µC3/Compact ユーザーズガイド

第 5 版　イー・フォース株式会社

はじめに

μC3（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μITRON4.0 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。

μC3 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、μC3/Compact 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。なお、カーネルのバージョンアップ、コンフィグレータのバージョンアップにより、旧バージョンとで仕様の異なる機能があります。

そのような箇所では、旧カーネルバージョンを**Ver.1.x カーネル**とし、旧コンフィグレータバージョンを**Ver.2.x コンフィグレータ**とし新バージョンとの差異を説明しています。

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

μITRON は、"Micro Industrial TRON"の略称です。

μITRON4.0 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。

μC3 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
	レイアウトの変更
32	CPU 選択について説明を追加
32,34,35,37,39, 41,42,4445,46, 47,49,50,52	コンフィグレータのバージョンアップにともない、画像を変更
35,37,39	「カーネル共通部」を「カーネル共通」に変更
37	カーネル共通について説明を変更

第 4 版で訂正された項目

ページ	内容
23	セマフォの資源「獲得・返却」の記載訂正
28	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
116	COM ポート初期化関数のパラメータの型名を訂正
123	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
125,126	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

第 5 版で訂正された項目

ページ	内容
33-92	新コンフィグレータに対応。それに伴い、章構成を変更した。 章 4.1: Ver2.x コンフィギュレータ操作説明 章 4.2: 現バージョンコンフィギュレータ操作説明

目次

第１章　μC3/Compact とは

１．１　　特長

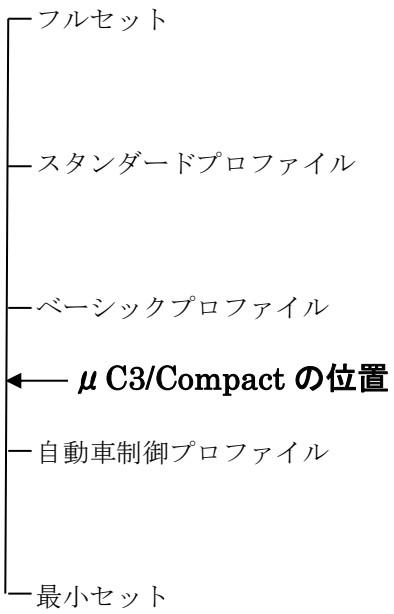
μC3/Compact は、μC3 の 3 つのコンセプトの中で Compact に特化し、RTOS の占めるメモリ消費量、特に RAM の消費量を極限まで抑えた製品です。つまり、ワンチップマイコンと呼ばれる内蔵 ROM/RAM だけのために省メモリを要求され、RTOS の採用を躊躇っていたシステムにも使われることをコンセプトとした製品です。

オブジェクトの生成を静的のみにすることで、コードサイズを抑え、RAM 領域に配置される管理データは最適化され少なくなりました。また、静的 API を採用せず、GUI 方式のコンフィグレータを採用し、必要とされるコンフィグレーションデータを効率よく管理データに変換でき、消費メモリを抑えています。

RTOS を使用したアプリケーションプログラムを設計した際、タスク毎に割り当てるスタック領域が RAM 領域を逼迫しがちですが、これを緩和できるよう共有スタックに対応しています。

１．２　　μITRON 仕様での位置づけ

μITRON4.0 仕様にはプロファイルの概念を持っていますが、μC3/Compact はどのプロファイルからも逸脱しています。ただし、μC3/Compact の開発コンセプトの決定に際し、ベーシックプロファイルよりも低い自動車制御プロファイルを基本としました。



自動車制御プロファイルから逸脱した仕様として、GUI 方式のコンフィグレータを採用したために静的 API が無いことと、CPU 例外ハンドラが無いことが挙げられます。逆に、必須ではないがサポートしている機能としては、次の機能があります。

・タイムアウト付きのシステムコール
・メールボックス（メッセージ優先度順は非サポート）
・固定長メモリプール

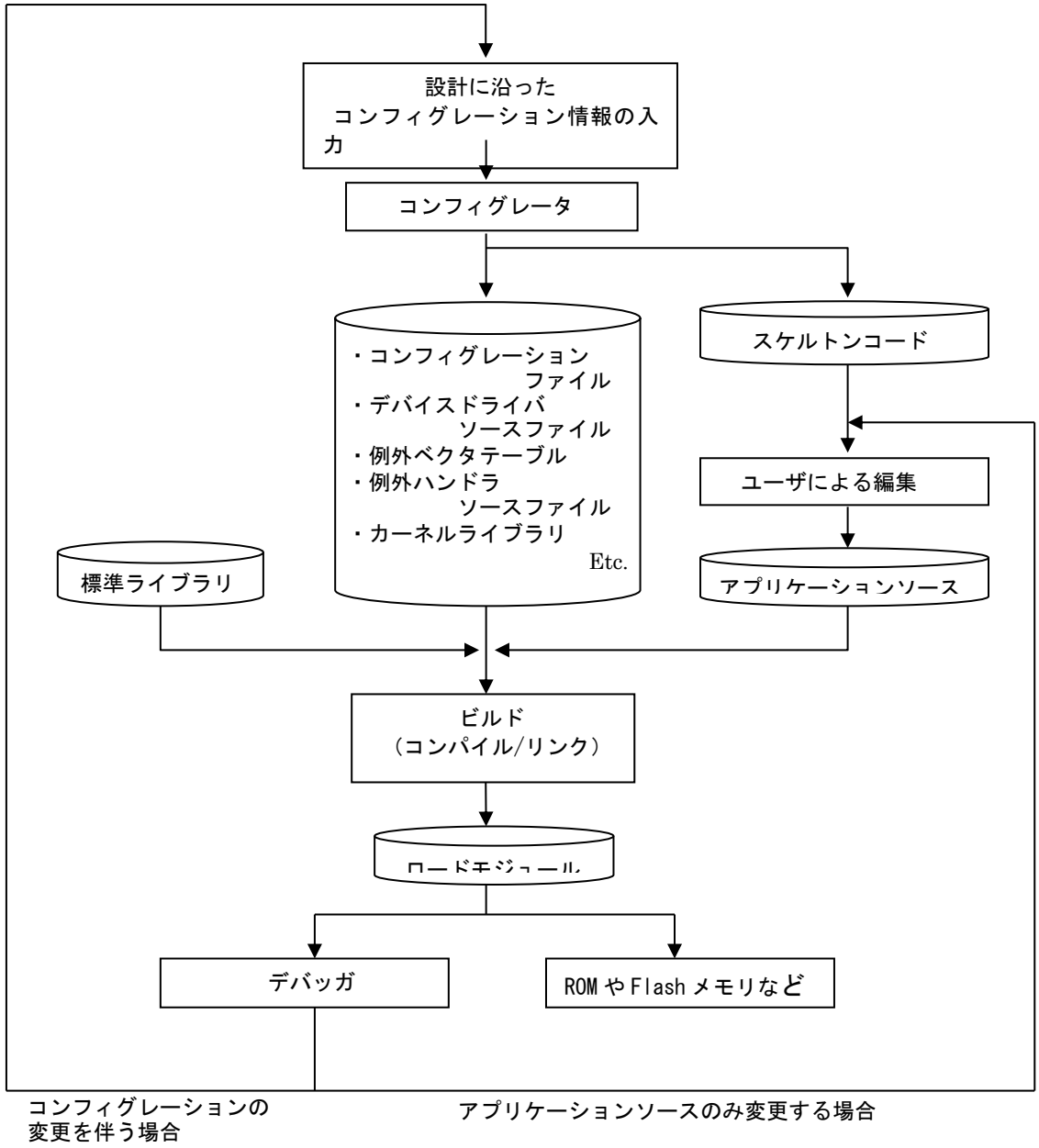
１．３　開発手順

μC3/Compact を使用したシステムの開発手順を図に示します。

最初に、システム設計で決定した RTOS のコンフィグレーション情報（必要とするオブジェクトの個数や属性など）をコンフィグレータに入力します。コンフィグレータはコンフィグレーション情報を元にコードを生成します。生成されたコードは、修正せずに使用するファイルとスケルトンコードがあります。このスケルトンコードは、必要なアプリケーションプログラムを記述する際の助けとなるよう作られています。

アプリケーションプログラムを記述した後、ビルド（コンパイル/リンク）してロードモジュールを生成します。このロードモジュールをデバッグし、デバッグが終了した場合は ROM や Flash メモリなどに書き込み、システムは完成します。また、RTOS のコンフィグレーションに誤りがあった場合は、再度コンフィグレータを起動し、コンフィグレーションからやり直します。

スケルトンコード以外のファイルを修正して使用することを禁止します。



開発手順図

【推奨】

コンフィグレータはコード生成によりスケルトンコードを出力します。そのため、スケルトンコードを直接編集した場合、コンフィグレーションの変更によるコード生成で上書きされてしまいます。これを防ぐため、スケルトンコードを直接編集せず、テンプレートとして用いてアプリケーションプログラムを作成することを推奨します。

第２章　μC3/Compact 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、μC3 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。μC3/Compact では ID 番号をコンフィグレータが割り当てるため、アプリケーションでは ID 番号の定義名（マクロ名）を用いてシステムコールを呼び出します。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、メールボックス、固定長メモリプール、データキュー、周期ハンドラ、割込みサービスルーチン、共有スタックがあります。ただし、割込みサービスルーチンは、参照するシステムコールが存在しないため、ID 番号を持ちません。

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。μC3 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。μC3/Compact にはミューテックスは実装されていないため、ベース優先度と現在優先度は常に同じ優先度になります。

２．１．７　　制約タスク

共有スタックを可能にするためにタスクの機能を制限するタスクの属性で、待ちに入ること と、優先度を変更することができません。また、制約タスクであることが共有スタックを使うことを意味することではなく、共有スタックを使わなくても制約タスクの属性を持たせることはできます。ただし、同一の共有スタックを持つ制約タスク属性のタスクは、同一のタスク優先度の必要があります。

２．１．８　　共有スタック

RAM 領域の少ないシステムで複数のタスクが同一のスタック空間を使用する場合、そのスタックを「共有スタック」と呼びます。μC3/Compact では、自動車制御プロファイルに定義された制約タスクを用いる方法と、カーネルが排他制御を行う方法を用意し、安全に共有スタックを使うことができるようになっています。

２．１．９　　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。

２．１．１０　タイムチック

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「タイムチック」と呼びます。つまり、タイムチックの周期が１ﾐﾘ秒であれば、システム時刻は１ﾐﾘ秒の精度に、２ﾐﾘ秒の周期であれば２ﾐﾘ秒の精度になります

２．１．１１　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。

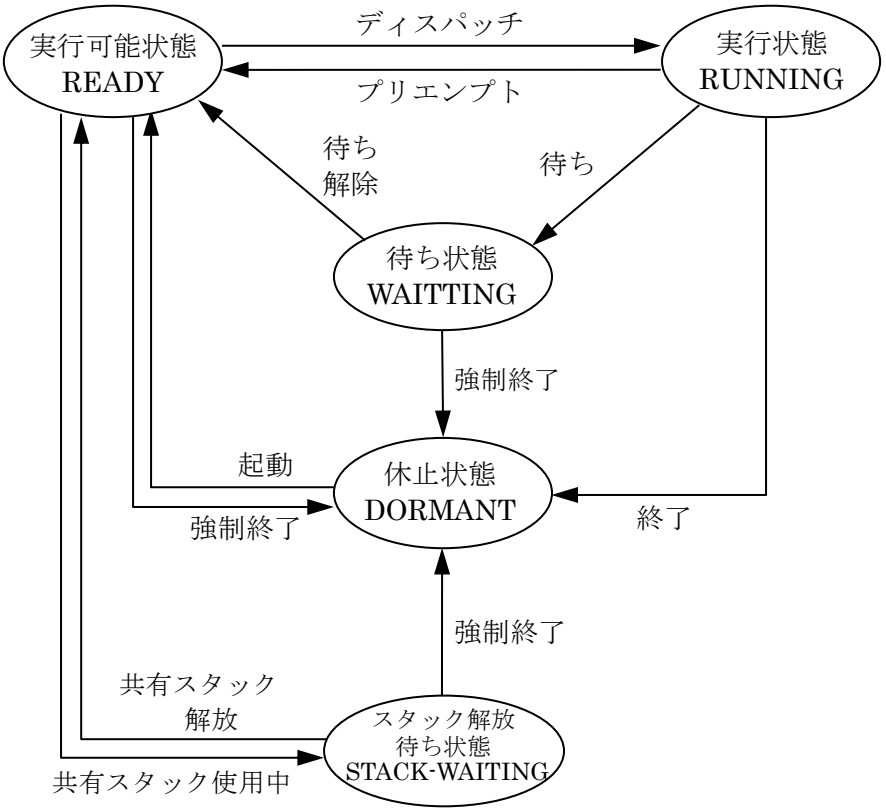
２．１．１２　待ち行列

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、タスク優先度順待ち行列もありますが、μC3/Compact では対応していません。

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

μC3/Compact でのタスクの状態は、大別して４つの状態があり、広義の待ち状態は２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から決定し、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。μC3 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、待ち状態と共有スタック解放待ち状態の２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、固定長メモリブロックの獲得待ちがあります。

C．２　共有スタック解放待ち（STACK-WAITING）

共有スタックが他のタスクにより占有状態にあり、その共有スタックの解放を待っている状態です。また、共有スタック解放状態は待ち状態とは異なり、システムコールによる共有スタック解放待ちの解除はありません。

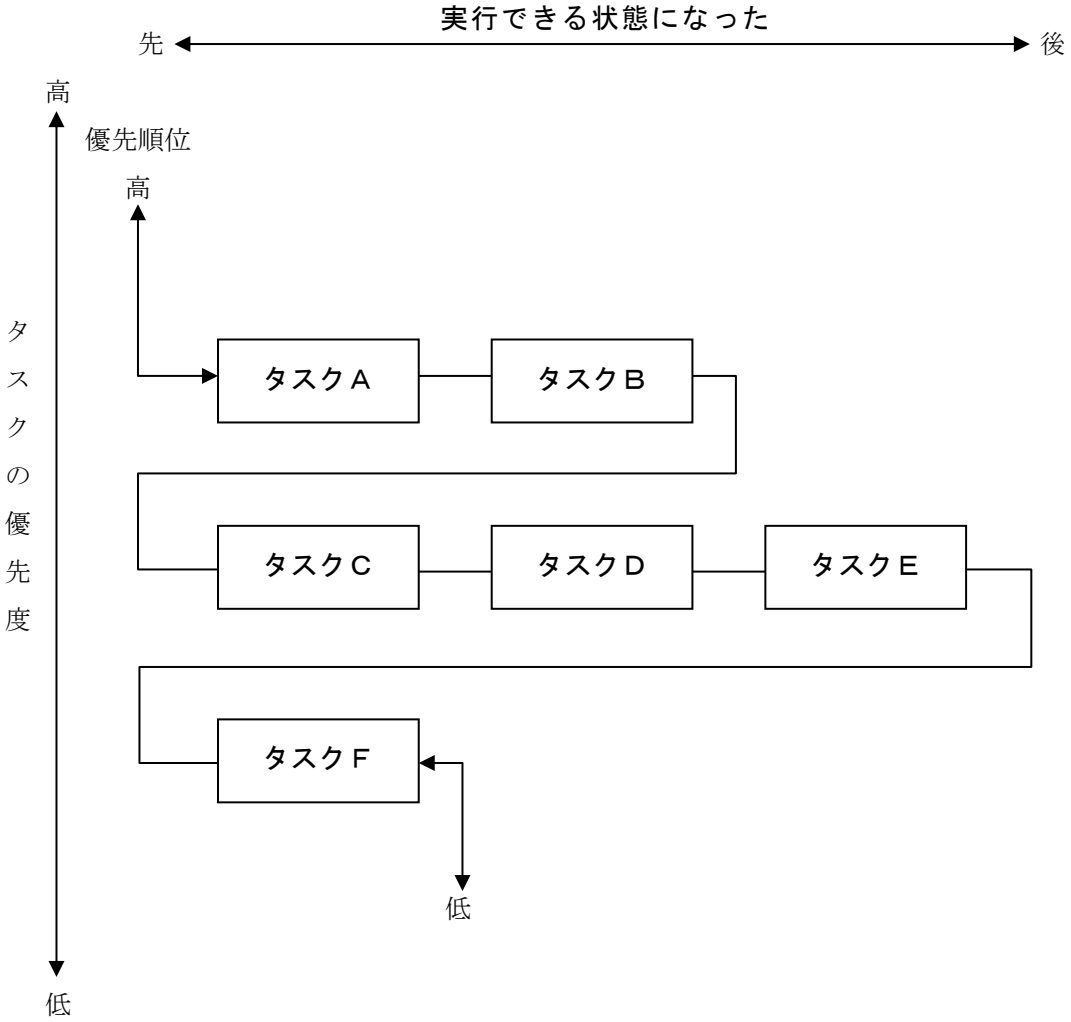
D　休止状態（DORMANT）

タスクの起動前か実行終了後の状態。休止状態にある間は、実行状態における情報は保存されていない。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する。

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）の呼出により、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。



優先順位と優先度の関係図

第３章　μC3/Compact 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

μC3/Compact のカーネルは、次の処理単位で実行されます。

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. タスク

D. アイドル

μC3/Compact では、割込みハンドラをカーネル内部でのみ使用し、割り込み処理は割込みサービスルーチンで記述します。

タイムイベントハンドラは、時間をきっかけとして起動される処理で、周期ハンドラのみを実装しています。

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、割込みサービスルーチン、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があります。

μC3/Compact 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、割込みサービスルーチン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」と言い、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」と言います。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、割込みサービスルーチン開始直後の状態は、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、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。

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」と言い、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」と言います。

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。

割込みサービスルーチン、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／ルーチン実行開始直後は、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／ルーチン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。μC3/Compact 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っています。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。

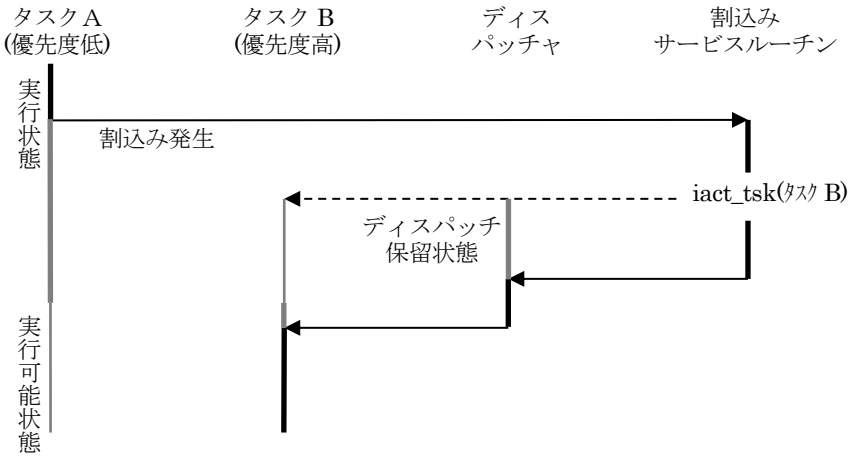
カーネルが用意するアイドルは、処理のない単純ループになっています。このアイドルには、コンフィグレーションでユーザによるアイドル関数を定義することができます。

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　　共有スタック

共有スタックは、複数のタスクが使用できるスタック空間です。また、共有スタックは、同時に複数のタスクで使用できるのではなく、複数のタスクが同時に同じスタック空間を使わないよう、排他制御する機能を持っています。共有スタックの排他制御を実現する方法には、制約タスクの属性を指定する方法と、制約タスクの属性なしにスタック解放待ち状態を使う方法の、2 つがあります。ただし、同じ共有スタックを、制約タスクの属性を持つタスクと、制約タスクの属性を持たないタスクを、混在させることはできません。

３．２．１　　制約タスクの属性を使う方法

共有スタックを使用するタスクに対して制約タスクの属性を指定し、同じ共有スタックを使用するタスクはすべて同じタスク優先度を指定します。これにより、一旦、実行状態となったタスクが終了するまで、同じ共有スタックを使用するタスクにディスパッチされないように排他制御が行われます。ある共有スタックを指定した、タスク優先度 N のタスク A とタスク B を連続して起動し、タスク A にディスパッチされた場合を考えます。

タスク A が終了する前に、タスク B が実行状態になれば、スタック空間を 2 つのタスクが同時に使用し、スタックの内容を破壊します。タスク A が終了する前にタスク B にディスパッチされる要因は、タスク A が待ち状態になることと、タスク優先度 N の優先順位に移動が起こる場合が考えられます。制約タスクの属性により、これを防ぎます。制約タスクは待ち状態に遷移させないので、タスク優先順位から外れることはありません。また、タスク優先順位の順序に変更が加えられるシステムコールとして、chg_pri,rot_rdq があります。そのため、制約タスクを指定した chg_pri と、最も優先順位の高いタスクが制約タスクのある優先度を指定した rot_rdq を呼び出した場合には、E_CTX エラーを返します。つまり、タスク A を指定した chg_pri と優先度 N を指定した rot_rdq は、E_CTX エラーを返します

３．２．２　　スタック解放待ち状態を使う方法

制約タスクの属性を指定せずに共有スタックを使用します。共有スタックを指定したタスクが実行可能状態から実行状態に遷移する際に、共有スタックが他のタスクに占有されていなければ共有スタックを占有しディスパッチされ、共有スタックが他のタスクに占有されていれば共有スタック解放待ち状態に遷移します。共有スタックを占有していたタスクが終了した場合には、共有スタックを解放し、その共有スタックの共有スタック解放待ち状態のタスクの優先順位が高いタスクを実行可能状態に遷移させます。

この方法が有用な場合としては､同時に実行されることのないタスクなので､アプリケーションで排他制御を行い、共有スタックを指定したタスクを起動する場合があります。このような場合でも、タスクが完全に休止状態に遷移したことを確認しなくても、安全に共有スタックを使用できます。また、排他的にタスクを起動しなければならない場合にも、共有スタックを指定することで、容易に実現できます。

３．３　コンフィグレータ

システム設計で決定される、各オブジェクトのパラメータとなるコンフィグレーション情報をコンフィグレータに入力し、必要なソースファイルとアプリケーションプログラムのテンプレートになるスケルトンコードを生成します。プロセッサに依存しないカーネルのコンフィグレーション情報と、プロセッサに依存したデバイスドライバのコンフィグレーションがあります。さらに、カーネルのコンフィグレーションには、カーネル共通のコンフィグレーション情報とオブジェクトのコンフィグレーション情報があります。また、デバイスドライバのコンフィグレーションは、プロセッサ依存になり、これらの説明は「プロセッサ依存部マニュアル」、「デバイス依存部マニュアル」を参照してください。

３．３．１　カーネル共通のコンフィグレーション情報

カーネル共通のコンフィグレーション情報には、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を指定するタスク優先度数
- コンフィグレーションファイルに追加インクルードする追加ヘッダファイル
- ユーザ定義のアイドル関数
- システムスタックのサイズ
- カーネル割込みレベル（Ver.2.x カーネルのみ）

３．３．２　オブジェクトのコンフィグレーション情報

次の静的 API に相当するコンフィグレーションと、共有スタック生成のコンフィグレーションを行います。

- CRE_TSK　タスクの生成
- CRE_SEM　セマフォの生成
- CRE_FLG　イベントフラグの生成
- CRE_DTQ　データキューの生成
- CRE_MBX　メールボックスの生成
- CRE_MPF　固定長メモリプールの生成
- CRE_CYC　周期ハンドラの生成
- ATT_ISR　割込みサービスルーチンの追加

３．３．３　生成されるソースファイル

大別すると、これらのコンフィグレーション情報を元にしたファイルを生成します。ただし、生成されるファイルの種類はコンフィグレーション内容により異なります。

- 必ず生成されるプロセッサに依存しないファイル
- 必ず生成されるプロセッサに依存したファイル

・　デバイスドライバに依存したファイル

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）

タスクに対する起動要求は、キューイングされます。すなわち、すでに起動されているタスクを再度起動しようとすると、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の初期化と、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からリターンした場合には、ext_tsk を呼び出した場合と同じ振舞いをします。

タスク管理機能に関連して、次のカーネル構成定数を定義します。

TMAX_ACTCNT　　タスクの起動要求キューイング数の最大値（255）

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）
- 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と、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。

タスク付属同期機能に関連して、次のカーネル構成定数を定義します。

TMAX_WUPCNT	タスクの起床要求キューイング数の最大値（255）

３．６　　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。

３．６．１　　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（具体的には、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(具体的には､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、次のカーネル構成定数を定義します。

TMAX_MAXSEM セマフォの最大資源数の最大値（255）

３．６．２　　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可能です。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またはいずれかがセットされるまで、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、次のカーネル構成定数を定義します。

TBIT_FLGPTN　イベントフラグのビット数（ プロセッサに依存）

３．６．３　　データキュー

データキューは、 1 ワードのメッセージ（これをデータと呼ぶ）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。

- 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,prcv_dtq,trcv_dtq）
- データキューの状態を参照する機能（ref_dtq）

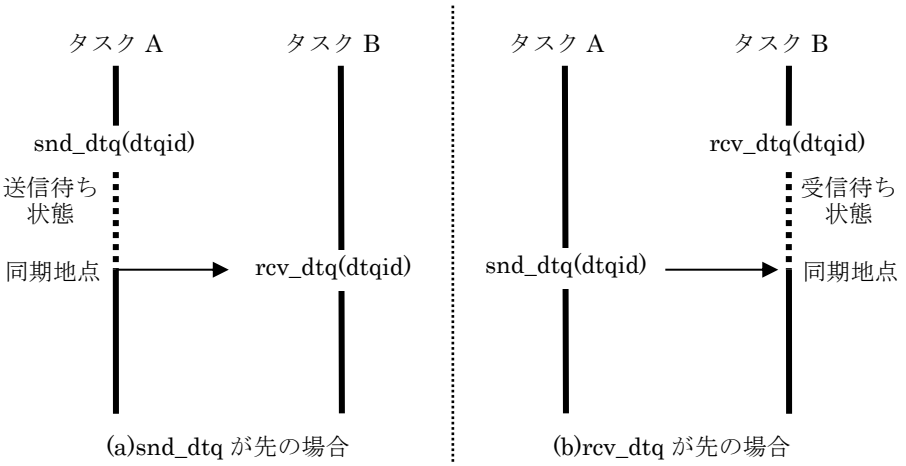
データキューは、データの送信を待つタスクの送信待ち行列とデータの受信を待つタスクの受信待ち行列を持ちます。また、送信されたデータを格納するためのリングバッファ状のデータキュー領域を持ちます。データを送信する側では、送信したいデータをデータキューに入れます。データキュー領域に空きがない場合は、データキュー領域に空きができるまで、あるいは許された時刻まで、送信待ち行列につながれ、データキューへの送信待ち状態に遷移します。

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、受信待ち行列につながれ、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、データの受渡しが行われます。

データキューで送受信されるデータ（ 1 ワードのメッセージ）は、整数値であっても、送信側と受信側で共有しているメモリ上に置かれたメッセージの先頭番地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

３．６．４　　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,pol_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行います。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。

例えば、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*          pk_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;
```

アプリケーションは、メッセージキューに入っているメッセージの内容を決して書き換えてはなりません。また、既にメッセージキューに入っているメッセージを再度メールボックスに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。

【推奨】

メッセージパケットとしては、固定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。

３．７　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があります.

３．７．１　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なります。

アプリケーションは、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、既に返却されたメモリブロックを重複して返却してもなりません。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。

３．８　　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、周期ハンドラの各機能が含まれます。

３．８．１　　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、システムコールやコンフィグレータ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。

３．８．２　　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。

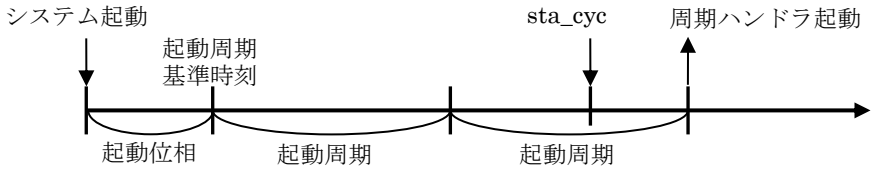
- 周期ハンドラの動作を開始する機能（sta_cyc）
- 周期ハンドラの動作を停止する機能（stp_cyc）
- 周期ハンドラの状態を参照する機能（ref_cyc）

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。

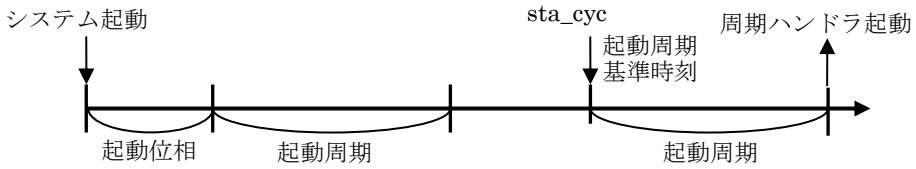
システム起動時には、TA_STA 属性を指定した場合では、動作している状態になります。また、TA_STA 属性か TA_PHS 属性のいずれかを指定した場合には、システム起動時刻に起動位相を加えた時刻を、次に起動すべき時刻とします。いずれの属性も指定していない場合には、周期ハンドラの動作を開始するシステムコール（sta_cyc）が呼び出された時刻に起動位相を加えた時刻を、次に起動すべき時刻とします。周期ハンドラを起動すべき時刻になると、その周期ハンドラの拡張情報（exinf）をパラメータとして、周期ハンドラを起動します。この時、周期ハンドラの起動すべき時刻に起動周期を加えた時刻を、次に起動すべ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

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と解釈します。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るが、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×（n－1））以上の時間が経過した後に行うことを保証します。例えば、タイムチックの周期が 10 ﾐﾘ秒のシステムにおいて、起動位相が 15 ﾐﾘ秒、起動周期が 25 ﾐﾘ秒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 ﾐﾘ秒、40 ﾐﾘ秒、70 ﾐﾘ秒、90 ﾐﾘ秒、120 ﾐﾘ秒、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
}
```

３．９　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）

３．１０　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.

INTNO	割込み番号
IMASK	割込みマスク

割込みマスクのデータ型の一部の 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
}
```

３．１１　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。

・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）

第４章　コンフィグレータの使い方

４．１　　Ver.2.x コンフィグレータの操作説明

４．１．１　　コンフィグレータの起動

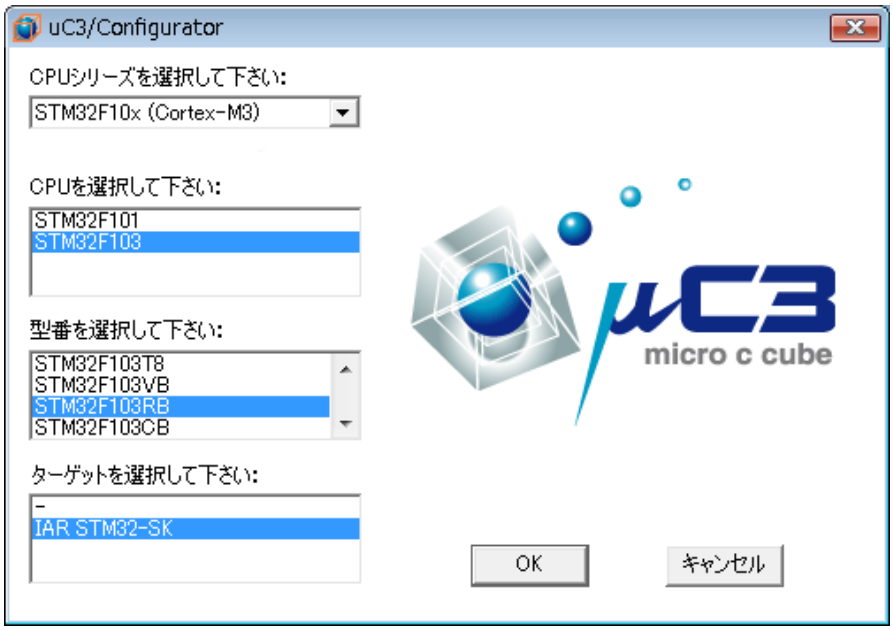
「μC3conf.exe」をダブルクリックし、起動してください。

Ａ．新規でプロジェクトを生成する場合

「新規プロジェクトの生成」を選択後に「OK」をクリックし、「CPU 選択」へ進みます。

CPU 選択

リストよりＣＰＵシリーズ、CPU、型番、ターゲットを選択後に「OK」をクリックし、「メイン画面」へ進みます。

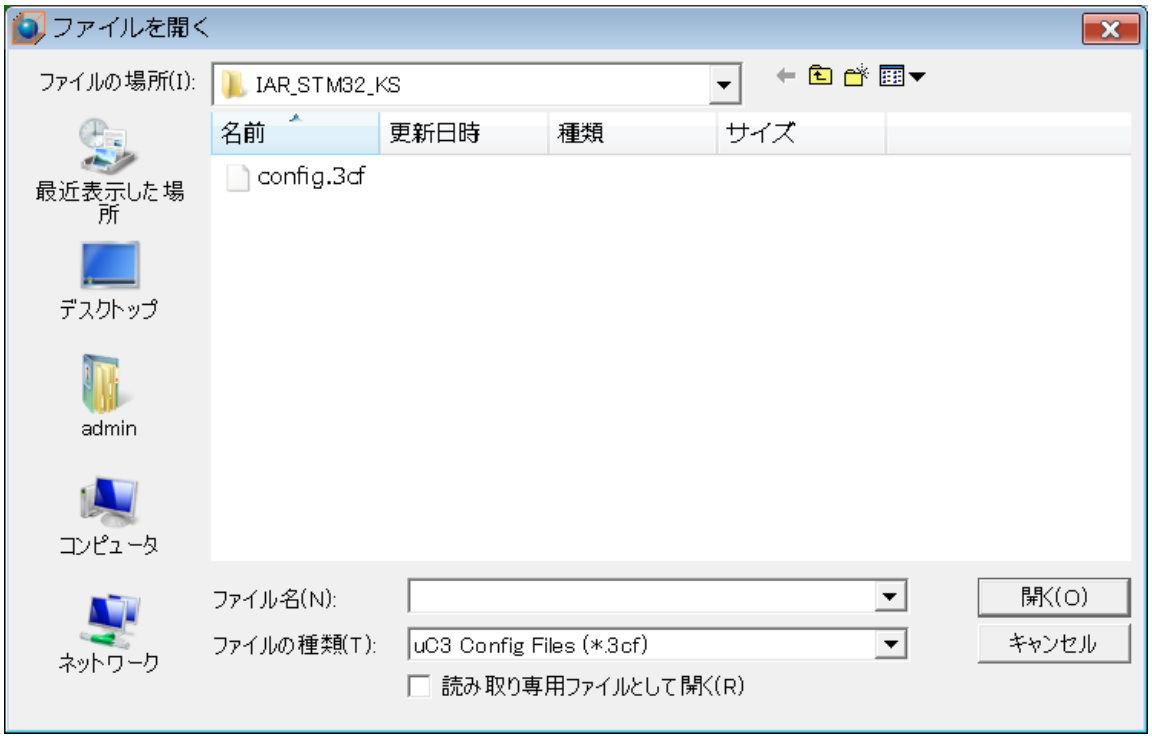


B．既存のプロジェクトを開く場合

「既存のプロジェクトを開く」を選択後に「OK」をクリックし、「ファイルを開く」へ進みます。

ファイルを開く

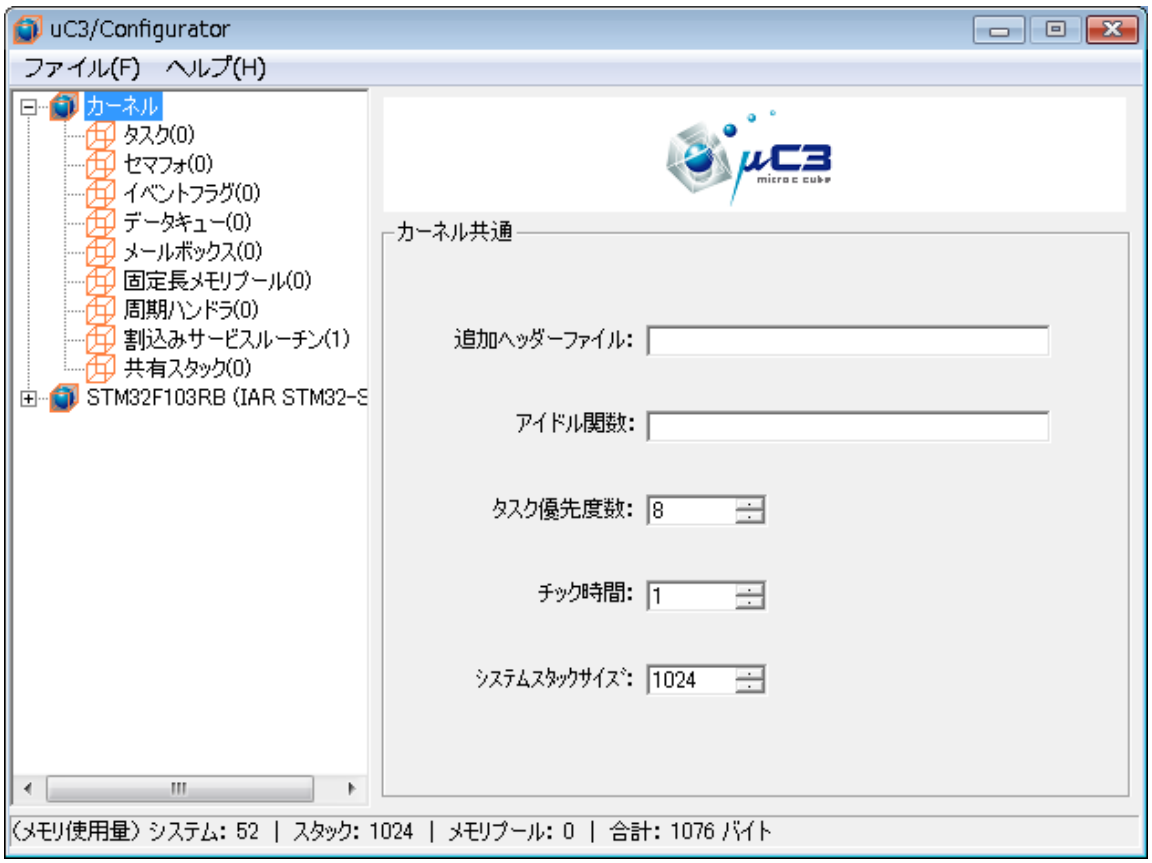
保存されていたプロジェクトファイル（拡張子.3cf）を選択後に「開く」をクリックし、「メイン画面」へ進みます。



C．メイン画面

起動後はプロジェクトの参照と編集が可能なメイン画面になります。ツリー表示のオブジェクト名をクリックすることで各オブジェクトのコンフィグレーション画面に切り替えることができます。

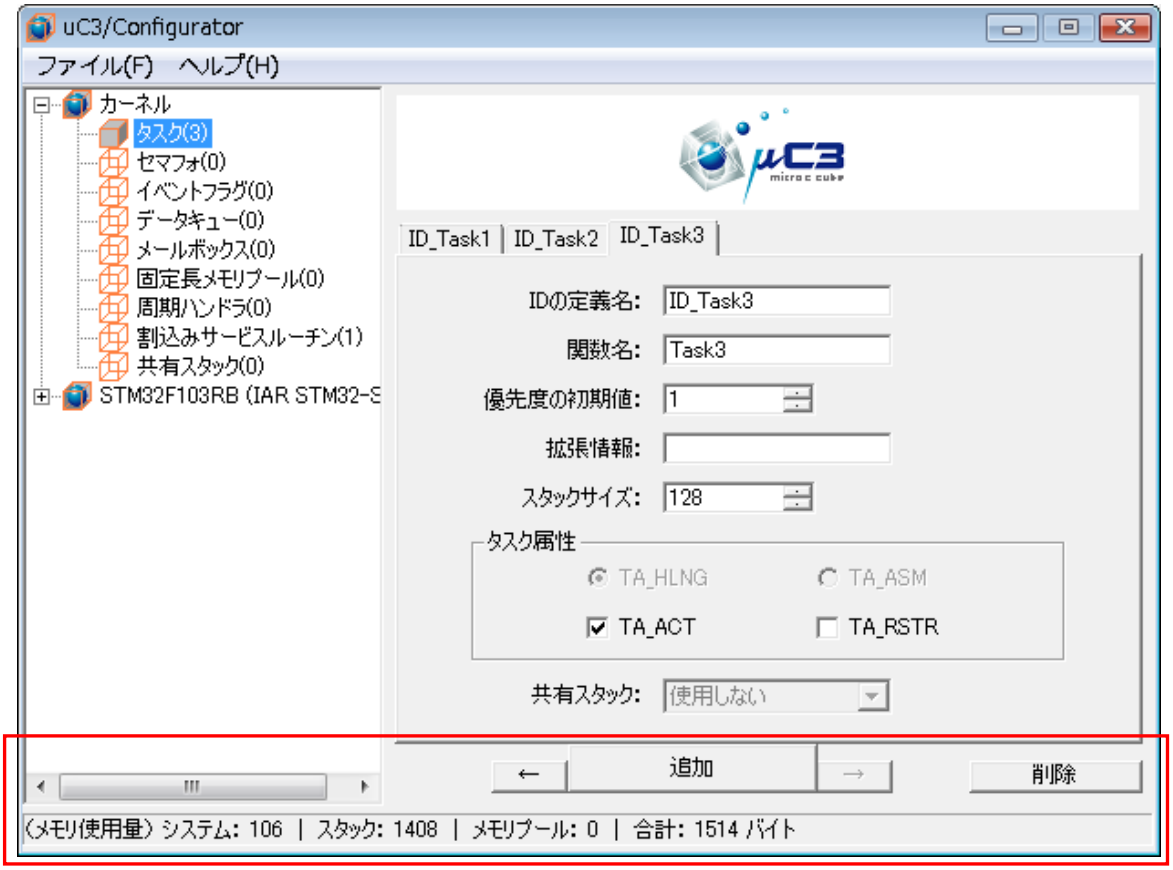
ここには、カーネルやプロセッサ依存部などのコンフィグレーションがあり、プロセッサ依存部の説明は「プロセッサ依存部マニュアル」を参照してください。



４．１．２　　カーネルの設定

カーネルのコンフィグレーションには、カーネル共通のコンフィグレーションと、タスク、セマフォ等のオブジェクトのコンフィグレーションがあります。各種オブジェクトのコンフィグレーション画面では、１つのオブジェクトが１つのタブに対応しています。

下端のステータスバーには、カーネルが管理するメモリの使用量が常に表示されます。以下はオブジェクトのコンフィグレーション画面の例です。



各項目がグレー表示され変更できない、「削除」ボタンも選択できないタブがあります。このようなタブは、デバイスドライバのコンフィグレーションに連動して追加されたオブジェクトです。

「追加」ボタン

新しいオブジェクトとそれに対応するタブを追加します。

「削除」ボタン

現在選択されているタブのオブジェクトを削除します。

「←」ボタン

現在選択されているタブを左へ移動します。

「→」ボタン

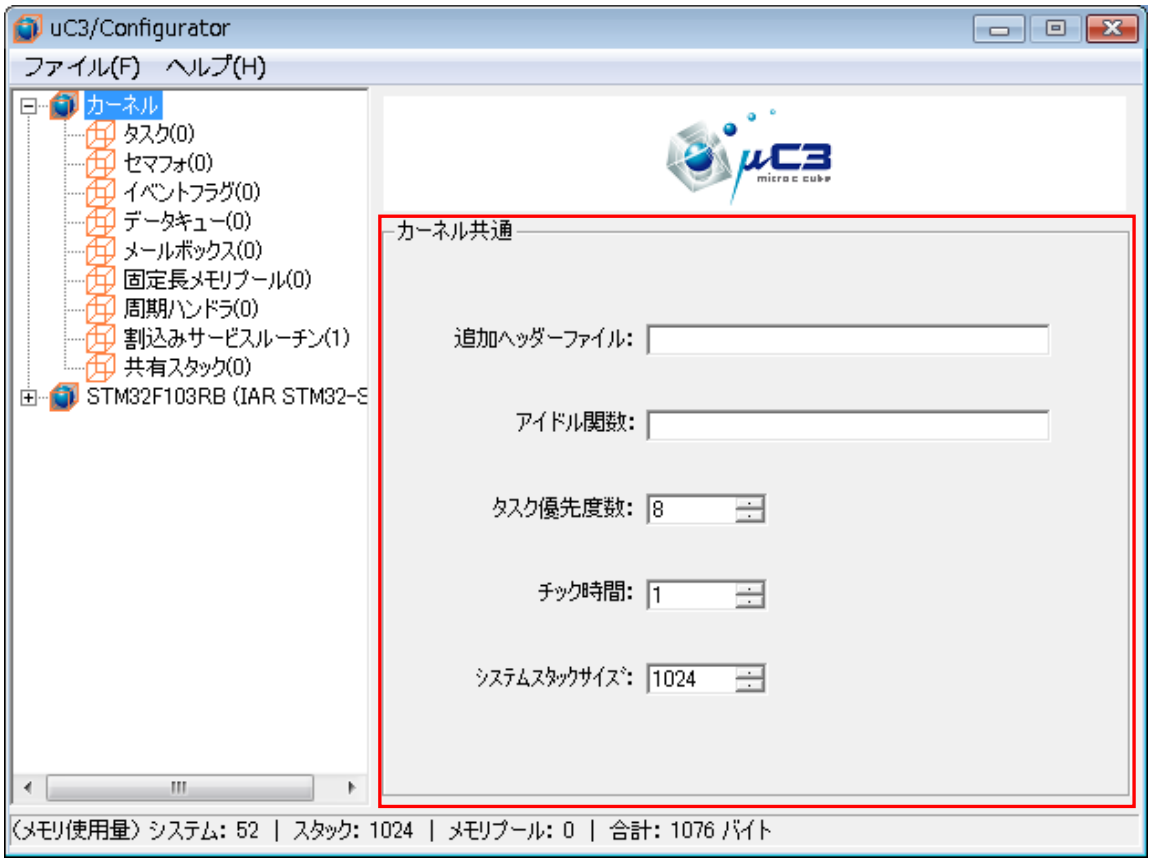
現在選択されているタブを右へ移動します。

メモリ使用量

項目	内容
システム	カーネル自体が使用するメモリ、オブジェクトの管理領域、データキューのバッファ
スタック	システムスタック、タスク個別のスタック、共有スタック
メモリプール	固定長メモリプールのメモリプール領域

４．１．２．１　　カーネル共通のコンフィグレーション

ツリー表示のカーネルをクリックすると、カーネル共通のコンフィグレーション画面が表示され、ここではカーネル共通のコンフィグレーションを行います。



追加ヘッダーファイル

タスクと周期ハンドラの拡張情報として、マクロの値か変数へのポインタを定義する場合には、マクロ定義や変数の外部宣言を記述したヘッダファイルのファイル名を指定します。具体的には、ここでファイル名を指定した場合には、そのファイルを kernel_cfg.c 内でインクルードします。

アイドル関数

カーネル内部の標準アイドル関数を使わず、ユーザ定義のアイドル関数に置き換える場合には、その関数名を指定します。

タスク優先度数

1 から 16 までを指定でき、この値を上限とするタスク優先度を指定できます。

チック時間

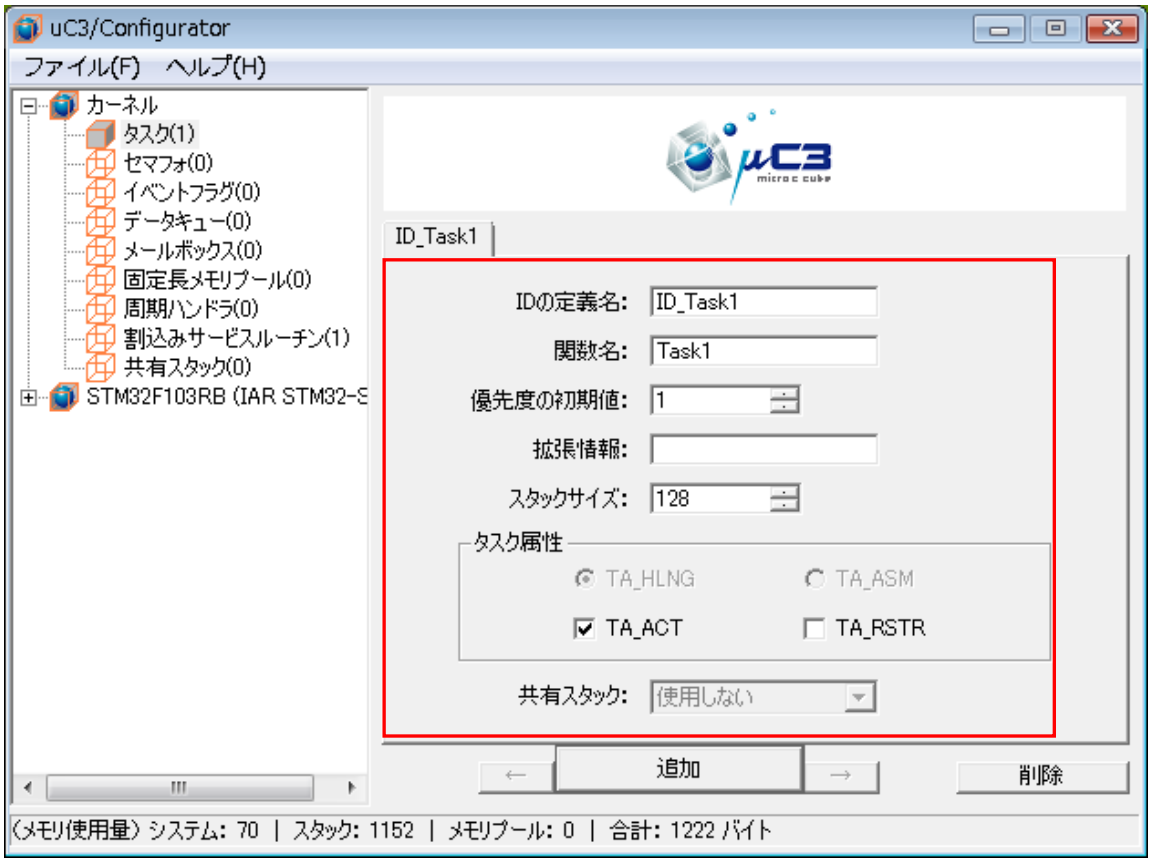
タイムチックの周期をﾐﾘ秒単位で指定します。小さな値ほど時間精度は上がりますが、オーバヘッドは大きくなります。

システムスタックサイズ

タイムイベントハンドラと割込みサービスルーチンで使うスタック領域のサイズをバイト単位で指定します。

４．１．２．２　　タスクのコンフィグレーション

ツリー表示のタスクをクリックすると、タスクのコンフィグレーション画面が表示され、ここではタスク生成 API の CRE_TSK に相当するコンフィグレーションを行います。



ID の定義名

タスクの ID 番号を表す任意の定義名を指定してください。この定義名は kernel_id.h 内でマクロ定義されます。

関数名

任意のタスクの関数名を指定します。

優先度の初期値

タスク起動時の初期タスク優先度を、カーネル共通のタスク優先度数を超えない値で指定してください。共有スタックを指定し制約タスクの属性を指定（TA_RSTR=ON）している場合には、その共有スタックを指定した他のタスクと同じタスク優先度を指定します。

拡張情報

タスクに渡す拡張情報がある場合には、それを指定し、不要の場合は空白のままにします。拡張情報には、数値、マクロ定義された値、変数へのポインタが指定できます。変数へのポインタを渡す場合は、変数名の先頭に「&」を付けます。

スタックサイズ

タスク固有のスタックのサイズを指定してください。共有スタックのフィールドが「使用しない」以外に設定されている場合には、無効となり変更できません。

TA_HLNG/TA_ASM

TA_HLNG 固定になり、μC3/Compact では変更できません。

TA_ACT

チェックすると TA_ACT 属性が ON になり、タスクを実行可能状態で生成します。

TA_RSTR

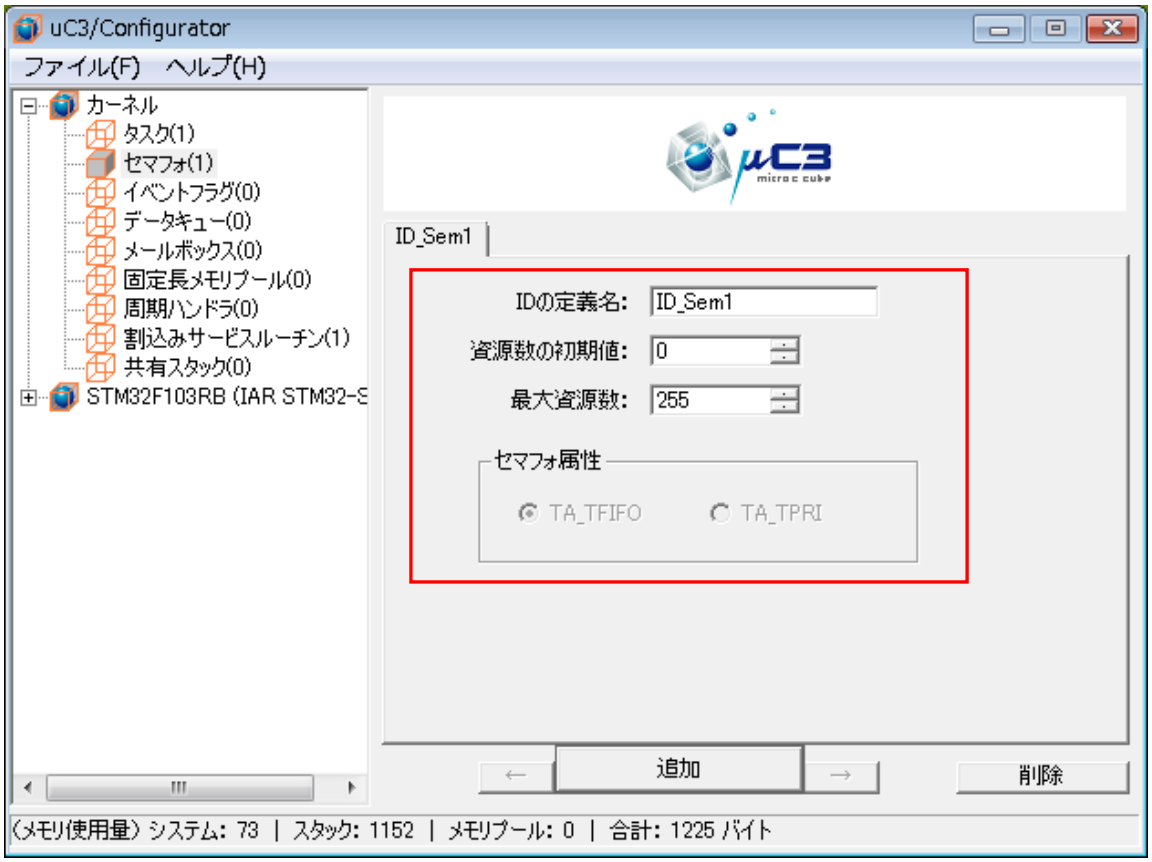
チェックすると TA_RSTR 属性が ON になり、制約タスクの属性を与えます。後述の共有スタックを選択した場合には、自動的にチェックされますが、その後チェックをはずすことも可能です。ある共有スタックを複数のタスクが使用する場合、それらのタスクはすべて TA_RSTR=ON、または、すべて TA_RSTR=OFF のどちらかでなければなりません。

共有スタック

共有スタックのフィールドは、共有スタックが１つ以上定義されている場合には、その定義名が選択可能となります。

４．１．２．３　　セマフォのコンフィグレーション

ツリー表示のセマフォをクリックすると、セマフォのコンフィグレーション画面が表示され、ここではセマフォ生成 API の CRE_SEM に相当するコンフィグレーションを行います。



ID の定義名

セマフォの ID 番号を表す任意の定義名を指定してください。この定義名は kernel_id.h 内でマクロ定義されます。

資源数の初期値

セマフォカウントの初期値を、最大資源数を超えない値を指定します。

最大資源数

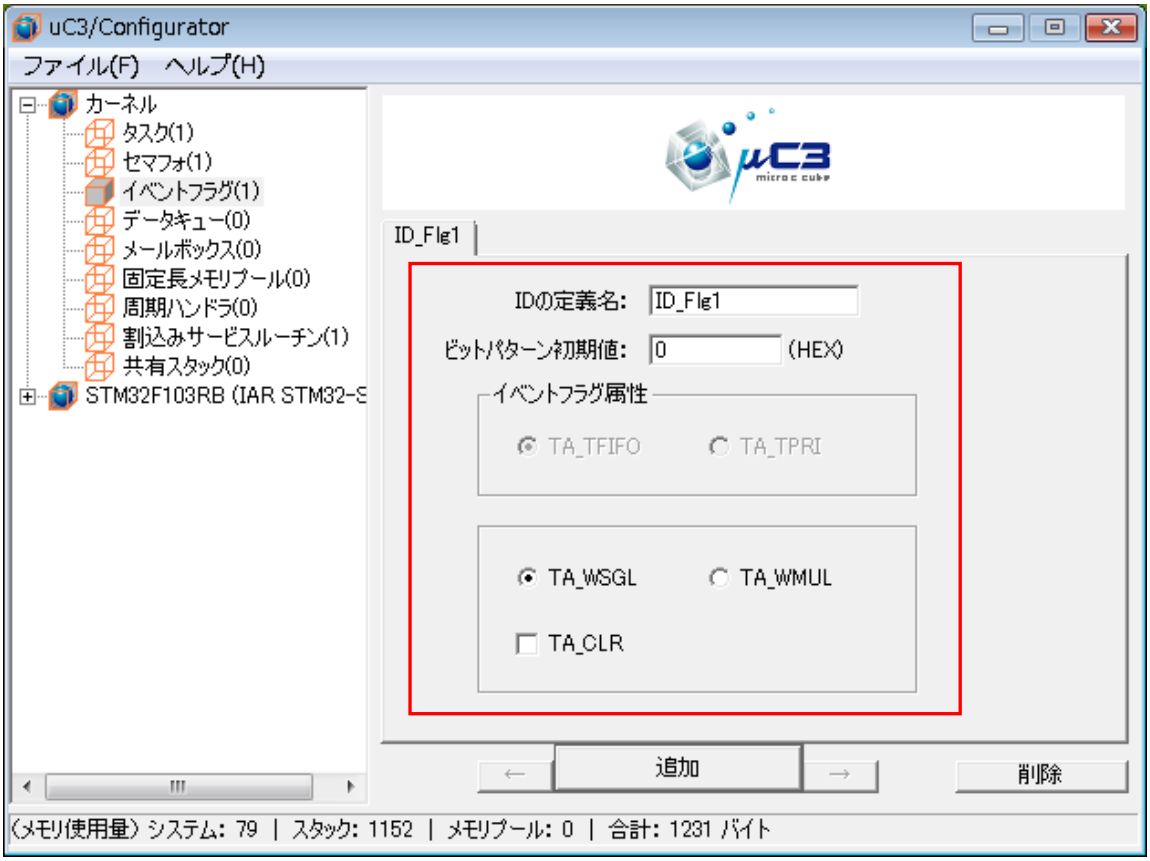
セマフォカウントの最大値を指定してください。 指定できる最大値は 255 です。

TA_TFIFO/TA_TPRI

TA_TFIFO 固定になり、μC3/Compact では変更できません。

４．１．２．４　　イベントフラグのコンフィグレーション

ツリー表示のイベントフラグをクリックすると、イベントフラグのコンフィグレーション画面が表示され、ここではイベントフラグ生成 API の CRE_FLG に相当するコンフィグレーションを行います。



ID の定義名

イベントフラグの ID 番号を表す任意の定義名を指定してください。この定義名は kernel_id.h 内でマクロ定義されます。

ビットパターン初期値

イベントフラグの初期値を 16 進数で指定してください。

TA_TFIFO/TA_TPRI

TA_TFIFO 固定になり、μC3/Compact では変更できません。

TA_WSGL/TA_WMUL

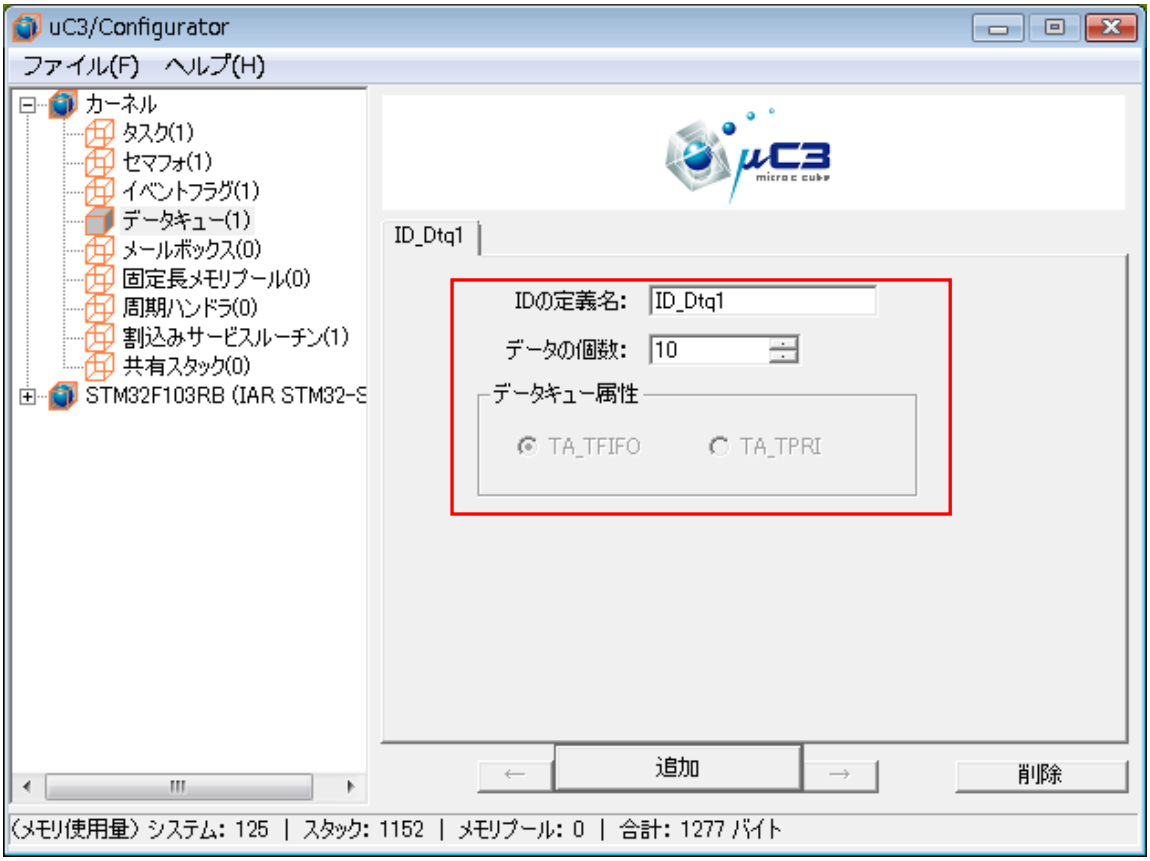
TA_WSGL の指定で、複数タスクの待ちが禁止されます。TA_WMUL の指定で、複数タスク待ちが許可されます。

TA_CLR

チェックすると TA_CLR 属性が ON になり、タスクがイベントフラグ待ちから、条件成立により待ち解除される時に、ビットパターンのすべてのビットをクリアします。

４．１．２．５　　データキューのコンフィグレーション

ツリー表示のデータキューをクリックすると、データキューのコンフィグレーション画面が表示され、ここではデータキュー生成 API の CRE_DTQ に相当するコンフィグレーションを行います。



ID の定義名

データキューの ID 番号を表す任意の定義名を指定してください。この定義名は kernel_id.h 内でマクロ定義されます。

データの個数

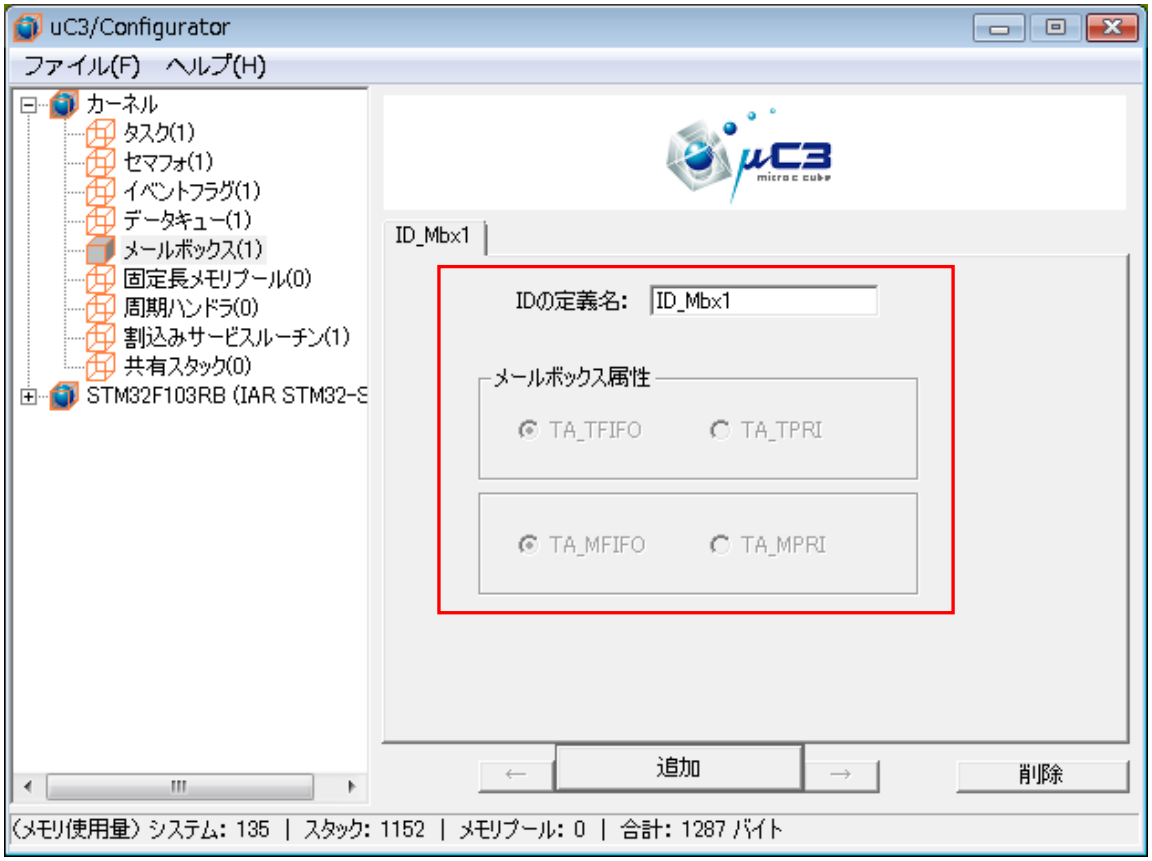
データキューの個数（データの個数）を指定します。

TA_TFIFO/TA_TPRI

TA_TFIFO 固定になり、μC3/Compact では変更できません。

４．１．２．６　　メールボックスのコンフィグレーション

ツリー表示のメールボックスをクリックすると、メールボックスのコンフィグレーション画面が表示され、ここではメールボックス生成APIのCRE_MBXに相当するコンフィグレーションを行います。



ID の定義名

メールボックスの ID 番号を表す任意の定義名を指定してください。この定義名は kernel_id.h 内でマクロ定義されます。

TA_TFIFO/TA_TPRI

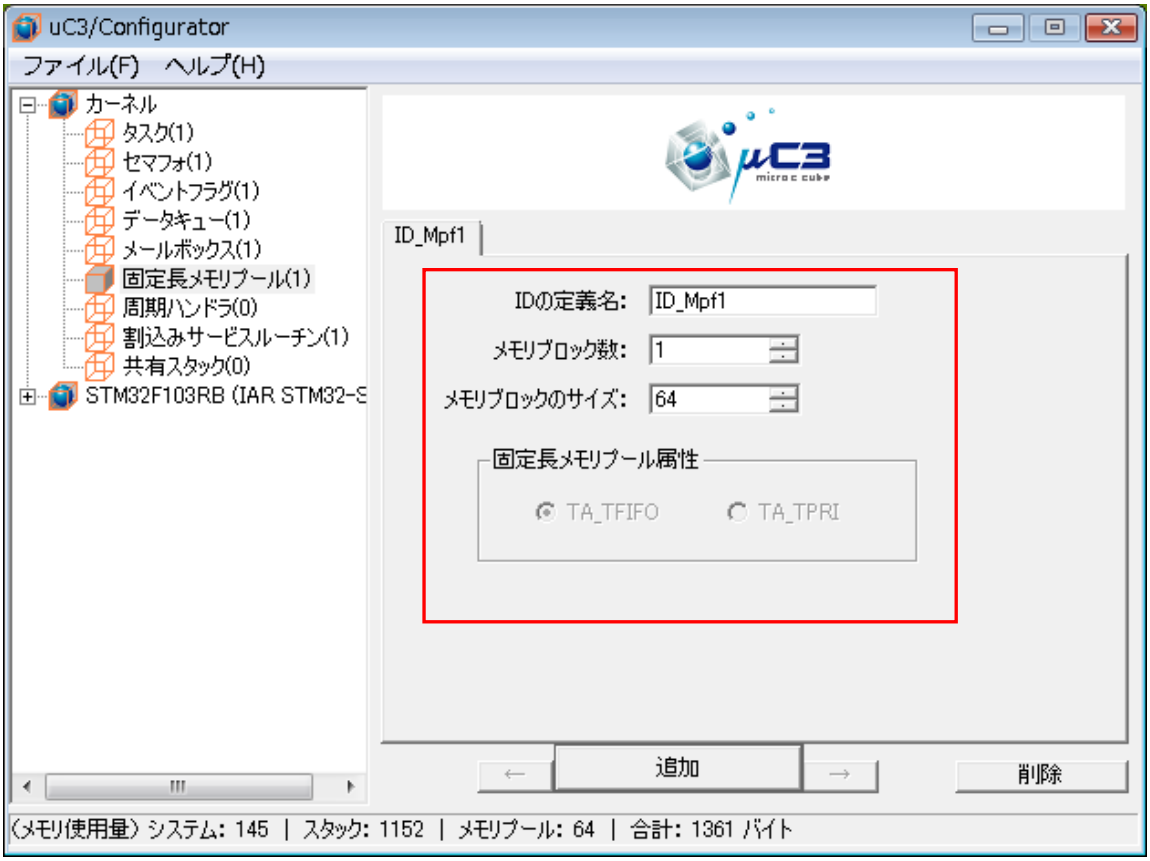
TA_TFIFO 固定になり、μC3/Compact では変更できません。

TA_MFIFO/TA_MPRI

TA_MFIFO 固定になり、μC3/Compact では変更できません。

４．１．２．７　　固定長メモリプールのコンフィグレーション

ツリー表示の固定長メモリプールをクリックすると、固定長メモリプールのコンフィグレーション画面が表示され、ここでは固定長メモリプール生成 API の CRE_MPF に相当するコンフィグレーションを行います。



ID の定義名

固定長メモリプールの ID 番号を表す任意の定義名を指定してください。この定義名は kernel_id.h 内でマクロ定義されます。

メモリブロック数

メモリブロックの個数を指定します。

メモリブロックサイズ

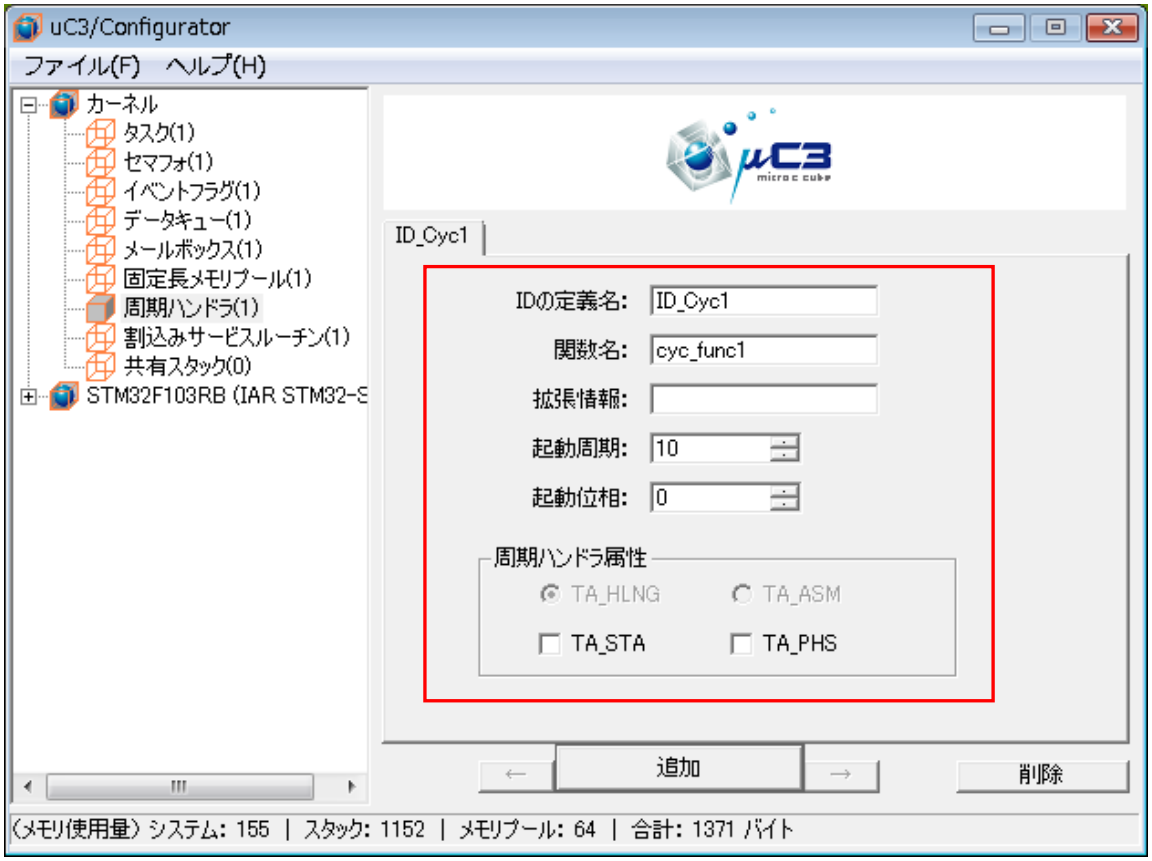
メモリブロックのサイズ（バイト数）を指定します。

TA_TFIFO/TA_TPRI

TA_TFIFO 固定になり、μC3/Compact では変更できません。

４．１．２．８　　周期ハンドラのコンフィグレーション

ツリー表示の周期ハンドラをクリックすると、周期ハンドラのコンフィグレーション画面が表示され、ここでは周期ハンドラ生成 API の CRE_CYC に相当するコンフィグレーションを行います。



ID の定義名

周期ハンドラの ID 番号を表す任意の定義名を指定してください。この定義名は kernel_id.h 内でマクロ定義されます。

関数名

任意の周期ハンドラの関数名を指定します。

拡張情報

周期ハンドラに渡す拡張情報がある場合には、それを指定し、不要の場合は空白のままにします。拡張情報には、数値、マクロ定義された値、変数へのポインタが指定できます。変数へのポインタを渡す場合は、変数名の先頭に「&」を付けます。

起動周期

周期ハンドラの起動周期をﾐﾘ秒単で指定します。ただし、チック時間より小さい値は指定できません。

起動位相

周期ハンドラの起動位相をﾐﾘ秒単で指定します。

TA_HLNG/TA_ASM

μC3/Compact では変更できません。

TA_STA

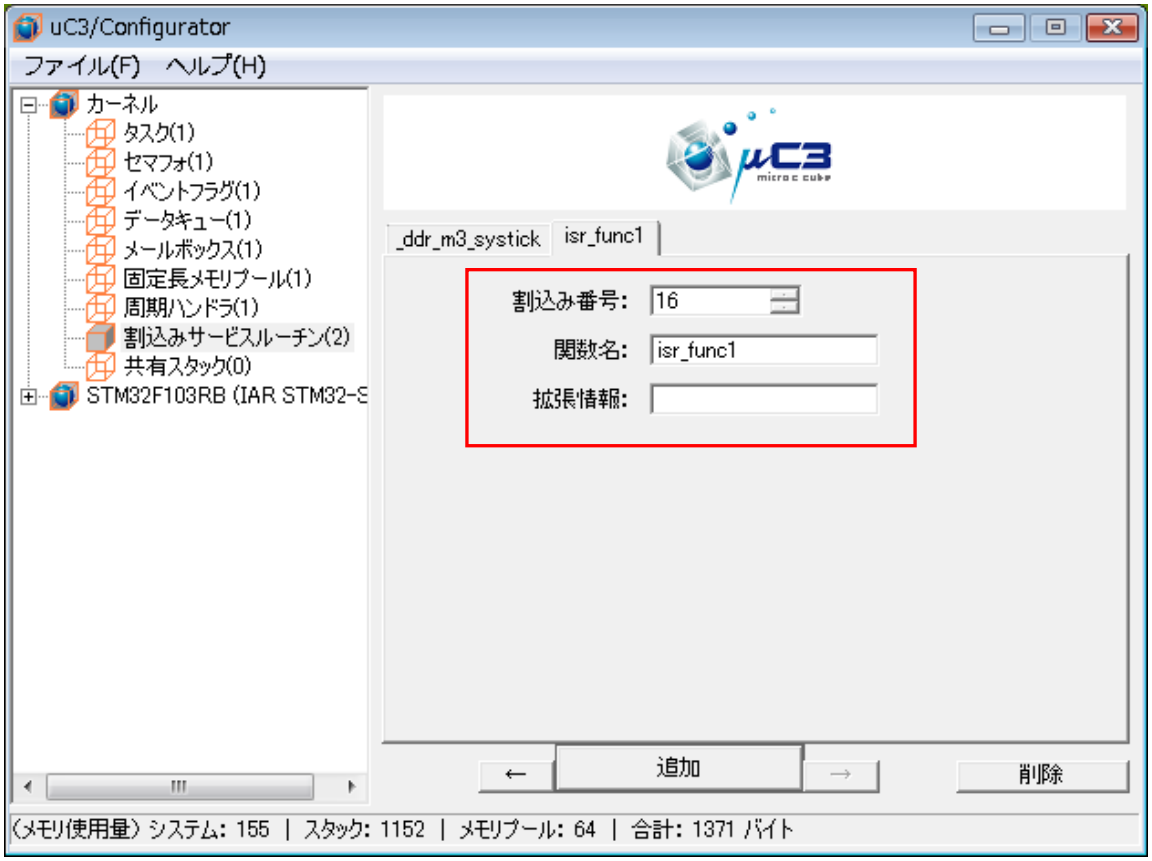
チェックすると TA_STA 属性が ON になり、周期ハンドラは動作状態で生成されます。

TA_PHS

チェックすると TA_PHS 属性が ON になり、周期ハンドラ生成時の位相を保存します。

４．１．２．９　　割込みサービスルーチンのコンフィグレーション

ツリー表示の割込みサービスルーチンをクリックすると、割込みサービスルーチンのコンフィグレーション画面が表示され、ここでは割込みサービスルーチンの追加 API の ATT_ISR に相当するコンフィグレーションを行います。



割込み番号

割込み番号を指定してください。同一の割込み番号に対して、複数の割込みサービスルーチンをコンフィグレーションした場合には、その呼出しの順序はタブ上での順序に従い、左にあるものほど早く呼ばれます。

関数名

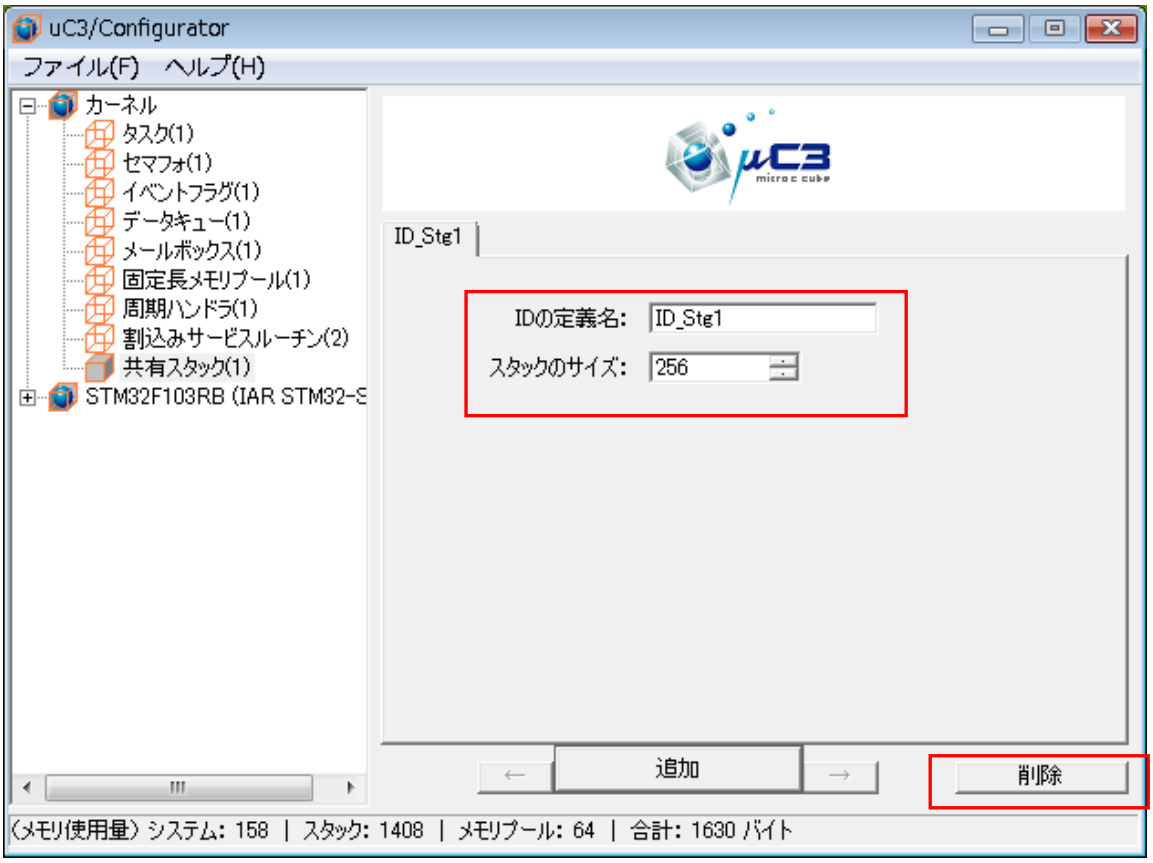
任意の割込みサービスルーチンの関数名を指定します。

拡張情報

割込みサービスルーチンに渡す拡張情報がある場合には、それを指定し、不要の場合は空白のままにします。拡張情報には、数値、変数へのポインタが指定できます。

４．１．２．１０　共有スタックのコンフィグレーション

ツリー表示の共有スタックをクリックすると、共有スタックのコンフィグレーション画面が表示され、ここでは共有スタックのコンフィグレーションを行います。



ID の定義名

共有スタックの ID 番号を表す任意の定義名を指定してください。この定義名は、タスクのコンフィグレーション画面で共有スタックを選択するために使用されます。
この共有スタックを使用しているタスクが１つでもある場合には、定義名を変更することはできません。

スタックのサイズ

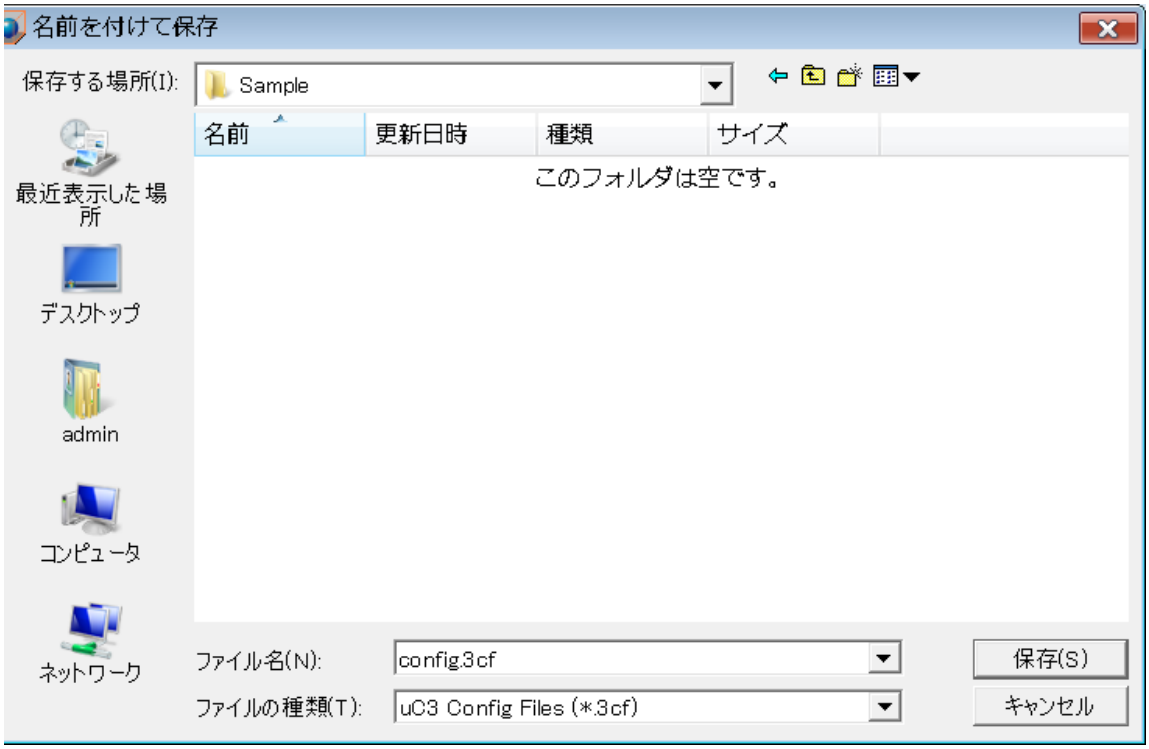
共有スタックのサイズ（バイト数）を指定します。共有スタックの使用を選択したタスクのスタックサイズは、共有スタックのサイズに固定されます。そのため、この共有スタックを指定したタスクで、一番多くスタックを使用するタスクのスタックサイズを指定します。

削除

その共有スタックを使用しているタスクが１つでもある場合、警告メッセージが表示され、削除されます。その際、タスクの共有スタックは「使用しない」に変更されます。

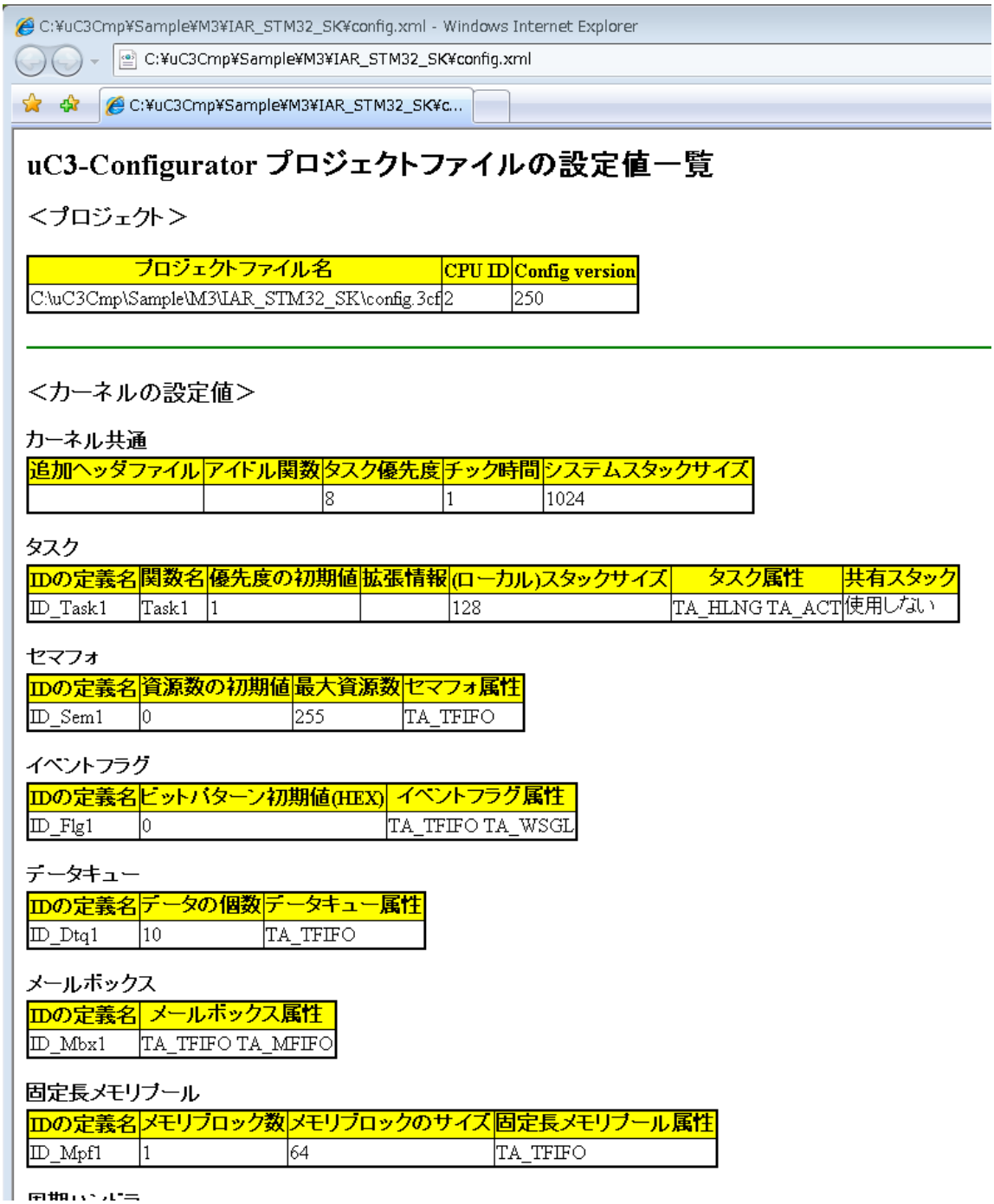
４．１．３　　プロジェクトファイルの保存

「ファイル」→「保存...(S)」により、「名前を付けて保存画面」を開き、プロジェクトファイルの保存先フォルダを指定し、「OK」をクリックします。



保存されるファイルは、プロジェクトファイル（デフォルト config..3cf）と拡張子を「xml」に変えたファイルが保存されます。

このファイルをブラウザで開くことにより、コンフィグレーション情報を確認することができます。



uC3-Configurator プロジェクトファイルの設定値一覧

＜プロジェクト＞

プロジェクトファイル名	CPU ID	Config version
C:\uC3Cmp\Sample\M3\IAR_STM32_SK\config.3cf	2	250

＜カーネルの設定値＞

カーネル共通

追加ヘッダファイル	アイドル関数	タスク優先度	チック時間	システムスタックサイズ
		8	1	1024

タスク

IDの定義名	関数名	優先度の初期値	拡張情報	(ローカル)スタックサイズ	タスク属性	共有スタック
ID_Task1	Task1	1		128	TA_HLNG TA_ACT	使用しない

セマフォ

IDの定義名	資源数の初期値	最大資源数	セマフォ属性
ID_Sem1	0	255	TA_TFIFO

イベントフラグ

IDの定義名	ビットパターン初期値(HEX)	イベントフラグ属性
ID_Flg1	0	TA_TFIFO TA_WSGL

データキュー

IDの定義名	データの個数	データキュー属性
ID_Dtq1	10	TA_TFIFO

メールボックス

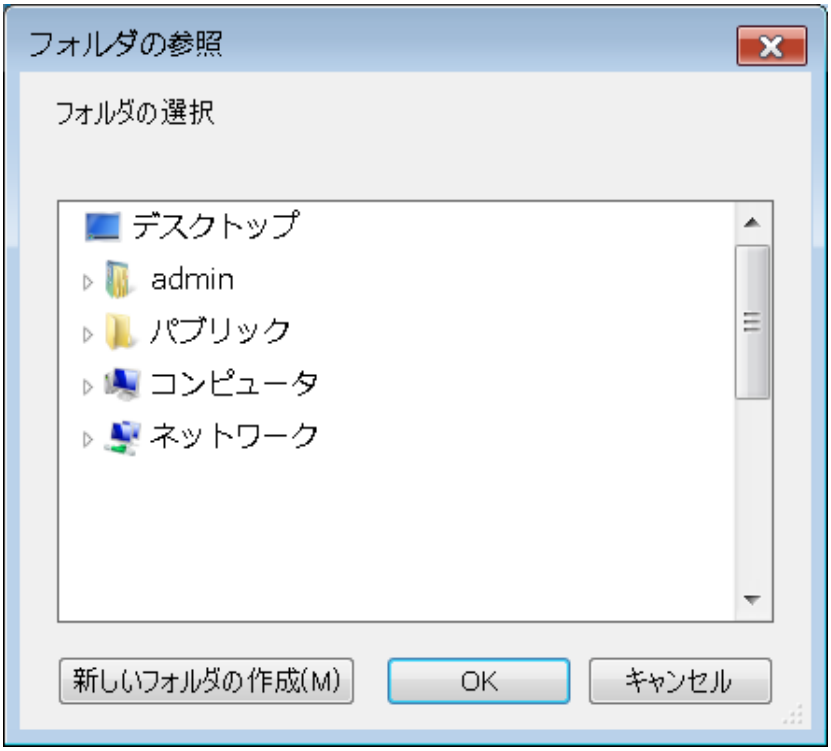
IDの定義名	メールボックス属性
ID_Mbx1	TA_TFIFO TA_MFIFO

固定長メモリプール

IDの定義名	メモリブロック数	メモリブロックのサイズ	固定長メモリプール属性
ID_Mpf1	1	64	TA_TFIFO

４．１．４　　ソース生成

「ファイル」→「ソース生成...(G)」により、「フォルダの参照画面」を開き、生成するファイルを展開する任意のフォルダを指定し、「OK」をクリックします。



スケルトンコード main.c が既に存在していた場合には、編集済みのアプリケーションファイルを上書きで誤って消去しないよう、確認のメッセージが表示されます。

【推奨】

スケルトンコードの上書きによる消去を防ぐため、スケルトンコードを直接編集せず、テンプレートとして用いてアプリケーションプログラムを作成することを推奨します。

Ａ．必ず生成されるプロセッサに依存しないファイル

ファイル	内容
kernel_id.h	オブジェクト ID、デバイス ID 等の定義ヘッダファイル
kernel_cfg.c	カーネルのコンフィグレーション情報ファイル
kernel.h	カーネルのヘッダファイル
main.c	main()、初期設定関数、タスクやハンドラなどのスケルトンコード

B．必ず生成されるプロセッサに依存したファイル

ファイル	内容
itron.h	カーネルのヘッダファイル
スタートアップ	パワーオンリセットによる初期化処理（アセンブラ言語）
ベクタテーブル	割込みベクタテーブル（アセンブラ言語）
例外ハンドラ	割込みハンドラを含めた例外ハンドラ（アセンブラ言語）
カーネルライブラリ	カーネル基本部とシステムコール群をまとめたライブラリ

C．デバイスドライバに依存したファイル

ファイル	内容
I/O 定義ファイル	プロセッサの I/O を定義したヘッダファイル
DDR_xxxxx.c	デバイスドライバのソースファイル
DDR_xxxxx.h	デバイスドライバのヘッダファイル
DDR_xxxxx_cfg.h	デバイスドライバのコンフィグレーションファイル

これらの生成されるファイルは、コンフィグレーションやプロセッサ、さらにはデバイスによっても異なります。

４．１．５　　ソース生成時のエラーチェック

ソース生成時には、以下の項目についてチェックを行います。問題がある場合には、エラーメッセージが表示され、ファイルは生成されません。

- ID や関数名など空にしてはいけない項目のチェック
- ID 総数の範囲チェック
- タスク優先度の範囲チェック
- スタックを共有するタスク間のタスク優先度と制約タスク属性の関連チェック
- セマフォの初期値の範囲チェック
- 周期ハンドラの起動周期の範囲チェック

４．１．５．１　　ID 総数

ユーザに見えない RTOS 内部で使用する ID も含め､すべてのオブジェクト ID をユニークな 8bit の値で管理しています。そのため、ID 総数の最大は 255 となり、オブジェクトを生成できる数は 255 よりも少なくなります
ID 総数は、次の計算式で算出されます。

タスク優先度上限
共有スタックの個数
タスクの個数
セマフォの個数
イベントフラグの個数
メールボックスの個数
データキューの個数の 2 倍
固定長メモリプールの個数
＋）　周期ハンドラの個数
―――――――――――――――――――――――――――――――
ID 総数

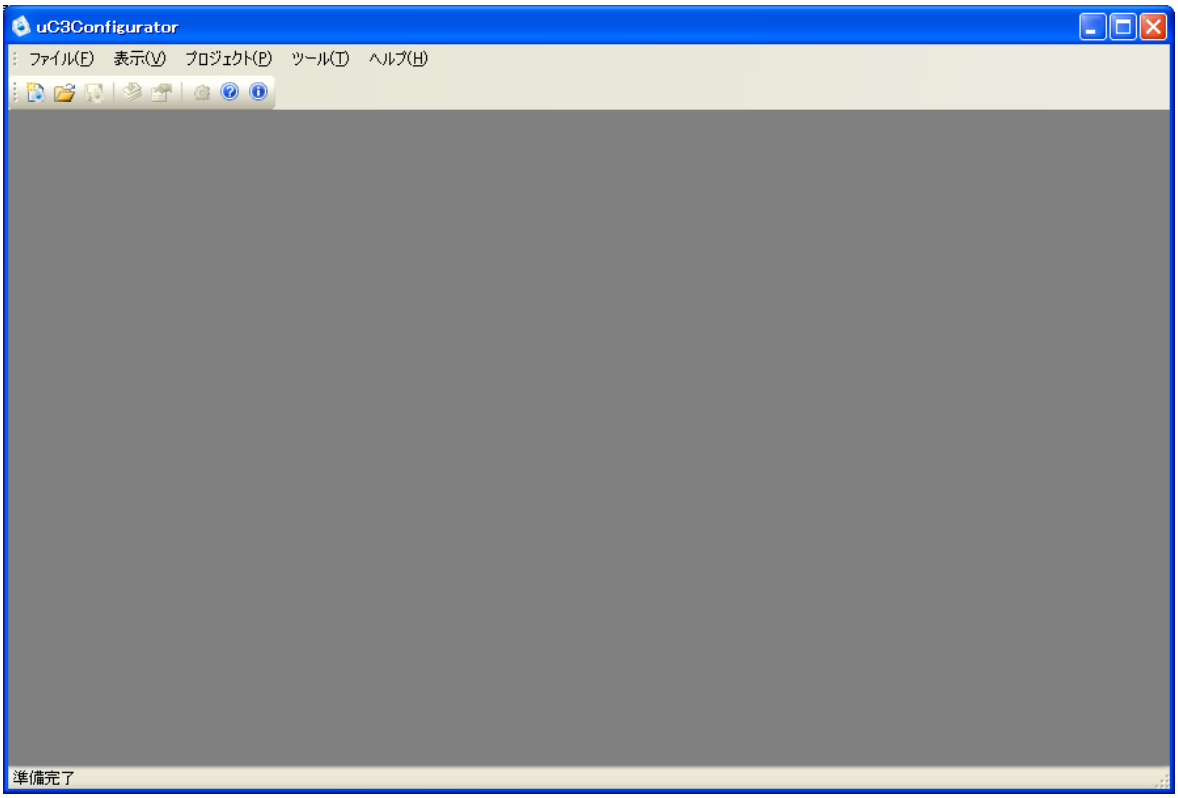
【補足】

μC3/Compact の評価版では、ID 総数を 16 に制限しています。

４．２　　現バージョンの操作説明

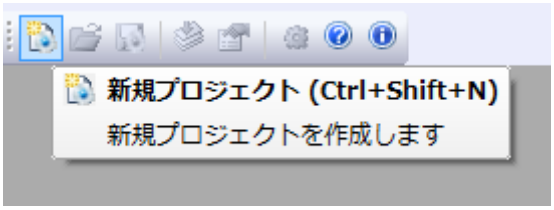
４．２．１　　コンフィグレータの起動

「Configurator.exe」をダブルクリックし、起動してください。



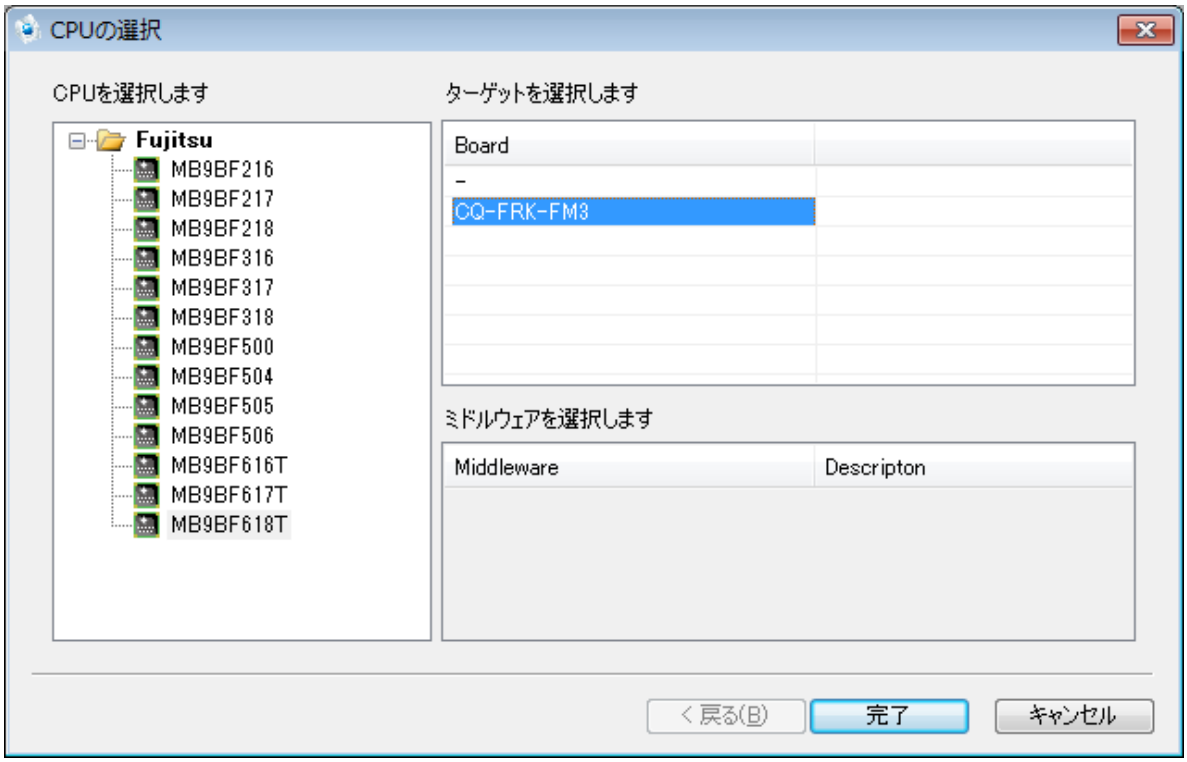
Ａ．新規でプロジェクトを生成する場合

ツールバーの「新規プロジェクト」をクリックし、「CPU の選択」へ進みます。



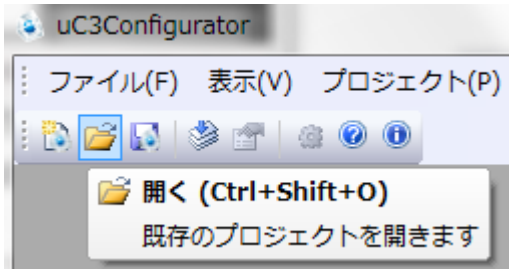
CPU 選択

リストよりベンダー、CPU 型番、ターゲットを選択後に「完了」をクリックし、「メイン画面」へ進みます。



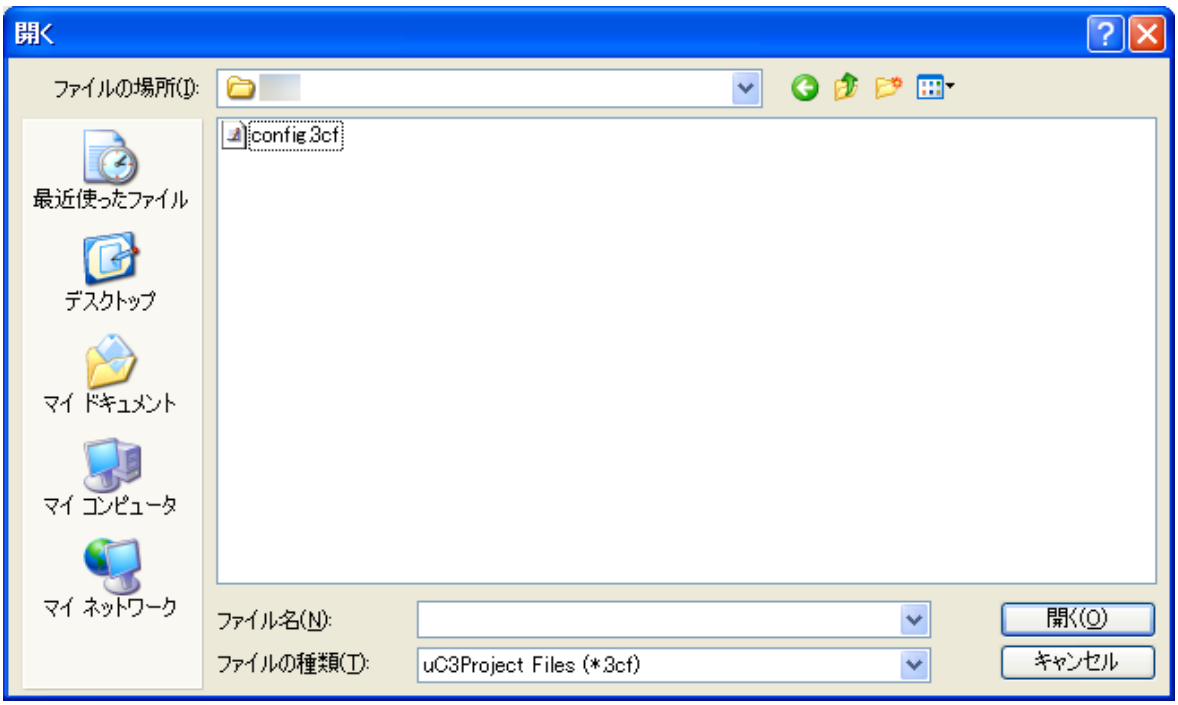
B．既存のプロジェクトを開く場合

ツールバーの「開く」をクリックし、「ファイルを開く」へ進みます。



ファイルを開く

保存されていたプロジェクトファイル（拡張子.3cf）を選択後に「開く」をクリックし、「メイン画面」へ進みます。

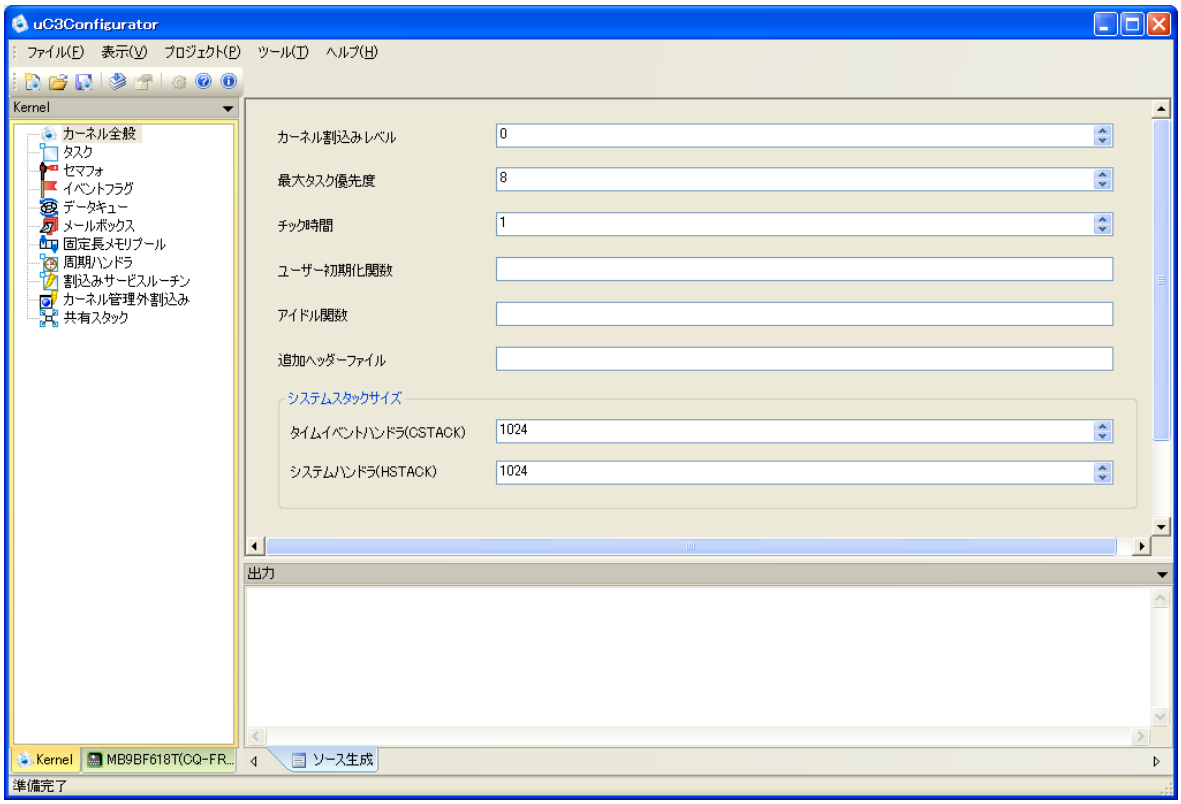


※Ver.2.x コンフィグレータで作成されたプロジェクトファイルはカーネル設定のみ読み込まれます。（CPU 設定は読込まれない）

C．メイン画面

起動後はプロジェクトの参照と編集が可能なメイン画面になります。画面左側にあるメニュー画面にあるオブジェクト名をクリックすることで各オブジェクトのコンフィグレーション画面に切り替えることができます。

ここには、カーネルやプロセッサ、デバイス依存部などのコンフィグレーションがあります。デバイス依存の説明は「プロセッサ依存部マニュアル」もしくは「デバイス依存部マニュアル」を参照してください。

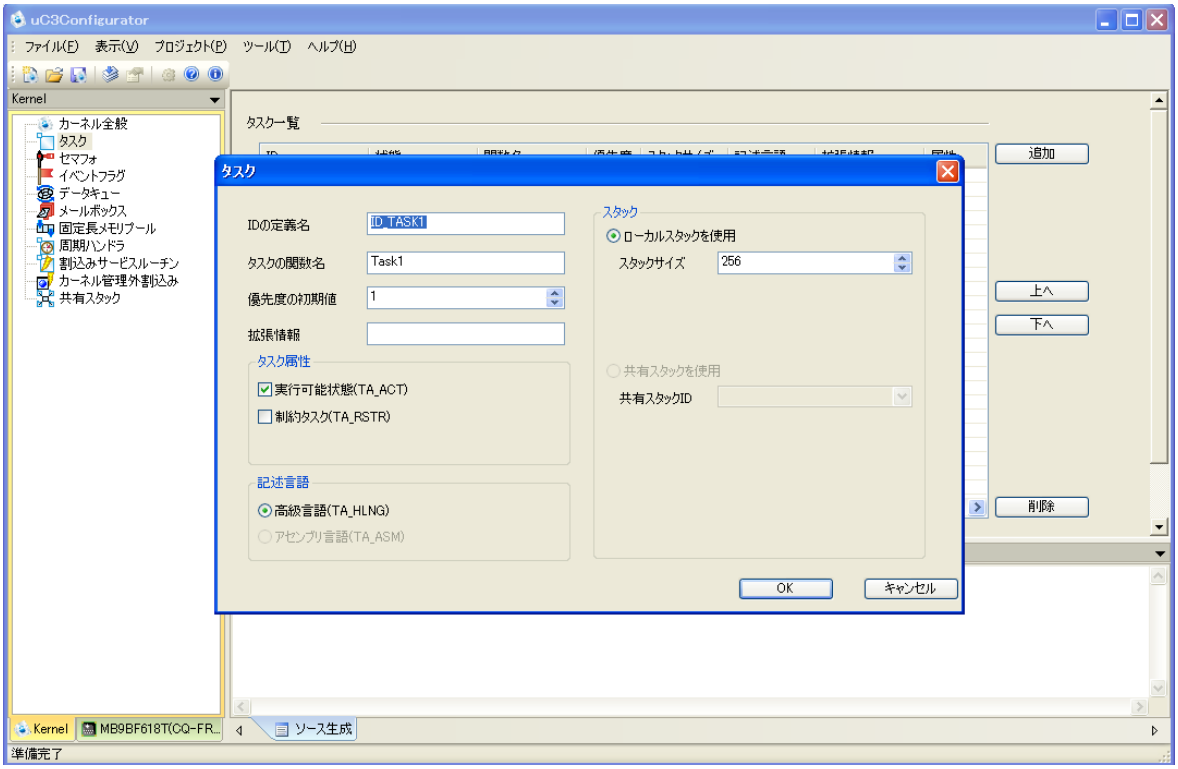


※本マニュアルでは紙面の都合上、以降の画面説明はオブジェクト選択とコンフィグレーション画面を分けた状態で説明しています。

４．２．２　　カーネルの設定

カーネルのコンフィグレーションには、カーネル全般のコンフィグレーションと、タスク、セマフォ等のオブジェクトのコンフィグレーションがあります。各種オブジェクトのコンフィグレーション画面では、１行の表示内容が１つのオブジェクトに対応しています。

以下はオブジェクトのコンフィグレーション画面の例です。



「追加」ボタン

新しいオブジェクトを追加するためのダイアログが表示されます。 必要な項目を入力・設定することでオブジェクトが追加されます。また、一覧の表示をダブルクリックすることで、オブジェクトの内容を更新することが可能となります。

「削除」ボタン

現在選択されているオブジェクトを削除します。「削除」ボタンが使用できな場合があります。このようなオブジェクトは、デバイスドライバのコンフィグレーションに連動して追加されたオブジェクトです。

「上へ」ボタン

現在選択されているオブジェクトを上へ移動します。

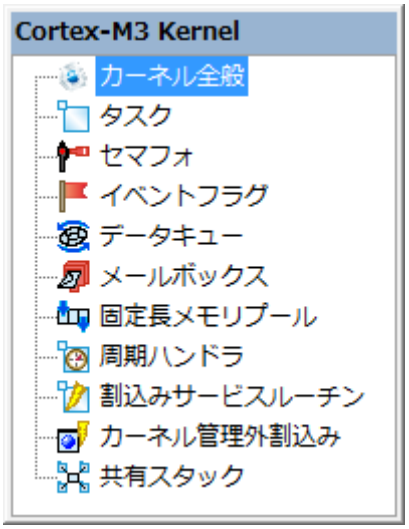
「下へ」ボタン

現在選択されているオブジェクトを下へ移動します。

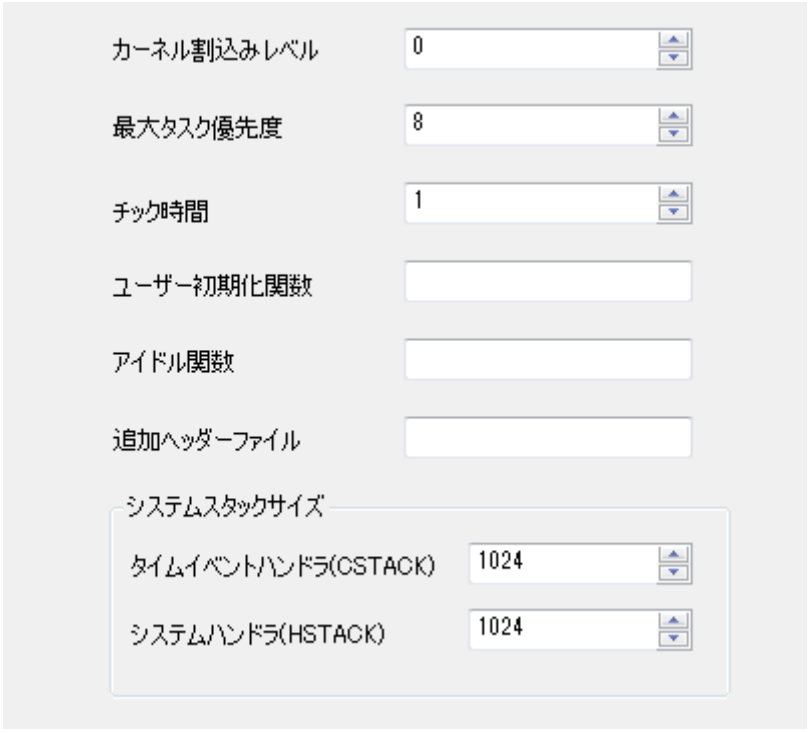
４．２．２．１　　カーネル全般のコンフィグレーション

メニュー画面のカーネル全般をクリックすると、カーネル全般のコンフィグレーション画面が表示され、ここではカーネル共通のコンフィグレーションを行います。

メニュー画面



コンフィグレーション画面



カーネル割込みレベル

カーネルが排他制御に用いる割込みレベルをカーネル割込みレベルと呼び、この割込みレベルを設定します。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。

最大タスク優先度

１から 16 までを指定でき、この値を上限とするタスク優先度を指定できます。

チック時間

タイムチックの周期をﾐﾘ秒単位で指定します。小さな値ほど時間精度は上がりますが、オーバヘッドは大きくなります。

ユーザー初期化関数

アプリケーションで初期化処理が必要な処理を記述する関数名を指定します。

アイドル関数

カーネル内部の標準アイドル関数を使わず、ユーザ定義のアイドル関数に置き換える場合には、その関数名を指定します。

追加ヘッダーファイル

タスクと周期ハンドラの拡張情報として、マクロの値か変数へのポインタを定義する場合には、マクロ定義や変数の外部宣言を記述したヘッダファイルのファイル名を指定します。具体的には、ここでファイル名を指定した場合には、そのファイルを kernel_cfg.c 内でインクルードします。

タイムイベントハンドラ(CSTACK)

アイドルと周期ハンドラで使用されるスタック領域です。サイズをバイト単位で指定します。

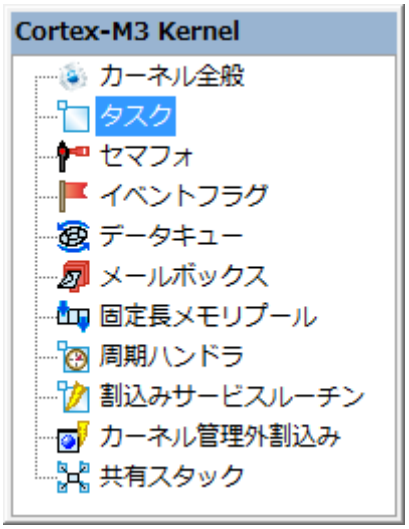
システムイベントハンドラ(HSTACK)

例外発生時や割込みサービスルーチンで使うスタック領域。サイズをバイト単位で指定します。

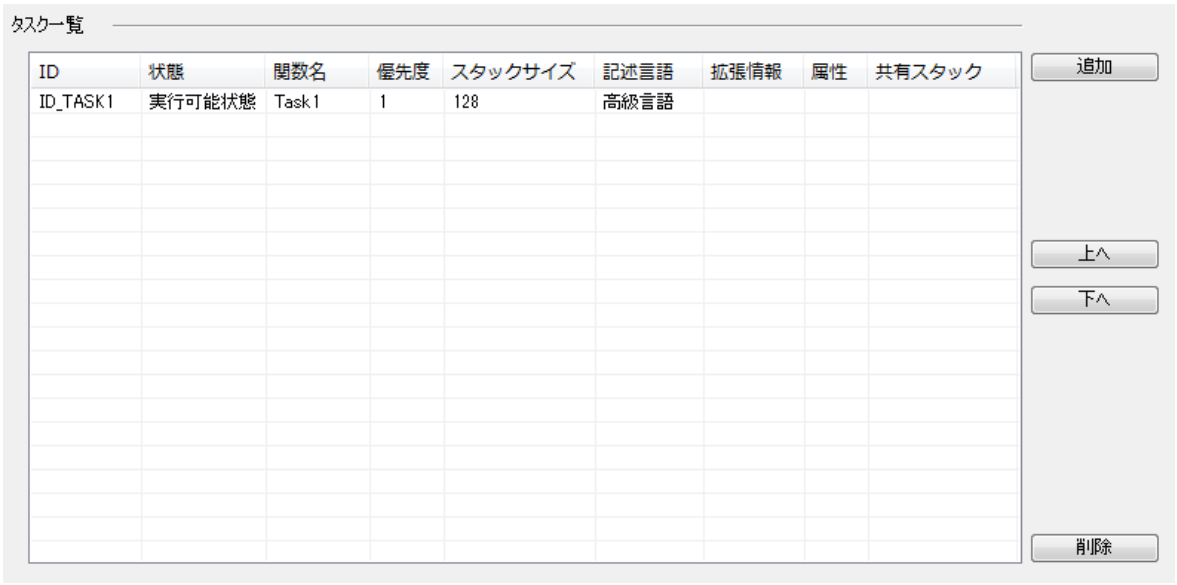
４．２．２．２　　タスクのコンフィグレーション

メニュー画面のタスクをクリックすると、タスクのコンフィグレーション画面が表示され、ここではタスク生成 API の CRE_TSK に相当するコンフィグレーションを行います。

メニュー画面



コンフィグレーション画面



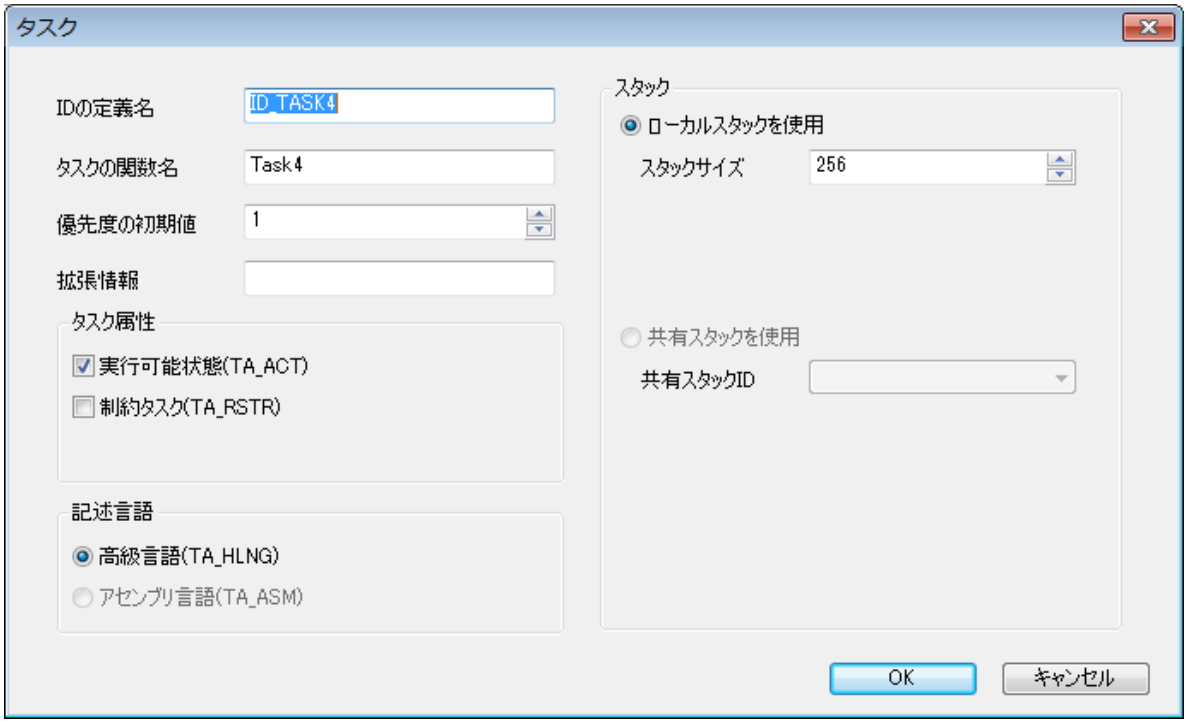
タスク一覧

現在設定されているタスクの一覧が表示されます。リスト内タスクをダブルクリックすることで編集画面が表示します。

追加

新規タスクを追加する画面が表示します。

タスク追加・編集画面



ID の定義名

タスクの ID 番号を表す任意の定義名を指定してください。この定義名は kernel_id.h 内でマクロ定義されます。

タスクの関数名

任意のタスクの関数名を指定します。

優先度の初期値

タスク起動時の初期タスク優先度を、カーネル共通のタスク優先度数を超えない値で指定してください。共有スタックを指定し制約タスクの属性を指定（TA_RSTR=ON）している場合には、その共有スタックを指定した他のタスクと同じタスク優先度を指定します。

拡張情報

タスクに渡す拡張情報がある場合には、それを指定し、不要の場合は空白のままにします。拡張情報には、数値、マクロ定義された値、変数へのポインタが指定できます。変数へのポインタを渡す場合は、変数名の先頭に「&」を付けます。

実行可能状態(TA_ACT)

チェックすると TA_ACT 属性が ON になり、タスクを実行可能状態で生成します。

制約タスク(TA_RSTR)

チェックすると TA_RSTR 属性が ON になり、制約タスクの属性を与えます。後述の共有スタックを選択した場合には、自動的にチェックされますが、その後チェックをはずすことも可能です。ある共有スタックを複数のタスクが使用する場合、それらのタスクはすべて TA_RSTR=ON、または、すべて TA_RSTR=OFF のどちらかでなければなりません。

高級言語(TA_HLNG) / アセンブリ言語(TA_ASM)

TA_HLNG 固定になり、μC3/Compact では変更できません。

ローカルスタックを使用 / 共有スタックを使用

ローカルスタック、共有スタックのどちらを使用するのかを指定します。共有スタックが 1 つ以上定義されている場合には、共有スタックを使用が選択可能になります。

スタックサイズ

タスク固有のスタックのサイズを指定してください。ローカルスタックを使用時に有効となり設定可能になります。

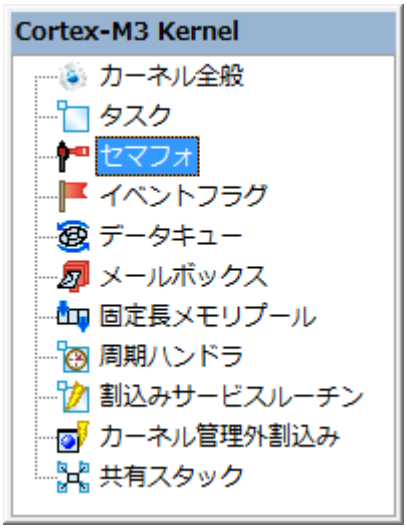
共有スタック ID

共有スタックの ID を指定します。共有スタックを使用時に有効となり設定可能になります。

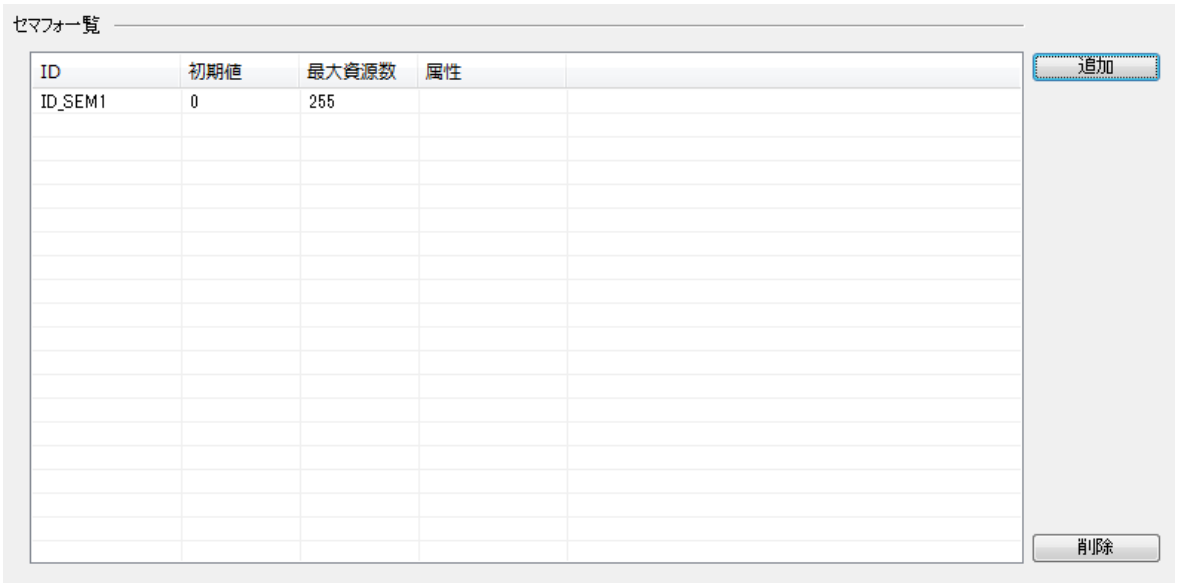
４．２．２．３　　セマフォのコンフィグレーション

メニュー画面のセマフォをクリックすると、セマフォのコンフィグレーション画面が表示され、ここではセマフォ生成 API の CRE_SEM に相当するコンフィグレーションを行います。

メニュー画面



コンフィグレーション画面



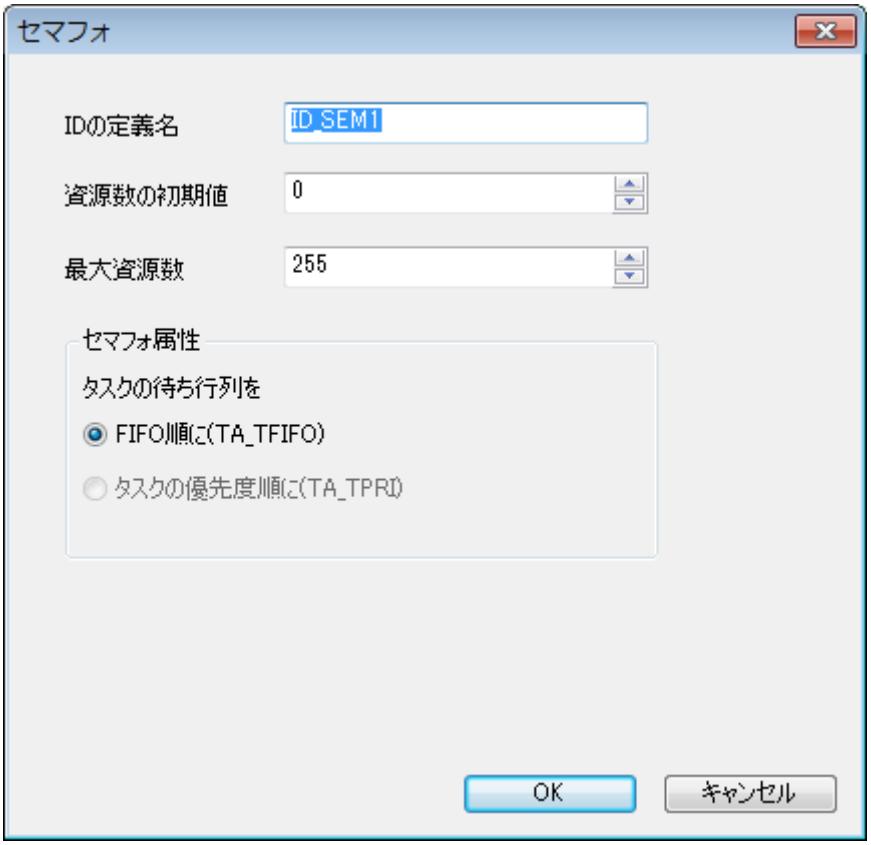
セマフォ一覧

現在設定されているセマフォの一覧が表示されます。リスト内セマフォをダブルクリックすることで編集画面が表示します。

追加

新規セマフォを追加する画面が表示します。

セマフォ追加・編集画面



ID の定義名

セマフォの ID 番号を表す任意の定義名を指定してください。この定義名は kernel_id.h 内でマクロ定義されます。

資源数の初期値

セマフォカウントの初期値を、最大資源数を超えない値を指定します。

最大資源数

セマフォカウントの最大値を指定してください。 指定できる最大値は 255 です。

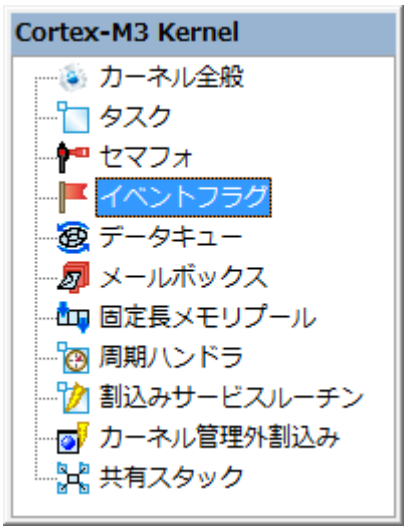
FIFO 順に(TA_TFIFO) / タスクの優先度順に(TA_TPRI)

TA_TFIFO 固定になり、μC3/Compact では変更できません。

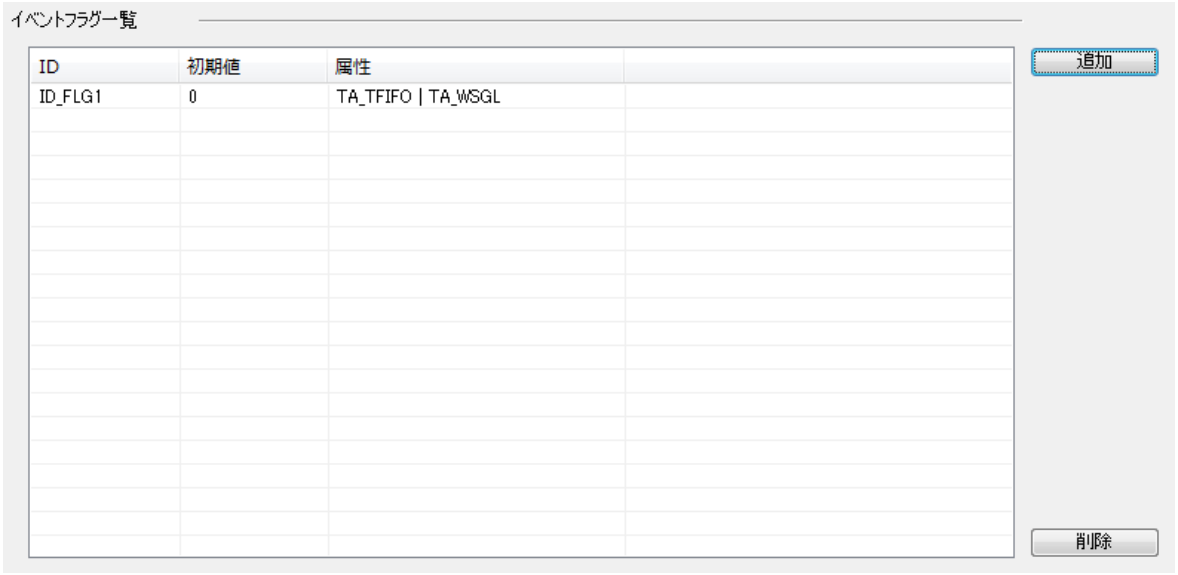
４．２．２．４　　イベントフラグのコンフィグレーション

メニュー画面のイベントフラグをクリックすると、イベントフラグのコンフィグレーション画面が表示され、ここではイベントフラグ生成 API の CRE_FLG に相当するコンフィグレーションを行います。

メニュー画面



コンフィグレーション画面



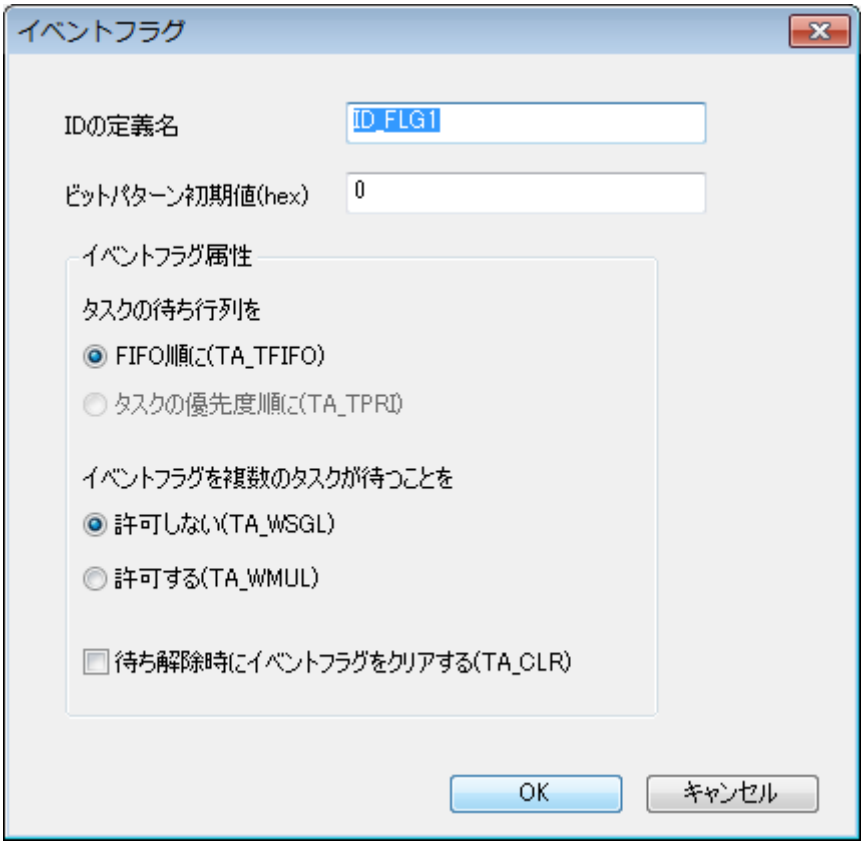
イベントフラグ一覧

現在設定されているイベントフラグの一覧が表示されます。リスト内イベントフラグをダブルクリックすることで編集画面が表示します。

追加

新規イベントフラグを追加する画面が表示します。

イベントフラグ追加・編集画面



ID の定義名

イベントフラグの ID 番号を表す任意の定義名を指定してください。この定義名は kernel_id.h 内でマクロ定義されます。

ビットパターン初期値(hex)

イベントフラグの初期値を 16 進数で指定してください。

FIFO 順に(TA_TFIFO) / タスクの優先度順に(TA_TPRI)

TA_TFIFO 固定になり、μC3/Compact では変更できません。

許可しない(TA_WSGL) / 許可する(TA_WMUL)

TA_WSGL の指定で、複数タスクの待ちが禁止されます。TA_WMUL の指定で、複数タスク待ちが許可されます。

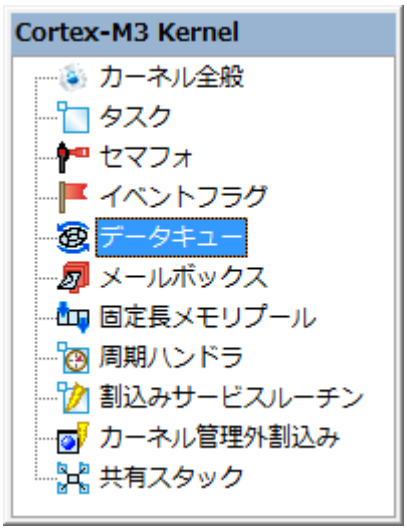
待ち解除時にイベントフラグをクリアする(TA_CLR)

チェックすると TA_CLR 属性が ON になり、タスクがイベントフラグ待ちから、条件成立により待ち解除される時に、ビットパターンのすべてのビットをクリアします。

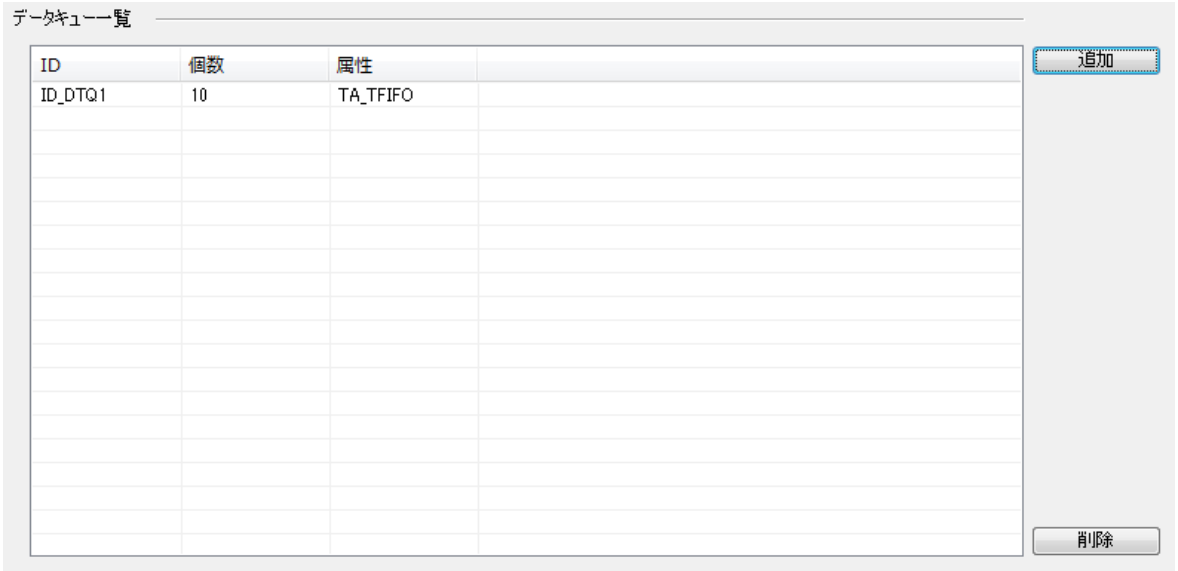
４．２．２．５　　データキューのコンフィグレーション

メニュー画面のデータキューをクリックすると、データキューのコンフィグレーション画面が表示され、ここではデータキュー生成 API の CRE_DTQ に相当するコンフィグレーションを行います。

メニュー画面



コンフィグレーション画面



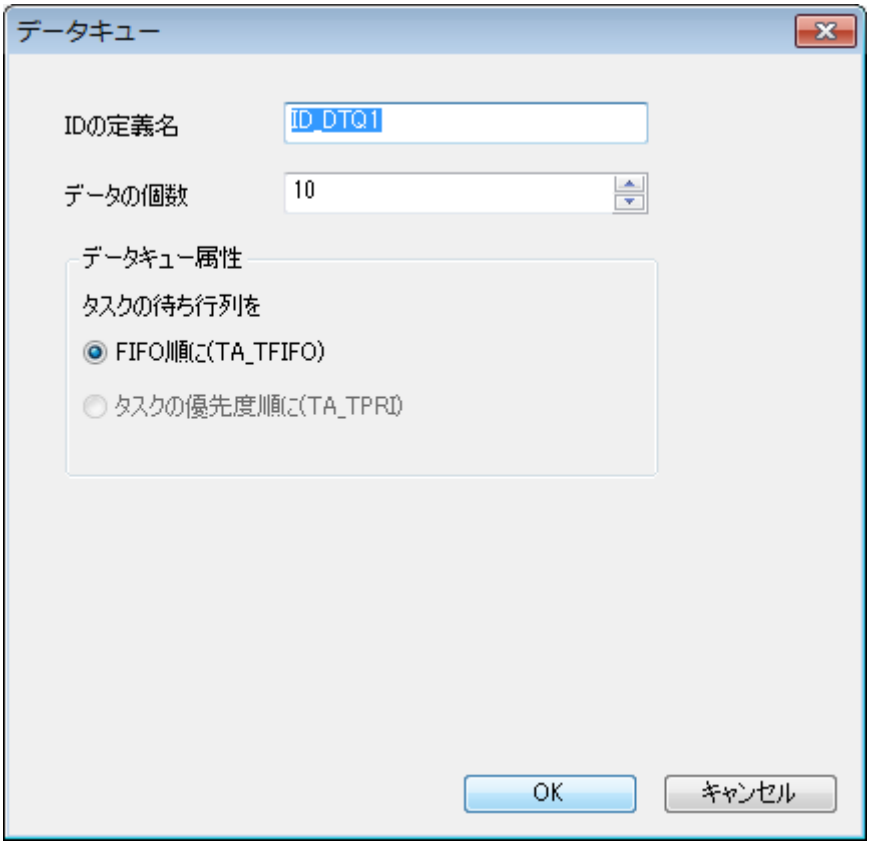
データキュー一覧

現在設定されているデータキューの一覧が表示されます。リスト内データキューをダブルクリックすることで編集画面が表示します。

追加

新規データキューを追加する画面が表示します。

データキュー追加・編集画面



ID の定義名

データキューの ID 番号を表す任意の定義名を指定してください。この定義名は kernel_id.h 内でマクロ定義されます。

データの個数

データキューの個数（データの個数）を指定します。

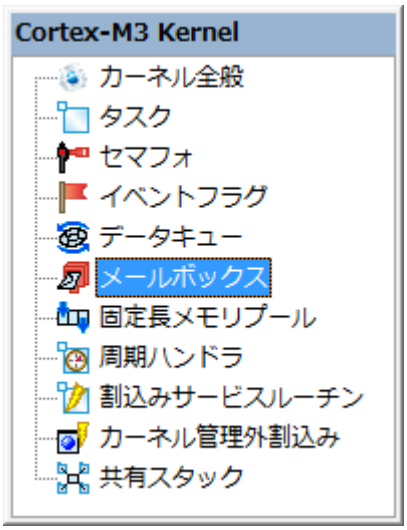
FIFO 順に(TA_TFIFO) / タスクの優先度順に(TA_TPRI)

TA_TFIFO 固定になり、μC3/Compact では変更できません。

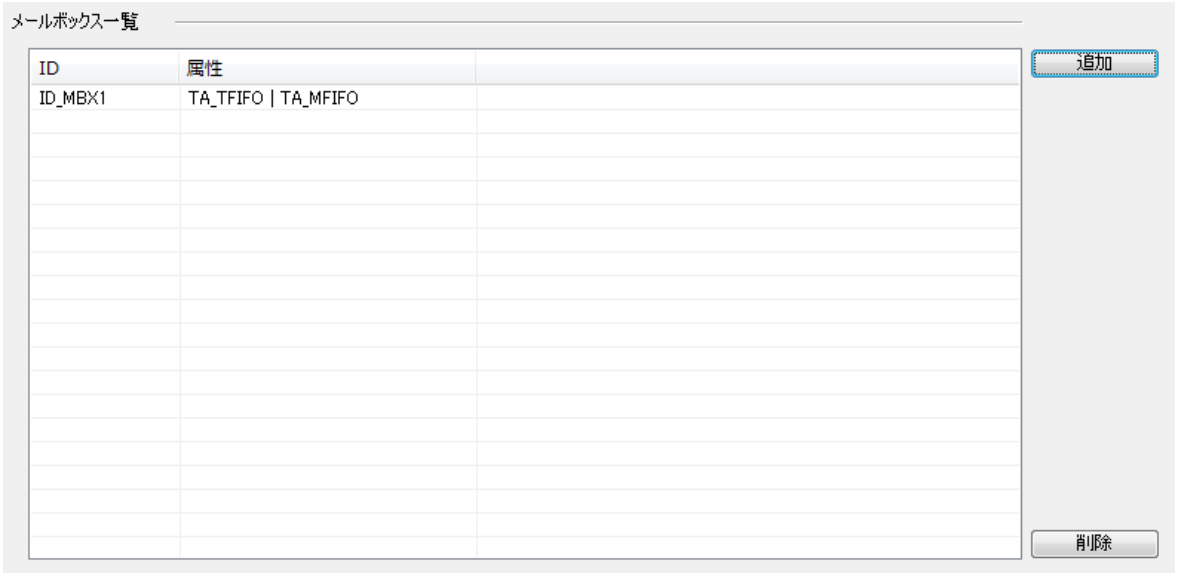
４．２．２．６　　メールボックスのコンフィグレーション

メニュー画面のメールボックスをクリックすると、メールボックスのコンフィグレーション画面が表示され、ここではメールボックス生成 API の CRE_MBX に相当するコンフィグレーションを行います。

メニュー画面



コンフィグレーション画面



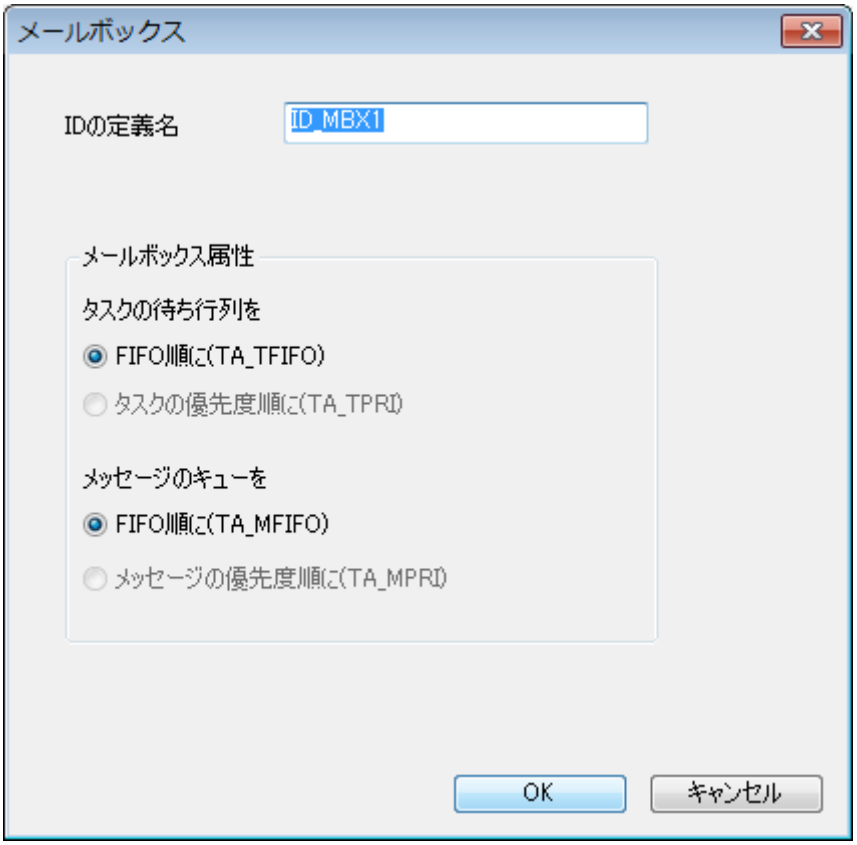
メールボックス一覧

現在設定されているメールボックスの一覧が表示されます。リスト内メールボックスをダブルクリックすることで編集画面が表示します。

追加

新規メールボックスを追加する画面が表示します。

メールボックス追加・編集画面



ID の定義名

メールボックスの ID 番号を表す任意の定義名を指定してください。この定義名は kernel_id.h 内でマクロ定義されます。

FIFO 順に(TA_TFIFO) / タスクの優先度順に(TA_TPRI)

TA_TFIFO 固定になり、μC3/Compact では変更できません。

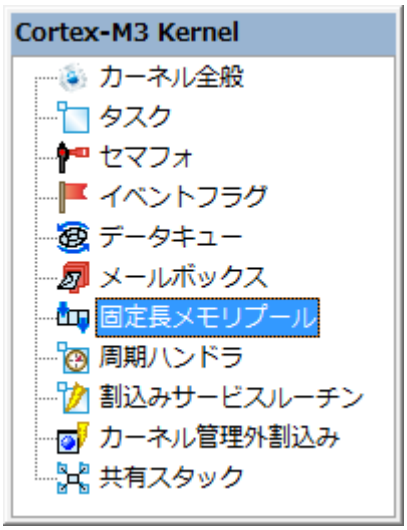
FIFO 順に(TA_MFIFO) / メッセージの優先度順に(TA_MPRI)

TA_MFIFO 固定になり、μC3/Compact では変更できません。

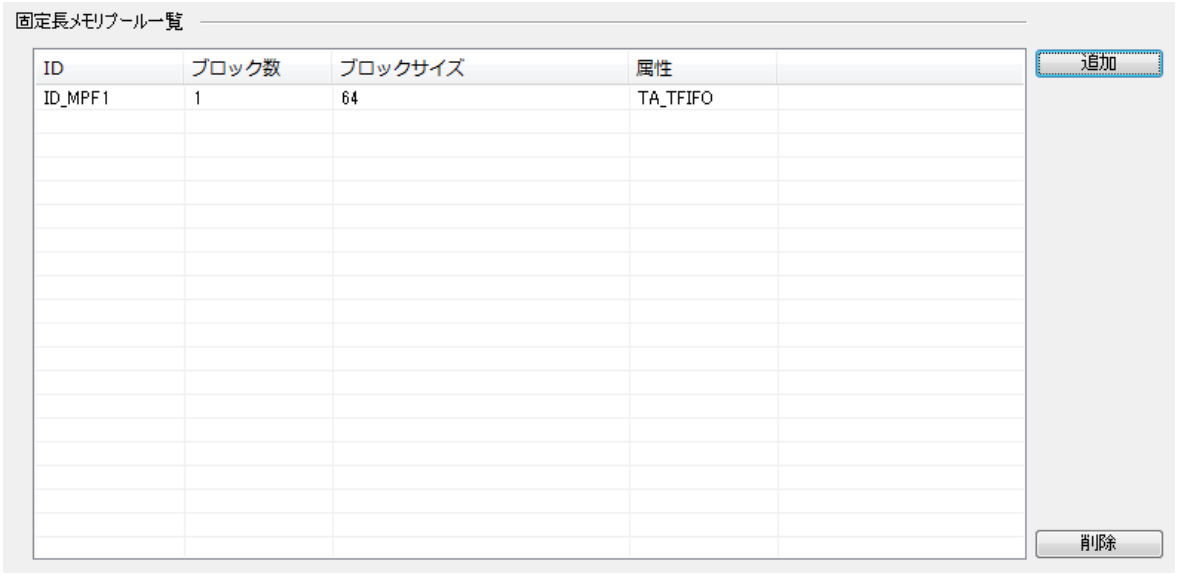
４．２．２．７　　固定長メモリプールのコンフィグレーション

メニュー画面の固定長メモリプールをクリックすると、固定長メモリプールのコンフィグレーション画面が表示され、ここでは固定長メモリプール生成 API の CRE_MPF に相当するコンフィグレーションを行います。

メニュー画面



コンフィグレーション画面



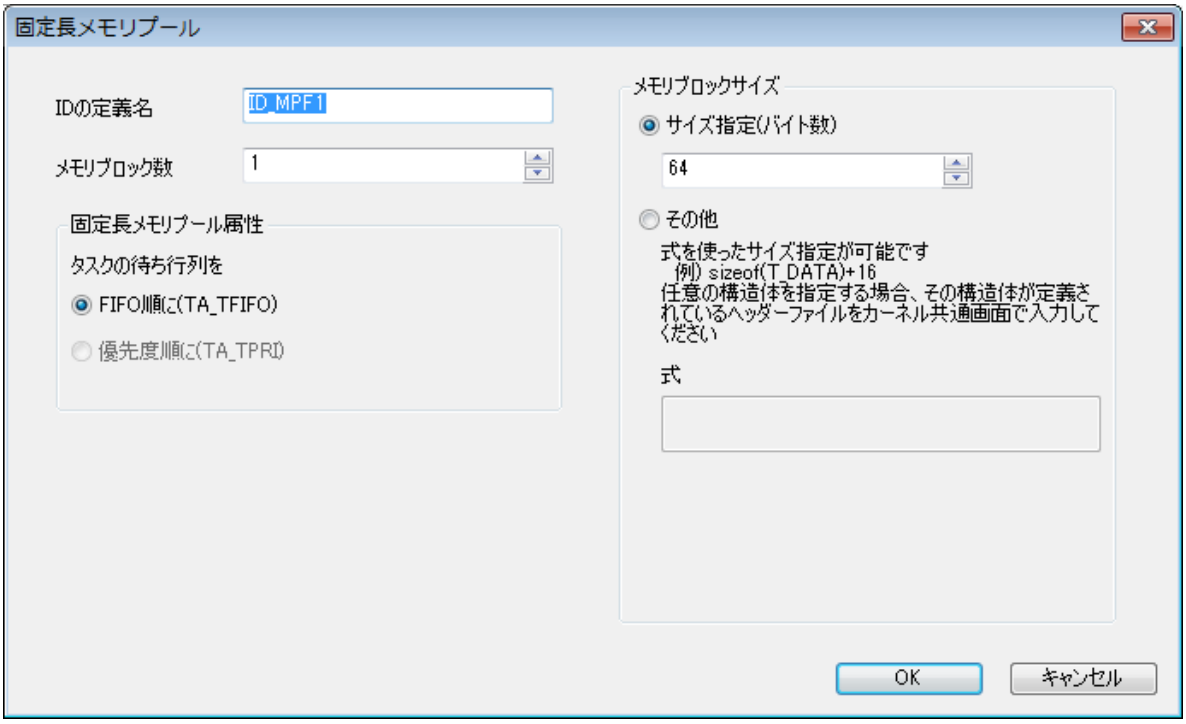
メールボックス一覧

現在設定されている固定長メモリプールの一覧が表示されます。リスト内固定長メモリプールをダブルクリックすることで編集画面が表示します。

追加

新規固定長メモリプールを追加する画面が表示します。

固定長メモリプール追加・編集画面



ID の定義名

固定長メモリプールの ID 番号を表す任意の定義名を指定してください。この定義名は kernel_id.h 内でマクロ定義されます。

メモリブロック数

メモリブロックの個数を指定します。

FIFO 順に(TA_TFIFO) / 優先度順に(TA_TPRI)

TA_TFIFO 固定になり、μC3/Compact ではタスクの優先度順に変更できません。

サイズ指定(バイト数)

メモリブロックのサイズ（バイト数）を指定します。

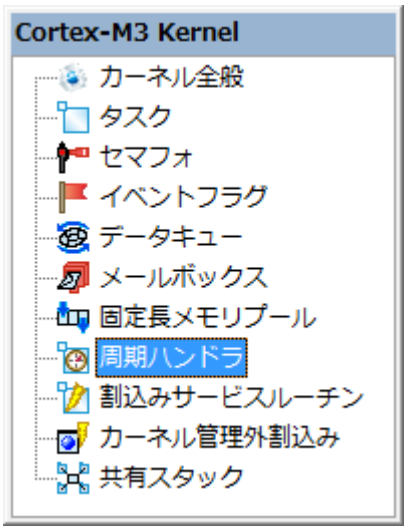
その他

sizeof 演算子、算術演算子などを用いてメモリブロックのサイズ（バイト数）を指定できます。この場合のメモリ使用量は概算値となります。また、ユーザ定義構造体を指定する場合は「カーネル全般」コンフィグレーションで構造体が定義されているヘッダーファイルを指定する必要があります。

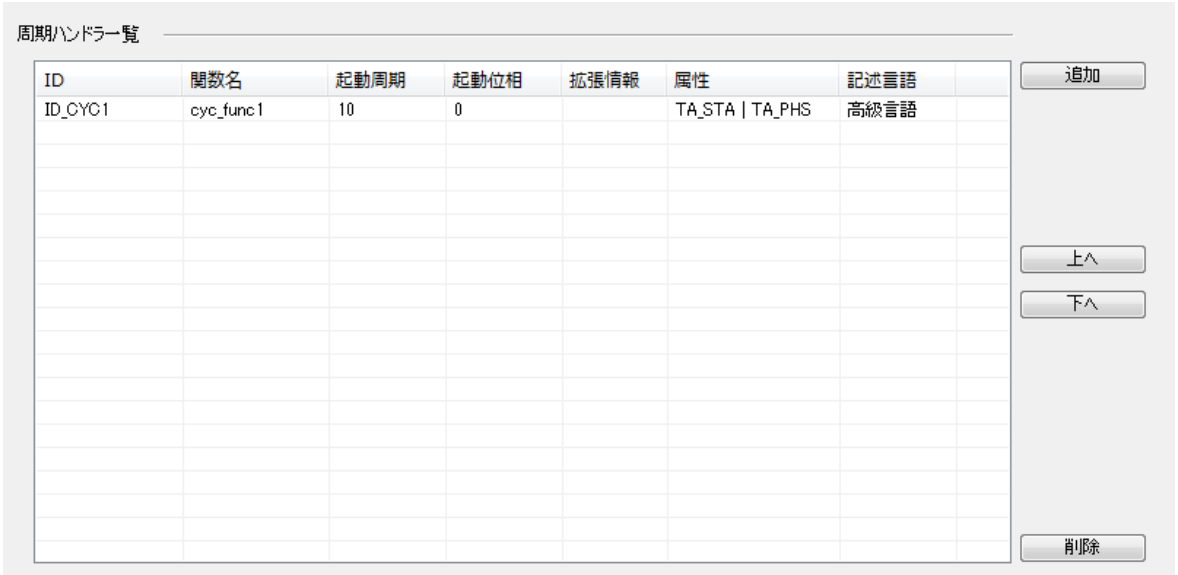
４．２．２．８　　周期ハンドラのコンフィグレーション

メニュー画面の周期ハンドラをクリックすると、周期ハンドラのコンフィグレーション画面が表示され、ここでは周期ハンドラ生成 API の CRE_CYC に相当するコンフィグレーションを行います。

メニュー画面



コンフィグレーション画面

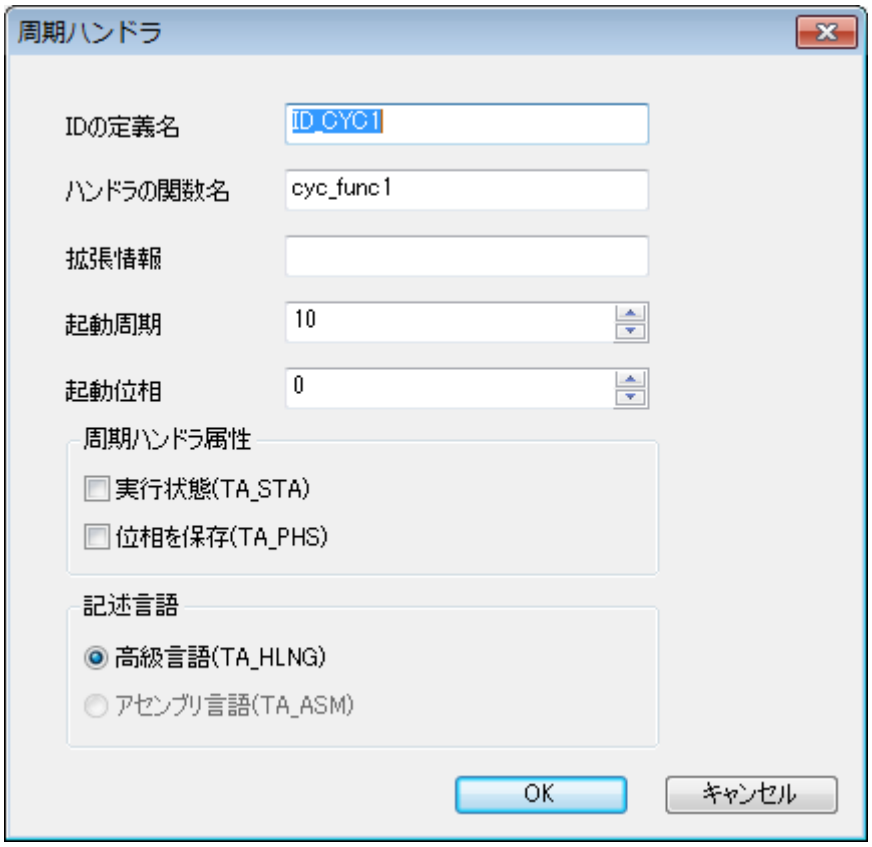


周期ハンドラ一覧

現在設定されている周期ハンドラの一覧が表示されます。 新規プロジェクト作成直後は何も表示されていません。リスト内周期ハンドラをダブルクリックすることで編集画面が表示します。

追加

新規周期ハンドラを追加する画面が表示します。



ID の定義名

周期ハンドラの ID 番号を表す任意の定義名を指定してください。この定義名は kernel_id.h 内でマクロ定義されます。

関数名

任意の周期ハンドラの関数名を指定します。

拡張情報

周期ハンドラに渡す拡張情報がある場合には、それを指定し、不要の場合は空白のままにします。拡張情報には、数値、マクロ定義された値、変数へのポインタが指定できます。変数へのポインタを渡す場合は、変数名の先頭に「&」を付けます。

起動周期

周期ハンドラの起動周期をﾐﾘ秒単で指定します。ただし、チック時間より小さい値は指定できません。

起動位相

周期ハンドラの起動位相をﾐﾘ秒単で指定します。

TA_HLNG/TA_ASM

μC3/Compact では変更できません。

周期ハンドラ属性

TA_STA

チェックすると TA_STA 属性が ON になり、周期ハンドラは動作状態で生成されます。

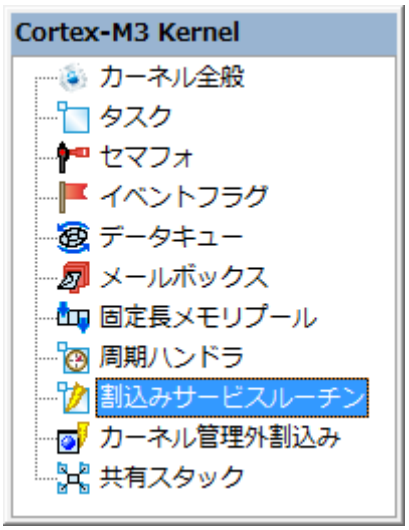
TA_PHS

チェックすると TA_PHS 属性が ON になり、周期ハンドラ生成時の位相を保存します。

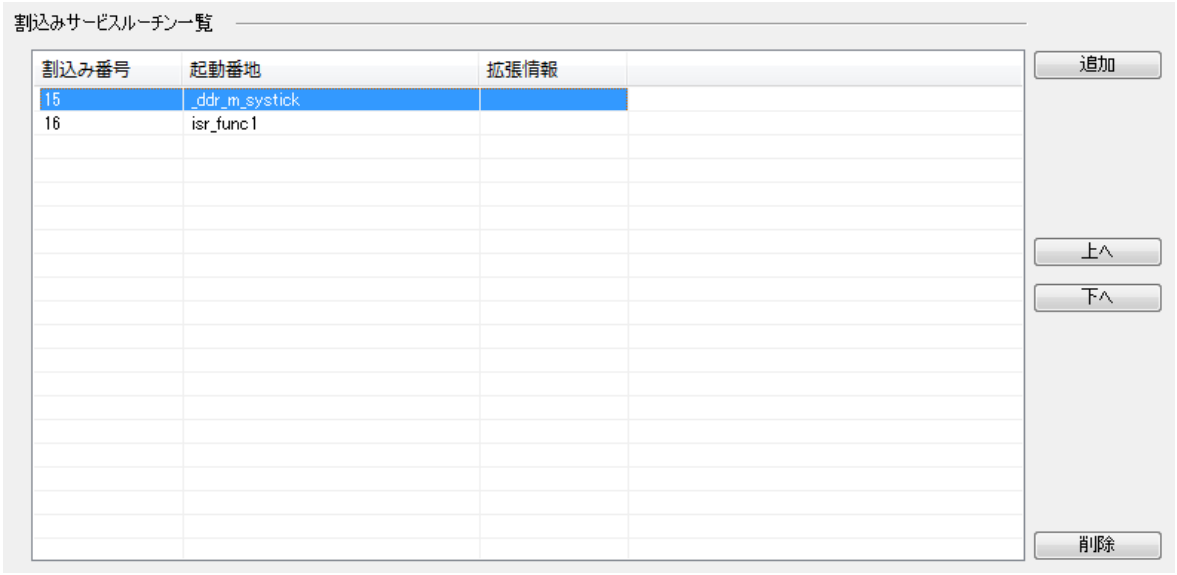
４．２．２．９　　割込みサービスルーチンのコンフィグレーション

メニュー画面の割込みサービスルーチンをクリックすると、割込みサービスルーチンのコンフィグレーション画面が表示され、ここでは割込みサービスルーチンの追加 API の ATT_ISR に相当するコンフィグレーションを行います。

メニュー画面



コンフィグレーション画面



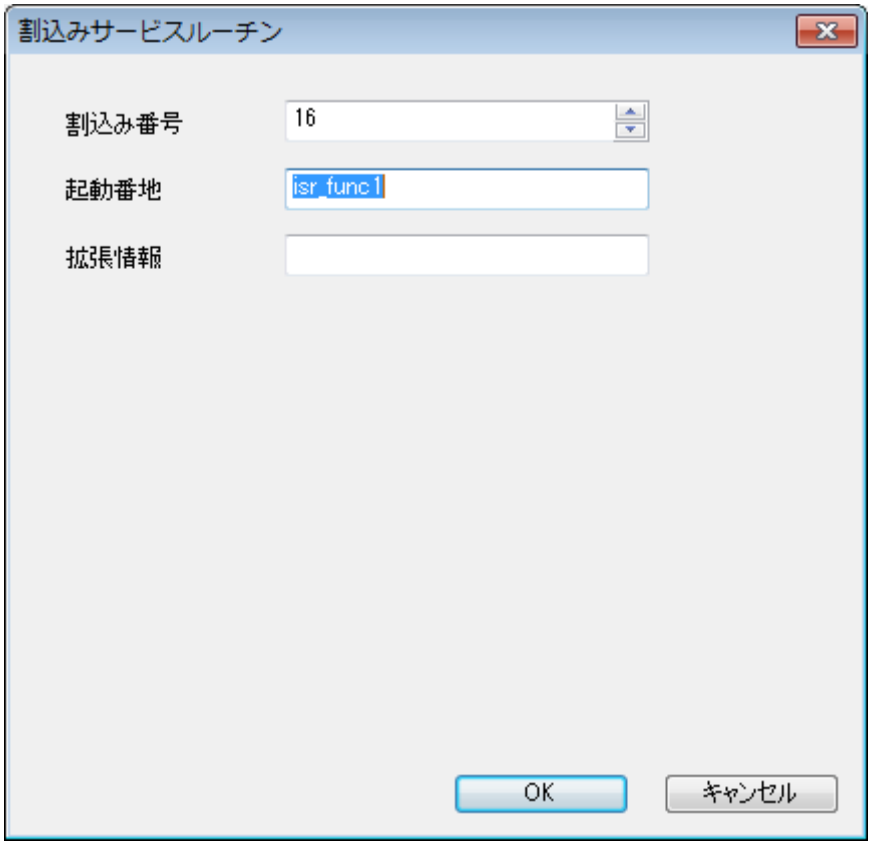
割込みサービスルーチン一覧

現在設定されている割込みサービスルーチンの一覧が表示されます。リスト内、割込みサービスルーチンをダブルクリックすることで編集画面が表示します。

追加

新規割込みサービスルーチンを追加する画面が表示します。

割込みサービスルーチン追加・編集画面



割込み番号

割込み番号を指定してください。同一の割込み番号に対して、複数の割込みサービスルーチンをコンフィグレーションした場合には、その呼出しの順序はリスト上での順序に従い、上にあるものほど早く呼ばれます。

起動番地

任意の割込みサービスルーチンの関数名を指定します。

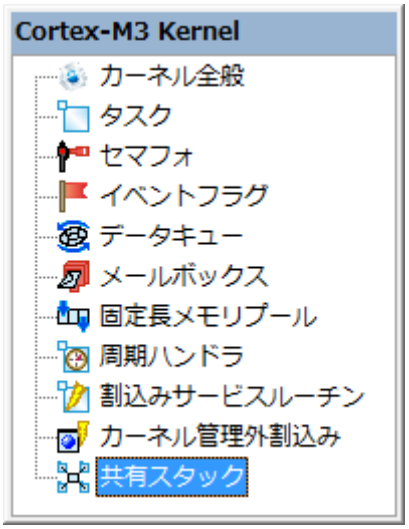
拡張情報

割込みサービスルーチンに渡す拡張情報がある場合には、それを指定し、不要の場合は空白のままにします。拡張情報には、数値、変数へのポインタが指定できます。

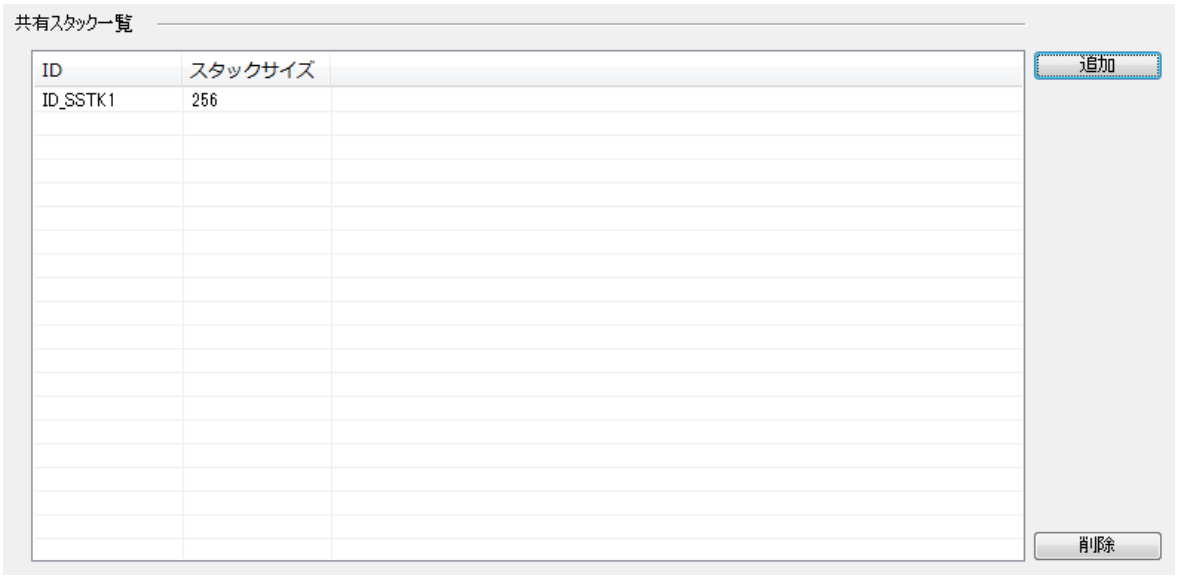
４．２．２．１０　共有スタックのコンフィグレーション

メニュー画面の共有スタックをクリックすると、共有スタックのコンフィグレーション画面が表示され、ここでは共有スタックのコンフィグレーションを行います。

メニュー画面



コンフィグレーション画面



共有スタック一覧

現在設定されている共有スタックの一覧が表示されます。リスト内共有スタックをダブルクリックすることで編集画面が表示します。

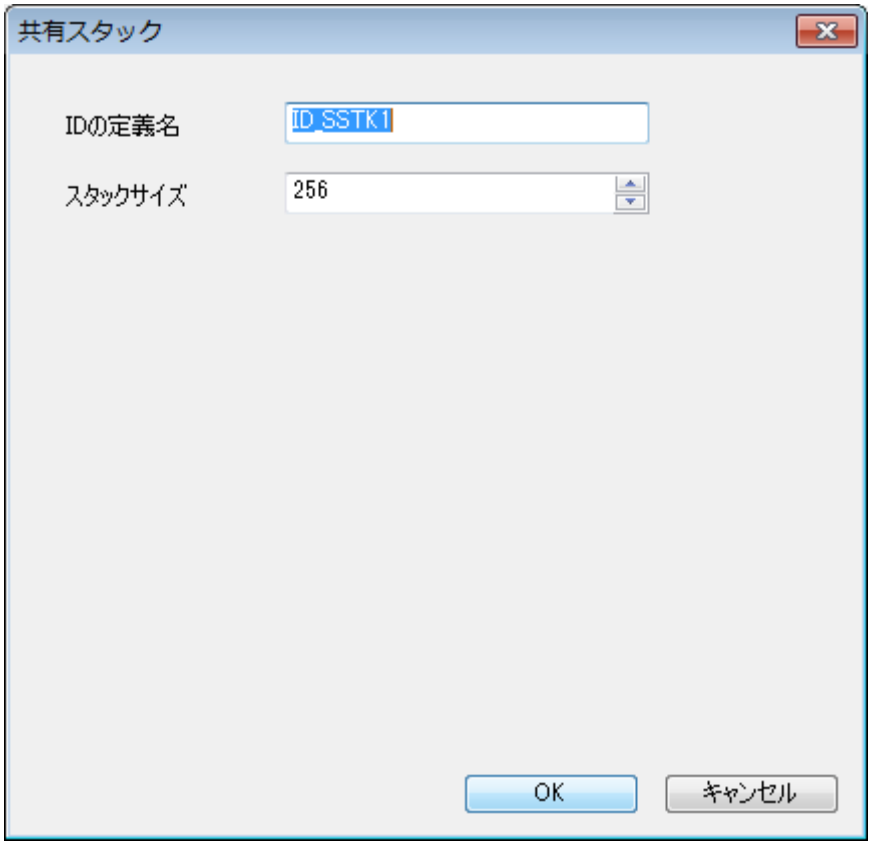
追加

新規共有スタックを追加する画面が表示します。

削除

その共有スタックを使用しているタスクが１つでもある場合、警告メッセージが表示され、削除されます。その際、タスクの共有スタックは「使用しない」状態に変更されます。

共有スタック追加・編集画面



ID の定義名

共有スタックの ID 番号を表す任意の定義名を指定してください。この定義名は、タスクのコンフィグレーション画面で共有スタックを選択するために使用されます。
この共有スタックを使用しているタスクが１つでもある場合には、定義名を変更することはできません。

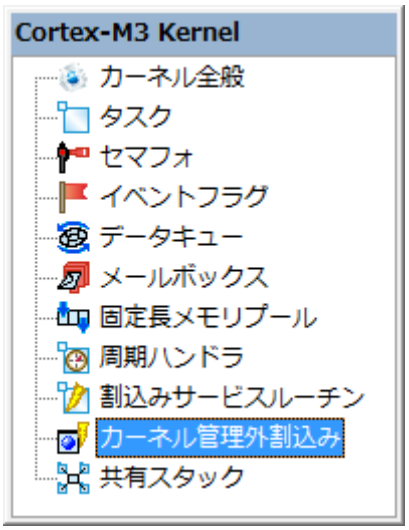
スタックのサイズ

共有スタックのサイズ（バイト数）を指定します。共有スタックの使用を選択したタスクのスタックサイズは、共有スタックのサイズに固定されます。そのため、この共有スタックを指定したタスクで、一番多くスタックを使用するタスクのスタックサイズを指定します。

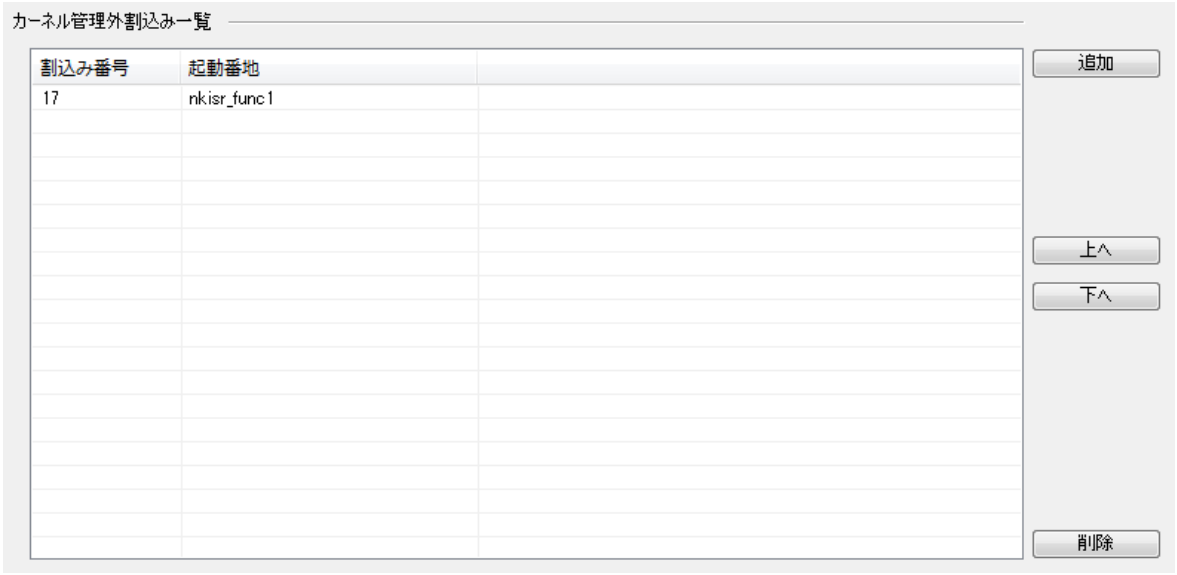
４．２．２．１１　カーネル管理外割込みのコンフィグレーション

メニュー画面のカーネル管理外割込みをクリックすると、カーネル管理外割込みのコンフィグレーション画面が表示され、ここではカーネル管理外割込みのコンフィグレーションを行います。

メニュー画面



コンフィグレーション画面



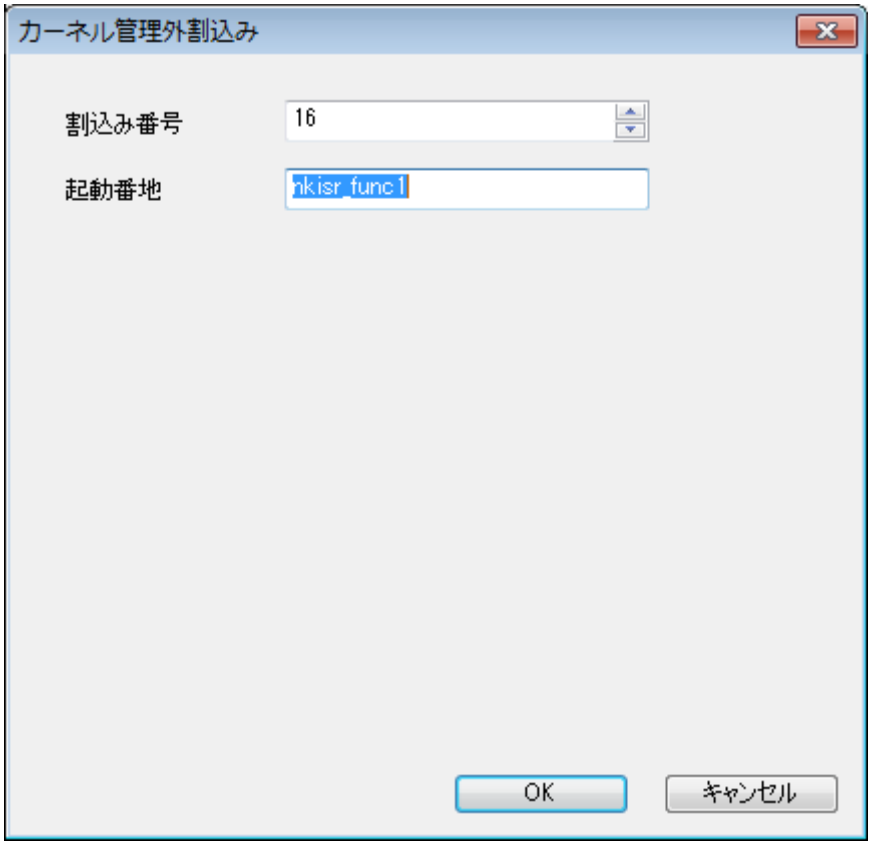
カーネル管理外割込み一覧

現在設定されているカーネル管理外割込みの一覧が表示されます。リスト内、カーネル管理外割込みをダブルクリックすることで編集画面が表示します。

追加

新規カーネル管理外割込みを追加する画面が表示します。

カーネル管理外割込み追加・編集画面



割込み番号

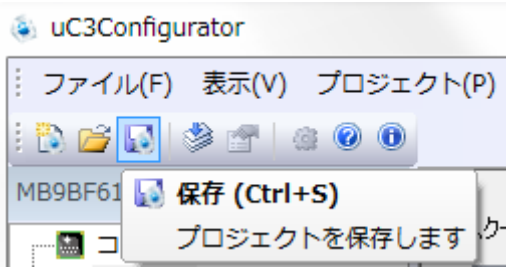
割込み番号を指定してください。同一の割込み番号に対して、カーネル管理外割込みを登録することは出来ません。また、指定した割込み番号が「割込みサービスルーチン」で使用済みの場合は登録することは出来ません。

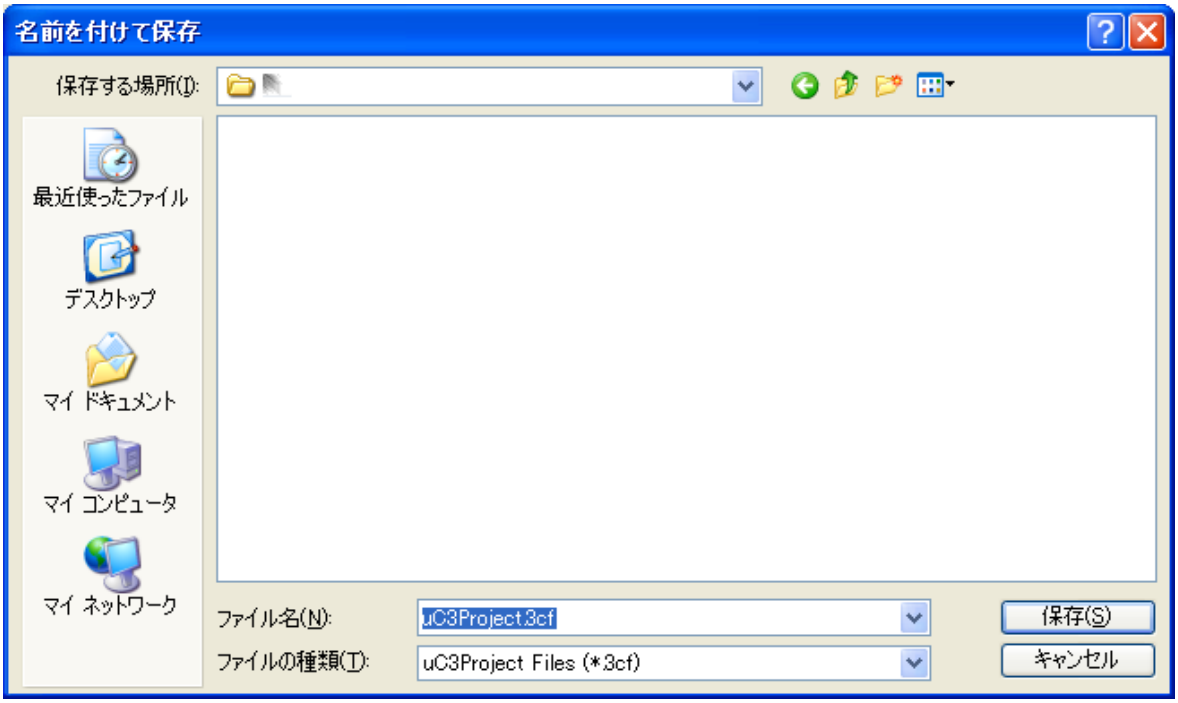
起動番地

任意のカーネル管理外割込みの関数名を指定します。

４．２．３　　プロジェクトファイルの保存

ツールバーの「保存」をクリックし、「名前を付けて保存画面」を開き、プロジェクトファイルの保存先フォルダを指定し、「OK」をクリックします。





保存されるファイルは、プロジェクトファイル（デフォルト config..3cf）と拡張子を「xml」に変えたファイルが保存されます。

このファイルをブラウザで開くことにより、コンフィグレーション情報を確認することができます。

uC3/Configurator プロジェクトファイルの設定値一覧

[使用プラグイン]

ファイル名
D:\20120523_NewConfig\uC3\Configurator\Compact\Kernel\ARM\CortexM3\uC3CmpCortexM3.plugin
D:\20120523_NewConfig\uC3\Configurator\Compact\CPU\Fujitsu\MB9BXXX\uC3CmpCpuMB9Bxxxx.plugin

---

[カーネルの設定値]

カーネル全般

カーネル割込みレベル	タスク優先度	チック時間	初期化関数	アイドル関数	追加ヘッダファイル	タイムイベントハンドラスタックサイズ(CSTACK)	システムハンドラスタックサイズ(HSTACK)
0	8	1				1024	1024

タスク

IDの定義名	関数名	優先度の初期値	拡張情報	(ローカル)スタックサイズ	タスク属性	共有スタック
ID_TASK1	Task1	1		256	TA_HLNG | TA_ACT	
ID_TASK2	Task2	1		256	TA_HLNG | TA_ACT	

セマフォ

IDの定義名	資源数の初期値	最大資源数	セマフォ属性

イベントフラグ

IDの定義名	ビットパターン初期値(HEX)	イベントフラグ属性

データキュー

IDの定義名	データの個数	データキュー属性

メールボックス

IDの定義名	メールボックス属性

固定長メモリプール

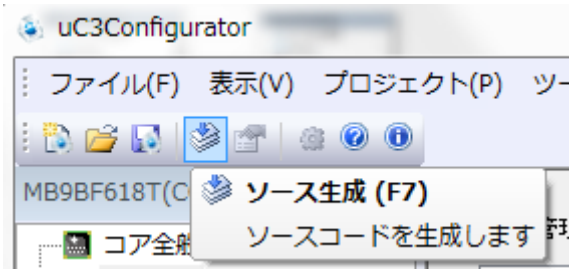
IDの定義名	メモリブロック数	メモリブロックのサイズ	固定長メモリプール属性

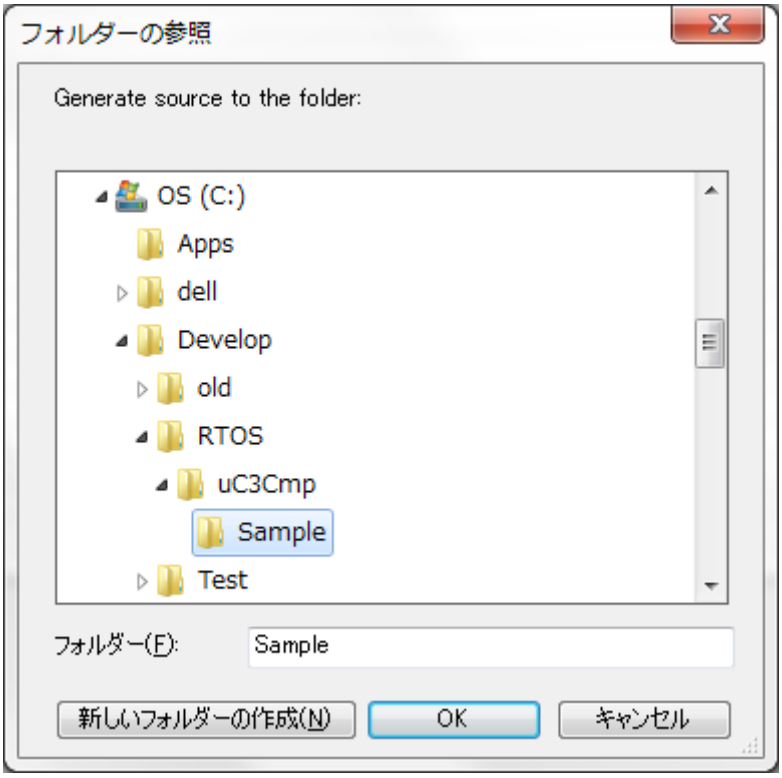
周期ハンドラ

IDの定義名	関数名	拡張情報	起動周期	起動位相	周期ハンドラ属性

４．２．４　　ソース生成

ツールバーの「ソース生成」をクリックし、「フォルダの参照画面」を開き、生成するファイルを展開する任意のフォルダを指定し、「OK」をクリックします。





スケルトンコード main.c が既に存在していた場合には、以前のファイルは main.bak としてバックアップされます。

【推奨】

スケルトンコードの上書きによる消去を防ぐため、スケルトンコードを直接編集せず、テンプレートとして用いてアプリケーションプログラムを作成することを推奨します。

Ａ．必ず生成されるプロセッサに依存しないファイル

ファイル	内容
kernel_id.h	オブジェクト ID の定義ヘッダファイル
kernel_cfg.c	カーネルのコンフィグレーション情報ファイル
kernel.h	カーネルのヘッダファイル
main.c	main()、初期設定関数、タスクやハンドラなどのスケルトンコード

Ｂ．必ず生成されるプロセッサに依存したファイル

ファイル	内容
itron.h	カーネルのヘッダファイル
hw_init.c	ハードウェア、デバイス依存部の初期化ファイル
hw_dep.h	ハードウェア、デバイスに依存したヘッダファイル
スタートアップ	パワーオンリセットによる初期化処理（アセンブラ言語）
ベクタテーブル	割込みベクタテーブル（アセンブラ言語）
例外ハンドラ	割込みハンドラを含めた例外ハンドラ（アセンブラ言語）
カーネルライブラリ	カーネル基本部とシステムコール群をまとめたライブラリ

Ｃ．デバイスドライバに依存したファイル

ファイル	内容
I/O 定義ファイル	プロセッサの I/O を定義したヘッダファイル
device_id.h	デバイス ID の定義ヘッダファイル
DDR_xxxxx.c	デバイスドライバのソースファイル
DDR_xxxxx.h	デバイスドライバのヘッダファイル
DDR_xxxxx_cfg.h	デバイスドライバのコンフィグレーションファイル

これらの生成されるファイルは、コンフィグレーションやプロセッサ、さらにはデバイスによっても異なります。

４．２．５　ソース生成時のエラーチェック

ソース生成時には、以下の項目についてチェックを行います。問題がある場合には、エラーメッセージが表示され、ファイルは生成されません。

- ID や関数名など空にしてはいけない項目のチェック
- ID 総数の範囲チェック
- タスク優先度の範囲チェック
- スタックを共有するタスク間のタスク優先度と制約タスク属性の関連チェック
- セマフォの初期値の範囲チェック
- 周期ハンドラの起動周期の範囲チェック
- カーネル割込みレベルと、割込みサービスルーチンの割込みレベルのチェック

４．２．５．１　ID 総数

ユーザに見えない RTOS 内部で使用する ID も含め、すべてのオブジェクト ID をユニークな 8bit の値で管理しています。そのため、ID 総数の最大は 255 となり、オブジェクトを生成できる数は 255 よりも少なくなります

ID 総数は、次の計算式で算出されます。

タスク優先度上限
共有スタックの個数
タスクの個数
セマフォの個数
イベントフラグの個数
メールボックスの個数
データキューの個数の 2 倍
固定長メモリプールの個数
＋）　周期ハンドラの個数
―――――――――――――――――――――――――――――――
ID 総数

【補足】

μC3/Compact の評価版では、ID 総数を 16 に制限しています。

第５章　システムコールの説明

５．１　　タスク管理機能

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_QOVR	キューイングオーバフロー（起動要求キューイング数のオーバフロー)

【呼び出しコンテキスト】	act_tsk	iact_tsk
タスク	可	可
タイムイベントハンドラ	可	可
割込みサービスルーチン	可	可

【解説】

tskid で指定されるタスクを起動します。具体的には、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 を加えます。ただし、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。

tskid には、コンフィグレータで生成した際のタスク ID の定義名を使って指定します。また、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】

μC3/Compact 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）

【呼び出しコンテキスト】

タスク	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】

tskid で指定されるタスクに対し、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 には、コンフィグレータで生成した際のタスク ID の定義名を使って指定します。また、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。

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_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、または対象タスクが共有スタックを使っている）

【呼び出しコンテキスト】

タスク	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】

tskid で指定されるタスクを起動します。具体的には、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、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。対象タスクが、共有スタックを指定している場合も、E_OBJ エラーを返します。

tskid には、コンフィグレータで生成した際のタスク ID の定義名を使って指定します。

【推奨】

sta_tsk は、μITRON3.0 仕様との互換性のために存在しています。μC3／Compact では、共有スタック使用時に制限もあり、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

【戻値】

なし

【呼び出しコンテキスト】

タスク	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】

自タスクを終了させます。具体的には、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。

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_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）

【呼び出しコンテキスト】

タスク	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】

tskid で指定されるタスクを強制的に休止状態に遷移させます。

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。

tskid には、コンフィグレータで生成した際のタスク ID の定義名を使って指定します。

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_NOSPT	対象タスクが制約タスク属性
E_PAR	パラメータエラー（tskpri が不正）
E_OBJ	オブジェクト状態エラー（対象タスクが休止状態）

【呼び出しコンテキスト】

タスク	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】

tskid で指定されるタスクの現在優先度を、tskpri で指定される値に変更します。 tskid には、コンフィグレータで生成した際のタスク ID の定義名を使って指定します。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、対象タスクの現在優先度を、タスクの起動時優先度に変更します。

対象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の間では、対象タスクの優先順位を最低にします。

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_OBJ	オブジェクト状態エラー（対象タスクが休止状態）

【呼び出しコンテキスト】

タスク	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】

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

tskid には、コンフィグレータで生成した際のタスク ID の定義名を使って指定します。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（μC3/Compact は未対応）
VB const*	name	Ver.2.x カーネルで予約（Ver.1.x カーネルでは未使用）

【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）

【呼び出しコンテキスト】

タスク	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】

tskid で指定される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_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。μC3/Compact では、常に、現在優先度とベース優先度は同じになります。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_MPF	0x2000	固定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返します。

対象タスクが休止状態でない場合に、wupcnt には対象タスクの起床要求キューイング数、suscnt には強制待ち要求ネスト数を返します。対象タスクが休止状態の場合には不定値を返します。

tskid には、コンフィグレータで生成した際のタスク ID の定義名を使って指定します。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）

【呼び出しコンテキスト】

タスク	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】

tskid で指定される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 の簡易版です。tskstat と tskwait には、ref_tsk で返すのと同じ値を返します。

tskid には、コンフィグレータで生成した際のタスク ID の定義名を使って指定します。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_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
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】

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/Compact 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_OBJ	オブジェクト状態エラー（対象タスクが休止状態）
E_QOVR	キューイングオーバフロー（起床要求キューイング数のオーバフロー）

【呼び出しコンテキスト】	wup_tsk	iwup_tsk
タスク	可	可
タイムイベントハンドラ	可	可
割込みサービスルーチン	可	可

【解説】

tskid で指定されるタスクを、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。また、休止状態の場合には、E_OBJ エラーとなります。

tskid には、コンフィグレータで生成した際のタスク ID の定義名を使って指定します。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】

μC3/Compact 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_OBJ	オブジェクト状態エラー（対象タスクが休止状態）

【呼び出しコンテキスト】

タスク	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】

tskid で指定されるタスクに対し、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返します。

tskid には、コンフィグレータで生成した際のタスク ID の定義名を使って指定します。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_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）

【呼び出しコンテキスト】	rel_wai	irel_wai
タスク	可	可
タイムイベントハンドラ	可	可
割込みサービスルーチン	可	可

【解説】

tskid で指定される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。

tskid には、コンフィグレータで生成した際のタスク ID の定義名を使って指定します。

【推奨】

μC3/Compact 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。

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_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
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】

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　同期・通信機能

５．３．１　セマフォ

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_QOVR	キューイングオーバフロー（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
タイムイベントハンドラ	可	可
割込みサービスルーチン	可	可

【解説】

semid で指定される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。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。

semid には、コンフィグレータで生成した際のセマフォ ID の定義名を使って指定します。

【推奨】

μC3/Compact 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_PAR	パラメータエラー（tmout が不正 ; twai_sem のみ）
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）

【呼び出しコンテキスト】	**wai_sem**	**pol_sem**	**twai_sem**
タスク	可	可	可
タイムイベントハンドラ	不可	可	不可
割込みサービスルーチン	不可	不可	不可

【解説】

semid で指定されるセマフォから、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 のままにし、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。他のタスクがすでに待ち行列につながっている場合、自タスクを待ち行列の末尾につなぎます。

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。μC3/Compact 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。

semid には、コンフィグレータで生成した際のセマフォ ID の定義名を使って指定します。

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）

【呼び出しコンテキスト】

タスク	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】

semid で指定される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。

semid には、コンフィグレータで生成した際のセマフォ ID の定義名を使って指定します。

５．３．２　イベントフラグ

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_PAR	パラメータエラー（setptn が不正）

【呼び出しコンテキスト】	set_flg	iset_flg
タスク	可	可
タイムイベントハンドラ	可	可
割込みサービスルーチン	可	可

【解説】

flgid で指定される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 の値のビット毎の論理和（OR）により更新します。対象イベントフラグのビットパターンを更新し、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かれば、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、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、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、 続けて待ち行列を調べます。

flgid は、コンフィグレータで生成した際のイベントフラグ ID の定義名を使って指定します。

【推奨】

μC3/Compact 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_NOEXS	オブジェクト未生成（対象イベントフラグが未登録）
E_PAR	パラメータエラー（clrptn が不正）

【呼び出しコンテキスト】

タスク	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】

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

flgid には、コンフィグレータで生成した際のイベントフラグ ID の定義名を使って指定します。

【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, ~0x0001);
```

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   *pflgptn, 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_ID	不正 ID 番号（flgid が不正、あるいは使用できない）
E_PAR	パラメータエラー（waiptn, wfmode が不正）
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_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）

【呼び出しコンテキスト】	**wai_flg**	**pol_flg**	**twai_flg**
タスク	可	可	可
タイムイベントハンドラ	不可	可	不可
割込みサービスルーチン	不可	不可	不可

【解説】

flgid で指定されるイベントフラグのビットパターンが、waiptn 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。waiptn 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_WSGL 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。waiptn と wfmode で指定される待ち解除条件とは、wfmode に TWF_ANDW が指定された場合には、 対象イベントフラグのビットパターンの waiptn 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 が指定された場合には、対象イベントフラグのビットパターンの waiptn 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。μC3/Compact 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。

flgid には、コンフィグレータで生成した際のイベントフラグ ID の定義名を使って指定します。

【使い方】

```
FLGPTN waiptn;          /* 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, 0x0003, TWF_ORW, &waiptn);
if (ercd == E_OK) {
    if ((waiptn & 0x0001) != 0) {
        ercd = clr_flg(ID_Flag1, ~0x0001);
    }
}
```

ref_flg　　イベントフラグの状態参照

【書式】

ER ercd = ref_fig(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）

【呼び出しコンテキスト】

タスク	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】

flgid で指定される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。

flgid には、コンフィグレータで生成した際のイベントフラグ ID の定義名を使って指定します。

５．３．３　データキュー

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_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）

【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
タイムイベントハンドラ	不可	可	可	不可
割込みサービスルーチン	不可	可	可	不可

【解説】

dtqid で指定される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 エラーを返します。

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。μC3/Compact 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。

dtqid には、コンフィグレータで生成した際のデータキューID の定義名を使って指定します。

【推奨】

μC3/Compact 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）

【呼び出しコンテキスト】	**fsnd_dtq**	**ifsnd_dtq**
タスク	可	可
タイムイベントハンドラ	可	可
割込みサービスルーチン	可	可

【解説】

dtqid で指定される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いれます。つまり、一番古いデータを削除します。これらのシステムコールで、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 エラーを返します。

dtqid には、コンフィグレータで生成した際のデータキューID の定義名を使って指定します。

【推奨】

μC3/Compact 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_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）

【呼び出しコンテキスト】	**rcv_dtq**	**prcv_dtq**	**trcv_dtq**
タスク	可	可	可
タイムイベントハンドラ	可	可	不可
割込みサービスルーチン	不可	不可	不可

【解説】

dtqid で指定される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。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。μC3/Compact 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。

dtqid には、コンフィグレータで生成した際のデータキューID の定義名を使って指定します。

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）

【呼び出しコンテキスト】

タスク	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】

dtqid で指定される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。

sdqcnt 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。

dtqid には、コンフィグレータで生成した際のデータキューID の定義名を使って指定します。

５．３．４　メールボックス

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）

【呼び出しコンテキスト】

タスク	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】

mbxid で指定される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の末尾に入れます。この時、送信するメッセージパケットは、既にメールボックスのメッセージキューにながれていてはなりません。

mbxid には、コンフィグレータで生成した際のメールボックス ID の定義名を使って指定します。

【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 = 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);
}
```

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_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）

【呼び出しコンテキスト】	**rcv_mbx**	**prcv_mbx**	**trcv_mbx**
タスク	可	可	可
タイムイベントハンドラ	不可	可	不可
割込みサービスルーチン	不可	不可	不可

【解説】

mbxid で指定される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。μC3/Compact 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。

mbxid には、コンフィグレータで生成した際のメールボックス ID の定義名を使って指定します。

【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確認する */
                    /* メモリブロックの番地をパラメータにして呼び出す */
    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）

【呼び出しコンテキスト】

タスク	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】

mbxid で指定される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。

mbxid には、コンフィグレータで生成した際のメールボックス ID の定義名を使って指定します。

５．４　　メモリプール管理機能

５．４．１　　固定長メモリプール

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_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）

【呼び出しコンテキスト】	**get_mpf**	**pget_mpf**	**tget_pmf**
タスク	可	可	可
タイムイベントハンドラ	不可	可	不可
割込みサービスルーチン	不可	不可	不可

【解説】

mpfid で指定される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。μC3/Compact 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。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。

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）

【呼び出しコンテキスト】

タスク	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】

mpfid で指定される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。

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。

ref_mpf　　固定長メモリプールの状態参照

【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プールの ID 番号
T_RMPF*	pk_rmpf	固定長メモリプール状態を返すパケットへのポインタ

【戻値】

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）

【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）

【呼び出しコンテキスト】

タスク	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】

mpfid で指定される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。

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　時間管理機能

５．５．１　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
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】

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、既に呼び出されたシステムコールのタイムアウト時刻に変更を与えません。

get_tim　　システム時刻の参照

【書式】

ER ercd = get_tim(SYSTIM　*p_systim)；

【パラメータ】

なし

【戻値】

ER	ercd	正常終了（E_OK）またはエラーコード
SYSTIM	systim	現在のシステム時刻

【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
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】

現在のシステム時刻を呼び出し、systim 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	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	可

【解説】

システム時刻にチック時間を加算します。

５．５．２　　周期ハンドラ

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）

【呼び出しコンテキスト】

タスク	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】

cycid で指定される周期ハンドラに TA_PHS 属性が指定されていない場合には、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、既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。

cycid には、コンフィグレータで生成した際の周期ハンドラ ID の定義名を使って指定します。

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_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
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】

cycid で指定される周期ハンドラが動作している状態の場合には、動作していない状態に移行させます。動作していない状態の場合には、何もしません。

cycid には、コンフィグレータで生成した際の周期ハンドラ ID の定義名を使って指定します。

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）

【呼び出しコンテキスト】

タスク	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】

cycid で指定される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。

cycid には、コンフィグレータで生成した際の周期ハンドラ ID の定義名を使って指定します。

５．６　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_NOSPT	対象優先度の優先順位の先頭タスクが制約タスク属性

【呼び出しコンテキスト】	rot_rdq	irot_rdq
タスク	可	可
タイムイベントハンドラ	可	可
割込みサービスルーチン	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とします。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とします。

【推奨】

μC3/Compact 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】

E_PAR	パラメータエラー（p_tskid が不正）

【呼び出しコンテキスト】	**get_tid**	**iget_tid**
タスク	可	可
タイムイベントハンドラ	可	可
割込みサービスルーチン	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】

μC3/Compact 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
タイムイベントハンドラ	可	可
割込みサービスルーチン	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】

μC3/Compact 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
タイムイベントハンドラ	可	可
割込みサービスルーチン	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しません。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】

μC3/Compact 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
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】

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しません。

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
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】

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しません。

sns_ctx　　コンテキストの参照

【書式】

BOOL state = sns_ctx( ) ;

【パラメータ】

なし

【戻値】

BOOL	state	コンテキスト

【呼び出しコンテキスト】

タスク	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】

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

【戻値】

BOOL	state	CPU ロック状態

【呼び出しコンテキスト】

タスク	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
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
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

特記すべきフィールドはない

【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
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】

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。

【補足】

本マニュアル作成時点では、システムの状態を参照する情報はありません。

５．７　　割込み管理機能

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
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レベルを、imask で指定される値に変更します。このシステムコールは、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。

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
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を参照し、imask に返します。

５．８　システム構成管理機能

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	tick	タイムチックの周期時間
UH	tskpri_max	タスク優先度の上限
UH	id_max	最大の 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】

タスク	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で指定された情報を参照し、pk_rcfg で指定されるパケットに返します。

tick には、チック時間として指定した、タイムチックの周期時間を返します。

tskpri_max には、タスク優先度数として指定した、タスク優先度の上限を返します。

id_max には、システム内部で使用している ID 番号で、最大の ID 番号を返します。

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
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、0x000 を返します。

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

μC3/Compact 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。

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_ID	不正 ID 番号（DevID が不正、あるいは使用できない）
E_PAR	パラメータエラー

【解説】

DecID で指定されるデバイスを、初期化情報パケットの内容で初期化します。DevID には、コンフィグレータで生成した際のデバイス ID の定義名を使って指定します。

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。

BLEN8	８ビットデータ長
BLEN7	７ビットデータ長
BLEN6	６ビットデータ長
BLEN5	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_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
E_PAR	パラメータエラー

【解説】

DecID で指定されるデバイスを、command で指定された内容の制御を行います。DevID には、コンフィグレータで生成した際のデバイス ID の定義名を使って指定します。

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】

DecID で指定されるデバイスから、送信文字 chr を送信します。DevID には、コンフィグレータで生成した際のデバイス ID の定義名を使って指定します。

tmout は、送信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】

DecID で指定されるデバイスから、送信文字列 schr を送信文字数 scnt だけ送信します。DevID には、コンフィグレータで生成した際のデバイス ID の定義名を使って指定します。

tmout は、送信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】

DecID で指定されるデバイスから、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。DevID には、コンフィグレータで生成した際のデバイス ID の定義名を使って指定します。

tmout は、送信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】

DecID で指定されるデバイスから、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。DevID には、コンフィグレータで生成した際の、デバイス ID の定義名を使って指定します。

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、送信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】

DecID で指定される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

μITRON4.0 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）

SYSTEM	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）
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
INTNO	割込み番号
IMASK	割込みマスク

【補足】

INT,UINT,VP_INT,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         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 */
        VB const*   name            /* Ver.2.x カーネルで予約 */
} 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        tskwait ;       /* 待ち要因 */
} T_RTST ;
```

(2) 同期・通信機能

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 ;
```

(3) メモリプール管理機能

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 ;
```

(4) 時間管理機能

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 ;
```

(5) 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 {
        /* フィールドはない */
} T_RSYS ;
```

（6）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      tick        タイムチックの周期時間
        UH      tskpri_max  タスク優先度の上限
        UH      id_max      最大の ID 番号
} 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) 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

(3) 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

(4) 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_MPF	0x2000	固定長メモリブロックの獲得待ち状態
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

（5）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）

(2) バージョン情報

TKERNEL_MAKER	カーネルのメーカコード
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
TMAX_WUPCNT	タスクの起床要求キューイング数の最大値

(4) ビットパターンのビット数

TBIT_FLGPTN	イベントフラグのビット数

(5) その他

TMAX_MAXSEM	セマフォの最大資源数の最大値

７．５　エラーコード一覧

E_SYS	-5	0xFFFFFFFB	システムエラー
E_NOSPT	-9	0xFFFFFFF7	未サポート機能
E_RSFN	-10	0xFFFFFFF6	予約機能コード
E_RSATR	-11	0xFFFFFFF5	予約属性
E_PAR	-17	0xFFFFFFEF	パラメータエラー
E_ID	-18	0xFFFFFFEE	不正 ID 番号
E_CTX	-25	0xFFFFFFE7	コンテキストエラー
E_MACV	-26	0xFFFFFFE6	メモリアクセス違反
E_OACV	-27	0xFFFFFFE5	オブジェクトアクセス違反
E_ILUSE	-28	0xFFFFFFE4	サービスコール不正使用
E_NOMEM	-33	0xFFFFFFDF	メモリ不足
E_NOID	-34	0xFFFFFFDE	ID 番号不足
E_OBJ	-41	0xFFFFFFD7	オブジェクト状態エラー
E_NOEXS	-42	0xFFFFFFD6	オブジェクト未生成
E_QOVR	-43	0xFFFFFFD5	キューイングオーバフロー
E_RLWAI	-49	0xFFFFFFCF	待ち状態の強制解除
E_TMOUT	-50	0xFFFFFFCE	ポーリング失敗またはタイムアウト
E_DLT	-51	0xFFFFFFCD	待ちオブジェクトの削除
E_CLS	-52	0xFFFFFFCC	待ちオブジェクトの状態変化
E_WBLK	-57	0xFFFFFFC7	ノンブロッキング受付
E_BOVR	-58	0xFFFFFFC6	バッファオーバフロー

７．６　システムコール一覧

システムコール名	タスク	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			
act_tsk／iact_tsk	○	○	○
can_act	○	○	×
sta_tsk	○	○	○
ext_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	○	○	○
dly_tsk	○	×	×
C）　同期・通信　セマフォ			
sig_sem／isig_sem	○	○	○
wai_sem	○	×	×
pol_sem	○	○	×
twai_sem	○	×	×
ref_sem	○	○	×
D）　同期・通信　イベントフラグ			
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	タイムイベントハンドラ	割込みサービスルーチン
E)　同期・通信　データキュー			
snd_dtq	○	×	×
psnd_dtq／ipsnd_dtq	○	○	○
tsnd_dtq	○	×	×
fsnd_dtq／ifsnd_dtq	○	○	○
rcv_dtq	○	○	×
prcv_dtq	○	○	×
trcv_dtq	○	×	×
ref_dtq	○	○	×
F)　同期・通信　メールボックス			
snd_mbx	○	○	×
rcv_mbx	○	×	×
prcv_mbx	○	○	×
trcv_mbx	○	×	×
ref_mbx	○	○	×
G)　メモリプール管理　固定長メモリプール			
get_mpf	○	×	×
pget_mpf	○	○	×
tget_mpf	○	×	×
rel_mpf	○	○	×
ref_mpf	○	○	×
H)　時間管理システム時刻管理			
set_tim	○	○	×
get_tim	○	○	×
isig_tim	×	×	○
I)　　時間管理周期ハンドラ			
sta_cyc	○	○	×
stp_cyc	○	○	×
ref_cyc	○	○	×

システムコール名	タスク	タイムイベント ハンドラ	割込みサービス ルーチン
J) システム状態管理			
rot_rdq／irot_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	○	○	×
K) 割込み管理			
chg_ims	○	○	○
get_ims	○	○	○
L) システム構成管理機能			
ref_cfg	○	○	○
ref_ver	○	○	○

○：使用可

×：使用不可

索引

R

S

T

U

W

あ

い

お

き

け

こ

さ

し

す

せ

た

て

と

ひ

ふ

へ

ま

め

ゆ

わ

μC3/Compact ユーザーズガイド

2008 年 5 月　　初版
2008 年 8 月　　第 2 版
2009 年 3 月　　第 3 版
2010 年 6 月　　第 4 版
2012 年 7 月　　第 5 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