られる.

11126

11127 マルチプロセッサ対応カーネルでグローバルタイマ方式を用いている場合には、
11128 アラームハンドラは、割付け可能プロセッサがシステム時刻管理プロセッサの
11129 みであるクラスにのみ属することができる. すなわち、アラームハンドラは、
11130 システム時刻管理プロセッサによって実行される.

11131

11132 C言語によるアラームハンドラの記述形式は次の通り.

11133

```
11134     void alarm_handler(intptr_t exinf)
11135     {
11136         アラームハンドラ本体
11137     }
```

11138

11139 exinfには、アラームハンドラの拡張情報が渡される.

11140

11141 アラームハンドラ機能に関連するカーネル構成マクロは次の通り.

11142

11143	TNUM_ALMID	登録できるアラームハンドラの数 (動的生成対応でない
11144		カーネルでは、静的APIによって登録されたアラームハン
11145		ドラの数に一致)

11146

11147 【 μ ITRON4.0仕様との関係】

11148

11149 TNUM_ALMIDは、 μ ITRON4.0仕様に規定されていないカーネル構成マクロである.

11150

11151 CRE_ALM アラームハンドラの生成 [S]
 11152 acre_alm アラームハンドラの生成 [TD]
 11153
 11154 **【静的API】**
 11155 CRE_ALM(ID almid, { ATR almatr, intptr_t exinf, ALMHDR almhdr })
 11156
 11157 **【C言語API】**
 11158 ER_ID almid = acre_alm(const T_CALM *pk_calm)
 11159
 11160 **【パラメータ】**
 11161 ID almid 生成するアラームハンドラのID番号 (CRE_ALM
 11162 の場合)
 11163 T_CALM * pk_calm アラームハンドラの生成情報を入れたパケット
 11164 へのポインタ (静的APIを除く)
 11165
 11166 *アラームハンドラの生成情報 (パケットの内容)
 11167 ATR almatr アラームハンドラ属性
 11168 intptr_t exinf アラームハンドラの拡張情報
 11169 ALMHDR almhdr アラームハンドラの先頭番地
 11170
 11171 **【リターンパラメータ】**
 11172 ER_ID almid 生成されたアラームハンドラのID番号 (正の値)
 11173 またはエラーコード
 11174
 11175 **【エラーコード】**
 11176 E_CTX [s] コンテキストエラー (非タスクコンテキストからの呼出し,
 11177 CPUロック状態からの呼出し)
 11178 E_RSATR 予約属性 (almatrが不正または使用できない, 属する保
 11179 護ドメインかクラスが不正)
 11180 E_PAR パラメータエラー (almhdrが不正)
 11181 E_OACV [sP] オブジェクトアクセス違反 (システム状態に対する管理
 11182 操作が許可されていない)
 11183 E_MACV [sP] メモリアクセス違反 (pk_calmが指すメモリ領域への読出
 11184 しアクセスが許可されていない)
 11185 E_NOID [sD] ID番号不足 (割り付けられるアラームハンドラIDがない)
 11186 E_OBJ オブジェクト状態エラー (almidで指定したアラームハン
 11187 ドラが登録済み: CRE_ALMの場合)
 11188
 11189 **【機能】**
 11190
 11191 各パラメータで指定したアラームハンドラ生成情報に従って, アラームハンド
 11192 ラを生成する. 対象アラームハンドラは, 動作していない状態に初期化される.
 11193
 11194 静的APIにおいては, almidはオブジェクト識別名, almatrは整数定数式パラメー
 11195 タ, exinfとalmhdrは一般定数式パラメータである.
 11196
 11197 保護機能対応カーネルにおいて, CRE_ALMは, カーネルドメインの囲みの中に記
 11198 述しなければならない. そうでない場合には, E_RSATRエラーとなる. また,
 11199 acre_almで, 生成するアラームハンドラが属する保護ドメインとしてカーネル
 11200 ドメイン以外を指定した場合には, E_RSATRエラーとなる.


```

11201 マルチプロセッサ対応カーネルでグローバルタイマ方式を用いている場合で、
11202 生成するアラームハンドラの属するクラスの割付け可能プロセッサが、システ
11203 ム時刻管理プロセッサのみでない場合には、E_RSATRエラーとなる。
11204
11205 【TOPPERS/ASPカーネルにおける規定】
11206
11207 ASPカーネルでは、CRE_ALMのみをサポートする。ただし、動的生成機能拡張パッ
11208 ケージでは、acre_almもサポートする。
11209
11210 【TOPPERS/FMPカーネルにおける規定】
11211
11212 FMPカーネルでは、CRE_ALMのみをサポートする。
11213
11214 【TOPPERS/HRP2カーネルにおける規定】
11215
11216 HRP2カーネルでは、CRE_ALMのみをサポートする。
11217
11218 【μITRON4.0仕様との関係】
11219
11220 almhdrのデータ型をALMHDRに変更した。
11221 -----
11222 AID_ALM      割付け可能なアラームハンドラIDの数の指定 [SD]
11223
11224 【静的API】
11225     AID_ALM(uint_t noalm)
11226
11227 【パラメータ】
11228     uint_t      noalm      割付け可能なアラームハンドラIDの数
11229
11230 【エラーコード】
11231     E_RSATR      予約属性（属する保護ドメインまたはクラスが不正）
11232
11233 【機能】
11234
11235 noalmで指定した数のアラームハンドラIDを、アラームハンドラを生成するサー
11236 ビスコールによって割付け可能なアラームハンドラIDとして確保する。
11237
11238 noalmは整数定数式パラメータである。
11239 -----
11240 SAC_ALM      アラームハンドラのアクセス許可ベクタの設定 [SP]
11241 sac_alm      アラームハンドラのアクセス許可ベクタの設定 [TPD]
11242
11243 【静的API】
11244     SAC_ALM(ID almid, { ACPTN acptn1, ACPTN acptn2,
11245                          ACPTN acptn3, ACPTN acptn4 })
11246
11247 【C言語API】
11248     ER ercd = sac_alm(ID almid, const ACVCT *p_acvct)
11249
11250

```

11251 **【パラメータ】**

11252	ID	almid	対象アラームハンドラのID番号
11253	ACVCT *	p_acvct	アクセス許可ベクタを入れたパケットへのポインタ（静的APIを除く）

11255

11256 *アクセス許可ベクタ（パケットの内容）

11257	ACPTN	acptn1	通常操作1のアクセス許可パターン
11258	ACPTN	acptn2	通常操作2のアクセス許可パターン
11259	ACPTN	acptn3	管理操作のアクセス許可パターン
11260	ACPTN	acptn4	参照操作のアクセス許可パターン

11261

11262 **【リターンパラメータ】**

11263	ER	ercd	正常終了（E_OK）またはエラーコード
-------	----	------	---------------------

11264

11265 **【エラーコード】**

11266	E_CTX [s]	コンテキストエラー（非タスクコンテキストからの呼出し，CPUロック状態からの呼出し）
11267		
11268	E_ID	不正ID番号（almidが不正）
11269	E_RSATR	予約属性（属する保護ドメインかクラスが不正：SAC_ALMの場合）
11270		
11271	E_NOEXS [D]	オブジェクト未登録（対象アラームハンドラが未登録）
11272	E_OACV [sP]	オブジェクトアクセス違反（対象アラームハンドラに対する管理操作が許可されていない）
11273		
11274	E_MACV [sP]	メモリアクセス違反（p_acvctが指すメモリ領域への読出しアクセスが許可されていない）
11275		
11276	E_OBJ	オブジェクト状態エラー（対象アラームハンドラは静的APIで生成された：sac_almの場合，対象アラームハンドラに対してアクセス許可ベクタが設定済み：SAC_ALMの場合）
11277		
11278		
11279		

11280

11281 **【機能】**

11282

11283 almidで指定したアラームハンドラ（対象アラームハンドラ）のアクセス許可ベクタ（4つのアクセス許可パターンの組）を，各パラメータで指定した値に設定する。

11286

11287 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。

11288

11289 SAC_ALMは，対象アラームハンドラが属する保護ドメイン（この仕様ではカーネルドメインに限られる）の囲みの中に記述しなければならない。そうでない場合には，E_RSATRエラーとなる。

11293

11294 **【TOPPERS/ASPカーネルにおける規定】**

11295

11296 ASPカーネルでは，SAC_ALM，sac_almをサポートしない。

11297

11298 **【TOPPERS/FMPカーネルにおける規定】**

11299

11300 FMPカーネルでは，SAC_ALM，sac_almをサポートしない。

11301

11302 **【TOPPERS/HRP2カーネルにおける規定】**

11303

11304 HRP2カーネルでは、SAC_ALMのみをサポートする。

11305

11306 del_alm アラームハンドラの削除 [TD]

11307

11308 **【C言語API】**

11309 ER ercd = del_alm(ID almid)

11310

11311 **【パラメータ】**

11312 ID almid 対象アラームハンドラのID番号

11313

11314 **【リターンパラメータ】**

11315 ER ercd 正常終了 (E_OK) またはエラーコード

11316

11317 **【エラーコード】**

11318 E_CTX コンテキストエラー (非タスクコンテキストからの呼出し、CPUロック状態からの呼出し)

11319 E_ID 不正ID番号 (almidが不正)

11320 E_NOEXS [D] オブジェクト未登録 (対象アラームハンドラが未登録)

11321 E_OACV [P] オブジェクトアクセス違反 (対象アラームハンドラに対する管理操作が許可されていない)

11322 E_OBJ オブジェクト状態エラー (対象アラームハンドラは静的APIで生成された)

11323

11324

11325

11326

11327 **【機能】**

11328

11329 almidで指定したアラームハンドラ (対象アラームハンドラ) を削除する。具体的な振舞いは以下の通り。

11330

11331

11332 対象アラームハンドラの登録が解除され、そのアラームハンドラIDが未使用の状態に戻される。対象アラームハンドラが動作している状態であった場合には、登録解除の前に、アラームハンドラが動作していない状態となる。

11333

11334

11335

11336 **【TOPPERS/ASPカーネルにおける規定】**

11337

11338 ASPカーネルでは、del_almをサポートしない。ただし、動的生成機能拡張パッケージでは、del_almをサポートする。

11339

11340

11341 **【TOPPERS/FMPカーネルにおける規定】**

11342

11343 FMPカーネルでは、del_almをサポートしない。

11344

11345 **【TOPPERS/HRP2カーネルにおける規定】**

11346

11347 HRP2カーネルでは、del_almをサポートしない。

11348

11349 sta_alm アラームハンドラの動作開始 [T]

11350 ista_alm アラームハンドラの動作開始 [I]

11351

11352 **【C言語API】**

11353 ER ercd = sta_alm(ID almid, RELTIM almtim)

11354 ER ercd = ista_alm(ID almid, RELTIM almtim)

11355

11356 **【パラメータ】**

11357 ID almid 対象アラームハンドラのID番号

11358 RELTIM almtim アラームハンドラの起動時刻（相対時間）

11359

11360 **【リターンパラメータ】**

11361 ER ercd 正常終了（E_OK）またはエラーコード

11362

11363 **【エラーコード】**

11364 E_CTX コンテキストエラー（非タスクコンテキストからの呼出し：sta_almの場合，タスクコンテキストからの呼出し：ista_almの場合，CPUロック状態からの呼出し）

11365 E_ID 不正ID番号（almidが不正）

11366 E_PAR パラメータエラー（almtimが不正）

11367 E_NOEXS [D] オブジェクト未登録（対象アラームハンドラが未登録）

11368 E_OACV [P] オブジェクトアクセス違反（対象アラームハンドラに対する通常操作1が許可されていない：sta_almの場合）

11371

11372

11373 **【機能】**

11374

11375 almidで指定したアラームハンドラ（対象アラームハンドラ）を動作開始する。

11376 具体的な振舞いは以下の通り。

11377

11378 対象アラームハンドラが動作していない状態であれば，対象アラームハンドラは動作している状態となる．アラームハンドラを起動する時刻は，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。

11381

11382 対象アラームハンドラが動作している状態であれば，アラームハンドラを起動する時刻の再設定のみが行われる。

11384

11385 almtimは，TMAX_RELTIM以下でなければならない。

11386

11387 msta_alm 割付けプロセッサ指定でのアラームハンドラの動作開始 [TM]

11388 imsta_alm 割付けプロセッサ指定でのアラームハンドラの動作開始 [IM]

11389

11390 **【C言語API】**

11391 ER ercd = msta_alm(ID almid, RELTIM almtim, ID prcid)

11392 ER ercd = imsta_alm(ID almid, RELTIM almtim, ID prcid)

11393

11394 **【パラメータ】**

11395 ID almid 対象アラームハンドラのID番号

11396 RELTIM almtim アラームハンドラの起動時刻（相対時間）

11397 ID prcid アラームハンドラの割付け対象のプロセッサのID番号

11398

11399

11400 **【リターンパラメータ】**

11401	ER	ercd	正常終了 (E_OK) またはエラーコード
11402			
11403	【エラーコード】		
11404	E_CTX		コンテキストエラー (非タスクコンテキストからの呼出し: msta_almの場合, タスクコンテキストからの呼出し: imsta_almの場合, CPUロック状態からの呼出し)
11405			
11406			
11407	E_NOSPT		未サポート機能 (グローバルタイマ方式を用いている場合)
11408			
11409	E_ID		不正ID番号 (almid, prcidが不正)
11410	E_PAR		パラメータエラー (almtimが不正, 対象アラームハンドラはprcidで指定したプロセッサに割り付けられない)
11411			
11412	E_NOEXS [D]		オブジェクト未登録 (対象アラームハンドラが未登録)
11413	E_OACV [P]		オブジェクトアクセス違反 (対象アラームハンドラに対する通常操作1が許可されていない: msta_almの場合)
11414			
11415			
11416	【機能】		
11417			
11418	prcidで指定したプロセッサを割付けプロセッサとして, almidで指定したアラームハンドラ (対象アラームハンドラ) を動作開始する. 具体的な振舞いは以下の通り.		
11419			
11420			
11421			
11422	対象アラームハンドラが動作していない状態であれば, 対象アラームハンドラの割付けプロセッサがprcidで指定したプロセッサに変更された後, 対象アラームハンドラは動作している状態となる. アラームハンドラを起動する時刻は, msta_almを呼び出してから, almtimで指定した相対時間後に設定される.		
11423			
11424			
11425			
11426			
11427	対象アラームハンドラが動作している状態であれば, 対象アラームハンドラの割付けプロセッサがprcidで指定したプロセッサに変更された後, アラームハンドラを起動する時刻の再設定が行われる.		
11428			
11429			
11430			
11431	対象アラームハンドラが実行中である場合には, 割付けプロセッサを変更しても, 実行中のアラームハンドラを実行するプロセッサは変更されない. 対象アラームハンドラが変更後の割付けプロセッサで実行されるのは, 次に起動される時からである.		
11432			
11433			
11434			
11435			
11436	対象アラームハンドラの属するクラスの割付け可能プロセッサが, prcidで指定したプロセッサを含んでいない場合には, E_PARエラーとなる.		
11437			
11438			
11439	prcidにTPRC_INI (=0) を指定すると, 対象アラームハンドラの割付けプロセッサを, それが属するクラスの初期割付けプロセッサとする.		
11440			
11441			
11442	almtimは, TMAX_RELTIM以下でなければならない.		
11443			
11444	グローバルタイマ方式を用いている場合, msta_alm/imsta_almはE_NOSPTを返す.		
11445			
11446			
11447	【使用上の注意】		
11448			
11449	msta_alm/imsta_almで実行中のアラームハンドラの割付けプロセッサを変更した場合, 同じアラームハンドラが異なるプロセッサで同時に実行される可能性		
11450			

11451 がある．特に，almtimに0を指定する場合に，注意が必要である．

11452

11453 **【 μ ITRON4.0仕様との関係】**

11454

11455 μ ITRON4.0仕様に定義されていないサービスコールである．

11456

11457 stp_alm アラームハンドラの動作停止 [T]

11458 istp_alm アラームハンドラの動作停止 [I]

11459

11460 **【C言語API】**

11461 ER ercd = stp_alm(ID almid)

11462 ER ercd = istp_alm(ID almid)

11463

11464 **【パラメータ】**

11465 ID almid 対象アラームハンドラのID番号

11466

11467 **【リターンパラメータ】**

11468 ER ercd 正常終了 (E_OK) またはエラーコード

11469

11470 **【エラーコード】**

11471 E_CTX コンテキストエラー (非タスクコンテキストからの呼出し : stp_almの場合, タスクコンテキストからの呼出し : istp_almの場合, CPUロック状態からの呼出し)

11472

11473 E_ID 不正ID番号 (almidが不正)

11474 E_NOEXS [D] オブジェクト未登録 (対象アラームハンドラが未登録)

11475 E_OACV [P] オブジェクトアクセス違反 (対象アラームハンドラに対する通常操作2が許可されていない : stp_almの場合)

11476

11477

11478

11479

11480 **【機能】**

11481

11482 almidで指定したアラームハンドラ (対象アラームハンドラ) を動作停止する．

11483 具体的な振舞いは以下の通り．

11484

11485 対象アラームハンドラが動作している状態であれば，動作していない状態となる．対象アラームハンドラが動作していない状態であれば，何も行われずに正常終了する．

11486

11487

11488

11489

11490

11491

11492

11493

11494

11495

11496

11497

11498

11499

11500

ref_alm アラームハンドラの状態参照 [T]

【C言語API】

ER ercd = ref_alm(ID almid, T_RALM *pk_ralm)

【パラメータ】

ID almid 対象アラームハンドラのID番号

T_RALM * pk_ralm アラームハンドラの現在状態を入れるのポインタ

【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

11501 *アラームハンドラの現在状態（パケットの内容）
 11502 STAT almstat アラームハンドラの動作状態
 11503 RELTIM lefttim アラームハンドラを起動する時刻までの相対時間
 11504 ID pcrid アラームハンドラの割付けプロセッサのID（マルチ
 11505 プロセッサ対応カーネルの場合）
 11506

11507 【エラーコード】

11508 E_CTX コンテキストエラー（非タスクコンテキストからの呼出し、CPUロック状態からの呼出し）
 11509 E_ID 不正ID番号（almidが不正）
 11510 E_NOEXS [D] オブジェクト未登録（対象アラームハンドラが未登録）
 11511 E_OACV [P] オブジェクトアクセス違反（対象アラームハンドラに対する参照操作が許可されていない）
 11512 E_MACV [P] メモリアクセス違反（pk_ralmが指すメモリ領域への書き込みアクセスが許可されていない）
 11513
 11514
 11515
 11516

11517 【機能】

11518
 11519 almidで指定したアラームハンドラ（対象アラームハンドラ）の現在状態を参照
 11520 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。
 11521

11522 almstatには、対象アラームハンドラの現在の動作状態を表す次のいずれかの値
 11523 が返される。
 11524

11525 TALM_STP 0x01U アラームハンドラが動作していない状態
 11526 TALM_STA 0x02U アラームハンドラが動作している状態
 11527

11528 対象アラームハンドラが動作している状態である場合には、lefttimに、アラーム
 11529 ハンドラ起動する時刻までの相対時間が返される。対象アラームハンドラが
 11530 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。
 11531

11532 マルチプロセッサ対応カーネルでは、pcridに、対象アラームハンドラの割付け
 11533 プロセッサのID番号が返される。
 11534

11535 【使用上の注意】

11536
 11537 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
 11538 ない。これは、ref_almを呼び出し、対象アラームハンドラの現在状態を参照し
 11539 た直後に割込みが発生した場合、ref_almから戻ってきた時には対象アラームハ
 11540 ンドラの状態が変化している可能性があるためである。
 11541

11542 【μITRON4.0仕様との関係】

11543
 11544 TALM_STPとTALM_STAを値を変更した。
 11545 -----

11546 4.6.4 オーバランハンドラ

11547
 11548
 11549 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
 11550 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

11551 は、システムで1つのみ登録することができる。
11552
11553 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。
11554
11555 ・オーバランハンドラの動作状態
11556 ・残りプロセッサ時間
11557
11558 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
11559 いない状態のいずれかをとる。残りプロセッサ時間は、オーバランハンドラが
11560 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。
11561
11562 オーバランハンドラの動作状態は、タスクの起動時に、動作していない状態に
11563 初期化される。
11564
11565 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でタスクが実
11566 行している間、タスクが使用したプロセッサ時間の分だけ減少する。残りプロ
11567 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
11568 動される。
11569
11570 タスクが使用したプロセッサ時間には、そのタスク自身とタスク例外処理ルー
11571 チン、それらから呼び出したサービスクール（拡張サービスクールを含む）の
11572 実行時間を含む。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
11573 ンドラ（割込みサービスクール、周期ハンドラ、アラームハンドラ、オーバ
11574 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
11575 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
11576 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。また、
11577 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
11578 外のCPU例外ハンドラの実行時間も含む。
11579
11580 プロセッサ時間は、符号無しの整数型であるOVRTIM型で表し、単位はマイクロ
11581 秒とする。ただし、プロセッサ時間には、OVRTIM型に格納できる任意の値を指
11582 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。
11583 プロセッサ時間に指定できる最大値は、構成マクロTMAX_OVRTIMに定義されてい
11584 る。また、タスクが使用したプロセッサ時間の計測精度はターゲットに依存す
11585 る。
11586
11587 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
11588 属する。
11589
11590 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
11591 る。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、
11592 TOPPERS_SUPPORT_OVRHDRがマクロ定義される。サポートされていない場合にオー
11593 バランハンドラ機能のサービスクールを呼び出すと、E_NOSPTエラーが返るか、
11594 リンク時にエラーとなる。
11595
11596 オーバランハンドラ機能に用いるデータ型は次の通り。
11597
11598 OVRTIM プロセッサ時間（符号無し整数、単位はマイクロ秒、ulong_t
11599 に定義）
11600

11601 オーバランハンドラ属性に指定できる属性はない。そのためオーバランハンド
11602 ラ属性には、TA_NULLを指定しなければならない。

11603

11604 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。

11605

```
11606     void overrun_handler(ID tskid, intptr_t exinf)
11607     {
11608         オーバランハンドラ本体
11609     }
```

11610

11611 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
11612 情報が、それぞれ渡される。

11613

11614 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。

11615

11616 TMAX_OVRTIM プロセッサ時間に指定できる最大値

11617

11618 TOPPERS_SUPPORT_OVRHDR オーバランハンドラ機能がサポートされて
11619 いる

11620

11621 **【使用上の注意】**

11622

11623 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
11624 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。

11625

11626 **【TOPPERS/ASPカーネルにおける規定】**

11627

11628 ASPカーネルでは、オーバランハンドラをサポートしない。ただし、オーバラン
11629 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
11630 ことができる。

11631

11632 **【TOPPERS/FMPカーネルにおける規定】**

11633

11634 FMPカーネルでは、オーバランハンドラをサポートしない。

11635

11636 **【TOPPERS/HRP2カーネルにおける規定】**

11637

11638 HRP2カーネルでは、オーバランハンドラをサポートする。

11639

11640 **【 μ ITRON4.0仕様との関係】**

11641

11642 OVRTIMの時間単位は、 μ ITRON4.0仕様では実装定義としていたが、この仕様で
11643 はマイクロ秒と規定した。

11644

11645 TMAX_OVRTIMは、 μ ITRON4.0仕様に規定されていないカーネル構成マクロである。

11646 -----

11647 DEF_OVR オーバランハンドラの定義 [S]

11648 def_ovr オーバランハンドラの定義 [TD]

11649

11650 **【静的API】**

11651 DEF_OVR({ ATR ovratr, OVRHDR ovrhdr })

11652

11653 **【C言語API】**

11654 ER ercd = def_ovr(const T_DOVR *pk_dovr)

11655

11656 **【パラメータ】**

11657 T_DOVR * pk_dovr オーバランハンドラの定義情報を入れたパケッ
11658 トへのポインタ（静的APIを除く）

11659

11660 * オーバランハンドラの定義情報（パケットの内容）

11661 ATR ovratr オーバランハンドラ属性

11662 OVRHDR ovrhdr オーバランハンドラの先頭番地

11663

11664 **【リターンパラメータ】**

11665 ER ercd 正常終了（E_OK）またはエラーコード

11666

11667 **【エラーコード】**

11668 E_CTX [s] コンテキストエラー（非タスクコンテキストからの呼出
11669 し、CPUロック状態からの呼出し）

11670 E_RSATR 予約属性（ovratrが不正または使用できない、属する保
11671 護ドメインがクラスが不正）

11672 E_OACV [sP] オブジェクトアクセス違反（システム状態に対する管理
11673 操作が許可されていない）

11674 E_MACV [sP] メモリアクセス違反（pk_dovrが指すメモリ領域への読出
11675 しアクセスが許可されていない）

11676 E_PAR パラメータエラー（ovrhdrが不正）

11677 E_OBJ オブジェクト状態エラー（条件については機能の項を参
11678 照すること）

11679

11680 **【機能】**

11681

11682 各パラメータで指定したオーバランハンドラ定義情報に従って、オーバランハ
11683 ンドラを定義する。ただし、def_ovrにおいてpk_dovrをNULLにした場合には、
11684 オーバランハンドラの定義を解除する。

11685

11686 静的APIにおいては、ovratrは整数定数式パラメータ、ovrhdrは一般定数式パラ
11687 メータである。

11688

11689 オーバランハンドラを定義する場合（DEF_OVRの場合およびdef_ovrにおいて
11690 pk_dovrをNULL以外にした場合）で、すでにオーバランハンドラが定義されてい
11691 る場合には、E_OBJエラーとなる。

11692

11693 保護機能対応カーネルにおいて、DEF_OVRは、カーネルドメインの囲みの中に記
11694 述しなければならない。そうでない場合には、E_RSATRエラーとなる。また、
11695 def_ovrでオーバランハンドラを定義する場合には、オーバランハンドラの属す
11696 る保護ドメインを設定する必要はなく、オーバランハンドラ属性に
11697 TA_DOM(domid)を指定した場合にはE_RSATRエラーとなる。ただし、
11698 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
11699 出されない。

11700

11701 マルチプロセッサ対応カーネルでは、DEF_OVRは、クラスの囲みの外に記述しな
 11702 ければならない。そうでない場合には、E_RSATRエラーとなる。また、def_ovr
 11703 でオーバランハンドラを定義する場合には、オーバランハンドラの属するクラ
 11704 スを設定する必要はなく、オーバランハンドラ属性にTA_CLS(clsid)を指定した
 11705 場合にはE_RSATRエラーとなる。ただし、TA_CLS(TCLS_SELF)を指定した場合に
 11706 は、指定が無視され、E_RSATRエラーは検出されない。

11707
 11708 オーバランハンドラの定義を解除する場合（def_ovrにおいてpk_dovrをNULLに
 11709 した場合）で、オーバランハンドラが定義されていない場合には、E_OBJエラー
 11710 となる。

11711
 11712 オーバランハンドラの定義を解除すると、オーバランハンドラの動作状態は、
 11713 すべてのタスクに対して動作していない状態となる。

11714
 11715 **【使用上の注意】**

11716
 11717 def_ovrによりオーバランハンドラの定義を解除する場合、サービスコールの処
 11718 理時間およびカーネル内での割込み禁止時間が、タスクの総数に比例して長く
 11719 なる。特に、タスクの総数が多い場合、カーネル内での割込み禁止時間が長く
 11720 なるため、注意が必要である。

11721
 11722 **【TOPPERS/ASPカーネルにおける規定】**

11723
 11724 ASPカーネルのオーバランハンドラ機能拡張パッケージでは、DEF_OVRのみをサ
 11725 ポートする。

11726
 11727 **【TOPPERS/HRP2カーネルにおける規定】**

11728
 11729 HRP2カーネルでは、DEF_OVRのみをサポートする。

11730
 11731 **【 μ ITRON4.0仕様との関係】**

11732
 11733 ovrrhdrのデータ型をOVRHDRに変更した。

11734
 11735 def_ovrによって定義済みのオーバランハンドラを再定義しようとした場合に、
 11736 E_OBJエラーとすることにした。オーバランハンドラの定義を変更するには、一
 11737 度定義を解除してから、再度定義する必要がある。

11738 -----
 11739 sta_ovr オーバランハンドラの動作開始 [T]
 11740 ista_ovr オーバランハンドラの動作開始 [I]

11741
 11742 **【C言語API】**

11743 ER ercd = sta_ovr(ID tskid, OVRTIM ovrtime)
 11744 ER ercd = ista_ovr(ID tskid, OVRTIM ovrtime)

11745
 11746 **【パラメータ】**

11747 ID tskid 対象タスクのID番号
 11748 OVRTIM ovrtime 対象タスクの残りプロセッサ時間

11749
 11750 **【リターンパラメータ】**

11751 ER ercd 正常終了 (E_OK) またはエラーコード

11752

11753 **【エラーコード】**

11754 E_CTX コンテキストエラー (非タスクコンテキストからの呼出し : sta_ovrの場合, タスクコンテキストからの呼出し : ista_ovrの場合, CPUロック状態からの呼出し)

11755

11756

11757 E_ID 不正ID番号 (tskidが不正)

11758 E_NOEXS [D] オブジェクト未登録 (対象タスクが未登録)

11759 E_OACV [P] オブジェクトアクセス違反 (対象タスクに対する通常操作2が許可されていない : sta_ovrの場合)

11760

11761 E_PAR パラメータエラー (ovrtimが不正)

11762 E_OBJ オブジェクト状態エラー (オーバランハンドラが定義されていない)

11763

11764

11765 **【機能】**

11766

11767 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.

11768

11769

11770 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. 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.

11771

11772

11773

11774

11775 sta_ovr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.

11776

11777

11778 ovrtimは, 0より大きく, TMAX_OVRTIM以下の値でなければならない.

11779

11780 **【μ ITRON4.0仕様との関係】**

11781

11782 ista_ovrは, μ ITRON4.0仕様に定義されていないサービスコールである.

11783 -----

11784 stp_ovr オーバランハンドラの動作停止 [T]

11785 istp_ovr オーバランハンドラの動作停止 [I]

11786

11787 **【C言語API】**

11788 ER ercd = stp_ovr(ID tskid)

11789 ER ercd = istp_ovr(ID tskid)

11790

11791 **【パラメータ】**

11792 ID tskid 対象タスクのID番号

11793

11794 **【リターンパラメータ】**

11795 ER ercd 正常終了 (E_OK) またはエラーコード

11796

11797 **【エラーコード】**

11798 E_CTX コンテキストエラー (非タスクコンテキストからの呼出し : stp_ovrの場合, タスクコンテキストからの呼出し : istp_ovrの場合, CPUロック状態からの呼出し)

11799

11800

11801	E_ID	不正ID番号 (tskidが不正)
11802	E_NOEXS [D]	オブジェクト未登録 (対象タスクが未登録)
11803	E_OACV [P]	オブジェクトアクセス違反 (対象タスクに対する通常操
11804		作2が許可されていない: stp_ovrの場合)
11805	E_OBJ	オブジェクト状態エラー (オーバランハンドラが定義さ
11806		れていない)

11807

11808 **【機能】**

11809

11810 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
11811 停止する. 具体的な振舞いは以下の通り.

11812

11813 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
11814 なる. 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,
11815 何も行われずに正常終了する.

11816

11817 stp_ovr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
11818 となる.

11819

11820 **【μITRON4.0仕様との関係】**

11821

11822 istp_ovrは, μITRON4.0仕様に定義されていないサービスコールである.

11823

11824 ref_ovr オーバランハンドラの状態参照 [T]

11825

11826 **【C言語API】**

11827 ER ercd = ref_ovr(ID tskid, T_ROVR *pk_rovr)

11828

11829 **【パラメータ】**

11830	ID	tskid	対象タスクのID番号
11831	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
11832			トへのポインタ

11833

11834 **【リターンパラメータ】**

11835	ER	ercd	正常終了 (E_OK) またはエラーコード
-------	----	------	-----------------------

11836

11837 *タスクの現在状態 (パケットの内容)

11838	STAT	ovrstat	オーバランハンドラの動作状態
11839	OVRTIM	leftotm	残りプロセッサ時間

11840

11841 **【エラーコード】**

11842	E_CTX	コンテキストエラー (非タスクコンテキストからの呼出
11843		し, CPUロック状態からの呼出し)
11844	E_ID	不正ID番号 (tskidが不正)
11845	E_NOEXS [D]	オブジェクト未登録 (対象タスクが未登録)
11846	E_OACV [P]	オブジェクトアクセス違反 (対象タスクに対する参照操
11847		作が許可されていない)
11848	E_MACV [P]	メモリアクセス違反 (pk_rovrが指すメモリ領域への書込
11849		みアクセスが許可されていない)
11850	E_OBJ	オブジェクト状態エラー (オーバランハンドラが定義さ

11852

11854

11857

11860

11862

11864

11869

11871

11873

11878

11880

11884

11886

11888

11891

11894

11897

11898

11900

11901

11902 **【C言語API】**

11903 ER ercd = sac_sys(const ACVCT *p_acvct)

11904

11905 **【パラメータ】**11906 ACVCT * p_acvct アクセス許可ベクタを入れたパケットへのポ
11907 インタ（静的APIを除く）

11908

11909 *アクセス許可ベクタ（パケットの内容）

11910 ACPTN acptn1 通常操作1のアクセス許可パターン

11911 ACPTN acptn2 通常操作2のアクセス許可パターン

11912 ACPTN acptn3 管理操作のアクセス許可パターン

11913 ACPTN acptn4 参照操作のアクセス許可パターン

11914

11915 **【リターンパラメータ】**

11916 ER ercd 正常終了（E_OK）またはエラーコード

11917

11918 **【エラーコード】**11919 E_CTX [s] コンテキストエラー（非タスクコンテキストからの呼出
11920 し、CPUロック状態からの呼出し）11921 E_RSATR 予約属性（属する保護ドメインかクラスが不正：SAC_SYS
11922 の場合）11923 E_OACV [sP] オブジェクトアクセス違反（カーネルドメイン以外から
11924 の呼出し）11925 E_MACV [sP] メモリアクセス違反（p_acvctが指すメモリ領域への読出
11926 しアクセスが許可されていない）11927 E_OBJ オブジェクト状態エラー（システム状態のアクセス許可
11928 ベクタが設定済み：SAC_SYSの場合）

11929

11930 **【機能】**

11931

11932 システム状態のアクセス許可ベクタ（4つのアクセス許可パターンの組）を、各
11933 パラメータで指定した値に設定する。

11934

11935 静的APIにおいては、acptn1～acptn4は整数定数式パラメータである。

11936

11937 SAC_SYSは、カーネルドメインの囲みの中に記述しなければならない。そうでな
11938 い場合には、E_RSATRエラーとなる。

11939

11940 **【TOPPERS/ASPカーネルにおける規定】**

11941

11942 ASPカーネルでは、SAC_SYS, sac_sysをサポートしない。

11943

11944 **【TOPPERS/FMPカーネルにおける規定】**

11945

11946 FMPカーネルでは、SAC_SYS, sac_sysをサポートしない。

11947

11948 **【TOPPERS/HRP2カーネルにおける規定】**

11949

11950 HRP2カーネルでは、SAC_SYSのみをサポートする。

11951

11952 **【TOPPERS/SSPカーネルにおける規定】**

11953

11954 SSPカーネルでは、SAC_SYS, sac_sysをサポートしない.

11955

11956 rot_rdq タスクの優先順位の回転 [T]

11957 irot_rdq タスクの優先順位の回転 [I]

11958

11959 **【C言語API】**

11960 ER ercd = rot_rdq(PRI tskpri)

11961 ER ercd = irot_rdq(PRI tskpri)

11962

11963 **【パラメータ】**

11964 PRI tskpri 回転対象の優先度 (対象優先度)

11965

11966 **【リターンパラメータ】**

11967 ER ercd 正常終了 (E_OK) またはエラーコード

11968

11969 **【エラーコード】**11970 E_CTX コンテキストエラー (非タスクコンテキストからの呼出
11971 し: rot_rdqの場合, タスクコンテキストからの呼出し:
11972 irot_rdqの場合, CPUロック状態からの呼出し)11973 E_NOSPT 未サポート機能 (対象優先度の最も優先順位が高いタ
11974 スクが制約タスク)

11975 E_PAR パラメータエラー (tskpriが不正)

11976 E_OACV [P] オブジェクトアクセス違反 (システム状態に対する通常
11977 操作1が許可されていない)

11978

11979 **【機能】**

11980

11981 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,
11982 最も優先順位が高いタスクを, 同じ優先度のタスクの中で最も優先順位が低い
11983 状態にする. 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
11984 には, 何も行われずに正常終了する.

11985

11986 rot_rdqにおいて, tskpriにTPRI_SELF (=0) を指定すると, 自タスクのベース
11987 優先度が対象優先度となる.

11988

11989 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
11990 が制約タスクの場合には, E_NOSPTエラーとなる.

11991

11992 tskpriは, TPRI_SELFであるか (rot_rdqの場合のみ) , TMIN_TPRI以上,
11993 TMAX_TPRI以下でなければならない.

11994

11995 **【TOPPERS/SSPカーネルにおける規定】**

11996

11997 SSPカーネルでは, rot_rdq, irot_rdqをサポートしない.

11998

11999 mrot_rdq プロセッサ指定でのタスクの優先順位の回転 [TM]

12000 imrot_rdq プロセッサ指定でのタスクの優先順位の回転 [IM]

12001

12002 **【C言語API】**

12003 ER ercd = mrot_rdq(PRI tskpri, ID preid)

12004 ER ercd = imrot_rdq(PRI tskpri, ID preid)

12005

12006 **【パラメータ】**

12007 PRI tskpri 回転対象の優先度（対象優先度）

12008 ID preid 優先順位の回転対象とするプロセッサのID番号

12009

12010 **【リターンパラメータ】**

12011 ER ercd 正常終了（E_OK）またはエラーコード

12012

12013 **【エラーコード】**

12014 E_CTX コンテキストエラー（非タスクコンテキストからの呼出し：mrot_rdqの場合、タスクコンテキストからの呼出し：imrot_rdqの場合、CPUロック状態からの呼出し）

12016 E_NOSPT 未サポート機能（対象優先度の最も優先順位が高いタスクが制約タスク）

12018 E_ID 不正ID番号（preidが不正）

12019 E_PAR パラメータエラー（tskpriが不正）

12020 E_OACV [P] オブジェクトアクセス違反（システム状態に対する通常操作1が許可されていない）

12021

12022

12023

12024 **【機能】**

12025

12026 preidで指定したプロセッサに割り付けられており、tskpriで指定した優先度（対象優先度）を持つ実行できる状態のタスクの中で、最も優先順位が高いタスクを、同じ優先度のタスクの中で最も優先順位が低い状態にする。対象優先度を持つ実行できる状態のタスクが無いか1つのみの場合には、何も行われずに正常終了する。

12027

12028 mrot_rdqにおいて、tskpriにTPRI_SELF（=0）を指定すると、自タスクのベース優先度が対象優先度となる。

12029

12030 preidで指定したプロセッサに割り付けられており、対象優先度を持つ実行できる状態のタスクの中で、最も優先順位が高いタスクが制約タスクの場合には、E_NOSPTエラーとなる。

12031

12032 tskpriは、TPRI_SELFであるか（mrot_rdqの場合のみ）、TMIN_TPRI以上、TMAX_TPRI以下でなければならない。

12033

12034 **【TOPPERS/ASPカーネルにおける規定】**

12042

12043 ASPカーネルでは、mrot_rdq、imrot_rdqをサポートしない。

12044

12045 **【TOPPERS/HRP2カーネルにおける規定】**

12046

12047 HRP2カーネルでは、mrot_rdq、imrot_rdqをサポートしない。

12048

12049 **【TOPPERS/SSPカーネルにおける規定】**

12050

12051
 12052 SSPカーネルでは、mrot_rdq, imrot_rdqをサポートしない。
 12053
 12054 **【μITRON4.0仕様との関係】**
 12055
 12056 μITRON4.0仕様に定義されていないサービスコールである。
 12057 -----
 12058 get_tid 実行状態のタスクIDの参照 [T]
 12059 iget_tid 実行状態のタスクIDの参照 [I]
 12060
 12061 **【C言語API】**
 12062 ER ercd = get_tid(ID *p_tskid)
 12063 ER ercd = iget_tid(ID *p_tskid)
 12064
 12065 **【パラメータ】**
 12066 ID * p_tskid タスクIDを入れるメモリ領域へのポインタ
 12067
 12068 **【リターンパラメータ】**
 12069 ER ercd 正常終了 (E_OK) またはエラーコード
 12070 ID tskid タスクID
 12071
 12072 **【エラーコード】**
 12073 E_CTX コンテキストエラー (非タスクコンテキストからの呼出
 12074 し: get_tidの場合、タスクコンテキストからの呼出し:
 12075 iget_tidの場合、CPUロック状態からの呼出し)
 12076 E_MACV [P] メモリアクセス違反 (p_tskidが指すメモリ領域への書込
 12077 みアクセスが許可されていない)
 12078
 12079 **【機能】**
 12080
 12081 実行状態のタスク (get_tidの場合には自タスク) のID番号を参照する。参照し
 12082 たタスクIDは、p_tskidで指定したメモリ領域に返される。
 12083
 12084 iget_tidにおいて、実行状態のタスクがない場合には、TSK_NONE (=0) が返さ
 12085 れる。
 12086
 12087 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
 12088 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。
 12089
 12090 **【TOPPERS/SSPカーネルにおける規定】**
 12091
 12092 SSPカーネルでは、get_tidをサポートしない。
 12093 -----
 12094 get_did 実行状態のタスクが属する保護ドメインIDの参照 [TP]
 12095
 12096 **【C言語API】**
 12097 ER ercd = get_did(ID *p_domid)
 12098
 12099 **【パラメータ】**
 12100 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

12101

12102 **【リターンパラメータ】**

12103 ER ercd 正常終了 (E_OK) またはエラーコード

12104 ID domid 保護ドメインID

12105

12106 **【エラーコード】**

12107 E_CTX コンテキストエラー (非タスクコンテキストからの呼出

12108 し, CPUロック状態からの呼出し)

12109 E_MACV メモリアクセス違反 (p_domidが指すメモリ領域への書込

12110 みアクセスが許可されていない)

12111

12112 **【機能】**

12113

12114 実行状態のタスク (自タスク) が属する保護ドメインのID番号を参照する. 参

12115 照した保護ドメインIDは, p_domidで指定したメモリ領域に返される.

12116

12117 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

12118 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

12119 ンのID番号を参照する.

12120

12121 **【TOPPERS/ASPカーネルにおける規定】**

12122

12123 ASPカーネルでは, get_didをサポートしない.

12124

12125 **【TOPPERS/FMPカーネルにおける規定】**

12126

12127 FMPカーネルでは, get_didをサポートしない.

12128

12129 **【TOPPERS/SSPカーネルにおける規定】**

12130

12131 SSPカーネルでは, get_didをサポートしない.

12132 -----

12133 get_pid 割付けプロセッサのID番号の参照 [TM]

12134 iget_pid 割付けプロセッサのID番号の参照 [IM]

12135

12136 **【C言語API】**

12137 ER ercd = get_pid(ID *p_prcid)

12138 ER ercd = iget_pid(ID *p_prcid)

12139

12140 **【パラメータ】**

12141 ID * p_prcid プロセッサIDを入れるメモリ領域へのポインタ

12142

12143 **【リターンパラメータ】**

12144 ER ercd 正常終了 (E_OK) またはエラーコード

12145 ID prcid プロセッサID

12146

12147 **【エラーコード】**

12148 E_CTX コンテキストエラー (非タスクコンテキストからの呼出

12149 し: get_pidの場合, タスクコンテキストからの呼出し:

12150 iget_pidの場合, CPUロック状態からの呼出し)

12151 E_MACV [P] メモリアクセス違反 (p_prcidが指すメモリ領域への書込
12152 みアクセスが許可されていない)
12153

12154 【機能】

12155

12156 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。
12157 参照したプロセッサIDは、p_prcidで指定したメモリ領域に返される。
12158

12159 【使用上の注意】

12160

12161 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
12162 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
12163 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
12164 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。
12165

12166 【TOPPERS/ASPカーネルにおける規定】

12167

12168 ASPカーネルでは、get_pid、iget_pidをサポートしない。
12169

12170 【TOPPERS/HRP2カーネルにおける規定】

12171

12172 HRP2カーネルでは、get_pid、iget_pidをサポートしない。
12173

12174 【TOPPERS/SSPカーネルにおける規定】

12175

12176 SSPカーネルでは、get_pid、iget_pidをサポートしない。
12177

12178 【μITRON4.0仕様との関係】

12179

12180 μITRON4.0仕様に定義されていないサービスコールである。
12181

12182 loc_cpu CPUロック状態への遷移 [T]

12183 iloc_cpu CPUロック状態への遷移 [I]

12184

12185 【C言語API】

12186

ER ercd = loc_cpu()

12187

ER ercd = iloc_cpu()

12188

12189 【パラメータ】

12190

なし

12191

12192 【リターンパラメータ】

12193

ER ercd 正常終了 (E_OK) またはエラーコード

12194

12195 【エラーコード】

12196

E_CTX コンテキストエラー (非タスクコンテキストからの呼出し : loc_cpuの場合、タスクコンテキストからの呼出し : iloc_cpuの場合)

12197

12198

12199

E_OACV [P] オブジェクトアクセス違反 (システム状態に対する通常操作2が許可されていない : loc_cpuの場合)

12200

12201

12202 **【機能】**

12203

12204 CPUロックフラグをセットし、CPUロック状態へ遷移する。CPUロック状態で呼び
12205 出した場合には、何も行われずに正常終了する。

12206

12207 unl_cpu CPUロック状態の解除 [T]

12208 iunl_cpu CPUロック状態の解除 [I]

12209

12210 **【C言語API】**

12211 ER ercd = unl_cpu()

12212 ER ercd = iunl_cpu()

12213

12214 **【パラメータ】**

12215 なし

12216

12217 **【リターンパラメータ】**

12218 ER ercd 正常終了 (E_OK) またはエラーコード

12219

12220 **【エラーコード】**

12221 E_CTX コンテキストエラー（非タスクコンテキストからの呼出
12222 し：unl_cpuの場合、タスクコンテキストからの呼出し：
12223 iunl_cpuの場合）

12224 E_OACV [P] オブジェクトアクセス違反（システム状態に対する通常
12225 操作2が許可されていない：unl_cpuの場合）

12226

12227 **【機能】**

12228

12229 CPUロックフラグをクリアし、CPUロック解除状態へ遷移する。CPUロック解除状
12230 態で呼び出した場合には、何も行われずに正常終了する。

12231

12232 マルチプロセッサ対応カーネルにおいて、unl_cpu/iunl_cpuを呼び出したプロ
12233 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、
12234 unl_cpu/iunl_cpuによってCPUロック解除状態に遷移しない（何も行われずに
12235 正常終了する）。

12236

12237 **【補足説明】**

12238

12239 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
12240 スパッチ保留状態が解除され、ディスパッチが起こる可能性がある。また、保
12241 護機能対応カーネルとマルチプロセッサ対応カーネルでは、タスク例外処理ルー
12242 チンの実行が開始される可能性がある。

12243

12244 dis_dsp ディスパッチの禁止 [T]

12245

12246 **【C言語API】**

12247 ER ercd = dis_dsp()

12248

12249 **【パラメータ】**

12250 なし

12251
12252 **【リターンパラメータ】**
12253 ER ercd 正常終了 (E_OK) またはエラーコード
12254
12255 **【エラーコード】**
12256 E_CTX コンテキストエラー (非タスクコンテキストからの呼出
12257 し, CPUロック状態からの呼出し)
12258 E_OACV [P] オブジェクトアクセス違反 (システム状態に対する通常
12259 操作1が許可されていない)
12260
12261 **【機能】**
12262
12263 ディスパッチ禁止フラグをセットし, ディスパッチ禁止状態へ遷移する. ディ
12264 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.
12265 -----
12266 ena_dsp ディスパッチの許可 [T]
12267
12268 **【C言語API】**
12269 ER ercd = ena_dsp()
12270
12271 **【パラメータ】**
12272 なし
12273
12274 **【リターンパラメータ】**
12275 ER ercd 正常終了 (E_OK) またはエラーコード
12276
12277 **【エラーコード】**
12278 E_CTX コンテキストエラー (非タスクコンテキストからの呼出
12279 し, CPUロック状態からの呼出し)
12280 E_OACV [P] オブジェクトアクセス違反 (システム状態に対する通常
12281 操作1が許可されていない)
12282
12283 **【機能】**
12284
12285 ディスパッチ禁止フラグをクリアし, ディスパッチ許可状態へ遷移する. ディ
12286 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.
12287
12288 **【補足説明】**
12289
12290 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
12291 スパッチが起こる可能性がある.
12292 -----
12293 sns_ctx コンテキストの参照 [TI]
12294
12295 **【C言語API】**
12296 bool_t state = sns_ctx()
12297
12298 **【パラメータ】**
12299 なし
12300

12301 **【リターンパラメータ】**
12302 bool_t state コンテキスト
12303
12304 **【機能】**
12305
12306 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。
12307
12308 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
12309 ストから呼び出した場合にはfalseが返る。
12310 -----
12311 sns_loc CPUロック状態の参照 [TI]
12312
12313 **【C言語API】**
12314 bool_t state = sns_loc()
12315
12316 **【パラメータ】**
12317 なし
12318
12319 **【リターンパラメータ】**
12320 bool_t state CPUロックフラグ
12321
12322 **【機能】**
12323
12324 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。
12325
12326 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
12327 出した場合にはfalseが返る。
12328 -----
12329 sns_dsp ディスパッチ禁止状態の参照 [TI]
12330
12331 **【C言語API】**
12332 bool_t state = sns_dsp()
12333
12334 **【パラメータ】**
12335 なし
12336
12337 **【リターンパラメータ】**
12338 bool_t state ディスパッチ禁止フラグ
12339
12340 **【機能】**
12341
12342 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。
12343
12344 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
12345 状態で呼び出した場合にはfalseが返る。
12346 -----
12347 sns_dpn ディスパッチ保留状態の参照 [TI]
12348
12349 **【C言語API】**
12350 bool_t state = sns_dpn()

12351
12352 **【パラメータ】**
12353 なし
12354
12355 **【リターンパラメータ】**
12356 bool_t state ディスパッチ保留状態
12357
12358 **【機能】**
12359
12360 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．
12361
12362 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
12363 状態でない状態で呼び出した場合にはfalseが返る．
12364 -----
12365 sns_ker カーネル非動作状態の参照 [TI]
12366
12367 **【C言語API】**
12368 bool_t state = sns_ker()
12369
12370 **【パラメータ】**
12371 なし
12372
12373 **【リターンパラメータ】**
12374 bool_t state カーネル非動作状態
12375
12376 **【機能】**
12377
12378 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．
12379
12380 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
12381 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
12382 ネルの動作中に呼び出した場合にはfalseが返る．
12383
12384 **【使用方法】**
12385
12386 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
12387 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
12388 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
12389 合の動作は保証されない．
12390
12391 **【使用上の注意】**
12392
12393 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．
12394
12395 **【μITRON4.0仕様との関係】**
12396
12397 μITRON4.0仕様に定義されていないサービスコールである．
12398 -----
12399 ext_ker カーネルの終了 [TI]
12400

12401 **【C言語API】**
12402 ER ercd = ext_ker()
12403
12404 **【パラメータ】**
12405 なし
12406
12407 **【リターンパラメータ】**
12408 ER ercd エラーコード
12409
12410 **【エラーコード】**
12411 E_SYS システムエラー（カーネルの誤動作）
12412 E_OACV [P] オブジェクトアクセス違反（カーネルドメイン以外から
12413 の呼出し）
12414
12415 **【機能】**
12416
12417 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」
12418 の節を参照すること．
12419
12420 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．
12421
12422 **【μITRON4.0仕様との関係】**
12423
12424 μITRON4.0仕様に定義されていないサービスコールである．
12425 -----
12426 ref_sys システムの状態参照 [T]
12427
12428 **【C言語API】**
12429 ER ercd = ref_sys(T_RSYS *pk_rsys)
12430
12431 ☆未完成
12432
12433 **【TOPPERS/ASPカーネルにおける規定】**
12434
12435 ASPカーネルでは，ref_sysをサポートしない．
12436
12437 **【TOPPERS/FMPカーネルにおける規定】**
12438
12439 FMPカーネルでは，ref_sysをサポートしない．
12440
12441 **【TOPPERS/HRP2カーネルにおける規定】**
12442
12443 HRP2カーネルでは，ref_sysをサポートしない．
12444
12445 **【TOPPERS/SSPカーネルにおける規定】**
12446
12447 SSPカーネルでは，ref_sysをサポートしない．
12448 -----
12449
12450 4.8 メモリオブジェクト管理機能

12451
12452 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
12453 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
12454 をサポートしない。
12455
12456 〔メモリリージョン〕
12457
12458 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
12459 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、
12460 メモリリージョン名によって識別する。
12461
12462 各メモリリージョンが持つ情報は次の通り。
12463
12464 • 先頭番地
12465 • サイズ
12466 • メモリリージョン属性
12467
12468 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
12469 る場合がある。
12470
12471 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。
12472
12473 TA_NOWRITE 0x01U 書込みアクセス禁止
12474 TA_STDRAM 0x02U 標準ROMリージョン
12475 TA_STDRAM 0x04U 標準RAMリージョン
12476
12477 標準ROMリージョン（TA_STDRAM属性が指定されたメモリリージョン）は、
12478 ATT_MOD/ATA_MODにおいて、書込みアクセスを行わない標準のセクションを配
12479 置するメモリリージョンである。また、標準RAMリージョン（TA_STDRAM属性が
12480 指定されたメモリリージョン）は、ATT_MOD/ATA_MODにおいて、書込みアクセ
12481 スを行う標準のセクションを配置するメモリリージョンである。
12482
12483 標準ROMリージョンは、書込みアクセス禁止と扱われる。すなわち、TA_STDRAM
12484 属性を指定すれば、TA_NOWRITE属性も指定したことになる（TA_NOWRITE属性を
12485 指定しても指定しなくても、同じ振舞いとなる）。
12486
12487 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
12488 場合がある。
12489
12490 〔メモリオブジェクト〕
12491
12492 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
12493 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
12494 て識別する。
12495
12496 各メモリオブジェクトが持つ情報は次の通り。
12497
12498 • 先頭番地
12499 • サイズ
12500 • メモリオブジェクト属性

12501	・アクセス許可ベクタ
12502	・属する保護ドメイン
12503	・属するクラス（マルチプロセッサ対応カーネルの場合）
12504	
12505	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
12506	れる。
12507	
12508	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。
12509	
12510	TA_NOWRITE 0x01U 書込みアクセス禁止
12511	TA_NOREAD 0x02U 読出しアクセス禁止
12512	TA_EXEC 0x04U 実行アクセス許可
12513	TA_MEMINI 0x08U メモリの初期化を行う
12514	TA_MEMPRSV 0x10U メモリの初期化を行わない
12515	TA_SDATA 0x20U ショートデータ領域に配置
12516	TA_UNCACHE 0x40U キャッシュ禁止
12517	TA_IODEV 0x80U 周辺デバイスの領域
12518	
12519	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
12520	属性に書込みアクセス禁止（TA_NOWRITE属性）が指定されておらず、アクセス
12521	許可ベクタにより書込みアクセスが許可されている場合である。また、読出し
12522	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
12523	属性）が指定されておらず、アクセス許可ベクタにより読出し・実行アクセス
12524	が許可されている場合である。実行アクセスできるのは、メモリオブジェクト
12525	属性に実行アクセス許可（TA_EXEC属性）が指定されており、アクセス許可ベク
12526	タにより読出し・実行アクセスが許可されている場合である。
12527	
12528	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
12529	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、
12530	TA_NOWRITEを指定しても無視される。また、読出しアクセス禁止が実現できな
12531	い場合には、TA_NOREADを指定しても無視される。実行アクセス禁止が実現でき
12532	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
12533	無視される。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
12534	定義である。
12535	
12536	TA_MEMINI属性は、システム初期化時に初期化するメモリオブジェクトであるこ
12537	とを、TA_MEMPRSV属性は、システム初期化時に初期化を行わないメモリオブジェ
12538	クトであることを示す。いずれの属性も指定しない場合、そのメモリオブジェ
12539	クトは、システム初期化時にクリア（言い換えると、0に初期化）される。
12540	TA_MEMINIとTA_MEMPRSVの両方を指定した場合には、E_RSATRエラーとなる。
12541	
12542	TA_NOWRITEが指定されている場合には、TA_MEMINIとTA_MEMPRSVは無視される
12543	（指定しても指定しなくても、同じ振舞いとなる）。
12544	
12545	TA_MEMINI属性を設定したメモリオブジェクトを初期化に用いる初期化データは、
12546	標準ROMリージョンに配置され、メモリオブジェクトとしては登録されない。
12547	
12548	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
12549	示す。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
12550	トされていないターゲットでは、この属性は無視される。また、ターゲットに

12551 よっては、TA_NOWRITEを指定した場合に、TA_SDATAが無視される場合がある。

12552

12553 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、

12554 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

12555 示す。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

12556 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

12557 される。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

12558 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

12559 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

12560 る。

12561

12562 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

12563 る場合がある。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

12564 予約している。

12565

12566 TA_WTHROUGH ライトスルーキャッシュを用いる

12567

12568 [カーネル構成マクロ]

12569

12570 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。

12571

12572 TOPPERS_SUPPORT_ATT_MOD ATT_MOD/ATA_MODがサポートされている

12573 TOPPERS_SUPPORT_ATT_PMA ATT_PMA/ATA_PMA/att_pmaがサポートさ

12574 れている

12575

12576 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

12577 め、サポートされているかを判定するには、TOPPERS_SUPPORT_DYNAMIC_CREと

12578 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

12579 る。

12580

12581 【補足説明】

12582

12583 メモリオブジェクトが属するクラスは、ATT_MOD/ATA_MODにおいて、標準のセ

12584 クションが配置されるメモリリージョンを決定するためのみに使用される。

12585

12586 【TOPPERS/ASPカーネルにおける規定】

12587

12588 ASP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。

12589

12590 【TOPPERS/FMPカーネルにおける規定】

12591

12592 FMP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。

12593

12594 【TOPPERS/HRP2カーネルにおける規定】

12595

12596 HRP2カーネルでは、メモリオブジェクト管理機能をサポートする。

12597

12598 【TOPPERS/SSPカーネルにおける規定】

12599

12600 SSP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。

12601

12602 **【 μ ITRON4.0/PX仕様との関係】**

12603

12604 値が0のメモリオブジェクト属性 (TA_RW, TA_CACHE) は、デフォルトの扱いに
 12605 して廃止した。TA_ROはTA_NOWRITEに改名し、TA_NOREAD, TA_EXEC, TA_MEMINI,
 12606 TA_MEMPRSV, TA_IODEVを追加した。また、TA_UNCACHEの値を変更し、ターゲッ
 12607 ト定義のメモリオブジェクト属性としてTA_WTHROUGHを予約した。

12608

12609 メモリリージョンは、 μ ITRON4.0/PX仕様にはない概念である。

12610

12611 **【仕様決定の理由】**

12612

12613 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
 12614 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
 12615 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
 12616 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
 12617 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
 12618 るため、TA_IODEV属性を用意することにした。

12619

12620 ATT_REG メモリリージョンの登録 [SP]

12621

12622 **【静的API】**

12623 ATT_REG("メモリリージョン名", { ATR regatr, void *base, SIZE size })

12624

12625 **【パラメータ】**

12626 "メモリリージョン名" 登録するメモリリージョンを指定する文字列

12627 ATR regatr メモリリージョン属性

12628 void * base 登録するメモリリージョンの先頭番地

12629 SIZE size 登録するメモリリージョンのサイズ (バイト数)

12630

12631 **【エラーコード】**

12632 E_RSATR 予約属性 (regatrが不正または使用できない、保護ドメ
 12633 インの囲みの中に記述)

12634 E_PAR パラメータエラー (メモリリージョン名, base, sizeが
 12635 不正)

12636 E_OBJ オブジェクト状態エラー (登録済みのメモリリージョン
 12637 の再登録, その他の条件については機能の項を参照する
 12638 こと)

12639

12640 **【機能】**

12641

12642 各パラメータで指定したメモリリージョン登録情報に従って、指定したメモリ
 12643 リージョンを登録する。具体的な振舞いは以下の通り。

12644

12645 baseとsizeで指定したメモリ領域が、メモリリージョンとして登録される。登
 12646 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
 12647 定される。

12648

12649 メモリリージョン名は文字列パラメータ, regatr, base, sizeは整数定数式パ
 12650 ラメータである。

12651
 12652 ATT_MOD/ATA_MODがサポートされているターゲットでは、標準ROMリージョンと
 12653 標準RAMリージョンを、それぞれ1つ登録しなければならない。標準ROMリージョ
 12654 ンと標準RAMリージョンを登録していない場合や、複数登録した場合には、
 12655 E_OBJエラーとなる。

12656
 12657 マルチプロセッサ対応カーネルでは、ATT_REGをクラスの囲みの中に記述すると
 12658 そのクラス専用のメモリーリージョンとなり、ATT_REGをクラスの囲みの外に記述
 12659 すると共通のメモリーリージョンとなる。ATT_MOD/ATA_MODがサポートされてい
 12660 るターゲットでは、共通の標準ROMリージョンと共通の標準RAMリージョンを、
 12661 それぞれ1つ登録しなければならない。また、あるクラスの囲みの中でATT_MOD/
 12662 ATA_MODを使用する場合には、そのクラス専用の標準ROMリージョン/標準RAMリー
 12663 ジョンを1つ登録するか、共通の標準ROMリージョン/標準RAMリージョンを1つ
 12664 登録しなければならない。これらを登録しない場合や、複数登録した場合には、
 12665 E_OBJエラーとなる。

12666
 12667 mematrに、TA_STDROMとTA_STDRAMを同時に指定することはできない。指定した
 12668 場合には、E_RSATRエラーとなる。

12669
 12670 ATT_REGは、保護ドメインの囲みの外に記述しなければならない。そうでない場
 12671 合には、E_RSATRエラーとなる。

12672
 12673 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
 12674 た時には、E_PARエラーとなる。また、sizeが0の場合には、E_PARエラーとなる。
 12675 登録しようとしたメモリーリージョンが、登録済みのメモリーリージョンとメモリ
 12676 領域が重なる場合には、E_OBJエラーとなる。

12677
 12678 【μ ITRON4.0/PX仕様との関係】

12679
 12680 μ ITRON4.0/PX仕様に定義されていない静的APIである。

12681
 12682 ATT_SEC セクションの登録 [SP]
 12683 ATA_SEC セクションの登録（アクセス許可ベクタ付き） [SP]
 12684

12685 【静的API】
 12686 ATT_SEC("セクション名", { ATR mematr, "メモリーリージョン名" })
 12687 ATA_SEC("セクション名", { ATR mematr, "メモリーリージョン名" },
 12688 { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
 12689

12690 【パラメータ】
 12691 "セクション名" 登録するセクションを指定する文字列
 12692 ATR mematr メモリオブジェクト属性
 12693 "メモリーリージョン名" セクションを配置するメモリーリージョンを指定
 12694 する文字列
 12695

12696 *アクセス許可ベクタ（パケットの内容）
 12697 ACPTN acptn1 通常操作1のアクセス許可パターン
 12698 ACPTN acptn2 通常操作2のアクセス許可パターン
 12699 ACPTN acptn3 管理操作のアクセス許可パターン
 12700 ACPTN acptn4 参照操作のアクセス許可パターン

12701

12702 **【エラーコード】**12703 E_RSATR 予約属性（mematrが不正または使用できない、属するク
12704 ラスが不正）12705 E_PAR パラメータエラー（セクション名、メモリリージョン名
12706 が不正）12707 E_OBJ オブジェクト状態エラー（登録済みのセクションの再登
12708 録）

12709

12710 **【機能】**

12711

12712 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
12713 する。具体的な振舞いは以下の通り。

12714

12715 各オブジェクトモジュールに含まれるセクション名で指定したセクションが、
12716 メモリリージョン名で指定したメモリリージョンに配置され、メモリオブジェ
12717 クトとして登録される。登録されるメモリオブジェクトには、mematrで指定し
12718 たメモリオブジェクト属性が設定される。ATA_SECの場合には、登録されるメモ
12719 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、
12720 acptn1～acptn4で指定した値に設定される。

12721

12722 指定したメモリリージョンがTA_NOWRITE属性である場合には、メモリオブジェ
12723 クト属性にTA_NOWRITE属性を指定したことになる（TA_NOWRITE属性を指定して
12724 も指定しなくても、同じ振舞いとなる）。

12725

12726 mematrに、TA_MEMINIとTA_MEMPRSVを同時に指定することはできない。指定した
12727 場合には、E_RSATRエラーとなる。

12728

12729 登録されるメモリオブジェクトと同じ保護ドメインに属し、メモリオブジェク
12730 ト属性とアクセス許可ベクタがすべて一致するメモリオブジェクトがある場合
12731 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。

12732

12733 セクション名とメモリリージョン名は文字列パラメータ、mematr、acptn1～
12734 acptn4は整数定数式パラメータである。

12735

12736 ターゲット定義で、ATA_SECにより登録できるセクションが属する保護ドメイン
12737 や登録できる数に制限がある場合がある。

12738

12739 ATT_MOD／ATA_MODがサポートされているターゲットでは、セクション名として、
12740 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
12741 となる。

12742

12743 保護ドメイン毎の標準セクションは、コンフィギュレータによってカーネルに
12744 登録されるため、ATT_SEC／ATA_SECで登録することはできない。セクション名
12745 として指定した場合には、E_PARエラーとなる。

12746

12747 マルチプロセッサ対応カーネルにおいて、指定したメモリリージョンがあるク
12748 ラス専用のメモリリージョンの場合で、ATT_SEC／ATA_SECをクラスの囲みの外
12749 に記述するか、他のクラスの囲みの中に記述した場合には、E_RSATRエラーとな
12750 る。

12751

12752 **【 μ ITRON4.0/PX仕様との関係】**

12753

12754 μ ITRON4.0/PX仕様に定義されていない静的APIである。

12755

12756 LNK_SEC セクションの配置 [SP]

12757

12758 **【静的API】**

12759 LNK_SEC("セクション名", { "メモリアリージョン名" })

12760

12761 **【パラメータ】**

12762 "セクション名" 配置するセクションを指定する文字列

12763 "メモリアリージョン名" セクションを配置するメモリアリージョンを指定
12764 する文字列

12765

12766 **【エラーコード】**

12767 E_RSATR 予約属性（属するクラスが不正）

12768 E_PAR パラメータエラー（セクション名、メモリアリージョン名
12769 が不正）

12770 E_OBJ オブジェクト状態エラー（登録済みのセクションの再登
12771 録）

12772

12773 **【機能】**

12774

12775 各オブジェクトモジュールに含まれるセクション名で指定したセクションを、
12776 メモリアリージョン名で指定したメモリアリージョンに配置する。

12777

12778 セクション名として、標準のセクションや保護ドメイン毎の標準セクションを
12779 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。

12780

12781 マルチプロセッサ対応カーネルにおいて、指定したメモリアリージョンがあるク
12782 ラス専用のメモリアリージョンの場合で、LNK_SECをクラスの囲みの外に記述する
12783 か、他のクラスの囲みの中に記述した場合には、E_RSATRエラーとなる。

12784

12785 **【使用上の注意】**

12786

12787 LNK_SECにより配置されたセクションは、メモリオブジェクトとしてカーネルに
12788 登録されず、メモリ保護が実現できる先頭番地とサイズになるとは限らない。

12789

12790 **【 μ ITRON4.0/PX仕様との関係】**

12791

12792 μ ITRON4.0/PX仕様に定義されていない静的APIである。

12793

12794 ATT_MOD オブジェクトモジュールの登録 [SP]

12795 ATA_MOD オブジェクトモジュールの登録（アクセス許可ベクタ付き） [SP]

12796

12797 **【静的API】**

12798 ATT_MOD("オブジェクトモジュール名")

12799 ATA_MOD("オブジェクトモジュール名",

12800 { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })

- 12801
- 12802 **【パラメータ】**
- 12803 “オブジェクトモジュール名” 登録するオブジェクトモジュールを指
- 12804 定する文字列
- 12805
- 12806 *アクセス許可ベクタ (パケットの内容)
- 12807 ACPTN acptn1 通常操作1のアクセス許可パターン
- 12808 ACPTN acptn2 通常操作2のアクセス許可パターン
- 12809 ACPTN acptn3 管理操作のアクセス許可パターン
- 12810 ACPTN acptn4 参照操作のアクセス許可パターン
- 12811
- 12812 **【エラーコード】**
- 12813 E_RSATR 予約属性 (mematrが不正または使用できない)
- 12814 E_NOSPT 未サポート機能 (ATT_MOD/ATA_MODがサポートされてい
- 12815 ない)
- 12816 E_OBJ オブジェクト状態エラー (登録済みのオブジェクトモジ
- 12817 ュールの再登録)
- 12818
- 12819 **【機能】**
- 12820
- 12821 各パラメータで指定した情報に従って、指定したオブジェクトモジュールをカー
- 12822 ネルに登録する。具体的な振舞いは以下の通り。
- 12823
- 12824 オブジェクトモジュール名で指定したオブジェクトモジュールに含まれる標準
- 12825 のセクションの内、書込みアクセスを行わないセクションは標準ROMリージョン
- 12826 に、書込みアクセスを行うセクションは標準RAMリージョンに配置され、メモリ
- 12827 オブジェクトとして登録される。登録されるメモリオブジェクトには、ターゲッ
- 12828 ト定義でセクション毎に定まるメモリオブジェクト属性が設定される。ATA_MOD
- 12829 の場合には、登録されるメモリオブジェクトのアクセス許可ベクタ (4つのアク
- 12830 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。
- 12831
- 12832 マルチプロセッサ対応カーネルでは、ATT_MOD/ATA_MODを、クラスの囲みの外
- 12833 に記述することも、クラスの囲みの中に記述することもできる。ATT_MOD/
- 12834 ATA_MODをクラスの囲みの外に記述した場合、標準のセクションが配置されるメ
- 12835 モリリージョンは、共通の標準ROMリージョン/標準RAMリージョンとなる。ク
- 12836 ラスの囲みの中に記述した場合、そのクラス専用の標準ROMリージョン/標準
- 12837 RAMリージョンが登録されていればそのリージョン、登録されていなければ共通
- 12838 の標準ROMリージョン/標準RAMリージョンとなる。
- 12839
- 12840 登録されるメモリオブジェクトと同じ保護ドメインに属し、メモリオブジェク
- 12841 ト属性とアクセス許可ベクタがすべて一致するメモリオブジェクトがある場合
- 12842 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。
- 12843
- 12844 オブジェクトモジュール名は文字列パラメータ、acptn1～acptn4は整数定数式
- 12845 パラメータである。
- 12846
- 12847 ターゲット定義で、ATA_MODにより登録できるオブジェクトモジュールが属する
- 12848 保護ドメインや登録できる数に制限がある場合がある。
- 12849
- 12850 ターゲット定義で、ATT_MOD/ATA_MODがサポートされていない場合がある。

12851 ATT_MOD/ATA_MODがサポートされている場合には、TOPPERS_SUPPORT_ATT_MODが
 12852 マクロ定義される。サポートされていない場合にATT_MOD/ATA_MODを使用する
 12853 と、コンフィギュレータがE_NOSPTエラーを報告する。

12854
 12855 **【補足説明】**

12856
 12857 各セクションが配置されるメモリリージョンとそれに設定されるメモリオブジェ
 12858 クト属性は、例えば、プログラムコードを含むセクションであれば、フラッシュ
 12859 メモリのメモリリージョンに配置し、書込みアクセス禁止のメモリオブジェク
 12860 ト属性が設定されるというように、ターゲットシステム毎に定められる。

12861
 12862 ATT_MOD/ATA_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
 12863 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/ATA_SECを用
 12864 いる必要がある。

12865
 12866 **【μITRON4.0/PX仕様との関係】**

12867
 12868 オブジェクトモジュールに含まれるセクションの配置場所が、標準ROMリージョ
 12869 ンと標準RAMリージョンであることを明確化した。

12870 -----
 12871 ATT_MEM メモリオブジェクトの登録 [SP]
 12872 ATA_MEM メモリオブジェクトの登録（アクセス許可ベクタ付き） [SP]
 12873 att_mem メモリオブジェクトの登録 [TPD]

12874
 12875 **【静的API】**

12876 ATT_MEM({ ATR mematr, void *base, SIZE size })
 12877 ATA_MEM({ ATR mematr, void *base, SIZE size },
 12878 { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })

12879
 12880 **【C言語API】**

12881 ER ercd = att_mem(const T_AMEM *pk_amem)

12882
 12883 **【パラメータ】**

12884 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
 12885 トへのポインタ（静的APIを除く）

12886
 12887 *メモリオブジェクトの登録情報（パケットの内容）

12888 ATR mematr メモリオブジェクト属性
 12889 void * base 登録するメモリ領域の先頭番地
 12890 SIZE size 登録するメモリ領域のサイズ（バイト数）

12891
 12892 *アクセス許可ベクタ（パケットの内容）

12893 ACPTN acptn1 通常操作1のアクセス許可パターン
 12894 ACPTN acptn2 通常操作2のアクセス許可パターン
 12895 ACPTN acptn3 管理操作のアクセス許可パターン
 12896 ACPTN acptn4 参照操作のアクセス許可パターン

12897
 12898 **【リターンパラメータ】**

12899 ER ercd 正常終了（E_OK）またはエラーコード

12900

12901 **【エラーコード】**

12902	E_CTX [s]	コンテキストエラー（非タスクコンテキストからの呼出し、CPUロック状態からの呼出し）
12903		
12904	E_RSATR	予約属性（mematrが不正または使用できない、属する保護ドメインが不正）
12905		
12906	E_NOSPT	未サポート機能（条件はターゲット定義）
12907	E_PAR	パラメータエラー（base, sizeが不正）
12908	E_OACV [sP]	オブジェクトアクセス違反（システム状態に対する管理操作が許可されていない）
12909		
12910	E_MACV [sP]	メモリアクセス違反（pk_amemが指すメモリ領域への読出しアクセスが許可されていない）
12911		
12912	E_OBJ	オブジェクト状態エラー（登録済みのメモリオブジェクトとメモリ領域が重なる）
12913		

12914

12915 **【機能】**

12916

12917 各パラメータで指定したメモリオブジェクト登録情報に従って、メモリオブジェクトを登録する。具体的な振舞いは以下の通り。

12918

12919

12920 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。

12921 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。ATA_MEMの場合には、登録されるメモリオブジェクトのアクセス許可ベクタ（4つのアクセス許可パターンの組）が、acptn1～acptn4で指定した値に設定される。

12922

12923

12924 mematrには、TA_MEMPRSVを指定しなければならず、TA_MEMINIを指定することはできない。TA_MEMPRSVを指定しない場合や、TA_MEMINIを指定した場合には、E_RSATRエラーとなる。

12925

12926

12927 静的APIにおいては、mematr, size, acptn1～acptn4は整数定数式パラメータ、

12928

12929

12930 baseは一般定数式パラメータである。

12931

12932

12933 ターゲット定義で、ATT_MEM／ATA_MEMにより登録できるメモリオブジェクトが

12934

12935

12936 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。

12937

12938

12939

12940

12941

12942

12943

12944

12945

12946

12947

12948

12949

12950

【使用上の注意】

ATT_MEM／ATA_MEMは、ユーザタスクからメモリ空間にマッピングされたI/O領域にアクセスできるようにするために使用することを想定した静的APIである。メモリ領域に対しては、ATT_SEC／ATA_SECかATT_MOD／ATA_MODを使用することを推奨する。

ATT_MEM／ATA_MEMで登録したメモリオブジェクトのメモリ領域が、ATT_REGで登録したメモリージョンと重なっても、直ちにエラーとはならない。ただし、メモリージョン内に配置されたメモリオブジェクトと、ATT_MEM／ATA_MEMで

12951 登録したメモリオブジェクトのメモリ領域が重なった場合には、E_OBJエラーと
12952 なる。
12953
12954 **【TOPPERS/HRP2カーネルにおける規定】**
12955
12956 HRP2カーネルでは、ATT_MEMとATA_MEMのみをサポートする。
12957
12958 **【μITRON4.0/PX仕様との関係】**
12959
12960 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
12961 (ata_mem) は廃止した。
12962
12963 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
12964 ターゲット定義の制約に合致するようにメモリ領域を広げることとしていたが、
12965 この仕様ではE_PARエラーとなることとした。
12966 -----
12967 ATT_PMA 物理メモリ領域の登録 [SP]
12968 ATA_PMA 物理メモリ領域の登録（アクセス許可ベクタ付き） [SP]
12969 att_pma 物理メモリ領域の登録 [TPD]
12970
12971 **【静的API】**
12972 ATT_PMA({ ATR mematr, void *base, SIZE size, void *paddr })
12973 ATA_PMA({ ATR mematr, void *base, SIZE size, void *paddr },
12974 { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
12975
12976 **【C言語API】**
12977 ER ercd = att_pma(const T_APMA *pk_apma)
12978
12979 **【パラメータ】**
12980 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
12981 のポインタ（静的APIを除く）
12982
12983 *物理メモリ領域の登録情報（パケットの内容）
12984 ATR mematr メモリオブジェクト属性
12985 void * base 登録するメモリ領域の先頭番地
12986 SIZE size 登録するメモリ領域のサイズ（バイト数）
12987 void * paddr 登録するメモリ領域の物理アドレスの先頭番地
12988
12989 *アクセス許可ベクタ（パケットの内容）
12990 ACPTN acptn1 通常操作1のアクセス許可パターン
12991 ACPTN acptn2 通常操作2のアクセス許可パターン
12992 ACPTN acptn3 管理操作のアクセス許可パターン
12993 ACPTN acptn4 参照操作のアクセス許可パターン
12994
12995 **【リターンパラメータ】**
12996 ER ercd 正常終了 (E_OK) またはエラーコード
12997
12998 **【エラーコード】**
12999 E_CTX [s] コンテキストエラー（非タスクコンテキストからの呼出
13000 し、CPUロック状態からの呼出し）

13001	E_RSATR	予約属性 (mematrが不正または使用できない, 属する保護ドメインが不正)
13002		
13003	E_NOSPT	未サポート機能 (条件はターゲット定義)
13004	E_PAR	パラメータエラー (base, size, paddrが不正)
13005	E_OACV [sP]	オブジェクトアクセス違反 (システム状態に対する管理操作が許可されていない)
13006		
13007	E_MACV [sP]	メモリアクセス違反 (pk_apmaが指すメモリ領域への読出しアクセスが許可されていない)
13008		
13009	E_OBJ	オブジェクト状態エラー (登録済みのメモリオブジェクトとメモリ領域が重なる)
13010		
13011		
13012	【機能】	
13013		
13014	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.	
13015		
13016		
13017	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. 登録されるメモリオブジェクトには,	
13018	mematrで指定したメモリオブジェクト属性が設定される. ATA_PMAの場合には,	
13019	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.	
13020		
13021		
13022		
13023		
13024	mematrには, TA_MEMPRSVを指定しなければならず, TA_MEMINIを指定することはできない. TA_MEMPRSVを指定しない場合や, TA_MEMINIを指定した場合には,	
13025	E_RSATRエラーとなる.	
13026		
13027		
13028	静的APIにおいては, mematr, 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.	
13029		
13030		
13031	ターゲット定義で, ATT_PMA/ATA_PMAにより登録できるメモリオブジェクトが属する保護ドメインや登録できる数に制限がある場合がある.	
13032		
13033		
13034	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. また, sizeが0の場合には, E_PARエラーとなる. 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.	
13035		
13036		
13037		
13038		
13039		
13040	ATT_PMA/ATA_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. ATT_PMA/ATA_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. ATT_PMA/ATA_PMAがサポートされていない場合にこれらの静的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. 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.	
13041		
13042		
13043		
13044		
13045		
13046		
13047		
13048	【TOPPERS/HRP2カーネルにおける規定】	
13049		
13050	HRP2カーネルでは, ターゲット定義で, ATT_PMAとATA_PMAのみをサポートする.	

13051

13052 **【 μ ITRON4.0/PX仕様との関係】**

13053

13054 μ ITRON4.0/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。

13055

13056 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]

13057

13058 **【C言語API】**

13059 ER ercd = sac_mem(const void *base, const ACVCT *p_acvct)

13060

13061 **【パラメータ】**

13062 void * base メモリオブジェクトの先頭番地

13063 ACVCT * p_acvct アクセス許可ベクタを入れたパケットへのポ
13064 インタ

13065

13066 *アクセス許可ベクタ (パケットの内容)

13067 ACPTN acptn1 通常操作1のアクセス許可パターン

13068 ACPTN acptn2 通常操作2のアクセス許可パターン

13069 ACPTN acptn3 管理操作のアクセス許可パターン

13070 ACPTN acptn4 参照操作のアクセス許可パターン

13071

13072 **【リターンパラメータ】**

13073 ER ercd 正常終了 (E_OK) またはエラーコード

13074

13075 **【エラーコード】**13076 E_CTX コンテキストエラー (非タスクコンテキストからの呼出
13077 し, CPUロック状態からの呼出し)

13078 E_PAR パラメータエラー (baseが不正)

13079 E_NOEXS [D] オブジェクト未登録 (baseで指定した番地を含むメモリ
13080 オブジェクトが登録されていない)13081 E_OACV [P] オブジェクトアクセス違反 (対象メモリオブジェクトに
13082 対する管理操作が許可されていない)13083 E_MACV [P] メモリアクセス違反 (p_acvctが指すメモリ領域への読出
13084 しアクセスが許可されていない)13085 E_OBJ オブジェクト状態エラー (対象メモリオブジェクトは静的
13086 APIで登録された)

13087

13088 **【機能】**

13089

13090 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
13091 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
13092 設定する。

13093

13094 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
13095 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
13096 ラーとなる。

13097

13098 baseに, メモリオブジェクトの先頭番地以外を指定した場合には, E_PARエラー
13099 となる。

13100

13101 【TOPPERS/HRP2カーネルにおける規定】
 13102
 13103 HRP2カーネルでは、`sac_mem`をサポートしない。
 13104
 13105 【 μ ITRON4.0/PX仕様との関係】
 13106
 13107 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
 13108 ることができないこととした。
 13109
 13110 μ ITRON4.0/PX仕様では、`base`はメモリオブジェクトに含まれる番地を指定する
 13111 ものとしていたが、この仕様では、メモリオブジェクトの先頭番地でなければ
 13112 ならないものとした。
 13113 -----
 13114 `det_mem` メモリオブジェクトの登録解除 [TPD]
 13115
 13116 【C言語API】
 13117 `ER ercd = det_mem(const void *base)`
 13118
 13119 【パラメータ】
 13120 `void *` `base` メモリオブジェクトの先頭番地
 13121
 13122 【リターンパラメータ】
 13123 `ER` `ercd` 正常終了 (E_OK) またはエラーコード
 13124
 13125 【エラーコード】
 13126 `E_CTX` コンテキストエラー (非タスクコンテキストからの呼出
 13127 し、CPUロック状態からの呼出し)
 13128 `E_PAR` パラメータエラー (baseが不正)
 13129 `E_NOEXS [D]` オブジェクト未登録 (baseで指定される番地を含むメモ
 13130 リオブジェクトが登録されていない)
 13131 `E_OACV [P]` オブジェクトアクセス違反 (対象メモリオブジェクトに
 13132 対する管理操作が許可されていない)
 13133 `E_OBJ` オブジェクト状態エラー (対象メモリオブジェクトは静的
 13134 APIで登録された)
 13135
 13136 【機能】
 13137
 13138 `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
 13139 る。
 13140
 13141 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
 13142 い。登録を解除しようとした場合には、`E_OBJ`エラーとなる。
 13143
 13144 `base`に、メモリオブジェクトの先頭番地以外を指定した場合には、`E_PAR`エラー
 13145 となる。
 13146
 13147 【TOPPERS/HRP2カーネルにおける規定】
 13148
 13149 HRP2カーネルでは、`det_mem`をサポートしない。
 13150

13151 【μ ITRON4.0/PX仕様との関係】

13152

13153 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
13154 いこととした。

13155

13156 μ ITRON4.0/PX仕様では、baseはメモリオブジェクトに含まれる番地を指定する
13157 ものとしていたが、この仕様では、メモリオブジェクトの先頭番地でなければ
13158 ならないものとした。

13159

13160 prb_mem メモリ領域に対するアクセス権のチェック [TP]

13161

13162 【C言語API】

13163 ER ercd = prb_mem(const void *base, SIZE size, ID tskid, MODE pmmode)

13164

13165 【パラメータ】

13166	void *	base	メモリ領域の先頭番地
13167	SIZE	size	メモリ領域のサイズ (バイト数)
13168	ID	tskid	アクセス元のタスクのID番号
13169	MODE	pmmode	アクセスモード

13170

13171 【リターンパラメータ】

13172 ER ercd 正常終了 (E_OK) またはエラーコード

13173

13174 【エラーコード】

13175	E_CTX	コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
13176		
13177	E_ID	不正ID番号 (tskidが不正)
13178	E_PAR	パラメータエラー (base, size, pmmodeが不正)
13179	E_NOEXS [D]	オブジェクト未登録 (baseで指定される番地を含むメモリオブジェクトが登録されていない)
13180		
13181	E_OACV [P]	オブジェクトアクセス違反 (対象メモリ領域を含むメモリオブジェクトに対する参照操作が許可されていない)
13182		
13183	E_MACV [P]	メモリアクセス違反 (対象メモリ領域へのアクセスが許可されていない)
13184		
13185	E_OBJ	オブジェクト状態エラー (対象メモリ領域がメモリオブジェクトの境界を越えている)
13186		

13187

13188 【機能】

13189

13190 tskidで指定したタスクから、baseとsizeで指定したメモリ領域 (対象メモリ領
13191 域) に対して、pmmodeで指定した種別のアクセスが許可されているかをチェッ
13192 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
13193 る。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
13194 とはない。

13195

13196 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
13197 "|") を指定することができる。TPM_WRITE, TPM_READ, TPM_EXECを指定した場
13198 合には、それぞれ、読出しアクセス、書込みアクセス、実行アクセスが許可さ
13199 れているかをチェックする。また、いくつかのビット毎論理和を指定した場合
13200

13201 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
13202 する.

13203
13204 tskidにTSK_SELF (=0) を指定すると、自タスクから対象メモリ領域に対して
13205 アクセスが許可されているかをチェックする.

13206
13207 sizeが0の場合には、E_PARエラーとなる.
13208

13209 【μITRON4.0/PX仕様との関係】

13210
13211 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
13212 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
13213 TPM_EXECを追加し、TPM_WRITEとTPM_READの値を入れ換えた.

13214 -----
13215 ref_mem メモリオブジェクトの状態参照 [TP]

13216
13217 【C言語API】
13218 ER ercd = ref_mem(const void *base, T_RMEM *pk_rmem)

13219
13220 ☆未完成

13221 【TOPPERS/HRP2カーネルにおける規定】

13222
13223
13224 HRP2カーネルでは、ref_memをサポートしない.
13225 -----

13226 4.9 割込み管理機能

13227
13228
13229 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
13230 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、
13231 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
13232 おく必要がある. 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
13233 経由して呼び出される.

13234
13235 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
13236 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. こ
13237 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
13238 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
13239 ておく必要がある.

13240
13241 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、
13242 次の属性を指定することができる.

13243
13244 TA_ENAINT 0x01U 割込み要求禁止フラグをクリア
13245 TA_EDGE 0x02U エッジトリガ

13246
13247 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
13248 場合がある. 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
13249 している.
13250

13251 TA_POSEDGE ポジティブエッジトリガ
 13252 TA_NEGEDGE ネガティブエッジトリガ
 13253 TA_BOTHEDGE 両エッジトリガ
 13254 TA_LOWLEVEL ローレベルトリガ
 13255 TA_HIGHLEVEL ハイレベルトリガ
 13256 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
 13257 サ対応カーネルの場合）
 13258
 13259 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
 13260 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
 13261 する。
 13262
 13263 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、
 13264 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
 13265 にすべて呼び出される。割込みサービスルーチン優先度が同じ場合には、登録
 13266 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
 13267 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。
 13268
 13269 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
 13270 る保護ドメインは、カーネルドメインに限られる。
 13271
 13272 割込みサービスルーチン属性に指定できる属性はない。そのため割込みサービ
 13273 スルーチン属性には、TA_NULLを指定しなければならない。
 13274
 13275 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。
 13276
 13277 void interrupt_service_routine(intptr_t exinf)
 13278 {
 13279 割込みサービスルーチン本体
 13280 }
 13281
 13282 exinfには、割込みサービスルーチンの拡張情報が渡される。
 13283
 13284 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
 13285 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。
 13286
 13287 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
 13288 る。
 13289
 13290 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
 13291 定義で、次の属性を指定することができる。
 13292
 13293 TA_NONKERNEL 0x02U カーネル管理外の割込み
 13294
 13295 TA_NONKERNELを指定しない場合、カーネル管理の割込みとなる。また、ターゲッ
 13296 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
 13297 合がある。
 13298
 13299 C言語による割込みハンドラの記述形式は次の通り。
 13300

```

13301     void interrupt_handler(void)
13302     {
13303         割込みハンドラ本体
13304     }
13305 
```

13306 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.

```

13307
13308     TMIN_INTPRI    割込み優先度の最小値 (最高値)
13309     TMAX_INTPRI    割込み優先度の最大値 (最低値, =-1)
13310
13311     TMIN_ISRPRI    割込みサービスルーチン優先度の最小値 (=1)
13312     TMAX_ISRPRI    割込みサービスルーチン優先度の最大値
13313
13314     TOPPERS_SUPPORT_DIS_INT    dis_intがサポートされている
13315     TOPPERS_SUPPORT_ENA_INT    ena_intがサポートされている
13316 
```

13317 【TOPPERS/ASPカーネルにおける規定】

13318
13319 ASPカーネルでは、割込みサービスルーチン優先度の最大値 (=TMAX_ISRPRI)
13320 は16に固定されている. ただし、タスク優先度拡張パッケージでは、
13321 TMAX_ISRPRIを256に拡張する.

13322 【TOPPERS/FMPカーネルにおける規定】

13323
13324 FMPカーネルでは、割込みサービスルーチン優先度の最大値 (=TMAX_ISRPRI)
13325 は16に固定されている.

13326 【TOPPERS/HRP2カーネルにおける規定】

13327
13328 HRP2カーネルでは、割込みサービスルーチン優先度の最大値 (=TMAX_ISRPRI)
13329 は16に固定されている.

13330 【TOPPERS/SSPカーネルにおける規定】

13331
13332 SSPカーネルでは、割込みサービスルーチン優先度の最大値 (=TMAX_ISRPRI)
13333 は16に固定されている.

13334 【μITRON4.0仕様との関係】

13335
13336 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、
13337 μITRON4.0仕様にはない概念であり、TMIN_INTPRI, TMAX_INTPRI, TMIN_ISRPRI,
13338 TMAX_ISRPRIは、μITRON4.0仕様に定義のないカーネル構成マクロである. また、
13339 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.

```

13340     CFG_INT        割込み要求ラインの属性の設定 [S]
13341     cfg_int        割込み要求ラインの属性の設定 [TD]
13342 
```

13343 【静的API】

```

13344
13345     CFG_INT(INTNO intno, { ATR intatr, PRI intpri })
13346 
```

13351 **【C言語API】**

13352 ER ercd = cfg_int(INTNO intno, const T_CINT *pk_cint)

13353

13354 **【パラメータ】**

13355 INTNO intno 割込み番号

13356 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
13357 ケットへのポインタ（静的APIを除く）

13358

13359 * 割込み要求ラインの属性の設定情報（パケットの内容）

13360 ATR intatr 割込み要求ライン属性

13361 PRI intpri 割込み優先度

13362

13363 **【リターンパラメータ】**

13364 ER ercd 正常終了 (E_OK) またはエラーコード

13365

13366 **【エラーコード】**13367 E_CTX [s] コンテキストエラー（非タスクコンテキストからの呼出
13368 し、CPUロック状態からの呼出し）13369 E_RSATR 予約属性（intatrが不正または使用できない、属する保
13370 護ドメインがクラスが不正）13371 E_OACV [sP] オブジェクトアクセス違反（システム状態に対する管理
13372 操作が許可されていない）13373 E_MACV [sP] メモリアクセス違反（pk_cintが指すメモリ領域への読出
13374 しアクセスが許可されていない）

13375 E_PAR パラメータエラー（intno, intpriが不正）

13376 E_OBJ オブジェクト状態エラー（対象割込み要求ラインに対し
13377 てすでに属性が設定されている：CFG_INTの場合、カーネ
13378 ル管理の割込みか否かとintpriの値が整合していない）

13379

13380 **【機能】**

13381

13382 intnoで指定した割込み要求ライン（対象割込み要求ライン）に対して、各パラ
13383 メータで指定した属性を設定する。

13384

13385 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
13386 た場合にクリアされ、指定しない場合にセットされる。

13387

13388 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。

13389

13390 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
13391 が連動して設定される場合がある。

13392

13393 CFG_INTにおいて、対象割込み要求ラインに対してすでに属性が設定されている
13394 場合（言い換えると、同じ割込み番号に対するCFG_INTが複数ある場合）には、
13395 E_OBJエラーとなる。

13396

13397 intpriに指定できる値は、基本的には、TMIN_INTPRI以上、TMAX_INTPRI以下の

13398 値である。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

13399 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

1400 ができる。このように拡張されている場合、カーネル管理外の割込み要求ライ

13401 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
13402 ラーとなる。逆に、カーネル管理の割込み要求ラインを対象として、intpriが
13403 TMIN_INTPRIよりも小さい値である場合にも、E_OBJエラーとなる。

13404

13405 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
13406 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
13407 した場合にはE_PARエラーとなる。ここで、設定できない割込み要求ライン属性／
13408 割込み優先度には、ターゲット定義の制限によって設定できない値も含む。ま
13409 た、マルチプロセッサ対応カーネルにおいて、cfg_intを呼び出したタスクが割
13410 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
13411 い場合も、これに該当する。

13412

13413 保護機能対応カーネルにおいて、CFG_INTは、カーネルドメインの囲みの中に記
13414 述しなければならない。そうでない場合には、E_RSATRエラーとなる。また、
13415 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
13416 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。
13417 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
13418 ラーは検出されない。

13419

13420 マルチプロセッサ対応カーネルで、CFG_INTの記述が、対象割込み要求ラインに
13421 対して登録された割込みサービスルーチン（または対象割込み番号に対応する
13422 割込みハンドラ番号に対して登録された割込みハンドラ）と異なるクラスの囲
13423 み中にある場合には、E_RSATRエラーとなる。

13424

13425 **【補足説明】**

13426

13427 ターゲット定義の制限によって設定できない割込み要求ライン属性／割込み優
13428 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
13429 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、
13430 変更できないケースが考えられる。

13431

13432 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
13433 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
13434 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
13435 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
13436 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
13437 合には、E_PARエラーとなる。

13438

13439 **【TOPPERS/ASPカーネルにおける規定】**

13440

13441 ASPカーネルでは、CFG_INTのみをサポートする。

13442

13443 **【TOPPERS/FMPカーネルにおける規定】**

13444

13445 FMPカーネルでは、CFG_INTのみをサポートする。

13446

13447 **【TOPPERS/HRP2カーネルにおける規定】**

13448

13449 HRP2カーネルでは、CFG_INTのみをサポートする。

13450

13451 【TOPPERS/SSPカーネルにおける規定】

13452

13453 SSPカーネルでは、CFG_INTのみをサポートする。

13454

13455 【μ ITRON4.0仕様との関係】

13456

13457 μ ITRON4.0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。

13458

13459 CRE_ISR 割込みサービスルーチンの生成 [S]

13460 ATT_ISR 割込みサービスルーチンの追加 [S]

13461 acre_isr 割込みサービスルーチンの生成 [TD]

13462

13463 【静的API】

13464 CRE_ISR(ID isrid, { ATR isratr, intptr_t exinf,

13465 INTNO intno, ISR isr, PRI isrpri })

13466 ATT_ISR({ ATR isratr, intptr_t exinf, INTNO intno, ISR isr, PRI isrpri })

13467

13468 【C言語API】

13469 ER_ID isrid = acre_isr(const T_CISR *pk_cisr)

13470

13471 【パラメータ】

13472 ID isrid 対象割込みサービスルーチンのID番号 (CRE_ISR
13473 の場合)

13474 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
13475 ケットへのポインタ (静的APIを除く)

13476

13477 *割込みサービスルーチンの生成情報 (パケットの内容)

13478 ATR isratr 割込みサービスルーチン属性

13479 intptr_t exinf 割込みサービスルーチンの拡張情報

13480 INTNO intno 割込みサービスルーチンを登録する割込み番号

13481 ISR isr 割込みサービスルーチンの先頭番地

13482 PRI isrpri 割込みサービスルーチン優先度

13483

13484 【リターンパラメータ】

13485 ER_ID isrid 生成された割込みサービスルーチンのID番号 (正
13486 の値) またはエラーコード

13487

13488 【エラーコード】

13489 E_CTX [s] コンテキストエラー (非タスクコンテキストからの呼出
13490 し, CPUロック状態からの呼出し)

13491 E_RSATR 予約属性 (isratrが不正または使用できない, 属する保
13492 護ドメインかクラスが不正)

13493 E_PAR パラメータエラー (intno, isr, isrpriが不正)

13494 E_OACV [sP] オブジェクトアクセス違反 (システム状態に対する管理
13495 操作が許可されていない)

13496 E_MACV [sP] メモリアクセス違反 (pk_cisrが指すメモリ領域への読出
13497 しアクセスが許可されていない)

13498 E_NOID [sD] ID番号不足 (割り付けられる割込みサービスルーチンID
13499 がない)

13500 E_OBJ オブジェクト状態エラー (isridで指定した割込みサービ

スルーチンが登録済み：CRE_ISRの場合、その他の条件については機能の項を参照すること)

【機能】

各パラメータで指定した割込みサービスルーチン生成情報に従って、割込みサービスルーチンを生成する。

ATT_ISRによって生成された割込みサービスルーチンは、ID番号を持たない。

intnoで指定した割込み要求ラインの属性が設定されていない場合には、E_OBJエラーとなる。また、intnoで指定した割込み番号に対応する割込みハンドラ番号に対して、割込みハンドラを定義する機能 (DEF_INH, def_inh) によって割込みハンドラが定義されている場合にも、E_OBJエラーとなる。さらに、intnoでカーネル管理外の割込みを指定した場合にも、E_OBJエラーとなる。

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。

保護機能対応カーネルにおいて、CRE_ISRおよびATT_ISRは、カーネルドメインの囲みの中に記述しなければならない。そうでない場合には、E_RSATRエラーとなる。また、acre_isrで、生成する割込みサービスルーチンが属する保護ドメインとしてカーネルドメイン以外を指定した場合には、E_RSATRエラーとなる。

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。また、intnoで指定した割込み要求ラインに対して登録済みの割込みサービスルーチンがある場合に、生成する割込みサービスルーチンがそれと異なるクラスに属する場合にも、E_RSATRエラーとなる。さらに、ターゲット定義で、割込みサービスルーチンが属することができるクラスに制限がある場合がある。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。

isrpriは、TMIN_ISRPRI以上、TMAX_ISRPRI以下でなければならない。

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。

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、ATT_ISRのみをサポートする。ただし、動的生成機能拡張パッケージでは、acre_isrもサポートする。

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、ATT_ISRのみをサポートする。

【TOPPERS/HRP2カーネルにおける規定】

13551 HRP2カーネルでは、ATT_ISRのみをサポートする。

13552

13553 **【TOPPERS/SSPカーネルにおける規定】**

13554

13555 SSPカーネルでは、ATT_ISRのみをサポートする。

13556

13557 **【 μ ITRON4.0仕様との関係】**

13558

13559 割込みサービスルーチンの生成情報に、isrpri（割込みサービスルーチンの割
13560 込み優先度）を追加した。CRE_ISRは、 μ ITRON4.0仕様に定義されていない静的
13561 APIである。

13562

13563 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]

13564

13565 **【静的API】**

13566 AID_ISR(uint_t noisr)

13567

13568 **【パラメータ】**

13569 uint_t noisr 割付け可能な割込みサービスルーチンIDの数

13570

13571 **【エラーコード】**

13572 E_RSATR 予約属性（属する保護ドメインまたはクラスが不正）

13573

13574 **【機能】**

13575

13576 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
13577 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
13578 て確保する。

13579

13580 noisrは整数定数式パラメータである。

13581

13582 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]

13583 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]

13584

13585 **【静的API】**

13586 SAC_ISR(ID isrid, { ACPTN acptn1, ACPTN acptn2,
13587 ACPTN acptn3, ACPTN acptn4 })

13588

13589 **【C言語API】**

13590 ER ercd = sac_isr(ID isrid, const ACVCT *p_acvct)

13591

13592

13593 **【パラメータ】**

13594 ID isrid 対象割込みサービスルーチンのID番号

13595 ACVCT * p_acvct アクセス許可ベクタを入れたパケットへのポ
13596 インタ（静的APIを除く）

13597

13598 *アクセス許可ベクタ（パケットの内容）

13599 ACPTN acptn1 通常操作1のアクセス許可パターン

13600 ACPTN acptn2 通常操作2のアクセス許可パターン

13601	ACPTN	acptn3	管理操作のアクセス許可パターン
13602	ACPTN	acptn4	参照操作のアクセス許可パターン
13603			
13604	【リターンパラメータ】		
13605	ER	ercd	正常終了 (E_OK) またはエラーコード
13606			
13607	【エラーコード】		
13608	E_CTX [s]		コンテキストエラー (非タスクコンテキストからの呼出し, CPUロック状態からの呼出し)
13609			
13610	E_ID		不正ID番号 (isridが不正)
13611	E_RSATR		予約属性 (属する保護ドメインかクラスが不正: SAC_ISRの場合)
13612			
13613	E_NOEXS [D]		オブジェクト未登録 (対象割込みサービスルーチンが未登録)
13614			
13615	E_OACV [sP]		オブジェクトアクセス違反 (対象割込みサービスルーチンに対する管理操作が許可されていない)
13616			
13617	E_MACV [sP]		メモリアクセス違反 (p_acvctが指すメモリ領域への読出しアクセスが許可されていない)
13618			
13619	E_OBJ		オブジェクト状態エラー (対象割込みサービスルーチンは静的APIで生成された: sac_isrの場合, 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: SAC_ISRの場合)
13620			
13621			
13622			
13623			
13624	【機能】		
13625			
13626	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.		
13627			
13628			
13629			
13630	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.		
13631			
13632			
13633	SAC_ISRは, 対象割込みサービスルーチンが属する保護ドメイン (この仕様ではカーネルドメインに限られる) の囲みの中に記述しなければならない. そうでない場合には, E_RSATRエラーとなる.		
13634			
13635			
13636			
13637	【TOPPERS/ASPカーネルにおける規定】		
13638			
13639	ASPカーネルでは, SAC_ISR, sac_isrをサポートしない.		
13640			
13641	【TOPPERS/FMPカーネルにおける規定】		
13642			
13643	FMPカーネルでは, SAC_ISR, sac_isrをサポートしない.		
13644			
13645	【TOPPERS/HRP2カーネルにおける規定】		
13646			
13647	HRP2カーネルでは, SAC_ISR, sac_isrをサポートしない.		
13648			
13649	【TOPPERS/SSPカーネルにおける規定】		
13650			

13651 SSPカーネルでは、SAC_ISR, sac_isrをサポートしない.

13652

13653 **【未決定事項】**

13654

13655 割込みサービスルーチンのアクセス許可ベクタを設けず、システム状態のアク
13656 セス許可ベクタでアクセス保護する方法も考えられる.

13657 -----

13658 del_isr 割込みサービスルーチンの削除 [TD]

13659

13660 **【C言語API】**

13661 ER ercd = del_isr(ID isrid)

13662

13663 **【パラメータ】**

13664 ID isrid 対象割込みサービスルーチンのID番号

13665

13666 **【リターンパラメータ】**

13667 ER ercd 正常終了 (E_OK) またはエラーコード

13668

13669 **【エラーコード】**

13670 E_CTX コンテキストエラー (非タスクコンテキストからの呼出
13671 し, CPUロック状態からの呼出し)

13672 E_ID 不正ID番号 (cycidが不正)

13673 E_NOEXS [D] オブジェクト未登録 (対象割込みサービスルーチンが未
13674 登録)

13675 E_OACV [P] オブジェクトアクセス違反 (対象割込みサービスルーチ
13676 ンに対する管理操作が許可されていない)

13677 E_OBJ オブジェクト状態エラー (対象割込みサービスルーチン
13678 は静的APIで生成された)

13679

13680 **【機能】**

13681

13682 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
13683 除する. 具体的な振舞いは以下の通り.

13684

13685 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
13686 IDが未使用の状態に戻される.

13687

13688 **【TOPPERS/ASPカーネルにおける規定】**

13689

13690 ASPカーネルでは, del_isrをサポートしない. ただし, 動的生成機能拡張パッ
13691 ケージでは, del_isrをサポートする.

13692

13693 **【TOPPERS/FMPカーネルにおける規定】**

13694

13695 FMPカーネルでは, del_isrをサポートしない.

13696

13697 **【TOPPERS/HRP2カーネルにおける規定】**

13698

13699 HRP2カーネルでは, del_isrをサポートしない.

13700

13701 【TOPPERS/SSPカーネルにおける規定】

13702

13703 SSPカーネルでは、del_isrをサポートしない。

13704

13705 ref_isr 割込みサービスルーチンの状態参照 [T]

13706

13707 【C言語API】

13708 ER ercd = ref_isr(ID isrid, T_RISR *pk_risr)

13709

13710 ☆未完成

13711

13712 【TOPPERS/ASPカーネルにおける規定】

13713

13714 ASPカーネルでは、ref_isrをサポートしない。

13715

13716 【TOPPERS/FMPカーネルにおける規定】

13717

13718 FMPカーネルでは、ref_isrをサポートしない。

13719

13720 【TOPPERS/HRP2カーネルにおける規定】

13721

13722 HRP2カーネルでは、ref_isrをサポートしない。

13723

13724 【TOPPERS/SSPカーネルにおける規定】

13725

13726 SSPカーネルでは、ref_isrをサポートしない。

13727

13728 DEF_INH 割込みハンドラの定義 [S]

13729 def_inh 割込みハンドラの定義 [TD]

13730

13731 【静的API】

13732 DEF_INH(INHNO inhno, { ATR inhatr, INTHDR inthdr })

13733

13734 【C言語API】

13735 ER ercd = def_inh(INHNO inhno, const T_DINH *pk_dinh)

13736

13737 【パラメータ】

13738 INHNO inhno 割込みハンドラ番号

13739 T_DINH * pk_dinh 割込みハンドラの定義情報を入れたパケットへのポインタ（静的APIを除く）

13740

13741 *割込みハンドラの定義情報（パケットの内容）

13742 ATR inhatr 割込みハンドラ属性

13743 INTHDR inthdr 割込みハンドラの先頭番地

13744

13745 【リターンパラメータ】

13746 ER ercd 正常終了 (E_OK) またはエラーコード

13747

13748 【エラーコード】

13749 E_CTX [s] コンテキストエラー（非タスクコンテキストからの呼出

13750

13751		し、CPUロック状態からの呼出し)
13752	E_RSATR	予約属性 (inhatrが不正または使用できない、属する保護ドメインかクラスが不正)
13753		
13754	E_OACV [sP]	オブジェクトアクセス違反 (システム状態に対する管理操作が許可されていない)
13755		
13756	E_MACV [sP]	メモリアクセス違反 (pk_dinhが指すメモリ領域への読出しアクセスが許可されていない)
13757		
13758	E_PAR	パラメータエラー (inhno, inthdrが不正)
13759	E_OBJ	オブジェクト状態エラー (条件については機能の項を参照すること)
13760		
13761		
13762	【機能】	
13763		
13764		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
13765		パラメータで指定した割込みハンドラ定義情報に従って、割込みハンドラを定義する。ただし、def_inhにおいてpk_dinhをNULLにした場合には、対象割込み
13766		ハンドラ番号に対する割込みハンドラの定義を解除する。
13767		
13768		
13769		静的APIにおいては、inhnoとinhatrは整数定数式パラメータ、inthdrは一般定
13770		数式パラメータである。
13771		
13772		割込みハンドラを定義する場合 (DEF_INHの場合およびdef_inhにおいて
13773		pk_dinhをNULL以外にした場合) には、次のエラーが検出される。
13774		
13775		対象割込みハンドラ番号に対応する割込み要求ラインの属性が設定されていない場合には、E_OBJエラーとなる。また、対象割込みハンドラ番号に対してすでに割込みハンドラが定義されている場合と、対象割込みハンドラ番号に対応する割込み番号を対象に割込みサービスルーチンが登録されている場合にも、E_OBJエラーとなる。
13776		
13777		
13778		
13779		
13780		
13781		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
13782		を定義できる場合には、次のエラーが検出される。カーネル管理外の割込みハンドラを対象として、inhatrにTA_NONKERNELを指定しない場合には、E_OBJエラーとなる。逆に、カーネル管理の割込みハンドラを対象として、inhatrにTA_NONKERNELを指定した場合にも、E_OBJエラーとなる。また、ターゲット定義でカーネル管理外に固定されている割込みハンドラがある場合には、それを対象割込みハンドラに指定して、inhatrにTA_NONKERNELを指定しない場合には、E_RSATRエラーとなる。逆に、ターゲット定義でカーネル管理に固定されている割込みハンドラがある場合には、それを対象割込みハンドラに指定して、inhatrにTA_NONKERNELを指定した場合には、E_RSATRエラーとなる。
13783		
13784		
13785		
13786		
13787		
13788		
13789		
13790		
13791		
13792		保護機能対応カーネルにおいて、DEF_INHは、カーネルドメインの囲みの中に記述しなければならない。そうでない場合には、E_RSATRエラーとなる。また、def_inhで割込みハンドラを定義する場合には、割込みハンドラの属する保護ドメインを設定する必要はなく、割込みハンドラ属性にTA_DOM(domid)を指定した場合にはE_RSATRエラーとなる。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。
13793		
13794		
13795		
13796		
13797		
13798		
13799		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、
13800		

13801 E_RSATRエラーとなる。また、ターゲット定義で、割込みハンドラが属すること
 13802 ができるクラスに制限がある場合がある。登録する割込みハンドラの属するク
 13803 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。

13804
 13805 割込みハンドラの定義を解除する場合（def_inhにおいてpk_dinhをNULLにした
 13806 場合）で、対象割込みハンドラ番号に対して割込みハンドラが定義されてい
 13807 ない場合には、E_OBJエラーとなる。また、対象割込みハンドラ番号に対して定義
 13808 された割込みハンドラが、静的APIで定義されたものである場合には、ターゲッ
 13809 ト定義でE_OBJエラーとなる場合がある。

13810
 13811 ターゲット定義で、対象割込みハンドラを定義（または定義解除）できない場
 13812 合には、E_PARエラーとなる。具体的には、マルチプロセッサ対応カーネルにお
 13813 いて、def_inhを呼び出したタスクが割り付けられているプロセッサから、対象
 13814 割込みハンドラを定義（または定義解除）できない場合が、これに該当する。

13815
 13816 静的APIにおいて、inthdrが不正である場合にE_PARエラーが検出されるか否か
 13817 は、ターゲット定義である。

13818
 13819 **【TOPPERS/ASPカーネルにおける規定】**

13820
 13821 ASPカーネルでは、DEF_INHのみをサポートする。

13822
 13823 **【TOPPERS/FMPカーネルにおける規定】**

13824
 13825 FMPカーネルでは、DEF_INHのみをサポートする。

13826
 13827 **【TOPPERS/HRP2カーネルにおける規定】**

13828
 13829 HRP2カーネルでは、DEF_INHのみをサポートする。

13830
 13831 **【TOPPERS/SSPカーネルにおける規定】**

13832
 13833 SSPカーネルでは、DEF_INHのみをサポートする。

13834
 13835 **【 μ ITRON4.0仕様との関係】**

13836
 13837 inthdrのデータ型をINTHDRに変更した。

13838
 13839 def_inhによって定義済みの割込みハンドラを再定義しようとした場合に、
 13840 E_OBJエラーとすることにした。割込みハンドラの定義を変更するには、一度定
 13841 義を解除してから、再度定義する必要がある。

13842 -----
 13843 dis_int 割込みの禁止 [T]

13844
 13845 **【C言語API】**
 13846 ER ercd = dis_int(INTNO intno)

13847
 13848 **【パラメータ】**
 13849 INTNO intno 割込み番号

13850

13851 **【リターンパラメータ】**

13852 ER ercd 正常終了 (E_OK) またはエラーコード

13853

13854 **【エラーコード】**

13855 E_CTX コンテキストエラー (非タスクコンテキストからの呼出し)

13856

13857 E_NOSPT 未サポートエラー (dis_intがサポートされていない)

13858 E_PAR パラメータエラー (intnoが不正, その他の条件については機能の項を参照すること)

13859 E_OACV [P] オブジェクトアクセス違反 (システム状態に対する通常操作2が許可されていない)

13860 E_OBJ オブジェクト状態エラー (intnoで指定した割込み要求ラインに対して割込み要求ライン属性が設定されていない)

13861

13862 **【機能】**

13863

13864 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.

13865

13866 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. 具体的には, 対象割込み要求ラインに対して割込み要求禁止フラグがサポートされていない場合や, マルチプロセッサ対応カーネルにおいて, dis_intを呼び出したタスクが割り付けられているプロセッサから, 対象割込み要求ラインの割込み要求禁止フラグが操作できない場合が, これに該当する.

13867

13868 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. 特にマルチプロセッサ対応カーネルでは, あるプロセッサから dis_intを呼び出して割込み要求禁止フラグをセットしても, 他のプロセッサに対しては割込みがマスクされない場合がある.

13869

13870 ターゲット定義で, dis_intがサポートされていない場合がある. dis_intがサポートされている場合には, TOPPERS_SUPPORT_DIS_INTがマクロ定義される. サポートされていない場合にdis_intを呼び出すと, E_NOSPTエラーが返るか, リンク時にエラーとなる.

13871

13872 **【μITRON4.0仕様との関係】**

13873

13874 μITRON4.0仕様で実装定義としていたintnoの意味を標準化した.

13875

13876 CPUロック状態でも呼び出せるものとした.

13877

13878 -----
13879 ena_int 割込みの許可 [T]

13880

13881 **【C言語API】**

13882 ER ercd = ena_int(INTNO intno)

13883

13884 **【パラメータ】**

13885 INTNO intno 割込み番号

13886

13901 **【リターンパラメータ】**

13902 ER ercd 正常終了 (E_OK) またはエラーコード

13903

13904 **【エラーコード】**

13905 E_CTX コンテキストエラー (非タスクコンテキストからの呼出し)

13906

13907 E_NOSPT 未サポートエラー (ena_intがサポートされていない)

13908

13909 E_PAR パラメータエラー (intnoが不正, その他の条件については機能の項を参照すること)

13910

13911 E_OACV [P] オブジェクトアクセス違反 (システム状態に対する通常操作2が許可されていない)

13912

13913 E_OBJ オブジェクト状態エラー (intnoで指定した割込み要求ラインに対して割込み要求ライン属性が設定されていない)

13914

13915 **【機能】**

13916

13917 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.

13918

13919
13920 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. 具体的には, 対象割込み要求ラインに対して割込み要求禁止フラグがサポートされていない場合や, マルチプロセッサ対応カーネルにおいて, ena_intを呼び出したタスクが割り付けられているプロセッサから, 対象割込み要求ラインの割込み要求禁止フラグが操作できない場合が, これに該当する.

13926

13927 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. 特にマルチプロセッサ対応カーネルでは, あるプロセッサから ena_intを呼び出して割込み要求禁止フラグをクリアしても, 他のプロセッサに対しては割込みがマスク解除されない場合がある.

13931

13932 ターゲット定義で, ena_intがサポートされていない場合がある. ena_intがサポートされている場合には, TOPPERS_SUPPORT_ENA_INTがマクロ定義される. サポートされていない場合にena_intを呼び出すと, E_NOSPTエラーが返るか, リンク時にエラーとなる.

13936

13937 **【μITRON4.0仕様との関係】**

13938

13939 μITRON4.0仕様で実装定義としていたintnoの意味を標準化した.

13940

13941 CPUロック状態でも呼び出せるものとした.

13942

13943 ref_int 割込み要求ラインの参照 [T]

13944

13945 **【C言語API】**

13946 ER ercd = ref_int(INTNO intno, T_RINT *pk_rint)

13947

13948 ☆未完成

13949

13950 **【TOPPERS/ASPカーネルにおける規定】**

13951
13952 ASPカーネルでは、ref_intをサポートしない。
13953
13954 **【TOPPERS/FMPカーネルにおける規定】**
13955
13956 FMPカーネルでは、ref_intをサポートしない。
13957
13958 **【TOPPERS/HRP2カーネルにおける規定】**
13959
13960 HRP2カーネルでは、ref_intをサポートしない。
13961
13962 **【TOPPERS/SSPカーネルにおける規定】**
13963
13964 SSPカーネルでは、ref_intをサポートしない。
13965
13966 **【 μ ITRON4.0仕様との関係】**
13967
13968 μ ITRON4.0仕様に定義されていないサービスコールである。
13969 -----
13970 chg_ipm 割込み優先度マスクの変更 [T]
13971
13972 **【C言語API】**
13973 ER ercd = chg_ipm(PRI intpri)
13974
13975 **【パラメータ】**
13976 PRI intpri 割込み優先度マスク
13977
13978 **【リターンパラメータ】**
13979 ER ercd 正常終了 (E_OK) またはエラーコード
13980
13981 **【エラーコード】**
13982 E_CTX コンテキストエラー (非タスクコンテキストからの呼出
13983 し、CPUロック状態からの呼出し)
13984 E_PAR パラメータエラー (intpriが不正)
13985 E_OACV [P] オブジェクトアクセス違反 (システム状態に対する通常
13986 操作2が許可されていない)
13987
13988 **【機能】**
13989
13990 割込み優先度マスクを、intpriで指定した値に変更する。
13991
13992 intpriは、TMIN_INTPRI以上、TIPM_ENAALL以下でなければならない。ただし、
13993 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
13994 ある。
13995
13996 **【補足説明】**
13997
13998 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
13999 除され、ディスパッチが起こる可能性がある。また、タスク例外処理ルーチン
14000 の実行が開始される可能性がある。

14001
14002 **【TOPPERS/SSPカーネルにおける規定】**
14003
14004 SSPカーネルでは、chg_ipmをサポートしない。
14005
14006 **【μ ITRON4.0仕様との関係】**
14007
14008 μ ITRON4.0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
14009 義となっているサービスコールである。
14010 -----
14011 get_ipm 割込み優先度マスクの参照 [T]
14012
14013 **【C言語API】**
14014 ER ercd = get_ipm(PRI *p_intpri)
14015
14016 **【パラメータ】**
14017 PRI * p_intpri 割込み優先度マスクを入れるメモリ領域へのポ
14018 インタ
14019
14020 **【リターンパラメータ】**
14021 ER ercd エラーコード
14022 PRI intpri 割込み優先度マスク
14023
14024 **【エラーコード】**
14025 E_CTX コンテキストエラー（非タスクコンテキストからの呼出
14026 し、CPUロック状態からの呼出し）
14027 E_OACV [P] オブジェクトアクセス違反（システム状態に対する参照
14028 操作が許可されていない）
14029 E_MACV [P] メモリアクセス違反（p_intpriが指すメモリ領域への書
14030 込みアクセスが許可されていない）
14031
14032 **【機能】**
14033
14034 割込み優先度マスクの現在値を参照する。
14035
14036 **【TOPPERS/SSPカーネルにおける規定】**
14037
14038 SSPカーネルでは、get_ipmをサポートしない。
14039
14040 **【μ ITRON4.0仕様との関係】**
14041
14042 μ ITRON4.0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
14043 義となっているサービスコールである。
14044 -----
14045
14046 4.10 CPU例外管理機能
14047
14048 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
14049 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。
14050

14051 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
 14052 る。
 14053
 14054 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、
 14055 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。ターゲット定
 14056 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。
 14057
 14058 TA_DIRECT CPU例外ハンドラを直接呼び出す
 14059
 14060 C言語によるCPU例外ハンドラの記述形式は次の通り。
 14061
 14062 void cpu_exception_handler(void *p_excinf)
 14063 {
 14064 CPU例外ハンドラ本体
 14065 }
 14066
 14067 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。
 14068 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
 14069 る。
 14070 -----
 14071 DEF_EXC CPU例外ハンドラの定義 [S]
 14072 def_exc CPU例外ハンドラの定義 [TD]
 14073
 14074 【静的API】
 14075 DEF_EXC(EXCNO excno, { ATR excatr, EXCHDR exchdr })
 14076
 14077 【C言語API】
 14078 ER ercd = def_exc(EXCNO excno, const T_DEXC *pk_dexc)
 14079
 14080 【パラメータ】
 14081 EXCNO excno CPU例外ハンドラ番号
 14082 T_DEXC * pk_dexc CPU例外ハンドラの定義情報を入れたパケットへ
 14083 のポインタ（静的APIを除く）
 14084
 14085 *CPU例外ハンドラの定義情報（パケットの内容）
 14086 ATR excatr CPU例外ハンドラ属性
 14087 EXCHDR exchdr CPU例外ハンドラの先頭番地
 14088
 14089 【リターンパラメータ】
 14090 ER ercd 正常終了 (E_OK) またはエラーコード
 14091
 14092 【エラーコード】
 14093 E_CTX [s] コンテキストエラー（非タスクコンテキストからの呼出
 14094 し、CPUロック状態からの呼出し）
 14095 E_RSATR 予約属性（excatrが不正または使用できない、属する保
 14096 護ドメインがクラスが不正）
 14097 E_PAR パラメータエラー（excno, exchdrが不正）
 14098 E_OACV [sP] オブジェクトアクセス違反（システム状態に対する管理
 14099 操作が許可されていない）
 14100 E_MACV [sP] メモリアクセス違反（pk_dexcが指すメモリ領域への読出

14101 しアクセスが許可されていない)
14102 E_OBJ オブジェクト状態エラー（定義済みのCPU例外ハンドラ番号
14103 に対する再定義，未定義のCPU例外ハンドラ番号に対す
14104 る定義解除）
14105
14106 **【機能】**
14107
14108 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，
14109 各パラメータで指定したCPU例外ハンドラ定義情報に従って，CPU例外ハンドラ
14110 を定義する．ただし，def_excにおいてpk_dexcをNULLにした場合には，対象
14111 CPU例外ハンドラ番号に対するCPU例外ハンドラの定義を解除する．
14112
14113 静的APIにおいては，excnoとexcatrは整数定数式パラメータ，exchdrは一般定
14114 数式パラメータである．
14115
14116 CPU例外ハンドラを定義する場合（DEF_EXCの場合およびdef_excにおいて
14117 pk_dexcをNULL以外にした場合）で，対象CPU例外ハンドラ番号に対してすでに
14118 CPU例外ハンドラが定義されている場合には，E_OBJエラーとなる．
14119
14120 保護機能対応カーネルにおいて，DEF_EXCは，カーネルドメインの囲みの中に記
14121 述しなければならない．そうでない場合には，E_RSATRエラーとなる．また，
14122 def_excでCPU例外ハンドラを定義する場合には，CPU例外ハンドラの属する保護
14123 ドメインを設定する必要はなく，CPU例外ハンドラ属性にTA_DOM(domid)を指定
14124 した場合にはE_RSATRエラーとなる．ただし，TA_DOM(TDOM_SELF)を指定した場
14125 合には，指定が無視され，E_RSATRエラーは検出されない．
14126
14127 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
14128 初期割付けプロセッサが，そのCPU例外が発生するプロセッサでない場合には，
14129 E_RSATRエラーとなる．
14130
14131 CPU例外ハンドラの定義を解除する場合（def_excにおいてpk_dexcをNULLにした
14132 場合）で，対象CPU例外ハンドラ番号に対してCPU例外ハンドラが定義されてい
14133 ない場合には，E_OBJエラーとなる．また，対象CPU例外ハンドラ番号に対して
14134 定義されたCPU例外ハンドラが，静的APIで定義されたものである場合には，ター
14135 ゲット定義でE_OBJエラーとなる場合がある．
14136
14137 静的APIにおいて，exchdrが不正である場合にE_PARエラーが検出されるか否か
14138 は，ターゲット定義である．
14139
14140 **【TOPPERS/ASPカーネルにおける規定】**
14141
14142 ASPカーネルでは，DEF_EXCのみをサポートする．
14143
14144 **【TOPPERS/FMPカーネルにおける規定】**
14145
14146 FMPカーネルでは，DEF_EXCのみをサポートする．
14147
14148 **【TOPPERS/HRP2カーネルにおける規定】**
14149
14150 HRP2カーネルでは，DEF_EXCのみをサポートする．

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、DEF_EXCのみをサポートする。

【 μ ITRON4.0仕様との関係】

def_excによって、定義済みのCPU例外ハンドラを再定義しようとした場合に、E_OBJエラーとすることにした。

xsns_dpn CPU例外発生時のディスパッチ保留状態の参照 [TI]

【C言語API】

bool_t stat = xsns_dpn(void *p_excinf)

【パラメータ】

void * p_excinf CPU例外の情報を記憶しているメモリ領域の先頭番地

【リターンパラメータ】

bool_t state ディスパッチ保留状態

【機能】

CPU例外発生時のディスパッチ保留状態を参照する。具体的な振舞いは以下の通り。

実行中のCPU例外ハンドラの起動原因となったCPU例外が、カーネル管理外のCPU例外でなく、タスクコンテキストで発生し、そのタスクがディスパッチ保留状態でなかった場合にfalse、そうでない場合にtrueが返る。

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。

p_excinfには、CPU例外ハンドラに渡されるp_excinfパラメータをそのまま渡す。

【使用方法】

xsns_dpnは、CPU例外ハンドラの中で、どのようなリカバリ処理が可能かを判別したい場合に使用する。xsns_dpnがfalseを返した場合（trueを返した場合ではないので注意すること）、非タスクコンテキスト用のサービスコールを用いてCPU例外を起こしたタスクよりも優先度の高いタスクを起動または待ち解除し、そのタスクでリカバリ処理を行うことができる。ただし、CPU例外を起こしたタスクが最高優先度の場合には、この方法でリカバリ処理を行うことはできない。

【使用上の注意】

xsns_dpnは、E_CTXエラーを返すことがないために [TI] となっているが、CPU例外ハンドラから呼び出すためのものである。CPU例外ハンドラ以外から呼び出した場合や、p_excinfに正しい値を渡さなかった場合、xsns_dpnが返す値は意

14201 味を持たない。
14202
14203 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。
14204
14205 **【TOPPERS/SSPカーネルにおける規定】**
14206
14207 SSPカーネルでは、xsns_dpnをサポートしない。
14208
14209 **【μITRON4.0仕様との関係】**
14210
14211 μITRON4.0仕様に定義されていないサービスコールである。
14212
14213 **【仕様決定の理由】**
14214
14215 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
14216 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
14217 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
14218 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
14219 的に禁止している。
14220 -----
14221 xsns_xpn CPU例外発生時のタスク例外処理保留状態の参照 [TI]
14222
14223 **【C言語API】**
14224 bool_t stat = xsns_xpn(void *p_excinf)
14225
14226 **【パラメータ】**
14227 void * p_excinf CPU例外の情報を記憶しているメモリ領域の先頭
14228 番地
14229
14230 **【リターンパラメータ】**
14231 bool_t state タスク例外処理保留状態
14232
14233 **【機能】**
14234
14235 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
14236 参照する。具体的な振舞いは以下の通り。
14237
14238 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
14239 CPU例外でなく、タスクコンテキストで発生し、そのタスクがタスク例外処理ルー
14240 チンを実行開始できる状態であった場合にfalse、そうでない場合にtrueが返る。
14241
14242 保護機能対応カーネルにおいて、CPU例外が発生したタスクがユーザタスクの場
14243 合には、ユーザスタック領域の残りが少なく、タスク例外処理ルーチンを実行
14244 開始できない（タスク例外処理ルーチンを実行開始しようとする、タスク例
14245 外実行開始時スタック不正例外が発生する）場合にも、trueを返す。
14246
14247 保護機能対応のカーネルにおいて、xsns_xpnをタスクコンテキストから呼び出
14248 した場合には、trueが返る。
14249
14250 p_excinfには、CPU例外ハンドラに渡されるp_excinfパラメータをそのまま渡す。

【使用方法】

xsns_xpnは、CPU例外ハンドラの中で、どのようなリカバリ処理が可能かを判別したい場合に使用する。xsns_xpnがfalseを返した場合（trueを返した場合ではないので注意すること）、非タスクコンテキスト用のサービスコールを用いてCPU例外を起こしたタスクにタスク例外を要求し、タスク例外処理ルーチンでリカバリ処理を行うことができる。

【使用上の注意】

xsns_xpnは、E_CTXエラーを返すことがないために〔TI〕となっているが、CPU例外ハンドラから呼び出すためのものである。CPU例外ハンドラ以外から呼び出した場合や、p_excinfに正しい値を渡さなかった場合、xsns_xpnが返す値は意味を持たない。

どちらの条件でtrueが返るか間違いやすいので注意すること。

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、xsns_xpnをサポートしない。

【 μ ITRON4.0仕様との関係】

μ ITRON4.0仕様に定義されていないサービスコールである。

【仕様決定の理由】

保護機能対応のカーネルにおいては、xsns_xpnをユーザドメインから呼び出すことは禁止すべきである。ユーザドメインの実行中は、必ずタスクコンテキストであるため、xsns_xpnをタスクコンテキストから呼び出した場合に必ずtrueを返す仕様とすることで、xsns_xpnをユーザドメインから呼び出すことを実質的に禁止している。

4.11 拡張サービスコール管理機能

拡張サービスコールは、非特権モードで実行される処理単位から、特権モードで実行すべきルーチンを呼び出すための機能である。特権モードで実行するルーチンを、拡張サービスコールと呼ぶ。拡張サービスコールは、特権モードで実行される処理単位からも呼び出すことができる。

保護機能対応カーネルにおいて、拡張サービスコールは、カーネルドメインに属する。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に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。そのため、拡張サービスコールからは、すべてのカーネルオブジェクトに対して、すべての種別のアクセスを行うことが許可される。

保護機能対応でないカーネルでは、非特権モードと特権モードの区別がないた

14301 め、拡張サービスコール管理機能をサポートしない。

14302

14303 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。

14304

```
14305     ER_UINT extended_svc(intptr_t par1, intptr_t par2, intptr_t par3,  
14306                           intptr_t par4, intptr_t par5, ID cdmid)  
14307     {  
14308         拡張サービスコール本体  
14309     }
```

14310

14311 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
14312 ID番号が渡される。すなわち、拡張サービスコールから呼び出した場合には
14313 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
14314 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。

14315

14316 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。

14317

14318 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。

14319

```
14320     TMAX_FNCD      拡張サービスコールの機能番号の最大値（動的生成対応  
14321                     カーネルでは、登録できる拡張サービスコールの数に一  
14322                     致）
```

14323

14324 **【TOPPERS/ASPカーネルにおける規定】**

14325

14326 ASPカーネルでは、拡張サービスコール管理機能をサポートしない。

14327

14328 **【TOPPERS/FMPカーネルにおける規定】**

14329

14330 FMPカーネルでは、拡張サービスコール管理機能をサポートしない。

14331

14332 **【TOPPERS/HRP2カーネルにおける規定】**

14333

14334 HRP2カーネルでは、拡張サービスコール管理機能をサポートする。

14335

14336 **【TOPPERS/SSPカーネルにおける規定】**

14337

14338 SSPカーネルでは、拡張サービスコール管理機能をサポートしない。

14339

14340 **【未決定事項】**

14341

14342 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
14343 は未決定である。

14344

14345 **【μ ITRON4.0仕様との関係】**

14346

14347 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
14348 メータ5個に固定した。

14349

14350 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

14351 番号を渡す機能を追加した。

14352

14353 TMAX_FNCDは、 μ ITRON4.0仕様に規定されていないカーネル構成マクロである。

14354

14355 DEF_SVC 拡張サービスコールの定義 [SP]

14356 def_svc 拡張サービスコールの定義 [TPD]

14357

14358 **【静的API】**

14359 DEF_SVC(FN fncd, { ATR svcatr, EXTSVC extsvc, SIZE stksz })

14360

14361 **【C言語API】**

14362 ER ercd = def_svc(FN fncd, const T_DSVC *pk_dsvc)

14363

14364 **【パラメータ】**

14365 FN fncd 拡張サービスコールの機能コード

14366 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

14367 トへのポインタ（静的APIを除く）

14368

14369 * 拡張サービスコールの定義情報（パケットの内容）

14370 ATR svcatr 拡張サービスコール属性

14371 EXTSVC extsvc 拡張サービスコールの先頭番地

14372 SIZE stksz 拡張サービスコールで使用するスタックサイズ

14373

14374 **【リターンパラメータ】**

14375 ER ercd 正常終了 (E_OK) またはエラーコード

14376

14377 **【エラーコード】**

14378 E_CTX [s] コンテキストエラー（非タスクコンテキストからの呼出し、CPUロック状態からの呼出し）

14379 E_RSATR 予約属性（svcatrが不正または使用できない、属する保護ドメインかクラスが不正）

14380 E_PAR パラメータエラー（fncd, extsvcが不正）

14381 E_OACV [sP] オブジェクトアクセス違反（システム状態に対する管理操作が許可されていない）

14382 E_MACV [sP] メモリアクセス違反（pk_dsvcが指すメモリ領域への読出しアクセスが許可されていない）

14383 E_OBJ オブジェクト状態エラー（定義済みの機能コードに対する再定義、未定義の機能コードに対する定義解除）

14384

14385 **【機能】**

14386

14387 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

14388 した拡張サービスコール定義情報に従って、拡張サービスコールを定義する。

14389 ただし、def_svcにおいてpk_dsvcをNULLにした場合には、対象機能コードに対

14390 する拡張サービスコールの定義を解除する。

14391

14392 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。

14393

14394 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

14401 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
 14402 スコールが定義されている場合には、E_OBJエラーとなる。
 14403

14404 DEF_SVCは、カーネルドメインの囲みの中に記述しなければならない。そうでな
 14405 い場合には、E_RSATRエラーとなる。また、def_svcで拡張サービスコールを定
 14406 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
 14407 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
 14408 ラーとなる。ただし、TA_DOM(TDOM_SELF)を指定した場合には、指定が無視され、
 14409 E_RSATRエラーは検出されない。
 14410

14411 マルチプロセッサ対応カーネルでは、DEF_SVCは、クラスの囲みの外に記述しな
 14412 なければならない。そうでない場合には、E_RSATRエラーとなる。また、def_svc
 14413 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
 14414 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
 14415 場合にはE_RSATRエラーとなる。ただし、TA_CLS(TCLS_SELF)を指定した場合に
 14416 は、指定が無視され、E_RSATRエラーは検出されない。
 14417

14418 拡張サービスコールの定義を解除する場合 (def_svcにおいてpk_dsvcをNULLに
 14419 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
 14420 い場合には、E_OBJエラーとなる。
 14421

14422 拡張サービスコールの機能コードには、正の値を用いる。fncdが0または負の値
 14423 の場合には、E_PARエラーとなる。また、fncdがTMAX_FNCDよりも大きい場合
 14424 にも、E_PARエラーとなる。
 14425

14426 **【TOPPERS/HRP2カーネルにおける規定】**
 14427

14428 HRP2カーネルでは、DEF_SVCのみをサポートする。
 14429

14430 **【μITRON4.0仕様との関係】**
 14431

14432 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
 14433 ックサイズ)を追加した。
 14434

14435 extsvcのデータ型を、EXTSVCに変更した。
 14436 -----

14437 cal_svc 拡張サービスコールの呼出し [TIP]
 14438

14439 **【C言語API】**
 14440 ER_UINT ercd = cal_svc(FN fcnd, intptr_t par1, intptr_t par2,
 14441 intptr_t par3, intptr_t par4, intptr_t par5)
 14442

14443 **【パラメータ】**

14444	FN	fncd	呼び出す拡張サービスコールの機能コード
14445	intptr_t	par1	拡張サービスコールへの第1パラメータ
14446	intptr_t	par2	拡張サービスコールへの第2パラメータ
14447	intptr_t	par3	拡張サービスコールへの第3パラメータ
14448	intptr_t	par4	拡張サービスコールへの第4パラメータ
14449	intptr_t	par5	拡張サービスコールへの第5パラメータ
14450			

14451 **【リターンパラメータ】**
14452 ER_UINT ercd 正常終了（正の値または0）またはエラーコード
14453
14454 **【エラーコード】**
14455 E_SYS システムエラー（拡張サービスコールのネストレベルが
14456 上限を超える）
14457 E_RSFN 予約機能コード（fncdが不正．fncdに対して拡張サービ
14458 スコールが定義されていない）
14459 E_NOMEM メモリ不足（スタックの残り領域が不足）
14460 *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。
14461
14462 **【機能】**
14463
14464 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
14465 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。
14466
14467 fncdが不正な値である場合や、fncdで指定した機能コードに対して拡張サービ
14468 スコールが定義されていない場合には、E_RSFNエラーとなる。
14469
14470 また、タスクコンテキストから呼び出した場合には、次のエラーが検出される。
14471 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
14472 小さい場合には、E_NOMEMエラーとなる。また、拡張サービスコールのネストレ
14473 ベルが上限（255）を超える場合には、E_SYSエラーが返る。
14474
14475 **【μ ITRON4.0仕様との関係】**
14476
14477 μ ITRON4.0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
14478 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
14479 出せないこととした。
14480
14481 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
14482 クする機能を追加した。
14483
14484 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
14485 し、cal_svcから返るエラー（E_SYS, E_RSFN, E_NOMEM）について規定した。
14486
14487 **【仕様決定の理由】**
14488
14489 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
14490 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
14491 うにするためである。
14492 -----
14493
14494 4.12 システム構成管理機能
14495
14496 システム構成管理機能には、非タスクコンテキスト用スタック領域を設定する
14497 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
14498 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。
14499
14500 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

14501 処理単位が用いるスタック領域である。
14502
14503 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、
14504 カーネルの用いるオブジェクト管理領域と同様に扱われる。
14505
14506 初期化ルーチンは、カーネルが実行を制御する処理単位で、カーネルの動作開
14507 始の直前に、カーネル非動作状態で実行される。
14508
14509 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
14510 る。
14511
14512 初期化ルーチン属性に指定できる属性はない。そのため初期化ルーチン属性に
14513 は、TA_NULLを指定しなければならない。
14514
14515 C言語による初期化ルーチンの記述形式は次の通り。
14516
14517 void initialization_routine(intptr_t exinf)
14518 {
14519 初期化ルーチン本体
14520 }
14521
14522 exinfには、初期化ルーチンの拡張情報が渡される。
14523
14524 終了処理ルーチンは、カーネルが実行を制御する処理単位で、カーネルの動作
14525 終了の直後に、カーネル非動作状態で実行される。
14526
14527 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
14528 する。
14529
14530 終了処理ルーチン属性に指定できる属性はない。そのため終了処理ルーチン属
14531 性には、TA_NULLを指定しなければならない。
14532
14533 C言語による終了処理ルーチンの記述形式は次の通り。
14534
14535 void termination_routine(intptr_t exinf)
14536 {
14537 終了処理ルーチン本体
14538 }
14539
14540 exinfには、終了処理ルーチンの拡張情報が渡される。
14541
14542 【 μ ITRON4.0仕様との関係】
14543
14544 非タスクコンテキスト用スタック領域の設定と、終了処理ルーチンは、
14545 μ ITRON4.0仕様に規定されていない機能である。
14546 -----
14547 DEF_ICS 非タスクコンテキスト用スタック領域の設定 [S]
14548
14549 【静的API】
14550 DEF_ICS({ SIZE istksz, STK_T *istk })

14551
14552
14553
14554
14555
14556
14557
14558
14559
14560
14561
14562
14563
14564
14565
14566
14567
14568
14569
14570
14571
14572
14573
14574
14575
14576
14577
14578
14579
14580
14581
14582
14583
14584
14585
14586
14587
14588
14589
14590
14591
14592
14593
14594
14595
14596
14597
14598
14599
14600

【パラメータ】

＊非タスクコンテキスト用スタック領域の設定情報

SIZE	istksz	非タスクコンテキスト用スタック領域のサイズ (バイト数)
STK_T	istk	非タスクコンテキスト用スタック領域の先頭番地

【エラーコード】

E_RSATR	予約属性 (属する保護ドメインかクラスが不正)
E_PAR	パラメータエラー (istksz, istkが不正)
E_NOMEM	メモリ不足 (非タスクコンテキスト用スタック領域が確保できない)
E_OBJ	オブジェクト状態エラー (非タスクコンテキスト用スタック領域がすでに設定されている, その他の条件については機能の項を参照すること)

【機能】

各パラメータで指定した非タスクコンテキスト用スタック領域の設定情報に従って, 非タスクコンテキスト用スタック領域を設定する.

istkszは整数定数式パラメータ, istkは一般定数式パラメータである. コンフィギュレータは, 静的APIのメモリ不足 (E_NOMEM) エラーを検出することができない.

istkをNULLとした場合, istkszで指定したサイズのスタック領域を, コンフィギュレータが確保する. istkszにターゲット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サイズを大きい方に丸めて確保する.

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. 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.

保護機能対応カーネルでは, istkとistkszで指定した非タスクコンテキスト用のスタック領域がカーネル専用のメモリオブジェクトに含まれない場合, E_OBJエラーとなる.

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を, コンフィギュレータが確保する.

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. 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. 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.

14601 保護機能対応カーネルにおいて、DEF_ICSは、カーネルドメインの囲みの中に記
14602 述しなければならない。そうでない場合には、E_RSATRエラーとなる。

14603

14604 **【TOPPERS/SSPカーネルにおける規定】**

14605

14606 SSPカーネルでは、istkにはNULLを指定しなくてはならず、その場合でも、コン
14607 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。これ
14608 は、SSPカーネルでは、すべての処理単位が共有スタック領域を使用し、非タス
14609 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、
14610 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
14611 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。

14612

14613 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。

14614

14615 **【 μ ITRON4.0仕様との関係】**

14616

14617 μ ITRON4.0仕様に定義されていない静的APIである。

14618

14619 DEF_STK 共有スタック領域の設定 [S]

14620

14621 **【静的API】**

14622 DEF_STK({ SIZE stksz, STK_T *stk })

14623

14624 **【パラメータ】**

14625 *共有スタック領域の設定情報

14626 SIZE stksz 共有スタック領域のサイズ (バイト数)

14627 STK_T stk 共有スタック領域の先頭番地

14628

14629 **【エラーコード】**

14630 E_RSATR 予約属性 (属する保護ドメインかクラスが不正)

14631 E_PAR パラメータエラー (stksz, stkが不正)

14632 E_NOMEM メモリ不足 (共有スタック領域が確保できない)

14633 E_OBJ オブジェクト状態エラー (共有スタック領域がすでに設
14634 定されている, その他の条件について
14635 は機能の項を参照すること)

14636

14637 **【サポートするカーネル】**

14638

14639 DEF_STKは、TOPPERS/SSPカーネルのみがサポートする静的APIである。他のカー
14640 ネルは、DEF_STKをサポートしない。

14641

14642 **【機能】**

14643

14644 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
14645 領域を設定する。

14646

14647 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
14648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
14649 ない。

14650

stkをNULLとした場合、stkkszで指定したサイズのスタック領域を、コンフィギュレータが確保する。stkksz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サイズを大きい方に丸めて確保する。

stkにNULL以外を指定した場合、stkとstkksz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。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stkksz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。

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。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を、コンフィギュレータが確保する。

stkksz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。

【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていない静的APIである。

ATT_INI 初期化ルーチンの追加 [S]

【静的API】

ATT_INI({ ATR iniatr, intptr_t exinf, INIRTN inirtn })

【パラメータ】

* 初期化ルーチンの追加情報

ATR	iniatr	初期化ルーチン属性
intptr_t	exinf	初期化ルーチンの拡張情報
INIRTN	inirtn	初期化ルーチンの先頭番地

【エラーコード】

E_RSATR 予約属性 (iniatrが不正または使用できない, 属する保護ドメインが不正)

E_PAR パラメータエラー (inirtnが不正)

【機能】

各パラメータで指定した初期化ルーチン追加情報に従って、初期化ルーチンを追加する。

iniatrは整数定数式パラメータ、exinfとinirtnは一般定数式パラメータである。

保護機能対応カーネルにおいて、ATT_INIは、カーネルドメインの囲みの中に記述しなければならない。そうでない場合には、E_RSATRエラーとなる。

inirtnが不正である場合にE_PARエラーが検出されるか否かは、ターゲット定義

14701 である.

14702

14703 **【補足説明】**

14704

14705 マルチプロセッサ対応カーネルでは、クラスに属さないグローバル初期化ルー
14706 チンはマスタプロセッサで実行され、クラスに属するローカル初期化ルーチン
14707 はそのクラスの初期割付けプロセッサにより実行される.

14708 -----

14709 ATT_TER 終了処理ルーチンの追加 [S]

14710

14711 **【静的API】**

14712 ATT_TER({ ATR teratr, intptr_t exinf, TERRTN terrtn })

14713

14714 **【パラメータ】**

14715 * 終了処理ルーチンの追加情報

14716 ATR teratr 終了処理ルーチン属性

14717 intptr_t exinf 終了処理ルーチンの拡張情報

14718 TERRTN terrtn 終了処理ルーチンの先頭番地

14719

14720 **【エラーコード】**

14721 E_RSATR 予約属性 (teratrが不正または使用できない, 属する保
14722 護ドメインが不正)

14723 E_PAR パラメータエラー (terrtnが不正)

14724

14725 **【機能】**

14726

14727 各パラメータで指定した終了処理ルーチン追加情報に従って, 終了処理ルーチ
14728 ンを追加する.

14729

14730 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.

14731

14732 保護機能対応カーネルにおいて, ATT_TERは, カーネルドメインの囲みの中に記
14733 述しなければならない. そうでない場合には, E_RSATRエラーとなる.

14734

14735 terrtnが不正である場合にE_PARエラーが検出されるか否かは, ターゲット定義
14736 である.

14737

14738 **【補足説明】**

14739

14740 マルチプロセッサ対応カーネルでは、クラスに属さないグローバル終了処理ルー
14741 チンはマスタプロセッサで実行され、クラスに属するローカル終了処理ルーチ
14742 ンはそのクラスの初期割付けプロセッサにより実行される.

14743

14744 **【μ ITRON4.0仕様との関係】**

14745

14746 μ ITRON4.0仕様に定義されていない静的APIである.

14747 -----

14748 ref_cfg コンフィギュレーション情報の参照 [T]

14749

14750 **【C言語API】**

14751 ER ercd = ref_cfg(T_RCFG *pk_rcfg)

14752

14753 ☆未完成

14754

14755 【TOPPERS/ASPカーネルにおける規定】

14756

14757 ASPカーネルでは、ref_cfgをサポートしない。

14758

14759 【TOPPERS/FMPカーネルにおける規定】

14760

14761 FMPカーネルでは、ref_cfgをサポートしない。

14762

14763 【TOPPERS/HRP2カーネルにおける規定】

14764

14765 HRP2カーネルでは、ref_cfgをサポートしない。

14766

14767 【TOPPERS/SSPカーネルにおける規定】

14768

14769 SSPカーネルでは、ref_cfgをサポートしない。

14770

14771 ref_ver バージョン情報の参照 [T]

14772

14773 【C言語API】

14774 ER ercd = ref_ver(T_RVER *pk_rver)

14775

14776 ☆未完成

14777

14778 【TOPPERS/ASPカーネルにおける規定】

14779

14780 ASPカーネルでは、ref_verをサポートしない。

14781

14782 【TOPPERS/FMPカーネルにおける規定】

14783

14784 FMPカーネルでは、ref_verをサポートしない。

14785

14786 【TOPPERS/HRP2カーネルにおける規定】

14787

14788 HRP2カーネルでは、ref_verをサポートしない。

14789

14790 【TOPPERS/SSPカーネルにおける規定】

14791

14792 SSPカーネルでは、ref_verをサポートしない。

14793

14794

14795

14796 第5章 リファレンス

14797

14798 5.1 サービスコール一覧

14799

14800 (1) タスク管理機能


```

14801
14802     ER_ID tskid = acre_tsk(const T_CTSK *pk_ctsk)           [TD]
14803     ER ercd = sac_tsk(ID tskid, const ACVCT *p_acvct)      [TPD]
14804     ER ercd = del_tsk(ID tskid)                             [TD]
14805     ER ercd = act_tsk(ID tskid)                             [T]
14806     ER ercd = iact_tsk(ID tskid)                           [I]
14807     ER ercd = mact_tsk(ID tskid, ID prcid)                 [TM]
14808     ER ercd = imact_tsk(ID tskid, ID prcid)                [IM]
14809     ER_UINT actcnt = can_act(ID tskid)                     [T]
14810     ER ercd = mig_tsk(ID tskid, ID prcid)                  [TM]
14811     ER ercd = ext_tsk()                                     [T]
14812     ER ercd = ter_tsk(ID tskid)                             [T]
14813     ER ercd = chg_pri(ID tskid, PRI tskpri)                [T]
14814     ER ercd = get_pri(ID tskid, PRI *p_tskpri)             [T]
14815     ER ercd = get_inf(intptr_t *p_exinf)                   [T]
14816     ER ercd = ref_tsk(ID tskid, T_RTsk *pk_rtsk)           [T]
14817
14818 (2) タスク付属同期機能
14819
14820     ER ercd = slp_tsk()                                     [T]
14821     ER ercd = tslp_tsk(TMO tmout)                          [T]
14822     ER ercd = wup_tsk(ID tskid)                             [T]
14823     ER ercd = iwup_tsk(ID tskid)                           [I]
14824     ER_UINT wupcnt = can_wup(ID tskid)                     [T]
14825     ER ercd = rel_wai(ID tskid)                             [T]
14826     ER ercd = irel_wai(ID tskid)                           [I]
14827     ER ercd = sus_tsk(ID tskid)                             [T]
14828     ER ercd = rsm_tsk(ID tskid)                             [T]
14829     ER ercd = dis_wai(ID tskid)                             [TP]
14830     ER ercd = idis_wai(ID tskid)                           [IP]
14831     ER ercd = ena_wai(ID tskid)                             [TP]
14832     ER ercd = iena_wai(ID tskid)                           [IP]
14833     ER ercd = dly_tsk(RELTIM dlytim)                       [T]
14834
14835 (3) タスク例外処理機能
14836
14837     ER ercd = def_tex(ID tskid, const T_DTEX *pk_dtex)     [TD]
14838     ER ercd = ras_tex(ID tskid, TEXPTN rasptn)             [T]
14839     ER ercd = iras_tex(ID tskid, TEXPTN rasptn)            [I]
14840     ER ercd = dis_tex()                                     [T]
14841     ER ercd = ena_tex()                                     [T]
14842     bool_t state = sns_tex()                                [TI]
14843     ER ercd = ref_tex(ID tskid, T_RTEx *pk_rtex)           [T]
14844
14845 (4) 同期・通信機能
14846
14847 セマフォ
14848
14849     ER_ID semid = acre_sem(const T_CSEM *pk_csem)           [TD]
14850     ER ercd = sac_sem(ID semid, const ACVCT *p_acvct)       [TPD]

```

```

14851      ER ercd = del_sem(ID semid)                                [TD]
14852      ER ercd = sig_sem(ID semid)                                [T]
14853      ER ercd = isig_sem(ID semid)                               [I]
14854      ER ercd = wai_sem(ID semid)                                [T]
14855      ER ercd = pol_sem(ID semid)                                [T]
14856      ER ercd = twai_sem(ID semid, TMO tmout)                    [T]
14857      ER ercd = ini_sem(ID semid)                                [T]
14858      ER ercd = ref_sem(ID semid, T_RSEM *pk_rsem)                [T]
14859
14860      イベントフラグ
14861
14862      ER_ID flgid = acre_flg(const T_CFLG *pk_cflg)               [TD]
14863      ER ercd = sac_flg(ID flgid, const ACVCT *p_acvct)           [TPD]
14864      ER ercd = del_flg(ID flgid)                                 [TD]
14865      ER ercd = set_flg(ID flgid, FLGPTN setptn)                  [T]
14866      ER ercd = iset_flg(ID flgid, FLGPTN setptn)                [I]
14867      ER ercd = clr_flg(ID flgid, FLGPTN clrptn)                  [T]
14868      ER ercd = wai_flg(ID flgid, FLGPTN waiptn,
14869                                MODE wfmode, FLGPTN *p_flgptn)    [T]
14870      ER ercd = pol_flg(ID flgid, FLGPTN waiptn,
14871                                MODE wfmode, FLGPTN *p_flgptn)    [T]
14872      ER ercd = twai_flg(ID flgid, FLGPTN waiptn,
14873                                MODE wfmode, FLGPTN *p_flgptn, TMO tmout) [T]
14874      ER ercd = ini_flg(ID flgid)                                 [T]
14875      ER ercd = ref_flg(ID flgid, T_RFLG *pk_rflg)                [T]
14876
14877      データキュー
14878
14879      ER_ID dtqid = acre_dtq(const T_CDTQ *pk_cdtq)               [TD]
14880      ER ercd = sac_dtq(ID dtqid, const ACVCT *p_acvct)           [TPD]
14881      ER ercd = del_dtq(ID dtqid)                                 [TD]
14882      ER ercd = snd_dtq(ID dtqid, intptr_t data)                  [T]
14883      ER ercd = psnd_dtq(ID dtqid, intptr_t data)                 [T]
14884      ER ercd = ipsnd_dtq(ID dtqid, intptr_t data)                [I]
14885      ER ercd = tsnd_dtq(ID dtqid, intptr_t data, TMO tmout)      [T]
14886      ER ercd = fsnd_dtq(ID dtqid, intptr_t data)                 [T]
14887      ER ercd = ifsnd_dtq(ID dtqid, intptr_t data)                [I]
14888      ER ercd = rcv_dtq(ID dtqid, intptr_t *p_data)               [T]
14889      ER ercd = prcv_dtq(ID dtqid, intptr_t *p_data)              [T]
14890      ER ercd = trcv_dtq(ID dtqid, intptr_t *p_data, TMO tmout)    [T]
14891      ER ercd = ini_dtq(ID dtqid)                                 [T]
14892      ER ercd = ref_dtq(ID dtqid, T_RDTQ *pk_rdtq)                [T]
14893
14894      優先度データキュー
14895
14896      ER_ID pdqid = acre_pdq(const T_CPDQ *pk_cpdq)               [TD]
14897      ER ercd = sac_pdq(ID pdqid, const ACVCT *p_acvct)           [TPD]
14898      ER ercd = del_pdq(ID pdqid)                                 [TD]
14899      ER ercd = snd_pdq(ID pdqid, intptr_t data, PRI datapri)      [T]
14900      ER ercd = psnd_pdq(ID pdqid, intptr_t data, PRI datapri)    [T]

```

```

14901      ER ercd = ipsnd_pdq(ID pdqid, intptr_t data, PRI datapri)      [I]
14902      ER ercd = tsnd_pdq(ID pdqid, intptr_t data,                    [T]
14903                          PRI datapri, TMO tmout)
14904      ER ercd = rcv_pdq(ID pdqid, intptr_t *p_data, PRI *p_datapri)  [T]
14905      ER ercd = prev_pdq(ID pdqid, intptr_t *p_data, PRI *p_datapri) [T]
14906      ER ercd = trcv_pdq(ID pdqid, intptr_t *p_data,                [T]
14907                          PRI *p_datapri, TMO tmout)
14908      ER ercd = ini_pdq(ID pdqid)                                    [T]
14909      ER ercd = ref_pdq(ID pdqid, T_RPDQ *pk_rpdq)                  [T]
14910
14911      メールボックス
14912
14913      ER_ID mbxid = acre_mbx(const T_CMBX *pk_cmbx)                  [TDp]
14914      ER ercd = del_mbx(ID mbxid)                                    [TDp]
14915      ER ercd = snd_mbx(ID mbxid, T_MSG *pk_msg)                    [Tp]
14916      ER ercd = rcv_mbx(ID mbxid, T_MSG **ppk_msg)                  [Tp]
14917      ER ercd = prev_mbx(ID mbxid, T_MSG **ppk_msg)                  [Tp]
14918      ER ercd = trcv_mbx(ID mbxid, T_MSG **ppk_msg, TMO tmout)      [Tp]
14919      ER ercd = ini_mbx(ID mbxid)                                    [Tp]
14920      ER ercd = ref_mbx(ID mbxid, T_RMBX *pk_rmbx)                  [Tp]
14921
14922      ミューテックス
14923
14924      ER_ID mtxid = acre_mtx(const T_CMTX *pk_cmtx)                  [TD]
14925      ER ercd = sac_mtx(ID mtxid, const ACVCT *p_acvct)              [TPD]
14926      ER ercd = del_mtx(ID mtxid)                                    [TD]
14927      ER ercd = loc_mtx(ID mtxid)                                    [T]
14928      ER ercd = ploc_mtx(ID mtxid)                                   [T]
14929      ER ercd = tloc_mtx(ID mtxid, TMO tmout)                        [T]
14930      ER ercd = unl_mtx(ID mtxid)                                    [T]
14931      ER ercd = ini_mtx(ID mtxid)                                    [T]
14932      ER ercd = ref_mtx(ID mtxid, T_RMTX *pk_rmtx)                  [T]
14933
14934      メッセージバッファ
14935
14936      ☆未完成
14937
14938      スピンロック
14939
14940      ER_ID spnid = acre_spn(const T_CSPN *pk_cspn)                  [TMD]
14941      ER ercd = sac_spn(ID spnid, const ACVCT *p_acvct)              [TPMD]
14942      ER ercd = del_spn(ID spnid)                                    [TMD]
14943      ER ercd = loc_spn(ID spnid)                                    [TM]
14944      ER ercd = iloc_spn(ID spnid)                                   [IM]
14945      ER ercd = try_spn(ID spnid)                                    [TM]
14946      ER ercd = itry_spn(ID spnid)                                   [IM]
14947      ER ercd = unl_spn(ID spnid)                                    [TM]
14948      ER ercd = iunl_spn(ID spnid)                                   [IM]
14949      ER ercd = ref_spn(ID spnid, T_RSPN *pk_rspn)                  [TM]
14950

```

14951 (5) メモリプール管理機能

14952

14953 固定長メモリプール

14954

```

14955     ER_ID mpfid = acre_mpf(const T_CMPF *pk_cmpf)           [TD]
14956     ER ercd = sac_mpf(ID mpfid, const ACVCT *p_acvct)       [TPD]
14957     ER ercd = del_mpf(ID mpfid)                             [TD]
14958     ER ercd = get_mpf(ID mpfid, void **p_blk)               [T]
14959     ER ercd = pget_mpf(ID mpfid, void **p_blk)              [T]
14960     ER ercd = tget_mpf(ID mpfid, void **p_blk, TMO tmout)   [T]
14961     ER ercd = rel_mpf(ID mpfid, void *blk)                  [T]
14962     ER ercd = ini_mpf(ID mpfid)                             [T]
14963     ER ercd = ref_mpf(ID mpfid, T_RMPF *pk_rmpf)            [T]

```

14964

14965 (6) 時間管理機能

14966

14967 システム時刻管理

14968

```

14969     ER ercd = get_tim(SYSTIM *p_system)                     [T]
14970     ER ercd = get_utm(SYSUTM *p_sysutm)                     [TI]

```

14971

14972 周期ハンドラ

14973

```

14974     ER_ID cycid = acre_cyc(const T_CCYC *pk_ccyc)           [TD]
14975     ER ercd = sac_cyc(ID cycid, const ACVCT *p_acvct)       [TPD]
14976     ER ercd = del_cyc(ID cycid)                             [TD]
14977     ER ercd = sta_cyc(ID cycid)                             [T]
14978     ER ercd = msta_cyc(ID cycid, ID prcid)                  [TM]
14979     ER ercd = stp_cyc(ID cycid)                             [T]
14980     ER ercd = ref_cyc(ID cycid, T_RCYC *pk_rcyc)            [T]

```

14981

14982 アラームハンドラ

14983

```

14984     ER_ID almid = acre_alm(const T_CALM *pk_calm)           [TD]
14985     ER ercd = sac_alm(ID almid, const ACVCT *p_acvct)       [TPD]
14986     ER ercd = del_alm(ID almid)                             [TD]
14987     ER ercd = sta_alm(ID almid, RELTIM almtim)              [T]
14988     ER ercd = ista_alm(ID almid, RELTIM almtim)             [I]
14989     ER ercd = msta_alm(ID almid, RELTIM almtim, ID prcid)   [TM]
14990     ER ercd = imsta_alm(ID almid, RELTIM almtim, ID prcid)  [IM]
14991     ER ercd = stp_alm(ID almid)                             [T]
14992     ER ercd = istp_alm(ID almid)                            [I]
14993     ER ercd = ref_alm(ID almid, T_RALM *pk_ralm)            [T]

```

14994

14995 オーバランハンドラ

14996

```

14997     ER ercd = def_ovr(const T_DOVR *pk_dovr)               [TD]
14998     ER ercd = sta_ovr(ID tskid, OVRTIM ovrtime)            [T]
14999     ER ercd = ista_ovr(ID tskid, OVRTIM ovrtime)           [I]
15000     ER ercd = stp_ovr(ID tskid)                            [T]

```

301

15051	ER ercd = chg_ipm(PRI intpri)	[T]
15052	ER ercd = get_ipm(PRI *p_intpri)	[T]
15053		
15054	(10) CPU例外管理機能	
15055		
15056	ER ercd = def_exc(EXCNO excno, const T_DEXC *pk_dexc)	[TD]
15057	bool_t stat = xsns_dpn(void *p_excinf)	[TI]
15058	bool_t stat = xsns_xpn(void *p_excinf)	[TI]
15059		
15060	(11) 拡張サービスコール管理機能	
15061		
15062	ER ercd = def_svc(FN fncd, const T_DSVC *pk_dsvc)	[TPD]
15063	ER_UINT ercd = cal_svc(FN fcnd, intptr_t par1, intptr_t par2,	[TIP]
15064	intptr_t par3, intptr_t par4, intptr_t par5)	
15065		
15066	(12) システム構成管理機能	
15067		
15068	ER ercd = ref_cfg(T_RCFG *pk_rcfg)	[T]
15069	ER ercd = ref_ver(T_RVER *pk_rver)	[T]
15070		
15071	5.2 静的API一覧	
15072		
15073	(1) タスク管理機能	
15074		
15075	*保護機能対応でないカーネルの場合	
15076	CRE_TSK(ID tskid, { ATR tskatr, intptr_t exinf, TASK task,	[S]
15077	PRI itskpri, SIZE stksz, STK_T *stk })	
15078		
15079	*保護機能対応カーネルの場合	
15080	CRE_TSK(ID tskid, { ATR tskatr, intptr_t exinf, TASK task,	[SP]
15081	PRI itskpri, SIZE stksz, STK_T *stk,	
15082	SIZE sstksz, STK_T *sstk })	
15083	※ sstkszおよびsstkの記述は省略することができる。	
15084		
15085	AID_TSK(uint_t notsk)	[SD]
15086	SAC_TSK(ID tskid, { ACPTN acptn1, ACPTN acptn2,	[SP]
15087	ACPTN acptn3, ACPTN acptn4 })	
15088	DEF_EPR(ID tskid, { PRI exepri })	[S]
15089		
15090	(2) タスク付属同期機能	
15091		
15092	なし	
15093		
15094	(3) タスク例外処理機能	
15095		
15096	DEF_TEX(ID tskid, { ATR texatr, TEXRTN texrtn })	[S]
15097		
15098	(4) 同期・通信機能	
15099		
15100	セマフォ	

15101		
15102	CRE_SEM(ID semid, { ATR sematr, uint_t isemcnt, uint_t maxsem })	[S]
15103	AID_SEM(uint_t noseem)	[SD]
15104	SAC_SEM(ID semid, { ACPTN acptn1, ACPTN acptn2,	[SP]
15105	ACPTN acptn3, ACPTN acptn4 })	
15106		
15107	イベントフラグ	
15108		
15109	CRE_FLG(ID flgid, { ATR flgatr, FLGPTN iflgptn })	[S]
15110	AID_FLG(uint_t noflg)	[SD]
15111	SAC_FLG(ID flgid, { ACPTN acptn1, ACPTN acptn2,	[SP]
15112	ACPTN acptn3, ACPTN acptn4 })	
15113		
15114	データキュー	
15115		
15116	CRE_DTQ(ID dtqid, { ATR dtqatr, uint_t dtqcnt, void *dtqmb })	[S]
15117	AID_DTQ(uint_t nodtq)	[SD]
15118	SAC_DTQ(ID dtqid, { ACPTN acptn1, ACPTN acptn2,	[SP]
15119	ACPTN acptn3, ACPTN acptn4 })	
15120		
15121	優先度データキュー	
15122		
15123	CRE_PDQ(ID pdqid, { ATR pdqatr, uint_t pdqcnt,	[S]
15124	PRI maxdpri, void *pdqmb })	
15125	AID_PDQ(uint_t nopdq)	[SD]
15126	SAC_PDQ(ID pdqid, { ACPTN acptn1, ACPTN acptn2,	[SP]
15127	ACPTN acptn3, ACPTN acptn4 })	
15128		
15129	メールボックス	
15130		
15131	CRE_MBX(ID mbxid, { ATR mbxatr, PRI maxmpri, void *mprihd })	[Sp]
15132	AID_MBX(uint_t nombx)	[SpD]
15133		
15134	ミューテックス	
15135		
15136	CRE_MTX(ID mtxid, { ATR mtxatr, PRI ceilpri })	[S]
15137	AID_MTX(uint_t nomtx)	[SD]
15138	SAC_MTX(ID mtxid, { ACPTN acptn1, ACPTN acptn2,	[SP]
15139	ACPTN acptn3, ACPTN acptn4 })	
15140		
15141	メッセージバッファ	
15142		
15143	☆未完成	
15144		
15145	スピンロック	
15146		
15147	CRE_SPN(ID spnid, { ATR spnatr })	[SM]
15148	AID_SPN(uint_t nospn)	[SMD]
15149	SAC_SPN(ID spnid, { ACPTN acptn1, ACPTN acptn2,	[SPM]
15150	ACPTN acptn3, ACPTN acptn4 })	

```

15151
15152 (5) メモリプール管理機能
15153
15154 固定長メモリプール
15155
15156     CRE_MPF(ID mpfid, { ATR mpfatr, uint_t blkcnt, uint_t blksz,      [S]
15157                      MPF_T *mpf, void *mpfmb })
15158     AID_MPF(uint_t nompf)                                           [SD]
15159     SAC_MPF(ID mpfid, { ACPTN acptn1, ACPTN acptn2,                  [SP]
15160                      ACPTN acptn3, ACPTN acptn4 })
15161
15162 (6) 時間管理機能
15163
15164 周期ハンドラ
15165
15166     CRE_CYC(ID cycid, { ATR cycatr, intptr_t exinf, CYCHDR cychdr,   [S]
15167                      RELTIM cycetim, RELTIM cycphs })
15168     AID_CYC(uint_t nocyc)                                           [SD]
15169     SAC_CYC(ID cycid, { ACPTN acptn1, ACPTN acptn2,                  [SP]
15170                      ACPTN acptn3, ACPTN acptn4 })
15171
15172 アラームハンドラ
15173
15174     CRE_ALM(ID almid, { ATR almatr, intptr_t exinf, ALMHDR almhdr }) [S]
15175     AID_ALM(uint_t noalm)                                           [SD]
15176     SAC_ALM(ID almid, { ACPTN acptn1, ACPTN acptn2,                  [SP]
15177                      ACPTN acptn3, ACPTN acptn4 })
15178
15179 オーバランハンドラ
15180
15181     DEF_OVR({ ATR ovratr, OVRHDR ovrhdr })                          [S]
15182
15183 (7) システム状態管理機能
15184
15185     SAC_SYS({ ACPTN acptn1, ACPTN acptn2,                            [SP]
15186             ACPTN acptn3, ACPTN acptn4 })
15187
15188 (8) メモリオブジェクト管理機能
15189
15190     ATT_REG("メモリリージョン名",                                  [SP]
15191             { ATR regatr, void *base, SIZE size })
15192     ATT_SEC("セクション名", { ATR mematr, "メモリリージョン名" }) [SP]
15193     ATA_SEC("セクション名", { ATR mematr, "メモリリージョン名" }, [SP]
15194             { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
15195     LNK_SEC("セクション名", { "メモリリージョン名" })             [SP]
15196     ATT_MOD("オブジェクトモジュール名")                           [SP]
15197     ATA_MOD("オブジェクトモジュール名",                            [SP]
15198             { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
15199     ATT_MEM({ ATR mematr, void *base, SIZE size })                 [SP]
15200     ATA_MEM({ ATR mematr, void *base, SIZE size },                 [SP]

```



```

15201         { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
15202     ATT_PMA({ ATR mematr, void *base, SIZE size, void *paddr })      [SP]
15203     ATA_PMA({ ATR mematr, void *base, SIZE size, void *paddr },      [SP]
15204         { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
15205
15206     (9) 割込み管理機能
15207
15208     CFG_INT(INTNO intno, { ATR intatr, PRI intpri })                  [S]
15209     CRE_ISR(ID isrid, { ATR isratr, intptr_t exinf,                    [S]
15210         INTNO intno, ISR isr, PRI isrpri })
15211     ATT_ISR({ ATR isratr, intptr_t exinf,                              [S]
15212         INTNO intno, ISR isr, PRI isrpri })
15213     AID_ISR(uint_t noisr)                                             [SD]
15214     SAC_ISR(ID isrid, { ACPTN acptn1, ACPTN acptn2,                  [SP]
15215         ACPTN acptn3, ACPTN acptn4 })
15216     DEF_INH(INHNO inhno, { ATR inhatr, INTHDR inthdr })              [S]
15217
15218     (10) CPU例外管理機能
15219
15220     DEF_EXC(EXCNO excno, { ATR excatr, EXCHDR exchr })                [S]
15221
15222     (11) 拡張サービスコール管理機能
15223
15224     DEF_SVC(FN fncd, { ATR svcatr, EXTSVC svcrtm, SIZE stksz })       [SP]
15225
15226     (12) システム構成管理機能
15227
15228     DEF_ICS({ SIZE istksz, STK_T *istk })                             [S]
15229     DEF_STK({ SIZE stksz, STK_T *stk })                               [S]
15230     ATT_INI({ ATR iniatr, intptr_t exinf, INIRTN inirtn })            [S]
15231     ATT_TER({ ATR teratr, intptr_t exinf, TERRTN terrtn })            [S]
15232
15233     5.3 データ型
15234
15235     5.3.1 TOPPERS共通データ型
15236
15237     int8_t      符号付き8ビット整数 (オプション, C99準拠)
15238     uint8_t     符号無し8ビット整数 (オプション, C99準拠)
15239     int16_t     符号付き16ビット整数 (C99準拠)
15240     uint16_t    符号無し16ビット整数 (C99準拠)
15241     int32_t     符号付き32ビット整数 (C99準拠)
15242     uint32_t    符号無し32ビット整数 (C99準拠)
15243     int64_t     符号付き64ビット整数 (オプション, C99準拠)
15244     uint64_t    符号無し64ビット整数 (オプション, C99準拠)
15245     int128_t    符号付き128ビット整数 (オプション, C99準拠)
15246     uint128_t   符号無し128ビット整数 (オプション, C99準拠)
15247
15248     int_least8_t 8ビット以上の符号付き整数 (C99準拠)
15249     uint_least8_t int_least8_t型と同じサイズの符号無し整数 (C99準拠)
15250

```

15251	float32_t	IEEE754準拠の32ビット単精度浮動小数点数 (オプション)
15252	double64_t	IEEE754準拠の64ビット倍精度浮動小数点数 (オプション)
15253		
15254	bool_t	真偽値 (trueまたはfalse)
15255	int_t	16ビット以上の符号付き整数
15256	uint_t	int_t型と同じサイズの符号無し整数
15257	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
15258	ulong_t	long_t型と同じサイズの符号無し整数
15259		
15260	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)
15261	uintptr_t	intptr_t型と同じサイズの符号無し整数 (C99準拠)
15262		
15263	FN	機能コード (符号付き整数, int_tに定義)
15264	ER	エラーコード (符号付き整数, int_tに定義)
15265	ID	オブジェクトのID番号 (符号付き整数, int_tに定義)
15266	ATR	オブジェクト属性 (符号無し整数, uint_tに定義)
15267	STAT	オブジェクトの状態 (符号無し整数, uint_tに定義)
15268	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)
15269	PRI	優先度 (符号付き整数, int_tに定義)
15270	SIZE	メモリ領域のサイズ (符号無し整数, ポインタを格納できる
15271		サイズの符号無し整数型に定義)
15272		
15273	TMO	タイムアウト指定 (符号付き整数, 単位はミリ秒, int_tに定義)
15274	RELTIM	相対時間 (符号無し整数, 単位はミリ秒, uint_tに定義)
15275	SYSTIM	システム時刻 (符号無し整数, 単位はミリ秒, ulong_tに定義)
15276	SYSUTM	性能評価用システム時刻 (符号無し整数, 単位はマイクロ秒,
15277		ulong_tに定義)
15278		
15279	FP	プログラムの起動番地 (型の定まらない関数ポインタ)
15280		
15281	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)
15282	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,
15283		負のID番号は格納できない)
15284	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
15285		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
15286		より1ビット短い)
15287		
15288	MB_T	オブジェクト管理領域を確保するためのデータ型
15289		
15290	ACPTN	アクセス許可パターン (符号無し32ビット整数, uint32_tに
15291		定義)
15292		
15293	typedef struct acvct {	/* アクセス許可ベクタ */
15294	ACPTN acptn1;	/* 通常操作1のアクセス許可パターン */
15295	ACPTN acptn2;	/* 通常操作2のアクセス許可パターン */
15296	ACPTN acptn3;	/* 管理操作のアクセス許可パターン */
15297	ACPTN acptn4;	/* 参照操作のアクセス許可パターン */
15298	} ACVCT;	
15299		
15300	5.3.2 カーネルの使用データ型	

```

15301
15302     TEXTN      タスク例外要因のビットパターン (符号無し整数, uint_tに定義)
15303     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)
15304     OVRTIM     プロセッサ時間 (符号無し整数, 単位はマイクロ秒, ulong_tに定義)
15305     INTNO      割込み番号 (符号無し整数, uint_tに定義)
15306     INHNO      割込みハンドラ番号 (符号無し整数, uint_tに定義)
15307     EXCNO      CPU例外ハンドラ番号 (符号無し整数, uint_tに定義)
15308
15309     TASK       タスクのメインルーチン (関数ポインタ)
15310     TEXRTN     タスク例外処理ルーチン (関数ポインタ)
15311     CYCHDR     周期ハンドラ (関数ポインタ)
15312     ALMHDR     アラームハンドラ (関数ポインタ)
15313     OVRHDR     オーバランハンドラ (関数ポインタ)
15314     ISR        割込みサービ斯拉ーチン (関数ポインタ)
15315     INTHDR     割込みハンドラ (関数ポインタ)
15316     EXCHDR     CPU例外ハンドラ (関数ポインタ)
15317     EXTSVC     拡張サービスコール (関数ポインタ)
15318     INIRTN     初期化ルーチン (関数ポインタ)
15319     TERRTN     終了処理ルーチン (関数ポインタ)
15320
15321     STK_T       スタック領域を確保するためのデータ型
15322     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
15323

```

```

15324     typedef struct t_msg {          /* メールボックスのメッセージヘッダ */
15325         struct t_msg    *pk_next;
15326     } T_MSG;
15327
15328     typedef struct t_msg_pri {      /* 優先度付きメッセージヘッダ */
15329         T_MSG            msgque;    /* メッセージヘッダ */
15330         PRI              msgpri;    /* メッセージ優先度 */
15331     } T_MSG_PRI;
15332

```

15333 5.3.3 カーネルの使用するパケット形式

15334

15335 (1) タスク管理機能

15336

15337 タスクの生成情報のパケット形式

15338

```

15339     typedef struct t_ctsk {
15340         ATR          tskatr;        /* タスク属性 */
15341         intptr_t     exinf;         /* タスクの拡張情報 */
15342         TASK         task;          /* タスクのメインルーチンの先頭番地 */
15343         PRI          itskpri;       /* タスクの起動時優先度 */
15344         SIZE         stksz;         /* タスクのスタック領域のサイズ */
15345         STK_T *      stk;           /* タスクのスタック領域の先頭番地 */
15346         /* 以下は、保護機能対応カーネルの場合 */
15347         SIZE         sstksz;        /* タスクのシステムスタック領域のサイズ */
15348         STK_T *      sstk;          /* タスクのシステムスタック領域の先頭番地 */
15349     } T_CTSK;
15350

```

15351 タスクの現在状態のパケット形式

15352

```
15353     typedef struct t_rtsk {
15354         STAT      tskstat;    /* タスク状態 */
15355         PRI       tskpri;     /* タスクの現在優先度 */
15356         PRI       tskbpri;    /* タスクのベース優先度 */
15357         STAT      tskwait;    /* 待ち要因 */
15358         ID        wobjid;     /* 待ち対象のオブジェクトのID */
15359         TMO       lefttmo;    /* タイムアウトするまでの時間 */
15360         uint_t    actcnt;     /* 起動要求キューイング数 */
15361         uint_t    wupcnt;     /* 起床要求キューイング数 */
15362         /* 以下は、保護機能対応カーネルの場合 */
15363         bool_t    texmsk;     /* タスク例外マスク状態か否か */
15364         bool_t    waifbd;     /* 待ち禁止状態か否か */
15365         uint_t    svclevel;   /* 拡張サービスコールのネストレベル */
15366         /* 以下は、マルチプロセッサ対応カーネルの場合 */
15367         ID        prcid;      /* 割付けプロセッサのID */
15368         ID        actprc      /* 次の起動時の割付けプロセッサのID */
15369     } T_RTSK;
```

15370

15371 (2) タスク付属同期機能

15372

15373 なし

15374

15375 (3) タスク例外処理機能

15376

15377 タスク例外処理ルーチンの定義情報のパケット形式

15378

```
15379     typedef struct t_dtex {
15380         ATR       texatr;     /* タスク例外処理ルーチン属性 */
15381         TEXRTN    texrtn;     /* タスク例外処理ルーチンの先頭番地 */
15382     } T_DTEX;
```

15383

15384 タスク例外処理の現在状態のパケット形式

15385

```
15386     typedef struct t_rtex {
15387         STAT      texstat;    /* タスク例外処理の状態 */
15388         TEXPTN    pndptn;     /* 保留例外要因 */
15389     } T_RTEX;
```

15390

15391 (4) 同期・通信機能

15392

15393 セマフォの生成情報のパケット形式

15394

```
15395     typedef struct t_csem {
15396         ATR       sematr;     /* セマフォ属性 */
15397         uint_t    isemcnt;    /* セマフォの初期資源数 */
15398         uint_t    maxsem;     /* セマフォの最大資源数 */
15399     } T_CSEM;
```

15400

15401 セマフォの現在状態のパケット形式

15402

```
15403     typedef struct t_rsem {
15404         ID          wtskid;      /* セマフォの待ち行列の先頭のタスクのID番号 */
15405         uint_t      semcnt;      /* セマフォの資源数 */
15406     } T_RSEM;
```

15407

15408 イベントフラグの生成情報のパケット形式

15409

```
15410     typedef struct t_cflg {
15411         ATR          flgatr;      /* イベントフラグ属性 */
15412         FLGPTN       iflgptn;     /* イベントフラグの初期ビットパターン */
15413     } T_CFLG;
```

15414

15415 イベントフラグの現在状態のパケット形式

15416

```
15417     typedef struct t_rflg {
15418         ID          wtskid;      /* イベントフラグの待ち行列の先頭のタ
15419                                スクのID番号 */
15420         FLGPTN       flgptn;     /* イベントフラグのビットパターン */
15421     } T_RFLG;
```

15422

15423 データキューの生成情報のパケット形式

15424

```
15425     typedef struct t_cdtq {
15426         ATR          dtqatr;      /* データキュー属性 */
15427         uint_t        dtqcnt;      /* データキュー管理領域に格納できるデータ数 */
15428         void *        dtqmb;      /* データキュー管理領域の先頭番地 */
15429     } T_CDTQ;
```

15430

15431 データキューの現在状態のパケット形式

15432

```
15433     typedef struct t_rdtq {
15434         ID          stskid;      /* データキューの送信待ち行列の先頭のタ
15435                                スクのID番号 */
15436         ID          rtskid;      /* データキューの受信待ち行列の先頭のタ
15437                                スクのID番号 */
15438         uint_t        sdtqcnt;    /* データキュー管理領域に格納されている
15439                                データの数 */
15440     } T_RDTQ;
```

15441

15442 優先度データキューの生成情報のパケット形式

15443

```
15444     typedef struct t_cpdq {
15445         ATR          pdqatr;      /* 優先度データキュー属性 */
15446         uint_t        pdqcnt;      /* 優先度データキュー管理領域に格納でき
15447                                るデータ数 */
15448         PRI          maxdpri;     /* 優先度データキューに送信できるデータ
15449                                優先度の最大値 */
15450         void *        pdqmb;      /* 優先度データキュー管理領域の先頭番地 */
15451     } T_CPDQ;
```

15451 } T_CPDQ;

15452

15453 優先度データキューの現在状態のパケット形式

15454

15455 typedef struct t_rpdq {

15456 ID stskid; /* 優先度データキューの送信待ち行列の先
15457 頭のタスクのID番号 */

15458 ID rtskid; /* 優先度データキューの受信待ち行列の先
15459 頭のタスクのID番号 */

15460 uint_t spdqcnt; /* 優先度データキュー管理領域に格納され
15461 ているデータの数 */

15462 } T_RPDQ;

15463

15464 メールボックスの生成情報のパケット形式

15465

15466 typedef struct t_cmbx {

15467 ATR mbxatr; /* メールボックス属性 */

15468 PRI maxmpri; /* 優先度メールボックスに送信できるメッ
15469 セージ優先度の最大値 */

15470 void * mprihd; /* 優先度別のメッセージキューヘッダ領域
15471 の先頭番地 */

15472 } T_CMBX;

15473

15474 メールボックスの現在状態のパケット形式

15475

15476 typedef struct t_rmbx {

15477 ID wtskid; /* メールボックスの待ち行列の先頭のタスク
15478 のID番号 */

15479 T_MSG *pk_msg; /* メッセージキューの先頭につながれたメッ
15480 セージの先頭番地 */

15481 } T_RMBX;

15482

15483 ミューテックスの生成情報のパケット形式

15484

15485 typedef struct t_cmtx {

15486 ATR mtxatr; /* ミューテックス属性 */

15487 PRI ceilpri; /* ミューテックスの上限優先度 */

15488 } T_CMTX;

15489

15490 ミューテックスの現在状態のパケット形式

15491

15492 typedef struct t_rmtx {

15493 ID htskid; /* ミューテックスをロックしているタス
15494 クのID番号 */

15495 ID wtskid; /* ミューテックスの待ち行列の先頭のタ
15496 スクのID番号 */

15497 } T_RMTX;

15498

15499 メッセージバッファの生成情報のパケット形式

15500

```

15501 ☆未完成
15502
15503 メッセージバッファの現在状態のパケット形式
15504
15505 ☆未完成
15506
15507 スピンロックの生成情報のパケット形式
15508
15509     typedef struct t_cspn {
15510         ATR          spnatr;      /* スピンロック属性 */
15511     } T_CSPN;
15512
15513 スピンロックの現在状態のパケット形式
15514
15515     typedef struct t_rspn {
15516         STAT          spnstat     /* スピンロックのロック状態 */
15517     } T_RSPN;
15518
15519 (5) メモリプール管理機能
15520
15521 固定長メモリアールの生成情報のパケット形式
15522
15523     typedef struct t_cmpf {
15524         ATR          mpfatr;      /* 固定長メモリアール属性 */
15525         uint_t       blkcnt;      /* 獲得できる固定長メモリアブロックの数 */
15526         uint_t       blksz;      /* 固定長メモリアブロックのサイズ */
15527         MPF_T *      mpf;        /* 固定長メモリアール領域の先頭番地 */
15528         void *       mpfmb;      /* 固定長メモリアール管理領域の先頭番地 */
15529     } T_CMPF;
15530
15531 固定長メモリアールの現在状態のパケット形式
15532
15533     typedef struct t_rmpf {
15534         ID           wtskid;      /* 固定長メモリアールの待ち行列の先頭の
15535                                タスクのID番号 */
15536         uint_t       fblkcnt;     /* 固定長メモリアール領域の空きメモリア領
15537                                域に割り付けることができる固定長メモ
15538                                リブロックの数 */
15539     } T_RMPF;
15540
15541 (6) 時間管理機能
15542
15543 周期ハンドラの生成情報のパケット形式
15544
15545     typedef struct t_ccyc {
15546         ATR          cycatr;      /* 周期ハンドラ属性 */
15547         intptr_t     exinf;       /* 周期ハンドラの拡張情報 */
15548         CYCHDR       cychdr;     /* 周期ハンドラ先頭番地 */
15549         RELTIM       cyctim;     /* 周期ハンドラの起動周期 */
15550         RELTIM       cycphs;     /* 周期ハンドラの起動位相 */

```

15551 } T_CCYC;

15552

15553 周期ハンドラの現在状態のパケット形式

15554

15555 typedef struct t_rcyc {

15556 STAT cycstat; /* 周期ハンドラの動作状態 */

15557 RELTIM lefttim; /* 次に周期ハンドラを起動する時刻までの
15558 相対時間 */

15559 /* 以下は、マルチプロセッサ対応カーネルの場合 */

15560 ID prcid; /* 割付けプロセッサのID */

15561 } T_RCYC;

15562

15563 アラームハンドラの生成情報のパケット形式

15564

15565 typedef struct t_calm {

15566 ATR almatr; /* アラームハンドラ属性 */

15567 intptr_t exinf; /* アラームハンドラの拡張情報 */

15568 ALMHDR almhdr; /* アラームハンドラの先頭番地 */

15569 } T_CALM;

15570

15571 アラームハンドラの現在状態のパケット形式

15572

15573 typedef struct t_ralm {

15574 STAT almstat; /* アラームハンドラの動作状態 */

15575 RELTIM lefttim; /* アラームハンドラを起動する時刻までの
15576 相対時間 */

15577 /* 以下は、マルチプロセッサ対応カーネルの場合 */

15578 ID prcid; /* 割付けプロセッサのID */

15579 } T_RALM;

15580

15581 オーバランハンドラの定義情報のパケット形式

15582

15583 typedef struct t_dovr {

15584 ATR ovratr; /* オーバランハンドラ属性 */

15585 OVRHDR ovrhdr; /* オーバランハンドラ先頭番地 */

15586 } T_DOVR;

15587

15588 オーバランハンドラの現在状態のパケット形式

15589

15590 typedef struct t_rovr {

15591 STAT ovrstat; /* オーバランハンドラの動作状態 */

15592 OVRTIM leftotm; /* 残りプロセッサ時間 */

15593 } T_ROVR;

15594

15595 (7) システム状態管理機能

15596

15597 システムの現在状態のパケット形式

15598

15599 ☆未完成

15600

15601 (8) メモリオブジェクト管理機能

15602

15603 メモリオブジェクトの登録情報のパケット形式

15604

```
15605     typedef struct t_amem {
15606         ATR          mematr      /* メモリオブジェクト属性 */
15607         void *       base        /* 登録するメモリ領域の先頭番地 */
15608         SIZE         size        /* 登録するメモリ領域のサイズ (バイト数) */
15609     } T_AMEM;
```

15610

15611 物理メモリ領域の登録情報のパケット形式

15612

```
15613     typedef struct t_apma {
15614         ATR          mematr      /* メモリオブジェクト属性 */
15615         void *       base        /* 登録するメモリ領域の先頭番地 */
15616         SIZE         size        /* 登録するメモリ領域のサイズ (バイト数) */
15617         void *       paddr       /* 登録するメモリ領域の物理アドレスの先頭
15618                                     番地 */
15619     } T_APMA;
```

15620

15621 メモリオブジェクトの現在状態のパケット形式

15622

15623 ☆未完成

15624

15625 (9) 割込み管理機能

15626

15627 割込み要求ラインの属性の設定情報のパケット形式

15628

```
15629     typedef struct t_cint {
15630         ATR          intatr;      /* 割込み要求ライン属性 */
15631         PRI          intpri;      /* 割込み優先度 */
15632     } T_CINT;
```

15633

15634 割込みサービスルーチンの生成情報のパケット形式

15635

```
15636     typedef struct t_cisr {
15637         ATR          isratr;      /* 割込みサービスルーチン属性 */
15638         intptr_t     exinf;       /* 割込みサービスルーチンの拡張情報 */
15639         INTNO        intno;       /* 割込みサービスルーチンを登録する割込
15640                                     み番号 */
15641         ISR          isr;         /* 割込みサービスルーチンの先頭番地 */
15642         PRI          isrpri;      /* 割込みサービスルーチン優先度 */
15643     } T_CISR;
```

15644

15645 割込みサービスルーチンの現在状態のパケット形式

15646

15647 ☆未完成

15648

15649 割込みハンドラの定義情報のパケット形式

15650

```

15651     typedef struct t_dinh {
15652         ATR          inhatr;    /* 割込みハンドラ属性 */
15653         INTHDR       inthdr;    /* 割込みハンドラの先頭番地 */
15654     } T_DINH;
15655
15656     割込み要求ラインの現在状態のパケット形式
15657
15658     ☆未完成
15659
15660     (10) CPU例外管理機能
15661
15662     CPU例外ハンドラの定義情報のパケット形式
15663
15664     typedef struct t_dexc {
15665         ATR          excatr;    /* CPU例外ハンドラ属性 */
15666         EXCHDR       exchdr;    /* CPU例外ハンドラの先頭番地 */
15667     } T_DEXC;
15668
15669     (11) 拡張サービスコール管理機能
15670
15671     拡張サービスコールの定義情報のパケット形式
15672
15673     typedef struct t_dsvc {
15674         ATR          svcatr     /* 拡張サービスコール属性 */
15675         EXTSVC       svertn     /* 拡張サービスコールの先頭番地 */
15676         SIZE         stksz      /* 拡張サービスコールで使用するスタック
15677                                サイズ */
15678     } T_DSVC;
15679
15680     (12) システム構成管理機能
15681
15682     コンフィギュレーション情報のパケット形式
15683
15684     ☆未完成
15685
15686     バージョン情報のパケット形式
15687
15688     ☆未完成
15689
15690     5.4 定数とマクロ
15691
15692     5.4.1 TOPPERS共通定数
15693
15694     (1) 一般定数
15695
15696     NULL                                無効ポインタ
15697
15698     true                                1            真
15699     false                               0            偽
15700

```

15701	E_OK	0	正常終了
15702			
15703	(2)	整数型に格納できる最大値と最小値	
15704			
15705	INT8_MAX		int8_tに格納できる最大値 (オプション, C99準拠)
15706	INT8_MIN		int8_tに格納できる最小値 (オプション, C99準拠)
15707	UINT8_MAX		uint8_tに格納できる最大値 (オプション, C99準拠)
15708	INT16_MAX		int16_tに格納できる最大値 (C99準拠)
15709	INT16_MIN		int16_tに格納できる最小値 (C99準拠)
15710	UINT16_MAX		uint16_tに格納できる最大値 (C99準拠)
15711	INT32_MAX		int32_tに格納できる最大値 (C99準拠)
15712	INT32_MIN		int32_tに格納できる最小値 (C99準拠)
15713	UINT32_MAX		uint32_tに格納できる最大値 (C99準拠)
15714	INT64_MAX		int64_tに格納できる最大値 (オプション, C99準拠)
15715	INT64_MIN		int64_tに格納できる最小値 (オプション, C99準拠)
15716	UINT64_MAX		uint64_tに格納できる最大値 (オプション, C99準拠)
15717	INT128_MAX		int128_tに格納できる最大値 (オプション, C99準拠)
15718	INT128_MIN		int128_tに格納できる最小値 (オプション, C99準拠)
15719	UINT128_MAX		uint128_tに格納できる最大値 (オプション, C99準拠)
15720			
15721	INT_LEAST8_MAX		int_least8_tに格納できる最大値 (C99準拠)
15722	INT_LEAST8_MIN		int_least8_tに格納できる最小値 (C99準拠)
15723	UINT_LEAST8_MAX		uint_least8_tに格納できる最大値 (C99準拠)
15724	INT_MAX		int_tに格納できる最大値 (C90準拠)
15725	INT_MIN		int_tに格納できる最小値 (C90準拠)
15726	UINT_MAX		uint_tに格納できる最大値 (C90準拠)
15727	LONG_MAX		long_tに格納できる最大値 (C90準拠)
15728	LONG_MIN		long_tに格納できる最小値 (C90準拠)
15729	ULONG_MAX		ulong_tに格納できる最大値 (C90準拠)
15730			
15731	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)
15732			
15733	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)
15734			
15735	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)
15736			
15737	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)
15738			
15739			
15740	(3)	整数型のビット数	
15741			
15742	CHAR_BIT		char型のビット数 (C90準拠)
15743			
15744	(4)	オブジェクト属性	
15745			
15746	TA_NULL	0U	オブジェクト属性を指定しない
15747			
15748	(5)	タイムアウト指定	
15749			
15750	TMO_POL	0	ポーリング

15751	TMO_FEVR	-1	永久待ち
15752	TMO_NBLK	-2	ノンブロッキング
15753			
15754	(6) アクセス許可パターン		
15755			
15756	TACP_KERNEL	0U	カーネルドメインのみにアクセスを許可
15757	TACP_SHARED	~0U	すべての保護ドメインにアクセスを許可
15758			
15759	5.4.2 TOPPERS共通マクロ		
15760			
15761	(1) 整数定数を作るマクロ		
15762			
15763	INT8_C(val)		int_least8_t型の定数を作るマクロ (C99準拠)
15764	UINT8_C(val)		uint_least8_t型の定数を作るマクロ (C99準拠)
15765	INT16_C(val)		int16_t型の定数を作るマクロ (C99準拠)
15766	UINT16_C(val)		uint16_t型の定数を作るマクロ (C99準拠)
15767	INT32_C(val)		int32_t型の定数を作るマクロ (C99準拠)
15768	UINT32_C(val)		uint32_t型の定数を作るマクロ (C99準拠)
15769	INT64_C(val)		int64_t型の定数を作るマクロ (オプション, C99準拠)
15770	UINT64_C(val)		uint64_t型の定数を作るマクロ (オプション, C99準拠)
15771	INT128_C(val)		int128_t型の定数を作るマクロ (オプション, C99準拠)
15772	UINT128_C(val)		uint128_t型の定数を作るマクロ (オプション, C99準拠)
15773			
15774	UINT_C(val)		uint_t型の定数を作るマクロ
15775	ULONG_C(val)		ulong_t型の定数を作るマクロ
15776			
15777	(2) 型に関する情報を取り出すためのマクロ		
15778			
15779	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)
15780			
15781			
15782	alignof(type)		型typeのアラインメント単位を返すマクロ
15783			
15784	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
15785			
15786			
15787	(3) assertマクロ		
15788			
15789	assert(exp)		expが成立しているかを検査するマクロ (C90準拠)
15790			
15791	(4) コンパイラの拡張機能のためのマクロ		
15792			
15793	inline		インライン関数
15794	Inline		ファイルローカルなインライン関数
15795	asm		インラインアセンブラ
15796	Asm		インラインアセンブラ (最適化抑止)
15797	throw()		例外を発生しない関数
15798	NoReturn		リターンしない関数
15799			
15800	(5) エラーコード生成・分解マクロ		

15801			
15802	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
15803			
15804			
15805	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
15806			
15807	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
15808			
15809			
15810	(6) アクセス許可パターン生成マクロ		
15811			
15812	TACP(domid)		domidで指定される保護ドメインに属する処理単位のみにアクセスを許可するアクセス許可パターン
15813			
15814			
15815	5.4.3 カーネル共通定数		
15816			
15817	(1) オブジェクト属性		
15818			
15819	TA_TPRI	0x01U	タスクの待ち行列をタスクの優先度順に
15820			
15821	(2) 保護ドメインID		
15822			
15823	TDOM_SELF	0	自タスクの属する保護ドメイン
15824	TDOM_KERNEL	-1	カーネルドメイン
15825	TDOM_NONE	-2	無所属（保護ドメインに属さない）
15826			
15827	(3) その他のカーネル共通定数		
15828			
15829	TCLS_SELF	0	自タスクの属するクラス
15830			
15831	TPRC_NONE	0	割付けプロセッサの指定がない
15832	TPRC_INI	0	初期割付けプロセッサ
15833			
15834	TSK_SELF	0	自タスク指定
15835	TSK_NONE	0	該当するタスクがない
15836			
15837	TPRI_SELF	0	自タスクのベース優先度の指定
15838	TPRI_INI	0	タスクの起動時優先度の指定
15839			
15840	TIPM_ENAALL	0	割込み優先度マスク全解除
15841			
15842	5.4.4 カーネル共通マクロ		
15843			
15844	(1) オブジェクト属性を作るマクロ		
15845			
15846	TA_DOM(domid)		domidで指定される保護ドメインに属する
15847	TA_CLS(clsid)		clsidで指定されるクラスに属する
15848			
15849	(2) サービスコールの呼出し方法を指定するマクロ		
15850			

15851 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
15852 て呼び出すための名称
15853

15854 5.4.5 カーネルの機能毎の定数

15855 (1) タスク管理機能

15858	TA_ACT	0x02U	タスクの生成時にタスクを起動する
15859	TA_RSTR	0x04U	生成するタスクを制約タスクとする
15860	TA_FPU		FPUレジスタをコンテキストに含める
15861			
15862	TTS_RUN	0x01U	実行状態
15863	TTS_RDY	0x02U	実行可能状態
15864	TTS_WAI	0x04U	待ち状態
15865	TTS_SUS	0x08U	強制待ち状態
15866	TTS_WAS	0x0cU	二重待ち状態
15867	TTS_DMT	0x10U	休止状態
15868			
15869	TTW_SLP	0x0001U	起床待ち
15870	TTW_DLY	0x0002U	時間経過待ち
15871	TTW_SEM	0x0004U	セマフォの資源獲得待ち
15872	TTW_FLG	0x0008U	イベントフラグ待ち
15873	TTW_SDTQ	0x0010U	データキューへの送信待ち
15874	TTW_RDTQ	0x0020U	データキューからの受信待ち
15875	TTW_SPDQ	0x0100U	優先度データキューへの送信待ち
15876	TTW_RPDQ	0x0200U	優先度データキューからの受信待ち
15877	TTW_MBX	0x0040U	メールボックスからの受信待ち
15878	TTW_MTX	0x0080U	ミューテックスのロック待ち状態
15879	TTW_MPF	0x2000U	固定長メモリブロックの獲得待ち

15880
15881 TA_FPUの値は、ターゲット定義とする。
15882

15883 (3) タスク例外処理機能

15885	TTEX_ENA	0x01U	タスク例外処理許可状態
15886	TTEX_DIS	0x02U	タスク例外処理禁止状態

15887 (4) 同期・通信機能

15888 イベントフラグ

15892	TA_WMUL	0x02U	複数のタスクが待つのを許す
15893	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
15894			
15895	TWF_ORW	0x01U	イベントフラグのOR待ちモード
15896	TWF_ANDW	0x02U	イベントフラグのAND待ちモード

15897 メールボックス

15900	TA_MPRI	0x02U	メッセージキューをメッセージの優先度順にする
-------	---------	-------	------------------------

15901			
15902	スピンロック		
15903			
15904	TSPN_UNL	0x01U	取得されていない状態
15905	TSPN_LOC	0x02U	取得されている状態
15906			
15907	(6) 時間管理機能		
15908			
15909	周期ハンドラ		
15910			
15911	TA_STA	0x02U	周期ハンドラの生成時に周期ハンドラを動作開始する
15912	TA_PHS	0x04U	周期ハンドラを生成した時刻を基準時刻とする
15913			
15914	TCYC_STP	0x01U	周期ハンドラが動作していない状態
15915	TCYC_STA	0x02U	周期ハンドラが動作している状態
15916			
15917	アラームハンドラ		
15918			
15919	TALM_STP	0x01U	アラームハンドラが動作していない状態
15920	TALM_STA	0x02U	アラームハンドラが動作している状態
15921			
15922	オーバランハンドラ		
15923			
15924	TOVR_STP	0x01U	オーバランハンドラが動作していない状態
15925	TOVR_STA	0x02U	オーバランハンドラが動作している状態
15926			
15927	(8) メモリオブジェクト管理機能		
15928			
15929	TA_NOWRITE	0x01U	書込みアクセス禁止
15930	TA_NOREAD	0x02U	読出しアクセス禁止
15931	TA_EXEC	0x04U	実行アクセス許可
15932	TA_MEMINI	0x08U	メモリの初期化を行う
15933	TA_MEMPRSV	0x10U	メモリの初期化を行わない
15934	TA_SDATA	0x20U	ショートデータ領域に配置
15935	TA_UNCACHE	0x40U	キャッシュ禁止
15936	TA_IODEV	0x80U	周辺デバイスの領域
15937	TA_WTHROUGH		ライトスルーキャッシュを用いる
15938			
15939	TA_STDRAM	0x02U	標準ROMリージョン
15940	TA_STDRAM	0x04U	標準RAMリージョン
15941			
15942	TPM_WRITE	0x01U	書込みアクセス権のチェック
15943	TPM_READ	0x02U	読出しアクセス権のチェック
15944	TPM_EXEC	0x04U	実行アクセス権のチェック
15945			
15946	TA_WTHROUGHの値は、ターゲット定義とする。		
15947			
15948	(9) 割込み管理機能		
15949			
15950	TA_ENAINT	0x01U	割込み要求禁止フラグをクリア

15951	TA_EDGE	0x02U	エッジトリガ
15952	TA_POSEDGE		ポジティブエッジトリガ
15953	TA_NEGEDGE		ネガティブエッジトリガ
15954	TA_BOTHEDGE		両エッジトリガ
15955	TA_LOWLEVEL		ローレベルトリガ
15956	TA_HIGHLEVEL		ハイレベルトリガ
15957			
15958	TA_NONKERNEL	0x02U	カーネル管理外の割込み
15959			
15960	TA_POSEDGE, TA_NEGEDGE, TA_BOTHEDGE, TA_LOWLEVEL, TA_HIGHLEVELの値は、		
15961	ターゲット定義とする.		
15962			
15963	(10) CPU例外管理機能		
15964			
15965	TA_DIRECT		CPU例外ハンドラを直接呼び出す
15966			
15967	TA_DIRECTの値は、ターゲット定義とする.		
15968			
15969	5.4.6 カーネルの機能毎のマクロ		
15970			
15971	(1) タスク管理機能		
15972			
15973	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
15974			STK_T型の配列の要素数
15975	ROUND_STK_T(sz)		要素数COUNT_STK_T(sz)のSTK_T型の配列のサイズ (sz
15976			を, STK_T型のサイズの倍数になるように大きい方に
15977			丸めた値)
15978			
15979	(4) 同期・通信機能		
15980			
15981	TSZ_DTQMB(dtqcnt)		dtqcntで指定した数のデータを格納できるデータ
15982			キュー管理領域のサイズ (バイト数)
15983	TCNT_DTQMB(dtqcnt)		dtqcntで指定した数のデータを格納できるデータ
15984			キュー管理領域を確保するために必要なMB_T型の配
15985			列の要素数
15986			
15987	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
15988			タキュー管理領域のサイズ (バイト数)
15989	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
15990			タキュー管理領域を確保するために必要なMB_T型の
15991			配列の要素数
15992			
15993	(5) メモリプール管理機能		
15994			
15995	COUNT_MPF_T(b1ksz)		固定長メモリブロックのサイズがb1kszの固定長メモ
15996			リプール領域を確保するために, 固定長メモリブロッ
15997			ク1つあたりに必要なMPF_T型の配列の要素数を求め
15998			るマクロ
15999	ROUND_MPF_T(b1ksz)		要素数COUNT_MPF_T(b1ksz)のMPF_T型の配列のサイズ
16000			(b1kszを, MPF_T型のサイズの倍数になるように大き

16001		い方に丸めた値)
16002		
16003	TSZ_MPFMB(blkent)	blkentで指定した数の固定長メモリブロックを管理
16004		することができる固定長メモリプール管理領域のサ
16005		イズ (バイト数)
16006	TCNT_MPFMB(blkent)	blkentで指定した数の固定長メモリブロックを管理
16007		することができる固定長メモリプール管理領域を確
16008		保するために必要なMB_T型の配列の要素数
16009		
16010	5.5 構成マクロ	
16011		
16012	5.5.1 TOPPERS共通構成マクロ	
16013		
16014	(1) 相対時間の範囲	
16015		
16016	TMAX_RELTIM	相対時間に指定できる最大値
16017		
16018	5.5.2 カーネル共通構成マクロ	
16019		
16020	(1) サポートする機能	
16021		
16022	TOPPERS_SUPPORT_PROTECT	保護機能対応のカーネル
16023	TOPPERS_SUPPORT_MULTI_PRC	マルチプロセッサ対応のカーネル
16024	TOPPERS_SUPPORT_DYNAMIC_CRE	動的生成対応のカーネル
16025		
16026	(2) 優先度の範囲	
16027		
16028	TMIN_TPRI	タスク優先度の最小値 (=1)
16029	TMAX_TPRI	タスク優先度の最大値
16030		
16031	(3) プロセッサの数	
16032		
16033	TNUM_PRCID	プロセッサの数
16034		
16035	(4) 特殊な役割を持ったプロセッサ	
16036		
16037	TOPPERS_MASTER_PRCID	マスタプロセッサのID番号
16038	TOPPERS_SYSTIM_PRCID	システム時刻管理プロセッサのID番号
16039		
16040	(5) タイマ方式	
16041		
16042	TOPPERS_SYSTIM_LOCAL	ローカルタイマ方式の場合にマクロ定義
16043	TOPPERS_SYSTIM_GLOBAL	グローバルタイマ方式の場合にマクロ定義
16044		
16045	(6) バージョン情報	
16046		
16047	TKERNEL_MAKER	カーネルのメーカーコード (=0x0118)
16048	TKERNEL_PRID	カーネルの識別番号
16049	TKERNEL_SPVER	カーネル仕様のバージョン番号
16050	TKERNEL_PRVER	カーネルのバージョン番号

16051		
16052	5.5.3 カーネルの機能毎の構成マクロ	
16053		
16054	(1) タスク管理機能	
16055		
16056	TMAX_ACTCNT	タスクの起動要求キューイング数の最大値
16057		
16058	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）
16059		
16060		
16061	(2) タスク付属同期機能	
16062		
16063	TMAX_WUPCNT	タスクの起床要求キューイング数の最大値
16064		
16065	(3) タスク例外処理機能	
16066		
16067	TBIT_TEXPTN	タスク例外要因のビット数（TEXPTNの有効ビット数）
16068		
16069	(4) 同期・通信機能	
16070		
16071	セマフォ	
16072		
16073	TMAX_MAXSEM	セマフォの最大資源数の最大値
16074		
16075	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）
16076		
16077		
16078	イベントフラグ	
16079		
16080	TBIT_FLGPTN	イベントフラグのビット数（FLGPTNの有効ビット数）
16081		
16082	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）
16083		
16084		
16085		
16086	データキュー	
16087		
16088	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）
16089		
16090		
16091		
16092	優先度データキュー	
16093		
16094	TMIN_DPRI	データ優先度の最小値（=1）
16095	TMAX_DPRI	データ優先度の最大値
16096		
16097	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）
16098		
16099		
16100		

16101	メールボックス	
16102		
16103	TMIN_MPRI	メッセージ優先度の最小値 (=1)
16104	TMAX_MPRI	メッセージ優先度の最大値
16105		
16106	TNUM_MBXID	登録できるメールボックスの数 (動的生成対応でないカーネルでは, 静的APIによって登録されたメールボックスの数に一致)
16107		
16108		
16109		
16110	ミューテックス	
16111		
16112	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)
16113		
16114		
16115		
16116	スピンロック	
16117		
16118	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)
16119		
16120		
16121		
16122	(5) メモリプール管理機能	
16123		
16124	固定長メモリプール	
16125		
16126	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)
16127		
16128		
16129		
16130	(6) 時間管理機能	
16131		
16132	システム時刻管理	
16133		
16134	TIC_NUME	タイムティックの周期 (単位はミリ秒) の分子
16135	TIC_DENO	タイムティックの周期 (単位はミリ秒) の分母
16136		
16137	TOPPERS_SUPPORT_GET_UTM	get_utmがサポートされている
16138		
16139	周期ハンドラ	
16140		
16141	TNUM_CYCID	登録できる周期ハンドラの数 (動的生成対応でないカーネルでは, 静的APIによって登録された周期ハンドラの数に一致)
16142		
16143		
16144		
16145	アラームハンドラ	
16146		
16147	TNUM_ALMID	登録できるアラームハンドラの数 (動的生成対応でないカーネルでは, 静的APIによって登録されたアラームハンドラの数に一致)
16148		
16149		
16150		

16151 オーバランハンドラ

16152

16153 TMAX_OVRTIM プロセッサ時間に指定できる最大値

16154

16155 TOPPERS_SUPPORT_OVRHDR オーバランハンドラ機能がサポートされて
16156 いる

16157

16158 (7) システム状態管理機能

16159

16160 なし

16161

16162 (8) メモリオブジェクト管理機能

16163

16164 TOPPERS_SUPPORT_ATT_MOD ATT_MOD/ATA_MODがサポートされている

16165 TOPPERS_SUPPORT_ATT_PMA ATT_PMA/ATA_PMA/att_pmaがサポートさ
16166 れている

16167

16168 (9) 割込み管理機能

16169

16170 TMIN_INTPRI 割込み優先度の最小値 (最高値)

16171 TMAX_INTPRI 割込み優先度の最大値 (最低値, =-1)

16172

16173 TMIN_ISRPRI 割込みサービスルーチン優先度の最小値 (=1)

16174 TMAX_ISRPRI 割込みサービスルーチン優先度の最大値

16175

16176 TOPPERS_SUPPORT_DIS_INT dis_intがサポートされている

16177 TOPPERS_SUPPORT_ENA_INT ena_intがサポートされている

16178

16179 (10) CPU例外管理機能

16180

16181 なし

16182

16183 (11) 拡張サービスコール管理機能

16184

16185 TNUM_FNCD 登録できる拡張サービスコールの数 (動的生成対応でな
16186 いカーネルでは、静的APIによって登録された拡張サービ
16187 スコールの数に一致)

16188

16189 (12) システム構成管理機能

16190

16191 なし

16192

16193 5.6 エラーコード一覧

16194

16195 (1) メインエラーコード

16196

16197 E_SYS -5 システムエラー

16198 E_NOSPT -9 未サポート機能

16199 E_RSFN -10 予約機能コード

16200 E_RSATR -11 予約属性

16201	E_PAR	-17	パラメータエラー
16202	E_ID	-18	不正ID番号
16203	E_CTX	-25	コンテキストエラー
16204	E_MACV	-26	メモリアクセス違反
16205	E_OACV	-27	オブジェクトアクセス違反
16206	E_ILUSE	-28	サービスコール不正使用
16207	E_NOMEM	-33	メモリ不足
16208	E_NOID	-34	ID番号不足
16209	E_NORES	-35	資源不足
16210	E_OBJ	-41	オブジェクト状態エラー
16211	E_NOEXS	-42	オブジェクト未登録
16212	E_QOVR	-43	キューイングオーバーフロー
16213	E_RLWAI	-49	待ち禁止状態または待ち状態の強制解除
16214	E_TMOUT	-50	ポーリング失敗またはタイムアウト
16215	E_DLT	-51	待ちオブジェクトの削除または再初期化
16216	E_CLS	-52	待ちオブジェクトの状態変化
16217	E_WBLK	-57	ノンブロッキング受付け
16218	E_BOVR	-58	バッファオーバーフロー

16219

16220 5.7 機能コード一覧

16221

16222

16223		-0	-1	-2	-3
16224					
16225	-0x01	予約	予約	予約	予約
16226	-0x05	act_tsk	iact_tsk	can_act	ext_tsk
16227	-0x09	ter_tsk	chg_pri	get_pri	get_inf
16228	-0x0d	slp_tsk	tslp_tsk	wup_tsk	iwup_tsk
16229	-0x11	can_wup	rel_wai	irel_wai	予約
16230	-0x15	dis_wai	idis_wai	ena_wai	iena_wai
16231	-0x19	sus_tsk	rsm_tsk	dly_tsk	予約
16232	-0x1d	ras_tex	iras_tex	dis_tex	ena_tex
16233	-0x21	sns_tex	ref_tex	予約	予約
16234	-0x25	sig_sem	isig_sem	wai_sem	pol_sem
16235	-0x29	twai_sem	予約	予約	予約
16236	-0x2d	set_flg	iset_flg	clr_flg	wai_flg
16237	-0x31	pol_flg	twai_flg	予約	予約
16238	-0x35	snd_dtq	psnd_dtq	ipsnd_dtq	tsnd_dtq
16239	-0x39	fsnd_dtq	ifsnd_dtq	rcv_dtq	prcv_dtq
16240	-0x3d	trcv_dtq	予約	予約	予約
16241	-0x41	snd_pdq	psnd_pdq	ipsnd_pdq	tsnd_pdq
16242	-0x45	rcv_pdq	prcv_pdq	trcv_pdq	予約
16243	-0x49	snd_mbx	rcv_mbx	prcv_mbx	trcv_mbx
16244	-0x4d	loc_mtx	ploc_mtx	tloc_mtx	unl_mtx
16245	-0x51	snd_mbf	psnd_mbf	tsnd_mbf	rcv_mbf
16246	-0x55	prcv_mbf	trcv_mbf	予約	予約
16247	-0x59	get_mpf	pget_mpf	tget_mpf	rel_mpf
16248	-0x5d	get_tim	get_utm	予約	ref_ovr
16249	-0x61	sta_cyc	stp_cyc	予約	予約
16250	-0x65	sta_alm	ista_alm	stp_alm	istp_alm

16251	-0x69	sta_ovr	ista_ovr	stp_ovr	istp_ovr
16252	-0x6d	sac_sys	ref_sys	rot_rdq	irotd_rdq
16253	-0x71	get_did	予約	get_tid	iget_tid
16254	-0x75	loc_cpu	iloc_cpu	unl_cpu	iunl_cpu
16255	-0x79	dis_dsp	ena_dsp	sns_ctx	sns_loc
16256	-0x7d	sns_dsp	sns_dpn	sns_ker	ext_ker
16257	-0x81	att_mem	det_mem	sac_mem	prb_mem
16258	-0x85	ref_mem	予約	att_pma	予約
16259	-0x89	cfg_int	dis_int	ena_int	ref_int
16260	-0x8d	chg_ipm	get_ipm	予約	予約
16261	-0x91	xsns_dpn	xsns_xpn	予約	予約
16262	-0x95	ref_cfg	ref_ver	予約	予約
16263	-0x99	予約	予約	予約	予約
16264	-0x9d	予約	予約	予約	予約
16265	-0xa1	予約	ini_sem	ini_flg	ini_dtq
16266	-0xa5	ini_pdq	ini_mbx	ini_mtx	ini_mbf
16267	-0xa9	ini_mpf	予約	予約	予約
16268	-0xad	予約	予約	予約	予約
16269	-0xb1	ref_tsk	ref_sem	ref_flg	ref_dtq
16270	-0xb5	ref_pdq	ref_mbx	ref_mtx	ref_mbf
16271	-0xb9	ref_mpf	ref_cyc	ref_alm	ref_isr
16272	-0xbd	ref_spn	予約	予約	予約
16273	-0xc1	acre_tsk	acre_sem	acre_flg	acre_dtq
16274	-0xc5	acre_pdq	acre_mbx	acre_mtx	acre_mbf
16275	-0xc9	acre_mpf	acre_cyc	acre_alm	acre_isr
16276	-0xcd	acre_spn	予約	予約	予約
16277	-0xd1	del_tsk	del_sem	del_flg	del_dtq
16278	-0xd5	del_pdq	del_mbx	del_mtx	del_mbf
16279	-0xd9	del_mpf	del_cyc	del_alm	del_isr
16280	-0xdd	del_spn	予約	予約	予約
16281	-0xe1	sac_tsk	sac_sem	sac_flg	sac_dtq
16282	-0xe5	sac_pdq	予約	sac_mtx	sac_mbf
16283	-0xe9	sac_mpf	sac_cyc	sac_alm	sac_isr
16284	-0xed	sac_spn	予約	予約	予約
16285	-0xf1	def_tex	def_ovr	def_inh	def_exc
16286	-0xf5	def_svc	予約	予約	予約
16287	-0xf9	予約	予約	予約	予約
16288	-0xfd	予約	予約	予約	予約
16289	-0x101	mact_tsk	imact_tsk	migt_tsk	予約
16290	-0x105	msta_cyc	予約	msta_alm	imsta_alm
16291	-0x109	mrot_rdq	imrot_rdq	get_pid	iget_pid
16292	-0x10d	予約	予約	予約	予約
16293	-0x111	loc_spn	iloc_spn	try_spn	itry_spn
16294	-0x115	unl_spn	iunl_spn	予約	予約
16295	-0x119	予約	予約	予約	予約
16296	-0x11d	予約	予約	予約	予約
16297	-----				
16298					
16299	【μ ITRON4.0仕様との関係】				
16300					

16301 サービスコールの機能コードを割り当てなおした.

16302

16303 5.8 カーネルオブジェクトに対するアクセスの種別

16304

16305

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 実行	det_mem sac_mem	ref_mem prb_mem
タスク	act_tsk mact_tsk can_act mig_tsk wup_tsk can_wup	ter_tsk chg_pri rel_wai sus_tsk rsm_tsk dis_wai ena_wai ras_tex sta_ovr stp_ovr	del_tsk sac_tsk def_tex	get_pri ref_tsk ref_tex ref_ovr
セマフォ	sig_sem	wai_sem pol_sem twai_sem	del_sem ini_sem sac_sem	ref_sem
イベントフラグ	set_flg clr_flg	wai_flg pol_flg twai_flg	del_flg ini_flg sac_flg	ref_flg
データキュー	snd_dtq psnd_dtq tsnd_dtq fsnd_dtq	rcv_dtq prcv_dtq trcv_dtq	del_dtq ini_dtq sac_dtq	ref_dtq
優先度データキュー	snd_pdq psnd_pdq tsnd_pdq	rcv_pdq prcv_pdq trcv_pdq	del_pdq ini_pdq sac_pdq	ref_pdq
ミューテックス	loc_mtx ploc_mtx tloc_mtx	-	del_mtx ini_mtx sac_mtx	ref_mtx
スピンロック	loc_spn try_spn unl_spn	-	del_spn sac_spn	ref_spn
固定長メモリプール	get_mpf pget_mpf tget_mpf	rel_mpf	del_mpf ini_mpf sac_mpf	ref_mpf

16350

16351	周期ハンドラ	sta_cyc	stp_cyc	del_cyc	ref_cyc
16352		msta_cyc		sac_cyc	
16353	-----				
16354	アラームハンドラ	sta_alm	stp_alm	del_alm	ref_alm
16355		msta_alm		sac_alm	
16356	-----				
16357	割込みサービスルーチン	-	-	del_isr	ref_isr
16358				sac_isr	
16359	-----				
16360	システム状態	rot_rdq	loc_cpu	acre_yyy	get_tim
16361		mrot_rdq	unl_cpu	att_mem	get_ipm
16362		dis_dsp	dis_int	att_pma	ref_sys
16363		ena_dsp	ena_int	cfg_int	ref_int
16364			chg_ipm	def_inh	ref_cfg
16365				def_exc	ref_ver
16366				def_svc	
16367				def_ovr	
16368	-----				
16369					
16370	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：				
16371					
16372	・ 自タスクへの操作 (ext_tsk, get_inf, slp_tsk, tslp_tsk, dly_tsk,				
16373	dis_tex, ena_tex)				
16374	・ タスク例外状態参照 (sns_tex)				
16375	・ 性能評価用システム時刻の参照 (get_utm)				
16376	・ システム状態参照 (get_tid, get_did, get_pid, sns_ctx, sns_loc,				
16377	sns_dsp, sns_dpn, sns_ker)				
16378	・ CPU例外発生時の状態参照 (xsns_dpn, xsns_xpn)				
16379	・ 拡張サービスコールの呼出し (cal_svc)				
16380					
16381	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：				
16382					
16383	・ システム状態のアクセス許可ベクタの設定 (sac_sys)				
16384	・ カーネルの終了 (ext_ker)				
16385	・ 非タスクコンテキスト専用のサービスコール				
16386					
16387	アクセス許可ベクタによるアクセス保護を行わないサービスコール：				
16388					
16389	・ ミューテックスのロック解除 (unl_mtx)				
16390					
16391	【補足説明】				
16392					
16393	xsns_dpnとxsns_xpnは、エラーコードを返さないために、すべての保護ドメイ				
16394	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				
16395	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				
16396	ルドメインのみから呼び出すことができる。				
16397					
16398	unl_mtxは、アクセス許可ベクタによるアクセス保護を行わないサービスコール				
16399	としているが、ミューテックスをロックしたタスク以外が呼び出すとE_ILUSEエ				
16400	ラーとなるため、実質的には対象ミューテックスの通常操作1としてアクセス保				

16401 護されているとみなすことができる（ミューテックスのロック中にアクセス許
16402 可ベクタを変更した場合の振舞いは異なる）。

16403

16404 【 μ ITRON4.0/PX仕様との関係】

16405

16406 get_priは、 μ ITRON4.0/PX仕様ではタスクに対する通常操作1としていたのを、
16407 タスクに対する参照操作に変更した。また、get_ipm（ μ ITRON4.0/PX仕様では
16408 get_ixx）をシステム状態に対する通常操作2から参照操作に、sac_sysをシステ
16409 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
16410 サービスコールに変更した。システム時刻に対するアクセス許可ベクタは廃止
16411 し、get_timはシステム状態に対する参照操作とした。

16412

16413 5.9 省略名の元になった英語

16414

16415 5.9.1 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

16416

16417 xxx 元になった英語

16418 -----

16419	act	activate
16420	aid	automatically assigned ID
16421	ata	attach with access control vector
16422	att	attach
16423	cal	call
16424	can	cancel
16425	cfg	configure
16426	chg	change
16427	clr	clear
16428	cre	create
16429	def	define
16430	del	delete
16431	det	detach
16432	dis	disable
16433	dly	delay
16434	ena	enable
16435	epr	execution priority
16436	ext	exit
16437	get	get
16438	ini	initialize
16439	lnk	link
16440	loc	lock
16441	mig	migrate
16442	pol	poll
16443	prb	probe
16444	ras	raise
16445	rcv	receive
16446	ref	reference
16447	rel	release
16448	rot	rotate
16449	rsm	resume
16450	sac	set access control vector

16451	set	set
16452	sig	signal
16453	slp	sleep
16454	snd	send
16455	sns	sense
16456	sta	start
16457	stp	stop
16458	sus	suspend
16459	ter	terminate
16460	try	try
16461	unl	unlock
16462	wai	wait
16463	wup	wake up

16464

16465 5.9.2 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

16466

16467	yyy	元になった英語
16468	-----	
16469	act	activation
16470	alm	alarm handler
16471	cfg	configuration
16472	cpu	CPU
16473	ctx	context
16474	cyc	cyclic handler
16475	did	domain ID
16476	dpn	dispatch pending
16477	dsp	dispatch
16478	dtq	data queue
16479	exc	exception
16480	flg	eventflag
16481	ics	interrupt context stack
16482	inf	information
16483	inh	interrupt handler
16484	ini	initilization
16485	int	interrupt
16486	ipm	interrupt priority mask
16487	isr	interrupt service routine
16488	ker	kernel
16489	loc	lock
16490	mbf	message buffer
16491	mbx	mailbox
16492	mpf	fixed-sized memory pool
16493	mem	memory
16494	mod	module
16495	mtx	mutex
16496	ovr	overflow handler
16497	pdq	priority data queue
16498	pid	processor ID
16499	pma	physical memory area
16500	pri	priority

16501	rdq	ready queue
16502	reg	region
16503	sec	section
16504	sem	semaphore
16505	spn	spin lock
16506	stk	stack
16507	sys	system
16508	svc	service call
16509	ter	termination
16510	tex	task exception
16511	tid	task ID
16512	tim	time
16513	tsk	task
16514	utm	time in micro second
16515	ver	version
16516	wai	wait
16517	wup	wake up
16518	xpn	exception pending

16519

16520 5.9.3 サービスコールの名称の中のzの元になった英語

16521

16522 z 元になった英語

16523 -----

16524	a	automatic ID assignment
16525	f	force
16526	i	interrupt
16527	m	multiprocessor
16528	p	poll
16529	t	timeout
16530	x	exception

16531

16532 5.10 バージョン履歴

16533

16534	2008年11月19日	Release 1.0.0	最初のリリース
16535	2009年5月8日	Release 1.1.0	FMPカーネルに関する記述が完成
16536	2010年5月10日	Release 1.2.0	
16537	2011年5月5日	Release 1.3.0	HRP2カーネルに関する記述が完成

16538

16539 以上

アプリケーションシステム

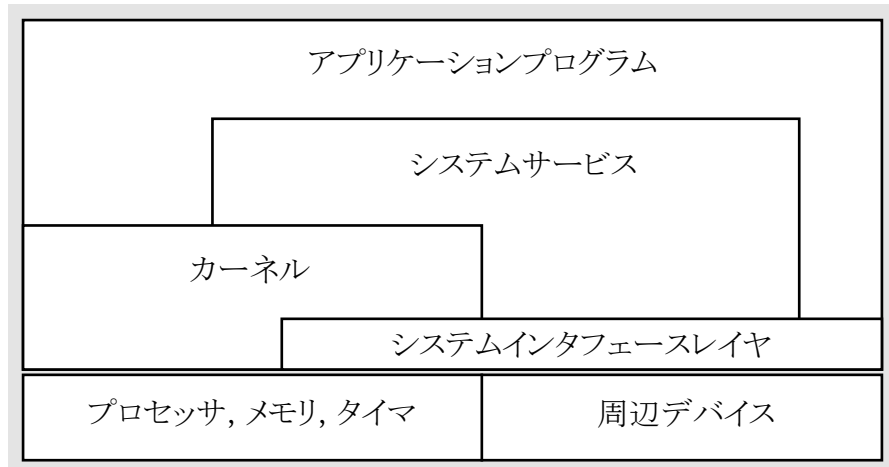


図2-1. 想定するソフトウェア構成

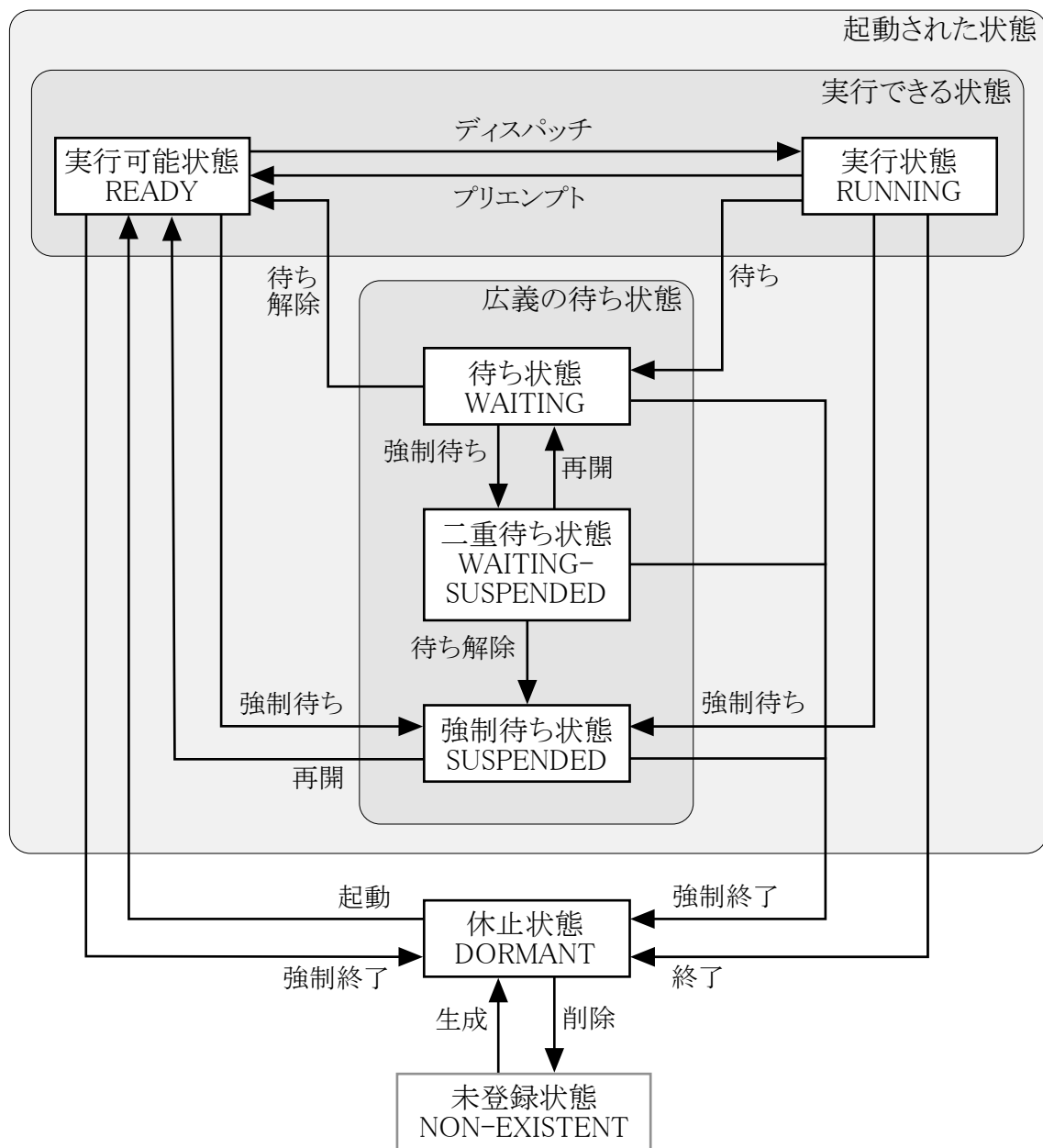


図2-2. タスクの状態遷移

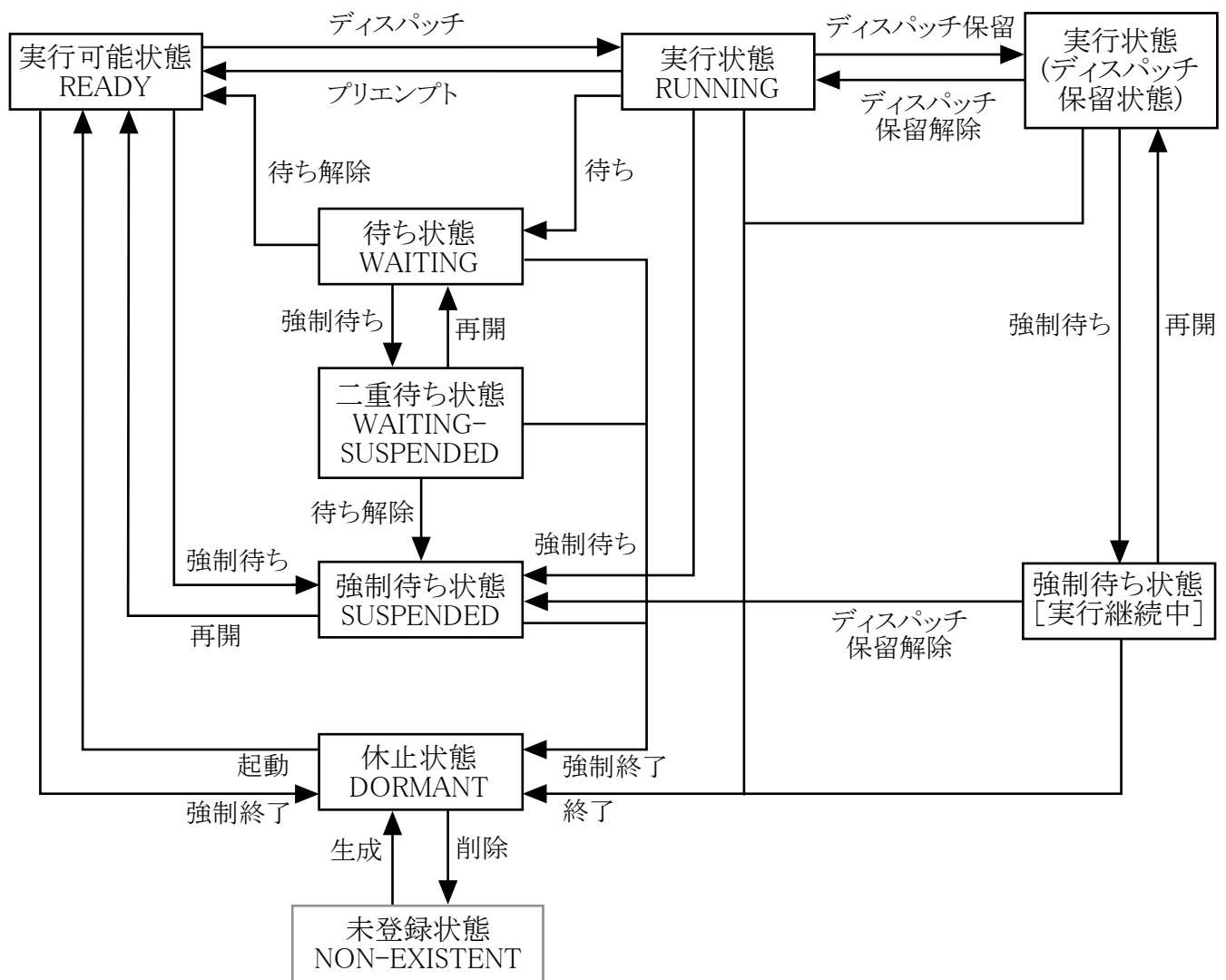


図2-3. 過渡的な状態も含めたタスクの状態遷移

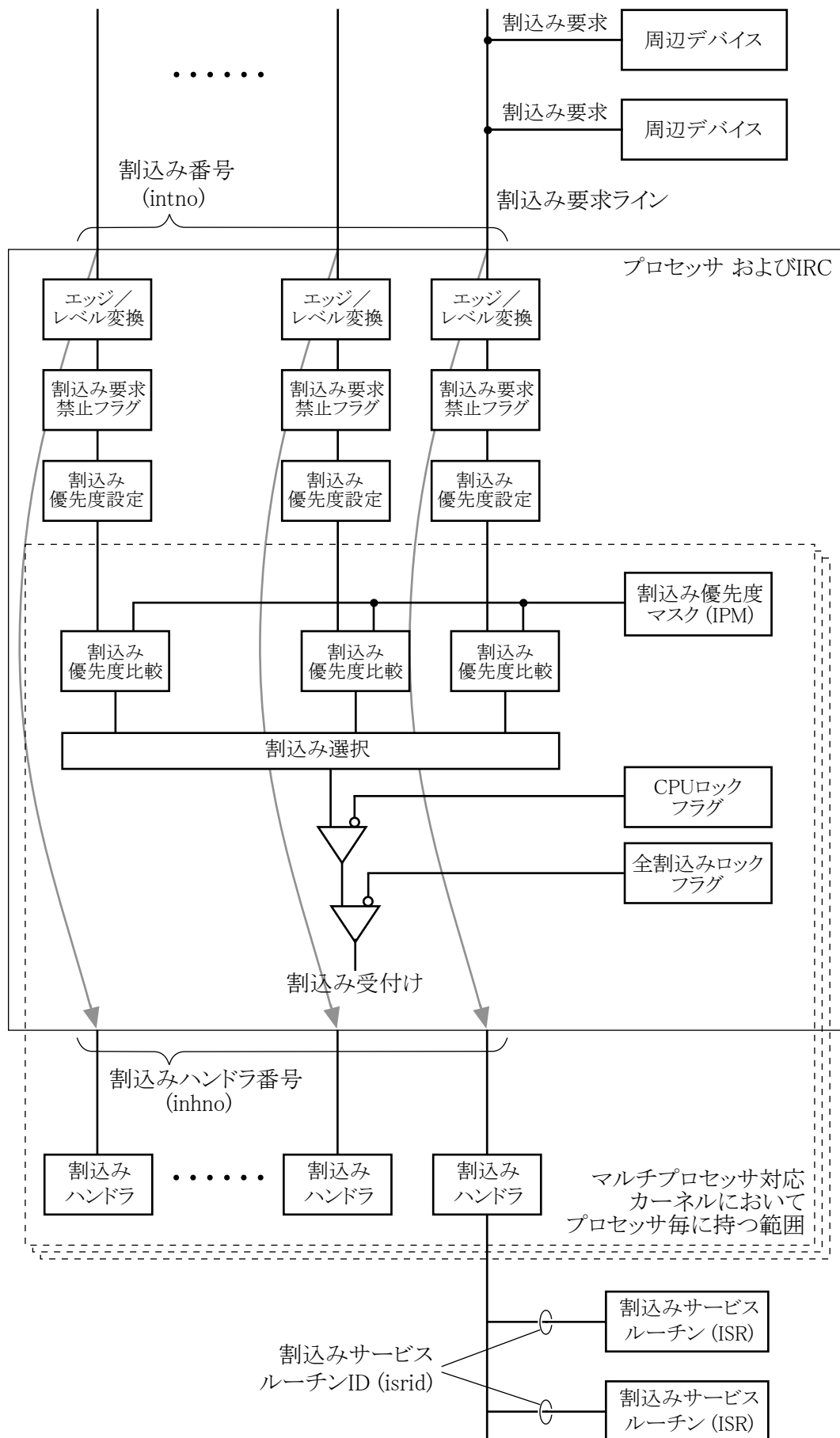


図2-4. TOPPERS標準割り込み処理モデルの概念図

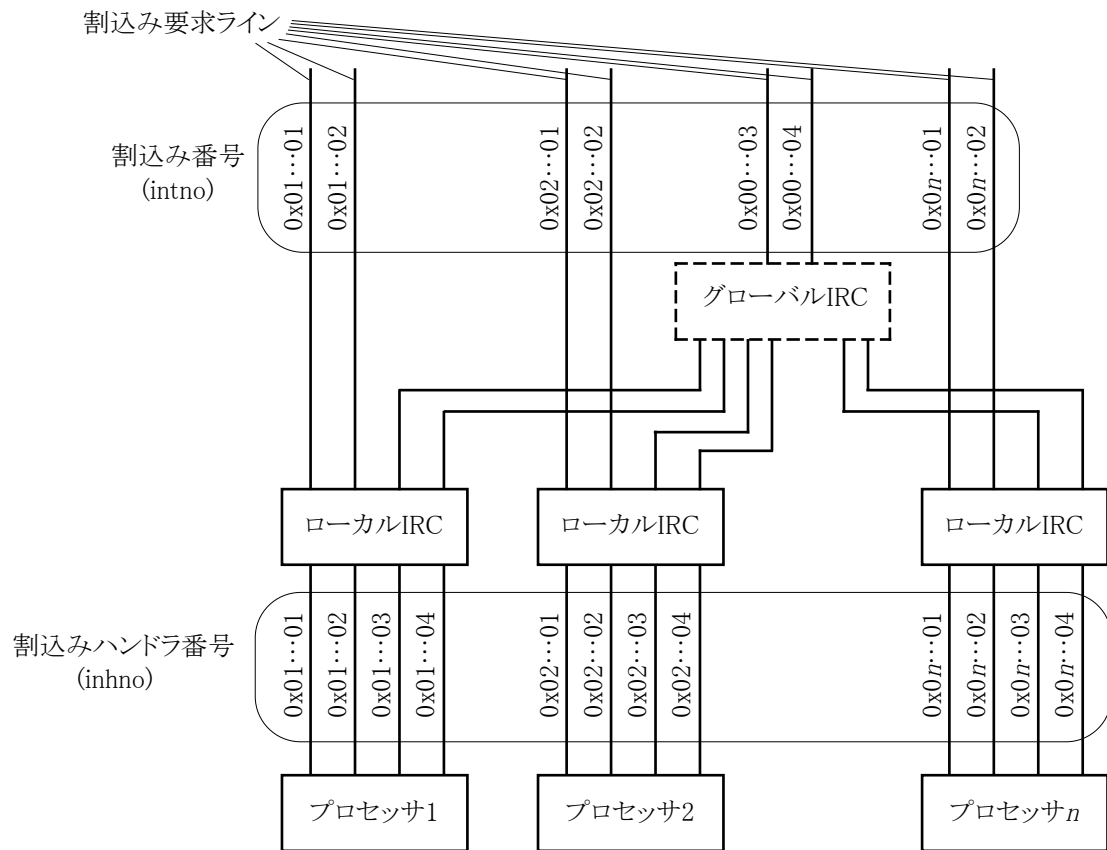


図2-5. マルチプロセッサ対応カーネルにおける割り込み番号と割り込みハンドラ番号

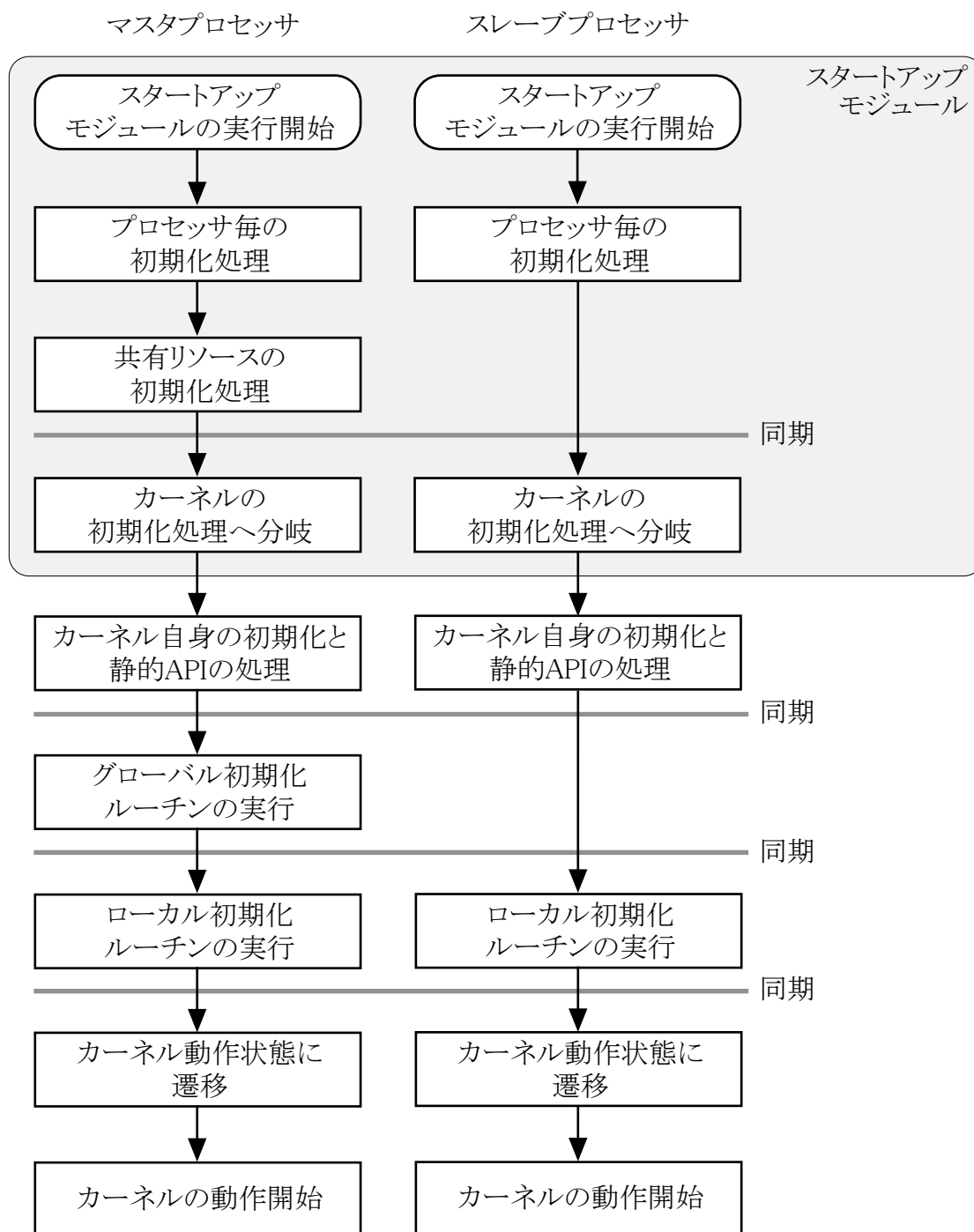


図2-6. マルチプロセッサ対応カーネルにおけるシステム初期化の流れ

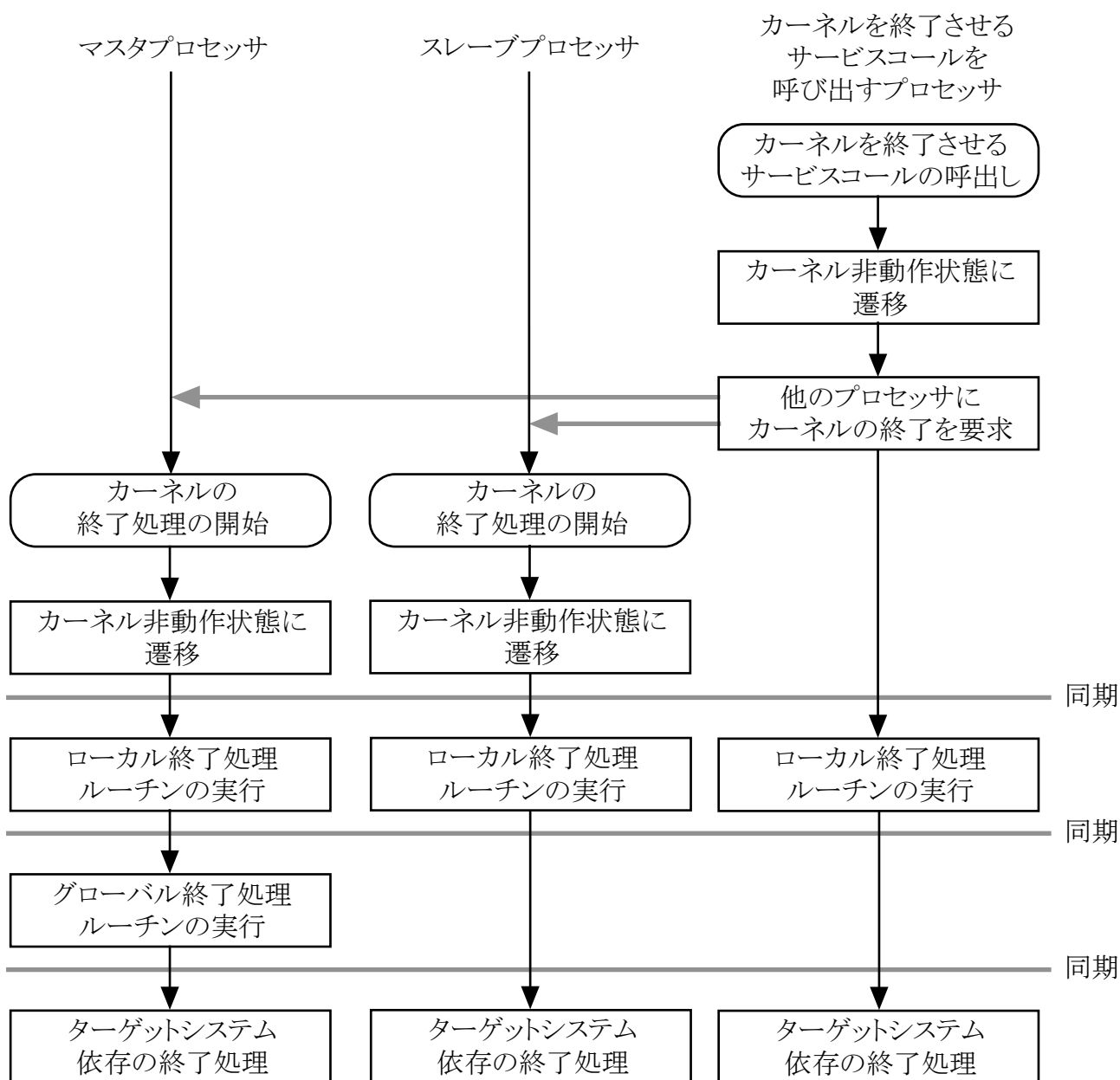


図2-7. マルチプロセッサ対応カーネルにおけるシステム終了処理の流れ

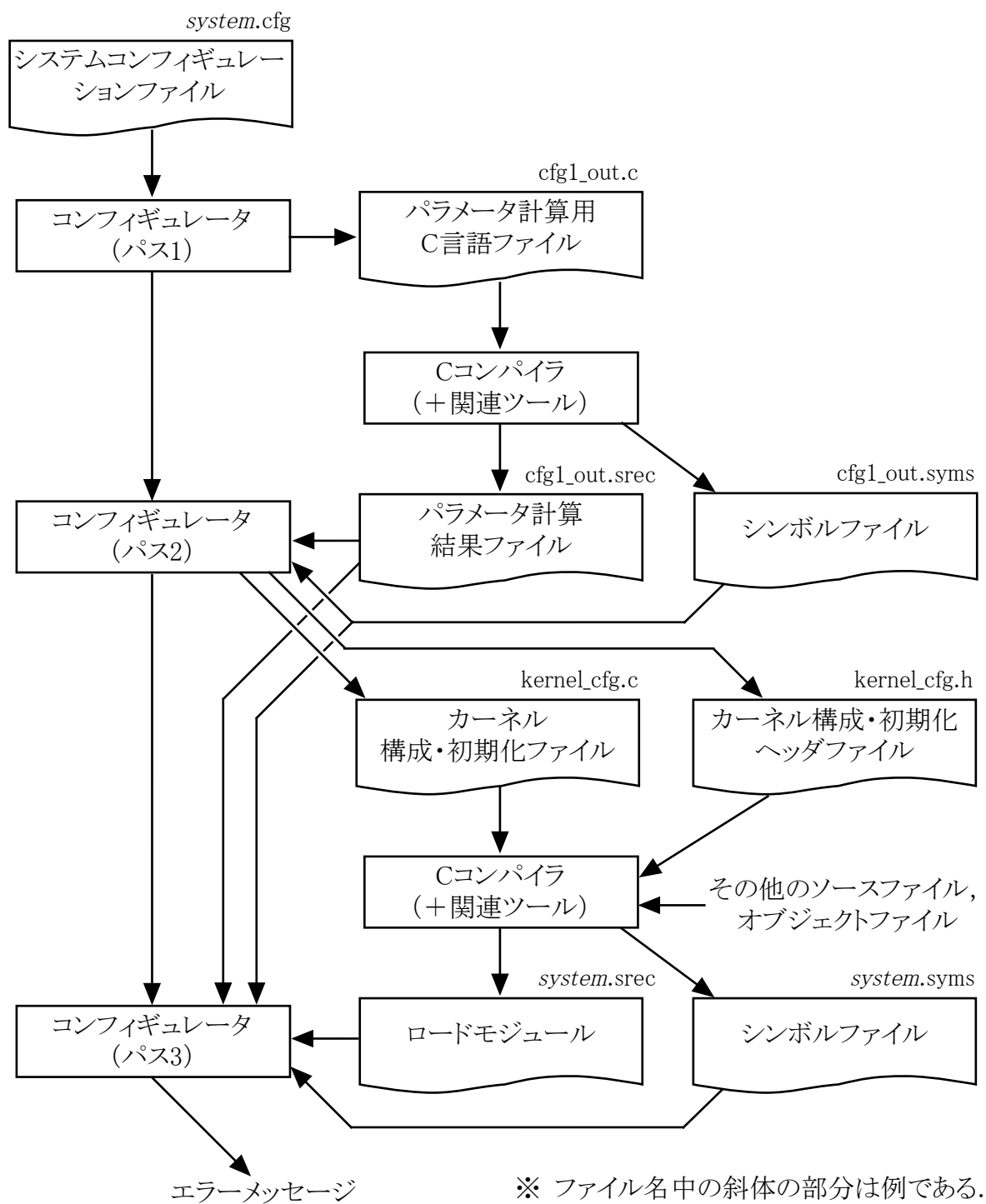
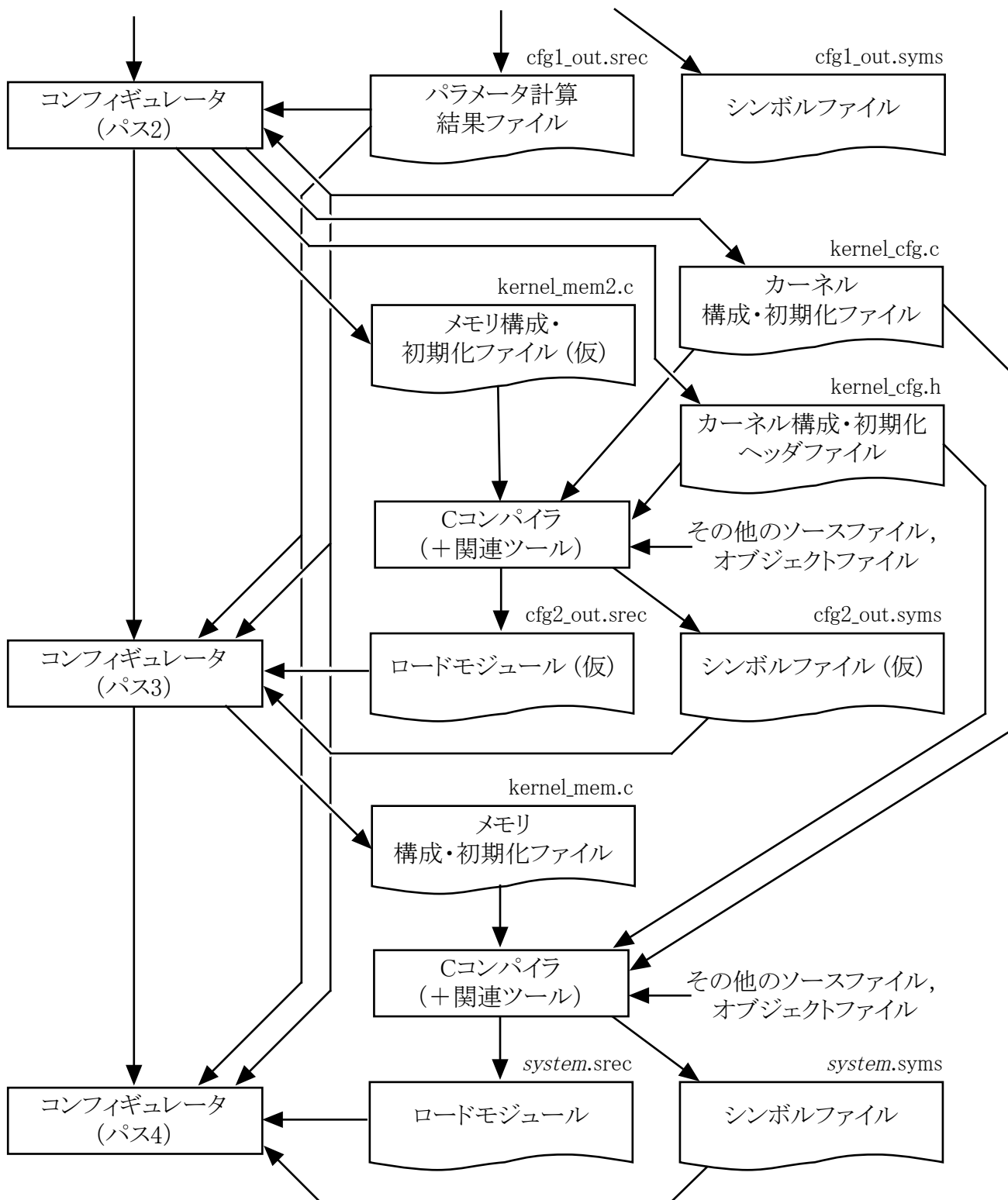


図2-8. コンフィギュレータの処理モデル



※ ファイル名中の斜体の部分は例である。

図2-9. 最終的なロードモジュールのパス3での生成