



μC3/Standard ユーザーズガイド

第 12 版 イー・フォース株式会社

はじめに

μ C3 (マイクロ・シー・キューブ) とは、社団法人トロン協会がオープンなリアルタイムカーネルとして策定した μ ITRON4.0 仕様に準拠したカーネルをコアに持った RTOS (リアルタイム・オペレーティング・システム) です。

μ C3 の C3 とは、**Compact** (省メモリ)、**Connectivity** (接続性)、**Capability** (性能) の 3 つのコンセプトを表しています。また、キューブの名称は、これらによる三乗の効果をも生み出す可能性を表しています。

本書の位置づけ

本書は、μ C3/Standard のカーネル機能の共通マニュアルとし、各 CPU に依存したカーネル機能やミドルウェアに関しては別マニュアルとします。必要に応じて、これらのマニュアルを参照してください。

TRON は、"The Real-time Operation system Nucleus"の略称です。

μ ITRON は、"Micro Industrial TRON"の略称です。

μ ITRON4.0 仕様の仕様書は、トロン協会のホームページ (<http://www.assoc.tron.org/>) から入手することができます。

μ C3 は、イー・フォース株式会社の登録商標です。

本書で記載されている内容は、予告無く変更される場合があります。

改訂記録

第 2 版で訂正された項目

ページ	内容
	レイアウトの変更

第 3 版で訂正された項目

ページ	内容
17	システムサービスブロックの説明文章で、「割込み処理内で発行する」を「割込み処理内で呼び出す」に修正
17	遅延実行でのシステムコールのエラー検出の説明文章で、「実行時に発生するエラー (E_OK 以外) を検出するには、エラーハンドラを定義します。」を追記
24	セマフォの資源「獲得・返却」の記載訂正
32	メモリプール管理機能の説明で、「固定長メモリプール」の後に、「と可変長メモリプール」の記載追加
34	システム時刻の更新やカーネルで扱われる時間単位、タイムアウト時刻の変更タイミングの説明を追記
42	独自機能として、エラーハンドラを追記し、説明を記載
50,52,63,66,74,90	各システムコールの説明で、「割込みハンドラからの呼び出しでは遅延実行するため、」の説明の後、「戻り値のエラーコードでは」を追記
112	cre_mbf の解説文書で、「実装定義の最大値」を「65535」に修正
120	cre_por の解説文書で、「実装定義の最大値」を「65535」に修正
132	cre_mpf の解説文書で、「や、実装定義の最大値よりも大きい値が指定された場合」を削除
190	システムコールとして、「vdef_err」を追記し説明を記載
195	標準 COM ポートドライバシステムコールのタイトルで、「一文字受信」を「文字列受信」に訂正
200,201	データ型「TMO」「RELTIM」「SYSTIM」の説明文で、「時間単位は実装依存」を「時間単位は 1 ミリ秒」に修正
201	データ型「OVRTIM」の説明文で、「時間単位は実装依存」を「時間単位はユーザ定義」に修正

第 4 版で訂正された項目

ページ	内容
183	chg_ims についての呼出コンテキストで、割込みサービスルーチンについては、不可へと訂正
192	6.1 標準 COM ポートドライバの概要に注意事項を追加

第 5 版で訂正された項目

ページ	内容
184	chg_ims について、呼び出しコンテキストについては、「プロセッサ依存部マニュアル」を参照する記述としました。

第 6 版で訂正された項目

ページ	内容
56	エラーコード「E_NOEXS」を追記
57	タイムイベントハンドラからエラーコード「E_PAR」,「E_ID」を返すケースについて追記
64	エラーコード「E_PAR」を追記
118	エラーコード「E_PAR」を追記
155	周期ハンドラ内から自身の状態参照した場合, lefttim は不定値になる説明を追記。
161	アラームハンドラ内から自身の状態参照した場合, lefttim は不定値になる説明を追記。
164	タスクが終了しても, 上限プロセッサ時間と使用プロセッサ時間は保持される説明を追記

第 7 版で訂正された項目

ページ	内容
88	エラーコード「E_PAR」を追記
153,155	エラーコード「E_NOEXS」を追記
150,156	エラーコード「E_RSATR」を削除
179	E_PAR から imask が不正という記述を削除

第 8 版で訂正された項目

ページ	内容
203	SYSTEM を SYSTM に修正

第 9 版で訂正された項目

ページ	内容
226-231	索引のページ数表記を修正

第 10 版で訂正された項目

ページ	内容
48	DISABLE_TRACE を TRACE_DISABLE に修正 DISABLE_AGENT を AGENT_DISABLE に修正
164	ivsig_ovr のシステムコール説明で、「オーバランハンドラの時刻を更新」の記述を追加

第 11 版で訂正された項目

ページ	内容
194	5. 1 1 注意事項 5. 1 1. 1 dly_tsk とタイムアウトありシステムコール txxx_yyy の差異を追加

第 12 版で訂正された項目

ページ	内容
110	5. 4. 5 ミューテックス loc_mtx のシステムコール説明のコンテキストエラーの記述にディスパッチ保留状態を追加

目次

はじめに	1
目次	6
第 1 章 μ C3/Standard とは	9
1. 1 特長	9
1. 2 μ ITRON 仕様での位置づけ	9
第 2 章 μ C3/Standard の基本概念	10
2. 1 用語の意味	10
2. 1. 1 タスク	10
2. 1. 2 ディスパッチとスケジューリング	10
2. 1. 3 コンテキスト	10
2. 1. 4 オブジェクトと ID 番号	10
2. 1. 5 サービスコールとシステムコール	11
2. 1. 6 優先順位と優先度	11
2. 1. 7 プリエンプティブ	11
2. 1. 8 タイムチェック	11
2. 1. 9 キューイング	11
2. 1. 10 待ち行列	11
2. 2 タスクの状態とスケジューリング規則	12
2. 2. 1 タスクの状態	12
2. 2. 2 スケジューリング規則	14
第 3 章 μ C3/Standard の機能概要	16
3. 1 コンテキストとシステム状態	16
3. 1. 1 処理単位とコンテキスト	16
3. 1. 2 タスクコンテキストと非タスクコンテキスト	16
3. 1. 3 CPU ロック状態	16
3. 1. 4 ディスパッチ禁止状態	17
3. 1. 5 アイドル状態	17
3. 1. 6 ディスパッチ保留状態の間のタスク状態	18
3. 2 システムコールの遅延実行	19
3. 2. 1 システムサービスブロック	19
3. 2. 2 遅延実行でのシステムコールのエラー検出	19
3. 3 システムの起動	20
3. 3. 1 オブジェクトの ID 番号上限のコンフィグレーション情報	20

3. 3. 2	メモリ管理のコンフィグレーション情報	21
3. 3. 3	その他のコンフィグレーション情報	21
3. 4	タスク管理機能	23
3. 5	タスク付属同期機能	24
3. 6	タスク例外処理	25
3. 7	同期・通信機能	26
3. 7. 1	セマフォ	26
3. 7. 2	イベントフラグ	26
3. 7. 3	データキュー	27
3. 7. 4	メールボックス	28
3. 8	拡張同期・通信機能	30
3. 8. 1	ミューテックス	30
3. 8. 2	メッセージバッファ	31
3. 8. 3	ランデブ	32
3. 9	メモリプール管理機能	34
3. 9. 1	固定長メモリプール	34
3. 9. 2	可変長メモリプール	35
3. 10	時間管理機能	36
3. 10. 1	システム時刻管理	36
3. 10. 2	周期ハンドラ	36
3. 10. 3	アラームハンドラ	38
3. 10. 4	オーバランハンドラ	39
3. 11	システム状態管理機能	40
3. 12	割込み管理機能	41
3. 13	システム構成管理機能	42
3. 14	独自機能	43
3. 14. 1	デバイスドライバ管理	43
3. 14. 2	エラーハンドラ	44
第4章	コンフィグレーションとシステムの起動	45
4. 1	カーネルの起動	45
4. 2	システムスタック	46
4. 3	カーネルのコンフィグレーション	47
第5章	システムコールの説明	49
5. 1	タスク管理機能	49
5. 2	タスク付属同期機能	65
5. 3	タスク例外処理	73
5. 4	同期・通信機能	74

5. 4. 1	セマフォ.....	74
5. 4. 2	イベントフラグ.....	81
5. 4. 3	データキュー.....	89
5. 4. 4	メールボックス.....	99
5. 4. 5	ミューテックス.....	107
5. 4. 6	メッセージバッファ.....	114
5. 4. 7	ランデブ.....	122
5. 5	メモリプール管理機能	134
5. 5. 1	固定長メモリプール.....	134
5. 5. 2	可変長メモリプール.....	141
5. 6	時間管理機能	148
5. 6. 1	システム時刻管理.....	148
5. 6. 2	周期ハンドラ.....	151
5. 6. 3	アラームハンドラ.....	157
5. 6. 4	オーバランハンドラ.....	163
5. 7	システム状態管理機能	168
5. 8	割込み管理機能	179
5. 9	システム構成管理機能	188
5. 10	独自機能	191
5. 10. 1	デバイスドライバ管理機能.....	191
5. 10. 2	エラーハンドラ.....	193
5. 11	注意事項	194
5. 11. 1	dly_tsk とタイムアウトありシステムコール txxx_yyy の差異	194
第6章	標準 COM ポートドライバの説明.....	195
6. 1	標準 COM ポートドライバの概要	195
6. 2	標準 COM ポートドライバのサービスコール	196
第7章	付録.....	204
7. 1	データ型	204
7. 2	パケット形式	206
7. 3	定数とマクロ	215
7. 4	構成定数とマクロ	218
7. 5	エラーコード一覧	220
7. 6	システムコール一覧	221
索引		228

第1章 μ C3/Standard とは

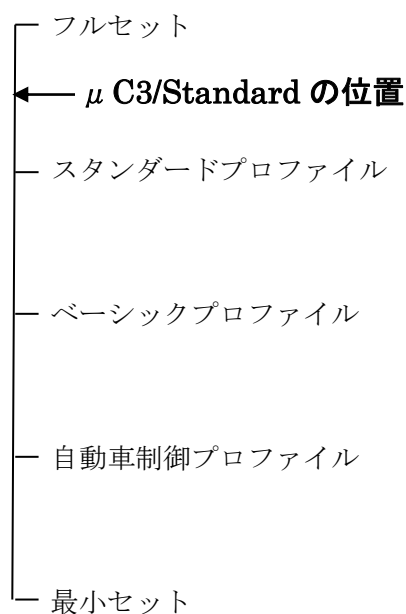
1. 1 特長

μ C3/Standard は、 μ C3 の3つのコンセプトの中では Capability に重点を置き、割込み応答性を犠牲にすることなく、 μ ITRON の複雑で高度な機能を組み込んだ製品です。

複雑で高度な機能は上限のないループを必要とする場合がありますが、 μ C3/Standard には割込み禁止期間中の割込み応答性を悪化させるループは一切ありません。更に、システムコールやディスパッチャの大部分が割込み許可状態で実行されるなど、割込み応答性を重視しています。

1. 2 μ ITRON 仕様での位置づけ

μ ITRON4.0 仕様にはプロファイルの概念を持っていますが、 μ C3/Standard はどのプロファイルにも該当せず、フルセットを基本にし、使われる可能性の少ないと思われる機能を省きました。



フルセットに対し省いた機能として、次があります。

- ・静的 API と、これを解釈するコンフィグレータ
- ・タスク例外処理機能
- ・サービスコール管理機能

第 2 章 μ C3/Standard の基本概念

2. 1 用語の意味

2. 1. 1 タスク

並行処理されるプログラムの実行単位を「タスク」と呼びます。つまり、アプリケーションから見ると、それぞれのタスクは並行処理しているかのように実行されます。実際には、同時実行されるプログラムはプロセッサの個数となるため、カーネルはスケジューリング規則に則り、各タスクを細かな時間で区切って実行することにより並行処理に見せかけています。また、システムコールを呼び出したタスクを「自タスク」と呼びます。

2. 1. 2 ディスパッチとスケジューリング

プロセッサが、実行しているタスクの切り替えを行うことを「ディスパッチ」と呼び、ディスパッチの機構を「ディスパッチャ」と呼びます。

次に実行させるタスクを選択することを「スケジューリング」と呼び、スケジューリングの機構を「スケジューラ」と呼びます。

一般的には、ディスパッチャとスケジューラは明確に分離されていることは少なく、μ C3 では、これらを一体化して「ディスパッチャ」や「ディスパッチ」と呼びます。

2. 1. 3 コンテキスト

プログラムが実行される環境を「コンテキスト」と呼び、タスクやタイムイベントハンドラや割込みハンドラはそれぞれのコンテキストを持っていると考えます。異なるコンテキストに切り替わる場合は、再開するために必要なデータを待避・復元しなければいけないことから、コンテキストをプロセッサのレジスタ値として扱うことが一般的です。

2. 1. 4 オブジェクトと ID 番号

カーネルやソフトウェア部品の操作対象を総称してオブジェクトと呼び、そのオブジェクトの識別に用いるのが ID 番号です。μ C3/Standard では ID 番号を、1 からコンフィギュレーションで指定する上限までの値をとります。オブジェクトの ID 番号は、タスク ID、セマフォ ID などのように、オブジェクト名+ID の形式で呼びます。

カーネルのオブジェクトには、タスク、セマフォ、イベントフラグ、データキュー、メールボックス、ミューテックス、メッセージバッファ、ランデブ、固定長メモリプール、可変長メモリプール、周期ハンドラ、アラームハンドラ、割込みサービスルーチンがあります。

2. 1. 5 サービスコールとシステムコール

アプリケーションからカーネルやソフトウェア部品を呼び出すインタフェースをサービスコールと呼びます。 μ C3 では、カーネルのサービスコールをシステムコールと呼びます。

2. 1. 6 優先順位と優先度

処理が実行される順序を決める順序関係を「優先順位」と呼び、そのためにアプリケーションが与えるパラメータを「優先度」と呼びます。優先度は数値（自然数）で表し、小さな値ほど優先度は高く、大きな値ほど優先度は低くなります。

タスク優先度には、ベース優先度と現在優先度とあります。

2. 1. 7 プリエンプティブ

実行中のタスクより優先順位の高いタスクが実行可能状態になった場合、そのタスクにディスパッチできることを「プリエンプティブ」と呼びます。

2. 1. 8 タイムチェック

カーネル内部では、システム時刻を管理しています。システム時刻を計時するための一定周期のイベントを「タイムチェック」と呼びます。つまり、タイムチェックの周期が 1ms であれば、システム時刻は 1ms の精度に、 2ms の周期であれば 2ms の精度になります。

2. 1. 9 キューイング

何らかの処理要求を、即時実行できない場合に保持する機能を「キューイング」と呼び、要求数をカウントするカウンタとして実装されています。キューイングには、起動要求キューイングと起床要求キューイングとがあります。

2. 1. 10 待ち行列

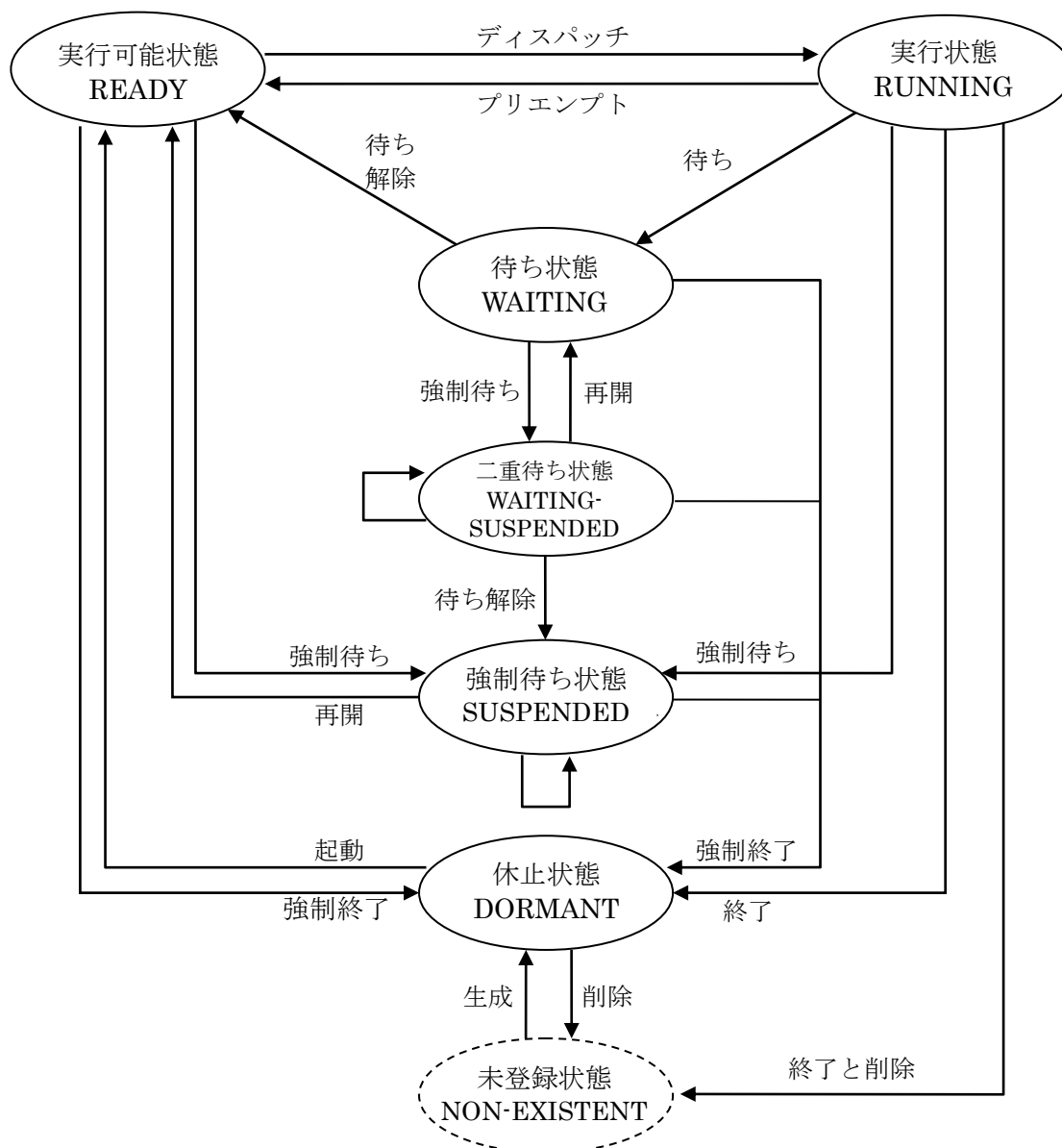
あるオブジェクトに要求するシステムコールを呼び出した場合、即時処理できない時には、処理が行えるまで、あるいは許された時間内で待つことのできるシステムコールがあります。このようなシステムコールでは、システムコールを呼び出した順に並べ、早いものから順に処理します。この機能を「待ち行列」と呼びます。

待ち行列には、タスク優先度の高いものから順に処理するタスク優先度順待ち行列もあります。この場合に、同一優先度のタスク間では、早いものから順に処理します。

2. 2 タスクの状態とスケジューリング規則

2. 2. 1 タスクの状態

μ C3/Standard でタスクの状態は、大別して7つの状態があり、広義の待ち状態は3つの状態があります。また、実行状態と実行可能状態を総称して、「実行できる状態」と呼びます。これらのタスクの状態遷移を以下の図に示します。



タスクの状態遷移図

A 実行状態 (RUNNING)

現在、実行している状態で、この実行状態に遷移するタスクの個数は同時に1個までです。スケジューラが実行可能状態のタスクの中から一つを決定し、ディスパッチャにより実行状態へと遷移されます。つまり、タスクの実行中に非タスクコンテキストに切り替わった場合でも、実行状態のタスクに変化はありません。

B 実行可能状態 (READY)

タスクとしては実行できる状態なのに、何らかの理由で実行できない状態です。つまり、そのタスクよりも優先順位の高いタスクが実行中の場合とディスパッチが起こらない状態の場合です。 μ C3 での「ディスパッチが起こらない状態」とは、ディスパッチ禁止状態と、割り込みマスクをタスクレベルより高くしている状態です。

C 広義の待ち状態

何らかの条件が成り立つのを待っている状態で、そのタスクのコンテキストは、再開が可能なように、タスクの管理領域に格納されています。広義の待ち状態には、次の3つの状態があります。

C. 1 待ち状態 (WAITING)

呼び出したシステムコールの条件が成り立たず、実行が中断された状態です。具体的には、起床待ち、時間経過待ち、イベントフラグ待ち、セマフォ待ち、メールボックスのメッセージ受信待ち、データキューのメッセージ受信待ちと送信待ち、ミューテックスのロック待ち、メッセージバッファの受信待ちと送信待ち、ランデブ固定長・可変長メモリブロックの獲得待ち、ランデブの呼出し待ちと受付待ちと終了待ちがあります。

C. 2 強制待ち状態 (SUSPENDED)

他のタスクによって、あるいは自タスクにより、強制的に実行を中断させられた状態です。

C. 3 二重待ち状態 (WAITING-SUSPENDED)

待ち状態と強制待ち状態が重なった状態です。待ち状態にあるタスクに対して、強制待ち状態への移行が要求されると、二重待ち状態に遷移させます。

D 休止状態 (DORMANT)

タスクの起動前か実行終了後の状態です。休止状態にある間は、実行状態であった時の情報は保存されいません。休止状態から起動する時には、タスクの起動番地から実行を開始します。

E 未登録状態 (NON-EXISTENT)

タスクがまだ生成されていないか、削除された後の、システムに登録されていない仮想的な状態です。

タスクの起動とは、タスクを休止状態から実行可能状態に遷移させることをいい、休止状態と未登録状態以外の状態を総称して「起動された状態」と呼びます。タスクの終了とは、起動された状態のタスクを休止状態に遷移させることをいいます。

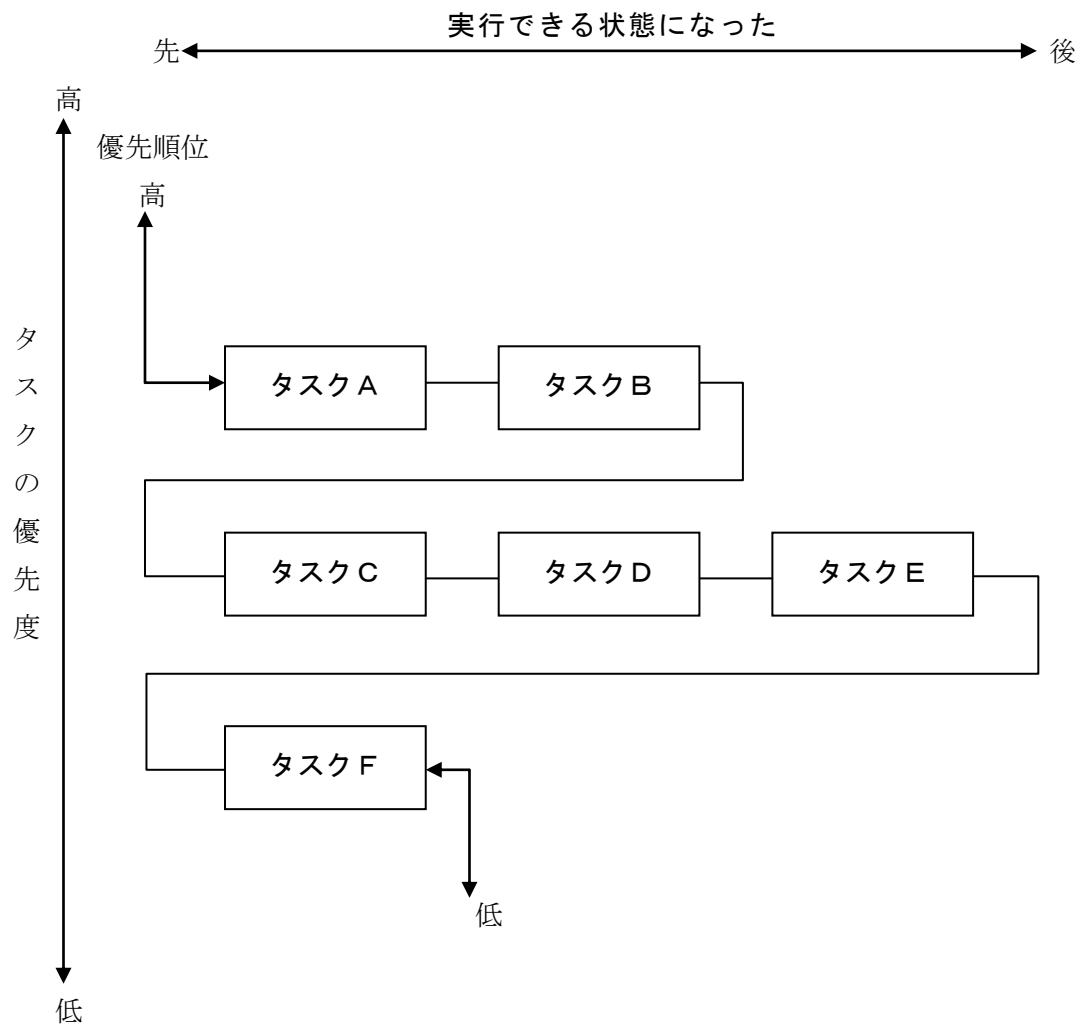
タスクの待ち解除とは、タスクが待ち状態の時は実行可能状態に、二重待ち状態の時は強制待ち状態に遷移させることをいいます。

2. 2. 2 スケジューリング規則

タスクに与えられる優先度に基づきプリエンティブな優先度ベーススケジューリングを行います。実行できる状態のタスクが複数ある場合は、その中で最も優先順位の高いタスクが実行状態になります。ただし、ディスパッチが起こらない状態では、ディスパッチが起こる状態になるまでディスパッチは保留されます。

最も優先順位の高いタスクとは、一番高い優先度を持ち、同一優先度のタスクが複数ある場合は、先に実行できる状態になったタスクです。この関係を図にすると、次のようになります。ただし、システムコール (chg_pri, rot_rdq) の呼び出しにより、同じ優先度を持つタスク間で優先順位が変更される場合があります。

例えば、タスク A とタスク B のタスク優先度は同じですが、タスク A は実行できる状態になったのが早いため、タスク優先順位はタスク A の方が高いと考えます。



優先順位と優先度の関係図

第3章 μ C3/Standard の機能概要

3. 1 コンテキストとシステム状態

3. 1. 1 処理単位とコンテキスト

μ C3/Standard のカーネルは、次の処理単位で実行されます。本書では、割込みハンドラに関する記述は、違いを明記していない箇所を除き、割込みサービスルーチンにも適用されます。

A. 割込みハンドラ

A.1 割込みサービスルーチン

B. タイムイベントハンドラ

C. CPU 例外ハンドラ

D. タスク

E. アイドル

3. 1. 2 タスクコンテキストと非タスクコンテキスト

タスクの処理の一部とみなすことのできるコンテキストをタスクコンテキスト、そうでないコンテキストを非タスクコンテキストと呼びます。非タスクコンテキストには、割込みハンドラ、タイムイベントハンドラ、アイドルが実行されるコンテキストなどがあります。

μ C3/Standard では、タスクコンテキストから呼び出すシステムコールと、割込みハンドラから呼び出すシステムコールと、タイムイベントハンドラから呼び出すシステムコールと、初期化ハンドラから呼び出すシステムコールとをそれぞれ区別して扱います。アイドルからは、システムコールを呼び出すことはできません。非タスクコンテキストから、自タスクを指定するシステムコールや、自タスクを指定するパラメータを用いることはできません。また、自タスクを広義の待ち状態にする可能性のあるシステムコールを呼び出すことはできません。

3. 1. 3 CPU ロック状態

システム状態は、CPU ロック状態か CPU ロック解除状態かのいずれかの状態をとります。CPU ロック状態では、カーネルの管理外の割込みを除くすべての割込みが禁止され、ディスパッチは起こりません。また、割込みが禁止されているため、タイムイベントハンドラの起動も保留されます。

CPU ロック状態に遷移することを「CPU ロックする」といい、CPU ロック解除状態に遷移することを「CPU ロックを解除する」といいます。具体的な、CPU ロックする処理と CPU ロックを解除する処理や、割込みハンドラ開始直後の状態は、プロセッサにより異なり、それらの説明は「プロセッサ依存部マニュアル」を参照してください。

CPU ロック状態では、広義の待ち状態に遷移する可能性があるシステムコールが呼び出さ

れた場合には、E_CTX エラーを返します。タイムイベントハンドラの実行開始直後は CPU ロック解除状態になっており、アプリケーションで CPU ロック状態にした場合は、ハンドラから戻る前に CPU ロック解除状態にしなければなりません。

タスクの実行開始直後は、CPU ロック解除状態になっています。アプリケーションは、自タスクを終了させる前に CPU ロック解除状態にしなければなりません。CPU ロック解除状態であっても、割込みが許可されているとは限りません。その関係はプロセッサにより異なり、詳細は「プロセッサ依存部マニュアル」を参照してください。

3. 1. 4 ディスパッチ禁止状態

システム状態は、ディスパッチ禁止状態かディスパッチ許可状態かのいずれかの状態をとります。ディスパッチ禁止状態では、ディスパッチは起こりません。

ディスパッチ禁止状態に遷移することを「ディスパッチを禁止する」といい、ディスパッチ許可状態に遷移することを「ディスパッチを許可する」といいます。

ディスパッチ禁止状態では、タスクコンテキストから呼び出した自タスクを広義の待ち状態にする可能性のあるシステムコールが呼び出された場合には、E_CTX エラーを返します。また、非タスクコンテキストから呼び出せるシステムコールは、ディスパッチ禁止状態でも制限されません。

割込みハンドラ、タイムイベントハンドラの実行はディスパッチ禁止／許可状態に影響を与えません。これらのハンドラ実行開始直後は、その前に実行していたタスクコンテキストのディスパッチ禁止／許可状態が保持されます。また、これらのハンドラ内でディスパッチ禁止／許可状態を変えるシステムコールが呼び出された場合には、E_CTX エラーを返します。

3. 1. 5 アイドル状態

実行できる状態のタスクがなく、タイムイベントハンドラもなく、割込み処理も実行していない場合にはアイドルを実行し、その状態を「アイドル状態」と呼びます。アイドルの実行開始直後は CPU ロック解除状態、ディスパッチ許可状態になっています。 μ C3/Standard では、アイドル状態は独立したコンテキストを持ちますが、他のコンテキストとは性質が異なります。その異なる性質とは、他のコンテキストに切り替わる際には、そのコンテキストを保存しないことです。

コンテキストを保存しないため、アイドル実行中に割込みが発生し、非タスクコンテキストが実行された後に、アイドル内の割込み発生箇所に戻ることはありません。アイドルコンテキストに切り替わった場合には、必ず、アイドルの先頭から実行されます。

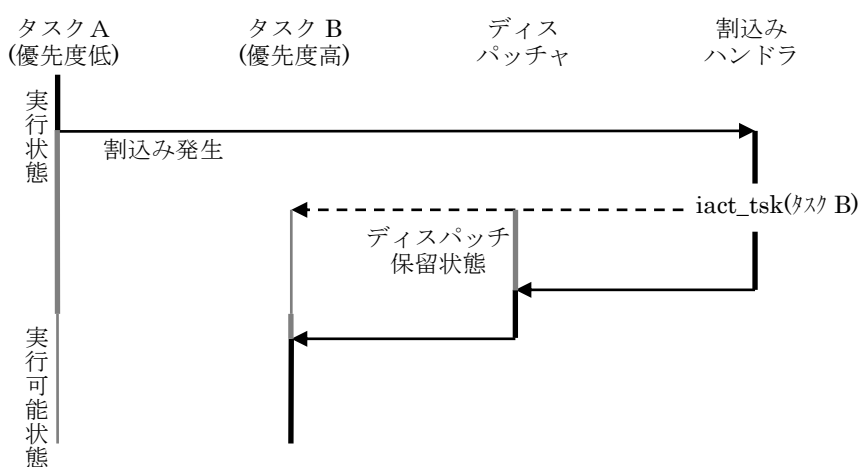
カーネルが用意するアイドルは、処理のない単純ループになっています。このアイドルには、コンフィグレーションでユーザによるアイドル関数を定義することができます。

3. 1. 6 ディスパッチ保留状態の間のタスク状態

ディスパッチャよりも優先順位の高い処理が実行されている間、CPU ロック状態の間、割込みレベルをタスクレベルより高くしている間、およびディスパッチ禁止状態の間は、ディスパッチが起こりません。この状態をディスパッチ保留状態と呼びます。

ディスパッチ保留状態では、実行中のタスクより優先順位の高いタスクが発生しても、その優先順位の高いタスクにはディスパッチされません。優先順位の高いタスクへのディスパッチは、ディスパッチが起こる状態になるまで保留されます。ディスパッチが保留されている間は、それまで実行中であったタスクが実行状態であり、ディスパッチが起こる状態となった後に実行すべきタスクは実行可能状態のままです。

ディスパッチ保留状態の間のタスク状態を次の図を用いて説明します。



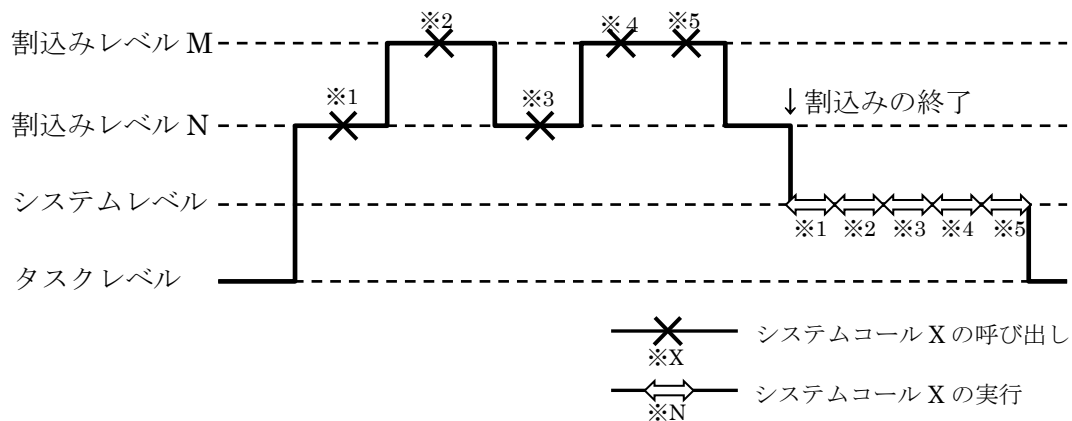
ディスパッチ保留状態とタスク状態の図

タスク A の実行中に発生した割込みによって起動された割込みサービスルーチンから、タスク A よりも高い優先度を持つタスク B を起動した場合を考えます。割込みサービスルーチンの優先順位はディスパッチャの優先順位より高いため、タスク B を起動した後も割込みサービスルーチンが実行されている間はディスパッチ保留状態になり、ディスパッチは起こりません。割込みサービスルーチンの実行が終了すると、ディスパッチャが実行され、タスク A を実行可能状態に、タスク B を実行中に遷移します。

割込みハンドラ内でタスク B が起動された後も、ディスパッチャが実行されるまでの間はタスク A が実行状態であり、タスク B は実行可能状態のままです。

3. 2 システムコールの遅延実行

μ C3/Standard では、割込みハンドラでのオーバーヘッドを軽減し、割込み以外で呼び出されたシステムコールの割込み禁止区間の最小限にするため、割込みハンドラで呼び出されたシステムコールの遅延実行を行います。つまり、割込みハンドラ内でシステムコールを発行した場合は、パラメータチェックなどオブジェクトの状態に依存しないエラーのみ検出し、システムコールを一旦終了します。そして、割込みが終了した後、システムコールの残りを処理します。



システムコールの遅延実行の図

3. 2. 1 システムサービスブロック

システムコールの遅延実行では、呼び出しから実行まで、パラメータの情報をシステムサービスブロック (SSB と略します) と呼ぶメモリブロックに FIFO 順に記憶しています。このシステムサービスブロックは、最低でも、割込み処理内で呼び出すシステムコールの数だけで必要で、図の例では5個のシステムサービスブロックが使われています。また、システムコールの実行までに同一割込みを複数回受け付ける場合は、より多くの個数が必要です。

3. 2. 2 遅延実行でのシステムコールのエラー検出

システムコールの遅延実行は、オブジェクトの状態に起因するエラーをシステムコールの呼び出し時点で検出できないため、正常 (E_OK) を返します。実行時に発生するエラー (E_OK 以外) を検出するには、エラーハンドラを定義します。

非タスクコンテキストに分類されるタイムイベントハンドラからのシステムコールの呼び出しは、遅延実行されません。

3. 3 システムの起動

システム起動は、コンフィグレーション情報を基にして、μ C3/Standard を初期化しマルチタスクの状態を開始させる機能です。

- ・ システムを起動する機能 (start_uC3)

システム設計により決定される、各オブジェクトの ID 番号の上限、動的なメモリ管理に用いる領域、その他のコンフィグレーション情報を元にして基にして μ C3/Standard を起動します。

初期化ハンドラは次の形式で記述します。

```
void initpr(void)
{
    マルチタスクに遷移する前に必要な処理を記述
}
```

3. 3. 1 オブジェクトの ID 番号上限のコンフィグレーション情報

オブジェクトの ID 番号上限として、次の項目があります。

- ・ タスク ID の上限値
- ・ セマフォ ID の上限値
- ・ イベントフラグ ID の上限値
- ・ データキューID の上限値
- ・ メールボックス ID の上限値
- ・ ミューテックス ID の上限値
- ・ メッセージバッファ ID の上限値
- ・ ランデブポート ID の上限値
- ・ 固定長メモリプール ID の上限値
- ・ 可変長メモリプール ID の上限値
- ・ 周期ハンドラ ID の上限値
- ・ アラームハンドラ ID の上限値
- ・ 割込みサービスルーチン ID の上限値
- ・ 標準 COM ドライバ ID の上限値

ID の上限値とは、生成時に指定、あるいは割り付けられる ID 番号の上限になります。つまり、そのオブジェクトを生成できる最大の個数のことです。タスクは 255 個、その他は 999 個が生成できる上限で、ID の最大値になります。

また、標準 COM ドライバ ID の上限値は、μ C3 の標準 COM ドライバを使用する場合に、その個数を指定します。

3. 3. 2 メモリ管理のコンフィグレーション情報

メモリ管理の対象となる領域として、次の項目があります。

- ・ システムメモリ領域
- ・ スタック用メモリ領域
- ・ メモリプール用メモリ領域

メモリプール用メモリ領域は、可変長／固定長メモリプールのメモリブロック領域、データキュー領域、メッセージバッファ領域に割り当てられます。NULL を指定した場合には、メモリプール用メモリ領域は生成されず、代わってこれらにはスタック用メモリが用いられます。

スタック用メモリ領域は、タスク生成時にスタック領域を指定しなかった場合に、この領域から割り当てられます。NULL を指定した場合には、スタック用メモリ領域は生成されず、代わってスタックにはシステムメモリ領域が用いられます。

システムメモリ領域は、各オブジェクトの生成時に必要とされる管理領域に割り当てられます。この領域は、必ず指定し、省略することはできません。

3. 3. 3 その他のコンフィグレーション情報

その他に、次の項目があります。

- ・ タイムチックの周期を指定するチック時間
- ・ タスク優先度の上限値
- ・ SSB の個数
- ・ ユーザ定義アイドル関数
- ・ デバッグ支援

チック時間は、システムコールの時間監視や、オーバランハンドラ以外のタイムイベントハンドラの時間精度を決定し、小さな値ほど精度が上がりますが、一方でオーバヘッドも大きくなります。ミリ秒単位で指定し、1 から 10 ミリ秒程度が一般的です。

タスク優先度は最大で 31 に制限され、上限値は 31 以下を指定します。

ユーザ定義のアイドル関数は、次の形式で記述し、 μ C3 に組み込まれた標準のアイドル関数を置き換えることができます。

```
void user_idle(void)
{
    アイドル状態に必要な処理
}
```

デバッグ支援には、タスク生成時のスタック領域の初期化があり、初期化なしや 0 の初期化やタスク ID に初期化があります。また、次の形式で、ユーザ独自の初期化関数を記述できます。

```
void user_stack_init(INT *stk, SIZE sz, ID tskid)
```

```
{
```

```
    ID 番号 tskid のタスクのスタック先頭番地 stk からバイトサイズ sz を初期化
```

```
}
```

その他のデバッグ支援として、コンテキスト切り替え時のトレースとシステムコール発行時のトレース機能を支援があります。これらの詳細は、μ C3/Standard に対応したデバッガのマニュアルを参照してください。

3. 4 タスク管理機能

タスク管理機能は、タスクの状態を直接的に操作／参照するための機能です。具体的には、次の機能が含まれています。

- ・ タスクを生成する機能 (`cre_tsk`, `acre_tsk`)
- ・ タスクを削除する機能 (`del_tsk`)
- ・ タスクを起動する機能 (`act_tsk`, `iact_tsk`, `sta_tsk`)
- ・ タスクを終了させる機能 (`ext_tsk`, `ter_tsk`)
- ・ タスクに対する起動要求をキャンセルする機能 (`can_act`)
- ・ タスクの優先度を変更する機能 (`chg_pri`)
- ・ タスクの状態を参照する機能 (`get_pri`, `ref_tsk`, `ref_tst`)

タスクは、ベース優先度と現在優先度によって実行順序が制御されます。ミューテックス使用時は、現在優先度がベース優先度よりも高くなっている場合がありますが、ミューテックス未使用時は、現在優先度とベース優先度は常に一致しています。単に、タスク優先度といった場合は、現在優先度の方を指します。

タスクに対する起動要求は、キューイングされます。つまり、すでに起動されているタスクを再度起動しようとする、そのタスクを起動しようとした要求を保持し、後でそのタスクが終了した時に、タスクを自動的に再起動します。ただし、起動コードを指定してタスクを起動するシステムコール (`sta_tsk`) では、起動要求はキューイングされません。タスクに対する起動要求のキューイングを実現するために、タスクは起動要求キューイング数を持ちます。タスクの起動要求キューイング数は、タスクの生成時に 0 にクリアします。

タスクを起動する際には、そのタスクの拡張情報 (`exinf`) をパラメータとして渡します。ただし、起動コードを指定してタスクを起動するシステムコール (`sta_tsk`) によってタスクが起動された場合には、拡張情報に代えて、指定された起動コードを渡します。

タスクの起動時には、タスクの現在優先度とベース優先度は起動時優先度に初期化され、起床要求キューイング数のクリアが行われます。

タスクは次の形式で記述します。

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

タスクのメインルーチンの途中からリターンした場合でも、`ext_tsk` を呼び出した場合と同様に振る舞います。

タスク管理機能に関連して、次のマクロを用意しています。

`TMAX_ACTCNT` タスクの起動要求キューイング数の最大値 (999)

3. 5 タスク付属同期機能

タスク付属同期機能は、タスクの状態を直接的に操作することによって同期を行うための機能です。具体的には、次の機能が含まれています。

- ・ タスクを起床待ちにする機能 (slp_tsk, tslp_tsk)
- ・ タスクを起床待ちから起床させる機能 (wup_tsk, iwup_tsk)
- ・ タスクの起床要求をキャンセルする機能 (can_wup)
- ・ タスクの待ち状態を強制解除する機能 (rel_wai, irel_wai)
- ・ タスクを強制待ちにする機能 (sus_tsk)
- ・ タスクの強制待ち状態から再開する機能 (rsm_tsk)
- ・ タスクの強制待ち状態から強制再開する機能 (frsm_tsk)
- ・ 自タスクの実行を遅延する機能 (dly_tsk)

タスクに対する起床要求は、キューイングされます。つまり、起床待ち状態でないタスクを起床しようとする、そのタスクを起床しようとした起床要求が保持され、後でそのタスクが起床待ちに遷移しようとした時に、タスクを起床待ち状態にしません。タスクに対する起床要求のキューイングさせるために、タスクは起床要求キューイング数を持ちます。タスクの起床要求キューイング数は、タスクの起動時に 0 にクリアします。

タスクに対する強制待ち要求は、ネストされます。つまり、すでに強制待ち状態（二重待ち状態を含む）になっているタスクを再度強制待ち状態に遷移させようとする、そのタスクを強制待ち状態に遷移させようとしたという要求が保持され、後でそのタスクを強制待ち状態（二重待ち状態を含む）から再開させようとした時に、強制待ちは解除されません。タスクに対する強制待ち要求のネストを実現するために、タスクは強制待ち要求ネスト数を持ちます。タスクの強制待ち要求ネスト数は、タスクの起動時に 0 にクリアされます。

タスク付属同期機能に関連して、次のマクロを用意しています。

TMAX_WUPCNT	タスクの起床要求キューイング数の最大値 (999)
TMAX_SUSCNT	タスクの強制待ち要求ネスト数の最大値 (999)

3. 6 タスク例外処理

タスク例外処理は、現バージョンでは対応していません。

3. 7 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、タスク間の同期・通信を行うための機能です。セマフォ、イベントフラグ、データキュー、メールボックスの各機能が含まれます。

3. 7. 1 セマフォ

セマフォは、使用されていない資源の有無や数量を数値で表現することにより、その資源を使用する際の排他制御や同期を行うためのオブジェクトです。具体的には、次の機能が含まれています。

- ・ セマフォの生成する機能 (cre_sem, acre_sem)
- ・ セマフォの削除する機能 (del_sem)
- ・ セマフォの資源を返却する機能 (sig_sem, isig_sem)
- ・ セマフォの資源を獲得する機能 (wai_sem, pol_sem, twai_sem)
- ・ セマフォの状態を参照する機能 (ref_sem)

セマフォは、対応する資源の有無や数量を表現する資源数と、資源の獲得を待つタスクのFIFO 順、あるいはタスク優先度順の待ち行列を持ちます。資源を返却する側では、セマフォの資源数に 1 を加算します。一方、資源を獲得する側では、セマフォの資源数から 1 を減算します。セマフォの資源数が足りなくなった場合（資源数を減算すると負になる場合）、資源を獲得しようとしたタスクは、次に資源が返却されるまで、あるいは許された時刻まで、待ち行列につながれセマフォ資源の獲得待ち状態に遷移します。また、セマフォに対して資源が返却され過ぎるのを防ぐために、セマフォ毎に最大資源数を設定することができます。最大資源数を越える資源がセマフォに返却されようとした場合（セマフォの資源数を加算すると最大資源数を越える場合）には、エラーを返します。

セマフォ機能に関連して、次のマクロを用意しています。

TMAX_MAXSEM セマフォの最大資源数の最大値 (999)

3. 7. 2 イベントフラグ

イベントフラグは、イベントの有無をビット毎のフラグで表現することにより、同期を行うためのオブジェクトです。具体的には、次の機能が含まれています。

- ・ イベントフラグを生成する機能 (cre_flg, acre_flg)
- ・ イベントフラグを削除する機能 (del_flg)
- ・ イベントフラグをセットする機能 (set_flg, iset_flg)
- ・ イベントフラグをクリアする機能 (clr_flg)
- ・ イベントフラグで待つ機能 (wai_flg, pol_flg, twai_flg)
- ・ イベントフラグの状態を参照 (ref_flg)

イベントフラグは、対応するイベントの有無をビット毎に表現するビットパターンと、そのイベントフラグで待つタスクの **FIFO** 順、あるいはタスク優先度順の待ち行列を持ちます。イベントフラグのビットパターンを、単にイベントフラグと呼ぶ場合もあります。

イベントを知らせる側では、イベントフラグのビットパターンの指定したビットをセット、あるいはクリアすることができます。一方、イベントを待つ側では、イベントフラグのビットパターンの指定したビットのすべて (**AND**) またはいずれか (**OR**) の待ち条件があり、条件が成り立っていない場合は、それが成り立つまで、あるいは許された時刻まで、待ち行列につながれイベントフラグ待ち状態に遷移します。

イベントフラグ機能に関連して、次のマクロを用意しています。

TBIT_FLGPTN イベントフラグのビット数 (プロセッサに依存)

3. 7. 3 データキュー

データキューは、1 ワードのメッセージ (これをデータと呼びます) を受け渡しすることにより、同期と通信を行うためのオブジェクトです。具体的には、次の機能が含まれています。

- ・ データキューを生成する機能 (**cre_dtq,acre_dtq**)
- ・ データキューを削除する機能 (**del_dtq**)
- ・ データキューにデータを送信する機能 (**snd_dtq,psnd_dtq,ipsnd_dtq,tsnd_dtq**)
- ・ データキューにデータを強制送信する機能 (**fsnd_dtq,ifsnd_dtq**)
- ・ データキューからデータを受信する機能 (**rcv_dtq,prev_dtq,trcv_dtq**)
- ・ データキューの状態を参照する機能 (**ref_dtq**)

データキューは、データの送信を待つタスクの **FIFO** 順、あるいはタスク優先度順の送信待ち行列と、データの受信を待つタスクの **FIFO** 順の受信待ち行列を持ちます。また、送信されたデータを格納するためのリング状のデータキュー領域を持ちます。

データを送信する側では、送信したいデータをデータキューに入れます。データキュー領域に空きがない場合は、データキュー領域に空きができるまで、あるいは許された時刻まで、送信待ち行列につながれ、データキューへの送信待ち状態に遷移します。

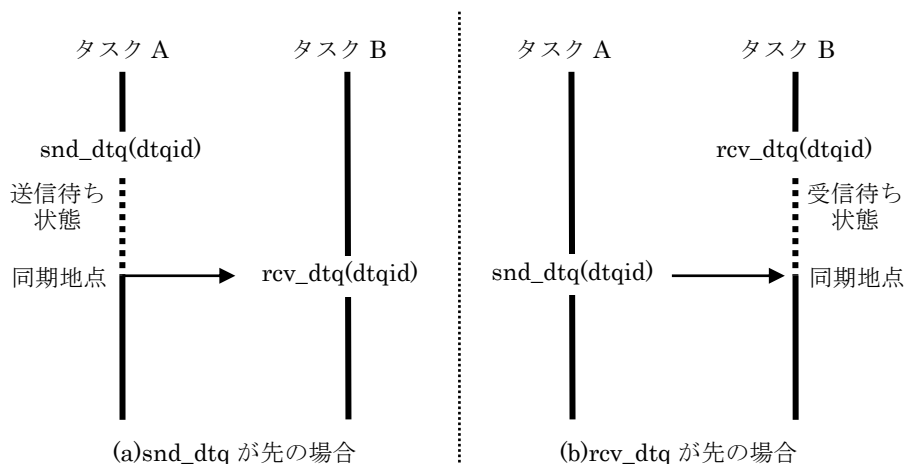
一方、データを受信する側では、データキューに入っているデータを一つ取り出します。データキューにデータが入っていない場合は、データが送られてくるまで、あるいは許された時刻まで、受信待ち行列につながれ、データキューからの受信待ち状態に遷移します。

データキュー領域に格納できるデータの数を 0 にすることで、同期メッセージ機能を実現することができます。つまり、送信側のタスクと受信側のタスクが、それぞれ相手のタスクが送受信のシステムコールを呼び出すのを待ち合わせ、ともにシステムコールを呼び出した時点で、データの受け渡しが行われます。

データキューで送受信されるデータは、整数値であっても、メモリの番地 (ポインタ型) であっても構いません。送受信されるデータは、送信側から受信側にコピーされます。

先に **snd_dtq** を呼び出した場合には、**rcv_dtq** が呼び出されるまで、送信待ち状態に遷移し

ます。逆に、先に `rcv_dtq` を呼び出した場合には、`snd_dtq` を呼び出されるまで、受信待ち状態に遷移します。



データキューによる同期通信の図

データキュー機能に関連して、次のマクロを用意しています。

`SIZE dtqsz = TSZ_DTQ(UINT dtqcnt)`

`dtqcnt` 個のデータを格納するのに必要なデータキュー領域のサイズ (バイト数)

3. 7. 4 メールボックス

メールボックスは、メモリ上に置かれたメッセージを受け渡すことにより、同期と通信を行うためのオブジェクトです。具体的には、次の機能が含まれています。

- ・ メールボックスを生成する機能 (`cre_mbx, acre_mbx`)
- ・ メールボックスを削除する機能 (`del_mbx`)
- ・ メールボックスへメッセージを送信する機能 (`snd_mbx`)
- ・ メールボックスからメッセージを受信する機能 (`rcv_mbx, prcv_mbx, trcv_mbx`)
- ・ メールボックスの状態を参照する機能 (`ref_mbx`)

メールボックスは、メッセージの受信を待つタスクの **FIFO 順**、あるいはタスク優先度順の待ち行列とメッセージキューを持ちます。メールボックスへメッセージを送信する場合には、受信待ち状態のタスクがあればメッセージを渡し、そのタスクを待ち状態から実行可能状態へ遷移させます。受信待ち状態のタスクがなければ、メッセージキューにメッセージを入れます。

メールボックスからメッセージを受信する場合には、メッセージキューにメッセージがあればメッセージを取り出します。メッセージがなければ、メッセージが送信されるまで、あるいは許された時刻まで、受信待ち行列につながれ、受信待ち状態に遷移します。メールボックスによって実際に送受信されるのは、メモリ上に置かれたメッセージの先頭番地のみで、送受信されるメッセージの内容はコピーしません。

メールボックスのメッセージの送信では、アプリケーションで用意するメッセージパケットの先頭番地をパラメータとして用います。また、メッセージの受信では、メッセージパケットの先頭番地をリターンパラメータとして受け取ります。具体的に、メッセージパケットとは、カーネルがメッセージキューの順番を特定できるメッセージヘッダを先頭のフィールドに、それに続けたアプリケーションで使われるメッセージ本体で構成されます。

メッセージヘッダのデータ型として、次のデータ型を定義しています。

T_MSG メールボックスのメッセージヘッダ
T_MSG_PRI メールボックスの優先度付きメッセージヘッダ

例えば、FIFO 順のメッセージパケット **T_MSGPKT** 型の定義は次のようになります。

```
typedef struct t_msgpkt{
    T_MSG      msg;    /* メッセージヘッダ */
                  /* アプリケーションで使うメッセージの本体 */
} T_MSGPKT;
```

メッセージ優先度順のメッセージパケット **T_MSGPRIPKT** 型の定義は次のようになります。

```
typedef struct t_msgpripkt{
    T_MSG_PRI  pk_msg; /* 優先度付きのメッセージヘッダ */
                  /* アプリケーションで使うメッセージの本体 */
} T_MSGPRIPKT;
```

ユーザプログラムは、メッセージキューに入っているメッセージパケットの内容を決して書き換えてはなりません。また、すでにメッセージキューに入っているメッセージパケットを再度メールボックスに重複して送信してはなりません。いずれの場合も違反した場合には、メッセージキューが破壊され、致命的なエラーにつながります。

メールボックス機能に関連して、次のマクロを用意しています。

SIZE mprihdsz = TSZ_MPRIHD(PRI maxmpri)

送受信するメッセージの優先度の最大値が **maxmpri** のメールボックスに必要な優先度別メッセージキューヘッダ領域のサイズ (バイト数)

【推奨】

メッセージパケットとして、固定長／可変長メモリプールから動的に確保したメモリブロックを用いることも、静的に確保した領域を用いることも可能です。使い方としては、送信側のタスクがメモリプールからメモリブロックを確保し、それをメッセージパケットとして送信し、受信側のタスクはメッセージの内容を取り出した後にそのメモリブロックを直接メモリプールに返却する方法を推奨します。

3. 8 拡張同期・通信機能

拡張同期・通信機能は、タスクとは独立したオブジェクトにより、タスク間の高度な同期・通信を行うための機能です。ミューテックス、メッセージバッファ、ランデブの各機能が含まれます。

3. 8. 1 ミューテックス

ミューテックスは、共有資源を使用する際にタスク間で排他制御を行うためのオブジェクトです。具体的には、次の機能が含まれています。

- ・ ミューテックスを生成する機能 (cre_mtx, acre_mtx)
- ・ ミューテックスを削除する機能 (del_mtx)
- ・ ミューテックスをロックする機能 (loc_mtx, ploc_mtx, tloc_mtx)
- ・ ミューテックスをロック解除する機能 (unl_mtx)
- ・ ミューテックスの状態を参照する機能 (ref_mtx)

ミューテックスは、ロックされているかどうかの状態と、ロックを待つタスクの FIFO 順、あるいはタスク優先度順の待ち行列を持ちます。ミューテックスは、排他制御に伴う優先度逆転を防ぐための機構として、優先度継承プロトコル (priority inheritance protocol) と優先度上限プロトコル (priority ceiling protocol) をサポートします。

タスクは、資源を使用する前に、ミューテックスをロックします。ミューテックスが他のタスクにロックされていた場合には、ミューテックスがロック解除されるまで、あるいは許された時刻まで、ミューテックスのロック待ち状態に遷移します。ミューテックスのロック待ち状態になったタスクは、そのミューテックスの待ち行列につながれます。タスクは、資源の使用を終えると、ミューテックスのロックを解除します。

優先度上限プロトコルのミューテックスは、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中で最も高いベース優先度を持つタスクのベース優先度を、ミューテックス生成時に上限優先度として設定します。優先度上限プロトコルのミューテックスを、その上限優先度よりも高いベース優先度を持つタスクがロックしようとした場合、E_ILUSE エラーとなります。また、優先度上限プロトコルのミューテックスをロックしているかロックを待っているタスクのベース優先度を、chg_pri によってそのミューテックスの上限優先度よりも高く設定しようとした場合、chg_pri は E_ILUSE エラーを返します。

μ C3/Standard の優先度制御規則では、タスクの現在優先度を、次に挙げる優先度の最高値に常に一致するように変更します。

- ・ タスクのベース優先度
- ・ タスクが優先度継承プロトコルのミューテックスをロックしている場合、それらのミューテックスのロックを待っているタスクの中で、最も高い現在優先度を持つタスクの現在優先度
- ・ タスクが優先度上限プロトコルのミューテックスをロックしている場合、それらのミューテックス中で、最も高い上限優先度を持つミューテックスの上限優先度

ここで、優先度継承プロトコルのミューテックスを待っているタスクの現在優先度が、別のミューテックスによる操作か、`chg_pri` によるベース優先度の変更に伴い変更された場合、そのミューテックスをロックしているタスクは必要に応じて現在優先度を変更します。さらにそのタスクが、別の優先度上限プロトコルのミューテックスを待っていた場合には、そのミューテックスをロックしているタスクに対しても優先度の継承が行われ、現在優先度は変化します。

ミューテックスの操作に伴ってタスクの現在優先度を変更した場合には、次の処理を行います。優先度を変更されたタスクが実行できる状態である場合、タスクの優先順位を、変更後の優先度にしたがって変化させます。変更後の優先度と同じ優先度を持つタスクが存在する場合には、そのタスクよりも低い優先順位になります。優先度を変更されたタスクが何らかのタスク優先度順の待ち行列につながれている場合にも、その待ち行列の中での順序を、変更後の優先度にしたがって変化させます。タスクが終了する時に、そのタスクがロックしているミューテックスが残っている場合には、それらのミューテックスをすべてロック解除します。

FIFO 順またはタスク優先度順のミューテックスは、最大資源数が 1 のタスク間排他制御に用いるセマフォと同等の機能を持ちます。ただし、ミューテックスは、タスクコンテキスト以外から呼び出すことができない、ロックしたタスク以外はロック解除できない、タスク終了時に自動的にロック解除される、などの違いがあります。

3. 8. 2 メッセージバッファ

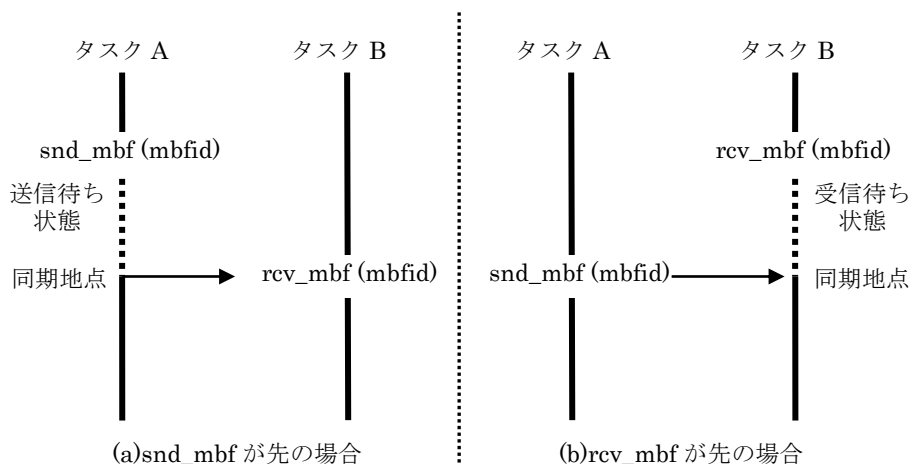
メッセージバッファは、可変長のメッセージを受渡しすることにより、同期と通信を行うためのオブジェクトです。具体的には、次の機能が含まれています。

- ・ メッセージバッファを生成する機能 (`cre_mbf,acre_mbf`)
- ・ メッセージバッファを削除する機能 (`del_mbf`)
- ・ メッセージバッファに対してメッセージを送信する機能 (`snd_mbf,psnd_mbf,tsnd_mbf`.)
- ・ メッセージバッファに対してメッセージを受信する機能 (`rcv_mbf,prcv_mbf,trev_mbf`)
- ・ メッセージバッファの状態を参照する機能 (`ref_mbf`)

メッセージバッファは、メッセージの送信を待つタスクの FIFO 順、あるいはタスク優先度順の送信待ち行列と、メッセージの受信を待つタスクの FIFO 順の受信待ち行列を持ちます。また、送信されたメッセージを格納するためのリング状のメッセージバッファ領域を持ちます。メッセージを送信する側では、送信したいメッセージをメッセージバッファにコピーします。メッセージバッファ領域の空き領域が足りなくなった場合、メッセージバッファ領域に十分な空きができるまで、あるいは許された時刻まで、メッセージバッファの送信待ち行列につながれ、メッセージバッファへの送信待ち状態に遷移します。一方、メッセージを受信する側では、メッセージバッファに入っているメッセージを一つ取り出します。メッセージバッファにメッセージが入っていない場合は、次にメッセージが送られてくるまで、あるいは許された時刻ま

で、メッセージバッファの受信待ち行列につなぐれ、メッセージバッファからの受信待ち状態に遷移します。

メッセージバッファ領域のサイズを 0 にすることで、同期メッセージ機能を実現することができます。つまり、送信側のタスクと受信側のタスクが、それぞれ相手のタスクがシステムコールを呼び出すのを待ち合わせ、双方がシステムコールを呼び出した時点で、メッセージの受け渡しが行われます。



メッセージバッファによる同期通信の図

メッセージバッファ機能に関連して、次のマクロを用意しています。

`SIZE mbfsz = TSZ_MBF(UINT msgcnt, UINT msgsz)`

サイズが `msgsz` バイトのメッセージを `msgcnt` 個格納するのに必要なメッセージバッファ領域のサイズ (目安のバイト数)

3. 8. 3 ランデブ

ランデブ機能は、タスク間で同期通信を行うための機能で、あるタスクから別のタスクへの処理依頼と、依頼されたタスクから依頼したタスクへ処理結果をと、情報の伝達をサポートします。この双方のタスクが待ち合わせるためのオブジェクトを、ランデブポートと呼びます。具体的には、次の機能が含まれています。

- ・ ランデブポートを生成する機能 (`cre_por`, `acre_por`)
- ・ ランデブポートを削除する機能 (`del_por`)
- ・ ランデブポートに対して処理の依頼を行う機能 (`cal_por`, `tcal_por`)
- ・ ランデブポートで処理依頼を受け付ける機能 (`acp_por`, `pacp_por`, `tacp_por`)
- ・ 処理結果を返す機能 (`fwd_por`)
- ・ 受け付けた処理依頼を他のランデブポートに回送する機能 (`rpl_rdv`)
- ・ ランデブポートおよびランデブの状態を参照する機能 (`ref_por`, `ref_rdv`)

ランデブポートは、処理の依頼を待つタスクの FIFO 順、あるいはタスク優先度順のランデブ呼出し待ち行列と、処理を依頼されるのを待つタスクの FIFO 順のランデブ受付待ち行列を

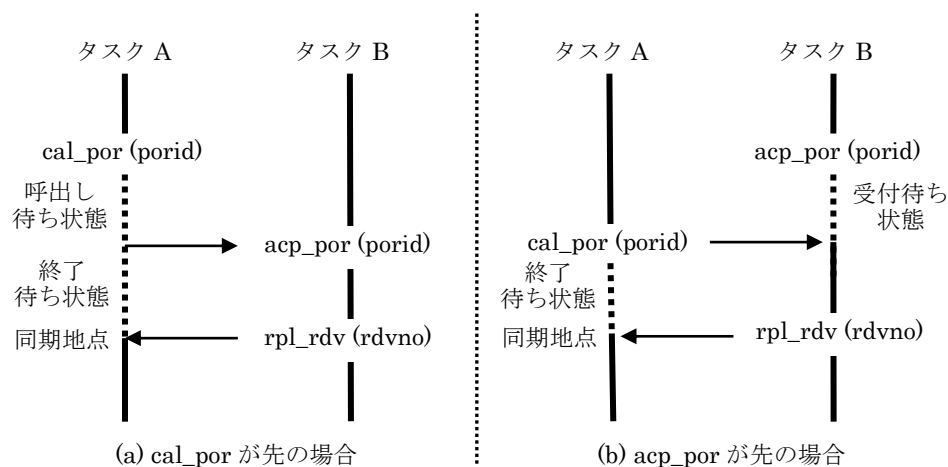
持ちます。

ランデブポートに処理を依頼するタスクは、ランデブポートとランデブ条件、依頼する処理に関する情報を入れた呼出しメッセージを指定して、ランデブの呼出しを行います。ランデブが成立しなければ、成立するまで、あるいは許された時刻までランデブ呼出し待ち行列につながれ、呼出し待ち状態に遷移します。成立した場合は、ランデブ呼出し待ち行列から切り離され、ランデブ終了待ち状態に遷移します。

一方、ランデブポートで処理を依頼されるタスクは、ランデブポートとランデブ条件を指定して、ランデブの受付を行います。ランデブが成立しなければ、成立するまで、あるいは許された時刻までランデブ受付待ち行列につながれ、受付待ち状態に遷移します。成立した場合は、呼出しメッセージを受け取り、呼出し待ち状態のタスクを待ち行列から切り離し、終了待ち状態へと遷移させます。この時、成立したランデブを識別するためのランデブ番号を割り付けます。

ランデブを受け付けたタスクが依頼された処理を完了すると、処理結果を返答メッセージの形で呼び出したタスクに渡し、ランデブを終了します。同時に、ランデブを呼び出したタスクはランデブ終了待ち状態から実行可能状態へ遷移します。

ランデブ条件は、ビットパターンで指定します。あるランデブポートに対して、呼び出したタスクのランデブ条件のビットパターンと、受け付けたタスクのランデブ条件のビットパターンをビット毎に論理積をとり、結果が 0 以外の場合にランデブが成立します。



ランデブ動作の図

3. 9 メモリプール管理機能

メモリプール管理機能は、ソフトウェアによって動的なメモリ管理を行うための機能で、固定長メモリプールと可変長メモリプールがあります。

3. 9. 1 固定長メモリプール

固定長メモリプールは、固定サイズのメモリブロックを動的に管理するためのオブジェクトです。具体的には、次の機能が含まれています。

- ・ 固定長メモリプールを生成する機能 (`cre_mpf`, `acre_mpf`)
- ・ 固定長メモリプールを削除する機能 (`del_mpf`)
- ・ 固定長メモリプールからメモリブロックを獲得する機能 (`get_mpf`, `pget_mpf`, `tget_mpf`)
- ・ 固定長メモリプールへメモリブロックを返却する機能 (`rel_mpf`)
- ・ 固定長メモリプールの状態を参照する機能 (`ref_mpf`)

固定長メモリプールは、固定長メモリプールとして利用するメモリ領域と、メモリブロックの獲得を待つタスクの **FIFO** 順、あるいはタスク優先度順の待ち行列を持ちます。固定長メモリプールからメモリブロックを獲得するタスクは、メモリプール領域に空きがなくなった場合には、メモリブロックが返却されるまで、あるいは許された時刻まで、待ち行列につながれ、固定長メモリブロックの獲得待ち状態に遷移します。

ユーザプログラムは、メモリブロックを返却する場合は、メモリブロックを獲得した同じ **ID** 番号の固定長メモリプールへメモリブロックを返却し、獲得したメモリブロックの先頭番地を、そのまま返却時に用いなければなりません。また、すでに返却されたメモリブロックを重複して返却してもいけません。いずれの場合も違反した場合には、固定長メモリプールの管理情報が破壊され、致命的なエラーにつながります。

固定長メモリプール機能に関連して、次のマクロを用意しています。

`SIZE mpfsz = TSZ_MPF(UINT blkcnt, UINT blksz)`

サイズが `blkksz` バイトのメモリブロックを `blkcnt` 個獲得するのに必要な固定長メモリプール領域のサイズ (バイト数)

3. 9. 2 可変長メモリプール

可変長メモリプールは、任意のサイズのメモリブロックを動的に管理するためのオブジェクトです。具体的には、次の機能が含まれています。

- ・ 可変長メモリプールを生成する機能 (`cre_mpl`, `acre_mpl`)
- ・ 可変長メモリプールを削除する機能 (`del_mpl`)
- ・ 可変長メモリプールからメモリブロックを獲得する機能 (`get_mpl`, `pget_mpl`, `tget_mpl`)
- ・ 可変長メモリプールへメモリブロックを返却する機能 (`rel_mpl`)
- ・ 可変長メモリプールの状態を参照する機能 (`ref_mpl`)

可変長メモリプールは、可変長メモリプールとして利用するメモリ領域と、メモリブロックの獲得を待つタスクの **FIFO** 順、あるいはタスク優先度順の待ち行列を持ちます。可変長メモリプールからメモリブロックを獲得するタスクは、メモリプール領域に空きがなくなった場合には、メモリブロックが返却されるまで、あるいは許された時刻まで、待ち行列につながれ、可変長メモリブロックの獲得待ち状態に遷移します。

ユーザプログラムは、メモリブロックを返却する場合は、メモリブロックを獲得した同じ ID 番号の可変長メモリプールへメモリブロックを返却し、獲得したメモリブロックの先頭番地を、そのまま返却時に用いなければなりません。また、すでに返却されたメモリブロックを重複して返却してもいけません。いずれの場合も違反した場合には、可変長メモリプールの管理情報が破壊され、致命的なエラーにつながります。

可変長メモリプール機能に関連して、次のマクロを用意しています。

SIZE mplsiz = TSZ_MPL(UINT blkcnt, UINT blksiz)

サイズが **blksiz** バイトのメモリブロックを **blkcnt** 個獲得するのに必要な可変長メモリプール領域のサイズ (目安のバイト数)

可変長メモリプールは、メモリブロックの獲得と返却を繰り返すことにより可変長メモリプールのメモリ領域のフラグメンテーション (断片化の意味) が進み、空き領域の合計サイズは十分あるのに連続したメモリがなくなる場合があります。

3. 10 時間管理機能

時間管理機能は、時間に依存した処理を行うための機能です。システム時刻管理、周期ハンドラ、アラームハンドラ、オーバランハンドラの各機能が含まれます。周期ハンドラ、アラームハンドラ、オーバランハンドラを総称して、タイムイベントハンドラと呼びます。

3. 10. 1 システム時刻管理

システム時刻管理機能は、システム時刻を操作するための機能です。具体的には、次の機能が含まれています。

- ・ システム時刻を設定／参照する機能 (`set_tim`, `get_tim`)
- ・ タイムチェックを供給してシステム時刻を更新する機能 (`isig_tim`)

システム時刻は、システム初期化時に 0 に初期化され、以降は、アプリケーションによってタイムチェックを供給するシステムコール (`isig_tim`) が呼び出される度に更新します。尚、システム時刻はミリ秒単位で管理され、オーバランハンドラ機能を除き、システムコールで指定される時間もすべてミリ秒単位です。

システム時刻を元にして、タイムアウト処理、`dly_tsk` による時間経過待ち状態からの解除、周期ハンドラやアラームハンドラの起動の各処理を行います。ただし、システム時刻の設定 (`set_tim`) によりシステム時刻を変更しても、すでに呼び出されたシステムコールのタイムアウト時刻などは変更されません。

3. 10. 2 周期ハンドラ

周期ハンドラは、一定周期で起動されるタイムイベントハンドラです。具体的には、次の機能が含まれています。

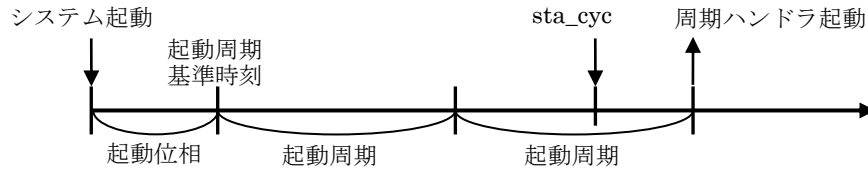
- ・ 周期ハンドラを生成する機能 (`cre_cyc`, `acre_cyc`)
- ・ 周期ハンドラを削除する機能 (`dle_cyc`)
- ・ 周期ハンドラの動作を開始する機能 (`sta_cyc`)
- ・ 周期ハンドラの動作を停止する機能 (`stp_cyc`)
- ・ 周期ハンドラの状態を参照する機能 (`ref_cyc`)

周期ハンドラは、動作している状態か動作していない状態かのいずれかの状態をとります。

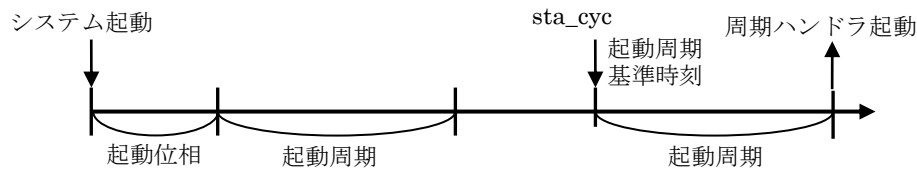
システム起動時には、`TA_STA` 属性を指定した場合では、動作している状態で生成されます。また、`TA_STA` 属性か `TA_PHS` 属性のいずれかを指定した場合には、システム起動時刻に起動位相を加えた時刻を、次に起動すべき時刻とします。いずれの属性も指定していない場合には、周期ハンドラの動作を開始するシステムコール (`sta_cyc`) が呼び出された時刻に起動周期を加えた時刻を、次に起動すべき時刻とします。周期ハンドラを起動すべき時刻になると、その周期ハンドラの拡張情報 (`exinf`) をパラメータとして、周期ハンドラを起動します。この時、

周期ハンドラの起動すべき時刻に起動周期を加えた時刻を、次に起動すべき時刻とします。

TA_PHS 属性を持ち、周期ハンドラが動作していない状態の時には、周期ハンドラを起動すべき時刻となっても周期ハンドラを起動せず、次に起動すべき時刻の決定のみを行います。周期ハンドラの動作を開始するシステムコール (sta_cyc) が呼び出されると、周期ハンドラを動作している状態に遷移させ、必要なら周期ハンドラを次に起動すべき時刻を決定しなおします。周期ハンドラの動作を停止するシステムコール (stp_cyc) が呼び出されると、周期ハンドラを動作していない状態に遷移させます。



(a) 起動位相を保存する場合 (TA_PHS 指定あり)



(b) 起動位相を保存しない場合 (TA_PHS 指定なし)

起動位相における保存の図

周期ハンドラの起動周期は、周期ハンドラを（起動した時刻ではなく）起動すべきであった時刻を基準に、周期ハンドラを次に起動する時刻を指定する相対時間です。そのため、周期ハンドラが起動される時刻の間隔は、個々には起動周期よりも短くなる場合がありますが、長い期間で平均すると起動周期に一致します。

周期ハンドラの n 回目の起動は、周期ハンドラを生成するシステムコールが呼び出されてから、(起動位相 + 起動周期 \times ($n - 1$)) 以上の時間が経過した後にを行うことを保証します。例えば、タイムチェックの周期が $10 \text{ } \mu\text{s}$ のシステムにおいて、起動位相が $15 \text{ } \mu\text{s}$ 、起動周期が $25 \text{ } \mu\text{s}$ の周期ハンドラを生成すると、周期起動ハンドラが起動されるシステム時刻は、 $20 \text{ } \mu\text{s}$ 、 $40 \text{ } \mu\text{s}$ 、 $70 \text{ } \mu\text{s}$ 、 $90 \text{ } \mu\text{s}$ 、 $120 \text{ } \mu\text{s}$ 、のようになります。

周期ハンドラは次の形式で記述します。

```
void cychdr(VP_INT exinf)
{
    周期ハンドラ本体
}
```

μ C3/Standard の制限として、起動周期にはチック時間より短い時間を指定することができません。

3. 10. 3 アラームハンドラ

アラームハンドラは、指定した時刻に起動されるタイムイベントハンドラです。具体的には、次の機能が含まれています。

- ・ アラームハンドラを生成する機能 (`cre_alm`, `acre_alm`)
- ・ アラームハンドラを削除する機能 (`del_alm`)
- ・ アラームハンドラの動作を開始する機能 (`sta_alm`)
- ・ アラームハンドラの動作を停止する機能 (`stp_alm`)
- ・ アラームハンドラの状態を参照する機能 (`ref_alm`)

アラームハンドラは、動作している状態か動作していない状態かのいずれかの状態をとります。アラームハンドラの起動時刻と呼ばれるアラームハンドラ起動する時刻を、アラームハンドラ毎に設定することができ、起動時刻になるとアラームハンドラの拡張情報 (`exinf`) をパラメータとしてアラームハンドラを起動します。アラームハンドラの生成直後は、アラームハンドラの起動時刻は設定されておらず、動作していない状態になります。

アラームハンドラの動作を開始するシステムコール (`sta_alm`) が呼び出されると、指定された相対時間後にアラームハンドラの起動時刻を設定し、動作している状態に遷移させます。このとき、すでに動作している状態だった場合は、アラームハンドラの起動時刻のみ再設定します。

アラームハンドラが起動する際には、アラームハンドラの起動時刻を解除し、動作していない状態にします。そのため、アラームハンドラ内で、アラームハンドラの動作を開始するシステムコール (`sta_alm`) を呼び出すこともできます。

アラームハンドラが動作状態の場合に、アラームハンドラの動作を停止するシステムコール (`stp_alm`) を呼び出すと、アラームハンドラの起動時刻を解除し、動作していない状態に遷移します。

アラームハンドラは次の形式で記述します。

```
void almhdr(VP_INT exinf)
{
    アラームハンドラ本体
}
```

3. 10. 4 オーバランハンドラ

オーバランハンドラは、タスクが設定された時間を越えてプロセッサを使用した場合に起動されるタイムイベントハンドラです。具体的には、次の機能が含まれています。

- ・ オーバランハンドラを定義する機能 (def_ovr)
- ・ オーバランハンドラの動作を開始する機能 (sta_ovr)
- ・ オーバランハンドラの動作を停止する機能 (stp_ovr)
- ・ オーバランハンドラの時刻を更新する機能 (ivsig_ovr)
- ・ オーバランハンドラの状態を参照する機能 (ref_ovr)

オーバランハンドラは全タスクで共通ですが、使用プロセッサ時間を計時するタイマはタスク毎に持っています。この使用プロセッサ時間とは、オーバランハンドラの動作を開始してから、タスクがプロセッサを占有していた時間の累積です。

オーバランハンドラは、使用プロセッサ時間がオーバランハンドラの起動時に設定される上限プロセッサ時間を超えた場合に、そのタスクの ID 番号と拡張情報 (exinf) をパラメータとして起動します。

使用プロセッサ時間と上限プロセッサ時間には、任意の時間単位を用い、システム設計で要求された時間精度の間隔で、オーバランハンドラの時刻を更新するシステムコール (ivsig_ovr) を呼び出します。つまり、オーバランハンドラの時刻を更新するシステムコールを呼び出すことによって、オーバランハンドラが動作している状態の使用プロセッサ時間に 1 (時間単位) を加算します。

オーバランハンドラの動作を開始するシステムコール (sta_ovr) が呼び出されると、指定された時間を上限プロセッサ時間に設定し、使用プロセッサ時間を 0 にクリアして動作している状態に遷移します。すでに動作状態の場合に、オーバランハンドラの動作を開始するシステムコールを呼び出した場合は、再設定します。

オーバランハンドラは次の形式で記述します。

```
void ovrhdr(IDtskid, VP_INTexinf)
{
    オーバランハンドラ本体
}
```

3. 1 1 システム状態管理機能

システム状態管理機能は、システムの状態を変更／参照するための機能です。具体的には、次の機能が含まれています。

- ・ タスクの優先順位を回転する機能 (rot_rdq, irot_rdq)
- ・ 実行状態のタスク ID を参照する機能 (get_tid, iget_tid)
- ・ CPU ロック状態へ移行／解除する機能 (loc_cpu, iloc_cpu, unl_cpu, iunl_cpu)
- ・ タスクディスパッチを禁止／解除する機能 (dis_dsp, ena_dsp)
- ・ コンテキストやシステム状態を参照する機能 (sns_ctx, sns_loc, sns_dsp, sns_dpn, ref_sys)

3. 1 2 割込み管理機能

割込み管理機能は、外部割込みによって起動される割込みサービスルーチンを管理するための機能です。具体的には、次の機能が含まれています。

- ・ 割込みハンドラを定義する機能 (def_inh)
- ・ 割込みサービスルーチンを生成／削除する機能 (cre_isr, acre_isr, del_isr)
- ・ 割込みサービスルーチンの状態を参照する機能 (ref_isr)
- ・ 割込みを禁止／許可する機能 (dis_int, ena_int)
- ・ 割込みマスクを変更／参照する機能 (chg_ims, get_ims)

割込み管理機能では、次のデータ型を用います。

INHNO	割込みハンドラ番号
INTNO	割込み番号
IMASK	割込みマスク

μ C3/Standard では、割込みハンドラ番号と割込み番号は同じ意味合いを持ち、同じ割込み番号の割込みに対して、割込みハンドラと割込みサービスルーチンを定義または生成できません。割込み番号の割り付けの説明は「プロセッサ依存部マニュアル」を参照してください。

割込みマスクのデータ型 IMASK は、プロセッサのアーキテクチャにより内容が異なります。また、割込みを禁止／許可する機能 (dis_int, ena_int) の実装は、プロセッサにより異なります。これらの説明は「プロセッサ依存部マニュアル」を参照してください。

割込みハンドラは次の形式で記述します。

```
void inhhdr(void)
{
    割込みハンドラの本体
}
```

割込みサービスルーチンを起動する際には、その割込みサービスルーチンの拡張情報 (exinf) をパラメータとして渡します。割込みサービスルーチンは次の形式で記述します。

```
void isr(VP_INT exinf)
{
    割込みサービスルーチン本体
}
```

3. 13 システム構成管理機能

システム構成管理機能は、システムのコンフィグレーションやバージョン情報を管理するための機能です。具体的には、次の機能が含まれています。

- CPU 例外ハンドラを定義する機能 (def_exc)
- コンフィグレーションを参照する機能 (ref_cfg)
- バージョン情報を参照する機能 (ref_ver)

システム構成管理機能では、次のデータ型を用います。

EXCNO CPU 例外ハンドラ番号

CPU 例外ハンドラの実装はプロセッサにより異なり、実装されていないものや制限のあるものがあります。これらの説明は「プロセッサ依存部マニュアル」を参照してください。

CPU 例外ハンドラは、次の形式で記述します。

```
void exchdr(void)
{
    CPU 例外ハンドラの本体
}
```

3. 1 4 独自機能

μ C3/Standard の独自の機能として、デバイスドライバ管理とエラーハンドラがあります。

3. 1 4. 1 デバイスドライバ管理

デバイスドライバ管理機能は、標準 COM ドライバの管理と制御するための機能です。具体的には、次の機能が含まれています。

- デバイスドライバを定義する機能 (vdef_dev)
- デバイスドライバを制御する機能 (vctr_dev)

デバイスドライバは定義することで、その ID 番号とデバイスドライバの起動番地を関連づけます。さらに、制御では、機能コードと制御内容を伴ってデバイスドライバを呼び出します。

デバイスドライバは次の形式で記述します。デバイスドライバの ID 番号から得られる制御情報 packets `ctrblk` と、機能コード `funcid` とデバイス制御情報 packets `ctrdev` をパラメータとしてデバイスドライバを呼び出します。

```
void devhdr(ID funid, VP ctrdev, VP ctrblk)
{
    デバイスドライバの本体
}
```

3. 1 4. 2 エラーハンドラ

エラーハンドラは、遅延実行されたシステムコールのエラーを検出する機能です。具体的には、次の機能が含まれています。

- ・エラーハンドラを定義する機能 (vdef_err)

エラーハンドラは、割込みハンドラから遅延実行するシステムコールを呼び出した場合で、実行時にエラー (E_OK 以外) が発生した場合に呼び出されます。

エラーハンドラは次の形式で記述します。対象となるシステムコールの機能コード `funcn`、発生したエラーコード `ercd`、システムコールを呼び出した際の引数の配列 `para` をパラメータとして呼び出します。

```
void err_hdl(FN funcn, ER ercd, VP_INT *para);
{
    エラーハンドラの本体
}
```

機能コードの種類には、以下があります。

機能コード	値	関数名
TFN_ACT_TSK	-0x07	iact_tsk (act_tsk)
TFN_STA_TSK	-0x09	sta_tsk
TFN_SET_FLG	-0x2B	iset_flg (set_flg)
TFN_SIG_SEM	-0x23	isig_sem (sig_sem)
TFN_PSND_DTQ	-0x36	ipsnd_dtq (psnd_dtq)
TFN_WUP_TSK	-0x13	iwup_tsk (wup_tsk)
TFN_REL_WAI	-0x15	irel_wai (rel_wai)

第4章 コンフィグレーションとシステムの起動

4. 1 カーネルの起動

μ C3 カーネルは、main 関数（あるいは、これに類する関数）内でカーネルの起動関数を呼び出すことにより、システムが初期化されマルチタスクが始動します。

start_uC3

カーネルの起動

【書式】

```
ER ercd = start_uC3(T_CSYS *pk_csys, FP inihdr);
```

【パラメータ】

T_CSYS *	pk_csys	システム生成情報を入れたパケットへのポインタ
FP	inihdr	初期化ハンドラの起動番地
pk_csys の内容 (T_CSYS 型)		
(4. 3 カーネルのコンフィグレーションを参照)		

【戻り値】

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

【エラーコード】

E_PAR	パラメータエラー (pk_csysが不正)
E_NOMEM	メモリ不足 (オブジェクトの管理領域などが確保できない)

【解説】

pk_csys で指定されるシステム生成情報に基づいてカーネルを生成し、inihdr で指定された初期化ハンドラの実行後にマルチタスクに遷移します。即ち、inihdr で必要最小限のオブジェクトが生成され、初期化ハンドラの終了後、レディ状態のタスクが存在する場合は、その中のタスク優先度が一番高いタスクに制御が移ります。また、inihdr は非タスクコンテキストの割込み許可状態で実行されます。

4. 2 システムスタック

μ C3/Standard では、非タスクコンテキストはすべてシステムスタックで実行されます。そのため、システムスタックのサイズは、一番多くスタック領域を消費するタイムイベントハンドラのスタックサイズと、割込み処理（割込みハンドラや割込みサービスルーチン）で消費するスタックサイズを加算した値が目安になります。また、多重割込みが想定される場合には、各割込みレベルで一番多くスタックを消費する割込みハンドラのスタックサイズを加算します。

4. 3 カーネルのコンフィグレーション

μ C3 カーネルは、カーネルの起動関数のシステム生成情報を基に、すべてのオブジェクトを動的にコンフィグレーションします。

T_CSYS型の内容

tskpri_max	タスク優先度の最大値 (1 ~ 31)
tskid_max	タスクIDの最大値 (1 ~ 255)
semid_max	セマフォIDの最大値 (0 ~ 999)
flgid_max	イベントフラグIDの最大値 (0 ~ 999)
dtqid_max	データキューIDの最大値 (0 ~ 999)
mbxid_max	メールボックスIDの最大値 (0 ~ 999)
mtxid_max	ミューテックスIDの最大値 (0 ~ 999)
mbfid_max	メッセージバッファIDの最大値 (0 ~ 999)
porid_max	ランデブポートIDの最大値 (0 ~ 999)
mpfid_max	固定長メモリプールIDの最大値 (0 ~ 999)
mplid_max	可変長メモリプールタスクIDの最大値 (0 ~ 999)
almid_max	アラームハンドラIDの最大値 (0 ~ 999)
cycid_max	周期ハンドラIDの最大値 (0 ~ 999)
isrid_max	割込みサービスルーチンIDの最大値 (0 ~ 999)
devid_max	デバイスドライバIDの最大値 (0 ~ 999)
tick	チック時間 (1/1000秒単位)
ssb_num	システムサービスブロックの個数
sysmem_top	システムメモリ領域の先頭番地
sysmem_end	システムメモリ領域の最終番地 (末尾番地 + 1 を指定)
stkmem_top	スタック用メモリ領域の先頭番地
stkmem_end	スタック用メモリ領域の最終番地 (末尾番地 + 1 を指定)
mplmem_top	メモリプール用メモリ領域の先頭番地
mplmem_end	メモリプール用メモリ領域の最終番地 (末尾番地 + 1 を指定)
ctrtim	タイマ制御関数 (将来拡張用)
sysidl	アイドル処理
	SYSTEM_IDLE システムが用意するアイドル処理
	USER_IDLE (func) ユーザ定義のアイドル関数
inistk	スタック初期化処理
	STACK_NO_INIT 初期化なし
	STACK_ZERO_INIT 0 による初期化
	STACK_ID_INIT タスクIDによる初期化
	STACK_USER_INIT(func) ユーザ定義の初期化関数

trace	トレース	
	TRACE_DISABLE	トレースなし
agent	エージェント	
	AGENT_DISABLE	エージェントの起動なし

第5章 システムコールの説明

5. 1 タスク管理機能

cre_tsk	タスクの生成
acre_tsk	タスクの生成 (ID 番号自動割付け)

【書式】

```
ER ercd = cre_tsk( ID tskid, T_CTSK *pk_ctsk );
```

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

【パラメータ】

ID	tskid	生成対象のタスクの ID 番号 (acre_tsk 以外)
T_CTSK *	pk_ctsk	タスク生成情報を入れたパケットへのポインタ
pk_ctsk の内容 (T_CTSK 型)		
ATR	tskatr	タスク属性
VP_INT	exinf	タスクの拡張情報
FP	task	タスクの起動番地
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)
VP	stk	タスクのスタック領域の先頭番地
VB const *	name	タスクの名称 (文字列)

【戻り値】

cre_tsk の場合

ER ercd 正常終了 (E_OK) またはエラーコード

acre_tsk の場合

ER_ID tskid 生成したタスクの ID 番号 (正の値) またはエラーコード

【エラーコード】

E_PAR	パラメータエラー (itskpri が不正)
E_ID	不正 ID 番号 (tskid が不正あるいは使用できない ; cre_tsk のみ)
E_CTX	コンテキストエラー (タスクと初期化ハンドラ以外からの呼び出し)
E_NOMEM	メモリ不足 (スタック領域などが確保できない)
E_NOID	ID 番号不足 (割り付け可能なタスク ID がない ; acre_tsk のみ)
E_OBJ	オブジェクト状態エラー (対象タスクが登録済み ; cre_tsk のみ)

【呼び出しコンテキスト】	cre_tsk	acre_tsk
タスク	可	可
初期化ハンドラ	可	可
タイムイベントハンドラ	不可	不可
割込みハンドラ	不可	不可

【解説】

tskid で指定される ID 番号のタスクを、pk_ctsk で指定されるタスク生成情報に基づいて生成します。具体的には、対象タスクを未登録状態から休止状態、または実行可能状態に遷移させます。tskatr はタスクの属性、exinf はタスクを起動する際にパラメータとして渡す拡張情報、task はタスクの起動番地、itskpri はタスクの起動時優先度、stksz はスタック領域のサイズ（バイト数）、stk はスタック領域の先頭番地です。

acre_tsk は、生成するタスクの ID 番号をタスクが登録されていない ID 番号の中から一番大きな値を割り付け、その ID 番号を戻り値として返します。

tskatr には、(TA_HLNG | [TA_ACT]) の指定ができます。TA_ACT (=0x02) を指定した場合には、対象タスクを実行可能状態に遷移させ、タスクの拡張情報をパラメータとして渡しタスクを起動します。

stk で指定された番地から stksz バイトのメモリ領域を、タスクを実行するためのスタック領域として使用します。stk に NULL (=0) が指定された場合には、stksz で指定されたサイズのメモリ領域を、コンフィグレーションで定義したスタック用メモリ領域から自動で確保します。

del_tsk**タスクの削除****【書式】**

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

【パラメータ】

ID	tskid	削除対象のタスクの ID 番号
----	-------	-----------------

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (tskid が不正あるいは使用できない)
E_CTX	コンテキストエラー (タスク以外からの呼び出し)
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)
E_NOEXS	オブジェクト未生成 (対象タスクが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	不可
タイムイベントハンドラ	不可
割込みハンドラ	不可

【解説】

tskid で指定される ID 番号のタスクを削除します。つまり、対象タスクを休止状態から未登録状態に遷移させます。

対象タスクが休止状態で無い場合には、E_OBJ エラーを返し、対象タスクが未登録状態の場合には、E_NOEXS エラーを返します。

act_tsk タスクの起動

iact_tsk

【書式】

ER ercd = act_tsk (ID tskid);

ER ercd = iact_tsk (ID tskid);

【パラメータ】

ID	tskid	起動対象のタスクの ID 番号
----	-------	-----------------

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (tskid が不正、あるいは使用できない)
E_NOMEM	メモリ不足 (SSB が不足：割込みハンドラからの呼び出しのみ)
E_NOEXS	オブジェクト未生成 (対象タスクが未登録)
E_QOVR	キューイングオーバーフロー (起動要求キューイング数のオーバーフロー)

【呼び出しコンテキスト】

	act_tsk	iact_tsk
タスク	可	可
初期化ハンドラ	可	可
タイムイベントハンドラ	可	可
割込みハンドラ	可	可

【解説】

tskid で指定される ID 番号のタスクを起動します。つまり、対象タスクを休止状態から実行可能状態に遷移させます。タスクを起動する際のパラメータとして、タスクの拡張情報を渡します。

対象タスクが休止状態でない場合には、タスクに対する起動要求をキューイングします。具体的には、タスクの起動要求キューイング数に 1 を加えると起動要求キューイング数の最大値を超える場合には、E_QOVR エラーを返します。ただし、割込みハンドラからの呼び出しでは遅延実行するため、戻り値のエラーコードでは E_QOVR エラーを検出できません。TSK_SELF (=0) が指定されると、自タスクを対象タスクとします。ただし、非タスクコンテキストからの呼び出しでこの指定が行われた場合には、E_ID エラーを返します。

【推奨】

μ C3/Standard の act_tsk と iact_tsk は、同じシステムコールとして実装するため、呼び出しコンテキストに関係なく、同じ使い方ができます。ただし、タスクコンテキストから呼び出す場合には act_tsk を、それ以外の場合には iact_tsk を使うことをお勧めします。

can_act**タスク起動要求のキャンセル****【書式】**

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

【パラメータ】

ID	tskid	起動要求のキャンセル対象のタスクの ID 番号
----	-------	-------------------------

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号（tskid が不正、あるいは使用できない）
E_CTX	コンテキストエラー（割込みハンドラからの呼び出し）
E_NOEXS	オブジェクト未生成（対象タスクが未登録）

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

tskid で指定される ID 番号のタスクに対し、起動要求キューイング数をクリアし、クリアする前の起動要求キューイング数を返します。

tskid に TSK_SELF (=0) が指定されると、自タスクを対象タスクとします。ただし、非タスクコンテキストからの呼び出しで、この指定が行われた場合には、E_ID エラーを返します。

sta_tsk

タスクの起動（起動コード指定）

【書式】

```
ER ercd = sta_tsk (ID tskid, VP_INT stacd);
```

【パラメータ】

ID	tskid	起動対象のタスクの ID 番号
VP_INT	stacd	タスクの起動コード

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (tskid が不正、あるいは使用できない)
E_NOMEM	メモリ不足 (SSB が不足：割込みハンドラからの呼び出しのみ)
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)
E_NOEXS	オブジェクト未生成 (対象タスクが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	可

【解説】

tskid で指定される ID 番号のタスクを起動します。つまり、対象タスクを休止状態から実行可能状態に遷移させます。また、タスクを起動する際のパラメータとして、起動コード (stacd) を渡します。

対象タスクが休止状態でない場合には、タスクに対する起動要求をキューイングせず、E_OBJ エラーを返します。ただし、割込みハンドラからの呼び出しでは遅延実行するため、戻り値のエラーコードでは E_OBJ エラーを検出できません。

【推奨】

sta_tsk は、μ ITRON3.0 仕様との互換性のために存在しています。μ C3/Standard では、sta_tsk を使わずに act_tsk, iact_tsk を使うことをお勧めします。

ext_tsk

自タスクの終了

【書式】

```
void ext_tsk ();
```

【パラメータ】なし

【戻り値】なし

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	不可
タイムイベントハンドラ	不可
割込みハンドラ	不可

【解説】

自タスクを終了させます。つまり、自タスクを実行状態から休止状態に遷移させます。自タスクの起動要求キューイング数が1以上の場合には、起動要求キューイング数から1を減算し、自タスクを実行可能状態に遷移させます。この時、タスクの起動時に行うべき処理として、タスク優先度の初期化、起床要求カウンタ数もクリア、スタックポインタの初期化を行います。タスクを起動する際のパラメータとして、タスクの拡張情報を渡します。

タスクコンテキストから呼び出した場合には、このシステムコールから戻ることはありません。しかし、非タスクコンテキストから呼び出した場合には、エラーコードを返さずに戻ります。

exd_tsk

自タスクの終了と削除

【書式】

```
void exd_tsk ( );
```

【パラメータ】

なし

【戻り値】

このシステムコールからはリターンしない

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	不可
タイムイベントハンドラ	不可
割込みハンドラ	不可

【解説】

自タスクを終了させ、さらに自タスクを削除します。つまり、自タスクを実行状態から未登録状態に遷移させ、タスクの終了時に行うべき処理と削除時に行うべき処理を行います。

タスクコンテキストから呼び出した場合には、このシステムコールから戻ることはありません。しかし、非タスクコンテキストから呼び出した場合には、エラーコードを返さずに戻ります。

ter_tsk**タスクの強制終了****【書式】**

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

【パラメータ】

ID	tskid	強制終了対象のタスクの ID 番号
----	-------	-------------------

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (tskid が不正、あるいは使用できない)
E_CTX	コンテキストエラー (タスク以外からの呼び出し)
E_ILUSE	システムコール不正使用 (対象タスクが自タスク)
E_OBJ	オブジェクト状態エラー (対象タスクが休止状態)
E_NOEXS	オブジェクト未生成 (対象タスクが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	不可
タイムイベントハンドラ	不可
割込みハンドラ	不可

【解説】

tskid で指定される ID 番号のタスクを、強制的に休止状態に遷移させます。ただし、対象タスクの起動要求キューイング数が 1 以上の場合には、起動要求キューイング数から 1 を減算し、実行可能状態に遷移させます。この時、タスクの起動時に行うべき処理として、タスク優先度の初期化、起床要求カウンタ数もクリア、スタックポインタの初期化を行います。タスクを起動する際のパラメータとして、タスクの拡張情報を渡します。

対象タスクが休止状態の時は E_OBJ エラーを返します。また、このシステムコールは自タスクを終了させることはできません。対象タスクが自タスクの場合には、E_ILUSE エラーを返します。

chg_pri**タスク優先度の変更****【書式】**

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

【パラメータ】

ID	tskid	変更対象のタスクの ID 番号
PRI	tskpri	変更後のベース優先度

【戻り値】

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

【エラーコード】

E_PAR	パラメータエラー (tskpri が不正。または、タイムイベントハンドラから tskpri に TPRI_INI (=0) を指定して呼び出した場合)
E_ID	不正 ID 番号 (tskid が不正、あるいは使用できない。または、タイムイベントハンドラから tskid に TSK_SELF (=0) を指定して呼び出した場合)
E_CTX	コンテキストエラー (割込みハンドラからの呼び出し)
E_ILUSE	システムコール不正使用 (解説を参照)
E_OBJ	オブジェクト状態エラー (対象タスクが休止状態)
E_NOEXS	オブジェクト未生成 (対象タスクが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

tskid で指定される ID 番号のタスクのベース優先度を、tskpri で指定される値に変更します。tskid に TSK_SELF (=0) が指定されると、自タスクを対象タスクとします。また、tskpri に TPRI_INI (=0) が指定されると、対象タスクのベース優先度をタスクの起動時優先度に変更します。ミューテックスをロックしているため、自タスクの現在優先度が μ C3/Standard の優先度制御規則に合致しなくなった場合には、適合するように自タスク以外も含め現在優先度を変更します。

対象タスクが実行できる状態で現在優先度を変更した場合、タスクの優先順位を、変更後の優先度にしたがって変化させます。変更後の優先度と同じ優先度を持つタスクの間では、対象タスクの優先順位を最低にします。

ミューテックスの優先度上限プロトコルをロックしているか、ロックを待っており、これら

の上限優先度よりも高くなる場合には、E_ILUSE エラーを返します。

get_pri**タスク優先度の参照****【書式】**

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

【パラメータ】

ID	tskid	参照対象のタスクの ID 番号
----	-------	-----------------

【戻り値】

ER	ercd	正常終了 (E_OK) またはエラーコード
PRI	tskpri	対象タスクの現在優先度

【エラーコード】

E_ID	不正 ID 番号 (tskid が不正、あるいは使用できない)
E_CTX	コンテキストエラー (割込みハンドラからの呼び出し)
E_OBJ	オブジェクト状態エラー (対象タスクが休止状態)
E_NOEXS	オブジェクト未生成 (対象タスクが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

tskid で指定される ID 番号のタスクの現在優先度を参照し、tskpri に返します。

tskid に TSK_SELF (=0) が指定されると、自タスクを対象タスクとします。

【使い方】

```
PRI tskpri;          /* タスクの現在優先度を格納する領域を確保 */
```

```
ER ercd;
```

```
/* 格納する領域へのポインタをパラメータにして呼び出す */
```

```
ercd = get_pri(ID_Task1, &tskpri);
```

ref_tsk**タスクの状態参照****【書式】**

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

【パラメータ】

ID	tskid	参照対象のタスクの ID 番号
T_RTsk *	pk_rtsk	タスク状態を返すパケットへのポインタ

【戻り値】

ER	ercd	正常終了 (E_OK) またはエラーコード
pk_rtsk の内容 (T_RTsk 型)		
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	待ち要因
ID	wobjid	待ち対象のオブジェクトの ID 番号
TMO	lefttmo	タイムアウトするまでの時間
UINT	actcnt	起動要求キューイング数
UINT	wupcnt	起床要求キューイング数
UINT	suscnt	強制待ち要求ネスト数

【エラーコード】

E_ID	不正 ID 番号 (tskid が不正、あるいは使用できない)
E_CTX	コンテキストエラー (割込みハンドラからの呼び出し)
E_NOEXS	オブジェクト未生成 (対象タスクが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

tskid で指定される ID 番号のタスクに関する状態を参照し、pk_rtsk で指定されるパケットに返します。

tskstat には、対象タスクの状態によって、次のいずれかの値を返します。

TTS_RUN	0x01	実行状態
TTS_RDY	0x02	実行可能状態
TTS_WAI	0x04	待ち状態

TTS_SUS	0x08	強制待ち状態
TTS_WAS	0x0C	二重待ち状態
TTS_DMT	0x10	休止状態

対象タスクが休止状態でない場合に、**tskpri** には対象タスクの現在優先度、**tskbpri** にはベース優先度を返します。対象タスクが休止状態の場合には、これらには不定値を返します。

対象タスクが待ち状態の場合の **tskwait** には、対象タスクが待ち状態になっている要因によって、次のいずれかの値を返します。対象タスクが待ち状態でない場合には、不定値を返します。

TTW_SLP	0x0001	起床待ち状態
TTW_DLY	0c0002	時間経過待ち状態
TTW_SEM	0x0004	セマフォ資源の獲得待ち状態
TTW_FLG	0x0008	イベントフラグ待ち状態
TTW_SDTQ	0x0010	データキューへの送信待ち状態
TTW_RDTQ	0x0020	データキューからの受信待ち状態
TTW_MBX	0x0040	メールボックスからの受信待ち状態
TTW_MTX	0x0080	ミューテックスのロック待ち状態
TTW_SMBF	0x0100	メッセージバッファへの送信待ち状態
TTW_RMBF	0x0200	メッセージバッファからの受信待ち状態
TTW_CAL	0x0400	ランデブの呼出し待ち状態
TTW_ACP	0x0800	ランデブの受付待ち状態
TTW_RDV	0x1000	ランデブの終了待ち状態
TTW_MPF	0x2000	固定長メモリブロックの獲得待ち状態
TTW_MPL	0x4000	可変長メモリブロックの獲得待ち状態

対象タスクが待ち状態で、起床待ち状態、時間経過待ち状態のいずれでもない場合の **wobjid** には、待ち対象のオブジェクトの ID 番号を返します。それ以外の場合の **wobjid** には不定値を返します。

対象タスクが待ち状態で、時間経過待ち状態でない場合の **lefttmo** には、対象タスクがタイムアウトするまでの時間を返します。具体的には、タイムアウトとなる時刻から現在時刻を減じた値を返します。ただし、**lefttmo** に返す値は、タイムアウトするまでの保証される時間となり、実際にタイムアウトまでの時間よりも小さくなります。そのため、次のタイムチェックでタイムアウトする場合には、**lefttmo** に 0 を返します。対象タスクが永久待ち（タイムアウトなし）で待ち状態になっている場合には、**lefttmo** に **TMO_FEVR** を返します。対象タスクが待ち状態でないか、時間経過待ち状態である場合の **lefttmo** には不定値を返します。

actcnt には対象タスクの起動要求キューイング数を、**wupcnt** には対象タスクの起床要求キューイング数を、**suscnt** には強制待ち要求ネスト数を返します。

tskid に **TSK_SELF** (=0) が指定されると、自タスクを対象タスクとします。

【使い方】

```
T_RTsk rtsk;          /* タスクの状態を格納する領域を確保 */  
ER ercd;  
  
          /* 格納する領域へのポインタをパラメータにして呼び出す */  
ercd = ref_tsk (ID_Task1, & rtsk);
```

ref_tst**タスクの状態参照（簡易版）****【書式】**

```
ER ercd = ref_tst( ID tskid, T_RTST *pk_rtst );
```

【パラメータ】

ID	tskid	参照対象のタスクの ID 番号
T_RTST *	pk_rtst	タスク状態を返すパケットへのポインタ

【戻り値】

ER	ercd	正常終了 (E_OK) またはエラーコード
pk_rtst の内容 (T_RTST 型)		
STAT	tskstat	タスク状態
STAT	tskwait	待ち要因

【エラーコード】

E_ID	不正 ID 番号 (tskid が不正あるいは使用できない)
E_CTX	コンテキストエラー (割込みハンドラからの呼び出し)
E_NOEXS	オブジェクト未生成 (対象タスクが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

tskid で指定される ID 番号のタスクに関する最低限の状態を参照し、pk_rtst で指定されるパケットに返します。このシステムコールは、ref_tsk の簡易版として実装され、tskstat と tskwait には、ref_tsk と同じ値を返します。

tskid に TSK_SELF (=0) が指定されると、自タスクを対象タスクとします。

5. 2 タスク付属同期機能

slp_tsk	起床待ち
tslp_tsk	起床待ち（タイムアウトあり）

【書式】

```
ER ercd = slp_tsk ( );
ER ercd = tslp_tsk (TMO tmout);
```

【パラメータ】

TMO	tmout	タイムアウト指定 (tslp_tsk のみ)
-----	-------	------------------------

【戻り値】

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

【エラーコード】

E_PAR	パラメータエラー (tmout が不正 ; tslp_tsk のみ)
E_CTX	コンテキストエラー (タスク以外、ディスパッチ保留状態)
E_RLWAI	待ち状態の強制解除 (待ち状態の間に rel_wai を受付)
E_TMOUT	ポーリング失敗またはタイムアウト (tslp_tsk)

【呼び出しコンテキスト】

	slp_tsk	tslp_tsk
タスク	可	可
初期化ハンドラ	不可	不可
タイムイベントハンドラ	不可	不可
割込みハンドラ	不可	不可

【解説】

自タスクを起床待ち状態に遷移させます。ただし、自タスクの起床要求キューイング数が 1 以上の場合には、起床要求キューイング数から 1 を減算し、自タスクを待ち状態に遷移することなく実行し続けます。

tslp_tsk は、slp_tsk にタイムアウトの機能を付け加えたシステムコールです。また、tmout に、TMO_POL (=0) や TMO_FEVR (= -1) を指定することもできます。μC3/Standard では、tmout に TMO_FEVR を指定した tslp_tsk は slp_tsk として扱われます。

wup_tsk タスクの起床

iwup_tsk

【書式】

```
ER ercd = wup_tsk (ID tskid);
```

```
ER ercd = iwup_tsk (ID tskid);
```

【パラメータ】

ID	tskid	起床対象のタスクの ID 番号
----	-------	-----------------

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (tskid が不正、あるいは使用できない)
E_NOMEM	メモリ不足 (SSB が不足：割込みハンドラからの呼び出しのみ)
E_OBJ	オブジェクト状態エラー (対象タスクが休止状態)
E_NOEXS	オブジェクト未生成 (対象タスクが未登録)
E_QOVR	キューイングオーバーフロー (起床要求キューイング数のオーバーフロー)

【呼び出しコンテキスト】

	wup_tsk	iwup_tsk
タスク	可	可
初期化ハンドラ	可	可
タイムイベントハンドラ	可	可
割込みハンドラ	可	可

【解説】

tskid で指定される ID 番号のタスクを、起床待ち状態から待ち解除します。待ち解除されたタスクには、待ち状態に入ったシステムコールの戻り値として E_OK を返します。対象タスクが起床待ち状態でも休止状態でもない場合には、タスクの起床要求キューイング数に 1 を加算します。ただし、タスクの起床要求キューイング数に 1 を加算すると起床要求キューイング数の最大値を超える場合には、E_QOVR エラーを返します。

対象タスクが休止状態の場合には、E_OBJ エラーを返します。ただし、割込みハンドラからの呼び出しでは遅延実行するため、戻り値のエラーコードでは E_OBJ エラーと E_QOVR エラーを検出できません。

tskid に TSK_SELF (=0) が指定されると、自タスクを対象タスクとします。ただし、非タスクコンテキストからの呼出しでこの指定が行われた場合には、E_ID エラーを返します。

【推奨】

μ C3/Standard の `wup_tsk` と `iwup_tsk` は、同じシステムコールとして実装するため、呼び出しコンテキストに関係なく、同じ使い方ができます。ただし、タスクコンテキストから呼び出す場合には `wup_tsk` を、それ以外の場合には `iwup_tsk` を使うことをお勧めします。

can_wup**タスク起床要求のキャンセル****【書式】**

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

【パラメータ】

ID	tskid	起床要求のキャンセル対象のタスクの ID 番号
----	-------	-------------------------

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号（tskid が不正、あるいは使用できない）
E_CTX	コンテキストエラー（割込みハンドラからの呼び出し）
E_OBJ	オブジェクト状態エラー（対象タスクが休止状態）
E_NOEXS	オブジェクト未生成（対象タスクが未登録）

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

tskid で指定される ID 番号のタスクに対し、起床要求キューイング数をクリアし、クリアする前の起床要求キューイング数を戻り値として返します。

対象タスクが休止状態の場合には、E_OBJ エラーを返します。

tskid に TSK_SELF (=0) が指定されると、自タスクを対象タスクとします。

rel_wai	待ち状態の強制解除
irel_wai	

【書式】

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

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

【パラメータ】

ID	tskid	待ち状態の強制解除対象のタスクの ID 番号
----	-------	------------------------

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (tskid が不正、あるいは使用できない)
E_NOMEM	メモリ不足 (SSB が不足：割込みハンドラからの呼び出しのみ)
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)
E_NOEXS	オブジェクト未生成 (対象タスクが未登録)

【呼び出しコンテキスト】

	rel_wai	irel_wai
タスク	可	可
初期化ハンドラ	可	可
タイムイベントハンドラ	可	可
割込みハンドラ	可	可

【解説】

tskid で指定される ID 番号のタスクが待ち状態にある場合に、強制的に実行可能状態に遷移させます。このシステムコールにより待ち解除されたタスクには、待ち状態に入ったシステムコールの戻り値として **E_RLWAI** エラーを返します。対象タスクが待ち状態でない場合には、**E_OBJ** エラーを返します。ただし、割込みハンドラからの呼び出しでは遅延実行するため、戻り値のエラーコードでは **E_OBJ** エラーを検出できません。

【推奨】

μ C3/Standard の **rel_wai** と **irel_wai** は、同じシステムコールとして実装するため、呼び出しコンテキストに関係なく、同じ使い方ができます。ただし、タスクコンテキストから呼び出す場合には **rel_wai** を、それ以外の場合には **irel_wai** を使うことをお勧めします。

sus_tsk**強制待ち状態への移行****【書式】**

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

【パラメータ】

ID	tskid	移行対象のタスクの ID 番号
----	-------	-----------------

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (tskid が不正あるいは使用できない)
E_CTX	コンテキストエラー (解説を参照)
E_OBJ	オブジェクト状態エラー (対象タスクが休止状態)
E_NOEXS	オブジェクト未生成 (対象タスクが未登録)
E_QOVR	キューイングオーバーフロー (強制待ち要求ネスト数のオーバーフロー)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

tskid で指定される ID 番号のタスクを強制待ち状態にして、タスクの実行を中断させます。具体的には、対象タスクが実行できる状態の時は強制待ち状態に、待ち状態の時は二重待ち状態に遷移します。同時に、対象タスクの強制待ち要求ネスト数に 1 を加算します。タスクの強制待ち要求ネスト数に 1 を加算すると強制待ち要求ネスト数の最大値を超える場合には、E_QOVR エラーを返します。

ディスパッチ禁止状態でも呼び出すことができますが、ディスパッチ禁止状態で自タスクを対象タスクとして呼び出された場合には、E_CTX エラーを返します。また、割込みハンドラからの呼び出しでも、E_CTX エラーを返します。

tskid に TSK_SELF (=0) が指定されると、自タスクを対象タスクとします。

rsm_tsk	強制待ち状態からの再開
frsm_tsk	強制待ち状態からの強制再開

【書式】

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

```
ER ercd = frsm_tsk (ID tskid);
```

【パラメータ】

ID	tskid	再開対象のタスクの ID 番号
----	-------	-----------------

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (tskid が不正あるいは使用できない)
E_CTX	コンテキストエラー (割込みハンドラからの呼び出し)
E_OBJ	オブジェクト状態エラー (対象タスクが強制待ち状態でない)
E_NOEXS	オブジェクト未生成 (対象タスクが未登録)

【呼び出しコンテキスト】

	rsm_tsk	frsm_tsk
タスク	可	可
初期化ハンドラ	可	可
タイムイベントハンドラ	可	可
割込みハンドラ	不可	不可

【解説】

tskid で指定される ID 番号のタスクの強制待ちを解除し、タスクの実行を再開させます。

rsm_tsk は、対象タスクの強制待ち要求ネスト数から 1 を減算し、減算後の強制待ち要求ネスト数が 0 の場合には、対象タスクが強制待ち状態の時は実行可能状態に、二重待ち状態の時は待ち状態に遷移させます。減算後の強制待ち要求ネスト数が 1 以上の場合には、対象タスクの状態は変化しません。

frsm_tsk は、対象タスクの強制待ち要求ネスト数を 0 にし、対象タスクが強制待ち状態の時は実行可能状態に、二重待ち状態の時は待ち状態に遷移させます。

対象タスクが未登録状態の場合には E_NOEXS エラーを、強制待ち状態でも二重待ち状態でもない場合には E_OBJ エラーを返します。

dly_tsk**自タスクの遅延****【書式】**

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

【パラメータ】

RELTIM	dlytim	自タスクの遅延時間（相対時間）
--------	--------	-----------------

【戻り値】

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（タスク以外、あるいはディスパッチ保留状態）
E_RLWAI	待ち状態の強制解除（待ち状態の間に rel_wai を受付）

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	不可
タイムイベントハンドラ	不可
割込みハンドラ	不可

【解説】

自タスクをシステムコールが呼び出された時刻から dlytim で指定される時間の間、時間経過待ち状態に遷移させます。指定された時間後に待ち解除された場合には、このシステムコールは正常終了し、E_OK を返します。

5. 3 タスク例外処理

タスク例外処理は、現バージョンでは対応していません。

5. 4 同期・通信機能

5. 4. 1 セマフォ

cre_sem	セマフォの生成
acre_sem	セマフォの生成 (ID 番号自動割付け)

【書式】

```
ER ercd = cre_sem(ID semid, T_CSEM*pk_csem);
```

```
ER_ID semid = acre_sem(T_CSEM*pk_csem);
```

【パラメータ】

ID	semid	生成対象のセマフォの ID 番号 (acre_sem 以外)
T_CSEM*	pk_csem	セマフォ生成情報を入れたパケットへのポインタ pk_csem の内容 (T_CSEM 型)
ATR	sematr	セマフォ属性
UINT	isemcnt	セマフォの資源数の初期値
UINT	maxsem	セマフォの最大資源数
VB const *	name	セマフォの名称 (文字列)

【戻り値】

cre_sem の場合

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

acre_sem の場合

ER_ID	semid	生成したセマフォの ID 番号 (正の値) または エラーコード
-------	-------	----------------------------------

【エラーコード】

E_RSATR	予約属性 (sematr が不正あるいは使用できない)
E_PAR	パラメータエラー (isemcnt, maxsem が不正)
E_ID	不正 ID 番号 (semid が不正あるいは使用できない ; cre_sem のみ)
E_CTX	コンテキストエラー (タスクと初期化ハンドラ以外からの呼び出し)
E_NOMEM	メモリ不足 (管理領域などが確保できない)
E_NOID	ID 番号不足 (割り付け可能なセマフォ ID がない ; acre_sem のみ)
E_OBJ	オブジェクト状態エラー (対象セマフォが登録済み ; cre_sem のみ)

【呼び出しコンテキスト】

	cre_sem	acre_sem
タスク	可	可
初期化ハンドラ	可	可
タイムイベントハンドラ	不可	不可
割込みハンドラ	不可	不可

【解説】

`semid` で指定される ID 番号を持つセマフォを、`pk_csem` で指定されるセマフォ生成情報に基づいて生成します。`sematr` はセマフォの属性、`isemcnt` はセマフォ生成後の資源数の初期値、`maxsem` はセマフォの最大資源数です。

`acre_sem` は、生成するセマフォの ID 番号をセマフォが登録されていない ID 番号の中から一番大きな値を割り付け、その ID 番号を戻り値として返します。

`sematr` には、(`TA_TFIFO` || `TA_TPRI`) の指定ができます。セマフォの待ち行列は、`TA_TFIFO` (=0x00) が指定された場合には FIFO 順に、`TA_TPRI` (=0x01) が指定された場合にはタスクの優先度順になります。

`isemcnt` に `maxsem` よりも大きい値が指定された場合には、`E_PAR` エラーを返します。また、`maxsem` に 0 が指定された場合や、セマフォの最大資源数の最大値 (`TMAX_MAXSEM`) よりも大きい値が指定された場合にも、`E_PAR` エラーを返します。

del_sem

セマフォの削除

【書式】

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

【パラメータ】

ID	semid	削除対象のセマフォの ID 番号
----	-------	------------------

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (semid が不正あるいは使用できない)
E_CTX	コンテキストエラー (タスク以外からの呼び出し)
E_NOEXS	オブジェクト未生成 (対象セマフォが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	不可
タイムイベントハンドラ	不可
割込みハンドラ	不可

【解説】

semid で指定される ID 番号のセマフォを削除します。

sig_sem	セマフォ資源の返却
isig_sem	

【書式】

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

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

【パラメータ】

ID	semid	資源返却対象のセマフォの ID 番号
----	-------	--------------------

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (semid が不正、あるいは使用できない)
E_NOMEM	メモリ不足 (SSB が不足：割込みハンドラからの呼び出しのみ)
E_NOEXS	オブジェクト未生成 (対象セマフォが未登録)
E_QOVR	キューイングオーバーフロー (最大資源数を超える返却)

【呼び出しコンテキスト】

	sig_sem	isig_sem
タスク	可	可
初期化ハンドラ	可	可
タイムイベントハンドラ	可	可
割込みハンドラ	可	可

【解説】

semid で指定される ID 番号のセマフォに対して資源の獲得を待っているタスクがある場合には、待ち行列の先頭のタスクを待ち解除し、実行可能状態に遷移させます。この時、対象セマフォの資源数は変化しません。また、待ち解除されたタスクには、待ち状態に入ったシステムコールの戻り値として E_OK を返します。

資源の獲得を待っているタスクがない場合には、対象セマフォの資源数に 1 を加算します。この時、セマフォの資源数に 1 を加算するとセマフォの最大資源数を超える場合には、E_QOVR エラーを返します。ただし、割込みハンドラからの呼び出しでは遅延実行するため、戻り値のエラーコードでは E_QOVR エラーを検出できません。

【推奨】

μ C3/Standard の sig_sem と isig_sem は、同じシステムコールとして実装するため、呼び出しコンテキストに関係なく、同じ使い方ができます。ただし、タスクコンテキストから呼び出す場合には sig_sem を、それ以外の場合には isig_sem を使うことをお勧めします。

wai_sem	セマフォ資源の獲得
pol_sem	セマフォ資源の獲得（ポーリング）
twai_sem	セマフォ資源の獲得（タイムアウトあり）

【書式】

```
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	資源獲得対象のセマフォの ID 番号
TMO	tmout	タイムアウト指定（twai_sem のみ）

【戻り値】

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

【エラーコード】

E_PAR	パラメータエラー（tmout が不正；twai_sem のみ）
E_ID	不正 ID 番号（semid が不正、あるいは使用できない）
E_CTX	コンテキストエラー（タスク以外からの呼び出し：pol_sem 以外、 割込みハンドラからの呼び出し：pol_sem のみ、 ディスパッチ保留状態：pol_sem 以外）
E_NOEXS	オブジェクト未生成（対象セマフォが未登録）
E_RLWAI	待ち状態の強制解除（待ち状態の間に rel_wai を受付；pol_sem 以外）
E_TMOUT	ポーリング失敗またはタイムアウト（wai_sem 以外）
E_DLT	待ちオブジェクトの削除 (待ち状態の間に対象セマフォが削除；pol_sem 以外)

【呼び出しコンテキスト】

	wai_sem	pol_sem	twai_sem
タスク	可	可	可
初期化ハンドラ	不可	可	不可
タイムイベントハンドラ	不可	可	不可
割込みハンドラ	不可	不可	不可

【解説】

semid で指定される ID 番号のセマフォから、資源を 1 つ獲得します。対象セマフォの資源数が 1 以上の場合には、セマフォの資源数から 1 を減算し、待ち状態にはならずシステムコールを終了します。対象セマフォの資源数が 0 の場合には、資源数は 0 のまま、自タスクを待ち行列につなぎ、セマフォ資源の獲得待ち状態に遷移させます。

pol_sem は wai_sem の処理をポーリングで行うシステムコール、twai_sem は wai_sem に

タイムアウトの機能を付け加えたシステムコールです。また、`tmout` に、`TMO_POL (=0)` や `TMO_FEVR (= -1)` を指定することもできます。 μ C3/Standard では、`tmout` に `TMO_FEVR` を指定した `twai_sem` は `wai_sem` として扱い、`tmout` に `TMO_POL` を指定した `twai_sem` は `pol_sem` として扱います。

ref_sem

セマフォの状態参照

【書式】

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

【パラメータ】

ID	semid	状態参照対象のセマフォの ID 番号
T_RSEM*	pk_rsem	セマフォ状態を返すパケットへのポインタ

【戻り値】

ER	ercd	正常終了 (E_OK) またはエラーコード
pk_rsem の内容 (T_RSEM 型)		
ID	wtskid	セマフォの待ち行列の先頭のタスクの ID 番号
UINT	semcnt	セマフォの現在の資源数

【エラーコード】

E_ID	不正 ID 番号 (semid が不正、あるいは使用できない)
E_CTX	コンテキストエラー (割込みハンドラからの呼び出し)
E_NOEXS	オブジェクト未生成 (対象セマフォが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

semid で指定される ID 番号のセマフォに関する状態を参照し、pk_rsem で指定されるパケットに返します。

wtskid には、対象セマフォの待ち行列の先頭のタスクの ID 番号を返します。資源の獲得を待っているタスクが無い場合には、TSK_NONE (=0) を返します。

semcnt には、対象セマフォの現在の資源数を返します。

5. 4. 2 イベントフラグ

cre_flg	イベントフラグの生成
acre_flg	イベントフラグの生成 (ID 番号自動割付け)

【書式】

```
ER ercd = cre_flg(ID flgid, T_CFLG*pk_cflg);
```

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

【パラメータ】

ID	flgid	生成対象のイベントフラグの ID 番号 (acre_flg 以外)
T_CFLG *	pk_cflg	イベントフラグ生成情報を入れたパケットへの ポインタ
pk_cflg の内容 (T_CFLG 型)		
ATR	flgatr	イベントフラグ属性
FLGPTN	iflgptn	イベントフラグのビットパターンの初期値
VB const *	name	イベントフラグの名称 (文字列)

【戻り値】

cre_flg の場合

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

acre_flg の場合

ER_ID	flgid	生成したイベントフラグの ID 番号 (正の値) または エラーコード
-------	-------	--

【エラーコード】

E_RSATR	予約属性 (flgatr が不正あるいは使用できない)
E_PAR	パラメータエラー (iflgptn が不正)
E_ID	不正 ID 番号 (flgid が不正あるいは使用できない ; cre_flg のみ)
E_CTX	コンテキストエラー (タスクと初期化ハンドラ以外からの呼び出し)
E_NOMEM	メモリ不足 (管理領域などが確保できない)
E_NOID	ID 番号不足 (割り付け可能なイベントフラグ ID がない ; acre_flg のみ)
E_OBJ	オブジェクト状態エラー (対象イベントフラグが登録済み ; cre_flg のみ)

【呼び出しコンテキスト】

	cre_flg	acre_flg
タスク	可	可
初期化ハンドラ	可	可
タイムイベントハンドラ	不可	不可
割込みハンドラ	不可	不可

【解説】

flgid で指定される ID 番号を持つイベントフラグを、pk_cflg で指定されるイベントフラグ生成情報に基づいて生成します。flgatr はイベントフラグの属性、iflgptn はイベントフラグ生成後のビットパターンの初期値です。

acre_flg は、生成するイベントフラグの ID 番号をイベントフラグが登録されていない ID 番号の中から一番大きな値を割り付け、その ID 番号を戻り値として返します。

flgatr には、((TA_TFIFO || TA_TPRI) | (TA_WSGL || TA_WMUL) | [TA_CLR]) の指定ができます。イベントフラグの待ち行列は、TA_TFIFO (=0x00) が指定された場合には FIFO 順に、TA_TPRI (=0x01) が指定された場合にはタスクの優先度順になります。

TA_WSGL (=0x00) が指定された場合には、一つのイベントフラグで同時に複数のタスクが待ち状態となることができません。逆に、TA_WMUL (=0x02) が指定された場合には、同時に複数のタスクが待ち状態にできます。

TA_CLR (=0x04) が指定された場合には、イベントフラグ待ちの解除条件が成り立った時に、タスクをイベントフラグ待ちから解除すると同時に、イベントフラグのビットパターンのすべてのビットをクリアします。

del_flg	イベントフラグの削除
----------------	-------------------

【書式】

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

【パラメータ】

ID	flgid	削除対象のイベントフラグの ID 番号
----	-------	---------------------

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (flgid が不正あるいは使用できない)
E_CTX	コンテキストエラー (タスク以外からの呼び出し)
E_NOEXS	オブジェクト未生成 (対象イベントフラグが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	不可
タイムイベントハンドラ	不可
割込みハンドラ	不可

【解説】

flgid で指定される ID 番号のイベントフラグを削除します。

set_flg	イベントフラグのセット
iset_flg	

【書式】

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

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

【パラメータ】

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

【戻り値】

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

【エラーコード】

E_PAR	パラメータエラー (setptn が不正)
E_ID	不正 ID 番号 (flgid が不正、あるいは使用できない)
E_NOMEM	メモリ不足 (SSB が不足: 割込みハンドラからの呼び出しのみ)
E_NOEXS	オブジェクト未生成 (対象イベントフラグが未登録)

【呼び出しコンテキスト】

	set_flg	iset_flg
タスク	可	可
初期化ハンドラ	可	可
タイムイベントハンドラ	可	可
割込みハンドラ	可	可

【解説】

flgid で指定される ID 番号のイベントフラグのビットパターンを、システムコール呼び出し前のビットパターンと setptn のビット毎の論理和 (OR) により更新します。イベントフラグのビットパターンが変化した場合は、イベントフラグの待ち行列の先頭のタスクから順に待ち解除条件を満たしているかを調べ、待ち解除条件を満たしているタスクが見つければ、そのタスクを待ち解除します。また、待ち解除されたタスクには、待ち状態に入ったシステムコールの戻り値として E_OK を返し、待ち解除時のビットパターンとして、この時のビットパターンを返します。この時、対象イベントフラグ属性に TA_CLR 属性が指定されている場合には、ビットパターンのすべてのビットをクリアし、システムコールの処理を終了します。TA_CLR 属性が指定されていない場合には、続けて待ち行列を最後尾まで調べます。

【推奨】

μ C3/Standard の set_flg と iset_flg は、同じシステムコールとして実装するため、呼び出しコンテキストに関係なく、同じ使い方ができます。ただし、タスクコンテキストから呼び出す場合には set_flg を、それ以外の場合には iset_flg を使うことをお勧めします。

clr_flg**イベントフラグのクリア****【書式】**

```
ER ercd = clr_flg(ID flgid, FLGPtn clrptn);
```

【パラメータ】

ID	flgid	セット対象のイベントフラグの ID 番号
FLGPtn	clrptn	クリアするビットパターン（ビット毎の反転値）

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号（flgid が不正、あるいは使用できない）
E_CTX	コンテキストエラー（タスク以外からの呼び出し）
E_NOEXS	オブジェクト未生成（対象イベントフラグが未登録）

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

flgid で指定される ID 番号のイベントフラグのビットパターンを、システムコール呼び出し前のビットパターンと clrptn のビット毎の論理積（AND）により更新します。

【使い方】

```
ER ercd;
```

```
/* クリアするビットのみ 0 にした値をパラメータにして呼び出す */
```

```
ercd = clr_flg(ID_Flag1, ~0x00000001); /* イベントフラグの bit 0 のみクリアする */
```

wai_flg	イベントフラグ待ち
pol_flg	イベントフラグ待ち（ポーリング）
twai_flg	イベントフラグ待ち（タイムアウトあり）

【書式】

```
ER ercd = wai_flg(ID flgid, FLGPTN waiptn, MODE wfmode,  
                  FLGPTN *p_flgptn);
```

```
ER ercd = pol_flg(ID flgid, FLGPTN waiptn, MODE wfmode,  
                  FLGPTN *p_flgptn);
```

```
ER ercd = twai_flg(ID flgid, FLGPTN waiptn, MODE wfmode,  
                   FLGPTN *p_flgptn, TMO tmout);
```

【パラメータ】

ID	flgid	待ち対象のイベントフラグの ID 番号
FLGPTN	waiptn	待ちビットパターン
MODE	wfmode	待ちモード
TMO	tmout	タイムアウト指定（twai_flg のみ）

【戻り値】

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_PAR	パラメータエラー（waiptn, wfmode が不正）
E_ID	不正 ID 番号（flgid が不正、あるいは使用できない）
E_CTX	コンテキストエラー（タスク以外からの呼び出し：pol_flg 以外、 割込みハンドラからの呼び出し：pol_flg のみ、 ディスパッチ保留状態：pol_flg 以外）
E_ILUSE	システムコール不正使用 （TA_WSGL 属性が指定されたイベントフラグで待ちタスクあり）
E_NOEXS	オブジェクト未生成（対象イベントフラグが未登録）
E_RLWAI	待ち状態の強制解除（待ち状態の間に rel_wai を受付；pol_flg 以外）
E_TMOUT	ポーリング失敗またはタイムアウト（wai_flg 以外）
E_DLT	待ちオブジェクトの削除 （待ち状態の間に対象イベントフラグが削除；pol_flg 以外）

【呼び出しコンテキスト】

	wai_flg	pol_flg	twai_flg
タスク	可	可	可
初期化ハンドラ	不可	可	不可
タイムイベントハンドラ	不可	可	不可
割込みハンドラ	不可	不可	不可

【解説】

flgid で指定される ID 番号のイベントフラグのビットパターンが、waipn と wfmode で指定される待ち解除条件を満たしていない場合には、条件が満たすまで自タスクを待ち行列につなぎ、イベントフラグ待ち状態に遷移させます。waipn と wfmode で指定される待ち解除条件を満たしている場合には、自タスクを待ち状態とせずにシステムコールの処理を終了し、flgptn に待ち解除条件を満たしたビットパターンを返します。この時、対象イベントフラグ属性に TA_CLR 属性が指定されている場合には、イベントフラグのビットパターンのすべてのビットをクリアします。

対象イベントフラグ属性に TA_WSG 属性が指定され、イベントフラグの待ち行列に他のタスクがつながれている場合には、待ち解除条件に関係なく E_ILUSE エラーとなります。

wfmode には、TWF_ANDW または TWF_ORW のいずれかが指定できます。wfmode に TWF_ANDW が指定された場合の待ち解除条件とは、対象イベントフラグのビットパターンと waipn で指定されるビットのすべてがセットされるという条件です。TWF_ORW が指定された場合の待ち解除条件とは、対象イベントフラグのビットパターンと waipn で指定されるビットのいずれかがセットされるという条件です。

pol_flg は wai_flg の処理をポーリングで行うシステムコール、twai_flg は wai_flg にタイムアウトの機能を付け加えたシステムコールです。また、tmout に、TMO_POL (=0) や TMO_FEVR (= -1) を指定することもできます。μ C3/Standard では、tmout に TMO_FEVR を指定した twai_flg は、wai_flg として扱い、tmout に TMO_POL を指定した twai_flg は、pol_flg として扱います。

【使い方】

```

FLGPTN waipn;          /* フラグパターンを格納する領域を確保 */
ER ercd;

                          /* 格納する領域へのポインタをパラメータにして呼び出す */
ercd = wai_flg(ID_Flag1, 0x00000003, TWF_ORF, &waipn);
if (ercd == E_OK) {
    ercd = clr_flg(ID_Flag1, ~waipn);
    if ((waipn & 0x00000001) != 0) {
        /* イベントフラグ bit 0 に対応する処理 */
    }
    if ((waipn & 0x00000002) != 0) {
        /* イベントフラグ bit 1 に対応する処理 */
    }
}

```

ref_flg イベントフラグの状態参照

【書式】

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

【パラメータ】

ID	flgid	状態参照対象のイベントフラグの ID 番号
T_RFLG*	pk_rflg	イベントフラグ状態を返すパケットへのポインタ

【戻り値】

ER	ercd	正常終了 (E_OK) またはエラーコード
pk_rflg の内容 (T_RFLG 型)		
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
FLGPTN	flgptn	イベントフラグの現在のビットパターン

【エラーコード】

E_ID	不正 ID 番号 (flgid が不正、あるいは使用できない)
E_CTX	コンテキストエラー (割込みハンドラからの呼び出し)
E_NOEXS	オブジェクト未生成 (対象イベントフラグが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

flgid で指定される ID 番号のイベントフラグに関する状態を参照し、pk_rflg で指定されるパケットに返します。

wtskid には、対象イベントフラグの待ち行列の先頭のタスクの ID 番号を返します。イベントを待っているタスクがない場合には、TSK_NONE (=0) を返します。

flgptn には、対象イベントフラグの現在のビットパターンを返します。

5. 4. 3 データキュー

cre_dtq	データキューの生成
acre_dtq	データキューの生成 (ID 番号自動割付け)

【書式】

```
ER ercd = cre_dtq(ID dtqid, T_CDTQ*pk_cdtq);
```

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

【パラメータ】

ID	dtqid	生成対象のデータキューの ID 番号 (acre_dtq 以外)
T_CDTQ*	pk_cdtq	データキュー生成情報を入れたパケットへのポインタ pk_cdtq の内容 (T_CDTQ 型)
ATR	dtqatr	データキュー属性
UINT	dtqcnt	データキュー領域の容量 (データの個数)
VP	dtq	データキュー領域の先頭番地
VB const *	name	データキューの名称 (文字列)

【戻り値】

cre_dtq の場合

ER ercd 正常終了 (E_OK) またはエラーコード

acre_dtq の場合

ER_ID dtqid 生成したデータキューの ID 番号 (正の値) またはエラーコード

【エラーコード】

E_RSATR	予約属性 (dtqatr が不正あるいは使用できない)
E_PAR	パラメータエラー(dtqcnt が不正)
E_ID	不正 ID 番号 (dtqid が不正あるいは使用できない ; cre_dtq のみ)
E_CTX	コンテキストエラー (タスクと初期化ハンドラ以外からの呼び出し)
E_NOMEM	メモリ不足 (管理領域やデータキュー領域が確保できない)
E_NOID	ID 番号不足 (割り付け可能なデータキューID がない; acre_dtq のみ)
E_OBJ	オブジェクト状態エラー

(対象データキューが登録済み ; cre_dtq のみ)

【呼び出しコンテキスト】

	cre_dtq	acre_dtq
タスク	可	可
初期化ハンドラ	可	可
タイムイベントハンドラ	不可	不可
割込みハンドラ	不可	不可

【解説】

dtqid で指定される ID 番号を持つデータキューを、pk_cdtq で指定されるデータキュー生成情報に基づいて生成します。dtqatr はデータキューの属性、dtqcnt はデータキュー領域に格納できるデータの個数、dtq はデータキュー領域の先頭番地です。

acre_dtq は、生成するデータキューの ID 番号をデータキューが登録されていない ID 番号の中から一番大きな値を割り付け、その ID 番号を戻り値として返します。

dtqatr には、(TA_TFIFO || TA_TPRI) の指定ができます。データキューの送信待ち行列は、TA_TFIFO (0x00) が指定された場合には FIFO 順に、TA_TPRI (0x01) が指定された場合にはタスクの優先度順になります。

dtq で指定された番地から、dtqcnt 個のデータを格納するのに必要なサイズのメモリ領域を、データキュー領域として使用します。dtq に NULL (=0) が指定された場合には、必要なサイズのメモリ領域を、コンフィグレーションで定義したメモリプール用メモリ領域から自動で確保します

同期メッセージ機能を使用する場合には、dtqcnt に 0 を指定します。

del_dtq	データキューの削除
----------------	------------------

【書式】

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

【パラメータ】

ID	dtqid	削除対象のデータキューの ID 番号
----	-------	--------------------

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (dtqid が不正あるいは使用できない)
E_CTX	コンテキストエラー (タスク以外からの呼び出し)
E_NOEXS	オブジェクト未生成 (対象データキューが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	不可
タイムイベントハンドラ	不可
割込みハンドラ	不可

【解説】

dtqid で指定される ID 番号のデータキューを削除します。

snd_dtq	データキューへの送信
psnd_dtq	データキューへの送信（ポーリング）
ipsnd_dtq	
tsnd_dtq	データキューへの送信（タイムアウトあり）

【書式】

```

ER ercd = snd_dtq(ID dtqid, VP_INT data);
ER ercd = psnd_dtq(ID dtqid, VP_INT data);
ER ercd = ipsnd_dtq(ID dtqid, VP_INT data);
ER ercd = tsnd_dtq(ID dtqid, VP_INT data, TMO tmout)

```

【パラメータ】

ID	dtqid	送信対象のデータキューの ID 番号
VP_INT	data	データキューへ送信するデータ
TMO	tmout	タイムアウト指定（tsnd_dtq のみ）

【戻り値】

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

【エラーコード】

E_ID 不正 ID 番号（dtqid が不正、あるいは使用できない）
E_CTX コンテキストエラー（タスク以外からの呼び出し：(i)psnd_dtq 以外、
割込みハンドラからの呼び出し：(i)psnd_dtq のみ、
ディスパッチ保留状態：(i)psnd_dtq 以外）
E_NOMEM メモリ不足（SSB が不足：割込みハンドラからの呼び出しのみ）
E_NOEXS オブジェクト未生成（対象データキューが未登録）
E_RLWAI 待ち状態の強制解除
（待ち状態の間に rel_wai を受付；snd_dtq, tsnd_dtq のみ）
E_TMOUT ポーリング失敗またはタイムアウト（snd_dtq 以外）
E_DLT 待ちオブジェクトの削除
（待ち状態の間に対象データキューが削除；(i)psnd_dtq 以外）

【呼び出しコンテキスト】

	snd_dtq	psnd_dtq	ipsnd_dtq	tsnd_dtq
タスク	可	可	可	可
初期化ハンドラ	不可	可	可	不可
タイムイベントハンドラ	不可	可	可	不可
割込みハンドラ	不可	可	可	不可

【解説】

dtqid で指定される ID 番号のデータキューに受信を待っているタスクがある場合には、受信

待ち行列の先頭のタスクに送信するデータを渡し、そのタスクを待ち解除します。また、待ち解除されたタスクには、待ち状態に入ったシステムコールの戻り値として **E_OK** を返し、データキューから受信したデータとして **data** の値を返します。受信を待っているタスクがない場合は、送信するデータをデータキューの末尾に入れます。データキュー領域に空きがない場合には、自タスクを送信待ち行列につなぎ、データキューへの送信待ち状態に遷移させます。

psnd_dtq と **ipsnd_dtq** は、対象データキューで受信を待っているタスクがなく、データキュー領域に空きがない場合には、**E_TMOUT** エラーを戻り値として返します。ただし、割込みハンドラからの呼び出しでは遅延実行するため、戻り値のエラーコードでは **E_TMOUT** エラーを検出できません。

psnd_dtq と **ipsnd_dtq** は **snd_dtq** の処理をポーリングで行うシステムコール、**tsnd_dtq** は **snd_dtq** にタイムアウトの機能を付け加えたシステムコールです。また、**tmout** に、**TMO_POL** (=0) や **TMO_FEVR** (=−1) を指定することもできます。 μ C3/Standard では、**tmout** に **TMO_FEVR** を指定した **tsnd_dtq** は **snd_dtq** として扱い、**tmout** に **TMO_POL** を指定した **tsnd_dtq** は **psnd_dtq** として扱います。

【推奨】

μ C3/Standard の **psnd_dtq** と **ipsnd_dtq** は、同じシステムコールとして実装するため、呼び出しコンテキストに関係なく、同じ使い方ができます。ただし、タスクコンテキストから呼び出す場合には **psnd_dtq** を、それ以外の場合には **ipsnd_dtq** を使うことをお勧めします。

fsnd_dtq データキューへの強制送信

ifsnd_dtq

【書式】

ER ercd = fsnd_dtq(ID dtqid, VP_INT data);

ER ercd = ifsnd_dtq(ID dtqid, VP_INT data);

【パラメータ】

ID	dtqid	送信対象のデータキューの ID 番号
VP_INT	data	データキューへ送信するデータ

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (dtqid が不正、あるいは使用できない)
E_ILUSE	システムコール不正使用 (対象データキューのデータキュー領域の容量が 0)
E_NOMEM	メモリ不足 (SSB が不足 : 割込みハンドラからの呼び出しのみ)
E_NOEXS	オブジェクト未生成 (対象データキューが未登録)

【呼び出しコンテキスト】

	fsnd_dtq	ifsnd_dtq
タスク	可	可
初期化ハンドラ	可	可
タイムイベントハンドラ	可	可
割込みハンドラ	可	可

【解説】

dtqid で指定される ID 番号のデータキューで受信を待っているタスクがある場合には、受信待ち行列の先頭のタスクに送信するデータを渡し、そのタスクを待ち解除します。また、待ち解除されたタスクには、待ち状態に入ったシステムコールの戻り値として E_OK を返し、データキューから受信したデータとして data の値を返します。

受信を待っているタスクがない場合は、送信するデータをデータキューの末尾に入れます。ここで、データキュー領域に空きがない場合には、データキューの先頭のデータを削除し、データキュー領域に必要な領域を確保し、送信するデータをデータキューの末尾に入れます。つまり、一番古いデータを削除します。

データキュー領域の容量が 0 のデータキューに対してデータの強制送信を試みた場合には、E_ILUSE エラーを戻り値として返します。

【推奨】

μ C3/Standard の `fsnd_dtq` と `ifsnd_dtq` は、同じシステムコールとして実装するため、呼び出しコンテキストに関係なく、同じ使い方ができます。ただし、タスクコンテキストから呼び出す場合には `fsnd_dtq` を、それ以外の場合には `ifsnd_dtq` を使うことをお勧めします。

rcv_dtq	データキューからの受信
prcv_dtq	データキューからの受信（ポーリング）
trcv_dtq	データキューからの受信（タイムアウトあり）

【書式】

```
ER ercd = rcv_dtq(ID dtqid, VP_INT *p_data);
ER ercd = prcv_dtq(ID dtqid, VP_INT *p_data);
ER ercd = trcv_dtq(ID dtqid, VP_INT *p_data, TMO tmout);
```

【パラメータ】

ID	dtqid	受信対象のデータキューの ID 番号
TMO	tmout	タイムアウト指定（trcv_dtq のみ）

【戻り値】

ER	ercd	正常終了（E_OK）またはエラーコード
VP_INT	data	データキューから受信したデータ

【エラーコード】

E_ID	不正 ID 番号（dtqid が不正あるいは使用できない）
E_CTX	コンテキストエラー（タスク以外からの呼び出し：prcv_dtq 以外、 割り込みハンドラからの呼び出し：prcv_dtq のみ、 ディスパッチ保留状態：prcv_dtq 以外）
E_NOEXS	オブジェクト未生成（対象データキューが未登録）
E_RLWAI	待ち状態の強制解除 (待ち状態の間に rel_wai を受付；prcv_dtq 以外)
E_TMOUT	ポーリング失敗またはタイムアウト（rcv_dtq 以外）
E_DLT	待ちオブジェクトの削除 (待ち状態の間に対象データキューが削除；prcv_dtq 以外)

【呼び出しコンテキスト】

	rcv_dtq	prcv_dtq	trcv_dtq
タスク	可	可	可
初期化ハンドラ	不可	可	不可
タイムイベントハンドラ	不可	可	不可
割り込みハンドラ	不可	不可	不可

【解説】

dtqid で指定される ID 番号のデータキューにデータが入っている場合には、その先頭のデータを取り出し、data に返します。この時、データキューで送信を待っているタスクがある場合には、送信待ち行列の先頭のタスクが送信しようとしているデータをデータキューの末尾に入れ、そのタスクを待ち解除します。また、待ち解除されたタスクには、待ち状態に入ったシス

テムコールの戻り値として **E_OK** を返します。

データが入っていない状態で、対象データキューで送信を待っているタスクがある場合には、送信待ち行列の先頭のタスクから、そのタスクが送信しようとしているデータを受け取り、そのタスクを待ち解除します。また、待ち解除されたタスクには、待ち状態に入ったシステムコールの戻り値として **E_OK** を返します。**data** には、受け取ったデータを返します。

データが入っていない場合で、送信を待っているタスクもない場合には、自タスクを受信待ち行列につなぎ、データキューからの受信待ち状態に遷移させます。

prcv_dtq は **rcv_dtq** の処理をポーリングで行うシステムコール、**trcv_dtq** は **rcv_dtq** にタイムアウトの機能を付け加えたシステムコールです。また、**tmout** に、**TMO_POL** (=0) や **TMO_FEVR** (=−1) を指定することもできます。 μ C3/Standard では、**tmout** に **TMO_FEVR** を指定した **trcv_dtq** は **rcv_dtq** として扱い、**tmout** に **TMO_POL** を指定した **trcv_dtq** は **prcv_dtq** として扱います。

ref_dtq**データキューの状態参照****【書式】**

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

【パラメータ】

ID	dtqid	状態参照対象のデータキューの ID 番号
T_RDTQ*	pk_rdtq	データキュー状態を返すパケットへのポインタ

【戻り値】

ER	ercd	正常終了 (E_OK) またはエラーコード
pk_rdtq の内容 (T_RDTQ 型)		
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	sdtqcnt	データキューに入っているデータの数

【エラーコード】

E_ID	不正 ID 番号 (dtqid が不正、あるいは使用できない)
E_CTX	コンテキストエラー (割込みハンドラからの呼び出し)
E_NOEXS	オブジェクト未生成 (対象データキューが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

dtqid で指定される ID 番号のデータキューに関する状態を参照し、pk_rdtq で指定されるパケットに返します。

stskid には、対象データキューの送信待ち行列の先頭のタスクの ID 番号を返します。送信を待っているタスクがない場合には、TSK_NONE (=0) を返します。

rtskid には、対象データキューの受信待ち行列の先頭のタスクの ID 番号を返します。受信を待っているタスクがない場合には、TSK_NONE (=0) を返します。

sdtqcnt には、対象データキューに現在入っているデータの個数を返します。

5. 4. 4 メールボックス

cre_mbx	メールボックスの生成
acre_mbx	メールボックスの生成 (ID 番号自動割付け)

【書式】

```
ER ercd = cre_mbx(ID mbxid, T_CMBX *pk_cmbx);
```

```
ER_ID mbxid = acre_mbx(T_CMBX *pk_cmbx);
```

【パラメータ】

ID	mbxid	生成対象のメールボックスの ID 番号 (acre_mbx 以外)
T_CMBX*	pk_cmbx	メールボックス生成情報を入れたパケットへの ポインタ
pk_cmbx の内容 (T_CMBX 型)		
ATR	mbxatr	メールボックス属性
PRI	maxmpri	送信されるメッセージの優先度の最大値
VP	mprihd	優先度別のメッセージキューヘッダ領域の先頭番地
VB const *	name	メールボックスの名称 (文字列)

【戻り値】

cre_mbx の場合

```
ER ercd
```

正常終了 (E_OK) またはエラーコード

acre_mbx の場合

```
ER_ID mbxid
```

生成したメールボックスの ID 番号 (正の値) または
エラーコード

【エラーコード】

E_RSATR	予約属性 (mbxatr が不正あるいは使用できない)
E_PAR	パラメータエラー (maxmpri が不正)
E_ID	不正 ID 番号 (mbxid が不正あるいは使用できない ; cre_mbx のみ)
E_CTX	コンテキストエラー (タスクと初期化ハンドラ以外からの呼び出し)
E_NOMEM	メモリ不足 (管理領域が確保できない)
E_NOID	ID 番号不足 (割り付け可能なメールボックス ID がない ; acre_mbx のみ)

【呼び出しコンテキスト】

	cre_mbx	acre_mbx
タスク	可	可
初期化ハンドラ	可	可
タイムイベントハンドラ	不可	不可
割込みハンドラ	不可	不可

【解説】

mbxid で指定される ID 番号を持つメールボックスを、pk_cmbx で指定されるメールボックス生成情報に基づいて生成します。mbxatr はメールボックスの属性、maxmpri はメールボックスに送信されるメッセージの優先度の最大値、mprihd はメールボックスの優先度別のメッセージキューヘッダ領域の先頭番地です。maxmpri と mprihd は、mbxatr に TA_MRI (=0x02) が指定された場合にのみ有効です。

acre_mbx は、生成するメールボックスの ID 番号をメールボックスが登録されていない ID 番号の中から一番大きな値を割り付け、その番号を戻り値として返します。

mbxatr には、((TA_TFIFO || TA_TPRI) | (TA_MFIFO || TA_MPRI)) の指定ができます。メールボックスの待ち行列は、TA_TFIFO (=0x00) が指定された場合には FIFO 順に、TA_TPRI (=0x01) が指定された場合にはタスクの優先度順になります。また、メールボックスのメッセージキューは、TA_MFIFO (=0x00) が指定された場合には FIFO 順、TA_MPRI (=0x02) が指定された場合にはメッセージの優先度順になります。

mbxatr に TA_MPRI が指定された場合、mprihd で指定された番地から、送信されるメッセージの優先度の最大値が maxmpri の場合に必要なサイズのメモリ領域を、優先度別のメッセージキューヘッダ領域として使用します。mprihd に NULL (=0) が指定された場合には、必要なサイズのメモリ領域をコンフィグレーションで定義したシステム用メモリ領域から自動で確保します。また、maxmpri に 0 が指定された場合や、メッセージ優先度の最大値 (TMAX_MPRI) よりも大きい値が指定された場合には、E_PAR エラーを戻り値として返します。

del_mbx	メールボックスの削除
----------------	-------------------

【書式】

```
ER ercd = del_mbx(ID mbxid);
```

【パラメータ】

ID	mbxid	削除対象のメールボックスの ID 番号
----	-------	---------------------

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (mbxid が不正あるいは使用できない)
E_CTX	コンテキストエラー (タスク以外からの呼び出し)
E_NOEXS	オブジェクト未生成 (対象メールボックスが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	不可
タイムイベントハンドラ	不可
割込みハンドラ	不可

【解説】

mbxid で指定される ID 番号のメールボックスを削除します。

snd_mbx**メールボックスへの送信****【書式】**

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

【パラメータ】

ID	mbxid	送信対象のメールボックスの ID 番号
T_MSG *	pk_msg	メールボックスへ送信するメッセージパケットの 先頭番地

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (mbxid が不正、あるいは使用できない)
E_CTX	コンテキストエラー (割込みハンドラからの呼び出し)
E_NOEXS	オブジェクト未生成 (対象メールボックスが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

mbxid で指定される ID 番号のメールボックスで受信を待っているタスクがある場合には、待ち行列の先頭のタスクに pk_msg で指定されたメッセージパケットの先頭番地を渡し、そのタスクを待ち解除します。また、待ち解除されたタスクには、待ち状態に入ったシステムコールの戻り値として E_OK を返し、メールボックスから受信したメッセージパケットの先頭番地として pk_msg の値を返します。

受信を待っているタスクがない場合には、pk_msg を先頭番地とするメッセージパケットをメッセージキューにつなぎます。メッセージキューが FIFO 順の場合には末尾につなぎ、メッセージ優先度順の場合にはメッセージパケットのメッセージ優先度順につなぎます。この時、同じ優先度のメッセージがあった場合には、同じ優先度のメッセージの最後尾につなぎます。

【使い方】

FIFO 順のメッセージを送信する場合

```
T_MSGPKT* pk_msgpkt; /* メッセージパケットの先頭番地を格納する領域を確保 */
ER ercd;
ercd = get_mpf(ID_Mpf1, &pk_msgpkt);
if (ercd == E_OK) {
    /* メッセージパケットを編集する */
    /* 格納する領域へのポインタをパラメータにして呼び出す */
    ercd = snd_mbx(ID_Mbx1, (T_MSG*)pk_msgpkt);
}
```

メッセージ優先度順のメッセージを送信する場合

```
T_MSGPRIPKT* pk_msgpripkt;
ER ercd;
ercd = get_mpf(ID_Mpf1, &pk_msgpripkt);
if (ercd == E_OK) {
    /* メッセージパケットを編集する */
    /* メッセージ優先度を 1 にする */
    pk_msgpripkt->pk_msg.msgpri = 1;
    /* 格納する領域へのポインタをパラメータにして呼び出す */
    ercd = snd_mbx(ID_Mbx1, (T_MSG*)pk_msgpkt);
}
```

rcv_mbx	メールボックスからの受信
prcv_mbx	メールボックスからの受信（ポーリング）
trcv_mbx	メールボックスからの受信（タイムアウトあり）

【書式】

```
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, TMO tmout);
```

【パラメータ】

ID	mbxid	受信対象のメールボックスの ID 番号
TMO	tmout	タイムアウト指定（trcv_mbx のみ）

【戻り値】

ER	ercd	正常終了（E_OK）またはエラーコード
T_MSG*	pk_msg	メールボックスから受信したメッセージパケットの 先頭番地

【エラーコード】

E_ID	不正 ID 番号（mbxid が不正、あるいは使用できない）
E_CTX	コンテキストエラー（タスク以外からの呼び出し：prcv_mbx 以外、 割込みハンドラからの呼び出し：prcv_mbx のみ、 ディスパッチ保留状態：prcv_mbx 以外）
E_NOEXS	オブジェクト未生成（対象メールボックスが未登録）
E_RLWAI	待ち状態の強制解除（待ち状態の間に rel_wai を受付；prcv_mbx 以外）
E_TMOUT	ポーリング失敗またはタイムアウト（rcv_mbx 以外）
E_DLT	待ちオブジェクトの削除 (待ち状態の間に対象メールボックスが削除；prcv_mbx 以外)

【呼び出しコンテキスト】

	rcv_mbx	prcv_mbx	trcv_mbx
タスク	可	可	可
初期化ハンドラ	不可	可	不可
タイムイベントハンドラ	不可	可	不可
割込みハンドラ	不可	不可	不可

【解説】

mbxid で指定される ID 番号のメールボックスのメッセージキューにメッセージが入っている場合には、その先頭のメッセージパケットを取り出し、その先頭番地を **pk_msg** に返します。メッセージが入っていない場合には、自タスクを待ち行列につなぎ、メールボックスからの受信待ち状態に遷移させます。

prcv_mbx は rcv_mbx の処理をポーリングで行うシステムコール、trcv_mbx は rcv_mbx にタイムアウトの機能を付け加えたシステムコールです。また、tmout に、TMO_POL (=0) や TMO_FEVR (= -1) を指定することもできます。 μ C3/Standard では、tmout に TMO_FEVR を指定した trcv_mbx は rcv_mbx として扱い、tmout に TMO_POL を指定した trcv_mbx は prcv_mbx として扱います。

【使い方】

```
T_MSGPKT* pk_msgpkt; /* メッセージパケットの先頭番地を格納する領域を確保 */
ER ercd;
```

```
ercd = rcv_mbx(ID_Mbx1, (T_MSG **)&pk_msgpkt);
if (ercd == E_OK) {
    /* メッセージパケットを処理する */
    /* 固定長メモリプールを使った場合は、メモリブロックを返却する */
    ercd = rel_mpf(ID_mpf1, pk_msgpkt);
}
```

ref_mbx**メールボックスの状態参照****【書式】**

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

【パラメータ】

ID	mbxid	状態参照対象のメールボックスの ID 番号
T_RMBX *	pk_rmbx	メールボックス状態を返すパケットへのポインタ

【戻り値】

ER	ercd	正常終了 (E_OK) またはエラーコード
pk_rmbx の内容 (T_RMBX 型)		
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のメッセージパケットの 先頭番地

【エラーコード】

E_ID	不正 ID 番号 (mbxid が不正、あるいは使用できない)
E_CTX	コンテキストエラー (割込みハンドラからの呼び出し)
E_NOEXS	オブジェクト未生成 (対象メールボックスが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

mbxid で指定される ID 番号のメールボックスに関する状態を参照し、pk_rmbx で指定されるパケットに返します。

wtskid には、対象メールボックスの待ち行列の先頭のタスクの ID 番号を返します。受信を待っているタスクがない場合には、TSK_NONE (=0) を返します。

pk_msg には、対象メールボックスのメッセージキューの先頭のメッセージパケットの先頭番地を返します。メッセージキューにメッセージが入っていない場合には、NULL (=0) を返します。

5. 4. 5 ミューテックス

cre_mtx	ミューテックスの生成
acre_mtx	ミューテックスの生成 (ID 番号自動割付け)

【書式】

```
ER ercd = cre_mtx(ID mtxid, T_CMTX*pk_cmtx);
```

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

【パラメータ】

ID	mtxid	生成対象のミューテックスの ID 番号 (acre_mtx 以外)
T_CMTX*	pk_cmtx	ミューテックス生成情報を入れたパケットへのポインタ pk_cmtx の内容 (T_CMTX 型)
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
VB const *	name	ミューテックスの名称 (文字列)

【戻り値】

cre_mtx の場合		
ER	ercd	正常終了 (E_OK) またはエラーコード
acre_mtx の場合		
ER_ID	mtxid	生成したミューテックスの ID 番号 (正の値) または エラーコード

【エラーコード】

E_RSATR	予約属性 (mtxatr が不正あるいは使用できない)
E_PAR	パラメータエラー (ceilpri が不正)
E_ID	不正 ID 番号 (mtxid が不正あるいは使用できない ; cre_mtx のみ)
E_CTX	コンテキストエラー (タスクと初期化ハンドラ以外からの呼び出し)
E_NOMEM	メモリ不足 (管理領域が確保できない)
E_NOID	ID 番号不足 (割り付け可能なミューテックス ID がない ; acre_mtx のみ)
E_OBJ	オブジェクト状態エラー (対象ミューテックスが登録済み ; cre_mtx のみ)

【呼び出しコンテキスト】

	cre_mtx	acre_mtx
タスク	可	可
初期化ハンドラ	可	可
タイムイベントハンドラ	不可	不可
割込みハンドラ	不可	不可

【解説】

mtxid で指定される ID 番号をもつミューテックスを、pk_cmtx で指定されるミューテックス生成情報に基づいて生成します。mtxatr はミューテックスの属性、ceilpri はミューテックスの上限優先度です。ceilpri は、mtxatr に TA_CEILING (=0x03) が指定された場合にのみ有効です。

acre_mtx は、生成するミューテックスの ID 番号をミューテックスが登録されていない ID 番号の中から一番大きな値を割り付け、その ID 番号を戻り値として返します。

mtxatr には、(TA_TFIFO || TA_TPRI || TA_INHERIT || TA_CEILING) の指定ができます。ミューテックスの待ち行列は、TA_TFIFO (=0x00) が指定された場合には FIFO 順に、その他が指定された場合にはタスクの優先度順になります。また、TA_INHERIT (=0x02) が指定された場合には優先度継承プロトコル、TA_CEILING (=0x03) が指定された場合には優先度上限プロトコルにしたがって、タスクの現在優先度を制御します。

del_mtx **ミューテックスの削除**

【書式】

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

【パラメータ】

ID	mtxid	削除対象のミューテックスの ID 番号
----	-------	---------------------

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (mtxid が不正あるいは使用できない)
E_CTX	コンテキストエラー (タスク以外からの呼び出し)
E_NOEXS	オブジェクト未生成 (対象ミューテックスが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	不可
タイムイベントハンドラ	不可
割込みハンドラ	不可

【解説】

mtxid で指定される ID 番号のミューテックスを削除します。

loc_mtx	ミューテックスのロック
ploc_mtx	ミューテックスのロック（ポーリング）
tloc_mtx	ミューテックスのロック（タイムアウトあり）

【書式】

```
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	ロック対象のミューテックスの ID 番号
TMO	tmout	タイムアウト指定（tloc_mtx のみ）

【戻り値】

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

【エラーコード】

E_ID 不正 ID 番号（mtxid が不正あるいは使用できない）

E_CTX コンテキストエラー（タスクと初期化ハンドラ以外からの呼び出し
ディスパッチ保留状態：ロック済みかつ、ploc_mtx 以外）

E_ILUSE サービスコール不正使用
(ミューテックスの多重ロック、上限優先度の違反)

E_NOEXS オブジェクト未生成（対象ミューテックスが未登録）

E_RLWAI 待ち状態の強制解除（待ち状態の間に rel_wai を受付；ploc_mtx 以外）

E_TMOUT ポーリング失敗またはタイムアウト（loc_mtx 以外）

E_DLT 待ちオブジェクトの削除
(待ち状態の間に対象ミューテックスが削除；ploc_mtx 以外)

【呼び出しコンテキスト】

	loc_mtx	ploc_mtx	tloc_mtx
タスク	可	可	可
初期化ハンドラ	不可	不可	不可
タイムイベントハンドラ	不可	不可	不可
割込みハンドラ	不可	不可	不可

【解説】

mtxid で指定される ID 番号のミューテックスをロックします。つまり、対象ミューテックスがロックされていない場合には、自タスクがミューテックスをロックした状態にします。対象ミューテックスがロックされている場合には、自タスクを待ち行列につなぎ、ミューテックスのロック待ち状態に遷移させます。

対象ミューテックスをロックした結果、自タスクの現在優先度が μ C3/Standard の優先度制御規則に合致していない場合には、適合するように自タスク以外も含め現在優先度を変更しま

す。

自タスクがすでに対象ミューテックスをロックしている場合には、**E_ILUSE** エラーを戻り値として返します。また、対象ミューテックスが **TA_CEILING** 属性の場合で、自タスクのベース優先度が対象ミューテックスの上限優先度よりも高い場合にも、**E_ILUSE** エラーを戻り値として返します。

ploc_mtx は、**loc_mtx** の処理をポーリングで行うシステムコール、**tloc_mtx** は、**loc_mtx** にタイムアウトの機能を付け加えたシステムコールです。**tmout** には、正の値のタイムアウト時間に加えて、**TMO_POL** (=0) と **TMO_FEVR** (=−1) を指定することができます。

μ C3/Standard では、**tmout** に **TMO_FEVR** を指定した **tloc_mtx** は **loc_mtx** として扱い、**tmout** に **TMO_POL** を指定した **tloc_mtx** は **ploc_mtx** として扱います。

unl_mtx**ミューテックスのロック解除****【書式】**

```
ER ercd = unl_mtx(ID mtxid);
```

【パラメータ】

ID	mtxid	ロック解除対象のミューテックスの ID 番号
----	-------	------------------------

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (mtxid が不正あるいは使用できない)
E_CTX	コンテキストエラー (タスク以外からの呼び出し)
E_ILUSE	サービスコール不正使用 (対象ミューテックスをロックしていない)
E_NOEXS	オブジェクト未生成 (対象ミューテックスが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	不可
タイムイベントハンドラ	不可
割込みハンドラ	不可

【解説】

mtxid で指定される ID 番号のミューテックスをロック解除します。つまり、対象ミューテックスに対してロックを待っているタスクがある場合には、待ち行列の先頭のタスクを待ち解除し、待ち解除されたタスクが対象ミューテックスをロックした状態にします。この時、待ち解除されたタスクに対しては、待ち状態に入ったシステムコールの戻り値として E_OK を返します。ロックを待っているタスクがない場合には、対象ミューテックスをロックされていない状態にします。

対象ミューテックスをロック解除した結果、自タスクの現在優先度が μ C3/Standard の優先度制御規則に合致していない場合には、適合するように自タスク以外も含め現在優先度を変更します。

対象ミューテックスを自タスクがロックしていない場合には、E_ILUSE エラーを戻り値として返します。

ref_mtx**ミューテックスの状態参照****【書式】**

```
ER ercd = ref_mtx(ID mtxid, T_RMTX*pk_rmtx);
```

【パラメータ】

ID	mtxid	状態参照対象のミューテックスの ID 番号
T_RMTX*	pk_rmtx	ミューテックス状態を返すパケットへのポインタ

【戻り値】

ER	ercd	正常終了 (E_OK) またはエラーコード
pk_rmtx の内容 (T_RMTX 型)		
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_ID	不正 ID 番号 (mtxid が不正あるいは使用できない)
E_CTX	コンテキストエラー (割込みハンドラからの呼び出し)
E_NOEXS	オブジェクト未生成 (対象ミューテックスが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

mtxid で指定される ID 番号のミューテックスに関する状態を参照し、pk_rmtx で指定されるパケットに返します。

htskid には、対象ミューテックスをロックしているタスクの ID 番号を返します。対象ミューテックスがロックされていない場合には、TSK_NONE (=0) を返します。

wtskid には、対象ミューテックスの待ち行列の先頭のタスクの ID 番号を返します。ロックを待っているタスクがない場合には、TSK_NONE (=0) を返します。

5. 4. 6 メッセージバッファ

cre_mbf	メッセージバッファの生成
acre_mbf	メッセージバッファの生成 (ID 番号自動割付け)

【書式】

ER ercd = cre_mbf(ID mbfid, T_CMBF * pk_cmbf);

ER_ID mbfid = acre_mbf(T_CMBF * pk_cmbf);

【パラメータ】

ID	mbfid	生成対象のメッセージバッファの ID 番号 (acre_mbf 以外)
T_CMBF*	pk_cmbf	メッセージバッファ生成情報を入れたパケットへのポインタ
pk_cmbf の内容 (T_CMBF 型)		
ATR	mbfatr	メッセージバッファ属性
UINT	maxmsz	メッセージの最大サイズ (バイト数)
SIZE	mbfsz	メッセージバッファ領域のサイズ (バイト数)
VP	mbf	メッセージバッファ領域の先頭番地
VB const *	name	メッセージバッファの名称 (文字列)

【戻り値】

cre_mbf の場合

ER ercd 正常終了 (E_OK) またはエラーコード

acre_mbf の場合

ER_ID mbfid 生成し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_RSATR	予約属性 (mbfatr が不正あるいは使用できない)
E_PAR	パラメータエラー (maxmsz, mbfsz が不正)
E_ID	不正 ID 番号 (mbfid が不正あるいは使用できない; cre_mbf のみ)
E_CTX	コンテキストエラー (タスクと初期化ハンドラ以外からの呼び出し)
E_NOMEM	メモリ不足 (管理領域やメッセージバッファ領域が確保できない)
E_NOID	ID 番号不足 (割り付け可能なメッセージバッファ ID がない; acre_mbf のみ)
E_OBJ	オブジェクト状態エラー (対象メッセージバッファが登録済み; cre_mbf のみ)

【呼び出しコンテキスト】	cre_mbf	acre_mbf
タスク	可	可
初期化ハンドラ	可	可
タイムイベントハンドラ	不可	不可
割込みハンドラ	不可	不可

【解説】

mbfid で指定される ID 番号を持つメッセージバッファを、pk_cmbf で指定されるメッセージバッファ生成情報に基づいて生成します。mbfatr はメッセージバッファの属性、maxmsz はメッセージバッファに送信できるメッセージの最大サイズ（バイト数）、mbfsz はメッセージバッファ領域のサイズ（バイト数）、mbf はメッセージバッファ領域の先頭番地です。

acre_mbf は、生成するメッセージバッファの ID 番号をメッセージバッファが登録されていない ID 番号の中から一番大きな値を割り付け、その ID 番号を戻り値として返します。

mbfatr には、(TA_TFIFO || TA_TPRI) の指定ができます。メッセージバッファの送信待ち行列は、TA_TFIFO (=0x00) が指定された場合には FIFO 順に、TA_TPRI (=0x01) が指定された場合にはタスクの優先度順になります。

mbf で指定された番地から mbfsz バイトのメモリ領域を、メッセージバッファ領域として使用します。メッセージバッファ領域内には、メッセージを管理するための情報も置くため、メッセージバッファ領域のすべてがメッセージを格納するために使えるわけではありません。TSZ_MBF を用いると、アプリケーションプログラムから、mbfsz に指定すべきサイズの目安を知ることができます。mbf に NULL (=0) が指定された場合には、mbfsz で指定されたサイズのメモリ領域を、コンフィグレーション時で定義したメモリプール用メモリ領域から自動で確保します。同期メッセージ機能を使用する場合には、mbfsz に 0 を指定します。

maxmsz に 0 が指定された場合や、65535 よりも大きい値が指定された場合には、E_PAR エラーを戻り値として返します。

del_mbf**メッセージバッファの削除****【書式】**

```
ER ercd = del_mbf(ID mbfid);
```

【パラメータ】

ID	mbfid	削除対象のメッセージバッファの ID 番号
----	-------	-----------------------

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (mbfid が不正あるいは使用できない)
E_CTX	コンテキストエラー (タスク以外からの呼び出し)
E_NOEXS	オブジェクト未生成 (対象メッセージバッファが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	不可
タイムイベントハンドラ	不可
割込みハンドラ	不可

【解説】

mbfid で指定される ID 番号のメッセージバッファを削除します。

snd_mbf	メッセージバッファへの送信
psnd_mbf	メッセージバッファへの送信（ポーリング）
tsnd_mbf	メッセージバッファへの送信（タイムアウトあり）

【書式】

```
ER ercd = snd_mbf(ID mbfid, VP msg, UINT msgsz);
```

```
ER ercd = psnd_mbf(ID mbfid, VP msg, UINT msgsz);
```

```
ER ercd = tsnd_mbf(ID mbfid, VP msg, UINT msgsz, TMO tmout);
```

【パラメータ】

ID	mbfid	送信対象のメッセージバッファの ID 番号
VP	msg	送信メッセージの先頭番地
UINT	msgsz	送信メッセージのサイズ（バイト数）
TMO	tmout	タイムアウト指定（tsnd_mbf のみ）

【戻り値】

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

【エラーコード】

E_PAR	パラメータエラー（msg, msgsz, tmout が不正）
E_ID	不正 ID 番号（mbfid が不正あるいは使用できない）
E_CTX	コンテキストエラー（タスク以外からの呼び出し：psnd_mbf 以外、 割込みハンドラからの呼び出し：psnd_mbf のみ、 ディスパッチ保留状態：psnd_mbf 以外）
E_NOEXS	オブジェクト未生成（対象メッセージバッファが未登録）
E_RLWAI	待ち状態の強制解除 (待ち状態の間に rel_wai を受付;psnd_mbf 以外)
E_TMOUT	ポーリング失敗またはタイムアウト（snd_mbf 以外）
E_DLT	待ちオブジェクトの削除 (待ち状態の間に対象メッセージバッファが削除;psnd_mbf 以外)

【呼び出しコンテキスト】

	snd_mbf	psnd_mbf	tsnd_mbf
タスク	可	可	可
初期化ハンドラ	不可	可	不可
タイムイベントハンドラ	不可	可	不可
割込みハンドラ	不可	不可	不可

【解説】

mbfid で指定される ID 番号のメッセージバッファに、msg で指定される番地から msgsz で指定されるバイト数のメモリ領域に格納されたメッセージを送信します。

対象メッセージバッファで受信を待っているタスクがある場合には、受信待ち行列の先頭のタスクが受信メッセージを格納する領域に送信メッセージをコピーし、そのタスクを待ち解除します。この時、待ち解除されたタスクに対しては、待ち状態に入ったシステムコールの戻り値として、送信メッセージのサイズ (**msgsz**) を返します。

対象メッセージバッファで受信を待っているタスクがない場合は、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待っているかどうかによって処理が異なります。対象メッセージバッファで送信を待っているタスクがない場合や、タスク優先度順の待ち行列において送信を待っているタスクの優先度がいずれも自タスクの優先度よりも低い場合には、送信メッセージをメッセージバッファの末尾にコピーします。この条件以外の場合や、メッセージバッファ領域に送信メッセージを格納するために必要な空き領域がない場合には、自タスクを送信待ち行列につなぎ、メッセージバッファへの送信待ち状態に遷移させます。

msgsz が、メッセージバッファの最大メッセージサイズよりも大きい場合には、**E_PAR** エラーを返します。また、**msgsz** に 0 が指定された場合にも、**E_PAR** エラーを戻り値として返します。

psnd_mbf は、**snd_mbf** の処理をポーリングで行うシステムコール、**tsnd_mbf** は、**snd_mbf** にタイムアウトの機能を付け加えたシステムコールです。**tmout** には、正の値のタイムアウト時間に加えて、**TMO_POL** (=0) と **TMO_FEVR** (=−1) を指定することができます。
μ C3/Standard では、**tmout** に **TMO_FEVR** を指定した **tsnd_mbf** は **snd_mbf** として扱い、**tmout** に **TMO_POL** を指定した **tsnd_mbf** は **psnd_mbf** として扱います。

rcv_mbf	メッセージバッファからの受信
prcv_mbf	メッセージバッファからの受信（ポーリング）
trcv_mbf	メッセージバッファからの受信（タイムアウトあり）

【書式】

```
ER_UINT msgsz = rcv_mbf(ID mbfid, VP msg);
ER_UINT msgsz = prcv_mbf(ID mbfid, VP msg);
ER_UINT msgsz = trcv_mbf(ID mbfid, VP msg, TMO tmout);
```

【パラメータ】

ID	mbfid	受信対象のメッセージバッファの ID 番号
VP	msg	受信メッセージを格納する先頭番地
TMO	tmout	タイムアウト指定（trcv_mbf のみ）

【戻り値】

ER_UINT	msgsz	受信メッセージのサイズ（バイト数、正の値）またはエラーコード
---------	-------	--------------------------------

【エラーコード】

E_PAR	パラメータエラー（msg、tmout が不正）
E_ID	不正 ID 番号（mbfid が不正あるいは使用できない）
E_CTX	コンテキストエラー（タスク以外からの呼び出し：prcv_mbf 以外、 割込みハンドラからの呼び出し：prcv_mbf のみ、 ディスパッチ保留状態：prcv_mbf 以外）
E_NOEXS	オブジェクト未生成（対象メッセージバッファが未登録）
E_RLWAI	待ち状態の強制解除（待ち状態の間に rel_wai を受付；prcv_mbf 以外）
E_TMOUT	ポーリング失敗またはタイムアウト（rcv_mbf 以外）
E_DLT	待ちオブジェクトの削除 (待ち状態の間に対象メッセージバッファが削除；prcv_mbf 以外)

【呼び出しコンテキスト】	rcv_mbf	prcv_mbf	trcv_mbf
タスク	可	可	可
初期化ハンドラ	不可	可	不可
タイムイベントハンドラ	不可	可	不可
割込みハンドラ	不可	不可	不可

【解説】

mbfid で指定される ID 番号のメッセージバッファからメッセージを受信し、msg で指定される番地以降に格納します。受信したメッセージのバイト数は、msgsz に返します。

対象メッセージバッファにメッセージが格納されている場合には、その先頭のメッセージを

msg で指定した番地以降にコピーし、そのメッセージサイズを **msgsz** に返します。コピーしたメッセージは、メッセージバッファ領域から削除します。メッセージバッファで送信を待っているタスクがある場合には、メッセージを削除した結果、メッセージバッファ領域に、送信待ち行列の先頭のタスクが送信しようとしているメッセージを格納するために必要な空き領域ができたかを調べ、可能であればメッセージをメッセージバッファの末尾にコピーし、そのタスクを待ち解除します。この時、待ち解除されたタスクに対しては、待ち状態に入ったシステムコールの戻り値として **E_OK** を返します。さらに、送信を待っているタスクが残っている場合には、新たに送信待ち行列の先頭になったタスクに対して同じ処理を繰り返します。

メッセージが格納されていない場合で、対象メッセージバッファで送信を待っているタスクがある場合には、送信待ち行列の先頭のタスクが送信しようとしているメッセージを **msg** で指定された番地以降にコピーし、そのタスクを待ち解除します。また、コピーしたメッセージのサイズを **msgsz** に返します。待ち解除されたタスクに対しては、待ち状態に入ったシステムコールの戻り値として **E_OK** を返します。

メッセージが格納されていない場合で、送信を待っているタスクもない場合には、自タスクを受信待ち行列につなぎ、メッセージバッファからの受信待ち状態に遷移させます。

prcv_mbf は、**rcv_mbf** の処理をポーリングで行うシステムコール、**trcv_mbf** は、**rcv_mbf** にタイムアウトの機能を付け加えたシステムコールです。**tmout** には、正の値のタイムアウト時間に加えて、**TMO_POL** (=0) と **TMO_FEVR** (=−1) を指定することができます。

μ C3/Standard では、**tmout** に **TMO_FEVR** を指定した **trcv_mbf** は **rcv_mbf** として扱い、**tmout** に **TMO_POL** を指定した **trcv_mbf** は **prcv_mbf** として扱います。

ref_mbf**メッセージバッファの状態参照****【書式】**

```
ER ercd = ref_mbf(ID mbfid, T_RMBF*pk_rmbf);
```

【パラメータ】

ID	mbfid	状態参照対象の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状態を返すパケットへのポインタ

【戻り値】

ER	ercd	正常終了 (E_OK) またはエラーコード
pk_rmbf の内容 (T_RMBF 型)		
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	smsgcnt	メッセージバッファに入っているメッセージの数
SIZE	fmbfsz	メッセージバッファ領域の空き領域のサイズ (バイト数、最低限の管理領域を除く)

【エラーコード】

E_ID	不正 ID 番号 (mbfid が不正あるいは使用できない)
E_CTX	コンテキストエラー (割込みハンドラからの呼び出し)
E_NOEXS	オブジェクト未生成 (対象メッセージバッファが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

mbfid で指定される ID 番号のメッセージバッファに関する状態を参照し、pk_rmbf で指定されるパケットに返します。

stskid には、対象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 番号を返します。送信を待っているタスクがない場合には、TSK_NONE (=0) を返します。

rtskid には、対象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 番号を返します。受信を待っているタスクがない場合には、TSK_NONE (=0) を返します。

smsgcnt には、対象メッセージバッファに現在入っているメッセージの個数を返します。

fmbfsz には、対象メッセージバッファ領域の空き領域のサイズ (バイト数) に返します。

5. 4. 7 ランデブ

cre_por	ランデブポートの生成
acre_por	ランデブポートの生成 (ID 番号自動割付け)

【書式】

```
ER ercd = cre_por(ID porid, T_CPOR*pk_cpor);
```

```
ER_ID porid = acre_por(T_CPOR*pk_cpor);
```

【パラメータ】

ID	porid	生成対象のランデブポートの ID 番号 (acre_por 以外)
T_CPOR*	pk_cpor	ランデブポート生成情報を入れたパケットへのポインタ pk_cpor の内容 (T_CPOR 型)
ATR	poratr	ランデブポート属性
UINT	maxcmsz	呼出しメッセージの最大サイズ (バイト数)
UINT	maxrmsz	返答メッセージの最大サイズ (バイト数)
VB const *	name	ランデブポートの名称 (文字列)

【戻り値】

cre_por の場合

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

acre_por の場合

ER_ID	porid	生成したランデブポートの ID 番号 (正の値) またはエラーコード
-------	-------	------------------------------------

【エラーコード】

E_RSATR	予約属性 (poratr が不正あるいは使用できない)
E_PAR	パラメータエラー (maxcmsz, maxrmsz が不正)
E_ID	不正 ID 番号 (porid が不正あるいは使用できない; cre_por のみ)
E_CTX	コンテキストエラー (タスクと初期化ハンドラ以外からの呼び出し)
E_NOMEM	メモリ不足 (管理領域が確保できない)
E_NOID	ID 番号不足 (割り付け可能なランデブポート ID がない; acre_por のみ)
E_OBJ	オブジェクト状態エラー (対象ランデブポートが登録済み; cre_por のみ)

【呼び出しコンテキスト】

	cre_por	acre_por
タスク	可	可
初期化ハンドラ	可	可
タイムイベントハンドラ	不可	不可
割込みハンドラ	不可	不可

【解説】

`porid` で指定される ID 番号を持つランデブポートを、`pk_cpor` で指定されるランデブポート生成情報に基づいて生成します。`poratr` はランデブポートの属性、`maxcmsz` は呼出しメッセージの最大サイズ（バイト数）、`maxrmsz` は返答メッセージの最大サイズ（バイト数）です。

`acre_por` は、生成するランデブポートの ID 番号をランデブポートが登録されていない ID 番号の中から一番大きな値を割り付け、その ID 番号を戻り値として返します。

`poratr` には、(`TA_TFIFO` || `TA_TPRI`) の指定ができます。ランデブポートの呼び出し待ち行列は、`TA_TFIFO` (=0x00) が指定された場合には FIFO 順に、`TA_TPRI` (=0x01) が指定された場合にはタスクの優先度順になります。

`maxcmsz` または `mazrmsz` に、65535 よりも大きい値が指定された場合には、`E_PAR` エラーを戻り値として返します。`maxcmsz` と `mazrmsz` に 0 を指定することもできます。

del_por**ランデブポートの削除****【書式】**

```
ER ercd = del_por(ID porid);
```

【パラメータ】

ID	porid	削除対象のランデブポートの ID 番号
----	-------	---------------------

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (porid が不正あるいは使用できない)
E_CTX	コンテキストエラー (タスク以外からの呼び出し)
E_NOEXS	オブジェクト未生成 (対象ランデブポートが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	不可
タイムイベントハンドラ	不可
割込みハンドラ	不可

【解説】

porid で指定される ID 番号のランデブポートを削除します。

cal_por	ランデブの呼出し
tcal_por	ランデブの呼出し（タイムアウトあり）

【書式】

```
ER_UINT rmsgsz = cal_por(ID porid, RDVPTN calptn, VP msg, UINT cmsgsz);
ER_UINT rmsgsz = tcal_por(ID porid, RDVPTN calptn, VP msg,
                           UINT cmsgsz, TMO tmout);
```

【パラメータ】

ID	porid	呼出し対象のランデブポートの ID 番号
RDVPTN	calptn	呼出し側のランデブ条件を示すビットパターン
VP	msg	呼出しメッセージの先頭番地／返答メッセージを格納する先頭番地
UINT	cmsgsz	呼出しメッセージのサイズ（バイト数）
TMO	tmout	タイムアウト指定（tcal_por のみ）

【戻り値】

ER_UINT	rmsgsz	返答メッセージのサイズ（バイト数、正の値または0） またはエラーコード
---------	--------	--

【エラーコード】

E_PAR	パラメータエラー（calptn, cmsgsz, tmout が不正）
E_ID	不正 ID 番号（porid が不正あるいは使用できない）
E_CTX	コンテキストエラー（タスク以外からの呼び出し、 ディスパッチ保留状態）
E_NOEXS	オブジェクト未生成（対象ランデブポートが未登録）
E_RLWAI	待ち状態の強制解除（呼出し待ち状態の間に rel_wai を受付）
E_TMOUT	ポーリング失敗またはタイムアウト（tcal_por のみ）
E_DLT	待ちオブジェクトの削除 (ランデブの呼出し待ち状態の間に対象ランデブポートが削除)

【呼び出しコンテキスト】

	cal_por	tcal_por
タスク	可	可
初期化ハンドラ	不可	不可
タイムイベントハンドラ	不可	不可
割込みハンドラ	不可	不可

【解説】

porid で指定される ID 番号のランデブポートに対して、calptn で指定されるランデブ条件により、ランデブを呼び出します。msg で指定される番地から cmsgsz で指定されるバイト数

のメモリ領域に格納されたメッセージを、呼出しメッセージとします。また、返答メッセージを **msg** で指定される番地以降に格納し、返答メッセージのバイト数を **rmsgsz** に返します。

対象ランデブポートにランデブ受付待ち状態のタスクがあり、そのタスクのランデブ条件と **calptn** で指定されたランデブ条件が成立する場合には、ランデブを成立させます。ランデブ受付待ち状態のタスクが複数ある場合には、受付待ち行列の先頭のタスクから順にランデブ条件を調べ、条件が成立する最初のタスクとの間でランデブを成立させます。

ランデブが成立した場合には、ランデブを識別するためのランデブ番号を割り付け、自タスクをランデブ終了待ち状態に遷移させます。また、成立したランデブの相手タスク（ランデブ受付待ち状態であったタスク）が呼出しメッセージを格納する領域に、**msg** と **cmsgsz** で指定される呼出しメッセージをコピーし、そのタスクを待ち解除します。待ち解除されたタスクに対しては、待ち状態に入ったシステムコールの戻り値として呼出しメッセージのサイズ（**cmsgsz**）とランデブ番号を返します。

対象ランデブポートにランデブ受付待ち状態のタスクがない場合や、ランデブ受付待ち状態のタスクがあってもランデブ条件が成立しない場合には、自タスクを呼出し待ち行列につなぎ、ランデブ呼出し待ち状態に遷移させます。

tcal_por は、**cal_por** にタイムアウトの機能を付け加えたシステムコールです。**tmout** には、正の値のタイムアウト時間に加えて、**TMO_FEVR** (=-1) を指定することができます。**tmout** に **TMO_POL** (=0) が指定された場合には、**E_PAR** エラーを返します。μ C3/Standard では、**tmout** に **TMO_POL** を指定した **tcal_por** は **cal_por** として扱います。

tcal_por が呼び出され、ランデブが成立した後にタイムアウトした場合、一旦成立したランデブを成立する前の状態には戻りません。この場合、ランデブ受付タスクには、ランデブを終了させようとした時点でエラーが報告されます。また、ランデブ終了待ち状態のタスクに対して **rel_wai** が呼び出され、ランデブ終了待ち状態が強制解除された場合、**E_RLWAI** エラーを返す違いはありますが、ランデブ受付タスクは同様に、ランデブを終了させようとした時点でエラーが報告されます。

calptn に 0 が指定された場合には、**E_PAR** エラーを戻り値として返します。また、**cmsgsz** が、対象ランデブポートの呼出しメッセージの最大サイズよりも大きい場合にも、**E_PAR** エラーを戻り値として返します。**cmsgsz** に 0 を指定することもできます。

acp_por	ランデブの受付
pacp_por	ランデブの受付（ポーリング）
tacp_por	ランデブの受付（タイムアウトあり）

【書式】

```

ER_UINT cmsgsz = acp_por(ID porid, RDVPTN acpptn,
                          RDVNO*p_rdvno, VP msg);

ER_UINT cmsgsz = pacp_por(ID porid, RDVPTN acpptn,
                          RDVNO*p_rdvno, VP msg);

ER_UINT cmsgsz = tacp_por(ID porid, RDVPTN acpptn,
                          RDVNO*p_rdvno, VP msg, TMO tmout);

```

【パラメータ】

ID	porid	受付対象のランデブポートの ID 番号
RDVPTN	acpptn	受付側のランデブ条件を示すビットパターン
VP	msg	呼出しメッセージを格納する先頭番地
TMO	tmout	タイムアウト指定（tacp_por のみ）

【戻り値】

ER_UINT	cmsgsz	呼出しメッセージのサイズ（バイト数、正の値または 0）またはエラーコード
RDVNO	rdvno	成立したランデブ番号

【エラーコード】

E_PAR	パラメータエラー（acpptn が不正）
E_ID	不正 ID 番号（porid が不正あるいは使用できない）
E_CTX	コンテキストエラー（タスク以外からの呼び出し、 ディスパッチ保留状態）
E_NOEXS	オブジェクト未生成（対象ランデブポートが未登録）
E_RLWAI	待ち状態の強制解除（待ち状態の間に rel_wai を受付；pacp_por 以外）
E_TMOUT	ポーリング失敗またはタイムアウト（acp_por 以外）
E_DLT	待ちオブジェクトの削除 (待ち状態の間に対象ランデブポートが削除；pacp_por 以外)

【呼び出しコンテキスト】	acp_por	pacp_por	tacp_por
タスク	可	可	可
初期化ハンドラ	不可	不可	不可
タイムイベントハンドラ	不可	不可	不可
割込みハンドラ	不可	不可	不可

【解説】

porid で指定される ID 場号のランデブポートに対して、acpptn で指定されるランデブ条件により、ランデブを受け付けます。呼出しメッセージは msg で指定される番地以降に格納し、呼出しメッセージのバイト数を cmsgsz に、割り付けたランデブ番号を rdvno に返します。

対象ランデブポートにランデブ呼出し待ち状態のタスクがあり、acpptn で指定されたランデブ条件とそのタスクのランデブ条件が成立する場合には、ランデブを成立させます。ランデブ呼出し待ち状態のタスクが複数ある場合には、呼出し待ち行列の先頭のタスクから順にランデブ条件を調べ、条件が成立する最初のタスクとの間でランデブを成立させます。

ランデブが成立した場合には、ランデブを識別するためのランデブ番号を割り付け、rdvno に返します。成立したランデブの相手のタスク（ランデブ呼出し待ち状態であったタスク）の呼出しメッセージを msg で指定された番地以降にコピーし、そのメッセージサイズを cmsgsz に返します。ランデブの相手タスクは、ランデブ呼出し待ち行列からはずし、ランデブ終了待ち状態に遷移させます。

対象ランデブポートにランデブ呼出し待ち状態のタスクがない場合や、ランデブ呼出し待ち状態のタスクがあってもランデブ条件が成立しない場合には、自タスクを受付待ち行列につなぎ、ランデブ受付待ち状態に遷移させます。

pacp_por は、acp_por の処理をポーリングで行うシステムコール、tacp_por は、acp_por にタイムアウトの機能を付け加えたシステムコールです。tmout には、正の値のタイムアウト時間に加えて、TMO_POL (=0) と TMO_FEXR (= -1) を指定することができます。

μ C3/Standard では、tmout に TMO_FEXR を指定した tacp_por は acp_por として扱い、tmout に TMO_POL を指定した tacp_por は pacp_por として扱います。

acpptn に 0 が指定された場合には、E_PAR エラーを戻り値として返します。

fwd_por**ランデブの回送****【書式】**

```
ER ercd = fwd_por(ID porid, RDVPTN calptn, RDVNO rdvno,
                  VP msg, UINT cmsgsz);
```

【パラメータ】

ID	porid	回送先のランデブポートの ID 番号
RDVPTN	calptn	呼出し側のランデブ条件を示すビットパターン
RDVNO	rdvno	回送するランデブ番号
VP	msg	呼出しメッセージの先頭番地
UINT	cmsgsz	呼出しメッセージのサイズ (バイト数)

【戻り値】

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

【エラーコード】

E_PAR	パラメータエラー (calptn, cmsgsz が不正)
E_ID	不正 ID 番号 (porid が不正あるいは使用できない)
E_CTX	コンテキストエラー (タスク以外からの呼び出し)
E_ILUSE	サービスコール不正使用 (回送先のランデブポートの 返答メッセージの最大サイズが大きすぎる)
E_OBJ	オブジェクト状態エラー (rdvno が不正)
E_NOEXS	オブジェクト未生成 (対象ランデブポートが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	不可
タイムイベントハンドラ	不可
割込みハンドラ	不可

【解説】

rdvno で指定されるランデブ番号を割り付けられたランデブを、porid で指定される ID 番号のランデブポートに対して、calptn で指定されるランデブ条件により回送します。msg で指定される番地から cmsgsz で指定されるバイト数のメモリ領域に格納されたメッセージを、回送後の呼出しメッセージとします。ランデブを受け付けたタスク以外が、このシステムコールによりランデブを回送することもできます。

fwd_por を呼び出すと、rdvno で指定されるランデブを呼び出したタスク (以下、これを呼出しタスクとする)が、porid で指定されるランデブポートに対して、fwd_por のパラメータとして指定されたランデブ条件と呼出しメッセージにより、ランデブを呼び出したのと同じ結果

になります。

回送先のランデブポートにランデブ受付待ち状態のタスクがあり、そのタスクのランデブ条件と **calptn** で指定されたランデブ条件が成立する場合には、呼出しタスクとの間でランデブを成立させます。ランデブ受付待ち状態のタスクが複数ある場合には、受付待ち行列の先頭のタスクから順にランデブ条件を調べ、条件が成立する最初のタスクとの間でランデブを成立させます。

ランデブが成立した場合には、新たなランデブ番号を割り付け、呼出しタスクを新しいランデブに対するランデブ終了待ち状態に遷移させます。また、成立したランデブの相手タスク（ランデブ受付待ち状態であったタスク）が呼出しメッセージを格納する領域に、**msg** と **cmsgsz** で指定される呼出しメッセージをコピーし、そのタスクを待ち解除します。待ち解除されたタスクに対しては、待ち状態に入ったシステムコールの戻り値として呼出しメッセージのサイズ（**cmsgsz**）と新たなランデブ番号を返します。

回送先のランデブポートにランデブ受付待ち状態のタスクがない場合や、ランデブ受付待ち状態のタスクがあってもランデブ条件が成立しない場合には、呼出しタスクを回送先のランデブポートの呼出し待ち行列につなぎ、ランデブ呼出し待ち状態に遷移させます。この時、呼出しタスクが返答メッセージを格納する領域に、**msg** と **cmsgsz** で指定される呼出しメッセージをコピーします。

回送先のランデブポートの返答メッセージの最大サイズは、回送するランデブが成立したランデブポートの返答メッセージの最大サイズ以下でなければなりません。この条件が満たされない場合には、**E_ILUSE** エラーを戻り値として返します。

cmsgsz が、回送先のランデブポートの呼出しメッセージの最大サイズよりも大きい場合や、回送するランデブが成立したランデブポートの返答メッセージの最大サイズよりも大きい場合には、**E_PAR** エラーを返します。**cmsgsz** に 0 を指定することもできます。

呼出しタスクが、指定されたランデブの終了待ち状態でない場合には、**E_OBJ** エラーを戻り値として返します。また、**rdvno** に指定された値が、ランデブ番号として認識できなかった場合にも、**E_OBJ** エラーを戻り値として返します。

calptn に 0 が指定された場合には、**E_PAR** エラーを戻り値として返します。

rpl_rdv **ランデブの終了**

【書式】

```
ER ercd = rpl_rdv(RDVNO rdvno, VP msg, UINT rmsgsz);
```

【パラメータ】

RDVNO	rdvno	終了させるランデブ番号
VP	msg	返答メッセージの先頭番地
UINT	rmsgsz	返答メッセージのサイズ (バイト数)

【戻り値】

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

【エラーコード】

E_PAR	パラメータエラー (msg, rmsgsz が不正)
E_CTX	コンテキストエラー (タスク以外からの呼び出し)
E_OBJ	オブジェクト状態エラー (rdvno が不正)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	不可
タイムイベントハンドラ	不可
割込みハンドラ	不可

【解説】

rdvno で指定されるランデブ番号を割り付けたランデブを、msg で指定される番地から rmsgsz で指定されるバイト数のメモリ領域に格納されたメッセージを返答メッセージとして、終了させます。ランデブを受け付けたタスク以外が、このシステムコールによりランデブを終了させることもできます。

rdvno で指定されたランデブを呼び出したタスクが、指定されたランデブの終了待ち状態である場合には、相手タスクが返答メッセージを格納する領域に msg と rmsgsz で指定される返答メッセージをコピーし、相手タスクを待ち解除します。また、待ち解除された相手タスクに対しては、待ち状態に入ったシステムコールの戻り値として返答メッセージのサイズ(rmsgsz)を返します。

rdvno で指定されたランデブを呼び出したタスクが、指定されたランデブの終了待ち状態でない場合には、E_OBJ エラーを戻り値として返します。また、rdvno に指定された値が、ランデブ番号として認識できない場合にも、E_OBJ エラーを戻り値として返します。

rmsgsz が、ランデブが成立したランデブポートの返答メッセージの最大サイズよりも大きい場合には、E_PAR エラーを戻り値として返します。rmsgsz に 0 を指定することもできます。

ref_por**ランデブポートの状態参照****【書式】**

```
ER ercd = ref_por(ID porid, T_RPOR*pk_rpor);
```

【パラメータ】

ID	porid	状態参照対象のランデブポートの ID 番号
T_RPOR*	pk_rpor	ランデブポート状態を返すパケットへのポインタ

【戻り値】

ER	ercd	正常終了 (E_OK) またはエラーコード
pk_rpor の内容 (T_RPOR 型)		
ID	ctskid	ランデブポートの呼出し待ち行列の先頭のタスクの ID 番号
ID	atskid	ランデブポートの受付待ち行列の先頭のタスクの ID 番号

【エラーコード】

E_ID	不正 ID 番号 (porid が不正あるいは使用できない)
E_CTX	コンテキストエラー (割込みハンドラからの呼び出し)
E_NOEXS	オブジェクト未生成 (対象ランデブポートが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

porid で指定される ID 番号のランデブポートに関する状態を参照し、pk_rpor で指定されるパケットに返します。

ctskid には、対象ランデブポートの呼出し待ち行列の先頭のタスクの ID 番号を返します。ランデブ呼出し待ち状態で待っているタスクがない場合には、TSK_NONE (=0) を返します。

atskid には、対象ランデブポートの受付待ち行列の先頭のタスクの ID 番号を返します。ランデブ受付待ち状態で待っているタスクがない場合には、TSK_NONE (=0) を返します。

ref_rdv**ランデブの状態参照****【書式】**

```
ER ercd = ref_rdv(RDVNO rdvno, T_RRDV*pk_rrdv);
```

【パラメータ】

RDVNO	rdvno	状態参照対象のランデブ番号
T_RRDV*	pk_rrdv	ランデブ状態を返すパケットへのポインタ

【戻り値】

ER	ercd	正常終了 (E_OK) またはエラーコード
pk_rrdv の内容 (T_RRDV 型)		
ID	wtskid	ランデブ終了待ち状態のタスクの ID 番号

【エラーコード】

E_CTX	コンテキストエラー (割込みハンドラからの呼び出し)
-------	----------------------------

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

rdvno で指定されるランデブ番号を割り付けられたランデブに関する状態を参照し、pk_rrdv で指定されるパケットに返します。

rdvno で指定されたランデブを呼び出したタスクが、指定されたランデブの終了待ち状態である場合には、そのタスクの ID 番号を wtskid に返します。そのタスクが指定されたランデブの終了待ち状態でない場合や、rdvno に指定された値がランデブ番号として認識できない場合、wtskid には TSK_NONE (=0) を返します。

5. 5 メモリプール管理機能

5. 5. 1 固定長メモリプール

cre_mpf	固定長メモリプールの生成
acre_mpf	固定長メモリプールの生成 (ID 番号自動割付け)

【書式】

ER ercd = cre_mpf(ID mpfid, T_CMPF*pk_cmpf);

ER_ID mpfid = acre_mpf(T_CMPF*pk_cmpf);

【パラメータ】

ID	mpfid	生成対象の固定長メモリプールの ID 番号 (acre_mpf 以外)
T_CMPF*	pk_cmpf	固定長メモリプール生成情報を入れたパケットへのポインタ
pk_cmpf の内容 (T_CMPF 型)		
ATR	mpfatr	固定長メモリプール属性
UINT	blkcnt	獲得できるメモリブロック数 (個数)
UINT	blksiz	メモリブロックのサイズ (バイト数)
VP	mpf	固定長メモリプール領域の先頭番地
VB const *	name	固定長メモリプールの名称 (文字列)

【戻り値】

cre_mpf の場合

ER ercd 正常終了 (E_OK) またはエラーコード

acre_mpf の場合

ER_ID mpfid 生成し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_RSATR	予約属性 (mpfatr が不正あるいは使用できない)
E_PAR	パラメータエラー (blkcnt, blksiz が不正)
E_ID	不正 ID 番号 (mpfid が不正あるいは使用できない ; cre_mpf のみ)
E_CTX	コンテキストエラー (タスクと初期化ハンドラ以外からの呼び出し)
E_NOMEM	メモリ不足 (管理領域かメモリプール領域が確保できない)
E_NOID	ID 番号不足 (割り付け可能な固定長メモリプール ID がない ; acre_mpf のみ)
E_OBJ	オブジェクト状態エラー (対象固定長メモリプールが登録済み ; cre_mpf のみ)

【呼び出しコンテキスト】	cre_mpf	acre_mpf
タスク	可	可
初期化ハンドラ	可	可
タイムイベントハンドラ	不可	不可
割込みハンドラ	不可	不可

【解説】

mpfid で指定される ID 番号を持つ固定長メモリプールを、pk_mpf で指定される固定長メモリプール生成情報に基づいて生成します。mpfatr は固定長メモリプールの属性、blkcnt は固定長メモリプールから獲得できるメモリブロックの個数、blksz は獲得するメモリブロックのサイズ（バイト数）、mpf は固定長メモリプール領域の先頭番地です。

acre_mpf は、生成する固定長メモリプールの ID 番号を固定長メモリプールが登録されていない ID 番号の中から一番大きな値を割り付け、その ID 番号を戻り値として返します。

mpfatr には、(TA_TFIFO || TA_TPRI) の指定ができます。固定長メモリプールの待ち行列は、TA_TFIFO (=0x00) が指定された場合には FIFO 順に、TA_TPRI (=0x01) が指定された場合にはタスクの優先度順になります。

mpf で指定された番地から、blksz バイトのメモリブロックを blkcnt 個獲得できるのに必要なサイズのメモリ領域を、メモリプール領域として使用します。

TSZ_MPF を用いると、アプリケーションプログラムから、blksz バイトのメモリブロックを blkcnt 個獲得できるのに必要なサイズを知ることができます。mpf に NULL (=0) が指定された場合には、コンフィグレーション時に定義したメモリプール用メモリ領域から自動で確保します。

blkcnt または blksz に 0 が指定された場合、E_PAR エラーを戻り値として返します。

del_mpf**固定長メモリの削除****【書式】**

```
ER ercd = del_mpf(ID mpfid);
```

【パラメータ】

ID	mpfid	削除対象の固定長メモリの ID 番号
----	-------	--------------------

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (mpfid が不正あるいは使用できない)
E_CTX	コンテキストエラー (タスク以外からの呼び出し)
E_NOEXS	オブジェクト未生成 (対象固定長メモリプールが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	不可
タイムイベントハンドラ	不可
割込みハンドラ	不可

【解説】

mpfid で指定される ID 番号の固定長メモリプールを削除します。

get_mpf	固定長メモリブロックの獲得
pget_mpf	固定長メモリブロックの獲得（ポーリング）
tget_mpf	固定長メモリブロックの獲得（タイムアウトあり）

【書式】

```
ER ercd = get_mpf(ID mpfid, VP *p_blk);
ER ercd = pget_mpf(ID mpfid, VP *p_blk);
ER ercd = tget_mpf(ID mpfid, VP *p_blk, TMO tmout);
```

【パラメータ】

ID	mpfid	メモリブロック獲得対象の固定長メモリプールの ID 番号
TMO	tmout	タイムアウト指定（tget_mpf のみ）

【戻り値】

ER	ercd	正常終了（E_OK）またはエラーコード
VP	blk	獲得したメモリブロックの先頭番地

【エラーコード】

E_ID	不正 ID 番号（mpfid が不正、あるいは使用できない）
E_CTX	コンテキストエラー（タスク以外からの呼び出し：pget_mpf 以外、 割込みハンドラからの呼び出し：pget_mpf のみ、 ディスパッチ保留状態：pget_mpf 以外）
E_NOEXS	オブジェクト未生成（対象固定長メモリプールが未登録）
E_RLWAI	待ち状態の強制解除（待ち状態の間に rel_wai を受付；pget_mpf 以外）
E_TMOUT	ポーリング失敗またはタイムアウト（get_mpf 以外）
E_DLT	待ちオブジェクトの削除 (待ち状態の間に固定長メモリプールが削除；pget_mpf 以外)

【呼び出しコンテキスト】

	get_mpf	pget_mpf	tget_mpf
タスク	可	可	可
初期化ハンドラ	不可	可	不可
タイムイベントハンドラ	不可	可	不可
割込みハンドラ	不可	不可	不可

【解説】

mpfid で指定される ID 番号の固定長メモリプールのメモリ領域に空きメモリブロックがある場合には、その内のいずれかを選んで獲得された状態とし、その先頭番地を blk に返します。空きメモリブロックがない場合には、自タスクを待ち行列につなぎ、固定長メモリブロックの獲得待ち状態に遷移させます。

pget_mpf は get_mpf の処理をポーリングで行うシステムコール、tget_mpf は get_mpf にタイムアウトの機能を付け加えたシステムコールです。また、tmout に、TMO_POL (=0) や TMO_FEVR (= -1) を指定することもできます。μ C3/Standard では、tmout に TMO_FEVR を指定した tget_mpf は get_mpf として扱い、tmout に TMO_POL を指定した tget_mpf は pget_mpf として扱います。

rel_mpf **固定長メモリブロックの返却**

【書式】

```
ER ercd = rel_mpf(ID mpfid, VP blk);
```

【パラメータ】

ID	mpfid	メモリブロック返却対象の固定長メモリプールの ID 番号
VP	blk	返却するメモリブロックの先頭番地

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (mpfid が不正、あるいは使用できない)
E_CTX	コンテキストエラー (割込みハンドラからの呼び出し)
E_NOEXS	オブジェクト未生成 (対象固定長メモリプールが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

mpfid で指定される ID 場号の固定長メモリプールで、メモリブロックの獲得を待っているタスクがない場合には、blk を先頭番地とするメモリブロックをその固定長メモリプールのメモリ領域に返却します。

獲得を待っているタスクがある場合には、返却したメモリブロックを待ち行列の先頭のタスクに獲得させ、そのタスクを待ち解除します。また、待ち解除されたタスクには、待ち状態に入ったシステムコールの戻り値として E_OK を返し、固定長メモリブロックから獲得したメモリブロックの先頭番地として blk の値を返します。

返却するメモリブロックの先頭番地は、mpfid で指定される固定長メモリプールから取得されたメモリブロックの先頭番地を返したもので、まだ返却されていないものでなければなりません。

ref_mpf**固定長メモリの状態参照****【書式】**

```
ER ercd = ref_mpf(ID mpfid, T_RMPF *pk_rmpf);
```

【パラメータ】

ID	mpfid	状態参照対象の固定長メモリの ID 番号
T_RMPF*	pk_rmpf	固定長メモリ状態を返すパケットへのポインタ

【戻り値】

ER	ercd	正常終了 (E_OK) またはエラーコード
pk_rmpf の内容 (T_RMPF 型)		
ID	wtskid	固定長メモリの待ち行列の先頭のタスクの ID 番号
UINT	fblkcnt	固定長メモリの空きメモリブロック数 (個数)

【エラーコード】

E_ID	不正 ID 番号 (mpfid が不正、あるいは使用できない)
E_CTX	コンテキストエラー (割込みハンドラからの呼び出し)
E_NOEXS	オブジェクト未生成 (対象固定長メモリプールが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

mpfid で指定される ID 番号の固定長メモリプールに関する状態を参照し、pk_rmpf で指定されるパケットに返します。

wtskid には、対象固定長メモリの待ち行列の先頭のタスクの ID 番号を返します。メモリブロックの獲得を待っているタスクがない場合には、TSK_NONE (=0) を返します。

fblkcnt には、対象固定長メモリプール領域内の空きメモリブロックの個数を返します。

mpfid には、コンフィグレータで生成した際の固定長メモリプール ID の定義名を使って指定します。

5. 5. 2 可変長メモリプール

cre_mpl	可変長メモリプールの生成
acre_mpl	可変長メモリプールの生成 (ID 番号自動割付け)

【書式】

```
ER ercd = cre_mpl(ID mplid, T_CMPL*pk_cmpl);
```

```
ER_ID mplid = acre_mpl(T_CMPL*pk_cmpl);
```

【パラメータ】

ID	mplid	生成対象の可変長メモリプールの ID 番号 (acre_mpl 以外)
T_CMPL*	pk_cmpl	可変長メモリプール生成情報を入れたパケットへのポインタ pk_cmpl の内容 (T_CMPL 型)
ATR	mplatr	可変長メモリプール属性
SIZE	mplsz	可変長メモリプール領域のサイズ (バイト数)
VP	mpl	可変長メモリプール領域の先頭番地
VB const *	name	可変長メモリプールの名称 (文字列)

【戻り値】

cre_mpl の場合

ER ercd 正常終了 (E_OK) またはエラーコード

acre_mpl の場合

ER_ID mplid 生成した可変長メモリプールの ID 番号 (正の値) またはエラーコード

【エラーコード】

E_RSATR	予約属性 (mplatr が不正あるいは使用できない)
E_PAR	パラメータエラー (mplsz が不正)
E_ID	不正 ID 番号 (mplid が不正あるいは使用できない ; cre_mpl のみ)
E_CTX	コンテキストエラー (割込みハンドラからの呼び出し)
E_NOMEM	メモリ不足 (メモリプール領域などが確保できない)
E_NOID	ID 番号不足 (割り付け可能な可変長メモリプール ID がない ; acre_mpl のみ)
E_OBJ	オブジェクト状態エラー (対象可変長メモリプールが登録済み ; cre_mpl のみ)

【呼び出しコンテキスト】	cre_mpl	cre_mpl
タスク	可	可
初期化ハンドラ	可	可
タイムイベントハンドラ	不可	不可
割込みハンドラ	不可	不可

【解説】

mplid で指定される ID 番号を持つ可変長メモリプールを、pk_cmpl で指定される可変長メモリプール生成情報に基づいて生成します。mplatr は可変長メモリプールの属性、mplsz は可変長メモリプール領域のサイズ（バイト数）、mpl は可変長メモリプール領域の先頭番地です。

acre_mpl は、生成する可変長メモリプールの ID 番号を可変長メモリプールが登録されていない ID 番号の中から一番大きな値を割り付け、その ID 番号を戻り値として返します。

mplatr には、(TA_TFIFO || TA_TPRI) の指定ができます。可変長メモリプールの待ち行列は、T_TFIFO (=0x00) が指定された場合には FIFO 順に、TA_TPRI (=0x01) が指定された場合にはタスクの優先度順になります。

mpl で指定された番地から mplsz バイトのメモリ領域を、メモリプール領域として使用します。また、mpl に NULL (=0) が指定された場合には、コンフィグレーションで定義したメモリプール用メモリ領域から、mplsz バイトのメモリ領域を自動で確保します。

mplsz に 0 が指定された場合には、E_PAR エラーを戻り値として返します。

del_mpl **可変長メモリプールの削除**

【書式】

```
ER ercd = del_mpl(ID mplid);
```

【パラメータ】

ID	mplid	削除対象の可変長メモリプールの ID 番号
----	-------	-----------------------

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (mplid が不正あるいは使用できない)
E_CTX	コンテキストエラー (タスク以外からの呼び出し)
E_NOEXS	オブジェクト未生成(対象可変長メモリプールが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	不可
タイムイベントハンドラ	不可
割込みハンドラ	不可

【解説】

mplid で指定される ID 番号の可変長メモリプールを削除します。

get_mpl	可変長メモリブロックの獲得
pget_mpl	可変長メモリブロックの獲得（ポーリング）
tget_mpl	可変長メモリブロックの獲得（タイムアウトあり）

【書式】

```
ER ercd = get_mpl(ID mplid, UINT blksize, VP * p_blk);
ER ercd = pget_mpl(ID mplid, UINT blksize, VP * p_blk);
ER ercd = tget_mpl(ID mplid, UINT blksize, VP * p_blk, TMO tmout);
```

【パラメータ】

ID	mplid	メモリブロック獲得対象の可変長メモリプールの ID 番号
UINT	blksize	獲得するメモリブロックのサイズ（バイト数）
TMO	tmout	タイムアウト指定（tget_mpl のみ）

【戻り値】

ER	ercd	正常終了（E_OK）またはエラーコード
VP	blk	獲得したメモリブロックの先頭番地

【エラーコード】

E_PAR	パラメータエラー（blksize が不正）
E_ID	不正 ID 番号（mplid が不正あるいは使用できない）
E_CTX	コンテキストエラー（タスク以外からの呼び出し：pget_mpl 以外、 割込みハンドラからの呼び出し：pget_mpl のみ、 ディスパッチ保留状態：pget_mpl 以外）
E_NOEXS	オブジェクト未生成(対象可変長メモリプールが未登録)
E_RLWAI	待ち状態の強制解除 (待ち状態の間に rel_wai を受付；pget_mpl 以外)
E_TMOUT	ポーリング失敗またはタイムアウト（get_mpl 以外）
E_DLT	待ちオブジェクトの削除 (待ち状態の間に対象可変長メモリプールが削除；pget_mpl 以外)

【呼び出しコンテキスト】

	get_mpl	pget_mpl	tget_mpl
タスク	可	可	可
初期化ハンドラ	不可	可	不可
タイムイベントハンドラ	不可	可	不可
割込みハンドラ	不可	不可	不可

【解説】

mplid で指定される ID 番号の可変長メモリプールから、blksize で指定されるサイズのメモリ

ブロックを獲得し、その先頭番地を **blk** に返します。

処理内容は、自タスクより優先してメモリブロックを獲得できるタスクが待っているかどうかによって異なります。対象可変長メモリプールでメモリブロックの獲得を待っているタスクがない場合や、タスク優先度順の待ち行列で、待っているタスクの優先度がいずれも自タスクの優先度よりも低い場合には、メモリプール領域から **blkksz** バイトのメモリブロックを獲得します。この条件を満たさない場合や、ブロックを獲得するのに十分な空きメモリ領域がない場合には、自タスクを待ち行列につなぎ、可変長メモリブロックの獲得待ち状態に遷移させます。

可変長メモリブロックの獲得を待っているタスクが、**rel_wai** や **ter_tsk** により待ち解除されたり、タイムアウトにより待ち解除されたりした結果、待ち行列の先頭のタスクが変化する場合には、新たに先頭になったタスクから順に可能であることを調べ、可能であればメモリブロックを獲得させます。

pget_mpl は **get_mpl** の処理をポーリングで行うシステムコール、**tget_mpl** は **get_mpl** にタイムアウトの機能を付け加えたシステムコールです。**tmout** には、正の値のタイムアウト時間に加えて、**TMO_POL**(=0) と **TMO_FEVR**(=-1) を指定することができます。 μ C3/Standard では、**tmout** に **TMO_FEVR** を指定した **tget_mpl** は **get_mpl** として扱い、**tmout** に **TMO_POL** を指定した **tget_mpl** は **pget_mpl** として扱います。

blkksz に 0 が指定された場合や、可変長メモリプールから取得可能な最大のメモリブロックサイズよりも大きい値が **blkksz** に指定された場合は、**E_PAR** エラーを戻り値として返します。

rel_mpl**可変長メモリブロックの返却****【書式】**

```
ER ercd = rel_mpl(ID mplid, VP blk);
```

【パラメータ】

ID	mplid	メモリブロック返却対象の可変長メモリプールの ID 番号
VP	blk	返却するメモリブロックの先頭番地

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (mplid が不正あるいは使用できない)
E_CTX	コンテキストエラー (割込みハンドラからの呼び出し)
E_NOEXS	オブジェクト未生成 (対象可変長メモリプールが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

mplid で指定される ID 番号の可変長メモリプールに対して、blk を先頭番地とするメモリブロックを返却します。

対象可変長メモリプールでメモリブロックの獲得を待っているタスクがある場合には、メモリブロックを返却した結果、待ち行列の先頭のタスクが獲得しようとしているサイズのメモリブロックが獲得できるようになったかを調べます。獲得できる場合には、そのタスクにメモリブロックを獲得させ、そのタスクを待ち解除します。この時、待ち解除されたタスクに対しては、待ち状態に入ったサービスコールの戻り値として E_OK を返し、可変長メモリブロックから獲得したメモリブロックの先頭番地として獲得したメモリブロックの先頭番地を返します。さらに、メモリブロックの獲得を待っているタスクが残っている場合には、新たに待ち行列の先頭になったタスクに対して同じ処理を繰り返します。

メモリブロックを返却する対象の可変長メモリプールは、メモリブロックの獲得を行った同じ ID 番号の可変長メモリプールでなければなりません。

また、返却するメモリブロックの先頭番地は、get_mpl, pget_mpl, tget_mpl のいずれかのサービスコールが、獲得したメモリブロックの先頭番地として返したもので、まだ返却されていないものでなければなりません。

ref_mpl **可変長メモリプールの状態参照**

【書式】

```
ER ercd = ref_mpl(ID mplid, T_RMPL*pk_rmpl);
```

【パラメータ】

ID	mplid	状態参照対象の可変長メモリプールの ID 番号
T_RMPL*	pk_rmpl	可変長メモリプール状態を返すパケットへのポインタ

【戻り値】

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

pk_rmpl の内容 (T_RMPL 型)

ID	wtskid	可変長メモリプールの待ち行列の先頭のタスクの ID 番号
SIZE	fmplsz	可変長メモリプールの空き領域の合計サイズ (バイト数)
UINT	fblksiz	すぐに獲得可能な最大メモリブロックサイズ (バイト数)

【エラーコード】

E_ID	不正 ID 番号 (mplid が不正あるいは使用できない)
E_CTX	コンテキストエラー (割込みハンドラからの呼び出し)
E_NOEXS	オブジェクト未生成 (対象可変長メモリプールが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

mplid で指定される ID 番号の可変長メモリプールに関する状態を参照し、pk_rmpl で指定されるパケットに返します。

wtskid には、対象可変長メモリプールの待ち行列の先頭のタスクの ID 番号を返します。メモリブロックの獲得を待っているタスクがない場合には、TSK_NONE (=0) を返します。

fmplsz には、対象可変長メモリプールの現在の空き領域の合計サイズ(バイト数)を返します。

fblksiz には、対象可変長メモリプールからすぐに獲得できる最大のメモリブロックサイズ (バイト数) を返します。

5. 6 時間管理機能

5. 6. 1 システム時刻管理

set_tim システム時刻の設定

【書式】

```
ER ercd = set_tim(SYSTIM *p_systim);
```

【パラメータ】

SYSTIM	systim	システム時刻に設定する時刻
--------	--------	---------------

【戻り値】

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 (割込みハンドラからの呼び出し)
-------	----------------------------

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

現在のシステム時刻を、`systim` で示される時刻に設定します。また、システム時刻を変更することによって、すでに呼び出されたシステムコールのタイムアウト時刻は変更されません。

get_tim

システム時刻の参照

【書式】

```
ER ercd = get_tim(SYSTIM *p_system);
```

【パラメータ】なし

【戻り値】

ER	ercd	正常終了 (E_OK) またはエラーコード
SYSTIM	system	現在のシステム時刻

【エラーコード】

E_CTX	コンテキストエラー (割込みハンドラからの呼び出し)
-------	----------------------------

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

現在のシステム時刻を呼び出し、system に返します。

isig_tim

タイムチックの供給

【書式】

```
ER ercd = isig_tim();
```

【パラメータ】

なし

【戻り値】

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 (割込みハンドラ以外からの呼び出し)
E_NOMEM	メモリ不足 (SSB が不足)

【呼び出しコンテキスト】

タスク	不可
初期化ハンドラ	不可
タイムイベントハンドラ	不可
割込みハンドラ	可

【解説】

システム時刻にチック時間を加算します。

5. 6. 2 周期ハンドラ

cre_cyc	周期ハンドラの生成
acre_cyc	周期ハンドラの生成 (ID 番号自動割付け)

【書式】

```
ER ercd = cre_cyc(ID cycid, T_CCYC*pk_ccyc);
```

```
ER_ID cycid = acre_cyc(T_CCYC*pk_ccyc);
```

【パラメータ】

ID	cycid	生成対象の周期ハンドラの ID 番号 (acre_cyc 以外)
T_CCYC*	pk_ccyc	周期ハンドラ生成情報を入れたパケットへのポインタ pk_ccyc の内容 (T_CCYC 型)
ATR	cycatr	周期ハンドラ属性
VP_INT	exinf	周期ハンドラの拡張情報
FP	cychdr	周期ハンドラの起動番地
RELTIM	cycetim	周期ハンドラの起動周期
RELTIM	cycphs	周期ハンドラの起動位相
VB const *	name	周期ハンドラの名称 (文字列)

【戻り値】

cre_cyc の場合

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

acre_cyc の場合

ER_ID	cycid	生成した周期ハンドラの ID 番号 (正の値) またはエラーコード
-------	-------	-----------------------------------

【エラーコード】

E_PAR	パラメータエラー (cycetim が不正)
E_ID	不正 ID 番号 (cycid が不正あるいは使用できない ; cre_cyc のみ)
E_CTX	コンテキストエラー (タスクと初期化ハンドラ以外からの呼び出し)
E_NOMEM	メモリ不足 (管理領域が確保できない)
E_NOID	ID 番号不足 (割り付け可能な周期ハンドラ ID がない ; acre_cyc のみ)
E_OBJ	オブジェクト状態エラー (対象周期ハンドラが登録済み ; cre_cyc のみ)

【呼び出しコンテキスト】

	cre_cyc	acre_cyc
タスク	可	可
初期化ハンドラ	可	可
タイムイベントハンドラ	不可	不可
割込みハンドラ	不可	不可

【解説】

cycid で指定される ID 番号を持つ周期ハンドラを、pk_ccyc で指定される周期ハンドラ生成情報に基づいて生成します。cycatr は周期ハンドラの属性、exinf は周期ハンドラを起動する時にパラメータとして渡す拡張情報、cychdr は周期ハンドラの起動番地、cyctim は周期ハンドラを起動する周期、cycphs は周期ハンドラを起動する位相です。

acre_cyc は、生成する周期ハンドラの ID 番号を周期ハンドラが登録されていない ID 番号の中から一番大きな値を割り付け、その ID 番号を戻り値として返します。

cycatr には、((TA_HLNG || TA_ASM) | [TA_STA] | [TA_PHS]) の指定ができます。TA_STA (=0x02) が指定された場合には、動作している状態で周期ハンドラを生成します。そうでない場合には、周期ハンドラを動作していない状態で生成します。TA_PHS (=0x04) が指定された場合には、周期ハンドラの生成時の位相を保存します。

cyctim に 0 が指定された場合には、E_PAR エラーを戻り値として返します。

del_cyc	周期ハンドラの削除
----------------	------------------

【書式】

```
ER ercd = del_cyc(ID cycid);
```

【パラメータ】

ID	cycid	削除対象の周期ハンドラの ID 番号
----	-------	--------------------

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (cycid が不正あるいは使用できない)
E_CTX	コンテキストエラー (タスク以外からの呼び出し)
E_NOEXS	オブジェクト未生成 (対象周期ハンドラが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	不可
タイムイベントハンドラ	不可
割込みハンドラ	不可

【解説】

cycid で指定される ID 番号の周期ハンドラを削除します。

sta_cyc**周期ハンドラの動作開始****【書式】**

```
ER ercd = sta_cyc(ID cycid);
```

【パラメータ】

ID	cycid	動作開始対象の周期ハンドラの ID 番号
----	-------	----------------------

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (cycid が不正、あるいは使用できない)
E_CTX	コンテキストエラー (割込みハンドラからの呼び出し)
E_NOEXS	オブジェクト未生成 (対象タスクが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

cycid で指定される ID 番号の周期ハンドラを、動作している状態に遷移させます。また、システムコールが呼び出された時刻に、周期ハンドラの起動周期を加えた時刻を、周期ハンドラを次に起動すべき時刻とします。この時、すでに動作中だった場合には、次に起動すべき時刻の更新のみ行います。TA_PHS 属性が指定されている場合には、動作していない状態は動作状態に遷移させ、動作している状態であれば何もしません。

stp_cyc**周期ハンドラの動作停止****【書式】**

```
ER ercd = stp_cyc(ID cycid);
```

【パラメータ】

ID	cycid	動作停止対象の周期ハンドラの ID 番号
----	-------	----------------------

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (cycid が不正、あるいは使用できない)
E_CTX	コンテキストエラー (割込みハンドラからの呼び出し)
E_NOEXS	オブジェクト未生成 (対象周期ハンドラが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

cycid で指定される ID 番号の周期ハンドラを、動作していない状態に遷移させます。動作していない状態の場合には、何もしません。

ref_cyc **周期ハンドラの状態参照**

【書式】

```
ER ercd = ref_cyc(ID cycid, T_RCYC *pk_rcyc);
```

【パラメータ】

ID	cycid	状態参照対象の周期ハンドラの ID 番号
T_RCYC*	pk_rcyc	周期ハンドラ状態を返すパケットへのポインタ

【戻り値】

ER	ercd	正常終了 (E_OK) またはエラーコード
pk_rcyc の内容 (T_RCYC 型)		
STAT	cycstat	周期ハンドラの動作状態
RELTIM	lefttim	周期ハンドラを次に起動する時刻までの時間

【エラーコード】

E_ID	不正 ID 番号 (cycid が不正、あるいは使用できない)
E_CTX	コンテキストエラー (割込みハンドラからの呼び出し)
E_NOEXS	オブジェクト未生成 (対象タスクが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

cycid で指定される ID 番号の周期ハンドラに関する状態を参照し、pk_rcyc で指定されるパケットに返します。

cycstat には、対象周期ハンドラが動作している状態か、動作していない状態かによって、次のいずれかの値を返します。

TCYC_STP	0x00	周期ハンドラが動作していない
TCYC_STA	0x01	周期ハンドラが動作している

lefttim には、対象周期ハンドラが動作している状態の場合に、対象周期ハンドラを次に起動する時刻までの時間を返します。ただし、lefttim に返す値は、次に周期ハンドラが起動するまでの保証する時間です。そのため、次のタイムチェックで周期ハンドラが起動される場合には、lefttim に 0 を返します。対象周期ハンドラが動作していない状態の場合には、lefttim に返す値は不定値です。また、周期ハンドラ内から自身の状態参照した場合も、lefttim に返す値は不定値です。

5. 6. 3 アラームハンドラ

cre_alm	アラームハンドラの生成
acre_alm	アラームハンドラの生成 (ID 番号自動割付け)

【書式】

```
ER ercd = cre_alm(ID almid, T_CALM*pk_calm);
```

```
ER_ID almid = acre_alm(T_CALM*pk_calm);
```

【パラメータ】

ID	almid	生成対象のアラームハンドラの ID 番号 (acre_alm 以外)
T_CALM*	pk_calm	アラームハンドラ生成情報を入れたパケットへの ポインタ
pk_calm の内容 (T_CALM 型)		
ATR	almatr	アラームハンドラ属性
VP_INT	exinf	アラームハンドラの拡張情報
FP	almhdr	アラームハンドラの起動番地
VB const *	name	アラームハンドラの名称 (文字列)

【戻り値】

cre_alm の場合		
ER	ercd	正常終了 (E_OK) またはエラーコード
acre_alm の場合		
ER_ID	almid	生成したアラームハンドラの ID 番号 (正の値) またはエラーコード

【エラーコード】

E_ID	不正 ID 番号 (almid が不正あるいは使用できない ; cre_alm のみ)
E_CTX	コンテキストエラー (タスクと初期化ハンドラ以外からの呼び出し)
E_NOMEM	メモリ不足 (管理領域が確保できない)
E_NOID	ID 番号不足 (割り付け可能なアラームハンドラ ID がない ; acre_alm のみ)
E_OBJ	オブジェクト状態エラー (対象アラームハンドラが登録済み ; cre_alm のみ)

【呼び出しコンテキスト】	cre_alm	cre_alm
タスク	可	可
初期化ハンドラ	可	可
タイムイベントハンドラ	不可	不可
割込みハンドラ	不可	不可

【解説】

almid で指定される ID 番号を持つアラームハンドラを、pk_calm で指定されるアラームハンドラ生成情報に基づいて生成します。almatr はアラームハンドラの属性、exinf はアラームハンドラを起動する時にパラメータとして渡す拡張情報、almhdr はアラームハンドラの起動番地です。

acre_alm は、生成するアラームハンドラの ID 番号をアラームハンドラが登録されていない ID 番号の中から一番大きな値を割り付け、その ID 番号を返値として返します。

アラームハンドラの生成直後には、アラームハンドラの起動時刻は設定されておらず、アラームハンドラの動作は停止しています。

almatr には、(TA_HLNG) のみの指定ができます。

del_alm**アラームハンドラの削除****【書式】**

```
ER ercd = del_alm(ID almid);
```

【パラメータ】

ID	almid	削除対象のアラームハンドラの ID 番号
----	-------	----------------------

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (almid が不正あるいは使用できない)
E_CTX	コンテキストエラー (タスク以外からの呼び出し)
E_NOEXS	オブジェクト未生成 (対象アラームハンドラが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	不可
タイムイベントハンドラ	不可
割込みハンドラ	不可

【解説】

almid で指定される ID 番号のアラームハンドラを削除します。

sta_alm**アラームハンドラの動作開始****【書式】**

```
ER ercd = sta_alm(ID almid, RELTIM almtim);
```

【パラメータ】

ID	almid	動作開始対象の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）

【戻り値】

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

【エラーコード】

E_PAR	パラメータエラー（almtim が不正）
E_ID	不正 ID 番号（almid が不正あるいは使用できない）
E_CTX	コンテキストエラー（割込みハンドラからの呼び出し）
E_NOEXS	オブジェクト未生成（対象アラームハンドラが未登録）

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

almid で指定される ID 番号のアラームハンドラの起動時刻を、システムコールが呼び出された時刻から almtim で指定された相対時間後に設定し、アラームハンドラの動作を開始します。

すでに動作しているアラームハンドラが指定された場合には、以前の起動時刻の設定を解除し、新しい起動時刻を再設定します。

stp_alm**アラームハンドラの動作停止****【書式】**

```
ER ercd = stp_alm(ID almid);
```

【パラメータ】

ID	almid	動作停止対象のアラームハンドラの ID 番号
----	-------	------------------------

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (almid が不正あるいは使用できない)
E_CTX	コンテキストエラー (割込みハンドラからの呼び出し)
E_NOEXS	オブジェクト未生成 (対象アラームハンドラが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

almid で指定される ID 番号のアラームハンドラの起動時刻の設定を解除し、アラームハンドラの動作を停止します。動作していないアラームハンドラが指定された場合には、何もありません。

ref_alm

アラームハンドラの状態参照

【書式】

```
ER ercd = ref_alm(ID almid, T_RALM*pk_ralm);
```

【パラメータ】

ID	almid	状態参照対象のアラームハンドラの ID 番号
T_RALM*	pk_ralm	アラームハンドラ状態を返すパケットへのポインタ

【戻り値】

ER	ercd	正常終了 (E_OK) またはエラーコード
pk_ralm の内容 (T_RALM 型)		
STAT	almstat	アラームハンドラの動作状態
RELTIM	lefttim	アラームハンドラの起動時刻までの時間

【エラーコード】

E_ID	不正 ID 番号 (almid が不正あるいは使用できない)
E_CTX	コンテキストエラー (割込みハンドラからの呼び出し)
E_NOEXS	オブジェクト未生成 (対象アラームハンドラが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

almid で指定される ID 番号のアラームハンドラに関する状態を参照し、pk_ralm で指定されるパケットに返します。

almstat には、対象アラームハンドラが動作しているかどうかによって、次のいずれかの値を返します。

TALM_STP	0x00	アラームハンドラが動作していない
TALM_STA	0x01	アラームハンドラが動作している

lefttim には、対象アラームハンドラが動作している場合に、対象アラームハンドラの起動時刻までの相対時間を返します。ただし、lefttim に返す値は、対象アラームハンドラが起動されるまでの保証された時間です。そのため、次のタイムチェックでアラームハンドラが起動される場合には、lefttim に 0 を返します。対象アラームハンドラが動作していない場合には、lefttim に返す値は不定値です。また、アラームハンドラ内から自身の状態参照した場合も、lefttim に返す値は不定値です。

5. 6. 4 オーバランハンドラ

def_ovr オーバランハンドラの定義

【書式】

```
ER ercd = def_ovr(T_DOVR * pk_dovr);
```

【パラメータ】

T_DOVR*	pk_dovr	オーバランハンドラ定義情報を入れたパケットへのポインタ
pk_dovr の内容 (T_DOVR 型)		
ATR	ovratr	オーバランハンドラ属性
FP	ovrhdr	オーバランハンドラの起動番地

【戻り値】

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

【エラーコード】

E_RSATR	予約属性 (ovratr が不正あるいは使用できない)
E_CTX	コンテキストエラー (タスクと初期化ハンドラ以外からの呼び出し)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	不可
割込みハンドラ	不可

【解説】

pk_dovr で指定されるオーバランハンドラ定義情報に基づいて、オーバランハンドラを定義します。ovratr はオーバランハンドラの属性、ovrhdr はオーバランハンドラの起動番地です。

pk_dovr に NULL (=0) が指定されると、すでに定義されているオーバランハンドラの定義を解除し、オーバランハンドラが定義されていない状態にします。

この時、すべてのタスクの上限プロセッサ時間の設定は解除されます。また、すでにオーバランハンドラが定義されている状態で、再度オーバランハンドラが定義された場合には、以前の定義を解除し、新しい定義に置き換えます。タスクの上限プロセッサ時間の設定解除は行いません。

ovratr には、(TA_HLNG) のみが指定できます。

ivsig_ovr**オーバランハンドラの時刻を更新****【書式】**

```
ER ercd = ivsig_ovr();
```

【パラメータ】

なし

【戻り値】

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（割込みハンドラ以外からの呼び出し）
E_NOMEM	メモリ不足（SSB が不足）

【呼び出しコンテキスト】

タスク	不可
初期化ハンドラ	不可
タイムイベントハンドラ	不可
割込みハンドラ	可

【解説】

実行中タスクのオーバランハンドラの動作が開始中だった場合には、使用プロセッサ時間を更新します。

sta_ovr**オーバランハンドラの動作開始****【書式】**

```
ER ercd = sta_ovr(ID tskid, OVRTIM ovrtime);
```

【パラメータ】

ID	tskid	動作開始対象のタスクの ID 番号
OVRTIM	ovrtim	設定するタスクの上限プロセッサ時間

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (tskid が不正あるいは使用できない)
E_CTX	コンテキストエラー (割込みハンドラからの呼び出し)
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)
E_NOEXS	オブジェクト未生成 (対象タスクが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

tskid で指定される ID 番号のタスクに対して、オーバランハンドラの動作を開始します。具体的には、対象タスクの上限プロセッサ時間を ovrtime で指定される時間に設定し、使用プロセッサ時間を 0 にクリアします。

すでに上限プロセッサ時間が設定されているタスクが指定された場合には、以前の上限プロセッサ時間の設定を解除し、新しい上限プロセッサ時間を再設定します。この時にも、使用プロセッサ時間を 0 にクリアします。

tskid に TSK_SELF (=0) が指定されると、自タスクを対象タスクとします。

tskid で指定されるタスクが終了しても、上限プロセッサ時間は解除せず保持され、使用プロセッサ時間も保持されます。

stp_ovr**オーバランハンドラの動作停止****【書式】**

```
ER ercd = stp_ovr(ID tskid);
```

【パラメータ】

ID	tskid	動作停止対象のタスクの ID 番号
----	-------	-------------------

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (tskid が不正あるいは使用できない)
E_CTX	コンテキストエラー (割込みハンドラからの呼び出し)
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)
E_NOEXS	オブジェクト未生成 (対象タスクが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

tskid で指定される ID 番号のタスクに対して、オーバランハンドラの動作を停止します。上限プロセッサ時間が設定されていないタスクが指定された場合には、何もしません。

tskid に TSK_SELF (=0) が指定されると、自タスクを対象タスクとします。

ref_ovr**オーバランハンドラの状態参照****【書式】**

```
ER ercd = ref_ovr(ID tskid, T_ROVR*pk_rovr);
```

【パラメータ】

ID	tskid	状態参照対象のタスクの ID 番号
T_ROVR*	pk_rovr	オーバランハンドラ状態を返すパケットへのポインタ

【戻り値】

ER	ercd	正常終了 (E_OK) またはエラーコード
pk_rovr の内容 (T_ROVR 型)		
STAT	ovrstat	オーバランハンドラの動作状態
OVRTIM	leftotm	残りのプロセッサ時間

【エラーコード】

E_ID	不正 ID 番号 (tskid が不正あるいは使用できない)
E_CTX	コンテキストエラー (割込みハンドラからの呼び出し)
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)
E_NOEXS	オブジェクト未生成 (対象タスクが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

tskid で指定される ID 番号のタスクの、オーバランハンドラに関する状態を参照し、pk_rovr で指定されるパケットに返します。

ovrstat には、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かを返します。

TOVR_STP	0x00	上限プロセッサ時間が設定されていない
TOVR_STA	0x01	上限プロセッサ時間が設定されている

leftotm には、対象タスクに対して上限プロセッサ時間が設定されている場合に、対象タスクを原因として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を返します。具体的には、対象タスクの上限プロセッサ時間から使用プロセッサ時間を減じた値を返します。また、対象タスクに対して上限プロセッサ時間が設定されていない場合には不定値を返します。

tskid に TSK_SELF (=0) が指定されると、自タスクを対象タスクとします。

5. 7 システム状態管理機能

rot_rdq		タスクの優先順位の回転
irot_rdq		
【書式】		
ER ercd = rot_rdq(PRI tskpri) ;		
ER ercd = irot_rdq(PRI tskpri) ;		
【パラメータ】		
PRI	tskpri	優先順位を回転する対象の優先度
【戻り値】		
ER	ercd	正常終了（E_OK）またはエラーコード
【エラーコード】		
E_PAR	パラメータエラー（tskpri が不正）	
E_NOMEM	メモリ不足（SSB が不足；割込みハンドラからの呼び出し）	
【呼び出しコンテキスト】		
	rot_rdq	irot_rdq
タスク	可	可
初期化ハンドラ	可	可
タイムイベントハンドラ	可	可
割込みハンドラ	可	可

【解説】

tskpri で指定される優先度のタスクの優先順位を回転します。つまり、対象優先度を持った実行できる状態のタスクの中で、最も高い優先順位を持つタスクを、同じ優先度を持つタスクの中で最低の優先順位にします。tskpri に TPRI_SELF (=0) が指定されると、自タスクのベース優先度を対象優先度になります。

【推奨】

μ C3/Standard の rot_rdq と irot_rdq は、同じシステムコールとして実装するため、呼び出しコンテキストに関係なく、同じ使い方ができます。ただし、タスクコンテキストから呼び出す場合には rot_rdq を、それ以外の場合には irot_rdq を使うことをお勧めします。

get_tid	実行状態のタスク ID の参照
iget_tid	

【書式】

```
ER ercd = get_tid(ID *p_tskid);
ER ercd = iget_tid(ID *p_tskid);
```

【パラメータ】

なし

【戻り値】

ER	ercd	正常終了 (E_OK) またはエラーコード
ID	tskid	実行状態のタスクの ID 番号

【エラーコード】

特記すべきエラーはない

【呼び出しコンテキスト】

get_tid

iget_tid

タスク	可	可
初期化ハンドラ	可	可
タイムイベントハンドラ	可	可
割込みハンドラ	可	可

【解説】

実行状態のタスクの ID 番号を参照し、tskid に返します。非タスクコンテキストから呼び出された場合で、実行状態のタスクがない時には、tskid に TSK_NONE (=0) を返します。

【推奨】

μ C3/Standard の get_tid と iget_tid は、同じシステムコールとして実装するため、呼び出しコンテキストに関係なく、同じ使い方ができます。ただし、タスクコンテキストから呼び出す場合には get_tid を、それ以外の場合には iget_tid を使うことをお勧めします。

loc_cpu	CPU ロック状態への移行
iloc_cpu	

【書式】

```
ER ercd = loc_cpu();
```

```
ER ercd = iloc_cpu();
```

【パラメータ】

なし

【戻り値】

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

【エラーコード】

特記すべきエラーはない

【呼び出しコンテキスト】**loc_cpu****iloc_cpu**

タスク

可

可

初期化ハンドラ

可

可

タイムイベントハンドラ

可

可

割込みハンドラ

可

可

【解説】

CPU ロック状態に遷移します。CPU ロック状態で呼び出された場合には何もしません。

CPU ロック状態は、プロセッサに依存し、この詳細は「プロセッサ依存部マニュアル」を参照してください。

【推奨】

μ C3/Standard の loc_cpu と iloc_cpu は、同じシステムコールとして実装するため、呼び出しコンテキストに関係なく、同じ使い方ができます。ただし、タスクコンテキストから呼び出す場合には loc_cpu を、それ以外の場合には iloc_cpu を使うことをお勧めします。

unl_cpu	CPU ロック状態の解除
iunl_cpu	

【書式】

```
ER ercd = unl_cpu();
```

```
ER ercd = iunl_cpu();
```

【パラメータ】

なし

【戻り値】

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

【エラーコード】

特記すべきエラーはない

【呼び出しコンテキスト】

	unl_cpu	iunl_cpu
タスク	可	可
初期化ハンドラ	可	可
タイムイベントハンドラ	可	可
割込みハンドラ	可	可

【解説】

CPU ロック解除状態に移行します。CPU ロック解除状態で呼び出された場合には何もありません。CPU ロック解除状態は、プロセッサに依存し、この詳細は「プロセッサ依存部マニュアル」を参照してください。

【推奨】

μ C3/Standard の unl_cpu と iunl_cpu は、同じシステムコールとして実装するため、呼び出しコンテキストに関係なく、同じ使い方ができます。ただし、タスクコンテキストから呼び出す場合には unl_cpu を、それ以外の場合には iunl_cpu を使うことをお勧めします。

dis_dsp

ディスパッチの禁止

【書式】

```
ER ercd = dis_dsp();
```

【パラメータ】

なし

【戻り値】

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 (タスク以外からの呼び出し)
-------	--------------------------

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	不可
タイムイベントハンドラ	不可
割込みハンドラ	不可

【解説】

ディスパッチ禁止状態に遷移します。ディスパッチ禁止状態で呼び出された場合には何もありません。

ena_dsp ディスパッチの許可

【書式】

```
ER ercd = ena_dsp();
```

【パラメータ】

なし

【戻り値】

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 (タスク以外からの呼び出し)
-------	--------------------------

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	不可
タイムイベントハンドラ	不可
割込みハンドラ	不可

【解説】

ディスパッチ許可状態に移行します。ディスパッチ許可状態で呼び出された場合には何もありません。

sns_ctx**コンテキストの参照****【書式】**

```
BOOL state = sns_ctx();
```

【パラメータ】

なし

【戻り値】

BOOL

state

コンテキスト

【呼び出しコンテキスト】

タスク

可

初期化ハンドラ

可

タイムイベントハンドラ

可

割込みハンドラ

可

【解説】

非タスクコンテキストから呼び出された場合に TRUE、タスクコンテキストから呼び出された場合に FALSE を返します。

sns_loc

CPU ロック状態の参照

【書式】

```
BOOL state = sns_loc();
```

【パラメータ】なし

【戻り値】

BOOL	state	CPU ロック状態
------	-------	-----------

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	可

【解説】

システムが CPU ロック状態の場合に TRUE、CPU 解除状態の場合に FALSE を返します。

sns_dsp

ディスパッチ禁止状態の参照

【書式】

```
BOOL state = sns_dsp();
```

【パラメータ】

なし

【戻り値】

BOOL	state	ディスパッチ禁止状態
------	-------	------------

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	可

【解説】

システムがディスパッチ禁止状態の場合に TRUE、ディスパッチ許可状態の場合に FALSE を返します。

sns_dpn

ディスパッチ保留状態の参照

【書式】

```
BOOL state = sns_dpn();
```

【パラメータ】なし

【戻り値】

BOOL	state	ディスパッチ保留状態
------	-------	------------

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	可

【解説】

システムがディスパッチ保留状態の場合に **TRUE**、それ以外の場合に **FALSE** を返します。つまり、CPU ロック状態か、ディスパッチ禁止状態か、割込みレベルがタスクレベルより高いかのいずれかの場合に、**TRUE** を返します。

ref_sys**システムの状態参照****【書式】**

```
ER ercd = ref_sys(T_RSYS *pk_rsys);
```

【パラメータ】

T_RSYS*	pk_rsys	システム状態を返すパケットへのポインタ
---------	---------	---------------------

【戻り値】

ER	ercd	正常終了 (E_OK) またはエラーコード
pk_rsys の内容 (T_RSYS) 型		
SIZE	fsyssz	システムメモリの空き領域の合計サイズ
SIZE	fstksz	スタック用メモリの空き領域の合計サイズ
SIZE	fmpsz	メモリプール用メモリの空き領域の合計サイズ
UH	utskid	生成済みタスク ID の個数
UH	usemid	生成済みセマフォ ID の個数
UH	uflgid	生成済みイベントフラグ ID の個数
UH	udtqid	生成済みデータキューID の個数
UH	umbxid	生成済みメールボックス ID の個数
UH	umbfid	生成済みメッセージバッファ ID の個数
UH	uporid	生成済みランデブポート ID の個数
UH	umpfid	生成済み固定長メモリプール ID の個数
UH	umplid	生成済み可変長メモリプール ID の個数
UH	ualmid	生成済みアラームハンドラ ID の個数
UH	ucycid	生成済み周期ハンドラ ID の個数
UH	uisrid	生成済み割込みサービスルーチン ID の個数
UH	ssbent	SSB が最少になった時の個数

【エラーコード】

E_CTX	コンテキストエラー (割込みハンドラからの呼び出し)
-------	----------------------------

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

システムの状態を参照し、pk_rsys で指定されるパケットに返します。

5. 8 割込み管理機能

def_inh 割込みハンドラの定義

【書式】

```
ER ercd = def_inh(INHNO inhno, T_DINH*pk_dinh);
```

【パラメータ】

INHNO	inhno	定義対象の割込みハンドラ番号
T_DINH*	pk_dinh	割込みハンドラ定義情報を入れたパケットへのポインタ
pk_dinh の内容 (T_DINH 型)		
ATR	inhatr	割込みハンドラ属性
FP	inthdr	割込みハンドラの起動番地
IMASK	imask	割込みハンドラの割込みレベル

【戻り値】

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

【エラーコード】

E_PAR	パラメータエラー (inhno, imask が不正)
E_NOMEM	メモリ不足 (管理領域が確保できない)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	不可

【解説】

inhno で指定される割込みハンドラ番号に対して、pk_dinh で指定される割込みハンドラ定義情報に基づいて、割込みハンドラを定義します。inhatr は割込みハンドラの属性、inthdr は割込みハンドラの起動番地、imask は割込みハンドラの割込みレベルです。

この割込みハンドラ番号は、プロセッサに依存し、詳細は「プロセッサ依存部マニュアル」を参照してください。

pk_dinh に NULL (=0) が指定されると、すでに定義されている割込みハンドラの定義を解除します。また、すでに割込みハンドラが定義されている割込みハンドラ番号に対して、再度割込みハンドラを定義した場合には、新しい定義に置き換えます。

μ C3/Standard では、割込みハンドラ番号と割込みサービスルーチンの割込み番号は同じ意味を持ちます。したがって、すでに割込みサービスルーチンに使用された割込み番号に、割込みハンドラを定義しようとした場合には、E_PAR エラーを戻り値として返します。

cre_isr	割込みサービスルーチンの生成
acre_isr	割込みサービスルーチンの生成（ID 番号自動割付け）

【書式】

```
ER ercd = cre_isr(ID isrid, T_CISR*pk_cisr);
```

```
ER_ID isrid = acre_isr(T_CISR*pk_cisr);
```

【パラメータ】

ID	isrid	生成対象の割込みサービスルーチンの ID 番号 (cre_isr のみ)
T_CISR*	pk_cisr	割込みサービスルーチン生成情報を入れたパケット へのポインタ
pk_cisr の内容 (T_CISR 型)		
ATR	isratr	割込みサービスルーチン属性
VP_INT	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付加する割込み番号
FP	isr	割込みサービスルーチンの起動番地
IMASK	imask	割込みサービスルーチンの割込みレベル

【戻り値】

cre_isr の場合

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

acre_isr の場合

ER_ID	isrid	生成し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_PAR	パラメータエラー (intno が不正)
E_ID	不正 ID 番号 (isrid が不正あるいは使用できない ; cre_isr のみ)
E_CTX	コンテキストエラー (タスクと初期化ハンドラ以外からの呼び出し)
E_NOMEM	メモリ不足 (管理領域が確保できない)
E_NOID	ID 番号不足 (割り付け可能な割込みサービスルーチン ID がない ; acre_isr のみ)
E_OBJ	オブジェクト状態エラー (対象割込みサービスルーチンが登録済み ; cre_isr のみ)

【呼び出しコンテキスト】

	cre_isr	acre_isr
タスク	可	可
初期化ハンドラ	可	可
タイムイベントハンドラ	不可	不可
割込みサービスルーチン	不可	不可

【解説】

`isrid` で指定される ID 番号を持つ割り込みサービスルーチンを、`pk_cisr` で指定される割り込みサービスルーチン生成情報に基づいて生成します。`isratr` は割り込みサービスルーチンの属性、`exinf` は割り込みサービスルーチンを起動する時にパラメータとして渡す拡張情報、`intno` は割り込みサービスルーチンを起動する割り込みを指定する割り込み番号、`isr` は割り込みサービスルーチンの起動番地、`imask` は割り込みハンドラの割り込みレベルです。

この割り込み番号は、プロセッサに依存し、詳細は「プロセッサ依存部マニュアル」を参照してください。

`acre_isr` は、生成する割り込みサービスルーチンの ID 番号を割り込みサービスルーチンが登録されていない ID 番号の中から一番大きな値を割り付け、その ID 番号を戻り値として返します。

μ C3/Standard では、割り込みハンドラ番号と割り込みサービスルーチンの割り込み番号は同じ意味を持ちます。したがって、すでに割り込みハンドラに使用された割り込みハンドラ番号に、割り込みサービスルーチンを生成しようとした場合には、**E_PAR** エラーを戻り値として返します。

`isratr` には、(TA_HLNG) のみの指定ができます。

del_isr**割込みサービスルーチンの削除****【書式】**

```
ER ercd = del_isr(ID isrid);
```

【パラメータ】

ID	isrid	削除対象の割込みサービスルーチンの ID 番号
----	-------	-------------------------

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (isrid が不正あるいは使用できない)
E_CTX	コンテキストエラー (タスク以外からの呼び出し)
E_NOEXS	オブジェクト未生成 (対象割込みサービスルーチンが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	不可
タイムイベントハンドラ	不可
割込みサービスルーチン	不可

【解説】

isrid で指定される ID 番号の割込みサービスルーチンを削除します。

ref_isr**割込みサービスルーチンの状態参照****【書式】**

```
ER ercd = ref_isr(ID isrid, T_RISR*pk_risr);
```

【パラメータ】

ID	isrid	状態参照対象の割込みサービスルーチンの ID 番号
T_RISR*	pk_risr	割込みサービスルーチン状態を返すパケットへのポインタ

【戻り値】

ER	ercd	正常終了 (E_OK) またはエラーコード
pk_risr の内容 (T_RISR 型)		
INTNO	intno	割込みサービスルーチンを付加する割込み番号
FP	isr	割込みサービスルーチンの起動番地

【エラーコード】

E_ID	不正 ID 番号 (isrid が不正あるいは使用できない)
E_CTX	コンテキストエラー (タスク以外からの呼び出し)
E_NOEXS	オブジェクト未生成 (対象割込みサービスルーチンが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みサービスルーチン	不可

【解説】

isrid で指定される ID 番号の割込みサービスルーチンの状態を参照し、pk_risr で指定されるパケットに返します。

この割込み番号は、プロセッサに依存し、詳細は「プロセッサ依存部マニュアル」を参照してください。

dis_int

割込みの禁止

【書式】

```
ER ercd = dis_int(INTNO intno);
```

【パラメータ】

INTNO	intno	割込みを禁止する割込み番号
-------	-------	---------------

【戻り値】

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

【エラーコード】

E_PAR	パラメータエラー (intno が不正)
-------	----------------------

【呼び出しコンテキスト】

タスク	不可
初期化ハンドラ	不可
タイムイベントハンドラ	不可
割込みサービスルーチン	不可

【解説】

このシステムコールは、実装されているかどうかも含め、プロセッサに依存し、詳細は「プロセッサ依存部マニュアル」を参照してください。

ena_int

割込みの許可

【書式】

```
ER ercd = ena_int(INTNO intno);
```

【パラメータ】

INTNO	intno	割込みを許可する割込み番号
-------	-------	---------------

【戻り値】

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

【エラーコード】

E_PAR	パラメータエラー (intno が不正)
-------	----------------------

【呼び出しコンテキスト】

タスク	不可
初期化ハンドラ	不可
タイムイベントハンドラ	不可
割込みサービスルーチン	不可

【解説】

このシステムコールは、実装されているかどうかも含め、プロセッサに依存し、詳細は「プロセッサ依存部マニュアル」を参照してください。

chg_ims**割込みマスクの変更****【書式】**

```
ER ercd = chg_ims(IMASK imask);
```

【パラメータ】

IMASK	imask	変更後の割込みマスク
-------	-------	------------

【戻り値】

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

【エラーコード】

特記すべきエラーはない

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	---
割込みサービスルーチン	---

【解説】

プロセッサの割込みマスクレベルを、**imask** で指定される値に変更します。この割込みマスクレベルは、プロセッサに依存し、この詳細は「プロセッサ依存部マニュアル」を参照してください。

呼び出しコンテキストについては、「プロセッサ依存部マニュアル」を参照してください。

get_ims**割込みマスクの参照****【書式】**

```
ER ercd = get_ims(IMASK *p_imask);
```

【パラメータ】

なし

【戻り値】

ER	ercd	正常終了 (E_OK) またはエラーコード
IMASK	imask	現在の割込みマスク

【エラーコード】

特記すべきエラーはない

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みサービスルーチン	可

【解説】

プロセッサの割込みマスクレベルを参照し、**imask** に返します。この割込みマスクレベルは、プロセッサに依存し、この詳細は「プロセッサ依存部マニュアル」を参照してください。

5. 9 システム構成管理機能

def_exc CPU 例外ハンドラの定義

【書式】

```
ER ercd = def_exc(EXCNO excno, T_DEXC*pk_dexc);
```

【パラメータ】

EXCNO	excno	定義対象の CPU 例外ハンドラ番号
T_DEXC*	pk_dexc	CPU 例外ハンドラ定義情報を入れたパケットへのポインタ

pk_dexc の内容 (T_DEXC 型)

ATR	excattr	CPU 例外ハンドラ属性
FP	exchdr	CPU 例外ハンドラの起動番地

【戻り値】

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

【エラーコード】

E_PAR	パラメータエラー (excno が不正)
E_CTX	コンテキストエラー (タスク以外からの呼び出し)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みサービスルーチン	不可

【解説】

excno で指定される CPU 例外ハンドラ番号に対して、pk_dexc で指定される CPU 例外ハンドラ定義情報に基づいて、CPU 例外ハンドラを定義します。excattr は CPU 例外ハンドラの属性、exchdr は CPU 例外ハンドラの起動番地です。

pk_dexc に NULL (=0) が指定されると、すでに定義されている CPU 例外ハンドラの定義を解除します。また、すでに CPU 例外ハンドラが定義されている CPU 例外ハンドラ番号に対して、再度 CPU 例外ハンドラを定義した場合には、新しい定義に置き換えます。

このシステムコールは、実装されているかどうかも含め、プロセッサに依存し、詳細は「プロセッサ依存部マニュアル」を参照してください。

ref_cfg**コンフィグレーション情報の参照****【書式】**

```
ER ercd = ref_cfg(T_RCFG * pk_rcfg);
```

【パラメータ】

T_RCFG*	pk_rcfg	コンフィグレーション情報を返すパケットへのポインタ
---------	---------	---------------------------

【戻り値】

ER	ercd	正常終了 (E_OK) またはエラーコード
pk_rcfg の内容 (T_RCFG 型)		
UH	tskpri_max	タスク優先度の上限
UH	tskid_max	タスク ID の上限
UH	semid_max	セマフォ ID の上限
UH	flgid_max	イベントフラグ ID の上限
UH	dtqid_max	データキューID の上限
UH	mbxid_max	メールボックス ID の上限
UH	mbfid_max	メッセージバッファ ID の上限
UH	porid_max	ランデブポート ID の上限
UH	mpfid_max	固定長メモリプール ID の上限
UH	mplid_max	可変長メモリプール ID の上限
UH	almid_max	アラームハンドラ ID の上限
UH	cycid_max	周期ハンドラ ID の上限
UH	isrid_max	割込みサービスルーチン ID の上限
UH	devid_max	デバイスドライバ ID の上限
UH	tick	タイムチックの周期時間
UH	ssb_cnt	SSB の生成個数

【エラーコード】

特記すべきエラーはない

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みサービスルーチン	可

【解説】

システムのコンフィグレーション参照し、pk_rcfg で指定されるパケットに返します。

ref_ver

バージョン情報の参照

【書式】

```
ER ercd = ref_ver(T_RVER *pk_rver);
```

【パラメータ】

T_RVER *	pk_rver	バージョン情報を返すパケットへのポインタ
----------	---------	----------------------

【戻り値】

ER	ercd	正常終了 (E_OK) またはエラーコード
pk_rver の内容 (T_RVER 型)		
UH	maker	カーネルのメーカコード
UH	prid	カーネルの識別番号
UH	spver	ITRON 仕様のバージョン番号
UH	prver	カーネルのバージョン番号
UH	prno[4]	カーネル製品の管理情報

【エラーコード】

特記すべきエラーはない

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みサービスルーチン	可

【解説】

使用しているカーネルのバージョン情報を参照し、pk_rver で指定されるパケットに返します。

【補足】

本マニュアル作成時点では、メーカコードを取得していません。そのため、メーカコードとして 0x000 を返します。

5. 10 独自機能

5. 10. 1 デバイスドライバ管理機能

vdef_dev デバイスドライバの定義

【書式】

```
ER ercd = vdef_dev(ID devid, T_CDEV * pk_cdev);
```

【パラメータ】

ID	devid	定義対象のデバイスドライバの ID 番号
T_CDEV*	pk_cdev	デバイスドライバの生成情報を入れたパケットへのポインタ

pk_cdev の内容 (T_CDEV 型)

VP	ctrblk	制御情報パケットの先頭番地
FP	devhdr	デバイスドライバの起動番地
VB const *	name	デバイスドライバの名称 (文字列)

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (devid が不正あるいは使用できない)
E_CTX	コンテキストエラー (タスクと初期化ハンドラ以外からの呼出し)
E_NOMEM	メモリ不足 (管理領域が確保できない)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	不可
割込みサービスルーチン	不可

【解説】

devid で指定される ID 番号を持つデバイスドライバを、pk_cdev で指定されるデバイスドライバ生成情報に基づいて生成します。ctrblk はデバイスドライバの制御情報、devhdr はデバイスドライバの起動番地です。

このシステムコールは、標準 COM ドライバのために実装され、ユーザに使用を強制する機能ではありません。

vctr_dev**デバイスドライバの制御****【書式】**

```
ER ercd = vctr_dev(ID devid, ID funcid, VP ctrdev);
```

【パラメータ】

ID	devid	デバイスドライバの ID 番号
ID	funcid	デバイスドライバの機能コード
VP	ctrdev	デバイス制御情報パケットの先頭番地

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (devid が不正あるいは使用できない)
E_CTX	コンテキストエラー (タスク以外からの呼出し)
E_NOEXS	オブジェクト未生成 (デバイスドライバが未登録)

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	不可
タイムイベントハンドラ	不可
割込みサービスルーチン	不可

【解説】

devid で指定される ID 番号を持つデバイスドライバを起動します。funcid はデバイスドライバの機能コード、ctrdev はデバイス制御情報パケットです。

デバイスドライバには、第 1 引数には機能コード、第 2 引数にはデバイス制御情報パケットの先頭番地、第 3 引数には制御情報パケットの先頭番地が渡されます。

このシステムコールは、標準 COM ドライバのために実装され、ユーザに使用を強制する機能ではありません。

5. 10. 2 エラーハンドラ

vdef_err エラーハンドラの定義

【書式】

```
ER ercd = vdef_err (ATR atr, FP errhdr);
```

【パラメータ】

ATR	erratr	エラーハンドラ属性
FP	errhdr	エラーハンドラの起動番地

【戻り値】

ER	ercd	正常終了 (E_OK)
----	------	-------------

【呼び出しコンテキスト】

タスク	可
初期化ハンドラ	可
タイムイベントハンドラ	可
割込みハンドラ	可

【解説】

エラーハンドラ属性 **erratr** を持つ、エラーハンドラ起動番地 **errhdr** のエラーハンドラを定義します。エラーハンドラの起動番地 **errhdr** に **NULL (=0)** が指定されると、すでに定義されているエラーハンドラの定義を解除します。

erratr には、(TA_HLNG) のみが指定できます。

5. 1 1 注意事項

5. 1 1. 1 dly_tsk とタイムアウトありシステムコール txxx_yyy の差異

dly_tsk に関しては μ ITRON4.0 仕様で、txxx_yyy に関しては μ ITRON3.0 仕様になっています。

dly_tsk について、μ ITRON4.0 仕様書では、

「時間経過待ち状態からの解除は、このサービスコールが呼び出されてから、dlytim で指定された以上の時間が経過した後に行うことを保証しなければならない(2.1.9節参照)。このサービスコールは、dlytim に 0 が指定された場合にも、自タスクを待ち状態に移行させる。」

と説明があるため、μ C3/Standard では次の挙動となります。

dlytim に 0 を指定すると、次回(1回目)のチック割込みまで待ち状態になります。同様に dlytim に T を指定すると、(T+1)回目のチック割込みまで待ち状態になります。そのため、遅延時間は「指定時間+チック時間」未満になりますので、dly_tsk(1)とした場合には、

$$1\text{msec} \leq \text{遅延時間} < 2\text{msec}$$

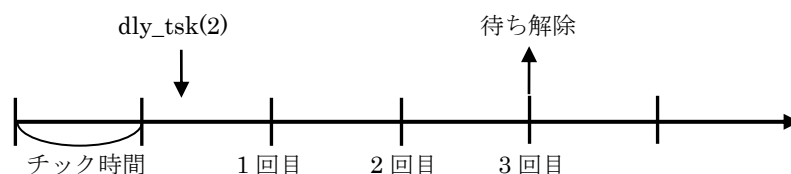
となります。(チック時間の設定を 1 とした場合)

なお、txxx_yyy でのタイムアウト時間は μ ITRON3.0 仕様となっているため、0 を指定するとポーリングになり、1 指定で次回(1回目)のチック割込みまで待ってタイムアウトになります。そのため、tslp_tsk(1)で呼び出した場合は、

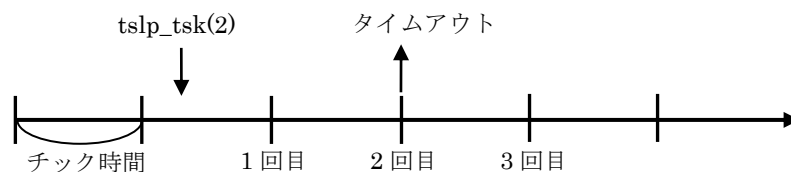
$$0\text{msec} < \text{タイムアウト時間} < 1\text{msec}$$

となります。(1msec 以上の時間が経過したことを保証しません)

dly_tsk(T)と同等のタイムアウト時間を実現する場合には、txxx_yyy(,T+1) として呼び出す必要があります。



(a) T=2 とした場合、dly_tsk(2)の遅延時間



(b) T=2 とした場合、tslp_tsk(2)のタイムアウト時間

遅延時間とタイムアウト時間の比較図

第6章 標準 COM ポートドライバの説明

6. 1 標準 COM ポートドライバの概要

μ C3/Standard 上で COM ポートを使用する際の使用方法を規定し、そのドライバを標準 COM ポートドライバと呼びます。ここでは、標準 COM ポートドライバのサービスコールを説明します。

サービスコールの呼び出しは、タスクコンテキストからのみに対応し、ディスパッチ保留状態で使うことはできません。

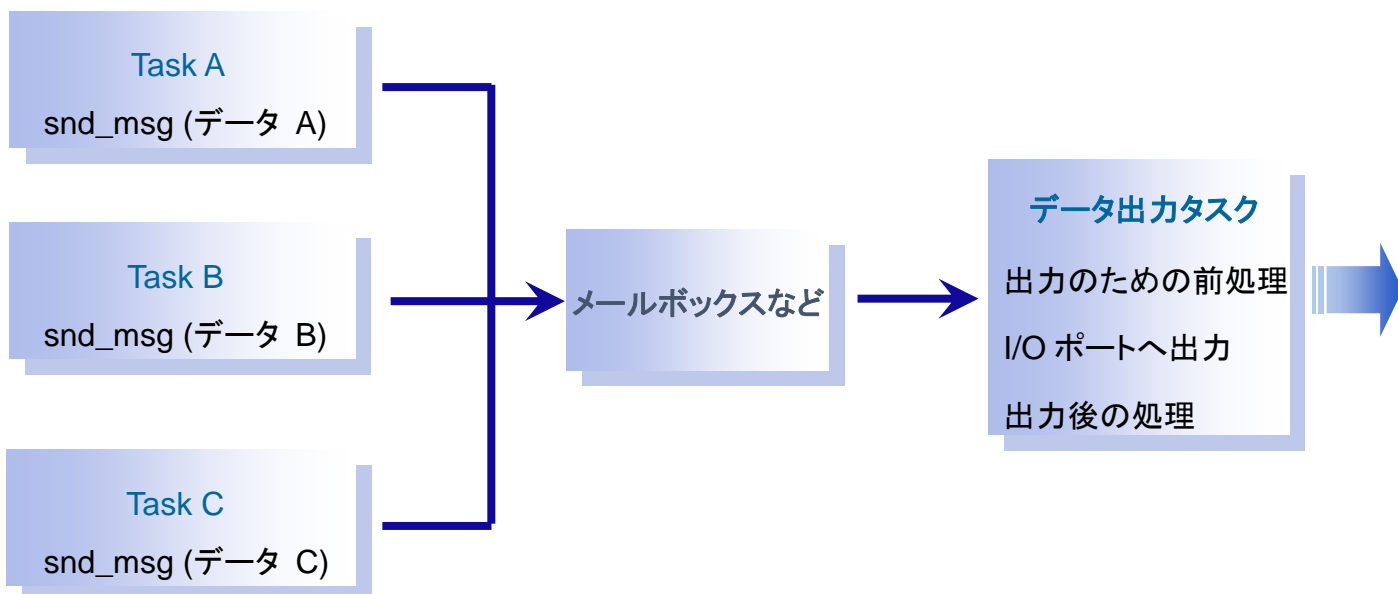
【注意事項】

標準 COM ポートドライバを利用した、送信・受信については、複数のタスクからの呼び出しを実施することは可能ですが、その場合、同時に呼び出しを実施しないようにセマフォ等を利用し、アプリケーション側で排他制御をすることが前提(必要)となります。

例えば、複数のアプリケーションから、送信処理(puts_com)を実施する場合には、アプリケーション側で、送信処理に対して、セマフォ等を利用することで、排他制御をお願いします。

また、排他の方法としては、下記の概要図のように、セマフォ以外に、シリアル出力を実施するタスクを一つにし、複数のタスクから、出力するタスクに送信するデータを渡す例などもありますので、参考にしてください。

I/O や資源などにアクセスするタスクを 1 つにし、メールボックスやデータキューでデータを送信することで排他制御にもなる



なお、関連する情報として、ctr_com サービスコールの制御コマンドを LOC_TX または、LOC_RX として呼び出すことで、ctr_com を呼び出したタスク以外から、標準 COM ポートドライバの送信または、受信のサービスコールを呼び出した場合には、送受信処理は実施されないことができます(この場合には、E_OBJ が返ります)。

6. 2 標準 COM ポートドライバのサービスコール

ini_com

COM ポートの初期化

【書式】

```
ER ercd = ini_com(ID DevID, T_COM_SMOD const * pk_SerialMode);
```

【パラメータ】

ID	DevID	デバイスの ID 番号
T_COM_SMOD const *	pk_SerialMode	初期化情報パケットへのポインタ
pk_SerialMode の内容 (T_COM_SMOD 型)		
UW	baud	ボーレート
UB	blen	データビット
UB	par	パリティ
UB	sbit	ストップビット
UB	flow	フロー制御

【戻り値】

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

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号 (DevID が不正、あるいは使用できない)

【解説】

DevID で指定される ID 番号のデバイスを、初期化情報パケットの内容で初期化します。

baud には、シリアルデバイスのボーレートを指定します。

blen には、データビットを次のいずれかで指定します。

BLLEN8	8 ビットデータ長
BLLEN7	7 ビットデータ長
BLLEN6	6 ビットデータ長
BLLEN5	5 ビットデータ長

par には、パリティビットを次のいずれかで指定します。

PAR_NONE	パリティビット無効
PAR_EVEN	愚数パリティビット有効
PAR_ODD	奇数パリティビット有効

sbit には、ストップビットを次のいずれかで指定します。

SBIT1	1 ビットストップ
SBIT15	1. 5 ビットデータ長
SBIT2	2 ビットデータ長

flow には、フロー制御を次のいずれかで指定します。

FLW_NONE	フロー制御無効
FLW_XON	ソフトウェアフロー制御有効
FLW_HARD	ハードウェアフロー制御有効

ctr_com**COM ポートの制御****【書式】**

ER ercd = ctr_com (ID DevID, UH command, TMO tmout) ;

【パラメータ】

ID	DevID	デバイスの ID 番号
UH	command	制御コマンド
TMO	tmout	タイムアウト指定

【戻り値】

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

【エラーコード】

E_PAR	パラメータエラー
E_ID	不正 ID 番号 (DevID が不正、あるいは使用できない)
E_TMOUT	ポーリング失敗またはタイムアウト

【解説】

DevID で指定される ID 番号のデバイスを、command で指定された内容の制御を行います。

Command には次の種類があり、論理和 (OR) で複数コマンドを指定できます。この場合には、上のコマンドから順次行います。

RST_COM	0xF800	COM ポートのリセット
CLN_TXBUF	0x8000	送信バッファの送出待ち
RST_BUF	0x6000	送受信バッファのクリア
RST_TXBUF	0x4000	送信バッファのクリア
RST_RXBUF	0x2000	受信バッファのクリア
STP_COM	0x1800	送受信の禁止
STP_TX	0x1000	送信の禁止
STP_RX	0x0800	受信の禁止
SND_BRK	0x0400	ブレークキャラクタの送出
STA_COM	0x0300	送受信の許可
STA_TX	0x0200	送信の許可
STA_RX	0x0100	受信の許可
LOC_TX	0x0080	送信のロック
LOC_RX	0x0040	受信のロック
UNL_TX	0x0020	送信のロック解除
UNL_RX	0x0010	受信のロック解除

tmout は、CLN_TXBUF の場合にはタイムアウト時間を、SND_BRK の場合には送出時間を指定します。その他の場合には、無視されます。

putc_com**COM ポートへの一文字送信****【書式】**

```
ER ercd = putc_com (ID DevID, VB chr, TMO tmout);
```

【パラメータ】

ID	DevID	デバイスの ID 番号
VB	chr	送信文字
TMO	tmout	タイムアウト指定

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (DevID が不正、あるいは使用できない)
E_TMOUT	ポーリング失敗またはタイムアウト

【解説】

DevID で指定される ID 番号のデバイスから、送信文字 chr を送信します。
tmout は、送信完了までのタイムアウト時間を指定します。

puts_com

COM ポートへの文字列送信

【書式】

```
ER ercd = puts_com (ID DevID, VB const *p_schr, UINT *p_scnt, TMO tmout);
```

【パラメータ】

ID	DevID	デバイスの ID 番号
VB const *	schr	送信文字列
UINT *	scnt	送信文字数
TMO	tmout	タイムアウト指定

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (DevID が不正、あるいは使用できない)
E_TMOUT	ポーリング失敗またはタイムアウト

【解説】

DevID で指定される ID 番号のデバイスから、送信文字列 **schr** を送信文字数 **scnt** だけ送信します。

tmout は、送信完了までのタイムアウト時間を指定します。

getc_com**COM ポートからの一文字受信****【書式】**

```
ER ercd = getc_com(ID DevID, VB *p_rbuf, UB *p_sbuf, TMO tmout);
```

【パラメータ】

ID	DevID	デバイスの ID 番号
VB *	rbuf	受信文字
UB *	sbuf	受信ステータス
TMO	tmout	タイムアウト指定

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (DevID が不正、あるいは使用できない)
E_TMOUT	ポーリング失敗またはタイムアウト

【解説】

DevID で指定される ID 番号のデバイスから、rbuf に受信した文字を返し、受信ステータスを sbuf に返します。この時、受信ステータスが不要であれば、p_sbuf に 0 を指定します。

tmout は、受信完了までのタイムアウト時間を指定します。

gets_com

COM ポートからの文字列受信

【書式】

```
ER ercd = gets_com(ID DevID, VB *p_rbuf, UB *p_sbuf, INT eos, UINT *p_rcnt,
                  TMO tmout);
```

【パラメータ】

ID	DevID	デバイスの ID 番号
VB *	rbuf	受信文字の配列列
UB *	sbuf	受信ステータスの配列
INT	eos	終端文字
UINT	rcnt	受信文字数
TMO	tmout	タイムアウト指定

【戻り値】

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

【エラーコード】

E_ID	不正 ID 番号 (DevID が不正、あるいは使用できない)
E_TMOUT	ポーリング失敗またはタイムアウト

【解説】

DevID で指定される ID 番号のデバイスから、rbuf に受信した文字を返し、受信ステータスを sbuf に返します。受信データの格納領域のサイズは、rcnt に指定し、受信した文字数は、rcnt に返します。この時、受信ステータスが不要であれば、p_sbuf に 0 を指定します。

受信は、格納領域が満杯になったか、終端文字を受信したか、受信ステータス有効の場合にエラーが発生したか、いずれかの場合に正常終了します。

tmout は、受信完了までのタイムアウト時間を指定します。

ref_com**COM ポートの状態参照****【書式】**

```
ER ercd = ref_com( ID tskid, T_COM_REF *pk_SerialRef );
```

【パラメータ】

ID	DevID	デバイスの ID 番号
T_COM_REF *	pk_SerialRef	COM ポートの状態

【戻り値】

ER	ercd	正常終了 (E_OK) またはエラーコード
pk_SerialRef の内容 (T_COM_REF 型)		
UH	rxcnt	受信済み文字数
UH	txcnti	未送信文字数
UH	status	ステータス

【エラーコード】

E_ID	不正 ID 番号 (tskid が不正、あるいは使用できない)
------	---------------------------------

【解説】

DevID で指定される ID 番号のデバイスに関する状態を参照し、pk_SerialRef で指定されるパケットに返します。

rxcnt に、ドライバ内の受信済み文字数を、txcnt に未送信文字数を返します。

status には、状態により、次の状態が論理和 (OR) した値で返します。

T_COM_EROVB	0x0001	FIFO オーバラン
T_COM_EROR	0x0002	オーバランエラー
T_COM_ERP	0x0004	パリティエラー
T_COM_ERF	0x0008	フレーミングエラー
T_COM_BRK	0x0010	ブレークキャラクタ受信
T_COM_TXOFF	0x0020	送信 XOFF 受信
T_COM_RXOFF	0x0040	受信 XOFF 送信
T_COM_RTS	0x0080	RTS 信号アクティブ
T_COM_CTS	0x0100	CTS 信号アクティブ
T_COM_DTR	0x0200	DTR 信号アクティブ
T_COM_DSR	0x0400	DSR 信号アクティブ
T_COM_CD	0x0800	CD 信号アクティブ
T_COM_RI	0x1000	RI 信号アクティブ
T_COM_ENARX	0x2000	受信許可状態
T_COM_ENATX	0x4000	送信許可状態
T_COM_INIT	0x8000	COM ポート初期化済み

第7章 付録

7. 1 データ型

μ ITRON4.0 仕様で規定しているデータ型は次の通りです。（パケットの為のデータ型を除く）

B	符号付き 8 ビット整数
H	符号付き 16 ビット整数
W	符号付き 32 ビット整数
UB	符号無し 8 ビット整数
UH	符号無し 16 ビット整数
UW	符号無し 32 ビット整数
VB	データタイプが定まらない 8 ビットの値
VH	データタイプが定まらない 16 ビットの値
VW	データタイプが定まらない 32 ビットの値
VP	データタイプが定まらないものへのポインタ
FP	プログラムの起動番地（ポインタ）
INT	プロセッサに自然なサイズの符号付き整数
UINT	プロセッサに自然なサイズの符号無し整数
BOOL	真偽値（TRUE または FALSE）
FN	機能コード（符号付き整数）
ER	エラーコード（符号付き整数）
ID	オブジェクトの ID 番号（符号付き整数）
ATR	オブジェクト属性（符号無し整数）
STAT	オブジェクトの状態（符号無し整数）
MODE	サービスコールの動作モード（符号無し整数）
PRI	優先度（符号付き整数）
SIZE	メモリ領域のサイズ（符号無し整数）
TMO	タイムアウト指定（符号付き整数，時間単位は 1 ミリ秒）
RELTIM	相対時間（符号無し整数，時間単位は 1 ミリ秒）

SYSTM	システム時刻（符号無し整数，時間単位は1ミリ秒）
VP_INT	データタイプが定まらないものへのポインタまたはプロセッサに自然なサイズの符号付き整数
ER_BOOL	エラーコードまたは真偽値
ER_ID	エラーコードまたはID番号（負のID番号は表現できない）
ER_UINT	エラーコードまたは符号無し整数（符号無し整数の有効ビット数はUINTより1ビット短い）
TEXPTN	タスク例外要因のビットパターン（符号無し整数）
FLGPTN	イベントフラグのビットパターン（符号無し整数）
T_MSG	メールボックスへのメッセージヘッダ
T_MSG_PRI	メールボックスの優先度付きメッセージヘッダ
RDVPTN	ランデブ条件のビットパターン（符号無し整数）
RDVNO	ランデブ番号
OVRTIM	プロセッサ時間（符号無し整数，時間単位はユーザ定義）
INHNO	割込みハンドラ番号
INTNO	割込み番号
IMASK	割込みマスク
EXCNO	CPU 例外ハンドラ番号

【補足】

INT, UINT, VP_INT, TEXPTN, FLGPTN は、プロセッサに依存し、これらの詳細は「プロセッサ依存部実装マニュアル」を参照してください。

7. 2 パケット形式

(1) タスク管理機能

タスク生成情報のパケット形式

```
typedef struct t_ctsk {
    ATR      tskatr;      /* タスク属性 */
    VP_INT   exinf;      /* タスクの拡張情報 */
    FP       task;        /* タスクの起動番地 */
    PRI      itskpri;     /* タスクの起動時優先度 */
    SIZE     stksz;       /* タスクのスタックサイズ (バイト数) */
    VP       stk;         /* タスクのスタック領域の先頭番地 */
    VB const * name;      /* タスクの名称 */
} T_CTSK;
```

タスク状態のパケット形式

```
typedef struct t_rtsk {
    STAT     tskstat;     /* タスク状態 */
    PRI      tskpri;      /* タスクの現在優先度 */
    PRI      tsbpri;      /* タスクのベース優先度 */
    STAT     tsawait;     /* 待ち要因 */
    ID       wobjid;      /* 待ち対象のオブジェクトの ID 番号 */
    TMO      lefttmo;     /* タイムアウトするまでの時間 */
    UINT     actcnt;      /* 起動要求キューイング数 */
    UINT     wupcnt;      /* 起床要求キューイング数 */
    UINT     suscnt;      /* 強制待ち要求ネスト数 */
} T_RTSK;
```

タスク状態（簡易版）のパケット形式

```
typedef struct t_rtst {
    STAT     tskstat;     /* タスク状態 */
    STAT     tsawait;     /* 待ち要因 */
} T_RTST;
```

(2) タスク例外処理機能

タスク例外処理ルーチン定義情報のパケット形式

```
typedef struct t_dtex {
    ATR          texatr;      /* タスク例外処理ルーチン属性 */
    FP           texrtn;      /* タスク例外処理ルーチンの起動番地 */
} T_DTEX;
```

タスク例外処理状態のパケット形式

```
typedef struct t_rtex {
    STAT         texstat;     /* タスク例外処理の状態 */
    TEXPTN       pndptn;     /* 保留例外要因 */
} T_RTEX;
```

(3) 同期・通信機能

セマフォ生成情報のパケット形式

```
typedef struct t_csem {
    ATR          sematr;      /* セマフォ属性 */
    UINT         isemcnt;     /* セマフォの資源数の初期値 */
    UINT         maxsem;     /* セマフォの最大資源数 */
    VB const *   name;       /* セマフォの名称 */
} T_CSEM;
```

セマフォ状態のパケット形式

```
typedef struct t_rsem {
    ID           wtskid;      /* セマフォの待ち行列の先頭のタスク
                               の ID 番号 */
    UINT         semcnt;     /* セマフォの現在の資源数 */
} T_RSEM;
```

イベントフラグ生成情報のパケット形式

```
typedef struct t_cflg {
    ATR          flgatr;      /* イベントフラグ属性 */
    FLGPTN       iflgptn;    /* イベントフラグのビットパターンの
                               初期値 */
    VB const *   name;       /* イベントフラグの名称 */
} T_CFLG;
```

イベントフラグ状態のパケット形式

```
typedef struct t_rflg {
    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 */
    FLGPTN      flgptn;     /* イベントフラグの現在のビットパターン */
} T_RFLG;
```

データキュー生成情報のパケット形式

```
typedef struct t_cdtq {
    ATR          dtqatr;     /* データキュー属性 */
    UINT         dtqcnt;     /* データキュー領域の容量 (データの個数) */
    VP           dtq;        /* データキュー領域の先頭番地 */
    VB const *   name;       /* データキューの名称 */
} T_CDTQ;
```

データキュー状態のパケット形式

```
typedef struct t_rdtq {
    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         sdtqcnt;    /* データキューに入っているデータの数 */
} T_RDTQ;
```

メールボックス生成情報のパケット形式

```
typedef struct t_cmbx {
    ATR          mbxatr;     /* メールボックス属性 */
    PRI          maxmpri;    /* 送信されるメッセージの優先度の最大値 */
    VP           mprihd;     /* 優先度別のメッセージキューヘッダ
                             領域の先頭番地 */
    VB const *   name;       /* メールボックスの名称 */
} T_CMBX;
```

メールボックス状態のパケット形式

```
typedef struct t_rmbx {
    ID           wtskid;     /* 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のメッセージ
                             パケットの先頭番地 */
} T_RMBX;
```


(4) 拡張同期・通信機能

ミューテックス生成情報のパケット形式

```
typedef struct t_cmtx {
    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 */
    VB const * name;    /* ミューテックスの名称 */
} T_CMTX;
```

ミューテックス状態のパケット形式

```
typedef struct t_rmtx {
    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 */
} T_RMTX;
```

メッセージバッファ生成情報のパケット形式

```
typedef struct t_cmbf {
    ATR      mbfatr;    /* メッセージバッファ属性 */
    UINT      maxmsz;   /* メッセージの最大サイズ (バイト数) */
    SIZE      mbfsz;    /* メッセージバッファ領域のサイズ
                        (バイト数) */
    VP      mbf;        /* メッセージバッファ領域の先頭番地 */
    VB const * name;    /* メッセージバッファの名称 */
} T_CMBF;
```

メッセージバッファ状態のパケット形式

```
typedef struct t_rmbf {
    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      smsgcnt;   /* メッセージバッファに入っている
                        メッセージの数 */
    SIZE      fmbfsz;    /* メッセージバッファ領域の空き領域
                        のサイズ (バイト数, 最低限の管理
                        領域を除く) */
} T_RMBF;
```

ランデブポート生成情報のパケット形式

```
typedef struct t_cpor {
    ATR      poratr;      /* ランデブポート属性 */
    UINT     maxcmsz;     /* 呼出しメッセージの最大サイズ
                           (バイト数) */
    UINT     maxrmsz;     /* 返答メッセージの最大サイズ (バイト数) */
    VB const * name;      /* ランデブポートの名称 */
} T_CPOR;
```

ランデブポート状態のパケット形式

```
typedef struct t_rpor {
    ID       ctskid;      /* ランデブポートの呼出し待ち行列の
                           先頭のタスクの ID 番号 */
    ID       atskid;      /* ランデブポートの受付待ち行列の先
                           頭のタスクの ID 番号 */
} T_RPOR;
```

ランデブ状態のパケット形式

```
typedef struct t_rrdv {
    ID       wtskid;      /* ランデブ終了待ち状態のタスクの ID 番号 */
} T_RRDV;
```

(5) メモリプール管理機能

固定長メモリプール生成情報のパケット形式

```
typedef struct t_cmpf {
    ATR      mpfatr;      /* 固定長メモリプール属性 */
    UINT     blkcnt;      /* 獲得できるメモリブロック数 (個数) */
    UINT     blkksz;      /* メモリブロックのサイズ (バイト数) */
    VP       mpf;         /* 固定長メモリプール領域の先頭番地 */
    VB const * name;      /* 固定長メモリプールの名称 */
} T_CMPF;
```

固定長メモリプール状態のパケット形式

```
typedef struct t_rmpf {
    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     fblkcnt;     /* 固定長メモリプールの空きメモリブ
                           ロック数 (個数) */
} T_RMPF;
```

可変長メモリプール生成情報のパケット形式

```
typedef struct t_cmpl {
    ATR      mplatr;    /* 可変長メモリプール属性 */
    SIZE      mplsz;    /* 可変長メモリプール領域のサイズ
                        (バイト数) */
    VP        mpl;      /* 可変長メモリプール領域の先頭番地 */
    VB const * name;    /* 可変長メモリプールの名称 */
} T_CMPL;
```

可変長メモリプール状態のパケット形式

```
typedef struct t_rmpl {
    ID        wtskid;   /* 可変長メモリプールの待ち行列の先
                        頭のタスクの ID 番号 */
    SIZE      fmplsz;   /* 可変長メモリプールの空き領域の合
                        計サイズ (バイト数) */
    UINT      fblksz;   /* すぐに獲得可能な最大メモリブロッ
                        クサイズ (バイト数) */
} T_RMPL;
```

(6) 時間管理機能

周期ハンドラ生成情報のパケット形式

```
typedef struct t_ccyc {
    ATR      cycatr;    /* 周期ハンドラ属性 */
    VP_INT   exinf;     /* 周期ハンドラの拡張情報 */
    FP        cychdr;   /* 周期ハンドラの起動番地 */
    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 */
    VB const * name;    /* 周期ハンドラの名称 */
} T_CCYC;
```

周期ハンドラ状態のパケット形式

```
typedef struct t_rcyc {
    STAT      cycstat;  /* 周期ハンドラの動作状態 */
    RELTIM    lefttim;  /* 周期ハンドラを次に起動すべき時刻
                        までの時間 */
} T_RCYC;
```

アラームハンドラ生成情報のパケット形式

```
typedef struct t_calm {
    ATR      almatr;    /* アラームハンドラ属性 */
    VP_INT   exinf;     /* アラームハンドラの拡張情報 */
    FP       almhdr;    /* アラームハンドラの起動番地 */
    VB const * name;    /* アラームハンドラの名称 */
} T_CALM;
```

アラームハンドラ状態のパケット形式

```
typedef struct t_ralm {
    STAT      almstat;  /* アラームハンドラの動作状態 */
    RELTIM    lefttim;  /* アラームハンドラの起動時刻までの時間 */
} T_RALM;
```

オーバランハンドラ定義情報のパケット形式

```
typedef struct t_dovr {
    ATR      ovratr;    /* オーバランハンドラ属性 */
    FP       ovrrhdr;   /* オーバランハンドラの起動番地 */
} T_DOVR;
```

オーバランハンドラ状態のパケット形式

```
typedef struct t_rovr {
    STAT      ovrrstat; /* オーバランハンドラの動作状態 */
    OVRTIM    leftotm;  /* 残りのプロセッサ時間 */
} T_ROVR;
```

(7) システム状態管理機能

システム状態のパケット形式

```
typedef struct t_rsys {
    SIZE      fsyssz;    /* システムメモリの空き領域 (バイト数) */
    SIZE      fstksiz;   /* スタック用メモリの空き領域 (バイト数) */
    SIZE      fmplsz;    /* メモリプール用メモリの空き領域 (バイト数) */
    UH        utskid;    /* 生成済みタスク ID の個数 */
    UH        usemid;    /* 生成済みセマフォ ID の個数 */
    UH        uflgid;    /* 生成済みイベントフラグ ID の個数 */
    UH        udtqid;    /* 生成済みデータキューID の個数 */
    UH        umbxid;    /* 生成済みメールボックス ID の個数 */
}
```

```

UH      umtxid;    /* 生成済みミューテックス ID の個数*/
UH      umbfid;    /* 生成済みメッセージバッファ ID の個数*/
UH      uporid;    /* 生成済みランデブポート ID の個数*/
UH      umpfid;    /* 生成済み固定長メモリプール ID の個数*/
UH      umplid;    /* 生成済み可変長メモリプール ID の個数*/
UH      ualmid;    /* 生成済みアラームハンドラ ID の個数*/
UH      ucycid;    /* 生成済み周期ハンドラ ID の個数*/
UH      uisrid;    /* 生成済み割込みサービスルーチン ID の
UH              個数*/
UH      ssbcnt;    /* SSB が最少になった時の個数*/
} T_RSYS;

```

(8) 割込み管理機能

割込みハンドラ定義情報のパケット形式

```

typedef struct t_dinh {
    ATR      inhatr;    /* 割込みハンドラ属性 */
    FP      inthdr;    /* 割込みハンドラの起動番地 */
    IMASK    imask;    /* 割込みハンドラの割込みマスクレベル */
} T_DINH;

```

割込みサービスルーチン生成情報のパケット形式

```

typedef struct t_cisr {
    ATR      isratr;    /* 割込みサービスルーチン属性 */
    VP_INT   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付加する
                        割込み番号 */
    FP      isr;    /* 割込みサービスルーチンの起動番地 */
    IMASK    imask;    /* 割込みハンドラの割込みマスクレベル */
} T_CISR;

```

割込みサービスルーチン状態のパケット形式

```

typedef struct t_risr {
    INTNO    intno;    /* 割込みサービスルーチンを付加する
                        割込み番号 */
    FP      isr;    /* 割込みサービスルーチンの起動番地 */
} T_RISR;

```

(9) システム構成管理機能

コンフィグレーション情報のパケット形式

```
typedef struct t_rcfg {
    UH      tskpri_max;   タスク優先度の上限
    UH      tskid_max;    タスク ID の上限
    UH      semid_max;    セマフォ ID の上限
    UH      flgid_max;    イベントフラグ ID の上限
    UH      dtqid_max;    データキューID の上限
    UH      mbxid_max;    メールボックス ID の上限
    UH      mtxid_max;    ミューテックス ID の上限
    UH      mbfid_max;    メッセージバッファ ID の上限
    UH      porid_max;    ランデブポート ID の上限
    UH      mpfid_max;    固定長メモリプール ID の上限
    UH      mplid_max;    可変長メモリプール ID の上限
    UH      almid_max;    アラームハンドラ ID の上限
    UH      cycid_max;    周期ハンドラ ID の上限
    UH      isrid_max;    割込みサービスルーチン ID の上限
    UH      devid_max;    デバイスドライバ ID の上限
    UH      tick;         チック時間
    UH      ssb_cnt;      システムサービスブロックの生成個数
} T_RCFG;
```

バージョン情報のパケット形式

```
typedef struct t_rver {
    UH      maker;        /* カーネルのメーカーコード */
    UH      prid;         /* カーネルの識別番号 */
    UH      spver;        /* ITRON 仕様のバージョン番号 */
    UH      prver;        /* カーネルのバージョン番号 */
    UH      prno[4];      /* カーネル製品の管理情報 */
} T_RVER;
```

7. 3 定数とマクロ

(1) 一般

NULL	0	無効ポインタ
TRUE	1	真
FALSE	0	偽
E_OK	0	正常終了

(2) オブジェクト属性

TA_NULL	0	オブジェクト属性を指定しない
TA_HLNG	0x00	高級言語用のインタフェースで処理単位を起動
TA_ASM	0x01	アセンブリ言語用のインタフェースで処理単位を起動
TA_TFIFO	0x00	タスクの待ち行列を FIFO 順に
TA_TPRI	0x01	タスクの待ち行列をタスクの優先度順に
TA_MFIFO	0x00	メッセージのキューを FIFO 順に
TA_MPRI	0x02	メッセージのキューをメッセージの優先度順に
TA_ACT	0x02	タスクを起動された状態で生成
TA_RSTR	0x04	制約タスク
TA_WSGL	0x00	イベントフラグを複数のタスクが待つことを許さない
TA_WMUL	0x02	イベントフラグを複数のタスクが待つことを許す
TA_CLR	0x04	待ち解除時にイベントフラグをクリア
TA_INHERIT	0x02	ミューテックスが優先度継承プロトコルをサポート
TA_CEILING	0x03	ミューテックスが優先度上限プロトコルをサポート
TA_STA	0x02	周期ハンドラを動作している状態で生成
TA_PHS	0x04	周期ハンドラの位相を保存

(3) タイムアウト指定

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

(4) サービスコールの動作モード

TWF_ANDW	0x00	イベントフラグの AND 待ち
TWF_ORW	0x01	イベントフラグの OR 待ち

(5) オブジェクトの状態

TTS_RUN	0x01	実行状態
TTS_RDY	0x02	実行可能状態
TTS_WAI	0x04	待ち状態
TTS_SUS	0x08	強制待ち状態
TTS_WAS	0x0c	二重待ち状態
TTS_DMT	0x10	休止状態
TTW_SLP	0x0001	起床待ち状態
TTW_DLY	0x0002	時間経過待ち状態
TTW_SEM	0x0004	セマフォ資源の獲得待ち状態
TTW_FLG	0x0008	イベントフラグ待ち状態
TTW_SDTQ	0x0010	データキューへの送信待ち状態
TTW_RDTQ	0x0020	データキューからの受信待ち状態
TTW_MBX	0x0040	メールボックスからの受信待ち状態
TTW_MTX	0x0080	ミューテックスのロック待ち状態
TTW_SMBF	0x0100	メッセージバッファへの送信待ち状態
TTW_RMBF	0x0200	メッセージバッファからの受信待ち状態
TTW_CAL	0x0400	ランデブの呼出し待ち状態
TTW_ACP	0x0800	ランデブの受付待ち状態
TTW_RDV	0x1000	ランデブの終了待ち状態
TTW_MPF	0x2000	固定長メモリブロックの獲得待ち状態
TTW_MPL	0x4000	可変長メモリブロックの獲得待ち状態
TTEX_ENA	0x00	タスク例外処理許可状態
TTEX_DIS	0x01	タスク例外処理禁止状態
TCYC_STP	0x00	周期ハンドラが動作していない
TCYC_STA	0x01	周期ハンドラが動作している
TALM_STP	0x00	アラームハンドラが動作していない
TALM_STA	0x01	アラームハンドラが動作している
TOVR_STP	0x00	上限プロセッサ時間が設定されていない
TOVR_STA	0x01	上限プロセッサ時間が設定されている

(6) その他の定数

TSK_SELF	0	自タスク指定
TSK_NONE	0	該当するタスクが無い
TPRI_SELF	0	自タスクのベース優先度の指定
TPRI_INI	0	タスクの起動時優先度の指定

7. 4 構成定数とマクロ

(1) 優先度の範囲

TMIN_TPRI	タスク優先度の最小値 (=1)
TMAX_TPRI	タスク優先度の最大値 (=31)
TMIN_MPRI	メッセージ優先度の最小値 (=1)
TMAX_MPRI	メッセージ優先度の最大値 (=31)

(2) バージョン情報

TKERNEL_MAKER	カーネルのメーカーコード
TKERNEL_PRID	カーネルの識別番号
TKERNEL_SPVER	ITRON 仕様のバージョン番号
TKERNEL_PRVER	カーネルのバージョン番号

(3) キューイング／ネスト回数の最大値

TMAX_ACTCNT	タスクの起動要求キューイング数の最大値 (=999)
TMAX_WUPCNT	タスクの起床要求キューイング数の最大値 (=999)
TMAX_SUSCNT	タスクの強制待ち要求ネスト数の最大値 (=999)

(4) ビットパターンのビット数

TBIT_TEXPTN	タスク例外要因のビット数
TBIT_FLGPTN	イベントフラグのビット数
TBIT_RDVPTN	ランデブ条件のビット数

(5) 必要なメモリ領域のサイズ

SIZE dtqsz = TSZ_DTQ (UINT dtqcnt)

dtqcnt 個のデータを格納するのに必要なデータキュー領域のサイズ (バイト数)

SIZE mprihdsz = TSZ_MPRIHD (PRI maxmpri)

送信されるメッセージの優先度の最大値が maxmpri のメールボックスに必要な優先度別メッセージキューヘッダ領域のサイズ (バイト数)

SIZE mbfsz = TSZ_MBF (UINT msgcnt, UINT msgsz)

サイズが msgsz バイトのメッセージを msgcnt 個バッファリングするのに必要なメッセージバッファ領域のサイズ (目安のバイト数)

SIZE mpfsz = TSZ_MPF (UINT blkcnt, UINT blkksz)

サイズが blkksz バイトのメモリブロックを blkcnt 個獲得するのに必要な固定長メモリプール領域のサイズ (バイト数)

SIZE mplsz = TSZ_MPL (UINT blkcnt, UINT blkksz)

サイズが blkksz バイトのメモリブロックを blkcnt 個獲得するのに必要な可変長メモリプール領域のサイズ (目安のバイト数)

(6) その他

TMAX_MAXSEM セマフォの最大資源数の最大値 (=999)

7. 5 エラーコード一覧

E_SYS	-5	0xFFFFFFFFB	システムエラー
E_NOSPT	-9	0xFFFFFFFF7	未サポート機能
E_RSFN	-10	0xFFFFFFFF6	予約機能コード
E_RSATR	-11	0xFFFFFFFF5	予約属性
E_PAR	-17	0xFFFFFFFFEF	パラメータエラー
E_ID	-18	0xFFFFFFFFEE	不正 ID 番号
E_CTX	-25	0xFFFFFFFFE7	コンテキストエラー
E_MACV	-26	0xFFFFFFFFE6	メモリアクセス違反
E_OACV	-27	0xFFFFFFFFE5	オブジェクトアクセス違反
E_ILUSE	-28	0xFFFFFFFFE4	サービスコール不正使用
E_NOMEM	-33	0xFFFFFFFFDF	メモリ不足
E_NOID	-34	0xFFFFFFFFDE	ID 番号不足
E_OBJ	-41	0xFFFFFFFFD7	オブジェクト状態エラー
E_NOEXS	-42	0xFFFFFFFFD6	オブジェクト未生成
E_QOVR	-43	0xFFFFFFFFD5	キューイングオーバーフロー
E_RLWAI	-49	0xFFFFFFFFCF	待ち状態の強制解除
E_TMOUT	-50	0xFFFFFFFFCE	ポーリング失敗またはタイムアウト
E_DLT	-51	0xFFFFFFFFCD	待ちオブジェクトの削除
E_CLS	-52	0xFFFFFFFFCC	待ちオブジェクトの状態変化
E_WBLK	-57	0xFFFFFFFFC7	ノンブロッキング受付
E_BOVR	-58	0xFFFFFFFFC6	バッファオーバーフロー

7. 6 システムコール一覧

システムコール名	タスク	初期化 ハンドラ	タイム イベント ハンドラ	割込み サービス ルーチン
A) タスク管理機能				
cre_tsk / acre_tsk	○	○	×	×
del_tsk	○	×	×	×
act_tsk / iact_tsk	○	○	○	○
can_act	○	○	○	×
sta_tsk	○	○	○	○
ext_tsk	○	×	×	×
exd_tsk	○	×	×	×
ter_tsk	○	×	×	×
chg_pri	○	○	○	×
get_pri	○	○	○	×
ref_tsk	○	○	○	×
ref_tst	○	○	○	×
B) タスク付属同期				
slp_tsk	○	×	×	×
tslp_tsk	○	×	×	×
wup_tsk / iwup_tsk	○	○	○	○
can_wup	○	○	○	×
rel_wai / irel_wai	○	○	○	○
sus_tsk	○	○	○	×
rsm_tsk	○	○	○	×
frsm_tsk	○	○	○	×
dly_tsk	○	×	×	×

システムコール名	タスク	初期化 ハンドラ	タイム イベント ハンドラ	割込み サービス ルーチン
C) タスク例外処理				
def_tex	×	×	×	×
ras_tex	×	×	×	×
iras_tex	×	×	×	×
dis_tex	×	×	×	×
ena_tex	×	×	×	×
sns_tex	×	×	×	×
ref_tex	×	×	×	×
D) 同期・通信 セマフォ				
cre_sem／acre_sem	○	○	×	×
del_sem	○	×	×	×
sig_sem／isig_sem	○	○	○	○
wai_sem	○	×	×	×
pol_sem	○	○	○	×
twai_sem	○	×	×	×
ref_sem	○	○	○	×
E) 同期・通信 イベントフラグ				
cre_flg／acre_flg	○	○	×	×
del_flg	○	×	×	×
set_flg／iset_flg	○	○	○	○
clr_flg	○	○	○	×
wai_flg	○	×	×	×
pol_flg	○	○	○	×
twai_flg	○	×	×	×
ref_flg	○	○	○	×

システムコール名	タスク	初期化 ハンドラ	タイム イベント ハンドラ	割込み サービス ルーチン
F) 同期・通信 データキュー				
cre_dtq／acre_dtq	○	○	×	×
del_dtq	○	×	×	×
snd_dtq	○	×	×	×
psnd_dtq／ipsnd_dtq	○	○	○	○
tsnd_dtq	○	×	×	×
fsnd_dtq／ifsnd_dtq	○	○	○	○
rcv_dtq	○	○	○	×
prcv_dtq	○	○	○	×
trcv_dtq	○	×	×	×
ref_dtq	○	○	○	×
G) 同期・通信 メールボックス				
cre_mbx／acre_mbx	○	○	×	×
del_mbx	○	×	×	×
snd_mbx	○	○	○	×
rcv_mbx	○	×	×	×
prcv_mbx	○	○	○	×
trcv_mbx	○	×	×	×
ref_mbx	○	○	○	×
H) 拡張同期・通信 ミューテック				
cre_mtx／acre_mtx	○	○	×	×
del_mtx	○	×	×	×
loc_mtx	○	×	×	×
ploc_mtx	○	×	×	×
tloc_mtx	○	×	×	×
unl_mtx	○	×	×	×
ref_mtx	○	○	○	×

システムコール名	タスク	初期化 ハンドラ	タイム イベント ハンドラ	割込み サービス ルーチン
I) 拡張同期・通信 メッセージバッファ				
cre_mbf/acre_mbf	○	○	×	×
del_mbf	○	×	×	×
snd_mbf	○	×	×	×
psnd_mbf	○	○	○	×
tsnd_mbf	○	×	×	×
rcv_mbf	○	×	×	×
prcv_mbf	○	○	○	×
trcv_mbf	○	×	×	×
ref_mbf	○	○	○	×
J) 拡張同期・通信 ランデブ				
cre_por/acre_por	○	○	×	×
del_por	○	×	×	×
cal_por	○	×	×	×
tcal_por	○	×	×	×
acp_por	○	×	×	×
pacp_por	○	×	×	×
tacp_por	○	×	×	×
fwd_por	○	×	×	×
rpl_rdv	○	×	×	×
ref_por	○	○	○	×
ref_rdv	○	○	○	×
K) メモリプール管理 固定長メモリプール				
cre_mpf/acre_mpf	○	○	×	×
del_mpf	○	×	×	×
get_mpf	○	×	×	×
pget_mpf	○	○	○	×
tget_mpf	○	×	×	×
rel_mpf	○	○	○	×
ref_mpf	○	○	○	×

システムコール名	タスク	初期化 ハンドラ	タイム イベント ハンドラ	割込み サービス ルーチン
L) メモリプール管理 可変長メモリプール				
cre_mpl/acre_mpl	○	○	×	×
del_mpl	○	×	×	×
get_mpl	○	×	×	×
pget_mpl	○	○	○	×
tget_mpl	○	×	×	×
rel_mpl	○	○	○	×
ref_mpl	○	○	○	×
M) 時間管理システム時刻管理				
set_tim	○	○	○	×
get_tim	○	○	○	×
isig_tim	×	×	×	○
N) 時間管理周期ハンドラ				
cre_cyc/acre_cyc	○	○	×	×
del_cyc	○	×	×	×
sta_cyc	○	○	○	×
stp_cyc	○	○	○	×
ref_cyc	○	○	○	×
O) 時間管理アラームハンドラ				
cre_alm/acre_alm	○	○	×	×
del_alm	○	×	×	×
sta_alm	○	○	○	×
stp_alm	○	○	○	×
ref_alm	○	○	○	×

システムコール名	タスク	初期化 ハンドラ	タイム イベント ハンドラ	割込み サービス ルーチン
P) 時間管理オーバランハンドラ				
def_ovr	○	○	×	×
ivsig_ovr	×	×	×	○
sta_ovr	○	○	○	×
stp_ovr	○	○	○	×
ref_ovr	○	○	○	×
Q) システム状態管理				
rot_rdq/irotd_rdq	○	○	○	○
get_tid/iget_tid	○	○	○	○
loc_cpu/iloc_cpu	○	○	○	○
unl_cpu/iunl_cpu	○	○	○	○
dis_dsp	○	×	×	×
ena_dsp	○	×	×	×
sns_ctx	○	○	○	○
sns_loc	○	○	○	○
sns_dsp	○	○	○	○
sns_dpn	○	○	○	○
ref_sys	○	○	○	×
R) 割込み管理				
def_inh	○	○	○	×
cre_isr/acre_isr	○	○	×	×
del_isr	○	×	×	×
ref_isr	○	○	○	×
dis_int	△	△	△	△
ena_int	△	△	△	△
chg_ims	○	○	○	○
get_ims	○	○	○	○
S) サービスコール管理機能				
def_svc	×	×	×	×
cal_svc	×	×	×	×

システムコール名	タスク	初期化 ハンドラ	タイム イベント ハンドラ	割込み サービス ルーチン
T) システム構成管理機能				
def_exc	△	△	△	△
ref_cfg	○	○	○	○
ref_ver	○	○	○	○
U) 独自機能・デバイスドライバ管理機能				
vdef_dev	○	○	×	×
vcctr_dev	○	×	×	×

○：使用可

×：使用不可

△：プロセッサ依存

索引

A

acp_por	127
acre_alm	157
acre_cyc	151
acre_dtq	89
acre_flg	81
acre_isr	180
acre_mbf	114
acre_mbx	99
acre_mpf	134
acre_mpl	141
acre_mtx	107
acre_por	122
acre_sem	74
acre_tsk	49
act_tsk	52

C

cal_por	125
can_act	53
can_wup	68
chg_ims	186
chg_pri	58
clr_flg	85
COM ポートドライバ	195
CPU ロック解除状態	16
CPU ロック状態	16
cre_alm	157
cre_cyc	151
cre_dtq	89
cre_flg	81
cre_isr	180

cre_mbf	114
cre_mbx	99
cre_mpf	134
cre_mpl	141
cre_mtx	107
cre_por	122
cre_sem	74
cre_tsk	49
ctr_com	198

D

def_exc	188
def_inh	179
def_ovr	163
del_alm	159
del_cyc	153
del_dtq	91
del_flg	83
del_isr	182
del_mbf	116
del_mbx	101
del_mpf	136
del_mpl	143
del_mtx	109
del_por	124
del_sem	76
del_tsk	51
dis_dsp	172
dis_int	184
dly_tsk	72

E

ena_dsp	173
---------------	-----

ena_int.....	185
exd_tsk.....	56
ext_tsk.....	55

F

frsm_tsk.....	71
fsnd_dtq.....	94
fwd_por.....	129

G

get_ims.....	187
get_mpf.....	137
get_mpl.....	144
get_pri.....	60
get_tid.....	169
get_tim.....	149
getc_com.....	201
gets_com.....	202

I

iact_tsk.....	52
ID 番号	10
ifsnd_dtq.....	94
iget_tid.....	169
iloc_cpu.....	170
ini_com.....	196
ipsnd_dtq.....	92
irel_wai.....	69
irotdtq.....	168
iset_flg.....	84
isig_sem.....	77
isig_tim.....	150
iunl_cpu.....	171

ivsig_ovr.....	164
iwup_tsk.....	66

L

loc_cpu.....	170
loc_mtx.....	110

P

pacp_por.....	127
pget_mpf.....	137
pget_mpl.....	144
ploc_mtx.....	110
pol_flg.....	86
pol_sem.....	78
prev_dtq.....	96
prev_mbf.....	119
prev_mbx.....	104
psnd_dtq.....	92
psnd_mbf.....	117
putc_com.....	199
puts_com.....	200

R

rev_dtq.....	96
rev_mbf.....	119
rev_mbx.....	104
ref_alm.....	162
ref_cfg.....	189
ref_com.....	203
ref_cyc.....	156
ref_dtq.....	98
ref_flg.....	88
ref_isr.....	183

ref_mbf.....	121
ref_mbx.....	106
ref_mpf.....	140
ref_mpl.....	147
ref_mtx.....	113
ref_ovr.....	167
ref_por.....	132
ref_rdv.....	133
ref_sem.....	80
ref_sys.....	178
ref_tsk.....	61
ref_tst.....	64
ref_ver.....	190
rel_mpf.....	139
rel_mpl.....	146
rel_wai.....	69
rot_rdq.....	168
rpl_rdv.....	131
rsm_tsk.....	71

S

set_flg.....	84
set_tim.....	148
sig_sem.....	77
slp_tsk.....	65
snd_dtq.....	92
snd_mbf.....	117
snd_mbx.....	102
sns_ctx.....	174
sns_dpn.....	177
sns_dsp.....	176
sns_loc.....	175
SSB.....	19
sta_alm.....	160
sta_cyc.....	154

sta_ovr.....	165
sta_tsk.....	54
start_uC3.....	45
stp_alm.....	161
stp_cyc.....	155
stp_ovr.....	166
sus_tsk.....	70

T

tacp_por.....	127
tcal_por.....	125
ter_tsk.....	57
tget_mpl.....	144
tget_pmf.....	137
tloc_mtx.....	110
trev_dtq.....	96
trev_mbf.....	119
trev_mbx.....	104
tslp_tsk.....	65
tsnd_dtq.....	92
tsnd_mbf.....	117
twai_flg.....	86
twai_sem.....	78

U

unl_cpu.....	171
unl_mtx.....	112

V

vctr_dev.....	192
vdef_dev.....	191
vdef_err.....	193

W

wai_flg.....	86
wai_sem.....	78
wup_tsk.....	66

あ

ID の上限値.....	20
アイドル状態.....	17
アラームハンドラ.....	38

い

イベントフラグ.....	26
--------------	----

え

エラーハンドラ.....	44
--------------	----

お

オーバランハンドラ.....	39
オブジェクト.....	10

か

カーネルの起動.....	45
拡張同期・通信機能.....	30
可変長メモリプール.....	35

き

起床待ち状態.....	24
起動された状態.....	14
キューイング.....	11

休止状態.....	13
強制待ち状態.....	13

け

現在優先度.....	11
------------	----

こ

広義の待ち状態.....	13
固定長メモリプール.....	34
コンテキスト.....	10

さ

サービスコール.....	11
--------------	----

し

時間管理機能.....	36
時間経過待ち状態.....	72
システム構成管理機能.....	42
システムコール.....	11
システムコールの遅延実行.....	19
システムサービスブロック.....	19
システム時刻.....	11, 36
システム状態管理機能.....	40
システムスタック.....	46
システムの起動.....	20
システムメモリ領域.....	21
自タスク.....	10
実行可能状態.....	13
実行状態.....	13
実行できる状態.....	12
周期ハンドラ.....	36, 163
状態遷移.....	12

処理単位	16
------------	----

す

スケジューラ	10
スケジューリング	10
スケジューリング規則	14
スタック用メモリ領域	21

せ

セマフォ	26
------------	----

た

タイムイベントハンドラ	36
タイムチェック	11
タスク	10
タスク管理機能	23
タスクコンテキスト	16
タスクの起動	14
タスクの終了	14
タスクの状態	12
タスクの待ち解除	14
タスク付属同期機能	24
タスク例外処理	25

て

ディスパッチ	10
ディスパッチが起こらない状態	13
ディスパッチ禁止状態	17
ディスパッチ保留状態	18
ディスパッチャ	10
データ型	204
データキュー	27

デバイスドライバ管理機能	43
--------------------	----

と

同期・通信機能	26
---------------	----

に

二重待ち状態	13
--------------	----

は

パケット形式	206
--------------	-----

ひ

非タスクコンテキスト	16
------------------	----

ふ

ブリエンプティブ	11
----------------	----

へ

並行処理	10
ベース優先度	11

ま

待ち行列	11
待ち状態	13

み

未登録状態	13
ミューテックス	30

め

メールボックス	28
メッセージバッファ	31
メモリプール管理機能	34
メモリプール用メモリ領域	21

ゆ

優先順位	11, 14
------------	--------

優先度	11
-----------	----

ら

ランデブ	32
------------	----

わ

割込み管理機能	41
割込みサービスルーチン	41
割込みマスク	187

μC3/Standard ユーザーズガイド

2008年 9月	初版
2008年11月	第 2 版
2010年 6月	第 3 版
2015年 9月	第 4 版
2017年 3月	第 5 版
2017年 8月18日	第 6 版
2018年 3月27日	第 7 版
2019年 8月29日	第 8 版
2020年 6月10日	第 9 版
2022年 7月22日	第 1 0 版
2023年 8月30日	第 1 1 版
2023年 9月5日	第 1 2 版

イー・フォース株式会社 <http://www.eforce.co.jp/>

TEL 03-5614-6918 FAX 03-5614-6919

お問い合わせ info@eforce.co.jp

Copyright (C) 2008 - 2023 eForce Co.,Ltd. All Rights Reserved.
