

TOPPERS新世代カーネル統合仕様書

バージョン: Release 1.7.1

最終更新: 2015年5月30日

このドキュメントは、TOPPERS新世代カーネルに属する一連のリアルタイムカーネルの仕様を、統合的に記述したものである。今後、この仕様に対して、大きい機能追加や仕様改変は行わず、これ以降は第3世代カーネル仕様として検討を行う計画である。ただし、仕様が未完成の部分（特に動的生成対応カーネルに関しては、仕様検討が不十分なところが多い）については、それを実装する時点で追加で決定していくこととする。

なお、本文中から参照している図は、ファイルの最後にまとめて掲載してある。

TOPPERS New Generation Kernel Specification

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

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

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

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

○目次

- ・ 目次
- ・ 仕様書で用いる記述項目と記号
- ・ タグの付与方法

51	第1章 TOPPERS新世代カーネルの概要
52	
53	1.1 TOPPERS新世代カーネル仕様の位置付け
54	1.2 TOPPERS新世代カーネル仕様の設計方針
55	1.3 TOPPERS/ASPカーネルの適用対象領域と仕様設計方針
56	1.4 TOPPERS/FMPカーネルの適用対象領域と仕様設計方針
57	1.5 TOPPERS/HRP2カーネルの適用対象領域と仕様設計方針
58	1.6 TOPPERS/SSPカーネルの適用対象領域と仕様設計方針
59	1.7 TOPPERS/ASP Safetyカーネルの適用対象領域と仕様設計方針
60	
61	第2章 主要な概念と共通定義
62	
63	2.1 仕様の位置付け
64	2.1.1 カーネルの機能セット
65	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
66	2.1.3 想定するソフトウェア構成
67	2.1.4 想定するハードウェア構成
68	2.1.5 想定するプログラミング言語
69	2.2 APIの構成要素とコンベンション
70	2.2.1 APIの構成要素
71	2.2.2 パラメータとリターンパラメータ
72	2.2.3 返値とエラーコード
73	2.2.4 機能コード
74	2.2.5 ヘッダファイル
75	2.3 主な概念
76	2.3.1 オブジェクトと処理単位
77	2.3.2 サービスコールとパラメータ
78	2.3.3 保護機能
79	2.3.4 マルチプロセッサ対応
80	2.3.5 その他
81	2.4 処理単位の種類と実行
82	2.4.1 処理単位の種類
83	2.4.2 処理単位の実行順序
84	2.4.3 カーネル処理の不可分性
85	2.4.4 処理単位を実行するプロセッサ
86	2.5 システム状態とコンテキスト
87	2.5.1 カーネル動作状態と非動作状態
88	2.5.2 タスクコンテキストと非タスクコンテキスト
89	2.5.3 カーネルの振舞いに影響を与える状態
90	2.5.4 全割込みロック状態と全割込みロック解除状態
91	2.5.5 CPUロック状態とCPUロック解除状態
92	2.5.6 割込み優先度マスク
93	2.5.7 ディスパッチ禁止状態とディスパッチ許可状態
94	2.5.8 ディスパッチ保留状態
95	2.5.9 カーネル管理外の状態
96	2.5.10 処理単位の開始・終了とシステム状態
97	2.6 タスクの状態遷移とスケジューリング規則
98	2.6.1 基本的なタスク状態
99	2.6.2 タスクの状態遷移
100	2.6.3 タスクのスケジューリング規則

101	2.6.4 待ち行列と待ち解除の順序
102	2.6.5 タスク例外処理マスク状態と待ち禁止状態
103	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
104	2.6.7 制約タスク
105	2.7 割込み処理モデル
106	2.7.1 割込み処理の流れ
107	2.7.2 割込み優先度
108	2.7.3 割込み要求ラインの属性
109	2.7.4 割込みを受け付ける条件
110	2.7.5 割込み番号と割込みハンドラ番号
111	2.7.6 マルチプロセッサにおける割込み処理
112	2.7.7 カーネル管理外の割込み
113	2.7.8 カーネル管理外の割込みの設定方法
114	2.8 CPU例外処理モデル
115	2.8.1 CPU例外処理の流れ
116	2.8.2 CPU例外ハンドラから呼び出せるサービスコール
117	2.8.3 エミュレートされたCPU例外ハンドラ
118	2.8.4 カーネル管理外のCPU例外
119	2.9 システムの初期化と終了
120	2.9.1 システム初期化手順
121	2.9.2 システム終了手順
122	2.10 オブジェクトの登録とその解除
123	2.10.1 ID番号で識別するオブジェクト
124	2.10.2 オブジェクト番号で識別するオブジェクト
125	2.10.3 識別番号を持たないオブジェクト
126	2.10.4 オブジェクト生成に必要なメモリ領域
127	2.10.5 オブジェクトが属する保護ドメインの設定
128	2.10.6 オブジェクトが属するクラスの設定
129	2.10.7 オブジェクトの状態参照
130	2.11 オブジェクトのアクセス保護
131	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
132	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
133	2.11.3 デフォルトのアクセス許可ベクタ
134	2.11.4 アクセス許可ベクタの設定
135	2.11.5 カーネルの管理領域のアクセス保護
136	2.11.6 ユーザタスクのユーザスタック領域
137	2.12 システムコンフィギュレーション手順
138	2.12.1 システムコンフィギュレーションファイル
139	2.12.2 静的APIの文法とパラメータ
140	2.12.3 保護ドメインの指定
141	2.12.4 クラスの指定
142	2.12.5 コンフィギュレータの処理モデル
143	2.12.6 静的APIのパラメータに関するエラー検出
144	2.12.7 オブジェクトのID番号の指定
145	2.13 TOPPERSネーミングコンベンション
146	2.13.1 モジュール識別名
147	2.13.2 データ型名
148	2.13.3 関数名
149	2.13.4 変数名
150	2.13.5 定数名

151	2.13.6 マクロ名
152	2.13.7 静的API名
153	2.13.8 ファイル名
154	2.13.9 モジュール内部の名称の衝突回避
155	2.14 TOPPERS共通定義
156	2.14.1 TOPPERS共通ヘッダファイル
157	2.14.2 TOPPERS共通データ型
158	2.14.3 TOPPERS共通定数
159	2.14.4 TOPPERS共通エラーコード
160	2.14.5 TOPPERS共通マクロ
161	2.14.6 TOPPERS共通構成マクロ
162	2.15 カーネル共通定義
163	2.15.1 カーネルヘッダファイル
164	2.15.2 カーネル共通定数
165	2.15.3 カーネル共通マクロ
166	2.15.4 カーネル共通構成マクロ
167	
168	第3章 システムインタフェースレイヤAPI仕様
169	
170	3.1 システムインタフェースレイヤの概要
171	3.2 SILヘッダファイル
172	3.3 全割込みロック状態の制御
173	3.4 SILスピンロック
174	3.5 微少時間待ち
175	3.6 エンディアンの取得
176	3.7 メモリ空間アクセス関数
177	3.8 I/O空間アクセス関数
178	3.9 プロセッサIDの参照
179	
180	第4章 カーネルAPI仕様
181	
182	4.1 タスク管理機能
183	4.2 タスク付属同期機能
184	4.3 タスク例外処理機能
185	4.4 同期・通信機能
186	4.4.1 セマフォ
187	4.4.2 イベントフラグ
188	4.4.3 データキュー
189	4.4.4 優先度データキュー
190	4.4.5 メールボックス
191	4.4.6 ミューテックス
192	4.4.7 メッセージバッファ
193	4.4.8 スピンロック
194	4.5 メモリプール管理機能
195	4.5.1 固定長メモリプール
196	4.6 時間管理機能
197	4.6.1 システム時刻管理
198	4.6.2 周期ハンドラ
199	4.6.3 アラームハンドラ
200	4.6.4 オーバランハンドラ

201	4.7 システム状態管理機能
202	4.8 メモリオブジェクト管理機能
203	4.9 割込み管理機能
204	4.10 CPU例外管理機能
205	4.11 拡張サービスコール管理機能
206	4.12 システム構成管理機能
207	
208	第5章 リファレンス
209	
210	5.1 サービスコール一覧
211	5.2 静的API一覧
212	5.3 データ型
213	5.3.1 TOPPERS共通データ型
214	5.3.2 カーネルの使用データ型
215	5.3.3 カーネルの使用データ形式
216	5.4 定数とマクロ
217	5.4.1 TOPPERS共通定数
218	5.4.2 TOPPERS共通マクロ
219	5.4.3 カーネル共通定数
220	5.4.4 カーネル共通マクロ
221	5.4.5 カーネルの機能毎の定数
222	5.4.6 カーネルの機能毎のマクロ
223	5.5 構成マクロ
224	5.5.1 TOPPERS共通構成マクロ
225	5.5.2 カーネル共通構成マクロ
226	5.5.3 カーネルの機能毎の構成マクロ
227	5.6 エラーコード一覧
228	5.7 機能コード一覧
229	5.8 カーネルオブジェクトに対するアクセスの種別
230	5.9 ターゲット定義事項一覧
231	5.10 省略名の元になった英語
232	5.10.1 サービスコールと静的APIの名称の中のxxxの元になった英語
233	5.10.2 サービスコールと静的APIの名称の中のyyyの元になった英語
234	5.10.3 サービスコールの名称の中のzの元になった英語
235	5.11 バージョン履歴
236	
237	
238	○仕様書で用いる記述項目と記号
239	
240	この仕様書では、以下の記述項目を用いる。
241	
242	【補足説明】の項では、仕様本体の記述に対する補足事項を説明する。
243	
244	【～～カーネルにおける規定】の項では、TOPPERS新世代カーネルに属する特定
245	のカーネルにおける追加仕様を規定する。
246	
247	【～～仕様との関係】の項では、この仕様と、 μ ITRON4.0仕様または
248	μ ITRON4.0/PX仕様との違いについて説明する。
249	
250	【未決定事項】の項では、この仕様書の現時点のバージョンでは、決定されず

に残っている事項について記述する.

【仕様決定の理由】の項では、仕様を決定するにあたって考慮した事項について説明する.

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

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

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

【機能】の項では、サービスコールおよび静的APIの機能を規定する.

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

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

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

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

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

[S] は静的APIを示す.

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

[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. 保護機能対応カーネルでは、このAPIはサポートされない.

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

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

また、エラーが発生する条件を表すために、次の記号を用いる.

301
302 [s] は、サービスコールのみで発生するエラーを示す。静的APIでは、このエ
303 ラーは発生しない。
304
305 [S] は静的APIのみで発生するエラーを示す。サービスコールでは、このエラー
306 は発生しない。
307
308 [P] は保護機能対応カーネルのみで発生するエラーを示す。保護機能対応でな
309 いカーネルでは、このエラーは発生しない。
310
311 [D] は動的生成対応カーネルのみで発生するエラーを示す。動的生成対応でな
312 いカーネルでは、このエラーは発生しない。

313
314
315 ○タグの付与方法

316
317 この仕様書では、トレーサビリティの確保のために、記述事項に対してタグを
318 付与する。具体的には、以下に該当する記述事項を、タグを付与する対象とす
319 る。

- 320
321 • 対象ソフトウェアの実装に対する要求事項や制限事項
322 • 対象ソフトウェアの仕様に対する一般要求事項
323 • 対象ソフトウェアの動作環境に対する要求事項
324 • ターゲット定義の規定

325
326 それに対して、用語の定義や補足説明、対象ソフトウェアを使用する上での推
327 奨事項や注意事項、仕様決定の理由、他の仕様との関係に対しては、タグを付
328 与しない。

329
330 タグの形式と意味は次の通りである（xxxxは4桁の数字を表す）。

331

332	NGKIxxxx	TOPPERS新世代カーネル全体を対象とした記述
333	ASPSxxxx	TOPPERS/ASPカーネルを対象とした記述
334	FMPSxxxx	TOPPERS/FMPカーネルを対象とした記述
335	HRPSxxxx	TOPPERS/HRP2カーネルを対象とした記述
336	SSPSxxxx	TOPPERS/SSPカーネルを対象とした記述
337	ASSSxxxx	TOPPERS/ASP Safetyカーネルを対象とした記述

338

339 仕様書中では、ある記述事項に、タグYYYYxxxx（YYYYは4文字の英文字、xxxxは
340 4桁の数字を表す）が付与されていることを、【YYYYxxxx】で表現する。それ
341 に対して、タグYYYYxxxxを参照する場合には、[YYYYxxxx]と表記する。

342
343
344 第1章 TOPPERS新世代カーネルの概要

345
346 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
347 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、
348 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
349 適用対象領域と設計方針について述べる。

350

1.1 TOPPERS新世代カーネル仕様の位置付け

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）に準拠したリアルタイムカーネルを開発してきた。

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

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

451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500

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、アプリケーションで動的な負荷分散を実現することが可能である。

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が不要な範囲に留める。

1.5 TOPPERS/HRP2カーネルの適用対象領域と仕様設計方針

TOPPERS/HRP2カーネル（HRPは、High Reliable system Profileの略。2はバージョン番号を示す。以下、HRP2カーネル）は、さらに高い信頼性・安全性を要求される組込みシステムや、より大規模な組込みシステム向けに適用できるように、ASPカーネルを拡張したリアルタイムカーネルである。

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、特権モードと非特権モードを備え、メモリ保護のためにMMU（Memory Management Unit）またはMPU（Memory Protection Unit）を持つプロセッサを用いたシステムである。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、プログラムサイズ（バイナリコード）が数百KB以上のシステムである。

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。

1.6 TOPPERS/SSPカーネルの適用対象領域と仕様設計方針

TOPPERS/SSPカーネル（SSPは、Smallest Set Profileの略。以下、SSPカーネル）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絞り込んだリアルタイムカーネルである。

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カーネルにおいては、タスクは待ち状態を持たない（言い換えると、制約タスクのみをサポートする）のが最大の特徴である。また、ASPカーネルに対して下位互換性を持つように配慮しているが、システム全体のメモリ使用量を最小化するために有用な機能は、ASPカーネルに対して追加している。

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）が数KB～数十KB程度の極めて小規模な組込みシステムである。

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

TOPPERS/ASP Safetyカーネル（以下、ASP Safetyカーネル）は、小規模な安全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のかかる機能はサポートしない。

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ステムとする。ソフトウェア規模の面では、プログラムサイズ（バイナリコード）が数十KB～1MB程度のシステムを主な適用対象とする。それより大規模なシステムには、保護機能を持ったカーネルを適用すべきと考えられる。

第2章 主要な概念と共通定義

2.1 仕様の位置付け

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

2.1.1 カーネルの機能セット

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

この仕様では、TOPPERS新世代カーネルを構成する一連のカーネルの仕様を統合的に記述するが、言うまでもなく、カーネルの種類によってサポートする機能は異なる。サポートする機能をカーネルの種類毎に記述する方法もあるが、カーネルの種類はユーザ要求に対応して増える可能性もあり、その度に仕様書を修正するのは得策ではない。

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

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルは、保護機能対応カーネル、マルチプロセッサ対応カーネル、動的生成対応カーネルのいずれでもない【ASPS0001】。ただし、動的生成機能拡張パッケージを用いると、動的生成対応カーネルの機能の一部がサポートされる【ASPS0002】。

【TOPPERS/FMPカーネルにおける規定】

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

【TOPPERS/HRP2カーネルにおける規定】

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

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルは、保護機能対応カーネル、マルチプロセッサ対応カーネル、動的生成対応カーネルのいずれでもない【SSPS0001】。

【 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）を規定する。

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

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

【 μ ITRON4.0仕様との関係】

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

2.1.4 想定するハードウェア構成

この仕様では、カーネルがサポートするハードウェア構成として、以下のことを想定している。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責任になる。

(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、異なるメモリを同一のメモリ番地でアクセスすることがないこと）【NGKI0001】。マルチプロセッサ対応カーネルにおいては、同一のメモリに対しては、各プロセッサから同一の番地でアクセスできること【NGKI0002】。

(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命令を実行できること【NGKI0003】。

2.1.5 想定するプログラミング言語

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ISO/IEC 9899:1999（以下、C99と呼ぶ）に準拠したC言語を、フリースタANDING環境で用いることを想定して規定している【NGKI0004】。

ただし、C90の規定に加えて、以下のことを仮定している。

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

2.2 APIの構成要素とコンベンション

2.2.1 APIの構成要素

(1) サービスコール

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

(2) コールバック

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

(3) 静的API

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

(4) 構成マクロ

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

2.2.2 パラメータとリターンパラメータ

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

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

【NGKI0007】。また、パラメータのサイズが大きい場合にも、パラメータを入れた領域へのポインタをパラメータとして渡す場合がある【NGKI0008】。

C言語APIでは、リターンパラメータは、関数の返値とするか、リターンパラメータを入れる領域へのポインタをパラメータとして渡すことで実現する

【NGKI0009】。オブジェクトの状態を参照するサービスコールなど、リターンパラメータの数が多い場合やターゲット定義のリターンパラメータを追加する可能性がある場合には、複数のリターンパラメータを1つの構造体に入れて返すこととし、その領域へのポインタをパラメータとして渡す【NGKI0010】。

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

サービスコールやコールバックに、パケットを置く領域へのポインタやリターンパラメータを入れる領域へのポインタを渡す場合、別に規定がない限りは、サービスコールやコールバックの処理が完了した後は、それらの領域が参照されることはなく、別の目的に使用できる【NGKI0011】。

2.2.3 返値とエラーコード

一部の例外を除いて、サービスコールおよびコールバックの返値は、処理が正常終了したかを表す符号付き整数とする。処理が正常終了した場合には、E_OK (=0) または正の値が返るものとし、値の意味はサービスコールまたはコールバック毎に定める【NGKI0012】。処理が正常終了しなかった場合には、その原因を表す負の値が返る【NGKI0013】。処理が正常終了しなかった原因を表す値を、エラーコード (error code) と呼ぶ。

エラーコードは、いずれも負の値のメインエラーコードとサブエラーコードで構成される【NGKI0014】。メインエラーコードとサブエラーコードからエラーコードを構成するマクロ (ERCD) と、エラーコードからメインエラーコードを取り出すマクロ (MERCD)、サブエラーコードを取り出すマクロ (SERCD) が用意されている【NGKI0015】。

メインエラーコードの名称・意味・値は、カーネルとシステムサービスで共通に定める（「2.14.4 TOPPERS共通エラーコード」の節を参照）【NGKI0016】。サービスコールおよびコールバックの機能説明中の「E_XXXXXエラーとなる」または「E_XXXXXエラーが返る」という記述は、メインエラーコードとしてE_XXXXXが返ることを意味する。

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

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

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

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

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

2.2.4 機能コード

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

2.2.5 ヘッダファイル

カーネルやシステムサービスを用いるために必要な定義を含むファイル。

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

2.3 主な概念

2.3.1 オブジェクトと処理単位

(1) オブジェクト

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

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

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

(2) 処理単位

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

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

拡張情報 (extended information) は、処理単位が呼び出される時にパラメータとして渡される情報で、処理単位の登録時に指定する【NGKI0031】. 拡張情報は、カーネルやシステムサービスの動作には影響しない【NGKI0032】.

(3) タスク

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

サービスコールの機能説明において、サービスコールを呼び出したタスクを、自タスク (invoking task) と呼ぶ. 拡張サービスコールからサービスコールを呼び出した場合には、拡張サービスコールを呼び出したタスクが自タスクである.

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

【補足説明】

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

(4) ディスパッチとスケジューリング

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

ディスパッチが起こるべき状態 (すなわち、スケジューリングによって、現在実行しているタスクとは異なるタスクが、実行すべきタスクに決定されている状態) となっても、何らかの理由でディスパッチを行わないことを、ディスパッチの保留 (pend dispatching) という. ディスパッチを行わない理由が解除された時点で、ディスパッチが起こる【NGKI0034】.

(5) 割込みとCPU例外

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

周辺デバイスからの割込み要求をプロセッサに伝える経路を遮断し、割込み要

851 求が受け付けられるのを抑止することを、割込みのマスキング (mask interrupt)
852 または割込みの禁止 (disable interrupt) という。マスキングが解除された時点で、
853 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
854 【NGKI0035】。

855
856 マスキングすることができない割込みを、NMI (non-maskable interrupt) と呼ぶ。
857

858 【μ ITRON4.0仕様との関係】

859
860 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
861 語を定義した。
862

863 (6) タイムイベントとタイムイベントハンドラ

864
865 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
866 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、
867 タイムイベントハンドラ (time event handler) と呼ぶ。
868

869 2.3.2 サービスコールとパラメータ

870 871 (1) 優先順位と優先度

872
873 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
874 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
875 高い処理単位が実行される【NGKI0036】。

876
877 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
878 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
879 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
880 続した正の値を用いるのを原則とする【NGKI0037】。優先度は、値が小さいほ
881 ど優先度が高い（すなわち、先に実行または配送される）ものとする
882 【NGKI0038】。

883 884 (2) システム時刻と相対時間

885
886 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
887 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
888 【NGKI0039】。システム時刻は、タイムティック (time tick) を通知するため
889 のタイマ割込みが発生する毎に更新される【NGKI0040】。

890
891 イベントが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
892 相対時間 (relative time) によって指定する【NGKI0041】。基準時刻は、別に
893 規定がない限りは、相対時間を指定するサービスコールを呼び出した時刻とな
894 る【NGKI0042】。

895
896 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
897 同一、すなわちミリ秒とする【NGKI0043】。相対時間には、少なくとも、16ビッ
898 トの符号無しの整数型 (uint16_t型) に格納できる任意の値を指定することが
899 できるが、RELTIM型 (uint_t型に定義される) に格納できる任意の値を指定で
900 けるとは限らない【NGKI0044】。相対時間に指定できる最大値は、構成マクロ

901 TMAX_RELTIMに定義されている【NGKI0045】.

902

903 イベントを発生させる時刻を相対時間で指定した場合、イベントの処理が行われ
904 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
905 なる【NGKI0046】. ただし、基準時刻を定めるサービスコールを呼び出した時
906 に、タイムティックを通知するためのタイマ割込みがマスクされている場合
907 (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
908 む) は、相対時間によって指定した以上の時間が経過した後となることは保証
909 されない【NGKI0047】.

910

911 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
912 返される【NGKI0048】. 基準時刻は、相対時間を返すサービスコールを呼び出
913 した時刻となる【NGKI0049】.

914

915 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
916 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
917 【NGKI0050】. ただし、相対時間を返すサービスコールを呼び出した時に、タ
918 イムティックを通知するためのタイマ割込みがマスクされている場合(タイマ
919 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、
920 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
921 【NGKI0051】.

922

923 【補足説明】

924

925 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
926 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
927 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
928 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
929 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
930 ある.

931

932 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
933 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
934 降のタイムティックでイベントの処理が行われる.

935

936 【 μ ITRON4.0仕様との関係】

937

938 相対時間(RELTIM型)とシステム時刻(SYSTIM型)の時間単位は、 μ ITRON4.0
939 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
940 時間の解釈について、より厳密に規定した.

941

942 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.

943

944 (3) タイムアウトとポーリング

945

946 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
947 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
948 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
949 エラーが返る【NGKI0052】.

950

タイムアウトを起こすまでの時間（タイムアウト時間）は、符号付きの整数型であるTMO型で表し、単位はシステム時刻と同一、すなわちミリ秒とする

【NGKI0053】．タイムアウト時間に正の値を指定した場合には、タイムアウトを起こすまでの相対時間を表す【NGKI0054】．すなわち、タイムアウトの処理が行われるのは、サービスコールを呼び出してから指定した以上の時間が経過した後となる．

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

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

タイムアウト付きのサービスコールは、別に規定がない限りは、タイムアウト時間にTMO_POL（＝0）を指定した場合にはポーリングを行い、TMO_FEVR（＝-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【NGKI0059】．

【補足説明】

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

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

【μ ITRON4.0仕様との関係】

タイムアウト時間（TMO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．

【仕様決定の理由】

ディスパッチ保留状態において、ポーリングを行うサービスコールを呼び出せる場合があるのに対して、タイムアウト付きのサービスコールをタイムアウト時間をTMO_POLとして呼び出すとエラーになるのは、割込み優先度マスクが全解除でない状態やディスパッチ禁止状態では、自タスクを広義の待ち状態に遷移させる可能性のあるサービスコール（タイムアウト付きのサービスコールはこれに該当）を呼び出すことはできないという原則【NGKI0175】と【NGKI0179】

1001 があるためである.

1002

1003 (4) ノンブロッキング

1004

1005 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
1006 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
1007 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
1008 する場合、サービスコールからはE_WBLKエラーが返る【NGKI0060】。E_WBLKは
1009 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
1010 則は適用されない【NGKI0061】。

1011

1012 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
1013 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
1014 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
1015 きない【NGKI0062】。継続している処理が完了した場合や、何らかの理由で処
1016 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
1017 スコールを呼び出したソフトウェアに通知するものとする【NGKI0063】。

1018

1019 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
1020 ことによって行う【NGKI0064】。ノンブロッキングの指定を行えるサービスコー
1021 ルは、指定した場合の振舞いをサービスコール毎に規定する【NGKI0065】。

1022

1023 【補足説明】

1024

1025 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
1026 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
1027 していない。

1028

1029 2.3.3 保護機能

1030

1031 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、
1032 保護機能対応カーネルにのみ適用される。

1033

1034 (1) アクセス保護

1035

1036 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
1037 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
1038 を防ぐアクセス保護機能を提供する【NGKI0066】。

1039

1040 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
1041 が対象 (object) ということになる。

1042

1043 (2) メモリオブジェクト

1044

1045 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
1046 扱い、アクセス保護の対象とする【NGKI0067】。カーネルがアクセス保護の対
1047 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。
1048 メモリオブジェクトは、互いに重なりあうことはない【NGKI0068】。

1049

1050 メモリオブジェクトは、その先頭番地によって識別する【NGKI0069】。言い換

1051 えると、先頭番地がオブジェクト番号となる。
1052
1053 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
1054 リ保護が実現できるように、ターゲット定義の制約が課せられる【NGKI0070】。
1055
1056 (3) 保護ドメイン
1057
1058 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
1059 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
1060 号によって識別する【NGKI0071】。
1061
1062 カーネルオブジェクトは、ただだか1つの保護ドメインに属する。処理単位は、
1063 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
1064 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
1065 【NGKI0072】。いずれの保護ドメインにも属さないカーネルオブジェクトを、
1066 無所属のカーネルオブジェクト (independent kernel object) と呼ぶ。
1067
1068 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
1069 する保護ドメインにより決まるのが原則である【NGKI0073】。すなわち、カー
1070 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
1071 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
1072 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。
1073
1074 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
1075 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
1076 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）【NGKI0074】。
1077 これは、【NGKI0073】の原則の例外となっている。
1078
1079 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
1080 メイン（とカーネルドメイン）のみからアクセスできる【NGKI0075】。また、
1081 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
1082 【NGKI0076】。
1083
1084 (4) カーネルドメインとユーザドメイン
1085
1086 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
1087 1つ存在する【NGKI0077】。カーネルドメインに属する処理単位は、プロセッサ
1088 の特権モードで実行される【NGKI0078】。また、すべてのカーネルオブジェク
1089 トに対して、すべての種別のアクセスを行うことが許可される【NGKI0079】。
1090 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」
1091 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
1092 る。
1093
1094 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
1095 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
1096 れる【NGKI0080】。また、どのカーネルオブジェクトに対してどの種別のアク
1097 セスを行えるかを制限することができる【NGKI0081】。
1098
1099 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
1100 【NGKI0082】。カーネルドメインの保護ドメインIDは、TDOM_KERNEL (=-1) で

1101 ある【NGKI0083】.

1102

1103 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
1104 する【NGKI0084】. 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
1105 ンフィギュレータがエラーを報告する【NGKI0085】.

1106

1107 【補足説明】

1108

1109 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
1110 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
1111 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
1112 意していない.

1113

1114 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
1115 みなすこともできる.

1116

1117 【μ ITRON4.0/PX仕様との関係】

1118

1119 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
1120 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
1121 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
1122 できる保護ドメインである.

1123

1124 (5) システムタスクとユーザタスク

1125

1126 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
1127 メインに属するタスクをユーザタスク（user task）と呼ぶ.

1128

1129 【補足説明】

1130

1131 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
1132 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
1133 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.

1134

1135 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
1136 ぶことになる.

1137

1138 (6) アクセス許可パターン

1139

1140 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
1141 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
1142 許可パターン（access permission pattern）と呼ぶ. アクセス許可パターンの
1143 各ビットは、1つのユーザドメインに対応する【NGKI0086】. カーネルドメイン
1144 には、すべてのアクセスが許可されているため、カーネルドメインに対応する
1145 ビットは用意されていない.

1146

1147 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

1148 （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
1149 可されていることを表す【NGKI0087】. そのため、2つのアクセス許可パターン
1150 のビット毎論理和（C言語の“|”）を求めることで、アクセスを許可されている

1151 ユーザドメインの和集合 (union) を得ることができる。また、2つのアクセス
1152 許可パターンのビット毎論理積 (C言語の"&") を求めることで、アクセスを許
1153 可されているユーザドメインの積集合 (intersection) を得ることができる。
1154

1155 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
1156 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)
1157 が用意されている【NGKI0088】。また、カーネルドメインのみにアクセスを許
1158 可することを示すアクセス許可パターンを表す定数 (TACP_KERNEL) と、すべて
1159 の保護ドメインにアクセスを許可することを示すアクセス許可パターンを表す
1160 定数 (TACP_SHARED) が用意されている【NGKI0089】。
1161

1162 (7) アクセス許可ベクタ

1163
1164 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、
1165 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
1166 【NGKI0090】。あるカーネルオブジェクトに対する4つの種別のアクセスに関す
1167 るアクセス許可パターンをひとまとめにしたものを、アクセス許可ベクタ
1168 (access permission vector) と呼び、次のように定義されるデータ型
1169 (ACVCT) で保持する【NGKI0091】。
1170

```
1171     typedef struct acvct {  
1172         ACPTN    acptn1;    /* 通常操作1のアクセス許可パターン */  
1173         ACPTN    acptn2;    /* 通常操作2のアクセス許可パターン */  
1174         ACPTN    acptn3;    /* 管理操作のアクセス許可パターン */  
1175         ACPTN    acptn4;    /* 参照操作のアクセス許可パターン */  
1176     } ACVCT;
```

1177 【補足説明】

1178
1179
1180 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
1181 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。
1182

1183 【μ ITRON4.0/PX仕様との関係】

1184
1185 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
1186 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
1187 めている。
1188

1189 (8) サービスコールの呼出し方法

1190
1191 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
1192 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、
1193 ソフトウェア割込みによって呼び出すコードが生成される【NGKI0092】。
1194

1195 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
1196 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
1197 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。
1198 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
1199 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。
1200

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

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

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

【補足説明】

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

(9) ユーザドメインから行える処理に対する制限

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

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

- ・指定できる最高のタスク優先度

なお、カーネルドメインに対しては、制限を設ける機能を用意していない。すなわち、カーネルドメインに属する処理単位は、すべてのタスク優先度を使うことができる【NGKI0532】。

2.3.4 マルチプロセッサ対応

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

(1) クラス

マルチプロセッサに対応するために用いるカーネルオブジェクトの集合を、クラス (class) と呼ぶ。クラスは、クラスIDと呼ぶID番号によって識別する【NGKI0095】。

カーネルオブジェクトは、いずれか1つのクラスに属するのが原則である【NGKI0096】。カーネルオブジェクトが属するクラスは、オブジェクトの登録時に決定し、登録後に変更することはできない【NGKI0097】。

【補足説明】

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

【NGKI0096】の原則に関わらず、以下のオブジェクトはいずれのクラスにも属さない。

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

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

(2) プロセッサ

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

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

システムの初期化時と終了時に特別な役割を果たすプロセッサを、マスタプロセッサ (master processor) と呼び、システムに1つ存在する【NGKI0100】。どのプロセッサをマスタプロセッサとするかは、ターゲット定義である

【NGKI0101】。マスタプロセッサ以外のプロセッサを、スレーブプロセッサ (slave processor) と呼ぶ。なお、カーネル動作状態では、マスタプロセッサとスレーブプロセッサの振舞いに違いはない【NGKI0102】。

(3) 処理単位の割付けとマイグレーション

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

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

1301
1302 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
1303 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
1304 とができる【NGKI0105】。

1305
1306 (4) クラスの持つ属性とカーネルオブジェクト

1307
1308 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
1309 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
1310 ブジェクトが属するクラスによって定まる。

1311
1312 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
1313 の通りである【NGKI0106】。

1314
1315 ・初期割付けプロセッサ
1316 ・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
1317 サを含む）
1318 ・ATT_MOD／ATA_MODによって、オブジェクトモジュールに含まれる標準のセ
1319 クションが配置されるメモリリージョン（標準メモリリージョン）
1320 ・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
1321 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
1322 場所
1323 ・その他の管理情報（ロック単位など）

1324
1325 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【NGKI0107】。

1326
1327 【仕様決定の理由】

1328
1329 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
1330 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
1331 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
1332 しておく方が良いと考えたためである。

1333
1334 (5) ローカルタイマ方式とグローバルタイマ方式

1335
1336 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
1337 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
1338 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
1339 である【NGKI0108】。

1340
1341 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
1342 サが更新する【NGKI0109】。異なるプロセッサのシステム時刻を同期させる機
1343 能は、カーネルでは用意しない。

1344
1345 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
1346 新する【NGKI0110】。これを、システム時刻管理プロセッサと呼ぶ。どのプロ
1347 セッサをシス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
1348 【NGKI0111】。

1349
1350 【補足説明】

1351
1352 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。
1353
1354 **【未決定事項】**
1355
1356 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
1357 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、
1358 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。
1359
1360 2.3.5 その他
1361
1362 (1) オブジェクトモジュール
1363
1364 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
1365 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
1366 ブジェクトモジュールである。
1367
1368 (2) メモリリージョン
1369
1370 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
1371 た連続したメモリ領域をメモリリージョン (memory region) と呼ぶ。
1372
1373 メモリリージョンは、文字列によって識別する【NGKI0112】。メモリリージョ
1374 ンを識別する文字列を、メモリリージョン名と呼ぶ。
1375
1376 **【補足説明】**
1377
1378 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
1379 囲という一般的な意味で使っている。
1380
1381 (3) 標準のセクション
1382
1383 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
1384 ン (standard sections) と呼ぶ。コンパイラが出力しないセクションの中で、
1385 ターゲット定義のものを、標準のセクションと扱う場合もある【NGKI0113】。
1386
1387 (4) 保護ドメイン毎の標準セクション
1388
1389 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
1390 置するためのセクションが登録される【NGKI0114】。また、無所属の標準のセ
1391 クションを配置するためのセクションが登録される【NGKI0115】。これらのセ
1392 クションを、保護ドメイン毎の標準セクションと呼ぶ (standard sections
1393 for each protection domain)。保護ドメイン毎の標準セクションのセクショ
1394 ン名は、ターゲット定義で別に規定がない限りは、標準のセクション名と保護
1395 ドメイン名 (カーネルドメインの場合は "kernel", 無所属の場合は "shared")
1396 を "_" でつないだものとする【NGKI0116】。例えば、カーネルドメインの
1397 ".text" セクションのセクション名は、".text_kernel" とする。
1398
1399 2.4 処理単位の種類と実行順序
1400

1401 2.4.1 処理単位の種類

1402

1403 カーネルが実行を制御する処理単位の種類は次の通りである【NGKI0117】.

1404

1405 (a) タスク

1406 (a.1) タスク例外処理ルーチン

1407 (b) 割込みハンドラ

1408 (b.1) 割込みサービスルーチン

1409 (b.2) タイムイベントハンドラ

1410 (c) CPU例外ハンドラ

1411 (d) 拡張サービスコール

1412 (e) 初期化ルーチン

1413 (f) 終了処理ルーチン

1414

1415 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
1416 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
1417 ある.

1418

1419 【TOPPERS/ASPカーネルにおける規定】

1420

1421 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
1422 ない【ASPS0003】. ただし、オーバランハンドラ機能拡張パッケージを用いる
1423 と、オーバランハンドラ機能を追加することができる【ASPS0004】.

1424

1425 【TOPPERS/FMPカーネルにおける規定】

1426

1427 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
1428 ない【FMPS0002】.

1429

1430 【TOPPERS/SSPカーネルにおける規定】

1431

1432 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
1433 ビスコールをサポートしていない【SSPS0002】.

1434

1435 2.4.2 処理単位の実行順序

1436

1437 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
1438 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
1439 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
1440 る.

1441

1442 タスクの優先順位は、ディスパッチャの優先順位よりも低い【NGKI0118】. タ
1443 スク間では、高い優先度を持つ方が優先順位が高く、同じ優先度を持つタスク
1444 間では、先に実行できる状態となった方が優先順位が高い【NGKI0119】. 詳し
1445 くは、「2.6.3 タスクのスケジューリング規則」の節を参照すること.

1446

1447 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
1448 が、タスクよりも先に実行される【NGKI0120】.

1449

1450 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

1451 【NGKI0121】. 割込みハンドラ間では、高い割込み優先度を持つ方が優先順位
1452 が高く、同じ割込み優先度を持つ割込みハンドラ間では、先に実行開始された
1453 方が優先順位が高い【NGKI0122】. 同じ割込み優先度を持つ割込みハンドラ間
1454 での実行開始順序は、この仕様では規定しない. 詳しくは、「2.7.2 割込み優
1455 先度」の節を参照すること.

1456

1457 割込みサービスルーチンとタイムイベントハンドラの優先順位は、それを呼び
1458 出す割込みハンドラと同じである【NGKI0123】.

1459

1460 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
1461 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
1462 ッチャよりも先に実行される【NGKI0124】. CPU例外がその他の処理単位で発生
1463 した場合には、CPU例外ハンドラの優先順位は、その処理単位の優先順位と同じ
1464 であるが、その処理単位よりも先に実行される【NGKI0125】.

1465

1466 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、
1467 それを呼び出した処理単位よりも先に実行される【NGKI0126】.

1468

1469 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
1470 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
1471 行される【NGKI0127】. 終了処理ルーチンは、カーネルの動作終了後に、終了
1472 処理ルーチンを登録する静的APIを記述したのと逆の順序で実行される
1473 【NGKI0128】.

1474

1475 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
1476 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
1477 【NGKI0129】. グローバル初期化ルーチンがマスタプロセッサで実行された後
1478 に、各プロセッサでローカル初期化ルーチンが実行される【NGKI0130】. また、
1479 終了処理ルーチンには、クラスに属さないグローバル終了処理ルーチンと、ク
1480 ラスに属するローカル終了処理ルーチンがある【NGKI0131】. ローカル終了処
1481 理ルーチンが各プロセッサで実行された後に、マスタプロセッサでグローバル
1482 終了処理ルーチンが実行される【NGKI0132】.

1483

1484 【仕様決定の理由】

1485

1486 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
1487 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
1488 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
1489 順に記述するのは難しい）.

1490

1491 2.4.3 カーネル処理の不可分性

1492

1493 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
1494 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
1495 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
1496 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、
1497 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
1498 【NGKI0133】. これを、カーネル処理の不可分性という.

1499

1500 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

1501 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
1502 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
1503 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
1504 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
1505 【NGKI0134】。

1506
1507 【補足説明】

1508
1509 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
1510 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
1511 スパッチは、すべてのタスクの状態遷移が完了した後に行われる。例えば、低
1512 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
1513 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
1514 れた後に、タスクCへのディスパッチが行われる。

1515
1516 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
1517 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
1518 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
1519 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
1520 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
1521 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
1522 がある。

1523
1524 2.4.4 処理単位を実行するプロセッサ

1525
1526 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
1527 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
1528 け可能プロセッサから、次のように決まる。

1529
1530 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
1531 割付けプロセッサに割り付けられる【NGKI0135】。また、割付けプロセッサを
1532 変更するサービスコール（mact_tsk/imact_tsk, mig_tsk, msta_cyc,
1533 msta_alm/imsta_alm）によって、割付けプロセッサを、クラスの割付け可能プ
1534 ロセッサのいずれかに変更することができる【NGKI0136】。

1535
1536 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
1537 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
1538 【NGKI0137】。クラスの割付け可能プロセッサの情報は用いられない。

1539
1540 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
1541 （オプション設定によりすべて）で実行される【NGKI0138】。クラスの初期割
1542 付けプロセッサの情報は用いられない。

1543
1544 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
1545 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。

1546	初期割付けプロセッサ	割付け可能プロセッサ
1547	-----	
1548		
1549	タスク（タスク例外処理	○
1550	ルーチンを含む）	○

1551	-----		
1552	割込みハンドラ	○	—
1553	割込みサービスルーチン	—	○
1554	周期ハンドラ	○	○
1555	アラームハンドラ	○	○
1556	-----		
1557	CPU例外ハンドラ	○	—
1558	-----		
1559	ローカル初期化ルーチン	○	—
1560	ローカル終了処理ルーチン	○	—
1561	-----		
1562			
1563	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		
1564	バル終了処理ルーチンは，いずれのクラスにも属さない【NGKI0139】．オーバ		
1565	ランハンドラは，オーバランを起こしたタスクの割付けプロセッサによって実		
1566	行される【NGKI0140】．拡張サービスコールは，それを呼び出した処理単位		
1567	の割付けプロセッサによって実行される【NGKI0141】．グローバル初期化ルーチ		
1568	ンとグローバル終了処理ルーチンは，マスタプロセッサによって実行される		
1569	【NGKI0142】．		
1570			
1571	2.5 システム状態とコンテキスト		
1572			
1573	2.5.1 カーネル動作状態と非動作状態		
1574			
1575	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，		
1576	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		
1577	（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		
1578	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		
1579	態かカーネル非動作状態のいずれかの状態を取る【NGKI0143】．		
1580			
1581	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		
1582	る【NGKI0144】．		
1583			
1584	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		
1585	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		
1586	【NGKI0145】．カーネル非動作状態で，その他のサービスコールを呼び出した		
1587	場合の動作は，保証されない【NGKI0146】．		
1588			
1589	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		
1590	ネル非動作状態のいずれかの状態を取る【NGKI0147】．		
1591			
1592	2.5.2 タスクコンテキストと非タスクコンテキスト		
1593			
1594	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		
1595	ど）をコンテキストと呼ぶ．		
1596			
1597	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		
1598	ンテキストと非タスクコンテキストに分類される【NGKI0148】．		
1599			
1600	タスク（タスク例外処理ルーチンを含む）が実行されるコンテキストは，タス		

クコンテキストに分類される【NGKI0149】。また、タスクコンテキストから呼び出した拡張サービスコールが実行されるコンテキストは、タスクコンテキストに分類される【NGKI0150】。

割込みハンドラ（割込みサービスルーチンおよびタイムイベントハンドラを含む）とCPU例外ハンドラが実行されるコンテキストは、非タスクコンテキストに分類される【NGKI0151】。また、非タスクコンテキストから呼び出した拡張サービスコールが実行されるコンテキストは、非タスクコンテキストに分類される【NGKI0152】。

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

タスクコンテキストからは、非タスクコンテキスト専用のサービスコールを呼び出すことはできない【NGKI0155】。逆に、非タスクコンテキストからは、タスクコンテキスト専用のサービスコールを呼び出すことはできない【NGKI0156】。いずれも、呼び出した場合にはE_CTXエラーとなる【NGKI0157】。

2.5.3 カーネルの振舞いに影響を与える状態

カーネル動作状態において、プロセッサは、カーネルの振舞いに影響を与える状態として、次の状態を持つ【NGKI0158】。

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

これらの状態は、それぞれ独立な状態である。すなわち、プロセッサは上記の状態の任意の組合せを取ることができ、それぞれの状態を独立に変化させることができる【NGKI0159】。

2.5.4 全割込みロック状態と全割込みロック解除状態

プロセッサは、NMIを除くすべての割込みをマスクするための全割込みロックフラグを持つ【NGKI0160】。全割込みロックフラグがセットされた状態を全割込みロック状態、クリアされた状態を全割込みロック解除状態と呼ぶ。すなわち、全割込みロック状態では、NMIを除くすべての割込みがマスクされる。

全割込みロック状態では、システムインタフェースレイヤのAPIとカーネル非動作状態を参照するサービスコール（sns_ker）、カーネルを終了するサービスコール（ext_ker）のみを呼び出すことができる【NGKI0161】。全割込みロック状態で、その他のサービスコール（拡張サービスコールを含む）を呼び出した場合の動作は、保証されない【NGKI0162】。また、全割込みロック状態で、実行中の処理単位からリターンしてはならない。リターンした場合の動作は保証されない【NGKI0164】。

マルチプロセッサ対応カーネルでは、プロセッサ毎に、全割込みロックフラグ

1651 を持つ【NGKI0165】。すなわち、プロセッサ毎に、全割込みロック状態か全割
1652 込みロック解除状態のいずれかの状態を取る。

1653

1654 2.5.5 CPUロック状態とCPUロック解除状態

1655

1656 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
1657 節を参照）をすべてマスクするためのCPUロックフラグを持つ【NGKI0166】。
1658 CPUロックフラグがセットされた状態をCPUロック状態、クリアされた状態を
1659 CPUロック解除状態と呼ぶ。CPUロック状態では、すべてのカーネル管理の割込
1660 みがマスクされ、ディスパッチが保留される【NGKI0167】。

1661

1662 CPUロック状態で呼び出すことができるサービスコールは次の通り【NGKI0168】。

1663

- 1664 ・システムインタフェースレイヤのAPI
- 1665 ・loc_cpu/iloc_cpu, unl_cpu/iunl_cpu
- 1666 ・unl_spn/iunl_spn（マルチプロセッサ対応カーネルのみ）
- 1667 ・dis_int, ena_int
- 1668 ・sns_yyy
- 1669 ・xsns_yyy（CPU例外ハンドラからのみ）
- 1670 ・get_utm
- 1671 ・ext_tsk, ext_ker
- 1672 ・prb_mem（保護機能対応カーネルのみ）
- 1673 ・cal_svc（保護機能対応カーネルのみ）

1674

1675 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
1676 となる【NGKI0169】。

1677

1678 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
1679 【NGKI0170】。すなわち、プロセッサ毎に、CPUロック状態かCPUロック解除状
1680 態のいずれかの状態を取る。

1681

1682 【補足説明】

1683

1684 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
1685 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。

1686

1687 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
1688 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
1689 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、
1690 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
1691 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。

1692

1693 2.5.6 割込み優先度マスク

1694

1695 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
1696 マスクを持つ【NGKI0171】。割込み優先度マスクがTIPM_ENAALL（=0）の時は、
1697 いずれの割込み要求もマスクされない【NGKI0172】。この状態を割込み優先度
1698 マスク全解除状態と呼ぶ。割込み優先度マスクがTIPM_ENAALL（=0）以外の時
1699 は、割込み優先度マスクと同じかそれより低い割込み優先度を持つ割込みはマ
1700 スクされ、ディスパッチは保留される【NGKI0173】。この状態を割込み優先度

1701 マスクが全解除でない状態と呼ぶ。

1702

1703 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
1704 スクを広義の待ち状態に遷移させる可能性のあるサービスコールを呼び出すこと
1705 はできない。呼び出した場合には、E_CTXエラーとなる【NGKI0175】。

1706

1707 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
1708 持つ【NGKI0176】。

1709

1710 2.5.7 ディスパッチ禁止状態とディスパッチ許可状態

1711

1712 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
1713 【NGKI0177】。ディスパッチ禁止フラグがセットされた状態をディスパッチ禁
1714 止状態、クリアされた状態をディスパッチ許可状態と呼ぶ。すなわち、ディス
1715 パッチ禁止状態では、ディスパッチは保留される。

1716

1717 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
1718 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
1719 出した場合には、E_CTXエラーとなる【NGKI0179】。

1720

1721 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
1722 グを持つ【NGKI0180】。すなわち、プロセッサ毎に、ディスパッチ禁止状態か
1723 ディスパッチ許可状態のいずれかの状態を取る。

1724

1725 【補足説明】

1726

1727 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
1728 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。
1729 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
1730 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
1731 現することはできない。

1732

1733 2.5.8 ディスパッチ保留状態

1734

1735 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
1736 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
1737 【NGKI0181】。これらの状態を総称して、ディスパッチ保留状態と呼ぶ。

1738

1739 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
1740 かそうでない状態のいずれかの状態を取る【NGKI0182】。

1741

1742 【補足説明】

1743

1744 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
1745 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。

1746

1747 2.5.9 カーネル管理外の状態

1748

1749 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
1750 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

1751 （「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
1752 外の状態と呼ぶ。

1753

1754 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,
1755 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
1756 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
1757 出すことはできない【NGKI0543】。カーネル管理外の状態から、その他のサー
1758 ビスコールを呼び出した場合の動作は、保証されない【NGKI0544】。

1759

1760 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
1761 ている【NGKI0545】。カーネル管理外の割込み（の一部）もマスクされている
1762 場合もある【NGKI0546】。保護機能対応カーネルでは、カーネル管理外の状態
1763 になるのは、特権モードで実行している間に限られる【NGKI0547】。

1764

1765 2.5.10 処理単位の開始・終了とシステム状態

1766

1767 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
1768 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
1769 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
1770 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
1771 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
1772 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。

1773

	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
--	---------------	---------------	-----------------

1776

1777 【タスク】 【NGKI0183】

1778 実行開始条件	解除	全解除	許可
1779 実行開始時処理	そのまま	そのまま	そのまま
1780 終了前	原則解除(*1)	原則全解除(*1)	原則許可(*1)
1781 終了時処理	解除する	全解除する	許可する

1782

1783 【タスク例外処理ルーチン】 【NGKI0184】

1784 実行開始条件	解除	全解除	任意
1785 実行開始時処理	そのまま	そのまま	そのまま
1786 リターン前	原則解除(*1)	原則全解除(*1)	元に戻す
1787 リターン時処理	解除する	全解除する	元に戻す(*4)

1788

1789 【カーネル管理の割込みハンドラ】 【NGKI0185】

1790 【割込みサービスルーチン】 【NGKI0186】

1791 【タイムイベントハンドラ】 【NGKI0187】

1792 実行開始条件	解除	自優先度より低い	任意
1793 実行開始時処理	そのまま	自優先度に(*2)	そのまま
1794 リターン前	原則解除(*1)	変更不可(*3)	変更不可(*3)
1795 リターン時処理	解除する	元に戻す(*5)	そのまま

1796

1797 【CPU例外ハンドラ】 【NGKI0188】

1798 実行開始条件	任意	任意	任意
1799 実行開始時処理	そのまま(*6)	そのまま	そのまま
1800 リターン前	原則元に(*1)	変更不可(*3)	変更不可(*3)

1801 リターン時処理 元に戻す 元に戻す(*5) そのまま
1802 -----

1803 **【拡張サービスコール】 【NGKI0189】**

1804	実行開始条件	任意	任意	任意
1805	実行開始時処理	そのまま	そのまま	そのまま
1806	リターン前	任意	任意	任意
1807	リターン時処理	そのまま	そのまま	そのまま

1808 -----

1810 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）
1811 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、
1812 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
1813 態が設定されるため、支障がないことを意味する。

1815 「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
1816 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
1817 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
1818 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。

1820 「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
1821 が用意されていないことを示す。

1823 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
1824 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
1825 の変更を許可している場合にのみ行われる【NGKI0529】。カーネルは、ディ
1826 スパッチ禁止フラグの元の状態をユーザスタック上に保存する【NGKI0530】。ア
1827 プリケーションがユーザスタック上に保存されたディスパッチ禁止フラグの状
1828 態を書き換えた場合、タスク例外処理ルーチンからのリターン時には、書き換
1829 えた後のディスパッチ禁止フラグの状態に変更される（すなわち、元に戻され
1830 るとは限らない）【NGKI0190】。

1832 また、保護機能対応カーネルでは、タスクにディスパッチ禁止フラグの変更を
1833 許可していない場合で、タスク例外処理ルーチン中で拡張サービスコールを用
1834 いてディスパッチ禁止フラグを変更した場合、カーネルは元の状態に戻さない
1835 【NGKI0191】。このことから、タスク例外処理ルーチンからの終了前に、ディ
1836 スパッチ禁止フラグを元の状態に戻すのは、アプリケーションの責任とする
1837 【NGKI0192】。

1839 **【補足説明】**

1841 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
1842 を実行中にマイグレーションされた場合、マイグレーション先のプロセッサにお
1843 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。

1845 **【仕様決定の理由】**

1847 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
1848 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
1849 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
1850 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

1851 を防止するためである。

1852

1853 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
1854 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
1855 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
1856 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
1857 めである。

1858

1859 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
1860 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
1861 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
1862 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,
1863 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
1864 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
1865 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
1866 に戻すこととしている。

1867

1868 2.6 タスクの状態遷移とスケジューリング規則

1869

1870 2.6.1 基本的なタスク状態

1871

1872 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
1873 いずれかの状態を取る【NGKI0193】。また、実行できる状態と広義の待ち状態
1874 を総称して、起動された状態と呼ぶ。さらに、タスクをカーネルに登録してい
1875 ない仮想的な状態を、未登録状態と呼ぶ。

1876

1877 (a) 実行できる状態 (runnable)

1878

1879 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
1880 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
1881 る。

1882

1883 (a.1) 実行状態 (running)

1884

1885 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
1886 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
1887 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。

1888

1889 (a.2) 実行可能状態 (ready)

1890

1891 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
1892 行状態にあるために、そのタスクが実行されない状態。

1893

1894 (b) 休止状態 (dormant)

1895

1896 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
1897 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
1898 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)
1899 は保存されていない【NGKI0194】。

1900

1901 (c) 広義の待ち状態 (blocked)

1902

1903 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
1904 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
1905 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
1906 状態に遷移する前の状態に戻される【NGKI0195】。広義の待ち状態は、さらに、
1907 （狭義の）待ち状態、強制待ち状態、二重待ち状態に分類される。

1908

1909 (c.1) (狭義の) 待ち状態 (waiting)

1910

1911 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。

1912

1913 (c.2) 強制待ち状態 (suspended)

1914

1915 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
1916 を強制待ち状態にすることも可能である。

1917

1918 (c.3) 二重待ち状態 (waiting-suspended)

1919

1920 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
1921 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
1922 実行を止められている状態。

1923

1924 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
1925 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
1926 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
1927 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
1928 重待ち状態でもないことを意味する。

1929

1930 (d) 未登録状態 (non-existent)

1931

1932 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、
1933 タスクは未登録状態にあるとみなす。

1934

1935 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
1936 合がある【NGKI0196】。過渡的な状態については、「2.6.6 ディスパッチ保留
1937 状態で実行中のタスクに対する強制待ち」の節を参照すること。

1938

1939 【TOPPERS/ASPカーネルにおける規定】

1940

1941 ASPカーネルでは、タスクが未登録状態になることはない【ASPS0005】。また、
1942 上記のタスク状態以外の過渡的な状態になることもない【ASPS0006】。ただし、
1943 動的生成機能拡張パッケージでは、タスクが未登録状態になる【ASPS0007】。

1944

1945 【TOPPERS/FMPカーネルにおける規定】

1946

1947 FMPカーネルでは、タスクが未登録状態になることはない【FMPS0003】。上記の
1948 タスク状態以外の過渡的な状態として、タスクが強制待ち状態〔実行継続中〕
1949 になることがある【FMPS0004】。詳しくは、「2.6.6 ディスパッチ保留状態で
1950 実行中のタスクに対する強制待ち」の節を参照すること。

1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、タスクが未登録状態になることはない【HRPS0002】。また、上記のタスク状態以外の過渡的な状態になることもない【HRPS0003】。ただし、動的生成機能拡張パッケージでは、タスクが未登録状態になる【HRPS0010】。

【TOPPERS/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【SSPS0003】。また、上記のタスク状態以外の過渡的な状態になることもない【SSPS0004】。

2.6.2 タスクの状態遷移

タスクの状態遷移を図2-2に示す【NGKI0197】。

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

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

実行できる状態になったタスクは、まずは実行可能状態に遷移するが、そのタスクの優先順位が実行状態のタスクよりも高い場合には、ディスパッチ保留状態でない限りはただちにディスパッチが起こり、実行状態へ遷移する

【NGKI0201】。この時、それまで実行状態であったタスクは実行可能状態に遷移する【NGKI0202】。この時、実行状態に遷移したタスクは、実行可能状態に遷移したタスクをプリエンプトしたという。逆に、実行可能状態に遷移したタスクは、プリエンプトされたという。

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

【補足説明】

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

2.6.3 タスクのスケジューリング規則

実行できるタスクは、優先順位の高いものから順に実行される【NGKI0203】。

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

タスクの優先順位は、タスクの優先度とタスクが実行できる状態になった順序から、次のように定まる。優先度の異なるタスクの間では、優先度の高いタスクが高い優先順位を持つ【NGKI0204】。優先度が同一のタスクの間では、先に実行できる状態になったタスクが高い優先順位を持つ【NGKI0205】。すなわち、同じ優先度を持つタスクは、FCFS (First Come First Served) 方式でスケジューリングされる。ただし、サービスコールの呼出しにより、同じ優先度を持つタスク間の優先順位を変更することも可能である【NGKI0206】。

最も高い優先順位を持つタスクが変化した場合には、ディスパッチ保留状態でない限りはただちにディスパッチが起こり、最も高い優先順位を持つタスクが実行状態となる【NGKI0207】。ディスパッチ保留状態においては、実行状態のタスクは切り換わらず、最も高い優先順位を持つタスクは実行可能状態にとどまる【NGKI0208】。

マルチプロセッサ対応カーネルでは、プロセッサ毎に、上記のスケジューリング規則を適用して、タスクスケジューリングを行う【NGKI0209】。すなわち、プロセッサがディスパッチ保留状態でない限りは、そのプロセッサに割り付けられた実行できるタスクの中で最も高い優先順位を持つタスクが実行状態となり、他は実行可能状態となる。そのため、実行状態のタスクは、プロセッサ毎に存在する。

2.6.4 待ち行列と待ち解除の順序

タスクが待ち解除される順序の管理のために、待ち状態のタスクがつながれているキューを、待ち行列と呼ぶ。また、タスクが同期・通信オブジェクトの待ち行列につながれている場合に、そのオブジェクトを、タスクの待ちオブジェクトと呼ぶ。

待ち行列にタスクをつなぐ順序には、FIFO順とタスクの優先度順がある。どちらの順序でつなぐかは、待ち行列毎に規定される【NGKI0210】。多くの待ち行列において、どちらの順序でつなぐかを、オブジェクト属性により指定できる【NGKI0211】。

FIFO順の待ち行列においては、新たに待ち状態に遷移したタスクは待ち行列の最後につながれる【NGKI0212】。それに対してタスクの優先度順の待ち行列においては、新たに待ち状態に遷移したタスクは、優先度の高い順に待ち行列につながれる【NGKI0213】。同じ優先度のタスクが待ち行列につながれている場合には、新たに待ち状態に遷移したタスクが、同じ優先度のタスクの中で最後につながれる【NGKI0214】。

待ち解除の条件がタスクによって異なる場合には、待ち行列の先頭のタスクは待ち解除の条件を満たさないが、後方のタスクが待ち解除の条件を満たす場合がある。このような場合の振舞いとして、次の2つのケースがある。どちらの振舞いをするかは、待ち行列毎に規定される【NGKI0215】。

(a) 待ち解除の条件を満たしたタスクの中で、待ち行列の前方につながれたものから順に待ち解除される【NGKI0216】。すなわち、待ち行列の前方に待ち解

2051 除の条件を満たさないタスクがあっても、後方のタスクが待ち解除の条件を満たして
2052 いたれば、先に待ち解除される。

2053

2054 (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

2055 【NGKI0217】．すなわち、待ち行列の前方に待ち解除の条件を満たさないタ
2056 スクがあると、後方のタスクが待ち解除の条件を満たしても、待ち解除されない。

2057

2058 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
2059 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）、
2060 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。
2061 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
2062 ち解除の条件を満たしていれば、ただちに待ち解除される【NGKI0218】．さら
2063 に、この待ち解除により新たに待ち行列の先頭になったタスクに対しても、同
2064 じ処理が繰り返される【NGKI0219】．

2065

2066 2. 6. 5 タスク例外処理マスク状態と待ち禁止状態

2067

2068 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
2069 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
2070 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
2071 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）、
2072 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
2073 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
2074 合を含む）は、タスク例外処理ルーチンの実行は開始されない【NGKI0220】．
2075 これらの状態を、タスク例外処理マスク状態と呼ぶ。

2076

2077 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
2078 して、待ち禁止状態になることができる【NGKI0221】．待ち禁止状態とは、タ
2079 スクが待ち状態に入ることが一時的に禁止された状態である．待ち禁止状態に
2080 あるタスクが、サービスコールを呼び出して待ち状態に遷移しようとした場合、
2081 サービスコールはE_RLWAIエラーとなる【NGKI0222】．

2082

2083 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
2084 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

2085 【NGKI0223】．その後、タスクがタスク例外処理マスク状態でなくなる時点
2086 （ユーザタスクにつ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
2087 拡張サービスコールからリターンする時点）で、待ち禁止状態が解除される

2088 【NGKI0224】．また、タスクの待ち禁止状態を解除するサービスコールによ
2089 っても、待ち禁止状態を解除することができる【NGKI0225】．

2090

2091 【仕様決定の理由】

2092

2093 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
2094 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
2095 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
2096 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
2097 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
2098 ールのための例外処理が行われないことになる。

2099

2100 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

2101 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
2102 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
2103 ムスタック上に不要な情報が残ってしまう。

2104
2105 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
2106 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
2107 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
2108 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
2109 る。

2110
2111 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、
2112 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、
2113 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、
2114 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
2115 外処理ルーチンを実行させることができる。

2116
2117 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
2118 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
2119 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
2120 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
2121 されないことになる。

2122
2123 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
2124 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`/
2125 `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
2126 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
2127 ンを実行させることができる。

2128
2129 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
2130 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
2131 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。

2132
2133 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
2134 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
2135 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
2136 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
2137 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
2138 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
2139 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。

2140
2141 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
2142 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
2143 るように実装すべきである。

2144
2145 【 μ ITRON4.0仕様、 μ ITRON4.0/PX仕様との関係】

2146
2147 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
2148 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
2149 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
2150 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

2151 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。
2152
2153 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
2154 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
2155 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
2156 は互換性がない。
2157
2158 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
2159
2160 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
2161 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
2162 チ保留状態が解除されるまで保留される【NGKI0226】。
2163
2164 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
2165 過渡的な状態にあると考える【NGKI0227】。この状態を、強制待ち状態〔実行
2166 継続中〕と呼ぶ。一方、ディスパッチ保留状態が解除された後に実行すべきタ
2167 スクは、実行可能状態にとどまる【NGKI0228】。
2168
2169 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
2170 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
2171 る【NGKI0229】。
2172
2173 過渡的な状態も含めたタスクの状態遷移を図2-3に示す【NGKI0230】。
2174
2175 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。
2176
2177 (a) プロセッサを占有して実行を継続する。
2178
2179 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
2180 続して実行される【NGKI0231】。
2181
2182 (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
2183 であるものと扱う。
2184
2185 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/`
2186 `iget_tid`, `get_did`, `sns_tex`) では、強制待ち状態〔実行継続中〕のタスクが、
2187 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
2188 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/`
2189 `iget_tid`を発行すると、そのタスクのID番号を参照する【NGKI0232】。また、
2190 `get_did`を発行するとそのタスクが属する保護ドメインのID番号を、`sns_tex`を
2191 発行するとそのタスクのタスク例外処理禁止フラグを参照する【NGKI0233】。
2192
2193 (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。
2194
2195 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
2196 待ち状態であるものと扱う【NGKI0234】。
2197
2198 【TOPPERS/ASPカーネルにおける規定】
2199
2200 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

2201 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
2202 ち状態 [実行継続中] になることはない【ASPS0008】。

2203

2204 【TOPPERS/FMPカーネルにおける規定】

2205

2206 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
2207 ち状態へ遷移させるサービスコールを、他のプロセッサから呼び出すことができ
2208 るため、タスクが強制待ち状態 [実行継続中] になる場合がある【FMPS0005】。

2209

2210 【TOPPERS/HRP2カーネルにおける規定】

2211

2212 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
2213 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
2214 待ち状態 [実行継続中] になることはない【HRPS0004】。

2215

2216 【TOPPERS/SSPカーネルにおける規定】

2217

2218 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
2219 制待ち状態 [実行継続中] になることもない【SSPS0005】。

2220

2221 【補足説明】

2222

2223 この仕様では、ディスパッチ保留状態において、実行状態のタスクを強制終
2224 らせるサービスコールはサポートしていない。そのため、実行状態と休止状態
2225 の間の過渡的な状態は存在しない。

2226

2227 2.6.7 制約タスク

2228

2229 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
2230 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
2231 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
2232 は以下の機能制限がある。

2233

2234 (a) 広義の待ち状態に入ることができない【NGKI0235】。

2235

2236 (b) サービスコールによりベース優先度を変更することができない【NGKI0236】。

2237

2238 (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
2239 先順位の回転 (rot_rdq/irot_rdq) を行うことができない【NGKI0237】。

2240

2241 (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
2242 できない【NGKI0238】。

2243

2244 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
2245 び出した場合には、E_NOSPTエラーとなる【NGKI0239】。E_NOSPTエラーが返る
2246 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
2247 ことができる【NGKI0240】。

2248

2249 【未決定事項】

2250

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

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、制約タスクをサポートしていない【ASPS0009】。ただし、制約タスク拡張パッケージを用いると、制約タスクの機能を追加することができる【ASPS0010】。

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、制約タスクをサポートしていない【FMPS0006】。

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、制約タスクをサポートしていない【HRPS0005】。

【TOPPERS/SSPカーネルにおける規定】

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

【 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【NGKI0241】。この図は、割込み処理モデルの持つすべての機能が、ハードウェア（プロセッサおよび割込みコントローラ）で実現されているとして描いた概念図である。実際のハードウェアで不足している機能については、カーネル内の割込み処理のソフトウェアで実現される。

【 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。

2301 2.7.1 割込み処理の流れ

2302
2303 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、
2304
2305 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。
2306

2307
2308 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、
2309 割込み要求を受け付ける【NGKI0242】。受け付けた割込み要求が、カーネル管理の割込みである場合には、カーネル内の割込みハンドラの入口処理（割込み入口処理）を経由して、カーネル内の割込みハンドラを実行する【NGKI0243】。
2310
2311

2312
2313 カーネル内の割込みハンドラは、アプリケーションが割込み要求ラインに対して登録した割込みサービスルーチン（ISR）を呼び出す【NGKI0244】。割込みサービスルーチンは、プロセッサの割込みアーキテクチャや割込みコントローラに依存せず、割込みを要求したデバイスのみに依存して記述するのが原則である
2314
2315 【NGKI0245】。1つの割込み要求ラインに対して複数のデバイスが接続されることから、1つの割込み要求ラインに対して複数の割込みサービスルーチンを登録
2316
2317 することができる【NGKI0246】。
2318
2319

2320
2321 ただし、カーネルが標準的に用意している割込みハンドラで対応できない特殊なケースも考えられる。このような場合に対応するために、アプリケーション
2322
2323 が用意した割込みハンドラをカーネルに登録することもできる【NGKI0247】。
2324

2325
2326 カーネルが用いるタイマデバイスからの割込み要求の場合、カーネル内の割込みハンドラにより、タイムイベントの処理が行われる。具体的には、タイムアウト処理等が行われることに加えて、アプリケーションが登録したタイムイベントハンドラが呼び出される【NGKI0248】。
2327
2328

2329
2330 なお、受け付けた割込み要求に対して、割込みサービスルーチンも割込みハンドラも登録していない場合の振舞いは、ターゲット定義である【NGKI0249】。
2331
2332

2333 2.7.2 割込み優先度

2334
2335 割込み要求は、割込み処理の優先順位を指定するための割込み優先度を持つ【NGKI0250】。プロセッサは、割込み優先度マスクの現在値よりも高い割込み優先度を持つ割込み要求のみを受け付ける【NGKI0251】。逆に言うと、割込み優先度マスクの現在値と同じか、それより低い割込み優先度を持つ割込みは、
2336
2337 マスクされる。
2338
2339

2340
2341 プロセッサは、割込み要求を受け付けると、割込み優先度マスクを、受け付けた割込み要求の割込み優先度に設定する（ただし、受け付けた割込みがNMIである場合には例外とする）【NGKI0252】。また、割込み処理からのリターンにより、割込み優先度マスクを、割込み要求を受け付ける前の値に戻す
2342
2343 【NGKI0253】。
2344
2345

2346
2347 これらのことから、他の方法で割込みをマスクしていない限り、ある割込み要求の処理中は、それと同じかそれより低い割込み優先度を持つ割込み要求は受け付けられず、それより高い割込み優先度を持つ割込み要求は受け付けられることになる。つまり、割込み優先度は、多重割込みを制御するためのものと位
2348
2349
2350

2351 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、
2352 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
2353 【NGKI0254】。

2354
2355 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、
2356 【NGKI0037】の原則には従わず、-1から連続した負の値を用いる【NGKI0255】。

2357
2358 割込み優先度の段階数は、ターゲット定義である【NGKI0256】。プロセッサが
2359 割込み優先度マスクを実現するための機能を持たないか、実現するために大き
2360 いオーバーヘッドを生じる場合には、ターゲット定義で、割込み優先度の段階数
2361 を1にする（すなわち、多重割込みを許さない）場合がある。

2362 【仕様決定の理由】

2363
2364
2365 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
2366 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
2367 先度マスクの値を0にできるためである。

2368 2369 2.7.3 割込み要求ラインの属性

2370
2371 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
2372 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
2373 同一の属性を持つ【NGKI0257】。それらの割込み要求に別々の属性を設定する
2374 ことはできない。

2375 2376 (1) 割込み要求禁止フラグ

2377
2378 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
2379 つ【NGKI0258】。割込み要求禁止フラグをセットすると、その割込み要求ライ
2380 ンによって伝えられる割込み要求はマスクされる【NGKI0259】。

2381
2382 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
2383 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
2384 要求禁止フラグをサポートしない場合がある【NGKI0260】。また、プロセッサ
2385 の持つ割込み要求禁止フラグの機能がこの仕様に合致しない場合には、ターゲッ
2386 ト定義で、割込み要求禁止フラグをサポートしないか、振舞いが異なるものと
2387 する場合がある【NGKI0261】。

2388 2389 (2) 割込み優先度

2390
2391 割込み要求ライン毎に、割込み優先度を設定することができる【NGKI0262】。
2392 割込み要求の割込み優先度とは、その割込み要求を伝える割込み要求ラインに
2393 対して設定された割込み優先度のことである【NGKI0263】。

2394 2395 (3) トリガモード

2396
2397 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
2398 であるかを設定することができる【NGKI0264】。エッジトリガの場合には、さ
2399 らに、ターゲット定義で、ポジティブエッジトリガかネガティブエッジトリガ
2400 か両エッジトリガかを設定できる場合もある【NGKI0265】。また、レベルトリ

ガの場合には、ターゲット定義で、ローレベルトリガかハイレベルトリガかを設定できる場合もある【NGKI0266】。

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

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

【使用上の注意】

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

複数のデバイスからの割込み要求が接続されている割込み要求ラインを、エッジトリガに設定することは推奨されない。これは、次のような状況において、割込み要求を取りこぼす可能性があるためである。ある割込み要求ラインに、デバイスAとデバイスBからの割込み要求が接続されており、デバイスAの割込み処理を先に行う場合を考える。この時、デバイスBからの割込み要求によって割込みハンドラが実行され、デバイスAの割込み処理を行った後、デバイスBの割込み処理を行う前に、デバイスAからの割込み要求が発生した場合に、デバイスAからの割込み要求を取りこぼしてしまう。

2.7.4 割込みを受け付ける条件

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

(a) 割込み要求ラインに対する割込み要求禁止フラグがクリアされていること

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

(c) 全割込みロックフラグがクリアされていること

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

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

2.7.5 割込み番号と割込みハンドラ番号

割込み要求ラインを識別するための番号を、割込み番号と呼ぶ。割込み番号は、符号無しの整数型であるINTNO型で表し、ターゲットハードウェアの仕様から決まる自然な番号付けを基本として、ターゲット定義で付与される【NGKI0272】。そのため、1から連続した正の値であるとは限らない。

それに対して、アプリケーションが用意した割込みハンドラをカーネルに登録する場合に、割込みハンドラの登録対象となる割込みを識別するための番号を、割込みハンドラ番号と呼ぶ。割込みハンドラ番号は、符号無しの整数型であるINHNO型で表し、ターゲットハードウェアの仕様から決まる自然な番号付けを基本として、ターゲット定義で付与される【NGKI0273】。そのため、1から連続した正の値であるとは限らない。

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

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

2.7.6 マルチプロセッサにおける割込み処理

この節では、マルチプロセッサにおける割込み処理について説明する。この節の内容は、マルチプロセッサ対応カーネルにのみ適用される。

マルチプロセッサ対応カーネルでは、TOPPERS標準割込み処理モデルの構成要素の中で、図2-4の破線に囲まれた部分はプロセッサ毎に持ち、それ以外の部分はシステム全体で1つのみ持つ【NGKI0278】。すなわち、全割込みロックフラグ、CPUロックフラグ、割込み優先度マスクはプロセッサ毎に持つのに対して、割込み要求ラインおよびその属性（割込み要求禁止フラグ、割込み優先度、トリガモード）はシステム全体で共通に持つ。

割込み番号は、割込み要求ラインを識別するための番号であることから、割込み要求ラインが複数のプロセッサに接続されている場合でも、1つの割込み要求ラインには1つの割込み番号を付与する【NGKI0279】。逆に、複数のプロセッサが同じ種類のデバイスを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）【NGKI0280】。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、

2501 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
2502 のとする【NGKI0281】。

2503

2504 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
2505 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
2506 る【NGKI0282】。ただし、その内のどのプロセッサで割込みサービスルーチン
2507 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
2508 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）。

2509

2510 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
2511 の割込み要求ラインが接続されたプロセッサの集合に含まれていなければならない【NGKI0283】。また、同一の割込み要求ラインに対して登録する割込みサー
2512 ビスルーチンは、同一のクラスに属していなければならない【NGKI0284】。

2513

2514

2515 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
2516 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
2517 ハンドラ番号を付与する（図2-5）【NGKI0285】。割込みハンドラが属するクラ
2518 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
2519 なければならない【NGKI0286】。

2520

2521 **【補足説明】**

2522

2523 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
2524 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
2525 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
2526 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
2527 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
2528 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
2529 とする（図2-5）。

2530

2531 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
2532 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
2533 スに属していなければならない。

2534

2535 **【使用上の注意】**

2536

2537 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
2538 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
2539 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。

2540

2541 2.7.7 カーネル管理外の割込み

2542

2543 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
2544 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
2545 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、
2546 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
2547 としている【NGKI0287】。TMIN_INTPRIを固定するか設定できるようにするか、
2548 設定できるようにする場合の設定方法は、ターゲット定義である【NGKI0288】。

2549

2550 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

2551 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
2552 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、
2553 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
2554 るか（設けられるようにするか）どうかは、ターゲット定義である【NGKI0289】。

2555
2556 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
2557 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
2558 ンドラをカーネル管理の割込みハンドラと呼ぶ。

2559
2560 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
2561 に実行するのが基本である【NGKI0290】。ただし、すべての割込みで同じ番地
2562 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
2563 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
2564 由してカーネル管理外の割込みハンドラが実行されることになる【NGKI0291】。

2565
2566 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
2567 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
2568 は、ターゲット定義である【NGKI0292】。カーネル管理外の割込みハンドラか
2569 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
2570 ことができ、その他のサービスコールを呼び出すことはできない【NGKI0293】。
2571 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
2572 場合の動作は、保証されない【NGKI0294】。

2573 2574 2.7.8 カーネル管理外の割込みの設定方法 2575

2576 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
2577 ずれかが採用される【NGKI0295】。

- 2578
2579 (a-1) NMI以外にカーネル管理外の割込みを設けない
2580 (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
2581

2582 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
2583 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
2584 する割込みをアプリケーション側で設定する必要はない【NGKI0296】。ここで、
2585 カーネル管理外とされた割込みに対して、カーネルのAPIにより割込みハンドラ
2586 を登録できるかと、割込み要求ラインの属性を設定できるかは、ターゲット定
2587 義である【NGKI0297】。割込みハンドラを登録できる場合には、それを定義す
2588 るAPIにおいて、カーネル管理外であることを示す割込みハンドラ属性
2589 (TA_NONKERNEL)を指定する【NGKI0298】。また、割込み要求ラインの属性を
2590 設定できる場合には、設定する割込み優先度をTMIN_INTPRIよりも高い値とする
2591 【NGKI0299】。

- 2592
2593 (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
2594

2595 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、
2596 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
2597 であることを示す割込みハンドラ属性 (TA_NONKERNEL)を指定する
2598 【NGKI0300】。また、カーネル管理外とする割込みの割込み要求ラインに対し
2599 て設定する割込み優先度を、TMIN_INTPRIよりも高い値とする【NGKI0301】。
2600

2601 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
2602 る割込み優先度は、TMIN_INTPRIより高い値であってはならない【NGKI0302】。
2603 また、カーネル管理外の割込みに対して、割込みサービスルーチンを登録する
2604 ことはできない【NGKI0303】。

2605

2606 2.8 CPU例外処理モデル

2607

2608 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
2609 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
2610 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
2611 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、
2612 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。

2613

2614 2.8.1 CPU例外処理の流れ

2615

2616 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
2617 ドラを登録することができる【NGKI0304】。プロセッサがCPU例外の発生を検出
2618 すると、カーネル内のCPU例外ハンドラの入口処理（CPU例外入口処理）を経由
2619 して、発生したCPU例外に対して登録したCPU例外ハンドラが呼び出される
2620 【NGKI0305】。

2621

2622 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
2623 ンドラ番号と呼ぶ。CPU例外ハンドラ番号は、符号無しの整数型であるEXCNO型
2624 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
2625 て、ターゲット定義で付与される【NGKI0306】。そのため、1から連続した正の
2626 値であるとは限らない。

2627

2628 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、
2629 異なるCPU例外であると扱う【NGKI0307】。すなわち、同じ種類のCPU例外であっ
2630 ても、異なるプロセッサのCPU例外には異なるCPU例外ハンドラ番号を付与し、
2631 プロセッサ毎にCPU例外ハンドラを登録する。CPU例外ハンドラが属するクラス
2632 の初期割付けプロセッサは、CPU例外が発生するプロセッサと一致していなければ
2633 ならない【NGKI0308】。

2634

2635 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
2636 う【NGKI0309】。どのようなリカバリ処理を行うかは、一般にはCPU例外の種類
2637 やそれが発生したコンテキストおよび状態に依存するが、大きく次の4つの方法
2638 が考えられる【NGKI0310】。

2639

2640 (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。

2641

2642 (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
2643 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
2644 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
2645 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
2646 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
2647 させることで、リカバリ処理を行える可能性があるが、CPU例外を起こしたタス
2648 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）

2649 【NGKI0311】。

2650

2651 (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
2652 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）。

2653

2654 (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
2655 る）。

2656

2657 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
2658 生したコンテキストおよび状態に依存せずに常に行える【NGKI0312】。それ
2659 に対して(b)と(c)の方法は、CPU例外ハンドラからそのためのサービスコールを呼
2660 び出せることが必要であり、それが行えるかどうかは、CPU例外が発生したコン
2661 テキストおよび状態に依存する【NGKI0313】。

2662

2663 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
2664 舞いは、ターゲット定義である【NGKI0314】。

2665

2666 **【使用上の注意】**

2667

2668 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
2669 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
2670 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
2671 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
2672 きない場合には、CPU例外入口処理を経由せずに、アプリケーションが用意した
2673 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。

2674

2675 **【補足説明】**

2676

2677 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、
2678 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
2679 番号を含める方法を基本とする。

2680

2681 **【 μ ITRON4.0仕様との関係】**

2682

2683 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
2684 ていない。

2685

2686 2.8.2 CPU例外ハンドラから呼び出せるサービスコール

2687

2688 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
2689 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
2690 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
2691 ことができる【NGKI0315】。

2692

2693 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
2694 チできる状態であった場合にfalseを返す【NGKI0316】。xsns_dpnがfalseを返
2695 した場合、そのCPU例外ハンドラから、非タスクコンテキストから呼び出せるす
2696 べてのサービスコールを呼び出すことができ、(b)の方法によるリカバリ処理が
2697 可能である【NGKI0317】。ただし、CPU例外を起こしたタスクが最高優先度の場
2698 合には、この方法でリカバリ処理を行うことはできない【NGKI0318】。

2699

2700 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

2701 処理ルーチンを実行できる状態であった場合にfalseを返す【NGKI0319】。
2702 xsns_xpnがfalse を返した場合、そのCPU例外ハンドラから、非タスクコンテキ
2703 ストから呼び出せるすべてのサービスコールを呼び出すことができ、(c)の方法
2704 によるリカバリ処理が可能である【NGKI0320】。

2705
2706 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
2707 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
2708 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
2709 ルを呼び出すことはできない【NGKI0321】。いずれのサービスコールもtrueを
2710 返したにもかかわらず、その他のサービスコールを呼び出した場合の動作は、
2711 保証されない【NGKI0322】。この場合には、(b)と(c)の方法によるリカバリ処
2712 理は行うことはできず、(a)または(d)の方法によるリカバリ処理を行うしか
2713 ないことになる。

2714
2715 【μ ITRON4.0仕様との関係】
2716

2717 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
2718 修正した。

2719
2720 2.8.3 エミュレートされたCPU例外ハンドラ
2721

2722 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
2723 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
2724 す場合がある【NGKI0323】。この場合に、カーネルが検出するエラーをCPU例外
2725 と同等に扱うものとし、エミュレートされたCPU例外と呼ぶ【NGKI0324】。また、
2726 エラー処理のためのプログラムをCPU例外ハンドラと同等に扱うものとし、エミュ
2727 レートされたCPU例外ハンドラと呼ぶ【NGKI0325】。

2728
2729 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
2730 番号が付与され、CPU例外ハンドラと同じ方法で登録できる【NGKI0326】。また、
2731 エミュレートされたCPU例外ハンドラからも、CPU例外ハンドラから呼び出せる
2732 サービスコールを呼び出すことができ、CPU例外ハンドラと同様のリカバリ処理
2733 を行うことができる【NGKI0327】。

2734
2735 【μ ITRON4.0仕様との関係】
2736

2737 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
2738 されていない概念である。

2739
2740 2.8.4 カーネル管理外のCPU例外
2741

2742 カーネル非動作状態、カーネル内のクリティカルセクションの実行中、全割込
2743 みロック状態、CPUロック状態、カーネル管理外の割込みハンドラ実行中のい
2744 れかで発生したCPU例外を、カーネル管理外のCPU例外と呼ぶ。また、それによ
2745 って起動されるCPU例外ハンドラを、カーネル管理外のCPU例外ハンドラと呼ぶ。
2746 さらに、カーネル管理外のCPU例外ハンドラ実行中に発生したCPU例外も、カー
2747 ネル管理外のCPU例外とする。

2748
2749 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
2750 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

2751 理のCPU例外ハンドラと呼ぶ。

2752

2753 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
2754 サービスコールもtrueを返す【NGKI0330】。そのため、「2.8.2 CPU例外ハンド
2755 ラから呼び出せるサービスコール」の節で述べた制限【NGKI0321】【NGKI0322】
2756 が課される。

2757

2758 【補足説明】

2759

2760 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
2761 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
2762 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。

2763

2764 2.9 システムの初期化と終了

2765

2766 2.9.1 システム初期化手順

2767

2768 システムのリセット後、最初に実行するプログラムを、スタートアップモジュー
2769 ルと呼ぶ。スタートアップモジュールはカーネルの管理外であり、アプリケー
2770 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
2771 理を明確にするために、カーネルの配布パッケージの中に、標準のスタートアップ
2772 モジュールが用意されている【NGKI0331】。

2773

2774 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
2775 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
2776 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
2777 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
2778 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
2779 の呼出しを行った後、カーネルの初期化処理へ分岐する【NGKI0332】。ここで
2780 呼び出すターゲットシステム依存の初期化フックでは、リセット後に速やかに
2781 行うべき初期化処理を行うことが想定されている。

2782

2783 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
2784 ジュールを実行し、カーネルの初期化処理へ分岐する【NGKI0333】。ただし、
2785 共有リソースの初期化処理（非初期化データセクションのクリア、初期化デー
2786 タセクションの初期化、ソフトウェア環境依存の初期化フックの呼出しなど）
2787 は、マスタプロセッサのみで実行する【NGKI0334】。各プロセッサがカーネル
2788 の初期化処理へ分岐するのは、共有リソースの初期化処理が完了した後でなけ
2789 ればならないため、スレーブプロセッサは、カーネルの初期化処理へ分岐する
2790 前に、マスタプロセッサによる共有リソースの初期化処理の完了を待ち合わせ
2791 する必要がある【NGKI0335】。

2792

2793 カーネルの初期化処理においては、まず、カーネル自身の初期化処理（カーネ
2794 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
2795 APIの処理（オブジェクトの登録など）が行われる【NGKI0336】。静的APIのパ
2796 ラメータに関するエラーは、コンフィギュレータによって検出されるのが原則
2797 であるが、コンフィギュレータで検出できないエラーが、この処理中に検出さ
2798 れる場合もある【NGKI0337】。

2799

2800 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

2801 システムコンフィギュレーションファイルにおける静的APIの記述順と同じ順序で
2802 静的APIが処理された場合と、同じ振舞いとなる【NGKI0338】。例えば、静的
2803 APIによって同じ優先度のタスクを複数生成・起動した場合、静的APIの記述順
2804 が先のタスクが高い優先順位を持つ。それに対して、周期ハンドラの動作開始
2805 順序は、同じタイムティックで行うべき処理が複数ある場合の処理順序が規定
2806 されないことから（「4.6.1 システム時刻管理」の節を参照）、静的APIの記述
2807 順となるとは限らない。

2808
2809 次に、静的API（ATT_INI）により登録した初期化ルーチンが、システムコンフィ
2810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
2811【NGKI0339】。

2812
2813 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
2814 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
2815 化ルーチンを実行する【NGKI0340】。グローバル初期化ルーチンの実行が完了
2816 した後に、各プロセッサは、自プロセッサに割り付けられたローカル初期化ルー
2817 チンを実行する【NGKI0341】。すなわち、ローカル初期化ルーチンは、初期割
2818 付けプロセッサにより実行される。

2819
2820 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
2821 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
2822【NGKI0342】。具体的には、システム状態が、全割込みロック解除状態・CPUロッ
2823 ク解除状態・割込み優先度マスク全解除状態・ディスパッチ許可状態に設定さ
2824 れ（すなわち、割込みがマスク解除され）、タスクの実行が開始される。

2825
2826 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
2827 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
2828 ネルの動作が開始される【NGKI0343】。マルチプロセッサ対応カーネルにおけ
2829 るシステム初期化の流れと、各プロセッサが同期を取るタイミングを、図2-6に
2830 示す【NGKI0344】。

2831
2832 【μITRON4.0仕様との関係】

2833
2834 μITRON4.0仕様においては、初期化ルーチンの実行は静的APIの処理に含まれる
2835 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、
2836 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
2837 いものとした。

2838
2839 2.9.2 システム終了手順

2840
2841 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
2842 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
2843 節を参照）【NGKI0345】。具体的には、NMIを除くすべての割込みがマスクされ、
2844 タスクの実行が停止される。

2845
2846 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
2847 （ext_ker）は、どのプロセッサからでも呼び出すことができる【NGKI0346】。
2848 1つのプロセッサでカーネルを終了させるサービスコールを呼び出すと、そのプ
2849 ロセッサがカーネル動作状態から非動作状態に遷移した後、他のプロセッサに
2850 対してカーネル終了処理の開始を要求する【NGKI0347】。複数のプロセッサが

2851 ら、カーネルを終了させるサービスコール (ext_ker) を呼び出してもよい
2852 【NGKI0348】。

2853

2854 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
2855 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
2856 【NGKI0349】。

2857

2858 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
2859 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
2860 終了処理ルーチンを実行する【NGKI0350】。すなわち、ローカル終了処理ルー
2861 チンは、初期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
2862 カル終了処理ルーチンの実行が完了した後に、マスタプロセッサがグローバル
2863 終了処理ルーチンを実行する【NGKI0351】。

2864

2865 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
2866 【NGKI0352】。ターゲットシステム依存の終了処理は、カーネルの管理外であ
2867 り、アプリケーションで用意するのが基本であるが、カーネルの配布パッケ
2868 ジの中に、ターゲットシステム毎に標準的なルーチンが用意されている
2869 【NGKI0353】。標準のターゲットシステム依存の終了処理では、ソフトウェア
2870 環境 (ライブラリなど) 依存の終了処理フックを呼び出す【NGKI0354】。

2871

2872 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
2873 テム依存の終了処理が呼び出される【NGKI0355】。マルチプロセッサ対応カー
2874 ネルにおけるシステム終了処理の流れと、各プロセッサが同期を取るタイミン
2875 グを、図2-7に示す【NGKI0356】。

2876

2877 【使用上の注意】

2878

2879 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
2880 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
2881 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。

2882

2883 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
2884 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
2885 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
2886 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
2887 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
2888 ネルを終了させることができる。

2889

2890 【μ ITRON4.0仕様との関係】

2891

2892 μ ITRON4.0仕様には、システム終了に関する規定はない。

2893

2894 2.10 オブジェクトの登録とその解除

2895

2896 2.10.1 ID番号で識別するオブジェクト

2897

2898 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
2899 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
2900 る静的API (ATT_YYY, ATA_YYY) によってカーネルに登録する【NGKI0357】。オ

2901 オブジェクトを追加する静的APIによって登録されたオブジェクトはID番号を持た
2902 ないため、ID番号を指定して操作することができない【NGKI0358】。

2903

2904 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
2905 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
2906 けたID番号にマクロ定義する【NGKI0359】。同じ識別名のオブジェクトが生成
2907 済みの場合には、E_OBJエラーとなる【NGKI0360】。

2908

2909 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
2910 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
2911 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
2912 【NGKI0361】。割り付けたID番号は、サービスコールの返値としてアプリケー
2913 ションに通知する【NGKI0362】。使用されていないID番号が残っていない場合
2914 には、E_NOIDエラーとなる【NGKI0363】。

2915

2916 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
2917 ギュレーションファイル中に複数記述することができる【NGKI0364】。その場
2918 合、各静的APIで指定した数の合計の数のID番号が確保される【NGKI0365】。

2919

2920 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
2921 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
2922 除することができる【NGKI0366】。登録解除したオブジェクトのID番号は、未
2923 使用の状態に戻され、そのID番号を用いて新しいオブジェクトを登録すること
2924 ができる【NGKI0367】。この場合に、登録解除前のオブジェクトに対して行う
2925 つものの操作が、新たに登録したオブジェクトに対して行われないように、注
2926 意が必要である。

2927

2928 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、
2929 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
2930 となる【NGKI0369】。

2931

2932 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
2933 ることができる【NGKI0370】。この場合、登録解除された処理単位に実行が強
2934 制的に終了させられることはなく、処理単位が自ら実行を終了するまで、処理
2935 単位の実行は継続される【NGKI0371】。

2936

2937 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
2938 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
2939 ビスコールはE_DLTエラーとなる【NGKI0372】。複数のタスクが待ち解除される
2940 場合には、待ち行列につながれていた順序で待ち解除される【NGKI0373】。削
2941 除した同期・通信オブジェクトが複数の待ち行列を持つ場合には、別の待ち行
2942 列で待っていたタスクの間の待ち解除の順序は、該当するサービスコール毎に
2943 規定する【NGKI0374】。

2944

2945 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
2946 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
2947 【NGKI0375】。ただし、オブジェクトを生成または追加する静的APIによって登
2948 録したオブジェクトも、再初期化することができる【NGKI0376】。

2949

2950 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

2951 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
2952 ブジェクトのアクセス許可ベクタを設定するサービスコール (sac_yyy) , オブ
2953 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
2954 【NGKI0377】 .

2955

2956 【μ ITRON4.0仕様との関係】

2957

2958 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
2959 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
2960 クトは, 登録解除できないこととした.

2961

2962 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)
2963 は規定されていない.

2964

2965 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
2966 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
2967 されている.

2968

2969 【μ ITRON4.0/PX仕様との関係】

2970

2971 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
2972 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
2973 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
2974 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,
2975 ata_yyy) も廃止した.

2976

2977 【仕様決定の理由】

2978

2979 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
2980 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,
2981 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.
2982 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.

2983

2984 2.10.2 オブジェクト番号で識別するオブジェクト

2985

2986 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
2987 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
2988 【NGKI0378】 .

2989

2990 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
2991 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
2992 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
2993 【NGKI0379】 . 登録解除したオブジェクト番号は, オブジェクト登録前の状態
2994 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
2995 ができる【NGKI0380】 . 登録解除されていないオブジェクト番号に対して再度
2996 オブジェクトを登録しようとした場合には, E_OBJエラーとなる【NGKI0381】 .

2997

2998 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
2999 することができない【NGKI0382】 . 登録を解除しようとした場合には, E_OBJエ
3000 ラーとなる【NGKI0383】 .

3001
3002 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
3003 ル（def_yyy）はサポートされない【NGKI0384】。
3004
3005 【μ ITRON4.0仕様との関係】
3006
3007 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
3008 後に、新たにオブジェクトを定義する必要がある。また、オブジェクトを定義
3009 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
3010 いこととした。
3011
3012 2.10.3 識別番号を持たないオブジェクト
3013
3014 識別する必要がないために、識別番号を持たないオブジェクトは、オブジェク
3015 トを追加する静的API（ATT_YYY）によってカーネルに登録する。
3016
3017 2.10.4 オブジェクト生成に必要なメモリ領域
3018
3019 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
3020 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
3021 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
3022 【NGKI0385】。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
3023 フィギュレータまたはカーネルにより確保される【NGKI0386】。
3024
3025 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、
3026 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
3027 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。
3028
3029 ・データキュー管理領域
3030 ・優先度データキュー管理領域
3031 ・優先度別のメッセージキューヘッダ領域
3032 ・固定長メモリプール管理領域
3033
3034 【補足説明】
3035
3036 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
3037 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
3038 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
3039 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。
3040
3041 2.10.5 オブジェクトが属する保護ドメインの設定
3042
3043 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
3044 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
3045 【NGKI0387】。
3046
3047 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、オブジェクトに登
3048 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
3049 する【NGKI0388】。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
3050 の囲みの外に記述する（「2.12.3 保護ドメインの指定」の節を参照）

3051 【NGKI0389】.

3052

3053 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
3054 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
3055 ドメインを設定する【NGKI0390】. ここでdomidは、そのオブジェクトを属させ
3056 る保護ドメインのID番号であり、TDOM_KERNEL(=-1)を指定することでカーネ
3057 ルドメインに属させることができる. また、domidにTDOM_SELF(=0)を指定す
3058 るか、オブジェクト属性にTA_DOM(domid)を指定しないことで、自タスクが属す
3059 る保護ドメインに属させることができる. さらに、無所属のオブジェクトに登
3060 録する場合には、domidにTDOM_NONE(=-2)を指定する.

3061

3062 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
3063 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
3064 オブジェクト属性で設定する必要がないものもある【NGKI0391】.

3065

3066 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、
3067 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
3068 も)割り付けられる【NGKI0392】. これらの静的APIは、保護ドメインの囲みの
3069 外に記述しなければならない. 保護ドメインの囲みの中に記述した場合には、
3070 E_RSATRエラーとなる【NGKI0394】.

3071

3072 【補足説明】

3073

3074 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
3075 用意していない.

3076

3077 【仕様決定の理由】

3078

3079 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
3080 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
3081 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
3082 タを増やさない方が望ましいためである.

3083

3084 2. 10. 6 オブジェクトが属するクラスの設定

3085

3086 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
3087 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

3088 【NGKI0395】.

3089

3090 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、オブジェクトに登
3091 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

3092 【NGKI0396】. クラスに属さないオブジェクトを登録する静的APIは、クラスの
3093 囲みの外に記述する(「2. 12. 4 クラスの指定」の節を参照)【NGKI0397】.

3094

3095 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
3096 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
3097 スを設定する【NGKI0398】. ここでclsidは、そのオブジェクトを属させるクラ
3098 スのID番号であり、clsidにTCLS_SELF(=0)を指定するか、オブジェクト属性
3099 にTA_CLS(clsid)を指定しないことで、自タスクが属するクラスに属させること
3100 ができる.

3101
3102 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、
3103 静的APIを囲むクラスに属するオブジェクトにのみ割り付けられる【NGKI0399】。
3104 これらの静的APIは、確保したID番号を割り付けるオブジェクトの属すべきクラ
3105 スの囲みの中に記述しなければならない。クラスの囲みの外に記述した場合に
3106 は、E_RSATRエラーとなる【NGKI0401】。
3107
3108 【補足説明】
3109
3110 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
3111 ていない。
3112
3113 【仕様決定の理由】
3114
3115 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
3116 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
3117 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
3118 タを増やさない方が望ましいためである。
3119
3120 2. 10. 7 オブジェクトの状態参照
3121
3122 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
3123 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
3124 (ref_yyy, get_yyy) を用意する【NGKI0402】。
3125
3126 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
3127 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）
3128 を参照するための機能は用意しないことを原則とする【NGKI0403】。自タスク
3129 の拡張情報の参照するサービスコール (get_inf) は、この原則に対する例外で
3130 ある【NGKI0404】。
3131
3132 2. 11 オブジェクトのアクセス保護
3133
3134 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
3135 内容は、保護機能対応カーネルにのみ適用される。
3136
3137 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
3138
3139 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
3140 れたアクセス許可ベクタによって保護される【NGKI0405】。ただし、アクセス
3141 許可ベクタを持たないオブジェクトに対するアクセスは、システム状態に対す
3142 るアクセス許可ベクタによって保護される【NGKI0406】。また、オブジェクト
3143 を登録するサービスコールと、特定のオブジェクトに関連しないシステムの状
3144 態に対するアクセスについては、システム状態のアクセス許可ベクタによって
3145 保護される【NGKI0407】。
3146
3147 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、
3148 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。
3149
3150 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

3151 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
3152 E_OACVエラーが返る【NGKI0408】。また、メモリオブジェクトに対して、許可
3153 されていない管理操作または参照操作を行おうとした場合も、サービスコール
3154 からE_OACVエラーが返る【NGKI0409】。

3155
3156 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
3157 ない書込みアクセスまたは読出しアクセス（実行アクセスを含む）を行おうと
3158 した場合、CPU例外ハンドラが起動される【NGKI0410】。どのCPU例外ハンドラが
3159 起動されるかは、ターゲット定義である【NGKI0411】。ターゲットによっては、
3160 エミュレートされたCPU例外ハンドラの場合もある。また、ターゲット定義で、
3161 アクセス違反の状況に応じて異なるCPU例外ハンドラが起動される場合もある。
3162 この（これらの）CPU例外ハンドラを、メモリアクセス違反ハンドラと呼ぶ。

3163
3164 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
3165 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
3166 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
3167 【NGKI0412】。E_MACVエラーが返るかメモリアクセス違反ハンドラが起動され
3168 るかは、ターゲット定義である【NGKI0413】。

3169
3170 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
3171 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）
3172 を参照する方法を、ターゲット定義で用意する【NGKI0414】。

3173
3174 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、
3175 ユーザドメインから書込みアクセスまたは読出しアクセス（実行アクセスを含
3176 む）を行おうとした場合には、メモリオブジェクトに対するアクセスが許可さ
3177 れていない場合と同様に扱われる【NGKI0415】。カーネルドメインから同様の
3178 アクセスを行おうとした場合の動作は保証されない【NGKI0416】。

3179
3180 **【未決定事項】**

3181
3182 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
3183 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。

3184
3185 **【μ ITRON4.0/PX仕様との関係】**

3186
3187 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
3188 るが、この仕様では、メモリオブジェクトに対するアクセス許可ベクタのターゲッ
3189 ト定義の制限以外については規定していない。

3190
3191 **【仕様決定の理由】**

3192
3193 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
3194 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
3195 可ベクタが設定されていないためである。

3196
3197 **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**

3198
3199 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）
3200 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

3201 合がある【NGKI0417】。

3202

3203 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。

3204

3205 (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
3206 アクセスを含む）のみを許可する【NGKI0418】。これを、専有リードオン
3207 リー（private read only）と呼ぶ。

3208

3209 (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
3210 しアクセス（実行アクセスを含む）を許可する【NGKI0419】。これを、専
3211 有リードライト（private read/write）と呼ぶ。

3212

3213 (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
3214 許可する【NGKI0420】。これを、共有リードオンリー（shared read only）
3215 と呼ぶ。

3216

3217 (d) すべての保護ドメインに、書込みアクセスと読出しアクセス（実行アクセ
3218 スを含む）を許可する【NGKI0421】。これを、共有リードライト（shared
3219 read/write）と呼ぶ。

3220

3221 (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
3222 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
3223 アクセス（実行アクセスを含む）のみを許可する【NGKI0422】。これを、
3224 共有リード専有ライト（shared read private write）と呼ぶ。

3225

3226 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
3227 の数が制限される場合がある【NGKI0423】。

3228

3229 2. 11. 3 デフォルトのアクセス許可ベクタ

3230

3231 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ルトのアクセ
3232 ス許可ベクタが設定される。

3233

3234 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
3235 がいずれも、その保護ドメインのみに許可される【NGKI0424】。すなわち、カー
3236 ネルドメインに属するオブジェクトに対しては、4つのアクセス許可パターンが
3237 いずれもTACP_KERNELに、ユーザドメインに属するオブジェクトに対しては、4
3238 つのアクセス許可パターンがいずれもTACP(domid)（domidはオブジェクトが属
3239 する保護ドメインのID番号）に設定される。ただし、カーネルオブジェクトを
3240 サービスコールにより登録した場合には、管理操作に対するアクセスは、サー
3241 ビスコールを呼び出した処理単位が属する保護ドメインにも許可される
3242 【NGKI3427】。

3243

3244 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、
3245 すべての保護ドメインに許可される【NGKI0425】。すなわち、4つのアクセス許
3246 可パターンがいずれも、TACP_SHAREDに設定される。

3247

3248 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
3249 ネルドメインのみに許可される【NGKI0426】。すなわち、4つのアクセス許可パ
3250 ターンがいずれも、TACP_KERNELに設定される。

2. 11. 4 アクセス許可ベクタの設定

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

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

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

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

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

【使用上の注意】

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

【μ ITRON4. 0/PX仕様との関係】

アクセス許可ベクタを指定してオブジェクトを生成する静的API (CRA_YYY) は廃止し、オブジェクトの登録後にアクセス許可ベクタを設定する静的API (SAC_YYY) をサポートすることとした。

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

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

3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350

2. 11. 5 カーネルの管理領域のアクセス保護

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

カーネルの用いるオブジェクト管理領域（カーネルの管理領域に該当する。「2. 10. 4 オブジェクト生成に必要なメモリ領域」の節を参照）として、カーネル専用のメモリオブジェクトに含まれないメモリ領域を指定した場合、E_OBJエラーとなる【NGKI0434】。また、カーネルの用いるオブジェクト管理領域の先頭番地にNULL を指定した場合、必要なメモリ領域が、カーネル専用のメモリオブジェクトの中に確保される【NGKI0435】。

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

2. 11. 6 ユーザタスクのユーザスタック領域

ユーザタスクが非特権モードで実行する間に用いるスタック領域を、システムスタック領域（「4. 1 タスク管理機能」の節を参照）と対比させて、ユーザスタック領域と呼ぶ。ユーザスタック領域は、そのタスクと同じ保護ドメインに属する1つのメモリオブジェクトとしてカーネルに登録される【NGKI0437】。ただし、他のメモリオブジェクトとは異なり、次のように扱われる。

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

ただし、上記の仕様を実現するために大きいオーバーヘッドを生じる場合には、ターゲット定義で、タスクのユーザスタック領域を、そのタスクが属する保護ドメイン全体からアクセスできるものとする場合がある【NGKI0441】。

【 μ ITRON4. 0/PX仕様との関係】

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

2. 12 システムコンフィギュレーション手順

2. 12. 1 システムコンフィギュレーションファイル

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

システムコンフィギュレーションファイルには、カーネルの静的API、システムサービスの静的API、保護ドメインの囲み、クラスの囲み、コンフィギュレータに対するINCLUDEディレクティブ、C言語プリプロセッサのインクルードディレクティブ (#include) と条件ディレクティブ (#if, #ifdef など) のみを記述することができる【NGKI0442】。

コンフィギュレータに対するINCLUDEディレクティブは、システムコンフィギュレーションファイルを複数のファイルに分割して記述するために用いるもので、その文法は次のいずれかである (両者の違いは、指定されたファイルを探すディレクトリの違いのみ)【NGKI0443】。

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

コンフィギュレータは、INCLUDEディレクティブによって指定されたファイル中の記述を、システムコンフィギュレーションファイルの一部として解釈する【NGKI0444】。すなわち、INCLUDEディレクティブによって指定されたファイル中には、カーネルの静的API、システムサービスの静的API、コンフィギュレータに対するINCLUDEディレクティブ、C言語プリプロセッサのインクルードディレクティブと条件ディレクティブのみを記述することができる。

C言語プリプロセッサのインクルードディレクティブは、静的APIのパラメータを解釈するために必要なC言語のヘッダファイルを指定するために用いる【NGKI0445】。また、条件ディレクティブは、有効とする静的APIを選択するために用いることができる【NGKI0446】。ただし、インクルードディレクティブは、コンフィギュレータが生成するファイルでは先頭に集められる【NGKI0447】。そのため、条件ディレクティブの中にインクルードディレクティブを記述しても、インクルードディレクティブは常に有効となる。また、1つの静的APIの記述の途中に、条件ディレクティブを記述することはできない【NGKI0448】。

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

【μ ITRON4. 0仕様との関係】

システムコンフィギュレーションファイルにおけるC言語プリプロセッサのディレクティブの扱いを全面的に見直し、コンフィギュレータに対するINCLUDEディ

3401 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
3402 #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
3403 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
3404 は#includeディレクティブに置き換わる。

3405

3406 2.12.2 静的APIの文法とパラメータ

3407

3408 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
3409 述する【NGKI0450】。すなわち、静的APIの名称に続けて、静的APIの各パラメー
3410 タを“, ”で区切って列挙したものを“(”と”) ”で囲んで記述し、最後に”; ”を記述
3411 する。ただし、静的APIのパラメータに構造体（または構造体へのポインタ）を
3412 記述する場合には、構造体の各フィールドを“, ”で区切って列挙したものを“{ ”
3413 と”} ”で囲んだ形で記述する【NGKI0451】。

3414

3415 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
3416 サービスコールのパラメータと同一とすることを原則とする【NGKI0452】。

3417

3418 静的APIのパラメータは、次の4種類に分類される。

3419

3420 (a) オブジェクト識別名

3421

3422 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
3423 の識別名のみを記述することができる。

3424

3425 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
3426 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、
3427 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
3428 ディレクティブ（#define）を生成する【NGKI0453】。

3429

3430 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
3431 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
3432 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
3433 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
3434ギュレータがエラーを報告する【NGKI0455】。

3435

3436 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
3437 はできない【NGKI0456】。

3438

3439 (b) 整数定数式パラメータ

3440

3441 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、
3442 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
3443 決まる整数定数式を記述することができる。

3444

3445 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
3446 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
3447 クルードするファイルに含まれていなければならない【NGKI0457】。

3448

3449 (c) 一般定数式パラメータ

3450

3451 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
3452 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．
3453
3454 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
3455 イルからC言語プリプロセッサのインクルードディレクティブによってインクルー
3456 ドするファイルに含まれていなければならない【NGKI0458】．
3457
3458 (d) 文字列パラメータ
3459
3460 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．
3461 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．
3462
3463 【 μ ITRON4.0仕様との関係】
3464
3465 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
3466 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．
3467
3468 (A) 自動割付け対応整数値パラメータ
3469 (B) 自動割付け非対応整数値パラメータ
3470 (C) プリプロセッサ定数式パラメータ
3471 (D) 一般定数式パラメータ
3472
3473 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
3474 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
3475 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．
3476
3477 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
3478 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
3479 るC言語プリプロセッサの条件ディレクティブ（#if）中に含めたい可能性のあ
3480 るパラメータを(C)としていた．
3481
3482 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
3483 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
3484 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
3485 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
3486 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．
3487
3488 2.12.3 保護ドメインの指定
3489
3490 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
3491 クトが属する保護ドメインの囲みの中に記述する【NGKI0459】．無所属のオブ
3492 ジェクトを登録する静的APIは，保護ドメインの囲みの外に記述する
3493 【NGKI0460】．保護ドメインに属すべきオブジェクトを登録する静的API等を，
3494 保護ドメインの囲みの外に記述した場合には，コンフィギュレータがE_RSATRエ
3495 ラーを報告する【NGKI0461】．
3496
3497 ユーザドメインの囲みの文法は次の通り【NGKI0462】．
3498
3499 DOMAIN(保護ドメイン名) {
3500 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

3501 }

3502

3503 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
3504 ることができる【NGKI0463】。

3505

3506 コンフィギュレータは、ユーザドメインの囲み进行处理する際に、ユーザドメイ
3507 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
3508 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
3509 サのディレクティブ（#define）を生成する【NGKI0464】。また、ユーザドメイ
3510 ンの囲みの中およびそれ以降に記述する静的APIの整数定数式パラメータの記述
3511 に保護ドメイン名を記述すると、割り付けた保護ドメインIDの値に評価される
3512 【NGKI0465】。

3513

3514 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
3515 ンIDの割付けのみを行うことができる【NGKI0466】。

3516

3517 カーネルドメインの囲みの文法は次の通り【NGKI0467】。

3518

```
3519     KERNEL_DOMAIN {  
3520         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  
3521     }
```

3522

3523 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
3524 囲みを、複数回記述してもよい【NGKI0468】。保護機能対応でないカーネルで
3525 保護ドメインの囲みを記述した場合や、保護ドメインの囲みの中に保護ドメイ
3526 ンの囲みを記述した場合には、コンフィギュレータがエラーを報告する
3527 【NGKI0469】。

3528

3529 【μITRON4.0/PX仕様との関係】

3530

3531 保護ドメインの囲みの文法を変更した。

3532

3533 【仕様決定の理由】

3534

3535 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
3536 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
3537 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
3538 クト属性によって指定するためである。

3539

3540 2.12.4 クラスの指定

3541

3542 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
3543 のオブジェクトが属するクラスの囲みの中に記述する【NGKI0470】。クラスに
3544 属すべきオブジェクトを登録する静的API等を、クラスの囲みの外に記述した場
3545 合には、コンフィギュレータがE_RSATRエラーを報告する【NGKI0471】。

3546

3547 クラスの囲みの文法は次の通り【NGKI0472】。

3548

```
3549     CLASS(クラスID) {  
3550         クラスに属するオブジェクトを登録する静的API等
```

3551 }

3552

3553 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
3554 ができる【NGKI0473】。使用できないクラスIDを指定した場合には、コンフィ
3555 ギュレータがE_IDエラーを報告する【NGKI0474】。

3556

3557 同じクラスIDを指定したクラスの囲みを複数回記述してもよい【NGKI0475】。
3558 マルチプロセッサ対応でないカーネルでクラスの囲みを記述した場合や、クラ
3559 スの囲みの中にクラスの囲みを記述した場合には、コンフィギュレータがエラー
3560 を報告する【NGKI0476】。

3561

3562 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
3563 インの囲みとクラスの囲みはどちらが外側になっていてもよい【NGKI0477】。

3564

3565 【仕様決定の理由】

3566

3567 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
3568 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
3569 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
3570 定するためである。

3571

3572 2. 12. 5 コンフィギュレータの処理モデル

3573

3574 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
3575 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
3576 成する（図2-8）。

3577

3578 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
3579 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
3580 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
3581 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
3582 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
3583 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
3584 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
3585 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。

3586

3587 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
3588 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
3589 マットの形に変換したSレコードファイル（cfg1_out.srec）と、その中の各シン
3590 ボルとアドレスの対応表を含むシンボルファイル（cfg1_out.syms）を生成す
3591 る。静的APIの整数定数式パラメータに解釈できない式が記述された場合には、
3592 この段階でエラーが検出される。

3593

3594 コンフィギュレータのパス2では、パス1で生成されたロードモジュールのSレコー
3595 ドファイルとシンボルファイルから、C言語プリプロセッサの条件ディレクティ
3596 ブによりどの静的APIが有効となったかと、それらの静的APIの整数定数式パラ
3597 メータの値を取り出し、カーネルおよびシステムサービスの構成・初期化ファ
3598 イル（kernel_cfg.cなど）と構成・初期化ヘッダファイル（kernel_cfg.hなど）
3599 を生成する。構成・初期化ヘッダファイルには、登録できるオブジェクトの数
3600 （動的生成対応カーネル以外では、静的APIによって登録されたオブジェクトの

3601 数に一致) やオブジェクトのID番号などの定義を出力する。静的APIの整数定数
3602 式パラメータに不正がある場合には、この段階でエラーが検出される。
3603
3604 パス2で生成されたファイルを、他のソースファイルとあわせてコンパイルし、
3605 アプリケーションのロードモジュールを生成する。また、そのSレコードファイ
3606 ル (system.srec) とシンボルフайル (system.syms) を生成する。静的APIの
3607 一般定数式パラメータに解釈できない式が記述された場合には、この段階でエ
3608 ラーが検出される。
3609
3610 コンフィギュレータのパス3では、パス1およびパス2で生成されたロードモジュー
3611 ルのSレコードファイルとシンボルフайルから、静的APIのパラメータの値な
3612 どを取り出し、妥当性のチェックを行う。静的APIの一般定数式パラメータに不
3613 正がある場合には、この段階でエラーが検出される。
3614
3615 保護機能対応カーネルにおいては、メモリ配置を決定し、メモリ保護のための
3616 設定情報を生成するために、さらに以下の処理を行う (図2-9)。
3617
3618 コンフィギュレータは、決定したメモリ配置に従ってロードモジュールを生成
3619 するために、リンクスクリプト (ldscript.ld) を生成する。また、メモリ保護
3620 のための設定情報を、メモリ構成・初期化ファイル (kernel_mem.c) に生成す
3621 る。これらのファイルを生成するためには、パス3以降で初めて得られる情報が
3622 必要となるため、これらのファイルはパス3以降でしか生成できず、最終的なロー
3623 ドモジュールも、パス3以降で生成する。
3624
3625 そのため、パス2で生成されたロードモジュールは、仮のロードモジュールとい
3626 う位置付けになる。ここで、パス3以降に必要な情報を取り出し、最終的なロー
3627 ドモジュールのサイズを割り出せるように、パス3以降でメモリ構成・初期化フ
3628 ァイルに生成するのと同様のデータ構造を、パス2において仮のメモリ構成・初
3629 期化ファイル (kernel_mem2.c) に生成する。また、これをリンクするための仮の
3630 リンクスクリプト (cfg2_out.ld) を生成し、これらを用いて仮のロードモジュー
3631 ルを生成する。さらに、仮のロードモジュールのSレコードファイル
3632 (cfg2_out.srec) とシンボルフайル (cfg2_out.syms) も、最終的なものと
3633 混同しないように、異なるファイル名で生成する。
3634
3635 パス3は、ターゲット依存で用いるパスで、メモリ配置やメモリ保護のための設
3636 定情報のサイズを最適化するための処理を行う。パス2で生成された仮のロード
3637 モジュールのSレコードファイルとシンボルフайルから必要な情報を取り出し、
3638 再度、仮のメモリ構成・初期化ファイル (kernel_mem3.c) と仮のリンクスクリ
3639 プト (cfg3_out.ld) を生成する。また、これらのファイルを他のソースファイ
3640 ルとあわせてコンパイルして仮のロードモジュールを生成し、そのSレコードフ
3641 ァイル (cfg3_out.srec) とシンボルフайル (cfg3_out.syms) を生成する。こ
3642 の段階で、メモリオブジェクトに重なりがあるなどのエラーが検出される場合
3643 もある。
3644
3645 パス4では、パス3 (パス3を用いない場合はパス2) で生成された仮のロードモ
3646 ジュールのSレコードファイルとシンボルフайルから必要な情報を取り出し、
3647 最終的なメモリ構成・初期化ファイル (kernel_mem.c) とリンクスクリプト
3648 (ldscript.ld) を生成する。またパス4では、保護機能対応でないカーネルに
3649 においてパス3で行っていた静的APIパラメータの値などの妥当性のチェックも行
3650 う。そのため、静的APIの一般定数式パラメータに不正がある場合には、この段

3651 階でエラーが検出される.

3652

3653 パス4で生成されたファイルを, 他のソースファイルとあわせてコンパイルし,
3654 アプリケーションの最終的なロードモジュールを生成する. また, そのSレコー
3655 ドファイル (system.srec, 必要な場合のみ) とシンボルフайル
3656 (system.syms) を生成する.

3657

3658 最後に, 最終的なロードモジュールが, パス3 (パス3を用いない場合はパス2)
3659 で生成された仮のロードモジュールと同じメモリ配置であることをチェックす
3660 る. 両者のメモリ配置が異なっていた場合には, ロードモジュールが正しく生
3661 成されていない可能性があるが, これは, コンフィギュレーション処理の不具
3662 合を示すものである.

3663

3664 【μ ITRON4.0仕様との関係】

3665

3666 コンフィギュレータの処理モデルは全面的に変更した.

3667

3668 2.12.6 静的APIのパラメータに関するエラー検出

3669

3670 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
3671 呼ばれた場合と同等とすることを原則とする【NGKI0478】. 言い換えると, サー
3672 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.
3673 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」
3674 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.

3675

3676 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
3677 ことが難しいため, そのようなものについては例外とする【NGKI0479】. 例え
3678 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.

3679

3680 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
3681 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,
3682 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
3683 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
3684 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
3685 とせず, 検出できた場合には警告として報告する【NGKI0480】.

3686

3687 【μ ITRON4.0仕様との関係】

3688

3689 μ ITRON4.0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
3690 れていない.

3691

3692 2.12.7 オブジェクトのID番号の指定

3693

3694 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
3695 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.

3696

3697 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
3698 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
3699 クトにID番号を割り付ける【NGKI0481】. それに従ったID番号割付けができな
3700 い場合 (ID番号に抜けができる場合など) には, コンフィギュレータはエラー

3701 を報告する【NGKI0482】。
3702
3703 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
3704 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
3705 レータに渡すファイルと同じフォーマットで生成する【NGKI0483】。
3706
3707 【μ ITRON4.0仕様との関係】
3708
3709 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
3710 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。
3711
3712 2.13 TOPPERSネーミングコンベンション
3713
3714 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
3715 コンベンションについて述べる。このネーミングコンベンションは、モジュール
3716 間のインタフェースに関わる名称に適用することを想定しているが、モジュール
3717 内部の名称に適用してもよい。
3718
3719 2.13.1 モジュール識別名
3720
3721 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
3722 ジュールに対して、それを識別するためのモジュール識別名を定める。モジュール
3723 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。
3724
3725 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
3726 ジュール識別名は“sil”とする。
3727
3728 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
3729 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
3730 識別名を含めない名称を使用する。
3731
3732 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
3733 したものをWWWと表記する。
3734
3735 2.13.2 データ型名
3736
3737 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
3738 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
3739 “_t”である名称とする。
3740
3741 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，
3742 “_”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
3743 が“_T”である名称とする場合もある。
3744
3745 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。
3746
3747 (A) パケットのデータ型
3748
3749 T_CYYY acre_yyyに渡すパケットのデータ型
3750 T_DYYY def_yyyに渡すパケットのデータ型

3751	T_RYYY	ref_yyyに渡すパケットのデータ型
3752	T_WWW_CYYY	www_acre_yyyに渡すパケットのデータ型
3753	T_WWW_DYYY	www_def_yyyに渡すパケットのデータ型
3754	T_WWW_RYYY	www_ref_yyyに渡すパケットのデータ型

3755

3756 2.13.3 関数名

3757

3758 関数の名称は、英小文字、数字、“_”で構成する。

3759

3760 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。

3761

3762 (A) サービスコール

3763

3764 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
 3765 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
 3766 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、
 3767 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
 3768 は、zzxxx_yyyまたはwww_zzzxxx_yyyの名称となる。

3769

3770 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
 3771 字として“i”を付加し、ixxx_yyy, izxxx_yyy, www_ixxx_yyy, www_izxxx_yyyと
 3772 いった名称とする。

3773

3774 【補足説明】

3775

3776 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
 3777 いては、「5.10 省略名の元になった英語」の節を参照すること。

3778

3779 (B) コールバック

3780

3781 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。

3782

3783 2.13.4 変数名

3784

3785 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
 3786 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。

3787

3788 変数の名称に関して、次のガイドラインを設ける。

3789

3790	～id	～ID（オブジェクトのID番号，ID型）
3791	～no	～番号（オブジェクト番号）
3792	～atr	～属性（オブジェクト属性，ATR型）
3793	～stat	～状態（オブジェクト状態，STAT型）
3794	～mode	～モード（サービスコールの動作モード，MODE型）
3795	～pri	～優先度（優先度，PRI型）
3796	～sz	～サイズ（単位はバイト数，SIZE型またはuint_t型）
3797	～cnt	～の個数（単位は個数，uint_t型）
3798	～ptn	～パターン
3799	～tim	～時刻，～時間
3800	～cd	～コード

3801 i～ ～の初期値
3802 max～ ～の最大値
3803 min～ ～の最小値
3804 left～ ～の残り

3805

3806 また、ポインタ変数（関数ポインタを除く）の名称に関して、次のガイドライン
3807 を設ける.

3808

3809 p_～ ポインタ
3810 pp_～ ポインタを入れる領域へのポインタ
3811 pk_～ パケットへのポインタ
3812 ppk_～ パケットへのポインタを入れる領域へのポインタ

3813

3814 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.

3815

3816 (A) パケットへのポインタ

3817

3818 pk_cyyy acre_yyyに渡すパケットへのポインタ
3819 pk_dyyy def_yyyに渡すパケットへのポインタ
3820 pk_ryyy ref_yyyに渡すパケットへのポインタ
3821 pk_www_cyyy www_acre_yyyに渡すパケットへのポインタ
3822 pk_www_dyyy www_def_yyyに渡すパケットへのポインタ
3823 pk_www_ryyy www_ref_yyyに渡すパケットへのポインタ

3824

3825 2. 13. 5 定数名

3826

3827 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
3828 字、"_"で構成する.

3829

3830 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.

3831

3832 (A) メインエラーコード

3833

3834 メインエラーコードは、先頭が"E_"である名称とする.

3835

3836 (B) 機能コード

3837

3838 TFN_XXX_YYY xxx_yyyの機能コード
3839 TFN_WWW_XXX_YYY www_xxx_yyyの機能コード

3840

3841 (C) その他の定数

3842

3843 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、
3844 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
3845 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.

3846

3847 また、定数の名称に関して、次のガイドラインを設ける.

3848

3849 TA_～ オブジェクトの属性値
3850 TSZ_～ ～のサイズ

3851 TBIT_～ ～のビット数
3852 TMAX_～ ～の最大値
3853 TMIN_～ ～の最小値

3854

3855 2. 13. 6 マクロ名

3856

3857 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
3858 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
3859 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
3860 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
3861 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
3862 する場合もある。

3863

3864 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。

3865

3866 (A) 構成マクロ

3867

3868 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
3869 る。

3870

3871 TSZ_～ ～のサイズ
3872 TBIT_～ ～のビット数
3873 TMAX_～ ～の最大値
3874 TMIN_～ ～の最小値

3875

3876 2. 13. 7 静的API名

3877

3878 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
3879 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
3880 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
3881 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。

3882

3883 2. 13. 8 ファイル名

3884

3885 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
3886 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。
3887 また、“-”も使用しない。

3888

3889 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。

3890

3891 (A) ヘッダファイル

3892

3893 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
3894 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。

3895

3896 2. 13. 9 モジュール内部の名称の衝突回避

3897

3898 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
3899 ために、次のガイドラインを設ける。

3900

3901 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
3902 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
3903 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、
3904 C言語レベルで、先頭が`_kernel_`または`_KERNEL_`である名称とする。

3905

3906 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
3907 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
3908 能性のある名称は、`"TOPPERS_"`で始まる名称とする。

3909

3910 2. 14 TOPPERS共通定義

3911

3912 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。

3913

3914 2. 14. 1 TOPPERS共通ヘッダファイル

3915

3916 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
3917 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
3918 れている【NGKI0484】。TOPPERS共通定義を用いる場合には、TOPPERS共通ヘッ
3919 ダファイルをインクルードする【NGKI0485】。

3920

3921 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
3922 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、
3923 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
3924 直接インクルードする必要はない【NGKI0486】。

3925

3926 2. 14. 2 TOPPERS共通データ型

3927

3928 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
3929 タ型は次の通りである【NGKI0487】。

3930

3931	<code>int8_t</code>	符号付き8ビット整数（オプション，C99準拠）
3932	<code>uint8_t</code>	符号無し8ビット整数（オプション，C99準拠）
3933	<code>int16_t</code>	符号付き16ビット整数（C99準拠）
3934	<code>uint16_t</code>	符号無し16ビット整数（C99準拠）
3935	<code>int32_t</code>	符号付き32ビット整数（C99準拠）
3936	<code>uint32_t</code>	符号無し32ビット整数（C99準拠）
3937	<code>int64_t</code>	符号付き64ビット整数（オプション，C99準拠）
3938	<code>uint64_t</code>	符号無し64ビット整数（オプション，C99準拠）
3939	<code>int128_t</code>	符号付き128ビット整数（オプション，C99準拠）
3940	<code>uint128_t</code>	符号無し128ビット整数（オプション，C99準拠）

3941

3942	<code>int_least8_t</code>	8ビット以上の符号付き整数（C99準拠）
3943	<code>uint_least8_t</code>	<code>int_least8_t</code> 型と同じサイズの符号無し整数（C99準拠）

3944

3945	<code>float32_t</code>	IEEE754準拠の32ビット単精度浮動小数点数（オプション）
3946	<code>double64_t</code>	IEEE754準拠の64ビット倍精度浮動小数点数（オプション）

3947

3948	<code>bool_t</code>	真偽値（ <code>true</code> または <code>false</code> ）
------	---------------------	---

3949	<code>int_t</code>	16ビット以上の符号付き整数
------	--------------------	----------------

3950	<code>uint_t</code>	<code>int_t</code> 型と同じサイズの符号無し整数
------	---------------------	-----------------------------------

3951	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
3952	ulong_t	long_t型と同じサイズの符号無し整数
3953		
3954	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)
3955	uintptr_t	intptr_t型と同じサイズの符号無し整数 (C99準拠)
3956		
3957	FN	機能コード (符号付き整数, int_tに定義)
3958	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)
3959		
3960	ID	オブジェクトのID番号 (符号付き整数, int_tに定義)
3961	ATR	オブジェクト属性 (符号無し整数, uint_tに定義)
3962	STAT	オブジェクトの状態 (符号無し整数, uint_tに定義)
3963	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)
3964	PRI	優先度 (符号付き整数, int_tに定義)
3965	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)
3966		
3967		
3968	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)
3969	RELTIM	相対時間 (符号無し整数, 単位はミリ秒, uint_tに定義)
3970	SYSTIM	システム時刻 (符号無し整数, 単位はミリ秒, ulong_tに定義)
3971	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)
3972		
3973		
3974	FP	プログラムの起動番地 (型の定まらない関数ポインタ)
3975		
3976	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)
3977	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)
3978		
3979	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)
3980		
3981		
3982		
3983	MB_T	オブジェクト管理領域を確保するためのデータ型
3984		
3985	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)
3986		
3987	ACVCT	アクセス許可ベクタ
3988		
3989	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。	
3990	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。	
3991		
3992	int8_t, uint8_t, int64_t, uint64_t, int128_t, uint128_t, float32_t, double64_tが使用できるかどうかは、ターゲット定義である【NGKI0488】。これらが使用できるかどうかは、それぞれ、INT8_MAX, UINT8_MAX, INT64_MAX, UINT64_MAX, INT128_MAX, UINT128_MAX, FLOAT32_MAX, DOUBLE64_MAXがマクロ定義されているかどうかで判別することができる【NGKI0489】。IEEE754準拠の浮動小数点数がサポートされていない場合には、ターゲット定義で、float32_tとdouble64_tは使用できないものとする【NGKI0490】。	
3993		
3994		
3995		
3996		
3997		
3998		
3999		
4000	【μ ITRON4.0仕様との関係】	

4001
4002 B, UB, H, UH, W, UW, D, UD, VP_INTに代えて, C99準拠のint8_t, uint8_t,
4003 int16_t, uint16_t, int32_t, uint32_t, int64_t, uint64_t, intptr_tを用い
4004 ることにした. また, uintptr_t, int128_t, uint128_tを用意することにした.
4005
4006 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
4007 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
4008 は用意しないことにした.
4009
4010 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.
4011 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
4012 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
4013 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
4014 た. int_least16_t, uint_least16_t, int_least32_t, uint_least32_tを導入
4015 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,
4016 それぞれint16_t, uint16_t, int32_t, uint32_tで代用できるためである.
4017
4018 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
4019 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
4020 の64ビットを表す型としてdouble64_tを導入した.
4021
4022 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
4023 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
4024
4025 2. 14. 3 TOPPERS共通定数
4026
4027 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
4028 次の通りである (一部, C90に規定されているものも含む) .
4029
4030 (1) 一般定数 【NGKI0491】
4031
4032 NULL 無効ポインタ
4033
4034 true 1 真
4035 false 0 偽
4036
4037 E_OK 0 正常終了
4038
4039 【μ ITRON4. 0仕様との関係】
4040
4041 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
4042 を用いることにした.
4043
4044 (2) 整数型に格納できる最大値と最小値 【NGKI0492】
4045
4046 INT8_MAX int8_tに格納できる最大値 (オプション, C99準拠)
4047 INT8_MIN int8_tに格納できる最小値 (オプション, C99準拠)
4048 UINT8_MAX uint8_tに格納できる最大値 (オプション, C99準拠)
4049 INT16_MAX int16_tに格納できる最大値 (C99準拠)
4050 INT16_MIN int16_tに格納できる最小値 (C99準拠)

4051	UINT16_MAX	uint16_tに格納できる最大値 (C99準拠)
4052	INT32_MAX	int32_tに格納できる最大値 (C99準拠)
4053	INT32_MIN	int32_tに格納できる最小値 (C99準拠)
4054	UINT32_MAX	uint32_tに格納できる最大値 (C99準拠)
4055	INT64_MAX	int64_tに格納できる最大値 (オプション, C99準拠)
4056	INT64_MIN	int64_tに格納できる最小値 (オプション, C99準拠)
4057	UINT64_MAX	uint64_tに格納できる最大値 (オプション, C99準拠)
4058	INT128_MAX	int128_tに格納できる最大値 (オプション, C99準拠)
4059	INT128_MIN	int128_tに格納できる最小値 (オプション, C99準拠)
4060	UINT128_MAX	uint128_tに格納できる最大値 (オプション, C99準拠)
4061		
4062	INT_LEAST8_MAX	int_least8_tに格納できる最大値 (C99準拠)
4063	INT_LEAST8_MIN	int_least8_tに格納できる最小値 (C99準拠)
4064	UINT_LEAST8_MAX	uint_least8_tに格納できる最大値 (C99準拠)
4065	INT_MAX	int_tに格納できる最大値 (C90準拠)
4066	INT_MIN	int_tに格納できる最小値 (C90準拠)
4067	UINT_MAX	uint_tに格納できる最大値 (C90準拠)
4068	LONG_MAX	long_tに格納できる最大値 (C90準拠)
4069	LONG_MIN	long_tに格納できる最小値 (C90準拠)
4070	ULONG_MAX	ulong_tに格納できる最大値 (C90準拠)
4071		
4072	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)
4073		
4074	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)
4075		
4076	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)
4077		
4078	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)
4079		
4080		
4081	(3) 整数型のビット数 【NGKI0493】	
4082		
4083	CHAR_BIT	char型のビット数 (C90準拠)
4084		
4085	(4) オブジェクト属性 【NGKI0494】	
4086		
4087	TA_NULL	0U オブジェクト属性を指定しない
4088		
4089	(5) タイムアウト指定 【NGKI0495】	
4090		
4091	TMO_POL	0 ポーリング
4092	TMO_FEVR	-1 永久待ち
4093	TMO_NBLK	-2 ノンブロッキング
4094		
4095	(6) アクセス許可パターン 【NGKI0496】	
4096		
4097	TACP_KERNEL	0U カーネルドメインのみにアクセスを許可
4098	TACP_SHARED	~0U すべての保護ドメインにアクセスを許可
4099		
4100	2. 14. 4 TOPPERS共通エラーコード	

4101
4102 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
4103 【NGKI0497】 .
4104
4105 (A) 内部エラークラス (EC_SYS, -5~-8)
4106
4107 E_SYS -5 システムエラー
4108
4109 (B) 未サポートエラークラス (EC_NOSPT, -9~-16)
4110
4111 E_NOSPT -9 未サポート機能
4112 E_RSFN -10 予約機能コード
4113 E_RSATR -11 予約属性
4114
4115 (C) パラメータエラークラス (EC_PAR, -17~-24)
4116
4117 E_PAR -17 パラメータエラー
4118 E_ID -18 不正ID番号
4119
4120 (D) 呼出しコンテキストエラークラス (EC_CTX, -25~-32)
4121
4122 E_CTX -25 コンテキストエラー
4123 E_MACV -26 メモリアクセス違反
4124 E_OACV -27 オブジェクトアクセス違反
4125 E_ILUSE -28 サービスコール不正使用
4126
4127 (E) 資源不足エラークラス (EC_NOMEM, -33~-40)
4128
4129 E_NOMEM -33 メモリ不足
4130 E_NOID -34 ID番号不足
4131 E_NORES -35 資源不足
4132
4133 (F) オブジェクト状態エラークラス (EC_OBJ, -41~-48)
4134
4135 E_OBJ -41 オブジェクト状態エラー
4136 E_NOEXS -42 オブジェクト未登録
4137 E_QOVR -43 キューイングオーバフロー
4138
4139 (G) 待ち解除エラークラス (EC_RLWAI, -49~-56)
4140
4141 E_RLWAI -49 待ち禁止状態または待ち状態の強制解除
4142 E_TMOUT -50 ポーリング失敗またはタイムアウト
4143 E_DLT -51 待ちオブジェクトの削除または再初期化
4144 E_CLS -52 待ちオブジェクトの状態変化
4145
4146 (H) 警告クラス (EC_WARN, -57~-64)
4147
4148 E_WBLK -57 ノンブロッキング受付け
4149 E_BOVR -58 バッファオーバフロー
4150

4151 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、
4152 [NGKI0019] の原則では例外としている。

4153
4154 **【 μ ITRON4.0仕様との関係】**

4155
4156 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。

4157
4158 2.14.5 TOPPERS共通マクロ

4159
4160 (1) 整数定数を作るマクロ **【NGKI0498】**

4161
4162 INT8_C(val) int_least8_t型の定数を作るマクロ (C99準拠)
4163 UINT8_C(val) uint_least8_t型の定数を作るマクロ (C99準拠)
4164 INT16_C(val) int16_t型の定数を作るマクロ (C99準拠)
4165 UINT16_C(val) uint16_t型の定数を作るマクロ (C99準拠)
4166 INT32_C(val) int32_t型の定数を作るマクロ (C99準拠)
4167 UINT32_C(val) uint32_t型の定数を作るマクロ (C99準拠)
4168 INT64_C(val) int64_t型の定数を作るマクロ (オプション, C99準拠)
4169 UINT64_C(val) uint64_t型の定数を作るマクロ (オプション, C99準拠)
4170 INT128_C(val) int128_t型の定数を作るマクロ (オプション, C99準拠)
4171 UINT128_C(val) uint128_t型の定数を作るマクロ (オプション, C99準拠)
4172
4173 UINT_C(val) uint_t型の定数を作るマクロ
4174 ULONG_C(val) ulong_t型の定数を作るマクロ

4175
4176 **【仕様決定の理由】**

4177
4178 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
4179 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
4180 クロを使って記述する必要はない。

4181
4182 (2) 型に関する情報を取り出すためのマクロ **【NGKI0499】**

4183
4184 offsetof(structure, field) 構造体structure中のフィールドfieldの
4185 バイト位置を返すマクロ (C90準拠)
4186
4187 alignof(type) 型typeのアラインメント単位を返すマクロ
4188
4189 ALIGN_TYPE(addr, type) 番地addrが型typeに対してアラインしてい
4190 るかどうかを返すマクロ

4191
4192 (3) assertマクロ **【NGKI0500】**

4193
4194 assert(exp) expが成立しているかを検査するマクロ (C90準拠)

4195
4196 (4) コンパイラの拡張機能のためのマクロ **【NGKI0501】**

4197
4198 inline インライン関数
4199 Inline ファイルローカルなインライン関数
4200 asm インラインアセンブラ

4201 Asm インラインアセンブラ（最適化抑止）
 4202 throw() 例外を発生しない関数
 4203 NoReturn リターンしない関数
 4204
 4205 (5) エラーコード構成・分解マクロ【NGKI0502】
 4206
 4207 ERCD(mercd, sercd) メインエラーコードmercdとサブエラーコードsercdか
 4208 ら、エラーコードを構成するためのマクロ
 4209
 4210 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
 4211 ためのマクロ
 4212 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
 4213 めのマクロ
 4214
 4215 (6) アクセス許可パターン構成マクロ【NGKI0503】
 4216
 4217 TACP(domid) domidで指定されるユーザドメインのみにアクセスを
 4218 許可するアクセス許可パターンを構成するためのマ
 4219 クロ
 4220
 4221 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
 4222 することができる【NGKI0504】．TDOM_SELF, TDOM_KERNEL, TDOM_NONEを指定し
 4223 た場合、どのようなアクセス許可パターンが構成されるかは保証されない
 4224 【NGKI0505】．
 4225
 4226 2. 14. 6 TOPPERS共通構成マクロ
 4227
 4228 (1) 相対時間の範囲【NGKI0506】
 4229
 4230 TMAX_RELTIM 相対時間に指定できる最大値
 4231
 4232 2. 15 カーネル共通定義
 4233
 4234 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。
 4235
 4236 2. 15. 1 カーネルヘッダファイル
 4237
 4238 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）
 4239 およびそこからインクルードされるファイルに含まれている【NGKI0507】．カー
 4240 ネルを用いる場合には、カーネルヘッダファイルをインクルードする
 4241 【NGKI0508】．
 4242
 4243 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
 4244 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）
 4245 に含まれる【NGKI0509】．具体的には、登録できるオブジェクトの数
 4246 （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
 4247 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
 4248 ドする【NGKI0510】．
 4249
 4250 μ ITRON4.0仕様に規定されており、この仕様で廃止されたデータ型および定数

4251 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
4252 する【NGKI0511】.

4253

4254 【μ ITRON4.0仕様との関係】

4255

4256 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
4257 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
4258 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
4259 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
4260 化ヘッダファイル (kernel_cfg.h) に変更した。

4261

4262 2.15.2 カーネル共通定数

4263

4264 (1) オブジェクト属性【NGKI0512】

4265

4266 TA_TPRI 0x01U タスクの待ち行列をタスクの優先度順に

4267

4268 【μ ITRON4.0仕様との関係】

4269

4270 値が0のオブジェクト属性 (TA_HLNG, TA_TFIFO, TA_MFIFO, TA_WSGI) は、デフォ
4271 ルトの扱いにして廃止した。これは、「(tskatr & TA_HLNG) != 0U」のような
4272 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。
4273 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
4274 した。

4275

4276 (2) 保護ドメインID【NGKI0513】

4277

4278 TDOM_SELF 0 自タスクの属する保護ドメイン

4279 TDOM_KERNEL -1 カーネルドメイン

4280 TDOM_NONE -2 無所属 (保護ドメインに属さない)

4281

4282 (3) その他のカーネル共通定数【NGKI0514】

4283

4284 TCLS_SELF 0 自タスクの属するクラス

4285

4286 TPRC_NONE 0 割付けプロセッサの指定がない

4287 TPRC_INI 0 初期割付けプロセッサ

4288

4289 TSK_SELF 0 自タスク指定

4290 TSK_NONE 0 該当するタスクがない

4291

4292 TPRI_SELF 0 自タスクのベース優先度の指定

4293 TPRI_INI 0 タスクの起動時優先度の指定

4294

4295 TIPM_ENAALL 0 割込み優先度マスク全解除

4296

4297 (4) カーネルで用いるメインエラーコード

4298

4299 「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
4300 で、E_CLS, E_WBLK, E_BOVRの3つは、カーネルでは使用しない【NGKI0515】.

4301
4302 【TOPPERS/ASPカーネルにおける規定】
4303
4304 ASPカーネルでは、サービスコールから、E_RSFN, E_RSATR, E_MACV, E_OACV,
4305 E_NOMEM, E_NOID, E_NORES, E_NOEXSが返る状況は起こらない【ASPS0011】。
4306 E_RSATRは、コンフィギュレータによって検出される【ASPS0012】。ただし、動
4307 的生成機能拡張パッケージでは、サービスコールから、E_RSATR, E_NOMEM,
4308 E_NOID, E_NOEXSが返る状況が起こる【ASPS0013】。
4309
4310 【TOPPERS/FMPカーネルにおける規定】
4311
4312 FMPカーネルでは、サービスコールから、E_RSFN, E_RSATR, E_MACV, E_OACV,
4313 E_NOMEM, E_NOID, E_NORES, E_NOEXSが返る状況は起こらない【FMPS0007】。
4314 E_RSATRとE_NORESは、コンフィギュレータによって検出される【FMPS0008】。
4315
4316 【TOPPERS/HRP2カーネルにおける規定】
4317
4318 HRP2カーネルでは、サービスコールから、E_RSATR, E_NOMEM, E_NOID,
4319 E_NORES, E_NOEXSが返る状況は起こらない【HRPS0006】。E_RSATRは、コンフィ
4320 ギュレータによって検出される【HRPS0007】。ただし、動的生成機能拡張パッ
4321 ケージでは、サービスコールから、E_RSATR, E_NOMEM, E_NOID, E_NOEXSが返る
4322 状況が起こる【HRPS0011】。
4323
4324 【TOPPERS/SSPカーネルにおける規定】
4325
4326 SSPカーネルでは、サービスコールから、E_RSFN, E_RSATR, E_MACV, E_OACV,
4327 E_ILUSE, E_NOMEM, E_NOID, E_NORES, E_NOEXS, E_RLWAI, E_TMOUT, E_DLTが返
4328 る状況は起こらない【SSPS0008】。E_RSATRは、コンフィギュレータによって検
4329 出される【SSPS0009】。
4330
4331 2. 15. 3 カーネル共通マクロ
4332
4333 (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
4334
4335 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
4336 用意している【NGKI0516】。
4337
4338 STK_T スタック領域を確保するためのデータ型
4339
4340 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
4341 STK_T型の配列の要素数
4342 ROUND_STK_T(sz) 要素数COUNT_STK_T(sz)のSTK_T型の配列のサイズ (sz
4343 を、STK_T型のサイズの倍数になるように大きい方に
4344 丸めた値)
4345
4346 これらを用いてスタック領域を確保する方法は次の通り【NGKI0517】。
4347
4348 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>)];
4349
4350 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

4351 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
4352 サイズにROUND_STK_T(<スタック領域のサイズ>)を指定する【NGKI0518】。

4353

4354 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
4355 タック領域を確保することはできない【NGKI0519】。詳しくは、「4.1 タスク
4356 管理機能」の節のCRE_TSKの機能の項を参照すること。

4357

4358 (2) オブジェクト属性を作るマクロ

4359

4360 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
4361 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

4362 【NGKI0520】。

4363

4364 TA_DOM(domid) domidで指定される保護ドメインに属する

4365

4366 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
4367 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

4368 【NGKI0521】。

4369

4370 TA_CLS(clsid) clsidで指定されるクラスに属する

4371

4372 (3) サービスコールの呼出し方法を指定するマクロ

4373

4374 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
4375 クロとして、次のマクロを用意している【NGKI0522】。

4376

4377 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
4378 て呼び出すための名称

4379

4380 2.15.4 カーネル共通構成マクロ

4381

4382 (1) サポートする機能【NGKI0523】

4383

4384	TOPPERS_SUPPORT_PROTECT	保護機能対応のカーネル
4385	TOPPERS_SUPPORT_MULTI_PRC	マルチプロセッサ対応のカーネル
4386	TOPPERS_SUPPORT_DYNAMIC_CRE	動的生成対応のカーネル

4387

4388 【未決定事項】

4389

4390 マクロ名は、今後変更する可能性がある。

4391

4392 (2) 優先度の範囲【NGKI0524】

4393

4394 TMIN_TPRI タスク優先度の最小値 (=1)

4395 TMAX_TPRI タスク優先度の最大値

4396

4397 【TOPPERS/ASPカーネルにおける規定】

4398

4399 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

4400 【ASPS0014】。ただし、タスク優先度拡張パッケージを用いると、TMAX_TPRIを

4401 256に拡張することができる【ASPS0015】.

4402

4403 【TOPPERS/FMPカーネルにおける規定】

4404

4405 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

4406 【FMPS0009】.

4407

4408 【TOPPERS/HRP2カーネルにおける規定】

4409

4410 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

4411 【HRPS0008】.

4412

4413 【TOPPERS/SSPカーネルにおける規定】

4414

4415 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

4416 【SSPS0010】.

4417

4418 【 μ ITRON4.0仕様との関係】

4419

4420 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

4421 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.

4422

4423 (3) プロセッサの数

4424

4425 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するためのマクロとし

4426 て、次の構成マクロを用意している【NGKI0525】.

4427

4428 TNUM_PRCID プロセッサの数

4429

4430 (4) 特殊な役割を持ったプロセッサ

4431

4432 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

4433 めのマクロとして、次の構成マクロを用意している【NGKI0526】.

4434

4435 TOPPERS_MASTER_PRCID マスタプロセッサのID番号

4436 TOPPERS_SYSTIM_PRCID システム時刻管理プロセッサのID番号 (グ

4437 ローバルタイマ方式の場合のみ)

4438

4439 (5) タイマ方式

4440

4441 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

4442 として、次の構成マクロを用意している【NGKI0527】.

4443

4444 TOPPERS_SYSTIM_LOCAL ローカルタイマ方式の場合にマクロ定義

4445 TOPPERS_SYSTIM_GLOBAL グローバルタイマ方式の場合にマクロ定義

4446

4447 (6) バージョン情報【NGKI0528】

4448

4449 TKERNEL_MAKER カーネルのメーカーコード (=0x0118)

4450 TKERNEL_PRID カーネルの識別番号

4451 TKERNEL_SPVER カーネル仕様のバージョン番号
4452 TKERNEL_PRVER カーネルのバージョン番号

4453

4454 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
4455 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。

4456

4457 カーネルの識別番号 (TKERNEL_PRID) は、TOPPERSカーネルの種類を表す。

4458

4459	0x0001	TOPPERS/JSPカーネル
4460	0x0002	予約 (IIMPカーネル)
4461	0x0003	予約 (IDLカーネル)
4462	0x0004	TOPPERS/FI4カーネル
4463	0x0005	TOPPERS/FDMPカーネル
4464	0x0006	TOPPERS/HRPカーネル
4465	0x0007	TOPPERS/ASPカーネル
4466	0x0008	TOPPERS/FMPカーネル
4467	0x0009	TOPPERS/SSPカーネル
4468	0x000a	TOPPERS/ASP Safetyカーネル

4469

4470 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
4471 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
4472 番号、下位4ビットがマイナーバージョン番号を表す。

4473

4474 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
4475 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
4476 ベルを表す。

4477

4478

4479 第3章 システムインタフェースレイヤAPI仕様

4480

4481 3.1 システムインタフェースレイヤの概要

4482

4483 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
4484 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
4485 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
4486 トにおいて修正を加えて用いている。

4487

4488 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
4489 ことを想定している【NGKI0801】。非特権モードで実行されているプログラム
4490 からSILの機能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
4491 【NGKI0802】。

4492

- 4493 ・微少時間待ちの機能呼び出すこと
- 4494 ・エンディアンの取得のためのマクロを参照すること
- 4495 ・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
4496 スすること
- 4497 ・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
4498 こと

4499

4500 3.2 SILヘッダファイル

4501
4502 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
4503 インクルードされるファイルに含まれている【NGKI0803】。SILを用いる場合に
4504 は、SILヘッダファイルをインクルードする【NGKI0804】。
4505
4506 3.3 全割込みロック状態の制御
4507
4508 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)
4509 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
4510 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)
4511 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。
4512
4513 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
4514 ための以下の機能を用意している。
4515
4516 (1) SIL_PRE_LOC
4517
4518 全割込みロック状態の制御に必要な変数を宣言するマクロ【NGKI0805】。通常
4519 は、型と変数名を並べたもので、最後に";"を含まない。
4520
4521 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
4522 の変数宣言部に記述しなければならない【NGKI0806】。SIL_LOC_INT、
4523 SIL_UNL_INTを1つの関数内でネストして用いることは可能であるが、その場合
4524 には、ネストレベル毎にブロックを作り、そのブロックの先頭の変数宣言部に
4525 SIL_PRE_LOCを記述しなければならない【NGKI0807】。そのように記述しなかつ
4526 た場合の動作は保証されない【NGKI0808】。
4527
4528 (2) SIL_LOC_INT()
4529
4530 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
4531 し、全割込みロック状態に遷移する【NGKI0809】。
4532
4533 (3) SIL_UNL_INT()
4534
4535 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
4536 【NGKI0810】。SIL_LOC_INTを実行せずにSIL_UNL_INTを呼び出した場合の動作
4537 は保証されない【NGKI0811】。
4538
4539 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
4540 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
4541 ること。
4542
4543 【補足説明】
4544
4545 全割込みロック状態の制御機能の使用例は次の通り。
4546
4547 {
4548 SIL_PRE_LOC;
4549
4550 SIL_LOC_INT();

```
4551         // この間はNMIを除くすべての割込みがマスクされる.  
4552         // 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).  
4553         SIL_UNL_INT();  
4554     }
```

4556 3.4 SILスピンのロック

4557
4558 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
4559 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
4560 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のロックの機
4561 能を用意している。これを、カーネルのスピンのロック機能と区別するために、
4562 SILスピンのロックと呼ぶ。

4563
4564 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
4565 全ての割込み（NMIを除く）がマスクされる【NGKI0812】。ロックが他のプロセッ
4566 サに取得されている場合には、ロックが取得できるまでループによって待つ
4567 【NGKI0813】。ロックの取得を待つ間は、割込みはマスクされない（ロックの
4568 取得を試みる前にマスクしていた割込みは、マスク解除されない）
4569 【NGKI0814】。プロセッサ間ロックを取得し割込みをマスクすることを、SILス
4570 ピンのロックを取得するという。また、プロセッサ間ロックを返却し割込みをマ
4571 スク解除することを、SILスピンのロックを返却するという。

4572
4573 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
4574 【NGKI0815】。

4576 (1) SIL_PRE_LOC

4577
4578 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
4579 ンのロックの取得・解放にも兼用する【NGKI0816】。

4580
4581 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
4582 の変数宣言部に記述しなければならない【NGKI0817】。SIL_LOC_SPN、
4583 SIL_UNL_SPNを、同じ関数内のSIL_LOC_INT、SIL_UNL_INTとネストして用いるこ
4584 とは可能であるが、その場合には、ネストレベル毎にブロックを作り、そのブ
4585 ロックの先頭の変数宣言部にSIL_PRE_LOCを記述しなければならない
4586 【NGKI0818】。そのように記述しなかった場合の動作は保証されない
4587 【NGKI0819】。

4588 (2) SIL_LOC_SPN()

4589
4590
4591 SILスピンの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
4592 ックの取得を試みる【NGKI0820】。ロックが他のプロセッサに取得されている状態
4593 である場合や、他のプロセッサがロックの取得に成功した場合には、ロックが
4594 返却されるまでループによって待ち、返却されたらロックの取得を試みる
4595 【NGKI0821】。ロックの取得に成功した場合には、全割込みロックフラグをセッ
4596 トし、全割込みロック状態に遷移する【NGKI0822】。

4597 (3) SIL_UNL_SPN()

4598
4599
4600 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

4601 実行する前の状態に戻す【NGKI0823】。

4602

4603 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
4604 保証されない【NGKI0824】。逆に、SILスピンロックを取得していない状態で
4605 SIL_UNL_SPNを呼び出した場合の動作も保証されない【NGKI0825】。

4606

4607 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
4608 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
4609 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。

4610

4611 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
4612 されていない【NGKI0826】。

4613

4614 【使用上の注意】

4615

4616 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、
4617 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
4618 べきである。

4619

4620 【補足説明】

4621

4622 SILスピンロック機能の使用例は次の通り。

4623

```
4624 {  
4625     SIL_PRE_LOC;  
4626  
4627     SIL_LOC_SPN();  
4628     // この間はSILスピンロックを取得している。  
4629     // この間はNMIを除くすべての割込みがマスクされる。  
4630     // 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）。  
4631     SIL_UNL_SPN();  
4632 }
```

4633

4634 3.5 微少時間待ち

4635

4636 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
4637 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、
4638 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
4639 ための以下の機能を用意している。

4640

4641 (1) void sil_dly_nse(ulong_t dlytim)

4642

4643 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
4644 【NGKI0827】。指定した値によっては、指定した時間よりもかなり長く待つ場
4645 合があるので注意すること。

4646

4647 3.6 エンディアンの取得

4648

4649 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
4650 下のマクロを定義している。

4651
4652 (1) SIL_ENDIAN_BIG, SIL_ENDIAN_LITTLE
4653
4654 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
4655 プロセッサではSIL_ENDIAN_LITTLEを、マクロ定義している【NGKI0828】。
4656
4657 3.7 メモリ空間アクセス関数
4658
4659 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
4660 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。
4661
4662 (1) uint8_t sil_reb_mem(const uint8_t *mem)
4663
4664 memで指定されるアドレスから8ビット単位で読み出した値を返す【NGKI0829】。
4665
4666 (2) void sil_wrb_mem(uint8_t *mem, uint8_t data)
4667
4668 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
4669 【NGKI0830】。
4670
4671 (3) uint16_t sil_reh_mem(const uint16_t *mem)
4672
4673 memで指定されるアドレスから16ビット単位で読み出した値を返す【NGKI0831】。
4674
4675 (4) void sil_wrh_mem(uint16_t *mem, uint16_t data)
4676
4677 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
4678 【NGKI0832】。
4679
4680 (5) uint16_t sil_reh_lem(const uint16_t *mem)
4681
4682 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
4683 値を返す【NGKI0833】。リトルエンディアンプロセッサでは、sil_reh_memと一
4684 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
4685 アン変換した値を返す。
4686
4687 (6) void sil_wrh_lem(uint16_t *mem, uint16_t data)
4688
4689 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
4690 アンで書き込む【NGKI0834】。リトルエンディアンプロセッサでは、
4691 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
4692 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。
4693
4694 (7) uint16_t sil_reh_bem(const uint16_t *mem)
4695
4696 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
4697 値を返す【NGKI0835】。ビッグエンディアンプロセッサでは、sil_reh_memと一
4698 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
4699 アン変換した値を返す。
4700

```

4701     (8) void sil_wrh_bem(uint16_t *mem, uint16_t data)
4702
4703     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
4704     アンで書き込む【NGKI0836】． ビッグエンディアンプロセッサでは,
4705     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
4706     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.
4707
4708     (9) uint32_t sil_rew_mem(const uint32_t *mem)
4709
4710     memで指定されるアドレスから32ビット単位で読み出した値を返す【NGKI0837】 .
4711
4712     (10) void sil_wrw_mem(uint32_t *mem, uint32_t data)
4713
4714     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
4715     【NGKI0838】 .
4716
4717     (11) uint32_t sil_rew_lem(const uint32_t *mem)
4718
4719     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
4720     値を返す【NGKI0839】． リトルエンディアンプロセッサでは, sil_rew_memと一
4721     致する． ビッグエンディアンプロセッサでは, sil_rew_memが返す値を, エンディ
4722     アン変換した値を返す.
4723
4724     (12) void sil_wrw_lem(uint32_t *mem, uint32_t data)
4725
4726     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
4727     アンで書き込む【NGKI0840】． リトルエンディアンプロセッサでは,
4728     sil_wrw_memと一致する． ビッグエンディアンプロセッサでは, dataをエンディ
4729     アン変換した値を, sil_wrw_memで書き込むのと同じ結果となる.
4730
4731     (13) uint32_t sil_rew_bem(const uint32_t *mem)
4732
4733     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
4734     値を返す【NGKI0841】． ビッグエンディアンプロセッサでは, sil_rew_memと一
4735     致する． リトルエンディアンプロセッサでは, sil_rew_memが返す値を, エンディ
4736     アン変換した値を返す.
4737
4738     (14) void sil_wrw_bem(uint32_t *mem, uint32_t data)
4739
4740     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
4741     アンで書き込む【NGKI0842】． ビッグエンディアンプロセッサでは,
4742     sil_wrw_memと一致する． リトルエンディアンプロセッサでは, dataをエンディ
4743     アン変換した値を, sil_wrw_memで書き込むのと同じ結果となる.
4744
4745     3.8 I/O空間アクセス関数
4746
4747     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
4748     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
4749     用意している【NGKI0843】 .
4750

```

4751 (1) uint8_t sil_reb_iop(const uint8_t *iop)
4752 (2) void sil_wrb_iop(uint8_t *iop, uint8_t data)
4753 (3) uint16_t sil_reh_iop(const uint16_t *iop)
4754 (4) void sil_wrh_iop(uint16_t *iop, uint16_t data)
4755 (5) uint16_t sil_reh_lep(const uint16_t *iop)
4756 (6) void sil_wrh_lep(uint16_t *iop, uint16_t data)
4757 (7) uint16_t sil_reh_bep(const uint16_t *iop)
4758 (8) void sil_wrh_bep(uint16_t *iop, uint16_t data)
4759 (9) uint32_t sil_rew_iop(const uint32_t *iop)
4760 (10) void sil_wrw_iop(uint32_t *iop, uint32_t data)
4761 (11) uint32_t sil_rew_lep(const uint32_t *iop)
4762 (12) void sil_wrw_lep(uint32_t *iop, uint32_t data)
4763 (13) uint32_t sil_rew_bep(const uint32_t *iop)
4764 (14) void sil_wrw_bep(uint32_t *iop, uint32_t data)

4765

4766 3.9 プロセッサIDの参照

4767

4768 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
4769 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。

4770

4771 (1) void sil_get_pid(ID *p_prcid)

4772

4773 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、
4774 p_prcidで指定したメモリ領域に返す【NGKI0844】。

4775

4776 【使用上の注意】

4777

4778 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
4779 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
4780 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、
4781 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
4782 ている可能性があるためである。

4783

4784

4785 第4章 カーネルAPI仕様

4786

4787 この章では、カーネルのAPI仕様について規定する。

4788

4789 【 μ ITRON4.0仕様との関係】

4790

4791 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
4792 らの変更については、個別のAPI仕様では記述しない。

4793

4794 INT \rightarrow int_t

4795 UINT \rightarrow uint_t

4796 VP \rightarrow void *

4797 VP_INT \rightarrow intptr_t

4798

4799 【 μ ITRON4.0/PX仕様との関係】

4800

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

4.1 タスク管理機能

タスクは、プログラムの並行実行の単位で、カーネルが実行を制御する処理単位である。タスクは、タスクIDと呼ぶID番号によって識別する【NGKI1001】。

タスク管理機能に関連して、各タスクが持つ情報は次の通り【NGKI1002】。

- ・タスク属性
- ・タスク状態
- ・ベース優先度
- ・現在優先度
- ・起動要求キューイング数
- ・割付けプロセッサ (マルチプロセッサ対応カーネルの場合)
- ・次回起動時の割付けプロセッサ (マルチプロセッサ対応カーネルの場合)
- ・拡張情報
- ・メインルーチンの先頭番地
- ・起動時優先度
- ・実行時優先度 (TOPPERS/SSPカーネルの場合)
- ・スタック領域
- ・システムスタック領域 (保護機能対応カーネルの場合)
- ・アクセス許可ベクタ (保護機能対応カーネルの場合)
- ・属する保護ドメイン (保護機能対応カーネルの場合)
- ・属するクラス (マルチプロセッサ対応カーネルの場合)

タスクのベース優先度は、タスクの現在優先度を決定するために使われる優先度であり、タスクの起動時に起動時優先度に初期化される【NGKI1003】。

タスクの現在優先度は、タスクの実行順位を決定するために使われる優先度である。単にタスクの優先度と言った場合には、現在優先度のことを指す。タスクがミューテックスをロックしていない間は、タスクの現在優先度はベース優先度に一致する【NGKI1004】。ミューテックスをロックしている間のタスクの現在優先度については、「4.4.6 ミューテックス」の節を参照すること。

タスクの起動要求キューイング数は、処理されていないタスクの起動要求の数であり、タスクの生成時に0に初期化される【NGKI1005】。

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

次回起動時の割付けプロセッサは、マルチプロセッサ対応カーネルにおいて、タスクが次に起動される時に割り付けられるプロセッサで、タスクの生成時に未設定の状態に初期化される【NGKI1007】。タスクの起動時に、次回起動時の割付けプロセッサが設定されていれば、タスクの割付けプロセッサがそのプロ

4851 セッサに変更され、次回起動時の割付けプロセッサは未設定の状態に戻される
4852 【NGKI1008】。次回起動時の割付けプロセッサが未設定の場合には、タスクの
4853 割付けプロセッサは変更されない（つまり、タスクが前に実行されていたのと
4854 同じプロセッサで実行される）【NGKI1009】。

4855
4856 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
4857 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
4858 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
4859 ぶ【NGKI1010】。その扱いについては、「2.11.6 ユーザタスクのユーザスタ
4860 ック領域」の節を参照すること。システムタスクのスタック領域は、カーネルの
4861 用いるオブジェクト管理領域と同様に扱われる【NGKI1011】。

4862
4863 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
4864 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
4865 間に用いるスタック領域である【NGKI1012】。システムスタック領域は、カー
4866 ネルの用いるオブジェクト管理領域と同様に扱われる【NGKI1013】。

4867
4868 タスク属性には、次の属性を指定することができる【NGKI1014】。

4869
4870 TA_ACT 0x02U タスクの生成時にタスクを起動する
4871 TA_RSTR 0x04U 生成するタスクを制約タスクとする

4872
4873 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
4874 【NGKI1015】。また、ターゲットによっては、ターゲット定義のタスク属性を
4875 指定できる場合がある【NGKI1016】。ターゲット定義のタスク属性として、次
4876 の属性を予約している【NGKI1017】。

4877
4878 TA_FPU FPUレジスタをコンテキストに含める

4879
4880 タスク終了時には、次の処理が行われる。まず、終了するタスク（対象タスク）
4881 に対してタスク終了時に行うべきその他の処理が行われた後、対象タスクは休
4882 止状態になる【NGKI1178】。対象タスクの起動要求キューイング数が0でない場
4883 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
4884 クは実行できる状態になる【NGKI1179】。またこの時、起動要求キューイング
4885 数から1が減ぜられる【NGKI1180】。

4886
4887 C言語によるタスクの記述形式は次の通り【NGKI1018】。

4888
4889 void task(intptr_t exinf)
4890 {
4891 タスク本体
4892 ext_tsk();
4893 }

4894
4895 exinfには、タスクの拡張情報が渡される【NGKI1019】。ext_tskを呼び出さず、
4896 タスクのメインルーチンからリターンした場合、ext_tskを呼び出した場合と同
4897 じ動作をする【NGKI1020】。

4898
4899 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。

4900

4901 TMAX_ACTCNT タスクの起動要求キューイング数の最大値【NGKI1021】
4902
4903 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
4904 は、静的APIによって登録されたタスクの数に一致）
4905 【NGKI1022】
4906
4907 【TOPPERS/ASPカーネルにおける規定】
4908
4909 ASPカーネルでは、TMAX_ACTCNTは1に固定されている【ASPS0101】。また、制約
4910 タスクはサポートしていない【ASPS0102】。ただし、制約タスク拡張パッケー
4911 ジを用いると、制約タスクの機能を追加することができる【ASPS0103】。
4912
4913 【TOPPERS/FMPカーネルにおける規定】
4914
4915 FMPカーネルでは、TMAX_ACTCNTは1に固定されている【FMPS0101】。また、制約
4916 タスクはサポートしていない【FMPS0102】。
4917
4918 【TOPPERS/HRP2カーネルにおける規定】
4919
4920 HRP2カーネルでは、TMAX_ACTCNTは1に固定されている【HRPS0101】。また、制
4921 約タスクはサポートしていない【HRPS0102】。
4922
4923 【TOPPERS/SSPカーネルにおける規定】
4924
4925 SSPカーネルでは、TMAX_ACTCNTは1に固定されている【SSPS0101】。
4926
4927 SSPカーネルは、制約タスクのみをサポートすることから、すべてのタスクでス
4928 タック領域を共有しており、タスク毎にスタック領域の情報を持たない
4929 【SSPS0102】。
4930
4931 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
4932 持つ【SSPS0103】。SSPカーネルにおいては、タスクが起動された後、最初に実
4933 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
4934 れる【SSPS0104】。実行時優先度の機能は、起動時優先度よりも高い優先度で
4935 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
4936 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。
4937
4938 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
4939 て設定する【SSPS0105】。実行時優先度を定義しない場合、タスクの実行時優
4940 先度は、起動時優先度と同じ値に設定される【SSPS0106】。
4941
4942 〔実行時優先度によるスタック領域の節約〕
4943
4944 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
4945 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、
4946 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
4947 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。
4948
4949 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
4950 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

5001	ER_ID	tskid	生成されたタスクのID番号（正の値）またはエラーコード
5002			
5003			
5004	【エラーコード】		
5005	E_CTX	コンテキストエラー	
5006		・非タスクコンテキストからの呼出し [s]	【NGKI1026】
5007		・CPUロック状態からの呼出し [s]	【NGKI1027】
5008	E_RSATR	予約属性	
5009		・tskatrが無効	【NGKI1028】
5010		・属する保護ドメインの指定が有効範囲外または無所属 [sP]	
5011		【NGKI1029】	
5012		・保護ドメインの囲みの中に記述されていない [SP]	【NGKI1030】
5013		・属するクラスの指定が有効範囲外 [sM]	【NGKI1031】
5014		・クラスの囲みの中に記述されていない [SM]	【NGKI1032】
5015	E_PAR	パラメータエラー	
5016		・taskがプログラムの先頭番地として正しくない	【NGKI1033】
5017		・itskpriが有効範囲外	【NGKI1034】
5018		・その他の条件については機能の項を参照	
5019	E_OACV	オブジェクトアクセス違反	
5020		・システム状態に対する管理操作が許可されていない [sP]	
5021		【NGKI1035】	
5022	E_MACV	メモリアクセス違反	
5023		・pk_ctskが指すメモリ領域への読出しアクセスが許可されていない [sP]	【NGKI1036】
5024			
5025	E_NOID	ID番号不足	
5026		・割り付けられるタスクIDがない [sD]	【NGKI1037】
5027	E_NOMEM	メモリ不足	
5028		・スタック領域が確保できない	【NGKI1038】
5029		・システムスタック領域が確保できない [P]	【NGKI1039】
5030	E_OBJ	オブジェクト状態エラー	
5031		・tskidで指定したタスクが登録済み（CRE_TSKの場合）	【NGKI1040】
5032		・その他の条件については機能の項を参照	
5033			
5034	【機能】		
5035			
5036	各パラメータで指定したタスク生成情報に従って、タスクを生成する。具体的な振舞いは以下の通り。		
5037			
5038			
5039	まず、stkとstkszからタスクが用いるスタック領域が設定される【NGKI1041】。		
5040	ただし、保護機能対応カーネルで、生成するタスクがシステムタスクの場合には、		
5041	スタック領域の設定にsstkszも用いられる。stkszに0以下の値を指定した時や、		
5042	設定されるスタック領域のサイズがターゲット定義の最小値よりも小さくなる時には、E_PARエラーとなる【NGKI1042】。		
5043			
5044			
5045	また、保護機能対応カーネルで、生成するタスクがユーザタスクの場合には、		
5046	sstkとsstkszからシステムスタック領域が設定される【NGKI1043】。この場合、		
5047	sstkszに0以下の値を指定した時や、ターゲット定義の最小値よりも小さい値を		
5048	指定した時には、E_PARエラーとなる【NGKI1044】。		
5049			
5050	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、		

5051 生成されたタスクは休止状態になる【NGKI1045】。さらに、tskatrにTA_ACTを
5052 指定した場合には、タスク起動時に行うべき初期化処理が行われ、生成された
5053 タスクは実行できる状態になる【NGKI1046】。

5054

5055 静的APIにおいては、tskidはオブジェクト識別名、tskatr, itskpri, stkszは
5056 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
5057 【NGKI1047】。コンフィギュレータは、静的APIのメモリ不足 (E_NOMEM) エラー
5058 を検出することができない【NGKI1048】。

5059

5060 [stkにNULLを指定した場合]

5061

5062 stkをNULLとした場合、stkszで指定したサイズのスタック領域が、コンフィギュ
5063 レータまたはカーネルにより確保される【NGKI1049】。stkszにターゲット定義
5064 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
5065 るように大きい方に丸めたサイズで確保される【NGKI1050】。

5066

5067 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
5068 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
5069 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
5070 て、メモリオブジェクトとしてカーネルに登録される【NGKI1051】。

5071

5072 静的APIにより制約タスクを生成する場合（tskatrにTA_RSTRを指定して生成す
5073 る場合）、スタック領域は、制約タスクの起動時優先度毎に確保され、同じ起
5074 動時優先度を持つ制約タスクで共有される【NGKI1052】。確保されるスタック
5075 領域のサイズは、それを共有する制約タスクのスタック領域のサイズ（stksz）
5076 の最大値となる【NGKI1053】。マルチプロセッサ対応カーネルでは、以上のス
5077 タック領域の確保処理を、制約タスクの初期割付けプロセッサ毎に行う
5078 【NGKI1054】。

5079

5080 [stkにNULL以外を指定した場合]

5081

5082 stkにNULL以外を指定した場合、stkとstkszで指定したスタック領域は、アプリ
5083 ケーションで確保しておく必要がある【NGKI1055】。スタック領域をアプリケー
5084 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
5085 すること。その方法に従わず、stkやstkszにターゲット定義の制約に合致しな
5086 い先頭番地やサイズを指定した時には、E_PARエラーとなる【NGKI1056】。

5087

5088 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、
5089 stkとstkszで指定したスタック領域がカーネル専用のメモリオブジェクトに含
5090 まれない場合、E_OBJエラーとなる【NGKI1057】。

5091

5092 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
5093 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
5094 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
5095 カーネルに登録される【NGKI1058】。そのため、上の方法を用いてスタック領
5096 域を確保しても、ターゲット定義の制約に合致する先頭番地とサイズとなると
5097 は限らず、スタック領域をアプリケーションで確保する方法は、ターゲット定
5098 義である【NGKI1059】。また、stkとstkszで指定したスタック領域が、登録済
5099 みのメモリオブジェクトとメモリ領域が重なる場合には、E_OBJエラーとなる
5100 【NGKI1060】。

5101
5102 [sstkとsstkszsの扱い]
5103
5104 保護機能対応カーネルにおけるsstkとsstkszsの扱いは、生成するタスクがユー
5105 ザタスクの場合とシステムタスクの場合で異なる。
5106
5107 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。
5108
5109 sstkの記述を省略するか、sstkをNULLとした場合、sstkszsで指定したサイズの
5110 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
5111 る【NGKI1061】。sstkszsにターゲット定義の制約に合致しないサイズを指定し
5112 た時には、ターゲット定義の制約に合致するように大きい方に丸めたサイズで
5113 確保される【NGKI1062】。sstkszsの記述も省略した場合には、ターゲット定義
5114 のデフォルトのサイズで確保される【NGKI1063】。
5115
5116 sstkにNULL以外を指定した場合、sstkとsstkszsで指定したスタック領域は、ア
5117 プリケーションで確保しておく必要がある【NGKI1064】。スタック領域をアプ
5118 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
5119 を参照すること。その方法に従わず、sstkやsstkszsにターゲット定義の制約に
5120 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
5121 【NGKI1065】。また、stkとstkszsで指定したシステムスタック領域がカーネル
5122 専用のメモリオブジェクトに含まれない場合、E_OBJエラーとなる【NGKI1066】。
5123
5124 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。
5125
5126 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
5127 た場合には、E_PARエラーとなる【NGKI1068】。
5128
5129 sstkszs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
5130 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、sstkszsが加えら
5131 れる【NGKI1069】。stkszsにsstkszsを加えた値が、ターゲット定義の制約に合致
5132 しないサイズになる時には、ターゲット定義の制約に合致するように大きい方
5133 に丸めたサイズで確保される【NGKI1070】。
5134
5135 sstkszs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
5136 となる【NGKI1071】。
5137
5138 sstkszs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
5139 【NGKI1072】。
5140
5141 【TOPPERS/ASPカーネルにおける規定】
5142
5143 ASPカーネルでは、CRE_TSKのみをサポートする【ASPS0104】。ただし、動的生
5144 成機能拡張パッケージでは、acre_tskもサポートする【ASPS0105】。
5145
5146 【TOPPERS/FMPカーネルにおける規定】
5147
5148 FMPカーネルでは、CRE_TSKのみをサポートする【FMPS0103】。
5149
5150 【TOPPERS/HRP2カーネルにおける規定】

5151
5152 HRP2カーネルでは、CRE_TSKのみをサポートする【HRPS0103】。
5153
5154 動的生成機能拡張パッケージでは、acre_tskもサポートする【HRPS0175】。た
5155 だし、生成するタスクがユーザタスクの場合、stkにNULLが指定されるとカーネ
5156 ルがスタック領域を確保する機能はサポートしない。stkにNULLを指定した場合
5157 には、E_NOSPTエラーとなる【HRPS0176】。
5158
5159 【TOPPERS/SSPカーネルにおける規定】
5160
5161 SSPカーネルでは、CRE_TSKのみをサポートする【SSPS0107】。
5162
5163 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
5164 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
5165 【SSPS0109】。
5166
5167 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
5168 を指定しない場合でも、生成されるタスクは制約タスクとなる【SSPS0110】。
5169
5170 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
5171 フィギュレータはタスクのスタック領域を確保しない【SSPS0111】。これは、
5172 SSPカーネルでは、すべての処理単位が共有スタック領域を使用し、タスク毎に
5173 スタック領域を持たないためである。stkにNULL以外を指定した場合には、
5174 E_PARエラーとなる【SSPS0112】。
5175
5176 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。
5177
5178 【μITRON4.0仕様との関係】
5179
5180 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
5181 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
5182 した。
5183
5184 【μITRON4.0/PX仕様との関係】
5185
5186 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
5187 で確保する方法を規定した。
5188
5189 【未決定事項】
5190
5191 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
5192 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。
5193
5194 【仕様決定の理由】
5195
5196 保護機能対応カーネルにおいて、sstkszおよびssstkの記述は省略することがで
5197 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
5198 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
5199 るためである。
5200 -----

5201 AID_TSK 割付け可能なタスクIDの数の指定〔SD〕【NGKI1073】

5202

5203 【静的API】

5204 AID_TSK(uint_t notsk)

5205

5206 【パラメータ】

5207 uint_t notsk 割付け可能なタスクIDの数

5208

5209 【エラーコード】

5210 E_RSATR 予約属性

5211 ・保護ドメインの囲みの中に記述されている〔P〕【NGKI3428】

5212 ・クラスの囲みの中に記述されていない〔M〕【NGKI1075】

5213 E_PAR パラメータエラー

5214 ・notskが負の値【NGKI3276】

5215

5216 【機能】

5217

5218 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

5219 付け可能なタスクIDとして確保する【NGKI1076】。

5220

5221 notskは整数定数式パラメータである【NGKI1077】。

5222

5223 【TOPPERS/ASPカーネルにおける規定】

5224

5225 ASPカーネルの動的生成機能拡張パッケージでは、AID_TSKをサポートする

5226 【ASPS0210】。

5227

5228 【TOPPERS/HRP2カーネルにおける規定】

5229

5230 HRP2カーネルの動的生成機能拡張パッケージでは、AID_TSKをサポートする

5231 【HRPS0211】。

5232 -----

5233 SAC_TSK タスクのアクセス許可ベクタの設定〔SP〕【NGKI1078】

5234 sac_tsk タスクのアクセス許可ベクタの設定〔TPD〕【NGKI1079】

5235

5236 【静的API】

5237 SAC_TSK(ID tskid, { ACPTN acptn1, ACPTN acptn2,

5238 ACPTN acptn3, ACPTN acptn4 })

5239

5240 【C言語API】

5241 ER ercd = sac_tsk(ID tskid, const ACVCT *p_acvct)

5242

5243 【パラメータ】

5244 ID tskid 対象タスクのID番号

5245 ACVCT * p_acvct アクセス許可ベクタを入れたパケットへのポ

5246 インタ（静的APIを除く）

5247

5248 *アクセス許可ベクタ（パケットの内容）

5249 ACPTN acptn1 通常操作1のアクセス許可パターン

5250 ACPTN acptn2 通常操作2のアクセス許可パターン

5251	ACPTN	acptn3	管理操作のアクセス許可パターン
5252	ACPTN	acptn4	参照操作のアクセス許可パターン
5253			
5254	【リターンパラメータ】		
5255	ER	ercd	正常終了 (E_OK) またはエラーコード
5256			
5257	【エラーコード】		
5258	E_CTX	コンテキストエラー	
5259		・ 非タスクコンテキストからの呼出し [s] 【NGKI1080】	
5260		・ CPUロック状態からの呼出し [s] 【NGKI1081】	
5261	E_ID	不正ID番号	
5262		・ tskidが有効範囲外 [s] 【NGKI1082】	
5263	E_RSATR	予約属性	
5264		・ 対象タスクが属する保護ドメインの囲みの中に記述されて	
5265		いない [S] 【NGKI1083】	
5266		・ 対象タスクが属するクラスの囲みの中に記述されていない	
5267		[SM] 【NGKI1084】	
5268	E_NOEXS	オブジェクト未登録	
5269		・ 対象タスクが未登録 【NGKI1085】	
5270	E_OACV	オブジェクトアクセス違反	
5271		・ 対象タスクに対する管理操作が許可されていない [s] 【NGKI1086】	
5272	E_MACV	メモリアクセス違反	
5273		・ p_acvetが指すメモリ領域への読出しアクセスが許可されて	
5274		いない [s] 【NGKI1087】	
5275	E_OBJ	オブジェクト状態エラー	
5276		・ 対象タスクは静的APIで生成された [s] 【NGKI1088】	
5277		・ 対象タスクに対してアクセス許可ベクタが設定済み [S]	
5278		【NGKI1089】	
5279			
5280	【機能】		
5281			
5282	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		
5283	許可パターンの組) を, 各パラメータで指定した値に設定する 【NGKI1090】 .		
5284			
5285	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		
5286	式パラメータである 【NGKI1091】 .		
5287			
5288	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		
5289	となる 【NGKI1092】 .		
5290			
5291	【TOPPERS/HRP2カーネルにおける規定】		
5292			
5293	HRP2カーネルでは, SAC_TSKのみをサポートする 【HRPS0104】 . ただし, 動的生		
5294	成機能拡張パッケージでは, sac_tskもサポートする 【HRPS0177】 .		
5295	-----		
5296	DEF_EPR	タスクの実行時優先度の定義 [S] 【NGKI1093】	
5297			
5298	【静的API】		
5299	DEF_EPR(ID tskid, { PRI exePRI })		
5300			

5301 **【パラメータ】**
5302 ID tskid 対象タスクのID番号
5303 PRI exepri タスクの実行時優先度
5304

5305 **【エラーコード】**
5306 E_PAR パラメータエラー
5307 ・ exepriが有効範囲外【NGKI1094】
5308 E_ILUSE サービスコール不正使用
5309 ・ 条件については機能の項を参照
5310 E_OBJ オブジェクト状態エラー
5311 ・ 対象タスクに対して実行優先度が設定済み【NGKI1095】
5312

5313 **【サポートするカーネル】**
5314
5315 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
5316 ネルは、DEF_EPRをサポートしない【NGKI1096】。
5317

5318 **【機能】**
5319
5320 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
5321 先度に設定する【NGKI1097】。
5322
5323 tskidはオブジェクト識別名、exepriは整数定数式パラメータである【NGKI1098】。
5324
5325 exepriが、対象タスクの起動時優先度よりも低い場合には、E_ILUSEエラーとな
5326 る【NGKI1099】。
5327

5328 **【 μ ITRON4.0仕様との関係】**
5329
5330 μ ITRON4.0仕様に定義されていない静的APIである。
5331 -----

5332 del_tsk タスクの削除〔TD〕【NGKI1100】
5333

5334 **【C言語API】**
5335 ER ercd = del_tsk(ID tskid)
5336

5337 **【パラメータ】**
5338 ID tskid 対象タスクのID番号
5339

5340 **【リターンパラメータ】**
5341 ER ercd 正常終了（E_OK）またはエラーコード
5342

5343 **【エラーコード】**
5344 E_CTX コンテキストエラー
5345 ・ 非タスクコンテキストからの呼出し【NGKI1101】
5346 ・ CPUロック状態からの呼出し【NGKI1102】
5347 E_ID 不正ID番号
5348 ・ tskidが有効範囲外【NGKI1103】
5349 E_NOEXS オブジェクト未登録
5350 ・ 対象タスクが未登録【NGKI1104】

5351 E_OACV オブジェクトアクセス違反
5352 ・対象タスクに対する管理操作が許可されていない [P] 【NGKI1105】
5353 E_OBJ オブジェクト状態エラー
5354 ・対象タスクが休止状態でない 【NGKI1106】
5355 ・対象タスクは静的APIで生成された 【NGKI1107】
5356
5357 **【機能】**
5358
5359 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
5360 り．
5361
5362 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
5363 スクIDが未使用の状態に戻される 【NGKI1108】．また，タスクの生成時にタス
5364 クのスタック領域およびシステムスタック領域がカーネルによって確保された
5365 場合は，それらのメモリ領域が解放される 【NGKI1109】．
5366
5367 **【TOPPERS/ASPカーネルにおける規定】**
5368
5369 ASPカーネルでは，del_tskをサポートしない 【ASPS0107】．ただし，動的生成
5370 機能拡張パッケージでは，del_tskをサポートする 【ASPS0108】．
5371
5372 **【TOPPERS/FMPカーネルにおける規定】**
5373
5374 FMPカーネルでは，del_tskをサポートしない 【FMPS0105】．
5375
5376 **【TOPPERS/HRP2カーネルにおける規定】**
5377
5378 HRP2カーネルでは，del_tskをサポートしない 【HRPS0105】．ただし，動的生成
5379 機能拡張パッケージでは，del_tskをサポートする 【HRPS0178】．
5380
5381 **【TOPPERS/SSPカーネルにおける規定】**
5382
5383 SSPカーネルでは，del_tskをサポートしない 【SSPS0114】．
5384 -----
5385 act_tsk タスクの起動 [T] 【NGKI1110】
5386 iact_tsk タスクの起動 [I] 【NGKI1111】
5387
5388 **【C言語API】**
5389 ER ercd = act_tsk(ID tskid)
5390 ER ercd = iact_tsk(ID tskid)
5391
5392 **【パラメータ】**
5393 ID tskid 対象タスクのID番号
5394
5395 **【リターンパラメータ】**
5396 ER ercd 正常終了 (E_OK) またはエラーコード
5397
5398 **【エラーコード】**
5399 E_CTX コンテキストエラー
5400 ・非タスクコンテキストからの呼出し (act_tskの場合) 【NGKI1112】

5401 ・タスクコンテキストからの呼出し (iact_tskの場合) 【NGKI1113】
 5402 ・CPUロック状態からの呼出し 【NGKI1114】
 5403 E_ID 不正ID番号
 5404 ・tskidが有効範囲外 【NGKI1115】
 5405 E_NOEXS オブジェクト未登録
 5406 ・対象タスクが未登録 [D] 【NGKI1116】
 5407 E_OACV オブジェクトアクセス違反
 5408 ・対象タスクに対する通常操作1が許可されていない (act_tsk
 5409 の場合) [P] 【NGKI1117】
 5410 E_QOVR キューイングオーバフロー
 5411 ・条件については機能の項を参照

5412

5413 【機能】

5414

5415 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
 5416 いは以下の通り。

5417

5418 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
 5419 うべき初期化処理が行われ、対象タスクは実行できる状態になる 【NGKI1118】。

5420

5421 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
 5422 に1が加えられる 【NGKI1119】。起動要求キューイング数に1を加えると

5423 TMAX_ACTCNTを超える場合には、E_QOVRエラーとなる 【NGKI1120】。

5424

5425 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
 5426 となる 【NGKI1121】。

5427

5428 【補足説明】

5429

5430 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
 5431 起動時の割付けプロセッサを変更しない。

5432

5433 mact_tsk 割付けプロセッサ指定でのタスクの起動 [TM] 【NGKI1122】

5434 imact_tsk 割付けプロセッサ指定でのタスクの起動 [IM] 【NGKI1123】

5435

5436 【C言語API】

5437 ER ercd = mact_tsk(ID tskid, ID prcid)

5438 ER ercd = imact_tsk(ID tskid, ID prcid)

5439

5440 【パラメータ】

5441 ID tskid 対象タスクのID番号

5442 ID prcid タスクの割付け対象のプロセッサのID番号

5443

5444 【リターンパラメータ】

5445 ER ercd 正常終了 (E_OK) またはエラーコード

5446

5447 【エラーコード】

5448 E_CTX コンテキストエラー

5449 ・非タスクコンテキストからの呼出し (mact_tskの場合)

5450 【NGKI1124】

5451 ・タスクコンテキストからの呼出し (imact_tskの場合)
5452 【NGKI1125】
5453 ・CPUロック状態からの呼出し【NGKI1126】
5454 E_NOSPT 未サポート機能
5455 ・対象タスクが制約タスク【NGKI1127】
5456 E_ID 不正ID番号
5457 ・tskidが有効範囲外【NGKI1128】
5458 ・prcidが有効範囲外【NGKI1129】
5459 E_PAR パラメータエラー
5460 ・条件については機能の項を参照
5461 E_NOEXS オブジェクト未登録
5462 ・対象タスクが未登録 [D] 【NGKI1130】
5463 E_OACV オブジェクトアクセス違反
5464 ・対象タスクに対する通常操作1が許可されていない (mact_tsk
5465 の場合) [P] 【NGKI1131】
5466 E_QOVR キューイングオーバフロー
5467 ・条件については機能の項を参照

5468 【機能】

5469
5470
5471 prcidで指定したプロセッサを割付けプロセッサとして、tskidで指定したタス
5472 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。

5473
5474 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
5475 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
5476 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
5477 【NGKI1132】。

5478
5479 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
5480 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
5481 に変更される【NGKI1133】。起動要求キューイング数に1を加えると
5482 TMAX_ACTCNTを超える場合には、E_QOVRエラーとなる【NGKI1134】。

5483
5484 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
5485 クとなる【NGKI1135】。

5486
5487 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
5488 サを含んでいない場合には、E_PARエラーとなる【NGKI1136】。

5489
5490 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
5491 れが属するクラスの初期割付けプロセッサとする【NGKI1137】。

5492 【補足説明】

5493
5494
5495 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
5496 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
5497 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
5498 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
5499 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
5500 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

5501 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
5502 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
5503 変更されない。

5504
5505 **【TOPPERS/ASPカーネルにおける規定】**

5506
5507 ASPカーネルでは、mact_tsk, imact_tskをサポートしない【ASPS0109】。

5508
5509 **【TOPPERS/HRP2カーネルにおける規定】**

5510
5511 HRP2カーネルでは、mact_tsk, imact_tskをサポートしない【HRPS0106】。

5512
5513 **【TOPPERS/SSPカーネルにおける規定】**

5514
5515 SSPカーネルでは、mact_tsk, imact_tskをサポートしない【SSPS0115】。

5516
5517 **【μITRON4.0仕様との関係】**

5518
5519 μITRON4.0仕様に定義されていないサービスコールである。

5520 -----

5521 can_act タスク起動要求のキャンセル [T] 【NGKI1138】

5522
5523 **【C言語API】**

5524 ER_UINT actcnt = can_act(ID tskid)

5525
5526 **【パラメータ】**

5527 ID tskid 対象タスクのID番号

5528
5529 **【リターンパラメータ】**

5530 ER_UINT actcnt キューイングされていた起動要求の数（正の値
5531 または0）またはエラーコード

5532
5533 **【エラーコード】**

5534 E_CTX コンテキストエラー

5535 ・非タスクコンテキストからの呼出し【NGKI1139】

5536 ・CPUロック状態からの呼出し【NGKI1140】

5537 E_ID 不正ID番号

5538 ・tskidが有効範囲外【NGKI1141】

5539 E_NOEXS オブジェクト未登録

5540 ・対象タスクが未登録 [D] 【NGKI1142】

5541 E_OACV オブジェクトアクセス違反

5542 ・対象タスクに対する通常操作1が許可されていない [P]

5543 【NGKI1143】

5544
5545 **【機能】**

5546
5547 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
5548 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
5549 下の通り。

5550

5551 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
5552 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【NGKI1144】。ま
5553 た、マルチプロセッサ対応カーネルにおいては、対象タスクの次回起動時の割
5554 付けプロセッサが未設定状態に戻される【NGKI1145】。

5555
5556 tskidにTSK_SELF (=0) を指定すると、自タスクが対象タスクとなる
5557 【NGKI1146】。

5558
5559 【TOPPERS/SSPカーネルにおける規定】
5560

5561 SSPカーネルでは、can_actをサポートしない【SSPS0116】。

5562 -----
5563 mig_tsk タスクの割付けプロセッサの変更 [TM] 【NGKI1147】

5564
5565 【C言語API】
5566 ER ercd = mig_tsk(ID tskid, ID prcid)
5567

5568 【パラメータ】
5569 ID tskid 対象タスクのID番号
5570 ID prcid タスクの割付けプロセッサのID番号
5571

5572 【リターンパラメータ】
5573 ER ercd 正常終了 (E_OK) またはエラーコード
5574

5575 【エラーコード】
5576 E_CTX コンテキストエラー
5577 ・非タスクコンテキストからの呼出し【NGKI1148】
5578 ・CPUロック状態からの呼出し【NGKI1149】
5579 ・その他の条件については機能の項を参照
5580 E_NOSPT 未サポート機能
5581 ・対象タスクが制約タスク【NGKI1150】
5582 E_ID 不正ID番号
5583 ・tskidが有効範囲外【NGKI1151】
5584 ・prcidが有効範囲外【NGKI1152】
5585 E_PAR パラメータエラー
5586 ・条件については機能の項を参照
5587 E_NOEXS オブジェクト未登録
5588 ・対象タスクが未登録 [D] 【NGKI1153】
5589 E_OACV オブジェクトアクセス違反
5590 ・対象タスクに対する通常操作1が許可されていない [P]
5591 【NGKI1154】
5592 E_OBJ オブジェクト状態エラー
5593 ・条件については機能の項を参照
5594

5595 【機能】

5596
5597 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
5598 変更する。具体的な振舞いは以下の通り。

5599
5600 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

5601 合には、対象タスクをpcridで指定したプロセッサに割り付ける【NGKI1155】。
5602 対象タスクが実行できる状態の場合には、pcridで指定したプロセッサに割り付
5603 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
5604 【NGKI1156】。
5605
5606 対象タスクが、自タスクが割付けられたプロセッサと異なるプロセッサに割り
5607 付けられている場合には、E_OBJエラーとなる【NGKI1157】。
5608
5609 tskidにTSK_SELF (=0) を指定すると、自タスクが対象タスクとなる
5610 【NGKI1158】。
5611
5612 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、
5613 E_CTXエラーとなる【NGKI1159】。
5614
5615 対象タスクの属するクラスの割付け可能プロセッサが、pcridで指定したプロセッ
5616 サを含んでいない場合には、E_PARエラーとなる【NGKI1160】。
5617
5618 pcrd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
5619 れが属するクラスの初期割付けプロセッサに変更する【NGKI1161】。
5620
5621 【補足説明】
5622
5623 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
5624 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
5625 状態やディスパッチ禁止状態を用いて、他のタスクへのディスパッチが起こら
5626 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
5627 のを防ぐことができる。
5628
5629 対象タスクが、最初からpcridで指定したプロセッサに割り付けられている場合
5630 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
5631 クの中で最低となる。
5632
5633 【TOPPERS/ASPカーネルにおける規定】
5634
5635 ASPカーネルでは、mig_tskをサポートしない【ASPS0110】。
5636
5637 【TOPPERS/HRP2カーネルにおける規定】
5638
5639 HRP2カーネルでは、mig_tskをサポートしない【HRPS0107】。
5640
5641 【TOPPERS/SSPカーネルにおける規定】
5642
5643 SSPカーネルでは、mig_tskをサポートしない【SSPS0117】。
5644
5645 【μITRON4.0仕様との関係】
5646
5647 μITRON4.0仕様に定義されていないサービスコールである。
5648 -----
5649 ext_tsk 自タスクの終了 [T] 【NGKI1162】
5650

```

5651  【C言語API】
5652      ER ercd = ext_tsk()
5653
5654  【パラメータ】
5655      なし
5656
5657  【リターンパラメータ】
5658      ER          ercd          エラーコード
5659
5660  【エラーコード】
5661      E_SYS        システムエラー
5662                  ・カーネルの誤動作【NGKI1163】
5663      E_CTX        コンテキストエラー
5664                  ・非タスクコンテキストからの呼出し【NGKI1164】
5665
5666  【機能】
5667
5668  自タスクを終了させる．具体的には，自タスクに対してタスク終了時に行うべき
5669  処理が行われる【NGKI3449】．
5670
5671  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
5672  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
5673  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
5674  状態に遷移させた後，自タスクを終了させる【NGKI1168】．
5675
5676  ext_tskが正常に処理された場合，ext_tskからはリターンしない【NGKI1169】．
5677
5678  【TOPPERS/SSPカーネルにおける規定】
5679
5680  SSPカーネルでは，ext_tskをサポートしない【SSPS0118】．自タスクを終了さ
5681  せる場合には，タスクのメインルーチンからリターンする【SSPS0119】．
5682
5683  【μITRON4.0仕様との関係】
5684
5685  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
5686  とした．μITRON4.0仕様においては，ext_tskからはリターンしないと規定さ
5687  れている．
5688  -----
5689  ter_tsk      タスクの強制終了〔T〕【NGKI1170】
5690
5691  【C言語API】
5692      ER ercd = ter_tsk(ID tskid)
5693
5694  【パラメータ】
5695      ID          tskid          対象タスクのID番号
5696
5697  【リターンパラメータ】
5698      ER          ercd          正常終了（E_OK）またはエラーコード
5699
5700  【エラーコード】

```

5701 E_CTX コンテキストエラー
5702 ・非タスクコンテキストからの呼出し【NGKI1171】
5703 ・CPUロック状態からの呼出し【NGKI1172】
5704 E_ID 不正ID番号
5705 ・tskidが有効範囲外【NGKI1173】
5706 E_NOEXS オブジェクト未登録
5707 ・対象タスクが未登録 [D] 【NGKI1174】
5708 E_OACV オブジェクトアクセス違反
5709 ・対象タスクに対する通常操作2が許可されていない [P]
5710 【NGKI1175】
5711 E_ILUSE サービスコール不正使用
5712 ・対象タスクが自タスク【NGKI1176】
5713 E_OBJ オブジェクト状態エラー
5714 ・対象タスクが休止状態【NGKI1177】
5715 ・その他の条件については機能の項を参照
5716

5717 【機能】

5718
5719 tskidで指定したタスク（対象タスク）を終了させる．具体的には，対象タスク
5720 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
5721 が行われる【NGKI3450】．
5722

5723 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
5724 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
5725 ロセッサに割り付けられている場合には，E_OBJエラーとなる【NGKI1182】．
5726

5727 【TOPPERS/FMPカーネルにおける使用上の注意】

5728
5729 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
5730 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
5731 方式にも依存する）．
5732

5733 【TOPPERS/SSPカーネルにおける規定】

5734
5735 SSPカーネルでは，ter_tskをサポートしない【SSPS0120】．
5736

5737 chg_pri タスクのベース優先度の変更 [T] 【NGKI1183】
5738

5739 【C言語API】

5740 ER ercd = chg_pri(ID tskid, PRI tskpri)
5741

5742 【パラメータ】

5743 ID tskid 対象タスクのID番号
5744 PRI tskpri ベース優先度
5745

5746 【リターンパラメータ】

5747 ER ercd 正常終了 (E_OK) またはエラーコード
5748

5749 【エラーコード】

5750 E_CTX コンテキストエラー

5751 ・非タスクコンテキストからの呼出し【NGKI1184】
5752 ・CPUロック状態からの呼出し【NGKI1185】
5753 E_NOSPT 未サポート機能
5754 ・対象タスクが制約タスク【NGKI1186】
5755 E_ID 不正ID番号
5756 ・tskidが有効範囲外【NGKI1187】
5757 E_PAR パラメータエラー
5758 ・tskpriが有効範囲外【NGKI1188】
5759 E_NOEXS オブジェクト未登録
5760 ・対象タスクが未登録 [D] 【NGKI1189】
5761 E_OACV オブジェクトアクセス違反
5762 ・対象タスクに対する通常操作2が許可されていない [P]
5763 【NGKI1190】
5764 E_ILUSE サービスコール不正使用
5765 ・条件については機能の項を参照
5766 E_OBJ オブジェクト状態エラー
5767 ・対象タスクが休止状態【NGKI1191】
5768
5769 **【機能】**
5770
5771 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
5772 先度に変更する．具体的な振舞いは以下の通り．
5773
5774 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
5775 で指定した優先度に変更される【NGKI1192】．それに伴って、対象タスクの現
5776 在優先度も変更される【NGKI1193】．
5777
5778 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
5779 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
5780 クの中で最低優先順位となる【NGKI1194】．対象タスクが待ち状態で、タスク
5781 の優先度順の待ち行列につながれている場合には、対象タスクの変更後の現在
5782 優先度に従って、その待ち行列中での順序が変更される【NGKI1195】．待ち行
5783 列中に同じ現在優先度のタスクがある場合には、対象タスクの順序はそれら
5784 の中で最後になる【NGKI1196】．
5785
5786 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
5787 スクの現在優先度に変更されることはなく、優先順位も変更されない
5788 【NGKI1197】．
5789
5790 tskidにTSK_SELF (=0) を指定すると、自タスクが対象タスクとなる
5791 【NGKI1198】．また、tskpriにTPRI_INI (=0) を指定すると、対象タスクのベー
5792 ス優先度が、起動時優先度に変更される【NGKI1199】．
5793
5794 対象タスクが優先度上限ミューテックスをロックしているかロックを待ってい
5795 る場合、tskpriは、それらのミューテックスの上限優先度と同じかそれより低
5796 くなければならない．そうでない場合には、E_ILUSEエラーとなる【NGKI1201】．
5797
5798 保護機能対応カーネルで、chg_priを呼び出した処理単位がユーザドメインに属
5799 する場合、tskpriは、そのユーザドメインが指定できる最高のタスク優先度と
5800 同じかそれより低くなければならない．そうでない場合には、E_ILUSEエラーと

5801 なる【NGKI3440】.

5802

5803 【TOPPERS/SSPカーネルにおける規定】

5804

5805 SSPカーネルでは、chg_priをサポートしない【SSPS0121】.

5806

5807 【μITRON4.0仕様との関係】

5808

5809 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
5810 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
5811 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.

5812

5813 get_pri タスク優先度の参照 [T] 【NGKI1202】

5814

5815 【C言語API】

5816 ER ercd = get_pri(ID tskid, PRI *p_tskpri)

5817

5818 【パラメータ】

5819 ID tskid 対象タスクのID番号

5820 PRI * p_tskpri 現在優先度を入れるメモリ領域へのポインタ

5821

5822 【リターンパラメータ】

5823 ER ercd 正常終了 (E_OK) またはエラーコード

5824 PRI tskpri 現在優先度

5825

5826 【エラーコード】

5827 E_CTX コンテキストエラー

5828 ・非タスクコンテキストからの呼出し【NGKI1203】

5829 ・CPUロック状態からの呼出し【NGKI1204】

5830 E_ID 不正ID番号

5831 ・tskidが有効範囲外【NGKI1205】

5832 E_NOEXS オブジェクト未登録

5833 ・対象タスクが未登録 [D] 【NGKI1206】

5834 E_OACV オブジェクトアクセス違反

5835 ・対象タスクに対する参照操作が許可されていない [P] 【NGKI1207】

5836 E_MACV メモリアクセス違反

5837 ・p_tskpriが指すメモリ領域への書込みアクセスが許可され
5838 ていない [P] 【NGKI1208】

5839 E_OBJ オブジェクト状態エラー

5840 ・対象タスクが休止状態【NGKI1209】

5841

5842 【機能】

5843

5844 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
5845 いは以下の通り．

5846

5847 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
5848 が指すメモリ領域に返される【NGKI1210】．

5849

5850 tskidにTSK_SELF (=0) を指定すると、自タスクが対象タスクとなる

5851 【NGKI1211】 .

5852

5853 【TOPPERS/SSPカーネルにおける規定】

5854

5855 SSPカーネルでは、get_priをサポートしない【SSPS0122】 .

5856 -----

5857 get_inf 自タスクの拡張情報の参照 [T] 【NGKI1212】

5858

5859 【C言語API】

5860 ER ercd = get_inf(intptr_t *p_exinf)

5861

5862 【パラメータ】

5863 intptr_t * p_exinf 拡張情報を入れるメモリ領域へのポインタ

5864

5865 【リターンパラメータ】

5866 ER ercd 正常終了 (E_OK) またはエラーコード

5867 intptr_t exinf 拡張情報

5868

5869 【エラーコード】

5870 E_CTX コンテキストエラー

5871 ・ 非タスクコンテキストからの呼出し【NGKI1213】

5872 ・ CPUロック状態からの呼出し【NGKI1214】

5873 E_MACV メモリアクセス違反

5874 ・ p_exinfが指すメモリ領域への書込みアクセスが許可されて

5875 いない [P] 【NGKI1215】

5876

5877 【機能】

5878

5879 自タスクの拡張情報を参照する．参照した拡張情報は、p_exinfが指すメモリ領

5880 域に返される【NGKI1216】 .

5881

5882 【TOPPERS/SSPカーネルにおける規定】

5883

5884 SSPカーネルでは、get_infをサポートしない【SSPS0123】 .

5885

5886 【μITRON4.0仕様との関係】

5887

5888 μITRON4.0仕様に定義されていないサービスコールである．

5889 -----

5890 ref_tsk タスクの状態参照 [T] 【NGKI1217】

5891

5892 【C言語API】

5893 ER ercd = ref_tsk(ID tskid, T_RTsk *pk_rtsk)

5894

5895 【パラメータ】

5896 ID tskid 対象タスクのID番号

5897 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

5898

5899 【リターンパラメータ】

5900 ER ercd 正常終了 (E_OK) またはエラーコード

```

5901
5902     *タスクの現在状態（パケットの内容）
5903     STAT      tskstat   タスク状態
5904     PRI        tskpri   タスクの現在優先度
5905     PRI        tsbpri   タスクのベース優先度
5906     STAT      tskwait   タスクの待ち要因
5907     ID         wobjid   タスクの待ち対象のオブジェクトのID
5908     TMO        lefttmo  タスクがタイムアウトするまでの時間
5909     uint_t     actcnt   タスクの起動要求キューイング数
5910     uint_t     wupcnt   タスクの起床要求キューイング数
5911     bool_t     texmsk   タスクがタスク例外処理マスク状態か否か（保
5912                        護機能対応カーネルの場合）
5913     bool_t     waifbd   タスクが待ち禁止状態か否か（保護機能対応カー
5914                        ネルの場合）
5915     uint_t     svclevel タスクの拡張サービスコールのネストレベル（保
5916                        護機能対応カーネルの場合）
5917     ID         prcid    タスクの割付けプロセッサのID（マルチプロセッ
5918                        サ対応カーネルの場合）
5919     ID         actprec  タスクの次回起動時の割付けプロセッサのID（マ
5920                        ルチプロセッサ対応カーネルの場合）
5921
5922     【エラーコード】
5923     E_CTX      コンテキストエラー
5924                ・非タスクコンテキストからの呼出し【NGKI1218】
5925                ・CPUロック状態からの呼出し【NGKI1219】
5926     E_ID       不正ID番号
5927                ・tskidが有効範囲外【NGKI1220】
5928     E_NOEXS    オブジェクト未登録
5929                ・対象タスクが未登録〔D〕【NGKI1221】
5930     E_OACV     オブジェクトアクセス違反
5931                ・対象タスクに対する参照操作が許可されていない〔P〕【NGKI1222】
5932     E_MACV     メモリアクセス違反
5933                ・pk_rtskが指すメモリ領域への書込みアクセスが許可されて
5934                いない〔P〕【NGKI1223】
5935
5936     【機能】
5937
5938     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
5939     態は、pk_rtskで指定したメモリ領域に返される【NGKI1224】．
5940
5941     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
5942     る【NGKI1225】．
5943
5944     TTS_RUN    0x01U    実行状態
5945     TTS_RDY    0x02U    実行可能状態
5946     TTS_WAI    0x04U    待ち状態
5947     TTS_SUS    0x08U    強制待ち状態
5948     TTS_WAS    0x0cU    二重待ち状態
5949     TTS_DMT    0x10U    休止状態
5950

```

5951 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、
5952 tskstatがTTS_SUSとなる場合がある【NGKI1226】。この状況は、自タスクに対
5953 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
5954 るタスクから同じタスクに対してsus_tskが発行された場合に発生する可能性が
5955 ある。

5956

5957 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、
5958 tsbpriには対象タスクのベース優先度が返される【NGKI1227】。対象タスクが
5959 休止状態である場合には、tskpriとtsbpriの値は保証されない【NGKI1228】。

5960

5961 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
5962 いる状態であるかを表す次のいずれかの値が返される【NGKI1229】。

5963

5964	TTW_SLP	0x0001U	起床待ち
5965	TTW_DLY	0x0002U	時間経過待ち
5966	TTW_SEM	0x0004U	セマフォの資源獲得待ち
5967	TTW_FLG	0x0008U	イベントフラグ待ち
5968	TTW_SDTQ	0x0010U	データキューへの送信待ち
5969	TTW_RDTQ	0x0020U	データキューからの受信待ち
5970	TTW_SPDQ	0x0100U	優先度データキューへの送信待ち
5971	TTW_RPDQ	0x0200U	優先度データキューからの受信待ち
5972	TTW_MBX	0x0040U	メールボックスからの受信待ち
5973	TTW_MTX	0x0080U	ミューテックスのロック待ち状態
5974	TTW_SMBF	0x0400U	メッセージバッファへの送信待ち
5975	TTW_RMBF	0x0800U	メッセージバッファからの受信待ち
5976	TTW_MPF	0x2000U	固定長メモリブロックの獲得待ち

5977

5978 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
5979 【NGKI1230】。

5980

5981 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
5982 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
5983 【NGKI1231】。対象タスクが待ち状態でない場合や、起床待ち状態または時間
5984 経過待ち状態である場合には、wobjidの値は保証されない【NGKI1232】。

5985

5986 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
5987 スクがタイムアウトを起こすまでの相対時間が返される【NGKI1233】。タスク
5988 がタイムアウトを起こさない場合には、TMO_FEVR (=-1) が返される
5989 【NGKI1234】。

5990

5991 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
5992 が経過して待ち解除されるまでの相対時間が返される【NGKI1235】。ただし、
5993 返されるべき相対時間がTMO型に格納することができない場合がありうる。この
5994 場合には、相対時間 (RELTIM型、uint_t型に定義される) をTMO型 (int_t型に
5995 定義される) に型キャストした値が返される【NGKI1236】。

5996

5997 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
5998 【NGKI1237】。

5999

6000 actcntには、対象タスクの起動要求キューイング数が返される【NGKI1238】。

6001
6002 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
6003 ング数が返される【NGKI1239】。対象タスクが休止状態である場合には、
6004 wupcntの値は保証されない【NGKI1240】。
6005
6006 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、
6007 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
6008 falseが返される【NGKI1241】。waifbdには、対象タスクが待ち禁止状態の場合
6009 にtrue、そうでない場合にfalseが返される【NGKI1242】。またsvclevelには、
6010 対象タスクが拡張サービスコールを呼び出していない場合には0、呼び出してい
6011 る場合には、実行中の拡張サービスコールがネスト段数が返される
6012 【NGKI1243】。対象タスクが休止状態である場合には、texmsk, waifbd,
6013 svclevelの値は保証されない【NGKI1244】。
6014
6015 マルチプロセッサ対応カーネルでは、preidに、対象タスクの割付けプロセッサ
6016 のID番号が返される【NGKI1245】。またactprcには、対象タスクの次回起動時
6017 の割付けプロセッサのID番号が返される【NGKI1246】。次回起動時の割付けプ
6018 ロセッサが未設定の場合には、actprcにTPRC_NONE (=0) が返される
6019 【NGKI1247】。
6020
6021 tskidにTSK_SELF (=0) を指定すると、自タスクが対象タスクとなる
6022 【NGKI1248】。
6023
6024 【補足説明】
6025
6026 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
6027 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
6028 を正しく得ることができる。
6029
6030 【TOPPERS/ASPカーネルにおける規定】
6031
6032 ASPカーネルでは、tskwaitにTTW_MTX, TTW_SMBF, TTW_RMBFが返ることはない
6033 【ASPS0111】。ただし、ミューテックス機能拡張パッケージを用いると、
6034 tskwaitにTTW_MTXが返る場合がある【ASPS0112】。また、メッセージバッファ
6035 機能拡張パッケージを用いると、tskwaitにTTW_SMBFとTTW_RMBFが返る場合があ
6036 る【ASPS0208】。
6037
6038 【TOPPERS/FMPカーネルにおける規定】
6039
6040 FMPカーネルでは、tskwaitにTTW_MTX, TTW_SMBF, TTW_RMBFが返ることはない
6041 【FMPS0106】。
6042
6043 【TOPPERS/HRP2カーネルにおける規定】
6044
6045 HRP2カーネルでは、tskwaitにTTW_MBX, TTW_SMBF, TTW_RMBFが返ることはない
6046 【HRPS0108】。ただし、メッセージバッファ機能拡張パッケージを用いると、
6047 tskwaitにTTW_SMBFとTTW_RMBFが返る場合がある【HRPS0174】。
6048
6049 【TOPPERS/SSPカーネルにおける規定】
6050

6051 SSPカーネルでは、ref_tskをサポートしない【SSPS0124】.

6052

6053 **【使用上の注意】**

6054

6055 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
6056 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
6057 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
6058 ている可能性があるためである。

6059

6060 **【 μ ITRON4.0仕様との関係】**

6061

6062 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。
6063 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
6064 した。

6065

6066 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
6067 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。

6068

6069 **【 μ ITRON4.0/PX仕様との関係】**

6070

6071 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
6072 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
6073 トレベル(svclevel)を追加した。

6074

6075

6076 4.2 タスク付属同期機能

6077

6078 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
6079 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。

6080

6081 タスク付属同期機能に関連して、各タスクが持つ情報は次の通り【NGKI1249】.

6082

6083 ・起床要求キューイング数

6084

6085 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
6086 であり、タスクの起動時に0に初期化される【NGKI1250】.

6087

6088 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.

6089

6090 TMAX_WUPCNT タスクの起床要求キューイング数の最大値【NGKI1251】

6091

6092 **【TOPPERS/ASPカーネルにおける規定】**

6093

6094 ASPカーネルでは、TMAX_WUPCNTは1に固定されている【ASPS0113】.

6095

6096 **【TOPPERS/FMPカーネルにおける規定】**

6097

6098 FMPカーネルでは、TMAX_WUPCNTは1に固定されている【FMPS0107】.

6099

6100 **【TOPPERS/HRP2カーネルにおける規定】**

6101
6102 HRP2カーネルでは、TMAX_WUPCNTは1に固定されている【HRPS0109】。
6103
6104 【TOPPERS/SSPカーネルにおける規定】
6105
6106 SSPカーネルでは、タスク付属同期機能をサポートしない【SSPS0125】。
6107
6108 【μITRON4.0仕様との関係】
6109
6110 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。
6111 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
6112 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
6113 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。
6114 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
6115 スト数 (suscnt) を除外した。
6116 -----
6117 slp_tsk 起床待ち [T] 【NGKI1252】
6118 tslp_tsk 起床待ち (タイムアウト付き) [T] 【NGKI1253】
6119
6120 【C言語API】
6121 ER ercd = slp_tsk()
6122 ER ercd = tslp_tsk(TMO tmout)
6123
6124 【パラメータ】
6125 TMO tmout タイムアウト時間 (tslp_tskの場合)
6126
6127 【リターンパラメータ】
6128 ER ercd 正常終了 (E_OK) またはエラーコード
6129
6130 【エラーコード】
6131 E_CTX コンテキストエラー
6132 ・ディスパッチ保留状態からの呼出し【NGKI1254】
6133 E_NOSPT 未サポート機能
6134 ・制約タスクからの呼出し【NGKI1255】
6135 E_PAR パラメータエラー
6136 ・tmoutが無効 (tslp_tskの場合)【NGKI1256】
6137 E_TMOUT ポーリング失敗またはタイムアウト (slp_tskを除く)【NGKI1257】
6138 E_RLWAI 待ち禁止状態または待ち状態の強制解除【NGKI1258】
6139
6140 【機能】
6141
6142 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。
6143
6144 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
6145 数から1が減ぜられる【NGKI1259】。起床要求キューイング数が0の場合には、
6146 自タスクは起床待ち状態となる【NGKI1260】。
6147
6148 【補足説明】
6149
6150 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

6151 状態を維持し、自タスクの優先順位は変化しない。
6152 -----

6153 wup_tsk タスクの起床 [T] 【NGKI1261】
6154 iwup_tsk タスクの起床 [I] 【NGKI1262】
6155

6156 【C言語API】
6157 ER ercd = wup_tsk(ID tskid)
6158 ER ercd = iwup_tsk(ID tskid)
6159

6160 【パラメータ】
6161 ID tskid 対象タスクのID番号
6162

6163 【リターンパラメータ】
6164 ER ercd 正常終了 (E_OK) またはエラーコード
6165

6166 【エラーコード】
6167 E_CTX コンテキストエラー
6168 ・非タスクコンテキストからの呼出し (wup_tskの場合) 【NGKI1263】
6169 ・タスクコンテキストからの呼出し (iwup_tskの場合) 【NGKI1264】
6170 ・CPUロック状態からの呼出し 【NGKI1265】
6171 E_NOSPT 未サポート機能
6172 ・対象タスクが制約タスク 【NGKI1266】
6173 E_ID 不正ID番号
6174 ・tskidが有効範囲外 【NGKI1267】
6175 E_NOEXS オブジェクト未登録
6176 ・対象タスクが未登録 [D] 【NGKI1268】
6177 E_OACV オブジェクトアクセス違反
6178 ・対象タスクに対する通常操作1が許可されていない (wup_tsk
6179 の場合) [P] 【NGKI1269】
6180 E_OBJ オブジェクト状態エラー
6181 ・対象タスクが休止状態 【NGKI1270】
6182 E_QOVR キューイングオーバフロー
6183 ・条件については機能の項を参照
6184

6185 【機能】
6186

6187 tskidで指定したタスク (対象タスク) を起床する。具体的な振舞いは以下の通り。
6188
6189

6190 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
6191 【NGKI1271】。待ち解除されたタスクには、待ち状態となったサービスコール
6192 からE_OKが返る 【NGKI1272】。
6193

6194 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
6195 起床要求キューイング数に1が加えられる 【NGKI1273】。起床要求キューイング
6196 数に1を加えるとTMAX_WUPCNTを超える場合には、E_QOVRエラーとなる
6197 【NGKI1274】。
6198

6199 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
6200 となる 【NGKI1275】。

```

6201 -----
6202 can_wup      タスク起床要求のキャンセル [T]  【NGKI1276】
6203
6204 【C言語API】
6205     ER_UINT wupcnt = can_wup(ID tskid)
6206
6207 【パラメータ】
6208     ID          tskid      対象タスクのID番号
6209
6210 【リターンパラメータ】
6211     ER_UINT      wupcnt    キューイングされていた起床要求の数（正の値
6212                             または0）またはエラーコード
6213
6214 【エラーコード】
6215     E_CTX        コンテキストエラー
6216                             ・非タスクコンテキストからの呼出し 【NGKI1277】
6217                             ・CPUロック状態からの呼出し 【NGKI1278】
6218     E_NOSPT      未サポート機能
6219                             ・対象タスクが制約タスク 【NGKI1279】
6220     E_ID         不正ID番号
6221                             ・tskidが有効範囲外 【NGKI1280】
6222     E_NOEXS      オブジェクト未登録
6223                             ・対象タスクが未登録 [D]  【NGKI1281】
6224     E_OACV       オブジェクトアクセス違反
6225                             ・対象タスクに対する通常操作1が許可されていない [P]
6226                             【NGKI1282】
6227     E_OBJ        オブジェクト状態エラー
6228                             ・対象タスクが休止状態 【NGKI1283】
6229
6230 【機能】
6231
6232 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
6233 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
6234 下の通り。
6235
6236 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
6237 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
6238 の返値として返される 【NGKI1284】。
6239
6240 tskidにTSK_SELF（=0）を指定すると、自タスクが対象タスクとなる
6241 【NGKI1285】。
6242 -----
6243 rel_wai      強制的な待ち解除 [T]  【NGKI1286】
6244 irel_wai     強制的な待ち解除 [I]  【NGKI1287】
6245
6246 【C言語API】
6247     ER ercd = rel_wai(ID tskid)
6248     ER ercd = irel_wai(ID tskid)
6249
6250 【パラメータ】

```

6251	ID	tskid	対象タスクのID番号
6252			
6253	【リターンパラメータ】		
6254	ER	ercd	正常終了 (E_OK) またはエラーコード
6255			
6256	【エラーコード】		
6257	E_CTX	コンテキストエラー	
6258		・ 非タスクコンテキストからの呼出し (rel_waiの場合) 【NGKI1288】	
6259		・ タスクコンテキストからの呼出し (irel_waiの場合) 【NGKI1289】	
6260		・ CPUロック状態からの呼出し 【NGKI1290】	
6261	E_NOSPT	未サポート機能	
6262		・ 対象タスクが制約タスク 【NGKI1291】	
6263	E_ID	不正ID番号	
6264		・ tskidが有効範囲外 【NGKI1292】	
6265	E_NOEXS	オブジェクト未登録	
6266		・ 対象タスクが未登録 [D] 【NGKI1293】	
6267	E_OACV	オブジェクトアクセス違反	
6268		・ 対象タスクに対する通常操作2が許可されていない (rel_waiの場合) [P] 【NGKI1294】	
6269			
6270	E_OBJ	オブジェクト状態エラー	
6271		・ 対象タスクが待ち状態でない 【NGKI1295】	
6272			
6273	【機能】		
6274			
6275	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		
6276	舞いは以下の通り.		
6277			
6278	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		
6279	【NGKI1296】. 待ち解除されたタスクには, 待ち状態となったサービスコール		
6280	からE_RLWAIが返る 【NGKI1297】.		
6281	-----		
6282	sus_tsk	強制待ち状態への遷移 [T] 【NGKI1298】	
6283			
6284	【C言語API】		
6285	ER	ercd	= sus_tsk(ID tskid)
6286			
6287	【パラメータ】		
6288	ID	tskid	対象タスクのID番号
6289			
6290	【リターンパラメータ】		
6291	ER	ercd	正常終了 (E_OK) またはエラーコード
6292			
6293	【エラーコード】		
6294	E_CTX	コンテキストエラー	
6295		・ 非タスクコンテキストからの呼出し 【NGKI1299】	
6296		・ CPUロック状態からの呼出し 【NGKI1300】	
6297		・ その他の条件については機能の項を参照	
6298	E_NOSPT	未サポート機能	
6299		・ 対象タスクが制約タスク 【NGKI1301】	
6300	E_ID	不正ID番号	

6301		・ tskidが有効範囲外【NGKI1302】
6302	E_NOEXS	オブジェクト未登録
6303		・ 対象タスクが未登録 [D] 【NGKI1303】
6304	E_OACV	オブジェクトアクセス違反
6305		・ 対象タスクに対する通常操作2が許可されていない [P]
6306		【NGKI1304】
6307	E_OBJ	オブジェクト状態エラー
6308		・ 対象タスクが休止状態【NGKI1305】
6309	E_QOVR	キューイングオーバフロー
6310		・ 対象タスクが強制待ち状態（二重待ち状態を含む）【NGKI1306】

6312 【機能】

6314 tskidで指定したタスク（対象タスク）を強制待ちにする．具体的な振舞いは以下
6315 の通り．

6317 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
6318 る【NGKI1307】．また，待ち状態（二重待ち状態を除く）である場合には，二
6319 重待ち状態となる【NGKI1308】．

6321 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，
6322 E_QOVRエラーとなる場合がある【NGKI1309】．この状況は，自タスクに対して
6323 sus_tskを発行するのと同じタイミングで，他のプロセッサで実行されているタ
6324 スクから同じタスクに対してsus_tskが発行された場合に発生する可能性がある．

6326 tskidにTSK_SELF（=0）を指定すると，自タスクが対象タスクとなる
6327 【NGKI1310】．

6329 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，
6330 E_CTXエラーとなる【NGKI1311】．なお，sus_tskは，自タスクを広義の待ち状
6331 態に遷移させる可能性のあるサービスコールであるが，対象タスクが自タスク
6332 でない場合には，割込み優先度マスクが全解除でない状態やディスパッチ禁止
6333 状態で呼び出しても，E_CTXエラーにはならない【NGKI3604】．これは，
6334 [NGKI0175] と [NGKI0179] の原則の例外となっている．

6336 rsm_tsk 強制待ち状態からの再開 [T] 【NGKI1312】

6338 【C言語API】

6339 ER ercd = rsm_tsk(ID tskid)

6341 【パラメータ】

6342	ID	tskid	対象タスクのID番号
------	----	-------	------------

6344 【リターンパラメータ】

6345	ER	ercd	正常終了 (E_OK) またはエラーコード
------	----	------	-----------------------

6347 【エラーコード】

6348	E_CTX	コンテキストエラー
6349		・ 非タスクコンテキストからの呼出し【NGKI1313】
6350		・ CPUロック状態からの呼出し【NGKI1314】

6351	E_NOSPT	未サポート機能
6352		・対象タスクが制約タスク【NGKI1315】
6353	E_ID	不正ID番号
6354		・tskidが有効範囲外【NGKI1316】
6355	E_NOEXS	オブジェクト未登録
6356		・対象タスクが未登録 [D] 【NGKI1317】
6357	E_OACV	オブジェクトアクセス違反
6358		・対象タスクに対する通常操作2が許可されていない [P]
6359		【NGKI1318】
6360	E_OBJ	オブジェクト状態エラー
6361		・対象タスクが強制待ち状態（二重待ち状態を含む）でない
6362		【NGKI1319】
6363		
6364		【機能】
6365		
6366		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
6367		舞いは以下の通り．
6368		
6369		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
6370		れる【NGKI1320】．
6371		-----
6372	dis_wai	待ち禁止状態への遷移 [TP] 【NGKI1321】
6373	idis_wai	待ち禁止状態への遷移 [IP] 【NGKI1322】
6374		
6375		【C言語API】
6376	ER ercd = dis_wai(ID tskid)	
6377	ER ercd = idis_wai(ID tskid)	
6378		
6379		【パラメータ】
6380	ID	tskid 対象タスクのID番号
6381		
6382		【リターンパラメータ】
6383	ER	ercd 正常終了 (E_OK) またはエラーコード
6384		
6385		【エラーコード】
6386	E_CTX	コンテキストエラー
6387		・非タスクコンテキストからの呼出し (dis_waiの場合) 【NGKI1323】
6388		・タスクコンテキストからの呼出し (idis_waiの場合) 【NGKI1324】
6389		・CPUロック状態からの呼出し【NGKI1325】
6390	E_NOSPT	未サポート機能
6391		・対象タスクが制約タスク【NGKI1326】
6392	E_ID	不正ID番号
6393		・tskidが有効範囲外【NGKI1327】
6394	E_NOEXS	オブジェクト未登録
6395		・対象タスクが未登録 [D] 【NGKI1328】
6396	E_OACV	オブジェクトアクセス違反
6397		・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
6398		の場合) 【NGKI1329】
6399	E_OBJ	オブジェクト状態エラー
6400		・対象タスクが休止状態【NGKI1330】

6451 ER ercd 正常終了 (E_OK) またはエラーコード
6452
6453 **【エラーコード】**
6454 E_CTX コンテキストエラー
6455 ・非タスクコンテキストからの呼出し (ena_waiの場合) 【NGKI1337】
6456 ・タスクコンテキストからの呼出し (iena_waiの場合) 【NGKI1338】
6457 ・CPUロック状態からの呼出し 【NGKI1339】
6458 E_NOSPT 未サポート機能
6459 ・対象タスクが制約タスク 【NGKI1340】
6460 E_ID 不正ID番号
6461 ・tskidが有効範囲外 【NGKI1341】
6462 E_NOEXS オブジェクト未登録
6463 ・対象タスクが未登録 [D] 【NGKI1342】
6464 E_OACV オブジェクトアクセス違反
6465 ・対象タスクに対する通常操作2が許可されていない (ena_wai
6466 の場合) 【NGKI1343】
6467 E_OBJ オブジェクト状態エラー
6468 ・対象タスクが休止状態 【NGKI1344】
6469 ・対象タスクが待ち禁止状態でない 【NGKI1345】
6470
6471 **【機能】**
6472
6473 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
6474 舞いは以下の通り。
6475
6476 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
6477 【NGKI1346】。
6478
6479 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
6480 となる 【NGKI1347】。
6481
6482 **【TOPPERS/ASPカーネルにおける規定】**
6483
6484 ASPカーネルでは、ena_waiをサポートしない 【ASPS0115】。
6485
6486 **【TOPPERS/FMPカーネルにおける規定】**
6487
6488 FMPカーネルでは、ena_waiをサポートしない 【FMPS0109】。
6489
6490 **【未決定事項】**
6491
6492 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
6493 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、
6494 現時点では未決定である。
6495
6496 **【 μ ITRON4.0/PX仕様との関係】**
6497
6498 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。
6499 -----
6500 dly_tsk 自タスクの遅延 [T] 【NGKI1348】

6501

6502 **【C言語API】**

6503 ER ercd = dly_tsk(RELTIM dlytim)

6504

6505 **【パラメータ】**

6506 RELTIM dlytim 遅延時間

6507

6508 **【リターンパラメータ】**

6509 ER ercd 正常終了 (E_OK) またはエラーコード

6510

6511 **【エラーコード】**

6512 E_CTX コンテキストエラー

6513 ・ディスパッチ保留状態からの呼出し【NGKI1349】

6514 E_NOSPT 未サポート機能

6515 ・制約タスクからの呼出し【NGKI1350】

6516 E_PAR パラメータエラー

6517 ・dlytimがTMAX_RELTIMより大きい【NGKI1351】

6518 E_RLWAI 待ち禁止状態または待ち状態の強制解除【NGKI1352】

6519

6520 **【機能】**

6521

6522 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．

6523

6524 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

6525 なる【NGKI1353】．dly_tskを呼び出してからdlytimで指定した相対時間後に，

6526 自タスクは待ち解除され，dly_tskからE_OKが返る【NGKI1354】．

6527 -----

6528

6529 4.3 タスク例外処理機能

6530

6531 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

6532 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

6533 1つのみ登録できるため，タスクIDによって識別する【NGKI1355】．

6534

6535 タスク例外処理機能に関連して，各タスクが持つ情報は次の通り【NGKI1356】．

6536

6537 ・タスク例外処理ルーチン属性

6538 ・タスク例外処理禁止フラグ

6539 ・保留例外要因

6540 ・タスク例外処理ルーチンの先頭番地

6541

6542 タスク例外処理ルーチン属性に指定できる属性はない【NGKI1357】．そのため，

6543 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

6544 【NGKI1358】．

6545

6546 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

6547 止フラグを持つ【NGKI1359】．タスク例外処理禁止フラグがセットされた状態

6548 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

6549 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

6550 化される【NGKI1361】．

6551
6552 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
6553 のビットマップであり、タスクの起動時に0に初期化される【NGKI1362】。
6554
6555 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
6556 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
6557 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
6558 つの条件が揃った場合に実行が開始される【NGKI1363】。保護機能対応カーネ
6559 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
6560 加される【NGKI1364】。タスク例外処理マスク状態については、「2.6.5 タス
6561 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。
6562
6563 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
6564 セットされ、保留例外要因は0にクリアされる【NGKI1365】。また、タスク例外
6565 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
6566 る【NGKI1366】。
6567
6568 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
6569 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
6570 る【NGKI1367】。ここで、ユーザスタック領域に十分な空きがない場合や、ユー
6571 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、
6572 エミュレートされたCPU例外を発生させる【NGKI1368】。これを、タスク例外実
6573 行開始時スタック不正例外と呼ぶ。
6574
6575 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
6576 ステム状態等が、ユーザスタック上から取り出される【NGKI1369】。ここで、
6577 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
6578 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
6579 トされたCPU例外を発生させる【NGKI1370】。これを、タスク例外リターン時ス
6580 タック不正例外と呼ぶ。
6581
6582 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
6583 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、
6584 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、
6585 「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
6586 必要がある【NGKI1371】。この方法に従わなかった場合の動作は、保証されな
6587 い【NGKI1372】。
6588
6589 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
6590 護ドメインに属する【NGKI1373】。
6591
6592 タスク例外処理機能に用いるデータ型は次の通り。
6593
6594 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
6595 定義）【NGKI1374】
6596
6597 C言語によるタスク例外処理ルーチンの記述形式は次の通り【NGKI1375】。
6598
6599 void task_exception_routine(TEXPTN texptn, intptr_t exinf)
6600 {

6601 タスク例外処理ルーチン本体
6602 }
6603
6604 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
6605 の拡張情報が，それぞれ渡される【NGKI1376】．
6606
6607 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．
6608
6609 TBIT_TEXPTN タスク例外要因のビット数（TEXPTNの有効ビット数）
6610 【NGKI1377】
6611
6612 【補足説明】
6613
6614 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
6615 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
6616 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
6617 は，CPUロック状態で拡張サービスコールからリターンした場合（言い換えると，
6618 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
6619 揃うことになる．
6620
6621 【TOPPERS/ASPカーネルにおける規定】
6622
6623 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
6624 【ASPS0116】．
6625
6626 【TOPPERS/FMPカーネルにおける規定】
6627
6628 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
6629 【FMPS0110】．
6630
6631 【TOPPERS/HRP2カーネルにおける規定】
6632
6633 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
6634 【HRPS0110】．
6635
6636 【TOPPERS/SSPカーネルにおける規定】
6637
6638 SSPカーネルでは，タスク例外処理機能をサポートしない【SSPS0126】．
6639
6640 【μ ITRON4.0仕様との関係】
6641
6642 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
6643 行が開始されないという仕様に変更した．
6644
6645 【μ ITRON4.0/PX仕様との関係】
6646
6647 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
6648 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．
6649
6650 【仕様変更の経緯】

6651
6652 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
6653 「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
6654 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
6655 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
6656 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
6657 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
6658 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。

6659
6660 DEF_TEX タスク例外処理ルーチンの定義 [S] 【NGKI1378】
6661 def_tex タスク例外処理ルーチンの定義 [TD] 【NGKI1379】
6662
6663 **【静的API】**
6664 DEF_TEX(ID tskid, { ATR texatr, TEXRTN texrtn })
6665
6666 **【C言語API】**
6667 ER ercd = def_tex(ID tskid, const T_DTEX *pk_dtex)
6668
6669 **【パラメータ】**
6670 ID tskid 対象タスクのID番号
6671 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
6672 ケットへのポインタ（静的APIを除く）
6673
6674 *タスク例外処理ルーチンの定義情報（パケットの内容）
6675 ATR texatr タスク例外処理ルーチン属性
6676 TEXRTN texrtn タスク例外処理ルーチンの先頭番地
6677
6678 **【リターンパラメータ】**
6679 ER ercd 正常終了（E_OK）またはエラーコード
6680
6681 **【エラーコード】**
6682 E_CTX コンテキストエラー
6683 ・非タスクコンテキストからの呼出し [s] 【NGKI1380】
6684 ・CPUロック状態からの呼出し [s] 【NGKI1381】
6685 E_ID 不正ID番号
6686 ・tskidが有効範囲外 [s] 【NGKI1382】
6687 E_RSATR 予約属性
6688 ・texatrが無効【NGKI1383】
6689 ・その他の条件については機能の項を参照
6690 E_PAR パラメータエラー
6691 ・texrtnがプログラムの先頭番地として正しくない【NGKI1384】
6692 E_NOEXS オブジェクト未登録
6693 ・対象タスクが未登録【NGKI1385】
6694 E_OACV オブジェクトアクセス違反
6695 ・対象タスクに対する管理操作が許可されていない [sP]
6696 【NGKI1386】
6697 E_MACV メモリアクセス違反
6698 ・pk_dtexが指すメモリ領域への読出しアクセスが許可されて
6699 いない [sP] 【NGKI1387】
6700 E_OBJ オブジェクト状態エラー

- 6701 ・対象タスクは静的APIで生成された [s] 【NGKI1388】
- 6702 ・その他の条件については機能の項を参照

6703 【機能】

6704 tskidで指定したタスク（対象タスク）に対して、各パラメータで指定したタスク例外処理ルーチン定義情報に従って、タスク例外処理ルーチンを定義する

6705 【NGKI1389】.

6706 ただし、def_texにおいてpk_dtexをNULLにした場合には、対象タスクに対するタスク例外処理ルーチンの定義を解除する【NGKI1390】. また、対象タスクのタスク例外処理禁止フラグをセットし、保留例外要因を0に初期化する

6707 【NGKI1391】.

6708 静的APIにおいては、tskidはオブジェクト識別名、texatrは整数定数式パラメータ、texrtnは一般定数式パラメータである【NGKI1392】.

6709 タスク例外処理ルーチンを定義する場合（DEF_TEXの場合およびdef_texにおいてpk_dtexをNULL以外にした場合）で、対象タスクに対してすでにタスク例外処理ルーチンが定義されている場合には、E_OBJエラーとなる【NGKI1393】.

6710 保護機能対応カーネルにおいて、DEF_TEXは、対象タスクが属する保護ドメインの囲みの中に記述しなければならない. そうでない場合には、E_RSATRエラーとなる【NGKI1395】. また、def_texでタスク例外処理ルーチンを定義する場合には、タスク例外処理ルーチンの属する保護ドメインを設定する必要はなく、タスク例外処理ルーチン属性にTA_DOM(domid)を指定した場合にはE_RSATRエラーとなる【NGKI1396】. ただし、TA_DOM(TDOM_SELF)を指定した場合には、指定が無視され、E_RSATRエラーは検出されない【NGKI1397】.

6711 マルチプロセッサ対応カーネルにおいて、DEF_TEXは、対象タスクが属するクラスの囲みの中に記述しなければならない. そうでない場合には、E_RSATRエラーとなる【NGKI1399】. また、def_texでタスク例外処理ルーチンを定義する場合には、タスク例外処理ルーチンの属するクラスを設定する必要はなく、タスク例外処理ルーチン属性にTA_CLS(clsid)を指定した場合にはE_RSATRエラーとなる【NGKI1400】. ただし、TA_CLS(CLS_SELF)を指定した場合には、指定が無視され、E_RSATRエラーは検出されない【NGKI1401】.

6712 タスク例外処理ルーチンの定義を解除する場合（def_texにおいてpk_dtexをNULLにした場合）で、対象タスクに対してタスク例外処理ルーチンが定義されていない場合には、E_OBJエラーとなる【NGKI1402】.

6713 def_texにおいてtskidにTSK_SELF（=0）を指定すると、自タスクが対象タスクとなる【NGKI1403】.

6714 【TOPPERS/ASPカーネルにおける規定】

6715 ASPカーネルでは、DEF_TEXのみをサポートする【ASPS0117】. ただし、動的生成機能拡張パッケージでは、def_texもサポートする【ASPS0118】.

6716 【TOPPERS/FMPカーネルにおける規定】

6751
6752 FMPカーネルでは、DEF_TEXのみをサポートする【FMPS0111】。
6753
6754 【TOPPERS/HRP2カーネルにおける規定】
6755
6756 HRP2カーネルでは、DEF_TEXのみをサポートする【HRPS0111】。ただし、動的生
6757 成機能拡張パッケージでは、def_texもサポートする【HRPS0179】。
6758
6759 【 μ ITRON4.0仕様との関係】
6760
6761 texrtnのデータ型をTEXRTNに変更した。
6762
6763 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
6764 合に、E_OBJエラーとすることにした。
6765 -----
6766 ras_tex タスク例外処理の要求 [T] 【NGKI1404】
6767 iras_tex タスク例外処理の要求 [I] 【NGKI1405】
6768
6769 【C言語API】
6770 ER ercd = ras_tex(ID tskid, TEXPTN rasptn)
6771 ER ercd = iras_tex(ID tskid, TEXPTN rasptn)
6772
6773 【パラメータ】
6774 ID tskid 対象タスクのID番号
6775 TEXPTN rasptn 要求するタスク例外処理のタスク例外要因
6776
6777 【リターンパラメータ】
6778 ER ercd 正常終了 (E_OK) またはエラーコード
6779
6780 【エラーコード】
6781 E_CTX コンテキストエラー
6782 ・非タスクコンテキストからの呼出し (ras_texの場合) 【NGKI1406】
6783 ・タスクコンテキストからの呼出し (iras_texの場合) 【NGKI1407】
6784 ・CPUロック状態からの呼出し 【NGKI1408】
6785 E_ID 不正ID番号
6786 ・tskidが有効範囲外 【NGKI1409】
6787 E_PAR パラメータエラー
6788 ・rasptnが0 【NGKI1410】
6789 E_NOEXS オブジェクト未登録
6790 ・対象タスクが未登録 [D] 【NGKI1411】
6791 E_OACV オブジェクトアクセス違反
6792 ・対象タスクに対する通常操作2が許可されていない (ras_tex
6793 の場合) [P] 【NGKI1412】
6794 E_OBJ オブジェクト状態エラー
6795 ・対象タスクが休止状態 【NGKI1413】
6796 ・対象タスクに対してタスク例外処理ルーチンが定義されてい
6797 ない 【NGKI1414】
6798
6799 【機能】
6800

6801 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
6802 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
6803 値とrasptnで指定した値のビット毎論理和（C言語の“|”）に更新される
6804 【NGKI1415】．
6805
6806 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
6807 となる【NGKI1416】．
6808 -----
6809 dis_tex タスク例外処理の禁止 [T] 【NGKI1417】
6810
6811 【C言語API】
6812 ER ercd = dis_tex()
6813
6814 【パラメータ】
6815 なし
6816
6817 【リターンパラメータ】
6818 ER ercd 正常終了（E_OK）またはエラーコード
6819
6820 【エラーコード】
6821 E_CTX コンテキストエラー
6822 ・非タスクコンテキストからの呼出し【NGKI1419】
6823 ・CPUロック状態からの呼出し【NGKI1420】
6824 E_OBJ オブジェクト状態エラー
6825 ・自タスクに対してタスク例外処理ルーチンが定義されてい
6826 い【NGKI1421】
6827
6828 【機能】
6829
6830 自タスクのタスク例外処理禁止フラグをセットする【NGKI1422】．すなわち，
6831 自タスクをタスク例外処理禁止状態に遷移させる．
6832 -----
6833 ena_tex タスク例外処理の許可 [T] 【NGKI1423】
6834
6835 【C言語API】
6836 ER ercd = ena_tex()
6837
6838 【パラメータ】
6839 なし
6840
6841 【リターンパラメータ】
6842 ER ercd 正常終了（E_OK）またはエラーコード
6843
6844 【エラーコード】
6845 E_CTX コンテキストエラー
6846 ・非タスクコンテキストからの呼出し【NGKI1424】
6847 ・CPUロック状態からの呼出し【NGKI1425】
6848 E_OBJ オブジェクト状態エラー
6849 ・自タスクに対してタスク例外処理ルーチンが定義されてい
6850 い【NGKI1426】

6851
6852 **【機能】**
6853
6854 自タスクのタスク例外処理禁止フラグをクリアする【NGKI1427】．すなわち，
6855 自タスクをタスク例外処理許可状態に遷移させる．
6856
6857 **【補足説明】**
6858
6859 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
6860 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
6861 のは，アプリケーションの責任である．
6862 -----
6863 sns_tex タスク例外処理禁止状態の参照 [TI] **【NGKI1428】**
6864
6865 **【C言語API】**
6866 bool_t state = sns_tex()
6867
6868 **【パラメータ】**
6869 なし
6870
6871 **【リターンパラメータ】**
6872 bool_t state タスク例外処理禁止状態
6873
6874 **【機能】**
6875
6876 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
6877 以下の通り．
6878
6879 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
6880 許可状態の場合にfalseが返る【NGKI1429】．sns_texを非タスクコンテキスト
6881 から呼び出した場合で，実行状態のタスクがない場合には，trueが返る
6882 【NGKI1430】．
6883
6884 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
6885 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
6886 止フラグを参照する【NGKI1431】．
6887
6888 **【補足説明】**
6889
6890 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
6891 クに一致する．
6892 -----
6893 ref_tex タスク例外処理の状態参照 [T] **【NGKI1432】**
6894
6895 **【C言語API】**
6896 ER ercd = ref_tex(ID tskid, T_RTEX *pk_rtex)
6897
6898 **【パラメータ】**
6899 ID tskid 対象タスクのID番号
6900 T_RTEX * pk_rtex タスク例外処理の現在状態を入れるパケットへ

のポインタ

【リターンパラメータ】

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

＊タスク例外処理の現在状態 (パケットの内容)

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

【エラーコード】

E_CTX	コンテキストエラー
	・非タスクコンテキストからの呼出し【NGKI1433】
	・CPUロック状態からの呼出し【NGKI1434】
E_ID	不正ID番号
	・tskidが有効範囲外【NGKI1435】
E_NOEXS	オブジェクト未登録
	・対象タスクが未登録 [D] 【NGKI1436】
E_OACV	オブジェクトアクセス違反
	・対象タスクに対する参照操作が許可されていない [P] 【NGKI1437】
E_MACV	メモリアクセス違反
	・pk_rtexが指すメモリ領域への書込みアクセスが許可されていない [P] 【NGKI1438】
E_OBJ	オブジェクト状態エラー
	・対象タスクが休止状態【NGKI1439】
	・対象タスクに対してタスク例外処理ルーチンが定義されていない【NGKI1440】

【機能】

tskidで指定したタスク (対象タスク) のタスク例外処理に関する現在状態を参照する. 参照した現在状態は, pk_rtexで指定したパケットに返される【NGKI1441】.

texstatには, 対象タスクの現在のタスク例外処理禁止フラグを表す次のいずれかの値が返される【NGKI1442】.

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

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

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

4.4 同期・通信機能

同期・通信機能は, タスクとは独立したオブジェクトにより, タスクとタスクの間, または非タスクコンテキストの処理とタスクの間で同期・通信を行うための機能である.

6951
6952 【TOPPERS/SSPカーネルにおける規定】
6953
6954 SSPカーネルでは、同期・通信機能をサポートしない【SSPS0127】。
6955
6956 【μ ITRON4.0仕様との関係】
6957
6958 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
6959 デブ機能をサポートすることに変更する可能性もある。
6960
6961 4.4.1 セマフォ
6962
6963 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、
6964 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
6965 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
6966 却と呼ぶ。セマフォは、セマフォIDと呼ぶID番号によって識別する【NGKI1445】。
6967
6968 各セマフォが持つ情報は次の通り【NGKI1446】。
6969
6970 ・セマフォ属性
6971 ・資源数（の現在値）
6972 ・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）
6973 ・初期資源数
6974 ・最大資源数
6975 ・アクセス許可ベクタ（保護機能対応カーネルの場合）
6976 ・属する保護ドメイン（保護機能対応カーネルの場合）
6977 ・属するクラス（マルチプロセッサ対応カーネルの場合）
6978
6979 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
6980 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
6981 である。
6982
6983 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
6984 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
6985 る。資源数が最大資源数に一致している時に資源を返却しようとする、と
6986 E_QOVRエラーとなる【NGKI1447】。
6987
6988 セマフォ属性には、次の属性を指定することができる【NGKI1448】。
6989
6990 TA_TPRI 0x01U 待ち行列をタスクの優先度順にする
6991
6992 TA_TPRIを指定しない場合、待ち行列はFIFO順になる【NGKI1449】。
6993
6994 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。
6995
6996 TMAX_MAXSEM セマフォの最大資源数の最大値（=UINT_MAX）【NGKI1450】
6997
6998 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
6999 では、静的APIによって登録されたセマフォの数に一致）
7000 【NGKI1451】

7001
7002
7003
7004
7005
7006
7007
7008
7009
7010
7011
7012
7013
7014
7015
7016
7017
7018
7019
7020
7021
7022
7023
7024
7025
7026
7027
7028
7029
7030
7031
7032
7033
7034
7035
7036
7037
7038
7039
7040
7041
7042
7043
7044
7045
7046
7047
7048
7049
7050

【 μ ITRON4.0仕様との関係】

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。

CRE_SEM セマフォの生成 [S] 【NGKI1452】
acre_sem セマフォの生成 [TD] 【NGKI1453】

【静的API】

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

【C言語API】

ER_ID semid = acre_sem(const T_CSEM *pk_csem)

【パラメータ】

ID	semid	生成するセマフォのID番号 (CRE_SEMの場合)
T_CSEM *	pk_csem	セマフォの生成情報を入れたパッケージへのポインタ (静的APIを除く)

*セマフォの生成情報 (パッケージの内容)

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

【リターンパラメータ】

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

【エラーコード】

E_CTX	コンテキストエラー
	・非タスクコンテキストからの呼出し [s] 【NGKI1454】
	・CPUロック状態からの呼出し [s] 【NGKI1455】
E_RSATR	予約属性
	・sematrが無効 【NGKI1456】
	・属する保護ドメインの指定が有効範囲外 [sP] 【NGKI1457】
	・属するクラスの指定が有効範囲外 [sM] 【NGKI1458】
	・クラスの囲みの中に記述されていない [SM] 【NGKI1459】
E_PAR	パラメータエラー
	・maxsemが有効範囲 (1以上TMAX_MAXSEM以下) 外 【NGKI1468】
	・isemcntが有効範囲 (0以上maxsem以下) 外 【NGKI1466】
E_OACV	オブジェクトアクセス違反
	・システム状態に対する管理操作が許可されていない [sP] 【NGKI1460】
E_MACV	メモリアクセス違反
	・pk_csemが指すメモリ領域への読出しアクセスが許可されていない [sP] 【NGKI1461】
E_NOID	ID番号不足
	・割り付けられるセマフォIDがない [sD] 【NGKI1462】
E_OBJ	オブジェクト状態エラー
	・semidで指定したセマフォが登録済み (CRE_SEMの場合)

7051 【NGKI1463】

7052

7053 【機能】

7054

7055 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

7056 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

7057 る【NGKI1464】。

7058

7059 静的APIにおいては、semidはオブジェクト識別名、sematr, isemcnt, maxsemは

7060 整数定数式パラメータである【NGKI1465】。

7061

7062 【TOPPERS/ASPカーネルにおける規定】

7063

7064 ASPカーネルでは、CRE_SEMのみをサポートする【ASPS0119】。ただし、動的生

7065 成機能拡張パッケージでは、acre_semもサポートする【ASPS0120】。

7066

7067 【TOPPERS/FMPカーネルにおける規定】

7068

7069 FMPカーネルでは、CRE_SEMのみをサポートする【FMPS0112】。

7070

7071 【TOPPERS/HRP2カーネルにおける規定】

7072

7073 HRP2カーネルでは、CRE_SEMのみをサポートする【HRPS0112】。ただし、動的生

7074 成機能拡張パッケージでは、acre_semもサポートする【HRPS0180】。

7075 -----

7076 AID_SEM 割付け可能なセマフォIDの数の指定〔SD〕【NGKI1469】

7077

7078 【静的API】

7079 AID_SEM(uint_t nosem)

7080

7081 【パラメータ】

7082 uint_t nosem 割付け可能なセマフォIDの数

7083

7084 【エラーコード】

7085 E_RSATR 予約属性

7086 ・保護ドメインの囲みの中に記述されている〔P〕【NGKI3429】

7087 ・クラスの囲みの中に記述されていない〔M〕【NGKI1470】

7088 E_PAR パラメータエラー

7089 ・nosemが負の値【NGKI3277】

7090

7091 【機能】

7092

7093 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

7094 て割付け可能なセマフォIDとして確保する【NGKI1471】。

7095

7096 nosemは整数定数式パラメータである【NGKI1472】。

7097

7098 【TOPPERS/ASPカーネルにおける規定】

7099

7100 ASPカーネルの動的生成機能拡張パッケージでは、AID_SEMをサポートする

7102

7104

7107

7109

7110

7112

7113

7114

7116

7117

7119

7120

7121

7122

7124

7125	ACPTN	acptn?	通常操作?のアクセス許可パターン
------	-------	--------	------------------

7125	ACPTN	acptn2	運用操作プログラムの許可パターン
7126	ACPTN	acptn2	管理操作のアクセス許可パターン

7126	ACPTN	depths	官経探作のククミ六計のククミ
7187	ACPTN	depths	参照探作のククミ六計のククミ

7127	ACL IN	acptn4	参照操作のメタヒストリが	✓
7133				

7128

(129) 【リターンハフメータ】

7130	ER	ercd	正常終了 (E OK) またはエラーコード
------	----	------	-----------------------

7131

7132 【エラーコード】

7133 E CTX コンテキストエラー

7134 ・非タスクコンテキストからの呼出し [s] 【NGKI1475】

7135 ・CPUロック状態からの呼出し [s] 【NGKI1476】

7136	E ID	不正ID番号
------	------	--------

7137 ・ semidが有効範囲外 [s] 【NGKI1477】

7138 E RSATR 予約属性

7139 ・対象セマフォが属する保護ドメインの囲みの中（対象セマ
7140 フォが無所属の場合は、保護ドメインの囲みの外）に記述
7141 されていない [S] 【NGKI1478】

7142 ・対象セマフォが属するクラスの囲みの中に記述されていない
7143 「SM」【NGKI1479】

7144 E NOEXS オブジェクト未登録

7145 ・対象セマフォが未登録【NGKI1480】

7146 E OACV オブジェクトアクセス違反

7147 ・対象セマフォに対する管理操作が許可されていない [s]
7148 【NGKI1481】

7149 E MACV メモリアクセス違反

7150 ・ 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

7151 いない [s] 【NGKI1482】
7152 E_OBJ オブジェクト状態エラー
7153 ・対象セマフォは静的APIで生成された [s] 【NGKI1483】
7154 ・対象セマフォに対してアクセス許可ベクタが設定済み [S]
7155 【NGKI1484】
7156
7157 【機能】
7158
7159 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
7160 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
7161 【NGKI1485】。
7162
7163 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
7164 式パラメータである【NGKI1486】。
7165
7166 【TOPPERS/HRP2カーネルにおける規定】
7167
7168 HRP2カーネルでは、SAC_SEMのみをサポートする【HRPS0113】。ただし、動的生
7169 成機能拡張パッケージでは、sac_semもサポートする【HRPS0181】。
7170 -----
7171 del_sem セマフォの削除 [TD] 【NGKI1487】
7172
7173 【C言語API】
7174 ER ercd = del_sem(ID semid)
7175
7176 【パラメータ】
7177 ID semid 対象セマフォのID番号
7178
7179 【リターンパラメータ】
7180 ER ercd 正常終了（E_OK）またはエラーコード
7181
7182 【エラーコード】
7183 E_CTX コンテキストエラー
7184 ・非タスクコンテキストからの呼出し【NGKI1488】
7185 ・CPUロック状態からの呼出し【NGKI1489】
7186 E_ID 不正ID番号
7187 ・semidが有効範囲外【NGKI1490】
7188 E_NOEXS オブジェクト未登録
7189 ・対象セマフォが未登録【NGKI1491】
7190 E_OACV オブジェクトアクセス違反
7191 ・対象セマフォに対する管理操作が許可されていない [P]
7192 【NGKI1492】
7193 E_OBJ オブジェクト状態エラー
7194 ・対象セマフォは静的APIで生成された【NGKI1493】
7195
7196 【機能】
7197
7198 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
7199 の通り。
7200

7201 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
7202 【NGKI1494】。また、対象セマフォの待ち行列につながれたタスクは、待ち行
7203 列の先頭のタスクから順に待ち解除される【NGKI1495】。待ち解除されたタス
7204 クには、待ち状態となったサービスコールからE_DLTエラーが返る【NGKI1496】。
7205
7206 【使用上の注意】
7207
7208 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
7209 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
7210 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
7211 み禁止時間が長くなるため、注意が必要である。
7212
7213 【TOPPERS/ASPカーネルにおける規定】
7214
7215 ASPカーネルでは、del_semをサポートしない【ASPS0122】。ただし、動的生成
7216 機能拡張パッケージでは、del_semをサポートする【ASPS0123】。
7217
7218 【TOPPERS/FMPカーネルにおける規定】
7219
7220 FMPカーネルでは、del_semをサポートしない【FMPS0114】。
7221
7222 【TOPPERS/HRP2カーネルにおける規定】
7223
7224 HRP2カーネルでは、del_semをサポートしない【HRPS0114】。ただし、動的生成
7225 機能拡張パッケージでは、del_semをサポートする【HRPS0182】。
7226 -----
7227 sig_sem セマフォの資源の返却 [T] 【NGKI1497】
7228 isig_sem セマフォの資源の返却 [I] 【NGKI1498】
7229
7230 【C言語API】
7231 ER ercd = sig_sem(ID semid)
7232 ER ercd = isig_sem(ID semid)
7233
7234 【パラメータ】
7235 ID semid 対象セマフォのID番号
7236
7237 【リターンパラメータ】
7238 ER ercd 正常終了 (E_OK) またはエラーコード
7239
7240 【エラーコード】
7241 E_CTX コンテキストエラー
7242 ・非タスクコンテキストからの呼出し (sig_semの場合) 【NGKI1499】
7243 ・タスクコンテキストからの呼出し (isig_semの場合) 【NGKI1500】
7244 ・CPUロック状態からの呼出し 【NGKI1501】
7245 E_ID 不正ID番号
7246 ・semidが有効範囲外 【NGKI1502】
7247 E_NOEXS オブジェクト未登録
7248 ・対象セマフォが未登録 [D] 【NGKI1503】
7249 E_OACV オブジェクトアクセス違反
7250 ・対象セマフォに対する通常操作1が許可されていない (sig_sem

7287	E_CTX	コンテキストエラー
7288		・非タスクコンテキストからの呼出し【NGKI1513】
7289		・CPUロック状態からの呼出し【NGKI1514】
7290		・ディスパッチ保留状態からの呼出し（pol_semを除く）【NGKI1515】
7291	E_NOSPT	未サポート機能
7292		・制約タスクからの呼出し（pol_semを除く）【NGKI1516】
7293	E_ID	不正ID番号
7294		・semidが有効範囲外【NGKI1517】
7295	E_PAR	パラメータエラー
7296		・tmoutが無効（twai_semの場合）【NGKI1518】
7297	E_NOEXS	オブジェクト未登録
7298		・対象セマフォが未登録〔D〕【NGKI1519】
7299	E_OACV	オブジェクトアクセス違反
7300		・対象セマフォに対する通常操作2が許可されていない〔P〕

7301 【NGKI1520】
7302 E_TMOUT ポーリング失敗またはタイムアウト (wai_semを除く) 【NGKI1521】
7303 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)
7304 【NGKI1522】
7305 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)
7306 【NGKI1523】

7307

7308 【機能】

7309

7310 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
7311 いは以下の通り.

7312

7313 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

7314 【NGKI1524】. 資源数が0の場合には, 自タスクはセマフォの資源獲得待ち状態
7315 となり, 対象セマフォの待ち行列につながる 【NGKI1525】.

7316

7317 ini_sem セマフォの再初期化 [T] 【NGKI1526】

7318

7319 【C言語API】

7320 ER ercd = ini_sem(ID semid)

7321

7322 【パラメータ】

7323 ID semid 対象セマフォのID番号

7324

7325 【リターンパラメータ】

7326 ER ercd 正常終了 (E_OK) またはエラーコード

7327

7328 【エラーコード】

7329 E_CTX コンテキストエラー

7330 ・非タスクコンテキストからの呼出し 【NGKI1527】

7331 ・CPUロック状態からの呼出し 【NGKI1528】

7332 E_ID 不正ID番号

7333 ・semidが有効範囲外 【NGKI1529】

7334 E_NOEXS オブジェクト未登録

7335 ・対象セマフォが未登録 [D] 【NGKI1530】

7336 E_OACV オブジェクトアクセス違反

7337 ・対象セマフォに対する管理操作が許可されていない [P]

7338 【NGKI1531】

7339

7340 【機能】

7341

7342 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
7343 以下の通り.

7344

7345 対象セマフォの資源数は, 初期資源数に初期化される 【NGKI1532】. また, 対
7346 象セマフォの待ち行列につながれたタスクは, 待ち行列の先頭のタスクから順
7347 に待ち解除される 【NGKI1533】. 待ち解除されたタスクには, 待ち状態となっ
7348 たサービスコールからE_DLTエラーが返る 【NGKI1534】.

7349

7350 【使用上の注意】

7351
7352 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
7353 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
7354 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
7355 み禁止時間が長くなるため、注意が必要である。

7356
7357 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
7358 プリケーションの責任である。

7359
7360 【μITRON4.0仕様との関係】

7361
7362 μITRON4.0仕様に定義されていないサービスコールである。

7363 -----

7364 ref_sem セマフォの状態参照 [T] 【NGKI1535】

7365
7366 【C言語API】

7367 ER ercd = ref_sem(ID semid, T_RSEM *pk_rsem)

7368
7369 【パラメータ】

7370 ID semid 対象セマフォのID番号

7371 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
7372 ンタ

7373
7374 【リターンパラメータ】

7375 ER ercd 正常終了 (E_OK) またはエラーコード

7376
7377 *セマフォの現在状態 (パケットの内容)

7378 ID wtskid セマフォの待ち行列の先頭のタスクのID番号

7379 uint_t semcnt セマフォの資源数

7380
7381 【エラーコード】

7382 E_CTX コンテキストエラー

7383 ・非タスクコンテキストからの呼出し 【NGKI1536】

7384 ・CPUロック状態からの呼出し 【NGKI1537】

7385 E_ID 不正ID番号

7386 ・semidが有効範囲外 【NGKI1538】

7387 E_NOEXS オブジェクト未登録

7388 ・対象セマフォが未登録 [D] 【NGKI1539】

7389 E_OACV オブジェクトアクセス違反

7390 ・対象セマフォに対する参照操作が許可されていない [P]

7391 【NGKI1540】

7392 E_MACV メモリアクセス違反

7393 ・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
7394 いない [P] 【NGKI1541】

7395
7396 【機能】

7397
7398 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
7399 在状態は、pk_rsemで指定したパケットに返される 【NGKI1542】。

7400

7401 対象セマフォの待ち行列にタスクが存在しない場合、wtskidにはTSK_NONE (= 0) が返る【NGKI1543】。

7403

7404 【使用上の注意】

7405

7406 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。

7409 -----

7410

7411

7412

7413

7414

7415

7416

7417

7418

7419

7420

7421

7422

7423

7424

7425

7426

7427

7428

7429

7430

7431

7432

7433

7434

7435

7436

7437

7438

7439

7440

7441

7442

7443

7444

7445

7446

7447

7448

7449

7450

4.4.2 イベントフラグ

イベントフラグは、イベントの発生の有無を表すビットの集合（ビットパターン）を介して、イベント通知を行うための同期・通信オブジェクトである。イベントが発生している状態を1、発生していない状態を0とし、ビットパターンにより複数のイベントの発生の有無を表す【NGKI1544】。イベントフラグは、イベントフラグIDと呼ぶID番号によって識別する【NGKI1545】。

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

イベントフラグによりイベントの通知を受ける側のタスクは、待ちビットパターンと待ちモードにより、どのビットがセットされるのを待つかを指定する。待ちモードにTWF_ORW（=0x01U）を指定した場合、待ちビットパターンに含まれるいずれかのビットがセットされるのを待つ【NGKI1546】。待ちモードにTWF_ANDW（=0x02U）を指定した場合、待ちビットパターンに含まれるすべてのビットがセットされるのを待つ【NGKI1547】。この条件を、イベントフラグの待ち解除の条件と呼ぶ。

各イベントフラグが持つ情報は次の通り【NGKI1548】。

- ・ イベントフラグ属性
- ・ ビットパターン（の現在値）
- ・ 待ち行列（イベントフラグ待ち状態のタスクのキュー）
- ・ 初期ビットパターン
- ・ アクセス許可ベクタ（保護機能対応カーネルの場合）
- ・ 属する保護ドメイン（保護機能対応カーネルの場合）
- ・ 属するクラス（マルチプロセッサ対応カーネルの場合）

待ち行列は、イベントフラグが指定した待ち解除の条件を満たすまで待っている状態（イベントフラグ待ち状態）のタスクがつながれているキューである。待ち行列につながれたタスクの待ち解除は、待ち解除の条件を満たした中で、待ち行列の前方につながれたものから順に行われる（〔NGKI0216〕に該当）

【NGKI1549】。

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

7451
7452 イベントフラグ属性には、次の属性を指定することができる【NGKI1550】。
7453
7454 TA_TPRI 0x01U 待ち行列をタスクの優先度順にする
7455 TA_WMUL 0x02U 複数のタスクが待つのを許す
7456 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
7457
7458 TA_TPRIを指定しない場合、待ち行列はFIFO順になる【NGKI1551】。TA_WMULを
7459 指定しない場合、1つのイベントフラグに複数のタスクが待つことを禁止する
7460 【NGKI1552】。
7461
7462 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
7463 ンを0にクリアする【NGKI1553】。TA_CLRを指定しない場合、タスクの待ち解除
7464 時にイベントフラグをクリアしない【NGKI1554】。
7465
7466 イベントフラグ機能に用いるデータ型は次の通り。
7467
7468 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
7469 定義）【NGKI1555】
7470
7471 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。
7472
7473 TBIT_FLGPTN イベントフラグのビット数（FLGPTNの有効ビット数）
7474 【NGKI1556】
7475
7476 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
7477 ネルでは、静的APIによって登録されたイベントフラグの
7478 数に一致）【NGKI1557】
7479
7480 【TOPPERS/ASPカーネルにおける規定】
7481
7482 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
7483 【ASPS0124】。
7484
7485 【TOPPERS/FMPカーネルにおける規定】
7486
7487 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
7488 【FMPS0115】。
7489
7490 【TOPPERS/HRP2カーネルにおける規定】
7491
7492 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
7493 【HRPS0115】。
7494
7495 【μ ITRON4.0仕様との関係】
7496
7497 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。
7498 -----
7499 CRE_FLG イベントフラグの生成〔S〕【NGKI1558】
7500 acre_flg イベントフラグの生成〔TD〕【NGKI1559】

```

7501
7502 【静的API】
7503     CRE_FLG(ID flgid, { ATR flgatr, FLGPTN iflgptn })
7504
7505 【C言語API】
7506     ER_ID flgid = acre_flg(const T_CFLG *pk_cflg)
7507
7508 【パラメータ】
7509     ID          flgid      生成するイベントフラグのID番号（CRE_FLGの
7510                             場合）
7511     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
7512                             のポインタ（静的APIを除く）
7513
7514     * イベントフラグの生成情報（パケットの内容）
7515     ATR          flgatr    イベントフラグ属性
7516     FLGPTN      iflgptn   イベントフラグの初期ビットパターン
7517
7518 【リターンパラメータ】
7519     ER_ID      flgid      生成されたイベントフラグのID番号（正の値）
7520                             またはエラーコード
7521
7522 【エラーコード】
7523     E_CTX      コンテキストエラー
7524                 ・ 非タスクコンテキストからの呼出し [s] 【NGKI1560】
7525                 ・ CPUロック状態からの呼出し [s] 【NGKI1561】
7526     E_RSATR    予約属性
7527                 ・ flgatrが無効 【NGKI1562】
7528                 ・ 属する保護ドメインの指定が有効範囲外 [sP] 【NGKI1563】
7529                 ・ 属するクラスの指定が有効範囲外 [sM] 【NGKI1564】
7530                 ・ クラスの囲みの中に記述されていない [SM] 【NGKI1565】
7531     E_PAR      パラメータエラー
7532                 ・ iflgptnがFLGPTNに格納できない [S] 【NGKI3275】
7533     E_OACV     オブジェクトアクセス違反
7534                 ・ システム状態に対する管理操作が許可されていない [sP]
7535                     【NGKI1566】
7536     E_MACV     メモリアクセス違反
7537                 ・ 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
7538                     いない [sP] 【NGKI1567】
7539     E_NOID     ID番号不足
7540                 ・ 割り付けられるイベントフラグIDがない [sD] 【NGKI1568】
7541     E_OBJ      オブジェクト状態エラー
7542                 ・ flgidで指定したイベントフラグが登録済み（CRE_FLGの場合）
7543                     【NGKI1569】
7544
7545 【機能】
7546
7547     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
7548     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
7549     に、待ち行列は空の状態に初期化される 【NGKI1570】。
7550

```

7551 静的APIにおいては、flgidはオブジェクト識別名、flgatrとiflgptnは整数定数
7552 式パラメータである【NGKI1571】。

7553

7554 【TOPPERS/ASPカーネルにおける規定】

7555

7556 ASPカーネルでは、CRE_FLGのみをサポートする【ASPS0125】。ただし、動的生
7557 成機能拡張パッケージでは、acre_flgもサポートする【ASPS0126】。

7558

7559 【TOPPERS/FMPカーネルにおける規定】

7560

7561 FMPカーネルでは、CRE_FLGのみをサポートする【FMPS0116】。

7562

7563 【TOPPERS/HRP2カーネルにおける規定】

7564

7565 HRP2カーネルでは、CRE_FLGのみをサポートする【HRPS0116】。ただし、動的生
7566 成機能拡張パッケージでは、acre_flgもサポートする【HRPS0183】。

7567 -----

7568 AID_FLG 割付け可能なイベントフラグIDの数の指定〔SD〕【NGKI1572】

7569

7570 【静的API】

7571 AID_FLG(uint_t noflg)

7572

7573 【パラメータ】

7574 uint_t noflg 割付け可能なイベントフラグIDの数

7575

7576 【エラーコード】

7577 E_RSATR 予約属性

7578 ・保護ドメインの囲みの中に記述されている〔P〕【NGKI3430】

7579 ・クラスの囲みの中に記述されていない〔M〕【NGKI1573】

7580 E_PAR パラメータエラー

7581 ・noflgが負の値【NGKI3278】

7582

7583 【機能】

7584

7585 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
7586 コールによって割付け可能なイベントフラグIDとして確保する【NGKI1574】。

7587

7588 noflgは整数定数式パラメータである【NGKI1575】。

7589

7590 【TOPPERS/ASPカーネルにおける規定】

7591

7592 ASPカーネルの動的生成機能拡張パッケージでは、AID_FLGをサポートする
7593 【ASPS0212】。

7594

7595 【TOPPERS/HRP2カーネルにおける規定】

7596

7597 HRP2カーネルの動的生成機能拡張パッケージでは、AID_FLGをサポートする
7598 【HRPS0213】。

7599 -----

7600 SAC_FLG イベントフラグのアクセス許可ベクタの設定〔SP〕【NGKI1576】

7601 sac_flg イベントフラグのアクセス許可ベクタの設定 [TPD] 【NGKI1577】
7602
7603 **【静的API】**
7604 SAC_FLG(ID flgid, { ACPTN acptn1, ACPTN acptn2,
7605 ACPTN acptn3, ACPTN acptn4 })
7606
7607 **【C言語API】**
7608 ER ercd = sac_flg(ID flgid, const ACVCT *p_acvct)
7609
7610 **【パラメータ】**
7611 ID flgid 対象イベントフラグのID番号
7612 ACVCT * p_acvct アクセス許可ベクタを入れたパケットへのポ
7613 インタ（静的APIを除く）
7614
7615 *アクセス許可ベクタ（パケットの内容）
7616 ACPTN acptn1 通常操作1のアクセス許可パターン
7617 ACPTN acptn2 通常操作2のアクセス許可パターン
7618 ACPTN acptn3 管理操作のアクセス許可パターン
7619 ACPTN acptn4 参照操作のアクセス許可パターン
7620
7621 **【リターンパラメータ】**
7622 ER ercd 正常終了（E_OK）またはエラーコード
7623
7624 **【エラーコード】**
7625 E_CTX コンテキストエラー
7626 ・非タスクコンテキストからの呼出し [s] 【NGKI1578】
7627 ・CPUロック状態からの呼出し [s] 【NGKI1579】
7628 E_ID 不正ID番号
7629 ・flgidが有効範囲外 [s] 【NGKI1580】
7630 E_RSATR 予約属性
7631 ・対象イベントフラグが属する保護ドメインの囲みの中（対
7632 象イベントフラグが無所属の場合は、保護ドメインの囲み
7633 の外）に記述されていない [S] 【NGKI1581】
7634 ・対象イベントフラグが属するクラスの囲みの中に記述され
7635 ていない [SM] 【NGKI1582】
7636 E_NOEXS オブジェクト未登録
7637 ・対象イベントフラグが未登録 【NGKI1583】
7638 E_OACV オブジェクトアクセス違反
7639 ・対象イベントフラグに対する管理操作が許可されていない [s]
7640 【NGKI1584】
7641 E_MACV メモリアクセス違反
7642 ・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
7643 いない [s] 【NGKI1585】
7644 E_OBJ オブジェクト状態エラー
7645 ・対象イベントフラグは静的APIで生成された [s] 【NGKI1586】
7646 ・対象イベントフラグに対してアクセス許可ベクタが設定済
7647 み [S] 【NGKI1587】
7648
7649 **【機能】**
7650

7651 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
7652 （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
7653 【NGKI1588】．
7654
7655 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
7656 式パラメータである【NGKI1589】．
7657
7658 【TOPPERS/HRP2カーネルにおける規定】
7659
7660 HRP2カーネルでは、SAC_FLGのみをサポートする【HRPS0117】．ただし、動的生
7661 成機能拡張パッケージでは、sac_flgもサポートする【HRPS0184】．
7662 -----
7663 del_flg イベントフラグの削除〔TD〕【NGKI1590】
7664
7665 【C言語API】
7666 ER ercd = del_flg(ID flgid)
7667
7668 【パラメータ】
7669 ID flgid 対象イベントフラグのID番号
7670
7671 【リターンパラメータ】
7672 ER ercd 正常終了（E_OK）またはエラーコード
7673
7674 【エラーコード】
7675 E_CTX コンテキストエラー
7676 ・非タスクコンテキストからの呼出し【NGKI1591】
7677 ・CPUロック状態からの呼出し【NGKI1592】
7678 E_ID 不正ID番号
7679 ・flgidが有効範囲外【NGKI1593】
7680 E_NOEXS オブジェクト未登録
7681 ・対象イベントフラグが未登録【NGKI1594】
7682 E_OACV オブジェクトアクセス違反
7683 ・対象イベントフラグに対する管理操作が許可されていない〔P〕
7684 【NGKI1595】
7685 E_OBJ オブジェクト状態エラー
7686 ・対象イベントフラグは静的APIで生成された【NGKI1596】
7687
7688 【機能】
7689
7690 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
7691 振舞いは以下の通り．
7692
7693 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
7694 に戻される【NGKI1597】．また、対象イベントフラグの待ち行列につながれた
7695 タスクは、待ち行列の先頭のタスクから順に待ち解除される【NGKI1598】．待
7696 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
7697 返る【NGKI1599】．
7698
7699 【使用上の注意】
7700

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

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、del_flgをサポートしない【ASPS0128】。ただし、動的生成機能拡張パッケージでは、del_flgをサポートする【ASPS0129】。

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、del_flgをサポートしない【FMPS0118】。

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、del_flgをサポートしない【HRPS0118】。ただし、動的生成機能拡張パッケージでは、del_flgをサポートする【HRPS0185】。

set_flg イベントフラグのセット [T] 【NGKI1600】
 iset_flg イベントフラグのセット [I] 【NGKI1601】

【C言語API】

```
ER ercd = set_flg(ID flgid, FLGPTN setptn)
ER ercd = iset_flg(ID flgid, FLGPTN setptn)
```

【パラメータ】

ID	flgid	対象イベントフラグのID番号
FLGPTN	setptn	セットするビットパターン

【リターンパラメータ】

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

【エラーコード】

E_CTX	コンテキストエラー
	・非タスクコンテキストからの呼出し (set_flgの場合) 【NGKI1602】
	・タスクコンテキストからの呼出し (iset_flgの場合) 【NGKI1603】
	・CPUロック状態からの呼出し 【NGKI1604】
E_ID	不正ID番号
	・flgidが有効範囲外 【NGKI1605】
E_NOEXS	オブジェクト未登録
	・対象イベントフラグが未登録 [D] 【NGKI1606】
E_OACV	オブジェクトアクセス違反
	・対象イベントフラグに対する通常操作1が許可されていない (set_flgの場合) [P] 【NGKI1607】

【機能】

flgidで指定したイベントフラグ (対象イベントフラグ) のsetptnで指定したビットをセットする。具体的な振舞いは以下の通り。

7751
7752 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
7753 のビット毎論理和（C言語の“|”）に更新される【NGKI1608】。対象イベントフ
7754 ラグの待ち行列にタスクが存在する場合には、待ち解除の条件を満たしたタス
7755 クが、待ち行列の前方につながれたものから順に待ち解除される【NGKI1609】。
7756 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
7757 【NGKI1610】。
7758
7759 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
7760 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
7761 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。
7762
7763 【使用上の注意】
7764
7765 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
7766 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、
7767 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
7768 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
7769 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。
7770 -----
7771 clr_flg イベントフラグのクリア [T] 【NGKI1611】
7772
7773 【C言語API】
7774 ER ercd = clr_flg(ID flgid, FLGPTN clrptn)
7775
7776 【パラメータ】
7777 ID flgid 対象イベントフラグのID番号
7778 FLGPTN clrptn クリアするビットパターン（クリアしないビッ
7779 トを1、クリアするビットを0とする）
7780
7781 【リターンパラメータ】
7782 ER ercd 正常終了（E_OK）またはエラーコード
7783
7784 【エラーコード】
7785 E_CTX コンテキストエラー
7786 ・非タスクコンテキストからの呼出し【NGKI1612】
7787 ・CPUロック状態からの呼出し【NGKI1613】
7788 E_ID 不正ID番号
7789 ・flgidが有効範囲外【NGKI1614】
7790 E_NOEXS オブジェクト未登録
7791 ・対象イベントフラグが未登録 [D] 【NGKI1615】
7792 E_OACV オブジェクトアクセス違反
7793 ・対象イベントフラグに対する通常操作1が許可されていない [P]
7794 【NGKI1616】
7795
7796 【機能】
7797
7798 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
7799 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
7800 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

```

7801      【NGKI1617】 .
7802      -----
7803      wai_flg      イベントフラグ待ち [T]      【NGKI1618】
7804      pol_flg      イベントフラグ待ち（ポーリング） [T]      【NGKI1619】
7805      twai_flg     イベントフラグ待ち（タイムアウト付き） [T]      【NGKI1620】
7806
7807      【C言語API】
7808          ER ercd = wai_flg(ID flgid, FLGPTN waiptn, MODE wfmode, FLGPTN *p_flgptn)
7809          ER ercd = pol_flg(ID flgid, FLGPTN waiptn, MODE wfmode, FLGPTN *p_flgptn)
7810          ER ercd = twai_flg(ID flgid, FLGPTN waiptn,
7811                               MODE wfmode, FLGPTN *p_flgptn, TMO tmout)
7812
7813      【パラメータ】
7814          ID          flgid      対象イベントフラグのID番号
7815          FLGPTN      waiptn     待ちビットパターン
7816          MODE        wfmode     待ちモード
7817          FLGPTN *    p_flgptn   待ち解除時のビットパターンを入れるメモリ領
7818                               域へのポインタ
7819          TMO         tmout      タイムアウト時間（twai_flgの場合）
7820
7821      【リターンパラメータ】
7822          ER          ercd      正常終了（E_OK）またはエラーコード
7823          FLGPTN      flgptn   待ち解除時のビットパターン
7824
7825      【エラーコード】
7826          E_CTX      コンテキストエラー
7827                      ・非タスクコンテキストからの呼出し 【NGKI1621】
7828                      ・CPUロック状態からの呼出し 【NGKI1622】
7829                      ・ディスパッチ保留状態からの呼出し（pol_flgを除く） 【NGKI1623】
7830          E_NOSPT    未サポート機能
7831                      ・制約タスクからの呼出し（pol_flgを除く） 【NGKI1624】
7832          E_ID       不正ID番号
7833                      ・flgidが有効範囲外 【NGKI1625】
7834          E_PAR      パラメータエラー
7835                      ・waiptnが0 【NGKI1626】
7836                      ・wfmodeが無効（TWF_ORWまたはTWF_ANDWでない） 【NGKI1627】
7837                      ・tmoutが無効（twai_flgの場合） 【NGKI1628】
7838          E_NOEXS    オブジェクト未登録
7839                      ・対象イベントフラグが未登録 [D] 【NGKI1629】
7840          E_OACV     オブジェクトアクセス違反
7841                      ・対象イベントフラグに対する通常操作2が許可されていない [P]
7842                      【NGKI1630】
7843          E_MACV     メモリアクセス違反
7844                      ・p_flgptnが指すメモリ領域への書込みアクセスが許可され
7845                      ていない [P] 【NGKI1631】
7846          E_ILUSE    サービスコール不正使用
7847                      ・TA_WMUL属性でないイベントフラグで待ちタスクあり 【NGKI1632】
7848          E_TMOUT    ポーリング失敗またはタイムアウト（wai_flgを除く） 【NGKI1633】
7849          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）
7850                      【NGKI1634】

```

7851 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)
7852 【NGKI1635】

7853
7854 **【機能】**

7855
7856 flgidで指定したイベントフラグ (対象イベントフラグ) が, waipnとwfmdeで
7857 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.

7858
7859 対象イベントフラグが, waipnとwfmdeで指定した待ち解除の条件を満たして
7860 いる場合には, 対象イベントフラグのビットパターンの現在値がp_flgptnが指
7861 すメモリ領域に返される 【NGKI1636】. 対象イベントフラグがTA_CLR属性であ
7862 る場合には, 対象イベントフラグのビットパターンが0にクリアされる
7863 【NGKI1637】.

7864
7865 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
7866 態となり, 対象イベントフラグの待ち行列につながる 【NGKI1638】.

7867 -----
7868 ini_flg イベントフラグの再初期化 [T] 【NGKI1639】

7869
7870 **【C言語API】**

7871 ER ercd = ini_flg(ID flgid)

7872
7873 **【パラメータ】**

7874 ID flgid 対象イベントフラグのID番号

7875
7876 **【リターンパラメータ】**

7877 ER ercd 正常終了 (E_OK) またはエラーコード

7878
7879 **【エラーコード】**

7880 E_CTX コンテキストエラー
7881 ・非タスクコンテキストからの呼出し 【NGKI1640】
7882 ・CPUロック状態からの呼出し 【NGKI1641】
7883 E_ID 不正ID番号
7884 ・flgidが有効範囲外 【NGKI1642】
7885 E_NOEXS オブジェクト未登録
7886 ・対象イベントフラグが未登録 [D] 【NGKI1643】
7887 E_OACV オブジェクトアクセス違反
7888 ・対象イベントフラグに対する管理操作が許可されていない [P]
7889 【NGKI1644】

7890
7891 **【機能】**

7892
7893 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
7894 的な振舞いは以下の通り.

7895
7896 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
7897 【NGKI1645】. また, 対象イベントフラグの待ち行列につながれたタスクは,
7898 待ち行列の先頭のタスクから順に待ち解除される 【NGKI1646】. 待ち解除され
7899 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
7900 【NGKI1647】.

7901
7902
7903
7904
7905
7906
7907
7908
7909
7910
7911
7912
7913
7914
7915
7916
7917
7918
7919
7920
7921
7922
7923
7924
7925
7926
7927
7928
7929
7930
7931
7932
7933
7934
7935
7936
7937
7938
7939
7940
7941
7942
7943
7944
7945
7946
7947
7948
7949
7950

【使用上の注意】

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

イベントフラグを再初期化した場合に、アプリケーションとの整合性を保つのは、アプリケーションの責任である。

【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである。

ref_flg イベントフラグの状態参照 [T] 【NGKI1648】

【C言語API】

ER ercd = ref_flg(ID flgid, T_RFLG *pk_rflg)

【パラメータ】

ID	flgid	対象イベントフラグのID番号
T_RFLG *	pk_rflg	イベントフラグの現在状態を入れるパケットへのポインタ

【リターンパラメータ】

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

* イベントフラグの現在状態 (パケットの内容)

ID	wtskid	イベントフラグの待ち行列の先頭のタスクのID番号
uint_t	flgptn	イベントフラグのビットパターン

【エラーコード】

E_CTX	コンテキストエラー
	・ 非タスクコンテキストからの呼出し 【NGKI1649】
	・ CPUロック状態からの呼出し 【NGKI1650】
E_ID	不正ID番号
	・ flgidが有効範囲外 【NGKI1651】
E_NOEXS	オブジェクト未登録
	・ 対象イベントフラグが未登録 [D] 【NGKI1652】
E_OACV	オブジェクトアクセス違反
	・ 対象イベントフラグに対する参照操作が許可されていない [P] 【NGKI1653】
E_MACV	メモリアクセス違反
	・ pk_rflgが指すメモリ領域への書込みアクセスが許可されていない [P] 【NGKI1654】

【機能】

7951 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。
7952 参照した現在状態は、pk_rflgで指定したパケットに返される【NGKI1655】。

7953

7954 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
7955 TSK_NONE（=0）が返る【NGKI1656】。

7956

7957 【使用上の注意】

7958

7959 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
7960 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
7961 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
7962 グの状態が変化している可能性があるためである。

7963 -----

7964

7965 4.4.3 データキュー

7966

7967 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
7968 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
7969 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
7970 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
7971 号によって識別する【NGKI1657】。

7972

7973 各データキューが持つ情報は次の通り【NGKI1658】。

7974

- 7975 ・データキュー属性
- 7976 ・データキュー管理領域
- 7977 ・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）
- 7978 ・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）
- 7979 ・アクセス許可バクタ（保護機能対応カーネルの場合）
- 7980 ・属する保護ドメイン（保護機能対応カーネルの場合）
- 7981 ・属するクラス（マルチプロセッサ対応カーネルの場合）

7982

7983 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
7984 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
7985 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
7986 ズを0とすることができる【NGKI1659】。

7987

7988 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
7989 オブジェクト管理領域として扱われる【NGKI1660】。

7990

7991 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
7992 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
7993 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
7994 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
7995 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。

7996

7997 データキュー属性には、次の属性を指定することができる【NGKI1661】。

7998

7999 TA_TPRI 0x01U 送信待ち行列をタスクの優先度順にする

8000

8001 TA_TPRIを指定しない場合、送信待ち行列はFIFO順になる【NGKI1662】．受信待
8002 ち行列は、FIFO順に固定されている【NGKI1663】．

8003

8004 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．

8005

8006 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
8007 ネルでは、静的APIによって登録されたデータキューの数
8008 に一致）【NGKI1664】

8009

8010 【μITRON4.0仕様との関係】

8011

8012 TNUM_DTQIDは、μITRON4.0仕様に規定されていないカーネル構成マクロである．

8013

8014 CRE_DTQ データキューの生成〔S〕【NGKI1665】

8015 acre_dtq データキューの生成〔TD〕【NGKI1666】

8016

8017 【静的API】

8018 CRE_DTQ(ID dtqid, { ATR dtqatr, uint_t dtqcnt, void *dtqmb })

8019

8020 【C言語API】

8021 ER_ID dtqid = acre_dtq(const T_CDTQ *pk_cdtq)

8022

8023 【パラメータ】

8024 ID dtqid 生成するデータキューのID番号（CRE_DTQの場合）

8025 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
8026 ポインタ（静的APIを除く）

8027

8028 *データキューの生成情報（パケットの内容）

8029 ATR dtqatr データキュー属性

8030 uint_t dtqcnt データキュー管理領域に格納できるデータ数

8031 void * dtqmb データキュー管理領域の先頭番地

8032

8033 【リターンパラメータ】

8034 ER_ID dtqid 生成されたデータキューのID番号（正の値）ま
8035 たはエラーコード

8036

8037 【エラーコード】

8038 E_CTX コンテキストエラー

8039 ・非タスクコンテキストからの呼出し〔s〕【NGKI1667】

8040 ・CPUロック状態からの呼出し〔s〕【NGKI1668】

8041 E_RSATR 予約属性

8042 ・dtqatrが無効【NGKI1669】

8043 ・属する保護ドメインの指定が有効範囲外〔sP〕【NGKI1670】

8044 ・属するクラスの指定が有効範囲外〔sM〕【NGKI1671】

8045 ・クラスの囲みの中に記述されていない〔SM〕【NGKI1672】

8046 E_NOSPT 未サポート機能

8047 ・条件については各カーネルにおける規定の項を参照

8048 E_PAR パラメータエラー

8049 ・dtqcntが負の値〔S〕【NGKI3288】

8050 ・その他の条件については機能の項を参照

8051 E_OACV オブジェクトアクセス違反
8052 ・システム状態に対する管理操作が許可されていない [sP]
8053 【NGKI1673】
8054 E_MACV メモリアクセス違反
8055 ・pk_cdtqが指すメモリ領域への読出しアクセスが許可されて
8056 いない [sP] 【NGKI1674】
8057 E_NOID ID番号不足
8058 ・割り付けられるデータキューIDがない [sD] 【NGKI1675】
8059 E_NOMEM メモリ不足
8060 ・データキュー管理領域が確保できない 【NGKI1676】
8061 E_OBJ オブジェクト状態エラー
8062 ・dtqidで指定したデータキューが登録済み (CRE_DTQの場合)
8063 【NGKI1677】
8064 ・その他の条件については機能の項を参照
8065
8066 【機能】
8067
8068 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
8069 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
8070 タがない状態に初期化される【NGKI1678】。また、送信待ち行列と受信待ち行
8071 列は、空の状態に初期化される【NGKI1679】。
8072
8073 静的APIにおいては、dtqidはオブジェクト識別名、dtqatrとdtqcntは整数定数
8074 式パラメータ、dtqmbは一般定数式パラメータである【NGKI1680】。コンフィギュ
8075 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
8076 【NGKI1681】。
8077
8078 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
8079 管理領域が、コンフィギュレータまたはカーネルにより確保される【NGKI1682】。
8080
8081 [dtqmbにNULL以外を指定した場合]
8082
8083 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
8084 域は、アプリケーションで確保しておく必要がある【NGKI1683】。データキュー
8085 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
8086 【NGKI1684】。
8087
8088 TSZ_DTQMB(dtqcnt) dtqcntで指定した数のデータを格納できるデータ
8089 キュー管理領域のサイズ (バイト数)
8090 TCNT_DTQMB(dtqcnt) dtqcntで指定した数のデータを格納できるデータ
8091 キュー管理領域を確保するために必要なMB_T型の配
8092 列の要素数
8093
8094 これらを用いてデータキュー管理領域を確保する方法は次の通り【NGKI1685】。
8095
8096 MB_T <データキュー管理領域の変数名>[TCNT_DTQMB(dtqcnt)];
8097
8098 この時、dtqmbには<データキュー管理領域の変数名>を指定する【NGKI1686】。
8099
8100 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

8101 した時には、E_PARエラーとなる【NGKI1687】。また、保護機能対応カーネルに
8102 おいて、dtqmbで指定したデータキュー管理領域がカーネル専用のメモリオブジェ
8103 クトに含まれない場合、E_OBJエラーとなる【NGKI1688】。

8104
8105 **【TOPPERS/ASPカーネルにおける規定】**

8106
8107 ASPカーネルでは、CRE_DTQのみをサポートする【ASPS0130】。また、dtqmbには
8108 NULLのみを指定することができる。NULL以外を指定した場合には、E_NOSPTエラー
8109 となる【ASPS0132】。ただし、動的生成機能拡張パッケージでは、acre_dtqも
8110 サポートする【ASPS0133】。acre_dtqに対しては、dtqmbにNULL以外を指定でき
8111 ないという制限はない【ASPS0134】。

8112
8113 **【TOPPERS/FMPカーネルにおける規定】**

8114
8115 FMPカーネルでは、CRE_DTQのみをサポートする【FMPS0119】。また、dtqmbには
8116 NULLのみを指定することができる。NULL以外を指定した場合には、E_NOSPTエラー
8117 となる【FMPS0121】。

8118
8119 **【TOPPERS/HRP2カーネルにおける規定】**

8120
8121 HRP2カーネルでは、CRE_DTQのみをサポートする【HRPS0119】。また、dtqmbに
8122 はNULLのみを指定することができる。NULL以外を指定した場合には、E_NOSPTエ
8123 ラーとなる【HRPS0121】。ただし、動的生成機能拡張パッケージでは、
8124 acre_dtqもサポートする【HRPS0186】。acre_dtqに対しては、dtqmbにNULL以外
8125 を指定できないという制限はない【HRPS0187】。

8126
8127 **【μITRON4.0仕様との関係】**

8128
8129 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、
8130 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
8131 頭番地）に改名した。また、TSZ_DTQをTSZ_DTQMBに改名した。

8132
8133 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
8134 法を規定した。

8135 -----
8136 AID_DTQ 割付け可能なデータキューIDの数の指定〔SD〕【NGKI1689】

8137
8138 **【静的API】**

8139 AID_DTQ(uint_t nodtq)

8140
8141 **【パラメータ】**

8142 uint_t nodtq 割付け可能なデータキューIDの数

8143
8144 **【エラーコード】**

8145 E_RSATR 予約属性
8146 ・保護ドメインの囲みの中に記述されている〔P〕【NGKI3431】
8147 ・クラスの囲みの中に記述されていない〔M〕【NGKI1690】
8148 E_PAR パラメータエラー
8149 ・nodtqが負の値【NGKI3279】
8150

8201 に記述されていない [S] 【NGKI1698】
8202 ・対象データキューが属するクラスの囲みの中に記述されて
8203 いない [SM] 【NGKI1699】
8204 E_NOEXS オブジェクト未登録
8205 ・対象データキューが未登録 【NGKI1700】
8206 E_OACV オブジェクトアクセス違反
8207 ・対象データキューに対する管理操作が許可されていない [s]
8208 【NGKI1701】
8209 E_MACV メモリアクセス違反
8210 ・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
8211 いない [s] 【NGKI1702】
8212 E_OBJ オブジェクト状態エラー
8213 ・対象データキューは静的APIで生成された [s] 【NGKI1703】
8214 ・対象データキューに対してアクセス許可ベクタが設定済み [S]
8215 【NGKI1704】

8217 【機能】

8218
8219 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
8220 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
8221 【NGKI1705】。

8222
8223 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
8224 式パラメータである 【NGKI1706】。

8226 【TOPPERS/HRP2カーネルにおける規定】

8227
8228 HRP2カーネルでは、SAC_DTQのみをサポートする 【HRPS0122】。ただし、動的生
8229 成機能拡張パッケージでは、sac_dtqもサポートする 【HRPS0188】。

8230 -----
8231 del_dtq データキューの削除 [TD] 【NGKI1707】

8233 【C言語API】

8234 ER ercd = del_dtq(ID dtqid)

8236 【パラメータ】

8237 ID dtqid 対象データキューのID番号

8239 【リターンパラメータ】

8240 ER ercd 正常終了 (E_OK) またはエラーコード

8242 【エラーコード】

8243 E_CTX コンテキストエラー
8244 ・非タスクコンテキストからの呼出し 【NGKI1708】
8245 ・CPUロック状態からの呼出し 【NGKI1709】
8246 E_ID 不正ID番号
8247 ・dtqidが有効範囲外 【NGKI1710】
8248 E_NOEXS オブジェクト未登録
8249 ・対象データキューが未登録 【NGKI1711】
8250 E_OACV オブジェクトアクセス違反

・対象データキューに対する管理操作が許可されていない【P】
 【NGKI1712】
 E_OBJ オブジェクト状態エラー
 ・対象データキューは静的APIで生成された【NGKI1713】

【機能】

dtqidで指定したデータキュー（対象データキュー）を削除する．具体的な振舞いは以下の通り．

対象データキューの登録が解除され，そのデータキューIDが未使用の状態に戻される【NGKI1714】．また，対象データキューの送信待ち行列と受信待ち行列につながれたタスクは，それぞれの待ち行列の先頭のタスクから順に待ち解除される【NGKI1715】．待ち解除されたタスクには，待ち状態となったサービスコールからE_DLTエラーが返る【NGKI1716】．

データキューの生成時に，データキュー管理領域がカーネルによって確保された場合は，そのメモリ領域が解放される【NGKI1717】．

【補足説明】

送信待ち行列と受信待ち行列の両方にタスクがつながれていることはないため，別の待ち行列で待っていたタスクの間の待ち解除の順序は，規定する必要がない．

【使用上の注意】

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

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは，del_dtqをサポートしない【ASPS0136】．ただし，動的生成機能拡張パッケージでは，del_dtqをサポートする【ASPS0137】．

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは，del_dtqをサポートしない【FMPS0123】．

【TOPPERS/HRP2カーネルにおける規定】

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

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

```

8301
8302 【C言語API】
8303     ER ercd = snd_dtq(ID dtqid, intptr_t data)
8304     ER ercd = psnd_dtq(ID dtqid, intptr_t data)
8305     ER ercd = ipsnd_dtq(ID dtqid, intptr_t data)
8306     ER ercd = tsnd_dtq(ID dtqid, intptr_t data, TMO tmout)
8307
8308 【パラメータ】
8309     ID          dtqid      対象データキューのID番号
8310     intptr_t     data       送信データ
8311     TMO          tmout      タイムアウト時間 (tsnd_dtqの場合)
8312
8313 【リターンパラメータ】
8314     ER          ercd        正常終了 (E_OK) またはエラーコード
8315
8316 【エラーコード】
8317     E_CTX        コンテキストエラー
8318                 ・非タスクコンテキストからの呼出し (ipsnd_dtqを除く)
8319                 【NGKI1722】
8320                 ・タスクコンテキストからの呼出し (ipsnd_dtqの場合)
8321                 【NGKI1723】
8322                 ・CPUロック状態からの呼出し 【NGKI1724】
8323                 ・ディスパッチ保留状態からの呼出し (snd_dtqとtsnd_dtqの
8324                 場合) 【NGKI1725】
8325     E_NOSPT      未サポート機能
8326                 ・制約タスクからの呼出し (snd_dtqとtsnd_dtqの場合) 【NGKI1726】
8327     E_ID         不正ID番号
8328                 ・dtqidが有効範囲外 【NGKI1727】
8329     E_PAR        パラメータエラー
8330                 ・tmoutが無効 (tsnd_dtqの場合) 【NGKI1728】
8331     E_NOEXS      オブジェクト未登録
8332                 ・対象データキューが未登録 [D] 【NGKI1729】
8333     E_OACV       オブジェクトアクセス違反
8334                 ・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
8335                 (ipsnd_dtqを除く) [P] 【NGKI1730】
8336     E_TMOUT      ポーリング失敗またはタイムアウト (snd_dtqを除く) 【NGKI1731】
8337     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
8338                 の場合) 【NGKI1732】
8339     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
8340                 の場合) 【NGKI1733】
8341
8342 【機能】
8343
8344     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
8345     を送信する. 具体的な振舞いは以下の通り.
8346
8347     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
8348     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
8349     【NGKI1734】. 待ち解除されたタスクには、待ち状態となったサービスコール
8350     からE_OKが返る 【NGKI1735】.

```

8351
8352 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
8353 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
8354 順でデータキュー管理領域に格納される【NGKI1736】。
8355
8356 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
8357 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
8358 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
8359 【NGKI1737】。
8360 -----
8361 fsnd_dtq データキューへの強制送信 [T] 【NGKI1738】
8362 ifsnd_dtq データキューへの強制送信 [I] 【NGKI1739】
8363
8364 【C言語API】
8365 ER ercd = fsnd_dtq(ID dtqid, intptr_t data)
8366 ER ercd = ifsnd_dtq(ID dtqid, intptr_t data)
8367
8368 【パラメータ】
8369 ID dtqid 対象データキューのID番号
8370 intptr_t data 送信データ
8371
8372 【リターンパラメータ】
8373 ER ercd 正常終了 (E_OK) またはエラーコード
8374
8375 【エラーコード】
8376 E_CTX コンテキストエラー
8377 ・非タスクコンテキストからの呼出し (fsnd_dtqの場合) 【NGKI1740】
8378 ・タスクコンテキストからの呼出し (ifsnd_dtqの場合) 【NGKI1741】
8379 ・CPUロック状態からの呼出し 【NGKI1742】
8380 E_ID 不正ID番号
8381 ・dtqidが有効範囲外 【NGKI1743】
8382 E_NOEXS オブジェクト未登録
8383 ・対象データキューが未登録 [D] 【NGKI1744】
8384 E_OACV オブジェクトアクセス違反
8385 ・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
8386 (fsnd_dtqの場合) [P] 【NGKI1745】
8387 E_ILUSE サービスコール不正使用
8388 ・対象データキューのデータキュー管理領域のサイズが0 【NGKI1746】
8389
8390 【機能】
8391
8392 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
8393 を強制送信する。具体的な振舞いは以下の通り。
8394
8395 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
8396 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
8397 【NGKI1747】。待ち解除されたタスクには、待ち状態となったサービスコール
8398 からE_OKが返る【NGKI1748】。
8399
8400 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

8401 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
8402 順でデータキュー管理領域に格納される【NGKI1749】.

8403

8404 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
8405 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
8406 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
8407 が、FIFO順でデータキュー管理領域に格納される【NGKI1750】.

8408 -----

8409 rcv_dtq データキューからの受信 [T] 【NGKI1751】

8410 prcv_dtq データキューからの受信 (ポーリング) [T] 【NGKI1752】

8411 trcv_dtq データキューからの受信 (タイムアウト付き) [T] 【NGKI1753】

8412

8413 【C言語API】

8414 ER ercd = rcv_dtq(ID dtqid, intptr_t *p_data)

8415 ER ercd = prcv_dtq(ID dtqid, intptr_t *p_data)

8416 ER ercd = trcv_dtq(ID dtqid, intptr_t *p_data, TMO tmout)

8417

8418 【パラメータ】

8419 ID dtqid 対象データキューのID番号

8420 intptr_t * p_data 受信データを入れるメモリ領域へのポインタ

8421 TMO tmout タイムアウト時間 (trcv_dtqの場合)

8422

8423 【リターンパラメータ】

8424 ER ercd 正常終了 (E_OK) またはエラーコード

8425 intptr_t data 受信データ

8426

8427 【エラーコード】

8428 E_CTX コンテキストエラー

8429 ・非タスクコンテキストからの呼出し【NGKI1754】

8430 ・CPUロック状態からの呼出し【NGKI1755】

8431 ・ディスパッチ保留状態からの呼出し (prcv_dtqを除く)

8432 【NGKI1756】

8433 E_NOSPT 未サポート機能

8434 ・制約タスクからの呼出し (prcv_dtqを除く)【NGKI1757】

8435 E_ID 不正ID番号

8436 ・dtqidが有効範囲外【NGKI1758】

8437 E_PAR パラメータエラー

8438 ・tmoutが無効 (trcv_dtqの場合)【NGKI1759】

8439 E_NOEXS オブジェクト未登録

8440 ・対象データキューが未登録 [D] 【NGKI1760】

8441 E_OACV オブジェクトアクセス違反

8442 ・対象データキューに対する通常操作2が許可されていない [P]

8443 【NGKI1761】

8444 E_MACV メモリアクセス違反

8445 ・p_dataが指すメモリ領域への書込みアクセスが許可されて
8446 いない [P] 【NGKI1762】

8447 E_TMOUT ポーリング失敗またはタイムアウト (rcv_dtqを除く)【NGKI1763】

8448 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)

8449 【NGKI1764】

8450 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)

8451 【NGKI1765】

8452

8453 【機能】

8454

8455 dtqidで指定したデータキュー（対象データキュー）からデータを受信する．デー

8456 タの受信に成功した場合，受信したデータはp_dataが指すメモリ領域に返され

8457 る【NGKI3421】．具体的な振舞いは以下の通り．

8458

8459 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，

8460 データキュー管理領域の先頭に格納されたデータを受信する【NGKI1766】．ま

8461 た，送信待ち行列にタスクが存在する場合には，送信待ち行列の先頭のタスク

8462 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

8463 ち解除される【NGKI1767】．待ち解除されたタスクには，待ち状態となったサー

8464 ビスコールからE_OKが返る【NGKI1768】．

8465

8466 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

8467 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

8468 タを受信する【NGKI1769】．送信待ち行列の先頭のタスクは，待ち解除される

8469 【NGKI3422】．待ち解除されたタスクには，待ち状態となったサービスコール

8470 からE_OKが返る【NGKI1770】．

8471

8472 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

8473 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

8474 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【NGKI1771】．

8475 -----

8476 ini_dtq データキューの再初期化〔T〕【NGKI1772】

8477

8478 【C言語API】

8479 ER ercd = ini_dtq(ID dtqid)

8480

8481 【パラメータ】

8482 ID dtqid 対象データキューのID番号

8483

8484 【リターンパラメータ】

8485 ER ercd 正常終了（E_OK）またはエラーコード

8486

8487 【エラーコード】

8488 E_CTX コンテキストエラー

8489 ・非タスクコンテキストからの呼出し【NGKI1773】

8490 ・CPUロック状態からの呼出し【NGKI1774】

8491 E_ID 不正ID番号

8492 ・dtqidが有効範囲外【NGKI1775】

8493 E_NOEXS オブジェクト未登録

8494 ・対象データキューが未登録〔D〕【NGKI1776】

8495 E_OACV オブジェクトアクセス違反

8496 ・対象データキューに対する管理操作が許可されていない〔P〕

8497 【NGKI1777】

8498

8499 【機能】

8500

8501 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
8502 振舞いは以下の通り．

8503
8504 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
8505 態に初期化される【NGKI1778】．また，対象データキューの送信待ち行列と受
8506 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
8507 に待ち解除される【NGKI1779】．待ち解除されたタスクには，待ち状態となっ
8508 たサービスコールからE_DLTエラーが返る【NGKI1780】．

8509
8510 **【補足説明】**

8511
8512 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，
8513 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
8514 い．

8515
8516 **【使用上の注意】**

8517
8518 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
8519 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
8520 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
8521 み禁止時間が長くなるため，注意が必要である．

8522
8523 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，
8524 アプリケーションの責任である．

8525
8526 **【μ ITRON4.0仕様との関係】**

8527
8528 μ ITRON4.0仕様に定義されていないサービスコールである．
8529 -----

8530 ref_dtq データキューの状態参照 [T] 【NGKI1781】
8531

8532 **【C言語API】**

8533 ER ercd = ref_dtq(ID dtqid, T_RDTQ *pk_rdtq)

8534
8535 **【パラメータ】**

8536 ID dtqid 対象データキューのID番号
8537 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
8538 ポインタ

8539
8540 **【リターンパラメータ】**

8541 ER ercd 正常終了 (E_OK) またはエラーコード

8542
8543 *データキューの現在状態 (パケットの内容)

8544 ID stskid データキューの送信待ち行列の先頭のタスクの
8545 ID番号

8546 ID rtskid データキューの受信待ち行列の先頭のタスクの
8547 ID番号

8548 uint_t sdtqcnt データキュー管理領域に格納されているデータ
8549 の数

8550

8551 **【エラーコード】**
8552 E_CTX コンテキストエラー
8553 • 非タスクコンテキストからの呼出し【NGKI1782】
8554 • CPUロック状態からの呼出し【NGKI1783】
8555 E_ID 不正ID番号
8556 • dtqidが有効範囲外【NGKI1784】
8557 E_NOEXS オブジェクト未登録
8558 • 対象データキューが未登録〔D〕【NGKI1785】
8559 E_OACV オブジェクトアクセス違反
8560 • 対象データキューに対する参照操作が許可されていない〔P〕
8561 【NGKI1786】
8562 E_MACV メモリアクセス違反
8563 • pk_rdtqが指すメモリ領域への書き込みアクセスが許可されて
8564 いない〔P〕【NGKI1787】

8565 **【機能】**

8566
8567
8568 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
8569 照した現在状態は，pk_rdtqで指定したパケットに返される【NGKI1788】．

8570
8571 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
8572 TSK_NONE（=0）が返る【NGKI1789】．また，受信待ち行列にタスクが存在しな
8573 い場合，rtskidにはTSK_NONE（=0）が返る【NGKI1790】．

8574 **【使用上の注意】**

8575
8576
8577 ref_dttq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
8578 ない．これは，ref_dttqを呼び出し，対象データキューの現在状態を参照した直
8579 後に割り込みが発生した場合，ref_dttqから戻ってきた時には対象データキューの
8580 状態が変化している可能性があるためである．

8581 -----

8582 **4.4.4 優先度データキュー**

8583
8584
8585 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
8586 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
8587 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
8588 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
8589 は，優先度データキューIDと呼ぶID番号によって識別する【NGKI1791】．

8590
8591 各優先度データキューが持つ情報は次の通り【NGKI1792】．

- 8592
- 8593 • 優先度データキュー属性
 - 8594 • 優先度データキュー管理領域
 - 8595 • 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）
 - 8596 • 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）
 - 8597 • 送信できるデータ優先度の最大値
 - 8598 • アクセス許可バクタ（保護機能対応カーネルの場合）
 - 8599 • 属する保護ドメイン（保護機能対応カーネルの場合）
 - 8600 • 属するクラス（マルチプロセッサ対応カーネルの場合）

8601
8602 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、
8603 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
8604 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、
8605 優先度データキュー管理領域のサイズを0とすることができる【NGKI1793】。
8606
8607 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
8608 用いるオブジェクト管理領域として扱われる【NGKI1794】。
8609
8610 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
8611 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
8612 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
8613 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
8614 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
8615 キューである。
8616
8617 優先度データキュー属性には、次の属性を指定することができる【NGKI1795】。
8618
8619 TA_TPRI 0x01U 送信待ち行列をタスクの優先度順にする
8620
8621 TA_TPRIを指定しない場合、送信待ち行列はFIFO順になる【NGKI1796】。受信待
8622 ち行列は、FIFO順に固定されている【NGKI1797】。
8623
8624 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。
8625
8626 TMIN_DPRI データ優先度の最小値（=1） 【NGKI1798】
8627 TMAX_DPRI データ優先度の最大値
8628
8629 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
8630 いカーネルでは、静的APIによって登録された優先度デー
8631 タキューの数に一致）【NGKI1799】
8632
8633 【TOPPERS/ASPカーネルにおける規定】
8634
8635 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
8636 【ASPS0138】。ただし、タスク優先度拡張パッケージでは、TMAX_DPRIを256に
8637 拡張する【ASPS0139】。
8638
8639 【TOPPERS/FMPカーネルにおける規定】
8640
8641 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
8642 【FMPS0124】。
8643
8644 【TOPPERS/HRP2カーネルにおける規定】
8645
8646 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
8647 【HRPS0124】。
8648
8649 【使用上の注意】
8650

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

【μITRON4.0仕様との関係】

μITRON4.0仕様に規定されていない機能である。

CRE_PDQ 優先度データキューの生成 [S] 【NGKI1800】
acre_pdq 優先度データキューの生成 [TD] 【NGKI1801】

【静的API】

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

【C言語API】

ER_ID pdqid = acre_pdq(const T_CPDQ *pk_cpdq)

【パラメータ】

ID	pdqid	生成する優先度データキューのID番号（CRE_PDQの場合）
T_CPDQ *	pk_cpdq	優先度データキューの生成情報を入れたパケットへのポインタ（静的APIを除く）

*優先度データキューの生成情報（パケットの内容）

ATR	pdqatr	優先度データキュー属性
uint_t	pdqcnt	優先度データキュー管理領域に格納できるデータ数
PRI	maxdpri	優先度データキューに送信できるデータ優先度の最大値
void *	pdqmb	優先度データキュー管理領域の先頭番地

【リターンパラメータ】

ER_ID	pdqid	生成された優先度データキューのID番号（正の値）またはエラーコード
-------	-------	-----------------------------------

【エラーコード】

E_CTX	コンテキストエラー
	・非タスクコンテキストからの呼出し [s] 【NGKI1802】
	・CPUロック状態からの呼出し [s] 【NGKI1803】
E_RSATR	予約属性
	・pdqatrが無効 【NGKI1804】
	・属する保護ドメインの指定が有効範囲外 [sP] 【NGKI1805】
	・属するクラスの指定が有効範囲外 [sM] 【NGKI1806】
	・クラスの囲みの中に記述されていない [SM] 【NGKI1807】
E_NOSPT	未サポート機能

8701		・条件については各カーネルにおける規定の項を参照
8702	E_PAR	パラメータエラー
8703		・pdqcntが負の値 [S] 【NGKI3289】
8704		・maxdpriがTMIN_DPRIより小さい, またはTMAX_DPRIより大きい 【NGKI1819】
8705		
8706		・その他の条件については機能の項を参照
8707	E_OACV	オブジェクトアクセス違反
8708		・システム状態に対する管理操作が許可されていない [sP] 【NGKI1808】
8709		
8710	E_MACV	メモリアクセス違反
8711		・pk_cpdqが指すメモリ領域への読出しアクセスが許可されていない [sP] 【NGKI1809】
8712		
8713	E_NOID	ID番号不足
8714		・割り付けられる優先度データキューIDがない [sD] 【NGKI1810】
8715	E_NOMEM	メモリ不足
8716		・優先度データキュー管理領域が確保できない 【NGKI1811】
8717	E_OBJ	オブジェクト状態エラー
8718		・pdqidで指定した優先度データキューが登録済み (CRE_PDQの場合) 【NGKI1812】
8719		
8720		・その他の条件については機能の項を参照
8721		
8722	【機能】	
8723		
8724	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	
8725	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,	
8726	格納されているデータがない状態に初期化される 【NGKI1813】. また, 送信待ち	
8727	行列と受信待ち行列は, 空の状態に初期化される 【NGKI1814】.	
8728		
8729	静的APIにおいては, pdqidはオブジェクト識別名, pdqatr, pdqcnt, maxdpriは	
8730	整数定数式パラメータ, pdqmbは一般定数式パラメータである 【NGKI1815】. コ	
8731	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	
8732	ができない 【NGKI1816】.	
8733		
8734	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	
8735	タキュー管理領域が, コンフィギュレータまたはカーネルにより確保される	
8736	【NGKI1817】.	
8737		
8738	[pdqmbにNULL以外を指定した場合]	
8739		
8740	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	
8741	管理領域は, アプリケーションで確保しておく必要がある 【NGKI1820】. 優先	
8742	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	
8743	用意している 【NGKI1821】.	
8744		
8745	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
8746		タキュー管理領域のサイズ (バイト数)
8747	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
8748		タキュー管理領域を確保するために必要なMB_T型の
8749		配列の要素数
8750		

8751 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
8752 【NGKI1822】 .
8753
8754 MB_T <優先度データキュー管理領域の変数名>[TCNT_PDQMB(pdqcnt)];
8755
8756 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
8757 【NGKI1823】 .
8758
8759 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
8760 した時には、E_PARエラーとなる【NGKI1824】 . また、保護機能対応カーネルに
8761 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
8762 オブジェクトに含まれない場合、E_OBJエラーとなる【NGKI1825】 .
8763
8764 【TOPPERS/ASPカーネルにおける規定】
8765
8766 ASPカーネルでは、CRE_PDQのみをサポートする【ASPS0140】 . また、pdqmbには
8767 NULLのみを指定することができる . NULL以外を指定した場合には、E_NOSPTエラー
8768 となる【ASPS0142】 . ただし、動的生成機能拡張パッケージでは、acre_pdqも
8769 サポートする【ASPS0143】 . acre_pdqに対しては、pdqmbにNULL以外を指定でき
8770 ないという制限はない【ASPS0144】 .
8771
8772 【TOPPERS/FMPカーネルにおける規定】
8773
8774 FMPカーネルでは、CRE_PDQのみをサポートする【FMPS0125】 . また、pdqmbには
8775 NULLのみを指定することができる . NULL以外を指定した場合には、E_NOSPTエラー
8776 となる【FMPS0127】 .
8777
8778 【TOPPERS/HRP2カーネルにおける規定】
8779
8780 HRP2カーネルでは、CRE_PDQのみをサポートする【HRPS0125】 . また、pdqmbに
8781 はNULLのみを指定することができる . NULL以外を指定した場合には、E_NOSPTエ
8782 ラーとなる【HRPS0127】 . ただし、動的生成機能拡張パッケージでは、
8783 acre_pdqもサポートする【HRPS0190】 . acre_pdqに対しては、pdqmbにNULL以外
8784 を指定できないという制限はない【HRPS0191】 .
8785 -----
8786 AID_PDQ 割付け可能な優先度データキューIDの数の指定 [SD] 【NGKI1826】
8787
8788 【静的API】
8789 AID_PDQ(uint_t nopdq)
8790
8791 【パラメータ】
8792 uint_t nopdq 割付け可能な優先度データキューIDの数
8793
8794 【エラーコード】
8795 E_RSATR 予約属性
8796 ・保護ドメインの囲みの中に記述されている [P] 【NGKI3432】
8797 ・クラスの囲みの中に記述されていない [M] 【NGKI1827】
8798 E_PAR パラメータエラー
8799 ・nopdqが負の値【NGKI3280】
8800

8851 (対象優先度データキューが無所属の場合は、保護ドメイン
8852 の囲みの外)に記述されていない [S] 【NGKI1835】
8853 ・対象優先度データキューが属するクラスの囲みの中に記述
8854 されていない [SM] 【NGKI1836】
8855 E_NOEXS オブジェクト未登録
8856 ・対象優先度データキューが未登録 【NGKI1837】
8857 E_OACV オブジェクトアクセス違反
8858 ・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
8859 ない [s] 【NGKI1838】
8860 E_MACV メモリアクセス違反
8861 ・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
8862 いない [s] 【NGKI1839】
8863 E_OBJ オブジェクト状態エラー
8864 ・対象優先度データキューは静的APIで生成された [s] 【NGKI1840】
8865 ・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
8866 定済み [S] 【NGKI1841】
8867

8868 【機能】

8869
8870 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
8871 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
8872 設定する 【NGKI1842】 .
8873

8874 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
8875 式パラメータである 【NGKI1843】 .
8876

8877 【TOPPERS/HRP2カーネルにおける規定】

8878
8879 HRP2カーネルでは、SAC_PDQのみをサポートする 【HRPS0128】 . ただし、動的生成
8880 機能拡張パッケージでは、sac_pdqもサポートする 【HRPS0192】 .
8881

8882 del_pdq 優先度データキューの削除 [TD] 【NGKI1844】
8883

8884 【C言語API】

8885 ER ercd = del_pdq(ID pdqid)
8886

8887 【パラメータ】

8888 ID pdqid 対象優先度データキューのID番号
8889

8890 【リターンパラメータ】

8891 ER ercd 正常終了 (E_OK) またはエラーコード
8892

8893 【エラーコード】

8894 E_CTX コンテキストエラー
8895 ・非タスクコンテキストからの呼出し 【NGKI1845】
8896 ・CPUロック状態からの呼出し 【NGKI1846】
8897 E_ID 不正ID番号
8898 ・pdqidが有効範囲外 【NGKI1847】
8899 E_NOEXS オブジェクト未登録
8900 ・対象優先度データキューが未登録 【NGKI1848】

8901 E_OACV オブジェクトアクセス違反
8902 ・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
8903 ない [P] 【NGKI1849】
8904 E_OBJ オブジェクト状態エラー
8905 ・対象優先度データキューは静的APIで生成された 【NGKI1850】
8906
8907 **【機能】**
8908
8909 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。
8910 具体的な振舞いは以下の通り。
8911
8912 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
8913 用の状態に戻される 【NGKI1851】。また、対象優先度データキューの送信待ち
8914 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
8915 クから順に待ち解除される 【NGKI1852】。待ち解除されたタスクには、待ち状
8916 態となったサービスコールからE_DLTエラーが返る 【NGKI1853】。
8917
8918 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
8919 て確保された場合は、そのメモリ領域が解放される 【NGKI1854】。
8920
8921 **【補足説明】**
8922
8923 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、
8924 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
8925 い。
8926
8927 **【使用上の注意】**
8928
8929 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
8930 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
8931 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
8932 み禁止時間が長くなるため、注意が必要である。
8933
8934 **【TOPPERS/ASPカーネルにおける規定】**
8935
8936 ASPカーネルでは、del_pdqをサポートしない 【ASPS0146】。ただし、動的生成
8937 機能拡張パッケージでは、del_pdqをサポートする 【ASPS0147】。
8938
8939 **【TOPPERS/FMPカーネルにおける規定】**
8940
8941 FMPカーネルでは、del_pdqをサポートしない 【FMPS0129】。
8942
8943 **【TOPPERS/HRP2カーネルにおける規定】**
8944
8945 HRP2カーネルでは、del_pdqをサポートしない 【HRPS0129】。ただし、動的生成
8946 機能拡張パッケージでは、del_pdqをサポートする 【HRPS0193】。
8947 -----
8948 snd_pdq 優先度データキューへの送信 [T] 【NGKI1855】
8949 psnd_pdq 優先度データキューへの送信（ポーリング） [T] 【NGKI1856】
8950 ipsnd_pdq 優先度データキューへの送信（ポーリング） [I] 【NGKI1857】

8951 tsnd_pdq 優先度データキューへの送信（タイムアウト付き） [T] 【NGKI1858】
8952
8953 **【C言語API】**
8954 ER ercd = snd_pdq(ID pdqid, intptr_t data, PRI datapri)
8955 ER ercd = psnd_pdq(ID pdqid, intptr_t data, PRI datapri)
8956 ER ercd = ipsnd_pdq(ID pdqid, intptr_t data, PRI datapri)
8957 ER ercd = tsnd_pdq(ID pdqid, intptr_t data, PRI datapri, TMO tmout)
8958
8959 **【パラメータ】**
8960 ID pdqid 対象優先度データキューのID番号
8961 intptr_t data 送信データ
8962 PRI datapri 送信データの優先度
8963 TMO tmout タイムアウト時間（tsnd_pdqの場合）
8964
8965 **【リターンパラメータ】**
8966 ER ercd 正常終了（E_OK）またはエラーコード
8967
8968 **【エラーコード】**
8969 E_CTX コンテキストエラー
8970 ・非タスクコンテキストからの呼出し（ipsnd_pdqを除く）
8971 **【NGKI1859】**
8972 ・タスクコンテキストからの呼出し（ipsnd_pdqの場合） **【NGKI1860】**
8973 ・CPUロック状態からの呼出し **【NGKI1861】**
8974 ・ディスパッチ保留状態からの呼出し（snd_pdqとtsnd_pdqの
8975 場合） **【NGKI1862】**
8976 E_NOSPT 未サポート機能
8977 ・制約タスクからの呼出し（snd_pdqとtsnd_pdqの場合） **【NGKI1863】**
8978 E_ID 不正ID番号
8979 ・pdqidが有効範囲外 **【NGKI1864】**
8980 E_PAR パラメータエラー
8981 ・tmoutが無効（tsnd_pdqの場合） **【NGKI1865】**
8982 ・その他の条件については機能の項を参照
8983 E_NOEXS オブジェクト未登録
8984 ・対象優先度データキューが未登録 [D] **【NGKI1866】**
8985 E_OACV オブジェクトアクセス違反
8986 ・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
8987 ない（ipsnd_pdqを除く） [P] **【NGKI1867】**
8988 E_TMOUT ポーリング失敗またはタイムアウト（snd_pdqを除く） **【NGKI1868】**
8989 E_RLWAI 待ち禁止状態または待ち状態の強制解除（snd_pdqとtsnd_pdq
8990 の場合） **【NGKI1869】**
8991 E_DLT 待ちオブジェクトの削除または再初期化（snd_pdqとtsnd_pdq
8992 の場合） **【NGKI1870】**
8993
8994 **【機能】**
8995
8996 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
8997 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
8998 の通り。
8999
9000 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

9001 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
9002 【NGKI1871】。待ち解除されたタスクには、待ち状態となったサービスコール
9003 からE_OKが返る【NGKI1872】。
9004
9005 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
9006 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
9007 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
9008 に格納される【NGKI1873】。
9009
9010 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
9011 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
9012 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
9013 行列につながる【NGKI1874】。
9014
9015 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
9016 最大値以下でなければならない。そうでない場合には、E_PARエラーとなる
9017 【NGKI1876】。
9018 -----

9019 rcv_pdq 優先度データキューからの受信 [T] 【NGKI1877】
9020 prcv_pdq 優先度データキューからの受信（ポーリング） [T] 【NGKI1878】
9021 trcv_pdq 優先度データキューからの受信（タイムアウト付き） [T] 【NGKI1879】
9022

9023 【C言語API】
9024 ER ercd = rcv_pdq(ID pdqid, intptr_t *p_data, PRI *p_datapri)
9025 ER ercd = prcv_pdq(ID pdqid, intptr_t *p_data, PRI *p_datapri)
9026 ER ercd = trcv_pdq(ID pdqid, intptr_t *p_data, PRI *p_datapri, TMO tmout)
9027

9028 【パラメータ】
9029 ID pdqid 対象優先度データキューのID番号
9030 intptr_t * p_data 受信データを入れるメモリ領域へのポインタ
9031 PRI * p_datapri 受信データの優先度を入れるメモリ領域へのポ
9032 インタ
9033 TMO tmout タイムアウト時間（trcv_pdqの場合）
9034

9035 【リターンパラメータ】
9036 ER ercd 正常終了（E_OK）またはエラーコード
9037 intptr_t data 受信データ
9038 PRI datapri 受信データの優先度
9039

9040 【エラーコード】
9041 E_CTX コンテキストエラー
9042 ・非タスクコンテキストからの呼出し【NGKI1880】
9043 ・CPUロック状態からの呼出し【NGKI1881】
9044 ・ディスパッチ保留状態からの呼出し（prcv_pdqを除く）【NGKI1882】
9045 E_NOSPT 未サポート機能
9046 ・制約タスクからの呼出し（prcv_pdqを除く）【NGKI1883】
9047 E_ID 不正ID番号
9048 ・pdqidが有効範囲外【NGKI1884】
9049 E_PAR パラメータエラー
9050 ・tmoutが無効（trcv_pdqの場合）【NGKI1885】

9051 E_NOEXS オブジェクト未登録
 9052 ・対象優先度データキューが未登録 [D] 【NGKI1886】
 9053 E_OACV オブジェクトアクセス違反
 9054 ・対象優先度データキューに対する通常操作2が許可されてい
 9055 ない [P] 【NGKI1887】
 9056 E_MACV メモリアクセス違反
 9057 ・p_dataが指すメモリ領域への書込みアクセスが許可されて
 9058 いない [P] 【NGKI1888】
 9059 ・p_datapriが指すメモリ領域への書込みアクセスが許可され
 9060 ていない [P] 【NGKI1889】
 9061 E_TMOUT ポーリング失敗またはタイムアウト (rcv_pdqを除く) 【NGKI1890】
 9062 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)
 9063 【NGKI1891】
 9064 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)
 9065 【NGKI1892】
 9066

9067 【機能】

9068
 9069 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
 9070 受信する．データの受信に成功した場合、受信したデータはp_dataが指すメモ
 9071 リ領域に、その優先度はp_datapriが指すメモリ領域に返される 【NGKI1894】．
 9072 具体的な振舞いは以下の通り．
 9073

9074 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
 9075 いる場合には、優先度データキュー管理領域の先頭に格納されたデータを受信
 9076 する 【NGKI1893】．また、送信待ち行列にタスクが存在する場合には、送信待
 9077 ち行列の先頭のタスクの送信データが、データの優先度順で優先度データキュー
 9078 管理領域に格納され、そのタスクは待ち解除される 【NGKI1895】．待ち解除さ
 9079 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
 9080 【NGKI1896】．
 9081

9082 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
 9083 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
 9084 スクの送信データを受信する 【NGKI1897】．送信待ち行列の先頭のタスクは、
 9085 待ち解除される 【NGKI1899】．待ち解除されたタスクには、待ち状態となった
 9086 サービスコールからE_OKが返る 【NGKI1900】．
 9087

9088 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
 9089 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
 9090 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
 9091 につながる 【NGKI1901】．
 9092

9093 -----
 9093 ini_pdq 優先度データキューの再初期化 [T] 【NGKI1902】
 9094

9095 【C言語API】

9096 ER ercd = ini_pdq(ID pdqid)
 9097

9098 【パラメータ】

9099 ID pdqid 対象優先度データキューのID番号
 9100

9101 【リターンパラメータ】
9102 ER ercd 正常終了 (E_OK) またはエラーコード
9103
9104 【エラーコード】
9105 E_CTX コンテキストエラー
9106 ・非タスクコンテキストからの呼出し【NGKI1903】
9107 ・CPUロック状態からの呼出し【NGKI1904】
9108 E_ID 不正ID番号
9109 ・pdqidが有効範囲外【NGKI1905】
9110 E_NOEXS オブジェクト未登録
9111 ・対象優先度データキューが未登録 [D] 【NGKI1906】
9112 E_OACV オブジェクトアクセス違反
9113 ・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
9114 ない [P] 【NGKI1907】
9115
9116 【機能】
9117
9118 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
9119 る．具体的な振舞いは以下の通り．
9120
9121 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
9122 タがない状態に初期化される【NGKI1908】．また、対象優先度データキューの
9123 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
9124 頭のタスクから順に待ち解除される【NGKI1909】．待ち解除されたタスクには、
9125 待ち状態となったサービスコールからE_DLTエラーが返る【NGKI1910】．
9126
9127 【補足説明】
9128
9129 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、
9130 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
9131 い．
9132
9133 【使用上の注意】
9134
9135 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
9136 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
9137 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
9138 み禁止時間が長くなるため、注意が必要である．
9139
9140 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
9141 つのは、アプリケーションの責任である．
9142 -----
9143 ref_pdq 優先度データキューの状態参照 [T] 【NGKI1911】
9144
9145 【C言語API】
9146 ER ercd = ref_pdq(ID pdqid, T_RPDQ *pk_rpdq)
9147
9148 【パラメータ】
9149 ID pdqid 対象優先度データキューのID番号
9150 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

9151 トへのポインタ

9152

9153 【リターンパラメータ】

9154 ER ercd 正常終了 (E_OK) またはエラーコード

9155

9156 * 優先度データキューの現在状態 (パケットの内容)

9157 ID stskid 優先度データキューの送信待ち行列の先頭のタ
9158 スクのID番号

9159 ID rtskid 優先度データキューの受信待ち行列の先頭のタ
9160 スクのID番号

9161 uint_t spdqcnt 優先度データキュー管理領域に格納されている
9162 データの数

9163

9164 【エラーコード】

9165 E_CTX コンテキストエラー

9166 ・非タスクコンテキストからの呼出し【NGKI1912】

9167 ・CPUロック状態からの呼出し【NGKI1913】

9168 E_ID 不正ID番号

9169 ・pdqidが有効範囲外【NGKI1914】

9170 E_NOEXS オブジェクト未登録

9171 ・対象優先度データキューが未登録 [D]【NGKI1915】

9172 E_OACV オブジェクトアクセス違反

9173 ・対象優先度データキューに対する参照操作が許可されてい
9174 ない [P]【NGKI1916】

9175 E_MACV メモリアクセス違反

9176 ・pk_rpdqが指すメモリ領域への書込みアクセスが許可されて
9177 いない [P]【NGKI1917】

9178

9179 【機能】

9180

9181 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
9182 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

9183 【NGKI1918】.

9184

9185 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
9186 はTSK_NONE (=0) が返る【NGKI1919】. また, 受信待ち行列にタスクが存在し
9187 ない場合, rtskidにはTSK_NONE (=0) が返る【NGKI1920】.

9188

9189 【使用上の注意】

9190

9191 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
9192 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
9193 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
9194 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.

9195

9196

9197 4.4.5 メールボックス

9198

9199 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
9200 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

9201 クスは、メールボックスIDと呼ぶID番号によって識別する【NGKI1921】。

9202

9203 各メールボックスが持つ情報は次の通り【NGKI1922】。

9204

9205 ・メールボックス属性

9206 ・メッセージキュー

9207 ・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）

9208 ・送信できるメッセージ優先度の最大値

9209 ・優先度別のメッセージキューヘッダ領域

9210 ・属するクラス（マルチプロセッサ対応カーネルの場合）

9211

9212 メッセージキューは、メールボックスに送信されたメッセージを、FIFO順または
9213 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。

9214

9215 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
9216 （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
9217 順序でつながれているキューである。

9218

9219 メールボックス属性には、次の属性を指定することができる【NGKI1923】。

9220

9221 TA_TPRI 0x01U 待ち行列をタスクの優先度順にする

9222 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

9223

9224 TA_TPRIを指定しない場合、待ち行列はFIFO順になる【NGKI1924】。TA_MPRIを
9225 指定しない場合、メッセージキューはFIFO順になる【NGKI1925】。

9226

9227 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
9228 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。

9229

9230 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
9231 なぐために、メッセージの先頭のメモリ領域を使用する【NGKI1926】。そのた
9232 めアプリケーションは、メールボックスに送信するメッセージの先頭に、カー
9233 ネルが利用するためのメッセージヘッダを置かなければならない【NGKI1927】。
9234 メッセージヘッダのデータ型として、メールボックス属性にTA_MPRIが指定され
9235 ているか否かにより、以下のいずれかを用いる【NGKI1928】。

9236

9237 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

9238 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

9239

9240 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
9241 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
9242 によって使用される【NGKI1929】。そのため、メッセージキューにつながれて
9243 いるメッセージのメッセージヘッダの領域をアプリケーションが書き換えた場
9244 合や、メッセージキューにつながれているメッセージを再度メールボックスに
9245 送信した場合の動作は保証されない【NGKI1930】。

9246

9247 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
9248 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
9249 ルドに設定する【NGKI1931】。

9250

9251 保護機能対応カーネルでは、メールボックス機能はサポートしない【NGKI1932】。

9252

9253 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。

9254

9255 TMIN_MPRI メッセージ優先度の最小値 (=1) 【NGKI1933】

9256 TMAX_MPRI メッセージ優先度の最大値

9257

9258 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
9259 ネルでは、静的APIによって登録されたメールボックスの
9260 数に一致）【NGKI1934】

9261

9262 【補足説明】

9263

9264 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
9265 ダ領域は用いていない。

9266

9267 【使用上の注意】

9268

9269 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、
9270 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。

9271 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
9272 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
9273 する方法で置き換えることができる。

9274

9275 【TOPPERS/ASPカーネルにおける規定】

9276

9277 ASPカーネルでは、メールボックス機能をサポートする【ASPS0147】。メッセー
9278 ジ優先度の最大値（TMAX_MPRI）は16に固定されている【ASPS0148】。ただし、
9279 タスク優先度拡張パッケージでは、TMAX_MPRIを256に拡張する【ASPS0149】。

9280

9281 【TOPPERS/FMPカーネルにおける規定】

9282

9283 FMPカーネルでは、メールボックス機能をサポートする【FMPS0130】。メッセー
9284 ジ優先度の最大値（TMAX_MPRI）は16に固定されている【FMPS0131】。

9285

9286 【TOPPERS/HRP2カーネルにおける規定】

9287

9288 HRP2カーネルでは、メールボックス機能をサポートしない【HRPS0130】。

9289

9290 【 μ ITRON4.0仕様との関係】

9291

9292 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。

9293

9294 CRE_MBX メールボックスの生成 [Sp] 【NGKI1935】

9295 acre_mbx メールボックスの生成 [TpD] 【NGKI1936】

9296

9297 【静的API】

9298 CRE_MBX(ID mbxid, { ATR mbxatr, PRI maxmpri, void *mprihd })

9299

9300 【C言語API】

9301 ER_ID mbxid = acre_mbx(const T_CMBX *pk_cmbx)

9302

9303 **【パラメータ】**

9304 ID mbxid 生成するメールボックスのID番号 (CRE_MBXの場合)
9305
9306 T_CMBX * pk_cmbx メールボックスの生成情報を入れたパケットへのポインタ (静的APIを除く)
9307

9308

9309 * メールボックスの生成情報 (パケットの内容)

9310 ATR mbxatr メールボックス属性
9311 PRI maxmpri 優先度メールボックスに送信できるメッセージ優先度の最大値
9312
9313 void * mprihd 優先度別のメッセージキューヘッダ領域の先頭番地
9314

9315

9316 **【リターンパラメータ】**

9317 ER_ID mbxid 生成されたメールボックスのID番号 (正の値) またはエラーコード
9318

9319

9320 **【エラーコード】**

9321 E_CTX コンテキストエラー
9322 ・非タスクコンテキストからの呼出し [s] 【NGKI1937】
9323 ・CPUロック状態からの呼出し [s] 【NGKI1938】
9324 E_RSATR 予約属性
9325 ・mbxatrが無効 【NGKI1939】
9326 ・属するクラスの指定が有効範囲外 [sM] 【NGKI1940】
9327 ・クラスの囲みの中に記述されていない [SM] 【NGKI1941】
9328 E_NOSPT 未サポート機能
9329 ・条件については各カーネルにおける規定の項を参照
9330 E_PAR パラメータエラー
9331 ・maxmpriがTMIN_MPRIより小さい, またはTMAX_MPRIより大きい 【NGKI1951】
9332
9333 E_NOID ID番号不足
9334 ・割り付けられるメールボックスIDがない [sD] 【NGKI1942】
9335 E_NOMEM メモリ不足
9336 ・優先度別のメッセージキューヘッダ領域が確保できない 【NGKI1943】
9337
9338 E_OBJ オブジェクト状態エラー
9339 ・mbxidで指定したメールボックスが登録済み (CRE_MBXの場合) 【NGKI1944】
9340

9341

9342 **【機能】**

9343

9344 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
9345 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
9346 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
9347 る 【NGKI1945】. また, 待ち行列は空の状態に初期化される 【NGKI1946】.

9348

9349 静的APIにおいては, mbxidはオブジェクト識別名, mbxatrとmaxmpriは整数定数
9350 式パラメータ, mprihdは一般定数式パラメータである 【NGKI1947】. コンフィ

9351 ギューレータは、静的APIのメモリ不足 (E_NOMEM) エラーを検出することができ
9352 ない【NGKI1948】。

9353

9354 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
9355 ジキューヘッダ領域が、コンフィギュレータまたはカーネルにより確保される
9356 【NGKI1949】。

9357

9358 【未決定事項】

9359

9360 mprihdにNULL以外を指定した場合の扱いについては、この仕様では規定してい
9361 ない。

9362

9363 【TOPPERS/ASPカーネルにおける規定】

9364

9365 ASPカーネルでは、CRE_MBXのみをサポートする【ASPS0150】。また、優先度別
9366 のメッセージキューヘッダ領域は使用しておらず、mprihdにはNULLのみを指定
9367 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
9368 【ASPS0152】。ただし、動的生成機能拡張パッケージでは、acre_mbxもサポー
9369 トする【ASPS0153】。acre_mbxに対しても、mprihdにはNULLのみを指定するこ
9370 とができる【ASPS0154】。優先度別のメッセージキューヘッダ領域を使用しな
9371 いため、E_NOMEMが返ることはない【ASPS0155】。

9372

9373 【TOPPERS/FMPカーネルにおける規定】

9374

9375 FMPカーネルでは、CRE_MBXのみをサポートする【FMPS0132】。また、優先度別
9376 のメッセージキューヘッダ領域は使用しておらず、mprihdにはNULLのみを指定
9377 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
9378 【FMPS0134】。優先度別のメッセージキューヘッダ領域を使用しないため、
9379 E_NOMEMが返ることはない【FMPS0135】。

9380

9381 AID_MBX 割付け可能なメールボックスIDの数の指定 [SpD] 【NGKI1952】

9382

9383 【静的API】

9384 AID_MBX(uint_t nombx)

9385

9386 【パラメータ】

9387 uint_t nombx 割付け可能なメールボックスIDの数

9388

9389 【エラーコード】

9390 E_RSATR 予約属性

9391 ・クラスの囲みの中に記述されていない [M] 【NGKI1953】

9392 E_PAR パラメータエラー

9393 ・nombxが負の値【NGKI3281】

9394

9395 【機能】

9396

9397 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
9398 コールによって割付け可能なメールボックスIDとして確保する【NGKI1954】。

9399

9400 nombxは整数定数式パラメータである【NGKI1955】。

9401
9402 【TOPPERS/ASPカーネルにおける規定】
9403
9404 ASPカーネルの動的生成機能拡張パッケージでは、AID_MBXをサポートする
9405 【ASPS0215】。
9406 -----
9407 del_mbx メールボックスの削除 [TpD] 【NGKI1956】
9408
9409 【C言語API】
9410 ER ercd = del_mbx(ID mbxid)
9411
9412 【パラメータ】
9413 ID mbxid 対象メールボックスのID番号
9414
9415 【リターンパラメータ】
9416 ER ercd 正常終了 (E_OK) またはエラーコード
9417
9418 【エラーコード】
9419 E_CTX コンテキストエラー
9420 ・非タスクコンテキストからの呼出し【NGKI1957】
9421 ・CPUロック状態からの呼出し【NGKI1958】
9422 E_ID 不正ID番号
9423 ・mbxidが有効範囲外【NGKI1959】
9424 E_NOEXS オブジェクト未登録
9425 ・対象メールボックスが未登録【NGKI1960】
9426 E_OBJ オブジェクト状態エラー
9427 ・対象メールボックスは静的APIで生成された【NGKI1961】
9428
9429 【機能】
9430
9431 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
9432 振舞いは以下の通り。
9433
9434 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
9435 に戻される【NGKI1962】。また、対象メールボックスの待ち行列につながれた
9436 タスクは、待ち行列の先頭のタスクから順に待ち解除される【NGKI1963】。待
9437 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
9438 返る【NGKI1964】。
9439
9440 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
9441 ルによって確保された場合は、そのメモリ領域が解放される【NGKI1965】。
9442
9443 【使用上の注意】
9444
9445 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
9446 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
9447 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
9448 み禁止時間が長くなるため、注意が必要である。
9449
9450 【TOPPERS/ASPカーネルにおける規定】

9451
 9452 ASPカーネルでは、del_mbxをサポートしない【ASPS0156】。ただし、動的生成
 9453 機能拡張パッケージでは、del_mbxをサポートする【ASPS0157】。
 9454
 9455 【TOPPERS/FMPカーネルにおける規定】
 9456
 9457 FMPカーネルでは、del_mbxをサポートしない【FMPS0136】。
 9458 -----
 9459 snd_mbx メールボックスへの送信 [Tp] 【NGKI1966】
 9460
 9461 【C言語API】
 9462 ER ercd = snd_mbx(ID mbxid, T_MSG *pk_msg)
 9463
 9464 【パラメータ】
 9465 ID mbxid 対象メールボックスのID番号
 9466 T_MSG *pk_msg 送信メッセージの先頭番地
 9467
 9468 【リターンパラメータ】
 9469 ER ercd 正常終了 (E_OK) またはエラーコード
 9470
 9471 【エラーコード】
 9472 E_CTX コンテキストエラー
 9473 ・非タスクコンテキストからの呼出し【NGKI1967】
 9474 ・CPUロック状態からの呼出し【NGKI1968】
 9475 E_ID 不正ID番号
 9476 ・mbxidが有効範囲外【NGKI1969】
 9477 E_PAR パラメータエラー
 9478 ・条件については機能の項を参照
 9479 E_NOEXS オブジェクト未登録
 9480 ・対象メールボックスが未登録 [D] 【NGKI1970】
 9481
 9482 【機能】
 9483
 9484 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
 9485 メッセージを送信する。具体的な振舞いは以下の通り。
 9486
 9487 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
 9488 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
 9489 【NGKI1971】。待ち解除されたタスクには、待ち状態となったサービスコール
 9490 からE_OKが返る【NGKI1972】。
 9491
 9492 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
 9493 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
 9494 る順序で、メッセージキューにつながる【NGKI1973】。
 9495
 9496 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
 9497 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、
 9498 対象メールボックスに送信できるメッセージ優先度の最大値以下でなければなら
 9499 ない。そうでない場合には、E_PARエラーとなる【NGKI1975】。
 9500 -----

9501 rcv_mbx メールボックスからの受信 [Tp] 【NGKI1976】
9502 prcv_mbx メールボックスからの受信（ポーリング） [Tp] 【NGKI1977】
9503 trcv_mbx メールボックスからの受信（タイムアウト付き） [Tp] 【NGKI1978】
9504

【C言語API】

9506 ER ercd = rcv_mbx(ID mbxid, T_MSG **ppk_msg)
9507 ER ercd = prcv_mbx(ID mbxid, T_MSG **ppk_msg)
9508 ER ercd = trcv_mbx(ID mbxid, T_MSG **ppk_msg, TMO tmout)
9509

【パラメータ】

9511 ID mbxid 対象メールボックスのID番号
9512 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
9513 へのポインタ
9514 TMO tmout タイムアウト時間（trcv_mbxの場合）
9515

【リターンパラメータ】

9517 ER ercd 正常終了（E_OK）またはエラーコード
9518 T_MSG * ppk_msg 受信メッセージの先頭番地
9519

【エラーコード】

9521 E_CTX コンテキストエラー
9522 ・非タスクコンテキストからの呼出し 【NGKI1979】
9523 ・CPUロック状態からの呼出し 【NGKI1980】
9524 ・ディスパッチ保留状態からの呼出し（prcv_mbxを除く） 【NGKI1981】
9525 E_NOSPT 未サポート機能
9526 ・制約タスクからの呼出し（prcv_mbxを除く） 【NGKI1982】
9527 E_ID 不正ID番号
9528 ・mbxidが有効範囲外 【NGKI1983】
9529 E_PAR パラメータエラー
9530 ・tmoutが無効（trcv_mbxの場合） 【NGKI1984】
9531 E_NOEXS オブジェクト未登録
9532 ・対象メールボックスが未登録 [D] 【NGKI1985】
9533 E_TMOUT ポーリング失敗またはタイムアウト（rcv_mbxを除く） 【NGKI1986】
9534 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）
9535 【NGKI1987】
9536 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）
9537 【NGKI1988】
9538

【機能】

9540
9541 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
9542 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
9543 れる。具体的な振舞いは以下の通り。
9544
9545 対象メールボックスのメッセージキューにメッセージがつながれている場合に
9546 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
9547 指定したメモリ領域に返される 【NGKI1989】。
9548
9549 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
9550 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

9551 タスの待ち行列につなされる【NGKI1990】.

9552 -----

9553 ini_mbx メールボックスの再初期化 [Tp] 【NGKI1991】

9554

9555 【C言語API】

9556 ER ercd = ini_mbx(ID mbxid)

9557

9558 【パラメータ】

9559 ID mbxid 対象メールボックスのID番号

9560

9561 【リターンパラメータ】

9562 ER ercd 正常終了 (E_OK) またはエラーコード

9563

9564 【エラーコード】

9565 E_CTX コンテキストエラー

9566 • 非タスクコンテキストからの呼出し【NGKI1992】

9567 • CPUロック状態からの呼出し【NGKI1993】

9568 E_ID 不正ID番号

9569 • mbxidが有効範囲外【NGKI1994】

9570 E_NOEXS オブジェクト未登録

9571 • 対象メールボックスが未登録 [D] 【NGKI1995】

9572

9573 【機能】

9574

9575 mbxidで指定したメールボックス (対象メールボックス) を再初期化する. 具体的振舞いは以下の通り.

9576

9577

9578 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

9579 れているメッセージがない状態に初期化される【NGKI1996】. また, 対象メール

9580 ボックスの待ち行列につなされたタスクは, 待ち行列の先頭のタスクから順

9581 に待ち解除される【NGKI1997】. 待ち解除されたタスクには, 待ち状態となっ

9582 たサービスコールからE_DLTエラーが返る【NGKI1998】.

9583

9584 【使用上の注意】

9585

9586 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

9587 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

9588 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

9589 み禁止時間が長くなるため, 注意が必要である.

9590

9591 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,

9592 アプリケーションの責任である.

9593

9594 【μ ITRON4.0仕様との関係】

9595

9596 μ ITRON4.0仕様に定義されていないサービスコールである.

9597 -----

9598 ref_mbx メールボックスの状態参照 [Tp] 【NGKI1999】

9599

9600 【C言語API】


```

9601         ER ercd = ref_mbx(ID mbxid, T_RMBX *pk_rmbx)
9602
9603     【パラメータ】
9604         ID          mbxid      対象メールボックスのID番号
9605         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
9606                                のポインタ
9607
9608     【リターンパラメータ】
9609         ER          ercd      正常終了 (E_OK) またはエラーコード
9610
9611     * メールボックスの現在状態 (パケットの内容)
9612         ID          wtskid     メールボックスの待ち行列の先頭のタスクのID
9613                                番号
9614         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
9615                                ジの先頭番地
9616
9617     【エラーコード】
9618         E_CTX      コンテキストエラー
9619                     ・ 非タスクコンテキストからの呼出し 【NGKI2000】
9620                     ・ CPUロック状態からの呼出し 【NGKI2001】
9621         E_ID       不正ID番号
9622                     ・ mbxidが有効範囲外 【NGKI2002】
9623         E_NOEXS    オブジェクト未登録
9624                     ・ 対象メールボックスが未登録 [D] 【NGKI2003】
9625
9626     【機能】
9627
9628     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。
9629     参照した現在状態は, pk_rmbxで指定したパケットに返される 【NGKI2004】。
9630
9631     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
9632     TSK_NONE (=0) が返る 【NGKI2005】。 また, メッセージキューにメッセージが
9633     つながれていない場合, pk_msgにはNULLが返る 【NGKI2006】。
9634
9635     【使用上の注意】
9636
9637     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
9638     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
9639     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
9640     クスの状態が変化している可能性があるためである。
9641     -----
9642
9643     4.4.6 ミューテックス
9644
9645     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
9646     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
9647     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
9648     呼ぶID番号によって識別する 【NGKI2007】。
9649
9650     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

```

9651 優先度上限プロトコル (priority ceiling protocol) をサポートする。ミュー
9652 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
9653 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
9654 が行われる。

9655
9656 各ミューテックスが持つ情報は次の通り【NGKI2008】。

- 9657
9658 • ミューテックス属性
9659 • 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)
9660 • ミューテックスをロックしているタスク
9661 • 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)
9662 • 上限優先度 (優先度上限ミューテックスの場合)
9663 • アクセス許可ベクタ (保護機能対応カーネルの場合)
9664 • 属する保護ドメイン (保護機能対応カーネルの場合)
9665 • 属するクラス (マルチプロセッサ対応カーネルの場合)

9666
9667 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
9668 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
9669 ながれているキューである。

9670
9671 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
9672 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
9673 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
9674 【NGKI2009】。

9675
9676 ミューテックス属性には、次の属性を指定することができる【NGKI2010】。

9677
9678 TA_TPRI 0x01U 待ち行列をタスクの優先度順にする
9679 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
9680 クの優先度順にする

9681
9682 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
9683 【NGKI2011】。

9684
9685 ミューテックス機能に関連して、各タスクが持つ情報は次の通り【NGKI2012】。

- 9686
9687 • ロックしているミューテックスのリスト

9688
9689 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
9690 る【NGKI2013】。

9691
9692 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
9693 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
9694 される【NGKI2014】。

9695
9696 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
9697 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
9698 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
9699 なる【NGKI2015】。そのサービスコールにより、当該タスクが実行できる状態
9700 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

9701 【NGKI2016】. そのサービスコールの後で、当該タスクが待ち状態で、タスク
9702 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
9703 優先度に従って、その待ち行列中での順序が変更される【NGKI2017】. 待ち行
9704 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
9705 中で最後になる【NGKI2018】.

9706
9707 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
9708 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
9709 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
9710 ックしたのと逆の順序でロック解除される【NGKI2019】.

9711
9712 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.
9713

9714	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
9715		ネルでは、静的APIによって登録されたミューテックスの
9716		数に一致）【NGKI2020】

9717
9718 【使用上の注意】
9719

9720 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
9721 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
9722 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
9723 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
9724 与しない場合に得られる利点である.

9725
9726 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
9727 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
9728 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、
9729 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
9730 コルでは、タスクがミューテックスのロック待ち状態になることはない）.

9731
9732 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
9733 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
9734 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
9735 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、
9736 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.
9737 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
9738 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
9739 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
9740 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
9741 になる.

9742
9743 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
9744 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
9745 らないploc_mtxを用いるのが安全である.

9746
9747 【補足説明】
9748

9749 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
9750 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

9751 でもpriority ceiling protocolと呼ばれているが、学術論文や他のOSでは、
 9752 immediate ceiling priority protocol, priority protection protocol,
 9753 priority ceiling emulation, highest locker protocolなどと呼ばれている。
 9754

9755 **【TOPPERS/ASPカーネルにおける規定】**
 9756

9757 ASPカーネルでは、ミューテックス機能をサポートしない【ASPS0158】。ただし、
 9758 ミューテックス機能拡張パッケージを用いると、ミューテックス機能を追加す
 9759 ることができる【ASPS0159】。
 9760

9761 **【TOPPERS/FMPカーネルにおける規定】**
 9762

9763 FMPカーネルでは、ミューテックス機能をサポートしない【FMPS0137】。
 9764

9765 **【TOPPERS/HRP2カーネルにおける規定】**
 9766

9767 HRP2カーネルでは、ミューテックス機能をサポートする【HRPS0131】。
 9768

9769 **【未決定事項】**
 9770

9771 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
 9772 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
 9773 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
 9774 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
 9775 課題である。
 9776

9777 **【 μ ITRON4.0仕様との関係】**
 9778

9779 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
 9780 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
 9781 プロトコル (priority inheritance protocol) は、現時点ではサポートしてい
 9782 ない。
 9783

9784 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、
 9785 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。
 9786

9787 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
 9788 る。
 9789 -----

9790 CRE_MTX ミューテックスの生成 [S] 【NGKI2021】
 9791 acre_mtx ミューテックスの生成 [TD] 【NGKI2022】
 9792

9793 **【静的API】**
 9794 CRE_MTX(ID mtxid, { ATR mtxatr, PRI ceilpri })
 9795

9796 **【C言語API】**
 9797 ER_ID mtxid = acre_mtx(const T_CMTX *pk_cmtx)
 9798

9799 **【パラメータ】**
 9800 ID mtxid 生成するミューテックスのID番号 (CRE_MTXの

9801 場合)

9802 T_CMTX * pk_cmtx ミューテックスの生成情報を入れたパケット

9803 へのポインタ (静的APIを除く)

9804

9805 * ミューテックスの生成情報 (パケットの内容)

9806 ATR mtxatr ミューテックス属性

9807 PRI ceilpri ミューテックスの上限優先度

9808

9809 【リターンパラメータ】

9810 ER_ID mtxid 生成されたミューテックスのID番号 (正の値)

9811 またはエラーコード

9812

9813 【エラーコード】

9814 E_CTX コンテキストエラー

9815 ・非タスクコンテキストからの呼出し [s] 【NGKI2023】

9816 ・CPUロック状態からの呼出し [s] 【NGKI2024】

9817 E_RSATR 予約属性

9818 ・mtxatrが無効 【NGKI2025】

9819 ・属する保護ドメインの指定が有効範囲外 [sP] 【NGKI2026】

9820 ・属するクラスの指定が有効範囲外 [sM] 【NGKI2027】

9821 ・クラスの囲みの中に記述されていない [SM] 【NGKI2028】

9822 E_PAR パラメータエラー

9823 ・条件については機能の項を参照

9824 E_OACV オブジェクトアクセス違反

9825 ・システム状態に対する管理操作が許可されていない [sP]

9826 【NGKI2029】

9827 E_MACV メモリアクセス違反

9828 ・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

9829 いない [sP] 【NGKI2030】

9830 E_NOID ID番号不足

9831 ・割り付けられるミューテックスIDがない [sD] 【NGKI2031】

9832 E_OBJ オブジェクト状態エラー

9833 ・mtxidで指定したセマフォが登録済み (CRE_MTXの場合) 【NGKI2032】

9834

9835 【機能】

9836

9837 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

9838 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

9839 に、待ち行列は空の状態に初期化される 【NGKI2033】。

9840

9841 静的APIにおいては、mtxidはオブジェクト識別名、mtxatrとceilpriは整数定数

9842 式パラメータである 【NGKI2034】。優先度上限ミューテックス以外の場合には、

9843 ceilpriの指定を省略することができる 【NGKI2035】。

9844

9845 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、

9846 TMAX_TPRI以下でなければならない。そうでない場合には、E_PARエラーとなる

9847 【NGKI2037】。

9848

9849 【TOPPERS/ASPカーネルにおける規定】

9850

9851 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
9852 トする【ASPS0160】。

9854 【TOPPERS/HRP2カーネルにおける規定】

9856 HRP2カーネルでは、CRE_MTXのみをサポートする【HRPS0132】。ただし、動的生
9857 成機能拡張パッケージでは、acre_mtxもサポートする【HRPS0194】。

9859 AID MTX 割付け可能なミューテックスIDの数の指定 [SD] 【NGKI2038】

9861 【静的API】

```
9862      AID_MTX(uint_t nomtx)
```

9864 【パラメータ】

```
9865      uint t      nomtx      割付け可能なミューテックスIDの数
```

9867 【エラーコード】

9868 E RSATR 予約属性

9869 ・保護ドメインの囲みの中に記述されている [P] 【NGKI3433】

9870 ・クラスの囲みの中に記述されていない [M] 【NGKI2039】

9871 E PAR パラメータエラー

9872 ・ nomtxが負の値【NGKI3282】

9874 【機能】

9876 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
9877 コールによって割付け可能なミューテックスIDとして確保する【NGKI2040】。

9879 nomtxは整数定数式パラメータである【NGKI2041】.

9881 【TOPPERS/HRP2カーネルにおける規定】

9883 HRP2カーネルの動的生成機能拡張パッケージでは、AID_MTXをサポートする
9884 **【HRPS0216】** .

9886 SAC MTX ミューテックスのアクセス許可ベクタの設定 [SP] 【NGKI2042】

9887	sac mtx	ミューテックスのアクセス許可ベクタの設定 [TPD] 【NGKI2043】
------	---------	---------------------------------------

9889 **【静的API】**

```

9890     SAC_MTX(ID mtxid, { ACPTN acptn1, ACPTN acptn2,
9891                          ACPTN acptn3, ACPTN acptn4 })

```

9893 【C言語API】

```
9894     ER ercd = sac_mtx(ID mtxid, const ACVCT *p_acvct)
```

9896 【パラメータ】

9897	ID	mtxid	対象ミューテックスのID番号
------	----	-------	----------------

```
9898      ACVCT *      p_acvct      アクセス許可ベクタを入れたパケットへのポ
```

9899 インタ（静的APIを除く）

9900

9901 *アクセス許可ベクタ (パケットの内容)

9902 ACPTN acptn1 通常操作1のアクセス許可パターン

9903 ACPTN acptn2 通常操作2のアクセス許可パターン

9904 ACPTN acptn3 管理操作のアクセス許可パターン

9905 ACPTN acptn4 参照操作のアクセス許可パターン

9906

9907 【リターンパラメータ】

9908 ER ercd 正常終了 (E_OK) またはエラーコード

9909

9910 【エラーコード】

9911 E_CTX コンテキストエラー

9912 ・非タスクコンテキストからの呼出し [s] 【NGKI2044】

9913 ・CPUロック状態からの呼出し [s] 【NGKI2045】

9914 E_ID 不正ID番号

9915 ・mtxidが有効範囲外 [s] 【NGKI2046】

9916 E_RSATR 予約属性

9917 ・対象ミューテックスが属する保護ドメインの囲みの中 (対

9918 象ミューテックスが無所属の場合は、保護ドメインの囲み

9919 の外) に記述されていない [S] 【NGKI2047】

9920 ・対象ミューテックスが属するクラスの囲みの中に記述され

9921 ていない [SM] 【NGKI2048】

9922 E_NOEXS オブジェクト未登録

9923 ・対象ミューテックスが未登録 【NGKI2049】

9924 E_OACV オブジェクトアクセス違反

9925 ・対象ミューテックスに対する管理操作が許可されていない [s]

9926 【NGKI2050】

9927 E_MACV メモリアクセス違反

9928 ・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

9929 いない [s] 【NGKI2051】

9930 E_OBJ オブジェクト状態エラー

9931 ・対象ミューテックスは静的APIで生成された [s] 【NGKI2052】

9932 ・対象ミューテックスに対してアクセス許可ベクタが設定済

9933 み [S] 【NGKI2053】

9934

9935 【機能】

9936

9937 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

9938 (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

9939 【NGKI2054】 .

9940

9941 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

9942 式パラメータである 【NGKI2055】 .

9943

9944 【TOPPERS/HRP2カーネルにおける規定】

9945

9946 HRP2カーネルでは、SAC_MTXのみをサポートする 【HRPS0133】 . ただし、動的生

9947 成機能拡張パッケージでは、sac_mtxもサポートする 【HRPS0195】 .

9948 -----

9949 del_mtx ミューテックスの削除 [TD] 【NGKI2056】

9950

9951 **【C言語API】**
9952 ER ercd = del_mtx(ID mtxid)
9953
9954 **【パラメータ】**
9955 ID mtxid 対象ミューテックスのID番号
9956
9957 **【リターンパラメータ】**
9958 ER ercd 正常終了 (E_OK) またはエラーコード
9959
9960 **【エラーコード】**
9961 E_CTX コンテキストエラー
9962 ・非タスクコンテキストからの呼出し【NGKI2057】
9963 ・CPUロック状態からの呼出し【NGKI2058】
9964 E_ID 不正ID番号
9965 ・mtxidが有効範囲外【NGKI2059】
9966 E_NOEXS オブジェクト未登録
9967 ・対象ミューテックスが未登録【NGKI2060】
9968 E_OACV オブジェクトアクセス違反
9969 ・対象ミューテックスに対する管理操作が許可されていない [P]
9970 **【NGKI2061】**
9971 E_OBJ オブジェクト状態エラー
9972 ・対象ミューテックスは静的APIで生成された【NGKI2062】
9973
9974 **【機能】**
9975
9976 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
9977 振舞いは以下の通り．
9978
9979 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
9980 に戻される【NGKI2063】．対象ミューテックスをロックしているタスクがある
9981 場合には，そのタスクがロックしているミューテックスのリストから対象ミュー
9982 テックスが削除され，必要な場合にはそのタスクの現在優先度の変更される
9983 **【NGKI2064】**．また，対象ミューテックスの待ち行列につながれたタスクは，
9984 待ち行列の先頭のタスクから順に待ち解除される【NGKI2065】．待ち解除され
9985 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
9986 **【NGKI2066】**．
9987
9988 **【使用上の注意】**
9989
9990 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
9991 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
9992 る．これが不都合な場合には，ミューテックスを削除しようとするタスクが
9993 ミューテックスをロックした状態で，ミューテックスを削除すればよい．
9994
9995 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
9996 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
9997 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
9998 み禁止時間が長くなるため，注意が必要である．
9999
10000 **【TOPPERS/ASPカーネルにおける規定】**

10001
10002 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
10003 ない【ASPS0162】。
10004
10005 【TOPPERS/HRP2カーネルにおける規定】
10006
10007 HRP2カーネルでは、del_mtxをサポートしない【HRPS0134】。ただし、動的生成
10008 機能拡張パッケージでは、del_mtxをサポートする【HRPS0196】。
10009 -----
10010 loc_mtx ミューテックスのロック [T] 【NGKI2067】
10011 ploc_mtx ミューテックスのロック (ポーリング) [T] 【NGKI2068】
10012 tloc_mtx ミューテックスのロック (タイムアウト付き) [T] 【NGKI2069】
10013
10014 【C言語API】
10015 ER ercd = loc_mtx(ID mtxid)
10016 ER ercd = ploc_mtx(ID mtxid)
10017 ER ercd = tloc_mtx(ID mtxid, TMO tmout)
10018
10019 【パラメータ】
10020 ID mtxid 対象ミューテックスのID番号
10021 TMO tmout タイムアウト時間 (tloc_mtxの場合)
10022
10023 【リターンパラメータ】
10024 ER ercd 正常終了 (E_OK) またはエラーコード
10025
10026 【エラーコード】
10027 E_CTX コンテキストエラー
10028 ・非タスクコンテキストからの呼出し【NGKI2070】
10029 ・CPUロック状態からの呼出し【NGKI2071】
10030 ・ディスパッチ保留状態からの呼出し (ploc_mtxを除く)【NGKI2072】
10031 E_NOSPT 未サポート機能
10032 ・制約タスクからの呼出し (ploc_mtxを除く)【NGKI2073】
10033 E_ID 不正ID番号
10034 ・mtxidが有効範囲外【NGKI2074】
10035 E_PAR パラメータエラー
10036 ・tmoutが無効 (tloc_mtxの場合)【NGKI2075】
10037 E_NOEXS オブジェクト未登録
10038 ・対象ミューテックスが未登録 [D]【NGKI2076】
10039 E_OACV オブジェクトアクセス違反
10040 ・対象ミューテックスに対する通常操作1が許可されていない [P]
10041 【NGKI2077】
10042 E_ILUSE サービスコール不正使用
10043 ・条件については機能の項を参照
10044 E_OBJ オブジェクト状態エラー
10045 ・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
10046 【NGKI3609】
10047 E_TMOUT ポーリング失敗またはタイムアウト (loc_mtxを除く)【NGKI2078】
10048 E_RLWAI 待ち禁止状態または待ち状態の強制解除 (ploc_mtxを除く)
10049 【NGKI2079】
10050 E_DLT 待ちオブジェクトの削除または再初期化 (ploc_mtxを除く)

10051 【NGKI2080】

10052

10053 【機能】

10054

10055 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
10056 な振舞いは以下の通り．

10057

10058 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
10059 されている状態になる【NGKI2081】．自タスクがロックしているミューテック
10060 スのリストに対象ミューテックスが追加され，必要な場合には自タスクの現在
10061 優先度に変更される【NGKI2082】．

10062

10063 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
10064 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
10065 の待ち行列につながる【NGKI2083】．

10066

10067 対象ミューテックスが優先度上限ミューテックスで，その上限優先度より自タ
10068 スクのベース優先度が高い場合には，E_ILUSEエラーとなる【NGKI2085】．

10069

10070 【仕様変更の経緯】

10071

10072 この仕様のRelease 1.6以前では，対象ミューテックスが自タスクによってロッ
10073 クされている場合には，E_ILUSEエラーとなることとしていたが，Release 1.7
10074 以降でE_OBJエラーに変更した．これは，ミューテックスを用いて，リエントラ
10075 ントロックを実現できるようにするためである．

10076

10077 -----
10077 unl_mtx ミューテックスのロック解除 [T] 【NGKI2086】

10078

10079 【C言語API】

10080 ER ercd = unl_mtx(ID mtxid)

10081

10082 【パラメータ】

10083 ID mtxid 対象ミューテックスのID番号

10084

10085 【リターンパラメータ】

10086 ER ercd 正常終了 (E_OK) またはエラーコード

10087

10088 【エラーコード】

10089 E_CTX コンテキストエラー

10090 ・非タスクコンテキストからの呼出し【NGKI2087】

10091 ・CPUロック状態からの呼出し【NGKI2088】

10092 E_ID 不正ID番号

10093 ・mtxidが有効範囲外【NGKI2089】

10094 E_NOEXS オブジェクト未登録

10095 ・対象ミューテックスが未登録 [D] 【NGKI2090】

10096 E_OACV オブジェクトアクセス違反

10097 ・対象ミューテックスに対する通常操作1が許可されていない [P]
10098 【NGKI3273】

10099 E_OBJ オブジェクト状態エラー

10100 ・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

10101 い【NGKI3610】

10102

10103 **【機能】**

10104

10105 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
10106 体的な振舞いは以下の通り．

10107

10108 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
10109 クスが削除され，必要な場合には自タスクの現在優先度に変更される

10110 【NGKI2091】．

10111

10112 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
10113 のタスクが待ち解除される【NGKI2092】．対象ミューテックスは，待ち解除さ
10114 れたタスクによってロックされている状態になる【NGKI2093】．待ち解除され
10115 たタスクがロックしているミューテックスのリストに対象ミューテックスが追
10116 加され，必要な場合にはそのタスクの現在優先度に変更される【NGKI2094】．
10117 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
10118 【NGKI2095】．

10119

10120 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
10121 いない状態になる【NGKI2096】．

10122

10123 ini_mtx ミューテックスの再初期化〔T〕【NGKI2098】

10124

10125 **【C言語API】**

10126 ER ercd = ini_mtx(ID mtxid)

10127

10128 **【パラメータ】**

10129 ID mtxid 対象ミューテックスのID番号

10130

10131 **【リターンパラメータ】**

10132 ER ercd 正常終了（E_OK）またはエラーコード

10133

10134 **【エラーコード】**

10135 E_CTX コンテキストエラー

10136 ・非タスクコンテキストからの呼出し【NGKI2099】

10137 ・CPUロック状態からの呼出し【NGKI2100】

10138 E_ID 不正ID番号

10139 ・mtxidが有効範囲外【NGKI2101】

10140 E_NOEXS オブジェクト未登録

10141 ・対象ミューテックスが未登録〔D〕【NGKI2102】

10142 E_OACV オブジェクトアクセス違反

10143 ・対象ミューテックスに対する管理操作が許可されていない〔P〕
10144 【NGKI2103】

10145

10146 **【機能】**

10147

10148 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
10149 体的な振舞いは以下の通り．

10150

10151 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
10152 【NGKI2104】．対象ミューテックスをロックしているタスクがある場合には、
10153 そのタスクがロックしているミューテックスのリストから対象ミューテックス
10154 が削除され、必要な場合にはそのタスクの現在優先度の変更される
10155 【NGKI2105】．また、対象ミューテックスの待ち行列につながれたタスクは、
10156 待ち行列の先頭のタスクから順に待ち解除される【NGKI2106】．待ち解除され
10157 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
10158 【NGKI2107】．

10159

10160 **【使用上の注意】**

10161

10162 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
10163 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
10164 となる．これが不都合な場合には、ミューテックスを再初期化しようとするタ
10165 スクがミューテックスをロックした状態で、ミューテックスを再初期化すれば
10166 よい．

10167

10168 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
10169 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
10170 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
10171 み禁止時間が長くなるため、注意が必要である．

10172

10173 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
10174 は、アプリケーションの責任である．

10175

10176 **【μ ITRON4.0仕様との関係】**

10177

10178 μ ITRON4.0仕様に定義されていないサービスコールである．

10179 -----

10180 ref_mtx ミューテックスの状態参照 [T] 【NGKI2108】

10181

10182 **【C言語API】**

10183 ER ercd = ref_mtx(ID mtxid, T_RMTX *pk_rmtx)

10184

10185 **【パラメータ】**

10186 ID mtxid 対象ミューテックスのID番号
10187 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
10188 のポインタ

10189

10190 **【リターンパラメータ】**

10191 ER ercd 正常終了 (E_OK) またはエラーコード

10192

10193 * ミューテックスの現在状態 (パケットの内容)

10194 ID htskid ミューテックスをロックしているタスクのID番号
10195 ID wtskid ミューテックスの待ち行列の先頭のタスクのID
10196 番号

10197

10198 **【エラーコード】**

10199 E_CTX コンテキストエラー
10200 ・非タスクコンテキストからの呼出し【NGKI2109】

10201 ・CPUロック状態からの呼出し【NGKI2110】
10202 E_ID 不正ID番号
10203 ・mtxidが有効範囲外【NGKI2111】
10204 E_NOEXS オブジェクト未登録
10205 ・対象ミューテックスが未登録〔D〕【NGKI2112】
10206 E_OACV オブジェクトアクセス違反
10207 ・対象ミューテックスに対する参照操作が許可されていない〔P〕
10208 【NGKI2113】
10209 E_MACV メモリアクセス違反
10210 ・pk_rmtxが指すメモリ領域への書き込みアクセスが許可されて
10211 いない〔P〕【NGKI2114】

10213 【機能】

10214
10215 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。
10216 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。

10217
10218 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
10219 返る【NGKI2115】。

10220
10221 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
10222 TSK_NONE（=0）が返る【NGKI2116】。

10224 【使用上の注意】

10225
10226 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
10227 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
10228 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
10229 スの状態が変化している可能性があるためである。

10230 -----

10232 4.4.7 メッセージバッファ

10233
10234 メッセージバッファは、指定した長さのバイト列をメッセージとして、FIFO順
10235 で送受信するための同期・通信オブジェクトである。メッセージバッファは、
10236 メッセージバッファIDと呼ぶID番号によって識別する【NGKI3291】。

10237
10238 各メッセージバッファが持つ情報は次の通り【NGKI3292】。

- 10239
- 10240 ・メッセージバッファ属性
 - 10241 ・最大メッセージサイズ
 - 10242 ・メッセージバッファ管理領域
 - 10243 ・送信待ち行列（メッセージバッファへの送信待ち状態のタスクのキュー）
 - 10244 ・受信待ち行列（メッセージバッファからの受信待ち状態のタスクのキュー）
 - 10245 ・アクセス許可ベクタ（保護機能対応カーネルの場合）
 - 10246 ・属する保護ドメイン（保護機能対応カーネルの場合）
 - 10247 ・属するクラス（マルチプロセッサ対応カーネルの場合）
- 10248

10249 メッセージバッファ管理領域は、メッセージバッファに送信されたメッセージ
10250 を、送信された順に格納しておくためのメモリ領域である。メッセージバッファ

10251 生成時の指定により、メッセージバッファ管理領域のサイズを0とすることができ
10252 ける【NGKI3293】。

10253

10254 保護機能対応カーネルにおいて、メッセージバッファ管理領域は、カーネルの
10255 用いるオブジェクト管理領域として扱われる【NGKI3294】。

10256

10257 送信待ち行列は、メッセージバッファに対してメッセージが送信できるまで待つ
10258 ている状態（メッセージバッファへの送信待ち状態）のタスクが、メッセージ
10259 を送信できる順序でつながれているキューである。また、受信待ち行列は、メッ
10260 セージバッファからメッセージが受信できるまで待っている状態（メッセージ
10261 バッファからの受信待ち状態）のタスクが、メッセージを受信できる順序でつ
10262 ながれているキューである。

10263

10264 メッセージバッファ属性には、次の属性を指定することができる【NGKI3295】。

10265

10266 TA_TPRI 0x01U 送信待ち行列をタスクの優先度順にする

10267

10268 TA_TPRIを指定しない場合、送信待ち行列はFIFO順になる【NGKI3296】。受信待
10269 ち行列は、FIFO順に固定されている【NGKI3297】。

10270

10271 メッセージバッファ機能に関連するカーネル構成マクロは次の通り。

10272

10273 TNUM_MBFID 登録できるメッセージバッファの数（動的生成対応でな
10274 いカーネルでは、静的APIによって登録されたメッセー
10275 ジバッファの数に一致）【NGKI3298】

10276

10277 【TOPPERS/ASPカーネルにおける規定】

10278

10279 ASPカーネルでは、メッセージバッファ機能をサポートしない【ASPS0202】。た
10280 だし、メッセージバッファ機能拡張パッケージを用いると、メッセージバッファ
10281 機能を追加することができる【ASPS0203】。

10282

10283 【TOPPERS/FMPカーネルにおける規定】

10284

10285 FMPカーネルでは、メッセージバッファ機能をサポートしない【FMPS0167】。

10286

10287 【TOPPERS/HRP2カーネルにおける規定】

10288

10289 HRP2カーネルでは、メッセージバッファ機能をサポートしない【HRPS0168】。
10290 ただし、メッセージバッファ機能拡張パッケージを用いると、メッセージバッ
10291 ファ機能を追加することができる【HRPS0169】。

10292

10293 【μITRON4.0仕様との関係】

10294

10295 TNUM_MBFIDは、μITRON4.0仕様に規定されていないカーネル構成マクロである。
10296 -----

10297 CRE_MBF メッセージバッファの生成〔S〕【NGKI3299】

10298 acre_mbf メッセージバッファの生成〔TD〕【NGKI3300】

10299

10300 【静的API】

10301 CRE_MBF(ID mbfid, { ATR mbfatr, uint_t maxmsz, SIZE mbfsz, void *mbfmb })

10302

10303 **【C言語API】**

10304 ER_ID mbfid = acre_mbf(const T_CMBF *pk_cmbf)

10305

10306 **【パラメータ】**

10307 ID mbfid 生成するメッセージバッファのID番号 (CRE_MBF
10308 の場合)

10309 T_CMBF * pk_cmbf メッセージバッファの生成情報を入れたパケッ
10310 トへのポインタ (静的APIを除く)

10311

10312 * メッセージバッファの生成情報 (パケットの内容)

10313 ATR mbfatr メッセージバッファ属性

10314 uint_t maxmsz メッセージバッファの最大メッセージサイズ (バ
10315 イト数)

10316 SIZE mbfsz メッセージバッファ管理領域のサイズ (バイト数)

10317 void * mbfmb メッセージバッファ管理領域の先頭番地

10318

10319 **【リターンパラメータ】**

10320 ER_ID mbfid 生成されたメッセージバッファのID番号 (正の
10321 値) またはエラーコード

10322

10323 **【エラーコード】**

10324 E_CTX コンテキストエラー

10325 ・非タスクコンテキストからの呼出し [s] **【NGKI3301】**

10326 ・CPUロック状態からの呼出し [s] **【NGKI3302】**

10327 E_RSATR 予約属性

10328 ・mbfatrが無効 **【NGKI3303】**

10329 ・属する保護ドメインの指定が有効範囲外 [sP] **【NGKI3304】**

10330 ・属するクラスの指定が有効範囲外 [sM] **【NGKI3305】**

10331 ・クラスの囲みの中に記述されていない [SM] **【NGKI3306】**

10332 E_NOSPT 未サポート機能

10333 ・条件については各カーネルにおける規定の項を参照

10334 E_PAR パラメータエラー

10335 ・maxmszが0以下 **【NGKI3307】**

10336 ・mbfszが負の値 [S] **【NGKI3308】**

10337 ・その他の条件については機能の項を参照

10338 E_OACV オブジェクトアクセス違反

10339 ・システム状態に対する管理操作が許可されていない [sP]

10340 **【NGKI3309】**

10341 E_MACV メモリアクセス違反

10342 ・pk_cmbfが指すメモリ領域への読出しアクセスが許可されて
10343 いない [sP] **【NGKI3310】**

10344 E_NOID ID番号不足

10345 ・割り付けられるメッセージバッファIDがない [sD] **【NGKI3311】**

10346 E_NOMEM メモリ不足

10347 ・メッセージバッファ管理領域が確保できない **【NGKI3312】**

10348 E_OBJ オブジェクト状態エラー

10349 ・mbfidで指定したメッセージバッファが登録済み (CRE_MBF
10350 の場合) **【NGKI3313】**

10351 ・その他の条件については機能の項を参照

10353 【機能】

10355 各パラメータで指定したメッセージバッファ生成情報に従って、メッセージバッ
10356 ファを生成する。mbfszとmbfmbからメッセージバッファ管理領域が設定され、
10357 格納されているメッセージがない状態に初期化される【NGKI3314】。また、送
10358 信待ち行列と受信待ち行列は、空の状態に初期化される【NGKI3315】。

静的APIにおいては、mbfidはオブジェクト識別名、mbfatr、maxmsz、mbfszは整数定数式パラメータ、mbfmbは一般定数式パラメータである【NGKI3316】。コンフィギュレータは、静的APIのメモリ不足 (E_NOMEM) エラーを検出することができない【NGKI3317】。

10365 mbfmbをNULLとした場合、mbfszで指定したサイズのメッセージバッファ管理領
10366 域が、コンフィギュレータまたはカーネルにより確保される【NGKI3318】。

10367 mbfszにターゲット定義の制約に合致しないサイズを指定した時には、ターゲッ
10368 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
10369 **【NGKI3319】** .

10371 [mbfmbにNULL以外を指定した場合]

mbfmbにNULL以外を指定した場合、mbfmbとmbfszで指定したメッセージバッファ管理領域は、アプリケーションで確保しておく必要がある【NGKI3320】。メッセージバッファ管理領域をアプリケーションで確保するために、次のマクロを用意している【NGKI3321】。

10378	TSZ_MBFMB(msgcnt, msgsz)	msgszで指定したサイズのメッセージを、
10379		msgcntで指定した数だけ格納できるメッセー
10380		ジバッファ管理領域のサイズ (バイト数)
10381	TCNT_MBFMB(msgcnt, msgsz)	msgszで指定したサイズのメッセージを、
10382		msgcntで指定した数だけ格納できるメッセー
10383		ジバッファ管理領域を確保するために必要
10384		なMB_T型の配列の要素数

10386 これらを用いてメッセージバッファ管理領域を確保する方法は次の通り
10387 **【NGKI3322】** .

```
10389 MB_T    <メッセージバッファ管理領域の変数名>[TCNT_MBFMB(msgcnt, msgsz)];
```

10391 この時、mbfszにはTSZ_MBFMB(msgcnt, msgsz)を、mbfmbには<メッセージバッファ
10392 管理領域の変数名>を指定する【NGKI3323】。

10394 この方法に従わず、mbfmbとmbfszにターゲット定義の制約に合致しない先頭番
10395 地やサイズを指定した時には、E_PARエラーとなる【NGKI3324】。また、保護機
10396 能対応カーネルにおいて、mbfmbとmbfszで指定したメッセージバッファ管理領
10397 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
10398 【NGKI3325】。

10400 なお、TSZ MBFMBは、mbfmbにNULLを指定した場合にも、メッセージバッファ管

10401 理領域のサイズを決めるために用いることができる。
10402
10403 **【TOPPERS/ASPカーネルにおける規定】**
10404
10405 ASPカーネルのメッセージバッファ機能拡張パッケージでは、CRE_MBFのみをサ
10406 ポートする【ASPS0204】。また、mbfmbにはNULLのみを指定することができる。
10407 NULL以外を指定した場合には、E_NOSPTエラーとなる【ASPS0205】。
10408
10409 **【TOPPERS/HRP2カーネルにおける規定】**
10410
10411 HRP2カーネルのメッセージバッファ機能拡張パッケージでは、CRE_MBFのみをサ
10412 ポートする【HRPS0170】。また、mbfmbにはNULLのみを指定することができる。
10413 NULL以外を指定した場合には、E_NOSPTエラーとなる【HRPS0171】。
10414
10415 **【 μ ITRON4.0仕様との関係】**
10416
10417 μ ITRON4.0/PX仕様にあわせて、メッセージバッファ生成情報の最後のパラメー
10418 タを、mbf（メッセージバッファ領域の先頭番地）から、mbfmb（メッセージバッ
10419 ファ管理領域の先頭番地）に改名した。また、TSZ_MBFをTSZ_MBFMBに改名した。
10420
10421 TCNT_MBFMBを新設し、メッセージバッファ管理領域をアプリケーションで確保
10422 する方法を規定した。
10423 -----
10424 AID_MBF 割付け可能なメッセージバッファIDの数の指定〔SD〕【NGKI3326】
10425
10426 **【静的API】**
10427 AID_MBF(uint_t nombf)
10428
10429 **【パラメータ】**
10430 uint_t nombf 割付け可能なメッセージバッファIDの数
10431
10432 **【エラーコード】**
10433 E_RSATR 予約属性
10434 ・保護ドメインの囲みの中に記述されている〔P〕【NGKI3434】
10435 ・クラスの囲みの中に記述されていない〔M〕【NGKI3327】
10436 E_PAR パラメータエラー
10437 ・nombfが負の値【NGKI3328】
10438
10439 **【機能】**
10440
10441 nombfで指定した数のメッセージバッファIDを、メッセージバッファを生成する
10442 サービスコールによって割付け可能なメッセージバッファIDとして確保する
10443 【NGKI3329】。
10444
10445 nombfは整数定数式パラメータである【NGKI3330】。
10446 -----
10447 SAC_MBF メッセージバッファのアクセス許可ベクタの設定〔SP〕【NGKI3331】
10448 sac_mbf メッセージバッファのアクセス許可ベクタの設定〔TPD〕【NGKI3332】
10449
10450 **【静的API】**

10451 SAC_MBF(ID mbfid, { ACPTN acptn1, ACPTN acptn2,
10452 ACPTN acptn3, ACPTN acptn4 })
10453

10454 **【C言語API】**

10455 ER ercd = sac_mbf(ID mbfid, const ACVCT *p_acvct)
10456

10457 **【パラメータ】**

10458 ID mbfid 対象メッセージバッファのID番号
10459 ACVCT * p_acvct アクセス許可ベクタを入れたパケットへのポ
10460 インタ（静的APIを除く）
10461

10462 *アクセス許可ベクタ（パケットの内容）

10463 ACPTN acptn1 通常操作1のアクセス許可パターン
10464 ACPTN acptn2 通常操作2のアクセス許可パターン
10465 ACPTN acptn3 管理操作のアクセス許可パターン
10466 ACPTN acptn4 参照操作のアクセス許可パターン
10467

10468 **【リターンパラメータ】**

10469 ER ercd 正常終了 (E_OK) またはエラーコード
10470

10471 **【エラーコード】**

10472 E_CTX コンテキストエラー
10473 ・非タスクコンテキストからの呼出し [s] 【NGKI3333】
10474 ・CPUロック状態からの呼出し [s] 【NGKI3334】
10475 E_ID 不正ID番号
10476 ・mbfidが有効範囲外 [s] 【NGKI3335】
10477 E_RSATR 予約属性
10478 ・対象メッセージバッファが属する保護ドメインの囲みの中
10479 （対象メッセージバッファが無所属の場合は、保護ドメイ
10480 ンの囲みの外）に記述されていない [S] 【NGKI3336】
10481 ・対象メッセージバッファが属するクラスの囲みの中に記述
10482 されていない [SM] 【NGKI3337】
10483 E_NOEXS オブジェクト未登録
10484 ・対象メッセージバッファが未登録 【NGKI3338】
10485 E_OACV オブジェクトアクセス違反
10486 ・対象メッセージバッファに対する管理操作が許可されてい
10487 ない [s] 【NGKI3339】
10488 E_MACV メモリアクセス違反
10489 ・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
10490 いない [s] 【NGKI3340】
10491 E_OBJ オブジェクト状態エラー
10492 ・対象メッセージバッファは静的APIで生成された [s] 【NGKI3341】
10493 ・対象メッセージバッファに対してアクセス許可ベクタが設
10494 定済み [S] 【NGKI3342】
10495

10496 **【機能】**
10497

10498 mbfidで指定したメッセージバッファ（対象メッセージバッファ）のアクセス許
10499 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
10500 設定する 【NGKI3343】。

10501
10502 静的APIにおいては、mbfidはオブジェクト識別名、acptn1～acptn4は整数定数
10503 式パラメータである【NGKI3344】。
10504
10505 【TOPPERS/HRP2カーネルにおける規定】
10506
10507 HRP2カーネルのメッセージバッファ機能拡張パッケージでは、SAC_MBFのみをサ
10508 ポートする【HRPS0172】。
10509 -----
10510 del_mbf メッセージバッファの削除 [TD] 【NGKI3345】
10511
10512 【C言語API】
10513 ER ercd = del_mbf(ID mbfid)
10514
10515 【パラメータ】
10516 ID mbfid 対象メッセージバッファのID番号
10517
10518 【リターンパラメータ】
10519 ER ercd 正常終了 (E_OK) またはエラーコード
10520
10521 【エラーコード】
10522 E_CTX コンテキストエラー
10523 ・非タスクコンテキストからの呼出し【NGKI3346】
10524 ・CPUロック状態からの呼出し【NGKI3347】
10525 E_ID 不正ID番号
10526 ・mbfidが有効範囲外【NGKI3348】
10527 E_NOEXS オブジェクト未登録
10528 ・対象メッセージバッファが未登録【NGKI3349】
10529 E_OACV オブジェクトアクセス違反
10530 ・対象メッセージバッファに対する管理操作が許可されてい
10531 ない [P] 【NGKI3350】
10532 E_OBJ オブジェクト状態エラー
10533 ・対象メッセージバッファは静的APIで生成された【NGKI3351】
10534
10535 【機能】
10536
10537 mbfidで指定したメッセージバッファ（対象メッセージバッファ）を削除する。
10538 具体的な振舞いは以下の通り。
10539
10540 対象メッセージバッファの登録が解除され、そのメッセージバッファIDが未使
10541 用の状態に戻される【NGKI3352】。また、対象メッセージバッファの送信待ち
10542 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
10543 クから順に待ち解除される【NGKI3353】。待ち解除されたタスクには、待ち状
10544 態となったサービスコールからE_DLTエラーが返る【NGKI3354】。
10545
10546 メッセージバッファの生成時に、メッセージバッファ管理領域がカーネルによっ
10547 て確保された場合は、そのメモリ領域が解放される【NGKI3355】。
10548
10549 【補足説明】
10550

10551 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、
10552 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。
10553
10554
10555 **【使用上の注意】**
10556
10557 del_mbfにより複数のタスクが待ち解除される場合、サービスコールの処理時間
10558 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
10559 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
10560 み禁止時間が長くなるため、注意が必要である。
10561
10562 **【TOPPERS/ASPカーネルにおける規定】**
10563
10564 ASPカーネルのメッセージバッファ機能拡張パッケージでは、del_mbfをサポート
10565 しない【ASPS0207】。
10566
10567 **【TOPPERS/HRP2カーネルにおける規定】**
10568
10569 HRP2カーネルのメッセージバッファ機能拡張パッケージでは、del_mbfをサポート
10570 しない【HRPS0173】。
10571 -----
10572 snd_mbf メッセージバッファへの送信 [T] **【NGKI3356】**
10573 psnd_mbf メッセージバッファへの送信（ポーリング） [T] **【NGKI3357】**
10574 tsnd_mbf メッセージバッファへの送信（タイムアウト付き） [T] **【NGKI3358】**
10575
10576 **【C言語API】**
10577 ER ercd = snd_mbf(ID mbfid, const void *msg, uint_t msgsz)
10578 ER ercd = psnd_mbf(ID mbfid, const void *msg, uint_t msgsz)
10579 ER ercd = tsnd_mbf(ID mbfid, const void *msg, uint_t msgsz, TMO tmout)
10580
10581 **【パラメータ】**
10582 ID mbfid 対象メッセージバッファのID番号
10583 void * msg 送信メッセージの先頭番地
10584 uint_t msgsz 送信メッセージのサイズ（バイト数）
10585 TMO tmout タイムアウト時間（tsnd_mbfの場合）
10586
10587 **【リターンパラメータ】**
10588 ER ercd 正常終了（E_OK）またはエラーコード
10589
10590 **【エラーコード】**
10591 E_CTX コンテキストエラー
10592 ・非タスクコンテキストからの呼出し **【NGKI3359】**
10593 ・CPUロック状態からの呼出し **【NGKI3360】**
10594 ・ディスパッチ保留状態からの呼出し（psnd_mbfを除く）
10595 **【NGKI3361】**
10596 E_NOSPT 未サポート機能
10597 ・制約タスクからの呼出し（psnd_mbfを除く） **【NGKI3362】**
10598 E_ID 不正ID番号
10599 ・mbfidが有効範囲外 **【NGKI3363】**
10600 E_PAR パラメータエラー

10601 ・msgszが有効範囲（0より大きく対象メッセージバッファの
10602 最大メッセージサイズ以下）外【NGKI3364】
10603 ・tmoutが無効（tsnd_mbfの場合）【NGKI3365】
10604 E_NOEXS オブジェクト未登録
10605 ・対象メッセージバッファが未登録 [D] 【NGKI3366】
10606 E_OACV オブジェクトアクセス違反
10607 ・対象メッセージバッファに対する通常操作1が許可されて
10608 いない [P] 【NGKI3367】
10609 E_MACV メモリアクセス違反
10610 ・msgとmsgszが指すメモリ領域への読出しアクセスが許可さ
10611 れていない [P] 【NGKI3368】
10612 E_TMOUT ポーリング失敗またはタイムアウト（snd_mbfを除く）【NGKI3369】
10613 E_RLWAI 待ち禁止状態または待ち状態の強制解除（psnd_mbfを除く）
10614 【NGKI3370】
10615 E_DLT 待ちオブジェクトの削除または再初期化（psnd_mbfを除く）
10616 【NGKI3371】
10617

10618 【機能】

10619
10620 mbfidで指定したメッセージバッファ（対象メッセージバッファ）に、msgと
10621 msgszで指定したメッセージ（送信メッセージ）を送信する．具体的な振舞いは
10622 以下の通り．
10623

10624 対象メッセージバッファの受信待ち行列にタスクが存在する場合には、受信待
10625 ち行列の先頭のタスクが、送信メッセージを受信し、待ち解除される
10626 【NGKI3372】．待ち解除されたタスクには、待ち状態となったサービスコール
10627 から、受信したメッセージのサイズが返る【NGKI3373】．
10628

10629 対象メッセージバッファの受信待ち行列にタスクが存在しない場合で、送信待
10630 ち行列に自タスクより優先してメッセージを送信できるタスクが存在せず、メッ
10631 セージバッファ管理領域に送信メッセージを格納するスペースがある場合には、
10632 送信メッセージが、FIFO順でメッセージバッファ管理領域に格納される
10633 【NGKI3374】．ここで、送信待ち行列に自タスクより優先してメッセージを送
10634 信できるタスクが存在するとは、送信待ち行列がFIFO順の場合には送信待ち行
10635 列に何らかのタスクが存在すること、タスクの優先度順の場合には自タスクと
10636 優先度が同じかより高いタスクが存在することを意味する．
10637

10638 対象メッセージバッファの受信待ち行列にタスクが存在しない場合で、送信待
10639 ち行列に自タスクより優先してメッセージを送信できるタスクが存在するか、
10640 メッセージバッファ管理領域に送信メッセージを格納するスペースがない場合
10641 には、自タスクはメッセージバッファへの送信待ち状態となり、対象メッセー
10642 ジバッファの送信待ち行列につながる【NGKI3375】．
10643

10644 メッセージバッファの送信待ち行列の先頭につながれているタスクが、
10645 ter_tskにより強制終了した場合や、rel_wai／irel_waiやタイムアウトにより
10646 待ち解除された場合、新たに送信待ち行列の先頭になったタスクの送信メッセー
10647 ジを、メッセージバッファ管理領域に格納できる可能性がある．そのため、こ
10648 れらの場合には、メッセージバッファからの受信によりメッセージバッファ管
10649 理領域に空きができた時の処理【NGKI3393】【NGKI3394】【NGKI3395】と同じ
10650 処理が行われる【NGKI3419】．さらに、送信待ち行列がタスクの優先度順の時

10651 には、chg_priやミューテックスの操作によりタスクの優先度が変化し、送信待
10652 ち行列の先頭につながれているタスクが変わった場合にも、同じ処理が行われ
10653 る【NGKI3420】。

10654

10655 【使用上の注意】

10656

10657 送信待ち行列の先頭につながれているタスクの強制終了、待ち解除、優先度変
10658 更に伴う処理で、送信待ち行列につながれていたタスクが複数待ち解除される
10659 場合がある。この時、サービスコールの処理時間およびカーネル内での割込み
10660 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
10661 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、
10662 注意が必要である。

10663

10664 -----
10664 rcv_mbf メッセージバッファからの受信 [T] 【NGKI3376】

10665 prcv_mbf メッセージバッファからの受信（ポーリング） [T] 【NGKI3377】

10666 trcv_mbf メッセージバッファからの受信（タイムアウト付き） [T] 【NGKI3378】

10667

10668 【C言語API】

10669 ER_UINT msgsz = rcv_mbf(ID mbfid, void *msg)

10670 ER_UINT msgsz = prcv_mbf(ID mbfid, void *msg)

10671 ER_UINT msgsz = trcv_mbf(ID mbfid, void *msg, TMO tmout)

10672

10673 【パラメータ】

10674 ID mbfid 対象メッセージバッファのID番号

10675 void * msg 受信メッセージを入れるメモリ領域の先頭番地

10676 TMO tmout タイムアウト時間（trcv_mbfの場合）

10677

10678 【リターンパラメータ】

10679 ER_UINT msgsz 受信メッセージサイズ（正の値）またはエラー
10680 コード

10681

10682 【エラーコード】

10683 E_CTX コンテキストエラー

10684 ・非タスクコンテキストからの呼出し【NGKI3379】

10685 ・CPUロック状態からの呼出し【NGKI3380】

10686 ・ディスパッチ保留状態からの呼出し（prcv_mbfを除く）

10687 【NGKI3381】

10688 E_NOSPT 未サポート機能

10689 ・制約タスクからの呼出し（prcv_mbfを除く）【NGKI3382】

10690 E_ID 不正ID番号

10691 ・mbfidが有効範囲外【NGKI3383】

10692 E_PAR パラメータエラー

10693 ・tmoutが無効（trcv_mbfの場合）【NGKI3384】

10694 E_NOEXS オブジェクト未登録

10695 ・対象メッセージバッファが未登録 [D] 【NGKI3385】

10696 E_OACV オブジェクトアクセス違反

10697 ・対象メッセージバッファに対する通常操作2が許可されてい
10698 ない [P] 【NGKI3386】

10699 E_MACV メモリアクセス違反

10700 ・msgを先頭番地とし、対象メッセージバッファの最大メッセー

10701 ジサイズ分のメモリ領域への書込みアクセスが許可されて
10702 いない [P] 【NGKI3387】
10703 E_TMOUT ポーリング失敗またはタイムアウト (rcv_mbfを除く) 【NGKI3388】
10704 E_RLWAI 待ち禁止状態または待ち状態の強制解除 (prcv_mbfを除く)
10705 【NGKI3389】
10706 E_DLT 待ちオブジェクトの削除または再初期化 (prcv_mbfを除く)
10707 【NGKI3390】
10708

10709 **【機能】**

10710

10711 mbfidで指定したメッセージバッファ (対象メッセージバッファ) からメッセー
10712 ジを受信する. メッセージの受信に成功した場合, 受信したメッセージはmsgを
10713 先頭番地とするメモリ領域に格納され, そのサイズはサービスコールの返値と
10714 して返される 【NGKI3391】. 具体的な振舞いは以下の通り.
10715

10716 対象メッセージバッファのメッセージバッファ管理領域にメッセージが格納さ
10717 れている場合には, メッセージバッファ管理領域の先頭に格納されたメッセー
10718 ジを受信する 【NGKI3392】. また, 送信待ち行列にタスクが存在し, メッセー
10719 ジバッファ管理領域に送信待ち行列の先頭のタスクの送信メッセージを格納す
10720 るスペースがある場合には, 送信メッセージがFIFO順でデータキュー管理領域
10721 に格納され, そのタスクは待ち解除される 【NGKI3393】. 待ち解除されたタス
10722 クには, 待ち状態となったサービスコールからE_OKが返る 【NGKI3394】. この
10723 処理を, 送信待ち行列にタスクが存在しなくなるか, メッセージバッファ管理
10724 領域に送信待ち行列の先頭のタスクの送信メッセージを格納するスペースがな
10725 くなるまで繰り返す 【NGKI3395】.
10726

10727 対象メッセージバッファのメッセージバッファ管理領域にメッセージが格納さ
10728 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
10729 のタスクの送信メッセージを受信する 【NGKI3396】. 送信待ち行列の先頭のタ
10730 スクは, 待ち解除される 【NGKI3397】. 待ち解除されたタスクには, 待ち状態
10731 となったサービスコールからE_OKが返る 【NGKI3398】.
10732

10733 対象メッセージバッファのメッセージバッファ管理領域にメッセージが格納さ
10734 れておらず, 送信待ち行列にタスクが存在しない場合には, 自タスクはメッセー
10735 ジバッファからの受信待ち状態となり, 対象メッセージバッファの受信待ち行
10736 列につながる 【NGKI3399】.
10737

10738 **【使用上の注意】**

10739

10740 メッセージバッファ管理領域に格納されたメッセージを受信した結果, 送信待
10741 ち行列につながれていたタスクが複数待ち解除される場合がある. この時, サー
10742 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
10743 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,
10744 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.
10745

10746 -----
10746 ini_mbf メッセージバッファの再初期化 [T] 【NGKI3400】
10747

10748 **【C言語API】**

10749 ER ercd = ini_mbf(ID mbfid)
10750

10751 **【パラメータ】**

10752 ID mbfid 対象メッセージバッファのID番号

10753

10754 **【リターンパラメータ】**

10755 ER ercd 正常終了 (E_OK) またはエラーコード

10756

10757 **【エラーコード】**

10758 E_CTX コンテキストエラー

10759 ・非タスクコンテキストからの呼出し【NGKI3401】

10760 ・CPUロック状態からの呼出し【NGKI3402】

10761 E_ID 不正ID番号

10762 ・mbfidが有効範囲外【NGKI3403】

10763 E_NOEXS オブジェクト未登録

10764 ・対象メッセージバッファが未登録 [D] 【NGKI3404】

10765 E_OACV オブジェクトアクセス違反

10766 ・対象メッセージバッファに対する管理操作が許可されてい

10767 ない [P] 【NGKI3405】

10768

10769 **【機能】**

10770

10771 mbfidで指定したメッセージバッファ (対象メッセージバッファ) を再初期化す

10772 る. 具体的な振舞いは以下の通り.

10773

10774 対象メッセージバッファのメッセージバッファ管理領域は, 格納されているメッ

10775 セージがない状態に初期化される【NGKI3406】. また, 対象メッセージバッファ

10776 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

10777 先頭のタスクから順に待ち解除される【NGKI3407】. 待ち解除されたタスクに

10778 は, 待ち状態となったサービスコールからE_DLTエラーが返る【NGKI3408】.

10779

10780 **【補足説明】**

10781

10782 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,

10783 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

10784 い.

10785

10786 **【使用上の注意】**

10787

10788 ini_mbfにより複数のタスクが待ち解除される場合, サービスコールの処理時間

10789 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

10790 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

10791 み禁止時間が長くなるため, 注意が必要である.

10792

10793 メッセージバッファを再初期化した場合に, アプリケーションとの整合性を保

10794 つのは, アプリケーションの責任である.

10795

10796 **【μ ITRON4.0仕様との関係】**

10797

10798 μ ITRON4.0仕様に定義されていないサービスコールである.

10799 -----

10800 ref_mbf メッセージバッファの状態参照 [T] 【NGKI3409】


```

10801
10802 【C言語API】
10803     ER ercd = ref_mbf(ID mbfid, T_RMBF *pk_rmbf)
10804
10805 【パラメータ】
10806     ID          mbfid      対象メッセージバッファのID番号
10807     T_RMBF *    pk_rmbf    メッセージバッファの現在状態を入れるパケッ
10808                          トへのポインタ
10809
10810 【リターンパラメータ】
10811     ER          ercd      正常終了 (E_OK) またはエラーコード
10812
10813     *メッセージバッファの現在状態 (パケットの内容)
10814     ID          stskid     メッセージバッファの送信待ち行列の先頭のタ
10815                          スクのID番号
10816     ID          rtskid     メッセージバッファの受信待ち行列の先頭のタ
10817                          スクのID番号
10818     uint_t      smbfcnt    メッセージバッファ管理領域に格納されている
10819                          メッセージの数
10820     SIZE        fmbfsz     メッセージバッファ管理領域中の空き領域のサ
10821                          イズ
10822
10823 【エラーコード】
10824     E_CTX       コンテキストエラー
10825                ・非タスクコンテキストからの呼出し【NGKI3410】
10826                ・CPUロック状態からの呼出し【NGKI3411】
10827     E_ID        不正ID番号
10828                ・mbfidが有効範囲外【NGKI3412】
10829     E_NOEXS     オブジェクト未登録
10830                ・対象メッセージバッファが未登録 [D] 【NGKI3413】
10831     E_OACV      オブジェクトアクセス違反
10832                ・対象メッセージバッファに対する参照操作が許可されてい
10833                ない [P] 【NGKI3414】
10834     E_MACV      メモリアクセス違反
10835                ・pk_rmbfが指すメモリ領域への書込みアクセスが許可されて
10836                いない [P] 【NGKI3415】
10837
10838 【機能】
10839
10840     mbfidで指定したメッセージバッファ (対象メッセージバッファ) の現在状態を
10841     参照する. 参照した現在状態は, pk_rmbfで指定したパケットに返される
10842     【NGKI3416】.
10843
10844     対象メッセージバッファの送信待ち行列にタスクが存在しない場合, stskidには
10845     TSK_NONE (=0) が返る【NGKI3417】. また, 受信待ち行列にタスクが存在しな
10846     い場合, rtskidにはTSK_NONE (=0) が返る【NGKI3418】.
10847
10848 【使用上の注意】
10849
10850     ref_mbfはデバッグ時向けの機能であり, その他の目的に使用することは推奨し

```

10851 ない。これは、ref_mbfを呼び出し、対象メッセージバッファの現在状態を参照
10852 した直後に割込みが発生した場合、ref_mbfから戻ってきた時には対象メッセ
10853 ジバッファの状態が変化している可能性があるためである。
10854 -----

10855 10856 4.4.8 スピンロック 10857

10858 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
10859 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
10860 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
10861 【NGKI2117】。

10862
10863 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
10864 カーネル管理の割込みがマスクされ、ディスパッチが保留される【NGKI2118】。
10865 ロックが他のプロセッサに取得されている場合には、ロックが取得できるまで
10866 ループによって待つ【NGKI2119】。ロックの取得を待つ間は、CPUロック解除状
10867 態であり、割込みはマスクされない【NGKI2120】。プロセッサ間ロックを取得
10868 し、CPUロック状態に遷移することを、スピンロックを取得するという。また、
10869 プロセッサ間ロックを返却し、CPUロック状態を解除することを、スピンロック
10870 を返却するという。

10871
10872 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
10873 ルーチン、割込みハンドラ、割込みサービ斯拉ーチン、タイムイベントハンド
10874 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
10875 てスピンロックが返却される【NGKI2121】。また、スピンロックを取得してい
10876 ない状態で発生したCPU例外によって呼び出されたCPU例外ハンドラが、取得し
10877 たスピンロックを返却せずにリターンした場合には、カーネルによってスピン
10878 ロックが返却される【NGKI2122】。一方、拡張サービスコールからのリターン
10879 では、スピンロックは返却されない【NGKI2123】。

10880
10881 各スピンロックが持つ情報は次の通り【NGKI2124】。
10882

- 10883 ・スピンロック属性
- 10884 ・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）
- 10885 ・アクセス許可ベクタ（保護機能対応カーネルの場合）
- 10886 ・属する保護ドメイン（保護機能対応カーネルの場合）
- 10887 ・属するクラス

10888
10889 スピンロック属性に指定できる属性はない【NGKI2125】。そのためスピンロッ
10890 ク属性には、TA_NULLを指定しなければならない【NGKI2126】。
10891

10892 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。
10893

10894 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 10895 ネルでは、静的APIによって登録されたスピンロックの数 10896 に一致）【NGKI2127】
------------------	--

10897 10898 【補足説明】 10899

10900 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

10901 できないため、スピンロックを取得しているプロセッサが、さらにスピンロック
 10902 を取得することはできない。そのため、1つの処理単位が、複数のスピンロック
 10903 を取得した状態になることはできない。
 10904
 10905 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
 10906 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
 10907 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
 10908 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
 10909 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
 10910 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。
 10911
 10912 **【TOPPERS/ASPカーネルにおける規定】**
 10913
 10914 ASPカーネルでは、スピンロック機能をサポートしない【ASPS0163】。
 10915
 10916 **【TOPPERS/FMPカーネルにおける規定】**
 10917
 10918 FMPカーネルでは、スピンロック機能をサポートする【FMPS0138】。
 10919
 10920 **【TOPPERS/HRP2カーネルにおける規定】**
 10921
 10922 HRP2カーネルでは、スピンロック機能をサポートしない【HRPS0135】。
 10923
 10924 **【μITRON4.0仕様との関係】**
 10925
 10926 スピンロック機能は、μITRON4.0仕様に定義されていない機能である。
 10927 -----
 10928 CRE_SPN スピンロックの生成〔SM〕 【NGKI2128】
 10929 acre_spn スピンロックの生成〔TMD〕 【NGKI2129】
 10930
 10931 **【静的API】**
 10932 CRE_SPN(ID spnid, { ATR spnatr })
 10933
 10934 **【C言語API】**
 10935 ER_ID spnid = acre_spn(const T_CSPN *pk_cspn)
 10936
 10937 **【パラメータ】**
 10938 ID spnid 生成するスピンロックのID番号（CRE_SPNの場合）
 10939 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
 10940 ポインタ（静的APIを除く）
 10941
 10942 * スピンロックの生成情報（パケットの内容）
 10943 ATR spnatr スピンロック属性
 10944
 10945 **【リターンパラメータ】**
 10946 ER_ID spnid 生成されたスピンロックのID番号（正の値）ま
 10947 たはエラーコード
 10948
 10949 **【エラーコード】**
 10950 E_CTX コンテキストエラー

10951 ・非タスクコンテキストからの呼出し [s] 【NGKI2130】
 10952 ・CPUロック状態からの呼出し [s] 【NGKI2131】
 10953 E_RSATR 予約属性
 10954 ・spnattrが無効 【NGKI2132】
 10955 ・属する保護ドメインの指定が有効範囲外 [sP] 【NGKI2133】
 10956 ・属するクラスの指定が有効範囲外 [s] 【NGKI2134】
 10957 ・クラスの囲みの中に記述されていない [S] 【NGKI2135】
 10958 E_OACV オブジェクトアクセス違反
 10959 ・システム状態に対する管理操作が許可されていない [sP]
 10960 【NGKI2136】
 10961 E_MACV メモリアクセス違反
 10962 ・pk_cspnが指すメモリ領域への読出しアクセスが許可されて
 10963 いない [sP] 【NGKI2137】
 10964 E_NOID ID番号不足
 10965 ・割り付けられるスピンロックIDがない [sD] 【NGKI2138】
 10966 E_NORES 資源不足
 10967 ・条件については機能の項を参照
 10968 E_OBJ オブジェクト状態エラー
 10969 ・spnidで指定したスピンロックが登録済み (CRE_SPNの場合)
 10970 【NGKI2139】

10971 【機能】

10972
 10973
 10974 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
 10975 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
 10976 化される 【NGKI2140】。

10977
 10978 静的APIにおいては、spnidはオブジェクト識別名、spnattrは整数定数式パラメー
 10979 タである 【NGKI2141】。

10980
 10981 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
 10982 で、生成できるスピンロックの数に上限がある 【NGKI2142】。この上限を超え
 10983 てスピンロックを生成しようとした場合には、E_NORESエラーとなる
 10984 【NGKI2143】。

10985 【補足説明】

10986
 10987
 10988 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
 10989 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
 10990 ない。

10991 【TOPPERS/FMPカーネルにおける規定】

10992
 10993 FMPカーネルでは、CRE_SPNのみをサポートする 【FMPS0139】。

10994 -----
 10995 AID_SPN 割付け可能なスピンロックIDの数の指定 [SMD] 【NGKI2144】

10996 【静的API】

10997 AID_SPN(uint_t nospn)

10998
 10999
 11000

```

11001  【パラメータ】
11002      uint_t      nospn      割付け可能なスピンロックIDの数
11003
11004  【エラーコード】
11005      E_RSATR      予約属性
11006                  ・保護ドメインの囲みの中に記述されている [P]  【NGKI3435】
11007                  ・クラスの囲みの中に記述されていない 【NGKI2145】
11008      E_PAR        パラメータエラー
11009                  ・nospnが負の値 【NGKI3283】
11010
11011  【機能】
11012
11013  nospnで指定した数のスピンロックIDを、スピンロックを生成するサービスコー
11014  ルによって割付け可能なスピンロックIDとして確保する 【NGKI2146】 .
11015
11016  nospnは整数定数式パラメータである 【NGKI2147】 .
11017  -----
11018  SAC_SPN          スピンロックのアクセス許可ベクタの設定 [SPM]  【NGKI2148】
11019  sac_spn          スピンロックのアクセス許可ベクタの設定 [TPMD] 【NGKI2149】
11020
11021  【静的API】
11022      SAC_SPN(ID spnid, { ACPTN acptn1, ACPTN acptn2,
11023                          ACPTN acptn3, ACPTN acptn4 })
11024
11025  【C言語API】
11026      ER ercd = sac_spn(ID spnid, const ACVCT *p_acvct)
11027
11028  【パラメータ】
11029      ID          spnid      対象スピンロックのID番号
11030      ACVCT *     p_acvct    アクセス許可ベクタを入れたパケットへのポ
11031                          インタ（静的APIを除く）
11032
11033      *アクセス許可ベクタ（パケットの内容）
11034      ACPTN      acptn1      通常操作1のアクセス許可パターン
11035      ACPTN      acptn2      通常操作2のアクセス許可パターン
11036      ACPTN      acptn3      管理操作のアクセス許可パターン
11037      ACPTN      acptn4      参照操作のアクセス許可パターン
11038
11039  【リターンパラメータ】
11040      ER          ercd      正常終了（E_OK）またはエラーコード
11041
11042  【エラーコード】
11043      E_CTX      コンテキストエラー
11044                  ・非タスクコンテキストからの呼出し [s]  【NGKI2150】
11045                  ・CPUロック状態からの呼出し [s]  【NGKI2151】
11046      E_ID      不正ID番号
11047                  ・spnidが有効範囲外 [s]  【NGKI2152】
11048      E_RSATR      予約属性
11049                  ・対象スピンロックが属する保護ドメインの囲みの中（対象
11050                  スピンロックが無所属の場合は、保護ドメインの囲みの外）

```

11051 に記述されていない [S] 【NGKI2153】

11052 ・対象スピンロックが属するクラスの囲みの中に記述されて

11053 いない [S] 【NGKI2154】

11054 E_NOEXS オブジェクト未登録

11055 ・対象スピンロックが未登録 【NGKI2155】

11056 E_OACV オブジェクトアクセス違反

11057 ・対象スピンロックに対する管理操作が許可されていない [s]

11058 【NGKI2156】

11059 E_MACV メモリアクセス違反

11060 ・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

11061 いない [s] 【NGKI2157】

11062 E_OBJ オブジェクト状態エラー

11063 ・対象スピンロックは静的APIで生成された [s] 【NGKI2158】

11064 ・対象スピンロックに対してアクセス許可ベクタが設定済み [S]

11065 【NGKI2159】

11066

11067 **【機能】**

11068

11069 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

11070 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

11071 【NGKI2160】。

11072

11073 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

11074 式パラメータである 【NGKI2161】。

11075 -----

11076 del_spn スピンロックの削除 [TMD] 【NGKI2162】

11077

11078 **【C言語API】**

11079 ER ercd = del_spn(ID snid)

11080

11081 **【パラメータ】**

11082 ID snid 対象スピンロックのID番号

11083

11084 **【リターンパラメータ】**

11085 ER ercd 正常終了 (E_OK) またはエラーコード

11086

11087 **【エラーコード】**

11088 E_CTX コンテキストエラー

11089 ・非タスクコンテキストからの呼出し 【NGKI2163】

11090 ・CPUロック状態からの呼出し 【NGKI2164】

11091 E_ID 不正ID番号

11092 ・spnidが有効範囲外 【NGKI2165】

11093 E_NOEXS オブジェクト未登録

11094 ・対象スピンロックが未登録 【NGKI2166】

11095 E_OACV オブジェクトアクセス違反

11096 ・対象スピンロックに対する管理操作が許可されていない [P]

11097 【NGKI2167】

11098 E_OBJ オブジェクト状態エラー

11099 ・対象スピンロックは静的APIで生成された 【NGKI2168】

11100

11101 **【機能】**

11102

11103 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
11104 いは以下の通り．

11105

11106 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
11107 される【NGKI2169】．

11108

11109 **【TOPPERS/FMPカーネルにおける規定】**

11110

11111 FMPカーネルでは，del_spnをサポートしない【FMPS0141】．

11112

11113 **【未決定事項】**

11114

11115 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．

11116

11117 loc_spn スピンロックの取得〔TM〕 【NGKI2170】

11118 iloc_spn スピンロックの取得〔IM〕 【NGKI2171】

11119

11120 **【C言語API】**

11121 ER ercd = loc_spn(ID snid)

11122 ER ercd = iloc_spn(ID snid)

11123

11124 **【パラメータ】**

11125 ID snid 対象スピンロックのID番号

11126

11127 **【リターンパラメータ】**

11128 ER ercd 正常終了（E_OK）またはエラーコード

11129

11130 **【エラーコード】**

11131 E_CTX コンテキストエラー

11132 ・非タスクコンテキストからの呼出し（loc_spnの場合）【NGKI2172】

11133 ・タスクコンテキストからの呼出し（iloc_spnの場合）【NGKI2173】

11134 ・CPUロック状態からの呼出し【NGKI2174】

11135 E_ID 不正ID番号

11136 ・snidが有効範囲外【NGKI2175】

11137 E_NOEXS オブジェクト未登録

11138 ・対象スピンロックが未登録〔D〕【NGKI2176】

11139 E_OACV オブジェクトアクセス違反

11140 ・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
11141 （loc_spnの場合）〔P〕【NGKI2177】

11142

11143 **【機能】**

11144

11145 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
11146 いは以下の通り．

11147

11148 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
11149 ックの取得を試みる【NGKI2178】．ロックが他のプロセッサによって取得されて
11150 いる状態である場合や，他のプロセッサがロックの取得に成功した場合には，

11151 ロックが返却されるまでループによって待ち、返却されたらロックの取得を試
11152 みる【NGKI2179】。これを、ロックの取得に成功するまで繰り返す
11153 【NGKI2180】。
11154
11155 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
11156 【NGKI2181】。また、CPUロックフラグをセットしてCPUロック状態へ遷移し、
11157 サービスコールからリターンする【NGKI2182】。
11158

11159 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
11160 最初にロックを取得できるかは、現時点ではターゲット定義とする【NGKI2183】。
11161

11162 【補足説明】

11163
11164 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
11165 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
11166 ているため、loc_spn/iloc_spnはE_CTXエラーとなる。
11167

11168 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、
11169 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
11170 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
11171 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
11172 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。
11173

11174 try_spn スピンロックの取得（ポーリング） [TM] 【NGKI2184】

11175 itry_spn スピンロックの取得（ポーリング） [IM] 【NGKI2185】

11176 【C言語API】

11178 ER ercd = try_spn(ID spnid)

11179 ER ercd = itry_spn(ID spnid)

11181 【パラメータ】

11182 ID spnid 対象スピンロックのID番号

11184 【リターンパラメータ】

11185 ER ercd 正常終了 (E_OK) またはエラーコード

11187 【エラーコード】

11188 E_CTX コンテキストエラー
11189 ・非タスクコンテキストからの呼出し (try_spnの場合) 【NGKI2186】
11190 ・タスクコンテキストからの呼出し (itry_spnの場合) 【NGKI2187】
11191 ・CPUロック状態からの呼出し 【NGKI2188】
11192 E_ID 不正ID番号
11193 ・spnidが有効範囲外 【NGKI2189】
11194 E_NOEXS オブジェクト未登録
11195 ・対象スピンロックが未登録 [D] 【NGKI2190】
11196 E_OACV オブジェクトアクセス違反
11197 ・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
11198 (try_spnの場合) [P] 【NGKI2191】
11199 E_OBJ オブジェクト状態エラー
1200 ・条件については機能の項を参照

11201
11202
11203
11204
11205
11206
11207
11208
11209
11210
11211
11212
11213
11214
11215
11216
11217
11218
11219
11220
11221
11222
11223
11224
11225
11226
11227
11228
11229
11230
11231
11232
11233
11234
11235
11236
11237
11238
11239
11240
11241
11242
11243
11244
11245
11246
11247
11248
11249
11250

【機能】

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

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

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

【使用上の注意】

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

unl_spn	スピロックの返却 [TM]	【NGKI2196】
iunl_spn	スピロックの返却 [IM]	【NGKI2197】

【C言語API】

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

【パラメータ】

ID	spnid	対象スピロックのID番号
----	-------	--------------

【リターンパラメータ】

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

【エラーコード】

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

【機能】

11251 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
 11252 いは以下の通り．

11253

11254 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
 11255 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
 11256 いない状態とする【NGKI2203】．また，CPUロックフラグをクリアし，CPUロ
 11257 ック解除状態へ遷移する【NGKI2204】．

11258

11259 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
 11260 よって取得されている状態である場合には，E_ILUSEエラーとなる【NGKI2205】．

11261 -----

11262 ref_spn スピンロックの状態参照 [TM] 【NGKI2206】

11263

11264 【C言語API】

11265 ER ercd = ref_spn(ID spnid, T_RSPN *pk_rspn)

11266

11267 【パラメータ】

11268 ID spnid 対象スピンロックのID番号

11269 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
 11270 ポインタ

11271

11272 【リターンパラメータ】

11273 ER ercd 正常終了 (E_OK) またはエラーコード

11274

11275 * スピンロックの現在状態 (パケットの内容)

11276 STAT spnstat ロック状態

11277

11278 【エラーコード】

11279 E_CTX コンテキストエラー

11280 ・非タスクコンテキストからの呼出し【NGKI2207】

11281 ・CPUロック状態からの呼出し【NGKI2208】

11282 E_ID 不正ID番号

11283 ・spnidが有効範囲外【NGKI2209】

11284 E_NOEXS オブジェクト未登録

11285 ・対象スピンロックが未登録 [D] 【NGKI2210】

11286 E_OACV オブジェクトアクセス違反

11287 ・対象スピンロックに対する参照操作が許可されていない [P]
 11288 【NGKI2211】

11289 E_MACV メモリアクセス違反

11290 ・pk_rspnが指すメモリ領域への書込みアクセスが許可されて
 11291 いない) [P] 【NGKI2212】

11292

11293 【機能】

11294

11295 spnidで指定したスピンロック（対象スピンロック）の現在状態を参照する．参
 11296 照した現在状態は，pk_rspnで指定したパケットに返される【NGKI2213】．

11297

11298 spnstatには，対象スピンロックの現在のロック状態を表す次のいずれかの値が
 11299 返される【NGKI2214】．

11300

11301	TSPN_UNL	0x01U	取得されていない状態
11302	TSPN_LOC	0x02U	取得されている状態

【使用上の注意】

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

4.5 メモリプール管理機能

【TOPPERS/SSPカーネルにおける規定】

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

【 μ ITRON4.0仕様との関係】

この仕様では，可変長メモリプール機能はサポートしないこととした．

【仕様決定の理由】

可変長メモリプール機能をサポートしないこととしたのは，メモリ割付けの処理時間とフラグメンテーションの発生を考えると，最適なメモリ管理アルゴリズムはアプリケーション依存となるため，カーネル内で実現するより，ライブラリとして実現する方が適切と考えたためである．

4.5.1 固定長メモリプール

固定長メモリプールは，生成時に決めたサイズのメモリブロック（固定長メモリブロック）を動的に獲得・返却するための同期・通信オブジェクトである．固定長メモリプールは，固定長メモリプールIDと呼ぶID番号で識別する【NGKI2215】．

各固定長メモリプールが持つ情報は次の通り【NGKI2216】．

- ・ 固定長メモリプール属性
- ・ 待ち行列（固定長メモリブロックの獲得待ち状態のタスクのキュー）
- ・ 固定長メモリプール領域
- ・ 固定長メモリプール管理領域
- ・ アクセス許可ベクタ（保護機能対応カーネルの場合）
- ・ 属する保護ドメイン（保護機能対応カーネルの場合）
- ・ 属するクラス（マルチプロセッサ対応カーネルの場合）

待ち行列は，固定長メモリブロックが獲得できるまで待っている状態（固定長メモリブロックの獲得待ち状態）のタスクが，固定長メモリブロックを獲得できる順序でつながれているキューである．

固定長メモリプール領域は，その中から固定長メモリブロックを割り付けるた

11351 めのメモリ領域である。
11352
11353 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
11354 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
11355 のメモリ領域である。
11356
11357 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
11358 用いるオブジェクト管理領域として扱われる【NGKI2217】。
11359
11360 固定長メモリプール属性には、次の属性を指定することができる【NGKI2218】。
11361
11362 TA_TPRI 0x01U 待ち行列をタスクの優先度順にする
11363
11364 TA_TPRIを指定しない場合、待ち行列はFIFO順になる【NGKI2219】。
11365
11366 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。
11367
11368 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
11369 カーネルでは、静的APIによって登録された固定長メモリ
11370 プールの数に一致）【NGKI2220】
11371
11372 【μITRON4.0仕様との関係】
11373
11374 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
11375 クロ（TSZ_MPF）は廃止した。これは、固定長メモリプール領域をアプリケーション
11376 で確保する方法を定めた結果、そのサイズは(blkcnt * ROUND_MPF_T(blksz))
11377 で求めることができるようになったためである。
11378
11379 TNUM_MPFIDは、μITRON4.0仕様に規定されていないカーネル構成マクロである。
11380 -----
11381 CRE_MPF 固定長メモリプールの生成 [S] 【NGKI2221】
11382 acre_mpf 固定長メモリプールの生成 [TD] 【NGKI2222】
11383
11384 【静的API】
11385 CRE_MPF(ID mpfid, { ATR mpfatr, uint_t blkcnt, uint_t blksz,
11386 MPF_T *mpf, void *mpfmb })
11387
11388 【C言語API】
11389 ER_ID mpfid = acre_mpf(const T_CMPF *pk_cmpf)
11390
11391 【パラメータ】
11392 ID mpfid 生成する固定長メモリプールのID番号（CRE_MPF
11393 の場合）
11394 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
11395 トへのポインタ（静的APIを除く）
11396
11397 * 固定長メモリプールの生成情報（パケットの内容）
11398 ATR mpfatr 固定長メモリプール属性
11399 uint_t blkcnt 獲得できる固定長メモリブロックの数
11400 uint_t blksz 固定長メモリブロックのサイズ（バイト数）

11401	MPF_T *	mpf	固定長メモリプール領域の先頭番地
11402	void *	mpfmb	固定長メモリプール管理領域の先頭番地
11403			
11404	【リターンパラメータ】		
11405	ER_ID	mpfid	生成された固定長メモリプールのID番号（正の値）またはエラーコード
11406			
11407			
11408	【エラーコード】		
11409	E_CTX	コンテキストエラー	
11410			・非タスクコンテキストからの呼出し [s] 【NGKI2223】
11411			・CPUロック状態からの呼出し [s] 【NGKI2224】
11412	E_RSATR	予約属性	
11413			・mpfatrが無効 【NGKI2225】
11414			・属する保護ドメインの指定が有効範囲外 [sP] 【NGKI2226】
11415			・属するクラスの指定が有効範囲外 [sM] 【NGKI2227】
11416			・クラスの囲みの中に記述されていない [SM] 【NGKI2228】
11417	E_NOSPT	未サポート機能	
11418			・条件については各カーネルにおける規定の項を参照
11419	E_PAR	パラメータエラー	
11420			・blkcntが0以下 【NGKI2229】
11421			・blkkszが0以下 【NGKI2230】
11422			・その他の条件については機能の項を参照
11423	E_OACV	オブジェクトアクセス違反	
11424			・システム状態に対する管理操作が許可されていない [sP] 【NGKI2231】
11425			
11426	E_MACV	メモリアクセス違反	
11427			・pk_cmpfが指すメモリ領域への読出しアクセスが許可されていない [sP] 【NGKI2232】
11428			
11429	E_NOID	ID番号不足	
11430			・割り付けられる固定長メモリプールIDがない [sD] 【NGKI2233】
11431	E_NOMEM	メモリ不足	
11432			・固定長メモリプール領域が確保できない 【NGKI2234】
11433			・固定長メモリプール管理領域が確保できない 【NGKI2235】
11434	E_OBJ	オブジェクト状態エラー	
11435			・mpfidで指定した固定長メモリプールが登録済み（CRE_MPFの場合） 【NGKI2236】
11436			
11437			・その他の条件については機能の項を参照
11438			
11439	【機能】		
11440			
11441	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		
11442	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、		
11443	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		
11444	ル領域全体が未割当ての状態に初期化される【NGKI2237】。また、待ち行列は		
11445	空の状態に初期化される【NGKI2238】。		
11446			
11447	静的APIにおいては、mpfidはオブジェクト識別名、mpfatr, blkcnt, blkkszは整		
11448	数定数式パラメータ、mpfとmpfmbは一般定数式パラメータである【NGKI2239】。		
11449	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		
11450	とができない【NGKI2240】。		

11451
11452 mpfをNULLとした場合、blkcntとblksizから決まるサイズの固定長メモリプール
11453 領域が、コンフィギュレータまたはカーネルにより確保される【NGKI2241】。
11454
11455 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
11456 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
11457 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
11458 ト中に確保される【NGKI2242】。
11459
11460 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
11461 域が、コンフィギュレータまたはカーネルにより確保される【NGKI2243】。
11462
11463 [mpfにNULL以外を指定した場合]
11464
11465 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
11466 は、アプリケーションで確保しておく必要がある【NGKI2244】。固定長メモリ
11467 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
11468 意している【NGKI2245】。
11469
11470 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
11471
11472 COUNT_MPF_T(blksiz) 固定長メモリブロックのサイズがblksizの固定長メモ
11473 リプール領域を確保するために、固定長メモリブロッ
11474 ク1つあたりに必要なMPF_T型の配列の要素数
11475 ROUND_MPF_T(blksiz) 要素数COUNT_MPF_T(blksiz)のMPF_T型の配列のサイズ
11476 (blksizを、MPF_T型のサイズの倍数になるように大き
11477 い方に丸めた値)
11478
11479 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【NGKI2246】。
11480
11481 MPF_T <固定長メモリプール領域の変数名>[(blkcnt) * COUNT_MPF_T(blksiz)];
11482
11483 この時、mpfには<固定長メモリプール領域の変数名>を指定する【NGKI2247】。
11484
11485 これ以外の方法で固定長メモリプール領域を確保する場合には、上記の配列と
11486 同じサイズのメモリ領域を確保しなければならない【NGKI2248】。また、その
11487 先頭番地がターゲット定義の制約に合致していなければならない。mpfにターゲッ
11488 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
11489 【NGKI2249】。
11490
11491 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
11492 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。
11493 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
11494 ジェクトに含まれていない場合、E_OBJエラーとなる【NGKI2251】。
11495
11496 [mpfmbにNULL以外を指定した場合]
11497
11498 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
11499 管理領域は、アプリケーションで確保しておく必要がある【NGKI2252】。固定
11500 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

11501 用意している【NGKI2253】。

11502

11503 TSZ_MPFMB(blkent) blkentで指定した数の固定長メモリブロックを管理
11504 することができる固定長メモリプール管理領域のサ
11505 イズ (バイト数)

11506 TCNT_MPFMB(blkent) blkentで指定した数の固定長メモリブロックを管理
11507 することができる固定長メモリプール管理領域を確
11508 保するために必要なMB_T型の配列の要素数

11509

11510 これらを用いて固定長メモリプール管理領域を確保する方法は次の通り

11511 【NGKI2254】。

11512

11513 MB_T <固定長メモリプール管理領域の変数名>[TCNT_MPFMB(blkent)];

11514

11515 この時、mpfmbには<固定長メモリプール管理領域の変数名>を指定する

11516 【NGKI2255】。

11517

11518 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
11519 した時には、E_PARエラーとなる【NGKI2256】。また、保護機能対応カーネルに
11520 おいて、mpfmbで指定した固定長メモリプール管理領域がカーネル専用のメモリ
11521 オブジェクトに含まれない場合、E_OBJエラーとなる【NGKI2257】。

11522

11523 【補足説明】

11524

11525 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
11526 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
11527 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
11528 ンの責任である。

11529

11530 【TOPPERS/ASPカーネルにおける規定】

11531

11532 ASPカーネルでは、CRE_MPFのみをサポートする【ASPS0164】。また、mpfmbには
11533 NULLのみを指定することができる。NULL以外を指定した場合には、E_NOSPTエラー
11534 となる【ASPS0166】。ただし、動的生成機能拡張パッケージでは、acre_mpfも
11535 サポートする【ASPS0167】。acre_mpfに対しては、mpfmbにNULL以外を指定でき
11536 ないという制限はない【ASPS0168】。

11537

11538 【TOPPERS/FMPカーネルにおける規定】

11539

11540 FMPカーネルでは、CRE_MPFのみをサポートする【FMPS0142】。また、mpfmbには
11541 NULLのみを指定することができる。NULL以外を指定した場合には、E_NOSPTエラー
11542 となる【FMPS0144】。

11543

11544 【TOPPERS/HRP2カーネルにおける規定】

11545

11546 HRP2カーネルでは、CRE_MPFのみをサポートする【HRPS0136】。また、mpfmbに
11547 はNULLのみを指定することができる。NULL以外を指定した場合には、E_NOSPTエ
11548 ラーとなる【HRPS0138】。

11549

11550 動的生成機能拡張パッケージでは、acre_mpfもサポートする【HRPS0197】。

11551 acre_mpfに対しては、mpfmbにNULL以外を指定できないという制限はない
11552 【HRPS0198】。ただし、mpfにNULLが指定されるとカーネルが固定長メモリープ
11553 ル領域を確保する機能はサポートしない。mpfにNULLを指定した場合には、
11554 E_NOSPTエラーとなる【HRPS0199】。

11555
11556 【μITRON4.0仕様との関係】
11557

11558 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
11559 定長メモリープ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、
11560 μITRON4.0/PX仕様にあわせて、固定長メモリープ生成情報に、mpfmbを追加
11561 した。

11562
11563 【μITRON4.0/PX仕様との関係】
11564

11565 TCNT_MPFMBを新設し、固定長メモリープ管理領域をアプリケーションで確保
11566 する方法を規定した。

11567 -----

11568 AID_MPF 割付け可能な固定長メモリープIDの数の指定 [SD] 【NGKI2258】
11569

11570 【静的API】
11571 AID_MPF(uint_t nompf)
11572

11573 【パラメータ】
11574 uint_t nompf 割付け可能な固定長メモリープIDの数
11575

11576 【エラーコード】
11577 E_RSATR 予約属性
11578 ・保護ドメインの囲みの中に記述されている [P] 【NGKI3436】
11579 ・クラスの囲みの中に記述されていない [M] 【NGKI2259】
11580 E_PAR パラメータエラー
11581 ・nompfが負の値【NGKI3284】
11582

11583 【機能】
11584

11585 nompfで指定した数の固定長メモリープIDを、固定長メモリープを生成する
11586 サービスコールによって割付け可能な固定長メモリープIDとして確保する
11587 【NGKI2260】。
11588

11589 nompfは整数定数式パラメータである【NGKI2261】。
11590

11591 【TOPPERS/ASPカーネルにおける規定】
11592

11593 ASPカーネルの動的生成機能拡張パッケージでは、AID_MPFをサポートする
11594 【ASPS0216】。
11595

11596 【TOPPERS/HRP2カーネルにおける規定】
11597

11598 HRP2カーネルの動的生成機能拡張パッケージでは、AID_MPFをサポートする
11599 【HRPS0217】。
11600 -----

11601 SAC_MPF 固定長メモリプールのアクセス許可ベクタの設定 [SP] 【NGKI2262】
 11602 sac_mpf 固定長メモリプールのアクセス許可ベクタの設定 [TPD] 【NGKI2263】
 11603
 11604 【静的API】
 11605 SAC_MPF(ID mpfid, { ACPTN acptn1, ACPTN acptn2,
 11606 ACPTN acptn3, ACPTN acptn4 })
 11607
 11608 【C言語API】
 11609 ER ercd = sac_mpf(ID mpfid, const ACVCT *p_acvct)
 11610
 11611 【パラメータ】
 11612 ID mpfid 対象固定長メモリプールのID番号
 11613 ACVCT * p_acvct アクセス許可ベクタを入れたパケットへのポ
 11614 インタ（静的APIを除く）
 11615
 11616 *アクセス許可ベクタ（パケットの内容）
 11617 ACPTN acptn1 通常操作1のアクセス許可パターン
 11618 ACPTN acptn2 通常操作2のアクセス許可パターン
 11619 ACPTN acptn3 管理操作のアクセス許可パターン
 11620 ACPTN acptn4 参照操作のアクセス許可パターン
 11621
 11622 【リターンパラメータ】
 11623 ER ercd 正常終了（E_OK）またはエラーコード
 11624
 11625 【エラーコード】
 11626 E_CTX コンテキストエラー
 11627 ・非タスクコンテキストからの呼出し [s] 【NGKI2264】
 11628 ・CPUロック状態からの呼出し [s] 【NGKI2265】
 11629 E_ID 不正ID番号
 11630 ・mpfidが有効範囲外 [s] 【NGKI2266】
 11631 E_RSATR 予約属性
 11632 ・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
 11633 （対象固定長メモリプールが無所属の場合は、保護ドメイ
 11634 ンの囲みの外）に記述されていない [S] 【NGKI2267】
 11635 ・対象固定長メモリプールが属するクラスの囲みの中に記述
 11636 されていない [SM] 【NGKI2268】
 11637 E_NOEXS オブジェクト未登録
 11638 ・対象固定長メモリプールが未登録 【NGKI2269】
 11639 E_OACV オブジェクトアクセス違反
 11640 ・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
 11641 ない [s] 【NGKI2270】
 11642 E_MACV メモリアクセス違反
 11643 ・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
 11644 いない [s] 【NGKI2271】
 11645 E_OBJ オブジェクト状態エラー
 11646 ・対象固定長メモリプールは静的APIで生成された [s] 【NGKI2272】
 11647 ・対象固定長メモリプールに対してアクセス許可ベクタが設
 11648 定済み [S] 【NGKI2273】
 11649
 11650 【機能】

11651
11652 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
11653 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
11654 設定する【NGKI2274】．対象固定長メモリプールの固定長メモリプール領域が
11655 コンフィギュレータまたはカーネルにより確保されたものである場合には、固
11656 定長メモリプール領域のアクセス許可ベクタも、各パラメータで指定した値に
11657 設定する【NGKI2275】．
11658
11659 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
11660 式パラメータである【NGKI2276】．
11661
11662 【TOPPERS/HRP2カーネルにおける規定】
11663
11664 HRP2カーネルでは、SAC_MPFのみをサポートする【HRPS0139】．ただし、動的生
11665 成機能拡張パッケージでは、sac_mpfもサポートする【HRPS0200】．
11666 -----
11667 del_mpf 固定長メモリプールの削除〔TD〕【NGKI2277】
11668
11669 【C言語API】
11670 ER ercd = del_mpf(ID mpfid)
11671
11672 【パラメータ】
11673 ID mpfid 対象固定長メモリプールのID番号
11674
11675 【リターンパラメータ】
11676 ER ercd 正常終了（E_OK）またはエラーコード
11677
11678 【エラーコード】
11679 E_CTX コンテキストエラー
11680 ・非タスクコンテキストからの呼出し【NGKI2278】
11681 ・CPUロック状態からの呼出し【NGKI2279】
11682 E_ID 不正ID番号
11683 ・mpfidが有効範囲外【NGKI2280】
11684 E_NOEXS オブジェクト未登録
11685 ・対象固定長メモリプールが未登録【NGKI2281】
11686 E_OACV オブジェクトアクセス違反
11687 ・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
11688 ない〔P〕【NGKI2282】
11689 E_OBJ オブジェクト状態エラー
11690 ・対象固定長メモリプールは静的APIで生成された【NGKI2283】
11691
11692 【機能】
11693
11694 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．
11695 具体的な振舞いは以下の通り．
11696
11697 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
11698 用の状態に戻される【NGKI2284】．また、対象固定長メモリプールの待ち行列
11699 につながれたタスクは、待ち行列の先頭のタスクから順に待ち解除される
11700 【NGKI2285】．待ち解除されたタスクには、待ち状態となったサービスコール

11701 からE_DLTエラーが返る【NGKI2286】。

11702

11703 【使用上の注意】

11704

11705 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
11706 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
11707 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
11708 み禁止時間が長くなるため、注意が必要である。

11709

11710 【TOPPERS/ASPカーネルにおける規定】

11711

11712 ASPカーネルでは、del_mpfをサポートしない【ASPS0170】。ただし、動的生成
11713 機能拡張パッケージでは、del_mpfをサポートする【ASPS0171】。

11714

11715 【TOPPERS/FMPカーネルにおける規定】

11716

11717 FMPカーネルでは、del_mpfをサポートしない【FMPS0146】。

11718

11719 【TOPPERS/HRP2カーネルにおける規定】

11720

11721 HRP2カーネルでは、del_mpfをサポートしない【HRPS0140】。ただし、動的生成
11722 機能拡張パッケージでは、del_mpfをサポートする【HRPS0201】。

11723

11724 get_mpf 固定長メモリブロックの獲得 [T] 【NGKI2287】

11725 pget_mpf 固定長メモリブロックの獲得（ポーリング） [T] 【NGKI2288】

11726 tget_mpf 固定長メモリブロックの獲得（タイムアウト付き） [T] 【NGKI2289】

11727

11728 【C言語API】

11729 ER ercd = get_mpf(ID mpfid, void **p_blk)

11730 ER ercd = pget_mpf(ID mpfid, void **p_blk)

11731 ER ercd = tget_mpf(ID mpfid, void **p_blk, TMO tmout)

11732

11733 【パラメータ】

11734 ID mpfid 対象固定長メモリプールのID番号

11735 void ** p_blk 獲得した固定長メモリブロックの先頭番地を入
11736 れるメモリ領域へのポインタ

11737 TMO tmout タイムアウト時間（twai_mpfの場合）

11738

11739 【リターンパラメータ】

11740 ER ercd 正常終了（E_OK）またはエラーコード

11741 void * blk 獲得した固定長メモリブロックの先頭番地

11742

11743 【エラーコード】

11744 E_CTX コンテキストエラー

11745 ・非タスクコンテキストからの呼出し【NGKI2290】

11746 ・CPUロック状態からの呼出し【NGKI2291】

11747 ・ディスパッチ保留状態からの呼出し（pget_mpfを除く）

11748 【NGKI2292】

11749 E_NOSPT 未サポート機能

11750 ・制約タスクからの呼出し（pget_mpfを除く）【NGKI2293】

11751	E_ID	不正ID番号
11752		・mpfidが有効範囲外【NGKI2294】
11753	E_PAR	パラメータエラー
11754		・tmoutが無効 (tget_mpfの場合) 【NGKI2295】
11755	E_NOEXS	オブジェクト未登録
11756		・対象固定長メモリプールが未登録 [D] 【NGKI2296】
11757	E_OACV	オブジェクトアクセス違反
11758		・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
11759		ない [P] 【NGKI2297】
11760	E_MACV	メモリアクセス違反
11761		・p_blkが指すメモリ領域への書込みアクセスが許可されてい
11762		ない) [P] 【NGKI2298】
11763	E_TMOUT	ポーリング失敗またはタイムアウト (get_mpfを除く) 【NGKI2299】
11764	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)
11765		【NGKI2300】
11766	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)
11767		【NGKI2301】

11768 【機能】

11769
11770
11771 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
11772 モリブロックを獲得し, その先頭番地をp_blkが指すメモリ領域に返す. 具体的
11773 な振舞いは以下の通り.

11774
11775 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
11776 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
11777 モリブロックが1つ割り付けられ, その先頭番地がblkに返される【NGKI2302】.

11778
11779 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
11780 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【NGKI2303】.

11781 -----
11782 rel_mpf 固定長メモリブロックの返却 [T] 【NGKI2304】

11783 【C言語API】

11784 ER ercd = rel_mpf(ID mpfid, void *blk)

11786 【パラメータ】

11788	ID	mpfid	対象固定長メモリプールのID番号
11789	void *	blk	返却する固定長メモリブロックの先頭番地

11790 【リターンパラメータ】

11791	ER	ercd	正常終了 (E_OK) またはエラーコード
-------	----	------	-----------------------

11793 【エラーコード】

11795	E_CTX	コンテキストエラー
11796		・非タスクコンテキストからの呼出し【NGKI2305】
11797		・CPUロック状態からの呼出し【NGKI2306】
11798	E_ID	不正ID番号
11799		・mpfidが有効範囲外【NGKI2307】
11800	E_PAR	パラメータエラー

11801 ・条件については機能の項を参照
11802 E_NOEXS オブジェクト未登録
11803 ・対象固定長メモリプールが未登録 [D] 【NGKI2308】
11804 E_OACV オブジェクトアクセス違反
11805 ・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
11806 ない [P] 【NGKI2309】
11807

11808 **【機能】**

11809
11810 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
11811 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．
11812

11813 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
11814 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
11815 る【NGKI2310】．待ち解除されたタスクには、待ち状態となったサービスコー
11816 ルからE_OKが返る【NGKI2311】．
11817

11818 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
11819 は、対象固定長メモリプールのメモリプール領域に返却される【NGKI2312】．
11820

11821 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
11822 でない場合には、E_PARエラーとなる【NGKI2313】．
11823

11824 -----
11824 ini_mpf 固定長メモリプールの再初期化 [T] 【NGKI2314】
11825

11826 **【C言語API】**

11827 ER ercd = ini_mpf(ID mpfid)
11828

11829 **【パラメータ】**

11830 ID mpfid 対象固定長メモリプールのID番号
11831

11832 **【リターンパラメータ】**

11833 ER ercd 正常終了 (E_OK) またはエラーコード
11834

11835 **【エラーコード】**

11836 E_CTX コンテキストエラー
11837 ・非タスクコンテキストからの呼出し【NGKI2315】
11838 ・CPUロック状態からの呼出し【NGKI2316】
11839 E_ID 不正ID番号
11840 ・mpfidが有効範囲外【NGKI2317】
11841 E_NOEXS オブジェクト未登録
11842 ・対象固定長メモリプールが未登録 [D] 【NGKI2318】
11843 E_OACV オブジェクトアクセス違反
11844 ・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
11845 ない [P] 【NGKI2319】
11846

11847 **【機能】**

11848
11849 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
11850 る．具体的な振舞いは以下の通り．

11851
11852 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
11853 れる【NGKI2320】。また、対象固定長メモリプールの待ち行列につながれたタ
11854 スクは、待ち行列の先頭のタスクから順に待ち解除される【NGKI2321】。待ち
11855 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
11856 る【NGKI2322】。

11858 【使用上の注意】

```

11860     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
11861     およびカーネル内での割り込み禁止時間が、待ち解除されるタスクの数に比例し
11862     て長くなる。特に、多くのタスクが待ち解除される場合、カーネル内での割り込
11863     み禁止時間が長くなるため、注意が必要である。

```

11865 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
11866 つのは、アプリケーションの責任である。

11868 【μITRON4.0仕様との関係】

11870 μ ITRON4.0仕様に定義されていないサービスコールである.

11872 ref mpf 固定長メモリプールの状態参照 [T] 【NGKI2323】

11874 【C言語API】

```
11875      ER ercd = ref mpf(ID mpfid, T RMPF *pk rmpf)
```

11877 【パラメータ】

11878	ID	mpfid	対象固定長メモリプールのID番号
11879	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
11880			

11882 【リターンパラメータ】

11883	ER	ercd	正常終了 (E OK) またはエラーコード
-------	----	------	-----------------------

11885 *固定長メモリプールの現在状態 (パケットの内容)

11886	ID	wtskid	固定長メモリプールの待ち行列の先頭のタスク
11887			のID番号
11888	uint_t	fbblkcnt	固定長メモリプール領域の空きメモリ領域に割
11889			り付けることができる固定長メモリブロックの
11890			数

11892 【エラーコード】

11893	E_CTX	コンテキストエラー
11894		・非タスクコンテキストからの呼出し【NGKI2324】
11895		・CPUロック状態からの呼出し【NGKI2325】
11896	E_ID	不正ID番号
11897		・mpfidが有効範囲外【NGKI2326】
11898	E_NOEXS	オブジェクト未登録
11899		・対象固定長メモリプールが未登録 [D] 【NGKI2327】
11900	E_OACV	オブジェクトアクセス違反

11901 ・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
11902 ない [P] 【NGKI2328】
11903 E_MACV メモリアクセス違反
11904 ・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
11905 いない) [P] 【NGKI2329】
11906

11907 【機能】

11908
11909 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
11910 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
11911 【NGKI2330】．
11912

11913 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
11914 TSK_NONE (=0) が返る 【NGKI2331】．
11915

11916 【使用上の注意】

11917
11918 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
11919 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
11920 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
11921 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．
11922 -----

11923

11924

11924 4.6 時間管理機能

11925

11926 4.6.1 システム時刻管理

11927

11928 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
11929 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
11930 るカーネルオブジェクトである 【NGKI3603】．システム時刻は、符号無しの整
11931 数型であるSYSTIM型で表され、単位はミリ秒である 【NGKI2332】．
11932

11933 システム時刻は、カーネルの初期化時に0に初期化される 【NGKI2333】．タイム
11934 ティックを通知するためのタイマ割込みが発生する毎にカーネルによって更新
11935 され、SYSTIM型で表せる最大値 (ULONG_MAX) を超えると0に戻される
11936 【NGKI2334】．タイムティックの周期は、ターゲット定義である 【NGKI2335】．
11937 また、システム時刻の精度はターゲットに依存する 【NGKI2336】．
11938

11939

11940 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
11941 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
11942 み存在する 【NGKI2337】．マルチプロセッサ対応カーネルでローカルタイマ方
11943 式を用いている場合には、システム時刻は、プロセッサ毎に存在する
11944 【NGKI2338】．ローカルタイマ方式とグローバルタイマ方式については、
11945 「2.3.4 マルチプロセッサ対応」の節を参照すること．

11946

11947 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、
11948 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
11949 られているプロセッサのシステム時刻が用いられる 【NGKI2339】．また、周期
11950 ハンドラとアラームハンドラの起動には、それが割り付けられているプロセッ
 サのシステム時刻が用いられる 【NGKI2340】．これらの処理単位がマイグレー

ションする場合には、用いられるシステム時刻も変更される【NGKI2341】。この場合にも、イベントの処理が行われるのは、基準時刻から相対時間によって指定した以上の時間が経過した後となるという規則は維持される【NGKI2342】。

1回のタイムティックの発生により、複数のイベントの処理を行うべき状況になった場合、それらの処理の間の処理順序は規定されない【NGKI2343】。

性能評価用システム時刻は、性能評価に使用することを目的とした、システム時刻よりも精度の高い時刻である。性能評価用システム時刻は、符号無しの整数型であるSYSUTM型で表され、単位はマイクロ秒である【NGKI2344】。ただし、実際の精度はターゲットに依存する【NGKI2345】。

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

システム時刻管理機能に関連するカーネル構成マクロは次の通り。

TIC_NUME	タイムティックの周期（単位はミリ秒）の分子	【NGKI2347】
TIC_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【NGKI2348】

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、時間管理機能をサポートしない【SSPS0129】。

【使用上の注意】

タイムティックを通知するためのタイマ割込みが長時間マスクされた場合（タイマ割込みより優先して実行される割込み処理が長時間続けて実行された場合を含む）や、シミュレーション環境においてシミュレータのプロセスが長時間スケジュールされなかった場合には、システム時刻が正しく更新されない可能性があるため、注意が必要である。

【μITRON4.0仕様との関係】

システム時刻を設定するサービスコール（set_tim）を廃止した。また、タイムティックを供給する機能は、カーネル内に実現することとし、そのためのサービスコール（isig_tim）は廃止した。

【μITRON4.0/PX仕様との関係】

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

get_tim システム時刻の参照 [T] 【NGKI2349】

【C言語API】

```

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


12001

12002 **【パラメータ】**

12003 SYSTIM * p_systim システム時刻を入れるメモリ領域へのポインタ

12004

12005 **【リターンパラメータ】**

12006 ER ercd 正常終了 (E_OK) またはエラーコード

12007 SYSTIM systim システム時刻の現在値

12008

12009 **【エラーコード】**

12010 E_CTX コンテキストエラー

12011 ・非タスクコンテキストからの呼出し【NGKI2350】

12012 ・CPUロック状態からの呼出し【NGKI2351】

12013 E_OACV オブジェクトアクセス違反

12014 ・システム状態に対する参照操作が許可されていない [P]

12015 【NGKI2352】

12016 E_MACV メモリアクセス違反

12017 ・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 [P] 【NGKI2353】

12018

12019

12020 **【機能】**

12021

12022 システム時刻の現在値を参照する．参照したシステム時刻は，p_systimが指す

12023 メモリ領域に返される【NGKI2354】．

12024

12025 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，

12026 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

12027 【NGKI2355】．

12028

12029 **【補足説明】**

12030

12031 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

12032 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．

12033 -----

12034 get_utm 性能評価用システム時刻の参照 [TI] 【NGKI2356】

12035

12036 **【C言語API】**

12037 ER ercd = get_utm(SYSUTM *p_sysutm)

12038

12039 **【パラメータ】**

12040 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

12041

12042

12043 **【リターンパラメータ】**

12044 ER ercd 正常終了 (E_OK) またはエラーコード

12045 SYSUTM sysutm 性能評価用システム時刻の現在値

12046

12047 **【エラーコード】**

12048 E_NOSPT 未サポート機能

12049 ・条件については機能の項を参照

12050 E_MACV メモリアクセス違反

12051 ・ p_sysutmが指すメモリ領域へ書込みアクセスが許可されて
12052 いない) [P] 【NGKI2357】
12053

12054 【機能】

12055
12056 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
12057 刻は、p_sysutmが指すメモリ領域に返される【NGKI2358】．
12058

12059 get_utmは、任意の状態から呼び出すことができる【NGKI2359】．タスクコンテ
12060 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
12061 状態であっても呼び出すことができる．
12062

12063 ターゲット定義で、get_utmがサポートされていない場合がある【NGKI2360】．
12064 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
12065 義される【NGKI2361】．サポートされていない場合にget_utmを呼び出すと、
12066 E_NOSPTエラーが返るか、リンク時にエラーとなる【NGKI2362】．
12067

12068 【使用方法】

12069
12070 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．
12071 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
12072 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
12073 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．
12074

12075 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
12076 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
12077 限らない．
12078

12079 【使用上の注意】

12080
12081 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
12082 ない．
12083

12084 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
12085 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
12086 込みの応答性が低下する．
12087

12088 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
12089 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
12090 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
12091 考慮に入れて使用しなければならない．
12092

12093 【μ ITRON4.0仕様との関係】

12094 μ ITRON4.0仕様に定義されていないサービスコールである．
12095 -----
12096

12097 4.6.2 周期ハンドラ

12098 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．
12099
12100

12101 周期ハンドラは、周期ハンドラIDと呼ぶID番号によって識別する【NGKI2363】。

12102

12103 各周期ハンドラが持つ情報は次の通り【NGKI2364】。

12104

12105 • 周期ハンドラ属性

12106 • 周期ハンドラの動作状態

12107 • 次に周期ハンドラを起動する時刻

12108 • 拡張情報

12109 • 周期ハンドラ先頭番地

12110 • 起動周期

12111 • 起動位相

12112 • アクセス許可ベクタ（保護機能対応カーネルの場合）

12113 • 属する保護ドメイン（保護機能対応カーネルの場合）

12114 • 属するクラス（マルチプロセッサ対応カーネルの場合）

12115

12116 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

12117 対時間後である【NGKI2365】。

12118

12119 起動位相 + 起動周期 × (n-1) n=1, 2, ...

12120

12121 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

12122 かをとる【NGKI2366】。周期ハンドラを動作している状態にすることを動作開

12123 始、動作していない状態にすることを動作停止という。

12124

12125 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

12126 になると、周期ハンドラの起動処理が行われる【NGKI2367】。具体的には、拡張

12127 情報をパラメータとして、周期ハンドラが呼び出される【NGKI2368】。

12128

12129 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

12130 インは、カーネルドメインに限られる【NGKI2369】。

12131

12132 周期ハンドラ属性には、次の属性を指定することができる【NGKI2370】。

12133

12134 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

12135 TA_PHS 0x04U 周期ハンドラを生成した時刻を基準時刻とする

12136

12137 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

12138 していない状態となる【NGKI2371】。

12139

12140 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

12141 ドラを起動する時刻の基準時刻となる【NGKI2372】。TA_PHSを指定した場合には、

12142 周期ハンドラを生成した時刻（静的APIで生成した場合にはカーネルの起動

12143 時刻）が、基準時刻となる【NGKI2373】。

12144

12145 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

12146 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

12147 ハンドラの起動処理時に設定される【NGKI2374】。

12148

12149 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、

12150 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

12151 あるクラスにのみ属することができる【NGKI2375】。すなわち、周期ハンドラ
12152 は、システム時刻管理プロセッサによって実行される。

12153

12154 C言語による周期ハンドラの記述形式は次の通り【NGKI2376】。

12155

```
12156     void cyclic_handler(intptr_t exinf)
12157     {
12158         周期ハンドラ本体
12159     }
```

12160

12161 exinfには、周期ハンドラの拡張情報が渡される【NGKI2377】。

12162

12163 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。

12164

```
12165     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
12166                   ネルでは、静的APIによって登録された周期ハンドラの数
12167                   に一致）【NGKI2378】
```

12168

12169 【TOPPERS/ASPカーネルにおける規定】

12170

12171 ASPカーネルでは、TA_PHS属性の周期ハンドラをサポートしない【ASPS0172】。

12172

12173 【TOPPERS/FMPカーネルにおける規定】

12174

12175 FMPカーネルでは、TA_PHS属性の周期ハンドラをサポートしない【FMPS0147】。

12176

12177 【TOPPERS/HRP2カーネルにおける規定】

12178

12179 HRP2カーネルでは、TA_PHS属性の周期ハンドラをサポートしない【HRPS0141】。

12180

12181 【μITRON4.0仕様との関係】

12182

12183 TNUM_CYCIDは、μITRON4.0仕様に規定されていないカーネル構成マクロである。

12184 -----

```
12185     CRE_CYC      周期ハンドラの生成〔S〕【NGKI2379】
12186     acre_cyc     周期ハンドラの生成〔TD〕【NGKI2380】
```

12187

12188 【静的API】

```
12189     CRE_CYC(ID cycid, { ATR cycatr, intptr_t exinf, CYCHDR cychdr,
12190                      RELTIM cycetim, RELTIM cycphs })
```

12191

12192 【C言語API】

```
12193     ER_ID cycid = acre_cyc(const T_CCYC *pk_ccyc)
```

12194

12195 【パラメータ】

```
12196     ID      cycid      生成する周期ハンドラのID番号（CRE_CYCの場合）
12197     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
12198                       ポインタ（静的APIを除く）
```

12199

12200 *周期ハンドラの生成情報（パケットの内容）

12201	ATR	cycatr	周期ハンドラ属性
12202	intptr_t	exinf	周期ハンドラの拡張情報
12203	CYCHDR	cychdr	周期ハンドラの手頭番地
12204	RELTIM	cyctim	周期ハンドラの起動周期
12205	RELTIM	cycphs	周期ハンドラの起動位相
12206			
12207	【リターンパラメータ】		
12208	ER_ID	cycid	生成された周期ハンドラのID番号（正の値）またはエラーコード
12209			
12210			
12211	【エラーコード】		
12212	E_CTX	コンテキストエラー	
12213			・非タスクコンテキストからの呼出し [s] 【NGKI2381】
12214			・CPUロック状態からの呼出し [s] 【NGKI2382】
12215	E_RSATR	予約属性	
12216			・cycatrが無効 【NGKI2383】
12217			・属する保護ドメインの指定が有効範囲外またはカーネルドメイン以外 [sP] 【NGKI2384】
12218			・カーネルドメインの囲みの中に記述されていない [SP] 【NGKI2385】
12219			・属するクラスの指定が有効範囲外 [sM] 【NGKI2386】
12220			・クラスの囲みの中に記述されていない [SM] 【NGKI2387】
12221			・その他の条件については機能の項を参照
12222	E_PAR	パラメータエラー	
12223			・cychdrがプログラムの手頭番地として正しくない 【NGKI2388】
12224			・cyctimが有効範囲（0より大きくTMAX_RELTIM以下）外 【NGKI2397】
12225			・cycphsが有効範囲（0以上TMAX_RELTIM以下）外 【NGKI2399】
12226	E_OACV	オブジェクトアクセス違反	
12227			・システム状態に対する管理操作が許可されていない [sP] 【NGKI2389】
12228	E_MACV	メモリアクセス違反	
12229			・pk_ccycが指すメモリ領域への読出しアクセスが許可されていない [sP] 【NGKI2390】
12230	E_NOID	ID番号不足	
12231			・割り付けられる周期ハンドラIDがない [sD] 【NGKI2391】
12232	E_OBJ	オブジェクト状態エラー	
12233			・cycidで指定した周期ハンドラが登録済み（CRE_CYCの場合） 【NGKI2392】
12234			
12235			
12236			
12237			
12238			
12239			
12240	【機能】		
12241			
12242	各パラメータで指定した周期ハンドラ生成情報に従って、周期ハンドラを生成する。具体的な振舞いは以下の通り。		
12243			
12244			
12245	cycatrにTA_STAを指定した場合、対象周期ハンドラは動作している状態となる 【NGKI2393】。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APIの場合はカーネルの起動時刻）から、cycphsで指定した相対時間後に設定される 【NGKI2394】。cycphsにcyctimより大きい値を指定してもよい 【NGKI2400】。		
12246			
12247			
12248			
12249			
12250			

12251 cychdr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
12252 初期化される【NGKI2395】。

12253

12254 静的APIにおいては、cycidはオブジェクト識別名、cycatr, cyctim, cycphsは
12255 整数定数式パラメータ、exinfとcyhdrは一般定数式パラメータである
12256 【NGKI2396】。

12257

12258 マルチプロセッサ対応カーネルでグローバルタイム方式を用いている場合で、
12259 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
12260 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【NGKI2401】。

12261

12262 【補足説明】

12263

12264 静的APIにおいて、cycatrにTA_STAを、cycphsに0を指定した場合、周期ハンド
12265 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。
12266 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cycatrにTA_STA
12267 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
12268ギュレータが警告メッセージを出力する。

12269

12270 【TOPPERS/ASPカーネルにおける規定】

12271

12272 ASPカーネルでは、CRE_CYCのみをサポートする【ASPS0173】。ただし、TA_PHS
12273 属性の周期ハンドラはサポートしない【ASPS0174】。動的生成機能拡張パッケー
12274 ジでは、acre_cycもサポートする【ASPS0175】。

12275

12276 【TOPPERS/FMPカーネルにおける規定】

12277

12278 FMPカーネルでは、CRE_CYCのみをサポートする【FMPS0148】。ただし、TA_PHS
12279 属性の周期ハンドラはサポートしない【FMPS0149】。

12280

12281 【TOPPERS/HRP2カーネルにおける規定】

12282

12283 HRP2カーネルでは、CRE_CYCのみをサポートする【HRPS0142】。ただし、
12284 TA_PHS属性の周期ハンドラはサポートしない【HRPS0143】。動的生成機能拡張
12285 パッケージでは、acre_cycもサポートする【HRPS0202】。

12286

12287 【μITRON4.0仕様との関係】

12288

12289 cyhdrのデータ型をCYCHDRに変更した。また、cycphsにcyctimより大きい値を
12290 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
12291 した。

12292 -----

12293 AID_CYC 割付け可能な周期ハンドラIDの数の指定〔SD〕【NGKI2402】

12294

12295 【静的API】

12296 AID_CYC(uint_t nocyc)

12297

12298 【パラメータ】

12299 uint_t nocyc 割付け可能な周期ハンドラIDの数

12300

12351 **【リターンパラメータ】**
12352 ER ercd 正常終了 (E_OK) またはエラーコード
12353
12354 **【エラーコード】**
12355 E_CTX コンテキストエラー
12356 ・非タスクコンテキストからの呼出し [s] 【NGKI2410】
12357 ・CPUロック状態からの呼出し [s] 【NGKI2411】
12358 E_ID 不正ID番号
12359 ・cycidが有効範囲外 [s] 【NGKI2412】
12360 E_RSATR 予約属性
12361 ・カーネルドメインの囲みの中に記述されていない [S] 【NGKI2413】
12362 ・対象周期ハンドラが属するクラスの囲みの中に記述されて
12363 いない [SM] 【NGKI2414】
12364 E_NOEXS オブジェクト未登録
12365 ・対象周期ハンドラが未登録 【NGKI2415】
12366 E_OACV オブジェクトアクセス違反
12367 ・対象周期ハンドラに対する管理操作が許可されていない [s]
12368 【NGKI2416】
12369 E_MACV メモリアクセス違反
12370 ・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
12371 いない [s] 【NGKI2417】
12372 E_OBJ オブジェクト状態エラー
12373 ・対象周期ハンドラは静的APIで生成された [s] 【NGKI2418】
12374 ・対象周期ハンドラに対してアクセス許可ベクタが設定済み
12375 [S] 【NGKI2419】
12376
12377 **【機能】**
12378
12379 cycidで指定した周期ハンドラ（対象周期ハンドラ）のアクセス許可ベクタ（4
12380 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
12381 **【NGKI2420】**。
12382
12383 静的APIにおいては、cycidはオブジェクト識別名、acptn1～acptn4は整数定数
12384 式パラメータである **【NGKI2421】**。
12385
12386 **【TOPPERS/HRP2カーネルにおける規定】**
12387
12388 HRP2カーネルでは、SAC_CYCのみをサポートする **【HRPS0144】**。ただし、動的生
12389 成機能拡張パッケージでは、sac_cycもサポートする **【HRPS0203】**。
12390 -----
12391 del_cyc 周期ハンドラの削除 [TD] **【NGKI2422】**
12392
12393 **【C言語API】**
12394 ER ercd = del_cyc(ID cycid)
12395
12396 **【パラメータ】**
12397 ID cycid 対象周期ハンドラのID番号
12398
12399 **【リターンパラメータ】**
12400 ER ercd 正常終了 (E_OK) またはエラーコード

12401

12402 **【エラーコード】**

12403 E_CTX コンテキストエラー

12404 ・非タスクコンテキストからの呼出し【NGKI2423】

12405 ・CPUロック状態からの呼出し【NGKI2424】

12406 E_ID 不正ID番号

12407 ・cycidが有効範囲外【NGKI2425】

12408 E_NOEXS オブジェクト未登録

12409 ・対象周期ハンドラが未登録【NGKI2426】

12410 E_OACV オブジェクトアクセス違反

12411 ・対象周期ハンドラに対する管理操作が許可されていない〔P〕

12412 【NGKI2427】

12413 E_OBJ オブジェクト状態エラー

12414 ・対象周期ハンドラは静的APIで生成された【NGKI2428】

12415

12416 **【機能】**

12417

12418 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

12419 いは以下の通り．

12420

12421 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

12422 される【NGKI2429】．対象周期ハンドラが動作している状態であった場合には，

12423 動作していない状態にされた後に，登録が解除される【NGKI2430】．

12424

12425 **【TOPPERS/ASPカーネルにおける規定】**

12426

12427 ASPカーネルでは，del_cycをサポートしない【ASPS0177】．ただし，動的生成

12428 機能拡張パッケージでは，del_cycをサポートする【ASPS0178】．

12429

12430 **【TOPPERS/FMPカーネルにおける規定】**

12431

12432 FMPカーネルでは，del_cycをサポートしない【FMPS0151】．

12433

12434 **【TOPPERS/HRP2カーネルにおける規定】**

12435

12436 HRP2カーネルでは，del_cycをサポートしない【HRPS0145】．ただし，動的生成

12437 機能拡張パッケージでは，del_cycをサポートする【HRPS0204】．

12438 -----

12439 sta_cyc 周期ハンドラの動作開始〔T〕【NGKI2431】

12440

12441 **【C言語API】**

12442 ER ercd = sta_cyc(ID cycid)

12443

12444 **【パラメータ】**

12445 ID cycid 対象周期ハンドラのID番号

12446

12447 **【リターンパラメータ】**

12448 ER ercd 正常終了（E_OK）またはエラーコード

12449

12450 **【エラーコード】**

12451	E_CTX	コンテキストエラー
12452		・非タスクコンテキストからの呼出し【NGKI2432】
12453		・CPUロック状態からの呼出し【NGKI2433】
12454	E_ID	不正ID番号
12455		・cycidが有効範囲外【NGKI2434】
12456	E_NOEXS	オブジェクト未登録
12457		・対象周期ハンドラが未登録〔D〕【NGKI2435】
12458	E_OACV	オブジェクトアクセス違反
12459		・対象周期ハンドラに対する通常操作1が許可されていない〔P〕
12460		【NGKI2436】

12461 12462 【機能】

12463
12464 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
12465 振舞いは以下の通り．

12466
12467 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
12468 ている状態となる【NGKI2437】．次に周期ハンドラを起動する時刻は，
12469 sta_cycを呼び出して以降の最初の起動時刻に設定される【NGKI2438】．

12470
12471 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
12472 時刻の再設定のみが行われる【NGKI2439】．

12473 12474 【補足説明】

12475
12476 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，
12477 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
12478 に設定される．

12479
12480 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
12481 ドラを起動する時刻は変化しない．

12482 12483 【μITRON4.0仕様との関係】

12484
12485 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
12486 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
12487 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
12488 仕様では，起動位相で指定した相対時間後とした．

12489 -----
12490 msta_cyc 割付けプロセッサ指定での周期ハンドラの動作開始〔TM〕【NGKI2440】

12491 12492 【C言語API】

12493 ER ercd = msta_cyc(ID cycid, ID prcid)

12494 12495 【パラメータ】

12496	ID	cycid	対象周期ハンドラのID番号
12497	ID	prcid	周期ハンドラの割付け対象のプロセッサのID番号

12498 12499 【リターンパラメータ】

12500	ER	ercd	正常終了（E_OK）またはエラーコード
-------	----	------	---------------------

12501

12502 **【エラーコード】**

12503 E_CTX コンテキストエラー

12504 ・非タスクコンテキストからの呼出し【NGKI2441】

12505 ・CPUロック状態からの呼出し【NGKI2442】

12506 E_NOSPT 未サポート機能

12507 ・条件については機能の項を参照

12508 E_ID 不正ID番号

12509 ・cycidが有効範囲外【NGKI2443】

12510 ・prcidが有効範囲外【NGKI2444】

12511 E_PAR パラメータエラー

12512 ・条件については機能の項を参照

12513 E_NOEXS オブジェクト未登録

12514 ・対象周期ハンドラが未登録〔D〕【NGKI2445】

12515 E_OACV オブジェクトアクセス違反

12516 ・対象周期ハンドラに対する通常操作1が許可されていない〔P〕

12517 【NGKI2446】

12518

12519 **【機能】**

12520

12521 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

12522 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。

12523

12524 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

12525 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

12526 動作している状態となる【NGKI2447】。次に周期ハンドラを起動する時刻は、

12527 msta_cycを呼び出して以降の最初の起動時刻に設定される【NGKI2448】。

12528

12529 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

12530 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

12531 動する時刻の再設定が行われる【NGKI2449】。

12532

12533 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、

12534 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【NGKI2450】。対

12535 象周期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

12536 る時からである【NGKI2451】。

12537

12538 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

12539 プロセッサを含んでいない場合には、E_PARエラーとなる【NGKI2452】。

12540

12541 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

12542 を、それが属するクラスの初期割付けプロセッサとする【NGKI2453】。

12543

12544 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

12545 【NGKI2454】。

12546

12547 **【補足説明】**

12548

12549 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、

12550 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

12551 後に設定される.

12552

12553 **【使用上の注意】**

12554

12555 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
12556 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
12557 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。

12558

12559 **【 μ ITRON4.0仕様との関係】**

12560

12561 μ ITRON4.0仕様に定義されていないサービスコールである。

12562

12563 stp_cyc 周期ハンドラの動作停止 [T] **【NGKI2455】**

12564

12565 **【C言語API】**

12566 ER ercd = stp_cyc(ID cycid)

12567

12568 **【パラメータ】**

12569 ID cycid 対象周期ハンドラのID番号

12570

12571 **【リターンパラメータ】**

12572 ER ercd 正常終了 (E_OK) またはエラーコード

12573

12574 **【エラーコード】**

12575 E_CTX コンテキストエラー

12576 ・非タスクコンテキストからの呼出し **【NGKI2456】**

12577 ・CPUロック状態からの呼出し **【NGKI2457】**

12578 E_ID 不正ID番号

12579 ・cycidが有効範囲外 **【NGKI2458】**

12580 E_NOEXS オブジェクト未登録

12581 ・対象周期ハンドラが未登録 [D] **【NGKI2459】**

12582 E_OACV オブジェクトアクセス違反

12583 ・対象周期ハンドラに対する通常操作2が許可されていない [P]

12584 **【NGKI2460】**

12585

12586 **【機能】**

12587

12588 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
12589 振舞いは以下の通り。

12590

12591 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

12592 **【NGKI2461】**。対象周期ハンドラが動作していない状態であれば、何も行われ

12593 ずに正常終了する **【NGKI2462】**。

12594

12595 ref_cyc 周期ハンドラの状態参照 [T] **【NGKI2463】**

12596

12597 **【C言語API】**

12598 ER ercd = ref_cyc(ID cycid, T_RCYC *pk_rcyc)

12599

12600 **【パラメータ】**

12601	ID	cycid	対象周期ハンドラのID番号
12602	T_RCYC *	pk_rcyc	周期ハンドラの現在状態を入れるパケットへのポインタ
12603			
12604			
12605	【リターンパラメータ】		
12606	ER	ercd	正常終了 (E_OK) またはエラーコード
12607			
12608	* 周期ハンドラの現在状態 (パケットの内容)		
12609	STAT	cycstat	周期ハンドラの動作状態
12610	RELTIM	lefttim	次に周期ハンドラを起動する時刻までの相対時間
12611	ID	prcid	周期ハンドラの割付けプロセッサのID (マルチプロセッサ対応カーネルの場合)
12612			
12613			
12614	【エラーコード】		
12615	E_CTX	コンテキストエラー	
12616		・ 非タスクコンテキストからの呼出し 【NGKI2464】	
12617		・ CPUロック状態からの呼出し 【NGKI2465】	
12618	E_ID	不正ID番号	
12619		・ cycidが有効範囲外 【NGKI2466】	
12620	E_NOEXS	オブジェクト未登録	
12621		・ 対象周期ハンドラが未登録 [D] 【NGKI2467】	
12622	E_OACV	オブジェクトアクセス違反	
12623		・ 対象周期ハンドラに対する参照操作が許可されていない [P] 【NGKI2468】	
12624			
12625	E_MACV	メモリアクセス違反	
12626		・ pk_rcycが指すメモリ領域への書込みアクセスが許可されていない [P] 【NGKI2469】	
12627			
12628			
12629	【機能】		
12630			
12631	cycidで指定した周期ハンドラ (対象周期ハンドラ) の現在状態を参照する. 参照した現在状態は, pk_rcycで指定したパケットに返される 【NGKI2470】 .		
12632			
12633			
12634	cycstatには, 対象周期ハンドラの現在の動作状態を表す次のいずれかの値が返される 【NGKI2471】 .		
12635			
12636			
12637	TCYC_STP	0x01U	周期ハンドラが動作していない状態
12638	TCYC_STA	0x02U	周期ハンドラが動作している状態
12639			
12640	対象周期ハンドラが動作している状態である場合には, lefttimに, 次に周期ハンドラ起動する時刻までの相対時間が返される 【NGKI2472】 . 対象周期ハンドラが動作していない状態である場合には, lefttimの値は保証されない 【NGKI2473】 .		
12641			
12642			
12643			
12644			
12645	マルチプロセッサ対応カーネルでは, prcidに, 対象周期ハンドラの割付けプロセッサのID番号が返される 【NGKI2474】 .		
12646			
12647			
12648	【使用上の注意】		
12649			
12650	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		

12651 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
12652 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
12653 状態が変化している可能性があるためである。

12654
12655 【μ ITRON4.0仕様との関係】

12656
12657 TCYC_STPとTCYC_STAを値を変更した。

12658 -----

12659 4.6.3 アラームハンドラ

12660
12661
12662 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
12663 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
12664 別する【NGKI2475】。

12665
12666 各アラームハンドラが持つ情報は次の通り【NGKI2476】。

- 12667
12668 ・アラームハンドラ属性
12669 ・アラームハンドラの動作状態
12670 ・アラームハンドラを起動する時刻
12671 ・拡張情報
12672 ・アラームハンドラの先頭番地
12673 ・アクセス許可ベクタ（保護機能対応カーネルの場合）
12674 ・属する保護ドメイン（保護機能対応カーネルの場合）
12675 ・属するクラス（マルチプロセッサ対応カーネルの場合）

12676
12677 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
12678 ずれかをとる【NGKI2477】。アラームハンドラを動作している状態にすること
12679 を動作開始、動作していない状態にすることを動作停止という。

12680
12681 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
12682 定される【NGKI2478】。

12683
12684 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
12685 る時刻になると、アラームハンドラの起動処理が行われる【NGKI2479】。具体
12686 的には、まず、アラームハンドラが動作していない状態にされる【NGKI2480】。
12687 その後に、拡張情報をパラメータとして、アラームハンドラが呼び出される
12688 【NGKI2481】。

12689
12690 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
12691 ドメインは、カーネルドメインに限られる【NGKI2482】。

12692
12693 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、
12694 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
12695 みであるクラスにのみ属することができる【NGKI2483】。すなわち、アラーム
12696 ハンドラは、システム時刻管理プロセッサによって実行される。

12697
12698 アラームハンドラ属性に指定できる属性はない【NGKI3423】。そのためアラーム
12699 ハンドラ属性には、TA_NULLを指定しなければならない【NGKI3424】。

12700

12701 C言語によるアラームハンドラの記述形式は次の通り【NGKI2484】.

12702

```
12703     void alarm_handler(intptr_t exinf)
12704     {
12705         アラームハンドラ本体
12706     }
```

12707

12708 exinfには、アラームハンドラの拡張情報が渡される【NGKI2485】.

12709

12710 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.

12711

12712 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
12713 カーネルでは、静的APIによって登録されたアラームハン
12714 ドラの数に一致）【NGKI2486】

12715

12716 【μITRON4.0仕様との関係】

12717

12718 TNUM_ALMIDは、μITRON4.0仕様に規定されていないカーネル構成マクロである.

12719 -----

12720 CRE_ALM アラームハンドラの生成 [S] 【NGKI2487】

12721 acre_alm アラームハンドラの生成 [TD] 【NGKI2488】

12722

12723 【静的API】

12724 CRE_ALM(ID almid, { ATR almatr, intptr_t exinf, ALMHDR almhdr })

12725

12726 【C言語API】

12727 ER_ID almid = acre_alm(const T_CALM *pk_calm)

12728

12729 【パラメータ】

12730 ID almid 生成するアラームハンドラのID番号（CRE_ALM
12731 の場合）

12732 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
12733 へのポインタ（静的APIを除く）

12734

12735 *アラームハンドラの生成情報（パケットの内容）

12736 ATR almatr アラームハンドラ属性

12737 intptr_t exinf アラームハンドラの拡張情報

12738 ALMHDR almhdr アラームハンドラの先頭番地

12739

12740 【リターンパラメータ】

12741 ER_ID almid 生成されたアラームハンドラのID番号（正の値）
12742 またはエラーコード

12743

12744 【エラーコード】

12745 E_CTX コンテキストエラー

12746 ・非タスクコンテキストからの呼出し [s] 【NGKI2489】

12747 ・CPUロック状態からの呼出し [s] 【NGKI2490】

12748 E_RSATR 予約属性

12749 ・almatrが無効【NGKI2491】

12750 ・属する保護ドメインの指定が有効範囲外またはカーネルド

12751 メイン以外 [sP] 【NGKI2492】
12752 ・カーネルドメインの囲みの中に記述されていない [SP]
12753 【NGKI2493】
12754 ・属するクラスの指定が有効範囲外 [sM] 【NGKI2494】
12755 ・クラスの囲みの中に記述されていない [SM] 【NGKI2495】
12756 ・その他の条件については機能の項を参照
12757 E_PAR パラメータエラー
12758 ・almhdrがプログラムの先頭番地として正しくない 【NGKI2496】
12759 E_OACV オブジェクトアクセス違反
12760 ・システム状態に対する管理操作が許可されていない [sP]
12761 【NGKI2497】
12762 E_MACV メモリアクセス違反
12763 ・pk_calmが指すメモリ領域への読出しアクセスが許可されて
12764 いない [sP] 【NGKI2498】
12765 E_NOID ID番号不足
12766 ・割り付けられるアラームハンドラIDがない [sD] 【NGKI2499】
12767 E_OBJ オブジェクト状態エラー
12768 ・almidで指定したアラームハンドラが登録済み (CRE_ALMの
12769 場合) 【NGKI2500】
12770
12771 【機能】
12772
12773 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
12774 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
12775 【NGKI2501】。
12776
12777 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
12778 タ、exinfとalmhdrは一般定数式パラメータである 【NGKI2502】。
12779
12780 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、
12781 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
12782 ム時刻管理プロセッサのみでない場合には、E_RSATRエラーとなる 【NGKI2503】。
12783
12784 【TOPPERS/ASPカーネルにおける規定】
12785
12786 ASPカーネルでは、CRE_ALMのみをサポートする 【ASPS0179】。ただし、動的生
12787 成機能拡張パッケージでは、acre_almもサポートする 【ASPS0180】。
12788
12789 【TOPPERS/FMPカーネルにおける規定】
12790
12791 FMPカーネルでは、CRE_ALMのみをサポートする 【FMPS0152】。
12792
12793 【TOPPERS/HRP2カーネルにおける規定】
12794
12795 HRP2カーネルでは、CRE_ALMのみをサポートする 【HRPS0146】。ただし、動的生
12796 成機能拡張パッケージでは、acre_almもサポートする 【HRPS0205】。
12797
12798 【 μ ITRON4.0仕様との関係】
12799
12800 almhdrのデータ型をALMHDRに変更した。


```

12801 -----
12802 AID_ALM      割付け可能なアラームハンドラIDの数の指定 [SD] 【NGKI2504】
12803
12804 【静的API】
12805     AID_ALM(uint_t noalm)
12806
12807 【パラメータ】
12808     uint_t      noalm      割付け可能なアラームハンドラIDの数
12809
12810 【エラーコード】
12811     E_RSATR      予約属性
12812                  ・保護ドメインの囲みの中に記述されている [P] 【NGKI3438】
12813                  ・クラスの囲みの中に記述されていない [M] 【NGKI2506】
12814                  ・その他の条件については機能の項を参照
12815     E_PAR        パラメータエラー
12816                  ・noalmが負の値 【NGKI3286】
12817
12818 【機能】
12819
12820 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
12821 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
12822 【NGKI2507】．
12823
12824 noalmは整数定数式パラメータである 【NGKI2508】．
12825
12826 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，
12827 AID_ALMが属するクラスの割付け可能プロセッサが，システム時刻管理プロセッ
12828 サのみでない場合には，E_RSATRエラーとなる 【NGKI2509】．
12829
12830 【TOPPERS/ASPカーネルにおける規定】
12831
12832 ASPカーネルの動的生成機能拡張パッケージでは，AID_ALMをサポートする
12833 【ASPS0218】．
12834
12835 【TOPPERS/HRP2カーネルにおける規定】
12836
12837 HRP2カーネルの動的生成機能拡張パッケージでは，AID_ALMをサポートする
12838 【HRPS0219】．
12839 -----
12840 SAC_ALM      アラームハンドラのアクセス許可ベクタの設定 [SP] 【NGKI2510】
12841 sac_alm      アラームハンドラのアクセス許可ベクタの設定 [TPD] 【NGKI2511】
12842
12843 【静的API】
12844     SAC_ALM(ID almid, { ACPTN acptn1, ACPTN acptn2,
12845                        ACPTN acptn3, ACPTN acptn4 })
12846
12847 【C言語API】
12848     ER ercd = sac_alm(ID almid, const ACVCT *p_acvct)
12849
12850 【パラメータ】

```

12851	ID	almid	対象アラームハンドラのID番号
12852	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）
12853			
12854			
12855	* アクセス許可ベクタ（パケットの内容）		
12856	ACPTN	acptn1	通常操作1のアクセス許可パターン
12857	ACPTN	acptn2	通常操作2のアクセス許可パターン
12858	ACPTN	acptn3	管理操作のアクセス許可パターン
12859	ACPTN	acptn4	参照操作のアクセス許可パターン
12860			
12861	【リターンパラメータ】		
12862	ER	ercd	正常終了（E_OK）またはエラーコード
12863			
12864	【エラーコード】		
12865	E_CTX	コンテキストエラー	
12866		・ 非タスクコンテキストからの呼出し [s] 【NGKI2512】	
12867		・ CPUロック状態からの呼出し [s] 【NGKI2513】	
12868	E_ID	不正ID番号	
12869		・ almidが有効範囲外 [s] 【NGKI2514】	
12870	E_RSATR	予約属性	
12871		・ カーネルドメインの囲みの中に記述されていない [S] 【NGKI2515】	
12872		・ 対象アラームハンドラが属するクラスの囲みの中に記述されていない [SM] 【NGKI2516】	
12873			
12874	E_NOEXS	オブジェクト未登録	
12875		・ 対象アラームハンドラが未登録 【NGKI2517】	
12876	E_OACV	オブジェクトアクセス違反	
12877		・ 対象アラームハンドラに対する管理操作が許可されていない [s] 【NGKI2518】	
12878			
12879	E_MACV	メモリアクセス違反	
12880		・ p_acvctが指すメモリ領域への読出しアクセスが許可されていない [s] 【NGKI2519】	
12881			
12882	E_OBJ	オブジェクト状態エラー	
12883		・ 対象アラームハンドラは静的APIで生成された [s] 【NGKI2520】	
12884		・ 対象アラームハンドラに対してアクセス許可ベクタが設定済み [S] 【NGKI2521】	
12885			
12886			
12887	【機能】		
12888			
12889	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【NGKI2522】。		
12890			
12891			
12892			
12893	静的APIにおいては、almidはオブジェクト識別名、acptn1～acptn4は整数定数式パラメータである【NGKI2523】。		
12894			
12895			
12896	【TOPPERS/HRP2カーネルにおける規定】		
12897			
12898	HRP2カーネルでは、SAC_ALMのみをサポートする【HRPS0147】。ただし、動的生成機能拡張パッケージでは、sac_almもサポートする【HRPS0206】。		
12899			
12900	-----		

12901 del_alm アラームハンドラの削除 [TD] 【NGKI2524】

12902

12903 【C言語API】

12904 ER ercd = del_alm(ID almid)

12905

12906 【パラメータ】

12907 ID almid 対象アラームハンドラのID番号

12908

12909 【リターンパラメータ】

12910 ER ercd 正常終了 (E_OK) またはエラーコード

12911

12912 【エラーコード】

12913 E_CTX コンテキストエラー

12914 ・非タスクコンテキストからの呼出し 【NGKI2525】

12915 ・CPUロック状態からの呼出し 【NGKI2526】

12916 E_ID 不正ID番号

12917 ・almidが有効範囲外 【NGKI2527】

12918 E_NOEXS オブジェクト未登録

12919 ・対象アラームハンドラが未登録 【NGKI2528】

12920 E_OACV オブジェクトアクセス違反

12921 ・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

12922 ない [P] 【NGKI2529】

12923 E_OBJ オブジェクト状態エラー

12924 ・対象アラームハンドラは静的APIで生成された 【NGKI2530】

12925

12926 【機能】

12927

12928 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

12929 的な振舞いは以下の通り．

12930

12931 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

12932 状態に戻る 【NGKI2531】． 対象アラームハンドラが動作している状態であっ

12933 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

12934 【NGKI2532】．

12935

12936 【TOPPERS/ASPカーネルにおける規定】

12937

12938 ASPカーネルでは，del_almをサポートしない 【ASPS0182】． ただし，動的生成

12939 機能拡張パッケージでは，del_almをサポートする 【ASPS0183】．

12940

12941 【TOPPERS/FMPカーネルにおける規定】

12942

12943 FMPカーネルでは，del_almをサポートしない 【FMPS0154】．

12944

12945 【TOPPERS/HRP2カーネルにおける規定】

12946

12947 HRP2カーネルでは，del_almをサポートしない 【HRPS0148】． ただし，動的生成

12948 機能拡張パッケージでは，del_almをサポートする 【HRPS0207】．

12949 -----

12950 sta_alm アラームハンドラの動作開始 [T] 【NGKI2533】

```

12951 ista_alm    アラームハンドラの動作開始 [I]  【NGKI2534】
12952
12953 【C言語API】
12954     ER ercd = sta_alm(ID almid, RELTIM almtim)
12955     ER ercd = ista_alm(ID almid, RELTIM almtim)
12956
12957 【パラメータ】
12958     ID          almid          対象アラームハンドラのID番号
12959     RELTIM      almtim        アラームハンドラの起動時刻（相対時間）
12960
12961 【リターンパラメータ】
12962     ER          ercd          正常終了（E_OK）またはエラーコード
12963
12964 【エラーコード】
12965     E_CTX      コンテキストエラー
12966               ・非タスクコンテキストからの呼出し（sta_almの場合） 【NGKI2535】
12967               ・タスクコンテキストからの呼出し（ista_almの場合） 【NGKI2536】
12968               ・CPUロック状態からの呼出し
12969     E_ID       不正ID番号
12970               ・almidが有効範囲外 【NGKI2537】
12971     E_PAR      パラメータエラー
12972               ・almtimがTMAX_RELTIMより大きい 【NGKI2538】
12973     E_NOEXS    オブジェクト未登録
12974               ・対象アラームハンドラが未登録 [D]  【NGKI2539】
12975     E_OACV     オブジェクトアクセス違反
12976               ・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
12977                 ない（sta_almの場合） [P]  【NGKI2540】
12978
12979 【機能】
12980
12981 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．
12982 具体的な振舞いは以下の通り．
12983
12984 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
12985 は動作している状態となる 【NGKI2541】． アラームハンドラを起動する時刻は，
12986 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
12987 【NGKI2542】．
12988
12989 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
12990 する時刻の再設定のみが行われる 【NGKI2543】．
12991 -----
12992 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]  【NGKI2544】
12993 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]  【NGKI2545】
12994
12995 【C言語API】
12996     ER ercd = msta_alm(ID almid, RELTIM almtim, ID pccid)
12997     ER ercd = imsta_alm(ID almid, RELTIM almtim, ID pccid)
12998
12999 【パラメータ】
13000     ID          almid          対象アラームハンドラのID番号

```

13001	RELTIM	almtim	アラームハンドラの起動時刻（相対時間）
13002	ID	prcid	アラームハンドラの割付け対象のプロセッサの
13003			ID番号
13004			
13005	【リターンパラメータ】		
13006	ER	ercd	正常終了（E_OK）またはエラーコード
13007			
13008	【エラーコード】		
13009	E_CTX	コンテキストエラー	
13010		・非タスクコンテキストからの呼出し（msta_almの場合）	
13011		【NGKI2546】	
13012		・タスクコンテキストからの呼出し（imsta_almの場合）【NGKI2547】	
13013		・CPUロック状態からの呼出し【NGKI2548】	
13014	E_NOSPT	未サポート機能	
13015		・条件については機能の項を参照	
13016	E_ID	不正ID番号	
13017		・almidが有効範囲外【NGKI2549】	
13018		・prcidが有効範囲外【NGKI2550】	
13019	E_PAR	パラメータエラー	
13020		・almtimがTMAX_RELTIMより大きい【NGKI2551】	
13021		・その他の条件については機能の項を参照	
13022	E_NOEXS	オブジェクト未登録	
13023		・対象アラームハンドラが未登録 [D] 【NGKI2552】	
13024	E_OACV	オブジェクトアクセス違反	
13025		・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） [P] 【NGKI2553】	
13026			
13027			
13028	【機能】		
13029			
13030	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。		
13031			
13032			
13033			
13034	対象アラームハンドラが動作していない状態であれば、対象アラームハンドラの割付けプロセッサがprcidで指定したプロセッサに変更された後、対象アラームハンドラは動作している状態となる【NGKI2554】。アラームハンドラを起動する時刻は、msta_almを呼び出してから、almtimで指定した相対時間後に設定される【NGKI2555】。		
13035			
13036			
13037			
13038			
13039			
13040	対象アラームハンドラが動作している状態であれば、対象アラームハンドラの割付けプロセッサがprcidで指定したプロセッサに変更された後、アラームハンドラを起動する時刻の再設定が行われる【NGKI2556】。		
13041			
13042			
13043			
13044	対象アラームハンドラが実行中である場合には、割付けプロセッサを変更しても、実行中のアラームハンドラを実行するプロセッサは変更されない【NGKI2557】。対象アラームハンドラが変更後の割付けプロセッサで実行されるのは、次に起動される時からである【NGKI2558】。		
13045			
13046			
13047			
13048			
13049	対象アラームハンドラの属するクラスの割付け可能プロセッサが、prcidで指定したプロセッサを含んでいない場合には、E_PARエラーとなる【NGKI2559】。		
13050			

13051
13052 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
13053 サを、それが属するクラスの初期割付けプロセッサとする【NGKI2560】。
13054
13055 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
13056 す【NGKI2561】。
13057
13058 【使用上の注意】
13059
13060 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
13061 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
13062 がある。特に、almtimに0を指定する場合に、注意が必要である。
13063
13064 【μITRON4.0仕様との関係】
13065
13066 μITRON4.0仕様に定義されていないサービスコールである。
13067 -----
13068 stp_alm アラームハンドラの動作停止 [T] 【NGKI2562】
13069 istp_alm アラームハンドラの動作停止 [I] 【NGKI2563】
13070
13071 【C言語API】
13072 ER ercd = stp_alm(ID almid)
13073 ER ercd = istp_alm(ID almid)
13074
13075 【パラメータ】
13076 ID almid 対象アラームハンドラのID番号
13077
13078 【リターンパラメータ】
13079 ER ercd 正常終了 (E_OK) またはエラーコード
13080
13081 【エラーコード】
13082 E_CTX コンテキストエラー
13083 ・非タスクコンテキストからの呼出し (stp_almの場合) 【NGKI2564】
13084 ・タスクコンテキストからの呼出し (istp_almの場合) 【NGKI2565】
13085 ・CPUロック状態からの呼出し【NGKI2566】
13086 E_ID 不正ID番号
13087 ・almidが有効範囲外【NGKI2567】
13088 E_NOEXS オブジェクト未登録
13089 ・対象アラームハンドラが未登録 [D] 【NGKI2568】
13090 E_OACV オブジェクトアクセス違反
13091 ・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
13092 い (stp_almの場合) [P] 【NGKI2569】
13093
13094 【機能】
13095
13096 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。
13097 具体的な振舞いは以下の通り。
13098
13099 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
13100 る【NGKI2570】。対象アラームハンドラが動作していない状態であれば、何も

13101 行われずに正常終了する【NGKI2571】。

13102 -----

13103 ref_alm アラームハンドラの状態参照 [T] 【NGKI2572】

13104

13105 【C言語API】

13106 ER ercd = ref_alm(ID almid, T_RALM *pk_ralm)

13107

13108 【パラメータ】

13109 ID almid 対象アラームハンドラのID番号

13110 T_RALM * pk_ralm アラームハンドラの現在状態を入れるパケット

13111 へのポインタ

13112

13113 【リターンパラメータ】

13114 ER ercd 正常終了 (E_OK) またはエラーコード

13115

13116 *アラームハンドラの現在状態 (パケットの内容)

13117 STAT almstat アラームハンドラの動作状態

13118 RELTIM lefttim アラームハンドラを起動する時刻までの相対時間

13119 ID prcid アラームハンドラの割付けプロセッサのID (マル

13120 チプロセッサ対応カーネルの場合)

13121

13122 【エラーコード】

13123 E_CTX コンテキストエラー

13124 ・非タスクコンテキストからの呼出し【NGKI2573】

13125 ・CPUロック状態からの呼出し【NGKI2574】

13126 E_ID 不正ID番号

13127 ・almidが有効範囲外【NGKI2575】

13128 E_NOEXS オブジェクト未登録

13129 ・対象アラームハンドラが未登録 [D] 【NGKI2576】

13130 E_OACV オブジェクトアクセス違反

13131 ・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

13132 い [P] 【NGKI2577】

13133 E_MACV メモリアクセス違反

13134 ・pk_ralmが指すメモリ領域への書込みアクセスが許可されて

13135 いない [P] 【NGKI2578】

13136

13137 【機能】

13138

13139 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

13140 する。参照した現在状態は、pk_ralmで指定したパケットに返される【NGKI2579】。

13141

13142 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

13143 が返される【NGKI2580】。

13144

13145 TALM_STP 0x01U アラームハンドラが動作していない状態

13146 TALM_STA 0x02U アラームハンドラが動作している状態

13147

13148 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

13149 ハンドラ起動する時刻までの相対時間が返される【NGKI2581】。対象アラーム

13150 ハンドラが動作していない状態である場合には、lefttimの値は保証されない

13151 【NGKI2582】 .

13152

13153 マルチプロセッサ対応カーネルでは、preidに、対象アラームハンドラの割付け
13154 プロセッサのID番号が返される【NGKI2583】 .

13155

13156 【使用上の注意】

13157

13158 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
13159 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
13160 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
13161 ンドラの状態が変化している可能性があるためである。

13162

13163 【μ ITRON4.0仕様との関係】

13164

13165 TALM_STPとTALM_STAを値を変更した。

13166 -----

13167

13168 4.6.4 オーバランハンドラ

13169

13170 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
13171 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
13172 は、システムで1つのみ登録することができる【NGKI2584】 .

13173

13174 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

13175 【NGKI2585】 .

13176

- 13177 ・ オーバランハンドラの動作状態
- 13178 ・ 残りプロセッサ時間

13179

13180 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
13181 いない状態のいずれかをとる【NGKI2586】 . 残りプロセッサ時間は、オーバ
13182 ランハンドラが動作している状態の時に、タスクが使用できる残りのプロセッサ
13183 時間を表す。

13184

13185 オーバランハンドラの動作状態は、タスクの登録時と、タスクが休止状態に遷
13186 移する時に、動作していない状態に初期化される【NGKI2587】 .

13187

13188 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
13189 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【NGKI2588】 .
13190 残りプロセッサ時間が0になると（これをオーバランと呼ぶ）、オーバランハン
13191 ドラが起動される【NGKI2589】 .

13192

13193 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
13194 チン、それらから呼び出したサービルコール（拡張サービスコールを含む）の
13195 実行時間を含む【NGKI2590】 . 一方、タスクの実行中に起動されたカーネル管
13196 理の割込みハンドラ（割込みサービスルーチン、周期ハンドラ、アラームハン
13197 ドラ、オーバランハンドラの実行時間を含む）とカーネル管理のCPU例外ハン
13198 ドラの実行時間は含まないが、割込みハンドラおよびCPU例外ハンドラの呼出し/
13199 復帰にかかる時間と、それらの入口処理と出口処理の一部の実行時間は含ん
13200 しまう【NGKI2591】 . また、タスクの実行中に起動されたカーネル管理外の割

13201 込みハンドラとカーネル管理外のCPU例外ハンドラの実行時間も含む
13202 【NGKI2592】。

13203

13204 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
13205 秒とする【NGKI2593】。ただし、プロセッサ時間には、OVRTIM型に格納できる
13206 任意の値を指定できるとは限らず、指定できる値にターゲット定義の上限があ
13207 る場合がある【NGKI2594】。プロセッサ時間に指定できる最大値は、構成マク
13208 ロTMAX_OVRTIMに定義されている【NGKI2595】。また、タスクが使用したプロセッ
13209 サ時間の計測精度はターゲットに依存する【NGKI2596】。

13210

13211 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
13212 属する【NGKI2597】。

13213

13214 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
13215 る【NGKI2598】。オーバランハンドラ機能がサポートされている場合には、
13216 TOPPERS_SUPPORT_OVRHDRがマクロ定義される【NGKI2599】。サポートされてい
13217 ない場合にオーバランハンドラ機能のサービスコールを呼び出すと、E_NOSPTエ
13218 ラーが返るか、リンク時にエラーとなる【NGKI2600】。

13219

13220 オーバランハンドラ機能に用いるデータ型は次の通り。

13221

13222 OVRTIM プロセッサ時間（符号無し整数，単位はマイクロ秒，ulong_t
13223 に定義）【NGKI2601】

13224

13225 オーバランハンドラ属性に指定できる属性はない【NGKI2602】。そのためオー
13226 バランハンドラ属性には、TA_NULLを指定しなければならない【NGKI2603】。

13227

13228 C言語によるオーバランハンドラの記述形式は次の通り【NGKI2604】。

13229

13230 void overrun_handler(ID tskid, intptr_t exinf)
13231 {
13232 オーバランハンドラ本体
13233 }

13234

13235 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
13236 張情報が、それぞれ渡される【NGKI2605】。

13237

13238 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。

13239

13240 TMAX_OVRTIM プロセッサ時間に指定できる最大値【NGKI2606】

13241

13242 TOPPERS_SUPPORT_OVRHDR オーバランハンドラ機能がサポートされて
13243 いる【NGKI2607】

13244

13245 【使用上の注意】

13246

13247 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
13248 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。

13249

13250 【TOPPERS/ASPカーネルにおける規定】

13251
13252 ASPカーネルでは、オーバランハンドラをサポートしない【ASPS0184】。ただし、
13253 オーバランハンドラ機能拡張パッケージを用いると、オーバランハンドラ機能
13254 を追加することができる【ASPS0185】。
13255
13256 【TOPPERS/FMPカーネルにおける規定】
13257
13258 FMPカーネルでは、オーバランハンドラをサポートしない【FMPS0155】。
13259
13260 【TOPPERS/HRP2カーネルにおける規定】
13261
13262 HRP2カーネルでは、オーバランハンドラをサポートする【HRPS0149】。
13263
13264 【 μ ITRON4.0仕様との関係】
13265
13266 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
13267 はマイクロ秒と規定した。
13268
13269 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。
13270 -----
13271 DEF_OVR オーバランハンドラの定義 [S] 【NGKI2608】
13272 def_ovr オーバランハンドラの定義 [TD] 【NGKI2609】
13273
13274 【静的API】
13275 DEF_OVR({ ATR ovratr, OVRHDR ovrhdr })
13276
13277 【C言語API】
13278 ER ercd = def_ovr(const T_DOVR *pk_dovr)
13279
13280 【パラメータ】
13281 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
13282 トへのポインタ（静的APIを除く）
13283
13284 * オーバランハンドラの定義情報（パケットの内容）
13285 ATR ovratr オーバランハンドラ属性
13286 OVRHDR ovrhdr オーバランハンドラの先頭番地
13287
13288 【リターンパラメータ】
13289 ER ercd 正常終了 (E_OK) またはエラーコード
13290
13291 【エラーコード】
13292 E_CTX コンテキストエラー
13293 ・非タスクコンテキストからの呼出し [s] 【NGKI2610】
13294 ・CPUロック状態からの呼出し [s] 【NGKI2611】
13295 E_RSATR 予約属性
13296 ・ovratrが無効【NGKI2612】
13297 ・その他の条件については機能の項を参照
13298 E_PAR パラメータエラー
13299 ・ovrhdrがプログラムの先頭番地として正しくない【NGKI2613】
13300 E_OACV オブジェクトアクセス違反

・システム状態に対する管理操作が許可されていない [sP] 【NGKI2614】

E_MACV メモリアクセス違反

・pk_dovrが指すメモリ領域への読出しアクセスが許可されていない [sP] 【NGKI2615】

E_OBJ オブジェクト状態エラー

・条件については機能の項を参照

【機能】

各パラメータで指定したオーバランハンドラ定義情報に従って、オーバランハンドラを定義する【NGKI2616】。ただし、def_ovrにおいてpk_dovrをNULLにした場合には、オーバランハンドラの定義を解除する【NGKI2617】。

静的APIにおいては、ovratrは整数定数式パラメータ、ovrhdrは一般定数式パラメータである【NGKI2618】。

オーバランハンドラを定義する場合（DEF_OVRの場合およびdef_ovrにおいてpk_dovrをNULL以外にした場合）で、すでにオーバランハンドラが定義されている場合には、E_OBJエラーとなる【NGKI2619】。

保護機能対応カーネルにおいて、DEF_OVRは、カーネルドメインの囲みの中に記述しなければならない。そうでない場合には、E_RSATRエラーとなる【NGKI2621】。また、def_ovrでオーバランハンドラを定義する場合には、オーバランハンドラの属する保護ドメインを設定する必要はなく、オーバランハンドラ属性にTA_DOM(domid)を指定した場合にはE_RSATRエラーとなる【NGKI2622】。ただし、TA_DOM(TDOM_SELF)を指定した場合には、指定が無視され、E_RSATRエラーは検出されない【NGKI2623】。

マルチプロセッサ対応カーネルでは、DEF_OVRは、クラスの囲みの外に記述しなければならない。そうでない場合には、E_RSATRエラーとなる【NGKI2625】。また、def_ovrオーバランハンドラを定義する場合には、オーバランハンドラの属するクラスを設定する必要はなく、オーバランハンドラ属性にTA_CLS(clsid)を指定した場合にはE_RSATRエラーとなる【NGKI2626】。ただし、TA_CLS(TCLS_SELF)を指定した場合には、指定が無視され、E_RSATRエラーは検出されない【NGKI2627】。

オーバランハンドラの定義を解除する場合（def_ovrにおいてpk_dovrをNULLにした場合）で、オーバランハンドラが定義されていない場合には、E_OBJエラーとなる【NGKI2628】。

オーバランハンドラの定義を解除すると、オーバランハンドラの動作状態は、すべてのタスクに対して動作していない状態となる【NGKI2629】。

【使用上の注意】

def_ovrによりオーバランハンドラの定義を解除する場合、サービスコールの処理時間およびカーネル内での割込み禁止時間が、タスクの総数に比例して長くなる。特に、タスクの総数が多い場合、カーネル内での割込み禁止時間が長くなるため、注意が必要である。

13351
13352 **【TOPPERS/ASPカーネルにおける規定】**
13353
13354 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
13355 ポートする【ASPS0186】。
13356
13357 **【TOPPERS/HRP2カーネルにおける規定】**
13358
13359 HRP2カーネルでは、DEF_OVRのみをサポートする【HRPS0150】。
13360
13361 **【μITRON4.0仕様との関係】**
13362
13363 ovrhdrのデータ型をOVRHDRに変更した。
13364
13365 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、
13366 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
13367 度定義を解除してから、再度定義する必要がある。
13368 -----
13369 sta_ovr オーバランハンドラの動作開始 [T] 【NGKI2630】
13370 ista_ovr オーバランハンドラの動作開始 [I] 【NGKI2631】
13371
13372 **【C言語API】**
13373 ER ercd = sta_ovr(ID tskid, OVRTIM ovrtime)
13374 ER ercd = ista_ovr(ID tskid, OVRTIM ovrtime)
13375
13376 **【パラメータ】**
13377 ID tskid 対象タスクのID番号
13378 OVRTIM ovrtim 対象タスクの残りプロセッサ時間
13379
13380 **【リターンパラメータ】**
13381 ER ercd 正常終了 (E_OK) またはエラーコード
13382
13383 **【エラーコード】**
13384 E_CTX コンテキストエラー
13385 ・非タスクコンテキストからの呼出し (sta_ovrの場合) 【NGKI2632】
13386 ・タスクコンテキストからの呼出し (ista_ovrの場合) 【NGKI2633】
13387 ・CPUロック状態からの呼出し 【NGKI2634】
13388 E_ID 不正ID番号
13389 ・tskidが有効範囲外 【NGKI2635】
13390 E_NOEXS オブジェクト未登録
13391 ・対象タスクが未登録 [D] 【NGKI2636】
13392 E_OACV オブジェクトアクセス違反
13393 ・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
13394 の場合) [P] 【NGKI2637】
13395 E_PAR パラメータエラー
13396 ・ovrtimが0, またはTMAX_OVRTIMより大きい 【NGKI2643】
13397 E_OBJ オブジェクト状態エラー
13398 ・オーバランハンドラが定義されていない 【NGKI2638】
13399
13400 **【機能】**

13401
13402 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
13403 開始する．具体的な振舞いは以下の通り．
13404
13405 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
13406 り，残りプロセッサ時間は，ovrtimに指定した時間に設定される【NGKI2639】．
13407 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
13408 ロセッサ時間の設定のみが行われる【NGKI2640】．
13409
13410 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
13411 となる【NGKI2641】．
13412
13413 【μITRON4.0仕様との関係】
13414
13415 ista_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．
13416 -----
13417 stp_ovr オーバランハンドラの動作停止 [T] 【NGKI2644】
13418 istp_ovr オーバランハンドラの動作停止 [I] 【NGKI2645】
13419
13420 【C言語API】
13421 ER ercd = stp_ovr(ID tskid)
13422 ER ercd = istp_ovr(ID tskid)
13423
13424 【パラメータ】
13425 ID tskid 対象タスクのID番号
13426
13427 【リターンパラメータ】
13428 ER ercd 正常終了（E_OK）またはエラーコード
13429
13430 【エラーコード】
13431 E_CTX コンテキストエラー
13432 ・非タスクコンテキストからの呼出し（stp_ovrの場合）【NGKI2646】
13433 ・タスクコンテキストからの呼出し（istp_ovrの場合）【NGKI2647】
13434 ・CPUロック状態からの呼出し【NGKI2648】
13435 E_ID 不正ID番号
13436 ・tskidが有効範囲外【NGKI2649】
13437 E_NOEXS オブジェクト未登録
13438 ・対象タスクが未登録 [D] 【NGKI2650】
13439 E_OACV オブジェクトアクセス違反
13440 ・対象タスクに対する通常操作2が許可されていない（stp_ovr
13441 の場合） [P] 【NGKI2651】
13442 E_OBJ オブジェクト状態エラー
13443 ・オーバランハンドラが定義されていない【NGKI2652】
13444
13445 【機能】
13446
13447 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
13448 停止する．具体的な振舞いは以下の通り．
13449
13450 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

13451 なる【NGKI2653】. 対象タスクに対してオーバランハンドラが動作していない
 13452 状態であれば、何も行われずに正常終了する【NGKI2654】.
 13453
 13454 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
 13455 となる【NGKI2655】.
 13456
 13457 【μITRON4.0仕様との関係】
 13458
 13459 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.
 13460 -----
 13461 ref_ovr オーバランハンドラの状態参照 [T] 【NGKI2656】
 13462
 13463 【C言語API】
 13464 ER ercd = ref_ovr(ID tskid, T_ROVR *pk_rovr)
 13465
 13466 【パラメータ】
 13467 ID tskid 対象タスクのID番号
 13468 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
 13469 トへのポインタ
 13470
 13471 【リターンパラメータ】
 13472 ER ercd 正常終了 (E_OK) またはエラーコード
 13473
 13474 *タスクの現在状態 (パケットの内容)
 13475 STAT ovrstat オーバランハンドラの動作状態
 13476 OVRTIM leftotm 残りプロセッサ時間
 13477
 13478 【エラーコード】
 13479 E_CTX コンテキストエラー
 13480 ・非タスクコンテキストからの呼出し【NGKI2657】
 13481 ・CPUロック状態からの呼出し【NGKI2658】
 13482 E_ID 不正ID番号
 13483 ・tskidが有効範囲外【NGKI2659】
 13484 E_NOEXS オブジェクト未登録
 13485 ・対象タスクが未登録 [D] 【NGKI2660】
 13486 E_OACV オブジェクトアクセス違反
 13487 ・対象タスクに対する参照操作が許可されていない [P] 【NGKI2661】
 13488 E_MACV メモリアクセス違反
 13489 ・pk_rovrが指すメモリ領域への書込みアクセスが許可されて
 13490 いない [P] 【NGKI2662】
 13491 E_OBJ オブジェクト状態エラー
 13492 ・オーバランハンドラが定義されていない【NGKI2663】
 13493
 13494 【機能】
 13495
 13496 tskidで指定したタスク (対象タスク) に対するオーバランハンドラの現在状態
 13497 を参照する. 参照した現在状態は、pk_rovrで指定したメモリ領域に返される
 13498 【NGKI2664】.
 13499
 13500 ovrstatには、対象タスクに対するオーバランハンドラの動作状態を表す次のい

13501 ずれかの値が返される【NGKI2665】。

13502

13503 TOVR_STP 0x01U オーバランハンドラが動作していない状態

13504 TOVR_STA 0x02U オーバランハンドラが動作している状態

13505

13506 対象タスクに対してオーバランハンドラが動作している状態の場合には、
13507 leftotmに、オーバランハンドラが起動されるまでの残りプロセッサ時間が返さ
13508 れる【NGKI2666】。オーバランハンドラが起動される直前には、leftotmに0が
13509 返される可能性がある【NGKI2667】。オーバランハンドラが動作していない状
13510 態の場合には、leftotmの値は保証されない【NGKI2668】。

13511

13512 tskidにTSK_SELF (=0) を指定すると、自タスクが対象タスクとなる
13513 【NGKI2669】。

13514

13515 【使用上の注意】

13516

13517 ref_ovrはデバッグ時向けの機能であり、その他の目的に使用することは推奨し
13518 ない。これは、ref_ovrを呼び出し、対象オーバランハンドラの現在状態を参照
13519 した直後に割込みが発生した場合、ref_ovrから戻ってきた時には対象オーバ
13520 ンハンドラの状態が変化している可能性があるためである。

13521

13522 【未決定事項】

13523

13524 マルチプロセッサ対応カーネルにおいて、対象タスクが、自タスクが割付けら
13525 れたプロセッサと異なるプロセッサに割り付けられている場合に、leftotmを参
13526 照できるとするかどうかは、今後の課題である。

13527

13528 【μ ITRON4.0仕様との関係】

13529

13530 TOVR_STPとTOVR_STAを値を変更した。

13531

13532 -----

13533

13534 4.7 システム状態管理機能

13535

13536 システム状態管理機能は、特定のオブジェクトに関連しないシステムの状態を
13537 変更／参照するための機能である。

13538

13539 -----

13539 SAC_SYS システム状態のアクセス許可ベクタの設定〔SP〕【NGKI2670】

13540 sac_sys システム状態のアクセス許可ベクタの設定〔TPD〕【NGKI2671】

13541

13542 【静的API】

13543 SAC_SYS({ ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })

13544

13545 【C言語API】

13546 ER ercd = sac_sys(const ACVCT *p_acvct)

13547

13548 【パラメータ】

13549 ACVCT * p_acvct アクセス許可ベクタを入れたパケットへのポ
13550 インタ（静的APIを除く）

13551

13552 *アクセス許可ベクタ (パケットの内容)

13553 ACPTN acptn1 通常操作1のアクセス許可パターン

13554 ACPTN acptn2 通常操作2のアクセス許可パターン

13555 ACPTN acptn3 管理操作のアクセス許可パターン

13556 ACPTN acptn4 参照操作のアクセス許可パターン

13557

13558 【リターンパラメータ】

13559 ER ercd 正常終了 (E_OK) またはエラーコード

13560

13561 【エラーコード】

13562 E_CTX コンテキストエラー

13563 ・非タスクコンテキストからの呼出し [s] 【NGKI2672】

13564 ・CPUロック状態からの呼出し [s] 【NGKI2673】

13565 E_RSATR 予約属性

13566 ・カーネルドメインの囲みの中に記述されていない [S] 【NGKI2674】

13567 ・クラスの囲みの中に記述されている [SM] 【NGKI2675】

13568 E_OACV オブジェクトアクセス違反

13569 ・カーネルドメイン以外からの呼出し [s] 【NGKI2676】

13570 E_OBJ オブジェクト状態エラー

13571 ・システム状態のアクセス許可ベクタが設定済み [S] 【NGKI2677】

13572

13573 【機能】

13574

13575 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

13576 パラメータで指定した値に設定する【NGKI2678】.

13577

13578 静的APIにおいては、acptn1～acptn4は整数定数式パラメータである【NGKI2679】.

13579

13580 【TOPPERS/HRP2カーネルにおける規定】

13581

13582 HRP2カーネルでは、SAC_SYSのみをサポートする【HRPS0151】.

13583 -----

13584 rot_rdq タスクの優先順位の回転 [T] 【NGKI2680】

13585 irot_rdq タスクの優先順位の回転 [I] 【NGKI2681】

13586

13587 【C言語API】

13588 ER ercd = rot_rdq(PRI tskpri)

13589 ER ercd = irot_rdq(PRI tskpri)

13590

13591 【パラメータ】

13592 PRI tskpri 回転対象の優先度 (対象優先度)

13593

13594 【リターンパラメータ】

13595 ER ercd 正常終了 (E_OK) またはエラーコード

13596

13597 【エラーコード】

13598 E_CTX コンテキストエラー

13599 ・非タスクコンテキストからの呼出し (rot_rdqの場合) 【NGKI2682】

13600 ・タスクコンテキストからの呼出し (irot_rdqの場合) 【NGKI2683】

13601 ・CPUロック状態からの呼出し【NGKI2684】
13602 E_NOSPT 未サポート機能
13603 ・条件については機能の項を参照
13604 E_PAR パラメータエラー
13605 ・tskpriが有効範囲外【NGKI2685】
13606 E_OACV オブジェクトアクセス違反
13607 ・システム状態に対する通常操作1が許可されていない (rot_rdq
13608 の場合) [P] 【NGKI2686】
13609
13610 **【機能】**
13611
13612 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、
13613 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
13614 状態にする【NGKI2687】。対象優先度を持つ実行できる状態のタスクが無いか
13615 1つのみの場合には、何も行われずに正常終了する【NGKI2688】。
13616
13617 マルチプロセッサ対応カーネルにおいては、自タスクと同じプロセッサに割り
13618 付けられているタスクのみを操作対象とする【NGKI3622】。
13619
13620 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
13621 優先度が対象優先度となる【NGKI2689】。
13622
13623 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
13624 が制約タスクの場合には、E_NOSPTエラーとなる【NGKI2690】。
13625
13626 **【TOPPERS/SSPカーネルにおける規定】**
13627
13628 SSPカーネルでは、rot_rdq、irot_rdqをサポートしない【SSPS0131】。
13629 -----
13630 mrot_rdq プロセッサ指定でのタスクの優先順位の回転 [TM] 【NGKI2691】
13631 imrot_rdq プロセッサ指定でのタスクの優先順位の回転 [IM] 【NGKI2692】
13632
13633 **【C言語API】**
13634 ER ercd = mrot_rdq(PRI tskpri, ID prcid)
13635 ER ercd = imrot_rdq(PRI tskpri, ID prcid)
13636
13637 **【パラメータ】**
13638 PRI tskpri 回転対象の優先度 (対象優先度)
13639 ID prcid 優先順位の回転対象とするプロセッサのID番号
13640
13641 **【リターンパラメータ】**
13642 ER ercd 正常終了 (E_OK) またはエラーコード
13643
13644 **【エラーコード】**
13645 E_CTX コンテキストエラー
13646 ・非タスクコンテキストからの呼出し (mrot_rdqの場合)
13647 【NGKI2693】
13648 ・タスクコンテキストからの呼出し (imrot_rdqの場合) 【NGKI2694】
13649 ・CPUロック状態からの呼出し【NGKI2695】
13650 E_NOSPT 未サポート機能

13651 ・条件については機能の項を参照
 13652 E_ID 不正ID番号
 13653 ・prcidが有効範囲外【NGKI2696】
 13654 E_PAR パラメータエラー
 13655 ・tskpriが有効範囲外【NGKI2697】
 13656 E_OACV オブジェクトアクセス違反
 13657 ・システム状態に対する通常操作1が許可されていない (mrot_rdq
 13658 の場合) [P] 【NGKI2698】

13659 【機能】

13661
 13662 prcidで指定したプロセッサに割り付けられており, tskpriで指定した優先度
 13663 (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
 13664 スクを, 同じ優先度のタスクの中で最も優先順位が低い状態にする【NGKI2699】.
 13665 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
 13666 われずに正常終了する【NGKI2700】.

13667
 13668 mrot_rdqにおいて, tskpriにTPRI_SELF (=0) を指定すると, 自タスクのベー
 13669 ス優先度が対象優先度となる【NGKI2701】.

13670
 13671 prcidで指定したプロセッサに割り付けられており, 対象優先度を持つ実行でき
 13672 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,
 13673 E_NOSPTエラーとなる【NGKI2702】.

13674 【TOPPERS/ASPカーネルにおける規定】

13675
 13676 ASPカーネルでは, mrot_rdq, imrot_rdqをサポートしない【ASPS0188】.

13677 【TOPPERS/HRP2カーネルにおける規定】

13678
 13679 HRP2カーネルでは, mrot_rdq, imrot_rdqをサポートしない【HRPS0152】.

13680 【TOPPERS/SSPカーネルにおける規定】

13681
 13682 SSPカーネルでは, mrot_rdq, imrot_rdqをサポートしない【SSPS0132】.

13683 【μ ITRON4.0仕様との関係】

13684
 13685 μ ITRON4.0仕様に定義されていないサービスコールである.

13686
 13687 -----
 13688 get_tid 実行状態のタスクIDの参照 [T] 【NGKI2703】
 13689 iget_tid 実行状態のタスクIDの参照 [I] 【NGKI2704】

13690 【C言語API】

13691 ER ercd = get_tid(ID *p_tskid)
 13692 ER ercd = iget_tid(ID *p_tskid)

13693 【パラメータ】

13694 ID * p_tskid タスクIDを入れるメモリ領域へのポインタ
 13695
 13696
 13697
 13698
 13699
 13700

13701 **【リターンパラメータ】**
13702 ER ercd 正常終了 (E_OK) またはエラーコード
13703 ID tskid タスクID
13704
13705 **【エラーコード】**
13706 E_CTX コンテキストエラー
13707 ・非タスクコンテキストからの呼出し (get_tidの場合) 【NGKI2705】
13708 ・タスクコンテキストからの呼出し (iget_tidの場合) 【NGKI2706】
13709 ・CPUロック状態からの呼出し 【NGKI2707】
13710 E_MACV メモリアクセス違反
13711 ・p_tskidが指すメモリ領域への書込みアクセスが許可されて
13712 いない (get_tidの場合) [P] 【NGKI2708】
13713
13714 **【機能】**
13715
13716 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
13717 たタスクIDは、p_tskidが指すメモリ領域に返される 【NGKI2709】。
13718
13719 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
13720 れる 【NGKI2710】。
13721
13722 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
13723 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
13724 【NGKI2711】。
13725
13726 **【TOPPERS/SSPカーネルにおける規定】**
13727
13728 SSPカーネルでは、get_tidをサポートしない 【SSPS0133】。
13729 -----
13730 get_did 実行状態のタスクが属する保護ドメインIDの参照 [TP] 【NGKI2712】
13731
13732 **【C言語API】**
13733 ER ercd = get_did(ID *p_domid)
13734
13735 **【パラメータ】**
13736 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
13737
13738 **【リターンパラメータ】**
13739 ER ercd 正常終了 (E_OK) またはエラーコード
13740 ID domid 保護ドメインID
13741
13742 **【エラーコード】**
13743 E_CTX コンテキストエラー
13744 ・非タスクコンテキストからの呼出し 【NGKI2713】
13745 ・CPUロック状態からの呼出し 【NGKI2714】
13746 E_MACV メモリアクセス違反
13747 ・p_domidが指すメモリ領域への書込みアクセスが許可されて
13748 いない 【NGKI2715】
13749
13750 **【機能】**

13751
13752 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
13753 照した保護ドメインIDは，p_domidが指すメモリ領域に返される【NGKI2716】．
13754
13755 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
13756 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
13757 ンのID番号を参照する【NGKI2717】．
13758
13759 【TOPPERS/ASPカーネルにおける規定】
13760
13761 ASPカーネルでは，get_didをサポートしない【ASPS0189】．
13762
13763 【TOPPERS/FMPカーネルにおける規定】
13764
13765 FMPカーネルでは，get_didをサポートしない【FMPS0157】．
13766
13767 【TOPPERS/SSPカーネルにおける規定】
13768
13769 SSPカーネルでは，get_didをサポートしない【SSPS0134】．
13770 -----
13771 get_pid 割付けプロセッサのID番号の参照 [TM] 【NGKI2718】
13772 iget_pid 割付けプロセッサのID番号の参照 [IM] 【NGKI2719】
13773
13774 【C言語API】
13775 ER ercd = get_pid(ID *p_prcid)
13776 ER ercd = iget_pid(ID *p_prcid)
13777
13778 【パラメータ】
13779 ID * p_prcid プロセッサIDを入れるメモリ領域へのポインタ
13780
13781 【リターンパラメータ】
13782 ER ercd 正常終了 (E_OK) またはエラーコード
13783 ID prcid プロセッサID
13784
13785 【エラーコード】
13786 E_CTX コンテキストエラー
13787 ・非タスクコンテキストからの呼出し (get_pidの場合) 【NGKI2720】
13788 ・タスクコンテキストからの呼出し (iget_pidの場合) 【NGKI2721】
13789 ・CPUロック状態からの呼出し 【NGKI2722】
13790 E_MACV メモリアクセス違反
13791 ・p_prcidが指すメモリ領域への書込みアクセスが許可されて
13792 いない (get_pidの場合) [P] 【NGKI2723】
13793
13794 【機能】
13795
13796 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．
13797 参照したプロセッサIDは，p_prcidが指すメモリ領域に返される
13798 【NGKI2724】．
13799
13800 【使用上の注意】

13801
13802 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
13803 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
13804 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
13805 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。
13806
13807 **【TOPPERS/ASPカーネルにおける規定】**
13808
13809 ASPカーネルでは、get_pid、iget_pidをサポートしない【ASPS0190】。
13810
13811 **【TOPPERS/HRP2カーネルにおける規定】**
13812
13813 HRP2カーネルでは、get_pid、iget_pidをサポートしない【HRPS0153】。
13814
13815 **【TOPPERS/SSPカーネルにおける規定】**
13816
13817 SSPカーネルでは、get_pid、iget_pidをサポートしない【SSPS0135】。
13818
13819 **【μITRON4.0仕様との関係】**
13820
13821 μITRON4.0仕様に定義されていないサービスコールである。
13822 -----
13823 loc_cpu CPUロック状態への遷移 [T] 【NGKI2725】
13824 iloc_cpu CPUロック状態への遷移 [I] 【NGKI2726】
13825
13826 **【C言語API】**
13827 ER ercd = loc_cpu()
13828 ER ercd = iloc_cpu()
13829
13830 **【パラメータ】**
13831 なし
13832
13833 **【リターンパラメータ】**
13834 ER ercd 正常終了 (E_OK) またはエラーコード
13835
13836 **【エラーコード】**
13837 E_CTX コンテキストエラー
13838 ・非タスクコンテキストからの呼出し (loc_cpuの場合) 【NGKI2727】
13839 ・タスクコンテキストからの呼出し (iloc_cpuの場合) 【NGKI2728】
13840 E_OACV オブジェクトアクセス違反
13841 ・システム状態に対する通常操作2が許可されていない
13842 (loc_cpuの場合) [P] 【NGKI2729】
13843
13844 **【機能】**
13845
13846 CPUロックフラグをセットし、CPUロック状態へ遷移する【NGKI2730】。CPUロッ
13847 ク状態で呼び出した場合には、何も行われずに正常終了する【NGKI2731】。
13848 -----
13849 unl_cpu CPUロック状態の解除 [T] 【NGKI2732】
13850 iunl_cpu CPUロック状態の解除 [I] 【NGKI2733】

13851

13852 **【C言語API】**

13853 ER ercd = unl_cpu()

13854 ER ercd = iunl_cpu()

13855

13856 **【パラメータ】**

13857 なし

13858

13859 **【リターンパラメータ】**

13860 ER ercd 正常終了 (E_OK) またはエラーコード

13861

13862 **【エラーコード】**

13863 E_CTX コンテキストエラー

13864 ・非タスクコンテキストからの呼出し (unl_cpuの場合) **【NGKI2734】**

13865 ・タスクコンテキストからの呼出し (iunl_cpuの場合) **【NGKI2735】**

13866 E_OACV オブジェクトアクセス違反

13867 ・システム状態に対する通常操作2が許可されていない

13868 (unl_cpuの場合) [P] **【NGKI2736】**

13869

13870 **【機能】**

13871

13872 CPUロックフラグをクリアし、CPUロック解除状態へ遷移する **【NGKI2737】** .

13873 CPUロック解除状態で呼び出した場合には、何も行われずに正常終了する

13874 **【NGKI2738】** .

13875

13876 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

13877 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、

13878 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

13879 正常終了する) **【NGKI2739】** .

13880

13881 **【補足説明】**

13882

13883 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

13884 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

13885 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

13886 チンの実行が開始される可能性がある。

13887 -----

13888 dis_dsp ディスパッチの禁止 [T] **【NGKI2740】**

13889

13890 **【C言語API】**

13891 ER ercd = dis_dsp()

13892

13893 **【パラメータ】**

13894 なし

13895

13896 **【リターンパラメータ】**

13897 ER ercd 正常終了 (E_OK) またはエラーコード

13898

13899 **【エラーコード】**

13900 E_CTX コンテキストエラー

13951 【リターンパラメータ】
13952 bool_t state コンテキスト
13953
13954 【機能】
13955
13956 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．
13957
13958 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
13959 ストから呼び出した場合にはfalseが返る【NGKI2753】．
13960 -----
13961 sns_loc CPUロック状態の参照 [TI] 【NGKI2754】
13962
13963 【C言語API】
13964 bool_t state = sns_loc()
13965
13966 【パラメータ】
13967 なし
13968
13969 【リターンパラメータ】
13970 bool_t state CPUロックフラグ
13971
13972 【機能】
13973
13974 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．
13975
13976 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
13977 出した場合にはfalseが返る【NGKI2755】．
13978 -----
13979 sns_dsp ディスパッチ禁止状態の参照 [TI] 【NGKI2756】
13980
13981 【C言語API】
13982 bool_t state = sns_dsp()
13983
13984 【パラメータ】
13985 なし
13986
13987 【リターンパラメータ】
13988 bool_t state ディスパッチ禁止フラグ
13989
13990 【機能】
13991
13992 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．
13993
13994 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
13995 状態で呼び出した場合にはfalseが返る【NGKI2757】．
13996 -----
13997 sns_dpn ディスパッチ保留状態の参照 [TI] 【NGKI2758】
13998
13999 【C言語API】
14000 bool_t state = sns_dpn()

14001
14002 **【パラメータ】**
14003 なし
14004
14005 **【リターンパラメータ】**
14006 bool_t state ディスパッチ保留状態
14007
14008 **【機能】**
14009
14010 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．
14011
14012 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
14013 状態でない状態で呼び出した場合にはfalseが返る【NGKI2759】．
14014 -----
14015 sns_ker カーネル非動作状態の参照 [TI] 【NGKI2760】
14016
14017 **【C言語API】**
14018 bool_t state = sns_ker()
14019
14020 **【パラメータ】**
14021 なし
14022
14023 **【リターンパラメータ】**
14024 bool_t state カーネル非動作状態
14025
14026 **【機能】**
14027
14028 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．
14029
14030 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
14031 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
14032 ネルの動作中に呼び出した場合にはfalseが返る【NGKI2761】．
14033
14034 **【使用方法】**
14035
14036 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
14037 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
14038 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
14039 合の動作は保証されない．
14040
14041 **【使用上の注意】**
14042
14043 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．
14044
14045 **【μ ITRON4.0仕様との関係】**
14046
14047 μ ITRON4.0仕様に定義されていないサービスコールである．
14048 -----
14049 ext_ker カーネルの終了 [TI] 【NGKI2762】
14050

14051 **【C言語API】**
14052 ER ercd = ext_ker()
14053
14054 **【パラメータ】**
14055 なし
14056
14057 **【リターンパラメータ】**
14058 ER ercd エラーコード
14059
14060 **【エラーコード】**
14061 E_SYS システムエラー
14062 ・カーネルの誤動作【NGKI2763】
14063 E_OACV オブジェクトアクセス違反
14064 ・カーネルドメイン以外からの呼出し〔P〕【NGKI2764】
14065
14066 **【機能】**
14067
14068 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」
14069 の節を参照すること．
14070
14071 ext_kerが正常に処理された場合，ext_kerからはリターンしない【NGKI2765】．
14072
14073 **【μITRON4.0仕様との関係】**
14074
14075 μITRON4.0仕様に定義されていないサービスコールである．
14076 -----
14077 ref_sys システムの状態参照〔T〕
14078
14079 **【C言語API】**
14080 ER ercd = ref_sys(T_RSYS *pk_rsys)
14081
14082 ☆未完成
14083
14084 **【TOPPERS/ASPカーネルにおける規定】**
14085
14086 ASPカーネルでは，ref_sysをサポートしない．
14087
14088 **【TOPPERS/FMPカーネルにおける規定】**
14089
14090 FMPカーネルでは，ref_sysをサポートしない．
14091
14092 **【TOPPERS/HRP2カーネルにおける規定】**
14093
14094 HRP2カーネルでは，ref_sysをサポートしない．
14095
14096 **【TOPPERS/SSPカーネルにおける規定】**
14097
14098 SSPカーネルでは，ref_sysをサポートしない．
14099 -----
14100

14101 4.8 メモリオブジェクト管理機能

14102

14103 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
14104 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
14105 をサポートしない。

14106

14107 [メモリアリージョン]

14108

14109 メモリアリージョンは、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
14110 象となる同じ性質を持った連続したメモリ領域である。メモリアリージョンは、
14111 メモリアリージョン名によって識別する【NGKI2766】。

14112

14113 各メモリアリージョンが持つ情報は次の通り【NGKI2767】。

14114

- 14115 ・先頭番地
- 14116 ・サイズ
- 14117 ・メモリアリージョン属性

14118

14119 メモリア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
14120 る場合がある【NGKI2768】。

14121

14122 メモリアリージョン属性には、次の属性を指定することができる【NGKI3256】。

14123

14124 TA_NOWRITE 0x01U 書込みアクセス禁止

14125

14126 ターゲットによっては、ターゲット定義のメモリアリージョン属性を指定できる
14127 場合がある【NGKI2771】。

14128

14129 標準メモリアリージョンとは、ATT_MOD/ATA_MODによって、オブジェクトモジュール
14130 に含まれる標準のセクションが配置されるメモリアリージョンである。標準メ
14131 モリアリージョンには、標準のセクションの中で、書込みアクセスを行わないも
14132 のが配置される標準ROMリージョンと、書込みアクセスを行うものが配置される
14133 標準RAMリージョンが含まれる。

14134

14135 マルチプロセッサ対応カーネルでは、ATT_MOD/ATA_MODがクラスの囲みの外に
14136 記述された場合に適用される共通の標準メモリアリージョンに加えて、クラス毎
14137 の標準メモリアリージョンを定義することができる【NGKI3257】。

14138

14139 標準メモリアリージョン（マルチプロセッサ対応カーネルでは、共通の標準メモ
14140 リアリージョン）は、必ず定義しなければならない。定義しない場合には、コン
14141 フィギュレータがエラーを報告する【NGKI3259】。

14142

14143 [メモリオブジェクト]

14144

14145 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
14146 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
14147 て識別する【NGKI2772】。

14148

14149 各メモリオブジェクトが持つ情報は次の通り【NGKI2773】。

14150

14151 ・先頭番地
14152 ・サイズ
14153 ・メモリオブジェクト属性
14154 ・アクセス許可ベクタ
14155 ・属する保護ドメイン
14156 ・属するクラス（マルチプロセッサ対応カーネルの場合）
14157
14158 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
14159 れる【NGKI2774】．
14160
14161 メモリオブジェクト属性には，次の属性を指定することができる【NGKI2775】．
14162
14163 TA_NOWRITE 0x01U 書込みアクセス禁止
14164 TA_NOREAD 0x02U 読出しアクセス禁止
14165 TA_EXEC 0x04U 実行アクセス許可
14166 TA_MEMINI 0x08U メモリの初期化を行う
14167 TA_MEMPRSV 0x10U メモリの初期化を行わない
14168 TA_SDATA 0x20U ショートデータ領域に配置
14169 TA_UNCACHE 0x40U キャッシュ禁止
14170 TA_IODEV 0x80U 周辺デバイスの領域
14171
14172 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
14173 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
14174 許可ベクタにより書込みアクセスが許可されている場合である【NGKI2776】．
14175 また，読出しアクセスできるのは，メモリオブジェクト属性に読出しアクセス
14176 禁止（TA_NOREAD属性）が指定されておらず，アクセス許可ベクタにより読出し・
14177 実行アクセスが許可されている場合である【NGKI2777】．実行アクセスできる
14178 のは，メモリオブジェクト属性に実行アクセス許可（TA_EXEC属性）が指定され
14179 ており，アクセス許可ベクタにより読出し・実行アクセスが許可されている場
14180 合である【NGKI2778】．
14181
14182 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
14183 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，
14184 TA_NOWRITEを指定しても無視される【NGKI2779】．また，読出しアクセス禁止
14185 が実現できない場合には，TA_NOREADを指定しても無視される【NGKI2780】．実
14186 行アクセス禁止が実現できない場合には，TA_EXECを指定しなくても実行アクセ
14187 ス許可となり，TA_EXECは無視される【NGKI2781】．どのような場合にどの属性
14188 の指定が無視されるかは，ターゲット定義である【NGKI2782】．
14189
14190 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
14191 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
14192 クトであることを示す【NGKI2783】．いずれの属性も指定しない場合，そのメ
14193 モリオブジェクトは，システム初期化時にクリア（言い換えると，0に初期化）
14194 される【NGKI2784】．
14195
14196 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，
14197 標準ROMリージョン（マルチプロセッサ対応カーネルでは，共通の標準ROMリー
14198 ジョン）に配置され，メモリオブジェクトとしては登録されない【NGKI2787】．
14199
14200 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

14201 示す【NGKI2788】. 具体的な扱いはターゲット定義であるが、ショートデータ
14202 領域がサポートされていないターゲットでは、この属性は無視される
14203 【NGKI2789】. また、ターゲットによっては、TA_NOWRITEを指定した場合に、
14204 TA_SDATAが無視される場合がある【NGKI2790】.

14205
14206 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、
14207 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
14208 示す【NGKI2791】. 具体的な扱いはターゲット定義であるが、これらの属性を
14209 指定しても意味がないターゲット（例えば、キャッシュを持たないターゲット
14210 プロセッサでのTA_UNCACHE）では、これらの属性は無視される【NGKI2792】.
14211 逆に、キャッシュ禁止にできないメモリオブジェクトに対してTA_UNCACHEを指
14212 定した場合や、周辺デバイスの領域として扱うことができないメモリオブジェ
14213 クトに対してTA_IODEVを指定した場合には、E_RSATRエラーとなる【NGKI2793】.

14214
14215 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
14216 る場合がある【NGKI2794】. ターゲット定義のメモリオブジェクト属性として、
14217 次の属性を予約している【NGKI2795】.

14218
14219 TA_WTHROUGH ライトスルーキャッシュを用いる
14220
14221 [カーネル構成マクロ]
14222
14223 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.
14224
14225 TOPPERS_SUPPORT_ATT_MOD ATT_MOD/ATA_MODがサポートされている
14226 【NGKI2796】
14227 TOPPERS_SUPPORT_ATT_PMA ATT_PMA/ATA_PMA/att_pmaがサポートさ
14228 れている【NGKI2797】
14229
14230 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
14231 め、サポートされているかを判定するには、TOPPERS_SUPPORT_DYNAMIC_CREと
14232 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
14233 る【NGKI2798】.

14234
14235 【補足説明】
14236
14237 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
14238 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.
14239
14240 【TOPPERS/ASPカーネルにおける規定】
14241
14242 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【ASPS0191】.
14243
14244 【TOPPERS/FMPカーネルにおける規定】
14245
14246 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【FMPS0158】.
14247
14248 【TOPPERS/HRP2カーネルにおける規定】
14249
14250 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【HRPS0154】.

14251

14252 **【TOPPERS/SSPカーネルにおける規定】**

14253

14254 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【SSPS0136】.

14255

14256 **【μITRON4.0/PX仕様との関係】**

14257

14258 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

14259 して廃止した. TA_ROはTA_NOWRITEに改名し, TA_NOREAD, TA_EXEC, TA_MEMINI,

14260 TA_MEMPRSV, TA_IODEVを追加した. また, TA_UNCACHEの値を変更し, ターゲッ

14261 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.

14262

14263 メモリリージョンは, μITRON4.0/PX仕様にはない概念である.

14264

14265 **【仕様決定の理由】**

14266

14267 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

14268 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

14269 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

14270 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

14271 で用意してもよいが, それを使うとアプリケーションのポータビリティが下がる

14272 ため, TA_IODEV属性を用意することにした.

14273 -----

14274 ATT_REG メモリリージョンの登録 [SP] **【NGKI2799】**

14275

14276 **【静的API】**

14277 ATT_REG("メモリリージョン名", { ATR regatr, void *base, SIZE size })

14278

14279 **【パラメータ】**

14280 "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

14281 ATR regatr メモリリージョン属性

14282 void * base 登録するメモリリージョンの先頭番地

14283 SIZE size 登録するメモリリージョンのサイズ (バイト数)

14284

14285 **【エラーコード】**

14286 E_RSATR 予約属性

14287 • regatrが無効 **【NGKI2800】**

14288 • 保護ドメインの囲みの中に記述されている **【NGKI2814】**

14289 • クラスの囲みの中に記述されている [M] **【NGKI3260】**

14290 E_PAR パラメータエラー

14291 • sizeが0以下 **【NGKI2816】**

14292 • その他の条件については機能の項を参照

14293 E_OBJ オブジェクト状態エラー

14294 • 登録済みのメモリリージョンの再登録 **【NGKI2801】**

14295 • その他の条件については機能の項を参照

14296

14297 **【機能】**

14298

14299 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

14300 リージョンを登録する. 具体的な振舞いは以下の通り.

14301
14302 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
14303 【NGKI2802】。登録されるメモリリージョンには、regatrで指定したメモリリー
14304 ジョン属性が設定される【NGKI2803】。
14305
14306 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
14307 ラメータである【NGKI2804】。
14308
14309 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
14310 た時には、E_PARエラーとなる【NGKI2815】。登録しようとしたメモリリージョ
14311 ンが、登録済みのメモリリージョンとメモリ領域が重なる場合には、E_OBJエラー
14312 となる【NGKI2817】。
14313
14314 【μ ITRON4.0/PX仕様との関係】
14315
14316 μ ITRON4.0/PX仕様に定義されていない静的APIである。
14317 -----
14318 DEF_SRG 標準メモリリージョンの定義〔SP〕【NGKI3261】
14319
14320 【静的API】
14321 DEF_SRG(“標準ROMリージョン名”, “標準RAMリージョン名”)
14322
14323 【パラメータ】
14324 “標準ROMリージョン名” 標準ROMリージョンとするメモリリージョンを
14325 指定する文字列
14326 “標準RAMリージョン名” 標準RAMリージョンとするメモリリージョンを
14327 指定する文字列
14328
14329 【エラーコード】
14330 E_RSATR 予約属性
14331 ・保護ドメインの囲みの中に記述されている【NGKI3262】
14332 E_OBJ オブジェクト状態エラー
14333 ・標準メモリリージョンが定義済み【NGKI3263】
14334 ・標準ROMリージョンに指定したメモリリージョンが未登録
14335 【NGKI3264】
14336 ・標準RAMリージョンに指定したメモリリージョンが未登録
14337 【NGKI3272】
14338 ・その他の条件については機能の項を参照
14339
14340 【機能】
14341
14342 各パラメータに従って、標準ROMリージョンと標準RAMリージョンを定義する
14343 【NGKI3265】。
14344
14345 マルチプロセッサ対応カーネルでは、DEF_SRGをクラスの囲みの外に記述すると、
14346 共通の標準ROMリージョンと標準RAMリージョンを定義し、クラスの囲みの中に
14347 記述すると、そのクラスの標準ROMリージョンと標準RAMリージョンを定義する
14348 【NGKI3266】。
14349
14350 標準ROMリージョンは、TA_NOWRITE属性のメモリリージョンでなければならない。

14351 標準ROMリージョンとして指定したメモリリージョンが、TA_NOWRITE属性でない
14352 場合には、E_OBJエラーとなる【NGKI3268】。また、標準RAMリージョンは、
14353 TA_NOWRITE属性でないメモリリージョンでなければならない。標準RAMリージョ
14354 ンとして指定したメモリリージョンが、TA_NOWRITE属性である場合には、
14355 E_OBJエラーとなる【NGKI3270】。

14356
14357 【μ ITRON4.0/PX仕様との関係】

14358
14359 μ ITRON4.0/PX仕様に定義されていない静的APIである。
14360 -----

14361 ATT_SEC セクションの登録 [SP] 【NGKI2818】

14362 ATA_SEC セクションの登録（アクセス許可ベクタ付き） [SP] 【NGKI2819】

14363

14364 【静的API】

14365 ATT_SEC("セクション名", { ATR mematr, "メモリリージョン名" })

14366 ATA_SEC("セクション名", { ATR mematr, "メモリリージョン名" },

14367 { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })

14368

14369 【パラメータ】

14370 "セクション名" 登録するセクションを指定する文字列

14371 ATR mematr メモリオブジェクト属性

14372 "メモリリージョン名" セクションを配置するメモリリージョンを指定

14373 する文字列

14374

14375 *アクセス許可ベクタ（パケットの内容）

14376 ACPTN acptn1 通常操作1のアクセス許可パターン

14377 ACPTN acptn2 通常操作2のアクセス許可パターン

14378 ACPTN acptn3 管理操作のアクセス許可パターン

14379 ACPTN acptn4 参照操作のアクセス許可パターン

14380

14381 【エラーコード】

14382 E_RSATR 予約属性

14383 ・mematrが無効【NGKI2820】

14384 ・その他の条件については機能の項を参照

14385 E_NOSPT 未サポート機能

14386 ・条件については機能の項を参照

14387 E_PAR パラメータエラー

14388 ・条件については機能の項を参照

14389 E_OBJ オブジェクト状態エラー

14390 ・登録済みのセクションの再登録【NGKI2821】

14391 ・指定したメモリリージョンが未登録【NGKI2822】

14392

14393 【機能】

14394

14395 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
14396 する。具体的な振舞いは以下の通り。

14397

14398 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、
14399 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
14400 クトとして登録される【NGKI2823】。登録されるメモリオブジェクトには、

14401 mematrで指定したメモリオブジェクト属性が設定される【NGKI2824】。

14402 ATA_SECの場合には、登録されるメモリオブジェクトのアクセス許可ベクタ（4

14403 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

14404 る【NGKI2825】。

14405

14406 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

14407 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

14408 も指定しなくても、同じ振舞いとなる）【NGKI2826】。また、メモリオブジェ

14409 クト属性のTA_MEMINIとTA_MEMPRSVは無視される（指定しても指定しなくても、

14410 同じ振舞いとなる）【NGKI2786】。

14411

14412 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

14413 場合には、E_RSATRエラーとなる【NGKI2828】。

14414

14415 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

14416 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

14417 には、1つのメモリオブジェクトにまとめて登録される場合がある【NGKI2829】。

14418

14419 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～

14420 acptn4は整数定数式パラメータである【NGKI2830】。

14421

14422 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

14423 や登録できる数に制限がある場合がある【NGKI2831】。この制限に違反した場

14424 合には、E_NOSPTエラーとなる【NGKI2832】。

14425

14426 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、

14427 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

14428 となる【NGKI2834】。

14429

14430 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

14431 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

14432 として指定した場合には、E_PARエラーとなる【NGKI2836】。

14433

14434 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

14435 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

14436 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

14437 る【NGKI2837】。

14438

14439 【μ ITRON4.0/PX仕様との関係】

14440

14441 μ ITRON4.0/PX仕様に定義されていない静的APIである。

14442 -----

14443 LNK_SEC セクションの配置 [SP] 【NGKI2838】

14444

14445 【静的API】

14446 LNK_SEC(“セクション名”, { “メモリリージョン名” })

14447

14448 【パラメータ】

14449 “セクション名” 配置するセクションを指定する文字列

14450 “メモリリージョン名” セクションを配置するメモリリージョンを指定

14451 する文字列

14452

14453 **【エラーコード】**

14454 E_RSATR 予約属性

14455 ・保護ドメインの囲みの中に記述されている【NGKI3685】

14456 ・その他の条件については機能の項を参照

14457 E_PAR パラメータエラー

14458 ・条件については機能の項を参照

14459 E_OBJ オブジェクト状態エラー

14460 ・登録済みのセクションの再登録【NGKI2839】

14461 ・指定したメモリーレンジが未登録【NGKI2840】

14462

14463 **【機能】**

14464

14465 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、

14466 メモリーレンジ名で指定したメモリーレンジに配置する【NGKI2841】。

14467

14468 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

14469 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【NGKI2843】。

14470

14471 マルチプロセッサ対応カーネルにおいて、指定したメモリーレンジがあるク

14472 ラス専用のメモリーレンジの場合で、LNK_SECをクラスの囲みの外に記述する

14473 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

14474 【NGKI2844】。

14475

14476 **【使用上の注意】**

14477

14478 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

14479 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。

14480

14481 **【μ ITRON4.0/PX仕様との関係】**

14482

14483 μ ITRON4.0/PX仕様に定義されていない静的APIである。

14484 -----

14485 ATT_MOD オブジェクトモジュールの登録 [SP] 【NGKI2845】

14486 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]

14487 【NGKI2846】

14488

14489 **【静的API】**

14490 ATT_MOD("オブジェクトモジュール名")

14491 ATA_MOD("オブジェクトモジュール名",

14492 { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })

14493

14494 **【パラメータ】**

14495 "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

14496 定する文字列

14497

14498 * アクセス許可ベクタ（パケットの内容）

14499 ACPTN acptn1 通常操作1のアクセス許可パターン

14500 ACPTN acptn2 通常操作2のアクセス許可パターン

14501	ACPTN	acptn3	管理操作のアクセス許可パターン
14502	ACPTN	acptn4	参照操作のアクセス許可パターン

14503

14504 【エラーコード】

14505	E_RSATR	予約属性
14506		・mematrが無効【NGKI2847】
14507	E_NOSPT	未サポート機能
14508		・条件については機能の項を参照
14509	E_OBJ	オブジェクト状態エラー
14510		・登録済みのオブジェクトモジュールの再登録【NGKI2848】

14511

14512 【機能】

14513

14514 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
14515 ネルに登録する。具体的な振舞いは以下の通り。

14516

14517 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
14518 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
14519 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
14520 オブジェクトとして登録される【NGKI2849】。登録されるメモリオブジェクト
14521 には、ターゲット定義でセクション毎に定まるメモリオブジェクト属性が設定
14522 される【NGKI2850】。ATA_MODの場合には、登録されるメモリオブジェクトのア
14523 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
14524 定した値に設定される【NGKI2851】。

14525

14526 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
14527 に記述することも、クラスの囲みの中に記述することもできる【NGKI2852】。
14528 ATT_MOD／ATA_MODをクラスの囲みの外に記述した場合、標準のセクションは、
14529 共通の標準メモリリージョンに配置される【NGKI2853】。クラスの囲みの中に
14530 記述した場合、そのクラスの標準メモリリージョンが定義されていればそれら
14531 のメモリリージョン、定義されていなければ共通の標準メモリリージョンに配
14532 置される【NGKI2854】。ただし、セクションによっては、ターゲット定義で、
14533 クラスの標準メモリリージョンが定義されている場合でも、共通の標準メモリ
14534 リージョンに配置される場合がある【NGKI3271】。

14535

14536 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
14537 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
14538 には、1つのメモリオブジェクトにまとめて登録される場合がある【NGKI2855】。

14539

14540 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
14541 パラメータである【NGKI2856】。

14542

14543 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
14544 保護ドメインや登録できる数に制限がある場合がある【NGKI2857】。この制限
14545 に違反した場合には、E_NOSPTエラーとなる【NGKI2858】。

14546

14547 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
14548 【NGKI2859】。ATT_MOD／ATA_MODがサポートされている場合には、
14549 TOPPERS_SUPPORT_ATT_MODがマクロ定義される【NGKI2860】。サポートされてい
14550 ない場合にATT_MOD／ATA_MODを使用すると、コンフィギュレータがE_NOSPTエラー

14551 を報告する【NGKI2861】.

14552

14553 **【補足説明】**

14554

14555 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

14556 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

14557 いる必要がある。

14558

14559 **【μ ITRON4.0/PX仕様との関係】**

14560

14561 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

14562 ンと標準RAMリージョンであることを明確化した。

14563 -----

14564 ATT_MEM メモリオブジェクトの登録 [SP] 【NGKI2862】

14565 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] 【NGKI2863】

14566 att_mem メモリオブジェクトの登録 [TPD] 【NGKI2864】

14567

14568 **【静的API】**

14569 ATT_MEM({ ATR mematr, void *base, SIZE size })

14570 ATA_MEM({ ATR mematr, void *base, SIZE size },

14571 { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })

14572

14573 **【C言語API】**

14574 ER ercd = att_mem(const T_AMEM *pk_amem)

14575

14576 **【パラメータ】**

14577 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

14578 トへのポインタ（静的APIを除く）

14579

14580 *メモリオブジェクトの登録情報（パケットの内容）

14581 ATR mematr メモリオブジェクト属性

14582 void * base 登録するメモリ領域の先頭番地

14583 SIZE size 登録するメモリ領域のサイズ（バイト数）

14584

14585 *アクセス許可ベクタ（パケットの内容）

14586 ACPTN acptn1 通常操作1のアクセス許可パターン

14587 ACPTN acptn2 通常操作2のアクセス許可パターン

14588 ACPTN acptn3 管理操作のアクセス許可パターン

14589 ACPTN acptn4 参照操作のアクセス許可パターン

14590

14591 **【リターンパラメータ】**

14592 ER ercd 正常終了（E_OK）またはエラーコード

14593

14594 **【エラーコード】**

14595 E_CTX コンテキストエラー

14596 ・非タスクコンテキストからの呼出し [s] 【NGKI2865】

14597 ・CPUロック状態からの呼出し [s] 【NGKI2866】

14598 E_RSATR 予約属性

14599 ・mematrが無効【NGKI2867】

14600 ・属する保護ドメインの指定が有効範囲外 [sP] 【NGKI2868】

14601 ・属するクラスの指定が有効範囲外 [sM] 【NGKI2869】
14602 ・その他の条件については機能の項を参照
14603 E_NOSPT 未サポート機能
14604 ・条件については機能の項を参照
14605 E_PAR パラメータエラー
14606 ・sizeが0以下 【NGKI2881】
14607 ・その他の条件については機能の項を参照
14608 E_OACV オブジェクトアクセス違反
14609 ・システム状態に対する管理操作が許可されていない [sP]
14610 【NGKI2870】
14611 E_MACV メモリアクセス違反
14612 ・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
14613 いない [sP] 【NGKI2871】
14614 E_OBJ オブジェクト状態エラー
14615 ・条件については機能の項を参照
14616
14617 **【機能】**
14618
14619 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
14620 クトを登録する。具体的な振舞いは以下の通り。
14621
14622 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
14623 【NGKI2872】。登録されるメモリオブジェクトには、mematrで指定したメモリ
14624 オブジェクト属性が設定される【NGKI2873】。ATA_MEMの場合には、登録される
14625 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）
14626 が、acptn1～acptn4で指定した値に設定される【NGKI2874】。
14627
14628 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
14629 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、
14630 E_RSATRエラーとなる【NGKI2876】。また、mematrにTA_SDATAを指定することは
14631 できない。TA_SDATAを指定した場合には、E_RSATRエラーとなる【NGKI3274】。
14632
14633 静的APIにおいては、mematr、size、acptn1～acptn4は整数定数式パラメータ、
14634 baseは一般定数式パラメータである【NGKI2877】。
14635
14636 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
14637 属する保護ドメインや登録できる数に制限がある場合がある【NGKI2878】。こ
14638 の制限に違反した場合には、E_NOSPTエラーとなる【NGKI2879】。
14639
14640 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
14641 た時には、E_PARエラーとなる【NGKI2880】。登録しようとしたメモリオブジェ
14642 クトが、登録済みのメモリオブジェクトとメモリ領域が重なる場合には、
14643 E_OBJエラーとなる【NGKI2882】。
14644
14645 **【使用上の注意】**
14646
14647 ATT_MEM／ATA_MEMは、メモリ空間にマッピングされたI/O領域にアクセスできる
14648 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
14649 は、ATT_SEC／ATA_SECかATT_MOD／ATA_MODを使用することを推奨する。
14650

14651 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
 14652 録したメモリリージョンと重なっても、直ちにエラーとはならない。ただし、
 14653 メモリリージョン内に配置されたメモリオブジェクトと、ATT_MEM/ATA_MEMで
 14654 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
 14655 なる。

14656 **【TOPPERS/HRP2カーネルにおける規定】**

14657

14658

14659 HRP2カーネルでは、ATT_MEMとATA_MEMのみをサポートする【HRPS0155】。

14660

14661 **【μITRON4.0/PX仕様との関係】**

14662

14663 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
 14664 (ata_mem) は廃止した。

14665

14666 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
 14667 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、
 14668 この仕様ではE_PARエラーとなることとした。

14669 -----

14670 ATT_PMA 物理メモリ領域の登録 [SP] 【NGKI2883】

14671 ATA_PMA 物理メモリ領域の登録（アクセス許可ベクタ付き） [SP] 【NGKI2884】

14672 att_pma 物理メモリ領域の登録 [TPD] 【NGKI2885】

14673

14674 **【静的API】**

14675 ATT_PMA({ ATR mematr, void *base, SIZE size, void *paddr })

14676 ATA_PMA({ ATR mematr, void *base, SIZE size, void *paddr },

14677 { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })

14678

14679 **【C言語API】**

14680 ER ercd = att_pma(const T_APMA *pk_apma)

14681

14682 **【パラメータ】**

14683 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
 14684 のポインタ（静的APIを除く）

14685

14686 *物理メモリ領域の登録情報（パケットの内容）

14687 ATR mematr メモリオブジェクト属性

14688 void * base 登録するメモリ領域の先頭番地

14689 SIZE size 登録するメモリ領域のサイズ（バイト数）

14690 void * paddr 登録するメモリ領域の物理アドレス空間における
 14691 先頭番地

14692

14693 *アクセス許可ベクタ（パケットの内容）

14694 ACPTN acptn1 通常操作1のアクセス許可パターン

14695 ACPTN acptn2 通常操作2のアクセス許可パターン

14696 ACPTN acptn3 管理操作のアクセス許可パターン

14697 ACPTN acptn4 参照操作のアクセス許可パターン

14698

14699 **【リターンパラメータ】**

14700 ER ercd 正常終了 (E_OK) またはエラーコード

14701

14702 **【エラーコード】**

14703 E_CTX コンテキストエラー

14704 ・非タスクコンテキストからの呼出し〔s〕 【NGKI2886】

14705 ・CPUロック状態からの呼出し〔s〕 【NGKI2887】

14706 E_RSATR 予約属性

14707 ・mematrが無効

14708 ・属する保護ドメインの指定が有効範囲外〔sP〕 【NGKI2888】

14709 ・属するクラスの指定が有効範囲外〔sM〕 【NGKI2889】

14710 ・その他の条件については機能の項を参照

14711 E_NOSPT 未サポート機能

14712 ・条件については機能の項を参照

14713 E_PAR パラメータエラー

14714 ・sizeが0以下 【NGKI2901】

14715 ・その他の条件については機能の項を参照

14716 E_OACV オブジェクトアクセス違反

14717 ・システム状態に対する管理操作が許可されていない〔sP〕

14718 【NGKI2890】

14719 E_MACV メモリアクセス違反

14720 ・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

14721 いない〔sP〕 【NGKI2891】

14722 E_OBJ オブジェクト状態エラー

14723 ・条件については機能の項を参照

14724

14725 **【機能】**

14726

14727 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
14728 クトを登録する。具体的な振舞いは以下の通り。

14729

14730 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
14731 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
14732 リオブジェクトとして登録される【NGKI2892】。登録されるメモリオブジェク
14733 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【NGKI2893】。
14734 ATA_PMA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
14735 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
14736 る【NGKI2894】。

14737

14738 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
14739 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、
14740 E_RSATRエラーとなる【NGKI2896】。

14741

14742 静的APIにおいては、mematr、size、paddr、acptn1～acptn4は整数定数式パラ
14743 メータ、baseは一般定数式パラメータである【NGKI2897】。

14744

14745 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
14746 属する保護ドメインや登録できる数に制限がある場合がある【NGKI2898】。こ
14747 の制限に違反した場合には、E_NOSPTエラーとなる【NGKI2899】。

14748

14749 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
14750 指定した時には、E_PARエラーとなる【NGKI2900】。登録しようとしたメモリオ

14801 ・対象メモリオブジェクトに対する管理操作が許可されてい
14802 ない【NGKI2912】
14803 E_MACV メモリアクセス違反
14804 ・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
14805 いない【NGKI2913】
14806 E_OBJ オブジェクト状態エラー
14807 ・対象メモリオブジェクトは静的APIで登録された【NGKI2914】
14808

14809 【機能】

14810
14811 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
14812 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
14813 設定する【NGKI2915】。

14814
14815 【TOPPERS/HRP2カーネルにおける規定】

14816
14817 HRP2カーネルでは、sac_memをサポートしない【HRPS0157】。

14818
14819 【μITRON4.0/PX仕様との関係】

14820
14821 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
14822 ることができないこととした。

14823
14824 μITRON4.0/PX仕様では、baseはメモリオブジェクトに含まれる番地を指定する
14825 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
14826 ならないものとした。

14827 -----
14828 det_mem メモリオブジェクトの登録解除 [TPD] 【NGKI2916】

14829
14830 【C言語API】

14831 ER ercd = det_mem(const void *base)

14832
14833 【パラメータ】

14834 void * base メモリオブジェクトの先頭番地

14835
14836 【リターンパラメータ】

14837 ER ercd 正常終了 (E_OK) またはエラーコード

14838
14839 【エラーコード】

14840 E_CTX コンテキストエラー
14841 ・非タスクコンテキストからの呼出し【NGKI2917】
14842 ・CPUロック状態からの呼出し【NGKI2918】
14843 E_PAR パラメータエラー
14844 ・baseがメモリオブジェクトの先頭番地でない【NGKI2919】
14845 E_NOEXS オブジェクト未登録
14846 ・baseで指定した番地を含むメモリオブジェクトが登録され
14847 ていない【NGKI2920】
14848 E_OACV オブジェクトアクセス違反
14849 ・対象メモリオブジェクトに対する管理操作が許可されてい
14850 ない【NGKI2921】

14851 E_OBJ オブジェクト状態エラー
14852 ・対象メモリオブジェクトは静的APIで登録された【NGKI2922】
14853
14854 **【機能】**
14855
14856 baseで指定したメモリオブジェクト（対象メモリオブジェクト）を登録解除する【NGKI2923】。
14857
14858
14859 **【TOPPERS/HRP2カーネルにおける規定】**
14860
14861 HRP2カーネルでは、det_memをサポートしない【HRPS0158】。
14862
14863 **【μ ITRON4.0/PX仕様との関係】**
14864
14865 静的APIによって登録したメモリオブジェクトは、登録を解除することができないこととした。
14866
14867 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ものとしていたが、この仕様では、メモリオブジェクトの先頭番地でなければならないものとした。
14868
14869
14870
14871 -----
14872 prb_mem メモリ領域に対するアクセス権のチェック [TP] 【NGKI2924】
14873
14874 **【C言語API】**
14875 ER ercd = prb_mem(const void *base, SIZE size, ID tskid, MODE pmmode)
14876
14877 **【パラメータ】**
14878 void * base メモリ領域の先頭番地
14879 SIZE size メモリ領域のサイズ（バイト数）
14880 ID tskid アクセス元のタスクのID番号
14881 MODE pmmode アクセスモード
14882
14883 **【リターンパラメータ】**
14884 ER ercd 正常終了（E_OK）またはエラーコード
14885
14886 **【エラーコード】**
14887 E_CTX コンテキストエラー
14888 ・非タスクコンテキストからの呼出し【NGKI2925】
14889 E_ID 不正ID番号
14890 ・tskidが有効範囲外【NGKI2927】
14891 E_PAR パラメータエラー
14892 ・sizeが0【NGKI2929】
14893 ・その他の条件については機能の項を参照
14894 E_NOEXS オブジェクト未登録
14895 ・baseで指定した番地を含むメモリオブジェクトが登録されていない【NGKI2930】
14896 ・tskidで指定したタスクが未登録 [D] 【NGKI3425】
14897 E_OACV オブジェクトアクセス違反
14898 ・対象メモリ領域を含むメモリオブジェクトに対する参照操作が許可されていない【NGKI2931】
14899
14900

14901 • tskidで指定したタスクに対する参照操作が許可されてい
14902 ない【NGKI3426】
14903 E_MACV メモリアクセス違反
14904 • 条件については機能の項を参照
14905 E_OBJ オブジェクト状態エラー
14906 • 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
14907 【NGKI2932】

14908

14909 【機能】

14910

14911 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
14912 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
14913 ックする．アクセスが許可されている場合にE_OK，そうでない場合にE_MACVが返
14914 る【NGKI2933】．tskidで指定したタスクがカーネルドメインに属する場合，
14915 E_MACVが返ることはない【NGKI2934】．

14916

14917 pmmodeには、TPM_WRITE（＝0x01U），TPM_READ（＝0x02U），TPM_EXEC（＝
14918 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
14919 "|"）を指定することができる【NGKI2935】．TPM_WRITE，TPM_READ，TPM_EXEC
14920 を指定した場合には、それぞれ、読出しアクセス、書込みアクセス、実行ア
14921 クセスが許可されているかをチェックする【NGKI2936】．また、いくつかのビ
14922 ット毎論理和を指定した場合には、それらに対応した種別のアクセスがすべて許
14923 可されているかをチェックする【NGKI2937】．pmmodeにそれ以外の値を指定し
14924 た場合には、E_PARエラーとなる【NGKI2938】．

14925

14926 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
14927 アクセスが許可されているかをチェックする【NGKI2939】．

14928

14929 【μITRON4.0/PX仕様との関係】

14930

14931 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
14932 スクによる指定（tskid）に変更した．また、pmmodeに指定できるアクセス種別に
14933 TPM_EXECを追加し、TPM_WRITEとTPM_READの値を入れ換えた．CPUロック状態か
14934 らも呼び出せるものとした．

14935

14936 【仕様決定の理由】

14937

14938 prb_memを、CPUロック状態からも呼び出せるものとしたのは、次の理由による．
14939 prb_memは、拡張サービスコールの中で、タスクから渡されたポインタが、その
14940 タスクからアクセスできる領域であるかを調べるために用いることを想定して
14941 いる．拡張サービスコールの中には、CPUロック状態でも呼び出せるものがあり、
14942 そのような拡張サービスコールを実現するには、prb_memがCPUロック状態から
14943 呼び出せることが必要である．

14944

14945 なお、prb_memを非タスクコンテキストから呼び出すことはできないが、非タ
14946 スクコンテキストで実行される処理単位は必ずカーネルドメインに属するために、
14947 prb_memを使ってアクセス権を調べる必要がないことから、支障がない．

14948

14949 ref_mem メモリオブジェクトの状態参照 [TP]

14950

14951 【C言語API】

14952 ER ercd = ref_mem(const void *base, T_RMEM *pk_rmem)

14953

14954 ☆未完成

14955

14956 【TOPPERS/HRP2カーネルにおける規定】

14957

14958 HRP2カーネルでは、ref_memをサポートしない。

14959

14960

14961 4.9 割込み管理機能

14962

14963 割込み処理のプログラムは、割込みサービスルーチン（ISR）として実現すること
14964 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、
14965 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
14966 おく必要がある【NGKI2940】。割込みサービスルーチンは、カーネル内の割込
14967 みハンドラを経由して呼び出される【NGKI2941】。

14968

14969 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
14970 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

14971 【NGKI2942】。この場合にも、割込みハンドラをカーネルに登録する前に、割
14972 込みハンドラの登録対象となる割込みハンドラ番号に対応する割込み要求ライ
14973 ンの属性を設定しておく必要がある【NGKI2943】。

14974

14975 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、
14976 次の属性を指定することができる【NGKI2944】。

14977

14978 TA_ENAINT 0x01U 割込み要求禁止フラグをクリア

14979 TA_EDGE 0x02U エッジトリガ

14980

14981 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
14982 場合がある【NGKI2945】。ターゲット定義の割込み要求ライン属性として、次
14983 の属性を予約している【NGKI2946】。

14984

14985 TA_POSEDGE ポジティブエッジトリガ

14986 TA_NEGEDGE ネガティブエッジトリガ

14987 TA_BOTHEDGE 両エッジトリガ

14988 TA_LOWLEVEL ローレベルトリガ

14989 TA_HIGHLEVEL ハイレベルトリガ

14990 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
14991 サ対応カーネルの場合）

14992

14993 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
14994 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
14995 する【NGKI2947】。

14996

14997 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、
14998 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
14999 にすべて呼び出される【NGKI2948】。割込みサービスルーチン優先度が同じ場
15000 合には、登録した順（静的APIにより登録した場合には、割込みサービスルーチ

15001 ンを生成するAPIをコンフィギュレーションファイル中に記述した順）で呼び出
 15002 される【NGKI2949】。

15003

15004 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
 15005 る保護ドメインは、カーネルドメインに限られる【NGKI2950】。

15006

15007 割込みサービスルーチン属性に指定できる属性はない【NGKI2951】。そのため
 15008 割込みサービスルーチン属性には、TA_NULLを指定しなければならない
 15009 【NGKI2952】。

15010

15011 C言語による割込みサービスルーチンの記述形式は次の通り【NGKI2953】。

15012

```

15013       void interrupt_service_routine(intptr_t exinf)
15014       {
15015           割込みサービスルーチン本体
15016       }
  
```

15017

15018 exinfには、割込みサービスルーチンの拡張情報が渡される【NGKI2954】。

15019

15020 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
 15021 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
 15022 【NGKI2955】。

15023

15024 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
 15025 る【NGKI2956】。

15026

15027 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
 15028 定義で、次の属性を指定することができる【NGKI2957】。

15029

```

15030       TA_NONKERNEL     0x02U     カーネル管理外の割込み
15031
  
```

15032 TA_NONKERNELを指定しない場合、カーネル管理の割込みとなる【NGKI2958】。
 15033 また、ターゲットによっては、その他のターゲット定義の割込みハンドラ属性
 15034 を指定できる場合がある【NGKI2959】。

15035

15036 C言語による割込みハンドラの記述形式は次の通り【NGKI2960】。

15037

```

15038       void interrupt_handler(void)
15039       {
15040           割込みハンドラ本体
15041       }
  
```

15042

15043 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。

15044

```

15045       TMIN_INTPRI       割込み優先度の最小値（最高値）     【NGKI2961】
15046       TMAX_INTPRI       割込み優先度の最大値（最低値，=-1）
15047
15048       TMIN_ISRPRI       割込みサービスルーチン優先度の最小値（=1）【NGKI2962】
15049       TMAX_ISRPRI       割込みサービスルーチン優先度の最大値
15050
  
```

15051 TOPPERS_SUPPORT_DIS_INT dis_intがサポートされている【NGKI2963】
 15052 TOPPERS_SUPPORT_ENA_INT ena_intがサポートされている【NGKI2964】
 15053
 15054 **【使用上の注意】**
 15055
 15056 1つの割込み要求ラインに複数のデバイスからの割込み要求が接続されている場
 15057 合に対応するために、割込みサービスルーチンは、それが処理する割込み要求
 15058 が発生しているかをチェックし、割込み要求が発生していない場合には何もせ
 15059 ずにリターンするように実装すべきである。
 15060
 15061 **【TOPPERS/ASPカーネルにおける規定】**
 15062
 15063 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）
 15064 は16に固定されている【ASPS0192】。ただし、タスク優先度拡張パッケージで
 15065 は、TMAX_ISRPRIを256に拡張する【ASPS0193】。
 15066
 15067 **【TOPPERS/FMPカーネルにおける規定】**
 15068
 15069 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）
 15070 は16に固定されている【FMPS0159】。
 15071
 15072 **【TOPPERS/HRP2カーネルにおける規定】**
 15073
 15074 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）
 15075 は16に固定されている【HRPS0159】。
 15076
 15077 **【TOPPERS/SSPカーネルにおける規定】**
 15078
 15079 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）
 15080 は16に固定されている【SSPS0137】。
 15081
 15082 **【 μ ITRON4.0仕様との関係】**
 15083
 15084 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、
 15085 μ ITRON4.0仕様がない概念であり、TMIN_INTPRI, TMAX_INTPRI, TMIN_ISRPRI,
 15086 TMAX_ISRPRIは、 μ ITRON4.0仕様に定義のないカーネル構成マクロである。また、
 15087 TA_NONKERNELは、 μ ITRON4.0仕様に定義のない割込みハンドラ属性である。
 15088 -----
 15089 CFG_INT 割込み要求ラインの属性の設定 [S] 【NGKI2965】
 15090 cfg_int 割込み要求ラインの属性の設定 [TD] 【NGKI2966】
 15091
 15092 **【静的API】**
 15093 CFG_INT(INTNO intno, { ATR intatr, PRI intpri })
 15094
 15095 **【C言語API】**
 15096 ER ercd = cfg_int(INTNO intno, const T_CINT *pk_cint)
 15097
 15098 **【パラメータ】**
 15099 INTNO intno 割込み番号
 15100 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

```

15101                                ケットへのポインタ（静的APIを除く）
15102
15103     * 割込み要求ラインの属性の設定情報（パケットの内容）
15104     ATR                intatr    割込み要求ライン属性
15105     PRI                intpri    割込み優先度
15106
15107     【リターンパラメータ】
15108     ER                ercd       正常終了（E_OK）またはエラーコード
15109
15110     【エラーコード】
15111     E_CTX             コンテキストエラー
15112                     ・非タスクコンテキストからの呼出し〔s〕【NGKI2967】
15113                     ・CPUロック状態からの呼出し〔s〕【NGKI2968】
15114     E_RSATR           予約属性
15115                     ・intatrが無効【NGKI2969】
15116                     ・属するクラスの指定が有効範囲外〔sM〕【NGKI2970】
15117                     ・クラスの囲みの中に記述されていない〔SM〕【NGKI2971】
15118                     ・その他の条件については機能の項を参照
15119     E_PAR             パラメータエラー
15120                     ・intnoが有効範囲外【NGKI2972】
15121                     ・intpriが有効範囲外【NGKI2973】
15122                     ・その他の条件については機能の項を参照
15123     E_OACV            オブジェクトアクセス違反
15124                     ・システム状態に対する管理操作が許可されていない〔sP〕
15125                     【NGKI2974】
15126     E_MACV            メモリアクセス違反
15127                     ・pk_cintが指すメモリ領域への読出しアクセスが許可されて
15128                     いない〔sP〕【NGKI2975】
15129     E_OBJ             オブジェクト状態エラー
15130                     ・対象割込み要求ラインに対して属性が設定済み〔S〕【NGKI2976】
15131                     ・その他の条件については機能の項を参照
15132
15133     【機能】
15134
15135     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
15136     メータで指定した属性を設定する【NGKI2977】。
15137
15138     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
15139     た場合にクリアされ、指定しない場合にセットされる【NGKI2978】。
15140
15141     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
15142     【NGKI2979】。
15143
15144     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
15145     が連動して設定される場合がある【NGKI2980】。
15146
15147     intpriに指定できる値は、基本的には、TMIN_INTPRI以上、TMAX_INTPRI以下の
15148     値である【NGKI2981】。ターゲット定義の拡張で、カーネル管理外の割込み要
15149     求ラインに対しても属性を設定できる場合には、TMIN_INTPRIよりも小さい値を
15150     指定することができる【NGKI2982】。このように拡張されている場合、カーネ
  
```

15151 ル管理外の割込み要求ラインを対象として、intpriにTMIN_INTPRI以上の値を指
15152 定した場合には、E_OBJエラーとなる【NGKI2983】。逆に、カーネル管理の割込
15153 み要求ラインを対象として、intpriがTMIN_INTPRIよりも小さい値である場合に
15154 も、E_OBJエラーとなる【NGKI2984】。

15155
15156 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
15157 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
15158 した場合にはE_PARエラーとなる【NGKI2985】。ここで、設定できない割込み要
15159 求ライン属性／割込み優先度には、ターゲット定義の制限によって設定できな
15160 い値も含む【NGKI2986】。また、マルチプロセッサ対応カーネルにおいて、
15161 cfg_intを呼び出したタスクが割り付けられているプロセッサから、対象割込み
15162 要求ラインの属性を設定できない場合も、これに該当する【NGKI2987】。

15163
15164 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
15165 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
15166 【NGKI2989】。また、cfg_intはカーネルオブジェクトを登録するサービスコー
15167 ルではないため、割込み要求ライン属性にTA_DOM(domid)を指定した場合には
15168 E_RSATRエラーとなる【NGKI2990】。ただし、TA_DOM(TDOM_SELF)を指定した場
15169 合には、指定が無視され、E_RSATRエラーは検出されない【NGKI2991】。

15170
15171 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
15172 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
15173 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
15174 み中にある場合には、E_RSATRエラーとなる【NGKI2992】。

15175
15176 【補足説明】
15177

15178 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
15179 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
15180 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、
15181 変更できないケースが考えられる。

15182
15183 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
15184 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
15185 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
15186 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
15187 か設定できない割込み要求ラインに対して異なる割込み優先度を設定した場合
15188 には、E_PARエラーとなる。

15189
15190 【TOPPERS/ASPカーネルにおける規定】
15191

15192 ASPカーネルでは、CFG_INTのみをサポートする【ASPS0194】。

15193
15194 【TOPPERS/FMPカーネルにおける規定】
15195

15196 FMPカーネルでは、CFG_INTのみをサポートする【FMPS0160】。

15197
15198 【TOPPERS/HRP2カーネルにおける規定】
15199

15200 HRP2カーネルでは、CFG_INTのみをサポートする【HRPS0160】。

15251 ・その他の条件については機能の項を参照
15252 E_PAR パラメータエラー
15253 ・intnoが有効範囲外【NGKI3003】
15254 ・isrがプログラムの先頭番地として正しくない【NGKI3004】
15255 ・isrpriが有効範囲外【NGKI3005】
15256 E_OACV オブジェクトアクセス違反
15257 ・システム状態に対する管理操作が許可されていない〔sP〕
15258 【NGKI3006】
15259 E_MACV メモリアクセス違反
15260 ・pk_cisrが指すメモリ領域への読出しアクセスが許可されて
15261 いない〔sP〕【NGKI3007】
15262 E_NOID ID番号不足
15263 ・割り付けられる割込みサービスルーチンIDがない〔sD〕
15264 【NGKI3008】
15265 E_OBJ オブジェクト状態エラー
15266 ・isridで指定した割込みサービスルーチンが登録済み
15267 (CRE_ISRの場合)【NGKI3009】
15268 ・その他の条件については機能の項を参照
15269
15270 【機能】
15271
15272 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
15273 ビスルーチンを生成する【NGKI3010】。
15274
15275 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
15276 【NGKI3011】。
15277
15278 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
15279 エラーとなる【NGKI3012】。また、intnoで指定した割込み番号に対応する割込
15280 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)
15281 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
15282 【NGKI3013】。さらに、intnoでカーネル管理外の割込みを指定した場合にも、
15283 E_OBJエラーとなる【NGKI3014】。
15284
15285 静的APIにおいては、isridはオブジェクト識別名、isratr, intno, isrpriは整
15286 数定数式パラメータ、exinfとisrは一般定数式パラメータである【NGKI3015】。
15287
15288 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
15289 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
15290 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
15291 【NGKI3016】。また、intnoで指定した割込み要求ラインに対して登録済みの割
15292 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
15293 と異なるクラスに属する場合にも、E_RSATRエラーとなる【NGKI3017】。さらに、
15294 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
15295 限がある場合がある【NGKI3018】。生成する割込みサービスルーチンの属する
15296 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
15297 【NGKI3019】。
15298
15299 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、
15300 ターゲット定義である【NGKI3020】。

15301

15302 【TOPPERS/ASPカーネルにおける規定】

15303

15304 ASPカーネルでは、ATT_ISRのみをサポートする【ASPS0209】。ただし、動的生

15305 成機能拡張パッケージでは、acre_isrもサポートする【ASPS0195】。

15306

15307 【TOPPERS/FMPカーネルにおける規定】

15308

15309 FMPカーネルでは、ATT_ISRのみをサポートする【FMPS0161】。

15310

15311 【TOPPERS/HRP2カーネルにおける規定】

15312

15313 HRP2カーネルでは、ATT_ISRのみをサポートする【HRPS0161】。ただし、動的生

15314 成機能拡張パッケージでは、acre_isrもサポートする【HRPS0208】。

15315

15316 【TOPPERS/SSPカーネルにおける規定】

15317

15318 SSPカーネルでは、ATT_ISRのみをサポートする【SSPS0139】。

15319

15320 【 μ ITRON4.0仕様との関係】

15321

15322 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

15323 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

15324 APIである。

15325 -----

15326 AID_ISR 割付け可能な割込みサービスルーチンIDの数の指定〔SD〕【NGKI3021】

15327

15328 【静的API】

15329 AID_ISR(uint_t noisr)

15330

15331 【パラメータ】

15332 uint_t noisr 割付け可能な割込みサービスルーチンIDの数

15333

15334 【エラーコード】

15335 E_RSATR 予約属性

15336 ・保護ドメインの囲みの中に記述されている〔P〕【NGKI3439】

15337 ・クラスの囲みの中に記述されていない〔M〕【NGKI3022】

15338 E_PAR パラメータエラー

15339 ・noisrが負の値【NGKI3287】

15340

15341 【機能】

15342

15343 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

15344 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

15345 て確保する【NGKI3024】。

15346

15347 noisrは整数定数式パラメータである【NGKI3025】。

15348

15349 【TOPPERS/ASPカーネルにおける規定】

15350

```

15351 ASPカーネルの動的生成機能拡張パッケージでは、AID_ISRをサポートする
15352 【ASPS0219】。
15353
15354 【TOPPERS/HRP2カーネルにおける規定】
15355
15356 HRP2カーネルの動的生成機能拡張パッケージでは、AID_ISRをサポートする
15357 【HRPS0220】。
15358 -----
15359 SAC_ISR      割込みサービスルーチンのアクセス許可ベクタの設定 [SP] 【NGKI3026】
15360 sac_isr      割込みサービスルーチンのアクセス許可ベクタの設定 [TPD] 【NGKI3027】
15361
15362 【静的API】
15363     SAC_ISR(ID isrid, { ACPTN acptn1, ACPTN acptn2,
15364                      ACPTN acptn3, ACPTN acptn4 })
15365
15366 【C言語API】
15367     ER ercd = sac_isr(ID isrid, const ACVCT *p_acvct)
15368
15369
15370 【パラメータ】
15371     ID          isrid      対象割込みサービスルーチンのID番号
15372     ACVCT *      p_acvct    アクセス許可ベクタを入れたパケットへのポ
15373                             インタ（静的APIを除く）
15374
15375     *アクセス許可ベクタ（パケットの内容）
15376     ACPTN        acptn1     通常操作1のアクセス許可パターン
15377     ACPTN        acptn2     通常操作2のアクセス許可パターン
15378     ACPTN        acptn3     管理操作のアクセス許可パターン
15379     ACPTN        acptn4     参照操作のアクセス許可パターン
15380
15381 【リターンパラメータ】
15382     ER          ercd        正常終了（E_OK）またはエラーコード
15383
15384 【エラーコード】
15385     E_CTX        コンテキストエラー
15386                 ・非タスクコンテキストからの呼出し [s] 【NGKI3028】
15387                 ・CPUロック状態からの呼出し [s] 【NGKI3029】
15388     E_ID          不正ID番号
15389                 ・isridが有効範囲外 [s] 【NGKI3030】
15390     E_RSATR       予約属性
15391                 ・カーネルドメインの囲みの中に記述されていない [S] 【NGKI3031】
15392                 ・対象割込みサービスルーチンが属するクラスの囲みの中に
15393                   記述されていない [SM] 【NGKI3032】
15394     E_NOEXS       オブジェクト未登録
15395                 ・対象割込みサービスルーチンが未登録 【NGKI3033】
15396     E_OACV        オブジェクトアクセス違反
15397                 ・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
15398                   ていない [s] 【NGKI3034】
15399     E_MACV        メモリアクセス違反
15400                 ・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

```

15401 いない) [s] 【NGKI3035】

15402 E_OBJ オブジェクト状態エラー

15403 ・対象割込みサービスルーチンは静的APIで生成された [s]

15404 【NGKI3036】

15405 ・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

15406 が設定済み [S] 【NGKI3037】

15407

15408 【機能】

15409

15410 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

15411 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

15412 した値に設定する 【NGKI3038】 .

15413

15414 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

15415 式パラメータである 【NGKI3039】 .

15416

15417 【TOPPERS/HRP2カーネルにおける規定】

15418

15419 HRP2カーネルでは、SAC_ISR, sac_isrをサポートしない 【HRPS0162】 . ただし、

15420 動的生成機能拡張パッケージでは、sac_isrをサポートする 【HRPS0209】 .

15421

15422 【未決定事項】

15423

15424 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

15425 セス許可ベクタでアクセス保護する方法も考えられる.

15426 -----

15427 del_isr 割込みサービスルーチンの削除 [TD] 【NGKI3040】

15428

15429 【C言語API】

15430 ER ercd = del_isr(ID isrid)

15431

15432 【パラメータ】

15433 ID isrid 対象割込みサービスルーチンのID番号

15434

15435 【リターンパラメータ】

15436 ER ercd 正常終了 (E_OK) またはエラーコード

15437

15438 【エラーコード】

15439 E_CTX コンテキストエラー

15440 ・非タスクコンテキストからの呼出し 【NGKI3041】

15441 ・CPUロック状態からの呼出し 【NGKI3042】

15442 E_ID 不正ID番号

15443 ・isridが有効範囲外 【NGKI3043】

15444 E_NOEXS オブジェクト未登録

15445 ・対象割込みサービスルーチンが未登録 【NGKI3044】

15446 E_OACV オブジェクトアクセス違反

15447 ・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

15448 ていない [P] 【NGKI3045】

15449 E_OBJ オブジェクト状態エラー

15450 ・対象割込みサービスルーチンは静的APIで生成された 【NGKI3046】

15451
15452 **【機能】**
15453
15454 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
15455 除する．具体的な振舞いは以下の通り．
15456
15457 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
15458 IDが未使用の状態に戻される【NGKI3047】．
15459
15460 **【TOPPERS/ASPカーネルにおける規定】**
15461
15462 ASPカーネルでは，del_isrをサポートしない【ASPS0197】．ただし，動的生成
15463 機能拡張パッケージでは，del_isrをサポートする【ASPS0198】．
15464
15465 **【TOPPERS/FMPカーネルにおける規定】**
15466
15467 FMPカーネルでは，del_isrをサポートしない【FMPS0163】．
15468
15469 **【TOPPERS/HRP2カーネルにおける規定】**
15470
15471 HRP2カーネルでは，del_isrをサポートしない【HRPS0163】．ただし，動的生成
15472 機能拡張パッケージでは，del_isrをサポートする【HRPS0210】．
15473
15474 **【TOPPERS/SSPカーネルにおける規定】**
15475
15476 SSPカーネルでは，del_isrをサポートしない【SSPS0141】．
15477 -----
15478 ref_isr 割込みサービスルーチンの状態参照 [T]
15479
15480 **【C言語API】**
15481 ER ercd = ref_isr(ID isrid, T_RISR *pk_risr)
15482
15483 ☆未完成
15484
15485 **【TOPPERS/ASPカーネルにおける規定】**
15486
15487 ASPカーネルでは，ref_isrをサポートしない．
15488
15489 **【TOPPERS/FMPカーネルにおける規定】**
15490
15491 FMPカーネルでは，ref_isrをサポートしない．
15492
15493 **【TOPPERS/HRP2カーネルにおける規定】**
15494
15495 HRP2カーネルでは，ref_isrをサポートしない．
15496
15497 **【TOPPERS/SSPカーネルにおける規定】**
15498
15499 SSPカーネルでは，ref_isrをサポートしない．
15500 -----

15501 DEF_INH 割込みハンドラの定義 [S] 【NGKI3048】
15502 def_inh 割込みハンドラの定義 [TD] 【NGKI3049】
15503
15504 **【静的API】**
15505 DEF_INH(INHNO inhno, { ATR inhatr, INTHDR inthdr })
15506
15507 **【C言語API】**
15508 ER ercd = def_inh(INHNO inhno, const T_DINH *pk_dinh)
15509
15510 **【パラメータ】**
15511 INHNO inhno 割込みハンドラ番号
15512 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）
15513
15514
15515 * 割込みハンドラの定義情報（パケットの内容）
15516 ATR inhatr 割込みハンドラ属性
15517 INTHDR inthdr 割込みハンドラ先頭番地
15518
15519 **【リターンパラメータ】**
15520 ER ercd 正常終了 (E_OK) またはエラーコード
15521
15522 **【エラーコード】**
15523 E_CTX コンテキストエラー
15524 ・非タスクコンテキストからの呼出し [s] 【NGKI3050】
15525 ・CPUロック状態からの呼出し [s] 【NGKI3051】
15526 E_RSATR 予約属性
15527 ・inhatrが無効 【NGKI3052】
15528 ・属するクラスの指定が有効範囲外 [sM] 【NGKI3053】
15529 ・クラスの囲みの中に記述されていない [SM] 【NGKI3054】
15530 ・その他の条件については機能の項を参照
15531 E_PAR パラメータエラー
15532 ・inhnoが有効範囲外 【NGKI3055】
15533 ・inthdrがプログラムの先頭番地として正しくない 【NGKI3056】
15534 ・その他の条件については機能の項を参照
15535 E_OACV オブジェクトアクセス違反
15536 ・システム状態に対する管理操作が許可されていない [sP]
15537 【NGKI3057】
15538 E_MACV メモリアクセス違反
15539 ・pk_dinhが指すメモリ領域への読出しアクセスが許可されていない [sP] 【NGKI3058】
15540
15541 E_OBJ オブジェクト状態エラー
15542 ・条件については機能の項を参照
15543
15544 **【機能】**
15545
15546 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
15547 パラメータで指定した割込みハンドラ定義情報に従って、割込みハンドラを定義する【NGKI3059】。ただし、def_inhにおいてpk_dinhをNULLにした場合には、対象割込みハンドラ番号に対する割込みハンドラの定義を解除する【NGKI3060】。
15548
15549
15550

15551 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
15552 数式パラメータである【NGKI3061】。

15553

15554 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
15555 pk_dinhをNULL以外にした場合）には、次のエラーが検出される。

15556

15557 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
15558 ない場合には、E_OBJエラーとなる【NGKI3062】。また、対象割込みハンドラ番号
15559 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
15560 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
15561 場合にも、E_OBJエラーとなる【NGKI3063】。

15562

15563 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
15564 を定義できる場合には、次のエラーが検出される【NGKI3064】。カーネル管理
15565 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
15566 は、E_OBJエラーとなる【NGKI3065】。逆に、カーネル管理の割込みハンドラを
15567 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
15568 【NGKI3066】。また、ターゲット定義でカーネル管理外に固定されている割込
15569 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
15570 TA_NONKERNELを指定しない場合には、E_RSATRエラーとなる【NGKI3067】。逆に、
15571 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
15572 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
15573 場合には、E_RSATRエラーとなる【NGKI3068】。

15574

15575 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
15576 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
15577 【NGKI3070】。また、def_inhで割込みハンドラを定義する場合には、割込みハ
15578 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
15579 TA_DOM(domid)を指定した場合にはE_RSATRエラーとなる【NGKI3071】。ただし、
15580 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
15581 出されない【NGKI3072】。

15582

15583 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
15584 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、
15585 E_RSATRエラーとなる【NGKI3073】。また、ターゲット定義で、割込みハンドラ
15586 が属することができるクラスに制限がある場合がある【NGKI3074】。登録する
15587 割込みハンドラの属するクラスが、ターゲット定義の制限に合致しない場合に
15588 も、E_RSATRエラーとなる【NGKI3075】。

15589

15590 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
15591 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
15592 ない場合には、E_OBJエラーとなる【NGKI3076】。また、対象割込みハンドラ番号
15593 に対して定義された割込みハンドラが、静的APIで定義されたものである場合に
15594 は、ターゲット定義でE_OBJエラーとなる場合がある【NGKI3077】。

15595

15596 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
15597 合には、E_PARエラーとなる【NGKI3078】。具体的には、マルチプロセッサ対応
15598 カーネルにおいて、def_inhを呼び出したタスクが割り付けられているプロセッ
15599 サから、対象割込みハンドラを定義（または定義解除）できない場合が、これ
16000 に該当する【NGKI3079】。

15601
15602 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
15603 は、ターゲット定義である【NGKI3080】。
15604
15605 【TOPPERS/ASPカーネルにおける規定】
15606
15607 ASPカーネルでは、DEF_INHのみをサポートする【ASPS0199】。
15608
15609 【TOPPERS/FMPカーネルにおける規定】
15610
15611 FMPカーネルでは、DEF_INHのみをサポートする【FMPS0164】。
15612
15613 【TOPPERS/HRP2カーネルにおける規定】
15614
15615 HRP2カーネルでは、DEF_INHのみをサポートする【HRPS0164】。
15616
15617 【TOPPERS/SSPカーネルにおける規定】
15618
15619 SSPカーネルでは、DEF_INHのみをサポートする【SSPS0142】。
15620
15621 【 μ ITRON4.0仕様との関係】
15622
15623 inthdrのデータ型をINTHDRに変更した。
15624
15625 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、
15626 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
15627 義を解除してから、再度定義する必要がある。
15628 -----
15629 dis_int 割込みの禁止 [T] 【NGKI3081】
15630
15631 【C言語API】
15632 ER ercd = dis_int(INTNO intno)
15633
15634 【パラメータ】
15635 INTNO intno 割込み番号
15636
15637 【リターンパラメータ】
15638 ER ercd 正常終了 (E_OK) またはエラーコード
15639
15640 【エラーコード】
15641 E_CTX コンテキストエラー
15642 ・非タスクコンテキストからの呼出し【NGKI3082】
15643 E_NOSPT 未サポートエラー
15644 ・条件については機能の項を参照
15645 E_PAR パラメータエラー
15646 ・intnoが有効範囲外【NGKI3083】
15647 ・その他の条件については機能の項を参照
15648 E_OACV オブジェクトアクセス違反
15649 ・システム状態に対する通常操作2が許可されていない [P]
15650 【NGKI3084】

15651 E_OBJ オブジェクト状態エラー
15652 ・対象割込み要求ラインに対して割込み要求ライン属性が設
15653 定されていない【NGKI3085】
15654

15655 **【機能】**

15656
15657 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
15658 フラグをセットする【NGKI3086】．

15659
15660 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
15661 きない場合には、E_PARエラーとなる【NGKI3087】．具体的には、対象割込み要
15662 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
15663 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
15664 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
15665 作できない場合が、これに該当する．

15666
15667 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
15668 る場合がある【NGKI3089】．特にマルチプロセッサ対応カーネルでは、あるプ
15669 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
15670 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．

15671
15672 ターゲット定義で、dis_intがサポートされていない場合がある【NGKI3091】．
15673 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
15674 義される【NGKI3092】．サポートされていない場合にdis_intを呼び出すと、
15675 E_NOSPTエラーが返るか、リンク時にエラーとなる【NGKI3093】．

15676
15677 **【μITRON4.0仕様との関係】**

15678
15679 μITRON4.0仕様で実装定義としていたintnoの意味を標準化した．

15680
15681 CPUロック状態でも呼び出せるものとした．

15682 -----
15683 ena_int 割込みの許可〔T〕【NGKI3094】

15684
15685 **【C言語API】**

15686 ER ercd = ena_int(INTNO intno)

15687
15688 **【パラメータ】**

15689 INTNO intno 割込み番号

15690
15691 **【リターンパラメータ】**

15692 ER ercd 正常終了（E_OK）またはエラーコード

15693
15694 **【エラーコード】**

15695 E_CTX コンテキストエラー
15696 ・非タスクコンテキストからの呼出し【NGKI3095】
15697 E_NOSPT 未サポートエラー
15698 ・条件については機能の項を参照
15699 E_PAR パラメータエラー
15700 ・intnoが有効範囲外【NGKI3096】

15701 ・その他の条件については機能の項を参照
15702 E_OACV オブジェクトアクセス違反
15703 ・システム状態に対する通常操作2が許可されていない [P]
15704 【NGKI3097】
15705 E_OBJ オブジェクト状態エラー
15706 ・対象割込み要求ラインに対して割込み要求ライン属性が設
15707 定されていない【NGKI3098】
15708
15709 【機能】
15710
15711 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
15712 フラグをクリアする【NGKI3099】。
15713
15714 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
15715 きない場合には、E_PARエラーとなる【NGKI3100】。具体的には、対象割込み要
15716 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
15717 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
15718 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
15719 作できない場合が、これに該当する。
15720
15721 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
15722 る場合がある【NGKI3102】。特にマルチプロセッサ対応カーネルでは、あるプ
15723 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
15724 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。
15725
15726 ターゲット定義で、ena_intがサポートされていない場合がある【NGKI3104】。
15727 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
15728 義される【NGKI3105】。サポートされていない場合にena_intを呼び出すと、
15729 E_NOSPTエラーが返るか、リンク時にエラーとなる【NGKI3106】。
15730
15731 【μITRON4.0仕様との関係】
15732
15733 μITRON4.0仕様で実装定義としていたintnoの意味を標準化した。
15734
15735 CPUロック状態でも呼び出せるものとした。
15736 -----
15737 ref_int 割込み要求ラインの参照 [T]
15738
15739 【C言語API】
15740 ER ercd = ref_int(INTNO intno, T_RINT *pk_rint)
15741
15742 ☆未完成
15743
15744 【TOPPERS/ASPカーネルにおける規定】
15745
15746 ASPカーネルでは、ref_intをサポートしない。
15747
15748 【TOPPERS/FMPカーネルにおける規定】
15749
15750 FMPカーネルでは、ref_intをサポートしない。

15751
15752 **【TOPPERS/HRP2カーネルにおける規定】**
15753
15754 HRP2カーネルでは、ref_intをサポートしない。
15755
15756 **【TOPPERS/SSPカーネルにおける規定】**
15757
15758 SSPカーネルでは、ref_intをサポートしない。
15759
15760 **【 μ ITRON4.0仕様との関係】**
15761
15762 μ ITRON4.0仕様に定義されていないサービスコールである。
15763 -----
15764 chg_ipm 割込み優先度マスクの変更 [T] **【NGKI3107】**
15765
15766 **【C言語API】**
15767 ER ercd = chg_ipm(PRI intpri)
15768
15769 **【パラメータ】**
15770 PRI intpri 割込み優先度マスク
15771
15772 **【リターンパラメータ】**
15773 ER ercd 正常終了 (E_OK) またはエラーコード
15774
15775 **【エラーコード】**
15776 E_CTX コンテキストエラー
15777 ・非タスクコンテキストからの呼出し **【NGKI3108】**
15778 ・CPUロック状態からの呼出し **【NGKI3109】**
15779 E_PAR パラメータエラー
15780 ・条件については機能の項を参照
15781 E_OACV オブジェクトアクセス違反
15782 ・システム状態に対する通常操作2が許可されていない [P]
15783 **【NGKI3110】**
15784
15785 **【機能】**
15786
15787 割込み優先度マスクを、intpriで指定した値に変更する **【NGKI3111】**。
15788
15789 intpriは、TMIN_INTPRI以上、TIPM_ENAALL以下でなければならない。そうでな
15790 い場合には、E_PARエラーとなる **【NGKI3113】**。ただし、ターゲット定義の拡張
15791 として、TMIN_INTPRIよりも小さい値を指定できる場合がある **【NGKI3114】**。
15792
15793 **【補足説明】**
15794
15795 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
15796 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
15797 の実行が開始される可能性がある。
15798
15799 **【TOPPERS/SSPカーネルにおける規定】**
15800

15801 SSPカーネルでは、chg_ipmをサポートしない【SSPS0143】。

15802

15803 【 μ ITRON4.0仕様との関係】

15804

15805 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
15806 義となっているサービスコールである。

15807 -----

15808 get_ipm 割込み優先度マスクの参照 [T] 【NGKI3115】

15809

15810 【C言語API】

15811 ER ercd = get_ipm(PRI *p_intpri)

15812

15813 【パラメータ】

15814 PRI * p_intpri 割込み優先度マスクを入れるメモリ領域へのポ
15815 インタ

15816

15817 【リターンパラメータ】

15818 ER ercd エラーコード
15819 PRI intpri 割込み優先度マスク

15820

15821 【エラーコード】

15822 E_CTX コンテキストエラー
15823 ・非タスクコンテキストからの呼出し【NGKI3116】
15824 ・CPUロック状態からの呼出し【NGKI3117】
15825 E_OACV オブジェクトアクセス違反
15826 ・システム状態に対する参照操作が許可されていない [P]
15827 【NGKI3118】
15828 E_MACV メモリアクセス違反
15829 ・p_intpriが指すメモリ領域への書込みアクセスが許可され
15830 ていない [P] 【NGKI3119】

15831

15832 【機能】

15833

15834 割込み優先度マスクの現在値を参照する。参照した割込み優先度マスクは、
15835 p_intpriが指すメモリ領域に返される【NGKI3120】。

15836

15837 【TOPPERS/SSPカーネルにおける規定】

15838

15839 SSPカーネルでは、get_ipmをサポートしない【SSPS0144】。

15840

15841 【 μ ITRON4.0仕様との関係】

15842

15843 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
15844 義となっているサービスコールである。

15845 -----

15846

15847 4.10 CPU例外管理機能

15848

15849 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
15850 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

15851 【NGKI3121】.

15852

15853 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
15854 る【NGKI3122】.

15855

15856 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、
15857 ターゲット定義のCPU例外ハンドラ属性を指定できる場合がある【NGKI3123】.
15858 ターゲット定義のCPU例外ハンドラ属性として、次の属性を予約している
15859 【NGKI3124】.

15860

15861 TA_DIRECT CPU例外ハンドラを直接呼び出す

15862

15863 C言語によるCPU例外ハンドラの記述形式は次の通り【NGKI3125】.

15864

```
15865       void cpu_exception_handler(void *p_excinfo)
15866       {
15867           CPU例外ハンドラ本体
15868       }
```

15869

15870 p_excinfo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
15871 【NGKI3126】. これは、CPU例外ハンドラ内で、CPU例外発生時の状態を参照す
15872 る際に必要となる.

15873

15874 DEF_EXC CPU例外ハンドラの定義 [S] 【NGKI3127】

15875 def_exc CPU例外ハンドラの定義 [TD] 【NGKI3128】

15876

15877 【静的API】

15878 DEF_EXC(EXCNO excno, { ATR excatr, EXCHDR exchdr })

15879

15880 【C言語API】

15881 ER ercd = def_exc(EXCNO excno, const T_DEXC *pk_dexc)

15882

15883 【パラメータ】

15884 EXCNO excno CPU例外ハンドラ番号
15885 T_DEXC * pk_dexc CPU例外ハンドラの定義情報を入れたバケットへ
15886 のポインタ（静的APIを除く）

15887

15888 *CPU例外ハンドラの定義情報（パケットの内容）

15889 ATR excatr CPU例外ハンドラ属性

15890 EXCHDR exchdr CPU例外ハンドラの先頭番地

15891

15892 【リターンパラメータ】

15893 ER ercd 正常終了 (E_OK) またはエラーコード

15894

15895 【エラーコード】

15896 E_CTX コンテキストエラー

15897 • 非タスクコンテキストからの呼出し [s] 【NGKI3129】

15898 • CPUロック状態からの呼出し [s] 【NGKI3130】

15899 E_RSATR 予約属性

15900 • excatrが無効【NGKI3131】

15901 ・ 属するクラスの指定が有効範囲外 [sM] 【NGKI3132】
15902 ・ クラスの囲みの中に記述されていない [SM] 【NGKI3133】
15903 ・ その他の条件については機能の項を参照
15904 E_PAR パラメータエラー
15905 ・ excnoが有効範囲外 【NGKI3134】
15906 ・ exchdrがプログラムの先頭番地として正しくない 【NGKI3135】
15907 E_OACV オブジェクトアクセス違反
15908 ・ システム状態に対する管理操作が許可されていない [sP]
15909 【NGKI3136】
15910 E_MACV メモリアクセス違反
15911 ・ pk_dexcが指すメモリ領域への読出しアクセスが許可されて
15912 いない [sP] 【NGKI3137】
15913 E_OBJ オブジェクト状態エラー
15914 ・ 条件については機能の項を参照
15915
15916 【機能】
15917
15918 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、
15919 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
15920 を定義する 【NGKI3138】。ただし、def_excにおいてpk_dexcをNULLにした場合
15921 には、対象CPU例外ハンドラ番号に対するCPU例外ハンドラの定義を解除する
15922 【NGKI3139】。
15923
15924 静的APIにおいては、excnoとexcattrは整数定数式パラメータ、exchdrは一般定
15925 数式パラメータである 【NGKI3140】。
15926
15927 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
15928 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
15929 CPU例外ハンドラが定義されている場合には、E_OBJエラーとなる 【NGKI3141】。
15930
15931 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
15932 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
15933 【NGKI3143】。また、def_excでCPU例外ハンドラを定義する場合には、CPU例外
15934 ハンドラの属する保護ドメインを設定する必要はなく、CPU例外ハンドラ属性に
15935 TA_DOM(domid)を指定した場合にはE_RSATRエラーとなる 【NGKI3144】。ただし、
15936 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
15937 出されない 【NGKI3145】。
15938
15939 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
15940 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、
15941 E_RSATRエラーとなる 【NGKI3146】。
15942
15943 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
15944 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
15945 ない場合には、E_OBJエラーとなる 【NGKI3147】。また、対象CPU例外ハンドラ
15946 番号に対して定義されたCPU例外ハンドラが、静的APIで定義されたものである
15947 場合には、ターゲット定義でE_OBJエラーとなる場合がある 【NGKI3148】。
15948
15949 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
15950 は、ターゲット定義である 【NGKI3149】。

15951

15952 【TOPPERS/ASPカーネルにおける規定】

15953

15954 ASPカーネルでは、DEF_EXCのみをサポートする【ASPS0200】。

15955

15956 【TOPPERS/FMPカーネルにおける規定】

15957

15958 FMPカーネルでは、DEF_EXCのみをサポートする【FMPS0165】。

15959

15960 【TOPPERS/HRP2カーネルにおける規定】

15961

15962 HRP2カーネルでは、DEF_EXCのみをサポートする【HRPS0165】。

15963

15964 【TOPPERS/SSPカーネルにおける規定】

15965

15966 SSPカーネルでは、DEF_EXCのみをサポートする【SSPS0145】。

15967

15968 【 μ ITRON4.0仕様との関係】

15969

15970 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、

15971 E_OBJエラーとすることにした。

15972 -----

15973 xsns_dpn CPU例外発生時のディスパッチ保留状態の参照 [TI] 【NGKI3150】

15974

15975 【C言語API】

15976 bool_t stat = xsns_dpn(void *p_excinf)

15977

15978 【パラメータ】

15979 void * p_excinf CPU例外の情報を記憶しているメモリ領域の先頭

15980 番地

15981

15982 【リターンパラメータ】

15983 bool_t state ディスパッチ保留状態

15984

15985 【機能】

15986

15987 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。

15988

15989

15990 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

15991 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

15992 状態でなかった場合にfalse、そうでない場合にtrueが返る【NGKI3151】。

15993

15994 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

15995 した場合には、trueが返る【NGKI3152】。

15996

15997 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

15998 【NGKI3153】。それ以外の値を渡した場合の動作は保証されない【NGKI3552】。

15999

16000 【使用方法】

16001
16002 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
16003 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
16004 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
16005 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、
16006 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
16007 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。
16008
16009 **【使用上の注意】**
16010
16011 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
16012 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
16013 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
16014 味を持たない。
16015
16016 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。
16017
16018 **【TOPPERS/SSPカーネルにおける規定】**
16019
16020 SSPカーネルでは、xsns_dpnをサポートしない【SSPS0146】。
16021
16022 **【μITRON4.0仕様との関係】**
16023
16024 μITRON4.0仕様に定義されていないサービスコールである。
16025
16026 **【仕様決定の理由】**
16027
16028 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
16029 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
16030 ストであるため、xsns_dpnをタスクコンテキストから呼び出した場合に必ずtrue
16031 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
16032 的に禁止している。
16033 -----
16034 xsns_xpn CPU例外発生時のタスク例外処理保留状態の参照〔TI〕【NGKI3154】
16035
16036 **【C言語API】**
16037 bool_t stat = xsns_xpn(void *p_excinf)
16038
16039 **【パラメータ】**
16040 void * p_excinf CPU例外の情報を記憶しているメモリ領域の先頭
16041 番地
16042
16043 **【リターンパラメータ】**
16044 bool_t state タスク例外処理保留状態
16045
16046 **【機能】**
16047
16048 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
16049 参照する。具体的な振舞いは以下の通り。
16050

16051 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
16052 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
16053 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
16054 【NGKI3155】。

16055
16056 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
16057 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
16058 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
16059 外実行開始時スタック不正例外が発生する）場合にも、trueを返す【NGKI3156】。

16060
16061 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
16062 した場合には、trueが返る【NGKI3157】。

16063
16064 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
16065 【NGKI3158】。

16066
16067 **【使用方法】**

16068
16069 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
16070 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
16071 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
16072 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
16073 カバリ処理を行うことができる。

16074
16075 **【使用上の注意】**

16076
16077 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
16078 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
16079 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
16080 味を持たない。

16081
16082 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。

16083
16084 **【TOPPERS/SSPカーネルにおける規定】**

16085
16086 SSPカーネルでは、xsns_xpnをサポートしない【SSPS0147】。

16087
16088 **【μITRON4.0仕様との関係】**

16089
16090 μITRON4.0仕様に定義されていないサービスコールである。

16091
16092 **【仕様決定の理由】**

16093
16094 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
16095 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
16096 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
16097 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
16098 的に禁止している。

16099 -----
16100

16101 4.11 拡張サービスコール管理機能

16102

16103 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
16104 で実行すべきルーチンと呼び出すための機能である【NGKI3159】。特権モード
16105 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、
16106 特権モードで実行される処理単位からも呼び出すことができる【NGKI3160】。

16107

16108 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
16109 属する【NGKI3161】。拡張サービスコールは、それを呼び出す処理単位とは別
16110 の処理単位であり、拡張サービスコールからカーネルオブジェクトをアクセス
16111 する場合には、拡張サービスコールがアクセスの主体となる【NGKI3162】。そ
16112 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、
16113 すべての種別のアクセスを行うことが許可される。

16114

16115 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
16116 め、拡張サービスコール管理機能をサポートしない【NGKI3163】。

16117

16118 拡張サービスコール属性に指定できる属性はない【NGKI3686】。そのため拡張
16119 サービスコール属性には、TA_NULLを指定しなければならない【NGKI3687】。

16120

16121 C言語による拡張サービスコールの記述形式は次の通り【NGKI3164】。

16122

```
16123     ER_UINT extended_svc(intptr_t par1, intptr_t par2, intptr_t par3,  
16124                           intptr_t par4, intptr_t par5, ID cdmid)  
16125     {  
16126         拡張サービスコール本体  
16127     }
```

16128

16129 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
16130 ID番号が渡される【NGKI3165】。すなわち、拡張サービスコールから呼び出し
16131 た場合にはTDOM_KERNEL(=-1)が、タスク本体(拡張サービスコールを除く)
16132 から呼び出した場合にはそのタスク(自タスク)の属する保護ドメインIDが渡
16133 される。

16134

16135 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
16136 【NGKI3166】。

16137

16138 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。

16139

```
16140     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  
16141                     カーネルでは、登録できる拡張サービスコールの数に一  
16142                     致)【NGKI3167】
```

16143

16144 【TOPPERS/ASPカーネルにおける規定】

16145

16146 ASPカーネルでは、拡張サービスコール管理機能をサポートしない【ASPS0201】。

16147

16148 【TOPPERS/FMPカーネルにおける規定】

16149

16150 FMPカーネルでは、拡張サービスコール管理機能をサポートしない【FMPS0166】。

16151
16152 【TOPPERS/HRP2カーネルにおける規定】
16153
16154 HRP2カーネルでは、拡張サービスコール管理機能をサポートする【HRPS0166】。
16155
16156 【TOPPERS/SSPカーネルにおける規定】
16157
16158 SSPカーネルでは、拡張サービスコール管理機能をサポートしない【SSPS0148】。
16159
16160 【未決定事項】
16161
16162 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
16163 は未決定である。
16164
16165 【 μ ITRON4.0仕様との関係】
16166
16167 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
16168 メータ5個に固定した。
16169
16170 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
16171 番号を渡す機能を追加した。
16172
16173 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。
16174 -----
16175 DEF_SVC 拡張サービスコールの定義 [SP] 【NGKI3168】
16176 def_svc 拡張サービスコールの定義 [TPD] 【NGKI3169】
16177
16178 【静的API】
16179 DEF_SVC(FN fncd, { ATR svcatr, EXTSVC extsvc, SIZE stksz })
16180
16181 【C言語API】
16182 ER ercd = def_svc(FN fncd, const T_DSVC *pk_dsvc)
16183
16184 【パラメータ】
16185 FN fncd 拡張サービスコールの機能コード
16186 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
16187 トへのポインタ（静的APIを除く）
16188
16189 *拡張サービスコールの定義情報（パケットの内容）
16190 ATR svcatr 拡張サービスコール属性
16191 EXTSVC extsvc 拡張サービスコールの先頭番地
16192 SIZE stksz 拡張サービスコールで使用するスタックサイズ
16193
16194 【リターンパラメータ】
16195 ER ercd 正常終了 (E_OK) またはエラーコード
16196
16197 【エラーコード】
16198 E_CTX コンテキストエラー
16199 ・非タスクコンテキストからの呼出し [s] 【NGKI3170】
16200 ・CPUロック状態からの呼出し [s] 【NGKI3171】

16201 E_RSATR 予約属性
 16202 ・svcatrが無効【NGKI3172】
 16203 ・その他の条件については機能の項を参照
 16204 E_PAR パラメータエラー
 16205 ・fncdが0または負の値【NGKI3173】
 16206 ・fncdがTMAX_FNCDよりも大きい[s]【NGKI3174】
 16207 ・extsvcがプログラムの先頭番地として正しくない【NGKI3175】
 16208 ・stkszが負の値[S]【NGKI3290】
 16209 E_OACV オブジェクトアクセス違反
 16210 ・システム状態に対する管理操作が許可されていない[s]
 16211 【NGKI3176】
 16212 E_MACV メモリアクセス違反
 16213 ・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
 16214 いない[s]【NGKI3177】
 16215 E_OBJ オブジェクト状態エラー
 16216 ・条件については機能の項を参照

16217 【機能】

16218
 16219
 16220 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
 16221 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
 16222 【NGKI3178】。ただし、def_svcにおいてpk_dsvcをNULLにした場合には、対象
 16223 機能コードに対する拡張サービスコールの定義を解除する【NGKI3179】。

16224
 16225 静的APIにおいては、fncd、svcatr、stkszは整数定数式パラメータ、svchdrは
 16226 一般定数式パラメータである【NGKI3180】。

16227
 16228 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
 16229 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
 16230 スコールが定義されている場合には、E_OBJエラーとなる【NGKI3181】。

16231
 16232 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
 16233 い場合には、E_RSATRエラーとなる【NGKI3183】。また、def_svcで拡張サービ
 16234 スコールを定義する場合には、拡張サービスコールの属する保護ドメインを設
 16235 定する必要はなく、拡張サービスコール属性にTA_DOM(domid)を指定した場合に
 16236 はE_RSATRエラーとなる【NGKI3184】。ただし、TA_DOM(TDOM_SELF)を指定した
 16237 場合には、指定が無視され、E_RSATRエラーは検出されない【NGKI3185】。

16238
 16239 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
 16240 なければならない。そうでない場合には、E_RSATRエラーとなる【NGKI3187】。ま
 16241 た、def_svcで拡張サービスコールを定義する場合には、拡張サービスコールの
 16242 属するクラスを設定する必要はなく、拡張サービスコール属性に
 16243 TA_CLS(clsid)を指定した場合にはE_RSATRエラーとなる【NGKI3188】。ただし、
 16244 TA_CLS(TCLS_SELF)を指定した場合には、指定が無視され、E_RSATRエラーは検
 16245 出されない【NGKI3189】。

16246
 16247 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
 16248 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
 16249 い場合には、E_OBJエラーとなる【NGKI3190】。

16250

16251 【TOPPERS/HRP2カーネルにおける規定】
16252
16253 HRP2カーネルでは、DEF_SVCのみをサポートする【HRPS0167】。
16254
16255 【μ ITRON4.0仕様との関係】
16256
16257 拡張サービスコールの定義情報に、stksz（拡張サービスコールで使用するスタックサイズ）を追加した。
16258
16259
16260 extsvcのデータ型を、EXTSVCに変更した。
16261 -----
16262 cal_svc 拡張サービスコールの呼出し〔TIP〕【NGKI3191】
16263
16264 【C言語API】
16265 ER_UINT ercd = cal_svc(FN fncd, intptr_t par1, intptr_t par2,
16266 intptr_t par3, intptr_t par4, intptr_t par5)
16267
16268 【パラメータ】
16269 FN fncd 呼び出す拡張サービスコールの機能コード
16270 intptr_t par1 拡張サービスコールへの第1パラメータ
16271 intptr_t par2 拡張サービスコールへの第2パラメータ
16272 intptr_t par3 拡張サービスコールへの第3パラメータ
16273 intptr_t par4 拡張サービスコールへの第4パラメータ
16274 intptr_t par5 拡張サービスコールへの第5パラメータ
16275
16276 【リターンパラメータ】
16277 ER_UINT ercd 正常終了（正の値または0）またはエラーコード
16278
16279 【エラーコード】
16280 E_SYS システムエラー
16281 ・条件については機能の項を参照
16282 E_RSFN 予約機能コード
16283 ・fncdが0または負の値【NGKI3192】
16284 ・fncdがTMAX_FNCDよりも大きい【NGKI3193】
16285 ・fncdで指定した機能コードに対して拡張サービスコールが
16286 定義されていない【NGKI3194】
16287 E_NOMEM メモリ不足
16288 ・条件については機能の項を参照
16289 *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。
16290
16291 【機能】
16292
16293 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
16294 パラメータとして呼び出し、拡張サービスコールの返値を返す【NGKI3195】。
16295
16296 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
16297 【NGKI3196】。スタック（ユーザタスクの場合はシステムスタック）の残り領域が、
16298 拡張サービスコールで使用するスタックサイズよりも小さい場合には、
16299 E_NOMEMエラーとなる【NGKI3197】。また、拡張サービスコールのネストレベル
16300 が上限（＝255）を超える場合には、E_SYSエラーが返る【NGKI3198】。

16301
16302
16303
16304
16305
16306
16307
16308
16309
16310
16311
16312
16313
16314
16315
16316
16317
16318
16319
16320
16321
16322
16323
16324
16325
16326
16327
16328
16329
16330
16331
16332
16333
16334
16335
16336
16337
16338
16339
16340
16341
16342
16343
16344
16345
16346
16347
16348
16349
16350

【 μ ITRON4.0仕様との関係】

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かは実装定義としているが、この仕様では、カーネルのサービスコールを呼び出せないこととした。

拡張サービスコールが呼び出される時に、スタックの残り領域のサイズをチェックする機能を追加した。

拡張サービスコールに対するパラメータを、intptr_t型のパラメータ5個に固定し、cal_svcから返るエラー (E_SYS, E_RSFN, E_NOMEM) について規定した。

【仕様決定の理由】

パラメータの型と数を固定したのは、型チェックを厳密にできるようにし、パラメータをコンパイラやコーリングコンベンションによらずに正しく渡せるようにするためである。

4.12 システム構成管理機能

システム構成管理機能には、非タスクコンテキスト用スタック領域を設定する機能、初期化ルーチンと終了処理ルーチンを登録する機能、カーネルのコンフィギュレーション情報やバージョン情報を参照する機能が含まれる。

非タスクコンテキスト用スタック領域は、非タスクコンテキストで実行される処理単位が用いるスタック領域である。

保護機能対応カーネルにおいて、非タスクコンテキスト用のスタック領域は、カーネルの用いるオブジェクト管理領域と同様に扱われる【NGKI3199】。

初期化ルーチンは、カーネルが実行を制御する処理単位で、カーネルの動作開始の直前に、カーネル非動作状態で実行される【NGKI3200】。

保護機能対応カーネルにおいて、初期化ルーチンは、カーネルドメインに属する【NGKI3201】。

初期化ルーチン属性に指定できる属性はない【NGKI3202】。そのため初期化ルーチン属性には、TA_NULLを指定しなければならない【NGKI3203】。

C言語による初期化ルーチンの記述形式は次の通り【NGKI3204】。

```
void initialization_routine(intptr_t exinf)
{
    初期化ルーチン本体
}
```

exinfには、初期化ルーチンの拡張情報が渡される【NGKI3205】。

16351 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
16352 終了の直後に、カーネル非動作状態で実行される【NGKI3206】。

16353
16354 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
16355 する【NGKI3207】。

16356
16357 終了処理ルーチン属性に指定できる属性はない【NGKI3208】。そのため終了処
16358 理ルーチン属性には、TA_NULLを指定しなければならない【NGKI3209】。

16359
16360 C言語による終了処理ルーチンの記述形式は次の通り【NGKI3210】。

```
16361  
16362     void termination_routine(intptr_t exinf)  
16363     {  
16364         終了処理ルーチン本体  
16365     }
```

16366
16367 exinfには、終了処理ルーチンの拡張情報が渡される【NGKI3211】。

16368
16369 【μITRON4.0仕様との関係】

16370
16371 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、
16372 μITRON4.0仕様に規定されていない機能である。

16373 -----
16374 LMT_DOM 保護ドメインに対する制限の設定〔SP〕【NGKI3441】

16375
16376 【静的API】
16377 LMT_DOM({ PRI mintpri })

16378
16379 【パラメータ】
16380 *保護ドメインに対する制限の設定情報
16381 PRI mintpri 指定できる最高のタスク優先度

16382
16383 【エラーコード】
16384 E_RSATR 予約属性
16385 ・ユーザドメインの囲みの中に記述されていない【NGKI3442】
16386 ・クラスの囲みの中に記述されている〔M〕【NGKI3443】
16387 E_OBJ オブジェクト状態エラー
16388 ・保護ドメインに対する制限が設定済み【NGKI3444】
16389 E_PAR パラメータエラー
16390 ・mintpriが有効範囲外【NGKI3445】

16391
16392 【機能】
16393
16394 パラメータで指定した保護ドメインに対する制限の設定情報に従って、ユーザ
16395 ドメインに対する制限を設定する【NGKI3446】。

16396
16397 mintpriは整数定数式パラメータである【NGKI3447】。

16398
16399 LMT_DOMにより保護ドメインに対する制限を設定しないユーザドメインに対して
16400 は、指定できる最高のタスク優先度はTMIN_TPRI+1に設定される【NGKI3448】。

16401
16402
16403
16404
16405
16406
16407
16408
16409
16410
16411
16412
16413
16414
16415
16416
16417
16418
16419
16420
16421
16422
16423
16424
16425
16426
16427
16428
16429
16430
16431
16432
16433
16434
16435
16436
16437
16438
16439
16440
16441
16442
16443
16444
16445
16446
16447
16448
16449
16450

【 μ ITRON4.0/PX仕様との関係】

μ ITRON4.0/PX仕様に定義されていない静的APIである。

DEF_ICS 非タスクコンテキスト用スタック領域の設定 [S] 【NGKI3212】

【静的API】

```
DEF_ICS({ SIZE istksz, STK_T *istk })
```

【パラメータ】

*非タスクコンテキスト用スタック領域の設定情報

SIZE	istksz	非タスクコンテキスト用スタック領域のサイズ (バイト数)
------	--------	---------------------------------

STK_T	istk	非タスクコンテキスト用スタック領域の先頭番地
-------	------	------------------------

【エラーコード】

E_RSATR	予約属性
---------	------

・カーネルドメインの囲みの中に記述されていない [P] 【NGKI3213】

・クラスの囲みの中に記述されていない [M] 【NGKI3214】

E_PAR パラメータエラー

- ・条件については機能の項を参照

E_NOMEM メモリ不足

- ・非タスクコンテキスト用スタック領域が確保できない【NGKI3215】

E_OBJ オブジェクト状態エラー

- ・非タスクコンテキスト用スタック領域が設定済み【NGKI3216】

- ・その他の条件については機能の項を参照

【機能】

各パラメータで指定した非タスクコンテキスト用スタック領域の設定情報に従って、非タスクコンテキスト用スタック領域を設定する【NGKI3217】。istkszに0以下の値を指定した時や、ターゲット定義の最小値よりも小さい値を指定した時には、E_PARエラーとなる【NGKI3254】。

istkszは整数定数式パラメータ，istkは一般定数式パラメータである．コンフィギュレータは，静的APIのメモリ不足（E_NOMEM）エラーを検出することができない【NGKI3218】．

istkszをNULLとした場合、istkszで指定したサイズのスタック領域が、コンフィギュレータにより確保される【NGKI3219】。istkszにターゲット定義の制約に合致しないサイズを指定した時には、ターゲット定義の制約に合致するように大きい方に丸めたサイズで確保される【NGKI3220】。

istkにNULL以外を指定した場合、istkとistksizで指定したスタック領域は、アプリケーションで確保しておく必要がある【NGKI3221】。スタック領域をアプリケーションで確保する方法については、「2.15.3 カーネル共通マクロ」の節を参照すること。その方法に従わず、istkやistksizにターゲット定義の制約に合致しない先頭番地やサイズを指定した時には、E_PARエラーとなる【NGKI3222】。

16451
 16452 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
 16453 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、
 16454 E_OBJエラーとなる【NGKI3223】。
 16455
 16456 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
 16457 ト定義のデフォルトのサイズのスタック領域が、コンフィギュレータにより確
 16458 保される【NGKI3224】。
 16459
 16460 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
 16461 プロセッサ毎に確保する必要がある【NGKI3225】。DEF_ICSにより設定する非タ
 16462 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
 16463 スの初期割付けプロセッサが使用する【NGKI3226】。そのプロセッサに対して
 16464 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、
 16465 E_OBJエラーとなる【NGKI3227】。
 16466
 16467 【TOPPERS/SSPカーネルにおける規定】
 16468
 16469 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
 16470 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
 16471 【SSPS0149】。これは、SSPカーネルでは、すべての処理単位が共有スタック領
 16472 域を使用し、非タスクコンテキストのみが用いるスタック領域を持たないため
 16473 である。そのため、DEF_ICSの役割は、非タスクコンテキストが用いるスタック
 16474 領域のサイズを指定することのみとなる。itskにNULL以外を指定した場合には、
 16475 E_PARエラーとなる【SSPS0150】。
 16476
 16477 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。
 16478
 16479 【μITRON4.0仕様との関係】
 16480
 16481 μITRON4.0仕様に定義されていない静的APIである。
 16482 -----
 16483 DEF_STK 共有スタック領域の設定 [S] 【NGKI3228】
 16484
 16485 【静的API】
 16486 DEF_STK({ SIZE stksz, STK_T *stk })
 16487
 16488 【パラメータ】
 16489 *共有スタック領域の設定情報
 16490 SIZE stksz 共有スタック領域のサイズ (バイト数)
 16491 STK_T stk 共有スタック領域の先頭番地
 16492
 16493 【エラーコード】
 16494 E_PAR パラメータエラー
 16495 ・条件については機能の項を参照
 16496 E_NOMEM メモリ不足
 16497 ・共有スタック領域が確保できない【NGKI3229】
 16498 E_OBJ オブジェクト状態エラー
 16499 ・共有スタック領域が設定済み
 16500

16501 【サポートするカーネル】
16502
16503 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
16504 ネルは、DEF_STKをサポートしない【NGKI3230】。
16505
16506 【機能】
16507
16508 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
16509 領域を設定する【NGKI3231】。stkkszに0以下の値を指定した時や、ターゲット
16510 定義の最小値よりも小さい値を指定した時には、E_PARエラーとなる【NGKI3255】。
16511
16512 stkksz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
16513 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
16514 ない【NGKI3232】。
16515
16516 stkをNULLとした場合、stkkszで指定したサイズのスタック領域が、コンフィギュ
16517 レータにより確保される【NGKI3233】。stkkszにターゲット定義の制約に合致し
16518 ないサイズを指定した時には、ターゲット定義の制約に合致するように大きい
16519 方に丸めたサイズで確保される【NGKI3234】。
16520
16521 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
16522 ケーションで確保しておく必要がある【NGKI3235】。スタック領域をアプリケー
16523 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
16524 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
16525 い先頭番地やサイズを指定した時には、E_PARエラーとなる【NGKI3236】。
16526
16527 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
16528 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
16529 算する【NGKI3237】。DEF_STKにより共有スタック領域を設定しない場合、必要
16530 なサイズの共有スタック領域が、コンフィギュレータにより確保される
16531 【NGKI3238】。
16532
16533 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
16534 りも小さい場合、コンフィギュレータは警告メッセージを出力する【NGKI3239】。
16535
16536 【μITRON4.0仕様との関係】
16537
16538 μITRON4.0仕様に定義されていない静的APIである。
16539 -----
16540 ATT_INI 初期化ルーチンの追加 [S] 【NGKI3240】
16541
16542 【静的API】
16543 ATT_INI({ ATR iniatr, intptr_t exinf, INIRTN inirtn })
16544
16545 【パラメータ】
16546 * 初期化ルーチンの追加情報
16547 ATR iniatr 初期化ルーチン属性
16548 intptr_t exinf 初期化ルーチンの拡張情報
16549 INIRTN inirtn 初期化ルーチンの先頭番地
16550

16551 **【エラーコード】**
16552 E_RSATR 予約属性
16553 ・ iniatrが無効【NGKI3241】
16554 ・ カーネルドメインの囲みの中に記述されていない [P] 【NGKI3242】
16555 E_PAR パラメータエラー
16556 ・ inirtnがプログラムの先頭番地として正しくない【NGKI3243】
16557
16558 **【機能】**
16559
16560 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
16561 追加する【NGKI3244】。
16562
16563 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
16564 【NGKI3245】。
16565
16566 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
16567 である【NGKI3246】。
16568
16569 **【補足説明】**
16570
16571 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
16572 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
16573 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。
16574 -----
16575 ATT_TER 終了処理ルーチンの追加 [S] 【NGKI3247】
16576
16577 **【静的API】**
16578 ATT_TER({ ATR teratr, intptr_t exinf, TERRTN terrtn })
16579
16580 **【パラメータ】**
16581 * 終了処理ルーチンの追加情報
16582 ATR teratr 終了処理ルーチン属性
16583 intptr_t exinf 終了処理ルーチンの拡張情報
16584 TERRTN terrtn 終了処理ルーチンの先頭番地
16585
16586 **【エラーコード】**
16587 E_RSATR 予約属性
16588 ・ teratrが無効【NGKI3248】
16589 ・ カーネルドメインの囲みの中に記述されていない [P] 【NGKI3249】
16590 E_PAR パラメータエラー
16591 ・ terrtnがプログラムの先頭番地として正しくない【NGKI3250】
16592
16593 **【機能】**
16594
16595 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
16596 ンを追加する【NGKI3251】。
16597
16598 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
16599 【NGKI3252】。
16600

16601 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
 16602 である【NGKI3253】.

16603

16604 **【補足説明】**

16605

16606 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
 16607 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
 16608 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.

16609

16610 **【 μ ITRON4.0仕様との関係】**

16611

16612 μ ITRON4.0仕様に定義されていない静的APIである.

16613 -----

16614 ref_cfg コンフィギュレーション情報の参照 [T]

16615

16616 **【C言語API】**

16617 ER ercd = ref_cfg(T_RCFG *pk_rcfg)

16618

16619 ☆未完成

16620

16621 **【TOPPERS/ASPカーネルにおける規定】**

16622

16623 ASPカーネルでは、ref_cfgをサポートしない.

16624

16625 **【TOPPERS/FMPカーネルにおける規定】**

16626

16627 FMPカーネルでは、ref_cfgをサポートしない.

16628

16629 **【TOPPERS/HRP2カーネルにおける規定】**

16630

16631 HRP2カーネルでは、ref_cfgをサポートしない.

16632

16633 **【TOPPERS/SSPカーネルにおける規定】**

16634

16635 SSPカーネルでは、ref_cfgをサポートしない.

16636 -----

16637 ref_ver バージョン情報の参照 [T]

16638

16639 **【C言語API】**

16640 ER ercd = ref_ver(T_RVER *pk_rver)

16641

16642 ☆未完成

16643

16644 **【TOPPERS/ASPカーネルにおける規定】**

16645

16646 ASPカーネルでは、ref_verをサポートしない.

16647

16648 **【TOPPERS/FMPカーネルにおける規定】**

16649

16650 FMPカーネルでは、ref_verをサポートしない.

```

16651
16652     【TOPPERS/HRP2カーネルにおける規定】
16653
16654     HRP2カーネルでは、ref_verをサポートしない。
16655
16656     【TOPPERS/SSPカーネルにおける規定】
16657
16658     SSPカーネルでは、ref_verをサポートしない。
16659     -----
16660
16661
16662     第5章 リファレンス
16663
16664     5.1 サービスコール一覧
16665
16666     (1) タスク管理機能
16667
16668         ER_ID tskid = acre_tsk(const T_CTSK *pk_ctsk)           [TD]
16669         ER ercd = sac_tsk(ID tskid, const ACVCT *p_acvct)      [TPD]
16670         ER ercd = del_tsk(ID tskid)                            [TD]
16671         ER ercd = act_tsk(ID tskid)                            [T]
16672         ER ercd = iact_tsk(ID tskid)                           [I]
16673         ER ercd = mact_tsk(ID tskid, ID prcid)                 [TM]
16674         ER ercd = imact_tsk(ID tskid, ID prcid)                [IM]
16675         ER_UINT actcnt = can_act(ID tskid)                     [T]
16676         ER ercd = mig_tsk(ID tskid, ID prcid)                  [TM]
16677         ER ercd = ext_tsk()                                     [T]
16678         ER ercd = ter_tsk(ID tskid)                             [T]
16679         ER ercd = chg_pri(ID tskid, PRI tskpri)                [T]
16680         ER ercd = get_pri(ID tskid, PRI *p_tskpri)             [T]
16681         ER ercd = get_inf(intptr_t *p_exinf)                   [T]
16682         ER ercd = ref_tsk(ID tskid, T_RTsk *pk_rtsk)           [T]
16683
16684     (2) タスク付属同期機能
16685
16686         ER ercd = slp_tsk()                                     [T]
16687         ER ercd = tslp_tsk(TMO tmout)                           [T]
16688         ER ercd = wup_tsk(ID tskid)                             [T]
16689         ER ercd = iwup_tsk(ID tskid)                           [I]
16690         ER_UINT wupcnt = can_wup(ID tskid)                     [T]
16691         ER ercd = rel_wai(ID tskid)                             [T]
16692         ER ercd = irel_wai(ID tskid)                           [I]
16693         ER ercd = sus_tsk(ID tskid)                             [T]
16694         ER ercd = rsm_tsk(ID tskid)                             [T]
16695         ER ercd = dis_wai(ID tskid)                             [TP]
16696         ER ercd = idis_wai(ID tskid)                           [IP]
16697         ER ercd = ena_wai(ID tskid)                             [TP]
16698         ER ercd = iena_wai(ID tskid)                           [IP]
16699         ER ercd = dly_tsk(RELTIM dlytim)                       [T]
16700

```

```

16701      (3) タスク例外処理機能
16702
16703      ER ercd = def_tex(ID tskid, const T_DTEX *pk_dtex)      [TD]
16704      ER ercd = ras_tex(ID tskid, TEXTPTN rasptn)            [T]
16705      ER ercd = iras_tex(ID tskid, TEXTPTN rasptn)            [I]
16706      ER ercd = dis_tex()                                     [T]
16707      ER ercd = ena_tex()                                      [T]
16708      bool_t state = sns_tex()                                 [TI]
16709      ER ercd = ref_tex(ID tskid, T_RTEX *pk_rtex)            [T]
16710
16711      (4) 同期・通信機能
16712
16713      セマフォ
16714
16715      ER_ID semid = acre_sem(const T_CSEM *pk_csem)            [TD]
16716      ER ercd = sac_sem(ID semid, const ACVCT *p_acvct)        [TPD]
16717      ER ercd = del_sem(ID semid)                               [TD]
16718      ER ercd = sig_sem(ID semid)                               [T]
16719      ER ercd = isig_sem(ID semid)                              [I]
16720      ER ercd = wai_sem(ID semid)                               [T]
16721      ER ercd = pol_sem(ID semid)                               [T]
16722      ER ercd = twai_sem(ID semid, TMO tmout)                  [T]
16723      ER ercd = ini_sem(ID semid)                               [T]
16724      ER ercd = ref_sem(ID semid, T_RSEM *pk_rsem)             [T]
16725
16726      イベントフラグ
16727
16728      ER_ID flgid = acre_flg(const T_CFLG *pk_cflg)            [TD]
16729      ER ercd = sac_flg(ID flgid, const ACVCT *p_acvct)        [TPD]
16730      ER ercd = del_flg(ID flgid)                               [TD]
16731      ER ercd = set_flg(ID flgid, FLGPTN setptn)               [T]
16732      ER ercd = iset_flg(ID flgid, FLGPTN setptn)              [I]
16733      ER ercd = clr_flg(ID flgid, FLGPTN clrptn)               [T]
16734      ER ercd = wai_flg(ID flgid, FLGPTN waiptn,
16735                          MODE wfmode, FLGPTN *p_flgptn)
16736      ER ercd = pol_flg(ID flgid, FLGPTN waiptn,               [T]
16737                          MODE wfmode, FLGPTN *p_flgptn)
16738      ER ercd = twai_flg(ID flgid, FLGPTN waiptn,              [T]
16739                          MODE wfmode, FLGPTN *p_flgptn, TMO tmout)
16740      ER ercd = ini_flg(ID flgid)                               [T]
16741      ER ercd = ref_flg(ID flgid, T_RFLG *pk_rflg)             [T]
16742
16743      データキュー
16744
16745      ER_ID dtqid = acre_dtq(const T_CDTQ *pk_cdtq)            [TD]
16746      ER ercd = sac_dtq(ID dtqid, const ACVCT *p_acvct)        [TPD]
16747      ER ercd = del_dtq(ID dtqid)                               [TD]
16748      ER ercd = snd_dtq(ID dtqid, intptr_t data)               [T]
16749      ER ercd = psnd_dtq(ID dtqid, intptr_t data)              [T]
16750      ER ercd = ipsnd_dtq(ID dtqid, intptr_t data)             [I]

```

16751	ER ercd = tsnd_dtq(ID dtqid, intptr_t data, TMO tmout)	[T]
16752	ER ercd = fsnd_dtq(ID dtqid, intptr_t data)	[T]
16753	ER ercd = ifsnd_dtq(ID dtqid, intptr_t data)	[I]
16754	ER ercd = rcv_dtq(ID dtqid, intptr_t *p_data)	[T]
16755	ER ercd = prcv_dtq(ID dtqid, intptr_t *p_data)	[T]
16756	ER ercd = trecv_dtq(ID dtqid, intptr_t *p_data, TMO tmout)	[T]
16757	ER ercd = ini_dtq(ID dtqid)	[T]
16758	ER ercd = ref_dtq(ID dtqid, T_RDTQ *pk_rdtq)	[T]
16759		
16760	優先度データキュー	
16761		
16762	ER_ID pdqid = acre_pdq(const T_CPDQ *pk_cpdq)	[TD]
16763	ER ercd = sac_pdq(ID pdqid, const ACVCT *p_acvct)	[TPD]
16764	ER ercd = del_pdq(ID pdqid)	[TD]
16765	ER ercd = snd_pdq(ID pdqid, intptr_t data, PRI datapri)	[T]
16766	ER ercd = psnd_pdq(ID pdqid, intptr_t data, PRI datapri)	[T]
16767	ER ercd = ipsnd_pdq(ID pdqid, intptr_t data, PRI datapri)	[I]
16768	ER ercd = tsnd_pdq(ID pdqid, intptr_t data,	[T]
16769	PRI datapri, TMO tmout)	
16770	ER ercd = rcv_pdq(ID pdqid, intptr_t *p_data, PRI *p_datapri)	[T]
16771	ER ercd = prcv_pdq(ID pdqid, intptr_t *p_data, PRI *p_datapri)	[T]
16772	ER ercd = trecv_pdq(ID pdqid, intptr_t *p_data,	[T]
16773	PRI *p_datapri, TMO tmout)	
16774	ER ercd = ini_pdq(ID pdqid)	[T]
16775	ER ercd = ref_pdq(ID pdqid, T_RPDQ *pk_rpdq)	[T]
16776		
16777	メールボックス	
16778		
16779	ER_ID mbxid = acre_mbx(const T_CMBX *pk_cmbx)	[TDp]
16780	ER ercd = del_mbx(ID mbxid)	[TDp]
16781	ER ercd = snd_mbx(ID mbxid, T_MSG *pk_msg)	[Tp]
16782	ER ercd = rcv_mbx(ID mbxid, T_MSG **ppk_msg)	[Tp]
16783	ER ercd = prcv_mbx(ID mbxid, T_MSG **ppk_msg)	[Tp]
16784	ER ercd = trecv_mbx(ID mbxid, T_MSG **ppk_msg, TMO tmout)	[Tp]
16785	ER ercd = ini_mbx(ID mbxid)	[Tp]
16786	ER ercd = ref_mbx(ID mbxid, T_RMBX *pk_rmbx)	[Tp]
16787		
16788	ミューテックス	
16789		
16790	ER_ID mtxid = acre_mtx(const T_CMTX *pk_cmtx)	[TD]
16791	ER ercd = sac_mtx(ID mtxid, const ACVCT *p_acvct)	[TPD]
16792	ER ercd = del_mtx(ID mtxid)	[TD]
16793	ER ercd = loc_mtx(ID mtxid)	[T]
16794	ER ercd = ploc_mtx(ID mtxid)	[T]
16795	ER ercd = tloc_mtx(ID mtxid, TMO tmout)	[T]
16796	ER ercd = unl_mtx(ID mtxid)	[T]
16797	ER ercd = ini_mtx(ID mtxid)	[T]
16798	ER ercd = ref_mtx(ID mtxid, T_RMTX *pk_rmtx)	[T]
16799		
16800	メッセージバッファ	

16801		
16802	ER_ID mbfid = acre_mbf(const T_CMBF *pk_cmbf)	[TD]
16803	ER ercd = sac_mbf(ID mbfid, const ACVCT *p_acvct)	[TPD]
16804	ER ercd = del_mbf(ID mbfid)	[TD]
16805	ER ercd = snd_mbf(ID mbfid, const void *msg, uint_t msgsz)	[T]
16806	ER ercd = psnd_mbf(ID mbfid, const void *msg, uint_t msgsz)	[T]
16807	ER ercd = tsnd_mbf(ID mbfid, const void *msg,	[T]
16808	uint_t msgsz, TMO tmout)	
16809	ER_UINT msgsz = rcv_mbf(ID mbfid, void *msg)	[T]
16810	ER_UINT msgsz = prcv_mbf(ID mbfid, void *msg)	[T]
16811	ER_UINT msgsz = trcv_mbf(ID mbfid, void *msg, TMO tmout)	[T]
16812	ER ercd = ini_mbf(ID mbfid)	[T]
16813	ER ercd = ref_mbf(ID mbfid, T_RMBF *pk_rmbf)	[T]
16814		
16815	スピンロック	
16816		
16817	ER_ID spnid = acre_spn(const T_CSPN *pk_cspn)	[TMD]
16818	ER ercd = sac_spn(ID spnid, const ACVCT *p_acvct)	[TPMD]
16819	ER ercd = del_spn(ID spnid)	[TMD]
16820	ER ercd = loc_spn(ID spnid)	[TM]
16821	ER ercd = iloc_spn(ID spnid)	[IM]
16822	ER ercd = try_spn(ID spnid)	[TM]
16823	ER ercd = itry_spn(ID spnid)	[IM]
16824	ER ercd = unl_spn(ID spnid)	[TM]
16825	ER ercd = iunl_spn(ID spnid)	[IM]
16826	ER ercd = ref_spn(ID spnid, T_RSPN *pk_rspn)	[TM]
16827		
16828	(5) メモリプール管理機能	
16829		
16830	固定長メモリプール	
16831		
16832	ER_ID mpfid = acre_mpf(const T_CMPF *pk_cmpf)	[TD]
16833	ER ercd = sac_mpf(ID mpfid, const ACVCT *p_acvct)	[TPD]
16834	ER ercd = del_mpf(ID mpfid)	[TD]
16835	ER ercd = get_mpf(ID mpfid, void **p_blk)	[T]
16836	ER ercd = pget_mpf(ID mpfid, void **p_blk)	[T]
16837	ER ercd = tget_mpf(ID mpfid, void **p_blk, TMO tmout)	[T]
16838	ER ercd = rel_mpf(ID mpfid, void *blk)	[T]
16839	ER ercd = ini_mpf(ID mpfid)	[T]
16840	ER ercd = ref_mpf(ID mpfid, T_RMPF *pk_rmpf)	[T]
16841		
16842	(6) 時間管理機能	
16843		
16844	システム時刻管理	
16845		
16846	ER ercd = get_tim(SYSTIM *p_system)	[T]
16847	ER ercd = get_utm(SYSUTM *p_sysutm)	[TI]
16848		
16849	周期ハンドラ	
16850		

16851	ER_ID cycid = acre_cyc(const T_CCYC *pk_ccyc)	[TD]
16852	ER ercd = sac_cyc(ID cycid, const ACVCT *p_acvct)	[TPD]
16853	ER ercd = del_cyc(ID cycid)	[TD]
16854	ER ercd = sta_cyc(ID cycid)	[T]
16855	ER ercd = msta_cyc(ID cycid, ID prcid)	[TM]
16856	ER ercd = stp_cyc(ID cycid)	[T]
16857	ER ercd = ref_cyc(ID cycid, T_RCYC *pk_rcyc)	[T]
16858		
16859	アラームハンドラ	
16860		
16861	ER_ID almid = acre_alm(const T_CALM *pk_calm)	[TD]
16862	ER ercd = sac_alm(ID almid, const ACVCT *p_acvct)	[TPD]
16863	ER ercd = del_alm(ID almid)	[TD]
16864	ER ercd = sta_alm(ID almid, RELTIM almtim)	[T]
16865	ER ercd = ista_alm(ID almid, RELTIM almtim)	[I]
16866	ER ercd = msta_alm(ID almid, RELTIM almtim, ID prcid)	[TM]
16867	ER ercd = imsta_alm(ID almid, RELTIM almtim, ID prcid)	[IM]
16868	ER ercd = stp_alm(ID almid)	[T]
16869	ER ercd = istp_alm(ID almid)	[I]
16870	ER ercd = ref_alm(ID almid, T_RALM *pk_ralm)	[T]
16871		
16872	オーバランハンドラ	
16873		
16874	ER ercd = def_ovr(const T_DOVR *pk_dovr)	[TD]
16875	ER ercd = sta_ovr(ID tskid, OVRTIM ovrtime)	[T]
16876	ER ercd = ista_ovr(ID tskid, OVRTIM ovrtime)	[I]
16877	ER ercd = stp_ovr(ID tskid)	[T]
16878	ER ercd = istp_ovr(ID tskid)	[I]
16879	ER ercd = ref_ovr(ID tskid, T_ROVR *pk_rovr)	[T]
16880		
16881	(7) システム状態管理機能	
16882		
16883	ER ercd = sac_sys(const ACVCT *p_acvct)	[TPD]
16884	ER ercd = rot_rdq(PRI tskpri)	[T]
16885	ER ercd = irot_rdq(PRI tskpri)	[I]
16886	ER ercd = mrot_rdq(PRI tskpri, ID prcid)	[TM]
16887	ER ercd = imrot_rdq(PRI tskpri, ID prcid)	[IM]
16888	ER ercd = get_tid(ID *p_tskid)	[T]
16889	ER ercd = iget_tid(ID *p_tskid)	[I]
16890	ER ercd = get_did(ID *p_domid)	[TP]
16891	ER ercd = get_pid(ID *p_prcid)	[TM]
16892	ER ercd = iget_pid(ID *p_prcid)	[IM]
16893	ER ercd = loc_cpu()	[T]
16894	ER ercd = iloc_cpu()	[I]
16895	ER ercd = unl_cpu()	[T]
16896	ER ercd = iunl_cpu()	[I]
16897	ER ercd = dis_dsp()	[T]
16898	ER ercd = ena_dsp()	[T]
16899	bool_t state = sns_ctx()	[TI]
16900	bool_t state = sns_loc()	[TI]

```

16901      bool_t state = sns_dsp() [TI]
16902      bool_t state = sns_dpn() [TI]
16903      bool_t state = sns_ker() [TI]
16904      ER ercd = ext_ker() [TI]
16905      ER ercd = ref_sys(T_RSYS *pk_rsys) [T]
16906
16907 (8) メモリオブジェクト管理機能
16908
16909      ER ercd = att_mem(const T_AMEM *pk_amem) [TPD]
16910      ER ercd = att_pma(const T_AMEM *pk_apma) [TPD]
16911      ER ercd = sac_mem(const void *base, const ACVCT *p_acvct) [TPD]
16912      ER ercd = det_mem(const void *base) [TPD]
16913      ER ercd = prb_mem(const void *base, SIZE size, [TP]
16914                      ID tskid, MODE pmmode)
16915      ER ercd = ref_mem(const void *base, T_RMEM *pk_rmem) [TP]
16916
16917 (9) 割込み管理機能
16918
16919      ER ercd = cfg_int(INTNO intno, const T_CINT *pk_cint) [TD]
16920      ER_ID isrid = acre_isr(const T_CISR *pk_cisr) [TD]
16921      ER ercd = sac_isr(ID isrid, const ACVCT *p_acvct) [TPD]
16922      ER ercd = del_isr(ID isrid) [TD]
16923      ER ercd = ref_isr(ID isrid, T_RISR *pk_risr) [T]
16924      ER ercd = def_inh(INHNO inhno, const T_DINH *pk_dinh) [TD]
16925      ER ercd = dis_int(INTNO intno) [T]
16926      ER ercd = ena_int(INTNO intno) [T]
16927      ER ercd = ref_int(INTNO intno, T_RINT *pk_rint) [T]
16928      ER ercd = chg_ipm(PRI intpri) [T]
16929      ER ercd = get_ipm(PRI *p_intpri) [T]
16930
16931 (10) CPU例外管理機能
16932
16933      ER ercd = def_exc(EXCNO excno, const T_DEXC *pk_dexc) [TD]
16934      bool_t stat = xsns_dpn(void *p_excinf) [TI]
16935      bool_t stat = xsns_xpn(void *p_excinf) [TI]
16936
16937 (11) 拡張サービスコール管理機能
16938
16939      ER ercd = def_svc(FN fncd, const T_DSVC *pk_dsvc) [TPD]
16940      ER_UINT ercd = cal_svc(FN fncd, intptr_t par1, intptr_t par2, [TIP]
16941                          intptr_t par3, intptr_t par4, intptr_t par5)
16942
16943 (12) システム構成管理機能
16944
16945      ER ercd = ref_cfg(T_RCFG *pk_rcfg) [T]
16946      ER ercd = ref_ver(T_RVER *pk_rver) [T]
16947
16948 5.2 静的API一覧
16949
16950 (1) タスク管理機能

```

16951

16952 *保護機能対応でないカーネルの場合

16953 CRE_TSK(ID tskid, { ATR tskatr, intptr_t exinf, TASK task, [S]

16954 PRI itskpri, SIZE stksz, STK_T *stk })

16955

16956 *保護機能対応カーネルの場合

16957 CRE_TSK(ID tskid, { ATR tskatr, intptr_t exinf, TASK task, [SP]

16958 PRI itskpri, SIZE stksz, STK_T *stk,

16959 SIZE sstksz, STK_T *sstk })

16960 ※ sstkszおよびsstkの記述は省略することができる。

16961

16962 AID_TSK(uint_t notsk) [SD]

16963 SAC_TSK(ID tskid, { ACPTN acptn1, ACPTN acptn2, [SP]

16964 ACPTN acptn3, ACPTN acptn4 })

16965 DEF_EPR(ID tskid, { PRI exepr }) [S]

16966

16967 (2) タスク付属同期機能

16968

16969 なし

16970

16971 (3) タスク例外処理機能

16972

16973 DEF_TEX(ID tskid, { ATR texatr, TEXRTN texrtn }) [S]

16974

16975 (4) 同期・通信機能

16976

16977 セマフォ

16978

16979 CRE_SEM(ID semid, { ATR sematr, uint_t isemcnt, uint_t maxsem }) [S]

16980 AID_SEM(uint_t nose) [SD]

16981 SAC_SEM(ID semid, { ACPTN acptn1, ACPTN acptn2, [SP]

16982 ACPTN acptn3, ACPTN acptn4 })

16983

16984 イベントフラグ

16985

16986 CRE_FLG(ID flgid, { ATR flgatr, FLGPTN iflgptn }) [S]

16987 AID_FLG(uint_t noflg) [SD]

16988 SAC_FLG(ID flgid, { ACPTN acptn1, ACPTN acptn2, [SP]

16989 ACPTN acptn3, ACPTN acptn4 })

16990

16991 データキュー

16992

16993 CRE_DTQ(ID dtqid, { ATR dtqatr, uint_t dtqcnt, void *dtqmb }) [S]

16994 AID_DTQ(uint_t nodtq) [SD]

16995 SAC_DTQ(ID dtqid, { ACPTN acptn1, ACPTN acptn2, [SP]

16996 ACPTN acptn3, ACPTN acptn4 })

16997

16998 優先度データキュー

16999

17000 CRE_PDQ(ID pdqid, { ATR pdqatr, uint_t pdqcnt, [S]

17001		PRI maxdpri, void *pdqmb })	
17002	AID_PDQ(uint_t nopdq)		[SD]
17003	SAC_PDQ(ID pdqid, { ACPTN acptn1, ACPTN acptn2,		[SP]
17004	ACPTN acptn3, ACPTN acptn4 })		
17005			
17006	メールボックス		
17007			
17008	CRE_MBX(ID mbxid, { ATR mbxatr, PRI maxmpri, void *mprihd })		[Sp]
17009	AID_MBX(uint_t nombx)		[SpD]
17010			
17011	ミューテックス		
17012			
17013	CRE_MTX(ID mtxid, { ATR mtxatr, PRI ceilpri })		[S]
17014	AID_MTX(uint_t nomtx)		[SD]
17015	SAC_MTX(ID mtxid, { ACPTN acptn1, ACPTN acptn2,		[SP]
17016	ACPTN acptn3, ACPTN acptn4 })		
17017			
17018	メッセージバッファ		
17019			
17020	CRE_MBF(ID mbfid, { ATR mbfatr, uint_t maxmsz,		
17021	SIZE mbfsz, void *mbfmb })		[S]
17022	AID_MBF(uint_t nombf)		[SD]
17023	SAC_MBF(ID mbfid, { ACPTN acptn1, ACPTN acptn2,		[SP]
17024	ACPTN acptn3, ACPTN acptn4 })		
17025			
17026	スピンロック		
17027			
17028	CRE_SPN(ID spnid, { ATR spnatr })		[SM]
17029	AID_SPN(uint_t nospn)		[SMD]
17030	SAC_SPN(ID spnid, { ACPTN acptn1, ACPTN acptn2,		[SPM]
17031	ACPTN acptn3, ACPTN acptn4 })		
17032			
17033	(5) メモリプール管理機能		
17034			
17035	固定長メモリプール		
17036			
17037	CRE_MPF(ID mpfid, { ATR mpfatr, uint_t blkent, uint_t blkksz,		[S]
17038	MPF_T *mpf, void *mpfmb })		
17039	AID_MPF(uint_t nompf)		[SD]
17040	SAC_MPF(ID mpfid, { ACPTN acptn1, ACPTN acptn2,		[SP]
17041	ACPTN acptn3, ACPTN acptn4 })		
17042			
17043	(6) 時間管理機能		
17044			
17045	周期ハンドラ		
17046			
17047	CRE_CYC(ID cycid, { ATR cycatr, intptr_t exinf, CYCHDR cychdr,		[S]
17048	RELTIM cycetim, RELTIM cycphs })		
17049	AID_CYC(uint_t nocyc)		[SD]
17050	SAC_CYC(ID cycid, { ACPTN acptn1, ACPTN acptn2,		[SP]

```

17051                                     ACPTN acptn3, ACPTN acptn4 })
17052
17053 アラームハンドラ
17054
17055     CRE_ALM(ID almid, { ATR almatr, intptr_t exinf, ALMHDR almhdr }) [S]
17056     AID_ALM(uint_t noalm) [SD]
17057     SAC_ALM(ID almid, { ACPTN acptn1, ACPTN acptn2, [SP]
17058                                     ACPTN acptn3, ACPTN acptn4 })
17059
17060 オーバランハンドラ
17061
17062     DEF_OVR({ ATR ovratr, OVRHDR ovrrhdr }) [S]
17063
17064 (7) システム状態管理機能
17065
17066     SAC_SYS({ ACPTN acptn1, ACPTN acptn2, [SP]
17067                                     ACPTN acptn3, ACPTN acptn4 })
17068
17069 (8) メモリオブジェクト管理機能
17070
17071     ATT_REG("メモリリージョン名", [SP]
17072                                     { ATR regatr, void *base, SIZE size })
17073     DEF_SRG("標準ROMリージョン名", "標準RAMリージョン名") [SP]
17074     ATT_SEC("セクション名", { ATR mematr, "メモリリージョン名" }) [SP]
17075     ATA_SEC("セクション名", { ATR mematr, "メモリリージョン名" }, [SP]
17076                                     { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
17077     LNK_SEC("セクション名", { "メモリリージョン名" }) [SP]
17078     ATT_MOD("オブジェクトモジュール名") [SP]
17079     ATA_MOD("オブジェクトモジュール名", [SP]
17080                                     { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
17081     ATT_MEM({ ATR mematr, void *base, SIZE size }) [SP]
17082     ATA_MEM({ ATR mematr, void *base, SIZE size }, [SP]
17083                                     { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
17084     ATT_PMA({ ATR mematr, void *base, SIZE size, void *paddr }) [SP]
17085     ATA_PMA({ ATR mematr, void *base, SIZE size, void *paddr }, [SP]
17086                                     { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
17087
17088 (9) 割込み管理機能
17089
17090     CFG_INT(INTNO intno, { ATR intatr, PRI intpri }) [S]
17091     CRE_ISR(ID isrid, { ATR isratr, intptr_t exinf, [S]
17092                                     INTNO intno, ISR isr, PRI isrpri })
17093     ATT_ISR({ ATR isratr, intptr_t exinf, [S]
17094                                     INTNO intno, ISR isr, PRI isrpri })
17095     AID_ISR(uint_t noisr) [SD]
17096     SAC_ISR(ID isrid, { ACPTN acptn1, ACPTN acptn2, [SP]
17097                                     ACPTN acptn3, ACPTN acptn4 })
17098     DEF_INH(INHNO inhno, { ATR inhatr, INTHDR inthdr }) [S]
17099
17100 (10) CPU例外管理機能

```

```

17101
17102     DEF_EXC(EXCNO excno, { ATR excatr, EXCHDR exchdr })           [S]
17103
17104 (11) 拡張サービスコール管理機能
17105
17106     DEF_SVC(FN fncd, { ATR svcatr, EXTSVC svcrtm, SIZE stksz })   [SP]
17107
17108 (12) システム構成管理機能
17109
17110     LMT_DOM({ PRI mintpri })                                       [SP]
17111     DEF_ICS({ SIZE istksz, STK_T *istk })                         [S]
17112     DEF_STK({ SIZE stksz, STK_T *stk })                           [S]
17113     ATT_INI({ ATR iniatr, intptr_t exinf, INIRTN inirtn })       [S]
17114     ATT_TER({ ATR teratr, intptr_t exinf, TERRTN terrtn })       [S]
17115
17116 5.3 データ型
17117
17118 5.3.1 TOPPERS共通データ型
17119
17120     int8_t      符号付き8ビット整数 (オプション, C99準拠)
17121     uint8_t     符号無し8ビット整数 (オプション, C99準拠)
17122     int16_t     符号付き16ビット整数 (C99準拠)
17123     uint16_t    符号無し16ビット整数 (C99準拠)
17124     int32_t     符号付き32ビット整数 (C99準拠)
17125     uint32_t    符号無し32ビット整数 (C99準拠)
17126     int64_t     符号付き64ビット整数 (オプション, C99準拠)
17127     uint64_t    符号無し64ビット整数 (オプション, C99準拠)
17128     int128_t    符号付き128ビット整数 (オプション, C99準拠)
17129     uint128_t   符号無し128ビット整数 (オプション, C99準拠)
17130
17131     int_least8_t 8ビット以上の符号付き整数 (C99準拠)
17132     uint_least8_t int_least8_t型と同じサイズの符号無し整数 (C99準拠)
17133
17134     float32_t   IEEE754準拠の32ビット単精度浮動小数点数 (オプション)
17135     double64_t  IEEE754準拠の64ビット倍精度浮動小数点数 (オプション)
17136
17137     bool_t      真偽値 (trueまたはfalse)
17138     int_t       16ビット以上の符号付き整数
17139     uint_t      int_t型と同じサイズの符号無し整数
17140     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
17141     ulong_t     long_t型と同じサイズの符号無し整数
17142
17143     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)
17144     uintptr_t   intptr_t型と同じサイズの符号無し整数 (C99準拠)
17145
17146     FN          機能コード (符号付き整数, int_tに定義)
17147     ER          エラーコード (符号付き整数, int_tに定義)
17148     ID          オブジェクトのID番号 (符号付き整数, int_tに定義)
17149     ATR         オブジェクト属性 (符号無し整数, uint_tに定義)
17150     STAT        オブジェクトの状態 (符号無し整数, uint_tに定義)

```

17151	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)
17152	PRI	優先度 (符号付き整数, int_tに定義)
17153	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)
17155		
17156	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)
17157	RELTIM	相対時間 (符号無し整数, 単位はミリ秒, uint_tに定義)
17158	SYSTIM	システム時刻 (符号無し整数, 単位はミリ秒, ulong_tに定義)
17159	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)
17160		
17161		
17162	FP	プログラムの起動番地 (型の定まらない関数ポインタ)
17163		
17164	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)
17165	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)
17166		
17167	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)
17168		
17169		
17170		
17171	MB_T	オブジェクト管理領域を確保するためのデータ型
17172		
17173	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)
17174		
17175		
17176	typedef struct acvct { /* アクセス許可ベクタ */	
17177	ACPTN	acptn1; /* 通常操作1のアクセス許可パターン */
17178	ACPTN	acptn2; /* 通常操作2のアクセス許可パターン */
17179	ACPTN	acptn3; /* 管理操作のアクセス許可パターン */
17180	ACPTN	acptn4; /* 参照操作のアクセス許可パターン */
17181	} ACVCT;	
17182		

5.3.2 カーネルの使用するデータ型

17184		
17185	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)
17186	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)
17187	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)
17188	INTNO	割込み番号 (符号無し整数, uint_tに定義)
17189	INHNO	割込みハンドラ番号 (符号無し整数, uint_tに定義)
17190	EXCNO	CPU例外ハンドラ番号 (符号無し整数, uint_tに定義)
17191		
17192	TASK	タスクのメインルーチン (関数ポインタ)
17193	TEXRTN	タスク例外処理ルーチン (関数ポインタ)
17194	CYCHDR	周期ハンドラ (関数ポインタ)
17195	ALMHDR	アラームハンドラ (関数ポインタ)
17196	OVHRDR	オーバランハンドラ (関数ポインタ)
17197	ISR	割込みサービスルーチン (関数ポインタ)
17198	INTHDR	割込みハンドラ (関数ポインタ)
17199	EXCHDR	CPU例外ハンドラ (関数ポインタ)
17200	EXTSVC	拡張サービスコール (関数ポインタ)

17201	INIRTN	初期化ルーチン（関数ポインタ）
17202	TERRTN	終了処理ルーチン（関数ポインタ）
17203		
17204	STK_T	スタック領域を確保するためのデータ型
17205	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

17206

17207 メールボックスのメッセージヘッダ【NGKI4001】

```
17208
17209     typedef struct t_msg {
17210         struct t_msg    *pk_next;
17211     } T_MSG;
```

17212

17213 メールボックスの優先度付きメッセージヘッダ【NGKI4002】

```
17214
17215     typedef struct t_msg_pri {
17216         T_MSG          msgque;          /* メールボックスのメッセージヘッダ */
17217         PRI             msgpri;         /* メッセージ優先度 */
17218     } T_MSG_PRI;
```

17219

17220 5.3.3 カーネルの使用するパケット形式

17221

17222 (1) タスク管理機能

17223

17224 タスクの生成情報のパケット形式【NGKI4003】

```
17225
17226     typedef struct t_ctsk {
17227         ATR          tskatr;          /* タスク属性 */
17228         intptr_t     exinf;          /* タスクの拡張情報 */
17229         TASK         task;          /* タスクのメインルーチンの先頭番地 */
17230         PRI          itskpri;        /* タスクの起動時優先度 */
17231         SIZE         stksz;          /* タスクのスタック領域のサイズ */
17232         STK_T *      stk;          /* タスクのスタック領域の先頭番地 */
17233         /* 以下は、保護機能対応カーネルの場合 */
17234         SIZE         sstksz;        /* タスクのシステムスタック領域のサイズ */
17235         STK_T *      sstk;          /* タスクのシステムスタック領域の先頭番地 */
17236     } T_CTSK;
```

17237

17238 タスクの現在状態のパケット形式【NGKI4004】

```
17239
17240     typedef struct t_rtsk {
17241         STAT         tskstat;        /* タスク状態 */
17242         PRI          tskpri;        /* タスクの現在優先度 */
17243         PRI          tskbpri;       /* タスクのベース優先度 */
17244         STAT         tskwait;       /* 待ち要因 */
17245         ID           wobjid;        /* 待ち対象のオブジェクトのID */
17246         TMO          lefttmo;       /* タイムアウトするまでの時間 */
17247         uint_t       actcnt;        /* 起動要求キューイング数 */
17248         uint_t       wupcnt;        /* 起床要求キューイング数 */
17249         /* 以下は、保護機能対応カーネルの場合 */
17250         bool_t       texmsk;        /* タスク例外マスク状態か否か */
```

```

17251         bool_t      waifbd;    /* 待ち禁止状態か否か */
17252         uint_t      svclevel;   /* 拡張サービスコールのネストレベル */
17253         /* 以下は、マルチプロセッサ対応カーネルの場合 */
17254         ID          pccid;      /* 割付けプロセッサのID */
17255         ID          actprc      /* 次の起動時の割付けプロセッサのID */
17256     } T_RTsk;
17257

```

17258 (2) タスク付属同期機能

17259

17260 なし

17261

17262 (3) タスク例外処理機能

17263

17264 タスク例外処理ルーチンの定義情報のパケット形式【NGKI4005】

17265

```

17266     typedef struct t_dtex {
17267         ATR          texatr;     /* タスク例外処理ルーチン属性 */
17268         TEXRTN       texrtn;     /* タスク例外処理ルーチンの先頭番地 */
17269     } T_DTEX;
17270

```

17271 タスク例外処理の現在状態のパケット形式【NGKI4006】

17272

```

17273     typedef struct t_rtex {
17274         STAT         texstat;    /* タスク例外処理の状態 */
17275         TEXPTN       pndptn;     /* 保留例外要因 */
17276     } T_RTEx;
17277

```

17278 (4) 同期・通信機能

17279

17280 セマフォの生成情報のパケット形式【NGKI4007】

17281

```

17282     typedef struct t_csem {
17283         ATR          sematr;     /* セマフォ属性 */
17284         uint_t       isemcnt;    /* セマフォの初期資源数 */
17285         uint_t       maxsem;     /* セマフォの最大資源数 */
17286     } T_CSEM;
17287

```

17288 セマフォの現在状態のパケット形式【NGKI4008】

17289

```

17290     typedef struct t_rsem {
17291         ID          wtskid;      /* セマフォの待ち行列の先頭のタスクのID番号 */
17292         uint_t       semcnt;     /* セマフォの資源数 */
17293     } T_RSEM;
17294

```

17295 イベントフラグの生成情報のパケット形式【NGKI4009】

17296

```

17297     typedef struct t_cflg {
17298         ATR          flgatr;     /* イベントフラグ属性 */
17299         FLGPTN       iflgptn;   /* イベントフラグの初期ビットパターン */
17300     } T_CFLG;

```

17301

17302 イベントフラグの現在状態のパケット形式【NGKI4010】

17303

```
17304     typedef struct t_rflg {
17305         ID          wtskid;      /* イベントフラグの待ち行列の先頭のタス
17306                                   クのID番号 */
17307         FLGPTN      flgpntn;    /* イベントフラグのビットパターン */
17308     } T_RFLG;
```

17309

17310 データキューの生成情報のパケット形式【NGKI4011】

17311

```
17312     typedef struct t_cdtq {
17313         ATR          dtqatr;     /* データキュー属性 */
17314         uint_t       dtqcnt;     /* データキュー管理領域に格納できるデータ数 */
17315         void *       dtqmb;     /* データキュー管理領域の先頭番地 */
17316     } T_CDTQ;
```

17317

17318 データキューの現在状態のパケット形式【NGKI4012】

17319

```
17320     typedef struct t_rdtq {
17321         ID          stskid;     /* データキューの送信待ち行列の先頭のタ
17322                                   スクのID番号 */
17323         ID          rtskid;     /* データキューの受信待ち行列の先頭のタ
17324                                   スクのID番号 */
17325         uint_t       sdtqcnt;   /* データキュー管理領域に格納されている
17326                                   データの数 */
17327     } T_RDTQ;
```

17328

17329 優先度データキューの生成情報のパケット形式【NGKI4013】

17330

```
17331     typedef struct t_cpdq {
17332         ATR          pdqatr;     /* 優先度データキュー属性 */
17333         uint_t       pdqcnt;     /* 優先度データキュー管理領域に格納でき
17334                                   るデータ数 */
17335         PRI          maxdpri;    /* 優先度データキューに送信できるデータ
17336                                   優先度の最大値 */
17337         void *       pdqmb;     /* 優先度データキュー管理領域の先頭番地 */
17338     } T_CPDQ;
```

17339

17340 優先度データキューの現在状態のパケット形式【NGKI4014】

17341

```
17342     typedef struct t_rpdq {
17343         ID          stskid;     /* 優先度データキューの送信待ち行列の先
17344                                   頭のタスクのID番号 */
17345         ID          rtskid;     /* 優先度データキューの受信待ち行列の先
17346                                   頭のタスクのID番号 */
17347         uint_t       spdqcnt;   /* 優先度データキュー管理領域に格納され
17348                                   ているデータの数 */
17349     } T_RPDQ;
```

17350

17351 メールボックスの生成情報のパケット形式【NGKI4015】

17352

```
17353     typedef struct t_cmbx {
17354         ATR          mbxatr;      /* メールボックス属性 */
17355         PRI          maxmpri;     /* 優先度メールボックスに送信できるメッ
17356                                     セージ優先度の最大値 */
17357         void *       mprihd;      /* 優先度別のメッセージキューヘッダ領域
17358                                     の先頭番地 */
17359     } T_CMBX;
```

17360

17361 メールボックスの現在状態のパケット形式【NGKI4016】

17362

```
17363     typedef struct t_rmbx {
17364         ID           wtskid;      /* メールボックスの待ち行列の先頭のタスク
17365                                     のID番号 */
17366         T_MSG        *pk_msg;     /* メッセージキューの先頭につながれたメッ
17367                                     セージの先頭番地 */
17368     } T_RMBX;
```

17369

17370 ミューテックスの生成情報のパケット形式【NGKI4017】

17371

```
17372     typedef struct t_cmtx {
17373         ATR          mtxatr;      /* ミューテックス属性 */
17374         PRI          ceilpri;     /* ミューテックスの上限優先度 */
17375     } T_CMTX;
```

17376

17377 ミューテックスの現在状態のパケット形式【NGKI4018】

17378

```
17379     typedef struct t_rmtx {
17380         ID           htiskid;     /* ミューテックスをロックしているタス
17381                                     クのID番号 */
17382         ID           wtskid;      /* ミューテックスの待ち行列の先頭のタ
17383                                     スクのID番号 */
17384     } T_RMTX;
```

17385

17386 メッセージバッファの生成情報のパケット形式【NGKI4037】

17387

```
17388     typedef struct t_cmbf {
17389         ATR          mbfatr;      /* メッセージバッファ属性 */
17390         uint_t        maxmsz;     /* メッセージバッファの最大メッセージ
17391                                     サイズ (バイト数) */
17392         SIZE          mbfsz;      /* メッセージバッファ管理領域のサイズ
17393                                     (バイト数) */
17394         void *       mbfmb;      /* メッセージバッファ管理領域の先頭番地 */
17395     } T_CMBF;
```

17396

17397 メッセージバッファの現在状態のパケット形式【NGKI4038】

17398

```
17399     typedef struct t_rmbf {
17400         ID           stskid;      /* メッセージバッファの送信待ち行列の先頭の
```

```

17401                                     タスクのID番号 */
17402         ID      rtskid;      /* メッセージバッファの受信待ち行列の先頭の
17403                                     タスクのID番号 */
17404         uint_t   smbfcnt;     /* メッセージバッファ管理領域に格納されてい
17405                                     るメッセージの数 */
17406         SIZE     fmbfsz;     /* メッセージバッファ管理領域中の空き領域の
17407                                     サイズ */
17408     } T_RMBF;
17409
17410     スピンロックの生成情報のパケット形式【NGKI4019】
17411
17412     typedef struct t_cspn {
17413         ATR      spnatr;      /* スピンロック属性 */
17414     } T_CSPN;
17415
17416     スピンロックの現在状態のパケット形式【NGKI4020】
17417
17418     typedef struct t_rspn {
17419         STAT      spnstat     /* スピンロックのロック状態 */
17420     } T_RSPN;
17421
17422     (5) メモリプール管理機能
17423
17424     固定長メモリアプールの生成情報のパケット形式【NGKI4021】
17425
17426     typedef struct t_cmpf {
17427         ATR      mpfatr;      /* 固定長メモリアプール属性 */
17428         uint_t   blkcnt;      /* 獲得できる固定長メモリアブロックの数 */
17429         uint_t   blkksz;      /* 固定長メモリアブロックのサイズ */
17430         MPF_T *   mpf;        /* 固定長メモリアプール領域の先頭番地 */
17431         void *    mpfmb;      /* 固定長メモリアプール管理領域の先頭番地 */
17432     } T_CMPF;
17433
17434     固定長メモリアプールの現在状態のパケット形式【NGKI4022】
17435
17436     typedef struct t_rmpf {
17437         ID      wtskid;      /* 固定長メモリアプールの待ち行列の先頭の
17438                                     タスクのID番号 */
17439         uint_t   fblkcnt;     /* 固定長メモリアプール領域の空きメモリア領
17440                                     域に割り付けることができる固定長メモ
17441                                     リアブロックの数 */
17442     } T_RMPF;
17443
17444     (6) 時間管理機能
17445
17446     周期ハンドラの生成情報のパケット形式【NGKI4023】
17447
17448     typedef struct t_ccyc {
17449         ATR      cycatr;      /* 周期ハンドラ属性 */
17450         intptr_t exinf;      /* 周期ハンドラの拡張情報 */

```

```

17451          CYCHDR      cychdr;      /* 周期ハンドラの先頭番地 */
17452          RELTIM      cycstim;      /* 周期ハンドラの起動周期 */
17453          RELTIM      cycphs;      /* 周期ハンドラの起動位相 */
17454      } T_CCYC;

```

17455

17456 周期ハンドラの現在状態のパケット形式【NGKI4024】

17457

```

17458      typedef struct t_rcyc {
17459          STAT          cycstat;      /* 周期ハンドラの動作状態 */
17460          RELTIM      lefttim;      /* 次に周期ハンドラを起動する時刻までの
17461                                  相対時間 */
17462          /* 以下は、マルチプロセッサ対応カーネルの場合 */
17463          ID            prcid;      /* 割付けプロセッサのID */
17464      } T_RCYC;

```

17465

17466 アラームハンドラの生成情報のパケット形式【NGKI4025】

17467

```

17468      typedef struct t_calm {
17469          ATR          almatr;      /* アラームハンドラ属性 */
17470          intptr_t     exinf;      /* アラームハンドラの拡張情報 */
17471          ALMHDR      almhdr;      /* アラームハンドラ先頭番地 */
17472      } T_CALM;

```

17473

17474 アラームハンドラの現在状態のパケット形式【NGKI4026】

17475

```

17476      typedef struct t_ralm {
17477          STAT          almstat;      /* アラームハンドラの動作状態 */
17478          RELTIM      lefttim;      /* アラームハンドラを起動する時刻までの
17479                                  相対時間 */
17480          /* 以下は、マルチプロセッサ対応カーネルの場合 */
17481          ID            prcid;      /* 割付けプロセッサのID */
17482      } T_RALM;

```

17483

17484 オーバランハンドラの定義情報のパケット形式【NGKI4027】

17485

```

17486      typedef struct t_dovr {
17487          ATR          ovratr;      /* オーバランハンドラ属性 */
17488          OVRHDR      ovrhdr;      /* オーバランハンドラ先頭番地 */
17489      } T_DOVR;

```

17490

17491 オーバランハンドラの現在状態のパケット形式【NGKI4028】

17492

```

17493      typedef struct t_rovr {
17494          STAT          ovrstat;      /* オーバランハンドラの動作状態 */
17495          OVRTIM      leftotm;      /* 残りプロセッサ時間 */
17496      } T_OVR;

```

17497

17498 (7) システム状態管理機能

17499

17500 システムの現在状態のパケット形式

```

17501
17502 ☆未完成
17503
17504 (8) メモリオブジェクト管理機能
17505
17506 メモリオブジェクトの登録情報のパケット形式【NGKI4029】
17507
17508     typedef struct t_amem {
17509         ATR          mematr      /* メモリオブジェクト属性 */
17510         void *       base        /* 登録するメモリ領域の先頭番地 */
17511         SIZE         size        /* 登録するメモリ領域のサイズ (バイト数) */
17512     } T_AMEM;
17513
17514 物理メモリ領域の登録情報のパケット形式【NGKI4030】
17515
17516     typedef struct t_apma {
17517         ATR          mematr      /* メモリオブジェクト属性 */
17518         void *       base        /* 登録するメモリ領域の先頭番地 */
17519         SIZE         size        /* 登録するメモリ領域のサイズ (バイト数) */
17520         void *       paddr       /* 登録するメモリ領域の物理アドレスの先頭
17521                                番地 */
17522     } T_APMA;
17523
17524 メモリオブジェクトの現在状態のパケット形式
17525
17526 ☆未完成
17527
17528 (9) 割込み管理機能
17529
17530 割込み要求ラインの属性の設定情報のパケット形式【NGKI4031】
17531
17532     typedef struct t_cint {
17533         ATR          intatr;      /* 割込み要求ライン属性 */
17534         PRI          intpri;      /* 割込み優先度 */
17535     } T_CINT;
17536
17537 割込みサービスルーチンの生成情報のパケット形式【NGKI4032】
17538
17539     typedef struct t_cisr {
17540         ATR          isratr;      /* 割込みサービスルーチン属性 */
17541         intptr_t     exinf;        /* 割込みサービスルーチンの拡張情報 */
17542         INTNO        intno;        /* 割込みサービスルーチンを登録する割込
17543                                み番号 */
17544         ISR          isr;          /* 割込みサービスルーチンの先頭番地 */
17545         PRI          isrpri;       /* 割込みサービスルーチン優先度 */
17546     } T_CISR;
17547
17548 割込みサービスルーチンの現在状態のパケット形式
17549
17550 ☆未完成

```

17551
17552
17553
17554
17555
17556
17557
17558
17559
17560
17561
17562
17563
17564
17565
17566
17567
17568
17569
17570
17571
17572
17573
17574
17575
17576
17577
17578
17579
17580
17581
17582
17583
17584
17585
17586
17587
17588
17589
17590
17591
17592
17593
17594
17595
17596
17597
17598
17599
17600

割込みハンドラの定義情報のパケット形式【NGKI4033】

```
typedef struct t_dinh {  
    ATR        inhatr;    /* 割込みハンドラ属性 */  
    INTHDR     inthdr;    /* 割込みハンドラの先頭番地 */  
} T_DINH;
```

割込み要求ラインの現在状態のパケット形式

☆未完成

(10) CPU例外管理機能

CPU例外ハンドラの定義情報のパケット形式【NGKI4034】

```
typedef struct t_dexc {  
    ATR        excatr;    /* CPU例外ハンドラ属性 */  
    EXCHDR     exchdr;    /* CPU例外ハンドラの先頭番地 */  
} T_DEXC;
```

(11) 拡張サービスコール管理機能

拡張サービスコールの定義情報のパケット形式【NGKI4035】

```
typedef struct t_dsvc {  
    ATR        svcatr     /* 拡張サービスコール属性 */  
    EXT SVC    svcrtm     /* 拡張サービスコールの先頭番地 */  
    SIZE       stksz      /* 拡張サービスコールで使用するスタック  
                           サイズ */  
} T_DSVC;
```

(12) システム構成管理機能

コンフィギュレーション情報のパケット形式

☆未完成

バージョン情報のパケット形式

☆未完成

5.4 定数とマクロ

5.4.1 TOPPERS共通定数

(1) 一般定数

NULL 無効ポインタ

17601	true	1	真
17602	false	0	偽
17603			
17604	E_OK	0	正常終了
17605			
17606	(2) 整数型に格納できる最大値と最小値		
17607			
17608	INT8_MAX	int8_tに格納できる最大値 (オプション, C99準拠)	
17609	INT8_MIN	int8_tに格納できる最小値 (オプション, C99準拠)	
17610	UINT8_MAX	uint8_tに格納できる最大値 (オプション, C99準拠)	
17611	INT16_MAX	int16_tに格納できる最大値 (C99準拠)	
17612	INT16_MIN	int16_tに格納できる最小値 (C99準拠)	
17613	UINT16_MAX	uint16_tに格納できる最大値 (C99準拠)	
17614	INT32_MAX	int32_tに格納できる最大値 (C99準拠)	
17615	INT32_MIN	int32_tに格納できる最小値 (C99準拠)	
17616	UINT32_MAX	uint32_tに格納できる最大値 (C99準拠)	
17617	INT64_MAX	int64_tに格納できる最大値 (オプション, C99準拠)	
17618	INT64_MIN	int64_tに格納できる最小値 (オプション, C99準拠)	
17619	UINT64_MAX	uint64_tに格納できる最大値 (オプション, C99準拠)	
17620	INT128_MAX	int128_tに格納できる最大値 (オプション, C99準拠)	
17621	INT128_MIN	int128_tに格納できる最小値 (オプション, C99準拠)	
17622	UINT128_MAX	uint128_tに格納できる最大値 (オプション, C99準拠)	
17623			
17624	INT_LEAST8_MAX	int_least8_tに格納できる最大値 (C99準拠)	
17625	INT_LEAST8_MIN	int_least8_tに格納できる最小値 (C99準拠)	
17626	UINT_LEAST8_MAX	uint_least8_tに格納できる最大値 (C99準拠)	
17627	INT_MAX	int_tに格納できる最大値 (C90準拠)	
17628	INT_MIN	int_tに格納できる最小値 (C90準拠)	
17629	UINT_MAX	uint_tに格納できる最大値 (C90準拠)	
17630	LONG_MAX	long_tに格納できる最大値 (C90準拠)	
17631	LONG_MIN	long_tに格納できる最小値 (C90準拠)	
17632	ULONG_MAX	ulong_tに格納できる最大値 (C90準拠)	
17633			
17634	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)	
17635			
17636	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)	
17637			
17638	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)	
17639			
17640	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)	
17641			
17642			
17643	(3) 整数型のビット数		
17644			
17645	CHAR_BIT	char型のビット数 (C90準拠)	
17646			
17647	(4) オブジェクト属性		
17648			
17649	TA_NULL	0U	オブジェクト属性を指定しない
17650			

17651 (5) タイムアウト指定

17652

17653 TMO_POL 0 ポーリング

17654 TMO_FEVR -1 永久待ち

17655 TMO_NBLK -2 ノンブロッキング

17656

17657 (6) アクセス許可パターン

17658

17659 TACP_KERNEL OU カーネルドメインのみにアクセスを許可

17660 TACP_SHARED ~OU すべての保護ドメインにアクセスを許可

17661

17662 5.4.2 TOPPERS共通マクロ

17663

17664 (1) 整数定数を作るマクロ

17665

17666 INT8_C(val) int_least8_t型の定数を作るマクロ (C99準拠)

17667 UINT8_C(val) uint_least8_t型の定数を作るマクロ (C99準拠)

17668 INT16_C(val) int16_t型の定数を作るマクロ (C99準拠)

17669 UINT16_C(val) uint16_t型の定数を作るマクロ (C99準拠)

17670 INT32_C(val) int32_t型の定数を作るマクロ (C99準拠)

17671 UINT32_C(val) uint32_t型の定数を作るマクロ (C99準拠)

17672 INT64_C(val) int64_t型の定数を作るマクロ (オプション, C99準拠)

17673 UINT64_C(val) uint64_t型の定数を作るマクロ (オプション, C99準拠)

17674 INT128_C(val) int128_t型の定数を作るマクロ (オプション, C99準拠)

17675 UINT128_C(val) uint128_t型の定数を作るマクロ (オプション, C99準拠)

17676

17677 UINT_C(val) uint_t型の定数を作るマクロ

17678 ULONG_C(val) ulong_t型の定数を作るマクロ

17679

17680 (2) 型に関する情報を取り出すためのマクロ

17681

17682 offsetof(structure, field) 構造体structure中のフィールドfieldの
17683 バイト位置を返すマクロ (C90準拠)

17684

17685 alignof(type) 型typeのアラインメント単位を返すマクロ

17686

17687 ALIGN_TYPE(addr, type) 番地addrが型typeに対してアラインしてい
17688 るかどうかを返すマクロ

17689

17690 (3) assertマクロ

17691

17692 assert(exp) expが成立しているかを検査するマクロ (C90準拠)

17693

17694 (4) コンパイラの拡張機能のためのマクロ

17695

17696 inline インライン関数

17697 Inline ファイルローカルなインライン関数

17698 asm インラインアセンブラ

17699 Asm インラインアセンブラ (最適化抑止)

17700 throw() 例外を発生しない関数

17701	NoReturn		リターンしない関数
17702			
17703	(5) エラーコード生成・分解マクロ		
17704			
17705	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
17706			
17707			
17708	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
17709			
17710	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
17711			
17712			
17713	(6) アクセス許可パターン生成マクロ		
17714			
17715	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
17716			
17717			
17718	5.4.3 カーネル共通定数		
17719			
17720	(1) オブジェクト属性		
17721			
17722	TA_TPRI	0x01U	タスクの待ち行列をタスクの優先度順に
17723			
17724	(2) 保護ドメインID		
17725			
17726	TDOM_SELF	0	自タスクの属する保護ドメイン
17727	TDOM_KERNEL	-1	カーネルドメイン
17728	TDOM_NONE	-2	無所属 (保護ドメインに属さない)
17729			
17730	(3) その他のカーネル共通定数		
17731			
17732	TCLS_SELF	0	自タスクの属するクラス
17733			
17734	TPRC_NONE	0	割付けプロセッサの指定がない
17735	TPRC_INI	0	初期割付けプロセッサ
17736			
17737	TSK_SELF	0	自タスク指定
17738	TSK_NONE	0	該当するタスクがない
17739			
17740	TPRI_SELF	0	自タスクのベース優先度の指定
17741	TPRI_INI	0	タスクの起動時優先度の指定
17742			
17743	TIPM_ENAALL	0	割込み優先度マスク全解除
17744			
17745	5.4.4 カーネル共通マクロ		
17746			
17747	(1) オブジェクト属性を作るマクロ		
17748			
17749	TA_DOM(domid)		domidで指定される保護ドメインに属する
17750	TA_CLS(clsid)		clsidで指定されるクラスに属する

17751			
17752	(2)	サービスコールの呼出し方法を指定するマクロ	
17753			
17754	SVC_CALL(svc)	svc	で指定されるサービスコールを関数呼出しによって呼び出すための名称
17755			
17756			
17757	5.4.5	カーネルの機能毎の定数	
17758			
17759	(1)	タスク管理機能	
17760			
17761	TA_ACT	0x02U	タスクの生成時にタスクを起動する
17762	TA_RSTR	0x04U	生成するタスクを制約タスクとする
17763	TA_FPU		FPUレジスタをコンテキストに含める
17764			
17765	TTS_RUN	0x01U	実行状態
17766	TTS_RDY	0x02U	実行可能状態
17767	TTS_WAI	0x04U	待ち状態
17768	TTS_SUS	0x08U	強制待ち状態
17769	TTS_WAS	0x0cU	二重待ち状態
17770	TTS_DMT	0x10U	休止状態
17771			
17772	TTW_SLP	0x0001U	起床待ち
17773	TTW_DLY	0x0002U	時間経過待ち
17774	TTW_SEM	0x0004U	セマフォの資源獲得待ち
17775	TTW_FLG	0x0008U	イベントフラグ待ち
17776	TTW_SDTQ	0x0010U	データキューへの送信待ち
17777	TTW_RDTQ	0x0020U	データキューからの受信待ち
17778	TTW_SPDQ	0x0100U	優先度データキューへの送信待ち
17779	TTW_RPDQ	0x0200U	優先度データキューからの受信待ち
17780	TTW_MBX	0x0040U	メールボックスからの受信待ち
17781	TTW_MTX	0x0080U	ミューテックスのロック待ち状態
17782	TTW_SMBF	0x0400U	メッセージバッファへの送信待ち
17783	TTW_RMBF	0x0800U	メッセージバッファからの受信待ち
17784	TTW_MPF	0x2000U	固定長メモリブロックの獲得待ち
17785			
17786	TA_FPU	の値は、ターゲット定義とする.	
17787			
17788	(3)	タスク例外処理機能	
17789			
17790	TTEX_ENA	0x01U	タスク例外処理許可状態
17791	TTEX_DIS	0x02U	タスク例外処理禁止状態
17792			
17793	(4)	同期・通信機能	
17794			
17795	イベントフラグ		
17796			
17797	TA_WMUL	0x02U	複数のタスクが待つのを許す
17798	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
17799			
17800	TWF_ORW	0x01U	イベントフラグのOR待ちモード

17801	TWF_ANDW	0x02U	イベントフラグのAND待ちモード
17802			
17803	メールボックス		
17804			
17805	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
17806			
17807	スピンロック		
17808			
17809	TSPN_UNL	0x01U	取得されていない状態
17810	TSPN_LOC	0x02U	取得されている状態
17811			
17812	(6) 時間管理機能		
17813			
17814	周期ハンドラ		
17815			
17816	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
17817	TA_PHS	0x04U	周期ハンドラを生成した時刻を基準時刻とする
17818			
17819	TCYC_STP	0x01U	周期ハンドラが動作していない状態
17820	TCYC_STA	0x02U	周期ハンドラが動作している状態
17821			
17822	アラームハンドラ		
17823			
17824	TALM_STP	0x01U	アラームハンドラが動作していない状態
17825	TALM_STA	0x02U	アラームハンドラが動作している状態
17826			
17827	オーバランハンドラ		
17828			
17829	TOVR_STP	0x01U	オーバランハンドラが動作していない状態
17830	TOVR_STA	0x02U	オーバランハンドラが動作している状態
17831			
17832	(8) メモリオブジェクト管理機能		
17833			
17834	TA_NOWRITE	0x01U	書込みアクセス禁止
17835	TA_NOREAD	0x02U	読出しアクセス禁止
17836	TA_EXEC	0x04U	実行アクセス許可
17837	TA_MEMINI	0x08U	メモリの初期化を行う
17838	TA_MEMPRSV	0x10U	メモリの初期化を行わない
17839	TA_SDATA	0x20U	ショートデータ領域に配置
17840	TA_UNCACHE	0x40U	キャッシュ禁止
17841	TA_IODEV	0x80U	周辺デバイスの領域
17842	TA_WTHROUGH		ライトスルーキャッシュを用いる
17843			
17844	TPM_WRITE	0x01U	書込みアクセス権のチェック
17845	TPM_READ	0x02U	読出しアクセス権のチェック
17846	TPM_EXEC	0x04U	実行アクセス権のチェック
17847			
17848	TA_WTHROUGHの値は、ターゲット定義とする。		
17849			
17850	(9) 割り込み管理機能		

17851			
17852	TA_ENAINT	0x01U	割込み要求禁止フラグをクリア
17853	TA_EDGE	0x02U	エッジトリガ
17854	TA_POSEDGE		ポジティブエッジトリガ
17855	TA_NEGEDGE		ネガティブエッジトリガ
17856	TA_BOTHEDGE		両エッジトリガ
17857	TA_LOWLEVEL		ローレベルトリガ
17858	TA_HIGHLEVEL		ハイレベルトリガ
17859			
17860	TA_NONKERNEL	0x02U	カーネル管理外の割込み
17861			
17862	TA_POSEDGE, TA_NEGEDGE, TA_BOTHEDGE, TA_LOWLEVEL, TA_HIGHLEVELの値は、		
17863	ターゲット定義とする.		
17864			
17865	(10) CPU例外管理機能		
17866			
17867	TA_DIRECT		CPU例外ハンドラを直接呼び出す
17868			
17869	TA_DIRECTの値は、ターゲット定義とする.		
17870			
17871	5.4.6 カーネルの機能毎のマクロ		
17872			
17873	(1) タスク管理機能		
17874			
17875	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
17876			STK_T型の配列の要素数
17877	ROUND_STK_T(sz)		要素数COUNT_STK_T(sz)のSTK_T型の配列のサイズ (sz
17878			を, STK_T型のサイズの倍数になるように大きい方に
17879			丸めた値)
17880			
17881	(4) 同期・通信機能		
17882			
17883	TSZ_DTQMB(dtqcnt)		dtqcntで指定した数のデータを格納できるデータ
17884			キュー管理領域のサイズ (バイト数)
17885	TCNT_DTQMB(dtqcnt)		dtqcntで指定した数のデータを格納できるデータ
17886			キュー管理領域を確保するために必要なMB_T型の配
17887			列の要素数
17888			
17889	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
17890			タキュー管理領域のサイズ (バイト数)
17891	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
17892			タキュー管理領域を確保するために必要なMB_T型の
17893			配列の要素数
17894			
17895	TSZ_MBFMB(msgcnt, msgsz)		msgszで指定したサイズのメッセージを,
17896			msgcntで指定した数だけ格納できるメッセー
17897			ジバッファ管理領域のサイズ (バイト数)
17898	TCNT_MBFMB(msgcnt, msgsz)		msgszで指定したサイズのメッセージを,
17899			msgcntで指定した数だけ格納できるメッセー
17900			ジバッファ管理領域を確保するために必要

17901		なMB_T型の配列の要素数
17902		
17903	(5) メモリプール管理機能	
17904		
17905	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
17906		プール領域を確保するために、固定長メモリブロッ
17907		ク1つあたりに必要なMPF_T型の配列の要素数を求め
17908		るマクロ
17909	ROUND_MPF_T(blksz)	要素数COUNT_MPF_T(blksz)のMPF_T型の配列のサイズ
17910		(blkszを、MPF_T型のサイズの倍数になるように大き
17911		い方に丸めた値)
17912		
17913	TSZ_MPFMB(blkent)	blkentで指定した数の固定長メモリブロックを管理
17914		することができる固定長メモリプール管理領域のサ
17915		イズ (バイト数)
17916	TCNT_MPFMB(blkent)	blkentで指定した数の固定長メモリブロックを管理
17917		することができる固定長メモリプール管理領域を確
17918		保するために必要なMB_T型の配列の要素数
17919		
17920	5.5 構成マクロ	
17921		
17922	5.5.1 TOPPERS共通構成マクロ	
17923		
17924	(1) 相対時間の範囲	
17925		
17926	TMAX_RELTIM	相対時間に指定できる最大値
17927		
17928	5.5.2 カーネル共通構成マクロ	
17929		
17930	(1) サポートする機能	
17931		
17932	TOPPERS_SUPPORT_PROTECT	保護機能対応のカーネル
17933	TOPPERS_SUPPORT_MULTI_PRC	マルチプロセッサ対応のカーネル
17934	TOPPERS_SUPPORT_DYNAMIC_CRE	動的生成対応のカーネル
17935		
17936	(2) 優先度の範囲	
17937		
17938	TMIN_TPRI	タスク優先度の最小値 (=1)
17939	TMAX_TPRI	タスク優先度の最大値
17940		
17941	(3) プロセッサの数	
17942		
17943	TNUM_PRCID	プロセッサの数
17944		
17945	(4) 特殊な役割を持ったプロセッサ	
17946		
17947	TOPPERS_MASTER_PRCID	マスタプロセッサのID番号
17948	TOPPERS_SYSTIM_PRCID	システム時刻管理プロセッサのID番号
17949		
17950	(5) タイマ方式	

17951		
17952	TOPPERS_SYSTIM_LOCAL	ローカルタイマ方式の場合にマクロ定義
17953	TOPPERS_SYSTIM_GLOBAL	グローバルタイマ方式の場合にマクロ定義
17954		
17955	(6) バージョン情報	
17956		
17957	TKERNEL_MAKER	カーネルのメーカーコード (=0x0118)
17958	TKERNEL_PRID	カーネルの識別番号
17959	TKERNEL_SPVER	カーネル仕様のバージョン番号
17960	TKERNEL_PRVER	カーネルのバージョン番号
17961		
17962	5.5.3 カーネルの機能毎の構成マクロ	
17963		
17964	(1) タスク管理機能	
17965		
17966	TMAX_ACTCNT	タスクの起動要求キューイング数の最大値
17967		
17968	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)
17969		
17970		
17971	(2) タスク付属同期機能	
17972		
17973	TMAX_WUPCNT	タスクの起床要求キューイング数の最大値
17974		
17975	(3) タスク例外処理機能	
17976		
17977	TBIT_TEXPTN	タスク例外要因のビット数 (TEXPTNの有効ビット数)
17978		
17979	(4) 同期・通信機能	
17980		
17981	セマフォ	
17982		
17983	TMAX_MAXSEM	セマフォの最大資源数の最大値
17984		
17985	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)
17986		
17987		
17988	イベントフラグ	
17989		
17990	TBIT_FLGPTN	イベントフラグのビット数 (FLGPTNの有効ビット数)
17991		
17992	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)
17993		
17994		
17995		
17996	データキュー	
17997		
17998	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)
17999		
18000		

18001		
18002	優先度データキュー	
18003		
18004	TMIN_DPRI	データ優先度の最小値 (=1)
18005	TMAX_DPRI	データ優先度の最大値
18006		
18007	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)
18008		
18009		
18010		
18011	メールボックス	
18012		
18013	TMIN_MPRI	メッセージ優先度の最小値 (=1)
18014	TMAX_MPRI	メッセージ優先度の最大値
18015		
18016	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)
18017		
18018		
18019		
18020	ミューテックス	
18021		
18022	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)
18023		
18024		
18025		
18026	メッセージバッファ	
18027		
18028	TNUM_MBFID	登録できるメッセージバッファの数 (動的生成対応でないカーネルでは, 静的APIによって登録されたメッセージバッファの数に一致)
18029		
18030		
18031		
18032	スピンロック	
18033		
18034	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)
18035		
18036		
18037		
18038	(5) メモリプール管理機能	
18039		
18040	固定長メモリプール	
18041		
18042	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)
18043		
18044		
18045		
18046	(6) 時間管理機能	
18047		
18048	システム時刻管理	
18049		
18050	TIC_NUME	タイムティックの周期 (単位はミリ秒) の分子

18051	TIC_DENO	タイムティックの周期（単位はミリ秒）の分母
18052		
18053	TOPPERS_SUPPORT_GET_UTM	get_utmがサポートされている
18054		
18055	周期ハンドラ	
18056		
18057	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）
18058		
18059		
18060		
18061	アラームハンドラ	
18062		
18063	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）
18064		
18065		
18066		
18067	オーバランハンドラ	
18068		
18069	TMAX_OVRTIM	プロセッサ時間に指定できる最大値
18070		
18071	TOPPERS_SUPPORT_OVRHDR	オーバランハンドラ機能がサポートされている
18072		
18073		
18074	(7) システム状態管理機能	
18075		
18076	なし	
18077		
18078	(8) メモリオブジェクト管理機能	
18079		
18080	TOPPERS_SUPPORT_ATT_MOD	ATT_MOD／ATA_MODがサポートされている
18081	TOPPERS_SUPPORT_ATT_PMA	ATT_PMA／ATA_PMA／att_pmaがサポートされている
18082		
18083		
18084	(9) 割込み管理機能	
18085		
18086	TMIN_INTPRI	割込み優先度の最小値（最高値）
18087	TMAX_INTPRI	割込み優先度の最大値（最低値，=-1）
18088		
18089	TMIN_ISRPRI	割込みサバスルーチン優先度の最小値（=1）
18090	TMAX_ISRPRI	割込みサバスルーチン優先度の最大値
18091		
18092	TOPPERS_SUPPORT_DIS_INT	dis_intがサポートされている
18093	TOPPERS_SUPPORT_ENA_INT	ena_intがサポートされている
18094		
18095	(10) CPU例外管理機能	
18096		
18097	なし	
18098		
18099	(11) 拡張サバスコール管理機能	
18100		

18101 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
18102 いカーネルでは、静的APIによって登録された拡張サービ
18103 スコールの数に一致）
18104

18105 (12) システム構成管理機能

18106
18107 なし

18108

18109 5.6 エラーコード一覧

18110

18111 (1) メインエラーコード

18112

18113	E_SYS	-5	システムエラー
18114	E_NOSPT	-9	未サポート機能
18115	E_RSFN	-10	予約機能コード
18116	E_RSATR	-11	予約属性
18117	E_PAR	-17	パラメータエラー
18118	E_ID	-18	不正ID番号
18119	E_CTX	-25	コンテキストエラー
18120	E_MACV	-26	メモリアクセス違反
18121	E_OACV	-27	オブジェクトアクセス違反
18122	E_ILUSE	-28	サービスコール不正使用
18123	E_NOMEM	-33	メモリ不足
18124	E_NOID	-34	ID番号不足
18125	E_NORES	-35	資源不足
18126	E_OBJ	-41	オブジェクト状態エラー
18127	E_NOEXS	-42	オブジェクト未登録
18128	E_QOVR	-43	キューイングオーバーフロー
18129	E_RLWAI	-49	待ち禁止状態または待ち状態の強制解除
18130	E_TMOUT	-50	ポーリング失敗またはタイムアウト
18131	E_DLT	-51	待ちオブジェクトの削除または再初期化
18132	E_CLS	-52	待ちオブジェクトの状態変化
18133	E_WBLK	-57	ノンブロッキング受付け
18134	E_BOVR	-58	バッファオーバーフロー

18135

18136 5.7 機能コード一覧【NGKI4036】

18137

18138	-----				
18139		-0	-1	-2	-3
18140	-----				
18141	-0x01	予約	予約	予約	予約
18142	-0x05	act_tsk	iact_tsk	can_act	ext_tsk
18143	-0x09	ter_tsk	chg_pri	get_pri	get_inf
18144	-0x0d	slp_tsk	tslp_tsk	wup_tsk	iwup_tsk
18145	-0x11	can_wup	rel_wai	irel_wai	予約
18146	-0x15	dis_wai	idis_wai	ena_wai	iena_wai
18147	-0x19	sus_tsk	rsm_tsk	dly_tsk	予約
18148	-0x1d	ras_tex	iras_tex	dis_tex	ena_tex
18149	-0x21	sns_tex	ref_tex	予約	予約
18150	-0x25	sig_sem	isig_sem	wai_sem	pol_sem

18151	-0x29	twai_sem	予約	予約	予約
18152	-0x2d	set_flg	iset_flg	clr_flg	wai_flg
18153	-0x31	pol_flg	twai_flg	予約	予約
18154	-0x35	snd_dtq	psnd_dtq	ipsnd_dtq	tsnd_dtq
18155	-0x39	fsnd_dtq	ifsnd_dtq	rcv_dtq	prcv_dtq
18156	-0x3d	trcv_dtq	予約	予約	予約
18157	-0x41	snd_pdq	psnd_pdq	ipsnd_pdq	tsnd_pdq
18158	-0x45	rcv_pdq	prcv_pdq	trcv_pdq	予約
18159	-0x49	snd_mbx	rcv_mbx	prcv_mbx	trcv_mbx
18160	-0x4d	loc_mtx	ploc_mtx	tloc_mtx	unl_mtx
18161	-0x51	snd_mbf	psnd_mbf	tsnd_mbf	rcv_mbf
18162	-0x55	prcv_mbf	trcv_mbf	予約	予約
18163	-0x59	get_mpf	pget_mpf	tget_mpf	rel_mpf
18164	-0x5d	get_tim	get_utm	予約	ref_ovr
18165	-0x61	sta_cyc	stp_cyc	予約	予約
18166	-0x65	sta_alm	ista_alm	stp_alm	istp_alm
18167	-0x69	sta_ovr	ista_ovr	stp_ovr	istp_ovr
18168	-0x6d	sac_sys	ref_sys	rot_rdq	irotd_rdq
18169	-0x71	get_did	予約	get_tid	iget_tid
18170	-0x75	loc_cpu	iloc_cpu	unl_cpu	iunl_cpu
18171	-0x79	dis_dsp	ena_dsp	sns_ctx	sns_loc
18172	-0x7d	sns_dsp	sns_dpn	sns_ker	ext_ker
18173	-0x81	att_mem	det_mem	sac_mem	prb_mem
18174	-0x85	ref_mem	予約	att_pma	予約
18175	-0x89	cfg_int	dis_int	ena_int	ref_int
18176	-0x8d	chg_ipm	get_ipm	予約	予約
18177	-0x91	xsns_dpn	xsns_xpn	予約	予約
18178	-0x95	ref_cfg	ref_ver	予約	予約
18179	-0x99	予約	予約	予約	予約
18180	-0x9d	予約	予約	予約	予約
18181	-0xa1	予約	ini_sem	ini_flg	ini_dtq
18182	-0xa5	ini_pdq	ini_mbx	ini_mtx	ini_mbf
18183	-0xa9	ini_mpf	予約	予約	予約
18184	-0xad	予約	予約	予約	予約
18185	-0xb1	ref_tsk	ref_sem	ref_flg	ref_dtq
18186	-0xb5	ref_pdq	ref_mbx	ref_mtx	ref_mbf
18187	-0xb9	ref_mpf	ref_cyc	ref_alm	ref_isr
18188	-0xbd	ref_spn	予約	予約	予約
18189	-0xc1	acre_tsk	acre_sem	acre_flg	acre_dtq
18190	-0xc5	acre_pdq	acre_mbx	acre_mtx	acre_mbf
18191	-0xc9	acre_mpf	acre_cyc	acre_alm	acre_isr
18192	-0xcd	acre_spn	予約	予約	予約
18193	-0xd1	del_tsk	del_sem	del_flg	del_dtq
18194	-0xd5	del_pdq	del_mbx	del_mtx	del_mbf
18195	-0xd9	del_mpf	del_cyc	del_alm	del_isr
18196	-0xdd	del_spn	予約	予約	予約
18197	-0xe1	sac_tsk	sac_sem	sac_flg	sac_dtq
18198	-0xe5	sac_pdq	予約	sac_mtx	sac_mbf
18199	-0xe9	sac_mpf	sac_cyc	sac_alm	sac_isr
18200	-0xed	sac_spn	予約	予約	予約

18201	-0xf1	def_tex	def_ovr	def_inh	def_exc
18202	-0xf5	def_svc	予約	予約	予約
18203	-0xf9	予約	予約	予約	予約
18204	-0xfd	予約	予約	予約	予約
18205	-0x101	mact_tsk	imact_tsk	mig_tsk	予約
18206	-0x105	msta_cyc	予約	msta_alm	imsta_alm
18207	-0x109	mrot_rdq	imrot_rdq	get_pid	iget_pid
18208	-0x10d	予約	予約	予約	予約
18209	-0x111	loc_spn	iloc_spn	try_spn	itry_spn
18210	-0x115	unl_spn	iunl_spn	予約	予約
18211	-0x119	予約	予約	予約	予約
18212	-0x11d	予約	予約	予約	予約

18213 -----

18214

18215 【μ ITRON4.0仕様との関係】

18216

18217 サービスコールの機能コードを割り当てなおした.

18218

18219 5.8 カーネルオブジェクトに対するアクセスの種別

18220

18221	-----				
18222	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
18223	-----				
18224	メモリオブジェクト	書込み	読出し	det_mem	ref_mem
18225			実行	sac_mem	prb_mem
18226	-----				
18227	タスク	act_tsk	ter_tsk	del_tsk	get_pri
18228		mact_tsk	chg_pri	sac_tsk	ref_tsk
18229		can_act	rel_wai	def_tex	ref_tex
18230		mig_tsk	sus_tsk		ref_ovr
18231		wup_tsk	rsm_tsk		prb_mem
18232		can_wup	dis_wai		
18233			ena_wai		
18234			ras_tex		
18235			sta_ovr		
18236			stp_ovr		
18237	-----				
18238	セマフォ	sig_sem	wai_sem	del_sem	ref_sem
18239			pol_sem	ini_sem	
18240			twai_sem	sac_sem	
18241	-----				
18242	イベントフラグ	set_flg	wai_flg	del_flg	ref_flg
18243		clr_flg	pol_flg	ini_flg	
18244			twai_flg	sac_flg	
18245	-----				
18246	データキュー	snd_dtq	rcv_dtq	del_dtq	ref_dtq
18247		psnd_dtq	prcv_dtq	ini_dtq	
18248		tsnd_dtq	trcv_dtq	sac_dtq	
18249		fsnd_dtq			
18250	-----				

18251	優先度データキュー	snd_pdq	rcv_pdq	del_pdq	ref_pdq
18252		psnd_pdq	prcv_pdq	ini_pdq	
18253		tsnd_pdq	trcv_pdq	sac_pdq	
18254		-----			
18255	メッセージバッファ	snd_mbf	rcv_mbf	del_mbf	ref_mbf
18256		psnd_mbf	prcv_mbf	ini_mbf	
18257		tsnd_mbf	trcv_mbf	sac_mbf	
18258		-----			
18259	ミューテックス	loc_mtx	-	del_mtx	ref_mtx
18260		ploc_mtx		ini_mtx	
18261		tloc_mtx		sac_mtx	
18262		unl_mtx			
18263		-----			
18264	スピンロック	loc_spn	-	del_spn	ref_spn
18265		try_spn		sac_spn	
18266		unl_spn			
18267		-----			
18268	固定長メモリプール	get_mpf	rel_mpf	del_mpf	ref_mpf
18269		pget_mpf		ini_mpf	
18270		tget_mpf		sac_mpf	
18271		-----			
18272	周期ハンドラ	sta_cyc	stp_cyc	del_cyc	ref_cyc
18273		msta_cyc		sac_cyc	
18274		-----			
18275	アラームハンドラ	sta_alm	stp_alm	del_alm	ref_alm
18276		msta_alm		sac_alm	
18277		-----			
18278	割込みサービスルーチン	-	-	del_isr	ref_isr
18279				sac_isr	
18280		-----			
18281	システム状態	rot_rdq	loc_cpu	acre_yyy	get_tim
18282		mrot_rdq	unl_cpu	att_mem	get_ipm
18283		dis_dsp	dis_int	att_pma	ref_sys
18284		ena_dsp	ena_int	cfg_int	ref_int
18285			chg_ipm	def_inh	ref_cfg
18286				def_exc	ref_ver
18287				def_svc	
18288				def_ovr	
18289		-----			

18290

18291 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：

18292

- 18293 • 自タスクへの操作 (ext_tsk, get_inf, slp_tsk, tslp_tsk, dly_tsk,
- 18294 dis_tex, ena_tex)
- 18295 • タスク例外状態参照 (sns_tex)
- 18296 • 性能評価用システム時刻の参照 (get_utm)
- 18297 • システム状態参照 (get_tid, get_did, get_pid, sns_ctx, sns_loc,
- 18298 sns_dsp, sns_dpn, sns_ker)
- 18299 • CPU例外発生時の状態参照 (xsns_dpn, xsns_xpn)
- 18300 • 拡張サービスコールの呼出し (cal_svc)

18301
18302
18303
18304
18305
18306
18307
18308
18309
18310
18311
18312
18313
18314
18315
18316
18317
18318
18319
18320
18321
18322
18323
18324
18325
18326
18327
18328
18329
18330
18331
18332
18333
18334
18335
18336
18337
18338
18339
18340
18341
18342
18343
18344
18345
18346
18347
18348
18349
18350

カーネルドメインのみから呼び出すことができるサービスコール：

- ・システム状態のアクセス許可ベクタの設定 (sac_sys)
- ・カーネルの終了 (ext_ker)
- ・非タスクコンテキスト専用のサービスコール

【補足説明】

xsns_dpnとxsns_xpnは、エラーコードを返さないために、すべての保護ドメインから呼び出すことができるサービスコールとしているが、タスクコンテキストから呼び出した場合には必ずtrueが返ることとしており、実質的にはカーネルドメインのみから呼び出すことができる。

【 μ ITRON4.0/PX仕様との関係】

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、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get_ixx) をシステム状態に対する通常操作2から参照操作に、sac_sysをシステム状態に対する管理操作からカーネルドメインのみから呼び出すことができるサービスコールに変更した。システム時刻に対するアクセス許可ベクタは廃止し、get_timはシステム状態に対する参照操作とした。

【仕様変更の経緯】

この仕様のRelease 1.5以前では、unl_mtxは、アクセス許可ベクタによるアクセス保護を行わないサービスコールとしていた。これは、ミューテックスをロックしたタスク以外がunl_mtxを呼び出すとE_ILUSEエラーとなるため、実質的には対象ミューテックスの通常操作1としてアクセス保護されているとみなすことができると考えたためである。しかし、タスクが拡張サービスコールの中でミューテックスをロックした場合、アクセス許可ベクタではアクセスが許可されていないミューテックスをロックすることができる。このようなミューテックスのロック解除は、タスクから直接unl_mtxを呼んで行うのではなく、拡張サービスコールの中で行うべきと考えられる。そこで、unl_mtxを、対象ミューテックスの通常操作1としてアクセス保護する仕様に変更した。なお、HRP2カーネルRelease 2.1以前のバージョンは、古い仕様に従って実装されている。

5.9 ターゲット定義事項一覧

- ・割込み優先度の段階数 [NGKI0256]
- ・割込み番号の付与方法 [NGKI0272]
- ・割込みハンドラ番号の付与方法 [NGKI0273]
- ・割込み番号に対応しない割込みハンドラ番号や、割込みハンドラ番号に対応しない割込み番号を設けるか [NGKI0276]
- ・受け付けた割込み要求に対して、割込みサービスルーチンも割込みハンドラも登録していない場合の振舞い [NGKI0249]

18351
18352 • 割込み要求禁止フラグがサポートされているか [NGKI0260] [NGKI0261]
18353
18354 • 割込み要求禁止フラグの振舞いを仕様と異なるものとするか [NGKI0261]
18355
18356 • 割込み要求ラインのトリガモードの設定がサポートされているか [NGKI0267]
18357
18358 • 割込み要求ラインをエッジトリガに設定する場合に、ポジティブエッジトリ
18359 ガかネガティブエッジトリガか両エッジトリガかを設定できるか [NGKI0265]
18360
18361 • 割込み要求ラインをレベルトリガに設定する場合に、ローレベルトリガかハ
18362 イレベルトリガかを設定できるか [NGKI0266]
18363
18364 • 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
18365 のプロセッサに対しては割込みがマスク／マスク解除されないものとするか
18366 [M] [NGKI0281]
18367
18368 • TMIN_INTPRIを固定するか設定できるようにするかと、設定できるようにする
18369 場合の設定方法 [NGKI0288]
18370
18371 • 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）
18372 [NGKI0289]
18373
18374 • 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
18375 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
18376 方法 [NGKI0292]
18377
18378 • カーネル管理外の割込みの設定方法として、3つの方法のいずれを採用するか
18379 [NGKI0295]
18380
18381 • カーネル管理外とされた割込みに対して、カーネルのAPIにより割込みハンド
18382 ラを登録できるかと、割込み要求ラインの属性を設定できるか [NGKI0297]
18383
18384 • CPU例外ハンドラ番号の付与方法 [NGKI0306]
18385
18386 • 発生したCPU例外に対して、CPU例外ハンドラを登録していない場合の振舞い
18387 [NGKI0314]
18388
18389 • メモリオブジェクトの先頭番地とサイズに対する制約 [P] [NGKI0070]
18390 [NGKI2774]
18391
18392 • コンパイラが出力しないセクションの中で、どれを標準のセクションと扱う
18393 か [P] [NGKI0113]
18394
18395 • 保護ドメイン毎の標準セクションのセクション名を、標準のセクション名と
18396 保護ドメイン名を“_”でつないだものとする仕様を変更するか [P] [NGKI0116]
18397
18398 • タスクのユーザスタック領域はそのタスク（とカーネルドメインに属する処
18399 理単位）のみがアクセスできるという仕様を変更するか [P] [NGKI0074]
18400

- 18401 • 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
18402 ない書込みアクセスまたは読出しアクセス（実行アクセスを含む）を行おう
18403 とした場合に、どのCPU例外ハンドラが起動されるか [P] [NGKI0411]
- 18404
- 18405 • 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
18406 書込みアクセスまたは読出しアクセスを行おうとした場合に、サービスコー
18407 ルからE_MACVエラーが返るか、メモリアクセス違反ハンドラが起動されるか
18408 [P] [NGKI0413]
- 18409
- 18410 • メモリアクセス違反ハンドラで、アクセス違反を発生させたアクセスに関す
18411 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）
18412 を参照する方法 [P] [NGKI0414]
- 18413
- 18414 • 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
18415 む）に対して設定できるアクセス許可パターンに対する制限 [P] [NGKI0417]
- 18416
- 18417 • 1つの保護ドメインに登録できるメモリオブジェクトの数に対する制限 [P]
18418 [NGKI0423]
- 18419
- 18420 • ユーザスタック領域に対して実行アクセスを行えるか [P] [NGKI0440]
- 18421
- 18422 • タスクのユーザスタック領域を、そのタスクが属する保護ドメイン全体から
18423 アクセスできるものとするか [P] [NGKI0441]
- 18424
- 18425 • 使用できるクラスのID番号とその属性 [M] [NGKI0107]
- 18426
- 18427 • どのプロセッサをマスタプロセッサとするか [M] [NGKI0101]
- 18428
- 18429 • ローカルタイマ方式とグローバルタイマ方式のどちらの方式を用いることが
18430 できるか [M] [NGKI0108]
- 18431
- 18432 • グローバルタイマ方式の場合に、どのプロセッサをシステム時刻管理プロセッ
18433 サとするか [M] [NGKI0111]
- 18434
- 18435 • int8_t, uint8_t, int64_t, uint64_t, int128_t, uint128_t, float32_t,
18436 double64_tが使用できるか [NGKI0488] [NGKI0490]
- 18437
- 18438 • ターゲット定義のタスク属性 [NGKI1016]
- 18439
- 18440 • タスクが用いるスタック領域のサイズの最小値 [NGKI1042]
- 18441
- 18442 • タスクのシステムスタック領域のサイズの最小値 [P] [NGKI1044]
- 18443
- 18444 • タスクが用いるスタック領域の先頭番地とサイズに対する制約 [NGKI1050]
18445 [NGKI1056]
- 18446
- 18447 • ユーザスタックのスタック領域（ユーザスタック領域）をアプリケーション
18448 で確保する方法 [P] [NGKI1059]
- 18449
- 18450 • タスクのシステムスタック領域の先頭番地とサイズに対する制約 [P]

18451 [NGKI1062] [NGKI1065] [NGKI1070]
18452
18453 • データキュー管理領域の先頭番地に対する制約 [NGKI1687]
18454
18455 • 優先度データキュー管理領域の先頭番地に対する制約 [NGKI1824]
18456
18457 • メッセージバッファ管理領域の先頭番地とサイズに対する制約 [NGKI3319]
18458 [NGKI3324]
18459
18460 • 生成できるスピンロックの数の上限 [M] [NGKI2142]
18461
18462 • スピンロックに対して、複数のプロセッサがロックの取得を待っている時に、
18463 どのプロセッサが最初にロックを取得できるか [M] [NGKI2183]
18464
18465 • 固定長メモリプール領域の先頭番地に対する制約 [NGKI2249]
18466
18467 • 固定長メモリプール管理領域の先頭番地に対する制約 [NGKI2256]
18468
18469 • タイムティックの周期 [NGKI2335]
18470
18471 • マルチプロセッサ対応カーネルにおける性能評価用システム時刻の扱い [M]
18472 [NGKI2346]
18473
18474 • get_utmがサポートされているか [NGKI2360]
18475
18476 • オーバランハンドラ機能がサポートされているか [NGKI2598]
18477
18478 • オーバランハンドラ機能のプロセッサ時間に指定できる値の上限 [NGKI2594]
18479
18480 • ターゲット定義のメモリリージョン属性 [P]
18481
18482 • メモリリージョンの先頭番地とサイズに対する制約 [P] [NGKI2768]
18483
18484 • メモリオブジェクトに対するTA_NOWRITE属性, TA_NOREAD属性, TA_EXEC属性
18485 の内, どのような場合にどの属性の指定が無視されるか [P] [NGKI2782]
18486
18487 • ショートデータ領域がサポートされておらず, TA_SDATA属性が無視されるか
18488 [P] [NGKI2789]
18489
18490 • TA_NOWRITEを指定した場合に, TA_SDATAが無視されるか [P] [NGKI2790]
18491
18492 • TA_UNCACHE属性やTA_IODEV属性を指定しても意味がなく, これらの属性が無
18493 視されるか [P] [NGKI2792]
18494
18495 • キャッシュ禁止にできないメモリオブジェクトと周辺デバイスの領域として
18496 扱うことができないメモリオブジェクト [P] [NGKI2793]
18497
18498 • ターゲット定義のメモリオブジェクト属性 [P] [NGKI2794]
18499
18500 • ATA_SECにより登録できるセクションが属する保護ドメインや登録できる数

18501 に対する制限 [P] [NGKI2831]
18502
18503 • ATT_MOD／ATA_MODがサポートされているか [P] [NGKI2859]
18504
18505 • ATT_MOD／ATA_MODにより登録されるセクション毎のメモリオブジェクトに設
18506 定されるメモリオブジェクト属性 [P] [NGKI2850]
18507
18508 • クラスの囲みの中に記述されたATT_MOD／ATA_MODにおいて、クラスの標準メ
18509 モリリージョンが定義されている場合でも、共通の標準メモリリージョンに
18510 配置されるセクション [PM] [NGKI3271]
18511
18512 • ATA_MODにより登録できるオブジェクトモジュールが属する保護ドメインや登
18513 録できる数に対する制限 [P] [NGKI2857]
18514
18515 • ATT_MEM／ATA_MEMにより登録できるメモリオブジェクトが属する保護ドメイ
18516 ンや登録できる数に対する制限 [P] [NGKI2878]
18517
18518 • ATT_MEM／ATA_MEM／att_memにより登録するメモリ領域の先頭番地とサイズに
18519 対する制約 [P] [NGKI2880]
18520
18521 • ATT_PMA／ATA_PMA／att_pmaがサポートされているか [P] [NGKI2903]
18522 [HRPS0156]
18523
18524 • 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
18525 ンや登録できる数に対する制限 [P] [NGKI2898]
18526
18527 • ATT_PMA／ATA_PMA／att_pmaにより登録するメモリ領域の先頭番地とサイズ、
18528 物理アドレス空間における先頭番地に対する制約 [P] [NGKI2900]
18529
18530 • ターゲット定義の割込み要求ライン属性 [NGKI2945]
18531
18532 • 割込みハンドラ属性にTA_NONKERNELを指定できるか [NGKI2957]
18533
18534 • その他のターゲット定義の割込みハンドラ属性 [NGKI2959]
18535
18536 • cfg_intにおいて、複数の割込み要求ラインの割込み優先度が連動して設定さ
18537 れるか [D] [NGKI2980]
18538
18539 • CFG_INT／cfg_intで、カーネル管理外の割込み要求ラインに対しても属性を
18540 設定できるか [NGKI2982]
18541
18542 • CFG_INT／cfg_intで、各割込み要求ラインに対して設定できる割込み要求ラ
18543 イン属性／割込み優先度に対する制限 [NGKI2986]
18544
18545 • 割込みサービスルーチンが属することができるクラスに対する制限 [M]
18546 [NGKI3018]
18547
18548 • CRE_ISR／ATT_ISRにおいて、isrが不正である場合にE_PARエラーが検出され
18549 るか [NGKI3020]
18550

18551 • DEF_INH／def_inhで、カーネル管理外の割込みに対しても割込みハンドラを
18552 定義できるか [NGKI3064]
18553
18554 • カーネル管理外に固定されている割込みハンドラがあるか [NGKI3067]
18555
18556 • カーネル管理に固定されている割込みハンドラがあるか [NGKI3068]
18557
18558 • 割込みハンドラが属することができるクラスに対する制限 [M] [NGKI3074]
18559
18560 • def_inhで、静的APIで定義された割込みハンドラの定義を解除できるか [D]
18561 [NGKI3077]
18562
18563 • DEF_INH／def_inhで割込みハンドラを定義（または定義解除）できない割込
18564 みハンドラ番号 [NGKI3078]
18565
18566 • def_inhを呼び出したタスクが割り付けられているプロセッサから定義（また
18567 は定義解除）できない割込みハンドラ [M] [NGKI3079]
18568
18569 • DEF_INHにおいて、inthdrが不正である場合にE_PARエラーが検出されるか
18570 [NGKI3080]
18571
18572 • dis_intがサポートされているか [NGKI3091]
18573
18574 • dis_intにより、どのような場合に割込み要求ラインの割込み要求禁止フラグ
18575 をセットできないか [NGKI3087]
18576
18577 • dis_intにおいて、割込み要求禁止フラグの振舞いが、この仕様の規定と異な
18578 るか [NGKI3089]
18579
18580 • ena_intがサポートされているか [NGKI3104]
18581
18582 • ena_intにより、どのような場合に割込み要求ラインの割込み要求禁止フラグ
18583 をクリアできないか [NGKI3100]
18584
18585 • ena_intにおいて、割込み要求禁止フラグの振舞いが、この仕様の規定と異な
18586 るか [NGKI3102]
18587
18588 • chg_ipmにより、割込み優先度マスクをTMIN_INTPRIよりも小さい値に変更で
18589 きるか [NGKI3114]
18590
18591 • ターゲット定義のCPU例外ハンドラ属性 [NGKI3123]
18592
18593 • def_excで、静的APIで定義されたCPU例外ハンドラの定義を解除できるか [D]
18594 [NGKI3148]
18595
18596 • DEF_EXCにおいて、exchdrが不正である場合にE_PARエラーが検出されるか
18597 [NGKI3149]
18598
18599 • 非タスクコンテキスト用スタック領域のサイズの最小値 [NGKI3254]
18600

- 18601 • 非タスクコンテキスト用スタック領域の先頭番地とサイズに対する制約
- 18602 [NGKI3220] [NGKI3222]
- 18603
- 18604 • DEF_ICSにより非タスクコンテキスト用スタック領域を設定しない場合の、非
- 18605 タスクコンテキスト用スタック領域のデフォルトのサイズ [NGKI3224]
- 18606
- 18607 • 共有スタック領域のサイズの最小値 [NGKI3255]
- 18608
- 18609 • 共有スタック領域の先頭番地とサイズに対する制約 [NGKI3234] [NGKI3236]
- 18610
- 18611 • ATT_INIにおいて、inirtnが不正である場合にE_PARエラーが検出されるか
- 18612 [NGKI3246]
- 18613
- 18614 • ATT_TERにおいて、terrtnが不正である場合にE_PARエラーが検出されるか
- 18615 [NGKI3253]
- 18616

5. 10 省略名の元になった英語

5. 10. 1 サービスコールと静的APIの名称の中のxxxの元になった英語

xxx	元になった英語

act	activate
aid	automatically assigned ID
ata	attach with access control vector
att	attach
cal	call
can	cancel
cfg	configure
chg	change
clr	clear
cre	create
def	define
del	delete
det	detach
dis	disable
dly	delay
ena	enable
epr	execution priority
ext	exit
get	get
ini	initialize
lmt	limit
lnk	link
loc	lock
mig	migrate
pol	poll
prb	probe
ras	raise
rcv	receive

18651	ref	reference
18652	rel	release
18653	rot	rotate
18654	rsm	resume
18655	sac	set access control vector
18656	set	set
18657	sig	signal
18658	slp	sleep
18659	snd	send
18660	sns	sense
18661	sta	start
18662	stp	stop
18663	sus	suspend
18664	ter	terminate
18665	try	try
18666	unl	unlock
18667	wai	wait
18668	wup	wake up

18669

18670 5. 10. 2 サービスコールと静的APIの名称の中のyyyの元になった英語

18671

18672 yyy 元になった英語

18673

18674	act	activation
18675	alm	alarm handler
18676	cfg	configuration
18677	cpu	CPU
18678	ctx	context
18679	cyc	cyclic handler
18680	did	domain ID
18681	dom	domain
18682	dpn	dispatch pending
18683	dsp	dispatch
18684	dtq	data queue
18685	exc	exception
18686	flg	eventflag
18687	ics	interrupt context stack
18688	inf	information
18689	inh	interrupt handler
18690	ini	initilization
18691	int	interrupt
18692	ipm	interrupt priority mask
18693	isr	interrupt service routine
18694	ker	kernel
18695	loc	lock
18696	mbf	message buffer
18697	mbx	mailbox
18698	mpf	fixed-sized memory pool
18699	mem	memory
18700	mod	module

18701	mtx	mutex
18702	ovr	overflow handler
18703	pdq	priority data queue
18704	pid	processor ID
18705	pma	physical memory area
18706	pri	priority
18707	rdq	ready queue
18708	reg	region
18709	sec	section
18710	sem	semaphore
18711	srg	standard memory region
18712	spn	spin lock
18713	stk	stack
18714	sys	system
18715	svc	service call
18716	ter	termination
18717	tex	task exception
18718	tid	task ID
18719	tim	time
18720	tsk	task
18721	utm	time in micro second
18722	ver	version
18723	wai	wait
18724	wup	wake up
18725	xpn	exception pending

18726

18727 5.10.3 サービスコールの名称の中のzの元になった英語

18728

z	元になった英語

18731	a automatic ID assignment
18732	f force
18733	i interrupt
18734	m multiprocessor
18735	p poll
18736	t timeout
18737	x exception

18738

18739 5.11 バージョン履歴

18740

18741	2008年11月19日	Release 1.0.0	最初のリリース
18742	2009年5月8日	Release 1.1.0	FMPカーネルに関する記述が完成
18743	2010年5月10日	Release 1.2.0	
18744	2011年5月5日	Release 1.3.0	HRP2カーネルに関する記述が完成
18745	2012年5月16日	Release 1.4.0	SSPカーネルに関する記述が完成
18746	2012年12月19日	Release 1.5.0	HRP2カーネルの仕様変更を反映
18747	2014年1月16日	Release 1.6.0	
18748	2014年11月17日	Release 1.7.0	
18749	2015年5月30日	Release 1.7.1	
18750			

18751 以上

アプリケーションシステム

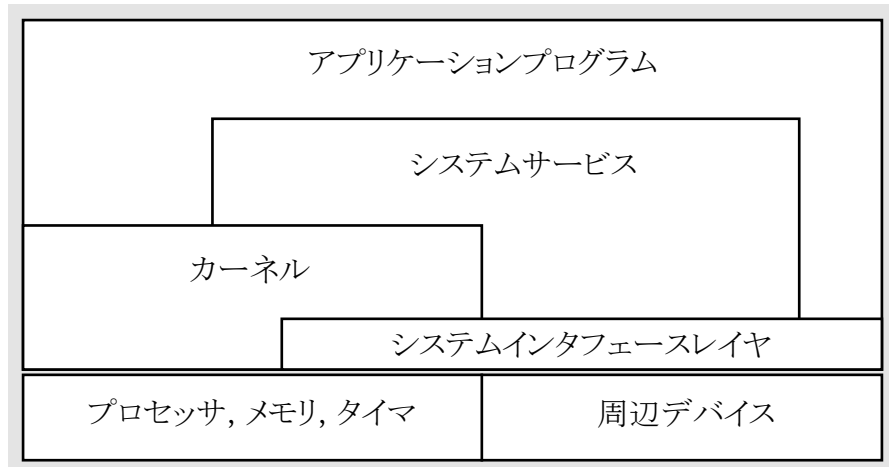


図2-1. 想定するソフトウェア構成

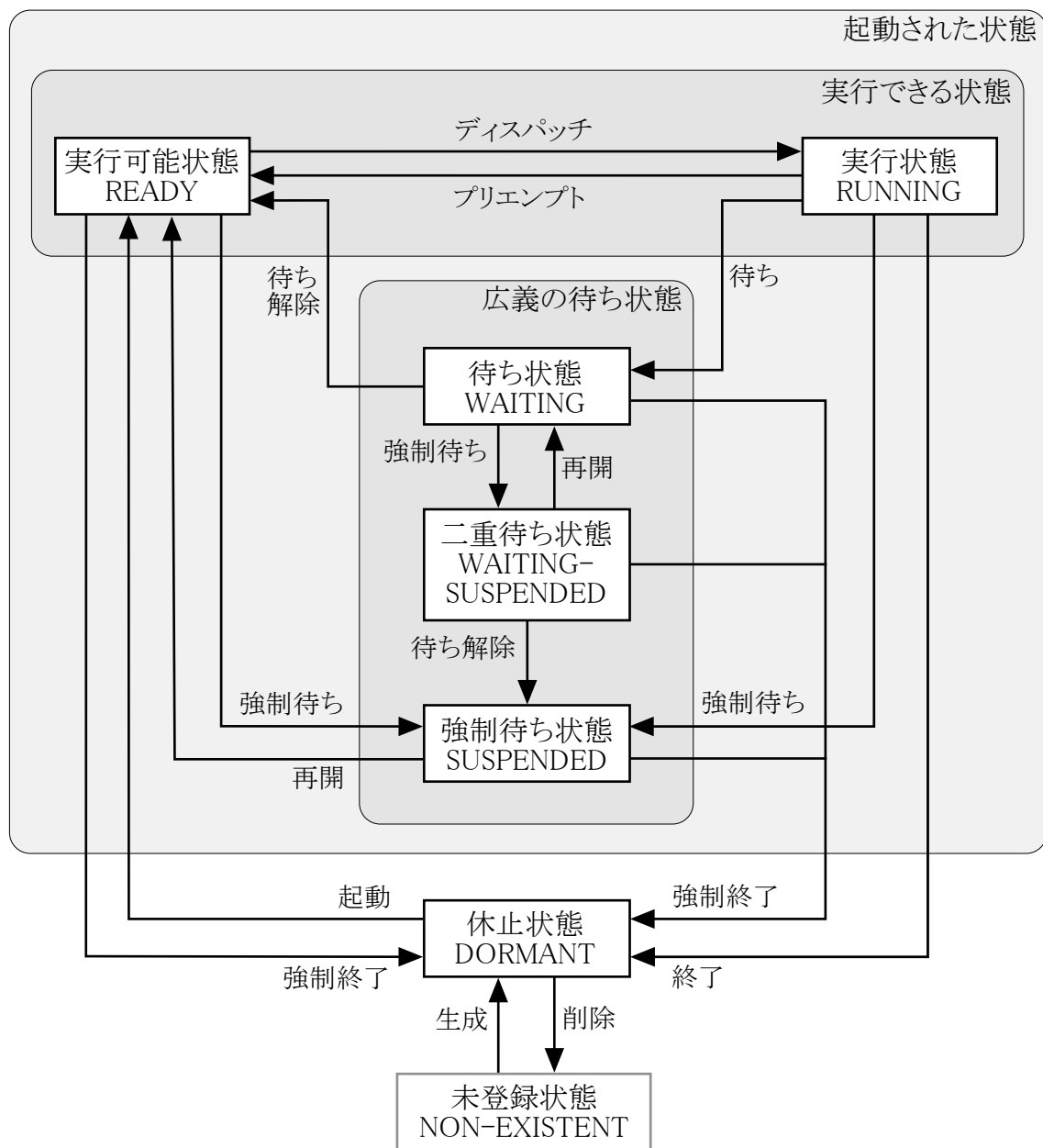


図2-2. タスクの状態遷移

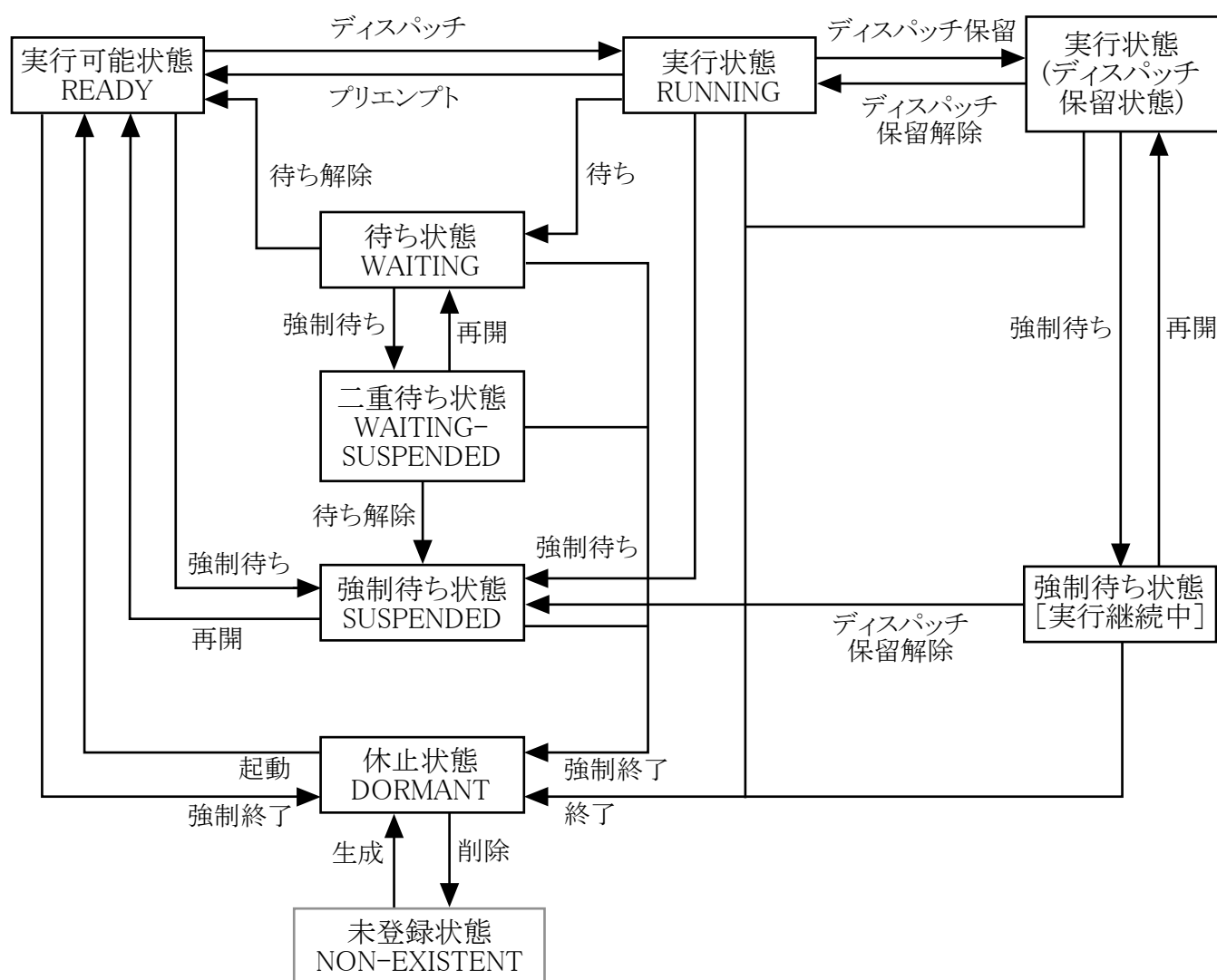


図2-3. 過渡的な状態も含めたタスクの状態遷移

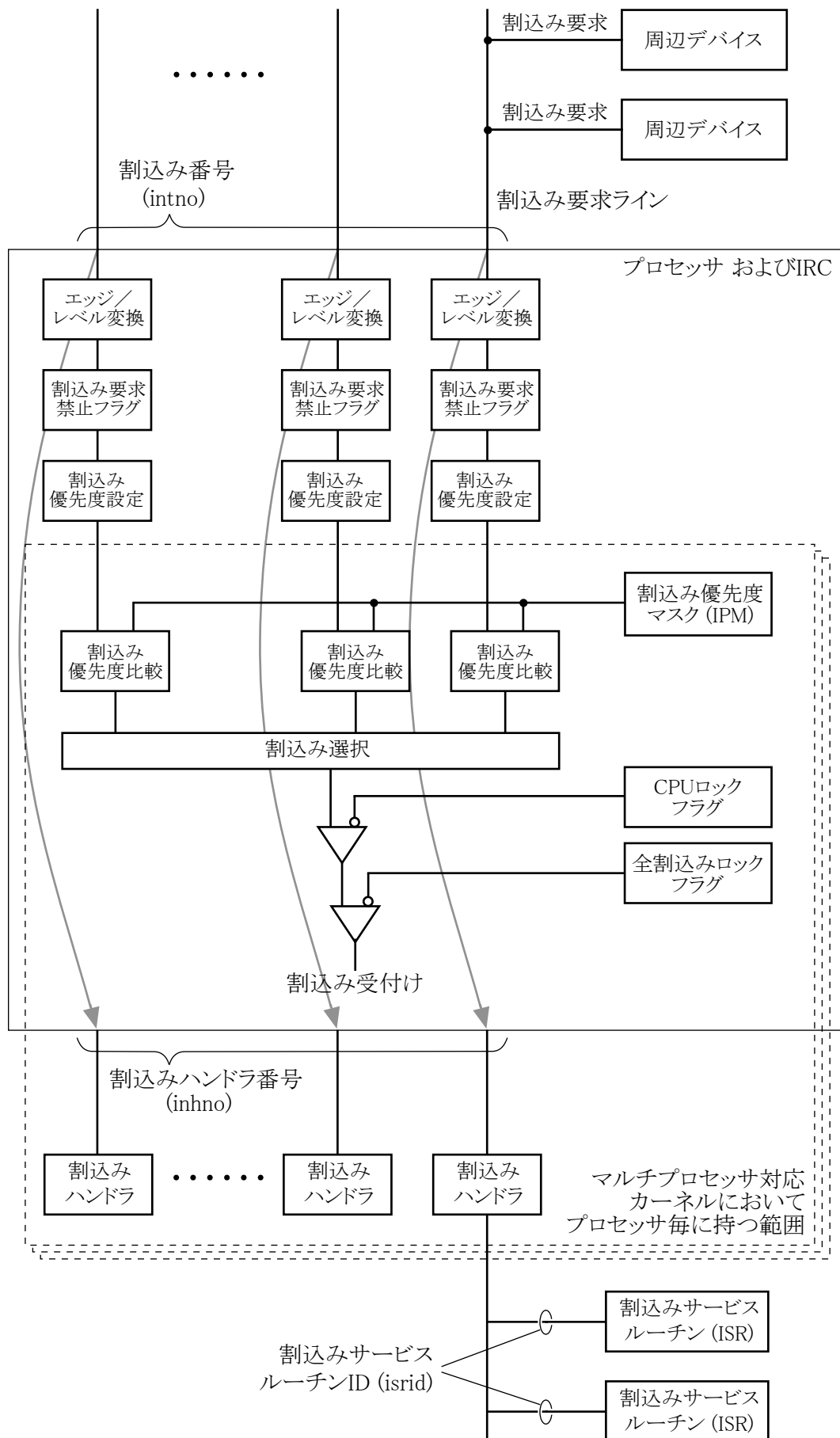


図2-4. TOPPERS標準割り込み処理モデルの概念図

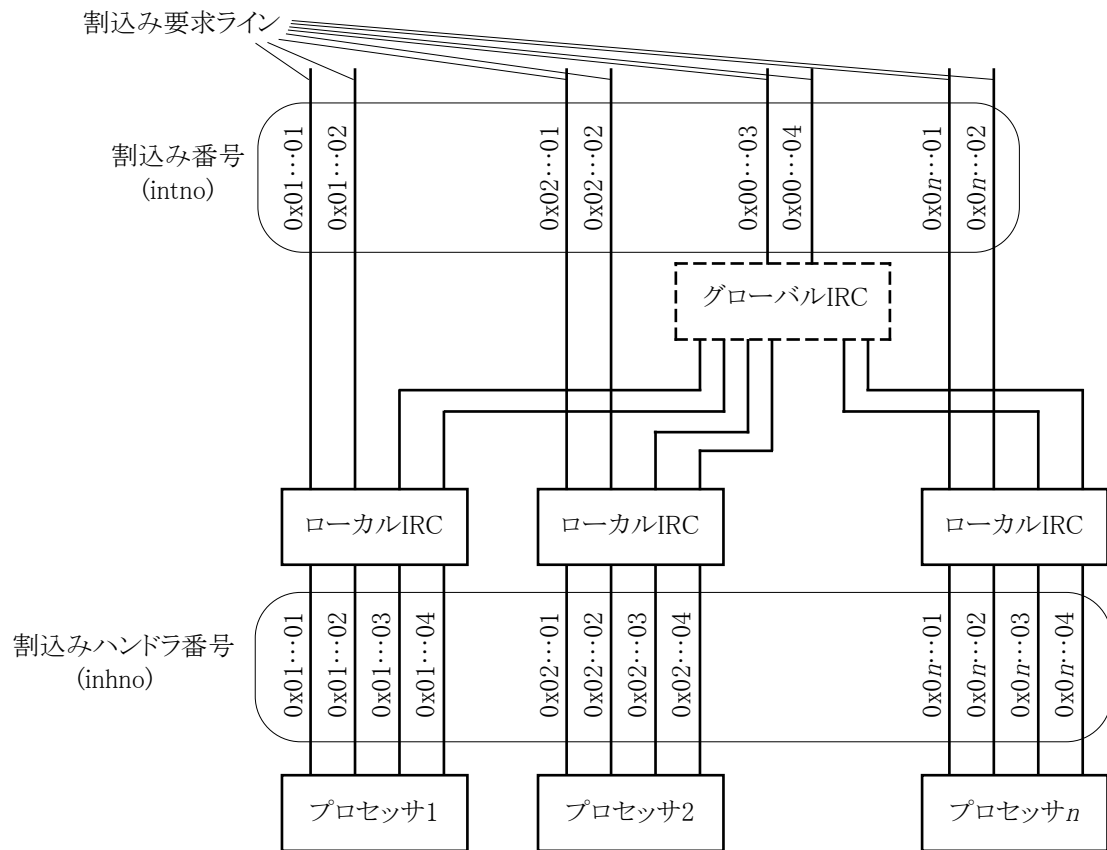


図2-5. マルチプロセッサ対応カーネルにおける割り込み番号と割り込みハンドラ番号

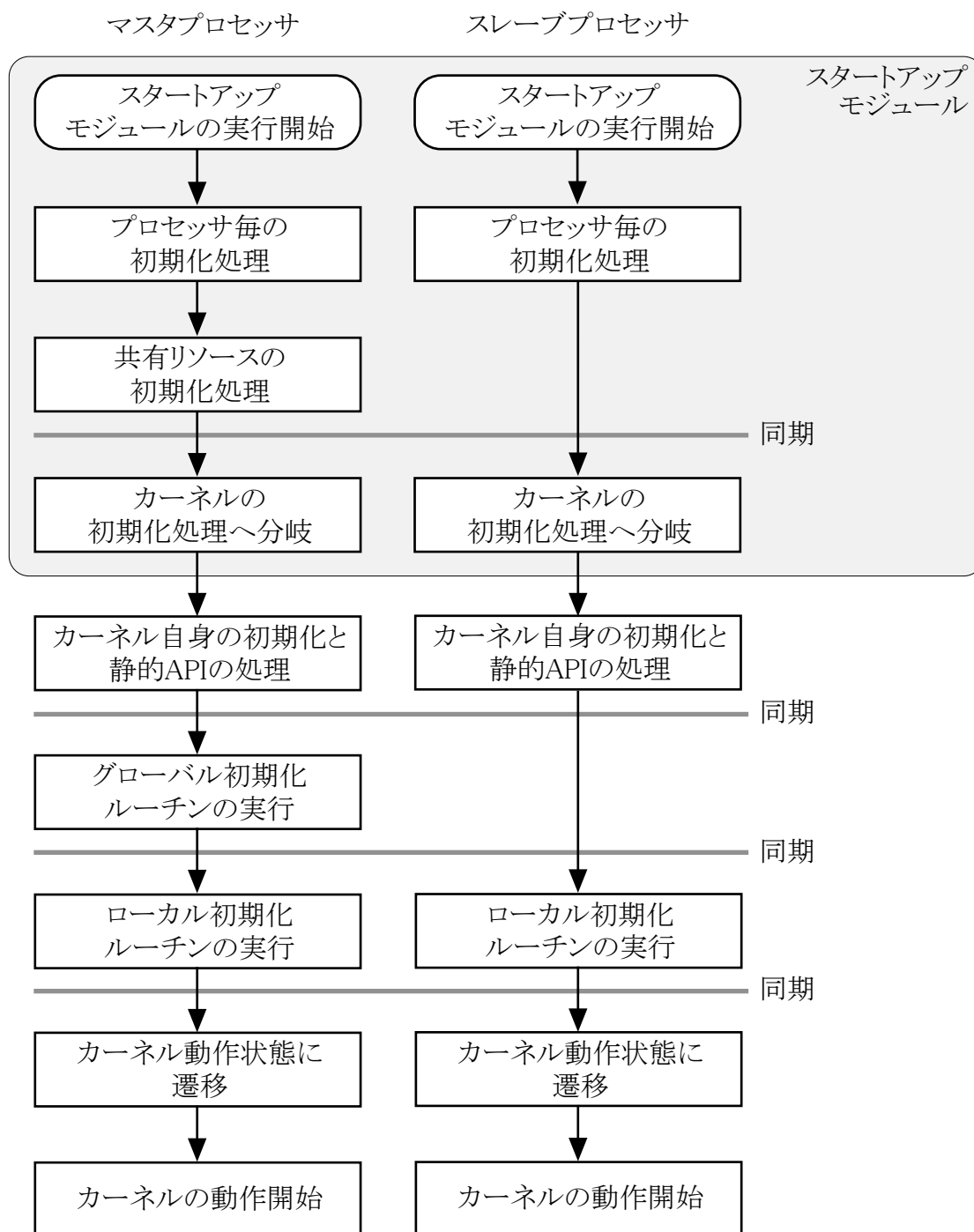


図2-6. マルチプロセッサ対応カーネルにおけるシステム初期化の流れ

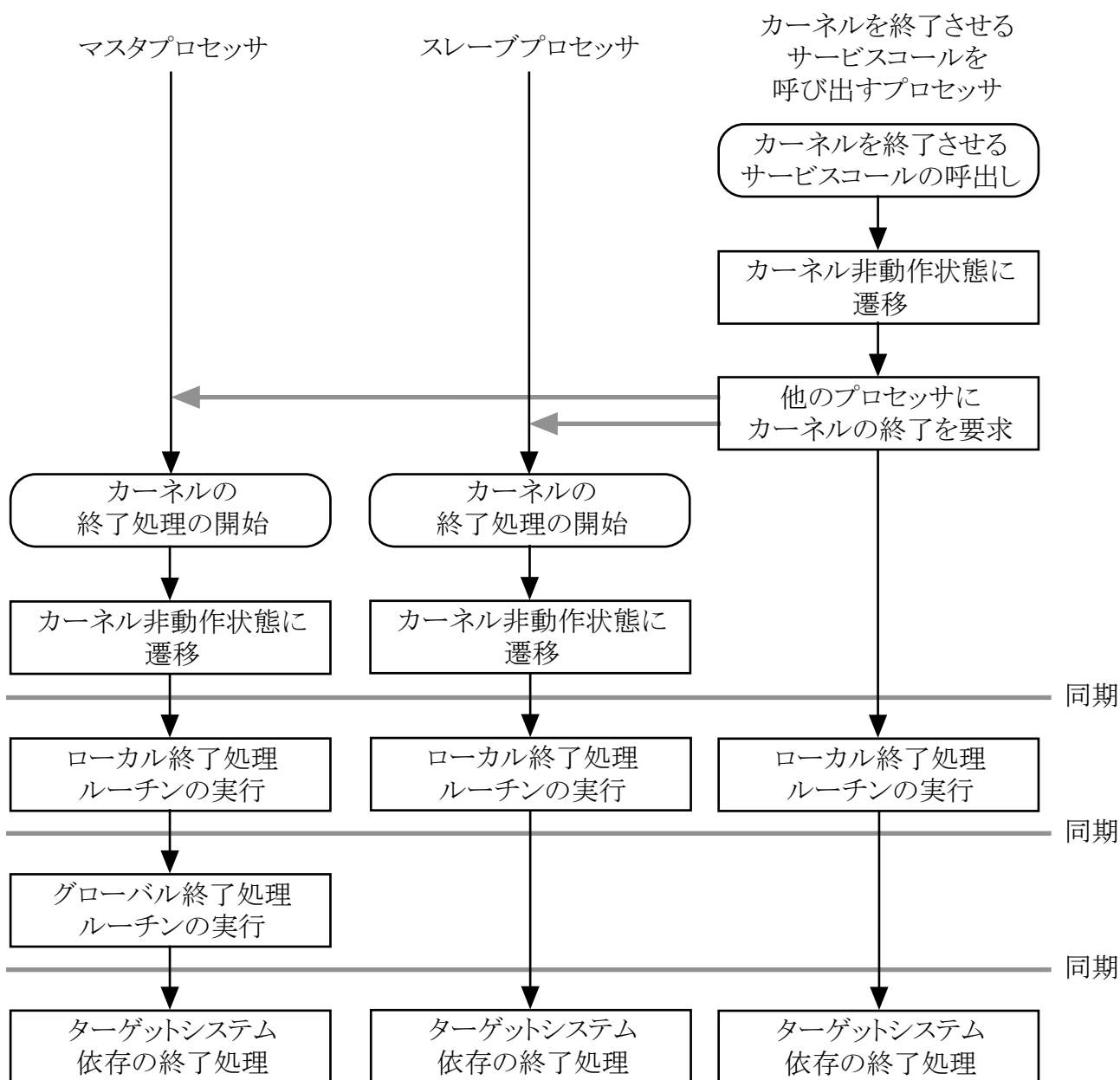


図2-7. マルチプロセッサ対応カーネルにおけるシステム終了処理の流れ

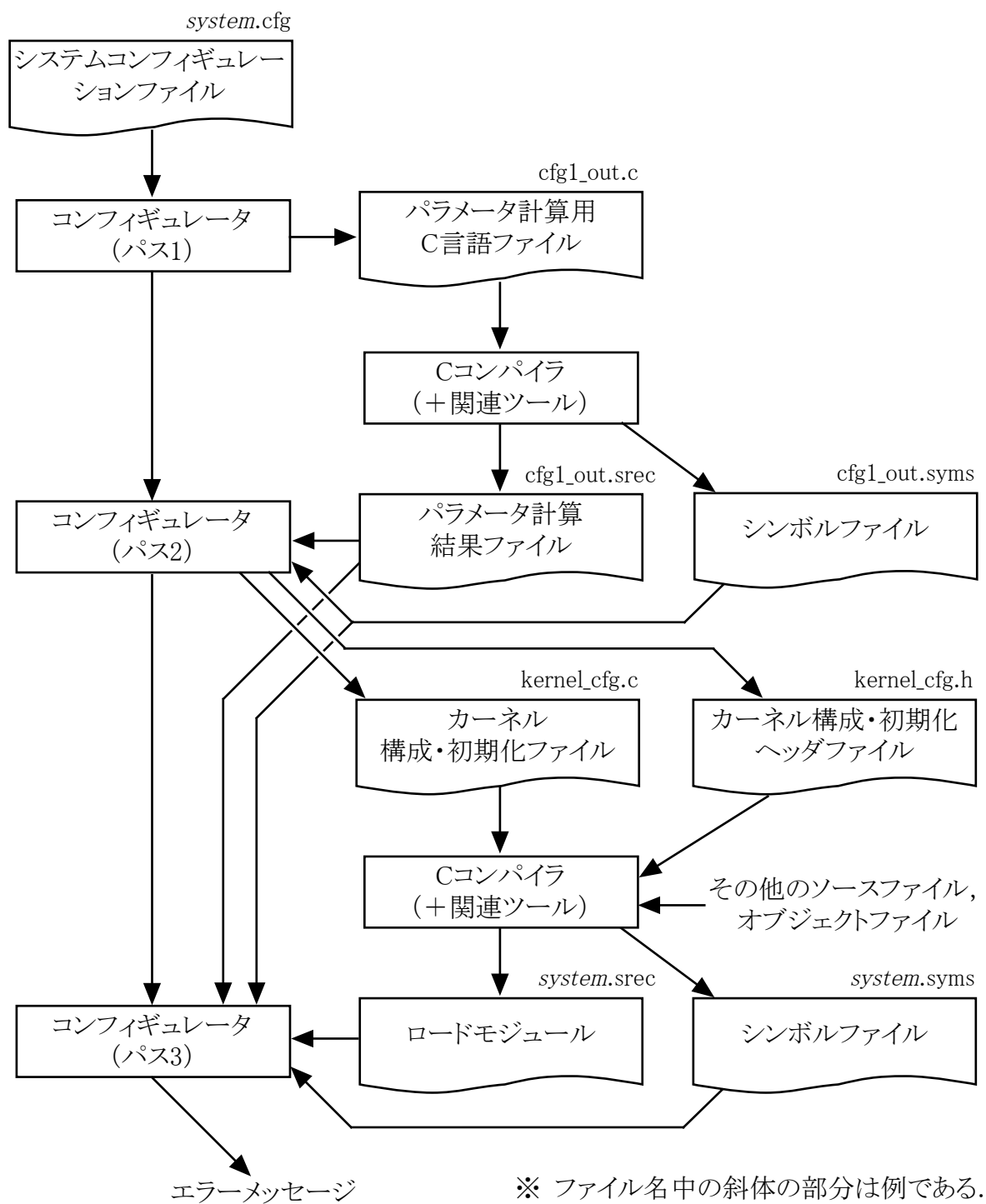
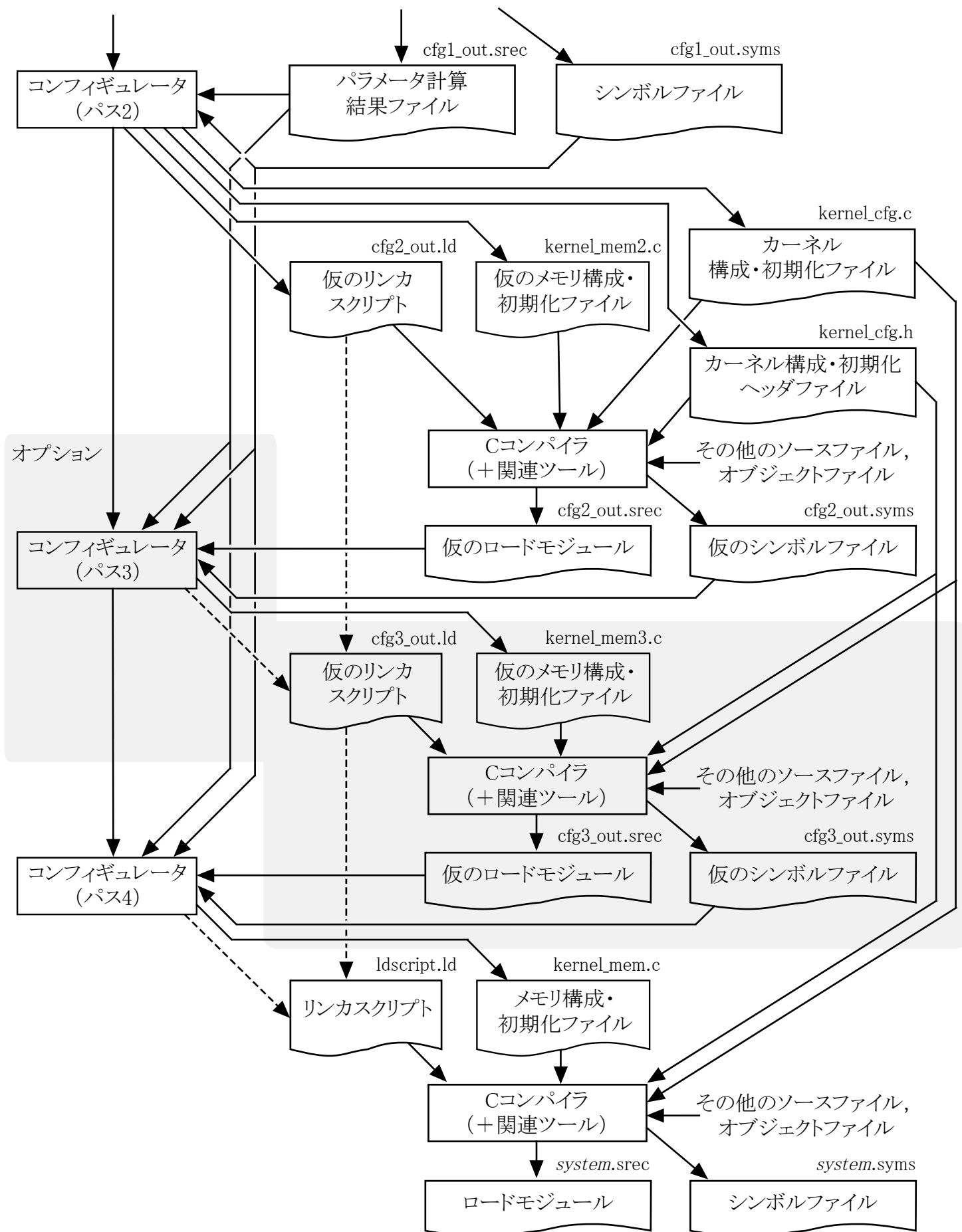


図2-8. コンフィギュレータの処理モデル



※ ファイル名中の斜体の部分は例である。

図2-9. 保護機能対応カーネルにおけるコンフィギュレータの処理モデル