

## TOPPERS新世代カーネル統合仕様書

バージョン: Release 1.7.0

最終更新: 2014年11月17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。

この仕様書に準拠している各カーネルのバージョンは、次の通りである。

TOPPERS/ASPカーネル	Release 1.9.1
TOPPERS/FMPカーネル	Release 1.2.0
TOPPERS/HRP2カーネル	Release 2.2.0
TOPPERS/SSPカーネル	Release 1.2.0

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) 2006-2014 by Embedded and Real-Time Systems Laboratory  
Graduate School of Information Science, Nagoya Univ., JAPAN  
Copyright (C) 2006-2014 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。

---

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

101	2.5.8 ディスパッチ保留状態
102	2.5.9 カーネル管理外の状態
103	2.5.10 処理単位の開始・終了とシステム状態
104	2.6 タスクの状態遷移とスケジューリング規則
105	2.6.1 基本的なタスク状態
106	2.6.2 タスクの状態遷移
107	2.6.3 タスクのスケジューリング規則
108	2.6.4 待ち行列と待ち解除の順序
109	2.6.5 タスク例外処理マスク状態と待ち禁止状態
110	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
111	2.6.7 制約タスク
112	2.7 割込み処理モデル
113	2.7.1 割込み処理の流れ
114	2.7.2 割込み優先度
115	2.7.3 割込み要求ラインの属性
116	2.7.4 割込みを受け付ける条件
117	2.7.5 割込み番号と割込みハンドラ番号
118	2.7.6 マルチプロセッサにおける割込み処理
119	2.7.7 カーネル管理外の割込み
120	2.7.8 カーネル管理外の割込みの設定方法
121	2.8 CPU例外処理モデル
122	2.8.1 CPU例外処理の流れ
123	2.8.2 CPU例外ハンドラから呼び出せるサービスコール
124	2.8.3 エミュレートされたCPU例外ハンドラ
125	2.8.4 カーネル管理外のCPU例外
126	2.9 システムの初期化と終了
127	2.9.1 システム初期化手順
128	2.9.2 システム終了手順
129	2.10 オブジェクトの登録とその解除
130	2.10.1 ID番号で識別するオブジェクト
131	2.10.2 オブジェクト番号で識別するオブジェクト
132	2.10.3 識別番号を持たないオブジェクト
133	2.10.4 オブジェクト生成に必要なメモリ領域
134	2.10.5 オブジェクトが属する保護ドメインの設定
135	2.10.6 オブジェクトが属するクラスの設定
136	2.10.7 オブジェクトの状態参照
137	2.11 オブジェクトのアクセス保護
138	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
139	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
140	2.11.3 デフォルトのアクセス許可ベクタ
141	2.11.4 アクセス許可ベクタの設定
142	2.11.5 カーネルの管理領域のアクセス保護
143	2.11.6 ユーザタスクのユーザスタック領域
144	2.12 システムコンフィギュレーション手順
145	2.12.1 システムコンフィギュレーションファイル
146	2.12.2 静的APIの文法とパラメータ
147	2.12.3 保護ドメインの指定
148	2.12.4 クラスの指定
149	2.12.5 コンフィギュレータの処理モデル
150	2.12.6 静的APIのパラメータに関するエラー検出

151	2.12.7 オブジェクトのID番号の指定
152	2.13 TOPPERSネーミングコンベンション
153	2.13.1 モジュール識別名
154	2.13.2 データ型名
155	2.13.3 関数名
156	2.13.4 変数名
157	2.13.5 定数名
158	2.13.6 マクロ名
159	2.13.7 静的API名
160	2.13.8 ファイル名
161	2.13.9 モジュール内部の名称の衝突回避
162	2.14 TOPPERS共通定義
163	2.14.1 TOPPERS共通ヘッダファイル
164	2.14.2 TOPPERS共通データ型
165	2.14.3 TOPPERS共通定数
166	2.14.4 TOPPERS共通エラーコード
167	2.14.5 TOPPERS共通マクロ
168	2.14.6 TOPPERS共通構成マクロ
169	2.15 カーネル共通定義
170	2.15.1 カーネルヘッダファイル
171	2.15.2 カーネル共通定数
172	2.15.3 カーネル共通マクロ
173	2.15.4 カーネル共通構成マクロ
174	
175	第3章 システムインタフェースレイヤAPI仕様
176	
177	3.1 システムインタフェースレイヤの概要
178	3.2 SILヘッダファイル
179	3.3 全割込みロック状態の制御
180	3.4 SILスピンロック
181	3.5 微少時間待ち
182	3.6 エンディアンの取得
183	3.7 メモリ空間アクセス関数
184	3.8 I/O空間アクセス関数
185	3.9 プロセッサIDの参照
186	
187	第4章 カーネルAPI仕様
188	
189	4.1 タスク管理機能
190	4.2 タスク付属同期機能
191	4.3 タスク例外処理機能
192	4.4 同期・通信機能
193	4.4.1 セマフォ
194	4.4.2 イベントフラグ
195	4.4.3 データキュー
196	4.4.4 優先度データキュー
197	4.4.5 メールボックス
198	4.4.6 ミューテックス
199	4.4.7 メッセージバッファ
200	4.4.8 スピンロック

201	4.5 メモリプール管理機能
202	4.5.1 固定長メモリプール
203	4.6 時間管理機能
204	4.6.1 システム時刻管理
205	4.6.2 周期ハンドラ
206	4.6.3 アラームハンドラ
207	4.6.4 オーバランハンドラ
208	4.7 システム状態管理機能
209	4.8 メモリオブジェクト管理機能
210	4.9 割込み管理機能
211	4.10 CPU例外管理機能
212	4.11 拡張サービスコール管理機能
213	4.12 システム構成管理機能
214	
215	第5章 リファレンス
216	
217	5.1 サービスコール一覧
218	5.2 静的API一覧
219	5.3 データ型
220	5.3.1 TOPPERS共通データ型
221	5.3.2 カーネルの使用データ型
222	5.3.3 カーネルの使用データ形式
223	5.4 定数とマクロ
224	5.4.1 TOPPERS共通定数
225	5.4.2 TOPPERS共通マクロ
226	5.4.3 カーネル共通定数
227	5.4.4 カーネル共通マクロ
228	5.4.5 カーネルの機能毎の定数
229	5.4.6 カーネルの機能毎のマクロ
230	5.5 構成マクロ
231	5.5.1 TOPPERS共通構成マクロ
232	5.5.2 カーネル共通構成マクロ
233	5.5.3 カーネルの機能毎の構成マクロ
234	5.6 エラーコード一覧
235	5.7 機能コード一覧
236	5.8 カーネルオブジェクトに対するアクセスの種別
237	5.9 ターゲット定義事項一覧
238	5.10 省略名の元になった英語
239	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
240	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
241	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
242	5.11 バージョン履歴
243	
244	
245	○仕様書で用いる記述項目と記号
246	
247	この仕様書では、以下の記述項目を用いる。
248	
249	【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。
250	

【～～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のカーネルにおける追加仕様を規定する。

【～～仕様との関係】の項では、この仕様と、 $\mu$ ITRON4.0仕様または $\mu$ ITRON4.0/PX仕様との違いについて説明する。

【未決定事項】の項では、この仕様書の現時点のバージョンでは、決定されずに残っている事項について記述する。

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

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

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

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

【機能】の項では、サービスコールおよび静的APIの機能を規定する。

TOPPERS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。

301 [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。  
302 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。

303  
304 [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  
305 対応でないカーネルでは、このAPIはサポートされない。

306  
307 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。

308  
309 [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  
310 ラーは発生しない。

311  
312 [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  
313 は発生しない。

314  
315 [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  
316 いカーネルでは、このエラーは発生しない。

317  
318 [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  
319 いカーネルでは、このエラーは発生しない。

320  
321  
322 ○タグの付与方法

323  
324 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  
325 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  
326 る。

- 327  
328 ・対象ソフトウェアの実装に対する要求事項や制限事項  
329 ・対象ソフトウェアの仕様に対する一般要求事項  
330 ・対象ソフトウェアの動作環境に対する要求事項  
331 ・ターゲット定義の規定

332  
333 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  
334 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  
335 与しない。

336  
337 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）。

338  
339 NGKIxxxx TOPPERS新世代カーネル全体を対象とした記述  
340 ASPSxxxx TOPPERS/ASPカーネルを対象とした記述  
341 FMPSxxxx TOPPERS/FMPカーネルを対象とした記述  
342 HRPSxxxx TOPPERS/HRP2カーネルを対象とした記述  
343 SSPSxxxx TOPPERS/SSPカーネルを対象とした記述  
344 ASSSxxxx TOPPERS/ASP Safetyカーネルを対象とした記述

345  
346 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  
347 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  
348 に対して、タグYYYYxxxxを参照する場合には、[YYYYxxxx]と表記する。

349  
350

## 第1章 TOPPERS新世代カーネルの概要

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

### 1.1 TOPPERS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、 $\mu$ ITRON4.0仕様およびその保護機能拡張（ $\mu$ 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。

$\mu$ ITRON4.0仕様は1999年に、 $\mu$ 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、 $\mu$ 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) $\mu$ ITRON4.0仕様をベースに拡張・改良を加える

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

#### (2) ソフトウェアの再利用性を重視する

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



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

### (3) 高信頼・安全なシステム構築を支援する

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

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

### (4) アプリケーションシステム構築に必要な機能は積極的に取り込む

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

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

## 1.3 TOPPERS/ASPカーネルの適用対象領域と仕様設計方針

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

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

ASPカーネルの主な適用対象は、高い信頼性・安全性・リアルタイム性を要求される組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。

452

#### 453 1.4 TOPPERS/FMPカーネルの適用対象領域と仕様設計方針

454

455 TOPPERS/FMPカーネル (FMPは、Flexible Multiprocessor Profileの略。以下、  
456 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  
457 イムカーネルである。

458

459 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  
460 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  
461 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  
462 リコードを実行できることが必要である。

463

464 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  
465 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  
466 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  
467 で動的な負荷分散を実現することが可能である。

468

469 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  
470 が不要な範囲に留める。

471

#### 472 1.5 TOPPERS/HRP2カーネルの適用対象領域と仕様設計方針

473

474 TOPPERS/HRP2カーネル (HRPは、High Reliable system Profileの略。2はバー  
475 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  
476 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  
477 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。

478

479 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  
480 権モードを備え、メモリ保護のためにMMU (Memory Management Unit) または  
481 MPU (Memory Protection Unit) を持つプロセッサを用いたシステムである。  
482 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  
483 ズ (バイナリコード) が数百KB以上のシステムである。

484

485 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  
486 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  
487 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  
488 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。

489

#### 490 1.6 TOPPERS/SSPカーネルの適用対象領域と仕様設計方針

491

492 TOPPERS/SSPカーネル (SSPは、Smallest Set Profileの略。以下、SSPカーネル)  
493 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  
494 絞り込んだリアルタイムカーネルである。

495

496 SSPカーネルの機能は、 $\mu$ ITRON4.0仕様の「仕様準拠の最低条件」の考え方を踏  
497 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  
498 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  
499 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  
500 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】.

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

$\mu$  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.  $\mu$  ITRON4.0/PX仕様は、 $\mu$  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。

#### 【 $\mu$ ITRON4.0仕様との関係】

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

### 2.1.4 想定するハードウェア構成

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、フリースタンドイング環境で用いることを想定して規定している【NGKI0004】。

ただし、C90の規定に加えて、以下のことを仮定している。

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

## 2.2 APIの構成要素とコンベンション

### 2.2.1 APIの構成要素

#### (1) サービスコール

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

#### (2) コールバック

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

#### (3) 静的API

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

#### (4) 構成マクロ

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

### 2.2.2 パラメータとリターンパラメータ

サービスコールやコールバックに渡すデータをパラメータ（parameter）、それ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】。

701 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  
702 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

703 【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  
704 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  
705 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  
706 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】.

707

708 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  
709 ト (packet) と呼ぶ.

710

711 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  
712 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、  
713 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  
714 れることはなく、別の目的に使用できる【NGKI0011】.

715

## 716 2.2.3 返値とエラーコード

717

718 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  
719 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  
720 (=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  
721 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  
722 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  
723 を、エラーコード (error code) と呼ぶ.

724

725 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  
726 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  
727 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  
728 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  
729 意されている【NGKI0015】.

730

731 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  
732 に定める (「2.14.4 TOPPERS共通エラーコード」の節を参照) 【NGKI0016】.  
733 サービスコールおよびコールバックの機能説明中の「E\_XXXXエラーとなる」ま  
734 たは「E\_XXXXエラーが返る」という記述は、メインエラーコードとして  
735 E\_XXXXが返ることを意味する.

736

737 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  
738 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

739 【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  
740 用するシステムサービスのAPI仕様において規定する【NGKI0018】.

741

742 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  
743 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  
744 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  
745 ルの機能説明にその旨を記述する【NGKI0020】.

746

747 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  
748 のエラーを示すエラーコードが返る【NGKI0021】.

749

750 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】。

752

753 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  
754 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  
755 「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。

756

## 757 2.2.4 機能コード

758

759 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  
760 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。  
761 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  
762 割り付け、拡張サービスコールには正の値を用いる【NGKI0023】。

763

## 764 2.2.5 ヘッドファイル

765

766 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。

767

768 ヘッドファイルは、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  
769 ように対処されている。具体的には、ヘッドファイルの先頭で特定の識別子  
770 (例えば、kernel.hなら"TOPPERS\_KERNEL\_H")がマクロ定義され、ヘッドファ  
771 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  
772 レクティブが付加されている【NGKI0024】。

773

## 774 2.3 主な概念

775

### 776 2.3.1 オブジェクトと処理単位

777

#### 778 (1) オブジェクト

779

780 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  
781 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  
782 源を、カーネルオブジェクト (kernel object) と呼ぶ。

783

784 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  
785 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  
786 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  
787 【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  
788 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  
789 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  
790 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  
791 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】。

792

793 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  
794 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  
795 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  
796 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  
797 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  
798 【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  
799 指定する【NGKI0030】。

800



## 801 (2) 処理単位

802

803 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  
804 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  
805 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、  
806 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。

807

808 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  
809 ることを実行開始（start）と呼ぶ。

810

811 拡張情報（extended information）は、処理単位が呼び出される時にパラメー  
812 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  
813 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】。

814

## 815 (3) タスク

816

817 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）  
818 と呼ぶ。タスクは、処理単位の1つである。

819

820 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、  
821 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  
822 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  
823 る。

824

825 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。  
826 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  
827 告する【NGKI0033】。

828

## 829 【補足説明】

830

831 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  
832 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  
833 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  
834 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」  
835 は、異なるものを指す。

836

## 837 (4) ディスパッチとスケジューリング

838

839 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  
840 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  
841 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  
842 （scheduling）と呼ぶ。

843

844 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  
845 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  
846 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  
847 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  
848 れた時点で、ディスパッチが起こる【NGKI0034】。

849

## 850 (5) 割込みとCPU例外

851  
852 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  
853 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  
854 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  
855 (CPU exception) と呼ぶ。  
856  
857 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  
858 求が受け付けられるのを抑止することを、割込みのマスク (mask interrupt)  
859 または割込みの禁止 (disable interrupt) という。マスクが解除された時点で、  
860 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  
861 【NGKI0035】。  
862  
863 マスクすることができない割込みを、NMI (non-maskable interrupt) と呼ぶ。  
864  
865 【 $\mu$  ITRON4.0仕様との関係】  
866  
867  $\mu$  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  
868 語を定義した。  
869  
870 (6) タイムイベントとタイムイベントハンドラ  
871  
872 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  
873 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、  
874 タイムイベントハンドラ (time event handler) と呼ぶ。  
875  
876 2.3.2 サービスコールとパラメータ  
877  
878 (1) 優先順位と優先度  
879  
880 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  
881 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  
882 高い処理単位が実行される【NGKI0036】。  
883  
884 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  
885 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  
886 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  
887 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  
888 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  
889 【NGKI0038】。  
890  
891 (2) システム時刻と相対時間  
892  
893 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  
894 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  
895 【NGKI0039】。システム時刻は、タイムティック (time tick) を通知するため  
896 のタイマ割込みが発生する毎に更新される【NGKI0040】。  
897  
898 イベントを発生させる時刻を指定する場合には、基準時刻 (base time) からの  
899 相対時間 (relative time) によって指定する【NGKI0041】。基準時刻は、別に  
900 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

901 る【NGKI0042】.

902

903 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  
904 同一、すなわちミリ秒とする【NGKI0043】. 相対時間には、少なくとも、16ビット  
905 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  
906 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  
907 きるとは限らない【NGKI0044】. 相対時間に指定できる最大値は、構成マクロ  
908 TMAX\_RELTIMに定義されている【NGKI0045】.

909

910 イベントが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  
911 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  
912 なる【NGKI0046】. ただし、基準時刻を定めるサービスコールを呼び出した時に  
913 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  
914 (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  
915 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  
916 されない【NGKI0047】.

917

918 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  
919 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  
920 した時刻となる【NGKI0049】.

921

922 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  
923 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  
924 【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  
925 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  
926 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、  
927 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  
928 【NGKI0051】.

929

930 【補足説明】

931

932 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  
933 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  
934 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  
935 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  
936 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  
937 ある.

938

939 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  
940 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  
941 降のタイムティックでイベントの処理が行われる.

942

943 【 $\mu$  ITRON4.0仕様との関係】

944

945 相対時間 (RELTIM型) とシステム時刻 (SYSTIM型) の時間単位は、 $\mu$  ITRON4.0  
946 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  
947 時間の解釈について、より厳密に規定した.

948

949 TMAX\_RELTIMは、 $\mu$  ITRON4.0仕様に規定されていないカーネル構成マクロである.

950

951 (3) タイムアウトとポーリング  
952  
953 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  
954 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  
955 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  
956 エラーが返る【NGKI0052】。  
957  
958 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  
959 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  
960 【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  
961 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  
962 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  
963 した後となる。  
964  
965 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  
966 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  
967 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  
968 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  
969 敗したサービスコールからは、E\_TMOUTエラーが返る【NGKI0055】。  
970  
971 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  
972 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  
973 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  
974 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  
975 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  
976 舞いは、該当するサービスコール毎に規定する【NGKI0058】。  
977  
978 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  
979 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)  
980 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】。  
981  
982 【補足説明】  
983  
984 [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  
985 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。  
986 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  
987 副作用があるかをサービスコール毎に規定する。  
988  
989 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  
990 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  
991 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  
992 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  
993 ビスコールと全く同じ振舞いをする。  
994  
995 【μITRON4.0仕様との関係】  
996  
997 タイムアウト時間 (TMO型) の時間単位は、μITRON4.0仕様では実装定義として  
998 いたが、この仕様ではミリ秒と規定した。  
999  
1000 【仕様決定の理由】

- 1001  
1002 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  
1003 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  
1004 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  
1005 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  
1006 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  
1007 れに該当）を呼び出すことはできないという原則 [NGKI0175] と [NGKI0179]  
1008 があるためである。  
1009  
1010 (4) ノンブロッキング  
1011  
1012 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  
1013 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  
1014 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  
1015 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  
1016 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  
1017 則は適用されない【NGKI0061】。  
1018  
1019 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  
1020 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  
1021 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  
1022 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  
1023 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  
1024 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】。  
1025  
1026 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  
1027 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  
1028 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】。  
1029  
1030 【補足説明】  
1031  
1032 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  
1033 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  
1034 していない。  
1035  
1036 2.3.3 保護機能  
1037  
1038 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、  
1039 保護機能対応カーネルにのみ適用される。  
1040  
1041 (1) アクセス保護  
1042  
1043 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  
1044 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  
1045 を防ぐアクセス保護機能を提供する【NGKI0066】。  
1046  
1047 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  
1048 が対象 (object) ということになる。  
1049  
1050 (2) メモリオブジェクト

1051  
1052 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  
1053 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  
1054 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。  
1055 メモリオブジェクトは、互いに重なりあうことはない【NGKI0068】。  
1056  
1057 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  
1058 えると、先頭番地がオブジェクト番号となる。  
1059  
1060 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  
1061 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】。  
1062  
1063 (3) 保護ドメイン  
1064  
1065 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  
1066 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  
1067 号によって識別する【NGKI0071】。  
1068  
1069 カーネルオブジェクトは、たかだか1つの保護ドメインに属する。処理単位は、  
1070 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  
1071 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  
1072 【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、  
1073 無所属のカーネルオブジェクト (independent kernel object) と呼ぶ。  
1074  
1075 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  
1076 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  
1077 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  
1078 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  
1079 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。  
1080  
1081 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  
1082 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  
1083 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】。  
1084 これは、【NGKI0073】の原則の例外となっている。  
1085  
1086 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  
1087 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、  
1088 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  
1089 【NGKI0076】。  
1090  
1091 (4) カーネルドメインとユーザドメイン  
1092  
1093 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  
1094 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  
1095 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  
1096 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】。  
1097 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」  
1098 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  
1099 る。  
1100

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

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

この仕様では、システムに登録できるユーザドメインの数は、32個以下に制限する【NGKI0084】。これを超える数のユーザドメインに登録した場合には、コンフィギュレータがエラーを報告する【NGKI0085】。

#### 【補足説明】

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

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

#### 【 $\mu$ ITRON4.0/PX仕様との関係】

$\mu$  ITRON4.0/PX仕様のシステムドメイン (system domain) は、現時点ではサポートしない。システムドメインは、それに属する処理単位が、プロセッサの特権モードで実行され、カーネルオブジェクトに対するアクセスを制限することができる保護ドメインである。

#### (5) システムタスクとユーザタスク

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

#### 【補足説明】

特権モードで実行されるタスクをシステムタスク、非特権モードで実行されるタスクをユーザタスクと定義する方法もあるが、ユーザタスクであっても、サービスク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。

$\mu$  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ぶことになる。

#### (6) アクセス許可パターン

あるカーネルオブジェクトに対するある種別のアクセスが、どの保護ドメインに属する処理単位に許可されているかを表現するビットパターンを、アクセス許可パターン (access permission pattern) と呼ぶ。アクセス許可パターンの各ビットは、1つのユーザドメインに対応する【NGKI0086】。カーネルドメイン

1151 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  
1152 ビットは用意されていない。

1153  
1154 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  
1155 (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  
1156 可されていることを表す【NGKI0087】。そのため、2つのアクセス許可パターンの  
1157 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  
1158 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  
1159 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  
1160 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。

1161  
1162 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  
1163 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)  
1164 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  
1165 可することを示すアクセス許可パターンを表す定数 (TACP\_KERNEL) と、すべての  
1166 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  
1167 定数 (TACP\_SHARED) が用意されている【NGKI0089】。

#### 1168 1169 (7) アクセス許可ベクタ

1170  
1171 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、  
1172 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  
1173 【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  
1174 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  
1175 (access permission vector) と呼び、次のように定義されるデータ型  
1176 (ACVCT) で保持する【NGKI0091】。

```
1177
1178     typedef struct acvct {
1179         ACPTN    acptn1;    /* 通常操作1のアクセス許可パターン */
1180         ACPTN    acptn2;    /* 通常操作2のアクセス許可パターン */
1181         ACPTN    acptn3;    /* 管理操作のアクセス許可パターン */
1182         ACPTN    acptn4;    /* 参照操作のアクセス許可パターン */
1183     } ACVCT;
```

#### 1184 1185 【補足説明】

1186  
1187 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  
1188 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。

#### 1189 1190 【μ ITRON4.0/PX仕様との関係】

1191  
1192 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  
1193 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  
1194 めている。

#### 1195 1196 (8) サービスコールの呼出し方法

1197  
1198 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  
1199 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、  
1200 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】。



1201  
1202 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  
1203 さい。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  
1204 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。  
1205 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  
1206 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。

1207  
1208 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  
1209 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、  
1210 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  
1211 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  
1212 【NGKI0093】。

1213  
1214 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  
1215 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  
1216 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)  
1217 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  
1218 【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  
1219 ように記述すればよい。

1220  
1221       ercd = SVC\_CALL(act\_tsk)(tskid);  
1222

#### 1223 【補足説明】

1224  
1225 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。  
1226 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  
1227 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  
1228 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  
1229 れない。

#### 1230 1231 (9) ユーザドメインから行える処理に対する制限

1232  
1233 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  
1234 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  
1235 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  
1236 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  
1237 る。

1238  
1239 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】。

1240  
1241       ・ 指定できる最高のタスク優先度  
1242

1243 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  
1244 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  
1245 ことができる【NGKI0532】。

#### 1246 1247 2.3.4 マルチプロセッサ対応

1248  
1249 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  
1250 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。

1251

1252 (1) クラス

1253

1254 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  
1255 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  
1256 【NGKI0095】。

1257

1258 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  
1259 【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  
1260 時に決定し、登録後に変更することはできない【NGKI0097】。

1261

1262 【補足説明】

1263

1264 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  
1265 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  
1266 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  
1267 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  
1268 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  
1269 が典型的である。

1270

1271 【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  
1272 さない。

1273

- 1274 ・オーバランハンドラ
- 1275 ・拡張サービスコール
- 1276 ・グローバル初期化ルーチン
- 1277 ・グローバル終了処理ルーチン

1278

1279 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  
1280 ラスのみをサポートしているとみなすこともできる。

1281

1282 (2) プロセッサ

1283

1284 たかだか1つの処理単位のみを同時に実行できるハードウェアの単位を、プロセッ  
1285 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  
1286 別する【NGKI0098】。

1287

1288 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)  
1289 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】。

1290

1291 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  
1292 セッサ (master processor) と呼び、システムに1つ存在する【NGKI0100】。ど  
1293 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

1294 【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  
1295 (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  
1296 とスレーブプロセッサの振舞いに違いはない【NGKI0102】。

1297

1298 (3) 処理単位の割付けとマイグレーション

1299

1300 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。

1304

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。

1308

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

1312

(4) クラスの持つ属性とカーネルオブジェクト

1314

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

1318

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、次の通りである【NGKI0106】。

1321

- 1322     ・ 初期割付けプロセッサ
- 1323     ・ 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サを含む)
- 1324     ・ ATT\_MOD/ATA\_MODによって、オブジェクトモジュールに含まれる標準のセクションが配置されるメモリリージョン (標準メモリリージョン)
- 1325     ・ 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) の配置場所
- 1326     ・ その他の管理情報 (ロック単位など)

1330

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】。

1332

【仕様決定の理由】

1335

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

1340

(5) ローカルタイマ方式とグローバルタイマ方式

1342

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タイマ方式と、システム全体で1つのシステム時刻を持つグローバルタイマ方式の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、ターゲット定義である【NGKI0108】。

1346

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能は、カーネルでは用意しない。

1350

1351  
1352 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  
1353 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  
1354 【NGKI0111】。  
1355  
1356  
1357 **【補足説明】**  
1358  
1359 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。  
1360  
1361 **【未決定事項】**  
1362  
1363 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  
1364 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、  
1365 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。  
1366  
1367 2.3.5 その他  
1368  
1369 (1) オブジェクトモジュール  
1370  
1371 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  
1372 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  
1373 ブジェクトモジュールである。  
1374  
1375 (2) メモリリージョン  
1376  
1377 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  
1378 た連続したメモリ領域をメモリリージョン (memory region) と呼ぶ。  
1379  
1380 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  
1381 ンを識別する文字列を、メモリリージョン名と呼ぶ。  
1382  
1383 **【補足説明】**  
1384  
1385 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  
1386 囲という一般的な意味で使っている。  
1387  
1388 (3) 標準のセクション  
1389  
1390 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  
1391 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、  
1392 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】。  
1393  
1394 (4) 保護ドメイン毎の標準セクション  
1395  
1396 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  
1397 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  
1398 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  
1399 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  
1400 for each protection domain)。保護ドメイン毎の標準セクションのセクショ

ン名は、ターゲット定義で別に規定がない限りは、標準のセクション名と保護  
ドメイン名（カーネルドメインの場合は“kernel”，無所属の場合は“shared”）  
を“\_”でつないだものとする【NGKI0116】。例えば、カーネルドメインの  
“.text”セクションのセクション名は，“.text\_kernel”とする。

1405

## 1406 2.4 処理単位の種類と実行順序

1407

### 1408 2.4.1 処理単位の種類

1409

1410 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】。

1411

#### 1412 (a) タスク

1413 (a.1) タスク例外処理ルーチン

#### 1414 (b) 割込みハンドラ

1415 (b.1) 割込みサービスルーチン

1416 (b.2) タイムイベントハンドラ

#### 1417 (c) CPU例外ハンドラ

#### 1418 (d) 拡張サービスコール

#### 1419 (e) 初期化ルーチン

#### 1420 (f) 終了処理ルーチン

1421

1422 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  
1423 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  
1424 ある。

1425

#### 1426 【TOPPERS/ASPカーネルにおける規定】

1427

1428 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  
1429 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  
1430 と、オーバランハンドラ機能を追加することができる【ASPS0004】。

1431

#### 1432 【TOPPERS/FMPカーネルにおける規定】

1433

1434 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  
1435 ない【FMPS0002】。

1436

#### 1437 【TOPPERS/SSPカーネルにおける規定】

1438

1439 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  
1440 ビスコールをサポートしていない【SSPS0002】。

1441

### 1442 2.4.2 処理単位の実行順序

1443

1444 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  
1445 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  
1446 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  
1447 る。

1448

1449 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  
1450 スク間では、高い優先度を持つ方が優先順位が高く、同じ優先度を持つタスク

間では、先に実行できる状態となった方が優先順位が高い【NGKI0119】。詳しくは、「2.6.3 タスクのスケジューリング規則」の節を参照すること。

タスク例外処理ルーチンの優先順位は、例外が要求されたタスクと同じであるが、タスクよりも先に実行される【NGKI0120】。

割込みハンドラの優先順位は、ディスパッチャの優先順位よりも高い【NGKI0121】。割込みハンドラ間では、高い割込み優先度を持つ方が優先順位が高く、同じ割込み優先度を持つ割込みハンドラ間では、先に実行開始された方が優先順位が高い【NGKI0122】。同じ割込み優先度を持つ割込みハンドラ間での実行開始順序は、この仕様では規定しない。詳しくは、「2.7.2 割込み優先度」の節を参照すること。

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

CPU例外ハンドラの優先順位は、CPU例外がタスクまたはタスク例外処理ルーチンで発生した場合には、ディスパッチャの優先順位と同じであるが、ディスパッチャよりも先に実行される【NGKI0124】。CPU例外がその他の処理単位で発生した場合には、CPU例外ハンドラの優先順位は、その処理単位の優先順位と同じであるが、その処理単位よりも先に実行される【NGKI0125】。

拡張サービスコールの優先順位は、それを呼び出した処理単位と同じであるが、それを呼び出した処理単位よりも先に実行される【NGKI0126】。

初期化ルーチンは、カーネルの動作開始前に、システムコンフィギュレーションファイル中に初期化ルーチンを登録する静的APIを記述したのと同じ順序で実行される【NGKI0127】。終了処理ルーチンは、カーネルの動作終了後に、終了処理ルーチンを登録する静的APIを記述したのと逆の順序で実行される【NGKI0128】。

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

【仕様決定の理由】

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

### 2.4.3 カーネル処理の不可分性

カーネルのサービスコール処理やディスパッチャ、割込みハンドラとCPU例外ハ

ンドラの入口処理と出口処理などのカーネル処理は不可分に実行されるのが基本である。実際には、カーネル処理の途中でアプリケーションが実行される場合はあるが、アプリケーションがサービスコールを用いて観測できる範囲で、カーネル処理が不可分に実行された場合と同様に振る舞うのが原則である【NGKI0133】。これを、カーネル処理の不可分性という。

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

#### 【補足説明】

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

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

#### 2.4.4 処理単位を実行するプロセッサ

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

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

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

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

1551 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  
1552 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。

	初期割付けプロセッサ	割付け可能プロセッサ
1554	-----	
1555	タスク（タスク例外処理	○
1556	ルーチンを含む）	○
1557	-----	
1558	割込みハンドラ	○
1559	割込みサービスルーチン	—
1560	周期ハンドラ	○
1561	アラームハンドラ	○
1562	-----	
1563	CPU例外ハンドラ	○
1564	-----	
1565	ローカル初期化ルーチン	○
1566	ローカル終了処理ルーチン	—
1567	-----	
1568		

1569  
1570 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  
1571 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  
1572 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  
1573 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  
1574 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  
1575 チンとグローバル終了処理ルーチンは、マスタプロセッサによって実行される  
1576 【NGKI0142】。

## 1577 2.5 システム状態とコンテキスト

### 1579 2.5.1 カーネル動作状態と非動作状態

1581  
1582 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、  
1583 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  
1584 （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  
1585 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  
1586 態かカーネル非動作状態のいずれかの状態を取る【NGKI0143】。

1587  
1588 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  
1589 る【NGKI0144】。

1590  
1591 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  
1592 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  
1593 【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  
1594 場合の動作は、保証されない【NGKI0146】。

1595  
1596 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  
1597 ネル非動作状態のいずれかの状態を取る【NGKI0147】。

### 1599 2.5.2 タスクコンテキストと非タスクコンテキスト

1600



1601 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  
1602 ど）をコンテキストと呼ぶ。

1603

1604 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  
1605 ンテキストと非タスクコンテキストに分類される【NGKI0148】。

1606

1607 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  
1608 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  
1609 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  
1610 ストに分類される【NGKI0150】。

1611

1612 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  
1613 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  
1614 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  
1615 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  
1616 【NGKI0152】。

1617

1618 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  
1619 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  
1620 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  
1621 域を用いて実行される【NGKI0154】。

1622

1623 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  
1624 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  
1625 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  
1626 【NGKI0156】。いずれも、呼び出した場合にはE\_CTXエラーとなる【NGKI0157】。

1627

#### 1628 2.5.3 カーネルの振舞いに影響を与える状態

1629

1630 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  
1631 状態として、次の状態を持つ【NGKI0158】。

1632

- 1633 ・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）
- 1634 ・CPUロックフラグ（CPUロック状態とCPUロック解除状態）
- 1635 ・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）
- 1636 ・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）

1637

1638 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  
1639 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  
1640 とができる【NGKI0159】。

1641

#### 1642 2.5.4 全割込みロック状態と全割込みロック解除状態

1643

1644 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  
1645 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  
1646 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、  
1647 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。

1648

1649 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  
1650 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

1651 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  
1652 で、その他のサービスコールを呼び出した場合の動作は、保証されない

1653 【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  
1654 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】。

1655

1656 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  
1657 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  
1658 込みロック解除状態のいずれかの状態を取る。

1659

## 1660 2.5.5 CPUロック状態とCPUロック解除状態

1661

1662 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  
1663 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】。  
1664 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  
1665 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  
1666 みがマスクされ、ディスパッチが保留される【NGKI0167】。

1667

1668 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】。

1669

- 1670 ・システムインタフェースレイヤのAPI
- 1671 ・loc\_cpu/iloc\_cpu, unl\_cpu/iunl\_cpu
- 1672 ・unl\_spn/iunl\_spn (マルチプロセッサ対応カーネルのみ)
- 1673 ・dis\_int, ena\_int
- 1674 ・sns\_yyy
- 1675 ・xsns\_yyy (CPU例外ハンドラからのみ)
- 1676 ・get\_utm
- 1677 ・ext\_tsk, ext\_ker
- 1678 ・prb\_mem (保護機能対応カーネルのみ)
- 1679 ・cal\_svc (保護機能対応カーネルのみ)

1680

1681 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  
1682 となる【NGKI0169】。

1683

1684 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  
1685 【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  
1686 態のいずれかの状態を取る。

1687

## 1688 【補足説明】

1689

1690 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  
1691 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。

1692

1693 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  
1694 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  
1695 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、  
1696 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  
1697 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。

1698

## 1699 2.5.6 割込み優先度マスク

1700

1701 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  
1702 マスクを持つ【NGKI0171】。割込み優先度マスクがTIPM\_ENAALL (=0) の時は、  
1703 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  
1704 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  
1705 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  
1706 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  
1707 マスクが全解除でない状態と呼ぶ。

1708

1709 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  
1710 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  
1711 とはできない。呼び出した場合には、E\_CTXエラーとなる【NGKI0175】。

1712

1713 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  
1714 持つ【NGKI0176】。

1715

#### 1716 2.5.7 ディスパッチ禁止状態とディスパッチ許可状態

1717

1718 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  
1719 【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  
1720 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  
1721 パッチ禁止状態では、ディスパッチは保留される。

1722

1723 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  
1724 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  
1725 出した場合には、E\_CTXエラーとなる【NGKI0179】。

1726

1727 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  
1728 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  
1729 ディスパッチ許可状態のいずれかの状態を取る。

1730

#### 1731 【補足説明】

1732

1733 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  
1734 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。  
1735 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  
1736 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  
1737 現することはできない。

1738

#### 1739 2.5.8 ディスパッチ保留状態

1740

1741 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  
1742 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  
1743 【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。

1744

1745 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  
1746 かそうでない状態のいずれかの状態を取る【NGKI0182】。

1747

#### 1748 【補足説明】

1749

1750 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

1751        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。

1752

## 1753        2.5.9 カーネル管理外の状態

1754

1755        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  
1756        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  
1757        （「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  
1758        外の状態と呼ぶ。

1759

1760        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，  
1761        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  
1762        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  
1763        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  
1764        ビスコールを呼び出した場合の動作は，保証されない【NGKI0544】．

1765

1766        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  
1767        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  
1768        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  
1769        になるのは，特権モードで実行している間に限られる【NGKI0547】．

1770

## 1771        2.5.10 処理単位の開始・終了とシステム状態

1772

1773        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  
1774        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  
1775        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  
1776        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  
1777        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  
1778        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。

1779

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

1782

### 1783        【タスク】 【NGKI0183】

1784        実行開始条件	解除	全解除	許可
1785        実行開始時処理	そのまま	そのまま	そのまま
1786        終了前	原則解除(*1)	原則全解除(*1)	原則許可(*1)
1787        終了時処理	解除する	全解除する	許可する

1788

### 1789        【タスク例外処理ルーチン】 【NGKI0184】

1790        実行開始条件	解除	全解除	任意
1791        実行開始時処理	そのまま	そのまま	そのまま
1792        リターン前	原則解除(*1)	原則全解除(*1)	元に戻す
1793        リターン時処理	解除する	全解除する	元に戻す(*4)

1794

### 1795        【カーネル管理の割込みハンドラ】 【NGKI0185】

### 1796        【割込みサービスルーチン】 【NGKI0186】

### 1797        【タイムイベントハンドラ】 【NGKI0187】

1798        実行開始条件	解除	自優先度より低い	任意
1799        実行開始時処理	そのまま	自優先度に(*2)	そのまま
1800        リターン前	原則解除(*1)	変更不可(*3)	変更不可(*3)

1801	リターン時処理	解除する	元に戻す(*5)	そのまま
1802	-----			
1803	【CPU例外ハンドラ】 【NGKI0188】			
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(*6)	そのまま	そのまま
1806	リターン前	原則元に(*1)	変更不可(*3)	変更不可(*3)
1807	リターン時処理	元に戻す	元に戻す(*5)	そのまま

1808	-----			
1809	【拡張サービスコール】 【NGKI0189】			
1810	実行開始条件	任意	任意	任意
1811	実行開始時処理	そのまま	そのまま	そのまま
1812	リターン前	任意	任意	任意
1813	リターン時処理	そのまま	そのまま	そのまま

1814 -----

1815

1816 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）  
 1817 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、  
 1818 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  
 1819 態が設定されるため、支障がないことを意味する。

1820

1821 「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  
 1822 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  
 1823 の場合にはタイマ割込みの割込み優先度、オーバランハンドラの場合にはオー  
 1824 バランタイマ割込みの割込み優先度に変更することを意味する。

1825

1826 「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  
 1827 が用意されていないことを示す。

1828

1829 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  
 1830 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  
 1831 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  
 1832 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  
 1833 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  
 1834 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  
 1835 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  
 1836 るとは限らない）【NGKI0190】。

1837

1838 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  
 1839 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  
 1840 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  
 1841 【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  
 1842 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  
 1843 【NGKI0192】。

1844

1845 【補足説明】

1846

1847 マルチプロセッサ対応カーネルにおいて、タスクがタスク例外処理ルーチンを  
 1848 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  
 1849 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。

1850

**【仕様決定の理由】**

保護機能対応カーネルにおいて、タスク例外処理ルーチンからのリターン時にディスパッチ禁止フラグを元に戻す処理(\*4)が、タスクにディスパッチ禁止フ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上の状態を書き換えることで、許可していない状態変更を起こしてしまうことを防止するためである。

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

CPU例外ハンドラの実行開始時には、CPUロックフラグは変更されない(\*6)ことから、CPUロック状態でCPU例外が発生した場合、CPU例外ハンドラの実行開始直後はCPUロック状態となっている。CPUロック状態でCPU例外が発生した場合、起動されるCPU例外ハンドラはカーネル管理外のCPU例外ハンドラであり（xsns\_dpn, 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ロック状態を解除しようとした場合の動作は保証されない。ただし、保証されないにも関わらずiunl\_cpuを呼び出した場合も考えられるため、リターン時には元に戻すこととしている。

**2.6 タスクの状態遷移とスケジューリング規則****2.6.1 基本的なタスク状態**

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

**(a) 実行できる状態 (runnable)**

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

**(a.1) 実行状態 (running)**

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

**(a.2) 実行可能状態 (ready)**

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

**(b) 休止状態 (dormant)**

1901

1902

1903

1904

1905

1906

1907

1908

1909

1910

1911

1912

1913

1914

1915

1916

1917

1918

1919

1920

1921

1922

1923

1924

1925

1926

1927

1928

1929

1930

1931

1932

1933

1934

1935

1936

1937

1938

1939

1940

1941

1942

1943

1944

1945

1946

1947

1948

1949

1950

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

(c) 広義の待ち状態 (blocked)

タスクが、処理の途中で実行を止められている状態。タスクが広義の待ち状態にある時には、タスクの実行を再開するための情報（実行再開番地やレジスタの内容など）は保存されており、タスクが実行を再開する時には、広義の待ち状態に遷移する前の状態に戻される【NGKI0195】。広義の待ち状態は、さらに、（狭義の）待ち状態、強制待ち状態、二重待ち状態に分類される。

(c.1) （狭義の）待ち状態 (waiting)

タスクが何らかの条件が揃うのを待つために、自ら実行を止めている状態。

(c.2) 強制待ち状態 (suspended)

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

(c.3) 二重待ち状態 (waiting-suspended)

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

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

(d) 未登録状態 (non-existent)

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

カーネルによっては、これらのタスク状態以外に、過渡的な状態が存在する場合がある【NGKI0196】。過渡的な状態については、「2.6.6 ディスパッチ保留状態で実行中のタスクに対する強制待ち」の節を参照すること。

【TOPPERS/ASPカーネルにおける規定】

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

**【TOPPERS/FMPカーネルにおける規定】**

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】。上記のタスク状態以外の過渡的な状態として、タスクが強制待ち状態〔実行継続中〕になることがある【FMPS0004】。詳しくは、「2.6.6 ディスパッチ保留状態で実行中のタスクに対する強制待ち」の節を参照すること。

**【TOPPERS/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ブトしたという。逆に、実行可能状態に遷移したタスクは、プリエンブトされたという。

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】**



2001 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  
2002 に遷移する）時のことをいう。

2003

### 2004 2.6.3 タスクのスケジューリング規則

2005

2006 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】。  
2007 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  
2008 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。

2009

2010 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  
2011 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  
2012 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  
2013 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、  
2014 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  
2015 リングされる。ただし、サービスクールと呼出しにより、同じ優先度を持つタ  
2016 スク間の優先順位を変更することも可能である【NGKI0206】。

2017

2018 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  
2019 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  
2020 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  
2021 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  
2022 まる【NGKI0208】。

2023

2024 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  
2025 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、  
2026 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  
2027 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  
2028 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  
2029 に存在する。

2030

### 2031 2.6.4 待ち行列と待ち解除の順序

2032

2033 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  
2034 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  
2035 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  
2036 クトと呼ぶ。

2037

2038 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  
2039 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  
2040 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  
2041 【NGKI0211】。

2042

2043 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  
2044 最後につながる【NGKI0212】。それに対してタスクの優先度順の待ち行列に  
2045 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  
2046 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  
2047 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  
2048 につながる【NGKI0214】。

2049

2050 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

2051 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  
2052 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  
2053 舞いをするかは、待ち行列毎に規定される【NGKI0215】。

2054  
2055 (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  
2056 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  
2057 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  
2058 たしていれば、先に待ち解除される。

2059  
2060 (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  
2061 【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  
2062 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。

2063  
2064 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  
2065 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）、  
2066 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。  
2067 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  
2068 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  
2069 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  
2070 じ処理が繰り返される【NGKI0219】。

## 2071 2072 2.6.5 タスク例外処理マスク状態と待ち禁止状態

2073  
2074 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  
2075 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  
2076 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  
2077 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）、  
2078 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  
2079 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  
2080 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】。  
2081 これらの状態を、タスク例外処理マスク状態と呼ぶ。

2082  
2083 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  
2084 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  
2085 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  
2086 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、  
2087 サービスコールはE\_RLWAIエラーとなる【NGKI0222】。

2088  
2089 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  
2090 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  
2091 【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  
2092 （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  
2093 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  
2094 【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っ  
2095 ても、待ち禁止状態を解除することができる【NGKI0225】。

## 2096 2097 【仕様決定の理由】

2098  
2099 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  
2100 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

2101 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  
2102 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  
2103 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、拡張サービスコー  
2104 ルのための例外処理が行われないことになる。

2105

2106 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  
2107 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  
2108 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  
2109 ムスタック上に不要な情報が残ってしまう。

2110

2111 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  
2112 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  
2113 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  
2114 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  
2115 る。

2116

2117 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、  
2118 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、  
2119 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、  
2120 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  
2121 外処理ルーチンを実行させることができる。

2122

2123 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  
2124 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  
2125 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  
2126 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  
2127 されないことになる。

2128

2129 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  
2130 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/  
2131 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  
2132 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  
2133 ンを実行させることができる。

2134

2135 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  
2136 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  
2137 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。

2138

2139 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  
2140 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  
2141 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  
2142 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  
2143 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  
2144 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  
2145 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。

2146

2147 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  
2148 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  
2149 るように実装すべきである。

2150

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

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

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

2.6.6 ディスパッチ保留状態で実行中のタスクに対する強制待ち

ディスパッチ保留状態において、実行状態のタスクを強制待ち状態へ遷移させるサービスコールを呼び出した場合、実行状態のタスクの切換えは、ディスパッチ保留状態が解除されるまで保留される【NGKI0226】。

この間、それまで実行状態であったタスクは、実行状態と強制待ち状態の間の過渡的な状態にあると考える【NGKI0227】。この状態を、強制待ち状態〔実行継続中〕と呼ぶ。一方、ディスパッチ保留状態が解除された後に実行すべきタスクは、実行可能状態にとどまる【NGKI0228】。

タスクが強制待ち状態〔実行継続中〕にある時に、ディスパッチ保留状態が解除されると、ただちにディスパッチが起り、タスクは強制待ち状態に遷移する【NGKI0229】。

過渡的な状態も含めたタスクの状態遷移を図2-3に示す【NGKI0230】。

タスクが強制待ち状態〔実行継続中〕である時の扱いは次の通りである。

(a) プロセッサを占有して実行を継続する。

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

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

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

(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。

- 2201 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  
2202 待ち状態であるものと扱う【NGKI0234】。
- 2203
- 2204 【TOPPERS/ASPカーネルにおける規定】
- 2205
- 2206 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  
2207 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  
2208 ち状態 [実行継続中] になることはない【ASPS0008】。
- 2209
- 2210 【TOPPERS/FMPカーネルにおける規定】
- 2211
- 2212 FMPカーネルでは、ディスパッチ保留状態において実行状態のタスクを強制待ち  
2213 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  
2214 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】。
- 2215
- 2216 【TOPPERS/HRP2カーネルにおける規定】
- 2217
- 2218 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  
2219 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  
2220 待ち状態 [実行継続中] になることはない【HRPS0004】。
- 2221
- 2222 【TOPPERS/SSPカーネルにおける規定】
- 2223
- 2224 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  
2225 制待ち状態 [実行継続中] になることもない【SSPS0005】。
- 2226
- 2227 【補足説明】
- 2228
- 2229 この仕様では、ディスパッチ保留状態において、実行状態のタスクを強制終了  
2230 させるサービスコールはサポートしていない。そのため、実行状態と休止状態  
2231 の間の過渡的な状態は存在しない。
- 2232
- 2233 2.6.7 制約タスク
- 2234
- 2235 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  
2236 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  
2237 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  
2238 は以下の機能制限がある。
- 2239
- 2240 (a) 広義の待ち状態に入ることができない【NGKI0235】。
- 2241
- 2242 (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】。
- 2243
- 2244 (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  
2245 先順位の回転 (rot\_rdq/irot\_rdq) を行うことができない【NGKI0237】。
- 2246
- 2247 (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  
2248 できない【NGKI0238】。
- 2249
- 2250 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

2251 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  
2252 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  
2253 ことができる【NGKI0240】。

2254

2255 **【未決定事項】**

2256

2257 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、  
2258 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、  
2259 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  
2260 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  
2261 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  
2262 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、  
2263 今後の課題である。

2264

2265 **【TOPPERS/ASPカーネルにおける規定】**

2266

2267 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  
2268 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  
2269 る【ASPS0010】。

2270

2271 **【TOPPERS/FMPカーネルにおける規定】**

2272

2273 FMPカーネルでは、制約タスクをサポートしていない【FMPS0006】。

2274

2275 **【TOPPERS/HRP2カーネルにおける規定】**

2276

2277 HRP2カーネルでは、制約タスクをサポートしていない【HRPS0005】。

2278

2279 **【TOPPERS/SSPカーネルにおける規定】**

2280

2281 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  
2282 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、  
2283 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  
2284 タック領域を、共有スタック領域と呼ぶ。

2285

2286 **【 $\mu$  ITRON4.0仕様との関係】**

2287

2288 制約タスクは、 $\mu$  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  
2289 ある。この仕様における制約タスクは、 $\mu$  ITRON4.0仕様の制約タスクよりも機  
2290 能制限が少なくなっている。

2291

## 2292 2.7 割込み処理モデル

2293

2294 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  
2295 理モデルに準拠している。

2296

2297 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、  
2298 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  
2299 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  
2300 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

2301 アで実現される。

2302

2303 【 $\mu$  ITRON4.0仕様との関係】

2304

2305 割込み処理モデルは、 $\mu$  ITRON4.0仕様から大幅に拡張している。

2306

## 2307 2.7.1 割込み処理の流れ

2308

2309 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、  
2310  
2311 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。  
2312

2313

2314 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、  
2315 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  
2316 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】。  
2317

2318

2319 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  
2320 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  
2321 ervice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  
2322 依存せず、割込みを要求したデバイスだけに依存して記述するのが原則である  
2323 【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  
2324 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  
2325 することができる【NGKI0246】。  
2326

2327

2328 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  
2329 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  
2330 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】。  
2330

2331 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  
2332 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  
2333 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  
2334 ントハンドラが呼び出される【NGKI0248】。  
2335

2336

2337 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  
2338 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】。  
2338

## 2339 2.7.2 割込み優先度

2340

2341 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  
2342 【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  
2343 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  
2344 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、  
2345 マスクされる。  
2346

2347

2348 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  
2349 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  
2350 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

2351       【NGKI0253】.

2352

2353       これらのことから、他の方法で割込みをマスクしていない限り、ある割込み要求の処理中は、それと同じかそれより低い割込み優先度を持つ割込み要求は受け付けられず、それより高い割込み優先度を持つ割込み要求は受け付けられることになる。つまり、割込み優先度は、多重割込みを制御するためのものと位置付けることができる。それに対して、同時に発生している割込み要求の中で、割込み優先度の高い割込み要求が先に受け付けられるとは限らない

2359       【NGKI0254】.

2360

2361       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、  
2362       【NGKI0037】の原則には従わず、-1から連続した負の値を用いる【NGKI0255】.

2363

2364       割込み優先度の段階数は、ターゲット定義である【NGKI0256】。プロセッサが割込み優先度マスクを実現するための機能を持たないか、実現するために大きいオーバ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を1にする（すなわち、多重割込みを許さない）場合がある。

2368

2369       【仕様決定の理由】

2370

2371       割込み優先度に-1から連続した負の値を用いるのは、割込み優先度とタスク優先度を比較できるようになることと、いずれの割込みもマスクしない割込み優先度マスクの値を0にできるためである。

2374

2375       2.7.3 割込み要求ラインの属性

2376

2377       各割込み要求ラインは、以下の属性を持つ。なお、1つの割込み要求ラインに複数のデバイスからの割込み要求が接続されている場合、それらの割込み要求は同一の属性を持つ【NGKI0257】。それらの割込み要求に別々の属性を設定することはできない。

2380

2381       (1) 割込み要求禁止フラグ

2382

2383       割込み要求ライン毎に、割込みをマスクするための割込み要求禁止フラグを持つ【NGKI0258】。割込み要求禁止フラグをセットすると、その割込み要求ラインによって伝えられる割込み要求はマスクされる【NGKI0259】。

2387

2388       プロセッサが割込み要求禁止フラグを実現するための機能を持たないか、実現するために大きいオーバーヘッドを生じる場合には、ターゲット定義で、割込み要求禁止フラグをサポートしない場合がある【NGKI0260】。また、プロセッサの持つ割込み要求禁止フラグの機能がこの仕様に合致しない場合には、ターゲット定義で、割込み要求禁止フラグをサポートしないか、振舞いが異なるものとする場合がある【NGKI0261】。

2394

2395       (2) 割込み優先度

2396

2397       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】。  
2398       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対して設定された割込み優先度のことである【NGKI0263】。

2400



## 2401 (3) トリガモード

2402

2403 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  
2404 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  
2405 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  
2406 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  
2407 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  
2408 設定できる場合もある【NGKI0266】。

2409

2410 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  
2411 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、トリガモード  
2412 の設定をサポートしない場合がある【NGKI0267】。

2413

2414 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  
2415 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  
2416 止フラグをクリアすることもできない【NGKI0269】。

2417

## 2418 【使用上の注意】

2419

2420 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  
2421 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、  
2422 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  
2423 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  
2424 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  
2425 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  
2426 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  
2427 実現すべきである。

2428

2429 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  
2430 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、  
2431 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、  
2432 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  
2433 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  
2434 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  
2435 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  
2436 Aからの割込み要求を取りこぼしてしまう。

2437

## 2438 2.7.4 割込みを受け付ける条件

2439

2440 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  
2441 【NGKI0270】。

2442

2443 (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

2444

2445 (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  
2446 在値よりも高い（優先度の値としては小さい）こと

2447

2448 (c) 全割込みロックフラグがクリアされていること

2449

2450 (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

2451 リアされていること

2452

2453 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  
2454 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  
2455 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。

2456

#### 2457 2.7.5 割込み番号と割込みハンドラ番号

2458

2459 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、  
2460 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  
2461 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】。  
2462 そのため、1から連続した正の値であるとは限らない。

2463

2464 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  
2465 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、  
2466 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  
2467 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  
2468 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  
2469 た正の値であるとは限らない。

2470

2471 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  
2472 一致する場合が多い）【NGKI0274】。

2473

2474 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  
2475 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  
2476 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、  
2477 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  
2478 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  
2479 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  
2480 係をカーネルが管理しているもののみである【NGKI0277】。

2481

#### 2482 2.7.6 マルチプロセッサにおける割込み処理

2483

2484 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  
2485 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。

2486

2487 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  
2488 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  
2489 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、  
2490 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  
2491 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  
2492 モード）はシステム全体で共通に持つ。

2493

2494 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  
2495 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  
2496 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  
2497 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  
2498 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  
2499 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  
2500 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

2501 である。グローバルIRCは、必ず備わっているとは限らない。

2502

2503 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  
2504 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  
2505 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  
2506 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、  
2507 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  
2508 のとする【NGKI0281】。

2509

2510 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  
2511 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  
2512 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  
2513 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  
2514 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）。

2515

2516 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  
2517 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  
2518 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  
2519 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】。

2520

2521 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  
2522 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  
2523 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  
2524 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  
2525 なければならない【NGKI0286】。

2526

2527 **【補足説明】**

2528

2529 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  
2530 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  
2531 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  
2532 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  
2533 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  
2534 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  
2535 とする（図2-5）。

2536

2537 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  
2538 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  
2539 スに属していなければならない。

2540

2541 **【使用上の注意】**

2542

2543 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  
2544 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  
2545 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。

2546

2547 2.7.7 カーネル管理外の割込み

2548

2549 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  
2550 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

2551 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、  
2552 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  
2553 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、  
2554 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】。  
2555  
2556 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  
2557 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  
2558 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、  
2559 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  
2560 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】。  
2561  
2562 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  
2563 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  
2564 ンドラをカーネル管理の割込みハンドラと呼ぶ。  
2565  
2566 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  
2567 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  
2568 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  
2569 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  
2570 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】。  
2571  
2572 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  
2573 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  
2574 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  
2575 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  
2576 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】。  
2577 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  
2578 場合の動作は、保証されない【NGKI0294】。  
2579  
2580 2.7.8 カーネル管理外の割込みの設定方法  
2581  
2582 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  
2583 ずれかが採用される【NGKI0295】。  
2584  
2585 (a-1) NMI以外にカーネル管理外の割込みを設けない  
2586 (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  
2587  
2588 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  
2589 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  
2590 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、  
2591 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  
2592 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  
2593 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  
2594 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  
2595 (TA\_NONKERNEL)を指定する【NGKI0298】。また、割込み要求ラインの属性を  
2596 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  
2597 【NGKI0299】。  
2598  
2599 (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  
2600

2601 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、  
2602 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  
2603 であることを示す割込みハンドラ属性 (TA\_NONKERNEL) を指定する

2604 【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  
2605 て設定する割込み優先度を、TMIN\_INTPRIよりも高い値とする【NGKI0301】。  
2606

2607 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  
2608 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】。  
2609 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  
2610 ことはできない【NGKI0303】。  
2611

## 2612 2.8 CPU例外処理モデル

2613 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  
2614 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  
2615 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  
2616 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、  
2617 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。  
2618  
2619

### 2620 2.8.1 CPU例外処理の流れ

2621 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  
2622 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  
2623 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  
2624 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  
2625 【NGKI0305】。  
2626

2627 CPU例外ハンドラの登録対象となるCPU例外を識別するための番号を、CPU例外ハン  
2628 ドラ番号と呼ぶ。CPU例外ハンドラ番号は、符号無し of 整数型であるEXCNO型  
2629 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  
2630 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  
2631 値であるとは限らない。  
2632

2633 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、  
2634 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  
2635 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、  
2636 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  
2637 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  
2638 ばならない【NGKI0308】。  
2639

2640 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  
2641 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  
2642 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  
2643 が考えられる【NGKI0310】。  
2644

2645 (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。  
2646

2647 (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  
2648 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  
2649 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  
2650

2651 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  
2652 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  
2653 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  
2654 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）

2655 【NGKI0311】.

2656

2657 (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  
2658 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）.

2659

2660 (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  
2661 る）.

2662

2663 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  
2664 生したコンテキストおよび状態に依存せずに常に行える【NGKI0312】. それに  
2665 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  
2666 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  
2667 テキストおよび状態に依存する【NGKI0313】.

2668

2669 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  
2670 舞いは、ターゲット定義である【NGKI0314】.

2671

2672 【使用上の注意】

2673

2674 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  
2675 入口処理で同じ原因でCPU例外が発生すると、CPU例外が繰り返して発生し、アプ  
2676 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  
2677 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  
2678 けない場合には、CPU例外入口処理を経由せずに、アプリケーションが用意した  
2679 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.

2680

2681 【補足説明】

2682

2683 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、  
2684 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  
2685 番号を含める方法を基本とする.

2686

2687 【 $\mu$  ITRON4.0仕様との関係】

2688

2689  $\mu$  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  
2690 ていない.

2691

2692 2.8.2 CPU例外ハンドラから呼び出せるサービスコール

2693

2694 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  
2695 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  
2696 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  
2697 ことができる【NGKI0315】.

2698

2699 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  
2700 チできる状態であった場合にfalseを返す【NGKI0316】. xsns\_dpnがfalseを返

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

xsns\_xpnは、CPU例外がタスクコンテキストで発生し、そのタスクがタスク例外処理ルーチンを実行できる状態であった場合にfalseを返す【NGKI0319】。

xsns\_xpnがfalse を返した場合、そのCPU例外ハンドラから、非タスクコンテキストから呼び出せるすべてのサービスコールを呼び出すことができ、(c)の方法によるリカバリ処理が可能である【NGKI0320】。

xsns\_dpnとxsns\_xpnのいずれのサービスコールもtrueを返した場合、そのCPU例外ハンドラからは、xsns\_dpnとxsns\_xpnに加えて、システムインタフェースレイヤのAPIとsns\_ker、ext\_kerのみを呼び出すことができ、その他のサービスコールを呼び出すことはできない【NGKI0321】。いずれのサービスコールもtrueを返したにもかかわらず、その他のサービスコールを呼び出した場合の動作は、保証されない【NGKI0322】。この場合には、(b)と(c)の方法によるリカバリ処理は行うことはできず、(a)または(d)の方法によるリカバリ処理を行うしかないことになる。

#### 【 $\mu$ ITRON4.0仕様との関係】

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

### 2.8.3 エミュレートされたCPU例外ハンドラ

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

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

#### 【 $\mu$ ITRON4.0仕様との関係】

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

### 2.8.4 カーネル管理外のCPU例外

カーネル非動作状態、カーネル内のクリティカルセクションの実行中、全割込みロック状態、CPUロック状態、カーネル管理外の割込みハンドラ実行中のいずれかで発生したCPU例外を、カーネル管理外のCPU例外と呼ぶ。また、それによ

2751 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。  
2752 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  
2753 ネル管理外のCPU例外とする。

2754

2755 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  
2756 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  
2757 理のCPU例外ハンドラと呼ぶ。

2758

2759 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  
2760 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  
2761 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】  
2762 が課される。

2763

2764 【補足説明】

2765

2766 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  
2767 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  
2768 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。

2769

## 2770 2.9 システムの初期化と終了

2771

### 2772 2.9.1 システム初期化手順

2773

2774 システムのリセット後、最初に実行するプログラムを、スタートアップモジュール  
2775 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  
2776 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  
2777 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  
2778 プモジュールが用意されている【NGKI0331】。

2779

2780 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  
2781 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  
2782 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  
2783 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  
2784 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  
2785 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  
2786 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  
2787 行うべき初期化処理を行うことが想定されている。

2788

2789 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  
2790 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、  
2791 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  
2792 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）  
2793 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  
2794 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  
2795 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  
2796 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  
2797 する必要がある【NGKI0335】。

2798

2799 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  
2800 ル内のデータ構造の初期化、カーネルが用いるデバイスの初期化など）と静的



APIの処理（オブジェクトの登録など）が行われる【NGKI0336】。静的APIのパラメータに関するエラーは、コンフィギュレータによって検出されるのが原則であるが、コンフィギュレータで検出できないエラーが、この処理中に検出される場合もある【NGKI0337】。

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

次に、静的API（ATT\_INI）により登録した初期化ルーチンが、システムコンフィギュレーションファイルにおける静的APIの記述順と同じ順序で実行される【NGKI0339】。

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

以上が終了すると、カーネル非動作状態から動作状態に遷移し（「2.5.1 カーネル動作状態と非動作状態」の節を参照）、カーネルの動作が開始される【NGKI0342】。具体的には、システム状態が、全割込みロック解除状態・CPUロック解除状態・割込み優先度マスク全解除状態・ディスパッチ許可状態に設定され（すなわち、割込みがマスク解除され）、タスクの実行が開始される。

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

#### 【μITRON4.0仕様との関係】

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

#### 2.9.2 システム終了手順

カーネルを終了させるサービスコール（ext\_ker）を呼び出すと、カーネル動作状態から非動作状態に遷移する（「2.5.1 カーネル動作状態と非動作状態」の節を参照）【NGKI0345】。具体的には、NMIを除くすべての割込みがマスクされ、タスクの実行が停止される。

2851  
2852 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  
2853 (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】。  
2854 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  
2855 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  
2856 対してカーネル終了処理の開始を要求する【NGKI0347】。複数のプロセッサか  
2857 ら、カーネルを終了させるサービスコール (ext\_ker) を呼び出してよい  
2858 【NGKI0348】。  
2859  
2860 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  
2861 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  
2862 【NGKI0349】。  
2863  
2864 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  
2865 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  
2866 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  
2867 チンは、初期割り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  
2868 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  
2869 終了処理ルーチンを実行する【NGKI0351】。  
2870  
2871 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  
2872 【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  
2873 り、アプリケーションで用意するのが基本であるが、カーネルの配布パッケー  
2874 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  
2875 【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  
2876 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】。  
2877  
2878 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  
2879 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  
2880 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  
2881 グを、図2-7に示す【NGKI0356】。  
2882  
2883 【使用上の注意】  
2884  
2885 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  
2886 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  
2887 態で割り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。  
2888  
2889 プロセッサが割り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  
2890 ケーションの責任である。例えば、ある時間を超えて割り込みをマスクしたまま  
2891 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  
2892 られる。割り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  
2893 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  
2894 ネルを終了させることができる。  
2895  
2896 【 $\mu$  ITRON4.0仕様との関係】  
2897  
2898  $\mu$  ITRON4.0仕様には、システム終了に関する規定はない。  
2899  
2900 2.10 オブジェクトの登録とその解除

2901  
2902 2.10.1 ID番号で識別するオブジェクト  
2903  
2904 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  
2905 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  
2906 る静的API (ATT\_YYY, ATA\_YYY) によってカーネルに登録する【NGKI0357】. オ  
2907 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  
2908 ないため、ID番号を指定して操作することができない【NGKI0358】.  
2909  
2910 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  
2911 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  
2912 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  
2913 済みの場合には、E\_OBJエラーとなる【NGKI0360】.  
2914  
2915 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  
2916 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  
2917 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  
2918 【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  
2919 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  
2920 には、E\_NOIDエラーとなる【NGKI0363】.  
2921  
2922 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  
2923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  
2924 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】.  
2925  
2926 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  
2927 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  
2928 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  
2929 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  
2930 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  
2931 つもりの操作が、新たに登録したオブジェクトに対して行われないように、注  
2932 意が必要である.  
2933  
2934 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、  
2935 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  
2936 となる【NGKI0369】.  
2937  
2938 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  
2939 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  
2940 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  
2941 単位の実行は継続される【NGKI0371】.  
2942  
2943 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  
2944 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  
2945 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  
2946 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  
2947 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  
2948 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  
2949 規定する【NGKI0374】.  
2950

オブジェクトを再初期化するサービスコール (ini\_yyy) は、指定したオブジェクトを削除した後に、同じパラメータで再度生成したのと等価の振舞いをする【NGKI0375】。ただし、オブジェクトを生成または追加する静的APIによって登録したオブジェクトも、再初期化することができる【NGKI0376】。

なお、動的生成対応カーネル以外では、オブジェクトを生成するサービスコール (acre\_yyy)、割付け可能なID番号の数を指定する静的API (AID\_YYY)、オブジェクトのアクセス許可ベクタを設定するサービスコール (sac\_yyy)、オブジェクトを削除するサービスコール (del\_yyy) は、サポートされない【NGKI0377】。

#### 【μ ITRON4.0仕様との関係】

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

μ ITRON4.0仕様では、割付け可能なID番号の数を指定する静的API (AID\_YYY) は規定されていない。

複数の待ち行列を持つ同期・通信オブジェクトを削除した時に、別の待ち行列で待っていたタスクの間の待ち解除の順序は、μ ITRON4.0仕様では実装依存とされている。

#### 【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。これにあわせて、アクセス許可ベクタを指定してオブジェクトを登録するサービスコール (cra\_yyy, acra\_yyy, ata\_yyy) も廃止した。

#### 【仕様決定の理由】

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

### 2.10.2 オブジェクト番号で識別するオブジェクト

オブジェクト番号で識別するオブジェクトは、オブジェクトを定義する静的API (DEF\_YYY) またはサービスコール (def\_yyy) によってカーネルに登録する【NGKI0378】。

オブジェクトを定義するサービスコール (def\_yyy) によって登録したオブジェクトは、同じサービスコールを、オブジェクトの定義情報を入れたパケットへのポインタをNULLとして呼び出すことによって、登録を解除することができる【NGKI0379】。登録解除したオブジェクト番号は、オブジェクト登録前の状態に戻され、同じオブジェクト番号に対して新たにオブジェクトを定義すること

3001 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  
3002 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】。

3003

3004 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  
3005 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  
3006 ラーとなる【NGKI0383】。

3007

3008 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  
3009 ル (def\_yyy) はサポートされない【NGKI0384】。

3010

3011 【μ ITRON4.0仕様との関係】

3012

3013 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  
3014 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  
3015 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  
3016 いこととした。

3017

3018 2.10.3 識別番号を持たないオブジェクト

3019

3020 識別する必要があるために、識別番号を持たないオブジェクトは、オブジェク  
3021 トを追加する静的API (ATT\_YYY) によってカーネルに登録する。

3022

3023 2.10.4 オブジェクト生成に必要なメモリ領域

3024

3025 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  
3026 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  
3027 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

3028 【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  
3029 フィギュレータまたはカーネルにより確保される【NGKI0386】。

3030

3031 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、  
3032 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  
3033 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。

3034

- 3035 ・データキュー管理領域
- 3036 ・優先度データキュー管理領域
- 3037 ・優先度別のメッセージキューヘッダ領域
- 3038 ・固定長メモリプール管理領域

3039

3040 【補足説明】

3041

3042 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  
3043 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  
3044 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  
3045 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。

3046

3047 2.10.5 オブジェクトが属する保護ドメインの設定

3048

3049 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  
3050 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

3051       【NGKI0387】.

3052

3053       カーネルオブジェクトを静的APIによって登録する場合には、オブジェクトを登  
3054       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  
3055       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  
3056       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）

3057       【NGKI0389】.

3058

3059       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  
3060       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  
3061       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  
3062       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  
3063       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  
3064       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  
3065       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  
3066       録する場合には、domidにTDOM\_NONE(=-2)を指定する.

3067

3068       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  
3069       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  
3070       オブジェクト属性で設定する必要がないものもある【NGKI0391】.

3071

3072       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、  
3073       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  
3074       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  
3075       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、  
3076       E\_RSATRエラーとなる【NGKI0394】.

3077

3078       【補足説明】

3079

3080       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  
3081       用意していない.

3082

3083       【仕様決定の理由】

3084

3085       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  
3086       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  
3087       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  
3088       タを増やさない方が望ましいためである.

3089

3090       2.10.6 オブジェクトが属するクラスの設定

3091

3092       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  
3093       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  
3094       【NGKI0395】.

3095

3096       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  
3097       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

3098       【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  
3099       囲みの外に記述する（「2.12.4 クラスの指定」の節を参照）【NGKI0397】.

3100

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

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

#### 【補足説明】

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

#### 【仕様決定の理由】

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

### 2.10.7 オブジェクトの状態参照

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

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

### 2.11 オブジェクトのアクセス保護

この節では、カーネルオブジェクトのアクセス保護について述べる。この節の内容は、保護機能対応カーネルにのみ適用される。

#### 2.11.1 オブジェクトのアクセス保護とアクセス違反の通知

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

3151 保護される【NGKI0407】.

3152

3153 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、  
3154 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.

3155

3156 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  
3157 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  
3158 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  
3159 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  
3160 からE\_OACVエラーが返る【NGKI0409】.

3161

3162 メモリオブジェクトに対して、通常のメモリアクセスにより、許可されていな  
3163 い書込みアクセスまたは読出しアクセス（実行アクセスを含む）を行おうとし  
3164 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  
3165 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、  
3166 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、  
3167 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.  
3168 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.

3169

3170 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  
3171 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  
3172 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  
3173 【NGKI0412】. E\_MACVエラーが返るかメモリアクセス違反ハンドラされるかは、  
3174 ターゲット定義である【NGKI0413】.

3175

3176 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  
3177 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）  
3178 を参照する方法を、ターゲット定義で用意する【NGKI0414】.

3179

3180 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、  
3181 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  
3182 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  
3183 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  
3184 アクセスを行おうとした場合の動作は保証されない【NGKI0416】.

3185

3186 【未決定事項】

3187

3188 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  
3189 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.

3190

3191 【 $\mu$  ITRON4.0/PX仕様との関係】

3192

3193  $\mu$  ITRON4.0/PX仕様では、アクセス保護の実装定義の制限について規定している  
3194 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  
3195 ト定義の制限以外については規定していない.

3196

3197 【仕様決定の理由】

3198

3199 オブジェクトに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  
3200 ベクタによって保護しないのは、オブジェクトに登録する前には、アクセス許



3201 可ベクタが設定されていないためである。

3202

## 3203 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

3204

3205 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）  
3206 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  
3207 合がある【NGKI0417】。

3208

3209 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。

3210

3211 (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  
3212 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  
3213 リー（private read only）と呼ぶ。

3214

3215 (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  
3216 しアクセス（実行アクセスを含む）を許可する【NGKI0419】。これを、専  
3217 有リードライト（private read/write）と呼ぶ。

3218

3219 (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  
3220 許可する【NGKI0420】。これを、共有リードオンリー（shared read only）  
3221 と呼ぶ。

3222

3223 (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  
3224 スを含む）を許可する【NGKI0421】。これを、共有リードライト（shared  
3225 read/write）と呼ぶ。

3226

3227 (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  
3228 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  
3229 アクセス（実行アクセスを含む）のみを許可する【NGKI0422】。これを、  
3230 共有リード専有ライト（shared read private write）と呼ぶ。

3231

3232 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  
3233 の数が制限される場合がある【NGKI0423】。

3234

## 3235 2.11.3 デフォルトのアクセス許可ベクタ

3236

3237 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  
3238 ス許可ベクタが設定される。

3239

3240 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  
3241 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  
3242 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  
3243 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  
3244 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  
3245 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  
3246 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  
3247 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  
3248 【NGKI3427】。

3249

3250 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、

3251 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  
3252 可パターンがいずれも、TACP\_SHAREDに設定される。

3253

3254 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  
3255 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  
3256 ターンがいずれも、TACP\_KERNELに設定される。

3257

#### 3258 2.11.4 アクセス許可ベクタの設定

3259

3260 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  
3261 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  
3262 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  
3263 【NGKI0427】。

3264

3265 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  
3266 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  
3267 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】。  
3268 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)  
3269 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  
3270 を設定しようとした場合には、E\_OBJエラーとなる【NGKI0430】。

3271

3272 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  
3273 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  
3274 的API (ATA\_YYY) が用意されている【NGKI0431】。

3275

3276 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  
3277 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】。  
3278 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  
3279 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  
3280 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  
3281 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  
3282 ベクタを変更しても、呼び出した時点ですでにミューテックスをロックしていた  
3283 タスクには影響しない。

3284

3285 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  
3286 する機能は用意していない。

3287

#### 3288 【μITRON4.0/PX仕様との関係】

3289

3290 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  
3291 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  
3292 API (SAC\_YYY) をサポートすることとした。

3293

3294 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  
3295 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。

3296

3297 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  
3298 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。  
3299 これは、【NGKI0403】の原則に合わせるための修正である。

3300

## 3301 2.11.5 カーネルの管理領域のアクセス保護

3302

3303 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  
3304 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  
3305 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。  
3306 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  
3307 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  
3308 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  
3309 【NGKI0433】。

3310

3311 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。  
3312 「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  
3313 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  
3314 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  
3315 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  
3316 ブジェクトの中に確保される【NGKI0435】。

3317

3318 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  
3319 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  
3320 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、  
3321 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  
3322 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  
3323 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  
3324 理領域とは異なる扱いとなる。

3325

## 3326 2.11.6 ユーザタスクのユーザスタック領域

3327

3328 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  
3329 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  
3330 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  
3331 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  
3332 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。

3333

3334 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  
3335 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  
3336 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  
3337 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  
3338 行えるかどうかは、ターゲット定義である【NGKI0440】。

3339

3340 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、  
3341 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  
3342 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】。

3343

3344 【μ ITRON4.0/PX仕様との関係】

3345

3346 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  
3347 きるものとした。

3348

## 3349 2.12 システムコンフィギュレーション手順

3350

## 3351 2.12.1 システムコンフィギュレーションファイル

3352

3353 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  
3354 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  
3355 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  
3356 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  
3357 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。

3358

3359 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  
3360 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  
3361 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  
3362 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  
3363 ることができる【NGKI0442】。

3364

3365 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  
3366 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、  
3367 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  
3368 レクトリの違いのみ）【NGKI0443】。

3369

3370 INCLUDE("ファイル名");

3371 INCLUDE(<ファイル名>);

3372

3373 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  
3374 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  
3375 【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  
3376 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  
3377 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  
3378 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。

3379

3380 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  
3381 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  
3382 【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  
3383 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  
3384 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  
3385 【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  
3386 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  
3387 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  
3388 【NGKI0448】。

3389

3390 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  
3391 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  
3392 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  
3393 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  
3394 ラーとなる。

3395

3396 【μ ITRON4.0仕様との関係】

3397

3398 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  
3399 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  
3400 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

3401 #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  
3402 換わる。逆に、 $\mu$ ITRON4.0仕様におけるINCLUDE静的APIの役割は、この仕様で  
3403 は#includeディレクティブに置き換わる。  
3404

## 3405 2.12.2 静的APIの文法とパラメータ

3406  
3407 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  
3408 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  
3409 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  
3410 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  
3411 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“{”  
3412 と”}”で囲んだ形で記述する【NGKI0451】。

3413  
3414 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  
3415 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】。

3416  
3417 静的APIのパラメータは、次の4種類に分類される。

### 3418 (a) オブジェクト識別名

3419  
3420 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  
3421 の識別名のみを記述することができる。

3422  
3423  
3424 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  
3425 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、  
3426 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  
3427 ディレクティブ（#define）を生成する【NGKI0453】。

3428  
3429 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  
3430 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  
3431 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  
3432 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  
3433ギュレータがエラーを報告する【NGKI0455】。

3434  
3435 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  
3436 はできない【NGKI0456】。

### 3437 (b) 整数定数式パラメータ

3438  
3439 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、  
3440 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  
3441 決まる整数定数式を記述することができる。

3442  
3443 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  
3444 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  
3445 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】。

### 3446 (c) 一般定数式パラメータ

3447  
3448 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  
3449  
3450

3451 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。  
3452  
3453 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  
3454 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  
3455 ドするファイルに含まれていなければならない【NGKI0458】。  
3456  
3457 (d) 文字列パラメータ  
3458  
3459 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。  
3460 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。  
3461  
3462 【 $\mu$  ITRON4.0仕様との関係】  
3463  
3464  $\mu$  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  
3465 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。  
3466  
3467 (A) 自動割付け対応整数値パラメータ  
3468 (B) 自動割付け非対応整数値パラメータ  
3469 (C) プリプロセッサ定数式パラメータ  
3470 (D) 一般定数式パラメータ  
3471  
3472 この仕様の(a)が、おおよそ $\mu$  ITRON4.0仕様の(A)に相当するが、(a)には整数値  
3473 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  
3474 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。  
3475  
3476  $\mu$  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  
3477 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  
3478 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  
3479 るパラメータを(C)としていた。  
3480  
3481 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  
3482 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  
3483 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  
3484 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  
3485 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。  
3486  
3487 2.12.3 保護ドメインの指定  
3488  
3489 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  
3490 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  
3491 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  
3492 【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、  
3493 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  
3494 ラーを報告する【NGKI0461】。  
3495  
3496 ユーザドメインの囲みの文法は次の通り【NGKI0462】。  
3497  
3498 DOMAIN(保護ドメイン名) {  
3499 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  
3500 }

3501  
3502 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  
3503 ることができる【NGKI0463】。  
3504  
3505 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  
3506 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  
3507 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  
3508 サのディレクティブ(#define)を生成する【NGKI0464】。また、ユーザドメイ  
3509 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  
3510 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  
3511 【NGKI0465】。  
3512  
3513 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  
3514 ンIDの割付けのみを行うことができる【NGKI0466】。  
3515  
3516 カーネルドメインの囲みの文法は次の通り【NGKI0467】。  
3517  
3518       KERNEL\_DOMAIN {  
3519           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  
3520       }  
3521  
3522 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  
3523 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  
3524 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  
3525 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  
3526 【NGKI0469】。  
3527  
3528       【μITRON4.0/PX仕様との関係】  
3529  
3530 保護ドメインの囲みの文法を変更した。  
3531  
3532       【仕様決定の理由】  
3533  
3534 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  
3535 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  
3536 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  
3537 クト属性によって指定するためである。  
3538  
3539 2.12.4 クラスの指定  
3540  
3541 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  
3542 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  
3543 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  
3544 合には、コンフィギュレータがE\_RSATRエラーを報告する【NGKI0471】。  
3545  
3546 クラスの囲みの文法は次の通り【NGKI0472】。  
3547  
3548       CLASS(クラスID) {  
3549           クラスに属するオブジェクトを登録する静的API等  
3550       }

3551  
3552 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  
3553 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  
3554 ギュレータがE\_IDエラーを報告する【NGKI0474】。  
3555  
3556 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】。  
3557 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  
3558 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  
3559 を報告する【NGKI0476】。  
3560  
3561 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  
3562 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】。  
3563  
3564 【仕様決定の理由】  
3565  
3566 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  
3567 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  
3568 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  
3569 定するためである。  
3570  
3571 2.12.5 コンフィギュレータの処理モデル  
3572  
3573 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  
3574 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  
3575 成する（図2-8）。  
3576  
3577 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  
3578 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  
3579 めに、パラメータ計算用C言語ファイル（cfig1\_out.c）を生成する。この時、シ  
3580 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  
3581 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  
3582 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  
3583 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  
3584 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。  
3585  
3586 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  
3587 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  
3588 マットの形に変換したSレコードファイル（cfig1\_out.srec）と、その中の各シン  
3589 ボールとアドレスの対応表を含むシンボルファイル（cfig1\_out.syms）を生成す  
3590 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、  
3591 この段階でエラーが検出される。  
3592  
3593 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  
3594 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  
3595 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  
3596 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  
3597 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）  
3598 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  
3599 （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  
3600 数に一致）やオブジェクトのID番号などの定義を出力する。静的APIの整数定数



式パラメータに不正がある場合には、この段階でエラーが検出される。

パス2で生成されたファイルを、他のソースファイルとあわせてコンパイルし、アプリケーションのロードモジュールを生成する。また、そのSレコードファイル (system.srec) とシンボルフайル (system.syms) を生成する。静的APIの一般定数式パラメータに解釈できない式が記述された場合には、この段階でエラーが検出される。

コンフィギュレータのパス3では、パス1およびパス2で生成されたロードモジュールのSレコードファイルとシンボルフайルから、静的APIのパラメータの値などを取り出し、妥当性のチェックを行う。静的APIの一般定数式パラメータに不正がある場合には、この段階でエラーが検出される。

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

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

そのため、パス2で生成されたロードモジュールは、仮のロードモジュールという位置付けになる。ここで、パス3以降に必要な情報を取り出し、最終的なロードモジュールのサイズを割り出せるように、パス3以降でメモリ構成・初期化ファイルに生成するのと同様のデータ構造を、パス2において仮のメモリ構成・初期化ファイル (kernel\_mem2.c) に生成する。また、これをリンクするための仮のリンクスクリプト (cfg2\_out.ld) を生成し、これらを用いて仮のロードモジュールを生成する。さらに、仮のロードモジュールのSレコードファイル (cfg2\_out.srec) とシンボルフайル (cfg2\_out.syms) も、最終的なものと混同しないように、異なるファイル名で生成する。

パス3は、ターゲット依存で用いるパスで、メモリ配置やメモリ保護のための設定情報のサイズを最適化するための処理を行う。パス2で生成された仮のロードモジュールのSレコードファイルとシンボルフайルから必要な情報を取り出し、再度、仮のメモリ構成・初期化ファイル (kernel\_mem3.c) と仮のリンクスクリプト (cfg3\_out.ld) を生成する。また、これらのファイルを他のソースファイルとあわせてコンパイルして仮のロードモジュールを生成し、そのSレコードファイル (cfg3\_out.srec) とシンボルフайル (cfg3\_out.syms) を生成する。この段階で、メモリオブジェクトに重なりがあるなどのエラーが検出される場合もある。

パス4では、パス3 (パス3を用いない場合はパス2) で生成された仮のロードモジュールのSレコードファイルとシンボルフайルから必要な情報を取り出し、最終的なメモリ構成・初期化ファイル (kernel\_mem.c) とリンクスクリプト (ldscript.ld) を生成する。またパス4では、保護機能対応でないカーネルにおいてパス3で行っていた静的APIパラメータの値などの妥当性のチェックも行う。そのため、静的APIの一般定数式パラメータに不正がある場合には、この段階でエラーが検出される。

3651  
3652 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，  
3653 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  
3654 ドファイル（system.srec，必要な場合のみ）とシンボルファイル  
3655 （system.syms）を生成する．  
3656  
3657 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）  
3658 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  
3659 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  
3660 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  
3661 合を示すものである．  
3662  
3663 【 $\mu$  ITRON4.0仕様との関係】  
3664  
3665 コンフィギュレータの処理モデルは全面的に変更した．  
3666  
3667 2.12.6 静的APIのパラメータに関するエラー検出  
3668  
3669 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  
3670 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  
3671 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．  
3672 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」  
3673 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．  
3674  
3675 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  
3676 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  
3677 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．  
3678  
3679 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  
3680 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，  
3681 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  
3682 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  
3683 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  
3684 とせず，検出できた場合には警告として報告する【NGKI0480】．  
3685  
3686 【 $\mu$  ITRON4.0仕様との関係】  
3687  
3688  $\mu$  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  
3689 れていない．  
3690  
3691 2.12.7 オブジェクトのID番号の指定  
3692  
3693 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  
3694 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．  
3695  
3696 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  
3697 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  
3698 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  
3699 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  
3700 を報告する【NGKI0482】．

3701  
3702 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  
3703 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  
3704 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】。

3705  
3706 【 $\mu$  ITRON4.0仕様との関係】

3707  
3708  $\mu$  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  
3709 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。

3710  
3711 2.13 TOPPERSネーミングコンベンション

3712  
3713 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  
3714 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  
3715 ル間のインタフェースに関わる名称に適用することを想定しているが、モジュー  
3716 ル内部の名称に適用してもよい。

3717  
3718 2.13.1 モジュール識別名

3719  
3720 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  
3721 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  
3722 ル識別名は、英文字と数字で構成し、2～8文字程度の長さとする。

3723  
3724 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  
3725 ジュール識別名は“sil”とする。

3726  
3727 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  
3728 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  
3729 識別名を含めない名称を使用する。

3730  
3731 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  
3732 したものをWWWと表記する。

3733  
3734 2.13.2 データ型名

3735  
3736 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  
3737 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  
3738 “\_t”である名称とする。

3739  
3740 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，  
3741 “\_”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  
3742 が“\_T”である名称とする場合もある。

3743  
3744 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。

3745  
3746 (A) パケットのデータ型

3747  
3748 T\_CYYY            acre\_yyyに渡すパケットのデータ型  
3749 T\_DYYY            def\_yyyに渡すパケットのデータ型  
3750 T\_RYYY            ref\_yyyに渡すパケットのデータ型

3751 T\_WWW\_CYYY www\_acre\_yyyに渡すパケットのデータ型  
 3752 T\_WWW\_DYYY www\_def\_yyyに渡すパケットのデータ型  
 3753 T\_WWW\_RYYY www\_ref\_yyyに渡すパケットのデータ型  
 3754

### 2.13.3 関数名

関数の名称は、英小文字、数字、“\_”で構成する。

関数の種類毎に、次のネーミングコンベンションを定める。

#### (A) サービスコール

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

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

#### 【補足説明】

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

#### (B) コールバック

コールバックの名称は、サービスコールのネーミングコンベンションに従う。

### 2.13.4 変数名

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

変数の名称に関して、次のガイドラインを設ける。

3789 ~id ~ID（オブジェクトのID番号、ID型）  
 3790 ~no ~番号（オブジェクト番号）  
 3791 ~atr ~属性（オブジェクト属性、ATR型）  
 3792 ~stat ~状態（オブジェクト状態、STAT型）  
 3793 ~mode ~モード（サービスコールの動作モード、MODE型）  
 3794 ~pri ~優先度（優先度、PRI型）  
 3795 ~sz ~サイズ（単位はバイト数、SIZE型またはuint\_t型）  
 3796 ~cnt ~の個数（単位は個数、uint\_t型）  
 3797 ~ptn ~パターン  
 3798 ~tim ~時刻、～時間  
 3799 ~cd ~コード  
 3800 i~ ~の初期値

3801           max～           ～の最大値  
 3802           min～           ～の最小値  
 3803           left～          ～の残り

3804

3805           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  
 3806           ンを設ける.

3807

3808           p\_～           ポインタ  
 3809           pp\_～          ポインタを入れる領域へのポインタ  
 3810           pk\_～          パケットへのポインタ  
 3811           ppk\_～         パケットへのポインタを入れる領域へのポインタ

3812

3813           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.

3814

3815           (A) パケットへのポインタ

3816

3817           pk\_cyyy         acre\_yyyに渡すパケットへのポインタ  
 3818           pk\_dyyy         def\_yyyに渡すパケットへのポインタ  
 3819           pk\_ryyy         ref\_yyyに渡すパケットへのポインタ  
 3820           pk\_www\_cyyy     www\_acre\_yyyに渡すパケットへのポインタ  
 3821           pk\_www\_dyyy     www\_def\_yyyに渡すパケットへのポインタ  
 3822           pk\_www\_ryyy     www\_ref\_yyyに渡すパケットへのポインタ

3823

3824           2.13.5 定数名

3825

3826           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  
 3827           字、“\_”で構成する.

3828

3829           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.

3830

3831           (A) メインエラーコード

3832

3833           メインエラーコードは、先頭が“E\_”である名称とする.

3834

3835           (B) 機能コード

3836

3837           TFN\_XXX\_YYY     xxx\_yyyの機能コード  
 3838           TFN\_WWW\_XXX\_YYY   www\_xxx\_yyyの機能コード

3839

3840           (C) その他の定数

3841

3842           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、  
 3843           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  
 3844           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.

3845

3846           また、定数の名称に関して、次のガイドラインを設ける.

3847

3848           TA\_～          オブジェクトの属性値  
 3849           TSZ\_～         ～のサイズ  
 3850           TBIT\_～        ～のビット数

3851 TMAX\_〜 〜の最大値

3852 TMIN\_〜 〜の最小値

3853

#### 3854 2.13.6 マクロ名

3855

3856 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  
3857 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  
3858 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  
3859 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  
3860 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  
3861 する場合もある。

3862

3863 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。

3864

##### 3865 (A) 構成マクロ

3866

3867 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  
3868 る。

3869

3870 TSZ\_〜 〜のサイズ

3871 TBIT\_〜 〜のビット数

3872 TMAX\_〜 〜の最大値

3873 TMIN\_〜 〜の最小値

3874

#### 3875 2.13.7 静的API名

3876

3877 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  
3878 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  
3879 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  
3880 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。

3881

#### 3882 2.13.8 ファイル名

3883

3884 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  
3885 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。  
3886 また、“-”も使用しない。

3887

3888 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。

3889

##### 3890 (A) ヘッダファイル

3891

3892 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  
3893 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。

3894

#### 3895 2.13.9 モジュール内部の名称の衝突回避

3896

3897 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  
3898 ために、次のガイドラインを設ける。

3899

3900 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

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

3904

3905 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  
 3906 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  
 3907 能性のある名称は、`"TOPPERS_"`で始まる名称とする。

3908

## 3909 2.14 TOPPERS共通定義

3910

3911 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。

3912

### 3913 2.14.1 TOPPERS共通ヘッダファイル

3914

3915 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  
 3916 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  
 3917 れている【NGKI0484】。TOPPERS共通定義を用いる場合には、TOPPERS共通ヘッ  
 3918 ダファイルをインクルードする【NGKI0485】。

3919

3920 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  
 3921 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、  
 3922 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  
 3923 直接インクルードする必要はない【NGKI0486】。

3924

### 3925 2.14.2 TOPPERS共通データ型

3926

3927 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  
 3928 タ型は次の通りである【NGKI0487】。

3929

3930	<code>int8_t</code>	符号付き8ビット整数（オプション，C99準拠）
3931	<code>uint8_t</code>	符号無し8ビット整数（オプション，C99準拠）
3932	<code>int16_t</code>	符号付き16ビット整数（C99準拠）
3933	<code>uint16_t</code>	符号無し16ビット整数（C99準拠）
3934	<code>int32_t</code>	符号付き32ビット整数（C99準拠）
3935	<code>uint32_t</code>	符号無し32ビット整数（C99準拠）
3936	<code>int64_t</code>	符号付き64ビット整数（オプション，C99準拠）
3937	<code>uint64_t</code>	符号無し64ビット整数（オプション，C99準拠）
3938	<code>int128_t</code>	符号付き128ビット整数（オプション，C99準拠）
3939	<code>uint128_t</code>	符号無し128ビット整数（オプション，C99準拠）

3940

3941	<code>int_least8_t</code>	8ビット以上の符号付き整数（C99準拠）
3942	<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）

3943

3944	<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）
3945	<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）

3946

3947	<code>bool_t</code>	真偽値（ <code>true</code> または <code>false</code> ）
3948	<code>int_t</code>	16ビット以上の符号付き整数
3949	<code>uint_t</code>	<code>int_t</code> 型と同じサイズの符号無し整数
3950	<code>long_t</code>	32ビット以上かつ <code>int_t</code> 型以上のサイズの符号付き整数

3951	ulong_t	long_t型と同じサイズの符号無し整数
3952		
3953	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)
3954	uintptr_t	intptr_t型と同じサイズの符号無し整数 (C99準拠)
3955		
3956	FN	機能コード (符号付き整数, int_tに定義)
3957	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)
3958		
3959	ID	オブジェクトのID番号 (符号付き整数, int_tに定義)
3960	ATR	オブジェクト属性 (符号無し整数, uint_tに定義)
3961	STAT	オブジェクトの状態 (符号無し整数, uint_tに定義)
3962	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)
3963	PRI	優先度 (符号付き整数, int_tに定義)
3964	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)
3965		
3966		
3967	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)
3968	RELTIM	相対時間 (符号無し整数, 単位はミリ秒, uint_tに定義)
3969	SYSTIM	システム時刻 (符号無し整数, 単位はミリ秒, ulong_tに定義)
3970	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)
3971		
3972		
3973	FP	プログラムの起動番地 (型の定まらない関数ポインタ)
3974		
3975	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)
3976	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)
3977		
3978	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)
3979		
3980		
3981		
3982	MB_T	オブジェクト管理領域を確保するためのデータ型
3983		
3984	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)
3985		
3986	ACVCT	アクセス許可ベクタ
3987		
3988	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。	
3989	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。	
3990		
3991	int8_t, uint8_t, int64_t, uint64_t, int128_t, uint128_t, float32_t,	
3992	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	
3993	れらが使用できるかどうかは、それぞれ、INT8_MAX, UINT8_MAX, INT64_MAX,	
3994	UINT64_MAX, INT128_MAX, UINT128_MAX, FLOAT32_MAX, DOUBLE64_MAXがマクロ	
3995	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	
3996	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、	
3997	float32_tとdouble64_tは使用できないものとする【NGKI0490】。	
3998		
3999	【μITRON4.0仕様との関係】	
4000		



4001 B, UB, H, UH, W, UW, D, UD, VP\_INTに代えて, C99準拠のint8\_t, uint8\_t,  
 4002 int16\_t, uint16\_t, int32\_t, uint32\_t, int64\_t, uint64\_t, intptr\_tを用い  
 4003 ることにした. また, uintptr\_t, int128\_t, uint128\_tを用意することにした.  
 4004  
 4005 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  
 4006 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  
 4007 は用意しないことにした.  
 4008  
 4009 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.  
 4010 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  
 4011 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  
 4012 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  
 4013 た. int\_least16\_t, uint\_least16\_t, int\_least32\_t, uint\_least32\_tを導入  
 4014 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,  
 4015 それぞれint16\_t, uint16\_t, int32\_t, uint32\_tで代用できるためである.  
 4016  
 4017 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  
 4018 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  
 4019 の64ビットを表す型としてdouble64\_tを導入した.  
 4020  
 4021 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  
 4022 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  
 4023  
 4024 2.14.3 TOPPERS共通定数  
 4025  
 4026 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  
 4027 次の通りである (一部, C90に規定されているものも含む) .  
 4028  
 4029 (1) 一般定数【NGKI0491】  
 4030  
 4031 NULL 無効ポインタ  
 4032  
 4033 true 1 真  
 4034 false 0 偽  
 4035  
 4036 E\_OK 0 正常終了  
 4037  
 4038 【μITRON4.0仕様との関係】  
 4039  
 4040 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  
 4041 を用いることにした.  
 4042  
 4043 (2) 整数型に格納できる最大値と最小値【NGKI0492】  
 4044  
 4045 INT8\_MAX int8\_tに格納できる最大値 (オプション, C99準拠)  
 4046 INT8\_MIN int8\_tに格納できる最小値 (オプション, C99準拠)  
 4047 UINT8\_MAX uint8\_tに格納できる最大値 (オプション, C99準拠)  
 4048 INT16\_MAX int16\_tに格納できる最大値 (C99準拠)  
 4049 INT16\_MIN int16\_tに格納できる最小値 (C99準拠)  
 4050 UINT16\_MAX uint16\_tに格納できる最大値 (C99準拠)

4051	INT32_MAX	int32_tに格納できる最大値 (C99準拠)
4052	INT32_MIN	int32_tに格納できる最小値 (C99準拠)
4053	UINT32_MAX	uint32_tに格納できる最大値 (C99準拠)
4054	INT64_MAX	int64_tに格納できる最大値 (オプション, C99準拠)
4055	INT64_MIN	int64_tに格納できる最小値 (オプション, C99準拠)
4056	UINT64_MAX	uint64_tに格納できる最大値 (オプション, C99準拠)
4057	INT128_MAX	int128_tに格納できる最大値 (オプション, C99準拠)
4058	INT128_MIN	int128_tに格納できる最小値 (オプション, C99準拠)
4059	UINT128_MAX	uint128_tに格納できる最大値 (オプション, C99準拠)
4060		
4061	INT_LEAST8_MAX	int_least8_tに格納できる最大値 (C99準拠)
4062	INT_LEAST8_MIN	int_least8_tに格納できる最小値 (C99準拠)
4063	UINT_LEAST8_MAX	uint_least8_tに格納できる最大値 (C99準拠)
4064	INT_MAX	int_tに格納できる最大値 (C90準拠)
4065	INT_MIN	int_tに格納できる最小値 (C90準拠)
4066	UINT_MAX	uint_tに格納できる最大値 (C90準拠)
4067	LONG_MAX	long_tに格納できる最大値 (C90準拠)
4068	LONG_MIN	long_tに格納できる最小値 (C90準拠)
4069	ULONG_MAX	ulong_tに格納できる最大値 (C90準拠)
4070		
4071	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)
4072		
4073	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)
4074		
4075	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)
4076		
4077	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)
4078		
4079		
4080	(3) 整数型のビット数 【NGKI0493】	
4081		
4082	CHAR_BIT	char型のビット数 (C90準拠)
4083		
4084	(4) オブジェクト属性 【NGKI0494】	
4085		
4086	TA_NULL	OU オブジェクト属性を指定しない
4087		
4088	(5) タイムアウト指定 【NGKI0495】	
4089		
4090	TMO_POL	0 ポーリング
4091	TMO_FEVR	-1 永久待ち
4092	TMO_NBLK	-2 ノンブロッキング
4093		
4094	(6) アクセス許可パターン 【NGKI0496】	
4095		
4096	TACP_KERNEL	OU カーネルドメインのみにアクセスを許可
4097	TACP_SHARED	~OU すべての保護ドメインにアクセスを許可
4098		
4099	2.14.4 TOPPERS共通エラーコード	
4100		

4101 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  
 4102 【NGKI0497】.

4103

4104 (A) 内部エラークラス (EC\_SYS, -5~-8)

4105

4106 E\_SYS -5 システムエラー

4107

4108 (B) 未サポートエラークラス (EC\_NOSPT, -9~-16)

4109

4110 E\_NOSPT -9 未サポート機能

4111 E\_RSFN -10 予約機能コード

4112 E\_RSATR -11 予約属性

4113

4114 (C) パラメータエラークラス (EC\_PAR, -17~-24)

4115

4116 E\_PAR -17 パラメータエラー

4117 E\_ID -18 不正ID番号

4118

4119 (D) 呼出しコンテキストエラークラス (EC\_CTX, -25~-32)

4120

4121 E\_CTX -25 コンテキストエラー

4122 E\_MACV -26 メモリアクセス違反

4123 E\_OACV -27 オブジェクトアクセス違反

4124 E\_ILUSE -28 サービスコール不正使用

4125

4126 (E) 資源不足エラークラス (EC\_NOMEM, -33~-40)

4127

4128 E\_NOMEM -33 メモリ不足

4129 E\_NOID -34 ID番号不足

4130 E\_NORES -35 資源不足

4131

4132 (F) オブジェクト状態エラークラス (EC\_OBJ, -41~-48)

4133

4134 E\_OBJ -41 オブジェクト状態エラー

4135 E\_NOEXS -42 オブジェクト未登録

4136 E\_QOVR -43 キューイングオーバーフロー

4137

4138 (G) 待ち解除エラークラス (EC\_RLWAI, -49~-56)

4139

4140 E\_RLWAI -49 待ち禁止状態または待ち状態の強制解除

4141 E\_TMOUT -50 ポーリング失敗またはタイムアウト

4142 E\_DLT -51 待ちオブジェクトの削除または再初期化

4143 E\_CLS -52 待ちオブジェクトの状態変化

4144

4145 (H) 警告クラス (EC\_WARN, -57~-64)

4146

4147 E\_WBLK -57 ノンブロッキング受付け

4148 E\_BOVR -58 バッファオーバーフロー

4149

4150 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、

4151 [NGKI0019] の原則では例外としている.

4152

4153 【 $\mu$  ITRON4.0仕様との関係】

4154

4155 E\_NORESは,  $\mu$  ITRON4.0仕様に規定されていないエラーコードである.

4156

## 4157 2.14.5 TOPPERS共通マクロ

4158

### 4159 (1) 整数定数を作るマクロ【NGKI0498】

4160

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

4171

4172	UINT_C(val)	uint_t型の定数を作るマクロ
------	-------------	------------------

4173	ULONG_C(val)	ulong_t型の定数を作るマクロ
------	--------------	-------------------

4174

### 4175 【仕様決定の理由】

4176

4177 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  
4178 も参照する定数を記述するためである. C言語のみで用いる定数をこれらのマク  
4179 ロを使って記述する必要はない.

4180

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

4182

4183	offsetof(structure, field)	構造体structure中のフィールドfieldの 4184 バイト位置を返すマクロ (C90準拠)
------	----------------------------	-------------------------------------------------------

4185

4186	alignof(type)	型typeのアラインメント単位を返すマクロ
------	---------------	-----------------------

4187

4188	ALIGN_TYPE(addr, type)	番地addrが型typeに対してアラインしてい 4189 るかどうかを返すマクロ
------	------------------------	---------------------------------------------

4190

### 4191 (3) assertマクロ【NGKI0500】

4192

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

4194

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

4196

4197	inline	インライン関数
------	--------	---------

4198	Inline	ファイルローカルなインライン関数
------	--------	------------------

4199	asm	インラインアセンブラ
------	-----	------------

4200	Asm	インラインアセンブラ (最適化抑止)
------	-----	--------------------

4201           throw()                   例外を発生しない関数  
 4202           NoReturn                  リターンしない関数  
 4203  
 4204   (5) エラーコード構成・分解マクロ【NGKI0502】  
 4205  
 4206           ERCD(mercd, sercd)       メインエラーコードmercdとサブエラーコードsercdか  
 4207                                      ら、エラーコードを構成するためのマクロ  
 4208  
 4209           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  
 4210                                      ためのマクロ  
 4211           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  
 4212                                      めのマクロ  
 4213  
 4214   (6) アクセス許可パターン構成マクロ【NGKI0503】  
 4215  
 4216           TACP(domid)               domidで指定されるユーザドメインのみにアクセスを  
 4217                                      許可するアクセス許可パターンを構成するためのマ  
 4218                                      クロ  
 4219  
 4220   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  
 4221   することができる【NGKI0504】. TDOM\_SELF, TDOM\_KERNEL, TDOM\_NONEを指定し  
 4222   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  
 4223   【NGKI0505】.  
 4224  
 4225   2.14.6 TOPPERS共通構成マクロ  
 4226  
 4227   (1) 相対時間の範囲【NGKI0506】  
 4228  
 4229           TMAX\_RELTIM               相対時間に指定できる最大値  
 4230  
 4231   2.15 カーネル共通定義  
 4232  
 4233   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.  
 4234  
 4235   2.15.1 カーネルヘッダファイル  
 4236  
 4237   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)  
 4238   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  
 4239   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  
 4240   【NGKI0508】.  
 4241  
 4242   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  
 4243   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)  
 4244   に含まれる【NGKI0509】. 具体的には、登録できるオブジェクトの数  
 4245   (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  
 4246   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  
 4247   ドする【NGKI0510】.  
 4248  
 4249   μ ITRON4.0仕様で規定されており、この仕様で廃止されたデータ型および定数  
 4250   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

4251 する【NGKI0511】.

4252

4253 【 $\mu$  ITRON4.0仕様との関係】

4254

4255 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  
4256 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  
4257 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、 $\mu$  ITRON4.0仕  
4258 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  
4259 化ヘッダファイル (kernel\_cfg.h) に変更した。

4260

## 4261 2.15.2 カーネル共通定数

4262

4263 (1) オブジェクト属性【NGKI0512】

4264

4265 TA\_TPRI 0x01U タスクの待ち行列をタスクの優先度順に

4266

4267 【 $\mu$  ITRON4.0仕様との関係】

4268

4269 値が0のオブジェクト属性 (TA\_HLNG, TA\_TFIFO, TA\_MFIFO, TA\_WSGL) は、デフォ  
4270 ルトの扱いにして廃止した。これは、「(tskatr & TA\_HLNG) != 0U」のような  
4271 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。  
4272 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  
4273 した。

4274

4275 (2) 保護ドメインID【NGKI0513】

4276

4277	TDOM_SELF	0	自タスクの属する保護ドメイン
4278	TDOM_KERNEL	-1	カーネルドメイン
4279	TDOM_NONE	-2	無所属 (保護ドメインに属さない)

4280

4281 (3) その他のカーネル共通定数【NGKI0514】

4282

4283	TCLS_SELF	0	自タスクの属するクラス
4284			
4285	TPRC_NONE	0	割付けプロセッサの指定がない
4286	TPRC_INI	0	初期割付けプロセッサ
4287			
4288	TSK_SELF	0	自タスク指定
4289	TSK_NONE	0	該当するタスクがない
4290			
4291	TPRI_SELF	0	自タスクのベース優先度の指定
4292	TPRI_INI	0	タスクの起動時優先度の指定
4293			
4294	TIPM_ENAALL	0	割込み優先度マスク全解除

4295

4296 (4) カーネルで用いるメインエラーコード

4297

4298 「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  
4299 で、E\_CLS, E\_WBLK, E\_BOVRの3つは、カーネルでは使用しない【NGKI0515】.

4300

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、サービスコールから、E\_RSFN, E\_RSATR, E\_MACV, E\_OACV, E\_NOMEM, E\_NOID, E\_NORES, E\_NOEXSが返る状況は起こらない【ASPS0011】。E\_RSATRは、コンフィギュレータによって検出される【ASPS0012】。ただし、動的生成機能拡張パッケージでは、サービスコールから、E\_RSATR, E\_NOMEM, E\_NOID, E\_NOEXSが返る状況が起こる【ASPS0013】。

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、サービスコールから、E\_RSFN, E\_RSATR, E\_MACV, E\_OACV, E\_NOMEM, E\_NOID, E\_NORES, E\_NOEXSが返る状況は起こらない【FMPS0007】。E\_RSATRとE\_NORESは、コンフィギュレータによって検出される【FMPS0008】。

【TOPPERS/HRP2カーネルにおける規定】

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

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、サービスコールから、E\_RSFN, E\_RSATR, E\_MACV, E\_OACV, E\_ILUSE, E\_NOMEM, E\_NOID, E\_NORES, E\_NOEXS, E\_RLWAI, E\_TMOUT, E\_DLTが返る状況は起こらない【SSPS0008】。E\_RSATRは、コンフィギュレータによって検出される【SSPS0009】。

### 2.15.3 カーネル共通マクロ

(1) スタック領域をアプリケーションで確保するためのデータ型とマクロ

スタック領域をアプリケーションで確保するために、次のデータ型とマクロを用意している【NGKI0516】。

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

これらを用いてスタック領域を確保する方法は次の通り【NGKI0517】。

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

この方法で確保したスタック領域を、サービスコールまたは静的APIに渡す場合には、スタック領域の先頭番地に<スタック領域の変数名>を、スタック領域の

4351 サイズにROUND\_STK\_T(<スタック領域のサイズ>)を指定する【NGKI0518】.

4352

4353 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  
4354 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  
4355 管理機能」の節のCRE\_TSKの機能の項を参照すること.

4356

4357 (2) オブジェクト属性を作るマクロ

4358

4359 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  
4360 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

4361 【NGKI0520】.

4362

4363 TA\_DOM(domid) domidで指定される保護ドメインに属する

4364

4365 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  
4366 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

4367 【NGKI0521】.

4368

4369 TA\_CLS(clsid) clsidで指定されるクラスに属する

4370

4371 (3) サービスコールの呼出し方法を指定するマクロ

4372

4373 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  
4374 クロとして、次のマクロを用意している【NGKI0522】.

4375

4376 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  
4377 て呼び出すための名称

4378

4379 2.15.4 カーネル共通構成マクロ

4380

4381 (1) サポートする機能【NGKI0523】

4382

4383	TOPPERS_SUPPORT_PROTECT	保護機能対応のカーネル
4384	TOPPERS_SUPPORT_MULTI_PRC	マルチプロセッサ対応のカーネル
4385	TOPPERS_SUPPORT_DYNAMIC_CRE	動的生成対応のカーネル

4386

4387 【未決定事項】

4388

4389 マクロ名は、今後変更する可能性がある.

4390

4391 (2) 優先度の範囲【NGKI0524】

4392

4393	TMIN_TPRI	タスク優先度の最小値 (=1)
4394	TMAX_TPRI	タスク優先度の最大値

4395

4396 【TOPPERS/ASPカーネルにおける規定】

4397

4398 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

4399 【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  
4400 256に拡張することができる【ASPS0015】.



4401  
4402     **【TOPPERS/FMPカーネルにおける規定】**  
4403  
4404     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  
4405     **【FMPS0009】**。  
4406  
4407     **【TOPPERS/HRP2カーネルにおける規定】**  
4408  
4409     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  
4410     **【HRPS0008】**。  
4411  
4412     **【TOPPERS/SSPカーネルにおける規定】**  
4413  
4414     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  
4415     **【SSPS0010】**。  
4416  
4417     **【 $\mu$  ITRON4.0仕様との関係】**  
4418  
4419     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  
4420     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。  
4421  
4422     (3) プロセッサの数  
4423  
4424     マルチプロセッサ対応カーネルでは、プロセッサの数を知らるためのマクロとし  
4425     て、次の構成マクロを用意している **【NGKI0525】**。  
4426  
4427         TNUM\_PRCID         プロセッサの数  
4428  
4429     (4) 特殊な役割を持ったプロセッサ  
4430  
4431     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らるた  
4432     めのマクロとして、次の構成マクロを用意している **【NGKI0526】**。  
4433  
4434         TOPPERS\_MASTER\_PRCID         マスタプロセッサのID番号  
4435         TOPPERS\_SYSTIM\_PRCID         システム時刻管理プロセッサのID番号（グ  
4436         ローバルタイマ方式の場合のみ）  
4437  
4438     (5) タイマ方式  
4439  
4440     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らるためのマクロ  
4441     として、次の構成マクロを用意している **【NGKI0527】**。  
4442  
4443         TOPPERS\_SYSTIM\_LOCAL         ローカルタイマ方式の場合にマクロ定義  
4444         TOPPERS\_SYSTIM\_GLOBAL         グローバルタイマ方式の場合にマクロ定義  
4445  
4446     (6) バージョン情報 **【NGKI0528】**  
4447  
4448         TKERNEL\_MAKER         カーネルのメーカコード（=0x0118）  
4449         TKERNEL\_PRID         カーネルの識別番号  
4450         TKERNEL\_SPVER         カーネル仕様のバージョン番号

4451           TKERNEL\_PRVER           カーネルのバージョン番号

4452

4453           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  
4454           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。

4455

4456           カーネルの識別番号 (TKERNEL\_PRID) は、TOPPERSカーネルの種類を表す。

4457

4458	0x0001	TOPPERS/JSPカーネル
4459	0x0002	予約 (IIMPカーネル)
4460	0x0003	予約 (IDLカーネル)
4461	0x0004	TOPPERS/FI4カーネル
4462	0x0005	TOPPERS/FDMPカーネル
4463	0x0006	TOPPERS/HRPカーネル
4464	0x0007	TOPPERS/ASPカーネル
4465	0x0008	TOPPERS/FMPカーネル
4466	0x0009	TOPPERS/SSPカーネル
4467	0x000a	TOPPERS/ASP Safetyカーネル

4468

4469           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  
4470           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  
4471           番号、下位4ビットがマイナーバージョン番号を表す。

4472

4473           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  
4474           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  
4475           ベルを表す。

4476

4477

## 4478           第3章 システムインタフェースレイヤAPI仕様

4479

### 4480           3.1 システムインタフェースレイヤの概要

4481

4482           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  
4483           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  
4484           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  
4485           トにおいて修正を加えて用いている。

4486

4487           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  
4488           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  
4489           からSILの機能を呼び出した場合の動作は、次の例外を除いては保証されない  
4490           【NGKI0802】。

4491

- 4492           ・ 微少時間待ちの機能を呼び出すこと
- 4493           ・ エンディアン取得のためのマクロを参照すること
- 4494           ・ 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  
4495           スすること
- 4496           ・ 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  
4497           こと

4498

### 4499           3.2 SILヘッダファイル

4500

SILを用いるために必要な定義は、SILヘッダファイル (sil.h) およびそこからインクルードされるファイルに含まれている【NGKI0803】。SILを用いる場合には、SILヘッダファイルをインクルードする【NGKI0804】。

### 3.3 全割込みロック状態の制御

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

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

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

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

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

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

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

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

全割込みロックフラグを、対応するSIL\_LOC\_INTを実行する前の状態に戻す【NGKI0810】。SIL\_LOC\_INTを実行せずにSIL\_UNL\_INTを呼び出した場合の動作は保証されない【NGKI0811】。

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

#### 【補足説明】

全割込みロック状態の制御機能の使用例は次の通り。

```
{
    SIL_PRE_LOC;

    SIL_LOC_INT();
    // この間はNMIを除くすべての割込みがマスクされる。
```

```
4551          // 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）.  
4552          SIL_UNL_INT();  
4553      }
```

### 4554 3.4 SILスピンロック

4555 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  
4556 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  
4557 スクとプロセッサ間ロックの取得により排他制御を行うためのスピンロックの機  
4558 能を用意している。これを、カーネルのスピンロック機能と区別するために、  
4559 SILスピンロックと呼ぶ。

4560 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す  
4561 べての割込み（NMIを除く）がマスクされる【NGKI0812】。ロックが他のプロセッ  
4562 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  
4563 【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  
4564 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）  
4565 【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  
4566 ピンロックを取得するという。また、プロセッサ間ロックを返却し割込みをマ  
4567 スク解除することを、SILスピンロックを返却するという。

4571 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  
4572 【NGKI0815】。

#### 4573 (1) SIL\_PRE\_LOC

4574 全割込みロック状態の制御に必要な変数を宣言するマクロであるが、SILスピン  
4575 ロックの取得・解放にも兼用する【NGKI0816】。

4576 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  
4577 の変数宣言部に記述しなければならない【NGKI0817】。SIL\_LOC\_SPN,  
4578 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  
4579 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  
4580 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  
4581 【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  
4582 【NGKI0819】。

#### 4583 (2) SIL\_LOC\_SPN()

4584 SILスピンロックが取得されていない状態である場合には、プロセッサ間ロック  
4585 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  
4586 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  
4587 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  
4588 【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  
4589 トし、全割込みロック状態に遷移する【NGKI0822】。

#### 4590 (3) SIL\_UNL\_SPN()

4591 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  
4592 実行する前の状態に戻す【NGKI0823】。

4601  
4602 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  
4603 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  
4604 SIL\_UNL\_SPNを呼び出した場合の動作も保証されない【NGKI0825】。

4605  
4606 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  
4607 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  
4608 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。

4609  
4610 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  
4611 されていない【NGKI0826】。

#### 4612 【使用上の注意】

4613  
4614  
4615 全割込みロック状態やCPUロック状態でSIL\_LOC\_SPNを呼び出すことはできるが、  
4616 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  
4617 べきである。

#### 4618 【補足説明】

4619  
4620  
4621 SILスピンロック機能の使用例は次の通り。

```
4622 {  
4623     SIL_PRE_LOC;  
4624  
4625     SIL_LOC_SPN();  
4626     // この間はSILスピンロックを取得している。  
4627     // この間はNMIを除くすべての割込みがマスクされる。  
4628     // 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）。  
4629     SIL_UNL_SPN();  
4630 }  
4631
```

4632

### 4633 3.5 微少時間待ち

4634

4635 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  
4636 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、  
4637 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  
4638 ための以下の機能を用意している。

4639

4640 (1) void sil\_dly\_nse(ulong\_t dlytim)

4641

4642 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  
4643 【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  
4644 合があるので注意すること。

4645

### 4646 3.6 エンディアンの取得

4647

4648 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  
4649 下のマクロを定義している。

4650

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

4701  
4702 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  
4703 アンで書き込む【NGKI0836】。ビッグエンディアンプロセッサでは、  
4704 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  
4705 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。  
4706  
4707 (9) uint32\_t sil\_rew\_mem(const uint32\_t \*mem)  
4708  
4709 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】。  
4710  
4711 (10) void sil\_wrw\_mem(uint32\_t \*mem, uint32\_t data)  
4712  
4713 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  
4714 【NGKI0838】。  
4715  
4716 (11) uint32\_t sil\_rew\_lem(const uint32\_t \*mem)  
4717  
4718 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  
4719 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  
4720 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  
4721 アン変換した値を返す。  
4722  
4723 (12) void sil\_wrw\_lem(uint32\_t \*mem, uint32\_t data)  
4724  
4725 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  
4726 アンで書き込む【NGKI0840】。リトルエンディアンプロセッサでは、  
4727 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  
4728 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。  
4729  
4730 (13) uint32\_t sil\_rew\_bem(const uint32\_t \*mem)  
4731  
4732 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  
4733 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  
4734 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  
4735 アン変換した値を返す。  
4736  
4737 (14) void sil\_wrw\_bem(uint32\_t \*mem, uint32\_t data)  
4738  
4739 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  
4740 アンで書き込む【NGKI0842】。ビッグエンディアンプロセッサでは、  
4741 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  
4742 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。  
4743  
4744 3.8 I/O空間アクセス関数  
4745  
4746 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  
4747 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  
4748 を用意している【NGKI0843】。  
4749  
4750 (1) uint8\_t sil\_reb\_iop(const uint8\_t \*iop)

```

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

```

4764

### 4765 3.9 プロセッサIDの参照

4766

4767 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  
 4768 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。

4769

```

4770 (1) void sil_get_pid(ID *p_prcid)

```

4771

4772 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、  
 4773 p\_prcidで指定したメモリ領域に返す【NGKI0844】。

4774

#### 4775 【使用上の注意】

4776

4777 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  
 4778 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  
 4779 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、  
 4780 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  
 4781 ている可能性があるためである。

4782

4783

## 4784 第4章 カーネルAPI仕様

4785

4786 この章では、カーネルのAPI仕様について規定する。

4787

### 4788 【μITRON4.0仕様との関係】

4789

4790 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  
 4791 らの変更については、個別のAPI仕様では記述しない。

4792

```

4793 INT → int_t
4794 UINT → uint_t
4795 VP → void *
4796 VP_INT → intptr_t

```

4797

### 4798 【μITRON4.0/PX仕様との関係】

4799

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



4801 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  
4802 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  
4803 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  
4804 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。

4805

#### 4806 4.1 タスク管理機能

4807

4808 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  
4809 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】。

4810

4811 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】。

4812

- 4813 ・タスク属性
- 4814 ・タスク状態
- 4815 ・ベース優先度
- 4816 ・現在優先度
- 4817 ・起動要求キューイング数
- 4818 ・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)
- 4819 ・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)
- 4820 ・拡張情報
- 4821 ・メインルーチンの先頭番地
- 4822 ・起動時優先度
- 4823 ・実行時優先度 (TOPPERS/SSPカーネルの場合)
- 4824 ・スタック領域
- 4825 ・システムスタック領域 (保護機能対応カーネルの場合)
- 4826 ・アクセス許可ベクタ (保護機能対応カーネルの場合)
- 4827 ・属する保護ドメイン (保護機能対応カーネルの場合)
- 4828 ・属するクラス (マルチプロセッサ対応カーネルの場合)

4829

4830 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  
4831 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】。

4832

4833 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  
4834 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  
4835 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  
4836 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  
4837 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。

4838

4839 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  
4840 であり、タスクの生成時に0に初期化される【NGKI1005】。

4841

4842 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  
4843 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  
4844 初期割付けプロセッサに初期化される【NGKI1006】。

4845

4846 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、  
4847 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  
4848 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  
4849 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  
4850 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

4851 【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  
 4852 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  
 4853 同じプロセッサで実行される）【NGKI1009】.

4854

4855 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  
 4856 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  
 4857 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  
 4858 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  
 4859 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  
 4860 用いるオブジェクト管理領域と同様に扱われる【NGKI1011】.

4861

4862 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  
 4863 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  
 4864 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  
 4865 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】.

4866

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

4868

4869 TA\_ACT 0x02U タスクの生成時にタスクを起動する  
 4870 TA\_RSTR 0x04U 生成するタスクを制約タスクとする

4871

4872 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  
 4873 【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  
 4874 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  
 4875 の属性を予約している【NGKI1017】.

4876

4877 TA\_FPU FPUレジスタをコンテキストに含める

4878

4879 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）  
 4880 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  
 4881 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  
 4882 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  
 4883 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  
 4884 数から1が減ぜられる【NGKI1180】.

4885

4886 C言語によるタスクの記述形式は次の通り【NGKI1018】.

4887

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

4893

4894 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、  
 4895 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  
 4896 じ動作をする【NGKI1020】.

4897

4898 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.

4899

4900 TMAX\_ACTCNT タスクの起動要求キューイング数の最大値【NGKI1021】

4901  
4902 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  
4903 は、静的APIによって登録されたタスクの数に一致）  
4904 【NGKI1022】  
4905  
4906 【TOPPERS/ASPカーネルにおける規定】  
4907  
4908 ASPカーネルでは、TMAX\_ACTCNTは1に固定されている【ASPS0101】。また、制約  
4909 タスクはサポートしていない【ASPS0102】。ただし、制約タスク拡張パッケージ  
4910 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】。  
4911  
4912 【TOPPERS/FMPカーネルにおける規定】  
4913  
4914 FMPカーネルでは、TMAX\_ACTCNTは1に固定されている【FMPS0101】。また、制約  
4915 タスクはサポートしていない【FMPS0102】。  
4916  
4917 【TOPPERS/HRP2カーネルにおける規定】  
4918  
4919 HRP2カーネルでは、TMAX\_ACTCNTは1に固定されている【HRPS0101】。また、制  
4920 約タスクはサポートしていない【HRPS0102】。  
4921  
4922 【TOPPERS/SSPカーネルにおける規定】  
4923  
4924 SSPカーネルでは、TMAX\_ACTCNTは1に固定されている【SSPS0101】。  
4925  
4926 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  
4927 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  
4928 【SSPS0102】。  
4929  
4930 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  
4931 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  
4932 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  
4933 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  
4934 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  
4935 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。  
4936  
4937 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  
4938 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  
4939 先度は、起動時優先度と同じ値に設定される【SSPS0106】。  
4940  
4941 [実行時優先度によるスタック領域の節約]  
4942  
4943 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  
4944 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、  
4945 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  
4946 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。  
4947  
4948 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  
4949 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  
4950 スクBがある場合、タスクAとタスクBは同時期に共有スタック領域を使用してい

る状態にならない。そのため、タスクAとタスクBの内、サイズが小さい方のスタック領域のサイズは、共有スタック領域のサイズに加える必要がなくなり、スタック領域を節約できることになる。

#### 【μITRON4.0仕様との関係】

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

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

-----  
 CRE\_TSK       タスクの生成 [S]   【NGKI1023】  
 acre\_tsk      タスクの生成 [TD] 【NGKI1024】

#### 【静的API】

\*保護機能対応でないカーネルの場合

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

\*保護機能対応カーネルの場合

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

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

#### 【C言語API】

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

#### 【パラメータ】

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

\*タスクの生成情報 (パケットの内容)

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

#### 【リターンパラメータ】

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

5001 ラーコード

5002

5003 【エラーコード】

5004 E\_CTX コンテキストエラー

5005 ・非タスクコンテキストからの呼出し [s] 【NGKI1026】

5006 ・CPUロック状態からの呼出し [s] 【NGKI1027】

5007 E\_RSATR 予約属性

5008 ・tskatrが無効 【NGKI1028】

5009 ・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]

5010 【NGKI1029】

5011 ・保護ドメインの囲みの中に記述されていない [SP] 【NGKI1030】

5012 ・属するクラスの指定が有効範囲外 [sM] 【NGKI1031】

5013 ・クラスの囲みの中に記述されていない [SM] 【NGKI1032】

5014 E\_PAR パラメータエラー

5015 ・taskがプログラムの先頭番地として正しくない 【NGKI1033】

5016 ・itskpriが有効範囲外 【NGKI1034】

5017 ・その他の条件については機能の項を参照

5018 E\_OACV オブジェクトアクセス違反

5019 ・システム状態に対する管理操作が許可されていない [sP]

5020 【NGKI1035】

5021 E\_MACV メモリアクセス違反

5022 ・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

5023 いない [sP] 【NGKI1036】

5024 E\_NOID ID番号不足

5025 ・割り付けられるタスクIDがない [sD] 【NGKI1037】

5026 E\_NOMEM メモリ不足

5027 ・スタック領域が確保できない 【NGKI1038】

5028 ・システムスタック領域が確保できない [P] 【NGKI1039】

5029 E\_OBJ オブジェクト状態エラー

5030 ・tskidで指定したタスクが登録済み (CRE\_TSKの場合) 【NGKI1040】

5031 ・その他の条件については機能の項を参照

5032

5033 【機能】

5034

5035 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

5036 な振舞いは以下の通り。

5037

5038 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】。

5039 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

5040 は、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

5041 時や、設定されるスタック領域のサイズがターゲット定義の最小値よりも小さ

5042 くなる時には、E\_PARエラーとなる 【NGKI1042】。

5043

5044 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、

5045 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、

5046 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

5047 指定した時には、E\_PARエラーとなる 【NGKI1044】。

5048

5049 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、

5050 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

5051 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  
5052 タスクは実行できる状態になる【NGKI1046】。

5053

5054 静的APIにおいては、tskidはオブジェクト識別名、tskatr, itskpri, stkszは  
5055 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  
5056 【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  
5057 を検出することができない【NGKI1048】。

5058

5059 [stkにNULLを指定した場合]

5060

5061 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  
5062 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  
5063 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  
5064 るように大きい方に丸めたサイズで確保される【NGKI1050】。

5065

5066 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  
5067 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（ユーザスタッ  
5068 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  
5069 て、メモリオブジェクトとしてカーネルに登録される【NGKI1051】。

5070

5071 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  
5072 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  
5073 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  
5074 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）  
5075 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  
5076 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  
5077 【NGKI1054】。

5078

5079 [stkにNULL以外を指定した場合]

5080

5081 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  
5082 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  
5083 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  
5084 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  
5085 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】。

5086

5087 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、  
5088 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  
5089 まれない場合、E\_OBJエラーとなる【NGKI1057】。

5090

5091 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  
5092 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  
5093 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  
5094 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  
5095 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  
5096 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  
5097 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  
5098 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  
5099 【NGKI1060】。

5100

5101       [sstkとsstkszの扱い]

5102

5103       保護機能対応カーネルにおけるsstkとsstkszの扱いは、生成するタスクがユー  
5104       ザタスクの場合とシステムタスクの場合で異なる。

5105

5106       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。

5107

5108       sstkの記述を省略するか、sstkをNULLとした場合、sstkszで指定したサイズの  
5109       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  
5110       る【NGKI1061】。sstkszにターゲット定義の制約に合致しないサイズを指定し  
5111       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  
5112       確保される【NGKI1062】。sstkszの記述も省略した場合には、ターゲット定義  
5113       のデフォルトのサイズで確保される【NGKI1063】。

5114

5115       sstkにNULL以外を指定した場合、sstkとsstkszで指定したスタック領域は、ア  
5116       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  
5117       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  
5118       を参照すること。その方法に従わず、sstkやsstkszにターゲット定義の制約に  
5119       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

5120       【NGKI1065】。また、stkとstkszで指定したシステムスタック領域がカーネル  
5121       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】。

5122

5123       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。

5124

5125       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  
5126       た場合には、E\_PARエラーとなる【NGKI1068】。

5127

5128       sstkszに0以外の値を指定した場合で、stkがNULLの場合には、コンフィギュレー  
5129       タまたはカーネルにより確保されるスタック領域のサイズに、sstkszが加えら  
5130       れる【NGKI1069】。stkszにsstkszを加えた値が、ターゲット定義の制約に合致  
5131       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  
5132       に丸めたサイズで確保される【NGKI1070】。

5133

5134       sstkszに0以外の値を指定した場合で、stkがNULLでない場合には、E\_PARエラー  
5135       となる【NGKI1071】。

5136

5137       sstkszに0を指定した場合、これらの処理は行わず、E\_PARエラーにもならない  
5138       【NGKI1072】。

5139

5140       【TOPPERS/ASPカーネルにおける規定】

5141

5142       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  
5143       成機能拡張パッケージでは、acre\_tskもサポートする【ASPS0105】。

5144

5145       【TOPPERS/FMPカーネルにおける規定】

5146

5147       FMPカーネルでは、CRE\_TSKのみをサポートする【FMPS0103】。

5148

5149       【TOPPERS/HRP2カーネルにおける規定】

5150

5151 HRP2カーネルでは、CRE\_TSKのみをサポートする【HRPS0103】。  
5152  
5153 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  
5154 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  
5155 ネルがスタック領域を確保する機能はサポートしない。stkにNULLを指定した場  
5156 合には、E\_NOSPTエラーとなる【HRPS0176】。  
5157  
5158 【TOPPERS/SSPカーネルにおける規定】  
5159  
5160 SSPカーネルでは、CRE\_TSKのみをサポートする【SSPS0107】。  
5161  
5162 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  
5163 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  
5164 【SSPS0109】。  
5165  
5166 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  
5167 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】。  
5168  
5169 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  
5170 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、  
5171 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  
5172 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、  
5173 E\_PARエラーとなる【SSPS0112】。  
5174  
5175 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。  
5176  
5177 【μITRON4.0仕様との関係】  
5178  
5179 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  
5180 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  
5181 した。  
5182  
5183 【μITRON4.0/PX仕様との関係】  
5184  
5185 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  
5186 で確保する方法を規定した。  
5187  
5188 【未決定事項】  
5189  
5190 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  
5191 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。  
5192  
5193 【仕様決定の理由】  
5194  
5195 保護機能対応カーネルにおいて、sstkszおよびsstkの記述は省略することがで  
5196 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  
5197 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  
5198 るためである。  
5199 -----  
5200 AID\_TSK 割付け可能なタスクIDの数の指定〔SD〕【NGKI1073】



5201  
5202 **【静的API】**  
5203     AID\_TSK(uint\_t notsk)  
5204  
5205 **【パラメータ】**  
5206     uint\_t           notsk           割付け可能なタスクIDの数  
5207  
5208 **【エラーコード】**  
5209     E\_RSATR          予約属性  
5210                      ・保護ドメインの囲みの中に記述されている [P] 【NGKI3428】  
5211                      ・クラスの囲みの中に記述されていない [M] 【NGKI1075】  
5212     E\_PAR            パラメータエラー  
5213                      ・notskが負の値 【NGKI3276】  
5214  
5215 **【機能】**  
5216  
5217 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  
5218 付け可能なタスクIDとして確保する 【NGKI1076】。  
5219  
5220 notskは整数定数式パラメータである 【NGKI1077】。  
5221  
5222 **【TOPPERS/ASPカーネルにおける規定】**  
5223  
5224 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  
5225 **【ASPS0210】**。  
5226  
5227 **【TOPPERS/HRP2カーネルにおける規定】**  
5228  
5229 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  
5230 **【HRPS0211】**。  
5231 -----  
5232 SAC\_TSK            タスクのアクセス許可ベクタの設定 [SP] 【NGKI1078】  
5233 sac\_tsk            タスクのアクセス許可ベクタの設定 [TPD] 【NGKI1079】  
5234  
5235 **【静的API】**  
5236     SAC\_TSK(ID tskid, { ACPTN acptn1, ACPTN acptn2,  
5237                                                           ACPTN acptn3, ACPTN acptn4 })  
5238  
5239 **【C言語API】**  
5240     ER ercd = sac\_tsk(ID tskid, const ACVCT \*p\_acvct)  
5241  
5242 **【パラメータ】**  
5243     ID              tskid           対象タスクのID番号  
5244     ACVCT \*        p\_acvct        アクセス許可ベクタを入れたパケットへのポ  
5245                                   インタ（静的APIを除く）  
5246  
5247     \*アクセス許可ベクタ（パケットの内容）  
5248         ACPTN       acptn1        通常操作1のアクセス許可パターン  
5249         ACPTN       acptn2        通常操作2のアクセス許可パターン  
5250         ACPTN       acptn3        管理操作のアクセス許可パターン

5251 ACPTN acptn4 参照操作のアクセス許可パターン  
5252  
5253 **【リターンパラメータ】**  
5254 ER ercd 正常終了 (E\_OK) またはエラーコード  
5255  
5256 **【エラーコード】**  
5257 E\_CTX コンテキストエラー  
5258 ・非タスクコンテキストからの呼出し [s] **【NGKI1080】**  
5259 ・CPUロック状態からの呼出し [s] **【NGKI1081】**  
5260 E\_ID 不正ID番号  
5261 ・tskidが有効範囲外 [s] **【NGKI1082】**  
5262 E\_RSATR 予約属性  
5263 ・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  
5264 いない [S] **【NGKI1083】**  
5265 ・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  
5266 [SM] **【NGKI1084】**  
5267 E\_NOEXS オブジェクト未登録  
5268 ・対象タスクが未登録 **【NGKI1085】**  
5269 E\_OACV オブジェクトアクセス違反  
5270 ・対象タスクに対する管理操作が許可されていない [s] **【NGKI1086】**  
5271 E\_MACV メモリアクセス違反  
5272 ・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  
5273 いない [s] **【NGKI1087】**  
5274 E\_OBJ オブジェクト状態エラー  
5275 ・対象タスクは静的APIで生成された [s] **【NGKI1088】**  
5276 ・対象タスクに対してアクセス許可ベクタが設定済み [S]  
5277 **【NGKI1089】**  
5278  
5279 **【機能】**  
5280  
5281 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  
5282 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】** .  
5283  
5284 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  
5285 式パラメータである **【NGKI1091】** .  
5286  
5287 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  
5288 となる **【NGKI1092】** .  
5289  
5290 **【TOPPERS/HRP2カーネルにおける規定】**  
5291  
5292 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  
5293 成機能拡張パッケージでは、sac\_tskもサポートする **【HRPS0177】** .  
5294 -----  
5295 DEF\_EPR タスクの実行時優先度の定義 [S] **【NGKI1093】**  
5296  
5297 **【静的API】**  
5298 DEF\_EPR(ID tskid, { PRI exePRI })  
5299  
5300 **【パラメータ】**

5301 ID tskid 対象タスクのID番号  
 5302 PRI exeprl タスクの実行時優先度  
 5303  
 5304 **【エラーコード】**  
 5305 E\_PAR パラメータエラー  
 5306 ・exeprlが有効範囲外【NGKI1094】  
 5307 E\_ILUSE サービスコール不正使用  
 5308 ・条件については機能の項を参照  
 5309 E\_OBJ オブジェクト状態エラー  
 5310 ・対象タスクに対して実行優先度が設定済み【NGKI1095】  
 5311  
 5312 **【サポートするカーネル】**  
 5313  
 5314 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  
 5315 ネルは、DEF\_EPRをサポートしない【NGKI1096】。  
 5316  
 5317 **【機能】**  
 5318  
 5319 tskidで指定したタスク（対象タスク）の実行時優先度を、exeprlで指定した優  
 5320 先度に設定する【NGKI1097】。  
 5321  
 5322 tskidはオブジェクト識別名、exeprlは整数定数式パラメータである【NGKI1098】。  
 5323  
 5324 exeprl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  
 5325 る【NGKI1099】。  
 5326  
 5327 **【 $\mu$  ITRON4.0仕様との関係】**  
 5328  
 5329  $\mu$  ITRON4.0仕様に定義されていない静的APIである。  
 5330 -----  
 5331 del\_tsk タスクの削除 [TD] 【NGKI1100】  
 5332  
 5333 **【C言語API】**  
 5334 ER ercd = del\_tsk(ID tskid)  
 5335  
 5336 **【パラメータ】**  
 5337 ID tskid 対象タスクのID番号  
 5338  
 5339 **【リターンパラメータ】**  
 5340 ER ercd 正常終了 (E\_OK) またはエラーコード  
 5341  
 5342 **【エラーコード】**  
 5343 E\_CTX コンテキストエラー  
 5344 ・非タスクコンテキストからの呼出し【NGKI1101】  
 5345 ・CPUロック状態からの呼出し【NGKI1102】  
 5346 E\_ID 不正ID番号  
 5347 ・tskidが有効範囲外【NGKI1103】  
 5348 E\_NOEXS オブジェクト未登録  
 5349 ・対象タスクが未登録【NGKI1104】  
 5350 E\_OACV オブジェクトアクセス違反

5351                   ・対象タスクに対する管理操作が許可されていない [P] 【NGKI1105】  
 5352           E\_OBJ   オブジェクト状態エラー  
 5353                   ・対象タスクが休止状態でない 【NGKI1106】  
 5354                   ・対象タスクは静的APIで生成された 【NGKI1107】  
 5355  
 5356       【機能】  
 5357  
 5358       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  
 5359       り．  
 5360  
 5361       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  
 5362       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  
 5363       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  
 5364       場合は，それらのメモリ領域が解放される 【NGKI1109】．  
 5365  
 5366       【TOPPERS/ASPカーネルにおける規定】  
 5367  
 5368       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  
 5369       機能拡張パッケージでは，del\_tskをサポートする 【ASPS0108】．  
 5370  
 5371       【TOPPERS/FMPカーネルにおける規定】  
 5372  
 5373       FMPカーネルでは，del\_tskをサポートしない 【FMPS0105】．  
 5374  
 5375       【TOPPERS/HRP2カーネルにおける規定】  
 5376  
 5377       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  
 5378       機能拡張パッケージでは，del\_tskをサポートする 【HRPS0178】．  
 5379  
 5380       【TOPPERS/SSPカーネルにおける規定】  
 5381  
 5382       SSPカーネルでは，del\_tskをサポートしない 【SSPS0114】．  
 5383       -----  
 5384       act\_tsk       タスクの起動 [T]   【NGKI1110】  
 5385       iact\_tsk     タスクの起動 [I]   【NGKI1111】  
 5386  
 5387       【C言語API】  
 5388           ER ercd = act\_tsk(ID tskid)  
 5389           ER ercd = iact\_tsk(ID tskid)  
 5390  
 5391       【パラメータ】  
 5392           ID           tskid           対象タスクのID番号  
 5393  
 5394       【リターンパラメータ】  
 5395           ER           ercd           正常終了 (E\_OK) またはエラーコード  
 5396  
 5397       【エラーコード】  
 5398           E\_CTX       コンテキストエラー  
 5399                    ・非タスクコンテキストからの呼出し (act\_tskの場合) 【NGKI1112】  
 5400                    ・タスクコンテキストからの呼出し (iact\_tskの場合) 【NGKI1113】

5401                   ・CPUロック状態からの呼出し【NGKI1114】  
 5402       E\_ID       不正ID番号  
 5403                   ・tskidが有効範囲外【NGKI1115】  
 5404       E\_NOEXS   オブジェクト未登録  
 5405                   ・対象タスクが未登録 [D] 【NGKI1116】  
 5406       E\_OACV   オブジェクトアクセス違反  
 5407                   ・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  
 5408                    の場合) [P] 【NGKI1117】  
 5409       E\_QOVR   キューイングオーバフロー  
 5410                   ・条件については機能の項を参照

#### 5411       【機能】

5412       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  
 5413       いは以下の通り．

5414       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  
 5415       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】．

5416       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  
 5417       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  
 5418       TMAX\_ACTCNTを超える場合には，E\_QOVRエラーとなる【NGKI1120】．

5419       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  
 5420       となる【NGKI1121】．

#### 5421       【補足説明】

5422       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  
 5423       起動時の割付けプロセッサを変更しない．

---

5424       mact\_tsk   割付けプロセッサ指定でのタスクの起動 [TM] 【NGKI1122】  
 5425       imact\_tsk   割付けプロセッサ指定でのタスクの起動 [IM] 【NGKI1123】

#### 5426       【C言語API】

5427       ER ercd = mact\_tsk(ID tskid, ID prcid)  
 5428       ER ercd = imact\_tsk(ID tskid, ID prcid)

#### 5429       【パラメータ】

5430       ID       tskid       対象タスクのID番号  
 5431       ID       prcid       タスクの割付け対象のプロセッサのID番号

#### 5432       【リターンパラメータ】

5433       ER       ercd       正常終了 (E\_OK) またはエラーコード

#### 5434       【エラーコード】

5435       E\_CTX     コンテキストエラー  
 5436                   ・非タスクコンテキストからの呼出し (mact\_tskの場合)  
 5437                    【NGKI1124】  
 5438                   ・タスクコンテキストからの呼出し (imact\_tskの場合)

5451                               【NGKI1125】  
5452                               ・CPUロック状態からの呼出し【NGKI1126】  
5453       E\_NOSPT       未サポート機能  
5454                               ・対象タスクが制約タスク【NGKI1127】  
5455       E\_ID       不正ID番号  
5456                               ・tskidが有効範囲外【NGKI1128】  
5457                               ・prcidが有効範囲外【NGKI1129】  
5458       E\_PAR       パラメータエラー  
5459                               ・条件については機能の項を参照  
5460       E\_NOEXS       オブジェクト未登録  
5461                               ・対象タスクが未登録 [D] 【NGKI1130】  
5462       E\_OACV       オブジェクトアクセス違反  
5463                               ・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  
5464                               の場合) [P] 【NGKI1131】  
5465       E\_QOVR       キューイングオーバーフロー  
5466                               ・条件については機能の項を参照  
5467  
5468       【機能】  
5469  
5470       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  
5471       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．  
5472  
5473       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  
5474       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  
5475       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  
5476       【NGKI1132】．  
5477  
5478       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  
5479       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  
5480       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  
5481       TMAX\_ACTCNTを超える場合には、E\_QOVRエラーとなる【NGKI1134】．  
5482  
5483       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  
5484       クとなる【NGKI1135】．  
5485  
5486       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  
5487       サを含んでいない場合には、E\_PARエラーとなる【NGKI1136】．  
5488  
5489       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  
5490       れが属するクラスの初期割付けプロセッサとする【NGKI1137】．  
5491  
5492       【補足説明】  
5493  
5494       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  
5495       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  
5496       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  
5497       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  
5498       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  
5499       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  
5500       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

5501 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  
5502 変更されない。

5503

5504 【TOPPERS/ASPカーネルにおける規定】

5505

5506 ASPカーネルでは、mact\_tsk, imact\_tskをサポートしない【ASPS0109】。

5507

5508 【TOPPERS/HRP2カーネルにおける規定】

5509

5510 HRP2カーネルでは、mact\_tsk, imact\_tskをサポートしない【HRPS0106】。

5511

5512 【TOPPERS/SSPカーネルにおける規定】

5513

5514 SSPカーネルでは、mact\_tsk, imact\_tskをサポートしない【SSPS0115】。

5515

5516 【 $\mu$  ITRON4.0仕様との関係】

5517

5518  $\mu$  ITRON4.0仕様に定義されていないサービスコールである。

5519

5520 can\_act      タスク起動要求のキャンセル [T]   【NGKI1138】

5521

5522 【C言語API】

5523      ER\_UINT actcnt = can\_act(ID tskid)

5524

5525 【パラメータ】

5526      ID                  tskid                  対象タスクのID番号

5527

5528 【リターンパラメータ】

5529      ER\_UINT          actcnt                  キューイングされていた起動要求の数（正の値  
5530                                                  または0）またはエラーコード

5531

5532 【エラーコード】

5533      E\_CTX                  コンテキストエラー

5534                                  ・非タスクコンテキストからの呼出し【NGKI1139】

5535                                  ・CPUロック状態からの呼出し【NGKI1140】

5536      E\_ID                  不正ID番号

5537                                  ・tskidが有効範囲外【NGKI1141】

5538      E\_NOEXS              オブジェクト未登録

5539                                  ・対象タスクが未登録 [D]   【NGKI1142】

5540      E\_OACV                  オブジェクトアクセス違反

5541                                  ・対象タスクに対する通常操作1が許可されていない [P]  
5542                                                  【NGKI1143】

5543

5544 【機能】

5545

5546 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  
5547 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  
5548 下の通り。

5549

5550 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

5551 求キューイング数が，サービスコールの返値として返される【NGKI1144】．ま  
5552 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  
5553 付けプロセッサが未設定状態に戻される【NGKI1145】．

5554  
5555 tskidにTSK\_SELF (=0) を指定すると，自タスクが対象タスクとなる  
5556 【NGKI1146】．

5557  
5558 【TOPPERS/SSPカーネルにおける規定】  
5559

5560 SSPカーネルでは，can\_actをサポートしない【SSPS0116】．

5561 -----  
5562 mig\_tsk      タスクの割付けプロセッサの変更 [TM]    【NGKI1147】

5563  
5564 【C言語API】  
5565     ER ercd = mig\_tsk(ID tskid, ID prcid)  
5566

5567 【パラメータ】  
5568     ID            tskid            対象タスクのID番号  
5569     ID            prcid            タスクの割付けプロセッサのID番号

5570  
5571 【リターンパラメータ】  
5572     ER            ercd            正常終了 (E\_OK) またはエラーコード

5573  
5574 【エラーコード】  
5575     E\_CTX        コンテキストエラー  
5576                   ・非タスクコンテキストからの呼出し【NGKI1148】  
5577                   ・CPUロック状態からの呼出し【NGKI1149】  
5578                   ・その他の条件については機能の項を参照  
5579     E\_NOSPT      未サポート機能  
5580                   ・対象タスクが制約タスク【NGKI1150】  
5581     E\_ID         不正ID番号  
5582                   ・tskidが有効範囲外【NGKI1151】  
5583                   ・prcidが有効範囲外【NGKI1152】  
5584     E\_PAR        パラメータエラー  
5585                   ・条件については機能の項を参照  
5586     E\_NOEXS      オブジェクト未登録  
5587                   ・対象タスクが未登録 [D]    【NGKI1153】  
5588     E\_OACV       オブジェクトアクセス違反  
5589                   ・対象タスクに対する通常操作1が許可されていない [P]  
5590                   【NGKI1154】  
5591     E\_OBJ        オブジェクト状態エラー  
5592                   ・条件については機能の項を参照

5593  
5594 【機能】  
5595

5596 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  
5597 変更する．具体的な振舞いは以下の通り．

5598  
5599 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  
5600 合には，対象タスクをprcidで指定したプロセッサに割り付ける【NGKI1155】．



5601 対象タスクが実行できる状態の場合には、prcidで指定したプロセッサに割り付  
5602 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  
5603 【NGKI1156】。  
5604  
5605 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  
5606 付けられている場合には、E\_OBJエラーとなる【NGKI1157】。  
5607  
5608 tskidにTSK\_SELF (=0) を指定すると、自タスクが対象タスクとなる  
5609 【NGKI1158】。  
5610  
5611 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、  
5612 E\_CTXエラーとなる【NGKI1159】。  
5613  
5614 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  
5615 サを含んでいない場合には、E\_PARエラーとなる【NGKI1160】。  
5616  
5617 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  
5618 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】。  
5619  
5620 【補足説明】  
5621  
5622 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  
5623 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  
5624 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  
5625 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  
5626 のを防ぐことができる。  
5627  
5628 対象タスクが、最初からprcidで指定したプロセッサに割り付けられている場合  
5629 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  
5630 クの中で最低となる。  
5631  
5632 【TOPPERS/ASPカーネルにおける規定】  
5633  
5634 ASPカーネルでは、mig\_tskをサポートしない【ASPS0110】。  
5635  
5636 【TOPPERS/HRP2カーネルにおける規定】  
5637  
5638 HRP2カーネルでは、mig\_tskをサポートしない【HRPS0107】。  
5639  
5640 【TOPPERS/SSPカーネルにおける規定】  
5641  
5642 SSPカーネルでは、mig\_tskをサポートしない【SSPS0117】。  
5643  
5644 【μITRON4.0仕様との関係】  
5645  
5646 μITRON4.0仕様に定義されていないサービスコールである。  
5647 -----  
5648 ext\_tsk 自タスクの終了 [T] 【NGKI1162】  
5649  
5650 【C言語API】

5651           ER ercd = ext\_tsk()

5652

5653   **【パラメータ】**

5654       なし

5655

5656   **【リターンパラメータ】**

5657       ER           ercd           エラーコード

5658

5659   **【エラーコード】**

5660       E\_SYS       システムエラー

5661                   ・カーネルの誤動作【NGKI1163】

5662       E\_CTX       コンテキストエラー

5663                   ・非タスクコンテキストからの呼出し【NGKI1164】

5664

5665   **【機能】**

5666

5667   自タスクを終了させる．具体的には，自タスクに対してタスク終了時に行うべき処理が行われる【NGKI3449】．

5669

5670   ext\_tskは，CPUロック解除状態，割込み優先度マスク全解除状態，ディスパッチ許可状態で呼び出すのが原則であるが，そうでない状態で呼び出された場合には，CPUロック解除状態，割込み優先度マスク全解除状態，ディスパッチ許可状態に遷移させた後，自タスクを終了させる【NGKI1168】．

5674

5675   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】．

5676

5677   **【TOPPERS/SSPカーネルにおける規定】**

5678

5679   SSPカーネルでは，ext\_tskをサポートしない【SSPS0118】．自タスクを終了させる場合には，タスクのメインルーチンからリターンする【SSPS0119】．

5681

5682   **【μITRON4.0仕様との関係】**

5683

5684   ext\_tskを非タスクコンテキストから呼び出した場合に，E\_CTXエラーが返ることとした．μITRON4.0仕様においては，ext\_tskからはリターンしないと規定されている．

5687

5688   -----  
ter\_tsk       タスクの強制終了〔T〕【NGKI1170】

5689

5690   **【C言語API】**

5691       ER ercd = ter\_tsk(ID tskid)

5692

5693   **【パラメータ】**

5694       ID           tskid       対象タスクのID番号

5695

5696   **【リターンパラメータ】**

5697       ER           ercd       正常終了 (E\_OK) またはエラーコード

5698

5699   **【エラーコード】**

5700       E\_CTX       コンテキストエラー

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

#### 5715           【機能】

5716  
 5717  
 5718   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  
 5719   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  
 5720   が行われる【NGKI3450】．

5721  
 5722   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  
 5723   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  
 5724   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】．

#### 5725           【TOPPERS/FMPカーネルにおける使用上の注意】

5726  
 5727  
 5728   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  
 5729   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  
 5730   方式にも依存する）．

#### 5731           【TOPPERS/SSPカーネルにおける規定】

5732  
 5733   SSPカーネルでは、ter\_tskをサポートしない【SSPS0120】．

5734  
 5735   -----  
 5736   chg\_pri       タスクのベース優先度の変更 [T] 【NGKI1183】

#### 5737           【C言語API】

5738           ER ercd = chg\_pri(ID tskid, PRI tskpri)

#### 5739           【パラメータ】

5740  
 5741           ID           tskid       対象タスクのID番号  
 5742           PRI          tskpri      ベース優先度

#### 5743           【リターンパラメータ】

5744           ER           ercd       正常終了 (E\_OK) またはエラーコード

#### 5745           【エラーコード】

5746           E\_CTX       コンテキストエラー  
 5747                   ・非タスクコンテキストからの呼出し【NGKI1184】  
 5748  
 5749  
 5750

5751                   ・CPUロック状態からの呼出し【NGKI1185】  
5752       E\_NOSPT   未サポート機能  
5753                   ・対象タスクが制約タスク【NGKI1186】  
5754       E\_ID      不正ID番号  
5755                   ・tskidが有効範囲外【NGKI1187】  
5756       E\_PAR      パラメータエラー  
5757                   ・tskpriが有効範囲外【NGKI1188】  
5758       E\_NOEXS   オブジェクト未登録  
5759                   ・対象タスクが未登録 [D] 【NGKI1189】  
5760       E\_OACV     オブジェクトアクセス違反  
5761                   ・対象タスクに対する通常操作2が許可されていない [P]  
5762                   【NGKI1190】  
5763       E\_ILUSE    サービスコール不正使用  
5764                   ・条件については機能の項を参照  
5765       E\_OBJ      オブジェクト状態エラー  
5766                   ・対象タスクが休止状態【NGKI1191】  
5767  
5768       **【機能】**  
5769  
5770       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  
5771       先度に変更する。具体的な振舞いは以下の通り。  
5772  
5773       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  
5774       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  
5775       在優先度も変更される【NGKI1193】。  
5776  
5777       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  
5778       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  
5779       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  
5780       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  
5781       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  
5782       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  
5783       の中で最後になる【NGKI1196】。  
5784  
5785       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  
5786       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  
5787       【NGKI1197】。  
5788  
5789       tskidにTSK\_SELF (=0) を指定すると、自タスクが対象タスクとなる  
5790       【NGKI1198】。また、tskpriにTPRI\_INI (=0) を指定すると、対象タスクのベー  
5791       ス優先度が、起動時優先度に変更される【NGKI1199】。  
5792  
5793       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  
5794       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  
5795       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】。  
5796  
5797       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  
5798       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  
5799       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  
5800       なる【NGKI3440】。

5801  
 5802     **【TOPPERS/SSPカーネルにおける規定】**  
 5803  
 5804     SSPカーネルでは、chg\_priをサポートしない【SSPS0121】。  
 5805  
 5806     **【 $\mu$ ITRON4.0仕様との関係】**  
 5807  
 5808     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  
 5809     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  
 5810     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。  
 5811  
 5812     -----  
 5812     get\_pri     タスク優先度の参照 [T]     【NGKI1202】  
 5813  
 5814     **【C言語API】**  
 5815     ER ercd = get\_pri(ID tskid, PRI \*p\_tskpri)  
 5816  
 5817     **【パラメータ】**  
 5818     ID           tskid           対象タスクのID番号  
 5819     PRI \*        p\_tskpri       現在優先度を入れるメモリ領域へのポインタ  
 5820  
 5821     **【リターンパラメータ】**  
 5822     ER           ercd           正常終了 (E\_OK) またはエラーコード  
 5823     PRI           tskpri       現在優先度  
 5824  
 5825     **【エラーコード】**  
 5826     E\_CTX        コンテキストエラー  
 5827                  ・非タスクコンテキストからの呼出し【NGKI1203】  
 5828                  ・CPUロック状態からの呼出し【NGKI1204】  
 5829     E\_ID         不正ID番号  
 5830                  ・tskidが有効範囲外【NGKI1205】  
 5831     E\_NOEXS      オブジェクト未登録  
 5832                  ・対象タスクが未登録 [D]     【NGKI1206】  
 5833     E\_OACV       オブジェクトアクセス違反  
 5834                  ・対象タスクに対する参照操作が許可されていない [P]     【NGKI1207】  
 5835     E\_MACV       メモリアクセス違反  
 5836                  ・p\_tskpriが指すメモリ領域への書込みアクセスが許可され  
 5837                  ていない [P]     【NGKI1208】  
 5838     E\_OBJ        オブジェクト状態エラー  
 5839                  ・対象タスクが休止状態【NGKI1209】  
 5840  
 5841     **【機能】**  
 5842  
 5843     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  
 5844     いは以下の通り。  
 5845  
 5846     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  
 5847     が指すメモリ領域に返される【NGKI1210】。  
 5848  
 5849     tskidにTSK\_SELF (=0) を指定すると、自タスクが対象タスクとなる  
 5850     【NGKI1211】。

5851  
5852     **【TOPPERS/SSPカーネルにおける規定】**  
5853  
5854     SSPカーネルでは、get\_priをサポートしない【SSPS0122】。  
5855     -----  
5856     get\_inf     自タスクの拡張情報の参照 [T]     【NGKI1212】  
5857  
5858     **【C言語API】**  
5859         ER ercd = get\_inf(intptr\_t \*p\_exinf)  
5860  
5861     **【パラメータ】**  
5862         intptr\_t \* p\_exinf     拡張情報を入れるメモリ領域へのポインタ  
5863  
5864     **【リターンパラメータ】**  
5865         ER             ercd     正常終了 (E\_OK) またはエラーコード  
5866         intptr\_t     exinf     拡張情報  
5867  
5868     **【エラーコード】**  
5869         E\_CTX         コンテキストエラー  
5870             ・非タスクコンテキストからの呼出し【NGKI1213】  
5871             ・CPUロック状態からの呼出し【NGKI1214】  
5872         E\_MACV         メモリアクセス違反  
5873             ・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  
5874             いない [P]     【NGKI1215】  
5875  
5876     **【機能】**  
5877  
5878     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  
5879     域に返される【NGKI1216】。  
5880  
5881     **【TOPPERS/SSPカーネルにおける規定】**  
5882  
5883     SSPカーネルでは、get\_infをサポートしない【SSPS0123】。  
5884  
5885     **【μITRON4.0仕様との関係】**  
5886  
5887     μITRON4.0仕様に定義されていないサービスコールである。  
5888     -----  
5889     ref\_tsk     タスクの状態参照 [T]     【NGKI1217】  
5890  
5891     **【C言語API】**  
5892         ER ercd = ref\_tsk(ID tskid, T\_RTsk \*pk\_rtsk)  
5893  
5894     **【パラメータ】**  
5895         ID             tskid     対象タスクのID番号  
5896         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  
5897  
5898     **【リターンパラメータ】**  
5899         ER             ercd     正常終了 (E\_OK) またはエラーコード  
5900

5901       \*タスクの現在状態（パケットの内容）

5902	STAT	tskstat	タスク状態
5903	PRI	tskpri	タスクの現在優先度
5904	PRI	tskbpri	タスクのベース優先度
5905	STAT	tskwait	タスクの待ち要因
5906	ID	wobjid	タスクの待ち対象のオブジェクトのID
5907	TMO	lefttmo	タスクがタイムアウトするまでの時間
5908	uint_t	actcnt	タスクの起動要求キューイング数
5909	uint_t	wupcnt	タスクの起床要求キューイング数
5910	bool_t	texmsk	タスクがタスク例外処理マスク状態か否か（保
5911			護機能対応カーネルの場合）
5912	bool_t	waifbd	タスクが待ち禁止状態か否か（保護機能対応カー
5913			ネルの場合）
5914	uint_t	svclevel	タスクの拡張サービスコールのネストレベル（保
5915			護機能対応カーネルの場合）
5916	ID	prcid	タスクの割付けプロセッサのID（マルチプロセッ
5917			サ対応カーネルの場合）
5918	ID	actprc	タスクの次回起動時の割付けプロセッサのID（マ
5919			ルチプロセッサ対応カーネルの場合）

5920

5921       【エラーコード】

5922	E_CTX	コンテキストエラー
5923		・非タスクコンテキストからの呼出し【NGKI1218】
5924		・CPUロック状態からの呼出し【NGKI1219】
5925	E_ID	不正ID番号
5926		・tskidが有効範囲外【NGKI1220】
5927	E_NOEXS	オブジェクト未登録
5928		・対象タスクが未登録 [D] 【NGKI1221】
5929	E_OACV	オブジェクトアクセス違反
5930		・対象タスクに対する参照操作が許可されていない [P] 【NGKI1222】
5931	E_MACV	メモリアクセス違反
5932		・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
5933		いない [P] 【NGKI1223】

5934

5935       【機能】

5936

5937       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

5938       態は、pk\_rtskで指定したメモリ領域に返される【NGKI1224】．

5939

5940       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

5941       る【NGKI1225】．

5942

5943	TTS_RUN	0x01U	実行状態
5944	TTS_RDY	0x02U	実行可能状態
5945	TTS_WAI	0x04U	待ち状態
5946	TTS_SUS	0x08U	強制待ち状態
5947	TTS_WAS	0x0cU	二重待ち状態
5948	TTS_DMT	0x10U	休止状態

5949

5950       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、

5951 tskstatがTTS\_SUSとなる場合がある【NGKI1226】。この状況は、自タスクに対  
5952 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  
5953 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  
5954 ある。

5955

5956 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、  
5957 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  
5958 休止状態である場合には、tskpriとtsbpriの値は保証されない【NGKI1228】。

5959

5960 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  
5961 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】。

5962

5963	TTW_SLP	0x0001U	起床待ち
5964	TTW_DLY	0x0002U	時間経過待ち
5965	TTW_SEM	0x0004U	セマフォの資源獲得待ち
5966	TTW_FLG	0x0008U	イベントフラグ待ち
5967	TTW_SDTQ	0x0010U	データキューへの送信待ち
5968	TTW_RDTQ	0x0020U	データキューからの受信待ち
5969	TTW_SPDQ	0x0100U	優先度データキューへの送信待ち
5970	TTW_RPDQ	0x0200U	優先度データキューからの受信待ち
5971	TTW_MBX	0x0040U	メールボックスからの受信待ち
5972	TTW_MTX	0x0080U	ミューテックスのロック待ち状態
5973	TTW_SMBF	0x0400U	メッセージバッファへの送信待ち
5974	TTW_RMBF	0x0800U	メッセージバッファからの受信待ち
5975	TTW_MPF	0x2000U	固定長メモリブロックの獲得待ち

5976

5977 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  
5978 【NGKI1230】。

5979

5980 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  
5981 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  
5982 【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  
5983 経過待ち状態である場合には、wobjidの値は保証されない【NGKI1232】。

5984

5985 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  
5986 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  
5987 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  
5988 【NGKI1234】。

5989

5990 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  
5991 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、  
5992 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  
5993 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  
5994 定義される) に型キャストした値が返される【NGKI1236】。

5995

5996 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  
5997 【NGKI1237】。

5998

5999 actent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】。

6000



6001 対象タスクが休止状態でない場合には、wupcntに、タスクの起床要求キューイ  
6002 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、  
6003 wupcntの値は保証されない【NGKI1240】。  
6004  
6005 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、  
6006 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  
6007 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  
6008 にtrue、そうでない場合にfalseが返される【NGKI1242】。またsvclevelには、  
6009 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  
6010 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  
6011 【NGKI1243】。対象タスクが休止状態である場合には、texmsk, waifbd,  
6012 svclevelの値は保証されない【NGKI1244】。  
6013  
6014 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  
6015 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  
6016 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  
6017 ロセッサが未設定の場合には、actprcにTPRC\_NONE (=0) が返される  
6018 【NGKI1247】。  
6019  
6020 tskidにTSK\_SELF (=0) を指定すると、自タスクが対象タスクとなる  
6021 【NGKI1248】。  
6022  
6023 【補足説明】  
6024  
6025 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  
6026 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  
6027 を正しく得ることができる。  
6028  
6029 【TOPPERS/ASPカーネルにおける規定】  
6030  
6031 ASPカーネルでは、tskwaitにTTW\_MTX, TTW\_SMBF, TTW\_RMBFが返ることはない  
6032 【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、  
6033 tskwaitにTTW\_MTXが返る場合がある【ASPS0112】。また、メッセージバッファ  
6034 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  
6035 る【ASPS0208】。  
6036  
6037 【TOPPERS/FMPカーネルにおける規定】  
6038  
6039 FMPカーネルでは、tskwaitにTTW\_MTX, TTW\_SMBF, TTW\_RMBFが返ることはない  
6040 【FMPS0106】。  
6041  
6042 【TOPPERS/HRP2カーネルにおける規定】  
6043  
6044 HRP2カーネルでは、tskwaitにTTW\_MBX, TTW\_SMBF, TTW\_RMBFが返ることはない  
6045 【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、  
6046 tskwaitにTTW\_SMBFとTTW\_RMBFが返る場合がある【HRPS0174】。  
6047  
6048 【TOPPERS/SSPカーネルにおける規定】  
6049  
6050 SSPカーネルでは、ref\_tskをサポートしない【SSPS0124】。

**【使用上の注意】**

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

**【 $\mu$  ITRON4.0仕様との関係】**

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

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

**【 $\mu$  ITRON4.0/PX仕様との関係】**

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

**4.2 タスク付属同期機能**

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

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

**・起床要求キューイング数**

タスクの起床要求キューイング数は、処理されていないタスクの起床要求の数であり、タスクの起動時に0に初期化される【NGKI1250】。

タスク付属同期機能に関連するカーネル構成マクロは次の通り。

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

**【TOPPERS/ASPカーネルにおける規定】**

ASPカーネルでは、TMAX\_WUPCNTは1に固定されている【ASPS0113】。

**【TOPPERS/FMPカーネルにおける規定】**

FMPカーネルでは、TMAX\_WUPCNTは1に固定されている【FMPS0107】。

**【TOPPERS/HRP2カーネルにおける規定】**

6101 HRP2カーネルでは、TMAX\_WUPCNTは1に固定されている【HRPS0109】.

6102

6103 【TOPPERS/SSPカーネルにおける規定】

6104

6105 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】.

6106

6107 【 $\mu$  ITRON4.0仕様との関係】

6108

6109 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.  
 6110 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  
 6111 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  
 6112 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.  
 6113 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  
 6114 スト数 (suscnt) を除外した.

6115

6116 slp\_tsk 起床待ち [T] 【NGKI1252】

6117 tslp\_tsk 起床待ち (タイムアウト付き) [T] 【NGKI1253】

6118

6119 【C言語API】

6120 ER ercd = slp\_tsk()

6121 ER ercd = tslp\_tsk(TMO tmout)

6122

6123 【パラメータ】

6124 TMO tmout タイムアウト時間 (tslp\_tskの場合)

6125

6126 【リターンパラメータ】

6127 ER ercd 正常終了 (E\_OK) またはエラーコード

6128

6129 【エラーコード】

6130 E\_CTX コンテキストエラー

6131 ・ディスパッチ保留状態からの呼出し【NGKI1254】

6132 E\_NOSPT 未サポート機能

6133 ・制約タスクからの呼出し【NGKI1255】

6134 E\_PAR パラメータエラー

6135 ・tmoutが無効 (tslp\_tskの場合)【NGKI1256】

6136 E\_TMOUT ポーリング失敗またはタイムアウト (slp\_tskを除く)【NGKI1257】

6137 E\_RLWAI 待ち禁止状態または待ち状態の強制解除【NGKI1258】

6138

6139 【機能】

6140

6141 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.

6142

6143 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  
 6144 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、  
 6145 自タスクは起床待ち状態となる【NGKI1260】.

6146

6147 【補足説明】

6148

6149 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  
 6150 状態を維持し、自タスクの優先順位は変化しない.

```

6151 -----
6152 wup_tsk      タスクの起床 [T]  【NGKI1261】
6153 iwup_tsk     タスクの起床 [I]  【NGKI1262】
6154
6155 【C言語API】
6156     ER ercd = wup_tsk(ID tskid)
6157     ER ercd = iwup_tsk(ID tskid)
6158
6159 【パラメータ】
6160     ID          tskid      対象タスクのID番号
6161
6162 【リターンパラメータ】
6163     ER          ercd      正常終了 (E_OK) またはエラーコード
6164
6165 【エラーコード】
6166     E_CTX      コンテキストエラー
6167                ・非タスクコンテキストからの呼出し (wup_tskの場合)  【NGKI1263】
6168                ・タスクコンテキストからの呼出し (iwup_tskの場合)  【NGKI1264】
6169                ・CPUロック状態からの呼出し 【NGKI1265】
6170     E_NOSPT    未サポート機能
6171                ・対象タスクが制約タスク 【NGKI1266】
6172     E_ID       不正ID番号
6173                ・tskidが有効範囲外 【NGKI1267】
6174     E_NOEXS    オブジェクト未登録
6175                ・対象タスクが未登録 [D]  【NGKI1268】
6176     E_OACV     オブジェクトアクセス違反
6177                ・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
6178                  の場合) [P]  【NGKI1269】
6179     E_OBJ      オブジェクト状態エラー
6180                ・対象タスクが休止状態 【NGKI1270】
6181     E_QOVR     キューイングオーバフロー
6182                ・条件については機能の項を参照
6183
6184 【機能】
6185
6186 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
6187 り.
6188
6189 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
6190 【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
6191 からE_OKが返る 【NGKI1272】.
6192
6193 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
6194 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
6195 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
6196 【NGKI1274】.
6197
6198 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
6199 となる 【NGKI1275】.
6200 -----

```

```

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

```

6251

6252 **【リターンパラメータ】**

6253 ER ercd 正常終了 (E\_OK) またはエラーコード

6254

6255 **【エラーコード】**

6256 E\_CTX コンテキストエラー

6257 ・非タスクコンテキストからの呼出し (rel\_waiの場合) **【NGKI1288】**

6258 ・タスクコンテキストからの呼出し (irel\_waiの場合) **【NGKI1289】**

6259 ・CPUロック状態からの呼出し **【NGKI1290】**

6260 E\_NOSPT 未サポート機能

6261 ・対象タスクが制約タスク **【NGKI1291】**

6262 E\_ID 不正ID番号

6263 ・tskidが有効範囲外 **【NGKI1292】**

6264 E\_NOEXS オブジェクト未登録

6265 ・対象タスクが未登録 [D] **【NGKI1293】**

6266 E\_OACV オブジェクトアクセス違反

6267 ・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

6268 の場合) [P] **【NGKI1294】**

6269 E\_OBJ オブジェクト状態エラー

6270 ・対象タスクが待ち状態でない **【NGKI1295】**

6271

6272 **【機能】**

6273

6274 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

6275 舞いは以下の通り。

6276

6277 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

6278 **【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

6279 からE\_RLWAIが返る **【NGKI1297】**。

6280 -----

6281 sus\_tsk 強制待ち状態への遷移 [T] **【NGKI1298】**

6282

6283 **【C言語API】**

6284 ER ercd = sus\_tsk(ID tskid)

6285

6286 **【パラメータ】**

6287 ID tskid 対象タスクのID番号

6288

6289 **【リターンパラメータ】**

6290 ER ercd 正常終了 (E\_OK) またはエラーコード

6291

6292 **【エラーコード】**

6293 E\_CTX コンテキストエラー

6294 ・非タスクコンテキストからの呼出し **【NGKI1299】**

6295 ・CPUロック状態からの呼出し **【NGKI1300】**

6296 ・その他の条件については機能の項を参照

6297 E\_NOSPT 未サポート機能

6298 ・対象タスクが制約タスク **【NGKI1301】**

6299 E\_ID 不正ID番号

6300 ・tskidが有効範囲外 **【NGKI1302】**

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

#### 6311   【機能】

6312  
 6313   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  
 6314   下の通り．  
 6315

6316   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  
 6317   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  
 6318   重待ち状態となる【NGKI1308】．  
 6319

6320   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，  
 6321   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  
 6322   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  
 6323   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．  
 6324

6325   tskidにTSK\_SELF（=0）を指定すると，自タスクが対象タスクとなる  
 6326   【NGKI1310】．  
 6327

6328   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，  
 6329   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  
 6330   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  
 6331   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  
 6332   状態で呼び出しても，E\_CTXエラーにはならない【NGKI3604】．これは，  
 6333   [NGKI0175] と [NGKI0179] の原則の例外となっている．  
 6334

6335   rsm\_tsk       強制待ち状態からの再開 [T] 【NGKI1312】  
 6336

#### 6337   【C言語API】

6338       ER ercd = rsm\_tsk(ID tskid)  
 6339

#### 6340   【パラメータ】

6341       ID           tskid       対象タスクのID番号  
 6342

#### 6343   【リターンパラメータ】

6344       ER           ercd       正常終了（E\_OK）またはエラーコード  
 6345

#### 6346   【エラーコード】

6347       E\_CTX       コンテキストエラー  
 6348                    ・非タスクコンテキストからの呼出し 【NGKI1313】  
 6349                    ・CPUロック状態からの呼出し 【NGKI1314】  
 6350       E\_NOSPT    未サポート機能

6351                   ・対象タスクが制約タスク【NGKI1315】  
6352       E\_ID       不正ID番号  
6353                   ・tskidが有効範囲外【NGKI1316】  
6354       E\_NOEXS   オブジェクト未登録  
6355                   ・対象タスクが未登録 [D] 【NGKI1317】  
6356       E\_OACV    オブジェクトアクセス違反  
6357                   ・対象タスクに対する通常操作2が許可されていない [P]  
6358                   【NGKI1318】  
6359       E\_OBJ     オブジェクト状態エラー  
6360                   ・対象タスクが強制待ち状態（二重待ち状態を含む）でない  
6361                   【NGKI1319】  
6362  
6363       **【機能】**  
6364  
6365       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  
6366       舞いは以下の通り．  
6367  
6368       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  
6369       れる【NGKI1320】．  
6370       -----  
6371       dis\_wai     待ち禁止状態への遷移 [TP] 【NGKI1321】  
6372       idis\_wai    待ち禁止状態への遷移 [IP] 【NGKI1322】  
6373  
6374       **【C言語API】**  
6375           ER ercd = dis\_wai(ID tskid)  
6376           ER ercd = idis\_wai(ID tskid)  
6377  
6378       **【パラメータ】**  
6379           ID           tskid           対象タスクのID番号  
6380  
6381       **【リターンパラメータ】**  
6382           ER           ercd           正常終了 (E\_OK) またはエラーコード  
6383  
6384       **【エラーコード】**  
6385       E\_CTX       コンテキストエラー  
6386                   ・非タスクコンテキストからの呼出し (dis\_waiの場合) 【NGKI1323】  
6387                   ・タスクコンテキストからの呼出し (idis\_waiの場合) 【NGKI1324】  
6388                   ・CPUロック状態からの呼出し【NGKI1325】  
6389       E\_NOSPT     未サポート機能  
6390                   ・対象タスクが制約タスク【NGKI1326】  
6391       E\_ID       不正ID番号  
6392                   ・tskidが有効範囲外【NGKI1327】  
6393       E\_NOEXS    オブジェクト未登録  
6394                   ・対象タスクが未登録 [D] 【NGKI1328】  
6395       E\_OACV     オブジェクトアクセス違反  
6396                   ・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  
6397                   の場合) 【NGKI1329】  
6398       E\_OBJ     オブジェクト状態エラー  
6399                   ・対象タスクが休止状態【NGKI1330】  
6400                   ・対象タスクがタスク例外処理マスク状態でない【NGKI1331】



6401           E\_QOVR           キューイングオーバーフロー  
6402                            ・対象タスクが待ち禁止状態【NGKI1332】  
6403  
6404       **【機能】**  
6405  
6406       tskidで指定したタスク（対象タスク）を待ち禁止状態にする．具体的な振舞い  
6407       は以下の通り．  
6408  
6409       対象タスクがタスク例外処理マスク状態であり，待ち禁止状態でない場合には，  
6410       対象タスクは待ち禁止状態になる【NGKI1333】．  
6411  
6412       dis\_waiにおいてtskidにTSK\_SELF（=0）を指定すると，自タスクが対象タスク  
6413       となる【NGKI1334】．  
6414  
6415       **【TOPPERS/ASPカーネルにおける規定】**  
6416  
6417       ASPカーネルでは，dis\_waiをサポートしない【ASPS0114】．  
6418  
6419       **【TOPPERS/FMPカーネルにおける規定】**  
6420  
6421       FMPカーネルでは，dis\_waiをサポートしない【FMPS0108】．  
6422  
6423       **【補足説明】**  
6424  
6425       dis\_waiは，対象タスクの待ち解除は行わない．対象タスクを待ち禁止状態にす  
6426       ることに加えて待ち解除したい場合には，dis\_waiを呼び出した後に，rel\_wai  
6427       を呼び出せばよい．  
6428  
6429       **【未決定事項】**  
6430  
6431       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  
6432       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，  
6433       現時点では未決定である．  
6434  
6435       **【μ ITRON4.0/PX仕様との関係】**  
6436  
6437       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．  
6438  
6439       ena\_wai       待ち禁止状態の解除〔TP〕   【NGKI1335】  
6440       iena\_wai      待ち禁止状態の解除〔IP〕   【NGKI1336】  
6441  
6442       **【C言語API】**  
6443           ER ercd = ena\_wai(ID tskid)  
6444           ER ercd = iena\_wai(ID tskid)  
6445  
6446       **【パラメータ】**  
6447           ID           tskid           対象タスクのID番号  
6448  
6449       **【リターンパラメータ】**  
6450           ER           ercd           正常終了（E\_OK）またはエラーコード

6451  
6452     **【エラーコード】**  
6453         E\_CTX         コンテキストエラー  
6454                     ・非タスクコンテキストからの呼出し (ena\_waiの場合) 【NGKI1337】  
6455                     ・タスクコンテキストからの呼出し (iena\_waiの場合) 【NGKI1338】  
6456                     ・CPUロック状態からの呼出し 【NGKI1339】  
6457         E\_NOSPT       未サポート機能  
6458                     ・対象タスクが制約タスク 【NGKI1340】  
6459         E\_ID          不正ID番号  
6460                     ・tskidが有効範囲外 【NGKI1341】  
6461         E\_NOEXS       オブジェクト未登録  
6462                     ・対象タスクが未登録 [D] 【NGKI1342】  
6463         E\_OACV        オブジェクトアクセス違反  
6464                     ・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  
6465                     の場合) 【NGKI1343】  
6466         E\_OBJ         オブジェクト状態エラー  
6467                     ・対象タスクが休止状態 【NGKI1344】  
6468                     ・対象タスクが待ち禁止状態でない 【NGKI1345】  
6469  
6470     **【機能】**  
6471  
6472     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  
6473     舞いは以下の通り.  
6474  
6475     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  
6476     【NGKI1346】.  
6477  
6478     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  
6479     となる 【NGKI1347】.  
6480  
6481     **【TOPPERS/ASPカーネルにおける規定】**  
6482  
6483     ASPカーネルでは, ena\_waiをサポートしない 【ASPS0115】.  
6484  
6485     **【TOPPERS/FMPカーネルにおける規定】**  
6486  
6487     FMPカーネルでは, ena\_waiをサポートしない 【FMPS0109】.  
6488  
6489     **【未決定事項】**  
6490  
6491     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  
6492     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,  
6493     現時点では未決定である.  
6494  
6495     **【 $\mu$  ITRON4.0/PX仕様との関係】**  
6496  
6497      $\mu$  ITRON4.0/PX仕様に定義されていないサービスコールである.  
6498     -----  
6499     dly\_tsk         自タスクの遅延 [T] 【NGKI1348】  
6500

6501       **【C言語API】**  
6502           ER ercd = dly\_tsk(RELTIM dlytim)  
6503  
6504       **【パラメータ】**  
6505           RELTIM           dlytim           遅延時間  
6506  
6507       **【リターンパラメータ】**  
6508           ER           ercd           正常終了 (E\_OK) またはエラーコード  
6509  
6510       **【エラーコード】**  
6511           E\_CTX           コンテキストエラー  
6512                            ・ディスパッチ保留状態からの呼出し【NGKI1349】  
6513           E\_NOSPT       未サポート機能  
6514                            ・制約タスクからの呼出し【NGKI1350】  
6515           E\_PAR           パラメータエラー  
6516                            ・dlytimがTMAX\_RELTIMより大きい【NGKI1351】  
6517           E\_RLWAI       待ち禁止状態または待ち状態の強制解除【NGKI1352】  
6518  
6519       **【機能】**  
6520  
6521       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。  
6522  
6523       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  
6524       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、  
6525       自タスクは待ち解除され、dly\_tskからE\_OKが返る【NGKI1354】。  
6526       -----  
6527  
6528       4.3 タスク例外処理機能  
6529  
6530       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  
6531       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  
6532       1つのみ登録できるため、タスクIDによって識別する【NGKI1355】。  
6533  
6534       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】。  
6535  
6536           ・タスク例外処理ルーチン属性  
6537           ・タスク例外処理禁止フラグ  
6538           ・保留例外要因  
6539           ・タスク例外処理ルーチンの先頭番地  
6540  
6541       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、  
6542       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  
6543       【NGKI1358】。  
6544  
6545       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  
6546       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  
6547       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  
6548       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  
6549       化される【NGKI1361】。  
6550

タスクの保留例外要因は、タスクに対して要求された例外要因を蓄積するためのビットマップであり、タスクの起動時に0に初期化される【NGKI1362】。

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

タスク例外処理ルーチンの実行が開始される時、タスク例外処理禁止フラグはセットされ、保留例外要因は0にクリアされる【NGKI1365】。また、タスク例外処理ルーチンからのリターン時には、タスク例外処理禁止フラグはクリアされる【NGKI1366】。

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

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

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

保護機能対応カーネルにおいて、タスク例外処理ルーチンは、タスクと同じ保護ドメインに属する【NGKI1373】。

タスク例外処理機能に用いるデータ型は次の通り。

TEXPTN      タスク例外要因のビットパターン（符号無し整数、uint\_tに定義）【NGKI1374】

C言語によるタスク例外処理ルーチンの記述形式は次の通り【NGKI1375】。

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

6601        }  
6602  
6603       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  
6604       の拡張情報が、それぞれ渡される【NGKI1376】。  
6605  
6606       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。  
6607  
6608       TBIT\_TEXPTN       タスク例外要因のビット数（TEXPTNの有効ビット数）  
6609                        【NGKI1377】  
6610  
6611       【補足説明】  
6612  
6613       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  
6614       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  
6615       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  
6616       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、  
6617       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  
6618       揃うことになる。  
6619  
6620       【TOPPERS/ASPカーネルにおける規定】  
6621  
6622       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  
6623       【ASPS0116】。  
6624  
6625       【TOPPERS/FMPカーネルにおける規定】  
6626  
6627       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  
6628       【FMPS0110】。  
6629  
6630       【TOPPERS/HRP2カーネルにおける規定】  
6631  
6632       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  
6633       【HRPS0110】。  
6634  
6635       【TOPPERS/SSPカーネルにおける規定】  
6636  
6637       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】。  
6638  
6639       【μ ITRON4.0仕様との関係】  
6640  
6641       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  
6642       行が開始されないという仕様に変更した。  
6643  
6644       【μ ITRON4.0/PX仕様との関係】  
6645  
6646       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  
6647       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。  
6648  
6649       【仕様変更の経緯】  
6650

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

---

DEF\_TEX      タスク例外処理ルーチンの定義 [S]    【NGKI1378】

def\_tex      タスク例外処理ルーチンの定義 [TD]    【NGKI1379】

#### 【静的API】

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

#### 【C言語API】

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

#### 【パラメータ】

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

\* タスク例外処理ルーチンの定義情報（パケットの内容）

ATR	texatr	タスク例外処理ルーチン属性
-----	--------	---------------

TEXRTN	texrtn	タスク例外処理ルーチンの先頭番地
--------	--------	------------------

#### 【リターンパラメータ】

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

#### 【エラーコード】

E_CTX	コンテキストエラー
	・非タスクコンテキストからの呼出し [s]    【NGKI1380】
	・CPUロック状態からの呼出し [s]    【NGKI1381】
E_ID	不正ID番号
	・tskidが有効範囲外 [s]    【NGKI1382】
E_RSATR	予約属性
	・texatrが無効 【NGKI1383】
	・その他の条件については機能の項を参照
E_PAR	パラメータエラー
	・texrtnがプログラムの先頭番地として正しくない 【NGKI1384】
E_NOEXS	オブジェクト未登録
	・対象タスクが未登録 【NGKI1385】
E_OACV	オブジェクトアクセス違反
	・対象タスクに対する管理操作が許可されていない [sP]    【NGKI1386】
E_MACV	メモリアクセス違反
	・pk_dtexが指すメモリ領域への読出しアクセスが許可されていない [sP]    【NGKI1387】
E_OBJ	オブジェクト状態エラー
	・対象タスクは静的APIで生成された [s]    【NGKI1388】

6701                   ・その他の条件については機能の項を参照  
6702  
6703       **【機能】**  
6704  
6705       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  
6706       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  
6707       **【NGKI1389】**。  
6708  
6709       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  
6710       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  
6711       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  
6712       **【NGKI1391】**。  
6713  
6714       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  
6715       タ、texrtnは一般定数式パラメータである**【NGKI1392】**。  
6716  
6717       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  
6718       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  
6719       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】**。  
6720  
6721       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  
6722       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  
6723       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  
6724       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  
6725       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  
6726       となる**【NGKI1396】**。ただし、TA\_DOM(TDOM\_SELF)を指定した場合には、指定が  
6727       無視され、E\_RSATRエラーは検出されない**【NGKI1397】**。  
6728  
6729       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  
6730       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  
6731       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  
6732       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  
6733       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  
6734       る**【NGKI1400】**。ただし、TA\_CLS(CLS\_SELF)を指定した場合には、指定が無視  
6735       され、E\_RSATRエラーは検出されない**【NGKI1401】**。  
6736  
6737       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  
6738       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  
6739       ていない場合には、E\_OBJエラーとなる**【NGKI1402】**。  
6740  
6741       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  
6742       となる**【NGKI1403】**。  
6743  
6744       **【TOPPERS/ASPカーネルにおける規定】**  
6745  
6746       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  
6747       成機能拡張パッケージでは、def\_texもサポートする**【ASPS0118】**。  
6748  
6749       **【TOPPERS/FMPカーネルにおける規定】**  
6750

6751 FMPカーネルでは、DEF\_TEXのみをサポートする【FMPS0111】.

6752

6753 **【TOPPERS/HRP2カーネルにおける規定】**

6754

6755 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

6756 成機能拡張パッケージでは、def\_texもサポートする【HRPS0179】.

6757

6758 **【 $\mu$  ITRON4.0仕様との関係】**

6759

6760 texrtnのデータ型をTEXTNに変更した.

6761

6762 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

6763 合に、E\_OBJエラーとすることにした.

6764 -----

6765 ras\_tex      タスク例外処理の要求 [T]   【NGKI1404】

6766 iras\_tex     タスク例外処理の要求 [I]   【NGKI1405】

6767

6768 **【C言語API】**

6769     ER ercd = ras\_tex(ID tskid, TEXPTN rasptn)

6770     ER ercd = iras\_tex(ID tskid, TEXPTN rasptn)

6771

6772 **【パラメータ】**

6773     ID           tskid           対象タスクのID番号

6774     TEXPTN       rasptn          要求するタスク例外処理のタスク例外要因

6775

6776 **【リターンパラメータ】**

6777     ER           ercd           正常終了 (E\_OK) またはエラーコード

6778

6779 **【エラーコード】**

6780     E\_CTX       コンテキストエラー

6781                ・非タスクコンテキストからの呼出し (ras\_texの場合)   【NGKI1406】

6782                ・タスクコンテキストからの呼出し (iras\_texの場合)  【NGKI1407】

6783                ・CPUロック状態からの呼出し   【NGKI1408】

6784     E\_ID       不正ID番号

6785                ・tskidが有効範囲外   【NGKI1409】

6786     E\_PAR       パラメータエラー

6787                ・rasptnが0   【NGKI1410】

6788     E\_NOEXS   オブジェクト未登録

6789                ・対象タスクが未登録 [D]   【NGKI1411】

6790     E\_OACV     オブジェクトアクセス違反

6791                ・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

6792                の場合) [P]   【NGKI1412】

6793     E\_OBJ       オブジェクト状態エラー

6794                ・対象タスクが休止状態   【NGKI1413】

6795                ・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

6796                ない   【NGKI1414】

6797

6798 **【機能】**

6799

6800 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



6801 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  
 6802 値とrasptnで指定した値のビット毎論理和 (C言語の" $\mid$ ") に更新される  
 6803 【NGKI1415】.  
 6804  
 6805 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  
 6806 となる 【NGKI1416】.  
 6807 -----  
 6808 dis\_tex      タスク例外処理の禁止 [T] 【NGKI1417】  
 6809  
 6810 【C言語API】  
 6811      ER ercd = dis\_tex()  
 6812  
 6813 【パラメータ】  
 6814      なし  
 6815  
 6816 【リターンパラメータ】  
 6817      ER              ercd              正常終了 (E\_OK) またはエラーコード  
 6818  
 6819 【エラーコード】  
 6820      E\_CTX           コンテキストエラー  
 6821                      ・非タスクコンテキストからの呼出し 【NGKI1419】  
 6822                      ・CPUロック状態からの呼出し 【NGKI1420】  
 6823      E\_OBJ           オブジェクト状態エラー  
 6824                      ・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  
 6825                      い 【NGKI1421】  
 6826  
 6827 【機能】  
 6828  
 6829 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,  
 6830 自タスクをタスク例外処理禁止状態に遷移させる.  
 6831 -----  
 6832 ena\_tex      タスク例外処理の許可 [T] 【NGKI1423】  
 6833  
 6834 【C言語API】  
 6835      ER ercd = ena\_tex()  
 6836  
 6837 【パラメータ】  
 6838      なし  
 6839  
 6840 【リターンパラメータ】  
 6841      ER              ercd              正常終了 (E\_OK) またはエラーコード  
 6842  
 6843 【エラーコード】  
 6844      E\_CTX           コンテキストエラー  
 6845                      ・非タスクコンテキストからの呼出し 【NGKI1424】  
 6846                      ・CPUロック状態からの呼出し 【NGKI1425】  
 6847      E\_OBJ           オブジェクト状態エラー  
 6848                      ・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  
 6849                      い 【NGKI1426】  
 6850

## 【機能】

自タスクのタスク例外処理禁止フラグをクリアする【NGKI1427】。すなわち、  
自タスクをタスク例外処理許可状態に遷移させる。

## 【補足説明】

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

-----  
`sns_tex`      タスク例外処理禁止状態の参照 [TI]   【NGKI1428】

## 【C言語API】

`bool_t state = sns_tex()`

## 【パラメータ】

なし

## 【リターンパラメータ】

`bool_t state`      タスク例外処理禁止状態

## 【機能】

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

実行状態のタスクが、タスク例外処理禁止状態の場合に`true`、タスク例外処理許可状態の場合に`false`が返る【NGKI1429】。`sns_tex`を非タスクコンテキストから呼び出した場合で、実行状態のタスクがない場合には、`true`が返る【NGKI1430】。

マルチプロセッサ対応カーネルにおいては、サービスコールを呼び出した処理単位を実行しているプロセッサにおいて実行状態のタスクのタスク例外処理禁止フラグを参照する【NGKI1431】。

## 【補足説明】

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

-----  
`ref_tex`      タスク例外処理の状態参照 [T]   【NGKI1432】

## 【C言語API】

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

## 【パラメータ】

<code>ID</code>	<code>tskid</code>	対象タスクのID番号
<code>T_RTEX *</code>	<code>pk_rtex</code>	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

6901  
 6902 **【リターンパラメータ】**  
 6903     ER            ercd            正常終了 (E\_OK) またはエラーコード  
 6904  
 6905     \* タスク例外処理の現在状態 (パケットの内容)  
 6906     STAT          texstat       タスク例外処理の状態  
 6907     TEXTN         pndptn       タスクの保留例外要因  
 6908  
 6909 **【エラーコード】**  
 6910     E\_CTX          コンテキストエラー  
 6911                    ・ 非タスクコンテキストからの呼出し 【NGKI1433】  
 6912                    ・ CPUロック状態からの呼出し 【NGKI1434】  
 6913     E\_ID          不正ID番号  
 6914                    ・ tskidが有効範囲外 【NGKI1435】  
 6915     E\_NOEXS       オブジェクト未登録  
 6916                    ・ 対象タスクが未登録 [D] 【NGKI1436】  
 6917     E\_OACV        オブジェクトアクセス違反  
 6918                    ・ 対象タスクに対する参照操作が許可されていない [P] 【NGKI1437】  
 6919     E\_MACV        メモリアクセス違反  
 6920                    ・ 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  
 6921                    いない [P] 【NGKI1438】  
 6922     E\_OBJ         オブジェクト状態エラー  
 6923                    ・ 対象タスクが休止状態 【NGKI1439】  
 6924                    ・ 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  
 6925                    ない 【NGKI1440】  
 6926  
 6927 **【機能】**  
 6928  
 6929     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  
 6930     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  
 6931     【NGKI1441】.  
 6932  
 6933     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  
 6934     かの値が返される 【NGKI1442】.  
 6935  
 6936     TTEX\_ENA      0x01U          タスク例外処理許可状態  
 6937     TTEX\_DIS      0x02U          タスク例外処理禁止状態  
 6938  
 6939     pndptnには, 対象タスクの現在の保留例外要因が返される 【NGKI1443】.  
 6940  
 6941     tskidにTSK\_SELF (=0) を指定すると, 自タスクが対象タスクとなる  
 6942     【NGKI1444】.  
 6943     -----  
 6944  
 6945     4.4 同期・通信機能  
 6946  
 6947     **【TOPPERS/SSPカーネルにおける規定】**  
 6948  
 6949     SSPカーネルでは, 同期・通信機能をサポートしない 【SSPS0127】.  
 6950

【 $\mu$  ITRON4.0仕様との関係】

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

## 4.4.1 セマフォ

セマフォは、資源の数を表す0以上の整数値を取るカウンタ（資源数）を介して、排他制御やイベント通知を行うための同期・通信オブジェクトである。セマフォの資源数から1を減ずることを資源の獲得、資源数に1を加えることを資源の返却と呼ぶ。セマフォは、セマフォIDと呼ぶID番号によって識別する【NGKI1445】。

各セマフォが持つ情報は次の通り【NGKI1446】。

- ・セマフォ属性
- ・資源数（の現在値）
- ・待ち行列（セマフォの資源獲得待ち状態のタスクのキュー）
- ・初期資源数
- ・最大資源数
- ・アクセス許可ベクタ（保護機能対応カーネルの場合）
- ・属する保護ドメイン（保護機能対応カーネルの場合）
- ・属するクラス（マルチプロセッサ対応カーネルの場合）

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

セマフォの初期資源数は、セマフォを生成または再初期化した際の、資源数の初期値である。また、セマフォの最大資源数は、資源数が取りうる最大値である。資源数が最大資源数に一致している時に資源を返却しようとする、E\_QOVRエラーとなる【NGKI1447】。

セマフォ属性には、次の属性を指定することができる【NGKI1448】。

TA\_TPRI      0x01U      待ち行列をタスクの優先度順にする

TA\_TPRIを指定しない場合、待ち行列はFIFO順になる【NGKI1449】。

セマフォ機能に関連するカーネル構成マクロは次の通り。

TMAX\_MAXSEM      セマフォの最大資源数の最大値（=UINT\_MAX）【NGKI1450】

TNUM\_SEMID      登録できるセマフォの数（動的生成対応でないカーネルでは、静的APIによって登録されたセマフォの数に一致）  
【NGKI1451】

【 $\mu$  ITRON4.0仕様との関係】

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

7001 CRE\_SEM セマフォの生成 [S] 【NGKI1452】  
 7002 acre\_sem セマフォの生成 [TD] 【NGKI1453】  
 7003  
 7004 **【静的API】**  
 7005 CRE\_SEM(ID semid, { ATR sematr, uint\_t isemcnt, uint\_t maxsem })  
 7006  
 7007 **【C言語API】**  
 7008 ER\_ID semid = acre\_sem(const T\_CSEM \*pk\_csem)  
 7009  
 7010 **【パラメータ】**  
 7011 ID semid 生成するセマフォのID番号 (CRE\_SEMの場合)  
 7012 T\_CSEM \* pk\_csem セマフォの生成情報を入れたパケットへのポイ  
 7013 ンタ (静的APIを除く)  
 7014  
 7015 \*セマフォの生成情報 (パケットの内容)  
 7016 ATR sematr セマフォ属性  
 7017 uint\_t isemcnt セマフォの初期資源数  
 7018 uint\_t maxsem セマフォの最大資源数  
 7019  
 7020 **【リターンパラメータ】**  
 7021 ER\_ID semid 生成されたセマフォのID番号 (正の値) または  
 7022 エラーコード  
 7023  
 7024 **【エラーコード】**  
 7025 E\_CTX コンテキストエラー  
 7026 ・非タスクコンテキストからの呼出し [s] 【NGKI1454】  
 7027 ・CPUロック状態からの呼出し [s] 【NGKI1455】  
 7028 E\_RSATR 予約属性  
 7029 ・sematrが無効 【NGKI1456】  
 7030 ・属する保護ドメインの指定が有効範囲外 [sP] 【NGKI1457】  
 7031 ・属するクラスの指定が有効範囲外 [sM] 【NGKI1458】  
 7032 ・クラスの囲みの中に記述されていない [SM] 【NGKI1459】  
 7033 E\_PAR パラメータエラー  
 7034 ・maxsemが有効範囲 (1以上TMAX\_MAXSEM以下) 外 【NGKI1468】  
 7035 ・isemcntが有効範囲 (0以上maxsem以下) 外 【NGKI1466】  
 7036 E\_OACV オブジェクトアクセス違反  
 7037 ・システム状態に対する管理操作が許可されていない [sP]  
 7038 【NGKI1460】  
 7039 E\_MACV メモリアクセス違反  
 7040 ・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  
 7041 いない [sP] 【NGKI1461】  
 7042 E\_NOID ID番号不足  
 7043 ・割り付けられるセマフォIDがない [sD] 【NGKI1462】  
 7044 E\_OBJ オブジェクト状態エラー  
 7045 ・semidで指定したセマフォが登録済み (CRE\_SEMの場合)  
 7046 【NGKI1463】  
 7047  
 7048 **【機能】**  
 7049  
 7050 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

7051 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  
7052 る【NGKI1464】。

7053

7054 静的APIにおいては、semidはオブジェクト識別名、sematr, isemcnt, maxsemは  
7055 整数定数式パラメータである【NGKI1465】。

7056

7057 【TOPPERS/ASPカーネルにおける規定】

7058

7059 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  
7060 成機能拡張パッケージでは、acre\_semもサポートする【ASPS0120】。

7061

7062 【TOPPERS/FMPカーネルにおける規定】

7063

7064 FMPカーネルでは、CRE\_SEMのみをサポートする【FMPS0112】。

7065

7066 【TOPPERS/HRP2カーネルにおける規定】

7067

7068 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  
7069 成機能拡張パッケージでは、acre\_semもサポートする【HRPS0180】。

7070 -----

7071 AID\_SEM 割付け可能なセマフォIDの数の指定〔SD〕【NGKI1469】

7072

7073 【静的API】

7074 AID\_SEM(uint\_t nosem)

7075

7076 【パラメータ】

7077 uint\_t nosem 割付け可能なセマフォIDの数

7078

7079 【エラーコード】

7080 E\_RSATR 予約属性

7081 ・保護ドメインの囲みの中に記述されている〔P〕【NGKI3429】

7082 ・クラスの囲みの中に記述されていない〔M〕【NGKI1470】

7083 E\_PAR パラメータエラー

7084 ・nosemが負の値【NGKI3277】

7085

7086 【機能】

7087

7088 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  
7089 て割付け可能なセマフォIDとして確保する【NGKI1471】。

7090

7091 nosemは整数定数式パラメータである【NGKI1472】。

7092

7093 【TOPPERS/ASPカーネルにおける規定】

7094

7095 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  
7096 【ASPS0211】。

7097

7098 【TOPPERS/HRP2カーネルにおける規定】

7099

7100 HRP2カーネルの動的生成機能拡張パッケージでは、AID\_SEMをサポートする

7101      **【HRPS0212】** .

7103 SAC\_SEM セマフォのアクセス許可ベクタの設定 [SP] 【NGKI1473】

7105

## 7106 【静的API】

```
7107     SAC_SEM(ID semid, { ACPTN acptn1, ACPTN acptn2,  
7108                        ACPTN acptn3, ACPTN acptn4 } )
```

## 7110 【C言語API】

```
7111      ER ercd = sac_sem(ID semid, const ACVCT *p_acvct)
```

## 7113 【パラメータ】

7114	ID	semid	対象セマフォのID番号
7115	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）
7116			

7118      \*アクセス許可ベクタ (パケットの内容)

7119	ACPTN	acptn1	通常操作1のアクセス許可パターン
------	-------	--------	------------------

7121	ACPTN	acptn3	管理操作のアクセス許可パターン
------	-------	--------	-----------------

7123

## 7125 ER ercd

7126

## 7128 E CTX

7129	E_CTR	・非タスクコンテキストからの呼出し [s] 【NGKI1475】
7130		・CPUロック状態からの呼出し [s] 【NGKI1476】
7131	E_ID	不正ID番号
7132		・semidが有効範囲外 [s] 【NGKI1477】
7133	E_RSATR	予約属性
7134		・対象セマフォが属する保護ドメインの囲みの中に記述され
7135		ていない [S] 【NGKI1478】
7136		・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
7137		ない [SM] 【NGKI1479】

```
7138      E_NOEXS      オブジェクト未登録
```

7139 ・対象セマフォが未登録【NGKI1480】

7140 E OACV オブジェクトアクセス違反

7141                   ・対象セマフォに対する管理操作が許可されていない [s]

7142 【NGK11481】

7143	E MACV	メモリアクセス違反
------	--------	-----------

```
7144      ・ 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
```

7145 いたい [s] 【NGK11482】

7146	F OBI	オブジェクト状態エラー
------	-------	-------------

7147 ・対象メモリは静的APIで生成された [s] 【NGKI1483】

7148 ・対象オブジェクトに対してアクセス許可ベクタが設定済み。[S]

7149 【NCKI1484】

7151      **【機能】**

7152

7153      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  
7154      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

7155      **【NGKI1485】** .

7156

7157      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  
7158      式パラメータである **【NGKI1486】** .

7159

7160      **【TOPPERS/HRP2カーネルにおける規定】**

7161

7162      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  
7163      成機能拡張パッケージでは、sac\_semもサポートする **【HRPS0181】** .

7164

7165      del\_sem      セマフォの削除 [TD]      **【NGKI1487】**

7166

7167      **【C言語API】**

7168           ER ercd = del\_sem(ID semid)

7169

7170      **【パラメータ】**

7171      ID              semid              対象セマフォのID番号

7172

7173      **【リターンパラメータ】**

7174      ER              ercd              正常終了 (E\_OK) またはエラーコード

7175

7176      **【エラーコード】**

7177      E\_CTX              コンテキストエラー

7178                      ・非タスクコンテキストからの呼出し **【NGKI1488】**

7179                      ・CPUロック状態からの呼出し **【NGKI1489】**

7180      E\_ID              不正ID番号

7181                      ・semidが有効範囲外 **【NGKI1490】**

7182      E\_NOEXS          オブジェクト未登録

7183                      ・対象セマフォが未登録 **【NGKI1491】**

7184      E\_OACV              オブジェクトアクセス違反

7185                      ・対象セマフォに対する管理操作が許可されていない [P]

7186                      **【NGKI1492】**

7187      E\_OBJ              オブジェクト状態エラー

7188                      ・対象セマフォは静的APIで生成された **【NGKI1493】**

7189

7190      **【機能】**

7191

7192      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  
7193      の通り．

7194

7195      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

7196      **【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

7197      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

7198      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】** .

7199

7200      **【使用上の注意】**



7201  
7202 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  
7203 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  
7204 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  
7205 み禁止時間が長くなるため、注意が必要である。

7206  
7207 **【TOPPERS/ASPカーネルにおける規定】**  
7208  
7209 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  
7210 機能拡張パッケージでは、del\_semをサポートする【ASPS0123】。

7211  
7212 **【TOPPERS/FMPカーネルにおける規定】**  
7213  
7214 FMPカーネルでは、del\_semをサポートしない【FMPS0114】。

7215  
7216 **【TOPPERS/HRP2カーネルにおける規定】**  
7217  
7218 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  
7219 機能拡張パッケージでは、del\_semをサポートする【HRPS0182】。

7220 -----

7221 sig\_sem セマフォの資源の返却 [T] 【NGKI1497】  
7222 isig\_sem セマフォの資源の返却 [I] 【NGKI1498】  
7223

7224 **【C言語API】**  
7225 ER ercd = sig\_sem(ID semid)  
7226 ER ercd = isig\_sem(ID semid)  
7227

7228 **【パラメータ】**  
7229 ID semid 対象セマフォのID番号  
7230

7231 **【リターンパラメータ】**  
7232 ER ercd 正常終了 (E\_OK) またはエラーコード  
7233

7234 **【エラーコード】**  
7235 E\_CTX コンテキストエラー  
7236 ・非タスクコンテキストからの呼出し (sig\_semの場合) 【NGKI1499】  
7237 ・タスクコンテキストからの呼出し (isig\_semの場合) 【NGKI1500】  
7238 ・CPUロック状態からの呼出し 【NGKI1501】  
7239 E\_ID 不正ID番号  
7240 ・semidが有効範囲外 【NGKI1502】  
7241 E\_NOEXS オブジェクト未登録  
7242 ・対象セマフォが未登録 [D] 【NGKI1503】  
7243 E\_OACV オブジェクトアクセス違反  
7244 ・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  
7245 の場合) [P] 【NGKI1504】  
7246 E\_QOVR キューイングオーバーフロー  
7247 ・条件については機能の項を参照  
7248

7249 **【機能】**  
7250

7251 semidで指定したセマフォ（対象セマフォ）に資源を返却する．具体的な振舞い  
 7252 は以下の通り．  
 7253  
 7254 対象セマフォの待ち行列にタスクが存在する場合には，待ち行列の先頭のタス  
 7255 クが待ち解除される【NGKI1505】．この時，待ち解除されたタスクが資源を獲  
 7256 得したことになるため，対象セマフォの資源数は変化しない【NGKI1506】．待  
 7257 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  
 7258 【NGKI1507】．  
 7259  
 7260 待ち行列にタスクが存在しない場合には，対象セマフォの資源数に1が加えられ  
 7261 る【NGKI1508】．資源数に1を加えるとそのセマフォの最大資源数を越える場合  
 7262 には，E\_QOVRエラーとなる【NGKI1509】．  
 7263 -----  
 7264 wai\_sem セマフォの資源の獲得 [T] 【NGKI1510】  
 7265 pol\_sem セマフォの資源の獲得（ポーリング） [T] 【NGKI1511】  
 7266 twai\_sem セマフォの資源の獲得（タイムアウト付き） [T] 【NGKI1512】  
 7267  
 7268 【C言語API】  
 7269 ER ercd = wai\_sem(ID semid)  
 7270 ER ercd = pol\_sem(ID semid)  
 7271 ER ercd = twai\_sem(ID semid, TMO tmout)  
 7272  
 7273 【パラメータ】  
 7274 ID semid 対象セマフォのID番号  
 7275 TMO tmout タイムアウト時間（twai\_semの場合）  
 7276  
 7277 【リターンパラメータ】  
 7278 ER ercd 正常終了（E\_OK）またはエラーコード  
 7279  
 7280 【エラーコード】  
 7281 E\_CTX コンテキストエラー  
 7282 ・非タスクコンテキストからの呼出し【NGKI1513】  
 7283 ・CPUロック状態からの呼出し【NGKI1514】  
 7284 ・ディスパッチ保留状態からの呼出し（pol\_semを除く）【NGKI1515】  
 7285 E\_NOSPT 未サポート機能  
 7286 ・制約タスクからの呼出し（pol\_semを除く）【NGKI1516】  
 7287 E\_ID 不正ID番号  
 7288 ・semidが有効範囲外【NGKI1517】  
 7289 E\_PAR パラメータエラー  
 7290 ・tmoutが無効（twai\_semの場合）【NGKI1518】  
 7291 E\_NOEXS オブジェクト未登録  
 7292 ・対象セマフォが未登録 [D] 【NGKI1519】  
 7293 E\_OACV オブジェクトアクセス違反  
 7294 ・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]  
 7295 【NGKI1520】  
 7296 E\_TMOUT ポーリング失敗またはタイムアウト（wai\_semを除く）【NGKI1521】  
 7297 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）  
 7298 【NGKI1522】  
 7299 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）  
 7300 【NGKI1523】

7301

7302     **【機能】**

7303

7304     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  
7305     いは以下の通り．

7306

7307     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

7308     **【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7309     となり，対象セマフォの待ち行列につながる **【NGKI1525】**．

7310

7311     -----  
7311     ini\_sem     セマフォの再初期化 [T]     **【NGKI1526】**

7312

7313     **【C言語API】**

7314     ER ercd = ini\_sem(ID semid)

7315

7316     **【パラメータ】**

7317     ID            semid            対象セマフォのID番号

7318

7319     **【リターンパラメータ】**

7320     ER            ercd            正常終了 (E\_OK) またはエラーコード

7321

7322     **【エラーコード】**

7323     E\_CTX         コンテキストエラー

7324                    ・非タスクコンテキストからの呼出し **【NGKI1527】**7325                    ・CPUロック状態からの呼出し **【NGKI1528】**

7326     E\_ID          不正ID番号

7327                    ・semidが有効範囲外 **【NGKI1529】**

7328     E\_NOEXS       オブジェクト未登録

7329                    ・対象セマフォが未登録 [D]     **【NGKI1530】**

7330     E\_OACV        オブジェクトアクセス違反

7331                    ・対象セマフォに対する管理操作が許可されていない [P]

7332                                 **【NGKI1531】**

7333

7334     **【機能】**

7335

7336     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  
7337     以下の通り．

7338

7339     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

7340     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

7341     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7342     たサービスコールからE\_DLTエラーが返る **【NGKI1534】**．

7343

7344     **【使用上の注意】**

7345

7346     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

7347     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

7348     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

7349     み禁止時間が長くなるため，注意が必要である．

7350

7351 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  
7352 プリケーションの責任である。

7353

7354 **【 $\mu$  ITRON4.0仕様との関係】**

7355

7356  $\mu$  ITRON4.0仕様に定義されていないサービスコールである。

7357 -----

7358 ref\_sem セマフォの状態参照 [T] **【NGKI1535】**

7359

7360 **【C言語API】**

7361 ER ercd = ref\_sem(ID semid, T\_RSEM \*pk\_rsem)

7362

7363 **【パラメータ】**

7364 ID semid 対象セマフォのID番号

7365 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  
7366 ンタ

7367

7368 **【リターンパラメータ】**

7369 ER ercd 正常終了 (E\_OK) またはエラーコード

7370

7371 \*セマフォの現在状態 (パケットの内容)

7372 ID wtskid セマフォの待ち行列の先頭のタスクのID番号

7373 uint\_t semcnt セマフォの資源数

7374

7375 **【エラーコード】**

7376 E\_CTX コンテキストエラー

7377 ・非タスクコンテキストからの呼出し **【NGKI1536】**

7378 ・CPUロック状態からの呼出し **【NGKI1537】**

7379 E\_ID 不正ID番号

7380 ・semidが有効範囲外 **【NGKI1538】**

7381 E\_NOEXS オブジェクト未登録

7382 ・対象セマフォが未登録 [D] **【NGKI1539】**

7383 E\_OACV オブジェクトアクセス違反

7384 ・対象セマフォに対する参照操作が許可されていない [P]

7385 **【NGKI1540】**

7386 E\_MACV メモリアクセス違反

7387 ・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  
7388 いない [P] **【NGKI1541】**

7389

7390 **【機能】**

7391

7392 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  
7393 在状態は、pk\_rsemで指定したパケットに返される **【NGKI1542】**。

7394

7395 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 (=   
7396 0) が返る **【NGKI1543】**。

7397

7398 **【使用上の注意】**

7399

7400 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

7401 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  
7402 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  
7403 化している可能性があるためである。

7404 -----

7405

#### 7406 4.4.2 イベントフラグ

7407

7408 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  
7409 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  
7410 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  
7411 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、  
7412 イベントフラグIDと呼ぶID番号によって識別する【NGKI1545】。

7413

7414 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  
7415 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  
7416 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、  
7417 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。

7418

7419 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  
7420 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  
7421 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  
7422 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  
7423 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  
7424 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  
7425 待ち解除の条件と呼ぶ。

7426

7427 各イベントフラグが持つ情報は次の通り【NGKI1548】。

7428

- 7429 ・ イベントフラグ属性
- 7430 ・ ビットパターン（の現在値）
- 7431 ・ 待ち行列（イベントフラグ待ち状態のタスクのキュー）
- 7432 ・ 初期ビットパターン
- 7433 ・ アクセス許可ベクタ（保護機能対応カーネルの場合）
- 7434 ・ 属する保護ドメイン（保護機能対応カーネルの場合）
- 7435 ・ 属するクラス（マルチプロセッサ対応カーネルの場合）

7436

7437 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  
7438 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。  
7439 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、  
7440 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）  
7441 【NGKI1549】。

7442

7443 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  
7444 化した際の、ビットパターンの初期値である。

7445

7446 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】。

7447

7448	TA_TPRI	0x01U	待ち行列をタスクの優先度順にする
7449	TA_WMUL	0x02U	複数のタスクが待つのを許す
7450	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

7451  
7452 TA\_TPRIを指定しない場合、待ち行列はFIFO順になる【NGKI1551】. TA\_WMULを  
7453 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  
7454 【NGKI1552】.  
7455  
7456 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  
7457 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  
7458 時にイベントフラグをクリアしない【NGKI1554】.  
7459  
7460 イベントフラグ機能に用いるデータ型は次の通り.  
7461  
7462       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  
7463                      定義）【NGKI1555】  
7464  
7465 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.  
7466  
7467       TBIT\_FLGPTN   イベントフラグのビット数（FLGPTNの有効ビット数）  
7468                      【NGKI1556】  
7469  
7470       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  
7471                      ネルでは、静的APIによって登録されたイベントフラグの  
7472                      数に一致）【NGKI1557】  
7473  
7474       【TOPPERS/ASPカーネルにおける規定】  
7475  
7476 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  
7477 【ASPS0124】.  
7478  
7479       【TOPPERS/FMPカーネルにおける規定】  
7480  
7481 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  
7482 【FMPS0115】.  
7483  
7484       【TOPPERS/HRP2カーネルにおける規定】  
7485  
7486 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  
7487 【HRPS0115】.  
7488  
7489       【 $\mu$  ITRON4.0仕様との関係】  
7490  
7491 TNUM\_FLGIDは、 $\mu$  ITRON4.0仕様に規定されていないカーネル構成マクロである.  
7492 -----  
7493 CRE\_FLG       イベントフラグの生成 [S]   【NGKI1558】  
7494 acre\_flg       イベントフラグの生成 [TD]   【NGKI1559】  
7495  
7496       【静的API】  
7497       CRE\_FLG(ID flgid, { ATR flgatr, FLGPTN iflgptn })  
7498  
7499       【C言語API】  
7500       ER\_ID flgid = acre\_flg(const T\_CFLG \*pk\_cflg)

7501

7502 **【パラメータ】**

7503	ID	flgid	生成するイベントフラグのID番号 (CRE_FLGの 7504 場合)
7505	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 7506 のポインタ (静的APIを除く)

7507

7508 \* イベントフラグの生成情報 (パケットの内容)

7509	ATR	flgatr	イベントフラグ属性
7510	FLGPTN	iflgptn	イベントフラグの初期ビットパターン

7511

7512 **【リターンパラメータ】**

7513	ER_ID	flgid	生成されたイベントフラグのID番号 (正の値) 7514 またはエラーコード
------	-------	-------	-------------------------------------------

7515

7516 **【エラーコード】**

7517	E_CTX	コンテキストエラー
7518		・非タスクコンテキストからの呼出し [s] 【NGKI1560】
7519		・CPUロック状態からの呼出し [s] 【NGKI1561】
7520	E_RSATR	予約属性
7521		・flgatrが無効 【NGKI1562】
7522		・属する保護ドメインの指定が有効範囲外 [sP] 【NGKI1563】
7523		・属するクラスの指定が有効範囲外 [sM] 【NGKI1564】
7524		・クラスの囲みの中に記述されていない [SM] 【NGKI1565】
7525	E_PAR	パラメータエラー
7526		・iflgptnがFLGPTNに格納できない [S] 【NGKI3275】
7527	E_OACV	オブジェクトアクセス違反
7528		・システム状態に対する管理操作が許可されていない [sP] 7529 【NGKI1566】
7530	E_MACV	メモリアクセス違反
7531		・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 7532 いない [sP] 【NGKI1567】
7533	E_NOID	ID番号不足
7534		・割り付けられるイベントフラグIDがない [sD] 【NGKI1568】
7535	E_OBJ	オブジェクト状態エラー
7536		・flgidで指定したイベントフラグが登録済み (CRE_FLGの場合) 7537 【NGKI1569】

7538

7539 **【機能】**

7540

7541 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  
7542 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  
7543 に、待ち行列は空の状態に初期化される 【NGKI1570】。

7544

7545 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  
7546 式パラメータである 【NGKI1571】。

7547

7548 **【TOPPERS/ASPカーネルにおける規定】**

7549

7550 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

152



```

7601  【C言語API】
7602      ER ercd = sac_flg(ID flgid, const ACVCT *p_acvct)
7603
7604  【パラメータ】
7605      ID          flgid      対象イベントフラグのID番号
7606      ACVCT *     p_acvct    アクセス許可ベクタを入れたパケットへのポ
7607                          インタ（静的APIを除く）
7608
7609      *アクセス許可ベクタ（パケットの内容）
7610      ACPTN      acptn1      通常操作1のアクセス許可パターン
7611      ACPTN      acptn2      通常操作2のアクセス許可パターン
7612      ACPTN      acptn3      管理操作のアクセス許可パターン
7613      ACPTN      acptn4      参照操作のアクセス許可パターン
7614
7615  【リターンパラメータ】
7616      ER          ercd      正常終了（E_OK）またはエラーコード
7617
7618  【エラーコード】
7619      E_CTX      コンテキストエラー
7620                  ・非タスクコンテキストからの呼出し [s] 【NGKI1578】
7621                  ・CPUロック状態からの呼出し [s] 【NGKI1579】
7622      E_ID      不正ID番号
7623                  ・flgidが有効範囲外 [s] 【NGKI1580】
7624      E_RSATR    予約属性
7625                  ・対象イベントフラグが属する保護ドメインの囲みの中に記
7626                  述されていない [S] 【NGKI1581】
7627                  ・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
7628                  ていない [SM] 【NGKI1582】
7629      E_NOEXS    オブジェクト未登録
7630                  ・対象イベントフラグが未登録 【NGKI1583】
7631      E_OACV     オブジェクトアクセス違反
7632                  ・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]
7633                  【NGKI1584】
7634      E_MACV     メモリアクセス違反
7635                  ・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
7636                  いない [s] 【NGKI1585】
7637      E_OBJ      オブジェクト状態エラー
7638                  ・対象イベントフラグは静的APIで生成された [s] 【NGKI1586】
7639                  ・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
7640                  み [S] 【NGKI1587】
7641
7642  【機能】
7643
7644      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
7645      （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
7646      【NGKI1588】 .
7647
7648      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
7649      式パラメータである 【NGKI1589】 .
7650

```

7651       【TOPPERS/HRP2カーネルにおける規定】  
7652  
7653       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  
7654       成機能拡張パッケージでは、sac\_flgもサポートする【HRPS0184】。  
7655       -----  
7656       del\_flg       イベントフラグの削除 [TD]   【NGKI1590】  
7657  
7658       【C言語API】  
7659       ER ercd = del\_flg(ID flgid)  
7660  
7661       【パラメータ】  
7662       ID           flgid           対象イベントフラグのID番号  
7663  
7664       【リターンパラメータ】  
7665       ER           ercd           正常終了 (E\_OK) またはエラーコード  
7666  
7667       【エラーコード】  
7668       E\_CTX       コンテキストエラー  
7669                   ・非タスクコンテキストからの呼出し【NGKI1591】  
7670                   ・CPUロック状態からの呼出し【NGKI1592】  
7671       E\_ID       不正ID番号  
7672                   ・flgidが有効範囲外【NGKI1593】  
7673       E\_NOEXS   オブジェクト未登録  
7674                   ・対象イベントフラグが未登録【NGKI1594】  
7675       E\_OACV   オブジェクトアクセス違反  
7676                   ・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]  
7677                   【NGKI1595】  
7678       E\_OBJ     オブジェクト状態エラー  
7679                   ・対象イベントフラグは静的APIで生成された【NGKI1596】  
7680  
7681       【機能】  
7682  
7683       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  
7684       振舞いは以下の通り。  
7685  
7686       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  
7687       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  
7688       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  
7689       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  
7690       返る【NGKI1599】。  
7691  
7692       【使用上の注意】  
7693  
7694       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  
7695       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  
7696       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  
7697       み禁止時間が長くなるため、注意が必要である。  
7698  
7699       【TOPPERS/ASPカーネルにおける規定】  
7700

7701 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  
7702 機能拡張パッケージでは、del\_flgをサポートする【ASPS0129】。

7703

7704 【TOPPERS/FMPカーネルにおける規定】

7705

7706 FMPカーネルでは、del\_flgをサポートしない【FMPS0118】。

7707

7708 【TOPPERS/HRP2カーネルにおける規定】

7709

7710 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  
7711 機能拡張パッケージでは、del\_flgをサポートする【HRPS0185】。

7712

7713 set\_flg イベントフラグのセット [T] 【NGKI1600】

7714 iset\_flg イベントフラグのセット [I] 【NGKI1601】

7715

7716 【C言語API】

7717 ER ercd = set\_flg(ID flgid, FLGPTN setptn)

7718 ER ercd = iset\_flg(ID flgid, FLGPTN setptn)

7719

7720 【パラメータ】

7721 ID flgid 対象イベントフラグのID番号

7722 FLGPTN setptn セットするビットパターン

7723

7724 【リターンパラメータ】

7725 ER ercd 正常終了 (E\_OK) またはエラーコード

7726

7727 【エラーコード】

7728 E\_CTX コンテキストエラー

7729 ・非タスクコンテキストからの呼出し (set\_flgの場合) 【NGKI1602】

7730 ・タスクコンテキストからの呼出し (iset\_flgの場合) 【NGKI1603】

7731 ・CPUロック状態からの呼出し 【NGKI1604】

7732 E\_ID 不正ID番号

7733 ・flgidが有効範囲外 【NGKI1605】

7734 E\_NOEXS オブジェクト未登録

7735 ・対象イベントフラグが未登録 [D] 【NGKI1606】

7736 E\_OACV オブジェクトアクセス違反

7737 ・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

7738 (set\_flgの場合) [P] 【NGKI1607】

7739

7740 【機能】

7741

7742 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  
7743 をセットする。具体的な振舞いは以下の通り。

7744

7745 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

7746 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

7747 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

7748 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】。

7749 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

7750 【NGKI1610】。

7751  
 7752 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  
 7753 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  
 7754 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。  
 7755  
 7756 **【使用上の注意】**  
 7757  
 7758 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  
 7759 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、  
 7760 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  
 7761 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  
 7762 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。  
 7763 -----  
 7764 clr\_flg イベントフラグのクリア [T] **【NGKI1611】**  
 7765  
 7766 **【C言語API】**  
 7767 ER ercd = clr\_flg(ID flgid, FLGPTN clrptn)  
 7768  
 7769 **【パラメータ】**  
 7770 ID flgid 対象イベントフラグのID番号  
 7771 FLGPTN clrptn クリアするビットパターン（クリアしないビッ  
 7772 トを1, クリアするビットを0とする）  
 7773  
 7774 **【リターンパラメータ】**  
 7775 ER ercd 正常終了 (E\_OK) またはエラーコード  
 7776  
 7777 **【エラーコード】**  
 7778 E\_CTX コンテキストエラー  
 7779 ・非タスクコンテキストからの呼出し **【NGKI1612】**  
 7780 ・CPUロック状態からの呼出し **【NGKI1613】**  
 7781 E\_ID 不正ID番号  
 7782 ・flgidが有効範囲外 **【NGKI1614】**  
 7783 E\_NOEXS オブジェクト未登録  
 7784 ・対象イベントフラグが未登録 [D] **【NGKI1615】**  
 7785 E\_OACV オブジェクトアクセス違反  
 7786 ・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]  
 7787 **【NGKI1616】**  
 7788  
 7789 **【機能】**  
 7790  
 7791 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  
 7792 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  
 7793 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  
 7794 **【NGKI1617】**。  
 7795 -----  
 7796 wai\_flg イベントフラグ待ち [T] **【NGKI1618】**  
 7797 pol\_flg イベントフラグ待ち（ポーリング） [T] **【NGKI1619】**  
 7798 twai\_flg イベントフラグ待ち（タイムアウト付き） [T] **【NGKI1620】**  
 7799  
 7800 **【C言語API】**

```

7801     ER ercd = wai_flg(ID flgid, FLGPTN waiptn, MODE wfmode, FLGPTN *p_flgptn)
7802     ER ercd = pol_flg(ID flgid, FLGPTN waiptn, MODE wfmode, FLGPTN *p_flgptn)
7803     ER ercd = twai_flg(ID flgid, FLGPTN waiptn,
7804                       MODE wfmode, FLGPTN *p_flgptn, TMO tmout)
7805

```

#### 【パラメータ】

```

7807     ID          flgid      対象イベントフラグのID番号
7808     FLGPTN      waiptn     待ちビットパターン
7809     MODE        wfmode     待ちモード
7810     FLGPTN *    p_flgptn   待ち解除時のビットパターンを入れるメモリ領
7811                               域へのポインタ
7812     TMO         tmout      タイムアウト時間 (twai_flgの場合)
7813

```

#### 【リターンパラメータ】

```

7815     ER          ercd      正常終了 (E_OK) またはエラーコード
7816     FLGPTN      flgptn    待ち解除時のビットパターン
7817

```

#### 【エラーコード】

```

7819     E_CTX       コンテキストエラー
7820                 ・非タスクコンテキストからの呼出し 【NGKI1621】
7821                 ・CPUロック状態からの呼出し 【NGKI1622】
7822                 ・ディスパッチ保留状態からの呼出し (pol_flgを除く) 【NGKI1623】
7823     E_NOSPT      未サポート機能
7824                 ・制約タスクからの呼出し (pol_flgを除く) 【NGKI1624】
7825     E_ID         不正ID番号
7826                 ・flgidが有効範囲外 【NGKI1625】
7827     E_PAR        パラメータエラー
7828                 ・waiptnが0 【NGKI1626】
7829                 ・wfmodeが無効 (TWF_ORWまたはTWF_ANDWでない) 【NGKI1627】
7830                 ・tmoutが無効 (twai_flgの場合) 【NGKI1628】
7831     E_NOEXS      オブジェクト未登録
7832                 ・対象イベントフラグが未登録 [D] 【NGKI1629】
7833     E_OACV       オブジェクトアクセス違反
7834                 ・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]
7835                 【NGKI1630】
7836     E_MACV       メモリアクセス違反
7837                 ・p_flgptnが指すメモリ領域への書込みアクセスが許可され
7838                 ていない [P] 【NGKI1631】
7839     E_ILUSE      サービスコール不正使用
7840                 ・TA_WMUL属性でないイベントフラグで待ちタスクあり 【NGKI1632】
7841     E_TMOUT      ポーリング失敗またはタイムアウト (wai_flgを除く) 【NGKI1633】
7842     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)
7843                 【NGKI1634】
7844     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)
7845                 【NGKI1635】
7846

```

#### 【機能】

```

7849     flgidで指定したイベントフラグ (対象イベントフラグ) が, waiptnとwfmodeで
7850     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.

```

7851  
 7852 対象イベントフラグが、waitpnとwfmodeで指定した待ち解除の条件を満たして  
 7853 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  
 7854 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  
 7855 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  
 7856 【NGKI1637】。  
 7857  
 7858 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  
 7859 態となり、対象イベントフラグの待ち行列につながる【NGKI1638】。  
 7860 -----  
 7861 ini\_flg イベントフラグの再初期化 [T] 【NGKI1639】  
 7862  
 7863 【C言語API】  
 7864 ER ercd = ini\_flg(ID flgid)  
 7865  
 7866 【パラメータ】  
 7867 ID flgid 対象イベントフラグのID番号  
 7868  
 7869 【リターンパラメータ】  
 7870 ER ercd 正常終了 (E\_OK) またはエラーコード  
 7871  
 7872 【エラーコード】  
 7873 E\_CTX コンテキストエラー  
 7874 ・非タスクコンテキストからの呼出し【NGKI1640】  
 7875 ・CPUロック状態からの呼出し【NGKI1641】  
 7876 E\_ID 不正ID番号  
 7877 ・flgidが有効範囲外【NGKI1642】  
 7878 E\_NOEXS オブジェクト未登録  
 7879 ・対象イベントフラグが未登録 [D] 【NGKI1643】  
 7880 E\_OACV オブジェクトアクセス違反  
 7881 ・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]  
 7882 【NGKI1644】  
 7883  
 7884 【機能】  
 7885  
 7886 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  
 7887 的な振舞いは以下の通り。  
 7888  
 7889 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  
 7890 【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、  
 7891 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  
 7892 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  
 7893 【NGKI1647】。  
 7894  
 7895 【使用上の注意】  
 7896  
 7897 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  
 7898 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  
 7899 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  
 7900 み禁止時間が長くなるため、注意が必要である。

7901  
 7902 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  
 7903 は、アプリケーションの責任である。  
 7904  
 7905 **【μITRON4.0仕様との関係】**  
 7906  
 7907 μITRON4.0仕様に定義されていないサービスコールである。  
 7908 -----  
 7909 ref\_flg イベントフラグの状態参照 [T] **【NGKI1648】**  
 7910  
 7911 **【C言語API】**  
 7912 ER ercd = ref\_flg(ID flgid, T\_RFLG \*pk\_rflg)  
 7913  
 7914 **【パラメータ】**  
 7915 ID flgid 対象イベントフラグのID番号  
 7916 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  
 7917 のポインタ  
 7918  
 7919 **【リターンパラメータ】**  
 7920 ER ercd 正常終了 (E\_OK) またはエラーコード  
 7921  
 7922 \* イベントフラグの現在状態 (パケットの内容)  
 7923 ID wtskid イベントフラグの待ち行列の先頭のタスクのID  
 7924 番号  
 7925 uint\_t flgptn イベントフラグのビットパターン  
 7926  
 7927 **【エラーコード】**  
 7928 E\_CTX コンテキストエラー  
 7929 ・非タスクコンテキストからの呼出し **【NGKI1649】**  
 7930 ・CPUロック状態からの呼出し **【NGKI1650】**  
 7931 E\_ID 不正ID番号  
 7932 ・flgidが有効範囲外 **【NGKI1651】**  
 7933 E\_NOEXS オブジェクト未登録  
 7934 ・対象イベントフラグが未登録 [D] **【NGKI1652】**  
 7935 E\_OACV オブジェクトアクセス違反  
 7936 ・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  
 7937 [P] **【NGKI1653】**  
 7938 E\_MACV メモリアクセス違反  
 7939 ・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  
 7940 いない [P] **【NGKI1654】**  
 7941  
 7942 **【機能】**  
 7943  
 7944 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。  
 7945 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】**。  
 7946  
 7947 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  
 7948 TSK\_NONE (=0) が返る **【NGKI1656】**。  
 7949  
 7950 **【使用上の注意】**

7951  
7952 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  
7953 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  
7954 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  
7955 グの状態が変化している可能性があるためである。  
7956 -----  
7957  
7958 4.4.3 データキュー  
7959  
7960 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  
7961 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  
7962 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  
7963 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  
7964 号によって識別する【NGKI1657】。  
7965  
7966 各データキューが持つ情報は次の通り【NGKI1658】。  
7967  
7968     ・データキュー属性  
7969     ・データキュー管理領域  
7970     ・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）  
7971     ・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）  
7972     ・アクセス許可ベクタ（保護機能対応カーネルの場合）  
7973     ・属する保護ドメイン（保護機能対応カーネルの場合）  
7974     ・属するクラス（マルチプロセッサ対応カーネルの場合）  
7975  
7976 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  
7977 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  
7978 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  
7979 ズを0とすることができる【NGKI1659】。  
7980  
7981 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  
7982 オブジェクト管理領域として扱われる【NGKI1660】。  
7983  
7984 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  
7985 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  
7986 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  
7987 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  
7988 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。  
7989  
7990 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】。  
7991  
7992     TA\_TPRI     0x01U     送信待ち行列をタスクの優先度順にする  
7993  
7994 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  
7995 ち行列は、FIFO順に固定されている【NGKI1663】。  
7996  
7997 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。  
7998  
7999     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  
8000                      ネルでは、静的APIによって登録されたデータキューの数





8051                   ・割り付けられるデータキューIDがない [sD] 【NGKI1675】  
8052           E\_NOMEM   メモリ不足  
8053                   ・データキュー管理領域が確保できない 【NGKI1676】  
8054           E\_OBJ     オブジェクト状態エラー  
8055                   ・dtqidで指定したデータキューが登録済み (CRE\_DTQの場合)  
8056                   【NGKI1677】  
8057                   ・その他の条件については機能の項を参照  
8058  
8059       【機能】  
8060  
8061       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  
8062       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  
8063       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  
8064       列は、空の状態に初期化される 【NGKI1679】。  
8065  
8066       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  
8067       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  
8068       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  
8069       【NGKI1681】。  
8070  
8071       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  
8072       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】。  
8073  
8074       [dtqmbにNULL以外を指定した場合]  
8075  
8076       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  
8077       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  
8078       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  
8079       【NGKI1684】。  
8080  
8081           TSZ\_DTQMB(dtqcnt)   dtqcntで指定した数のデータを格納できるデータ  
8082                                   キュー管理領域のサイズ (バイト数)  
8083           TCNT\_DTQMB(dtqcnt) dtqcntで指定した数のデータを格納できるデータ  
8084                                   キュー管理領域を確保するために必要なMB\_T型の配  
8085                                   列の要素数  
8086  
8087       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】。  
8088  
8089       MB\_T   <データキュー管理領域の変数名>[TCNT\_DTQMB(dtqcnt)];  
8090  
8091       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】。  
8092  
8093       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  
8094       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  
8095       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  
8096       クトに含まれない場合、E\_OBJエラーとなる 【NGKI1688】。  
8097  
8098       【TOPPERS/ASPカーネルにおける規定】  
8099  
8100       ASPカーネルでは、CRE\_DTQのみをサポートする 【ASPS0130】。また、dtqmbには

8101 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  
 8102 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  
 8103 サポートする【ASPS0133】。acre\_dtqに対しては、dtqmbにNULL以外を指定でき  
 8104 ないという制限はない【ASPS0134】。

8105

8106 【TOPPERS/FMPカーネルにおける規定】

8107

8108 FMPカーネルでは、CRE\_DTQのみをサポートする【FMPS0119】。また、dtqmbには  
 8109 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  
 8110 となる【FMPS0121】。

8111

8112 【TOPPERS/HRP2カーネルにおける規定】

8113

8114 HRP2カーネルでは、CRE\_DTQのみをサポートする【HRPS0119】。また、dtqmbに  
 8115 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  
 8116 ラーとなる【HRPS0121】。ただし、動的生成機能拡張パッケージでは、  
 8117 acre\_dtqもサポートする【HRPS0186】。acre\_dtqに対しては、dtqmbにNULL以外  
 8118 を指定できないという制限はない【HRPS0187】。

8119

8120 【μ ITRON4.0仕様との関係】

8121

8122 μ ITRON4.0/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、  
 8123 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  
 8124 頭番地）に改名した。また、TSZ\_DTQをTSZ\_DTQMBに改名した。

8125

8126 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  
 8127 法を規定した。

8128

8129 AID\_DTQ 割付け可能なデータキューIDの数の指定〔SD〕【NGKI1689】

8130

8131 【静的API】

8132 AID\_DTQ(uint\_t noddq)

8133

8134 【パラメータ】

8135 uint\_t noddq 割付け可能なデータキューIDの数

8136

8137 【エラーコード】

8138 E\_RSATR 予約属性

8139 ・保護ドメインの囲みの中に記述されている〔P〕【NGKI3431】

8140 ・クラスの囲みの中に記述されていない〔M〕【NGKI1690】

8141 E\_PAR パラメータエラー

8142 ・noddqが負の値【NGKI3279】

8143

8144 【機能】

8145

8146 noddqで指定した数のデータキューIDを、データキューを生成するサービスコー  
 8147 ルによって割付け可能なデータキューIDとして確保する【NGKI1691】。

8148

8149 noddqは整数定数式パラメータである【NGKI1692】。

8150

8151 **【TOPPERS/ASPカーネルにおける規定】**

8152

8153 ASPカーネルの動的生成機能拡張パッケージでは、AID\_DTQをサポートする

8154 **【ASPS0213】** .

8155

8156 **【TOPPERS/HRP2カーネルにおける規定】**

8157

8158 HRP2カーネルの動的生成機能拡張パッケージでは、AID\_DTQをサポートする

8159 **【HRPS0214】** .

8160

8161 SAC\_DTQ データキューのアクセス許可ベクタの設定 [SP] **【NGKI1693】**

8162 sac\_dtq データキューのアクセス許可ベクタの設定 [TPD] **【NGKI1694】**

8163

8164 **【静的API】**

8165 SAC\_DTQ(ID dtqid, { ACPTN acptn1, ACPTN acptn2,

8166 ACPTN acptn3, ACPTN acptn4 })

8167

8168 **【C言語API】**

8169 ER ercd = sac\_dtq(ID dtqid, const ACVCT \*p\_acvct)

8170

8171 **【パラメータ】**

8172 ID dtqid 対象データキューのID番号

8173 ACVCT \* p\_acvct アクセス許可ベクタを入れたパケットへのポ

8174 インタ（静的APIを除く）

8175

8176 \*アクセス許可ベクタ（パケットの内容）

8177 ACPTN acptn1 通常操作1のアクセス許可パターン

8178 ACPTN acptn2 通常操作2のアクセス許可パターン

8179 ACPTN acptn3 管理操作のアクセス許可パターン

8180 ACPTN acptn4 参照操作のアクセス許可パターン

8181

8182 **【リターンパラメータ】**

8183 ER ercd 正常終了 (E\_OK) またはエラーコード

8184

8185 **【エラーコード】**

8186 E\_CTX コンテキストエラー

8187 ・非タスクコンテキストからの呼出し [s] **【NGKI1695】**

8188 ・CPUロック状態からの呼出し [s] **【NGKI1696】**

8189 E\_ID 不正ID番号

8190 ・dtqidが有効範囲外 [s] **【NGKI1697】**

8191 E\_RSATR 予約属性

8192 ・対象データキューが属する保護ドメインの囲みの中に記述

8193 されていない [S] **【NGKI1698】**

8194 ・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

8195 いない [SM] **【NGKI1699】**

8196 E\_NOEXS オブジェクト未登録

8197 ・対象データキューが未登録 **【NGKI1700】**

8198 E\_OACV オブジェクトアクセス違反

8199 ・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]

8200 **【NGKI1701】**

8201 E\_MACV メモリアクセス違反  
 8202 ・ 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  
 8203 いない [s] 【NGKI1702】

8204 E\_OBJ オブジェクト状態エラー  
 8205 ・ 対象データキューは静的APIで生成された [s] 【NGKI1703】  
 8206 ・ 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]  
 8207 【NGKI1704】

8208  
 8209 【機能】  
 8210  
 8211 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  
 8212 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  
 8213 【NGKI1705】 .  
 8214  
 8215 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  
 8216 式パラメータである 【NGKI1706】 .  
 8217  
 8218 【TOPPERS/HRP2カーネルにおける規定】  
 8219  
 8220 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  
 8221 成機能拡張パッケージでは、sac\_dtqもサポートする 【HRPS0188】 .  
 8222 -----

8223 del\_dtq データキューの削除 [TD] 【NGKI1707】  
 8224

8225 【C言語API】  
 8226 ER ercd = del\_dtq(ID dtqid)  
 8227

8228 【パラメータ】  
 8229 ID dtqid 対象データキューのID番号  
 8230

8231 【リターンパラメータ】  
 8232 ER ercd 正常終了 (E\_OK) またはエラーコード  
 8233

8234 【エラーコード】  
 8235 E\_CTX コンテキストエラー  
 8236 ・ 非タスクコンテキストからの呼出し 【NGKI1708】  
 8237 ・ CPUロック状態からの呼出し 【NGKI1709】  
 8238 E\_ID 不正ID番号  
 8239 ・ dtqidが有効範囲外 【NGKI1710】  
 8240 E\_NOEXS オブジェクト未登録  
 8241 ・ 対象データキューが未登録 【NGKI1711】  
 8242 E\_OACV オブジェクトアクセス違反  
 8243 ・ 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]  
 8244 【NGKI1712】  
 8245 E\_OBJ オブジェクト状態エラー  
 8246 ・ 対象データキューは静的APIで生成された 【NGKI1713】  
 8247

8248 【機能】  
 8249  
 8250 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

8251 いは以下の通り.

8252

8253 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  
8254 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  
8255 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  
8256 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  
8257 コールからE\_DLTエラーが返る【NGKI1716】.

8258

8259 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  
8260 た場合は、そのメモリ領域が解放される【NGKI1717】.

8261

8262 【補足説明】

8263

8264 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、  
8265 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  
8266 い.

8267

8268 【使用上の注意】

8269

8270 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  
8271 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  
8272 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  
8273 み禁止時間が長くなるため、注意が必要である.

8274

8275 【TOPPERS/ASPカーネルにおける規定】

8276

8277 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  
8278 機能拡張パッケージでは、del\_dtqをサポートする【ASPS0137】.

8279

8280 【TOPPERS/FMPカーネルにおける規定】

8281

8282 FMPカーネルでは、del\_dtqをサポートしない【FMPS0123】.

8283

8284 【TOPPERS/HRP2カーネルにおける規定】

8285

8286 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  
8287 機能拡張パッケージでは、del\_dtqをサポートする【HRPS0189】.

8288

---

8289 snd\_dtq データキューへの送信 [T] 【NGKI1718】  
8290 psnd\_dtq データキューへの送信 (ポーリング) [T] 【NGKI1719】  
8291 ipsnd\_dtq データキューへの送信 (ポーリング) [I] 【NGKI1720】  
8292 tsnd\_dtq データキューへの送信 (タイムアウト付き) [T] 【NGKI1721】

8293

8294 【C言語API】

8295 ER ercd = snd\_dtq(ID dtqid, intptr\_t data)  
8296 ER ercd = psnd\_dtq(ID dtqid, intptr\_t data)  
8297 ER ercd = ipsnd\_dtq(ID dtqid, intptr\_t data)  
8298 ER ercd = tsnd\_dtq(ID dtqid, intptr\_t data, TMO tmout)

8299

8300 【パラメータ】

8301 ID dtqid 対象データキューのID番号

8302 intptr\_t data 送信データ

8303 TMO tmout タイムアウト時間 (tsnd\_dtqの場合)

8304

8305 【リターンパラメータ】

8306 ER ercd 正常終了 (E\_OK) またはエラーコード

8307

8308 【エラーコード】

8309 E\_CTX コンテキストエラー

8310 ・非タスクコンテキストからの呼出し (ipsnd\_dtqを除く)

8311 【NGKI1722】

8312 ・タスクコンテキストからの呼出し (ipsnd\_dtqの場合)

8313 【NGKI1723】

8314 ・CPUロック状態からの呼出し 【NGKI1724】

8315 ・ディスパッチ保留状態からの呼出し (snd\_dtqとtsnd\_dtqの

8316 場合) 【NGKI1725】

8317 E\_NOSPT 未サポート機能

8318 ・制約タスクからの呼出し (snd\_dtqとtsnd\_dtqの場合) 【NGKI1726】

8319 E\_ID 不正ID番号

8320 ・dtqidが有効範囲外 【NGKI1727】

8321 E\_PAR パラメータエラー

8322 ・tmoutが無効 (tsnd\_dtqの場合) 【NGKI1728】

8323 E\_NOEXS オブジェクト未登録

8324 ・対象データキューが未登録 [D] 【NGKI1729】

8325 E\_OACV オブジェクトアクセス違反

8326 ・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

8327 (ipsnd\_dtqを除く) [P] 【NGKI1730】

8328 E\_TMOUT ポーリング失敗またはタイムアウト (snd\_dtqを除く) 【NGKI1731】

8329 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

8330 の場合) 【NGKI1732】

8331 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

8332 の場合) 【NGKI1733】

8333

8334 【機能】

8335

8336 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

8337 を送信する。具体的な振舞いは以下の通り。

8338

8339 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

8340 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

8341 【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

8342 からE\_OKが返る 【NGKI1735】。

8343

8344 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

8345 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

8346 順でデータキュー管理領域に格納される 【NGKI1736】。

8347

8348 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

8349 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

8350 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

8351       【NGKI1737】 .

8352 -----

8353 fsnd\_dtq   データキューへの強制送信 [T]   【NGKI1738】

8354 ifsnd\_dtq   データキューへの強制送信 [I]   【NGKI1739】

8355

8356       【C言語API】

8357       ER ercd = fsnd\_dtq(ID dtqid, intptr\_t data)

8358       ER ercd = ifsnd\_dtq(ID dtqid, intptr\_t data)

8359

8360       【パラメータ】

8361       ID           dtqid           対象データキューのID番号

8362       intptr\_t     data           送信データ

8363

8364       【リターンパラメータ】

8365       ER           ercd           正常終了 (E\_OK) またはエラーコード

8366

8367       【エラーコード】

8368       E\_CTX       コンテキストエラー

8369                ・非タスクコンテキストからの呼出し (fsnd\_dtqの場合)   【NGKI1740】

8370                ・タスクコンテキストからの呼出し (ifsnd\_dtqの場合)   【NGKI1741】

8371                ・CPUロック状態からの呼出し   【NGKI1742】

8372       E\_ID       不正ID番号

8373                ・dtqidが有効範囲外   【NGKI1743】

8374       E\_NOEXS   オブジェクト未登録

8375                ・対象データキューが未登録 [D]   【NGKI1744】

8376       E\_OACV   オブジェクトアクセス違反

8377                ・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

8378                (fsnd\_dtqの場合) [P]   【NGKI1745】

8379       E\_ILUSE   サービスコール不正使用

8380                ・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】

8381

8382       【機能】

8383

8384 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

8385 を強制送信する。具体的な振舞いは以下の通り。

8386

8387 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

8388 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

8389   【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

8390 からE\_OKが返る   【NGKI1748】。

8391

8392 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

8393 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

8394 順でデータキュー管理領域に格納される   【NGKI1749】。

8395

8396 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

8397 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

8398 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

8399 が、FIFO順でデータキュー管理領域に格納される   【NGKI1750】。

8400 -----



8401 rcv\_dtq データキューからの受信 [T] 【NGKI1751】  
 8402 prcv\_dtq データキューからの受信 (ポーリング) [T] 【NGKI1752】  
 8403 trcv\_dtq データキューからの受信 (タイムアウト付き) [T] 【NGKI1753】  
 8404  
 8405 **【C言語API】**  
 8406 ER ercd = rcv\_dtq(ID dtqid, intptr\_t \*p\_data)  
 8407 ER ercd = prcv\_dtq(ID dtqid, intptr\_t \*p\_data)  
 8408 ER ercd = trcv\_dtq(ID dtqid, intptr\_t \*p\_data, TMO tmout)  
 8409  
 8410 **【パラメータ】**  
 8411 ID dtqid 対象データキューのID番号  
 8412 intptr\_t \* p\_data 受信データを入れるメモリ領域へのポインタ  
 8413 TMO tmout タイムアウト時間 (trcv\_dtqの場合)  
 8414  
 8415 **【リターンパラメータ】**  
 8416 ER ercd 正常終了 (E\_OK) またはエラーコード  
 8417 intptr\_t data 受信データ  
 8418  
 8419 **【エラーコード】**  
 8420 E\_CTX コンテキストエラー  
 8421 ・非タスクコンテキストからの呼出し 【NGKI1754】  
 8422 ・CPUロック状態からの呼出し 【NGKI1755】  
 8423 ・ディスパッチ保留状態からの呼出し (prcv\_dtqを除く)  
 8424 【NGKI1756】  
 8425 E\_NOSPT 未サポート機能  
 8426 ・制約タスクからの呼出し (prcv\_dtqを除く) 【NGKI1757】  
 8427 E\_ID 不正ID番号  
 8428 ・dtqidが有効範囲外 【NGKI1758】  
 8429 E\_PAR パラメータエラー  
 8430 ・tmoutが無効 (trcv\_dtqの場合) 【NGKI1759】  
 8431 E\_NOEXS オブジェクト未登録  
 8432 ・対象データキューが未登録 [D] 【NGKI1760】  
 8433 E\_OACV オブジェクトアクセス違反  
 8434 ・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]  
 8435 【NGKI1761】  
 8436 E\_MACV メモリアクセス違反  
 8437 ・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  
 8438 いない [P] 【NGKI1762】  
 8439 E\_TMOUT ポーリング失敗またはタイムアウト (rcv\_dtqを除く) 【NGKI1763】  
 8440 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)  
 8441 【NGKI1764】  
 8442 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)  
 8443 【NGKI1765】  
 8444  
 8445 **【機能】**  
 8446  
 8447 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  
 8448 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  
 8449 る【NGKI3421】。具体的な振舞いは以下の通り。  
 8450

8451 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、  
 8452 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  
 8453 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  
 8454 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  
 8455 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  
 8456 ビスコールからE\_OKが返る【NGKI1768】。

8457  
 8458 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  
 8459 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  
 8460 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  
 8461 【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  
 8462 からE\_OKが返る【NGKI1770】。

8463  
 8464 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  
 8465 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  
 8466 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】。

8467 -----

8468 ini\_dtq      データキューの再初期化 [T]   【NGKI1772】

8469

8470   【C言語API】

8471       ER ercd = ini\_dtq(ID dtqid)

8472

8473   【パラメータ】

8474       ID           dtqid           対象データキューのID番号

8475

8476   【リターンパラメータ】

8477       ER           ercd           正常終了 (E\_OK) またはエラーコード

8478

8479   【エラーコード】

8480       E\_CTX       コンテキストエラー

8481                   ・非タスクコンテキストからの呼出し【NGKI1773】

8482                   ・CPUロック状態からの呼出し【NGKI1774】

8483       E\_ID       不正ID番号

8484                   ・dtqidが有効範囲外【NGKI1775】

8485       E\_NOEXS   オブジェクト未登録

8486                   ・対象データキューが未登録 [D]   【NGKI1776】

8487       E\_OACV    オブジェクトアクセス違反

8488                   ・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]

8489                   【NGKI1777】

8490

8491   【機能】

8492

8493 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  
 8494 振舞いは以下の通り。

8495

8496 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  
 8497 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  
 8498 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  
 8499 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  
 8500 たサービスコールからE\_DLTエラーが返る【NGKI1780】。

8501

8502     **【補足説明】**

8503

8504     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、  
 8505     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。  
 8506

8507

8508     **【使用上の注意】**

8509

8510     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  
 8511     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  
 8512     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  
 8513     み禁止時間が長くなるため、注意が必要である。

8514

8515     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、  
 8516     アプリケーションの責任である。

8517

8518     **【 $\mu$  ITRON4.0仕様との関係】**

8519

8520      $\mu$  ITRON4.0仕様に定義されていないサービスコールである。

8521

8522     ref\_dtq     データキューの状態参照 [T]     **【NGKI1781】**

8523

8524     **【C言語API】**

8525     ER ercd = ref\_dtq(ID dtqid, T\_RDTQ \*pk\_rdtq)

8526

8527     **【パラメータ】**

8528	ID	dtqid	対象データキューのID番号
8529	T_RDTQ *	pk_rdtq	データキューの現在状態を入れるパケットへの ポインタ

8530

8531

8532     **【リターンパラメータ】**

8533	ER	ercd	正常終了 (E_OK) またはエラーコード
------	----	------	-----------------------

8534

8535     \* データキューの現在状態 (パケットの内容)

8536	ID	stskid	データキューの送信待ち行列の先頭のタスクの ID番号
8537			
8538	ID	rtskid	データキューの受信待ち行列の先頭のタスクの ID番号
8539			
8540	uint_t	sdtqcnt	データキュー管理領域に格納されているデータ の数

8541

8542

8543     **【エラーコード】**

8544	E_CTX	コンテキストエラー
8545		・ 非タスクコンテキストからの呼出し <b>【NGKI1782】</b>
8546		・ CPUロック状態からの呼出し <b>【NGKI1783】</b>
8547	E_ID	不正ID番号
8548		・ dtqidが有効範囲外 <b>【NGKI1784】</b>
8549	E_NOEXS	オブジェクト未登録
8550		・ 対象データキューが未登録 [D] <b>【NGKI1785】</b>

8551 E\_OACV オブジェクトアクセス違反  
 8552 ・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]  
 8553 【NGKI1786】  
 8554 E\_MACV メモリアクセス違反  
 8555 ・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  
 8556 いない [P] 【NGKI1787】  
 8557

#### 8558 【機能】

8559  
 8560 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  
 8561 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】．  
 8562  
 8563 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  
 8564 TSK\_NONE（=0）が返る 【NGKI1789】．また，受信待ち行列にタスクが存在しな  
 8565 い場合，rtskidにはTSK\_NONE（=0）が返る 【NGKI1790】．  
 8566

#### 8567 【使用上の注意】

8568  
 8569 ref\_dtqはデバッグ時向けの機能であり，その他の目的に使用することは推奨し  
 8570 ない．これは，ref\_dtqを呼び出し，対象データキューの現在状態を参照した直  
 8571 後に割込みが発生した場合，ref\_dtqから戻ってきた時には対象データキューの  
 8572 状態が変化している可能性があるためである．  
 8573 -----

#### 8574 4.4.4 優先度データキュー

8575  
 8576 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  
 8577 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  
 8578 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  
 8579 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  
 8580 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】．  
 8581

8582  
 8583 各優先度データキューが持つ情報は次の通り 【NGKI1792】．  
 8584

- 8585 ・優先度データキュー属性
- 8586 ・優先度データキュー管理領域
- 8587 ・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）
- 8588 ・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）
- 8589 ・送信できるデータ優先度の最大値
- 8590 ・アクセス許可ベクタ（保護機能対応カーネルの場合）
- 8591 ・属する保護ドメイン（保護機能対応カーネルの場合）
- 8592 ・属するクラス（マルチプロセッサ対応カーネルの場合）  
 8593

8594 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，  
 8595 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  
 8596 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，  
 8597 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】．  
 8598

8599 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  
 8600 用いるオブジェクト管理領域として扱われる 【NGKI1794】．

8601  
8602 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  
8603 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  
8604 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  
8605 タキューからデータが受信できるまで待っている状態（優先度データキューか  
8606 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  
8607 キューである。  
8608  
8609 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】。  
8610  
8611 TA\_TPRI 0x01U 送信待ち行列をタスクの優先度順にする  
8612  
8613 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  
8614 ち行列は、FIFO順に固定されている【NGKI1797】。  
8615  
8616 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。  
8617  
8618 TMIN\_DPRI データ優先度の最小値（=1）【NGKI1798】  
8619 TMAX\_DPRI データ優先度の最大値  
8620  
8621 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  
8622 いカーネルでは、静的APIによって登録された優先度デー  
8623 タキューの数に一致）【NGKI1799】  
8624  
8625 【TOPPERS/ASPカーネルにおける規定】  
8626  
8627 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  
8628 【ASPS0138】。ただし、タスク優先度拡張パッケージでは、TMAX\_DPRIを256に  
8629 拡張する【ASPS0139】。  
8630  
8631 【TOPPERS/FMPカーネルにおける規定】  
8632  
8633 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  
8634 【FMPS0124】。  
8635  
8636 【TOPPERS/HRP2カーネルにおける規定】  
8637  
8638 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  
8639 【HRPS0124】。  
8640  
8641 【使用上の注意】  
8642  
8643 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  
8644 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  
8645 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、  
8646 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  
8647 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  
8648 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  
8649 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  
8650 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。

8651

8652     **【 $\mu$  ITRON4.0仕様との関係】**

8653

8654      $\mu$  ITRON4.0仕様に規定されていない機能である。

8655

8656     CRE\_PDQ     優先度データキューの生成 [S]     **【NGKI1800】**8657     acre\_pdq     優先度データキューの生成 [TD]     **【NGKI1801】**

8658

8659     **【静的API】**

8660         CRE\_PDQ(ID pdqid, { ATR pdqatr, uint\_t pdqcnt, PRI maxdpri, void \*pdqmb })

8661

8662     **【C言語API】**

8663         ER\_ID pdqid = acre\_pdq(const T\_CPDQ \*pk\_cpdq)

8664

8665     **【パラメータ】**8666         ID             pdqid             生成する優先度データキューのID番号 (CRE\_PDQ  
8667                                             の場合)8668         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  
8669                                             トへのポインタ (静的APIを除く)

8670

8671     \* 優先度データキューの生成情報 (パケットの内容)

8672         ATR             pdqatr             優先度データキュー属性

8673         uint\_t         pdqcnt             優先度データキュー管理領域に格納できるデー  
8674                                             タ数8675         PRI             maxdpri           優先度データキューに送信できるデータ優先度  
8676                                             の最大値

8677         void \*         pdqmb             優先度データキュー管理領域の先頭番地

8678

8679     **【リターンパラメータ】**8680         ER\_ID             pdqid           生成された優先度データキューのID番号 (正の  
8681                                             値) またはエラーコード

8682

8683     **【エラーコード】**

8684         E\_CTX             コンテキストエラー

8685             ・非タスクコンテキストからの呼出し [s]     **【NGKI1802】**8686             ・CPUロック状態からの呼出し [s]     **【NGKI1803】**

8687         E\_RSATR           予約属性

8688             ・pdqatrが無効 **【NGKI1804】**8689             ・属する保護ドメインの指定が有効範囲外 [sP]     **【NGKI1805】**8690             ・属するクラスの指定が有効範囲外 [sM]     **【NGKI1806】**8691             ・クラスの囲みの中に記述されていない [SM]     **【NGKI1807】**

8692         E\_NOSPT           未サポート機能

8693             ・条件については各カーネルにおける規定の項を参照

8694         E\_PAR             パラメータエラー

8695             ・pdqcntが負の値 [S]     **【NGKI3289】**8696             ・maxdpriがTMIN\_DPRIより小さい, またはTMAX\_DPRIより大き  
8697                                             い **【NGKI1819】**

8698             ・その他の条件については機能の項を参照

8699         E\_OACV           オブジェクトアクセス違反

8700             ・システム状態に対する管理操作が許可されていない [sP]

8701                   【NGKI1808】  
8702       E\_MACV       メモリアクセス違反  
8703                   ・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  
8704                   いない [sP]   【NGKI1809】  
8705       E\_NOID       ID番号不足  
8706                   ・割り付けられる優先度データキューIDがない [sD]   【NGKI1810】  
8707       E\_NOMEM       メモリ不足  
8708                   ・優先度データキュー管理領域が確保できない 【NGKI1811】  
8709       E\_OBJ        オブジェクト状態エラー  
8710                   ・pdqidで指定した優先度データキューが登録済み (CRE\_PDQ  
8711                    の場合)   【NGKI1812】  
8712                   ・その他の条件については機能の項を参照  
8713  
8714       【機能】  
8715  
8716       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  
8717       タキューを生成する。pdqcntとpdqmbから優先度データキュー管理領域が設定され、  
8718       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  
8719       ち行列と受信待ち行列は、空の状態に初期化される 【NGKI1814】。  
8720  
8721       静的APIにおいては、pdqidはオブジェクト識別名、pdqatr, pdqcnt, maxdpriは  
8722       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  
8723       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  
8724       ができない 【NGKI1816】。  
8725  
8726       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  
8727       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  
8728       【NGKI1817】。  
8729  
8730       [pdqmbにNULL以外を指定した場合]  
8731  
8732       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  
8733       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  
8734       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  
8735       用意している 【NGKI1821】。  
8736  
8737       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  
8738                               タキュー管理領域のサイズ (バイト数)  
8739       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  
8740                               タキュー管理領域を確保するために必要なMB\_T型の  
8741                               配列の要素数  
8742  
8743       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  
8744       【NGKI1822】。  
8745  
8746       MB\_T     <優先度データキュー管理領域の変数名>[TCNT\_PDQMB(pdqcnt)];  
8747  
8748       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  
8749       【NGKI1823】。  
8750

この方法に従わず、pdqmbにターゲット定義の制約に合致しない先頭番地を指定した時には、E\_PARエラーとなる【NGKI1824】。また、保護機能対応カーネルにおいて、pdqmbで指定した優先度データキュー管理領域がカーネル専用のメモリオブジェクトに含まれない場合、E\_OBJエラーとなる【NGKI1825】。

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、CRE\_PDQのみをサポートする【ASPS0140】。また、pdqmbにはNULLのみを指定することができる。NULL以外を指定した場合には、E\_NOSPTエラーとなる【ASPS0142】。ただし、動的生成機能拡張パッケージでは、acre\_pdqもサポートする【ASPS0143】。acre\_pdqに対しては、pdqmbにNULL以外を指定できないという制限はない【ASPS0144】。

#### 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、CRE\_PDQのみをサポートする【FMPS0125】。また、pdqmbにはNULLのみを指定することができる。NULL以外を指定した場合には、E\_NOSPTエラーとなる【FMPS0127】。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、CRE\_PDQのみをサポートする【HRPS0125】。また、pdqmbにはNULLのみを指定することができる。NULL以外を指定した場合には、E\_NOSPTエラーとなる【HRPS0127】。ただし、動的生成機能拡張パッケージでは、acre\_pdqもサポートする【HRPS0190】。acre\_pdqに対しては、pdqmbにNULL以外を指定できないという制限はない【HRPS0191】。

-----  
AID\_PDQ      割付け可能な優先度データキューIDの数の指定 [SD] 【NGKI1826】

#### 【静的API】

AID\_PDQ(uint\_t nopdq)

#### 【パラメータ】

uint\_t      nopdq      割付け可能な優先度データキューIDの数

#### 【エラーコード】

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

#### 【機能】

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

nopdqは整数定数式パラメータである【NGKI1829】。





8851 E\_MACV メモリアクセス違反  
 8852 ・ 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  
 8853 いない [s] 【NGKI1839】  
 8854 E\_OBJ オブジェクト状態エラー  
 8855 ・ 対象優先度データキューは静的APIで生成された [s] 【NGKI1840】  
 8856 ・ 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  
 8857 定済み [S] 【NGKI1841】

#### 8859 【機能】

8860  
 8861 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  
 8862 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  
 8863 設定する【NGKI1842】。

8864  
 8865 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  
 8866 式パラメータである【NGKI1843】。

#### 8868 【TOPPERS/HRP2カーネルにおける規定】

8869  
 8870 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  
 8871 成機能拡張パッケージでは、sac\_pdqもサポートする【HRPS0192】。

8872 -----  
 8873 del\_pdq 優先度データキューの削除 [TD] 【NGKI1844】

#### 8875 【C言語API】

8876 ER ercd = del\_pdq(ID pdqid)

#### 8878 【パラメータ】

8879 ID pdqid 対象優先度データキューのID番号

#### 8881 【リターンパラメータ】

8882 ER ercd 正常終了 (E\_OK) またはエラーコード

#### 8884 【エラーコード】

8885 E\_CTX コンテキストエラー  
 8886 ・ 非タスクコンテキストからの呼出し【NGKI1845】  
 8887 ・ CPUロック状態からの呼出し【NGKI1846】  
 8888 E\_ID 不正ID番号  
 8889 ・ pdqidが有効範囲外【NGKI1847】  
 8890 E\_NOEXS オブジェクト未登録  
 8891 ・ 対象優先度データキューが未登録【NGKI1848】  
 8892 E\_OACV オブジェクトアクセス違反  
 8893 ・ 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  
 8894 ない [P] 【NGKI1849】  
 8895 E\_OBJ オブジェクト状態エラー  
 8896 ・ 対象優先度データキューは静的APIで生成された【NGKI1850】

#### 8898 【機能】

8899  
 8900 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。

8901 具体的な振舞いは以下の通り。  
 8902  
 8903 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  
 8904 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  
 8905 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  
 8906 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  
 8907 態となったサービスコールからE\_DLTエラーが返る【NGKI1853】。  
 8908  
 8909 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  
 8910 て確保された場合は、そのメモリ領域が解放される【NGKI1854】。  
 8911  
 8912 【補足説明】  
 8913  
 8914 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、  
 8915 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  
 8916 い。  
 8917  
 8918 【使用上の注意】  
 8919  
 8920 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  
 8921 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  
 8922 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  
 8923 み禁止時間が長くなるため、注意が必要である。  
 8924  
 8925 【TOPPERS/ASPカーネルにおける規定】  
 8926  
 8927 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  
 8928 機能拡張パッケージでは、del\_pdqをサポートする【ASPS0147】。  
 8929  
 8930 【TOPPERS/FMPカーネルにおける規定】  
 8931  
 8932 FMPカーネルでは、del\_pdqをサポートしない【FMPS0129】。  
 8933  
 8934 【TOPPERS/HRP2カーネルにおける規定】  
 8935  
 8936 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  
 8937 機能拡張パッケージでは、del\_pdqをサポートする【HRPS0193】。  
 8938 -----  
 8939 snd\_pdq 優先度データキューへの送信 [T] 【NGKI1855】  
 8940 psnd\_pdq 優先度データキューへの送信（ポーリング） [T] 【NGKI1856】  
 8941 ipsnd\_pdq 優先度データキューへの送信（ポーリング） [I] 【NGKI1857】  
 8942 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】  
 8943  
 8944 【C言語API】  
 8945 ER ercd = snd\_pdq(ID pdqid, intptr\_t data, PRI datapri)  
 8946 ER ercd = psnd\_pdq(ID pdqid, intptr\_t data, PRI datapri)  
 8947 ER ercd = ipsnd\_pdq(ID pdqid, intptr\_t data, PRI datapri)  
 8948 ER ercd = tsnd\_pdq(ID pdqid, intptr\_t data, PRI datapri, TMO tmout)  
 8949  
 8950 【パラメータ】

8951	ID	pdqid	対象優先度データキューのID番号
8952	intptr_t	data	送信データ
8953	PRI	datapri	送信データの優先度
8954	TMO	tmout	タイムアウト時間（tsnd_pdqの場合）
8955			
8956	【リターンパラメータ】		
8957	ER	ercd	正常終了（E_OK）またはエラーコード
8958			
8959	【エラーコード】		
8960	E_CTX	コンテキストエラー	
8961		・非タスクコンテキストからの呼出し（ipsnd_pdqを除く）	
8962		【NGKI1859】	
8963		・タスクコンテキストからの呼出し（ipsnd_pdqの場合）【NGKI1860】	
8964		・CPUロック状態からの呼出し【NGKI1861】	
8965		・ディスパッチ保留状態からの呼出し（snd_pdqとtsnd_pdqの	
8966		場合）【NGKI1862】	
8967	E_NOSPT	未サポート機能	
8968		・制約タスクからの呼出し（snd_pdqとtsnd_pdqの場合）【NGKI1863】	
8969	E_ID	不正ID番号	
8970		・pdqidが有効範囲外【NGKI1864】	
8971	E_PAR	パラメータエラー	
8972		・tmoutが無効（tsnd_pdqの場合）【NGKI1865】	
8973		・その他の条件については機能の項を参照	
8974	E_NOEXS	オブジェクト未登録	
8975		・対象優先度データキューが未登録〔D〕【NGKI1866】	
8976	E_OACV	オブジェクトアクセス違反	
8977		・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	
8978		ない（ipsnd_pdqを除く）〔P〕【NGKI1867】	
8979	E_TMOUT	ポーリング失敗またはタイムアウト（snd_pdqを除く）【NGKI1868】	
8980	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	
8981		の場合）【NGKI1869】	
8982	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	
8983		の場合）【NGKI1870】	
8984			
8985	【機能】		
8986			
8987	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		
8988	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		
8989	の通り．		
8990			
8991	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		
8992	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		
8993	【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		
8994	からE_OKが返る【NGKI1872】．		
8995			
8996	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		
8997	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		
8998	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		
8999	に格納される【NGKI1873】．		
9000			

9001 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  
 9002 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  
 9003 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  
 9004 行列につながる【NGKI1874】。

9005

9006 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  
 9007 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  
 9008 【NGKI1876】。

9009

9010 rcv\_pdq 優先度データキューからの受信 [T] 【NGKI1877】

9011 prcv\_pdq 優先度データキューからの受信（ポーリング） [T] 【NGKI1878】

9012 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】

9013

#### 9014 【C言語API】

9015 ER ercd = rcv\_pdq(ID pdqid, intptr\_t \*p\_data, PRI \*p\_datapri)

9016 ER ercd = prcv\_pdq(ID pdqid, intptr\_t \*p\_data, PRI \*p\_datapri)

9017 ER ercd = trcv\_pdq(ID pdqid, intptr\_t \*p\_data, PRI \*p\_datapri, TMO tmout)

9018

#### 9019 【パラメータ】

9020 ID pdqid 対象優先度データキューのID番号

9021 intptr\_t \* p\_data 受信データを入れるメモリ領域へのポインタ

9022 PRI \* p\_datapri 受信データの優先度を入れるメモリ領域へのポ  
 9023 インタ

9024 TMO tmout タイムアウト時間（trcv\_pdqの場合）

9025

#### 9026 【リターンパラメータ】

9027 ER ercd 正常終了（E\_OK）またはエラーコード

9028 intptr\_t data 受信データ

9029 PRI datapri 受信データの優先度

9030

#### 9031 【エラーコード】

9032 E\_CTX コンテキストエラー

9033 ・非タスクコンテキストからの呼出し【NGKI1880】

9034 ・CPUロック状態からの呼出し【NGKI1881】

9035 ・ディスパッチ保留状態からの呼出し（prcv\_pdqを除く）【NGKI1882】

9036 E\_NOSPT 未サポート機能

9037 ・制約タスクからの呼出し（prcv\_pdqを除く）【NGKI1883】

9038 E\_ID 不正ID番号

9039 ・pdqidが有効範囲外【NGKI1884】

9040 E\_PAR パラメータエラー

9041 ・tmoutが無効（trcv\_pdqの場合）【NGKI1885】

9042 E\_NOEXS オブジェクト未登録

9043 ・対象優先度データキューが未登録 [D] 【NGKI1886】

9044 E\_OACV オブジェクトアクセス違反

9045 ・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  
 9046 ない [P] 【NGKI1887】

9047 E\_MACV メモリアクセス違反

9048 ・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  
 9049 いない [P] 【NGKI1888】

9050 ・p\_datapriが指すメモリ領域への書込みアクセスが許可され

9051                    っていない [P] 【NGKI1889】  
 9052        E\_TMOUT    ポーリング失敗またはタイムアウト (rcv\_pdqを除く) 【NGKI1890】  
 9053        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)  
 9054                    【NGKI1891】  
 9055        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)  
 9056                    【NGKI1892】

# 9057 【機能】

9058  
 9059  
 9060 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  
 9061 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  
 9062 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】.  
 9063 具体的な振舞いは以下の通り.

9064  
 9065 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  
 9066 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  
 9067 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  
 9068 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  
 9069 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  
 9070 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  
 9071 【NGKI1896】.

9072  
 9073 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  
 9074 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  
 9075 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,  
 9076 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  
 9077 サービスコールからE\_OKが返る 【NGKI1900】.

9078  
 9079 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  
 9080 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  
 9081 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  
 9082 につながる 【NGKI1901】.

9083 -----  
 9084 ini\_pdq        優先度データキューの再初期化 [T] 【NGKI1902】

## 9085 【C言語API】

9086        ER ercd = ini\_pdq(ID pdqid)

## 9087 【パラメータ】

9088        ID            pdqid            対象優先度データキューのID番号

## 9089 【リターンパラメータ】

9090        ER            ercd            正常終了 (E\_OK) またはエラーコード

## 9091 【エラーコード】

9092        E\_CTX        コンテキストエラー  
 9093                    ・非タスクコンテキストからの呼出し 【NGKI1903】  
 9094                    ・CPUロック状態からの呼出し 【NGKI1904】  
 9095        E\_ID          不正ID番号  
 9096                    ・pdqidが有効範囲外 【NGKI1905】

9101 E\_NOEXS オブジェクト未登録  
 9102 ・対象優先度データキューが未登録 [D] 【NGKI1906】  
 9103 E\_OACV オブジェクトアクセス違反  
 9104 ・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  
 9105 ない [P] 【NGKI1907】  
 9106  
 9107 **【機能】**  
 9108  
 9109 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  
 9110 る．具体的な振舞いは以下の通り．  
 9111  
 9112 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  
 9113 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  
 9114 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  
 9115 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、  
 9116 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】．  
 9117  
 9118 **【補足説明】**  
 9119  
 9120 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、  
 9121 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  
 9122 い．  
 9123  
 9124 **【使用上の注意】**  
 9125  
 9126 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  
 9127 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  
 9128 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  
 9129 み禁止時間が長くなるため、注意が必要である．  
 9130  
 9131 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  
 9132 つのは、アプリケーションの責任である．  
 9133 -----  
 9134 ref\_pdq 優先度データキューの状態参照 [T] 【NGKI1911】  
 9135  
 9136 **【C言語API】**  
 9137 ER ercd = ref\_pdq(ID pdqid, T\_RPDQ \*pk\_rpdq)  
 9138  
 9139 **【パラメータ】**  
 9140 ID pdqid 対象優先度データキューのID番号  
 9141 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  
 9142 トへのポインタ  
 9143  
 9144 **【リターンパラメータ】**  
 9145 ER ercd 正常終了 (E\_OK) またはエラーコード  
 9146  
 9147 \* 優先度データキューの現在状態（パケットの内容）  
 9148 ID stskid 優先度データキューの送信待ち行列の先頭のタ  
 9149 スクのID番号  
 9150 ID rtskid 優先度データキューの受信待ち行列の先頭のタ

9151 スクのID番号  
 9152 uint\_t spdqcnt 優先度データキュー管理領域に格納されている  
 9153 データの数  
 9154  
 9155 **【エラーコード】**  
 9156 E\_CTX コンテキストエラー  
 9157 ・非タスクコンテキストからの呼出し【NGKI1912】  
 9158 ・CPUロック状態からの呼出し【NGKI1913】  
 9159 E\_ID 不正ID番号  
 9160 ・pdqidが有効範囲外【NGKI1914】  
 9161 E\_NOEXS オブジェクト未登録  
 9162 ・対象優先度データキューが未登録 [D] 【NGKI1915】  
 9163 E\_OACV オブジェクトアクセス違反  
 9164 ・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  
 9165 ない [P] 【NGKI1916】  
 9166 E\_MACV メモリアクセス違反  
 9167 ・pk\_rpdqが指すメモリ領域への書き込みアクセスが許可されて  
 9168 いない [P] 【NGKI1917】  
 9169  
 9170 **【機能】**  
 9171  
 9172 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  
 9173 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  
 9174 【NGKI1918】．  
 9175  
 9176 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  
 9177 はTSK\_NONE（=0）が返る【NGKI1919】．また、受信待ち行列にタスクが存在し  
 9178 ない場合、rtskidにはTSK\_NONE（=0）が返る【NGKI1920】．  
 9179  
 9180 **【使用上の注意】**  
 9181  
 9182 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  
 9183 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  
 9184 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  
 9185 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．  
 9186 -----  
 9187  
 9188 4.4.5 メールボックス  
 9189  
 9190 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  
 9191 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  
 9192 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】．  
 9193  
 9194 各メールボックスが持つ情報は次の通り【NGKI1922】．  
 9195  
 9196 ・メールボックス属性  
 9197 ・メッセージキュー  
 9198 ・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）  
 9199 ・送信できるメッセージ優先度の最大値  
 9200 ・優先度別のメッセージキューヘッダ領域



9201           ・ 属するクラス（マルチプロセッサ対応カーネルの場合）  
 9202  
 9203           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  
 9204           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。  
 9205  
 9206           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  
 9207           （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  
 9208           順序でつながれているキューである。  
 9209  
 9210           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】。  
 9211  
 9212           TA\_TPRI        0x01U    待ち行列をタスクの優先度順にする  
 9213           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  
 9214  
 9215           TA\_TPRIを指定しない場合、待ち行列はFIFO順になる【NGKI1924】。TA\_MPRIを  
 9216           指定しない場合、メッセージキューはFIFO順になる【NGKI1925】。  
 9217  
 9218           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  
 9219           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。  
 9220  
 9221           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  
 9222           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  
 9223           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  
 9224           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】。  
 9225           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  
 9226           ているか否かにより、以下のいずれかを用いる【NGKI1928】。  
 9227  
 9228           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  
 9229           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  
 9230  
 9231           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  
 9232           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  
 9233           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  
 9234           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  
 9235           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  
 9236           送信した場合の動作は保証されない【NGKI1930】。  
 9237  
 9238           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  
 9239           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  
 9240           ルドに設定する【NGKI1931】。  
 9241  
 9242           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】。  
 9243  
 9244           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。  
 9245  
 9246           TMIN\_MPRI        メッセージ優先度の最小値（=1）   【NGKI1933】  
 9247           TMAX\_MPRI        メッセージ優先度の最大値  
 9248  
 9249           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  
 9250           ネルでは、静的APIによって登録されたメールボックスの

9251 数に一致) 【NGKI1934】

9252

9253 【補足説明】

9254

9255 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

9256 ダ領域は用いていない。

9257

9258 【使用上の注意】

9259

9260 メールボックス機能は、 $\mu$  ITRON4.0仕様との互換性のために残した機能であり、

9261 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。

9262 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

9263 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

9264 する方法で置き換えることができる。

9265

9266 【TOPPERS/ASPカーネルにおける規定】

9267

9268 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

9269 ジ優先度の最大値 (TMAX\_MPRI) は16に固定されている【ASPS0148】。ただし、

9270 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】。

9271

9272 【TOPPERS/FMPカーネルにおける規定】

9273

9274 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

9275 ジ優先度の最大値 (TMAX\_MPRI) は16に固定されている【FMPS0131】。

9276

9277 【TOPPERS/HRP2カーネルにおける規定】

9278

9279 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】。

9280

9281 【 $\mu$  ITRON4.0仕様との関係】

9282

9283 TNUM\_MBXIDは、 $\mu$  ITRON4.0仕様に規定されていないカーネル構成マクロである。

9284 -----

9285 CRE\_MBX メールボックスの生成 [Sp] 【NGKI1935】

9286 acre\_mbx メールボックスの生成 [TpD] 【NGKI1936】

9287

9288 【静的API】

9289 CRE\_MBX(ID mbxid, { ATR mbxatr, PRI maxmpri, void \*mprihd })

9290

9291 【C言語API】

9292 ER\_ID mbxid = acre\_mbx(const T\_CMBX \*pk\_cmbx)

9293

9294 【パラメータ】

9295 ID mbxid 生成するメールボックスのID番号 (CRE\_MBXの場合)

9296

9297 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)

9298

9299

9300 \* メールボックスの生成情報 (パケットの内容)

9301

ATR

mbxatr

メールボックス属性

9302

PRI

maxmpri

優先度メールボックスに送信できるメッセージ

9303

優先度の最大値

9304

void \*

mprihd

優先度別のメッセージキューヘッダ領域の先頭

9305

番地

9306

9307

【リターンパラメータ】

9308

ER\_ID

mbxid

生成されたメールボックスのID番号（正の値）

9309

またはエラーコード

9310

9311

【エラーコード】

9312

E\_CTX

コンテキストエラー

9313

・非タスクコンテキストからの呼出し [s] 【NGKI1937】

9314

・CPUロック状態からの呼出し [s] 【NGKI1938】

9315

E\_RSATR

予約属性

9316

・mbxatrが無効 【NGKI1939】

9317

・属するクラスの指定が有効範囲外 [sM] 【NGKI1940】

9318

・クラスの囲みの中に記述されていない [SM] 【NGKI1941】

9319

E\_NOSPT

未サポート機能

9320

・条件については各カーネルにおける規定の項を参照

9321

E\_PAR

パラメータエラー

9322

・maxmpriがTMIN\_MPRIより小さい、またはTMAX\_MPRIより大きい 【NGKI1951】

9323

9324

E\_NOID

ID番号不足

9325

・割り付けられるメールボックスIDがない [sD] 【NGKI1942】

9326

E\_NOMEM

メモリ不足

9327

・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

9328

【NGKI1943】

9329

E\_OBJ

オブジェクト状態エラー

9330

・mbxidで指定したメールボックスが登録済み（CRE\_MBXの場合） 【NGKI1944】

9331

9332

9333

【機能】

9334

9335

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

9336

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

9337

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

9338

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

9339

9340

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

9341

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

9342

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

9343

ない【NGKI1948】。

9344

9345

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

9346

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

9347

【NGKI1949】。

9348

9349

【未決定事項】

9350

9351 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  
9352 ない。

9353

9354 **【TOPPERS/ASPカーネルにおける規定】**

9355

9356 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  
9357 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  
9358 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

9359 **【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  
9360 トする**【ASPS0153】**。acre\_mbxに対しても、mprihdにはNULLのみを指定するこ  
9361 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  
9362 いため、E\_NOMEMが返ることはない**【ASPS0155】**。

9363

9364 **【TOPPERS/FMPカーネルにおける規定】**

9365

9366 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  
9367 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  
9368 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

9369 **【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、  
9370 E\_NOMEMが返ることはない**【FMPS0135】**。

9371

9372 AID\_MBX 割付け可能なメールボックスIDの数の指定 [SpD] **【NGKI1952】**

9373

9374 **【静的API】**

9375 AID\_MBX(uint\_t nombx)

9376

9377 **【パラメータ】**

9378 uint\_t nombx 割付け可能なメールボックスIDの数

9379

9380 **【エラーコード】**

9381 E\_RSATR 予約属性

9382 ・クラスの囲みの中に記述されていない [M] **【NGKI1953】**

9383 E\_PAR パラメータエラー

9384 ・nombxが負の値 **【NGKI3281】**

9385

9386 **【機能】**

9387

9388 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  
9389 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】**。

9390

9391 nombxは整数定数式パラメータである**【NGKI1955】**。

9392

9393 **【TOPPERS/ASPカーネルにおける規定】**

9394

9395 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  
9396 **【ASPS0215】**。

9397

9398 del\_mbx メールボックスの削除 [TpD] **【NGKI1956】**

9399

9400 **【C言語API】**

9401 ER ercd = del\_mbx(ID mbxid)

9402

9403 **【パラメータ】**

9404 ID mbxid 対象メールボックスのID番号

9405

9406 **【リターンパラメータ】**

9407 ER ercd 正常終了 (E\_OK) またはエラーコード

9408

9409 **【エラーコード】**

9410 E\_CTX コンテキストエラー

9411 ・非タスクコンテキストからの呼出し【NGKI1957】

9412 ・CPUロック状態からの呼出し【NGKI1958】

9413 E\_ID 不正ID番号

9414 ・mbxidが有効範囲外【NGKI1959】

9415 E\_NOEXS オブジェクト未登録

9416 ・対象メールボックスが未登録【NGKI1960】

9417 E\_OBJ オブジェクト状態エラー

9418 ・対象メールボックスは静的APIで生成された【NGKI1961】

9419

9420 **【機能】**

9421

9422 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  
9423 振舞いは以下の通り．

9424

9425 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  
9426 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  
9427 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  
9428 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  
9429 返る【NGKI1964】．

9430

9431 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  
9432 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】．

9433

9434 **【使用上の注意】**

9435

9436 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  
9437 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  
9438 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  
9439 み禁止時間が長くなるため，注意が必要である．

9440

9441 **【TOPPERS/ASPカーネルにおける規定】**

9442

9443 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  
9444 機能拡張パッケージでは，del\_mbxをサポートする【ASPS0157】．

9445

9446 **【TOPPERS/FMPカーネルにおける規定】**

9447

9448 FMPカーネルでは，del\_mbxをサポートしない【FMPS0136】．

9449

9450 snd\_mbx メールボックスへの送信 [Tp] 【NGKI1966】

9451

9452 **【C言語API】**

9453     ER ercd = snd\_mbx(ID mbxid, T\_MSG \*pk\_msg)

9454

9455 **【パラメータ】**

9456     ID           mbxid       対象メールボックスのID番号

9457     T\_MSG        \*pk\_msg     送信メッセージの先頭番地

9458

9459 **【リターンパラメータ】**

9460     ER           ercd        正常終了 (E\_OK) またはエラーコード

9461

9462 **【エラーコード】**

9463     E\_CTX        コンテキストエラー

9464                    ・非タスクコンテキストからの呼出し【NGKI1967】

9465                    ・CPUロック状態からの呼出し【NGKI1968】

9466     E\_ID        不正ID番号

9467                    ・mbxidが有効範囲外【NGKI1969】

9468     E\_PAR        パラメータエラー

9469                    ・条件については機能の項を参照

9470     E\_NOEXS     オブジェクト未登録

9471                    ・対象メールボックスが未登録 [D] 【NGKI1970】

9472

9473 **【機能】**

9474

9475     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

9476     メッセージを送信する。具体的な振舞いは以下の通り。

9477

9478     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

9479     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

9480     【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

9481     からE\_OKが返る【NGKI1972】。

9482

9483     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

9484     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

9485     る順序で、メッセージキューにつながる【NGKI1973】。

9486

9487     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

9488     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、

9489     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

9490     らない。そうでない場合には、E\_PARエラーとなる【NGKI1975】。

9491     -----

9492     rcv\_mbx       メールボックスからの受信 [Tp] 【NGKI1976】

9493     prev\_mbx      メールボックスからの受信 (ポーリング) [Tp] 【NGKI1977】

9494     trcv\_mbx      メールボックスからの受信 (タイムアウト付き) [Tp] 【NGKI1978】

9495

9496 **【C言語API】**

9497     ER ercd = rcv\_mbx(ID mbxid, T\_MSG \*\*ppk\_msg)

9498     ER ercd = prev\_mbx(ID mbxid, T\_MSG \*\*ppk\_msg)

9499     ER ercd = trcv\_mbx(ID mbxid, T\_MSG \*\*ppk\_msg, TMO tmout)

9500

9501      **【パラメータ】**

9502          ID           mbxid       対象メールボックスのID番号

9503          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

9504                                へのポインタ

9505          TMO           tmout       タイムアウト時間（trcv\_mbxの場合）

9506

9507      **【リターンパラメータ】**

9508          ER           ercd        正常終了（E\_OK）またはエラーコード

9509          T\_MSG   \*    ppk\_msg    受信メッセージの先頭番地

9510

9511      **【エラーコード】**

9512          E\_CTX        コンテキストエラー

9513                        ・非タスクコンテキストからの呼出し【NGKI1979】

9514                        ・CPUロック状態からの呼出し【NGKI1980】

9515                        ・ディスパッチ保留状態からの呼出し（prcv\_mbxを除く）【NGKI1981】

9516          E\_NOSPT      未サポート機能

9517                        ・制約タスクからの呼出し（prcv\_mbxを除く）【NGKI1982】

9518          E\_ID         不正ID番号

9519                        ・mbxidが有効範囲外【NGKI1983】

9520          E\_PAR        パラメータエラー

9521                        ・tmoutが無効（trcv\_mbxの場合）【NGKI1984】

9522          E\_NOEXS      オブジェクト未登録

9523                        ・対象メールボックスが未登録〔D〕【NGKI1985】

9524          E\_TMOUT      ポーリング失敗またはタイムアウト（rcv\_mbxを除く）【NGKI1986】

9525          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）

9526                        【NGKI1987】

9527          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）

9528                        【NGKI1988】

9529

9530      **【機能】**

9531

9532      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

9533      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

9534      れる。具体的な振舞いは以下の通り。

9535

9536      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

9537      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

9538      指定したメモリ領域に返される【NGKI1989】。

9539

9540      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

9541      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

9542      クスの待ち行列につながれる【NGKI1990】。

9543      -----

9544      ini\_mbx        メールボックスの再初期化〔Tp〕【NGKI1991】

9545

9546      **【C言語API】**

9547          ER ercd = ini\_mbx(ID mbxid)

9548

9549      **【パラメータ】**

9550          ID           mbxid        対象メールボックスのID番号

9551  
 9552 **【リターンパラメータ】**  
 9553 ER ercd 正常終了 (E\_OK) またはエラーコード  
 9554  
 9555 **【エラーコード】**  
 9556 E\_CTX コンテキストエラー  
 9557 ・非タスクコンテキストからの呼出し【NGKI1992】  
 9558 ・CPUロック状態からの呼出し【NGKI1993】  
 9559 E\_ID 不正ID番号  
 9560 ・mbxidが有効範囲外【NGKI1994】  
 9561 E\_NOEXS オブジェクト未登録  
 9562 ・対象メールボックスが未登録 [D] 【NGKI1995】  
 9563  
 9564 **【機能】**  
 9565  
 9566 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  
 9567 的な振舞いは以下の通り。  
 9568  
 9569 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  
 9570 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  
 9571 ルボックスの待ち行列につながれたタスクは、待ち行列の先頭のタスクから順  
 9572 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  
 9573 たサービスコールからE\_DLTエラーが返る【NGKI1998】。  
 9574  
 9575 **【使用上の注意】**  
 9576  
 9577 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  
 9578 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  
 9579 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  
 9580 み禁止時間が長くなるため、注意が必要である。  
 9581  
 9582 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、  
 9583 アプリケーションの責任である。  
 9584  
 9585 **【μITRON4.0仕様との関係】**  
 9586  
 9587 μITRON4.0仕様に定義されていないサービスコールである。  
 9588 -----  
 9589 ref\_mbx メールボックスの状態参照 [Tp] 【NGKI1999】  
 9590  
 9591 **【C言語API】**  
 9592 ER ercd = ref\_mbx(ID mbxid, T\_RMBX \*pk\_rmbx)  
 9593  
 9594 **【パラメータ】**  
 9595 ID mbxid 対象メールボックスのID番号  
 9596 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  
 9597 のポインタ  
 9598  
 9599 **【リターンパラメータ】**  
 9600 ER ercd 正常終了 (E\_OK) またはエラーコード



9601

9602       \* メールボックスの現在状態 (パケットの内容)

9603       ID           wtskid       メールボックスの待ち行列の先頭のタスクのID

9604                               番号

9605       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

9606                               ジの先頭番地

9607

9608       【エラーコード】

9609       E\_CTX       コンテキストエラー

9610                • 非タスクコンテキストからの呼出し【NGKI2000】

9611                • CPUロック状態からの呼出し【NGKI2001】

9612       E\_ID       不正ID番号

9613                • mbxidが有効範囲外【NGKI2002】

9614       E\_NOEXS     オブジェクト未登録

9615                • 対象メールボックスが未登録 [D] 【NGKI2003】

9616

9617       【機能】

9618

9619       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.

9620       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】.

9621

9622       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

9623       TSK\_NONE (=0) が返る【NGKI2005】. また, メッセージキューにメッセージが

9624       つながれていない場合, pk\_msgにはNULLが返る【NGKI2006】.

9625

9626       【使用上の注意】

9627

9628       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

9629       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

9630       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

9631       クスの状態が変化している可能性があるためである.

9632       -----

9633

9634       4.4.6 ミューテックス

9635

9636       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

9637       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

9638       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

9639       呼ぶID番号によって識別する【NGKI2007】.

9640

9641       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

9642       優先度上限プロトコル (priority ceiling protocol) をサポートする. ミュー

9643       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

9644       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

9645       が行われる.

9646

9647       各ミューテックスが持つ情報は次の通り【NGKI2008】.

9648

9649       • ミューテックス属性

9650       • 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)

9651       ・ ミューテックスをロックしているタスク  
 9652       ・ 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）  
 9653       ・ 上限優先度（優先度上限ミューテックスの場合）  
 9654       ・ アクセス許可ベクタ（保護機能対応カーネルの場合）  
 9655       ・ 属する保護ドメイン（保護機能対応カーネルの場合）  
 9656       ・ 属するクラス（マルチプロセッサ対応カーネルの場合）  
 9657  
 9658       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  
 9659       クスをロック待ち状態）のタスクが、ミューテックスをロックできる順序でつ  
 9660       ながれているキューである。  
 9661  
 9662       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  
 9663       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  
 9664       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  
 9665       【NGKI2009】。  
 9666  
 9667       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】。  
 9668  
 9669               TA\_TPRI       0x01U   待ち行列をタスクの優先度順にする  
 9670               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  
 9671                               クの優先度順にする  
 9672  
 9673       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  
 9674       【NGKI2011】。  
 9675  
 9676       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】。  
 9677  
 9678       ・ ロックしているミューテックスのリスト  
 9679  
 9680       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  
 9681       る【NGKI2013】。  
 9682  
 9683       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  
 9684       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  
 9685       される【NGKI2014】。  
 9686  
 9687       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  
 9688       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  
 9689       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  
 9690       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  
 9691       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  
 9692       【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  
 9693       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  
 9694       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  
 9695       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  
 9696       の中で最後になる【NGKI2018】。  
 9697  
 9698       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  
 9699       クがロックしているミューテックスのロック解除がある。タスクの終了時にロッ  
 9700       クしているミューテックスが残っている場合、それらのミューテックスは、ロッ

9701 クしたのと逆の順序でロック解除される【NGKI2019】。

9702

9703 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。

9704

9705 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  
9706 ネルでは、静的APIによって登録されたミューテックスの  
9707 数に一致）【NGKI2020】

9708

9709 【使用上の注意】

9710

9711 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  
9712 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  
9713 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  
9714 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  
9715 与しない場合に得られる利点である。

9716

9717 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  
9718 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  
9719 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、  
9720 タスクがロック待ち状態になる状況に注意が必要である（優先度上限プロトコ  
9721 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）。

9722

9723 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  
9724 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  
9725 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  
9726 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、  
9727 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。  
9728 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  
9729 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  
9730 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  
9731 クAが実行開始後に、タスクBとタスクCの排他制御区間にブロックされることに  
9732 なる。

9733

9734 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  
9735 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  
9736 らないploc\_mtxを用いるのが安全である。

9737

9738 【補足説明】

9739

9740 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  
9741 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  
9742 でもpriority ceiling protocolと呼ばれているが、学术论文や他のOSでは、  
9743 immediate ceiling priority protocol, priority protection protocol,  
9744 priority ceiling emulation, highest locker protocolなどと呼ばれている。

9745

9746 【TOPPERS/ASPカーネルにおける規定】

9747

9748 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、  
9749 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  
9750 ることができる【ASPS0159】。

9751  
 9752       【TOPPERS/FMPカーネルにおける規定】  
 9753  
 9754 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】。  
 9755  
 9756       【TOPPERS/HRP2カーネルにおける規定】  
 9757  
 9758 HRP2カーネルでは、ミューテックス機能をサポートする【HRPS0131】。  
 9759  
 9760       【未決定事項】  
 9761  
 9762 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  
 9763 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  
 9764 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  
 9765 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  
 9766 課題である。  
 9767  
 9768       【 $\mu$  ITRON4.0仕様との関係】  
 9769  
 9770  $\mu$  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  
 9771 はサポートしていない。また、 $\mu$  ITRON4.0仕様でサポートしている優先度継承  
 9772 プロトコル (priority inheritance protocol) は、現時点ではサポートしてい  
 9773 ない。  
 9774  
 9775 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、  
 9776  $\mu$  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。  
 9777  
 9778 TNUM\_MTXIDは、 $\mu$  ITRON4.0仕様に規定されていないカーネル構成マクロであ  
 9779 る。  
 9780 -----  
 9781 CRE\_MTX       ミューテックスの生成 [S]   【NGKI2021】  
 9782 acre\_mtx       ミューテックスの生成 [TD]   【NGKI2022】  
 9783  
 9784       【静的API】  
 9785       CRE\_MTX(ID mtxid, { ATR mtxatr, PRI ceilpri })  
 9786  
 9787       【C言語API】  
 9788       ER\_ID mtxid = acre\_mtx(const T\_CMTX \*pk\_cmtx)  
 9789  
 9790       【パラメータ】  
 9791       ID           mtxid           生成するミューテックスのID番号 (CRE\_MTXの  
 9792                                    場合)  
 9793       T\_CMTX \*    pk\_cmtx       ミューテックスの生成情報を入れたパケット  
 9794                                    へのポインタ (静的APIを除く)  
 9795  
 9796       \* ミューテックスの生成情報 (パケットの内容)  
 9797       ATR           mtxatr       ミューテックス属性  
 9798       PRI           ceilpri      ミューテックスの上限優先度  
 9799  
 9800       【リターンパラメータ】

9801	ER_ID	mtxid	生成されたミューテックスのID番号（正の値）
9802			またはエラーコード
9803			
9804	【エラーコード】		
9805	E_CTX	コンテキストエラー	
9806		・非タスクコンテキストからの呼出し [s]	【NGKI2023】
9807		・CPUロック状態からの呼出し [s]	【NGKI2024】
9808	E_RSATR	予約属性	
9809		・mtxatrが無効	【NGKI2025】
9810		・属する保護ドメインの指定が有効範囲外 [sP]	【NGKI2026】
9811		・属するクラスの指定が有効範囲外 [sM]	【NGKI2027】
9812		・クラスの囲みの中に記述されていない [SM]	【NGKI2028】
9813	E_PAR	パラメータエラー	
9814		・条件については機能の項を参照	
9815	E_OACV	オブジェクトアクセス違反	
9816		・システム状態に対する管理操作が許可されていない [sP]	
9817		【NGKI2029】	
9818	E_MACV	メモリアクセス違反	
9819		・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	
9820		いない [sP]	【NGKI2030】
9821	E_NOID	ID番号不足	
9822		・割り付けられるミューテックスIDがない [sD]	【NGKI2031】
9823	E_OBJ	オブジェクト状態エラー	
9824		・mtxidで指定したセマフォが登録済み（CRE_MTXの場合）	【NGKI2032】
9825			
9826	【機能】		
9827			
9828	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		
9829	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		
9830	に、待ち行列は空の状態に初期化される【NGKI2033】。		
9831			
9832	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		
9833	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、		
9834	ceilpriの指定を省略することができる【NGKI2035】。		
9835			
9836	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、		
9837	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		
9838	【NGKI2037】。		
9839			
9840	【TOPPERS/ASPカーネルにおける規定】		
9841			
9842	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		
9843	する【ASPS0160】。		
9844			
9845	【TOPPERS/HRP2カーネルにおける規定】		
9846			
9847	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		
9848	成機能拡張パッケージでは、acre_mtxもサポートする【HRPS0194】。		
9849	-----		
9850	AID_MTX	割付け可能なミューテックスIDの数の指定 [SD]	【NGKI2038】

9851  
 9852 **【静的API】**  
 9853     AID\_MTX(uint\_t nomtx)  
 9854  
 9855 **【パラメータ】**  
 9856     uint\_t            nomtx           割付け可能なミューテックスIDの数  
 9857  
 9858 **【エラーコード】**  
 9859     E\_RSATR           予約属性  
 9860                        ・保護ドメインの囲みの中に記述されている [P] 【NGKI3433】  
 9861                        ・クラスの囲みの中に記述されていない [M] 【NGKI2039】  
 9862     E\_PAR             パラメータエラー  
 9863                        ・nomtxが負の値 【NGKI3282】  
 9864  
 9865 **【機能】**  
 9866  
 9867 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  
 9868 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】。  
 9869  
 9870 nomtxは整数定数式パラメータである 【NGKI2041】。  
 9871  
 9872 **【TOPPERS/HRP2カーネルにおける規定】**  
 9873  
 9874 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  
 9875 **【HRPS0216】**。  
 9876 -----  
 9877 SAC\_MTX           ミューテックスのアクセス許可ベクタの設定 [SP] 【NGKI2042】  
 9878 sac\_mtx           ミューテックスのアクセス許可ベクタの設定 [TPD] 【NGKI2043】  
 9879  
 9880 **【静的API】**  
 9881     SAC\_MTX(ID mtxid, { ACPTN acptn1, ACPTN acptn2,  
 9882                                                           ACPTN acptn3, ACPTN acptn4 })  
 9883  
 9884 **【C言語API】**  
 9885     ER ercd = sac\_mtx(ID mtxid, const ACVCT \*p\_acvct)  
 9886  
 9887 **【パラメータ】**  
 9888     ID                mtxid           対象ミューテックスのID番号  
 9889     ACVCT \*           p\_acvct        アクセス許可ベクタを入れたパケットへのポ  
 9890                                       インタ（静的APIを除く）  
 9891  
 9892     \*アクセス許可ベクタ（パケットの内容）  
 9893     ACPTN            acptn1           通常操作1のアクセス許可パターン  
 9894     ACPTN            acptn2           通常操作2のアクセス許可パターン  
 9895     ACPTN            acptn3           管理操作のアクセス許可パターン  
 9896     ACPTN            acptn4           参照操作のアクセス許可パターン  
 9897  
 9898 **【リターンパラメータ】**  
 9899     ER                ercd            正常終了（E\_OK）またはエラーコード  
 9900

9901       **【エラーコード】**  
9902       E\_CTX       コンテキストエラー  
9903                ・非タスクコンテキストからの呼出し [s]   【NGKI2044】  
9904                ・CPUロック状態からの呼出し [s]   【NGKI2045】  
9905       E\_ID       不正ID番号  
9906                ・mtxidが有効範囲外 [s]   【NGKI2046】  
9907       E\_RSATR     予約属性  
9908                ・対象ミューテックスが属する保護ドメインの囲みの中に記  
9909                述されていない [S]   【NGKI2047】  
9910                ・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  
9911                ていない [SM]   【NGKI2048】  
9912       E\_NOEXS     オブジェクト未登録  
9913                ・対象ミューテックスが未登録 【NGKI2049】  
9914       E\_OACV     オブジェクトアクセス違反  
9915                ・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]  
9916                【NGKI2050】  
9917       E\_MACV     メモリアクセス違反  
9918                ・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  
9919                いない [s]   【NGKI2051】  
9920       E\_OBJ     オブジェクト状態エラー  
9921                ・対象ミューテックスは静的APIで生成された [s]   【NGKI2052】  
9922                ・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  
9923                み [S]   【NGKI2053】  
9924  
9925       **【機能】**  
9926  
9927       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  
9928       （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  
9929       【NGKI2054】。  
9930  
9931       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  
9932       式パラメータである 【NGKI2055】。  
9933  
9934       **【TOPPERS/HRP2カーネルにおける規定】**  
9935  
9936       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  
9937       成機能拡張パッケージでは、sac\_mtxもサポートする 【HRPS0195】。  
9938       -----  
9939       del\_mtx       ミューテックスの削除 [TD]   【NGKI2056】  
9940  
9941       **【C言語API】**  
9942       ER ercd = del\_mtx(ID mtxid)  
9943  
9944       **【パラメータ】**  
9945       ID           mtxid       対象ミューテックスのID番号  
9946  
9947       **【リターンパラメータ】**  
9948       ER           ercd       正常終了 (E\_OK) またはエラーコード  
9949  
9950       **【エラーコード】**

9951	E_CTX	コンテキストエラー
9952		・非タスクコンテキストからの呼出し【NGKI2057】
9953		・CPUロック状態からの呼出し【NGKI2058】
9954	E_ID	不正ID番号
9955		・mtxidが有効範囲外【NGKI2059】
9956	E_NOEXS	オブジェクト未登録
9957		・対象ミューテックスが未登録【NGKI2060】
9958	E_OACV	オブジェクトアクセス違反
9959		・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】
9960		【NGKI2061】
9961	E_OBJ	オブジェクト状態エラー
9962		・対象ミューテックスは静的APIで生成された【NGKI2062】
9963		
9964	【機能】	
9965		
9966	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	
9967	振舞いは以下の通り．	
9968		
9969	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	
9970	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	
9971	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	
9972	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に変更される	
9973	【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，	
9974	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	
9975	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	
9976	【NGKI2066】．	
9977		
9978	【使用上の注意】	
9979		
9980	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	
9981	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	
9982	る．これが不都合な場合には，ミューテックスをロックした状態で，ミューテッ	
9983	クスを削除すればよい．	
9984		
9985	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	
9986	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	
9987	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	
9988	み禁止時間が長くなるため，注意が必要である．	
9989		
9990	【TOPPERS/ASPカーネルにおける規定】	
9991		
9992	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	
9993	ない【ASPS0162】．	
9994		
9995	【TOPPERS/HRP2カーネルにおける規定】	
9996		
9997	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	
9998	機能拡張パッケージでは，del_mtxをサポートする【HRPS0196】．	
9999	-----	
10000	loc_mtx	ミューテックスのロック【T】【NGKI2067】



10001 ploc\_mtx ミューテックスのロック (ポーリング) [T] 【NGKI2068】  
 10002 tloc\_mtx ミューテックスのロック (タイムアウト付き) [T] 【NGKI2069】  
 10003  
 10004 **【C言語API】**  
 10005 ER ercd = loc\_mtx(ID mtxid)  
 10006 ER ercd = ploc\_mtx(ID mtxid)  
 10007 ER ercd = tloc\_mtx(ID mtxid, TMO tmout)  
 10008  
 10009 **【パラメータ】**  
 10010 ID mtxid 対象ミューテックスのID番号  
 10011 TMO tmout タイムアウト時間 (tloc\_mtxの場合)  
 10012  
 10013 **【リターンパラメータ】**  
 10014 ER ercd 正常終了 (E\_OK) またはエラーコード  
 10015  
 10016 **【エラーコード】**  
 10017 E\_CTX コンテキストエラー  
 10018 ・非タスクコンテキストからの呼出し 【NGKI2070】  
 10019 ・CPUロック状態からの呼出し 【NGKI2071】  
 10020 ・ディスパッチ保留状態からの呼出し (ploc\_mtxを除く) 【NGKI2072】  
 10021 E\_NOSPT 未サポート機能  
 10022 ・制約タスクからの呼出し (ploc\_mtxを除く) 【NGKI2073】  
 10023 E\_ID 不正ID番号  
 10024 ・mtxidが有効範囲外 【NGKI2074】  
 10025 E\_PAR パラメータエラー  
 10026 ・tmoutが無効 (tloc\_mtxの場合) 【NGKI2075】  
 10027 E\_NOEXS オブジェクト未登録  
 10028 ・対象ミューテックスが未登録 [D] 【NGKI2076】  
 10029 E\_OACV オブジェクトアクセス違反  
 10030 ・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]  
 10031 【NGKI2077】  
 10032 E\_ILUSE サービスコール不正使用  
 10033 ・条件については機能の項を参照  
 10034 E\_OBJ オブジェクト状態エラー  
 10035 ・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  
 10036 【NGKI3609】  
 10037 E\_TMOUT ポーリング失敗またはタイムアウト (loc\_mtxを除く) 【NGKI2078】  
 10038 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)  
 10039 【NGKI2079】  
 10040 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)  
 10041 【NGKI2080】  
 10042  
 10043 **【機能】**  
 10044  
 10045 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  
 10046 な振舞いは以下の通り.  
 10047  
 10048 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  
 10049 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  
 10050 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

10051 優先度が変更される【NGKI2082】.

10052

10053 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  
10054 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  
10055 の待ち行列につながる【NGKI2083】.

10056

10057 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  
10058 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】.

10059

10060 【仕様変更の経緯】

10061

10062 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  
10063 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  
10064 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  
10065 ントロックを実現できるようにするためである.

10066

10067 unl\_mtx ミューテックスのロック解除 [T] 【NGKI2086】

10068

10069 【C言語API】

10070 ER ercd = unl\_mtx(ID mtxid)

10071

10072 【パラメータ】

10073 ID mtxid 対象ミューテックスのID番号

10074

10075 【リターンパラメータ】

10076 ER ercd 正常終了 (E\_OK) またはエラーコード

10077

10078 【エラーコード】

10079 E\_CTX コンテキストエラー

10080 ・非タスクコンテキストからの呼出し【NGKI2087】

10081 ・CPUロック状態からの呼出し【NGKI2088】

10082 E\_ID 不正ID番号

10083 ・mtxidが有効範囲外【NGKI2089】

10084 E\_NOEXS オブジェクト未登録

10085 ・対象ミューテックスが未登録 [D] 【NGKI2090】

10086 E\_OACV オブジェクトアクセス違反

10087 ・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]

10088 【NGKI3273】

10089 E\_OBJ オブジェクト状態エラー

10090 ・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  
10091 い【NGKI3610】

10092

10093 【機能】

10094

10095 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  
10096 体的な振舞いは以下の通り.

10097

10098 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  
10099 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

10100 【NGKI2091】.

10101  
 10102 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  
 10103 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  
 10104 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  
 10105 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  
 10106 加され、必要な場合にはそのタスクの現在優先度の変更される【NGKI2094】。  
 10107 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  
 10108 【NGKI2095】。  
 10109  
 10110 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  
 10111 いない状態になる【NGKI2096】。  
 10112 -----  
 10113 ini\_mtx ミューテックスの再初期化 [T] 【NGKI2098】  
 10114  
 10115 【C言語API】  
 10116 ER ercd = ini\_mtx(ID mtxid)  
 10117  
 10118 【パラメータ】  
 10119 ID mtxid 対象ミューテックスのID番号  
 10120  
 10121 【リターンパラメータ】  
 10122 ER ercd 正常終了 (E\_OK) またはエラーコード  
 10123  
 10124 【エラーコード】  
 10125 E\_CTX コンテキストエラー  
 10126 ・非タスクコンテキストからの呼出し【NGKI2099】  
 10127 ・CPUロック状態からの呼出し【NGKI2100】  
 10128 E\_ID 不正ID番号  
 10129 ・mtxidが有効範囲外【NGKI2101】  
 10130 E\_NOEXS オブジェクト未登録  
 10131 ・対象ミューテックスが未登録 [D] 【NGKI2102】  
 10132 E\_OACV オブジェクトアクセス違反  
 10133 ・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]  
 10134 【NGKI2103】  
 10135  
 10136 【機能】  
 10137  
 10138 mtxidで指定したミューテックス (対象ミューテックス) を再初期化する。具体  
 10139 的な振舞いは以下の通り。  
 10140  
 10141 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  
 10142 【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、  
 10143 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  
 10144 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  
 10145 【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、  
 10146 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  
 10147 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  
 10148 【NGKI2107】。  
 10149  
 10150 【使用上の注意】

10151  
 10152 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  
 10153 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  
 10154 となる。これが不都合な場合には、ミューテックスをロックした状態で、ミュー  
 10155 テックスを再初期化すればよい。  
 10156  
 10157 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  
 10158 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  
 10159 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  
 10160 み禁止時間が長くなるため、注意が必要である。  
 10161  
 10162 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  
 10163 は、アプリケーションの責任である。  
 10164  
 10165 【 $\mu$  ITRON4.0仕様との関係】  
 10166  
 10167  $\mu$  ITRON4.0仕様に定義されていないサービスコールである。  
 10168 -----  
 10169 ref\_mtx ミューテックスの状態参照 [T] 【NGKI2108】  
 10170  
 10171 【C言語API】  
 10172 ER ercd = ref\_mtx(ID mtxid, T\_RMTX \*pk\_rmtx)  
 10173  
 10174 【パラメータ】  
 10175 ID mtxid 対象ミューテックスのID番号  
 10176 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  
 10177 のポインタ  
 10178  
 10179 【リターンパラメータ】  
 10180 ER ercd 正常終了 (E\_OK) またはエラーコード  
 10181  
 10182 \* ミューテックスの現在状態 (パケットの内容)  
 10183 ID htsskid ミューテックスをロックしているタスクのID番号  
 10184 ID wtsskid ミューテックスの待ち行列の先頭のタスクのID  
 10185 番号  
 10186  
 10187 【エラーコード】  
 10188 E\_CTX コンテキストエラー  
 10189 ・非タスクコンテキストからの呼出し 【NGKI2109】  
 10190 ・CPUロック状態からの呼出し 【NGKI2110】  
 10191 E\_ID 不正ID番号  
 10192 ・mtxidが有効範囲外 【NGKI2111】  
 10193 E\_NOEXS オブジェクト未登録  
 10194 ・対象ミューテックスが未登録 [D] 【NGKI2112】  
 10195 E\_OACV オブジェクトアクセス違反  
 10196 ・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]  
 10197 【NGKI2113】  
 10198 E\_MACV メモリアクセス違反  
 10199 ・pk\_rmtxが指すメモリ領域への書込みアクセスが許可されて  
 10200 いない [P] 【NGKI2114】

**【機能】**

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

対象ミューテックスがロックされていない場合、htskidにはTSK\_NONE（=0）が  
返る【NGKI2115】。

対象ミューテックスの待ち行列にタスクが存在しない場合、wtskidには  
TSK\_NONE（=0）が返る【NGKI2116】。

**【使用上の注意】**

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

**4.4.7 メッセージバッファ**

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

各メッセージバッファが持つ情報は次の通り【NGKI3292】。

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

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

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

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  
ている状態（メッセージバッファへの送信待ち状態）のタスクが、メッセージ  
を送信できる順序でつながれているキューである。また、受信待ち行列は、メッ  
セージバッファからメッセージが受信できるまで待っている状態（メッセージ  
バッファからの受信待ち状態）のタスクが、メッセージを受信できる順序でつ

10251 ながれているキューである。

10252

10253 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】。

10254

10255       TA\_TPRI       0x01U   送信待ち行列をタスクの優先度順にする

10256

10257 TA\_TPRIを指定しない場合、送信待ち行列はFIFO順になる【NGKI3296】。受信待ち行列は、FIFO順に固定されている【NGKI3297】。

10259

10260 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。

10261

10262       TNUM\_MBFID       登録できるメッセージバッファの数（動的生成対応でないカーネルでは、静的APIによって登録されたメッセージバッファの数に一致）【NGKI3298】

10265

10266       【TOPPERS/ASPカーネルにおける規定】

10267

10268 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】。ただし、メッセージバッファ機能拡張パッケージを用いると、メッセージバッファ機能を追加することができる【ASPS0203】。

10271

10272       【TOPPERS/FMPカーネルにおける規定】

10273

10274 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】。

10275

10276       【TOPPERS/HRP2カーネルにおける規定】

10277

10278 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】。ただし、メッセージバッファ機能拡張パッケージを用いると、メッセージバッファ機能を追加することができる【HRPS0169】。

10281

10282       【μ ITRON4.0仕様との関係】

10283

10284 TNUM\_MBFID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。

10285

10286 CRE\_MBF       メッセージバッファの生成 [S]   【NGKI3299】

10287 acre\_mbf      メッセージバッファの生成 [TD]   【NGKI3300】

10288

10289       【静的API】

10290       CRE\_MBF(ID mbfid, { ATR mbfattr, uint\_t maxmsz, SIZE mbfsz, void \*mbfmb })

10291

10292       【C言語API】

10293       ER\_ID mbfid = acre\_mbf(const T\_CMBF \*pk\_cmbf)

10294

10295       【パラメータ】

10296       ID           mbfid       生成するメッセージバッファのID番号（CRE\_MBFの場合）

10298       T\_CMBF \*    pk\_cmbf      メッセージバッファの生成情報を入れたパケットへのポインタ（静的APIを除く）

10300

10301	* メッセージバッファの生成情報 (パケットの内容)		
10302	ATR	mbfatr	メッセージバッファ属性
10303	uint_t	maxmsz	メッセージバッファの最大メッセージサイズ (バイト数)
10304			
10305	SIZE	mbfsz	メッセージバッファ管理領域のサイズ (バイト数)
10306	void *	mbfmb	メッセージバッファ管理領域の先頭番地
10307			
10308	【リターンパラメータ】		
10309	ER_ID	mbfid	生成されたメッセージバッファのID番号 (正の値) またはエラーコード
10310			
10311			
10312	【エラーコード】		
10313	E_CTX	コンテキストエラー	
10314		・ 非タスクコンテキストからの呼出し [s] 【NGKI3301】	
10315		・ CPUロック状態からの呼出し [s] 【NGKI3302】	
10316	E_RSATR	予約属性	
10317		・ mbfatrが無効 【NGKI3303】	
10318		・ 属する保護ドメインの指定が有効範囲外 [sP] 【NGKI3304】	
10319		・ 属するクラスの指定が有効範囲外 [sM] 【NGKI3305】	
10320		・ クラスの囲みの中に記述されていない [SM] 【NGKI3306】	
10321	E_NOSPT	未サポート機能	
10322		・ 条件については各カーネルにおける規定の項を参照	
10323	E_PAR	パラメータエラー	
10324		・ maxmszが0以下 【NGKI3307】	
10325		・ mbfszが負の値 [S] 【NGKI3308】	
10326		・ その他の条件については機能の項を参照	
10327	E_OACV	オブジェクトアクセス違反	
10328		・ システム状態に対する管理操作が許可されていない [sP]	
10329		【NGKI3309】	
10330	E_MACV	メモリアクセス違反	
10331		・ pk_cmbfが指すメモリ領域への読出しアクセスが許可されていない [sP] 【NGKI3310】	
10332			
10333	E_NOID	ID番号不足	
10334		・ 割り付けられるメッセージバッファIDがない [sD] 【NGKI3311】	
10335	E_NOMEM	メモリ不足	
10336		・ メッセージバッファ管理領域が確保できない 【NGKI3312】	
10337	E_OBJ	オブジェクト状態エラー	
10338		・ mbfidで指定したメッセージバッファが登録済み (CRE_MBFの場合) 【NGKI3313】	
10339			
10340		・ その他の条件については機能の項を参照	
10341			
10342	【機能】		
10343			
10344	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、格納されているメッセージがない状態に初期化される【NGKI3314】。また、送信待ち行列と受信待ち行列は、空の状態に初期化される【NGKI3315】。		
10345			
10346			
10347			
10348			
10349	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		
10350			

10351 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  
 10352 できない【NGKI3317】。

10353

10354 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  
 10355 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】。

10356 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  
 10357 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  
 10358 【NGKI3319】。

10359

10360 [mbfmbにNULL以外を指定した場合]

10361

10362 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  
 10363 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  
 10364 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  
 10365 用意している【NGKI3321】。

10366

10367	TSZ_MBFMB(msgcnt, msgsz)	msgszで指定したサイズのメッセージを、
10368		msgcntで指定した数だけ格納できるメッセー
10369		ジバッファ管理領域のサイズ (バイト数)
10370	TCNT_MBFMB(msgcnt, msgsz)	msgszで指定したサイズのメッセージを、
10371		msgcntで指定した数だけ格納できるメッセー
10372		ジバッファ管理領域を確保するために必要
10373		なMB_T型の配列の要素数
10374		

10375 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  
 10376 【NGKI3322】。

10377

10378 MB\_T <メッセージバッファ管理領域の変数名>[TCNT\_MBFMB(msgcnt, msgsz)];

10379

10380 この時、mbfszにはTSZ\_MBFMB(msgcnt, msgsz)を、mbfmbには<メッセージバッファ  
 10381 管理領域の変数名>を指定する【NGKI3323】。

10382

10383 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  
 10384 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  
 10385 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  
 10386 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  
 10387 【NGKI3325】。

10388

10389 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  
 10390 理領域のサイズを決めるために用いることができる。

10391

10392 【TOPPERS/ASPカーネルにおける規定】

10393

10394 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  
 10395 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。  
 10396 NULL以外を指定した場合には、E\_NOSPTエラーとなる【ASPS0205】。

10397

10398 【TOPPERS/HRP2カーネルにおける規定】

10399

10400 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



10401 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.  
 10402 NULL以外を指定した場合には, E\_NOSPTエラーとなる【HRPS0171】.

10403

10404 **【 $\mu$  ITRON4.0仕様との関係】**

10405

10406  $\mu$  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  
 10407 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  
 10408 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.

10409

10410 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  
 10411 する方法を規定した.

10412 -----

10413 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】

10414

10415 **【静的API】**

10416 AID\_MBF(uint\_t nombf)

10417

10418 **【パラメータ】**

10419 uint\_t nombf 割付け可能なメッセージバッファIDの数

10420

10421 **【エラーコード】**

10422 E\_RSATR 予約属性

10423 ・保護ドメインの囲みの中に記述されている [P] 【NGKI3434】

10424 ・クラスの囲みの中に記述されていない [M] 【NGKI3327】

10425 E\_PAR パラメータエラー

10426 ・nombfが負の値【NGKI3328】

10427

10428 **【機能】**

10429

10430 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  
 10431 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  
 10432 【NGKI3329】.

10433

10434 nombfは整数定数式パラメータである【NGKI3330】.

10435 -----

10436 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】

10437 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】

10438

10439 **【静的API】**

10440 SAC\_MBF(ID mbfid, { ACPTN acptn1, ACPTN acptn2,  
 10441 ACPTN acptn3, ACPTN acptn4 })

10442

10443 **【C言語API】**

10444 ER ercd = sac\_mbf(ID mbfid, const ACVCT \*p\_acvct)

10445

10446 **【パラメータ】**

10447 ID mbfid 対象メッセージバッファのID番号

10448 ACVCT \* p\_acvct アクセス許可ベクタを入れたパケットへのポ  
 10449 インタ (静的APIを除く)

10450

10451       \*アクセス許可ベクタ (パケットの内容)

10452       ACPTN       acptn1       通常操作1のアクセス許可パターン

10453       ACPTN       acptn2       通常操作2のアクセス許可パターン

10454       ACPTN       acptn3       管理操作のアクセス許可パターン

10455       ACPTN       acptn4       参照操作のアクセス許可パターン

10456

10457       **【リターンパラメータ】**

10458       ER        ercd        正常終了 (E\_OK) またはエラーコード

10459

10460       **【エラーコード】**

10461       E\_CTX       コンテキストエラー

10462               ・非タスクコンテキストからの呼出し [s]   **【NGKI3333】**

10463               ・CPUロック状態からの呼出し [s]   **【NGKI3334】**

10464       E\_ID       不正ID番号

10465               ・mbfidが有効範囲外 [s]   **【NGKI3335】**

10466       E\_RSATR     予約属性

10467               ・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

10468               に記述されていない [S]   **【NGKI3336】**

10469               ・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

10470               されていない [SM]   **【NGKI3337】**

10471       E\_NOEXS     オブジェクト未登録

10472               ・対象メッセージバッファが未登録 **【NGKI3338】**

10473       E\_OACV     オブジェクトアクセス違反

10474               ・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

10475               ない [s]   **【NGKI3339】**

10476       E\_MACV     メモリアクセス違反

10477               ・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

10478               いない [s]   **【NGKI3340】**

10479       E\_OBJ     オブジェクト状態エラー

10480               ・対象メッセージバッファは静的APIで生成された [s]   **【NGKI3341】**

10481               ・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

10482               定済み [S]   **【NGKI3342】**

10483

10484       **【機能】**

10485

10486       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

10487       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

10488       設定する **【NGKI3343】** .

10489

10490       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

10491       式パラメータである **【NGKI3344】** .

10492

10493       **【TOPPERS/HRP2カーネルにおける規定】**

10494

10495       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

10496       ポートする **【HRPS0172】** .

10497       -----

10498       del\_mbf       メッセージバッファの削除 [TD]   **【NGKI3345】**

10499

10500       **【C言語API】**

10501           ER ercd = del\_mbf(ID mbfid)

10502

10503       **【パラメータ】**

10504           ID           mbfid           対象メッセージバッファのID番号

10505

10506       **【リターンパラメータ】**

10507           ER           ercd           正常終了 (E\_OK) またはエラーコード

10508

10509       **【エラーコード】**

10510           E\_CTX       コンテキストエラー

10511                    ・非タスクコンテキストからの呼出し【NGKI3346】

10512                    ・CPUロック状態からの呼出し【NGKI3347】

10513           E\_ID       不正ID番号

10514                    ・mbfidが有効範囲外【NGKI3348】

10515           E\_NOEXS     オブジェクト未登録

10516                    ・対象メッセージバッファが未登録【NGKI3349】

10517           E\_OACV     オブジェクトアクセス違反

10518                    ・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  
10519                    ない [P] 【NGKI3350】

10520           E\_OBJ       オブジェクト状態エラー

10521                    ・対象メッセージバッファは静的APIで生成された【NGKI3351】

10522

10523       **【機能】**

10524

10525       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。  
10526       具体的な振舞いは以下の通り。

10527

10528       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  
10529       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  
10530       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  
10531       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、待ち解  
10532       状態となったサービスコールからE\_DLTエラーが返る【NGKI3354】。

10533

10534       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  
10535       て確保された場合は、そのメモリ領域が解放される【NGKI3355】。

10536

10537       **【補足説明】**

10538

10539       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、  
10540       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  
10541       い。

10542

10543       **【使用上の注意】**

10544

10545       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  
10546       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  
10547       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  
10548       み禁止時間が長くなるため、注意が必要である。

10549

10550       **【TOPPERS/ASPカーネルにおける規定】**

10551  
 10552 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  
 10553 しない【ASPS0207】。  
 10554  
 10555 **【TOPPERS/HRP2カーネルにおける規定】**  
 10556  
 10557 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  
 10558 しない【HRPS0173】。  
 10559 -----  
 10560 snd\_mbf      メッセージバッファへの送信 [T]   【NGKI3356】  
 10561 psnd\_mbf    メッセージバッファへの送信（ポーリング） [T]   【NGKI3357】  
 10562 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】  
 10563  
 10564 **【C言語API】**  
 10565     ER ercd = snd\_mbf(ID mbfid, const void \*msg, uint\_t msgsz)  
 10566     ER ercd = psnd\_mbf(ID mbfid, const void \*msg, uint\_t msgsz)  
 10567     ER ercd = tsnd\_mbf(ID mbfid, const void \*msg, uint\_t msgsz, TMO tmout)  
 10568  
 10569 **【パラメータ】**  
 10570     ID           mbfid           対象メッセージバッファのID番号  
 10571     void \*       msg            送信メッセージの先頭番地  
 10572     uint\_t       msgsz          送信メッセージのサイズ（バイト数）  
 10573     TMO          tmout          タイムアウト時間（tsnd\_mbfの場合）  
 10574  
 10575 **【リターンパラメータ】**  
 10576     ER           ercd           正常終了（E\_OK）またはエラーコード  
 10577  
 10578 **【エラーコード】**  
 10579     E\_CTX        コンテキストエラー  
 10580                  ・非タスクコンテキストからの呼出し【NGKI3359】  
 10581                  ・CPUロック状態からの呼出し【NGKI3360】  
 10582                  ・ディスパッチ保留状態からの呼出し（psnd\_mbfを除く）  
 10583                    【NGKI3361】  
 10584     E\_NOSPT     未サポート機能  
 10585                  ・制約タスクからの呼出し（psnd\_mbfを除く）【NGKI3362】  
 10586     E\_ID        不正ID番号  
 10587                  ・mbfidが有効範囲外【NGKI3363】  
 10588     E\_PAR       パラメータエラー  
 10589                  ・msgszが有効範囲（0より大きく対象メッセージバッファの  
 10590                    最大メッセージサイズ以下）外【NGKI3364】  
 10591                  ・tmoutが無効（tsnd\_mbfの場合）【NGKI3365】  
 10592     E\_NOEXS     オブジェクト未登録  
 10593                  ・対象メッセージバッファが未登録 [D]   【NGKI3366】  
 10594     E\_OACV     オブジェクトアクセス違反  
 10595                  ・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  
 10596                    いない [P]   【NGKI3367】  
 10597     E\_MACV     メモリアクセス違反  
 10598                  ・msgとmsgszが指すメモリ領域への読出しアクセスが許可さ  
 10599                    れていない [P]   【NGKI3368】  
 10600     E\_TMOUT     ポーリング失敗またはタイムアウト（snd\_mbfを除く）【NGKI3369】

10601           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)  
10602                   【NGKI3370】  
10603           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)  
10604                   【NGKI3371】  
10605  
10606       **【機能】**  
10607  
10608       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  
10609       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  
10610       以下の通り.  
10611  
10612       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  
10613       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  
10614       【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  
10615       から, 受信したメッセージのサイズが返る 【NGKI3373】.  
10616  
10617       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  
10618       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  
10619       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,  
10620       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  
10621       【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  
10622       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  
10623       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  
10624       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.  
10625  
10626       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  
10627       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,  
10628       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  
10629       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  
10630       ジバッファの送信待ち行列につながれる 【NGKI3375】.  
10631  
10632       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,  
10633       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  
10634       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  
10635       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  
10636       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  
10637       理領域に空きができた時の処理 [NGKI3393] [NGKI3394] [NGKI3395] と同じ  
10638       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  
10639       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  
10640       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  
10641       る 【NGKI3420】.  
10642  
10643       **【使用上の注意】**  
10644  
10645       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  
10646       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  
10647       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  
10648       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  
10649       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,  
10650       注意が必要である.

```

10651 -----
10652 rcv_mbf      メッセージバッファからの受信 [T] 【NGKI3376】
10653 prcv_mbf     メッセージバッファからの受信（ポーリング） [T] 【NGKI3377】
10654 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】
10655
10656 【C言語API】
10657     ER_UINT msgsz = rcv_mbf(ID mbfid, void *msg)
10658     ER_UINT msgsz = prcv_mbf(ID mbfid, void *msg)
10659     ER_UINT msgsz = trcv_mbf(ID mbfid, void *msg, TMO tmout)
10660
10661 【パラメータ】
10662     ID          mbfid      対象メッセージバッファのID番号
10663     void *      msg        受信メッセージを入れるメモリ領域の先頭番地
10664     TMO         tmout      タイムアウト時間（trcv_mbfの場合）
10665
10666 【リターンパラメータ】
10667     ER_UINT     msgsz      受信メッセージサイズ（正の値）またはエラー
10668                      コード
10669
10670 【エラーコード】
10671     E_CTX       コンテキストエラー
10672                ・非タスクコンテキストからの呼出し 【NGKI3379】
10673                ・CPUロック状態からの呼出し 【NGKI3380】
10674                ・ディスパッチ保留状態からの呼出し（prcv_mbfを除く）
10675                【NGKI3381】
10676     E_NOSPT     未サポート機能
10677                ・制約タスクからの呼出し（prcv_mbfを除く） 【NGKI3382】
10678     E_ID        不正ID番号
10679                ・mbfidが有効範囲外 【NGKI3383】
10680     E_PAR       パラメータエラー
10681                ・tmoutが無効（trcv_mbfの場合） 【NGKI3384】
10682     E_NOEXS     オブジェクト未登録
10683                ・対象メッセージバッファが未登録 [D] 【NGKI3385】
10684     E_OACV      オブジェクトアクセス違反
10685                ・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
10686                  ない [P] 【NGKI3386】
10687     E_MACV      メモリアccess違反
10688                ・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
10689                  ジサイズ分のメモリ領域への書き込みアクセスが許可されて
10690                  いない [P] 【NGKI3387】
10691     E_TMOUT     ポーリング失敗またはタイムアウト（rcv_mbfを除く） 【NGKI3388】
10692     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）
10693                【NGKI3389】
10694     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）
10695                【NGKI3390】
10696
10697 【機能】
10698
10699 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
10700 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

```

10701 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  
 10702 して返される【NGKI3391】。具体的な振舞いは以下の通り。  
 10703  
 10704 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  
 10705 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  
 10706 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  
 10707 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  
 10708 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  
 10709 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  
 10710 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  
 10711 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  
 10712 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  
 10713 くなるまで繰り返す【NGKI3395】。  
 10714  
 10715 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  
 10716 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  
 10717 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  
 10718 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  
 10719 となったサービスコールからE\_OKが返る【NGKI3398】。  
 10720  
 10721 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  
 10722 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  
 10723 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  
 10724 列につながる【NGKI3399】。  
 10725  
 10726 **【使用上の注意】**  
 10727  
 10728 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  
 10729 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  
 10730 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  
 10731 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、  
 10732 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。  
 10733 -----  
 10734 ini\_mbf      メッセージバッファの再初期化 [T] 【NGKI3400】  
 10735  
 10736 **【C言語API】**  
 10737     ER ercd = ini\_mbf(ID mbfid)  
 10738  
 10739 **【パラメータ】**  
 10740     ID           mbfid           対象メッセージバッファのID番号  
 10741  
 10742 **【リターンパラメータ】**  
 10743     ER           ercd           正常終了 (E\_OK) またはエラーコード  
 10744  
 10745 **【エラーコード】**  
 10746     E\_CTX       コンテキストエラー  
 10747                  ・非タスクコンテキストからの呼出し【NGKI3401】  
 10748                  ・CPUロック状態からの呼出し【NGKI3402】  
 10749     E\_ID        不正ID番号  
 10750                  ・mbfidが有効範囲外【NGKI3403】

10751 E\_NOEXS オブジェクト未登録  
 10752 ・対象メッセージバッファが未登録 [D] 【NGKI3404】  
 10753 E\_OACV オブジェクトアクセス違反  
 10754 ・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  
 10755 ない [P] 【NGKI3405】  
 10756  
 10757 **【機能】**  
 10758  
 10759 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  
 10760 る．具体的な振舞いは以下の通り．  
 10761  
 10762 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  
 10763 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  
 10764 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  
 10765 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  
 10766 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】．  
 10767  
 10768 **【補足説明】**  
 10769  
 10770 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、  
 10771 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  
 10772 い．  
 10773  
 10774 **【使用上の注意】**  
 10775  
 10776 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  
 10777 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  
 10778 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  
 10779 み禁止時間が長くなるため、注意が必要である．  
 10780  
 10781 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  
 10782 つのは、アプリケーションの責任である．  
 10783  
 10784 **【 $\mu$  ITRON4.0仕様との関係】**  
 10785  
 10786  $\mu$  ITRON4.0仕様に定義されていないサービスコールである．  
 10787 -----  
 10788 ref\_mbf メッセージバッファの状態参照 [T] 【NGKI3409】  
 10789  
 10790 **【C言語API】**  
 10791 ER ercd = ref\_mbf(ID mbfid, T\_RMBF \*pk\_rmbf)  
 10792  
 10793 **【パラメータ】**  
 10794 ID mbfid 対象メッセージバッファのID番号  
 10795 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  
 10796 トへのポインタ  
 10797  
 10798 **【リターンパラメータ】**  
 10799 ER ercd 正常終了 (E\_OK) またはエラーコード  
 10800



10801       \*メッセージバッファの現在状態（パケットの内容）

10802	ID	stskid	メッセージバッファの送信待ち行列の先頭のタ
10803			スクのID番号
10804	ID	rtskid	メッセージバッファの受信待ち行列の先頭のタ
10805			スクのID番号
10806	uint_t	smbfcnt	メッセージバッファ管理領域に格納されている
10807			メッセージの数
10808	SIZE	fmbfsz	メッセージバッファ管理領域中の空き領域のサ
10809			イズ
10810			
10811	<b>【エラーコード】</b>		
10812	E_CTX	コンテキストエラー	
10813		・非タスクコンテキストからの呼出し【NGKI3410】	
10814		・CPUロック状態からの呼出し【NGKI3411】	
10815	E_ID	不正ID番号	
10816		・mbfidが有効範囲外【NGKI3412】	
10817	E_NOEXS	オブジェクト未登録	
10818		・対象メッセージバッファが未登録 [D] 【NGKI3413】	
10819	E_OACV	オブジェクトアクセス違反	
10820		・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	
10821		ない [P] 【NGKI3414】	
10822	E_MACV	メモリアクセス違反	
10823		・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	
10824		いない [P] 【NGKI3415】	
10825			
10826	<b>【機能】</b>		
10827			
10828	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		
10829	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		
10830	<b>【NGKI3416】</b> ．		
10831			
10832	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		
10833	TSK_NONE（=0）が返る【NGKI3417】．また、受信待ち行列にタスクが存在しな		
10834	い場合、rtskidにはTSK_NONE（=0）が返る【NGKI3418】．		
10835			
10836	<b>【使用上の注意】</b>		
10837			
10838	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		
10839	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		
10840	した直後に割込みが発生した場合、ref_mbfから戻ってきた時には対象メッセー		
10841	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．		
10842	-----		
10843			
10844	4.4.8 スピンロック		
10845			
10846	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		
10847	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		
10848	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		
10849	<b>【NGKI2117】</b> ．		
10850			

10851 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  
10852 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】。  
10853 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  
10854 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  
10855 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  
10856 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、  
10857 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  
10858 を返却するという。

10859

10860 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  
10861 ルーチン、割込みハンドラ、割込みサービスルーチン、タイムイベントハンド  
10862 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  
10863 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  
10864 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  
10865 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  
10866 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  
10867 では、スピンロックは返却されない【NGKI2123】。

10868

10869 各スピンロックが持つ情報は次の通り【NGKI2124】。

10870

- 10871 ・スピンロック属性
- 10872 ・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）
- 10873 ・アクセス許可ベクタ（保護機能対応カーネルの場合）
- 10874 ・属する保護ドメイン（保護機能対応カーネルの場合）
- 10875 ・属するクラス

10876

10877 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  
10878 ク属性には、TA\_NULLを指定しなければならない【NGKI2126】。

10879

10880 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。

10881

10882	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
10883		ネルでは、静的APIによって登録されたスピンロックの数
10884		に一致）【NGKI2127】

10885

10886 【補足説明】

10887

10888 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  
10889 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  
10890 クを取得することはできない。そのため、1つの処理単位が、複数のスピンロッ  
10891 クを取得した状態になることはできない。

10892

10893 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  
10894 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  
10895 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  
10896 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  
10897 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  
10898 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。

10899

10900 【TOPPERS/ASPカーネルにおける規定】

10901  
 10902 ASPカーネルでは、スピンロック機能をサポートしない【ASPS0163】。  
 10903  
 10904 【TOPPERS/FMPカーネルにおける規定】  
 10905  
 10906 FMPカーネルでは、スピンロック機能をサポートする【FMPS0138】。  
 10907  
 10908 【TOPPERS/HRP2カーネルにおける規定】  
 10909  
 10910 HRP2カーネルでは、スピンロック機能をサポートしない【HRPS0135】。  
 10911  
 10912 【 $\mu$  ITRON4.0仕様との関係】  
 10913  
 10914 スピンロック機能は、 $\mu$  ITRON4.0仕様に定義されていない機能である。  
 10915 -----  
 10916 CRE\_SPN スピンロックの生成 [SM] 【NGKI2128】  
 10917 acre\_spn スピンロックの生成 [TMD] 【NGKI2129】  
 10918  
 10919 【静的API】  
 10920 CRE\_SPN(ID spnid, { ATR spnatr })  
 10921  
 10922 【C言語API】  
 10923 ER\_ID spnid = acre\_spn(const T\_CSPN \*pk\_cspn)  
 10924  
 10925 【パラメータ】  
 10926 ID spnid 生成するスピンロックのID番号 (CRE\_SPNの場合)  
 10927 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  
 10928 ポインタ (静的APIを除く)  
 10929  
 10930 \* スピンロックの生成情報 (パケットの内容)  
 10931 ATR spnatr スピンロック属性  
 10932  
 10933 【リターンパラメータ】  
 10934 ER\_ID spnid 生成されたスピンロックのID番号 (正の値) ま  
 10935 たはエラーコード  
 10936  
 10937 【エラーコード】  
 10938 E\_CTX コンテキストエラー  
 10939 ・非タスクコンテキストからの呼出し [s] 【NGKI2130】  
 10940 ・CPUロック状態からの呼出し [s] 【NGKI2131】  
 10941 E\_RSATR 予約属性  
 10942 ・spnatrが無効 【NGKI2132】  
 10943 ・属する保護ドメインの指定が有効範囲外 [sP] 【NGKI2133】  
 10944 ・属するクラスの指定が有効範囲外 [s] 【NGKI2134】  
 10945 ・クラスの囲みの中に記述されていない [S] 【NGKI2135】  
 10946 E\_OACV オブジェクトアクセス違反  
 10947 ・システム状態に対する管理操作が許可されていない [sP]  
 10948 【NGKI2136】  
 10949 E\_MACV メモリアクセス違反  
 10950 ・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

10951                   いない [sP] 【NGKI2137】

10952           E\_NOID   ID番号不足

10953                   ・割り付けられるスピンロックIDがない [sD] 【NGKI2138】

10954           E\_NORES   資源不足

10955                   ・条件については機能の項を参照

10956           E\_OBJ    オブジェクト状態エラー

10957                   ・spnidで指定したスピンロックが登録済み (CRE\_SPNの場合)

10958                   【NGKI2139】

10959

10960       **【機能】**

10961

10962       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

10963       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

10964       化される【NGKI2140】。

10965

10966       静的APIにおいては、spnidはオブジェクト識別名、spnatrは整数定数式パラメー

10967       タである【NGKI2141】。

10968

10969       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

10970       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

10971       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

10972       【NGKI2143】。

10973

10974       **【補足説明】**

10975

10976       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

10977       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

10978       ない。

10979

10980       **【TOPPERS/FMPカーネルにおける規定】**

10981

10982       FMPカーネルでは、CRE\_SPNのみをサポートする【FMPS0139】。

10983       -----

10984       AID\_SPN        割付け可能なスピンロックIDの数の指定 [SMD] 【NGKI2144】

10985

10986       **【静的API】**

10987        AID\_SPN(uint\_t nospn)

10988

10989       **【パラメータ】**

10990        uint\_t        nospn        割付け可能なスピンロックIDの数

10991

10992       **【エラーコード】**

10993        E\_RSATR    予約属性

10994                   ・保護ドメインの囲みの中に記述されている [P] 【NGKI3435】

10995                   ・クラスの囲みの中に記述されていない【NGKI2145】

10996        E\_PAR     パラメータエラー

10997                   ・nospnが負の値【NGKI3283】

10998

10999       **【機能】**

11000

11001 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  
 11002 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】.

11003

11004 nospnは整数定数式パラメータである【NGKI2147】.

11005

11006 SAC\_SPN スピンロックのアクセス許可ベクタの設定 [SPM] 【NGKI2148】

11007 sac\_spn スピンロックのアクセス許可ベクタの設定 [TPMD] 【NGKI2149】

11008

11009 【静的API】

11010 SAC\_SPN(ID spnid, { ACPTN acptn1, ACPTN acptn2,  
 11011 ACPTN acptn3, ACPTN acptn4 })

11012

11013 【C言語API】

11014 ER ercd = sac\_spn(ID spnid, const ACVCT \*p\_acvct)

11015

11016 【パラメータ】

11017 ID spnid 対象スピンロックのID番号  
 11018 ACVCT \* p\_acvct アクセス許可ベクタを入れたパケットへのポ  
 11019 インタ（静的APIを除く）

11020

11021 \*アクセス許可ベクタ（パケットの内容）

11022 ACPTN acptn1 通常操作1のアクセス許可パターン

11023 ACPTN acptn2 通常操作2のアクセス許可パターン

11024 ACPTN acptn3 管理操作のアクセス許可パターン

11025 ACPTN acptn4 参照操作のアクセス許可パターン

11026

11027 【リターンパラメータ】

11028 ER ercd 正常終了 (E\_OK) またはエラーコード

11029

11030 【エラーコード】

11031 E\_CTX コンテキストエラー

11032 ・非タスクコンテキストからの呼出し [s] 【NGKI2150】

11033 ・CPUロック状態からの呼出し [s] 【NGKI2151】

11034 E\_ID 不正ID番号

11035 ・spnidが有効範囲外 [s] 【NGKI2152】

11036 E\_RSATR 予約属性

11037 ・対象スピンロックが属する保護ドメインの囲みの中に記述

11038 されていない [S] 【NGKI2153】

11039 ・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

11040 いない [S] 【NGKI2154】

11041 E\_NOEXS オブジェクト未登録

11042 ・対象スピンロックが未登録【NGKI2155】

11043 E\_OACV オブジェクトアクセス違反

11044 ・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]  
 11045 【NGKI2156】

11046 E\_MACV メモリアクセス違反

11047 ・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  
 11048 いない [s] 【NGKI2157】

11049 E\_OBJ オブジェクト状態エラー

11050 ・対象スピンロックは静的APIで生成された [s] 【NGKI2158】

11051                   ・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]  
11052                   【NGKI2159】

11053  
11054   **【機能】**

11055  
11056   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  
11057   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  
11058   【NGKI2160】.

11059  
11060   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  
11061   式パラメータである【NGKI2161】.

11062 -----  
11063   del\_spn       スピンロックの削除 [TMD]   【NGKI2162】

11064  
11065   **【C言語API】**

11066       ER ercd = del\_spn(ID snpid)

11067  
11068   **【パラメータ】**

11069       ID            spnid            対象スピンロックのID番号

11070  
11071   **【リターンパラメータ】**

11072       ER            ercd            正常終了 (E\_OK) またはエラーコード

11073  
11074   **【エラーコード】**

11075       E\_CTX        コンテキストエラー  
11076                    ・非タスクコンテキストからの呼出し【NGKI2163】  
11077                    ・CPUロック状態からの呼出し【NGKI2164】  
11078       E\_ID         不正ID番号  
11079                    ・spnidが有効範囲外【NGKI2165】  
11080       E\_NOEXS      オブジェクト未登録  
11081                    ・対象スピンロックが未登録【NGKI2166】  
11082       E\_OACV      オブジェクトアクセス違反  
11083                    ・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]  
11084                    【NGKI2167】  
11085       E\_OBJ       オブジェクト状態エラー  
11086                    ・対象スピンロックは静的APIで生成された【NGKI2168】

11087  
11088   **【機能】**

11089  
11090   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  
11091   いは以下の通り．

11092  
11093   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  
11094   される【NGKI2169】.

11095  
11096   **【TOPPERS/FMPカーネルにおける規定】**

11097  
11098   FMPカーネルでは、del\_spnをサポートしない【FMPS0141】.

11099  
11100   **【未決定事項】**

11101  
 11102 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。  
 11103 -----  
 11104 loc\_spn      スピンロックの取得 [TM]      【NGKI2170】  
 11105 iloc\_spn     スピンロックの取得 [IM]      【NGKI2171】  
 11106  
 11107 **【C言語API】**  
 11108     ER ercd = loc\_spn(ID spnid)  
 11109     ER ercd = iloc\_spn(ID spnid)  
 11110  
 11111 **【パラメータ】**  
 11112     ID                  spnid                  対象スピンロックのID番号  
 11113  
 11114 **【リターンパラメータ】**  
 11115     ER                  ercd                  正常終了 (E\_OK) またはエラーコード  
 11116  
 11117 **【エラーコード】**  
 11118     E\_CTX              コンテキストエラー  
 11119                          ・ 非タスクコンテキストからの呼出し (loc\_spnの場合)      【NGKI2172】  
 11120                          ・ タスクコンテキストからの呼出し (iloc\_spnの場合)      【NGKI2173】  
 11121                          ・ CPUロック状態からの呼出し      【NGKI2174】  
 11122     E\_ID                不正ID番号  
 11123                          ・ spnidが有効範囲外      【NGKI2175】  
 11124     E\_NOEXS            オブジェクト未登録  
 11125                          ・ 対象スピンロックが未登録 [D]      【NGKI2176】  
 11126     E\_OACV            オブジェクトアクセス違反  
 11127                          ・ 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  
 11128                                  (loc\_spnの場合) [P]      【NGKI2177】  
 11129  
 11130 **【機能】**  
 11131  
 11132 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  
 11133 いは以下の通り。  
 11134  
 11135 対象スピンロックが取得されていない状態である場合には、プロセッサ間ロッ  
 11136 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  
 11137 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、  
 11138 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  
 11139 みる【NGKI2179】。これを、ロックの取得に成功するまで繰り返す  
 11140 【NGKI2180】。  
 11141  
 11142 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  
 11143 【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、  
 11144 サービスコールからリターンする【NGKI2182】。  
 11145  
 11146 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  
 11147 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】。  
 11148  
 11149 **【補足説明】**  
 11150

11151 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  
 11152 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  
 11153 ているため、loc\_spn/iloc\_spnはE\_CTXエラーとなる。

11154

11155 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、  
 11156 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り込みが発生しな  
 11157 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  
 11158 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  
 11159 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。

11160

11161 try\_spn      スピンロックの取得（ポーリング） [TM] 【NGKI2184】

11162 itry\_spn     スピンロックの取得（ポーリング） [IM] 【NGKI2185】

11163

#### 11164 【C言語API】

11165        ER ercd = try\_spn(ID spnid)

11166        ER ercd = itry\_spn(ID spnid)

11167

#### 11168 【パラメータ】

11169        ID            spnid            対象スピンロックのID番号

11170

#### 11171 【リターンパラメータ】

11172        ER            ercd            正常終了 (E\_OK) またはエラーコード

11173

#### 11174 【エラーコード】

11175        E\_CTX        コンテキストエラー

11176                      ・非タスクコンテキストからの呼出し (try\_spnの場合) 【NGKI2186】

11177                      ・タスクコンテキストからの呼出し (itry\_spnの場合) 【NGKI2187】

11178                      ・CPUロック状態からの呼出し 【NGKI2188】

11179        E\_ID        不正ID番号

11180                      ・spnidが有効範囲外 【NGKI2189】

11181        E\_NOEXS     オブジェクト未登録

11182                      ・対象スピンロックが未登録 [D] 【NGKI2190】

11183        E\_OACV     オブジェクトアクセス違反

11184                      ・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

11185                      (try\_spnの場合) [P] 【NGKI2191】

11186        E\_OBJ        オブジェクト状態エラー

11187                      ・条件については機能の項を参照

11188

#### 11189 【機能】

11190

11191 spnidで指定したスピンロック（対象スピンロック）の取得を試みる。具体的な  
 11192 振舞いは以下の通り。

11193

11194 対象スピンロックが取得されていない状態である場合には、プロセッサ間ロッ  
 11195 クの取得を試みる【NGKI2192】。ロックの取得に成功した場合には、スピンロッ  
 11196 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  
 11197 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】。

11198

11199 対象スピンロックが他のプロセッサによって取得されている状態である場合や、  
 11200 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）



11201 には、E\_OBJエラーとする【NGKI2195】.

11202

11203 **【使用上の注意】**

11204

11205 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  
11206 りスピンロックを取得する方法は、loc\_spn/iloc\_spnによりスピンロックを取  
11207 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.

11208 -----

11209 unl\_spn      スピンロックの返却 [TM]      【NGKI2196】

11210 iunl\_spn      スピンロックの返却 [IM]      【NGKI2197】

11211

11212 **【C言語API】**

11213      ER ercd = unl\_spn(ID spnid)

11214      ER ercd = iunl\_spn(ID spnid)

11215

11216 **【パラメータ】**

11217      ID                  spnid                  対象スピンロックのID番号

11218

11219 **【リターンパラメータ】**

11220      ER                  ercd                  正常終了 (E\_OK) またはエラーコード

11221

11222 **【エラーコード】**

11223      E\_CTX                  コンテキストエラー

11224                  ・非タスクコンテキストからの呼出し (unl\_spnの場合)      【NGKI2198】

11225                  ・タスクコンテキストからの呼出し (iunl\_spnの場合)      【NGKI2199】

11226      E\_ID                  不正ID番号

11227                  ・spnidが有効範囲外【NGKI2200】

11228      E\_NOEXS              オブジェクト未登録

11229                  ・対象スピンロックが未登録 [D]      【NGKI2201】

11230      E\_OACV              オブジェクトアクセス違反

11231                  ・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  
11232                  (unl\_spnの場合) [P]      【NGKI2202】

11233      E\_ILUSE              サービスコール不正使用

11234                  ・条件については機能の項を参照

11235

11236 **【機能】**

11237

11238 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  
11239 いは以下の通り.

11240

11241 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  
11242 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  
11243 いない状態とする【NGKI2203】. また、CPUロックフラグをクリアし、CPUロッ  
11244 ク解除状態へ遷移する【NGKI2204】.

11245

11246 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  
11247 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】.

11248 -----

11249 ref\_spn      スピンロックの状態参照 [TM]      【NGKI2206】

11250

11251 **【C言語API】**  
 11252 ER ercd = ref\_spn(ID spnid, T\_RSPN \*pk\_rspn)  
 11253  
 11254 **【パラメータ】**  
 11255 ID spnid 対象スピンロックのID番号  
 11256 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  
 11257 ポインタ  
 11258  
 11259 **【リターンパラメータ】**  
 11260 ER ercd 正常終了 (E\_OK) またはエラーコード  
 11261  
 11262 \* スピンロックの現在状態 (パケットの内容)  
 11263 STAT spnstat ロック状態  
 11264  
 11265 **【エラーコード】**  
 11266 E\_CTX コンテキストエラー  
 11267 ・非タスクコンテキストからの呼出し【NGKI2207】  
 11268 ・CPUロック状態からの呼出し【NGKI2208】  
 11269 E\_ID 不正ID番号  
 11270 ・spnidが有効範囲外【NGKI2209】  
 11271 E\_NOEXS オブジェクト未登録  
 11272 ・対象スピンロックが未登録 [D] 【NGKI2210】  
 11273 E\_OACV オブジェクトアクセス違反  
 11274 ・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]  
 11275 【NGKI2211】  
 11276 E\_MACV メモリアクセス違反  
 11277 ・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  
 11278 いない [P] 【NGKI2212】  
 11279  
 11280 **【機能】**  
 11281  
 11282 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  
 11283 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】.  
 11284  
 11285 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  
 11286 返される【NGKI2214】.  
 11287  
 11288 TSPN\_UNL 0x01U 取得されていない状態  
 11289 TSPN\_LOC 0x02U 取得されている状態  
 11290  
 11291 **【使用上の注意】**  
 11292  
 11293 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  
 11294 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  
 11295 後に割り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  
 11296 状態が変化している可能性があるためである.  
 11297 -----  
 11298  
 11299 4.5 メモリプール管理機能  
 11300

11301       【TOPPERS/SSPカーネルにおける規定】

11302

11303       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】。

11304

11305       【 $\mu$  ITRON4.0仕様との関係】

11306

11307       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。

11308

11309       【仕様決定の理由】

11310

11311       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  
11312       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  
11313       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  
11314       ラリとして実現する方が適切と考えたためである。

11315

#### 11316       4.5.1 固定長メモリプール

11317

11318       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  
11319       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。

11320       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

11321       【NGKI2215】。

11322

11323       各固定長メモリプールが持つ情報は次の通り【NGKI2216】。

11324

- 11325       ・ 固定長メモリプール属性
- 11326       ・ 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）
- 11327       ・ 固定長メモリプール領域
- 11328       ・ 固定長メモリプール管理領域
- 11329       ・ アクセス許可ベクタ（保護機能対応カーネルの場合）
- 11330       ・ 属する保護ドメイン（保護機能対応カーネルの場合）
- 11331       ・ 属するクラス（マルチプロセッサ対応カーネルの場合）

11332

11333       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  
11334       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  
11335       きる順序でつながれているキューである。

11336

11337       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  
11338       めのメモリ領域である。

11339

11340       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  
11341       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  
11342       のメモリ領域である。

11343

11344       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  
11345       用いるオブジェクト管理領域として扱われる【NGKI2217】。

11346

11347       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】。

11348

11349       TA\_TPRI       0x01U   待ち行列をタスクの優先度順にする

11350

11351 TA\_TPRIを指定しない場合、待ち行列はFIFO順になる【NGKI2219】。

11352

11353 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。

11354

11355 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

11356 カーネルでは、静的APIによって登録された固定長メモリ

11357 プールの数に一致）【NGKI2220】

11358

11359 【μITRON4.0仕様との関係】

11360

11361 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

11362 クロ（TSZ\_MPF）は廃止した。これは、固定長メモリプール領域をアプリケーションで確保する方法を定めた結果、そのサイズは(blkcnt \* ROUND\_MPF\_T(blksz))

11363 で求めることができるようになったためである。

11364

11365

11366 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。

11367 -----

11368 CRE\_MPF 固定長メモリプールの生成 [S] 【NGKI2221】

11369 acre\_mpf 固定長メモリプールの生成 [TD] 【NGKI2222】

11370

11371 【静的API】

11372 CRE\_MPF(ID mpfid, { ATR mpfatr, uint\_t blkcnt, uint\_t blksz,

11373 MPF\_T \*mpf, void \*mpfmb })

11374

11375 【C言語API】

11376 ER\_ID mpfid = acre\_mpf(const T\_CMPF \*pk\_cmpf)

11377

11378 【パラメータ】

11379 ID mpfid 生成する固定長メモリプールのID番号（CRE\_MPF

11380 の場合）

11381 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

11382 トへのポインタ（静的APIを除く）

11383

11384 \* 固定長メモリプールの生成情報（パケットの内容）

11385 ATR mpfatr 固定長メモリプール属性

11386 uint\_t blkcnt 獲得できる固定長メモリブロックの数

11387 uint\_t blksz 固定長メモリブロックのサイズ（バイト数）

11388 MPF\_T \* mpf 固定長メモリプール領域の先頭番地

11389 void \* mpfmb 固定長メモリプール管理領域の先頭番地

11390

11391 【リターンパラメータ】

11392 ER\_ID mpfid 生成された固定長メモリプールのID番号（正の

11393 値）またはエラーコード

11394

11395 【エラーコード】

11396 E\_CTX コンテキストエラー

11397 ・非タスクコンテキストからの呼出し [s] 【NGKI2223】

11398 ・CPUロック状態からの呼出し [s] 【NGKI2224】

11399 E\_RSATR 予約属性

11400 ・mpfatrが無効【NGKI2225】

11401		・ 属する保護ドメインの指定が有効範囲外 [sP] 【NGKI2226】
11402		・ 属するクラスの指定が有効範囲外 [sM] 【NGKI2227】
11403		・ クラスの囲みの中に記述されていない [SM] 【NGKI2228】
11404	E_NOSPT	未サポート機能
11405		・ 条件については各カーネルにおける規定の項を参照
11406	E_PAR	パラメータエラー
11407		・ blkcntが0以下 【NGKI2229】
11408		・ blkkszが0以下 【NGKI2230】
11409		・ その他の条件については機能の項を参照
11410	E_OACV	オブジェクトアクセス違反
11411		・ システム状態に対する管理操作が許可されていない [sP]
11412		【NGKI2231】
11413	E_MACV	メモリアクセス違反
11414		・ 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
11415		いない [sP] 【NGKI2232】
11416	E_NOID	ID番号不足
11417		・ 割り付けられる固定長メモリプールIDがない [sD] 【NGKI2233】
11418	E_NOMEM	メモリ不足
11419		・ 固定長メモリプール領域が確保できない 【NGKI2234】
11420		・ 固定長メモリプール管理領域が確保できない 【NGKI2235】
11421	E_OBJ	オブジェクト状態エラー
11422		・ mpfidで指定した固定長メモリプールが登録済み (CRE_MPF
11423		の場合) 【NGKI2236】
11424		・ その他の条件については機能の項を参照
11425		
11426	【機能】	
11427		
11428		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
11429		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、
11430		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
11431		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
11432		空の状態に初期化される【NGKI2238】。
11433		
11434		静的APIにおいては、mpfidはオブジェクト識別名、mpfatr, blkcnt, blkkszは整
11435		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】。
11436		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
11437		とができない【NGKI2240】。
11438		
11439		mpfをNULLとした場合、blkcntとblkkszから決まるサイズの固定長メモリプール
11440		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】。
11441		
11442		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
11443		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
11444		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
11445		ト中に確保される【NGKI2242】。
11446		
11447		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
11448		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】。
11449		
11450		[mpfにNULL以外を指定した場合]

11451  
 11452 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  
 11453 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  
 11454 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  
 11455 意している【NGKI2245】。  
 11456  
 11457 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  
 11458  
 11459 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  
 11460 リプール領域を確保するために、固定長メモリブロッ  
 11461 ク1つあたりに必要なMPF\_T型の配列の要素数  
 11462 ROUND\_MPF\_T(blksz) 要素数COUNT\_MPF\_T(blksz)のMPF\_T型の配列のサイズ  
 11463 (blkszを、MPF\_T型のサイズの倍数になるように大き  
 11464 い方に丸めた値)  
 11465  
 11466 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】。  
 11467  
 11468 MPF\_T <固定長メモリプール領域の変数名>[(blkcnt) \* COUNT\_MPF\_T(blksz)];  
 11469  
 11470 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】。  
 11471  
 11472 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  
 11473 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  
 11474 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  
 11475 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  
 11476 【NGKI2249】。  
 11477  
 11478 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  
 11479 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。  
 11480 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  
 11481 ジェクトに含まれていない場合、E\_OBJエラーとなる【NGKI2251】。  
 11482  
 11483 [mpfmbにNULL以外を指定した場合]  
 11484  
 11485 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  
 11486 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  
 11487 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  
 11488 用意している【NGKI2253】。  
 11489  
 11490 TSZ\_MPFMB(blkcnt) blkcntで指定した数の固定長メモリブロックを管理  
 11491 することができる固定長メモリプール管理領域のサ  
 11492 イズ (バイト数)  
 11493 TCNT\_MPFMB(blkcnt) blkcntで指定した数の固定長メモリブロックを管理  
 11494 することができる固定長メモリプール管理領域を確  
 11495 保するために必要なMB\_T型の配列の要素数  
 11496  
 11497 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  
 11498 【NGKI2254】。  
 11499  
 11500 MB\_T <固定長メモリプール管理領域の変数名>[TCNT\_MPFMB(blkcnt)];

11501  
11502 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  
11503 【NGKI2255】。  
11504  
11505 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  
11506 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  
11507 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  
11508 オブジェクトに含まれない場合、E\_OBJエラーとなる【NGKI2257】。  
11509  
11510 【補足説明】  
11511  
11512 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  
11513 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  
11514 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  
11515 ンの責任である。  
11516  
11517 【TOPPERS/ASPカーネルにおける規定】  
11518  
11519 ASPカーネルでは、CRE\_MPFのみをサポートする【ASPS0164】。また、mpfmbには  
11520 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  
11521 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  
11522 サポートする【ASPS0167】。acre\_mpfに対しては、mpfmbにNULL以外を指定でき  
11523 ないという制限はない【ASPS0168】。  
11524  
11525 【TOPPERS/FMPカーネルにおける規定】  
11526  
11527 FMPカーネルでは、CRE\_MPFのみをサポートする【FMPS0142】。また、mpfmbには  
11528 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  
11529 となる【FMPS0144】。  
11530  
11531 【TOPPERS/HRP2カーネルにおける規定】  
11532  
11533 HRP2カーネルでは、CRE\_MPFのみをサポートする【HRPS0136】。また、mpfmbに  
11534 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  
11535 ラーとなる【HRPS0138】。  
11536  
11537 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】。  
11538 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  
11539 【HRPS0198】。ただし、mpfにNULLが指定されるとカーネルが固定長メモリプ  
11540 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、  
11541 E\_NOSPTエラーとなる【HRPS0199】。  
11542  
11543 【μITRON4.0仕様との関係】  
11544  
11545 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  
11546 定長メモリプール領域をアプリケーションで確保する方法を規定した。また、  
11547 μITRON4.0/PX仕様にあわせて、固定長メモリプール生成情報に、mpfmbを追加  
11548 した。  
11549  
11550 【μITRON4.0/PX仕様との関係】

11551  
 11552 TCNT\_MPFMBを新設し、固定長メモリプール管理領域をアプリケーションで確保  
 11553 する方法を規定した。  
 11554 -----

11555 AID\_MPF 割付け可能な固定長メモリプールIDの数の指定 [SD] 【NGKI2258】  
 11556

11557 【静的API】  
 11558 AID\_MPF(uint\_t nompf)  
 11559

11560 【パラメータ】  
 11561 uint\_t nompf 割付け可能な固定長メモリプールIDの数  
 11562

11563 【エラーコード】  
 11564 E\_RSATR 予約属性  
 11565 ・保護ドメインの囲みの中に記述されている [P] 【NGKI3436】  
 11566 ・クラスの囲みの中に記述されていない [M] 【NGKI2259】  
 11567 E\_PAR パラメータエラー  
 11568 ・nompfが負の値 【NGKI3284】  
 11569

11570 【機能】  
 11571  
 11572 nompfで指定した数の固定長メモリプールIDを、固定長メモリプールを生成する  
 11573 サービスコールによって割付け可能な固定長メモリプールIDとして確保する  
 11574 【NGKI2260】。  
 11575  
 11576 nompfは整数定数式パラメータである 【NGKI2261】。  
 11577

11578 【TOPPERS/ASPカーネルにおける規定】  
 11579  
 11580 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  
 11581 【ASPS0216】。  
 11582

11583 【TOPPERS/HRP2カーネルにおける規定】  
 11584  
 11585 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  
 11586 【HRPS0217】。  
 11587 -----

11588 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】  
 11589 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】  
 11590

11591 【静的API】  
 11592 SAC\_MPF(ID mpfid, { ACPTN acptn1, ACPTN acptn2,  
 11593 ACPTN acptn3, ACPTN acptn4 })  
 11594

11595 【C言語API】  
 11596 ER ercd = sac\_mpf(ID mpfid, const ACVCT \*p\_acvct)  
 11597

11598 【パラメータ】  
 11599 ID mpfid 対象固定長メモリプールのID番号  
 11600 ACVCT \* p\_acvct アクセス許可ベクタを入れたパケットへのポ



11601

11602

11603

11604

11605

11606

11607

11608

11609

11610

11611

11612

11613

11614

11615

11616

11617

11618

11619

11620

11621

11622

11623

11624

11625

11626

11627

11628

11629

11630

11631

11632

11633

11634

11635

11636

11637

11638

11639

11640

11641

11642

11643

11644

11645

11646

11647

11648

11649

11650

インタ（静的APIを除く）

\*アクセス許可ベクタ（パケットの内容）

ACPTN acptn1 通常操作1のアクセス許可パターン

ACPTN acptn2 通常操作2のアクセス許可パターン

ACPTN acptn3 管理操作のアクセス許可パターン

ACPTN acptn4 参照操作のアクセス許可パターン

【リターンパラメータ】

ER ercd 正常終了（E\_OK）またはエラーコード

【エラーコード】

E\_CTX コンテキストエラー

・非タスクコンテキストからの呼出し〔s〕【NGKI2264】

・CPUロック状態からの呼出し〔s〕【NGKI2265】

E\_ID 不正ID番号

・mpfidが有効範囲外〔s〕【NGKI2266】

E\_RSATR 予約属性

・対象固定長メモリプールが属する保護ドメインの囲みの中

に記述されていない〔S〕【NGKI2267】

・対象固定長メモリプールが属するクラスの囲みの中に記述

されていない〔SM〕【NGKI2268】

E\_NOEXS オブジェクト未登録

・対象固定長メモリプールが未登録【NGKI2269】

E\_OACV オブジェクトアクセス違反

・対象固定長メモリプールに対する管理操作が許可されてい

ない〔s〕【NGKI2270】

E\_MACV メモリアクセス違反

・p\_acvctが指すメモリ領域への読出しアクセスが許可されて

いない〔s〕【NGKI2271】

E\_OBJ オブジェクト状態エラー

・対象固定長メモリプールは静的APIで生成された〔s〕【NGKI2272】

・対象固定長メモリプールに対してアクセス許可ベクタが設

定済み〔S〕【NGKI2273】

【機能】

mpfidで指定した固定長メモリプール（対象固定長メモリプール）のアクセス許

可ベクタ（4つのアクセス許可パターンの組）を、各パラメータで指定した値に

設定する【NGKI2274】．対象固定長メモリプールの固定長メモリプール領域が

コンフィギュレータまたはカーネルにより確保されたものである場合には、固

定長メモリプール領域のアクセス許可ベクタも、各パラメータで指定した値に

設定する【NGKI2275】．

静的APIにおいては、mpfidはオブジェクト識別名、acptn1～acptn4は整数定数

式パラメータである【NGKI2276】．

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、SAC MPFのみをサポートする【HRPS0139】．ただし、動的生

11651 成機能拡張パッケージでは、`sac_mpf`もサポートする【HRPS0200】。

11652 -----

11653 `del_mpf`      固定長メモリプールの削除 [TD]      【NGKI2277】

11654

11655      **【C言語API】**

11656          `ER ercd = del_mpf(ID mpfid)`

11657

11658      **【パラメータ】**

11659          ID                  mpfid                  対象固定長メモリプールのID番号

11660

11661      **【リターンパラメータ】**

11662          ER                  ercd                  正常終了 (E\_OK) またはエラーコード

11663

11664      **【エラーコード】**

11665          E\_CTX              コンテキストエラー

11666                  ・非タスクコンテキストからの呼出し【NGKI2278】

11667                  ・CPUロック状態からの呼出し【NGKI2279】

11668          E\_ID              不正ID番号

11669                  ・mpfidが有効範囲外【NGKI2280】

11670          E\_NOEXS          オブジェクト未登録

11671                  ・対象固定長メモリプールが未登録【NGKI2281】

11672          E\_OACV          オブジェクトアクセス違反

11673                  ・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

11674                  ない [P]      【NGKI2282】

11675          E\_OBJ              オブジェクト状態エラー

11676                  ・対象固定長メモリプールは静的APIで生成された【NGKI2283】

11677

11678      **【機能】**

11679

11680      `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。

11681      具体的な振舞いは以下の通り。

11682

11683      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

11684      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

11685      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

11686      【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

11687      からE\_DLTエラーが返る【NGKI2286】。

11688

11689      **【使用上の注意】**

11690

11691      `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

11692      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

11693      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

11694      み禁止時間が長くなるため、注意が必要である。

11695

11696      **【TOPPERS/ASPカーネルにおける規定】**

11697

11698      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

11699      機能拡張パッケージでは、`del_mpf`をサポートする【ASPS0171】。

11700

11701       【TOPPERS/FMPカーネルにおける規定】

11702

11703       FMPカーネルでは、del\_mpfをサポートしない【FMPS0146】。

11704

11705       【TOPPERS/HRP2カーネルにおける規定】

11706

11707       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  
11708       機能拡張パッケージでは、del\_mpfをサポートする【HRPS0201】。

11709

11710       get\_mpf       固定長メモリブロックの獲得 [T]   【NGKI2287】

11711       pget\_mpf      固定長メモリブロックの獲得（ポーリング） [T]   【NGKI2288】

11712       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】

11713

11714       【C言語API】

11715           ER ercd = get\_mpf(ID mpfid, void \*\*p\_blk)

11716           ER ercd = pget\_mpf(ID mpfid, void \*\*p\_blk)

11717           ER ercd = tget\_mpf(ID mpfid, void \*\*p\_blk, TMO tmout)

11718

11719       【パラメータ】

11720           ID           mpfid       対象固定長メモリプールのID番号

11721           void \*\*       p\_blk       獲得した固定長メモリブロックの先頭番地を入  
11722                               れるメモリ領域へのポインタ

11723           TMO           tmout       タイムアウト時間（twai\_mpfの場合）

11724

11725       【リターンパラメータ】

11726           ER           ercd        正常終了（E\_OK）またはエラーコード

11727           void \*       blk         獲得した固定長メモリブロックの先頭番地

11728

11729       【エラーコード】

11730           E\_CTX       コンテキストエラー

11731                    ・非タスクコンテキストからの呼出し【NGKI2290】

11732                    ・CPUロック状態からの呼出し【NGKI2291】

11733                    ・ディスパッチ保留状態からの呼出し（pget\_mpfを除く）

11734                               【NGKI2292】

11735           E\_NOSPT     未サポート機能

11736                    ・制約タスクからの呼出し（pget\_mpfを除く）【NGKI2293】

11737           E\_ID       不正ID番号

11738                    ・mpfidが有効範囲外【NGKI2294】

11739           E\_PAR       パラメータエラー

11740                    ・tmoutが無効（tget\_mpfの場合）【NGKI2295】

11741           E\_NOEXS     オブジェクト未登録

11742                    ・対象固定長メモリプールが未登録 [D]   【NGKI2296】

11743           E\_OACV     オブジェクトアクセス違反

11744                    ・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  
11745                               ない [P]   【NGKI2297】

11746           E\_MACV     メモリアクセス違反

11747                    ・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  
11748                               ない) [P]   【NGKI2298】

11749           E\_TMOUT     ポーリング失敗またはタイムアウト（get\_mpfを除く）【NGKI2299】

11750           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）

11751                               【NGKI2300】  
 11752           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）  
 11753                               【NGKI2301】  
 11754  
 11755       【機能】  
 11756  
 11757       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  
 11758       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  
 11759       な振舞いは以下の通り。  
 11760  
 11761       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  
 11762       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  
 11763       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】。  
 11764  
 11765       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  
 11766       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】。  
 11767       -----  
 11768       rel\_mpf       固定長メモリブロックの返却 [T]   【NGKI2304】  
 11769  
 11770       【C言語API】  
 11771           ER ercd = rel\_mpf(ID mpfid, void \*blk)  
 11772  
 11773       【パラメータ】  
 11774           ID           mpfid           対象固定長メモリプールのID番号  
 11775           void \*       blk            返却する固定長メモリブロックの先頭番地  
 11776  
 11777       【リターンパラメータ】  
 11778           ER           ercd           正常終了（E\_OK）またはエラーコード  
 11779  
 11780       【エラーコード】  
 11781           E\_CTX       コンテキストエラー  
 11782                       ・非タスクコンテキストからの呼出し【NGKI2305】  
 11783                       ・CPUロック状態からの呼出し【NGKI2306】  
 11784           E\_ID       不正ID番号  
 11785                       ・mpfidが有効範囲外【NGKI2307】  
 11786           E\_PAR       パラメータエラー  
 11787                       ・条件については機能の項を参照  
 11788           E\_NOEXS     オブジェクト未登録  
 11789                       ・対象固定長メモリプールが未登録 [D]   【NGKI2308】  
 11790           E\_OACV     オブジェクトアクセス違反  
 11791                       ・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  
 11792                       ない [P]   【NGKI2309】  
 11793  
 11794       【機能】  
 11795  
 11796       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  
 11797       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。  
 11798  
 11799       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  
 11800       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

11801 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  
 11802 ルからE\_OKが返る【NGKI2311】.

11803

11804 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  
 11805 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】.

11806

11807 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  
 11808 でない場合には、E\_PARエラーとなる【NGKI2313】.

11809 -----

11810 ini\_mpf      固定長メモリプールの再初期化 [T]    【NGKI2314】

11811

11812 【C言語API】

11813     ER ercd = ini\_mpf(ID mpfid)

11814

11815 【パラメータ】

11816     ID            mpfid            対象固定長メモリプールのID番号

11817

11818 【リターンパラメータ】

11819     ER            ercd            正常終了 (E\_OK) またはエラーコード

11820

11821 【エラーコード】

11822     E\_CTX        コンテキストエラー

11823                    ・非タスクコンテキストからの呼出し【NGKI2315】

11824                    ・CPUロック状態からの呼出し【NGKI2316】

11825     E\_ID        不正ID番号

11826                    ・mpfidが有効範囲外【NGKI2317】

11827     E\_NOEXS     オブジェクト未登録

11828                    ・対象固定長メモリプールが未登録 [D]    【NGKI2318】

11829     E\_OACV     オブジェクトアクセス違反

11830                    ・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  
 11831                    ない [P]    【NGKI2319】

11832

11833 【機能】

11834

11835 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  
 11836 る. 具体的な振舞いは以下の通り.

11837

11838 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  
 11839 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  
 11840 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  
 11841 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  
 11842 る【NGKI2322】.

11843

11844 【使用上の注意】

11845

11846 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  
 11847 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  
 11848 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  
 11849 み禁止時間が長くなるため、注意が必要である.

11850

11851 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  
11852 つのは、アプリケーションの責任である。

11853  
11854 **【 $\mu$  ITRON4.0仕様との関係】**

11855  
11856  $\mu$  ITRON4.0仕様に定義されていないサービスコールである。

11857 -----  
11858 ref\_mpf      固定長メモリプールの状態参照 [T]    **【NGKI2323】**

11859  
11860 **【C言語API】**

11861     ER ercd = ref\_mpf(ID mpfid, T\_RMPF \*pk\_rmpf)

11862  
11863 **【パラメータ】**

11864     ID            mpfid          対象固定長メモリプールのID番号  
11865     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  
11866                                   トへのポインタ

11867  
11868 **【リターンパラメータ】**

11869     ER            ercd          正常終了 (E\_OK) またはエラーコード

11870  
11871     \* 固定長メモリプールの現在状態 (パケットの内容)

11872     ID            wtskid       固定長メモリプールの待ち行列の先頭のタスク  
11873                                   のID番号  
11874     uint\_t       fblkcnt      固定長メモリプール領域の空きメモリ領域に割  
11875                                   り付けることができる固定長メモリブロックの  
11876                                   数

11877  
11878 **【エラーコード】**

11879     E\_CTX       コンテキストエラー  
11880                    ・ 非タスクコンテキストからの呼出し **【NGKI2324】**  
11881                    ・ CPUロック状態からの呼出し **【NGKI2325】**  
11882     E\_ID        不正ID番号  
11883                    ・ mpfidが有効範囲外 **【NGKI2326】**  
11884     E\_NOEXS    オブジェクト未登録  
11885                    ・ 対象固定長メモリプールが未登録 [D]    **【NGKI2327】**  
11886     E\_OACV     オブジェクトアクセス違反  
11887                    ・ 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  
11888                    ない [P]    **【NGKI2328】**  
11889     E\_MACV     メモリアクセス違反  
11890                    ・ 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  
11891                    いない) [P]    **【NGKI2329】**

11892  
11893 **【機能】**

11894  
11895 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  
11896 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  
11897 **【NGKI2330】**。

11898  
11899 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  
11900 TSK\_NONE (=0) が返る **【NGKI2331】**。

## 【使用上の注意】

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

## 4.6 時間管理機能

### 4.6.1 システム時刻管理

システム時刻は、カーネルによって管理され、タイムアウト処理、タスクの遅延、周期ハンドラの起動、アラームハンドラの起動に使用される時刻を管理するカーネルオブジェクトである【NGKI3603】。システム時刻は、符号無しの整数型であるSYSTIM型で表され、単位はミリ秒である【NGKI2332】。

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

マルチプロセッサ対応でないカーネルと、マルチプロセッサ対応カーネルでグローバルタイマ方式を用いている場合には、システム時刻は、システムに1つのみ存在する【NGKI2337】。マルチプロセッサ対応カーネルでローカルタイマ方式を用いている場合には、システム時刻は、プロセッサ毎に存在する【NGKI2338】。ローカルタイマ方式とグローバルタイマ方式については、「2.3.4 マルチプロセッサ対応」の節を参照すること。

マルチプロセッサ対応カーネルでローカルタイマ方式を用いている場合には、タイムアウト処理とタスクの遅延処理には、待ち解除されるタスクが割り付けられているプロセッサのシステム時刻が用いられる【NGKI2339】。また、周期ハンドラとアラームハンドラの起動には、それが割り付けられているプロセッサのシステム時刻が用いられる【NGKI2340】。これらの処理単位がマイグレーションする場合には、用いられるシステム時刻も変更される【NGKI2341】。この場合にも、イベントの処理が行われるのは、基準時刻から相対時間によって指定した以上の時間が経過した後となるという規則は維持される【NGKI2342】。

1回のタイムティックの発生により、複数のイベントの処理を行うべき状況になった場合、それらの処理の間の処理順序は規定されない【NGKI2343】。

性能評価用システム時刻は、性能評価に使用することを目的とした、システム時刻よりも精度の高い時刻である。性能評価用システム時刻は、符号無しの整数型であるSYSUTM型で表され、単位はマイクロ秒である【NGKI2344】。ただし、実際の精度はターゲットに依存する【NGKI2345】。

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

11951  
 11952 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。  
 11953  
 11954       TIC\_NUME      タイムティックの周期（単位はミリ秒）の分子      【NGKI2347】  
 11955       TIC\_DEN0      タイムティックの周期（単位はミリ秒）の分母  
 11956  
 11957       TOPPERS\_SUPPORT\_GET\_UTM      get\_utmがサポートされている【NGKI2348】  
 11958  
 11959       【TOPPERS/SSPカーネルにおける規定】  
 11960  
 11961 SSPカーネルでは、時間管理機能をサポートしない【SSPS0129】。  
 11962  
 11963       【使用上の注意】  
 11964  
 11965       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。  
 11966  
 11967       【 $\mu$  ITRON4.0仕様との関係】  
 11968  
 11969       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。  
 11970  
 11971       【 $\mu$  ITRON4.0/PX仕様との関係】  
 11972  
 11973       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。  
 11974  
 11975       -----  
 11976  
 11977       get\_tim      システム時刻の参照 [T]      【NGKI2349】  
 11978  
 11979       【C言語API】  
 11980       ER ercd = get\_tim(SYSTIM \*p\_systim)  
 11981  
 11982       【パラメータ】  
 11983       SYSTIM \*      p\_systim      システム時刻を入れるメモリ領域へのポインタ  
 11984  
 11985       【リターンパラメータ】  
 11986       ER            ercd            正常終了（E\_OK）またはエラーコード  
 11987       SYSTIM       systim          システム時刻の現在値  
 11988  
 11989       【エラーコード】  
 11990       E\_CTX        コンテキストエラー  
 11991                    ・非タスクコンテキストからの呼出し【NGKI2350】  
 11992                    ・CPUロック状態からの呼出し【NGKI2351】  
 11993       E\_OACV       オブジェクトアクセス違反  
 11994                    ・システム状態に対する参照操作が許可されていない [P]  
 12000



12001                                   【NGKI2352】

12002           E\_MACV           メモリアクセス違反

12003                           ・ p\_systimが指すメモリ領域への書込みアクセスが許可され

12004                           ていない) [P] 【NGKI2353】

12005

12006           【機能】

12007

12008           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

12009           メモリ領域に返される 【NGKI2354】 .

12010

12011           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,

12012           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

12013           【NGKI2355】 .

12014

12015           【補足説明】

12016

12017           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

12018           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.

12019           -----

12020   get\_utm           性能評価用システム時刻の参照 [TI] 【NGKI2356】

12021

12022           【C言語API】

12023                   ER ercd = get\_utm(SYSUTM \*p\_sysutm)

12024

12025           【パラメータ】

12026                   SYSUTM \*   p\_sysutm   性能評価用システム時刻を入れるメモリ領域へ

12027                                           のポインタ

12028

12029           【リターンパラメータ】

12030                   ER           ercd           正常終了 (E\_OK) またはエラーコード

12031                   SYSUTM       sysutm       性能評価用システム時刻の現在値

12032

12033           【エラーコード】

12034                   E\_NOSPT       未サポート機能

12035                                   ・ 条件については機能の項を参照

12036                   E\_MACV       メモリアクセス違反

12037                                   ・ p\_sysutmが指すメモリ領域へ書込みアクセスが許可されて

12038                                   いない) [P] 【NGKI2357】

12039

12040           【機能】

12041

12042           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

12043           刻は, p\_sysutmが指すメモリ領域に返される 【NGKI2358】 .

12044

12045           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

12046           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

12047           状態であっても呼び出すことができる.

12048

12049           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】 .

12050           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

12051 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、  
12052 E\_NOSPTエラーが返るか、リンク時にエラーとなる【NGKI2362】。

12053

12054 【使用方法】

12055

12056 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。  
12057 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  
12058 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  
12059 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。

12060

12061 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  
12062 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  
12063 限らない。

12064

12065 【使用上の注意】

12066

12067 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  
12068 ない。

12069

12070 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  
12071 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  
12072 込みの応答性が低下する。

12073

12074 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  
12075 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  
12076 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  
12077 考慮に入れて使用しなければならない。

12078

12079 【μ ITRON4.0仕様との関係】

12080

12081 μ ITRON4.0仕様に定義されていないサービスコールである。

12082 -----

12083

#### 12084 4.6.2 周期ハンドラ

12085

12086 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。  
12087 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】。

12088

12089 各周期ハンドラが持つ情報は次の通り【NGKI2364】。

12090

- 12091 ・ 周期ハンドラ属性
- 12092 ・ 周期ハンドラの動作状態
- 12093 ・ 次に周期ハンドラを起動する時刻
- 12094 ・ 拡張情報
- 12095 ・ 周期ハンドラ先の頭番地
- 12096 ・ 起動周期
- 12097 ・ 起動位相
- 12098 ・ アクセス許可ベクタ（保護機能対応カーネルの場合）
- 12099 ・ 属する保護ドメイン（保護機能対応カーネルの場合）
- 12100 ・ 属するクラス（マルチプロセッサ対応カーネルの場合）

12101  
 12102 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  
 12103 対時間後である【NGKI2365】。

$$12104 \quad \text{起動位相} + \text{起動周期} \times (n-1) \quad n=1, 2, \dots$$

12106  
 12107 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  
 12108 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  
 12109 始、動作していない状態にすることを動作停止という。

12110  
 12111 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  
 12112 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  
 12113 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】。

12114  
 12115 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  
 12116 インは、カーネルドメインに限られる【NGKI2369】。

12117  
 12118 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】。

12119  
 12120 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  
 12121 TA\_PHS 0x04U 周期ハンドラを生成した時刻を基準時刻とする

12122  
 12123 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  
 12124 していない状態となる【NGKI2371】。

12125  
 12126 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  
 12127 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  
 12128 は、周期ハンドラを生成した時刻（静的APIで生成した場合にはカーネルの起動  
 12129 時刻）が、基準時刻となる【NGKI2373】。

12130  
 12131 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  
 12132 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  
 12133 ハンドラの起動処理時に設定される【NGKI2374】。

12134  
 12135 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、  
 12136 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  
 12137 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  
 12138 は、システム時刻管理プロセッサによって実行される。

12139  
 12140 C言語による周期ハンドラの記述形式は次の通り【NGKI2376】。

12141  
 12142 

```
void cyclic_handler(intptr_t exinf)
```

  
 12143 

```
{
```

  
 12144 

```
    周期ハンドラ本体
```

  
 12145 

```
}
```

12146  
 12147 exinfには、周期ハンドラの拡張情報が渡される【NGKI2377】。

12148  
 12149 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。

12150

12151 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  
 12152 ネルでは、静的APIによって登録された周期ハンドラの数  
 12153 に一致）【NGKI2378】  
 12154  
 12155 【TOPPERS/ASPカーネルにおける規定】  
 12156  
 12157 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】。  
 12158  
 12159 【TOPPERS/FMPカーネルにおける規定】  
 12160  
 12161 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】。  
 12162  
 12163 【TOPPERS/HRP2カーネルにおける規定】  
 12164  
 12165 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】。  
 12166  
 12167 【 $\mu$  ITRON4.0仕様との関係】  
 12168  
 12169 TNUM\_CYCIDは、 $\mu$  ITRON4.0仕様に規定されていないカーネル構成マクロである。  
 12170 -----  
 12171 CRE\_CYC 周期ハンドラの生成 [S] 【NGKI2379】  
 12172 acre\_cyc 周期ハンドラの生成 [TD] 【NGKI2380】  
 12173  
 12174 【静的API】  
 12175 CRE\_CYC(ID cycid, { ATR cycatr, intptr\_t exinf, CYCHDR cychdr,  
 12176 RELTIM cycetim, RELTIM cycphs })  
 12177  
 12178 【C言語API】  
 12179 ER\_ID cycid = acre\_cyc(const T\_CCYC \*pk\_ccyc)  
 12180  
 12181 【パラメータ】  
 12182 ID cycid 生成する周期ハンドラのID番号（CRE\_CYCの場合）  
 12183 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  
 12184 ポインタ（静的APIを除く）  
 12185  
 12186 \*周期ハンドラの生成情報（パケットの内容）  
 12187 ATR cycatr 周期ハンドラ属性  
 12188 intptr\_t exinf 周期ハンドラの拡張情報  
 12189 CYCHDR cychdr 周期ハンドラの先頭番地  
 12190 RELTIM cycetim 周期ハンドラの起動周期  
 12191 RELTIM cycphs 周期ハンドラの起動位相  
 12192  
 12193 【リターンパラメータ】  
 12194 ER\_ID cycid 生成された周期ハンドラのID番号（正の値）また  
 12195 はエラーコード  
 12196  
 12197 【エラーコード】  
 12198 E\_CTX コンテキストエラー  
 12199 ・非タスクコンテキストからの呼出し [s] 【NGKI2381】  
 12200 ・CPUロック状態からの呼出し [s] 【NGKI2382】

12201	E_RSATR	予約属性
12202		・ cycatrが無効【NGKI2383】
12203		・ 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】
12204		・ カーネルドメインの囲みの中に記述されていない〔SP〕
12205		【NGKI2385】
12206		・ 属するクラスの指定が有効範囲外〔sM〕【NGKI2386】
12207		・ クラスの囲みの中に記述されていない〔SM〕【NGKI2387】
12208		・ その他の条件については機能の項を参照
12209	E_PAR	パラメータエラー
12210		・ cychdrがプログラムの先頭番地として正しくない【NGKI2388】
12211		・ cyctimが有効範囲（0より大きくTMAX_RELTIM以下）外【NGKI2397】
12212		・ cycphsが有効範囲（0以上TMAX_RELTIM以下）外【NGKI2399】
12213	E_OACV	オブジェクトアクセス違反
12214		・ システム状態に対する管理操作が許可されていない〔sP〕
12215		【NGKI2389】
12216	E_MACV	メモリアクセス違反
12217		・ 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】
12218	E_NOID	ID番号不足
12219		・ 割り付けられる周期ハンドラIDがない〔sD〕【NGKI2391】
12220	E_OBJ	オブジェクト状態エラー
12221		・ cycidで指定した周期ハンドラが登録済み（CRE_CYCの場合）
12222		【NGKI2392】
12223		
12224		
12225		
12226		【機能】
12227		
12228		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
12229		する。具体的な振舞いは以下の通り。
12230		
12231		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
12232		【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
12233		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
12234		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
12235		よい【NGKI2400】。
12236		
12237		cycatrにTA_STAを指定しない場合、対象周期ハンドラは動作していない状態に
12238		初期化される【NGKI2395】。
12239		
12240		静的APIにおいては、cycidはオブジェクト識別名、cycatr、cyctim、cycphsは
12241		整数定数式パラメータ、exinfとcychdrは一般定数式パラメータである
12242		【NGKI2396】。
12243		
12244		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、
12245		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
12246		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】。
12247		
12248		【補足説明】
12249		
12250		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

12251 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。  
 12252 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  
 12253 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  
 12254 ギュレータが警告メッセージを出力する。  
 12255  
 12256 【TOPPERS/ASPカーネルにおける規定】  
 12257  
 12258 ASPカーネルでは、CRE\_CYCのみをサポートする【ASPS0173】。ただし、TA\_PHS  
 12259 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  
 12260 ジでは、acre\_cycもサポートする【ASPS0175】。  
 12261  
 12262 【TOPPERS/FMPカーネルにおける規定】  
 12263  
 12264 FMPカーネルでは、CRE\_CYCのみをサポートする【FMPS0148】。ただし、TA\_PHS  
 12265 属性の周期ハンドラはサポートしない【FMPS0149】。  
 12266  
 12267 【TOPPERS/HRP2カーネルにおける規定】  
 12268  
 12269 HRP2カーネルでは、CRE\_CYCのみをサポートする【HRPS0142】。ただし、  
 12270 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  
 12271 パッケージでは、acre\_cycもサポートする【HRPS0202】。  
 12272  
 12273 【μITRON4.0仕様との関係】  
 12274  
 12275 cychdrのデータ型をCYCHDRに変更した。また、cycphsにcycctimより大きい値を  
 12276 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  
 12277 した。  
 12278 -----  
 12279 AID\_CYC 割付け可能な周期ハンドラIDの数の指定〔SD〕【NGKI2402】  
 12280  
 12281 【静的API】  
 12282 AID\_CYC(uint\_t nocyc)  
 12283  
 12284 【パラメータ】  
 12285 uint\_t nocyc 割付け可能な周期ハンドラIDの数  
 12286  
 12287 【エラーコード】  
 12288 E\_RSATR 予約属性  
 12289 ・保護ドメインの囲みの中に記述されている〔P〕【NGKI3437】  
 12290 ・クラスの囲みの中に記述されていない〔M〕【NGKI2404】  
 12291 ・その他の条件については機能の項を参照  
 12292 E\_PAR パラメータエラー  
 12293 ・nocycが負の値【NGKI3285】  
 12294  
 12295 【機能】  
 12296  
 12297 nocycで指定した数の周期ハンドラIDを、周期ハンドラを生成するサービスコー  
 12298 ルによって割付け可能な周期ハンドラIDとして確保する【NGKI2405】。  
 12299  
 12300 nocycは整数定数式パラメータである【NGKI2406】。

12301  
 12302 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、  
 12303 AID\_CYCが属するクラスの割付け可能プロセッサが、システム時刻管理プロセッ  
 12304 サのみでない場合には、E\_RSATRエラーとなる【NGKI2407】。  
 12305  
 12306 【TOPPERS/ASPカーネルにおける規定】  
 12307  
 12308 ASPカーネルの動的生成機能拡張パッケージでは、AID\_CYCをサポートする  
 12309 【ASPS0217】。  
 12310  
 12311 【TOPPERS/HRP2カーネルにおける規定】  
 12312  
 12313 HRP2カーネルの動的生成機能拡張パッケージでは、AID\_CYCをサポートする  
 12314 【HRPS0218】。  
 12315 -----  
 12316 SAC\_CYC 周期ハンドラのアクセス許可ベクタの設定 [SP] 【NGKI2408】  
 12317 sac\_cyc 周期ハンドラのアクセス許可ベクタの設定 [TPD] 【NGKI2409】  
 12318  
 12319 【静的API】  
 12320 SAC\_CYC(ID cycid, { ACPTN acptn1, ACPTN acptn2,  
 12321 ACPTN acptn3, ACPTN acptn4 })  
 12322  
 12323 【C言語API】  
 12324 ER ercd = sac\_cyc(ID cycid, const ACVCT \*p\_acvct)  
 12325  
 12326 【パラメータ】  
 12327 ID cycid 対象周期ハンドラのID番号  
 12328 ACVCT \* p\_acvct アクセス許可ベクタを入れたパケットへのポ  
 12329 インタ（静的APIを除く）  
 12330  
 12331 \*アクセス許可ベクタ（パケットの内容）  
 12332 ACPTN acptn1 通常操作1のアクセス許可パターン  
 12333 ACPTN acptn2 通常操作2のアクセス許可パターン  
 12334 ACPTN acptn3 管理操作のアクセス許可パターン  
 12335 ACPTN acptn4 参照操作のアクセス許可パターン  
 12336  
 12337 【リターンパラメータ】  
 12338 ER ercd 正常終了（E\_OK）またはエラーコード  
 12339  
 12340 【エラーコード】  
 12341 E\_CTX コンテキストエラー  
 12342 ・非タスクコンテキストからの呼出し [s] 【NGKI2410】  
 12343 ・CPUロック状態からの呼出し [s] 【NGKI2411】  
 12344 E\_ID 不正ID番号  
 12345 ・cycidが有効範囲外 [s] 【NGKI2412】  
 12346 E\_RSATR 予約属性  
 12347 ・対象周期ハンドラが属する保護ドメインの囲みの中に記述  
 12348 されていない [S] 【NGKI2413】  
 12349 ・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  
 12350 いない [SM] 【NGKI2414】

12351	E_NOEXS	オブジェクト未登録
12352		・対象周期ハンドラが未登録【NGKI2415】
12353	E_OACV	オブジェクトアクセス違反
12354		・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]
12355		【NGKI2416】
12356	E_MACV	メモリアクセス違反
12357		・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
12358		いない [s] 【NGKI2417】
12359	E_OBJ	オブジェクト状態エラー
12360		・対象周期ハンドラは静的APIで生成された [s] 【NGKI2418】
12361		・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
12362		[S] 【NGKI2419】
12363		
12364	【機能】	
12365		
12366	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	
12367	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	
12368	【NGKI2420】。	
12369		
12370	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	
12371	式パラメータである【NGKI2421】。	
12372		
12373	【TOPPERS/HRP2カーネルにおける規定】	
12374		
12375	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	
12376	成機能拡張パッケージでは、sac_cycもサポートする【HRPS0203】。	
12377	-----	
12378	del_cyc	周期ハンドラの削除 [TD] 【NGKI2422】
12379		
12380	【C言語API】	
12381	ER ercd = del_cyc(ID cycid)	
12382		
12383	【パラメータ】	
12384	ID	cycid 対象周期ハンドラのID番号
12385		
12386	【リターンパラメータ】	
12387	ER	ercd 正常終了 (E_OK) またはエラーコード
12388		
12389	【エラーコード】	
12390	E_CTX	コンテキストエラー
12391		・非タスクコンテキストからの呼出し【NGKI2423】
12392		・CPUロック状態からの呼出し【NGKI2424】
12393	E_ID	不正ID番号
12394		・cycidが有効範囲外【NGKI2425】
12395	E_NOEXS	オブジェクト未登録
12396		・対象周期ハンドラが未登録【NGKI2426】
12397	E_OACV	オブジェクトアクセス違反
12398		・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]
12399		【NGKI2427】
12400	E_OBJ	オブジェクト状態エラー



12401                                   ・対象周期ハンドラは静的APIで生成された【NGKI2428】  
 12402  
 12403       **【機能】**  
 12404  
 12405       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  
 12406       いは以下の通り．  
 12407  
 12408       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  
 12409       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，  
 12410       動作していない状態にされた後に，登録が解除される【NGKI2430】．  
 12411  
 12412       **【TOPPERS/ASPカーネルにおける規定】**  
 12413  
 12414       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  
 12415       機能拡張パッケージでは，del\_cycをサポートする【ASPS0178】．  
 12416  
 12417       **【TOPPERS/FMPカーネルにおける規定】**  
 12418  
 12419       FMPカーネルでは，del\_cycをサポートしない【FMPS0151】．  
 12420  
 12421       **【TOPPERS/HRP2カーネルにおける規定】**  
 12422  
 12423       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  
 12424       機能拡張パッケージでは，del\_cycをサポートする【HRPS0204】．  
 12425  
 12426       -----  
 12427       sta\_cyc       周期ハンドラの動作開始 [T]   【NGKI2431】  
 12428  
 12429       **【C言語API】**  
 12430       ER ercd = sta\_cyc(ID cycid)  
 12431  
 12432       **【パラメータ】**  
 12433       ID           cycid           対象周期ハンドラのID番号  
 12434  
 12435       **【リターンパラメータ】**  
 12436       ER           ercd           正常終了 (E\_OK) またはエラーコード  
 12437  
 12438       **【エラーコード】**  
 12439       E\_CTX       コンテキストエラー  
 12440                   ・非タスクコンテキストからの呼出し【NGKI2432】  
 12441                   ・CPUロック状態からの呼出し【NGKI2433】  
 12442       E\_ID       不正ID番号  
 12443                   ・cycidが有効範囲外【NGKI2434】  
 12444       E\_NOEXS   オブジェクト未登録  
 12445                   ・対象周期ハンドラが未登録 [D]   【NGKI2435】  
 12446       E\_OACV   オブジェクトアクセス違反  
 12447                   ・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]  
 12448                   【NGKI2436】  
 12449       **【機能】**  
 12450

12451      `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  
 12452      振舞いは以下の通り．  
 12453  
 12454      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  
 12455      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，  
 12456      `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】．  
 12457  
 12458      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  
 12459      時刻の再設定のみが行われる【NGKI2439】．  
 12460  
 12461      **【補足説明】**  
 12462  
 12463      `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，  
 12464      `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  
 12465      に設定される．  
 12466  
 12467      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  
 12468      ドラを起動する時刻は変化しない．  
 12469  
 12470      **【 $\mu$  ITRON4.0仕様との関係】**  
 12471  
 12472      `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  
 12473      ハンドラが起動される時刻を変更した． $\mu$  ITRON4.0仕様では，`sta_cyc`を呼び出  
 12474      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  
 12475      仕様では，起動位相で指定した相対時間後とした．  
 12476      -----  
 12477      `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】  
 12478  
 12479      **【C言語API】**  
 12480           `ER ercd = msta_cyc(ID cycid, ID prcid)`  
 12481  
 12482      **【パラメータ】**  
 12483           `ID`              `cycid`              対象周期ハンドラのID番号  
 12484           `ID`              `prcid`              周期ハンドラの割付け対象のプロセッサのID番号  
 12485  
 12486      **【リターンパラメータ】**  
 12487           `ER`              `ercd`              正常終了（E\_OK）またはエラーコード  
 12488  
 12489      **【エラーコード】**  
 12490           `E_CTX`              コンテキストエラー  
 12491                           ・非タスクコンテキストからの呼出し【NGKI2441】  
 12492                           ・CPUロック状態からの呼出し【NGKI2442】  
 12493           `E_NOSPT`          未サポート機能  
 12494                           ・条件については機能の項を参照  
 12495           `E_ID`              不正ID番号  
 12496                           ・`cycid`が有効範囲外【NGKI2443】  
 12497                           ・`prcid`が有効範囲外【NGKI2444】  
 12498           `E_PAR`              パラメータエラー  
 12499                           ・条件については機能の項を参照  
 12500           `E_NOEXS`          オブジェクト未登録



```

12551
12552   【C言語API】
12553       ER ercd = stp_cyc(ID cycid)
12554
12555   【パラメータ】
12556       ID          cycid          対象周期ハンドラのID番号
12557
12558   【リターンパラメータ】
12559       ER          ercd          正常終了 (E_OK) またはエラーコード
12560
12561   【エラーコード】
12562       E_CTX       コンテキストエラー
12563                   ・非タスクコンテキストからの呼出し 【NGKI2456】
12564                   ・CPUロック状態からの呼出し 【NGKI2457】
12565       E_ID        不正ID番号
12566                   ・cycidが有効範囲外 【NGKI2458】
12567       E_NOEXS     オブジェクト未登録
12568                   ・対象周期ハンドラが未登録 [D] 【NGKI2459】
12569       E_OACV      オブジェクトアクセス違反
12570                   ・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]
12571                   【NGKI2460】
12572
12573   【機能】
12574
12575   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
12576   振舞いは以下の通り．
12577
12578   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
12579   【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
12580   ずに正常終了する 【NGKI2462】．
12581   -----
12582   ref_cyc        周期ハンドラの状態参照 [T] 【NGKI2463】
12583
12584   【C言語API】
12585       ER ercd = ref_cyc(ID cycid, T_RCYC *pk_rcyc)
12586
12587   【パラメータ】
12588       ID          cycid          対象周期ハンドラのID番号
12589       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
12590                                   ポインタ
12591
12592   【リターンパラメータ】
12593       ER          ercd          正常終了 (E_OK) またはエラーコード
12594
12595   *周期ハンドラの現在状態（パケットの内容）
12596       STAT        cycstat        周期ハンドラの動作状態
12597       RELTIM      lefttim        次に周期ハンドラを起動する時刻までの相対時間
12598       ID          prcid          周期ハンドラの割付けプロセッサのID（マルチプ
12599                                   ロセッサ対応カーネルの場合）
12600

```

12601      **【エラーコード】**

12602          E\_CTX      コンテキストエラー

12603                  ・非タスクコンテキストからの呼出し【NGKI2464】

12604                  ・CPUロック状態からの呼出し【NGKI2465】

12605          E\_ID      不正ID番号

12606                  ・cycidが有効範囲外【NGKI2466】

12607          E\_NOEXS      オブジェクト未登録

12608                  ・対象周期ハンドラが未登録 [D] 【NGKI2467】

12609          E\_OACV      オブジェクトアクセス違反

12610                  ・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]

12611                  【NGKI2468】

12612          E\_MACV      メモリアクセス違反

12613                  ・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

12614                  いない [P] 【NGKI2469】

12615

12616      **【機能】**

12617

12618      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

12619      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】．

12620

12621      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

12622      される【NGKI2471】．

12623

12624          TCYC\_STP      0x01U      周期ハンドラが動作していない状態

12625          TCYC\_STA      0x02U      周期ハンドラが動作している状態

12626

12627      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

12628      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

12629      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

12630      【NGKI2473】．

12631

12632      マルチプロセッサ対応カーネルでは、preidに、対象周期ハンドラの割付けプロ

12633      セッサのID番号が返される【NGKI2474】．

12634

12635      **【使用上の注意】**

12636

12637      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

12638      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

12639      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

12640      状態が変化している可能性があるためである．

12641

12642      **【μITRON4.0仕様との関係】**

12643

12644      TCYC\_STPとTCYC\_STAを値を変更した．

12645

12646

## 12647      4.6.3 アラームハンドラ

12648

12649      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

12650      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

12651 別する【NGKI2475】。

12652

12653 各アラームハンドラが持つ情報は次の通り【NGKI2476】。

12654

12655 ・アラームハンドラ属性

12656 ・アラームハンドラの動作状態

12657 ・アラームハンドラを起動する時刻

12658 ・拡張情報

12659 ・アラームハンドラの手頭番地

12660 ・アクセス許可ベクタ（保護機能対応カーネルの場合）

12661 ・属する保護ドメイン（保護機能対応カーネルの場合）

12662 ・属するクラス（マルチプロセッサ対応カーネルの場合）

12663

12664 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  
12665 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  
12666 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。

12667

12668 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  
12669 定される【NGKI2478】。

12670

12671 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  
12672 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  
12673 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】。  
12674 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  
12675 【NGKI2481】。

12676

12677 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  
12678 ドメインは、カーネルドメインに限られる【NGKI2482】。

12679

12680 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、  
12681 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  
12682 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  
12683 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。

12684

12685 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  
12686 ハンドラ属性には、TA\_NULLを指定しなければならない【NGKI3424】。

12687

12688 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】。

12689

```
12690     void alarm_handler(intptr_t exinf)  
12691     {  
12692         アラームハンドラ本体  
12693     }
```

12694

12695 exinfには、アラームハンドラの拡張情報が渡される【NGKI2485】。

12696

12697 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。

12698

12699 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  
12700 カーネルでは、静的APIによって登録されたアラームハン

12701 ドラの数に一致) 【NGKI2486】

12702

12703 【 $\mu$  ITRON4.0仕様との関係】

12704

12705 TNUM\_ALMIDは、 $\mu$  ITRON4.0仕様に規定されていないカーネル構成マクロである。

12706 -----

12707 CRE\_ALM アラームハンドラの生成 [S] 【NGKI2487】

12708 acre\_alm アラームハンドラの生成 [TD] 【NGKI2488】

12709

12710 【静的API】

12711 CRE\_ALM(ID almid, { ATR almatr, intptr\_t exinf, ALMHDR almhdr })

12712

12713 【C言語API】

12714 ER\_ID almid = acre\_alm(const T\_CALM \*pk\_calm)

12715

12716 【パラメータ】

12717 ID almid 生成するアラームハンドラのID番号 (CRE\_ALM

12718 の場合)

12719 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

12720 へのポインタ (静的APIを除く)

12721

12722 \* アラームハンドラの生成情報 (パケットの内容)

12723 ATR almatr アラームハンドラ属性

12724 intptr\_t exinf アラームハンドラの拡張情報

12725 ALMHDR almhdr アラームハンドラの先頭番地

12726

12727 【リターンパラメータ】

12728 ER\_ID almid 生成されたアラームハンドラのID番号 (正の値)

12729 またはエラーコード

12730

12731 【エラーコード】

12732 E\_CTX コンテキストエラー

12733 ・非タスクコンテキストからの呼出し [s] 【NGKI2489】

12734 ・CPUロック状態からの呼出し [s] 【NGKI2490】

12735 E\_RSATR 予約属性

12736 ・almatrが無効 【NGKI2491】

12737 ・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

12738 メイン以外 [sP] 【NGKI2492】

12739 ・カーネルドメインの囲みの中に記述されていない [SP]

12740 【NGKI2493】

12741 ・属するクラスの指定が有効範囲外 [sM] 【NGKI2494】

12742 ・クラスの囲みの中に記述されていない [SM] 【NGKI2495】

12743 ・その他の条件については機能の項を参照

12744 E\_PAR パラメータエラー

12745 ・almhdrがプログラムの先頭番地として正しくない 【NGKI2496】

12746 E\_OACV オブジェクトアクセス違反

12747 ・システム状態に対する管理操作が許可されていない [sP]

12748 【NGKI2497】

12749 E\_MACV メモリアクセス違反

12750 ・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

12751                   いない [sP] 【NGKI2498】

12752           E\_NOID       ID番号不足

12753                   ・割り付けられるアラームハンドラIDがない [sD] 【NGKI2499】

12754           E\_OBJ       オブジェクト状態エラー

12755                   ・almidで指定したアラームハンドラが登録済み (CRE\_ALMの

12756                   場合) 【NGKI2500】

12757

12758       【機能】

12759

12760       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

12761       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

12762       【NGKI2501】。

12763

12764       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

12765       タ、exinfとalmhdrは一般定数式パラメータである 【NGKI2502】。

12766

12767       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、

12768       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

12769       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】。

12770

12771       【TOPPERS/ASPカーネルにおける規定】

12772

12773       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

12774       成機能拡張パッケージでは、acre\_almもサポートする 【ASPS0180】。

12775

12776       【TOPPERS/FMPカーネルにおける規定】

12777

12778       FMPカーネルでは、CRE\_ALMのみをサポートする 【FMPS0152】。

12779

12780       【TOPPERS/HRP2カーネルにおける規定】

12781

12782       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

12783       成機能拡張パッケージでは、acre\_almもサポートする 【HRPS0205】。

12784

12785       【μITRON4.0仕様との関係】

12786

12787       almhdrのデータ型をALMHDRに変更した。

12788       -----

12789       AID\_ALM       割付け可能なアラームハンドラIDの数の指定 [SD] 【NGKI2504】

12790

12791       【静的API】

12792           AID\_ALM(uint\_t noalm)

12793

12794       【パラメータ】

12795           uint\_t       noalm       割付け可能なアラームハンドラIDの数

12796

12797       【エラーコード】

12798           E\_RSATR       予約属性

12799                   ・保護ドメインの囲みの中に記述されている [P] 【NGKI3438】

12800                   ・クラスの囲みの中に記述されていない [M] 【NGKI2506】



12801                   ・その他の条件については機能の項を参照  
12802                   E\_PAR           パラメータエラー  
12803                   ・noalmが負の値【NGKI3286】  
12804  
12805                   【機能】  
12806  
12807                   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  
12808                   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  
12809                   【NGKI2507】。  
12810  
12811                   noalmは整数定数式パラメータである【NGKI2508】。  
12812  
12813                   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、  
12814                   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  
12815                   サのみでない場合には、E\_RSATRエラーとなる【NGKI2509】。  
12816  
12817                   【TOPPERS/ASPカーネルにおける規定】  
12818  
12819                   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  
12820                   【ASPS0218】。  
12821  
12822                   【TOPPERS/HRP2カーネルにおける規定】  
12823  
12824                   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  
12825                   【HRPS0219】。  
12826  
12827                   -----  
12828                   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】  
12829                   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】  
12830  
12831                   【静的API】  
12832                   SAC\_ALM(ID almid, { ACPTN acptn1, ACPTN acptn2,  
12833                   ACPTN acptn3, ACPTN acptn4 })  
12834  
12835                   【C言語API】  
12836                   ER ercd = sac\_alm(ID almid, const ACVCT \*p\_acvct)  
12837  
12838                   【パラメータ】  
12839                   ID           almid           対象アラームハンドラのID番号  
12840                   ACVCT \*   p\_acvct        アクセス許可ベクタを入れたパケットへのポ  
12841                   インタ（静的APIを除く）  
12842  
12843                   \*アクセス許可ベクタ（パケットの内容）  
12844                   ACPTN       acptn1        通常操作1のアクセス許可パターン  
12845                   ACPTN       acptn2        通常操作2のアクセス許可パターン  
12846                   ACPTN       acptn3        管理操作のアクセス許可パターン  
12847                   ACPTN       acptn4        参照操作のアクセス許可パターン  
12848  
12849                   【リターンパラメータ】  
12850                   ER           ercd           正常終了（E\_OK）またはエラーコード

12851       **【エラーコード】**

12852       E\_CTX       コンテキストエラー

12853               ・非タスクコンテキストからの呼出し [s]   **【NGKI2512】**

12854               ・CPUロック状態からの呼出し [s]   **【NGKI2513】**

12855       E\_ID       不正ID番号

12856               ・almidが有効範囲外 [s]   **【NGKI2514】**

12857       E\_RSATR     予約属性

12858               ・対象アラームハンドラが属する保護ドメインの囲みの中に

12859               記述されていない [S]   **【NGKI2515】**

12860               ・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

12861               れていない [SM]   **【NGKI2516】**

12862       E\_NOEXS     オブジェクト未登録

12863               ・対象アラームハンドラが未登録 **【NGKI2517】**

12864       E\_OACV     オブジェクトアクセス違反

12865               ・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

12866               ない [s]   **【NGKI2518】**

12867       E\_MACV     メモリアクセス違反

12868               ・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

12869               いない [s]   **【NGKI2519】**

12870       E\_OBJ     オブジェクト状態エラー

12871               ・対象アラームハンドラは静的APIで生成された [s]   **【NGKI2520】**

12872               ・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

12873               済み [S]   **【NGKI2521】**

12874

12875       **【機能】**

12876

12877       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

12878       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

12879       する **【NGKI2522】** .

12880

12881       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

12882       式パラメータである **【NGKI2523】** .

12883

12884       **【TOPPERS/HRP2カーネルにおける規定】**

12885

12886       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

12887       成機能拡張パッケージでは、sac\_almもサポートする **【HRPS0206】** .

12888       -----

12889       del\_alm       アラームハンドラの削除 [TD]   **【NGKI2524】**

12890

12891       **【C言語API】**

12892       ER ercd = del\_alm(ID almid)

12893

12894       **【パラメータ】**

12895       ID           almid       対象アラームハンドラのID番号

12896

12897       **【リターンパラメータ】**

12898       ER           ercd       正常終了 (E\_OK) またはエラーコード

12899

12900       **【エラーコード】**

12901 E\_CTX コンテキストエラー  
 12902 ・非タスクコンテキストからの呼出し【NGKI2525】  
 12903 ・CPUロック状態からの呼出し【NGKI2526】  
 12904 E\_ID 不正ID番号  
 12905 ・almidが有効範囲外【NGKI2527】  
 12906 E\_NOEXS オブジェクト未登録  
 12907 ・対象アラームハンドラが未登録【NGKI2528】  
 12908 E\_OACV オブジェクトアクセス違反  
 12909 ・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  
 12910 い [P] 【NGKI2529】  
 12911 E\_OBJ オブジェクト状態エラー  
 12912 ・対象アラームハンドラは静的APIで生成された【NGKI2530】  
 12913  
 12914 **【機能】**  
 12915  
 12916 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  
 12917 的な振舞いは以下の通り．  
 12918  
 12919 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  
 12920 状態に戻される【NGKI2531】．対象アラームハンドラが動作している状態であっ  
 12921 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  
 12922 【NGKI2532】．  
 12923  
 12924 **【TOPPERS/ASPカーネルにおける規定】**  
 12925  
 12926 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  
 12927 機能拡張パッケージでは，del\_almをサポートする【ASPS0183】．  
 12928  
 12929 **【TOPPERS/FMPカーネルにおける規定】**  
 12930  
 12931 FMPカーネルでは，del\_almをサポートしない【FMPS0154】．  
 12932  
 12933 **【TOPPERS/HRP2カーネルにおける規定】**  
 12934  
 12935 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  
 12936 機能拡張パッケージでは，del\_almをサポートする【HRPS0207】．  
 12937 -----  
 12938 sta\_alm アラームハンドラの動作開始 [T] 【NGKI2533】  
 12939 ista\_alm アラームハンドラの動作開始 [I] 【NGKI2534】  
 12940  
 12941 **【C言語API】**  
 12942 ER ercd = sta\_alm(ID almid, RELTIM almtim)  
 12943 ER ercd = ista\_alm(ID almid, RELTIM almtim)  
 12944  
 12945 **【パラメータ】**  
 12946 ID almid 対象アラームハンドラのID番号  
 12947 RELTIM almtim アラームハンドラの起動時刻（相対時間）  
 12948  
 12949 **【リターンパラメータ】**  
 12950 ER ercd 正常終了（E\_OK）またはエラーコード

12951  
 12952 **【エラーコード】**  
 12953     E\_CTX     コンテキストエラー  
 12954                ・非タスクコンテキストからの呼出し (sta\_almの場合) 【NGKI2535】  
 12955                ・タスクコンテキストからの呼出し (ista\_almの場合) 【NGKI2536】  
 12956                ・CPUロック状態からの呼出し  
 12957     E\_ID     不正ID番号  
 12958                ・almidが有効範囲外 【NGKI2537】  
 12959     E\_PAR     パラメータエラー  
 12960                ・almtimがTMAX\_RELTIMより大きい 【NGKI2538】  
 12961     E\_NOEXS   オブジェクト未登録  
 12962                ・対象アラームハンドラが未登録 [D] 【NGKI2539】  
 12963     E\_OACV    オブジェクトアクセス違反  
 12964                ・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  
 12965                ない (sta\_almの場合) [P] 【NGKI2540】  
 12966  
 12967 **【機能】**  
 12968  
 12969 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。  
 12970 具体的な振舞いは以下の通り。  
 12971  
 12972 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  
 12973 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、  
 12974 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  
 12975 【NGKI2542】。  
 12976  
 12977 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  
 12978 する時刻の再設定のみが行われる 【NGKI2543】。  
 12979 -----  
 12980 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】  
 12981 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】  
 12982  
 12983 **【C言語API】**  
 12984     ER ercd = msta\_alm(ID almid, RELTIM almtim, ID prcid)  
 12985     ER ercd = imsta\_alm(ID almid, RELTIM almtim, ID prcid)  
 12986  
 12987 **【パラメータ】**  
 12988     ID        almid        対象アラームハンドラのID番号  
 12989     RELTIM    almtim      アラームハンドラの起動時刻 (相対時間)  
 12990     ID        prcid        アラームハンドラの割付け対象のプロセッサの  
 12991                ID番号  
 12992  
 12993 **【リターンパラメータ】**  
 12994     ER        ercd        正常終了 (E\_OK) またはエラーコード  
 12995  
 12996 **【エラーコード】**  
 12997     E\_CTX     コンテキストエラー  
 12998                ・非タスクコンテキストからの呼出し (msta\_almの場合)  
 12999                【NGKI2546】  
 13000                ・タスクコンテキストからの呼出し (imsta\_almの場合) 【NGKI2547】

13001		・CPUロック状態からの呼出し【NGKI2548】
13002	E_NOSPT	未サポート機能
13003		・条件については機能の項を参照
13004	E_ID	不正ID番号
13005		・almidが有効範囲外【NGKI2549】
13006		・prcidが有効範囲外【NGKI2550】
13007	E_PAR	パラメータエラー
13008		・almtimがTMAX_RELTIMより大きい【NGKI2551】
13009		・その他の条件については機能の項を参照
13010	E_NOEXS	オブジェクト未登録
13011		・対象アラームハンドラが未登録 [D] 【NGKI2552】
13012	E_OACV	オブジェクトアクセス違反
13013		・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】
13014		
13015		
13016		<b>【機能】</b>
13017		
13018		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。
13019		
13020		
13021		
13022		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】。
13023		
13024		
13025		
13026		
13027		
13028		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】。
13029		
13030		
13031		
13032		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】。
13033		
13034		
13035		
13036		
13037		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】。
13038		
13039		
13040		prcidにTPRC_INI (=0) を指定すると、対象アラームハンドラの割付けプロセッサを、それが属するクラスの初期割付けプロセッサとする【NGKI2560】。
13041		
13042		
13043		グローバルタイマ方式を用いている場合、msta_alm/imsta_almはE_NOSPTを返す【NGKI2561】。
13044		
13045		
13046		<b>【使用上の注意】</b>
13047		
13048		msta_alm/imsta_almで実行中のアラームハンドラの割付けプロセッサを変更した場合、同じアラームハンドラが異なるプロセッサで同時に実行される可能性がある。特に、almtimに0を指定する場合に、注意が必要である。
13049		
13050		

13051

13052     【 $\mu$  ITRON4.0仕様との関係】

13053

13054      $\mu$  ITRON4.0仕様に定義されていないサービスコールである。

13055

13056     stp\_alm     アラームハンドラの動作停止 [T]     【NGKI2562】

13057     istp\_alm    アラームハンドラの動作停止 [I]     【NGKI2563】

13058

13059     【C言語API】

13060         ER ercd = stp\_alm(ID almid)

13061         ER ercd = istp\_alm(ID almid)

13062

13063     【パラメータ】

13064         ID           almid           対象アラームハンドラのID番号

13065

13066     【リターンパラメータ】

13067         ER           ercd           正常終了 (E\_OK) またはエラーコード

13068

13069     【エラーコード】

13070         E\_CTX       コンテキストエラー

13071             ・非タスクコンテキストからの呼出し (stp\_almの場合)     【NGKI2564】

13072             ・タスクコンテキストからの呼出し (istp\_almの場合)   【NGKI2565】

13073             ・CPUロック状態からの呼出し     【NGKI2566】

13074         E\_ID       不正ID番号

13075             ・almidが有効範囲外     【NGKI2567】

13076         E\_NOEXS    オブジェクト未登録

13077             ・対象アラームハンドラが未登録 [D]     【NGKI2568】

13078         E\_OACV    オブジェクトアクセス違反

13079             ・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

13080                 ない (stp\_almの場合) [P]     【NGKI2569】

13081

13082     【機能】

13083

13084     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。

13085     具体的な振舞いは以下の通り。

13086

13087     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

13088     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

13089     行われずに正常終了する【NGKI2571】。

13090

13091     ref\_alm     アラームハンドラの状態参照 [T]     【NGKI2572】

13092

13093     【C言語API】

13094         ER ercd = ref\_alm(ID almid, T\_RALM \*pk\_ralm)

13095

13096     【パラメータ】

13097         ID           almid           対象アラームハンドラのID番号

13098         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

13099                 へのポインタ

13100

13101     **【リターンパラメータ】**

13102         ER             ercd             正常終了 (E\_OK) またはエラーコード

13103

13104         **\* アラームハンドラの現在状態 (パケットの内容)**

13105         STAT         almstat         アラームハンドラの動作状態

13106         RELTIM       lefttim         アラームハンドラを起動する時刻までの相対時間

13107         ID           prcid            アラームハンドラの割付けプロセッサのID (マルチ  
13108                                          プロセッサ対応カーネルの場合)

13109

13110         **【エラーコード】**

13111         E\_CTX         コンテキストエラー

13112                         ・非タスクコンテキストからの呼出し【NGKI2573】

13113                         ・CPUロック状態からの呼出し【NGKI2574】

13114         E\_ID         不正ID番号

13115                         ・almidが有効範囲外【NGKI2575】

13116         E\_NOEXS       オブジェクト未登録

13117                         ・対象アラームハンドラが未登録 [D] 【NGKI2576】

13118         E\_OACV       オブジェクトアクセス違反

13119                         ・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  
13120                                          ない [P] 【NGKI2577】

13121         E\_MACV       メモリアクセス違反

13122                         ・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  
13123                                          いない [P] 【NGKI2578】

13124

13125         **【機能】**

13126

13127         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  
13128         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】.

13129

13130         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  
13131         が返される【NGKI2580】.

13132

13133         TALM\_STP       0x01U         アラームハンドラが動作していない状態

13134         TALM\_STA       0x02U         アラームハンドラが動作している状態

13135

13136         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  
13137         ームハンドラ起動する時刻までの相対時間が返される【NGKI2581】. 対象アラ  
13138         ームハンドラが動作していない状態である場合には, lefttimの値は保証されない  
13139         【NGKI2582】.

13140

13141         マルチプロセッサ対応カーネルでは, prcidに, 対象アラームハンドラの割付け  
13142         プロセッサのID番号が返される【NGKI2583】.

13143

13144         **【使用上の注意】**

13145

13146         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  
13147         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  
13148         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  
13149         ンドラの状態が変化している可能性があるためである.

13150

13151 【 $\mu$  ITRON4.0仕様との関係】

13152

13153 TALM\_STPとTALM\_STAを値を変更した。

13154 -----

13155

#### 13156 4.6.4 オーバランハンドラ

13157

13158 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  
13159 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  
13160 は、システムで1つのみ登録することができる【NGKI2584】。

13161

13162 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

13163 【NGKI2585】。

13164

13165 ・オーバランハンドラの動作状態

13166 ・残りプロセッサ時間

13167

13168 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  
13169 いない状態のいずれかをとり【NGKI2586】。残りプロセッサ時間は、オーバ  
13170 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  
13171 時間を表す。

13172

13173 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  
13174 移する時に、動作していない状態に初期化される【NGKI2587】。

13175

13176 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  
13177 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】。  
13178 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  
13179 ドラが起動される【NGKI2589】。

13180

13181 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  
13182 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  
13183 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  
13184 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  
13185 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  
13186 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/  
13187 復帰にかかる時間と、それらの入口処理と出口処理の一部の実行時間は含んで  
13188 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  
13189 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  
13190 【NGKI2592】。

13191

13192 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  
13193 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  
13194 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  
13195 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  
13196 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  
13197 サ時間の計測精度はターゲットに依存する【NGKI2596】。

13198

13199 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  
13200 属する【NGKI2597】。



13201  
13202 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  
13203 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、  
13204 TOPPERS\_SUPPORT\_OVRHDRがマクロ定義される【NGKI2599】。サポートされてい  
13205 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  
13206 ラーが返るか、リンク時にエラーとなる【NGKI2600】。  
13207  
13208 オーバランハンドラ機能に用いるデータ型は次の通り。  
13209  
13210       OVRTIM       プロセッサ時間（符号無し整数，単位はマイクロ秒，ulong\_t  
13211                      に定義）【NGKI2601】  
13212  
13213 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  
13214 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】。  
13215  
13216 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】。  
13217  
13218       void overrun\_handler(ID tskid, intptr\_t exinf)  
13219       {  
13220           オーバランハンドラ本体  
13221       }  
13222  
13223 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  
13224 張情報が，それぞれ渡される【NGKI2605】。  
13225  
13226 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。  
13227  
13228       TMAX\_OVRTIM       プロセッサ時間に指定できる最大値【NGKI2606】  
13229  
13230       TOPPERS\_SUPPORT\_OVRHDR       オーバランハンドラ機能がサポートされて  
13231                                      いる【NGKI2607】  
13232  
13233       【使用上の注意】  
13234  
13235 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  
13236 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。  
13237  
13238       【TOPPERS/ASPカーネルにおける規定】  
13239  
13240 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，  
13241 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  
13242 を追加することができる【ASPS0185】。  
13243  
13244       【TOPPERS/FMPカーネルにおける規定】  
13245  
13246 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】。  
13247  
13248       【TOPPERS/HRP2カーネルにおける規定】  
13249  
13250 HRP2カーネルでは，オーバランハンドラをサポートする【HRPS0149】。

13251  
 13252     **【 $\mu$  ITRON4.0仕様との関係】**  
 13253  
 13254     OVRTIMの時間単位は、 $\mu$  ITRON4.0仕様では実装定義としていたが、この仕様で  
 13255     はマイクロ秒と規定した。  
 13256  
 13257     TMAX\_OVRTIMは、 $\mu$  ITRON4.0仕様に規定されていないカーネル構成マクロである。  
 13258     -----  
 13259     DEF\_OVR     オーバランハンドラの定義 [S]     【NGKI2608】  
 13260     def\_ovr     オーバランハンドラの定義 [TD]     【NGKI2609】  
 13261  
 13262     **【静的API】**  
 13263     DEF\_OVR({ ATR ovratr, OVRHDR ovrhdr })  
 13264  
 13265     **【C言語API】**  
 13266     ER ercd = def\_ovr(const T\_DOVR \*pk\_dovr)  
 13267  
 13268     **【パラメータ】**  
 13269     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  
 13270     トへのポインタ（静的APIを除く）  
 13271  
 13272     \* オーバランハンドラの定義情報（パケットの内容）  
 13273     ATR           ovratr     オーバランハンドラ属性  
 13274     OVRHDR       ovrhdr     オーバランハンドラの先頭番地  
 13275  
 13276     **【リターンパラメータ】**  
 13277     ER           ercd       正常終了 (E\_OK) またはエラーコード  
 13278  
 13279     **【エラーコード】**  
 13280     E\_CTX        コンテキストエラー  
 13281                ・非タスクコンテキストからの呼出し [s]     【NGKI2610】  
 13282                ・CPUロック状態からの呼出し [s]     【NGKI2611】  
 13283     E\_RSATR      予約属性  
 13284                ・ovratrが無効 【NGKI2612】  
 13285                ・その他の条件については機能の項を参照  
 13286     E\_PAR        パラメータエラー  
 13287                ・ovrhdrがプログラムの先頭番地として正しくない 【NGKI2613】  
 13288     E\_OACV       オブジェクトアクセス違反  
 13289                ・システム状態に対する管理操作が許可されていない [sP]  
 13290                【NGKI2614】  
 13291     E\_MACV       メモリアクセス違反  
 13292                ・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  
 13293                いない [sP]     【NGKI2615】  
 13294     E\_OBJ        オブジェクト状態エラー  
 13295                ・条件については機能の項を参照  
 13296  
 13297     **【機能】**  
 13298  
 13299     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  
 13300     ンドラを定義する 【NGKI2616】。ただし、def\_ovrにおいてpk\_dovrをNULLにし

13301 た場合には、オーバランハンドラの定義を解除する【NGKI2617】。  
13302  
13303 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  
13304 メータである【NGKI2618】。  
13305  
13306 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  
13307 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  
13308 る場合には、E\_OBJエラーとなる【NGKI2619】。  
13309  
13310 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  
13311 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  
13312 【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  
13313 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  
13314 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】。  
13315 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  
13316 ラーは検出されない【NGKI2623】。  
13317  
13318 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  
13319 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  
13320 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  
13321 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  
13322 指定した場合にはE\_RSATRエラーとなる【NGKI2626】。ただし、  
13323 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  
13324 出されない【NGKI2627】。  
13325  
13326 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  
13327 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  
13328 となる【NGKI2628】。  
13329  
13330 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、  
13331 すべてのタスクに対して動作していない状態となる【NGKI2629】。  
13332  
13333 【使用上の注意】  
13334  
13335 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  
13336 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  
13337 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  
13338 なるため、注意が必要である。  
13339  
13340 【TOPPERS/ASPカーネルにおける規定】  
13341  
13342 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  
13343 ポートする【ASPS0186】。  
13344  
13345 【TOPPERS/HRP2カーネルにおける規定】  
13346  
13347 HRP2カーネルでは、DEF\_OVRのみをサポートする【HRPS0150】。  
13348  
13349 【μITRON4.0仕様との関係】  
13350

13351      ovrhdrのデータ型をOVRHDRに変更した.  
 13352  
 13353      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,  
 13354      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  
 13355      度定義を解除してから, 再度定義する必要がある.  
 13356      -----  
 13357      sta\_ovr      オーバランハンドラの動作開始 [T]      【NGKI2630】  
 13358      ista\_ovr      オーバランハンドラの動作開始 [I]      【NGKI2631】  
 13359  
 13360      【C言語API】  
 13361          ER ercd = sta\_ovr(ID tskid, OVRTIM ovrtime)  
 13362          ER ercd = ista\_ovr(ID tskid, OVRTIM ovrtime)  
 13363  
 13364      【パラメータ】  
 13365          ID                  tskid                  対象タスクのID番号  
 13366          OVRTIM              ovrtime              対象タスクの残りプロセッサ時間  
 13367  
 13368      【リターンパラメータ】  
 13369          ER                  ercd                  正常終了 (E\_OK) またはエラーコード  
 13370  
 13371      【エラーコード】  
 13372          E\_CTX              コンテキストエラー  
 13373                  ・非タスクコンテキストからの呼出し (sta\_ovrの場合)      【NGKI2632】  
 13374                  ・タスクコンテキストからの呼出し (ista\_ovrの場合)      【NGKI2633】  
 13375                  ・CPUロック状態からの呼出し      【NGKI2634】  
 13376          E\_ID                  不正ID番号  
 13377                  ・tskidが有効範囲外      【NGKI2635】  
 13378          E\_NOEXS              オブジェクト未登録  
 13379                  ・対象タスクが未登録 [D]      【NGKI2636】  
 13380          E\_OACV              オブジェクトアクセス違反  
 13381                  ・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  
 13382                  の場合) [P]      【NGKI2637】  
 13383          E\_PAR                  パラメータエラー  
 13384                  ・ovrtimが0, またはTMAX\_OVRTIMより大きい      【NGKI2643】  
 13385          E\_OBJ                  オブジェクト状態エラー  
 13386                  ・オーバランハンドラが定義されていない      【NGKI2638】  
 13387  
 13388      【機能】  
 13389  
 13390      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  
 13391      開始する. 具体的な振舞いは以下の通り.  
 13392  
 13393      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  
 13394      り, 残りプロセッサ時間は, ovrtimeに指定した時間に設定される 【NGKI2639】.  
 13395      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  
 13396      ロセッサ時間の設定のみが行われる 【NGKI2640】.  
 13397  
 13398      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  
 13399      となる 【NGKI2641】.  
 13400

13401       【 $\mu$  ITRON4.0仕様との関係】

13402

13403       ista\_ovrは、 $\mu$  ITRON4.0仕様に定義されていないサービスコールである。

13404

13405       stp\_ovr       オーバランハンドラの動作停止 [T]   【NGKI2644】

13406       istp\_ovr     オーバランハンドラの動作停止 [I]   【NGKI2645】

13407

## 13408       【C言語API】

13409           ER ercd = stp\_ovr(ID tskid)

13410           ER ercd = istp\_ovr(ID tskid)

13411

## 13412       【パラメータ】

13413           ID           tskid           対象タスクのID番号

13414

## 13415       【リターンパラメータ】

13416           ER           ercd           正常終了 (E\_OK) またはエラーコード

13417

## 13418       【エラーコード】

13419           E\_CTX       コンテキストエラー

13420                   ・非タスクコンテキストからの呼出し (stp\_ovrの場合)   【NGKI2646】

13421                   ・タスクコンテキストからの呼出し (istp\_ovrの場合)   【NGKI2647】

13422                   ・CPUロック状態からの呼出し   【NGKI2648】

13423           E\_ID       不正ID番号

13424                   ・tskidが有効範囲外   【NGKI2649】

13425           E\_NOEXS   オブジェクト未登録

13426                   ・対象タスクが未登録 [D]   【NGKI2650】

13427           E\_OACV     オブジェクトアクセス違反

13428                   ・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  
13429                   の場合) [P]   【NGKI2651】

13430           E\_OBJ       オブジェクト状態エラー

13431                   ・オーバランハンドラが定義されていない   【NGKI2652】

13432

## 13433       【機能】

13434

13435       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  
13436       停止する。具体的な振舞いは以下の通り。

13437

13438       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  
13439       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  
13440       状態であれば、何も行われずに正常終了する【NGKI2654】。

13441

13442       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  
13443       となる【NGKI2655】。

13444

13445       【 $\mu$  ITRON4.0仕様との関係】

13446

13447       istp\_ovrは、 $\mu$  ITRON4.0仕様に定義されていないサービスコールである。

13448

13449       ref\_ovr       オーバランハンドラの状態参照 [T]   【NGKI2656】

13450

13451 **【C言語API】**  
 13452 ER ercd = ref\_ovr(ID tskid, T\_ROVR \*pk\_rovr)  
 13453  
 13454 **【パラメータ】**  
 13455 ID tskid 対象タスクのID番号  
 13456 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  
 13457 トへのポインタ  
 13458  
 13459 **【リターンパラメータ】**  
 13460 ER ercd 正常終了 (E\_OK) またはエラーコード  
 13461  
 13462 \*タスクの現在状態 (パケットの内容)  
 13463 STAT ovrstat オーバランハンドラの動作状態  
 13464 OVRTIM leftotm 残りプロセッサ時間  
 13465  
 13466 **【エラーコード】**  
 13467 E\_CTX コンテキストエラー  
 13468 ・非タスクコンテキストからの呼出し【NGKI2657】  
 13469 ・CPUロック状態からの呼出し【NGKI2658】  
 13470 E\_ID 不正ID番号  
 13471 ・tskidが有効範囲外【NGKI2659】  
 13472 E\_NOEXS オブジェクト未登録  
 13473 ・対象タスクが未登録 [D] 【NGKI2660】  
 13474 E\_OACV オブジェクトアクセス違反  
 13475 ・対象タスクに対する参照操作が許可されていない [P] 【NGKI2661】  
 13476 E\_MACV メモリアクセス違反  
 13477 ・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  
 13478 いない [P] 【NGKI2662】  
 13479 E\_OBJ オブジェクト状態エラー  
 13480 ・オーバランハンドラが定義されていない【NGKI2663】  
 13481  
 13482 **【機能】**  
 13483  
 13484 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  
 13485 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  
 13486 【NGKI2664】.  
 13487  
 13488 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  
 13489 ずれかの値が返される【NGKI2665】.  
 13490  
 13491 TOVR\_STP 0x01U オーバランハンドラが動作していない状態  
 13492 TOVR\_STA 0x02U オーバランハンドラが動作している状態  
 13493  
 13494 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,  
 13495 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  
 13496 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  
 13497 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  
 13498 態の場合には, leftotmの値は保証されない【NGKI2668】.  
 13499  
 13500 tskidにTSK\_SELF (=0) を指定すると, 自タスクが対象タスクとなる

13501       【NGKI2669】.

13502

13503       【使用上の注意】

13504

13505       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  
13506       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  
13507       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  
13508       ランハンドラの状態が変化している可能性があるためである．

13509

13510       【未決定事項】

13511

13512       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  
13513       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  
13514       照できるとするかどうかは，今後の課題である．

13515

13516       【 $\mu$  ITRON4.0仕様との関係】

13517

13518       TOVR\_STPとTOVR\_STAを値を変更した．

13519

13520

#### 13521       4.7 システム状態管理機能

13522

13523       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  
13524       変更／参照するための機能である．

13525

13526

13527       SAC\_SYS       システム状態のアクセス許可ベクタの設定 [SP]   【NGKI2670】

13528       sac\_sys       システム状態のアクセス許可ベクタの設定 [TPD]  【NGKI2671】

13529

13530       【静的API】

13531       SAC\_SYS({ ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })

13532

13533       【C言語API】

13534       ER ercd = sac\_sys(const ACVCT \*p\_acvct)

13535

13536       【パラメータ】

13537       ACVCT \*       p\_acvct       アクセス許可ベクタを入れたパケットへのポ  
13538       インタ（静的APIを除く）

13539

13540       \*アクセス許可ベクタ（パケットの内容）

13541       ACPTN       acptn1       通常操作1のアクセス許可パターン

13542       ACPTN       acptn2       通常操作2のアクセス許可パターン

13543       ACPTN       acptn3       管理操作のアクセス許可パターン

13544       ACPTN       acptn4       参照操作のアクセス許可パターン

13545

13546       【リターンパラメータ】

13547       ER           ercd           正常終了 (E\_OK) またはエラーコード

13548

13549       【エラーコード】

13550       E\_CTX       コンテキストエラー

13551                   ・非タスクコンテキストからの呼出し [s] 【NGKI2672】  
 13552                   ・CPUロック状態からの呼出し [s] 【NGKI2673】  
 13553           E\_RSATR   予約属性  
 13554                   ・カーネルドメインの囲みの中に記述されていない [S] 【NGKI2674】  
 13555                   ・クラスの囲みの中に記述されている [SM] 【NGKI2675】  
 13556           E\_OACV   オブジェクトアクセス違反  
 13557                   ・カーネルドメイン以外からの呼出し [s] 【NGKI2676】  
 13558           E\_OBJ    オブジェクト状態エラー  
 13559                   ・システム状態のアクセス許可ベクタが設定済み [S] 【NGKI2677】

## 13561   【機能】

13562  
 13563   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  
 13564   パラメータで指定した値に設定する【NGKI2678】。

13565  
 13566   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】。

## 13567   【TOPPERS/HRP2カーネルにおける規定】

13568  
 13569   HRP2カーネルでは、SAC\_SYSのみをサポートする【HRPS0151】。

13570  
 13571   -----  
 13572   rot\_rdq      タスクの優先順位の回転 [T]   【NGKI2680】

13573   irot\_rdq     タスクの優先順位の回転 [I]   【NGKI2681】

## 13574   【C言語API】

13575  
 13576   ER ercd = rot\_rdq(PRI tskpri)

13577   ER ercd = irot\_rdq(PRI tskpri)

## 13578   【パラメータ】

13579           PRI           tskpri           回転対象の優先度（対象優先度）

## 13580   【リターンパラメータ】

13581           ER           ercd           正常終了 (E\_OK) またはエラーコード

## 13582   【エラーコード】

13583           E\_CTX       コンテキストエラー

13584                   ・非タスクコンテキストからの呼出し (rot\_rdqの場合) 【NGKI2682】

13585                   ・タスクコンテキストからの呼出し (irot\_rdqの場合) 【NGKI2683】

13586                   ・CPUロック状態からの呼出し 【NGKI2684】

13587           E\_NOSPT    未サポート機能

13588                   ・条件については機能の項を参照

13589           E\_PAR      パラメータエラー

13590                   ・tskpriが有効範囲外 【NGKI2685】

13591           E\_OACV    オブジェクトアクセス違反

13592                   ・システム状態に対する通常操作1が許可されていない [P]

13593                   【NGKI2686】

## 13594   【機能】

13595  
 13596   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、



13601 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  
 13602 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  
 13603 1つのみの場合には、何も行われずに正常終了する【NGKI2688】。  
 13604  
 13605 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  
 13606 付けられているタスクのみを操作対象とする【NGKI3622】。  
 13607  
 13608 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  
 13609 優先度が対象優先度となる【NGKI2689】。  
 13610  
 13611 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  
 13612 が制約タスクの場合には、E\_NOSPTエラーとなる【NGKI2690】。  
 13613  
 13614 【TOPPERS/SSPカーネルにおける規定】  
 13615  
 13616 SSPカーネルでは、rot\_rdq, irot\_rdqをサポートしない【SSPS0131】。  
 13617 -----  
 13618 mrot\_rdq プロセッサ指定でのタスクの優先順位の回転 [TM] 【NGKI2691】  
 13619 imrot\_rdq プロセッサ指定でのタスクの優先順位の回転 [IM] 【NGKI2692】  
 13620  
 13621 【C言語API】  
 13622 ER ercd = mrot\_rdq(PRI tskpri, ID preid)  
 13623 ER ercd = imrot\_rdq(PRI tskpri, ID preid)  
 13624  
 13625 【パラメータ】  
 13626 PRI tskpri 回転対象の優先度 (対象優先度)  
 13627 ID preid 優先順位の回転対象とするプロセッサのID番号  
 13628  
 13629 【リターンパラメータ】  
 13630 ER ercd 正常終了 (E\_OK) またはエラーコード  
 13631  
 13632 【エラーコード】  
 13633 E\_CTX コンテキストエラー  
 13634 ・非タスクコンテキストからの呼出し (mrot\_rdqの場合)  
 13635 【NGKI2693】  
 13636 ・タスクコンテキストからの呼出し (imrot\_rdqの場合) 【NGKI2694】  
 13637 ・CPUロック状態からの呼出し【NGKI2695】  
 13638 E\_NOSPT 未サポート機能  
 13639 ・条件については機能の項を参照  
 13640 E\_ID 不正ID番号  
 13641 ・preidが有効範囲外【NGKI2696】  
 13642 E\_PAR パラメータエラー  
 13643 ・tskpriが有効範囲外【NGKI2697】  
 13644 E\_OACV オブジェクトアクセス違反  
 13645 ・システム状態に対する通常操作1が許可されていない [P]  
 13646 【NGKI2698】  
 13647  
 13648 【機能】  
 13649  
 13650 preidで指定したプロセッサに割り付けられており、tskpriで指定した優先度

13651 (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  
 13652 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】。  
 13653 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  
 13654 われずに正常終了する【NGKI2700】。

13655

13656 mrot\_rdqにおいて、tskpriにTPRI\_SELF (=0) を指定すると、自タスクのベー  
 13657 ス優先度が対象優先度となる【NGKI2701】。

13658

13659 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  
 13660 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、  
 13661 E\_NOSPTエラーとなる【NGKI2702】。

13662

13663 【TOPPERS/ASPカーネルにおける規定】

13664

13665 ASPカーネルでは、mrot\_rdq, imrot\_rdqをサポートしない【ASPS0188】。

13666

13667 【TOPPERS/HRP2カーネルにおける規定】

13668

13669 HRP2カーネルでは、mrot\_rdq, imrot\_rdqをサポートしない【HRPS0152】。

13670

13671 【TOPPERS/SSPカーネルにおける規定】

13672

13673 SSPカーネルでは、mrot\_rdq, imrot\_rdqをサポートしない【SSPS0132】。

13674

13675 【μITRON4.0仕様との関係】

13676

13677 μITRON4.0仕様に定義されていないサービスコールである。

13678 -----

13679 get\_tid 実行状態のタスクIDの参照 [T] 【NGKI2703】

13680 iget\_tid 実行状態のタスクIDの参照 [I] 【NGKI2704】

13681

13682 【C言語API】

13683 ER ercd = get\_tid(ID \*p\_tskid)

13684 ER ercd = iget\_tid(ID \*p\_tskid)

13685

13686 【パラメータ】

13687 ID \* p\_tskid タスクIDを入れるメモリ領域へのポインタ

13688

13689 【リターンパラメータ】

13690 ER ercd 正常終了 (E\_OK) またはエラーコード

13691 ID tskid タスクID

13692

13693 【エラーコード】

13694 E\_CTX コンテキストエラー

13695 ・非タスクコンテキストからの呼出し (get\_tidの場合) 【NGKI2705】

13696 ・タスクコンテキストからの呼出し (iget\_tidの場合) 【NGKI2706】

13697 ・CPUロック状態からの呼出し【NGKI2707】

13698 E\_MACV メモリアクセス違反

13699 ・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  
 13700 いない [P] 【NGKI2708】

13701

13702     **【機能】**

13703

13704     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  
 13705     たタスクIDは、p\_tskidが指すメモリ領域に返される【NGKI2709】．

13706

13707     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  
 13708     れる【NGKI2710】．

13709

13710     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  
 13711     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  
 13712     【NGKI2711】．

13713

13714     **【TOPPERS/SSPカーネルにおける規定】**

13715

13716     SSPカーネルでは、get\_tidをサポートしない【SSPS0133】．

13717

13718     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】

13719

13720     **【C言語API】**

13721         ER ercd = get\_did(ID \*p\_domid)

13722

13723     **【パラメータ】**

13724         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

13725

13726     **【リターンパラメータ】**

13727         ER             ercd     正常終了 (E\_OK) またはエラーコード

13728         ID             domid     保護ドメインID

13729

13730     **【エラーコード】**

13731         E\_CTX     コンテキストエラー

13732             ・非タスクコンテキストからの呼出し【NGKI2713】

13733             ・CPUロック状態からの呼出し【NGKI2714】

13734         E\_MACV     メモリアクセス違反

13735             ・p\_domidが指すメモリ領域への書き込みアクセスが許可されて  
 13736             いない【NGKI2715】

13737

13738     **【機能】**

13739

13740     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  
 13741     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】．

13742

13743     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  
 13744     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  
 13745     ンのID番号を参照する【NGKI2717】．

13746

13747     **【TOPPERS/ASPカーネルにおける規定】**

13748

13749     ASPカーネルでは、get\_didをサポートしない【ASPS0189】．

13750

13751       【TOPPERS/FMPカーネルにおける規定】

13752

13753       FMPカーネルでは、get\_pidをサポートしない【FMPS0157】。

13754

13755       【TOPPERS/SSPカーネルにおける規定】

13756

13757       SSPカーネルでは、get\_pidをサポートしない【SSPS0134】。

13758

13759       get\_pid       割付けプロセッサのID番号の参照 [TM]   【NGKI2718】

13760       iget\_pid     割付けプロセッサのID番号の参照 [IM]   【NGKI2719】

13761

13762       【C言語API】

13763           ER ercd = get\_pid(ID \*p\_prcid)

13764           ER ercd = iget\_pid(ID \*p\_prcid)

13765

13766       【パラメータ】

13767           ID \*           p\_prcid       プロセッサIDを入れるメモリ領域へのポインタ

13768

13769       【リターンパラメータ】

13770           ER            ercd        正常終了 (E\_OK) またはエラーコード

13771           ID            prcid       プロセッサID

13772

13773       【エラーコード】

13774           E\_CTX        コンテキストエラー

13775                    ・非タスクコンテキストからの呼出し (get\_pidの場合) 【NGKI2720】

13776                    ・タスクコンテキストからの呼出し (iget\_pidの場合) 【NGKI2721】

13777                    ・CPUロック状態からの呼出し 【NGKI2722】

13778           E\_MACV       メモリアクセス違反

13779                    ・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

13780                    いない [P] 【NGKI2723】

13781

13782       【機能】

13783

13784       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。

13785       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

13786       【NGKI2724】。

13787

13788       【使用上の注意】

13789

13790       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

13791       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

13792       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

13793       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。

13794

13795       【TOPPERS/ASPカーネルにおける規定】

13796

13797       ASPカーネルでは、get\_pid, iget\_pidをサポートしない【ASPS0190】。

13798

13799       【TOPPERS/HRP2カーネルにおける規定】

13800

13801 HRP2カーネルでは、get\_pid, iget\_pidをサポートしない【HRPS0153】.

13802

13803 【TOPPERS/SSPカーネルにおける規定】

13804

13805 SSPカーネルでは、get\_pid, iget\_pidをサポートしない【SSPS0135】.

13806

13807 【μ ITRON4.0仕様との関係】

13808

13809 μ ITRON4.0仕様に定義されていないサービスコールである.

13810

13811 loc\_cpu CPUロック状態への遷移 [T] 【NGKI2725】

13812 iloc\_cpu CPUロック状態への遷移 [I] 【NGKI2726】

13813

13814 【C言語API】

13815 ER ercd = loc\_cpu()

13816 ER ercd = iloc\_cpu()

13817

13818 【パラメータ】

13819 なし

13820

13821 【リターンパラメータ】

13822 ER ercd 正常終了 (E\_OK) またはエラーコード

13823

13824 【エラーコード】

13825 E\_CTX コンテキストエラー

13826 ・非タスクコンテキストからの呼出し (loc\_cpuの場合) 【NGKI2727】

13827 ・タスクコンテキストからの呼出し (iloc\_cpuの場合) 【NGKI2728】

13828 E\_OACV オブジェクトアクセス違反

13829 ・システム状態に対する通常操作2が許可されていない

13830 (loc\_cpuの場合) [P] 【NGKI2729】

13831

13832 【機能】

13833

13834 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

13835 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】.

13836

13837 unl\_cpu CPUロック状態の解除 [T] 【NGKI2732】

13838 iunl\_cpu CPUロック状態の解除 [I] 【NGKI2733】

13839

13840 【C言語API】

13841 ER ercd = unl\_cpu()

13842 ER ercd = iunl\_cpu()

13843

13844 【パラメータ】

13845 なし

13846

13847 【リターンパラメータ】

13848 ER ercd 正常終了 (E\_OK) またはエラーコード

13849

13850 【エラーコード】

13851           E\_CTX           コンテキストエラー  
 13852                           ・非タスクコンテキストからの呼出し (unl\_cpuの場合) 【NGKI2734】  
 13853                           ・タスクコンテキストからの呼出し (iunl\_cpuの場合) 【NGKI2735】  
 13854           E\_OACV           オブジェクトアクセス違反  
 13855                           ・システム状態に対する通常操作2が許可されていない  
 13856                           (unl\_cpuの場合) [P] 【NGKI2736】  
 13857  
 13858       **【機能】**  
 13859  
 13860       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】。  
 13861       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  
 13862       【NGKI2738】。  
 13863  
 13864       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  
 13865       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、  
 13866       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  
 13867       正常終了する）【NGKI2739】。  
 13868  
 13869       **【補足説明】**  
 13870  
 13871       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  
 13872       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  
 13873       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  
 13874       チンの実行が開始される可能性がある。  
 13875       -----  
 13876       dis\_dsp       ディスパッチの禁止 [T] 【NGKI2740】  
 13877  
 13878       **【C言語API】**  
 13879       ER ercd = dis\_dsp()  
 13880  
 13881       **【パラメータ】**  
 13882       なし  
 13883  
 13884       **【リターンパラメータ】**  
 13885       ER           ercd           正常終了 (E\_OK) またはエラーコード  
 13886  
 13887       **【エラーコード】**  
 13888       E\_CTX           コンテキストエラー  
 13889                           ・非タスクコンテキストからの呼出し【NGKI2741】  
 13890                           ・CPUロック状態からの呼出し【NGKI2742】  
 13891       E\_OACV           オブジェクトアクセス違反  
 13892                           ・システム状態に対する通常操作1が許可されていない [P]  
 13893                           【NGKI2743】  
 13894  
 13895       **【機能】**  
 13896  
 13897       ディスパッチ禁止フラグをセットし、ディスパッチ禁止状態へ遷移する  
 13898       【NGKI2744】。ディスパッチ禁止状態で呼び出した場合には、何も行われずに  
 13899       正常終了する【NGKI2745】。  
 13900       -----

13901    ena\_dsp      ディスパッチの許可 [T]   【NGKI2746】  
13902  
13903    【C言語API】  
13904      ER ercd = ena\_dsp()  
13905  
13906    【パラメータ】  
13907      なし  
13908  
13909    【リターンパラメータ】  
13910      ER            ercd            正常終了 (E\_OK) またはエラーコード  
13911  
13912    【エラーコード】  
13913      E\_CTX          コンテキストエラー  
13914                      ・非タスクコンテキストからの呼出し 【NGKI2747】  
13915                      ・CPUロック状態からの呼出し 【NGKI2748】  
13916      E\_OACV          オブジェクトアクセス違反  
13917                      ・システム状態に対する通常操作1が許可されていない [P]  
13918                      【NGKI2749】  
13919  
13920    【機能】  
13921  
13922    ディスパッチ禁止フラグをクリアし、ディスパッチ許可状態へ遷移する  
13923    【NGKI2750】．ディスパッチ許可状態で呼び出した場合には、何も行われずに  
13924    正常終了する 【NGKI2751】．  
13925  
13926    【補足説明】  
13927  
13928    ディスパッチ許可状態へ遷移した結果、ディスパッチ保留状態が解除され、ディ  
13929    スパッチが起こる可能性がある．  
13930  
13931    sns\_ctx      コンテキストの参照 [TI]   【NGKI2752】  
13932  
13933    【C言語API】  
13934      bool\_t state = sns\_ctx()  
13935  
13936    【パラメータ】  
13937      なし  
13938  
13939    【リターンパラメータ】  
13940      bool\_t    state            コンテキスト  
13941  
13942    【機能】  
13943  
13944    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．  
13945  
13946    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  
13947    ストから呼び出した場合にはfalseが返る 【NGKI2753】．  
13948  
13949    sns\_loc      CPUロック状態の参照 [TI]   【NGKI2754】  
13950

13951     **【C言語API】**  
13952         bool\_t state = sns\_loc()  
13953  
13954     **【パラメータ】**  
13955         なし  
13956  
13957     **【リターンパラメータ】**  
13958         bool\_t state         CPUロックフラグ  
13959  
13960     **【機能】**  
13961  
13962     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。  
13963  
13964     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  
13965     出した場合にはfalseが返る【NGKI2755】。  
13966     -----  
13967     sns\_dsp         ディスパッチ禁止状態の参照 [TI] 【NGKI2756】  
13968  
13969     **【C言語API】**  
13970         bool\_t state = sns\_dsp()  
13971  
13972     **【パラメータ】**  
13973         なし  
13974  
13975     **【リターンパラメータ】**  
13976         bool\_t state         ディスパッチ禁止フラグ  
13977  
13978     **【機能】**  
13979  
13980     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。  
13981  
13982     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  
13983     状態で呼び出した場合にはfalseが返る【NGKI2757】。  
13984     -----  
13985     sns\_dpn         ディスパッチ保留状態の参照 [TI] 【NGKI2758】  
13986  
13987     **【C言語API】**  
13988         bool\_t state = sns\_dpn()  
13989  
13990     **【パラメータ】**  
13991         なし  
13992  
13993     **【リターンパラメータ】**  
13994         bool\_t state         ディスパッチ保留状態  
13995  
13996     **【機能】**  
13997  
13998     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。  
13999  
14000     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



14001 状態でない状態で呼び出した場合にはfalseが返る【NGKI2759】.

14002 -----

14003 sns\_ker      カーネル非動作状態の参照 [TI] 【NGKI2760】

14004

14005 【C言語API】

14006      bool\_t state = sns\_ker()

14007

14008 【パラメータ】

14009      なし

14010

14011 【リターンパラメータ】

14012      bool\_t          state          カーネル非動作状態

14013

14014 【機能】

14015

14016 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.

14017

14018 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  
14019 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  
14020 ネルの動作中に呼び出した場合にはfalseが返る【NGKI2761】.

14021

14022 【使用方法】

14023

14024 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  
14025 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  
14026 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  
14027 合の動作は保証されない.

14028

14029 【使用上の注意】

14030

14031 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.

14032

14033 【μ ITRON4.0仕様との関係】

14034

14035 μ ITRON4.0仕様に定義されていないサービスコールである.

14036 -----

14037 ext\_ker      カーネルの終了 [TI] 【NGKI2762】

14038

14039 【C言語API】

14040      ER ercd = ext\_ker()

14041

14042 【パラメータ】

14043      なし

14044

14045 【リターンパラメータ】

14046      ER          ercd          エラーコード

14047

14048 【エラーコード】

14049      E\_SYS          システムエラー

14050          ・カーネルの誤動作【NGKI2763】

14051           E\_OACV       オブジェクトアクセス違反  
14052                    ・カーネルドメイン以外からの呼出し [P] 【NGKI2764】  
14053  
14054       **【機能】**  
14055  
14056       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」  
14057       の節を参照すること．  
14058  
14059       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】．  
14060  
14061       **【μ ITRON4.0仕様との関係】**  
14062  
14063       μ ITRON4.0仕様に定義されていないサービスコールである．  
14064       -----  
14065       ref\_sys       システムの状態参照 [T]  
14066  
14067       **【C言語API】**  
14068           ER ercd = ref\_sys(T\_RSYS \*pk\_rsys)  
14069  
14070       ☆未完成  
14071  
14072       **【TOPPERS/ASPカーネルにおける規定】**  
14073  
14074       ASPカーネルでは，ref\_sysをサポートしない．  
14075  
14076       **【TOPPERS/FMPカーネルにおける規定】**  
14077  
14078       FMPカーネルでは，ref\_sysをサポートしない．  
14079  
14080       **【TOPPERS/HRP2カーネルにおける規定】**  
14081  
14082       HRP2カーネルでは，ref\_sysをサポートしない．  
14083  
14084       **【TOPPERS/SSPカーネルにおける規定】**  
14085  
14086       SSPカーネルでは，ref\_sysをサポートしない．  
14087       -----  
14088  
14089       4.8 メモリオブジェクト管理機能  
14090  
14091       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  
14092       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  
14093       をサポートしない．  
14094  
14095       [メモリアリージョン]  
14096  
14097       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  
14098       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，  
14099       メモリアリージョン名によって識別する【NGKI2766】．  
14100

- 14101 各メモリリージョンが持つ情報は次の通り【NGKI2767】.
- 14102
- 14103 ・先頭番地
  - 14104 ・サイズ
  - 14105 ・メモリリージョン属性
- 14106
- 14107 メモリリージョンの先頭番地とサイズには、ターゲット定義の制約が課せられ
- 14108 る場合がある【NGKI2768】.
- 14109
- 14110 メモリリージョン属性には、次の属性を指定することができる【NGKI3256】.
- 14111
- 14112 TA\_NOWRITE 0x01U 書込みアクセス禁止
- 14113
- 14114 ターゲットによっては、ターゲット定義のメモリリージョン属性を指定できる
- 14115 場合がある【NGKI2771】.
- 14116
- 14117 標準メモリリージョンとは、ATT\_MOD/ATA\_MODによって、オブジェクトモジュール
- 14118 に含まれる標準のセクションが配置されるメモリリージョンである. 標準メ
- 14119 モリリージョンには、標準のセクションの中で、書込みアクセスを行わないも
- 14120 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
- 14121 標準RAMリージョンが含まれる.
- 14122
- 14123 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
- 14124 記述された場合に適用される共通の標準メモリリージョンに加えて、クラス毎
- 14125 の標準メモリリージョンを定義することができる【NGKI3257】.
- 14126
- 14127 標準メモリリージョン（マルチプロセッサ対応カーネルでは、共通の標準メモ
- 14128 リリージョン）は、必ず定義しなければならない. 定義しない場合には、コン
- 14129 フィギュレータがエラーを報告する【NGKI3259】.
- 14130
- 14131 [メモリオブジェクト]
- 14132
- 14133 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
- 14134 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
- 14135 て識別する【NGKI2772】.
- 14136
- 14137 各メモリオブジェクトが持つ情報は次の通り【NGKI2773】.
- 14138
- 14139 ・先頭番地
  - 14140 ・サイズ
  - 14141 ・メモリオブジェクト属性
  - 14142 ・アクセス許可ベクタ
  - 14143 ・属する保護ドメイン
  - 14144 ・属するクラス（マルチプロセッサ対応カーネルの場合）
- 14145
- 14146 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
- 14147 れる【NGKI2774】.
- 14148
- 14149 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】.
- 14150

14151	TA_NOWRITE	0x01U	書込みアクセス禁止
14152	TA_NOREAD	0x02U	読出しアクセス禁止
14153	TA_EXEC	0x04U	実行アクセス許可
14154	TA_MEMINI	0x08U	メモリの初期化を行う
14155	TA_MEMPRSV	0x10U	メモリの初期化を行わない
14156	TA_SDATA	0x20U	ショートデータ領域に配置
14157	TA_UNCACHE	0x40U	キャッシュ禁止
14158	TA_IODEV	0x80U	周辺デバイスの領域
14159			
14160	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		
14161	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		
14162	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】。		
14163	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		
14164	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・		
14165	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		
14166	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		
14167	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		
14168	合である【NGKI2778】。		
14169			
14170	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		
14171	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、		
14172	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		
14173	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		
14174	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		
14175	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		
14176	の指定が無視されるかは、ターゲット定義である【NGKI2782】。		
14177			
14178	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		
14179	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		
14180	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		
14181	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)		
14182	される【NGKI2784】。		
14183			
14184	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、		
14185	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		
14186	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】。		
14187			
14188	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		
14189	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		
14190	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		
14191	【NGKI2789】。また、ターゲットによっては、TA_NOWRITEを指定した場合に、		
14192	TA_SDATAが無視される場合がある【NGKI2790】。		
14193			
14194	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、		
14195	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		
14196	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		
14197	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		
14198	プロセッサでのTA_UNCACHE) では、これらの属性は無視される【NGKI2792】。		
14199	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		
14200	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		

14201 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】.

14202

14203 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  
14204 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、  
14205 次の属性を予約している【NGKI2795】.

14206

14207 TA\_WTHROUGH ライトスルーキャッシュを用いる

14208

14209 [カーネル構成マクロ]

14210

14211 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.

14212

14213 TOPPERS\_SUPPORT\_ATT\_MOD ATT\_MOD/ATA\_MODがサポートされている  
14214 【NGKI2796】

14215 TOPPERS\_SUPPORT\_ATT\_PMA ATT\_PMA/ATA\_PMA/att\_pmaがサポートさ  
14216 れている【NGKI2797】

14217

14218 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  
14219 め、サポートされているかを判定するには、TOPPERS\_SUPPORT\_DYNAMIC\_CREと  
14220 TOPPERS\_SUPPORT\_ATT\_PMAの両方が定義されていることをチェックする必要がある  
14221 【NGKI2798】.

14222

14223 【補足説明】

14224

14225 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  
14226 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.

14227

14228 【TOPPERS/ASPカーネルにおける規定】

14229

14230 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】.

14231

14232 【TOPPERS/FMPカーネルにおける規定】

14233

14234 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】.

14235

14236 【TOPPERS/HRP2カーネルにおける規定】

14237

14238 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】.

14239

14240 【TOPPERS/SSPカーネルにおける規定】

14241

14242 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】.

14243

14244 【μITRON4.0/PX仕様との関係】

14245

14246 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  
14247 して廃止した. TA\_ROはTA\_NOWRITEに改名し, TA\_NOREAD, TA\_EXEC, TA\_MEMINI,  
14248 TA\_MEMPRSV, TA\_IODEVを追加した. また, TA\_UNCACHEの値を変更し, ターゲッ  
14249 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.

14250

14251 メモリリージョンは、 $\mu$ ITRON4.0/PX仕様にはない概念である。

14252

14253 **【仕様決定の理由】**

14254

14255 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  
14256 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  
14257 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  
14258 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  
14259 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  
14260 ため、TA\_IODEV属性を用意することにした。

14261

14262 ATT\_REG      メモリリージョンの登録 [SP] **【NGKI2799】**

14263

14264 **【静的API】**

14265     ATT\_REG(“メモリリージョン名”, { ATR regatr, void \*base, SIZE size })

14266

14267 **【パラメータ】**

14268     “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

14269     ATR                    regatr      メモリリージョン属性

14270     void \*                base      登録するメモリリージョンの先頭番地

14271     SIZE                  size      登録するメモリリージョンのサイズ (バイト数)

14272

14273 **【エラーコード】**

14274     E\_RSATR      予約属性

14275                    • regatrが無効 **【NGKI2800】**

14276                    • 保護ドメインの囲みの中に記述されている **【NGKI2814】**

14277                    • クラスの囲みの中に記述されている [M] **【NGKI3260】**

14278     E\_PAR      パラメータエラー

14279                    • sizeが0以下 **【NGKI2816】**

14280                    • その他の条件については機能の項を参照

14281     E\_OBJ      オブジェクト状態エラー

14282                    • 登録済みのメモリリージョンの再登録 **【NGKI2801】**

14283                    • その他の条件については機能の項を参照

14284

14285 **【機能】**

14286

14287 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

14288 リージョンを登録する。具体的な振舞いは以下の通り。

14289

14290 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

14291 **【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

14292 ジョン属性が設定される **【NGKI2803】**。

14293

14294 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

14295 ラメータである **【NGKI2804】**。

14296

14297 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

14298 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

14299 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

14300 となる **【NGKI2817】**。

14301  
14302     **【 $\mu$  ITRON4.0/PX仕様との関係】**  
14303  
14304      $\mu$  ITRON4.0/PX仕様に定義されていない静的APIである。  
14305  
-----  
14306     DEF\_SRG     標準メモリリージョンの定義 [SP]     **【NGKI3261】**  
14307  
14308     **【静的API】**  
14309         DEF\_SRG("標準ROMリージョン名", "標準RAMリージョン名")  
14310  
14311     **【パラメータ】**  
14312         "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  
14313                                     指定する文字列  
14314         "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  
14315                                     指定する文字列  
14316  
14317     **【エラーコード】**  
14318         E\_RSATR     予約属性  
14319                     ・保護ドメインの囲みの中に記述されている **【NGKI3262】**  
14320         E\_OBJ     オブジェクト状態エラー  
14321                     ・標準メモリリージョンが定義済み **【NGKI3263】**  
14322                     ・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  
14323                         **【NGKI3264】**  
14324                     ・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  
14325                         **【NGKI3272】**  
14326                     ・その他の条件については機能の項を参照  
14327  
14328     **【機能】**  
14329  
14330     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  
14331     **【NGKI3265】**。  
14332  
14333     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、  
14334     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  
14335     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  
14336     **【NGKI3266】**。  
14337  
14338     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。  
14339     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  
14340     場合には、E\_OBJエラーとなる **【NGKI3268】**。また、標準RAMリージョンは、  
14341     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  
14342     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、  
14343     E\_OBJエラーとなる **【NGKI3270】**。  
14344  
14345     **【 $\mu$  ITRON4.0/PX仕様との関係】**  
14346  
14347      $\mu$  ITRON4.0/PX仕様に定義されていない静的APIである。  
14348  
-----  
14349     ATT\_SEC     セクションの登録 [SP]     **【NGKI2818】**  
14350     ATA\_SEC     セクションの登録（アクセス許可ベクタ付き） [SP]     **【NGKI2819】**

14351

14352

**【静的API】**

14353

ATT\_SEC("セクション名", { ATR mematr, "メモリリージョン名" })

14354

ATA\_SEC("セクション名", { ATR mematr, "メモリリージョン名" },

14355

{ ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })

14356

14357

**【パラメータ】**

14358

"セクション名" 登録するセクションを指定する文字列

14359

ATR mematr メモリオブジェクト属性

14360

"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

14361

する文字列

14362

14363

**\* アクセス許可ベクタ (パケットの内容)**

14364

ACPTN acptn1 通常操作1のアクセス許可パターン

14365

ACPTN acptn2 通常操作2のアクセス許可パターン

14366

ACPTN acptn3 管理操作のアクセス許可パターン

14367

ACPTN acptn4 参照操作のアクセス許可パターン

14368

14369

**【エラーコード】**

14370

E\_RSATR 予約属性

14371

・ mematrが無効【NGKI2820】

14372

・ その他の条件については機能の項を参照

14373

E\_NOSPT 未サポート機能

14374

・ 条件については機能の項を参照

14375

E\_PAR パラメータエラー

14376

・ 条件については機能の項を参照

14377

E\_OBJ オブジェクト状態エラー

14378

・ 登録済みのセクションの再登録【NGKI2821】

14379

・ 指定したメモリリージョンが未登録【NGKI2822】

14380

14381

**【機能】**

14382

14383

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

14384

する。具体的な振舞いは以下の通り。

14385

14386

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、

14387

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

14388

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、

14389

mematrで指定したメモリオブジェクト属性が設定される【NGKI2824】。

14390

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

14391

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

14392

る【NGKI2825】。

14393

14394

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

14395

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

14396

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

14397

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、

14398

同じ振舞いとなる）【NGKI2786】。

14399

14400

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



- 14401 場合には、E\_RSATRエラーとなる【NGKI2828】.
- 14402
- 14403 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
- 14404 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
- 14405 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】.
- 14406
- 14407 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～
- 14408 acptn4は整数定数式パラメータである【NGKI2830】.
- 14409
- 14410 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
- 14411 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
- 14412 合には、E\_NOSPTエラーとなる【NGKI2832】.
- 14413
- 14414 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、
- 14415 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
- 14416 となる【NGKI2834】.
- 14417
- 14418 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
- 14419 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
- 14420 として指定した場合には、E\_PARエラーとなる【NGKI2836】.
- 14421
- 14422 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
- 14423 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
- 14424 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
- 14425 る【NGKI2837】.
- 14426
- 14427 【μ ITRON4.0/PX仕様との関係】
- 14428
- 14429 μ ITRON4.0/PX仕様に定義されていない静的APIである.
- 14430 -----
- 14431 LNK\_SEC セクションの配置 [SP] 【NGKI2838】
- 14432
- 14433 【静的API】
- 14434 LNK\_SEC("セクション名", { "メモリリージョン名" })
- 14435
- 14436 【パラメータ】
- 14437 "セクション名" 配置するセクションを指定する文字列
- 14438 "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
- 14439 する文字列
- 14440
- 14441 【エラーコード】
- 14442 E\_RSATR 予約属性
- 14443 ・条件については機能の項を参照
- 14444 E\_PAR パラメータエラー
- 14445 ・条件については機能の項を参照
- 14446 E\_OBJ オブジェクト状態エラー
- 14447 ・登録済みのセクションの再登録【NGKI2839】
- 14448 ・指定したメモリリージョンが未登録【NGKI2840】
- 14449
- 14450 【機能】

14451  
 14452 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、  
 14453 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する【NGKI2841】。  
 14454  
 14455 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  
 14456 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】。  
 14457  
 14458 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  
 14459 ラス専用のメモリリージョンの場合で、LNK\_SECをクラスの囲みの外に記述する  
 14460 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  
 14461 【NGKI2844】。  
 14462  
 14463 【使用上の注意】  
 14464  
 14465 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  
 14466 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。  
 14467  
 14468 【μ ITRON4.0/PX仕様との関係】  
 14469  
 14470 μ ITRON4.0/PX仕様に定義されていない静的APIである。  
 14471 -----  
 14472 ATT\_MOD オブジェクトモジュールの登録 [SP] 【NGKI2845】  
 14473 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]  
 14474 【NGKI2846】  
 14475  
 14476 【静的API】  
 14477 ATT\_MOD("オブジェクトモジュール名")  
 14478 ATA\_MOD("オブジェクトモジュール名",  
 14479 { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 } )  
 14480  
 14481 【パラメータ】  
 14482 "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  
 14483 定する文字列  
 14484  
 14485 \* アクセス許可ベクタ（パケットの内容）  
 14486 ACPTN acptn1 通常操作1のアクセス許可パターン  
 14487 ACPTN acptn2 通常操作2のアクセス許可パターン  
 14488 ACPTN acptn3 管理操作のアクセス許可パターン  
 14489 ACPTN acptn4 参照操作のアクセス許可パターン  
 14490  
 14491 【エラーコード】  
 14492 E\_RSATR 予約属性  
 14493 ・ mematrが無効【NGKI2847】  
 14494 E\_NOSPT 未サポート機能  
 14495 ・ 条件については機能の項を参照  
 14496 E\_OBJ オブジェクト状態エラー  
 14497 ・ 登録済みのオブジェクトモジュールの再登録【NGKI2848】  
 14498  
 14499 【機能】  
 14500

14501 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  
14502 ネルに登録する。具体的な振舞いは以下の通り。  
14503  
14504 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  
14505 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  
14506 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  
14507 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  
14508 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  
14509 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  
14510 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  
14511 定した値に設定される【NGKI2851】。  
14512  
14513 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  
14514 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】。  
14515 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、  
14516 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  
14517 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  
14518 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  
14519 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、  
14520 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  
14521 リージョンに配置される場合がある【NGKI3271】。  
14522  
14523 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  
14524 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  
14525 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】。  
14526  
14527 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  
14528 パラメータである【NGKI2856】。  
14529  
14530 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  
14531 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  
14532 に違反した場合には、E\_NOSPTエラーとなる【NGKI2858】。  
14533  
14534 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  
14535 【NGKI2859】。ATT\_MOD／ATA\_MODがサポートされている場合には、  
14536 TOPPERS\_SUPPORT\_ATT\_MODがマクロ定義される【NGKI2860】。サポートされてい  
14537 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  
14538 を報告する【NGKI2861】。  
14539  
14540 【補足説明】  
14541  
14542 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  
14543 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  
14544 いる必要がある。  
14545  
14546 【μ ITRON4.0/PX仕様との関係】  
14547  
14548 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  
14549 ンと標準RAMリージョンであることを明確化した。  
14550 -----

14551 ATT\_MEM      メモリオブジェクトの登録 [SP] 【NGKI2862】  
 14552 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】  
 14553 att\_mem      メモリオブジェクトの登録 [TPD] 【NGKI2864】  
 14554  
 14555 **【静的API】**  
 14556      ATT\_MEM({ ATR mematr, void \*base, SIZE size })  
 14557      ATA\_MEM({ ATR mematr, void \*base, SIZE size },  
 14558              { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })  
 14559  
 14560 **【C言語API】**  
 14561      ER ercd = att\_mem(const T\_AMEM \*pk\_amem)  
 14562  
 14563 **【パラメータ】**  
 14564      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  
 14565                                      トへのポインタ（静的APIを除く）  
 14566  
 14567      \*メモリオブジェクトの登録情報（パケットの内容）  
 14568      ATR              mematr      メモリオブジェクト属性  
 14569      void \*              base      登録するメモリ領域の先頭番地  
 14570      SIZE              size      登録するメモリ領域のサイズ（バイト数）  
 14571  
 14572      \*アクセス許可ベクタ（パケットの内容）  
 14573      ACPTN              acptn1      通常操作1のアクセス許可パターン  
 14574      ACPTN              acptn2      通常操作2のアクセス許可パターン  
 14575      ACPTN              acptn3      管理操作のアクセス許可パターン  
 14576      ACPTN              acptn4      参照操作のアクセス許可パターン  
 14577  
 14578 **【リターンパラメータ】**  
 14579      ER              ercd      正常終了（E\_OK）またはエラーコード  
 14580  
 14581 **【エラーコード】**  
 14582      E\_CTX              コンテキストエラー  
 14583                                      ・非タスクコンテキストからの呼出し [s] 【NGKI2865】  
 14584                                      ・CPUロック状態からの呼出し [s] 【NGKI2866】  
 14585      E\_RSATR              予約属性  
 14586                                      ・mematrが無効 【NGKI2867】  
 14587                                      ・属する保護ドメインの指定が有効範囲外 [sP] 【NGKI2868】  
 14588                                      ・属するクラスの指定が有効範囲外 [sM] 【NGKI2869】  
 14589                                      ・その他の条件については機能の項を参照  
 14590      E\_NOSPT              未サポート機能  
 14591                                      ・条件については機能の項を参照  
 14592      E\_PAR              パラメータエラー  
 14593                                      ・sizeが0以下 【NGKI2881】  
 14594                                      ・その他の条件については機能の項を参照  
 14595      E\_OACV              オブジェクトアクセス違反  
 14596                                      ・システム状態に対する管理操作が許可されていない [sP]  
 14597                                      【NGKI2870】  
 14598      E\_MACV              メモリアクセス違反  
 14599                                      ・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  
 14600                                      いない [sP] 【NGKI2871】

14601           E\_OBJ           オブジェクト状態エラー  
14602                           ・条件については機能の項を参照  
14603  
14604           **【機能】**  
14605  
14606           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  
14607           クトを登録する。具体的な振舞いは以下の通り。  
14608  
14609           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  
14610           **【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  
14611           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  
14612           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）  
14613           が、acptn1～acptn4で指定した値に設定される**【NGKI2874】**。  
14614  
14615           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  
14616           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、  
14617           E\_RSATRエラーとなる**【NGKI2876】**。また、mematrにTA\_SDATAを指定することは  
14618           できない。TA\_SDATAを指定した場合には、E\_RSATRエラーとなる**【NGKI3274】**。  
14619  
14620           静的APIにおいては、mematr, size, acptn1～acptn4は整数定数式パラメータ、  
14621           baseは一般定数式パラメータである**【NGKI2877】**。  
14622  
14623           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  
14624           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  
14625           の制限に違反した場合には、E\_NOSPTエラーとなる**【NGKI2879】**。  
14626  
14627           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  
14628           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  
14629           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、  
14630           E\_OBJエラーとなる**【NGKI2882】**。  
14631  
14632           **【使用上の注意】**  
14633  
14634           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  
14635           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  
14636           は、ATT\_SEC/ATA\_SECかATT\_MOD/ATA\_MODを使用することを推奨する。  
14637  
14638           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  
14639           録したメモリア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、  
14640           メモリア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  
14641           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  
14642           なる。  
14643  
14644           **【TOPPERS/HRP2カーネルにおける規定】**  
14645  
14646           HRP2カーネルでは、ATT\_MEMとATA\_MEMのみをサポートする**【HRPS0155】**。  
14647  
14648           **【μITRON4.0/PX仕様との関係】**  
14649  
14650           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

14651 (ata\_mem) は廃止した。

14652

14653 baseやsizeがターゲット定義の制約に合致しない場合、 $\mu$  ITRON4.0/PX仕様では  
14654 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、  
14655 この仕様ではE\_PARエラーとなることとした。

14656

14657 ATT\_PMA 物理メモリ領域の登録 [SP] 【NGKI2883】

14658 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】

14659 att\_pma 物理メモリ領域の登録 [TPD] 【NGKI2885】

14660

#### 14661 【静的API】

14662 ATT\_PMA({ ATR mematr, void \*base, SIZE size, void \*paddr })

14663 ATA\_PMA({ ATR mematr, void \*base, SIZE size, void \*paddr },

14664 { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })

14665

#### 14666 【C言語API】

14667 ER ercd = att\_pma(const T\_APMA \*pk\_apma)

14668

#### 14669 【パラメータ】

14670 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  
14671 のポインタ (静的APIを除く)

14672

14673 \*物理メモリ領域の登録情報 (パケットの内容)

14674 ATR mematr メモリオブジェクト属性

14675 void \* base 登録するメモリ領域の先頭番地

14676 SIZE size 登録するメモリ領域のサイズ (バイト数)

14677 void \* paddr 登録するメモリ領域の物理アドレス空間における  
14678 先頭番地

14679

14680 \*アクセス許可ベクタ (パケットの内容)

14681 ACPTN acptn1 通常操作1のアクセス許可パターン

14682 ACPTN acptn2 通常操作2のアクセス許可パターン

14683 ACPTN acptn3 管理操作のアクセス許可パターン

14684 ACPTN acptn4 参照操作のアクセス許可パターン

14685

#### 14686 【リターンパラメータ】

14687 ER ercd 正常終了 (E\_OK) またはエラーコード

14688

#### 14689 【エラーコード】

14690 E\_CTX コンテキストエラー

14691 ・非タスクコンテキストからの呼出し [s] 【NGKI2886】

14692 ・CPUロック状態からの呼出し [s] 【NGKI2887】

14693 E\_RSATR 予約属性

14694 ・mematrが無効

14695 ・属する保護ドメインの指定が有効範囲外 [sP] 【NGKI2888】

14696 ・属するクラスの指定が有効範囲外 [sM] 【NGKI2889】

14697 ・その他の条件については機能の項を参照

14698 E\_NOSPT 未サポート機能

14699 ・条件については機能の項を参照

14700 E\_PAR パラメータエラー

14701                   ・sizeが0以下【NGKI2901】  
 14702                   ・その他の条件については機能の項を参照  
 14703           E\_OACV   オブジェクトアクセス違反  
 14704                   ・システム状態に対する管理操作が許可されていない〔sP〕  
 14705                   【NGKI2890】  
 14706           E\_MACV   メモリアクセス違反  
 14707                   ・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  
 14708                   いない〔sP〕【NGKI2891】  
 14709           E\_OBJ    オブジェクト状態エラー  
 14710                   ・条件については機能の項を参照  
 14711  
 14712   【機能】  
 14713  
 14714   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  
 14715   クトを登録する。具体的な振舞いは以下の通り。  
 14716  
 14717   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  
 14718   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  
 14719   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  
 14720   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】。  
 14721   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  
 14722   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  
 14723   る【NGKI2894】。  
 14724  
 14725   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  
 14726   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、  
 14727   E\_RSATRエラーとなる【NGKI2896】。  
 14728  
 14729   静的APIにおいては、mematr、size、paddr、acptn1～acptn4は整数定数式パラ  
 14730   メータ、baseは一般定数式パラメータである【NGKI2897】。  
 14731  
 14732   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  
 14733   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  
 14734   の制限に違反した場合には、E\_NOSPTエラーとなる【NGKI2899】。  
 14735  
 14736   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  
 14737   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  
 14738   ブジェクトが、登録済みのメモリオブジェクトと論理アドレス空間においてメ  
 14739   モリ領域が重なる場合には、E\_OBJエラーとなる【NGKI2902】。  
 14740  
 14741   ATT\_PMA／ATA\_PMA／att\_pmaは、MMU（Memory Management Unit）を持つターゲッ  
 14742   トシステムにおいて、ターゲット定義でサポートされる機能である【NGKI2903】。  
 14743   ATT\_PMA／ATA\_PMA／att\_pmaがサポートされている場合には、  
 14744   TOPPERS\_SUPPORT\_ATT\_PMAがマクロ定義される【NGKI2904】。ATT\_PMA／ATA\_PMA  
 14745   がサポートされていない場合にこれらの静的APIを使用すると、コンフィギュレ  
 14746   ータがE\_NOSPTエラーを報告する【NGKI2905】。また、att\_pmaがサポートされて  
 14747   いない場合にatt\_pmaを呼び出すと、E\_NOSPTエラーが返るか、リンク時にエラー  
 14748   となる【NGKI2906】。  
 14749  
 14750   【TOPPERS/HRP2カーネルにおける規定】

14751  
 14752 HRP2カーネルでは、ターゲット定義で、ATT\_PMAとATA\_PMAのみをサポートする  
 14753 【HRPS0156】。  
 14754  
 14755 【μ ITRON4.0/PX仕様との関係】  
 14756  
 14757 μ ITRON4.0/PX仕様に定義されていない静的APIおよびサービスコールである。  
 14758 -----  
 14759 sac\_mem      メモリオブジェクトのアクセス許可ベクタの設定 [TPD] 【NGKI2907】  
 14760  
 14761 【C言語API】  
 14762     ER ercd = sac\_mem(const void \*base, const ACVCT \*p\_acvct)  
 14763  
 14764 【パラメータ】  
 14765     void \*       base       メモリオブジェクトの先頭番地  
 14766     ACVCT \*     p\_acvct    アクセス許可ベクタを入れたパケットへのポ  
 14767                               インタ  
 14768  
 14769     \*アクセス許可ベクタ (パケットの内容)  
 14770     ACPTN       acptn1     通常操作1のアクセス許可パターン  
 14771     ACPTN       acptn2     通常操作2のアクセス許可パターン  
 14772     ACPTN       acptn3     管理操作のアクセス許可パターン  
 14773     ACPTN       acptn4     参照操作のアクセス許可パターン  
 14774  
 14775 【リターンパラメータ】  
 14776     ER           ercd       正常終了 (E\_OK) またはエラーコード  
 14777  
 14778 【エラーコード】  
 14779     E\_CTX       コンテキストエラー  
 14780                ・非タスクコンテキストからの呼出し 【NGKI2908】  
 14781                ・CPUロック状態からの呼出し 【NGKI2909】  
 14782     E\_PAR       パラメータエラー  
 14783                ・baseがメモリオブジェクトの先頭番地でない 【NGKI2910】  
 14784     E\_NOEXS     オブジェクト未登録  
 14785                ・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  
 14786                ていない 【NGKI2911】  
 14787     E\_OACV     オブジェクトアクセス違反  
 14788                ・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  
 14789                ない 【NGKI2912】  
 14790     E\_MACV     メモリアクセス違反  
 14791                ・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  
 14792                いない 【NGKI2913】  
 14793     E\_OBJ       オブジェクト状態エラー  
 14794                ・対象メモリオブジェクトは静的APIで登録された 【NGKI2914】  
 14795  
 14796 【機能】  
 14797  
 14798 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  
 14799 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  
 14800 設定する 【NGKI2915】。



14801  
14802     **【TOPPERS/HRP2カーネルにおける規定】**  
14803  
14804     HRP2カーネルでは、`sac_mem`をサポートしない【HRPS0157】。  
14805  
14806     **【 $\mu$  ITRON4.0/PX仕様との関係】**  
14807  
14808     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  
14809     ることができないこととした。  
14810  
14811      $\mu$  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  
14812     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  
14813     ならないものとした。  
14814     -----  
14815     `det_mem`       メモリオブジェクトの登録解除 [TPD]   **【NGKI2916】**  
14816  
14817     **【C言語API】**  
14818         `ER ercd = det_mem(const void *base)`  
14819  
14820     **【パラメータ】**  
14821         `void *`       `base`           メモリオブジェクトの先頭番地  
14822  
14823     **【リターンパラメータ】**  
14824         `ER`           `ercd`           正常終了 (E\_OK) またはエラーコード  
14825  
14826     **【エラーコード】**  
14827         `E_CTX`       コンテキストエラー  
14828             ・非タスクコンテキストからの呼出し【NGKI2917】  
14829             ・CPUロック状態からの呼出し【NGKI2918】  
14830         `E_PAR`       パラメータエラー  
14831             ・`base`がメモリオブジェクトの先頭番地でない【NGKI2919】  
14832         `E_NOEXS`   オブジェクト未登録  
14833             ・`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  
14834             ていない【NGKI2920】  
14835         `E_OACV`   オブジェクトアクセス違反  
14836             ・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  
14837             ない【NGKI2921】  
14838         `E_OBJ`       オブジェクト状態エラー  
14839             ・対象メモリオブジェクトは静的APIで登録された【NGKI2922】  
14840  
14841     **【機能】**  
14842  
14843     `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  
14844     る【NGKI2923】。  
14845  
14846     **【TOPPERS/HRP2カーネルにおける規定】**  
14847  
14848     HRP2カーネルでは、`det_mem`をサポートしない【HRPS0158】。  
14849  
14850     **【 $\mu$  ITRON4.0/PX仕様との関係】**

14851  
 14852 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  
 14853 いこととした。  
 14854  
 14855  $\mu$  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  
 14856 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  
 14857 ならないものとした。  
 14858 -----  
 14859 prb\_mem      メモリ領域に対するアクセス権のチェック [TP] 【NGKI2924】  
 14860  
 14861 【C言語API】  
 14862     ER ercd = prb\_mem(const void \*base, SIZE size, ID tskid, MODE pmmode)  
 14863  
 14864 【パラメータ】  
 14865     void \*      base            メモリ領域の先頭番地  
 14866     SIZE        size            メモリ領域のサイズ (バイト数)  
 14867     ID           tskid          アクセス元のタスクのID番号  
 14868     MODE        pmmode        アクセスモード  
 14869  
 14870 【リターンパラメータ】  
 14871     ER           ercd           正常終了 (E\_OK) またはエラーコード  
 14872  
 14873 【エラーコード】  
 14874     E\_CTX       コンテキストエラー  
 14875                ・非タスクコンテキストからの呼出し 【NGKI2925】  
 14876     E\_ID        不正ID番号  
 14877                ・tskidが有効範囲外 【NGKI2927】  
 14878     E\_PAR       パラメータエラー  
 14879                ・sizeが0 【NGKI2929】  
 14880                ・その他の条件については機能の項を参照  
 14881     E\_NOEXS    オブジェクト未登録  
 14882                ・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  
 14883                ていない 【NGKI2930】  
 14884                ・tskidで指定したタスクが未登録 [D] 【NGKI3425】  
 14885     E\_OACV     オブジェクトアクセス違反  
 14886                ・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  
 14887                作が許可されていない 【NGKI2931】  
 14888                ・tskidで指定したタスクに対する参照操作が許可されてい  
 14889                ない 【NGKI3426】  
 14890     E\_MACV     メモリアクセス違反  
 14891                ・条件については機能の項を参照  
 14892     E\_OBJ       オブジェクト状態エラー  
 14893                ・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  
 14894                【NGKI2932】  
 14895  
 14896 【機能】  
 14897  
 14898 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  
 14899 域) に対して、pmmodeで指定した種別のアクセスが許可されているかをチェッ  
 14900 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

14901 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、  
14902 E\_MACVが返ることはない【NGKI2934】.

14903  
14904 pmmodeには、TPM\_WRITE (=0x01U) , TPM\_READ (=0x02U) , TPM\_EXEC (=0x04U) のいずれか、またはそれらの内のいくつかのビット毎論理和 (C言語の  
14905 "||") を指定することができる【NGKI2935】. TPM\_WRITE, TPM\_READ, TPM\_EXEC  
14906 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  
14907 クセスが許可されているかをチェックする【NGKI2936】. また、いくつかのビッ  
14908 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  
14909 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  
14910 た場合には、E\_PARエラーとなる【NGKI2938】.

14911  
14912  
14913 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  
14914 アクセスが許可されているかをチェックする【NGKI2939】.

14915  
14916 【μITRON4.0/PX仕様との関係】  
14917

14918 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  
14919 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  
14920 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  
14921 らも呼び出せるものとした.

14922  
14923 【仕様決定の理由】  
14924

14925 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.  
14926 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  
14927 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  
14928 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、  
14929 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  
14930 呼び出せることが必要である.

14931  
14932 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  
14933 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、  
14934 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.

14935 -----  
14936 ref\_mem      メモリオブジェクトの状態参照 [TP]

14937  
14938 【C言語API】  
14939      ER ercd = ref\_mem(const void \*base, T\_RMEM \*pk\_rmem)

14940  
14941 ☆未完成  
14942

14943 【TOPPERS/HRP2カーネルにおける規定】  
14944

14945 HRP2カーネルでは、ref\_memをサポートしない.

14946 -----

14947  
14948 4.9 割込み管理機能  
14949

14950 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

14951 とを推奨する。割込みサービスルーチンをカーネルに登録する場合には、まず、  
 14952 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  
 14953 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  
 14954 みハンドラを経由して呼び出される【NGKI2941】。  
 14955  
 14956 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  
 14957 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  
 14958 【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  
 14959 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  
 14960 ンの属性を設定しておく必要がある【NGKI2943】。  
 14961  
 14962 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、  
 14963 次の属性を指定することができる【NGKI2944】。  
 14964  
 14965       TA\_ENAINT    0x01U    割込み要求禁止フラグをクリア  
 14966       TA\_EDGE      0x02U    エッジトリガ  
 14967  
 14968 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  
 14969 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  
 14970 の属性を予約している【NGKI2946】。  
 14971  
 14972       TA\_POSEDGE       ポジティブエッジトリガ  
 14973       TA\_NEGEDGE       ネガティブエッジトリガ  
 14974       TA\_BOTHEDGE      両エッジトリガ  
 14975       TA\_LOWLEVEL      ローレベルトリガ  
 14976       TA\_HIGHLEVEL     ハイレベルトリガ  
 14977       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  
 14978                            サ対応カーネルの場合）  
 14979  
 14980 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  
 14981 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  
 14982 する【NGKI2947】。  
 14983  
 14984 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、  
 14985 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  
 14986 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  
 14987 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  
 14988 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  
 14989 される【NGKI2949】。  
 14990  
 14991 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  
 14992 る保護ドメインは、カーネルドメインに限られる【NGKI2950】。  
 14993  
 14994 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  
 14995 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  
 14996 【NGKI2952】。  
 14997  
 14998 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】。  
 14999  
 15000       void interrupt\_service\_routine(intptr\_t exinf)

```
15001     {
15002         割込みサービスルーチン本体
15003     }
15004
15005     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】。
15006
15007     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
15008     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
15009     【NGKI2955】。
15010
15011     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
15012     る【NGKI2956】。
15013
15014     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
15015     定義で、次の属性を指定することができる【NGKI2957】。
15016
15017     TA_NONKERNEL    0x02U    カーネル管理外の割込み
15018
15019     TA_NONKERNELを指定しない場合、カーネル管理の割込みとなる【NGKI2958】。
15020     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
15021     を指定できる場合がある【NGKI2959】。
15022
15023     C言語による割込みハンドラの記述形式は次の通り【NGKI2960】。
15024
15025     void interrupt_handler(void)
15026     {
15027         割込みハンドラ本体
15028     }
15029
15030     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。
15031
15032     TMIN_INTPRI     割込み優先度の最小値（最高値）    【NGKI2961】
15033     TMAX_INTPRI     割込み優先度の最大値（最低値，=-1）
15034
15035     TMIN_ISRPRI     割込みサービスルーチン優先度の最小値（=1）【NGKI2962】
15036     TMAX_ISRPRI     割込みサービスルーチン優先度の最大値
15037
15038     TOPPERS_SUPPORT_DIS_INT    dis_intがサポートされている【NGKI2963】
15039     TOPPERS_SUPPORT_ENA_INT    ena_intがサポートされている【NGKI2964】
15040
15041     【使用上の注意】
15042
15043     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
15044     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
15045     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
15046     ずにリターンするように実装すべきである。
15047
15048     【TOPPERS/ASPカーネルにおける規定】
15049
15050     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）
```

15051 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  
 15052 は, TMAX\_ISRPRIを256に拡張する【ASPS0193】.

15053

15054 **【TOPPERS/FMPカーネルにおける規定】**

15055

15056 FMPカーネルでは, 割込みサービ斯拉ーチン優先度の最大値 (=TMAX\_ISRPRI)  
 15057 は16に固定されている【FMPS0159】.

15058

15059 **【TOPPERS/HRP2カーネルにおける規定】**

15060

15061 HRP2カーネルでは, 割込みサービ斯拉ーチン優先度の最大値 (=TMAX\_ISRPRI)  
 15062 は16に固定されている【HRPS0159】.

15063

15064 **【TOPPERS/SSPカーネルにおける規定】**

15065

15066 SSPカーネルでは, 割込みサービ斯拉ーチン優先度の最大値 (=TMAX\_ISRPRI)  
 15067 は16に固定されている【SSPS0137】.

15068

15069 **【 $\mu$  ITRON4.0仕様との関係】**

15070

15071 割込み要求ラインの属性, 割込み優先度, 割込みサービ斯拉ーチン優先度は,  
 15072  $\mu$  ITRON4.0仕様でない概念であり, TMIN\_INTPRI, TMAX\_INTPRI, TMIN\_ISRPRI,  
 15073 TMAX\_ISRPRIは,  $\mu$  ITRON4.0仕様に定義のないカーネル構成マクロである. また,  
 15074 TA\_NONKERNELは,  $\mu$  ITRON4.0仕様に定義のない割込みハンドラ属性である.

15075 -----

15076 CFG\_INT 割込み要求ラインの属性の設定 [S] 【NGKI2965】

15077 cfg\_int 割込み要求ラインの属性の設定 [TD] 【NGKI2966】

15078

15079 **【静的API】**

15080 CFG\_INT(INTNO intno, { ATR intatr, PRI intpri })

15081

15082 **【C言語API】**

15083 ER ercd = cfg\_int(INTNO intno, const T\_CINT \*pk\_cint)

15084

15085 **【パラメータ】**

15086 INTNO intno 割込み番号

15087 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  
 15088 ケットへのポインタ (静的APIを除く)

15089

15090 \*割込み要求ラインの属性の設定情報 (パケットの内容)

15091 ATR intatr 割込み要求ライン属性

15092 PRI intpri 割込み優先度

15093

15094 **【リターンパラメータ】**

15095 ER ercd 正常終了 (E\_OK) またはエラーコード

15096

15097 **【エラーコード】**

15098 E\_CTX コンテキストエラー

15099 ・非タスクコンテキストからの呼出し [s] 【NGKI2967】

15100 ・CPUロック状態からの呼出し [s] 【NGKI2968】

15101           E\_RSATR     予約属性  
 15102                     ・ intatrが無効【NGKI2969】  
 15103                     ・ 属するクラスの指定が有効範囲外〔sM〕【NGKI2970】  
 15104                     ・ クラスの囲みの中に記述されていない〔SM〕【NGKI2971】  
 15105                     ・ その他の条件については機能の項を参照  
 15106           E\_PAR       パラメータエラー  
 15107                     ・ intnoが有効範囲外【NGKI2972】  
 15108                     ・ intpriが有効範囲外【NGKI2973】  
 15109                     ・ その他の条件については機能の項を参照  
 15110           E\_OACV     オブジェクトアクセス違反  
 15111                     ・ システム状態に対する管理操作が許可されていない〔sP〕  
 15112                         【NGKI2974】  
 15113           E\_MACV     メモリアクセス違反  
 15114                     ・ 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  
 15115                         いない〔sP〕【NGKI2975】  
 15116           E\_OBJ       オブジェクト状態エラー  
 15117                     ・ 対象割込み要求ラインに対して属性が設定済み〔S〕【NGKI2976】  
 15118                     ・ その他の条件については機能の項を参照  
 15119  
 15120       **【機能】**  
 15121  
 15122       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  
 15123       メータで指定した属性を設定する【NGKI2977】。  
 15124  
 15125       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  
 15126       た場合にクリアされ、指定しない場合にセットされる【NGKI2978】。  
 15127  
 15128       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  
 15129       【NGKI2979】。  
 15130  
 15131       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  
 15132       が連動して設定される場合がある【NGKI2980】。  
 15133  
 15134       intpriに指定できる値は、基本的には、TMIN\_INTPRI以上、TMAX\_INTPRI以下の  
 15135       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  
 15136       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  
 15137       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  
 15138       ネル管理外の割込み要求ラインを対象として、intpriにTMIN\_INTPRI以上の値を指  
 15139       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  
 15140       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  
 15141       も、E\_OBJエラーとなる【NGKI2984】。  
 15142  
 15143       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  
 15144       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  
 15145       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  
 15146       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  
 15147       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、  
 15148       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  
 15149       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】。  
 15150

15151 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  
15152 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

15153 【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  
15154 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  
15155 E\_RSATRエラーとなる【NGKI2990】。ただし、TA\_DOM(TDOM\_SELF)を指定した場  
15156 合には、指定が無視され、E\_RSATRエラーは検出されない【NGKI2991】。

15157  
15158 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  
15159 対して登録された割込みサービ斯拉ーチン（または対象割込み番号に対応する  
15160 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  
15161 み中にある場合には、E\_RSATRエラーとなる【NGKI2992】。

#### 15162 【補足説明】

15163  
15164  
15165 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  
15166 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  
15167 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、  
15168 変更できないケースが考えられる。

15169  
15170 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  
15171 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  
15172 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  
15173 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  
15174 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  
15175 合には、E\_PARエラーとなる。

#### 15176 【TOPPERS/ASPカーネルにおける規定】

15177  
15178 ASPカーネルでは、CFG\_INTのみをサポートする【ASPS0194】。

#### 15180 【TOPPERS/FMPカーネルにおける規定】

15181  
15182 FMPカーネルでは、CFG\_INTのみをサポートする【FMPS0160】。

#### 15184 【TOPPERS/HRP2カーネルにおける規定】

15185  
15186 HRP2カーネルでは、CFG\_INTのみをサポートする【HRPS0160】。

#### 15188 【TOPPERS/SSPカーネルにおける規定】

15189  
15190 SSPカーネルでは、CFG\_INTのみをサポートする【SSPS0138】。

#### 15192 【μ ITRON4.0仕様との関係】

15193  
15194 μ ITRON4.0仕様に定義されていない静的APIおよびサービスコールである。

15195 -----  
15196  
15197 CRE\_ISR 割込みサービ斯拉ーチンの生成 [S] 【NGKI2993】  
15198 ATT\_ISR 割込みサービ斯拉ーチンの追加 [S] 【NGKI2994】  
15199 acre\_isr 割込みサービ斯拉ーチンの生成 [TD] 【NGKI2995】

15200



15201 **【静的API】**  
 15202 CRE\_ISR(ID isrid, { ATR isratr, intptr\_t exinf,  
 15203 INTNO intno, ISR isr, PRI isrpri })  
 15204 ATT\_ISR({ ATR isratr, intptr\_t exinf, INTNO intno, ISR isr, PRI isrpri })  
 15205  
 15206 **【C言語API】**  
 15207 ER\_ID isrid = acre\_isr(const T\_CISR \*pk\_cisr)  
 15208  
 15209 **【パラメータ】**  
 15210 ID isrid 対象割込みサービスルーチンのID番号 (CRE\_ISR  
 15211 の場合)  
 15212 T\_CISR \* pk\_cisr 割込みサービスルーチンの生成情報を入れたパ  
 15213 ケットへのポインタ (静的APIを除く)  
 15214  
 15215 \* 割込みサービスルーチンの生成情報 (パケットの内容)  
 15216 ATR isratr 割込みサービスルーチン属性  
 15217 intptr\_t exinf 割込みサービスルーチンの拡張情報  
 15218 INTNO intno 割込みサービスルーチンを登録する割込み番号  
 15219 ISR isr 割込みサービスルーチンの先頭番地  
 15220 PRI isrpri 割込みサービスルーチン優先度  
 15221  
 15222 **【リターンパラメータ】**  
 15223 ER\_ID isrid 生成された割込みサービスルーチンのID番号 (正  
 15224 の値) またはエラーコード  
 15225  
 15226 **【エラーコード】**  
 15227 E\_CTX コンテキストエラー  
 15228 ・非タスクコンテキストからの呼出し [s] 【NGKI2996】  
 15229 ・CPUロック状態からの呼出し [s] 【NGKI2997】  
 15230 E\_RSATR 予約属性  
 15231 ・isratrが無効 【NGKI2998】  
 15232 ・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  
 15233 メイン以外 [sP] 【NGKI2999】  
 15234 ・カーネルドメインの囲みの中に記述されていない [SP]  
 15235 【NGKI3000】  
 15236 ・属するクラスの指定が有効範囲外 [sM] 【NGKI3001】  
 15237 ・クラスの囲みの中に記述されていない [SM] 【NGKI3002】  
 15238 ・その他の条件については機能の項を参照  
 15239 E\_PAR パラメータエラー  
 15240 ・intnoが有効範囲外 【NGKI3003】  
 15241 ・isrがプログラムの先頭番地として正しくない 【NGKI3004】  
 15242 ・isrpriが有効範囲外 【NGKI3005】  
 15243 E\_OACV オブジェクトアクセス違反  
 15244 ・システム状態に対する管理操作が許可されていない [sP]  
 15245 【NGKI3006】  
 15246 E\_MACV メモリアクセス違反  
 15247 ・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  
 15248 いない [sP] 【NGKI3007】  
 15249 E\_NOID ID番号不足  
 15250 ・割り付けられる割込みサービスルーチンIDがない [sD]

15251                   【NGKI3008】  
15252           E\_OBJ       オブジェクト状態エラー  
15253                   ・ isridで指定した割込みサービスルーチンが登録済み  
15254                    (CRE\_ISRの場合) 【NGKI3009】  
15255                   ・ その他の条件については機能の項を参照  
15256  
15257       【機能】  
15258  
15259       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  
15260       ビスルーチンを生成する【NGKI3010】。  
15261  
15262       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  
15263       【NGKI3011】。  
15264  
15265       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  
15266       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  
15267       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)  
15268       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  
15269       【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、  
15270       E\_OBJエラーとなる【NGKI3014】。  
15271  
15272       静的APIにおいては、isridはオブジェクト識別名、isratr, intno, isrpriは整  
15273       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】。  
15274  
15275       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  
15276       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  
15277       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  
15278       【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  
15279       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  
15280       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、  
15281       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  
15282       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  
15283       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  
15284       【NGKI3019】。  
15285  
15286       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、  
15287       ターゲット定義である【NGKI3020】。  
15288  
15289       【TOPPERS/ASPカーネルにおける規定】  
15290  
15291       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  
15292       成機能拡張パッケージでは、acre\_isrもサポートする【ASPS0195】。  
15293  
15294       【TOPPERS/FMPカーネルにおける規定】  
15295  
15296       FMPカーネルでは、ATT\_ISRのみをサポートする【FMPS0161】。  
15297  
15298       【TOPPERS/HRP2カーネルにおける規定】  
15299  
15300       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

15301 成機能拡張パッケージでは、`acre_isr`もサポートする【HRPS0208】。

15302

15303 【TOPPERS/SSPカーネルにおける規定】

15304

15305 SSPカーネルでは、`ATT_ISR`のみをサポートする【SSPS0139】。

15306

15307 【 $\mu$ ITRON4.0仕様との関係】

15308

15309 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

15310 込み優先度）を追加した。`CRE_ISR`は、 $\mu$ ITRON4.0仕様に定義されていない静的

15311 APIである。

15312 -----

15313 `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】

15314

15315 【静的API】

15316 `AID_ISR(uint_t noisr)`

15317

15318 【パラメータ】

15319 `uint_t` `noisr` 割付け可能な割込みサービスルーチンIDの数

15320

15321 【エラーコード】

15322 `E_RSATR` 予約属性

15323 ・保護ドメインの囲みの中に記述されている〔P〕【NGKI3439】

15324 ・クラスの囲みの中に記述されていない〔M〕【NGKI3022】

15325 `E_PAR` パラメータエラー

15326 ・`noisr`が負の値【NGKI3287】

15327

15328 【機能】

15329

15330 `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

15331 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

15332 て確保する【NGKI3024】。

15333

15334 `noisr`は整数定数式パラメータである【NGKI3025】。

15335

15336 【TOPPERS/ASPカーネルにおける規定】

15337

15338 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

15339 【ASPS0219】。

15340

15341 【TOPPERS/HRP2カーネルにおける規定】

15342

15343 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

15344 【HRPS0220】。

15345 -----

15346 `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】

15347 `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】

15348

15349 【静的API】

15350 `SAC_ISR(ID isrid, { ACPTN acptn1, ACPTN acptn2,`

15351 ACPTN acptn3, ACPTN acptn4 })

15352

15353 **【C言語API】**

15354 ER ercd = sac\_isr(ID isrid, const ACVCT \*p\_acvct)

15355

15356

15357 **【パラメータ】**

15358	ID	isrid	対象割込みサービスルーチンのID番号
15359	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）

15360

15361

15362 \*アクセス許可ベクタ（パケットの内容）

15363	ACPTN	acptn1	通常操作1のアクセス許可パターン
15364	ACPTN	acptn2	通常操作2のアクセス許可パターン
15365	ACPTN	acptn3	管理操作のアクセス許可パターン
15366	ACPTN	acptn4	参照操作のアクセス許可パターン

15367

15368 **【リターンパラメータ】**

15369	ER	ercd	正常終了（E_OK）またはエラーコード
-------	----	------	---------------------

15370

15371 **【エラーコード】**

15372	E_CTX	コンテキストエラー
15373		・非タスクコンテキストからの呼出し [s] <b>【NGKI3028】</b>
15374		・CPUロック状態からの呼出し [s] <b>【NGKI3029】</b>
15375	E_ID	不正ID番号
15376		・isridが有効範囲外 [s] <b>【NGKI3030】</b>
15377	E_RSATR	予約属性
15378		・対象割込みサービスルーチンが属する保護ドメインの囲みの中に記述されていない [S] <b>【NGKI3031】</b>
15379		・対象割込みサービスルーチンが属するクラスの囲みの中に記述されていない [SM] <b>【NGKI3032】</b>
15380		
15381		
15382	E_NOEXS	オブジェクト未登録
15383		・対象割込みサービスルーチンが未登録 <b>【NGKI3033】</b>
15384	E_OACV	オブジェクトアクセス違反
15385		・対象割込みサービスルーチンに対する管理操作が許可されていない [s] <b>【NGKI3034】</b>
15386		
15387	E_MACV	メモリアクセス違反
15388		・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] <b>【NGKI3035】</b>
15389		
15390	E_OBJ	オブジェクト状態エラー
15391		・対象割込みサービスルーチンは静的APIで生成された [s] <b>【NGKI3036】</b>
15392		
15393		・対象割込みサービスルーチンに対してアクセス許可ベクタが設定済み [S] <b>【NGKI3037】</b>
15394		

15395

15396 **【機能】**

15397

15398 isridで指定した割込みサービスルーチン（対象割込みサービスルーチン）のアクセス許可ベクタ（4つのアクセス許可パターンの組）を、各パラメータで指定した値に設定する **【NGKI3038】**。

15399

15400

15401  
15402 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  
15403 式パラメータである【NGKI3039】。  
15404  
15405 【TOPPERS/HRP2カーネルにおける規定】  
15406  
15407 HRP2カーネルでは、SAC\_ISR、sac\_isrをサポートしない【HRPS0162】。ただし、  
15408 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】。  
15409  
15410 【未決定事項】  
15411  
15412 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  
15413 セス許可ベクタでアクセス保護する方法も考えられる。  
15414 -----  
15415 del\_isr 割込みサービスルーチンの削除 [TD] 【NGKI3040】  
15416  
15417 【C言語API】  
15418 ER ercd = del\_isr(ID isrid)  
15419  
15420 【パラメータ】  
15421 ID isrid 対象割込みサービスルーチンのID番号  
15422  
15423 【リターンパラメータ】  
15424 ER ercd 正常終了 (E\_OK) またはエラーコード  
15425  
15426 【エラーコード】  
15427 E\_CTX コンテキストエラー  
15428 ・非タスクコンテキストからの呼出し【NGKI3041】  
15429 ・CPUロック状態からの呼出し【NGKI3042】  
15430 E\_ID 不正ID番号  
15431 ・isridが有効範囲外【NGKI3043】  
15432 E\_NOEXS オブジェクト未登録  
15433 ・対象割込みサービスルーチンが未登録【NGKI3044】  
15434 E\_OACV オブジェクトアクセス違反  
15435 ・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  
15436 ていない [P] 【NGKI3045】  
15437 E\_OBJ オブジェクト状態エラー  
15438 ・対象割込みサービスルーチンは静的APIで生成された【NGKI3046】  
15439  
15440 【機能】  
15441  
15442 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  
15443 除する。具体的な振舞いは以下の通り。  
15444  
15445 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  
15446 IDが未使用の状態に戻される【NGKI3047】。  
15447  
15448 【TOPPERS/ASPカーネルにおける規定】  
15449  
15450 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

15451 機能拡張パッケージでは、del\_isrをサポートする【ASPS0198】。  
15452  
15453 【TOPPERS/FMPカーネルにおける規定】  
15454  
15455 FMPカーネルでは、del\_isrをサポートしない【FMPS0163】。  
15456  
15457 【TOPPERS/HRP2カーネルにおける規定】  
15458  
15459 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  
15460 機能拡張パッケージでは、del\_isrをサポートする【HRPS0210】。  
15461  
15462 【TOPPERS/SSPカーネルにおける規定】  
15463  
15464 SSPカーネルでは、del\_isrをサポートしない【SSPS0141】。  
15465 -----  
15466 ref\_isr 割込みサービスルーチンの状態参照 [T]  
15467  
15468 【C言語API】  
15469 ER ercd = ref\_isr(ID isrid, T\_RISR \*pk\_risr)  
15470  
15471 ☆未完成  
15472  
15473 【TOPPERS/ASPカーネルにおける規定】  
15474  
15475 ASPカーネルでは、ref\_isrをサポートしない。  
15476  
15477 【TOPPERS/FMPカーネルにおける規定】  
15478  
15479 FMPカーネルでは、ref\_isrをサポートしない。  
15480  
15481 【TOPPERS/HRP2カーネルにおける規定】  
15482  
15483 HRP2カーネルでは、ref\_isrをサポートしない。  
15484  
15485 【TOPPERS/SSPカーネルにおける規定】  
15486  
15487 SSPカーネルでは、ref\_isrをサポートしない。  
15488 -----  
15489 DEF\_INH 割込みハンドラの定義 [S] 【NGKI3048】  
15490 def\_inh 割込みハンドラの定義 [TD] 【NGKI3049】  
15491  
15492 【静的API】  
15493 DEF\_INH(INHNO inhno, { ATR inhatr, INTHDR inthdr })  
15494  
15495 【C言語API】  
15496 ER ercd = def\_inh(INHNO inhno, const T\_DINH \*pk\_dinh)  
15497  
15498 【パラメータ】  
15499 INHNO inhno 割込みハンドラ番号  
15500 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）

15501

15502

15503       \* 割込みハンドラの定義情報（パケットの内容）

15504           ATR           inhatr       割込みハンドラ属性

15505           INTHDR       inthdr       割込みハンドラの先頭番地

15506

15507       **【リターンパラメータ】**

15508           ER           ercd           正常終了（E\_OK）またはエラーコード

15509

15510       **【エラーコード】**

15511           E\_CTX       コンテキストエラー

15512                    ・ 非タスクコンテキストからの呼出し [s]   **【NGKI3050】**

15513                    ・ CPUロック状態からの呼出し [s]   **【NGKI3051】**

15514           E\_RSATR    予約属性

15515                    ・ inhatrが無効 **【NGKI3052】**

15516                    ・ 属するクラスの指定が有効範囲外 [sM]   **【NGKI3053】**

15517                    ・ クラスの囲みの中に記述されていない [SM]   **【NGKI3054】**

15518                    ・ その他の条件については機能の項を参照

15519           E\_PAR       パラメータエラー

15520                    ・ inhnoが有効範囲外 **【NGKI3055】**

15521                    ・ inthdrがプログラムの先頭番地として正しくない **【NGKI3056】**

15522                    ・ その他の条件については機能の項を参照

15523           E\_OACV    オブジェクトアクセス違反

15524                    ・ システム状態に対する管理操作が許可されていない [sP]   **【NGKI3057】**

15525                    **【NGKI3057】**

15526           E\_MACV    メモリアクセス違反

15527                    ・ 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

15528                    いない [sP]   **【NGKI3058】**

15529           E\_OBJ      オブジェクト状態エラー

15530                    ・ 条件については機能の項を参照

15531

15532       **【機能】**

15533

15534       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

15535       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

15536       義する **【NGKI3059】**。ただし、def\_inhにおいてpk\_dinhをNULLにした場合には、

15537       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】**。

15538

15539       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

15540       数式パラメータである **【NGKI3061】**。

15541

15542       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

15543       pk\_dinhをNULL以外にした場合）には、次のエラーが検出される。

15544

15545       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

15546       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

15547       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

15548       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

15549       場合にも、E\_OBJエラーとなる **【NGKI3063】**。

15550

15551 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  
15552 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  
15553 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  
15554 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  
15555 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  
15556 【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  
15557 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  
15558 `TA_NONKERNEL`を指定しない場合には、`E_RSATR`エラーとなる【NGKI3067】。逆に、  
15559 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  
15560 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  
15561 場合には、`E_RSATR`エラーとなる【NGKI3068】。

15562

15563 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  
15564 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  
15565 【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  
15566 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  
15567 `TA_DOM(domid)`を指定した場合には`E_RSATR`エラーとなる【NGKI3071】。ただし、  
15568 `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  
15569 出されない【NGKI3072】。

15570

15571 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  
15572 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、  
15573 `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  
15574 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  
15575 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  
15576 も、`E_RSATR`エラーとなる【NGKI3075】。

15577

15578 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  
15579 場合）で、対象割込みハンドラ番号に対して割込みハンドラが定義されていな  
15580 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  
15581 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  
15582 は、ターゲット定義で`E_OBJ`エラーとなる場合がある【NGKI3077】。

15583

15584 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  
15585 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  
15586 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  
15587 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  
15588 に該当する【NGKI3079】。

15589

15590 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  
15591 は、ターゲット定義である【NGKI3080】。

15592

15593 【TOPPERS/ASPカーネルにおける規定】

15594

15595 ASPカーネルでは、`DEF_INH`のみをサポートする【ASPS0199】。

15596

15597 【TOPPERS/FMPカーネルにおける規定】

15598

15599 FMPカーネルでは、`DEF_INH`のみをサポートする【FMPS0164】。

15600



15601       【TOPPERS/HRP2カーネルにおける規定】  
 15602  
 15603       HRP2カーネルでは、DEF\_INHのみをサポートする【HRPS0164】。  
 15604  
 15605       【TOPPERS/SSPカーネルにおける規定】  
 15606  
 15607       SSPカーネルでは、DEF\_INHのみをサポートする【SSPS0142】。  
 15608  
 15609       【 $\mu$  ITRON4.0仕様との関係】  
 15610  
 15611       inthdrのデータ型をINTHDRに変更した。  
 15612  
 15613       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、  
 15614       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  
 15615       義を解除してから、再度定義する必要がある。  
 15616  
 -----  
 15617       dis\_int       割込みの禁止 [T]   【NGKI3081】  
 15618  
 15619       【C言語API】  
 15620           ER ercd = dis\_int(INTNO intno)  
 15621  
 15622       【パラメータ】  
 15623           INTNO       intno       割込み番号  
 15624  
 15625       【リターンパラメータ】  
 15626           ER        ercd        正常終了 (E\_OK) またはエラーコード  
 15627  
 15628       【エラーコード】  
 15629           E\_CTX       コンテキストエラー  
 15630                    ・非タスクコンテキストからの呼出し【NGKI3082】  
 15631           E\_NOSPT     未サポートエラー  
 15632                    ・条件については機能の項を参照  
 15633           E\_PAR       パラメータエラー  
 15634                    ・intnoが有効範囲外【NGKI3083】  
 15635                    ・その他の条件については機能の項を参照  
 15636           E\_OACV       オブジェクトアクセス違反  
 15637                    ・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]  
 15638                    【NGKI3084】  
 15639           E\_OBJ       オブジェクト状態エラー  
 15640                    ・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  
 15641                    定されていない【NGKI3085】  
 15642  
 15643       【機能】  
 15644  
 15645       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  
 15646       フラグをセットする【NGKI3086】。  
 15647  
 15648       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  
 15649       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  
 15650       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

15651 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  
 15652 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  
 15653 作できない場合が、これに該当する。

15654  
 15655 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  
 15656 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  
 15657 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  
 15658 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。

15659  
 15660 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】。  
 15661 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  
 15662 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、  
 15663 E\_NOSPTエラーが返るか、リンク時にエラーとなる【NGKI3093】。

15664  
 15665 【μITRON4.0仕様との関係】  
 15666

15667 μITRON4.0仕様で実装定義としていたintnoの意味を標準化した。  
 15668

15669 CPUロック状態でも呼び出せるものとした。  
 15670

-----

15671 ena\_int 割込みの許可 [T] 【NGKI3094】

15672

15673 【C言語API】

15674 ER ercd = ena\_int(INTNO intno)

15675

15676 【パラメータ】

15677 INTNO intno 割込み番号

15678

15679 【リターンパラメータ】

15680 ER ercd 正常終了 (E\_OK) またはエラーコード

15681

15682 【エラーコード】

15683 E\_CTX コンテキストエラー

15684 ・非タスクコンテキストからの呼出し【NGKI3095】

15685 E\_NOSPT 未サポートエラー

15686 ・条件については機能の項を参照

15687 E\_PAR パラメータエラー

15688 ・intnoが有効範囲外【NGKI3096】

15689 ・その他の条件については機能の項を参照

15690 E\_OACV オブジェクトアクセス違反

15691 ・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]

15692 【NGKI3097】

15693 E\_OBJ オブジェクト状態エラー

15694 ・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

15695 定されていない【NGKI3098】

15696

15697 【機能】

15698

15699 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  
 15700 フラグをクリアする【NGKI3099】。

15701  
15702 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  
15703 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  
15704 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  
15705 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  
15706 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  
15707 作できない場合が、これに該当する。  
15708  
15709 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  
15710 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  
15711 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  
15712 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。  
15713  
15714 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】。  
15715 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  
15716 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、  
15717 E\_NOSPTエラーが返るか、リンク時にエラーとなる【NGKI3106】。  
15718  
15719 【 $\mu$  ITRON4.0仕様との関係】  
15720  
15721  $\mu$  ITRON4.0仕様で実装定義としていたintnoの意味を標準化した。  
15722  
15723 CPUロック状態でも呼び出せるものとした。  
15724 -----  
15725 ref\_int 割込み要求ラインの参照 [T]  
15726  
15727 【C言語API】  
15728 ER ercd = ref\_int(INTNO intno, T\_RINT \*pk\_rint)  
15729  
15730 ☆未完成  
15731  
15732 【TOPPERS/ASPカーネルにおける規定】  
15733  
15734 ASPカーネルでは、ref\_intをサポートしない。  
15735  
15736 【TOPPERS/FMPカーネルにおける規定】  
15737  
15738 FMPカーネルでは、ref\_intをサポートしない。  
15739  
15740 【TOPPERS/HRP2カーネルにおける規定】  
15741  
15742 HRP2カーネルでは、ref\_intをサポートしない。  
15743  
15744 【TOPPERS/SSPカーネルにおける規定】  
15745  
15746 SSPカーネルでは、ref\_intをサポートしない。  
15747  
15748 【 $\mu$  ITRON4.0仕様との関係】  
15749  
15750  $\mu$  ITRON4.0仕様に定義されていないサービスコールである。

```

15751 -----
15752 chg_ipm      割込み優先度マスクの変更 [T] 【NGKI3107】
15753
15754 【C言語API】
15755     ER ercd = chg_ipm(PRI intpri)
15756
15757 【パラメータ】
15758     PRI          intpri      割込み優先度マスク
15759
15760 【リターンパラメータ】
15761     ER          ercd        正常終了 (E_OK) またはエラーコード
15762
15763 【エラーコード】
15764     E_CTX      コンテキストエラー
15765                ・非タスクコンテキストからの呼出し 【NGKI3108】
15766                ・CPUロック状態からの呼出し 【NGKI3109】
15767     E_PAR      パラメータエラー
15768                ・条件については機能の項を参照
15769     E_OACV     オブジェクトアクセス違反
15770                ・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]
15771                【NGKI3110】
15772
15773 【機能】
15774
15775 割込み優先度マスクを、intpriで指定した値に変更する 【NGKI3111】。
15776
15777 intpriは、TMIN_INTPRI以上、TIPM_ENAALL以下でなければならない。そうでな
15778 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
15779 として、TMIN_INTPRIよりも小さい値を指定できる場合がある 【NGKI3114】。
15780
15781 【補足説明】
15782
15783 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
15784 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
15785 の実行が開始される可能性がある。
15786
15787 【TOPPERS/SSPカーネルにおける規定】
15788
15789 SSPカーネルでは、chg_ipmをサポートしない 【SSPS0143】。
15790
15791 【μITRON4.0仕様との関係】
15792
15793 μITRON4.0仕様では、サービスコールの名称およびパラメータの名称が実装定
15794 義となっているサービスコールである。
15795 -----
15796 get_ipm      割込み優先度マスクの参照 [T] 【NGKI3115】
15797
15798 【C言語API】
15799     ER ercd = get_ipm(PRI *p_intpri)
15800

```

15801      **【パラメータ】**

15802          PRI \*          p\_intpri      割込み優先度マスクを入れるメモリ領域へのポ  
15803                                  インタ

15804

15805      **【リターンパラメータ】**

15806          ER              ercd          エラーコード  
15807          PRI              intpri      割込み優先度マスク

15808

15809      **【エラーコード】**

15810          E\_CTX          コンテキストエラー  
15811                                  ・非タスクコンテキストからの呼出し【NGKI3116】  
15812                                  ・CPUロック状態からの呼出し【NGKI3117】  
15813          E\_OACV          オブジェクトアクセス違反  
15814                                  ・システム状態に対する参照操作が許可されていない [P]  
15815                                  【NGKI3118】  
15816          E\_MACV          メモリアクセス違反  
15817                                  ・p\_intpriが指すメモリ領域への書込みアクセスが許可され  
15818                                  ていない [P] 【NGKI3119】

15819

15820      **【機能】**

15821

15822      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、  
15823      p\_intpriが指すメモリ領域に返される【NGKI3120】．

15824

15825      **【TOPPERS/SSPカーネルにおける規定】**

15826

15827      SSPカーネルでは、get\_ipmをサポートしない【SSPS0144】．

15828

15829      **【μ ITRON4.0仕様との関係】**

15830

15831      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  
15832      義となっているサービスコールである．

15833

15834

15835      **4.10 CPU例外管理機能**

15836

15837      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  
15838      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  
15839      【NGKI3121】．

15840

15841      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  
15842      る【NGKI3122】．

15843

15844      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、  
15845      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】．  
15846      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  
15847      【NGKI3124】．

15848

15849          TA\_DIRECT                  CPU例外ハンドラを直接呼び出す

15850

15851 C言語によるCPU例外ハンドラの記述形式は次の通り【NGKI3125】.

```
15852
15853     void cpu_exception_handler(void *p_excinf)
15854     {
15855         CPU例外ハンドラ本体
15856     }
```

15857  
15858 p\_excinfには、CPU例外の情報を記憶しているメモリ領域の先頭番地が渡される  
15859 【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  
15860 る際に必要となる.

15861 -----  
15862 DEF\_EXC CPU例外ハンドラの定義 [S] 【NGKI3127】

15863 def\_exc CPU例外ハンドラの定義 [TD] 【NGKI3128】

#### 15864 【静的API】

15865 DEF\_EXC(EXCNO excno, { ATR excatr, EXCHDR exchdr })

#### 15866 【C言語API】

15867 ER ercd = def\_exc(EXCNO excno, const T\_DEXC \*pk\_dexc)

#### 15870 【パラメータ】

15871 EXCNO excno CPU例外ハンドラ番号  
15872 T\_DEXC \* pk\_dexc CPU例外ハンドラの定義情報を入れたパケットへ  
15873 のポインタ (静的APIを除く)  
15874

15875 \*CPU例外ハンドラの定義情報 (パケットの内容)

15876 ATR excatr CPU例外ハンドラ属性  
15877 EXCHDR exchdr CPU例外ハンドラの先頭番地  
15878

#### 15879 【リターンパラメータ】

15880 ER ercd 正常終了 (E\_OK) またはエラーコード

#### 15881 【エラーコード】

15882 E\_CTX コンテキストエラー  
15883  
15884  
15885 ・非タスクコンテキストからの呼出し [s] 【NGKI3129】  
15886 ・CPUロック状態からの呼出し [s] 【NGKI3130】  
15887 E\_RSATR 予約属性  
15888  
15889 ・excatrが無効【NGKI3131】  
15890 ・属するクラスの指定が有効範囲外 [sM] 【NGKI3132】  
15891 ・クラスの囲みの中に記述されていない [SM] 【NGKI3133】  
15892 ・その他の条件については機能の項を参照  
15893 E\_PAR パラメータエラー  
15894  
15895 ・excnoが有効範囲外【NGKI3134】  
15896 ・exchdrがプログラムの先頭番地として正しくない【NGKI3135】  
15897 E\_OACV オブジェクトアクセス違反  
15898  
15899 ・システム状態に対する管理操作が許可されていない [sP]  
15900 【NGKI3136】  
15901 E\_MACV メモリアクセス違反  
15902  
15903 ・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  
15904 いない [sP] 【NGKI3137】

15901           E\_OBJ           オブジェクト状態エラー  
15902                    ・条件については機能の項を参照  
15903  
15904       **【機能】**  
15905  
15906       exchno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、  
15907       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  
15908       を定義する【NGKI3138】。ただし、def\_excにおいてpk\_dexcをNULLにした場合  
15909       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  
15910       【NGKI3139】。  
15911  
15912       静的APIにおいては、exchnoとexcattrは整数定数式パラメータ、exchnoは一般定  
15913       数式パラメータである【NGKI3140】。  
15914  
15915       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  
15916       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  
15917       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】。  
15918  
15919       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  
15920       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  
15921       【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  
15922       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  
15923       TA\_DOM(domid)を指定した場合にはE\_RSATRエラーとなる【NGKI3144】。ただし、  
15924       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  
15925       出されない【NGKI3145】。  
15926  
15927       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  
15928       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、  
15929       E\_RSATRエラーとなる【NGKI3146】。  
15930  
15931       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  
15932       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  
15933       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  
15934       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  
15935       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】。  
15936  
15937       静的APIにおいて、exchnoが不正である場合にE\_PARエラーが検出されるか否か  
15938       は、ターゲット定義である【NGKI3149】。  
15939  
15940       **【TOPPERS/ASPカーネルにおける規定】**  
15941  
15942       ASPカーネルでは、DEF\_EXCのみをサポートする【ASPS0200】。  
15943  
15944       **【TOPPERS/FMPカーネルにおける規定】**  
15945  
15946       FMPカーネルでは、DEF\_EXCのみをサポートする【FMPS0165】。  
15947  
15948       **【TOPPERS/HRP2カーネルにおける規定】**  
15949  
15950       HRP2カーネルでは、DEF\_EXCのみをサポートする【HRPS0165】。

15951  
15952       **【TOPPERS/SSPカーネルにおける規定】**  
15953  
15954       SSPカーネルでは、DEF\_EXCのみをサポートする【SSPS0145】。  
15955  
15956       **【μITRON4.0仕様との関係】**  
15957  
15958       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、  
15959       E\_OBJエラーとすることにした。  
15960  
-----  
15961       xsns\_dpn      CPU例外発生時のディスパッチ保留状態の参照 [TI]   **【NGKI3150】**  
15962  
15963       **【C言語API】**  
15964               bool\_t stat = xsns\_dpn(void \*p\_excinf)  
15965  
15966       **【パラメータ】**  
15967               void \*           p\_excinf      CPU例外の情報を記憶しているメモリ領域の先頭  
15968                                               番地  
15969  
15970       **【リターンパラメータ】**  
15971               bool\_t           state        ディスパッチ保留状態  
15972  
15973       **【機能】**  
15974  
15975       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。  
15976  
15977  
15978       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  
15979       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  
15980       状態でなかった場合にfalse、そうでない場合にtrueが返る【NGKI3151】。  
15981  
15982       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  
15983       した場合には、trueが返る【NGKI3152】。  
15984  
15985       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  
15986       【NGKI3153】。それ以外の値を渡した場合の動作は保証されない【NGKI3352】。  
15987  
15988       **【使用方法】**  
15989  
15990       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  
15991       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  
15992       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  
15993       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、  
15994       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  
15995       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。  
15996  
15997       **【使用上の注意】**  
15998  
15999       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  
16000       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



16001 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  
16002 味を持たない。  
16003  
16004 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。  
16005  
16006 **【TOPPERS/SSPカーネルにおける規定】**  
16007  
16008 SSPカーネルでは、xsns\_dpnをサポートしない【SSPS0146】。  
16009  
16010 **【μITRON4.0仕様との関係】**  
16011  
16012 μITRON4.0仕様に定義されていないサービスコールである。  
16013  
16014 **【仕様決定の理由】**  
16015  
16016 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  
16017 ことは禁止すべきである。ユーザドメインの実行中は、必ずタスクコンテキス  
16018 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  
16019 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  
16020 的に禁止している。  
16021 -----  
16022 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】  
16023  
16024 **【C言語API】**  
16025 bool\_t stat = xsns\_xpn(void \*p\_excinf)  
16026  
16027 **【パラメータ】**  
16028 void \* p\_excinf CPU例外の情報を記憶しているメモリ領域の先頭  
16029 番地  
16030  
16031 **【リターンパラメータ】**  
16032 bool\_t state タスク例外処理保留状態  
16033  
16034 **【機能】**  
16035  
16036 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  
16037 参照する。具体的な振舞いは以下の通り。  
16038  
16039 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  
16040 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  
16041 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  
16042 【NGKI3155】。  
16043  
16044 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  
16045 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  
16046 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  
16047 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】。  
16048  
16049 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  
16050 した場合には、trueが返る【NGKI3157】。

16051  
16052 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  
16053 【NGKI3158】。

16054  
16055 【使用方法】

16056  
16057 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  
16058 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  
16059 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  
16060 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  
16061 カバリ処理を行うことができる。

16062  
16063 【使用上の注意】

16064  
16065 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  
16066 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  
16067 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  
16068 味を持たない。

16069  
16070 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。

16071  
16072 【TOPPERS/SSPカーネルにおける規定】

16073  
16074 SSPカーネルでは、xsns\_xpnをサポートしない【SSPS0147】。

16075  
16076 【μITRON4.0仕様との関係】

16077  
16078 μITRON4.0仕様に定義されていないサービスコールである。

16079  
16080 【仕様決定の理由】

16081  
16082 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  
16083 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  
16084 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  
16085 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  
16086 的に禁止している。

16087 -----

16088  
16089 4.11 拡張サービスコール管理機能

16090  
16091 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  
16092 で実行すべきルーチンを呼び出すための機能である【NGKI3159】。特権モード  
16093 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、  
16094 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】。

16095  
16096 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  
16097 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  
16098 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  
16099 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  
16100 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、

16101 すべての種別のアクセスを行うことが許可される。

16102

16103 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  
16104 め、拡張サービスコール管理機能をサポートしない【NGKI3163】。

16105

16106 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】。

16107

```
16108     ER_UINT extended_svc(intptr_t par1, intptr_t par2, intptr_t par3,  
16109                           intptr_t par4, intptr_t par5, ID cdmid)
```

16110 {

16111 拡張サービスコール本体

16112 }

16113

16114 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  
16115 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  
16116 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）  
16117 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  
16118 される。

16119

16120 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

16121 【NGKI3166】。

16122

16123 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。

16124

16125 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  
16126 カーネルでは、登録できる拡張サービスコールの数に一  
16127 致）【NGKI3167】

16128

16129 【TOPPERS/ASPカーネルにおける規定】

16130

16131 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】。

16132

16133 【TOPPERS/FMPカーネルにおける規定】

16134

16135 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】。

16136

16137 【TOPPERS/HRP2カーネルにおける規定】

16138

16139 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】。

16140

16141 【TOPPERS/SSPカーネルにおける規定】

16142

16143 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】。

16144

16145 【未決定事項】

16146

16147 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  
16148 は未決定である。

16149

16150 【μITRON4.0仕様との関係】

16151  
 16152 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  
 16153 メータ5個に固定した。  
 16154  
 16155 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  
 16156 番号を渡す機能を追加した。  
 16157  
 16158 TMAX\_FNCDDは、 $\mu$ ITRON4.0仕様に規定されていないカーネル構成マクロである。  
 16159 -----  
 16160 DEF\_SVC      拡張サービスコールの定義 [SP]   【NGKI3168】  
 16161 def\_svc      拡張サービスコールの定義 [TPD]   【NGKI3169】  
 16162  
 16163 【静的API】  
 16164     DEF\_SVC(FN fncd, { ATR svcatr, EXTSVC extsvc, SIZE stksz })  
 16165  
 16166 【C言語API】  
 16167     ER ercd = def\_svc(FN fncd, const T\_DSVC \*pk\_dsvc)  
 16168  
 16169 【パラメータ】  
 16170     FN           fncd           拡張サービスコールの機能コード  
 16171     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  
 16172                               トへのポインタ（静的APIを除く）  
 16173  
 16174     \* 拡張サービスコールの定義情報（パケットの内容）  
 16175     ATR        svcatr        拡張サービスコール属性  
 16176     EXTSVC    extsvc        拡張サービスコールの先頭番地  
 16177     SIZE       stksz        拡張サービスコールで使用するスタックサイズ  
 16178  
 16179 【リターンパラメータ】  
 16180     ER           ercd           正常終了（E\_OK）またはエラーコード  
 16181  
 16182 【エラーコード】  
 16183     E\_CTX       コンテキストエラー  
 16184                ・非タスクコンテキストからの呼出し [s]   【NGKI3170】  
 16185                ・CPUロック状態からの呼出し [s]   【NGKI3171】  
 16186     E\_RSATR     予約属性  
 16187                ・svcatrが無効【NGKI3172】  
 16188                ・その他の条件については機能の項を参照  
 16189     E\_PAR       パラメータエラー  
 16190                ・fncdが0または負の値【NGKI3173】  
 16191                ・fncdがTMAX\_FNCDDよりも大きい [s]   【NGKI3174】  
 16192                ・extsvcがプログラムの先頭番地として正しくない【NGKI3175】  
 16193                ・stkszが負の値 [S]   【NGKI3290】  
 16194     E\_OACV     オブジェクトアクセス違反  
 16195                ・システム状態に対する管理操作が許可されていない [s]  
 16196                               【NGKI3176】  
 16197     E\_MACV     メモリアクセス違反  
 16198                ・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  
 16199                               いない [s]   【NGKI3177】  
 16200     E\_OBJ       オブジェクト状態エラー

・条件については機能の項を参照

## 【機能】

fncdで指定した機能コード（対象機能コード）に対して、各パラメータで指定した拡張サービスコール定義情報に従って、拡張サービスコールを定義する【NGKI3178】。ただし、def\_svcにおいてpk\_dsvcをNULLにした場合には、対象機能コードに対する拡張サービスコールの定義を解除する【NGKI3179】。

静的APIにおいては、fncd, svcatr, stkszは整数定数式パラメータ、svchdrは一般定数式パラメータである【NGKI3180】。

拡張サービスコールを定義する場合（DEF\_SVCの場合およびdef\_svcにおいてpk\_dsvcをNULL以外にした場合）で、対象機能コードに対してすでに拡張サービスコールが定義されている場合には、E\_OBJエラーとなる【NGKI3181】。

DEF\_SVCは、カーネルドメインの囲みの中に記述しなければならない。そうでない場合には、E\_RSATRエラーとなる【NGKI3183】。また、def\_svcで拡張サービスコールを定義する場合には、拡張サービスコールの属する保護ドメインを設定する必要はなく、拡張サービスコール属性にTA\_DOM(domid)を指定した場合にはE\_RSATRエラーとなる【NGKI3184】。ただし、TA\_DOM(TDOM\_SELF)を指定した場合には、指定が無視され、E\_RSATRエラーは検出されない【NGKI3185】。

マルチプロセッサ対応カーネルでは、DEF\_SVCは、クラスの囲みの外に記述しなければならない。そうでない場合には、E\_RSATRエラーとなる【NGKI3187】。また、def\_svcで拡張サービスコールを定義する場合には、拡張サービスコールの属するクラスを設定する必要はなく、拡張サービスコール属性にTA\_CLS(clsid)を指定した場合にはE\_RSATRエラーとなる【NGKI3188】。ただし、TA\_CLS(TCLS\_SELF)を指定した場合には、指定が無視され、E\_RSATRエラーは検出されない【NGKI3189】。

拡張サービスコールの定義を解除する場合（def\_svcにおいてpk\_dsvcをNULLにした場合）で、対象機能コードに対して拡張サービスコールが定義されていない場合には、E\_OBJエラーとなる【NGKI3190】。

## 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、DEF\_SVCのみをサポートする【HRPS0167】。

## 【μITRON4.0仕様との関係】

拡張サービスコールの定義情報に、stksz（拡張サービスコールで使用するスタックサイズ）を追加した。

extsvcのデータ型を、EXTSVCに変更した。

---

cal\_svc      拡張サービスコールの呼出し〔TIP〕【NGKI3191】

## 【C言語API】

ER\_UINT ercd = cal\_svc(FN fncd, intptr\_t par1, intptr\_t par2,

16251 intptr\_t par3, intptr\_t par4, intptr\_t par5)

16252

16253 **【パラメータ】**

16254	FN	fncd	呼び出す拡張サービスコールの機能コード
16255	intptr_t	par1	拡張サービスコールへの第1パラメータ
16256	intptr_t	par2	拡張サービスコールへの第2パラメータ
16257	intptr_t	par3	拡張サービスコールへの第3パラメータ
16258	intptr_t	par4	拡張サービスコールへの第4パラメータ
16259	intptr_t	par5	拡張サービスコールへの第5パラメータ

16260

16261 **【リターンパラメータ】**

16262	ER_UINT	ercd	正常終了（正の値または0）またはエラーコード
-------	---------	------	------------------------

16263

16264 **【エラーコード】**

16265	E_SYS	システムエラー
16266		・条件については機能の項を参照
16267	E_RSFN	予約機能コード
16268		・fncdが0または負の値【NGKI3192】
16269		・fncdがTMAX_FNCDよりも大きい【NGKI3193】
16270		・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
16271		定義されていない【NGKI3194】
16272	E_NOMEM	メモリ不足
16273		・条件については機能の項を参照

16274 \* 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。

16275

16276 **【機能】**

16277

16278 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  
16279 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】。

16280

16281 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  
16282 【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  
16283 域が、拡張サービスコールで使用するスタックサイズよりも小さい場合には、  
16284 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  
16285 が上限（=255）を超える場合には、E\_SYSエラーが返る【NGKI3198】。

16286

16287 **【μITRON4.0仕様との関係】**

16288

16289 μITRON4.0仕様では、cal\_svcでカーネルのサービスコールを呼び出せるかどう  
16290 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  
16291 出せないこととした。

16292

16293 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  
16294 クする機能を追加した。

16295

16296 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  
16297 し、cal\_svcから返るエラー（E\_SYS, E\_RSFN, E\_NOMEM）について規定した。

16298

16299 **【仕様決定の理由】**

16300

16301 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  
16302 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  
16303 うにするためである。  
16304 -----

#### 16306 4.12 システム構成管理機能

16308 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  
16309 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  
16310 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。

16312 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  
16313 処理単位が用いるスタック領域である。

16315 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、  
16316 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】。

16318 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  
16319 始の直前に、カーネル非動作状態で実行される【NGKI3200】。

16321 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  
16322 る【NGKI3201】。

16324 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  
16325 チン属性には、TA\_NULLを指定しなければならない【NGKI3203】。

16327 C言語による初期化ルーチンの記述形式は次の通り【NGKI3204】。

```
16329 void initialization_routine(intptr_t exinf)
16330 {
16331     初期化ルーチン本体
16332 }
```

16334 exinfには、初期化ルーチンの拡張情報が渡される【NGKI3205】。

16336 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  
16337 終了の直後に、カーネル非動作状態で実行される【NGKI3206】。

16339 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  
16340 する【NGKI3207】。

16342 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  
16343 理ルーチン属性には、TA\_NULLを指定しなければならない【NGKI3209】。

16345 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】。

```
16347 void termination_routine(intptr_t exinf)
16348 {
16349     終了処理ルーチン本体
16350 }
```

16351  
 16352 exinfには、終了処理ルーチンの拡張情報が渡される【NGKI3211】。  
 16353  
 16354 【μITRON4.0仕様との関係】  
 16355  
 16356 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、  
 16357 μITRON4.0仕様に規定されていない機能である。  
 16358 -----  
 16359 LMT\_DOM 保護ドメインに対する制限の設定 [SP] 【NGKI3441】  
 16360  
 16361 【静的API】  
 16362 LMT\_DOM({ PRI mintpri })  
 16363  
 16364 【パラメータ】  
 16365 \*保護ドメインに対する制限の設定情報  
 16366 PRI mintpri 指定できる最高のタスク優先度  
 16367  
 16368 【エラーコード】  
 16369 E\_RSATR 予約属性  
 16370 ・ユーザドメインの囲みの中に記述されていない【NGKI3442】  
 16371 ・クラスの囲みの中に記述されている [M] 【NGKI3443】  
 16372 E\_OBJ オブジェクト状態エラー  
 16373 ・保護ドメインに対する制限が設定済み【NGKI3444】  
 16374 E\_PAR パラメータエラー  
 16375 ・mintpriが有効範囲外【NGKI3445】  
 16376  
 16377 【機能】  
 16378  
 16379 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  
 16380 ドメインに対する制限を設定する【NGKI3446】。  
 16381  
 16382 mintpriは整数定数式パラメータである【NGKI3447】。  
 16383  
 16384 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  
 16385 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】。  
 16386  
 16387 【μITRON4.0/PX仕様との関係】  
 16388  
 16389 μITRON4.0/PX仕様に定義されていない静的APIである。  
 16390 -----  
 16391 DEF\_ICS 非タスクコンテキスト用スタック領域の設定 [S] 【NGKI3212】  
 16392  
 16393 【静的API】  
 16394 DEF\_ICS({ SIZE istksz, STK\_T \*istk })  
 16395  
 16396 【パラメータ】  
 16397 \*非タスクコンテキスト用スタック領域の設定情報  
 16398 SIZE istksz 非タスクコンテキスト用スタック領域のサイズ  
 16399 (バイト数)  
 16400 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



16401  
16402     **【エラーコード】**  
16403         E\_RSATR     予約属性  
16404             ・カーネルドメインの囲みの中に記述されていない [P]     【NGKI3213】  
16405             ・クラスの囲みの中に記述されていない [M]     【NGKI3214】  
16406         E\_PAR     パラメータエラー  
16407             ・条件については機能の項を参照  
16408         E\_NOMEM     メモリ不足  
16409             ・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】  
16410         E\_OBJ     オブジェクト状態エラー  
16411             ・非タスクコンテキスト用スタック領域が設定済み 【NGKI3216】  
16412             ・その他の条件については機能の項を参照  
16413  
16414     **【機能】**  
16415  
16416     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  
16417     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  
16418     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  
16419     時には、E\_PARエラーとなる 【NGKI3254】。  
16420  
16421     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  
16422     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  
16423     ない 【NGKI3218】。  
16424  
16425     istkをNULLとした場合、istkszで指定したサイズのスタック領域が、コンフィ  
16426     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  
16427     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  
16428     大きい方に丸めたサイズで確保される 【NGKI3220】。  
16429  
16430     istkにNULL以外を指定した場合、istkとistkszで指定したスタック領域は、ア  
16431     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  
16432     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  
16433     を参照すること。その方法に従わず、istkやistkszにターゲット定義の制約に  
16434     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  
16435     【NGKI3222】。  
16436  
16437     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  
16438     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、  
16439     E\_OBJエラーとなる 【NGKI3223】。  
16440  
16441     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  
16442     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  
16443     保される 【NGKI3224】。  
16444  
16445     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  
16446     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  
16447     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  
16448     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  
16449     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、  
16450     E\_OBJエラーとなる 【NGKI3227】。

16451  
16452 **【TOPPERS/SSPカーネルにおける規定】**

16453  
16454 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  
16455 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

16456 **【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  
16457 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  
16458 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  
16459 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、  
16460 E\_PARエラーとなる **【SSPS0150】**.

16461  
16462 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.

16463  
16464 **【 $\mu$  ITRON4.0仕様との関係】**

16465  
16466  $\mu$  ITRON4.0仕様に定義されていない静的APIである.

16467 -----  
16468 DEF\_STK 共有スタック領域の設定 [S] **【NGKI3228】**

16469  
16470 **【静的API】**

16471 DEF\_STK({ SIZE stksz, STK\_T \*stk })

16472  
16473 **【パラメータ】**

16474 \* 共有スタック領域の設定情報

16475	SIZE	stksz	共有スタック領域のサイズ (バイト数)
16476	STK_T	stk	共有スタック領域の先頭番地

16477  
16478 **【エラーコード】**

16479	E_PAR	パラメータエラー
16480		・条件については機能の項を参照
16481	E_NOMEM	メモリ不足
16482		・共有スタック領域が確保できない <b>【NGKI3229】</b>
16483	E_OBJ	オブジェクト状態エラー
16484		・共有スタック領域が設定済み

16485  
16486 **【サポートするカーネル】**

16487  
16488 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  
16489 ネルは、DEF\_STKをサポートしない **【NGKI3230】**.

16490  
16491 **【機能】**

16492  
16493 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  
16494 領域を設定する **【NGKI3231】**. stkszに0以下の値を指定した時や、ターゲット  
16495 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】**.

16496  
16497 stkszは整数定数式パラメータ、stkは一般定数式パラメータである. コンフィ  
16498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  
16499ない **【NGKI3232】**.

stkをNULLとした場合、stkszで指定したサイズのスタック領域が、コンフィギュレータにより確保される【NGKI3233】。stkszにターゲット定義の制約に合致しないサイズを指定した時には、ターゲット定義の制約に合致するように大きい方に丸めたサイズで確保される【NGKI3234】。

stkにNULL以外を指定した場合、stkとstkszで指定したスタック領域は、アプリケーションで確保しておく必要がある【NGKI3235】。スタック領域をアプリケーションで確保する方法については、「2.15.3 カーネル共通マクロ」の節を参照すること。その方法に従わず、stkやstkszにターゲット定義の制約に合致しない先頭番地やサイズを指定した時には、E\_PARエラーとなる【NGKI3236】。

コンフィギュレータは、各タスクのスタック領域のサイズと、非タスクコンテキスト用のスタック領域のサイズから、共有スタック領域に必要なサイズを計算する【NGKI3237】。DEF\_STKにより共有スタック領域を設定しない場合、必要なサイズの共有スタック領域が、コンフィギュレータにより確保される【NGKI3238】。

stkszに指定したスタック領域のサイズが、共有スタック領域に必要なサイズよりも小さい場合、コンフィギュレータは警告メッセージを出力する【NGKI3239】。

【 $\mu$  ITRON4.0仕様との関係】

$\mu$  ITRON4.0仕様に定義されていない静的APIである。

---

ATT\_INI      初期化ルーチンの追加 [S] 【NGKI3240】

【静的API】

```
ATT_INI({ ATR iniatr, intptr_t exinf, INIRTN inirtn })
```

【パラメータ】

＊初期化ルーチンの追加情報

ATR	iniatr	初期化ルーチン属性
intptr_t	exinf	初期化ルーチンの拡張情報
INIRTN	inirtn	初期化ルーチンの先頭番地

【エラーコード】

E_RSATR	予約属性	・ iniatrが無効【NGKI3241】
		・ カーネルドメインの囲みの中に記述されていない [P] 【NGKI3242】
E_PAR	パラメータエラー	・ inirtnがプログラムの先頭番地として正しくない【NGKI3243】

【機能】

各パラメータで指定した初期化ルーチン追加情報に従って、初期化ルーチンを追加する【NGKI3244】。

iniatrは整数定数式パラメータ、exinfとinirtnは一般定数式パラメータである【NGKI3245】。

16551 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  
 16552 である【NGKI3246】.

16553

16554 **【補足説明】**

16555

16556 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  
 16557 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  
 16558 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.

16559 -----

16560 ATT\_TER      終了処理ルーチンの追加 [S]   【NGKI3247】

16561

16562 **【静的API】**

16563     ATT\_TER({ ATR teratr, intptr\_t exinf, TERRTN terrtn })

16564

16565 **【パラメータ】**

16566     \* 終了処理ルーチンの追加情報

16567         ATR           teratr       終了処理ルーチン属性

16568         intptr\_t      exinf       終了処理ルーチンの拡張情報

16569         TERRTN       terrtn      終了処理ルーチンの先頭番地

16570

16571 **【エラーコード】**

16572     E\_RSATR        予約属性

16573                    • teratrが無効【NGKI3248】

16574                    • カーネルドメインの囲みの中に記述されていない [P]   【NGKI3249】

16575     E\_PAR          パラメータエラー

16576                    • terrtnがプログラムの先頭番地として正しくない【NGKI3250】

16577

16578 **【機能】**

16579

16580 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  
 16581 ンを追加する【NGKI3251】.

16582

16583 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  
 16584 【NGKI3252】.

16585

16586 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  
 16587 である【NGKI3253】.

16588

16589 **【補足説明】**

16590

16591 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  
 16592 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  
 16593 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.

16594

16595 **【μ ITRON4.0仕様との関係】**

16596

16597 μ ITRON4.0仕様に定義されていない静的APIである.

16598 -----

16599 ref\_cfg        コンフィギュレーション情報の参照 [T]

16600

16601       【C言語API】  
16602           ER ercd = ref\_cfg(T\_RCFG \*pk\_rcfg)

16603  
16604       ☆未完成

16605  
16606       【TOPPERS/ASPカーネルにおける規定】  
16607  
16608       ASPカーネルでは、ref\_cfgをサポートしない。

16609  
16610       【TOPPERS/FMPカーネルにおける規定】  
16611  
16612       FMPカーネルでは、ref\_cfgをサポートしない。

16613  
16614       【TOPPERS/HRP2カーネルにおける規定】  
16615  
16616       HRP2カーネルでは、ref\_cfgをサポートしない。

16617  
16618       【TOPPERS/SSPカーネルにおける規定】  
16619  
16620       SSPカーネルでは、ref\_cfgをサポートしない。

16621       -----  
16622       ref\_ver       バージョン情報の参照 [T]

16623  
16624       【C言語API】  
16625           ER ercd = ref\_ver(T\_RVER \*pk\_rver)

16626  
16627       ☆未完成

16628  
16629       【TOPPERS/ASPカーネルにおける規定】  
16630  
16631       ASPカーネルでは、ref\_verをサポートしない。

16632  
16633       【TOPPERS/FMPカーネルにおける規定】  
16634  
16635       FMPカーネルでは、ref\_verをサポートしない。

16636  
16637       【TOPPERS/HRP2カーネルにおける規定】  
16638  
16639       HRP2カーネルでは、ref\_verをサポートしない。

16640  
16641       【TOPPERS/SSPカーネルにおける規定】  
16642  
16643       SSPカーネルでは、ref\_verをサポートしない。

16644       -----  
16645

16646  
16647       第5章   リファレンス

16648  
16649       5.1   サービスコール一覧

16650

## 16651 (1) タスク管理機能

16652

16653 ER\_ID tskid = acre\_tsk(const T\_CTSK \*pk\_ctsk) [TD]

16654 ER ercd = sac\_tsk(ID tskid, const ACVCT \*p\_acvct) [TPD]

16655 ER ercd = del\_tsk(ID tskid) [TD]

16656 ER ercd = act\_tsk(ID tskid) [T]

16657 ER ercd = iact\_tsk(ID tskid) [I]

16658 ER ercd = mact\_tsk(ID tskid, ID prcid) [TM]

16659 ER ercd = imact\_tsk(ID tskid, ID prcid) [IM]

16660 ER\_UINT actent = can\_act(ID tskid) [T]

16661 ER ercd = mig\_tsk(ID tskid, ID prcid) [TM]

16662 ER ercd = ext\_tsk() [T]

16663 ER ercd = ter\_tsk(ID tskid) [T]

16664 ER ercd = chg\_pri(ID tskid, PRI tskpri) [T]

16665 ER ercd = get\_pri(ID tskid, PRI \*p\_tskpri) [T]

16666 ER ercd = get\_inf(intptr\_t \*p\_exinf) [T]

16667 ER ercd = ref\_tsk(ID tskid, T\_RTsk \*pk\_rtsk) [T]

16668

## 16669 (2) タスク付属同期機能

16670

16671 ER ercd = slp\_tsk() [T]

16672 ER ercd = tslp\_tsk(TMO tmout) [T]

16673 ER ercd = wup\_tsk(ID tskid) [T]

16674 ER ercd = iwup\_tsk(ID tskid) [I]

16675 ER\_UINT wupcnt = can\_wup(ID tskid) [T]

16676 ER ercd = rel\_wai(ID tskid) [T]

16677 ER ercd = irel\_wai(ID tskid) [I]

16678 ER ercd = sus\_tsk(ID tskid) [T]

16679 ER ercd = rsm\_tsk(ID tskid) [T]

16680 ER ercd = dis\_wai(ID tskid) [TP]

16681 ER ercd = idis\_wai(ID tskid) [IP]

16682 ER ercd = ena\_wai(ID tskid) [TP]

16683 ER ercd = iena\_wai(ID tskid) [IP]

16684 ER ercd = dly\_tsk(RELTIM dlytim) [T]

16685

## 16686 (3) タスク例外処理機能

16687

16688 ER ercd = def\_tex(ID tskid, const T\_DTEX \*pk\_dtex) [TD]

16689 ER ercd = ras\_tex(ID tskid, TEXPTN rasptn) [T]

16690 ER ercd = iras\_tex(ID tskid, TEXPTN rasptn) [I]

16691 ER ercd = dis\_tex() [T]

16692 ER ercd = ena\_tex() [T]

16693 bool\_t state = sns\_tex() [TI]

16694 ER ercd = ref\_tex(ID tskid, T\_RTEx \*pk\_rtex) [T]

16695

## 16696 (4) 同期・通信機能

16697

16698 セマフォ

16699

16700 ER\_ID semid = acre\_sem(const T\_CSEM \*pk\_csem) [TD]

16701	ER ercd = sac_sem(ID semid, const ACVCT *p_acvct)	[TPD]
16702	ER ercd = del_sem(ID semid)	[TD]
16703	ER ercd = sig_sem(ID semid)	[T]
16704	ER ercd = isig_sem(ID semid)	[I]
16705	ER ercd = wai_sem(ID semid)	[T]
16706	ER ercd = pol_sem(ID semid)	[T]
16707	ER ercd = twai_sem(ID semid, TMO tmout)	[T]
16708	ER ercd = ini_sem(ID semid)	[T]
16709	ER ercd = ref_sem(ID semid, T_RSEM *pk_rsem)	[T]
16710		
16711	イベントフラグ	
16712		
16713	ER_ID flgid = acre_flg(const T_CFLG *pk_cflg)	[TD]
16714	ER ercd = sac_flg(ID flgid, const ACVCT *p_acvct)	[TPD]
16715	ER ercd = del_flg(ID flgid)	[TD]
16716	ER ercd = set_flg(ID flgid, FLGPTN setptn)	[T]
16717	ER ercd = iset_flg(ID flgid, FLGPTN setptn)	[I]
16718	ER ercd = clr_flg(ID flgid, FLGPTN clrptn)	[T]
16719	ER ercd = wai_flg(ID flgid, FLGPTN waiptn,	[T]
16720	MODE wfmode, FLGPTN *p_flgptn)	
16721	ER ercd = pol_flg(ID flgid, FLGPTN waiptn,	[T]
16722	MODE wfmode, FLGPTN *p_flgptn)	
16723	ER ercd = twai_flg(ID flgid, FLGPTN waiptn,	[T]
16724	MODE wfmode, FLGPTN *p_flgptn, TMO tmout)	
16725	ER ercd = ini_flg(ID flgid)	[T]
16726	ER ercd = ref_flg(ID flgid, T_RFLG *pk_rflg)	[T]
16727		
16728	データキュー	
16729		
16730	ER_ID dtqid = acre_dtq(const T_CDTQ *pk_cdtq)	[TD]
16731	ER ercd = sac_dtq(ID dtqid, const ACVCT *p_acvct)	[TPD]
16732	ER ercd = del_dtq(ID dtqid)	[TD]
16733	ER ercd = snd_dtq(ID dtqid, intptr_t data)	[T]
16734	ER ercd = psnd_dtq(ID dtqid, intptr_t data)	[T]
16735	ER ercd = ipsnd_dtq(ID dtqid, intptr_t data)	[I]
16736	ER ercd = tsnd_dtq(ID dtqid, intptr_t data, TMO tmout)	[T]
16737	ER ercd = fsnd_dtq(ID dtqid, intptr_t data)	[T]
16738	ER ercd = ifsnd_dtq(ID dtqid, intptr_t data)	[I]
16739	ER ercd = rcv_dtq(ID dtqid, intptr_t *p_data)	[T]
16740	ER ercd = prcv_dtq(ID dtqid, intptr_t *p_data)	[T]
16741	ER ercd = trecv_dtq(ID dtqid, intptr_t *p_data, TMO tmout)	[T]
16742	ER ercd = ini_dtq(ID dtqid)	[T]
16743	ER ercd = ref_dtq(ID dtqid, T_RDTQ *pk_rdtq)	[T]
16744		
16745	優先度データキュー	
16746		
16747	ER_ID pdqid = acre_pdq(const T_CPDQ *pk_cpdq)	[TD]
16748	ER ercd = sac_pdq(ID pdqid, const ACVCT *p_acvct)	[TPD]
16749	ER ercd = del_pdq(ID pdqid)	[TD]
16750	ER ercd = snd_pdq(ID pdqid, intptr_t data, PRI datapri)	[T]

```

16751      ER ercd = psnd_pdq(ID pdqid, intptr_t data, PRI datapri)      [T]
16752      ER ercd = ipsnd_pdq(ID pdqid, intptr_t data, PRI datapri)     [I]
16753      ER ercd = tsnd_pdq(ID pdqid, intptr_t data,                    [T]
16754                          PRI datapri, TMO tmout)
16755      ER ercd = rcv_pdq(ID pdqid, intptr_t *p_data, PRI *p_datapri) [T]
16756      ER ercd = prcv_pdq(ID pdqid, intptr_t *p_data, PRI *p_datapri) [T]
16757      ER ercd = trcv_pdq(ID pdqid, intptr_t *p_data,                [T]
16758                          PRI *p_datapri, TMO tmout)
16759      ER ercd = ini_pdq(ID pdqid)                                    [T]
16760      ER ercd = ref_pdq(ID pdqid, T_RPDQ *pk_rpdq)                  [T]
16761
16762      メールボックス
16763
16764      ER_ID mbxid = acre_mbx(const T_CMBX *pk_cmbx)                  [TDp]
16765      ER ercd = del_mbx(ID mbxid)                                    [TDp]
16766      ER ercd = snd_mbx(ID mbxid, T_MSG *pk_msg)                    [Tp]
16767      ER ercd = rcv_mbx(ID mbxid, T_MSG **ppk_msg)                  [Tp]
16768      ER ercd = prcv_mbx(ID mbxid, T_MSG **ppk_msg)                  [Tp]
16769      ER ercd = trcv_mbx(ID mbxid, T_MSG **ppk_msg, TMO tmout)      [Tp]
16770      ER ercd = ini_mbx(ID mbxid)                                    [Tp]
16771      ER ercd = ref_mbx(ID mbxid, T_RMBX *pk_rmbx)                  [Tp]
16772
16773      ミューテックス
16774
16775      ER_ID mtxid = acre_mtx(const T_CMTX *pk_cmtx)                  [TD]
16776      ER ercd = sac_mtx(ID mtxid, const ACVCT *p_acvct)              [TPD]
16777      ER ercd = del_mtx(ID mtxid)                                    [TD]
16778      ER ercd = loc_mtx(ID mtxid)                                    [T]
16779      ER ercd = ploc_mtx(ID mtxid)                                    [T]
16780      ER ercd = tloc_mtx(ID mtxid, TMO tmout)                        [T]
16781      ER ercd = unl_mtx(ID mtxid)                                    [T]
16782      ER ercd = ini_mtx(ID mtxid)                                    [T]
16783      ER ercd = ref_mtx(ID mtxid, T_RMTX *pk_rmtx)                  [T]
16784
16785      メッセージバッファ
16786
16787      ER_ID mbfid = acre_mbf(const T_CMBF *pk_cmbf)                  [TD]
16788      ER ercd = sac_mbf(ID mbfid, const ACVCT *p_acvct)              [TPD]
16789      ER ercd = del_mbf(ID mbfid)                                    [TD]
16790      ER ercd = snd_mbf(ID mbfid, const void *msg, uint_t msgsz)      [T]
16791      ER ercd = psnd_mbf(ID mbfid, const void *msg, uint_t msgsz)    [T]
16792      ER ercd = tsnd_mbf(ID mbfid, const void *msg,                  [T]
16793                          uint_t msgsz, TMO tmout)
16794      ER_UINT msgsz = rcv_mbf(ID mbfid, void *msg)                    [T]
16795      ER_UINT msgsz = prcv_mbf(ID mbfid, void *msg)                  [T]
16796      ER_UINT msgsz = trcv_mbf(ID mbfid, void *msg, TMO tmout)        [T]
16797      ER ercd = ini_mbf(ID mbfid)                                    [T]
16798      ER ercd = ref_mbf(ID mbfid, T_RMBF *pk_rmbf)                  [T]
16799
16800      スピンロック

```



```

16801
16802     ER_ID spnid = acre_spn(const T_CSPN *pk_cspn)                [TMD]
16803     ER ercd = sac_spn(ID spnid, const ACVCT *p_acvct)           [TPMD]
16804     ER ercd = del_spn(ID spnid)                                  [TMD]
16805     ER ercd = loc_spn(ID spnid)                                  [TM]
16806     ER ercd = iloc_spn(ID spnid)                                [IM]
16807     ER ercd = try_spn(ID spnid)                                  [TM]
16808     ER ercd = itry_spn(ID spnid)                                 [IM]
16809     ER ercd = unl_spn(ID spnid)                                  [TM]
16810     ER ercd = iunl_spn(ID spnid)                                [IM]
16811     ER ercd = ref_spn(ID spnid, T_RSPN *pk_rspn)                [TM]
16812
16813     (5) メモリプール管理機能
16814
16815     固定長メモリプール
16816
16817     ER_ID mpfid = acre_mpf(const T_CMPF *pk_cmpf)                [TD]
16818     ER ercd = sac_mpf(ID mpfid, const ACVCT *p_acvct)           [TPD]
16819     ER ercd = del_mpf(ID mpfid)                                  [TD]
16820     ER ercd = get_mpf(ID mpfid, void **p_blk)                    [T]
16821     ER ercd = pget_mpf(ID mpfid, void **p_blk)                  [T]
16822     ER ercd = tget_mpf(ID mpfid, void **p_blk, TMO tmout)       [T]
16823     ER ercd = rel_mpf(ID mpfid, void *blk)                       [T]
16824     ER ercd = ini_mpf(ID mpfid)                                  [T]
16825     ER ercd = ref_mpf(ID mpfid, T_RMPF *pk_rmpf)                [T]
16826
16827     (6) 時間管理機能
16828
16829     システム時刻管理
16830
16831     ER ercd = get_tim(SYSTIM *p_system)                          [T]
16832     ER ercd = get_utm(SYSUTM *p_sysutm)                          [TI]
16833
16834     周期ハンドラ
16835
16836     ER_ID cycid = acre_cyc(const T_CCYC *pk_ccyc)                [TD]
16837     ER ercd = sac_cyc(ID cycid, const ACVCT *p_acvct)           [TPD]
16838     ER ercd = del_cyc(ID cycid)                                  [TD]
16839     ER ercd = sta_cyc(ID cycid)                                  [T]
16840     ER ercd = msta_cyc(ID cycid, ID prcid)                       [TM]
16841     ER ercd = stp_cyc(ID cycid)                                  [T]
16842     ER ercd = ref_cyc(ID cycid, T_RCYC *pk_rcyc)                [T]
16843
16844     アラームハンドラ
16845
16846     ER_ID almid = acre_alm(const T_CALM *pk_calm)                [TD]
16847     ER ercd = sac_alm(ID almid, const ACVCT *p_acvct)           [TPD]
16848     ER ercd = del_alm(ID almid)                                  [TD]
16849     ER ercd = sta_alm(ID almid, RELTIM almtim)                   [T]
16850     ER ercd = ista_alm(ID almid, RELTIM almtim)                  [I]

```

```

16851      ER ercd = msta_alm(ID almid, RELTIM almtim, ID prcid)      [TM]
16852      ER ercd = imsta_alm(ID almid, RELTIM almtim, ID prcid)    [IM]
16853      ER ercd = stp_alm(ID almid)                                [T]
16854      ER ercd = istp_alm(ID almid)                               [I]
16855      ER ercd = ref_alm(ID almid, T_RALM *pk_ralm)              [T]
16856
16857      オーバランハンドラ
16858
16859      ER ercd = def_ovr(const T_DOVR *pk_dovr)                  [TD]
16860      ER ercd = sta_ovr(ID tskid, OVRTIM ovrtime)              [T]
16861      ER ercd = ista_ovr(ID tskid, OVRTIM ovrtime)             [I]
16862      ER ercd = stp_ovr(ID tskid)                              [T]
16863      ER ercd = istp_ovr(ID tskid)                             [I]
16864      ER ercd = ref_ovr(ID tskid, T_ROVR *pk_rovr)             [T]
16865
16866      (7) システム状態管理機能
16867
16868      ER ercd = sac_sys(const ACVCT *p_acvct)                  [TPD]
16869      ER ercd = rot_rdq(PRI tskpri)                            [T]
16870      ER ercd = irot_rdq(PRI tskpri)                           [I]
16871      ER ercd = mrot_rdq(PRI tskpri, ID prcid)                 [TM]
16872      ER ercd = imrot_rdq(PRI tskpri, ID prcid)               [IM]
16873      ER ercd = get_tid(ID *p_tskid)                           [T]
16874      ER ercd = iget_tid(ID *p_tskid)                         [I]
16875      ER ercd = get_did(ID *p_domid)                           [TP]
16876      ER ercd = get_pid(ID *p_prcid)                           [TM]
16877      ER ercd = iget_pid(ID *p_prcid)                         [IM]
16878      ER ercd = loc_cpu()                                       [T]
16879      ER ercd = iloc_cpu()                                       [I]
16880      ER ercd = unl_cpu()                                       [T]
16881      ER ercd = iunl_cpu()                                       [I]
16882      ER ercd = dis_dsp()                                       [T]
16883      ER ercd = ena_dsp()                                       [T]
16884      bool_t state = sns_ctx()                                   [TI]
16885      bool_t state = sns_loc()                                   [TI]
16886      bool_t state = sns_dsp()                                   [TI]
16887      bool_t state = sns_dpn()                                   [TI]
16888      bool_t state = sns_ker()                                   [TI]
16889      ER ercd = ext_ker()                                       [TI]
16890      ER ercd = ref_sys(T_RSYS *pk_rsys)                        [T]
16891
16892      (8) メモリオブジェクト管理機能
16893
16894      ER ercd = att_mem(const T_AMEM *pk_amem)                  [TPD]
16895      ER ercd = att_pma(const T_AMEM *pk_apma)                  [TPD]
16896      ER ercd = sac_mem(const void *base, const ACVCT *p_acvct) [TPD]
16897      ER ercd = det_mem(const void *base)                       [TPD]
16898      ER ercd = prb_mem(const void *base, SIZE size,           [TP]
16899                      ID tskid, MODE pmmode)
16900      ER ercd = ref_mem(const void *base, T_RMEM *pk_rmem)      [TP]

```

```

16901
16902 (9) 割込み管理機能
16903
16904 ER ercd = cfg_int(INTNO intno, const T_CINT *pk_cint) [TD]
16905 ER_ID isrid = acre_isr(const T_CISR *pk_cisr) [TD]
16906 ER ercd = sac_isr(ID isrid, const ACVCT *p_acvct) [TPD]
16907 ER ercd = del_isr(ID isrid) [TD]
16908 ER ercd = ref_isr(ID isrid, T_RISR *pk_risr) [T]
16909 ER ercd = def_inh(INHNO inhno, const T_DINH *pk_dinh) [TD]
16910 ER ercd = dis_int(INTNO intno) [T]
16911 ER ercd = ena_int(INTNO intno) [T]
16912 ER ercd = ref_int(INTNO intno, T_RINT *pk_rint) [T]
16913 ER ercd = chg_ipm(PRI intpri) [T]
16914 ER ercd = get_ipm(PRI *p_intpri) [T]
16915
16916 (10) CPU例外管理機能
16917
16918 ER ercd = def_exc(EXCNO excno, const T_DEXC *pk_dexc) [TD]
16919 bool_t stat = xsns_dpn(void *p_excinf) [TI]
16920 bool_t stat = xsns_xpn(void *p_excinf) [TI]
16921
16922 (11) 拡張サービスコール管理機能
16923
16924 ER ercd = def_svc(FN fned, const T_DSVC *pk_dsvc) [TPD]
16925 ER_UINT ercd = cal_svc(FN fned, intptr_t par1, intptr_t par2, [TIP]
16926 intptr_t par3, intptr_t par4, intptr_t par5)
16927
16928 (12) システム構成管理機能
16929
16930 ER ercd = ref_cfg(T_RCFG *pk_rcfg) [T]
16931 ER ercd = ref_ver(T_RVER *pk_rver) [T]
16932
16933 5.2 静的API一覧
16934
16935 (1) タスク管理機能
16936
16937 *保護機能対応でないカーネルの場合
16938 CRE_TSK(ID tskid, { ATR tskatr, intptr_t exinf, TASK task, [S]
16939 PRI itskpri, SIZE stksz, STK_T *stk })
16940
16941 *保護機能対応カーネルの場合
16942 CRE_TSK(ID tskid, { ATR tskatr, intptr_t exinf, TASK task, [SP]
16943 PRI itskpri, SIZE stksz, STK_T *stk,
16944 SIZE sstksz, STK_T *sstk })
16945 ※ sstkszおよびsstkの記述は省略することができる。
16946
16947 AID_TSK(uint_t notsk) [SD]
16948 SAC_TSK(ID tskid, { ACPTN acptn1, ACPTN acptn2, [SP]
16949 ACPTN acptn3, ACPTN acptn4 })
16950 DEF_EPR(ID tskid, { PRI exepri }) [S]

```

16951			
16952	(2) タスク付属同期機能		
16953			
16954	なし		
16955			
16956	(3) タスク例外処理機能		
16957			
16958	DEF_TEX(ID tskid, { ATR texatr, TEXTN texrtn })	[S]	
16959			
16960	(4) 同期・通信機能		
16961			
16962	セマフォ		
16963			
16964	CRE_SEM(ID semid, { ATR sematr, uint_t isemcnt, uint_t maxsem })	[S]	
16965	AID_SEM(uint_t nose)	[SD]	
16966	SAC_SEM(ID semid, { ACPTN acptn1, ACPTN acptn2,	[SP]	
16967	ACPTN acptn3, ACPTN acptn4 })		
16968			
16969	イベントフラグ		
16970			
16971	CRE_FLG(ID flgid, { ATR flgatr, FLGPTN iflgptn })	[S]	
16972	AID_FLG(uint_t noflg)	[SD]	
16973	SAC_FLG(ID flgid, { ACPTN acptn1, ACPTN acptn2,	[SP]	
16974	ACPTN acptn3, ACPTN acptn4 })		
16975			
16976	データキュー		
16977			
16978	CRE_DTQ(ID dtqid, { ATR dtqatr, uint_t dtqcnt, void *dtqmb })	[S]	
16979	AID_DTQ(uint_t nodtq)	[SD]	
16980	SAC_DTQ(ID dtqid, { ACPTN acptn1, ACPTN acptn2,	[SP]	
16981	ACPTN acptn3, ACPTN acptn4 })		
16982			
16983	優先度データキュー		
16984			
16985	CRE_PDQ(ID pdqid, { ATR pdqatr, uint_t pdqcnt,	[S]	
16986	PRI maxdpri, void *pdqmb })		
16987	AID_PDQ(uint_t nopdq)	[SD]	
16988	SAC_PDQ(ID pdqid, { ACPTN acptn1, ACPTN acptn2,	[SP]	
16989	ACPTN acptn3, ACPTN acptn4 })		
16990			
16991	メールボックス		
16992			
16993	CRE_MBX(ID mbxid, { ATR mbxatr, PRI maxmpri, void *mprihd })	[Sp]	
16994	AID_MBX(uint_t nombx)	[SpD]	
16995			
16996	ミューテックス		
16997			
16998	CRE_MTX(ID mtxid, { ATR mtxatr, PRI ceilpri })	[S]	
16999	AID_MTX(uint_t nomtx)	[SD]	
17000	SAC_MTX(ID mtxid, { ACPTN acptn1, ACPTN acptn2,	[SP]	

```

17001                                     ACPTN acptn3, ACPTN acptn4 })
17002
17003     メッセージバッファ
17004
17005         CRE_MBF(ID mbfid, { ATR mbfatr, uint_t maxmsz,
17006                             SIZE mbfsz, void *mbfmb })           [S]
17007         AID_MBF(uint_t nombf)                                     [SD]
17008         SAC_MBF(ID mbfid, { ACPTN acptn1, ACPTN acptn2,
17009                             ACPTN acptn3, ACPTN acptn4 })       [SP]
17010
17011     スピンロック
17012
17013         CRE_SPN(ID spnid, { ATR spnatr })                         [SM]
17014         AID_SPN(uint_t nospn)                                     [SMD]
17015         SAC_SPN(ID spnid, { ACPTN acptn1, ACPTN acptn2,
17016                             ACPTN acptn3, ACPTN acptn4 })       [SPM]
17017
17018     (5) メモリプール管理機能
17019
17020     固定長メモリプール
17021
17022         CRE_MPF(ID mpfid, { ATR mpfatr, uint_t blkcnt, uint_t blksz,
17023                             MPF_T *mpf, void *mpfmb })           [S]
17024         AID_MPF(uint_t nompf)                                     [SD]
17025         SAC_MPF(ID mpfid, { ACPTN acptn1, ACPTN acptn2,
17026                             ACPTN acptn3, ACPTN acptn4 })       [SP]
17027
17028     (6) 時間管理機能
17029
17030     周期ハンドラ
17031
17032         CRE_CYC(ID cycid, { ATR cycatr, intptr_t exinf, CYCHDR cychdr,
17033                             RELTIM cycetim, RELTIM cycphs })     [S]
17034         AID_CYC(uint_t nocyc)                                     [SD]
17035         SAC_CYC(ID cycid, { ACPTN acptn1, ACPTN acptn2,
17036                             ACPTN acptn3, ACPTN acptn4 })       [SP]
17037
17038     アラームハンドラ
17039
17040         CRE_ALM(ID almid, { ATR almatr, intptr_t exinf, ALMHDR almhdr }) [S]
17041         AID_ALM(uint_t noalm)                                     [SD]
17042         SAC_ALM(ID almid, { ACPTN acptn1, ACPTN acptn2,
17043                             ACPTN acptn3, ACPTN acptn4 })       [SP]
17044
17045     オーバランハンドラ
17046
17047         DEF_OVR({ ATR ovratr, OVRHDR ovrrhdr })                 [S]
17048
17049     (7) システム状態管理機能
17050

```

```

17051      SAC_SYS({ ACPTN acptn1, ACPTN acptn2,           [SP]
17052                ACPTN acptn3, ACPTN acptn4 })
17053
17054  (8) メモリオブジェクト管理機能
17055
17056      ATT_REG("メモリリージョン名",           [SP]
17057              { ATR regatr, void *base, SIZE size })
17058      DEF_SRG("標準ROMリージョン名", "標準RAMリージョン名") [SP]
17059      ATT_SEC("セクション名", { ATR mematr, "メモリリージョン名" }) [SP]
17060      ATA_SEC("セクション名", { ATR mematr, "メモリリージョン名" }, [SP]
17061              { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
17062      LNK_SEC("セクション名", { "メモリリージョン名" }) [SP]
17063      ATT_MOD("オブジェクトモジュール名") [SP]
17064      ATA_MOD("オブジェクトモジュール名", [SP]
17065              { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
17066      ATT_MEM({ ATR mematr, void *base, SIZE size }) [SP]
17067      ATA_MEM({ ATR mematr, void *base, SIZE size }, [SP]
17068              { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
17069      ATT_PMA({ ATR mematr, void *base, SIZE size, void *paddr }) [SP]
17070      ATA_PMA({ ATR mematr, void *base, SIZE size, void *paddr }, [SP]
17071              { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
17072
17073  (9) 割込み管理機能
17074
17075      CFG_INT(INTNO intno, { ATR intatr, PRI intpri }) [S]
17076      CRE_ISR(ID isrid, { ATR isratr, intptr_t exinf, [S]
17077                          INTNO intno, ISR isr, PRI isrpri })
17078      ATT_ISR({ ATR isratr, intptr_t exinf, [S]
17079               INTNO intno, ISR isr, PRI isrpri })
17080      AID_ISR(uint_t noisr) [SD]
17081      SAC_ISR(ID isrid, { ACPTN acptn1, ACPTN acptn2, [SP]
17082                       ACPTN acptn3, ACPTN acptn4 })
17083      DEF_INH(INHNO inhno, { ATR inhatr, INTHDR inthdr }) [S]
17084
17085  (10) CPU例外管理機能
17086
17087      DEF_EXC(EXCNO excno, { ATR excatr, EXCHDR exchr }) [S]
17088
17089  (11) 拡張サービスコール管理機能
17090
17091      DEF_SVC(FN fncd, { ATR svcatr, EXTSVC svertn, SIZE stksz }) [SP]
17092
17093  (12) システム構成管理機能
17094
17095      LMT_DOM({ PRI mintpri }) [SP]
17096      DEF_ICS({ SIZE istksz, STK_T *istk }) [S]
17097      DEF_STK({ SIZE stksz, STK_T *stk }) [S]
17098      ATT_INI({ ATR iniatr, intptr_t exinf, INIRTN inirtn }) [S]
17099      ATT_TER({ ATR teratr, intptr_t exinf, TERRTN terrtn }) [S]
17100

```

## 17101 5.3 データ型

17102

## 17103 5.3.1 TOPPERS共通データ型

17104

17105 int8\_t 符号付き8ビット整数 (オプション, C99準拠)

17106 uint8\_t 符号無し8ビット整数 (オプション, C99準拠)

17107 int16\_t 符号付き16ビット整数 (C99準拠)

17108 uint16\_t 符号無し16ビット整数 (C99準拠)

17109 int32\_t 符号付き32ビット整数 (C99準拠)

17110 uint32\_t 符号無し32ビット整数 (C99準拠)

17111 int64\_t 符号付き64ビット整数 (オプション, C99準拠)

17112 uint64\_t 符号無し64ビット整数 (オプション, C99準拠)

17113 int128\_t 符号付き128ビット整数 (オプション, C99準拠)

17114 uint128\_t 符号無し128ビット整数 (オプション, C99準拠)

17115

17116 int\_least8\_t 8ビット以上の符号付き整数 (C99準拠)

17117 uint\_least8\_t int\_least8\_t型と同じサイズの符号無し整数 (C99準拠)

17118

17119 float32\_t IEEE754準拠の32ビット単精度浮動小数点数 (オプション)

17120 double64\_t IEEE754準拠の64ビット倍精度浮動小数点数 (オプション)

17121

17122 bool\_t 真偽値 (trueまたはfalse)

17123 int\_t 16ビット以上の符号付き整数

17124 uint\_t int\_t型と同じサイズの符号無し整数

17125 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

17126 ulong\_t long\_t型と同じサイズの符号無し整数

17127

17128 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)

17129 uintptr\_t intptr\_t型と同じサイズの符号無し整数 (C99準拠)

17130

17131 FN 機能コード (符号付き整数, int\_tに定義)

17132 ER エラーコード (符号付き整数, int\_tに定義)

17133 ID オブジェクトのID番号 (符号付き整数, int\_tに定義)

17134 ATR オブジェクト属性 (符号無し整数, uint\_tに定義)

17135 STAT オブジェクトの状態 (符号無し整数, uint\_tに定義)

17136 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)

17137 PRI 優先度 (符号付き整数, int\_tに定義)

17138 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  
17139 サイズの符号無し整数型に定義)

17140

17141 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)

17142 RELTIM 相対時間 (符号無し整数, 単位はミリ秒, uint\_tに定義)

17143 SYSTIM システム時刻 (符号無し整数, 単位はミリ秒, ulong\_tに定義)

17144 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,  
17145 ulong\_tに定義)

17146

17147 FP プログラムの起動番地 (型の定まらない関数ポインタ)

17148

17149 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)

17150 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)

```

17151
17152     ER_UINT    負のID番号は格納できない)
17153               エラーコードまたは符号無し整数 (符号付き整数, int_tに
17154               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
17155               より1ビット短い)
17156
17157     MB_T       オブジェクト管理領域を確保するためのデータ型
17158
17159     ACPTN      アクセス許可パターン (符号無し32ビット整数, uint32_tに
17160               定義)
17161
17162     typedef struct acvct {          /* アクセス許可ベクタ */
17163         ACPTN    acptn1;           /* 通常操作1のアクセス許可パターン */
17164         ACPTN    acptn2;           /* 通常操作2のアクセス許可パターン */
17165         ACPTN    acptn3;           /* 管理操作のアクセス許可パターン */
17166         ACPTN    acptn4;           /* 参照操作のアクセス許可パターン */
17167     } ACVCT;
17168
17169 5.3.2 カーネルの使用するデータ型
17170
17171     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)
17172     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)
17173     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)
17174     INTNO      割込み番号 (符号無し整数, uint_tに定義)
17175     INHNO      割込みハンドラ番号 (符号無し整数, uint_tに定義)
17176     EXCNO      CPU例外ハンドラ番号 (符号無し整数, uint_tに定義)
17177
17178     TASK       タスクのメインルーチン (関数ポインタ)
17179     TEXRTN     タスク例外処理ルーチン (関数ポインタ)
17180     CYCHDR     周期ハンドラ (関数ポインタ)
17181     ALMHDR     アラームハンドラ (関数ポインタ)
17182     OVRHDR     オーバランハンドラ (関数ポインタ)
17183     ISR        割込みサービ斯拉ーチン (関数ポインタ)
17184     INTHDR     割込みハンドラ (関数ポインタ)
17185     EXCHDR     CPU例外ハンドラ (関数ポインタ)
17186     EXT SVC     拡張サービスコール (関数ポインタ)
17187     INIRTN     初期化ルーチン (関数ポインタ)
17188     TERRTN     終了処理ルーチン (関数ポインタ)
17189
17190     STK_T      スタック領域を確保するためのデータ型
17191     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
17192
17193 メールボックスのメッセージヘッダ【NGKI4001】
17194
17195     typedef struct t_msg {
17196         struct t_msg    *pk_next;
17197     } T_MSG;
17198
17199 メールボックスの優先度付きメッセージヘッダ【NGKI4002】
17200
17201     typedef struct t_msg_pri {

```



```

17201         T_MSG      msgque;      /* メールボックスのメッセージヘッダ */
17202         PRI         msgpri;      /* メッセージ優先度 */
17203     } T_MSG_PRI;

```

### 5.3.3 カーネルの使用するパケット形式

#### (1) タスク管理機能

##### タスクの生成情報のパケット形式【NGKI4003】

```

17211     typedef struct t_ctsk {
17212         ATR          tskatr;      /* タスク属性 */
17213         intptr_t     exinf;      /* タスクの拡張情報 */
17214         TASK         task;      /* タスクのメインルーチンの先頭番地 */
17215         PRI          itskpri;     /* タスクの起動時優先度 */
17216         SIZE         stksz;      /* タスクのスタック領域のサイズ */
17217         STK_T *      stk;        /* タスクのスタック領域の先頭番地 */
17218         /* 以下は、保護機能対応カーネルの場合 */
17219         SIZE         sstksz;     /* タスクのシステムスタック領域のサイズ */
17220         STK_T *      sstk;      /* タスクのシステムスタック領域の先頭番地 */
17221     } T_CTSK;

```

##### タスクの現在状態のパケット形式【NGKI4004】

```

17225     typedef struct t_rtsk {
17226         STAT         tskstat;     /* タスク状態 */
17227         PRI          tskpri;     /* タスクの現在優先度 */
17228         PRI          tsbpri;     /* タスクのベース優先度 */
17229         STAT         tsawait;     /* 待ち要因 */
17230         ID           wobjid;     /* 待ち対象のオブジェクトのID */
17231         TMO          lefttmo;    /* タイムアウトするまでの時間 */
17232         uint_t       actcnt;     /* 起動要求キューイング数 */
17233         uint_t       wupcnt;     /* 起床要求キューイング数 */
17234         /* 以下は、保護機能対応カーネルの場合 */
17235         bool_t       texmsk;     /* タスク例外マスク状態か否か */
17236         bool_t       waifbd;     /* 待ち禁止状態か否か */
17237         uint_t       svclevel;   /* 拡張サービスコールのネストレベル */
17238         /* 以下は、マルチプロセッサ対応カーネルの場合 */
17239         ID           prcid;      /* 割付けプロセッサのID */
17240         ID           actprc      /* 次の起動時の割付けプロセッサのID */
17241     } T_RTSK;

```

#### (2) タスク付属同期機能

なし

#### (3) タスク例外処理機能

##### タスク例外処理ルーチンの定義情報のパケット形式【NGKI4005】

17250

```

17251     typedef struct t_dtex {
17252         ATR          texatr;      /* タスク例外処理ルーチン属性 */
17253         TEXRTN       texrtn;      /* タスク例外処理ルーチンの先頭番地 */
17254     } T_DTEX;

```

17255

17256 タスク例外処理の現在状態のパケット形式【NGKI4006】

17257

```

17258     typedef struct t_rtex {
17259         STAT         texstat;      /* タスク例外処理の状態 */
17260         TEXPTN       pndptn;      /* 保留例外要因 */
17261     } T_RTEX;

```

17262

17263 (4) 同期・通信機能

17264

17265 セマフォの生成情報のパケット形式【NGKI4007】

17266

```

17267     typedef struct t_csem {
17268         ATR          sematr;      /* セマフォ属性 */
17269         uint_t        isemcnt;     /* セマフォの初期資源数 */
17270         uint_t        maxsem;     /* セマフォの最大資源数 */
17271     } T_CSEM;

```

17272

17273 セマフォの現在状態のパケット形式【NGKI4008】

17274

```

17275     typedef struct t_rsem {
17276         ID           wtskid;      /* セマフォの待ち行列の先頭のタスクのID番号 */
17277         uint_t        semcnt;     /* セマフォの資源数 */
17278     } T_RSEM;

```

17279

17280 イベントフラグの生成情報のパケット形式【NGKI4009】

17281

```

17282     typedef struct t_cflg {
17283         ATR          flgatr;      /* イベントフラグ属性 */
17284         FLGPTN       iflgptn;     /* イベントフラグの初期ビットパターン */
17285     } T_CFLG;

```

17286

17287 イベントフラグの現在状態のパケット形式【NGKI4010】

17288

```

17289     typedef struct t_rflg {
17290         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 */
17291         FLGPTN       flgptn;     /* イベントフラグのビットパターン */
17292     } T_RFLG;

```

17293

17294 データキューの生成情報のパケット形式【NGKI4011】

17295

```

17296     typedef struct t_cdtq {
17297         ATR          dtqatr;      /* データキュー属性 */
17298         uint_t        dtqcnt;     /* データキュー管理領域に格納できるデータ数 */
17299         void *        dtqmb;      /* データキュー管理領域の先頭番地 */
17300     }

```

```
17301     } T_CDTQ;
```

```
17302
```

```
17303 データキューの現在状態のパケット形式【NGKI4012】
```

```
17304
```

```
17305     typedef struct t_rdtq {
17306         ID          stskid;    /* データキューの送信待ち行列の先頭のタ
17307                                スクのID番号 */
17308         ID          rtskid;    /* データキューの受信待ち行列の先頭のタ
17309                                スクのID番号 */
17310         uint_t      sdtqcnt;   /* データキュー管理領域に格納されている
17311                                データの数 */
17312     } T_RDTQ;
```

```
17312
```

```
17313
```

```
17314 優先度データキューの生成情報のパケット形式【NGKI4013】
```

```
17315
```

```
17316     typedef struct t_cpdq {
17317         ATR          pdqatr;    /* 優先度データキュー属性 */
17318         uint_t       pdqcnt;    /* 優先度データキュー管理領域に格納でき
17319                                るデータ数 */
17320         PRI          maxdpri;   /* 優先度データキューに送信できるデータ
17321                                優先度の最大値 */
17322         void *       pdqmb;     /* 優先度データキュー管理領域の先頭番地 */
17323     } T_CPDQ;
```

```
17324
```

```
17325 優先度データキューの現在状態のパケット形式【NGKI4014】
```

```
17326
```

```
17327     typedef struct t_rpdq {
17328         ID          stskid;    /* 優先度データキューの送信待ち行列の先
17329                                頭のタスクのID番号 */
17330         ID          rtskid;    /* 優先度データキューの受信待ち行列の先
17331                                頭のタスクのID番号 */
17332         uint_t      spdqcnt;   /* 優先度データキュー管理領域に格納され
17333                                ているデータの数 */
17334     } T_RPDQ;
```

```
17335
```

```
17336 メールボックスの生成情報のパケット形式【NGKI4015】
```

```
17337
```

```
17338     typedef struct t_cmbx {
17339         ATR          mbxatr;    /* メールボックス属性 */
17340         PRI          maxmpri;   /* 優先度メールボックスに送信できるメッ
17341                                セージ優先度の最大値 */
17342         void *       mprihd;    /* 優先度別のメッセージキューヘッダ領域
17343                                の先頭番地 */
17344     } T_CMBX;
```

```
17345
```

```
17346 メールボックスの現在状態のパケット形式【NGKI4016】
```

```
17347
```

```
17348     typedef struct t_rmbx {
17349         ID          wtskid;    /* メールボックスの待ち行列の先頭のタスク
17350                                のID番号 */
17351     }
```

```

17351         T_MSG         *pk_msg;    /* メッセージキューの先頭につながれたメッ
17352                                セージの先頭番地 */
17353     } T_RMBX;
17354
17355     ミューテックスの生成情報のパケット形式【NGKI4017】
17356
17357     typedef struct t_cmtx {
17358         ATR             mtxatr;    /* ミューテックス属性 */
17359         PRI             ceilpri;   /* ミューテックスの上限優先度 */
17360     } T_CMTX;
17361
17362     ミューテックスの現在状態のパケット形式【NGKI4018】
17363
17364     typedef struct t_rmtx {
17365         ID              htskid;    /* ミューテックスをロックしているタス
17366                                クのID番号 */
17367         ID              wtskid;    /* ミューテックスの待ち行列の先頭のタ
17368                                スクのID番号 */
17369     } T_RMTX;
17370
17371     メッセージバッファの生成情報のパケット形式【NGKI4037】
17372
17373     typedef struct t_cmbf {
17374         ATR             mbfatr;    /* メッセージバッファ属性 */
17375         uint_t          maxmsz;    /* メッセージバッファの最大メッセージ
17376                                サイズ (バイト数) */
17377         SIZE            mbfsz;     /* メッセージバッファ管理領域のサイズ
17378                                (バイト数) */
17379         void *          mbfmb;     /* メッセージバッファ管理領域の先頭番地 */
17380     } T_CMBF;
17381
17382     メッセージバッファの現在状態のパケット形式【NGKI4038】
17383
17384     typedef struct t_rmbf {
17385         ID              stskid;    /* メッセージバッファの送信待ち行列の先頭の
17386                                タスクのID番号 */
17387         ID              rtskid;    /* メッセージバッファの受信待ち行列の先頭の
17388                                タスクのID番号 */
17389         uint_t          smbfcnt;   /* メッセージバッファ管理領域に格納されてい
17390                                るメッセージの数 */
17391         SIZE            fmbfsz;    /* メッセージバッファ管理領域中の空き領域の
17392                                サイズ */
17393     } T_RMBF;
17394
17395     スピンロックの生成情報のパケット形式【NGKI4019】
17396
17397     typedef struct t_cspn {
17398         ATR             spnatr;    /* スピンロック属性 */
17399     } T_CSPN;
17400

```

17401 スピンロックの現在状態のパケット形式【NGKI4020】

17402

```
17403     typedef struct t_rspn {  
17404         STAT          spnstat      /* スピンロックのロック状態 */  
17405     } T_RSPN;
```

17406

17407 (5) メモリプール管理機能

17408

17409 固定長メモリアプールの生成情報のパケット形式【NGKI4021】

17410

```
17411     typedef struct t_cmpf {  
17412         ATR          mpfatr;      /* 固定長メモリアプール属性 */  
17413         uint_t       blkcnt;      /* 獲得できる固定長メモリアブロックの数 */  
17414         uint_t       blksiz;      /* 固定長メモリアブロックのサイズ */  
17415         MPF_T *      mpf;         /* 固定長メモリアプール領域の先頭番地 */  
17416         void *       mpfmb;      /* 固定長メモリアプール管理領域の先頭番地 */  
17417     } T_CMPF;
```

17418

17419 固定長メモリアプールの現在状態のパケット形式【NGKI4022】

17420

```
17421     typedef struct t_rmpf {  
17422         ID          wtskid;      /* 固定長メモリアプールの待ち行列の先頭の  
17423                                タスクのID番号 */  
17424         uint_t      fblkcnt;     /* 固定長メモリアプール領域の空きメモリア領  
17425                                域に割り付けることができる固定長メモ  
17426                                リブロックの数 */  
17427     } T_RMPF;
```

17428

17429 (6) 時間管理機能

17430

17431 周期ハンドラの生成情報のパケット形式【NGKI4023】

17432

```
17433     typedef struct t_ccyc {  
17434         ATR          cycatr;      /* 周期ハンドラ属性 */  
17435         intptr_t     exinf;       /* 周期ハンドラの拡張情報 */  
17436         CYCHDR       cychdr;     /* 周期ハンドラ先頭番地 */  
17437         RELTIM       cyctim;     /* 周期ハンドラの起動周期 */  
17438         RELTIM       cycphs;     /* 周期ハンドラの起動位相 */  
17439     } T_CCYC;
```

17440

17441 周期ハンドラの現在状態のパケット形式【NGKI4024】

17442

```
17443     typedef struct t_rcyc {  
17444         STAT          cycstat;    /* 周期ハンドラの動作状態 */  
17445         RELTIM       lefttim;     /* 次に周期ハンドラを起動する時刻までの  
17446                                相対時間 */  
17447         /* 以下は、マルチプロセッサ対応カーネルの場合 */  
17448         ID          prcid;       /* 割付けプロセッサのID */  
17449     } T_RCYC;
```

17450

17451 アラームハンドラの生成情報のパケット形式【NGKI4025】

17452

```
17453     typedef struct t_calm {  
17454         ATR          almatr;      /* アラームハンドラ属性 */  
17455         intptr_t     exinf;       /* アラームハンドラの拡張情報 */  
17456         ALMHDR       almhdr;     /* アラームハンドラの先頭番地 */  
17457     } T_CALM;
```

17458

17459 アラームハンドラの現在状態のパケット形式【NGKI4026】

17460

```
17461     typedef struct t_ralm {  
17462         STAT          almstat;    /* アラームハンドラの動作状態 */  
17463         RELTIM        lefttim;    /* アラームハンドラを起動する時刻までの  
17464                                 相対時間 */  
17465         /* 以下は、マルチプロセッサ対応カーネルの場合 */  
17466         ID            prcid;      /* 割付けプロセッサのID */  
17467     } T_RALM;
```

17468

17469 オーバランハンドラの定義情報のパケット形式【NGKI4027】

17470

```
17471     typedef struct t_dovr {  
17472         ATR          ovratr;      /* オーバランハンドラ属性 */  
17473         OVRHDR       ovrhdr;     /* オーバランハンドラ先頭番地 */  
17474     } T_DOVR;
```

17475

17476 オーバランハンドラの現在状態のパケット形式【NGKI4028】

17477

```
17478     typedef struct t_rovr {  
17479         STAT          ovrstat;    /* オーバランハンドラの動作状態 */  
17480         OVRTIM        leftotm;    /* 残りプロセッサ時間 */  
17481     } T_ROVR;
```

17482

17483 (7) システム状態管理機能

17484

17485 システムの現在状態のパケット形式

17486

17487 ☆未完成

17488

17489 (8) メモリオブジェクト管理機能

17490

17491 メモリオブジェクトの登録情報のパケット形式【NGKI4029】

17492

```
17493     typedef struct t_amem {  
17494         ATR          mematr      /* メモリオブジェクト属性 */  
17495         void *       base        /* 登録するメモリ領域先頭番地 */  
17496         SIZE         size        /* 登録するメモリ領域のサイズ (バイト数) */  
17497     } T_AMEM;
```

17498

17499 物理メモリ領域の登録情報のパケット形式【NGKI4030】

17500

```

17501     typedef struct t_apma {
17502         ATR          mematr      /* メモリオブジェクト属性 */
17503         void *       base        /* 登録するメモリ領域の先頭番地 */
17504         SIZE         size        /* 登録するメモリ領域のサイズ (バイト数) */
17505         void *       paddr       /* 登録するメモリ領域の物理アドレスの先頭
17506                                番地 */
17507     } T_APMA;
17508
17509     メモリオブジェクトの現在状態のパケット形式
17510
17511     ☆未完成
17512
17513     (9) 割込み管理機能
17514
17515     割込み要求ラインの属性の設定情報のパケット形式【NGKI4031】
17516
17517     typedef struct t_cint {
17518         ATR          intatr;      /* 割込み要求ライン属性 */
17519         PRI          intpri;      /* 割込み優先度 */
17520     } T_CINT;
17521
17522     割込みサービ斯拉ーチンの生成情報のパケット形式【NGKI4032】
17523
17524     typedef struct t_cisr {
17525         ATR          isratr;      /* 割込みサービ斯拉ーチン属性 */
17526         intptr_t     exinf;       /* 割込みサービ斯拉ーチンの拡張情報 */
17527         INTNO        intno;       /* 割込みサービ斯拉ーチンを登録する割込
17528                                み番号 */
17529         ISR          isr;         /* 割込みサービ斯拉ーチンの先頭番地 */
17530         PRI          isrpri;      /* 割込みサービ斯拉ーチン優先度 */
17531     } T_CISR;
17532
17533     割込みサービ斯拉ーチンの現在状態のパケット形式
17534
17535     ☆未完成
17536
17537     割込みハンドラの定義情報のパケット形式【NGKI4033】
17538
17539     typedef struct t_dinh {
17540         ATR          inhatr;      /* 割込みハンドラ属性 */
17541         INTHDR        inthdr;     /* 割込みハンドラ先頭番地 */
17542     } T_DINH;
17543
17544     割込み要求ラインの現在状態のパケット形式
17545
17546     ☆未完成
17547
17548     (10) CPU例外管理機能
17549
17550     CPU例外ハンドラの定義情報のパケット形式【NGKI4034】

```

```

17551
17552     typedef struct t_dexc {
17553         ATR          excatr;      /* CPU例外ハンドラ属性 */
17554         EXCHDR       exchdr;      /* CPU例外ハンドラの先頭番地 */
17555     } T_DEXC;
17556
17557 (11) 拡張サービスコール管理機能
17558
17559 拡張サービスコールの定義情報のパケット形式【NGKI4035】
17560
17561     typedef struct t_dsvc {
17562         ATR          svcatr       /* 拡張サービスコール属性 */
17563         EXTSVC       svcrttn      /* 拡張サービスコールの先頭番地 */
17564         SIZE         stksz        /* 拡張サービスコールで使用するスタック
17565                                サイズ */
17566     } T_DSVC;
17567
17568 (12) システム構成管理機能
17569
17570 コンフィギュレーション情報のパケット形式
17571
17572 ☆未完成
17573
17574 バージョン情報のパケット形式
17575
17576 ☆未完成
17577
17578 5.4 定数とマクロ
17579
17580 5.4.1 TOPPERS共通定数
17581
17582 (1) 一般定数
17583
17584     NULL                無効ポインタ
17585
17586     true                1      真
17587     false               0      偽
17588
17589     E_OK                0      正常終了
17590
17591 (2) 整数型に格納できる最大値と最小値
17592
17593     INT8_MAX            int8_tに格納できる最大値 (オプション, C99準拠)
17594     INT8_MIN            int8_tに格納できる最小値 (オプション, C99準拠)
17595     UINT8_MAX           uint8_tに格納できる最大値 (オプション, C99準拠)
17596     INT16_MAX           int16_tに格納できる最大値 (C99準拠)
17597     INT16_MIN           int16_tに格納できる最小値 (C99準拠)
17598     UINT16_MAX          uint16_tに格納できる最大値 (C99準拠)
17599     INT32_MAX           int32_tに格納できる最大値 (C99準拠)
17600     INT32_MIN           int32_tに格納できる最小値 (C99準拠)

```



17601	UINT32_MAX		uint32_tに格納できる最大値 (C99準拠)
17602	INT64_MAX		int64_tに格納できる最大値 (オプション, C99準拠)
17603	INT64_MIN		int64_tに格納できる最小値 (オプション, C99準拠)
17604	UINT64_MAX		uint64_tに格納できる最大値 (オプション, C99準拠)
17605	INT128_MAX		int128_tに格納できる最大値 (オプション, C99準拠)
17606	INT128_MIN		int128_tに格納できる最小値 (オプション, C99準拠)
17607	UINT128_MAX		uint128_tに格納できる最大値 (オプション, C99準拠)
17608			
17609	INT_LEAST8_MAX		int_least8_tに格納できる最大値 (C99準拠)
17610	INT_LEAST8_MIN		int_least8_tに格納できる最小値 (C99準拠)
17611	UINT_LEAST8_MAX		uint_least8_tに格納できる最大値 (C99準拠)
17612	INT_MAX		int_tに格納できる最大値 (C90準拠)
17613	INT_MIN		int_tに格納できる最小値 (C90準拠)
17614	UINT_MAX		uint_tに格納できる最大値 (C90準拠)
17615	LONG_MAX		long_tに格納できる最大値 (C90準拠)
17616	LONG_MIN		long_tに格納できる最小値 (C90準拠)
17617	ULONG_MAX		ulong_tに格納できる最大値 (C90準拠)
17618			
17619	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)
17620			
17621	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)
17622			
17623	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)
17624			
17625	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)
17626			
17627			
17628	(3) 整数型のビット数		
17629			
17630	CHAR_BIT		char型のビット数 (C90準拠)
17631			
17632	(4) オブジェクト属性		
17633			
17634	TA_NULL	0U	オブジェクト属性を指定しない
17635			
17636	(5) タイムアウト指定		
17637			
17638	TMO_POL	0	ポーリング
17639	TMO_FEVR	-1	永久待ち
17640	TMO_NBLK	-2	ノンブロッキング
17641			
17642	(6) アクセス許可パターン		
17643			
17644	TACP_KERNEL	0U	カーネルドメインのみにアクセスを許可
17645	TACP_SHARED	~0U	すべての保護ドメインにアクセスを許可
17646			
17647	5.4.2 TOPPERS共通マクロ		
17648			
17649	(1) 整数定数を作るマクロ		
17650			

17651	INT8_C(val)	int_least8_t型の定数を作るマクロ (C99準拠)
17652	UINT8_C(val)	uint_least8_t型の定数を作るマクロ (C99準拠)
17653	INT16_C(val)	int16_t型の定数を作るマクロ (C99準拠)
17654	UINT16_C(val)	uint16_t型の定数を作るマクロ (C99準拠)
17655	INT32_C(val)	int32_t型の定数を作るマクロ (C99準拠)
17656	UINT32_C(val)	uint32_t型の定数を作るマクロ (C99準拠)
17657	INT64_C(val)	int64_t型の定数を作るマクロ (オプション, C99準拠)
17658	UINT64_C(val)	uint64_t型の定数を作るマクロ (オプション, C99準拠)
17659	INT128_C(val)	int128_t型の定数を作るマクロ (オプション, C99準拠)
17660	UINT128_C(val)	uint128_t型の定数を作るマクロ (オプション, C99準拠)
17661		
17662	UINT_C(val)	uint_t型の定数を作るマクロ
17663	ULONG_C(val)	ulong_t型の定数を作るマクロ
17664		
17665	(2) 型に関する情報を取り出すためのマクロ	
17666		
17667	offsetof(structure, field)	構造体structure中のフィールドfieldの バイト位置を返すマクロ (C90準拠)
17668		
17669		
17670	alignof(type)	型typeのアラインメント単位を返すマクロ
17671		
17672	ALIGN_TYPE(addr, type)	番地addrが型typeに対してアラインしてい るかどうかを返すマクロ
17673		
17674		
17675	(3) assertマクロ	
17676		
17677	assert(exp)	expが成立しているかを検査するマクロ (C90準拠)
17678		
17679	(4) コンパイラの拡張機能のためのマクロ	
17680		
17681	inline	インライン関数
17682	Inline	ファイルローカルなインライン関数
17683	asm	インラインアセンブラ
17684	Asm	インラインアセンブラ (最適化抑止)
17685	throw()	例外を発生しない関数
17686	NoReturn	リターンしない関数
17687		
17688	(5) エラーコード生成・分解マクロ	
17689		
17690	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
17691		
17692		
17693	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
17694		
17695	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
17696		
17697		
17698	(6) アクセス許可パターン生成マクロ	
17699		
17700	TACP(domid)	domidで指定される保護ドメインに属する処理単位の

17701 みにアクセスを許可するアクセス許可パターン

17702

#### 17703 5.4.3 カーネル共通定数

17704

##### 17705 (1) オブジェクト属性

17706

17707 TA\_TPRI 0x01U タスクの待ち行列をタスクの優先度順に

17708

##### 17709 (2) 保護ドメインID

17710

17711 TDOM\_SELF 0 自タスクの属する保護ドメイン

17712 TDOM\_KERNEL -1 カーネルドメイン

17713 TDOM\_NONE -2 無所属（保護ドメインに属さない）

17714

##### 17715 (3) その他のカーネル共通定数

17716

17717 TCLS\_SELF 0 自タスクの属するクラス

17718

17719 TPRC\_NONE 0 割付けプロセッサの指定がない

17720 TPRC\_INI 0 初期割付けプロセッサ

17721

17722 TSK\_SELF 0 自タスク指定

17723 TSK\_NONE 0 該当するタスクがない

17724

17725 TPRI\_SELF 0 自タスクのベース優先度の指定

17726 TPRI\_INI 0 タスクの起動時優先度の指定

17727

17728 TIPM\_ENAALL 0 割込み優先度マスク全解除

17729

#### 17730 5.4.4 カーネル共通マクロ

17731

##### 17732 (1) オブジェクト属性を作るマクロ

17733

17734 TA\_DOM(domid) domidで指定される保護ドメインに属する

17735 TA\_CLS(clsid) clsidで指定されるクラスに属する

17736

##### 17737 (2) サービスコールの呼出し方法を指定するマクロ

17738

17739 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

17740

#### 17742 5.4.5 カーネルの機能毎の定数

17743

##### 17744 (1) タスク管理機能

17745

17746 TA\_ACT 0x02U タスクの生成時にタスクを起動する

17747 TA\_RSTR 0x04U 生成するタスクを制約タスクとする

17748 TA\_FPU FPUレジスタをコンテキストに含める

17749

17750 TTS\_RUN 0x01U 実行状態

17751	TTS_RDY	0x02U	実行可能状態
17752	TTS_WAI	0x04U	待ち状態
17753	TTS_SUS	0x08U	強制待ち状態
17754	TTS_WAS	0x0cU	二重待ち状態
17755	TTS_DMT	0x10U	休止状態
17756			
17757	TTW_SLP	0x0001U	起床待ち
17758	TTW_DLY	0x0002U	時間経過待ち
17759	TTW_SEM	0x0004U	セマフォの資源獲得待ち
17760	TTW_FLG	0x0008U	イベントフラグ待ち
17761	TTW_SDTQ	0x0010U	データキューへの送信待ち
17762	TTW_RDTQ	0x0020U	データキューからの受信待ち
17763	TTW_SPDQ	0x0100U	優先度データキューへの送信待ち
17764	TTW_RPDQ	0x0200U	優先度データキューからの受信待ち
17765	TTW_MBX	0x0040U	メールボックスからの受信待ち
17766	TTW_MTX	0x0080U	ミューテックスのロック待ち状態
17767	TTW_SMBF	0x0400U	メッセージバッファへの送信待ち
17768	TTW_RMBF	0x0800U	メッセージバッファからの受信待ち
17769	TTW_MPF	0x2000U	固定長メモリブロックの獲得待ち
17770			
17771	TA_FPUの値は、ターゲット定義とする.		
17772			
17773	(3) タスク例外処理機能		
17774			
17775	TTEX_ENA	0x01U	タスク例外処理許可状態
17776	TTEX_DIS	0x02U	タスク例外処理禁止状態
17777			
17778	(4) 同期・通信機能		
17779			
17780	イベントフラグ		
17781			
17782	TA_WMUL	0x02U	複数のタスクが待つのを許す
17783	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
17784			
17785	TWF_ORW	0x01U	イベントフラグのOR待ちモード
17786	TWF_ANDW	0x02U	イベントフラグのAND待ちモード
17787			
17788	メールボックス		
17789			
17790	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
17791			
17792	スピンロック		
17793			
17794	TSPN_UNL	0x01U	取得されていない状態
17795	TSPN_LOC	0x02U	取得されている状態
17796			
17797	(6) 時間管理機能		
17798			
17799	周期ハンドラ		
17800			

17801	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
17802	TA_PHS	0x04U	周期ハンドラを生成した時刻を基準時刻とする
17803			
17804	TCYC_STP	0x01U	周期ハンドラが動作していない状態
17805	TCYC_STA	0x02U	周期ハンドラが動作している状態
17806			
17807	アラームハンドラ		
17808			
17809	TALM_STP	0x01U	アラームハンドラが動作していない状態
17810	TALM_STA	0x02U	アラームハンドラが動作している状態
17811			
17812	オーバランハンドラ		
17813			
17814	TOVR_STP	0x01U	オーバランハンドラが動作していない状態
17815	TOVR_STA	0x02U	オーバランハンドラが動作している状態
17816			
17817	(8) メモリオブジェクト管理機能		
17818			
17819	TA_NOWRITE	0x01U	書き込みアクセス禁止
17820	TA_NOREAD	0x02U	読み出しアクセス禁止
17821	TA_EXEC	0x04U	実行アクセス許可
17822	TA_MEMINI	0x08U	メモリの初期化を行う
17823	TA_MEMPRSV	0x10U	メモリの初期化を行わない
17824	TA_SDATA	0x20U	ショートデータ領域に配置
17825	TA_UNCACHE	0x40U	キャッシュ禁止
17826	TA_IODEV	0x80U	周辺デバイスの領域
17827	TA_WTHROUGH		ライトスルーキャッシュを用いる
17828			
17829	TPM_WRITE	0x01U	書き込みアクセス権のチェック
17830	TPM_READ	0x02U	読み出しアクセス権のチェック
17831	TPM_EXEC	0x04U	実行アクセス権のチェック
17832			
17833	TA_WTHROUGHの値は、ターゲット定義とする。		
17834			
17835	(9) 割込み管理機能		
17836			
17837	TA_ENAINT	0x01U	割込み要求禁止フラグをクリア
17838	TA_EDGE	0x02U	エッジトリガ
17839	TA_POSEDGE		ポジティブエッジトリガ
17840	TA_NEGEDGE		ネガティブエッジトリガ
17841	TA_BOTHEDGE		両エッジトリガ
17842	TA_LOWLEVEL		ローレベルトリガ
17843	TA_HIGHLEVEL		ハイレベルトリガ
17844			
17845	TA_NONKERNEL	0x02U	カーネル管理外の割込み
17846			
17847	TA_POSEDGE, TA_NEGEDGE, TA_BOTHEDGE, TA_LOWLEVEL, TA_HIGHLEVELの値は、		
17848	ターゲット定義とする。		
17849			
17850	(10) CPU例外管理機能		

17851		
17852	TA_DIRECT	CPU例外ハンドラを直接呼び出す
17853		
17854	TA_DIRECTの値は、ターゲット定義とする.	
17855		
17856	5.4.6 カーネルの機能毎のマクロ	
17857		
17858	(1) タスク管理機能	
17859		
17860	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
17861		STK_T型の配列の要素数
17862	ROUND_STK_T(sz)	要素数COUNT_STK_T(sz)のSTK_T型の配列のサイズ (sz
17863		を, STK_T型のサイズの倍数になるように大きい方に
17864		丸めた値)
17865		
17866	(4) 同期・通信機能	
17867		
17868	TSZ_DTQMB(dtqcnt)	dtqcntで指定した数のデータを格納できるデータ
17869		キュー管理領域のサイズ (バイト数)
17870	TCNT_DTQMB(dtqcnt)	dtqcntで指定した数のデータを格納できるデータ
17871		キュー管理領域を確保するために必要なMB_T型の配
17872		列の要素数
17873		
17874	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
17875		タキュー管理領域のサイズ (バイト数)
17876	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
17877		タキュー管理領域を確保するために必要なMB_T型の
17878		配列の要素数
17879		
17880	TSZ_MBFMB(msgcnt, msgsz)	msgszで指定したサイズのメッセージを,
17881		msgcntで指定した数だけ格納できるメッセー
17882		ジバッファ管理領域のサイズ (バイト数)
17883	TCNT_MBFMB(msgcnt, msgsz)	msgszで指定したサイズのメッセージを,
17884		msgcntで指定した数だけ格納できるメッセー
17885		ジバッファ管理領域を確保するために必要
17886		なMB_T型の配列の要素数
17887		
17888	(5) メモリプール管理機能	
17889		
17890	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
17891		リプール領域を確保するために, 固定長メモリブロッ
17892		ク1つあたりに必要なMPF_T型の配列の要素数を求め
17893		るマクロ
17894	ROUND_MPF_T(blksz)	要素数COUNT_MPF_T(blksz)のMPF_T型の配列のサイズ
17895		(blkszを, MPF_T型のサイズの倍数になるように大き
17896		い方に丸めた値)
17897		
17898	TSZ_MPFMB(blkcnt)	blkcntで指定した数の固定長メモリブロックを管理
17899		することができる固定長メモリプール管理領域のサ
17900		イズ (バイト数)

17901	TCNT_MPFMB(blkent)	blkentで指定した数の固定長メモリブロックを管理
17902		することができる固定長メモリプール管理領域を確
17903		保するために必要なMB_T型の配列の要素数
17904		
17905	5.5 構成マクロ	
17906		
17907	5.5.1 TOPPERS共通構成マクロ	
17908		
17909	(1) 相対時間の範囲	
17910		
17911	TMAX_RELTIM	相対時間に指定できる最大値
17912		
17913	5.5.2 カーネル共通構成マクロ	
17914		
17915	(1) サポートする機能	
17916		
17917	TOPPERS_SUPPORT_PROTECT	保護機能対応のカーネル
17918	TOPPERS_SUPPORT_MULTI_PRC	マルチプロセッサ対応のカーネル
17919	TOPPERS_SUPPORT_DYNAMIC_CRE	動的生成対応のカーネル
17920		
17921	(2) 優先度の範囲	
17922		
17923	TMIN_TPRI	タスク優先度の最小値 (=1)
17924	TMAX_TPRI	タスク優先度の最大値
17925		
17926	(3) プロセッサの数	
17927		
17928	TNUM_PRCID	プロセッサの数
17929		
17930	(4) 特殊な役割を持ったプロセッサ	
17931		
17932	TOPPERS_MASTER_PRCID	マスタプロセッサのID番号
17933	TOPPERS_SYSTIM_PRCID	システム時刻管理プロセッサのID番号
17934		
17935	(5) タイマ方式	
17936		
17937	TOPPERS_SYSTIM_LOCAL	ローカルタイマ方式の場合にマクロ定義
17938	TOPPERS_SYSTIM_GLOBAL	グローバルタイマ方式の場合にマクロ定義
17939		
17940	(6) バージョン情報	
17941		
17942	TKERNEL_MAKER	カーネルのメーカーコード (=0x0118)
17943	TKERNEL_PRID	カーネルの識別番号
17944	TKERNEL_SPVER	カーネル仕様のバージョン番号
17945	TKERNEL_PRVER	カーネルのバージョン番号
17946		
17947	5.5.3 カーネルの機能毎の構成マクロ	
17948		
17949	(1) タスク管理機能	
17950		

17951	TMAX_ACTCNT	タスクの起動要求キューイング数の最大値
17952		
17953	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）
17954		
17955		
17956	(2) タスク付属同期機能	
17957		
17958	TMAX_WUPCNT	タスクの起床要求キューイング数の最大値
17959		
17960	(3) タスク例外処理機能	
17961		
17962	TBIT_TEXPTN	タスク例外要因のビット数（TEXPTNの有効ビット数）
17963		
17964	(4) 同期・通信機能	
17965		
17966	セマフォ	
17967		
17968	TMAX_MAXSEM	セマフォの最大資源数の最大値
17969		
17970	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）
17971		
17972		
17973	イベントフラグ	
17974		
17975	TBIT_FLGPTN	イベントフラグのビット数（FLGPTNの有効ビット数）
17976		
17977	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）
17978		
17979		
17980		
17981	データキュー	
17982		
17983	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）
17984		
17985		
17986		
17987	優先度データキュー	
17988		
17989	TMIN_DPRI	データ優先度の最小値（=1）
17990	TMAX_DPRI	データ優先度の最大値
17991		
17992	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）
17993		
17994		
17995		
17996	メールボックス	
17997		
17998	TMIN_MPRI	メッセージ優先度の最小値（=1）
17999	TMAX_MPRI	メッセージ優先度の最大値
18000		



18001	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）
18002		
18003		
18004		
18005	ミューテックス	
18006		
18007	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）
18008		
18009		
18010		
18011	メッセージバッファ	
18012		
18013	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）
18014		
18015		
18016		
18017	スピンロック	
18018		
18019	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）
18020		
18021		
18022		
18023	(5) メモリプール管理機能	
18024		
18025	固定長メモリプール	
18026		
18027	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）
18028		
18029		
18030		
18031	(6) 時間管理機能	
18032		
18033	システム時刻管理	
18034		
18035	TIC_NUME	タイムティックの周期（単位はミリ秒）の分子
18036	TIC_DENO	タイムティックの周期（単位はミリ秒）の分母
18037		
18038	TOPPERS_SUPPORT_GET_UTM	get_utmがサポートされている
18039		
18040	周期ハンドラ	
18041		
18042	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）
18043		
18044		
18045		
18046	アラームハンドラ	
18047		
18048	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）
18049		
18050		

18051			
18052	オーバランハンドラ		
18053			
18054	TMAX_OVRTIM	プロセッサ時間に指定できる最大値	
18055			
18056	TOPPERS_SUPPORT_OVRHDR	オーバランハンドラ機能がサポートされて	
18057		いる	
18058			
18059	(7) システム状態管理機能		
18060			
18061	なし		
18062			
18063	(8) メモリオブジェクト管理機能		
18064			
18065	TOPPERS_SUPPORT_ATT_MOD	ATT_MOD/ATA_MODがサポートされている	
18066	TOPPERS_SUPPORT_ATT_PMA	ATT_PMA/ATA_PMA/att_pmaがサポートさ	
18067		れている	
18068			
18069	(9) 割込み管理機能		
18070			
18071	TMIN_INTPRI	割込み優先度の最小値 (最高値)	
18072	TMAX_INTPRI	割込み優先度の最大値 (最低値, =-1)	
18073			
18074	TMIN_ISRPRI	割込みサービスルーチン優先度の最小値 (=1)	
18075	TMAX_ISRPRI	割込みサービスルーチン優先度の最大値	
18076			
18077	TOPPERS_SUPPORT_DIS_INT	dis_intがサポートされている	
18078	TOPPERS_SUPPORT_ENA_INT	ena_intがサポートされている	
18079			
18080	(10) CPU例外管理機能		
18081			
18082	なし		
18083			
18084	(11) 拡張サービスコール管理機能		
18085			
18086	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	
18087		いカーネルでは, 静的APIによって登録された拡張サービ	
18088		スコールの数に一致)	
18089			
18090	(12) システム構成管理機能		
18091			
18092	なし		
18093			
18094	5.6 エラーコード一覧		
18095			
18096	(1) メインエラーコード		
18097			
18098	E_SYS	-5	システムエラー
18099	E_NOSPT	-9	未サポート機能
18100	E_RSFN	-10	予約機能コード

18101	E_RSATR	-11	予約属性
18102	E_PAR	-17	パラメータエラー
18103	E_ID	-18	不正ID番号
18104	E_CTX	-25	コンテキストエラー
18105	E_MACV	-26	メモリアクセス違反
18106	E_OACV	-27	オブジェクトアクセス違反
18107	E_ILUSE	-28	サービスコール不正使用
18108	E_NOMEM	-33	メモリ不足
18109	E_NOID	-34	ID番号不足
18110	E_NORES	-35	資源不足
18111	E_OBJ	-41	オブジェクト状態エラー
18112	E_NOEXS	-42	オブジェクト未登録
18113	E_QOVR	-43	キューイングオーバーフロー
18114	E_RLWAI	-49	待ち禁止状態または待ち状態の強制解除
18115	E_TMOUT	-50	ポーリング失敗またはタイムアウト
18116	E_DLT	-51	待ちオブジェクトの削除または再初期化
18117	E_CLS	-52	待ちオブジェクトの状態変化
18118	E_WBLK	-57	ノンブロッキング受付け
18119	E_BOVR	-58	バッファオーバーフロー

18120

18121 5.7 機能コード一覧【NGKI4036】

18122

18123

	-0	-1	-2	-3
18124				
18125				
18126	-0x01	予約	予約	予約
18127	-0x05	act_tsk	iact_tsk	can_act
18128	-0x09	ter_tsk	chg_pri	get_pri
18129	-0x0d	slp_tsk	tslp_tsk	wup_tsk
18130	-0x11	can_wup	rel_wai	irel_wai
18131	-0x15	dis_wai	idis_wai	ena_wai
18132	-0x19	sus_tsk	rsm_tsk	dly_tsk
18133	-0x1d	ras_tex	iras_tex	dis_tex
18134	-0x21	sns_tex	ref_tex	予約
18135	-0x25	sig_sem	isig_sem	wai_sem
18136	-0x29	twai_sem	予約	予約
18137	-0x2d	set_flg	iset_flg	clr_flg
18138	-0x31	pol_flg	twai_flg	予約
18139	-0x35	snd_dtq	psnd_dtq	ipsnd_dtq
18140	-0x39	fsnd_dtq	ifsnd_dtq	rcv_dtq
18141	-0x3d	trcv_dtq	予約	予約
18142	-0x41	snd_pdq	psnd_pdq	ipsnd_pdq
18143	-0x45	rcv_pdq	prcv_pdq	trcv_pdq
18144	-0x49	snd_mbx	rcv_mbx	prcv_mbx
18145	-0x4d	loc_mtx	ploc_mtx	tloc_mtx
18146	-0x51	snd_mbf	psnd_mbf	tsnd_mbf
18147	-0x55	prcv_mbf	trcv_mbf	予約
18148	-0x59	get_mpf	pget_mpf	tget_mpf
18149	-0x5d	get_tim	get_utm	予約
18150	-0x61	sta_cyc	stp_cyc	予約

18151	-0x65	sta_alm	ista_alm	stp_alm	istp_alm
18152	-0x69	sta_ovr	ista_ovr	stp_ovr	istp_ovr
18153	-0x6d	sac_sys	ref_sys	rot_rdq	irotd_rdq
18154	-0x71	get_did	予約	get_tid	iget_tid
18155	-0x75	loc_cpu	iloc_cpu	unl_cpu	iunl_cpu
18156	-0x79	dis_dsp	ena_dsp	sns_ctx	sns_loc
18157	-0x7d	sns_dsp	sns_dpn	sns_ker	ext_ker
18158	-0x81	att_mem	det_mem	sac_mem	prb_mem
18159	-0x85	ref_mem	予約	att_pma	予約
18160	-0x89	cfg_int	dis_int	ena_int	ref_int
18161	-0x8d	chg_ipm	get_ipm	予約	予約
18162	-0x91	xsns_dpn	xsns_xpn	予約	予約
18163	-0x95	ref_cfg	ref_ver	予約	予約
18164	-0x99	予約	予約	予約	予約
18165	-0x9d	予約	予約	予約	予約
18166	-0xa1	予約	ini_sem	ini_flg	ini_dtq
18167	-0xa5	ini_pdq	ini_mbx	ini_mtx	ini_mbf
18168	-0xa9	ini_mpf	予約	予約	予約
18169	-0xad	予約	予約	予約	予約
18170	-0xb1	ref_tsk	ref_sem	ref_flg	ref_dtq
18171	-0xb5	ref_pdq	ref_mbx	ref_mtx	ref_mbf
18172	-0xb9	ref_mpf	ref_cyc	ref_alm	ref_isr
18173	-0xbd	ref_spn	予約	予約	予約
18174	-0xc1	acre_tsk	acre_sem	acre_flg	acre_dtq
18175	-0xc5	acre_pdq	acre_mbx	acre_mtx	acre_mbf
18176	-0xc9	acre_mpf	acre_cyc	acre_alm	acre_isr
18177	-0xcd	acre_spn	予約	予約	予約
18178	-0xd1	del_tsk	del_sem	del_flg	del_dtq
18179	-0xd5	del_pdq	del_mbx	del_mtx	del_mbf
18180	-0xd9	del_mpf	del_cyc	del_alm	del_isr
18181	-0xdd	del_spn	予約	予約	予約
18182	-0xe1	sac_tsk	sac_sem	sac_flg	sac_dtq
18183	-0xe5	sac_pdq	予約	sac_mtx	sac_mbf
18184	-0xe9	sac_mpf	sac_cyc	sac_alm	sac_isr
18185	-0xed	sac_spn	予約	予約	予約
18186	-0xf1	def_tex	def_ovr	def_inh	def_exc
18187	-0xf5	def_svc	予約	予約	予約
18188	-0xf9	予約	予約	予約	予約
18189	-0xfd	予約	予約	予約	予約
18190	-0x101	mact_tsk	imact_tsk	migt_tsk	予約
18191	-0x105	msta_cyc	予約	msta_alm	imsta_alm
18192	-0x109	mrot_rdq	imrot_rdq	get_pid	iget_pid
18193	-0x10d	予約	予約	予約	予約
18194	-0x111	loc_spn	iloc_spn	try_spn	itry_spn
18195	-0x115	unl_spn	iunl_spn	予約	予約
18196	-0x119	予約	予約	予約	予約
18197	-0x11d	予約	予約	予約	予約
18198	-----				
18199					
18200	【μ ITRON4.0仕様との関係】				

18201  
18202 サービスコールの機能コードを割り当てなおした.  
18203

18204 5.8 カーネルオブジェクトに対するアクセスの種別  
18205

オブジェクトの種類	通常操作1	通常操作2	管理操作	参照操作
メモリオブジェクト	書込み	読出し 実行	det_mem sac_mem	ref_mem prb_mem
タスク	act_tsk mact_tsk can_act mig_tsk wup_tsk can_wup	ter_tsk chg_pri rel_wai sus_tsk rsm_tsk dis_wai ena_wai ras_tex sta_ovr stp_ovr	del_tsk sac_tsk def_tex	get_pri ref_tsk ref_tex ref_ovr
セマフォ	sig_sem	wai_sem pol_sem twai_sem	del_sem ini_sem sac_sem	ref_sem
イベントフラグ	set_flg clr_flg	wai_flg pol_flg twai_flg	del_flg ini_flg sac_flg	ref_flg
データキュー	snd_dtq psnd_dtq tsnd_dtq fsnd_dtq	rcv_dtq prcv_dtq trcv_dtq	del_dtq ini_dtq sac_dtq	ref_dtq
優先度データキュー	snd_pdq psnd_pdq tsnd_pdq	rcv_pdq prcv_pdq trcv_pdq	del_pdq ini_pdq sac_pdq	ref_pdq
メッセージバッファ	snd_mbf psnd_mbf tsnd_mbf	rcv_mbf prcv_mbf trcv_mbf	del_mbf ini_mbf sac_mbf	ref_mbf
ミューテックス	loc_mtx ploc_mtx tloc_mtx unl_mtx	-	del_mtx ini_mtx sac_mtx	ref_mtx
スピンロック	loc_spn try_spn	-	del_spn sac_spn	ref_spn

18251		unl_spn			
18252	-----				
18253	固定長メモリプール	get_mpf	rel_mpf	del_mpf	ref_mpf
18254		pget_mpf		ini_mpf	
18255		tget_mpf		sac_mpf	
18256	-----				
18257	周期ハンドラ	sta_cyc	stp_cyc	del_cyc	ref_cyc
18258		msta_cyc		sac_cyc	
18259	-----				
18260	アラームハンドラ	sta_alm	stp_alm	del_alm	ref_alm
18261		msta_alm		sac_alm	
18262	-----				
18263	割込みサービスルーチン	-	-	del_isr	ref_isr
18264				sac_isr	
18265	-----				
18266	システム状態	rot_rdq	loc_cpu	acre_yyy	get_tim
18267		mrot_rdq	unl_cpu	att_mem	get_ipm
18268		dis_dsp	dis_int	att_pma	ref_sys
18269		ena_dsp	ena_int	cfg_int	ref_int
18270			chg_ipm	def_inh	ref_cfg
18271				def_exc	ref_ver
18272				def_svc	
18273				def_ovr	
18274	-----				

18275

18276 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：

18277

- 18278 ・ 自タスクへの操作 (ext\_tsk, get\_inf, slp\_tsk, tslp\_tsk, dly\_tsk,
- 18279   dis\_tex, ena\_tex)
- 18280 ・ タスク例外状態参照 (sns\_tex)
- 18281 ・ 性能評価用システム時刻の参照 (get\_utm)
- 18282 ・ システム状態参照 (get\_tid, get\_did, get\_pid, sns\_ctx, sns\_loc,
- 18283   sns\_dsp, sns\_dpn, sns\_ker)
- 18284 ・ CPU例外発生時の状態参照 (xsns\_dpn, xsns\_xpn)
- 18285 ・ 拡張サービスコールの呼出し (cal\_svc)

18286

18287 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：

18288

- 18289 ・ システム状態のアクセス許可ベクタの設定 (sac\_sys)
- 18290 ・ カーネルの終了 (ext\_ker)
- 18291 ・ 非タスクコンテキスト専用のサービスコール

18292

## 18293 【補足説明】

18294

18295 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  
 18296 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  
 18297 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  
 18298 ネルドメインのみから呼び出すことができる。

18299

18300 【μITRON4.0/PX仕様との関係】

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

get\_priは、 $\mu$ 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 ( $\mu$ 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]
- 割込み要求禁止フラグがサポートされているか [NGKI0260] [NGKI0261]
- 割込み要求禁止フラグの振舞いを仕様と異なるものとするか [NGKI0261]
- 割込み要求ラインのトリガモードの設定がサポートされているか [NGKI0267]
- 割込み要求ラインをエッジトリガに設定する場合に、ポジティブエッジトリガかネガティブエッジトリガか両エッジトリガかを設定できるか [NGKI0265]
- 割込み要求ラインをレベルトリガに設定する場合に、ローレベルトリガかハイレベルトリガかを設定できるか [NGKI0266]
- あるプロセッサで割込み要求禁止フラグを動的にセット／クリアしても、他のプロセッサに対しては割込みがマスク／マスク解除されないものとするか

- 18351 [M] [NGKI0281]  
18352  
18353 • 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  
18354 場合の設定方法 [NGKI0288]  
18355  
18356 • 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）  
18357 [NGKI0289]  
18358  
18359 • 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  
18360 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  
18361 方法 [NGKI0292]  
18362  
18363 • 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  
18364 [NGKI0295]  
18365  
18366 • 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  
18367 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]  
18368  
18369 • CPU例外ハンドラ番号の付与方法 [NGKI0306]  
18370  
18371 • 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  
18372 [NGKI0314]  
18373  
18374 • 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]  
18375 [NGKI2774]  
18376  
18377 • 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  
18378 か [P] [NGKI0113]  
18379  
18380 • 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  
18381 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]  
18382  
18383 • 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  
18384 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]  
18385  
18386 • 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  
18387 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  
18388 とした場合に、どのCPU例外ハンドラが起動されるか [P] [NGKI0411]  
18389  
18390 • 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  
18391 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  
18392 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  
18393 [P] [NGKI0413]  
18394  
18395 • 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  
18396 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）  
18397 を参照する方法 [P] [NGKI0414]  
18398  
18399 • 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  
18400 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]



- 18401
- 18402     • 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]
- 18403         [NGKI0423]
- 18404
- 18405     • 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]
- 18406
- 18407     • 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
- 18408         アクセスできるものとするか [P] [NGKI0441]
- 18409
- 18410     • 使用できるクラスのID番号とその属性 [M] [NGKI0107]
- 18411
- 18412     • どのプロセッサをマスタプロセッサとするか [M] [NGKI0101]
- 18413
- 18414     • 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
- 18415         できるか [M] [NGKI0108]
- 18416
- 18417     • 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
- 18418         サとするか [M] [NGKI0111]
- 18419
- 18420     • `int8_t`, `uint8_t`, `int64_t`, `uint64_t`, `int128_t`, `uint128_t`, `float32_t`,
- 18421         `double64_t`が使用できるか [NGKI0488] [NGKI0490]
- 18422
- 18423     • ターゲット定義のタスク属性 [NGKI1016]
- 18424
- 18425     • タスクが用いるスタック領域のサイズの最小値 [NGKI1042]
- 18426
- 18427     • タスクのシステムスタック領域のサイズの最小値 [P] [NGKI1044]
- 18428
- 18429     • 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]
- 18430         [NGKI1056]
- 18431
- 18432     • 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
- 18433         で確保する方法 [P] [NGKI1059]
- 18434
- 18435     • 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]
- 18436         [NGKI1062] [NGKI1065] [NGKI1070]
- 18437
- 18438     • データキュー管理領域の先頭番地に対する制約 [NGKI1687]
- 18439
- 18440     • 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]
- 18441
- 18442     • 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]
- 18443         [NGKI3324]
- 18444
- 18445     • 生成できるスピンロックの数の上限 [M] [NGKI2142]
- 18446
- 18447     • 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、
- 18448         どのプロセッサが最初にロックを取得できるか [M] [NGKI2183]
- 18449
- 18450     • 固定長メモリプール領域の先頭番地に対する制約 [NGKI2249]

- 18451
- 18452     • 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]
- 18453
- 18454     • タイムティックの周期 [NGKI2335]
- 18455
- 18456     • 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]
- 18457         [NGKI2346]
- 18458
- 18459     • get\_utmがサポートされているか [NGKI2360]
- 18460
- 18461     • オーバランハンドラ機能がサポートされているか [NGKI2598]
- 18462
- 18463     • 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]
- 18464
- 18465     • ターゲット定義のメモリリージョン属性 [P]
- 18466
- 18467     • 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]
- 18468
- 18469     • 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
- 18470         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]
- 18471
- 18472     • 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
- 18473         [P] [NGKI2789]
- 18474
- 18475     • TA\_NOWRITEを指定した場合に, TA\_SDATAが無視されるか [P] [NGKI2790]
- 18476
- 18477     • 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
- 18478         視されるか [P] [NGKI2792]
- 18479
- 18480     • 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
- 18481         扱うことができないメモリオブジェクト [P] [NGKI2793]
- 18482
- 18483     • ターゲット定義のメモリオブジェクト属性 [P] [NGKI2794]
- 18484
- 18485     • 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
- 18486         に対する制限 [P] [NGKI2831]
- 18487
- 18488     • ATT\_MOD/ATA\_MODがサポートされているか [P] [NGKI2859]
- 18489
- 18490     • 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
- 18491         定されるメモリオブジェクト属性 [P] [NGKI2850]
- 18492
- 18493     • 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
- 18494         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
- 18495         配置されるセクション [PM] [NGKI3271]
- 18496
- 18497     • 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
- 18498         録できる数に対する制限 [P] [NGKI2857]
- 18499
- 18500     • 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

- 18501       ンや登録できる数に対する制限 [P] [NGKI2878]  
18502  
18503       • 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  
18504       対する制約 [P] [NGKI2880]  
18505  
18506       • ATT\_PMA/ATA\_PMA/att\_pmaがサポートされているか [P] [NGKI2903]  
18507       [HRPS0156]  
18508  
18509       • 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  
18510       ンや登録できる数に対する制限 [P] [NGKI2898]  
18511  
18512       • 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、  
18513       物理アドレス空間における先頭番地に対する制約 [P] [NGKI2900]  
18514  
18515       • ターゲット定義の割込み要求ライン属性 [NGKI2945]  
18516  
18517       • 割込みハンドラ属性にTA\_NONKERNELを指定できるか [NGKI2957]  
18518  
18519       • その他のターゲット定義の割込みハンドラ属性 [NGKI2959]  
18520  
18521       • 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  
18522       れるか [D] [NGKI2980]  
18523  
18524       • 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  
18525       設定できるか [NGKI2982]  
18526  
18527       • 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  
18528       イン属性/割込み優先度に対する制限 [NGKI2986]  
18529  
18530       • 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]  
18531       [NGKI3018]  
18532  
18533       • 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  
18534       るか [NGKI3020]  
18535  
18536       • 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  
18537       定義できるか [NGKI3064]  
18538  
18539       • 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]  
18540  
18541       • 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]  
18542  
18543       • 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]  
18544  
18545       • 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]  
18546       [NGKI3077]  
18547  
18548       • 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  
18549       みハンドラ番号 [NGKI3078]  
18550

- 18551     • 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
- 18552         は定義解除）できない割込みハンドラ [M] [NGKI3079]
- 18553
- 18554     • 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
- 18555         [NGKI3080]
- 18556
- 18557     • dis\_intがサポートされているか [NGKI3091]
- 18558
- 18559     • 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
- 18560         をセットできないか [NGKI3087]
- 18561
- 18562     • dis\_intにおいて、割込み要求禁止フラグの振舞いが、この仕様の規定と異なる
- 18563         か [NGKI3089]
- 18564
- 18565     • ena\_intがサポートされているか [NGKI3104]
- 18566
- 18567     • 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
- 18568         をクリアできないか [NGKI3100]
- 18569
- 18570     • ena\_intにおいて、割込み要求禁止フラグの振舞いが、この仕様の規定と異なる
- 18571         か [NGKI3102]
- 18572
- 18573     • 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
- 18574         きるか [NGKI3114]
- 18575
- 18576     • ターゲット定義のCPU例外ハンドラ属性 [NGKI3123]
- 18577
- 18578     • 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]
- 18579         [NGKI3148]
- 18580
- 18581     • 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
- 18582         [NGKI3149]
- 18583
- 18584     • 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]
- 18585
- 18586     • 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
- 18587         [NGKI3220] [NGKI3222]
- 18588
- 18589     • 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
- 18590         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]
- 18591
- 18592     • 共有スタック領域のサイズの最小値 [NGKI3255]
- 18593
- 18594     • 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]
- 18595
- 18596     • 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
- 18597         [NGKI3246]
- 18598
- 18599     • 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
- 18600         [NGKI3253]

18601		
18602	5.10	省略名の元になった英語
18603		
18604	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
18605		
18606	xxx	元になった英語
18607	-----	
18608	act	activate
18609	aid	automatically assigned ID
18610	ata	attach with access control vector
18611	att	attach
18612	cal	call
18613	can	cancel
18614	cfg	configure
18615	chg	change
18616	clr	clear
18617	cre	create
18618	def	define
18619	del	delete
18620	det	detach
18621	dis	disable
18622	dly	delay
18623	ena	enable
18624	epr	execution priority
18625	ext	exit
18626	get	get
18627	ini	initialize
18628	lmt	limit
18629	lnk	link
18630	loc	lock
18631	mig	migrate
18632	pol	poll
18633	prb	probe
18634	ras	raise
18635	rcv	receive
18636	ref	reference
18637	rel	release
18638	rot	rotate
18639	rsm	resume
18640	sac	set access control vector
18641	set	set
18642	sig	signal
18643	slp	sleep
18644	snd	send
18645	sns	sense
18646	sta	start
18647	stp	stop
18648	sus	suspend
18649	ter	terminate
18650	try	try

18651        unl        unlock  
18652        wai        wait  
18653        wup        wake up

18654

18655        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

18656

18657        yyy        元になった英語

18658

	yyy	元になった英語
	-----	
18659	act	activation
18660	alm	alarm handler
18661	cfg	configuration
18662	cpu	CPU
18663	ctx	context
18664	cyc	cyclic handler
18665	did	domain ID
18666	dom	domain
18667	dpn	dispatch pending
18668	dsp	dispatch
18669	dtq	data queue
18670	exc	exception
18671	flg	eventflag
18672	ics	interrupt context stack
18673	inf	information
18674	inh	interrupt handler
18675	ini	initilization
18676	int	interrupt
18677	ipm	interrupt priority mask
18678	isr	interrupt service routine
18679	ker	kernel
18680	loc	lock
18681	mbf	message buffer
18682	mbx	mailbox
18683	mpf	fixed-sized memory pool
18684	mem	memory
18685	mod	module
18686	mtx	mutex
18687	ovr	overflow handler
18688	pdq	priority data queue
18689	pid	processor ID
18690	pma	physical memory area
18691	pri	priority
18692	rdq	ready queue
18693	reg	region
18694	sec	section
18695	sem	semaphore
18696	srg	standard memory region
18697	spn	spin lock
18698	stk	stack
18699	sys	system
18700	svc	service call

18701	ter	termination
18702	tex	task exception
18703	tid	task ID
18704	tim	time
18705	tsk	task
18706	utm	time in micro second
18707	ver	version
18708	wai	wait
18709	wup	wake up
18710	xpn	exception pending

18711

18712 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

18713

18714	z	元になった英語
-------	---	---------

18715

18716	a	automatic ID assignment
18717	f	force
18718	i	interrupt
18719	m	multiprocessor
18720	p	poll
18721	t	timeout
18722	x	exception

18723

18724 5.11 バージョン履歴

18725

18726	2008年11月19日	Release 1.0.0	最初のリリース
18727	2009年5月8日	Release 1.1.0	FMPカーネルに関する記述が完成
18728	2010年5月10日	Release 1.2.0	
18729	2011年5月5日	Release 1.3.0	HRP2カーネルに関する記述が完成
18730	2012年5月16日	Release 1.4.0	SSPカーネルに関する記述が完成
18731	2012年12月19日	Release 1.5.0	HRP2カーネルの仕様変更を反映
18732	2014年1月16日	Release 1.6.0	
18733	2014年11月17日	Release 1.7.0	

18734

18735 以上

## アプリケーションシステム

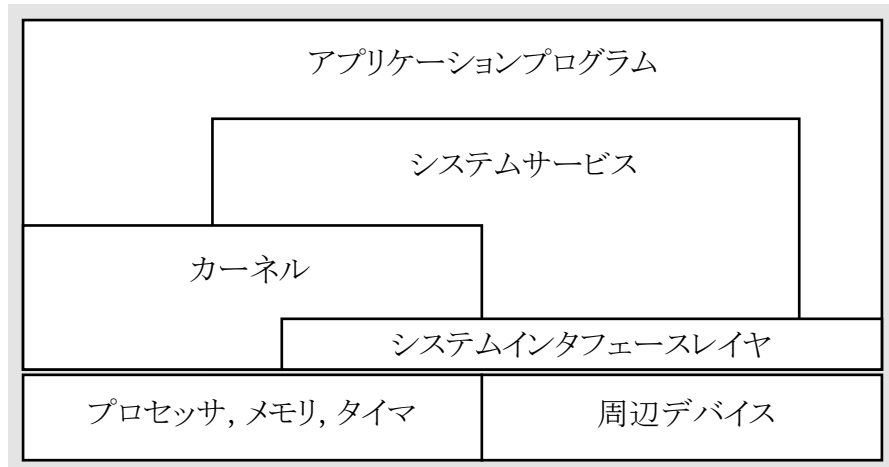


図2-1. 想定するソフトウェア構成





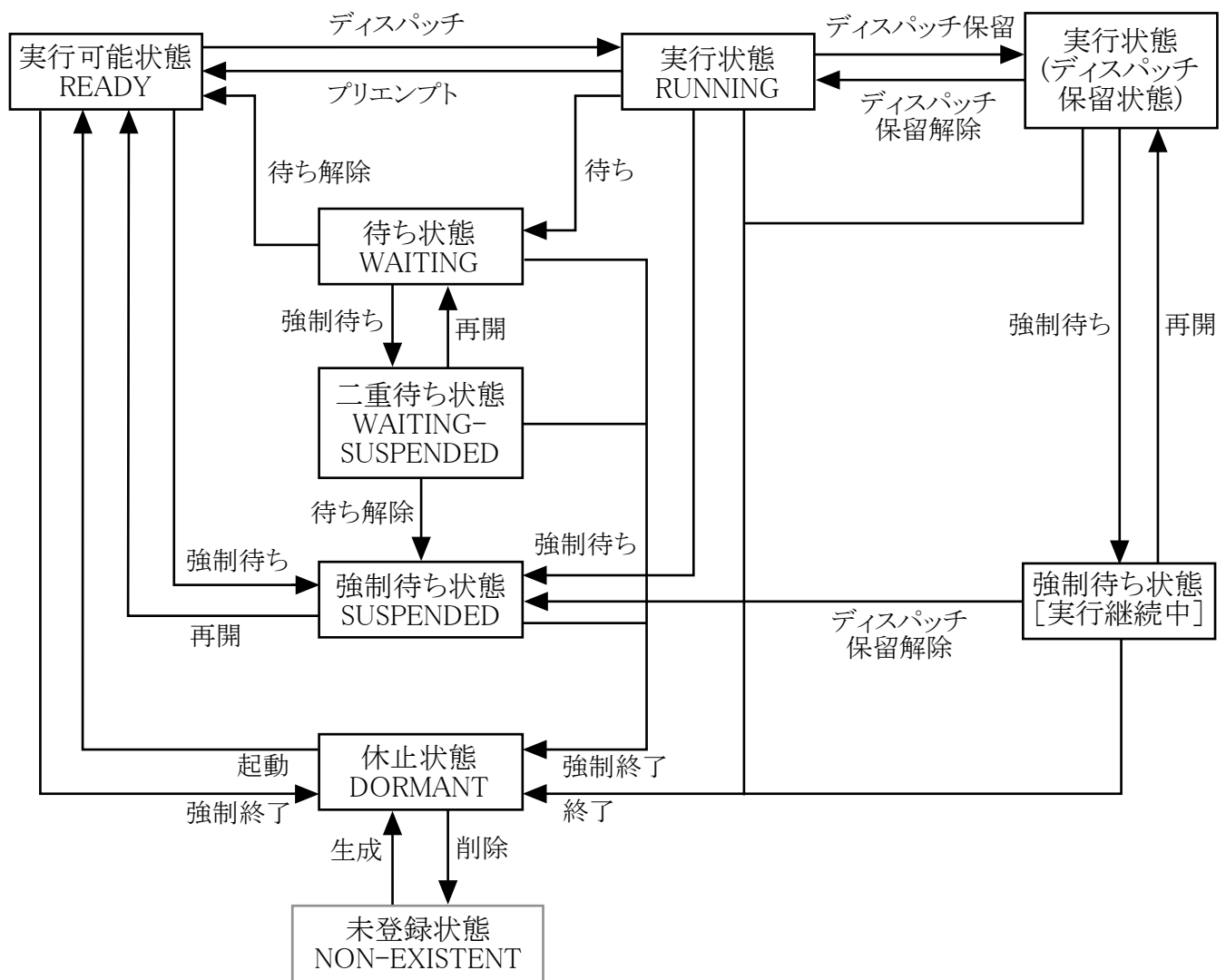


図2-3. 過渡的な状態も含めたタスクの状態遷移

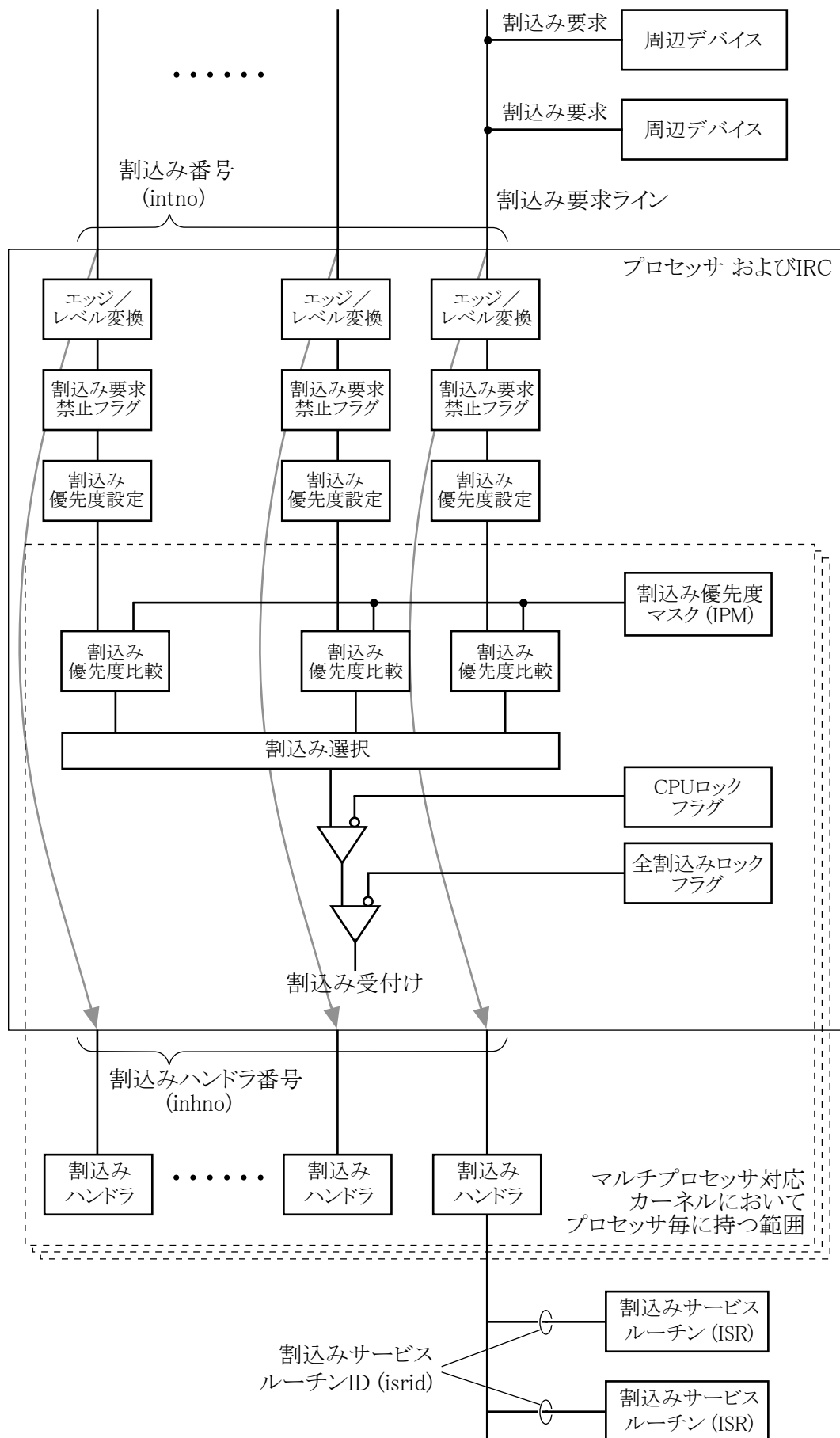


図2-4. TOPPERS標準割り込み処理モデルの概念図

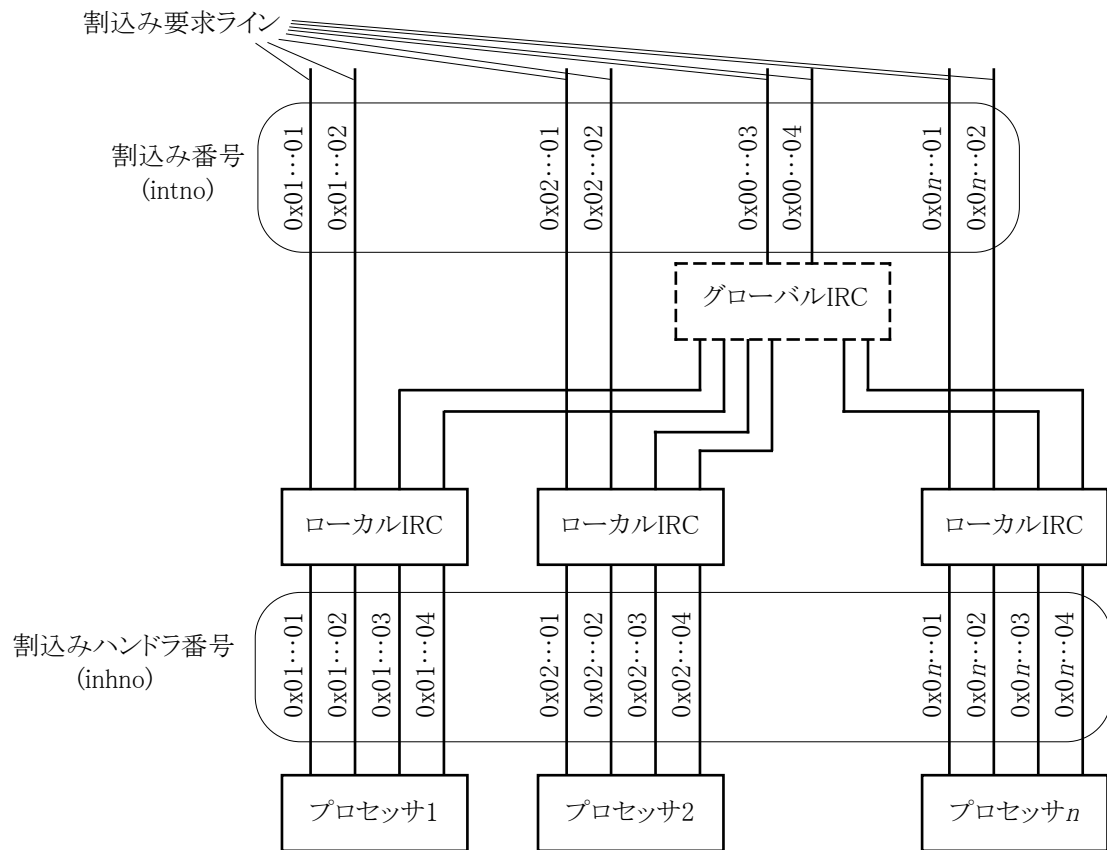


図2-5. マルチプロセッサ対応カーネルにおける割り込み番号と割り込みハンドラ番号

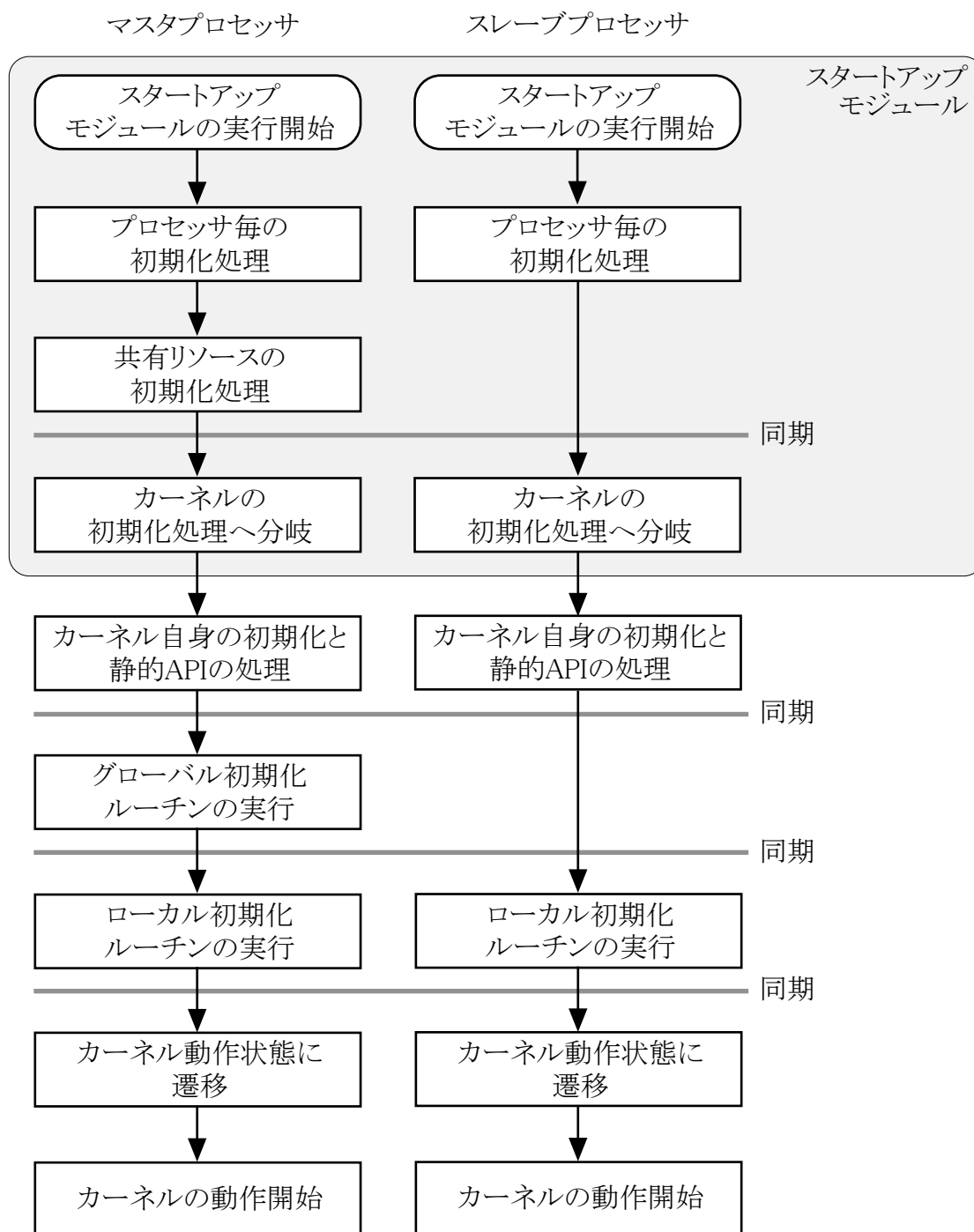


図2-6. マルチプロセッサ対応カーネルにおけるシステム初期化の流れ

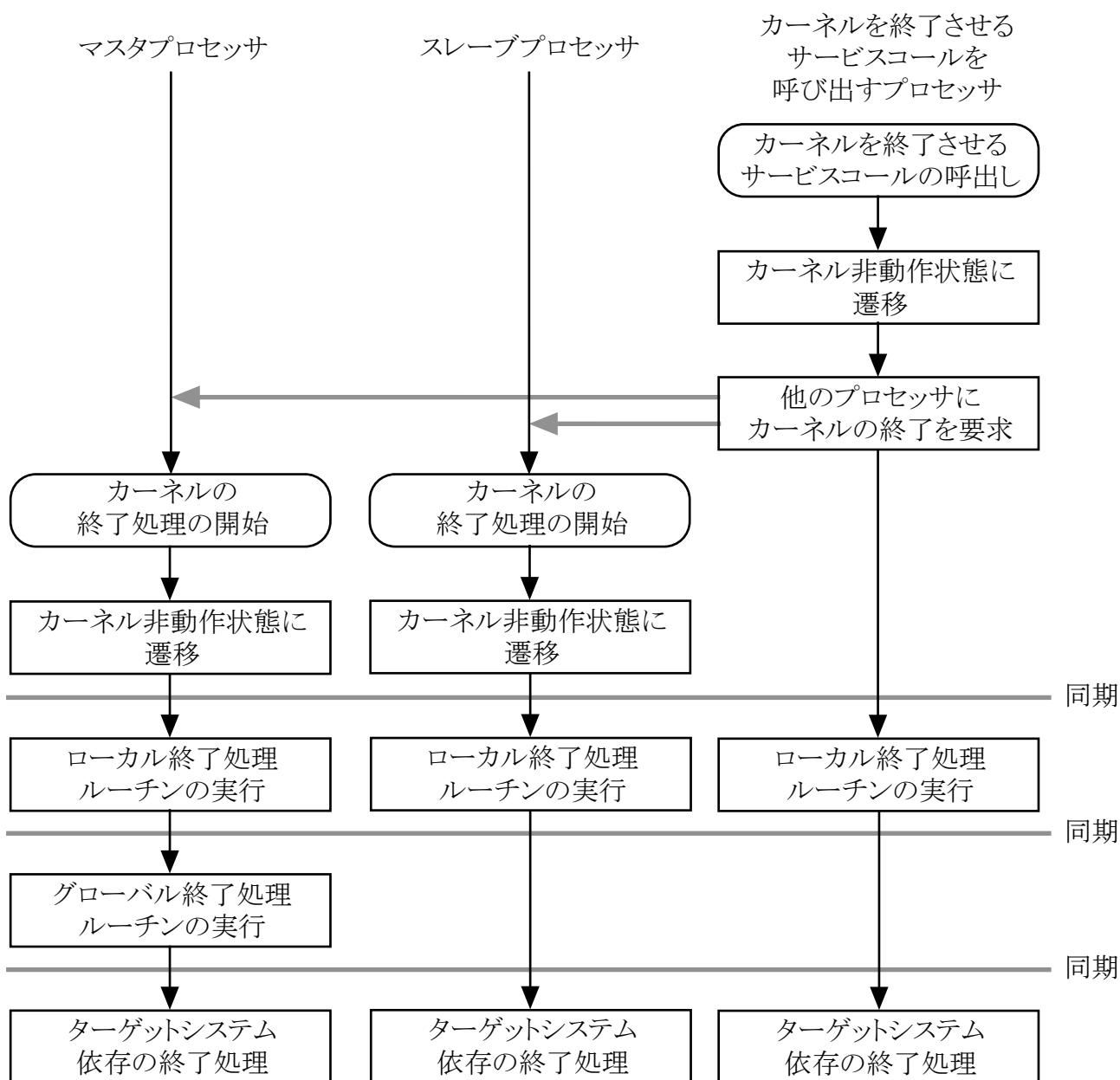


図2-7. マルチプロセッサ対応カーネルにおけるシステム終了処理の流れ

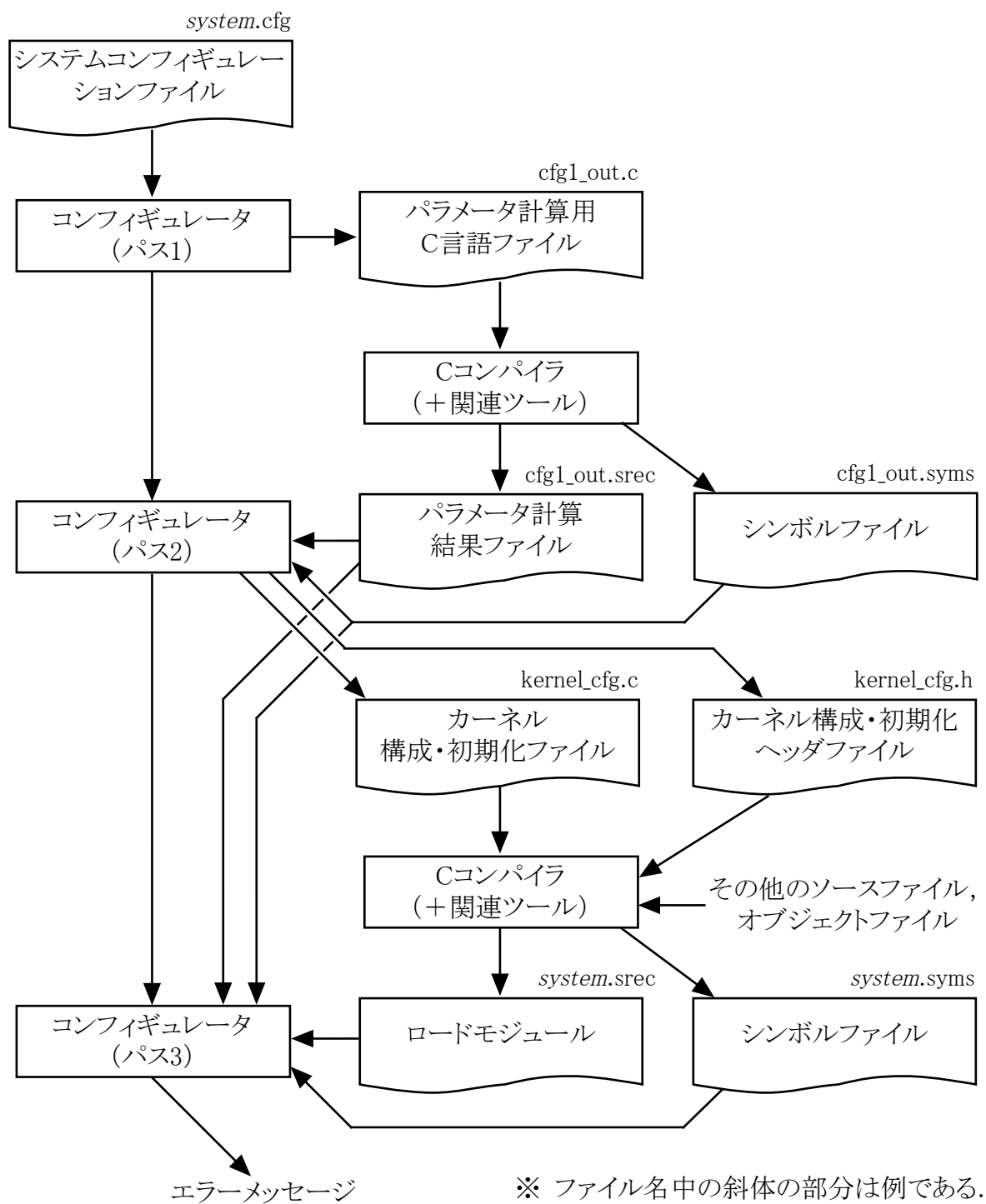
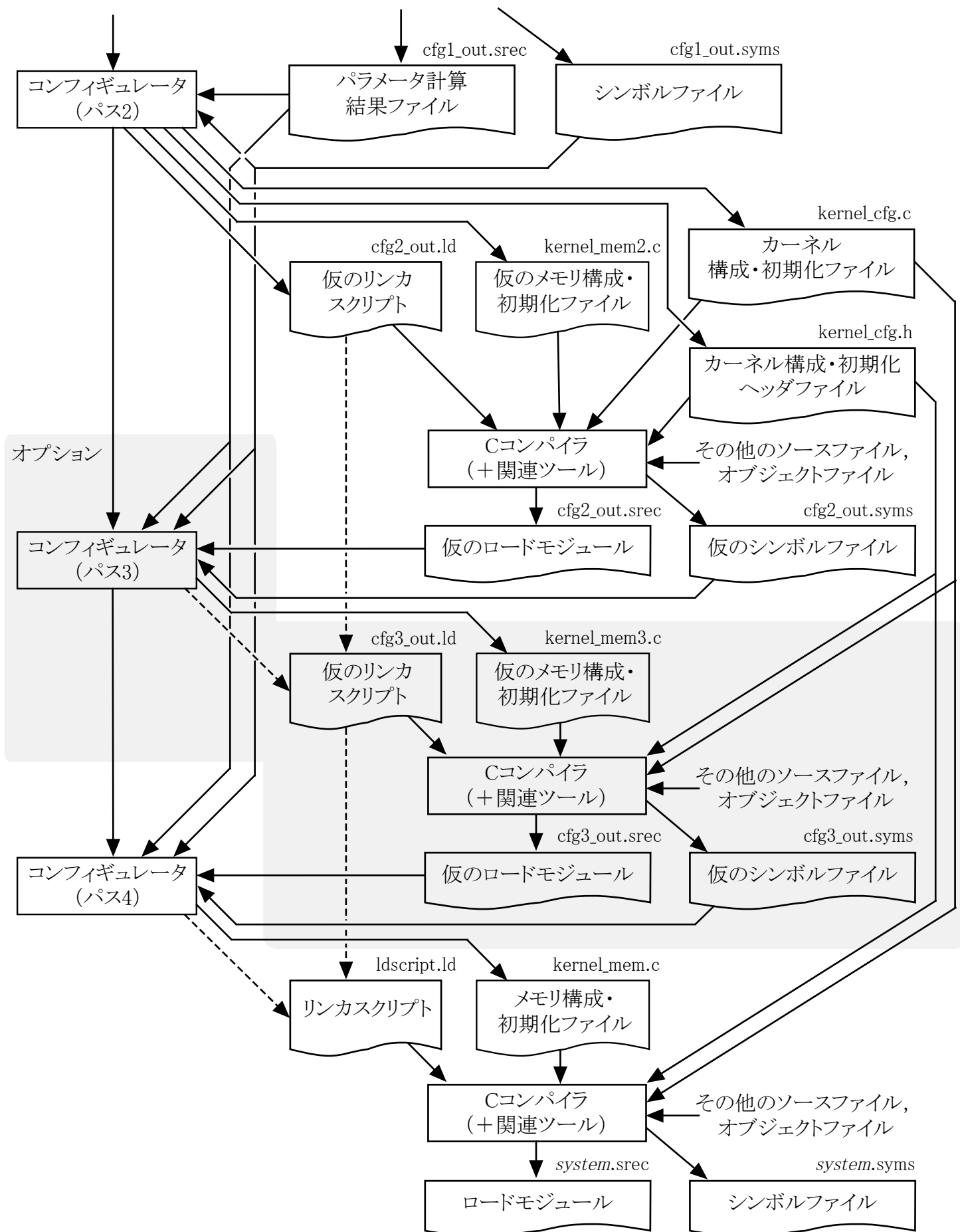


図2-8. コンフィギュレータの処理モデル



※ 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