

TOPPERS新世代カーネル統合仕様書

バージョン: Release 1.5.0
最終更新: 2012年12月19日

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

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 1.7.0, 1.8.0
TOPPERS/FMPカーネル	Release 1.2.0
TOPPERS/HRP2カーネル	Release 2.1.0
TOPPERS/SSPカーネル	Release 1.2.0

TOPPERS New Generation Kernel Specification

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

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

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

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

○目次

51	
52	・ 目次
53	・ 仕様書で用いる記述項目と記号
54	・ タグの付与方法
55	
56	第 1 章 TOPPERS新世代カーネルの概要
57	
58	1. 1 TOPPERS新世代カーネル仕様の位置付け
59	1. 2 TOPPERS新世代カーネル仕様の設計方針
60	1. 3 TOPPERS/ASPカーネルの適用対象領域と仕様設計方針
61	1. 4 TOPPERS/FMPカーネルの適用対象領域と仕様設計方針
62	1. 5 TOPPERS/HRP2カーネルの適用対象領域と仕様設計方針
63	1. 6 TOPPERS/SSPカーネルの適用対象領域と仕様設計方針
64	1. 7 TOPPERS/ASP Safetyカーネルの適用対象領域と仕様設計方針
65	
66	第 2 章 主要な概念と共通定義
67	
68	2. 1 仕様の位置付け
69	2. 1. 1 カーネルの機能セット
70	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
71	2. 1. 3 想定するソフトウェア構成
72	2. 1. 4 想定するハードウェア構成
73	2. 1. 5 想定するプログラミング言語
74	2. 2 APIの構成要素とコンベンション
75	2. 2. 1 APIの構成要素
76	2. 2. 2 パラメータとリターンパラメータ
77	2. 2. 3 返値とエラーコード
78	2. 2. 4 機能コード
79	2. 2. 5 ヘッダファイル
80	2. 3 主な概念
81	2. 3. 1 オブジェクトと処理単位
82	2. 3. 2 サービスコールとパラメータ
83	2. 3. 3 保護機能
84	2. 3. 4 マルチプロセッサ対応
85	2. 3. 5 その他
86	2. 4 処理単位の種類と実行
87	2. 4. 1 処理単位の種類
88	2. 4. 2 処理単位の実行順序
89	2. 4. 3 カーネル処理の不可分性
90	2. 4. 4 処理単位を実行するプロセッサ
91	2. 5 システム状態とコンテキスト
92	2. 5. 1 カーネル動作状態と非動作状態
93	2. 5. 2 タスクコンテキストと非タスクコンテキスト
94	2. 5. 3 カーネルの振舞いに影響を与える状態
95	2. 5. 4 全割込みロック状態と全割込みロック解除状態
96	2. 5. 5 CPUロック状態とCPUロック解除状態
97	2. 5. 6 割込み優先度マスク
98	2. 5. 7 ディスパッチ禁止状態とディスパッチ許可状態
99	2. 5. 8 ディスパッチ保留状態
100	2. 5. 9 カーネル管理外の状態

101	2.5.10 処理単位の開始・終了とシステム状態
102	2.6 タスクの状態遷移とスケジューリング規則
103	2.6.1 基本的なタスク状態
104	2.6.2 タスクの状態遷移
105	2.6.3 タスクのスケジューリング規則
106	2.6.4 待ち行列と待ち解除の順序
107	2.6.5 タスク例外処理マスク状態と待ち禁止状態
108	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
109	2.6.7 制約タスク
110	2.7 割込み処理モデル
111	2.7.1 割込み処理の流れ
112	2.7.2 割込み優先度
113	2.7.3 割込み要求ラインの属性
114	2.7.4 割込みを受け付ける条件
115	2.7.5 割込み番号と割込みハンドラ番号
116	2.7.6 マルチプロセッサにおける割込み処理
117	2.7.7 カーネル管理外の割込み
118	2.7.8 カーネル管理外の割込みの設定方法
119	2.8 CPU例外処理モデル
120	2.8.1 CPU例外処理の流れ
121	2.8.2 CPU例外ハンドラから呼び出せるサービスコール
122	2.8.3 エミュレートされたCPU例外ハンドラ
123	2.8.4 カーネル管理外のCPU例外
124	2.9 システムの初期化と終了
125	2.9.1 システム初期化手順
126	2.9.2 システム終了手順
127	2.10 オブジェクトの登録とその解除
128	2.10.1 ID番号で識別するオブジェクト
129	2.10.2 オブジェクト番号で識別するオブジェクト
130	2.10.3 識別番号を持たないオブジェクト
131	2.10.4 オブジェクト生成に必要なメモリ領域
132	2.10.5 オブジェクトが属する保護ドメインの設定
133	2.10.6 オブジェクトが属するクラスの設定
134	2.10.7 オブジェクトの状態参照
135	2.11 オブジェクトのアクセス保護
136	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
137	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
138	2.11.3 デフォルトのアクセス許可ベクタ
139	2.11.4 アクセス許可ベクタの設定
140	2.11.5 カーネルの管理領域のアクセス保護
141	2.11.6 ユーザタスクのユーザスタック領域
142	2.12 システムコンフィギュレーション手順
143	2.12.1 システムコンフィギュレーションファイル
144	2.12.2 静的APIの文法とパラメータ
145	2.12.3 保護ドメインの指定
146	2.12.4 クラスの指定
147	2.12.5 コンフィギュレータの処理モデル
148	2.12.6 静的APIのパラメータに関するエラー検出
149	2.12.7 オブジェクトのID番号の指定
150	2.13 TOPPERSネーミングコンベンション

151	2.13.1 モジュール識別名
152	2.13.2 データ型名
153	2.13.3 関数名
154	2.13.4 変数名
155	2.13.5 定数名
156	2.13.6 マクロ名
157	2.13.7 静的API名
158	2.13.8 ファイル名
159	2.13.9 モジュール内部の名称の衝突回避
160	2.14 TOPPERS共通定義
161	2.14.1 TOPPERS共通ヘッダファイル
162	2.14.2 TOPPERS共通データ型
163	2.14.3 TOPPERS共通定数
164	2.14.4 TOPPERS共通エラーコード
165	2.14.5 TOPPERS共通マクロ
166	2.14.6 TOPPERS共通構成マクロ
167	2.15 カーネル共通定義
168	2.15.1 カーネルヘッダファイル
169	2.15.2 カーネル共通定数
170	2.15.3 カーネル共通マクロ
171	2.15.4 カーネル共通構成マクロ
172	
173	第3章 システムインタフェースレイヤAPI仕様
174	
175	3.1 システムインタフェースレイヤの概要
176	3.2 SILヘッダファイル
177	3.3 全割込みロック状態の制御
178	3.4 SILスピンロック
179	3.5 微少時間待ち
180	3.6 エンディアンの取得
181	3.7 メモリ空間アクセス関数
182	3.8 I/O空間アクセス関数
183	3.9 プロセッサIDの参照
184	
185	第4章 カーネルAPI仕様
186	
187	4.1 タスク管理機能
188	4.2 タスク付属同期機能
189	4.3 タスク例外処理機能
190	4.4 同期・通信機能
191	4.4.1 セマフォ
192	4.4.2 イベントフラグ
193	4.4.3 データキュー
194	4.4.4 優先度データキュー
195	4.4.5 メールボックス
196	4.4.6 ミューテックス
197	4.4.7 メッセージバッファ (☆未完成)
198	4.4.8 スピンロック
199	4.5 メモリプール管理機能
200	4.5.1 固定長メモリプール

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

251
252 【～～仕様との関係】の項では、この仕様と、 μ ITRON4.0仕様または
253 μ ITRON4.0/PX仕様との違いについて説明する。
254
255 【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
256 に残っている事項について記述する。
257
258 【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
259 て説明する。
260
261 「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
262 においては、以下の記述項目を用いる。
263
264 【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
265 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
266 コールを呼び出す形式を規定する。
267
268 【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
269 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
270 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、
271 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
272 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。
273
274 【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。
275
276 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
277 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。
278
279 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
280 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。
281
282 [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
283 ストから呼び出すと、E_CTXエラーとなる。
284
285 [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
286 ストから呼び出すと、E_CTXエラーとなる。
287
288 [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
289 のできるサービスコールを示す。
290
291 [S] は静的APIを示す。
292
293 [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
294 対応でないカーネルでは、このAPIはサポートされない。
295
296 [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
297 護機能対応カーネルでは、このAPIはサポートされない。
298
299 [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。
300 マルチプロセッサ対応でないカーネルでは、このAPIはサポートされない。

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

また、エラーが発生する条件を表すために、次の記号を用いる。

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

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

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

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

○タグの付与方法

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

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

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

タグの形式と意味は次の通りである（xxxxは4桁の数字を表す）。

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

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

第1章 TOPPERS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、 μ ITRON4.0仕様およびその保護機能拡張（ μ ITRON4.0/PX仕様）に準拠したリアルタイムカーネルを開発してきた。

μ ITRON4.0仕様は1999年に、 μ ITRON4.0/PX仕様は2002年に公表されたが、それ以降現在までの間に、大きな仕様改訂は実施されていない。その間に、組込みシステムおよびソフトウェアのますますの大規模化・複雑化、これまで以上に高い信頼性・安全性に対する要求、小さい消費エネルギー下での高い性能要求など、組込みシステム開発を取り巻く状況は刻々変化している。リアルタイムカーネルに対しても、マルチプロセッサへの対応、発展的な保護機能のサポート、機能安全対応、省エネルギー制御機能のサポートなど、新しい要求が生じている。

TOPPERSプロジェクトでは、リアルタイムカーネルに対するこのような新しい要求に対応するために、 μ ITRON4.0仕様を発展させる形で、TOPPERS新世代カーネル仕様を策定することになった。

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

1.2 TOPPERS新世代カーネル仕様の設計方針

TOPPERS新世代カーネル仕様を設計するにあたり、次の方針を設定する。

(1) μ ITRON4.0仕様をベースに拡張・改良を加える

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

(2) ソフトウェアの再利用性を重視する

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

そこで、TOPPERS新世代カーネル仕様では、 μ ITRON4.0仕様においてオーバーヘッ

401 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
402 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。

403

404 (3) 高信頼・安全なシステム構築を支援する

405

406 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
407 ものとする。

408

409 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
410 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。
411 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
412 己診断のための機能についても、順次取り込んでいく。

413

414 (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

415

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

418

419 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
420 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
421 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
422 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
423 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。

424

425 1.3 TOPPERS/ASPカーネルの適用対象領域と仕様設計方針

426

427 TOPPERS/ASPカーネル（ASPは、Advanced Standard Profileの略。以下、ASPカー
428 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。
429 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
430 を拡張する形で開発する。

431

432 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
433 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
434 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
435 留める。

436

437 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
438 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
439 （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
440 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
441 べきと考えられる。

442

443 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。
444 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
445 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
446 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
447 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
448 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
449 る。

450

451 1.4 TOPPERS/FMPカーネルの適用対象領域と仕様設計方針

452

453 TOPPERS/FMPカーネル (FMPは、Flexible Multiprocessor Profileの略。以下、
454 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
455 イムカーネルである。

456

457 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
458 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
459 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
460 リコードを実行できることが必要である。

461

462 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
463 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
464 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
465 で動的な負荷分散を実現することが可能である。

466

467 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
468 が不要な範囲に留める。

469

470 1.5 TOPPERS/HRP2カーネルの適用対象領域と仕様設計方針

471

472 TOPPERS/HRP2カーネル (HRPは、High Reliable system Profileの略。2はバー
473 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
474 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
475 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。

476

477 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
478 権モードを備え、メモリ保護のためにMMU (Memory Management Unit) または
479 MPU (Memory Protection Unit) を持つプロセッサを用いたシステムである。
480 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
481 ズ (バイナリコード) が数百KB以上のシステムである。

482

483 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
484 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
485 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
486 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。

487

488 1.6 TOPPERS/SSPカーネルの適用対象領域と仕様設計方針

489

490 TOPPERS/SSPカーネル (SSPは、Smallest Set Profileの略。以下、SSPカーネル)
491 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
492 絞り込んだリアルタイムカーネルである。

493

494 SSPカーネルの機能は、 μ ITRON4.0仕様の「仕様準拠の最低条件」の考え方を踏
495 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
496 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
497 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
498 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
499 に有用な機能は、ASPカーネルに対して追加している。

500

501 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）
502 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。

504 1.7 TOPPERS/ASP Safetyカーネルの適用対象領域と仕様設計方針

506 TOPPERS/ASP Safetyカーネル（以下、ASP Safetyカーネル）は、小規模な安全
507 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
508 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
509 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
510 のかかる機能はサポートしない。

512 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
513 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
514 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
515 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。

518 第2章 主要な概念と共通定義

520 2.1 仕様の位置付け

522 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
523 述することを目標としている。また、TOPPERS新世代カーネル上で動作する各種
524 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。

526 2.1.1 カーネルの機能セット

528 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
529 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
530 カーネルで構成される。

532 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
533 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
534 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
535 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
536 正するのは得策ではない。

538 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
539 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
540 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
541 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
542 ネルを動的生成対応カーネルと呼ぶことにする。

544 【TOPPERS/ASPカーネルにおける規定】

546 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
547 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
548 パッケージを用いると、動的生成対応カーネルとなる【ASPS0002】。

550 【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】。

【TOPPERS/SSPカーネルにおける規定】

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

【 μ ITRON4.0仕様、 μ ITRON4.0/PX仕様との関係】

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

2.1.2 ターゲット非依存の規定とターゲット定義の規定

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

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

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

2.1.3 想定するソフトウェア構成

この仕様では、アプリケーションシステムを構成するソフトウェアを、アプリケーションプログラム (以下、単にアプリケーションと呼ぶ)、システムサー

ビス, カーネルの3階層に分けて考える (図2-1) . カーネルとシステムサービスをあわせて, ソフトウェアプラットフォームと呼ぶ.

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

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

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

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

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

【 μ ITRON4.0仕様との関係】

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

2.1.4 想定するハードウェア構成

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

(a) メモリ番地は, 常に同一のメモリを指すこと (オーバーレイのように, 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, フリースタンドィ

651 ング環境で用いることを想定して規定している【NGKI0004】。

652

653 ただし、C90の規定に加えて、以下のことを仮定している。

654

655 ・16ビットおよび32ビットの整数型があること【NGKI0005】

656 ・ポインタが格納できるサイズの整数型があること【NGKI0006】

657

658 2.2 APIの構成要素とコンベンション

659

660 2.2.1 APIの構成要素

661

662 (1) サービスコール

663

664 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
665 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、
666 システムコール (system call) と呼ぶ場合もある。

667

668 (2) コールバック

669

670 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
671 スをコールバック (callback) と呼ぶ。

672

673 (3) 静的API

674

675 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
676 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
677 API) と呼ぶ。

678

679 (4) 構成マクロ

680

681 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
682 フトウェアが用いるマクロを、構成マクロ (configuration macro) と呼ぶ。

683

684 2.2.2 パラメータとリターンパラメータ

685

686 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
687 らが返すデータをリターンパラメータ (return parameter) と呼ぶ。また、静
688 的APIに渡すデータもパラメータと呼ぶ。

689

690 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
691 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
692 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

693 【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
694 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】。

695

696 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
697 タを入れる領域へのポインタをパラメータとして渡すことで実現する

698 【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
699 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
700 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

701 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】。

702

703 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
704 ト (packet) と呼ぶ。

705

706 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
707 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、
708 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
709 れることはなく、別の目的に使用できる【NGKI0011】。

710

711 2.2.3 返値とエラーコード

712

713 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
714 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
715 (=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
716 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
717 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
718 を、エラーコード (error code) と呼ぶ。

719

720 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
721 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
722 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
723 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
724 意されている【NGKI0015】。

725

726 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
727 に定める（「2.14.4 TOPPERS共通エラーコード」の節を参照）【NGKI0016】。
728 サービスコールおよびコールバックの機能説明中の「E_XXXXXエラーとなる」ま
729 たは「E_XXXXXエラーが返る」という記述は、メインエラーコードとして
730 E_XXXXXが返ることを意味する。

731

732 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
733 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
734 【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
735 用するシステムサービスのAPI仕様において規定する【NGKI0018】。

736

737 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
738 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
739 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
740 ルの機能説明にその旨を記述する【NGKI0020】。

741

742 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
743 のエラーを示すエラーコードが返る【NGKI0021】。

744

745 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
746 エラーを示すエラーコードを返せばよい【NGKI0022】。

747

748 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
749 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
750 「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。

751
752 2.2.4 機能コード
753
754 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
755 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。
756 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
757 割り付け、拡張サービスコールには正の値を用いる【NGKI0023】。
758
759 2.2.5 ヘッダファイル
760
761 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。
762
763 ヘッダファイルは、原則として、複数回インクルードしてもエラーにならない
764 ように対処されている。具体的には、ヘッダファイルの先頭で特定の識別子
765 (例えば、kernel.hなら"TOPPERS_KERNEL_H") がマクロ定義され、ヘッダフ
766 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
767 レクティブが付加されている【NGKI0024】。
768
769 2.3 主な概念
770
771 2.3.1 オブジェクトと処理単位
772
773 (1) オブジェクト
774
775 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
776 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
777 源を、カーネルオブジェクト (kernel object) と呼ぶ。
778
779 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
780 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
781 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
782 【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
783 号 (ID number) と呼ぶ。そうでない場合、すなわちカーネルまたはシステムサー
784 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
785 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
786 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】。
787
788 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
789 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
790 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
791 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
792 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
793 【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
794 指定する【NGKI0030】。
795
796 (2) 処理単位
797
798 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
799 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
800 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、

801 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。
802
803 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
804 ることを実行開始 (start) と呼ぶ。

805
806 拡張情報 (extended information) は、処理単位が呼び出される時にパラメー
807 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
808 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】。

809 (3) タスク

810
811
812 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)
813 と呼ぶ。タスクは、処理単位の1つである。

814
815 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、
816 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
817 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
818 る。

819
820 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。
821 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
822 告する【NGKI0033】。

823 【補足説明】

824
825
826 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
827 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
828 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
829 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」
830 は、異なるものを指す。

831 (4) ディスパッチとスケジューリング

832
833
834 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
835 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
836 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
837 (scheduling) と呼ぶ。

838
839 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
840 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
841 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
842 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
843 れた時点で、ディスパッチが起こる【NGKI0034】。

844 (5) 割込みとCPU例外

845
846
847 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
848 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
849 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
850 (CPU exception) と呼ぶ。

851
852 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
853 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)
854 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、
855 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
856 【NGKI0035】。
857
858 マスクすることができない割込みを、NMI (non-maskable interrupt) と呼ぶ。
859
860 【 μ ITRON4.0仕様との関係】
861
862 μ ITRON4.0仕様において、未定義のまま使われていた割込みとCPU例外という用
863 語を定義した。
864
865 (6) タイムイベントとタイムイベントハンドラ
866
867 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
868 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、
869 タイムイベントハンドラ (time event handler) と呼ぶ。
870
871 2.3.2 サービスコールとパラメータ
872
873 (1) 優先順位と優先度
874
875 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
876 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
877 高い処理単位が実行される【NGKI0036】。
878
879 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
880 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
881 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
882 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
883 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
884 【NGKI0038】。
885
886 (2) システム時刻と相対時間
887
888 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
889 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
890 【NGKI0039】。システム時刻は、タイムティック (time tick) を通知するた
891 めのタイマ割込みが発生する毎に更新される【NGKI0040】。
892
893 イベントが発生させる時刻を指定する場合には、基準時刻 (base time) からの
894 相対時間 (relative time) によって指定する【NGKI0041】。基準時刻は、別に
895 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
896 る【NGKI0042】。
897
898 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
899 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
900 トの符号無しの整数型 (uint16_t型) に格納できる任意の値を指定することが

901 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
902 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
903 TMAX_RELTIMに定義されている【NGKI0045】。

904

905 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
906 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
907 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
908 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
909 (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
910 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
911 されない【NGKI0047】。

912

913 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
914 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
915 した時刻となる【NGKI0049】。

916

917 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
918 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
919 【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
920 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
921 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、
922 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
923 【NGKI0051】。

924

925 **【補足説明】**

926

927 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
928 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
929 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
930 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
931 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
932 ある。

933

934 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
935 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
936 降のタイムティックでイベントの処理が行われる。

937

938 **【μITRON4.0仕様との関係】**

939

940 相対時間 (RELTIM型) とシステム時刻 (SYSTIM型) の時間単位は、μITRON4.0
941 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
942 時間の解釈について、より厳密に規定した。

943

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

945

946 (3) タイムアウトとポーリング

947

948 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
949 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
950 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

951 エラーが返る【NGKI0052】。

952

953 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

954 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

955 【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

956 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

957 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

958 した後となる。

959

960 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

961 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

962 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

963 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

964 敗したサービスコールからは、E_TMOUTエラーが返る【NGKI0055】。

965

966 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

967 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

968 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

969 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

970 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

971 舞いは、該当するサービスコール毎に規定する【NGKI0058】。

972

973 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

974 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）

975 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】。

976

977 【補足説明】

978

979 [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

980 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。

981 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

982 副作用があるかをサービスコール毎に規定する。

983

984 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

985 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

986 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

987 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

988 ビスコールと全く同じ振舞いをする。

989

990 【μITRON4.0仕様との関係】

991

992 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

993 いたが、この仕様ではミリ秒と規定した。

994

995 【仕様決定の理由】

996

997 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

998 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

999 時間をTMO_POLとして呼び出すとエラーになるのは、ディスパッチ保留状態では、

1000 別に規定がない限り、自タスクを広義の待ち状態に遷移させる可能性のあるサー

1001 ビスコール（タイムアウト付きのサービスコールはこれに該当）を呼び出すこ
1002 とはできないと規定されているためである。

1003

1004 (4) ノンブロッキング

1005

1006 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
1007 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
1008 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
1009 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
1010 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
1011 則は適用されない【NGKI0061】。

1012

1013 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
1014 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
1015 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
1016 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
1017 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
1018 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】。

1019

1020 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
1021 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
1022 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】。

1023

1024 【補足説明】

1025

1026 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
1027 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
1028 していない。

1029

1030 2.3.3 保護機能

1031

1032 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、
1033 保護機能対応カーネルにのみ適用される。

1034

1035 (1) アクセス保護

1036

1037 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
1038 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
1039 を防ぐアクセス保護機能を提供する【NGKI0066】。

1040

1041 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
1042 が対象（object）ということになる。

1043

1044 (2) メモリオブジェクト

1045

1046 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
1047 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
1048 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。
1049 メモリオブジェクトは、互いに重なりあうことはない【NGKI0068】。

1050

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

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

(3) 保護ドメイン

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

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

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

ただし、タスクのユーザスタック領域は、ターゲット定義での変更がない限りは、そのタスク（とカーネルドメインに属する処理単位）のみがアクセスできる（「2.11.6 ユーザタスクのユーザスタック領域」の節を参照）【NGKI0074】。これは、【NGKI0073】の原則の例外となっている。

デフォルトでは、保護ドメインに属するカーネルオブジェクトは、同じ保護ドメイン（とカーネルドメイン）のみからアクセスできる【NGKI0075】。また、無所属のカーネルオブジェクトは、すべての保護ドメインからアクセスできる【NGKI0076】。

(4) カーネルドメインとユーザドメイン

システムには、カーネルドメイン (kernel domain) と呼ばれる保護ドメインが1つ存在する【NGKI0077】。カーネルドメインに属する処理単位は、プロセッサの特権モードで実行される【NGKI0078】。また、すべてのカーネルオブジェクトに対して、すべての種別のアクセスを行うことが許可される【NGKI0079】。この仕様で、「ある保護ドメイン（またはタスク）のみからアクセスできる」といった場合でも、カーネルドメインドメインからはアクセスすることができる。

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

ユーザドメインには、1から連続する正の整数値の保護ドメインIDが付与される

1101 【NGKI0082】. カーネルドメインの保護ドメインIDは, TDOM_KERNEL (= -1) で
1102 ある【NGKI0083】.

1103
1104 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
1105 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
1106 ンフィギュレータがエラーを報告する【NGKI0085】.

1107
1108 【補足説明】

1109
1110 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
1111 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
1112 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
1113 意していない.

1114
1115 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
1116 みなすこともできる.

1117
1118 【μ ITRON4.0/PX仕様との関係】

1119
1120 μ ITRON4.0/PX仕様のシステムドメイン (system domain) は, 現時点ではサポー
1121 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
1122 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
1123 できる保護ドメインである.

1124
1125 (5) システムタスクとユーザタスク

1126
1127 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
1128 メインに属するタスクをユーザタスク (user task) と呼ぶ.

1129
1130 【補足説明】

1131
1132 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
1133 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
1134 ビスコールの実行中は特権モードで実行されるため, 上記の定義とした.

1135
1136 μ ITRON4.0/PX仕様のシステムドメインに属するタスクは, システムタスクと呼
1137 ぶことになる.

1138
1139 (6) アクセス許可パターン

1140
1141 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
1142 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
1143 許可パターン (access permission pattern) と呼ぶ. アクセス許可パターンの
1144 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
1145 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
1146 ビットは用意されていない.

1147
1148 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
1149 (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
1150 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

1151 のビット毎論理和（C言語の“|”）を求めることで、アクセスを許可されている
1152 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
1153 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
1154 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。
1155

1156 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
1157 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）
1158 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
1159 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
1160 の保護ドメインにアクセスを許可することを示すアクセス許可パターンを表す
1161 定数（TACP_SHARED）が用意されている【NGKI0089】。
1162

1163 (7) アクセス許可ベクタ

1164

1165 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、
1166 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
1167 【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
1168 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
1169 （access permission vector）と呼び、次のように定義されるデータ型
1170 （ACVCT）で保持する【NGKI0091】。
1171

```
1172     typedef struct acvct {  
1173         ACPTN    acptn1;    /* 通常操作1のアクセス許可パターン */  
1174         ACPTN    acptn2;    /* 通常操作2のアクセス許可パターン */  
1175         ACPTN    acptn3;    /* 管理操作のアクセス許可パターン */  
1176         ACPTN    acptn4;    /* 参照操作のアクセス許可パターン */  
1177     } ACVCT;
```

1179 【補足説明】

1180

1181 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
1182 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。
1183

1184 【μITRON4.0/PX仕様との関係】

1185

1186 μITRON4.0/PX仕様では、アクセス許可ベクタを、1つまたは2つのアクセス許可
1187 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
1188 めている。
1189

1190 (8) サービスコールの呼出し方法

1191

1192 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
1193 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、
1194 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】。
1195

1196 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
1197 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
1198 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。
1199 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
1200 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。

1201
1202 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
1203 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、
1204 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
1205 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
1206 【NGKI0093】。
1207
1208 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
1209 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
1210 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)
1211 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
1212 【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
1213 ように記述すればよい。
1214
1215 ercd = SVC_CALL(act_tsk)(tskid);
1216
1217 【補足説明】
1218
1219 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。
1220 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
1221 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
1222 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
1223 れない。
1224
1225 2.3.4 マルチプロセッサ対応
1226
1227 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
1228 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。
1229
1230 (1) クラス
1231
1232 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
1233 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
1234 【NGKI0095】。
1235
1236 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
1237 【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
1238 時に決定し、登録後に変更することはできない【NGKI0097】。
1239
1240 【補足説明】
1241
1242 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
1243 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
1244 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
1245 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
1246 けるクラスを（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
1247 が典型的である。
1248
1249 【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
1250 さない。

- 1251
1252 • オーバランハンドラ
1253 • 拡張サービスコール
1254 • グローバル初期化ルーチン
1255 • グローバル終了処理ルーチン
1256
1257 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
1258 ラスのみをサポートしているとみなすこともできる。
1259
1260 (2) プロセッサ
1261
1262 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
1263 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
1264 別する【NGKI0098】。
1265
1266 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)
1267 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】。
1268
1269 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
1270 セッサ (master processor) と呼び、システムに1つ存在する【NGKI0100】。ど
1271 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
1272 【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
1273 (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
1274 とスレーブプロセッサの振舞いに違いはない【NGKI0102】。
1275
1276 (3) 処理単位の割付けとマイグレーション
1277
1278 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
1279 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
1280 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
1281 ロセッサを、初期割付けプロセッサと呼ぶ。
1282
1283 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
1284 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
1285 ことを、処理単位のマイグレーション (migration) と呼ぶ。
1286
1287 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
1288 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
1289 とができる【NGKI0105】。
1290
1291 (4) クラスの持つ属性とカーネルオブジェクト
1292
1293 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
1294 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
1295 ブジェクトが属するクラスによって定まる。
1296
1297 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
1298 の通りである【NGKI0106】。
1299
1300 • 初期割付けプロセッサ

- 1301 • 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
1302 サを含む）
- 1303 • 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
1304 クションが配置されるメモリアリージョン（標準メモリアリージョン）
- 1305 • 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
1306 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
1307 場所
- 1308 • その他の管理情報（ロック単位など）

1309

1310 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】．

1311

1312 【仕様決定の理由】

1313

1314 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
1315 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
1316 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
1317 しておく方が良いと考えたためである．

1318

1319 (5) ローカルタイマ方式とグローバルタイマ方式

1320

1321 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
1322 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
1323 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
1324 である【NGKI0108】．

1325

1326 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
1327 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
1328 能は，カーネルでは用意しない．

1329

1330 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
1331 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
1332 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
1333 【NGKI0111】．

1334

1335 【補足説明】

1336

1337 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．

1338

1339 【未決定事項】

1340

1341 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
1342 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，
1343 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．

1344

1345 2.3.5 その他

1346

1347 (1) オブジェクトモジュール

1348

1349 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
1350 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

1351 プロジェクトモジュールである.

1352

1353 (2) メモリリージョン

1354

1355 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
1356 た連続したメモリ領域をメモリリージョン (memory region) と呼ぶ.

1357

1358 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
1359 ンを識別する文字列を、メモリリージョン名と呼ぶ.

1360

1361 【補足説明】

1362

1363 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
1364 囲という一般的な意味で使っている.

1365

1366 (3) 標準のセクション

1367

1368 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
1369 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、
1370 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】.

1371

1372 (4) 保護ドメイン毎の標準セクション

1373

1374 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
1375 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
1376 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
1377 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
1378 for each protection domain). 保護ドメイン毎の標準セクションのセクショ
1379 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
1380 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")
1381 を "_" でつないだものとする【NGKI0116】. 例えば、カーネルドメインの
1382 ".text" セクションのセクション名は, ".text_kernel" とする.

1383

1384 2.4 処理単位の種類と実行順序

1385

1386 2.4.1 処理単位の種類

1387

1388 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】.

1389

1390 (a) タスク

1391 (a.1) タスク例外処理ルーチン

1392 (b) 割込みハンドラ

1393 (b.1) 割込みサービスルーチン

1394 (b.2) タイムイベントハンドラ

1395 (c) CPU例外ハンドラ

1396 (d) 拡張サービスコール

1397 (e) 初期化ルーチン

1398 (f) 終了処理ルーチン

1399

1400 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

1401 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
1402 ある。

1403
1404 【TOPPERS/ASPカーネルにおける規定】
1405

1406 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
1407 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
1408 と、オーバランハンドラ機能を追加することができる【ASPS0004】。

1409
1410 【TOPPERS/FMPカーネルにおける規定】
1411

1412 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
1413 ない【FMPS0002】。

1414
1415 【TOPPERS/SSPカーネルにおける規定】
1416

1417 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
1418 ビスコールをサポートしていない【SSPS0002】。

1419
1420 2.4.2 処理単位の実行順序
1421

1422 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
1423 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
1424 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
1425 る。

1426
1427 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
1428 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
1429 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
1430 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。

1431
1432 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
1433 が、タスクよりも先に実行される【NGKI0120】。

1434
1435 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
1436 【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
1437 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
1438 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
1439 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
1440 先度」の節を参照すること。

1441
1442 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
1443 出す割込みハンドラと同じである【NGKI0123】。

1444
1445 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
1446 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
1447 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
1448 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
1449 あるが、その処理単位よりも先に実行される【NGKI0125】。

1450

1451 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、
1452 それを呼び出した処理単位よりも先に実行される【NGKI0126】。

1453
1454 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
1455 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
1456 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
1457 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
1458 【NGKI0128】。

1459
1460 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
1461 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
1462 【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
1463 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、
1464 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
1465 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
1466 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
1467 終了処理ルーチンが実行される【NGKI0132】。

1468
1469 【仕様決定の理由】

1470
1471 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
1472 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
1473 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
1474 順に記述するのは難しい）。

1475
1476 2.4.3 カーネル処理の不可分性

1477
1478 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
1479 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
1480 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
1481 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、
1482 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
1483 【NGKI0133】。これを、カーネル処理の不可分性という。

1484
1485 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
1486 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
1487 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
1488 の状態が変化する場合に、一部のオブジェクトの状態のみが変化し、残りのオ
1489 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
1490 【NGKI0134】。

1491
1492 【補足説明】

1493
1494 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
1495 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
1496 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
1497 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
1498 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
1499 れた後に、タスクCへのディスパッチが行われる。

1500

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

2.4.4 処理単位を実行するプロセッサ

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

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

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

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

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

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

オーバランハンドラ、拡張サービスコール、グローバル初期化ルーチン、グローバル終了処理ルーチンは、いずれのクラスにも属さない【NGKI0139】。オーバランハンドラは、オーバランを起こしたタスクの割付けプロセッサによって実

1551 行される【NGKI0140】。拡張サービスコールは、それを呼び出した処理単位の
1552 割付けプロセッサによって実行される【NGKI0141】。グローバル初期化ルーチン
1553 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
1554 【NGKI0142】。

1555

1556 2.5 システム状態とコンテキスト

1557

1558 2.5.1 カーネル動作状態と非動作状態

1559

1560 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、
1561 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
1562 （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
1563 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
1564 態かカーネル非動作状態のいずれかの状態を取る【NGKI0143】。

1565

1566 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
1567 る【NGKI0144】。

1568

1569 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
1570 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

1571 【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
1572 場合の動作は、保証されない【NGKI0146】。

1573

1574 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
1575 ネル非動作状態のいずれかの状態を取る【NGKI0147】。

1576

1577 2.5.2 タスクコンテキストと非タスクコンテキスト

1578

1579 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
1580 ど）をコンテキストと呼ぶ。

1581

1582 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
1583 ンテキストと非タスクコンテキストに分類される【NGKI0148】。

1584

1585 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
1586 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
1587 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
1588 ストに分類される【NGKI0150】。

1589

1590 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
1591 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
1592 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
1593 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
1594 【NGKI0152】。

1595

1596 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
1597 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
1598 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
1599 域を用いて実行される【NGKI0154】。

1600

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

2.5.3 カーネルの振舞いに影響を与える状態

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

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

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

2.5.4 全割込みロック状態と全割込みロック解除状態

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

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

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

2.5.5 CPUロック状態とCPUロック解除状態

プロセッサは、カーネル管理の割込み（「2.7.7 カーネル管理外の割込み」の節を参照）をすべてマスクするためのCPUロックフラグを持つ【NGKI0166】。CPUロックフラグがセットされた状態をCPUロック状態、クリアされた状態をCPUロック解除状態と呼ぶ。CPUロック状態では、すべてのカーネル管理の割込みがマスクされ、ディスパッチが保留される【NGKI0167】。

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

- ・システムインタフェースレイヤのAPI
- ・loc_cpu/iloc_cpu, unl_cpu/iunl_cpu
- ・unl_spn/iunl_spn（マルチプロセッサ対応カーネルのみ）

1651 • dis_int, ena_int
1652 • sns_yyy, xsns_yyy (CPU例外ハンドラからのみ)
1653 • get_utm
1654 • ext_tsk, ext_ker
1655 • prb_mem (保護機能対応カーネルのみ)
1656 • cal_svc (保護機能対応カーネルのみ)
1657
1658 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
1659 となる【NGKI0169】。
1660
1661 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
1662 【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
1663 態のいずれかの状態を取る。
1664
1665 【補足説明】
1666
1667 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
1668 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。
1669
1670 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
1671 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
1672 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、
1673 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
1674 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。
1675
1676 2.5.6 割込み優先度マスク
1677
1678 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
1679 マスクを持つ【NGKI0171】。割込み優先度マスクがTIPM_ENAALL (=0) の時は、
1680 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
1681 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
1682 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
1683 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
1684 マスクが全解除でない状態と呼ぶ。
1685
1686 割込み優先度マスクが全解除でない状態では、別に規定がない限りは、自タス
1687 クを広義の待ち状態に遷移させる可能性のあるサービスコールを呼び出すこと
1688 はできない【NGKI0174】。呼び出した場合には、E_CTXエラーとなる
1689 【NGKI0175】。
1690
1691 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
1692 持つ【NGKI0176】。
1693
1694 2.5.7 ディスパッチ禁止状態とディスパッチ許可状態
1695
1696 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
1697 【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
1698 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
1699 パッチ禁止状態では、ディスパッチは保留される。
1700

ディスパッチ禁止状態では、別に規定がない限りは、自タスクを広義の待ち状態に遷移させる可能性のあるサービスコールを呼び出すことはできない

【NGKI0178】. 呼び出した場合には、E_CTXエラーとなる【NGKI0179】.

1704

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

1708

1709 【補足説明】

1710

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

1716

1717 2.5.8 ディスパッチ保留状態

1718

非タスクコンテキストの実行中、CPUロック状態、割込み優先度マスクが全解除でない状態、ディスパッチ禁止状態では、ディスパッチが保留される

1721 【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.

1722

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

1725

1726 【補足説明】

1727

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

1730

1731 2.5.9 カーネル管理外の状態

1732

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

1737

それぞれの節で規定する通り、カーネル管理外の状態では、システムインタフェースレイヤのAPIとsns_ker, ext_kerのみ（カーネル管理外のCPU例外ハンドラからは、それに加えてxsns_dpnとxsns_xpn）を呼び出すことができ、その他のサービスコールを呼び出すことはできない. カーネル管理外の状態から、その他のサービスコールを呼び出した場合の動作は、保証されない.

1743

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

1748

1749 2.5.10 処理単位の開始・終了とシステム状態

1750

1751 各処理単位が実行開始されるシステム状態の条件（実行開始条件），各処理単位
 1752 の実行開始時にカーネルによって行われるシステム状態の変更処理（実行開始
 1753 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
 1754 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
 1755 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
 1756 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．

1757		CPUロック	割込み優先度	ディスパッチ
1758		フラグ	マスク	禁止フラグ
1759				
1760	-----			
1761	【タスク】 【NGKI0183】			
1762	実行開始条件	解除	全解除	許可
1763	実行開始時処理	そのまま	そのまま	そのまま
1764	終了前	原則解除(*1)	原則全解除(*1)	原則許可(*1)
1765	終了時処理	解除する	全解除する	許可する
1766	-----			
1767	【タスク例外処理ルーチン】 【NGKI0184】			
1768	実行開始条件	解除	全解除	任意
1769	実行開始時処理	そのまま	そのまま	そのまま
1770	リターン前	原則解除(*1)	原則全解除(*1)	元に戻す
1771	リターン時処理	解除する	全解除する	元に戻す(*4)
1772	-----			
1773	【カーネル管理の割込みハンドラ】 【NGKI0185】			
1774	【割込みサービスルーチン】 【NGKI0186】			
1775	【タイムイベントハンドラ】 【NGKI0187】			
1776	実行開始条件	解除	自優先度より低い	任意
1777	実行開始時処理	そのまま	自優先度に(*2)	そのまま
1778	リターン前	原則解除(*1)	変更不可(*3)	変更不可(*3)
1779	リターン時処理	解除する	元に戻す(*5)	そのまま
1780	-----			
1781	【CPU例外ハンドラ】 【NGKI0188】			
1782	実行開始条件	任意	任意	任意
1783	実行開始時処理	そのまま(*6)	そのまま	そのまま
1784	リターン前	原則元に(*1)	変更不可(*3)	変更不可(*3)
1785	リターン時処理	元に戻す	元に戻す(*5)	そのまま
1786	-----			
1787	【拡張サービスコール】 【NGKI0189】			
1788	実行開始条件	任意	任意	任意
1789	実行開始時処理	そのまま	そのまま	そのまま
1790	リターン前	任意	任意	任意
1791	リターン時処理	そのまま	そのまま	そのまま
1792	-----			

1793
 1794 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）
 1795 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，
 1796 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
 1797 態が設定されるため，支障がないことを意味する．

1798
 1799 「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
 1800 はそれを要求した割込みの割込み優先度，周期ハンドラとアラームハンドラの

1801 場合にはタイマ割込みの割込み優先度、オーバランハンドラの場合にはオーバ
1802 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。

1803

1804 「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
1805 が用意されていないことを示す。

1806

1807 保護機能対応カーネルでは、タスク例外処理ルーチンからのリターン時にディス
1808 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
1809 の変更を許可している場合にのみ行われる【NGKI0529】。カーネルは、ディス
1810 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
1811 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
1812 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
1813 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
1814 るとは限らない）【NGKI0190】。

1815

1816 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
1817 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
1818 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
1819 【NGKI0191】。このことから、タスク例外処理ルーチンからの終了前に、ディス
1820 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
1821 【NGKI0192】。

1822

1823 【補足説明】

1824

1825 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で
1826 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
1827 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。

1828

1829 【仕様決定の理由】

1830

1831 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
1832 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
1833 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
1834 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
1835 を防止するためである。

1836

1837 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
1838 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
1839 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
1840 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
1841 めである。

1842

1843 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
1844 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
1845 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
1846 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,
1847 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
1848 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
1849 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
1850 に戻すこととしている。

1851

1852 2.6 タスクの状態遷移とスケジューリング規則

1853

1854 2.6.1 基本的なタスク状態

1855

1856 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
1857 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
1858 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
1859 ない仮想的な状態を、未登録状態と呼ぶ。

1860

1861 (a) 実行できる状態 (runnable)

1862

1863 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
1864 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
1865 る。

1866

1867 (a.1) 実行状態 (running)

1868

1869 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
1870 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
1871 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。

1872

1873 (a.2) 実行可能状態 (ready)

1874

1875 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
1876 行状態にあるために、そのタスクが実行されない状態。

1877

1878 (b) 休止状態 (dormant)

1879

1880 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
1881 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
1882 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）
1883 は保存されていない【NGKI0194】。

1884

1885 (c) 広義の待ち状態 (blocked)

1886

1887 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
1888 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
1889 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
1890 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、
1891 （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。

1892

1893 (c.1) （狭義の）待ち状態 (waiting)

1894

1895 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。

1896

1897 (c.2) 強制待ち状態 (suspended)

1898

1899 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
1900 を強制待ち状態にすることも可能である。

1901

1902 (c.3) 二重待ち状態 (waiting-suspended)

1903

1904 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
1905 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
1906 実行を止められている状態。

1907

1908 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
1909 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
1910 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
1911 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
1912 重待ち状態でもないことを意味する。

1913

1914 (d) 未登録状態 (non-existent)

1915

1916 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、
1917 タスクは未登録状態にあるとみなす。

1918

1919 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
1920 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
1921 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。

1922

1923 【TOPPERS/ASPカーネルにおける規定】

1924

1925 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、
1926 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、
1927 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】。

1928

1929 【TOPPERS/FMPカーネルにおける規定】

1930

1931 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
1932 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]
1933 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
1934 実行中のタスクに対する強制待ち」の節を参照すること。

1935

1936 【TOPPERS/HRP2カーネルにおける規定】

1937

1938 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、
1939 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】。

1940

1941 【TOPPERS/SSPカーネルにおける規定】

1942

1943 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
1944 【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
1945 【SSPS0004】。

1946

1947 2.6.2 タスクの状態遷移

1948

1949 タスクの状態遷移を図2-2に示す【NGKI0197】。

1950

1951 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)
1952 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
1953 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
1954 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
1955 させることを、タスクを削除する (delete) という。

1956

1957 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
1958 (activate) という。起動されたタスクは、実行できる状態になる
1959 【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
1960 態) に遷移させることを、タスクを終了する (terminate) という。

1961

1962 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
1963 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
1964 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
1965 【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
1966 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
1967 遷移したタスクをプリエンプトしたという。逆に、実行可能状態に遷移したタ
1968 スクは、プリエンプトされたという。

1969

1970 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
1971 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
1972 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
1973 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
1974 態に遷移させることをいう。

1975

1976 【補足説明】

1977

1978 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
1979 に遷移する) 時のことをいう。

1980

1981 2.6.3 タスクのスケジューリング規則

1982

1983 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】。
1984 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
1985 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。

1986

1987 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
1988 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
1989 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
1990 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、
1991 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
1992 リングされる。ただし、サービスコールの呼出しにより、同じ優先度を持つタ
1993 スク間の優先順位を変更することも可能である【NGKI0206】。

1994

1995 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
1996 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
1997 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
1998 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
1999 まる【NGKI0208】。

2000

2001 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
2002 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、
2003 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
2004 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
2005 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
2006 に存在する。

2007

2008 2.6.4 待ち行列と待ち解除の順序

2009

2010 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
2011 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
2012 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
2013 クトと呼ぶ。

2014

2015 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
2016 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
2017 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
2018 【NGKI0211】。

2019

2020 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
2021 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
2022 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
2023 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
2024 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
2025 につながる【NGKI0214】。

2026

2027 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
2028 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
2029 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
2030 舞いをするかは、待ち行列毎に規定される【NGKI0215】。

2031

2032 (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
2033 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
2034 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
2035 たしていれば、先に待ち解除される。

2036

2037 (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
2038 【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
2039 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。

2040

2041 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
2042 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）、
2043 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。
2044 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
2045 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
2046 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
2047 じ処理が繰り返される【NGKI0219】。

2048

2049 2.6.5 タスク例外処理マスク状態と待ち禁止状態

2050

2051 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
2052 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
2053 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
2054 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）、
2055 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
2056 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
2057 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】。
2058 これらの状態を、タスク例外処理マスク状態と呼ぶ。
2059
2060 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
2061 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
2062 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
2063 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、
2064 サービスコールはE_RLWAIエラーとなる【NGKI0222】。
2065
2066 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
2067 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
2068 【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
2069 （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
2070 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
2071 【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
2072 ても、待ち禁止状態を解除することができる【NGKI0225】。
2073
2074 【仕様決定の理由】
2075
2076 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
2077 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
2078 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
2079 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
2080 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
2081 ルのための例外処理が行われないことになる。
2082
2083 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
2084 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
2085 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
2086 ムスタック上に不要な情報が残ってしまう。
2087
2088 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
2089 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
2090 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
2091 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
2092 る。
2093
2094 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、
2095 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、
2096 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、
2097 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
2098 外処理ルーチンを実行させることができる。
2099
2100 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

2101 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
2102 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
2103 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
2104 されないことになる。

2105
2106 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
2107 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/
2108 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
2109 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
2110 ンを実行させることができる。

2111
2112 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
2113 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
2114 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。

2115
2116 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
2117 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
2118 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
2119 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
2120 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
2121 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
2122 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。

2123
2124 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
2125 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
2126 るように実装すべきである。

2127
2128 【 μ ITRON4.0仕様、 μ ITRON4.0/PX仕様との関係】
2129

2130 待ち禁止状態は、 μ ITRON4.0仕様にはない概念であり、 μ ITRON4.0/PX仕様で導
2131 入された。ただし、 μ ITRON4.0/PX仕様では、タスクの待ち状態を強制解除する
2132 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
2133 いる。その結果 μ ITRON4.0/PX仕様は、待ち状態を強制解除するサービスコー
2134 ルの仕様において、 μ ITRON4.0仕様との互換性がなくなっている。

2135
2136 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
2137 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
2138 ルの仕様が、 μ ITRON4.0仕様と互換になっている。一方、 μ ITRON4.0/PX仕様と
2139 は互換性がない。

2140
2141 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
2142

2143 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
2144 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
2145 チ保留状態が解除されるまで保留される【NGKI0226】。

2146
2147 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
2148 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
2149 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
2150 スクは、実行可能状態にとどまる【NGKI0228】。

2151
2152 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
2153 除されると、ただちにディスパッチが起こり、タスクは強制待ち状態に遷移す
2154 る【NGKI0229】。
2155
2156 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】。
2157
2158 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。
2159
2160 (a) プロセッサを占有して実行を継続する。
2161
2162 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
2163 続して実行される【NGKI0231】。
2164
2165 (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
2166 であるものと扱う。
2167
2168 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/
2169 iget_tid, get_did, sns_tex) では、強制待ち状態 [実行継続中] のタスクが、
2170 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
2171 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/
2172 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、
2173 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
2174 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】。
2175
2176 (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。
2177
2178 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
2179 待ち状態であるものと扱う【NGKI0234】。
2180
2181 【TOPPERS/ASPカーネルにおける規定】
2182
2183 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
2184 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
2185 ち状態 [実行継続中] になることはない【ASPS0008】。
2186
2187 【TOPPERS/FMPカーネルにおける規定】
2188
2189 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
2190 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
2191 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】。
2192
2193 【TOPPERS/HRP2カーネルにおける規定】
2194
2195 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
2196 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
2197 待ち状態 [実行継続中] になることはない【HRPS0004】。
2198
2199 【TOPPERS/SSPカーネルにおける規定】
2200

2201 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
2202 制待ち状態 [実行継続中] になることもない【SSPS0005】。

2203

2204 **【補足説明】**

2205

2206 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
2207 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
2208 の間の過渡的な状態は存在しない。

2209

2210 2.6.7 制約タスク

2211

2212 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
2213 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
2214 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
2215 は以下の機能制限がある。

2216

2217 (a) 広義の待ち状態に入ることができない【NGKI0235】。

2218

2219 (b) サービスコールにより優先度を変更することができない【NGKI0236】。

2220

2221 (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
2222 先順位の回転 (rot_rdq/irotd_rdq) を行うことができない【NGKI0237】。

2223

2224 (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
2225 できない【NGKI0238】。

2226

2227 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
2228 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
2229 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
2230 ことができる【NGKI0240】。

2231

2232 **【未決定事項】**

2233

2234 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、
2235 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、
2236 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
2237 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
2238 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
2239 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、
2240 今後の課題である。

2241

2242 **【TOPPERS/ASPカーネルにおける規定】**

2243

2244 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
2245 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
2246 る【ASPS0010】。

2247

2248 **【TOPPERS/FMPカーネルにおける規定】**

2249

2250 FMPカーネルでは、制約タスクをサポートしていない【FMPS0006】。

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、制約タスクをサポートしていない【HRPS0005】。

【TOPPERS/SSPカーネルにおける規定】

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

【 μ ITRON4.0仕様との関係】

制約タスクは、 μ ITRON4.0仕様の自動車制御プロファイルで導入された機能である。この仕様における制約タスクは、 μ ITRON4.0仕様の制約タスクよりも機能制限が少なくなっている。

2.7 割込み処理モデル

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

TOPPERS標準割込み処理モデルの概念図を図2-4に示す【NGKI0241】。この図は、割込み処理モデルの持つすべての機能が、ハードウェア（プロセッサおよび割込みコントローラ）で実現されているとして描いた概念図である。実際のハードウェアで不足している機能については、カーネル内の割込み処理のソフトウェアで実現される。

【 μ ITRON4.0仕様との関係】

割込み処理モデルは、 μ ITRON4.0仕様から大幅に拡張している。

2.7.1 割込み処理の流れ

周辺デバイス（以下、デバイスと呼ぶ）からの割込み要求は、割込みコントローラ（IRC）を経由して、プロセッサに伝えられる。デバイスから割込みコントローラに割込み要求を伝えるための信号線を、割込み要求ラインと呼ぶ。一般には、1つの割込み要求ラインに、複数のデバイスからの割込み要求が接続される。

プロセッサは、デバイスからの割込み要求を受け付ける条件が満たされた場合、割込み要求を受け付ける【NGKI0242】。受け付けた割込み要求が、カーネル管理の割込みである場合には、カーネル内の割込みハンドラの入口処理（割込み入口処理）を経由して、カーネル内の割込みハンドラを実行する【NGKI0243】。

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

2301 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
2302 することができる【NGKI0246】。

2303

2304 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
2305 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
2306 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】。

2307

2308 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
2309 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
2310ウト処理等が行われることに加えて、アプリケーションが登録したタイムイベ
2311ントハンドラが呼び出される【NGKI0248】。

2312

2313 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
2314 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】。

2315

2316 2.7.2 割込み優先度

2317

2318 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
2319 【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
2320 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
2321 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、
2322 マスクされる。

2323

2324 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
2325 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
2326 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
2327 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
2328 【NGKI0253】。

2329

2330 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
2331 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
2332 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
2333 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
2334 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、
2335 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
2336 【NGKI0254】。

2337

2338 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、
2339 【NGKI0037】の原則には従わず、-1から連続した負の値を用いる【NGKI0255】。

2340

2341 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
2342 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
2343 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
2344 を1にする（すなわち、多重割込みを許さない）場合がある。

2345

2346 【仕様決定の理由】

2347

2348 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
2349 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
2350 先度マスクの値を0にできるためである。

2351

2352 2.7.3 割込み要求ラインの属性

2353

2354 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
2355 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
2356 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
2357 ことはできない。

2358

2359 (1) 割込み要求禁止フラグ

2360

2361 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
2362 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
2363 ンによって伝えられる割込み要求はマスクされる【NGKI0259】。

2364

2365 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
2366 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
2367 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
2368 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
2369 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
2370 する場合がある【NGKI0261】。

2371

2372 (2) 割込み優先度

2373

2374 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】。
2375 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
2376 対して設定された割込み優先度のことである【NGKI0263】。

2377

2378 (3) トリガモード

2379

2380 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
2381 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
2382 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
2383 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
2384 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
2385 設定できる場合もある【NGKI0266】。

2386

2387 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
2388 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、トリガモードの
2389 設定をサポートしない場合がある【NGKI0267】。

2390

2391 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
2392 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
2393 止フラグをクリアすることもできない【NGKI0269】。

2394

2395 【使用上の注意】

2396

2397 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
2398 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、
2399 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
2400 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

2401 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
2402 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
2403 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
2404 実現すべきである。

2405

2406 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
2407 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、
2408 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、
2409 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
2410 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
2411 割込みハンドラが実行され、デバイスAの割込み処理を行った後、デバイスBの
2412 割込み処理を行う前に、デバイスAからの割込み要求が発生した場合に、デバイ
2413 スAからの割込み要求を取りこぼしてしまう。

2414

2415 2.7.4 割込みを受け付ける条件

2416

2417 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
2418 【NGKI0270】。

2419

2420 (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

2421

2422 (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
2423 在値よりも高い（優先度の値としては小さい）こと

2424

2425 (c) 全割込みロックフラグがクリアされていること

2426

2427 (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
2428 リアされていること

2429

2430 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
2431 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
2432 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。

2433

2434 2.7.5 割込み番号と割込みハンドラ番号

2435

2436 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、
2437 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
2438 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】。
2439 そのため、1から連続した正の値であるとは限らない。

2440

2441 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
2442 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、
2443 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
2444 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
2445 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
2446 た正の値であるとは限らない。

2447

2448 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
2449 一致する場合が多い）【NGKI0274】。

2450

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

2.7.6 マルチプロセッサにおける割込み処理

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

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

割込み番号は、割込み要求ラインを識別するための番号であることから、割込み要求ラインが複数のプロセッサに接続されている場合でも、1つの割込み要求ラインには1つの割込み番号を付与する【NGKI0279】。逆に、複数のプロセッサが同じ種類のデバイスを持っている場合でも、別のデバイスからの割込み要求ラインには異なる割込み番号を付与する（図2-5）【NGKI0280】。図2-5において、ローカルIRCは個々のプロセッサに対する割込みを制御するための回路であり、グローバルIRCはデバイスからの割込みをプロセッサに分配するための回路である。グローバルIRCは、必ず備わっているとは限らない。

割込み要求禁止フラグは、この仕様上はシステム全体で共通に持つこととしているが、実際のターゲットハードウェア（特に、グローバルIRCを備えていないもの）では、プロセッサ毎に持っている場合がある。そのため、ターゲット定義で、あるプロセッサで割込み要求禁止フラグを動的にセット／クリアしても、他のプロセッサに対しては割込みがマスク／マスク解除されない場合があるものとする【NGKI0281】。

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

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

それに対して、割込みハンドラはプロセッサ毎に登録する。そのため、同じ割込み要求に対応する割込みハンドラであっても、プロセッサ毎に異なる割込みハンドラ番号を付与する（図2-5）【NGKI0285】。割込みハンドラが属するクラ

2501 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
2502 ければならない【NGKI0286】。

2503

2504 【補足説明】

2505

2506 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
2507 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
2508 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
2509 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
2510 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
2511 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
2512 とする（図2-5）。

2513

2514 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
2515 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
2516 スに属していなければならない。

2517

2518 【使用上の注意】

2519

2520 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
2521 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
2522 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。

2523

2524 2.7.7 カーネル管理外の割込み

2525

2526 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
2527 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
2528 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、
2529 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
2530 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、
2531 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】。

2532

2533 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
2534 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
2535 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、
2536 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
2537 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】。

2538

2539 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
2540 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
2541 ンドラをカーネル管理の割込みハンドラと呼ぶ。

2542

2543 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
2544 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
2545 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
2546 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
2547 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】。

2548

2549 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
2550 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

2551 は、ターゲット定義である【NGKI0292】。カーネル管理外の割込みハンドラから
2552 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
2553 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】。
2554 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
2555 場合の動作は、保証されない【NGKI0294】。

2556

2557 2.7.8 カーネル管理外の割込みの設定方法

2558

2559 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
2560 ずれかが採用される【NGKI0295】。

2561

2562 (a-1) NMI以外にカーネル管理外の割込みを設けない

2563 (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

2564

2565 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
2566 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
2567 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、
2568 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
2569 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
2570 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
2571 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
2572 (TA_NONKERNEL)を指定する【NGKI0298】。また、割込み要求ラインの属性を
2573 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
2574 【NGKI0299】。

2575

2576 (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

2577

2578 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、
2579 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
2580 であることを示す割込みハンドラ属性(TA_NONKERNEL)を指定する
2581 【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
2582 て設定する割込み優先度を、TMIN_INTPRIよりも高い値とする【NGKI0301】。

2583

2584 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
2585 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】。
2586 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
2587 ことはできない【NGKI0303】。

2588

2589 2.8 CPU例外処理モデル

2590

2591 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
2592 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
2593 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
2594 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、
2595 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。

2596

2597 2.8.1 CPU例外処理の流れ

2598

2599 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
2600 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

2601 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
2602 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
2603 【NGKI0305】。
2604

2605 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
2606 ンドラ番号と呼ぶ。CPU例外ハンドラ番号は、符号無しの整数型であるEXCNO型
2607 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
2608 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
2609 値であるとは限らない。
2610

2611 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、
2612 異なるCPU例外であると扱う【NGKI0307】。すなわち、同じ種類のCPU例外であつ
2613 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、
2614 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
2615 の初期割付けプロセッサは、CPU例外が発生するプロセッサと一致していなければ
2616 ならない【NGKI0308】。
2617

2618 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
2619 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
2620 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
2621 が考えられる【NGKI0310】。
2622

2623 (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。
2624

2625 (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
2626 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
2627 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
2628 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
2629 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
2630 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
2631 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）
2632 【NGKI0311】。
2633

2634 (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
2635 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）。
2636

2637 (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
2638 る）。
2639

2640 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
2641 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
2642 に対して(b)と(c)の方法は、CPU例外ハンドラからそのためのサービスコールを呼
2643 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
2644 テキストおよび状態に依存する【NGKI0313】。
2645

2646 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
2647 舞いは、ターゲット定義である【NGKI0314】。
2648

2649 【使用上の注意】
2650

2651 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
2652 入口処理で同じ原因でCPU例外が発生すると、CPU例外が繰り返し発生し、アプ
2653 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
2654 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
2655 きない場合には、CPU例外入口処理を経由せずに、アプリケーションが用意した
2656 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。

2657

2658 **【補足説明】**

2659

2660 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、
2661 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
2662 番号を含める方法を基本とする。

2663

2664 **【 μ ITRON4.0仕様との関係】**

2665

2666 μ ITRON4.0仕様では、CPU例外からのリカバリ処理の方法については、記述され
2667 ていない。

2668

2669 2.8.2 CPU例外ハンドラから呼び出せるサービスコール

2670

2671 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
2672 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
2673 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
2674 ことができる【NGKI0315】。

2675

2676 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
2677 チできる状態であった場合にfalseを返す【NGKI0316】。xsns_dpnがfalseを返
2678 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
2679 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
2680 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
2681 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】。

2682

2683 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
2684 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】。

2685 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
2686 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
2687 によるリカバリ処理が可能である【NGKI0320】。

2688

2689 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
2690 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
2691 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
2692 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
2693 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、
2694 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
2695 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
2696 ないことになる。

2697

2698 **【 μ ITRON4.0仕様との関係】**

2699

2700 CPU例外ハンドラで行える操作に関しては、 μ ITRON4.0仕様を見直し、全面的に

2701 修正した.

2702

2703 2.8.3 エミュレートされたCPU例外ハンドラ

2704

2705 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
2706 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
2707 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
2708 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、
2709 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
2710 レートされたCPU例外ハンドラと呼ぶ【NGKI0325】.

2711

2712 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
2713 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、
2714 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
2715 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
2716 を行うことができる【NGKI0327】.

2717

2718 【μITRON4.0仕様との関係】

2719

2720 エミュレートされたCPU例外およびCPU例外ハンドラは、μITRON4.0仕様に定義
2721 されていない概念である.

2722

2723 2.8.4 カーネル管理外のCPU例外

2724

2725 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（これを、
2726 カーネル実行中と呼ぶ）、全割込みロック状態、CPUロック状態、カーネル管理
2727 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
2728 CPU例外と呼ぶ. また、それによって起動されるCPU例外ハンドラを、カーネル
2729 管理外のCPU例外ハンドラと呼ぶ. さらに、カーネル管理外のCPU例外ハンドラ
2730 実行中に発生したCPU例外も、カーネル管理外のCPU例外とする.

2731

2732 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
2733 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
2734 理のCPU例外ハンドラと呼ぶ.

2735

2736 カーネル管理外のCPU例外ハンドラからは、システムインタフェースレイヤの
2737 APIとsns_ker, ext_ker, xsns_dpn, xsns_xpnのみを呼び出すことができ、その
2738 他のサービスコールを呼び出すことはできない【NGKI0328】. カーネル管理外
2739 のCPU例外ハンドラから、その他のサービスコールを呼び出した場合の動作は、
2740 保証されない【NGKI0329】.

2741

2742 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
2743 サービスコールもtrueを返す【NGKI0330】. そのため、カーネル管理外のCPU例
2744 外からは、(a)または(d)の方法によるリカバリ処理しか行えない.

2745

2746 【補足説明】

2747

2748 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
2749 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
2750 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.

2751
2752 2.9 システムの初期化と終了
2753
2754 2.9.1 システム初期化手順
2755
2756 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
2757 と呼ぶ。スタートアップモジュールはカーネルの管理外であり、アプリケーション
2758 で用意するのが基本であるが、スタートアップモジュールで行うべき処理
2759 を明確にするために、カーネルの配布パッケージの中に、標準のスタートアップ
2760 モジュールが用意されている【NGKI0331】。
2761
2762 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
2763 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
2764 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
2765 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
2766 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
2767 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
2768 呼び出すターゲットシステム依存の初期化フックでは、リセット後に速やかに行
2769 うべき初期化処理を行うことが想定されている。
2770
2771 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
2772 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、
2773 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
2774 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）
2775 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
2776 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
2777 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
2778 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
2779 する必要がある【NGKI0335】。
2780
2781 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
2782 ネル内のデータ構造の初期化、カーネルが用いるデバイスの初期化など）と静的
2783 APIの処理（オブジェクトの登録など）が行われる【NGKI0336】。静的APIのバ
2784 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
2785 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
2786 れる場合もある【NGKI0337】。
2787
2788 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
2789 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
2790 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
2791 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
2792 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
2793 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
2794 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
2795 順となるとは限らない。
2796
2797 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
2798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
2799 【NGKI0339】。
2800

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。

マルチプロセッサ対応カーネルでは、カーネルを終了させるサービスコール（ext_ker）は、どのプロセッサからでも呼び出すことができる【NGKI0346】。1つのプロセッサでカーネルを終了させるサービスコールを呼び出すと、そのプロセッサがカーネル動作状態から非動作状態に遷移した後、他のプロセッサに対してカーネル終了処理の開始を要求する【NGKI0347】。複数のプロセッサから、カーネルを終了させるサービスコール（ext_ker）を呼び出してもよい【NGKI0348】。

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

マルチプロセッサ対応カーネルでは、すべてのプロセッサがカーネル非動作状態に遷移した後に、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チンは、初期割付けプロセッサにより実行される。すべてのプロセッサでロー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

2851 終了処理ルーチンを実行する【NGKI0351】。

2852

2853 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

2854 【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
2855 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
2856 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

2857 【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
2858 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】。

2859

2860 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
2861 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
2862 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
2863 グを、図2-7に示す【NGKI0356】。

2864

2865 【使用上の注意】

2866

2867 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
2868 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
2869 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。

2870

2871 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
2872 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
2873 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
2874 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
2875 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
2876 ネルを終了させることができる。

2877

2878 【 μ ITRON4.0仕様との関係】

2879

2880 μ ITRON4.0仕様には、システム終了に関する規定はない。

2881

2882 2.10 オブジェクトの登録とその解除

2883

2884 2.10.1 ID番号で識別するオブジェクト

2885

2886 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
2887 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
2888 る静的API（ATT_YYY，ATA_YYY）によってカーネルに登録する【NGKI0357】。オ
2889 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
2890 ないため、ID番号を指定して操作することができない【NGKI0358】。

2891

2892 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
2893 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
2894 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
2895 済みの場合には、E_OBJエラーとなる【NGKI0360】。

2896

2897 オブジェクトを生成するサービスコール（acre_yyy）は、割り付け可能なID番号
2898 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
2899 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

2900 【NGKI0361】。割り付けたID番号は、サービスコールの返回值としてアプリケー

2901 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
2902 には、E_NOID エラーとなる【NGKI0363】．
2903
2904 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
2905 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
2906 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】．
2907
2908 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
2909 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
2910 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
2911 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
2912 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
2913 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
2914 意が必要である．
2915
2916 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、
2917 登録を解除することができない【NGKI0368】．登録を解除しようとした場合に
2918 は、E_OBJエラーとなる【NGKI0369】．
2919
2920 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
2921 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
2922 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
2923 単位の実行は継続される【NGKI0371】．
2924
2925 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
2926 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
2927 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
2928 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
2929 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
2930 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
2931 規定する【NGKI0374】．
2932
2933 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
2934 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
2935 【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
2936 録したオブジェクトも、再初期化することができる【NGKI0376】．
2937
2938 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
2939 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
2940 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
2941 【NGKI0377】．
2942
2943 【μITRON4.0仕様との関係】
2944
2945 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
2946 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
2947 クトは、登録解除できないこととした．
2948
2949 μITRON4.0仕様では、割付け可能なID番号の数を指定する静的API (AID_YYY)
2950 は規定されていない．

2951

2952

2953

2954

2955

2956

2957

2958

2959

2960

2961

2962

2963

2964

2965

2966

2967

2968

2969

2970

2971

2972

2973

2974

2975

2976

2977

2978

2979

2980

2981

2982

2983

2984

2985

2986

2987

2988

2989

2990

2991

2992

2993

2994

2995

2996

2997

2998

2999

3000

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ができる【NGKI0380】。登録解除されていないオブジェクト番号に対して再度オブジェクトを登録しようとした場合には、E_OBJエラーとなる【NGKI0381】。

オブジェクトを定義する静的APIによって登録したオブジェクトは、登録を解除することができない【NGKI0382】。登録を解除しようとした場合には、E_OBJエラーとなる【NGKI0383】。

なお、動的生成対応カーネル以外では、オブジェクトを定義するサービスコール (def_yyy) はサポートされない【NGKI0384】。

【 μ ITRON4.0仕様との関係】

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

2.10.3 識別番号を持たないオブジェクト

3001 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
3002 トを追加する静的API (ATT_YYY) によってカーネルに登録する。

3003

3004 2.10.4 オブジェクト生成に必要なメモリ領域

3005

3006 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
3007 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
3008 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

3009 【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
3010 フィギュレータまたはカーネルにより確保される【NGKI0386】.

3011

3012 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、
3013 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
3014 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。

3015

- 3016 ・データキュー管理領域
- 3017 ・優先度データキュー管理領域
- 3018 ・優先度別のメッセージキューヘッダ領域
- 3019 ・固定長メモリプール管理領域

3020

3021 【補足説明】

3022

3023 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
3024 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
3025 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
3026 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。

3027

3028 2.10.5 オブジェクトが属する保護ドメインの設定

3029

3030 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
3031 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

3032 【NGKI0387】.

3033

3034 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
3035 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
3036 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
3037 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）

3038 【NGKI0389】.

3039

3040 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
3041 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
3042 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
3043 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
3044 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
3045 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
3046 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
3047 録する場合には、domidにTDOM_NONE (=-2)を指定する。

3048

3049 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
3050 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

3051 オブジェクト属性で設定する必要がないものもある【NGKI0391】。

3052

3053 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、
3054 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
3055 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
3056 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、
3057 E_RSATRエラーとなる【NGKI0394】。

3058

3059 **【補足説明】**

3060

3061 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
3062 用意していない。

3063

3064 **【仕様決定の理由】**

3065

3066 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
3067 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
3068 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
3069 タを増やさない方が望ましいためである。

3070

3071 2.10.6 オブジェクトが属するクラスの設定

3072

3073 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
3074 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

3075 【NGKI0395】。

3076

3077 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
3078 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
3079 【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
3080 囲みの外に記述する（「2.12.4 クラスの指定」の節を参照）【NGKI0397】。

3081

3082 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
3083 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
3084 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
3085 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
3086 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
3087 ができる。

3088

3089 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、
3090 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】。
3091 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
3092 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
3093 は、E_RSATRエラーとなる【NGKI0401】。

3094

3095 **【補足説明】**

3096

3097 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
3098 ていない。

3099

3100 **【仕様決定の理由】**

3101
3102 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
3103 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
3104 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
3105 タを増やさない方が望ましいためである。

3106 3107 2.10.7 オブジェクトの状態参照

3108
3109 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
3110 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
3111 (`ref_yyy`, `get_yyy`)を用意する【NGKI0402】。

3112
3113 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
3114 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）
3115 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
3116 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
3117 ある【NGKI0404】。

3118 3119 2.11 オブジェクトのアクセス保護

3120
3121 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
3122 内容は、保護機能対応カーネルにのみ適用される。

3123 3124 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

3125
3126 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
3127 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
3128 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
3129 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
3130 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
3131 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
3132 保護される【NGKI0407】。

3133
3134 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、
3135 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。

3136
3137 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
3138 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
3139 `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
3140 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
3141 から`E_OACV`エラーが返る【NGKI0409】。

3142
3143 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
3144 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
3145 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
3146 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、
3147 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、
3148 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。
3149 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。

3150

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

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

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

【未決定事項】

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

【 μ ITRON4.0/PX仕様との関係】

μ ITRON4.0/PX仕様では、アクセス保護の実装定義の制限について規定しているが、この仕様では、メモリオブジェクトに対するアクセス許可ベクタのターゲット定義の制限以外については規定していない。

【仕様決定の理由】

オブジェクト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、アクセス許可ベクタが設定されていないためである。

2.11.2 メモリオブジェクトに対するアクセス許可ベクタの制限

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

ただし、少なくとも、次の5つの組み合わせの設定は、行うことができる。

- (a) メモリオブジェクトが属する保護ドメインのみに、読出しアクセス（実行アクセスを含む）のみを許可する【NGKI0418】。これを、専有リードオンリー（private read only）と呼ぶ。
- (b) メモリオブジェクトが属する保護ドメインのみに、書込みアクセスと読出しアクセス（実行アクセスを含む）を許可する【NGKI0419】。これを、専有リードライト（private read/write）と呼ぶ。
- (c) すべての保護ドメインに、読出しアクセス（実行アクセスを含む）のみを

3201 許可する【NGKI0420】。これを、共有リードオンリー (shared read only)
3202 と呼ぶ。

3203

3204 (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
3205 スを含む) を許可する【NGKI0421】。これを、共有リードライト (shared
3206 read/write) と呼ぶ。

3207

3208 (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
3209 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
3210 アクセス (実行アクセスを含む) のみを許可する【NGKI0422】。これを、
3211 共有リード専有ライト (shared read private write) と呼ぶ。

3212

3213 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
3214 の数が制限される場合がある【NGKI0423】。

3215

3216 2.11.3 デフォルトのアクセス許可ベクタ

3217

3218 静的APIによりカーネルオブジェクトに登録した直後は、次に規定されるデフォ
3219 ルトのアクセス許可ベクタが設定される。

3220

3221 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
3222 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
3223 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
3224 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
3225 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
3226 する保護ドメインのID番号) に設定される。

3227

3228 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、
3229 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
3230 可パターンがいずれも、TACP_SHAREDに設定される。

3231

3232 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
3233 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
3234 ターンがいずれも、TACP_KERNELに設定される。

3235

3236 【未決定事項】

3237

3238 サービスコールによりカーネルオブジェクトに登録した直後のアクセス許可ベ
3239 クタについては、今後の課題である。

3240

3241 2.11.4 アクセス許可ベクタの設定

3242

3243 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
3244 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
3245 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
3246 【NGKI0427】。

3247

3248 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
3249 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
3250 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】。

3251 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)
3252 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【NGKI0429】。アクセ
3253 ス許可ベクタを設定しようとした場合には、E_OBJエラーとなる【NGKI0430】。
3254
3255 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
3256 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
3257 的API (ATA_YYY) が用意されている【NGKI0431】。
3258
3259 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
3260 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】。
3261 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
3262 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
3263 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
3264 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
3265 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
3266 タスクには影響しない。
3267
3268 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
3269 する機能は用意していない。
3270
3271 【μ ITRON4.0/PX仕様との関係】
3272
3273 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
3274 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
3275 API (SAC_YYY) をサポートすることとした。
3276
3277 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
3278 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。
3279
3280 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
3281 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。
3282 これは、【NGKI0403】の原則に合わせるための修正である。
3283
3284 2.11.5 カーネルの管理領域のアクセス保護
3285
3286 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
3287 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
3288 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。
3289 そのため、カーネルの管理領域は、4つの種別のアクセスがカーネルドメインの
3290 みに許可されたメモリオブジェクト（これを、カーネル専用のメモリオブジェ
3291 クトと呼ぶ）の中に置かれる【NGKI0433】。
3292
3293 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。
3294 「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
3295 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
3296 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
3297 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
3298 ブジェクトの中に確保される【NGKI0435】。
3299
3300 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

3301 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
3302 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、
3303 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
3304 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
3305 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
3306 理領域とは異なる扱いとなる。

3307

3308 2.11.6 ユーザタスクのユーザスタック領域

3309

3310 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
3311 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
3312 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
3313 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
3314 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。

3315

3316 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
3317 よび読出しアクセスを行うことができる【NGKI0438】。そのため、書込みアク
3318 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
3319 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
3320 行えるかどうかは、ターゲット定義である【NGKI0440】。

3321

3322 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、
3323 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
3324 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】。

3325

3326 【μITRON4.0/PX仕様との関係】

3327

3328 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
3329 きるものとした。

3330

3331 2.12 システムコンフィギュレーション手順

3332

3333 2.12.1 システムコンフィギュレーションファイル

3334

3335 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
3336 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
3337 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
3338 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
3339 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。

3340

3341 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
3342 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
3343 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
3344 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
3345 ることができる【NGKI0442】。

3346

3347 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
3348 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、
3349 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
3350 レクトリの違いのみ）【NGKI0443】。

3351
3352 INCLUDE("ファイル名");
3353 INCLUDE(<ファイル名>);
3354
3355 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
3356 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
3357 【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
3358 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
3359 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
3360 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。
3361
3362 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
3363 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
3364 【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
3365 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
3366 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
3367 【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
3368 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
3369 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
3370 【NGKI0448】。
3371
3372 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
3373 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
3374 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
3375 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
3376 ラーとなる。
3377
3378 【μITRON4.0仕様との関係】
3379
3380 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
3381 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
3382 レクティブを設けた。また、共通静的APIを廃止した。μITRON4.0仕様における
3383 #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
3384 換わる。逆に、μITRON4.0仕様におけるINCLUDE静的APIの役割は、この仕様で
3385 は#includeディレクティブに置き換わる。
3386
3387 2.12.2 静的APIの文法とパラメータ
3388
3389 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
3390 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
3391 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
3392 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
3393 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"{
3394 と"}"で囲んだ形で記述する【NGKI0451】。
3395
3396 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
3397 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】。
3398
3399 静的APIのパラメータは、次の4種類に分類される。
3400

3401 (a) オブジェクト識別名

3402

3403 オブジェクトのID番号を指定するパラメータ。オブジェクトの名称を表す単一の識別名のみを記述することができる。

3405

3406 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
3407 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、
3408 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
3409 ディレクティブ (#define) を生成する【NGKI0453】。

3410

3411 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
3412 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
3413 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
3414 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
3415ギュレータがエラーを報告する【NGKI0455】。

3416

3417 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
3418 はできない【NGKI0456】。

3419

3420 (b) 整数定数式パラメータ

3421

3422 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、
3423 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
3424 決まる整数定数式を記述することができる。

3425

3426 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
3427 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
3428 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】。

3429

3430 (c) 一般定数式パラメータ

3431

3432 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
3433 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。

3434

3435 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
3436 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
3437ードするファイルに含まれていなければならない【NGKI0458】。

3438

3439 (d) 文字列パラメータ

3440

3441 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。
3442 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。

3443

3444 【μ ITRON4.0仕様との関係】

3445

3446 μ ITRON4.0仕様においては、静的APIのパラメータを次の4種類に分類していた
3447 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。

3448

3449 (A) 自動割付け対応整数値パラメータ

3450 (B) 自動割付け非対応整数値パラメータ

- 3451 (C) プリプロセッサ定数式パラメータ
3452 (D) 一般定数式パラメータ

3453

3454 この仕様の(a)が、おおよそ μ ITRON4.0仕様の(A)に相当するが、(a)には整数値
3455 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
3456 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。

3457

3458 μ ITRON4.0仕様では、静的APIのパラメータは基本的には(D)とし、コンフィギュ
3459 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
3460 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
3461 るパラメータを(C)としていた。

3462

3463 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
3464 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
3465 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
3466 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
3467 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。

3468

3469 2.12.3 保護ドメインの指定

3470

3471 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
3472 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
3473 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

3474 【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、
3475 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
3476 ラーを報告する【NGKI0461】。

3477

3478 ユーザドメインの囲みの文法は次の通り【NGKI0462】。

3479

```
3480     DOMAIN(保護ドメイン名) {  
3481         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  
3482     }
```

3483

3484 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
3485 ることができる【NGKI0463】。

3486

3487 コンフィギュレータは、ユーザドメインの囲みを処理する際に、ユーザドメイ
3488 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
3489 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
3490 サのディレクティブ (#define) を生成する【NGKI0464】。また、ユーザドメイ
3491 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
3492 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
3493 【NGKI0465】。

3494

3495 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
3496 ンIDの割付けのみを行うことができる【NGKI0466】。

3497

3498 カーネルドメインの囲みの文法は次の通り【NGKI0467】。

3499

```
3500     KERNEL_DOMAIN {
```

3501 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
3502 }

3503
3504 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
3505 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
3506 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
3507 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
3508 【NGKI0469】。

3509
3510 【μ ITRON4.0/PX仕様との関係】

3511
3512 ユーザドメインの囲みの文法を変更した。

3513
3514 【仕様決定の理由】

3515
3516 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
3517 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
3518 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
3519 クト属性によって指定するためである。

3520
3521 2.12.4 クラスの指定

3522
3523 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
3524 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
3525 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
3526 合には、コンフィギュレータがE_RSATRエラーを報告する【NGKI0471】。

3527
3528 クラスの囲みの文法は次の通り【NGKI0472】。

3529
3530 CLASS(クラスID) {
3531 クラスに属するオブジェクトを登録する静的API等
3532 }

3533
3534 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
3535 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
3536 ギュレータがE_IDエラーを報告する【NGKI0474】。

3537
3538 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】。
3539 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
3540 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
3541 を報告する【NGKI0476】。

3542
3543 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
3544 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】。

3545
3546 【仕様決定の理由】

3547
3548 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
3549 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
3550 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

3551 定するためである.

3552

3553 2.12.5 コンフィギュレータの処理モデル

3554

3555 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
3556 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
3557 成する (図2-8) .

3558

3559 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
3560 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
3561 めに、パラメータ計算用C言語ファイル (cfg1_out.c) を生成する. この時、シ
3562 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
3563 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
3564 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
3565 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
3566 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.

3567

3568 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
3569 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
3570 マットの形に変換したSレコードファイル (cfg1_out.srec) と、その中の各シン
3571 ボールとアドレスの対応表を含むシンボルファイル (cfg1_out.syms) を生成す
3572 る. 静的APIのパラメータに解釈できない式が記述された場合には、この段階で
3573 エラーが検出される.

3574

3575 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
3576 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
3577 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
3578 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
3579 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)
3580 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
3581 (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
3582 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
3583 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.

3584

3585 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、
3586 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
3587 ル (system.srec) とシンボルファイル (system.syms) を生成する.

3588

3589 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
3590 のSレコードファイルとシンボルファイルから、静的APIのパラメータの値な
3591 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
3592 正がある場合には、この段階でエラーが検出される.

3593

3594 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
3595 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) .

3596

3597 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
3598 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
3599 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
3600 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) とシンボルファイル (cfg2_out.syms) も、最終的なものと混同しないように、異なるファイル名で生成する。

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

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

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

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

【μITRON4.0仕様との関係】

コンフィギュレータの処理モデルは全面的に変更した。

2.12.6 静的APIのパラメータに関するエラー検出

静的APIのパラメータに関するエラー検出は、同じものがサービスコールとして呼ばれた場合と同等とすることを原則とする【NGKI0478】。言い換えると、サー

ビスコールによっても検出できないエラーは、静的APIにおいても検出しない。
静的APIの機能説明中の「E_XXXXXエラーとなる」または「E_XXXXXエラーが返る」という記述は、コンフィギュレータがそのエラーを検出することを意味する。

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

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

【μ ITRON4.0仕様との関係】

μ ITRON4.0仕様では、静的APIのパラメータに関するエラー検出について規定されていない。

2.12.7 オブジェクトのID番号の指定

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。

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

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

【μ ITRON4.0仕様との関係】

μ ITRON4.0仕様では、オブジェクト生成のための静的APIのID番号を指定するパラメータに整数値を記述できるため、このような機能は用意されていない。

2.13 TOPPERSネーミングコンベンション

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

2.13.1 モジュール識別名

異なるモジュールのAPIの構成要素の名称が衝突することを避けるために、各モ

3701 ジュールに対して、それを識別するためのモジュール識別名を定める。モジュール
3702 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。

3703

3704 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
3705 ジュール識別名は"sil"とする。

3706

3707 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
3708 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
3709 識別名を含めない名称を使用する。

3710

3711 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
3712 したものをWWWと表記する。

3713

3714 2.13.2 データ型名

3715

3716 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
3717 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
3718 "_t"である名称とする。

3719

3720 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、
3721 "_"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
3722 が"_T"である名称とする場合もある。

3723

3724 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。

3725

3726 (A) パケットのデータ型

3727

3728	T_CYYY	acre_yyyに渡すパケットのデータ型
3729	T_DYYY	def_yyyに渡すパケットのデータ型
3730	T_RYYY	ref_yyyに渡すパケットのデータ型
3731	T_WWW_CYYY	www_acre_yyyに渡すパケットのデータ型
3732	T_WWW_DYYY	www_def_yyyに渡すパケットのデータ型
3733	T_WWW_RYYY	www_ref_yyyに渡すパケットのデータ型

3734

3735 2.13.3 関数名

3736

3737 関数の名称は、英小文字、数字、"_"で構成する。

3738

3739 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。

3740

3741 (A) サービスコール

3742

3743 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
3744 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
3745 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、
3746 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
3747 は、zzxxx_yyyまたはwww_zzxxx_yyyの名称となる。

3748

3749 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
3750 字として"i"を付加し、ixxx_yyy, izxxx_yyy, www_ixxx_yyy, www_izxxx_yyyと

3751 いった名称とする.

3752

3753 **【補足説明】**

3754

3755 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
3756 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.

3757

3758 (B) コールバック

3759

3760 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.

3761

3762 2.13.4 変数名

3763

3764 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
3765 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.

3766

3767 変数の名称に関して, 次のガイドラインを設ける.

3768

3769	~id	~ID (オブジェクトのID番号, ID型)
3770	~no	~番号 (オブジェクト番号)
3771	~atr	~属性 (オブジェクト属性, ATR型)
3772	~stat	~状態 (オブジェクト状態, STAT型)
3773	~mode	~モード (サービスコールの動作モード, MODE型)
3774	~pri	~優先度 (優先度, PRI型)
3775	~sz	~サイズ (単位はバイト数, SIZE型またはuint_t型)
3776	~cnt	~の個数 (単位は個数, uint_t型)
3777	~ptn	~パターン
3778	~tim	~時刻, ~時間
3779	~cd	~コード
3780	i~	~の初期値
3781	max~	~の最大値
3782	min~	~の最小値
3783	left~	~の残り

3784

3785 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
3786 ンを設ける.

3787

3788	p_~	ポインタ
3789	pp_~	ポインタを入れる領域へのポインタ
3790	pk_~	パケットへのポインタ
3791	ppk_~	パケットへのポインタを入れる領域へのポインタ

3792

3793 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.

3794

3795 (A) パケットへのポインタ

3796

3797	pk_cyyy	acre_yyyに渡すパケットへのポインタ
3798	pk_dyyy	def_yyyに渡すパケットへのポインタ
3799	pk_ryyy	ref_yyyに渡すパケットへのポインタ
3800	pk_www_cyyy	www_acre_yyyに渡すパケットへのポインタ

3801 pk_www_dydy www_def_yyyに渡すパケットへのポインタ
3802 pk_www_ryyy www_ref_yyyに渡すパケットへのポインタ
3803

3804 2.13.5 定数名

3805

3806 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
3807 字、“_”で構成する。

3808

3809 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。

3810

3811 (A) メインエラーコード

3812

3813 メインエラーコードは、先頭が“E_”である名称とする。

3814

3815 (B) 機能コード

3816

3817 TFN_XXX_YYY xxx_yyyの機能コード

3818 TFN_WWW_XXX_YYY www_xxx_yyyの機能コード

3819

3820 (C) その他の定数

3821

3822 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、
3823 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
3824 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。

3825

3826 また、定数の名称に関して、次のガイドラインを設ける。

3827

3828 TA_～ オブジェクトの属性値

3829 TSZ_～ ～のサイズ

3830 TBIT_～ ～のビット数

3831 TMAX_～ ～の最大値

3832 TMIN_～ ～の最小値

3833

3834 2.13.6 マクロ名

3835

3836 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
3837 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
3838 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
3839 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
3840 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
3841 する場合もある。

3842

3843 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。

3844

3845 (A) 構成マクロ

3846

3847 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
3848 る。

3849

3850 TSZ_～ ～のサイズ

3851 TBIT_〜 〜のビット数
3852 TMAX_〜 〜の最大値
3853 TMIN_〜 〜の最小値
3854

3855 2.13.7 静的API名

3856

3857 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
3858 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
3859 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
3860 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。
3861

3862 2.13.8 ファイル名

3863

3864 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
3865 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。
3866 また、“-”も使用しない。
3867

3868 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。
3869

3870 (A) ヘッダファイル

3871

3872 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
3873 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。
3874

3875 2.13.9 モジュール内部の名称の衝突回避

3876

3877 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
3878 ために、次のガイドラインを設ける。
3879

3880 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
3881 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
3882 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、
3883 C言語レベルで、先頭が“_kernel_”または“_KERNEL_”である名称とする。
3884

3885 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
3886 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
3887 能性のある名称は、“TOPPERS_”で始まる名称とする。
3888

3889 2.14 TOPPERS共通定義

3890

3891 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。
3892

3893 2.14.1 TOPPERS共通ヘッダファイル

3894

3895 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
3896 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
3897 れている【NGKI0484】。TOPPERS共通定義を用いる場合には、TOPPERS共通ヘッ
3898 ダファイルをインクルードする【NGKI0485】。
3899

3900 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

3901 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、
 3902 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
 3903 直接インクルードする必要はない【NGKI0486】。

3904

3905 2.14.2 TOPPERS共通データ型

3906

3907 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
 3908 タ型は次の通りである【NGKI0487】。

3909

3910	int8_t	符号付き8ビット整数 (オプション, C99準拠)
3911	uint8_t	符号無し8ビット整数 (オプション, C99準拠)
3912	int16_t	符号付き16ビット整数 (C99準拠)
3913	uint16_t	符号無し16ビット整数 (C99準拠)
3914	int32_t	符号付き32ビット整数 (C99準拠)
3915	uint32_t	符号無し32ビット整数 (C99準拠)
3916	int64_t	符号付き64ビット整数 (オプション, C99準拠)
3917	uint64_t	符号無し64ビット整数 (オプション, C99準拠)
3918	int128_t	符号付き128ビット整数 (オプション, C99準拠)
3919	uint128_t	符号無し128ビット整数 (オプション, C99準拠)
3920		
3921	int_least8_t	8ビット以上の符号付き整数 (C99準拠)
3922	uint_least8_t	int_least8_t型と同じサイズの符号無し整数 (C99準拠)
3923		
3924	float32_t	IEEE754準拠の32ビット単精度浮動小数点数 (オプション)
3925	double64_t	IEEE754準拠の64ビット倍精度浮動小数点数 (オプション)
3926		
3927	bool_t	真偽値 (trueまたはfalse)
3928	int_t	16ビット以上の符号付き整数
3929	uint_t	int_t型と同じサイズの符号無し整数
3930	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
3931	ulong_t	long_t型と同じサイズの符号無し整数
3932		
3933	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)
3934	uintptr_t	intptr_t型と同じサイズの符号無し整数 (C99準拠)
3935		
3936	FN	機能コード (符号付き整数, int_tに定義)
3937	ER	正常終了 (E_OK) またはエラーコード (符号付き整数, int_t 3938 に定義)
3939	ID	オブジェクトのID番号 (符号付き整数, int_tに定義)
3940	ATR	オブジェクト属性 (符号無し整数, uint_tに定義)
3941	STAT	オブジェクトの状態 (符号無し整数, uint_tに定義)
3942	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)
3943	PRI	優先度 (符号付き整数, int_tに定義)
3944	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 3945 サイズの符号無し整数型に定義)
3946		
3947	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)
3948	RELTIM	相対時間 (符号無し整数, 単位はミリ秒, uint_tに定義)
3949	SYSTIM	システム時刻 (符号無し整数, 単位はミリ秒, ulong_tに定義)
3950	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,

3951 ulong_tに定義)

3952

3953 FP プログラムの起動番地 (型の定まらない関数ポインタ)

3954

3955 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)

3956 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,

3957 負のID番号は格納できない)

3958 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

3959 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

3960 より1ビット短い)

3961

3962 MB_T オブジェクト管理領域を確保するためのデータ型

3963

3964 ACPTN アクセス許可パターン (符号無し32ビット整数, uint32_tに

3965 定義)

3966 ACVCT アクセス許可ベクタ

3967

3968 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。

3969 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。

3970

3971 int8_t, uint8_t, int64_t, uint64_t, int128_t, uint128_t, float32_t,

3972 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

3973 れらが使用できるかどうかは、それぞれ、INT8_MAX, UINT8_MAX, INT64_MAX,

3974 UINT64_MAX, INT128_MAX, UINT128_MAX, FLOAT32_MAX, DOUBLE64_MAXがマクロ

3975 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

3976 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、

3977 float32_tとdouble64_tは使用できないものとする【NGKI0490】。

3978

3979 【μITRON4.0仕様との関係】

3980

3981 B, UB, H, UH, W, UW, D, UD, VP_INTに代えて、C99準拠のint8_t, uint8_t,

3982 int16_t, uint16_t, int32_t, uint32_t, int64_t, uint64_t, intptr_tを用い

3983 ることにした。また、uintptr_t, int128_t, uint128_tを用意することにした。

3984

3985 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

3986 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

3987 は用意しないことにした。

3988

3989 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。

3990 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

3991 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

3992 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

3993 た。int_least16_t, uint_least16_t, int_least32_t, uint_least32_tを導入

3994 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、

3995 それぞれint16_t, uint16_t, int32_t, uint32_tで代用できるためである。

3996

3997 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

3998 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

3999 の64ビットを表す型としてdouble64_tを導入した。

4000

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

2.14.3 TOPPERS共通定数

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

(1) 一般定数【NGKI0491】

NULL		無効ポインタ
true	1	真
false	0	偽
E_OK	0	正常終了

【μITRON4.0仕様との関係】

BOOLをbool_tに代えたことから，TRUEおよびFALSEに代えて，trueおよびfalseを用いることにした．

(2) 整数型に格納できる最大値と最小値【NGKI0492】

INT8_MAX	int8_tに格納できる最大値（オプション，C99準拠）
INT8_MIN	int8_tに格納できる最小値（オプション，C99準拠）
UINT8_MAX	uint8_tに格納できる最大値（オプション，C99準拠）
INT16_MAX	int16_tに格納できる最大値（C99準拠）
INT16_MIN	int16_tに格納できる最小値（C99準拠）
UINT16_MAX	uint16_tに格納できる最大値（C99準拠）
INT32_MAX	int32_tに格納できる最大値（C99準拠）
INT32_MIN	int32_tに格納できる最小値（C99準拠）
UINT32_MAX	uint32_tに格納できる最大値（C99準拠）
INT64_MAX	int64_tに格納できる最大値（オプション，C99準拠）
INT64_MIN	int64_tに格納できる最小値（オプション，C99準拠）
UINT64_MAX	uint64_tに格納できる最大値（オプション，C99準拠）
INT128_MAX	int128_tに格納できる最大値（オプション，C99準拠）
INT128_MIN	int128_tに格納できる最小値（オプション，C99準拠）
UINT128_MAX	uint128_tに格納できる最大値（オプション，C99準拠）
INT_LEAST8_MAX	int_least8_tに格納できる最大値（C99準拠）
INT_LEAST8_MIN	int_least8_tに格納できる最小値（C99準拠）
UINT_LEAST8_MAX	uint_least8_tに格納できる最大値（C99準拠）
INT_MAX	int_tに格納できる最大値（C90準拠）
INT_MIN	int_tに格納できる最小値（C90準拠）
UINT_MAX	uint_tに格納できる最大値（C90準拠）
LONG_MAX	long_tに格納できる最大値（C90準拠）
LONG_MIN	long_tに格納できる最小値（C90準拠）
ULONG_MAX	ulong_tに格納できる最大値（C90準拠）

4051	FLOAT32_MIN	float32_tに格納できる最小の正規化された正の浮
4052		動小数点数 (オプション)
4053	FLOAT32_MAX	float32_tに格納できる表現可能な最大の有限浮動
4054		小数点数 (オプション)
4055	DOUBLE64_MIN	double64_tに格納できる最小の正規化された正の浮
4056		動小数点数 (オプション)
4057	DOUBLE64_MAX	double64_tに格納できる表現可能な最大の有限浮動
4058		小数点数 (オプション)

4059

4060 (3) 整数型のビット数 【NGKI0493】

4061

4062	CHAR_BIT	char型のビット数 (C90準拠)
------	----------	--------------------

4063

4064 (4) オブジェクト属性 【NGKI0494】

4065

4066	TA_NULL	0U	オブジェクト属性を指定しない
------	---------	----	----------------

4067

4068 (5) タイムアウト指定 【NGKI0495】

4069

4070	TMO_POL	0	ポーリング
4071	TMO_FEVR	-1	永久待ち
4072	TMO_NBLK	-2	ノンブロッキング

4073

4074 (6) アクセス許可パターン 【NGKI0496】

4075

4076	TACP_KERNEL	0U	カーネルドメインのみにアクセスを許可
4077	TACP_SHARED	~0U	すべての保護ドメインにアクセスを許可

4078

4079 2.14.4 TOPPERS共通エラーコード

4080

4081 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
 4082 【NGKI0497】 .

4083

4084 (A) 内部エラークラス (EC_SYS, -5~-8)

4085

4086	E_SYS	-5	システムエラー
------	-------	----	---------

4087

4088 (B) 未サポートエラークラス (EC_NOSPT, -9~-16)

4089

4090	E_NOSPT	-9	未サポート機能
4091	E_RSFN	-10	予約機能コード
4092	E_RSATR	-11	予約属性

4093

4094 (C) パラメータエラークラス (EC_PAR, -17~-24)

4095

4096	E_PAR	-17	パラメータエラー
4097	E_ID	-18	不正ID番号

4098

4099 (D) 呼出しコンテキストエラークラス (EC_CTX, -25~-32)

4100

4101	E_CTX	-25	コンテキストエラー
4102	E_MACV	-26	メモリアクセス違反
4103	E_OACV	-27	オブジェクトアクセス違反
4104	E_ILUSE	-28	サービスコール不正使用

4105

4106 (E) 資源不足エラークラス (EC_NOMEM, -33~-40)

4107

4108	E_NOMEM	-33	メモリ不足
4109	E_NOID	-34	ID番号不足
4110	E_NORES	-35	資源不足

4111

4112 (F) オブジェクト状態エラークラス (EC_OBJ, -41~-48)

4113

4114	E_OBJ	-41	オブジェクト状態エラー
4115	E_NOEXS	-42	オブジェクト未登録
4116	E_QOVR	-43	キューイングオーバーフロー

4117

4118 (G) 待ち解除エラークラス (EC_RLWAI, -49~-56)

4119

4120	E_RLWAI	-49	待ち禁止状態または待ち状態の強制解除
4121	E_TMOUT	-50	ポーリング失敗またはタイムアウト
4122	E_DLT	-51	待ちオブジェクトの削除または再初期化
4123	E_CLS	-52	待ちオブジェクトの状態変化

4124

4125 (H) 警告クラス (EC_WARN, -57~-64)

4126

4127	E_WBLK	-57	ノンブロッキング受け付け
4128	E_BOVR	-58	バッファオーバーフロー

4129

4130 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、
4131 [NGKI0019] の原則では例外としている。

4132

4133 【 μ ITRON4.0仕様との関係】

4134

4135 E_NORESは、 μ ITRON4.0仕様に規定されていないエラーコードである。

4136

4137 2.14.5 TOPPERS共通マクロ

4138

4139 (1) 整数定数を作るマクロ 【NGKI0498】

4140

4141	INT8_C(val)	int_least8_t型の定数を作るマクロ (C99準拠)
4142	UINT8_C(val)	uint_least8_t型の定数を作るマクロ (C99準拠)
4143	INT16_C(val)	int16_t型の定数を作るマクロ (C99準拠)
4144	UINT16_C(val)	uint16_t型の定数を作るマクロ (C99準拠)
4145	INT32_C(val)	int32_t型の定数を作るマクロ (C99準拠)
4146	UINT32_C(val)	uint32_t型の定数を作るマクロ (C99準拠)
4147	INT64_C(val)	int64_t型の定数を作るマクロ (オプション, C99準拠)
4148	UINT64_C(val)	uint64_t型の定数を作るマクロ (オプション, C99準拠)
4149	INT128_C(val)	int128_t型の定数を作るマクロ (オプション, C99準拠)
4150	UINT128_C(val)	uint128_t型の定数を作るマクロ (オプション, C99準拠)

4151
 4152 UINT_C(val) uint_t型の定数を作るマクロ
 4153 ULONG_C(val) ulong_t型の定数を作るマクロ
 4154
 4155 **【仕様決定の理由】**
 4156
 4157 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
 4158 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
 4159 ロを使って記述する必要はない．
 4160
 4161 (2) 型に関する情報を取り出すためのマクロ **【NGKI0499】**
 4162
 4163 offsetof(structure, field) 構造体structure中のフィールドfieldの
 4164 バイト位置を返すマクロ (C90準拠)
 4165
 4166 alignof(type) 型typeのアラインメント単位を返すマクロ
 4167
 4168 ALIGN_TYPE(addr, type) 番地addrが型typeに対してアラインしてい
 4169 るかどうかを返すマクロ
 4170
 4171 (3) assertマクロ **【NGKI0500】**
 4172
 4173 assert(exp) expが成立しているかを検査するマクロ (C90準拠)
 4174
 4175 (4) コンパイラの拡張機能のためのマクロ **【NGKI0501】**
 4176
 4177 inline インライン関数
 4178 Inline ファイルローカルなインライン関数
 4179 asm インラインアセンブラ
 4180 Asm インラインアセンブラ (最適化抑止)
 4181 throw() 例外を発生しない関数
 4182 NoReturn リターンしない関数
 4183
 4184 (5) エラーコード構成・分解マクロ **【NGKI0502】**
 4185
 4186 ERCD(mercd, sercd) メインエラーコードmercdとサブエラーコードsercdか
 4187 ら、エラーコードを構成するためのマクロ
 4188
 4189 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
 4190 ためのマクロ
 4191 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
 4192 めのマクロ
 4193
 4194 (6) アクセス許可パターン構成マクロ **【NGKI0503】**
 4195
 4196 TACP(domid) domidで指定されるユーザドメインのみにアクセスを
 4197 許可するアクセス許可パターンを構成するためのマ
 4198 クロ
 4199
 4200 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

4201 することができる【NGKI0504】. TDOM_SELF, TDOM_KERNEL, TDOM_NONEを指定し
4202 た場合の動作は, 保証されない【NGKI0505】.

4203

4204 2.14.6 TOPPERS共通構成マクロ

4205

4206 (1) 相対時間の範囲【NGKI0506】

4207

4208 TMAX_RELTIM 相対時間に指定できる最大値

4209

4210 2.15 カーネル共通定義

4211

4212 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.

4213

4214 2.15.1 カーネルヘッダファイル

4215

4216 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)
4217 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
4218 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
4219 【NGKI0508】.

4220

4221 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
4222 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)
4223 に含まれる【NGKI0509】. 具体的には, 登録できるオブジェクトの数
4224 (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
4225 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
4226 ドする【NGKI0510】.

4227

4228 μ ITRON4.0仕様に規定されており, この仕様で廃止されたデータ型および定数
4229 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
4230 する【NGKI0511】.

4231

4232 【 μ ITRON4.0仕様との関係】

4233

4234 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
4235 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
4236 が含まれることとした. これに伴い, ヘッダファイルの名称を, μ ITRON4.0仕
4237 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
4238 化ヘッダファイル (kernel_cfg.h) に変更した.

4239

4240 2.15.2 カーネル共通定数

4241

4242 (1) オブジェクト属性【NGKI0512】

4243

4244 TA_TPRI 0x01U タスクの待ち行列をタスクの優先度順に

4245

4246 【 μ ITRON4.0仕様との関係】

4247

4248 値が0のオブジェクト属性 (TA_HLNG, TA_TFIFO, TA_MFIFO, TA_WSGL) は, デフォ
4249 ルトの扱いにして廃止した. これは, 「(tskatr & TA_HLNG) != 0U」のような
4250 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.

4251 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
4252 した。

4253

4254 (2) 保護ドメインID 【NGKI0513】

4255

4256	TDOM_SELF	0	自タスクの属する保護ドメイン
4257	TDOM_KERNEL	-1	カーネルドメイン
4258	TDOM_NONE	-2	無所属（保護ドメインに属さない）

4259

4260 (3) その他のカーネル共通定数 【NGKI0514】

4261

4262	TCLS_SELF	0	自タスクの属するクラス
4263			
4264	TPRC_NONE	0	割付けプロセッサの指定がない
4265	TPRC_INI	0	初期割付けプロセッサ
4266			
4267	TSK_SELF	0	自タスク指定
4268	TSK_NONE	0	該当するタスクがない
4269			
4270	TPRI_SELF	0	自タスクのベース優先度の指定
4271	TPRI_INI	0	タスクの起動時優先度の指定
4272			
4273	TIPM_ENAALL	0	割込み優先度マスク全解除

4274

4275 (4) カーネルで用いるメインエラーコード

4276

4277 「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
4278 で、E_CLS、E_WBLK、E_BOVRの3つは、カーネルでは使用しない【NGKI0515】。

4279

4280 【TOPPERS/ASPカーネルにおける規定】

4281

4282 ASPカーネルでは、サービスコールから、E_RSFN、E_RSATR、E_MACV、E_OACV、
4283 E_NOMEM、E_NOID、E_NORES、E_NOEXSが返る状況は起こらない【ASPS0011】。
4284 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
4285 的生成機能拡張パッケージでは、E_RSATR、E_NOMEM、E_NOID、E_NOEXSが返る状
4286 況が起こる【ASPS0013】。

4287

4288 【TOPPERS/FMPカーネルにおける規定】

4289

4290 FMPカーネルでは、サービスコールから、E_RSFN、E_RSATR、E_MACV、E_OACV、
4291 E_NOMEM、E_NOID、E_NORES、E_NOEXSが返る状況は起こらない【FMPS0007】。
4292 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】。

4293

4294 【TOPPERS/HRP2カーネルにおける規定】

4295

4296 HRP2カーネルでは、サービスコールから、E_RSATR、E_NOID、E_NORES、
4297 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
4298 タによって検出される【HRPS0007】。

4299

4300 【TOPPERS/SSPカーネルにおける規定】

4301
 4302 SSPカーネルでは、サービスコールから、E_RSFN, E_RSATR, E_MACV, E_OACV,
 4303 E_ILUSE, E_NOMEM, E_NOID, E_NORES, E_NOEXS, E_RLWAI, E_TMOUT, E_DLTが返
 4304 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
 4305 出される【SSPS0009】。
 4306
 4307 2.15.3 カーネル共通マクロ
 4308
 4309 (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
 4310
 4311 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
 4312 用意している【NGKI0516】。
 4313
 4314 STK_T スタック領域を確保するためのデータ型
 4315
 4316 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
 4317 STK_T型の配列の要素数
 4318 ROUND_STK_T(sz) 要素数COUNT_STK_T(sz)のSTK_T型の配列のサイズ (sz
 4319 を、STK_T型のサイズの倍数になるように大きい方に
 4320 丸めた値)
 4321
 4322 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】。
 4323
 4324 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>)];
 4325
 4326 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
 4327 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
 4328 サイズにROUND_STK_T(<スタック領域のサイズ>)を指定する【NGKI0518】。
 4329
 4330 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
 4331 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
 4332 管理機能」の節のCRE_TSKの機能の項を参照すること。
 4333
 4334 (2) オブジェクト属性を作るマクロ
 4335
 4336 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
 4337 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
 4338 【NGKI0520】。
 4339
 4340 TA_DOM(domid) domidで指定される保護ドメインに属する
 4341
 4342 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
 4343 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
 4344 【NGKI0521】。
 4345
 4346 TA_CLS(clsid) clsidで指定されるクラスに属する
 4347
 4348 (3) サービスコールの呼出し方法を指定するマクロ
 4349
 4350 保護機能対応カーネルでは、サービスコールの呼出し方法を指定するためのマ

クロとして、次のマクロを用意している【NGKI0522】。

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

2.15.4 カーネル共通構成マクロ

(1) サポートする機能【NGKI0523】

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

【未決定事項】

マクロ名は、今後変更する可能性がある。

(2) 優先度の範囲【NGKI0524】

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

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、タスク優先度の最大値 (TMAX_TPRI) は16に固定されている【ASPS0014】。ただし、タスク優先度拡張パッケージを用いると、TMAX_TPRIを256に拡張することができる【ASPS0015】。

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、タスク優先度の最大値 (TMAX_TPRI) は16に固定されている【FMPS0009】。

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、タスク優先度の最大値 (TMAX_TPRI) は16に固定されている【HRPS0008】。

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、タスク優先度の最大値 (TMAX_TPRI) は16に固定されている【SSPS0010】。

【 μ ITRON4.0仕様との関係】

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

(3) プロセッサの数

4401 マルチプロセッサ対応カーネルでは、プロセッサの数を知らするためのマクロとし
4402 て、次の構成マクロを用意している【NGKI0525】。

4403
4404 TNUM_PRCID プロセッサの数

4405
4406 (4) 特殊な役割を持ったプロセッサ

4407
4408 マルチプロセッサ対応カーネルでは、特殊な役割を持ったプロセッサを知るた
4409 めのマクロとして、次の構成マクロを用意している【NGKI0526】。

4410
4411 TOPPERS_MASTER_PRCID マスタプロセッサのID番号
4412 TOPPERS_SYSTM_PRCID システム時刻管理プロセッサのID番号（グ
4413 ローバルタイマ方式の場合のみ）

4414
4415 (5) タイマ方式

4416
4417 マルチプロセッサ対応カーネルでは、システム時刻の方式を知るためのマクロ
4418 として、次の構成マクロを用意している【NGKI0527】。

4419
4420 TOPPERS_SYSTM_LOCAL ローカルタイマ方式の場合にマクロ定義
4421 TOPPERS_SYSTM_GLOBAL グローバルタイマ方式の場合にマクロ定義

4422
4423 (6) バージョン情報【NGKI0528】

4424
4425 TKERNEL_MAKER カーネルのメーカコード（=0x0118）
4426 TKERNEL_PRID カーネルの識別番号
4427 TKERNEL_SPVER カーネル仕様のバージョン番号
4428 TKERNEL_PRVER カーネルのバージョン番号

4429
4430 カーネルのメーカコード（TKERNEL_MAKER）は、TOPPERSプロジェクトから配布
4431 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。

4432
4433 カーネルの識別番号（TKERNEL_PRID）は、TOPPERSカーネルの種類を表す。

4434
4435 0x0001 TOPPERS/JSPカーネル
4436 0x0002 予約（IIMPカーネル）
4437 0x0003 予約（IDLカーネル）
4438 0x0004 TOPPERS/FI4カーネル
4439 0x0005 TOPPERS/FDMPカーネル
4440 0x0006 TOPPERS/HRPカーネル
4441 0x0007 TOPPERS/ASPカーネル
4442 0x0008 TOPPERS/FMPカーネル
4443 0x0009 TOPPERS/SSPカーネル
4444 0x000a TOPPERS/ASP Safetyカーネル

4445
4446 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
4447 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
4448 番号、下位4ビットがマイナーバージョン番号を表す。

4449
4450 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

4451 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
4452 ベルを表す.

4453

4454

4455 第3章 システムインタフェースレイヤAPI仕様

4456

4457 3.1 システムインタフェースレイヤの概要

4458

4459 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
4460 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
4461 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
4462 トにおいて修正を加えて用いている.

4463

4464 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
4465 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
4466 からSILの機能呼び出した場合の動作は, 次の例外を除いては保証されない
4467 【NGKI0802】.

4468

- 4469 ・微少時間待ちの機能呼び出すこと
- 4470 ・エンディアンの取得のためのマクロを参照すること
- 4471 ・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
4472 スすること
- 4473 ・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
4474 こと

4475

4476 3.2 SILヘッダファイル

4477

4478 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
4479 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
4480 は, SILヘッダファイルをインクルードする【NGKI0804】.

4481

4482 3.3 全割込みロック状態の制御

4483

4484 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)
4485 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
4486 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)
4487 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.

4488

4489 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
4490 ための以下の機能を用意している.

4491

4492 (1) SIL_PRE_LOC

4493

4494 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
4495 は, 型と変数名を並べたもので, 最後に";"を含まない.

4496

4497 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
4498 の変数宣言部に記述しなければならない【NGKI0806】. SIL_LOC_INT,
4499 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
4500 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

4501 SIL_PRE_LOCを記述しなければならない【NGKI0807】。そのように記述しなかつ
4502 た場合の動作は保証されない【NGKI0808】。

4503
4504 (2) SIL_LOC_INT()

4505
4506 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
4507 し、全割込みロック状態に遷移する【NGKI0809】。

4508
4509 (3) SIL_UNL_INT()

4510
4511 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
4512 【NGKI0810】。SIL_LOC_INTを実行せずにSIL_UNL_INTを呼び出した場合の動作
4513 は保証されない【NGKI0811】。

4514
4515 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
4516 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
4517 ること。

4518
4519 【補足説明】

4520
4521 全割込みロック状態の制御機能の使用例は次の通り。

```
4522 {  
4523     SIL_PRE_LOC;  
4524  
4525     SIL_LOC_INT();  
4526     // この間はNMIを除くすべての割込みがマスクされる。  
4527     // 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）。  
4528     SIL_UNL_INT();  
4529 }  
4530
```

4531
4532 3.4 SILスピンロック

4533
4534 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
4535 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
4536 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
4537 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、
4538 SILスピンロックと呼ぶ。

4539
4540 プロセッサ間ロックを取得している間は、全割込みロック状態にすることで
4541 すべての割込み（NMIを除く）がマスクされる【NGKI0812】。ロックが他のプロセッ
4542 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
4543 【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
4544 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）
4545 【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
4546 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
4547 スク解除することを、SILスピンロックを返却するという。

4548
4549 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
4550 【NGKI0815】。

4551
4552 (1) SIL_PRE_LOC
4553
4554 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
4555 ロックの取得・解放にも兼用する【NGKI0816】。
4556
4557 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
4558 の変数宣言部に記述しなければならない【NGKI0817】。SIL_LOC_SPN、
4559 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
4560 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
4561 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
4562 【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
4563 【NGKI0819】。
4564
4565 (2) SIL_LOC_SPN()
4566
4567 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
4568 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
4569 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
4570 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
4571 【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
4572 トし、全割込みロック状態に遷移する【NGKI0822】。
4573
4574 (3) SIL_UNL_SPN()
4575
4576 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
4577 実行する前の状態に戻す【NGKI0823】。
4578
4579 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
4580 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
4581 SIL_UNL_SPNを呼び出した場合の動作も保証されない【NGKI0825】。
4582
4583 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
4584 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
4585 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。
4586
4587 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
4588 されていない【NGKI0826】。
4589
4590 【使用上の注意】
4591
4592 全割込ロック状態やCPUロック状態でSIL_LOC_SPNを呼び出すことはできるが、
4593 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
4594 べきである。
4595
4596 【補足説明】
4597
4598 SILスピンロック機能の使用例は次の通り。
4599
4600 {

```
4601         SIL_PRE_LOC;
4602
4603         SIL_LOC_SPN();
4604         // この間はSILスピンロックを取得している.
4605         // この間はNMIを除くすべての割込みがマスクされる.
4606         // 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）.
4607         SIL_UNL_SPN();
4608     }
```

4610 3.5 微少時間待ち

4611
4612 デバイスをアクセスする際に、微少な時間待ちを入れなければならない場合がある。そのような場合に、NOP命令をいくつか入れるなどの方法で対応すると、ポータビリティを損なうことになる。そこで、SILでは、微少な時間待ちを行うための以下の機能を用意している。

4616
4617 (1) void sil_dly_nse(ulong_t dlytim)

4618
4619 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
4620 【NGKI0827】。指定した値によっては、指定した時間よりもかなり長く待つ場合があるので注意すること。

4623 3.6 エンディアンの取得

4624
4625 プロセッサのバイトエンディアンを取得するためのマクロとして、SILでは、以下のマクロを定義している。

4628 (1) SIL_ENDIAN_BIG, SIL_ENDIAN_LITTLE

4629
4630 ビッグエンディアンプロセッサではSIL_ENDIAN_BIGを、リトルエンディアンプロセッサではSIL_ENDIAN_LITTLEを、マクロ定義している【NGKI0828】。

4633 3.7 メモリ空間アクセス関数

4634
4635 メモリ空間にマッピングされたデバイスレジスタや、デバイスとの共有メモリをアクセスするために、SILでは、以下の関数を用意している。

4638 (1) uint8_t sil_reb_mem(const uint8_t *mem)

4639
4640 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】。

4642 (2) void sil_wrb_mem(uint8_t *mem, uint8_t data)

4643
4644 memで指定されるアドレスにdataで指定される値を8ビット単位で書き込む【NGKI0830】。

4647 (3) uint16_t sil_reh_mem(const uint16_t *mem)

4648
4649 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】。

4650

4651 (4) void sil_wrh_mem(uint16_t *mem, uint16_t data)
4652
4653 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
4654 【NGKI0832】.
4655
4656 (5) uint16_t sil_reh_lem(const uint16_t *mem)
4657
4658 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
4659 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
4660 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
4661 アン変換した値を返す.
4662
4663 (6) void sil_wrh_lem(uint16_t *mem, uint16_t data)
4664
4665 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
4666 アンで書き込む【NGKI0834】. リトルエンディアンプロセッサでは,
4667 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
4668 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.
4669
4670 (7) uint16_t sil_reh_bem(const uint16_t *mem)
4671
4672 memで指定されるアドレスから16ビット単位でビッグエンディアンで読み出した
4673 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
4674 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
4675 アン変換した値を返す.
4676
4677 (8) void sil_wrh_bem(uint16_t *mem, uint16_t data)
4678
4679 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
4680 アンで書き込む【NGKI0836】. ビッグエンディアンプロセッサでは,
4681 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
4682 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.
4683
4684 (9) uint32_t sil_rew_mem(const uint32_t *mem)
4685
4686 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】.
4687
4688 (10) void sil_wrw_mem(uint32_t *mem, uint32_t data)
4689
4690 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
4691 【NGKI0838】.
4692
4693 (11) uint32_t sil_rew_lem(const uint32_t *mem)
4694
4695 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
4696 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
4697 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
4698 アン変換した値を返す.
4699
4700 (12) void sil_wrw_lem(uint32_t *mem, uint32_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32ビット単位でリトルエンディ
4703 アンで書き込む【NGKI0840】。リトルエンディアンプロセッサでは、
4704 sil_rwr_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
4705 アン変換した値を、sil_rwr_memで書き込むのと同じ結果となる。

4706
4707 (13) uint32_t sil_rwr_bem(const uint32_t *mem)

4708
4709 memで指定されるアドレスから32ビット単位でビッグエンディアンで読み出した
4710 値を返す【NGKI0841】。ビッグエンディアンプロセッサでは、sil_rwr_memと一
4711 致する。リトルエンディアンプロセッサでは、sil_rwr_memが返す値を、エンディ
4712 アン変換した値を返す。

4713
4714 (14) void sil_rwr_bem(uint32_t *mem, uint32_t data)

4715
4716 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
4717 アンで書き込む【NGKI0842】。ビッグエンディアンプロセッサでは、
4718 sil_rwr_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
4719 アン変換した値を、sil_rwr_memで書き込むのと同じ結果となる。

4720 3.8 I/O空間アクセス関数

4721
4722 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
4723 ジスタにアクセスするために、メモリ空間アクセス関数と同等の以下の関数を
4724 用意している【NGKI0843】。

- 4725
4726
4727 (1) uint8_t sil_reb_iop(const uint8_t *iop)
4728 (2) void sil_wrb_iop(uint8_t *iop, uint8_t data)
4729 (3) uint16_t sil_reh_iop(const uint16_t *iop)
4730 (4) void sil_wrh_iop(uint16_t *iop, uint16_t data)
4731 (5) uint16_t sil_reh_lep(const uint16_t *iop)
4732 (6) void sil_wrh_lep(uint16_t *iop, uint16_t data)
4733 (7) uint16_t sil_reh_bep(const uint16_t *iop)
4734 (8) void sil_wrh_bep(uint16_t *iop, uint16_t data)
4735 (9) uint32_t sil_rwr_iop(const uint32_t *iop)
4736 (10) void sil_rwr_iop(uint32_t *iop, uint32_t data)
4737 (11) uint32_t sil_rwr_lep(const uint32_t *iop)
4738 (12) void sil_rwr_lep(uint32_t *iop, uint32_t data)
4739 (13) uint32_t sil_rwr_bep(const uint32_t *iop)
4740 (14) void sil_rwr_bep(uint32_t *iop, uint32_t data)

4741 3.9 プロセッサIDの参照

4742
4743 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
4744 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。

4745
4746 (1) void sil_get_pid(ID *p_prcid)

4747
4748 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、
4749 p_prcidで指定したメモリ領域に返す【NGKI0844】。

【使用上の注意】

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

第4章 カーネルAPI仕様

この章では、カーネルのAPI仕様について規定する。

【 μ ITRON4.0仕様との関係】

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

`INT` \rightarrow `int_t`
`UINT` \rightarrow `uint_t`
`VP` \rightarrow `void *`
`VP_INT` \rightarrow `intptr_t`

【 μ ITRON4.0/PX仕様との関係】

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

4.1 タスク管理機能

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

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

- ・タスク属性
- ・タスク状態
- ・ベース優先度
- ・現在優先度
- ・起動要求キューイング数
- ・割付けプロセッサ（マルチプロセッサ対応カーネルの場合）
- ・次回起動時の割付けプロセッサ（マルチプロセッサ対応カーネルの場合）
- ・拡張情報
- ・メインルーチンの先頭番地
- ・起動時優先度
- ・実行時優先度（TOPPERS/SSPカーネルの場合）

4801 ・スタック領域
4802 ・システムスタック領域（保護機能対応カーネルの場合）
4803 ・アクセス許可ベクタ（保護機能対応カーネルの場合）
4804 ・属する保護ドメイン（保護機能対応カーネルの場合）
4805 ・属するクラス（マルチプロセッサ対応カーネルの場合）
4806
4807 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
4808 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】。
4809
4810 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
4811 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
4812 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
4813 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
4814 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。
4815
4816 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
4817 であり、タスクの生成時に0に初期化される【NGKI1005】。
4818
4819 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
4820 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
4821 初期割付けプロセッサに初期化される【NGKI1006】。
4822
4823 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、
4824 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
4825 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
4826 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
4827 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
4828 【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
4829 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
4830 同じプロセッサで実行される）【NGKI1009】。
4831
4832 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
4833 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
4834 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
4835 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
4836 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
4837 用いるオブジェクト管理領域と同様に扱われる【NGKI1011】。
4838
4839 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
4840 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
4841 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
4842 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】。
4843
4844 タスク属性には、次の属性を指定することができる【NGKI1014】。
4845
4846 TA_ACT 0x02U タスクの生成時にタスクを起動する
4847 TA_RSTR 0x04U 生成するタスクを制約タスクとする
4848
4849 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
4850 【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

4851 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
4852 の属性を予約している【NGKI1017】。

4853
4854 TA_FPU FPUレジスタをコンテキストに含める

4855

4856 C言語によるタスクの記述形式は次の通り【NGKI1018】。

4857

4858 void task(intptr_t exinf)

4859 {

4860 タスク本体

4861 ext_tsk();

4862 }

4863

4864 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、
4865 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
4866 じ動作をする【NGKI1020】。

4867

4868 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。

4869

4870 TMAX_ACTCNT タスクの起動要求キューイング数の最大値【NGKI1021】

4871

4872 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
4873 は、静的APIによって登録されたタスクの数に一致）

4874 【NGKI1022】

4875

4876 【TOPPERS/ASPカーネルにおける規定】

4877

4878 ASPカーネルでは、TMAX_ACTCNTは1に固定されている【ASPS0101】。また、制約
4879 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
4880 ジを用いると、制約タスクの機能を追加することができる【ASPS0103】。

4881

4882 【TOPPERS/FMPカーネルにおける規定】

4883

4884 FMPカーネルでは、TMAX_ACTCNTは1に固定されている【FMPS0101】。また、制約
4885 タスクはサポートしていない【FMPS0102】。

4886

4887 【TOPPERS/HRP2カーネルにおける規定】

4888

4889 HRP2カーネルでは、TMAX_ACTCNTは1に固定されている【HRPS0101】。また、制
4890 約タスクはサポートしていない【HRPS0102】。

4891

4892 【TOPPERS/SSPカーネルにおける規定】

4893

4894 SSPカーネルでは、TMAX_ACTCNTは1に固定されている【SSPS0101】。

4895

4896 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
4897 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

4898 【SSPS0102】。

4899

4900 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

4901 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
 4902 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
 4903 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
 4904 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
 4905 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.

4906

4907 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
 4908 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
 4909 先度は、起動時優先度と同じ値に設定される【SSPS0106】.

4910

4911 [実行時優先度によるスタック領域の節約]

4912

4913 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
 4914 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、
 4915 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
 4916 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.

4917

4918 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
 4919 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
 4920 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
 4921 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
 4922 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、
 4923 スタック領域を節約できることになる.

4924

4925 【μITRON4.0仕様との関係】

4926

4927 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
 4928 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) ,
 4929 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
 4930 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.

4931

4932 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.

4933

4934 CRE_TSK タスクの生成 [S] 【NGKI1023】

4935 acre_tsk タスクの生成 [TD] 【NGKI1024】

4936

4937 【静的API】

4938 *保護機能対応でないカーネルの場合

4939 CRE_TSK(ID tskid, { ATR tskatr, intptr_t exinf, TASK task,
 4940 PRI itskpri, SIZE stksz, STK_T *stk })

4941

4942 *保護機能対応カーネルの場合

4943 CRE_TSK(ID tskid, { ATR tskatr, intptr_t exinf, TASK task,
 4944 PRI itskpri, SIZE stksz, STK_T *stk, SIZE sstksz, STK_T *sstk })

4945 ※ sstkszおよびsstkの記述は省略することができる【NGKI1025】.

4946

4947 【C言語API】

4948 ER_ID tskid = acre_tsk(const T_CTSK *pk_ctsk)

4949

4950 【パラメータ】

4951	ID	tskid	生成するタスクのID番号 (CRE_TSKの場合)
4952	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)
4953			
4954			
4955	*タスクの生成情報 (パケットの内容)		
4956	ATR	tskatr	タスク属性
4957	intptr_t	exinf	タスクの拡張情報
4958	TASK	task	タスクのメインルーチンの先頭番地
4959	PRI	itskpri	タスクの起動時優先度
4960	SIZE	stksz	タスクのスタック領域のサイズ (バイト数)
4961	STK_T *	stk	タスクのスタック領域の先頭番地
4962	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)
4963			
4964			
4965	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)
4966			
4967			
4968			
4969	【リターンパラメータ】		
4970	ER_ID	tskid	生成されたタスクのID番号 (正の値) またはエラーコード
4971			
4972			
4973	【エラーコード】		
4974	E_CTX	コンテキストエラー	
4975		・非タスクコンテキストからの呼出し [s] 【NGKI1026】	
4976		・CPUロック状態からの呼出し [s] 【NGKI1027】	
4977	E_RSATR	予約属性	
4978		・tskatrが無効 【NGKI1028】	
4979		・属する保護ドメインの指定が有効範囲外または無所属 [sP] 【NGKI1029】	
4980			
4981		・保護ドメインの囲みの中に記述されていない [SP] 【NGKI1030】	
4982		・属するクラスの指定が有効範囲外 [sM] 【NGKI1031】	
4983		・クラスの囲みの中に記述されていない [SM] 【NGKI1032】	
4984	E_PAR	パラメータエラー	
4985		・taskがプログラムの先頭番地として正しくない 【NGKI1033】	
4986		・itskpriが有効範囲外 【NGKI1034】	
4987		・その他の条件については機能の項を参照	
4988	E_OACV	オブジェクトアクセス違反	
4989		・システム状態に対する管理操作が許可されていない [sP] 【NGKI1035】	
4990			
4991	E_MACV	メモリアクセス違反	
4992		・pk_ctskが指すメモリ領域への読出しアクセスが許可されていない [sP] 【NGKI1036】	
4993			
4994	E_NOID	ID番号不足	
4995		・割り付けられるタスクIDがない [sD] 【NGKI1037】	
4996	E_NOMEM	メモリ不足	
4997		・スタック領域が確保できない 【NGKI1038】	
4998		・システムスタック領域が確保できない 【NGKI1039】	
4999	E_OBJ	オブジェクト状態エラー	
5000		・tskidで指定したタスクが登録済み (CRE_TSKの場合) 【NGKI1040】	

5001 ・その他の条件については機能の項を参照
5002
5003 **【機能】**
5004
5005 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
5006 な振舞いは以下の通り。
5007
5008 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】。
5009 stkszに0を指定した時や、ターゲット定義の最小値よりも小さい値を指定した
5010 時には、E_PARエラーとなる【NGKI1042】。また、保護機能対応カーネルで、生
5011 成するタスクがユーザタスクの場合には、sstkとsstkszからシステムスタック
5012 領域が設定される【NGKI1043】。この場合、sstkszに0を指定した時や、ターゲッ
5013 ト定義の最小値よりも小さい値を指定した時には、E_PARエラーとなる
5014 【NGKI1044】。
5015
5016 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、
5017 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
5018 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
5019 タスクは実行できる状態になる【NGKI1046】。
5020
5021 静的APIにおいては、tskidはオブジェクト識別名、tskatr, itskpri, stkszは
5022 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
5023 【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
5024 を検出することができない【NGKI1048】。
5025
5026 〔stkにNULLを指定した場合〕
5027
5028 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
5029 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
5030 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
5031 るように大きい方に丸めたサイズで確保される【NGKI1050】。
5032
5033 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
5034 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
5035 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
5036 て、メモリオブジェクトとしてカーネルに登録される【NGKI1051】。
5037
5038 静的APIにおいて、生成するタスクが制約タスクの場合（tskatrにTA_RSTRを指
5039 定した場合）、コンフィギュレータは、生成する制約タスクの起動時優先度毎
5040 にスタック領域を確保し、同じ起動時優先度を持つ制約タスクにそのスタック
5041 領域を共有させる【NGKI1052】。確保するスタック領域のサイズは、コンフィ
5042 ギュレータがスタック領域を確保し（stkにNULLを指定して生成され）、同じ起
5043 動時優先度を持つ制約タスクのスタック領域のサイズ（stksz）の最大値となる
5044 【NGKI1053】。マルチプロセッサ対応カーネルでは、以上のスタック領域の確
5045 保処理を、制約タスクの初期割付けプロセッサ毎に行う【NGKI1054】。
5046
5047 〔stkにNULL以外を指定した場合〕
5048
5049 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
5050 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

5051 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
5052 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
5053 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】。

5054

5055 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、
5056 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
5057 まれない場合、E_OBJエラーとなる【NGKI1057】。

5058

5059 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
5060 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
5061 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
5062 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
5063 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
5064 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
5065 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
5066 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
5067 【NGKI1060】。

5068

5069 [sstkとsstkszの扱い]

5070

5071 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
5072 ザタスクの場合とシステムタスクの場合で異なる。

5073

5074 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。

5075

5076 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
5077 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
5078 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
5079 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
5080 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
5081 のデフォルトのサイズで確保される【NGKI1063】。

5082

5083 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
5084 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
5085 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
5086 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
5087 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
5088 【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
5089 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】。

5090

5091 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。

5092

5093 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
5094 た場合には、E_PARエラーとなる【NGKI1068】。

5095

5096 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
5097 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
5098 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
5099 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
5100 に丸めたサイズで確保される【NGKI1070】。

5101
5102 sstk_{sz}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
5103 となる【NGKI1071】。
5104
5105 sstk_{sz}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
5106 【NGKI1072】。
5107
5108 【TOPPERS/ASPカーネルにおける規定】
5109
5110 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
5111 成機能拡張パッケージでは、acre_tskもサポートする【ASPS0105】。
5112
5113 【TOPPERS/FMPカーネルにおける規定】
5114
5115 FMPカーネルでは、CRE_TSKのみをサポートする【FMPS0103】。
5116
5117 【TOPPERS/HRP2カーネルにおける規定】
5118
5119 HRP2カーネルでは、CRE_TSKのみをサポートする【HRPS0103】。
5120
5121 【TOPPERS/SSPカーネルにおける規定】
5122
5123 SSPカーネルでは、CRE_TSKのみをサポートする【SSPS0107】。
5124
5125 SSPカーネルでは、複数のタスクに対して、同じ起動時優先度を設定することは
5126 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
5127 【SSPS0109】。
5128
5129 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
5130 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】。
5131
5132 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
5133 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、
5134 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
5135 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、
5136 E_PARエラーとなる【SSPS0112】。
5137
5138 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。
5139
5140 【μITRON4.0仕様との関係】
5141
5142 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
5143 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
5144 した。
5145
5146 【μITRON4.0/PX仕様との関係】
5147
5148 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
5149 ンで確保する方法を規定した。
5150

5151 【未決定事項】

5152

5153 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
 5154 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.

5155

5156 【仕様決定の理由】

5157

5158 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
 5159 きることにしたのは, 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
 5160 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
 5161 るためである.

5162

5163 AID_TSK 割付け可能なタスクIDの数の指定 [SD] 【NGKI1073】

5164

5165 【静的API】

5166 AID_TSK(uint_t notsk)

5167

5168 【パラメータ】

5169 uint_t notsk 割付け可能なタスクIDの数

5170

5171 【エラーコード】

5172 E_RSATR 予約属性

5173 ・保護ドメインの囲みの中に記述されていない [P] 【NGKI1074】

5174 ・クラスの囲みの中に記述されていない [M] 【NGKI1075】

5175

5176 【機能】

5177

5178 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
 5179 付け可能なタスクIDとして確保する【NGKI1076】.

5180

5181 notskは整数定数式パラメータである【NGKI1077】.

5182

5183 SAC_TSK タスクのアクセス許可ベクタの設定 [SP] 【NGKI1078】

5184 sac_tsk タスクのアクセス許可ベクタの設定 [TPD] 【NGKI1079】

5185

5186 【静的API】

5187 SAC_TSK(ID tskid, { ACPTN acptn1, ACPTN acptn2,
 5188 ACPTN acptn3, ACPTN acptn4 })

5189

5190 【C言語API】

5191 ER ercd = sac_tsk(ID tskid, const ACVCT *p_acvct)

5192

5193 【パラメータ】

5194 ID tskid 対象タスクのID番号

5195 ACVCT * p_acvct アクセス許可ベクタを入れたパケットへのポ
 5196 インタ (静的APIを除く)

5197

5198 *アクセス許可ベクタ (パケットの内容)

5199 ACPTN acptn1 通常操作1のアクセス許可パターン

5200 ACPTN acptn2 通常操作2のアクセス許可パターン

5201	ACPTN	acptn3	管理操作のアクセス許可パターン
5202	ACPTN	acptn4	参照操作のアクセス許可パターン
5203			
5204	【リターンパラメータ】		
5205	ER	ercd	正常終了 (E_OK) またはエラーコード
5206			
5207	【エラーコード】		
5208	E_CTX	コンテキストエラー	
5209			・ 非タスクコンテキストからの呼出し [s] 【NGKI1080】
5210			・ CPUロック状態からの呼出し [s] 【NGKI1081】
5211	E_ID	不正ID番号	
5212			・ tskidが有効範囲外 [s] 【NGKI1082】
5213	E_RSATR	予約属性	
5214			・ 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
5215			いない [S] 【NGKI1083】
5216			・ 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
5217			[SM] 【NGKI1084】
5218	E_NOEXS	オブジェクト未登録	
5219			・ 対象タスクが未登録 【NGKI1085】
5220	E_OACV	オブジェクトアクセス違反	
5221			・ 対象タスクに対する管理操作が許可されていない [s] 【NGKI1086】
5222	E_MACV	メモリアクセス違反	
5223			・ 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
5224			いない [s] 【NGKI1087】
5225	E_OBJ	オブジェクト状態エラー	
5226			・ 対象タスクは静的APIで生成された [s] 【NGKI1088】
5227			・ 対象タスクに対してアクセス許可ベクタが設定済み [S]
5228			【NGKI1089】
5229			
5230	【機能】		
5231			
5232	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		
5233	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】。		
5234			
5235	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		
5236	式パラメータである【NGKI1091】。		
5237			
5238	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		
5239	となる【NGKI1092】。		
5240			
5241	【TOPPERS/ASPカーネルにおける規定】		
5242			
5243	ASPカーネルでは、SAC_TSK, sac_tskをサポートしない【ASPS0106】。		
5244			
5245	【TOPPERS/FMPカーネルにおける規定】		
5246			
5247	FMPカーネルでは、SAC_TSK, sac_tskをサポートしない【FMPS0104】。		
5248			
5249	【TOPPERS/HRP2カーネルにおける規定】		
5250			

5251 HRP2カーネルでは、SAC_TSKのみをサポートする【HRPS0104】.

5252

5253 【TOPPERS/SSPカーネルにおける規定】

5254

5255 SSPカーネルでは、SAC_TSK, sac_tskをサポートしない【SSPS0113】.

5256

5257 DEF_EPR タスクの実行時優先度の定義 [S] 【NGKI1093】

5258

5259 【静的API】

5260 DEF_EPR(ID tskid, { PRI exePRI })

5261

5262 【パラメータ】

5263 ID tskid 対象タスクのID番号

5264 PRI exePRI タスクの実行時優先度

5265

5266 【エラーコード】

5267 E_PAR パラメータエラー

5268 ・ exePRIが有効範囲外【NGKI1094】

5269 E_ILUSE サービスコール不正使用

5270 ・ 条件については機能の項を参照

5271 E_OBJ オブジェクト状態エラー

5272 ・ 対象タスクに対して実行優先度が設定済み【NGKI1095】

5273

5274 【サポートするカーネル】

5275

5276 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
5277 ネルは、DEF_EPRをサポートしない【NGKI1096】.

5278

5279 【機能】

5280

5281 tskidで指定したタスク（対象タスク）の実行時優先度を、exePRIで指定した優
5282 先度に設定する【NGKI1097】.

5283

5284 tskidはオブジェクト識別名、exePRIは整数定数式パラメータである【NGKI1098】.

5285

5286 exePRIが、対象タスクの起動時優先度よりも低い場合には、E_ILUSEエラーとな
5287 る【NGKI1099】.

5288

5289 【μITRON4.0仕様との関係】

5290

5291 μITRON4.0仕様に定義されていない静的APIである.

5292

5293 del_tsk タスクの削除 [TD] 【NGKI1100】

5294

5295 【C言語API】

5296 ER ercd = del_tsk(ID tskid)

5297

5298 【パラメータ】

5299 ID tskid 対象タスクのID番号

5300

5301 **【リターンパラメータ】**

5302 ER ercd 正常終了 (E_OK) またはエラーコード

5303

5304 **【エラーコード】**

5305 E_CTX コンテキストエラー

5306 ・非タスクコンテキストからの呼出し【NGKI1101】

5307 ・CPUロック状態からの呼出し【NGKI1102】

5308 E_ID 不正ID番号

5309 ・tskidが有効範囲外【NGKI1103】

5310 E_NOEXS オブジェクト未登録

5311 ・対象タスクが未登録【NGKI1104】

5312 E_OACV オブジェクトアクセス違反

5313 ・対象タスクに対する管理操作が許可されていない [P] 【NGKI1105】

5314 E_OBJ オブジェクト状態エラー

5315 ・対象タスクが休止状態でない【NGKI1106】

5316 ・対象タスクは静的APIで生成された【NGKI1107】

5317

5318 **【機能】**

5319

5320 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
5321 り．

5322

5323 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
5324 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
5325 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
5326 場合は，それらのメモリ領域が解放される【NGKI1109】．

5327

5328 **【TOPPERS/ASPカーネルにおける規定】**

5329

5330 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
5331 機能拡張パッケージでは，del_tskをサポートする【ASPS0108】．

5332

5333 **【TOPPERS/FMPカーネルにおける規定】**

5334

5335 FMPカーネルでは，del_tskをサポートしない【FMPS0105】．

5336

5337 **【TOPPERS/HRP2カーネルにおける規定】**

5338

5339 HRP2カーネルでは，del_tskをサポートしない【HRPS0105】．

5340

5341 **【TOPPERS/SSPカーネルにおける規定】**

5342

5343 SSPカーネルでは，del_tskをサポートしない【SSPS0114】．

5344

5345 act_tsk タスクの起動 [T] 【NGKI1110】

5346 iact_tsk タスクの起動 [I] 【NGKI1111】

5347

5348 **【C言語API】**

5349 ER ercd = act_tsk(ID tskid)

5350 ER ercd = iact_tsk(ID tskid)

5351

5352 **【パラメータ】**

5353 ID tskid 対象タスクのID番号

5354

5355 **【リターンパラメータ】**

5356 ER ercd 正常終了 (E_OK) またはエラーコード

5357

5358 **【エラーコード】**

5359 E_CTX コンテキストエラー

5360 ・非タスクコンテキストからの呼出し (act_tskの場合) 【NGKI1112】

5361 ・タスクコンテキストからの呼出し (iact_tskの場合) 【NGKI1113】

5362 ・CPUロック状態からの呼出し 【NGKI1114】

5363 E_ID 不正ID番号

5364 ・tskidが有効範囲外 【NGKI1115】

5365 E_NOEXS オブジェクト未登録

5366 ・対象タスクが未登録 [D] 【NGKI1116】

5367 E_OACV オブジェクトアクセス違反

5368 ・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

5369 の場合) [P] 【NGKI1117】

5370 E_QOVR キューイングオーバフロー

5371 ・条件については機能の項を参照

5372

5373 **【機能】**

5374

5375 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

5376 いは以下の通り。

5377

5378 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

5379 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】。

5380

5381 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

5382 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

5383 TMAX_ACTCNTを超える場合には、E_QOVRエラーとなる 【NGKI1120】。

5384

5385 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

5386 となる 【NGKI1121】。

5387

5388 **【補足説明】**

5389

5390 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

5391 起動時の割付けプロセッサを変更しない。

5392 -----

5393 mact_tsk 割付けプロセッサ指定でのタスクの起動 [TM] 【NGKI1122】

5394 imact_tsk 割付けプロセッサ指定でのタスクの起動 [IM] 【NGKI1123】

5395

5396 **【C言語API】**

5397 ER ercd = mact_tsk(ID tskid, ID pccid)

5398 ER ercd = imact_tsk(ID tskid, ID pccid)

5399

5400 **【パラメータ】**

5401 ID tskid 対象タスクのID番号

5402 ID prcid タスクの割付け対象のプロセッサのID番号

5403

5404 【リターンパラメータ】

5405 ER ercd 正常終了 (E_OK) またはエラーコード

5406

5407 【エラーコード】

5408 E_CTX コンテキストエラー

5409 ・非タスクコンテキストからの呼出し (mact_tskの場合)

5410 【NGKI1124】

5411 ・タスクコンテキストからの呼出し (imact_tskの場合)

5412 【NGKI1125】

5413 ・CPUロック状態からの呼出し 【NGKI1126】

5414 E_NOSPT 未サポート機能

5415 ・対象タスクが制約タスク 【NGKI1127】

5416 E_ID 不正ID番号

5417 ・tskidが有効範囲外 【NGKI1128】

5418 ・prcidが有効範囲外 【NGKI1129】

5419 E_PAR パラメータエラー

5420 ・条件については機能の項を参照

5421 E_NOEXS オブジェクト未登録

5422 ・対象タスクが未登録 [D] 【NGKI1130】

5423 E_OACV オブジェクトアクセス違反

5424 ・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

5425 の場合) [P] 【NGKI1131】

5426 E_QOVR キューイングオーバーフロー

5427 ・条件については機能の項を参照

5428

5429 【機能】

5430

5431 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

5432 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。

5433

5434 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

5435 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

5436 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

5437 【NGKI1132】。

5438

5439 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

5440 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

5441 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

5442 TMAX_ACTCNTを超える場合には、E_QOVRエラーとなる 【NGKI1134】。

5443

5444 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

5445 クとなる 【NGKI1135】。

5446

5447 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

5448 サを含んでいない場合には、E_PARエラーとなる 【NGKI1136】。

5449

5450 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

5451 れが属するクラスの初期割付けプロセッサとする【NGKI1137】.

5452

5453 **【補足説明】**

5454

5455 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
5456 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
5457 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
5458 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
5459 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
5460 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
5461 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
5462 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
5463 変更されない。

5464

5465 **【TOPPERS/ASPカーネルにおける規定】**

5466

5467 ASPカーネルでは、mact_tsk、imact_tskをサポートしない【ASPS0109】.

5468

5469 **【TOPPERS/HRP2カーネルにおける規定】**

5470

5471 HRP2カーネルでは、mact_tsk、imact_tskをサポートしない【HRPS0106】.

5472

5473 **【TOPPERS/SSPカーネルにおける規定】**

5474

5475 SSPカーネルでは、mact_tsk、imact_tskをサポートしない【SSPS0115】.

5476

5477 **【 μ ITRON4.0仕様との関係】**

5478

5479 μ ITRON4.0仕様に定義されていないサービスコールである。

5480

5481 can_act タスク起動要求のキャンセル [T] 【NGKI1138】

5482

5483 **【C言語API】**

5484 ER_UINT actcnt = can_act(ID tskid)

5485

5486 **【パラメータ】**

5487 ID tskid 対象タスクのID番号

5488

5489 **【リターンパラメータ】**

5490 ER_UINT actcnt キューイングされていた起動要求の数（正の値
5491 または0）またはエラーコード

5492

5493 **【エラーコード】**

5494 E_CTX コンテキストエラー
5495 ・非タスクコンテキストからの呼出し【NGKI1139】

5496 ・CPUロック状態からの呼出し【NGKI1140】

5497 E_ID 不正ID番号

5498 ・tskidが有効範囲外【NGKI1141】

5499 E_NOEXS オブジェクト未登録

5500 ・対象タスクが未登録 [D] 【NGKI1142】

5501 E_OACV オブジェクトアクセス違反
 5502 ・対象タスクに対する通常操作1が許可されていない [P]
 5503 【NGKI1143】
 5504
 5505 **【機能】**
 5506
 5507 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
 5508 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
 5509 下の通り。
 5510
 5511 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
 5512 求キューイング数が、サービスコールの返回值として返される【NGKI1144】。ま
 5513 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
 5514 付けプロセッサが未設定状態に戻される【NGKI1145】。
 5515
 5516 tskidにTSK_SELF（=0）を指定すると、自タスクが対象タスクとなる
 5517 【NGKI1146】。
 5518
 5519 **【TOPPERS/SSPカーネルにおける規定】**
 5520
 5521 SSPカーネルでは、can_actをサポートしない【SSPS0116】。
 5522 -----
 5523 mig_tsk タスクの割付けプロセッサの変更 [TM] 【NGKI1147】
 5524
 5525 **【C言語API】**
 5526 ER ercd = mig_tsk(ID tskid, ID prcid)
 5527
 5528 **【パラメータ】**
 5529 ID tskid 対象タスクのID番号
 5530 ID prcid タスクの割付けプロセッサのID番号
 5531
 5532 **【リターンパラメータ】**
 5533 ER ercd 正常終了（E_OK）またはエラーコード
 5534
 5535 **【エラーコード】**
 5536 E_CTX コンテキストエラー
 5537 ・非タスクコンテキストからの呼出し【NGKI1148】
 5538 ・CPUロック状態からの呼出し【NGKI1149】
 5539 ・その他の条件については機能の項を参照
 5540 E_NOSPT 未サポート機能
 5541 ・対象タスクが制約タスク【NGKI1150】
 5542 E_ID 不正ID番号
 5543 ・tskidが有効範囲外【NGKI1151】
 5544 ・prcidが有効範囲外【NGKI1152】
 5545 E_PAR パラメータエラー
 5546 ・条件については機能の項を参照
 5547 E_NOEXS オブジェクト未登録
 5548 ・対象タスクが未登録 [D] 【NGKI1153】
 5549 E_OACV オブジェクトアクセス違反
 5550 ・対象タスクに対する通常操作1が許可されていない [P]

5551 【NGKI1154】
5552 E_OBJ オブジェクト状態エラー
5553 ・条件については機能の項を参照
5554
5555 【機能】
5556
5557 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
5558 変更する．具体的な振舞いは以下の通り．
5559
5560 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
5561 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】．
5562 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
5563 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
5564 【NGKI1156】．
5565
5566 対象タスクが，自タスクが割り付けられたプロセッサと異なるプロセッサに割り
5567 付けられている場合には，E_OBJエラーとなる【NGKI1157】．
5568
5569 tskidにTSK_SELF（=0）を指定すると，自タスクが対象タスクとなる
5570 【NGKI1158】．
5571
5572 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，
5573 E_CTXエラーとなる【NGKI1159】．
5574
5575 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
5576 サを含んでいない場合には，E_PARエラーとなる【NGKI1160】．
5577
5578 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
5579 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】．
5580
5581 【補足説明】
5582
5583 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
5584 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
5585 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
5586 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
5587 のを防ぐことができる．
5588
5589 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
5590 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
5591 クの中で最低となる．
5592
5593 【TOPPERS/ASPカーネルにおける規定】
5594
5595 ASPカーネルでは，mig_tskをサポートしない【ASPS0110】．
5596
5597 【TOPPERS/HRP2カーネルにおける規定】
5598
5599 HRP2カーネルでは，mig_tskをサポートしない【HRPS0107】．
5600

5601 【TOPPERS/SSPカーネルにおける規定】

5602

5603 SSPカーネルでは、mig_tskをサポートしない【SSPS0117】。

5604

5605 【 μ ITRON4.0仕様との関係】

5606

5607 μ ITRON4.0仕様に定義されていないサービスコールである。

5608

5609 ext_tsk 自タスクの終了 [T] 【NGKI1162】

5610

5611 【C言語API】

5612 ER ercd = ext_tsk()

5613

5614 【パラメータ】

5615 なし

5616

5617 【リターンパラメータ】

5618 ER ercd エラーコード

5619

5620 【エラーコード】

5621 E_SYS システムエラー

5622 ・カーネルの誤動作【NGKI1163】

5623 E_CTX コンテキストエラー

5624 ・非タスクコンテキストからの呼出し【NGKI1164】

5625

5626 【機能】

5627

5628 自タスクを終了させる。具体的な振舞いは以下の通り。

5629

5630 自タスクに対してタスク終了時に行うべき処理が行われ、自タスクは休止状態
5631 になる【NGKI1165】。さらに、自タスクの起動要求キューイング数が0でない場
5632 合には、自タスクに対してタスク起動時に行うべき処理が行われ、自タスクは
5633 実行できる状態になる【NGKI1166】。またこの時、起動要求キューイング数か
5634 ら1が減ぜられる【NGKI1167】。

5635

5636 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
5637 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
5638 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
5639 状態に遷移させた後、自タスクを終了させる【NGKI1168】。

5640

5641 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】。

5642

5643 【TOPPERS/SSPカーネルにおける規定】

5644

5645 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
5646 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】。

5647

5648 【 μ ITRON4.0仕様との関係】

5649

5650 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

5651 とした. μ ITRON4.0仕様においては, ext_tskからはリターンしないと規定さ
5652 れている.

5653 -----

5654 ter_tsk タスクの強制終了 [T] 【NGKI1170】

5655

5656 【C言語API】

5657 ER ercd = ter_tsk(ID tskid)

5658

5659 【パラメータ】

5660 ID tskid 対象タスクのID番号

5661

5662 【リターンパラメータ】

5663 ER ercd 正常終了 (E_OK) またはエラーコード

5664

5665 【エラーコード】

5666 E_CTX コンテキストエラー

5667 ・非タスクコンテキストからの呼出し 【NGKI1171】

5668 ・CPUロック状態からの呼出し 【NGKI1172】

5669 E_ID 不正ID番号

5670 ・tskidが有効範囲外 【NGKI1173】

5671 E_NOEXS オブジェクト未登録

5672 ・対象タスクが未登録 [D] 【NGKI1174】

5673 E_OACV オブジェクトアクセス違反

5674 ・対象タスクに対する通常操作2が許可されていない [P]

5675 【NGKI1175】

5676 E_ILUSE サービスコール不正使用

5677 ・対象タスクが自タスク 【NGKI1176】

5678 E_OBJ オブジェクト状態エラー

5679 ・対象タスクが休止状態 【NGKI1177】

5680 ・その他の条件については機能の項を参照

5681

5682 【機能】

5683

5684 tskidで指定したタスク (対象タスク) を終了させる. 具体的な振舞いは以下の
5685 通り.

5686

5687 対象タスクが休止状態でない場合には, 対象タスクに対してタスク終了時に行
5688 うべき処理が行われ, 対象タスクは休止状態になる 【NGKI1178】. さらに, 対
5689 象タスクの起動要求キューイング数が0でない場合には, 対象タスクに対してタ
5690 スク起動時に行うべき処理が行われ, 対象タスクは実行できる状態になる

5691 【NGKI1179】. またこの時, 起動要求キューイング数から1が減ぜられる

5692 【NGKI1180】.

5693

5694 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
5695 サに割り付けられているタスクに限られる 【NGKI1181】. 対象タスクが自タ
5696 スクと異なるプロセッサに割り付けられている場合には, E_OBJエラーとなる

5697 【NGKI1182】.

5698

5699 【TOPPERS/FMPカーネルにおける使用上の注意】

5700

5701 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
 5702 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
 5703 方式にも依存する）。

5704

5705 **【TOPPERS/SSPカーネルにおける規定】**

5706

5707 SSPカーネルでは、ter_tskをサポートしない【SSPS0120】。

5708

5709 chg_pri タスクのベース優先度の変更 [T] 【NGKI1183】

5710

5711 **【C言語API】**

5712 ER ercd = chg_pri(ID tskid, PRI tskpri)

5713

5714 **【パラメータ】**

5715 ID tskid 対象タスクのID番号

5716 PRI tskpri ベース優先度

5717

5718 **【リターンパラメータ】**

5719 ER ercd 正常終了 (E_OK) またはエラーコード

5720

5721 **【エラーコード】**

5722 E_CTX コンテキストエラー

5723 ・非タスクコンテキストからの呼出し【NGKI1184】

5724 ・CPUロック状態からの呼出し【NGKI1185】

5725 E_NOSPT 未サポート機能

5726 ・対象タスクが制約タスク【NGKI1186】

5727 E_ID 不正ID番号

5728 ・tskidが有効範囲外【NGKI1187】

5729 E_PAR パラメータエラー

5730 ・tskpriが有効範囲外【NGKI1188】

5731 E_NOEXS オブジェクト未登録

5732 ・対象タスクが未登録 [D] 【NGKI1189】

5733 E_OACV オブジェクトアクセス違反

5734 ・対象タスクに対する通常操作2が許可されていない [P]

5735 【NGKI1190】

5736 E_ILUSE サービスコール不正使用

5737 ・条件については機能の項を参照

5738 E_OBJ オブジェクト状態エラー

5739 ・対象タスクが休止状態【NGKI1191】

5740

5741 **【機能】**

5742

5743 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
 5744 先度に変更する。具体的な振舞いは以下の通り。

5745

5746 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
 5747 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
 5748 在優先度も変更される【NGKI1193】。

5749

5750 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

5751 処理が行われる．対象タスクが実行できる状態の場合には，同じ優先度のタスク
 5752 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
 5753 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
 5754 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
 5755 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
 5756 の中で最後になる【NGKI1196】．

5757

5758 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
 5759 スクの現在優先度の変更されることはなく，優先順位も変更されない
 5760 【NGKI1197】．

5761

5762 tskidにTSK_SELF (=0) を指定すると，自タスクが対象タスクとなる
 5763 【NGKI1198】．また，tskpriにTPRI_INI (=0) を指定すると，対象タスクのベー
 5764 ス優先度が，起動時優先度に変更される【NGKI1199】．

5765

5766 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
 5767 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
 5768 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】．

5769

5770 【TOPPERS/SSPカーネルにおける規定】

5771

5772 SSPカーネルでは，chg_priをサポートしない【SSPS0121】．

5773

5774 【μITRON4.0仕様との関係】

5775

5776 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
 5777 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
 5778 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．

5779 -----

5780 get_pri タスク優先度の参照 [T] 【NGKI1202】

5781

5782 【C言語API】

5783 ER ercd = get_pri(ID tskid, PRI *p_tskpri)

5784

5785 【パラメータ】

5786 ID tskid 対象タスクのID番号

5787 PRI * p_tskpri 現在優先度を入れるメモリ領域へのポインタ

5788

5789 【リターンパラメータ】

5790 ER ercd 正常終了 (E_OK) またはエラーコード

5791 PRI tskpri 現在優先度

5792

5793 【エラーコード】

5794 E_CTX コンテキストエラー

5795 ・非タスクコンテキストからの呼出し【NGKI1203】

5796 ・CPUロック状態からの呼出し【NGKI1204】

5797 E_ID 不正ID番号

5798 ・tskidが有効範囲外【NGKI1205】

5799 E_NOEXS オブジェクト未登録

5800 ・対象タスクが未登録 [D] 【NGKI1206】

5801 E_OACV オブジェクトアクセス違反
5802 ・対象タスクに対する参照操作が許可されていない [P] 【NGKI1207】
5803 E_MACV メモリアクセス違反
5804 ・p_tskpriが指すメモリ領域への書込みアクセスが許可され
5805 ていない [P] 【NGKI1208】
5806 E_OBJ オブジェクト状態エラー
5807 ・対象タスクが休止状態 【NGKI1209】
5808

5809 **【機能】**

5810
5811 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
5812 いは以下の通り．

5813
5814 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
5815 で指定したメモリ領域に返される 【NGKI1210】．

5816
5817 tskidにTSK_SELF（=0）を指定すると，自タスクが対象タスクとなる
5818 【NGKI1211】．

5819
5820 **【TOPPERS/SSPカーネルにおける規定】**

5821
5822 SSPカーネルでは，get_priをサポートしない 【SSPS0122】．

5823 -----
5824 get_inf 自タスクの拡張情報の参照 [T] 【NGKI1212】

5825
5826 **【C言語API】**

5827 ER ercd = get_inf(intptr_t *p_exinf)

5828
5829 **【パラメータ】**

5830 intptr_t * p_exinf 拡張情報を入れるメモリ領域へのポインタ

5831
5832 **【リターンパラメータ】**

5833 ER ercd 正常終了 (E_OK) またはエラーコード
5834 intptr_t exinf 拡張情報

5835
5836 **【エラーコード】**

5837 E_CTX コンテキストエラー
5838 ・非タスクコンテキストからの呼出し 【NGKI1213】
5839 ・CPUロック状態からの呼出し 【NGKI1214】
5840 E_MACV メモリアクセス違反
5841 ・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
5842 いない [P] 【NGKI1215】
5843

5844 **【機能】**

5845
5846 自タスクの拡張情報を参照する．参照した拡張情報は，p_exinfで指定したメモ
5847 リ領域に返される 【NGKI1216】．

5848
5849 **【TOPPERS/SSPカーネルにおける規定】**

5850

5851 SSPカーネルでは、get_infをサポートしない【SSPS0123】。

5852

5853 【μITRON4.0仕様との関係】

5854

5855 μITRON4.0仕様に定義されていないサービスコールである。

5856

5857 ref_tsk タスクの状態参照 [T] 【NGKI1217】

5858

5859 【C言語API】

5860 ER ercd = ref_tsk(ID tskid, T_RTsk *pk_rtsk)

5861

5862 【パラメータ】

5863 ID tskid 対象タスクのID番号

5864 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

5865

5866 【リターンパラメータ】

5867 ER ercd 正常終了 (E_OK) またはエラーコード

5868

5869 *タスクの現在状態 (パケットの内容)

5870 STAT tskstat タスク状態

5871 PRI tskpri タスクの現在優先度

5872 PRI tsbpri タスクのベース優先度

5873 STAT tsawait タスクの待ち要因

5874 ID wobjid タスクの待ち対象のオブジェクトのID

5875 TMO lefttmo タスクがタイムアウトするまでの時間

5876 uint_t actcnt タスクの起動要求キューイング数

5877 uint_t wupcnt タスクの起床要求キューイング数

5878 bool_t texmsk タスクがタスク例外処理マスク状態か否か (保護機能対応カーネルの場合)

5879 bool_t waifbd タスクが待ち禁止状態か否か (保護機能対応カーネルの場合)

5880 uint_t svclevel タスクの拡張サービスコールのネストレベル (保護機能対応カーネルの場合)

5881 ID prcid タスクの割付けプロセッサのID (マルチプロセッサ対応カーネルの場合)

5882 ID actprc タスクの次回起動時の割付けプロセッサのID (マルチプロセッサ対応カーネルの場合)

5883

5884 【エラーコード】

5885 E_CTX コンテキストエラー

5886 ・非タスクコンテキストからの呼出し【NGKI1218】

5887 ・CPUロック状態からの呼出し【NGKI1219】

5888 E_ID 不正ID番号

5889 ・tskidが有効範囲外【NGKI1220】

5890 E_NOEXS オブジェクト未登録

5891 ・対象タスクが未登録 [D] 【NGKI1221】

5892 E_OACV オブジェクトアクセス違反

5893 ・対象タスクに対する参照操作が許可されていない [P] 【NGKI1222】

5894 E_MACV メモリアクセス違反

5895 ・pk_rtskが指すメモリ領域への書き込みアクセスが許可されて

5896

5901 いない [P] 【NGKI1223】

5902

5903 【機能】

5904

5905 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
5906 態は、pk_rtskで指定したメモリ領域に返される【NGKI1224】．

5907

5908 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
5909 る【NGKI1225】．

5910

5911	TTS_RUN	0x01U	実行状態
5912	TTS_RDY	0x02U	実行可能状態
5913	TTS_WAI	0x04U	待ち状態
5914	TTS_SUS	0x08U	強制待ち状態
5915	TTS_WAS	0x0cU	二重待ち状態
5916	TTS_DMT	0x10U	休止状態

5917

5918 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、
5919 tskstatがTTS_SUSとなる場合がある【NGKI1226】．この状況は、自タスクに対
5920 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
5921 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
5922 ある．

5923

5924 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、
5925 tskbpriには対象タスクのベース優先度が返される【NGKI1227】．対象タスクが
5926 休止状態である場合には、tskpriとtskbpriの値は保証されない【NGKI1228】．

5927

5928 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
5929 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】．

5930

5931	TTW_SLP	0x0001U	起床待ち
5932	TTW_DLY	0x0002U	時間経過待ち
5933	TTW_SEM	0x0004U	セマフォの資源獲得待ち
5934	TTW_FLG	0x0008U	イベントフラグ待ち
5935	TTW_SDTQ	0x0010U	データキューへの送信待ち
5936	TTW_RDTQ	0x0020U	データキューからの受信待ち
5937	TTW_SPDQ	0x0100U	優先度データキューへの送信待ち
5938	TTW_RPDQ	0x0200U	優先度データキューからの受信待ち
5939	TTW_MBX	0x0040U	メールボックスからの受信待ち
5940	TTW_MTX	0x0080U	ミューテックスのロック待ち状態
5941	TTW_MPF	0x2000U	固定長メモリブロックの獲得待ち

5942

5943 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
5944 【NGKI1230】．

5945

5946 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
5947 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

5948 【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

5949 経過待ち状態である場合には、wobjidの値は保証されない【NGKI1232】．

5950

5951 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
5952 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
5953 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
5954 【NGKI1234】。
5955
5956 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
5957 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、
5958 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
5959 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
5960 定義される) に型キャストした値が返される【NGKI1236】。
5961
5962 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
5963 【NGKI1237】。
5964
5965 actcntには、対象タスクの起動要求キューイング数が返される【NGKI1238】。
5966
5967 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
5968 イング数が返される【NGKI1239】。対象タスクが休止状態である場合には、
5969 wupcntの値は保証されない【NGKI1240】。
5970
5971 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、
5972 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
5973 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
5974 にtrue, そうでない場合にfalseが返される【NGKI1242】。またsvclevelには、
5975 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
5976 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
5977 【NGKI1243】。対象タスクが休止状態である場合には、texmsk, waifbd,
5978 svclevelの値は保証されない【NGKI1244】。
5979
5980 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
5981 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
5982 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
5983 ロセッサが未設定の場合には、actprcにTPRC_NONE (=0) が返される
5984 【NGKI1247】。
5985
5986 tskidにTSK_SELF (=0) を指定すると、自タスクが対象タスクとなる
5987 【NGKI1248】。
5988
5989 【補足説明】
5990
5991 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
5992 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
5993 を正しく得ることができる。
5994
5995 【TOPPERS/ASPカーネルにおける規定】
5996
5997 ASPカーネルでは、tskwaitにTTW_MTXが返ることはない【ASPS0111】。ただし、
5998 ミューテックス機能拡張パッケージを用いると、tskwaitにTTW_MTXが返る場合
5999 がある【ASPS0112】。
6000

6001 【TOPPERS/FMPカーネルにおける規定】

6002

6003 FMPカーネルでは、tskwaitにTTW_MTXが返ることはない【FMPS0106】.

6004

6005 【TOPPERS/HRP2カーネルにおける規定】

6006

6007 HRP2カーネルでは、tskwaitにTTW_MBXが返ることはない【HRPS0108】.

6008

6009 【TOPPERS/SSPカーネルにおける規定】

6010

6011 SSPカーネルでは、ref_tskをサポートしない【SSPS0124】.

6012

6013 【使用上の注意】

6014

6015 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
6016 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
6017 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
6018 ている可能性があるためである.

6019

6020 【μ ITRON4.0仕様との関係】

6021

6022 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.
6023 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
6024 した.

6025

6026 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
6027 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.

6028

6029 【μ ITRON4.0/PX仕様との関係】

6030

6031 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
6032 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
6033 トレベル (svclevel) を追加した.

6034

6035

6036 4.2 タスク付属同期機能

6037

6038 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
6039 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.

6040

6041 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】.

6042

6043 ・ 起床要求キューイング数

6044

6045 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
6046 であり、タスクの起動時に0に初期化される【NGKI1250】.

6047

6048 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.

6049

6050 TMAX_WUPCNT タスクの起床要求キューイング数の最大値【NGKI1251】

6051
6052 【TOPPERS/ASPカーネルにおける規定】
6053
6054 ASPカーネルでは、TMAX_WUPCNTは1に固定されている【ASPS0113】。
6055
6056 【TOPPERS/FMPカーネルにおける規定】
6057
6058 FMPカーネルでは、TMAX_WUPCNTは1に固定されている【FMPS0107】。
6059
6060 【TOPPERS/HRP2カーネルにおける規定】
6061
6062 HRP2カーネルでは、TMAX_WUPCNTは1に固定されている【HRPS0109】。
6063
6064 【TOPPERS/SSPカーネルにおける規定】
6065
6066 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】。
6067
6068 【μITRON4.0仕様との関係】
6069
6070 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。
6071 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
6072 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
6073 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。
6074 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
6075 スト数（suscnt）を除外した。
6076 -----
6077 slp_tsk 起床待ち〔T〕【NGKI1252】
6078 tslp_tsk 起床待ち（タイムアウト付き）〔T〕【NGKI1253】
6079
6080 【C言語API】
6081 ER ercd = slp_tsk()
6082 ER ercd = tslp_tsk(TMO tmout)
6083
6084 【パラメータ】
6085 TMO tmout タイムアウト時間（tslp_tskの場合）
6086
6087 【リターンパラメータ】
6088 ER ercd 正常終了（E_OK）またはエラーコード
6089
6090 【エラーコード】
6091 E_CTX コンテキストエラー
6092 ・ディスパッチ保留状態からの呼出し【NGKI1254】
6093 E_NOSPT 未サポート機能
6094 ・制約タスクからの呼出し【NGKI1255】
6095 E_PAR パラメータエラー
6096 ・tmoutが無効（tslp_tskの場合）【NGKI1256】
6097 E_TMOUT ポーリング失敗またはタイムアウト（slp_tskを除く）【NGKI1257】
6098 E_RLWAI 待ち禁止状態または待ち状態の強制解除【NGKI1258】
6099
6100 【機能】

6101
6102 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。
6103
6104 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
6105 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、
6106 自タスクは起床待ち状態となる【NGKI1260】。
6107
6108 **【補足説明】**
6109
6110 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
6111 状態を維持し、自タスクの優先順位は変化しない。
6112 -----
6113 wup_tsk タスクの起床 [T] 【NGKI1261】
6114 iwup_tsk タスクの起床 [I] 【NGKI1262】
6115
6116 **【C言語API】**
6117 ER ercd = wup_tsk(ID tskid)
6118 ER ercd = iwup_tsk(ID tskid)
6119
6120 **【パラメータ】**
6121 ID tskid 対象タスクのID番号
6122
6123 **【リターンパラメータ】**
6124 ER ercd 正常終了 (E_OK) またはエラーコード
6125
6126 **【エラーコード】**
6127 E_CTX コンテキストエラー
6128 ・非タスクコンテキストからの呼出し (wup_tskの場合) 【NGKI1263】
6129 ・タスクコンテキストからの呼出し (iwup_tskの場合) 【NGKI1264】
6130 ・CPUロック状態からの呼出し 【NGKI1265】
6131 E_NOSPT 未サポート機能
6132 ・対象タスクが制約タスク 【NGKI1266】
6133 E_ID 不正ID番号
6134 ・tskidが有効範囲外 【NGKI1267】
6135 E_NOEXS オブジェクト未登録
6136 ・対象タスクが未登録 [D] 【NGKI1268】
6137 E_OACV オブジェクトアクセス違反
6138 ・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
6139 の場合) [P] 【NGKI1269】
6140 E_OBJ オブジェクト状態エラー
6141 ・対象タスクが休止状態 【NGKI1270】
6142 E_QOVR キューイングオーバフロー
6143 ・条件については機能の項を参照
6144
6145 **【機能】**
6146
6147 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
6148 り。
6149
6150 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

6151 【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
6152 からE_OKが返る【NGKI1272】.

6153

6154 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
6155 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
6156 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
6157 【NGKI1274】.

6158

6159 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
6160 となる【NGKI1275】.

6161 -----

6162 can_wup タスク起床要求のキャンセル [T] 【NGKI1276】

6163

6164 【C言語API】

6165 ER_UINT wupcnt = can_wup(ID tskid)

6166

6167 【パラメータ】

6168 ID tskid 対象タスクのID番号

6169

6170 【リターンパラメータ】

6171 ER_UINT wupcnt キューイングされていた起床要求の数（正の値
6172 または0）またはエラーコード

6173

6174 【エラーコード】

6175 E_CTX コンテキストエラー

6176 ・非タスクコンテキストからの呼出し【NGKI1277】

6177 ・CPUロック状態からの呼出し【NGKI1278】

6178 E_NOSPT 未サポート機能

6179 ・対象タスクが制約タスク【NGKI1279】

6180 E_ID 不正ID番号

6181 ・tskidが有効範囲外【NGKI1280】

6182 E_NOEXS オブジェクト未登録

6183 ・対象タスクが未登録 [D] 【NGKI1281】

6184 E_OACV オブジェクトアクセス違反

6185 ・対象タスクに対する通常操作1が許可されていない [P]
6186 【NGKI1282】

6187 E_OBJ オブジェクト状態エラー

6188 ・対象タスクが休止状態【NGKI1283】

6189

6190 【機能】

6191

6192 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
6193 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
6194 下の通り.

6195

6196 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
6197 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
6198 の返値として返される【NGKI1284】.

6199

6200 tskidにTSK_SELF (=0) を指定すると、自タスクが対象タスクとなる

6201 【NGKI1285】 .
6202 -----
6203 rel_wai 強制的な待ち解除 [T] 【NGKI1286】
6204 irel_wai 強制的な待ち解除 [I] 【NGKI1287】
6205
6206 【C言語API】
6207 ER ercd = rel_wai(ID tskid)
6208 ER ercd = irel_wai(ID tskid)
6209
6210 【パラメータ】
6211 ID tskid 対象タスクのID番号
6212
6213 【リターンパラメータ】
6214 ER ercd 正常終了 (E_OK) またはエラーコード
6215
6216 【エラーコード】
6217 E_CTX コンテキストエラー
6218 ・ 非タスクコンテキストからの呼出し (rel_waiの場合) 【NGKI1288】
6219 ・ タスクコンテキストからの呼出し (irel_waiの場合) 【NGKI1289】
6220 ・ CPUロック状態からの呼出し 【NGKI1290】
6221 E_NOSPT 未サポート機能
6222 ・ 対象タスクが制約タスク 【NGKI1291】
6223 E_ID 不正ID番号
6224 ・ tskidが有効範囲外 【NGKI1292】
6225 E_NOEXS オブジェクト未登録
6226 ・ 対象タスクが未登録 [D] 【NGKI1293】
6227 E_OACV オブジェクトアクセス違反
6228 ・ 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
6229 の場合) [P] 【NGKI1294】
6230 E_OBJ オブジェクト状態エラー
6231 ・ 対象タスクが待ち状態でない 【NGKI1295】
6232
6233 【機能】
6234
6235 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
6236 舞いは以下の通り.
6237
6238 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
6239 【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
6240 からE_RLWAIが返る 【NGKI1297】 .
6241 -----
6242 sus_tsk 強制待ち状態への遷移 [T] 【NGKI1298】
6243
6244 【C言語API】
6245 ER ercd = sus_tsk(ID tskid)
6246
6247 【パラメータ】
6248 ID tskid 対象タスクのID番号
6249
6250 【リターンパラメータ】

6251 ER ercd 正常終了 (E_OK) またはエラーコード
6252
6253 **【エラーコード】**
6254 E_CTX コンテキストエラー
6255 ・非タスクコンテキストからの呼出し【NGKI1299】
6256 ・CPUロック状態からの呼出し【NGKI1300】
6257 ・その他の条件については機能の項を参照
6258 E_NOSPT 未サポート機能
6259 ・対象タスクが制約タスク【NGKI1301】
6260 E_ID 不正ID番号
6261 ・tskidが有効範囲外【NGKI1302】
6262 E_NOEXS オブジェクト未登録
6263 ・対象タスクが未登録 [D] 【NGKI1303】
6264 E_OACV オブジェクトアクセス違反
6265 ・対象タスクに対する通常操作2が許可されていない [P]
6266 【NGKI1304】
6267 E_OBJ オブジェクト状態エラー
6268 ・対象タスクが休止状態【NGKI1305】
6269 E_QOVR キューイングオーバフロー
6270 ・対象タスクが強制待ち状態（二重待ち状態を含む）【NGKI1306】
6271
6272 **【機能】**
6273
6274 tskidで指定したタスク（対象タスク）を強制待ちにする．具体的な振舞いは以下
6275 の通り．
6276
6277 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
6278 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
6279 重待ち状態となる【NGKI1308】．
6280
6281 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，
6282 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
6283 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
6284 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．
6285
6286 tskidにTSK_SELF (=0) を指定すると，自タスクが対象タスクとなる
6287 【NGKI1310】．
6288
6289 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，
6290 E_CTXエラーとなる【NGKI1311】．
6291 -----
6292 rsm_tsk 強制待ち状態からの再開 [T] 【NGKI1312】
6293
6294 **【C言語API】**
6295 ER ercd = rsm_tsk(ID tskid)
6296
6297 **【パラメータ】**
6298 ID tskid 対象タスクのID番号
6299
6300 **【リターンパラメータ】**

6301	ER	ercd	正常終了 (E_OK) またはエラーコード
6302			
6303	【エラーコード】		
6304	E_CTX	コンテキストエラー	
6305		・非タスクコンテキストからの呼出し	【NGKI1313】
6306		・CPUロック状態からの呼出し	【NGKI1314】
6307	E_NOSPT	未サポート機能	
6308		・対象タスクが制約タスク	【NGKI1315】
6309	E_ID	不正ID番号	
6310		・tskidが有効範囲外	【NGKI1316】
6311	E_NOEXS	オブジェクト未登録	
6312		・対象タスクが未登録 [D]	【NGKI1317】
6313	E_OACV	オブジェクトアクセス違反	
6314		・対象タスクに対する通常操作2が許可されていない [P]	
6315			【NGKI1318】
6316	E_OBJ	オブジェクト状態エラー	
6317		・対象タスクが強制待ち状態（二重待ち状態を含む）でない	
6318			【NGKI1319】
6319			
6320	【機能】		
6321			
6322	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		
6323	舞いは以下の通り．		
6324			
6325	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		
6326	れる 【NGKI1320】 ．		
6327	-----		
6328	dis_wai	待ち禁止状態への遷移 [TP]	【NGKI1321】
6329	idis_wai	待ち禁止状態への遷移 [IP]	【NGKI1322】
6330			
6331	【C言語API】		
6332	ER ercd = dis_wai(ID tskid)		
6333	ER ercd = idis_wai(ID tskid)		
6334			
6335	【パラメータ】		
6336	ID	tskid	対象タスクのID番号
6337			
6338	【リターンパラメータ】		
6339	ER	ercd	正常終了 (E_OK) またはエラーコード
6340			
6341	【エラーコード】		
6342	E_CTX	コンテキストエラー	
6343		・非タスクコンテキストからの呼出し (dis_waiの場合)	【NGKI1323】
6344		・タスクコンテキストからの呼出し (idis_waiの場合)	【NGKI1324】
6345		・CPUロック状態からの呼出し	【NGKI1325】
6346	E_NOSPT	未サポート機能	
6347		・対象タスクが制約タスク	【NGKI1326】
6348	E_ID	不正ID番号	
6349		・tskidが有効範囲外	【NGKI1327】
6350	E_NOEXS	オブジェクト未登録	

6351 ・対象タスクが未登録 [D] 【NGKI1328】
6352 E_OACV オブジェクトアクセス違反
6353 ・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
6354 の場合) 【NGKI1329】
6355 E_OBJ オブジェクト状態エラー
6356 ・対象タスクが休止状態 【NGKI1330】
6357 ・対象タスクがタスク例外処理マスク状態でない 【NGKI1331】
6358 E_QOVR キューイングオーバーフロー
6359 ・対象タスクが待ち禁止状態 【NGKI1332】

6361 **【機能】**

6362
6363 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
6364 は以下の通り.

6365
6366 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,
6367 対象タスクは待ち禁止状態になる 【NGKI1333】.

6368
6369 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
6370 となる 【NGKI1334】.

6371
6372 **【TOPPERS/ASPカーネルにおける規定】**

6373
6374 ASPカーネルでは, dis_waiをサポートしない 【ASPS0114】.

6375
6376 **【TOPPERS/FMPカーネルにおける規定】**

6377
6378 FMPカーネルでは, dis_waiをサポートしない 【FMPS0108】.

6379
6380 **【補足説明】**

6381
6382 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
6383 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
6384 を呼び出せばよい.

6385
6386 **【未決定事項】**

6387
6388 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
6389 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,
6390 現時点では未決定である.

6391
6392 **【 μ ITRON4.0/PX仕様との関係】**

6393
6394 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.

6395 -----
6396 ena_wai 待ち禁止状態の解除 [TP] 【NGKI1335】

6397 iena_wai 待ち禁止状態の解除 [IP] 【NGKI1336】

6398
6399 **【C言語API】**

6400 ER ercd = ena_wai(ID tskid)

6401 ER ercd = iena_wai(ID tskid)
6402
6403 **【パラメータ】**
6404 ID tskid 対象タスクのID番号
6405
6406 **【リターンパラメータ】**
6407 ER ercd 正常終了 (E_OK) またはエラーコード
6408
6409 **【エラーコード】**
6410 E_CTX コンテキストエラー
6411 ・ 非タスクコンテキストからの呼出し (ena_waiの場合) 【NGKI1337】
6412 ・ タスクコンテキストからの呼出し (iena_waiの場合) 【NGKI1338】
6413 ・ CPUロック状態からの呼出し 【NGKI1339】
6414 E_NOSPT 未サポート機能
6415 ・ 対象タスクが制約タスク 【NGKI1340】
6416 E_ID 不正ID番号
6417 ・ tskidが有効範囲外 【NGKI1341】
6418 E_NOEXS オブジェクト未登録
6419 ・ 対象タスクが未登録 [D] 【NGKI1342】
6420 E_OACV オブジェクトアクセス違反
6421 ・ 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
6422 の場合) 【NGKI1343】
6423 E_OBJ オブジェクト状態エラー
6424 ・ 対象タスクが休止状態 【NGKI1344】
6425 ・ 対象タスクが待ち禁止状態でない 【NGKI1345】
6426
6427 **【機能】**
6428
6429 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
6430 舞いは以下の通り.
6431
6432 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
6433 【NGKI1346】.
6434
6435 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
6436 となる 【NGKI1347】.
6437
6438 **【TOPPERS/ASPカーネルにおける規定】**
6439
6440 ASPカーネルでは, ena_waiをサポートしない 【ASPS0115】.
6441
6442 **【TOPPERS/FMPカーネルにおける規定】**
6443
6444 FMPカーネルでは, ena_waiをサポートしない 【FMPS0109】.
6445
6446 **【未決定事項】**
6447
6448 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
6449 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,
6450 現時点では未決定である.

6451
6452 【 μ ITRON4.0/PX仕様との関係】
6453
6454 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。
6455

6456 dly_tsk 自タスクの遅延 [T] 【NGKI1348】
6457
6458 【C言語API】
6459 ER ercd = dly_tsk(RELTIM dlytim)
6460
6461 【パラメータ】
6462 RELTIM dlytim 遅延時間
6463
6464 【リターンパラメータ】
6465 ER ercd 正常終了 (E_OK) またはエラーコード
6466
6467 【エラーコード】
6468 E_CTX コンテキストエラー
6469 ・ディスパッチ保留状態からの呼出し 【NGKI1349】
6470 E_NOSPT 未サポート機能
6471 ・制約タスクからの呼出し 【NGKI1350】
6472 E_PAR パラメータエラー
6473 ・dlytimがTMAX_RELTIMより大きい 【NGKI1351】
6474 E_RLWAI 待ち禁止状態または待ち状態の強制解除 【NGKI1352】
6475
6476 【機能】
6477
6478 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。
6479
6480 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
6481 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、
6482 自タスクは待ち解除され、dly_tskからE_OKが返る【NGKI1354】。
6483

6484
6485 4.3 タスク例外処理機能
6486
6487 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
6488 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
6489 1つのみ登録できるため、タスクIDによって識別する【NGKI1355】。
6490
6491 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】。
6492
6493 ・タスク例外処理ルーチン属性
6494 ・タスク例外処理禁止フラグ
6495 ・保留例外要因
6496 ・タスク例外処理ルーチンの先頭番地
6497
6498 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、
6499 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
6500 【NGKI1358】。

6501
6502 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
6503 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
6504 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
6505 ぶ【NGKI1360】。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
6506 た状態に初期化される【NGKI1361】。
6507
6508 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
6509 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】。
6510
6511 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
6512 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
6513 いる」「割り込み優先度マスク全解除状態である」「CPUロック状態でない」の6
6514 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
6515 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
6516 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
6517 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。
6518
6519 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
6520 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
6521 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
6522 る【NGKI1366】。
6523
6524 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
6525 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
6526 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
6527 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、
6528 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
6529 行開始時スタック不正例外と呼ぶ。
6530
6531 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
6532 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、
6533 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
6534 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
6535 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
6536 タック不正例外と呼ぶ。
6537
6538 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
6539 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、
6540 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、
6541 「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
6542 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
6543 い【NGKI1372】。
6544
6545 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
6546 護ドメインに属する【NGKI1373】。
6547
6548 タスク例外処理機能に用いるデータ型は次の通り。
6549
6550 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

6551 定義) 【NGKI1374】

6552

6553 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】 .

6554

```
6555 void task_exception_routine(TEXTN texptn, intptr_t exinf)
6556 {
6557     タスク例外処理ルーチン本体
6558 }
```

6559

6560 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
6561 の拡張情報が, それぞれ渡される 【NGKI1376】 .

6562

6563 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 .

6564

6565 TBIT_TEXPTN タスク例外要因のビット数 (TEXTNの有効ビット数)
6566 【NGKI1377】

6567

6568 【補足説明】

6569

6570 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
6571 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
6572 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
6573 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,
6574 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
6575 揃うことになる.

6576

6577 【TOPPERS/ASPカーネルにおける規定】

6578

6579 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
6580 【ASPS0116】 .

6581

6582 【TOPPERS/FMPカーネルにおける規定】

6583

6584 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
6585 【FMPS0110】 .

6586

6587 【TOPPERS/HRP2カーネルにおける規定】

6588

6589 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
6590 【HRPS0110】 .

6591

6592 【TOPPERS/SSPカーネルにおける規定】

6593

6594 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】 .

6595

6596 【μ ITRON4.0仕様との関係】

6597

6598 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
6599 行が開始されないという仕様に変更した.

6600

6601 【 μ ITRON4.0/PX仕様との関係】

6602

6603 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
 6604 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。

6605

6606 【仕様変更の経緯】

6607

6608 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
 6609 「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
 6610 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
 6611 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
 6612 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
 6613 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
 6614 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。

6615

6616 DEF_TEX タスク例外処理ルーチンの定義 [S] 【NGKI1378】

6617 def_tex タスク例外処理ルーチンの定義 [TD] 【NGKI1379】

6618

6619 【静的API】

6620 DEF_TEX(ID tskid, { ATR texatr, TEXRTN texrtn })

6621

6622 【C言語API】

6623 ER ercd = def_tex(ID tskid, const T_DTEX *pk_dtex)

6624

6625 【パラメータ】

6626	ID	tskid	対象タスクのID番号
6627	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）

6628

6629 *タスク例外処理ルーチンの定義情報（パケットの内容）

6630	ATR	texatr	タスク例外処理ルーチン属性
------	-----	--------	---------------

6631	TEXRTN	texrtn	タスク例外処理ルーチンの先頭番地
------	--------	--------	------------------

6632

6633 【リターンパラメータ】

6634	ER	ercd	正常終了 (E_OK) またはエラーコード
------	----	------	-----------------------

6635

6636 【エラーコード】

6637	E_CTX	コンテキストエラー
------	-------	-----------

6638		・非タスクコンテキストからの呼出し [s] 【NGKI1380】
------	--	------------------------------------

6639		・CPUロック状態からの呼出し [s] 【NGKI1381】
------	--	----------------------------------

6640	E_ID	不正ID番号
------	------	--------

6641		・tskidが有効範囲外 [s] 【NGKI1382】
------	--	-------------------------------

6642	E_RSATR	予約属性
------	---------	------

6643		・texatrが無効 【NGKI1383】
------	--	-----------------------

6644		・その他の条件については機能の項を参照
------	--	---------------------

6645	E_PAR	パラメータエラー
------	-------	----------

6646		・texrtnがプログラムの先頭番地として正しくない 【NGKI1384】
------	--	---------------------------------------

6647	E_NOEXS	オブジェクト未登録
------	---------	-----------

6648		・対象タスクが未登録 【NGKI1385】
------	--	-----------------------

6649	E_OACV	オブジェクトアクセス違反
------	--------	--------------

6650

6651 ・対象タスクに対する管理操作が許可されていない [sP]
6652 【NGKI1386】
6653 E_MACV メモリアクセス違反
6654 ・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
6655 いない [sP] 【NGKI1387】
6656 E_OBJ オブジェクト状態エラー
6657 ・対象タスクは静的APIで生成された [s] 【NGKI1388】
6658 ・その他の条件については機能の項を参照
6659
6660 【機能】
6661
6662 tskidで指定したタスク（対象タスク）に対して、各パラメータで指定したタスク
6663 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
6664 【NGKI1389】。
6665
6666 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
6667 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
6668 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
6669 【NGKI1391】。
6670
6671 静的APIにおいては、tskidはオブジェクト識別名、texatrは整数定数式パラメータ、
6672 texrtnは一般定数式パラメータである【NGKI1392】。
6673
6674 タスク例外処理ルーチンを定義する場合（DEF_TEXの場合およびdef_texにおいて
6675 pk_dtexをNULL以外にした場合）で、対象タスクに対してすでにタスク例外処理
6676 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】。
6677
6678 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
6679 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
6680 なる【NGKI1395】。また、def_texでタスク例外処理ルーチンを定義する場合には、
6681 タスク例外処理ルーチンの属する保護ドメインを設定する必要はなく、タスク
6682 例外処理ルーチン属性にTA_DOM(domid)を指定した場合にはE_RSATRエラーと
6683 なる【NGKI1396】。ただし、TA_DOM(TDOM_SELF)を指定した場合には、指定が無視
6684 され、E_RSATRエラーは検出されない【NGKI1397】。
6685
6686 マルチプロセッサ対応カーネルにおいて、DEF_TEXは、対象タスクが属するクラス
6687 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
6688 なる【NGKI1399】。また、def_texでタスク例外処理ルーチンを定義する場合には、
6689 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
6690 例外処理ルーチン属性にTA_CLS(clsid)を指定した場合にはE_RSATRエラーとなる
6691 【NGKI1400】。ただし、TA_CLS(CLS_SELF)を指定した場合には、指定が無視
6692 され、E_RSATRエラーは検出されない【NGKI1401】。
6693
6694 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
6695 NULLにした場合）で、対象タスクに対してタスク例外処理ルーチンが定義されて
6696 いない場合には、E_OBJエラーとなる【NGKI1402】。
6697
6698 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
6699 となる【NGKI1403】。
6700

6701 【TOPPERS/ASPカーネルにおける規定】

6702

6703 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
6704 成機能拡張パッケージでは、def_texもサポートする【ASPS0118】。

6705

6706 【TOPPERS/FMPカーネルにおける規定】

6707

6708 FMPカーネルでは、DEF_TEXのみをサポートする【FMPS0111】。

6709

6710 【TOPPERS/HRP2カーネルにおける規定】

6711

6712 HRP2カーネルでは、DEF_TEXのみをサポートする【HRPS0111】。

6713

6714 【 μ ITRON4.0仕様との関係】

6715

6716 texrtnのデータ型をTEXTNに変更した。

6717

6718 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
6719 合に、E_OBJエラーとすることにした。

6720

6721 -----
ras_tex タスク例外処理の要求 [T] 【NGKI1404】

6722 iras_tex タスク例外処理の要求 [I] 【NGKI1405】

6723

6724 【C言語API】

6725 ER ercd = ras_tex(ID tskid, TEXPTN rasptn)

6726 ER ercd = iras_tex(ID tskid, TEXPTN rasptn)

6727

6728 【パラメータ】

6729 ID tskid 対象タスクのID番号

6730 TEXPTN rasptn 要求するタスク例外処理のタスク例外要因

6731

6732 【リターンパラメータ】

6733 ER ercd 正常終了 (E_OK) またはエラーコード

6734

6735 【エラーコード】

6736 E_CTX コンテキストエラー

6737 ・非タスクコンテキストからの呼出し (ras_texの場合) 【NGKI1406】

6738 ・タスクコンテキストからの呼出し (iras_texの場合) 【NGKI1407】

6739 ・CPUロック状態からの呼出し 【NGKI1408】

6740 E_ID 不正ID番号

6741 ・tskidが有効範囲外 【NGKI1409】

6742 E_PAR パラメータエラー

6743 ・rasptnが0 【NGKI1410】

6744 E_NOEXS オブジェクト未登録

6745 ・対象タスクが未登録 [D] 【NGKI1411】

6746 E_OACV オブジェクトアクセス違反

6747 ・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
6748 の場合) [P] 【NGKI1412】

6749 E_OBJ オブジェクト状態エラー

6750 ・対象タスクが休止状態 【NGKI1413】

6751 ・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
6752 ない【NGKI1414】

6753
6754 **【機能】**

6755
6756 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
6757 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
6758 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
6759 【NGKI1415】．

6760
6761 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
6762 となる【NGKI1416】．

6763 -----
6764 dis_tex タスク例外処理の禁止 [T] 【NGKI1417】

6765
6766 **【C言語API】**

6767 ER ercd = dis_tex()

6768
6769 **【パラメータ】**

6770 なし

6771
6772 **【リターンパラメータ】**

6773 ER ercd 正常終了 (E_OK) またはエラーコード

6774
6775 **【エラーコード】**

6776 E_CTX コンテキストエラー
6777 ・非タスクコンテキストからの呼出し【NGKI1419】
6778 ・CPUロック状態からの呼出し【NGKI1420】
6779 E_OBJ オブジェクト状態エラー
6780 ・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
6781 ない【NGKI1421】

6782
6783 **【機能】**

6784
6785 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，
6786 自タスクをタスク例外処理禁止状態に遷移させる．

6787 -----
6788 ena_tex タスク例外処理の許可 [T] 【NGKI1423】

6789
6790 **【C言語API】**

6791 ER ercd = ena_tex()

6792
6793 **【パラメータ】**

6794 なし

6795
6796 **【リターンパラメータ】**

6797 ER ercd 正常終了 (E_OK) またはエラーコード

6798
6799 **【エラーコード】**

6800 E_CTX コンテキストエラー

6801 ・ 非タスクコンテキストからの呼出し【NGKI1424】
6802 ・ CPUロック状態からの呼出し【NGKI1425】
6803 E_OBJ オブジェクト状態エラー
6804 ・ 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
6805 ない【NGKI1426】
6806
6807 **【機能】**
6808
6809 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、
6810 自タスクをタスク例外処理許可状態に遷移させる。
6811
6812 **【補足説明】**
6813
6814 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
6815 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
6816 のは、アプリケーションの責任である。
6817 -----
6818 sns_tex タスク例外処理禁止状態の参照 [TI] 【NGKI1428】
6819
6820 **【C言語API】**
6821 bool_t state = sns_tex()
6822
6823 **【パラメータ】**
6824 なし
6825
6826 **【リターンパラメータ】**
6827 bool_t state タスク例外処理禁止状態
6828
6829 **【機能】**
6830
6831 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
6832 以下の通り。
6833
6834 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
6835 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
6836 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
6837 【NGKI1430】。
6838
6839 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
6840 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
6841 止フラグを参照する【NGKI1431】。
6842
6843 **【補足説明】**
6844
6845 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
6846 クに一致する。
6847 -----
6848 ref_tex タスク例外処理の状態参照 [T] 【NGKI1432】
6849
6850 **【C言語API】**

```

6851     ER ercd = ref_tex(ID tskid, T_RTEX *pk_rtex)
6852
6853     【パラメータ】
6854         ID          tskid          対象タスクのID番号
6855         T_RTEX *    pk_rtex        タスク例外処理の現在状態を入れるパケットへ
6856                                     のポインタ
6857
6858     【リターンパラメータ】
6859         ER          ercd          正常終了 (E_OK) またはエラーコード
6860
6861     *タスク例外処理の現在状態 (パケットの内容)
6862         STAT        texstat        タスク例外処理の状態
6863         TEXPTN      pndptn        タスクの保留例外要因
6864
6865     【エラーコード】
6866         E_CTX        コンテキストエラー
6867                     ・非タスクコンテキストからの呼出し 【NGKI1433】
6868                     ・CPUロック状態からの呼出し 【NGKI1434】
6869         E_ID          不正ID番号
6870                     ・tskidが有効範囲外 【NGKI1435】
6871         E_NOEXS       オブジェクト未登録
6872                     ・対象タスクが未登録 [D] 【NGKI1436】
6873         E_OACV        オブジェクトアクセス違反
6874                     ・対象タスクに対する参照操作が許可されていない [P] 【NGKI1437】
6875         E_MACV        メモリアクセス違反
6876                     ・pk_rtexが指すメモリ領域への書き込みアクセスが許可されて
6877                     いない [P] 【NGKI1438】
6878         E_OBJ         オブジェクト状態エラー
6879                     ・対象タスクが休止状態 【NGKI1439】
6880                     ・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
6881                     ない 【NGKI1440】
6882
6883     【機能】
6884
6885     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
6886     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
6887     【NGKI1441】.
6888
6889     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
6890     かの値が返される 【NGKI1442】.
6891
6892         TTEX_ENA      0x01U          タスク例外処理許可状態
6893         TTEX_DIS      0x02U          タスク例外処理禁止状態
6894
6895     pndptnには, 対象タスクの現在の保留例外要因が返される 【NGKI1443】.
6896
6897     tskidにTSK_SELF (=0) を指定すると, 自タスクが対象タスクとなる
6898     【NGKI1444】.
6899     -----
6900

```

6901 4.4 同期・通信機能

6902

6903 【TOPPERS/SSPカーネルにおける規定】

6904

6905 SSPカーネルでは、同期・通信機能をサポートしない【SSPS0127】。

6906

6907 【 μ ITRON4.0仕様との関係】

6908

6909 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
6910 デブ機能をサポートすることに変更する可能性もある。

6911

6912 4.4.1 セマフォ

6913

6914 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、
6915 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
6916 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
6917 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】。

6918

6919 各セマフォが持つ情報は次の通り【NGKI1446】。

6920

- 6921 ・セマフォ属性
- 6922 ・資源数（の現在値）
- 6923 ・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）
- 6924 ・初期資源数
- 6925 ・最大資源数
- 6926 ・アクセス許可ベクタ（保護機能対応カーネルの場合）
- 6927 ・属する保護ドメイン（保護機能対応カーネルの場合）
- 6928 ・属するクラス（マルチプロセッサ対応カーネルの場合）

6929

6930 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
6931 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
6932 である。

6933

6934 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
6935 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
6936 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時、
6937 E_QOVRエラーとなる【NGKI1447】。

6938

6939 セマフォ属性には、次の属性を指定することができる【NGKI1448】。

6940

6941 TA_TPRI 0x01U 待ち行列をタスクの優先度順にする

6942

6943 TA_TPRIを指定しない場合、待ち行列はFIFO順になる【NGKI1449】。

6944

6945 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。

6946

6947 TMAX_MAXSEM セマフォの最大資源数の最大値（=UINT_MAX）【NGKI1450】

6948

6949 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
6950 では、静的APIによって登録されたセマフォの数に一致）

6951 【NGKI1451】

6952

6953 【 μ ITRON4.0仕様との関係】

6954

6955 TNUM_SEMIDは、 μ ITRON4.0仕様に規定されていないカーネル構成マクロである。

6956 -----

6957 CRE_SEM セマフォの生成 [S] 【NGKI1452】

6958 acre_sem セマフォの生成 [TD] 【NGKI1453】

6959

6960 【静的API】

6961 CRE_SEM(ID semid, { ATR sematr, uint_t isemcnt, uint_t maxsem })

6962

6963 【C言語API】

6964 ER_ID semid = acre_sem(const T_CSEM *pk_csem)

6965

6966 【パラメータ】

6967 ID semid 生成するセマフォのID番号 (CRE_SEMの場合)

6968 T_CSEM * pk_csem セマフォの生成情報を入れたパケットへのポイン
6969 タ (静的APIを除く)

6970

6971 *セマフォの生成情報 (パケットの内容)

6972 ATR sematr セマフォ属性

6973 uint_t isemcnt セマフォの初期資源数

6974 uint_t maxsem セマフォの最大資源数

6975

6976 【リターンパラメータ】

6977 ER_ID semid 生成されたセマフォのID番号 (正の値) または
6978 エラーコード

6979

6980 【エラーコード】

6981 E_CTX コンテキストエラー

6982 ・非タスクコンテキストからの呼出し [s] 【NGKI1454】

6983 ・CPUロック状態からの呼出し [s] 【NGKI1455】

6984 E_RSATR 予約属性

6985 ・sematrが無効 【NGKI1456】

6986 ・属する保護ドメインの指定が有効範囲外 [sP] 【NGKI1457】

6987 ・属するクラスの指定が有効範囲外 [sM] 【NGKI1458】

6988 ・クラスの囲みの中に記述されていない [SM] 【NGKI1459】

6989 E_PAR パラメータエラー

6990 ・maxsemが0, またはTMAX_MAXSEMよりも大きい 【NGKI1468】

6991 ・isemcntがmaxsemよりも大きい 【NGKI1466】

6992 E_OACV オブジェクトアクセス違反

6993 ・システム状態に対する管理操作が許可されていない [sP]
6994 【NGKI1460】

6995 E_MACV メモリアクセス違反

6996 ・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
6997 いない [sP] 【NGKI1461】

6998 E_NOID ID番号不足

6999 ・割り付けられるセマフォIDがない [sD] 【NGKI1462】

7000 E_OBJ オブジェクト状態エラー

141

7051

7052 **【C言語API】**

7053 ER ercd = sac_sem(ID semid, const ACVCT *p_acvct)

7054

7055 **【パラメータ】**

7056	ID	semid	対象セマフォのID番号
7057	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）

7058

7059

7060

7061

7062

7063

7064

7065

7066

7067

7068

7069

7070

7071

7072

7073

7074

7075

7076

7077

7078

7079

7080

7081

7082

7083

7084

7085

7086

7087

7088

7089

7090

7091

7092

7093

7094

7095

7096

7097

7098

7099

7100

* アクセス許可ベクタ（パケットの内容）

ACPTN	acptn1	通常操作1のアクセス許可パターン
ACPTN	acptn2	通常操作2のアクセス許可パターン
ACPTN	acptn3	管理操作のアクセス許可パターン
ACPTN	acptn4	参照操作のアクセス許可パターン

【リターンパラメータ】

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

【エラーコード】

E_CTX	コンテキストエラー
	・ 非タスクコンテキストからの呼出し [s] 【NGKI1475】
	・ CPUロック状態からの呼出し [s] 【NGKI1476】
E_ID	不正ID番号
	・ semidが有効範囲外 [s] 【NGKI1477】
E_RSATR	予約属性
	・ 対象セマフォが属する保護ドメインの囲みの中に記述され ていない [S] 【NGKI1478】
	・ 対象セマフォが属するクラスの囲みの中に記述されてい ない [SM] 【NGKI1479】
E_NOEXS	オブジェクト未登録
	・ 対象セマフォが未登録 【NGKI1480】
E_OACV	オブジェクトアクセス違反
	・ 対象セマフォに対する管理操作が許可されていない [s] 【NGKI1481】
E_MACV	メモリアクセス違反
	・ p_acvctが指すメモリ領域への読出しアクセスが許可されて いない [s] 【NGKI1482】
E_OBJ	オブジェクト状態エラー
	・ 対象セマフォは静的APIで生成された [s] 【NGKI1483】
	・ 対象セマフォに対してアクセス許可ベクタが設定済み [S] 【NGKI1484】

【機能】

semidで指定したセマフォ（対象セマフォ）のアクセス許可ベクタ（4つのアク
セス許可パターンの組）を、各パラメータで指定した値に設定する
【NGKI1485】 .

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

7101
7102 **【TOPPERS/ASPカーネルにおける規定】**
7103
7104 ASPカーネルでは、SAC_SEM, sac_semをサポートしない【ASPS0121】。
7105
7106 **【TOPPERS/FMPカーネルにおける規定】**
7107
7108 FMPカーネルでは、SAC_SEM, sac_semをサポートしない【FMPS0113】。
7109
7110 **【TOPPERS/HRP2カーネルにおける規定】**
7111
7112 HRP2カーネルでは、SAC_SEMのみをサポートする【HRPS0113】。
7113 -----
7114 del_sem セマフォの削除 [TD] 【NGKI1487】
7115
7116 **【C言語API】**
7117 ER ercd = del_sem(ID semid)
7118
7119 **【パラメータ】**
7120 ID semid 対象セマフォのID番号
7121
7122 **【リターンパラメータ】**
7123 ER ercd 正常終了 (E_OK) またはエラーコード
7124
7125 **【エラーコード】**
7126 E_CTX コンテキストエラー
7127 ・非タスクコンテキストからの呼出し【NGKI1488】
7128 ・CPUロック状態からの呼出し【NGKI1489】
7129 E_ID 不正ID番号
7130 ・semidが有効範囲外【NGKI1490】
7131 E_NOEXS オブジェクト未登録
7132 ・対象セマフォが未登録【NGKI1491】
7133 E_OACV オブジェクトアクセス違反
7134 ・対象セマフォに対する管理操作が許可されていない [P]
7135 【NGKI1492】
7136 E_OBJ オブジェクト状態エラー
7137 ・対象セマフォは静的APIで生成された【NGKI1493】
7138
7139 **【機能】**
7140
7141 semidで指定したセマフォ（対象セマフォ）を削除する。具体的な振舞いは以下の通り。
7142
7143 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
7144 【NGKI1494】。また、対象セマフォの待ち行列につながれたタスクは、待ち行列の先頭のタスクから順に待ち解除される【NGKI1495】。待ち解除されたタスクには、待ち状態となったサービスコールからE_DLTエラーが返る【NGKI1496】。
7148
7149 **【使用上の注意】**
7150

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

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、del_semをサポートしない【ASPS0122】。ただし、動的生成機能拡張パッケージでは、del_semをサポートする【ASPS0123】。

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、del_semをサポートしない【FMPS0114】。

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、del_semをサポートしない【HRPS0114】。

sig_sem	セマフォの資源の返却 [T]	【NGKI1497】
isig_sem	セマフォの資源の返却 [I]	【NGKI1498】

【C言語API】

```
ER ercd = sig_sem(ID semid)
ER ercd = isig_sem(ID semid)
```

【パラメータ】

ID	semid	対象セマフォのID番号
----	-------	-------------

【リターンパラメータ】

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

【エラーコード】

E_CTX	コンテキストエラー
	・非タスクコンテキストからの呼出し (sig_semの場合) 【NGKI1499】
	・タスクコンテキストからの呼出し (isig_semの場合) 【NGKI1500】
	・CPUロック状態からの呼出し 【NGKI1501】
E_ID	不正ID番号
	・semidが有効範囲外 【NGKI1502】
E_NOEXS	オブジェクト未登録
	・対象セマフォが未登録 [D] 【NGKI1503】
E_OACV	オブジェクトアクセス違反
	・対象セマフォに対する通常操作1が許可されていない (sig_semの場合) [P] 【NGKI1504】
E_QOVR	キューイングオーバーフロー
	・条件については機能の項を参照

【機能】

semidで指定したセマフォ（対象セマフォ）に資源を返却する。具体的な振舞いは以下の通り。

7201
 7202 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
 7203 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
 7204 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
 7205 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
 7206 【NGKI1507】。
 7207
 7208 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
 7209 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
 7210 には、E_QOVRエラーとなる【NGKI1509】。
 7211 -----
 7212 wai_sem セマフォの資源の獲得 [T] 【NGKI1510】
 7213 pol_sem セマフォの資源の獲得（ポーリング） [T] 【NGKI1511】
 7214 twai_sem セマフォの資源の獲得（タイムアウト付き） [T] 【NGKI1512】
 7215
 7216 【C言語API】
 7217 ER ercd = wai_sem(ID semid)
 7218 ER ercd = pol_sem(ID semid)
 7219 ER ercd = twai_sem(ID semid, TMO tmout)
 7220
 7221 【パラメータ】
 7222 ID semid 対象セマフォのID番号
 7223 TMO tmout タイムアウト時間（twai_semの場合）
 7224
 7225 【リターンパラメータ】
 7226 ER ercd 正常終了（E_OK）またはエラーコード
 7227
 7228 【エラーコード】
 7229 E_CTX コンテキストエラー
 7230 ・非タスクコンテキストからの呼出し【NGKI1513】
 7231 ・CPUロック状態からの呼出し【NGKI1514】
 7232 ・ディスパッチ保留状態からの呼出し（pol_semを除く）【NGKI1515】
 7233 E_NOSPT 未サポート機能
 7234 ・制約タスクからの呼出し（pol_semを除く）【NGKI1516】
 7235 E_ID 不正ID番号
 7236 ・semidが有効範囲外【NGKI1517】
 7237 E_PAR パラメータエラー
 7238 ・tmoutが無効（twai_semの場合）【NGKI1518】
 7239 E_NOEXS オブジェクト未登録
 7240 ・対象セマフォが未登録 [D] 【NGKI1519】
 7241 E_OACV オブジェクトアクセス違反
 7242 ・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]
 7243 【NGKI1520】
 7244 E_TMOUT ポーリング失敗またはタイムアウト（wai_semを除く）【NGKI1521】
 7245 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）
 7246 【NGKI1522】
 7247 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）
 7248 【NGKI1523】
 7249
 7250 【機能】

7251
7252 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
7253 いは以下の通り．
7254
7255 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
7256 【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
7257 となり，対象セマフォの待ち行列につながれる【NGKI1525】．
7258 -----
7259 ini_sem セマフォの再初期化 [T] 【NGKI1526】
7260
7261 【C言語API】
7262 ER ercd = ini_sem(ID semid)
7263
7264 【パラメータ】
7265 ID semid 対象セマフォのID番号
7266
7267 【リターンパラメータ】
7268 ER ercd 正常終了 (E_OK) またはエラーコード
7269
7270 【エラーコード】
7271 E_CTX コンテキストエラー
7272 ・ 非タスクコンテキストからの呼出し 【NGKI1527】
7273 ・ CPUロック状態からの呼出し 【NGKI1528】
7274 E_ID 不正ID番号
7275 ・ semidが有効範囲外 【NGKI1529】
7276 E_NOEXS オブジェクト未登録
7277 ・ 対象セマフォが未登録 [D] 【NGKI1530】
7278 E_OACV オブジェクトアクセス違反
7279 ・ 対象セマフォに対する管理操作が許可されていない [P]
7280 【NGKI1531】
7281
7282 【機能】
7283
7284 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
7285 以下の通り．
7286
7287 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
7288 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
7289 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
7290 たサービスコールからE_DLTエラーが返る【NGKI1534】．
7291
7292 【使用上の注意】
7293
7294 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
7295 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
7296 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
7297 み禁止時間が長くなるため，注意が必要である．
7298
7299 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
7300 プリケーションの責任である．

7301

7302 **【 μ ITRON4.0仕様との関係】**

7303

7304 μ ITRON4.0仕様に定義されていないサービスコールである。

7305

7306 ref_sem セマフォの状態参照 [T] **【NGKI1535】**

7307

7308 **【C言語API】**

7309 ER ercd = ref_sem(ID semid, T_RSEM *pk_rsem)

7310

7311 **【パラメータ】**

7312 ID semid 対象セマフォのID番号

7313 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
7314 ンタ

7315

7316 **【リターンパラメータ】**

7317 ER ercd 正常終了 (E_OK) またはエラーコード

7318

7319 *セマフォの現在状態 (パケットの内容)

7320 ID wtskid セマフォの待ち行列の先頭のタスクのID番号

7321 uint_t semcnt セマフォの資源数

7322

7323 **【エラーコード】**

7324 E_CTX コンテキストエラー

7325 ・非タスクコンテキストからの呼出し **【NGKI1536】**7326 ・CPUロック状態からの呼出し **【NGKI1537】**

7327 E_ID 不正ID番号

7328 ・semidが有効範囲外 **【NGKI1538】**

7329 E_NOEXS オブジェクト未登録

7330 ・対象セマフォが未登録 [D] **【NGKI1539】**

7331 E_OACV オブジェクトアクセス違反

7332 ・対象セマフォに対する参照操作が許可されていない [P]

7333 **【NGKI1540】**

7334 E_MACV メモリアクセス違反

7335 ・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
7336 いない [P] **【NGKI1541】**

7337

7338 **【機能】**

7339

7340 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
7341 在状態は, pk_rsemで指定したパケットに返される **【NGKI1542】**.

7342

7343 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 (=
7344 0) が返る **【NGKI1543】**.

7345

7346 **【使用上の注意】**

7347

7348 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
7349 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
7350 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

7351 化している可能性があるためである。
 7352 -----
 7353

7354 4.4.2 イベントフラグ 7355

7356 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
 7357 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
 7358 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
 7359 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、
 7360 イベントフラグIDと呼ぶID番号によって識別する【NGKI1545】。
 7361

7362 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
 7363 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
 7364 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、
 7365 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。
 7366

7367 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
 7368 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
 7369 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
 7370 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
 7371 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
 7372 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
 7373 待ち解除の条件と呼ぶ。
 7374

7375 各イベントフラグが持つ情報は次の通り【NGKI1548】。
 7376

- 7377 ・ イベントフラグ属性
- 7378 ・ ビットパターン（の現在値）
- 7379 ・ 待ち行列（イベントフラグ待ち状態のタスクのキュー）
- 7380 ・ 初期ビットパターン
- 7381 ・ アクセス許可ベクタ（保護機能対応カーネルの場合）
- 7382 ・ 属する保護ドメイン（保護機能対応カーネルの場合）
- 7383 ・ 属するクラス（マルチプロセッサ対応カーネルの場合）
 7384

7385 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
 7386 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。
 7387 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、
 7388 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）
 7389 【NGKI1549】。
 7390

7391 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
 7392 化した際の、ビットパターンの初期値である。
 7393

7394 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】。
 7395

7396	TA_TPRI	0x01U	待ち行列をタスクの優先度順にする
7397	TA_WMUL	0x02U	複数のタスクが待つのを許す
7398	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

7399

7400 TA_TPRIを指定しない場合、待ち行列はFIFO順になる【NGKI1551】。TA_WMULを

7401 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
7402 【NGKI1552】。
7403
7404 TA_CLRを指定した場合、タスクの待ち解除時に、イベントフラグのビットパターンを0にクリアする【NGKI1553】。TA_CLRを指定しない場合、タスクの待ち解除時にイベントフラグをクリアしない【NGKI1554】。
7405
7406
7407
7408 イベントフラグ機能に用いるデータ型は次の通り。
7409
7410 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
7411 定義）【NGKI1555】
7412
7413 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。
7414
7415 TBIT_FLGPTN イベントフラグのビット数（FLGPTNの有効ビット数）
7416 【NGKI1556】
7417
7418 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
7419 ネルでは、静的APIによって登録されたイベントフラグの
7420 数に一致）【NGKI1557】
7421
7422 【TOPPERS/ASPカーネルにおける規定】
7423
7424 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
7425 【ASPS0124】。
7426
7427 【TOPPERS/FMPカーネルにおける規定】
7428
7429 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
7430 【FMPS0115】。
7431
7432 【TOPPERS/HRP2カーネルにおける規定】
7433
7434 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
7435 【HRPS0115】。
7436
7437 【 μ ITRON4.0仕様との関係】
7438
7439 TNUM_FLGIDは、 μ ITRON4.0仕様に規定されていないカーネル構成マクロである。
7440 -----
7441 CRE_FLG イベントフラグの生成 [S] 【NGKI1558】
7442 acre_flg イベントフラグの生成 [TD] 【NGKI1559】
7443
7444 【静的API】
7445 CRE_FLG(ID flgid, { ATR flgatr, FLGPTN iflgptn })
7446
7447 【C言語API】
7448 ER_ID flgid = acre_flg(const T_CFLG *pk_cflg)
7449
7450 【パラメータ】

7451	ID	flgid	生成するイベントフラグのID番号（CRE_FLGの場合）
7452			
7453	T_CFLG *	pk_cflg	イベントフラグの生成情報を入れたパケットへのポインタ（静的APIを除く）
7454			
7455			
7456	* イベントフラグの生成情報（パケットの内容）		
7457	ATR	flgatr	イベントフラグ属性
7458	FLGPTN	iflgptn	イベントフラグの初期ビットパターン
7459			
7460	【リターンパラメータ】		
7461	ER_ID	flgid	生成されたイベントフラグのID番号（正の値）
7462			またはエラーコード
7463			
7464	【エラーコード】		
7465	E_CTX	コンテキストエラー	
7466		・ 非タスクコンテキストからの呼出し [s] 【NGKI1560】	
7467		・ CPUロック状態からの呼出し [s] 【NGKI1561】	
7468	E_RSATR	予約属性	
7469		・ flgatrが無効 【NGKI1562】	
7470		・ 属する保護ドメインの指定が有効範囲外 [sP] 【NGKI1563】	
7471		・ 属するクラスの指定が有効範囲外 [sM] 【NGKI1564】	
7472		・ クラスの囲みの中に記述されていない [SM] 【NGKI1565】	
7473	E_OACV	オブジェクトアクセス違反	
7474		・ システム状態に対する管理操作が許可されていない [sP]	
7475		【NGKI1566】	
7476	E_MACV	メモリアクセス違反	
7477		・ pk_cflgが指すメモリ領域への読出しアクセスが許可されていない [sP] 【NGKI1567】	
7478			
7479	E_NOID	ID番号不足	
7480		・ 割り付けられるイベントフラグIDがない [sD] 【NGKI1568】	
7481	E_OBJ	オブジェクト状態エラー	
7482		・ flgidで指定したイベントフラグが登録済み（CRE_FLGの場合）	
7483		【NGKI1569】	
7484			
7485	【機能】		
7486			
7487	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		
7488	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		
7489	に、待ち行列は空の状態に初期化される 【NGKI1570】．		
7490			
7491	静的APIにおいては、flgidはオブジェクト識別名、iflgptnは整数定数式パラメータである 【NGKI1571】．		
7492			
7493			
7494	【TOPPERS/ASPカーネルにおける規定】		
7495			
7496	ASPカーネルでは、CRE_FLGのみをサポートする 【ASPS0125】．ただし、動的生成機能拡張パッケージでは、acre_flgもサポートする 【ASPS0126】．		
7497			
7498			
7499	【TOPPERS/FMPカーネルにおける規定】		
7500			

7501 FMPカーネルでは、CRE_FLGのみをサポートする【FMPS0116】。
7502
7503 【TOPPERS/HRP2カーネルにおける規定】
7504
7505 HRP2カーネルでは、CRE_FLGのみをサポートする【HRPS0116】。
7506 -----
7507 AID_FLG 割付け可能なイベントフラグIDの数の指定〔SD〕【NGKI1572】
7508
7509 【静的API】
7510 AID_FLG(uint_t noflg)
7511
7512 【パラメータ】
7513 uint_t noflg 割付け可能なイベントフラグIDの数
7514
7515 【エラーコード】
7516 E_RSATR 予約属性
7517 ・クラスの囲みの中に記述されていない〔M〕【NGKI1573】
7518
7519 【機能】
7520
7521 noflgで指定した数のイベントフラグIDを、イベントフラグを生成するサービス
7522 コールによって割付け可能なイベントフラグIDとして確保する【NGKI1574】。
7523
7524 noflgは整数定数式パラメータである【NGKI1575】。
7525 -----
7526 SAC_FLG イベントフラグのアクセス許可ベクタの設定〔SP〕【NGKI1576】
7527 sac_flg イベントフラグのアクセス許可ベクタの設定〔TPD〕【NGKI1577】
7528
7529 【静的API】
7530 SAC_FLG(ID flgid, { ACPTN acptn1, ACPTN acptn2,
7531 ACPTN acptn3, ACPTN acptn4 })
7532
7533 【C言語API】
7534 ER ercd = sac_flg(ID flgid, const ACVCT *p_acvct)
7535
7536 【パラメータ】
7537 ID flgid 対象イベントフラグのID番号
7538 ACVCT * p_acvct アクセス許可ベクタを入れたパケットへのポ
7539 インタ（静的APIを除く）
7540
7541 *アクセス許可ベクタ（パケットの内容）
7542 ACPTN acptn1 通常操作1のアクセス許可パターン
7543 ACPTN acptn2 通常操作2のアクセス許可パターン
7544 ACPTN acptn3 管理操作のアクセス許可パターン
7545 ACPTN acptn4 参照操作のアクセス許可パターン
7546
7547 【リターンパラメータ】
7548 ER ercd 正常終了（E_OK）またはエラーコード
7549
7550 【エラーコード】

7551	E_CTX	コンテキストエラー
7552		・非タスクコンテキストからの呼出し [s] 【NGKI1578】
7553		・CPUロック状態からの呼出し [s] 【NGKI1579】
7554	E_ID	不正ID番号
7555		・flgidが有効範囲外 [s] 【NGKI1580】
7556	E_RSATR	予約属性
7557		・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
7558		述されていない [S] 【NGKI1581】
7559		・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
7560		ていない [SM] 【NGKI1582】
7561	E_NOEXS	オブジェクト未登録
7562		・対象イベントフラグが未登録 【NGKI1583】
7563	E_OACV	オブジェクトアクセス違反
7564		・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]
7565		【NGKI1584】
7566	E_MACV	メモリアクセス違反
7567		・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
7568		いない [s] 【NGKI1585】
7569	E_OBJ	オブジェクト状態エラー
7570		・対象イベントフラグは静的APIで生成された [s] 【NGKI1586】
7571		・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
7572		み [S] 【NGKI1587】
7573		
7574		【機能】
7575		
7576		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
7577		(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
7578		【NGKI1588】 .
7579		
7580		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
7581		式パラメータである 【NGKI1589】 .
7582		
7583		【TOPPERS/ASPカーネルにおける規定】
7584		
7585		ASPカーネルでは、SAC_FLG, sac_flgをサポートしない 【ASPS0127】 .
7586		
7587		【TOPPERS/FMPカーネルにおける規定】
7588		
7589		FMPカーネルでは、SAC_FLG, sac_flgをサポートしない 【FMPS0117】 .
7590		
7591		【TOPPERS/HRP2カーネルにおける規定】
7592		
7593		HRP2カーネルでは、SAC_FLGのみをサポートする 【HRPS0117】 .
7594		-----
7595	del_flg	イベントフラグの削除 [TD] 【NGKI1590】
7596		
7597		【C言語API】
7598		ER ercd = del_flg(ID flgid)
7599		
7600		【パラメータ】

7601	ID	flgid	対象イベントフラグのID番号
7602			
7603	【リターンパラメータ】		
7604	ER	ercd	正常終了 (E_OK) またはエラーコード
7605			
7606	【エラーコード】		
7607	E_CTX	コンテキストエラー	
7608		・ 非タスクコンテキストからの呼出し 【NGKI1591】	
7609		・ CPUロック状態からの呼出し 【NGKI1592】	
7610	E_ID	不正ID番号	
7611		・ flgidが有効範囲外 【NGKI1593】	
7612	E_NOEXS	オブジェクト未登録	
7613		・ 対象イベントフラグが未登録 【NGKI1594】	
7614	E_OACV	オブジェクトアクセス違反	
7615		・ 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]	
7616		【NGKI1595】	
7617	E_OBJ	オブジェクト状態エラー	
7618		・ 対象イベントフラグは静的APIで生成された 【NGKI1596】	
7619			
7620	【機能】		
7621			
7622	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		
7623	振舞いは以下の通り.		
7624			
7625	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		
7626	に戻る 【NGKI1597】. また, 対象イベントフラグの待ち行列につながれた		
7627	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		
7628	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		
7629	返る 【NGKI1599】.		
7630			
7631	【使用上の注意】		
7632			
7633	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		
7634	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		
7635	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		
7636	み禁止時間が長くなるため, 注意が必要である.		
7637			
7638	【TOPPERS/ASPカーネルにおける規定】		
7639			
7640	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		
7641	機能拡張パッケージでは, del_flgをサポートする 【ASPS0129】.		
7642			
7643	【TOPPERS/FMPカーネルにおける規定】		
7644			
7645	FMPカーネルでは, del_flgをサポートしない 【FMPS0118】.		
7646			
7647	【TOPPERS/HRP2カーネルにおける規定】		
7648			
7649	HRP2カーネルでは, del_flgをサポートしない 【HRPS0118】.		
7650	-----		

7651 set_flg イベントフラグのセット [T] 【NGKI1600】
 7652 iset_flg イベントフラグのセット [I] 【NGKI1601】
 7653
 7654 **【C言語API】**
 7655 ER ercd = set_flg(ID flgid, FLGPTN setptn)
 7656 ER ercd = iset_flg(ID flgid, FLGPTN setptn)
 7657
 7658 **【パラメータ】**
 7659 ID flgid 対象イベントフラグのID番号
 7660 FLGPTN setptn セットするビットパターン
 7661
 7662 **【リターンパラメータ】**
 7663 ER ercd 正常終了 (E_OK) またはエラーコード
 7664
 7665 **【エラーコード】**
 7666 E_CTX コンテキストエラー
 7667 ・非タスクコンテキストからの呼出し (set_flgの場合) 【NGKI1602】
 7668 ・タスクコンテキストからの呼出し (iset_flgの場合) 【NGKI1603】
 7669 ・CPUロック状態からの呼出し 【NGKI1604】
 7670 E_ID 不正ID番号
 7671 ・flgidが有効範囲外 【NGKI1605】
 7672 E_NOEXS オブジェクト未登録
 7673 ・対象イベントフラグが未登録 [D] 【NGKI1606】
 7674 E_OACV オブジェクトアクセス違反
 7675 ・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
 7676 (set_flgの場合) [P] 【NGKI1607】
 7677
 7678 **【機能】**
 7679
 7680 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
 7681 をセットする. 具体的な振舞いは以下の通り.
 7682
 7683 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
 7684 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
 7685 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
 7686 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】.
 7687 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
 7688 【NGKI1610】.
 7689
 7690 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
 7691 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
 7692 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.
 7693
 7694 **【使用上の注意】**
 7695
 7696 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
 7697 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,
 7698 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
 7699 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
 7700 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.

```

7701 -----
7702 clr_flg      イベントフラグのクリア [T]  【NGKI1611】
7703
7704 【C言語API】
7705     ER ercd = clr_flg(ID flgid, FLGPTN clrptn)
7706
7707 【パラメータ】
7708     ID          flgid      対象イベントフラグのID番号
7709     FLGPTN      clrptn     クリアするビットパターン（クリアしないビッ
7710                             トを1, クリアするビットを0とする）
7711
7712 【リターンパラメータ】
7713     ER          ercd       正常終了 (E_OK) またはエラーコード
7714
7715 【エラーコード】
7716     E_CTX       コンテキストエラー
7717                 ・非タスクコンテキストからの呼出し 【NGKI1612】
7718                 ・CPUロック状態からの呼出し 【NGKI1613】
7719     E_ID        不正ID番号
7720                 ・flgidが有効範囲外 【NGKI1614】
7721     E_NOEXS     オブジェクト未登録
7722                 ・対象イベントフラグが未登録 [D]  【NGKI1615】
7723     E_OACV      オブジェクトアクセス違反
7724                 ・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]
7725                 【NGKI1616】
7726
7727 【機能】
7728
7729 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
7730 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
7731 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
7732 【NGKI1617】．
7733 -----
7734 wai_flg      イベントフラグ待ち [T]  【NGKI1618】
7735 pol_flg      イベントフラグ待ち（ポーリング） [T]  【NGKI1619】
7736 twai_flg     イベントフラグ待ち（タイムアウト付き） [T]  【NGKI1620】
7737
7738 【C言語API】
7739     ER ercd = wai_flg(ID flgid, FLGPTN waiptn, MODE wfmode, FLGPTN *p_flgptn)
7740     ER ercd = pol_flg(ID flgid, FLGPTN waiptn, MODE wfmode, FLGPTN *p_flgptn)
7741     ER ercd = twai_flg(ID flgid, FLGPTN waiptn,
7742                         MODE wfmode, FLGPTN *p_flgptn, TMO tmout)
7743
7744 【パラメータ】
7745     ID          flgid      対象イベントフラグのID番号
7746     FLGPTN      waiptn     待ちビットパターン
7747     MODE        wfmode     待ちモード
7748     FLGPTN *    p_flgptn   待ち解除時のビットパターンを入れるメモリ領
7749                             域へのポインタ
7750     TMO         tmout      タイムアウト時間（twai_flgの場合）

```

7751
7752 **【リターンパラメータ】**
7753 ER ercd 正常終了 (E_OK) またはエラーコード
7754 FLGPTN flgptn 待ち解除時のビットパターン
7755
7756 **【エラーコード】**
7757 E_CTX コンテキストエラー
7758 ・非タスクコンテキストからの呼出し【NGKI1621】
7759 ・CPUロック状態からの呼出し【NGKI1622】
7760 ・ディスパッチ保留状態からの呼出し (pol_flgを除く)【NGKI1623】
7761 E_NOSPT 未サポート機能
7762 ・制約タスクからの呼出し (pol_flgを除く)【NGKI1624】
7763 E_ID 不正ID番号
7764 ・flgidが有効範囲外【NGKI1625】
7765 E_PAR パラメータエラー
7766 ・waitptnが0【NGKI1626】
7767 ・wfmodeが無効 (TWF_ORWまたはTWF_ANDWでない)【NGKI1627】
7768 ・tmoutが無効 (twai_flgの場合)【NGKI1628】
7769 E_NOEXS オブジェクト未登録
7770 ・対象イベントフラグが未登録 [D]【NGKI1629】
7771 E_OACV オブジェクトアクセス違反
7772 ・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]
7773 【NGKI1630】
7774 E_MACV メモリアクセス違反
7775 ・p_flgptnが指すメモリ領域への書込みアクセスが許可され
7776 ていない [P]【NGKI1631】
7777 E_ILUSE サービスコール不正使用
7778 ・TA_WMUL属性でないイベントフラグで待ちタスクあり【NGKI1632】
7779 E_TMOUT ポーリング失敗またはタイムアウト (wai_flgを除く)【NGKI1633】
7780 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)
7781 【NGKI1634】
7782 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)
7783 【NGKI1635】
7784
7785 **【機能】**
7786
7787 flgidで指定したイベントフラグ (対象イベントフラグ) が, waitptnとwfmodeで
7788 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.
7789
7790 対象イベントフラグが, waitptnとwfmodeで指定した待ち解除の条件を満たして
7791 いる場合には, 対象イベントフラグのビットパターンの現在値がflgptnに返さ
7792 れる【NGKI1636】. 対象イベントフラグがTA_CLR属性である場合には, 対象イ
7793 ベントフラグのビットパターンが0にクリアされる【NGKI1637】.
7794
7795 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
7796 態となり, 対象イベントフラグの待ち行列につながる【NGKI1638】.
7797 -----
7798 ini_flg イベントフラグの再初期化 [T]【NGKI1639】
7799
7800 **【C言語API】**

7801 ER ercd = ini_flg(ID flgid)

7802

7803 **【パラメータ】**

7804 ID flgid 対象イベントフラグのID番号

7805

7806 **【リターンパラメータ】**

7807 ER ercd 正常終了 (E_OK) またはエラーコード

7808

7809 **【エラーコード】**

7810 E_CTX コンテキストエラー

7811 ・非タスクコンテキストからの呼出し【NGKI1640】

7812 ・CPUロック状態からの呼出し【NGKI1641】

7813 E_ID 不正ID番号

7814 ・flgidが有効範囲外【NGKI1642】

7815 E_NOEXS オブジェクト未登録

7816 ・対象イベントフラグが未登録 [D] 【NGKI1643】

7817 E_OACV オブジェクトアクセス違反

7818 ・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]

7819 【NGKI1644】

7820

7821 **【機能】**

7822

7823 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
7824 的な振舞いは以下の通り.

7825

7826 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

7827 【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,

7828 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

7829 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

7830 【NGKI1647】.

7831

7832 **【使用上の注意】**

7833

7834 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

7835 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

7836 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

7837 み禁止時間が長くなるため, 注意が必要である.

7838

7839 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

7840 は, アプリケーションの責任である.

7841

7842 **【μ ITRON4.0仕様との関係】**

7843

7844 μ ITRON4.0仕様に定義されていないサービスコールである.

7845 -----

7846 ref_flg イベントフラグの状態参照 [T] 【NGKI1648】

7847

7848 **【C言語API】**

7849 ER ercd = ref_flg(ID flgid, T_RFLG *pk_rflg)

7850

7851 **【パラメータ】**

7852 ID flgid 対象イベントフラグのID番号
 7853 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
 7854 のポインタ

7855

7856 **【リターンパラメータ】**

7857 ER ercd 正常終了 (E_OK) またはエラーコード

7858

7859 * イベントフラグの現在状態 (パケットの内容)

7860 ID wtsskid イベントフラグの待ち行列の先頭のタスクのID
 7861 番号

7862 uint_t flgptn イベントフラグのビットパターン

7863

7864 **【エラーコード】**

7865 E_CTX コンテキストエラー
 7866 ・ 非タスクコンテキストからの呼出し 【NGKI1649】

7867 ・ CPUロック状態からの呼出し 【NGKI1650】

7868 E_ID 不正ID番号

7869 ・ flgidが有効範囲外 【NGKI1651】

7870 E_NOEXS オブジェクト未登録

7871 ・ 対象イベントフラグが未登録 [D] 【NGKI1652】

7872 E_OACV オブジェクトアクセス違反

7873 ・ 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
 7874 [P] 【NGKI1653】

7875 E_MACV メモリアクセス違反

7876 ・ pk_rflgが指すメモリ領域への書き込みアクセスが許可されて
 7877 いない [P] 【NGKI1654】

7878

7879 **【機能】**

7880

7881 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。
 7882 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】。

7883

7884 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、wtsskidには
 7885 TSK_NONE (=0) が返る 【NGKI1656】。

7886

7887 **【使用上の注意】**

7888

7889 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
 7890 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
 7891 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
 7892 グの状態が変化している可能性があるためである。

7893

7894

7895 4.4.3 データキュー

7896

7897 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
 7898 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
 7899 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
 7900 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

7901 号によって識別する【NGKI1657】。

7902

7903 各データキューが持つ情報は次の通り【NGKI1658】。

7904

7905 ・データキュー属性

7906 ・データキュー管理領域

7907 ・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）

7908 ・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）

7909 ・アクセス許可ベクタ（保護機能対応カーネルの場合）

7910 ・属する保護ドメイン（保護機能対応カーネルの場合）

7911 ・属するクラス（マルチプロセッサ対応カーネルの場合）

7912

7913 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

7914 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

7915 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

7916 ズを0とすることができる【NGKI1659】。

7917

7918 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

7919 オブジェクト管理領域として扱われる【NGKI1660】。

7920

7921 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

7922 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

7923 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

7924 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

7925 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。

7926

7927 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】。

7928

7929 TA_TPRI 0x01U 送信待ち行列をタスクの優先度順にする

7930

7931 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

7932 ち行列は、FIFO順に固定されている【NGKI1663】。

7933

7934 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。

7935

7936 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

7937 ネルでは、静的APIによって登録されたデータキューの数

7938 に一致）【NGKI1664】

7939

7940 【 μ ITRON4.0仕様との関係】

7941

7942 TNUM_DTQIDは、 μ ITRON4.0仕様に規定されていないカーネル構成マクロである。

7943 -----

7944 CRE_DTQ データキューの生成 [S] 【NGKI1665】

7945 acre_dtq データキューの生成 [TD] 【NGKI1666】

7946

7947 【静的API】

7948 CRE_DTQ(ID dtqid, { ATR dtqatr, uint_t dtqcnt, void *dtqmb })

7949

7950 【C言語API】

```

7951     ER_ID dtqid = acre_dtq(const T_CDTQ *pk_cdtq)
7952
7953     【パラメータ】
7954         ID          dtqid      生成するデータキューのID番号 (CRE_DTQの場合)
7955         T_CDTQ *    pk_cdtq    データキューの生成情報を入れたパケットへの
7956                                 ポインタ (静的APIを除く)
7957
7958     * データキューの生成情報 (パケットの内容)
7959         ATR          dtqatr     データキュー属性
7960         uint_t       dtqcnt     データキュー管理領域に格納できるデータ数
7961         void *       dtqmb      データキュー管理領域の先頭番地
7962
7963     【リターンパラメータ】
7964         ER_ID        dtqid      生成されたデータキューのID番号 (正の値) ま
7965                                 たはエラーコード
7966
7967     【エラーコード】
7968         E_CTX        コンテキストエラー
7969                     ・ 非タスクコンテキストからの呼出し [s] 【NGKI1667】
7970                     ・ CPUロック状態からの呼出し [s] 【NGKI1668】
7971         E_RSATR      予約属性
7972                     ・ dtqatrが無効 【NGKI1669】
7973                     ・ 属する保護ドメインの指定が有効範囲外 [sP] 【NGKI1670】
7974                     ・ 属するクラスの指定が有効範囲外 [sM] 【NGKI1671】
7975                     ・ クラスの囲みの中に記述されていない [SM] 【NGKI1672】
7976         E_NOSPT      未サポート機能
7977                     ・ 条件については各カーネルにおける規定の項を参照
7978         E_PAR        パラメータエラー
7979                     ・ 条件については機能の項を参照
7980         E_OACV       オブジェクトアクセス違反
7981                     ・ システム状態に対する管理操作が許可されていない [sP]
7982                     【NGKI1673】
7983         E_MACV       メモリアクセス違反
7984                     ・ pk_cdtqが指すメモリ領域への読出しアクセスが許可されて
7985                     いない [sP] 【NGKI1674】
7986         E_NOID       ID番号不足
7987                     ・ 割り付けられるデータキューIDがない [sD] 【NGKI1675】
7988         E_NOMEM      メモリ不足
7989                     ・ データキュー管理領域が確保できない 【NGKI1676】
7990         E_OBJ        オブジェクト状態エラー
7991                     ・ dtqidで指定したデータキューが登録済み (CRE_DTQの場合)
7992                     【NGKI1677】
7993                     ・ その他の条件については機能の項を参照
7994
7995     【機能】
7996
7997     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
7998     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
7999     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
8000     列は、空の状態に初期化される【NGKI1679】。

```


8001
8002 静的APIにおいては、dtqidはオブジェクト識別名、dtqcntは整数定数式パラメータ、dtqmbは一般定数式パラメータである【NGKI1680】。コンフィギュレータは、
8003 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
8004 【NGKI1681】。
8005
8006
8007 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
8008 管理領域を、コンフィギュレータまたはカーネルが確保する【NGKI1682】。
8009
8010 [dtqmbにNULL以外を指定した場合]
8011
8012 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
8013 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
8014 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
8015 【NGKI1684】。
8016
8017 TSZ_DTQMB(dtqcnt) dtqcntで指定した数のデータを格納できるデータ
8018 キュー管理領域のサイズ (バイト数)
8019 TCNT_DTQMB(dtqcnt) dtqcntで指定した数のデータを格納できるデータ
8020 キュー管理領域を確保するために必要なMB_T型の配
8021 列の要素数
8022
8023 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】。
8024
8025 MB_T <データキュー管理領域の変数名>[TCNT_DTQMB(dtqcnt)];
8026
8027 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】。
8028
8029 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
8030 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
8031 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
8032 クトに含まれない場合、E_OBJエラーとなる【NGKI1688】。
8033
8034 【TOPPERS/ASPカーネルにおける規定】
8035
8036 ASPカーネルでは、CRE_DTQのみをサポートする【ASPS0130】。また、dtqmbには
8037 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
8038 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
8039 サポートする【ASPS0133】。acre_dtqに対しては、dtqmbにNULL以外を指定でき
8040 ないという制限はない【ASPS0134】。
8041
8042 【TOPPERS/FMPカーネルにおける規定】
8043
8044 FMPカーネルでは、CRE_DTQのみをサポートする【FMPS0119】。また、dtqmbには
8045 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
8046 となる【FMPS0121】。
8047
8048 【TOPPERS/HRP2カーネルにおける規定】
8049
8050 HRP2カーネルでは、CRE_DTQのみをサポートする【HRPS0119】。また、dtqmbに

8051 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
8052 ラーとなる【HRPS0121】。

8053

8054 【 μ ITRON4.0仕様との関係】

8055

8056 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、
8057 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
8058 頭番地）に改名した。また、TSZ_DTQをTSZ_DTQMBに改名した。

8059

8060 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
8061 法を規定した。

8062

8063 AID_DTQ 割付け可能なデータキューIDの数の指定 [SD] 【NGKI1689】

8064

8065 【静的API】

8066 AID_DTQ(uint_t nodtq)

8067

8068 【パラメータ】

8069 uint_t nodtq 割付け可能なデータキューIDの数

8070

8071 【エラーコード】

8072 E_RSATR 予約属性

8073 ・クラスの囲みの中に記述されていない [M] 【NGKI1690】

8074

8075 【機能】

8076

8077 nodtqで指定した数のデータキューIDを、データキューを生成するサービスコー
8078 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】。

8079

8080 nodtqは整数定数式パラメータである【NGKI1692】。

8081

8082 SAC_DTQ データキューのアクセス許可ベクタの設定 [SP] 【NGKI1693】

8083 sac_dtq データキューのアクセス許可ベクタの設定 [TPD] 【NGKI1694】

8084

8085 【静的API】

8086 SAC_DTQ(ID dtqid, { ACPTN acptn1, ACPTN acptn2,
8087 ACPTN acptn3, ACPTN acptn4 })

8088

8089 【C言語API】

8090 ER ercd = sac_dtq(ID dtqid, const ACVCT *p_acvct)

8091

8092 【パラメータ】

8093 ID dtqid 対象データキューのID番号
8094 ACVCT * p_acvct アクセス許可ベクタを入れたパケットへのポ
8095 インタ（静的APIを除く）

8096

8097 *アクセス許可ベクタ（パケットの内容）

8098 ACPTN acptn1 通常操作1のアクセス許可パターン

8099 ACPTN acptn2 通常操作2のアクセス許可パターン

8100 ACPTN acptn3 管理操作のアクセス許可パターン

8101	ACPTN	acptn4	参照操作のアクセス許可パターン
8102			
8103	【リターンパラメータ】		
8104	ER	ercd	正常終了 (E_OK) またはエラーコード
8105			
8106	【エラーコード】		
8107	E_CTX	コンテキストエラー	
8108			・非タスクコンテキストからの呼出し [s] 【NGKI1695】
8109			・CPUロック状態からの呼出し [s] 【NGKI1696】
8110	E_ID	不正ID番号	
8111			・dtqidが有効範囲外 [s] 【NGKI1697】
8112	E_RSATR	予約属性	
8113			・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
8114			されていない [S] 【NGKI1698】
8115			・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
8116			いない [SM] 【NGKI1699】
8117	E_NOEXS	オブジェクト未登録	
8118			・対象データキューが未登録 【NGKI1700】
8119	E_OACV	オブジェクトアクセス違反	
8120			・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]
8121			【NGKI1701】
8122	E_MACV	メモリアクセス違反	
8123			・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
8124			いない [s] 【NGKI1702】
8125	E_OBJ	オブジェクト状態エラー	
8126			・対象データキューは静的APIで生成された [s] 【NGKI1703】
8127			・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]
8128			【NGKI1704】
8129			
8130	【機能】		
8131			
8132	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		
8133	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		
8134	【NGKI1705】。		
8135			
8136	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		
8137	式パラメータである 【NGKI1706】。		
8138			
8139	【TOPPERS/ASPカーネルにおける規定】		
8140			
8141	ASPカーネルでは、SAC_DTQ, sac_dtqをサポートしない 【ASPS0135】。		
8142			
8143	【TOPPERS/FMPカーネルにおける規定】		
8144			
8145	FMPカーネルでは、SAC_DTQ, sac_dtqをサポートしない 【FMPS0122】。		
8146			
8147	【TOPPERS/HRP2カーネルにおける規定】		
8148			
8149	HRP2カーネルでは、SAC_DTQのみをサポートする 【HRPS0122】。		
8150	-----		

8151 del_dtq データキューの削除 [TD] 【NGKI1707】
8152
8153 【C言語API】
8154 ER ercd = del_dtq(ID dtqid)
8155
8156 【パラメータ】
8157 ID dtqid 対象データキューのID番号
8158
8159 【リターンパラメータ】
8160 ER ercd 正常終了 (E_OK) またはエラーコード
8161
8162 【エラーコード】
8163 E_CTX コンテキストエラー
8164 ・非タスクコンテキストからの呼出し 【NGKI1708】
8165 ・CPUロック状態からの呼出し 【NGKI1709】
8166 E_ID 不正ID番号
8167 ・dtqidが有効範囲外 【NGKI1710】
8168 E_NOEXS オブジェクト未登録
8169 ・対象データキューが未登録 【NGKI1711】
8170 E_OACV オブジェクトアクセス違反
8171 ・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]
8172 【NGKI1712】
8173 E_OBJ オブジェクト状態エラー
8174 ・対象データキューは静的APIで生成された 【NGKI1713】
8175
8176 【機能】
8177
8178 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
8179 いは以下の通り．
8180
8181 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
8182 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
8183 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
8184 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
8185 コールからE_DLTエラーが返る 【NGKI1716】．
8186
8187 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
8188 た場合は，そのメモリ領域が解放される 【NGKI1717】．
8189
8190 【補足説明】
8191
8192 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，
8193 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
8194 い．
8195
8196 【使用上の注意】
8197
8198 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
8199 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
8200 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

8201 み禁止時間が長くなるため、注意が必要である。

8202

8203 【TOPPERS/ASPカーネルにおける規定】

8204

8205 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
8206 機能拡張パッケージでは、del_dtqをサポートする【ASPS0137】。

8207

8208 【TOPPERS/FMPカーネルにおける規定】

8209

8210 FMPカーネルでは、del_dtqをサポートしない【FMPS0123】。

8211

8212 【TOPPERS/HRP2カーネルにおける規定】

8213

8214 HRP2カーネルでは、del_dtqをサポートしない【HRPS0123】。

8215

8216 snd_dtq データキューへの送信 [T] 【NGKI1718】
8217 psnd_dtq データキューへの送信（ポーリング） [T] 【NGKI1719】
8218 ipsnd_dtq データキューへの送信（ポーリング） [I] 【NGKI1720】
8219 tsnd_dtq データキューへの送信（タイムアウト付き） [T] 【NGKI1721】

8220

8221 【C言語API】

8222 ER ercd = snd_dtq(ID dtqid, intptr_t data)
8223 ER ercd = psnd_dtq(ID dtqid, intptr_t data)
8224 ER ercd = ipsnd_dtq(ID dtqid, intptr_t data)
8225 ER ercd = tsnd_dtq(ID dtqid, intptr_t data, TMO tmout)

8226

8227 【パラメータ】

8228	ID	dtqid	対象データキューのID番号
8229	intptr_t	data	送信データ
8230	TMO	tmout	タイムアウト時間（tsnd_dtqの場合）

8231

8232 【リターンパラメータ】

8233	ER	ercd	正常終了（E_OK）またはエラーコード
------	----	------	---------------------

8234

8235 【エラーコード】

8236	E_CTX	コンテキストエラー
8237		・非タスクコンテキストからの呼出し（ipsnd_dtqを除く）
8238		【NGKI1722】
8239		・タスクコンテキストからの呼出し（ipsnd_dtqの場合）
8240		【NGKI1723】
8241		・CPUロック状態からの呼出し【NGKI1724】
8242		・ディスパッチ保留状態からの呼出し（snd_dtqとtsnd_dtqの
8243		場合）【NGKI1725】
8244	E_NOSPT	未サポート機能
8245		・制約タスクからの呼出し（snd_dtqとtsnd_dtqの場合）【NGKI1726】
8246	E_ID	不正ID番号
8247		・dtqidが有効範囲外【NGKI1727】
8248	E_PAR	パラメータエラー
8249		・tmoutが無効（tsnd_dtqの場合）【NGKI1728】
8250	E_NOEXS	オブジェクト未登録

8251 ・対象データキューが未登録 [D] 【NGKI1729】
8252 E_OACV オブジェクトアクセス違反
8253 ・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
8254 (ipsnd_dtqを除く) [P] 【NGKI1730】
8255 E_TMOUT ポーリング失敗またはタイムアウト (snd_dtqを除く) 【NGKI1731】
8256 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
8257 の場合) 【NGKI1732】
8258 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
8259 の場合) 【NGKI1733】
8260
8261 **【機能】**
8262
8263 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
8264 を送信する. 具体的な振舞いは以下の通り.
8265
8266 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
8267 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
8268 【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
8269 からE_OKが返る【NGKI1735】.
8270
8271 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
8272 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
8273 順でデータキュー管理領域に格納される【NGKI1736】.
8274
8275 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
8276 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
8277 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
8278 【NGKI1737】.
8279 -----
8280 fsnd_dtq データキューへの強制送信 [T] 【NGKI1738】
8281 ifsnd_dtq データキューへの強制送信 [I] 【NGKI1739】
8282
8283 **【C言語API】**
8284 ER ercd = fsnd_dtq(ID dtqid, intptr_t data)
8285 ER ercd = ifsnd_dtq(ID dtqid, intptr_t data)
8286
8287 **【パラメータ】**
8288 ID dtqid 対象データキューのID番号
8289 intptr_t data 送信データ
8290
8291 **【リターンパラメータ】**
8292 ER ercd 正常終了 (E_OK) またはエラーコード
8293
8294 **【エラーコード】**
8295 E_CTX コンテキストエラー
8296 ・非タスクコンテキストからの呼出し (fsnd_dtqの場合) 【NGKI1740】
8297 ・タスクコンテキストからの呼出し (ifsnd_dtqの場合) 【NGKI1741】
8298 ・CPUロック状態からの呼出し 【NGKI1742】
8299 E_ID 不正ID番号
8300 ・dtqidが有効範囲外 【NGKI1743】

8301 E_NOEXS オブジェクト未登録
 8302 ・対象データキューが未登録 [D] 【NGKI1744】
 8303 E_OACV オブジェクトアクセス違反
 8304 ・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
 8305 (fsnd_dtqの場合) [P] 【NGKI1745】
 8306 E_ILUSE サービスコール不正使用
 8307 ・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】
 8308

8309 【機能】

8310
 8311 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
 8312 を強制送信する．具体的な振舞いは以下の通り．

8313
 8314 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
 8315 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

8316 【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
 8317 からE_OKが返る【NGKI1748】．

8318
 8319 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
 8320 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
 8321 順でデータキュー管理領域に格納される【NGKI1749】．

8322
 8323 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
 8324 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
 8325 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
 8326 が，FIFO順でデータキュー管理領域に格納される【NGKI1750】．

8327 -----
 8328 rcv_dtq データキューからの受信 [T] 【NGKI1751】
 8329 prev_dtq データキューからの受信（ポーリング） [T] 【NGKI1752】
 8330 trcv_dtq データキューからの受信（タイムアウト付き） [T] 【NGKI1753】
 8331

8332 【C言語API】

8333 ER ercd = rcv_dtq(ID dtqid, intptr_t *p_data)
 8334 ER ercd = prev_dtq(ID dtqid, intptr_t *p_data)
 8335 ER ercd = trcv_dtq(ID dtqid, intptr_t *p_data, TMO tmout)
 8336

8337 【パラメータ】

8338 ID dtqid 対象データキューのID番号
 8339 intptr_t * p_data 受信データを入れるメモリ領域へのポインタ
 8340 TMO tmout タイムアウト時間（trcv_dtqの場合）
 8341

8342 【リターンパラメータ】

8343 ER ercd 正常終了（E_OK）またはエラーコード
 8344 intptr_t data 受信データ
 8345

8346 【エラーコード】

8347 E_CTX コンテキストエラー
 8348 ・非タスクコンテキストからの呼出し【NGKI1754】
 8349 ・CPUロック状態からの呼出し【NGKI1755】
 8350 ・ディスパッチ保留状態からの呼出し（prev_dtqを除く）

8351 【NGKI1756】
8352 E_NOSPT 未サポート機能
8353 ・制約タスクからの呼出し（prcv_dtqを除く）【NGKI1757】
8354 E_ID 不正ID番号
8355 ・dtqidが有効範囲外【NGKI1758】
8356 E_PAR パラメータエラー
8357 ・tmoutが無効（trcv_dtqの場合）【NGKI1759】
8358 E_NOEXS オブジェクト未登録
8359 ・対象データキューが未登録 [D] 【NGKI1760】
8360 E_OACV オブジェクトアクセス違反
8361 ・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]
8362 【NGKI1761】
8363 E_MACV メモリアクセス違反
8364 ・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
8365 いない [P] 【NGKI1762】
8366 E_TMOUT ポーリング失敗またはタイムアウト（rcv_dtqを除く）【NGKI1763】
8367 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）
8368 【NGKI1764】
8369 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）
8370 【NGKI1765】

8372 【機能】

8373
8374 dtqidで指定したデータキュー（対象データキュー）からデータを受信する．受
8375 信したデータは、p_dataで指定したメモリ領域に返される．具体的な振舞いは
8376 以下の通り．
8377

8378 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、
8379 データキュー管理領域の先頭に格納されたデータが取り出され、p_dataで指定
8380 したメモリ領域に返される【NGKI1766】．また、送信待ち行列にタスクが存在
8381 する場合には、送信待ち行列の先頭のタスクの送信データが、FIFO順でデータ
8382 キュー管理領域に格納され、そのタスクは待ち解除される【NGKI1767】．待ち
8383 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
8384 【NGKI1768】．
8385

8386 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
8387 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
8388 タが、p_dataで指定したメモリ領域に返される【NGKI1769】．送信待ち行列の
8389 先頭のタスクは、待ち解除される．待ち解除されたタスクには、待ち状態となっ
8390 たサービスコールからE_OKが返る【NGKI1770】．
8391

8392 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
8393 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
8394 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される【NGKI1771】．
8395 -----

8396 ini_dtq データキューの再初期化 [T] 【NGKI1772】
8397

8398 【C言語API】

8399 ER ercd = ini_dtq(ID dtqid)
8400

8401 **【パラメータ】**
8402 ID dtqid 対象データキューのID番号
8403
8404 **【リターンパラメータ】**
8405 ER ercd 正常終了 (E_OK) またはエラーコード
8406
8407 **【エラーコード】**
8408 E_CTX コンテキストエラー
8409 ・非タスクコンテキストからの呼出し【NGKI1773】
8410 ・CPUロック状態からの呼出し【NGKI1774】
8411 E_ID 不正ID番号
8412 ・dtqidが有効範囲外【NGKI1775】
8413 E_NOEXS オブジェクト未登録
8414 ・対象データキューが未登録 [D]【NGKI1776】
8415 E_OACV オブジェクトアクセス違反
8416 ・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]
8417 【NGKI1777】
8418
8419 **【機能】**
8420
8421 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
8422 振舞いは以下の通り．
8423
8424 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
8425 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
8426 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
8427 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
8428 たサービスコールからE_DLTエラーが返る【NGKI1780】．
8429
8430 **【補足説明】**
8431
8432 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，
8433 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
8434 い．
8435
8436 **【使用上の注意】**
8437
8438 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
8439 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
8440 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
8441 み禁止時間が長くなるため，注意が必要である．
8442
8443 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，
8444 アプリケーションの責任である．
8445
8446 **【μ ITRON4.0仕様との関係】**
8447
8448 μ ITRON4.0仕様に定義されていないサービスコールである．
8449 -----
8450 ref_dtq データキューの状態参照 [T]【NGKI1781】

8451
8452 **【C言語API】**
8453 ER ercd = ref_dtq(ID dtqid, T_RDTQ *pk_rdtq)
8454
8455 **【パラメータ】**
8456 ID dtqid 対象データキューのID番号
8457 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
8458 ポインタ
8459
8460 **【リターンパラメータ】**
8461 ER ercd 正常終了 (E_OK) またはエラーコード
8462
8463 *データキューの現在状態 (パケットの内容)
8464 ID stskid データキューの送信待ち行列の先頭のタスクの
8465 ID番号
8466 ID rtskid データキューの受信待ち行列の先頭のタスクの
8467 ID番号
8468 uint_t sdtqcnt データキュー管理領域に格納されているデータ
8469 の数
8470
8471 **【エラーコード】**
8472 E_CTX コンテキストエラー
8473 ・非タスクコンテキストからの呼出し【NGKI1782】
8474 ・CPUロック状態からの呼出し【NGKI1783】
8475 E_ID 不正ID番号
8476 ・dtqidが有効範囲外【NGKI1784】
8477 E_NOEXS オブジェクト未登録
8478 ・対象データキューが未登録 [D] 【NGKI1785】
8479 E_OACV オブジェクトアクセス違反
8480 ・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]
8481 【NGKI1786】
8482 E_MACV メモリアクセス違反
8483 ・pk_rdtqが指すメモリ領域への書込みアクセスが許可されて
8484 いない [P] 【NGKI1787】
8485
8486 **【機能】**
8487
8488 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
8489 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】。
8490
8491 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
8492 TSK_NONE (=0) が返る【NGKI1789】。また、受信待ち行列にタスクが存在しな
8493 い場合、rtskidにはTSK_NONE (=0) が返る【NGKI1790】。
8494
8495 **【使用上の注意】**
8496
8497 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
8498 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
8499 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
8500 状態が変化している可能性があるためである。

4.4.4 優先度データキュー

優先度データキューは、1ワードのデータをメッセージとして、データの優先度順で送受信するための同期・通信カーネルオブジェクトである。より大きいサイズのメッセージを送受信したい場合には、メッセージを置いたメモリ領域へのポインタを1ワードのデータとして送受信する方法がある。優先度データキューは、優先度データキューIDと呼ぶID番号によって識別する【NGKI1791】。

各優先度データキューが持つ情報は次の通り【NGKI1792】。

- ・優先度データキュー属性
- ・優先度データキュー管理領域
- ・送信待ち行列（優先度データキューへの送信待ち状態のタスクのキュー）
- ・受信待ち行列（優先度データキューからの受信待ち状態のタスクのキュー）
- ・送信できるデータ優先度の最大値
- ・アクセス許可ベクタ（保護機能対応カーネルの場合）
- ・属する保護ドメイン（保護機能対応カーネルの場合）
- ・属するクラス（マルチプロセッサ対応カーネルの場合）

優先度データキュー管理領域は、優先度データキューに送信されたデータを、データの優先度順に格納しておくためのメモリ領域である。優先度データキュー生成時に、優先度データキュー管理領域に格納できるデータ数を0とすることで、優先度データキュー管理領域のサイズを0とすることができる【NGKI1793】。

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

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

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

TA_TPRI	0x01U	送信待ち行列をタスクの優先度順にする
---------	-------	--------------------

TA_TPRIを指定しない場合、送信待ち行列はFIFO順になる【NGKI1796】。受信待ち行列は、FIFO順に固定されている【NGKI1797】。

優先度データキュー機能に関連するカーネル構成マクロは次の通り。

TMIN_DPRI	データ優先度の最小値（=1）	【NGKI1798】
TMAX_DPRI	データ優先度の最大値	
TNUM_PDQID	登録できる優先度データキューの数（動的生成対応でないカーネルでは、静的APIによって登録された優先度デー	

172

8601	uint_t	pdqcnt	優先度データキュー管理領域に格納できるデータ数
8602			
8603	PRI	maxdpri	優先度データキューに送信できるデータ優先度の最大値
8604			
8605	void *	pdqmb	優先度データキュー管理領域の先頭番地
8606			
8607	【リターンパラメータ】		
8608	ER_ID	pdqid	生成された優先度データキューのID番号（正の値）またはエラーコード
8609			
8610			
8611	【エラーコード】		
8612	E_CTX	コンテキストエラー	
8613			・非タスクコンテキストからの呼出し [s] 【NGKI1802】
8614			・CPUロック状態からの呼出し [s] 【NGKI1803】
8615	E_RSATR	予約属性	
8616			・pdqatrが無効 【NGKI1804】
8617			・属する保護ドメインの指定が有効範囲外 [sP] 【NGKI1805】
8618			・属するクラスの指定が有効範囲外 [sM] 【NGKI1806】
8619			・クラスの囲みの中に記述されていない [SM] 【NGKI1807】
8620	E_NOSPT	未サポート機能	
8621			・条件については各カーネルにおける規定の項を参照
8622	E_PAR	パラメータエラー	
8623			・maxdpriがTMIN_DPRIより小さい、またはTMAX_DPRIより大きい 【NGKI1819】
8624			
8625	E_OACV	オブジェクトアクセス違反	
8626			・システム状態に対する管理操作が許可されていない [sP] 【NGKI1808】
8627			
8628	E_MACV	メモリアクセス違反	
8629			・pk_cpdqが指すメモリ領域への読出しアクセスが許可されていない [sP] 【NGKI1809】
8630			
8631	E_NOID	ID番号不足	
8632			・割り付けられる優先度データキューIDがない [sD] 【NGKI1810】
8633	E_NOMEM	メモリ不足	
8634			・優先度データキュー管理領域が確保できない 【NGKI1811】
8635	E_OBJ	オブジェクト状態エラー	
8636			・pdqidで指定した優先度データキューが登録済み（CRE_PDQの場合） 【NGKI1812】
8637			・その他の条件については機能の項を参照
8638			
8639			
8640	【機能】		
8641			
8642	各パラメータで指定した優先度データキュー生成情報に従って、優先度データキューを生成する。pdqcntとpdqmbから優先度データキュー管理領域が設定され、格納されているデータがない状態に初期化される 【NGKI1813】。また、送信待ち行列と受信待ち行列は、空の状態に初期化される 【NGKI1814】。		
8643			
8644			
8645			
8646			
8647	静的APIにおいては、pdqidはオブジェクト識別名、pdqcntとmaxdpriは整数定数式パラメータ、pdqmbは一般定数式パラメータである 【NGKI1815】。コンフィギュレータは、静的APIのメモリ不足（E_NOMEM）エラーを検出することができない 【NGKI1816】。		
8648			
8649			
8650			

8651
8652 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
8653 タキュー管理領域を、コンフィギュレータまたはカーネルが確保する
8654 【NGKI1817】。
8655
8656 [pdqmbにNULL以外を指定した場合]
8657
8658 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
8659 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
8660 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
8661 用意している【NGKI1821】。
8662
8663 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
8664 タキュー管理領域のサイズ (バイト数)
8665 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
8666 タキュー管理領域を確保するために必要なMB_T型の
8667 配列の要素数
8668
8669 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
8670 【NGKI1822】。
8671
8672 MB_T <優先度データキュー管理領域の変数名>[TCNT_PDQMB(pdqcnt)];
8673
8674 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
8675 【NGKI1823】。
8676
8677 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
8678 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
8679 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
8680 オブジェクトに含まれない場合、E_OBJエラーとなる【NGKI1825】。
8681
8682 【TOPPERS/ASPカーネルにおける規定】
8683
8684 ASPカーネルでは、CRE_PDQのみをサポートする【ASPS0140】。また、pdqmbには
8685 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
8686 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
8687 サポートする【ASPS0143】。acre_pdqに対しては、pdqmbにNULL以外を指定でき
8688 ないという制限はない【ASPS0144】。
8689
8690 【TOPPERS/FMPカーネルにおける規定】
8691
8692 FMPカーネルでは、CRE_PDQのみをサポートする【FMPS0125】。また、pdqmbには
8693 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
8694 となる【FMPS0127】。
8695
8696 【TOPPERS/HRP2カーネルにおける規定】
8697
8698 HRP2カーネルでは、CRE_PDQのみをサポートする【HRPS0125】。また、pdqmbに
8699 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
8700 ラーとなる【HRPS0127】。

```

8701 -----
8702 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】
8703
8704 【静的API】
8705     AID_PDQ(uint_t nopdq)
8706
8707 【パラメータ】
8708     uint_t      nopdq      割付け可能な優先度データキューIDの数
8709
8710 【エラーコード】
8711     E_RSATR      予約属性
8712                  ・クラスの囲みの中に記述されていない [M] 【NGKI1827】
8713
8714 【機能】
8715
8716 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
8717 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
8718 【NGKI1828】 .
8719
8720 nopdqは整数定数式パラメータである 【NGKI1829】 .
8721 -----
8722 SAC_PDQ      優先度データキューのアクセス許可ベクタの設定 [SP] 【NGKI1830】
8723 sac_pdq      優先度データキューのアクセス許可ベクタの設定 [TPD] 【NGKI1831】
8724
8725 【静的API】
8726     SAC_PDQ(ID pdqid, { ACPTN acptn1, ACPTN acptn2,
8727                        ACPTN acptn3, ACPTN acptn4 })
8728
8729 【C言語API】
8730     ER ercd = sac_pdq(ID pdqid, const ACVCT *p_acvct)
8731
8732 【パラメータ】
8733     ID      pdqid      対象優先度データキューのID番号
8734     ACVCT *  p_acvct    アクセス許可ベクタを入れたパケットへのポ
8735                        インタ（静的APIを除く）
8736
8737     *アクセス許可ベクタ（パケットの内容）
8738     ACPTN    acptn1     通常操作1のアクセス許可パターン
8739     ACPTN    acptn2     通常操作2のアクセス許可パターン
8740     ACPTN    acptn3     管理操作のアクセス許可パターン
8741     ACPTN    acptn4     参照操作のアクセス許可パターン
8742
8743 【リターンパラメータ】
8744     ER      ercd      正常終了 (E_OK) またはエラーコード
8745
8746 【エラーコード】
8747     E_CTX      コンテキストエラー
8748                  ・非タスクコンテキストからの呼出し [s] 【NGKI1832】
8749                  ・CPUロック状態からの呼出し [s] 【NGKI1833】
8750     E_ID      不正ID番号

```

8751 ・pdqidが有効範囲外 [s] 【NGKI1834】
8752 E_RSATR 予約属性
8753 ・対象優先度データキューが属する保護ドメインの囲みの中
8754 に記述されていない [S] 【NGKI1835】
8755 ・対象優先度データキューが属するクラスの囲みの中に記述
8756 されていない [SM] 【NGKI1836】
8757 E_NOEXS オブジェクト未登録
8758 ・対象優先度データキューが未登録 【NGKI1837】
8759 E_OACV オブジェクトアクセス違反
8760 ・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
8761 ない [s] 【NGKI1838】
8762 E_MACV メモリアクセス違反
8763 ・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
8764 いない [s] 【NGKI1839】
8765 E_OBJ オブジェクト状態エラー
8766 ・対象優先度データキューは静的APIで生成された [s] 【NGKI1840】
8767 ・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
8768 定済み [S] 【NGKI1841】
8769
8770 **【機能】**
8771
8772 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
8773 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
8774 設定する 【NGKI1842】 .
8775
8776 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
8777 式パラメータである 【NGKI1843】 .
8778
8779 **【TOPPERS/ASPカーネルにおける規定】**
8780
8781 ASPカーネルでは、SAC_PDQ, sac_pdqをサポートしない 【ASPS0145】 .
8782
8783 **【TOPPERS/FMPカーネルにおける規定】**
8784
8785 FMPカーネルでは、SAC_PDQ, sac_pdqをサポートしない 【FMPS0128】 .
8786
8787 **【TOPPERS/HRP2カーネルにおける規定】**
8788
8789 HRP2カーネルでは、SAC_PDQのみをサポートする 【HRPS0128】 .
8790
8791 del_pdq 優先度データキューの削除 [TD] 【NGKI1844】
8792
8793 **【C言語API】**
8794 ER ercd = del_pdq(ID pdqid)
8795
8796 **【パラメータ】**
8797 ID pdqid 対象優先度データキューのID番号
8798
8799 **【リターンパラメータ】**
8800 ER ercd 正常終了 (E_OK) またはエラーコード

【エラーコード】

- E_CTX コンテキストエラー
・非タスクコンテキストからの呼出し【NGKI1845】
・CPUロック状態からの呼出し【NGKI1846】
- E_ID 不正ID番号
・pdqidが有効範囲外【NGKI1847】
- E_NOEXS オブジェクト未登録
・対象優先度データキューが未登録【NGKI1848】
- E_OACV オブジェクトアクセス違反
・対象優先度データキューに対する管理操作が許可されていない【P】【NGKI1849】
- E_OBJ オブジェクト状態エラー
・対象優先度データキューは静的APIで生成された【NGKI1850】

【機能】

pdqidで指定した優先度データキュー（対象優先度データキュー）を削除する。
具体的な振舞いは以下の通り。

対象優先度データキューの登録が解除され、その優先度データキューIDが未使用の状態に戻される【NGKI1851】。また、対象優先度データキューの送信待ち行列と受信待ち行列につながれたタスクは、それぞれの待ち行列の先頭のタスクから順に待ち解除される【NGKI1852】。待ち解除されたタスクには、待ち状態となったサービスコールからE_DLTエラーが返る【NGKI1853】。

優先度データキューの生成時に、優先度データキュー管理領域がカーネルによって確保された場合は、そのメモリ領域が解放される【NGKI1854】。

【補足説明】

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

【使用上の注意】

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

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、del_pdqをサポートしない【ASPS0146】。ただし、動的生成機能拡張パッケージでは、del_pdqをサポートする【ASPS0147】。

【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、del_pdqをサポートしない【FMPS0129】。

8851

8852 **【TOPPERS/HRP2カーネルにおける規定】**

8853

8854 HRP2カーネルでは、del_pdqをサポートしない【HRPS0129】。

8855

8856 snd_pdq 優先度データキューへの送信 [T] 【NGKI1855】

8857 psnd_pdq 優先度データキューへの送信（ポーリング） [T] 【NGKI1856】

8858 ipsnd_pdq 優先度データキューへの送信（ポーリング） [I] 【NGKI1857】

8859 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】

8860

8861 **【C言語API】**

8862 ER ercd = snd_pdq(ID pdqid, intptr_t data, PRI datapri)

8863 ER ercd = psnd_pdq(ID pdqid, intptr_t data, PRI datapri)

8864 ER ercd = ipsnd_pdq(ID pdqid, intptr_t data, PRI datapri)

8865 ER ercd = tsnd_pdq(ID pdqid, intptr_t data, PRI datapri, TMO tmout)

8866

8867 **【パラメータ】**

8868 ID pdqid 対象優先度データキューのID番号

8869 intptr_t data 送信データ

8870 PRI datapri 送信データの優先度

8871 TMO tmout タイムアウト時間（tsnd_pdqの場合）

8872

8873 **【リターンパラメータ】**

8874 ER ercd 正常終了（E_OK）またはエラーコード

8875

8876 **【エラーコード】**

8877 E_CTX コンテキストエラー

8878 ・非タスクコンテキストからの呼出し（ipsnd_pdqを除く）

8879 【NGKI1859】

8880 ・タスクコンテキストからの呼出し（ipsnd_pdqの場合）【NGKI1860】

8881 ・CPUロック状態からの呼出し【NGKI1861】

8882 ・ディスパッチ保留状態からの呼出し（snd_pdqとtsnd_pdqの
8883 場合）【NGKI1862】

8884 E_NOSPT 未サポート機能

8885 ・制約タスクからの呼出し（snd_pdqとtsnd_pdqの場合）【NGKI1863】

8886 E_ID 不正ID番号

8887 ・pdqidが有効範囲外【NGKI1864】

8888 E_PAR パラメータエラー

8889 ・tmoutが無効（tsnd_pdqの場合）【NGKI1865】

8890 ・その他の条件については機能の項を参照

8891 E_NOEXS オブジェクト未登録

8892 ・対象優先度データキューが未登録 [D] 【NGKI1866】

8893 E_OACV オブジェクトアクセス違反

8894 ・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
8895 ない（ipsnd_pdqを除く） [P] 【NGKI1867】

8896 E_TMOUT ポーリング失敗またはタイムアウト（snd_pdqを除く）【NGKI1868】

8897 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
8898 の場合）【NGKI1869】8899 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
8900 の場合）【NGKI1870】

8901
8902 **【機能】**
8903
8904 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
8905 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
8906 の通り．
8907
8908 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
8909 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
8910 【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
8911 からE_OKが返る【NGKI1872】．
8912
8913 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
8914 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
8915 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
8916 に格納される【NGKI1873】．
8917
8918 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
8919 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
8920 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
8921 行列につながる【NGKI1874】．
8922
8923 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
8924 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
8925 【NGKI1876】．
8926 -----
8927 rcv_pdq 優先度データキューからの受信 [T] 【NGKI1877】
8928 prcv_pdq 優先度データキューからの受信（ポーリング） [T] 【NGKI1878】
8929 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】
8930
8931 **【C言語API】**
8932 ER ercd = rcv_pdq(ID pdqid, intptr_t *p_data, PRI *p_datapri)
8933 ER ercd = prcv_pdq(ID pdqid, intptr_t *p_data, PRI *p_datapri)
8934 ER ercd = trcv_pdq(ID pdqid, intptr_t *p_data, PRI *p_datapri, TMO tmout)
8935
8936 **【パラメータ】**
8937 ID pdqid 対象優先度データキューのID番号
8938 intptr_t * p_data 受信データを入れるメモリ領域へのポインタ
8939 PRI * p_datapri 受信データの優先度を入れるメモリ領域へのポ
8940 インタ
8941 TMO tmout タイムアウト時間（trcv_pdqの場合）
8942
8943 **【リターンパラメータ】**
8944 ER ercd 正常終了（E_OK）またはエラーコード
8945 intptr_t data 受信データ
8946 PRI datapri 受信データの優先度
8947
8948 **【エラーコード】**
8949 E_CTX コンテキストエラー
8950 ・非タスクコンテキストからの呼出し【NGKI1880】

8951		・CPUロック状態からの呼出し【NGKI1881】
8952		・ディスパッチ保留状態からの呼出し（prcv_pdqを除く）【NGKI1882】
8953	E_NOSPT	未サポート機能
8954		・制約タスクからの呼出し（prcv_pdqを除く）【NGKI1883】
8955	E_ID	不正ID番号
8956		・pdqidが有効範囲外【NGKI1884】
8957	E_PAR	パラメータエラー
8958		・tmoutが無効（trcv_pdqの場合）【NGKI1885】
8959	E_NOEXS	オブジェクト未登録
8960		・対象優先度データキューが未登録 [D] 【NGKI1886】
8961	E_OACV	オブジェクトアクセス違反
8962		・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
8963		ない [P] 【NGKI1887】
8964	E_MACV	メモリアクセス違反
8965		・p_dataが指すメモリ領域への書き込みアクセスが許可されて
8966		いない [P] 【NGKI1888】
8967		・p_datapriが指すメモリ領域への書き込みアクセスが許可され
8968		ていない [P] 【NGKI1889】
8969	E_TMOUT	ポーリング失敗またはタイムアウト（rcv_pdqを除く）【NGKI1890】
8970	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）
8971		【NGKI1891】
8972	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）
8973		【NGKI1892】

【機能】

8974		
8975		
8976		
8977		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
8978		受信する．受信したデータはp_dataで指定したメモリ領域に、その優先度は
8979		p_datapriで指定したメモリ領域に返される．具体的な振舞いは以下の通り．
8980		
8981		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
8982		いる場合には、優先度データキュー管理領域の先頭に格納されたデータが取り
8983		出され、p_dataで指定したメモリ領域に返される【NGKI1893】．また、その優
8984		先度がp_datapriで指定したメモリ領域に返される【NGKI1894】．さらに、送信
8985		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
8986		タが、データの優先度順で優先度データキュー管理領域に格納され、そのタス
8987		クは待ち解除される【NGKI1895】．待ち解除されたタスクには、待ち状態となっ
8988		たサービスコールからE_OKが返る【NGKI1896】．
8989		
8990		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
8991		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
8992		スクの送信データが、p_dataで指定したメモリ領域に返される【NGKI1897】．
8993		また、その優先度がp_datapriで指定したメモリ領域に返される【NGKI1898】．
8994		送信待ち行列の先頭のタスクは、待ち解除される【NGKI1899】．待ち解除され
8995		たタスクには、待ち状態となったサービスコールからE_OKが返る【NGKI1900】．
8996		
8997		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
8998		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
8999		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
9000		につながる【NGKI1901】．

```

9001 -----
9002 ini_pdq      優先度データキューの再初期化 [T] 【NGKI1902】
9003
9004 【C言語API】
9005     ER ercd = ini_pdq(ID pdqid)
9006
9007 【パラメータ】
9008     ID          pdqid      対象優先度データキューのID番号
9009
9010 【リターンパラメータ】
9011     ER          ercd      正常終了 (E_OK) またはエラーコード
9012
9013 【エラーコード】
9014     E_CTX      コンテキストエラー
9015                ・非タスクコンテキストからの呼出し 【NGKI1903】
9016                ・CPUロック状態からの呼出し 【NGKI1904】
9017     E_ID       不正ID番号
9018                ・pdqidが有効範囲外 【NGKI1905】
9019     E_NOEXS    オブジェクト未登録
9020                ・対象優先度データキューが未登録 [D] 【NGKI1906】
9021     E_OACV     オブジェクトアクセス違反
9022                ・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
9023                  ない [P] 【NGKI1907】
9024
9025 【機能】
9026
9027 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
9028 る．具体的な振舞いは以下の通り．
9029
9030 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
9031 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
9032 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
9033 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、
9034 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】．
9035
9036 【補足説明】
9037
9038 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、
9039 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
9040 い．
9041
9042 【使用上の注意】
9043
9044 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
9045 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
9046 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
9047 み禁止時間が長くなるため、注意が必要である．
9048
9049 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
9050 つのは、アプリケーションの責任である．

```

9051 -----

9052 ref_pdq 優先度データキューの状態参照 [T] 【NGKI1911】

9053

9054 【C言語API】

9055 ER ercd = ref_pdq(ID pdqid, T_RPDQ *pk_rpdq)

9056

9057 【パラメータ】

9058 ID pdqid 対象優先度データキューのID番号

9059 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

9060 トへのポインタ

9061

9062 【リターンパラメータ】

9063 ER ercd 正常終了 (E_OK) またはエラーコード

9064

9065 * 優先度データキューの現在状態 (パケットの内容)

9066 ID stskid 優先度データキューの送信待ち行列の先頭のタ

9067 スクのID番号

9068 ID rtskid 優先度データキューの受信待ち行列の先頭のタ

9069 スクのID番号

9070 uint_t spdqcnt 優先度データキュー管理領域に格納されている

9071 データの数

9072

9073 【エラーコード】

9074 E_CTX コンテキストエラー

9075 ・ 非タスクコンテキストからの呼出し 【NGKI1912】

9076 ・ CPUロック状態からの呼出し 【NGKI1913】

9077 E_ID 不正ID番号

9078 ・ pdqidが有効範囲外 【NGKI1914】

9079 E_NOEXS オブジェクト未登録

9080 ・ 対象優先度データキューが未登録 [D] 【NGKI1915】

9081 E_OACV オブジェクトアクセス違反

9082 ・ 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

9083 ない [P] 【NGKI1916】

9084 E_MACV メモリアクセス違反

9085 ・ pk_rpdqが指すメモリ領域への書込みアクセスが許可されて

9086 いない [P] 【NGKI1917】

9087

9088 【機能】

9089

9090 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

9091 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

9092 【NGKI1918】.

9093

9094 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

9095 はTSK_NONE (=0) が返る 【NGKI1919】. また, 受信待ち行列にタスクが存在し

9096 ない場合, rtskidにはTSK_NONE (=0) が返る 【NGKI1920】.

9097

9098 【使用上の注意】

9099

9100 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

9101 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
 9102 した直後に割込みが発生した場合、ref_pdqから戻ってきた時には対象優先度デー
 9103 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。

9104 -----

9105

9106 4.4.5 メールボックス

9107

9108 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
 9109 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
 9110 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】。

9111

9112 各メールボックスが持つ情報は次の通り【NGKI1922】。

9113

- 9114 ・メールボックス属性
- 9115 ・メッセージキュー
- 9116 ・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）
- 9117 ・送信できるメッセージ優先度の最大値
- 9118 ・優先度別のメッセージキューヘッダ領域
- 9119 ・属するクラス（マルチプロセッサ対応カーネルの場合）

9120

9121 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
 9122 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。

9123

9124 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
 9125 （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
 9126 順序でつながれているキューである。

9127

9128 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】。

9129

9130	TA_TPRI	0x01U	待ち行列をタスクの優先度順にする
9131	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

9132

9133 TA_TPRIを指定しない場合、待ち行列はFIFO順になる【NGKI1924】。TA_MPRIを
 9134 指定しない場合、メッセージキューはFIFO順になる【NGKI1925】。

9135

9136 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
 9137 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。

9138

9139 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
 9140 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
 9141 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
 9142 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】。
 9143 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
 9144 ているか否かにより、以下のいずれかを用いる【NGKI1928】。

9145

9146	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
9147	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

9148

9149 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
 9150 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

9151 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
9152 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
9153 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
9154 送信した場合の動作は保証されない【NGKI1930】。
9155
9156 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
9157 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
9158 ルドに設定する【NGKI1931】。
9159
9160 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】。
9161
9162 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。
9163
9164 TMIN_MPRI メッセージ優先度の最小値（=1） 【NGKI1933】
9165 TMAX_MPRI メッセージ優先度の最大値
9166
9167 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
9168 ネルでは、静的APIによって登録されたメールボックスの
9169 数に一致）【NGKI1934】
9170
9171 【補足説明】
9172
9173 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
9174 ダ領域は用いていない。
9175
9176 【使用上の注意】
9177
9178 メールボックス機能は、 μ ITRON4.0仕様との互換性のために残した機能であり、
9179 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。
9180 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
9181 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
9182 する方法で置き換えることができる。
9183
9184 【TOPPERS/ASPカーネルにおける規定】
9185
9186 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
9187 ジ優先度の最大値（TMAX_MPRI）は16に固定されている【ASPS0148】。ただし、
9188 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】。
9189
9190 【TOPPERS/FMPカーネルにおける規定】
9191
9192 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
9193 ジ優先度の最大値（TMAX_MPRI）は16に固定されている【FMPS0131】。
9194
9195 【TOPPERS/HRP2カーネルにおける規定】
9196
9197 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】。
9198
9199 【 μ ITRON4.0仕様との関係】
9200

9201 TNUM_MBXIDは、 μ ITRON4.0仕様に規定されていないカーネル構成マクロである。
 9202 -----

9203 CRE_MBX メールボックスの生成 [Sp] 【NGKI1935】
 9204 acre_mbx メールボックスの生成 [TpD] 【NGKI1936】
 9205

9206 【静的API】
 9207 CRE_MBX(ID mbxid, { ATR mbxatr, PRI maxmpri, void *mprihd })
 9208

9209 【C言語API】
 9210 ER_ID mbxid = acre_mbx(const T_CMBX *pk_cmbx)
 9211

9212 【パラメータ】
 9213 ID mbxid 生成するメールボックスのID番号 (CRE_MBXの場合)
 9214 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)
 9215
 9216
 9217

9218 * メールボックスの生成情報 (パケットの内容)
 9219 ATR mbxatr メールボックス属性
 9220 PRI maxmpri 優先度メールボックスに送信できるメッセージ優先度の最大値
 9221 void * mprihd 優先度別のメッセージキューヘッダ領域の先頭番地
 9222
 9223
 9224

9225 【リターンパラメータ】
 9226 ER_ID mbxid 生成されたメールボックスのID番号 (正の値) またはエラーコード
 9227
 9228

9229 【エラーコード】
 9230 E_CTX コンテキストエラー
 9231 ・非タスクコンテキストからの呼出し [s] 【NGKI1937】
 9232 ・CPUロック状態からの呼出し [s] 【NGKI1938】
 9233 E_RSATR 予約属性
 9234 ・mbxatrが無効 【NGKI1939】
 9235 ・属するクラスの指定が有効範囲外 [sM] 【NGKI1940】
 9236 ・クラスの囲みの中に記述されていない [SM] 【NGKI1941】
 9237 E_NOSPT 未サポート機能
 9238 ・条件については各カーネルにおける規定の項を参照
 9239 E_PAR パラメータエラー
 9240 ・maxmpriがTMIN_MPRIより小さい, またはTMAX_MPRIより大きい 【NGKI1951】
 9241 E_NOID ID番号不足
 9242 ・割り付けられるメールボックスIDがない [sD] 【NGKI1942】
 9243 E_NOMEM メモリ不足
 9244 ・優先度別のメッセージキューヘッダ領域が確保できない 【NGKI1943】
 9245 E_OBJ オブジェクト状態エラー
 9246 ・mbxidで指定した優先度データキューが登録済み (CRE_MBXの場合) 【NGKI1944】
 9247
 9248
 9249
 9250 【機能】

9251
9252 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
9253 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
9254 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
9255 る【NGKI1945】。また、待ち行列は空の状態に初期化される【NGKI1946】。
9256
9257 静的APIにおいては、mbxidはオブジェクト識別名、maxmpriは整数定数式パラメー
9258 タ、mprihdは一般定数式パラメータである【NGKI1947】。コンフィギュレータ
9259 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
9260 【NGKI1948】。
9261
9262 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
9263 ジキューヘッダ領域を、コンフィギュレータまたはカーネルが確保する
9264 【NGKI1949】。
9265
9266 【未決定事項】
9267
9268 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
9269 ない。
9270
9271 【TOPPERS/ASPカーネルにおける規定】
9272
9273 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
9274 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
9275 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
9276 【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
9277 トする【ASPS0153】。acre_mbxに対しても、mprihdにはNULLのみを指定するこ
9278 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
9279 いため、E_NOMEMが返ることはない【ASPS0155】。
9280
9281 【TOPPERS/FMPカーネルにおける規定】
9282
9283 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
9284 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
9285 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
9286 【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、
9287 E_NOMEMが返ることはない【FMPS0135】。
9288 -----
9289 AID_MBX 割付け可能なメールボックスIDの数の指定 [SpD] 【NGKI1952】
9290
9291 【静的API】
9292 AID_MBX(uint_t nombx)
9293
9294 【パラメータ】
9295 uint_t nombx 割付け可能なメールボックスIDの数
9296
9297 【エラーコード】
9298 E_RSATR 予約属性
9299 ・クラスの囲みの中に記述されていない [M] 【NGKI1953】
9300

9301 【機能】

9302

9303 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
9304 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】.

9305

9306 nombxは整数定数式パラメータである【NGKI1955】.

9307

9308 del_mbx メールボックスの削除 [TpD] 【NGKI1956】

9309

9310 【C言語API】

9311 ER ercd = del_mbx(ID mbxid)

9312

9313 【パラメータ】

9314 ID mbxid 対象メールボックスのID番号

9315

9316 【リターンパラメータ】

9317 ER ercd 正常終了 (E_OK) またはエラーコード

9318

9319 【エラーコード】

9320 E_CTX コンテキストエラー

9321 ・非タスクコンテキストからの呼出し【NGKI1957】

9322 ・CPUロック状態からの呼出し【NGKI1958】

9323 E_ID 不正ID番号

9324 ・mbxidが有効範囲外【NGKI1959】

9325 E_NOEXS オブジェクト未登録

9326 ・対象メールボックスが未登録【NGKI1960】

9327 E_OBJ オブジェクト状態エラー

9328 ・対象メールボックスは静的APIで生成された【NGKI1961】

9329

9330 【機能】

9331

9332 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
9333 振舞いは以下の通り．

9334

9335 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
9336 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
9337 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
9338 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
9339 返る【NGKI1964】．

9340

9341 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
9342 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】．

9343

9344 【使用上の注意】

9345

9346 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
9347 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
9348 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
9349 み禁止時間が長くなるため、注意が必要である．

9350

9351 【TOPPERS/ASPカーネルにおける規定】

9352

9353 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
9354 機能拡張パッケージでは、del_mbxをサポートする【ASPS0157】。

9355

9356 【TOPPERS/FMPカーネルにおける規定】

9357

9358 FMPカーネルでは、del_mbxをサポートしない【FMPS0136】。

9359

9360 snd_mbx メールボックスへの送信 [Tp] 【NGKI1966】

9361

9362 【C言語API】

9363 ER ercd = snd_mbx(ID mbxid, T_MSG *pk_msg)

9364

9365 【パラメータ】

9366 ID mbxid 対象メールボックスのID番号

9367 T_MSG *pk_msg 送信メッセージの先頭番地

9368

9369 【リターンパラメータ】

9370 ER ercd 正常終了 (E_OK) またはエラーコード

9371

9372 【エラーコード】

9373 E_CTX コンテキストエラー

9374 ・非タスクコンテキストからの呼出し【NGKI1967】

9375 ・CPUロック状態からの呼出し【NGKI1968】

9376 E_ID 不正ID番号

9377 ・mbxidが有効範囲外【NGKI1969】

9378 E_PAR パラメータエラー

9379 ・条件については機能の項を参照

9380 E_NOEXS オブジェクト未登録

9381 ・対象メールボックスが未登録 [D] 【NGKI1970】

9382

9383 【機能】

9384

9385 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
9386 メッセージを送信する。具体的な振舞いは以下の通り。

9387

9388 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
9389 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

9390 【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
9391 からE_OKが返る【NGKI1972】。

9392

9393 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
9394 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
9395 る順序で、メッセージキューにつながる【NGKI1973】。

9396

9397 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
9398 ージの先頭のメッセージヘッダ中のmsgpriフィールドの値が、TMIN_MPRI以上で、
9399 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
9400 らない。そうでない場合には、E_PARエラーとなる【NGKI1975】。

```

9401 -----
9402 rcv_mbx      メールボックスからの受信 [Tp] 【NGKI1976】
9403 prcv_mbx     メールボックスからの受信 (ポーリング) [Tp] 【NGKI1977】
9404 trcv_mbx     メールボックスからの受信 (タイムアウト付き) [Tp] 【NGKI1978】
9405
9406 【C言語API】
9407     ER ercd = rcv_mbx(ID mbxid, T_MSG **ppk_msg)
9408     ER ercd = prcv_mbx(ID mbxid, T_MSG **ppk_msg)
9409     ER ercd = trcv_mbx(ID mbxid, T_MSG **ppk_msg, TMO tmout)
9410
9411 【パラメータ】
9412     ID          mbxid      対象メールボックスのID番号
9413     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
9414                      へのポインタ
9415     TMO          tmout     タイムアウト時間 (trcv_mbxの場合)
9416
9417 【リターンパラメータ】
9418     ER          ercd      正常終了 (E_OK) またはエラーコード
9419     T_MSG *     ppk_msg   受信メッセージの先頭番地
9420
9421 【エラーコード】
9422     E_CTX       コンテキストエラー
9423                ・非タスクコンテキストからの呼出し 【NGKI1979】
9424                ・CPUロック状態からの呼出し 【NGKI1980】
9425                ・ディスパッチ保留状態からの呼出し (prcv_mbxを除く) 【NGKI1981】
9426     E_NOSPT     未サポート機能
9427                ・制約タスクからの呼出し (prcv_mbxを除く) 【NGKI1982】
9428     E_ID        不正ID番号
9429                ・mbxidが有効範囲外 【NGKI1983】
9430     E_PAR       パラメータエラー
9431                ・tmoutが無効 (trcv_mbxの場合) 【NGKI1984】
9432     E_NOEXS     オブジェクト未登録
9433                ・対象メールボックスが未登録 [D] 【NGKI1985】
9434     E_TMOUT     ポーリング失敗またはタイムアウト (rcv_mbxを除く) 【NGKI1986】
9435     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)
9436                【NGKI1987】
9437     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)
9438                【NGKI1988】
9439
9440 【機能】
9441
9442     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
9443     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
9444     れる. 具体的な振舞いは以下の通り.
9445
9446     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
9447     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
9448     指定したメモリ領域に返される 【NGKI1989】.
9449
9450     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

```

9451 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
9452 クスの待ち行列につながる【NGKI1990】。
9453 -----

9454 ini_mbx メールボックスの再初期化 [Tp] 【NGKI1991】
9455

9456 **【C言語API】**
9457 ER ercd = ini_mbx(ID mbxid)
9458

9459 **【パラメータ】**
9460 ID mbxid 対象メールボックスのID番号
9461

9462 **【リターンパラメータ】**
9463 ER ercd 正常終了 (E_OK) またはエラーコード
9464

9465 **【エラーコード】**
9466 E_CTX コンテキストエラー
9467 ・非タスクコンテキストからの呼出し【NGKI1992】
9468 ・CPUロック状態からの呼出し【NGKI1993】
9469 E_ID 不正ID番号
9470 ・mbxidが有効範囲外【NGKI1994】
9471 E_NOEXS オブジェクト未登録
9472 ・対象メールボックスが未登録 [D] 【NGKI1995】
9473

9474 **【機能】**
9475

9476 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
9477 的な振舞いは以下の通り。
9478

9479 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
9480 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
9481 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
9482 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
9483 たサービスコールからE_DLTエラーが返る【NGKI1998】。
9484

9485 **【使用上の注意】**
9486

9487 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
9488 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
9489 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
9490 み禁止時間が長くなるため、注意が必要である。
9491

9492 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、
9493 アプリケーションの責任である。
9494

9495 **【μ ITRON4.0仕様との関係】**
9496

9497 μ ITRON4.0仕様に定義されていないサービスコールである。
9498 -----

9499 ref_mbx メールボックスの状態参照 [Tp] 【NGKI1999】
9500

9501 **【C言語API】**

9502 ER ercd = ref_mbx(ID mbxid, T_RMBX *pk_rmbx)

9503

9504 **【パラメータ】**

9505 ID mbxid 対象メールボックスのID番号
 9506 T_RMBX * pk_rmbx メールボックスの現在状態を入れるパケットへのポインタ
 9507
 9508

9509 **【リターンパラメータ】**

9510 ER ercd 正常終了 (E_OK) またはエラーコード

9511

9512 * メールボックスの現在状態 (パケットの内容)

9513 ID wtskid メールボックスの待ち行列の先頭のタスクのID番号
 9514
 9515 T_MSG * pk_msg メッセージキューの先頭につながれたメッセージの先頭番地
 9516
 9517

9518 **【エラーコード】**

9519 E_CTX コンテキストエラー
 9520 ・ 非タスクコンテキストからの呼出し【NGKI2000】
 9521 ・ CPUロック状態からの呼出し【NGKI2001】
 9522 E_ID 不正ID番号
 9523 ・ mbxidが有効範囲外【NGKI2002】
 9524 E_NOEXS オブジェクト未登録
 9525 ・ 対象メールボックスが未登録 [D]【NGKI2003】
 9526

9527 **【機能】**

9528

9529 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。
 9530 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】。
 9531

9532 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
 9533 TSK_NONE (=0) が返る【NGKI2005】。また、メッセージキューにメッセージが
 9534 つながれていない場合、pk_msgにはNULLが返る【NGKI2006】。
 9535

9536 **【使用上の注意】**

9537

9538 ref_mbxはデバッグ時向けの機能であり、その他の目的に使用することは推奨しない。
 9539 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
 9540 直後に割込みが発生した場合、ref_mbxから戻ってきた時には対象メールボックス
 9541 の状態が変化している可能性があるためである。
 9542 -----

9543

9544 **4.4.6 ミューテックス**

9545

9546 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
 9547 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
 9548 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
 9549 呼ぶID番号によって識別する【NGKI2007】。
 9550

9551 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
9552 優先度上限プロトコル (priority ceiling protocol) をサポートする。ミュー
9553 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
9554 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
9555 が行われる。

9556

9557 各ミューテックスが持つ情報は次の通り【NGKI2008】。

9558

- 9559 ・ ミューテックス属性
- 9560 ・ 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)
- 9561 ・ ミューテックスをロックしているタスク
- 9562 ・ 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)
- 9563 ・ 上限優先度 (優先度上限ミューテックスの場合)
- 9564 ・ アクセス許可ベクタ (保護機能対応カーネルの場合)
- 9565 ・ 属する保護ドメイン (保護機能対応カーネルの場合)
- 9566 ・ 属するクラス (マルチプロセッサ対応カーネルの場合)

9567

9568 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
9569 クスのロック待ち状態) のタスクが、ミューテックスをロックできる順序でつ
9570 ながれているキューである。

9571

9572 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
9573 クスの生成時に、そのミューテックスをロックする可能性のあるタスクのベース
9574 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
9575 【NGKI2009】。

9576

9577 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】。

9578

9579	TA_TPRI	0x01U	待ち行列をタスクの優先度順にする
9580	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 9581 クの優先度順にする

9582

9583 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
9584 【NGKI2011】。

9585

9586 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】。

9587

- 9588 ・ ロックしているミューテックスのリスト

9589

9590 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
9591 る【NGKI2013】。

9592

9593 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
9594 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
9595 される【NGKI2014】。

9596

9597 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
9598 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
9599 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
9600 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

9601 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
9602 【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
9603 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
9604 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
9605 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
9606 の中で最後になる【NGKI2018】.

9607
9608 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
9609 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
9610 ックしているミューテックスが残っている場合、それらのミューテックスは、ロ
9611 ックしたのと逆の順序でロック解除される【NGKI2019】.

9612
9613 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.

9614
9615 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
9616 ネルでは、静的APIによって登録されたミューテックスの
9617 数に一致）【NGKI2020】

9618
9619 【使用上の注意】

9620
9621 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
9622 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
9623 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
9624 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
9625 与しない場合に得られる利点である.

9626
9627 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
9628 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
9629 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、
9630 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
9631 コルでは、タスクがミューテックスのロック待ち状態になることはない）.

9632
9633 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
9634 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
9635 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
9636 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、
9637 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.
9638 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
9639 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
9640 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
9641 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
9642 になる.

9643
9644 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
9645 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
9646 らないploc_mtxを用いるのが安全である.

9647
9648 【補足説明】

9649
9650 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

9651 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
9652 でもpriority ceiling protocolと呼ばれているが、学術論文や他のOSでは、
9653 immediate ceiling priority protocol, priority protection protocol,
9654 priority ceiling emulation, highest locker protocolなどと呼ばれている。

9655
9656 **【TOPPERS/ASPカーネルにおける規定】**

9657
9658 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、
9659 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
9660 ることができる【ASPS0159】。

9661
9662 **【TOPPERS/FMPカーネルにおける規定】**

9663
9664 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】。

9665
9666 **【TOPPERS/HRP2カーネルにおける規定】**

9667
9668 HRP2カーネルでは、ミューテックス機能をサポートする【HRPS0131】。

9669
9670 **【未決定事項】**

9671
9672 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
9673 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
9674 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
9675 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
9676 課題である。

9677
9678 **【 μ ITRON4.0仕様との関係】**

9679
9680 μ ITRON4.0仕様の厳密な優先度制御規則を採用し、簡略化した優先度制御規則
9681 はサポートしていない。また、 μ ITRON4.0仕様でサポートしている優先度継承
9682 プロトコル (priority inheritance protocol) は、現時点ではサポートしてい
9683 ない。

9684
9685 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、
9686 μ ITRON4.0仕様では実装依存となっているが、この仕様では規定している。

9687
9688 TNUM_MTXIDは、 μ ITRON4.0仕様に規定されていないカーネル構成マクロであ
9689 る。

9690 -----
9691 CRE_MTX ミューテックスの生成 [S] 【NGKI2021】
9692 acre_mtx ミューテックスの生成 [TD] 【NGKI2022】

9693
9694 **【静的API】**

9695 CRE_MTX(ID mtxid, { ATR mtxatr, PRI ceilpri })

9696
9697 **【C言語API】**

9698 ER_ID mtxid = acre_mtx(const T_CMTX *pk_cmtx)

9699
9700 **【パラメータ】**

9701 ID mtxid 生成するミューテックスのID番号（CRE_MTXの
9702 場合）
9703 T_CMTX * pk_cmtx ミューテックスの生成情報を入れたパケット
9704 へのポインタ（静的APIを除く）
9705
9706 *ミューテックスの生成情報（パケットの内容）
9707 ATR mtxatr ミューテックス属性
9708 PRI ceilpri ミューテックスの上限優先度
9709
9710 【リターンパラメータ】
9711 ER_ID mtxid 生成されたミューテックスのID番号（正の値）
9712 またはエラーコード
9713
9714 【エラーコード】
9715 E_CTX コンテキストエラー
9716 ・非タスクコンテキストからの呼出し [s] 【NGKI2023】
9717 ・CPUロック状態からの呼出し [s] 【NGKI2024】
9718 E_RSATR 予約属性
9719 ・mtxatrが無効 【NGKI2025】
9720 ・属する保護ドメインの指定が有効範囲外 [sP] 【NGKI2026】
9721 ・属するクラスの指定が有効範囲外 [sM] 【NGKI2027】
9722 ・クラスの囲みの中に記述されていない [SM] 【NGKI2028】
9723 E_PAR パラメータエラー
9724 ・条件については機能の項を参照
9725 E_OACV オブジェクトアクセス違反
9726 ・システム状態に対する管理操作が許可されていない [sP]
9727 【NGKI2029】
9728 E_MACV メモリアクセス違反
9729 ・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
9730 いない [sP] 【NGKI2030】
9731 E_NOID ID番号不足
9732 ・割り付けられるミューテックスIDがない [sD] 【NGKI2031】
9733 E_OBJ オブジェクト状態エラー
9734 ・mtxidで指定したセマフォが登録済み（CRE_MTXの場合） 【NGKI2032】
9735
9736 【機能】
9737
9738 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
9739 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
9740 に、待ち行列は空の状態に初期化される 【NGKI2033】．
9741
9742 静的APIにおいては、mtxidはオブジェクト識別名、ceilpriは整数定数式パラメータ
9743 である 【NGKI2034】．優先度上限ミューテックス以外の場合には、ceilpriの
9744 指定を省略することができる 【NGKI2035】．
9745
9746 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、
9747 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
9748 【NGKI2037】．
9749
9750 【TOPPERS/ASPカーネルにおける規定】

196

9801

9802 **【エラーコード】**

9803 E_CTX コンテキストエラー

9804 ・非タスクコンテキストからの呼出し [s] **【NGKI2044】**

9805 ・CPUロック状態からの呼出し [s] **【NGKI2045】**

9806 E_ID 不正ID番号

9807 ・mtxidが有効範囲外 [s] **【NGKI2046】**

9808 E_RSATR 予約属性

9809 ・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

9810 述されていない [S] **【NGKI2047】**

9811 ・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

9812 ていない [SM] **【NGKI2048】**

9813 E_NOEXS オブジェクト未登録

9814 ・対象ミューテックスが未登録 **【NGKI2049】**

9815 E_OACV オブジェクトアクセス違反

9816 ・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]

9817 **【NGKI2050】**

9818 E_MACV メモリアクセス違反

9819 ・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

9820 いない [s] **【NGKI2051】**

9821 E_OBJ オブジェクト状態エラー

9822 ・対象ミューテックスは静的APIで生成された [s] **【NGKI2052】**

9823 ・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

9824 み [S] **【NGKI2053】**

9825

9826 **【機能】**

9827

9828 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

9829 （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

9830 **【NGKI2054】** .

9831

9832 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

9833 式パラメータである **【NGKI2055】** .

9834

9835 **【TOPPERS/ASPカーネルにおける規定】**

9836

9837 ASPカーネルのミューテックス機能拡張パッケージでは、SAC_MTX, sac_mtxをサ

9838 ポートしない **【ASPS0161】** .

9839

9840 **【TOPPERS/HRP2カーネルにおける規定】**

9841

9842 HRP2カーネルでは、SAC_MTXのみをサポートする **【HRPS0133】** .

9843 -----

9844 del_mtx ミューテックスの削除 [TD] **【NGKI2056】**

9845

9846 **【C言語API】**

9847 ER ercd = del_mtx(ID mtxid)

9848

9849 **【パラメータ】**

9850 ID mtxid 対象ミューテックスのID番号

9851
9852 **【リターンパラメータ】**
9853 ER ercd 正常終了 (E_OK) またはエラーコード
9854
9855 **【エラーコード】**
9856 E_CTX コンテキストエラー
9857 ・非タスクコンテキストからの呼出し【NGKI2057】
9858 ・CPUロック状態からの呼出し【NGKI2058】
9859 E_ID 不正ID番号
9860 ・mtxidが有効範囲外【NGKI2059】
9861 E_NOEXS オブジェクト未登録
9862 ・対象ミューテックスが未登録【NGKI2060】
9863 E_OACV オブジェクトアクセス違反
9864 ・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]
9865 【NGKI2061】
9866 E_OBJ オブジェクト状態エラー
9867 ・対象ミューテックスは静的APIで生成された【NGKI2062】
9868
9869 **【機能】**
9870
9871 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
9872 振舞いは以下の通り.
9873
9874 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
9875 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
9876 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
9877 テックスが削除され, 必要な場合にはそのタスクの現在優先度の変更される
9878 【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,
9879 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
9880 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
9881 【NGKI2066】.
9882
9883 **【使用上の注意】**
9884
9885 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
9886 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
9887 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
9888 クスを削除すればよい.
9889
9890 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
9891 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
9892 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
9893 み禁止時間が長くなるため, 注意が必要である.
9894
9895 **【TOPPERS/ASPカーネルにおける規定】**
9896
9897 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
9898 ない【ASPS0162】.
9899
9900 **【TOPPERS/HRP2カーネルにおける規定】**

9901
 9902 HRP2カーネルでは、del_mtxをサポートしない【HRPS0134】。
 9903 -----

9904 loc_mtx ミューテックスのロック [T] 【NGKI2067】
 9905 ploc_mtx ミューテックスのロック (ポーリング) [T] 【NGKI2068】
 9906 tloc_mtx ミューテックスのロック (タイムアウト付き) [T] 【NGKI2069】
 9907

9908 **【C言語API】**
 9909 ER ercd = loc_mtx(ID mtxid)
 9910 ER ercd = ploc_mtx(ID mtxid)
 9911 ER ercd = tloc_mtx(ID mtxid, TMO tmout)
 9912

9913 **【パラメータ】**
 9914 ID mtxid 対象ミューテックスのID番号
 9915 TMO tmout タイムアウト時間 (twai_mtxの場合)
 9916

9917 **【リターンパラメータ】**
 9918 ER ercd 正常終了 (E_OK) またはエラーコード
 9919

9920 **【エラーコード】**
 9921 E_CTX コンテキストエラー
 9922 ・非タスクコンテキストからの呼出し【NGKI2070】
 9923 ・CPUロック状態からの呼出し【NGKI2071】
 9924 ・ディスパッチ保留状態からの呼出し (ploc_mtxを除く)【NGKI2072】
 9925 E_NOSPT 未サポート機能
 9926 ・制約タスクからの呼出し (ploc_mtxを除く)【NGKI2073】
 9927 E_ID 不正ID番号
 9928 ・mtxidが有効範囲外【NGKI2074】
 9929 E_PAR パラメータエラー
 9930 ・tmoutが無効 (tloc_mtxの場合)【NGKI2075】
 9931 E_NOEXS オブジェクト未登録
 9932 ・対象ミューテックスが未登録 [D]【NGKI2076】
 9933 E_OACV オブジェクトアクセス違反
 9934 ・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]
 9935 【NGKI2077】
 9936 E_ILUSE サービスコール不正使用
 9937 ・条件については機能の項を参照
 9938 E_TMOUT ポーリング失敗またはタイムアウト (loc_mtxを除く)【NGKI2078】
 9939 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)
 9940 【NGKI2079】
 9941 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)
 9942 【NGKI2080】
 9943

9944 **【機能】**
 9945
 9946 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
 9947 な振舞いは以下の通り。
 9948
 9949 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
 9950 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

9951 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
9952 優先度に変更される【NGKI2082】。
9953
9954 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
9955 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
9956 の待ち行列につながる【NGKI2083】。
9957
9958 対象ミューテックスが自タスクによってロックされている場合には、E_ILUSEエ
9959 ラーとなる【NGKI2084】。また、対象ミューテックスが優先度上限ミューテッ
9960 クスで、その上限優先度より自タスクのベース優先度が高い場合にも、
9961 E_ILUSEエラーとなる【NGKI2085】。
9962 -----
9963 unl_mtx ミューテックスのロック解除 [T] 【NGKI2086】
9964
9965 【C言語API】
9966 ER ercd = unl_mtx(ID mtxid)
9967
9968 【パラメータ】
9969 ID mtxid 対象ミューテックスのID番号
9970
9971 【リターンパラメータ】
9972 ER ercd 正常終了 (E_OK) またはエラーコード
9973
9974 【エラーコード】
9975 E_CTX コンテキストエラー
9976 ・非タスクコンテキストからの呼出し【NGKI2087】
9977 ・CPUロック状態からの呼出し【NGKI2088】
9978 E_ID 不正ID番号
9979 ・mtxidが有効範囲外【NGKI2089】
9980 E_NOEXS オブジェクト未登録
9981 ・対象ミューテックスが未登録 [D] 【NGKI2090】
9982 E_ILUSE サービスコール不正使用
9983 ・条件については機能の項を参照
9984
9985 【機能】
9986
9987 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
9988 体的な振舞いは以下の通り。
9989
9990 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
9991 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
9992 【NGKI2091】。
9993
9994 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
9995 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
9996 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
9997 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
9998 加され、必要な場合にはそのタスクの現在優先度に変更される【NGKI2094】。
9999 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
10000 【NGKI2095】。

10001
 10002 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
 10003 いない状態になる【NGKI2096】。
 10004
 10005 対象ミューテックスが自タスクによってロックされていない場合には、
 10006 E_ILUSEエラーとなる【NGKI2097】。
 10007 -----
 10008 ini_mtx ミューテックスの再初期化 [T] 【NGKI2098】
 10009
 10010 【C言語API】
 10011 ER ercd = ini_mtx(ID mtxid)
 10012
 10013 【パラメータ】
 10014 ID mtxid 対象ミューテックスのID番号
 10015
 10016 【リターンパラメータ】
 10017 ER ercd 正常終了 (E_OK) またはエラーコード
 10018
 10019 【エラーコード】
 10020 E_CTX コンテキストエラー
 10021 ・非タスクコンテキストからの呼出し【NGKI2099】
 10022 ・CPUロック状態からの呼出し【NGKI2100】
 10023 E_ID 不正ID番号
 10024 ・mtxidが有効範囲外【NGKI2101】
 10025 E_NOEXS オブジェクト未登録
 10026 ・対象ミューテックスが未登録 [D] 【NGKI2102】
 10027 E_OACV オブジェクトアクセス違反
 10028 ・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]
 10029 【NGKI2103】
 10030
 10031 【機能】
 10032
 10033 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
 10034 的な振舞いは以下の通り。
 10035
 10036 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
 10037 【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、
 10038 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
 10039 が削除され、必要な場合にはそのタスクの現在優先度が変更される
 10040 【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、
 10041 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
 10042 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
 10043 【NGKI2107】。
 10044
 10045 【使用上の注意】
 10046
 10047 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
 10048 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
 10049 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
 10050 テックスを再初期化すればよい。

10051
 10052 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
 10053 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
 10054 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
 10055 み禁止時間が長くなるため、注意が必要である。
 10056
 10057 ミューテックスを再初期化した場合に、アプリケーションとの整合性を保つの
 10058 は、アプリケーションの責任である。
 10059
 10060 **【μITRON4.0仕様との関係】**
 10061
 10062 μITRON4.0仕様に定義されていないサービスコールである。
 10063 -----
 10064 ref_mtx ミューテックスの状態参照 [T] **【NGKI2108】**
 10065
 10066 **【C言語API】**
 10067 ER ercd = ref_mtx(ID mtxid, T_RMTX *pk_rmtx)
 10068
 10069 **【パラメータ】**
 10070 ID mtxid 対象ミューテックスのID番号
 10071 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
 10072 のポインタ
 10073
 10074 **【リターンパラメータ】**
 10075 ER ercd 正常終了 (E_OK) またはエラーコード
 10076
 10077 *ミューテックスの現在状態 (パケットの内容)
 10078 ID htsskid ミューテックスをロックしているタスクのID番号
 10079 ID wtsskid ミューテックスの待ち行列の先頭のタスクのID
 10080 番号
 10081
 10082 **【エラーコード】**
 10083 E_CTX コンテキストエラー
 10084 ・非タスクコンテキストからの呼出し **【NGKI2109】**
 10085 ・CPUロック状態からの呼出し **【NGKI2110】**
 10086 E_ID 不正ID番号
 10087 ・mtxidが有効範囲外 **【NGKI2111】**
 10088 E_NOEXS オブジェクト未登録
 10089 ・対象ミューテックスが未登録 [D] **【NGKI2112】**
 10090 E_OACV オブジェクトアクセス違反
 10091 ・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]
 10092 **【NGKI2113】**
 10093 E_MACV メモリアクセス違反
 10094 ・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
 10095 いない [P] **【NGKI2114】**
 10096
 10097 **【機能】**
 10098
 10099 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。
 10100 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。

10101
10102 対象ミューテックスがロックされていない場合, htsskidにはTSK_NONE (=0) が
10103 返る【NGKI2115】.

10104
10105 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, wtsskidには
10106 TSK_NONE (=0) が返る【NGKI2116】.

10107
10108 【使用上の注意】

10109
10110 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
10111 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
10112 直後に割り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
10113 スの状態が変化している可能性があるためである.

10114 -----

10115 10116 4.4.7 メッセージバッファ

10117
10118 ☆未完成

10119 10120 4.4.8 スピンロック

10121
10122 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, 割り込みのマスクと
10123 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
10124 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
10125 【NGKI2117】.

10126
10127 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
10128 カーネル管理の割り込みがマスクされ, ディスパッチが保留される【NGKI2118】.
10129 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
10130 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
10131 態であり, 割り込みはマスクされない【NGKI2120】. プロセッサ間ロックを取得
10132 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,
10133 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
10134 を返却するという.

10135
10136 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
10137 ルーチン, 割り込みハンドラ, 割り込みサービ斯拉ーチン, タイムイベントハンド
10138 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
10139 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
10140 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
10141 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
10142 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
10143 では, スピンロックは返却されない【NGKI2123】.

10144
10145 各スピンロックが持つ情報は次の通り【NGKI2124】.

- 10146
10147 • スピンロック属性
10148 • 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)
10149 • アクセス許可ベクタ (保護機能対応カーネルの場合)
10150 • 属する保護ドメイン (保護機能対応カーネルの場合)

10151 ・属するクラス

10152

10153 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
10154 ック属性には，TA_NULLを指定しなければならない【NGKI2126】．

10155

10156 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．

10157

10158 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
10159 ネルでは，静的APIによって登録されたスピンロックの数
10160 に一致）【NGKI2127】

10161

10162 【補足説明】

10163

10164 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
10165 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
10166 ックを取得することはできない．そのため，1つの処理単位が，複数のスピンロ
10167 ックを取得した状態になることはできない．

10168

10169 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
10170 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
10171 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
10172 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
10173 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
10174 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．

10175

10176 【TOPPERS/ASPカーネルにおける規定】

10177

10178 ASPカーネルでは，スピンロック機能をサポートしない【ASPS0163】．

10179

10180 【TOPPERS/FMPカーネルにおける規定】

10181

10182 FMPカーネルでは，スピンロック機能をサポートする【FMPS0138】．

10183

10184 【TOPPERS/HRP2カーネルにおける規定】

10185

10186 HRP2カーネルでは，スピンロック機能をサポートしない【HRPS0135】．

10187

10188 【 μ ITRON4.0仕様との関係】

10189

10190 スピンロック機能は， μ ITRON4.0仕様に定義されていない機能である．

10191

10192 CRE_SPN スピンロックの生成〔SM〕 【NGKI2128】

10193 acre_spn スピンロックの生成〔TMD〕 【NGKI2129】

10194

10195 【静的API】

10196 CRE_SPN(ID spnid, { ATR spnatr })

10197

10198 【C言語API】

10199 ER_ID spnid = acre_spn(const T_CSPN *pk_cspn)

10200

10201 **【パラメータ】**

10202 ID spnid 生成するスピンロックのID番号（CRE_SPNの場合）

10203 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

10204 ポインタ（静的APIを除く）

10205

10206 * スピンロックの生成情報（パケットの内容）

10207 ATR spnatr スピンロック属性

10208

10209 **【リターンパラメータ】**

10210 ER_ID spnid 生成されたスピンロックのID番号（正の値）ま

10211 たはエラーコード

10212

10213 **【エラーコード】**

10214 E_CTX コンテキストエラー

10215 ・ 非タスクコンテキストからの呼出し [s] **【NGKI2130】**

10216 ・ CPUロック状態からの呼出し [s] **【NGKI2131】**

10217 E_RSATR 予約属性

10218 ・ spnatrが無効 **【NGKI2132】**

10219 ・ 属する保護ドメインの指定が有効範囲外 [sP] **【NGKI2133】**

10220 ・ 属するクラスの指定が有効範囲外 [s] **【NGKI2134】**

10221 ・ クラスの囲みの中に記述されていない [S] **【NGKI2135】**

10222 E_OACV オブジェクトアクセス違反

10223 ・ システム状態に対する管理操作が許可されていない [sP]

10224 **【NGKI2136】**

10225 E_MACV メモリアクセス違反

10226 ・ 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

10227 いない [sP] **【NGKI2137】**

10228 E_NOID ID番号不足

10229 ・ 割り付けられるスピンロックIDがない [sD] **【NGKI2138】**

10230 E_NORES 資源不足

10231 ・ 条件については機能の項を参照

10232 E_OBJ オブジェクト状態エラー

10233 ・ spnidで指定したスピンロックが登録済み（CRE_SPNの場合）

10234 **【NGKI2139】**

10235

10236 **【機能】**

10237

10238 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

10239 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

10240 化される **【NGKI2140】**．

10241

10242 静的APIにおいては、spnidはオブジェクト識別名である **【NGKI2141】**．

10243

10244 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

10245 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

10246 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

10247 **【NGKI2143】**．

10248

10249 **【補足説明】**

10250

10301

10302 **【エラーコード】**

10303 E_CTX コンテキストエラー

10304 ・非タスクコンテキストからの呼出し [s] **【NGKI2150】**

10305 ・CPUロック状態からの呼出し [s] **【NGKI2151】**

10306 E_ID 不正ID番号

10307 ・spnidが有効範囲外 [s] **【NGKI2152】**

10308 E_RSATR 予約属性

10309 ・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

10310 されていない [S] **【NGKI2153】**

10311 ・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

10312 いない [S] **【NGKI2154】**

10313 E_NOEXS オブジェクト未登録

10314 ・対象スピンロックが未登録 **【NGKI2155】**

10315 E_OACV オブジェクトアクセス違反

10316 ・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]

10317 **【NGKI2156】**

10318 E_MACV メモリアクセス違反

10319 ・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

10320 いない [s] **【NGKI2157】**

10321 E_OBJ オブジェクト状態エラー

10322 ・対象スピンロックは静的APIで生成された [s] **【NGKI2158】**

10323 ・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]

10324 **【NGKI2159】**

10325

10326 **【機能】**

10327

10328 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

10329 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

10330 **【NGKI2160】** .

10331

10332 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

10333 式パラメータである **【NGKI2161】** .

10334

10335 **【TOPPERS/FMPカーネルにおける規定】**

10336

10337 FMPカーネルでは、SAC_SPN、sac_spnをサポートしない **【FMPS0140】** .

10338 -----

10339 del_spn スピンロックの削除 [TMD] **【NGKI2162】**

10340

10341 **【C言語API】**

10342 ER ercd = del_spn(ID snpid)

10343

10344 **【パラメータ】**

10345 ID spnid 対象スピンロックのID番号

10346

10347 **【リターンパラメータ】**

10348 ER ercd 正常終了 (E_OK) またはエラーコード

10349

10350 **【エラーコード】**

10351	E_CTX	コンテキストエラー
10352		・非タスクコンテキストからの呼出し【NGKI2163】
10353		・CPUロック状態からの呼出し【NGKI2164】
10354	E_ID	不正ID番号
10355		・spnidが有効範囲外【NGKI2165】
10356	E_NOEXS	オブジェクト未登録
10357		・対象スピンロックが未登録【NGKI2166】
10358	E_OACV	オブジェクトアクセス違反
10359		・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]
10360		【NGKI2167】
10361	E_OBJ	オブジェクト状態エラー
10362		・対象スピンロックは静的APIで生成された【NGKI2168】
10363		
10364	【機能】	
10365		
10366	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	
10367	いは以下の通り．	
10368		
10369	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	
10370	される【NGKI2169】．	
10371		
10372	【TOPPERS/FMPカーネルにおける規定】	
10373		
10374	FMPカーネルでは，del_spnをサポートしない【FMPS0141】．	
10375		
10376	【未決定事項】	
10377		
10378	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．	
10379	-----	
10380	loc_spn	スピンロックの取得 [TM] 【NGKI2170】
10381	iloc_spn	スピンロックの取得 [IM] 【NGKI2171】
10382		
10383	【C言語API】	
10384	ER ercd = loc_spn(ID snpid)	
10385	ER ercd = iloc_spn(ID snpid)	
10386		
10387	【パラメータ】	
10388	ID	spnid 対象スピンロックのID番号
10389		
10390	【リターンパラメータ】	
10391	ER	ercd 正常終了 (E_OK) またはエラーコード
10392		
10393	【エラーコード】	
10394	E_CTX	コンテキストエラー
10395		・非タスクコンテキストからの呼出し (loc_spnの場合) 【NGKI2172】
10396		・タスクコンテキストからの呼出し (iloc_spnの場合) 【NGKI2173】
10397		・CPUロック状態からの呼出し【NGKI2174】
10398	E_ID	不正ID番号
10399		・spnidが有効範囲外【NGKI2175】
10400	E_NOEXS	オブジェクト未登録

209

10451 E_CTX コンテキストエラー
 10452 ・非タスクコンテキストからの呼出し (try_spnの場合) 【NGKI2186】
 10453 ・タスクコンテキストからの呼出し (itry_spnの場合) 【NGKI2187】
 10454 ・CPUロック状態からの呼出し 【NGKI2188】
 10455 E_ID 不正ID番号
 10456 ・spnidが有効範囲外 【NGKI2189】
 10457 E_NOEXS オブジェクト未登録
 10458 ・対象スピンロックが未登録 [D] 【NGKI2190】
 10459 E_OACV オブジェクトアクセス違反
 10460 ・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
 10461 (try_spnの場合) [P] 【NGKI2191】
 10462 E_OBJ オブジェクト状態エラー
 10463 ・条件については機能の項を参照
 10464

10465 【機能】

10466
 10467 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
 10468 振舞いは以下の通り.

10469
 10470 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
 10471 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
 10472 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
 10473 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】.

10474
 10475 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,
 10476 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)
 10477 には, E_OBJエラーとする 【NGKI2195】.

10478 【使用上の注意】

10479
 10480 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
 10481 りスピンロックを取得する方法は, loc_spn/iloc_spnによるスピンロックの取
 10482 得と, 同じ振舞いになるとは限らない.
 10483

10484 -----
 10485 unl_spn スピンロックの返却 [TM] 【NGKI2196】
 10486 iunl_spn スピンロックの返却 [IM] 【NGKI2197】

10487 【C言語API】

10488
 10489 ER ercd = unl_spn(ID spnid)
 10490 ER ercd = iunl_spn(ID spnid)
 10491

10492 【パラメータ】

10493 ID spnid 対象スピンロックのID番号
 10494

10495 【リターンパラメータ】

10496 ER ercd 正常終了 (E_OK) またはエラーコード
 10497

10498 【エラーコード】

10499 E_CTX コンテキストエラー
 10500 ・非タスクコンテキストからの呼出し (unl_spnの場合) 【NGKI2198】

10501 ・タスクコンテキストからの呼出し (iunl_spnの場合) 【NGKI2199】
 10502 E_ID 不正ID番号
 10503 ・spnidが有効範囲外 【NGKI2200】
 10504 E_NOEXS オブジェクト未登録
 10505 ・対象スピンロックが未登録 [D] 【NGKI2201】
 10506 E_OACV オブジェクトアクセス違反
 10507 ・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
 10508 (unl_spnの場合) [P] 【NGKI2202】
 10509 E_ILUSE サービスコール不正使用
 10510 ・条件については機能の項を参照

10511

10512 **【機能】**

10513

10514 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
 10515 いは以下の通り.

10516

10517 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
 10518 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
 10519 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
 10520 ク解除状態へ遷移する 【NGKI2204】.

10521

10522 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
 10523 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】.

10524

10525 ref_spn スピンロックの状態参照 [TM] 【NGKI2206】

10526

10527 **【C言語API】**

10528 ER ercd = ref_spn(ID spnid, T_RSPN *pk_rspn)

10529

10530 **【パラメータ】**

10531 ID spnid 対象スピンロックのID番号
 10532 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
 10533 ポインタ

10534

10535 **【リターンパラメータ】**

10536 ER ercd 正常終了 (E_OK) またはエラーコード

10537

10538 * スピンロックの現在状態 (パケットの内容)

10539 STAT spnstat ロック状態

10540

10541 **【エラーコード】**

10542 E_CTX コンテキストエラー
 10543 ・非タスクコンテキストからの呼出し 【NGKI2207】
 10544 ・CPUロック状態からの呼出し 【NGKI2208】
 10545 E_ID 不正ID番号
 10546 ・spnidが有効範囲外 【NGKI2209】
 10547 E_NOEXS オブジェクト未登録
 10548 ・対象スピンロックが未登録 [D] 【NGKI2210】
 10549 E_OACV オブジェクトアクセス違反
 10550 ・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]

10551 【NGKI2211】
10552 E_MACV メモリアクセス違反
10553 ・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
10554 いない) [P] 【NGKI2212】
10555
10556 【機能】
10557
10558 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
10559 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】．
10560
10561 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
10562 返される【NGKI2214】．
10563
10564 TSPN_UNL 0x01U 取得されていない状態
10565 TSPN_LOC 0x02U 取得されている状態
10566
10567 【使用上の注意】
10568
10569 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
10570 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
10571 後に割込みが発生した場合、ref_spnから戻ってきた時には対象スピンロックの
10572 状態が変化している可能性があるためである．
10573 -----
10574
10575 4.5 メモリプール管理機能
10576
10577 【TOPPERS/SSPカーネルにおける規定】
10578
10579 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】．
10580
10581 【μ ITRON4.0仕様との関係】
10582
10583 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．
10584
10585 【仕様決定の理由】
10586
10587 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
10588 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
10589 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
10590 ラリとして実現する方が適切と考えたためである．
10591
10592 4.5.1 固定長メモリプール
10593
10594 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
10595 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．
10596 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
10597 【NGKI2215】．
10598
10599 各固定長メモリプールが持つ情報は次の通り【NGKI2216】．
10600

10601 • 固定長メモリプール属性
 10602 • 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）
 10603 • 固定長メモリプール領域
 10604 • 固定長メモリプール管理領域
 10605 • アクセス許可ベクタ（保護機能対応カーネルの場合）
 10606 • 属する保護ドメイン（保護機能対応カーネルの場合）
 10607 • 属するクラス（マルチプロセッサ対応カーネルの場合）
 10608
 10609 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
 10610 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
 10611 きる順序でつながれているキューである。
 10612
 10613 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
 10614 めのメモリ領域である。
 10615
 10616 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
 10617 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
 10618 のメモリ領域である。
 10619
 10620 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
 10621 用いるオブジェクト管理領域として扱われる【NGKI2217】。
 10622
 10623 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】。
 10624
 10625 TA_TPRI 0x01U 待ち行列をタスクの優先度順にする
 10626
 10627 TA_TPRIを指定しない場合、待ち行列はFIFO順になる【NGKI2219】。
 10628
 10629 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。
 10630
 10631 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
 10632 カーネルでは、静的APIによって登録された固定長メモ
 10633 プールの数に一致）【NGKI2220】
 10634
 10635 【 μ ITRON4.0仕様との関係】
 10636
 10637 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
 10638 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
 10639 で確保する方法を定めた結果、そのサイズは($\text{blkcnt} * \text{ROUND_MPF_T}(\text{blksz})$)
 10640 で求めることができるようになったためである。
 10641
 10642 TNUM_MPFIDは、 μ ITRON4.0仕様に規定されていないカーネル構成マクロである。
 10643 -----
 10644 CRE_MPF 固定長メモリプールの生成 [S] 【NGKI2221】
 10645 acre_mpf 固定長メモリプールの生成 [TD] 【NGKI2222】
 10646
 10647 【静的API】
 10648 CRE_MPF(ID mpfid, { ATR mpfatr, uint_t blkcnt, uint_t blksz,
 10649 MPF_T *mpf, void *mpfmb })
 10650

10651 **【C言語API】**
 10652 ER_ID mpfid = acre_mpf(const T_CMPF *pk_cmpf)
 10653
 10654 **【パラメータ】**
 10655 ID mpfid 生成する固定長メモリプールのID番号（CRE_MPF
 10656 の場合）
 10657 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
 10658 トへのポインタ（静的APIを除く）
 10659
 10660 *固定長メモリプールの生成情報（パケットの内容）
 10661 ATR mpfattr 固定長メモリプール属性
 10662 uint_t blkcnt 獲得できる固定長メモリブロックの数
 10663 uint_t blksize 固定長メモリブロックのサイズ（バイト数）
 10664 MPF_T * mpf 固定長メモリプール領域の先頭番地
 10665 void * mpfmb 固定長メモリプール管理領域の先頭番地
 10666
 10667 **【リターンパラメータ】**
 10668 ER_ID mpfid 生成された固定長メモリプールのID番号（正の
 10669 値）またはエラーコード
 10670
 10671 **【エラーコード】**
 10672 E_CTX コンテキストエラー
 10673 ・非タスクコンテキストからの呼出し [s] 【NGKI2223】
 10674 ・CPUロック状態からの呼出し [s] 【NGKI2224】
 10675 E_RSATR 予約属性
 10676 ・mpfattrが無効 【NGKI2225】
 10677 ・属する保護ドメインの指定が有効範囲外 [sP] 【NGKI2226】
 10678 ・属するクラスの指定が有効範囲外 [sM] 【NGKI2227】
 10679 ・クラスの囲みの中に記述されていない [SM] 【NGKI2228】
 10680 E_NOSPT 未サポート機能
 10681 ・条件については各カーネルにおける規定の項を参照
 10682 E_PAR パラメータエラー
 10683 ・blkcntが0 【NGKI2229】
 10684 ・blksizeが0 【NGKI2230】
 10685 ・その他の条件については機能の項を参照
 10686 E_OACV オブジェクトアクセス違反
 10687 ・システム状態に対する管理操作が許可されていない [sP]
 10688 【NGKI2231】
 10689 E_MACV メモリアクセス違反
 10690 ・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
 10691 いない [sP] 【NGKI2232】
 10692 E_NOID ID番号不足
 10693 ・割り付けられる固定長メモリプールIDがない [sD] 【NGKI2233】
 10694 E_NOMEM メモリ不足
 10695 ・固定長メモリプール領域が確保できない 【NGKI2234】
 10696 ・固定長メモリプール管理領域が確保できない 【NGKI2235】
 10697 E_OBJ オブジェクト状態エラー
 10698 ・mpfidで指定したデータキューが登録済み（CRE_MPFの場合）
 10699 【NGKI2236】
 10700 ・その他の条件については機能の項を参照

10701
 10702 **【機能】**
 10703
 10704 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
 10705 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、
 10706 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
 10707 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
 10708 空の状態に初期化される【NGKI2238】。
 10709
 10710 静的APIにおいては、mpfidはオブジェクト識別名、blkcntとblkkszは整数定数式
 10711 パラメータ、mpfとmpfmbは一般定数式パラメータである【NGKI2239】。コンフィ
 10712 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
 10713 ない【NGKI2240】。
 10714
 10715 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
 10716 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】。
 10717
 10718 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
 10719 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
 10720 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
 10721 ト中に確保される【NGKI2242】。
 10722
 10723 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
 10724 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】。
 10725
 10726 〔mpfにNULL以外を指定した場合〕
 10727
 10728 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
 10729 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
 10730 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
 10731 意している【NGKI2245】。
 10732
 10733 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
 10734
 10735 COUNT_MPF_T(blkksz) 固定長メモリブロックのサイズがblkkszの固定長メモ
 10736 リプール領域を確保するために、固定長メモリブロッ
 10737 ク1つあたりに必要なMPF_T型の配列の要素数
 10738 ROUND_MPF_T(blkksz) 要素数COUNT_MPF_T(blkksz)のMPF_T型の配列のサイズ
 10739 (blkkszを、MPF_T型のサイズの倍数になるように大き
 10740 い方に丸めた値)
 10741
 10742 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】。
 10743
 10744 MPF_T <固定長メモリプール領域の変数名>[(blkcnt) * COUNT_MPF_T(blkksz)];
 10745
 10746 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】。
 10747
 10748 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
 10749 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
 10750 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

10751 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
10752 【NGKI2249】。

10753

10754 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
10755 領域は、カーネルに登録されたメモリオブジェクトに含まれていなければならない。指定した固定長メモリプール領域が、カーネルに登録されたメモリオブ
10756 ジェクトに含まれていない場合、E_OBJエラーとなる【NGKI2251】。

10757

10758

10759 [mpfmbにNULL以外を指定した場合]

10760

10761 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
10762 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
10763 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
10764 用意している【NGKI2253】。

10765

10766	TSZ_MPFMB(blkent)	blkentで指定した数の固定長メモリブロックを管理
10767		することができる固定長メモリプール管理領域のサ
10768		イズ (バイト数)
10769	TCNT_MPFMB(blkent)	blkentで指定した数の固定長メモリブロックを管理
10770		することができる固定長メモリプール管理領域を確
10771		保するために必要なMB_T型の配列の要素数
10772		

10773 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
10774 【NGKI2254】。

10775

10776 MB_T <固定長メモリプール管理領域の変数名>[TCNT_MPFMB(blkent)];

10777

10778 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
10779 【NGKI2255】。

10780

10781 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
10782 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
10783 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
10784 オブジェクトに含まれない場合、E_OBJエラーとなる【NGKI2257】。

10785

10786 【補足説明】

10787

10788 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
10789 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
10790 ス権の設定は変更されない。これらを適切に設定することは、アプリケーション
10791 ンの責任である。

10792

10793 【TOPPERS/ASPカーネルにおける規定】

10794

10795 ASPカーネルでは、CRE_MPFのみをサポートする【ASPS0164】。また、mpfmbには
10796 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
10797 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
10798 サポートする【ASPS0167】。acre_mpfに対しては、mpfmbにNULL以外を指定でき
10799 ないという制限はない【ASPS0168】。

10800

10801 【TOPPERS/FMPカーネルにおける規定】
10802
10803 FMPカーネルでは、CRE_MPFのみをサポートする【FMPS0142】。また、mpfmbには
10804 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
10805 となる【FMPS0144】。
10806
10807 【TOPPERS/HRP2カーネルにおける規定】
10808
10809 HRP2カーネルでは、CRE_MPFのみをサポートする【HRPS0136】。また、mpfmbに
10810 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
10811 ラーとなる【HRPS0138】。
10812
10813 【 μ ITRON4.0仕様との関係】
10814
10815 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
10816 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、
10817 μ ITRON4.0/PX仕様にあわせて、固定長メモリプール生成情報に、mpfmbを追加
10818 した。
10819
10820 【 μ ITRON4.0/PX仕様との関係】
10821
10822 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
10823 する方法を規定した。
10824
10825 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】
10826
10827 【静的API】
10828 AID_MPF(uint_t nompf)
10829
10830 【パラメータ】
10831 uint_t nompf 割付け可能な固定長メモリプールIDの数
10832
10833 【エラーコード】
10834 E_RSATR 予約属性
10835 ・クラスの囲みの中に記述されていない [M] 【NGKI2259】
10836
10837 【機能】
10838
10839 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
10840 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
10841 【NGKI2260】。
10842
10843 nompfは整数定数式パラメータである【NGKI2261】。
10844
10845 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】
10846 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】
10847
10848 【静的API】
10849 SAC_MPF(ID mpfid, { ACPTN acptn1, ACPTN acptn2,
10850 ACPTN acptn3, ACPTN acptn4 })

```

10851
10852 【C言語API】
10853     ER ercd = sac_mpf(ID mpfid, const ACVCT *p_acvct)
10854
10855 【パラメータ】
10856     ID          mpfid      対象固定長メモリプールのID番号
10857     ACVCT *     p_acvct    アクセス許可ベクタを入れたパケットへのポ
10858                          インタ（静的APIを除く）
10859
10860     *アクセス許可ベクタ（パケットの内容）
10861     ACPTN       acptn1     通常操作1のアクセス許可パターン
10862     ACPTN       acptn2     通常操作2のアクセス許可パターン
10863     ACPTN       acptn3     管理操作のアクセス許可パターン
10864     ACPTN       acptn4     参照操作のアクセス許可パターン
10865
10866 【リターンパラメータ】
10867     ER          ercd       正常終了（E_OK）またはエラーコード
10868
10869 【エラーコード】
10870     E_CTX       コンテキストエラー
10871                ・非タスクコンテキストからの呼出し [s] 【NGKI2264】
10872                ・CPUロック状態からの呼出し [s] 【NGKI2265】
10873     E_ID        不正ID番号
10874                ・mpfidが有効範囲外 [s] 【NGKI2266】
10875     E_RSATR     予約属性
10876                ・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
10877                  に記述されていない [S] 【NGKI2267】
10878                ・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
10879                  されていない [SM] 【NGKI2268】
10880     E_NOEXS     オブジェクト未登録
10881                ・対象固定長メモリプールが未登録 【NGKI2269】
10882     E_OACV      オブジェクトアクセス違反
10883                ・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
10884                  ない [s] 【NGKI2270】
10885     E_MACV      メモリアクセス違反
10886                ・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
10887                  いない [s] 【NGKI2271】
10888     E_OBJ       オブジェクト状態エラー
10889                ・対象固定長メモリプールは静的APIで生成された [s] 【NGKI2272】
10890                ・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
10891                  定済み [S] 【NGKI2273】
10892
10893 【機能】
10894
10895 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
10896 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
10897 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
10898 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
10899 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
10900 設定する 【NGKI2275】．

```

10901
10902 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
10903 式パラメータである【NGKI2276】。
10904
10905 【TOPPERS/ASPカーネルにおける規定】
10906
10907 ASPカーネルでは、SAC_MPF、sac_mpfをサポートしない【ASPS0169】。
10908
10909 【TOPPERS/FMPカーネルにおける規定】
10910
10911 FMPカーネルでは、SAC_MPF、sac_mpfをサポートしない【FMPS0145】。
10912
10913 【TOPPERS/HRP2カーネルにおける規定】
10914
10915 HRP2カーネルでは、SAC_MPFのみをサポートする【HRPS0139】。
10916 -----
10917 del_mpf 固定長メモリプールの削除〔TD〕【NGKI2277】
10918
10919 【C言語API】
10920 ER ercd = del_mpf(ID mpfid)
10921
10922 【パラメータ】
10923 ID mpfid 対象固定長メモリプールのID番号
10924
10925 【リターンパラメータ】
10926 ER ercd 正常終了 (E_OK) またはエラーコード
10927
10928 【エラーコード】
10929 E_CTX コンテキストエラー
10930 ・非タスクコンテキストからの呼出し【NGKI2278】
10931 ・CPUロック状態からの呼出し【NGKI2279】
10932 E_ID 不正ID番号
10933 ・mpfidが有効範囲外【NGKI2280】
10934 E_NOEXS オブジェクト未登録
10935 ・対象固定長メモリプールが未登録【NGKI2281】
10936 E_OACV オブジェクトアクセス違反
10937 ・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
10938 ない〔P〕【NGKI2282】
10939 E_OBJ オブジェクト状態エラー
10940 ・対象固定長メモリプールは静的APIで生成された【NGKI2283】
10941
10942 【機能】
10943
10944 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。
10945 具体的な振舞いは以下の通り。
10946
10947 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
10948 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
10949 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
10950 【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

10951 からE_DLTエラーが返る【NGKI2286】。

10952

10953 【使用上の注意】

10954

10955 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
10956 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
10957 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
10958 み禁止時間が長くなるため、注意が必要である。

10959

10960 【TOPPERS/ASPカーネルにおける規定】

10961

10962 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
10963 機能拡張パッケージでは、del_mpfをサポートする【ASPS0171】。

10964

10965 【TOPPERS/FMPカーネルにおける規定】

10966

10967 FMPカーネルでは、del_mpfをサポートしない【FMPS0146】。

10968

10969 【TOPPERS/HRP2カーネルにおける規定】

10970

10971 HRP2カーネルでは、del_mpfをサポートしない【HRPS0140】。

10972

10973 get_mpf 固定長メモリブロックの獲得 [T] 【NGKI2287】

10974 pget_mpf 固定長メモリブロックの獲得（ポーリング） [T] 【NGKI2288】

10975 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】

10976

10977 【C言語API】

10978 ER ercd = get_mpf(ID mpfid, void **p_blk)

10979 ER ercd = pget_mpf(ID mpfid, void **p_blk)

10980 ER ercd = tget_mpf(ID mpfid, void **p_blk, TMO tmout)

10981

10982 【パラメータ】

10983 ID mpfid 対象固定長メモリプールのID番号

10984 void ** p_blk 獲得した固定長メモリブロックの先頭番地を入
10985 れるメモリ領域へのポインタ

10986 TMO tmout タイムアウト時間（twai_mpfの場合）

10987

10988 【リターンパラメータ】

10989 ER ercd 正常終了（E_OK）またはエラーコード

10990 void * blk 獲得した固定長メモリブロックの先頭番地

10991

10992 【エラーコード】

10993 E_CTX コンテキストエラー

10994 ・非タスクコンテキストからの呼出し【NGKI2290】

10995 ・CPUロック状態からの呼出し【NGKI2291】

10996 ・ディスパッチ保留状態からの呼出し（pget_mpfを除く）

10997 【NGKI2292】

10998 E_NOSPT 未サポート機能

10999 ・制約タスクからの呼出し（pget_mpfを除く）【NGKI2293】

11000 E_ID 不正ID番号

11001 ・mpfidが有効範囲外【NGKI2294】
 11002 E_PAR パラメータエラー
 11003 ・tmoutが無効 (tget_mpfの場合) 【NGKI2295】
 11004 E_NOEXS オブジェクト未登録
 11005 ・対象固定長メモリプールが未登録 [D] 【NGKI2296】
 11006 E_OACV オブジェクトアクセス違反
 11007 ・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
 11008 ない [P] 【NGKI2297】
 11009 E_MACV メモリアクセス違反
 11010 ・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
 11011 ない) [P] 【NGKI2298】
 11012 E_TMOUT ポーリング失敗またはタイムアウト (get_mpfを除く) 【NGKI2299】
 11013 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)
 11014 【NGKI2300】
 11015 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)
 11016 【NGKI2301】
 11017

【機能】

11018
 11019
 11020 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
 11021 モリブロックを獲得し、その先頭番地をblkに返す。具体的な振舞いは以下の通
 11022 り。
 11023
 11024 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
 11025 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
 11026 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】。
 11027
 11028 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
 11029 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】。
 11030

11031 rel_mpf 固定長メモリブロックの返却 [T] 【NGKI2304】
 11032

【C言語API】

11033 ER ercd = rel_mpf(ID mpfid, void *blk)
 11034
 11035

【パラメータ】

11036 ID mpfid 対象固定長メモリプールのID番号
 11037 void * blk 返却する固定長メモリブロックの先頭番地
 11038
 11039

【リターンパラメータ】

11040 ER ercd 正常終了 (E_OK) またはエラーコード
 11041
 11042

【エラーコード】

11043 E_CTX コンテキストエラー
 11044 ・非タスクコンテキストからの呼出し【NGKI2305】
 11045 ・CPUロック状態からの呼出し【NGKI2306】
 11046 E_ID 不正ID番号
 11047 ・mpfidが有効範囲外【NGKI2307】
 11048 E_PAR パラメータエラー
 11049 ・条件については機能の項を参照
 11050

11051 E_NOEXS オブジェクト未登録
 11052 ・対象固定長メモリプールが未登録 [D] 【NGKI2308】
 11053 E_OACV オブジェクトアクセス違反
 11054 ・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
 11055 ない [P] 【NGKI2309】
 11056
 11057 **【機能】**
 11058
 11059 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
 11060 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．
 11061
 11062 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
 11063 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
 11064 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
 11065 ルからE_OKが返る【NGKI2311】．
 11066
 11067 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
 11068 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】．
 11069
 11070 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
 11071 でない場合には，E_PARエラーとなる【NGKI2313】．
 11072 -----
 11073 ini_mpf 固定長メモリプールの再初期化 [T] 【NGKI2314】
 11074
 11075 **【C言語API】**
 11076 ER ercd = ini_mpf(ID mpfid)
 11077
 11078 **【パラメータ】**
 11079 ID mpfid 対象固定長メモリプールのID番号
 11080
 11081 **【リターンパラメータ】**
 11082 ER ercd 正常終了 (E_OK) またはエラーコード
 11083
 11084 **【エラーコード】**
 11085 E_CTX コンテキストエラー
 11086 ・非タスクコンテキストからの呼出し【NGKI2315】
 11087 ・CPUロック状態からの呼出し【NGKI2316】
 11088 E_ID 不正ID番号
 11089 ・mpfidが有効範囲外【NGKI2317】
 11090 E_NOEXS オブジェクト未登録
 11091 ・対象固定長メモリプールが未登録 [D] 【NGKI2318】
 11092 E_OACV オブジェクトアクセス違反
 11093 ・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
 11094 ない [P] 【NGKI2319】
 11095
 11096 **【機能】**
 11097
 11098 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
 11099 る．具体的な振舞いは以下の通り．
 11100

11101 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
 11102 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
 11103 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
 11104 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
 11105 る【NGKI2322】。

11106

11107 【使用上の注意】

11108

11109 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
 11110 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
 11111 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
 11112 み禁止時間が長くなるため、注意が必要である。

11113

11114 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
 11115 つのは、アプリケーションの責任である。

11116

11117 【μ ITRON4.0仕様との関係】

11118

11119 μ ITRON4.0仕様に定義されていないサービスコールである。

11120

11121 ref_mpf 固定長メモリプールの状態参照 [T] 【NGKI2323】

11122

11123 【C言語API】

11124 ER ercd = ref_mpf(ID mpfid, T_RMPF *pk_rmpf)

11125

11126 【パラメータ】

11127	ID	mpfid	対象固定長メモリプールのID番号
11128	T_RMPF *	pk_rmpf	固定長メモリプールの現在状態を入れるパケッ トへのポインタ

11129

11130 【リターンパラメータ】

11131	ER	ercd	正常終了 (E_OK) またはエラーコード
-------	----	------	-----------------------

11132

11133 * 固定長メモリプールの現在状態 (パケットの内容)

11134	ID	wtskid	固定長メモリプールの待ち行列の先頭のタスク のID番号
11135	uint_t	fblkcnt	固定長メモリプール領域の空きメモリ領域に割 り付けることができる固定長メモリブロックの 数

11136

11137 【エラーコード】

11141	E_CTX	コンテキストエラー
11142		・非タスクコンテキストからの呼出し【NGKI2324】
11143		・CPUロック状態からの呼出し【NGKI2325】
11144	E_ID	不正ID番号
11145		・mpfidが有効範囲外【NGKI2326】
11146	E_NOEXS	オブジェクト未登録
11147		・対象固定長メモリプールが未登録 [D] 【NGKI2327】
11148	E_OACV	オブジェクトアクセス違反
11149		・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
11150		

11151 ない [P] 【NGKI2328】
11152 E_MACV メモリアクセス違反
11153 ・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
11154 いない) [P] 【NGKI2329】
11155
11156 【機能】
11157
11158 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
11159 参照する．参照した現在状態は、pk_rmpfで指定したパッケージに返される
11160 【NGKI2330】．
11161
11162 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、wtsskidには
11163 TSK_NONE (=0) が返る 【NGKI2331】．
11164
11165 【使用上の注意】
11166
11167 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
11168 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
11169 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
11170 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．
11171 -----
11172
11173 4.6 時間管理機能
11174
11175 4.6.1 システム時刻管理
11176
11177 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
11178 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
11179 るカーネルオブジェクトである．システム時刻は、符号無し of 整数型である
11180 SYSTIM型で表され、単位はミリ秒である 【NGKI2332】．
11181
11182 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
11183 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
11184 され、SYSTIM 型で表せる最大値 (ULONG_MAX) を超えると0に戻される
11185 【NGKI2334】．タイムティックの周期は、ターゲット定義である 【NGKI2335】．
11186 また、システム時刻の精度はターゲットに依存する 【NGKI2336】．
11187
11188 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
11189 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
11190 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
11191 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
11192 【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、
11193 「2.3.4 マルチプロセッサ対応」の節を参照すること．
11194
11195 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、
11196 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
11197 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
11198 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
11199 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
11200 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

11201 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
11202 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】。
11203
11204 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
11205 た場合、それらの処理の間の処理順序は規定されない【NGKI2343】。
11206
11207 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
11208 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
11209 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、
11210 実際の精度はターゲットに依存する【NGKI2345】。
11211
11212 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
11213 ゲット定義とする【NGKI2346】。
11214
11215 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。
11216
11217 TIC_NUME タイムティックの周期（単位はミリ秒）の分子 【NGKI2347】
11218 TIC_DEN0 タイムティックの周期（単位はミリ秒）の分母
11219
11220 TOPPERS_SUPPORT_GET_UTM get_utmがサポートされている【NGKI2348】
11221
11222 【TOPPERS/SSPカーネルにおける規定】
11223
11224 SSPカーネルでは、時間管理機能をサポートしない【SSPS0129】。
11225
11226 【使用上の注意】
11227
11228 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
11229 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
11230 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
11231 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
11232 性があるため、注意が必要である。
11233
11234 【 μ ITRON4.0仕様との関係】
11235
11236 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
11237 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
11238 ビスコール（isig_tim）は廃止した。
11239
11240 【 μ ITRON4.0/PX仕様との関係】
11241
11242 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
11243 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
11244 設定する静的API（SAC_TIM）は廃止した。
11245 -----
11246 get_tim システム時刻の参照 [T] 【NGKI2349】
11247
11248 【C言語API】
11249 ER ercd = get_tim(SYSTIM *p_systim)
11250

11251 **【パラメータ】**
 11252 SYSTIM * p_systim システム時刻を入れるメモリ領域へのポインタ
 11253
 11254 **【リターンパラメータ】**
 11255 ER ercd 正常終了 (E_OK) またはエラーコード
 11256 SYSTIM systim システム時刻の現在値
 11257
 11258 **【エラーコード】**
 11259 E_CTX コンテキストエラー
 11260 ・非タスクコンテキストからの呼出し【NGKI2350】
 11261 ・CPUロック状態からの呼出し【NGKI2351】
 11262 E_OACV オブジェクトアクセス違反
 11263 ・システム状態に対する参照操作が許可されていない [P]
 11264 【NGKI2352】
 11265 E_MACV メモリアクセス違反
 11266 ・p_systimが指すメモリ領域への書込みアクセスが許可されて
 11267 いない) [P] 【NGKI2353】
 11268
 11269 **【機能】**
 11270
 11271 システム時刻の現在値を参照する。参照したシステム時刻は、p_systimで指定
 11272 したメモリ領域に返される【NGKI2354】。
 11273
 11274 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、
 11275 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
 11276 【NGKI2355】。
 11277
 11278 **【補足説明】**
 11279
 11280 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
 11281 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。
 11282 -----
 11283 get_utm 性能評価用システム時刻の参照 [TI] 【NGKI2356】
 11284
 11285 **【C言語API】**
 11286 ER ercd = get_utm(SYSUTM *p_sysutm)
 11287
 11288 **【パラメータ】**
 11289 SYSUTM * p_sysutm 性能評価用システム時刻を入れるメモリ領域へ
 11290 のポインタ
 11291
 11292 **【リターンパラメータ】**
 11293 ER ercd 正常終了 (E_OK) またはエラーコード
 11294 SYSUTM sysutm 性能評価用システム時刻の現在値
 11295
 11296 **【エラーコード】**
 11297 E_NOSPT 未サポート機能
 11298 ・条件については機能の項を参照
 11299 E_MACV メモリアクセス違反
 11300 ・p_sysutmが指すメモリ領域へ書込みアクセスが許可されて

11301 いない) [P] 【NGKI2357】

11302

11303 【機能】

11304

11305 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
11306 刻は、p_sysutmで指定したメモリ領域に返される【NGKI2358】。

11307

11308 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
11309 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
11310 状態であっても呼び出すことができる。

11311

11312 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】。
11313 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
11314 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、
11315 E_NOSPTエラーが返るか、リンク時にエラーとなる【NGKI2362】。

11316

11317 【使用方法】

11318

11319 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。
11320 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
11321 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
11322 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。

11323

11324 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
11325 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
11326 限らない。

11327

11328 【使用上の注意】

11329

11330 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
11331 ない。

11332

11333 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
11334 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
11335 込みの応答性が低下する。

11336

11337 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
11338 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
11339 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
11340 考慮に入れて使用しなければならない。

11341

11342 【μ ITRON4.0仕様との関係】

11343

11344 μ ITRON4.0仕様に定義されていないサービスコールである。

11345 -----

11346

11347 4.6.2 周期ハンドラ

11348

11349 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。
11350 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】。

11351
11352 各周期ハンドラが持つ情報は次の通り【NGKI2364】。
11353
11354 ・周期ハンドラ属性
11355 ・周期ハンドラの動作状態
11356 ・次に周期ハンドラを起動する時刻
11357 ・拡張情報
11358 ・周期ハンドラの手頭番地
11359 ・起動周期
11360 ・起動位相
11361 ・アクセス許可ベクタ（保護機能対応カーネルの場合）
11362 ・属する保護ドメイン（保護機能対応カーネルの場合）
11363 ・属するクラス（マルチプロセッサ対応カーネルの場合）
11364
11365 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
11366 対時間後である【NGKI2365】。
11367
11368 起動位相＋起動周期×(n－1) n=1, 2, …
11369
11370 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づれ
11371 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
11372 始、動作していない状態にすることを動作停止という。
11373
11374 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
11375 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
11376 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】。
11377
11378 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
11379 インは、カーネルドメインに限られる【NGKI2369】。
11380
11381 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】。
11382
11383 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
11384 TA_PHS 0x04U 周期ハンドラを生成した時刻を基準時刻とする
11385
11386 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
11387 していない状態となる【NGKI2371】。
11388
11389 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
11390 ドラを起動する時刻の基準時刻となる【NGKI2372】。TA_PHSを指定した場合には、
11391 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
11392 時刻）が、基準時刻となる【NGKI2373】。
11393
11394 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
11395 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
11396 ハンドラの起動処理時に設定される【NGKI2374】。
11397
11398 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、
11399 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
11400 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

11401 は、システム時刻管理プロセッサによって実行される。

11402

11403 C言語による周期ハンドラの記述形式は次の通り【NGKI2376】。

11404

```
11405     void cyclic_handler(intptr_t exinf)
11406     {
11407         周期ハンドラ本体
11408     }
```

11409

11410 exinfには、周期ハンドラの拡張情報が渡される【NGKI2377】。

11411

11412 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。

11413

```
11414     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
11415                     ネルでは、静的APIによって登録された周期ハンドラの数
11416                     に一致）【NGKI2378】
```

11417

11418 【TOPPERS/ASPカーネルにおける規定】

11419

11420 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】。

11421

11422 【TOPPERS/FMPカーネルにおける規定】

11423

11424 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】。

11425

11426 【TOPPERS/HRP2カーネルにおける規定】

11427

11428 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】。

11429

11430 【μITRON4.0仕様との関係】

11431

11432 TNUM_CYCIDは、μITRON4.0仕様に規定されていないカーネル構成マクロである。

11433 -----

11434 CRE_CYC 周期ハンドラの生成 [S] 【NGKI2379】

11435 acre_cyc 周期ハンドラの生成 [TD] 【NGKI2380】

11436

11437 【静的API】

```
11438     CRE_CYC(ID cycid, { ATR cycatr, intptr_t exinf, CYCHDR cychdr,
11439                       RELTIM cycetim, RELTIM cycphs })
```

11440

11441 【C言語API】

```
11442     ER_ID cycid = acre_cyc(const T_CCYC *pk_ccyc)
```

11443

11444 【パラメータ】

```
11445     ID      cycid      生成する周期ハンドラのID番号（CRE_CYCの場合）
11446     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
11447                       ポインタ（静的APIを除く）
```

11448

11449 *周期ハンドラの生成情報（パケットの内容）

```
11450     ATR      cycatr     周期ハンドラ属性
```

11451	intptr_t	exinf	周期ハンドラの拡張情報
11452	CYCHDR	cychdr	周期ハンドラの先頭番地
11453	RELTIM	cycetim	周期ハンドラの起動周期
11454	RELTIM	cycphs	周期ハンドラの起動位相
11455			
11456	【リターンパラメータ】		
11457	ER_ID	cycid	生成された周期ハンドラのID番号（正の値）またはエラーコード
11458			
11459			
11460	【エラーコード】		
11461	E_CTX	コンテキストエラー	
11462		・非タスクコンテキストからの呼出し [s] 【NGKI2381】	
11463		・CPUロック状態からの呼出し [s] 【NGKI2382】	
11464	E_RSATR	予約属性	
11465		・cycatrが無効 【NGKI2383】	
11466		・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】	
11467		・カーネルドメインの囲みの中に記述されていない [SP] 【NGKI2385】	
11468		・属するクラスの指定が有効範囲外 [sM] 【NGKI2386】	
11469		・クラスの囲みの中に記述されていない [SM] 【NGKI2387】	
11470		・その他の条件については機能の項を参照	
11471	E_PAR	パラメータエラー	
11472		・cychdrがプログラムの先頭番地として正しくない 【NGKI2388】	
11473		・cycetimが0, またはTMAX_RELTIMより大きい 【NGKI2397】	
11474		・cycphsがTMAX_RELTIMより大きい 【NGKI2399】	
11475	E_OACV	オブジェクトアクセス違反	
11476		・システム状態に対する管理操作が許可されていない [sP] 【NGKI2389】	
11477	E_MACV	メモリアクセス違反	
11478		・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】	
11479	E_NOID	ID番号不足	
11480		・割り付けられる周期ハンドラIDがない [sD] 【NGKI2391】	
11481	E_OBJ	オブジェクト状態エラー	
11482		・cycidで指定した周期ハンドラが登録済み（CRE_CYCの場合） 【NGKI2392】	
11483			
11484			
11485			
11486			
11487			
11488			
11489	【機能】		
11490			
11491	各パラメータで指定した周期ハンドラ生成情報に従って、周期ハンドラを生成する。具体的な振舞いは以下の通り。		
11492			
11493			
11494	cycatrにTA_STAを指定した場合、対象周期ハンドラは動作している状態となる 【NGKI2393】。次に周期ハンドラを起動する時刻は、サービスコールを呼び出した時刻（静的APIの場合はカーネルの起動時刻）から、cycphsで指定した相対時間後に設定される 【NGKI2394】。cycphsにcycetimより大きい値を指定してもよい 【NGKI2400】。		
11495			
11496			
11497			
11498			
11499			
11500	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		

11501 初期化される【NGKI2395】.

11502

11503 静的APIにおいては, `cycid`はオブジェクト識別名, `cycatr`, `cycetim`, `cycphs`は

11504 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

11505 【NGKI2396】.

11506

11507 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,

11508 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

11509 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】.

11510

11511 【補足説明】

11512

11513 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

11514 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.

11515 `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`

11516 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

11517ギュレータが警告メッセージを出力する.

11518

11519 【TOPPERS/ASPカーネルにおける規定】

11520

11521 ASPカーネルでは, `CRE_CYC`のみをサポートする【ASPS0173】. ただし, `TA_PHS`

11522 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

11523 ジでは, `acre_cyc`もサポートする【ASPS0175】.

11524

11525 【TOPPERS/FMPカーネルにおける規定】

11526

11527 FMPカーネルでは, `CRE_CYC`のみをサポートする【FMPS0148】. ただし, `TA_PHS`

11528 属性の周期ハンドラはサポートしない【FMPS0149】.

11529

11530 【TOPPERS/HRP2カーネルにおける規定】

11531

11532 HRP2カーネルでは, `CRE_CYC`のみをサポートする【HRPS0142】. ただし,

11533 `TA_PHS`属性の周期ハンドラはサポートしない【HRPS0143】.

11534

11535 【 μ ITRON4.0仕様との関係】

11536

11537 `cychdr`のデータ型を`CYCHDR`に変更した. また, `cycphs`に`cycetim`より大きい値を

11538 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

11539 した.

11540 -----

11541 `AID_CYC` 割付け可能な周期ハンドラIDの数の指定 [SD] 【NGKI2402】

11542

11543 【静的API】

11544 `AID_CYC(uint_t nocyc)`

11545

11546 【パラメータ】

11547 `uint_t` `nocyc` 割付け可能な周期ハンドラIDの数

11548

11549 【エラーコード】

11550 `E_RSATR` 予約属性

232

11601 E_NOEXS オブジェクト未登録
 11602 ・対象周期ハンドラが未登録【NGKI2415】
 11603 E_OACV オブジェクトアクセス違反
 11604 ・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〕
 11605 【NGKI2416】
 11606 E_MACV メモリアクセス違反
 11607 ・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
 11608 いない〔s〕【NGKI2417】
 11609 E_OBJ オブジェクト状態エラー
 11610 ・対象周期ハンドラは静的APIで生成された〔s〕【NGKI2418】
 11611 ・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
 11612 〔S〕【NGKI2419】
 11613

11614 【機能】

11615
 11616 cycled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
 11617 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
 11618 【NGKI2420】。
 11619

11620 静的APIにおいては、cycledはオブジェクト識別名、acptn1～acptn4は整数定数
 11621 式パラメータである【NGKI2421】。
 11622

11623 【TOPPERS/ASPカーネルにおける規定】

11624
 11625 ASPカーネルでは、SAC_CYC、sac_cycをサポートしない【ASPS0176】。
 11626

11627 【TOPPERS/FMPカーネルにおける規定】

11628
 11629 FMPカーネルでは、SAC_CYC、sac_cycをサポートしない【FMPS0150】。
 11630

11631 【TOPPERS/HRP2カーネルにおける規定】

11632
 11633 HRP2カーネルでは、SAC_CYCのみをサポートする【HRPS0144】。
 11634

11635 del_cyc 周期ハンドラの削除〔TD〕【NGKI2422】
 11636

11637 【C言語API】

11638 ER ercd = del_cyc(ID cycled)
 11639

11640 【パラメータ】

11641 ID cycled 対象周期ハンドラのID番号
 11642

11643 【リターンパラメータ】

11644 ER ercd 正常終了（E_OK）またはエラーコード
 11645

11646 【エラーコード】

11647 E_CTX コンテキストエラー
 11648 ・非タスクコンテキストからの呼出し【NGKI2423】
 11649 ・CPUロック状態からの呼出し【NGKI2424】
 11650 E_ID 不正ID番号

11651 ・cycidが有効範囲外【NGKI2425】
 11652 E_NOEXS オブジェクト未登録
 11653 ・対象周期ハンドラが未登録【NGKI2426】
 11654 E_OACV オブジェクトアクセス違反
 11655 ・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]
 11656 【NGKI2427】
 11657 E_OBJ オブジェクト状態エラー
 11658 ・対象周期ハンドラは静的APIで生成された【NGKI2428】
 11659
 11660 **【機能】**
 11661
 11662 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
 11663 いは以下の通り．
 11664
 11665 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
 11666 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，
 11667 動作していない状態にされた後に，登録が解除される【NGKI2430】．
 11668
 11669 **【TOPPERS/ASPカーネルにおける規定】**
 11670
 11671 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
 11672 機能拡張パッケージでは，del_cycをサポートする【ASPS0178】．
 11673
 11674 **【TOPPERS/FMPカーネルにおける規定】**
 11675
 11676 FMPカーネルでは，del_cycをサポートしない【FMPS0151】．
 11677
 11678 **【TOPPERS/HRP2カーネルにおける規定】**
 11679
 11680 HRP2カーネルでは，del_cycをサポートしない【HRPS0145】．
 11681 -----
 11682 sta_cyc 周期ハンドラの動作開始 [T] 【NGKI2431】
 11683
 11684 **【C言語API】**
 11685 ER ercd = sta_cyc(ID cycid)
 11686
 11687 **【パラメータ】**
 11688 ID cycid 対象周期ハンドラのID番号
 11689
 11690 **【リターンパラメータ】**
 11691 ER ercd 正常終了 (E_OK) またはエラーコード
 11692
 11693 **【エラーコード】**
 11694 E_CTX コンテキストエラー
 11695 ・非タスクコンテキストからの呼出し【NGKI2432】
 11696 ・CPUロック状態からの呼出し【NGKI2433】
 11697 E_ID 不正ID番号
 11698 ・cycidが有効範囲外【NGKI2434】
 11699 E_NOEXS オブジェクト未登録
 11700 ・対象周期ハンドラが未登録 [D] 【NGKI2435】

11701 E_OACV オブジェクトアクセス違反
 11702 ・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]
 11703 【NGKI2436】
 11704
 11705 **【機能】**
 11706
 11707 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
 11708 振舞いは以下の通り．
 11709
 11710 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
 11711 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，
 11712 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】．
 11713
 11714 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
 11715 時刻の再設定のみが行われる【NGKI2439】．
 11716
 11717 **【補足説明】**
 11718
 11719 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，
 11720 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
 11721 に設定される．
 11722
 11723 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
 11724 ドラを起動する時刻は変化しない．
 11725
 11726 **【 μ ITRON4.0仕様との関係】**
 11727
 11728 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
 11729 ハンドラが起動される時刻を変更した． μ ITRON4.0仕様では，sta_cycを呼び出
 11730 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
 11731 仕様では，起動位相で指定した相対時間後とした．
 11732 -----
 11733 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】
 11734
 11735 **【C言語API】**
 11736 ER ercd = msta_cyc(ID cycid, ID prcid)
 11737
 11738 **【パラメータ】**
 11739 ID cycid 対象周期ハンドラのID番号
 11740 ID prcid 周期ハンドラの割付け対象のプロセッサのID番号
 11741
 11742 **【リターンパラメータ】**
 11743 ER ercd 正常終了 (E_OK) またはエラーコード
 11744
 11745 **【エラーコード】**
 11746 E_CTX コンテキストエラー
 11747 ・非タスクコンテキストからの呼出し【NGKI2441】
 11748 ・CPUロック状態からの呼出し【NGKI2442】
 11749 E_NOSPT 未サポート機能
 11750 ・条件については機能の項を参照

11751 E_ID 不正ID番号
11752 ・ cycidが有効範囲外【NGKI2443】
11753 ・ prcidが有効範囲外【NGKI2444】
11754 E_PAR パラメータエラー
11755 ・ 条件については機能の項を参照
11756 E_NOEXS オブジェクト未登録
11757 ・ 対象周期ハンドラが未登録 [D] 【NGKI2445】
11758 E_OACV オブジェクトアクセス違反
11759 ・ 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]
11760 【NGKI2446】

11761
11762 **【機能】**

11763
11764 prcidで指定したプロセッサを割付けプロセッサとして、cycidで指定した周期
11765 ハンドラ（対象周期ハンドラ）を動作開始する。具体的な振舞いは以下の通り。
11766

11767 対象周期ハンドラが動作していない状態であれば、対象周期ハンドラの割付け
11768 プロセッサがprcidで指定したプロセッサに変更された後、対象周期ハンドラは
11769 動作している状態となる【NGKI2447】。次に周期ハンドラを起動する時刻は、
11770 msta_cycを呼び出して以降の最初の起動時刻に設定される【NGKI2448】。
11771

11772 対象周期ハンドラが動作している状態であれば、対象周期ハンドラの割付けプ
11773 ロセッサがprcidで指定したプロセッサに変更された後、次に周期ハンドラを起
11774 動する時刻の再設定が行われる【NGKI2449】。
11775

11776 対象周期ハンドラが実行中である場合には、割付けプロセッサを変更しても、
11777 実行中の周期ハンドラを実行するプロセッサは変更されない【NGKI2450】。対
11778 象周期ハンドラが変更後の割付けプロセッサで実行されるのは、次に起動され
11779 る時からである【NGKI2451】。
11780

11781 対象周期ハンドラの属するクラスの割付け可能プロセッサが、prcidで指定した
11782 プロセッサを含んでいない場合には、E_PARエラーとなる【NGKI2452】。
11783

11784 prcidにTPRC_INI (=0) を指定すると、対象周期ハンドラの割付けプロセッサ
11785 を、それが属するクラスの初期割付けプロセッサとする【NGKI2453】。
11786

11787 グローバルタイマ方式を用いている場合、msta_cycはE_NOSPTを返す
11788 【NGKI2454】。
11789

11790 **【補足説明】**

11791
11792 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、
11793 msta_cycを呼び出してから、対象周期ハンドラの起動位相で指定した相対時間
11794 後に設定される。
11795

11796 **【使用上の注意】**

11797
11798 msta_cycで実行中の周期ハンドラの割付けプロセッサを変更した場合、同じ周
11799 期ハンドラが異なるプロセッサで同時に実行される可能性がある。特に、対象
11800 周期ハンドラの起動位相が0の場合に、注意が必要である。

11801
11802 **【 μ ITRON4.0仕様との関係】**
11803
11804 μ ITRON4.0仕様に定義されていないサービスコールである。
11805

11806 stp_cyc 周期ハンドラの動作停止 [T] **【NGKI2455】**
11807
11808 **【C言語API】**
11809 ER ercd = stp_cyc(ID cycid)
11810
11811 **【パラメータ】**
11812 ID cycid 対象周期ハンドラのID番号
11813
11814 **【リターンパラメータ】**
11815 ER ercd 正常終了 (E_OK) またはエラーコード
11816
11817 **【エラーコード】**
11818 E_CTX コンテキストエラー
11819 ・非タスクコンテキストからの呼出し **【NGKI2456】**
11820 ・CPUロック状態からの呼出し **【NGKI2457】**
11821 E_ID 不正ID番号
11822 ・cycidが有効範囲外 **【NGKI2458】**
11823 E_NOEXS オブジェクト未登録
11824 ・対象周期ハンドラが未登録 [D] **【NGKI2459】**
11825 E_OACV オブジェクトアクセス違反
11826 ・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]
11827 **【NGKI2460】**
11828
11829 **【機能】**
11830
11831 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
11832 振舞いは以下の通り。
11833
11834 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
11835 **【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
11836 ずに正常終了する **【NGKI2462】**。
11837

11838 ref_cyc 周期ハンドラの状態参照 [T] **【NGKI2463】**
11839
11840 **【C言語API】**
11841 ER ercd = ref_cyc(ID cycid, T_RCYC *pk_rcyc)
11842
11843 **【パラメータ】**
11844 ID cycid 対象周期ハンドラのID番号
11845 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
11846 ポインタ
11847
11848 **【リターンパラメータ】**
11849 ER ercd 正常終了 (E_OK) またはエラーコード
11850

11851 *周期ハンドラの現在状態（パケットの内容）

11852 STAT cycstat 周期ハンドラの動作状態

11853 RELTIM lefttim 次に周期ハンドラを起動する時刻までの相対時間

11854 ID prcid 周期ハンドラの割付けプロセッサのID（マルチプ

11855 ロセッサ対応カーネルの場合）

11856

11857 **【エラーコード】**

11858 E_CTX コンテキストエラー

11859 • 非タスクコンテキストからの呼出し【NGKI2464】

11860 • CPUロック状態からの呼出し【NGKI2465】

11861 E_ID 不正ID番号

11862 • cycidが有効範囲外【NGKI2466】

11863 E_NOEXS オブジェクト未登録

11864 • 対象周期ハンドラが未登録〔D〕【NGKI2467】

11865 E_OACV オブジェクトアクセス違反

11866 • 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〕

11867 【NGKI2468】

11868 E_MACV メモリアクセス違反

11869 • 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

11870 いない）〔P〕【NGKI2469】

11871

11872 **【機能】**

11873

11874 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

11875 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】．

11876

11877 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

11878 される【NGKI2471】．

11879

11880 TCYC_STP 0x01U 周期ハンドラが動作していない状態

11881 TCYC_STA 0x02U 周期ハンドラが動作している状態

11882

11883 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

11884 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

11885 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

11886 【NGKI2473】．

11887

11888 マルチプロセッサ対応カーネルでは、prcidに、対象周期ハンドラの割付けプロ

11889 セッサのID番号が返される【NGKI2474】．

11890

11891 **【使用上の注意】**

11892

11893 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

11894 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

11895 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

11896 状態が変化している可能性があるためである．

11897

11898 **【μITRON4.0仕様との関係】**

11899

11900 TCYC_STPとTCYC_STAを値を変更した．

11901
11902
11903 4.6.3 アラームハンドラ
11904
11905 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
11906 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
11907 別する【NGKI2475】。
11908
11909 各アラームハンドラが持つ情報は次の通り【NGKI2476】。
11910
11911 • アラームハンドラ属性
11912 • アラームハンドラの動作状態
11913 • アラームハンドラを起動する時刻
11914 • 拡張情報
11915 • アラームハンドラの手頭番地
11916 • アクセス許可ベクタ（保護機能対応カーネルの場合）
11917 • 属する保護ドメイン（保護機能対応カーネルの場合）
11918 • 属するクラス（マルチプロセッサ対応カーネルの場合）
11919
11920 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
11921 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
11922 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。
11923
11924 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
11925 定される【NGKI2478】。
11926
11927 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
11928 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
11929 体的には、まず、アラームハンドラが動作していない状態にされる【NGKI2480】。
11930 その後、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
11931 【NGKI2481】。
11932
11933 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
11934 ドメインは、カーネルドメインに限られる【NGKI2482】。
11935
11936 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、
11937 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
11938 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
11939 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。
11940
11941 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】。
11942
11943 void alarm_handler(intptr_t exinf)
11944 {
11945 アラームハンドラ本体
11946 }
11947
11948 exinfには、アラームハンドラの拡張情報が渡される【NGKI2485】。
11949
11950 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。

11951
 11952 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
 11953 カーネルでは、静的APIによって登録されたアラームハン
 11954 ドラの数に一致）【NGKI2486】
 11955
 11956 【μITRON4.0仕様との関係】
 11957
 11958 TNUM_ALMIDは、μITRON4.0仕様に規定されていないカーネル構成マクロである。
 11959 -----
 11960 CRE_ALM アラームハンドラの生成 [S] 【NGKI2487】
 11961 acre_alm アラームハンドラの生成 [TD] 【NGKI2488】
 11962
 11963 【静的API】
 11964 CRE_ALM(ID almid, { ATR almatr, intptr_t exinf, ALMHDR almhdr })
 11965
 11966 【C言語API】
 11967 ER_ID almid = acre_alm(const T_CALM *pk_calm)
 11968
 11969 【パラメータ】
 11970 ID almid 生成するアラームハンドラのID番号（CRE_ALM
 11971 の場合）
 11972 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
 11973 へのポインタ（静的APIを除く）
 11974
 11975 *アラームハンドラの生成情報（パケットの内容）
 11976 ATR almatr アラームハンドラ属性
 11977 intptr_t exinf アラームハンドラの拡張情報
 11978 ALMHDR almhdr アラームハンドラの先頭番地
 11979
 11980 【リターンパラメータ】
 11981 ER_ID almid 生成されたアラームハンドラのID番号（正の値）
 11982 またはエラーコード
 11983
 11984 【エラーコード】
 11985 E_CTX コンテキストエラー
 11986 ・非タスクコンテキストからの呼出し [s] 【NGKI2489】
 11987 ・CPUロック状態からの呼出し [s] 【NGKI2490】
 11988 E_RSATR 予約属性
 11989 ・almatrが無効【NGKI2491】
 11990 ・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
 11991 メイン以外 [sP] 【NGKI2492】
 11992 ・カーネルドメインの囲みの中に記述されていない [SP]
 11993 【NGKI2493】
 11994 ・属するクラスの指定が有効範囲外 [sM] 【NGKI2494】
 11995 ・クラスの囲みの中に記述されていない [SM] 【NGKI2495】
 11996 ・その他の条件については機能の項を参照
 11997 E_PAR パラメータエラー
 11998 ・almhdrがプログラムの先頭番地として正しくない【NGKI2496】
 11999 E_OACV オブジェクトアクセス違反
 12000 ・システム状態に対する管理操作が許可されていない [sP]

12001		【NGKI2497】
12002	E_MACV	メモリアクセス違反
12003		・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
12004		いない〔sP〕【NGKI2498】
12005	E_NOID	ID番号不足
12006		・割り付けられるアラームハンドラIDがない〔sD〕【NGKI2499】
12007	E_OBJ	オブジェクト状態エラー
12008		・almidで指定したアラームハンドラが登録済み（CRE_ALMの
12009		場合）【NGKI2500】
12010		
12011	【機能】	
12012		
12013	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	
12014	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	
12015	【NGKI2501】。	
12016		
12017	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	
12018	タ、exinfとalmhdrは一般定数式パラメータである【NGKI2502】。	
12019		
12020	マルチプロセッサ対応カーネルでグローバルタイム方式を用いている場合で、	
12021	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	
12022	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】。	
12023		
12024	【TOPPERS/ASPカーネルにおける規定】	
12025		
12026	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	
12027	成機能拡張パッケージでは、acre_almもサポートする【ASPS0180】。	
12028		
12029	【TOPPERS/FMPカーネルにおける規定】	
12030		
12031	FMPカーネルでは、CRE_ALMのみをサポートする【FMPS0152】。	
12032		
12033	【TOPPERS/HRP2カーネルにおける規定】	
12034		
12035	HRP2カーネルでは、CRE_ALMのみをサポートする【HRPS0146】。	
12036		
12037	【μITRON4.0仕様との関係】	
12038		
12039	almhdrのデータ型をALMHDRに変更した。	
12040	-----	
12041	AID_ALM	割付け可能なアラームハンドラIDの数の指定〔SD〕【NGKI2504】
12042		
12043	【静的API】	
12044	AID_ALM(uint_t noalm)	
12045		
12046	【パラメータ】	
12047	uint_t	noalm 割付け可能なアラームハンドラIDの数
12048		
12049	【エラーコード】	
12050	E_RSATR	予約属性

12051	・カーネルドメインの囲みの中に記述されていない [P] 【NGKI2505】
12052	・クラスの囲みの中に記述されていない [M] 【NGKI2506】
12053	・その他の条件については機能の項を参照

12055 【機能】

12057 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
12058 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
12059 **【NGKI2507】** .

12061 noalmは整数定数式パラメータである【NGKI2508】.

12063 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、
12064 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
12065 サのみでない場合には、E_RSATRエラーとなる【NGKI2509】。

12067 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】

12070 【静的API】

```

12071      SAC_ALM(ID almid, { ACPTN acptn1, ACPTN acptn2,
12072                          ACPTN acptn3, ACPTN acptn4 })

```

12074 【C言語API】

```
12075      ER ercd = sac_alm(ID almid, const ACVCT *p_acvct)
```

12077 【パラメータ】

12078	ID	almid	対象アラームハンドラのID番号
12079	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）
12080			

12082 *アクセス許可ベクタ (パケットの内容)

12083	ACPTN	acptn1	通常操作1のアクセス許可パターン
-------	-------	--------	------------------

12084	ACPTN	acptn2	通常操作2のアクセス許可パターン
-------	-------	--------	------------------

12085	ACPTN	acptn3	管理操作のアクセス許可パターン
-------	-------	--------	-----------------

12086	ACPTN	acptn4	参照操作のアクセス許可パターン
-------	-------	--------	-----------------

12088 【リターンパラメータ】

12089	ER	ercd	正常終了 (E_OK) またはエラーコード
-------	----	------	-----------------------

12091 【エラーコード】

12092	E CTX	コンテキストエラー
-------	-------	-----------

12093 ・非タスクコンテキストからの呼出し [s] 【NGKI2512】

12094 ・CPUロック状態からの呼出し〔s〕 【NGKI2513】

12095	E_ID	不正ID番号
12096		・almidが有効範囲外〔s〕【NGKI2514】

12097	E_RSATR	予約属性
-------	---------	------

12098 ・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
12099 記述されていない〔S〕 【NGKI2515】

12100 ・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

12101 れていない [SM] 【NGKI2516】

12102 E_NOEXS オブジェクト未登録

12103 ・対象アラームハンドラが未登録 【NGKI2517】

12104 E_OACV オブジェクトアクセス違反

12105 ・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

12106 ない [s] 【NGKI2518】

12107 E_MACV メモリアクセス違反

12108 ・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

12109 いない [s] 【NGKI2519】

12110 E_OBJ オブジェクト状態エラー

12111 ・対象アラームハンドラは静的APIで生成された [s] 【NGKI2520】

12112 ・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

12113 済み [S] 【NGKI2521】

12114

12115 **【機能】**

12116

12117 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

12118 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

12119 する 【NGKI2522】 .

12120

12121 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

12122 式パラメータである 【NGKI2523】 .

12123

12124 **【TOPPERS/ASPカーネルにおける規定】**

12125

12126 ASPカーネルでは、SAC_ALM, sac_almをサポートしない 【ASPS0181】 .

12127

12128 **【TOPPERS/FMPカーネルにおける規定】**

12129

12130 FMPカーネルでは、SAC_ALM, sac_almをサポートしない 【FMPS0153】 .

12131

12132 **【TOPPERS/HRP2カーネルにおける規定】**

12133

12134 HRP2カーネルでは、SAC_ALMのみをサポートする 【HRPS0147】 .

12135 -----

12136 del_alm アラームハンドラの削除 [TD] 【NGKI2524】

12137

12138 **【C言語API】**

12139 ER ercd = del_alm(ID almid)

12140

12141 **【パラメータ】**

12142 ID almid 対象アラームハンドラのID番号

12143

12144 **【リターンパラメータ】**

12145 ER ercd 正常終了 (E_OK) またはエラーコード

12146

12147 **【エラーコード】**

12148 E_CTX コンテキストエラー

12149 ・非タスクコンテキストからの呼出し 【NGKI2525】

12150 ・CPUロック状態からの呼出し 【NGKI2526】

12151 E_ID 不正ID番号
 12152 ・ almidが有効範囲外【NGKI2527】
 12153 E_NOEXS オブジェクト未登録
 12154 ・ 対象アラームハンドラが未登録【NGKI2528】
 12155 E_OACV オブジェクトアクセス違反
 12156 ・ 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
 12157 い【P】【NGKI2529】
 12158 E_OBJ オブジェクト状態エラー
 12159 ・ 対象アラームハンドラは静的APIで生成された【NGKI2530】
 12160
 12161 **【機能】**
 12162
 12163 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
 12164 的な振舞いは以下の通り．
 12165
 12166 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
 12167 状態に戻る【NGKI2531】．対象アラームハンドラが動作している状態であっ
 12168 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
 12169 【NGKI2532】．
 12170
 12171 **【TOPPERS/ASPカーネルにおける規定】**
 12172
 12173 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
 12174 機能拡張パッケージでは、del_almをサポートする【ASPS0183】．
 12175
 12176 **【TOPPERS/FMPカーネルにおける規定】**
 12177
 12178 FMPカーネルでは、del_almをサポートしない【FMPS0154】．
 12179
 12180 **【TOPPERS/HRP2カーネルにおける規定】**
 12181
 12182 HRP2カーネルでは、del_almをサポートしない【HRPS0148】．
 12183 -----
 12184 sta_alm アラームハンドラの動作開始【T】【NGKI2533】
 12185 ista_alm アラームハンドラの動作開始【I】【NGKI2534】
 12186
 12187 **【C言語API】**
 12188 ER ercd = sta_alm(ID almid, RELTIM almtim)
 12189 ER ercd = ista_alm(ID almid, RELTIM almtim)
 12190
 12191 **【パラメータ】**
 12192 ID almid 対象アラームハンドラのID番号
 12193 RELTIM almtim アラームハンドラの起動時刻（相対時間）
 12194
 12195 **【リターンパラメータ】**
 12196 ER ercd 正常終了（E_OK）またはエラーコード
 12197
 12198 **【エラーコード】**
 12199 E_CTX コンテキストエラー
 12200 ・ 非タスクコンテキストからの呼出し（sta_almの場合）【NGKI2535】

12201 ・タスクコンテキストからの呼出し (ista_almの場合) 【NGKI2536】
 12202 ・CPUロック状態からの呼出し
 12203 E_ID 不正ID番号
 12204 ・almidが有効範囲外 【NGKI2537】
 12205 E_PAR パラメータエラー
 12206 ・almtimがTMAX_RELTIMより大きい 【NGKI2538】
 12207 E_NOEXS オブジェクト未登録
 12208 ・対象アラームハンドラが未登録 [D] 【NGKI2539】
 12209 E_OACV オブジェクトアクセス違反
 12210 ・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
 12211 ない (sta_almの場合) [P] 【NGKI2540】
 12212

12213 【機能】

12214

12215 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。
 12216 具体的な振舞いは以下の通り。
 12217

12218 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
 12219 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、
 12220 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
 12221 【NGKI2542】。
 12222

12223 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
 12224 する時刻の再設定のみが行われる 【NGKI2543】。
 12225

12226 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】
 12227 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】
 12228

12229 【C言語API】

12230 ER ercd = msta_alm(ID almid, RELTIM almtim, ID prcid)
 12231 ER ercd = imsta_alm(ID almid, RELTIM almtim, ID prcid)
 12232

12233 【パラメータ】

12234 ID almid 対象アラームハンドラのID番号
 12235 RELTIM almtim アラームハンドラの起動時刻 (相対時間)
 12236 ID prcid アラームハンドラの割付け対象のプロセッサの
 12237 ID番号
 12238

12239 【リターンパラメータ】

12240 ER ercd 正常終了 (E_OK) またはエラーコード
 12241

12242 【エラーコード】

12243 E_CTX コンテキストエラー
 12244 ・非タスクコンテキストからの呼出し (msta_almの場合)
 12245 【NGKI2546】
 12246 ・タスクコンテキストからの呼出し (imsta_almの場合) 【NGKI2547】
 12247 ・CPUロック状態からの呼出し 【NGKI2548】
 12248 E_NOSPT 未サポート機能
 12249 ・条件については機能の項を参照
 12250 E_ID 不正ID番号

12251		・ almidが有効範囲外 【NGKI2549】
12252		・ prcidが有効範囲外 【NGKI2550】
12253	E_PAR	パラメータエラー
12254		・ almtimがTMAX_RELTIMより大きい 【NGKI2551】
12255		・ その他の条件については機能の項を参照
12256	E_NOEXS	オブジェクト未登録
12257		・ 対象アラームハンドラが未登録 [D] 【NGKI2552】
12258	E_OACV	オブジェクトアクセス違反
12259		・ 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】
12260		
12261		
12262	【機能】	
12263		
12264	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。	
12265		
12266		
12267		
12268	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】。	
12269		
12270		
12271		
12272		
12273		
12274	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】。	
12275		
12276		
12277		
12278	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】。	
12279		
12280		
12281		
12282		
12283	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】。	
12284		
12285		
12286	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】。	
12287		
12288		
12289	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】。	
12290		
12291		
12292	【使用上の注意】	
12293		
12294	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。	
12295		
12296		
12297		
12298	【μITRON4.0仕様との関係】	
12299		
12300	μITRON4.0仕様に定義されていないサービスコールである。	

```

12301 -----
12302 stp_alm      アラームハンドラの動作停止 [T]  【NGKI2562】
12303 istp_alm     アラームハンドラの動作停止 [I]  【NGKI2563】
12304
12305 【C言語API】
12306     ER ercd = stp_alm(ID almid)
12307     ER ercd = istp_alm(ID almid)
12308
12309 【パラメータ】
12310     ID          almid      対象アラームハンドラのID番号
12311
12312 【リターンパラメータ】
12313     ER          ercd      正常終了 (E_OK) またはエラーコード
12314
12315 【エラーコード】
12316     E_CTX      コンテキストエラー
12317                ・非タスクコンテキストからの呼出し (stp_almの場合)  【NGKI2564】
12318                ・タスクコンテキストからの呼出し (istp_almの場合)  【NGKI2565】
12319                ・CPUロック状態からの呼出し 【NGKI2566】
12320     E_ID       不正ID番号
12321                ・almidが有効範囲外 【NGKI2567】
12322     E_NOEXS    オブジェクト未登録
12323                ・対象アラームハンドラが未登録 [D]  【NGKI2568】
12324     E_OACV     オブジェクトアクセス違反
12325                ・対象アラームハンドラに対する通常操作2が許可されていな
12326                  い (stp_almの場合) [P]  【NGKI2569】
12327
12328 【機能】
12329
12330 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．
12331 具体的な振舞いは以下の通り．
12332
12333 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
12334 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
12335 行われずに正常終了する【NGKI2571】．
12336 -----
12337 ref_alm      アラームハンドラの状態参照 [T]  【NGKI2572】
12338
12339 【C言語API】
12340     ER ercd = ref_alm(ID almid, T_RALM *pk_ralm)
12341
12342 【パラメータ】
12343     ID          almid      対象アラームハンドラのID番号
12344     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
12345                          へのポインタ
12346
12347 【リターンパラメータ】
12348     ER          ercd      正常終了 (E_OK) またはエラーコード
12349
12350     ＊アラームハンドラの現在状態（パケットの内容）

```

12351	STAT	almstat	アラームハンドラの動作状態
12352	RELTIM	lefttim	アラームハンドラを起動する時刻までの相対時間
12353	ID	pcrid	アラームハンドラの割付けプロセッサのID (マルチプロセッサ対応カーネルの場合)

12355

12356 **【エラーコード】**

12357	E_CTX	コンテキストエラー
12358		・非タスクコンテキストからの呼出し【NGKI2573】
12359		・CPUロック状態からの呼出し【NGKI2574】
12360	E_ID	不正ID番号
12361		・almidが有効範囲外【NGKI2575】
12362	E_NOEXS	オブジェクト未登録
12363		・対象アラームハンドラが未登録 [D] 【NGKI2576】
12364	E_OACV	オブジェクトアクセス違反
12365		・対象アラームハンドラに対する参照操作が許可されていない [P] 【NGKI2577】
12366	E_MACV	メモリアクセス違反
12367		・pk_ralmが指すメモリ領域への書込みアクセスが許可されていない [P] 【NGKI2578】

12370

12371 **【機能】**

12372

12373 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
 12374 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】.

12375

12376 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
 12377 が返される【NGKI2580】.

12378

12379	TALM_STP	0x01U	アラームハンドラが動作していない状態
12380	TALM_STA	0x02U	アラームハンドラが動作している状態

12381

12382 対象アラームハンドラが動作している状態である場合には, lefttimに, アラーム
 12383 ハンドラ起動する時刻までの相対時間が返される【NGKI2581】. 対象アラーム
 12384 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
 12385 【NGKI2582】.

12386

12387 マルチプロセッサ対応カーネルでは, pcridに, 対象アラームハンドラの割付け
 12388 プロセッサのID番号が返される【NGKI2583】.

12389

12390 **【使用上の注意】**

12391

12392 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
 12393 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
 12394 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
 12395 ンドラの状態が変化している可能性があるためである.

12396

12397 **【μITRON4.0仕様との関係】**

12398

12399 TALM_STPとTALM_STAを値を変更した.

12400

12401

12402 4.6.4 オーバランハンドラ

12403

12404 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
12405 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
12406 は、システムで1つのみ登録することができる【NGKI2584】。

12407

12408 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
12409 【NGKI2585】。

12410

12411 ・オーバランハンドラの動作状態

12412 ・残りプロセッサ時間

12413

12414 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
12415 いない状態のいずれかをとる【NGKI2586】。残りプロセッサ時間は、オーバラ
12416 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
12417 時間を表す。

12418

12419 オーバランハンドラの動作状態は、タスクの起動時に、動作していない状態に
12420 初期化される【NGKI2587】。

12421

12422 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
12423 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】。
12424 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
12425 ドラが起動される【NGKI2589】。

12426

12427 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
12428 チン、それらから呼び出したサービスクール（拡張サービスクールを含む）の
12429 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
12430 理の割込みハンドラ（割込みサービスクール、周期ハンドラ、アラームハン
12431 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
12432 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/
12433 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
12434 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
12435 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
12436 【NGKI2592】。

12437

12438 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
12439 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
12440 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
12441 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
12442 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
12443 サ時間の計測精度はターゲットに依存する【NGKI2596】。

12444

12445 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
12446 属する【NGKI2597】。

12447

12448 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
12449 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、
12450 TOPPERS_SUPPORT_OVRHDRがマクロ定義される【NGKI2599】。サポートされてい

12451 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
12452 ラーが返るか、リンク時にエラーとなる【NGKI2600】。

12453

12454 オーバランハンドラ機能に用いるデータ型は次の通り。

12455

12456 OVRTIM プロセッサ時間（符号無し整数，単位はマイクロ秒，ulong_t
12457 に定義）【NGKI2601】

12458

12459 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
12460 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】。

12461

12462 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】。

12463

```
12464     void overrun_handler(ID tskid, intptr_t exinf)
12465     {
12466         オーバランハンドラ本体
12467     }
```

12468

12469 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
12470 張情報が，それぞれ渡される【NGKI2605】。

12471

12472 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。

12473

12474 TMAX_OVRTIM プロセッサ時間に指定できる最大値【NGKI2606】

12475

12476 TOPPERS_SUPPORT_OVRHDR オーバランハンドラ機能がサポートされて
12477 いる【NGKI2607】

12478

12479 【使用上の注意】

12480

12481 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
12482 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。

12483

12484 【TOPPERS/ASPカーネルにおける規定】

12485

12486 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，
12487 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
12488 を追加することができる【ASPS0185】。

12489

12490 【TOPPERS/FMPカーネルにおける規定】

12491

12492 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】。

12493

12494 【TOPPERS/HRP2カーネルにおける規定】

12495

12496 HRP2カーネルでは，オーバランハンドラをサポートする【HRPS0149】。

12497

12498 【μITRON4.0仕様との関係】

12499

12500 OVRTIMの時間単位は，μITRON4.0仕様では実装定義としていたが，この仕様で

12501 はマイクロ秒と規定した。
 12502
 12503 TMAX_OVRTIMは、 μ ITRON4.0仕様に規定されていないカーネル構成マクロである。
 12504 -----
 12505 DEF_OVR オーバランハンドラの定義 [S] 【NGKI2608】
 12506 def_ovr オーバランハンドラの定義 [TD] 【NGKI2609】
 12507
 12508 【静的API】
 12509 DEF_OVR({ ATR ovratr, OVRHDR ovrhdr })
 12510
 12511 【C言語API】
 12512 ER ercd = def_ovr(const T_DOVR *pk_dovr)
 12513
 12514 【パラメータ】
 12515 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
 12516 トへのポインタ（静的APIを除く）
 12517
 12518 * オーバランハンドラの定義情報（パケットの内容）
 12519 ATR ovratr オーバランハンドラ属性
 12520 OVRHDR ovrhdr オーバランハンドラの先頭番地
 12521
 12522 【リターンパラメータ】
 12523 ER ercd 正常終了 (E_OK) またはエラーコード
 12524
 12525 【エラーコード】
 12526 E_CTX コンテキストエラー
 12527 ・非タスクコンテキストからの呼出し [s] 【NGKI2610】
 12528 ・CPUロック状態からの呼出し [s] 【NGKI2611】
 12529 E_RSATR 予約属性
 12530 ・ovratrが無効 【NGKI2612】
 12531 ・その他の条件については機能の項を参照
 12532 E_PAR パラメータエラー
 12533 ・ovrhdrがプログラムの先頭番地として正しくない 【NGKI2613】
 12534 E_OACV オブジェクトアクセス違反
 12535 ・システム状態に対する管理操作が許可されていない [sP]
 12536 【NGKI2614】
 12537 E_MACV メモリアクセス違反
 12538 ・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
 12539 いない [sP] 【NGKI2615】
 12540 E_OBJ オブジェクト状態エラー
 12541 ・条件については機能の項を参照
 12542
 12543 【機能】
 12544
 12545 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
 12546 ンドラを定義する 【NGKI2616】。ただし、def_ovrにおいてpk_dovrをNULLにし
 12547 た場合には、オーバランハンドラの定義を解除する 【NGKI2617】。
 12548
 12549 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
 12550 メータである 【NGKI2618】。

12551
12552 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
12553 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
12554 る場合には、E_OBJエラーとなる【NGKI2619】。
12555
12556 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
12557 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
12558 【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
12559 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
12560 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】。
12561 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
12562 ラーは検出されない【NGKI2623】。
12563
12564 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
12565 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
12566 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
12567 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
12568 指定した場合にはE_RSATRエラーとなる【NGKI2626】。ただし、
12569 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
12570 出されない【NGKI2627】。
12571
12572 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
12573 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
12574 となる【NGKI2628】。
12575
12576 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、
12577 すべてのタスクに対して動作していない状態となる【NGKI2629】。
12578
12579 【使用上の注意】
12580
12581 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
12582 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
12583 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
12584 なるため、注意が必要である。
12585
12586 【TOPPERS/ASPカーネルにおける規定】
12587
12588 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
12589 ポートする【ASPS0186】。
12590
12591 【TOPPERS/HRP2カーネルにおける規定】
12592
12593 HRP2カーネルでは、DEF_OVRのみをサポートする【HRPS0150】。
12594
12595 【μITRON4.0仕様との関係】
12596
12597 ovrrhdrのデータ型をOVRHDRに変更した。
12598
12599 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、
12600 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

12601 度定義を解除してから，再度定義する必要がある．

12602 -----

12603 sta_ovr オーバランハンドラの動作開始 [T] 【NGKI2630】

12604 ista_ovr オーバランハンドラの動作開始 [I] 【NGKI2631】

12605

12606 【C言語API】

12607 ER ercd = sta_ovr(ID tskid, OVRTIM ovrtime)

12608 ER ercd = ista_ovr(ID tskid, OVRTIM ovrtime)

12609

12610 【パラメータ】

12611 ID tskid 対象タスクのID番号

12612 OVRTIM ovrtime 対象タスクの残りプロセッサ時間

12613

12614 【リターンパラメータ】

12615 ER ercd 正常終了 (E_OK) またはエラーコード

12616

12617 【エラーコード】

12618 E_CTX コンテキストエラー

12619 ・非タスクコンテキストからの呼出し (sta_ovrの場合) 【NGKI2632】

12620 ・タスクコンテキストからの呼出し (ista_ovrの場合) 【NGKI2633】

12621 ・CPUロック状態からの呼出し 【NGKI2634】

12622 E_ID 不正ID番号

12623 ・tskidが有効範囲外 【NGKI2635】

12624 E_NOEXS オブジェクト未登録

12625 ・対象タスクが未登録 [D] 【NGKI2636】

12626 E_OACV オブジェクトアクセス違反

12627 ・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
12628 の場合) [P] 【NGKI2637】

12629 E_PAR パラメータエラー

12630 ・ovrtimが0，またはTMAX_OVRTIMより大きい 【NGKI2643】

12631 E_OBJ オブジェクト状態エラー

12632 ・オーバランハンドラが定義されていない 【NGKI2638】

12633

12634 【機能】

12635

12636 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
12637 開始する．具体的な振舞いは以下の通り．

12638

12639 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

12640 り，残りプロセッサ時間は，ovrtimに指定した時間に設定される 【NGKI2639】．

12641 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

12642 ロセッサ時間の設定のみが行われる 【NGKI2640】．

12643

12644 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
12645 となる 【NGKI2641】．

12646

12647 【μITRON4.0仕様との関係】

12648

12649 ista_ovrは，μITRON4.0仕様に定義されていないサービスコールである．

12650 -----

12651 stp_ovr オーバランハンドラの動作停止 [T] 【NGKI2644】
 12652 istp_ovr オーバランハンドラの動作停止 [I] 【NGKI2645】
 12653
 12654 **【C言語API】**
 12655 ER ercd = stp_ovr(ID tskid)
 12656 ER ercd = istp_ovr(ID tskid)
 12657
 12658 **【パラメータ】**
 12659 ID tskid 対象タスクのID番号
 12660
 12661 **【リターンパラメータ】**
 12662 ER ercd 正常終了 (E_OK) またはエラーコード
 12663
 12664 **【エラーコード】**
 12665 E_CTX コンテキストエラー
 12666 ・非タスクコンテキストからの呼出し (stp_ovrの場合) 【NGKI2646】
 12667 ・タスクコンテキストからの呼出し (istp_ovrの場合) 【NGKI2647】
 12668 ・CPUロック状態からの呼出し 【NGKI2648】
 12669 E_ID 不正ID番号
 12670 ・tskidが有効範囲外 【NGKI2649】
 12671 E_NOEXS オブジェクト未登録
 12672 ・対象タスクが未登録 [D] 【NGKI2650】
 12673 E_OACV オブジェクトアクセス違反
 12674 ・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
 12675 の場合) [P] 【NGKI2651】
 12676 E_OBJ オブジェクト状態エラー
 12677 ・オーバランハンドラが定義されていない 【NGKI2652】
 12678
 12679 **【機能】**
 12680
 12681 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
 12682 停止する。具体的な振舞いは以下の通り。
 12683
 12684 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
 12685 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
 12686 状態であれば、何も行われずに正常終了する【NGKI2654】。
 12687
 12688 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
 12689 となる【NGKI2655】。
 12690
 12691 **【μITRON4.0仕様との関係】**
 12692
 12693 istp_ovrは、μITRON4.0仕様に定義されていないサービスコールである。
 12694 -----
 12695 ref_ovr オーバランハンドラの状態参照 [T] 【NGKI2656】
 12696
 12697 **【C言語API】**
 12698 ER ercd = ref_ovr(ID tskid, T_ROVR *pk_rovr)
 12699
 12700 **【パラメータ】**

12701	ID	tskid	対象タスクのID番号
12702	T_ROVR *	pk_rovr	オーバランハンドラ現在の状態を入れるパケットへのポインタ
12703			
12704			
12705	【リターンパラメータ】		
12706	ER	ered	正常終了 (E_OK) またはエラーコード
12707			
12708	* タスクの現在状態 (パケットの内容)		
12709	STAT	ovrstat	オーバランハンドラの動作状態
12710	OVRTIM	leftotm	残りプロセッサ時間
12711			
12712	【エラーコード】		
12713	E_CTX	コンテキストエラー	
12714		・ 非タスクコンテキストからの呼出し【NGKI2657】	
12715		・ CPUロック状態からの呼出し【NGKI2658】	
12716	E_ID	不正ID番号	
12717		・ tskidが有効範囲外【NGKI2659】	
12718	E_NOEXS	オブジェクト未登録	
12719		・ 対象タスクが未登録 [D]【NGKI2660】	
12720	E_OACV	オブジェクトアクセス違反	
12721		・ 対象タスクに対する参照操作が許可されていない [P]【NGKI2661】	
12722	E_MACV	メモリアクセス違反	
12723		・ pk_rovrが指すメモリ領域への書き込みアクセスが許可されていない [P]【NGKI2662】	
12724	E_OBJ	オブジェクト状態エラー	
12725		・ オーバランハンドラが定義されていない【NGKI2663】	
12726			
12727			
12728	【機能】		
12729			
12730	tskidで指定したタスク (対象タスク) に対するオーバランハンドラの現在状態を参照する. 参照した現在状態は, pk_rovrで指定したメモリ領域に返される【NGKI2664】.		
12731			
12732			
12733			
12734	ovrstatには, 対象タスクに対するオーバランハンドラの動作状態を表す次のいずれかの値が返される【NGKI2665】.		
12735			
12736			
12737	TOVR_STP	0x01U	オーバランハンドラが動作していない状態
12738	TOVR_STA	0x02U	オーバランハンドラが動作している状態
12739			
12740	対象タスクに対してオーバランハンドラが動作している状態の場合には, leftotmに, オーバランハンドラが起動されるまでの残りプロセッサ時間が返される【NGKI2666】. オーバランハンドラが起動される直前には, leftotmに0が返される可能性がある【NGKI2667】. オーバランハンドラが動作していない状態の場合には, leftotmの値は保証されない【NGKI2668】.		
12741			
12742			
12743			
12744			
12745			
12746	tskidにTSK_SELF (=0) を指定すると, 自タスクが対象タスクとなる【NGKI2669】.		
12747			
12748			
12749	【使用上の注意】		
12750			

12751 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
 12752 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
 12753 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
 12754 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。

12755

12756 **【未決定事項】**

12757

12758 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
 12759 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
 12760 照できるとするかどうかは、今後の課題である。

12761

12762 **【 μ ITRON4.0仕様との関係】**

12763

12764 TOVR_STPとTOVR_STAを値を変更した。

12765 -----

12766

12767 4.7 システム状態管理機能

12768

12769 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
 12770 変更／参照するための機能である。

12771

12772 -----

12773 SAC_SYS システム状態のアクセス許可ベクタの設定 [SP] **【NGKI2670】**

12774 sac_sys システム状態のアクセス許可ベクタの設定 [TPD] **【NGKI2671】**

12775

12776 **【静的API】**

12777 SAC_SYS({ ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })

12778

12779 **【C言語API】**

12780 ER ercd = sac_sys(const ACVCT *p_acvct)

12781

12782 **【パラメータ】**

12783 ACVCT * p_acvct アクセス許可ベクタを入れたパケットへのポ
 12784 イント（静的APIを除く）

12785

12786 *アクセス許可ベクタ（パケットの内容）

12787 ACPTN acptn1 通常操作1のアクセス許可パターン

12788 ACPTN acptn2 通常操作2のアクセス許可パターン

12789 ACPTN acptn3 管理操作のアクセス許可パターン

12790 ACPTN acptn4 参照操作のアクセス許可パターン

12791

12792 **【リターンパラメータ】**

12793 ER ercd 正常終了 (E_OK) またはエラーコード

12794

12795 **【エラーコード】**

12796 E_CTX コンテキストエラー

12797 ・非タスクコンテキストからの呼出し [s] **【NGKI2672】**

12798 ・CPUロック状態からの呼出し [s] **【NGKI2673】**

12799 E_RSATR 予約属性

12800 ・カーネルドメインの囲みの中に記述されていない [S] **【NGKI2674】**

12801 ・クラスの囲みの中に記述されている [SM] 【NGKI2675】
12802 E_OACV オブジェクトアクセス違反
12803 ・カーネルドメイン以外からの呼出し [s] 【NGKI2676】
12804 E_OBJ オブジェクト状態エラー
12805 ・システム状態のアクセス許可ベクタが設定済み [S] 【NGKI2677】
12806
12807 **【機能】**
12808
12809 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
12810 パラメータで指定した値に設定する 【NGKI2678】。
12811
12812 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】。
12813
12814 **【TOPPERS/ASPカーネルにおける規定】**
12815
12816 ASPカーネルでは、SAC_SYS, sac_sysをサポートしない 【ASPS0187】。
12817
12818 **【TOPPERS/FMPカーネルにおける規定】**
12819
12820 FMPカーネルでは、SAC_SYS, sac_sysをサポートしない 【FMPS0156】。
12821
12822 **【TOPPERS/HRP2カーネルにおける規定】**
12823
12824 HRP2カーネルでは、SAC_SYSのみをサポートする 【HRPS0151】。
12825
12826 **【TOPPERS/SSPカーネルにおける規定】**
12827
12828 SSPカーネルでは、SAC_SYS, sac_sysをサポートしない 【SSPS0130】。
12829 -----
12830 rot_rdq タスクの優先順位の回転 [T] 【NGKI2680】
12831 irot_rdq タスクの優先順位の回転 [I] 【NGKI2681】
12832
12833 **【C言語API】**
12834 ER ercd = rot_rdq(PRI tskpri)
12835 ER ercd = irot_rdq(PRI tskpri)
12836
12837 **【パラメータ】**
12838 PRI tskpri 回転対象の優先度（対象優先度）
12839
12840 **【リターンパラメータ】**
12841 ER ercd 正常終了（E_OK）またはエラーコード
12842
12843 **【エラーコード】**
12844 E_CTX コンテキストエラー
12845 ・非タスクコンテキストからの呼出し（rot_rdqの場合） 【NGKI2682】
12846 ・タスクコンテキストからの呼出し（irot_rdqの場合） 【NGKI2683】
12847 ・CPUロック状態からの呼出し 【NGKI2684】
12848 E_NOSPT 未サポート機能
12849 ・条件については機能の項を参照
12850 E_PAR パラメータエラー

12851 ・ tskpriが有効範囲外【NGKI2685】
 12852 E_OACV オブジェクトアクセス違反
 12853 ・ 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]
 12854 【NGKI2686】
 12855
 12856 【機能】
 12857
 12858 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、
 12859 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
 12860 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
 12861 1つのみの場合には、何も行われずに正常終了する【NGKI2688】．
 12862
 12863 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
 12864 優先度が対象優先度となる【NGKI2689】．
 12865
 12866 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
 12867 が制約タスクの場合には、E_NOSPTエラーとなる【NGKI2690】．
 12868
 12869 【TOPPERS/SSPカーネルにおける規定】
 12870
 12871 SSPカーネルでは、rot_rdq, irot_rdqをサポートしない【SSPS0131】．
 12872 -----
 12873 mrot_rdq プロセッサ指定でのタスクの優先順位の回転 [TM] 【NGKI2691】
 12874 imrot_rdq プロセッサ指定でのタスクの優先順位の回転 [IM] 【NGKI2692】
 12875
 12876 【C言語API】
 12877 ER ercd = mrot_rdq(PRI tskpri, ID preid)
 12878 ER ercd = imrot_rdq(PRI tskpri, ID preid)
 12879
 12880 【パラメータ】
 12881 PRI tskpri 回転対象の優先度（対象優先度）
 12882 ID preid 優先順位の回転対象とするプロセッサのID番号
 12883
 12884 【リターンパラメータ】
 12885 ER ercd 正常終了 (E_OK) またはエラーコード
 12886
 12887 【エラーコード】
 12888 E_CTX コンテキストエラー
 12889 ・ 非タスクコンテキストからの呼出し (mrot_rdqの場合)
 12890 【NGKI2693】
 12891 ・ タスクコンテキストからの呼出し (imrot_rdqの場合) 【NGKI2694】
 12892 ・ CPUロック状態からの呼出し【NGKI2695】
 12893 E_NOSPT 未サポート機能
 12894 ・ 条件については機能の項を参照
 12895 E_ID 不正ID番号
 12896 ・ preidが有効範囲外【NGKI2696】
 12897 E_PAR パラメータエラー
 12898 ・ tskpriが有効範囲外【NGKI2697】
 12899 E_OACV オブジェクトアクセス違反
 12900 ・ 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]

12901 【NGKI2698】
12902
12903 【機能】
12904
12905 prcidで指定したプロセッサに割り付けられており、tskpriで指定した優先度
12906 （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
12907 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】。
12908 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
12909 われずに正常終了する【NGKI2700】。
12910
12911 mrot_rdqにおいて、tskpriにTPRI_SELF (=0) を指定すると、自タスクのベー
12912 ス優先度が対象優先度となる【NGKI2701】。
12913
12914 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
12915 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、
12916 E_NOSPTエラーとなる【NGKI2702】。
12917
12918 【TOPPERS/ASPカーネルにおける規定】
12919
12920 ASPカーネルでは、mrot_rdq、imrot_rdqをサポートしない【ASPS0188】。
12921
12922 【TOPPERS/HRP2カーネルにおける規定】
12923
12924 HRP2カーネルでは、mrot_rdq、imrot_rdqをサポートしない【HRPS0152】。
12925
12926 【TOPPERS/SSPカーネルにおける規定】
12927
12928 SSPカーネルでは、mrot_rdq、imrot_rdqをサポートしない【SSPS0132】。
12929
12930 【μ ITRON4.0仕様との関係】
12931
12932 μ ITRON4.0仕様に定義されていないサービスコールである。
12933 -----
12934 get_tid 実行状態のタスクIDの参照 [T] 【NGKI2703】
12935 iget_tid 実行状態のタスクIDの参照 [I] 【NGKI2704】
12936
12937 【C言語API】
12938 ER ercd = get_tid(ID *p_tskid)
12939 ER ercd = iget_tid(ID *p_tskid)
12940
12941 【パラメータ】
12942 ID * p_tskid タスクIDを入れるメモリ領域へのポインタ
12943
12944 【リターンパラメータ】
12945 ER ercd 正常終了 (E_OK) またはエラーコード
12946 ID tskid タスクID
12947
12948 【エラーコード】
12949 E_CTX コンテキストエラー
12950 ・非タスクコンテキストからの呼出し (get_tidの場合) 【NGKI2705】

12951 ・タスクコンテキストからの呼出し (iget_tidの場合) 【NGKI2706】
12952 ・CPUロック状態からの呼出し 【NGKI2707】
12953 E_MACV メモリアクセス違反
12954 ・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
12955 いない [P] 【NGKI2708】
12956

12957 **【機能】**

12958
12959 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
12960 たタスクIDは, p_tskidで指定したメモリ領域に返される 【NGKI2709】.

12961
12962 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
12963 れる 【NGKI2710】.

12964
12965 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
12966 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
12967 【NGKI2711】.

12968
12969 **【TOPPERS/SSPカーネルにおける規定】**

12970
12971 SSPカーネルでは, get_tidをサポートしない 【SSPS0133】.

12972 -----
12973 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】
12974

12975 **【C言語API】**

12976 ER ercd = get_did(ID *p_domid)
12977

12978 **【パラメータ】**

12979 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
12980

12981 **【リターンパラメータ】**

12982 ER ercd 正常終了 (E_OK) またはエラーコード
12983 ID domid 保護ドメインID
12984

12985 **【エラーコード】**

12986 E_CTX コンテキストエラー
12987 ・非タスクコンテキストからの呼出し 【NGKI2713】
12988 ・CPUロック状態からの呼出し 【NGKI2714】
12989 E_MACV メモリアクセス違反
12990 ・p_domidが指すメモリ領域への書込みアクセスが許可されて
12991 いない 【NGKI2715】
12992

12993 **【機能】**

12994
12995 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
12996 照した保護ドメインIDは, p_domidで指定したメモリ領域に返される
12997 【NGKI2716】.

12998
12999 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
13000 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

13001 ンのID番号を参照する【NGKI2717】。
 13002
 13003 【TOPPERS/ASPカーネルにおける規定】
 13004
 13005 ASPカーネルでは、get_didをサポートしない【ASPS0189】。
 13006
 13007 【TOPPERS/FMPカーネルにおける規定】
 13008
 13009 FMPカーネルでは、get_didをサポートしない【FMPS0157】。
 13010
 13011 【TOPPERS/SSPカーネルにおける規定】
 13012
 13013 SSPカーネルでは、get_didをサポートしない【SSPS0134】。
 13014
 13015 get_pid 割付けプロセッサのID番号の参照 [TM] 【NGKI2718】
 13016 iget_pid 割付けプロセッサのID番号の参照 [IM] 【NGKI2719】
 13017
 13018 【C言語API】
 13019 ER ercd = get_pid(ID *p_prcid)
 13020 ER ercd = iget_pid(ID *p_prcid)
 13021
 13022 【パラメータ】
 13023 ID * p_prcid プロセッサIDを入れるメモリ領域へのポインタ
 13024
 13025 【リターンパラメータ】
 13026 ER ercd 正常終了 (E_OK) またはエラーコード
 13027 ID prcid プロセッサID
 13028
 13029 【エラーコード】
 13030 E_CTX コンテキストエラー
 13031 ・非タスクコンテキストからの呼出し (get_pidの場合) 【NGKI2720】
 13032 ・タスクコンテキストからの呼出し (iget_pidの場合) 【NGKI2721】
 13033 ・CPUロック状態からの呼出し 【NGKI2722】
 13034 E_MACV メモリアクセス違反
 13035 ・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
 13036 いない [P] 【NGKI2723】
 13037
 13038 【機能】
 13039
 13040 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。
 13041 参照したプロセッサIDは、p_prcidで指定したメモリ領域に返される
 13042 【NGKI2724】。
 13043
 13044 【使用上の注意】
 13045
 13046 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
 13047 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
 13048 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
 13049 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。
 13050

13051 【TOPPERS/ASPカーネルにおける規定】
 13052
 13053 ASPカーネルでは、get_pid, iget_pidをサポートしない【ASPS0190】。
 13054
 13055 【TOPPERS/HRP2カーネルにおける規定】
 13056
 13057 HRP2カーネルでは、get_pid, iget_pidをサポートしない【HRPS0153】。
 13058
 13059 【TOPPERS/SSPカーネルにおける規定】
 13060
 13061 SSPカーネルでは、get_pid, iget_pidをサポートしない【SSPS0135】。
 13062
 13063 【 μ ITRON4.0仕様との関係】
 13064
 13065 μ ITRON4.0仕様に定義されていないサービスコールである。
 13066

 13067 loc_cpu CPUロック状態への遷移 [T] 【NGKI2725】
 13068 iloc_cpu CPUロック状態への遷移 [I] 【NGKI2726】
 13069
 13070 【C言語API】
 13071 ER ercd = loc_cpu()
 13072 ER ercd = iloc_cpu()
 13073
 13074 【パラメータ】
 13075 なし
 13076
 13077 【リターンパラメータ】
 13078 ER ercd 正常終了 (E_OK) またはエラーコード
 13079
 13080 【エラーコード】
 13081 E_CTX コンテキストエラー
 13082 ・非タスクコンテキストからの呼出し (loc_cpuの場合) 【NGKI2727】
 13083 ・タスクコンテキストからの呼出し (iloc_cpuの場合) 【NGKI2728】
 13084 E_OACV オブジェクトアクセス違反
 13085 ・システム状態に対する通常操作2が許可されていない
 13086 (loc_cpuの場合) [P] 【NGKI2729】
 13087
 13088 【機能】
 13089
 13090 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
 13091 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】。
 13092

 13093 unl_cpu CPUロック状態の解除 [T] 【NGKI2732】
 13094 iunl_cpu CPUロック状態の解除 [I] 【NGKI2733】
 13095
 13096 【C言語API】
 13097 ER ercd = unl_cpu()
 13098 ER ercd = iunl_cpu()
 13099
 13100 【パラメータ】

13101 なし

13102

13103 **【リターンパラメータ】**

13104 ER ercd 正常終了 (E_OK) またはエラーコード

13105

13106 **【エラーコード】**

13107 E_CTX コンテキストエラー

13108 ・非タスクコンテキストからの呼出し (unl_cpuの場合) **【NGKI2734】**

13109 ・タスクコンテキストからの呼出し (iunl_cpuの場合) **【NGKI2735】**

13110 E_OACV オブジェクトアクセス違反

13111 ・システム状態に対する通常操作2が許可されていない

13112 (unl_cpuの場合) [P] **【NGKI2736】**

13113

13114 **【機能】**

13115

13116 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】** .

13117 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

13118 **【NGKI2738】** .

13119

13120 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

13121 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、

13122 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

13123 正常終了する) **【NGKI2739】** .

13124

13125 **【補足説明】**

13126

13127 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

13128 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

13129 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

13130 チンの実行が開始される可能性がある。

13131 -----

13132 dis_dsp ディスパッチの禁止 [T] **【NGKI2740】**

13133

13134 **【C言語API】**

13135 ER ercd = dis_dsp()

13136

13137 **【パラメータ】**

13138 なし

13139

13140 **【リターンパラメータ】**

13141 ER ercd 正常終了 (E_OK) またはエラーコード

13142

13143 **【エラーコード】**

13144 E_CTX コンテキストエラー

13145 ・非タスクコンテキストからの呼出し **【NGKI2741】**

13146 ・CPUロック状態からの呼出し **【NGKI2742】**

13147 E_OACV オブジェクトアクセス違反

13148 ・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]

13149 **【NGKI2743】**

13150

13151 **【機能】**

13152

13153 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

13154 **【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13155 正常終了する **【NGKI2745】**.

13156

13157 ena_dsp ディスパッチの許可 [T] **【NGKI2746】**

13158

13159 **【C言語API】**

13160 ER ercd = ena_dsp()

13161

13162 **【パラメータ】**

13163 なし

13164

13165 **【リターンパラメータ】**

13166 ER ercd 正常終了 (E_OK) またはエラーコード

13167

13168 **【エラーコード】**

13169 E_CTX コンテキストエラー

13170 ・非タスクコンテキストからの呼出し **【NGKI2747】**13171 ・CPUロック状態からの呼出し **【NGKI2748】**

13172 E_OACV オブジェクトアクセス違反

13173 ・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]

13174 **【NGKI2749】**

13175

13176 **【機能】**

13177

13178 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

13179 **【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13180 正常終了する **【NGKI2751】**.

13181

13182 **【補足説明】**

13183

13184 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

13185 スパッチが起こる可能性がある.

13186

13187 sns_ctx コンテキストの参照 [TI] **【NGKI2752】**

13188

13189 **【C言語API】**

13190 bool_t state = sns_ctx()

13191

13192 **【パラメータ】**

13193 なし

13194

13195 **【リターンパラメータ】**

13196 bool_t state コンテキスト

13197

13198 **【機能】**

13199

13200 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.

13201
13202 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
13203 ストから呼び出した場合にはfalseが返る【NGKI2753】.
13204 -----
13205 sns_loc CPUロック状態の参照 [TI] 【NGKI2754】
13206
13207 【C言語API】
13208 bool_t state = sns_loc()
13209
13210 【パラメータ】
13211 なし
13212
13213 【リターンパラメータ】
13214 bool_t state CPUロックフラグ
13215
13216 【機能】
13217
13218 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.
13219
13220 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
13221 出した場合にはfalseが返る【NGKI2755】.
13222 -----
13223 sns_dsp ディスパッチ禁止状態の参照 [TI] 【NGKI2756】
13224
13225 【C言語API】
13226 bool_t state = sns_dsp()
13227
13228 【パラメータ】
13229 なし
13230
13231 【リターンパラメータ】
13232 bool_t state ディスパッチ禁止フラグ
13233
13234 【機能】
13235
13236 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.
13237
13238 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
13239 状態で呼び出した場合にはfalseが返る【NGKI2757】.
13240 -----
13241 sns_dpn ディスパッチ保留状態の参照 [TI] 【NGKI2758】
13242
13243 【C言語API】
13244 bool_t state = sns_dpn()
13245
13246 【パラメータ】
13247 なし
13248
13249 【リターンパラメータ】
13250 bool_t state ディスパッチ保留状態

13251
13252 **【機能】**
13253
13254 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。
13255
13256 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
13257 状態でない状態で呼び出した場合にはfalseが返る【NGKI2759】。
13258 -----
13259 sns_ker カーネル非動作状態の参照 [TI] 【NGKI2760】
13260
13261 **【C言語API】**
13262 bool_t state = sns_ker()
13263
13264 **【パラメータ】**
13265 なし
13266
13267 **【リターンパラメータ】**
13268 bool_t state カーネル非動作状態
13269
13270 **【機能】**
13271
13272 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。
13273
13274 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
13275 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
13276 ネルの動作中に呼び出した場合にはfalseが返る【NGKI2761】。
13277
13278 **【使用方法】**
13279
13280 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
13281 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
13282 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
13283 合の動作は保証されない。
13284
13285 **【使用上の注意】**
13286
13287 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。
13288
13289 **【μ ITRON4.0仕様との関係】**
13290
13291 μ ITRON4.0仕様に定義されていないサービスコールである。
13292 -----
13293 ext_ker カーネルの終了 [TI] 【NGKI2762】
13294
13295 **【C言語API】**
13296 ER ercd = ext_ker()
13297
13298 **【パラメータ】**
13299 なし
13300

13301 **【リターンパラメータ】**
13302 ER ercd エラーコード
13303
13304 **【エラーコード】**
13305 E_SYS システムエラー
13306 ・カーネルの誤動作 **【NGKI2763】**
13307 E_OACV オブジェクトアクセス違反
13308 ・カーネルドメイン以外からの呼出し [P] **【NGKI2764】**
13309
13310 **【機能】**
13311
13312 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」
13313 の節を参照すること。
13314
13315 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】**。
13316
13317 **【 μ ITRON4.0仕様との関係】**
13318
13319 μ ITRON4.0仕様に定義されていないサービスコールである。
13320 -----
13321 ref_sys システムの状態参照 [T]
13322
13323 **【C言語API】**
13324 ER ercd = ref_sys(T_RSYS *pk_rsys)
13325
13326 ☆未完成
13327
13328 **【TOPPERS/ASPカーネルにおける規定】**
13329
13330 ASPカーネルでは、ref_sysをサポートしない。
13331
13332 **【TOPPERS/FMPカーネルにおける規定】**
13333
13334 FMPカーネルでは、ref_sysをサポートしない。
13335
13336 **【TOPPERS/HRP2カーネルにおける規定】**
13337
13338 HRP2カーネルでは、ref_sysをサポートしない。
13339
13340 **【TOPPERS/SSPカーネルにおける規定】**
13341
13342 SSPカーネルでは、ref_sysをサポートしない。
13343 -----
13344
13345 4.8 メモリオブジェクト管理機能
13346
13347 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
13348 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
13349 をサポートしない。
13350

13351 〔メモリリージョン〕

13352

13353 メモリリージョンは、オブジェクトモジュールに含まれるセクションの配置対
13354 象となる同じ性質を持った連続したメモリ領域である。メモリリージョンは、
13355 メモリリージョン名によって識別する【NGKI2766】。

13356

13357 各メモリリージョンが持つ情報は次の通り【NGKI2767】。

13358

13359 ・先頭番地

13360 ・サイズ

13361 ・メモリリージョン属性

13362

13363 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
13364 る場合がある【NGKI2768】。

13365

13366 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】。

13367

13368 TA_NOWRITE 0x01U 書込みアクセス禁止

13369

13370 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
13371 場合がある【NGKI2771】。

13372

13373 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
13374 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
13375 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
13376 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
13377 標準RAMリージョンが含まれる。

13378

13379 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
13380 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
13381 の標準メモリリージョンを定義することができる【NGKI3257】。

13382

13383 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
13384 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
13385 フィギュレータがエラーを報告する【NGKI3259】。

13386

13387 〔メモリオブジェクト〕

13388

13389 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
13390 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
13391 て識別する【NGKI2772】。

13392

13393 各メモリオブジェクトが持つ情報は次の通り【NGKI2773】。

13394

13395 ・先頭番地

13396 ・サイズ

13397 ・メモリオブジェクト属性

13398 ・アクセス許可ベクタ

13399 ・属する保護ドメイン

13400 ・属するクラス（マルチプロセッサ対応カーネルの場合）

13401
13402 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
13403 れる【NGKI2774】。
13404
13405 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】。
13406
13407 TA_NOWRITE 0x01U 書込みアクセス禁止
13408 TA_NOREAD 0x02U 読出しアクセス禁止
13409 TA_EXEC 0x04U 実行アクセス許可
13410 TA_MEMINI 0x08U メモリの初期化を行う
13411 TA_MEMPRSV 0x10U メモリの初期化を行わない
13412 TA_SDATA 0x20U ショートデータ領域に配置
13413 TA_UNCACHE 0x40U キャッシュ禁止
13414 TA_IODEV 0x80U 周辺デバイスの領域
13415
13416 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
13417 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
13418 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】。
13419 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
13420 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・
13421 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
13422 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
13423 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
13424 合である【NGKI2778】。
13425
13426 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
13427 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、
13428 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
13429 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
13430 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
13431 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
13432 の指定が無視されるかは、ターゲット定義である【NGKI2782】。
13433
13434 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
13435 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
13436 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
13437 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）
13438 される【NGKI2784】。TA_MEMINIとTA_MEMPRSVの両方を指定した場合には、
13439 E_RSATRエラーとなる【NGKI2785】。
13440
13441 TA_NOWRITEが指定されている場合には、TA_MEMINIとTA_MEMPRSVは無視される
13442 （指定しても指定しなくても、同じ振舞いとなる）【NGKI2786】。
13443
13444 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、
13445 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
13446 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】。
13447
13448 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
13449 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
13450 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

13451 【NGKI2789】. また、ターゲットによっては、TA_NOWRITEを指定した場合に、
13452 TA_SDATAが無視される場合がある【NGKI2790】.

13453

13454 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、
13455 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
13456 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
13457 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
13458 プロセッサでのTA_UNCACHE）では、これらの属性は無視される【NGKI2792】.
13459 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
13460 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
13461 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】.

13462

13463 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
13464 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、
13465 次の属性を予約している【NGKI2795】.

13466

13467 TA_WTHROUGH ライトスルーキャッシュを用いる

13468

13469 [カーネル構成マクロ]

13470

13471 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.

13472

13473 TOPPERS_SUPPORT_ATT_MOD ATT_MOD/ATA_MODがサポートされている
13474 【NGKI2796】

13475 TOPPERS_SUPPORT_ATT_PMA ATT_PMA/ATA_PMA/att_pmaがサポートさ
13476 れている【NGKI2797】

13477

13478 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
13479 め、サポートされているかを判定するには、TOPPERS_SUPPORT_DYNAMIC_CREと
13480 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
13481 る【NGKI2798】.

13482

13483 【補足説明】

13484

13485 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
13486 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.

13487

13488 【TOPPERS/ASPカーネルにおける規定】

13489

13490 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】.

13491

13492 【TOPPERS/FMPカーネルにおける規定】

13493

13494 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】.

13495

13496 【TOPPERS/HRP2カーネルにおける規定】

13497

13498 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】.

13499

13500 【TOPPERS/SSPカーネルにおける規定】

13501
 13502 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】。
 13503
 13504 【μITRON4.0/PX仕様との関係】
 13505
 13506 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
 13507 して廃止した。TA_ROはTA_NOWRITEに改名し、TA_NOREAD, TA_EXEC, TA_MEMINI,
 13508 TA_MEMPRSV, TA_IODEVを追加した。また、TA_UNCACHEの値を変更し、ターゲッ
 13509 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。
 13510
 13511 メモリリージョンは、μITRON4.0/PX仕様にはない概念である。
 13512
 13513 【仕様決定の理由】
 13514
 13515 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
 13516 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
 13517 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
 13518 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
 13519 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
 13520 るため、TA_IODEV属性を用意することにした。
 13521 -----
 13522 ATT_REG メモリリージョンの登録 [SP] 【NGKI2799】
 13523
 13524 【静的API】
 13525 ATT_REG("メモリリージョン名", { ATR regatr, void *base, SIZE size })
 13526
 13527 【パラメータ】
 13528 "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
 13529 ATR regatr メモリリージョン属性
 13530 void * base 登録するメモリリージョンの先頭番地
 13531 SIZE size 登録するメモリリージョンのサイズ (バイト数)
 13532
 13533 【エラーコード】
 13534 E_RSATR 予約属性
 13535 • regatrが無効【NGKI2800】
 13536 • 保護ドメインの囲みの中に記述されている【NGKI2814】
 13537 • クラスの囲みの中に記述されている [M] 【NGKI3260】
 13538 E_PAR パラメータエラー
 13539 • 条件については機能の項を参照
 13540 E_OBJ オブジェクト状態エラー
 13541 • 登録済みのメモリリージョンの再登録【NGKI2801】
 13542 • その他の条件については機能の項を参照
 13543
 13544 【機能】
 13545
 13546 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
 13547 リージョンを登録する。具体的な振舞いは以下の通り。
 13548
 13549 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
 13550 【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

13551 ジョン属性が設定される【NGKI2803】。

13552

13553 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

13554 ラメータである【NGKI2804】。

13555

13556 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

13557 た時には，E_PARエラーとなる【NGKI2815】。また，sizeが0の場合には，

13558 E_PARエラーとなる【NGKI2816】。登録しようとしたメモリリージョンが，登録

13559 済みのメモリリージョンとメモリ領域が重なる場合には，E_OBJエラーとなる

13560 【NGKI2817】。

13561

13562 【μ ITRON4.0/PX仕様との関係】

13563

13564 μ ITRON4.0/PX仕様に定義されていない静的APIである。

13565 -----

13566 DEF_SRG 標準メモリリージョンの定義 [SP] 【NGKI3261】

13567

13568 【静的API】

13569 DEF_SRG(“標準ROMリージョン名”，“標準RAMリージョン名”)

13570

13571 【パラメータ】

13572 “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

13573 指定する文字列

13574 “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

13575 指定する文字列

13576

13577 【エラーコード】

13578 E_RSATR 予約属性

13579 ・保護ドメインの囲みの中に記述されている【NGKI3262】

13580 E_OBJ オブジェクト状態エラー

13581 ・標準メモリリージョンが定義済み【NGKI3263】

13582 ・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

13583 【NGKI3264】

13584 ・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

13585 【NGKI3272】

13586 ・その他の条件については機能の項を参照

13587

13588 【機能】

13589

13590 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

13591 【NGKI3265】。

13592

13593 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，

13594 共通の標準ROMリージョンと標準のRAMリージョンを定義し，クラスの囲みの中

13595 に記述すると，そのクラスの標準ROMリージョンと標準RAMリージョンを定義す

13596 る【NGKI3266】。

13597

13598 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。

13599 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

13600 場合には，E_OBJエラーとなる【NGKI3268】。また，標準RAMリージョンは，

13601 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
 13602 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、
 13603 E_OBJエラーとなる【NGKI3270】
 13604
 13605 【μ ITRON4.0/PX仕様との関係】
 13606
 13607 μ ITRON4.0/PX仕様に定義されていない静的APIである。
 13608 -----
 13609 ATT_SEC セクションの登録 [SP] 【NGKI2818】
 13610 ATA_SEC セクションの登録（アクセス許可ベクタ付き） [SP] 【NGKI2819】
 13611
 13612 【静的API】
 13613 ATT_SEC("セクション名", { ATR mematr, "メモリリージョン名" })
 13614 ATA_SEC("セクション名", { ATR mematr, "メモリリージョン名" },
 13615 { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
 13616
 13617 【パラメータ】
 13618 "セクション名" 登録するセクションを指定する文字列
 13619 ATR mematr メモリオブジェクト属性
 13620 "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
 13621 する文字列
 13622
 13623 *アクセス許可ベクタ（パケットの内容）
 13624 ACPTN acptn1 通常操作1のアクセス許可パターン
 13625 ACPTN acptn2 通常操作2のアクセス許可パターン
 13626 ACPTN acptn3 管理操作のアクセス許可パターン
 13627 ACPTN acptn4 参照操作のアクセス許可パターン
 13628
 13629 【エラーコード】
 13630 E_RSATR 予約属性
 13631 ・mematrが無効【NGKI2820】
 13632 ・その他の条件については機能の項を参照
 13633 E_NOSPT 未サポート機能
 13634 ・条件については機能の項を参照
 13635 E_PAR パラメータエラー
 13636 ・条件については機能の項を参照
 13637 E_OBJ オブジェクト状態エラー
 13638 ・登録済みのセクションの再登録【NGKI2821】
 13639 ・指定したメモリリージョンが未登録【NGKI2822】
 13640
 13641 【機能】
 13642
 13643 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
 13644 する。具体的な振舞いは以下の通り。
 13645
 13646 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、
 13647 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
 13648 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、
 13649 mematrで指定したメモリオブジェクト属性が設定される【NGKI2824】。
 13650 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

13651 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
 13652 る【NGKI2825】.

13653

13654 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
 13655 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
 13656 も指定しなくても, 同じ振舞いとなる) 【NGKI2826】.

13657

13658 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
 13659 場合には, E_RSATRエラーとなる【NGKI2828】.

13660

13661 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
 13662 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
 13663 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】.

13664

13665 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～
 13666 acptn4は整数定数式パラメータである【NGKI2830】.

13667

13668 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
 13669 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
 13670 合には, E_NOSPTエラーとなる【NGKI2832】.

13671

13672 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,
 13673 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
 13674 となる【NGKI2834】.

13675

13676 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
 13677 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
 13678 として指定した場合には, E_PARエラーとなる【NGKI2836】.

13679

13680 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
 13681 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
 13682 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
 13683 る【NGKI2837】.

13684

13685 【μITRON4.0/PX仕様との関係】

13686

13687 μITRON4.0/PX仕様に定義されていない静的APIである.

13688 -----

13689 LNK_SEC セクションの配置 [SP] 【NGKI2838】

13690

13691 【静的API】

13692 LNK_SEC("セクション名", { "メモリリージョン名" })

13693

13694 【パラメータ】

13695 "セクション名" 配置するセクションを指定する文字列

13696 "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
 13697 する文字列

13698

13699 【エラーコード】

13700 E_RSATR 予約属性

275

13751 ・mematrが無効【NGKI2847】
13752 E_NOSPT 未サポート機能
13753 ・条件については機能の項を参照
13754 E_OBJ オブジェクト状態エラー
13755 ・登録済みのオブジェクトモジュールの再登録【NGKI2848】
13756
13757 **【機能】**
13758
13759 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
13760 ネルに登録する。具体的な振舞いは以下の通り。
13761
13762 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
13763 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
13764 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
13765 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
13766 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
13767 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
13768 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
13769 定した値に設定される【NGKI2851】。
13770
13771 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
13772 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】。
13773 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、
13774 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
13775 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
13776 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
13777 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、
13778 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
13779 リージョンに配置される場合がある【NGKI3271】。
13780
13781 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
13782 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
13783 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】。
13784
13785 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
13786 パラメータである【NGKI2856】。
13787
13788 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
13789 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
13790 に違反した場合には、E_NOSPTエラーとなる【NGKI2858】。
13791
13792 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
13793 【NGKI2859】。ATT_MOD／ATA_MODがサポートされている場合には、
13794 TOPPERS_SUPPORT_ATT_MODがマクロ定義される【NGKI2860】。サポートされてい
13795 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
13796 を報告する【NGKI2861】。
13797
13798 **【補足説明】**
13799
13800 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

13801 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
 13802 いる必要がある。

13803

13804 **【μITRON4.0/PX仕様との関係】**

13805

13806 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
 13807 ンと標準RAMリージョンであることを明確化した。

13808 -----

13809 ATT_MEM メモリオブジェクトの登録 [SP] **【NGKI2862】**

13810 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】**

13811 att_mem メモリオブジェクトの登録 [TPD] **【NGKI2864】**

13812

13813 **【静的API】**

13814 ATT_MEM({ ATR mematr, void *base, SIZE size })

13815 ATA_MEM({ ATR mematr, void *base, SIZE size },

13816 { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })

13817

13818 **【C言語API】**

13819 ER ercd = att_mem(const T_AMEM *pk_amem)

13820

13821 **【パラメータ】**

13822 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
 13823 トへのポインタ（静的APIを除く）

13824

13825 *メモリオブジェクトの登録情報（パケットの内容）

13826 ATR mematr メモリオブジェクト属性

13827 void * base 登録するメモリ領域の先頭番地

13828 SIZE size 登録するメモリ領域のサイズ（バイト数）

13829

13830 *アクセス許可ベクタ（パケットの内容）

13831 ACPTN acptn1 通常操作1のアクセス許可パターン

13832 ACPTN acptn2 通常操作2のアクセス許可パターン

13833 ACPTN acptn3 管理操作のアクセス許可パターン

13834 ACPTN acptn4 参照操作のアクセス許可パターン

13835

13836 **【リターンパラメータ】**

13837 ER ercd 正常終了（E_OK）またはエラーコード

13838

13839 **【エラーコード】**

13840 E_CTX コンテキストエラー

13841 ・非タスクコンテキストからの呼出し [s] **【NGKI2865】**

13842 ・CPUロック状態からの呼出し [s] **【NGKI2866】**

13843 E_RSATR 予約属性

13844 ・mematrが無効 **【NGKI2867】**

13845 ・属する保護ドメインの指定が有効範囲外 [sP] **【NGKI2868】**

13846 ・属するクラスの指定が有効範囲外 [sM] **【NGKI2869】**

13847 ・その他の条件については機能の項を参照

13848 E_NOSPT 未サポート機能

13849 ・条件については機能の項を参照

13850 E_PAR パラメータエラー

13851 ・条件については機能の項を参照
13852 E_OACV オブジェクトアクセス違反
13853 ・システム状態に対する管理操作が許可されていない [sP]
13854 【NGKI2870】
13855 E_MACV メモリアクセス違反
13856 ・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
13857 いない [sP] 【NGKI2871】
13858 E_OBJ オブジェクト状態エラー
13859 ・条件については機能の項を参照
13860
13861 【機能】
13862
13863 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
13864 クトを登録する。具体的な振舞いは以下の通り。
13865
13866 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
13867 【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
13868 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
13869 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）
13870 が、acptn1～acptn4で指定した値に設定される【NGKI2874】。
13871
13872 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
13873 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、
13874 E_RSATRエラーとなる【NGKI2876】。
13875
13876 静的APIにおいては、mematr、size、acptn1～acptn4は整数定数式パラメータ、
13877 baseは一般定数式パラメータである【NGKI2877】。
13878
13879 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
13880 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
13881 の制限に違反した場合には、E_NOSPTエラーとなる【NGKI2879】。
13882
13883 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
13884 た時には、E_PARエラーとなる【NGKI2880】。また、sizeが0の場合には、
13885 E_PARエラーとなる【NGKI2881】。登録しようとしたメモリオブジェクトが、登
13886 録済みのメモリオブジェクトとメモリ領域が重なる場合には、E_OBJエラーとな
13887 る【NGKI2882】。
13888
13889 【使用上の注意】
13890
13891 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
13892 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
13893 は、ATT_SEC／ATA_SECかATT_MOD／ATA_MODを使用することを推奨する。
13894
13895 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
13896 録したメモリリージョンと重なっても、直ちにエラーとはならない。ただし、
13897 メモリリージョン内に配置されたメモリオブジェクトと、ATT_MEM／ATA_MEMで
13898 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
13899 なる。
13900

13901 **【TOPPERS/HRP2カーネルにおける規定】**
 13902
 13903 HRP2カーネルでは、ATT_MEMとATA_MEMのみをサポートする【HRPS0155】。
 13904
 13905 **【μITRON4.0/PX仕様との関係】**
 13906
 13907 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
 13908 (ata_mem) は廃止した。
 13909
 13910 baseやsizeがターゲット定義の制約に合致しない場合、μITRON4.0/PX仕様では
 13911 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、
 13912 この仕様ではE_PARエラーとなることとした。
 13913 -----
 13914 ATT_PMA 物理メモリ領域の登録 [SP] 【NGKI2883】
 13915 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】
 13916 att_pma 物理メモリ領域の登録 [TPD] 【NGKI2885】
 13917
 13918 **【静的API】**
 13919 ATT_PMA({ ATR mematr, void *base, SIZE size, void *paddr })
 13920 ATA_PMA({ ATR mematr, void *base, SIZE size, void *paddr },
 13921 { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
 13922
 13923 **【C言語API】**
 13924 ER ercd = att_pma(const T_APMA *pk_apma)
 13925
 13926 **【パラメータ】**
 13927 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
 13928 のポインタ (静的APIを除く)
 13929
 13930 *物理メモリ領域の登録情報 (パケットの内容)
 13931 ATR mematr メモリオブジェクト属性
 13932 void * base 登録するメモリ領域の先頭番地
 13933 SIZE size 登録するメモリ領域のサイズ (バイト数)
 13934 void * paddr 登録するメモリ領域の物理アドレス空間における
 13935 先頭番地
 13936
 13937 *アクセス許可ベクタ (パケットの内容)
 13938 ACPTN acptn1 通常操作1のアクセス許可パターン
 13939 ACPTN acptn2 通常操作2のアクセス許可パターン
 13940 ACPTN acptn3 管理操作のアクセス許可パターン
 13941 ACPTN acptn4 参照操作のアクセス許可パターン
 13942
 13943 **【リターンパラメータ】**
 13944 ER ercd 正常終了 (E_OK) またはエラーコード
 13945
 13946 **【エラーコード】**
 13947 E_CTX コンテキストエラー
 13948 ・非タスクコンテキストからの呼出し [s] 【NGKI2886】
 13949 ・CPUロック状態からの呼出し [s] 【NGKI2887】
 13950 E_RSATR 予約属性

13951 • mematrが無効
13952 • 属する保護ドメインの指定が有効範囲外 [sP] 【NGKI2888】
13953 • 属するクラスの指定が有効範囲外 [sM] 【NGKI2889】
13954 • その他の条件については機能の項を参照
13955 E_NOSPT 未サポート機能
13956 • 条件については機能の項を参照
13957 E_PAR パラメータエラー
13958 • 条件については機能の項を参照
13959 E_OACV オブジェクトアクセス違反
13960 • システム状態に対する管理操作が許可されていない [sP]
13961 【NGKI2890】
13962 E_MACV メモリアクセス違反
13963 • 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
13964 いない [sP] 【NGKI2891】
13965 E_OBJ オブジェクト状態エラー
13966 • 条件については機能の項を参照
13967
13968 【機能】
13969
13970 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
13971 クトを登録する。具体的な振舞いは以下の通り。
13972
13973 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
13974 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
13975 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
13976 クトには、mematrで指定したメモリオブジェクト属性が設定される【NGKI2893】。
13977 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
13978 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
13979 る【NGKI2894】。
13980
13981 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
13982 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、
13983 E_RSATRエラーとなる【NGKI2896】。
13984
13985 静的APIにおいては、mematr、size、paddr、acptn1～acptn4は整数定数式パラ
13986 メータ、baseは一般定数式パラメータである【NGKI2897】。
13987
13988 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
13989 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
13990 の制限に違反した場合には、E_NOSPTエラーとなる【NGKI2899】。
13991
13992 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
13993 指定した時には、E_PARエラーとなる【NGKI2900】。また、sizeが0の場合には、
13994 E_PARエラーとなる【NGKI2901】。登録しようとしたメモリオブジェクトが、登
13995 録済みのメモリオブジェクトと論理アドレス空間においてメモリ領域が重なる
13996 場合には、E_OBJエラーとなる【NGKI2902】。
13997
13998 ATT_PMA／ATA_PMA／att_pmaは、MMU (Memory Management Unit) を持つターゲッ
13999 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】。
14000 ATT_PMA／ATA_PMA／att_pmaがサポートされている場合には、

14001 TOPPERS_SUPPORT_ATT_PMAがマクロ定義される【NGKI2904】。ATT_PMA/ATA_PMA
 14002 がサポートされていない場合にこれらの静的APIを使用すると、コンフィギュレー
 14003 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
 14004 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
 14005 となる【NGKI2906】。

14006

14007 【TOPPERS/HRP2カーネルにおける規定】

14008

14009 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

14010

【HRPS0156】。

14011

14012 【μITRON4.0/PX仕様との関係】

14013

14014 μITRON4.0/PX仕様に定義されていない静的APIおよびサービスコールである。

14015

14016 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】

14017

14018 【C言語API】

14019

ER ercd = sac_mem(const void *base, const ACVCT *p_acvct)

14020

14021 【パラメータ】

14022

void * base メモリオブジェクトの先頭番地

14023

ACVCT * p_acvct アクセス許可ベクタを入れたパケットへのポ

14024

インタ

14025

14026 *アクセス許可ベクタ (パケットの内容)

14027

ACPTN acptn1 通常操作1のアクセス許可パターン

14028

ACPTN acptn2 通常操作2のアクセス許可パターン

14029

ACPTN acptn3 管理操作のアクセス許可パターン

14030

ACPTN acptn4 参照操作のアクセス許可パターン

14031

14032 【リターンパラメータ】

14033

ER ercd 正常終了 (E_OK) またはエラーコード

14034

14035 【エラーコード】

14036

E_CTX コンテキストエラー

14037

・非タスクコンテキストからの呼出し【NGKI2908】

14038

・CPUロック状態からの呼出し【NGKI2909】

14039

E_PAR パラメータエラー

14040

・baseがメモリオブジェクトの先頭番地でない【NGKI2910】

14041

E_NOEXS オブジェクト未登録

14042

・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

14043

ていない【NGKI2911】

14044

E_OACV オブジェクトアクセス違反

14045

・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

14046

ない【NGKI2912】

14047

E_MACV メモリアクセス違反

14048

・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

14049

いない【NGKI2913】

14050

E_OBJ オブジェクト状態エラー

14051 ・対象メモリオブジェクトは静的APIで登録された【NGKI2914】

14052

14053 【機能】

14054

14055 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
14056 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
14057 設定する【NGKI2915】。

14058

14059 【TOPPERS/HRP2カーネルにおける規定】

14060

14061 HRP2カーネルでは、sac_memをサポートしない【HRPS0157】。

14062

14063 【μITRON4.0/PX仕様との関係】

14064

14065 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
14066 ることができないこととした。

14067

14068 μITRON4.0/PX仕様では、baseはメモリオブジェクトに含まれる番地を指定する
14069 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
14070 ならないものとした。

14071

14072 det_mem メモリオブジェクトの登録解除 [TPD] 【NGKI2916】

14073

14074 【C言語API】

14075 ER ercd = det_mem(const void *base)

14076

14077 【パラメータ】

14078 void * base メモリオブジェクトの先頭番地

14079

14080 【リターンパラメータ】

14081 ER ercd 正常終了 (E_OK) またはエラーコード

14082

14083 【エラーコード】

14084 E_CTX コンテキストエラー

14085 ・非タスクコンテキストからの呼出し【NGKI2917】

14086 ・CPUロック状態からの呼出し【NGKI2918】

14087 E_PAR パラメータエラー

14088 ・baseがメモリオブジェクトの先頭番地でない【NGKI2919】

14089 E_NOEXS オブジェクト未登録

14090 ・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
14091 ていない【NGKI2920】

14092 E_OACV オブジェクトアクセス違反

14093 ・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
14094 ない【NGKI2921】

14095 E_OBJ オブジェクト状態エラー

14096 ・対象メモリオブジェクトは静的APIで登録された【NGKI2922】

14097

14098 【機能】

14099

14100 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

14101 る【NGKI2923】.

14102

14103 【TOPPERS/HRP2カーネルにおける規定】

14104

14105 HRP2カーネルでは、det_memをサポートしない【HRPS0158】.

14106

14107 【μITRON4.0/PX仕様との関係】

14108

14109 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
14110 いこととした.

14111

14112 μITRON4.0/PX仕様では、baseはメモリオブジェクトに含まれる番地を指定する
14113 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
14114 ならないものとした.

14115

14116 prb_mem メモリ領域に対するアクセス権のチェック [TP] 【NGKI2924】

14117

14118 【C言語API】

14119 ER ercd = prb_mem(const void *base, SIZE size, ID tskid, MODE pmmode)

14120

14121 【パラメータ】

14122	void *	base	メモリ領域の先頭番地
14123	SIZE	size	メモリ領域のサイズ (バイト数)
14124	ID	tskid	アクセス元のタスクのID番号
14125	MODE	pmmode	アクセスモード

14126

14127 【リターンパラメータ】

14128	ER	ercd	正常終了 (E_OK) またはエラーコード
-------	----	------	-----------------------

14129

14130 【エラーコード】

14131	E_CTX	コンテキストエラー
14132		・非タスクコンテキストからの呼出し【NGKI2925】
14133	E_ID	不正ID番号
14134		・tskidが有効範囲外【NGKI2927】
14135	E_PAR	パラメータエラー
14136		・sizeが0【NGKI2929】
14137		・その他の条件については機能の項を参照
14138	E_NOEXS	オブジェクト未登録
14139		・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
14140		ていない【NGKI2930】
14141	E_OACV	オブジェクトアクセス違反
14142		・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
14143		作が許可されていない【NGKI2931】
14144	E_MACV	メモリアクセス違反
14145		・条件については機能の項を参照
14146	E_OBJ	オブジェクト状態エラー
14147		・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
14148		【NGKI2932】

14149

14150 【機能】

14151
14152 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
14153 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
14154 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
14155 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、
14156 E_MACVが返ることはない【NGKI2934】．
14157
14158 pmmodeには、TPM_WRITE（=0x01U），TPM_READ（=0x02U），TPM_EXEC（=
14159 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
14160 “|”）を指定することができる【NGKI2935】．TPM_WRITE，TPM_READ，TPM_EXEC
14161 を指定した場合には、それぞれ、読出しアクセス、書込みアクセス、実行アク
14162 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
14163 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
14164 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
14165 た場合には、E_PARエラーとなる【NGKI2938】．
14166
14167 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
14168 アクセスが許可されているかをチェックする【NGKI2939】．
14169
14170 【μITRON4.0/PX仕様との関係】
14171
14172 アクセスする主体の指定方法を、保護ドメインによる指定（domid）から、タス
14173 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
14174 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
14175 らも呼び出せるものとした．
14176
14177 【仕様決定の理由】
14178
14179 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．
14180 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
14181 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
14182 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、
14183 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
14184 呼び出せることが必要である．
14185
14186 なお、prb_memを非タスクコンテキストから呼び出すことはできないが、非タス
14187 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、
14188 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．
14189 -----
14190 ref_mem メモリオブジェクトの状態参照 [TP]
14191
14192 【C言語API】
14193 ER ercd = ref_mem(const void *base, T_RMEM *pk_rmem)
14194
14195 ☆未完成
14196
14197 【TOPPERS/HRP2カーネルにおける規定】
14198
14199 HRP2カーネルでは、ref_memをサポートしない．
14200 -----

14201

14202 4.9 割込み管理機能

14203

14204 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
 14205 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、
 14206 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
 14207 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
 14208 みハンドラを経由して呼び出される【NGKI2941】．

14209

14210 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
 14211 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

14212 【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
 14213 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
 14214 ンの属性を設定しておく必要がある【NGKI2943】．

14215

14216 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、
 14217 次の属性を指定することができる【NGKI2944】．

14218

14219	TA_ENAINT	0x01U	割込み要求禁止フラグをクリア
14220	TA_EDGE	0x02U	エッジトリガ

14221

14222 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
 14223 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
 14224 の属性を予約している【NGKI2946】．

14225

14226	TA_POSEDGE	ポジティブエッジトリガ
14227	TA_NEGEDGE	ネガティブエッジトリガ
14228	TA_BOTHEDGE	両エッジトリガ
14229	TA_LOWLEVEL	ローレベルトリガ
14230	TA_HIGHLEVEL	ハイレベルトリガ
14231	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 14232 サ対応カーネルの場合）

14233

14234 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
 14235 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
 14236 する【NGKI2947】．

14237

14238 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、
 14239 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
 14240 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
 14241 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
 14242 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
 14243 される【NGKI2949】．

14244

14245 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
 14246 る保護ドメインは、カーネルドメインに限られる【NGKI2950】．

14247

14248 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
 14249 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
 14250 【NGKI2952】．

14251
14252 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】.

```
14253  
14254     void interrupt_service_routine(intptr_t exinf)  
14255     {  
14256         割込みサービスルーチン本体  
14257     }
```

14258
14259 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】.

14260
14261 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
14262 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
14263 【NGKI2955】.

14264
14265 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
14266 る【NGKI2956】.

14267
14268 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
14269 定義で、次の属性を指定することができる【NGKI2957】.

14270
14271 TA_NONKERNEL 0x02U カーネル管理外の割込み
14272

14273 TA_NONKERNELを指定しない場合、カーネル管理の割込みとなる【NGKI2958】.
14274 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
14275 を指定できる場合がある【NGKI2959】.

14276
14277 C言語による割込みハンドラの記述形式は次の通り【NGKI2960】.

```
14278  
14279     void interrupt_handler(void)  
14280     {  
14281         割込みハンドラ本体  
14282     }
```

14283
14284 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.

14285
14286 TMIN_INTPRI 割込み優先度の最小値（最高値） 【NGKI2961】
14287 TMAX_INTPRI 割込み優先度の最大値（最低値，=-1）
14288
14289 TMIN_ISRPRI 割込みサービスルーチン優先度の最小値（=1）【NGKI2962】
14290 TMAX_ISRPRI 割込みサービスルーチン優先度の最大値
14291
14292 TOPPERS_SUPPORT_DIS_INT dis_intがサポートされている【NGKI2963】
14293 TOPPERS_SUPPORT_ENA_INT ena_intがサポートされている【NGKI2964】
14294

14295 **【使用上の注意】**
14296

14297 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
14298 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
14299 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
14300 ずにリターンするように実装すべきである.

14301
 14302 **【TOPPERS/ASPカーネルにおける規定】**
 14303
 14304 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）
 14305 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
 14306 は、TMAX_ISRPRIを256に拡張する【ASPS0193】。
 14307
 14308 **【TOPPERS/FMPカーネルにおける規定】**
 14309
 14310 FMPカーネルでは、割込みサービスルーチン優先度の最大値（=TMAX_ISRPRI）
 14311 は16に固定されている【FMPS0159】。
 14312
 14313 **【TOPPERS/HRP2カーネルにおける規定】**
 14314
 14315 HRP2カーネルでは、割込みサービスルーチン優先度の最大値（=TMAX_ISRPRI）
 14316 は16に固定されている【HRPS0159】。
 14317
 14318 **【TOPPERS/SSPカーネルにおける規定】**
 14319
 14320 SSPカーネルでは、割込みサービスルーチン優先度の最大値（=TMAX_ISRPRI）
 14321 は16に固定されている【SSPS0137】。
 14322
 14323 **【 μ ITRON4.0仕様との関係】**
 14324
 14325 割込み要求ラインの属性、割込み優先度、割込みサービスルーチン優先度は、
 14326 μ ITRON4.0仕様がない概念であり、TMIN_INTPRI, TMAX_INTPRI, TMIN_ISRPRI,
 14327 TMAX_ISRPRIは、 μ ITRON4.0仕様に定義のないカーネル構成マクロである。また、
 14328 TA_NONKERNELは、 μ ITRON4.0仕様に定義のない割込みハンドラ属性である。
 14329

 14330 CFG_INT 割込み要求ラインの属性の設定 [S] 【NGKI2965】
 14331 cfg_int 割込み要求ラインの属性の設定 [TD] 【NGKI2966】
 14332
 14333 **【静的API】**
 14334 CFG_INT(INTNO intno, { ATR intatr, PRI intpri })
 14335
 14336 **【C言語API】**
 14337 ER ercd = cfg_int(INTNO intno, const T_CINT *pk_cint)
 14338
 14339 **【パラメータ】**
 14340 INTNO intno 割込み番号
 14341 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
 14342 ケットへのポインタ（静的APIを除く）
 14343
 14344 *割込み要求ラインの属性の設定情報（パケットの内容）
 14345 ATR intatr 割込み要求ライン属性
 14346 PRI intpri 割込み優先度
 14347
 14348 **【リターンパラメータ】**
 14349 ER ercd 正常終了（E_OK）またはエラーコード
 14350

14351 **【エラーコード】**

- 14352 E_CTX コンテキストエラー
- 14353 ・非タスクコンテキストからの呼出し [s] **【NGKI2967】**
- 14354 ・CPUロック状態からの呼出し [s] **【NGKI2968】**
- 14355 E_RSATR 予約属性
- 14356 ・intatrが無効 **【NGKI2969】**
- 14357 ・属するクラスの指定が有効範囲外 [sM] **【NGKI2970】**
- 14358 ・クラスの囲みの中に記述されていない [SM] **【NGKI2971】**
- 14359 ・その他の条件については機能の項を参照
- 14360 E_PAR パラメータエラー
- 14361 ・intnoが有効範囲外 **【NGKI2972】**
- 14362 ・intpriが有効範囲外 **【NGKI2973】**
- 14363 ・その他の条件については機能の項を参照
- 14364 E_OACV オブジェクトアクセス違反
- 14365 ・システム状態に対する管理操作が許可されていない [sP]
- 14366 **【NGKI2974】**
- 14367 E_MACV メモリアクセス違反
- 14368 ・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
- 14369 いない [sP] **【NGKI2975】**
- 14370 E_OBJ オブジェクト状態エラー
- 14371 ・対象割込み要求ラインに対して属性が設定済み [S] **【NGKI2976】**
- 14372 ・その他の条件については機能の項を参照

14373

14374 **【機能】**

14375

14376 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

14377 メータで指定した属性を設定する **【NGKI2977】** .

14378

14379 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

14380 た場合にクリアされ、指定しない場合にセットされる **【NGKI2978】** .

14381

14382 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

14383 **【NGKI2979】** .

14384

14385 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

14386 が連動して設定される場合がある **【NGKI2980】** .

14387

14388 intpriに指定できる値は、基本的には、TMIN_INTPRI以上、TMAX_INTPRI以下の

14389 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

14390 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

14391 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

14392 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

14393 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

14394 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

14395 も、E_OBJエラーとなる **【NGKI2984】** .

14396

14397 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

14398 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

14399 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

14400 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

14401 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、
14402 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
14403 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】。
14404
14405 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
14406 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
14407 【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
14408 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
14409 E_RSATRエラーとなる【NGKI2990】。ただし、TA_DOM(TDOM_SELF)を指定した場
14410 合には、指定が無視され、E_RSATRエラーは検出されない【NGKI2991】。
14411
14412 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
14413 対して登録された割込みサービスルーチン（または対象割込み番号に対応する
14414 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
14415 み中にある場合には、E_RSATRエラーとなる【NGKI2992】。
14416
14417 【補足説明】
14418
14419 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
14420 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
14421 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、
14422 変更できないケースが考えられる。
14423
14424 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
14425 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
14426 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
14427 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
14428 か設定できない割込み要求ラインに対して異なる割込み優先度を設定した場合
14429 には、E_PARエラーとなる。
14430
14431 【TOPPERS/ASPカーネルにおける規定】
14432
14433 ASPカーネルでは、CFG_INTのみをサポートする【ASPS0194】。
14434
14435 【TOPPERS/FMPカーネルにおける規定】
14436
14437 FMPカーネルでは、CFG_INTのみをサポートする【FMPS0160】。
14438
14439 【TOPPERS/HRP2カーネルにおける規定】
14440
14441 HRP2カーネルでは、CFG_INTのみをサポートする【HRPS0160】。
14442
14443 【TOPPERS/SSPカーネルにおける規定】
14444
14445 SSPカーネルでは、CFG_INTのみをサポートする【SSPS0138】。
14446
14447 【μITRON4.0仕様との関係】
14448
14449 μITRON4.0仕様に定義されていない静的APIおよびサービスコールである。
14450 -----

14451	CRE_ISR	割込みサービスルーチンの生成 [S]	【NGKI2993】
14452	ATT_ISR	割込みサービスルーチンの追加 [S]	【NGKI2994】
14453	acre_isr	割込みサービスルーチンの生成 [TD]	【NGKI2995】
14454			
14455		【静的API】	
14456	CRE_ISR	(ID isrid, { ATR isratr, intptr_t exinf,	
14457		INTNO intno, ISR isr, PRI isrpri })	
14458	ATT_ISR	({ ATR isratr, intptr_t exinf, INTNO intno, ISR isr, PRI isrpri })	
14459			
14460		【C言語API】	
14461	ER_ID isrid	= acre_isr(const T_CISR *pk_cisr)	
14462			
14463		【パラメータ】	
14464	ID	isrid	対象割込みサービスルーチンのID番号 (CRE_ISR
14465			の場合)
14466	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
14467			ケットへのポインタ (静的APIを除く)
14468			
14469		* 割込みサービスルーチンの生成情報 (パケットの内容)	
14470	ATR	isratr	割込みサービスルーチン属性
14471	intptr_t	exinf	割込みサービスルーチンの拡張情報
14472	INTNO	intno	割込みサービスルーチンを登録する割込み番号
14473	ISR	isr	割込みサービスルーチンの先頭番地
14474	PRI	isrpri	割込みサービスルーチン優先度
14475			
14476		【リターンパラメータ】	
14477	ER_ID	isrid	生成された割込みサービスルーチンのID番号 (正
14478			の値) またはエラーコード
14479			
14480		【エラーコード】	
14481	E_CTX	コンテキストエラー	
14482		・ 非タスクコンテキストからの呼出し [s] 【NGKI2996】	
14483		・ CPUロック状態からの呼出し [s] 【NGKI2997】	
14484	E_RSATR	予約属性	
14485		・ isratrが無効 【NGKI2998】	
14486		・ 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	
14487		メイン以外 [sP] 【NGKI2999】	
14488		・ カーネルドメインの囲みの中に記述されていない [SP]	
14489		【NGKI3000】	
14490		・ 属するクラスの指定が有効範囲外 [sM] 【NGKI3001】	
14491		・ クラスの囲みの中に記述されていない [SM] 【NGKI3002】	
14492		・ その他の条件については機能の項を参照	
14493	E_PAR	パラメータエラー	
14494		・ intnoが有効範囲外 【NGKI3003】	
14495		・ isrがプログラムの先頭番地として正しくない 【NGKI3004】	
14496		・ isrpriが有効範囲外 【NGKI3005】	
14497	E_OACV	オブジェクトアクセス違反	
14498		・ システム状態に対する管理操作が許可されていない [sP]	
14499		【NGKI3006】	
14500	E_MACV	メモリアccess違反	

14501 ・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
14502 いない [sP] 【NGKI3007】
14503 E_NOID ID番号不足
14504 ・割り付けられる割込みサービスルーチンIDがない [sD]
14505 【NGKI3008】
14506 E_OBJ オブジェクト状態エラー
14507 ・isridで指定した割込みサービスルーチンが登録済み
14508 (CRE_ISRの場合) 【NGKI3009】
14509 ・その他の条件については機能の項を参照
14510
14511 【機能】
14512
14513 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
14514 ビスルーチンを生成する 【NGKI3010】。
14515
14516 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
14517 【NGKI3011】。
14518
14519 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
14520 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
14521 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)
14522 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
14523 【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、
14524 E_OBJエラーとなる 【NGKI3014】。
14525
14526 静的APIにおいては、isridはオブジェクト識別名、isratr, intno, isrpriは整
14527 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】。
14528
14529 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
14530 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
14531 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
14532 【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
14533 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
14534 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、
14535 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
14536 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
14537 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
14538 【NGKI3019】。
14539
14540 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、
14541 ターゲット定義である 【NGKI3020】。
14542
14543 【TOPPERS/ASPカーネルにおける規定】
14544
14545 ASPカーネルでは、ATT_ISRのみをサポートする。ただし、動的生成機能拡張パッ
14546 ケージでは、acre_isrもサポートする 【ASPS0195】。
14547
14548 【TOPPERS/FMPカーネルにおける規定】
14549
14550 FMPカーネルでは、ATT_ISRのみをサポートする 【FMPS0161】。

14551
14552 **【TOPPERS/HRP2カーネルにおける規定】**
14553
14554 HRP2カーネルでは、ATT_ISRのみをサポートする【HRPS0161】。
14555
14556 **【TOPPERS/SSPカーネルにおける規定】**
14557
14558 SSPカーネルでは、ATT_ISRのみをサポートする【SSPS0139】。
14559
14560 **【 μ ITRON4.0仕様との関係】**
14561
14562 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
14563 込み優先度）を追加した。CRE_ISRは、 μ ITRON4.0仕様に定義されていない静的
14564 APIである。
14565 -----
14566 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】
14567
14568 **【静的API】**
14569 AID_ISR(uint_t noisr)
14570
14571 **【パラメータ】**
14572 uint_t noisr 割付け可能な割込みサービスルーチンIDの数
14573
14574 **【エラーコード】**
14575 E_RSATR 予約属性
14576 ・クラスの囲みの中に記述されていない〔M〕【NGKI3022】
14577 ・カーネルドメインの囲みの中に記述されていない〔P〕【NGKI3023】
14578
14579 **【機能】**
14580
14581 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
14582 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
14583 て確保する【NGKI3024】。
14584
14585 noisrは整数定数式パラメータである【NGKI3025】。
14586 -----
14587 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】
14588 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】
14589
14590 **【静的API】**
14591 SAC_ISR(ID isrid, { ACPTN acptn1, ACPTN acptn2,
14592 ACPTN acptn3, ACPTN acptn4 })
14593
14594 **【C言語API】**
14595 ER ercd = sac_isr(ID isrid, const ACVCT *p_acvct)
14596
14597
14598 **【パラメータ】**
14599 ID isrid 対象割込みサービスルーチンのID番号
14600 ACVCT * p_acvct アクセス許可ベクタを入れたパケットへのポ

14601

14602

14603

14604

14605

14606

14607

14608

14609

14610

14611

14612

14613

14614

14615

14616

14617

14618

14619

14620

14621

14622

14623

14624

14625

14626

14627

14628

14629

14630

14631

14632

14633

14634

14635

14636

14637

14638

14639

14640

14641

14642

14643

14644

14645

14646

14647

14648

14649

14650

インタ（静的APIを除く）

* アクセス許可ベクタ（パケットの内容）

ACPTN acptn1 通常操作1のアクセス許可パターン

ACPTN acptn2 通常操作2のアクセス許可パターン

ACPTN acptn3 管理操作のアクセス許可パターン

ACPTN acptn4 参照操作のアクセス許可パターン

【リターンパラメータ】

ER ercd 正常終了（E_OK）またはエラーコード

【エラーコード】

E_CTX コンテキストエラー

・ 非タスクコンテキストからの呼出し [s] 【NGKI3028】

・ CPUロック状態からの呼出し [s] 【NGKI3029】

E_ID 不正ID番号

・ isridが有効範囲外 [s] 【NGKI3030】

E_RSATR 予約属性

・ 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] 【NGKI3031】

・ 対象割込みサービスルーチンが属するクラスの囲みの中に記述されていない [SM] 【NGKI3032】

E_NOEXS オブジェクト未登録

・ 対象割込みサービスルーチンが未登録 【NGKI3033】

E_OACV オブジェクトアクセス違反

・ 対象割込みサービスルーチンに対する管理操作が許可されていない [s] 【NGKI3034】

E_MACV メモリアクセス違反

・ p_acvctが指すメモリ領域への読出しアクセスが許可されていない [s] 【NGKI3035】

E_OBJ オブジェクト状態エラー

・ 対象割込みサービスルーチンは静的APIで生成された [s] 【NGKI3036】

・ 対象割込みサービスルーチンに対してアクセス許可ベクタが設定済み [S] 【NGKI3037】

【機能】

isridで指定した割込みサービスルーチン（対象割込みサービスルーチン）のアクセス許可ベクタ（4つのアクセス許可パターンの組）を、各パラメータで指定した値に設定する 【NGKI3038】。

静的APIにおいては、isridはオブジェクト識別名、acptn1～acptn4は整数定数式パラメータである 【NGKI3039】。

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、SAC_ISR, sac_isrをサポートしない 【ASPS0196】。

【TOPPERS/FMPカーネルにおける規定】

14651
14652 FMPカーネルでは、SAC_ISR, sac_isrをサポートしない【FMPS0162】。
14653
14654 【TOPPERS/HRP2カーネルにおける規定】
14655
14656 HRP2カーネルでは、SAC_ISR, sac_isrをサポートしない【HRPS0162】。
14657
14658 【TOPPERS/SSPカーネルにおける規定】
14659
14660 SSPカーネルでは、SAC_ISR, sac_isrをサポートしない【SSPS0140】。
14661
14662 【未決定事項】
14663
14664 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
14665 セス許可ベクタでアクセス保護する方法も考えられる。
14666 -----
14667 del_isr 割込みサービスルーチンの削除 [TD] 【NGKI3040】
14668
14669 【C言語API】
14670 ER ercd = del_isr(ID isrid)
14671
14672 【パラメータ】
14673 ID isrid 対象割込みサービスルーチンのID番号
14674
14675 【リターンパラメータ】
14676 ER ercd 正常終了 (E_OK) またはエラーコード
14677
14678 【エラーコード】
14679 E_CTX コンテキストエラー
14680 ・非タスクコンテキストからの呼出し【NGKI3041】
14681 ・CPUロック状態からの呼出し【NGKI3042】
14682 E_ID 不正ID番号
14683 ・isridが有効範囲外【NGKI3043】
14684 E_NOEXS オブジェクト未登録
14685 ・対象割込みサービスルーチンが未登録【NGKI3044】
14686 E_OACV オブジェクトアクセス違反
14687 ・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
14688 ていない [P] 【NGKI3045】
14689 E_OBJ オブジェクト状態エラー
14690 ・対象割込みサービスルーチンは静的APIで生成された【NGKI3046】
14691
14692 【機能】
14693
14694 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
14695 除する。具体的な振舞いは以下の通り。
14696
14697 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
14698 IDが未使用の状態に戻される【NGKI3047】。
14699
14700 【TOPPERS/ASPカーネルにおける規定】

14701
14702 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
14703 機能拡張パッケージでは、del_isrをサポートする【ASPS0198】。
14704
14705 【TOPPERS/FMPカーネルにおける規定】
14706
14707 FMPカーネルでは、del_isrをサポートしない【FMPS0163】。
14708
14709 【TOPPERS/HRP2カーネルにおける規定】
14710
14711 HRP2カーネルでは、del_isrをサポートしない【HRPS0163】。
14712
14713 【TOPPERS/SSPカーネルにおける規定】
14714
14715 SSPカーネルでは、del_isrをサポートしない【SSPS0141】。
14716 -----
14717 ref_isr 割込みサービスルーチンの状態参照 [T]
14718
14719 【C言語API】
14720 ER ercd = ref_isr(ID isrid, T_RISR *pk_risr)
14721
14722 ☆未完成
14723
14724 【TOPPERS/ASPカーネルにおける規定】
14725
14726 ASPカーネルでは、ref_isrをサポートしない。
14727
14728 【TOPPERS/FMPカーネルにおける規定】
14729
14730 FMPカーネルでは、ref_isrをサポートしない。
14731
14732 【TOPPERS/HRP2カーネルにおける規定】
14733
14734 HRP2カーネルでは、ref_isrをサポートしない。
14735
14736 【TOPPERS/SSPカーネルにおける規定】
14737
14738 SSPカーネルでは、ref_isrをサポートしない。
14739 -----
14740 DEF_INH 割込みハンドラの定義 [S] 【NGKI3048】
14741 def_inh 割込みハンドラの定義 [TD] 【NGKI3049】
14742
14743 【静的API】
14744 DEF_INH(INHNO inhno, { ATR inhatr, INTHDR inthdr })
14745
14746 【C言語API】
14747 ER ercd = def_inh(INHNO inhno, const T_DINH *pk_dinh)
14748
14749 【パラメータ】
14750 INHNO inhno 割込みハンドラ番号

14751	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）
14752			
14753			
14754	* 割込みハンドラの定義情報（パケットの内容）		
14755	ATR	inhatr	割込みハンドラ属性
14756	INTHDR	inthdr	割込みハンドラの先頭番地
14757			
14758	【リターンパラメータ】		
14759	ER	ercd	正常終了（E_OK）またはエラーコード
14760			
14761	【エラーコード】		
14762	E_CTX	コンテキストエラー	
14763		・ 非タスクコンテキストからの呼出し [s] 【NGKI3050】	
14764		・ CPUロック状態からの呼出し [s] 【NGKI3051】	
14765	E_RSATR	予約属性	
14766		・ inhatrが無効【NGKI3052】	
14767		・ 属するクラスの指定が有効範囲外 [sM] 【NGKI3053】	
14768		・ クラスの囲みの中に記述されていない [SM] 【NGKI3054】	
14769		・ その他の条件については機能の項を参照	
14770	E_PAR	パラメータエラー	
14771		・ inhnoが有効範囲外【NGKI3055】	
14772		・ inthdrがプログラムの先頭番地として正しくない【NGKI3056】	
14773		・ その他の条件については機能の項を参照	
14774	E_OACV	オブジェクトアクセス違反	
14775		・ システム状態に対する管理操作が許可されていない [sP]	
14776		【NGKI3057】	
14777	E_MACV	メモリアクセス違反	
14778		・ pk_dinhが指すメモリ領域への読出しアクセスが許可されていない [sP] 【NGKI3058】	
14779			
14780	E_OBJ	オブジェクト状態エラー	
14781		・ 条件については機能の項を参照	
14782			
14783	【機能】		
14784			
14785	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		
14786	パラメータで指定した割込みハンドラ定義情報に従って、割込みハンドラを定義する【NGKI3059】。ただし、def_inhにおいてpk_dinhをNULLにした場合には、		
14787	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】。		
14788			
14789			
14790	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		
14791	数式パラメータである【NGKI3061】。		
14792			
14793	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		
14794	pk_dinhをNULL以外にした場合）には、次のエラーが検出される。		
14795			
14796	対象割込みハンドラ番号に対応する割込み要求ラインの属性が設定されていない場合には、E_OBJエラーとなる【NGKI3062】。また、対象割込みハンドラ番号		
14797	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		
14798	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		
14799	場合にも、E_OBJエラーとなる【NGKI3063】。		
14800			

14801
14802 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
14803 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
14804 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
14805 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
14806 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
14807 【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
14808 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
14809 `TA_NONKERNEL`を指定しない場合には、`E_RSATR`エラーとなる【NGKI3067】。逆に、
14810 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
14811 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
14812 場合には、`E_RSATR`エラーとなる【NGKI3068】。
14813
14814 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
14815 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
14816 【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
14817 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
14818 `TA_DOM(domid)`を指定した場合には`E_RSATR`エラーとなる【NGKI3071】。ただし、
14819 `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
14820 出されない【NGKI3072】。
14821
14822 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
14823 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、
14824 `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
14825 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
14826 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
14827 も、`E_RSATR`エラーとなる【NGKI3075】。
14828
14829 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
14830 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
14831 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
14832 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
14833 は、ターゲット定義で`E_OBJ`エラーとなる場合がある【NGKI3077】。
14834
14835 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
14836 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
14837 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
14838 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
14839 に該当する【NGKI3079】。
14840
14841 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
14842 は、ターゲット定義である【NGKI3080】。
14843
14844 【TOPPERS/ASPカーネルにおける規定】
14845
14846 ASPカーネルでは、`DEF_INH`のみをサポートする【ASPS0199】。
14847
14848 【TOPPERS/FMPカーネルにおける規定】
14849
14850 FMPカーネルでは、`DEF_INH`のみをサポートする【FMPS0164】。

14851
14852 **【TOPPERS/HRP2カーネルにおける規定】**
14853
14854 HRP2カーネルでは、DEF_INHのみをサポートする【HRPS0164】。
14855
14856 **【TOPPERS/SSPカーネルにおける規定】**
14857
14858 SSPカーネルでは、DEF_INHのみをサポートする【SSPS0142】。
14859
14860 **【μITRON4.0仕様との関係】**
14861
14862 inthdrのデータ型をINTHDRに変更した。
14863
14864 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、
14865 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
14866 義を解除してから、再度定義する必要がある。
14867

14868 dis_int 割込みの禁止 [T] **【NGKI3081】**
14869
14870 **【C言語API】**
14871 ER ercd = dis_int(INTNO intno)
14872
14873 **【パラメータ】**
14874 INTNO intno 割込み番号
14875
14876 **【リターンパラメータ】**
14877 ER ercd 正常終了 (E_OK) またはエラーコード
14878
14879 **【エラーコード】**
14880 E_CTX コンテキストエラー
14881 ・非タスクコンテキストからの呼出し【NGKI3082】
14882 E_NOSPT 未サポートエラー
14883 ・条件については機能の項を参照
14884 E_PAR パラメータエラー
14885 ・intnoが有効範囲外【NGKI3083】
14886 ・その他の条件については機能の項を参照
14887 E_OACV オブジェクトアクセス違反
14888 ・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]
14889 **【NGKI3084】**
14890 E_OBJ オブジェクト状態エラー
14891 ・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
14892 定されていない【NGKI3085】
14893
14894 **【機能】**
14895
14896 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
14897 フラグをセットする【NGKI3086】。
14898
14899 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
14900 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

14901 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
 14902 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
 14903 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
 14904 作できない場合が、これに該当する【NGKI3088】。

14905
 14906 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
 14907 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
 14908 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
 14909 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【NGKI3090】。

14910
 14911 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】。
 14912 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
 14913 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、
 14914 E_NOSPTエラーが返るか、リンク時にエラーとなる【NGKI3093】。

14915
 14916 【μITRON4.0仕様との関係】

14917
 14918 μITRON4.0仕様で実装定義としていたintnoの意味を標準化した。

14919
 14920 CPUロック状態でも呼び出せるものとした。

14921 -----
 14922 ena_int 割込みの許可 [T] 【NGKI3094】

14923
 14924 【C言語API】
 14925 ER ercd = ena_int(INTNO intno)

14926
 14927 【パラメータ】
 14928 INTNO intno 割込み番号

14929
 14930 【リターンパラメータ】
 14931 ER ercd 正常終了 (E_OK) またはエラーコード

14932
 14933 【エラーコード】
 14934 E_CTX コンテキストエラー
 14935 ・非タスクコンテキストからの呼出し【NGKI3095】
 14936 E_NOSPT 未サポートエラー
 14937 ・条件については機能の項を参照
 14938 E_PAR パラメータエラー
 14939 ・intnoが有効範囲外【NGKI3096】
 14940 ・その他の条件については機能の項を参照
 14941 E_OACV オブジェクトアクセス違反
 14942 ・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]
 14943 【NGKI3097】
 14944 E_OBJ オブジェクト状態エラー
 14945 ・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
 14946 定されていない【NGKI3098】

14947
 14948 【機能】

14949
 14950 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

14951 フラグをクリアする【NGKI3099】.

14952

14953 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
14954 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
14955 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
14956 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
14957 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
14958 作できない場合が、これに該当する【NGKI3101】.

14959

14960 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
14961 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
14962 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
14963 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【NGKI3103】.

14964

14965 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】.
14966 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
14967 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、
14968 E_NOSPTエラーが返るか、リンク時にエラーとなる【NGKI3106】.

14969

14970 【 μ ITRON4.0仕様との関係】

14971

14972 μ ITRON4.0仕様で実装定義としていたintnoの意味を標準化した.

14973

14974 CPUロック状態でも呼び出せるものとした.

14975 -----

14976 ref_int 割込み要求ラインの参照 [T]

14977

14978 【C言語API】

14979 ER ercd = ref_int(INTNO intno, T_RINT *pk_rint)

14980

14981 ☆未完成

14982

14983 【TOPPERS/ASPカーネルにおける規定】

14984

14985 ASPカーネルでは、ref_intをサポートしない.

14986

14987 【TOPPERS/FMPカーネルにおける規定】

14988

14989 FMPカーネルでは、ref_intをサポートしない.

14990

14991 【TOPPERS/HRP2カーネルにおける規定】

14992

14993 HRP2カーネルでは、ref_intをサポートしない.

14994

14995 【TOPPERS/SSPカーネルにおける規定】

14996

14997 SSPカーネルでは、ref_intをサポートしない.

14998

14999 【 μ ITRON4.0仕様との関係】

15000

15001 μ ITRON4.0仕様に定義されていないサービスコールである.

15002 -----

15003 chg_ipm 割込み優先度マスクの変更 [T] 【NGKI3107】

15004

15005 【C言語API】

15006 ER ercd = chg_ipm(PRI intpri)

15007

15008 【パラメータ】

15009 PRI intpri 割込み優先度マスク

15010

15011 【リターンパラメータ】

15012 ER ercd 正常終了 (E_OK) またはエラーコード

15013

15014 【エラーコード】

15015 E_CTX コンテキストエラー

15016 ・非タスクコンテキストからの呼出し 【NGKI3108】

15017 ・CPUロック状態からの呼出し 【NGKI3109】

15018 E_PAR パラメータエラー

15019 ・条件については機能の項を参照

15020 E_OACV オブジェクトアクセス違反

15021 ・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]

15022 【NGKI3110】

15023

15024 【機能】

15025

15026 割込み優先度マスクを、intpriで指定した値に変更する 【NGKI3111】.

15027

15028 intpriは、TMIN_INTPRI以上、TIPM_ENAALL以下でなければならない. そうでな
15029 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
15030 として、TMIN_INTPRIよりも小さい値を指定できる場合がある 【NGKI3114】.

15031

15032 【補足説明】

15033

15034 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
15035 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
15036 の実行が開始される可能性がある.

15037

15038 【TOPPERS/SSPカーネルにおける規定】

15039

15040 SSPカーネルでは、chg_ipmをサポートしない 【SSPS0143】.

15041

15042 【 μ ITRON4.0仕様との関係】

15043

15044 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
15045 義となっているサービスコールである.

15046 -----

15047 get_ipm 割込み優先度マスクの参照 [T] 【NGKI3115】

15048

15049 【C言語API】

15050 ER ercd = get_ipm(PRI *p_intpri)

15051

15052 **【パラメータ】**

15053 PRI * p_intpri 割込み優先度マスクを入れるメモリ領域へのポ
15054 インタ

15055

15056 **【リターンパラメータ】**

15057 ER ercd エラーコード

15058 PRI intpri 割込み優先度マスク

15059

15060 **【エラーコード】**

15061 E_CTX コンテキストエラー

15062 ・非タスクコンテキストからの呼出し【NGKI3116】

15063 ・CPUロック状態からの呼出し【NGKI3117】

15064 E_OACV オブジェクトアクセス違反

15065 ・システム状態に対する参照操作が許可されていない [P]

15066 【NGKI3118】

15067 E_MACV メモリアクセス違反

15068 ・p_intpriが指すメモリ領域への書込みアクセスが許可され
15069 ていない [P] 【NGKI3119】

15070

15071 **【機能】**

15072

15073 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、
15074 p_intpriで指定したメモリ領域に返される【NGKI3120】．

15075

15076 **【TOPPERS/SSPカーネルにおける規定】**

15077

15078 SSPカーネルでは、get_ipmをサポートしない【SSPS0144】．

15079

15080 **【μ ITRON4.0仕様との関係】**

15081

15082 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
15083 義となっているサービスコールである．

15084 -----

15085

15086 4.10 CPU例外管理機能

15087

15088 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
15089 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
15090 【NGKI3121】．

15091

15092 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
15093 る【NGKI3122】．

15094

15095 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、
15096 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】．
15097 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
15098 【NGKI3124】．

15099

15100 TA_DIRECT CPU例外ハンドラを直接呼び出す

```

15101
15102 C言語によるCPU例外ハンドラの記述形式は次の通り【NGKI3125】.
15103
15104     void cpu_exception_handler(void *p_excinf)
15105     {
15106         CPU例外ハンドラ本体
15107     }
15108
15109 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
15110 【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
15111 る際に必要となる.
15112 -----
15113 DEF_EXC      CPU例外ハンドラの定義 [S] 【NGKI3127】
15114 def_exc      CPU例外ハンドラの定義 [TD] 【NGKI3128】
15115
15116 【静的API】
15117     DEF_EXC(EXCNO excno, { ATR excatr, EXCHDR exchdr })
15118
15119 【C言語API】
15120     ER ercd = def_exc(EXCNO excno, const T_DEXC *pk_dexc)
15121
15122 【パラメータ】
15123     EXCNO      excno      CPU例外ハンドラ番号
15124     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
15125                        のポインタ（静的APIを除く）
15126
15127     *CPU例外ハンドラの定義情報（パケットの内容）
15128     ATR        excatr     CPU例外ハンドラ属性
15129     EXCHDR     exchdr     CPU例外ハンドラの先頭番地
15130
15131 【リターンパラメータ】
15132     ER        ercd        正常終了 (E_OK) またはエラーコード
15133
15134 【エラーコード】
15135     E_CTX      コンテキストエラー
15136                ・非タスクコンテキストからの呼出し [s] 【NGKI3129】
15137                ・CPUロック状態からの呼出し [s] 【NGKI3130】
15138     E_RSATR    予約属性
15139                ・excatrが無効【NGKI3131】
15140                ・属するクラスの指定が有効範囲外 [sM] 【NGKI3132】
15141                ・クラスの囲みの中に記述されていない [SM] 【NGKI3133】
15142                ・その他の条件については機能の項を参照
15143     E_PAR      パラメータエラー
15144                ・excnoが有効範囲外【NGKI3134】
15145                ・exchdrがプログラムの先頭番地として正しくない【NGKI3135】
15146     E_OACV     オブジェクトアクセス違反
15147                ・システム状態に対する管理操作が許可されていない [sP]
15148                【NGKI3136】
15149     E_MACV     メモリアクセス違反
15150                ・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

```

15151 いない [sP] 【NGKI3137】
15152 E_OBJ オブジェクト状態エラー
15153 ・条件については機能の項を参照
15154
15155 【機能】
15156
15157 excnoで指定したCPU例外ハンドラ番号（対象CPU例外ハンドラ番号）に対して、
15158 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
15159 を定義する【NGKI3138】。ただし、def_excにおいてpk_dexcをNULLにした場合
15160 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
15161 【NGKI3139】。
15162
15163 静的APIにおいては、excnoとexcptrは整数定数式パラメータ、exchdrは一般定
15164 数式パラメータである【NGKI3140】。
15165
15166 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
15167 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
15168 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】。
15169
15170 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
15171 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
15172 【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
15173 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
15174 TA_DOM(domid)を指定した場合にはE_RSATRエラーとなる【NGKI3144】。ただし、
15175 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
15176 出されない【NGKI3145】。
15177
15178 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
15179 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、
15180 E_RSATRエラーとなる【NGKI3146】。
15181
15182 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
15183 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
15184 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
15185 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
15186 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】。
15187
15188 静的APIにおいて、exchdrが不正である場合にE_PARエラーが検出されるか否か
15189 は、ターゲット定義である【NGKI3149】。
15190
15191 【TOPPERS/ASPカーネルにおける規定】
15192
15193 ASPカーネルでは、DEF_EXCのみをサポートする【ASPS0200】。
15194
15195 【TOPPERS/FMPカーネルにおける規定】
15196
15197 FMPカーネルでは、DEF_EXCのみをサポートする【FMPS0165】。
15198
15199 【TOPPERS/HRP2カーネルにおける規定】
15200

15201 HRP2カーネルでは、DEF_EXCのみをサポートする【HRPS0165】。

15202

15203 【TOPPERS/SSPカーネルにおける規定】

15204

15205 SSPカーネルでは、DEF_EXCのみをサポートする【SSPS0145】。

15206

15207 【 μ ITRON4.0仕様との関係】

15208

15209 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、
15210 E_OBJエラーとすることにした。

15211

15212 xsns_dpn CPU例外発生時のディスパッチ保留状態の参照 [TI] 【NGKI3150】

15213

15214 【C言語API】

15215 bool_t stat = xsns_dpn(void *p_excinf)

15216

15217 【パラメータ】

15218 void * p_excinf CPU例外の情報を記憶しているメモリ領域の先頭
15219 番地

15220

15221 【リターンパラメータ】

15222 bool_t state ディスパッチ保留状態

15223

15224 【機能】

15225

15226 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
15227 り。

15228

15229 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
15230 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
15231 状態でなかった場合にfalse、そうでない場合にtrueが返る【NGKI3151】。

15232

15233 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
15234 した場合には、trueが返る【NGKI3152】。

15235

15236 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
15237 【NGKI3153】。

15238

15239 【使用方法】

15240

15241 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
15242 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
15243 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
15244 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、
15245 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
15246 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。

15247

15248 【使用上の注意】

15249

15250 xsns_dpnは、E_CTXエラーを返すことがないために [TI] となっているが、CPU

例外ハンドラから呼び出すためのものである。CPU例外ハンドラ以外から呼び出した場合や、`p_excinf`に正しい値を渡さなかった場合、`xsns_dpn`が返す値は意味を持たない。

どちらの条件で`true`が返るか間違いやすいので注意すること。

【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、`xsns_dpn`をサポートしない【SSPS0146】。

【 μ ITRON4.0仕様との関係】

μ ITRON4.0仕様に定義されていないサービスコールである。

【仕様決定の理由】

保護機能対応のカーネルにおいては、`xsns_dpn`をユーザドメインから呼び出すことは禁止すべきである。ユーザドメインの実行中は、必ずタスクコンテキストであるため、`xsns_dpn`をタスクコンテキストから呼び出した場合に必ず`true`を返す仕様とすることで、`xsns_dpn`をユーザドメインから呼び出すことを実質的に禁止している。

`xsns_xpn` CPU例外発生時のタスク例外処理保留状態の参照 [TI] 【NGKI3154】

【C言語API】

```
bool_t stat = xsns_xpn(void *p_excinf)
```

【パラメータ】

<code>void *</code>	<code>p_excinf</code>	CPU例外の情報を記憶しているメモリ領域の先頭番地
---------------------	-----------------------	---------------------------

【リターンパラメータ】

<code>bool_t</code>	<code>state</code>	タスク例外処理保留状態
---------------------	--------------------	-------------

【機能】

CPU例外発生時にタスク例外処理ルーチンを実行開始できない状態であったかを参照する。具体的な振舞いは以下の通り。

実行中のCPU例外ハンドラの起動原因となったCPU例外が、カーネル管理外のCPU例外でなく、タスクコンテキストで発生し、そのタスクがタスク例外処理ルーチンを実行開始できる状態であった場合に`false`、そうでない場合に`true`が返る【NGKI3155】。

保護機能対応カーネルにおいて、CPU例外が発生したタスクがユーザタスクの場合には、ユーザスタック領域の残りが少なく、タスク例外処理ルーチンを実行開始できない（タスク例外処理ルーチンを実行開始しようとする、タスク例外実行開始時スタック不正例外が発生する）場合にも、`true`を返す【NGKI3156】。

保護機能対応のカーネルにおいて、`xsns_xpn`をタスクコンテキストから呼び出

15301 した場合には、trueが返る【NGKI3157】。

15302

15303 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
15304 【NGKI3158】。

15305

15306 【使用方法】

15307

15308 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
15309 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
15310 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
15311 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
15312 カバリ処理を行うことができる。

15313

15314 【使用上の注意】

15315

15316 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
15317 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
15318 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
15319 味を持たない。

15320

15321 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。

15322

15323 【TOPPERS/SSPカーネルにおける規定】

15324

15325 SSPカーネルでは、xsns_xpnをサポートしない【SSPS0147】。

15326

15327 【μITRON4.0仕様との関係】

15328

15329 μITRON4.0仕様に定義されていないサービスコールである。

15330

15331 【仕様決定の理由】

15332

15333 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
15334 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
15335 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
15336 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
15337 的に禁止している。

15338

15339

15340 4.11 拡張サービスコール管理機能

15341

15342 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
15343 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
15344 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、
15345 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】。

15346

15347 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
15348 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
15349 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
15350 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

15351 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、
15352 すべての種別のアクセスを行うことが許可される。

15353
15354 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
15355 め、拡張サービスコール管理機能をサポートしない【NGKI3163】。

15356
15357 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】。

```
15358  
15359     ER_UINT extended_svc(intptr_t par1, intptr_t par2, intptr_t par3,  
15360                           intptr_t par4, intptr_t par5, ID cdmid)  
15361     {  
15362         拡張サービスコール本体  
15363     }
```

15364
15365 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
15366 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
15367 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)
15368 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
15369 される。

15370
15371 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
15372 【NGKI3166】。

15373
15374 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。

15375
15376 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
15377 カーネルでは、登録できる拡張サービスコールの数に一
15378 致)【NGKI3167】

15379
15380 【TOPPERS/ASPカーネルにおける規定】

15381
15382 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】。

15383
15384 【TOPPERS/FMPカーネルにおける規定】

15385
15386 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】。

15387
15388 【TOPPERS/HRP2カーネルにおける規定】

15389
15390 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】。

15391
15392 【TOPPERS/SSPカーネルにおける規定】

15393
15394 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】。

15395
15396 【未決定事項】

15397
15398 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
15399 は未決定である。

15400

15401 【 μ ITRON4.0仕様との関係】

15402

15403 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
15404 メータ5個に固定した。

15405

15406 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
15407 番号を渡す機能を追加した。

15408

15409 TMAX_FNCND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。
15410 -----

15411 DEF_SVC 拡張サービスコールの定義 [SP] 【NGKI3168】

15412 def_svc 拡張サービスコールの定義 [TPD] 【NGKI3169】

15413

15414 【静的API】

15415 DEF_SVC(FN fncd, { ATR svcatr, EXTSVC extsvc, SIZE stksz })

15416

15417 【C言語API】

15418 ER ercd = def_svc(FN fncd, const T_DSVC *pk_dsvc)

15419

15420 【パラメータ】

15421 FN fncd 拡張サービスコールの機能コード

15422 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
15423 トへのポインタ（静的APIを除く）

15424

15425

15426 *拡張サービスコールの定義情報（パケットの内容）

15427 ATR svcatr 拡張サービスコール属性

15428 EXTSVC extsvc 拡張サービスコールの先頭番地

15429 SIZE stksz 拡張サービスコールで使用するスタックサイズ

15429

15430 【リターンパラメータ】

15431 ER ercd 正常終了 (E_OK) またはエラーコード

15432

15433 【エラーコード】

15434 E_CTX コンテキストエラー

15435 ・非タスクコンテキストからの呼出し [s] 【NGKI3170】

15436 ・CPUロック状態からの呼出し [s] 【NGKI3171】

15437 E_RSATR 予約属性

15438 ・svcatrが無効 【NGKI3172】

15439 ・その他の条件については機能の項を参照

15440 E_PAR パラメータエラー

15441 ・fncdが0または負の値 【NGKI3173】

15442 ・fncdがTMAX_FNCNDよりも大きい 【NGKI3174】

15443 ・extsvcがプログラムの先頭番地として正しくない 【NGKI3175】

15444 E_OACV オブジェクトアクセス違反

15445 ・システム状態に対する管理操作が許可されていない [s]

15446 【NGKI3176】

15447 E_MACV メモリアクセス違反

15448 ・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

15449 いない [s] 【NGKI3177】

15450 E_OBJ オブジェクト状態エラー

15451 ・条件については機能の項を参照
15452
15453 **【機能】**
15454
15455 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
15456 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
15457 **【NGKI3178】**．ただし、def_svcにおいてpk_dsvcをNULLにした場合には、対象
15458 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】**．
15459
15460 静的APIにおいては、fncd, svcatr, stkszは整数定数式パラメータ、svchdrは
15461 一般定数式パラメータである **【NGKI3180】**．
15462
15463 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
15464 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
15465 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】**．
15466
15467 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
15468 い場合には、E_RSATRエラーとなる **【NGKI3183】**．また、def_svcで拡張サービ
15469 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
15470 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
15471 はE_RSATRエラーとなる **【NGKI3184】**．ただし、TA_DOM(TDOM_SELF)を指定した
15472 場合には、指定が無視され、E_RSATRエラーは検出されない **【NGKI3185】**．
15473
15474 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
15475 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
15476 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
15477 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
15478 TA_CLS(clsid)を指定した場合にはE_RSATRエラーとなる **【NGKI3188】**．ただし、
15479 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
15480 出されない **【NGKI3189】**．
15481
15482 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
15483 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
15484 い場合には、E_OBJエラーとなる **【NGKI3190】**．
15485
15486 **【TOPPERS/HRP2カーネルにおける規定】**
15487
15488 HRP2カーネルでは、DEF_SVCのみをサポートする **【HRPS0167】**．
15489
15490 **【μITRON4.0仕様との関係】**
15491
15492 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
15493 クサイズ）を追加した．
15494
15495 extsvcのデータ型を、EXTSVCに変更した．
15496 -----
15497 cal_svc 拡張サービスコールの呼出し [TIP] **【NGKI3191】**
15498
15499 **【C言語API】**
15500 ER_UINT ercd = cal_svc(FN fncd, intptr_t par1, intptr_t par2,

15501 intptr_t par3, intptr_t par4, intptr_t par5)

15502

15503 **【パラメータ】**

15504	FN	fncd	呼び出す拡張サービスコールの機能コード
15505	intptr_t	par1	拡張サービスコールへの第1パラメータ
15506	intptr_t	par2	拡張サービスコールへの第2パラメータ
15507	intptr_t	par3	拡張サービスコールへの第3パラメータ
15508	intptr_t	par4	拡張サービスコールへの第4パラメータ
15509	intptr_t	par5	拡張サービスコールへの第5パラメータ

15510

15511 **【リターンパラメータ】**

15512	ER_UINT	ered	正常終了（正の値または0）またはエラーコード
-------	---------	------	------------------------

15513

15514 **【エラーコード】**

15515	E_SYS	システムエラー
15516		・条件については機能の項を参照
15517	E_RSFN	予約機能コード
15518		・fncdが0または負の値【NGKI3192】
15519		・fncdがTMAX_FNCDよりも大きい【NGKI3193】
15520		・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
15521		定義されていない【NGKI3194】
15522	E_NOMEM	メモリ不足
15523		・条件については機能の項を参照

15524 *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。

15525

15526 **【機能】**

15527

15528 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

15529 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】。

15530

15531 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

15532 【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

15533 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、

15534 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

15535 が上限（=255）を超える場合には、E_SYSエラーが返る【NGKI3198】。

15536

15537 **【μITRON4.0仕様との関係】**

15538

15539 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

15540 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

15541 出せないこととした。

15542

15543 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

15544 クする機能を追加した。

15545

15546 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

15547 し、cal_svcから返るエラー（E_SYS, E_RSFN, E_NOMEM）について規定した。

15548

15549 **【仕様決定の理由】**

15550

15551 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
15552 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
15553 うにするためである。
15554 -----
15555
15556 4.12 システム構成管理機能
15557
15558 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
15559 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
15560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。
15561
15562 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
15563 処理単位が用いるスタック領域である。
15564
15565 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、
15566 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】。
15567
15568 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
15569 始の直前に、カーネル非動作状態で実行される【NGKI3200】。
15570
15571 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
15572 る【NGKI3201】。
15573
15574 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
15575 チン属性には、TA_NULLを指定しなければならない【NGKI3203】。
15576
15577 C言語による初期化ルーチンの記述形式は次の通り【NGKI3204】。
15578
15579 void initialization_routine(intptr_t exinf)
15580 {
15581 初期化ルーチン本体
15582 }
15583
15584 exinfには、初期化ルーチンの拡張情報が渡される【NGKI3205】。
15585
15586 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
15587 終了の直後に、カーネル非動作状態で実行される【NGKI3206】。
15588
15589 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
15590 する【NGKI3207】。
15591
15592 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
15593 理ルーチン属性には、TA_NULLを指定しなければならない【NGKI3209】。
15594
15595 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】。
15596
15597 void termination_routine(intptr_t exinf)
15598 {
15599 終了処理ルーチン本体
15600 }

15601
15602 exinfには、終了処理ルーチンの拡張情報が渡される【NGKI3211】。
15603
15604 【 μ ITRON4.0仕様との関係】
15605
15606 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、
15607 μ ITRON4.0仕様に規定されていない機能である。
15608 -----
15609 DEF_ICS 非タスクコンテキスト用スタック領域の設定 [S] 【NGKI3212】
15610
15611 【静的API】
15612 DEF_ICS({ SIZE istksz, STK_T *istk })
15613
15614 【パラメータ】
15615 *非タスクコンテキスト用スタック領域の設定情報
15616 SIZE istksz 非タスクコンテキスト用スタック領域のサイズ
15617 (バイト数)
15618 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
15619
15620 【エラーコード】
15621 E_RSATR 予約属性
15622 ・カーネルドメインの囲みの中に記述されていない [P] 【NGKI3213】
15623 ・クラスの囲みの中に記述されていない [M] 【NGKI3214】
15624 E_PAR パラメータエラー
15625 ・条件については機能の項を参照
15626 E_NOMEM メモリ不足
15627 ・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】
15628 E_OBJ オブジェクト状態エラー
15629 ・非タスクコンテキスト用スタック領域が設定済み【NGKI3216】
15630 ・その他の条件については機能の項を参照
15631
15632 【機能】
15633
15634 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
15635 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
15636 0を指定した時や、ターゲット定義の最小値よりも小さい値を指定した時には、
15637 E_PARエラーとなる【NGKI3254】。
15638
15639 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
15640 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
15641 ない【NGKI3218】。
15642
15643 istkをNULLとした場合、istkszで指定したサイズのスタック領域を、コンフィ
15644 ギュレータが確保する【NGKI3219】。istkszにターゲット定義の制約に合致し
15645 ないサイズを指定した時には、ターゲット定義の制約に合致するようにサイズ
15646 を大きい方に丸めて確保する【NGKI3220】。
15647
15648 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
15649 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
15650 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

15651 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
 15652 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
 15653 【NGKI3222】。

15654

15655 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
 15656 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、
 15657 E_OBJエラーとなる【NGKI3223】。

15658

15659 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
 15660 ト定義のデフォルトのサイズのスタック領域を、コンフィギュレータが確保す
 15661 る【NGKI3224】。

15662

15663 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
 15664 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
 15665 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
 15666 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
 15667 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、
 15668 E_OBJエラーとなる【NGKI3227】。

15669

15670 【TOPPERS/SSPカーネルにおける規定】

15671

15672 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
 15673 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
 15674 【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
 15675 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
 15676 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
 15677 領域のサイズを指定することのみとなる。istkにNULL以外を指定した場合には、
 15678 E_PARエラーとなる【SSPS0150】。

15679

15680 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。

15681

15682 【μITRON4.0仕様との関係】

15683

15684 μITRON4.0仕様に定義されていない静的APIである。

15685 -----

15686 DEF_STK 共有スタック領域の設定 [S] 【NGKI3228】

15687

15688 【静的API】

15689 DEF_STK({ SIZE stksz, STK_T *stk })

15690

15691 【パラメータ】

15692 * 共有スタック領域の設定情報

15693 SIZE stksz 共有スタック領域のサイズ (バイト数)

15694 STK_T stk 共有スタック領域の先頭番地

15695

15696 【エラーコード】

15697 E_PAR パラメータエラー

15698 ・条件については機能の項を参照

15699 E_NOMEM メモリ不足

15700 ・共有スタック領域が確保できない【NGKI3229】

15701 E_OBJ オブジェクト状態エラー
15702 ・共有スタック領域が設定済み
15703
15704 【サポートするカーネル】
15705
15706 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
15707 ネルは、DEF_STKをサポートしない【NGKI3230】。
15708
15709 【機能】
15710
15711 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
15712 領域を設定する【NGKI3231】。stkszに0を指定した時や、ターゲット定義の最
15713 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】。
15714
15715 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
15716 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
15717 ない【NGKI3232】。
15718
15719 stkをNULLとした場合、stkszで指定したサイズのスタック領域を、コンフィギュ
15720 レータが確保する【NGKI3233】。stkszにターゲット定義の制約に合致しないサ
15721 イズを指定した時には、ターゲット定義の制約に合致するようにサイズを大き
15722 い方に丸めて確保する【NGKI3234】。
15723
15724 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
15725 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
15726 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
15727 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
15728 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】。
15729
15730 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
15731 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
15732 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
15733 なサイズの共有スタック領域を、コンフィギュレータが確保する【NGKI3238】。
15734
15735 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
15736 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】。
15737
15738 【μITRON4.0仕様との関係】
15739
15740 μITRON4.0仕様に定義されていない静的APIである。
15741 -----
15742 ATT_INI 初期化ルーチンの追加 [S] 【NGKI3240】
15743
15744 【静的API】
15745 ATT_INI({ ATR iniatr, intptr_t exinf, INIRTN inirtn })
15746
15747 【パラメータ】
15748 * 初期化ルーチンの追加情報
15749 ATR iniatr 初期化ルーチン属性
15750 intptr_t exinf 初期化ルーチンの拡張情報

15751 INIRTN inirtn 初期化ルーチンの先頭番地
15752
15753 【エラーコード】
15754 E_RSATR 予約属性
15755 ・ iniatrが無効【NGKI3241】
15756 ・ カーネルドメインの囲みの中に記述されていない [P] 【NGKI3242】
15757 E_PAR パラメータエラー
15758 ・ inirtnがプログラムの先頭番地として正しくない【NGKI3243】
15759
15760 【機能】
15761
15762 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
15763 追加する【NGKI3244】。
15764
15765 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
15766 【NGKI3245】。
15767
15768 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
15769 である【NGKI3246】。
15770
15771 【補足説明】
15772
15773 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
15774 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
15775 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。
15776 -----
15777 ATT_TER 終了処理ルーチンの追加 [S] 【NGKI3247】
15778
15779 【静的API】
15780 ATT_TER({ ATR teratr, intptr_t exinf, TERRTN terrtn })
15781
15782 【パラメータ】
15783 ＊終了処理ルーチンの追加情報
15784 ATR teratr 終了処理ルーチン属性
15785 intptr_t exinf 終了処理ルーチンの拡張情報
15786 TERRTN terrtn 終了処理ルーチンの先頭番地
15787
15788 【エラーコード】
15789 E_RSATR 予約属性
15790 ・ teratrが無効【NGKI3248】
15791 ・ カーネルドメインの囲みの中に記述されていない [P] 【NGKI3249】
15792 E_PAR パラメータエラー
15793 ・ terrtnがプログラムの先頭番地として正しくない【NGKI3250】
15794
15795 【機能】
15796
15797 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
15798 ンを追加する【NGKI3251】。
15799
15800 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

15801 【NGKI3252】.
15802
15803 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
15804 である【NGKI3253】.
15805
15806 【補足説明】
15807
15808 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
15809 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
15810 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.
15811
15812 【 μ ITRON4.0仕様との関係】
15813
15814 μ ITRON4.0仕様に定義されていない静的APIである.
15815
15816 ref_cfg コンフィギュレーション情報の参照 [T]
15817
15818 【C言語API】
15819 ER ercd = ref_cfg(T_RCFG *pk_rcfg)
15820
15821 ☆未完成
15822
15823 【TOPPERS/ASPカーネルにおける規定】
15824
15825 ASPカーネルでは、ref_cfgをサポートしない.
15826
15827 【TOPPERS/FMPカーネルにおける規定】
15828
15829 FMPカーネルでは、ref_cfgをサポートしない.
15830
15831 【TOPPERS/HRP2カーネルにおける規定】
15832
15833 HRP2カーネルでは、ref_cfgをサポートしない.
15834
15835 【TOPPERS/SSPカーネルにおける規定】
15836
15837 SSPカーネルでは、ref_cfgをサポートしない.
15838
15839 ref_ver バージョン情報の参照 [T]
15840
15841 【C言語API】
15842 ER ercd = ref_ver(T_RVER *pk_rver)
15843
15844 ☆未完成
15845
15846 【TOPPERS/ASPカーネルにおける規定】
15847
15848 ASPカーネルでは、ref_verをサポートしない.
15849
15850 【TOPPERS/FMPカーネルにおける規定】

15851
15852 FMPカーネルでは、ref_verをサポートしない。
15853
15854 【TOPPERS/HRP2カーネルにおける規定】
15855
15856 HRP2カーネルでは、ref_verをサポートしない。
15857
15858 【TOPPERS/SSPカーネルにおける規定】
15859
15860 SSPカーネルでは、ref_verをサポートしない。
15861 -----
15862
15863
15864 第5章 リファレンス
15865
15866 5.1 サービスコール一覧
15867
15868 (1) タスク管理機能
15869
15870 ER_ID tskid = acre_tsk(const T_CTSK *pk_ctsk) [TD]
15871 ER ercd = sac_tsk(ID tskid, const ACVCT *p_acvct) [TPD]
15872 ER ercd = del_tsk(ID tskid) [TD]
15873 ER ercd = act_tsk(ID tskid) [T]
15874 ER ercd = iact_tsk(ID tskid) [I]
15875 ER ercd = mact_tsk(ID tskid, ID prcid) [TM]
15876 ER ercd = imact_tsk(ID tskid, ID prcid) [IM]
15877 ER_UINT actcnt = can_act(ID tskid) [T]
15878 ER ercd = mig_tsk(ID tskid, ID prcid) [TM]
15879 ER ercd = ext_tsk() [T]
15880 ER ercd = ter_tsk(ID tskid) [T]
15881 ER ercd = chg_pri(ID tskid, PRI tskpri) [T]
15882 ER ercd = get_pri(ID tskid, PRI *p_tskpri) [T]
15883 ER ercd = get_inf(intptr_t *p_exinf) [T]
15884 ER ercd = ref_tsk(ID tskid, T_RTSK *pk_rtsk) [T]
15885
15886 (2) タスク付属同期機能
15887
15888 ER ercd = slp_tsk() [T]
15889 ER ercd = tslp_tsk(TMO tmout) [T]
15890 ER ercd = wup_tsk(ID tskid) [T]
15891 ER ercd = iwup_tsk(ID tskid) [I]
15892 ER_UINT wupcnt = can_wup(ID tskid) [T]
15893 ER ercd = rel_wai(ID tskid) [T]
15894 ER ercd = irel_wai(ID tskid) [I]
15895 ER ercd = sus_tsk(ID tskid) [T]
15896 ER ercd = rsm_tsk(ID tskid) [T]
15897 ER ercd = dis_wai(ID tskid) [TP]
15898 ER ercd = idis_wai(ID tskid) [IP]
15899 ER ercd = ena_wai(ID tskid) [TP]
15900 ER ercd = iena_wai(ID tskid) [IP]

```

15901         ER ercd = dly_tsk(RELTIM dlytim)                                [T]
15902
15903     (3) タスク例外処理機能
15904
15905         ER ercd = def_tex(ID tskid, const T_DTEX *pk_dtex)                [TD]
15906         ER ercd = ras_tex(ID tskid, TEXPTN rasptn)                        [T]
15907         ER ercd = iras_tex(ID tskid, TEXPTN rasptn)                       [I]
15908         ER ercd = dis_tex()                                                [T]
15909         ER ercd = ena_tex()                                                [T]
15910         bool_t state = sns_tex()                                           [TI]
15911         ER ercd = ref_tex(ID tskid, T_RTEX *pk_rtex)                       [T]
15912
15913     (4) 同期・通信機能
15914
15915     セマフォ
15916
15917         ER_ID semid = acre_sem(const T_CSEM *pk_csem)                      [TD]
15918         ER ercd = sac_sem(ID semid, const ACVCT *p_acvct)                 [TPD]
15919         ER ercd = del_sem(ID semid)                                        [TD]
15920         ER ercd = sig_sem(ID semid)                                        [T]
15921         ER ercd = isig_sem(ID semid)                                       [I]
15922         ER ercd = wai_sem(ID semid)                                        [T]
15923         ER ercd = pol_sem(ID semid)                                        [T]
15924         ER ercd = twai_sem(ID semid, TMO tmout)                           [T]
15925         ER ercd = ini_sem(ID semid)                                        [T]
15926         ER ercd = ref_sem(ID semid, T_RSEM *pk_rsem)                       [T]
15927
15928     イベントフラグ
15929
15930         ER_ID flgid = acre_flg(const T_CFLG *pk_cflg)                      [TD]
15931         ER ercd = sac_flg(ID flgid, const ACVCT *p_acvct)                 [TPD]
15932         ER ercd = del_flg(ID flgid)                                        [TD]
15933         ER ercd = set_flg(ID flgid, FLGPTN setptn)                         [T]
15934         ER ercd = iset_flg(ID flgid, FLGPTN setptn)                       [I]
15935         ER ercd = clr_flg(ID flgid, FLGPTN clrptn)                         [T]
15936         ER ercd = wai_flg(ID flgid, FLGPTN waiptn,
15937                                MODE wfmode, FLGPTN *p_flgptn)              [T]
15938         ER ercd = pol_flg(ID flgid, FLGPTN waiptn,
15939                                MODE wfmode, FLGPTN *p_flgptn)              [T]
15940         ER ercd = twai_flg(ID flgid, FLGPTN waiptn,
15941                                MODE wfmode, FLGPTN *p_flgptn, TMO tmout)    [T]
15942         ER ercd = ini_flg(ID flgid)                                        [T]
15943         ER ercd = ref_flg(ID flgid, T_RFLG *pk_rflg)                       [T]
15944
15945     データキュー
15946
15947         ER_ID dtqid = acre_dtq(const T_CDTQ *pk_cdtq)                      [TD]
15948         ER ercd = sac_dtq(ID dtqid, const ACVCT *p_acvct)                 [TPD]
15949         ER ercd = del_dtq(ID dtqid)                                        [TD]
15950         ER ercd = snd_dtq(ID dtqid, intptr_t data)                         [T]

```

15951	ER ercd = psnd_dtq(ID dtqid, intptr_t data)	[T]
15952	ER ercd = ipsnd_dtq(ID dtqid, intptr_t data)	[I]
15953	ER ercd = tsnd_dtq(ID dtqid, intptr_t data, TMO tmout)	[T]
15954	ER ercd = fsnd_dtq(ID dtqid, intptr_t data)	[T]
15955	ER ercd = ifsnd_dtq(ID dtqid, intptr_t data)	[I]
15956	ER ercd = rcv_dtq(ID dtqid, intptr_t *p_data)	[T]
15957	ER ercd = prcv_dtq(ID dtqid, intptr_t *p_data)	[T]
15958	ER ercd = trcv_dtq(ID dtqid, intptr_t *p_data, TMO tmout)	[T]
15959	ER ercd = ini_dtq(ID dtqid)	[T]
15960	ER ercd = ref_dtq(ID dtqid, T_RDTQ *pk_rdtq)	[T]
15961		
15962	優先度データキュー	
15963		
15964	ER_ID pdqid = acre_pdq(const T_CPDQ *pk_cpdq)	[TD]
15965	ER ercd = sac_pdq(ID pdqid, const ACVCT *p_acvct)	[TPD]
15966	ER ercd = del_pdq(ID pdqid)	[TD]
15967	ER ercd = snd_pdq(ID pdqid, intptr_t data, PRI datapri)	[T]
15968	ER ercd = psnd_pdq(ID pdqid, intptr_t data, PRI datapri)	[T]
15969	ER ercd = ipsnd_pdq(ID pdqid, intptr_t data, PRI datapri)	[I]
15970	ER ercd = tsnd_pdq(ID pdqid, intptr_t data,	[T]
15971	PRI datapri, TMO tmout)	
15972	ER ercd = rcv_pdq(ID pdqid, intptr_t *p_data, PRI *p_datapri)	[T]
15973	ER ercd = prcv_pdq(ID pdqid, intptr_t *p_data, PRI *p_datapri)	[T]
15974	ER ercd = trcv_pdq(ID pdqid, intptr_t *p_data,	[T]
15975	PRI *p_datapri, TMO tmout)	
15976	ER ercd = ini_pdq(ID pdqid)	[T]
15977	ER ercd = ref_pdq(ID pdqid, T_RPDQ *pk_rpdq)	[T]
15978		
15979	メールボックス	
15980		
15981	ER_ID mbxid = acre_mbx(const T_CMBX *pk_cmbx)	[TDp]
15982	ER ercd = del_mbx(ID mbxid)	[TDp]
15983	ER ercd = snd_mbx(ID mbxid, T_MSG *pk_msg)	[Tp]
15984	ER ercd = rcv_mbx(ID mbxid, T_MSG **ppk_msg)	[Tp]
15985	ER ercd = prcv_mbx(ID mbxid, T_MSG **ppk_msg)	[Tp]
15986	ER ercd = trcv_mbx(ID mbxid, T_MSG **ppk_msg, TMO tmout)	[Tp]
15987	ER ercd = ini_mbx(ID mbxid)	[Tp]
15988	ER ercd = ref_mbx(ID mbxid, T_RMBX *pk_rmbx)	[Tp]
15989		
15990	ミューテックス	
15991		
15992	ER_ID mtxid = acre_mtx(const T_CMTX *pk_cmtx)	[TD]
15993	ER ercd = sac_mtx(ID mtxid, const ACVCT *p_acvct)	[TPD]
15994	ER ercd = del_mtx(ID mtxid)	[TD]
15995	ER ercd = loc_mtx(ID mtxid)	[T]
15996	ER ercd = ploc_mtx(ID mtxid)	[T]
15997	ER ercd = tloc_mtx(ID mtxid, TMO tmout)	[T]
15998	ER ercd = unl_mtx(ID mtxid)	[T]
15999	ER ercd = ini_mtx(ID mtxid)	[T]
16000	ER ercd = ref_mtx(ID mtxid, T_RMTX *pk_rmtx)	[T]

16001		
16002	メッセージバッファ	
16003		
16004	☆未完成	
16005		
16006	スピンロック	
16007		
16008	ER_ID spnid = acre_spn(const T_CSPN *pk_cspn)	[TMD]
16009	ER ercd = sac_spn(ID spnid, const ACVCT *p_acvct)	[TPMD]
16010	ER ercd = del_spn(ID spnid)	[TMD]
16011	ER ercd = loc_spn(ID spnid)	[TM]
16012	ER ercd = iloc_spn(ID spnid)	[IM]
16013	ER ercd = try_spn(ID spnid)	[TM]
16014	ER ercd = itry_spn(ID spnid)	[IM]
16015	ER ercd = unl_spn(ID spnid)	[TM]
16016	ER ercd = iunl_spn(ID spnid)	[IM]
16017	ER ercd = ref_spn(ID spnid, T_RSPN *pk_rspn)	[TM]
16018		
16019	(5) メモリプール管理機能	
16020		
16021	固定長メモリプール	
16022		
16023	ER_ID mpfid = acre_mpf(const T_CMPF *pk_cmpf)	[TD]
16024	ER ercd = sac_mpf(ID mpfid, const ACVCT *p_acvct)	[TPD]
16025	ER ercd = del_mpf(ID mpfid)	[TD]
16026	ER ercd = get_mpf(ID mpfid, void **p_blk)	[T]
16027	ER ercd = pget_mpf(ID mpfid, void **p_blk)	[T]
16028	ER ercd = tget_mpf(ID mpfid, void **p_blk, TMO tmout)	[T]
16029	ER ercd = rel_mpf(ID mpfid, void *blk)	[T]
16030	ER ercd = ini_mpf(ID mpfid)	[T]
16031	ER ercd = ref_mpf(ID mpfid, T_RMPF *pk_rmpf)	[T]
16032		
16033	(6) 時間管理機能	
16034		
16035	システム時刻管理	
16036		
16037	ER ercd = get_tim(SYSTIM *p_sysstim)	[T]
16038	ER ercd = get_utm(SYSUTM *p_sysutm)	[TI]
16039		
16040	周期ハンドラ	
16041		
16042	ER_ID cycid = acre_cyc(const T_CCYC *pk_ccyc)	[TD]
16043	ER ercd = sac_cyc(ID cycid, const ACVCT *p_acvct)	[TPD]
16044	ER ercd = del_cyc(ID cycid)	[TD]
16045	ER ercd = sta_cyc(ID cycid)	[T]
16046	ER ercd = msta_cyc(ID cycid, ID prcid)	[TM]
16047	ER ercd = stp_cyc(ID cycid)	[T]
16048	ER ercd = ref_cyc(ID cycid, T_RCYC *pk_rcyc)	[T]
16049		
16050	アラームハンドラ	

```

16051
16052     ER ercd = acre_alm(const T_CALM *pk_calm)                [TD]
16053     ER ercd = sac_alm(ID almid, const ACVCT *p_acvct)        [TPD]
16054     ER ercd = del_alm(ID almid)                              [TD]
16055     ER ercd = sta_alm(ID almid, RELTIM almtim)               [T]
16056     ER ercd = ista_alm(ID almid, RELTIM almtim)              [I]
16057     ER ercd = msta_alm(ID almid, RELTIM almtim, ID prcid)    [TM]
16058     ER ercd = imsta_alm(ID almid, RELTIM almtim, ID prcid)   [IM]
16059     ER ercd = stp_alm(ID almid)                              [T]
16060     ER ercd = istp_alm(ID almid)                             [I]
16061     ER ercd = ref_alm(ID almid, T_RALM *pk_ralm)             [T]
16062
16063 オーバランハンドラ
16064
16065     ER ercd = def_ovr(const T_DOVR *pk_dovr)                 [TD]
16066     ER ercd = sta_ovr(ID tskid, OVRTIM ovrtime)              [T]
16067     ER ercd = ista_ovr(ID tskid, OVRTIM ovrtime)             [I]
16068     ER ercd = stp_ovr(ID tskid)                              [T]
16069     ER ercd = istp_ovr(ID tskid)                             [I]
16070     ER ercd = ref_ovr(ID tskid, T_ROVR *pk_rovr)             [T]
16071
16072 (7) システム状態管理機能
16073
16074     ER ercd = sac_sys(const ACVCT *p_acvct)                  [TPD]
16075     ER ercd = rot_rdq(PRI tskpri)                            [T]
16076     ER ercd = irot_rdq(PRI tskpri)                           [I]
16077     ER ercd = mrot_rdq(PRI tskpri, ID prcid)                  [TM]
16078     ER ercd = imrot_rdq(PRI tskpri, ID prcid)                 [IM]
16079     ER ercd = get_tid(ID *p_tskid)                           [T]
16080     ER ercd = iget_tid(ID *p_tskid)                          [I]
16081     ER ercd = get_did(ID *p_domid)                            [TP]
16082     ER ercd = get_pid(ID *p_prcid)                            [TM]
16083     ER ercd = iget_pid(ID *p_prcid)                          [IM]
16084     ER ercd = loc_cpu()                                       [T]
16085     ER ercd = iloc_cpu()                                      [I]
16086     ER ercd = unl_cpu()                                       [T]
16087     ER ercd = iunl_cpu()                                      [I]
16088     ER ercd = dis_dsp()                                       [T]
16089     ER ercd = ena_dsp()                                       [T]
16090     bool_t state = sns_ctx()                                  [TI]
16091     bool_t state = sns_loc()                                   [TI]
16092     bool_t state = sns_dsp()                                   [TI]
16093     bool_t state = sns_dpn()                                   [TI]
16094     bool_t state = sns_ker()                                   [TI]
16095     ER ercd = ext_ker()                                       [TI]
16096     ER ercd = ref_sys(T_RSYS *pk_rsys)                        [T]
16097
16098 (8) メモリオブジェクト管理機能
16099
16100     ER ercd = att_mem(const T_AMEM *pk_amem)                  [TPD]

```

```

16101      ER ercd = att_pma(const T_AMEM *pk_apma) [TPD]
16102      ER ercd = sac_mem(const void *base, const ACVCT *p_acvct) [TPD]
16103      ER ercd = det_mem(const void *base) [TPD]
16104      ER ercd = prb_mem(const void *base, SIZE size, [TP]
16105                      ID tskid, MODE pmmode)
16106      ER ercd = ref_mem(const void *base, T_RMEM *pk_rmem) [TP]
16107
16108      (9) 割込み管理機能
16109
16110      ER ercd = cfg_int(INTNO intno, const T_CINT *pk_cint) [TD]
16111      ER_ID isrid = acre_isr(const T_CISR *pk_cisr) [TD]
16112      ER ercd = sac_isr(ID isrid, const ACVCT *p_acvct) [TPD]
16113      ER ercd = del_isr(ID isrid) [TD]
16114      ER ercd = ref_isr(ID isrid, T_RISR *pk_risr) [T]
16115      ER ercd = def_inh(INHNO inhno, const T_DINH *pk_dinh) [TD]
16116      ER ercd = dis_int(INTNO intno) [T]
16117      ER ercd = ena_int(INTNO intno) [T]
16118      ER ercd = ref_int(INTNO intno, T_RINT *pk_rint) [T]
16119      ER ercd = chg_ipm(PRI intpri) [T]
16120      ER ercd = get_ipm(PRI *p_intpri) [T]
16121
16122      (10) CPU例外管理機能
16123
16124      ER ercd = def_exc(EXCNO excno, const T_DEXC *pk_dexc) [TD]
16125      bool_t stat = xsns_dpn(void *p_excinf) [TI]
16126      bool_t stat = xsns_xpn(void *p_excinf) [TI]
16127
16128      (11) 拡張サービスコール管理機能
16129
16130      ER ercd = def_svc(FN fncd, const T_DSVC *pk_dsvc) [TPD]
16131      ER_UINT ercd = cal_svc(FN fncd, intptr_t par1, intptr_t par2, [TIP]
16132                      intptr_t par3, intptr_t par4, intptr_t par5)
16133
16134      (12) システム構成管理機能
16135
16136      ER ercd = ref_cfg(T_RCFG *pk_rcfg) [T]
16137      ER ercd = ref_ver(T_RVER *pk_rver) [T]
16138
16139      5.2 静的API一覧
16140
16141      (1) タスク管理機能
16142
16143      *保護機能対応でないカーネルの場合
16144      CRE_TSK(ID tskid, { ATR tskatr, intptr_t exinf, TASK task, [S]
16145                      PRI itskpri, SIZE stksz, STK_T *stk })
16146
16147      *保護機能対応カーネルの場合
16148      CRE_TSK(ID tskid, { ATR tskatr, intptr_t exinf, TASK task, [SP]
16149                      PRI itskpri, SIZE stksz, STK_T *stk,
16150                      SIZE sstksz, STK_T *sstk })

```

```

16151 ※ sstksizeおよびsstkの記述は省略することができる。
16152
16153 AID_TSK(uint_t notsk) [SD]
16154 SAC_TSK(ID tskid, { ACPTN acptn1, ACPTN acptn2, [SP]
16155 ACPTN acptn3, ACPTN acptn4 })
16156 DEF_EPR(ID tskid, { PRI exeprid }) [S]
16157
16158 (2) タスク付属同期機能
16159
16160 なし
16161
16162 (3) タスク例外処理機能
16163
16164 DEF_TEX(ID tskid, { ATR texatr, TEXRTN texrtn }) [S]
16165
16166 (4) 同期・通信機能
16167
16168 セマフォ
16169
16170 CRE_SEM(ID semid, { ATR sematr, uint_t isemcnt, uint_t maxsem }) [S]
16171 AID_SEM(uint_t noseid) [SD]
16172 SAC_SEM(ID semid, { ACPTN acptn1, ACPTN acptn2, [SP]
16173 ACPTN acptn3, ACPTN acptn4 })
16174
16175 イベントフラグ
16176
16177 CRE_FLG(ID flgid, { ATR flgatr, FLGPTR iflgptr }) [S]
16178 AID_FLG(uint_t noflg) [SD]
16179 SAC_FLG(ID flgid, { ACPTN acptn1, ACPTN acptn2, [SP]
16180 ACPTN acptn3, ACPTN acptn4 })
16181
16182 データキュー
16183
16184 CRE_DTQ(ID dtqid, { ATR dtqatr, uint_t dtqcnt, void *dtqmb }) [S]
16185 AID_DTQ(uint_t nodtq) [SD]
16186 SAC_DTQ(ID dtqid, { ACPTN acptn1, ACPTN acptn2, [SP]
16187 ACPTN acptn3, ACPTN acptn4 })
16188
16189 優先度データキュー
16190
16191 CRE_PDQ(ID pdqid, { ATR pdqatr, uint_t pdqcnt, [S]
16192 PRI maxdpri, void *pdqmb })
16193 AID_PDQ(uint_t nopdq) [SD]
16194 SAC_PDQ(ID pdqid, { ACPTN acptn1, ACPTN acptn2, [SP]
16195 ACPTN acptn3, ACPTN acptn4 })
16196
16197 メールボックス
16198
16199 CRE_MBX(ID mbxid, { ATR mbxatr, PRI maxmpri, void *mprihd }) [Sp]
16200 AID_MBX(uint_t nombx) [SpD]

```

```

16201
16202 ミューテックス
16203
16204     CRE_MTX(ID mtxid, { ATR mtxatr, PRI ceilpri }) [S]
16205     AID_MTX(uint_t nomtx) [SD]
16206     SAC_MTX(ID mtxid, { ACPTN acptn1, ACPTN acptn2, [SP]
16207                     ACPTN acptn3, ACPTN acptn4 })
16208
16209 メッセージバッファ
16210
16211 ☆未完成
16212
16213 スピンロック
16214
16215     CRE_SPN(ID spnid, { ATR spnatr }) [SM]
16216     AID_SPN(uint_t nospn) [SMD]
16217     SAC_SPN(ID spnid, { ACPTN acptn1, ACPTN acptn2, [SPM]
16218                     ACPTN acptn3, ACPTN acptn4 })
16219
16220 (5) メモリプール管理機能
16221
16222 固定長メモリプール
16223
16224     CRE_MPF(ID mpfid, { ATR mpfatr, uint_t blkent, uint_t blkksz, [S]
16225                     MPF_T *mpf, void *mpfmb })
16226     AID_MPF(uint_t nompf) [SD]
16227     SAC_MPF(ID mpfid, { ACPTN acptn1, ACPTN acptn2, [SP]
16228                     ACPTN acptn3, ACPTN acptn4 })
16229
16230 (6) 時間管理機能
16231
16232 周期ハンドラ
16233
16234     CRE_CYC(ID cycid, { ATR cycatr, intptr_t exinf, CYCHDR cychdr, [S]
16235                     RELTIM cyctim, RELTIM cycphs })
16236     AID_CYC(uint_t nocyc) [SD]
16237     SAC_CYC(ID cycid, { ACPTN acptn1, ACPTN acptn2, [SP]
16238                     ACPTN acptn3, ACPTN acptn4 })
16239
16240 アラームハンドラ
16241
16242     CRE_ALM(ID almid, { ATR almatr, intptr_t exinf, ALMHDR almhdr }) [S]
16243     AID_ALM(uint_t noalm) [SD]
16244     SAC_ALM(ID almid, { ACPTN acptn1, ACPTN acptn2, [SP]
16245                     ACPTN acptn3, ACPTN acptn4 })
16246
16247 オーバランハンドラ
16248
16249     DEF_OVR({ ATR ovratr, OVRHDR ovrhdr }) [S]
16250

```

```

16251      (7) システム状態管理機能
16252
16253      SAC_SYS({ ACPTN acptn1, ACPTN acptn2, [SP]
16254              ACPTN acptn3, ACPTN acptn4 })
16255
16256      (8) メモリオブジェクト管理機能
16257
16258      ATT_REG("メモリリージョン名", [SP]
16259            { ATR regatr, void *base, SIZE size })
16260      DEF_SRG("標準ROMリージョン名", "標準RAMリージョン名") [SP]
16261      ATT_SEC("セクション名", { ATR mematr, "メモリリージョン名" }) [SP]
16262      ATA_SEC("セクション名", { ATR mematr, "メモリリージョン名", [SP]
16263            { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
16264      LNK_SEC("セクション名", { "メモリリージョン名" }) [SP]
16265      ATT_MOD("オブジェクトモジュール名") [SP]
16266      ATA_MOD("オブジェクトモジュール名", [SP]
16267            { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
16268      ATT_MEM({ ATR mematr, void *base, SIZE size }) [SP]
16269      ATA_MEM({ ATR mematr, void *base, SIZE size, [SP]
16270            { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
16271      ATT_PMA({ ATR mematr, void *base, SIZE size, void *paddr }) [SP]
16272      ATA_PMA({ ATR mematr, void *base, SIZE size, void *paddr }, [SP]
16273            { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
16274
16275      (9) 割り込み管理機能
16276
16277      CFG_INT(INTNO intno, { ATR intatr, PRI intpri }) [S]
16278      CRE_ISR(ID isrid, { ATR isratr, intptr_t exinf, [S]
16279            INTNO intno, ISR isr, PRI isrpri })
16280      ATT_ISR({ ATR isratr, intptr_t exinf, [S]
16281            INTNO intno, ISR isr, PRI isrpri })
16282      AID_ISR(uint_t noisr) [SD]
16283      SAC_ISR(ID isrid, { ACPTN acptn1, ACPTN acptn2, [SP]
16284            ACPTN acptn3, ACPTN acptn4 })
16285      DEF_INH(INHNO inhno, { ATR inhatr, INTHDR inthdr }) [S]
16286
16287      (10) CPU例外管理機能
16288
16289      DEF_EXC(EXCNO excno, { ATR excatr, EXCHDR exchdr }) [S]
16290
16291      (11) 拡張サービスコール管理機能
16292
16293      DEF_SVC(FN fncd, { ATR svcatr, EXTSVC svcrtm, SIZE stksz }) [SP]
16294
16295      (12) システム構成管理機能
16296
16297      DEF_ICS({ SIZE istksz, STK_T *istk }) [S]
16298      DEF_STK({ SIZE stksz, STK_T *stk }) [S]
16299      ATT_INI({ ATR iniatr, intptr_t exinf, INIRTN inirtn }) [S]
16300      ATT_TER({ ATR teratr, intptr_t exinf, TERRTN terrtn }) [S]

```

16301

16302 5.3 データ型

16303

16304 5.3.1 TOPPERS共通データ型

16305

16306 int8_t 符号付き8ビット整数 (オプション, C99準拠)

16307 uint8_t 符号無し8ビット整数 (オプション, C99準拠)

16308 int16_t 符号付き16ビット整数 (C99準拠)

16309 uint16_t 符号無し16ビット整数 (C99準拠)

16310 int32_t 符号付き32ビット整数 (C99準拠)

16311 uint32_t 符号無し32ビット整数 (C99準拠)

16312 int64_t 符号付き64ビット整数 (オプション, C99準拠)

16313 uint64_t 符号無し64ビット整数 (オプション, C99準拠)

16314 int128_t 符号付き128ビット整数 (オプション, C99準拠)

16315 uint128_t 符号無し128ビット整数 (オプション, C99準拠)

16316

16317 int_least8_t 8ビット以上の符号付き整数 (C99準拠)

16318 uint_least8_t int_least8_t型と同じサイズの符号無し整数 (C99準拠)

16319

16320 float32_t IEEE754準拠の32ビット単精度浮動小数点数 (オプション)

16321 double64_t IEEE754準拠の64ビット倍精度浮動小数点数 (オプション)

16322

16323 bool_t 真偽値 (trueまたはfalse)

16324 int_t 16ビット以上の符号付き整数

16325 uint_t int_t型と同じサイズの符号無し整数

16326 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

16327 ulong_t long_t型と同じサイズの符号無し整数

16328

16329 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)

16330 uintptr_t intptr_t型と同じサイズの符号無し整数 (C99準拠)

16331

16332 FN 機能コード (符号付き整数, int_tに定義)

16333 ER エラーコード (符号付き整数, int_tに定義)

16334 ID オブジェクトのID番号 (符号付き整数, int_tに定義)

16335 ATR オブジェクト属性 (符号無し整数, uint_tに定義)

16336 STAT オブジェクトの状態 (符号無し整数, uint_tに定義)

16337 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)

16338 PRI 優先度 (符号付き整数, int_tに定義)

16339 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
16340 サイズの符号無し整数型に定義)

16341

16342 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)

16343 RELTIM 相対時間 (符号無し整数, 単位はミリ秒, uint_tに定義)

16344 SYSTIM システム時刻 (符号無し整数, 単位はミリ秒, ulong_tに定義)

16345 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,
16346 ulong_tに定義)

16347

16348 FP プログラムの起動番地 (型の定まらない関数ポインタ)

16349

16350 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)

```

16351      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，
16352                  負のID番号は格納できない）
16353      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
16354                  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
16355                  より1ビット短い）
16356
16357      MB_T       オブジェクト管理領域を確保するためのデータ型
16358
16359      ACPTN      アクセス許可パターン（符号無し32ビット整数，uint32_tに
16360                  定義）
16361
16362      typedef struct acvct {          /* アクセス許可ベクタ */
16363          ACPTN    acptn1;            /* 通常操作1のアクセス許可パターン */
16364          ACPTN    acptn2;            /* 通常操作2のアクセス許可パターン */
16365          ACPTN    acptn3;            /* 管理操作のアクセス許可パターン */
16366          ACPTN    acptn4;            /* 参照操作のアクセス許可パターン */
16367      } ACVCT;
16368
16369      5.3.2 カーネルの使用するデータ型
16370
16371      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）
16372      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）
16373      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）
16374      INTNO       割込み番号（符号無し整数，uint_tに定義）
16375      INHNO       割込みハンドラ番号（符号無し整数，uint_tに定義）
16376      EXCNO       CPU例外ハンドラ番号（符号無し整数，uint_tに定義）
16377
16378      TASK        タスクのメインルーチン（関数ポインタ）
16379      TEXRTN      タスク例外処理ルーチン（関数ポインタ）
16380      CYCHDR      周期ハンドラ（関数ポインタ）
16381      ALMHDR      アラームハンドラ（関数ポインタ）
16382      OVRHDR      オーバランハンドラ（関数ポインタ）
16383      ISR         割込みサービスルーチン（関数ポインタ）
16384      INTHDR      割込みハンドラ（関数ポインタ）
16385      EXCHDR      CPU例外ハンドラ（関数ポインタ）
16386      EXTSVC      拡張サービスコール（関数ポインタ）
16387      INIRTN      初期化ルーチン（関数ポインタ）
16388      TERRTN      終了処理ルーチン（関数ポインタ）
16389
16390      STK_T       スタック領域を確保するためのデータ型
16391      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
16392
16393      メールボックスのメッセージヘッダ【NGKI4001】
16394
16395      typedef struct t_msg {
16396          struct t_msg    *pk_next;
16397      } T_MSG;
16398
16399      メールボックスの優先度付きメッセージヘッダ【NGKI4002】
16400

```



```

16401     typedef struct t_msg_pri {
16402         T_MSG      msgque;          /* メールボックスのメッセージヘッダ */
16403         PRI        msgpri;          /* メッセージ優先度 */
16404     } T_MSG_PRI;
16405
16406 5.3.3 カーネルの使用するパケット形式
16407
16408 (1) タスク管理機能
16409
16410 タスクの生成情報のパケット形式【NGKI4003】
16411
16412     typedef struct t_ctsk {
16413         ATR        tskatr;          /* タスク属性 */
16414         intptr_t   exinf;           /* タスクの拡張情報 */
16415         TASK       task;            /* タスクのメインルーチンの先頭番地 */
16416         PRI        itskpri;         /* タスクの起動時優先度 */
16417         SIZE       stksz;           /* タスクのスタック領域のサイズ */
16418         STK_T *    stk;             /* タスクのスタック領域の先頭番地 */
16419         /* 以下は、保護機能対応カーネルの場合 */
16420         SIZE       sstksz;          /* タスクのシステムスタック領域のサイズ */
16421         STK_T *    sstk;           /* タスクのシステムスタック領域の先頭番地 */
16422     } T_CTSK;
16423
16424 タスクの現在状態のパケット形式【NGKI4004】
16425
16426     typedef struct t_rtsk {
16427         STAT       tskstat;         /* タスク状態 */
16428         PRI        tskpri;          /* タスクの現在優先度 */
16429         PRI        tskbpri;         /* タスクのベース優先度 */
16430         STAT       tskwait;         /* 待ち要因 */
16431         ID         wobjid;          /* 待ち対象のオブジェクトのID */
16432         TMO        lefttmo;         /* タイムアウトするまでの時間 */
16433         uint_t     actcnt;          /* 起動要求キューイング数 */
16434         uint_t     wupcnt;          /* 起床要求キューイング数 */
16435         /* 以下は、保護機能対応カーネルの場合 */
16436         bool_t     texmsk;          /* タスク例外マスク状態か否か */
16437         bool_t     waifbd;          /* 待ち禁止状態か否か */
16438         uint_t     svclevel;        /* 拡張サービスコールのネストレベル */
16439         /* 以下は、マルチプロセッサ対応カーネルの場合 */
16440         ID         prcid;           /* 割付けプロセッサのID */
16441         ID         actprc           /* 次の起動時の割付けプロセッサのID */
16442     } T_RTSK;
16443
16444 (2) タスク付属同期機能
16445
16446 なし
16447
16448 (3) タスク例外処理機能
16449
16450 タスク例外処理ルーチンの定義情報のパケット形式【NGKI4005】

```

```
16451
16452     typedef struct t_dtex {
16453         ATR          texatr;      /* タスク例外処理ルーチン属性 */
16454         TEXRTN       texrtn;      /* タスク例外処理ルーチンの先頭番地 */
16455     } T_DTEX;
16456
16457     タスク例外処理の現在状態のパケット形式【NGKI4006】
16458
16459     typedef struct t_rtex {
16460         STAT         texstat;      /* タスク例外処理の状態 */
16461         TEXPTN       pndptn;      /* 保留例外要因 */
16462     } T_RTEX;
16463
16464     (4) 同期・通信機能
16465
16466     セマフォの生成情報のパケット形式【NGKI4007】
16467
16468     typedef struct t_csem {
16469         ATR          sematr;      /* セマフォ属性 */
16470         uint_t       isemcnt;      /* セマフォの初期資源数 */
16471         uint_t       maxsem;      /* セマフォの最大資源数 */
16472     } T_CSEM;
16473
16474     セマフォの現在状態のパケット形式【NGKI4008】
16475
16476     typedef struct t_rsem {
16477         ID           wtskid;      /* セマフォの待ち行列の先頭のタスクのID番号 */
16478         uint_t       semcnt;      /* セマフォの資源数 */
16479     } T_RSEM;
16480
16481     イベントフラグの生成情報のパケット形式【NGKI4009】
16482
16483     typedef struct t_cflg {
16484         ATR          flgatr;      /* イベントフラグ属性 */
16485         FLGPTN       iflgptn;      /* イベントフラグの初期ビットパターン */
16486     } T_CFLG;
16487
16488     イベントフラグの現在状態のパケット形式【NGKI4010】
16489
16490     typedef struct t_rflg {
16491         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 */
16492         FLGPTN       flgptn;      /* イベントフラグのビットパターン */
16493     } T_RFLG;
16494
16495     データキューの生成情報のパケット形式【NGKI4011】
16496
16497     typedef struct t_cdtq {
16498         ATR          dtqatr;      /* データキュー属性 */
16499         uint_t       dtqcnt;      /* データキュー管理領域に格納できるデータ数 */
16500
```

```

16501         void *      dtqmb;      /* データキュー管理領域の先頭番地 */
16502     } T_CDTQ;
16503
16504     データキューの現在状態の packets 形式【NGKI4012】
16505
16506     typedef struct t_rdtq {
16507         ID          stskid;      /* データキューの送信待ち行列の先頭のタ
16508                                スクのID番号 */
16509         ID          rtskid;      /* データキューの受信待ち行列の先頭のタ
16510                                スクのID番号 */
16511         uint_t      sdtqcnt;     /* データキュー管理領域に格納されている
16512                                データの数 */
16513     } T_RDTQ;
16514
16515     優先度データキューの生成情報の packets 形式【NGKI4013】
16516
16517     typedef struct t_cpdq {
16518         ATR          pdqatr;      /* 優先度データキュー属性 */
16519         uint_t        pdqcnt;     /* 優先度データキュー管理領域に格納でき
16520                                るデータ数 */
16521         PRI          maxdpri;     /* 優先度データキューに送信できるデータ
16522                                優先度の最大値 */
16523         void *        pdqmb;      /* 優先度データキュー管理領域の先頭番地 */
16524     } T_CPDQ;
16525
16526     優先度データキューの現在状態の packets 形式【NGKI4014】
16527
16528     typedef struct t_rpdq {
16529         ID          stskid;      /* 優先度データキューの送信待ち行列の先
16530                                頭のタスクのID番号 */
16531         ID          rtskid;      /* 優先度データキューの受信待ち行列の先
16532                                頭のタスクのID番号 */
16533         uint_t      spdqcnt;     /* 優先度データキュー管理領域に格納され
16534                                ているデータの数 */
16535     } T_RPDQ;
16536
16537     メールボックスの生成情報の packets 形式【NGKI4015】
16538
16539     typedef struct t_cmbx {
16540         ATR          mbxatr;      /* メールボックス属性 */
16541         PRI          maxmpri;     /* 優先度メールボックスに送信できるメッ
16542                                セージ優先度の最大値 */
16543         void *        mprihd;     /* 優先度別のメッセージキューヘッダ領域
16544                                の先頭番地 */
16545     } T_CMBX;
16546
16547     メールボックスの現在状態の packets 形式【NGKI4016】
16548
16549     typedef struct t_rmbx {
16550         ID          wtskid;      /* メールボックスの待ち行列の先頭のタスク

```

```

16551                                     のID番号 */
16552         T_MSG          *pk_msg;    /* メッセージキューの先頭につながれたメッ
16553                                     セージの先頭番地 */
16554     } T_RMBX;
16555
16556 ミューテックスの生成情報のパケット形式【NGKI4017】
16557
16558     typedef struct t_cmtx {
16559         ATR          mtxatr;    /* ミューテックス属性 */
16560         PRI          ceilpri;   /* ミューテックスの上限優先度 */
16561     } T_CMTX;
16562
16563 ミューテックスの現在状態のパケット形式【NGKI4018】
16564
16565     typedef struct t_rmtx {
16566         ID          htskid;     /* ミューテックスをロックしているタス
16567                                     クのID番号 */
16568         ID          wtskid;     /* ミューテックスの待ち行列の先頭のタ
16569                                     スクのID番号 */
16570     } T_RMTX;
16571
16572 メッセージバッファの生成情報のパケット形式
16573
16574 ☆未完成
16575
16576 メッセージバッファの現在状態のパケット形式
16577
16578 ☆未完成
16579
16580 スピンロックの生成情報のパケット形式【NGKI4019】
16581
16582     typedef struct t_cspn {
16583         ATR          spnatr;    /* スピンロック属性 */
16584     } T_CSPN;
16585
16586 スピンロックの現在状態のパケット形式【NGKI4020】
16587
16588     typedef struct t_rspn {
16589         STAT          spnstat    /* スピンロックのロック状態 */
16590     } T_RSPN;
16591
16592 (5) メモリプール管理機能
16593
16594 固定長メモリアールの生成情報のパケット形式【NGKI4021】
16595
16596     typedef struct t_cmpf {
16597         ATR          mpfatr;    /* 固定長メモリアール属性 */
16598         uint_t       blkcnt;    /* 獲得できる固定長メモリブロックの数 */
16599         uint_t       blksiz;    /* 固定長メモリブロックのサイズ */
16600         MPF_T *      mpf;       /* 固定長メモリアール領域の先頭番地 */

```

```

16601         void *      mpfmb;      /* 固定長メモリプール管理領域の先頭番地 */
16602     } T_CMPF;

```

16603

16604 固定長メモリプールの現在状態の packets 形式【NGKI4022】

16605

```

16606     typedef struct t_rmpf {
16607         ID          wtskid;      /* 固定長メモリプールの待ち行列の先頭の
16608                                タスクのID番号 */
16609         uint_t      fblkcnt;     /* 固定長メモリプール領域の空きメモリ領
16610                                域に割り付けることができる固定長メモ
16611                                リブロックの数 */
16612     } T_RMPF;

```

16613

16614 (6) 時間管理機能

16615

16616 周期ハンドラの生成情報の packets 形式【NGKI4023】

16617

```

16618     typedef struct t_ccyc {
16619         ATR          cycatr;     /* 周期ハンドラ属性 */
16620         intptr_t     exinf;      /* 周期ハンドラの拡張情報 */
16621         CYCHDR       cychdr;     /* 周期ハンドラ先頭番地 */
16622         RELTIM       cycetim;    /* 周期ハンドラの起動周期 */
16623         RELTIM       cycphs;     /* 周期ハンドラの起動位相 */
16624     } T_CCYC;

```

16625

16626 周期ハンドラの現在状態の packets 形式【NGKI4024】

16627

```

16628     typedef struct t_rcyc {
16629         STAT          cycstat;    /* 周期ハンドラの動作状態 */
16630         RELTIM        lefttim;    /* 次に周期ハンドラを起動する時刻までの
16631                                相対時間 */
16632         /* 以下は、マルチプロセッサ対応カーネルの場合 */
16633         ID            prcid;      /* 割付けプロセッサのID */
16634     } T_RCYC;

```

16635

16636 アラームハンドラの生成情報の packets 形式【NGKI4025】

16637

```

16638     typedef struct t_calm {
16639         ATR          almatr;     /* アラームハンドラ属性 */
16640         intptr_t     exinf;      /* アラームハンドラの拡張情報 */
16641         ALMHDR       almhdr;     /* アラームハンドラ先頭番地 */
16642     } T_CALM;

```

16643

16644 アラームハンドラの現在状態の packets 形式【NGKI4026】

16645

```

16646     typedef struct t_ralm {
16647         STAT          almstat;    /* アラームハンドラの動作状態 */
16648         RELTIM        lefttim;    /* アラームハンドラを起動する時刻までの
16649                                相対時間 */
16650         /* 以下は、マルチプロセッサ対応カーネルの場合 */

```

```

16651         ID          prcid;      /* 割付けプロセッサのID */
16652     } T_RALM;
16653
16654 オーバランハンドラの定義情報のパケット形式【NGKI4027】
16655
16656     typedef struct t_dovr {
16657         ATR          ovratr;      /* オーバランハンドラ属性 */
16658         OVRHDR        ovrhdr;      /* オーバランハンドラ先頭番地 */
16659     } T_DOVR;
16660
16661 オーバランハンドラの現在状態のパケット形式【NGKI4028】
16662
16663     typedef struct t_rovr {
16664         STAT          ovrstat;      /* オーバランハンドラ動作状態 */
16665         OVRTIM        leftotm;      /* 残りプロセッサ時間 */
16666     } T_ROVR;
16667
16668 (7) システム状態管理機能
16669
16670 システムの現在状態のパケット形式
16671
16672 ☆未完成
16673
16674 (8) メモリオブジェクト管理機能
16675
16676 メモリオブジェクトの登録情報のパケット形式【NGKI4029】
16677
16678     typedef struct t_amem {
16679         ATR          mematr      /* メモリオブジェクト属性 */
16680         void *        base      /* 登録するメモリ領域先頭番地 */
16681         SIZE          size      /* 登録するメモリ領域サイズ (バイト数) */
16682     } T_AMEM;
16683
16684 物理メモリ領域の登録情報のパケット形式【NGKI4030】
16685
16686     typedef struct t_apma {
16687         ATR          mematr      /* メモリオブジェクト属性 */
16688         void *        base      /* 登録するメモリ領域先頭番地 */
16689         SIZE          size      /* 登録するメモリ領域サイズ (バイト数) */
16690         void *        paddr      /* 登録するメモリ領域物理アドレス先頭
16691                                番地 */
16692     } T_APMA;
16693
16694 メモリオブジェクトの現在状態のパケット形式
16695
16696 ☆未完成
16697
16698 (9) 割込み管理機能
16699
16700 割込み要求ラインの属性の設定情報のパケット形式【NGKI4031】

```

```
16701
16702     typedef struct t_cint {
16703         ATR          intatr;    /* 割込み要求ライン属性 */
16704         PRI          intpri;    /* 割込み優先度 */
16705     } T_CINT;
16706
16707 割込みサービスルーチンの生成情報のパケット形式【NGKI4032】
16708
16709     typedef struct t_cisr {
16710         ATR          isratr;    /* 割込みサービスルーチン属性 */
16711         intptr_t     exinf;     /* 割込みサービスルーチンの拡張情報 */
16712         INTNO        intno;     /* 割込みサービスルーチンを登録する割込
16713                                み番号 */
16714         ISR          isr;       /* 割込みサービスルーチンの先頭番地 */
16715         PRI          isrpri;    /* 割込みサービスルーチン優先度 */
16716     } T_CISR;
16717
16718 割込みサービスルーチンの現在状態のパケット形式
16719
16720 ☆未完成
16721
16722 割込みハンドラの定義情報のパケット形式【NGKI4033】
16723
16724     typedef struct t_dinh {
16725         ATR          inhatr;    /* 割込みハンドラ属性 */
16726         INTHDR        inthdr;    /* 割込みハンドラ先頭番地 */
16727     } T_DINH;
16728
16729 割込み要求ラインの現在状態のパケット形式
16730
16731 ☆未完成
16732
16733 (10) CPU例外管理機能
16734
16735 CPU例外ハンドラの定義情報のパケット形式【NGKI4034】
16736
16737     typedef struct t_dexc {
16738         ATR          excatr;    /* CPU例外ハンドラ属性 */
16739         EXCHDR        exchdr;    /* CPU例外ハンドラ先頭番地 */
16740     } T_DEXC;
16741
16742 (11) 拡張サービスコール管理機能
16743
16744 拡張サービスコールの定義情報のパケット形式【NGKI4035】
16745
16746     typedef struct t_dsvc {
16747         ATR          svcatr     /* 拡張サービスコール属性 */
16748         EXTSVC        svertn     /* 拡張サービスコール先頭番地 */
16749         SIZE          stksz      /* 拡張サービスコールで使用するスタック
16750                                サイズ */
```

```

16751         } T_DSVC;
16752
16753     (12) システム構成管理機能
16754
16755     コンフィギュレーション情報のパケット形式
16756
16757     ☆未完成
16758
16759     バージョン情報のパケット形式
16760
16761     ☆未完成
16762
16763     5.4 定数とマクロ
16764
16765     5.4.1 TOPPERS共通定数
16766
16767     (1) 一般定数
16768
16769         NULL                無効ポインタ
16770
16771         true                1        真
16772         false               0        偽
16773
16774         E_OK                0        正常終了
16775
16776     (2) 整数型に格納できる最大値と最小値
16777
16778         INT8_MAX             int8_tに格納できる最大値 (オプション, C99準拠)
16779         INT8_MIN             int8_tに格納できる最小値 (オプション, C99準拠)
16780         UINT8_MAX            uint8_tに格納できる最大値 (オプション, C99準拠)
16781         INT16_MAX            int16_tに格納できる最大値 (C99準拠)
16782         INT16_MIN            int16_tに格納できる最小値 (C99準拠)
16783         UINT16_MAX           uint16_tに格納できる最大値 (C99準拠)
16784         INT32_MAX            int32_tに格納できる最大値 (C99準拠)
16785         INT32_MIN            int32_tに格納できる最小値 (C99準拠)
16786         UINT32_MAX           uint32_tに格納できる最大値 (C99準拠)
16787         INT64_MAX            int64_tに格納できる最大値 (オプション, C99準拠)
16788         INT64_MIN            int64_tに格納できる最小値 (オプション, C99準拠)
16789         UINT64_MAX           uint64_tに格納できる最大値 (オプション, C99準拠)
16790         INT128_MAX           int128_tに格納できる最大値 (オプション, C99準拠)
16791         INT128_MIN           int128_tに格納できる最小値 (オプション, C99準拠)
16792         UINT128_MAX          uint128_tに格納できる最大値 (オプション, C99準拠)
16793
16794         INT_LEAST8_MAX       int_least8_tに格納できる最大値 (C99準拠)
16795         INT_LEAST8_MIN       int_least8_tに格納できる最小値 (C99準拠)
16796         UINT_LEAST8_MAX      uint_least8_tに格納できる最大値 (C99準拠)
16797         INT_MAX               int_tに格納できる最大値 (C90準拠)
16798         INT_MIN               int_tに格納できる最小値 (C90準拠)
16799         UINT_MAX              uint_tに格納できる最大値 (C90準拠)
16800         LONG_MAX              long_tに格納できる最大値 (C90準拠)

```


16801	LONG_MIN		long_tに格納できる最小値 (C90準拠)
16802	ULONG_MAX		ulong_tに格納できる最大値 (C90準拠)
16803			
16804	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)
16805			
16806	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)
16807			
16808	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)
16809			
16810	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)
16811			
16812			
16813	(3) 整数型のビット数		
16814			
16815	CHAR_BIT		char型のビット数 (C90準拠)
16816			
16817	(4) オブジェクト属性		
16818			
16819	TA_NULL	0U	オブジェクト属性を指定しない
16820			
16821	(5) タイムアウト指定		
16822			
16823	TMO_POL	0	ポーリング
16824	TMO_FEVR	-1	永久待ち
16825	TMO_NBLK	-2	ノンブロッキング
16826			
16827	(6) アクセス許可パターン		
16828			
16829	TACP_KERNEL	0U	カーネルドメインのみにアクセスを許可
16830	TACP_SHARED	~0U	すべての保護ドメインにアクセスを許可
16831			
16832	5.4.2 TOPPERS共通マクロ		
16833			
16834	(1) 整数定数を作るマクロ		
16835			
16836	INT8_C(val)		int_least8_t型の定数を作るマクロ (C99準拠)
16837	UINT8_C(val)		uint_least8_t型の定数を作るマクロ (C99準拠)
16838	INT16_C(val)		int16_t型の定数を作るマクロ (C99準拠)
16839	UINT16_C(val)		uint16_t型の定数を作るマクロ (C99準拠)
16840	INT32_C(val)		int32_t型の定数を作るマクロ (C99準拠)
16841	UINT32_C(val)		uint32_t型の定数を作るマクロ (C99準拠)
16842	INT64_C(val)		int64_t型の定数を作るマクロ (オプション, C99準拠)
16843	UINT64_C(val)		uint64_t型の定数を作るマクロ (オプション, C99準拠)
16844	INT128_C(val)		int128_t型の定数を作るマクロ (オプション, C99準拠)
16845	UINT128_C(val)		uint128_t型の定数を作るマクロ (オプション, C99準拠)
16846			
16847	UINT_C(val)		uint_t型の定数を作るマクロ
16848	ULONG_C(val)		ulong_t型の定数を作るマクロ
16849			
16850	(2) 型に関する情報を取り出すためのマクロ		

16851			
16852	<code>offsetof(structure, field)</code>	構造体structure中のフィールドfieldの	
16853		バイト位置を返すマクロ (C90準拠)	
16854			
16855	<code>alignof(type)</code>	型typeのアラインメント単位を返すマクロ	
16856			
16857	<code>ALIGN_TYPE(addr, type)</code>	番地addrが型typeに対してアラインしてい	
16858		るかどうかを返すマクロ	
16859			
16860	(3) <code>assert</code> マクロ		
16861			
16862	<code>assert(exp)</code>	<code>exp</code> が成立しているかを検査するマクロ (C90準拠)	
16863			
16864	(4) コンパイラの拡張機能のためのマクロ		
16865			
16866	<code>inline</code>	インライン関数	
16867	<code>Inline</code>	ファイルローカルなインライン関数	
16868	<code>asm</code>	インラインアセンブラ	
16869	<code>Asm</code>	インラインアセンブラ (最適化抑止)	
16870	<code>throw()</code>	例外を発生しない関数	
16871	<code>NoReturn</code>	リターンしない関数	
16872			
16873	(5) エラーコード生成・分解マクロ		
16874			
16875	<code>ERCD(mercd, sercd)</code>	メインエラーコードmercdとサブエラーコードsercdか	
16876		ら、エラーコードを生成するためのマクロ	
16877			
16878	<code>MERCD(ercd)</code>	エラーコードercdからメインエラーコードを抽出する	
16879		ためのマクロ	
16880	<code>SERCD(ercd)</code>	エラーコードercdからサブエラーコードを抽出するた	
16881		めのマクロ	
16882			
16883	(6) アクセス許可パターン生成マクロ		
16884			
16885	<code>TACP(domid)</code>	domidで指定される保護ドメインに属する処理単位	
16886		みにアクセスを許可するアクセス許可パターン	
16887			
16888	5.4.3 カーネル共通定数		
16889			
16890	(1) オブジェクト属性		
16891			
16892	<code>TA_TPRI</code>	<code>0x01U</code>	タスクの待ち行列をタスクの優先度順に
16893			
16894	(2) 保護ドメインID		
16895			
16896	<code>TDOM_SELF</code>	<code>0</code>	自タスクの属する保護ドメイン
16897	<code>TDOM_KERNEL</code>	<code>-1</code>	カーネルドメイン
16898	<code>TDOM_NONE</code>	<code>-2</code>	無所属 (保護ドメインに属さない)
16899			
16900	(3) その他のカーネル共通定数		

16901			
16902	TCLS_SELF	0	自タスクの属するクラス
16903			
16904	TPRC_NONE	0	割付けプロセッサの指定がない
16905	TPRC_INI	0	初期割付けプロセッサ
16906			
16907	TSK_SELF	0	自タスク指定
16908	TSK_NONE	0	該当するタスクがない
16909			
16910	TPRI_SELF	0	自タスクのベース優先度の指定
16911	TPRI_INI	0	タスクの起動時優先度の指定
16912			
16913	TIPM_ENAALL	0	割込み優先度マスク全解除
16914			
16915	5.4.4 カーネル共通マクロ		
16916			
16917	(1) オブジェクト属性を作るマクロ		
16918			
16919	TA_DOM(domid)	domidで指定される保護ドメインに属する	
16920	TA_CLS(clsid)	clsidで指定されるクラスに属する	
16921			
16922	(2) サービスコールの呼出し方法を指定するマクロ		
16923			
16924	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	
16925		て呼び出すための名称	
16926			
16927	5.4.5 カーネルの機能毎の定数		
16928			
16929	(1) タスク管理機能		
16930			
16931	TA_ACT	0x02U	タスクの生成時にタスクを起動する
16932	TA_RSTR	0x04U	生成するタスクを制約タスクとする
16933	TA_FPU		FPUレジスタをコンテキストに含める
16934			
16935	TTS_RUN	0x01U	実行状態
16936	TTS_RDY	0x02U	実行可能状態
16937	TTS_WAI	0x04U	待ち状態
16938	TTS_SUS	0x08U	強制待ち状態
16939	TTS_WAS	0x0cU	二重待ち状態
16940	TTS_DMT	0x10U	休止状態
16941			
16942	TTW_SLP	0x0001U	起床待ち
16943	TTW_DLY	0x0002U	時間経過待ち
16944	TTW_SEM	0x0004U	セマフォの資源獲得待ち
16945	TTW_FLG	0x0008U	イベントフラグ待ち
16946	TTW_SDTQ	0x0010U	データキューへの送信待ち
16947	TTW_RDTQ	0x0020U	データキューからの受信待ち
16948	TTW_SPDQ	0x0100U	優先度データキューへの送信待ち
16949	TTW_RPDQ	0x0200U	優先度データキューからの受信待ち
16950	TTW_MBX	0x0040U	メールボックスからの受信待ち

16951	TTW_MTX	0x0080U	ミューテックスのロック待ち状態
16952	TTW_MPF	0x2000U	固定長メモリブロックの獲得待ち
16953			
16954	TA_FPUの値は、ターゲット定義とする.		
16955			
16956	(3) タスク例外処理機能		
16957			
16958	TTEX_ENA	0x01U	タスク例外処理許可状態
16959	TTEX_DIS	0x02U	タスク例外処理禁止状態
16960			
16961	(4) 同期・通信機能		
16962			
16963	イベントフラグ		
16964			
16965	TA_WMUL	0x02U	複数のタスクが待つのを許す
16966	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
16967			
16968	TWF_ORW	0x01U	イベントフラグのOR待ちモード
16969	TWF_ANDW	0x02U	イベントフラグのAND待ちモード
16970			
16971	メールボックス		
16972			
16973	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
16974			
16975	スピンロック		
16976			
16977	TSPN_UNL	0x01U	取得されていない状態
16978	TSPN_LOC	0x02U	取得されている状態
16979			
16980	(6) 時間管理機能		
16981			
16982	周期ハンドラ		
16983			
16984	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
16985	TA_PHS	0x04U	周期ハンドラを生成した時刻を基準時刻とする
16986			
16987	TCYC_STP	0x01U	周期ハンドラが動作していない状態
16988	TCYC_STA	0x02U	周期ハンドラが動作している状態
16989			
16990	アラームハンドラ		
16991			
16992	TALM_STP	0x01U	アラームハンドラが動作していない状態
16993	TALM_STA	0x02U	アラームハンドラが動作している状態
16994			
16995	オーバランハンドラ		
16996			
16997	TOVR_STP	0x01U	オーバランハンドラが動作していない状態
16998	TOVR_STA	0x02U	オーバランハンドラが動作している状態
16999			
17000	(8) メモリオブジェクト管理機能		

17001			
17002	TA_NOWRITE	0x01U	書込みアクセス禁止
17003	TA_NOREAD	0x02U	読出しアクセス禁止
17004	TA_EXEC	0x04U	実行アクセス許可
17005	TA_MEMINI	0x08U	メモリの初期化を行う
17006	TA_MEMPRSV	0x10U	メモリの初期化を行わない
17007	TA_SDATA	0x20U	ショートデータ領域に配置
17008	TA_UNCACHE	0x40U	キャッシュ禁止
17009	TA_IODEV	0x80U	周辺デバイスの領域
17010	TA_WTHROUGH		ライトスルーキャッシュを用いる
17011			
17012	TPM_WRITE	0x01U	書込みアクセス権のチェック
17013	TPM_READ	0x02U	読出しアクセス権のチェック
17014	TPM_EXEC	0x04U	実行アクセス権のチェック
17015			
17016	TA_WTHROUGHの値は、ターゲット定義とする.		
17017			
17018	(9) 割込み管理機能		
17019			
17020	TA_ENAINT	0x01U	割込み要求禁止フラグをクリア
17021	TA_EDGE	0x02U	エッジトリガ
17022	TA_POSEDGE		ポジティブエッジトリガ
17023	TA_NEGEDGE		ネガティブエッジトリガ
17024	TA_BOTHEDGE		両エッジトリガ
17025	TA_LOWLEVEL		ローレベルトリガ
17026	TA_HIGHLEVEL		ハイレベルトリガ
17027			
17028	TA_NONKERNEL	0x02U	カーネル管理外の割込み
17029			
17030	TA_POSEDGE, TA_NEGEDGE, TA_BOTHEDGE, TA_LOWLEVEL, TA_HIGHLEVELの値は、		
17031	ターゲット定義とする.		
17032			
17033	(10) CPU例外管理機能		
17034			
17035	TA_DIRECT		CPU例外ハンドラを直接呼び出す
17036			
17037	TA_DIRECTの値は、ターゲット定義とする.		
17038			
17039	5.4.6 カーネルの機能毎のマクロ		
17040			
17041	(1) タスク管理機能		
17042			
17043	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
17044			STK_T型の配列の要素数
17045	ROUND_STK_T(sz)		要素数COUNT_STK_T(sz)のSTK_T型の配列のサイズ (sz
17046			を, STK_T型のサイズの倍数になるように大きい方に
17047			丸めた値)
17048			
17049	(4) 同期・通信機能		
17050			

17051	TSZ_DTQMB(dtqcnt)	dtqcntで指定した数のデータを格納できるデータ
17052		キュー管理領域のサイズ (バイト数)
17053	TCNT_DTQMB(dtqcnt)	dtqcntで指定した数のデータを格納できるデータ
17054		キュー管理領域を確保するために必要なMB_T型の配
17055		列の要素数
17056		
17057	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
17058		タキュー管理領域のサイズ (バイト数)
17059	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
17060		タキュー管理領域を確保するために必要なMB_T型の
17061		配列の要素数
17062		
17063	(5) メモリプール管理機能	
17064		
17065	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
17066		リプール領域を確保するために、固定長メモリブロッ
17067		ク1つあたりに必要なMPF_T型の配列の要素数を求め
17068		るマクロ
17069	ROUND_MPF_T(blksz)	要素数COUNT_MPF_T(blksz)のMPF_T型の配列のサイズ
17070		(blkszを、MPF_T型のサイズの倍数になるように大き
17071		い方に丸めた値)
17072		
17073	TSZ_MPFMB(blkent)	blkentで指定した数の固定長メモリブロックを管理
17074		することができる固定長メモリプール管理領域のサ
17075		イズ (バイト数)
17076	TCNT_MPFMB(blkent)	blkentで指定した数の固定長メモリブロックを管理
17077		することができる固定長メモリプール管理領域を確
17078		保するために必要なMB_T型の配列の要素数
17079		
17080	5.5 構成マクロ	
17081		
17082	5.5.1 TOPPERS共通構成マクロ	
17083		
17084	(1) 相対時間の範囲	
17085		
17086	TMAX_RELTIM	相対時間に指定できる最大値
17087		
17088	5.5.2 カーネル共通構成マクロ	
17089		
17090	(1) サポートする機能	
17091		
17092	TOPPERS_SUPPORT_PROTECT	保護機能対応のカーネル
17093	TOPPERS_SUPPORT_MULTI_PRC	マルチプロセッサ対応のカーネル
17094	TOPPERS_SUPPORT_DYNAMIC_CRE	動的生成対応のカーネル
17095		
17096	(2) 優先度の範囲	
17097		
17098	TMIN_TPRI	タスク優先度の最小値 (=1)
17099	TMAX_TPRI	タスク優先度の最大値
17100		

17101	(3) プロセッサの数	
17102		
17103	TNUM_PRCID	プロセッサの数
17104		
17105	(4) 特殊な役割を持ったプロセッサ	
17106		
17107	TOPPERS_MASTER_PRCID	マスタプロセッサのID番号
17108	TOPPERS_SYSTIM_PRCID	システム時刻管理プロセッサのID番号
17109		
17110	(5) タイマ方式	
17111		
17112	TOPPERS_SYSTIM_LOCAL	ローカルタイマ方式の場合にマクロ定義
17113	TOPPERS_SYSTIM_GLOBAL	グローバルタイマ方式の場合にマクロ定義
17114		
17115	(6) バージョン情報	
17116		
17117	TKERNEL_MAKER	カーネルのメーカコード (=0x0118)
17118	TKERNEL_PRID	カーネルの識別番号
17119	TKERNEL_SPVER	カーネル仕様のバージョン番号
17120	TKERNEL_PRVER	カーネルのバージョン番号
17121		
17122	5.5.3 カーネルの機能毎の構成マクロ	
17123		
17124	(1) タスク管理機能	
17125		
17126	TMAX_ACTCNT	タスクの起動要求キューイング数の最大値
17127		
17128	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)
17129		
17130		
17131	(2) タスク付属同期機能	
17132		
17133	TMAX_WUPCNT	タスクの起床要求キューイング数の最大値
17134		
17135	(3) タスク例外処理機能	
17136		
17137	TBIT_TEXPTN	タスク例外要因のビット数 (TEXPTNの有効ビット数)
17138		
17139	(4) 同期・通信機能	
17140		
17141	セマフォ	
17142		
17143	TMAX_MAXSEM	セマフォの最大資源数の最大値
17144		
17145	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)
17146		
17147		
17148	イベントフラグ	
17149		
17150	TBIT_FLGPTN	イベントフラグのビット数 (FLGPTNの有効ビット数)

17151		
17152	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）
17153		
17154		
17155		
17156	データキュー	
17157		
17158	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）
17159		
17160		
17161		
17162	優先度データキュー	
17163		
17164	TMIN_DPRI	データ優先度の最小値（=1）
17165	TMAX_DPRI	データ優先度の最大値
17166		
17167	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）
17168		
17169		
17170		
17171	メールボックス	
17172		
17173	TMIN_MPRI	メッセージ優先度の最小値（=1）
17174	TMAX_MPRI	メッセージ優先度の最大値
17175		
17176	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）
17177		
17178		
17179		
17180	ミューテックス	
17181		
17182	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）
17183		
17184		
17185		
17186	スピンロック	
17187		
17188	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）
17189		
17190		
17191		
17192	(5) メモリプール管理機能	
17193		
17194	固定長メモリプール	
17195		
17196	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）
17197		
17198		
17199		
17200	(6) 時間管理機能	

17201		
17202	システム時刻管理	
17203		
17204	TIC_NUME	タイムティックの周期（単位はミリ秒）の分子
17205	TIC_DENO	タイムティックの周期（単位はミリ秒）の分母
17206		
17207	TOPPERS_SUPPORT_GET_UTM	get_utmがサポートされている
17208		
17209	周期ハンドラ	
17210		
17211	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）
17212		
17213		
17214	アラームハンドラ	
17215		
17216		
17217	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）
17218		
17219		
17220		
17221	オーバランハンドラ	
17222		
17223	TMAX_OVRTIM	プロセッサ時間に指定できる最大値
17224		
17225	TOPPERS_SUPPORT_OVRHDR	オーバランハンドラ機能がサポートされている
17226		
17227		
17228	(7) システム状態管理機能	
17229		
17230	なし	
17231		
17232	(8) メモリオブジェクト管理機能	
17233		
17234	TOPPERS_SUPPORT_ATT_MOD	ATT_MOD／ATA_MODがサポートされている
17235	TOPPERS_SUPPORT_ATT_PMA	ATT_PMA／ATA_PMA／att_pmaがサポートされている
17236		
17237		
17238	(9) 割込み管理機能	
17239		
17240	TMIN_INTPRI	割込み優先度の最小値（最高値）
17241	TMAX_INTPRI	割込み優先度の最大値（最低値，=-1）
17242		
17243	TMIN_ISRPRI	割込みサービスルーチン優先度の最小値（=1）
17244	TMAX_ISRPRI	割込みサービスルーチン優先度の最大値
17245		
17246	TOPPERS_SUPPORT_DIS_INT	dis_intがサポートされている
17247	TOPPERS_SUPPORT_ENA_INT	ena_intがサポートされている
17248		
17249	(10) CPU例外管理機能	
17250		

17251 なし

17252

17253 (11) 拡張サービスコール管理機能

17254

17255 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
17256 いカーネルでは、静的APIによって登録された拡張サービ
17257 スコールの数に一致）

17258

17259 (12) システム構成管理機能

17260

17261 なし

17262

17263 5.6 エラーコード一覧

17264

17265 (1) メインエラーコード

17266

17267	E_SYS	-5	システムエラー
17268	E_NOSPT	-9	未サポート機能
17269	E_RSFN	-10	予約機能コード
17270	E_RSATR	-11	予約属性
17271	E_PAR	-17	パラメータエラー
17272	E_ID	-18	不正ID番号
17273	E_CTX	-25	コンテキストエラー
17274	E_MACV	-26	メモリアクセス違反
17275	E_OACV	-27	オブジェクトアクセス違反
17276	E_ILUSE	-28	サービスコール不正使用
17277	E_NOMEM	-33	メモリ不足
17278	E_NOID	-34	ID番号不足
17279	E_NORES	-35	資源不足
17280	E_OBJ	-41	オブジェクト状態エラー
17281	E_NOEXS	-42	オブジェクト未登録
17282	E_QOVR	-43	キューイングオーバーフロー
17283	E_RLWAI	-49	待ち禁止状態または待ち状態の強制解除
17284	E_TMOUT	-50	ポーリング失敗またはタイムアウト
17285	E_DLT	-51	待ちオブジェクトの削除または再初期化
17286	E_CLS	-52	待ちオブジェクトの状態変化
17287	E_WBLK	-57	ノンブロッキング受付け
17288	E_BOVR	-58	バッファオーバーフロー

17289

17290 5.7 機能コード一覧【NGKI4036】

17291

	-0	-1	-2	-3
17295	-0x01 予約	予約	予約	予約
17296	-0x05 act_tsk	iact_tsk	can_act	ext_tsk
17297	-0x09 ter_tsk	chg_pri	get_pri	get_inf
17298	-0x0d slp_tsk	tslp_tsk	wup_tsk	iwup_tsk
17299	-0x11 can_wup	rel_wai	irel_wai	予約
17300	-0x15 dis_wai	idis_wai	ena_wai	iena_wai

17301	-0x19	sus_tsk	rsm_tsk	dly_tsk	予約
17302	-0x1d	ras_tex	iras_tex	dis_tex	ena_tex
17303	-0x21	sns_tex	ref_tex	予約	予約
17304	-0x25	sig_sem	isig_sem	wai_sem	pol_sem
17305	-0x29	twai_sem	予約	予約	予約
17306	-0x2d	set_flg	iset_flg	clr_flg	wai_flg
17307	-0x31	pol_flg	twai_flg	予約	予約
17308	-0x35	snd_dtq	psnd_dtq	ipsnd_dtq	tsnd_dtq
17309	-0x39	fsnd_dtq	ifsnd_dtq	rcv_dtq	prcv_dtq
17310	-0x3d	trcv_dtq	予約	予約	予約
17311	-0x41	snd_pdq	psnd_pdq	ipsnd_pdq	tsnd_pdq
17312	-0x45	rcv_pdq	prcv_pdq	trcv_pdq	予約
17313	-0x49	snd_mbx	rcv_mbx	prcv_mbx	trcv_mbx
17314	-0x4d	loc_mtx	ploc_mtx	tlloc_mtx	unl_mtx
17315	-0x51	snd_mbf	psnd_mbf	tsnd_mbf	rcv_mbf
17316	-0x55	prcv_mbf	trcv_mbf	予約	予約
17317	-0x59	get_mpf	pget_mpf	tget_mpf	rel_mpf
17318	-0x5d	get_tim	get_utm	予約	ref_ovr
17319	-0x61	sta_cyc	stp_cyc	予約	予約
17320	-0x65	sta_alm	ista_alm	stp_alm	istp_alm
17321	-0x69	sta_ovr	ista_ovr	stp_ovr	istp_ovr
17322	-0x6d	sac_sys	ref_sys	rot_rdq	irotd_rdq
17323	-0x71	get_did	予約	get_tid	iget_tid
17324	-0x75	loc_cpu	iloc_cpu	unl_cpu	iunl_cpu
17325	-0x79	dis_dsp	ena_dsp	sns_ctx	sns_loc
17326	-0x7d	sns_dsp	sns_dpn	sns_ker	ext_ker
17327	-0x81	att_mem	det_mem	sac_mem	prb_mem
17328	-0x85	ref_mem	予約	att_pma	予約
17329	-0x89	cfg_int	dis_int	ena_int	ref_int
17330	-0x8d	chg_ipm	get_ipm	予約	予約
17331	-0x91	xsns_dpn	xsns_xpn	予約	予約
17332	-0x95	ref_cfg	ref_ver	予約	予約
17333	-0x99	予約	予約	予約	予約
17334	-0x9d	予約	予約	予約	予約
17335	-0xa1	予約	ini_sem	ini_flg	ini_dtq
17336	-0xa5	ini_pdq	ini_mbx	ini_mtx	ini_mbf
17337	-0xa9	ini_mpf	予約	予約	予約
17338	-0xad	予約	予約	予約	予約
17339	-0xb1	ref_tsk	ref_sem	ref_flg	ref_dtq
17340	-0xb5	ref_pdq	ref_mbx	ref_mtx	ref_mbf
17341	-0xb9	ref_mpf	ref_cyc	ref_alm	ref_isr
17342	-0xbd	ref_spn	予約	予約	予約
17343	-0xc1	acre_tsk	acre_sem	acre_flg	acre_dtq
17344	-0xc5	acre_pdq	acre_mbx	acre_mtx	acre_mbf
17345	-0xc9	acre_mpf	acre_cyc	acre_alm	acre_isr
17346	-0xcd	acre_spn	予約	予約	予約
17347	-0xd1	del_tsk	del_sem	del_flg	del_dtq
17348	-0xd5	del_pdq	del_mbx	del_mtx	del_mbf
17349	-0xd9	del_mpf	del_cyc	del_alm	del_isr
17350	-0xdd	del_spn	予約	予約	予約

17351	-0xe1	sac_tsk	sac_sem	sac_flg	sac_dtq
17352	-0xe5	sac_pdq	予約	sac_mtx	sac_mbf
17353	-0xe9	sac_mpf	sac_cyc	sac_alm	sac_isr
17354	-0xed	sac_spn	予約	予約	予約
17355	-0xf1	def_tex	def_ovr	def_inh	def_exc
17356	-0xf5	def_svc	予約	予約	予約
17357	-0xf9	予約	予約	予約	予約
17358	-0xfd	予約	予約	予約	予約
17359	-0x101	mact_tsk	imact_tsk	mig_tsk	予約
17360	-0x105	msta_cyc	予約	msta_alm	imsta_alm
17361	-0x109	mrot_rdq	imrot_rdq	get_pid	iget_pid
17362	-0x10d	予約	予約	予約	予約
17363	-0x111	loc_spn	iloc_spn	try_spn	itry_spn
17364	-0x115	unl_spn	iunl_spn	予約	予約
17365	-0x119	予約	予約	予約	予約
17366	-0x11d	予約	予約	予約	予約

17367

17368

17369 【μITRON4.0仕様との関係】

17370

17371 サービスコールの機能コードを割り当てなおした.

17372

17373 5.8 カーネルオブジェクトに対するアクセスの種別

17374

17375

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	rcv_dtq	del_dtq	ref_dtq

17401		psnd_dtq	prcv_dtq	ini_dtq	
17402		tsnd_dtq	trcv_dtq	sac_dtq	
17403		fsnd_dtq			
17404					
17405	優先度データキュー	snd_pdq	rcv_pdq	del_pdq	ref_pdq
17406		psnd_pdq	prcv_pdq	ini_pdq	
17407		tsnd_pdq	trcv_pdq	sac_pdq	
17408					
17409	ミューテックス	loc_mtx	-	del_mtx	ref_mtx
17410		ploc_mtx		ini_mtx	
17411		tloc_mtx		sac_mtx	
17412					
17413	スピンロック	loc_spn	-	del_spn	ref_spn
17414		try_spn		sac_spn	
17415		unl_spn			
17416					
17417	固定長メモリプール	get_mpf	rel_mpf	del_mpf	ref_mpf
17418		pget_mpf		ini_mpf	
17419		tget_mpf		sac_mpf	
17420					
17421	周期ハンドラ	sta_cyc	stp_cyc	del_cyc	ref_cyc
17422		msta_cyc		sac_cyc	
17423					
17424	アラームハンドラ	sta_alm	stp_alm	del_alm	ref_alm
17425		msta_alm		sac_alm	
17426					
17427	割込みサービスルーチン	-	-	del_isr	ref_isr
17428				sac_isr	
17429					
17430	システム状態	rot_rdq	loc_cpu	acre_yyy	get_tim
17431		mrot_rdq	unl_cpu	att_mem	get_ipm
17432		dis_dsp	dis_int	att_pma	ref_sys
17433		ena_dsp	ena_int	cfg_int	ref_int
17434			chg_ipm	def_inh	ref_cfg
17435				def_exc	ref_ver
17436				def_svc	
17437				def_ovr	
17438					
17439					
17440	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：				
17441					
17442	・ 自タスクへの操作 (ext_tsk, get_inf, slp_tsk, tslp_tsk, dly_tsk,				
17443	dis_tex, ena_tex)				
17444	・ タスク例外状態参照 (sns_tex)				
17445	・ 性能評価用システム時刻の参照 (get_utm)				
17446	・ システム状態参照 (get_tid, get_did, get_pid, sns_ctx, sns_loc,				
17447	sns_dsp, sns_dpn, sns_ker)				
17448	・ CPU例外発生時の状態参照 (xsns_dpn, xsns_xpn)				
17449	・ 拡張サービスコールの呼出し (cal_svc)				
17450					

17451 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：

17452

17453 ・ システム状態のアクセス許可ベクタの設定 (sac_sys)

17454 ・ カーネルの終了 (ext_ker)

17455 ・ 非タスクコンテキスト専用のサービスコール

17456

17457 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：

17458

17459 ・ ミューテックスのロック解除 (unl_mtx)

17460

17461 【補足説明】

17462

17463 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
17464 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
17465 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
17466 ネルドメインのみから呼び出すことができる。

17467

17468 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
17469 としているが、ミューテックスをロックしたタスク以外が呼び出すとE_ILUSEエ
17470 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
17471 護されているとみなすことができる（ミューテックスのロック中にアクセス許
17472 可ベクタを変更した場合の振舞いは異なる）。

17473

17474 【μ ITRON4.0/PX仕様との関係】

17475

17476 get_priは、μ ITRON4.0/PX仕様ではタスクに対する通常操作1としていたのを、
17477 タスクに対する参照操作に変更した。また、get_ipm（μ ITRON4.0/PX仕様では
17478 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
17479 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
17480 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
17481 し、get_timはシステム状態に対する参照操作とした。

17482

17483 5.9 ターゲット定義事項一覧

17484

17485 ・ 割込み優先度の段階数 [NGKI0256]

17486

17487 ・ 割込み番号の付与方法 [NGKI0272]

17488

17489 ・ 割込みハンドラ番号の付与方法 [NGKI0273]

17490

17491 ・ 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
17492 しない割込み番号を設けるか [NGKI0276]

17493

17494 ・ 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
17495 も登録していない場合の振舞い [NGKI0249]

17496

17497 ・ 割込み要求禁止フラグがサポートされているか [NGKI0260] [NGKI0261]

17498

17499 ・ 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]

17500

- 17501 • 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]
- 17502
- 17503 • 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
- 17504 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]
- 17505
- 17506 • 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
- 17507 イレベルトリガかを設定できるか [NGKI0266]
- 17508
- 17509 • 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
- 17510 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かものとする
- 17511 か [M] [NGKI0281]
- 17512
- 17513 • 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
- 17514 場合の設定方法 [NGKI0288]
- 17515
- 17516 • 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）
- 17517 [NGKI0289]
- 17518
- 17519 • 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
- 17520 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
- 17521 方法 [NGKI0292]
- 17522
- 17523 • 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
- 17524 [NGKI0295]
- 17525
- 17526 • 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
- 17527 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]
- 17528
- 17529 • CPU例外ハンドラ番号の付与方法 [NGKI0306]
- 17530
- 17531 • 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
- 17532 [NGKI0314]
- 17533
- 17534 • 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]
- 17535 [NGKI2774]
- 17536
- 17537 • 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
- 17538 か [P] [NGKI0113]
- 17539
- 17540 • 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
- 17541 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]
- 17542
- 17543 • 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
- 17544 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]
- 17545
- 17546 • 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
- 17547 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
- 17548 とした場合に、どのCPU例外ハンドラが起動されるか [P] [NGKI0411]
- 17549
- 17550 • 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

- 17551 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
17552 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
17553 [P] [NGKI0413]
17554
- 17555 • 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
17556 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）
17557 を参照する方法 [P] [NGKI0414]
17558
- 17559 • 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
17560 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]
17561
- 17562 • 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]
17563 [NGKI0423]
17564
- 17565 • 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]
17566
- 17567 • 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
17568 アクセスできるものとするか [P] [NGKI0441]
17569
- 17570 • 使用できるクラスのID番号とその属性 [M] [NGKI0107]
17571
- 17572 • どのプロセッサをマスタプロセッサとするか [M] [NGKI0101]
17573
- 17574 • 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
17575 できるか [M] [NGKI0108]
17576
- 17577 • 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
17578 サとするか [M] [NGKI0111]
17579
- 17580 • int8_t, uint8_t, int64_t, uint64_t, int128_t, uint128_t, float32_t,
17581 double64_tが使用できるか [NGKI0488] [NGKI0490]
17582
- 17583 • ターゲット定義のタスク属性 [NGKI1016]
17584
- 17585 • タスクが用いるスタック領域のサイズの最小値 [NGKI1042]
17586
- 17587 • タスクのシステムスタック領域のサイズの最小値 [P] [NGKI1044]
17588
- 17589 • タスクが用いるスタック領域の先頭番地サイズに対する制約 [NGKI1050]
17590 [NGKI1056]
17591
- 17592 • 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
17593 で確保する方法 [P] [NGKI1059]
17594
- 17595 • タスクのシステムスタック領域の先頭番地サイズに対する制約 [P]
17596 [NGKI1062] [NGKI1065] [NGKI1070]
17597
- 17598 • データキュー管理領域の先頭番地に対する制約 [NGKI1687]
17599
- 17600 • 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]

- 17601
- 17602 • 生成できるスピンロックの数の上限 [M] [NGKI2142]
- 17603
- 17604 • 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、
- 17605 どのプロセッサが最初にロックを取得できるか [M] [NGKI2183]
- 17606
- 17607 • 固定長メモリプール領域の先頭番地に対する制約 [NGKI2249]
- 17608
- 17609 • 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]
- 17610
- 17611 • タイムティックの周期 [NGKI2335]
- 17612
- 17613 • 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]
- 17614 [NGKI2346]
- 17615
- 17616 • get_utmがサポートされているか [NGKI2360]
- 17617
- 17618 • オーバランハンドラ機能がサポートされているか [NGKI2598]
- 17619
- 17620 • 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]
- 17621
- 17622 • ターゲット定義のメモリージョン属性 [P]
- 17623
- 17624 • メモリージョンの先頭番地とサイズに対する制約 [P] [NGKI2768]
- 17625
- 17626 • 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
- 17627 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]
- 17628
- 17629 • 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
- 17630 [P] [NGKI2789]
- 17631
- 17632 • TA_NOWRITEを指定した場合に、TA_SDATAが無視されるか [P] [NGKI2790]
- 17633
- 17634 • 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
- 17635 視されるか [P] [NGKI2792]
- 17636
- 17637 • 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
- 17638 扱うことができないメモリオブジェクト [P] [NGKI2793]
- 17639
- 17640 • ターゲット定義のメモリオブジェクト属性 [P] [NGKI2794]
- 17641
- 17642 • 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
- 17643 に対する制限 [P] [NGKI2831]
- 17644
- 17645 • ATT_MOD／ATA_MODがサポートされているか [P] [NGKI2859]
- 17646
- 17647 • 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
- 17648 定されるメモリオブジェクト属性 [P] [NGKI2850]
- 17649
- 17650 • 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

- 17651 メモリリージョンが定義されている場合でも、共通の標準メモリリージョンに
17652 配置されるセクション [PM] [NGKI3271]
- 17653
- 17654 • 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
17655 録できる数に対する制限 [P] [NGKI2857]
- 17656
- 17657 • 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
17658 ンや登録できる数に対する制限 [P] [NGKI2878]
- 17659
- 17660 • 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
17661 対する制約 [P] [NGKI2880]
- 17662
- 17663 • ATT_PMA／ATA_PMA／att_pmaがサポートされているか [P] [NGKI2903]
17664 [HRPS0156]
- 17665
- 17666 • 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
17667 ンや登録できる数に対する制限 [P] [NGKI2898]
- 17668
- 17669 • 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、
17670 物理アドレス空間における先頭番地に対する制約 [P] [NGKI2900]
- 17671
- 17672 • ターゲット定義の割込み要求ライン属性 [NGKI2945]
- 17673
- 17674 • 割込みハンドラ属性にTA_NONKERNELを指定できるか [NGKI2957]
- 17675
- 17676 • その他のターゲット定義の割込みハンドラ属性 [NGKI2959]
- 17677
- 17678 • 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
17679 れるか [D] [NGKI2980]
- 17680
- 17681 • 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
17682 設定できるか [NGKI2982]
- 17683
- 17684 • 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
17685 イン属性／割込み優先度に対する制限 [NGKI2986]
- 17686
- 17687 • 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]
17688 [NGKI3018]
- 17689
- 17690 • 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
17691 るか [NGKI3020]
- 17692
- 17693 • 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
17694 定義できるか [NGKI3064]
- 17695
- 17696 • 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]
- 17697
- 17698 • 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]
- 17699
- 17700 • 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]

- 17701
- 17702 • 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]
- 17703 [NGKI3077]
- 17704
- 17705 • 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
- 17706 みハンドラ番号 [NGKI3078]
- 17707
- 17708 • 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
- 17709 は定義解除）できない割込みハンドラ [M] [NGKI3079]
- 17710
- 17711 • 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
- 17712 [NGKI3080]
- 17713
- 17714 • dis_intがサポートされているか [NGKI3091]
- 17715
- 17716 • 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
- 17717 をセットできないか [NGKI3087]
- 17718
- 17719 • 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
- 17720 るか [NGKI3089]
- 17721
- 17722 • ena_intがサポートされているか [NGKI3104]
- 17723
- 17724 • 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
- 17725 をクリアできないか [NGKI3100]
- 17726
- 17727 • 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
- 17728 るか [NGKI3102]
- 17729
- 17730 • 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
- 17731 きるか [NGKI3114]
- 17732
- 17733 • ターゲット定義のCPU例外ハンドラ属性 [NGKI3123]
- 17734
- 17735 • 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]
- 17736 [NGKI3148]
- 17737
- 17738 • 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
- 17739 [NGKI3149]
- 17740
- 17741 • 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]
- 17742
- 17743 • 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
- 17744 [NGKI3220] [NGKI3222]
- 17745
- 17746 • 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
- 17747 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]
- 17748
- 17749 • 共有スタック領域のサイズの最小値 [NGKI3255]
- 17750

- 17751 • 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]
- 17752
- 17753 • 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
- 17754 [NGKI3246]
- 17755
- 17756 • 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
- 17757 [NGKI3253]
- 17758

17759 5.10 省略名の元になった英語

17760 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

17763	xxx	元になった英語
17764	-----	
17765	act	activate
17766	aid	automatically assigned ID
17767	ata	attach with access control vector
17768	att	attach
17769	cal	call
17770	can	cancel
17771	cfg	configure
17772	chg	change
17773	clr	clear
17774	cre	create
17775	def	define
17776	del	delete
17777	det	detach
17778	dis	disable
17779	dly	delay
17780	ena	enable
17781	epr	execution priority
17782	ext	exit
17783	get	get
17784	ini	initialize
17785	lnk	link
17786	loc	lock
17787	mig	migrate
17788	pol	poll
17789	prb	probe
17790	ras	raise
17791	rcv	receive
17792	ref	reference
17793	rel	release
17794	rot	rotate
17795	rsm	resume
17796	sac	set access control vector
17797	set	set
17798	sig	signal
17799	slp	sleep
17800	snd	send

17801	sns	sense
17802	sta	start
17803	stp	stop
17804	sus	suspend
17805	ter	terminate
17806	try	try
17807	unl	unlock
17808	wai	wait
17809	wup	wake up

17810

17811 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

17812

yyy	元になった英語
-----	---------

17814

17815	act	activation
17816	alm	alarm handler
17817	cfg	configuration
17818	cpu	CPU
17819	ctx	context
17820	cyc	cyclic handler
17821	did	domain ID
17822	dpn	dispatch pending
17823	dsp	dispatch
17824	dtq	data queue
17825	exc	exception
17826	flg	eventflag
17827	ics	interrupt context stack
17828	inf	information
17829	inh	interrupt handler
17830	ini	initilization
17831	int	interrupt
17832	ipm	interrupt priority mask
17833	isr	interrupt service routine
17834	ker	kernel
17835	loc	lock
17836	mbf	message buffer
17837	mbx	mailbox
17838	mpf	fixed-sized memory pool
17839	mem	memory
17840	mod	module
17841	mtx	mutex
17842	ovr	overflow handler
17843	pdq	priority data queue
17844	pid	processor ID
17845	pma	physical memory area
17846	pri	priority
17847	rdq	ready queue
17848	reg	region
17849	sec	section
17850	sem	semaphore

17851	srg	standard memory region
17852	spn	spin lock
17853	stk	stack
17854	sys	system
17855	svc	service call
17856	ter	termination
17857	tex	task exception
17858	tid	task ID
17859	tim	time
17860	tsk	task
17861	utm	time in micro second
17862	ver	version
17863	wai	wait
17864	wup	wake up
17865	xpn	exception pending

17866

17867 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

17868

17869 z 元になった英語

17870 -----

17871	a	automatic ID assignment
17872	f	force
17873	i	interrupt
17874	m	multiprocessor
17875	p	poll
17876	t	timeout
17877	x	exception

17878

17879 5.11 バージョン履歴

17880

17881	2008年11月19日	Release 1.0.0	最初のリリース
17882	2009年5月8日	Release 1.1.0	FMPカーネルに関する記述が完成
17883	2010年5月10日	Release 1.2.0	
17884	2011年5月5日	Release 1.3.0	HRP2カーネルに関する記述が完成
17885	2012年5月16日	Release 1.4.0	SSPカーネルに関する記述が完成
17886	2012年12月19日	Release 1.5.0	HRP2カーネルの仕様変更を反映

17887

17888 以上

アプリケーションシステム

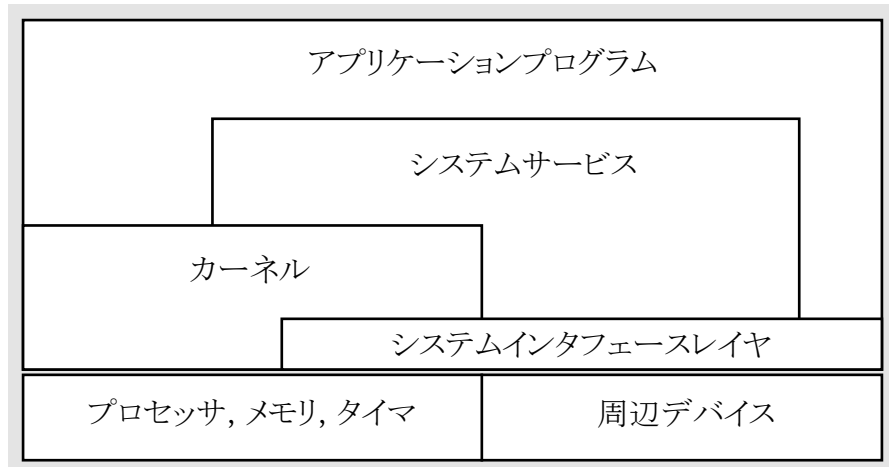


図2-1. 想定するソフトウェア構成

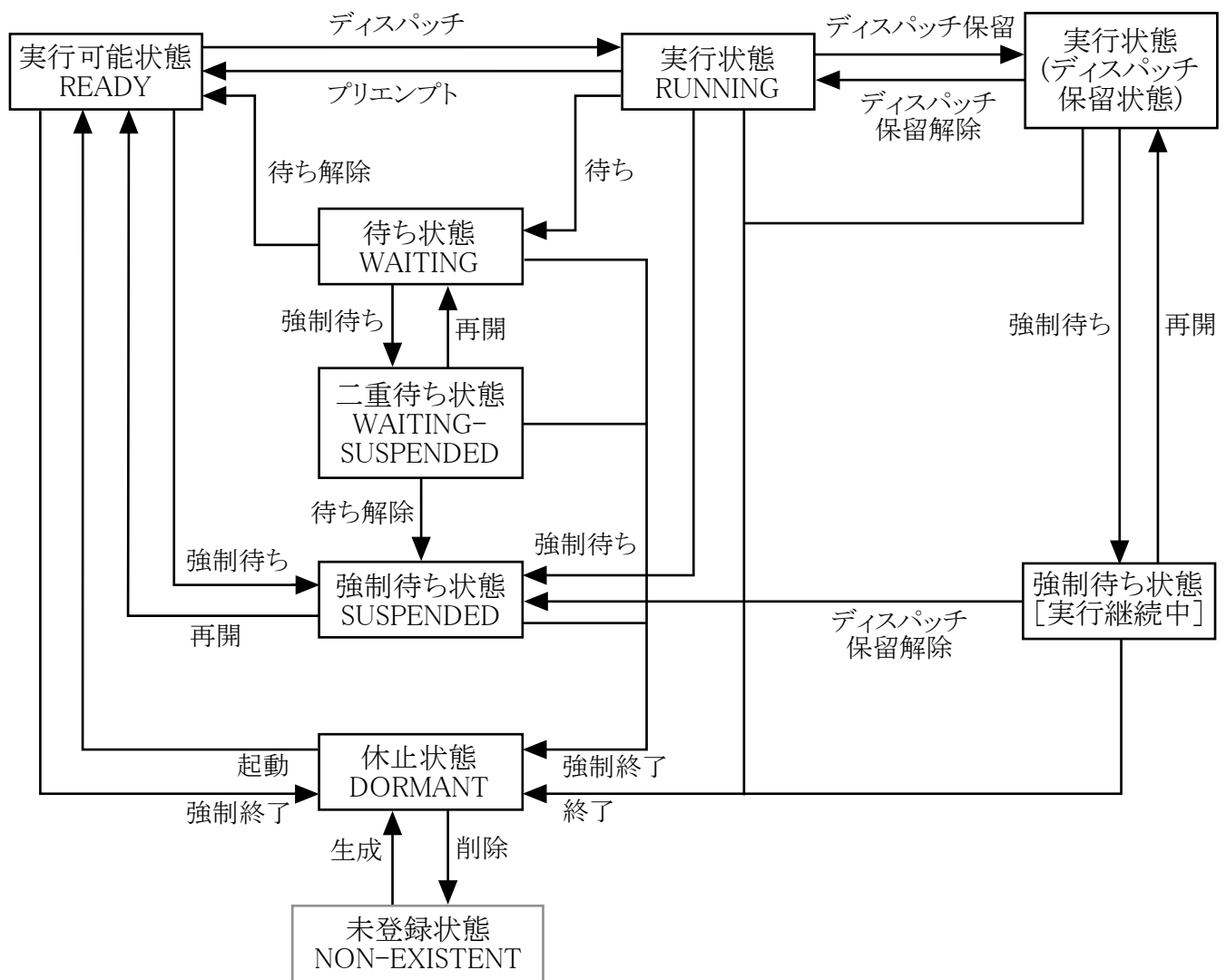


図2-3. 過渡的な状態も含めたタスクの状態遷移

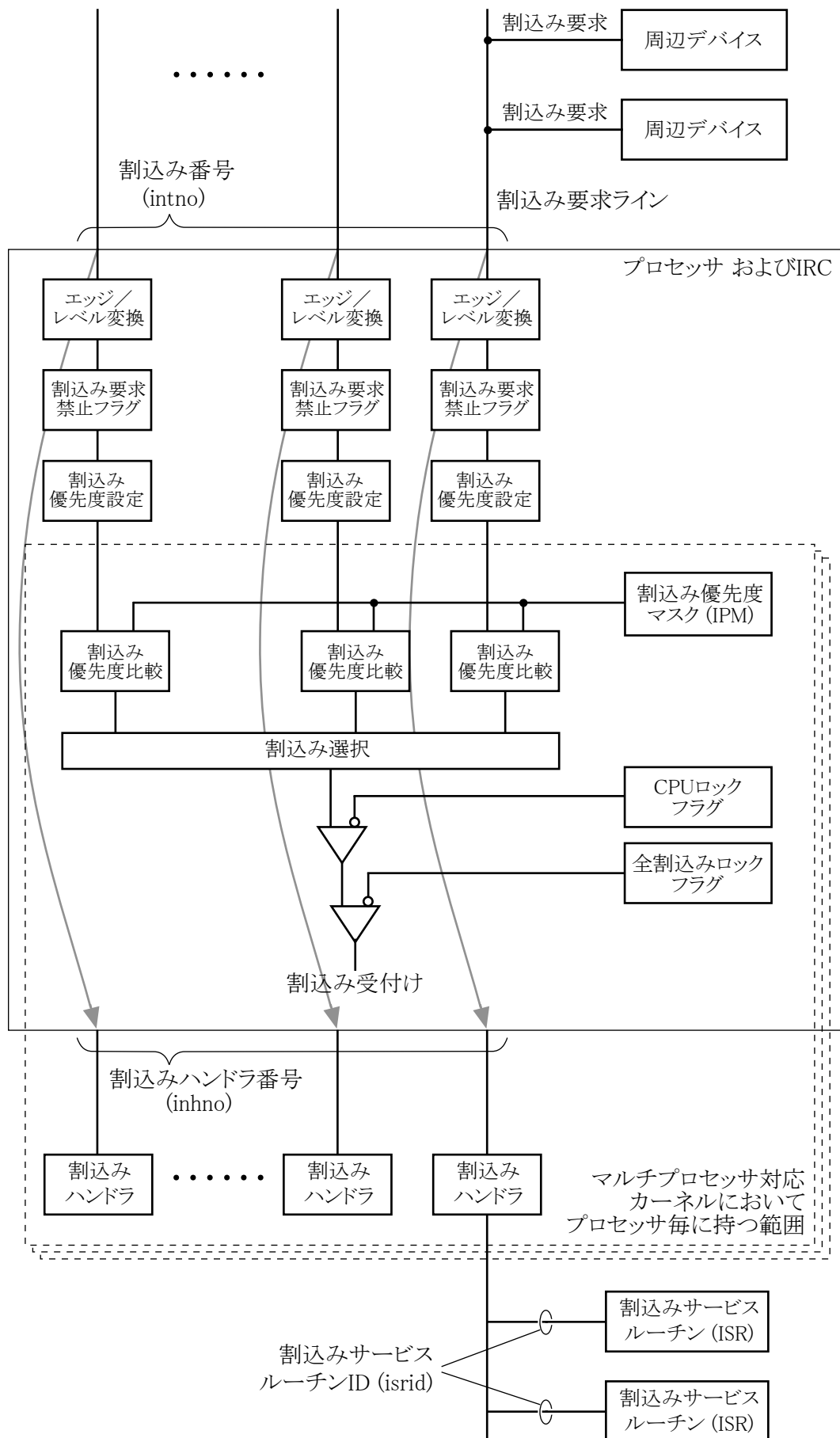


図2-4. TOPPERS標準割り込み処理モデルの概念図

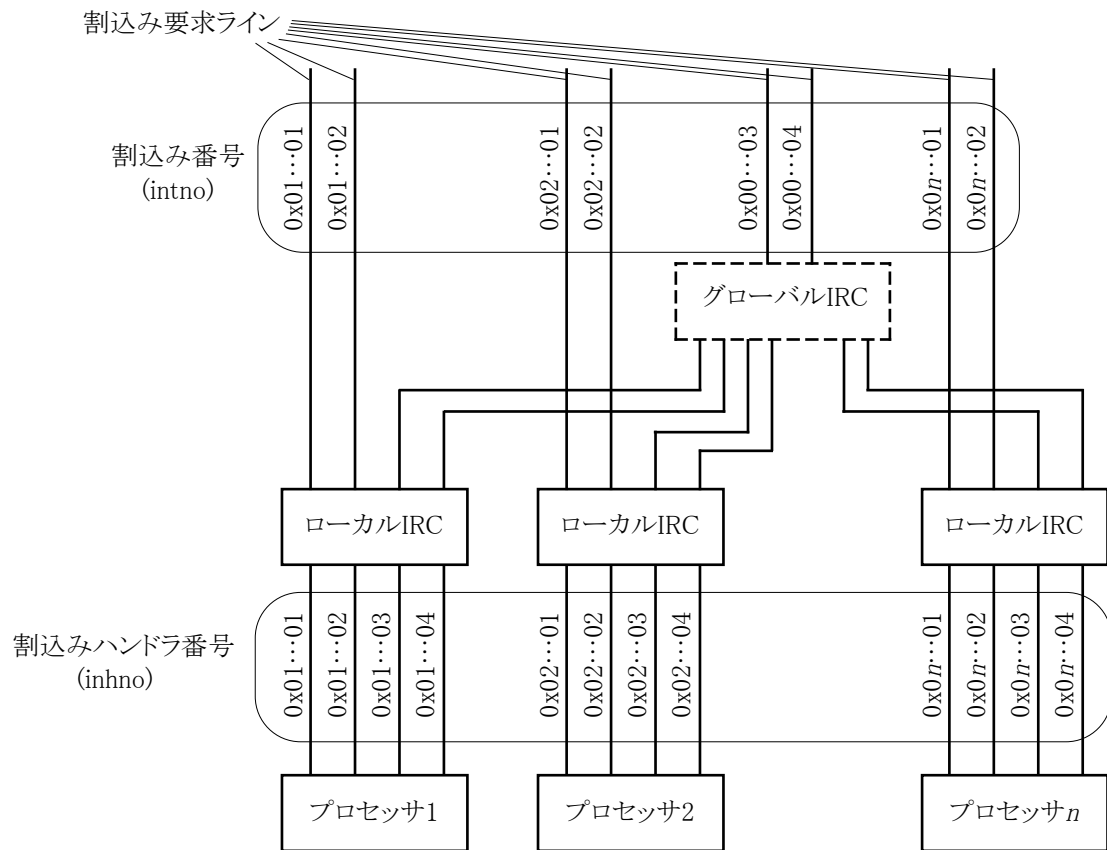


図2-5. マルチプロセッサ対応カーネルにおける割り込み番号と割り込みハンドラ番号

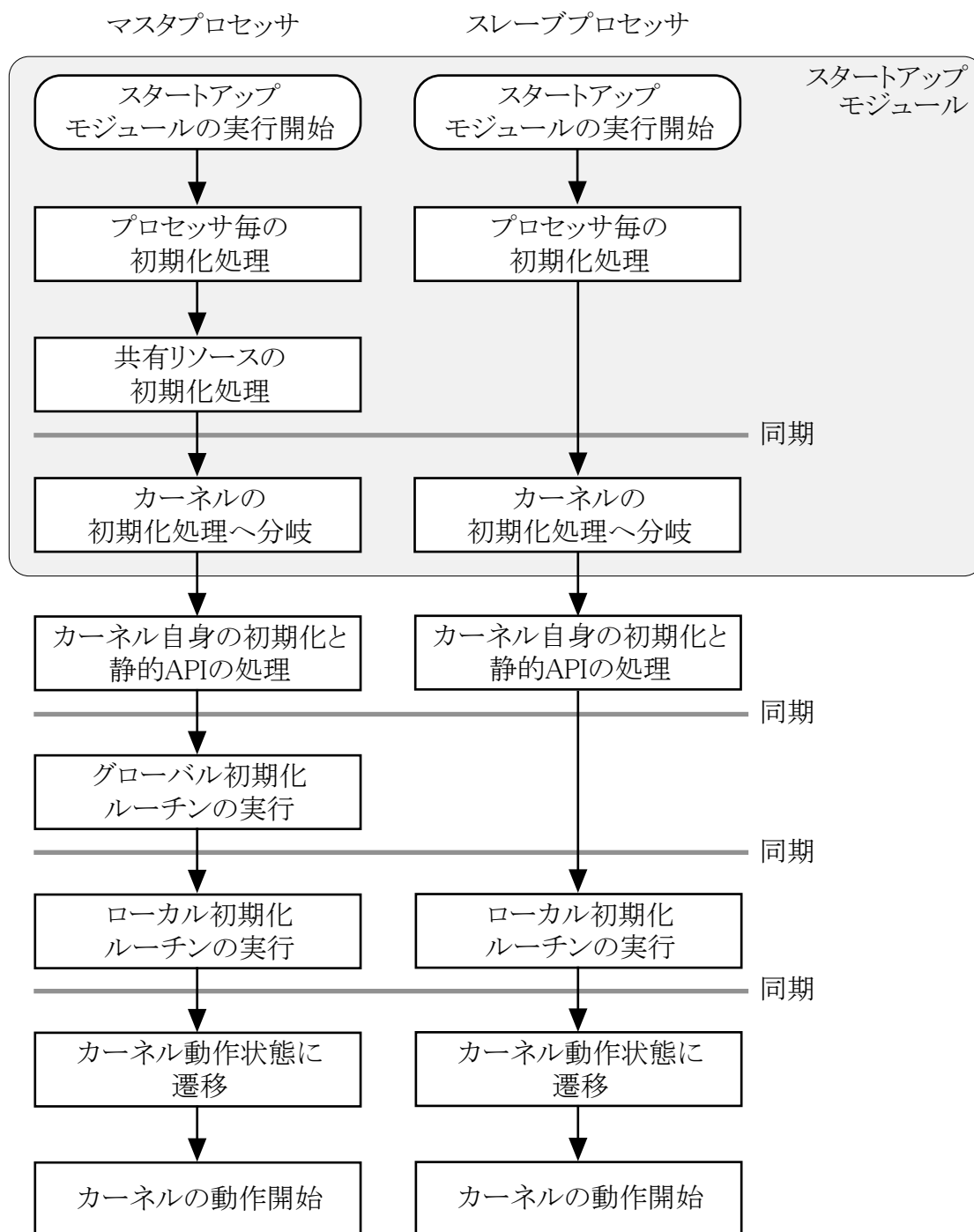


図2-6. マルチプロセッサ対応カーネルにおけるシステム初期化の流れ

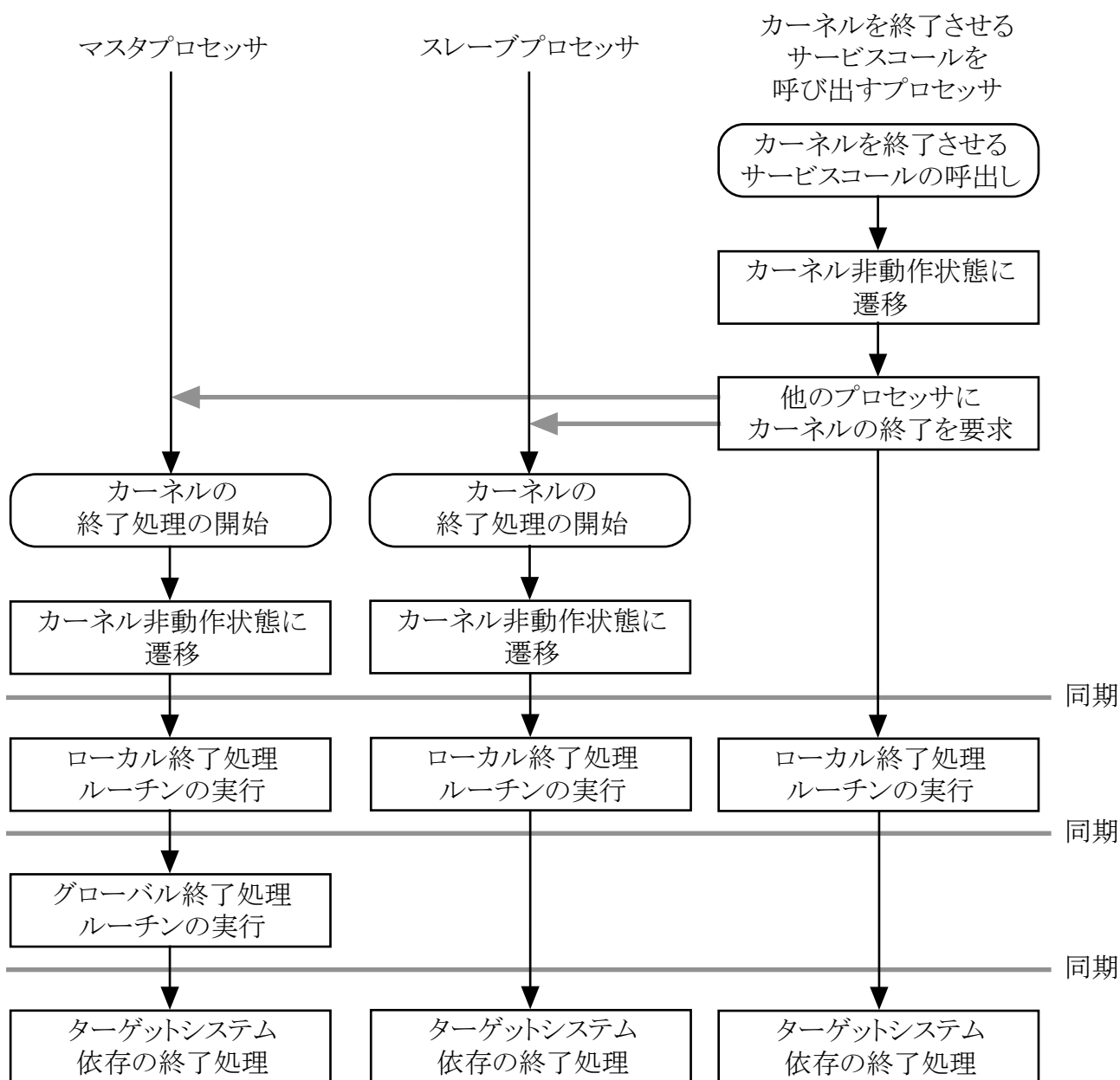


図2-7. マルチプロセッサ対応カーネルにおけるシステム終了処理の流れ

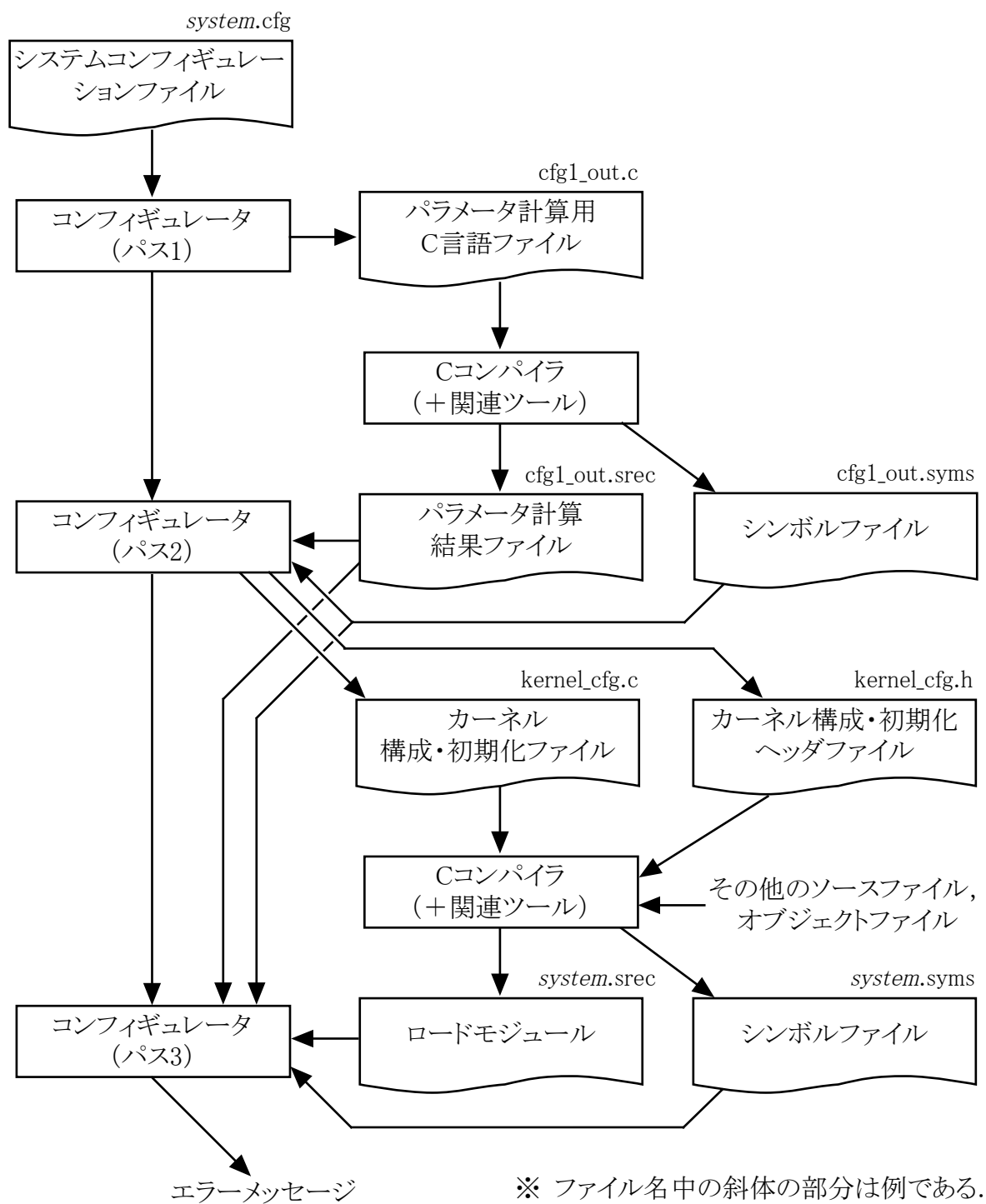
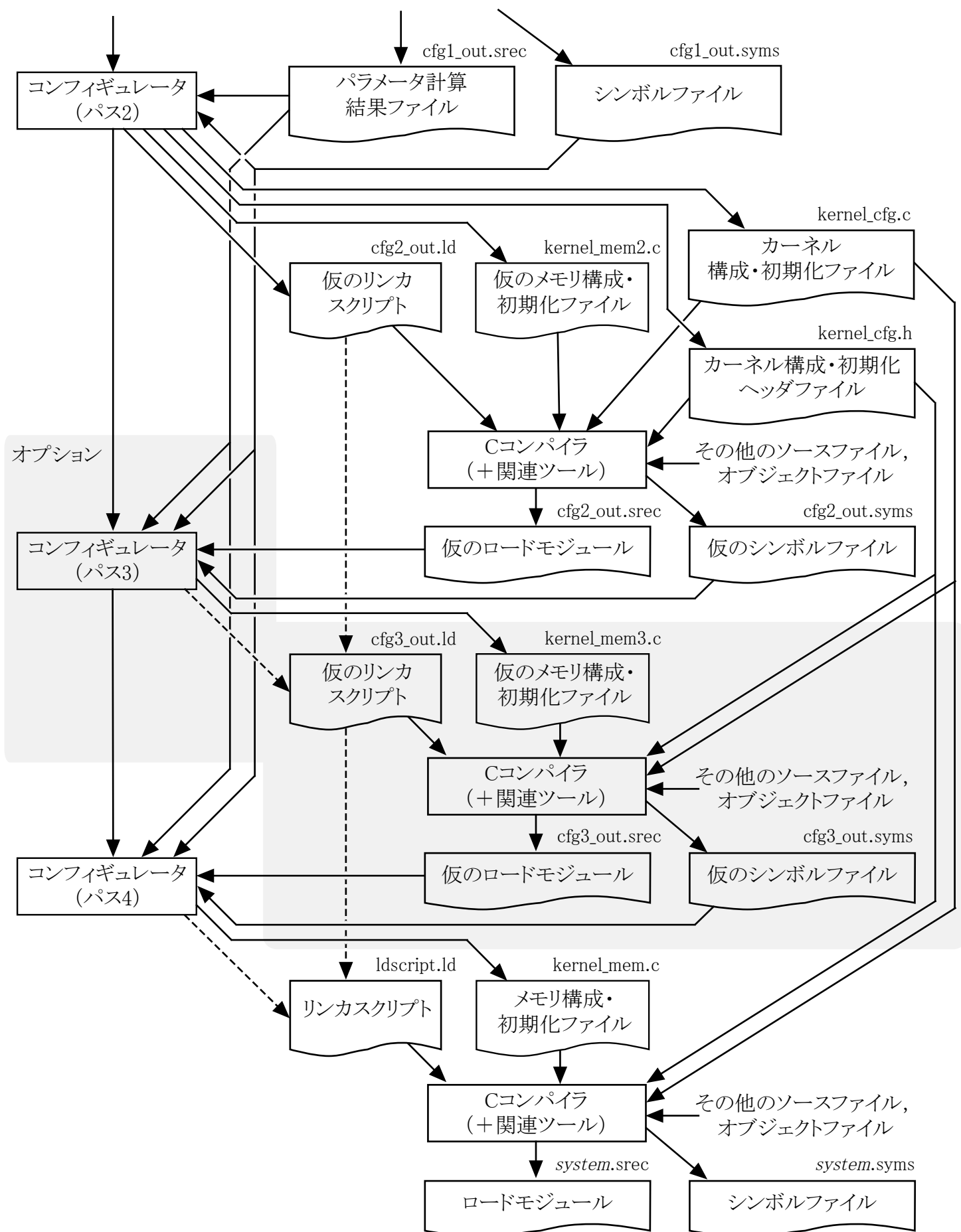


図2-8. コンフィギュレータの処理モデル



※ 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