

# 1. カーネルAPI仕様

この章では、カーネルのAPI仕様について規定する。

## 【μITRON4.0仕様との関係】

TOPPERS共通データ型に従い、パラメータのデータ型を次の通り変更した。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 → int_t  
UINT → uint_t  
VP → void *  
VP_INT → intptr_t
```

## 【μITRON4.0/PX仕様との関係】

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

## 1.1. タスク管理機能

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

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

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

- 属する保護ドメイン（保護機能対応カーネルの場合）
- 属するクラス（マルチプロセッサ対応カーネルの場合）

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

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

現在優先度については、「4.4.6 ミューテックス」の節を参照すること。

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

タスクの生成時に

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

次回起動時の割付けプロセッサは、マルチプロセッサ対応カーネルにおいて、タスクが次に起動される時に割り付けられるプロセッサで、タスクの生成時に未設定の状態に初期化される【NGKI1007】。タスクの起動時に、次回起動時の割付けプロセッサが設定されていれば、タスクの割付けプロセッサがそのプロセッサに変更され、次回起動時の割付けプロセッサは未設定の状態に戻される【NGKI1008】。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同じプロセッサで実行される）【NGKI1009】。

保護機能対応カーネルにおいては、スタック領域の扱いは、ユーザタスクとシステムタスクで異なる。ユーザタスクのスタック領域は、ユーザ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であり、ユーザスタック領域と呼ぶ【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、カーネルの用いるオブジェクト管理領域と同様に扱われる【NGKI1011】。

システムスタック領域は、保護機能対応カーネルにおいて、ユーザタスクがサービスコール（拡張サービスコールを含む）を呼び出し、特権モードで実行する間に用いるスタック領域である【NGKI1012】。システムスタック領域は、カーネルの用いるオブジェクト管理領域と同様に扱われる【NGKI1013】。

タスク属性には、次の属性を指定することができる【NGKI1014】。

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

TA\_ACTを指定しない場合、タスクの生成直後には、タスクは休止状態となる【NGKI1014】。

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

TA\_FPU

FPUレジスタをコンテキストに含める

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

C言語によるタスクの記述形式は次の通り【NGKI1018】。

```
void task(intptr_t exinf)
{
    タスク本体
    ext_tsk();
}
```

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

タスク管理機能に関するカーネル構成マクロは次の通り。

TMAX\_ACTCNT

タスクの起動要求キューイング数の最大値【NGKI1021】

TNUM\_TSKID

登録できるタスクの数（動的生成対応でないカーネルでは、静的APIによって登録されたタスクの数に一致）  
【NGKI1022】

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、TMAX\_ACTCNTは1に固定されている【ASPS0101】。また、制約タスクはサポートしていない【ASPS0102】。ただし、制約タスク拡張パッケージを用いると、制約タスクの機能を追加することができる【ASPS0103】。

#### 【TOPPERS/FMPカーネルにおける規定】

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

## 【TOPPERS/HRP2カーネルにおける規定】

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

## 【TOPPERS/SSPカーネルにおける規定】

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

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タック領域を共有しており、タスク毎にスタック領域の情報を持たない【SSPS0102】。

SSPカーネルにおける追加機能として、タスクに対して、実行時優先度の情報を持つ【SSPS0103】。  
SSPカーネルにおいては、タスクが起動された後、最初に実行状態になる時に、タスクのベース優先度が、タスクの実行時優先度に設定される【SSPS0104】。実行時優先度の機能は、起動時優先度よりも高い優先度でタスクを実行することで、同時期に共有スタック領域を使用している状態になるタスクの組み合わせを限定し、スタック領域を節約するための機能である。

タスクの実行時優先度は、実行時優先度を定義する静的API（DEF\_EPR）によって設定する【SSPS0105】。実行時優先度を定義しない場合、タスクの実行時優先度は、起動時優先度と同じ値に設定される【SSPS0106】。

## 〔実行時優先度によるスタック領域の節約〕

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

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

## 【μITRON4.0仕様との関係】

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

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

CRE\_TSK タスクの生成 [S] 【NGKI1023】

acre\_tsk タスクの生成 [TD] 【NGKI1024】

## 【静的API】

保護機能対応でないカーネルの場合

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

保護機能対応カーネルの場合

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

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

#### 【C言語API】

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

#### 【パラメータ】

|                |                  |  |
|----------------|------------------|--|
| ID<br>T_CTSK * | tskid<br>pk_ctsk | 生成するタスクのID番号 (CRE_TSKの場合)<br>タスクの生成情報を入ったパケットへのポイン<br>タ (静的APIを除く) |
|----------------|------------------|--|

#### \*タスクの生成情報 (パケットの内容)

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

#### 【リターンパラメータ】

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

## 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し [s] 【NGKI1026】<br>・CPUロック状態からの呼出し [s] 【NGKI1027】   |
| E_RSATR | 予約属性<br>・tskatrが無効 【NGKI1028】<br>・属する保護ドメインの指定が有効範囲外または無所属 [sP] 【NGKI1029】<br>・保護ドメインの囲みの中に記述されていない [SP] 【NGKI1030】<br>・属するクラスの指定が有効範囲外 [sM] 【NGKI1031】<br>・クラスの囲みの中に記述されていない [SM] 【NGKI1032】 |
| E_PAR   | パラメータエラー<br>・taskがプログラムの先頭番地として正しくない 【NGKI1033】<br>・itskpriが有効範囲外 【NGKI1034】<br>・その他の条件については機能の項を参照   |
| E_OACV  | オブジェクトアクセス違反<br>・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】  |
| E_MACV  | メモリアクセス違反<br>・pk_ctskが指すメモリ領域への読み出しが許可されていない [sP] 【NGKI1036】  |
| E_NOID  | ID番号不足<br>・割り付けられるタスクIDがない [sD] 【NGKI1037】  |
| E_NOMEM | メモリ不足<br>・スタック領域が確保できない 【NGKI1038】<br>・システムスタック領域が確保できない [P] 【NGKI1039】   |
| E_OBJ   | オブジェクト状態エラー<br>・tskidで指定したタスクが登録済み (CRE_TSKの場合) 【NGKI1040】<br>・その他の条件については機能の項を参照   |

## 【機能】

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

まず、stkとstkszからタスクが用いるスタック領域が設定される【NGKI1041】。

ただし、保護機能対応カーネルで、生成するタスクがシステムタスクの場合は、

スタック領域の設定にsstkszも用いられる。stkszに0以下の値を指定した時や、設定されるスタック領域のサイズがターゲット定義の最小値よりも小さくなる時には、

E\_PARエラーとなる【NGKI1042】。

また、保護機能対応カーネルで、生成するタスクがユーザタスクの場合には、

sstkszからシステムスタック領域が設定される【NGKI1043】。この場合、

0以下の値を指定した時や、ターゲット定義の最小値よりも小さい値を

E\_PARエラーとなる【NGKI1044】。

次に、生成されたタスクに対してタスク生成時に用いべき初期化処理が行われ、

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

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

#### 〔stkにNULLを指定した場合〕

stkをNULLとした場合、stkszで指定したサイズのスタック領域が、コンフィギュレータまたはカーネルにより確保される【NGKI1049】。stkszにターゲット定義の制約に合致しないサイズを指定した時には、ターゲット定義の制約に合致するように大きい方に丸めたサイズで確保される【NGKI1050】。

保護機能対応カーネルにおいて、生成するタスクがユーザタスクの場合、コンフィギュレータ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  
ク領域）は、「2.11.6  
ユーザタスクのユーザスタッ  
ク領域」の節の規定に従つ  
て、メモリオブジェクトとしてカーネルに登録される【NGKI1051】。

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

#### 〔stkにNULL以外を指定した場合〕

stkにNULL以外を指定した場合、stkとstkszで指定したスタック領域は、アプリケーションで確保しておく必要がある【NGKI1055】。スタック領域をアプリケーションで確保する方法については、「2.15.3  
カーネル共通マクロ」の節を参照すること。その方法に従わず、stkやstkszにターゲット定義の制約に合致しない先頭番地やサイズを指定した時には、E\_PARエラーとなる【NGKI1056】。

保護機能対応カーネルにおいて、生成するタスクがシステムタスクの場合、stkszで指定したスタック領域がカーネル専用のメモリオブジェクトに含まれる場合、stkとE\_OBJエラーとなる【NGKI1057】。

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

## 〔sstkとsstkszの扱い〕

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

生成するタスクがユーザタスクの場合の扱いは次の通り。

sstk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sstkszにターゲット定義の制約に合致しないサイズを指定した時には、ターゲット定義の制約に合致するように大きい方に丸めたサイズで確保される【NGKI1062】。sstkszの記述も省略した場合には、ターゲット定義のデフォルトのサイズで確保される【NGKI1063】。

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リケーションで確保する方法については、「2.15.3 カーネル共通マクロ」の節を参照すること。その方法に従わず、sstkやsstkszにターゲット定義の制約に合致しない先頭番地やサイズを指定した時には、E\_PARエラーとなる【NGKI1065】。また、stkとstkszで指定したシステムスタック領域がカーネル専用のメモリオブジェクトに含まれない場合、E\_OBJエラーとなる【NGKI1066】。

生成するタスクがシステムタスクの場合の扱いは次の通り。

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

sstkszに0以外の値を指定した場合で、stkがNULLの場合には、コンフィギュレータまたはカーネルにより確保されるスタック領域のサイズに、sstkszが加えられた値が、ターゲット定義の制約に合致しないサイズになる時には、ターゲット定義の制約に合致するように大きい方に丸めたサイズで確保される【NGKI1069】。sstkszにsstkszを加えた値が、ターゲット定義の制約に合致するように大きい方に丸めたサイズで確保される【NGKI1070】。

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

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

## 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、CRE\_TSKのみをサポートする【ASPS0104】。ただし、動的生成機能拡張パッケージでは、acre\_tskもサポートする【ASPS0105】。

## 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、CRE\_TSKのみをサポートする【FMP0103】。

## 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、CRE\_TSKのみをサポートする【HRPS0103】。

動的生成機能拡張パッケージでは、acre\_tskもサポートする【HRPS0175】。ただし、生成するタスクがユーザタスクの場合、stkにNULLが指定されるとカーネルがスタック領域を確保する機能はサポートしない。stkにNULLを指定した場合E\_NOSPTエラーとなる【HRPS0176】。

には、

#### 【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、CRE\_TSKのみをサポートする【SSPS0107】。

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できない。設定した場合には、コンフィギュレータがE\_PARエラーを報告する【SSPS0109】。

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】。

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、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、タスク毎にスタック領域を持たないためである。stkにNULL以外を指定した場合には、E\_PARエラーとなる【SSPS0112】。

共有スタック領域の設定方法については、DEF\_STKの項を参照すること。

#### 【μITRON4.0仕様との関係】

taskのデータ型をTASKに、stkのデータ型をSTK\_T \*に変更した。COUNT\_STK\_TとROUND\_STK\_Tを新設し、スタック領域をアプリケーションで確保する方法を規定した。

#### 【μITRON4.0/PX仕様との関係】

sstkのデータ型をSTK\_T \*に変更した。システムスタック領域をアプリケーションで確保する方法を規定した。

#### 【未決定事項】

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。

#### 【仕様決定の理由】

保護機能対応カーネルにおいて、sstkszおよびsstkの記述は省略することができることとしたのは、保護機能対応でないカーネル用のシステムコンフィギュレーションファイルを、保護機能対応カーネルにも変更なしに使えるようにするためにある。

AID\_TSK 割付け可能なタスクIDの数の指定 [SD] 【NGKI1073】

#### 【静的API】

```
AID_TSK(uint_t notsk)
```

#### 【パラメータ】

|        |       |               |
|--------|-------|---------------|
| uint_t | notsk | 割付け可能なタスクIDの数 |
|--------|-------|---------------|

#### 【エラーコード】

|         |                                     |
|---------|-------------------------------------|
| E_RSATR | 予約属性                                |
|         | ・保護ドメインの囲みの中に記述されている [P] 【NGKI3428】 |
|         | ・クラスの囲みの中に記述されていない [M] 【NGKI1075】   |
| E_PAR   | パラメータエラー                            |
|         | ・notskが負の値 【NGKI3276】               |

#### 【機能】

notskで指定した数のタスクIDを、タスクを生成するサービスコールによって割付け可能なタスクIDとして確保する【NGKI1076】.

notskは整数定数式パラメータである【NGKI1077】.

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルの動的生成機能拡張パッケージでは、AID\_TSKをサポートする【ASPS0210】.

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルの動的生成機能拡張パッケージでは、AID\_TSKをサポートする【HRPS0211】.

|         |                                   |
|---------|-----------------------------------|
| SAC_TSK | タスクのアクセス許可ベクタの設定 [SP] 【NGKI1078】  |
| sac_tsk | タスクのアクセス許可ベクタの設定 [TPD] 【NGKI1079】 |

#### 【静的API】

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

#### 【C言語API】

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

#### 【パラメータ】

|         |         |                                   |
|---------|---------|-----------------------------------|
| ID      | tskid   | 対象タスクのID番号                        |
| ACVCT * | p_acvct | アクセス許可ベクタを入れたパケットへのポインタ（静的APIを除く） |

\* アクセス許可ベクタ（パケットの内容）

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

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー                                      |
|         | ・非タスクコンテキストからの呼び出し [s] 【NGKI1080】              |
|         | ・CPUロック状態からの呼び出し [s] 【NGKI1081】                |
| E_ID    | 不正ID番号   |
|         | ・tskidが有効範囲外 [s] 【NGKI1082】                    |
| E_RSATR | 予約属性   |
|         | ・対象タスクが属する保護ドメインの囲みの中に記述されていない [S] 【NGKI1083】  |
|         | ・対象タスクが属するクラスの囲みの中に記述されていない [SM] 【NGKI1084】    |
| E_NOEXS | オブジェクト未登録                                      |
|         | ・対象タスクが未登録 【NGKI1085】                          |
| E_OACV  | オブジェクトアクセス違反                                   |
|         | ・対象タスクに対する管理操作が許可されていない [s] 【NGKI1086】         |
| E_MACV  | メモリアクセス違反                                      |
|         | ・p_acvctが指すメモリ領域への読み出しが許可されていない [s] 【NGKI1087】 |
| E_OBJ   | オブジェクト状態エラー                                    |
|         | ・対象タスクは静的APIで生成された [s] 【NGKI1088】              |
|         | ・対象タスクに対してアクセス許可ベクタが設定済み [S] 【NGKI1089】        |

#### 【機能】

tskidで指定したタスク（対象タスク）のアクセス許可ベクタ（4つのアクセス許可パターンの組）を、各パラメータで指定した値に設定する【NGKI1090】。

静的APIにおいては、tskidはオブジェクト識別名、acptn1～acptn4は整数定数式パラメータである【NGKI1091】。

sac\_tskにおいてtskidにTSK\_SELF (=0) を指定すると、自タスクが対象タスクとなる【NGKI1092】。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、SAC\_TSKのみをサポートする【HRPS0104】。ただし、動的生成機能拡張パッケージでは、sac\_tskもサポートする【HRPS0177】。

DEF\_EPR タスクの実行時優先度の定義 [S] 【NGKI1093】

#### 【静的API】

DEF\_EPR(ID tskid, { PRI exepri })

#### 【パラメータ】

|     |        |            |
|-----|--------|------------|
| ID  | tskid  | 対象タスクのID番号 |
| PRI | exepri | タスクの実行時優先度 |

#### 【エラーコード】

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

#### 【サポートするカーネル】

DEF\_EPRは、TOPPERS/SSPカーネルのみがサポートする静的APIである。他のカーネルは、DEF\_EPRをサポートしない【NGKI1096】。

#### 【機能】

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

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

exepriが、対象タスクの起動時優先度よりも低い場合には、E\_ILUSEエラーとなる【NGKI1099】。

#### 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていない静的APIである。

`del_tsk` タスクの削除 [TD] 【NGKI1100】

【C言語API】

```
ER ercd = del_tsk(ID tskid)
```

【パラメータ】

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

【リターンパラメータ】

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

【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー                              |
|         | ・非タスクコンテキストからの呼び出し 【NGKI1101】          |
|         | ・CPUロック状態からの呼び出し 【NGKI1102】            |
| E_ID    | 不正ID番号                                 |
|         | ・tskidが有効範囲外 【NGKI1103】                |
| E_NOEXS | オブジェクト未登録                              |
|         | ・対象タスクが未登録 【NGKI1104】                  |
| E_OACV  | オブジェクトアクセス違反                           |
|         | ・対象タスクに対する管理操作が許可されていない [P] 【NGKI1105】 |
| E_OBJ   | オブジェクト状態エラー                            |
|         | ・対象タスクが休止状態でない 【NGKI1106】              |
|         | ・対象タスクは静的APIで生成された 【NGKI1107】          |

【機能】

tskidで指定したタスク（対象タスク）を削除する。具体的な振舞いは以下の通り。

対象タスクが休止状態である場合には、対象タスクの登録が解除され、そのタスクIDが未使用の状態に戻される【NGKI1108】。また、タスクの生成時にタスクのスタック領域およびシステムスタック領域がカーネルによって確保された場合は、それらのメモリ領域が解放される【NGKI1109】。

スケ

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、`del_tsk`をサポートしない【ASPS0107】。ただし、動的生成機能拡張パッケージでは、`del_tsk`をサポートする【ASPS0108】。

### 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、`del_tsk`をサポートしない【FMP S0105】。

### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、`del_tsk`をサポートしない【HRP S0105】。ただし、動的生成機能拡張パッケージでは、`del_tsk`をサポートする【HRP S0178】。

### 【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、`del_tsk`をサポートしない【SSP S0114】。

```
act_tsk    タスクの起動 [T] 【NGKI1110】  
iact_tsk   タスクの起動 [I] 【NGKI1111】
```

### 【C言語API】

```
ER ercd = act_tsk(ID_tskid)  
ER ercd = iact_tsk(ID_tskid)
```

### 【パラメータ】

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

### 【リターンパラメータ】

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

### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し (act_tskの場合) 【NGKI1112】<br>・タスクコンテキストからの呼出し (iact_tskの場合) 【NGKI1113】<br>・CPUロック状態からの呼出し 【NGKI1114】 |
| E_ID    | 不正ID番号<br>・tskidが有効範囲外 【NGKI1115】   |
| E_NOEXS | オブジェクト未登録<br>・対象タスクが未登録 [D] 【NGKI1116】  |
| E_OACV  | オブジェクトアクセス違反<br>・対象タスクに対する通常操作1が許可されていない (act_tsk<br>の場合) [P] 【NGKI1117】  |
| E_QOVR  | キューイングオーバフロー<br>・条件については機能の項を参照   |

### 【機能】

tskidで指定したタスク（対象タスク）に対して起動要求を行う。具体的な振舞 いは以下の通り。

対象タスクが休止状態である場合には、対象タスクに対してタスク起動時に行  
うべき初期化処理が行われ、対象タスクは実行できる状態になる【NGKI1118】。

対象タスクが休止状態でない場合には、対象タスクの起動要求キューイング数  
に1が加えられる【  
NGKI1119】。起動要求キューイング数に1を加えると  
E\_QOVRエラーとなる【NGKI1120】。  
TMAX\_ACTCNTを超える場合には、

act\_tskにおいてtskid[=TSK\_SELF (=0) を指定すると、自タスクが対象タスク となる【NGKI1121】。

### 【補足説明】

マルチプロセッサ対応カーネルでは、act\_tsk/iact\_tskは、対象タスクの次回  
起動時の割付けプロセッサを変更しない。

|           |                                    |
|-----------|------------------------------------|
| mact_tsk  | 割付けプロセッサ指定でのタスクの起動 [TM] 【NGKI1122】 |
| imact_tsk | 割付けプロセッサ指定でのタスクの起動 [IM] 【NGKI1123】 |

### 【C言語API】

```
ER ercd = mact_tsk(ID tskid, ID prcid)
ER ercd = imact_tsk(ID tskid, ID prcid)
```

### 【パラメータ】

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

## 【リターンパラメータ】

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

## 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー   |
|         | ・非タスクコンテキストからの呼出し (mact_tskの場合)<br>【NGKI1124】             |
|         | ・タスクコンテキストからの呼出し (imact_tskの場合)<br>【NGKI1125】             |
|         | ・CPUロック状態からの呼出し 【NGKI1126】                                |
| E_NOSPT | 未サポート機能   |
|         | ・対象タスクが制約タスク 【NGKI1127】                                   |
| E_ID    | 不正ID番号  |
|         | ・tskidが有効範囲外 【NGKI1128】                                   |
|         | ・prcidが有効範囲外 【NGKI1129】                                   |
| E_PAR   | パラメータエラー  |
|         | ・条件については機能の項を参照   |
| E_NOEXS | オブジェクト未登録   |
|         | ・対象タスクが未登録 [D] 【NGKI1130】                                 |
| E_OACV  | オブジェクトアクセス違反  |
|         | ・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<br>の場合) [P] 【NGKI1131】 |
| E_QOVR  | キューイングオーバフロー  |
|         | ・条件については機能の項を参照   |

## 【機能】

prcidで指定したプロセッサを割付けプロセッサとして、tskidで指定したタスク（対象タスク）に対して起動要求を行う。具体的な振舞いは以下の通り。

対象タスクが休止状態である場合には、対象タスクの割付けプロセッサがprcidで指定したプロセッサに変更された後、対象タスクに対してタスク起動時に行うべき初期化処理が行われ、対象タスクは実行できる状態になる【NGKI1132】。

対象タスクが休止状態でない場合には、対象タスクの起動要求キューイング数1が加えられ、次回起動時の割付けプロセッサがprcidで指定したプロセッサに変更される【NGKI1133】。起動要求キューイング数に1を加えるとTMAX\_ACTCNTを超える場合には、E\_QOVRエラーとなる【NGKI1134】。

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クとなる【NGKI1135】。

対象タスクの属するクラスの割付け可能プロセッサが、prcidで指定したプロセッサを含んでいない場合には、E\_PARエラーとなる【NGKI1136】。

prcid[=TPRC\_INI (=0) を指定すると、対象タスクの割付けプロセッサを、それが属するクラスの初期割付けプロセッサとする【NGKI1137】.

#### 【補足説明】

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

#### 【TOPPERS/ASPカーネルにおける規定】

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

#### 【TOPPERS/HRP2カーネルにおける規定】

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

#### 【TOPPERS/SSPカーネルにおける規定】

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

#### 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである。

can\_act タスク起動要求のキャンセル [T] 【NGKI1138】

#### 【C言語API】

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

#### 【パラメータ】

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

#### 【リターンパラメータ】

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

## 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し 【NGKI1139】<br>・CPUロック状態からの呼出し 【NGKI1140】 |
| E_ID    | 不正ID番号<br>・tskidが有効範囲外 【NGKI1141】                                       |
| E_NOEXS | オブジェクト未登録<br>・対象タスクが未登録 [D] 【NGKI1142】                                  |
| E_OACV  | オブジェクトアクセス違反<br>・対象タスクに対する通常操作1が許可されていない [P]<br>【NGKI1143】              |

## 【機能】

tskidで指定したタスク（対象タスク）に対する処理されていない起動要求をすべてキャンセルし、キャンセルした起動要求の数を返す。具体的な振舞いは以下の通り。

対象タスクの起動要求キューリング数が0に設定され、0に設定する前の起動要求キューリング数が、サービスコールの返値として返される【NGKI1144】。また、マルチプロセッサ対応カーネルにおいては、対象タスクの次回起動時の割付けプロセッサが未設定状態に戻される【NGKI1145】。

tskidにTSK\_SELF (=0) を指定すると、自タスクが対象タスクとなる【NGKI1146】。

## 【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、can\_actをサポートしない【SSPS0116】。

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

## 【C言語API】

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

## 【パラメータ】

|    |       |                   |
|----|-------|-------------------|
| ID | tskid | 対象タスクのID番号        |
| ID | prcid | タスクの割付けプロセッサのID番号 |

## 【リターンパラメータ】

ER ercd 正常終了 (E\_OK) またはエラーコード

## 【エラーコード】

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

## 【機能】

tskidで指定したタスクの割付けプロセッサを、 prcidで指定したプロセッサに変更する。具体的な振舞いは以下の通り。

対象タスクが、自タスクが割り付けられたプロセッサに割り付けられている場合には、対象タスクを prcidで指定したプロセッサに割り付ける【NGKI1155】。  
対象タスクが実行できる状態の場合には、 prcidで指定したプロセッサに割り付けられた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【NGKI1156】。

対象タスクが、自タスクが割付けられたプロセッサと異なるプロセッサに割り付けられている場合には、 E\_OBJエラーとなる【NGKI1157】。

tskidにTSK\_SELF (=0) を指定すると、自タスクが対象タスクとなる【NGKI1158】。

ディスパッチ保留状態で、対象タスクを自タスクとしてmig\_tskを呼び出すと、 エラーとなる【NGKI1159】。

E\_CTX

対象タスクの属するクラスの割付け可能プロセッサが、 prcidで指定したプロセッサを含んでいない場合には、 E\_PARエラーとなる【NGKI1160】。

prcidにTPRC\_INI (=0) を指定すると、対象タスクの割付けプロセッサを、それが属するクラスの初期割付けプロセッサに変更する【NGKI1161】。

## 【補足説明】

この仕様では、タスクをマイグレーションさせることができるのは、そのタス

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

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

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、mig\_tskをサポートしない【ASPS0110】。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、mig\_tskをサポートしない【HRPS0107】。

#### 【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、mig\_tskをサポートしない【SSPS0117】。

#### 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである。

ext\_tsk      自タスクの終了【T】 【NGKI1162】

#### 【C言語API】

```
ER ercd = ext_tsk()
```

#### 【パラメータ】

なし

#### 【リターンパラメータ】

ER            ercd      エラーコード

#### 【エラーコード】

|       |   |
|-------|---|
| E_SYS | システムエラー<br>・カーネルの誤動作【NGKI1163】            |
| E_CTX | コンテキストエラー<br>・非タスクコンテキストからの呼び出し【NGKI1164】 |

## 【機能】

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

き処理が行われる【

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

には、

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】。

## 【TOPPERS/SSPカーネルにおける規定】

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

## 【μITRON4.0仕様との関係】

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

ter\_tsk タスクの強制終了 [T] 【NGKI1170】

## 【C言語API】

```
ER ercd = ter_tsk(ID tskid)
```

## 【パラメータ】

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

## 【リターンパラメータ】

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

## 【エラーコード】

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

### 【機能】

tskidで指定したタスク（対象タスク）を終了させる。具体的には、対象タスクが休止状態でない場合には、対象タスクに対してタスク終了時に行うべき処理が行われる【NGKI3450】。

マルチプロセッサ対応カーネルでは、対象タスクは、自タスクと同じプロセッサに割り付けられているタスクに限られる。対象タスクが自タスクと異なるプロセッサに割り付けられている場合には、E\_OBJエラーとなる【NGKI1182】。

### 【TOPPERS/FMPカーネルにおける使用上の注意】

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

### 【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、ter\_tskをサポートしない【SSPS0120】。

chg\_pri タスクのベース優先度の変更〔T〕【NGKI1183】

### 【C言語API】

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

### 【パラメータ】

|     |        |            |
|-----|--------|------------|
| ID  | tskid  | 対象タスクのID番号 |
| PRI | tskpri | ベース優先度     |

### 【リターンパラメータ】

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

### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し 【NGKI1184】<br>・CPUロック状態からの呼出し 【NGKI1185】 |
| E_NOSPT | 未サポート機能<br>・対象タスクが制約タスク 【NGKI1186】                                      |
| E_ID    | 不正ID番号<br>・tskidが有効範囲外 【NGKI1187】                                       |
| E_PAR   | パラメータエラー<br>・tskpriが有効範囲外 【NGKI1188】                                    |
| E_NOEXS | オブジェクト未登録<br>・対象タスクが未登録 [D] 【NGKI1189】                                  |
| E_OACV  | オブジェクトアクセス違反<br>・対象タスクに対する通常操作2が許可されていない [P]<br>【NGKI1190】              |
| E_ILUSE | サービスコール不正使用<br>・条件については機能の項を参照  |
| E_OBJ   | オブジェクト状態エラー<br>・対象タスクが休止状態 【NGKI1191】                                   |

### 【機能】

tskidで指定したタスク（対象タスク）のベース優先度を、tskpriで指定した優先度に変更する。具体的な振舞いは以下の通り。

対象タスクが休止状態でない場合には、対象タスクのベース優先度が、tskpriで指定した優先度に変更される【NGKI1192】。それに伴って、対象タスクの現在優先度も変更される【NGKI1193】。

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優先度に従って、その待ち行列中の順序が変更される【NGKI1195】。待ち行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のNGKI1196】。

中で最後になる【

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、対象タスクの現在優先度が変更されることはなく、優先順位も変更されない【NGKI1197】。

tskidにTSK\_SELF (=0) を指定すると、自タスクが対象タスクとなる【NGKI1198】。また、tskpriにTPRI\_INI (=0) を指定すると、対象タスクのベス優先度が、起動時優先度に変更される【NGKI1199】。

対象タスクが優先度上限ミューテックスをロックしているかロックを待つてい  
る場合、  
tskpriは、それらのミューテックスの上限優先度と同じかそれより低  
くなければならない。そうでない場合には、E\_ILUSEエラーとなる【NGKI1201】。

保護機能対応カーネルで、chg\_priを呼び出した処理単位がユーザドメインに属する場合、  
tskpriは、そのユーザドメインが指定できる最高のタスク優先度と  
同じかそれより低くなければならない。そうでない場合には、E\_ILUSEエラーとなる【NGKI3440】。

#### 【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、chg\_priをサポートしない【SSPS0121】。

#### 【μITRON4.0仕様との関係】

対象タスクが、同じ優先度のタスクの中で最低の優先順位となる（対象タスクが待ち状態で、タスクの優先度順の待ち行列につながれている場合には、同じ優先度のタスクの中での順序が最後になる）条件を変更した。

get\_pri タスク優先度の参照 [T] 【NGKI1202】

#### 【C言語API】

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

#### 【パラメータ】

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

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI1203】<br>・CPUロック状態からの呼出し【NGKI1204】 |
| E_ID    | 不正ID番号<br>・tskidが有効範囲外【NGKI1205】                                      |
| E_NOEXS | オブジェクト未登録<br>・対象タスクが未登録〔D〕【NGKI1206】                                  |
| E_OACV  | オブジェクトアクセス違反<br>・対象タスクに対する参照操作が許可されていない〔P〕【NGKI1207】                  |
| E_MACV  | メモリアクセス違反<br>・p_tskpriが指すメモリ領域への書き込みアクセスが許可されていない〔P〕【NGKI1208】        |
| E_OBJ   | オブジェクト状態エラー<br>・対象タスクが休止状態【NGKI1209】                                  |

### 【機能】

tskidで指定したタスク（対象タスク）の現在優先度を参照する。具体的な振舞いは以下の通り。

対象タスクが休止状態でない場合には、対象タスクの現在優先度が、p\_tskpriが指すメモリ領域に返される【NGKI1210】。

tskidにTSK\_SELF (=0) を指定すると、自タスクが対象タスクとなる【NGKI1211】。

### 【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、get\_priをサポートしない【SSPS0122】。

get\_inf      自タスクの拡張情報の参照〔T〕【NGKI1212】

### 【C言語API】

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

### 【パラメータ】

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

### 【リターンパラメータ】

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

## 【エラーコード】

|        |   |
|--------|---|
| E_CTX  | コンテキストエラー<br>・非タスクコンテキストからの呼出し 【NGKI1213】<br>・CPUロック状態からの呼出し 【NGKI1214】 |
| E_MACV | メモリアクセス違反<br>・p_exinfが指すメモリ領域への書き込みアクセスが許可されていない [P] 【NGKI1215】         |

## 【機能】

自タスクの拡張情報を参照する。参照した拡張情報は、p\_exinfが指すメモリ領域に返される【NGKI1216】。

## 【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、get\_infをサポートしない【SSPS0123】。

## 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである。

ref\_tsk タスクの状態参照 [T] 【NGKI1217】

## 【C言語API】

```
ER ercd = ref_tsk(ID tskid, T_RTSK *pk_rtsk)
```

## 【パラメータ】

|          |         |                        |
|----------|---------|------------------------|
| ID       | tskid   | 対象タスクのID番号             |
| T_RTSK * | pk_rtsk | タスクの現在状態を入れるパケットへのポインタ |

## 【リターンパラメータ】

ER ercd 正常終了 (E\_OK) またはエラーコード

\* タスクの現在状態 (パケットの内容)

|        |           |   |
|--------|-----------|---|
| STAT   | tskstat   | タスク状態                                     |
| PRI    | tskpri    | タスクの現在優先度                                 |
| PRI    | tskbpri   | タスクのベース優先度                                |
| STAT   | tskwait   | タスクの待ち要因                                  |
| ID     | wobjid    | タスクの待ち対象のオブジェクトのID                        |
| TMO    | lefttmo   | タスクがタイムアウトするまでの時間                         |
| uint_t | actcnt    | タスクの起動要求キューイング数                           |
| uint_t | wupcnt    | タスクの起床要求キューイング数                           |
| bool_t | 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)       |
| 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)             |
| uint_t | svcllevel | タスクの拡張サービスコールのネストレベル (保護機能対応カーネルの場合)      |
| ID     | prcid     | タスクの割付けプロセッサのID (マルチプロセッサ対応カーネルの場合)       |
| ID     | actprc    | タスクの次回起動時の割付けプロセッサのID (マルチプロセッサ対応カーネルの場合) |

【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー   |
|         | ・ 非タスクコンテキストからの呼び出し 【NGKI1218】                      |
|         | ・ CPUロック状態からの呼び出し 【NGKI1219】                        |
| E_ID    | 不正ID番号  |
|         | ・ tskidが有効範囲外 【NGKI1220】                            |
| E_NOEXS | オブジェクト未登録   |
|         | ・ 対象タスクが未登録 [D] 【NGKI1221】                          |
| E_OACV  | オブジェクトアクセス違反  |
|         | ・ 対象タスクに対する参照操作が許可されていない [P] 【NGKI1222】             |
| E_MACV  | メモリアクセス違反   |
|         | ・ pk_rtskが指すメモリ領域への書き込みアクセスが許可されていない [P] 【NGKI1223】 |

【機能】

tskidで指定したタスク (対象タスク) の現在状態を参照する。参照した現在状態は、pk\_rtskで指定したメモリ領域に返される【NGKI1224】。

tskstatには、対象タスクの現在のタスク状態を表す次のいずれかの値が返される【NGKI1225】。

|         |       |        |
|---------|-------|--------|
| TTS_RUN | 0x01U | 実行状態   |
| TTS_RDY | 0x02U | 実行可能状態 |
| TTS_WAI | 0x04U | 待ち状態   |
| TTS_SUS | 0x08U | 強制待ち状態 |
| TTS_WAS | 0x0cU | 二重待ち状態 |
| TTS_DMT | 0x10U | 休止状態   |

マルチプロセッサ対応カーネルでは、対象タスクが自タスクの場合にも、  
TTS\_SUSとなる場合がある【NGKI1226】。この状況は、自タスクに対  
ref\_tskを発行するのと同じタイミングで、他のプロセッサで実行されてい  
るタスクから同じタスクに対してsus\_tskが発行された場合に発生する可能性がある。

tskstatが  
して

対象タスクが休止状態でない場合には、tskpriには対象タスクの現在優先度が、  
tskbpriには対象タスクのベース優先度が返される【NGKI1227】。対象タスクが  
休止状態である場合には、tskpriとtskbpriの値は保証されない【NGKI1228】。

対象タスクが待ち状態である場合には、tskwaitには、対象タスクが何を待って  
いる状態であるかを表す次のいずれかの値が返される【NGKI1229】。

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

対象タスクが待ち状態でない場合には、tskwaitの値は保証されない【NGKI1230】。

対象タスクが起床待ち状態および時間経過待ち状態以外の待ち状態である場合  
wobjidに、対象タスクが待っているオブジェクトのID番号が返される  
【NGKI1231】。対象タスクが待ち状態でない場合や、起床待ち状態または時間  
経過待ち状態である場合には、wobjidの値は保証されない【NGKI1232】。

には、  
【

対象タスクが時間経過待ち状態以外の待ち状態である場合には、lefttmoに、タ  
スクがタイムアウトを起こすまでの相対時間が返される【NGKI1233】。タスク  
がタイムアウトを起こさない場合には、TMO\_FEVR (= -1) が返される【NGKI1234】。

対象タスクが時間経過待ち状態である場合には、lefttmoに、タスクの遅延時間

が経過して待ち解除されるまでの相対時間が返される【NGKI1235】。ただし、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型に定義される）に型キャストした値が返される【NGKI1236】。

対象タスクが待ち状態でない場合には、lefttmoの値は保証されない【NGKI1237】。

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

対象タスクが休止状態でない場合には、wupcntに、タスクの起床要求キューイング数が返される【NGKI1239】。対象タスクが休止状態である場合には、  
wupcntの値は保証されない【NGKI1240】。

保護機能対応カーネルで、対象タスクが休止状態でない場合には、texmskに、  
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にtrue、そうでない場合にfalseが返される【NGKI1242】。またsvclevelには、対象タスクが拡張サービスコールを呼び出していない場合には  
0、呼び出している場合には、実行中の拡張サービスコールがネスト段数が返される【NGKI1243】。対象タスクが休止状態である場合には、texmsk, waifbd,  
svclevelの値は保証されない【NGKI1244】。

マルチプロセッサ対応カーネルでは、prcidに、対象タスクの割付けプロセッサのID  
番号が返される【NGKI1245】。またactprcには、対象タスクの次回起動時  
の割付けプロセッサのID番号が返される【NGKI1246】。次回起動時の割付けプロセッサが未設定の場合には、actprc  
にTPRC\_NONE (=0) が返される【NGKI1247】。

tskidにTSK\_SELF (=0) を指定すると、自タスクが対象タスクとなる【NGKI1248】。

#### 【補足説明】

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  
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 正しく得ることができる。

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、tskwaitにTTW\_MTX, TTW\_SMBF, TTW\_RMBFが返ることはない【ASPS0111】。  
ただし、ミューテックス機能拡張パッケージを用いると、tskwaitに  
TTW\_MTXが返る場合がある【ASPS0112】。また、メッセージバッファ  
機能拡張パッケージを用いると、tskwaitにTTW\_SMBFとTTW\_RMBFが返る場合がある【ASPS0208】。

#### 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、tskwaitにTTW\_MTX, TTW\_SMBF, TTW\_RMBFが返ることはない【FMPS0106】。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、tskwaitにTTW\_MBX, TTW\_SMBF, TTW\_RMBFが返ることはない【HRPS0108】。  
ただし、メッセージバッファ機能拡張パッケージを用いると、tskwaitにTTW\_SMBF

とTTW\_RMBFが返る場合がある【HRPS0174】.

#### 【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、ref\_tskをサポートしない【SSPS0124】.

#### 【使用上の注意】

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

#### 【μITRON4.0仕様との関係】

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

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

#### 【μITRON4.0/PX仕様との関係】

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

## 1.2. タスク付属同期機能

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

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

- 起床要求キューイング数

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

タスク付属同期機能に関連するカーネル構成マクロは次の通り。

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

#### 【TOPPERS/ASPカーネルにおける規定】

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

#### 【TOPPERS/FMPカーネルにおける規定】

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

## 【TOPPERS/HRP2カーネルにおける規定】

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

## 【TOPPERS/SSPカーネルにおける規定】

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】。

## 【μITRON4.0仕様との関係】

この仕様では、強制待ち要求をネストする機能をサポートしないこととした。  
言い換えると、強制待ち要求ネスト数の最大値を1に固定する。これに伴い、強制待ち状態から強制再開するサービスコール(frsn\_tsk)とタスクの強制待ち要求ネスト数の最大値を表すカーネル構成マクロ(TMAX\_SUSCNT)は廃止した。また、ref\_tskで参照できる情報(T\_RTSKのフィールド)から、強制待ち要求ネスト数(suscnt)を除外した。

|          |                               |
|----------|-------------------------------|
| slp_tsk  | 起床待ち [T] 【NGKI1252】           |
| tslp_tsk | 起床待ち（タイムアウト付き） [T] 【NGKI1253】 |

## 【C言語API】

```
ER ercd = slp_tsk()  
ER ercd = tslp_tsk(TM0 tmout)
```

## 【パラメータ】

|     |       |                        |
|-----|-------|------------------------|
| TM0 | tmout | タイムアウト時間 (tslp_tskの場合) |
|-----|-------|------------------------|

## 【リターンパラメータ】

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

## 【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー<br>・ディスパッチ保留状態からの呼び出し 【NGKI1254】     |
| E_NOSPT | 未サポート機能<br>・制約タスクからの呼び出し 【NGKI1255】            |
| E_PAR   | パラメータエラー<br>・tmoutが無効 (tslp_tskの場合) 【NGKI1256】 |
| E_TMOUT | ポーリング失敗またはタイムアウト (slp_tskを除く) 【NGKI1257】       |
| E_RLWAI | 待ち禁止状態または待ち状態の強制解除 【NGKI1258】                  |

## 【機能】

自タスクを起床待ちさせる。具体的な振舞いは以下の通り。

自タスクの起床要求キューイング数が0でない場合には、起床要求キューイング  
が減ぜられる【NGKI1259】。起床要求キューイング数が0の場合には、  
自タスクは起床待ち状態となる【NGKI1260】。

数から1

## 【補足説明】

自タスクの起床要求キューイング数が0でない場合には、自タスクは実行できる  
状態を維持し、自タスクの優先順位は変化しない。

|          |                       |
|----------|-----------------------|
| wup_tsk  | タスクの起床 [T] 【NGKI1261】 |
| iwup_tsk | タスクの起床 [I] 【NGKI1262】 |

## 【C言語API】

|                              |
|------------------------------|
| ER ercd = wup_tsk(ID tskid)  |
| ER ercd = iwup_tsk(ID tskid) |

## 【パラメータ】

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

## 【リターンパラメータ】

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

## 【エラーコード】

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

### 【機能】

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

対象タスクが起床待ち状態である場合には、対象タスクが待ち解除される  
【NGKI1271】。待ち解除されたタスクには、待ち状態となったサービスコール  
からE\_OKが返る【NGKI1272】。

対象タスクが起床待ち状態でなく、休止状態でもない場合には、対象タスクの起床要求キューイング数に  
1が加えられる【NGKI1273】。起床要求キューイング  
数に1を加えるとTMAX\_WUPCNT  
を超える場合には、E\_QOVRエラーとなる【NGKI1274】。

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となる【NGKI1275】。

can\_wup タスク起床要求のキャンセル [T] 【NGKI1276】

### 【C言語API】

```
ER_UINT wupcnt = can_wup(ID tskid)
```

### 【パラメータ】

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

### 【リターンパラメータ】

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

### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI1277】<br>・CPUロック状態からの呼出し【NGKI1278】 |
| E_NOSPT | 未サポート機能<br>・対象タスクが制約タスク【NGKI1279】                                     |
| E_ID    | 不正ID番号<br>・tskidが有効範囲外【NGKI1280】                                      |
| E_NOEXS | オブジェクト未登録<br>・対象タスクが未登録[D]【NGKI1281】                                  |
| E_OACV  | オブジェクトアクセス違反<br>・対象タスクに対する通常操作1が許可されていない[P]<br>【NGKI1282】             |
| E_OBJ   | オブジェクト状態エラー<br>・対象タスクが休止状態【NGKI1283】                                  |

### 【機能】

tskidで指定したタスク（対象タスク）に対する処理されていない起床要求をすべてキャンセルし、キャンセルした起床要求の数を返す。具体的な振舞いは以下の通り。

対象タスクが休止状態でない場合には、対象タスクの起床要求キューイング数が0に設定され、0に設定する前の起床要求キューイング数が、サービスコールの返値として返される【NGKI1284】。

tskidにTSK\_SELF (=0) を指定すると、自タスクが対象タスクとなる【NGKI1285】。

|          |                       |
|----------|-----------------------|
| rel_wai  | 強制的な待ち解除[T]【NGKI1286】 |
| irel_wai | 強制的な待ち解除[I]【NGKI1287】 |

### 【C言語API】

```
ER ercd = rel_wai(ID tskid)
ER ercd = irel_wai(ID tskid)
```

### 【パラメータ】

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

### 【リターンパラメータ】

ER

ercd

正常終了 (E\_OK) またはエラーコード

### 【エラーコード】

E\_CTX

コンテキストエラー

- ・非タスクコンテキストからの呼出し (rel\_waiの場合) 【NGKI1288】
- ・タスクコンテキストからの呼出し (irel\_waiの場合) 【NGKI1289】
- ・CPUロック状態からの呼出し 【NGKI1290】

E\_NOSPT

未サポート機能

- ・対象タスクが制約タスク 【NGKI1291】

E\_ID

不正ID番号

- ・tskidが有効範囲外 【NGKI1292】

E\_NOEXS

オブジェクト未登録

- ・対象タスクが未登録 [D] 【NGKI1293】

E\_OACV

オブジェクトアクセス違反

- ・対象タスクに対する通常操作2が許可されていない (rel\_wai の場合) [P] 【NGKI1294】

E\_OBJ

オブジェクト状態エラー

- ・対象タスクが待ち状態でない 【NGKI1295】

### 【機能】

tskidで指定したタスク（対象タスク）を、強制的に待ち解除する。具体的な振舞いは以下の通り。

対象タスクが待ち状態である場合には、対象タスクが待ち解除される

NGKI1296】。待ち解除されたタスクには、待ち状態となったサービスコール  
が返る【NGKI1297】。

からE\_RLWAI

sus\_tsk 強制待ち状態への遷移 [T] 【NGKI1298】

### 【C言語API】

```
ER ercd = sus_tsk(ID tskid)
```

### 【パラメータ】

ID

tskid

対象タスクのID番号

### 【リターンパラメータ】

ER

ercd

正常終了 (E\_OK) またはエラーコード

## 【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し 【NGKI1299】<br>・CPUロック状態からの呼出し 【NGKI1300】<br>・その他の条件については機能の項を参照 |
| E_NOSPT | 未サポート機能<br>・対象タスクが制約タスク 【NGKI1301】   |
| E_ID    | 不正ID番号<br>・tskidが有効範囲外 【NGKI1302】  |
| E_NOEXS | オブジェクト未登録<br>・対象タスクが未登録 [D] 【NGKI1303】   |
| E_OACV  | オブジェクトアクセス違反<br>・対象タスクに対する通常操作2が許可されていない [P]<br>【NGKI1304】                                     |
| E_OBJ   | オブジェクト状態エラー<br>・対象タスクが休止状態 【NGKI1305】  |
| E_QOVR  | キューイングオーバフロー<br>・対象タスクが強制待ち状態（二重待ち状態を含む） 【NGKI1306】  |

## 【機能】

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

対象タスクが実行できる状態である場合には、対象タスクは強制待ち状態となる【NGKI1307】。また、待ち状態（二重待ち状態を除く）である場合には、二重待ち状態となる【NGKI1308】。

マルチプロセッサ対応カーネルでは、対象タスクが自タスクの場合にも、E\_QOVRエラーとなる場合がある【NGKI1309】。この状況は、自タスクに対してsus\_tskを発行するのと同じタイミングで、他のプロセッサで実行されているタスクから同じタスクに対してsus\_tskが発行された場合に発生する可能性がある。

tskidにTSK\_SELF (=0) を指定すると、自タスクが対象タスクとなる【NGKI1310】。

ディスパッチ保留状態で、対象タスクを自タスクとしてsus\_tskを呼び出すと、E\_CTXエラーとなる【NGKI1311】。なお、sus\_tskは、自タスクを広義の待ち状態に遷移させる可能性のあるサービスコールであるが、対象タスクが自タスクでない場合には、割込み優先度マスクが全解除でない状態やディスパッチ禁止E\_CTXエラーにはならない【NGKI3604】。これは、状態で呼び出しても、【NGKI0175】と【NGKI0179】の原則の例外となっている。

rsm\_tsk 強制待ち状態からの再開 [T] 【NGKI1312】

## 【C言語API】

```
ER ercd = rsm_tsk(ID tskid)
```

#### 【パラメータ】

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

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼び出し 【NGKI1313】<br>・CPUロック状態からの呼び出し 【NGKI1314】 |
| E_NOSPT | 未サポート機能<br>・対象タスクが制約タスク 【NGKI1315】  |
| E_ID    | 不正ID番号<br>・tskidが有効範囲外 【NGKI1316】   |
| E_NOEXS | オブジェクト未登録<br>・対象タスクが未登録 [D] 【NGKI1317】                                    |
| E_OACV  | オブジェクトアクセス違反<br>・対象タスクに対する通常操作2が許可されていない [P]<br>【NGKI1318】                |
| E_OBJ   | オブジェクト状態エラー<br>・対象タスクが強制待ち状態（二重待ち状態を含む）でない<br>【NGKI1319】                  |

#### 【機能】

tskidで指定したタスク（対象タスク）を、強制待ちから再開する。具体的な振舞いは以下の通り。

対象タスクが強制待ち状態である場合には、対象タスクは強制待ちから再開される 【NGKI1320】。

|          |                            |
|----------|----------------------------|
| dis_wai  | 待ち禁止状態への遷移 [TP] 【NGKI1321】 |
| idis_wai | 待ち禁止状態への遷移 [IP] 【NGKI1322】 |

#### 【C言語API】

|                              |
|------------------------------|
| ER ercd = dis_wai(ID tskid)  |
| ER ercd = idis_wai(ID tskid) |

## 【パラメータ】

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

## 【リターンパラメータ】

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

## 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し (dis_waiの場合) 【NGKI1323】<br>・タスクコンテキストからの呼出し (idis_waiの場合) 【NGKI1324】<br>・CPUロック状態からの呼出し 【NGKI1325】 |
| E_NOSPT | 未サポート機能<br>・対象タスクが制約タスク 【NGKI1326】  |
| E_ID    | 不正ID番号<br>・tskidが有効範囲外 【NGKI1327】   |
| E_NOEXS | オブジェクト未登録<br>・対象タスクが未登録 [D] 【NGKI1328】  |
| E_OACV  | オブジェクトアクセス違反<br>・対象タスクに対する通常操作2が許可されていない (dis_wai の場合) 【NGKI1329】   |
| E_OBJ   | オブジェクト状態エラー<br>・対象タスクが休止状態 【NGKI1330】<br>・対象タスクがタスク例外処理マスク状態でない 【NGKI1331】  |
| E_QOVR  | キューイングオーバフロー<br>・対象タスクが待ち禁止状態 【NGKI1332】  |

## 【機能】

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

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、  
対象タスクは待ち禁止状態になる【NGKI1333】。

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となる【NGKI1334】。

## 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、dis\_waiをサポートしない【ASPS0114】。

## 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、dis\_waiをサポートしない【FMPS0108】。

## 【補足説明】

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を呼び出せばよい。

## 【未決定事項】

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、現時点では未決定である。

## 【μITRON4.0/PX仕様との関係】

μITRON4.0/PX仕様に定義されていないサービスコールである。

|          |                           |
|----------|---------------------------|
| ena_wai  | 待ち禁止状態の解除 [TP] 【NGKI1335】 |
| iена_wai | 待ち禁止状態の解除 [IP] 【NGKI1336】 |

## 【C言語API】

```
ER ercd = ena_wai(ID tskid)
ER ercd = iena_wai(ID tskid)
```

## 【パラメータ】

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

## 【リターンパラメータ】

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

## 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し (ena_waiの場合) 【NGKI1337】<br>・タスクコンテキストからの呼出し (iena_waiの場合) 【NGKI1338】<br>・CPUロック状態からの呼出し 【NGKI1339】 |
| E_NOSPT | 未サポート機能<br>・対象タスクが制約タスク 【NGKI1340】  |
| E_ID    | 不正ID番号<br>・tskidが有効範囲外 【NGKI1341】   |
| E_NOEXS | オブジェクト未登録<br>・対象タスクが未登録 [D] 【NGKI1342】  |
| E_OACV  | オブジェクトアクセス違反<br>・対象タスクに対する通常操作2が許可されていない (ena_wai<br>の場合) 【NGKI1343】  |
| E_OBJ   | オブジェクト状態エラー<br>・対象タスクが休止状態 【NGKI1344】<br>・対象タスクが待ち禁止状態でない 【NGKI1345】  |

### 【機能】

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。具体的な振舞いは以下の通り。

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 【NGKI1346】。

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となる 【NGKI1347】。

### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、ena\_waiをサポートしない 【ASPS0115】。

### 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、ena\_waiをサポートしない 【FMPGS0109】。

### 【未決定事項】

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、現時点では未決定である。

### 【μITRON4.0/PX仕様との関係】

μITRON4.0/PX仕様に定義されていないサービスコールである。

dly\_tsk      自タスクの遅延 [T] 【NGKI1348】

### 【C言語API】

```
ER ercd = dly_tsk(RELTIM dlytim)
```

#### 【パラメータ】

|        |        |      |
|--------|--------|------|
| RELTIM | dlytim | 遅延時間 |
|--------|--------|------|

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・ディスパッチ保留状態からの呼び出し 【NGKI1349】      |
| E_NOSPT | 未サポート機能<br>・制約タスクからの呼び出し 【NGKI1350】             |
| E_PAR   | パラメータエラー<br>・dlytimがTMAX_RELTIMより大きい 【NGKI1351】 |
| E_RLWAI | 待ち禁止状態または待ち状態の強制解除 【NGKI1352】                   |

#### 【機能】

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

自タスクは、dlytimで指定した時間が経過するまでの間、時間経過待ち状態となる【NGKI1353】。  
dly\_tskを呼び出してからdlytimで指定した相対時間後に、自タスクは待ち解除され、dly\_tskからE\_OKが返る【NGKI1354】。

## 1.3. タスク例外処理機能

タスク例外処理ルーチンは、カーネルが実行を制御する処理単位で、タスクと同一のコンテキスト内で実行される。タスク例外処理ルーチンは、各タスクに1つのみ登録できるため、タスクIDによって識別する【NGKI1355】。

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】。

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

タスク例外処理ルーチン属性に指定できる属性はない【NGKI1357】。そのため、タスク例外処理ルーチン属性には、TA\_NULLを指定しなければならない【NGKI1358】。

タスクは、タスク例外処理ルーチンの実行を保留するためのタスク例外処理禁止フラグを持つ【NGKI1359】。タスク例外処理禁止フラグがセットされた状態をタスク例外処理禁止状態、クリアされた状態をタスク例外処理許可状態と呼ぶ。タスク例外処理禁止フラグは、タスクの起動時に、セットした状態に初期化される【NGKI1361】。

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

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

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

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

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

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

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

タスク例外処理機能に用いるデータ型は次の通り。

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

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

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

texptnにはタスク例外処理ルーチン起動時の保留例外要因が、exinfにはタスクの拡張情報が、それぞれ渡される【NGKI1376】.

タスク例外処理機能に関連するカーネル構成マクロは次の通り.

|             |   |
|-------------|---|
| TBIT_TEXPTN | タスク例外要因のビット数（TEXPTNの有効ビット数）<br>【NGKI1377】 |
|-------------|---|

#### 【補足説明】

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

#### 【TOPPERS/ASPカーネルにおける規定】

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【ASPS0116】.

#### 【TOPPERS/FMPカーネルにおける規定】

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【FMPS0110】.

#### 【TOPPERS/HRP2カーネルにおける規定】

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【HRPS0110】.

#### 【TOPPERS/SSPカーネルにおける規定】

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】.

#### 【μITRON4.0仕様との関係】

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行が開始されないという仕様に変更した。

#### 【μITRON4.0/PX仕様との関係】

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

タックが不正となる問題に関して、μITRON4.0/PX仕様では考慮されていない。

### 【仕様変更の経緯】

この仕様のRelease

1.2以前では、タスク例外処理ルーチンの実行開始条件に

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

|         |                               |
|---------|-------------------------------|
| DEF_TEX | タスク例外処理ルーチンの定義〔S〕 【NGKI1378】  |
| def_tex | タスク例外処理ルーチンの定義〔TD〕 【NGKI1379】 |

### 【静的API】

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

### 【C言語API】

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

### 【パラメータ】

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

\* タスク例外処理ルーチンの定義情報（パケットの内容）

|        |        |                  |
|--------|--------|------------------|
| ATR    | texatr | タスク例外処理ルーチン属性    |
| TEXRTN | texrtn | タスク例外処理ルーチンの先頭番地 |

### 【リターンパラメータ】

```
ER ercd 正常終了 (E_OK) またはエラーコード
```

### 【エラーコード】

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

## 【機能】

tskidで指定したタスク（対象タスク）に対して、各パラメータで指定したタスク例外処理ルーチン定義情報に従って、タスク例外処理ルーチンを定義する 【NGKI1389】。

ただし、def\_texにおいてpk\_dtexをNULLにした場合には、対象タスクに対するタスク例外処理ルーチンの定義を解除する 【NGKI1390】。また、対象タスクのタスク例外処理禁止フラグをセットし、保留例外要因を0に初期化する 【NGKI1391】。

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

タスク例外処理ルーチンを定義する場合（DEF\_TEXの場合およびdef\_texにおいてpk\_dtexをNULL以外にした場合）で、対象タスクに対してすでにタスク例外処理ルーチンが定義されている場合には、E\_OBJエラーとなる 【NGKI1393】。

保護機能対応カーネルにおいて、DEF\_TEXは、対象タスクが属する保護ドメインの囲みの中に記述しなければならない。そうでない場合には、E\_RSATRエラーとなる 【NGKI1395】。また、def\_texでタスク例外処理ルーチンを定義する場合には、タスク例外処理ルーチン属性にTA\_DOM(domid)を指定した場合にはE\_RSATRエラーとなる 【NGKI1396】。ただし、TA\_DOM(TDOM\_SELF)を指定した場合には、指定がE\_RSATRエラーは検出されない 【NGKI1397】。

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

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

def\_texにおいてtskidにTSK\_SELF（=0）を指定すると、自タスクが対象タスクとなる【NGKI1403】。

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、DEF\_TEXのみをサポートする【ASPS0117】。ただし、動的生成機能拡張パッケージでは、def\_texもサポートする【ASPS0118】。

#### 【TOPPERS/FMPカーネルにおける規定】

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

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、DEF\_TEXのみをサポートする【HRPS0111】。ただし、動的生成機能拡張パッケージでは、def\_texもサポートする【HRPS0179】。

#### 【μITRON4.0仕様との関係】

texrtnのデータ型をTEXRTNに変更した。

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合に、E\_OBJエラーとすることにした。

ras\_tex タスク例外処理の要求 [T] 【NGKI1404】  
iras\_tex タスク例外処理の要求 [I] 【NGKI1405】

#### 【C言語API】

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

#### 【パラメータ】

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

### 【リターンパラメータ】

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

### 【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼び出し (ras_texの場合) 【NGKI1406】<br>・タスクコンテキストからの呼び出し (iras_texの場合) 【NGKI1407】<br>・CPUロック状態からの呼び出し 【NGKI1408】 |
| E_ID    | 不正ID番号<br>・tskidが有効範囲外 【NGKI1409】  |
| E_PAR   | パラメータエラー<br>・rasptnが0 【NGKI1410】   |
| E_NOEXS | オブジェクト未登録<br>・対象タスクが未登録 [D] 【NGKI1411】   |
| E_OACV  | オブジェクトアクセス違反<br>・対象タスクに対する通常操作2が許可されていない (ras_tex<br>の場合) [P] 【NGKI1412】   |
| E_OBJ   | オブジェクト状態エラー<br>・対象タスクが休止状態 【NGKI1413】<br>・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<br>ない 【NGKI1414】   |

### 【機能】

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、それまでの値とrasptnで指定した値のビット毎論理和（C言語の"|"）に更新される【NGKI1415】。

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となる【NGKI1416】。

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

### 【C言語API】

```
ER ercd = dis_tex()
```

### 【パラメータ】

なし

#### 【リターンパラメータ】

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

#### 【エラーコード】

|       |  |
|-------|--|
| E_CTX | コンテキストエラー                                |
|       | ・非タスクコンテキストからの呼び出し 【NGKI1419】            |
|       | ・CPUロック状態からの呼び出し 【NGKI1420】              |
| E_OBJ | オブジェクト状態エラー                              |
|       | ・自タスクに対してタスク例外処理ルーチンが定義されていない 【NGKI1421】 |

#### 【機能】

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、  
自タスクをタスク例外処理禁止状態に遷移させる。

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

#### 【C言語API】

|                     |
|---------------------|
| ER ercd = ena_tex() |
|---------------------|

#### 【パラメータ】

なし

#### 【リターンパラメータ】

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

#### 【エラーコード】

|       |   |
|-------|---|
| E_CTX | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI1424】<br>・CPUロック状態からの呼出し【NGKI1425】 |
| E_OBJ | オブジェクト状態エラー<br>・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ない【NGKI1426】                |

## 【機能】

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

## 【補足説明】

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

|         |                              |
|---------|------------------------------|
| sns_tex | タスク例外処理禁止状態の参照〔TI〕【NGKI1428】 |
|---------|------------------------------|

## 【C言語API】

|                          |
|--------------------------|
| bool_t state = sns_tex() |
|--------------------------|

## 【パラメータ】

|    |
|----|
| なし |
|----|

## 【リターンパラメータ】

|              |             |
|--------------|-------------|
| bool_t state | タスク例外処理禁止状態 |
|--------------|-------------|

## 【機能】

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

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

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

## 【補足説明】

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

ref\_tex タスク例外処理の状態参照 [T] 【NGKI1432】

## 【C言語API】

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

## 【パラメータ】

|          |         |                            |
|----------|---------|----------------------------|
| ID       | tskid   | 対象タスクのID番号                 |
| T_RTEX * | pk_rtex | タスク例外処理の現在状態を入れるパケットへのポインタ |

## 【リターンパラメータ】

ER ercd 正常終了 (E\_OK) またはエラーコード

\* タスク例外処理の現在状態 (パケットの内容)

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

## 【エラーコード】

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

## 【機能】

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

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

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

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

tskidにTSK\_SELF (=0) を指定すると、自タスクが対象タスクとなる【NGKI1444】。

## 1.4. 同期・通信機能

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

### 【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、同期・通信機能をサポートしない【SSPS0127】。

### 【μITRON4.0仕様との関係】

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

### 1.4.1. セマフォ

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

各セマフォが持つ情報は次の通り【NGKI1446】。

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

- 属するクラス（マルチプロセッサ対応カーネルの場合）

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

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

セマフォ属性には、次の属性を指定することができる【NGKI1448】。

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

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

セマフォ機能に関連するカーネル構成マクロは次の通り。

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

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

### 【μITRON4.0仕様との関係】

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

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

### 【静的API】

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

### 【C言語API】

|   |
|---|
| ER_ID semid = acre_sem(const T_CSEM *pk_csem) |
|---|

### 【パラメータ】

|          |         |                                    |
|----------|---------|------------------------------------|
| ID       | semid   | 生成するセマフォのID番号 (CRE_SEMの場合)         |
| T_CSEM * | pk_csem | セマフォの生成情報を入れたパケットへのポイント (静的APIを除く) |

\*セマフォの生成情報 (パケットの内容)

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

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー                                       |
|         | ・非タスクコンテキストからの呼び出し [s] 【NGKI1454】               |
|         | ・CPUロック状態からの呼び出し [s] 【NGKI1455】                 |
| E_RSATR | 予約属性  |
|         | ・sematrが無効 【NGKI1456】                           |
|         | ・属する保護ドメインの指定が有効範囲外 [sP] 【NGKI1457】             |
|         | ・属するクラスの指定が有効範囲外 [sM] 【NGKI1458】                |
|         | ・クラスの囲みの中に記述されていない [SM] 【NGKI1459】              |
| E_PAR   | パラメータエラー  |
|         | ・maxsemが有効範囲 (1以上TMAX_MAXSEM以下) 外 【NGKI1468】    |
|         | ・isemcntが有効範囲 (0以上maxsem以下) 外 【NGKI1466】        |
| E_OACV  | オブジェクトアクセス違反                                    |
|         | ・システム状態に対する管理操作が許可されていない [sP] 【NGKI1460】        |
| E_MACV  | メモリアクセス違反                                       |
|         | ・pk_csemが指すメモリ領域への読み出しが許可されていない [sP] 【NGKI1461】 |
| E_NOID  | ID番号不足  |
|         | ・割り付けられるセマフォIDがない [sD] 【NGKI1462】               |
| E_OBJ   | オブジェクト状態エラー                                     |
|         | ・semidで指定したセマフォが登録済み (CRE_SEMの場合) 【NGKI1463】    |

#### 【機能】

各パラメータで指定したセマフォ生成情報に従って、セマフォを生成する。生成されたセマフォの資源数は初期資源数に、待ち行列は空の状態に初期化される 【NGKI1464】。

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

整数定数式パラメータである【NGKI1465】.

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、CRE\_SEMのみをサポートする【ASPS0119】. ただし、動的生成機能拡張パッケージでは、acre\_semもサポートする【ASPS0120】.

#### 【TOPPERS/FMPカーネルにおける規定】

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

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、CRE\_SEMのみをサポートする【HRPS0112】. ただし、動的生成機能拡張パッケージでは、acre\_semもサポートする【HRPS0180】.

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

#### 【静的API】

AID\_SEM(uint\_t nosem)

#### 【パラメータ】

uint\_t nosem 割付け可能なセマフォIDの数

#### 【エラーコード】

E\_RSATR

予約属性

- ・保護ドメインの囲みの中に記述されている [P] 【NGKI3429】
- ・クラスの囲みの中に記述されていない [M] 【NGKI1470】

E\_PAR

パラメータエラー

- ・nosemが負の値 【NGKI3277】

#### 【機能】

nosemで指定した数のセマフォIDを、セマフォを生成するサービスコールによって割付け可能なセマフォIDとして確保する【NGKI1471】.

nosemは整数定数式パラメータである【NGKI1472】.

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルの動的生成機能拡張パッケージでは、AID\_SEMをサポートする【ASPS0211】.

## 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルの動的生成機能拡張パッケージでは、AID\_SEMをサポートする 【HRPS0212】。

|         |                                   |
|---------|-----------------------------------|
| SAC_SEM | セマフォのアクセス許可ベクタの設定〔SP〕 【NGKI1473】  |
| sac_sem | セマフォのアクセス許可ベクタの設定〔TPD〕 【NGKI1474】 |

## 【静的API】

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

## 【C言語API】

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

## 【パラメータ】

|         |         |                                   |
|---------|---------|-----------------------------------|
| ID      | semid   | 対象セマフォのID番号                       |
| ACVCT * | p_acvct | アクセス許可ベクタを入れたパケットへのポインタ（静的APIを除く） |

\* アクセス許可ベクタ（パケットの内容）

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

## 【リターンパラメータ】

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

## 【エラーコード】

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

## 【機能】

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

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

## 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、SAC\_SEMのみをサポートする 【HRPS0113】。ただし、動的生成機能拡張パッケージでは、sac\_semもサポートする 【HRPS0181】。

del\_sem セマフォの削除 [TD] 【NGKI1487】

## 【C言語API】

```
ER ercd = del_sem(ID semid)
```

## 【パラメータ】

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

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼び出し 【NGKI1488】<br>・CPUロック状態からの呼び出し 【NGKI1489】 |
| E_ID    | 不正ID番号<br>・semidが有効範囲外 【NGKI1490】   |
| E_NOEXS | オブジェクト未登録<br>・対象セマフォが未登録 【NGKI1491】                                       |
| E_OACV  | オブジェクトアクセス違反<br>・対象セマフォに対する管理操作が許可されていない [P]<br>【NGKI1492】                |
| E_OBJ   | オブジェクト状態エラー<br>・対象セマフォは静的APIで生成された 【NGKI1493】                             |

#### 【機能】

semidで指定したセマフォ（対象セマフォ）を削除する。具体的な振舞いは以下の通り。

対象セマフォの登録が解除され、そのセマフォIDが未使用の状態に戻される  
【NGKI1494】。また、対象セマフォの待ち行列につながれたタスクは、待ち行  
列の先頭のタスクから順に待ち解除される 【NGKI1495】。待ち解除されたタス  
クには、待ち状態となったサービスコールからE\_DLTエラーが返る 【NGKI1496】。

#### 【使用上の注意】

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】。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【HRPS0182】。

|                       |                           |
|-----------------------|---------------------------|
| <code>sig_sem</code>  | セマフォの資源の返却 [T] 【NGKI1497】 |
| <code>isig_sem</code> | 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び出し ( <code>sig_sem</code> の場合) 【NGKI1499】            |
|         | ・タスクコンテキストからの呼び出し ( <code>isig_sem</code> の場合) 【NGKI1500】            |
|         | ・CPUロック状態からの呼び出し 【NGKI1501】  |
| E_ID    | 不正ID番号   |
|         | ・ <code>semid</code> 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 ( <code>sig_sem</code> の場合) [P] 【NGKI1504】 |
| E_QOVR  | キューイングオーバフロー   |
|         | ・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。

対象セマフォの待ち行列にタスクが存在する場合には、待ち行列の先頭のタスクが待ち解除される【NGKI1505】。この時、待ち解除されたタスクが資源を獲

得したことになるため、対象セマフォの資源数は変化しない【NGKI1506】。待ち解除されたタスクには、待ち状態となったサービスコールからE\_OKが返る【NGKI1507】。

待ち行列にタスクが存在しない場合には、対象セマフォの資源数に1が加えられる【NGKI1508】。資源数に1を加えるとそのセマフォの最大資源数を越える場合には、E\_QOVRエラーとなる【NGKI1509】。

|          |                                    |
|----------|------------------------------------|
| wai_sem  | セマフォの資源の獲得 [T] 【NGKI1510】          |
| pol_sem  | セマフォの資源の獲得（ポーリング）[T] 【NGKI1511】    |
| twai_sem | セマフォの資源の獲得（タイムアウト付き）[T] 【NGKI1512】 |

#### 【C言語API】

```
ER ercd = wai_sem(ID semid)
ER ercd = pol_sem(ID semid)
ER ercd = twai_sem(ID semid, TMO tmout)
```

#### 【パラメータ】

|     |       |                        |
|-----|-------|------------------------|
| ID  | semid | 対象セマフォのID番号            |
| TMO | tmout | タイムアウト時間 (twai_semの場合) |

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI1513】<br>・CPUロック状態からの呼出し【NGKI1514】<br>・ディスパッチ保留状態からの呼出し（pol_semを除く）【NGKI1515】 |
| E_NOSPT | 未サポート機能<br>・制約タスクからの呼出し（pol_semを除く）【NGKI1516】  |
| E_ID    | 不正ID番号<br>・semidが有効範囲外【NGKI1517】   |
| E_PAR   | パラメータエラー<br>・tmoutが無効（twai_semの場合）【NGKI1518】   |
| E_NOEXS | オブジェクト未登録<br>・対象セマフォが未登録[D]【NGKI1519】  |
| E_OACV  | オブジェクトアクセス違反<br>・対象セマフォに対する通常操作2が許可されていない[P]<br>【NGKI1520】   |
| E_TMOUT | ポーリング失敗またはタイムアウト（wai_semを除く）【NGKI1521】   |
| E_RLWAI | 待ち禁止状態または待ち状態の強制解除（pol_semを除く）<br>【NGKI1522】   |
| 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）<br>【NGKI1523】   |

### 【機能】

semidで指定したセマフォ（対象セマフォ）から資源を獲得する。具体的な振舞いは以下の通り。

対象セマフォの資源数が1以上の場合には、資源数から1が減ぜられる  
【NGKI1524】。資源数が0の場合には、自タスクはセマフォの資源獲得待ち状態となり、対象セマフォの待ち行列につながれる【NGKI1525】。

ini\_sem セマフォの再初期化[T] 【NGKI1526】

### 【C言語API】

```
ER ercd = ini_sem(ID semid)
```

### 【パラメータ】

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

### 【リターンパラメータ】

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

## 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し 【NGKI1527】<br>・CPUロック状態からの呼出し 【NGKI1528】 |
| E_ID    | 不正ID番号<br>・semidが有効範囲外 【NGKI1529】                                       |
| E_NOEXS | オブジェクト未登録<br>・対象セマフォが未登録 [D] 【NGKI1530】                                 |
| E_OACV  | オブジェクトアクセス違反<br>・対象セマフォに対する管理操作が許可されていない [P]<br>【NGKI1531】              |

## 【機能】

semidで指定したセマフォ（対象セマフォ）を再初期化する。具体的な振舞いは以下の通り。

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

## 【使用上の注意】

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。

セマフォを再初期化した場合に、アプリケーションとの整合性を保つのは、アプリケーションの責任である。

## 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである。

ref\_sem セマフォの状態参照 [T] 【NGKI1535】

## 【C言語API】

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

## 【パラメータ】

|          |         |                         |
|----------|---------|-------------------------|
| ID       | semid   | 対象セマフォのID番号             |
| T_RSEM * | pk_rsem | セマフォの現在状態を入れるパケットへのポインタ |

### 【リターンパラメータ】

ER ercd 正常終了 (E\_OK) またはエラーコード

\*セマフォの現在状態 (パケットの内容)

|        |        |                       |
|--------|--------|-----------------------|
| ID     | wtskid | セマフォの待ち行列の先頭のタスクのID番号 |
| uint_t | semcnt | セマフォの資源数              |

### 【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー  |
|         | ・非タスクコンテキストからの呼び出し 【NGKI1536】                      |
|         | ・CPUロック状態からの呼び出し 【NGKI1537】                        |
| E_ID    | 不正ID番号   |
|         | ・semidが有効範囲外 【NGKI1538】                            |
| E_NOEXS | オブジェクト未登録  |
|         | ・対象セマフォが未登録 [D] 【NGKI1539】                         |
| E_OACV  | オブジェクトアクセス違反                                       |
|         | ・対象セマフォに対する参照操作が許可されていない [P] 【NGKI1540】            |
| E_MACV  | メモリアクセス違反  |
|         | ・pk_rsem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] 【NGKI1541】 |

### 【機能】

semidで指定したセマフォ（対象セマフォ）の現在状態を参照する。参照した現在状態は、pk\_rsemで指定したパケットに返される【NGKI1542】。

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

### 【使用上の注意】

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

## 1.4.2. イベントフラグ

イベントフラグは、イベントの発生の有無を表すビットの集合（ビットパターン）を介して、イベント通知を行うための同期・通信オブジェクトである。イベントが発生している状態を

1, 発生していない状態を0とし, ビットパターンにより複数のイベントの発生の有無を表す【NGKI1544】. イベントフラグは, イベントフラグIDと呼ぶID番号によって識別する【NGKI1545】.

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

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

各イベントフラグが持つ情報は次の通り【NGKI1548】.

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

待ち行列は, イベントフラグが指定した待ち解除の条件を満たすまで待っている状態(イベントフラグ待ち状態)のタスクがつながれているキューである. 待ち行列につながれたタスクの待ち解除は, 待ち解除の条件を満たした中で, 待ち行列の前方につながれたものから順に行われる(【NGKI0216】に該当)【NGKI1549】.

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

イベントフラグ属性には, 次の属性を指定することができる【NGKI1550】.

|         |       |                         |
|---------|-------|-------------------------|
| TA_TPRI | 0x01U | 待ち行列をタスクの優先度順にする        |
| TA_WMUL | 0x02U | 複数のタスクが待つのを許す           |
| TA_CLR  | 0x04U | タスクの待ち解除時にイベントフラグをクリアする |

TA\_TPRIを指定しない場合, 待ち行列はFIFO順になる【NGKI1551】. TA\_WMULを1つのイベントフラグに複数のタスクが待つことを禁止する【NGKI1552】. 指定しない場合,

TA\_CLRを指定した場合, タスクの待ち解除時に, イベントフラグのビットパターンを0にクリアする【NGKI1553】. TA\_CLRを指定しない場合, タスクの待ち解除時にイベントフラグをクリアしない【NGKI1554】.

イベントフラグ機能に用いるデータ型は次の通り.

FLGPTN イベントフラグのビットパターン（符号無し整数, uint\_tに定義）【NGKI1555】

イベントフラグ機能に関連するカーネル構成マクロは次の通り.

TBIT\_FLGPTN イベントフラグのビット数（FLGPTNの有効ビット数）【NGKI1556】

TNUM\_FLGID 登録できるイベントフラグの数（動的生成対応でないカーネルでは、静的APIによって登録されたイベントフラグの数に一致）【NGKI1557】

#### 【TOPPERS/ASPカーネルにおける規定】

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【ASPS0124】.

#### 【TOPPERS/FMPカーネルにおける規定】

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【FMPSS0115】.

#### 【TOPPERS/HRP2カーネルにおける規定】

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【HRPS0115】.

#### 【μITRON4.0仕様との関係】

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

CRE\_FLG イベントフラグの生成 [S] 【NGKI1558】  
acre\_flg イベントフラグの生成 [TD] 【NGKI1559】

#### 【静的API】

```
CRE_FLG(ID flgid, { ATR flgatr, FLGPTN iflgptn })
```

#### 【C言語API】

```
ER_ID flgid = acre_flg(const T_CFLG *pk_cflg)
```

#### 【パラメータ】

|          |         |                                      |
|----------|---------|--------------------------------------|
| ID       | flgid   | 生成するイベントフラグのID番号（CRE_FLGの場合）         |
| T_CFLG * | pk_cflg | イベントフラグの生成情報を入ったパケットへのポインタ（静的APIを除く） |

\*イベントフラグの生成情報（パケットの内容）

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

#### 【リターンパラメータ】

|       |       |                                     |
|-------|-------|-------------------------------------|
| ER_ID | flgid | 生成されたイベントフラグのID番号（正の値）<br>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び出し [s] 【NGKI1560】               |
|         | ・CPUロック状態からの呼び出し [s] 【NGKI1561】                 |
| E_RSATR | 予約属性  |
|         | ・flgatrが無効 【NGKI1562】                           |
|         | ・属する保護ドメインの指定が有効範囲外 [sP] 【NGKI1563】             |
|         | ・属するクラスの指定が有効範囲外 [sM] 【NGKI1564】                |
|         | ・クラスの囲みの中に記述されていない [SM] 【NGKI1565】              |
| E_PAR   | パラメータエラー  |
|         | ・iflgptnがFLGPTNに格納できない [S] 【NGKI3275】           |
| E_OACV  | オブジェクトアクセス違反                                    |
|         | ・システム状態に対する管理操作が許可されていない [sP] 【NGKI1566】        |
| E_MACV  | メモリアクセス違反                                       |
|         | ・pk_cflgが指すメモリ領域への読み出しが許可されていない [sP] 【NGKI1567】 |
| E_NOID  | ID番号不足  |
|         | ・割り付けられるイベントフラグIDがない [sD] 【NGKI1568】            |
| E_OBJ   | オブジェクト状態エラー                                     |
|         | ・flgidで指定したイベントフラグが登録済み（CRE_FLGの場合）【NGKI1569】   |

#### 【機能】

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に、待ち行列は空の状態に初期化される【NGKI1570】。

静的APIにおいては、flgidはオブジェクト識別名、flgatrとiflgptnは整数定数

式パラメータである【NGKI1571】。

#### 【TOPPERS/ASPカーネルにおける規定】

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】。

#### 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、CRE\_FLGのみをサポートする【FMPMS0116】。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、CRE\_FLGのみをサポートする【HRPS0116】。ただし、動的生成機能拡張パッケージでは、acre\_flgもサポートする【HRPS0183】。

AID\_FLG 割付け可能なイベントフラグIDの数の指定 [SD] 【NGKI1572】

#### 【静的API】

AID\_FLG(uint\_t n oflg)

#### 【パラメータ】

uint\_t n oflg 割付け可能なイベントフラグIDの数

#### 【エラーコード】

E\_RSATR

予約属性

- ・保護ドメインの囲みの中に記述されている [P] 【NGKI3430】
- ・クラスの囲みの中に記述されていない [M] 【NGKI1573】

E\_PAR

パラメータエラー

- ・noflgが負の値【NGKI3278】

#### 【機能】

noflgで指定した数のイベントフラグIDを、イベントフラグを生成するサービスコールによって割付け可能なイベントフラグIDとして確保する【NGKI1574】。

noflgは整数定数式パラメータである【NGKI1575】。

#### 【TOPPERS/ASPカーネルにおける規定】

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

## 【TOPPERS/HRP2カーネルにおける規定】

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

|         |                                       |
|---------|---------------------------------------|
| SAC_FLG | イベントフラグのアクセス許可ベクタの設定 [SP] 【NGKI1576】  |
| sac_flg | イベントフラグのアクセス許可ベクタの設定 [TPD] 【NGKI1577】 |

## 【静的API】

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

## 【C言語API】

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

## 【パラメータ】

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

\* アクセス許可ベクタ（パケットの内容）

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

## 【リターンパラメータ】

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

## 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し [s] 【NGKI1578】<br>・CPUロック状態からの呼出し [s] 【NGKI1579】   |
| E_ID    | 不正ID番号<br>・flgidが有効範囲外 [s] 【NGKI1580】   |
| E_RSATR | 予約属性<br>・対象イベントフラグが属する保護ドメインの囲みの中（対象イベントフラグが無所属の場合は、保護ドメインの囲みの外）に記述されていない [S] 【NGKI1581】<br>・対象イベントフラグが属するクラスの囲みの中に記述されていない [SM] 【NGKI1582】 |
| E_NOEXS | オブジェクト未登録<br>・対象イベントフラグが未登録 【NGKI1583】  |
| E_OACV  | オブジェクトアクセス違反<br>・対象イベントフラグに対する管理操作が許可されていない [s] 【NGKI1584】  |
| E_MACV  | メモリアクセス違反<br>・p_acvctが指すメモリ領域への読み出しが許可されていない [s] 【NGKI1585】   |
| E_OBJ   | オブジェクト状態エラー<br>・対象イベントフラグは静的APIで生成された [s] 【NGKI1586】<br>・対象イベントフラグに対してアクセス許可ベクタが設定済み [S] 【NGKI1587】   |

## 【機能】

flgidで指定したイベントフラグ（対象イベントフラグ）のアクセス許可ベクタ（4つのアクセス許可パターンの組）を、各パラメータで指定した値に設定する 【NGKI1588】。

静的APIにおいては、flgidはオブジェクト識別名、acptn1～acptn4は整数定数式パラメータである 【NGKI1589】。

## 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、SAC\_FLGのみをサポートする 【HRPS0117】。ただし、動的生成機能拡張パッケージでは、sac\_flgもサポートする 【HRPS0184】。

del\_flg イベントフラグの削除 [TD] 【NGKI1590】

## 【C言語API】

```
ER ercd = del_flg(ID flgid)
```

## 【パラメータ】

|    |       |                |
|----|-------|----------------|
| ID | flgid | 対象イベントフラグのID番号 |
|----|-------|----------------|

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼び出し 【NGKI1591】<br>・CPUロック状態からの呼び出し 【NGKI1592】 |
| E_ID    | 不正ID番号<br>・flgidが有効範囲外 【NGKI1593】   |
| E_NOEXS | オブジェクト未登録<br>・対象イベントフラグが未登録 【NGKI1594】                                    |
| E_OACV  | オブジェクトアクセス違反<br>・対象イベントフラグに対する管理操作が許可されていない [P]<br>【NGKI1595】             |
| E_OBJ   | オブジェクト状態エラー<br>・対象イベントフラグは静的APIで生成された 【NGKI1596】                          |

#### 【機能】

flgidで指定したイベントフラグ（対象イベントフラグ）を削除する。具体的な振舞いは以下の通り。

対象イベントフラグの登録が解除され、そのイベントフラグIDが未使用の状態に戻される【NGKI1597】。また、対象イベントフラグの待ち行列につながれたタスクは、待ち行列の先頭のタスクから順に待ち解除される【NGKI1598】。待ち解除されたタスクには、待ち状態となったサービスコールからE\_DLTエラーが返る【NGKI1599】。

#### 【使用上の注意】

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

#### 【TOPPERS/ASPカーネルにおける規定】

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

#### 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、del\_flgをサポートしない【FMP0118】。

## 【TOPPERS/HRP2カーネルにおける規定】

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

|                       |                            |
|-----------------------|----------------------------|
| <code>set_flg</code>  | イベントフラグのセット [T] 【NGKI1600】 |
| <code>iset_flg</code> | イベントフラグのセット [I] 【NGKI1601】 |

## 【C言語API】

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

## 【パラメータ】

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

## 【リターンパラメータ】

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

## 【エラーコード】

|         |              |   |
|---------|--------------|---|
| E_CTX   | コンテキストエラー    | <ul style="list-style-type: none"><li>・非タスクコンテキストからの呼び出し (<code>set_flg</code>の場合) 【NGKI1602】</li><li>・タスクコンテキストからの呼び出し (<code>iset_flg</code>の場合) 【NGKI1603】</li><li>・CPUロック状態からの呼び出し 【NGKI1604】</li></ul> |
| E_ID    | 不正ID番号       | <ul style="list-style-type: none"><li>・<code>flgid</code>が有効範囲外 【NGKI1605】</li></ul>  |
| E_NOEXS | オブジェクト未登録    | <ul style="list-style-type: none"><li>・対象イベントフラグが未登録 [D] 【NGKI1606】</li></ul>   |
| E_OACV  | オブジェクトアクセス違反 | <ul style="list-style-type: none"><li>・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<br/>(<code>set_flg</code>の場合) [P] 【NGKI1607】</li></ul>   |

## 【機能】

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

対象イベントフラグのビットパターンは、それまでの値と`setptn`で指定した値（言語の"|"）に更新される【NGKI1608】。対象イベントフ

のビット毎論理和 (C

ラグの待ち行列にタスクが存在する場合には、待ち解除の条件を満たしたタスクが、待ち行列の前方につながれたものから順に待ち解除される【NGKI1609】。  
待ち解除されたタスクには、待ち状態となったサービスコールからE\_OKが返る【NGKI1610】。

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1つ待ち解除した時点で、対象イベントフラグのビットパターンが0にクリアされるため、他のタスクが待ち解除されることはない。

### 【使用上の注意】

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、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。

`clr_flg` イベントフラグのクリア [T] 【NGKI1611】

### 【C言語API】

```
ER ercd = clr_flg(ID_flgid, FLGPTN_clrptn)
```

### 【パラメータ】

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

### 【リターンパラメータ】

ER ercd 正常終了 (E\_OK) またはエラーコード

### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI1612】<br>・CPUロック状態からの呼出し【NGKI1613】 |
| E_ID    | 不正ID番号<br>・flgidが有効範囲外【NGKI1614】                                      |
| E_NOEXS | オブジェクト未登録<br>・対象イベントフラグが未登録〔D〕【NGKI1615】                              |
| E_OACV  | オブジェクトアクセス違反<br>・対象イベントフラグに対する通常操作1が許可されていない〔P〕<br>【NGKI1616】         |

### 【機能】

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

|          |                                  |
|----------|----------------------------------|
| wai_flg  | イベントフラグ待ち〔T〕【NGKI1618】           |
| pol_flg  | イベントフラグ待ち（ポーリング）〔T〕【NGKI1619】    |
| twai_flg | イベントフラグ待ち（タイムアウト付き）〔T〕【NGKI1620】 |

### 【C言語API】

```
ER ercd = wai_flg(ID flgid, FLGPTN waiptr, MODE wfmode, FLGPTN *p_flgptn)
ER ercd = pol_flg(ID flgid, FLGPTN waiptr, MODE wfmode, FLGPTN *p_flgptn)
ER ercd = twai_flg(ID flgid, FLGPTN waiptr,
                    MODE wfmode, FLGPTN *p_flgptn, TMO tmout)
```

### 【パラメータ】

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

### 【リターンパラメータ】

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

## 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し 【NGKI1621】<br>・CPUロック状態からの呼出し 【NGKI1622】<br>・ディスパッチ保留状態からの呼出し (pol_flgを除く) 【NGKI1623】    |
| E_NOSPT | 未サポート機能<br>・制約タスクからの呼出し (pol_flgを除く) 【NGKI1624】   |
| E_ID    | 不正ID番号<br>・flgidが有効範囲外 【NGKI1625】   |
| E_PAR   | パラメータエラー<br>・waiptnが0 【NGKI1626】<br>・wfmodeが無効 (TWF_ORWまたはTWF_ANDWでない) 【NGKI1627】<br>・tmoutが無効 (twai_flgの場合) 【NGKI1628】 |
| E_NOEXS | オブジェクト未登録<br>・対象イベントフラグが未登録 [D] 【NGKI1629】  |
| E_OACV  | オブジェクトアクセス違反<br>・対象イベントフラグに対する通常操作2が許可されていない [P]<br>【NGKI1630】  |
| E_MACV  | メモリアクセス違反<br>・p_flgptnが指すメモリ領域への書き込みアクセスが許可され<br>ていない [P] 【NGKI1631】  |
| E_ILUSE | サービスコール不正使用<br>・TA_WMUL属性でないイベントフラグで待ちタスクあり 【NGKI1632】  |
| E_TMOUT | ポーリング失敗またはタイムアウト (wai_flgを除く) 【NGKI1633】  |
| E_RLWAI | 待ち禁止状態または待ち状態の強制解除 (pol_flgを除く)<br>【NGKI1634】   |
| 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)<br>【NGKI1635】   |

## 【機能】

flgidで指定したイベントフラグ（対象イベントフラグ）が、waiptnとwfmodeで指定した待ち解除の条件を満たすのを待つ。具体的な振舞いは以下の通り。

対象イベントフラグが、waiptnとwfmodeで指定した待ち解除の条件を満たしている場合には、対象イベントフラグのビットパターンの現在値がp\_flgptnが指すメモリ領域に返される【NGKI1636】。対象イベントフラグがTA\_CLR属性である場合には、対象イベントフラグのビットパターンが0にクリアされる【NGKI1637】。

待ち解除の条件を満たしていない場合には、自タスクはイベントフラグ待ち状態となり、対象イベントフラグの待ち行列につながれる【NGKI1638】。

ini\_flg イベントフラグの再初期化 [T] 【NGKI1639】

## 【C言語API】

```
ER ercd = ini_flg(ID flgid)
```

#### 【パラメータ】

|    |       |                |
|----|-------|----------------|
| ID | flgid | 対象イベントフラグのID番号 |
|----|-------|----------------|

#### 【リターンパラメータ】

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

#### 【エラーコード】

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

#### 【機能】

flgidで指定したイベントフラグ（対象イベントフラグ）を再初期化する。具体的な振舞いは以下の通り。

対象イベントフラグのビットパターンは、初期ビットパターンに初期化される  
【NGKI1645】。また、対象イベントフラグの待ち行列につながれたタスクは、  
待ち行列の先頭のタスクから順に待ち解除される【NGKI1646】。待ち解除され  
たタスクには、待ち状態となったサービスコールからE\_DLTエラーが返る 【NGKI1647】。

#### 【使用上の注意】

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

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

#### 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである。

`ref_flg` イベントフラグの状態参照 [T] 【NGKI1648】

【C言語API】

```
ER ercd = ref_flg(ID_flgid, T_RFLG *pk_rflg)
```

【パラメータ】

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

【リターンパラメータ】

ER ercd 正常終了 (E\_OK) またはエラーコード

\*イベントフラグの現在状態 (パケットの内容)

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

【エラーコード】

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

【機能】

flgidで指定したイベントフラグ（対象イベントフラグ）の現在状態を参照する。

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

対象イベントフラグの待ち行列にタスクが存在しない場合、wtskidには

TSK\_NONE (=0

) が返る【NGKI1656】.

#### 【使用上の注意】

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

### 1.4.3. データキュー

データキューは、1ワードのデータをメッセージとして、FIFO順で送受信するための同期・通信オブジェクトである。より大きいサイズのメッセージを送受信したい場合には、メッセージを置いたメモリ領域へのポインタを1ワードのデータとして送受信する方法がある。データキューは、データキューIDと呼ぶID番号によって識別する【NGKI1657】。

各データキューが持つ情報は次の通り【NGKI1658】。

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

データキュー管理領域は、データキューに送信されたデータを、送信された順に格納しておくためのメモリ領域である。データキュー生成時に、データキュー管理領域に格納できるデータ数を0とすることで、データキュー管理領域のサイズを0とすることができます【NGKI1659】。

保護機能対応カーネルにおいて、データキュー管理領域は、カーネルの用いるオブジェクト管理領域として扱われる【NGKI1660】。

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

データキュー属性には、次の属性を指定することができる【NGKI1661】。

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

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

データキュー機能に関連するカーネル構成マクロは次の通り.

|            |  |
|------------|--|
| TNUM_DTQID | 登録できるデータキューの数（動的生成対応でないカーネルでは、静的APIによって登録されたデータキューの数に一致）【NGKI1664】 |
|------------|--|

#### 【μITRON4.0仕様との関係】

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

|          |                           |
|----------|---------------------------|
| CRE_DTQ  | データキューの生成 [S] 【NGKI1665】  |
| acre_dtq | データキューの生成 [TD] 【NGKI1666】 |

#### 【静的API】

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

#### 【C言語API】

```
ER_ID dtqid = acre_dtq(const T_CDTQ *pk_cdtq)
```

#### 【パラメータ】

|          |         |  |
|----------|---------|--|
| ID       | dtqid   | 生成するデータキューのID番号 (CRE_DTQの場合)             |
| T_CDTQ * | pk_cdtq | データキューの生成情報を入ったパケットへの<br>ポインタ (静的APIを除く) |

\*データキューの生成情報 (パケットの内容)

|        |        |                      |
|--------|--------|----------------------|
| ATR    | dtqatr | データキュー属性             |
| uint_t | dtqcnt | データキュー管理領域に格納できるデータ数 |
| void * | dtqmb  | データキュー管理領域の先頭番地      |

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し [s] 【NGKI1667】<br>・CPUロック状態からの呼出し [s] 【NGKI1668】  |
| E_RSATR | 予約属性<br>・dtqatrが無効 【NGKI1669】<br>・属する保護ドメインの指定が有効範囲外 [sP] 【NGKI1670】<br>・属するクラスの指定が有効範囲外 [sM] 【NGKI1671】<br>・クラスの団みの中に記述されていない [SM] 【NGKI1672】 |
| E_NOSPT | 未サポート機能<br>・条件については各カーネルにおける規定の項を参照  |
| E_PAR   | パラメータエラー<br>・dtqcntが負の値 [S] 【NGKI3288】<br>・その他の条件については機能の項を参照  |
| E_OACV  | オブジェクトアクセス違反<br>・システム状態に対する管理操作が許可されていない [sP]<br>【NGKI1673】  |
| E_MACV  | メモリアクセス違反<br>・pk_cdtqが指すメモリ領域への読み出しが許可されて<br>いない [sP] 【NGKI1674】   |
| E_NOID  | ID番号不足<br>・割り付けられるデータキューIDがない [sD] 【NGKI1675】  |
| E_NOMEM | メモリ不足<br>・データキュー管理領域が確保できない 【NGKI1676】   |
| E_OBJ   | オブジェクト状態エラー<br>・dtqidで指定したデータキューが登録済み (CRE_DTQの場合)<br>【NGKI1677】<br>・その他の条件については機能の項を参照  |

## 【機能】

各パラメータで指定したデータキュー生成情報に従って、データキューを生成  
dtqmbからデータキュー管理領域が設定され、格納されているデー  
タがない状態に初期化される【NGKI1678】。また、送信待ち行列と受信待ち行  
列は、空の状態に初期化される【NGKI1679】。

静的APIにおいては、dtqidはオブジェクト識別名、dtqatrとdtqcntは整数定数  
dtqmbは一般定数式パラメータである【NGKI1680】。コンフィギュ  
のメモリ不足 (E\_NOMEM) エラーを検出することができない 【NGKI1681】。

dtqmb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  
管理領域が、コンフィギュレータまたはカーネルにより確保される【NGKI1682】。

### 〔dtqmbにNULL以外を指定した場合〕

dtqmbにNULL以外を指定した場合、dtqmbを先頭番地とするデータキュー管理領  
域は、アプリケーションで確保しておく必要がある【NGKI1683】。データキュー

する。dtqcntと

式パラメータ,  
レータは、静的API

管理領域をアプリケーションで確保するために、次のマクロを用意している【NGKI1684】。

|                    |   |
|--------------------|---|
| TSZ_DTQMB(dtqcnt)  | dtqcntで指定した数のデータを格納できるデータキュー管理領域のサイズ（バイト数）              |
| TCNT_DTQMB(dtqcnt) | dtqcntで指定した数のデータを格納できるデータキュー管理領域を確保するために必要なMB_T型の配列の要素数 |

これらを用いてデータキュー管理領域を確保する方法は次の通り【NGKI1685】。

MB\_T<データキュー管理領域の変数名>[TCNT\_DTQMB(dtqcnt)];

この時、dtqmbには<データキュー管理領域の変数名>を指定する【NGKI1686】。

この方法に従わず、dtqmbにターゲット定義の制約に合致しない先頭番地を指定E\_PARエラーとなる【NGKI1687】。また、保護機能対応カーネルにdtqmbで指定したデータキュー管理領域がカーネル専用のメモリオブジェクトE\_OBJエラーとなる【NGKI1688】。

した時には、  
おいて、

クトに含まれない場合、

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、CRE\_DTQのみをサポートする【ASPS0130】。また、dtqmbにはNULLのみを指定することができる。NULL以外を指定した場合には、E\_NOSPTエラーとなる【ASPS0132】。ただし、動的生成機能拡張パッケージでは、acre\_dtqもサポートする【ASPS0133】。acre\_dtqに対しては、dtqmbにNULL以外を指定できないという制限はない【ASPS0134】。

#### 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、CRE\_DTQのみをサポートする【FMPS0119】。また、dtqmbにはNULLのみを指定することができる。NULL以外を指定した場合には、E\_NOSPTエラーとなる【FMPS0121】。

#### 【TOPPERS/HRP2カーネルにおける規定】

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

#### 【μITRON4.0仕様との関係】

μITRON4.0/PX仕様にあわせて、データキュー生成情報の最後のパラメータを、dtq（データキュー領域の先頭番地）から、dtqmb（データキュー管理領域の先頭番地）に改名した。また、TSZ\_DTQをTSZ\_DTQMBに改名した。

TCNT\_DTQMBを新設し、データキュー管理領域をアプリケーションで確保する方法を規定した。

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

【静的API】

AID\_DTQ(uint\_t nodtq)

【パラメータ】

uint\_t nodtq 割付け可能なデータキューIDの数

【エラーコード】

E\_RSATR 予約属性

- ・保護ドメインの囲みの中に記述されている〔P〕【NGKI3431】
- ・クラスの囲みの中に記述されていない〔M〕【NGKI1690】

E\_PAR パラメータエラー

- ・nodtqが負の値【NGKI3279】

【機能】

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

nodtqは整数定数式パラメータである【NGKI1692】。

【TOPPERS/ASPカーネルにおける規定】

ASPカーネルの動的生成機能拡張パッケージでは、AID\_DTQをサポートする【ASPS0213】。

【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルの動的生成機能拡張パッケージでは、AID\_DTQをサポートする【HRPS0214】。

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

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

【静的API】

SAC\_DTQ(ID dtqid, { ACPTN acptn1, ACPTN acptn2,  
ACPTN acptn3, ACPTN acptn4 })

【C言語API】

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

#### 【パラメータ】

|               |                  |  |
|---------------|------------------|--|
| ID<br>ACVCT * | dtqid<br>p_acvct | 対象データキューのID番号<br>アクセス許可ベクタを入れたパケットへのポ<br>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び出し [s] 【NGKI1695】  |
|         | ・CPUロック状態からの呼び出し [s] 【NGKI1696】  |
| E_ID    | 不正ID番号   |
|         | ・dtqidが有効範囲外 [s] 【NGKI1697】  |
| E_RSATR | 予約属性   |
|         | ・対象データキューが属する保護ドメインの囲みの中（対象<br>データキューが無所属の場合は、保護ドメインの囲みの外）<br>に記述されていない [S] 【NGKI1698】 |
|         | ・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<br>いない [SM] 【NGKI1699】                                     |
| E_NOEXS | オブジェクト未登録  |
|         | ・対象データキューが未登録 【NGKI1700】   |
| E_OACV  | オブジェクトアクセス違反   |
|         | ・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]<br>【NGKI1701】   |
| E_MACV  | メモリアクセス違反  |
|         | ・p_acvctが指すメモリ領域への読み出しが許可されて<br>いない [s] 【NGKI1702】                                     |
| E_OBJ   | オブジェクト状態エラー  |
|         | ・対象データキューは静的APIで生成された [s] 【NGKI1703】   |
|         | ・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]<br>【NGKI1704】  |

## 【機能】

dtqidで指定したデータキュー（対象データキュー）のアクセス許可ベクタ（4つのアクセス許可パターンの組）を、各パラメータで指定した値に設定する 【NGKI1705】。

静的APIにおいては、 dtqidはオブジェクト識別名、 acptn1～acptn4は整数定数式パラメータである 【NGKI1706】。

## 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、 SAC\_DTQのみをサポートする 【HRPS0122】。ただし、動的生成機能拡張パッケージでは、 sac\_dtqもサポートする 【HRPS0188】。

`del_dtq` データキューの削除 [TD] 【NGKI1707】

## 【C言語API】

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

## 【パラメータ】

ID dtqid 対象データキューのID番号

## 【リターンパラメータ】

ER ercd 正常終了 (E\_OK) またはエラーコード

## 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー                                 |
|         | ・非タスクコンテキストからの呼び出し 【NGKI1708】             |
|         | ・CPUロック状態からの呼び出し 【NGKI1709】               |
| E_ID    | 不正ID番号                                    |
|         | ・dtqidが有効範囲外 【NGKI1710】                   |
| E_NOEXS | オブジェクト未登録                                 |
|         | ・対象データキューが未登録 【NGKI1711】                  |
| E_OACV  | オブジェクトアクセス違反                              |
|         | ・対象データキューに対する管理操作が許可されていない [P] 【NGKI1712】 |
| E_OBJ   | オブジェクト状態エラー                               |
|         | ・対象データキューは静的APIで生成された 【NGKI1713】          |

## 【機能】

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

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

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

#### 【補足説明】

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

#### 【使用上の注意】

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

#### 【TOPPERS/ASPカーネルにおける規定】

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

#### 【TOPPERS/FMPカーネルにおける規定】

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

#### 【TOPPERS/HRP2カーネルにおける規定】

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

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

#### 【C言語API】

```
ER ercd = snd_dtq(ID dtqid, intptr_t data)
ER ercd = psnd_dtq(ID dtqid, intptr_t data)
ER ercd = ipsnd_dtq(ID dtqid, intptr_t data)
ER ercd = tsnd_dtq(ID dtqid, intptr_t data, TMO tmout)
```

### 【パラメータ】

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

### 【リターンパラメータ】

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

### 【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー  |
|         | ・非タスクコンテキストからの呼出し (ipsnd_dtqを除く)<br>【NGKI1722】               |
|         | ・タスクコンテキストからの呼出し (ipsnd_dtqの場合)<br>【NGKI1723】                |
|         | ・CPUロック状態からの呼出し 【NGKI1724】                                   |
|         | ・ディスパッチ保留状態からの呼出し (snd_dtqとtsnd_dtqの場合) 【NGKI1725】           |
| E_NOSPT | 未サポート機能  |
|         | ・制約タスクからの呼出し (snd_dtqとtsnd_dtqの場合) 【NGKI1726】                |
| E_ID    | 不正ID番号   |
|         | ・dtqidが有効範囲外 【NGKI1727】                                      |
| E_PAR   | パラメータエラー   |
|         | ・tmoutが無効 (tsnd_dtqの場合) 【NGKI1728】                           |
| E_NOEXS | オブジェクト未登録  |
|         | ・対象データキューが未登録 [D] 【NGKI1729】                                 |
| E_OACV  | オブジェクトアクセス違反   |
|         | ・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<br>(ipsnd_dtqを除く) [P] 【NGKI1730】 |
| E_TMOUT | ポーリング失敗またはタイムアウト (snd_dtqを除く) 【NGKI1731】                     |
| E_RLWAI | 待ち禁止状態または待ち状態の強制解除 (snd_dtqとtsnd_dtqの場合) 【NGKI1732】          |
| E_DLT   | 待ちオブジェクトの削除または再初期化 (snd_dtqとtsnd_dtqの場合) 【NGKI1733】          |

### 【機能】

`dtqid`で指定したデータキュー（対象データキュー）に、`data`で指定したデータを送信する。具体的な振舞いは以下の通り。

対象データキューの受信待ち行列にタスクが存在する場合には、受信待ち行列  
dataで指定したデータを受信し、待ち解除される  
【NGKI1734】。待ち解除されたタスクには、待ち状態となったサービスコール  
【NGKI1735】。

対象データキューの受信待ち行列にタスクが存在せず、データキュー管理領域  
にデータを格納するスペースがある場合には、`data`で指定したデータが、FIFO  
順でデータキュー管理領域に格納される【NGKI1736】。

対象データキューの受信待ち行列にタスクが存在せず、データキュー管理領域  
にデータを格納するスペースがない場合には、自タスクはデータキューへの送  
信待ち状態となり、対象データキューの送信待ち行列につながれる【NGKI1737】。

`fsnd_dtq` データキューへの強制送信 [T] 【NGKI1738】  
`ifsnd_dtq` データキューへの強制送信 [I] 【NGKI1739】

#### 【C言語API】

```
ER ercd = fsnd_dtq(ID dtqid, intptr_t data)
ER ercd = ifsnd_dtq(ID dtqid, intptr_t data)
```

#### 【パラメータ】

|                       |                    |               |
|-----------------------|--------------------|---------------|
| ID                    | <code>dtqid</code> | 対象データキューのID番号 |
| <code>intptr_t</code> | <code>data</code>  | 送信データ         |

#### 【リターンパラメータ】

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

#### 【エラーコード】

の先頭のタスクが、  
【  
からE\_OKが返る【

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し (fsnd_dtqの場合) 【NGKI1740】<br>・タスクコンテキストからの呼出し (ifsnd_dtqの場合) 【NGKI1741】<br>・CPUロック状態からの呼出し 【NGKI1742】 |
| E_ID    | 不正ID番号<br>・dtqidが有効範囲外 【NGKI1743】   |
| E_NOEXS | オブジェクト未登録<br>・対象データキューが未登録 [D] 【NGKI1744】   |
| E_OACV  | オブジェクトアクセス違反<br>・対象データキューに対する通常操作1が許可されていない<br>(fsnd_dtqの場合) [P] 【NGKI1745】   |
| E_ILUSE | サービスコール不正使用<br>・対象データキューのデータキュー管理領域のサイズが0 【NGKI1746】  |

## 【機能】

dtqidで指定したデータキュー（対象データキュー）に、dataで指定したデータを強制送信する。具体的な振舞いは以下の通り。

対象データキューの受信待ち行列にタスクが存在する場合には、受信待ち行列  
dataで指定したデータを受信し、待ち解除される

の先頭のタスクが、  
【  
からE\_OKが返る【

NGKI1747】。待ち解除されたタスクには、待ち状態となったサービスコール  
NGKI1748】。

対象データキューの受信待ち行列にタスクが存在せず、データキュー管理領域  
にデータを格納するスペースがある場合には、dataで指定したデータが、FIFO  
順でデータキュー管理領域に格納される【NGKI1749】。

対象データキューの受信待ち行列にタスクが存在せず、データキュー管理領域  
にデータを格納するスペースがない場合には、データキュー管理領域の先頭に  
格納されたデータを削除し、空いたスペースを用いて、dataで指定したデータ  
FIFO順でデータキュー管理領域に格納される【NGKI1750】。

が、

|          |                                     |
|----------|-------------------------------------|
| recv_dtq | データキューからの受信 [T] 【NGKI1751】          |
| prcv_dtq | データキューからの受信（ポーリング）[T] 【NGKI1752】    |
| trcv_dtq | データキューからの受信（タイムアウト付き）[T] 【NGKI1753】 |

## 【C言語API】

```
ER ercd = recv_dtq(ID dtqid, intptr_t *p_data)
ER ercd = prcv_dtq(ID dtqid, intptr_t *p_data)
ER ercd = trcv_dtq(ID dtqid, intptr_t *p_data, TMO tmout)
```

## 【パラメータ】

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

### 【リターンパラメータ】

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

### 【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し 【NGKI1754】<br>・CPUロック状態からの呼出し 【NGKI1755】<br>・ディスパッチ保留状態からの呼出し (recv_dtqを除く)<br>【NGKI1756】 |
| E_NOSPT | 未サポート機能<br>・制約タスクからの呼出し (recv_dtqを除く) 【NGKI1757】   |
| E_ID    | 不正ID番号<br>・dtqidが有効範囲外 【NGKI1758】  |
| E_PAR   | パラメータエラー<br>・tmoutが無効 (recv_dtqの場合) 【NGKI1759】   |
| E_NOEXS | オブジェクト未登録<br>・対象データキューが未登録 [D] 【NGKI1760】  |
| E_OACV  | オブジェクトアクセス違反<br>・対象データキューに対する通常操作2が許可されていない [P]<br>【NGKI1761】  |
| E_MACV  | メモリアクセス違反<br>・p_dataが指すメモリ領域への書き込みアクセスが許可されて<br>いない [P] 【NGKI1762】   |
| E_TMOUT | ポーリング失敗またはタイムアウト (recv_dtqを除く) 【NGKI1763】  |
| E_RLWAI | 待ち禁止状態または待ち状態の強制解除 (recv_dtqを除く)<br>【NGKI1764】   |
| E_DLT   | 待ちオブジェクトの削除または再初期化 (recv_dtqを除く)<br>【NGKI1765】   |

### 【機能】

dtqidで指定したデータキュー（対象データキュー）からデータを受信する。データの受信に成功した場合、受信したデータはp\_dataが指すメモリ領域に返される【NGKI3421】。具体的な振舞いは以下の通り。

対象データキューのデータキュー管理領域にデータが格納されている場合には、データキュー管理領域の先頭に格納されたデータを受信する【NGKI1766】。また、送信待ち行列にタスクが存在する場合には、送信待ち行列の先頭のタスク

の送信データが、

FIFO順でデータキュー管理領域に格納され、そのタスクは待ち解除される【  
NGKI1767】。待ち解除されたタスクには、待ち状態となったサービスコールからE\_OKが返る【  
NGKI1768】。

対象データキューのデータキュー管理領域にデータが格納されておらず、送信待ち行列にタスクが存在する場合には、送信待ち行列の先頭のタスクの送信データを受信する【  
NGKI1769】。送信待ち行列の先頭のタスクは、待ち解除される【  
NGKI3422】。待ち解除されたタスクには、待ち状態となったサービスコールからE\_OKが返る【  
NGKI1770】。

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

**ini\_dtq** データキューの再初期化 [T] 【NGKI1772】

【C言語API】

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

【パラメータ】

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

【リターンパラメータ】

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

【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼び出し【NGKI1773】<br>・CPUロック状態からの呼び出し【NGKI1774】 |
| E_ID    | 不正ID番号<br>・dtqidが有効範囲外【NGKI1775】  |
| E_NOEXS | オブジェクト未登録<br>・対象データキューが未登録 [D] 【NGKI1776】                               |
| E_OACV  | オブジェクトアクセス違反<br>・対象データキューに対する管理操作が許可されていない [P]<br>【NGKI1777】            |

【機能】

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

対象データキューのデータキュー管理領域は、格納されているデータがない状態に初期化される【NGKI1778】。また、対象データキューの送信待ち行列と受信待ち行列につながれたタスクは、それぞれの待ち行列の先頭のタスクから順に待ち解除される【NGKI1779】。待ち解除されたタスクには、待ち状態となつたサービスコールからE\_DLTエラーが返る【NGKI1780】。

#### 【補足説明】

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

#### 【使用上の注意】

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

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

#### 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである。

`ref_dtq` データキューの状態参照 [T] 【NGKI1781】

#### 【C言語API】

```
ER ercd = ref_dtq(ID dtqid, T_RDTQ *pk_rdtq)
```

#### 【パラメータ】

|                       |                      |                           |
|-----------------------|----------------------|---------------------------|
| ID                    | <code>dtqid</code>   | 対象データキューのID番号             |
| <code>T_RDTQ *</code> | <code>pk_rdtq</code> | データキューの現在状態を入れるパケットへのポインタ |

#### 【リターンパラメータ】

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

\*データキューの現在状態 (/パケットの内容)

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

### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>· 非タスクコンテキストからの呼び出し 【NGKI1782】<br>· CPUロック状態からの呼び出し 【NGKI1783】 |
| E_ID    | 不正ID番号<br>· dtqidが有効範囲外 【NGKI1784】  |
| E_NOEXS | オブジェクト未登録<br>· 対象データキューが未登録 [D] 【NGKI1785】                                  |
| E_OACV  | オブジェクトアクセス違反<br>· 対象データキューに対する参照操作が許可されていない [P]<br>【NGKI1786】               |
| E_MACV  | メモリアクセス違反<br>· pk_rdtqが指すメモリ領域への書き込みアクセスが許可されていない [P] 【NGKI1787】            |

### 【機能】

dtqidで指定したデータキュー（対象データキュー）の現在状態を参照する。参考照した現在状態は、  
pk\_rdtqで指定したパケットに返される【NGKI1788】。

対象データキューの送信待ち行列にタスクが存在しない場合、stskidには TSK\_NONE (=0)  
が返る【NGKI1789】。また、受信待ち行列にタスクが存在しない場合、rtskidにはTSK\_NONE (=0)  
が返る【NGKI1790】。

### 【使用上の注意】

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

## 1.4.4. 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ID番号によって識別する【NGKI1791】。

は、優先度データキューIDと呼ぶ

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とすることができます【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】。

ち行列は、

優先度データキュー機能に関連するカーネル構成マクロは次の通り。

TMIN\_DPRI データ優先度の最小値（=1）【NGKI1798】  
TMAX\_DPRI データ優先度の最大値

TNUM\_PDQID 登録できる優先度データキューの数（動的生成対応でないカーネルでは、静的APIによって登録された優先度データキューの数に一致）【NGKI1799】

## 【TOPPERS/ASPカーネルにおける規定】

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

## 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、データ優先度の最大値（TMAX\_DPRI）は16に固定されている【FMPS0124】。

## 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、データ優先度の最大値（TMAX\_DPRI）は16に固定されている【HRPS0124】。

## 【使用上の注意】

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

## 【μITRON4.0仕様との関係】

μITRON4.0仕様に規定されていない機能である。

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

## 【静的API】

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

## 【C言語API】

```
ER_ID pdqid = acre_pdq(const T_CPDQ *pk_cpdq)
```

## 【パラメータ】

|                          |         |  |
|--------------------------|---------|--|
| ID                       | pdqid   | 生成する優先度データキューのID番号（CRE_PDQの場合）         |
| T_CPDQ *                 | pk_cpdq | 優先度データキューの生成情報を入ったパケットへのポインタ（静的APIを除く） |
| *優先度データキューの生成情報（パケットの内容） |         |  |
| ATR                      | pdqatr  | 優先度データキュー属性                            |
| uint_t                   | pdqcnt  | 優先度データキュー管理領域に格納できるデータ数                |
| PRI                      | maxdpri | 優先度データキューに送信できるデータ優先度の最大値              |
| void *                   | pdqmb   | 優先度データキュー管理領域の先頭番地                     |

#### 【リターンパラメータ】

|       |       |                                   |
|-------|-------|-----------------------------------|
| ER_ID | pdqid | 生成された優先度データキューのID番号（正の値）またはエラーコード |
|-------|-------|-----------------------------------|

#### 【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し [s] 【NGKI1802】<br>・CPUロック状態からの呼出し [s] 【NGKI1803】  |
| E_RSATR | 予約属性<br>・pdqatrが無効 【NGKI1804】<br>・属する保護ドメインの指定が有効範囲外 [sP] 【NGKI1805】<br>・属するクラスの指定が有効範囲外 [sM] 【NGKI1806】<br>・クラスの団みの中に記述されていない [SM] 【NGKI1807】 |
| E_NOSPT | 未サポート機能<br>・条件については各カーネルにおける規定の項を参照  |
| E_PAR   | パラメータエラー<br>・pdqcntが負の値 [S] 【NGKI3289】<br>・maxdpriがTMIN_DPRIより小さい, またはTMAX_DPRIより大きい 【NGKI1819】<br>・その他の条件については機能の項を参照                         |
| E_OACV  | オブジェクトアクセス違反<br>・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】   |
| E_MACV  | メモリアクセス違反<br>・pk_cpdqが指すメモリ領域への読み出しが許可されていない [sP] 【NGKI1809】   |
| E_NOID  | ID番号不足<br>・割り付けられる優先度データキューIDがない [sD] 【NGKI1810】   |
| E_NOMEM | メモリ不足<br>・優先度データキュー管理領域が確保できない 【NGKI1811】  |
| E_OBJ   | オブジェクト状態エラー<br>・pdqidで指定した優先度データキューが登録済み (CRE_PDQ の場合) 【NGKI1812】<br>・その他の条件については機能の項を参照   |

## 【機能】

各パラメータで指定した優先度データキュー生成情報に従って、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】。

静的APIにおいては、pdqidはオブジェクト識別名、pdqatr, pdqcnt, maxdpriは整数定数式パラメータ、pdqmbは一般定数式パラメータである【NGKI1815】。コンフィギュレータは、静的APIのメモリ不足（E\_NOMEM）エラーを検出することができる【NGKI1816】。

pdqmbをNULLとした場合、pdqcntで指定した数のデータを格納できる優先度データキュー管理領域が、コンフィギュレータまたはカーネルにより確保される【NGKI1817】。

〔pdqmbにNULL以外を指定した場合〕

pdqmbにNULL以外を指定した場合、pdqmbを先頭番地とする優先度データキュー管理領域は、アプリケーションで確保しておく必要がある【NGKI1820】。優先度データキュー管理領域をアプリケーションで確保するために、次のマクロを用意している【NGKI1821】。

|                    |  |
|--------------------|--|
| TSZ_PDQMB(pdqcnt)  | pdqcntで指定した数のデータを格納できる優先度データキュー管理領域のサイズ（バイト数）              |
| TCNT_PDQMB(pdqcnt) | pdqcntで指定した数のデータを格納できる優先度データキュー管理領域を確保するために必要なMB_T型の配列の要素数 |

これらを用いて優先度データキュー管理領域を確保する方法は次の通り【NGKI1822】。

MB\_T< 優先度データキュー管理領域の変数名>[TCNT\_PDQMB(pdqcnt)];

この時、pdqmbには<優先度データキュー管理領域の変数名>を指定する【NGKI1823】。

この方法に従わず、pdqmbにターゲット定義の制約に合致しない先頭番地を指定E\_PARエラーとなる【NGKI1824】。また、保護機能対応カーネルにpdqmbで指定した優先度データキュー管理領域がカーネル専用のメモリオブジェクトに含まれない場合、E\_OBJエラーとなる【NGKI1825】。

した時には、  
いて、

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、CRE\_PDQのみをサポートする【ASPS0140】。また、pdqmbにはNULLのみを指定することができる。NULL以外を指定した場合には、E\_NOSPTエラーとなる【ASPS0142】。ただし、動的生成機能拡張パッケージでは、acre\_pdqもサポートする【ASPS0143】。acre\_pdqに対しては、pdqmbにNULL以外を指定できないという制限はない【ASPS0144】。

#### 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、CRE\_PDQのみをサポートする【FMPS0125】。また、pdqmbにはNULLのみを指定することができる。NULL以外を指定した場合には、E\_NOSPTエラーとなる【FMPS0127】。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、CRE\_PDQのみをサポートする【HRPS0125】。また、pdqmbにはNULLのみを指定することができる。NULL以外を指定した場合には、E\_NOSPTエラーとなる【HRPS0127】。ただし、動的生成機能拡張パッケージでは、acre\_pdqもサポートする【HRPS0190】。acre\_pdqに対しては、pdqmbにNULL以外を指定できないという制限はない【HRPS0191】。

|         |                                      |
|---------|--------------------------------------|
| AID_PDQ | 割付け可能な優先度データキューIDの数の指定〔SD〕【NGKI1826】 |
|---------|--------------------------------------|

#### 【静的API】

```
AID_PDQ(uint_t noppdq)
```

#### 【パラメータ】

|        |        |                     |
|--------|--------|---------------------|
| uint_t | noppdq | 割付け可能な優先度データキューIDの数 |
|--------|--------|---------------------|

#### 【エラーコード】

|         |          |  |
|---------|----------|--|
| E_RSATR | 予約属性     | ・保護ドメインの囲みの中に記述されている [P] 【NGKI3432】<br>・クラスの囲みの中に記述されていない [M] 【NGKI1827】 |
| E_PAR   | パラメータエラー | ・noppdqが負の値 【NGKI3280】   |

#### 【機能】

noppdqで指定した数の優先度データキューIDを、優先度データキューを生成するサービスコールによって割付け可能な優先度データキューIDとして確保する 【NGKI1828】。

noppdqは整数定数式パラメータである 【NGKI1829】。

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルの動的生成機能拡張パッケージでは、AID\_PDQをサポートする 【ASPS0214】。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルの動的生成機能拡張パッケージでは、AID\_PDQをサポートする 【HRPS0215】。

|         |   |
|---------|---|
| SAC_PDQ | 優先度データキューのアクセス許可ベクタの設定 [SP] 【NGKI1830】  |
| sac_pdq | 優先度データキューのアクセス許可ベクタの設定 [TPD] 【NGKI1831】 |

#### 【静的API】

```
SAC_PDQ(ID pdqid, { ACPTN acptn1, ACPTN acptn2,  
                      ACPTN acptn3, ACPTN acptn4 })
```

#### 【C言語API】

```
ER ercd = sac_pdq(ID pdqid, const ACVCT *p_acvct)
```

#### 【パラメータ】

|         |         |                                   |
|---------|---------|-----------------------------------|
| ID      | pdqid   | 対象優先度データキューのID番号                  |
| ACVCT * | p_acvct | アクセス許可ベクタを入れたパケットへのポインタ（静的APIを除く） |

\* アクセス許可ベクタ（パケットの内容）

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

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー  |
|         | ・非タスクコンテキストからの呼び出し [s] 【NGKI1832】  |
|         | ・CPUロック状態からの呼び出し [s] 【NGKI1833】  |
| E_ID    | 不正ID番号   |
|         | ・pdqidが有効範囲外 [s] 【NGKI1834】  |
| E_RSATR | 予約属性   |
|         | ・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<br>(対象優先度データキューが無所属の場合は、保護ドメインの囲みの外)に記述されていない [S] 【NGKI1835】 |
|         | ・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<br>されていない [SM] 【NGKI1836】                                    |
| 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 【NGKI1837】  |
| 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<br>ない [s] 【NGKI1838】   |
| E_MACV  | メモリアクセス違反  |
|         | ・p_acvctが指すメモリ領域への読み出しアクセスが許可されて<br>いない [s] 【NGKI1839】                                   |
| 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 [s] 【NGKI1840】  |
|         | ・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<br>定済み [S] 【NGKI1841】  |

#### 【機能】

pdqidで指定した優先度データキュー（対象優先度データキュー）のアクセス許可ベクタ（4つのアクセス許可パターンの組）を、各パラメータで指定した値に設定する【NGKI1842】。

静的APIにおいては、pdqidはオブジェクト識別名、acptn1～acptn4は整数定数式パラメータである【NGKI1843】。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、SAC\_PDQのみをサポートする【HRPS0128】。ただし、動的生成機能拡張パッケージでは、sac\_pdqもサポートする【HRPS0192】。

del\_pdq 優先度データキューの削除 [TD] 【NGKI1844】

#### 【C言語API】

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

#### 【パラメータ】

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

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー                                    |
|         | ・非タスクコンテキストからの呼び出し 【NGKI1845】                |
|         | ・CPUロック状態からの呼び出し 【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】。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、del\_pdqをサポートしない【HRPS0129】。ただし、動的生成機能拡張パッケージでは、del\_pdqをサポートする【HRPS0193】。

|           |                                       |
|-----------|---------------------------------------|
| snd_pdq   | 優先度データキューへの送信 [T] 【NGKI1855】          |
| psnd_pdq  | 優先度データキューへの送信（ポーリング）[T] 【NGKI1856】    |
| ipsnd_pdq | 優先度データキューへの送信（ポーリング）[I] 【NGKI1857】    |
| tsnd_pdq  | 優先度データキューへの送信（タイムアウト付き）[T] 【NGKI1858】 |

#### 【C言語API】

```
ER ercd = snd_pdq(ID pdqid, intptr_t data, PRI datapri)
ER ercd = psnd_pdq(ID pdqid, intptr_t data, PRI datapri)
ER ercd = ipsnd_pdq(ID pdqid, intptr_t data, PRI datapri)
ER ercd = tsnd_pdq(ID pdqid, intptr_t data, PRI datapri, TMO tmout)
```

#### 【パラメータ】

|          |         |                        |
|----------|---------|------------------------|
| ID       | pdqid   | 対象優先度データキューのID番号       |
| intptr_t | data    | 送信データ                  |
| PRI      | datapri | 送信データの優先度              |
| TMO      | tmout   | タイムアウト時間 (tsnd_pdqの場合) |

### 【リターンパラメータ】

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

### 【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し (ipsnd_pdqを除く)<br>【NGKI1859】                  |
|         | ・タスクコンテキストからの呼出し (ipsnd_pdqの場合) 【NGKI1860】                                   |
|         | ・CPUロック状態からの呼出し 【NGKI1861】   |
|         | ・ディスパッチ保留状態からの呼出し (snd_pdqとtsnd_pdqの場合) 【NGKI1862】                           |
| E_NOSPT | 未サポート機能<br>・制約タスクからの呼出し (snd_pdqとtsnd_pdqの場合) 【NGKI1863】                     |
| E_ID    | 不正ID番号<br>・pdqidが有効範囲外 【NGKI1864】  |
| E_PAR   | パラメータエラー<br>・tmoutが無効 (tsnd_pdqの場合) 【NGKI1865】<br>・その他の条件については機能の項を参照        |
| E_NOEXS | オブジェクト未登録<br>・対象優先度データキューが未登録 [D] 【NGKI1866】                                 |
| E_OACV  | オブジェクトアクセス違反<br>・対象優先度データキューに対する通常操作1が許可されていない (ipsnd_pdqを除く) [P] 【NGKI1867】 |
| E_TMOUT | ポーリング失敗またはタイムアウト (snd_pdqを除く) 【NGKI1868】                                     |
| E_RLWAI | 待ち禁止状態または待ち状態の強制解除 (snd_pdqとtsnd_pdqの場合) 【NGKI1869】                          |
| 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の場合) 【NGKI1870】                          |

### 【機能】

pdqidで指定した優先度データキュー（対象優先度データキュー）に、dataで指  
定したデータを、

定したデータを、

dataで指定したデータを受信し、待ち解除される  
【NGKI1871】。待ち解除されたタスクには、待ち状態となったサービスコール  
【NGKI1872】。

からE\_OKが返る【

対象優先度データキューの受信待ち行列にタスクが存在せず、優先度データキュー管理領域にデータを格納するスペースがある場合には、dataで指定したデータが、datapriで指定したデータの優先度順で優先度データキュー管理領域に格納される【NGKI1873】。

対象優先度データキューの受信待ち行列にタスクが存在せず、優先度データキュー管理領域にデータを格納するスペースがない場合には、自タスクは優先度データキューへの送信待ち状態となり、対象優先度データキューの送信待ち行列につながれる【NGKI1874】。

datapriは、TMIN\_DPRI以上で、対象データキューに送信できるデータ優先度の最大値以下でなければならない。そうでない場合には、E\_PARエラーとなる【NGKI1876】。

|          |   |
|----------|---|
| recv_pdq | 優先度データキューからの受信 [T] 【NGKI1877】           |
| prev_pdq | 優先度データキューからの受信（ポーリング） [T] 【NGKI1878】    |
| trcv_pdq | 優先度データキューからの受信（タイムアウト付き） [T] 【NGKI1879】 |

#### 【C言語API】

```
ER ercd = recv_pdq(ID pdqid, intptr_t *p_data, PRI *p_datapri)
ER ercd = prev_pdq(ID pdqid, intptr_t *p_data, PRI *p_datapri)
ER ercd = trcv_pdq(ID pdqid, intptr_t *p_data, PRI *p_datapri, TMO tmout)
```

#### 【パラメータ】

|            |           |                          |
|------------|-----------|--------------------------|
| ID         | pdqid     | 対象優先度データキューのID番号         |
| intptr_t * | p_data    | 受信データを入れるメモリ領域へのポインタ     |
| PRI *      | p_datapri | 受信データの優先度を入れるメモリ領域へのポインタ |
| TMO        | tmout     | タイムアウト時間 (trcv_pdqの場合)   |

#### 【リターンパラメータ】

|          |         |                       |
|----------|---------|-----------------------|
| ER       | ercd    | 正常終了 (E_OK) またはエラーコード |
| intptr_t | data    | 受信データ                 |
| PRI      | datapri | 受信データの優先度             |

#### 【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI1880】<br>・CPUロック状態からの呼出し【NGKI1881】<br>・ディスパッチ保留状態からの呼出し（recv_pdqを除く）【NGKI1882】  |
| E_NOSPT | 未サポート機能<br>・制約タスクからの呼出し（recv_pdqを除く）【NGKI1883】   |
| E_ID    | 不正ID番号<br>・pdqidが有効範囲外【NGKI1884】   |
| E_PAR   | パラメータエラー<br>・tmoutが無効（recv_pdqの場合）【NGKI1885】   |
| E_NOEXS | オブジェクト未登録<br>・対象優先度データキューが未登録〔D〕【NGKI1886】   |
| E_OACV  | オブジェクトアクセス違反<br>・対象優先度データキューに対する通常操作2が許可されていない〔P〕【NGKI1887】  |
| E_MACV  | メモリアクセス違反<br>・p_dataが指すメモリ領域への書き込みアクセスが許可されていない〔P〕【NGKI1888】<br>・p_datapriが指すメモリ領域への書き込みアクセスが許可されていない〔P〕【NGKI1889】 |
| E_TMOUT | ポーリング失敗またはタイムアウト（recv_pdqを除く）【NGKI1890】  |
| E_RLWAI | 待ち禁止状態または待ち状態の強制解除（recv_pdqを除く）【NGKI1891】  |
| E_DLT   | 待ちオブジェクトの削除または再初期化（recv_pdqを除く）【NGKI1892】  |

## 【機能】

pdqidで指定した優先度データキュー（対象優先度データキュー）からデータを受信する。データの受信に成功した場合、受信したデータはp\_dataが指すメモリ領域に、その優先度はp\_datapriが指すメモリ領域に返される【NGKI1894】。具体的な振舞いは以下の通り。

対象優先度データキューの優先度データキュー管理領域にデータが格納されている場合には、優先度データキュー管理領域の先頭に格納されたデータを受信する【NGKI1893】。また、送信待ち行列にタスクが存在する場合には、送信待ち行列の先頭のタスクの送信データが、データの優先度順で優先度データキュー管理領域に格納され、そのタスクは待ち解除される【NGKI1895】。待ち解除されたタスクには、待ち状態となったサービスコールからE\_OKが返る【NGKI1896】。

対象優先度データキューの優先度データキュー管理領域にデータが格納されておらず、送信待ち行列にタスクが存在する場合には、送信待ち行列の先頭のタスクの送信データを受信する【NGKI1897】。送信待ち行列の先頭のタスクは、待ち解除される【NGKI1899】。待ち解除されたタスクには、待ち状態となったサービスコールからE\_OKが返る【NGKI1900】。

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

ini\_pdq 優先度データキューの再初期化 [T] 【NGKI1902】

#### 【C言語API】

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

#### 【パラメータ】

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

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼び出し【NGKI1903】<br>・CPUロック状態からの呼び出し【NGKI1904】 |
| E_ID    | 不正ID番号<br>・pdqidが有効範囲外【NGKI1905】  |
| E_NOEXS | オブジェクト未登録<br>・対象優先度データキューが未登録 [D] 【NGKI1906】                            |
| E_OACV  | オブジェクトアクセス違反<br>・対象優先度データキューに対する管理操作が許可されていない [P] 【NGKI1907】            |

#### 【機能】

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

対象優先度データキューの優先度データキュー管理領域は、格納されているデータがない状態に初期化される【NGKI1908】。また、対象優先度データキューの送信待ち行列と受信待ち行列につながれたタスクは、それぞれの待ち行列の先頭のタスクから順に待ち解除される【NGKI1909】。待ち解除されたタスクには、待ち状態となったサービスコールからE\_DLTエラーが返る【NGKI1910】。

#### 【補足説明】

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

#### 【使用上の注意】

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

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

ref\_pdq 優先度データキューの状態参照 [T] 【NGKI1911】

#### 【C言語API】

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

#### 【パラメータ】

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

#### 【リターンパラメータ】

ER ercd 正常終了 (E\_OK) またはエラーコード

\*優先度データキューの現在状態 (パケットの内容)

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

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI1912】<br>・CPUロック状態からの呼出し【NGKI1913】 |
| E_ID    | 不正ID番号<br>・pdqidが有効範囲外【NGKI1914】                                      |
| E_NOEXS | オブジェクト未登録<br>・対象優先度データキューが未登録〔D〕【NGKI1915】                            |
| E_OACV  | オブジェクトアクセス違反<br>・対象優先度データキューに対する参照操作が許可されていない〔P〕【NGKI1916】            |
| E_MACV  | メモリアクセス違反<br>・pk_rpdqが指すメモリ領域への書き込みアクセスが許可されていない〔P〕【NGKI1917】         |

### 【機能】

pdqidで指定した優先度データキュー（対象優先度データキュー）の現在状態を参照する。参照した現在状態は、pk\_rpdqで指定したパケットに返される【NGKI1918】。

対象優先度データキューの送信待ち行列にタスクが存在しない場合、stskidにはTSK\_NONE (=0) が返る【NGKI1919】。また、受信待ち行列にタスクが存在しない場合、rtskidにはTSK\_NONE (=0) が返る【NGKI1920】。

### 【使用上の注意】

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

## 1.4.5. メールボックス

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。メールボックスは、メールボックスIDと呼ばれるID番号によって識別する【NGKI1921】。

各メールボックスが持つ情報は次の通り【NGKI1922】。

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

メッセージキューは、メールボックスに送信されたメッセージを、FIFO順または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。

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（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順序でつながれているキューである。

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】。

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

TA\_TPRIを指定しない場合、待ち行列はFIFO順になる【NGKI1924】。TA\_MPRIを指定しない場合、メッセージキューはFIFO順になる【NGKI1925】。

優先度別のメッセージキューへッダ領域は、TA\_MPRI属性のメールボックスに対して、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。

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】。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が指定されているか否かにより、以下のいずれかを用いる【NGKI1928】。

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

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送信した場合の動作は保証されない【NGKI1930】。

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フィールドに設定する【NGKI1931】。

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】。

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。

|           |                   |            |
|-----------|-------------------|------------|
| TMIN_MPRI | メッセージ優先度の最小値 (=1) | 【NGKI1933】 |
| TMAX_MPRI | メッセージ優先度の最大値      |            |

|            |  |
|------------|--|
| 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）【NGKI1934】 |
|------------|--|

#### 【補足説明】

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へッダ領域は用いていない。

#### 【使用上の注意】

メールボックス機能は、μITRON4.0仕様との互換性のために残した機能であり、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。メールボックス機能は、ほとんどの場合に、データキュー機能または優先度データキュー機能を用いて、メッセージを置いたメモリ領域へのポインタを送受信する方法で置き換えることができる。

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、メールボックス機能をサポートする【ASPS0147】。メッセージ優先度の最大値（TMAX\_MPRI）は16に固定されている【ASPS0148】。ただし、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】。

#### 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、メールボックス機能をサポートする【FMPS0130】。メッセージ優先度の最大値（TMAX\_MPRI）は16に固定されている【FMPS0131】。

#### 【TOPPERS/HRP2カーネルにおける規定】

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】。

#### 【μITRON4.0仕様との関係】

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

|          |                             |
|----------|-----------------------------|
| CRE_MBX  | メールボックスの生成 [Sp] 【NGKI1935】  |
| acre_mbx | メールボックスの生成 [TpD] 【NGKI1936】 |

#### 【静的API】

```
CRE_MBX(ID mbxid, { ATR mbxatr, PRI maxmpri, void *mprihd })
```

#### 【C言語API】

```
ER_ID mbxid = acre_mbxx(const T_CMBX *pk_cmbx)
```

#### 【パラメータ】

|          |         |                                       |
|----------|---------|---------------------------------------|
| ID       | mbxid   | 生成するメールボックスのID番号 (CRE_MBXXの場合)        |
| T_CMBX * | pk_cmbx | メールボックスの生成情報を入ったパケットへのポインタ (静的APIを除く) |

\* メールボックスの生成情報 (パケットの内容)

|        |         |                                  |
|--------|---------|----------------------------------|
| ATR    | mbxatr  | メールボックス属性                        |
| PRI    | maxmpri | 優先度メールボックスに送信できるメッセージ<br>優先度の最大値 |
| void * | mprihd  | 優先度別のメッセージキューへッダ領域の先頭<br>番地      |

#### 【リターンパラメータ】

|       |       |                                      |
|-------|-------|--------------------------------------|
| ER_ID | mbxid | 生成されたメールボックスのID番号 (正の値)<br>またはエラーコード |
|-------|-------|--------------------------------------|

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼び出し [s] 【NGKI1937】<br>・CPUロック状態からの呼び出し [s] 【NGKI1938】                       |
| E_RSATR | 予約属性<br>・mbxatrが無効 【NGKI1939】<br>・属するクラスの指定が有効範囲外 [sM] 【NGKI1940】<br>・クラスの囲みの中に記述されていない [SM] 【NGKI1941】 |
| E_NOSPT | 未サポート機能<br>・条件については各カーネルにおける規定の項を参照   |
| E_PAR   | パラメータエラー<br>・maxmpriがTMIN_MPRIより小さい、またはTMAX_MPRIより大きい 【NGKI1951】  |
| E_NOID  | ID番号不足<br>・割り付けられるメールボックスIDがない [sD] 【NGKI1942】  |
| E_NOMEM | メモリ不足<br>・優先度別のメッセージキューへッダ領域が確保できない<br>【NGKI1943】   |
| E_OBJ   | オブジェクト状態エラー<br>・mbxidで指定したメールボックスが登録済み (CRE_MBXXの場合) 【NGKI1944】   |

## 【機能】

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 maxmpriから優先度別のメッセージキューへッダ領域が設定され  
され、mprihdと  
NGKI1945】。また、待ち行列は空の状態に初期化される【NGKI1946】。

静的APIにおいては、mbxidはオブジェクト識別名、mbxatrとmaxmpriは整数定数 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【NGKI1948】。

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ジキューへッダ領域が、コンフィギュレータまたはカーネルにより確保される【NGKI1949】。

## 【未決定事項】

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ない。

## 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、CRE\_MBXのみをサポートする【ASPS0150】。また、優先度別のメッセージキューへッダ領域は使用しておらず、mprihdにはNULLのみを指定 することができる。  
NULL以外を指定した場合には、E\_NOSPTエラーとなる  
【  
ASPS0152】。ただし、動的生成機能拡張パッケージでは、acre\_mbxもサポートする【ASPS0153】。acre\_mbxに対しても、mprihdにはNULLのみを指定することができる【  
ASPS0154】。優先度別のメッセージキューへッダ領域を使用しないため、 E\_NOMEM が返ることはない【ASPS0155】。

## 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、CRE\_MBXのみをサポートする【FMPS0132】。また、優先度別のメッセージキューへッダ領域は使用しておらず、mprihdにはNULLのみを指定 することができる。  
NULL以外を指定した場合には、E\_NOSPTエラーとなる  
【  
FMPS0134】。優先度別のメッセージキューへッダ領域を使用しないため、 E\_NOMEM が返ることはない【FMPS0135】。

AID\_MBX 割付け可能なメールボックスIDの数の指定 [SpD] 【NGKI1952】

## 【静的API】

AID\_MBX(uint\_t nombx)

## 【パラメータ】

|                     |                    |                   |
|---------------------|--------------------|-------------------|
| <code>uint_t</code> | <code>nombx</code> | 割付け可能なメールボックスIDの数 |
|---------------------|--------------------|-------------------|

#### 【エラーコード】

|                      |  |
|----------------------|--|
| <code>E_RSATR</code> | 予約属性<br>・クラスの囲みの中に記述されていない [M] 【NGKI1953】        |
| <code>E_PAR</code>   | パラメータエラー<br>・ <code>nombx</code> が負の値 【NGKI3281】 |

#### 【機能】

`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】。

`nombx`は整数定数式パラメータである【NGKI1955】。

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルの動的生成機能拡張パッケージでは、`AID_MBX`をサポートする【ASPS0215】。

|                       |                  |            |
|-----------------------|------------------|------------|
| <code>del_mbxx</code> | メールボックスの削除 [TpD] | 【NGKI1956】 |
|-----------------------|------------------|------------|

#### 【C言語API】

|   |
|---|
| <code>ER ercd = del_mbxx(ID mbxid)</code> |
|---|

#### 【パラメータ】

|                 |                    |                |
|-----------------|--------------------|----------------|
| <code>ID</code> | <code>mbxid</code> | 対象メールボックスのID番号 |
|-----------------|--------------------|----------------|

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI1957】<br>・CPUロック状態からの呼出し【NGKI1958】 |
| E_ID    | 不正ID番号<br>・mbxidが有効範囲外【NGKI1959】                                      |
| E_NOEXS | オブジェクト未登録<br>・対象メールボックスが未登録【NGKI1960】                                 |
| E_OBJ   | オブジェクト状態エラー<br>・対象メールボックスは静的APIで生成された【NGKI1961】                       |

### 【機能】

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。具体的な振舞いは以下の通り。

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【NGKI1964】。

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へッダ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【NGKI1965】。

### 【使用上の注意】

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\_mbxをサポートしない【ASPS0156】。ただし、動的生成機能拡張パッケージでは、del\_mbxをサポートする【ASPS0157】。

### 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、del\_mbxをサポートしない【FMP0136】。

snd\_mbx メールボックスへの送信 [Tp] 【NGKI1966】

### 【C言語API】

```
ER ercd = snd_mbx(ID mbxid, T_MSG *pk_msg)
```

### 【パラメータ】

|       |         |                |
|-------|---------|----------------|
| ID    | mbxid   | 対象メールボックスのID番号 |
| T_MSG | *pk_msg | 送信メッセージの先頭番地   |

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し 【NGKI1967】<br>・CPUロック状態からの呼出し 【NGKI1968】 |
| E_ID    | 不正ID番号<br>・mbxidが有効範囲外 【NGKI1969】                                       |
| E_PAR   | パラメータエラー<br>・条件については機能の項を参照   |
| E_NOEXS | オブジェクト未登録<br>・対象メールボックスが未登録 [D] 【NGKI1970】                              |

#### 【機能】

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で指定したメッセージを送信する。具体的な振舞いは以下の通り。

対象メ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  
pk\_msgで指定したメッセージを受信し、待ち解除される  
【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  
【NGKI1972】。

のタスクが、

からE\_OKが返る【

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  
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  
る順序で、メッセージキューにつながれる【NGKI1973】。

対象メールボックスがTA\_MPRI属性である場合には、pk\_msgで指定したメッセー  
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、  
対象メールボックスに送信できるメッセージ優先度の最大値以下でなければな  
らない。そうでない場合には、E\_PARエラーとなる【NGKI1975】。

|          |  |
|----------|--|
| recv_mbx | メールボックスからの受信 [Tp] 【NGKI1976】           |
| prcv_mbx | メールボックスからの受信（ポーリング） [Tp] 【NGKI1977】    |
| trcv_mbx | メールボックスからの受信（タイムアウト付き） [Tp] 【NGKI1978】 |

#### 【C言語API】

```

ER ercd = rcv_mbx(ID mbxid, T_MSG **ppk_msg)
ER ercd = prcv_mbx(ID mbxid, T_MSG **ppk_msg)
ER ercd = trcv_mbx(ID mbxid, T_MSG **ppk_msg, TM0 tmout)

```

### 【パラメータ】

|          |         |                             |
|----------|---------|-----------------------------|
| ID       | mbxid   | 対象メールボックスのID番号              |
| 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へのポインタ |
| TM0      | tmout   | タイムアウト時間 (trcv_mbxの場合)      |

### 【リターンパラメータ】

|         |         |                       |
|---------|---------|-----------------------|
| ER      | ercd    | 正常終了 (E_OK) またはエラーコード |
| T_MSG * | ppk_msg | 受信メッセージの先頭番地          |

### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー <ul style="list-style-type: none"> <li>・非タスクコンテキストからの呼び出し 【NGKI1979】</li> <li>・CPUロック状態からの呼び出し 【NGKI1980】</li> <li>・ディスパッチ保留状態からの呼び出し (prcv_mbxを除く) 【NGKI1981】</li> </ul> |
| E_NOSPT | 未サポート機能 <ul style="list-style-type: none"> <li>・制約タスクからの呼び出し (prcv_mbxを除く) 【NGKI1982】</li> </ul>  |
| E_ID    | 不正ID番号 <ul style="list-style-type: none"> <li>・mbxidが有効範囲外 【NGKI1983】</li> </ul>  |
| E_PAR   | パラメータエラー <ul style="list-style-type: none"> <li>・tmoutが無効 (trcv_mbxの場合) 【NGKI1984】</li> </ul>   |
| E_NOEXS | オブジェクト未登録 <ul style="list-style-type: none"> <li>・対象メールボックスが未登録 [D] 【NGKI1985】</li> </ul>   |
| E_TMOUT | ポーリング失敗またはタイムアウト (rcv_mbxを除く) 【NGKI1986】  |
| 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) 【NGKI1987】   |
| 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) 【NGKI1988】   |

### 【機能】

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れる。具体的な振舞いは以下の通り。

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

指定したメモリ領域に返される【NGKI1989】.

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っていない場合には、自タスクはメールボックスからの受信待ち状態となり、対象メールボックスの待ち行列につながれる【NGKI1990】.

ini\_mbx メールボックスの再初期化 [Tp] 【NGKI1991】

#### 【C言語API】

```
ER ercd = ini_mbx(ID mbxid)
```

#### 【パラメータ】

|    |       |                |
|----|-------|----------------|
| ID | mbxid | 対象メールボックスのID番号 |
|----|-------|----------------|

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼び出し 【NGKI1992】<br>・CPUロック状態からの呼び出し 【NGKI1993】 |
| E_ID    | 不正ID番号<br>・mbxidが有効範囲外 【NGKI1994】   |
| E_NOEXS | オブジェクト未登録<br>・対象メールボックスが未登録 [D] 【NGKI1995】                                |

#### 【機能】

mbxidで指定したメールボックス（対象メールボックス）を再初期化する。具体的な振舞いは以下の通り。

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れているメッセージがない状態に初期化される【NGKI1996】。また、対象メール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、待ち状態となつたサービスコールからE\_DLTエラーが返る【NGKI1998】。

#### 【使用上の注意】

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

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。

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、アプリケーションの責任である。

#### 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである。

`ref_mbx` メールボックスの状態参照 [Tp] 【NGKI1999】

#### 【C言語API】

```
ER ercd = ref_mbx(ID mbxid, T_RMBX *pk_rmbx)
```

#### 【パラメータ】

|          |         |                            |
|----------|---------|----------------------------|
| ID       | mbxid   | 対象メールボックスのID番号             |
| 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 |

#### 【リターンパラメータ】

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

\* メールボックスの現在状態 (パケットの内容)

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

#### 【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び出し 【NGKI2000】 |
|         | ・CPUロック状態からの呼び出し 【NGKI2001】   |
| E_ID    | 不正ID番号                        |
|         | ・mbxidが有効範囲外 【NGKI2002】       |
| E_NOEXS | オブジェクト未登録                     |
|         | ・対象メールボックスが未登録 [D] 【NGKI2003】 |

## 【機能】

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。  
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】。

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 TSK\_NONE (=0) が返る【NGKI2005】。また、メッセージキューにメッセージがつながっていない場合、pk\_msg にはNULLが返る【NGKI2006】。

## 【使用上の注意】

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。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の状態が変化している可能性があるためである。

### 1.4.6. ミューテックス

ミューテックスは、タスク間の排他制御を行うための同期・通信オブジェクトである。タスクは、排他制御区間に入る時にミューテックスをロックし、排他制御区間を出る時にロック解除する。ミューテックスは、ミューテックスIDと ID番号によって識別する【NGKI2007】。

ミューテックスは、排他制御に伴う優先度逆転の時間を最小限に抑えるための優先度上限プロトコル（priority ceiling protocol）をサポートする。ミューテックス属性により優先度上限ミューテックスであると指定することで、そのミューテックスの操作時に、優先度上限プロトコルに従った現在優先度の制御が行われる。

各ミューテックスが持つ情報は次の通り【NGKI2008】。

- ミューテックス属性
- ロック状態（ロックされている状態とロック解除されている状態）
- ミューテックスをロックしているタスク
- 待ち行列（ミューテックスのロック待ち状態のタスクのキュー）
- 上限優先度（優先度上限ミューテックスの場合）
- アクセス許可ベクタ（保護機能対応カーネルの場合）
- 属する保護ドメイン（保護機能対応カーネルの場合）
- 属するクラス（マルチプロセッサ対応カーネルの場合）

待ち行列は、ミューテックスをロックできるまで待っている状態（ミューテックスのロック待ち状態）のタスクが、ミューテックスをロックできる順序でつながれているキューである。

上限優先度は、優先度上限ミューテックスに対してのみ有効で、ミューテックスの生成時に、そのミューテックスをロックする可能性のあるタスクのベース優先度の中で最も高い優先度（または、それより高い優先度）に設定する【NGKI2009】。

ミューテックス属性には、次の属性を指定することができる【NGKI2010】。

|            |       |                                  |
|------------|-------|----------------------------------|
| TA_TPRI    | 0x01U | 待ち行列をタスクの優先度順にする                 |
| TA_CEILING | 0x03U | 優先度上限ミューテックスとする。待ち行列をタスクの優先度順にする |

TA\_TPRI, TA\_CEILINGのいずれも指定しない場合、待ち行列はFIFO順になる【NGKI2011】。

ミューテックス機能に関連して、各タスクが持つ情報は次の通り【NGKI2012】。

- ロックしているミューテックスのリスト

ロックしているミューテックスのリストは、タスクの起動時に空に初期化される【NGKI2013】。

タスクの現在優先度は、そのタスクのベース優先度と、そのタスクがロックしている優先度上限ミューテックスの優先度上限の中で、最も高い優先度に設定される【NGKI2014】。

ミューテックス機能によりタスクの現在優先度が変化する場合には、次の処理が行われる。現在優先度を変化させるサービスコールの前後とも、当該タスクが実行できる状態である場合には、同じ優先度のタスクの中で最高優先順位となる【NGKI2015】。そのサービスコールにより、当該タスクが実行できる状態に遷移する場合には、同じ優先度のタスクの中で最低優先順位となる【NGKI2016】。そのサービスコールの後で、当該タスクが待ち状態で、タスクの優先度順の待ち行列につながれている場合には、当該タスクの変更後の現在優先度に従って、その待ち行列中の順序が変更される【NGKI2017】。待ち行列中に同じ現在優先度のタスクがある場合には、当該タスクの順序はそれらの中で最後になる【NGKI2018】。

ミューテックス機能に関連して、タスクの終了時に行うべき処理として、タスクがロックしているミューテックスのロック解除がある。タスクの終了時にロックしているミューテックスが残っている場合、それらのミューテックスは、ロックしたのと逆の順序でロック解除される【NGKI2019】。

ミューテックス機能に関連するカーネル構成マクロは次の通り。

|            |  |
|------------|--|
| TNUM_MTXID | 登録できるミューテックスの数（動的生成対応でないカーネルでは、静的APIによって登録されたミューテックスの数に一致）【NGKI2020】 |
|------------|--|

#### 【使用上の注意】

優先度上限プロトコルには、(a) 優先度の低いタスクの排他制御区間に最大1回しかロックされない、(b) タスクの実行が開始された以降は優先度の低いタスクにロックされないという利点があるが、これは、タスク間の同期に優先度上限ミューテックスのみを用い、他の方法でタスクのスケジューリングに関与しない場合に得られる利点である。

これらの利点を得るために、タスクの優先順位の回転やディスパッチの禁止

を行ってはならないことに加えて、優先度上限ミューテックスをロックしたタスクを待ち状態にしてはならない。特に、優先度上限ミューテックスに対して、タスクがロック待ち状態になる状況に注意が必要である（優先度上限プロトコルでは、タスクがミューテックスのロック待ち状態になることはない）。

例えば、着目するタスクAと、タスクAよりベース優先度の低いタスクBとタスクAよりも高い上限優先度を持った優先度上限ミューテックスがある場合を考える。タスクAがミューテックスをロックし、タスクBとタスクCがミューテックスを待っている状況で、タスクAがミューテックスをロック解除すると、タスクBがミューテックスをロックして優先度が上がり、タスクBに切り換わる。さらにタスクBがミューテックスをロック解除すると、タスクCがミューテックスをロックして優先度が上がり、タスクCに切り換わる。タスクAが実行されるのは、タスクCがミューテックスをロック解除した後である。この例では、タスクAが実行開始後に、タスクBとタスクCの排他制御区間にブロックされることになる。

優先度上限ミューテックスに対してタスクがロック待ち状態になる状況を回避するためには、優先度上限ミューテックスをロックする場合に、待ち状態にならないploc\_mtxを用いるのが安全である。

#### 【補足説明】

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 protocolとは異なるものである。この仕様の方式は、OSEK/VDX OS仕様でもpriority ceiling protocolと呼ばれているが、学術論文や他のOSでは、immediate ceiling priority protocol, priority protection protocol, priority ceiling emulation, highest locker protocolなどと呼ばれている。

#### 【TOPPERS/ASPカーネルにおける規定】

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、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ることができる【ASPS0159】。

#### 【TOPPERS/FMPカーネルにおける規定】

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】。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、ミューテックス機能をサポートする【HRPS0131】。

#### 【未決定事項】

マルチプロセッサにおいては、タスク間の同期に優先度上限ミューテックスのみを用い、他の方法でタスクのスケジューリングに関与しない場合でも、優先度上限ミューテックスに対してタスクがロック待ち状態になる。マルチプロセッ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、今後の課題である。

#### 【μITRON4.0仕様との関係】

μITRON4.0仕様の厳密な優先度制御規則を採用し、簡略化した優先度制御規則

はサポートしていない。また、μITRON4.0仕様でサポートしている優先度継承  
inheritance protocol) は、現時点ではサポートしていない。

プロトコル (priority

ミューテックス機能によりタスクの現在優先度が変化する場合の振舞いは、  
μITRON4.0仕様では実装依存となっているが、この仕様では規定している。

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

|          |                            |
|----------|----------------------------|
| CRE_MTX  | ミューテックスの生成 [S] 【NGKI2021】  |
| acre_mtx | ミューテックスの生成 [TD] 【NGKI2022】 |

#### 【静的API】

```
CRE_MTX(ID mtxid, { ATR mtxatr, PRI ceilpri })
```

#### 【C言語API】

```
ER_ID mtxid = acre_mtx(const T_CMTX *pk_cmtx)
```

#### 【パラメータ】

|          |         |                                       |
|----------|---------|---------------------------------------|
| ID       | mtxid   | 生成するミューテックスのID番号 (CRE_MTXの場合)         |
| T_CMTX * | pk_cmtx | ミューテックスの生成情報を入ったパケットへのポインタ (静的APIを除く) |

\* ミューテックスの生成情報 (パケットの内容)  
ATR            mtxatr      ミューテックス属性  
PRI            ceilpri     ミューテックスの上限優先度

#### 【リターンパラメータ】

|       |       |                                      |
|-------|-------|--------------------------------------|
| ER_ID | mtxid | 生成されたミューテックスのID番号 (正の値)<br>またはエラーコード |
|-------|-------|--------------------------------------|

#### 【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し [s] 【NGKI2023】<br>・CPUロック状態からの呼出し [s] 【NGKI2024】  |
| E_RSATR | 予約属性<br>・mtxatrが無効 【NGKI2025】<br>・属する保護ドメインの指定が有効範囲外 [sP] 【NGKI2026】<br>・属するクラスの指定が有効範囲外 [sM] 【NGKI2027】<br>・クラスの囲みの中に記述されていない [SM] 【NGKI2028】 |
| E_PAR   | パラメータエラー<br>・条件については機能の項を参照  |
| E_OACV  | オブジェクトアクセス違反<br>・システム状態に対する管理操作が許可されていない [sP] 【NGKI2029】   |
| E_MACV  | メモリアクセス違反<br>・pk_cmtxが指すメモリ領域への読み出しが許可されていない [sP] 【NGKI2030】   |
| E_NOID  | ID番号不足<br>・割り付けられるミューテックスIDがない [sD] 【NGKI2031】   |
| E_OBJ   | オブジェクト状態エラー<br>・mtxidで指定したセマフォが登録済み (CRE mtxの場合) 【NGKI2032】  |

### 【機能】

各パラメータで指定したミューテックス生成情報に従って、ミューテックスを生成する。生成されたミューテックスのロック状態はロックされていない状態に、待ち行列は空の状態に初期化される【NGKI2033】。

静的APIにおいては、mtxidはオブジェクト識別名、mtxatrとceilpriは整数定数式パラメータである【NGKI2034】。優先度上限ミューテックス以外の場合には、ceilpriの指定を省略することができる【NGKI2035】。

優先度上限ミューテックスを生成する場合、ceilpriは、TMIN\_TPRI以上、TMAX\_TPRI以下でなければならない。そうでない場合には、E\_PARエラーとなる【NGKI2037】。

### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルのミューテックス機能拡張パッケージでは、CRE mtxのみをサポートする【ASPS0160】。

### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、CRE mtxのみをサポートする【HRPS0132】。ただし、動的生成機能拡張パッケージでは、acre\_mtxもサポートする【HRPS0194】。

|         |                                      |
|---------|--------------------------------------|
| AID_MTX | 割付け可能なミューテックスIDの数の指定 [SD] 【NGKI2038】 |
|---------|--------------------------------------|

### 【静的API】

AID mtx(uint\_t nomtx)

#### 【パラメータ】

uint\_t nomtx 割付け可能なミューテックスIDの数

#### 【エラーコード】

E\_RSATR

予約属性

- ・保護ドメインの囲みの中に記述されている [P] 【NGKI3433】
- ・クラスの囲みの中に記述されていない [M] 【NGKI2039】

E\_PAR

パラメータエラー

- ・nomtxが負の値 【NGKI3282】

#### 【機能】

nomtxで指定した数のミューテックスIDを、ミューテックスを生成するサービスコールによって割付け可能なミューテックスIDとして確保する【NGKI2040】。

nomtxは整数定数式パラメータである【NGKI2041】。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルの動的生成機能拡張パッケージでは、AID mtxをサポートする【HRPS0216】。

SAC mtx ミューテックスのアクセス許可ベクタの設定 [SP] 【NGKI2042】

sac\_mtx ミューテックスのアクセス許可ベクタの設定 [TPD] 【NGKI2043】

#### 【静的API】

```
SAC mtx(ID mtxid, { ACPTN acptn1, ACPTN acptn2,
                      ACPTN acptn3, ACPTN acptn4 })
```

#### 【C言語API】

```
ER ercd = sac_mtx(ID mtxid, const ACVCT *p_acvct)
```

#### 【パラメータ】

|         |         |                                   |
|---------|---------|-----------------------------------|
| ID      | mtxid   | 対象ミューテックスのID番号                    |
| ACVCT * | p_acvct | アクセス許可ベクタを入れたパケットへのポインタ（静的APIを除く） |

\* アクセス許可ベクタ（パケットの内容）

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

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー  |
|         | ・非タスクコンテキストからの呼び出し [s] 【NGKI2044】  |
|         | ・CPUロック状態からの呼び出し [s] 【NGKI2045】  |
| E_ID    | 不正ID番号   |
|         | ・mtxidが有効範囲外 [s] 【NGKI2046】  |
| E_RSATR | 予約属性   |
|         | ・対象ミューテックスが属する保護ドメインの囲みの中（対象ミューテックスが無所属の場合は、保護ドメインの囲みの外）に記述されていない [S] 【NGKI2047】 |
|         | ・対象ミューテックスが属するクラスの囲みの中に記述されていない [SM] 【NGKI2048】                                  |
| E_NOEXS | オブジェクト未登録  |
|         | ・対象ミューテックスが未登録 【NGKI2049】  |
| E_OACV  | オブジェクトアクセス違反   |
|         | ・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] 【NGKI2050】                                       |
| E_MACV  | メモリアクセス違反  |
|         | ・p_acvctが指すメモリ領域への読み出しが許可されていない [s] 【NGKI2051】                                   |
| E_OBJ   | オブジェクト状態エラー  |
|         | ・対象ミューテックスは静的APIで生成された [s] 【NGKI2052】  |
|         | ・対象ミューテックスに対してアクセス許可ベクタが設定済み [S] 【NGKI2053】                                      |

#### 【機能】

mtxidで指定したミューテックス（対象ミューテックス）のアクセス許可ベクタ（4つのアクセス許可パターンの組）を、各パラメータで指定した値に設定する 【NGKI2054】。

静的APIにおいては、`mtxid`はオブジェクト識別名、`acptn1`～`acptn4`は整数定数式パラメータである【NGKI2055】。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、`SAC mtx`のみをサポートする【HRPS0133】。ただし、動的生成機能拡張パッケージでは、`sac_mtx`もサポートする【HRPS0195】。

`del_mtx` ミューテックスの削除 [TD] 【NGKI2056】

#### 【C言語API】

```
ER ercd = del_mtx(ID mtxid)
```

#### 【パラメータ】

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

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー                                  |
|         | ・非タスクコンテキストからの呼び出し 【NGKI2057】              |
|         | ・CPUロック状態からの呼び出し 【NGKI2058】                |
| E_ID    | 不正ID番号                                     |
|         | ・ <code>mtxid</code> が有効範囲外 【NGKI2059】     |
| E_NOEXS | オブジェクト未登録                                  |
|         | ・対象ミューテックスが未登録 【NGKI2060】                  |
| E_OACV  | オブジェクトアクセス違反                               |
|         | ・対象ミューテックスに対する管理操作が許可されていない [P] 【NGKI2061】 |
| E_OBJ   | オブジェクト状態エラー                                |
|         | ・対象ミューテックスは静的APIで生成された 【NGKI2062】          |

#### 【機能】

`mtxid`で指定したミューテックス（対象ミューテックス）を削除する。具体的な振舞いは以下の通り。

対象ミューテックスの登録が解除され、そのミューテックスIDが未使用の状態  
NGKI2063】。対象ミューテックスをロックしているタスクがある

に戻される【

場合には、そのタスクがロックしているミューテックスのリストから対象ミューテックスが削除され、必要な場合にはそのタスクの現在優先度が変更される【NGKI2064】。また、対象ミューテックスの待ち行列につながれたタスクは、待ち行列の先頭のタスクから順に待ち解除される【NGKI2065】。待ち解除されたタスクには、待ち状態となったサービスコールからE\_DLTエラーが返る【NGKI2066】。

#### 【使用上の注意】

対象ミューテックスをロックしているタスクには、ミューテックスが削除されたことが通知されず、そのミューテックスをロック解除する時点でエラーとなる。これが不都合な場合には、ミューテックスを削除しようとするタスクがミューテックスをロックした状態で、ミューテックスを削除すればよい。

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およびカーネル内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をサポートしない【ASPS0162】。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、del\_mtxをサポートしない【HRPS0134】。ただし、動的生成機能拡張パッケージでは、del\_mtxをサポートする【HRPS0196】。

|          |                                     |
|----------|-------------------------------------|
| loc_mtx  | ミューテックスのロック [T] 【NGKI2067】          |
| ploc_mtx | ミューテックスのロック（ポーリング）[T] 【NGKI2068】    |
| tloc_mtx | ミューテックスのロック（タイムアウト付き）[T] 【NGKI2069】 |

#### 【C言語API】

```
ER ercd = loc_mtx(ID mtxid)
ER ercd = ploc_mtx(ID mtxid)
ER ercd = tloc_mtx(ID mtxid, TMO tmout)
```

#### 【パラメータ】

|     |       |                        |
|-----|-------|------------------------|
| ID  | mtxid | 対象ミューテックスのID番号         |
| TMO | tmout | タイムアウト時間 (tloc_mtxの場合) |

#### 【リターンパラメータ】

ER

ercd

正常終了 (E\_OK) またはエラーコード

## 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し 【NGKI2070】<br>・CPUロック状態からの呼出し 【NGKI2071】<br>・ディスパッチ保留状態からの呼出し (ploc_mtxを除く) 【NGKI2072】 |
| E_NOSPT | 未サポート機能<br>・制約タスクからの呼出し (ploc_mtxを除く) 【NGKI2073】  |
| E_ID    | 不正ID番号<br>・mtxidが有効範囲外 【NGKI2074】   |
| E_PAR   | パラメータエラー<br>・tmoutが無効 (tloc_mtxの場合) 【NGKI2075】  |
| E_NOEXS | オブジェクト未登録<br>・対象ミューテックスが未登録 [D] 【NGKI2076】  |
| E_OACV  | オブジェクトアクセス違反<br>・対象ミューテックスに対する通常操作1が許可されていない [P]<br>【NGKI2077】  |
| E_ILUSE | サービスコール不正使用<br>・条件については機能の項を参照  |
| E_OBJ   | オブジェクト状態エラー<br>・対象ミューテックスが自タスクによってロックされている<br>【NGKI3609】  |
| E_TMOUT | ポーリング失敗またはタイムアウト (loc_mtxを除く) 【NGKI2078】  |
| E_RLWAI | 待ち禁止状態または待ち状態の強制解除 (ploc_mtxを除く)<br>【NGKI2079】  |
| 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)<br>【NGKI2080】  |

## 【機能】

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

対象ミューテックスがロックされていない場合には、自タスクによってロックされている状態になる【NGKI2081】。自タスクがロックしているミューテックスのリストに対象ミューテックスが追加され、必要な場合には自タスクの現在優先度が変更される【NGKI2082】。

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

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

## 【仕様変更の経緯】

この仕様のRelease 1.6以前では、対象ミューテックスが自タスクによってロックされている場合には、E\_ILUSEエラーとなることとしていたが、Release 1.7以降でE\_OBJエラーに変更した。これは、ミューテックスを用いて、リエンントロックを実現できるようにするためである。

unl\_mtx ミューテックスのロック解除 [T] 【NGKI2086】

## 【C言語API】

```
ER ercd = unl_mtx(ID mtxid)
```

### 【パラメータ】

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

### 【リターンパラメータ】

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

### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー                                   |
|         | ・非タスクコンテキストからの呼び出し 【NGKI2087】               |
|         | ・CPUロック状態からの呼び出し 【NGKI2088】                 |
| E_ID    | 不正ID番号                                      |
|         | ・mtxidが有効範囲外 【NGKI2089】                     |
| E_NOEXS | オブジェクト未登録                                   |
|         | ・対象ミューテックスが未登録 [D] 【NGKI2090】               |
| E_OACV  | オブジェクトアクセス違反                                |
|         | ・対象ミューテックスに対する通常操作1が許可されていない [P] 【NGKI3273】 |
| E_OBJ   | オブジェクト状態エラー                                 |
|         | ・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 【NGKI3610】     |

## 【機能】

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

まず、自タスクがロックしているミューテックスのリストから対象ミューテックスが削除され、必要な場合には自タスクの現在優先度が変更される 【NGKI2091】。

対象ミューテックスの待ち行列にタスクが存在する場合には、待ち行列の先頭のタスクが待ち解除される【NGKI2092】。対象ミューテックスは、待ち解除されたタスクによってロックされている状態になる【NGKI2093】。待ち解除されたタスクがロックしているミューテックスのリストに対象ミューテックスが追加され、必要な場合にはそのタスクの現在優先度が変更される【NGKI2094】。待ち解除されたタスクには、待ち状態となったサービスコールからE\_OKが返る【NGKI2095】。

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

ini\_mtx ミューテックスの再初期化 [T] 【NGKI2098】

#### 【C言語API】

```
ER ercd = ini_mtx(ID mtxid)
```

#### 【パラメータ】

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

#### 【リターンパラメータ】

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

#### 【エラーコード】

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

#### 【機能】

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

対象ミューテックスのロック状態は、ロックされていない状態に初期化される

【

NGKI2104】. 対象ミューテックスをロックしているタスクがある場合には、  
そのタスクがロックしているミューテックスのリストから対象ミューテックス  
が削除され、必要な場合にはそのタスクの現在優先度が変更される

【

NGKI2105】. また、対象ミューテックスの待ち行列につながれたタスクは、  
待ち行列の先頭のタスクから順に待ち解除される【NGKI2106】. 待ち解除され  
たタスクには、待ち状態となったサービスコールからE\_DLTエラーが返る【NGKI2107】.

#### 【使用上の注意】

対象ミューテックスをロックしているタスクには、ミューテックスが再初期化  
されたことが通知されず、そのミューテックスをロック解除する時点でエラー  
となる。これが不都合な場合には、ミューテックスを再初期化しようとするタ  
スクがミューテックスをロックした状態で、ミューテックスを再初期化すればよい。

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  
およびカーネル内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、カーネル内の割込  
み禁止時間が長くなるため、注意が必要である。

ミューテックスを再初期化した場合に、アプリケーションとの整合性を保つのは、  
アプリケーションの責任である。

#### 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである。

ref\_mtx ミューテックスの状態参照 [T] 【NGKI2108】

#### 【C言語API】

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

#### 【パラメータ】

|          |         |                                |
|----------|---------|--------------------------------|
| ID       | mtxid   | 対象ミューテックスのID番号                 |
| T_RMTX * | pk_rmtx | ミューテックスの現在状態を入れるパケットへ<br>のポインタ |

#### 【リターンパラメータ】

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

\* ミューテックスの現在状態 (パケットの内容)

ID            htskid      ミューテックスをロックしているタスクのID番号

ID            wtskid      ミューテックスの待ち行列の先頭のタスクのID  
番号

### 【エラーコード】

E\_CTX      コンテキストエラー

  ・非タスクコンテキストからの呼び出し 【NGKI2109】

  ・CPUロック状態からの呼び出し 【NGKI2110】

E\_ID        不正ID番号

  ・mtxidが有効範囲外 【NGKI2111】

E\_NOEXS     オブジェクト未登録

  ・対象ミューテックスが未登録 [D] 【NGKI2112】

E\_OACV      オブジェクトアクセス違反

  ・対象ミューテックスに対する参照操作が許可されていない [P]  
    【NGKI2113】

E\_MACV     メモリアクセス違反

  ・pk\_rmtxが指すメモリ領域への書き込みアクセスが許可されて  
    いない [P] 【NGKI2114】

### 【機能】

mtxidで指定したミューテックス（対象ミューテックス）の現在状態を参照する。

参照した現在状態は、pk\_rmtxで指定したパケットに返される。

対象ミューテックスがロックされていない場合、htskidにはTSK\_NONE (=0) が返る【NGKI2115】。

対象ミューテックスの待ち行列にタスクが存在しない場合、wtskidには TSK\_NONE (=0)  
) が返る【NGKI2116】。

### 【使用上の注意】

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

## 1.4.7. メッセージバッファ

メッセージバッファは、指定した長さのバイト列をメッセージとして、FIFO順  
で送受信するための同期・通信オブジェクトである。メッセージバッファは、  
と呼ぶID番号によって識別する【NGKI3291】。

各メッセージバッファが持つ情報は次の通り【NGKI3292】。

メッセージバッファID

- ・メッセージバッファ属性
- ・最大メッセージサイズ
- ・メッセージバッファ管理領域
- ・送信待ち行列（メッセージバッファへの送信待ち状態のタスクのキュー）
- ・受信待ち行列（メッセージバッファからの受信待ち状態のタスクのキュー）
- ・アクセス許可ベクタ（保護機能対応カーネルの場合）
- ・属する保護ドメイン（保護機能対応カーネルの場合）
- ・属するクラス（マルチプロセッサ対応カーネルの場合）

メッセージバッファ管理領域は、メッセージバッファに送信されたメッセージを、送信された順に格納しておくためのメモリ領域である。メッセージバッファ生成時の指定により、メッセージバッファ管理領域のサイズを0とすることができます【NGKI3293】。

保護機能対応カーネルにおいて、メッセージバッファ管理領域は、カーネルの用いるオブジェクト管理領域として扱われる【NGKI3294】。

送信待ち行列は、メッセージバッファに対してメッセージが送信できるまで待っている状態（メッセージバッファへの送信待ち状態）のタスクが、メッセージを送信できる順序でつながれているキューである。また、受信待ち行列は、メッセージバッファからメッセージが受信できるまで待っている状態（メッセージバッファからの受信待ち状態）のタスクが、メッセージを受信できる順序でつながれているキューである。

メッセージバッファ属性には、次の属性を指定することができます【NGKI3295】。

|         |       |                    |
|---------|-------|--------------------|
| 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【NGKI3296】。受信待ち行列は、FIFO順に固定されている【NGKI3297】。

メッセージバッファ機能に関するカーネル構成マクロは次の通り。

|            |  |
|------------|--|
| TNUM_MBFID | 登録できるメッセージバッファの数（動的生成対応でないカーネルでは、静的APIによって登録されたメッセージバッファの数に一致）【NGKI3298】 |
|------------|--|

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、メッセージバッファ機能をサポートしない【ASPS0202】。ただし、メッセージバッファ機能拡張パッケージを用いると、メッセージバッファ機能を追加することができる【ASPS0203】。

#### 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、メッセージバッファ機能をサポートしない【FMPS0167】。

## 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、メッセージバッファ機能をサポートしない【HRPS0168】。  
ただし、メッセージバッファ機能拡張パッケージを用いると、メッセージバッファ機能を追加することができる【HRPS0169】。

## 【μITRON4.0仕様との関係】

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

|          |                              |
|----------|------------------------------|
| CRE_MBF  | メッセージバッファの生成 [S] 【NGKI3299】  |
| acre_mbf | メッセージバッファの生成 [TD] 【NGKI3300】 |

## 【静的API】

```
CRE_MBF(ID mbfid, { ATR mbfatr, uint_t maxmsz, SIZE mbfsz, void *mbfmb })
```

## 【C言語API】

```
ER_ID mbfid = acre_mbf(const T_CMBF *pk_cmbf)
```

## 【パラメータ】

|          |         |   |
|----------|---------|---|
| ID       | mbfid   | 生成するメッセージバッファのID番号 (CRE_MBFの場合)         |
| T_CMBF * | pk_cmbf | メッセージバッファの生成情報を入ったパケットへのポインタ (静的APIを除く) |

\* メッセージバッファの生成情報 (パケットの内容)

|        |        |                             |
|--------|--------|-----------------------------|
| ATR    | mbfatr | メッセージバッファ属性                 |
| uint_t | maxmsz | メッセージバッファの最大メッセージサイズ (バイト数) |
| SIZE   | mbfsz  | メッセージバッファ管理領域のサイズ (バイト数)    |
| void * | mbfmb  | メッセージバッファ管理領域の先頭番地          |

## 【リターンパラメータ】

|       |       |                                     |
|-------|-------|-------------------------------------|
| ER_ID | mbfid | 生成されたメッセージバッファのID番号 (正の値) またはエラーコード |
|-------|-------|-------------------------------------|

## 【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し [s] 【NGKI3301】<br>・CPUロック状態からの呼出し [s] 【NGKI3302】  |
| E_RSATR | 予約属性<br>・mbfatrが無効 【NGKI3303】<br>・属する保護ドメインの指定が有効範囲外 [sP] 【NGKI3304】<br>・属するクラスの指定が有効範囲外 [sM] 【NGKI3305】<br>・クラスの団みの中に記述されていない [sM] 【NGKI3306】 |
| E_NOSPT | 未サポート機能<br>・条件については各カーネルにおける規定の項を参照  |
| E_PAR   | パラメータエラー<br>・maxmszが0以下 【NGKI3307】<br>・mbfszが負の値 [s] 【NGKI3308】<br>・その他の条件については機能の項を参照   |
| E_OACV  | オブジェクトアクセス違反<br>・システム状態に対する管理操作が許可されていない [sP] 【NGKI3309】   |
| E_MACV  | メモリアクセス違反<br>・pk_cmbfが指すメモリ領域への読み出しが許可されていない [sP] 【NGKI3310】   |
| E_NOID  | ID番号不足<br>・割り付けられるメッセージバッファIDがない [sD] 【NGKI3311】   |
| E_NOMEM | メモリ不足<br>・メッセージバッファ管理領域が確保できない 【NGKI3312】  |
| E_OBJ   | オブジェクト状態エラー<br>・mbfidで指定したメッセージバッファが登録済み (CRE_MBF の場合) 【NGKI3313】<br>・その他の条件については機能の項を参照   |

## 【機能】

各パラメータで指定したメッセージバッファ生成情報に従って、メッセージバッファを生成する。mbfszとmbfmbからメッセージバッファ管理領域が設定され、格納されているメッセージがない状態に初期化される【NGKI3314】。また、送信待ち行列と受信待ち行列は、空の状態に初期化される【NGKI3315】。

静的APIにおいては、mbfidはオブジェクト識別名、mbfatr、maxmsz、mbfszは整数定数式パラメータ、mbfmbは一般定数式パラメータである【NGKI3316】。コンフィギュレータは、静的APIのメモリ不足 (E\_NOMEM) エラーを検出することができない【NGKI3317】。

mbfmbをNULLとした場合、mbfszで指定したサイズのメッセージバッファ管理領域が、コンフィギュレータまたはカーネルにより確保される【NGKI3318】。  
mbfszにターゲット定義の制約に合致しないサイズを指定した時には、ターゲット定義の制約に合致するように大きい方に丸めたサイズで確保される【NGKI3319】。

〔mbfmbにNULL以外を指定した場合〕

mbfmbにNULL以外を指定した場合、mbfmbとmbfszで指定したメッセージバッファ管理領域は、アプリケーションで確保しておく必要がある【NGKI3320】。メッセージバッファ管理領域をアプリケーションで確保するために、次のマクロを用意している【NGKI3321】。

|                           |  |
|---------------------------|--|
| TSZ_MBFMB(msgcnt, msgsz)  | msgszで指定したサイズのメッセージを、msgcntで指定した数だけ格納できるメッセージバッファ管理領域のサイズ（バイト数）              |
| TCNT_MBFMB(msgcnt, msgsz) | msgszで指定したサイズのメッセージを、msgcntで指定した数だけ格納できるメッセージバッファ管理領域を確保するために必要なMB_T型の配列の要素数 |

これらを用いてメッセージバッファ管理領域を確保する方法は次の通り【NGKI3322】。

```
MB_T <メッセージバッファ管理領域の変数名>[TCNT_MBFMB(msgcnt, msgsz)];
```

この時、mbfszにはTSZ\_MBFMB(msgcnt, msgsz)を、mbfmbには<メッセージバッファ管理領域の変数名>を指定する【NGKI3323】。

この方法に従わず、mbfmbとmbfszにターゲット定義の制約に合致しない先頭番地やサイズを指定した時には、E\_PARエラーとなる【NGKI3324】。また、保護機能対応カーネルにおいて、mbfmbとmbfszで指定したメッセージバッファ管理領域がカーネル専用のメモリオブジェクトに含まれない場合、E\_OBJエラーとなる【NGKI3325】。

なお、TSZ\_MBFMBは、mbfmbにNULLを指定した場合にも、メッセージバッファ管理領域のサイズを決めるために用いることができる。

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルのメッセージバッファ機能拡張パッケージでは、CRE\_MBFのみをサポートする【ASPS0204】。また、mbfmbにはNULLのみを指定することができる。  
以外を指定した場合には、E\_NOSPTエラーとなる【ASPS0205】。

ポートする  
NULL

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルのメッセージバッファ機能拡張パッケージでは、CRE\_MBFのみをサポートする【HRPS0170】。また、mbfmbにはNULLのみを指定することができる。  
以外を指定した場合には、E\_NOSPTエラーとなる【HRPS0171】。

ポートする  
NULL

#### 【μITRON4.0仕様との関係】

μITRON4.0/PX仕様にあわせて、メッセージバッファ生成情報の最後のパラメータを、mbf（メッセージバッファ領域の先頭番地）から、mbfmb（メッセージバッファ管理領域の先頭番地）に改名した。また、TSZ\_MBFをTSZ\_MBFMBに改名した。

タを、

TCNT\_MBFMBを新設し、メッセージバッファ管理領域をアプリケーションで確保する方法を規定した。

AID\_MBF 割付け可能なメッセージバッファIDの数の指定〔SD〕【NGKI3326】

【静的API】

AID\_MBF(uint\_t nombf)

【パラメータ】

uint\_t nombf 割付け可能なメッセージバッファIDの数

【エラーコード】

E\_RSATR 予約属性

- ・保護ドメインの囲みの中に記述されている〔P〕【NGKI3434】
- ・クラスの囲みの中に記述されていない〔M〕【NGKI3327】

E\_PAR パラメータエラー

- ・nombfが負の値【NGKI3328】

【機能】

nombfで指定した数のメッセージバッファIDを、メッセージバッファを生成するサービスコールによって割付け可能なメッセージバッファIDとして確保する【NGKI3329】。

nombfは整数定数式パラメータである【NGKI3330】。

SAC\_MBF メッセージバッファのアクセス許可ベクタの設定〔SP〕【NGKI3331】

sac\_mbf メッセージバッファのアクセス許可ベクタの設定〔TPD〕【NGKI3332】

【静的API】

SAC\_MBF(ID mbfid, { ACPTN acptn1, ACPTN acptn2,  
ACPTN acptn3, ACPTN acptn4 })

【C言語API】

ER ercd = sac\_mbf(ID mbfid, const ACVCT \*p\_acvct)

【パラメータ】

|                      |                      |                                   |
|----------------------|----------------------|-----------------------------------|
| ID                   | <code>mbfid</code>   | 対象メッセージバッファのID番号                  |
| <code>ACVCT</code> * | <code>p_acvct</code> | アクセス許可ベクタを入れたパケットへのポインタ（静的APIを除く） |

\* アクセス許可ベクタ（パケットの内容）

|       |                     |                  |
|-------|---------------------|------------------|
| ACPTN | <code>acptn1</code> | 通常操作1のアクセス許可パターン |
| ACPTN | <code>acptn2</code> | 通常操作2のアクセス許可パターン |
| ACPTN | <code>acptn3</code> | 管理操作のアクセス許可パターン  |
| ACPTN | <code>acptn4</code> | 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 | <code>ercd</code> | 正常終了（E_OK）またはエラーコード |
|----|-------------------|---------------------|

#### 【エラーコード】

|                      |   |
|----------------------|---|
| <code>E_CTX</code>   | コンテキストエラー   |
|                      | ・非タスクコンテキストからの呼び出し [s] 【NGKI3333】   |
|                      | ・CPUロック状態からの呼び出し [s] 【NGKI3334】   |
| <code>E_ID</code>    | 不正ID番号  |
|                      | ・ <code>mbfid</code> が有効範囲外 [s] 【NGKI3335】  |
| <code>E_RSATR</code> | 予約属性  |
|                      | ・対象メッセージバッファが属する保護ドメインの固みの中<br>(対象メッセージバッファが無所属の場合は、保護ドメインの固みの外) に記述されていない [S] 【NGKI3336】 |
|                      | ・対象メッセージバッファが属するクラスの固みの中に記述<br>されていない [SM] 【NGKI3337】                                     |
| <code>E_NOEXS</code> | オブジェクト未登録   |
|                      | ・対象メッセージバッファが未登録 【NGKI3338】   |
| <code>E_OACV</code>  | オブジェクトアクセス違反  |
|                      | ・対象メッセージバッファに対する管理操作が許可されてい<br>ない [s] 【NGKI3339】  |
| <code>E_MACV</code>  | メモリアクセス違反   |
|                      | ・ <code>p_acvct</code> が指すメモリ領域への読み出しが許可されて<br>いない [s] 【NGKI3340】                         |
| <code>E_OBJ</code>   | オブジェクト状態エラー   |
|                      | ・対象メッセージバッファは静的APIで生成された [s] 【NGKI3341】   |
|                      | ・対象メッセージバッファに対してアクセス許可ベクタが設<br>定済み [S] 【NGKI3342】   |

#### 【機能】

`mbfid`で指定したメッセージバッファ（対象メッセージバッファ）のアクセス許可ベクタ（4つのアクセス許可パターンの組）を、各パラメータで指定した値に設定する【NGKI3343】。

静的APIにおいては、mbfidはオブジェクト識別名、acptn1～acptn4は整数定数式パラメータである【NGKI3344】。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルのメッセージバッファ機能拡張パッケージでは、SAC\_MBFのみをサポートする【HRPS0172】。

**del\_mbf** メッセージバッファの削除 [TD] 【NGKI3345】

#### 【C言語API】

ER ercd = del\_mbf(ID mbfid)

#### 【パラメータ】

ID mbfid 対象メッセージバッファのID番号

#### 【リターンパラメータ】

ER ercd 正常終了 (E\_OK) またはエラーコード

#### 【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー                                  |
|         | ・非タスクコンテキストからの呼び出し【NGKI3346】               |
|         | ・CPUロック状態からの呼び出し【NGKI3347】                 |
| E_ID    | 不正ID番号                                     |
|         | ・mbfidが有効範囲外【NGKI3348】                     |
| E_NOEXS | オブジェクト未登録                                  |
|         | ・対象メッセージバッファが未登録【NGKI3349】                 |
| E_OACV  | オブジェクトアクセス違反                               |
|         | ・対象メッセージバッファに対する管理操作が許可されていない[P]【NGKI3350】 |
| E_OBJ   | オブジェクト状態エラー                                |
|         | ・対象メッセージバッファは静的APIで生成された【NGKI3351】         |

#### 【機能】

mbfidで指定したメッセージバッファ（対象メッセージバッファ）を削除する。具体的な振舞いは以下の通り。

対象メッセージバッファの登録が解除され、そのメッセージバッファIDが未使

用の状態に戻される【NGKI3352】。また、対象メッセージバッファの送信待ち行列と受信待ち行列につながれたタスクは、それぞれの待ち行列の先頭のタスクから順に待ち解除される【NGKI3353】。待ち解除されたタスクには、待ち状態となったサービスコールからE\_DLTエラーが返る【NGKI3354】。

メッセージバッファの生成時に、メッセージバッファ管理領域がカーネルによって確保された場合は、そのメモリ領域が解放される【NGKI3355】。

#### 【補足説明】

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

#### 【使用上の注意】

del\_mbfにより複数のタスクが待ち解除される場合、サービスコールの処理時間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カーネルのメッセージバッファ機能拡張パッケージでは、del\_mbfをサポートしない【ASPS0207】。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルのメッセージバッファ機能拡張パッケージでは、del\_mbfをサポートしない【HRPS0173】。

|          |                                       |
|----------|---------------------------------------|
| snd_mbf  | メッセージバッファへの送信 [T] 【NGKI3356】          |
| psnd_mbf | メッセージバッファへの送信（ポーリング）[T] 【NGKI3357】    |
| tsnd_mbf | メッセージバッファへの送信（タイムアウト付き）[T] 【NGKI3358】 |

#### 【C言語API】

```
ER ercd = snd_mbf(ID mbfid, const void *msg, uint_t msgsz)
ER ercd = psnd_mbf(ID mbfid, const void *msg, uint_t msgsz)
ER ercd = tsnd_mbf(ID mbfid, const void *msg, uint_t msgsz, TM0 tmout)
```

#### 【パラメータ】

|        |       |                       |
|--------|-------|-----------------------|
| ID     | mbfid | 対象メッセージバッファのID番号      |
| void * | msg   | 送信メッセージの先頭番地          |
| uint_t | msgsz | 送信メッセージのサイズ（バイト数）     |
| TM0    | tmout | タイムアウト時間（tsnd_mbfの場合） |

#### 【リターンパラメータ】

ER

ercd

正常終了 (E\_OK) またはエラーコード

## 【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し 【NGKI3359】<br>・CPUロック状態からの呼出し 【NGKI3360】<br>・ディスパッチ保留状態からの呼出し (psnd_mbfを除く)<br>【NGKI3361】 |
| E_NOSPT | 未サポート機能<br>・制約タスクからの呼出し (psnd_mbfを除く) 【NGKI3362】   |
| E_ID    | 不正ID番号<br>・mbfidが有効範囲外 【NGKI3363】  |
| E_PAR   | パラメータエラー<br>・msgszが有効範囲 (0より大きく対象メッセージバッファの<br>最大メッセージサイズ以下) 外 【NGKI3364】<br>・toutが無効 (tsnd_mbfの場合) 【NGKI3365】           |
| E_NOEXS | オブジェクト未登録<br>・対象メッセージバッファが未登録 [D] 【NGKI3366】   |
| E_OACV  | オブジェクトアクセス違反<br>・対象メッセージバッファに対する通常操作1が許可されて<br>いない [P] 【NGKI3367】  |
| E_MACV  | メモリアクセス違反<br>・msgとmsgszが指すメモリ領域への読み出しアクセスが許可さ<br>れていない [P] 【NGKI3368】  |
| E_TMOUT | ポーリング失敗またはタイムアウト (snd_mbfを除く) 【NGKI3369】   |
| E_RLWAI | 待ち禁止状態または待ち状態の強制解除 (psnd_mbfを除く)<br>【NGKI3370】   |
| E_DLT   | 待ちオブジェクトの削除または再初期化 (psnd_mbfを除く)<br>【NGKI3371】   |

## 【機能】

mbfidで指定したメッセージバッファ（対象メッセージバッファ）に、msgとmsgszで指定したメッセージ（送信メッセージ）を送信する。具体的な振舞いは以下の通り。

対象メッセージバッファの受信待ち行列にタスクが存在する場合には、受信待  
ち行列の先頭のタスクが、送信メッセージを受信し、待ち解除される  
【NGKI3372】。待ち解除されたタスクには、待ち状態となったサービスコール  
から、受信したメッセージのサイズが返る【NGKI3373】。

対象メッセージバッファの受信待ち行列にタスクが存在しない場合で、送信待  
ち行列に自タスクより優先してメッセージを送信できるタスクが存在せず、メ  
ッセージバッファ管理領域に送信メッセージを格納するスペースがある場合には、  
FIFO順でメッセージバッファ管理領域に格納される  
【送信メッセージが、

NGKI3374】. ここで、送信待ち行列に自タスクより優先してメッセージを送信できるタスクが存在するとは、送信待ち行列がFIFO順の場合には送信待ち行列に何らかのタスクが存在すること、タスクの優先度順の場合には自タスクと優先度が同じかより高いタスクが存在することを意味する。

対象メッセージバッファの受信待ち行列にタスクが存在しない場合で、送信待ち行列に自タスクより優先してメッセージを送信できるタスクが存在するか、メッセージバッファ管理領域に送信メッセージを格納するスペースがない場合には、自タスクはメッセージバッファへの送信待ち状態となり、対象メッセージバッファの送信待ち行列につながれる【NGKI3375】.

メッセージバッファの送信待ち行列の先頭につながれているタスクが、  
により強制終了した場合や、rel\_wai / irel\_waiやタイムアウトにより  
待ち解除された場合、新たに送信待ち行列の先頭になったタスクの送信メッセージを、メッセージバッファ管理領域に格納できる可能性がある。そのため、これらの場合には、メッセージバッファからの受信によりメッセージバッファ管理領域に空きができた時の処理【NGKI3393】【NGKI3394】【NGKI3395】と同じ処理が行われる【NGKI3419】。さらに、送信待ち行列がタスクの優先度順の時  
chng\_priやミューテックスの操作によりタスクの優先度が変化し、送信待ち行列の先頭につながれているタスクが変わった場合にも、同じ処理が行われる【NGKI3420】。  
ter\_tsk  
には、

#### 【使用上の注意】

送信待ち行列の先頭につながれているタスクの強制終了、待ち解除、優先度変更に伴う処理で、送信待ち行列につながっていたタスクが複数待ち解除される場合がある。この時、サー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。

|           |  |
|-----------|--|
| recv_mbf  | メッセージバッファからの受信 [T] 【NGKI3376】          |
| precv_mbf | メッセージバッファからの受信（ポーリング）[T] 【NGKI3377】    |
| trcv_mbf  | メッセージバッファからの受信（タイムアウト付き）[T] 【NGKI3378】 |

#### 【C言語API】

```
ER_UINT msgsz = recv_mbf(ID mbfid, void *msg)
ER_UINT msgsz = precv_mbf(ID mbfid, void *msg)
ER_UINT msgsz = trcv_mbf(ID mbfid, void *msg, TMO tmout)
```

#### 【パラメータ】

|        |       |                        |
|--------|-------|------------------------|
| ID     | mbfid | 対象メッセージバッファのID番号       |
| void * | msg   | 受信メッセージを入れるメモリ領域の先頭番地  |
| TMO    | tmout | タイムアウト時間 (trcv_mbfの場合) |

## 【リターンパラメータ】

|         |       |                          |
|---------|-------|--------------------------|
| ER_UINT | msgsz | 受信メッセージサイズ（正の値）またはエラーコード |
|---------|-------|--------------------------|

## 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し 【NGKI3379】<br>・CPUロック状態からの呼出し 【NGKI3380】<br>・ディスパッチ保留状態からの呼出し（recv_mbfを除く）<br>【NGKI3381】 |
| E_NOSPT | 未サポート機能<br>・制約タスクからの呼出し（recv_mbfを除く） 【NGKI3382】   |
| E_ID    | 不正ID番号<br>・mbfidが有効範囲外 【NGKI3383】   |
| E_PAR   | パラメータエラー<br>・tmoutが無効（recv_mbfの場合） 【NGKI3384】   |
| E_NOEXS | オブジェクト未登録<br>・対象メッセージバッファが未登録〔D〕 【NGKI3385】   |
| E_OACV  | オブジェクトアクセス違反<br>・対象メッセージバッファに対する通常操作2が許可されていない〔P〕 【NGKI3386】  |
| E_MACV  | メモリアクセス違反<br>・msgを先頭番地とし、対象メッセージバッファの最大メッセージサイズ分のメモリ領域への書き込みアクセスが許可されていない〔P〕 【NGKI3387】                                 |
| E_TMOUT | ポーリング失敗またはタイムアウト（recv_mbfを除く） 【NGKI3388】  |
| E_RLWAI | 待ち禁止状態または待ち状態の強制解除（recv_mbfを除く）<br>【NGKI3389】   |
| E_DLT   | 待ちオブジェクトの削除または再初期化（recv_mbfを除く）<br>【NGKI3390】   |

## 【機能】

mbfidで指定したメッセージバッファ（対象メッセージバッファ）からメッセージを受信する。メッセージの受信に成功した場合、受信したメッセージはmsgを先頭番地とするメモリ領域に格納され、そのサイズはサービスコールの返値と NGKI3391】。具体的な振舞いは以下の通り。

して返される【

対象メッセージバッファのメッセージバッファ管理領域にメッセージが格納されている場合には、メッセージバッファ管理領域の先頭に格納されたメッセージ NGKI3392】。また、送信待ち行列にタスクが存在し、メッセージバッファ管理領域に送信待ち行列の先頭のタスクの送信メッセージを格納するスペースがある場合には、送信メッセージがFIFO順でデータキュー管理領域に格納され、そのタスクは待ち解除される【NGKI3393】。待ち解除されたタス

ジを受信する【

クには、待ち状態となったサービスコールからE\_OKが返る【NGKI3394】。この処理を、送信待ち行列にタスクが存在しなくなるか、メッセージバッファ管理領域に送信待ち行列の先頭のタスクの送信メッセージを格納するスペースがなくなるまで繰り返す【NGKI3395】。

対象メッセージバッファのメッセージバッファ管理領域にメッセージが格納されておらず、送信待ち行列にタスクが存在する場合には、送信待ち行列の先頭のタスクの送信メッセージを受信する【NGKI3396】。送信待ち行列の先頭のタスクは、待ち解除される【NGKI3397】。待ち解除されたタスクには、待ち状態となったサービスコールからE\_OKが返る【NGKI3398】。

対象メッセージバッファのメッセージバッファ管理領域にメッセージが格納されておらず、送信待ち行列にタスクが存在しない場合には、自タスクはメッセージバッファからの受信待ち状態となり、対象メッセージバッファの受信待ち行列につながれる【NGKI3399】。

#### 【使用上の注意】

メッセージバッファ管理領域に格納されたメッセージを受信した結果、送信待ち行列につながっていたタスクが複数待ち解除される場合がある。この時、サー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。

ini\_mbf メッセージバッファの再初期化 [T] 【NGKI3400】

#### 【C言語API】

```
ER ercd = ini_mbf(ID mbfid)
```

#### 【パラメータ】

|    |       |                  |
|----|-------|------------------|
| ID | mbfid | 対象メッセージバッファのID番号 |
|----|-------|------------------|

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI3401】<br>・CPUロック状態からの呼出し【NGKI3402】 |
| E_ID    | 不正ID番号<br>・mbfidが有効範囲外【NGKI3403】                                      |
| E_NOEXS | オブジェクト未登録<br>・対象メッセージバッファが未登録〔D〕【NGKI3404】                            |
| E_OACV  | オブジェクトアクセス違反<br>・対象メッセージバッファに対する管理操作が許可されていない〔P〕【NGKI3405】            |

## 【機能】

mbfidで指定したメッセージバッファ（対象メッセージバッファ）を再初期化する。具体的な振舞いは以下の通り。

対象メッセージバッファのメッセージバッファ管理領域は、格納されているメッセージがない状態に初期化される【NGKI3406】。また、対象メッセージバッファの送信待ち行列と受信待ち行列につながれたタスクは、それぞれの待ち行列の先頭のタスクから順に待ち解除される【NGKI3407】。待ち解除されたタスクには、待ち状態となったサービスコールからE\_DLTエラーが返る【NGKI3408】。

## 【補足説明】

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

## 【使用上の注意】

ini\_mbfにより複数のタスクが待ち解除される場合、サービスコールの処理時間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。

メッセージバッファを再初期化した場合に、アプリケーションとの整合性を保つのは、アプリケーションの責任である。

## 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである。

|         |                             |
|---------|-----------------------------|
| ref_mbf | メッセージバッファの状態参照〔T〕【NGKI3409】 |
|---------|-----------------------------|

## 【C言語API】

|  |
|--|
| ER ercd = ref_mbf(ID mbfid, T_RMBF *pk_rmbf) |
|--|

## 【パラメータ】

|          |         |                              |
|----------|---------|------------------------------|
| ID       | mbfid   | 対象メッセージバッファのID番号             |
| T_RMBF * | pk_rmbf | メッセージバッファの現在状態を入れるパケットへのポインタ |

## 【リターンパラメータ】

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

\*メッセージバッファの現在状態 (パケットの内容)

|        |         |                              |
|--------|---------|------------------------------|
| ID     | stskid  | メッセージバッファの送信待ち行列の先頭のタスクのID番号 |
| ID     | rtskid  | メッセージバッファの受信待ち行列の先頭のタスクのID番号 |
| uint_t | smbfcnt | メッセージバッファ管理領域に格納されているメッセージの数 |
| SIZE   | fmbfsz  | メッセージバッファ管理領域中の空き領域のサイズ      |

## 【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー  |
|         | ・非タスクコンテキストからの呼び出し 【NGKI3410】                      |
|         | ・CPUロック状態からの呼び出し 【NGKI3411】                        |
| E_ID    | 不正ID番号   |
|         | ・mbfidが有効範囲外 【NGKI3412】                            |
| E_NOEXS | オブジェクト未登録  |
|         | ・対象メッセージバッファが未登録 [D] 【NGKI3413】                    |
| E_OACV  | オブジェクトアクセス違反                                       |
|         | ・対象メッセージバッファに対する参照操作が許可されていない [P] 【NGKI3414】       |
| E_MACV  | メモリアクセス違反  |
|         | ・pk_rmbf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] 【NGKI3415】 |

## 【機能】

mbfidで指定したメッセージバッファ（対象メッセージバッファ）の現在状態を参照する。参照した現在状態は、pk\_rmbfで指定したパケットに返される 【NGKI3416】。

対象メッセージバッファの送信待ち行列にタスクが存在しない場合、stskidには TSK\_NONE (=0) が返る 【NGKI3417】。また、受信待ち行列にタスクが存在しない場合、rtskidにはTSK\_NONE (=0) が返る 【NGKI3418】。

## 【使用上の注意】

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

### 1.4.8. スピンロック

スピンロックは、マルチプロセッサ対応カーネルにおいて、割込みのマスクとプロセッサ間ロックの取得により、排他制御を行うための同期・通信オブジェクトである。スピンロックは、スピンロックIDと呼ぶID番号によって識別する【NGKI2117】。

プロセッサ間ロックを取得している間は、CPUロック状態にすることですべてのカーネル管理の割込みがマスクされ、ディスパッチが保留される【NGKI2118】。ロックが他のプロセッサに取得されている場合には、ロックが取得できるまでループによって待つ【NGKI2119】。ロックの取得を待つ間は、CPUロック解除状態であり、割込みはマスクされない【NGKI2120】。プロセッサ間ロックを取得し、CPUロック状態に遷移することを、スピンロックを取得するという。また、プロセッサ間ロックを返却し、CPUロック状態を解除することを、スピンロックを返却するという。

タスクが取得したスピンロックを返却せずに終了した場合や、タスク例外処理ルーチン、割込みハンドラ、割込みサービスルーチン、タイムイベントハンド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るCPU例外によって呼び出されたCPU例外ハンドラが、取得したスpin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では、スピンロックは返却されない【NGKI2123】。

各スピンロックが持つ情報は次の通り【NGKI2124】。

- スピンロック属性
- ロック状態（取得されている状態と取得されていない状態）
- アクセス許可ベクタ（保護機能対応カーネルの場合）
- 属する保護ドメイン（保護機能対応カーネルの場合）
- 属するクラス

スピンロック属性に指定できる属性はない【NGKI2125】。そのためスピンロッタ\_NULLを指定しなければならない【NGKI2126】。

スピンロック機能に関連するカーネル構成マクロは次の通り。

TNUM\_SPNID

登録できるスピンロックの数（動的生成対応でないカーネルでは、静的APIによって登録されたスピンロックの数に一致）【NGKI2127】

## 【補足説明】

CPUロック状態では、スピンロックを取得するサービスコールを呼び出すことができないため、スピンロックを取得しているプロセッサが、さらにスピンロックを取得することはできない。そのため、1つの処理単位が、複数のスピンロックを取得した状態になることはできない。

スピンロックを取得した状態でCPU例外が発生した場合、起動されるCPU例外ハンドラはカーネル管理外のCPU例外ハンドラであり（xsns\_dpn, xsns\_xpnとも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 trueを返す），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 を呼び出した場合には、CPU例外ハンドラからのリターン時に元の状態に戻らない。これは、CPUロック状態の扱いと一貫していないため、注意が必要である。

## 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、スピンロック機能をサポートしない【ASPS0163】。

## 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、スピンロック機能をサポートする【FMPS0138】。

## 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、スピンロック機能をサポートしない【HRPS0135】。

## 【μITRON4.0仕様との関係】

スピンロック機能は、μITRON4.0仕様に定義されていない機能である。

|          |                            |
|----------|----------------------------|
| CRE_SPN  | スピンロックの生成 [SM] 【NGKI2128】  |
| acre_spn | スピンロックの生成 [TMD] 【NGKI2129】 |

## 【静的API】

```
CRE_SPN(ID spnid, { ATR spnattr })
```

## 【C言語API】

```
ER_ID spnid = acre_spn(const T_CSPN *pk_cspn)
```

## 【パラメータ】

|          |         |  |
|----------|---------|--|
| ID       | spnid   | 生成するスピンロックのID番号 (CRE_SPNの場合)             |
| T_CSPN * | pk_cspn | スピンロックの生成情報を入ったパケットへの<br>ポインタ (静的APIを除く) |

\* スピンロックの生成情報 (パケットの内容)

|     |         |          |
|-----|---------|----------|
| ATR | spnattr | スピンロック属性 |
|-----|---------|----------|

### 【リターンパラメータ】

|       |       |                                   |
|-------|-------|-----------------------------------|
| ER_ID | spnid | 生成されたスpinロックのID番号 (正の値) またはエラーコード |
|-------|-------|-----------------------------------|

### 【エラーコード】

|          |   |
|----------|---|
| E_CTX    | コンテキストエラー                                       |
|          | ・非タスクコンテキストからの呼び出し [s] 【NGKI2130】               |
|          | ・CPUロック状態からの呼び出し [s] 【NGKI2131】                 |
| E_RSATTR | 予約属性  |
|          | ・spnattrが無効 【NGKI2132】                          |
|          | ・属する保護ドメインの指定が有効範囲外 [sP] 【NGKI2133】             |
|          | ・属するクラスの指定が有効範囲外 [s] 【NGKI2134】                 |
|          | ・クラスの団みの中に記述されていない [s] 【NGKI2135】               |
| E_OACV   | オブジェクトアクセス違反                                    |
|          | ・システム状態に対する管理操作が許可されていない [sP] 【NGKI2136】        |
| E_MACV   | メモリアクセス違反                                       |
|          | ・pk_cspnが指すメモリ領域への読み出しが許可されていない [sP] 【NGKI2137】 |
| E_NOID   | ID番号不足  |
|          | ・割り付けられるスピンロックIDがない [sD] 【NGKI2138】             |
| E_NORES  | 資源不足  |
|          | ・条件については機能の項を参照                                 |
| E_OBJ    | オブジェクト状態エラー                                     |
|          | ・spnidで指定したスピンロックが登録済み (CRE_SPNの場合) 【NGKI2139】  |

### 【機能】

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化される【NGKI2140】。

静的APIにおいては、spnidはオブジェクト識別名、spnattrは整数定数式パラメータである【NGKI2141】。

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

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【NGKI2143】。

#### 【補足説明】

スピンロックを動的に生成する場合に、生成できるスピンロックの数の上限はAID\_SPNによってチェックされるため、acre\_spnでE\_NORESエラーが返ることはない。

#### 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、CRE\_SPNのみをサポートする【FMPS0139】。

AID\_SPN 割付け可能なスピンロックIDの数の指定 [SMD] 【NGKI2144】

#### 【静的API】

AID\_SPN(uint\_t nospn)

#### 【パラメータ】

uint\_t nospn 割付け可能なスピンロックIDの数

#### 【エラーコード】

E\_RSATR

予約属性

- ・保護ドメインの囲みの中に記述されている [P] 【NGKI3435】
- ・クラスの囲みの中に記述されていない【NGKI2145】

E\_PAR

パラメータエラー

- ・nospnが負の値【NGKI3283】

#### 【機能】

nospnで指定した数のスピンロックIDを、スピンロックを生成するサービスコードによって割付け可能なスピンロックIDとして確保する【NGKI2146】。

nospnは整数定数式パラメータである【NGKI2147】。

SAC\_SPN スピンロックのアクセス許可ベクタの設定 [SPM] 【NGKI2148】

sac\_spn スピンロックのアクセス許可ベクタの設定 [TPMD] 【NGKI2149】

#### 【静的API】

```
SAC_SPN(ID spnid, { ACPTN acptn1, ACPTN acptn2,
                      ACPTN acptn3, ACPTN acptn4 })
```

#### 【C言語API】

```
ER ercd = sac_spn(ID spnid, const ACVCT *p_acvct)
```

#### 【パラメータ】

|         |         |                                   |
|---------|---------|-----------------------------------|
| ID      | spnid   | 対象スピンドルのID番号                      |
| ACVCT * | p_acvct | アクセス許可ベクタを入れたパケットへのポインタ（静的APIを除く） |

\* アクセス許可ベクタ（パケットの内容）

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

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し [s] 【NGKI2150】<br>・CPUロック状態からの呼出し [s] 【NGKI2151】   |
| E_ID    | 不正ID番号<br>・spnidが有効範囲外 [s] 【NGKI2152】   |
| E_RSATR | 予約属性<br>・対象スピンロックが属する保護ドメインの囲みの中（対象スピンロックが無所属の場合は、保護ドメインの囲みの外）に記述されていない [s] 【NGKI2153】<br>・対象スピンロックが属するクラスの囲みの中に記述されていない [s] 【NGKI2154】 |
| E_NOEXS | オブジェクト未登録<br>・対象スピンロックが未登録 【NGKI2155】   |
| E_OACV  | オブジェクトアクセス違反<br>・対象スPINロックに対する管理操作が許可されていない [s] 【NGKI2156】  |
| E_MACV  | メモリアクセス違反<br>・p_acvctが指すメモリ領域への読み出しが許可されていない [s] 【NGKI2157】   |
| E_OBJ   | オブジェクト状態エラー<br>・対象スPINロックは静的APIで生成された [s] 【NGKI2158】<br>・対象スPINロックに対してアクセス許可ベクタが設定済み [s] 【NGKI2159】                                     |

### 【機能】

spnidで指定したスPINロック（対象スPINロック）のアクセス許可ベクタ（4つのアクセス許可パターンの組）を、各パラメータで指定した値に設定する 【NGKI2160】。

静的APIにおいては、spnidはオブジェクト識別名、acptn1～acptn4は整数定数式パラメータである 【NGKI2161】。

del\_spn スPINロックの削除 [TMD] 【NGKI2162】

### 【C言語API】

```
ER ercd = del_spn(ID spnid)
```

### 【パラメータ】

|    |       |                |
|----|-------|----------------|
| ID | spnid | 対象スPINロックのID番号 |
|----|-------|----------------|

### 【リターンパラメータ】

ER

ercd

正常終了 (E\_OK) またはエラーコード

### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し 【NGKI2163】<br>・CPUロック状態からの呼出し 【NGKI2164】 |
| E_ID    | 不正ID番号<br>・spnidが有効範囲外 【NGKI2165】                                       |
| E_NOEXS | オブジェクト未登録<br>・対象スピントロックが未登録 【NGKI2166】                                  |
| E_OACV  | オブジェクトアクセス違反<br>・対象スピントロックに対する管理操作が許可されていない [P] 【NGKI2167】              |
| E_OBJ   | オブジェクト状態エラー<br>・対象スピントロックは静的APIで生成された 【NGKI2168】                        |

### 【機能】

spnidで指定したスピントロック（対象スピントロック）を削除する。具体的な振舞いは以下の通り。

対象スピントロックの登録が解除され、そのスピントロックIDが未使用の状態に戻される 【NGKI2169】。

### 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、del\_spnをサポートしない 【FMPS0141】。

### 【未決定事項】

対象スピントロックが取得されている状態の場合の振舞いは、今後の課題である。

|          |                            |
|----------|----------------------------|
| loc_spn  | スピントロックの取得 [TM] 【NGKI2170】 |
| iloc_spn | スピントロックの取得 [IM] 【NGKI2171】 |

### 【C言語API】

```
ER ercd = loc_spn(ID spnid)
ER ercd = iloc_spn(ID spnid)
```

### 【パラメータ】

|    |       |                |
|----|-------|----------------|
| ID | spnid | 対象スピントロックのID番号 |
|----|-------|----------------|

### 【リターンパラメータ】

ER

ercd

正常終了 (E\_OK) またはエラーコード

### 【エラーコード】

E\_CTX

コンテキストエラー

- ・非タスクコンテキストからの呼出し (loc\_spnの場合) 【NGKI2172】
- ・タスクコンテキストからの呼出し (iloc\_spnの場合) 【NGKI2173】
- ・CPUロック状態からの呼出し 【NGKI2174】

E\_ID

不正ID番号

- ・spnidが有効範囲外 【NGKI2175】

E\_NOEXS

オブジェクト未登録

- ・対象スピノロックが未登録 [D] 【NGKI2176】

E\_OACV

オブジェクトアクセス違反

- ・対象スピノロックに対する通常操作1が許可されていない  
(loc\_spnの場合) [P] 【NGKI2177】

### 【機能】

spnidで指定したスピノロック（対象スピノロック）を取得する。具体的な振舞いは以下の通り。

対象スピノロックが取得されていない状態である場合には、プロセッサ間ロックの取得を試みる【NGKI2178】。ロックが他のプロセッサによって取得されている状態である場合や、他のプロセッサがロックの取得に成功した場合には、ロックが返却されるまでループによって待ち、返却されたらロックの取得を試みる【NGKI2179】。これを、ロックの取得に成功するまで繰り返す【NGKI2180】。

ロックの取得に成功した場合には、スピノロックは取得されている状態になる【NGKI2181】。また、CPUロックフラグをセットしてCPUロック状態へ遷移し、サービスコールからリターンする【NGKI2182】。

なお、複数のプロセッサがロックの取得を待っている時に、どのプロセッサが最初にロックを取得できるかは、現時点ではターゲット定義とする【NGKI2183】。

### 【補足説明】

対象スピノロックが、loc\_spn/iloc\_spnを呼び出したプロセッサによって取得されている状態である場合には、スピノロックの取得によりCPUロック状態になつてゐるため、loc\_spn/iloc\_spnはE\_CTXエラーとなる。

プロセッサがロックを取得できる順序を、現時点ではターゲット定義としたが、リアルタイム性保証のためには、（ロックの取得待ちの間に割込みが発生しない限りは）loc\_spn/iloc\_spnを呼び出した順序でロックを取得できるとするのが望ましい。ただし、ターゲットハードウェアの制限で、そのような実装ができるとは限らないため、現時点ではターゲット定義としている。

|          |                                  |
|----------|----------------------------------|
| try_spn  | スピンロックの取得（ポーリング） [TM] 【NGKI2184】 |
| ityr_spn | スピンロックの取得（ポーリング） [IM] 【NGKI2185】 |

### 【C言語API】

```
ER ercd = try_spn(ID spnid)
ER ercd = itry_spn(ID spnid)
```

### 【パラメータ】

|    |       |               |
|----|-------|---------------|
| ID | spnid | 対象スピンロックのID番号 |
|----|-------|---------------|

### 【リターンパラメータ】

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

### 【エラーコード】

|         |              |  |
|---------|--------------|--|
| E_CTX   | コンテキストエラー    | ・非タスクコンテキストからの呼出し (try_spnの場合) 【NGKI2186】                  |
|         |              | ・タスクコンテキストからの呼出し (ityr_spnの場合) 【NGKI2187】                  |
|         |              | ・CPUロック状態からの呼出し 【NGKI2188】                                 |
| E_ID    | 不正ID番号       | ・spnidが有効範囲外 【NGKI2189】                                    |
| E_NOEXS | オブジェクト未登録    | ・対象スピンロックが未登録 [D] 【NGKI2190】                               |
| E_OACV  | オブジェクトアクセス違反 | ・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<br>(try_spnの場合) [P] 【NGKI2191】 |
| E_OBJ   | オブジェクト状態エラー  | ・条件については機能の項を参照  |

### 【機能】

spnidで指定したスピンロック（対象スピンロック）の取得を試みる。具体的な振舞いは以下の通り。

対象スピンロックが取得されていない状態である場合には、プロセッサ間ロックの取得を試みる【NGKI2192】。ロックの取得に成功した場合には、スピンロックは取得されている状態になる【NGKI2193】。また、CPUロックフラグをセットしてCPUロック状態へ遷移し、サービスコールからリターンする【NGKI2194】。

対象スピンロックが他のプロセッサによって取得されている状態である場合や、ロックの取得に失敗した場合（他のプロセッサがロックの取得に成功した場合）には、E\_OBJ

エラーとする【NGKI2195】.

#### 【使用上の注意】

try\_spn / itry\_spnを、ロックの取得に成功するまで繰り返し呼び出すことによりスピンロックを取得する方法は、loc\_spn / iloc\_spnによりスピンロックを取得する方法と、プロセッサがロックを取得できる順序が異なる可能性ある。

|          |                           |
|----------|---------------------------|
| unl_spn  | スピンロックの返却 [TM] 【NGKI2196】 |
| iunl_spn | スピンロックの返却 [IM] 【NGKI2197】 |

#### 【C言語API】

```
ER ercd = unl_spn(ID spnid)
ER ercd = iunl_spn(ID spnid)
```

#### 【パラメータ】

|    |       |               |
|----|-------|---------------|
| ID | spnid | 対象スピンロックのID番号 |
|----|-------|---------------|

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し (unl_spnの場合) 【NGKI2198】<br>・タスクコンテキストからの呼出し (iunl_spnの場合) 【NGKI2199】 |
| E_ID    | 不正ID番号<br>・spnidが有効範囲外 【NGKI2200】   |
| E_NOEXS | オブジェクト未登録<br>・対象スピンロックが未登録 [D] 【NGKI2201】   |
| E_OACV  | オブジェクトアクセス違反<br>・対象スピンロックに対する通常操作1が許可されていない<br>(unl_spnの場合) [P] 【NGKI2202】                          |
| E_ILUSE | サービスコール不正使用<br>・条件については機能の項を参照  |

#### 【機能】

spnidで指定したスピンロック（対象スピンロック）を返却する。具体的な振舞いは以下の通り。

対象スピンロックが、unl\_spn/iunl\_spnを呼び出したプロセッサによって取得されている状態である場合には、ロックを返却し、スピンロックを取得されていない状態とする【NGKI2203】。また、CPUロックフラグをクリアし、CPUロック解除状態へ遷移する【NGKI2204】。

対象スピンロックが、取得されていない状態である場合や、他のプロセッサによって取得されている状態である場合には、E\_ILUSEエラーとなる【NGKI2205】。

[[API\_ref\_spn]

ref\_spn      スピンロックの状態参照 [TM] 【NGKI2206】

【C言語API】

```
ER ercd = ref_spn(ID spnid, T_RSPN *pk_rspn)
```

【パラメータ】

|          |         |                               |
|----------|---------|-------------------------------|
| ID       | spnid   | 対象スピンロックのID番号                 |
| T_RSPN * | pk_rspn | スピンロックの現在状態を入れるパケットへの<br>ポインタ |

【リターンパラメータ】

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

\*スピンロックの現在状態 (パケットの内容)

STAT      spnstat      ロック状態

【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI2207】<br>・CPUロック状態からの呼出し【NGKI2208】 |
| E_ID    | 不正ID番号<br>・spnidが有効範囲外【NGKI2209】                                      |
| E_NOEXS | オブジェクト未登録<br>・対象スピントラックが未登録[D]【NGKI2210】                              |
| E_OACV  | オブジェクトアクセス違反<br>・対象スピントラックに対する参照操作が許可されていない[P]<br>【NGKI2211】          |
| E_MACV  | メモリアクセス違反<br>・pk_rspnが指すメモリ領域への書き込みアクセスが許可されて<br>いない) [P]【NGKI2212】   |

#### 【機能】

spnidで指定したスピントラック（対象スピントラック）の現在状態を参照する。参考照した現在状態は、  
pk\_rspnで指定したパケットに返される【NGKI2213】。

spnstatには、対象スピントラックの現在のロック状態を表す次のいずれかの値が返される【NGKI2214】。

|          |       |            |
|----------|-------|------------|
| TSPN_UNL | 0x01U | 取得されていない状態 |
| TSPN_LOC | 0x02U | 取得されている状態  |

#### 【使用上の注意】

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

## 1.5. メモリプール管理機能

#### 【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、メモリプール管理機能をサポートしない【SSPS0128】。

#### 【μITRON4.0仕様との関係】

この仕様では、可変長メモリプール機能はサポートしないこととした。

#### 【仕様決定の理由】

可変長メモリプール機能をサポートしないこととしたのは、メモリ割付けの処理時間とフラグメンテーションの発生を考えると、最適なメモリ管理アルゴリズムはアプリケーション依存となるため、カーネル内で実現するより、ライブラリとして実現する方が適切と考えたためである。

### 1.5.1. 固定長メモリプール

固定長メモリプールは、生成時に決めたサイズのメモリブロック（固定長メモリブロック）を動的に獲得・返却するための同期・通信オブジェクトである。

固定長メモリプールは、固定長メモリプールIDと呼ぶID番号で識別する【NGKI2215】。

各固定長メモリプールが持つ情報は次の通り【NGKI2216】。

- 固定長メモリプール属性
- 待ち行列（固定長メモリブロックの獲得待ち状態のタスクのキュー）
- 固定長メモリプール領域
- 固定長メモリプール管理領域
- アクセス許可ベクタ（保護機能対応カーネルの場合）
- 属する保護ドメイン（保護機能対応カーネルの場合）
- 属するクラス（マルチプロセッサ対応カーネルの場合）

待ち行列は、固定長メモリブロックが獲得できるまで待っている状態（固定長メモリブロックの獲得待ち状態）のタスクが、固定長メモリブロックを獲得できる順序でつながれているキューである。

固定長メモリプール領域は、その中から固定長メモリブロックを割り付けるためのメモリ領域である。

固定長メモリプール管理領域は、固定長メモリプール領域中の割当て済みの固定長メモリブロックと未割当てのメモリ領域に関する情報を格納しておくためのメモリ領域である。

保護機能対応カーネルにおいて、固定長メモリプール管理領域は、カーネルの用いるオブジェクト管理領域として扱われる【NGKI2217】。

固定長メモリプール属性には、次の属性を指定することができる【NGKI2218】。

TA\_TPRI 0x01U 待ち行列をタスクの優先度順にする

TA\_TPRIを指定しない場合、待ち行列はFIFO順になる【NGKI2219】。

固定長メモリプール機能に関連するカーネル構成マクロは次の通り。

TNUM\_MPFI 登録できる固定長メモリプールの数（動的生成対応でないカーネルでは、静的APIによって登録された固定長メモリプールの数に一致）【NGKI2220】

#### 【μITRON4.0仕様との関係】

固定長メモリプール領域として確保すべき領域のサイズを返すカーネル構成マクロ（TSZ\_MPFI）は廃止した。これは、固定長メモリプール領域をアプリケーションで確保する方法を定めた結果、そのサイズは $(blkcnt * ROUND_MPFI(blksz))$

で求めることができるようになったためである。

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

|           |                              |
|-----------|------------------------------|
| CRE_MPFI  | 固定長メモリプールの生成 [S] 【NGKI2221】  |
| acre_mpfi | 固定長メモリプールの生成 [TD] 【NGKI2222】 |

#### 【静的API】

```
CRE_MPFI ID mpfid, { ATR mpfatr, uint_t blkcnt, uint_t blksz,  
                         MPF_T *mpf, void *mpfmb }
```

#### 【C言語API】

```
ER_ID mpfid = acre_mpfi(const T_CMPF *pk_cmpf)
```

#### 【パラメータ】

|          |         |   |
|----------|---------|---|
| ID       | mpfid   | 生成する固定長メモリプールのID番号 (CRE_MPFIの場合)        |
| T_CMPF * | pk_cmpf | 固定長メモリプールの生成情報を入ったパケットへのポインタ (静的APIを除く) |

\* 固定長メモリプールの生成情報 (パケットの内容)

|         |        |                       |
|---------|--------|-----------------------|
| ATR     | mpfatr | 固定長メモリプール属性           |
| uint_t  | blkcnt | 獲得できる固定長メモリブロックの数     |
| uint_t  | blksz  | 固定長メモリブロックのサイズ (バイト数) |
| MPF_T * | mpf    | 固定長メモリプール領域の先頭番地      |
| void *  | mpfmb  | 固定長メモリプール管理領域の先頭番地    |

#### 【リターンパラメータ】

|       |       |                                     |
|-------|-------|-------------------------------------|
| ER_ID | mpfid | 生成された固定長メモリプールのID番号 (正の値) またはエラーコード |
|-------|-------|-------------------------------------|

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し〔s〕【NGKI2223】<br>・CPUロック状態からの呼出し〔s〕【NGKI2224】   |
| E_RSATR | 予約属性<br>・mpfatrが無効【NGKI2225】<br>・属する保護ドメインの指定が有効範囲外〔sP〕【NGKI2226】<br>・属するクラスの指定が有効範囲外〔sM〕【NGKI2227】<br>・クラスの団みの中に記述されていない〔sM〕【NGKI2228】 |
| E_NOSPT | 未サポート機能<br>・条件については各カーネルにおける規定の項を参照   |
| E_PAR   | パラメータエラー<br>・blkcntが0以下【NGKI2229】<br>・blkszが0以下【NGKI2230】<br>・その他の条件については機能の項を参照  |
| E_OACV  | オブジェクトアクセス違反<br>・システム状態に対する管理操作が許可されていない〔sP〕<br>【NGKI2231】  |
| E_MACV  | メモリアクセス違反<br>・pk_cmpfが指すメモリ領域への読み出しが許可されて<br>いない〔sP〕【NGKI2232】  |
| E_NOID  | ID番号不足<br>・割り付けられる固定長メモリプールIDがない〔sD〕【NGKI2233】  |
| E_NOMEM | メモリ不足<br>・固定長メモリプール領域が確保できない【NGKI2234】<br>・固定長メモリプール管理領域が確保できない【NGKI2235】   |
| E_OBJ   | オブジェクト状態エラー<br>・mpfidで指定した固定長メモリプールが登録済み(CRE_MP<br>Fの場合)【NGKI2236】<br>・その他の条件については機能の項を参照   |

## 【機能】

各パラメータで指定した固定長メモリプール生成情報に従って、固定長メモリプールを生成する。mpf, mpfmbとblkcnt, blkszから固定長メモリプール領域が、blkcntから固定長メモリプール管理領域がそれぞれ設定され、メモリプール領域全体が未割当ての状態に初期化される【NGKI2237】。また、待ち行列は空の状態に初期化される【NGKI2238】。

静的APIにおいては、mpfidはオブジェクト識別名、mpfatr, blkcnt, blkszは整数定数パラメータ、mpfとmpfmbは一般定数パラメータである【NGKI2239】。コンフィギュレータは、静的APIのメモリ不足(E\_NOMEM)エラーを検出することができない【NGKI2240】。

mpfをNULLとした場合、blkcntとblkszから決まるサイズの固定長メモリプール領域が、コンフィギュレータまたはカーネルにより確保される【NGKI2241】。

保護機能対応カーネルでは、コンフィギュレータまたはカーネルにより確保さ

れる固定長メモリプール領域は、固定長メモリプールと同じ保護ドメインに属し、固定長メモリプールと同じアクセス許可ベクタを持ったメモリオブジェクト中に確保される【NGKI2242】。

mpfmbをNULLとした場合、blkcntから決まるサイズの固定長メモリプール管理領域が、コンフィギュレータまたはカーネルにより確保される【NGKI2243】。

〔mpfにNULL以外を指定した場合〕

mpfにNULL以外を指定した場合、mpfを先頭番地とする固定長メモリプール領域は、アプリケーションで確保しておく必要がある【NGKI2244】。固定長メモリプール領域をアプリケーションで確保するために、次のデータ型とマクロを用意している【NGKI2245】。

MPF\_T

固定長メモリプール領域を確保するためのデータ型

COUNT\_MPFT(blksz) 固定長メモリブロックのサイズがblkszの固定長メモリプール領域を確保するために、固定長メモリブロック1つあたりに必要なMPF\_T型の配列の要素数  
ROUND\_MPFT(blksz) 要素数COUNT\_MPFT(blksz)のMPF\_T型の配列のサイズ（blkszを、MPF\_T型のサイズの倍数になるように大きい方に丸めた値）

これらを用いて固定長メモリプール領域を確保する方法は次の通り【NGKI2246】。

MPF\_T <固定長メモリプール領域の変数名>[(blkcnt) \* COUNT\_MPFT(blksz)];

この時、mpfには<固定長メモリプール領域の変数名>を指定する【NGKI2247】。

これ以外の方法で固定長メモリプール領域を確保する場合には、上記の配列と同じサイズのメモリ領域を確保しなければならない【NGKI2248】。また、その先頭番地がターゲット定義の制約に合致していないなければならない。mpfにターゲット定義の制約に合致しない先頭番地を指定した時には、E\_PARエラーとなる【NGKI2249】。

保護機能対応カーネルでは、アプリケーションで確保する固定長メモリプール領域は、カーネルに登録されたメモリオブジェクトに含まれていなければならない。指定した固定長メモリプール領域が、カーネルに登録されたメモリオブジェクトに含まれていない場合、E\_OBJエラーとなる【NGKI2251】。

〔mpfmbにNULL以外を指定した場合〕

mpfmbにNULL以外を指定した場合、mpfmbを先頭番地とする固定長メモリプール管理領域は、アプリケーションで確保しておく必要がある【NGKI2252】。固定長メモリプール管理領域をアプリケーションで確保するために、次のマクロを用意している【NGKI2253】。

|                   |  |
|-------------------|--|
| TSZ_MPMB(blkcnt)  | blkcntで指定した数の固定長メモリブロックを管理することができる固定長メモリプール管理領域のサイズ（バイト数）              |
| TCNT_MPMB(blkcnt) | blkcntで指定した数の固定長メモリブロックを管理することができる固定長メモリプール管理領域を確保するために必要なMB_T型の配列の要素数 |

これらを用いて固定長メモリプール管理領域を確保する方法は次の通り 【NGKI2254】。

MB\_T <固定長メモリプール管理領域の変数名>[TCNT\_MPMB(blkcnt)];

この時、mpfmbには<固定長メモリプール管理領域の変数名>を指定する 【NGKI2255】。

この方法に従わず、mpfmbにターゲット定義の制約に合致しない先頭番地を指定E\_PARエラーとなる【NGKI2256】。また、保護機能対応カーネルにmpfmbで指定した固定長メモリプール管理領域がカーネル専用のメモリオブジェクトに含まれない場合、E\_OBJエラーとなる【NGKI2257】。

した時には、  
おいて、

#### 【補足説明】

保護機能対応カーネルにおいて、固定長メモリプール領域をアプリケーションで確保する場合には、固定長メモリプール領域が属する保護ドメインとアクセス権の設定は変更されない。これらを適切に設定することは、アプリケーションの責任である。

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、CRE\_MPFIのみをサポートする【ASPS0164】。また、mpfmbにはNULLのみを指定することができる。NULL以外を指定した場合には、E\_NOSPTエラーとなる【ASPS0166】。ただし、動的生成機能拡張パッケージでは、acre\_mpfもサポートする【ASPS0167】。acre\_mpfに対しては、mpfmbにNULL以外を指定できないという制限はない【ASPS0168】。

#### 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、CRE\_MPFIのみをサポートする【FMPS0142】。また、mpfmbにはNULLのみを指定することができる。NULL以外を指定した場合には、E\_NOSPTエラーとなる【FMPS0144】。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、CRE\_MPFIのみをサポートする【HRPS0136】。また、mpfmbにNULLのみを指定することができる。NULL以外を指定した場合には、E\_NOSPTエラーとなる【HRPS0138】。

動的生成機能拡張パッケージでは、acre\_mpfもサポートする【HRPS0197】。acre\_mpfに対しては、mpfmbにNULL以外を指定できないという制限はない【HRPS0198】。ただし、mpfmbに

NULLが指定されるとカーネルが固定長メモリプール領域を確保する機能はサポートしない。mpfにNULLを指定した場合には、E\_NOSPTエラーとなる【HRPS0199】。

#### 【μITRON4.0仕様との関係】

mpfのデータ型をMPF\_T \*に変更した。COUNT\_MPFTとROUND\_MPFTを新設し、固定長メモリプール領域をアプリケーションで確保する方法を規定した。また、μITRON4.0/PX仕様にあわせて、固定長メモリプール生成情報に、mpfmbを追加した。

#### 【μITRON4.0/PX仕様との関係】

TCNT\_MPFBを新設し、固定長メモリプール管理領域をアプリケーションで確保する方法を規定した。

AID\_MPFT 割付け可能な固定長メモリプールIDの数〔SD〕 【NGKI2258】

#### 【静的API】

AID\_MPFT(uint\_t nompf)

#### 【パラメータ】

uint\_t nompf 割付け可能な固定長メモリプールIDの数

#### 【エラーコード】

|         |                                    |
|---------|------------------------------------|
| E_RSATR | 予約属性                               |
|         | ・保護ドメインの囲みの中に記述されている〔P〕 【NGKI3436】 |
|         | ・クラスの囲みの中に記述されていない〔M〕 【NGKI2259】   |
| E_PAR   | パラメータエラー                           |
|         | ・nompfが負の値 【NGKI3284】              |

#### 【機能】

nompfで指定した数の固定長メモリプールIDを、固定長メモリプールを生成するサービスコールによって割付け可能な固定長メモリプールIDとして確保する【NGKI2260】。

nompfは整数定数式パラメータである【NGKI2261】。

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルの動的生成機能拡張パッケージでは、AID\_MPFTをサポートする【ASPS0216】。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルの動的生成機能拡張パッケージでは、AID\_MPFIをサポートする【HRPS0217】。

|          |                                       |
|----------|---------------------------------------|
| SAC_MPFI | 固定長メモリプールのアクセス許可ベクタの設定〔SP〕【NGKI2262】  |
| sac_mpfi | 固定長メモリプールのアクセス許可ベクタの設定〔TPD〕【NGKI2263】 |

#### 【静的API】

```
SAC_MPFI(ID mpfid, { ACPTN acptn1, ACPTN acptn2,
                        ACPTN acptn3, ACPTN acptn4 })
```

#### 【C言語API】

```
ER ercd = sac_mpfi(ID mpfid, const ACVCT *p_acvct)
```

#### 【パラメータ】

|         |         |                                   |
|---------|---------|-----------------------------------|
| ID      | mpfid   | 対象固定長メモリプールのID番号                  |
| ACVCT * | p_acvct | アクセス許可ベクタを入れたパケットへのポインタ（静的APIを除く） |

\* アクセス許可ベクタ（パケットの内容）

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

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し [s] 【NGKI2264】<br>・CPUロック状態からの呼出し [s] 【NGKI2265】   |
| E_ID    | 不正ID番号<br>・mpfidが有効範囲外 [s] 【NGKI2266】   |
| E_RSATR | 予約属性<br>・対象固定長メモリプールが属する保護ドメインの囲みの中<br>(対象固定長メモリプールが無所属の場合は、保護ドメインの囲みの外)に記述されていない [S] 【NGKI2267】<br>・対象固定長メモリプールが属するクラスの囲みの中に記述<br>されていない [SM] 【NGKI2268】 |
| E_NOEXS | オブジェクト未登録<br>・対象固定長メモリプールが未登録 【NGKI2269】  |
| E_OACV  | オブジェクトアクセス違反<br>・対象固定長メモリプールに対する管理操作が許可されてい<br>ない [s] 【NGKI2270】  |
| E_MACV  | メモリアクセス違反<br>・p_acvctが指すメモリ領域への読み出しが許可されて<br>いない [s] 【NGKI2271】   |
| E_OBJ   | オブジェクト状態エラー<br>・対象固定長メモリプールは静的APIで生成された [s] 【NGKI2272】<br>・対象固定長メモリプールに対してアクセス許可ベクタが設<br>定済み [S] 【NGKI2273】   |

## 【機能】

mpfidで指定した固定長メモリプール（対象固定長メモリプール）のアクセス許可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 NGKI2274】。対象固定長メモリプールの固定長メモリプール領域がコンフィギュレータまたはカーネルにより確保されたものである場合には、固定長メモリプール領域のアクセス許可ベクタも、各パラメータで指定した値に 設定する【NGKI2275】。

静的APIにおいては、mpfidはオブジェクト識別名、acptn1～acptn4は整数定数式パラメータである【NGKI2276】。

## 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、SAC\_MPFIのみをサポートする【HRPS0139】。ただし、動的生成機能拡張パッケージでは、sac\_mpfもサポートする【HRPS0200】。

del\_mpf 固定長メモリプールの削除 [TD] 【NGKI2277】

## 【C言語API】

```
ER ercd = del_mpf(ID mpfid)
```

#### 【パラメータ】

|    |       |                  |
|----|-------|------------------|
| ID | mpfid | 対象固定長メモリプールのID番号 |
|----|-------|------------------|

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼び出し 【NGKI2278】<br>・CPUロック状態からの呼び出し 【NGKI2279】 |
| E_ID    | 不正ID番号<br>・mpfidが有効範囲外 【NGKI2280】   |
| E_NOEXS | オブジェクト未登録<br>・対象固定長メモリプールが未登録 【NGKI2281】                                  |
| E_OACV  | オブジェクトアクセス違反<br>・対象固定長メモリプールに対する管理操作が許可されていない [P] 【NGKI2282】              |
| E_OBJ   | オブジェクト状態エラー<br>・対象固定長メモリプールは静的APIで生成された 【NGKI2283】                        |

#### 【機能】

mpfidで指定した固定長メモリプール（対象固定長メモリプール）を削除する。  
具体的な振舞いは以下の通り。

対象固定長メモリプールの登録が解除され、その固定長メモリプールIDが未使用の状態に戻される【NGKI2284】。また、対象固定長メモリプールの待ち行列につながれたタスクは、待ち行列の先頭のタスクから順に待ち解除される【NGKI2285】。待ち解除されたタスクには、待ち状態となったサービスコールエラーが返る【NGKI2286】。

からE\_DLT

#### 【使用上の注意】

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_mpf`をサポートしない【ASPS0170】。ただし、動的生成機能拡張パッケージでは、`del_mpf`をサポートする【ASPS0171】。

#### 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、`del_mpf`をサポートしない【FMP0146】。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、`del_mpf`をサポートしない【HRPS0140】。ただし、動的生成機能拡張パッケージでは、`del_mpf`をサポートする【HRPS0201】。

|                       |                                       |
|-----------------------|---------------------------------------|
| <code>get_mpf</code>  | 固定長メモリブロックの獲得 [T] 【NGKI2287】          |
| <code>pget_mpf</code> | 固定長メモリブロックの獲得（ポーリング）[T] 【NGKI2288】    |
| <code>tget_mpf</code> | 固定長メモリブロックの獲得（タイムアウト付き）[T] 【NGKI2289】 |

#### 【C言語API】

```
ER ercd = get_mpf(ID mpfid, void **p_blk)
ER ercd = pget_mpf(ID mpfid, void **p_blk)
ER ercd = tget_mpf(ID mpfid, void **p_blk, TMO tmout)
```

#### 【パラメータ】

|         |       |                                    |
|---------|-------|------------------------------------|
| ID      | mpfid | 対象固定長メモリプールのID番号                   |
| void ** | p_blk | 獲得した固定長メモリブロックの先頭番地を入れるメモリ領域へのポインタ |
| TMO     | tmout | タイムアウト時間 (twai_mpfの場合)             |

#### 【リターンパラメータ】

|        |      |                       |
|--------|------|-----------------------|
| ER     | ercd | 正常終了 (E_OK) またはエラーコード |
| void * | blk  | 獲得した固定長メモリブロックの先頭番地   |

#### 【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI2290】<br>・CPUロック状態からの呼出し【NGKI2291】<br>・ディスパッチ保留状態からの呼出し( pget_mpfを除く)<br>【NGKI2292】 |
| E_NOSPT | 未サポート機能<br>・制約タスクからの呼出し( pget_mpfを除く) 【NGKI2293】   |
| E_ID    | 不正ID番号<br>・mpfidが有効範囲外【NGKI2294】   |
| E_PAR   | パラメータエラー<br>・tmoutが無効(tget_mpfの場合) 【NGKI2295】  |
| E_NOEXS | オブジェクト未登録<br>・対象固定長メモリプールが未登録[D] 【NGKI2296】  |
| E_OACV  | オブジェクトアクセス違反<br>・対象固定長メモリプールに対する通常操作1が許可されていない[P] 【NGKI2297】   |
| E_MACV  | メモリアクセス違反<br>・p_blkが指すメモリ領域への書き込みアクセスが許可されていない[P] 【NGKI2298】   |
| E_TMOUT | ポーリング失敗またはタイムアウト( get_mpfを除く) 【NGKI2299】   |
| E_RLWAI | 待ち禁止状態または待ち状態の強制解除( pget_mpfを除く)<br>【NGKI2300】   |
| 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)<br>【NGKI2301】   |

## 【機能】

mpfidで指定した固定長メモリプール(対象固定長メモリプール)から固定長メモリブロックを獲得し、その先頭番地をp\_blkが指すメモリ領域に返す。具体的な振舞いは以下の通り。

対象固定長メモリプールの固定長メモリプール領域の中に、固定長メモリブロックを割り付けることのできる未割当てのメモリ領域がある場合には、固定長メモリブロックが1つ割り付けられ、その先頭番地がblkに返される【NGKI2302】。

未割当てのメモリ領域がない場合には、自タスクは固定長メモリプールの獲得待ち状態となり、対象固定長メモリプールの待ち行列につながれる【NGKI2303】。

rel\_mpf 固定長メモリブロックの返却[T] 【NGKI2304】

## 【C言語API】

```
ER ercd = rel_mpf(ID mpfid, void *blk)
```

## 【パラメータ】

|              |              |   |
|--------------|--------------|---|
| ID<br>void * | mpfid<br>blk | 対象固定長メモリプールのID番号<br>返却する固定長メモリブロックの先頭番地 |
|--------------|--------------|---|

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼び出し 【NGKI2305】<br>・CPUロック状態からの呼び出し 【NGKI2306】 |
| E_ID    | 不正ID番号<br>・mpfidが有効範囲外 【NGKI2307】   |
| E_PAR   | パラメータエラー<br>・条件については機能の項を参照   |
| E_NOEXS | オブジェクト未登録<br>・対象固定長メモリプールが未登録 [D] 【NGKI2308】                              |
| E_OACV  | オブジェクトアクセス違反<br>・対象固定長メモリプールに対する通常操作2が許可されていない [P] 【NGKI2309】             |

#### 【機能】

mpfidで指定した固定長メモリプール（対象固定長メモリプール）に、blkで指定した固定長メモリブロックを返却する。具体的な振舞いは以下の通り。

対象固定長メモリプールの待ち行列にタスクが存在する場合には、待ち行列の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【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ド【NGKI2311】。

待ち行列にタスクが存在しない場合には、blkで指定した固定長メモリブロックは、対象固定長メモリプールのメモリプール領域に返却される【NGKI2312】。

blkが、対象固定長メモリプールから獲得した固定長メモリブロックの先頭番地E\_PARエラーとなる【NGKI2313】。

先頭のタスクが、  
る【  
ルからE\_OKが返る【

ini\_mpf 固定長メモリプールの再初期化 [T] 【NGKI2314】

#### 【C言語API】

```
ER ercd = ini_mpf(ID mpfid)
```

#### 【パラメータ】

|    |       |                  |
|----|-------|------------------|
| ID | mpfid | 対象固定長メモリプールのID番号 |
|----|-------|------------------|

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼び出し 【NGKI2315】<br>・CPUロック状態からの呼び出し 【NGKI2316】 |
| E_ID    | 不正ID番号<br>・mpfidが有効範囲外 【NGKI2317】   |
| E_NOEXS | オブジェクト未登録<br>・対象固定長メモリプールが未登録 [D] 【NGKI2318】                              |
| E_OACV  | オブジェクトアクセス違反<br>・対象固定長メモリプールに対する管理操作が許可されていない [P] 【NGKI2319】              |

#### 【機能】

mpfidで指定した固定長メモリプール（対象固定長メモリプール）を再初期化する。具体的な振舞いは以下の通り。

対象固定長メモリプールのメモリプール領域全体が未割当ての状態に初期化される【NGKI2320】。また、対象固定長メモリプールの待ち行列につながれたタスクは、待ち行列の先頭のタスクから順に待ち解除される【NGKI2321】。待ち解除されたタスクには、待ち状態となったサービスコールからE\_DLTエラーが返る【NGKI2322】。

#### 【使用上の注意】

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。

固定長メモリプールを再初期化した場合に、アプリケーションとの整合性を保つのは、アプリケーションの責任である。

#### 【μITRON4.0仕様との関係】

$\mu$ ITRON4.0仕様に定義されていないサービスコールである。

**ref\_mpf 固定長メモリプールの状態参照 [T] 【NGKI2323】**

#### 【C言語API】

```
ER ercd = ref_mpf(ID mpfid, T_RMPF *pk_rmpf)
```

#### 【パラメータ】

|          |         |                              |
|----------|---------|------------------------------|
| ID       | mpfid   | 対象固定長メモリプールのID番号             |
| T_RMPF * | pk_rmpf | 固定長メモリプールの現在状態を入れるパケットへのポインタ |

#### 【リターンパラメータ】

ER ercd 正常終了 (E\_OK) またはエラーコード

\*固定長メモリプールの現在状態 (パケットの内容)

|        |         |   |
|--------|---------|---|
| ID     | wtskid  | 固定長メモリプールの待ち行列の先頭のタスクのID番号                  |
| uint_t | fblkcnt | 固定長メモリプール領域の空きメモリ領域に割り付けることができる固定長メモリブロックの数 |

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー   |
|         | ・非タスクコンテキストからの呼び出し 【NGKI2324】                       |
|         | ・CPUロック状態からの呼び出し 【NGKI2325】                         |
| E_ID    | 不正ID番号  |
|         | ・mpfidが有効範囲外 【NGKI2326】                             |
| E_NOEXS | オブジェクト未登録   |
|         | ・対象固定長メモリプールが未登録 [D] 【NGKI2327】                     |
| E_OACV  | オブジェクトアクセス違反  |
|         | ・対象固定長メモリプールに対する参照操作が許可されていない [P] 【NGKI2328】        |
| E_MACV  | メモリアクセス違反   |
|         | ・pk_rmpf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] 【NGKI2329】 |

#### 【機能】

mpfidで指定した固定長メモリプール（対象固定長メモリプール）の現在状態を参照する。参照した現在状態は、pk\_rmpfで指定したパケットに返される【NGKI2330】。

対象固定長メモリプールの待ち行列にタスクが存在しない場合、wtskidには TSK\_NONE (=0) が返る【NGKI2331】。

#### 【使用上の注意】

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

## 1.6. 時間管理機能

### 1.6.1. システム時刻管理

システム時刻は、カーネルによって管理され、タイムアウト処理、タスクの遅延、周期ハンドラの起動、アラームハンドラの起動に使用される時刻を管理するカーネルオブジェクトである【NGKI3603】。システム時刻は、符号無しの整数型である SYSTIM 型で表され、単位はミリ秒である【NGKI2332】。

システム時刻は、カーネルの初期化時に 0 に初期化される【NGKI2333】。タイムティックを通知するためのタイマ割込みが発生する毎にカーネルによって更新され、SYSTIM 型で表せる最大値 (ULONG\_MAX) を超えると 0 に戻される【NGKI2334】。タイムティックの周期は、ターゲット定義である【NGKI2335】。また、システム時刻の精度はターゲットに依存する【NGKI2336】。

マルチプロセッサ対応でないカーネルと、マルチプロセッサ対応カーネルでグローバルタイマ方式を用いている場合には、システム時刻は、システムに 1 つの NGKI2337】。マルチプロセッサ対応カーネルでローカルタイマ方式を用いている場合には、システム時刻は、プロセッサ毎に存在する NGKI2338】。ローカルタイマ方式とグローバルタイマ方式については、「2.3.4 マルチプロセッサ対応」の節を参照すること。

マルチプロセッサ対応カーネルでローカルタイマ方式を用いている場合には、タイムアウト処理とタスクの遅延処理には、待ち解除されるタスクが割り付けられているプロセッサのシステム時刻が用いられる【NGKI2339】。また、周期ハンドラとアラームハンドラの起動には、それが割り付けられているプロセッサのシステム時刻が用いられる【NGKI2340】。これらの処理単位がマイグレーションする場合には、用いられるシステム時刻も変更される【NGKI2341】。この場合にも、イベントの処理が行われるのは、基準時刻から相対時間によって指定した以上の時間が経過した後となるという規則は維持される【NGKI2342】。

1 回のタイムティックの発生により、複数のイベントの処理を行うべき状況になつた場合、それらの処理の間の処理順序は規定されない【NGKI2343】。

性能評価用システム時刻は、性能評価に使用することを目的とした、システム

時刻よりも精度の高い時刻である。性能評価用システム時刻は、符号なしの整数型である  
SYSUTM型で表され、単位はマイクロ秒である【NGKI2344】。ただし、  
実際の精度はターゲットに依存する【NGKI2345】。

マルチプロセッサ対応カーネルにおける性能評価用システム時刻の扱いは、ターゲット定義とする【NGKI2346】。

システム時刻管理機能に関するカーネル構成マクロは次の通り。

TIC\_NUME タイムティックの周期（単位はミリ秒）の分子 【NGKI2347】  
TIC\_DENO タイムティックの周期（単位はミリ秒）の分母

TOPPERS\_SUPPORT\_GET\_UTM get\_utmがサポートされている【NGKI2348】

#### 【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、時間管理機能をサポートしない【SSPS0129】。

#### 【使用上の注意】

タイムティックを通知するためのタイマ割込みが長時間マスクされた場合（タイマ割込みより優先して実行される割込み処理が長時間続けて実行された場合を含む）や、シミュレーション環境においてシミュレータのプロセスが長時間スケジュールされなかった場合には、システム時刻が正しく更新されない可能性があるため、注意が必要である。

#### 【μITRON4.0仕様との関係】

システム時刻を設定するサービスコール（set\_tim）を廃止した。また、タイムティックを供給する機能は、カーネル内に実現することとし、そのためのサービスコール（isig\_tim）は廃止した。

#### 【μITRON4.0/PX仕様との関係】

システム時刻のアクセス許可ベクタは廃止し、システム状態のアクセス許可ベクタで代替することとした。そのため、システム時刻のアクセス許可ベクタをSAC\_TIM）とサービスコール（sac\_tim）は廃止した。

get\_tim システム時刻の参照 [T] 【NGKI2349】

#### 【C言語API】

```
ER ercd = get_tim(SYSTIM *p_systim)
```

## 【パラメータ】

SYSTIM \* p\_systim システム時刻を入れるメモリ領域へのポインタ

## 【リターンパラメータ】

|        |        |                       |
|--------|--------|-----------------------|
| ER     | ercd   | 正常終了 (E_OK) またはエラーコード |
| SYSTIM | systim | システム時刻の現在値            |

## 【エラーコード】

|        |   |
|--------|---|
| E_CTX  | コンテキストエラー<br>・非タスクコンテキストからの呼出し 【NGKI2350】<br>・CPUロック状態からの呼出し 【NGKI2351】 |
| E_OACV | オブジェクトアクセス違反<br>・システム状態に対する参照操作が許可されていない [P]<br>【NGKI2352】              |
| E_MACV | メモリアクセス違反<br>・p_systimが指すメモリ領域への書き込みアクセスが許可され<br>ていない) [P] 【NGKI2353】   |

## 【機能】

システム時刻の現在値を参照する。参照したシステム時刻は、p\_systimが指すメモリ領域に返される 【NGKI2354】。

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、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 【NGKI2355】。

## 【補足説明】

マルチプロセッサ対応カーネルでローカルタイマ方式を用いている場合に、他のプロセッサのシステム時刻の現在値を参照する機能は用意していない。

get\_utm 性能評価用システム時刻の参照 [TI] 【NGKI2356】

## 【C言語API】

```
ER ercd = get_utm(SYSUTM *p_sysutm)
```

## 【パラメータ】

SYSUTM \* p\_sysutm 性能評価用システム時刻を入れるメモリ領域へのポインタ

#### 【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 |
| SYSUTM | sysutm | 性能評価用システム時刻の現在値       |

#### 【エラーコード】

|         |   |
|---------|---|
| E_NOSPT | 未サポート機能<br>・条件については機能の項を参照                                      |
| E_MACV  | メモリアクセス違反<br>・p_sysutmが指すメモリ領域へ書込みアクセスが許可されていない) [P] 【NGKI2357】 |

#### 【機能】

性能評価用システム時刻の現在値を参照する。参照した性能評価用システム時刻は、  
p\_sysutmが指すメモリ領域に返される【NGKI2358】。

get\_utmは、任意の状態から呼び出すことができる【NGKI2359】。タスクコンテキストからも非タスクコンテキストからも呼び出すことができるし、CPUロック状態であっても呼び出すことができる。

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】。  
get\_utmがサポートされている場合には、TOPPERS\_SUPPORT\_GET\_UTMがマクロ定義される【  
NGKI2361】。サポートされていない場合にget\_utmを呼び出すと、  
E\_NOSPTエラーが返るか、リンク時にエラーとなる【NGKI2362】。

#### 【使用方法】

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。  
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。

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限らない。

#### 【使用上の注意】

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ない。

get\_utmは、任意の状態から呼び出すことができるよう、全割込みロック状態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込みの応答性が低下する。

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考慮に入れて使用しなければならない。

#### 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである。

### 1.6.2. 周期ハンドラ

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。

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】。

各周期ハンドラが持つ情報は次の通り【NGKI2364】。

- 周期ハンドラ属性
- 周期ハンドラの動作状態
- 次に周期ハンドラを起動する時刻
- 拡張情報
- 周期ハンドラの先頭番地
- 起動周期
- 起動位相
- アクセス許可ベクタ（保護機能対応カーネルの場合）
- 属する保護ドメイン（保護機能対応カーネルの場合）
- 属するクラス（マルチプロセッサ対応カーネルの場合）

周期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  
NGKI2365】。

対時間後である【

$$\text{起動位相} + \text{起動周期} \times (n-1) \quad n = 1, 2, \dots$$

周期ハンドラの動作状態は、動作している状態と動作していない状態のいずれかをとる【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始、動作していない状態にすることを動作停止という。

かをとる【

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】。

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インは、カーネルドメインに限られる【NGKI2369】。

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】。

|        |       |                          |
|--------|-------|--------------------------|
| 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 |
| TA_PHS | 0x04U | 周期ハンドラを生成した時刻を基準時刻とする    |

TA\_STAを指定しない場合、周期ハンドラの生成直後には、周期ハンドラは動作していない状態となる【NGKI2371】。

TA\_PHSを指定しない場合には、周期ハンドラを動作開始した時刻が、周期ハン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、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時刻）が、基準時刻となる【NGKI2373】。

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、周期ハンドラの起動処理時に設定される【NGKI2374】。

マルチプロセッサ対応カーネルでグローバルタイマ方式を用いている場合には、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、周期ハンドラは、システム時刻管理プロセッサによって実行される。

C言語による周期ハンドラの記述形式は次の通り【NGKI2376】。

```
void cyclic_handler(intptr_t exinf)
{
    周期ハンドラ本体
}
```

exinfには、周期ハンドラの拡張情報が渡される【NGKI2377】。

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。

|            |  |
|------------|--|
| 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）【NGKI2378】 |
|------------|--|

#### 【TOPPERS/ASPカーネルにおける規定】

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】。

#### 【TOPPERS/FMPカーネルにおける規定】

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】。

#### 【TOPPERS/HRP2カーネルにおける規定】

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】。

#### 【μITRON4.0仕様との関係】

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

|          |                           |
|----------|---------------------------|
| CRE_CYC  | 周期ハンドラの生成 [S] 【NGKI2379】  |
| acre_cyc | 周期ハンドラの生成 [TD] 【NGKI2380】 |

#### 【静的API】

```
CRE_CYC(ID cycid, { ATR cycatr, intptr_t exinf, CYCHDR cychdr,
                      RELTIM cyctim, RELTIM cycphs })
```

#### 【C言語API】

```
ER_ID cycid = acre_cyc(const T_CCYC *pk_ccyc)
```

#### 【パラメータ】

|          |         |  |
|----------|---------|--|
| ID       | cycid   | 生成する周期ハンドラのID番号 (CRE_CYCの場合)             |
| T_CCYC * | pk_ccyc | 周期ハンドラの生成情報を入ったパケットへの<br>ポインタ (静的APIを除く) |

\* 周期ハンドラの生成情報 (パケットの内容)

|          |        |             |
|----------|--------|-------------|
| ATR      | cycatr | 周期ハンドラ属性    |
| intptr_t | exinf  | 周期ハンドラの拡張情報 |
| CYCHDR   | cychdr | 周期ハンドラの先頭番地 |
| RELTIM   | cyctim | 周期ハンドラの起動周期 |
| RELTIM   | cycphs | 周期ハンドラの起動位相 |

#### 【リターンパラメータ】

|       |       |                                      |
|-------|-------|--------------------------------------|
| ER_ID | cycid | 生成された周期ハンドラのID番号 (正の値) または<br>エラーコード |
|-------|-------|--------------------------------------|

#### 【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー <ul style="list-style-type: none"> <li>・非タスクコンテキストからの呼出し [s] 【NGKI2381】</li> <li>・CPUロック状態からの呼出し [s] 【NGKI2382】</li> </ul>   |
| E_RSATR | 予約属性 <ul style="list-style-type: none"> <li>・cycatrが無効 【NGKI2383】</li> <li>・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】</li> <li>・カーネルドメインの囲みの中に記述されていない [SP] 【NGKI2385】</li> <li>・属するクラスの指定が有効範囲外 [sM] 【NGKI2386】</li> <li>・クラスの囲みの中に記述されていない [SM] 【NGKI2387】</li> <li>・その他の条件については機能の項を参照</li> </ul> |
| E_PAR   | パラメータエラー <ul style="list-style-type: none"> <li>・cychdrがプログラムの先頭番地として正しくない 【NGKI2388】</li> <li>・cyctimが有効範囲（0より大きくTMAX_RELTIM以下）外 【NGKI2397】</li> <li>・cycphsが有効範囲（0以上TMAX_RELTIM以下）外 【NGKI2399】</li> </ul>  |
| E_OACV  | オブジェクトアクセス違反 <ul style="list-style-type: none"> <li>・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】</li> </ul>  |
| E_MACV  | メモリアクセス違反 <ul style="list-style-type: none"> <li>・pk_ccycが指すメモリ領域への読み出しが許可されていない [sP] 【NGKI2390】</li> </ul>  |
| E_NOID  | ID番号不足 <ul style="list-style-type: none"> <li>・割り付けられる周期ハンドラIDがない [sD] 【NGKI2391】</li> </ul>   |
| E_OBJ   | オブジェクト状態エラー <ul style="list-style-type: none"> <li>・cycidで指定した周期ハンドラが登録済み（CRE_CYCの場合）【NGKI2392】</li> </ul>   |

## 【機能】

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。

cycatrにTA\_STAを指定した場合、対象周期ハンドラは動作している状態となる【NGKI2393】。次に周期ハンドラを起動する時刻は、サービスコールを呼び出した時刻（静的APIの場合はカーネルの起動時刻）から、cycphsで指定した相対時間後に設定される【NGKI2394】。cycphsにcyctimより大きい値を指定してもよい【NGKI2400】。

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初期化される【NGKI2395】。

静的APIにおいては、cycidはオブジェクト識別名、cycatr、cyctim、cycphsは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【NGKI2396】。

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、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】。

## 【補足説明】

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。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、コンフィギュレータが警告メッセージを出力する。

## 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、CRE\_CYCのみをサポートする【ASPS0173】。ただし、TA\_PHS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ジでは、acre\_cycもサポートする【ASPS0175】。

## 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、CRE\_CYCのみをサポートする【FMPS0148】。ただし、TA\_PHS属性の周期ハンドラはサポートしない【FMPS0149】。

## 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、CRE\_CYCのみをサポートする【HRPS0142】。ただし、TA\_PHS属性の周期ハンドラはサポートしない【HRPS0143】。動的生成機能拡張パッケージでは、acre\_cycもサポートする【HRPS0202】。

## 【μITRON4.0仕様との関係】

cychdrのデータ型をCYCHDRに変更した。また、cycphsにcyctimより大きい値を指定した場合の振舞いと、静的APIでcycphsに0を指定した場合の振舞いを規定した。

AID\_CYC 割付け可能な周期ハンドラIDの数の指定〔SD〕 【NGK12402】

## 【静的API】

AID\_CYC(uint\_t nocyc)

## 【パラメータ】

uint\_t nocyc 割付け可能な周期ハンドラIDの数

## 【エラーコード】

|         |   |
|---------|---|
| E_RSATR | 予約属性<br>・保護ドメインの囲みの中に記述されている〔P〕【NGKI3437】<br>・クラスの囲みの中に記述されていない〔M〕【NGKI2404】<br>・その他の条件については機能の項を参照 |
| E_PAR   | パラメータエラー<br>・nocycが負の値【NGKI3285】  |

### 【機能】

nocycで指定した数の周期ハンドラIDを、周期ハンドラを生成するサービスコールによって割付け可能な周期ハンドラIDとして確保する【NGKI2405】。

nocycは整数定数式パラメータである【NGKI2406】。

マルチプロセッサ対応カーネルでグローバルタイマ方式を用いている場合で、AID\_CYCが属するクラスの割付け可能プロセッサが、システム時刻管理プロセッサのみでない場合には、E\_RSATRエラーとなる【NGKI2407】。

### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルの動的生成機能拡張パッケージでは、AID\_CYCをサポートする【ASPS0217】。

### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルの動的生成機能拡張パッケージでは、AID\_CYCをサポートする【HRPS0218】。

|         |                                    |
|---------|------------------------------------|
| SAC_CYC | 周期ハンドラのアクセス許可ベクタの設定〔SP〕【NGKI2408】  |
| sac_cyc | 周期ハンドラのアクセス許可ベクタの設定〔TPD〕【NGKI2409】 |

### 【静的API】

```
SAC_CYC(ID cycid, { ACPTN acptn1, ACPTN acptn2,
                      ACPTN acptn3, ACPTN acptn4 })
```

### 【C言語API】

```
ER ercd = sac_cyc(ID cycid, const ACVCT *p_acvct)
```

### 【パラメータ】

|         |         |                                   |
|---------|---------|-----------------------------------|
| ID      | cycid   | 対象周期ハンドラのID番号                     |
| ACVCT * | p_acvct | アクセス許可ベクタを入れたパケットへのポインタ（静的APIを除く） |

\* アクセス許可ベクタ（パケットの内容）

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

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー                                      |
|         | ・非タスクコンテキストからの呼び出し [s] 【NGKI2410】              |
|         | ・CPUロック状態からの呼び出し [s] 【NGKI2411】                |
| E_ID    | 不正ID番号   |
|         | ・cycidが有効範囲外 [s] 【NGKI2412】                    |
| E_RSATR | 予約属性   |
|         | ・カーネルドメインの団みの中に記述されていない [S] 【NGKI2413】         |
|         | ・対象周期ハンドラが属するクラスの団みの中に記述されていない [SM] 【NGKI2414】 |
| E_NOEXS | オブジェクト未登録                                      |
|         | ・対象周期ハンドラが未登録 【NGKI2415】                       |
| E_OACV  | オブジェクトアクセス違反                                   |
|         | ・対象周期ハンドラに対する管理操作が許可されていない [s] 【NGKI2416】      |
| E_MACV  | メモリアクセス違反                                      |
|         | ・p_acvctが指すメモリ領域への読み出しが許可されていない [s] 【NGKI2417】 |
| E_OBJ   | オブジェクト状態エラー                                    |
|         | ・対象周期ハンドラは静的APIで生成された [s] 【NGKI2418】           |
|         | ・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 [s] 【NGKI2419】     |

#### 【機能】

cycidで指定した周期ハンドラ（対象周期ハンドラ）のアクセス許可ベクタ（4つのアクセス許可パターンの組）を、各パラメータで指定した値に設定する 【NGKI2420】。

静的APIにおいては、cycidはオブジェクト識別名、acptn1～acptn4は整数定数式パラメータである 【NGKI2421】。

## 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、SAC\_CYCのみをサポートする【HRPS0144】。ただし、動的生成機能拡張パッケージでは、sac\_cycもサポートする【HRPS0203】。

del\_cyc      周期ハンドラの削除 [TD] 【NGKI2422】

### 【C言語API】

```
ER ercd = del_cyc(ID cycid)
```

### 【パラメータ】

|    |       |               |
|----|-------|---------------|
| ID | cycid | 対象周期ハンドラの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び出し 【NGKI2423】             |
|         | ・CPUロック状態からの呼び出し 【NGKI2424】               |
| E_ID    | 不正ID番号                                    |
|         | ・cycidが有効範囲外 【NGKI2425】                   |
| E_NOEXS | オブジェクト未登録                                 |
|         | ・対象周期ハンドラが未登録 【NGKI2426】                  |
| E_OACV  | オブジェクトアクセス違反                              |
|         | ・対象周期ハンドラに対する管理操作が許可されていない [P] 【NGKI2427】 |
| E_OBJ   | オブジェクト状態エラー                               |
|         | ・対象周期ハンドラは静的APIで生成された 【NGKI2428】          |

### 【機能】

cycidで指定した周期ハンドラ（対象周期ハンドラ）を削除する。具体的な振舞いは以下の通り。

対象周期ハンドラの登録が解除され、その周期ハンドラIDが未使用の状態に戻る【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、動作していない状態にされた後に、登録が解除される【NGKI2430】。

## 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、`del_cyc`をサポートしない【ASPS0177】。ただし、動的生成機能拡張パッケージでは、`del_cyc`をサポートする【ASPS0178】。

#### 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、`del_cyc`をサポートしない【FMPS0151】。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、`del_cyc`をサポートしない【HRPS0145】。ただし、動的生成機能拡張パッケージでは、`del_cyc`をサポートする【HRPS0204】。

`sta_cyc`      周期ハンドラの動作開始 [T] 【NGKI2431】

#### 【C言語API】

```
ER ercd = sta_cyc(ID cycid)
```

#### 【パラメータ】

ID            cycid      対象周期ハンドラのID番号

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>· 非タスクコンテキストからの呼び出し 【NGKI2432】<br>· CPUロック状態からの呼び出し 【NGKI2433】 |
| E_ID    | 不正ID番号<br>· cycidが有効範囲外 【NGKI2434】  |
| E_NOEXS | オブジェクト未登録<br>· 対象周期ハンドラが未登録 [D] 【NGKI2435】                                  |
| E_OACV  | オブジェクトアクセス違反<br>· 対象周期ハンドラに対する通常操作1が許可されていない [P]<br>【NGKI2436】              |

#### 【機能】

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。具体的な振舞いは以下の通り。

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ない【NGKI2437】。次に周期ハンドラを起動する時刻は、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】。

ている状態となる【

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時刻の再設定のみが行われる【NGKI2439】。

#### 【補足説明】

TA\_PHS属性でない周期ハンドラの場合、次に周期ハンドラを起動する時刻は、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 に設定される。

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、次に周期ハンドラを起動する時刻は変化しない。

#### 【μITRON4.0仕様との関係】

TA\_PHS属性でない周期ハンドラにおいて、sta\_cycを呼び出した後、最初に周期ハンドラが起動される時刻を変更した。μITRON4.0仕様では、sta\_cycを呼び出してから周期ハンドラの起動周期で指定した相対時間後となっているが、この仕様では、起動位相で指定した相対時間後とした。

msta\_cyc 割付けプロセッサ指定での周期ハンドラの動作開始 [TM] 【NGKI2440】

#### 【C言語API】

```
ER ercd = msta_cyc(ID cycid, ID prcid)
```

#### 【パラメータ】

|    |       |                         |
|----|-------|-------------------------|
| ID | cycid | 対象周期ハンドラのID番号           |
| ID | prcid | 周期ハンドラの割付け対象のプロセッサのID番号 |

#### 【リターンパラメータ】

ER ercd 正常終了 (E\_OK) またはエラーコード

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI2441】<br>・CPUロック状態からの呼出し【NGKI2442】 |
| E_NOSPT | 未サポート機能<br>・条件については機能の項を参照  |
| E_ID    | 不正ID番号<br>・cycidが有効範囲外【NGKI2443】<br>・prcidが有効範囲外【NGKI2444】            |
| E_PAR   | パラメータエラー<br>・条件については機能の項を参照   |
| E_NOEXS | オブジェクト未登録<br>・対象周期ハンドラが未登録〔D〕【NGKI2445】                               |
| E_OACV  | オブジェクトアクセス違反<br>・対象周期ハンドラに対する通常操作1が許可されていない〔P〕<br>【NGKI2446】          |

### 【機能】

prcidで指定したプロセッサを割付けプロセッサとして、 cycidで指定した周期ハンドラ（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。

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  
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  
NGKI2447】。次に周期ハンドラを起動する時刻は、  
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】。

プロセッサが  
動作している状態となる【

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  
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  
動する時刻の再設定が行われる【NGKI2449】。

プロセッサが

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、  
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  
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  
NGKI2451】。

る時からである【

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  
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】。

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  
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】。

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【NGKI2454】。

### 【補足説明】

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、  
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 後に設定される。

## 【使用上の注意】

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0の場合に、注意が必要である。

周期ハンドラの起動位相が

## 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである。

stp\_cyc      周期ハンドラの動作停止 [T] 【NGKI2455】

## 【C言語API】

```
ER ercd = stp_cyc(ID cycid)
```

## 【パラメータ】

|    |       |               |
|----|-------|---------------|
| ID | cycid | 対象周期ハンドラのID番号 |
|----|-------|---------------|

## 【リターンパラメータ】

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

## 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼び出し 【NGKI2456】<br>・CPUロック状態からの呼び出し 【NGKI2457】 |
| E_ID    | 不正ID番号<br>・cycidが有効範囲外 【NGKI2458】   |
| E_NOEXS | オブジェクト未登録<br>・対象周期ハンドラが未登録 [D] 【NGKI2459】                                 |
| E_OACV  | オブジェクトアクセス違反<br>・対象周期ハンドラに対する通常操作2が許可されていない [P]<br>【NGKI2460】             |

## 【機能】

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。具体的な振舞いは以下の通り。

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

【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

ずに正常終了する【

NGKI2462】.

ref\_cyc 周期ハンドラの状態参照 [T] 【NGKI2463】

【C言語API】

```
ER ercd = ref_cyc(ID cycid, T_RCYC *pk_rcyc)
```

【パラメータ】

|          |         |                               |
|----------|---------|-------------------------------|
| ID       | cycid   | 対象周期ハンドラのID番号                 |
| 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<br>ポインタ |

【リターンパラメータ】

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

\*周期ハンドラの現在状態 (パケットの内容)

|        |         |  |
|--------|---------|--|
| STAT   | cycstat | 周期ハンドラの動作状態                            |
| RELTIM | lefttim | 次に周期ハンドラを起動する時刻までの相対時間                 |
| ID     | prcid   | 周期ハンドラの割付けプロセッサのID (マルチプロセッサ対応カーネルの場合) |

【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び出し 【NGKI2464】<br>・CPUロック状態からの呼び出し 【NGKI2465】 |
| E_ID    | 不正ID番号       | ・cycidが有効範囲外 【NGKI2466】                                      |
| E_NOEXS | オブジェクト未登録    | ・対象周期ハンドラが未登録 [D] 【NGKI2467】                                 |
| E_OACV  | オブジェクトアクセス違反 | ・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]<br>【NGKI2468】                 |
| E_MACV  | メモリアクセス違反    | ・pk_rcycが指すメモリ領域への書き込みアクセスが許可されて<br>いない) [P] 【NGKI2469】      |

【機能】

cycidで指定した周期ハンドラ (対象周期ハンドラ) の現在状態を参照する。参

照した現在状態は、

pk\_rcycで指定したパケットに返される【NGKI2470】.

cycstatには、対象周期ハンドラの現在の動作状態を表す次のいずれかの値が返される【NGKI2471】.

|          |       |                  |
|----------|-------|------------------|
| TCYC_STP | 0x01U | 周期ハンドラが動作していない状態 |
| TCYC_STA | 0x02U | 周期ハンドラが動作している状態  |

対象周期ハンドラが動作している状態である場合には、lefttimに、次に周期ハンドラ起動する時刻までの相対時間が返される【NGKI2472】. 対象周期ハンドラが動作していない状態である場合には、lefttimの値は保証されない【NGKI2473】.

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番号が返される【NGKI2474】.

セッサのID

#### 【使用上の注意】

ref\_cycはデバッグ時向けの機能であり、その他の目的に使用することは推奨しない. これは、ref\_cycを呼び出し、対象周期ハンドラの現在状態を参照した直後に割込みが発生した場合、ref\_cycから戻ってきた時には対象周期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.

#### 【μITRON4.0仕様との関係】

TCYC\_STPとTCYC\_STAを値を変更した.

### 1.6.3. アラームハンドラ

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番号によって識別する【NGKI2475】.

各アラームハンドラが持つ情報は次の通り【NGKI2476】.

- ・アラームハンドラ属性
- ・アラームハンドラの動作状態
- ・アラームハンドラを起動する時刻
- ・拡張情報
- ・アラームハンドラの先頭番地
- ・アクセス許可ベクタ（保護機能対応カーネルの場合）
- ・属する保護ドメイン（保護機能対応カーネルの場合）
- ・属するクラス（マルチプロセッサ対応カーネルの場合）

アラームハンドラの動作状態は、動作している状態と動作していない状態のいずれかをとる【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.

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定される【NGKI2478】.

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る時刻になると、アラームハンドラの起動処理が行われる【NGKI2479】。具体的には、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】。その後に、拡張情報をパラメータとして、アラームハンドラが呼び出される【NGKI2481】。

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ドメインは、カーネルドメインに限られる【NGKI2482】。

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、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。

アラームハンドラ属性に指定できる属性はない【NGKI3423】。そのためアラームハンドラ属性には、TA\_NULLを指定しなければならない【NGKI3424】。

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】。

```
void alarm_handler(intptr_t exinf)
{
    アラームハンドラ本体
}
```

exinfには、アラームハンドラの拡張情報が渡される【NGKI2485】。

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。

|            |  |
|------------|--|
| 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）【NGKI2486】 |
|------------|--|

#### 【μITRON4.0仕様との関係】

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

|          |                             |
|----------|-----------------------------|
| CRE_ALM  | アラームハンドラの生成 [S] 【NGKI2487】  |
| acre_alm | アラームハンドラの生成 [TD] 【NGKI2488】 |

#### 【静的API】

```
CRE_ALM(ID almid, { ATR almatr, intptr_t exinf, ALMHDR almhdr })
```

#### 【C言語API】

```
ER_ID almId = acre_alm(const T_CALM *pk_calm)
```

#### 【パラメータ】

|          |         |                                       |
|----------|---------|---------------------------------------|
| ID       | almid   | 生成するアラームハンドラのID番号（CRE_ALMの場合）         |
| T_CALM * | pk_calm | アラームハンドラの生成情報を入ったパケットへのポインタ（静的APIを除く） |

\*アラームハンドラの生成情報（パケットの内容）

|          |        |               |
|----------|--------|---------------|
| ATR      | almatr | アラームハンドラ属性    |
| intptr_t | exinf  | アラームハンドラの拡張情報 |
| ALMHDR   | almhdr | アラームハンドラの先頭番地 |

#### 【リターンパラメータ】

|       |       |                                      |
|-------|-------|--------------------------------------|
| ER_ID | almid | 生成されたアラームハンドラのID番号（正の値）<br>またはエラーコード |
|-------|-------|--------------------------------------|

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し [s] 【NGKI2489】<br>・CPUロック状態からの呼出し [s] 【NGKI2490】   |
| E_RSATR | 予約属性<br>・almatrが無効 【NGKI2491】<br>・属する保護ドメインの指定が有効範囲外またはカーネルドメイン以外 [sP] 【NGKI2492】<br>・カーネルドメインの囲みの中に記述されていない [SP] 【NGKI2493】<br>・属するクラスの指定が有効範囲外 [sM] 【NGKI2494】<br>・クラスの囲みの中に記述されていない [SM] 【NGKI2495】<br>・その他の条件については機能の項を参照 |
| E_PAR   | パラメータエラー<br>・almhdrがプログラムの先頭番地として正しくない 【NGKI2496】   |
| E_OACV  | オブジェクトアクセス違反<br>・システム状態に対する管理操作が許可されていない [sP] 【NGKI2497】  |
| E_MACV  | メモリアクセス違反<br>・pk_calmが指すメモリ領域への読み出しが許可されていない [sP] 【NGKI2498】  |
| E_NOID  | ID番号不足<br>・割り付けられるアラームハンドラIDがない [sD] 【NGKI2499】   |
| E_OBJ   | オブジェクト状態エラー<br>・almidで指定したアラームハンドラが登録済み (CRE_ALMの場合) 【NGKI2500】   |

## 【機能】

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【NGKI2501】。

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タ、exinfとalmhdrは一般定数式パラメータである【NGKI2502】。

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、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】。

## 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、CRE\_ALMのみをサポートする【ASPS0179】。ただし、動的生成機能拡張パッケージでは、acre\_almもサポートする【ASPS0180】。

## 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、CRE\_ALMのみをサポートする【FMPS0152】。

## 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、CRE\_ALMのみをサポートする【HRPS0146】。ただし、動的生成機能拡張パッケージでは、acre\_almもサポートする【HRPS0205】。

## 【μITRON4.0仕様との関係】

almhdrのデータ型をALMHDRに変更した。

AID\_ALM 割付け可能なアラームハンドラIDの数の指定 [SD] 【NGKI2504】

## 【静的API】

AID\_ALM(uint\_t noalm)

## 【パラメータ】

uint\_t noalm 割付け可能なアラームハンドラIDの数

## 【エラーコード】

E\_RSATR 予約属性

- ・保護ドメインの囲みの中に記述されている [P] 【NGKI3438】
- ・クラスの囲みの中に記述されていない [M] 【NGKI2506】
- ・その他の条件については機能の項を参照

E\_PAR パラメータエラー

- ・noalmが負の値 【NGKI3286】

## 【機能】

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【NGKI2507】。

noalmは整数定数式パラメータである【NGKI2508】。

マルチプロセッサ対応カーネルでグローバルタイマ方式を用いている場合で、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サのみでない場合には、E\_RSATRエラーとなる【NGKI2509】。

## 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルの動的生成機能拡張パッケージでは、AID\_ALMをサポートする【ASPS0218】。

## 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルの動的生成機能拡張パッケージでは、AID\_ALMをサポートする【HRPS0219】。

|         |                                      |
|---------|--------------------------------------|
| 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】  |
| sac_alm | アラームハンドラのアクセス許可ベクタの設定〔TPD〕【NGKI2511】 |

#### 【静的API】

```
SAC_ALM(ID almId, { ACPTN acptn1, ACPTN acptn2,
                      ACPTN acptn3, ACPTN acptn4 })
```

#### 【C言語API】

```
ER ercd = sac_alm(ID almId, const ACVCT *p_acvct)
```

#### 【パラメータ】

|         |         |                                   |
|---------|---------|-----------------------------------|
| ID      | almid   | 対象アラームハンドラのID番号                   |
| ACVCT * | p_acvct | アクセス許可ベクタを入れたパケットへのポインタ（静的APIを除く） |

##### \* アクセス許可ベクタ（パケットの内容）

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

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し [s] 【NGKI2512】<br>・CPUロック状態からの呼出し [s] 【NGKI2513】                       |
| E_ID    | 不正ID番号<br>・almidが有効範囲外 [s] 【NGKI2514】   |
| E_RSATR | 予約属性<br>・カーネルドメインの囲みの中に記述されていない [S] 【NGKI2515】<br>・対象アラームハンドラが属するクラスの囲みの中に記述されていない [SM] 【NGKI2516】    |
| E_NOEXS | オブジェクト未登録<br>・対象アラームハンドラが未登録 【NGKI2517】   |
| E_OACV  | オブジェクトアクセス違反<br>・対象アラームハンドラに対する管理操作が許可されていない [s] 【NGKI2518】   |
| E_MACV  | メモリアクセス違反<br>・p_acvctが指すメモリ領域への読み出しが許可されていない [s] 【NGKI2519】   |
| E_OBJ   | オブジェクト状態エラー<br>・対象アラームハンドラは静的APIで生成された [s] 【NGKI2520】<br>・対象アラームハンドラに対してアクセス許可ベクタが設定済み [S] 【NGKI2521】 |

## 【機能】

almidで指定したアラームハンドラ（対象アラームハンドラ）のアクセス許可ベクタ（4つのアクセス許可パターンの組）を、各パラメータで指定した値に設定する【NGKI2522】。

静的APIにおいては、almidはオブジェクト識別名、acptn1～acptn4は整数定数式パラメータである【NGKI2523】。

## 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、SAC\_ALMのみをサポートする【HRPS0147】。ただし、動的生成機能拡張パッケージでは、sac\_almもサポートする【HRPS0206】。

**del\_alm** アラームハンドラの削除 [TD] 【NGKI2524】

## 【C言語API】

```
ER ercd = del_alm(ID almid)
```

## 【パラメータ】

|    |       |                 |
|----|-------|-----------------|
| ID | almid | 対象アラームハンドラのID番号 |
|----|-------|-----------------|

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼び出し 【NGKI2525】<br>・CPUロック状態からの呼び出し 【NGKI2526】 |
| E_ID    | 不正ID番号<br>・almidが有効範囲外 【NGKI2527】   |
| E_NOEXS | オブジェクト未登録<br>・対象アラームハンドラが未登録 【NGKI2528】                                   |
| E_OACV  | オブジェクトアクセス違反<br>・対象アラームハンドラに対する管理操作が許可されていない [P] 【NGKI2529】               |
| E_OBJ   | オブジェクト状態エラー<br>・対象アラームハンドラは静的APIで生成された 【NGKI2530】                         |

#### 【機能】

almidで指定したアラームハンドラ（対象アラームハンドラ）を削除する。具体的な振舞いは以下の通り。

対象アラームハンドラの登録が解除され、そのアラームハンドラIDが未使用の状態に戻される【NGKI2531】。対象アラームハンドラが動作している状態であった場合には、登録解除の前に、アラームハンドラが動作していない状態となる【NGKI2532】。

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、del\_almをサポートしない【ASPS0182】。ただし、動的生成機能拡張パッケージでは、del\_almをサポートする【ASPS0183】。

#### 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、del\_almをサポートしない【FMP0154】。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、del\_almをサポートしない【HRPS0148】。ただし、動的生成機能拡張パッケージでは、del\_almをサポートする【HRPS0207】。

sta\_alm アラームハンドラの動作開始 [T] 【NGKI2533】  
ista\_alm アラームハンドラの動作開始 [I] 【NGKI2534】

#### 【C言語API】

```
ER ercd = sta_alm(ID almid, RELTIM almtim)
ER ercd = ista_alm(ID almid, RELTIM almtim)
```

#### 【パラメータ】

|        |        |                     |
|--------|--------|---------------------|
| ID     | almid  | 対象アラームハンドラのID番号     |
| RELTIM | almtim | アラームハンドラの起動時刻（相対時間） |

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |              |   |
|---------|--------------|---|
| E_CTX   | コンテキストエラー    | ・非タスクコンテキストからの呼び出し (sta_almの場合) 【NGKI2535】                |
|         |              | ・タスクコンテキストからの呼び出し (ista_almの場合) 【NGKI2536】                |
|         |              | ・CPUロック状態からの呼び出し  |
| E_ID    | 不正ID番号       | ・almidが有効範囲外 【NGKI2537】                                   |
| E_PAR   | パラメータエラー     | ・almtimがTMAX_RELTIMより大きい 【NGKI2538】                       |
| E_NOEXS | オブジェクト未登録    | ・対象アラームハンドラが未登録 [D] 【NGKI2539】                            |
| E_OACV  | オブジェクトアクセス違反 | ・対象アラームハンドラに対する通常操作1が許可されていない (sta_almの場合) [P] 【NGKI2540】 |

#### 【機能】

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。

対象アラームハンドラが動作していない状態であれば、対象アラームハンドラ  
は動作している状態となる【NGKI2541】。アラームハンドラを起動する時刻は、  
sta\_almを呼び出してから、almtimで指定した相対時間後に設定される【NGKI2542】。

対象アラームハンドラが動作している状態であれば、アラームハンドラを起動

する時刻の再設定のみが行われる【NGKI2543】。

|           |  |
|-----------|--|
| 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】 |
| 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】 |

#### 【C言語API】

```
ER ercd = msta_alm(ID almid, RELTIM almtim, ID prcid)
ER ercd = imsta_alm(ID almid, RELTIM almtim, ID prcid)
```

#### 【パラメータ】

|        |        |                               |
|--------|--------|-------------------------------|
| ID     | almid  | 対象アラームハンドラのID番号               |
| RELTIM | almtim | アラームハンドラの起動時刻（相対時間）           |
| ID     | prcid  | アラームハンドラの割付け対象のプロセッサの<br>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    | <ul style="list-style-type: none"><li>・非タスクコンテキストからの呼び出し（msta_almの場合）<br/>【NGKI2546】</li><li>・タスクコンテキストからの呼び出し（imsta_almの場合） 【NGKI2547】</li><li>・CPUロック状態からの呼び出し 【NGKI2548】</li></ul> |
| E_NOSPT | 未サポート機能      | <ul style="list-style-type: none"><li>・条件については機能の項を参照</li></ul>   |
| E_ID    | 不正ID番号       | <ul style="list-style-type: none"><li>・almidが有効範囲外 【NGKI2549】</li><li>・prcidが有効範囲外 【NGKI2550】</li></ul>   |
| E_PAR   | パラメータエラー     | <ul style="list-style-type: none"><li>・almtimがTMAX_RELTIMより大きい 【NGKI2551】</li><li>・その他の条件については機能の項を参照</li></ul>   |
| E_NOEXS | オブジェクト未登録    | <ul style="list-style-type: none"><li>・対象アラームハンドラが未登録〔D〕 【NGKI2552】</li></ul>   |
| E_OACV  | オブジェクトアクセス違反 | <ul style="list-style-type: none"><li>・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】</li></ul>  |

#### 【機能】

prcidで指定したプロセッサを割付けプロセッサとして、 almidで指定したアラームハンドラ（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。

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】。

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】。

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】。

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】。

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】。

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】。

#### 【使用上の注意】

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。

#### 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである。

|          |                              |
|----------|------------------------------|
| stp_alm  | アラームハンドラの動作停止 [T] 【NGKI2562】 |
| istp_alm | アラームハンドラの動作停止 [I] 【NGKI2563】 |

#### 【C言語API】

|                              |
|------------------------------|
| ER ercd = stp_alm(ID almid)  |
| ER ercd = istp_alm(ID almid) |

#### 【パラメータ】

|    |       |                 |
|----|-------|-----------------|
| ID | almid | 対象アラームハンドラの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 (stp_almの場合) 【NGKI2564】                 |
|         | ・タスクコンテキストからの呼出し (istp_almの場合) 【NGKI2565】                 |
|         | ・CPUロック状態からの呼出し 【NGKI2566】                                |
| E_ID    | 不正ID番号  |
|         | ・almidが有効範囲外 【NGKI2567】                                   |
| E_NOEXS | オブジェクト未登録   |
|         | ・対象アラームハンドラが未登録 [D] 【NGKI2568】                            |
| E_OACV  | オブジェクトアクセス違反  |
|         | ・対象アラームハンドラに対する通常操作2が許可されていない (stp_almの場合) [P] 【NGKI2569】 |

#### 【機能】

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。  
具体的な振舞いは以下の通り。

対象アラームハンドラが動作している状態であれば、動作していない状態となる【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、何も行われずに正常終了する【NGKI2571】。

|         |                              |
|---------|------------------------------|
| ref_alm | アラームハンドラの状態参照 [T] 【NGKI2572】 |
|---------|------------------------------|

#### 【C言語API】

```
ER ercd = ref_alm(ID almid, T_RALM *pk_ralm)
```

#### 【パラメータ】

|          |         |                             |
|----------|---------|-----------------------------|
| ID       | almid   | 対象アラームハンドラのID番号             |
| T_RALM * | pk_ralm | アラームハンドラの現在状態を入れるパケットへのポインタ |

#### 【リターンパラメータ】

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

\*アラームハンドラの現在状態 (パケットの内容)

|        |         |  |
|--------|---------|--|
| STAT   | almstat | アラームハンドラの動作状態                            |
| RELTIM | lefttim | アラームハンドラを起動する時刻までの相対時間                   |
| ID     | prcid   | 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) |

## 【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 【NGKI2573】                       |
|         | ・CPUロック状態からの呼出し 【NGKI2574】                         |
| E_ID    | 不正ID番号   |
|         | ・almidが有効範囲外 【NGKI2575】                            |
| E_NOEXS | オブジェクト未登録  |
|         | ・対象アラームハンドラが未登録 [D] 【NGKI2576】                     |
| E_OACV  | オブジェクトアクセス違反                                       |
|         | ・対象アラームハンドラに対する参照操作が許可されていない [P] 【NGKI2577】        |
| E_MACV  | メモリアクセス違反  |
|         | ・pk_ralmが指すメモリ領域への書き込みアクセスが許可されていない [P] 【NGKI2578】 |

## 【機能】

almidで指定したアラームハンドラ (対象アラームハンドラ) の現在状態を参照する。参照した現在状態は、pk\_ralmで指定したパケットに返される【NGKI2579】。

almstatには、対象アラームハンドラの現在の動作状態を表す次のいずれかの値が返される【NGKI2580】。

|          |       |                    |
|----------|-------|--------------------|
| TALM_STP | 0x01U | アラームハンドラが動作していない状態 |
| TALM_STA | 0x02U | アラームハンドラが動作している状態  |

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【NGKI2582】。

マルチプロセッサ対応カーネルでは、prcidに、対象アラームハンドラの割付けID番号が返される【NGKI2583】。プロセッサの

## 【使用上の注意】

ref\_almはデバッグ時向けの機能であり、その他の目的に使用することは推奨しない。これは、ref\_almを呼び出し、対象アラームハンドラの現在状態を参照した直後に割込みが発生した場合、

ref\_almから戻ってきた時には対象アラーム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。

#### 【μITRON4.0仕様との関係】

TALM\_STPとTALM\_STAを値を変更した。

### 1.6.4. オーバランハンドラ

オーバランハンドラは、タスクが使用したプロセッサ時間が、指定した時間を超えた場合に起動されるタイムイベントハンドラである。オーバランハンドラは、システムで1つのみ登録することができる【NGKI2584】。

オーバランハンドラ機能に関連して、各タスクが持つ情報は次の通り【NGKI2585】。

- オーバランハンドラの動作状態
- 残りプロセッサ時間

オーバランハンドラの動作状態は、タスク毎に、動作している状態と動作していない状態のいずれかをとる【NGKI2586】。残りプロセッサ時間は、オーバランハンドラが動作している状態の時に、タスクが使用できる残りのプロセッサ時間を表す。

オーバランハンドラの動作状態は、タスクの登録時と、タスクが休止状態に遷移する時に、動作していない状態に初期化される【NGKI2587】。

残りプロセッサ時間は、オーバランハンドラが動作している状態でタスクが実行している間、タスクが使用したプロセッサ時間の分だけ減少する【NGKI2588】。残りプロセッサ時間が0になると（これをオーバランと呼ぶ），オーバランハンドラが起動される【NGKI2589】。

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チン、それから呼び出したサービルコール（拡張サービスコールを含む）のNGKI2590】。一方、タスクの実行中に起動されたカーネル管理の割込みハンドラ（割込みサービスルーチン、周期ハンドラ、アラームハンドラ、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例外ハンドラの実行時間は含まないが、割込みハンドラおよびCPU例外ハンドラの呼出し／復帰にかかる時間と、それらの入口処理と出口処理の一部の実行時間は含んでしまう【NGKI2591】。また、タスクの実行中に起動されたカーネル管理外の割込みハンドラとカーネル管理外のCPU例外ハンドラの実行時間も含む【NGKI2592】。

プロセッサ時間は、符号なしの整数型であるOVRTIM型で表し、単位はマイクロ秒とする【NGKI2593】。ただし、プロセッサ時間には、OVRTIM型に格納できる任意の値を指定できるとは限らず、指定できる値にターゲット定義の上限があるNGKI2594】。プロセッサ時間に指定できる最大値は、構成マクロに定義されている【NGKI2595】。また、タスクが使用したプロセッサ時間の計測精度はターゲットに依存する【NGKI2596】。

保護機能対応カーネルにおいて、オーバランハンドラは、カーネルドメインに属する【NGKI2597】。

ターゲット定義で、オーバランハンドラ機能がサポートされていない場合がある【NGKI2598】。オーバランハンドラ機能がサポートされている場合には、TOPPERS\_SUPPORT\_OVRHDRがマクロ定義される【NGKI2599】。サポートされていない場合にオーバランハンドラ機能のサービスコールを呼び出すと、E\_NOSPTエラーが返るか、リンク時にエラーとなる【NGKI2600】。

る【

オーバランハンドラ機能に用いるデータ型は次の通り。

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）【NGKI2601】

オーバランハンドラ属性に指定できる属性はない【NGKI2602】。そのためオーバランハンドラ属性には、TA\_NULLを指定しなければならない【NGKI2603】。

C言語によるオーバランハンドラの記述形式は次の通り【NGKI2604】。

```
void overrun_handler(ID tskid, intptr_t exinf)
{
    オーバランハンドラ本体
}
```

tskidにはオーバランを起こしたタスクのID番号が、exinfにはそのタスクの拡張情報が、それぞれ渡される【NGKI2605】。

オーバランハンドラ機能に関連するカーネル構成マクロは次の通り。

TMAX\_OVRTIM プロセッサ時間に指定できる最大値【NGKI2606】

TOPPERS\_SUPPORT\_OVRHDR オーバランハンドラ機能がサポートされている【NGKI2607】

### 【使用上の注意】

マルチプロセッサ対応カーネルでは、オーバランハンドラが異なるプロセッサで同時に実行される可能性があるので、注意が必要である。

### 【TOPPERS/ASPカーネルにおける規定】

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、オーバランハンドラ機能拡張パッケージを用いると、オーバランハンドラ機能を追加することができる【ASPS0185】。

### 【TOPPERS/FMPカーネルにおける規定】

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】.

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、オーバランハンドラをサポートする【HRPS0149】.

#### 【μITRON4.0仕様との関係】

OVRTIMの時間単位は、μITRON4.0仕様では実装定義としていたが、この仕様ではマイクロ秒と規定した。

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

|         |                              |
|---------|------------------------------|
| DEF_OVR | オーバランハンドラの定義 [S] 【NGKI2608】  |
| def_ovr | オーバランハンドラの定義 [TD] 【NGKI2609】 |

#### 【静的API】

```
DEF_OVR({ ATR ovratr, OVRHDR ovrhdr })
```

#### 【C言語API】

```
ER ercd = def_ovr(const T_DOVR *pk_dovr)
```

#### 【パラメータ】

T\_DOVR \* pk\_dovr オーバランハンドラの定義情報を入ったパケットへのポインタ（静的APIを除く）

\*オーバランハンドラの定義情報（パケットの内容）

ATR ovratr オーバランハンドラ属性  
OVRHDR ovrhdr オーバランハンドラの先頭番地

#### 【リターンパラメータ】

ER ercd 正常終了 (E\_OK) またはエラーコード

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し [s] 【NGKI2610】<br>・CPUロック状態からの呼出し [s] 【NGKI2611】 |
| E_RSATR | 予約属性<br>・ovrattrが無効 【NGKI2612】<br>・その他の条件については機能の項を参照                           |
| E_PAR   | パラメータエラー<br>・ovrhdrがプログラムの先頭番地として正しくない 【NGKI2613】                               |
| E_OACV  | オブジェクトアクセス違反<br>・システム状態に対する管理操作が許可されていない [sP]<br>【NGKI2614】                     |
| E_MACV  | メモリアクセス違反<br>・pk_dovrが指すメモリ領域への読み出しが許可されて<br>いない [sP] 【NGKI2615】                |
| E_OBJ   | オブジェクト状態エラー<br>・条件については機能の項を参照  |

## 【機能】

各パラメータで指定したオーバランハンドラ定義情報に従って、オーバランハンドラを定義する【NGKI2616】。ただし、def\_ovrにおいてpk\_dovrをNULLにした場合には、オーバランハンドラの定義を解除する【NGKI2617】。

静的APIにおいては、ovrattrは整数定数式パラメータ、ovrhdrは一般定数式パラメータである【NGKI2618】。

オーバランハンドラを定義する場合（DEF\_OVRの場合およびdef\_ovrにおいてNULL以外にした場合）で、すでにオーバランハンドラが定義されているエラーとなる【NGKI2619】。

保護機能対応カーネルにおいて、DEF\_OVRは、カーネルドメインの囲みの中に記述しなければならない。そうでない場合には、E\_RSATRエラーとなる【NGKI2621】。また、def\_ovrでオーバランハンドラを定義する場合には、オーバランハンドラ属性にTA\_DOM(domid)を指定した場合にはE\_RSATRエラーとなる【NGKI2622】。ただし、TA\_DOM(TDOM\_SELF)を指定した場合には、指定が無視され、E\_RSATRエラーは検出されない【NGKI2623】。

マルチプロセッサ対応カーネルでは、DEF\_OVRは、クラスの囲みの外に記述しなければならない。そうでない場合には、E\_RSATRエラーとなる【NGKI2625】。また、def\_ovrオーバランハンドラを定義する場合には、オーバランハンドラの属するクラスを設定する必要はなく、オーバランハンドラ属性にTA\_CLS(clsid)をE\_RSATRエラーとなる【NGKI2626】。ただし、TA\_CLS(TCLS\_SELF)を指定した場合には、指定が無視され、E\_RSATRエラーは検出されない【NGKI2627】。

オーバランハンドラの定義を解除する場合 (def\_ovrにおいてpk\_dovrをNULLにした場合) で、オーバランハンドラが定義されていない場合には、E\_OBJエラーとなる【NGKI2628】。

オーバランハンドラの定義を解除すると、オーバランハンドラの動作状態は、すべてのタスクに対して動作していない状態となる【NGKI2629】。

#### 【使用上の注意】

def\_ovrによりオーバランハンドラの定義を解除する場合、サービスコールの処理時間およびカーネル内での割込み禁止時間が、タスクの総数に比例して長くなる。特に、タスクの総数が多い場合、カーネル内での割込み禁止時間が長くなるため、注意が必要である。

#### 【TOPPERS/ASPカーネルにおける規定】

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のみをサポートする【ASPS0186】。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、DEF\_OVRのみをサポートする【HRPS0150】。

#### 【μITRON4.0仕様との関係】

ovrhldrのデータ型をOVRHDRに変更した。

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、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、一度定義を解除してから、再度定義する必要がある。

|          |                               |
|----------|-------------------------------|
| sta_ovr  | オーバランハンドラの動作開始 [T] 【NGKI2630】 |
| ista_ovr | オーバランハンドラの動作開始 [I] 【NGKI2631】 |

#### 【C言語API】

```
ER ercd = sta_ovr(ID tskid, OVRTIM ovrtim)
ER ercd = ista_ovr(ID tskid, OVRTIM ovrtim)
```

#### 【パラメータ】

|        |        |                 |
|--------|--------|-----------------|
| ID     | tskid  | 対象タスクのID番号      |
| OVRTIM | ovrtim | 対象タスクの残りプロセッサ時間 |

#### 【リターンパラメータ】

ER

ercd

正常終了 (E\_OK) またはエラーコード

### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー   |
|         | ・非タスクコンテキストからの呼出し (sta_ovrの場合) 【NGKI2632】             |
|         | ・タスクコンテキストからの呼出し (ista_ovrの場合) 【NGKI2633】             |
|         | ・CPUロック状態からの呼出し 【NGKI2634】                            |
| E_ID    | 不正ID番号  |
|         | ・tskidが有効範囲外 【NGKI2635】                               |
| E_NOEXS | オブジェクト未登録   |
|         | ・対象タスクが未登録 [D] 【NGKI2636】                             |
| E_OACV  | オブジェクトアクセス違反  |
|         | ・対象タスクに対する通常操作2が許可されていない (sta_ovr の場合) [P] 【NGKI2637】 |
| E_PAR   | パラメータエラー  |
|         | ・ovrtimが0, またはTMAX_OVRTIMより大きい 【NGKI2643】             |
| E_OBJ   | オブジェクト状態エラー   |
|         | ・オーバランハンドラが定義されていない 【NGKI2638】                        |

### 【機能】

tskidで指定したタスク（対象タスク）に対して、オーバランハンドラの動作を開始する。具体的な振舞いは以下の通り。

対象タスクに対するオーバランハンドラの動作状態は、動作している状態となり、残りプロセッサ時間は、ovrtimに指定した時間に設定される【NGKI2639】。

対象タスクに対してオーバランハンドラが動作している状態であれば、残りプロセッサ時間の設定のみが行われる【NGKI2640】。

sta\_ovrにおいてtskidにTSK\_SELF (=0) を指定すると、自タスクが対象タスクとなる【NGKI2641】。

### 【μITRON4.0仕様との関係】

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。

|          |                               |
|----------|-------------------------------|
| stp_ovr  | オーバランハンドラの動作停止 [T] 【NGKI2644】 |
| istp_ovr | オーバランハンドラの動作停止 [I] 【NGKI2645】 |

### 【C言語API】

```
ER ercd = stp_ovr(ID tskid)
ER ercd = istp_ovr(ID tskid)
```

## 【パラメータ】

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

## 【リターンパラメータ】

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

## 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し (stp_ovrの場合) 【NGKI2646】<br>・タスクコンテキストからの呼出し (istp_ovrの場合) 【NGKI2647】<br>・CPUロック状態からの呼出し 【NGKI2648】 |
| E_ID    | 不正ID番号<br>・tskidが有効範囲外 【NGKI2649】   |
| E_NOEXS | オブジェクト未登録<br>・対象タスクが未登録 [D] 【NGKI2650】  |
| E_OACV  | オブジェクトアクセス違反<br>・対象タスクに対する通常操作2が許可されていない (stp_ovr<br>の場合) [P] 【NGKI2651】  |
| E_OBJ   | オブジェクト状態エラー<br>・オーバランハンドラが定義されていない 【NGKI2652】   |

## 【機能】

tskidで指定したタスク（対象タスク）に対して、オーバランハンドラの動作を停止する。具体的な振舞いは以下の通り。

対象タスクに対するオーバランハンドラの動作状態は、動作していない状態となる【NGKI2653】。対象タスクに対してオーバランハンドラが動作していない状態であれば、何も行われずに正常終了する【NGKI2654】。

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となる【NGKI2655】。

## 【μITRON4.0仕様との関係】

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。

|         |                               |
|---------|-------------------------------|
| ref_ovr | オーバランハンドラの状態参照 [T] 【NGKI2656】 |
|---------|-------------------------------|

## 【C言語API】

```
ER ercd = ref_ovr(ID tskid, T_ROVR *pk_rovr)
```

### 【パラメータ】

|          |         |                              |
|----------|---------|------------------------------|
| ID       | tskid   | 対象タスクのID番号                   |
| T_ROVR * | pk_rovr | オーバランハンドラの現在状態を入れるパケットへのポインタ |

### 【リターンパラメータ】

ER ercd 正常終了 (E\_OK) またはエラーコード

\* タスクの現在状態 (パケットの内容)

|        |          |                |
|--------|----------|----------------|
| STAT   | ovrstat  | オーバランハンドラの動作状態 |
| OVRTIM | lefttotm | 残りプロセッサ時間      |

### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼び出し 【NGKI2657】<br>・CPUロック状態からの呼び出し 【NGKI2658】 |
| E_ID    | 不正ID番号<br>・tskidが有効範囲外 【NGKI2659】   |
| E_NOEXS | オブジェクト未登録<br>・対象タスクが未登録 [D] 【NGKI2660】                                    |
| E_OACV  | オブジェクトアクセス違反<br>・対象タスクに対する参照操作が許可されていない [P] 【NGKI2661】                    |
| E_MACV  | メモリアクセス違反<br>・pk_rovrが指すメモリ領域への書き込みアクセスが許可されていない [P] 【NGKI2662】           |
| E_OBJ   | オブジェクト状態エラー<br>・オーバランハンドラが定義されていない 【NGKI2663】                             |

### 【機能】

tskidで指定したタスク（対象タスク）に対するオーバランハンドラの現在状態を参照する。参照した現在状態は、pk\_rovrで指定したメモリ領域に返される 【NGKI2664】。

ovrstatには、対象タスクに対するオーバランハンドラの動作状態を表す次のいずれかの値が返される 【NGKI2665】。

|          |       |                     |
|----------|-------|---------------------|
| TOVR_STP | 0x01U | オーバランハンドラが動作していない状態 |
| TOVR_STA | 0x02U | オーバランハンドラが動作している状態  |

対象タスクに対してオーバランハンドラが動作している状態の場合には、leftotmに、オーバランハンドラが起動されるまでの残りプロセッサ時間が返さ  
れる【NGKI2666】。オーバランハンドラが起動される直前には、leftotmに0が  
返される可能性がある【NGKI2667】。オーバランハンドラが動作していない状  
態の場合は保証されない【NGKI2668】。

tskidにTSK\_SELF (=0) を指定すると、自タスクが対象タスクとなる【NGKI2669】。

#### 【使用上の注意】

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

#### 【未決定事項】

マルチプロセッサ対応カーネルにおいて、対象タスクが、自タスクが割付けられたプロセッサと異なるプロセッサに割り付けられている場合に、leftotmを参照できるとするかどうかは、今後の課題である。

#### 【μITRON4.0仕様との関係】

TOVR\_STPとTOVR\_STAを値を変更した。

## 1.7. システム状態管理機能

システム状態管理機能は、特定のオブジェクトに関連しないシステムの状態を変更／参照するための機能である。

|         |                                      |
|---------|--------------------------------------|
| SAC_SYS | システム状態のアクセス許可ベクタの設定 [SP] 【NGKI2670】  |
| sac_sys | システム状態のアクセス許可ベクタの設定 [TPD] 【NGKI2671】 |

#### 【静的API】

```
SAC_SYS({ ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
```

#### 【C言語API】

```
ER ercd = sac_sys(const ACVCT *p_acvct)
```

#### 【パラメータ】

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

\*アクセス許可ベクタ（パケットの内容）

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

#### 【リターンパラメータ】

ER ercd 正常終了（E\_OK）またはエラーコード

#### 【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー                              |
|         | ・非タスクコンテキストからの呼び出し [s] 【NGKI2672】      |
|         | ・CPUロック状態からの呼び出し [s] 【NGKI2673】        |
| E_RSATR | 予約属性                                   |
|         | ・カーネルドメインの囲みの中に記述されていない [S] 【NGKI2674】 |
|         | ・クラスの囲みの中に記述されている [SM] 【NGKI2675】      |
| E_OACV  | オブジェクトアクセス違反                           |
|         | ・カーネルドメイン以外からの呼び出し [s] 【NGKI2676】      |
| E_OBJ   | オブジェクト状態エラー                            |
|         | ・システム状態のアクセス許可ベクタが設定済み [S] 【NGKI2677】  |

#### 【機能】

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、各パラメータで指定した値に設定する【NGKI2678】。

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】。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、SAC\_SYSのみをサポートする【HRPS0151】。

|          |                            |
|----------|----------------------------|
| rot_rdq  | タスクの優先順位の回転 [T] 【NGKI2680】 |
| irot_rdq | タスクの優先順位の回転 [I] 【NGKI2681】 |

#### 【C言語API】

```
ER ercd = rot_rdq(PRI tskpri)
ER ercd = irot_rdq(PRI tskpri)
```

### 【パラメータ】

|     |        |                 |
|-----|--------|-----------------|
| PRI | tskpri | 回転対象の優先度（対象優先度） |
|-----|--------|-----------------|

### 【リターンパラメータ】

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

### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー   |
|         | ・非タスクコンテキストからの呼び出し (rot_rdqの場合) 【NGKI2682】            |
|         | ・タスクコンテキストからの呼び出し (irot_rdqの場合) 【NGKI2683】            |
|         | ・CPUロック状態からの呼び出し 【NGKI2684】                           |
| E_NOSPT | 未サポート機能   |
|         | ・条件については機能の項を参照                                       |
| E_PAR   | パラメータエラー  |
|         | ・tskpriが有効範囲外 【NGKI2685】                              |
| E_OACV  | オブジェクトアクセス違反  |
|         | ・システム状態に対する通常操作1が許可されていない (rot_rdqの場合) [P] 【NGKI2686】 |

### 【機能】

tskpriで指定した優先度（対象優先度）を持つ実行できる状態のタスクの中で、最も優先順位が高いタスクを、同じ優先度のタスクの中で最も優先順位が低い状態にする【NGKI2687】。対象優先度を持つ実行できる状態のタスクが無いか1つのみの場合には、何も行われずに正常終了する【NGKI2688】。

マルチプロセッサ対応カーネルにおいては、自タスクと同じプロセッサに割り付けられているタスクのみを操作対象とする【NGKI3622】。

rot\_rdqにおいて、tskpriにTPRI\_SELF (=0) を指定すると、自タスクのベース優先度が対象優先度となる【NGKI2689】。

対象優先度を持つ実行できる状態のタスクの中で、最も優先順位が高いタスクが制約タスクの場合には、E\_NOSPTエラーとなる【NGKI2690】。

### 【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、rot\_rdq, irot\_rdqをサポートしない【SSPS0131】。

`mrot_rdq` プロセッサ指定でのタスクの優先順位の回転〔TM〕 【NGKI2691】  
`imrot_rdq` プロセッサ指定でのタスクの優先順位の回転〔IM〕 【NGKI2692】

#### 【C言語API】

```
ER ercd = mrot_rdq(PRI tskpri, ID prcid)
ER ercd = imrot_rdq(PRI tskpri, ID prcid)
```

#### 【パラメータ】

|     |        |                        |
|-----|--------|------------------------|
| PRI | tskpri | 回転対象の優先度（対象優先度）        |
| ID  | prcid  | 優先順位の回転対象とするプロセッサのID番号 |

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼び出し（ <code>mrot_rdq</code> の場合）<br>【NGKI2693】<br>・タスクコンテキストからの呼び出し（ <code>imrot_rdq</code> の場合）【NGKI2694】<br>・CPUロック状態からの呼び出し【NGKI2695】 |
| E_NOSPT | 未サポート機能<br>・条件については機能の項を参照  |
| E_ID    | 不正ID番号<br>・ <code>prcid</code> が有効範囲外【NGKI2696】   |
| E_PAR   | パラメータエラー<br>・ <code>tskpri</code> が有効範囲外【NGKI2697】  |
| E_OACV  | オブジェクトアクセス違反<br>・システム状態に対する通常操作1が許可されていない（ <code>mrot_rdq</code> の場合）〔P〕【NGKI2698】  |

#### 【機能】

`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（対象優先度）を持つ実行できる状態のタスクの中で、最も優先順位が高いタスクを、同じ優先度のタスクの中で最も優先順位が低い状態にする【NGKI2699】。  
対象優先度を持つ実行できる状態のタスクが無いか1つのみの場合には、何も行われずに正常終了する【NGKI2700】。

mrot\_rdqにおいて、tskpriにTPRI\_SELF (=0) を指定すると、自タスクのベース優先度が対象優先度となる【NGKI2701】.

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る状態のタスクの中で、最も優先順位が高いタスクが制約タスクの場合には、E\_NOSPTエラーとなる【NGKI2702】.

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、mrot\_rdq, imrot\_rdqをサポートしない【ASPS0188】.

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、mrot\_rdq, imrot\_rdqをサポートしない【HRPS0152】.

#### 【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、mrot\_rdq, imrot\_rdqをサポートしない【SSPS0132】.

#### 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである。

get\_tid 実行状態のタスクIDの参照 [T] 【NGKI2703】

iget\_tid 実行状態のタスクIDの参照 [I] 【NGKI2704】

#### 【C言語API】

```
ER ercd = get_tid(ID *p_tskid)
ER ercd = iget_tid(ID *p_tskid)
```

#### 【パラメータ】

ID \* p\_tskid タスクIDを入れるメモリ領域へのポインタ

#### 【リターンパラメータ】

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

#### 【エラーコード】

|        |   |
|--------|---|
| E_CTX  | コンテキストエラー<br>・非タスクコンテキストからの呼出し (get_tidの場合) 【NGKI2705】<br>・タスクコンテキストからの呼出し (iget_tidの場合) 【NGKI2706】<br>・CPUロック状態からの呼出し 【NGKI2707】 |
| E_MACV | メモリアクセス違反<br>・p_tskidが指すメモリ領域への書き込みアクセスが許可されていない (get_tidの場合) [P] 【NGKI2708】  |

## 【機能】

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。参照したタスクIDは、p\_tskidが指すメモリ領域に返される【NGKI2709】。

iget\_tidにおいて、実行状態のタスクがない場合には、TSK\_NONE (=0) が返される【NGKI2710】。

マルチプロセッサ対応カーネルにおいては、サービスコールを呼び出した処理単位を実行しているプロセッサにおいて実行状態のタスクのID番号を参照する【NGKI2711】。

## 【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、get\_tidをサポートしない【SSPS0133】。

|         |   |
|---------|---|
| 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】 |
|---------|---|

## 【C言語API】

|                                |
|--------------------------------|
| ER ercd = get_did(ID *p_domid) |
|--------------------------------|

## 【パラメータ】

|      |         |                         |
|------|---------|-------------------------|
| ID * | p_domid | 保護ドメインIDを入れるメモリ領域へのポインタ |
|------|---------|-------------------------|

## 【リターンパラメータ】

|    |       |                       |
|----|-------|-----------------------|
| ER | ercd  | 正常終了 (E_OK) またはエラーコード |
| ID | domid | 保護ドメインID              |

## 【エラーコード】

|        |   |
|--------|---|
| E_CTX  | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI2713】<br>・CPUロック状態からの呼出し【NGKI2714】 |
| E_MACV | メモリアクセス違反<br>・p_domidが指すメモリ領域への書き込みアクセスが許可されていない【NGKI2715】            |

## 【機能】

実行状態のタスク（自タスク）が属する保護ドメインのID番号を参照する。参考は、p\_domidが指すメモリ領域に返される【NGKI2716】。

マルチプロセッサ対応カーネルにおいては、サービスコールを呼び出した処理単位を実行しているプロセッサにおいて実行状態のタスクが属する保護ドメインのID番号を参照する【NGKI2717】。

## 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、get\_idをサポートしない【ASPS0189】。

## 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、get\_idをサポートしない【FMPS0157】。

## 【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、get\_idをサポートしない【SSPS0134】。

|          |                                  |
|----------|----------------------------------|
| get_pid  | 割付けプロセッサのID番号の参照 [TM] 【NGKI2718】 |
| iget_pid | 割付けプロセッサのID番号の参照 [IM] 【NGKI2719】 |

## 【C言語API】

```
ER ercd = get_pid(ID *p_prcid)
ER ercd = iget_pid(ID *p_prcid)
```

## 【パラメータ】

|      |         |                        |
|------|---------|------------------------|
| ID * | p_prcid | プロセッサIDを入れるメモリ領域へのポインタ |
|------|---------|------------------------|

## 【リターンパラメータ】

|    |       |                       |
|----|-------|-----------------------|
| ER | ercd  | 正常終了 (E_OK) またはエラーコード |
| ID | prcid | プロセッサID               |

## 【エラーコード】

|        |  |
|--------|--|
| E_CTX  | コンテキストエラー  |
|        | ・非タスクコンテキストからの呼び出し (get_pidの場合) 【NGKI2720】                       |
|        | ・タスクコンテキストからの呼び出し (iget_pidの場合) 【NGKI2721】                       |
|        | ・CPUロック状態からの呼び出し 【NGKI2722】                                      |
| E_MACV | メモリアクセス違反  |
|        | ・p_prclid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 (get_pidの場合) [P] 【NGKI2723】 |

## 【機能】

サービスコールを呼び出した処理単位の割付けプロセッサのID番号を参照する。参考したプロセッサIDは、p\_prclidが指すメモリ領域に返される【NGKI2724】。

## 【使用上の注意】

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

## 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、get\_pid, iget\_pidをサポートしない【ASPS0190】。

## 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、get\_pid, iget\_pidをサポートしない【HRPS0153】。

## 【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、get\_pid, iget\_pidをサポートしない【SSPS0135】。

## 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである。

|          |                             |
|----------|-----------------------------|
| loc_cpu  | CPUロック状態への遷移 [T] 【NGKI2725】 |
| iloc_cpu | CPUロック状態への遷移 [I] 【NGKI2726】 |

## 【C言語API】

```
ER ercd = loc_cpu()  
ER ercd = iloc_cpu()
```

### 【パラメータ】

なし

### 【リターンパラメータ】

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

### 【エラーコード】

|        |   |
|--------|---|
| E_CTX  | コンテキストエラー<br>・非タスクコンテキストからの呼び出し (loc_cpuの場合) 【NGKI2727】<br>・タスクコンテキストからの呼び出し (iloc_cpuの場合) 【NGKI2728】 |
| E_OACV | オブジェクトアクセス違反<br>・システム状態に対する通常操作2が許可されていない<br>(loc_cpuの場合) [P] 【NGKI2729】                              |

### 【機能】

CPUロックフラグをセットし、CPUロック状態へ遷移する【NGKI2730】。CPUロック状態で呼び出した場合には、何も行われずに正常終了する【NGKI2731】。

```
unl_cpu    CPUロック状態の解除 [T] 【NGKI2732】  
iunl_cpu   CPUロック状態の解除 [I] 【NGKI2733】
```

### 【C言語API】

```
ER ercd = unl_cpu()  
ER ercd = iunl_cpu()
```

### 【パラメータ】

なし

### 【リターンパラメータ】

ER

ercd

正常終了 (E\_OK) またはエラーコード

### 【エラーコード】

E\_CTX

コンテキストエラー

- ・非タスクコンテキストからの呼び出し (unl\_cpuの場合) 【NGKI2734】
- ・タスクコンテキストからの呼び出し (iunl\_cpuの場合) 【NGKI2735】

E\_OACV

オブジェクトアクセス違反

- ・システム状態に対する通常操作2が許可されていない  
(unl\_cpuの場合) [P] 【NGKI2736】

### 【機能】

CPUロックフラグをクリアし、CPUロック解除状態へ遷移する【NGKI2737】。

CPUロック解除状態で呼び出した場合には、何も行われずに正常終了する【NGKI2738】。

マルチプロセッサ対応カーネルにおいて、unl\_cpu/iunl\_cpuを呼び出したプロセッサによって取得されている状態となっているスピンドルロックがある場合には、unl\_cpu/iunl\_cpuによってCPUロック解除状態に遷移しない（何も行われずに正常終了する）【NGKI2739】。

### 【補足説明】

マルチプロセッサ対応カーネルでは、CPUロック解除状態へ遷移した結果、ディスパッチ保留状態が解除され、ディスパッチが起こる可能性がある。また、保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チンの実行が開始される可能性がある。

dis\_dsp ディスパッチの禁止 [T] 【NGKI2740】

### 【C言語API】

```
ER ercd = dis_dsp()
```

### 【パラメータ】

なし

### 【リターンパラメータ】

ER

ercd

正常終了 (E\_OK) またはエラーコード

### 【エラーコード】

|        |   |
|--------|---|
| E_CTX  | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI2741】<br>・CPUロック状態からの呼出し【NGKI2742】 |
| E_OACV | オブジェクトアクセス違反<br>・システム状態に対する通常操作1が許可されていない〔P〕<br>【NGKI2743】            |

## 【機能】

ディスパッチ禁止フラグをセットし、ディスパッチ禁止状態へ遷移する  
【NGKI2744】。ディスパッチ禁止状態で呼び出した場合には、何も行われずに正常終了する【NGKI2745】。

ena\_dsp ディスパッチの許可〔T〕 【NGKI2746】

## 【C言語API】

ER ercd = ena\_dsp()

## 【パラメータ】

なし

## 【リターンパラメータ】

ER ercd 正常終了（E\_OK）またはエラーコード

## 【エラーコード】

|        |   |
|--------|---|
| E_CTX  | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI2747】<br>・CPUロック状態からの呼出し【NGKI2748】 |
| E_OACV | オブジェクトアクセス違反<br>・システム状態に対する通常操作1が許可されていない〔P〕<br>【NGKI2749】            |

## 【機能】

ディスパッチ禁止フラグをクリアし、ディスパッチ許可状態へ遷移する  
【NGKI2750】。ディスパッチ許可状態で呼び出した場合には、何も行われずに正常終了する【NGKI2751】。

## 【補足説明】

ディスパッチ許可状態へ遷移した結果、ディスパッチ保留状態が解除され、ディスパッチが起こる可能性がある。

`sns_ctx` コンテキストの参照【TI】【NGKI2752】

【C言語API】

```
bool_t state = sns_ctx()
```

【パラメータ】

なし

【リターンパラメータ】

`bool_t state` コンテキスト

【機能】

実行中のコンテキストを参照する。具体的な振舞いは以下の通り。

`sns_ctx`を非タスクコンテキストから呼び出した場合にはtrue、タスクコンテキストから呼び出した場合にはfalseが返る【NGKI2753】。

`sns_loc` CPUロック状態の参照【TI】【NGKI2754】

【C言語API】

```
bool_t state = sns_loc()
```

【パラメータ】

なし

【リターンパラメータ】

`bool_t state` CPUロックフラグ

【機能】

CPUロックフラグを参照する。具体的な振舞いは以下の通り。

sns\_locをCPUロック状態で呼び出した場合にはtrue、CPUロック解除状態で呼びfalseが返る【NGKI2755】。

出した場合には

**sns\_dsp** ディスパッチ禁止状態の参照 [TI] 【NGKI2756】

#### 【C言語API】

```
bool_t state = sns_dsp()
```

#### 【パラメータ】

なし

#### 【リターンパラメータ】

**bool\_t state** ディスパッチ禁止フラグ

#### 【機能】

ディスパッチ禁止フラグを参照する。具体的な振舞いは以下の通り。

sns\_dspをディスパッチ禁止状態で呼び出した場合にはtrue、ディスパッチ許可状態で呼び出した場合にはfalseが返る【NGKI2757】。

**sns\_dpn** ディスパッチ保留状態の参照 [TI] 【NGKI2758】

#### 【C言語API】

```
bool_t state = sns_dpn()
```

#### 【パラメータ】

なし

#### 【リターンパラメータ】

**bool\_t state** ディスパッチ保留状態

## 【機能】

ディスパッチ保留状態であるか否かを参照する。具体的な振舞いは以下の通り。

sns\_dpnをディスパッチ保留状態で呼び出した場合にはtrue、ディスパッチ保留状態でない状態で呼び出した場合にはfalseが返る【NGKI2759】。

**sns\_ker** カーネル非動作状態の参照 [TI] 【NGKI2760】

## 【C言語API】

```
bool_t state = sns_ker()
```

## 【パラメータ】

なし

## 【リターンパラメータ】

**bool\_t state** カーネル非動作状態

## 【機能】

カーネルが動作中であるか否かを参照する。具体的な振舞いは以下の通り。

sns\_kerをカーネルの初期化完了前（初期化ルーチン実行中を含む）または終了処理開始後（終了処理ルーチン実行中を含む）に呼び出した場合にはtrue、カーネルの動作中に呼び出した場合にはfalseが返る【NGKI2761】。

## 【使用方法】

sns\_kerは、カーネルが動作している時とそうでない時で、処理内容を変えたいがtrueを返した場合、他のサービスコールを呼び出す場合に使用する。sns\_kerがtrueを返す時に他のサービスコールを呼び出した場合の動作は保証されない。

## 【使用上の注意】

どちらの条件でtrueが返るか間違いやさるので注意すること。

## 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである。

**ext\_ker** カーネルの終了 [TI] 【NGKI2762】

## 【C言語API】

```
ER ercd = ext_ker()
```

## 【パラメータ】

なし

## 【リターンパラメータ】

|    |      |        |
|----|------|--------|
| ER | ercd | エラーコード |
|----|------|--------|

## 【エラーコード】

|        |  |
|--------|--|
| E_SYS  | システムエラー<br>・カーネルの誤動作 【NGKI2763】                  |
| E_OACV | オブジェクトアクセス違反<br>・カーネルドメイン以外からの呼出し [P] 【NGKI2764】 |

## 【機能】

カーネルを終了する。具体的な振舞いについては、「2.9.2 システム終了手順」の節を参照すること。

ext\_kerが正常に処理された場合、ext\_kerからはリターンしない【NGKI2765】。

## 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである。

|         |               |
|---------|---------------|
| ref_sys | システムの状態参照 [T] |
|---------|---------------|

## 【C言語API】

```
ER ercd = ref_sys(T_RSYS *pk_rsys)
```

未完成

## 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、ref\_sysをサポートしない。

## 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、ref\_sysをサポートしない。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、ref\_sysをサポートしない。

#### 【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、ref\_sysをサポートしない。

## 1.8. メモリオブジェクト管理機能

メモリオブジェクト管理機能は、保護機能対応カーネルでのみサポートされる機能である。保護機能対応でないカーネルでは、メモリオブジェクト管理機能をサポートしない。

### 〔メモリリージョン〕

メモリリージョンは、オブジェクトモジュールに含まれるセクションの配置対象となる同じ性質を持った連続したメモリ領域である。メモリリージョンは、メモリリージョン名によって識別する【NGKI2766】。

各メモリリージョンが持つ情報は次の通り【NGKI2767】。

- ・先頭番地
- ・サイズ
- ・メモリリージョン属性

メモリリージョンの先頭番地とサイズには、ターゲット定義の制約が課せられる場合がある【NGKI2768】。

メモリリージョン属性には、次の属性を指定することができる【NGKI3256】。

TA\_NOWRITE 0x01U 書込みアクセス禁止

ターゲットによっては、ターゲット定義のメモリリージョン属性を指定できる場合がある【NGKI2771】。

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 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標準RAMリージョンが含まれる。

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、クラス毎の標準メモリリージョンを定義することができる【NGKI3257】。

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、共通の標準メモリリージョン）は、必ず定義しなければならない。定義しない場合には、コンフィギュレータがエラーを報告する【NGKI3259】。

## 〔メモリオブジェクト〕

メモリオブジェクトは、保護機能対応カーネルにおいてアクセス保護の対象とする連続したメモリ領域である。メモリオブジェクトは、その先頭番地によって識別する【NGKI2772】。

各メモリオブジェクトが持つ情報は次の通り【NGKI2773】。

- 先頭番地
- サイズ
- メモリオブジェクト属性
- アクセス許可ベクタ
- 属する保護ドメイン
- 属するクラス（マルチプロセッサ対応カーネルの場合）

メモリオブジェクトの先頭番地とサイズには、ターゲット定義の制約が課せられる【NGKI2774】。

メモリオブジェクト属性には、次の属性を指定することができる【NGKI2775】。

|            |       |              |
|------------|-------|--------------|
| TA_NOWRITE | 0x01U | 書込みアクセス禁止    |
| TA_NOREAD  | 0x02U | 読み出しアクセス禁止   |
| TA_EXEC    | 0x04U | 実行アクセス許可     |
| TA_MEMINI  | 0x08U | メモリの初期化を行う   |
| TA_MEMPRSV | 0x10U | メモリの初期化を行わない |
| TA_SDATA   | 0x20U | ショートデータ領域に配置 |
| TA_UNCACHE | 0x40U | キャッシュ禁止      |
| TA_IODEV   | 0x80U | 周辺デバイスの領域    |

メモリオブジェクトに対して書込みアクセスできるのは、メモリオブジェクト属性に書込みアクセス禁止（TA\_NOWRITE属性）が指定されておらず、アクセス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】。

また、読み出しアクセスできるのは、メモリオブジェクト属性に読み出しアクセス（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み出し・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）が指定されており、アクセス許可ベクタにより読み出し・実行アクセスが許可されている場合である【NGKI2778】。

ただし、ターゲットハードウェアの制約によってこれらの属性を実現できない場合には、次のように扱われる。書込みアクセス禁止が実現できない場合には、TA\_NOWRITEを指定しても無視される【NGKI2779】。また、読み出しアクセス禁止が実現できない場合には、TA\_NOREADを指定しても無視される【NGKI2780】。実行アクセス禁止が実現できない場合には、TA\_EXECを指定しなくても実行アクセス（TA\_EXEC）は無視される【NGKI2781】。どのような場合にどの属性の指定が無視されるかは、ターゲット定義である【NGKI2782】。

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）される【NGKI2784】。

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、  
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】。

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  
NGKI2788】。具体的な扱いはターゲット定義であるが、ショートデータ  
領域がサポートされていないターゲットでは、この属性は無視される  
NGKI2789】。また、ターゲットによっては、TA\_NOWRITEを指定した場合に、  
TA\_SDATAが無視される場合がある【NGKI2790】。

TA\_UNCACHE属性は、メモリオブジェクトをキャッシュ禁止に設定することを、  
TA\_IODEV属性は、メモリオブジェクトを周辺デバイスの領域として扱うことを  
NGKI2791】。具体的な扱いはターゲット定義であるが、これらの属性を  
指定しても意味がないターゲット（例えば、キャッシュを持たないターゲット  
TA\_UNCACHE）では、これらの属性は無視される【NGKI2792】。

逆に、キャッシュ禁止にできないメモリオブジェクトに対してTA\_UNCACHEを指  
定した場合や、周辺デバイスの領域として扱うことができないメモリオブジェ  
クトに対してTA\_IODEVを指定した場合には、E\_RSATRエラーとなる【NGKI2793】。

ターゲットによっては、ターゲット定義のメモリオブジェクト属性を指定でき  
る場合がある【NGKI2794】。ターゲット定義のメモリオブジェクト属性として、次の属性を予約している【NGKI2795】。

TA\_WTHROUGH ライトスルーキャッシュを用いる

#### 〔カーネル構成マクロ〕

メモリオブジェクト管理機能に関連するカーネル構成マクロは次の通り。

|                         |   |
|-------------------------|---|
| TOPPERS_SUPPORT_ATT_MOD | ATT_MOD／ATA_MODがサポートされている<br>【NGKI2796】         |
| TOPPERS_SUPPORT_ATT_PMA | ATT_PMA／ATA_PMA／att_pmaがサポートさ<br>れている【NGKI2797】 |

ただし、att\_pmaは、動的生成対応カーネルのみでサポートされるAPIであるため、サポートされているかを判定するには、TOPPERS\_SUPPORT\_DYNAMIC\_CREとTOPPERS\_SUPPORT\_ATT\_PMAの両方が定義されていることをチェックする必要がある【NGKI2798】。

#### 【補足説明】

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。

標準

示す【

】

示す【

プロセッサでの

クトに対して

る場合がある【

. 【NGKI2795】

### 【TOPPERS/ASPカーネルにおける規定】

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】。

### 【TOPPERS/FMPカーネルにおける規定】

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】。

### 【TOPPERS/HRP2カーネルにおける規定】

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】。

### 【TOPPERS/SSPカーネルにおける規定】

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】。

### 【μITRON4.0/PX仕様との関係】

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して廃止した。TA\_RO  
はTA\_NOWRITEに改名し、TA\_NOREAD, TA\_EXEC, TA\_MEMINI, TA\_MEMPRSV,  
TA\_IODEVを追加した。また、TA\_UNCACHEの値を変更し、ターゲッ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。

メモリリージョンは、μITRON4.0/PX仕様にはない概念である。

### 【仕様決定の理由】

TA\_IODEV属性を導入したのは、ターゲットプロセッサによっては、周辺デバイスの領域として扱うためには、キャッシュ禁止に加えて、メモリのアクセス順序を変更しないことを指定しなければならないためである。メモリのアクセス順序を変更しないことを指定するメモリオブジェクト属性を、ターゲット定義で用意してもよいが、それを使うとアプリケーションのポータビリティが下がるため、TA\_IODEV属性を用意することにした。

ATT\_REG メモリリージョンの登録 [SP] 【NGK12799】

### 【静的API】

```
ATT_REG("メモリリージョン名", { ATR regatr, void *base, SIZE size })
```

### 【パラメータ】

|             |                             |
|-------------|-----------------------------|
| "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        |
| ATR         | regatr メモリリージョン属性           |
| void *      | base 登録するメモリリージョンの先頭番地      |
| SIZE        | size 登録するメモリリージョンのサイズ（バイト数） |

### 【エラーコード】

|         |                                |
|---------|--------------------------------|
| E_RSATR | 予約属性                           |
|         | ・regatrが無効【NGKI2800】           |
|         | ・保護ドメインの囲みの中に記述されている【NGKI2814】 |
|         | ・クラスの囲みの中に記述されている〔M〕【NGKI3260】 |
| E_PAR   | パラメータエラー                       |
|         | ・sizeが0以下【NGKI2816】            |
|         | ・その他の条件については機能の項を参照            |
| E_OBJ   | オブジェクト状態エラー                    |
|         | ・登録済みのメモリリージョンの再登録【NGKI2801】   |
|         | ・その他の条件については機能の項を参照            |

### 【機能】

各パラメータで指定したメモリリージョン登録情報に従って、指定したメモリリージョンを登録する。具体的な振舞いは以下の通り。

baseとsizeで指定したメモリ領域が、メモリリージョンとして登録される【NGKI2802】。登録されるメモリリージョンには、regatrで指定したメモリリージョン属性が設定される【NGKI2803】。

メモリリージョン名は文字列パラメータ、regatr、base、sizeは整数定数式パラメータである【NGKI2804】。

baseやsizeに、ターゲット定義の制約に合致しない先頭番地やサイズを指定しE\_PARエラーとなる【NGKI2815】。登録しようとしたメモリリージョンが、登録済みのメモリリージョンとメモリ領域が重なる場合には、E\_OBJエラーとなる【NGKI2817】。

### 【μITRON4.0/PX仕様との関係】

μITRON4.0/PX仕様に定義されていない静的APIである。

DEF\_SRG 標準メモリリージョンの定義〔SP〕【NGKI3261】

### 【静的API】

```
DEF_SRG("標準ROMリージョン名", "標準RAMリージョン名")
```

## 【パラメータ】

|               |                               |
|---------------|-------------------------------|
| "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指定する文字列 |
| "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指定する文字列 |

## 【エラーコード】

|         |   |
|---------|---|
| E_RSATR | 予約属性<br>・保護ドomainの囲みの中に記述されている 【NGKI3262】   |
| E_OBJ   | オブジェクト状態エラー<br>・標準メモリリージョンが定義済み 【NGKI3263】<br>・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 【NGKI3264】<br>・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 【NGKI3272】<br>・その他の条件については機能の項を参照 |

## 【機能】

各パラメータに従って、標準ROMリージョンと標準RAMリージョンを定義する 【NGKI3265】。

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、  
共通の標準ROMリージョンと標準RAMリージョンを定義し、クラスの囲みの中に  
記述すると、そのクラスの標準ROMリージョンと標準RAMリージョンを定義する 【NGKI3266】。

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。  
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  
E\_OBJエラーとなる【NGKI3268】。また、標準RAMリージョンは、  
TA\_NOWRITE属性でないメモリリージョンでなければならない。標準RAMリージョンとして  
指定したメモリリージョンが、TA\_NOWRITE属性である場合には、  
E\_OBJエラーとなる【NGKI3270】。

## 【μITRON4.0/PX仕様との関係】

μITRON4.0/PX仕様に定義されていない静的APIである。

|         |                                       |
|---------|---------------------------------------|
| ATT_SEC | セクションの登録 [SP] 【NGKI2818】              |
| ATA_SEC | セクションの登録（アクセス許可ベクタ付き） [SP] 【NGKI2819】 |

## 【静的API】

```
ATT_SEC("セクション名", { ATR mematr, "メモリリージョン名" })
ATA_SEC("セクション名", { ATR mematr, "メモリリージョン名" },
        { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
```

### 【パラメータ】

|             |                            |
|-------------|----------------------------|
| "セクション名"    | 登録するセクションを指定する文字列          |
| ATR         | mematr メモリオブジェクト属性         |
| "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する文字列 |

\*アクセス許可ベクタ (パケットの内容)

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

### 【エラーコード】

|         |   |
|---------|---|
| E_RSATR | 予約属性<br>・ mematrが無効 【NGKI2820】<br>・ その他の条件については機能の項を参照                      |
| E_NOSPT | 未サポート機能<br>・ 条件については機能の項を参照   |
| E_PAR   | パラメータエラー<br>・ 条件については機能の項を参照  |
| E_OBJ   | オブジェクト状態エラー<br>・ 登録済みのセクションの再登録 【NGKI2821】<br>・ 指定したメモリリージョンが未登録 【NGKI2822】 |

### 【機能】

各パラメータで指定した情報に従って、指定したセクションをカーネルに登録する。具体的な振舞いは以下の通り。

各オブジェクトモジュールに含まれるセクション名で指定したセクションが、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、メモリオブジェクトとして登録される【NGKI2823】。登録されるメモリオブジェクトには、mematrで指定したメモリオブジェクト属性が設定される【NGKI2824】。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つのアクセス許可パターンの組) が、acptn1～acptn4で指定した値に設定される【NGKI2825】。

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属性を指定しても指定しなくても、同じ振舞いとなる) 【NGKI2826】。また、メモリオブジェクト属性のTA\_MEMINI

とTA\_MEMPRSVは無視される（指定しても指定しなくても、同じ振舞いとなる）【NGKI2786】。

mematrに、TA\_MEMINIとTA\_MEMPRSVを同時に指定することはできない。指定した場合には、E\_RSATRエラーとなる【NGKI2828】。

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】。

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～acptn4は整数定数式/パラメータである【NGKI2830】。

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や登録できる数に制限がある場合がある【NGKI2831】。この制限に違反した場合には、E\_NOSPTエラーとなる【NGKI2832】。

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、標準のセクションを指定することはできない。指定した場合には、E\_PARエラーとなる【NGKI2834】。

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。セクション名として指定した場合には、E\_PARエラーとなる【NGKI2836】。

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エラーとなる【NGKI2837】。

### 【μITRON4.0/PX仕様との関係】

μITRON4.0/PX仕様に定義されていない静的APIである。

LNK\_SEC セクションの配置 [SP] 【NGKI2838】

### 【静的API】

```
LNK_SEC("セクション名", { "メモリリージョン名" })
```

### 【パラメータ】

|             |                            |
|-------------|----------------------------|
| "セクション名"    | 配置するセクションを指定する文字列          |
| "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する文字列 |

### 【エラーコード】

|         |   |
|---------|---|
| E_RSATR | 予約属性<br>・保護ドメインの囲みの中に記述されている【NGKI3685】<br>・その他の条件については機能の項を参照           |
| E_PAR   | パラメータエラー<br>・条件については機能の項を参照   |
| E_OBJ   | オブジェクト状態エラー<br>・登録済みのセクションの再登録【NGKI2839】<br>・指定したメモリリージョンが未登録【NGKI2840】 |

### 【機能】

各オブジェクトモジュールに含まれるセクション名で指定したセクションを、メモリリージョン名で指定したメモリリージョンに配置する【NGKI2841】。

セクション名として、標準のセクションや保護ドメイン毎の標準セクションを指定することはできない。指定した場合には、E\_PARエラーとなる【NGKI2843】。

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ラス専用のメモリリージョンの場合で、LNK\_SECをクラスの囲みの外に記述するか、他のクラスの囲みの中に記述した場合には、E\_RSATRエラーとなる【NGKI2844】。

### 【使用上の注意】

LNK\_SECにより配置されたセクションは、メモリオブジェクトとしてカーネルに登録されず、メモリ保護が実現できる先頭番地とサイズになるとは限らない。

### 【μITRON4.0/PX仕様との関係】

μITRON4.0/PX仕様に定義されていない静的APIである。

|         |   |
|---------|---|
| ATT_MOD | オブジェクトモジュールの登録〔SP〕【NGKI2845】                  |
| ATA_MOD | オブジェクトモジュールの登録（アクセス許可ベクタ付き）〔SP〕<br>【NGKI2846】 |

### 【静的API】

```
ATT_MOD("オブジェクトモジュール名")
ATA_MOD("オブジェクトモジュール名",
        { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
```

### 【パラメータ】

"オブジェクトモジュール名" 登録するオブジェクトモジュールを指定する文字列

\*アクセス許可ベクタ (パケットの内容)

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

## 【エラーコード】

|         |   |
|---------|---|
| E_RSATR | 予約属性<br>・mematrが無効 【NGKI2847】                   |
| E_NOSPT | 未サポート機能<br>・条件については機能の項を参照                      |
| E_OBJ   | オブジェクト状態エラー<br>・登録済みのオブジェクトモジュールの再登録 【NGKI2848】 |

## 【機能】

各パラメータで指定した情報に従って、指定したオブジェクトモジュールをカーネルに登録する。具体的な振舞いは以下の通り。

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、メモリオブジェクトとして登録される【NGKI2849】。登録されるメモリオブジェクトには、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クセス許可ベクタ (4つのアクセス許可パターンの組) が、acptn1～acptn4で指定した値に設定される【NGKI2851】。

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】。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、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、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、共通の標準メモリリージョンに配置される場合がある【NGKI3271】。

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】。

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  
パラメータである【NGKI2856】.

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  
保護ドメインや登録できる数に制限がある場合がある【NGKI2857】. この制限  
に違反した場合には， E\_NOSPTエラーとなる【NGKI2858】.

ターゲット定義で， ATT\_MOD／ATA\_MODがサポートされていない場合がある  
ATT\_MOD／ATA\_MODがサポートされている場合には，  
TOPPERS\_SUPPORT\_ATT\_MODがマクロ定義される【NGKI2860】. サポートされてい  
ない場合にATT\_MOD／ATA\_MODを使用すると， コンフィギュレータがE\_NOSPTエラー  
を報告する【NGKI2861】.

#### 【補足説明】

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  
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を用いる必要がある.

#### 【μITRON4.0/PX仕様との関係】

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  
ンと標準  
RAMリージョンであることを明確化した.

|         |   |
|---------|---|
| ATT_MEM | メモリオブジェクトの登録 [SP] 【NGKI2862】              |
| 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】 |
| att_mem | メモリオブジェクトの登録 [TPD] 【NGKI2864】             |

#### 【静的API】

```
ATT_MEM({ ATR mematr, void *base, SIZE size })
ATA_MEM({ ATR mematr, void *base, SIZE size },
         { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
```

#### 【C言語API】

```
ER ercd = att_mem(const T_AMEM *pk_amem)
```

#### 【パラメータ】

T\_AMEM \* pk\_amem メモリオブジェクトの登録情報を入ったパケットへのポインタ（静的APIを除く）

\*メモリオブジェクトの登録情報（パケットの内容）

|        |        |                     |
|--------|--------|---------------------|
| ATR    | mematr | メモリオブジェクト属性         |
| void * | base   | 登録するメモリ領域の先頭番地      |
| SIZE   | size   | 登録するメモリ領域のサイズ（バイト数） |

\*アクセス許可ベクタ（パケットの内容）

|       |        |                  |
|-------|--------|------------------|
| 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び出し [s] 【NGKI2865】<br>・CPUロック状態からの呼び出し [s] 【NGKI2866】  |
| E_RSATR | 予約属性         | ・mematrが無効 【NGKI2867】<br>・属する保護ドメインの指定が有効範囲外 [sP] 【NGKI2868】<br>・属するクラスの指定が有効範囲外 [sM] 【NGKI2869】<br>・その他の条件については機能の項を参照 |
| E_NOSPT | 未サポート機能      | ・条件については機能の項を参照   |
| E_PAR   | パラメータエラー     | ・sizeが0以下 【NGKI2881】<br>・その他の条件については機能の項を参照   |
| E_OACV  | オブジェクトアクセス違反 | ・システム状態に対する管理操作が許可されていない [sP] 【NGKI2870】  |
| E_MACV  | メモリアクセス違反    | ・pk_amemが指すメモリ領域への読み出しが許可されていない [sP] 【NGKI2871】   |
| E_OBJ   | オブジェクト状態エラー  | ・条件については機能の項を参照   |

#### 【機能】

各パラメータで指定したメモリオブジェクト登録情報に従って、メモリオブジェ

クトを登録する。具体的な振舞いは以下の通り。

baseとsizeで指定したメモリ領域が、メモリオブジェクトとして登録される  
【NGKI2872】。登録されるメモリオブジェクトには、mematrで指定したメモリ  
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  
メモリオブジェクトのアクセス許可ベクタ（4つのアクセス許可パターンの組）  
が、acptn1～acptn4で指定した値に設定される【NGKI2874】。

mematrには、TA\_MEMPRSVを指定しなければならず、TA\_MEMINIを指定することは  
できない。TA\_MEMPRSVを指定しない場合や、TA\_MEMINIを指定した場合には、  
エラーとなる【NGKI2876】。また、mematrにTA\_SDATAを指定することは  
TA\_SDATAを指定した場合には、E\_RSATRエラーとなる【NGKI3274】。

静的APIにおいては、mematr、size、acptn1～acptn4は整数定数式パラメータ、  
baseは一般定数式パラメータである【NGKI2877】。

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  
属する保護ドメインや登録できる数に制限がある場合がある【NGKI2878】。こ  
の制限に違反した場合には、E\_NOSPTエラーとなる【NGKI2879】。

baseやsizeに、ターゲット定義の制約に合致しない先頭番地やサイズを指定し  
E\_PARエラーとなる【NGKI2880】。登録しようとしたメモリオブジェ  
クトが、登録済みのメモリオブジェクトとメモリ領域が重なる場合には、  
NGKI2882】。

た時には、

E\_OBJエラーとなる【

### 【使用上の注意】

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  
ようにするために使用することを想定した静的APIである。メモリ領域に対して は、ATT\_SEC／ATA\_SEC  
かATT\_MOD／ATA\_MODを使用することを推奨する。

ATT\_MEM／ATA\_MEMで登録したメモリオブジェクトのメモリ領域が、ATT\_REGで登  
録したメモリリージョンと重なっても、直ちにエラーとはならない。ただし、  
メモリ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  
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エラーとなる。

### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、ATT\_MEMとATA\_MEMのみをサポートする【HRPS0155】。

### 【μITRON4.0/PX仕様との関係】

アクセス許可ベクタを指定してメモリオブジェクトを登録するサービスコール（ata\_mem）は廃止した。

baseやsizeがターゲット定義の制約に合致しない場合、μITRON4.0/PX仕様では  
ターゲット定義の制約に合致するようにメモリ領域を広げることとしていたが、  
E\_PARエラーとなることとした。

この仕様では

|         |                                       |
|---------|---------------------------------------|
| ATT_PMA | 物理メモリ領域の登録〔SP〕【NGKI2883】              |
| 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】 |
| att_pma | 物理メモリ領域の登録〔TPD〕【NGKI2885】             |

### 【静的API】

```
ATT_PMA({ ATR mematr, void *base, SIZE size, void *paddr })
ATA_PMA({ ATR mematr, void *base, SIZE size, void *paddr },
         { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 })
```

### 【C言語API】

```
ER ercd = att_pma(const T_APMA *pk_apma)
```

### 【パラメータ】

T\_APMA \* pk\_apma 物理メモリ領域の登録情報を入ったパケットへのポインタ（静的APIを除く）

\*物理メモリ領域の登録情報（パケットの内容）

|        |        |                            |
|--------|--------|----------------------------|
| ATR    | mematr | メモリオブジェクト属性                |
| void * | base   | 登録するメモリ領域の先頭番地             |
| SIZE   | size   | 登録するメモリ領域のサイズ（バイト数）        |
| void * | paddr  | 登録するメモリ領域の物理アドレス空間における先頭番地 |

\*アクセス許可ベクタ（パケットの内容）

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

### 【リターンパラメータ】

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

### 【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し〔s〕【NGKI2886】<br>・CPUロック状態からの呼出し〔s〕【NGKI2887】                                      |
| E_RSATR | 予約属性<br>・mematrが無効<br>・属する保護ドメインの指定が有効範囲外〔sP〕【NGKI2888】<br>・属するクラスの指定が有効範囲外〔sM〕【NGKI2889】<br>・その他の条件については機能の項を参照 |
| E_NOSPT | 未サポート機能<br>・条件については機能の項を参照   |
| E_PAR   | パラメータエラー<br>・sizeが0以下【NGKI2901】<br>・その他の条件については機能の項を参照   |
| E_OACV  | オブジェクトアクセス違反<br>・システム状態に対する管理操作が許可されていない〔sP〕<br>【NGKI2890】   |
| E_MACV  | メモリアクセス違反<br>・pk_apmaが指すメモリ領域への読み出しが許可されて<br>いない〔sP〕【NGKI2891】   |
| E_OBJ   | オブジェクト状態エラー<br>・条件については機能の項を参照   |

## 【機能】

各パラメータで指定した物理メモリ領域の登録情報に従って、メモリオブジェクトを登録する。具体的な振舞いは以下の通り。

物理アドレス空間において先頭番地がpaddr、サイズがsizeのメモリ領域が、論理アドレス空間においてbaseで指定した番地からアクセスできるように、メモリオブジェクトとして登録される【NGKI2892】。登録されるメモリオブジェクトには、mematrで指定したメモリオブジェクト属性が設定される【NGKI2893】。  
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つのアクセス許可パターンの組）が、acptn1～acptn4で指定した値に設定される【NGKI2894】。

mematrには、TA\_MEMPRSVを指定しなければならず、TA\_MEMINIを指定することはできない。TA\_MEMPRSVを指定しない場合や、TA\_MEMINIを指定した場合には、E\_RSATRエラーとなる【NGKI2896】。

静的APIにおいては、mematr、size、paddr、acptn1～acptn4は整数定数式パラメータ、baseは一般定数式パラメータである【NGKI2897】。

ターゲット定義で、ATT\_PMA／ATA\_PMAにより登録できるメモリオブジェクトが属する保護ドメインや登録できる数に制限がある場合がある【NGKI2898】。この制限に違反した場合には、E\_NOSPTエラーとなる【NGKI2899】。

base、size、paddrに、ターゲット定義の制約に合致しない先頭番地やサイズをE\_PARエラーとなる【NGKI2900】。登録しようとしたメモリオブジェクトには、

ブジェクトが、登録済みのメモリオブジェクトと論理アドレス空間においてメモリ領域が重なる場合には、E\_OBJエラーとなる【NGKI2902】。

ATT\_PMA／ATA\_PMA／att\_pmaは、MMU（Memory Management Unit）を持つターゲットシステムにおいて、ターゲット定義でサポートされる機能である【NGKI2903】。ATT\_PMA／ATA\_PMA／att\_pmaがサポートされている場合には、TOPPERS\_SUPPORT\_ATT\_PMAがマクロ定義される【NGKI2904】。ATT\_PMA／ATA\_PMAがサポートされていない場合にこれらの静的APIを使用すると、コンフィギュレータがE\_NOSPTエラーを報告する【NGKI2905】。また、att\_pmaがサポートされていない場合にatt\_pmaを呼び出すと、E\_NOSPTエラーが返るか、リンク時にエラーとなる【NGKI2906】。

#### 【TOPPERS/HRP2カーネルにおける規定】

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【HRPS0156】。

#### 【μITRON4.0/PX仕様との関係】

μITRON4.0/PX仕様に定義されていない静的APIおよびサービスコールである。

sac\_mem メモリオブジェクトのアクセス許可ベクタの設定〔TPD〕【NGKI2907】

#### 【C言語API】

```
ER ercd = sac_mem(const void *base, const ACVCT *p_acvct)
```

#### 【パラメータ】

|         |         |                         |
|---------|---------|-------------------------|
| void *  | base    | メモリオブジェクトの先頭番地          |
| ACVCT * | p_acvct | アクセス許可ベクタを入れたパケットへのポインタ |

|                      |        |                  |
|----------------------|--------|------------------|
| * アクセス許可ベクタ（パケットの内容） |        |                  |
| ACPTN                | acptn1 | 通常操作1のアクセス許可パターン |
| ACPTN                | acptn2 | 通常操作2のアクセス許可パターン |
| ACPTN                | acptn3 | 管理操作のアクセス許可パターン  |
| ACPTN                | acptn4 | 参照操作のアクセス許可パターン  |

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI2908】<br>・CPUロック状態からの呼出し【NGKI2909】 |
| E_PAR   | パラメータエラー<br>・baseがメモリオブジェクトの先頭番地でない【NGKI2910】                         |
| E_NOEXS | オブジェクト未登録<br>・baseで指定した番地を含むメモリオブジェクトが登録されていない【NGKI2911】              |
| E_OACV  | オブジェクトアクセス違反<br>・対象メモリオブジェクトに対する管理操作が許可されていない【NGKI2912】               |
| E_MACV  | メモリアクセス違反<br>・p_acvctが指すメモリ領域への読み出しが許可されていない【NGKI2913】                |
| E_OBJ   | オブジェクト状態エラー<br>・対象メモリオブジェクトは静的APIで登録された【NGKI2914】                     |

## 【機能】

baseで指定したメモリオブジェクト（対象メモリオブジェクト）のアクセス許可ベクタ（4つのアクセス許可パターンの組）を、各パラメータで指定した値に設定する【NGKI2915】。

## 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、sac\_memをサポートしない【HRPS0157】。

## 【μITRON4.0/PX仕様との関係】

静的APIによって登録したメモリオブジェクトは、アクセス許可ベクタを設定することができないこととした。

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ものとしていたが、この仕様では、メモリオブジェクトの先頭番地でなければならないものとした。

det\_mem メモリオブジェクトの登録解除〔TPD〕【NGKI2916】

## 【C言語API】

```
ER ercd = det_mem(const void *base)
```

## 【パラメータ】

|        |      |                |
|--------|------|----------------|
| void * | base | メモリオブジェクトの先頭番地 |
|--------|------|----------------|

## 【リターンパラメータ】

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

## 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼び出し 【NGKI2917】<br>・CPUロック状態からの呼び出し 【NGKI2918】 |
| E_PAR   | パラメータエラー<br>・baseがメモリオブジェクトの先頭番地でない 【NGKI2919】                            |
| E_NOEXS | オブジェクト未登録<br>・baseで指定した番地を含むメモリオブジェクトが登録されていない 【NGKI2920】                 |
| E_OACV  | オブジェクトアクセス違反<br>・対象メモリオブジェクトに対する管理操作が許可されていない 【NGKI2921】                  |
| E_OBJ   | オブジェクト状態エラー<br>・対象メモリオブジェクトは静的APIで登録された 【NGKI2922】                        |

## 【機能】

baseで指定したメモリオブジェクト（対象メモリオブジェクト）を登録解除する 【NGKI2923】。

## 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、det\_memをサポートしない 【HRPS0158】。

## 【μITRON4.0/PX仕様との関係】

静的APIによって登録したメモリオブジェクトは、登録を解除することができないこととした。

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ものとしていたが、この仕様では、メモリオブジェクトの先頭番地でなければならないものとした。

prb\_mem メモリ領域に対するアクセス権のチェック [TP] 【NGKI2924】

## 【C言語API】

```
ER ercd = prb_mem(const void *base, SIZE size, ID tskid, MODE pmmode)
```

## 【パラメータ】

|        |        |                 |
|--------|--------|-----------------|
| void * | base   | メモリ領域の先頭番地      |
| SIZE   | size   | メモリ領域のサイズ（バイト数） |
| ID     | tskid  | アクセス元のタスクのID番号  |
| MODE   | pmmode | アクセスモード         |

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼び出し【NGKI2925】  |
| E_ID    | 不正ID番号<br>・tskidが有効範囲外【NGKI2927】   |
| E_PAR   | パラメータエラー<br>・sizeが0【NGKI2929】<br>・その他の条件については機能の項を参照   |
| E_NOEXS | オブジェクト未登録<br>・baseで指定した番地を含むメモリオブジェクトが登録され<br>ていない【NGKI2930】<br>・tskidで指定したタスクが未登録〔D〕【NGKI3425】                      |
| E_OACV  | オブジェクトアクセス違反<br>・対象メモリ領域を含むメモリオブジェクトに対する参照操<br>作が許可されていない【NGKI2931】<br>・tskidで指定したタスクに対する参照操作が許可されていな<br>い【NGKI3426】 |
| E_MACV  | メモリアクセス違反<br>・条件については機能の項を参照   |
| E_OBJ   | オブジェクト状態エラー<br>・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<br>【NGKI2932】   |

#### 【機能】

tskidで指定したタスクから、baseとsizeで指定したメモリ領域（対象メモリ領域）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クする。アクセスが許可されている場合にE\_OK、そうでない場合にE\_MACVが返る【NGKI2933】。tskidで指定したタスクがカーネルドメインに属する場合、E\_MACVが返ることはない【NGKI2934】。

pmmodeには、TPM\_WRITE（=0x01U）、TPM\_READ（=0x02U）、TPM\_EXEC（=0x04U）のいずれか、またはそれらの内のいくつかのビット毎論理和（C言語の"|"）を指定することができる【NGKI2935】。TPM\_WRITE、TPM\_READ、TPM\_EXECを指定した場合には、それぞれ、読み出しあクセス、書き込みアクセス、実行ア

セスが許可されているかをチェックする【NGKI2936】。また、いくつかのビット毎論理和を指定した場合には、それらに対応した種別のアクセスがすべて許可されているかをチェックする【NGKI2937】。pmmodeにそれ以外の値を指定した場合には、E\_PARエラーとなる【NGKI2938】。

tskidにTSK\_SELF (=0) を指定すると、自タスクから対象メモリ領域に対してアクセスが許可されているかをチェックする【NGKI2939】。

#### 【μITRON4.0/PX仕様との関係】

アクセスする主体の指定方法を、保護ドメインによる指定 (domid) から、タスクによる指定 (tskid) に変更した。また、pmmodeに指定できるアクセス種別に TPM\_EXECを追加し、TPM\_WRITEとTPM\_READの値を入れ換えた。CPUロック状態からも呼び出せるものとした。

#### 【仕様決定の理由】

prb\_memを、CPUロック状態からも呼び出せるものとしたのは、次の理由による。prb\_memは、拡張サービスコールの中で、タスクから渡されたポインタが、そのタスクからアクセスできる領域であるかを調べるために用いることを想定している。拡張サービスコールの中には、CPUロック状態でも呼び出せるものがあり、そのような拡張サービスコールを実現するには、prb\_memがCPUロック状態から呼び出せることが必要である。

なお、prb\_memを非タスクコンテキストから呼び出すことはできないが、非タスクコンテキストで実行される処理単位は必ずカーネルドメインに属するために、prb\_memを使ってアクセス権を調べる必要がないことから、支障がない。

ref\_mem メモリオブジェクトの状態参照 [TP]

#### 【C言語API】

```
ER ercd = ref_mem(const void *base, T_RMEM *pk_rmem)
```

未完成

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、ref\_memをサポートしない。

## 1.9. 割込み管理機能

割込み処理のプログラムは、割込みサービスルーチン (ISR) として実現することを推奨する。割込みサービスルーチンをカーネルに登録する場合には、まず、割込みサービスルーチンの登録対象となる割込み要求ラインの属性を設定して NGKI2940】。割込みサービスルーチンは、カーネル内の割込おく必要がある【

みハンドラを経由して呼び出される【NGKI2941】。

ただし、カーネルが用意する割込みハンドラで対応できないケースに対応するために、アプリケーションで割込みハンドラを用意することも可能である【NGKI2942】。この場合にも、割込みハンドラをカーネルに登録する前に、割込みハンドラの登録対象となる割込みハンドラ番号に対応する割込み要求ラインの属性を設定しておく必要がある【NGKI2943】。

割込み要求ラインの属性を設定する際に指定する割込み要求ライン属性には、次の属性を指定することができる【NGKI2944】。

|           |       |                |
|-----------|-------|----------------|
| TA_ENAINT | 0x01U | 割込み要求禁止フラグをクリア |
| TA_EDGE   | 0x02U | エッジトリガ         |

ターゲットによっては、ターゲット定義の割込み要求ライン属性を指定できる【NGKI2945】。ターゲット定義の割込み要求ライン属性として、次の属性を予約している【NGKI2946】。

|              |                                     |
|--------------|-------------------------------------|
| TA_POSEDGE   | ポジティブエッジトリガ                         |
| TA_NEGEDGE   | ネガティブエッジトリガ                         |
| TA_BOTHEDGE  | 両エッジトリガ                             |
| TA_LOWLEVEL  | ローレベルトリガ                            |
| TA_HIGHLEVEL | ハイレベルトリガ                            |
| TA_BROADCAST | すべてのプロセッサで割込みを処理（マルチプロセッサ対応カーネルの場合） |

割込みサービスルーチンは、カーネルが実行を制御する処理単位である。割込みサービスルーチンは、割込みサービスルーチンIDと呼ぶID番号によって識別する【NGKI2947】。

1つの割込み要求ラインに対して複数の割込みサービスルーチンを登録した場合、それらの割込みサービスルーチンは、割込みサービスルーチン優先度の高い順にすべて呼び出される【NGKI2948】。割込みサービスルーチン優先度が同じ場合には、登録した順（静的APIにより登録した場合には、割込みサービスルーチンを生成するAPIをコンフィギュレーションファイル中に記述した順）で呼び出される【NGKI2949】。

保護機能対応カーネルにおいて、割込みサービスルーチンが属することのできる保護ドメインは、カーネルドメインに限られる【NGKI2950】。

割込みサービスルーチン属性に指定できる属性はない【NGKI2951】。そのため割込みサービスルーチン属性には、TA\_NULLを指定しなければならない【NGKI2952】。

C言語による割込みサービスルーチンの記述形式は次の通り【NGKI2953】。

【

```
void interrupt_service_routine(intptr_t exinf)
{
    割込みサービスルーチン本体
}
```

exinfには、割込みサービスルーチンの拡張情報が渡される【NGKI2954】。

割込みハンドラは、カーネルが実行を制御する処理単位である。割込みハンドラは、割込みハンドラ番号と呼ぶオブジェクト番号によって識別する【NGKI2955】。

保護機能対応カーネルにおいて、割込みハンドラは、カーネルドメインに属する【NGKI2956】。

割込みハンドラを登録する際に指定する割込みハンドラ属性には、ターゲット定義で、次の属性を指定することができる【NGKI2957】。

|              |       |             |
|--------------|-------|-------------|
| TA_NONKERNEL | 0x02U | カーネル管理外の割込み |
|--------------|-------|-------------|

TA\_NONKERNELを指定しない場合、カーネル管理の割込みとなる【NGKI2958】。

また、ターゲットによっては、その他のターゲット定義の割込みハンドラ属性を指定できる場合がある【NGKI2959】。

C言語による割込みハンドラの記述形式は次の通り【NGKI2960】。

```
void interrupt_handler(void)
{
    割込みハンドラ本体
}
```

割込み管理機能に関連するカーネル構成マクロは次の通り。

|             |                           |
|-------------|---------------------------|
| TMIN_INTPRI | 割込み優先度の最小値（最高値）【NGKI2961】 |
| TMAX_INTPRI | 割込み優先度の最大値（最低値、=-1）       |

|             |                                  |
|-------------|----------------------------------|
| TMIN_ISRPRI | 割込みサービスルーチン優先度の最小値（=1）【NGKI2962】 |
| TMAX_ISRPRI | 割込みサービスルーチン優先度の最大値               |

|                         |                             |
|-------------------------|-----------------------------|
| TOPPERS_SUPPORT_DIS_INT | dis_intがサポートされている【NGKI2963】 |
| TOPPERS_SUPPORT_ENA_INT | ena_intがサポートされている【NGKI2964】 |

### 【使用上の注意】

1つの割込み要求ラインに複数のデバイスからの割込み要求が接続されている場合に対応するために、割込みサービスルーチンは、それが処理する割込み要求が発生しているかをチェックし、割込み要求が発生していない場合には何もせずにリターンするように実装すべきである。

#### 【TOPPERS/ASPカーネルにおける規定】

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) は16に固定されている【ASPS0192】。ただし、タスク優先度拡張パッケージで は、TMAX\_ISRPRIを256に拡張する【ASPS0193】。

#### 【TOPPERS/FMPカーネルにおける規定】

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) は16に固定されている【FMPS0159】。

#### 【TOPPERS/HRP2カーネルにおける規定】

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) は16に固定されている【HRPS0159】。

#### 【TOPPERS/SSPカーネルにおける規定】

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) は16に固定されている【SSPS0137】。

#### 【μITRON4.0仕様との関係】

割込み要求ラインの属性、割込み優先度、割込みサービスルーチン優先度は、  
μITRON4.0仕様にない概念であり、TMIN\_INTPRI、TMAX\_INTPRI、TMIN\_ISRPRI、 TMAX\_ISRPRI  
は、μITRON4.0仕様に定義のないカーネル構成マクロである。また、 TA\_NONKERNELは、  
μITRON4.0仕様に定義のない割込みハンドラ属性である。

|         |                               |
|---------|-------------------------------|
| CFG_INT | 割込み要求ラインの属性の設定〔S〕 【NGKI2965】  |
| cfg_int | 割込み要求ラインの属性の設定〔TD〕 【NGKI2966】 |

#### 【静的API】

```
CFG_INT(INTNO intno, { ATR intatr, PRI intpri })
```

#### 【C言語API】

```
ER ercd = cfg_int(INTNO intno, const T_CINT *pk_cint)
```

#### 【パラメータ】

|          |         |  |
|----------|---------|--|
| INTNO    | intno   | 割込み番号                                    |
| T_CINT * | pk_cint | 割込み要求ラインの属性の設定情報を入ったパケットへのポインタ（静的APIを除く） |

\*割込み要求ラインの属性の設定情報（パケットの内容）

|     |        |            |
|-----|--------|------------|
| ATR | intatr | 割込み要求ライン属性 |
| PRI | intpri | 割込み優先度     |

### 【リターンパラメータ】

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

### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー                                       |
|         | ・非タスクコンテキストからの呼出し [s] 【NGKI2967】                |
|         | ・CPUロック状態からの呼出し [s] 【NGKI2968】                  |
| E_RSATR | 予約属性  |
|         | ・intatrが無効 【NGKI2969】                           |
|         | ・属するクラスの指定が有効範囲外 [sM] 【NGKI2970】                |
|         | ・クラスの囲みの中に記述されていない [sM] 【NGKI2971】              |
|         | ・その他の条件については機能の項を参照                             |
| E_PAR   | パラメータエラー  |
|         | ・intnoが有効範囲外 【NGKI2972】                         |
|         | ・intpriが有効範囲外 【NGKI2973】                        |
|         | ・その他の条件については機能の項を参照                             |
| E_OACV  | オブジェクトアクセス違反                                    |
|         | ・システム状態に対する管理操作が許可されていない [sP] 【NGKI2974】        |
| E_MACV  | メモリアクセス違反                                       |
|         | ・pk_cintが指すメモリ領域への読み出しが許可されていない [sP] 【NGKI2975】 |
| E_OBJ   | オブジェクト状態エラー                                     |
|         | ・対象割込み要求ラインに対して属性が設定済み [s] 【NGKI2976】           |
|         | ・その他の条件については機能の項を参照                             |

### 【機能】

intnoで指定した割込み要求ライン（対象割込み要求ライン）に対して、各パラメータで指定した属性を設定する【NGKI2977】。

対象割込み要求ラインの割込み要求禁止フラグは、intatrにTA\_ENAINTを指定した場合にクリアされ、指定しない場合にセットされる【NGKI2978】。

静的APIにおいては、intno, intatr, intpriは整数定数式パラメータである【NGKI2979】。

`cfg_int`において、ターゲット定義で、複数の割込み要求ラインの割込み優先度が連動して設定される場合がある【NGKI2980】。

`intpri`に指定できる値は、基本的には、`TMIN_INTPRI`以上、`TMAX_INTPRI`以下の値である【NGKI2981】。ターゲット定義の拡張で、カーネル管理外の割込み要求ラインに対しても属性を設定できる場合には、`TMIN_INTPRI`よりも小さい値を指定することができる【NGKI2982】。このように拡張されている場合、カーネル管理外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み要求ラインを対象として、`intpri`が`TMIN_INTPRI`よりも小さい値である場合にも、`E_OBJ`エラーとなる【NGKI2984】。

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_E_RSATR`エラー、設定できない割込み優先度を`intpri`に指定した場合には`E_PAR`エラーとなる【NGKI2985】。ここで、設定できない割込み要求ライン属性／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`E_PAR`エラーとなる【NGKI2986】。また、マルチプロセッサ対応カーネルにおいて、`cfg_int`を呼び出したタスクが割り付けられているプロセッサから、対象割込み要求ラインの属性を設定できない場合も、これに該当する【NGKI2987】。

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【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ドではないため、割込み要求ライン属性に`TA_DOM(domid)`を指定した場合には`E_RSATR`エラーとなる【NGKI2990】。ただし、`TA_DOM(TDOM_SELF)`を指定した場合には、指定が無視され、`E_RSATR`エラーは検出されない【NGKI2991】。

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み中にある場合には、`E_RSATR`エラーとなる【NGKI2992】。

#### 【補足説明】

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／割込み優先度は、主にターゲットハードウェアの制限から来るものである。例えば、対象割込み要求ラインに対して、トリガモードや割込み優先度が固定されていて、変更できないケースが考えられる。

`cfg_int`において、ターゲット定義で、複数の割込み要求ラインの割込み優先度が連動して設定されるのは、ターゲットハードウェアの制限により、異なる割込み要求ラインに対して、同一の割込み優先度しか設定できないケースに対応するための仕様である。この場合、`CFG_INT`においては、同一の割込み優先度しか設定できない割込み要求ラインに対して異なる割込み優先度を設定した場合には、`E_PAR`エラーとなる。

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、`CFG_INT`のみをサポートする【ASPS0194】。

## 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、CFG\_INTのみをサポートする【FMPS0160】。

## 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、CFG\_INTのみをサポートする【HRPS0160】。

## 【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、CFG\_INTのみをサポートする【SSPS0138】。

## 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていない静的APIおよびサービスコールである。

|          |                                |
|----------|--------------------------------|
| CRE_ISR  | 割込みサービスルーチンの生成 [S] 【NGKI2993】  |
| ATT_ISR  | 割込みサービスルーチンの追加 [S] 【NGKI2994】  |
| acre_isr | 割込みサービスルーチンの生成 [TD] 【NGKI2995】 |

## 【静的API】

```
CRE_ISR(ID isrid, { ATR israttr, intptr_t exinf,
                      INTNO intno, ISR isr, PRI isrpri })
ATT_ISR({ ATR israttr, intptr_t exinf, INTNO intno, ISR isr, PRI isrpri })
```

## 【C言語API】

```
ER_ID isrid = acre_isr(const T_CISR *pk_cisr)
```

## 【パラメータ】

|          |         |   |
|----------|---------|---|
| ID       | isrid   | 対象割込みサービスルーチンのID番号 (CRE_ISR の場合)          |
| T_CISR * | pk_cisr | 割込みサービスルーチンの生成情報を入れたパケットへのポインタ (静的APIを除く) |

\*割込みサービスルーチンの生成情報 (パケットの内容)

|          |         |                       |
|----------|---------|-----------------------|
| ATR      | israttr | 割込みサービスルーチン属性         |
| intptr_t | exinf   | 割込みサービスルーチンの拡張情報      |
| INTNO    | intno   | 割込みサービスルーチンを登録する割込み番号 |
| ISR      | isr     | 割込みサービスルーチンの先頭番地      |
| PRI      | isrpri  | 割込みサービスルーチン優先度        |

## 【リターンパラメータ】

|       |       |                                     |
|-------|-------|-------------------------------------|
| ER_ID | isrid | 生成された割込みサービスルーチンのID番号（正の値）またはエラーコード |
|-------|-------|-------------------------------------|

## 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し [s] 【NGKI2996】<br>・CPUロック状態からの呼出し [s] 【NGKI2997】   |
| E_RSATR | 予約属性<br>・isratrが無効 【NGKI2998】<br>・属する保護ドメインの指定が有効範囲外またはカーネルドメイン以外 [sP] 【NGKI2999】<br>・カーネルドメインの囲みの中に記述されていない [sP] 【NGKI3000】<br>・属するクラスの指定が有効範囲外 [sM] 【NGKI3001】<br>・クラスの囲みの中に記述されていない [sM] 【NGKI3002】<br>・その他の条件については機能の項を参照 |
| E_PAR   | パラメータエラー<br>・intnoが有効範囲外 【NGKI3003】<br>・isrがプログラムの先頭番地として正しくない 【NGKI3004】<br>・isrpriが有効範囲外 【NGKI3005】   |
| E_OACV  | オブジェクトアクセス違反<br>・システム状態に対する管理操作が許可されていない [sP] 【NGKI3006】  |
| E_MACV  | メモリアクセス違反<br>・pk_cisrが指すメモリ領域への読み出しが許可されていない [sP] 【NGKI3007】  |
| E_NOID  | ID番号不足<br>・割り付けられる割込みサービスルーチンIDがない [sD] 【NGKI3008】  |
| E_OBJ   | オブジェクト状態エラー<br>・isridで指定した割込みサービスルーチンが登録済み (CRE_ISRの場合) 【NGKI3009】<br>・その他の条件については機能の項を参照   |

## 【機能】

各パラメータで指定した割込みサービスルーチン生成情報に従って、割込みサービスルーチンを生成する【NGKI3010】。

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【NGKI3011】。

intnoで指定した割込み要求ラインの属性が設定されていない場合には、E\_OBJ エラーとなる【NGKI3012】。また、intnoで指定した割込み番号に対応する割込

みハンドラ番号に対して、割込みハンドラを定義する機能（DEF\_INH, def\_inh）によって割込みハンドラが定義されている場合にも、E\_OBJエラーとなる【NGKI3013】。さらに、intnoでカーネル管理外の割込みを指定した場合にも、E\_OBJエラーとなる【NGKI3014】。

静的APIにおいては、isridはオブジェクト識別名、isratr, intno, isrpriは整数定数式パラメータ、exinfとisrは一般定数式パラメータである【NGKI3015】。

マルチプロセッサ対応カーネルで、生成する割込みサービスルーチンの属するクラスの割付け可能プロセッサが、intnoで指定した割込み要求ラインが接続されたプロセッサの集合に含まれていない場合には、E\_RSATRエラーとなる【NGKI3016】。また、intnoで指定した割込み要求ラインに対して登録済みの割込みサービスルーチンがある場合に、生成する割込みサービスルーチンがそれと異なるクラスに属する場合にも、E\_RSATRエラーとなる【NGKI3017】。さらに、ターゲット定義で、割込みサービスルーチンが属することができるクラスに制限がある場合がある【NGKI3018】。生成する割込みサービスルーチンの属するクラスが、ターゲット定義の制限に合致しない場合にも、E\_RSATRエラーとなる【NGKI3019】。

静的APIにおいて、isrが不正である場合にE\_PARエラーが検出されるか否かは、ターゲット定義である【NGKI3020】。

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、ATT\_ISRのみをサポートする【ASPS0209】。ただし、動的生成機能拡張パッケージでは、acre\_isrもサポートする【ASPS0195】。

#### 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、ATT\_ISRのみをサポートする【FMPS0161】。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、ATT\_ISRのみをサポートする【HRPS0161】。ただし、動的生成機能拡張パッケージでは、acre\_isrもサポートする【HRPS0208】。

#### 【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、ATT\_ISRのみをサポートする【SSPS0139】。

#### 【μITRON4.0仕様との関係】

割込みサービスルーチンの生成情報に、isrpri（割込みサービスルーチンの割込み優先度）を追加した。CRE\_ISRは、μITRON4.0仕様に定義されていない静的APIである。

AID\_ISR 割付け可能な割込みサービスルーチンIDの数の指定 [SD] 【NGKI3021】

#### 【静的API】

```
AID_ISR(uint_t noisr)
```

#### 【パラメータ】

|        |       |                       |
|--------|-------|-----------------------|
| uint_t | noisr | 割付け可能な割込みサービスルーチンIDの数 |
|--------|-------|-----------------------|

#### 【エラーコード】

|         |          |  |
|---------|----------|--|
| E_RSATR | 予約属性     | ・保護ドメインの囲みの中に記述されている [P] 【NGKI3439】<br>・クラスの囲みの中に記述されていない [M] 【NGKI3022】 |
| E_PAR   | パラメータエラー | ・noisrが負の値 【NGKI3287】  |

#### 【機能】

noisrで指定した数の割込みサービスルーチンIDを、割込みサービスルーチンを生成するサービスコールによって割付け可能な割込みサービスルーチンIDとして確保する【NGKI3024】。

noisrは整数定数式パラメータである【NGKI3025】。

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルの動的生成機能拡張パッケージでは、AID\_ISRをサポートする【ASPS0219】。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルの動的生成機能拡張パッケージでは、AID\_ISRをサポートする【HRPS0220】。

|         |                                |            |
|---------|--------------------------------|------------|
| SAC_ISR | 割込みサービスルーチンのアクセス許可ベクタの設定 [SP]  | 【NGKI3026】 |
| sac_isr | 割込みサービスルーチンのアクセス許可ベクタの設定 [TPD] | 【NGKI3027】 |

#### 【静的API】

```
SAC_ISR(ID isrid, { ACPTN acptn1, ACPTN acptn2,  
                      ACPTN acptn3, ACPTN acptn4 })
```

#### 【C言語API】

```
ER ercd = sac_isr(ID isrid, const ACVCT *p_acvct)
```

#### 【パラメータ】

|         |         |                                   |
|---------|---------|-----------------------------------|
| ID      | isrid   | 対象割込みサービスルーチンのID番号                |
| ACVCT * | p_acvct | アクセス許可ベクタを入れたパケットへのポインタ（静的APIを除く） |

\* アクセス許可ベクタ（パケットの内容）

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

### 【リターンパラメータ】

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

### 【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼び出し [s] 【NGKI3028】<br>・CPUロック状態からの呼び出し [s] 【NGKI3029】                                  |
| E_ID    | 不正ID番号<br>・isridが有効範囲外 [s] 【NGKI3030】  |
| E_RSATR | 予約属性<br>・カーネルドメインの囲みの中に記述されていない [S] 【NGKI3031】<br>・対象割込みサービスルーチンが属するクラスの囲みの中に<br>記述されていない [SM] 【NGKI3032】          |
| E_NOEXS | オブジェクト未登録<br>・対象割込みサービスルーチンが未登録 【NGKI3033】   |
| E_OACV  | オブジェクトアクセス違反<br>・対象割込みサービスルーチンに対する管理操作が許可され<br>ていない) [s] 【NGKI3034】  |
| E_MACV  | メモリアクセス違反<br>・p_acvctが指すメモリ領域への読み出しが許可されて<br>いない) [s] 【NGKI3035】   |
| E_OBJ   | オブジェクト状態エラー<br>・対象割込みサービスルーチンは静的APIで生成された [s]<br>【NGKI3036】<br>・対象割込みサービスルーチンに対してアクセス許可ベクタ<br>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/HRP2カーネルにおける規定】

HRP2カーネルでは、SAC\_ISR, sac\_isrをサポートしない【HRPS0162】。ただし、動的生成機能拡張パッケージでは、sac\_isrをサポートする【HRPS0209】。

#### 【未決定事項】

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セス許可ベクタでアクセス保護する方法も考えられる。

del\_isr 割込みサービスルーチンの削除 [TD] 【NGKI3040】

#### 【C言語API】

```
ER ercd = del_isr(ID_isrid)
```

#### 【パラメータ】

|    |       |                    |
|----|-------|--------------------|
| ID | isrid | 対象割込みサービスルーチンのID番号 |
|----|-------|--------------------|

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼び出し 【NGKI3041】<br>・CPUロック状態からの呼び出し 【NGKI3042】 |
| E_ID    | 不正ID番号<br>・isridが有効範囲外 【NGKI3043】   |
| E_NOEXS | オブジェクト未登録<br>・対象割込みサービスルーチンが未登録 【NGKI3044】                                |
| E_OACV  | オブジェクトアクセス違反<br>・対象割込みサービスルーチンに対する管理操作が許可されていない [P] 【NGKI3045】            |
| E_OBJ   | オブジェクト状態エラー<br>・対象割込みサービスルーチンは静的APIで生成された 【NGKI3046】                      |

#### 【機能】

isridで指定した割込みサービスルーチン（対象割込みサービスルーチン）を削

除する。具体的な振舞いは以下の通り。

対象割込みサービスルーチンの登録が解除され、その割込みサービスルーチンIDが未使用の状態に戻される【NGKI3047】。

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、`del_isr`をサポートしない【ASPS0197】。ただし、動的生成機能拡張パッケージでは、`del_isr`をサポートする【ASPS0198】。

#### 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、`del_isr`をサポートしない【FMP0163】。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、`del_isr`をサポートしない【HRPS0163】。ただし、動的生成機能拡張パッケージでは、`del_isr`をサポートする【HRPS0210】。

#### 【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、`del_isr`をサポートしない【SSPS0141】。

`ref_isr` 割込みサービスルーチンの状態参照 [T]

#### 【C言語API】

```
ER ercd = ref_isr(ID_isrid, T_RISR *pk_risr)
```

未完成

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、`ref_isr`をサポートしない。

#### 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、`ref_isr`をサポートしない。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、`ref_isr`をサポートしない。

#### 【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、`ref_isr`をサポートしない。

**DEF\_INH** 割込みハンドラの定義 [S] 【NGKI3048】  
**def\_inh** 割込みハンドラの定義 [TD] 【NGKI3049】

【静的API】

```
DEF_INH(INHNO inhno, { ATR inhatr, INTHDR inthdr })
```

【C言語API】

```
ER ercd = def_inh(INHNO inhno, const T_DINH *pk_dinh)
```

【パラメータ】

|                 |                |                                      |
|-----------------|----------------|--------------------------------------|
| <b>INHNO</b>    | <b>inhno</b>   | 割込みハンドラ番号                            |
| <b>T_DINH *</b> | <b>pk_dinh</b> | 割込みハンドラの定義情報を入ったパケットへのポインタ（静的APIを除く） |

\*割込みハンドラの定義情報（パケットの内容）

|               |               |              |
|---------------|---------------|--------------|
| <b>ATR</b>    | <b>inhatr</b> | 割込みハンドラ属性    |
| <b>INTHDR</b> | <b>inthdr</b> | 割込みハンドラの先頭番地 |

【リターンパラメータ】

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

【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し [s] 【NGKI3050】<br>・CPUロック状態からの呼出し [s] 【NGKI3051】   |
| E_RSATR | 予約属性<br>・inhattrが無効 【NGKI3052】<br>・属するクラスの指定が有効範囲外 [sM] 【NGKI3053】<br>・クラスの囲みの中に記述されていない [SM] 【NGKI3054】<br>・その他の条件については機能の項を参照 |
| E_PAR   | パラメータエラー<br>・inhnoが有効範囲外 【NGKI3055】<br>・inthdrがプログラムの先頭番地として正しくない 【NGKI3056】<br>・その他の条件については機能の項を参照                             |
| E_OACV  | オブジェクトアクセス違反<br>・システム状態に対する管理操作が許可されていない [sP] 【NGKI3057】  |
| E_MACV  | メモリアクセス違反<br>・pk_dinhが指すメモリ領域への読み出しが許可されていない [sP] 【NGKI3058】  |
| E_OBJ   | オブジェクト状態エラー<br>・条件については機能の項を参照  |

## 【機能】

inhnoで指定した割込みハンドラ番号（対象割込みハンドラ番号）に対して、各パラメータで指定した割込みハンドラ定義情報に従って、割込みハンドラを定義する【NGKI3059】。ただし、def\_inhにおいてpk\_dinhをNULLにした場合には、対象割込みハンドラ番号に対する割込みハンドラの定義を解除する【NGKI3060】。

静的APIにおいては、inhnoとinhattrは整数定数式パラメータ、inthdrは一般定数式パラメータである【NGKI3061】。

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pk\_dinhをNULL以外にした場合）には、次のエラーが検出される。

対象割込みハンドラ番号に対応する割込み要求ラインの属性が設定されていない場合には、E\_OBJエラーとなる【NGKI3062】。また、対象割込みハンドラ番号に対してすでに割込みハンドラが定義されている場合と、対象割込みハンドラ番号に対応する割込み番号を対象に割込みサービスルーチンが登録されている場合にも、E\_OBJエラーとなる【NGKI3063】。

ターゲット定義の拡張で、カーネル管理外の割込みに対しても割込みハンドラを定義できる場合には、次のエラーが検出される【NGKI3064】。カーネル管理外の割込みハンドラを対象として、inhattrにTA\_NONKERNELを指定しない場合にエラーとなる【NGKI3065】。逆に、カーネル管理の割込みハンドラをTA\_NONKERNELを指定した場合にも、E\_OBJエラーとなる【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

は、E\_OBJ対象として、inhattr

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、inhattrにTA\_NONKERNELを指定しない場合には、E\_RSATRエラーとなる【NGKI3067】。逆に、ターゲット定義でカーネル管理に固定されている割込みハンドラがある場合には、それを対象割込みハンドラに指定して、inhattrにTA\_NONKERNELを指定した場合には、E\_RSATRエラーとなる【NGKI3068】。

保護機能対応カーネルにおいて、DEF\_INHは、カーネルドメインの囲みの中に記述しなければならない。そうでない場合には、E\_RSATRエラーとなる【NGKI3070】。また、def\_inhで割込みハンドラを定義する場合には、割込みハンドラの属する保護ドメインを設定する必要はなく、割込みハンドラ属性にTA\_DOM(domid)を指定した場合にはE\_RSATRエラーとなる【NGKI3071】。ただし、TA\_DOM(TDOM\_SELF)を指定した場合には、指定が無視され、E\_RSATRエラーは検出されない【NGKI3072】。

マルチプロセッサ対応カーネルで、登録する割込みハンドラの属するクラスの初期割付けプロセッサが、その割込みが要求されるプロセッサでない場合には、E\_RSATRエラーとなる【NGKI3073】。また、ターゲット定義で、割込みハンドラ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】。登録する割込みハンドラの属するクラスが、ターゲット定義の制限に合致しない場合にも、E\_RSATRエラーとなる【NGKI3075】。

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いE\_OBJエラーとなる【NGKI3076】。また、対象割込みハンドラ番号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は、ターゲット定義でE\_OBJエラーとなる場合がある【NGKI3077】。

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）できない場合には、カーネルにおいて、E\_PARエラーとなる【NGKI3078】。具体的には、マルチプロセッサ対応def\_inhを呼び出したタスクが割り付けられているプロセッサから、対象割込みハンドラを定義（または定義解除）できない場合が、これに該当する【NGKI3079】。

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は、ターゲット定義である【NGKI3080】。

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、DEF\_INHのみをサポートする【ASPS0199】。

#### 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、DEF\_INHのみをサポートする【FMPS0164】。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、DEF\_INHのみをサポートする【HRPS0164】。

## 【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、DEF\_INHのみをサポートする【SSPS0142】。

## 【μITRON4.0仕様との関係】

inthdrのデータ型をINTHDRに変更した。

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、  
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、一度定義を解除してから、再度定義する必要がある。

dis\_int 割込みの禁止 [T] 【NGKI3081】

## 【C言語API】

```
ER ercd = dis_int(INTNO intno)
```

## 【パラメータ】

|       |       |       |
|-------|-------|-------|
| INTNO | intno | 割込み番号 |
|-------|-------|-------|

## 【リターンパラメータ】

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

## 【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼び出し 【NGKI3082】                   |
| E_NOSPT | 未サポートエラー<br>・条件については機能の項を参照                                  |
| E_PAR   | パラメータエラー<br>・intnoが有効範囲外 【NGKI3083】<br>・その他の条件については機能の項を参照   |
| E_OACV  | オブジェクトアクセス違反<br>・システム状態に対する通常操作2が許可されていない [P]<br>【NGKI3084】  |
| E_OBJ   | オブジェクト状態エラー<br>・対象割込み要求ラインに対して割込み要求ライン属性が設定されていない 【NGKI3085】 |

## 【機能】

intnoで指定した割込み要求ライン（対象割込み要求ライン）の割込み要求禁止フラグをセットする【NGKI3086】.

ターゲット定義で、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E\_PARエラーとなる【NGKI3087】. 具体的には、対象割込み要求ラインに対して割込み要求禁止フラグがサポートされていない場合や、マルチプロセッサ対応カーネルにおいて、dis\_intを呼び出したタスクが割り付けられているプロセッサから、対象割込み要求ラインの割込み要求禁止フラグが操作できない場合が、これに該当する。

ターゲット定義で、割込み要求禁止フラグの振舞いが、この仕様の規定と異なるNGKI3089】. 特にマルチプロセッサ対応カーネルでは、あるプロセッサがdis\_intを呼び出して割込み要求禁止フラグをセットしても、他のプロセッサに対しては割込みがマスクされない場合がある。

ターゲット定義で、dis\_intがサポートされていない場合がある【NGKI3091】. dis\_intがサポートされている場合には、TOPPERS\_SUPPORT\_DIS\_INTがマクロ定義される【NGKI3092】. サポートされていない場合にdis\_intを呼び出すと、E\_NOSPTエラーが返るか、リンク時にエラーとなる【NGKI3093】.

#### 【μITRON4.0仕様との関係】

μITRON4.0仕様で実装定義としていたintnoの意味を標準化した。

CPUロック状態でも呼び出せるものとした。

ena\_int 割込みの許可 [T] 【NGKI3094】

#### 【C言語API】

```
ER ercd = ena_int(INTNO intno)
```

#### 【パラメータ】

|       |       |       |
|-------|-------|-------|
| INTNO | intno | 割込み番号 |
|-------|-------|-------|

#### 【リターンパラメータ】

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

#### 【エラーコード】

きない場合には、

る場合がある【  
口セッサから

義される【

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI3095】                    |
| E_NOSPT | 未サポートエラー<br>・条件については機能の項を参照                                 |
| E_PAR   | パラメータエラー<br>・intnoが有効範囲外【NGKI3096】<br>・その他の条件については機能の項を参照   |
| E_OACV  | オブジェクトアクセス違反<br>・システム状態に対する通常操作2が許可されていない〔P〕<br>【NGKI3097】  |
| E_OBJ   | オブジェクト状態エラー<br>・対象割込み要求ラインに対して割込み要求ライン属性が設定されていない【NGKI3098】 |

### 【機能】

intnoで指定した割込み要求ライン（対象割込み要求ライン）の割込み要求禁止フラグをクリアする【NGKI3099】.

ターゲット定義で、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  
E\_PARエラーとなる【NGKI3100】. 具体的には、対象割込み要  
求ラインに対して割込み要求禁止フラグがサポートされていない場合や、マル  
チプロセッサ対応カーネルにおいて、ena\_intを呼び出したタスクが割り付けら  
れているプロセッサから、対象割込み要求ラインの割込み要求禁止フラグが操  
作できない場合が、これに該当する。

ターゲット定義で、割込み要求禁止フラグの振舞いが、この仕様の規定と異なる  
NGKI3102】. 特にマルチプロセッサ対応カーネルでは、あるプ  
ena\_intを呼び出して割込み要求禁止フラグをクリアしても、他の  
プロセッサに対しては割込みがマスク解除されない場合がある。

ターゲット定義で、ena\_intがサポートされていない場合がある【NGKI3104】.  
ena\_intがサポートされている場合には、TOPPERS\_SUPPORT\_ENA\_INTがマクロ定  
NGKI3105】. サポートされていない場合にena\_intを呼び出すと、  
E\_NOSPTエラーが返るか、リンク時にエラーとなる【NGKI3106】.

きない場合には、

る場合がある【  
口セッサから

義される【

### 【μITRON4.0仕様との関係】

μITRON4.0仕様で実装定義としていたintnoの意味を標準化した。

CPUロック状態でも呼び出せるものとした。

ref\_int 割込み要求ラインの参照〔T〕

### 【C言語API】

```
ER ercd = ref_int(INTNO intno, T_RINT *pk_rint)
```

未完成

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、ref\_intをサポートしない。

#### 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、ref\_intをサポートしない。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、ref\_intをサポートしない。

#### 【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、ref\_intをサポートしない。

#### 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである。

```
chg_ipm    割込み優先度マスクの変更 [T] 【NGKI3107】
```

#### 【C言語API】

```
ER ercd = chg_ipm(PRI intpri)
```

#### 【パラメータ】

|     |        |           |
|-----|--------|-----------|
| PRI | intpri | 割込み優先度マスク |
|-----|--------|-----------|

#### 【リターンパラメータ】

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

#### 【エラーコード】

|        |   |
|--------|---|
| E_CTX  | コンテキストエラー<br>・非タスクコンテキストからの呼出し【NGKI3108】<br>・CPUロック状態からの呼出し【NGKI3109】 |
| E_PAR  | パラメータエラー<br>・条件については機能の項を参照   |
| E_OACV | オブジェクトアクセス違反<br>・システム状態に対する通常操作2が許可されていない〔P〕<br>【NGKI3110】            |

### 【機能】

割込み優先度マスクを、intpriで指定した値に変更する【NGKI3111】。

intpriは、TMIN\_INTPRI以上、TIPM\_ENAALL以下でなければならない。そうでない場合には、E\_PARエラーとなる【NGKI3113】。ただし、ターゲット定義の拡張TMIN\_INTPRIよりも小さい値を指定できる場合がある【NGKI3114】。

として、

### 【補足説明】

割込み優先度マスクをTIPM\_ENAALLに変更した場合、ディスパッチ保留状態が解除され、ディスパッチが起こる可能性がある。また、タスク例外処理ルーチンの実行が開始される可能性がある。

### 【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、chg\_ipmをサポートしない【SSPS0143】。

### 【μITRON4.0仕様との関係】

μITRON4.0仕様では、サービスコールの名称およびパラメータの名称が実装定義となっているサービスコールである。

get\_ipm 割込み優先度マスクの参照〔T〕【NGKI3115】

### 【C言語API】

```
ER ercd = get_ipm(PRI *p_intpri)
```

### 【パラメータ】

|       |          |                          |
|-------|----------|--------------------------|
| PRI * | p_intpri | 割込み優先度マスクを入れるメモリ領域へのポインタ |
|-------|----------|--------------------------|

### 【リターンパラメータ】

|     |        |           |
|-----|--------|-----------|
| ER  | ercd   | エラーコード    |
| PRI | intpri | 割込み優先度マスク |

### 【エラーコード】

|        |   |
|--------|---|
| E_CTX  | コンテキストエラー<br>・非タスクコンテキストからの呼出し 【NGKI3116】<br>・CPUロック状態からの呼出し 【NGKI3117】 |
| E_OACV | オブジェクトアクセス違反<br>・システム状態に対する参照操作が許可されていない [P]<br>【NGKI3118】              |
| E_MACV | メモリアクセス違反<br>・p_intpriが指すメモリ領域への書き込みアクセスが許可され<br>ていない [P] 【NGKI3119】    |

### 【機能】

割込み優先度マスクの現在値を参照する。参照した割込み優先度マスクは、p\_intpriが指すメモリ領域に返される【NGKI3120】。

### 【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、get\_ipmをサポートしない【SSPS0144】。

### 【μITRON4.0仕様との関係】

μITRON4.0仕様では、サービスコールの名称およびパラメータの名称が実装定義となっているサービスコールである。

## 1.10. CPU例外管理機能

CPU例外ハンドラは、カーネルが実行を制御する処理単位である。CPU例外ハンドラ番号と呼ぶオブジェクト番号によって識別する【NGKI3121】。

ドラは、

保護機能対応カーネルにおいて、CPU例外ハンドラは、カーネルドメインに属する【NGKI3122】。

CPU例外ハンドラ属性に標準で指定できる属性はないが、ターゲットによっては、ターゲット定義のCPU例外ハンドラ属性を指定できる場合がある【NGKI3123】。

CPU例外ハンドラ属性として、次の属性を予約している【NGKI3124】。

ターゲット定義の  
ターゲット定義の

|           |                  |
|-----------|------------------|
| TA_DIRECT | CPU例外ハンドラを直接呼び出す |
|-----------|------------------|

C言語によるCPU例外ハンドラの記述形式は次の通り【NGKI3125】。

```
void cpu_exception_handler(void *p_excinf)
{
    CPU例外ハンドラ本体
}
```

p\_excinfには、CPU例外の情報を記憶しているメモリ領域の先頭番地が渡される。これは、CPU例外ハンドラ内で、CPU例外発生時の状態を参照する際に必要となる。

DEF\_EXC CPU例外ハンドラの定義 [S] 【NGKI3127】  
def\_exc CPU例外ハンドラの定義 [TD] 【NGKI3128】

#### 【静的API】

```
DEF_EXC(EXCNO excno, { ATR excatr, EXCHDR exchdr })
```

#### 【C言語API】

```
ER ercd = def_exc(EXCNO excno, const T_DEXC *pk_dexc)
```

#### 【パラメータ】

|                          |         |  |
|--------------------------|---------|--|
| EXCNO                    | excno   | CPU例外ハンドラ番号                            |
| T_DEXC *                 | pk_dexc | CPU例外ハンドラの定義情報を入ったパケットへのポインタ（静的APIを除く） |
| *CPU例外ハンドラの定義情報（パケットの内容） |         |  |
| ATR                      | excatr  | CPU例外ハンドラ属性                            |
| EXCHDR                   | exchdr  | CPU例外ハンドラの先頭番地                         |

#### 【リターンパラメータ】

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

#### 【エラーコード】

|         |   |
|---------|---|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼出し [s] 【NGKI3129】<br>・CPUロック状態からの呼出し [s] 【NGKI3130】   |
| E_RSATR | 予約属性<br>・excattrが無効 【NGKI3131】<br>・属するクラスの指定が有効範囲外 [sM] 【NGKI3132】<br>・クラスの囲みの中に記述されていない [sM] 【NGKI3133】<br>・その他の条件については機能の項を参照 |
| E_PAR   | パラメータエラー<br>・excnoが有効範囲外 【NGKI3134】<br>・exchdrがプログラムの先頭番地として正しくない 【NGKI3135】  |
| E_OACV  | オブジェクトアクセス違反<br>・システム状態に対する管理操作が許可されていない [sP]<br>【NGKI3136】   |
| E_MACV  | メモリアクセス違反<br>・pk_dexcが指すメモリ領域への読み出しが許可されて<br>いない [sP] 【NGKI3137】  |
| E_OBJ   | オブジェクト状態エラー<br>・条件については機能の項を参照  |

## 【機能】

excnoで指定したCPU例外ハンドラ番号（対象CPU例外ハンドラ番号）に対して、各パラメータで指定したCPU例外ハンドラ定義情報に従って、CPU例外ハンドラ【NGKI3138】。ただし、def\_excにおいてpk\_dexcをNULLにした場合を定義する【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【NGKI3139】。

静的APIにおいては、excnoとexcattrは整数定数式パラメータ、exchdrは一般定数式パラメータである【NGKI3140】。

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】。

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【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TA\_DOM(domid)を指定した場合にはE\_RSATRエラーとなる【NGKI3144】。ただし、TA\_DOM(TDOM\_SELF)を指定した場合には、指定が無視され、E\_RSATRエラーは検出されない【NGKI3145】。

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、E\_RSATRエラーとなる【NGKI3146】。

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  
エラーとなる【NGKI3147】。また、対象CPU例外ハンドラ  
CPU例外ハンドラが、静的APIで定義されたものである  
E\_OBJエラーとなる場合がある【NGKI3148】。

ない場合には、E\_OBJ  
番号に対して定義された  
場合には、ターゲット定義で

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  
は、ターゲット定義である【NGKI3149】。

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、DEF\_EXCのみをサポートする【ASPS0200】。

#### 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、DEF\_EXCのみをサポートする【FMPS0165】。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、DEF\_EXCのみをサポートする【HRPS0165】。

#### 【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、DEF\_EXCのみをサポートする【SSPS0145】。

#### 【μITRON4.0仕様との関係】

def\_excによって、定義済みのCPU例外ハンドラを再定義しようとした場合に、  
E\_OBJエラーとすることにした。

xsns\_dpn CPU例外発生時のディスパッチ保留状態の参照〔TI〕【NGKI3150】

#### 【C言語API】

```
bool_t stat = xsns_dpn(void *p_excinf)
```

#### 【パラメータ】

void \* p\_excinf CPU例外の情報を記憶しているメモリ領域の先頭  
番地

#### 【リターンパラメータ】

bool\_t state ディスパッチ保留状態

## 【機能】

CPU例外発生時のディスパッチ保留状態を参照する。具体的な振舞いは以下の通り。

実行中のCPU例外ハンドラの起動原因となったCPU例外が、カーネル管理外のCPU例外でなく、タスクコンテキストで発生し、そのタスクがディスパッチ保留状態でなかった場合にfalse、そうでない場合にtrueが返る【NGKI3151】。

保護機能対応のカーネルにおいて、xsns\_dpnをタスクコンテキストから呼び出しが返る【NGKI3152】。

p\_excinfには、CPU例外ハンドラに渡されるp\_excinfパラメータをそのまま渡す【NGKI3153】。それ以外の値を渡した場合の動作は保証されない【NGKI3552】。

## 【使用方法】

xsns\_dpnは、CPU例外ハンドラの中で、どのようなリカバリ処理が可能かを判別したい場合に使用する。xsns\_dpnがfalseを返した場合（trueを返した場合ではないので注意すること）、非タスクコンテキスト用のサービスコールを用いてCPU例外を起こしたタスクよりも優先度の高いタスクを起動または待ち解除し、そのタスクでリカバリ処理を行うことができる。ただし、CPU例外を起こしたタスクが最高優先度の場合には、この方法でリカバリ処理を行うことはできない。

## 【使用上の注意】

xsns\_dpnは、E\_CTXエラーを返すがないために〔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が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。した場合や、

どちらの条件でtrueが返るか間違いや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】。

## 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである。

## 【仕様決定の理由】

保護機能対応のカーネルにおいては、xsns\_dpnをユーザドメインから呼び出すことは禁止すべきである。ユーザドメインの実行中は、必ずタスクコンテキストであるため、xsns\_dpnをタスクコンテキストから呼び出した場合に必ずtrueを返す仕様とすることで、xsns\_dpnをユーザドメインから呼び出すことを実質的に禁止している。

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】

|        |          |                               |
|--------|----------|-------------------------------|
| void * | p_excinf | CPU例外の情報を記憶しているメモリ領域の先頭<br>番地 |
|--------|----------|-------------------------------|

### 【リターンパラメータ】

|        |       |             |
|--------|-------|-------------|
| bool_t | state | 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と、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した場合には、trueが返る 【NGKI3157】。

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 【NGKI3158】。

### 【使用方法】

xsns\_xpnは、CPU例外ハンドラの中で、どのようなリカバリ処理が可能かを判別したい場合に使用する。xsns\_xpnがfalseを返した場合（trueを返した場合ではないので注意すること）、非タスクコンテキスト用のサービスコールを用いてCPU例外を起こしたタスクにタスク例外を要求し、タスク例外処理ルーチンでリカバリ処理を行うことができる。

### 【使用上の注意】

xsns\_xpnは、E\_CTXエラーを返すことがないために〔TI〕となっているが、CPU例外ハンドラから呼び出すためのものである。CPU例外ハンドラ以外から呼び出p\_excinfに正しい値を渡さなかった場合、xsns\_xpnが返す値は意味を持たない。

どちらの条件でtrueが返るか間違いやさるので注意すること。

#### 【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、xsns\_xpnをサポートしない【SSPS0147】。

#### 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていないサービスコールである。

#### 【仕様決定の理由】

保護機能対応のカーネルにおいては、xsns\_xpnをユーザドメインから呼び出すことは禁止すべきである。ユーザドメインの実行中は、必ずタスクコンテキスト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 xsns\_xpnをユーザドメインから呼び出すことを実質的に禁止している。

トであるため、  
を返す仕様とすることで、

## 1.11. 拡張サービスコール管理機能

拡張サービスコールは、非特権モードで実行される処理単位から、特権モードで実行すべきルーチンを呼び出すための機能である【NGKI3159】。特権モードで実行するルーチンを、拡張サービスコールと呼ぶ。拡張サービスコールは、特権モードで実行される処理単位からも呼び出すことができる【NGKI3160】。

保護機能対応カーネルにおいて、拡張サービスコールは、カーネルドメインに属する【NGKI3161】。拡張サービスコールは、それを呼び出す処理単位とは別の処理単位であり、拡張サービスコールからカーネルオブジェクトをアクセスする場合には、拡張サービスコールがアクセスの主体となる【NGKI3162】。そのため、拡張サービスコールからは、すべてのカーネルオブジェクトに対して、すべての種別のアクセスを行うことが許可される。

保護機能対応でないカーネルでは、非特権モードと特権モードの区別がないため、拡張サービスコール管理機能をサポートしない【NGKI3163】。

拡張サービスコール属性に指定できる属性はない【NGKI3686】。そのため拡張サービスコール属性には、TA\_NULLを指定しなければならない【NGKI3687】。

C言語による拡張サービスコールの記述形式は次の通り【NGKI3164】。

```
ER_UINT extended_svc(intptr_t par1, intptr_t par2, intptr_t par3,  
                      intptr_t par4, intptr_t par5, ID cdmid)  
{  
    拡張サービスコール本体  
}
```

cdmidには、拡張サービスコールを呼び出した処理単位が属する保護ドメインの

ID

番号が渡される【NGKI3165】。すなわち、拡張サービスコールから呼び出し  
TDOM\_KERNEL (=1) が、タスク本体（拡張サービスコールを除く）  
から呼び出した場合にはそのタスク（自タスク）の属する保護ドメインIDが渡される。  
た場合には

par1～par5には、拡張サービスコールに対するパラメータが渡される【NGKI3166】。

拡張サービスコール管理機能に関連するカーネル構成マクロは次の通り。

|           |  |
|-----------|--|
| TMAX_FNCD | 拡張サービスコールの機能番号の最大値（動的生成対応<br>カーネルでは、登録できる拡張サービスコールの数に一<br>致）【NGKI3167】 |
|-----------|--|

#### 【TOPPERS/ASPカーネルにおける規定】

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】。

#### 【TOPPERS/FMPカーネルにおける規定】

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】。

#### 【TOPPERS/HRP2カーネルにおける規定】

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】。

#### 【TOPPERS/SSPカーネルにおける規定】

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】。

#### 【未決定事項】

動的生成対応カーネルにおいてTMAX\_FNCDを設定する方法については、現時点では未決定である。

#### 【μITRON4.0仕様との関係】

この仕様では、拡張サービスコールに対するパラメータを、intptr\_t型のパラメータ5個に固定した。

拡張サービスコールに、それを呼び出した処理単位が属する保護ドメインのID番号を渡す機能を追加した。

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

|         |                             |
|---------|-----------------------------|
| DEF_SVC | 拡張サービスコールの定義〔SP〕【NGKI3168】  |
| def_svc | 拡張サービスコールの定義〔TPD〕【NGKI3169】 |

#### 【静的API】

```
DEF_SVC(FN_fncd, { ATR_svcatr, EXTSVC_extsvc, SIZE_stksz })
```

## 【C言語API】

```
ER ercd = def_svc(FN fncl, const T_DSVC *pk_dsvc)
```

### 【パラメータ】

|          |         |  |
|----------|---------|--|
| FN       | fncl    | 拡張サービスコールの機能コード                        |
| T_DSVC * | pk_dsvc | 拡張サービスコールの定義情報を入ったパケットへのポインタ（静的APIを除く） |

\* 拡張サービスコールの定義情報（パケットの内容）

|        |        |                       |
|--------|--------|-----------------------|
| ATR    | svcatr | 拡張サービスコール属性           |
| EXTSVC | extsvc | 拡張サービスコールの先頭番地        |
| SIZE   | stksz  | 拡張サービスコールで使用するスタックサイズ |

### 【リターンパラメータ】

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

### 【エラーコード】

|         |  |
|---------|--|
| E_CTX   | コンテキストエラー<br>・非タスクコンテキストからの呼び出し [s] 【NGKI3170】<br>・CPUロック状態からの呼び出し [s] 【NGKI3171】  |
| E_RSATR | 予約属性<br>・svcatrが無効 【NGKI3172】<br>・その他の条件については機能の項を参照   |
| E_PAR   | パラメータエラー<br>・fnclが0または負の値 【NGKI3173】<br>・fnclがTMAX_FNCDよりも大きい [s] 【NGKI3174】<br>・extsvcがプログラムの先頭番地として正しくない 【NGKI3175】<br>・stkszが負の値 [s] 【NGKI3290】 |
| E_OACV  | オブジェクトアクセス違反<br>・システム状態に対する管理操作が許可されていない [s] 【NGKI3176】  |
| E_MACV  | メモリアクセス違反<br>・pk_dsvcが指すメモリ領域への読み出しが許可されていない [s] 【NGKI3177】  |
| E_OBJ   | オブジェクト状態エラー<br>・条件については機能の項を参照   |

### 【機能】

fnclで指定した機能コード（対象機能コード）に対して、各パラメータで指定

した拡張サービスコール定義情報に従って、拡張サービスコールを定義する】。ただし、def\_svcにおいてpk\_dsvcをNULLにした場合には、対象機能コードに対する拡張サービスコールの定義を解除する【NGKI3179】。

【NGKI3178

静的APIにおいては、fnacd, svcatr, stkszは整数定数式パラメータ、svchdrは一般定数式パラメータである【NGKI3180】。

拡張サービスコールを定義する場合（DEF\_SVCの場合およびdef\_svcにおいてNULL以外にした場合）で、対象機能コードに対してすでに拡張サービスコールが定義されている場合には、E\_OBJエラーとなる【NGKI3181】。

pk\_dsvcを

DEF\_SVCは、カーネルドメインの囲みの中に記述しなければならない。そうでない場合には、  
E\_RSATRエラーとなる【NGKI3183】。また、def\_svcで拡張サービスコールを定義する場合には、拡張サービスコールの属する保護ドメインを設定する必要はなく、拡張サービスコール属性にTA\_DOM(domid)を指定した場合にはE\_RSATRエラーとなる【NGKI3184】。ただし、TA\_DOM(TDOM\_SELF)を指定した場合には、指定が無視され、E\_RSATRエラーは検出されない【NGKI3185】。

い場合には、

マルチプロセッサ対応カーネルでは、DEF\_SVCは、クラスの囲みの外に記述しなければならない。そうでない場合には、E\_RSATRエラーとなる【NGKI3187】。また、def\_svcで拡張サービスコールを定義する場合には、拡張サービスコールの属するクラスを設定する必要はなく、拡張サービスコール属性にTA\_CLS(clsid)を指定した場合にはE\_RSATRエラーとなる【NGKI3188】。ただし、TA\_CLS(TCLS\_SELF)を指定した場合には、指定が無視され、E\_RSATRエラーは検出されない【NGKI3189】。

た、

出されない【

拡張サービスコールの定義を解除する場合（def\_svcにおいてpk\_dsvcをNULLにした場合）で、対象機能コードに対して拡張サービスコールが定義されていない場合には、E\_OBJエラーとなる【NGKI3190】。

い場合には、

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、DEF\_SVCのみをサポートする【HRPS0167】。

#### 【μITRON4.0仕様との関係】

拡張サービスコールの定義情報に、stksz（拡張サービスコールで使用するスタックサイズ）を追加した。

extsvcのデータ型を、EXTSVCに変更した。

cal\_svc 拡張サービスコールの呼出し [TIP] 【NGKI3191】

#### 【C言語API】

```
ER_UINT ercd = cal_svc(FN_fnCD, intptr_t par1, intptr_t par2,  
                        intptr_t par3, intptr_t par4, intptr_t par5)
```

### 【パラメータ】

|          |      |                     |
|----------|------|---------------------|
| FN       | fnCD | 呼び出す拡張サービスコールの機能コード |
| intptr_t | par1 | 拡張サービスコールへの第1/パラメータ |
| intptr_t | par2 | 拡張サービスコールへの第2/パラメータ |
| intptr_t | par3 | 拡張サービスコールへの第3/パラメータ |
| intptr_t | par4 | 拡張サービスコールへの第4/パラメータ |
| intptr_t | par5 | 拡張サービスコールへの第5/パラメータ |

### 【リターンパラメータ】

|         |      |                        |
|---------|------|------------------------|
| ER_UINT | ercd | 正常終了（正の値または0）またはエラーコード |
|---------|------|------------------------|

### 【エラーコード】

|         |  |
|---------|--|
| E_SYS   | システムエラー<br>・条件については機能の項を参照   |
| E_RSFN  | 予約機能コード<br>・fnCDが0または負の値【NGKI3192】<br>・fnCDがTMAX_FNCDよりも大きい【NGKI3193】<br>・fnCDで指定した機能コードに対して拡張サービスコールが定義されていない【NGKI3194】 |
| E_NOMEM | メモリ不足<br>・条件については機能の項を参照   |

\*その他、拡張サービスコールが返すエラーコードがそのまま返る。

### 【機能】

fnCDで指定した機能コードの拡張サービスコールを、par1, par2, ..., par5をパラメータとして呼び出し、拡張サービスコールの返値を返す【NGKI3195】。

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【NGKI3196】。スタック（ユーザタスクの場合はシステムスタック）の残り領域が、拡張サービスコールで使用するスタックサイズよりも小さい場合には、エラーとなる【NGKI3197】。また、拡張サービスコールのネストレベルが上限（=255）を超える場合には、E\_SYSエラーが返る【NGKI3198】。

### 【μITRON4.0仕様との関係】

μITRON4.0仕様では、cal\_svcでカーネルのサービスコールを呼び出せるかどうかは実装定義としているが、この仕様では、カーネルのサービスコールを呼び出せないこととした。

【

E\_NOMEM  
が上限（=

拡張サービスコールが呼び出される時に、スタックの残り領域のサイズをチェックする機能を追加した。

拡張サービスコールに対するパラメータを、intptr\_t型のパラメータ5個に固定  
から返るエラー（E\_SYS, E\_RSFN, E\_NOMEM）について規定した。  
し、cal\_svc

#### 【仕様決定の理由】

パラメータの型と数を固定したのは、型チェックを厳密にできるようにし、パ  
ラメータをコンパイラやコーリングコンベンションによらずに正しく渡せるようにするためである。

## 1.12. システム構成管理機能

システム構成管理機能には、非タスクコンテキスト用スタック領域を設定する  
機能、初期化ルーチンと終了処理ルーチンを登録する機能、カーネルのコンフィ  
ギュレーション情報やバージョン情報を参照する機能が含まれる。

非タスクコンテキスト用スタック領域は、非タスクコンテキストで実行される  
処理単位が用いるスタック領域である。

保護機能対応カーネルにおいて、非タスクコンテキスト用のスタック領域は、  
カーネルの用いるオブジェクト管理領域と同様に扱われる【NGKI3199】。

初期化ルーチンは、カーネルが実行を制御する処理単位で、カーネルの動作開  
始の直前に、カーネル非動作状態で実行される【NGKI3200】。

保護機能対応カーネルにおいて、初期化ルーチンは、カーネルドメインに属する【NGKI3201】。

初期化ルーチン属性に指定できる属性はない【NGKI3202】。そのため初期化ルー  
チん属性には、  
TA\_NULLを指定しなければならない【NGKI3203】。

C言語による初期化ルーチンの記述形式は次の通り【NGKI3204】。

```
void initialization_routine(intptr_t exinf)
{
    初期化ルーチン本体
}
```

exinfには、初期化ルーチンの拡張情報が渡される【NGKI3205】。

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  
終了の直後に、カーネル非動作状態で実行される【NGKI3206】。

保護機能対応カーネルにおいて、終了処理ルーチンは、カーネルドメインに属する【NGKI3207】。

終了処理ルーチン属性に指定できる属性はない【NGKI3208】。そのため終了処  
理ルーチン属性には、TA\_NULLを指定しなければならない【NGKI3209】。

C言語による終了処理ルーチンの記述形式は次の通り【NGKI3210】.

```
void termination_routine(intptr_t exinf)
{
    終了処理ルーチン本体
}
```

exinfには、終了処理ルーチンの拡張情報が渡される【NGKI3211】.

#### 【μITRON4.0仕様との関係】

非タスクコンテキスト用スタック領域の設定と、終了処理ルーチンは、  
μITRON4.0仕様に規定されていない機能である。

LMT\_DOM 保護ドメインに対する制限の設定 [SP] 【NGKI3441】

#### 【静的API】

```
LMT_DOM({ PRI mintpri })
```

#### 【パラメータ】

\*保護ドメインに対する制限の設定情報

PRI            mintpri        指定できる最高のタスク優先度

#### 【エラーコード】

|         |             |   |
|---------|-------------|---|
| E_RSATR | 予約属性        | <ul style="list-style-type: none"><li>・ユーザドメインの囲みの中に記述されていない【NGKI3442】</li><li>・クラスの囲みの中に記述されている〔M〕【NGKI3443】</li></ul> |
| E_OBJ   | オブジェクト状態エラー | <ul style="list-style-type: none"><li>・保護ドメインに対する制限が設定済み【NGKI3444】</li></ul>  |
| E_PAR   | パラメータエラー    | <ul style="list-style-type: none"><li>・mintpriが有効範囲外【NGKI3445】</li></ul>  |

#### 【機能】

パラメータで指定した保護ドメインに対する制限の設定情報に従って、ユーザ  
ドメインに対する制限を設定する【NGKI3446】.

mintpriは整数定数式パラメータである【NGKI3447】.

LMT\_DOMにより保護ドメインに対する制限を設定しないユーザドメインに対して

は、指定できる最高のタスク優先度はTMIN\_TPRI+1に設定される【NGKI3448】。

### 【μITRON4.0/PX仕様との関係】

μITRON4.0/PX仕様に定義されていない静的APIである。

DEF\_ICS 非タスクコンテキスト用スタック領域の設定 [S] 【NGKI3212】

### 【静的API】

DEF\_ICS({ SIZE istksz, STK\_T \*istk })

### 【パラメータ】

\*非タスクコンテキスト用スタック領域の設定情報

SIZE istksz 非タスクコンテキスト用スタック領域のサイズ  
(バイト数)

STK\_T istk 非タスクコンテキスト用スタック領域の先頭番地

### 【エラーコード】

E\_RSATR 予約属性

- ・カーネルドメインの囲みの中に記述されていない [P] 【NGKI3213】
- ・クラスの囲みの中に記述されていない [M] 【NGKI3214】

E\_PAR パラメータエラー

- ・条件については機能の項を参照

E\_NOMEM メモリ不足

- ・非タスクコンテキスト用スタック領域が確保できない【NGKI3215】

E\_OBJ オブジェクト状態エラー

- ・非タスクコンテキスト用スタック領域が設定済み【NGKI3216】
- ・その他の条件については機能の項を参照

### 【機能】

各パラメータで指定した非タスクコンテキスト用スタック領域の設定情報に従つて、非タスクコンテキスト用スタック領域を設定する【NGKI3217】。istkszに0以下の値を指定した時や、ターゲット定義の最小値よりも小さい値を指定したE\_PARエラーとなる【NGKI3254】。

時には、

istkszは整数定数式パラメータ、istkは一般定数式パラメータである。コンフィギュレータは、静的APIのメモリ不足(E\_NOMEM)エラーを検出することができない【NGKI3218】。

ギュレータは、静的

istkをNULLとした場合、istkszで指定したサイズのスタック領域が、コンフィギュレータにより確保される【NGKI3219】。istkszにターゲット定義の制約に

合致しないサイズを指定した時には、ターゲット定義の制約に合致するように  
大きい方に丸めたサイズで確保される【NGKI3220】.

istkにNULL以外を指定した場合、istkとistkszで指定したスタック領域は、ア  
プリケーションで確保しておく必要がある【NGKI3221】. スタック領域をアプ  
リケーションで確保する方法については、「2.15.3 カーネル共通マクロ」の節  
を参照すること。その方法に従わず、istkやistkszにターゲット定義の制約に  
合致しない先頭番地やサイズを指定した時には、E\_PARエラーとなる【NGKI3222】.

保護機能対応カーネルでは、istkとistkszで指定した非タスクコンテキスト用  
のスタック領域がカーネル専用のメモリオブジェクトに含まれない場合、  
E\_OBJエラーとなる【NGKI3223】.

DEF\_ICSにより非タスクコンテキスト用スタック領域を設定しない場合、ターゲッ  
ト定義のデフォルトのサイズのスタック領域が、コンフィギュレータにより確保される【NGKI3224】.

マルチプロセッサ対応カーネルでは、非タスクコンテキスト用スタック領域は  
プロセッサ毎に確保する必要がある【NGKI3225】. DEF\_ICSにより設定する非タ  
スクコンテキスト用スタック領域は、DEF\_ICSの記述をその囲みの中に含むクラ  
スの初期割付けプロセッサが使用する【NGKI3226】. そのプロセッサに対して  
すでに非タスクコンテキスト用スタック領域が設定されている場合には、  
E\_OBJエラーとなる【NGKI3227】.

#### 【TOPPERS/SSPカーネルにおける規定】

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  
フィギュレータは非タスクコンテキスト用のスタック領域を確保しない

【SSPS0149】. これは、

SSPカーネルでは、すべての処理単位が共有スタック領

域を使用し、非タスクコンテキストのみが用いるスタック領域を持たないため

である。そのため、

DEF\_ICSの役割は、非タスクコンテキストが用いるスタック

領域のサイズを指定することのみとなる。itsk

にNULL以外を指定した場合には、E\_PARエラーとなる【SSPS0150】.

共有スタック領域の設定方法については、DEF\_STKの項を参照すること。

#### 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていない静的APIである。

DEF\_STK 共有スタック領域の設定 [S] 【NGKI3228】

#### 【静的API】

DEF\_STK({ SIZE stksz, STK\_T \*stk })

#### 【パラメータ】

### \*共有スタック領域の設定情報

|       |       |                    |
|-------|-------|--------------------|
| SIZE  | stksz | 共有スタック領域のサイズ（バイト数） |
| STK_T | stk   | 共有スタック領域の先頭番地      |

### 【エラーコード】

|         |                                     |
|---------|-------------------------------------|
| E_PAR   | パラメータエラー<br>・条件については機能の項を参照         |
| E_NOMEM | メモリ不足<br>・共有スタック領域が確保できない【NGKI3229】 |
| E_OBJ   | オブジェクト状態エラー<br>・共有スタック領域が設定済み       |

### 【サポートするカーネル】

DEF\_STKは、TOPPERS/SSPカーネルのみがサポートする静的APIである。他のカーネルは、DEF\_STKをサポートしない【NGKI3230】。

### 【機能】

各パラメータで指定した共有スタック領域の設定情報に従って、共有スタック領域を設定する【NGKI3231】。stkszに0以下の値を指定した時や、ターゲット定義の最小値よりも小さい値を指定した時には、E\_PARエラーとなる【NGKI3255】。

stkszは整数定数式パラメータ、stkは一般定数式パラメータである。コンフィギュレータは、静的APIのメモリ不足（E\_NOMEM）エラーを検出することができない【NGKI3232】。

stkをNULLとした場合、stkszで指定したサイズのスタック領域が、コンフィギュレータにより確保される【NGKI3233】。stkszにターゲット定義の制約に合致しないサイズを指定した時には、ターゲット定義の制約に合致するように大きい方に丸めたサイズで確保される【NGKI3234】。

stkにNULL以外を指定した場合、stkとstkszで指定したスタック領域は、アプリケーションで確保しておく必要がある【NGKI3235】。スタック領域をアプリケーションで確保する方法については、「2.15.3 カーネル共通マクロ」の節を参照すること。その方法に従わず、stkやstkszにターゲット定義の制約に合致しない先頭番地やサイズを指定した時には、E\_PARエラーとなる【NGKI3236】。

コンフィギュレータは、各タスクのスタック領域のサイズと、非タスクコンテキスト用のスタック領域のサイズから、共有スタック領域に必要なサイズを計算する【NGKI3237】。DEF\_STKにより共有スタック領域を設定しない場合、必要なサイズの共有スタック領域が、コンフィギュレータにより確保される【NGKI3238】。

stkszに指定したスタック領域のサイズが、共有スタック領域に必要なサイズよりも小さい場合、コンフィギュレータは警告メッセージを出力する【NGKI3239】。

## 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていない静的APIである。

ATT\_INI 初期化ルーチンの追加 [S] 【NGKI3240】

## 【静的API】

```
ATT_INI({ ATR iniatr, intptr_t exinf, INIRTN inirtn })
```

## 【パラメータ】

\* 初期化ルーチンの追加情報

|          |        |              |
|----------|--------|--------------|
| ATR      | iniatr | 初期化ルーチン属性    |
| intptr_t | exinf  | 初期化ルーチンの拡張情報 |
| INIRTN   | inirtn | 初期化ルーチンの先頭番地 |

## 【エラーコード】

E\_RSATR 予約属性

- iniatrが無効 【NGKI3241】
- カーネルドメインの囲みの中に記述されていない [P] 【NGKI3242】

E\_PAR パラメータエラー

- inirtnがプログラムの先頭番地として正しくない 【NGKI3243】

## 【機能】

各パラメータで指定した初期化ルーチン追加情報に従って、初期化ルーチンを追加する【NGKI3244】。

iniatrは整数定数式パラメータ、exinfとinirtnは一般定数式パラメータである【NGKI3245】。

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、ターゲット定義である【NGKI3246】。

## 【補足説明】

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。

ATT\_TER 終了処理ルーチンの追加 [S] 【NGKI3247】

## 【静的API】

```
ATT_TER({ ATR teratr, intptr_t exinf, TERRTN terrtn })
```

### 【パラメータ】

\* 終了処理ルーチンの追加情報

|          |        |               |
|----------|--------|---------------|
| ATR      | teratr | 終了処理ルーチン属性    |
| intptr_t | exinf  | 終了処理ルーチンの拡張情報 |
| TERRTN   | terrtn | 終了処理ルーチンの先頭番地 |

### 【エラーコード】

E\_RSATR 予約属性

- teratrが無効 【NGKI3248】
- カーネルドメインの囲みの中に記述されていない [P] 【NGKI3249】

E\_PAR パラメータエラー

- terrtnがプログラムの先頭番地として正しくない 【NGKI3250】

### 【機能】

各パラメータで指定した終了処理ルーチン追加情報に従って、終了処理ルーチンを追加する 【NGKI3251】。

teratrは整数定数式パラメータ、exinfとterrtnは一般定数式パラメータである 【NGKI3252】。

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、ターゲット定義である 【NGKI3253】。

### 【補足説明】

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。

### 【μITRON4.0仕様との関係】

μITRON4.0仕様に定義されていない静的APIである。

```
ref_cfg    コンフィギュレーション情報の参照 [T]
```

### 【C言語API】

```
ER ercd = ref_cfg(T_RCFG *pk_rcfg)
```

未完成

### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、ref\_cfgをサポートしない。

#### 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、ref\_cfgをサポートしない。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、ref\_cfgをサポートしない。

#### 【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、ref\_cfgをサポートしない。

ref\_ver バージョン情報の参照 [T]

#### 【C言語API】

```
ER ercd = ref_ver(T_RVER *pk_rver)
```

未完成

#### 【TOPPERS/ASPカーネルにおける規定】

ASPカーネルでは、ref\_verをサポートしない。

#### 【TOPPERS/FMPカーネルにおける規定】

FMPカーネルでは、ref\_verをサポートしない。

#### 【TOPPERS/HRP2カーネルにおける規定】

HRP2カーネルでは、ref\_verをサポートしない。

#### 【TOPPERS/SSPカーネルにおける規定】

SSPカーネルでは、ref\_verをサポートしない。