TOPPERS/OSEK カーネル
外部仕様書

Ver.2.00

2006/05/30

株式会社ヴィッツ
株式会社ヴィッツ　開発第３部

TOPPERS/OSEK Kernel
Toyohashi Open Platform for Embedded Real-Time Systems/OSEK Kernel

＜目次＞

1. 本書の位置付け

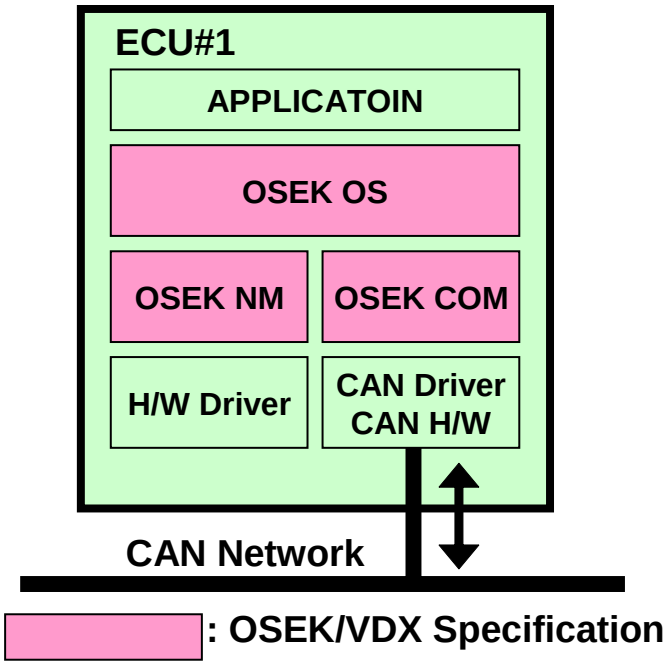
本仕様書は TOPPERS/OSEK カーネルの外部仕様書である。TOPPERS/OSEK カーネルとは、OSEK/VDX（以降 OSEK と記載）OS 仕様 Version2.2.1 に準拠しており、TOPPERS/JSP カーネル 1.4 を参考に作成されたオペレーションシステムである。TOPPERS/OSEK カーネルの位置付けは車載用 OS であるため、他のリアルタイムオペレーティングシステム(以降 RTOS と記載)より特に「厳格なリアルタイム」「低資源」「スケーラビリティ」「信頼度」「コスト」を重要視している。

※OSEK･･･Offene Systeme und deren Schnittestellen fur die Elektronik im Kraftfahrzeug
※VDX･･･Vehicle Distributed eXecutive
※スケーラビリティ･･･既存のハードウェアやソフトウェア構成などを大幅に変更することなく処理に対する要求の質的・量的変化に適応できる度合いを意味する。
※TOPPERS/JSP カーネル･･･RTOS のオープンソースの一つであり、改良が容易である利点を持つ。

1.1. 適用範囲

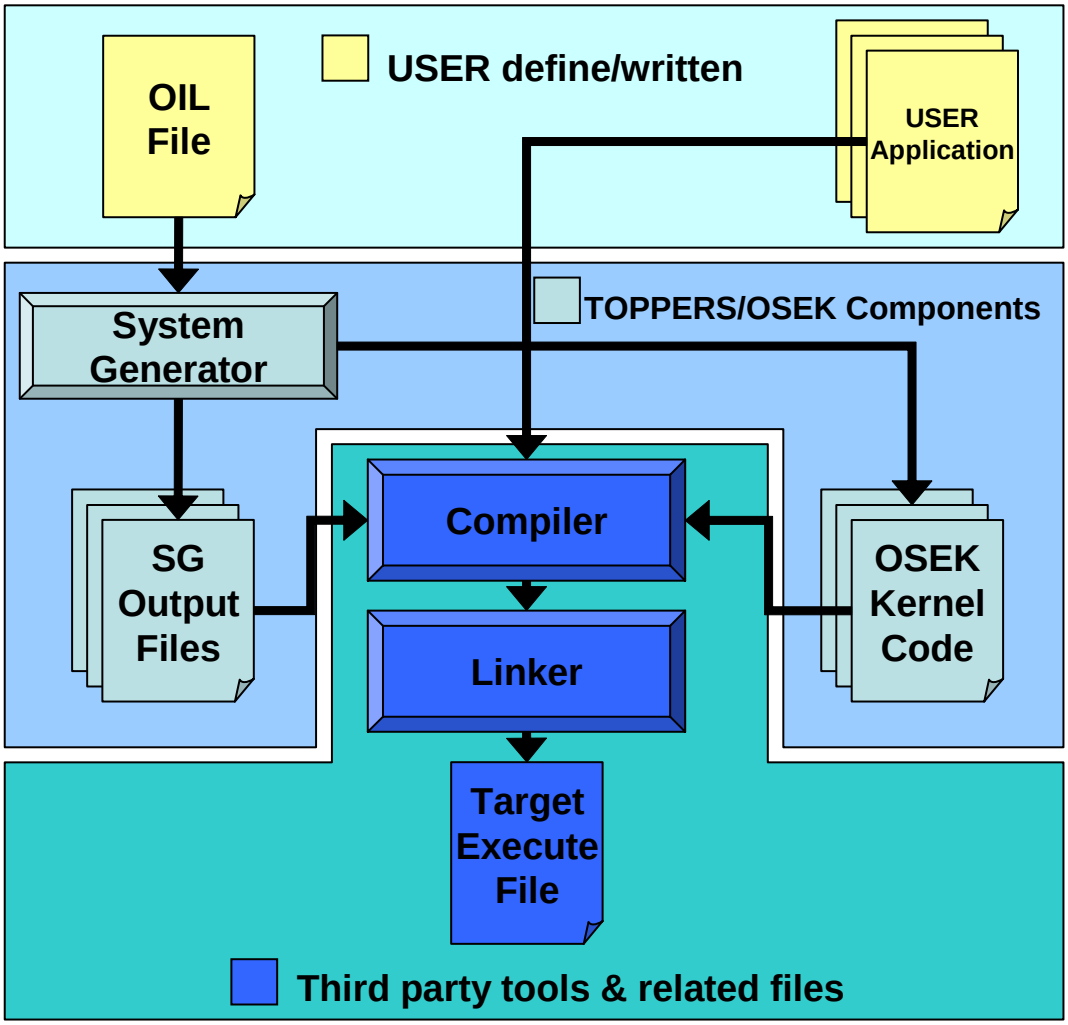
OSEK 仕様で規定されているものを下記に示す。
・Operation System (OSEK OS)
→他の OSEK/VDX モジュール(COM/NM)や ECU ソフトウェアのためのリアルタイム制御の基礎
・Network Management (NM)
→モニタリングなど
・Communication (COM)
→コントロールユニットのデータのやりとりにおける制御

本仕様書では、上記３項目のうち、Operation System を適用範囲とする。下記に OSEK の位置付けを表示する。



1.2. システムコンフィギュレーション

下記にシステムコンフィギュレーションの概要を示す。



単語	意味
アプリケーション コンフィグレーションファイル	静的に宣言された情報(タスク・優先度 etc)が記載されている。
OIL (OSEK Implementation Language)	アプリケーションコンフィグレーションファイルを記載する言語である。
System Generator	アプリケーションコンフィグレーションファイルの情報を静的に解析し、C 言語に変換しﾌｧｲﾙ(SG生成ﾌｧｲﾙ)を生成する。

1.3. 参考文献

次の文書は、本書の一部をなすものではないが、本書の内容を明確にするもの又は本文に関連して参考になるものである。

OSEK/VDX Operation System Specification 2.2.1
OSEK/VDX OSEK Implementation Language Specification 2.4.1
OSEK/VDX OSEK Communication Specification 3.0.2
OSEK/VDX Network Management Concept and Application Programming Interface Version 2.5.2

2. 提供するサービス

OSEK が提供するサービスを下記に記す。

・タスク管理
・同期
・割り込み
・アラーム
・メッセージ
・エラー処理

3. OSEK OS アーキテクチャー

3.1. プロセッシング(処理)レベル

OSEK では 3 つの処理レベルを定義している。

名称	値の範囲
割り込みレベル	0~126
スケジューラレベル	127
タスクレベル	128～254(実際に使用する範囲は実装依存)

ルール

・割り込みはタスクよりも優先

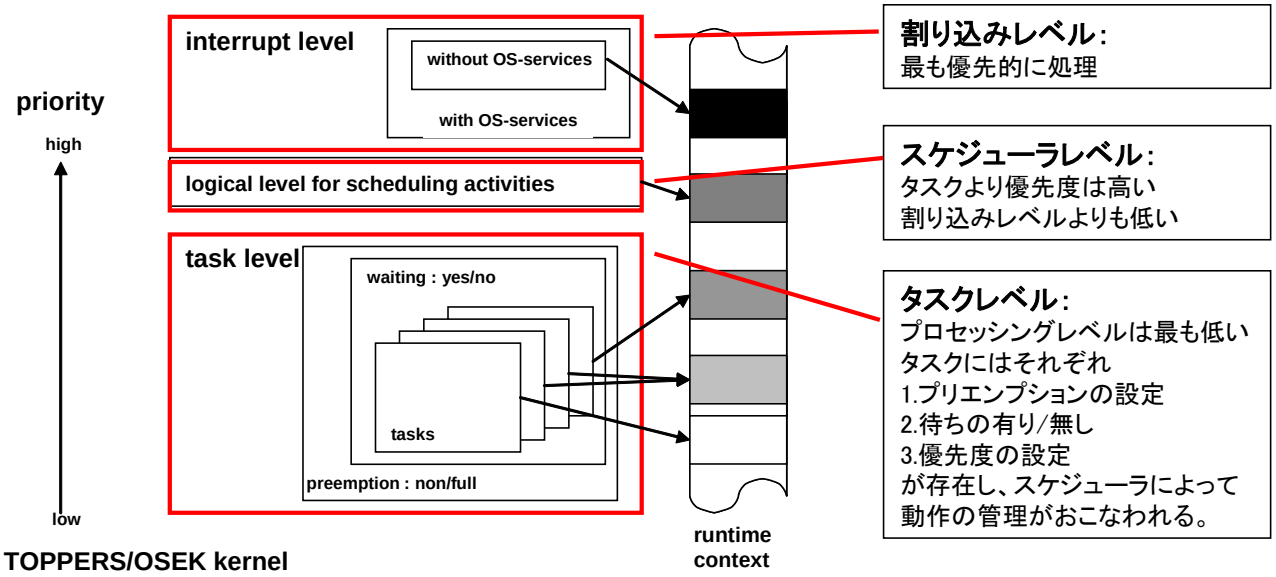
・割り込み処理レベルは、１つ以上から生成

・割り込みサービスルーチンは静的（※）に割り当てられた割り込み優先順位を持つ

・割り込みサービスルーチンの割り当ては、実装/ハード依存

・値が大きいほど高優先度

・タスクの優先順位は、ユーザにより静的に割り当てられる。



3.2. コンフォーマンスクラス

コンフォーマンスクラスは以下の項目をサポートするために作成された項目である。

・機能のモジュール化

・部分的な実装

・機能性のクラスからより高い機能性のクラスへのアップデート

コンフォーマンスクラスには 4 種類(BCC1/BCC2/ECC1/ECC2)存在する。TOPPERS/OSEK カーネル仕様においては ECC２を採用する。各種詳細情報を下記の表に記載する。

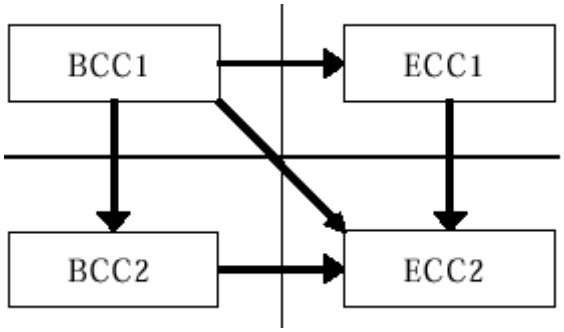
項目	BCC1	BCC2	ECC1	ECC2
タスクの種類	BT	BT	BT/ET	BT/ET
多重要求	×	○	×	○
休止状態でない タスク最大数	8		255 (※BT/ET の合計数)	
1 優先度あたりのタスク数	1	複数	1	複数
タスクの 最大イベント数	-		32	
優先度数	8		16	
リソース	RES_SCHDULER	255(RES_SCHDULER 含む)		
内部リソース数	2			
アラーム数	255			
アプリケーションモード	8			

※BT･･･基本タスク

ET･･･拡張タスクを指す(詳細内容は「４章.タスク」で記載)

3.2.1. コンフォーマンスクラスの関連図

上位レベルは下位レベルを網羅できる。



4. タスク

4.1.1. タスクに関する規定

TOPPERS/OSEK カーネル仕様に関する規定を下記に記載する。

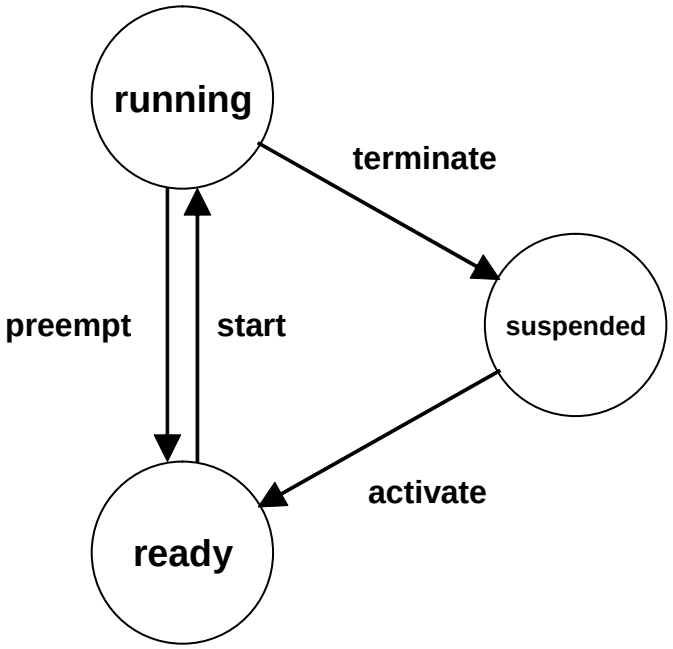
タスク最大数	255
タスクごとのイベント最大数	32
優先度の段階	16 段階

4.2. タスクコンセプト

タスクには 2 種類(基本タスク/拡張タスク)存在する。

4.2.1. 基本タスク

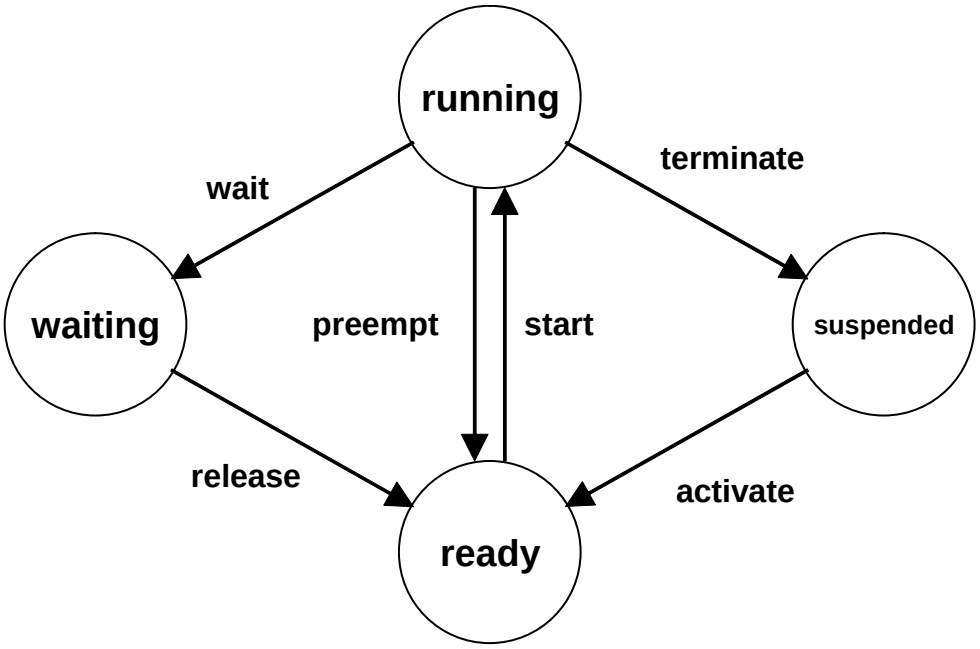
待ちの無いタスクであるが、プリエンプションはされる。



推移	状態		遷移条件
	前	現在	
activate	suspended	ready	システムサービスにより起動要求が発生
start	ready	running	スケジューリングが発生し実行
preempt	running	ready	スケジューリングが発生し、他のタスクが実行
terminate	running	suspended	システムサービスにより休止要求が発生

4.2.2. 拡張タスク

基本タスクの機能及び待ち状態をサポート。

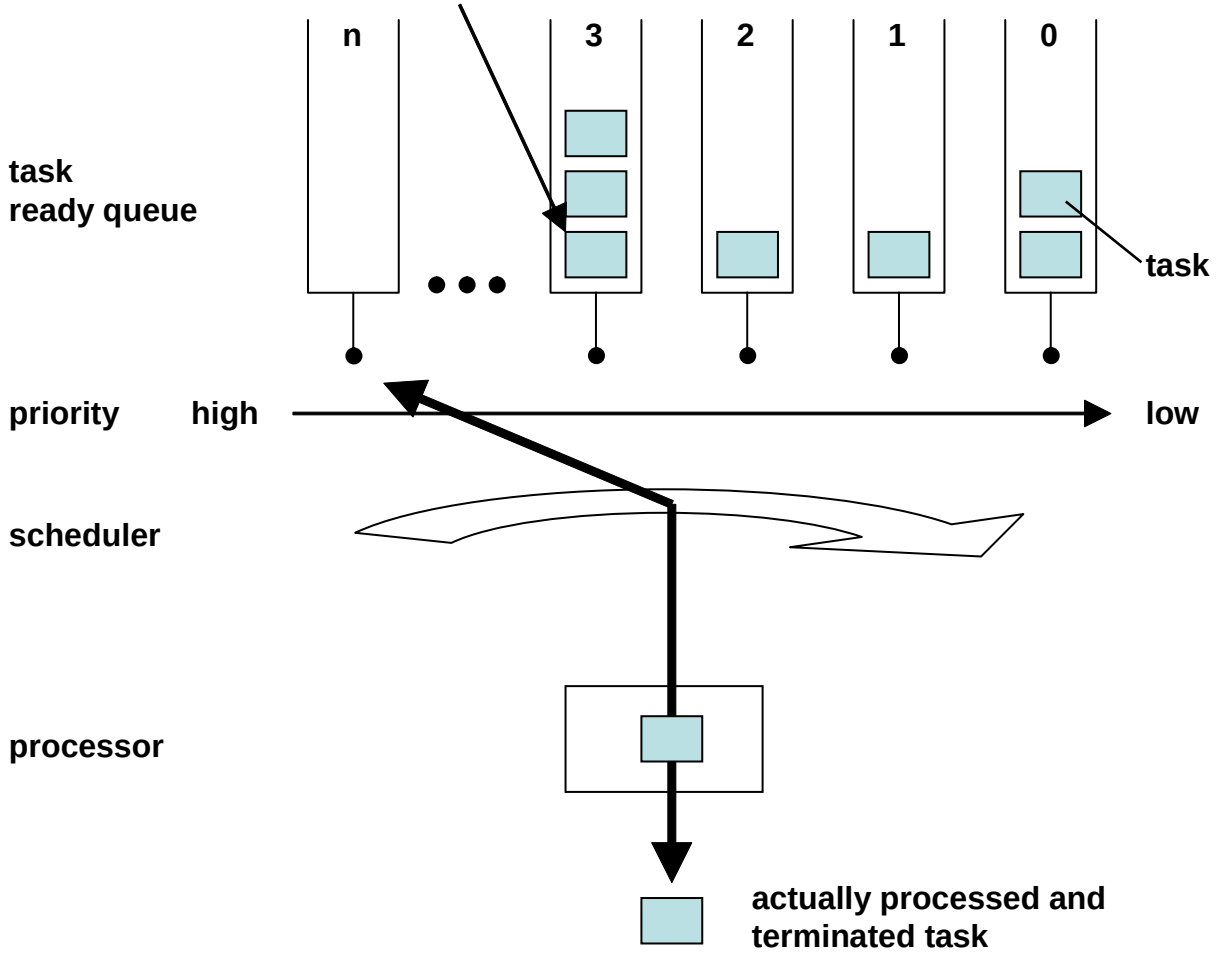


推移	状態		遷移条件
	前	現在	
activate	suspended	ready	システムサービスにより起動要求が発生
start	ready	running	スケジューリングが発生し実行
wait	running	waiting	システムサービスにより待ち要求が発生
release	waiting	ready	待っている要求が発生した場合
preempt	running	ready	スケジューリングが発生し、他のタスクが実行
terminate	running	suspended	システムサービスにより休止要求が発生

4.3. スケジューリング

4.3.1. タスクスケジューラ

タスクスケジューラの動作概要を説明する。



4.3.2. スケジューリング方法

OSEK には３つのスケジューリング方法が存在する。

スケジュール名	内容	備考
フルプリエンプティブ	優先度が高いタスクが優先	途中でタスクが切り替わる可能性あり
ノンプリエンプティブ	現在動作しているタスクが優先	途中でタスクが切り替わる可能性がない
ミクスドプリエンプティブ	フルプリエンプティブ・ノンプリエンプティブが混在	上記二つの特性をタスク毎に設定可能

TOPPERS/OSEK カーネルではミクスドプリアンプティブを採用する。

5. アプリケーションモード

アプリケーションモードとは OS 起動時に設定されるアプリケーションの起動オプションのことである。カーネル起動システムサービス StartOS()の引数がアプリケーションモードに該当する。

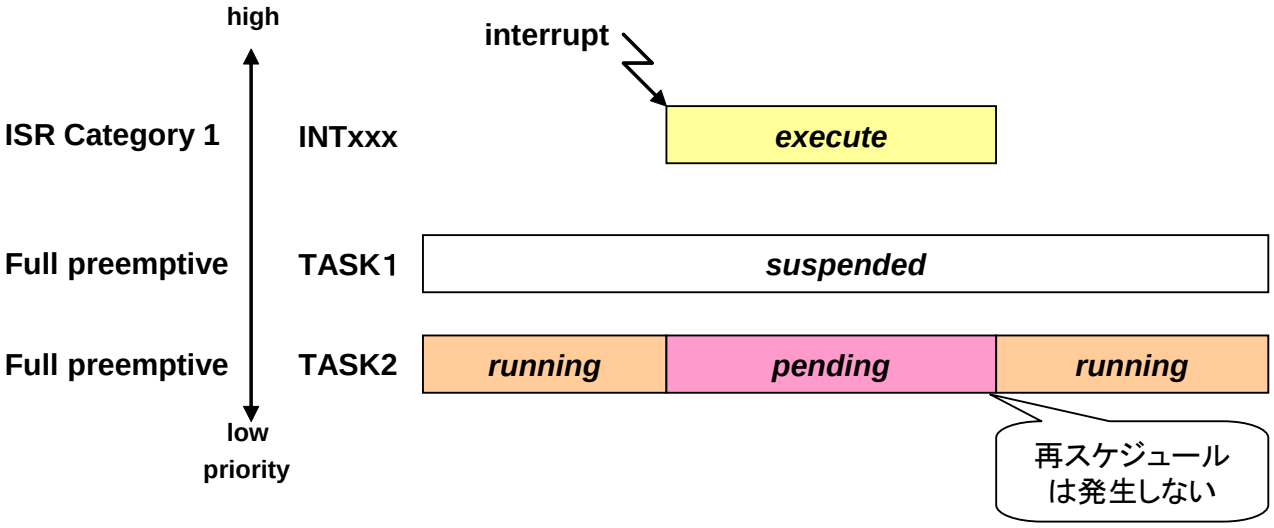
TOPPERS/OSEK カーネルでは 8 モードまで対応する。

6. ISR(割り込み処理)

割り込み処理とは、あるプログラムの実行中に何らかの要因が発生したことに何か処理を行いたい時に制御する処理である。割り込みには 2 つのカテゴリ(カテゴリ 1/カテゴリ 2)が存在する。

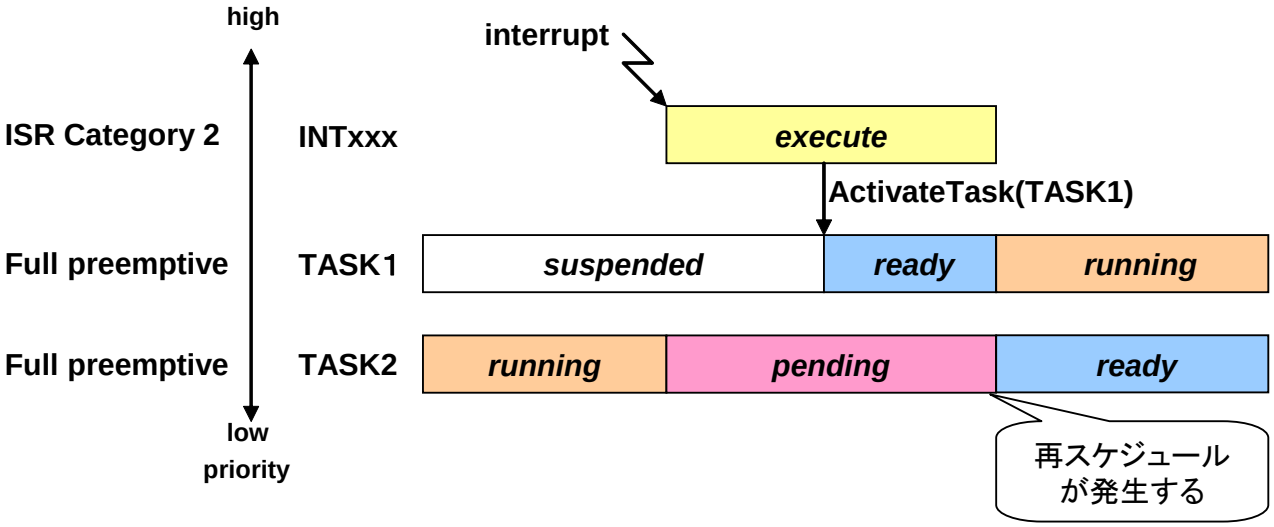
6.1. ISR カテゴリ 1

特徴	OS のサービスを使用しない。 →再スケジューリングを行わない。
	ISR 処理後、割り込みが起こった時点からの処理を継続する。
	システムサービスの呼び出しが不可能である。 ※割り込みの許可/禁止を扱うサービスは使用可能である
備考	OS を使用しないため、割り込み時のオーバーヘッドが少ない。
	処理速度は高速である。
	コンテキスト情報の退避は手動で行わなければならない



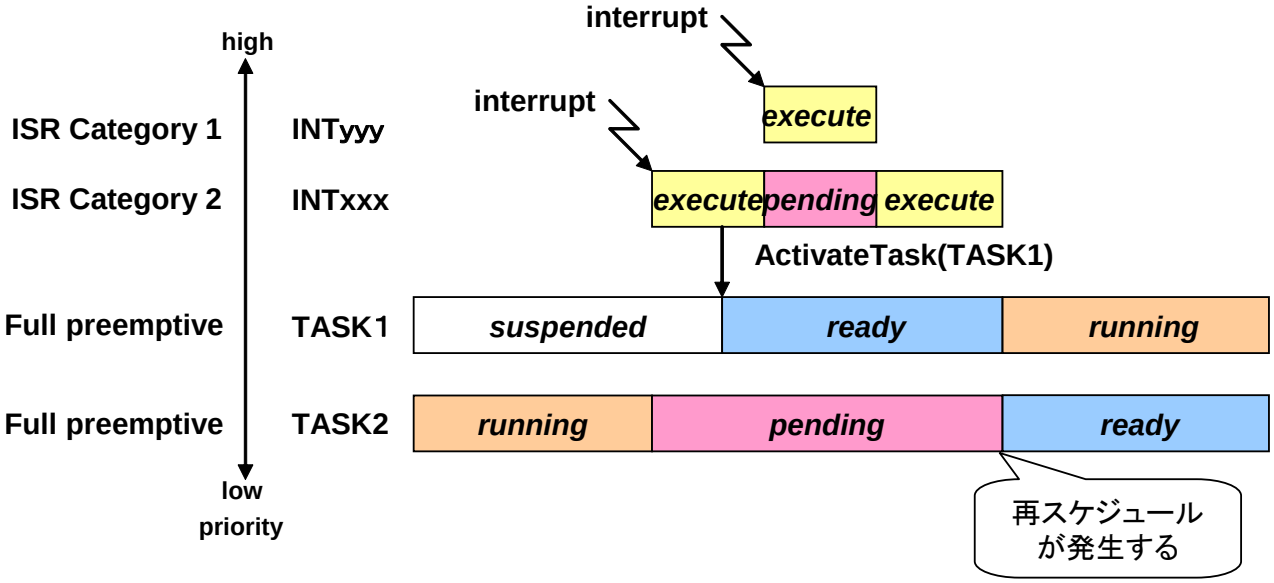
6.2. ISR カテゴリ 2

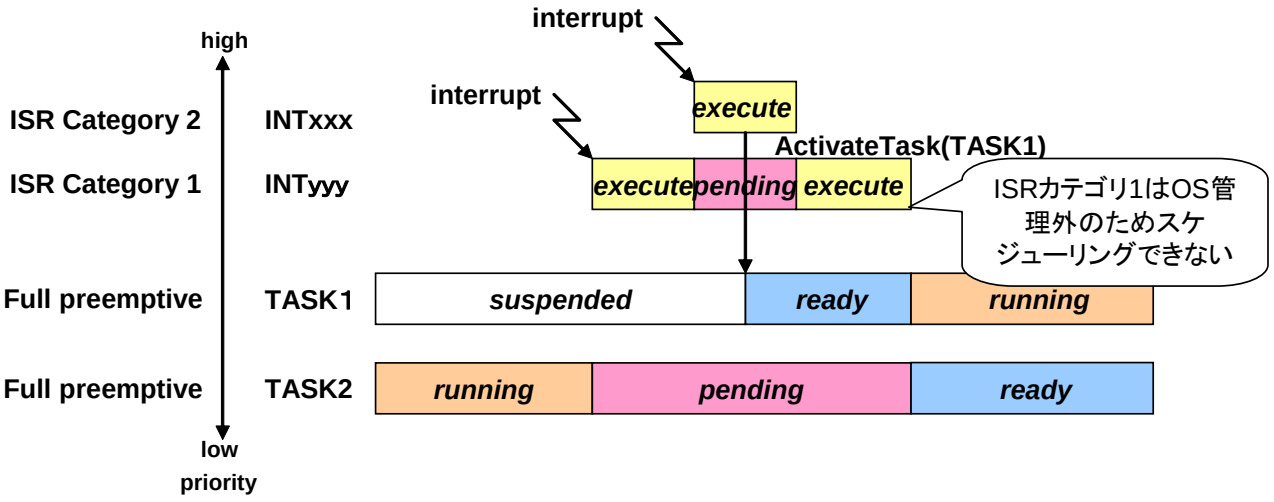
特徴	割り込み時には OS が介在する。
	OS が専用のユーザールーチンの実行環境を準備する。 →スタック・TCB の切り替えが発生する。その後 ISR ルーチンが呼ばれる。
	システムサービスが呼び出し可能である。
	再スケジューリングが発生する。 （ただし、割り込みネストが最上位時のみ発生する。）
備考	OS を使用するため、割り込み時のオーバーヘッドが大きい。
最大数	255(+プロセッサ毎の宣言)



6.3. 補足：多重割り込み発生時における注意点

今回の仕様では ISR カテゴリ 1 とカテゴリ 2 に優先度の境界値を設け、必ず ISR カテゴリ 1 が優先させる仕様とする。

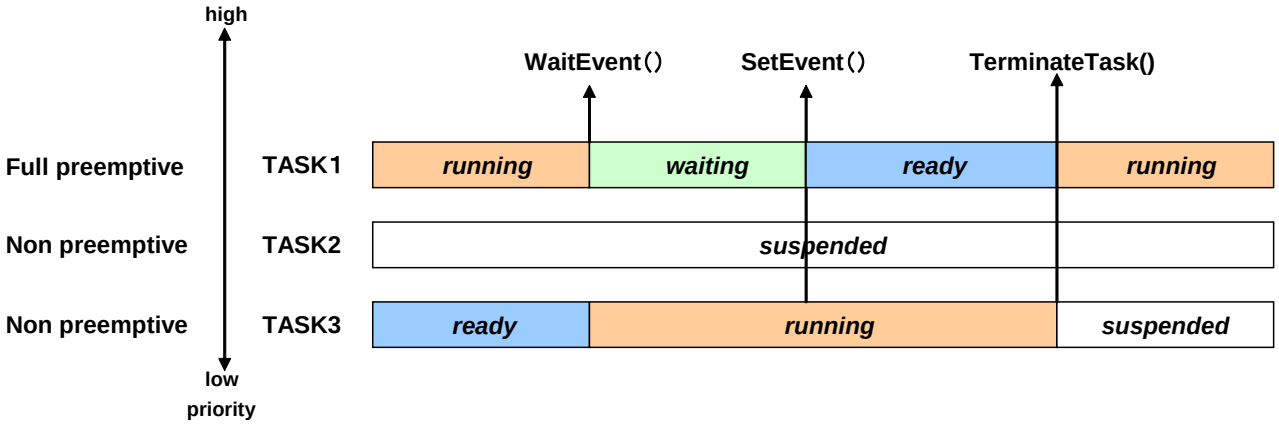




7. イベント

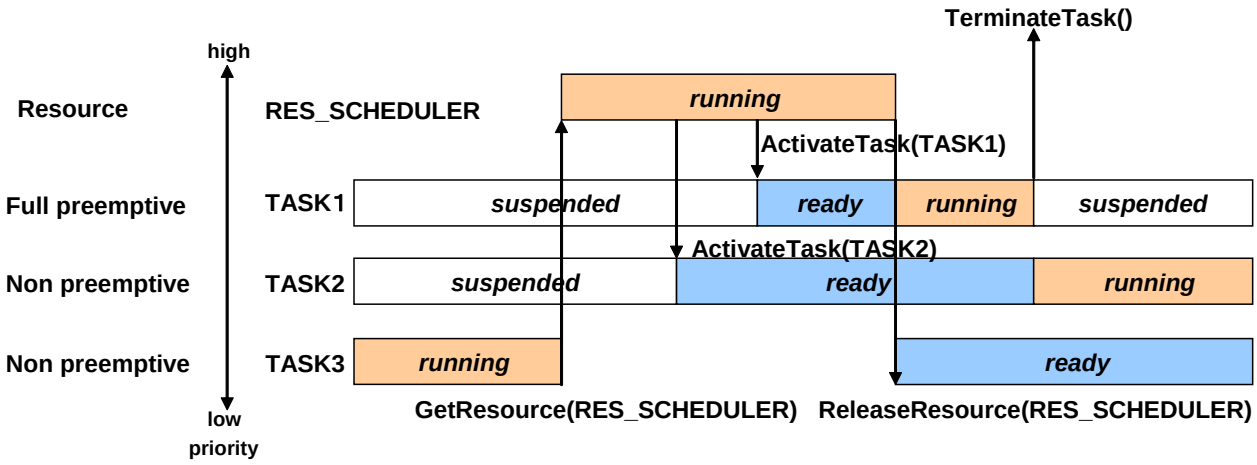
イベント機構はタスク間またはタスクと割り込み間で同期を取るための手段に用いられる。イベントは拡張タスクにのみ用意されている。（状態遷移を【waiting】に遷移）

特徴	拡張タスク毎に用意されている。 →イベント待ちができるのは自イベントのみである。
クリア タイミング	OS が自動的に全イベントをクリアする。（初期化時のみ）
	自分でイベントをクリアしなければならない。(通常時) ※システムサービスで提供
セット タイミング	システムサービスにより提供されている。 ※基本タスク・拡張タスク・ISR カテゴリ２とも使用可能である。
イベント数	８個以上
備考	【suspended 状態】のタスクに対してはセット・参照できない
	フルプリエンプティブの場合、イベントの設定によってスケジューリングが発生する。 ※ノンプリエンプティブの場合はスケジューリングが発生しない。



8. リソース管理

特徴	複数のタスク・ISR が同時に同じリソースを取得しないように制御 →リソース管理をすることによって排他制御を行うことが可能。
取得・解放	システムサービスにより提供されている。
対応	全コンフォーマンスクラスに対応
優先度値	静的に設定(OIL に記載) ※「８－１．リソース優先度における資料」に詳細情報記載
優先度範囲	【リソースを使用している最も低いタスクの優先度】<= 【値】<= 【リソースを使用していない最も高いタスクの優先度】 ※「８－１．リソース優先度における資料」に詳細情報記載
優先度変更タイミング	リソース取得・解放時に優先度を変更する。 ※内部リソースはタスク実行・終了時に変更する。
備考	取得・解放順序は、優先度の低いものから取得し、高いものから解放する。 →LIFO の原理に基づく
	同一リソースへの多重アクセスは禁止
	タスクがリソースを未開放の状態で終了することは禁止とする。 内部リソースは自動的に OS が解放する。
最大数	255



9. アラーム

指定した時間に発生するイベントを扱うための機能である。

特徴	カウンタ値及び定数で表現される。
	単位は【ticks】 ※カウンタの重みは実装依存である。
	OS はカウンタを直接操作するシステムサービスを提供しない。
	OS はタイマに関連するカウンタを最低一つは提供する。
	複数のカウンタを所有することが可能である。
提供するサービス※	タスク起動
	イベントの設定
	アラームコールバックルーチンのコール
カウンタ値の設定方法※	絶対アラーム →カウンタが指定した値になった時
	相対アラーム →経過値が指定した値になった時
アラームの設定方法※	シングルアラーム →一度のみ処理を行う
	周期アラーム →周期ごとに処理を行う
アラーム最大数	255
カウンタ最大数	255
Tick のビット数	32bit

10. メッセージ

メッセージ通信とはノード内・ノード間での通信のことであり、詳細はOSEK COM仕様を参照のこと。

TOPPERS/OSEK カーネルでは OSEK COM は内包していない。

11. エラー処理

11.1. フック処理

フックとはシステム処理内で、ユーザ定義の処理を実行させる仕組みのことである。

特徴	OS に呼ばれる
	実行コンテキストは実装依存
	全てのタスクより高い優先度
	インターフェースは OSEK OS 仕様で標準化
	移植性はない
	システムサービスを呼ぶことができる(制限あり)
	OIL で提供されているフックルーチンの使用有無を指定できる
備考	カテゴリ 2 の割り込みはフック割り込みを禁止した状態で実行
	システム処理に介入することができ、拡張・カスタマイズが容易である。

11.1.1. 提供されているフックルーチン

フックルーチン名	内容
ErrorHook	システムサービスのエラー
StartupHook/ShutdownHook	システムの起動・停止
PreTaskHook/PostTaskHook	タスク切り替えの前後

11.2. エラー

特徴	システムサービスが【E_OK】以外を返す時に呼ばれる。
	エラーフック内で再度エラーが発生してもエラーフックは呼ばれない。
	タスク実行中・イベント設定時にも呼ばれる。

11.2.1. エラー情報の取得(OSErrorGetServiceId/ OSError_xxx_yyy)

ここではどのようなエラーが発生したのか追及することができるシステムサービスの内容を記載する。情報を取得するためには、下記の２つのシステムサービスを使用する。

システムサービス名	内容
OSErrorGetServiceId	エラーが発生したシステムサービス情報を取得
OSError_xxx_yyy	システムサービスのパラメータを取得 xxx･･･システムサービス名 yyy･･･システムサービスの引数

11.2.2. エラーの種類 1(標準エラー/拡張エラー)

ここではシステムサービスの戻り値で取得できるエラーについて記載する。システムサービスで取得できるエラーには 2 種類(標準エラー/拡張エラー)存在する。静的にどちらのエラーを出力するか設定をすることが可能である。

エラー	内容	備考
標準エラー	標準的にチェックすべきエラー →動的なチェックが不可欠なエラー	デバッグモード時に使用
拡張エラー	リリース時にチェックすべきエラー →余分なエラーは発生しない	リリースモード時に使用

11.2.3. エラーの種類 2(Application Errors/ Fatal Errors)

ここでは OS が呼ぶエラーの種類を記載する。OS が呼ぶことができるエラーの種類は 2 種類(Application Errors/ Fatal Errors)存在する。

エラー	タイミング	その後の処理
Application Errors	OS に要求したサービスの実行に失敗時 →システムサービスのエラー	継続
Fatal Errors	OS 内部の致命的エラー発生時	中断→OS 停止

11.3. デバッグ(PreTaskHook/ PostTaskHook)

ここではタスクの起動時・起動前のタイミングで処理を行うことができるシステムサービスについて記載する。

システムサービス名	PreTaskHook	PostTaskHook
タイミング	新しいタスクの状態が実行状態になった後	実行タスクの状態が、実行状態でなくなる前
呼び出し方法	OS から呼ばれる	
用途	デバッグ・実測時間の計測に利用可能	
備考	―	

11.4. 起動処理(StartupHook)

用途	デバイスドライバの初期化などに利用
呼び出しタイミング	StartOS()後 ←OS の初期化が終了し、スケジューラが走る前に、OS から呼ばれる ※下記に図を記載
備考	実行中は全割り込み禁止

11.5. 終了処理(ShutdownHook)

用途	OS がシャットダウンする際に設定しなければならない内容を記載
呼び出しタイミング	ShutdownOS()内 ←OS がシャットダウンする間に、OS から呼ばれる ※下記に図を記載
	Fatal Errors 発生時
備考	－

12. システムサービス

12.1. システムサービスと各状態との対比表

各状態でのシステムサービス実行可否を表した一覧表を示す。

Service	Task	ISR category 1	ISR category 2	ErrorHook	PreTaskHook	PostTaskHook	StartupHook	ShutdownHook	alarm-callback
ActivateTask	○		○						
TerminateTask	○								
ChainTask	○								
Schedule	○								
GetTaskID	○		○	○	○	○			
GetTaskState	○		○	○	○	○			
DisableAllInterrupts	○	○	○						
EnableAllInterrupts	○	○	○						
SuspendAllInterrupts	○	○	○						○
ResumeAllInterrupts	○	○	○						○
SuspendOSInterrupts	○	○	○	○	○	○			
ResumeOSInterrupts	○	○	○	○	○	○			
GetResource	○		○						
ReleaseResource	○		○						
SetEvent	○		○						
ClearEvent	○								
GetEvent	○		○	○	○	○			
WaitEvent	○								
GetAlarmBase	○		○	○	○	○			
GetAlarm	○		○	○	○	○			
SetRelAlarm	○		○						
SetAbsAlarm	○		○						
CancelAlarm	○		○						
GetActiveApplicationMode	○		○	○	○	○	○	○	
StartOS									
ShutdownOS	○		○	○			○		

12.2. Task Management

＜概略＞

タスクの状態遷移システムサービスを提供する。

タスクの状態（情報）取得システムサービスを提供する。

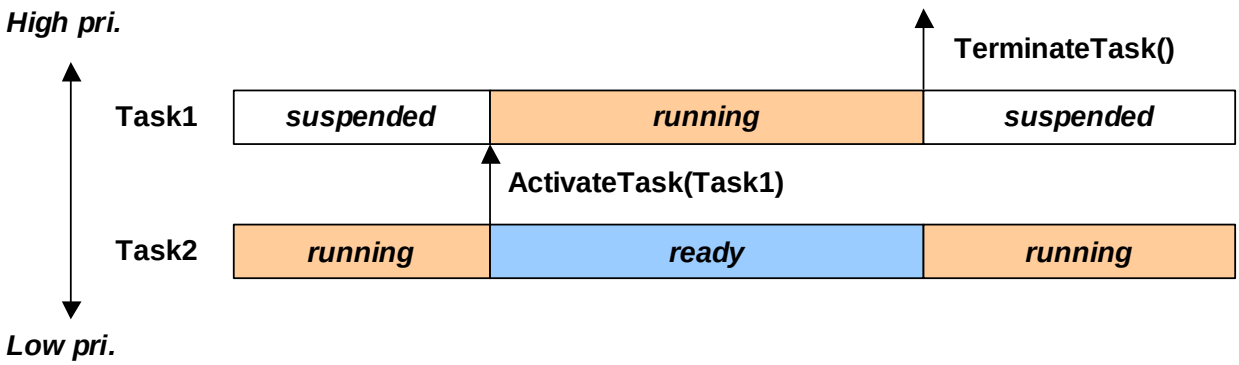
システムサービスはタスクレベルから呼び出される。

12.2.1. DeclareTask

構文		DeclareTask （ TaskIdentifier ）
パラメータ(in)		TaskIdentifier：タスク識別子
パラメータ(out)		なし
記述		タスクの外部宣言
特性		なし
状態	標準	なし
	拡張	なし
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		－

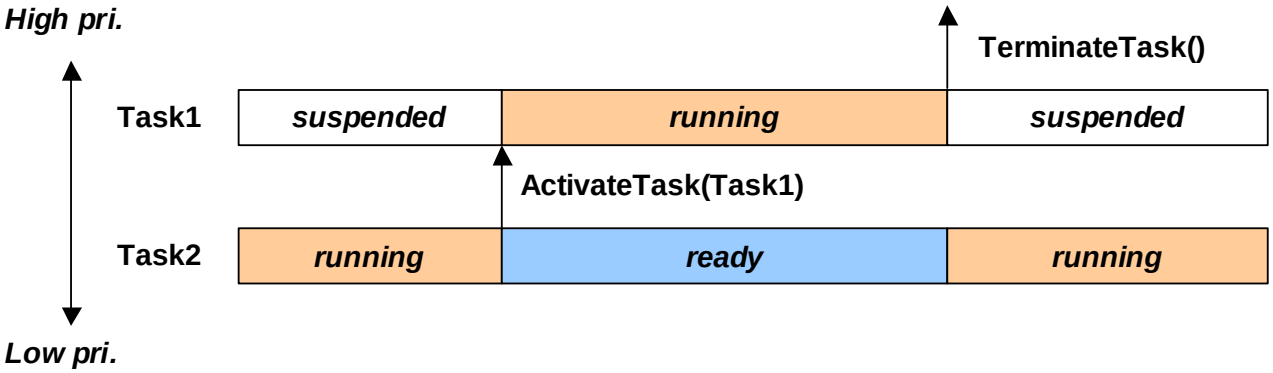
12.2.2. ActivateTask

構文		StatusType ActivateTask （ TaskType TaskID ）
パラメータ(in)		TaskID：タスク ID
パラメータ(out)		なし
記述		指定タスクを【suspend 状態】から【ready 状態】に遷移させる。
特性		サービスは割り込みレベルとタスクレベルからコールする。サービスコール後は再スケジューリングを行う。 既に【running 状態】ならば起動要求をキューイングする。
状態	標準	E_OK：正常終了
		E_OS_LIMIT：起動要求最大数を超えている
	拡張	E_OS_ID：無効な TaskID
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		拡張タスクを遷移させる場合は、イベントは全てクリアされる。



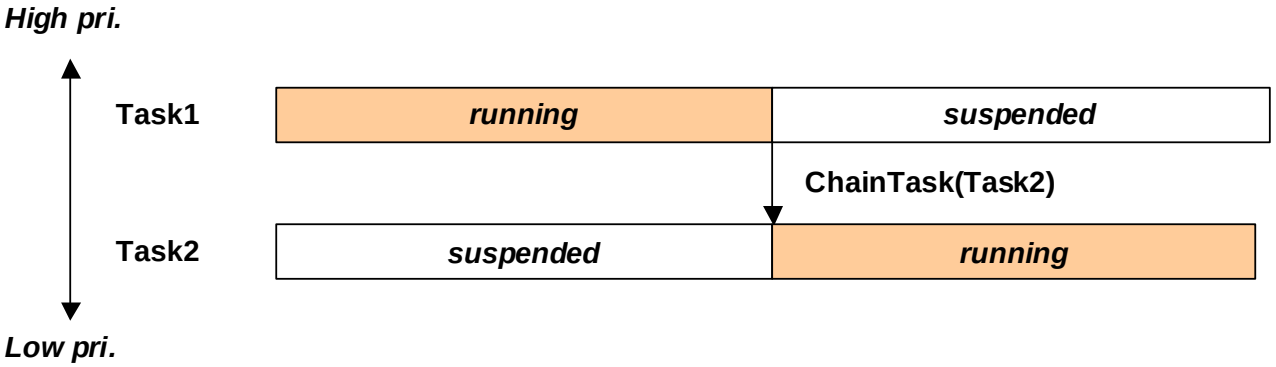
12.2.3. TerminateTask

構文		StatusType　TerminateTask　(　void　)
パラメータ(in)		なし
パラメータ(out)		なし
記述		自タスクを【running 状態】から【suspended 状態】に遷移させる。
特性		確保したリソースはサービスコール前にリリース(解放)すること。 複数起動要求がある場合は、【suspended 状態】から【ready 状態】に遷移させる。 サービスコール後は再スケジューリングを行う。
状態	標準	なし
	拡張	E_OS_RESIOURCE：リソース未解放
		E_OS_CALLLEVEL：割り込みレベルからの呼び出し
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		TerminateTask()、ChainTask()以外でのタスク終了は推奨しない。



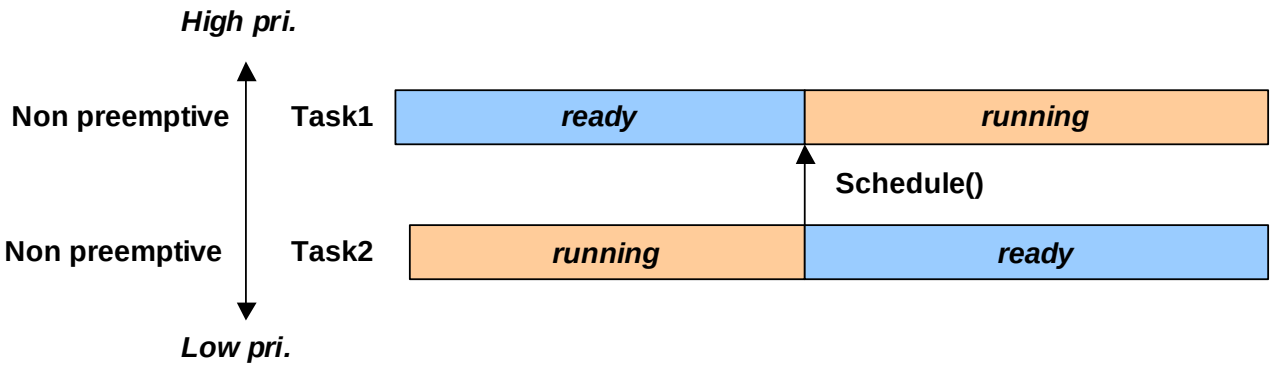
12.2.4. ChainTask

構文		StatusType　ChainTask　(　　TaskType　TaskID　)
パラメータ(in)		TaskID：タスク ID
パラメータ(out)		なし
記述		自タスクを【suspended 状態】に遷移させ、TaskID で指定したタスクを【ready 状態】に遷移させる。
特性		指定タスクを自タスクにした場合は【suspended 状態】にならない。 確保したリソースはサービスコール前にリリース(解放)すること。 サービスコール後は再スケジューリングを行う。
状態	標準	E_OS_LIMIT：起動要求最大数を超えている
	拡張	E_OS_ID：無効なタスク ID
		E_OS_RESOURCE：リソース未解放
		E_OS_CALLLEVEL：割り込みレベルからの呼び出し
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		TerminateTask()、ChainTask()以外でのタスク終了は推奨しない。



12.2.5. Schedule

構文		StatusType　Schedule　（　void　）
パラメータ(in)		なし
パラメータ(out)		なし
記述		高優先度タスクが【ready 状態】ならば、自タスクを【ready 状態】に遷移させ、高優先度タスクを【running 状態】に遷移させる。
特性		確保したリソースはサービスコール前にリリース(解放)すること。 サービスコール後は再スケジューリングを行う。
状態	標準	E_OK：正常終了
	拡張	E_OS_RESOURCE：リソース未解放
		E_OS_CALLLEVEL：割り込みレベルからの呼び出し
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		-



12.2.6. GetTaskID

構文		StatusType　GetTaskID　(　TaskRefType　TaskID　)
パラメータ(in)		なし
パラメータ(out)		TaskID
記述		【running 状態】のタスクの TaskID を取得する。
特性		タスクレベル、割り込みレベル、フックルーチンから呼び出すこと。 【running 状態】のタスクが存在しない場合は、TaskID 値に INVALID_TASK を返す。
状態	標準	E_OK：正常終了
	拡張	E_OK：正常終了
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		INVALID_TASK は未定義を指す。

12.2.7. GetTaskState

構文		StatusType　GetTaskState (　TaskType　TaskID　,　TaskStateRefType　Status　)
パラメータ(in)		TaskID：タスク ID
パラメータ(out)		Status：タスク状態(RUN,READY,WAIT,SUSPEND)
記述		指定タスクの状態を取得する。
特性		タスクレベル、割り込みレベル、フックルーチンから呼び出すこと。 プリエンプティブタスクの場合、【running 状態】のタスクが存在しない場合は、TaskID 値に INVALID_TASK を返す。
状態	標準	E_OK：正常終了
	拡張	E_OK：正常終了
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		INVALID_TASK は未定義を指す。

12.3. Interrupt handling

＜概略＞

割り込みを無効化/有効化するシステムサービスを提供。

クリティカルセクションを作成。

割り込み無効化期間は最小のオーバーヘッドで実装。

システムサービスはタスクレベル、割り込みレベルからの呼び出し。

12.3.1. EnableAllInterrupts

構文		void　EnableAllInterrupts　（　void　）
パラメータ(in)		なし
パラメータ(out)		なし
記述		ハードウェアがサポートする全ての割り込みを有効にする。
特性		DisableAllInterrupts により割り込みを全許可する。 タスクレベル、割り込みレベルから呼び出すこと。 フックルーチンからはコールできない。 DisableAllInterrupts で開始されたクリティカルセクションを終了。
状態	標準	なし
	拡張	なし
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		最小のオーバーヘッドでの実装を推奨する。

12.3.2. DisableAllInterrupts

構文		void　DisableAllInterrupts　（　void　）
パラメータ(in)		なし
パラメータ(out)		なし
記述		ハードウェアがサポートする全ての割り込みを無効にする。
特性		割り込みを全禁止する。 タスクレベル、割り込みレベルから呼び出すこと。 フックルーチンからはコールできない。 このサービスでクリティカルセクションを開始。 プロセッサレベルで中断・禁止を行う。 クリティカルセクション中はシステムサービスを呼び出せない。
状態	標準	なし
	拡張	なし
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		割り込みネストはサポートしない。 最小のオーバーヘッドでの実装を推奨する。

12.3.3. ResumeAllInterrupts

構文		void　ResumeAllInterrupts　（　void　）
パラメータ(in)		なし
パラメータ(out)		なし
記述		ハードウェアがサポートする全ての割り込みを復帰させる。
特性		SuspendAllInterrupts により割り込みを全復帰する。 タスクレベル、割り込みレベル、フックルーチンから呼び出すこと。 SuspendAllInterrupts で開始されたクリティカルセクションを終了。
状態	標準	なし
	拡張	なし
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		最小のオーバーヘッドでの実装を推奨する。

12.3.4. SuspendAllInterrupts

構文		void　SuspendAllInterrupts　（　void　）
パラメータ(in)		なし
パラメータ(out)		なし
記述		ハードウェアがサポートする全ての割り込みを保留する。
特性		割り込みを全保留する。 タスクレベル、割り込みレベル、フックルーチンから呼び出すこと。 このサービスでクリティカルセクションを開始する。 クリティカルセクション中のシステムサービスコールは、 SuspendAllInterrupts/ResumeAllInterrupts の組と SuspendOSInterrupts/ResumeOSInterrupts の組は使用可能。
状態	標準	なし
	拡張	なし
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		サービスのネストはサポート。 最小のオーバーヘッドでの実装を推奨する。

12.3.5. ResumeOSInterrupts

構文		void　ResumeOSInterrupts　（　void　）
パラメータ(in)		なし
パラメータ(out)		なし
記述		ISR カテゴリ 2 の割り込みを復帰する。
特性		ISR カテゴリ 2 に所属する割り込みを復帰する。 タスクレベル、割り込みレベルから呼び出すこと。 SuspendOSInterrupts で開始されたクリティカルセクションを終了。
状態	標準	なし
	拡張	なし
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		最小のオーバーヘッドでの実装を推奨する。

12.3.6. SuspendOSInterrupts

構文		void　SuspendOSInterrupts　(　void　)
パラメータ(in)		なし
パラメータ(out)		なし
記述		ISR カテゴリ 2 の割り込みを保留する。
特性		ISR カテゴリ 2 に所属する割り込みを保留する。 タスクレベル、割り込みレベルから呼び出すこと。 このサービスでクリティカルセクションを開始する。 クリティカルセクション中のシステムサービスコールは、 SuspendAllInterrupts/ResumeAllInterrupts の組と SuspendOSInterrupts/ResumeOSInterrupts の組は使用可能。
状態	標準	なし
	拡張	なし
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		最小のオーバーヘッドでの実装を推奨する。

12.4. Resource Management

＜概略＞

リソースの取得/解放するシステムサービスを提供。

同じ機能レベルの同じ機能の中でコールすること。

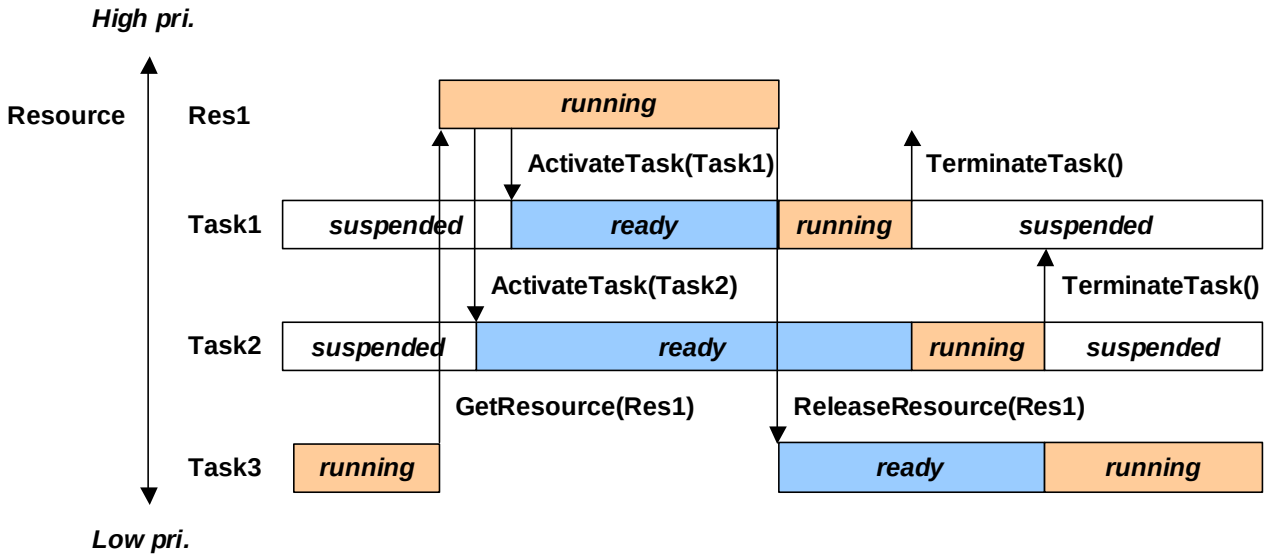
システムサービスはタスクレベル、カテゴリ 2 割込みレベルからの呼び出し。

12.4.1. DeclareResource

構文		DeclareResource （ ResourceIdentifier ）
パラメータ(in)		ResourceIdentifier：リソース識別子
パラメータ(out)		なし
記述		リソースの外部宣言。
特性		なし
状態	標準	なし
	拡張	なし
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		－

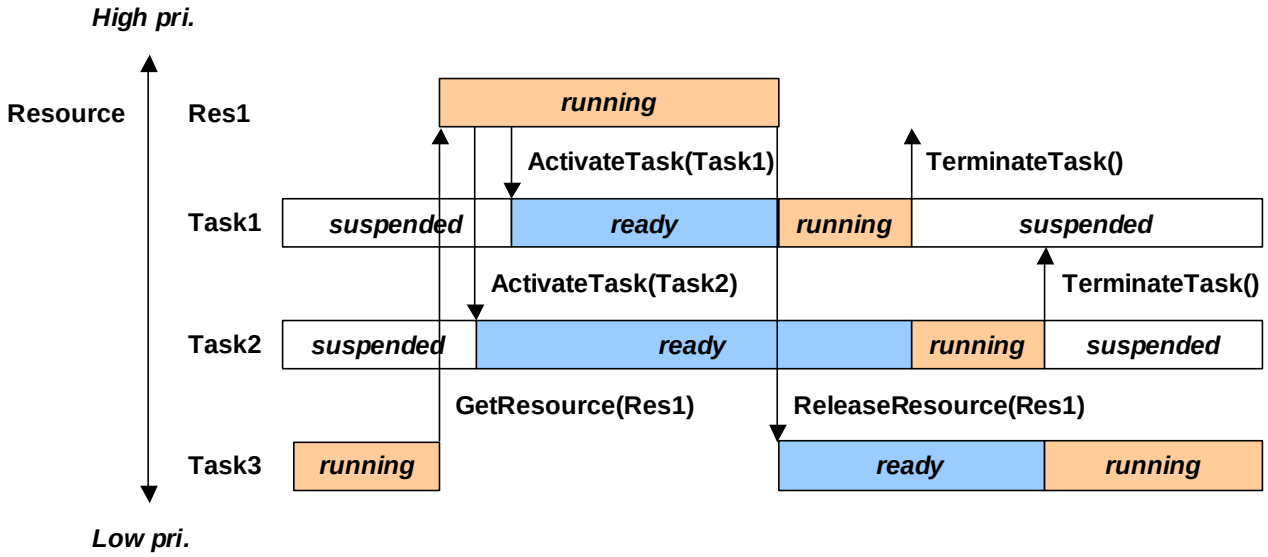
12.4.2. GetResource

構文		void　GetResource　（　ResourceType　ResID　）
パラメータ(in)		ResID：リソース ID
パラメータ(out)		なし
記述		リソース ID で指定されたリソースを取得する。
特性		リソースに割り当てられたクリティカルセクションに入る。 クリティカルセクションは ReleaseResource()で抜ける。
状態	標準	E_OK：正常終了
	拡張	E_OS_ID：リソース ID が無効
		E_OS_ACCESS:既にリソースを獲得済み
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		クリティカルセクション中に再スケジュールが発生するシステムサービスを呼ぶことは禁止。(TerminateTask , ChainTask , Schedule , WaitEvent)



12.4.3. ReleaseResource

構文		void　ReleaseResource　（　ResourceType　ResID　）
パラメータ(in)		ResID：リソース ID
パラメータ(out)		なし
記述		リソース ID で指定されたリソースを解放する。
特性		リソースに割り当てられたクリティカルセクションから抜ける。
状態	標準	E_OK：正常終了
	拡張	E_OS_ID：リソース ID が無効
		E_OS_NOFUNC：獲得していないリソースを解放しようとした
		E_OS_ACCESS：リソースより優先度が高いタスクが解放しようとした
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		-



12.5. Event Control

＜概略＞

イベント管理するシステムサービスを提供。

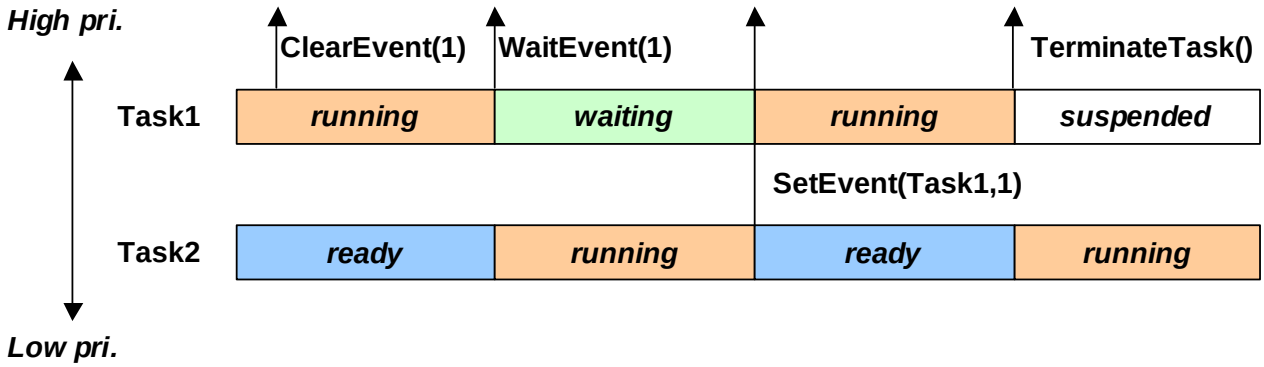
タスクレベル、カテゴリ 2 割込みレベルからの呼び出し。

12.5.1. DeclareEvent

構文		DeclareEvent （ EventIdentifier ）
パラメータ(in)		EventIdentifier：イベント識別子
パラメータ(out)		なし
記述		イベントの外部宣言。
特性		なし
状態	標準	なし
	拡張	なし
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		－

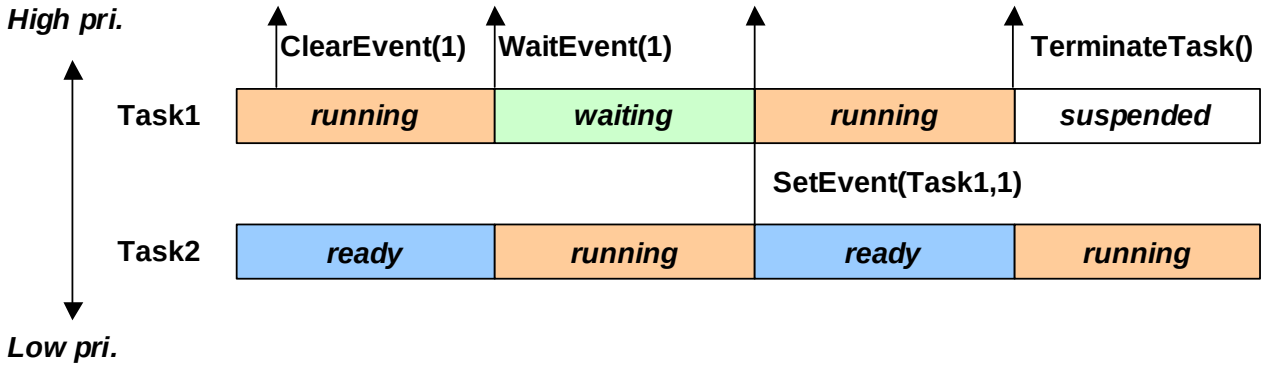
12.5.2. SetEvent

構文		StatusType　SetEvent (　TaskType　TaskID　,　EventMaskType　mask　)
パラメータ(in)		TaskID：指定タスク ID
		mask：イベントマスク
パラメータ(out)		なし
記述		指定タスクにイベントを設定する。
特性		タスク ID で指定したタスクに対してイベントマスクを設定する。 イベントを設定されたタスクは ready 状態に遷移する。 タスクレベル、割り込みレベルから呼び出すこと。
状態	標準	E_OK：正常終了
	拡張	E_OS_ID：TaskID が無効
		E_OS_STATE：指定タスクが suspended 状態のため、イベントが設定できない。
		E_OS_ACCESS：指定タスクが拡張タスクではない
コンフォーマンス		ECC1/ECC2
備考		-



12.5.3. ClearEvent

構文		StatusType　ClearEvent　（　EventMaskType　mask　）
パラメータ(in)		mask：イベントクリア対象マスク
パラメータ(out)		なし
記述		イベントをクリアする。
特性		自タスクに設定されてあるイベントに対して、イベントマスクに従ってクリアする。 イベントが設定されている拡張タスクのみに制限される。 タスクレベルから呼び出すこと。
状態	標準	E_OK：正常終了
	拡張	E_OS_CALLEVEL：割り込みレベルで呼び出した
		E_OS_ACCESS：自タスクは拡張タスクではない
コンフォーマンス		ECC1/ECC2
備考		－

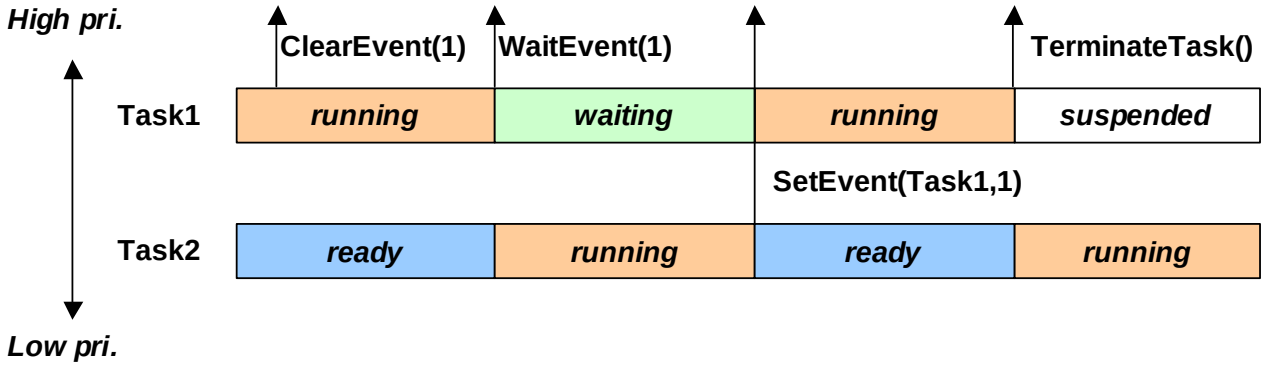


12.5.4. GetEvent

構文		StatusType　GetEvent （　TaskType　TaskID　，　EventMaskRefType　Event　）
パラメータ(in)		TaskID：指定タスク ID
パラメータ(out)		Event：指定タスクに設定されたイベント
記述		指定タスクに設定されているイベント状況を取得する。
特性		サービスはタスクが待っているイベントに対してではなく、タスクの全イベントビットの現状を返す。 指定するタスクは拡張タスクのみに制限される。 タスクレベル、割り込みレベル、フックルーチンから呼び出すこと。
状態	標準	E_OK：正常終了
	拡張	E_OS_ID：TaskID が無効
		E_OS_STATE：指定タスクが suspended 状態のため、イベントが参照できない
		E_OS_ACCESS：自タスクは拡張タスクではない
コンフォーマンス		ECC1/ECC2
備考		－

12.5.5. WaitEvent

構文		StatusType　WaitEvent　（　EventMaskType　mask　）
パラメータ(in)		mask：イベント待ち対象マスク
パラメータ(out)		なし
記述		指定イベントマスクでイベント待ち【waiting 状態】に遷移する。
特性		イベントマスクに従って自タスクを【waiting 状態】に遷移する。 システムサービス実行後、再スケジューリングが発生する。 イベントが設定されてある拡張タスクのみに制限される。 タスクレベルから呼び出すこと。
状態	標準	E_OK：正常終了
	拡張	E_OS_RESOURCE：未解放のリソースがある
		E_OS_CALLEVEL：割り込みレベルで呼び出した
		E_OS_ACCESS：自タスクは拡張タスクではない
コンフォーマンス		ECC1/ECC2
備考		待ち状態に遷移後、内部リソースがリリースされる。



12.6. Alarms

＜概略＞

アラーム管理するシステムサービスを提供

タスクレベル、カテゴリ 2 割込みレベルからの呼び出し

12.6.1. DeclareAlarm

構文		DeclareAlarm （ AlarmIdentifier ）
パラメータ(in)		AlarmIdentifier：アラーム識別子
パラメータ(out)		なし
記述		アラームの外部宣言
特性		なし
状態	標準	なし
	拡張	なし
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		－

12.6.2. GetAlarmBase

構文		StatusType GetAlarmBase (AlarmType AlarmID , AlarmBaseRefType info)
パラメータ(in)		AlarmID：指定アラーム ID
パラメータ(out)		info：AlarmBaseType 構造体のポインタ
記述		AlarmBaseType 構造体情報を取得する。
特性		AlarmBaseType 構造体情報を取得する。 タスクレベル、割り込みレベル、フックルーチンから呼び出すこと。
状態	標準	E_OK：正常終了
	拡張	E_OS_ID：AlarmID が無効
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		AlarmBaseType ・maxallowedvalue：最大の可能な許容カウント値 ・ticksperbase ：Ticks の数がカウンタ特有(重要な)のユニットに達することが必要。 ・mincycle ：SetRelAlarm/SetAbsAlarm(extended 状態があるシステムだけ)の サイクルパラメタのための最も小さい許容値。

12.6.3. GetAlarm

構文		StatusType　GetAlarm (　AlarmType　AlarmID　,　TickRefType　Tick　)
パラメータ(in)		AlarmID：指定アラーム ID
パラメータ(out)		Tick：アラーム満了まで Tick 値
記述		アラームが満了するまでの Tick 値を取得する。
特性		アラーム ID で指定したアラームが満了するまでの相対値を取得する。 アラーム ID が無効な場合の時の Tick 値は未定義。 タスクレベル、割り込みレベル、フックルーチンから呼び出すこと。
状態	標準	E_OK：正常終了
	拡張	E_OS_ID：AlarmID が無効
		E_OS_NOFUNC：AlarmID は使用されていない
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		－

12.6.4. SetRelAlarm

構文		StatusType SetRelAlarm (AlarmType AlarmID , TickType increment , TickType cycle)
パラメータ(in)		AlarmID：指定アラーム ID
		increment：Tick の相対値
		cycle：周期アラーム時の設定
パラメータ(out)		なし
記述		increment ticks 経過した後に、アラームに割り当てられたタスク(拡張タスクのみ)が起動される。もしくは割り当てられたイベントが設定されるか、またはアラームコールバックルーチンが実行される。
特性		increment が 0 の時は、実装依存。 cycle の値が 0 以外なら、アラームは満了後、その時点から cycle で指定される Tick 後に満了するように再設定される。 既に使用中のアラーム ID の値を変更する場合は、いったん CancelAlarm()をすること。 タスクレベル、割り込みレベルから呼び出すこと。
状態	標準	E_OK：正常終了
		E_STATE：アラームは既に使用中
	拡張	E_OS_ID：AlarmID が無効
		E_OS_VALUE：increment , cycle の値が不正
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		increment の相対値が非常に小さい場合、タスクが ready 状態に戻る前、または、アラームコールバックの処理が戻る前にアラームは満了している可能性がある。

12.6.5. SetAbsAlarm

構文		StatusType SetAbsAlarm (AlarmType AlarmID , TickType start, TickType cycle)
パラメータ(in)		AlarmID：指定アラーム ID
		start：Tick の絶対値
		cycle：周期アラーム時の設定
パラメータ(out)		なし
記述		絶対時間と start が同値の場合、アラームに割り当てられたタスク(拡張タスクのみ)が起動される。もしくは割り当てられたイベントが設定されるか、またはアラームコールバックルーチンが実行される。
特性		cycle の値が 0 以外なら、アラームは満了後、その時点から cycle で指定される Tick 後に満了するように再設定される。 既に使用中のアラーム ID の値を変更する場合は、いったん CancelAlarm()を実行すること。 タスクレベル、割り込みレベルから呼び出すこと。
状態	標準	E_OK：正常終了
		E_STATE：アラームは既に使用中
	拡張	E_OS_ID：AlarmID が無効
		E_OS_VALUE：start , cycle の値が不正
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		start の絶対値が非常に小さい場合、タスクが ready 状態に戻る前、または、アラームコールバックの処理が戻る前にアラームは満了している可能性がある。

12.6.6. CancelAlarm

構文		StatusType　CancelAlarm　（　AlarmType　AlarmID　）
パラメータ(in)		AlarmID：指定アラーム ID
パラメータ(out)		なし
記述		アラームを取り消す。
特性		アラーム ID で指定したアラームを取り消す。 タスクレベル、割り込みレベルから呼び出すこと。
状態	標準	E_OK：正常終了
	拡張	E_OS_ID：AlarmID が無効
		E_OS_NOFUNC：AlarmID は使用されていない
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		－

12.6.7. SignalCounter

構文		StatusType　SignalCounter　（　CounterType CounterID　）
パラメータ(in)		CounterID:カウンタ ID
パラメータ(out)		なし
記述		Tick 値を OIL にて定義した分加算する。
特性		割り込みレベルから呼び出すこと。
状態	標準	E_OK：正常終了
	拡張	E_OS_ID：CounterID が無効
		E_OS_CALLEVEL：割込みレベル以外で呼び出した
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		－

12.7. Operating System Execution Control

＜概略＞

OS 実行管理するシステムサービスを提供

システムサービスはタスクレベル、カテゴリ 2 割込みレベルからの呼び出し

12.7.1. GetActiveApplicationMode

構文		AppModeType　GetActiveApplicationMode　（　void　）
パラメータ(in)		mode：アプリケーションモード
パラメータ(out)		なし
記述		現在のアプリケーションモードを取得する。
特性		StartOS()で指定したアプリケーションモードを取得する。 アプリケーションモードに依存した実装する時に使用する。
状態	標準	なし
	拡張	なし
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		－

12.7.2. StartOS

構文		void　StartOS　（　AppModeType　mode　）
パラメータ(in)		mode：アプリケーションモード
パラメータ(out)		なし
記述		オペレーティングシステムを始動させる。
特性		指定されたアプリケーションモードで始動させる。 オペレーティングシステムの外からのみ許容される。
状態	標準	なし
	拡張	なし
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		実装の特定の制限が適用できる。

12.7.3. ShutdownOS

構文		void　ShutdownOS　（　StatusType　error　）
パラメータ(in)		error：エラー
パラメータ(out)		なし
記述		システム全体を停止させる。
特性		ShutdownHook がオペレーティングシステム停止前に呼ばれる。 パラメータにエラー値を ShutdownHook に渡す。 ShutdownHook から戻った場合の処理は実装依存。 タスクレベル、割り込みレベル、フックルーチンから呼び出すこと。
状態	標準	なし
	拡張	なし
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		オペレーティングシステム内部でコール可能。 ←その場合の error 時には E_OK を設定しない。 OSEK OS と OSEKtime OS が共存するシステムの場合には、error は OSEKtime OS で受け入れられた値である。 この場合、OSEKtime 構成パラメータによって可能にされると、OSEKtime OS は OSEK OS シャットダウンの後に終了する。

12.8. Hook Routines

＜概略＞
各種フックルーチンを設定するシステムサービスを提供
フックルーチンの呼び出しは実装依存

12.8.1. ErrorHook

構文		void　ErrorHook　（　StatusType　error　）
パラメータ(in)		error：エラー
パラメータ(out)		なし
記述		システムサービスエラー時に呼ばれる。
特性		このフックルーチンはシステムサービスが E_OK 以外の値を返す時に、且つタスクレベルに戻る前に呼ばれる。 アラームからのタスク起動やイベント設定に失敗した場合にも呼ばれる。
状態	標準	なし
	拡張	なし
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		ErrorHook から呼ばれるシステムサービスが E_OK 以外を返しても、ErrorHook は呼ばれない。

12.8.2. PreTaskHook

構文		void　PreTaskHook　（　void　）
パラメータ(in)		なし
パラメータ(out)		なし
記述		タスクの実行前に呼ばれる。
特性		このフックルーチンはオペレーティングシステムが新しいタスクを実行する前に、実行状態に遷移させた後に呼ばれる。
状態	標準	なし
	拡張	なし
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		－

12.8.3. PostTaskHook

構文		void　PostTaskHook　（　void　）
パラメータ(in)		なし
パラメータ(out)		なし
記述		タスクの実行後に呼ばれる。
特性		このフックルーチンはオペレーティングシステムが現在のタスクを実行した後、タスクの実行状態ではなくなる前に呼ばれる。 実行していたタスクの TaskID を GetTaskID()で参照できる。
状態	標準	なし
	拡張	なし
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		－

12.8.4. StartupHook

構文		void　StartupHook　（　void　）
パラメータ(in)		なし
パラメータ(out)		なし
記述		オペレーティングシステム初期化後に呼ばれる。
特性		オペレーティングシステム初期化処理が終わり、 スケジューラが起動する前に呼ばれる。
状態	標準	なし
	拡張	なし
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		デバイスドライバの初期化をすることができる。

12.8.5. ShutdownHook

構文		void　ShutdownHook　(　StatusType　error　)
パラメータ(in)		error：エラー
パラメータ(out)		なし
記述		ShutdownOS()がコールされたら呼ばれる。
特性		このフックルーチンはユーザ定義されたシャットダウン機能を実行する。
状態	標準	なし
	拡張	なし
コンフォーマンス		BCC1/BCC2/ECC1/ECC2
備考		－

13. 資料

TOPPERS/OSEK カーネルにおける型情報を下記の表で記載する。

型名	サイズ	備考
TaskIdentifier	-	
StatusType	unsigned long	
TaskType	unsigned long	
TaskRefType	実装依存	ポインタ変数
TaskStateRefType	実装依存	ポインタ変数
ResourceIdentifier	-	
ResourceType	unsigned long	
EventIdentifier	-	
EventMaskType	unsigned long	
EventMaskRefType	実装依存	ポインタ変数
AlarmIdentifier	-	
AlarmType	unsigned long	
AlarmBaseRefType	実装依存	ポインタ変数
maxallowedvalue	unsigned long	
ticksperbase	unsigned long	
mincycle	unsigned long	
TickType	unsigned long	
TickRefType	実装依存	ポインタ変数
AppModeType	unsigned long	

変更履歴

Version	Date	Detail	Editor
1.00	2004/04/02	・新規作成	ヴィッツ
1.10	2006/01/17	・修正	ヴィッツ
1.20	2006/02/16	・図の追加(システムサービス分)	ヴィッツ
1.30	2006/03/14	・文章修正	ヴィッツ
1.31	2006/05/22	・文章修正	ヴィッツ
2.00	2006/05/30	・TOPPERS/OSEK 公開用にバージョンアップ	ヴィッツ