お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

お客様各位

資料中の「日立製作所」、「日立XX」等名称の株式会社ルネサス テクノロジへの変更について

2003年4月1日を以って三菱電機株式会社及び株式会社日立製作所のマイコン、ロジック、アナログ、ディスクリート半導体、及びDRAMを除くメモリ（フラッシュメモリ・SRAM等）を含む半導体事業は株式会社ルネサス テクノロジに承継されました。従いまして、本資料中には「日立製作所」、「株式会社日立製作所」、「日立半導体」、「日立XX」といった表記が残っておりますが、これらの表記は全て「株式会社ルネサス テクノロジ」に変更されておりますのでご理解の程お願い致します。尚、会社商標・ロゴ・コーポレートステートメント以外の内容については一切変更しておりませんので資料としての内容更新ではありません。

ルネサステクノロジ ホームページ（http://www.renesas.com）

2003年4月1日
株式会社ルネサス テクノロジ
カスタマサポート部

RENESAS

ご注意

安全設計に関するお願い

1. 弊社は品質、信頼性の向上に努めておりますが、半導体製品は故障が発生したり、誤動作する場合があります。弊社の半導体製品の故障又は誤動作によって結果として、人身事故、火災事故、社会的損害などを生じさせないような安全性を考慮した冗長設計、延焼対策設計、誤動作防止設計などの安全設計に十分ご留意ください。

本資料ご利用に際しての留意事項

1. 本資料は、お客様が用途に応じた適切なルネサス テクノロジ製品をご購入いただくための参考資料であり、本資料中に記載の技術情報についてルネサス テクノロジが所有する知的財産権その他の権利の実施、使用を許諾するものではありません。
2. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例の使用に起因する損害、第三者所有の権利に対する侵害に関し、ルネサス テクノロジは責任を負いません。
3. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他全ての情報は本資料発行時点のものであり、ルネサス テクノロジは、予告なしに、本資料に記載した製品または仕様を変更することがあります。ルネサス テクノロジ半導体製品のご購入に当たりましては、事前にルネサス テクノロジ、ルネサス販売または特約店へ最新の情報をご確認頂きますとともに、ルネサス テクノロジホームページ (http://www.renesas.com)などを通じて公開される情報に常にご注意ください。
4. 本資料に記載した情報は、正確を期すため、慎重に制作したものですが万一本資料の記述誤りに起因する損害がお客様に生じた場合には、ルネサス テクノロジはその責任を負いません。
5. 本資料に記載の製品データ、図、表に示す技術的な内容、プログラム及びアルゴリズムを流用する場合は、技術内容、プログラム、アルゴリズム単位で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断してください。ルネサス テクノロジは、適用可否に対する責任は負いません。
6. 本資料に記載された製品は、人命にかかわるような状況の下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。本資料に記載の製品を運輸、移動体用、医療用、航空宇宙用、原子力制御用、海底中継用機器あるいはシステムなど、特殊用途へのご利用をご検討の際には、ルネサス テクノロジ、ルネサス販売または特約店へご照会ください。
7. 本資料の転載、複製については、文書によるルネサス テクノロジの事前の承諾が必要です。
8. 本資料に関し詳細についてのお問い合わせ、その他お気付きの点がございましたらルネサス テクノロジ、ルネサス販売または特約店までご照会ください。

User's Manual

Ho7000シリーズ
オペレーティングシステム

マニュアル

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　2002.06

ご注意

1. 本製品(ソフトウエア製品及びその関連ソフトウエア製品を含む。以下、同じ。)の使用に際しては、「外国為替及び外国貿易法」等、技術輸出に関する日本及び関連諸国の関係法規の遵守が必要となります。

2. 弊社は、本製品の使用に際しては、弊社もしくは第三者の特許権、著作権、商標権、その他の知的所有権等の権利に関し、別途、個別の契約書等(マニュアルの記載を含む。以下、同じ。)にて弊社による明示的な許諾がある場合を除き、その保証または実施権の許諾を行うものではありません。また本製品を使用したことにより第三者の知的所有権等の権利に関わる問題が生じた場合、弊社はその責を負いませんので予めご了承ください。

3. 本製品およびその仕様、またはマニュアルに記載されている事柄については、将来、事前の予告なしに変更することがありますので、最終的な設計、ご購入、ご使用に際しましては、事前に最新の製品規格または仕様書(マニュアルを含む)をご確認ください。

4. 本製品の使用(マニュアル記載事項に基づくものも含む)により直接または間接に生ずるいかなる損害についても、弊社は一切の責任を負いません。また、本製品の配布に使用される搭載機器や媒体が原因の損害に対しましても、弊社は一切の責任を負いません。

5. 本製品を、宇宙、航空、原子力、燃焼制御、運輸、交通、各種安全装置、ライフサポート関連の医療機器等のように、特別な品質・信頼性が要求され、その故障や誤動作が直接人命を脅かしたり、人体に危害を及ぼす恐れのある用途向けには使用できません。お客様の用途がこれに該当するかどうか疑問のある場合には、事前に弊社営業担当迄ご相談をお願い致します。

6. 本製品を使用してお客様のシステム製品を設計される際には、通常予測される故障発生率、故障モードをご考慮の上、本製品の動作が原因での事故、その他の拡大損害を生じないようにフェールセーフ等の充分なシステム上の対策を講じて頂きますようお願い致します。

7. 本製品およびマニュアルの著作権は弊社が所有しております。お客様は、弊社から提供された本製品を、別途、個別の契約書等にて定める場合を除き、いかなる場合においても全体的または部分的に複写・解析・改変することはできないものとします。

8. お客様は、別途、個別の契約書等にて定める場合を除き、本製品のマニュアルの一部または全部を無断で使用、複製することはできません。

9. 弊社は、本製品を１台のコンピュータで使用する権利をお客様に対してのみ許諾します。よって、本製品を第三者へ譲渡、貸与、賃借することは許諾しないものとします。但し、別途、個別の契約書等にて定められる場合はその条件に従います。

10. 本製品をはじめ弊社半導体およびその関連製品についてのお問い合わせ、ご相談は弊社営業担当迄お願い致します。

目次

1 概要

1.1 はじめに

本書は、OSEK/VDX（以下、OSEK と略します）仕様 オペレーティングシステム バージョン 2.0 リビジョン 1 に準拠した Ho7000 シリーズオペレーティングシステム V1（以下、OS と略します）について記述します。**本書は OSEK 仕様を理解していることを前提として記述しています。**
OS をご使用になる前に本書をよく読んで理解してください。また、次の関連ドキュメントもお読みの上、理解してください。

・“OSEK/VDX Operating System”, Version 2.0 revision 1 （OSEK/VDX 運営委員会）
・本製品のリリースノート
・SuperH RISC engine C/C++コンパイラパッケージマニュアル
・使用する SH マイコンのハードウェアマニュアル、プログラミングマニュアル

1.2 特長

- OS は静的に構築されます。必要なタスク、リソースなどのシステムオブジェクト（3.3節参照）の数はユーザによって静的に指定されます。

- OS は、ANSI C 仕様で定義されたアプリケーションプログラムインターフェースの提供によりアプリケーションの移植性を支援します。

- OS は、アプリケーションのリアルタイム、マルチタスク動作のために以下を提供します。

– **タスク管理**
– **スケジューリング処理**
– **割り込み管理**
– **リソース管理**
– **イベント管理**
– **カウンタ、アラーム管理**
– **エラー処理**

2 OS アプリケーションの構築

OS アプリケーションの構築のために以下のファイルが必要です。

- **カーネルライブラリ：**カーネルライブラリは OS の機能を含んだ１つのオブジェクトファイルです。中心的な機能（タスク管理、スケジューラ等）は常に含まれますが、必須でない機能（リソース管理、イベント管理等）はリンケージエディタによって自動的に結合されます。

- **OS 構築ファイル：**これらのファイルは、タスク、リソースなどのシステムオブジェクトに関する情報を登録します。ファイルは、OS コンフィギュレータによって自動的に生成されます。

- **アプリケーションプログラムファイル：**ユーザアプリケーションコードは C 言語またはアセンブリ言語で記述します。

図 2.1に OS アプリケーションの構築手順を示します。

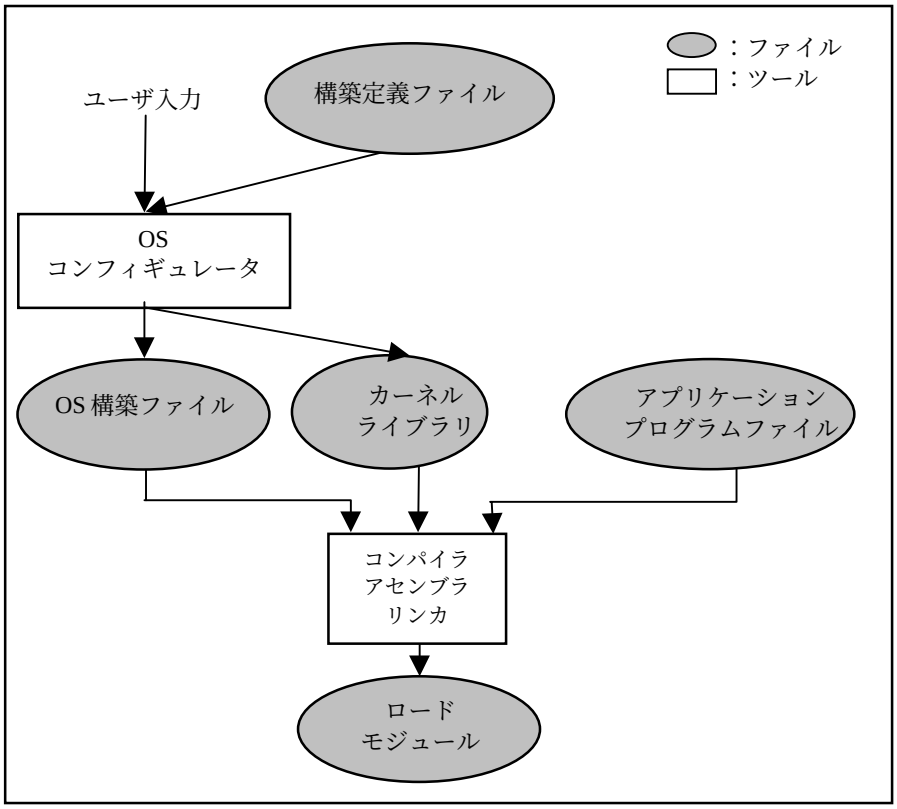

図 2.1 OS アプリケーション構築手順

コンフィギュレータは、ユーザ入力または構築定義ファイルをもとに OS 構築ファイルおよびカーネルライブラリファイルを生成します。これらのファイルはアプリケーションプログラムファイルと共にコンパイル、アセンブル、結合され、ロードモジュールを生成します。構築定義ファイルは OSEK 実装言語フォーマット（*.oil）または独自フォーマット（*.ocf）の 2 種類があります。

OS 構築ファイルの生成方法については OS コンフィギュレータのヘルプファイルを参照してください。

3 オペレーティングシステム機能

3.1 処理レベル

以下の 3 種類の処理があります:

- 割り込みサービスルーチン（以下、ISR と略します）
- OS
- タスク

最も高い処理優先度は ISR が実行される割り込みレベルに割り当てられます。タスク内で割り込みマスクレベルが０以外の場合も割り込みレベルに含まれます（これは**独自仕様**です）。オペレーティングシステムの処理レベルは、割り込みレベル以下です。 最低優先度はアプリケーションが実行されるタスクレベルです。 これを図 3.1に示します。

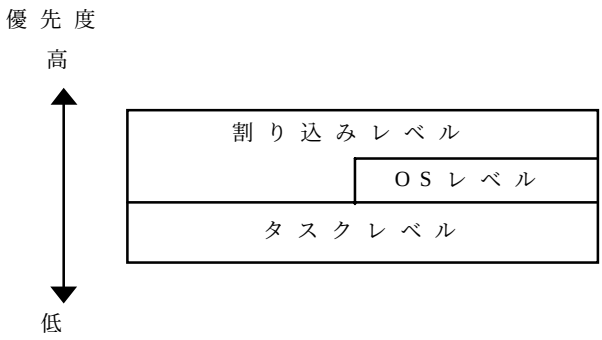

図 3.1 OSEK 処理レベル

3.2 制御

OS はアプリケーションのための以下の特長を持っています。

- **スケジューリングカーネル:**タスクに CPU を割り付ける機構を提供します。
- **タスク管理:** タスクを制御するシステムサービスを提供します。
- **割り込み管理:** 割り込みを制御するシステムサービスを提供します。
- **リソース管理:** タスク間で共有されたリソースアクセスの同期を制御するシステムサービスを提供します。
- **イベント管理:** イベントを制御するシステムサービスを提供します。
- **カウンタ管理:** カウンタを制御するシステムサービスを提供します。
- **アラーム管理:** アラームを制御するシステムサービスを提供します。
- **エラー処理、フックルーチン:** エラー処理と特定のタイミングで呼ばれるフックルーチンを制御するメカニズムを提供します。

3.3 アプリケーションモード

OS は、異なるモードでアプリケーションを開始することができる機能（アプリケショーンモード）を提供します。たとえばこれを使用することによって、通常時に動作する OS アプリケーションと保守時に動作する OS アプリケーションを区別することができます。

以下のシステムオブジェクトをアプリケーションモード毎に有効かどうか選択することができます。

- **タスク**
- **リソース**
- **アラーム**
- **カウンタ**
- **ISR**

システムオブジェクトが他のシステムオブジェクトと依存関係を持つ場合、選択したアプリケーションモードにおいてどちらのシステムオブジェクトも有効にしてください。例えば、あるアプリケーションモードにおいて有効なタスクがリソースを使用する場合、リソースもそのアプリケーションモードで有効にしなければなりません。また、使用しないオブジェクトも OS 構築ファイル生成時にコンフィギュレータによってエラーチェックの対象となりますので、ご注意ください。
OS が開始された後はアプリケショーンモードを変更することはできません。

3.4 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属性により決定されます。

– タスク起動のキューイング回数（多重起動回数）
– タスクタイプ（基本または拡張）
– 優先度毎のタスク数

コンフォーマンスクラスの種類を以下に示します。

- **BCC1** (基本タスクのみ、1 優先度につき 1 タスクおよび 1 回の起動、 タスクの優先度はすべて異なる)

- **BCC2** (BCC1 に、1 優先度につき 1 タスク以上および多重起動を加えたクラス)

- **ECC1** (BCC1 に拡張タスクを加えたクラス)

- **ECC2** (BCC2 に拡張タスクを加えたクラス)

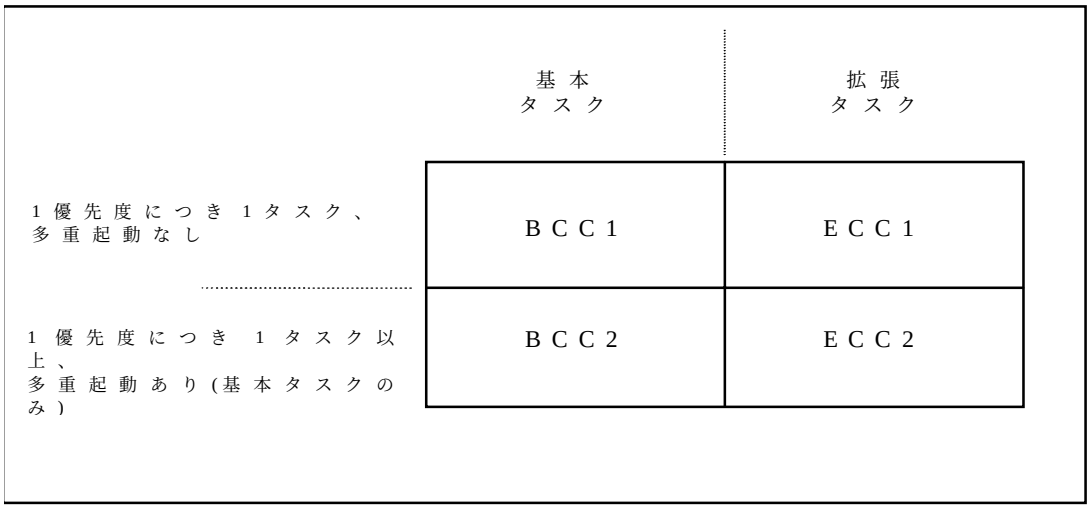

図 3.2 OS コンフォーマンスクラス

3.5 制限値

表 3.1 に OS の制限値を示します。

表 3.1 OS 制限値

項目	最大数			
	BCC1	BCC2	ECC1	ECC2
タスク数	1023			
アクティブタスク数	1023 (基本タスクは 128)			
優先度数	128			
優先度毎のタスク数	1	1023	1	1023
タスク多重起動回数 (拡張タスクは常に 1)	1	127	1	127
タスク毎のイベント数	8			
アラーム数	127			
カウンタ数	127			
リソース数 [1]	1	127		
アプリケーションモード数	8			
割り込みサービスルーチン数	256			

[1] OS で標準提供される **RES_SCHEDULER** リソース（リソース ID=0）を含みます。

4 タスク管理

4.1 タスク概念

複雑な制御ソフトウェアはリアルタイム要求に応じて細分化することが可能です。それはタスクを使用することで実現できます。OS はタスクスケジューラによってタスクの同期および非同期実行を実現します。

タスクは、OS のスタートアップ時、あるいは他のタスクによって起動できます。起動したタスクは終了することができます。あるいは、終了せず無限ループしても構いません。一度タスクが終了すると、他のタスクからの起動やアラーム満了によってリスタートします。

2 つの異なるタスクタイプを OS は提供します。

- 基本タスク
- 拡張タスク

基本タスクは次の場合にプロセッサを解放します。

- タスクが終了したとき
- より高優先度のタスクを実行しているとき
- 割込みが発生して ISR へ遷移したとき

拡張タスクは、上記に加えイベントを待つことによりプロセッサを解放することができます。
拡張タスクを多重起動することはできません。

4.2 タスク状態

4.2.1 はじめに

プロセッサは一つのタスクしか実行できないため、複数のタスクが競合する場合、タスクは様々な状態に遷移します。以下に基本タスク、拡張タスクの状態を示します。

4.2.2 基本タスク

基本タスクには、３つのタスク状態があります。

- ***running (実行状態)***
  タスクにプロセッサが割り当てられ、命令が実行されます。１つのタスクだけがこの状態になることができます。以下の他の状態は、複数のタスクが同じ状態になることができます。

- ***ready (実行可能状態)***
  タスクは、実行状態に遷移するための条件が整い、プロセッサの割り当てを待っています。スケジューラは、実行可能タスクの中から次に実行されるタスクを決定します。

- ***suspended (休止状態)***
  タスクは起動していません。この状態からタスクを起動することが可能です。

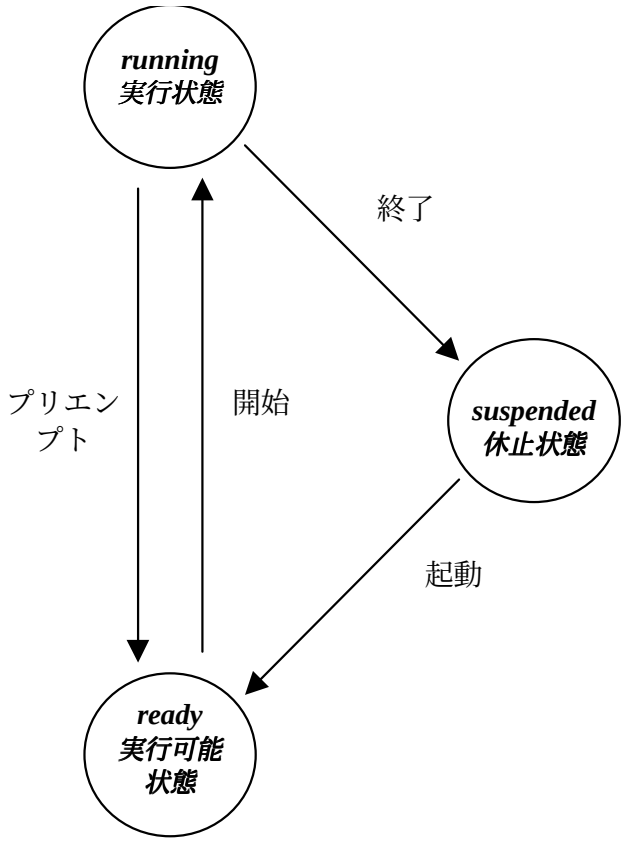

図 4.1 基本タスク状態

表 4.1 基本タスクの状態と遷移

遷移	遷移前の状態	遷移後の状態	説明
起動	休止状態	実行可能状態[2]	システムサービスによってタスクが レディキューに加えられます。実行状態に遷移するとき、タスクの先頭命令から実行されます。
開始	実行可能状態	実行状態	スケジューラで選択された実行可能タスクが実行されます。
プリエンプト	実行状態	実行可能状態	スケジューラは別のタスクを開始し、実行可能状態に遷移します。
終了	実行状態	休止状態	システムサービスにより休止状態へ遷移されます。

[2] タスクが休止状態でない場合（多重起動の場合）は、タスクの状態はすぐに変更されません。この場合、起動要求が記録され、後で実行されます。

4.2.3 拡張タスク

拡張タスクには、4つのタスク状態があります。

- ***running (実行状態)***
  タスクにプロセッサが割り当てられ、命令が実行されます。１つのタスクだけがこの状態になることができます。以下の他の状態は、複数のタスクが同じ状態になることができます。

- ***ready (実行可能状態)***
  タスクは、実行状態に遷移するための条件が整い、プロセッサの割り当てを待っています。スケジューラは、実行可能タスクの中から次に実行されるタスクを決定します。

- ***waiting (待ち状態)***
  タスクは少なくとも1つのイベントがセットされるのを待っています。

- ***suspended (休止状態)***
  タスクは起動していません。この状態からタスクを起動することが可能です。

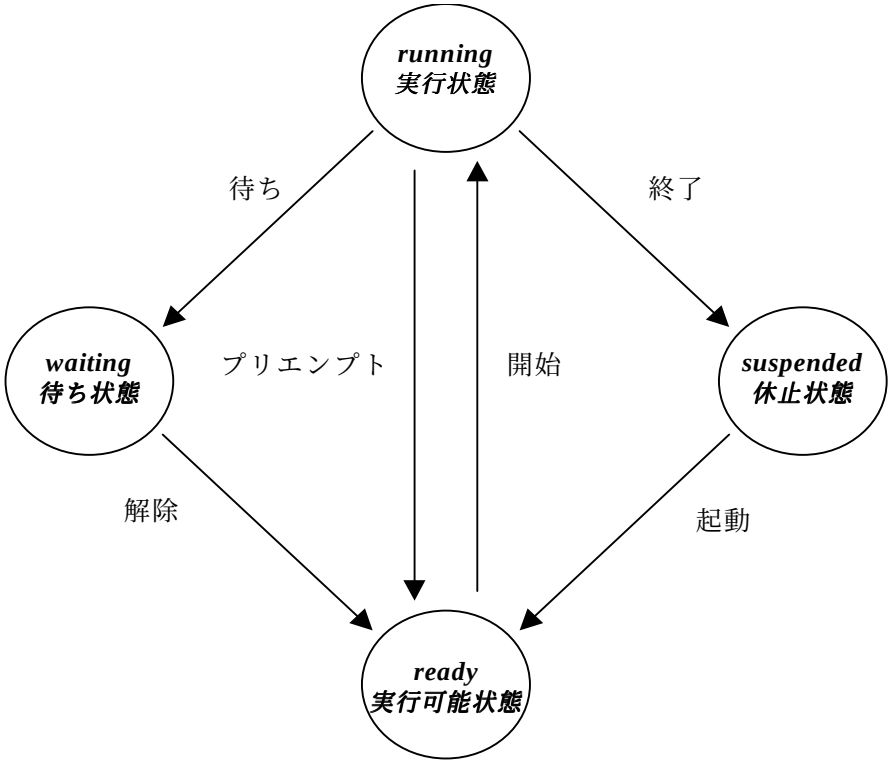

図 4.2 拡張タスク状態

表 4.2 拡張タスクの状態遷移

遷移	遷移前の状態	遷移後の状態	説明
起動	休止状態	実行可能状態	システムサービスによってタスクが レディキューに加えられます。実行状態に遷移するとき、タスクの先頭命令から実行されます。
開始	実行可能状態	実行状態	スケジューラで選択された実行可能タスクが実行されます。
待ち	実行状態	待ち状態	イベントを要求します。システムサービスにより待ち状態へ推移されます。
解除	待ち状態	実行可能状態	タスクが待つイベントが１つ以上セットされた場合、実行可能状態へ遷移されます。
プリエンプト	実行状態	実行可能状態	スケジューラは別のタスクを開始し、実行可能状態に遷移します。
終了	実行状態	休止状態	システムサービスにより休止状態へ遷移されます。

4.3 タスクの同期

基本タスクは、待ち状態を持ってないため同期をとるポイントはタスクの開始時、および終了時だけです。基本タスクの同期をとる場合はアプリケーションで行なう必要があります。

拡張タスクは、イベントを使用することによって同期をとることができます。

4.4 タスクの起動と終了

タスクの起動は、カーネル初期化時の自動起動、またはシステムサービスを使用して行ないます。

基本タスクは、コンフォーマンスクラスによって複数回起動することができます。この多重起動の最大数はシステム生成時に定義されます。多重起動が最大数に到達していない場合、起動順序を保持するため優先度毎のファーストインファーストアウト(FIFO)キューに加えられます。

タスクの終了はタスク自身で終了することのみ可能です。OS は、タスクの終了とともに新たにタスクを起動できるシステムサービスも提供します。タスクを終了する場合は TerminateTask または ChainTask システムサービスを必ず発行してください。

4.5 タスク優先度

各タスクは、0 から 127 までの優先度を持っています。最低の優先度は 0、最高の優先度は 127 です。この優先度は、システム生成時に静的に割り当てられ、スケジューラが次に実行するタスクを決定するために使用します。動的な優先度管理のしくみはサポートされません。しかし、優先度リソース管理機能は、より高い優先度を持つタスクを扱うことができます。詳細は7.2 節を参照してください。

4.6 タスクのプリエンプティブ属性

各タスクは**プリエンプティブ**または**非プリエンプティブ**を選択することができます。自優先度が高いため、現在実行中のタスクに替わって CPU を占有することをプリエンプトと呼びます。プリエンプトを許可するタスクはプリエンプティブタスク、許可しないタスクは非プリエンプティブタスクです。

プリエンプティブタスク実行中に、より高優先度のタスクが実行可能状態になった場合、スケジューラは高優先度のタスクに制御を渡します。

非プリエンプティブタスク実行中に、より高優先度のタスクが実行可能状態になった場合、タスクの切り換えは発生しません。スケジューラは、非プリエンプティブタスクが終了する、待ち状態になる、または明示的にスケジューラを呼び出したときのみ高優先度のタスクに制御を渡します。

4.7 タスクスタック

タスクタイプに依存してスタックが割り当てられます。以下に、各タスクタイプのスタック割り当て方法を示します。

- **基本タスク**
  すべての基本タスクは一つのスタックを共有します。基本タスクが別の基本タスクによってプリエンプトされる場合、実行する次のタスクはプリエンプトされたタスクの現在のスタックポインタを使用します。プリエンプトされたタスクが再び実行する前に、すべてのより高優先度の基本タスクが実行するため、タスクスタックが上書きされることはありません。

  留意事項: 基本タスクスタックサイズは優先度数に影響します。優先度が少なければ基本タスクスタックサイズも減少します。

- **拡張タスク**
  拡張タスクはそれぞれのスタックで実行します。基本タスクとは異なり、待ち状態に入ることができるため、スタックは共有できません。

4.8 タスクのシステムサービス

表 4.3にタスクを操作するシステムサービスを示します。

表 4.3 タスクシステムサービス

システムサービス	**機能**
ActivateTask	タスクの起動
TerminateTask	現在タスクの終了
ChainTask	タスクの起動と現在タスクの終了
Schedule	より高優先度タスクへ切り換えるためのスケジューラ呼出し（高優先度タスクが存在する場合）
GetTaskID	現在タスクのタスク ID の取得
GetTaskState	タスク状態の取得

5 スケジューラ

5.1 概要

OS は、タスク優先度およびプリエンプティブ属性に基づいたタスクスケジューリングを提供します。これは、タイムスライスアルゴリズム(例えば、ラウンドロビンスケジューリング)をもつ他のオペレーティングシステムとは対照的です。OS は、実行および実行可能状態のすべてのタスクを保持するためにアプリケーションに存在するタスク優先度ごとにレディキューをもっています。各レディキューは起動順序を保持するためにファーストインファーストアウト（FIFO）キューになっています。 図 5.1に例を示します。

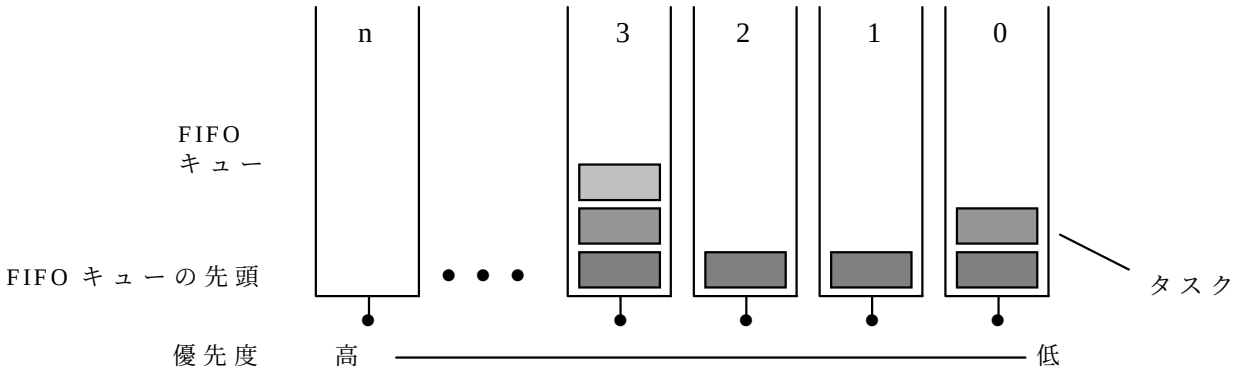

図 5.1 レディキュー

スケジューラは、次に処理するタスクを決定するために以下を実行します。

- 実行可能状態タスクの中から最高優先度タスクを決定する。
- その優先度の中でレディキューの先頭にあるタスクに制御を渡す。

レディキューの中にタスクがない場合、OS はアイドルモードに入ります。

以下の点に注意してください。

待ち状態を解除されたタスクは、その優先度のレディキューの最後に加えられます。

OS は３つのスケジューリングプリエンプティブ属性を提供します。

- 非プリエンプティブスケジューリング
- フルプリエンプティブスケジューリング
- ミックスプリエンプティブスケジューリング

これらを以下の節に示します。

5.2 非プリエンプティブスケジューリング

非プリエンプティブスケジューリングではすべてのタスクが非プリエンプティブです。非プリエンプティブタスクは、再スケジューリングの次の時点まで、高い優先度のタスクの開始を遅らせます。

非プリエンプティブタスクの場合、再スケジューリングは以下の場合に起こります。

- タスクの正常終了
- スケジューラ呼び出し（Schedule システムサービス発行）
- 待ち状態へ遷移

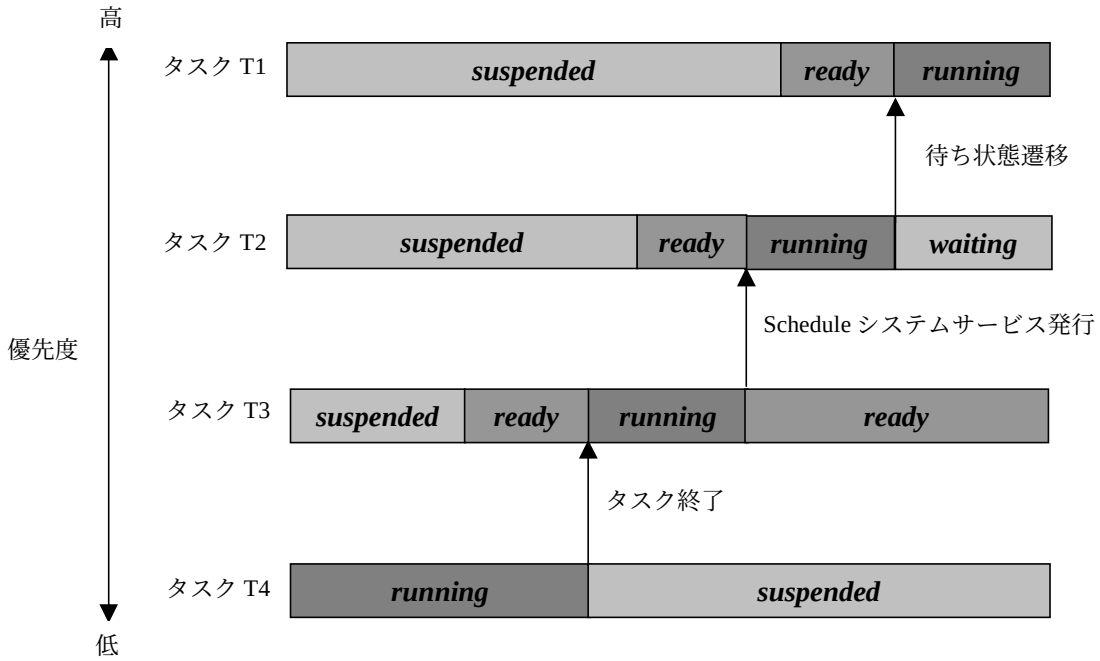

図 5.2 非プリエンプティブスケジューリング

5.3 フルプリエンプティブスケジューリング

フルプリエンプティブスケジューリングではすべてのタスクがプリエンプティブです。高優先度タスクが実行可能状態になると、直ちに現在のタスクは切り換えられます。フルプリエンプティブスケジューリングの待ち時間は、低優先度タスクの実行時間に依存しません。常にタスクの再スケジューリングされるため、他のタスクと共有するデータへのアクセスは同期をとる必要があります。

図 5.3では、低優先度タスク T2 は、高優先度タスク T1 のスケジューリングを遅らせません。

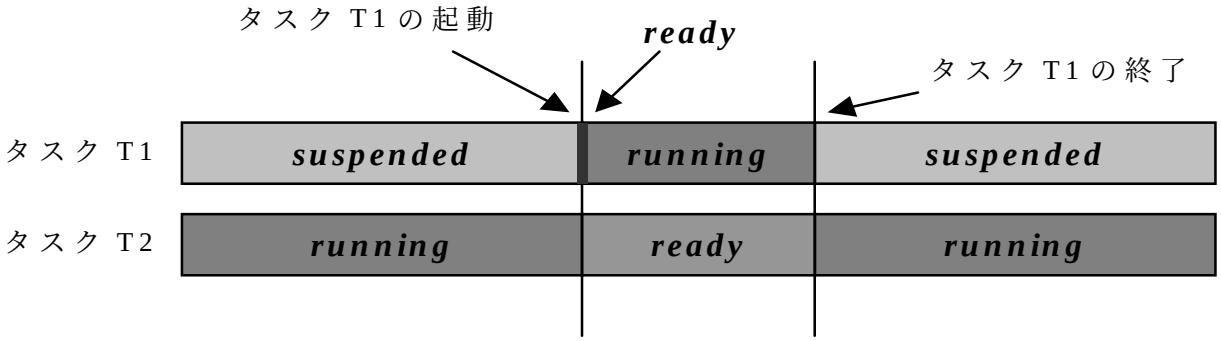

図 5.3 フルプリエンプティブスケジューリング

タスクを切り換えたくない場合は、*GetResource* システムサービスによってスケジューラを一時的にロックすることができます。.

プリエンプティブタスクの場合、再スケジューリングは以下の場合に起こります。

- タスクの正常終了
- 待ち状態へ遷移
- より高い優先度のタスク起動
- 待ちタスクへのイベントセット
- アラーム満了時のより高い優先度のタスク起動またはイベントセット
- リソース解放
- ISR からの復帰

5.4 ミックスプリエンプティブスケジューリング

ミックスプリエンプティブスケジューリングでは、プリエンプティブタスクと非プリエンプティブタスクが同じシステムの中に混在します。プリエンプティブタスク実行中はフルプリエンプティブスケジューリングが使用され、非プリエンプティブタスク実行中は非プリエンプティブスケジューリングが使用されます。

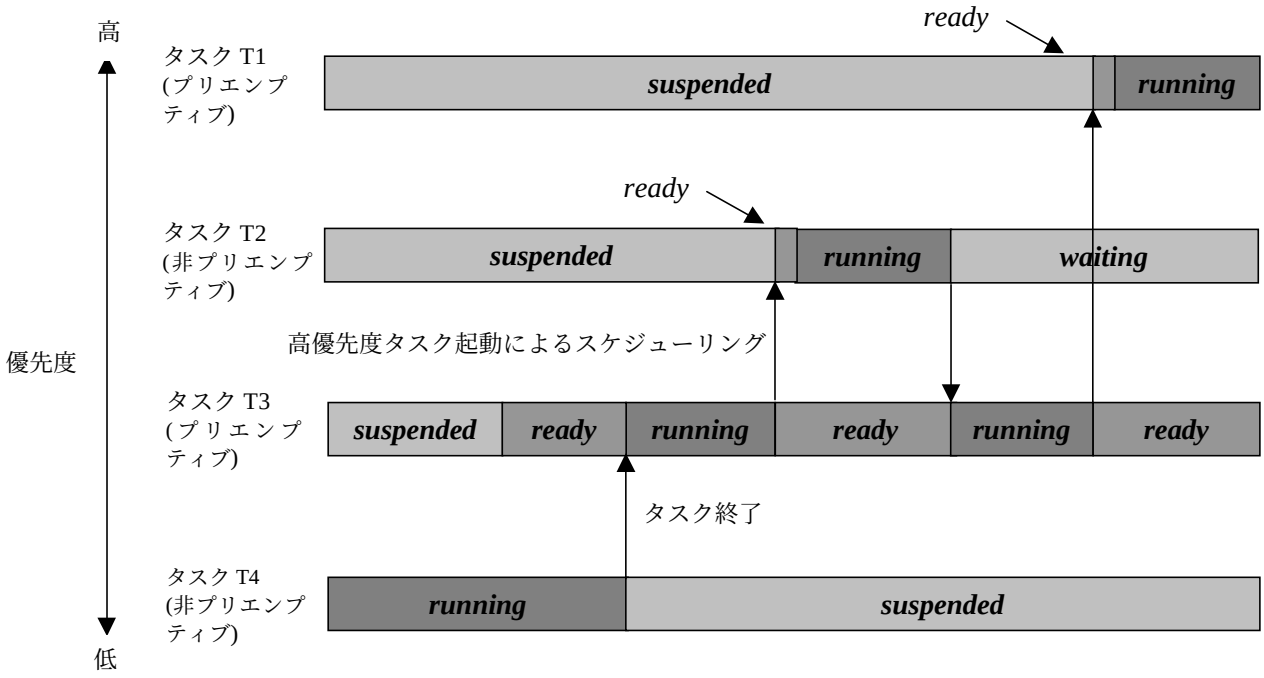

図 5.4 ミックスプリエンプティブスケジューリング

5.5 割り込みマスクレベルとタスクプリエンプション

タスクは割り込みマスクレベル（CPU のステータスレジスタの i ビット）が０の場合にプリエンプトされます。割り込みマスクレベルが０以外の場合、タスク切り換えは発生しません。この場合、割り込みマスクレベルを０にしたときにプリエンプトされます。

6 割り込み管理

6.1 割り込みカテゴリ

割り込み処理は以下の 3 つのカテゴリに分類されます

- *割り込みカテゴリ 1*

  カテゴリ 1 割り込みの ISR はシステムサービスを使用できません。ISR 終了後、割り込み発生点の命令から処理が継続されます。このカテゴリは OS が介在しません。したがって、割り込みが発生するとすぐに ISR へ分岐します。このカテゴリは最も速い割り込みです。

- *割り込みカテゴリ 2*

  割り込み発生時、OS が介在します。このとき処理レベル（3.1節参照）が割り込み処理レベルへ切り換えられ、スタックが変更されます。これを割り込み前処理と呼びます。前処理が終了した後 ISR へ処理が移行します。ISR 内で使用可能なシステムサービスは「付録 A.3 」を参照してください。

- *割り込みカテゴリ 3*

  この割り込みは OS の旧バージョンとの互換性のために存在します。割り込みカテゴリ 3 の構築や割り込み処理レベルへの切り換えおよび復帰のための *EnterISR* システムサービス、*LeaveISR* システムサービスの発行も可能ですが、この割り込みはカテゴリ 2 の割り込みと同じ構築、同じ動作をします。*EnterISR* システムサービス、*LeaveISR* システムサービスは一切処理を実行しないため、呼び出す必要はありません。以降、割り込みカテゴリ 3 については割り込みカテゴリ 2 を参照してください。

留意事項: この動作は独自仕様です。

図 6.1に割り込みカテゴリ 1 の流れを示します。

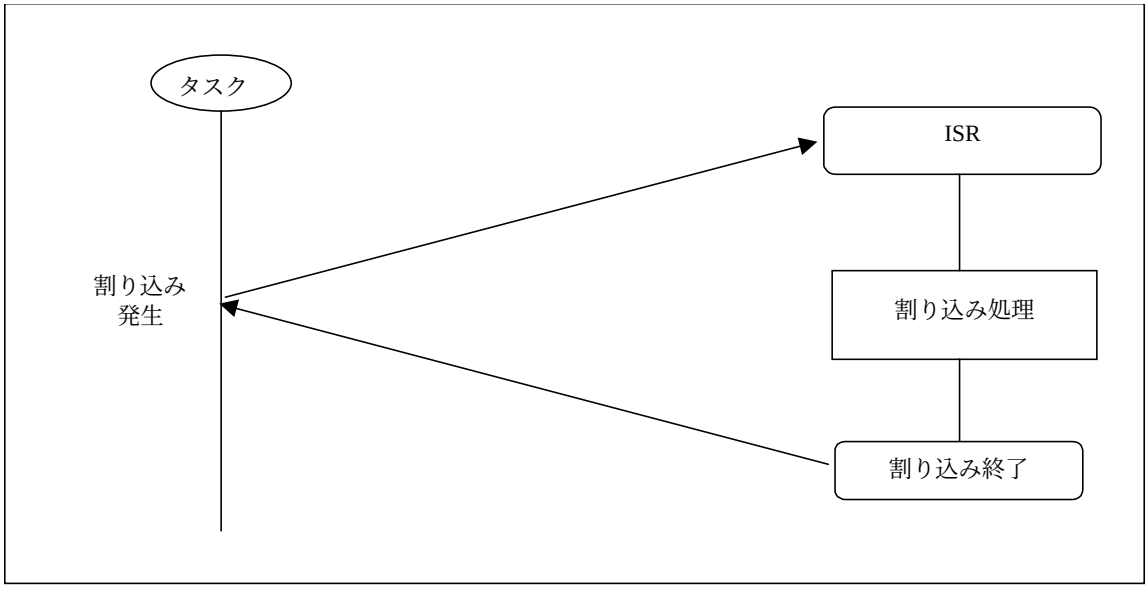

図 6.1 割り込みカテゴリ 1 の流れ

図 6.2に割り込みカテゴリ 2 の流れを示します。

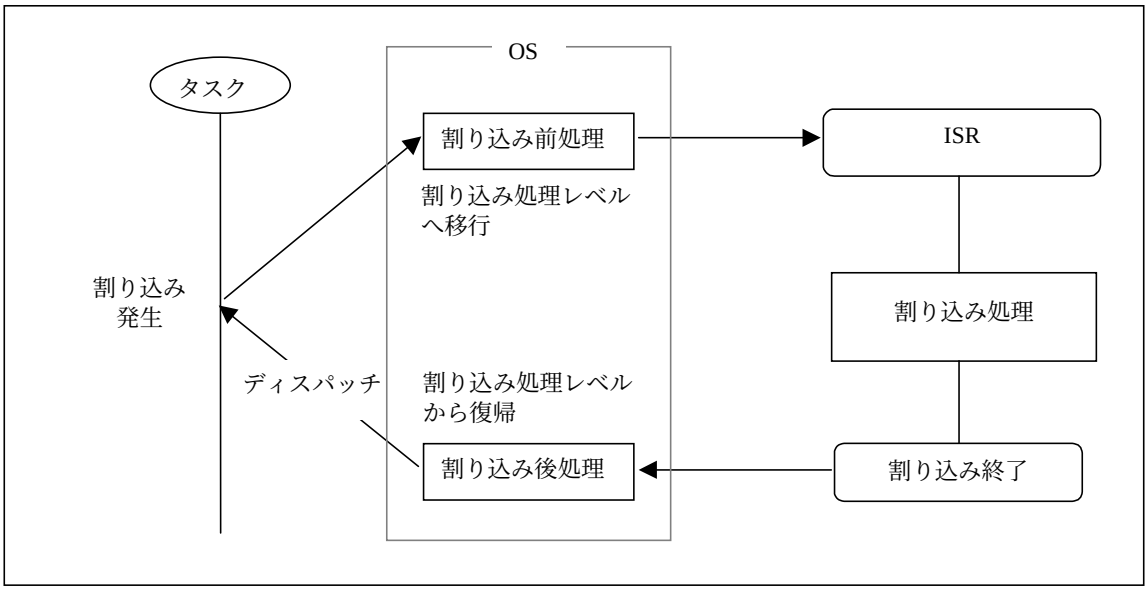

図 6.2 割り込みカテゴリ 2 の流れ

図 6.3に各カテゴリの ISR の記述内容を示します。

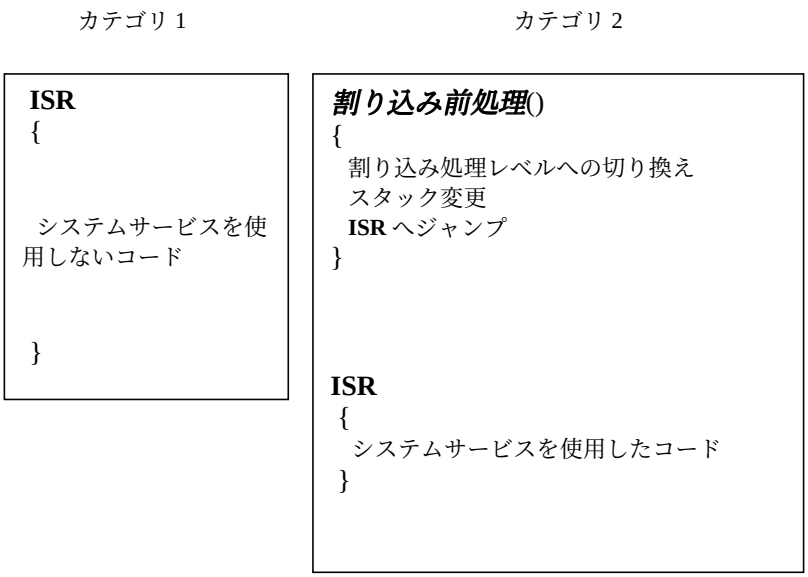

図 6.3 各カテゴリの ISR 記述

ISR 内では再スケジューリングは起こりません。再スケジューリングは、プリエンプティブタスクがカテゴリ 2 の割り込みによって中断され、ISR 終了時にスケジューラがロックされてない、他の割り込みが発生してない、かつ復帰先のタスクの割り込みマスクレベルが 0 の場合に起こります。

6.2 割り込みマスクレベルと割り込み制御

EnableInterrupt および DisableInterrupt システムサービスは割り込みの許可を操作するために提供されます。

割り込み制御方式として、割り込みマスクレベル（OS コンフィギュレータの OS ページで Use mask level）を選択した場合、これらのシステムサービスは、CPU のステータスレジスタ(SR)中の割り込みマスクビット(i)を変更することにより、割り込みを有効あるいは無効にします。なお、割り込みが無効(0 以外)の場合、タスク切り換えは発生しません。この場合、EnableInterrupt で全ての割り込みを有効(0)にしたときにプリエンプトされます。

割り込み制御方式として、割り込み要因（OS コンフィギュレータの OS ページで Use source）を選択した場合、これらのシステムサービスは、CPU の割り込み要求許可ビット、あるいは割り込み優先レベル設定レジスタ(IPR)を変更することにより、割り込みを有効あるいは無効にします。

6.2.1 割り込みマスクレベル方式

- *EnableInterrupt*(<Descriptor>)は、<Descriptor>より高いレベルの割り込みを有効にします。<Descriptor>の値の範囲は 16 の倍数で H’00000000 から H’000000f0 です。H’00000000 を指定したとき OS は再スケジューリングを行います。

- *DisableInterrupt*(<Descriptor>)は、<Descriptor>以下のレベルの割り込みを無効にします。<Descriptor>の値の範囲は 16 の倍数で H’00000000 から H’000000f0 です。H’00000000 を指定した場合、すべての割り込みが有効になります。リターン値として、割り込み無効前の SR の割り込みマスクビットの値が返ります。

> **留意事項**: これらのシステムサービスは、リアルタイム性を重視した高速化のため、OSEK OS 仕様 Version 2.0 revision 1 に準拠していません。エラーコードが返る条件は存在しないため **E_OS_NOFUNC** は返りません。

6.2.2 割り込み要因方式

- *EnableInterrupt*(<Descriptor>)は、<Descriptor>で指定した ISR ID に対応する割り込み要因の割り込み要求許可ビット、あるいは割り込み優先レベル設定レジスタ(IPR)を有効値に設定します。

- *DisableInterrupt*(<Descriptor>)は、<Descriptor>で指定した ISR ID に対応する割り込み要因の割り込み要求許可ビット、あるいは割り込み優先レベル設定レジスタ(IPR)を無効値に設定します。

<Descriptor>（ISR ID）は、ISR 名の最後に”_ID”を付加した名称を指定してください。
各割り込みに対して割り込み要因をコンフィギュレータ上で設定します。標準で提供される割り込み要因に対する設定値は、12.3.4を参照して下さい。また、この他にユーザ定義の要因（CUSTOM0 ～ CUSTOM31）を選択することもできます。この場合、割り込み制御はユーザ定義要因関数で行われます（12.3.5 参照）。

留意事項: これらのシステムサービスは、使い勝手向上のため、OSEK OS 仕様 Version 2.0 revision 1 に準拠していません。
Ho7047 は割り込み要因方式をサポートしていません。割り込み制御として割り込みマスクレベル方式のみ使用可能です。

ISR 実行中に割り込みマスクレベルを割り込み発生時のレベルより低くしないでください。
タスクは割り込みマスクレベルを変更することが可能です。タスク実行中の割り込みマスクレベルが 0 以外の場合、処理レベルはタスクレベルではなく割り込み処理レベルになります。また、割り込みマスクレベルが OS マスクレベルより高い場合、システムサービスは発行できません。

カテゴリ 1 の割り込みマスクレベルは、OS によって制御されないため、OS マスクレベル[3]以上でなければなりません。したがって、カテゴリ 1 割り込みのマスクレベルは、カテゴリ 2 割り込みの最高マスクレベル以上になるように設定してください。

図 6.4に割り込みレベルを示します。

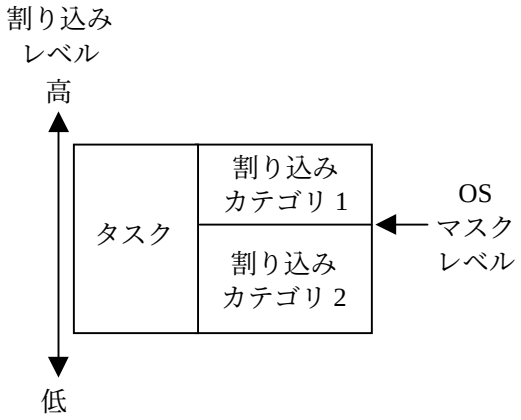

図 6.4 割り込みレベル

[3] OS の割り込みマスクレベルは、カテゴリ 2 割り込みの中で最も高い割り込みレベルにコンフィギュレータによって自動的に設定されます。

6.3 割り込みの分類

割り込みは、NMI またはその他の割り込み（外部割込み、内蔵周辺モジュール割り込み）の２つに分類されます。ソフトウェアトラップは割り込みではなく、例外として扱われます。
以下に割り込みの分類を示します。

6.3.1 ノンマスカブル割り込み

ノンマスカブル割り込み（**NMI**）はマスクされず、常に受け付けられます。この割り込みは割り込みカテゴリ 1 のみです。

6.3.2 その他の割り込み

その他の割り込みは上記以外の内蔵周辺モジュール割り込みおよび外部割り込みです。

6.4 割り込みスタック

以下のスタックが割り込みに割り付けられます。各割り込みが使用するスタックサイズはコンフィギュレータ上で指定します。

6.4.1 カテゴリ 2 割り込み

割り込みカテゴリ 2 の全ての ISR は基本タスクスタックと同じ領域のスタックを共有します。コンフィギュレータ上で指定した ISR のスタックサイズが基本タスクスタック上に確保されます。このカテゴリの割り込みが発生すると、OS は割り込みのためにスタックを切り換えます。カテゴリ 2 割り込みのネストが生じる場合、OS はスタックを切り換えません。

6.4.2 カテゴリ 1 割り込み

割り込みカテゴリ 1 の各割り込みは、同じ割り込みマスクレベルのスタックを共有します。

留意事項: OS は、割り込みカテゴリ 1 の制御を行いません。割り込みカテゴリ 1 のスタックは他のスタックとは独立しています。

6.4.3 NMI 割り込み

NMI の ISR は、割り込まれたプログラムと同じスタックを使用します。

6.5 例外

表 6.1 に示す例外はオペレーティングシステムによって致命的なエラーとして扱われます。

表 6.1 例外

例外	ハンドラ名
未定義例外	_OSEKUnhandledExceptionError
一般不当命令	_OSEKIllegalInstructionError
スロット不当命令	_OSEKIllegalSlotInstructionError
CPU アドレスエラー	_OSEKIllegalCPUAddressError

これらの例外をベクタテーブルに登録することができます。 これらの例外が発生すると OS はシャットダウン処理を行います。
上記以外の例外やソフトウェアトラップもベクタテーブルに登録することができます。 この場合、例外ハンドラを用意してください。これらの例外は、割り込まれたプログラムと同じスタックを使用します。

6.6 割り込みのシステムサービス

表 6.2 に割り込みを操作するためのシステムサービスを示します。

表 6.2 割り込みのシステムサービス

システムサービス	機能
EnableInterrupt または EnableInterruptMask	割り込みの有効化（割り込みマスクレベル方式）
EnableInterrupt または EnableInterruptSource	割り込みの有効化（割り込み要因方式）
DisableInterrupt または DisableInterruptMask	割り込みの無効化（割り込みマスクレベル方式）
DisableInterrupt または DisableInterruptSource	割り込みの無効化（割り込み要因方式）
GetInterruptDescriptor または GetInterruptDescriptorMask	割り込み状態の取得（割り込みマスクレベル方式）
GetInterruptDescriptor または GetInterruptDescriptorSource	割り込み状態の取得（割り込み要因方式）

7　リソース管理

7.1 概要

リソースを使用することにより、優先度リソースを獲得、開放し、プリエンプティブタスクの同期をとることができます。

リソース占有中、タスクの優先度が一時的に高くなります。これによって複数タスクによる同じ資源の占有を防止することができます。

7.2 優先度リソース管理機能

リソースの優先度割り当ては、リソースを獲得するタスクの優先度を一時的に高くします。これにより複数のタスクから資源を同時に占有することを不可能にします。このリソース優先度は、システム構築時に以下のように決定されます。

1. リソースを共有するタスクの最高優先度以上、かつ
2. リソースを共有するタスクの最高優先度より高い他のタスクの優先度より低い

図 7.1 にリソースの優先度割り当てを示します。優先度はタスク T0 が高く、T4 が低いことを示します。タスク T1 と T4 は同じリソースを共有します。これらのタスクがリソースを獲得する際、優先度がタスク T1 以上かつタスク T0 以下に上げられます。タスク T0,T2,T3 はリソースを共有しません。

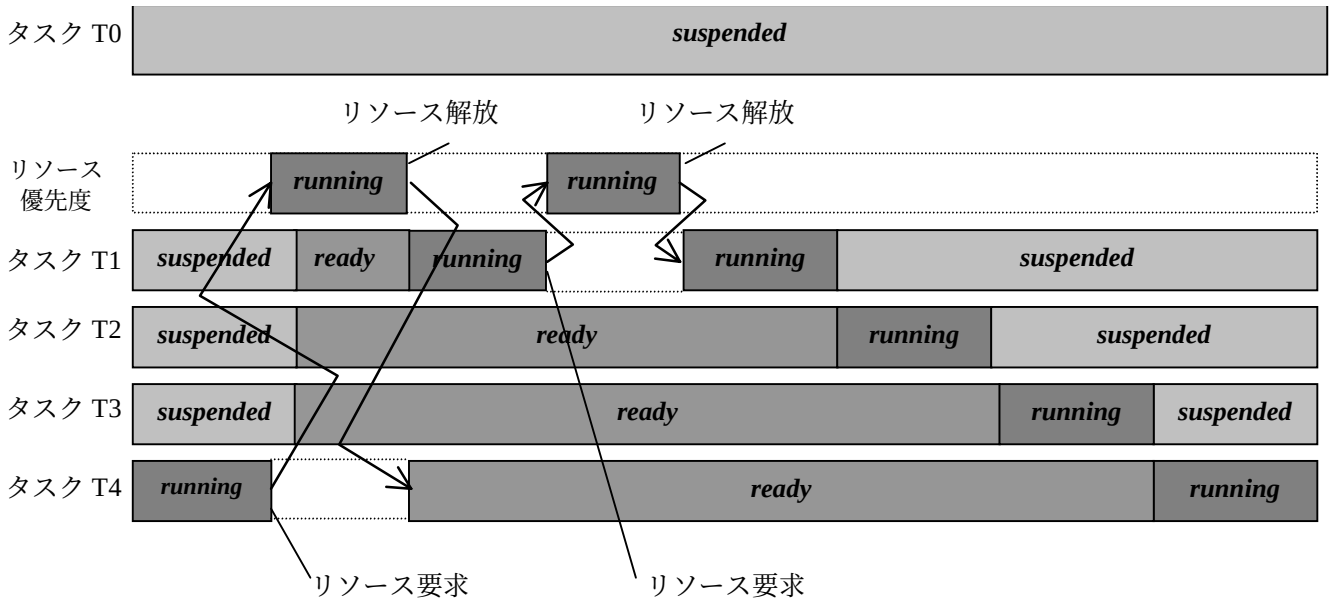

図 7.1 リソースの優先度割り当て

7.3 リソース占有のネスト

タスクが複数リソースを占有する場合、最後に獲得したリソースから順番に解放してください。
現在のタスクの優先度より低いリソースを獲得した場合、OSEK 仕様では優先度が下げられ、E_OK が返ります。しかし、本 OS はタスクの優先度を下げません。これによりデッドロックおよび優先度反転を防止します。また、このとき拡張エラーステータスの場合、独自のエラーを返します。

留意事項: この動作は独自仕様です。

7.4 タスク終了時のリソース占有

タスク終了前にリソースは解放してください。解放しなかった場合、拡張エラーステータスでは OS がエラーを返します。標準エラーステータスの場合、システムの動作は保証されません。

7.5 スケジューラリソース

タスク実行中、タスク切り換えが発生しないようにスケジューラをロックすることができます。これは OS で標準提供される **RES_SCHEDULER** リソースを使用することにより可能です。以下の点に注意してください。

1. **RES_SCHEDULER** リソースの獲得ではタスクの優先度は変更されません。

2. **RES_SCHEDULER** リソース解放タイミングは、他のリソースに影響されません。タスクが終了する前であればいつでも解放できます。

3. **RES_SCHEDULER** リソースの占有時、*Schedule* システムサービスを発行してもタスク切り換えは発生しません。

7.6 ECC1 コンフォーマンスクラスのリソース優先度

ECC1 コンフォーマンスクラスでは、リソース優先度を調節するためにコンフィギュレータがタスクの優先度を自動的にシフトします。**つまりユーザが指定した優先度は変更される可能性があります。**シフトされたタスクの優先度はコンフィギュレータ上で確認することができます。この機能により 1 優先度当たり 1 タスクまたは 1 リソースかつ単一起動の制限を持った ECC1 コンフォーマンスクラスの特長を活かし、高性能を引き出しています。

7.7 リソース使用時の制限

・リソース占有時、イベント待ち、タスク終了を行わないでください。
・リソース獲得による待ちはありません。リソース占有中のためリソースを獲得できなかった場合、再度リソース獲得を要求してください。
・BCC1 コンフォーマンスクラスは RES_SCHEDULER リソースしか使用できません。

7.8 リソースシステムサービス

表 7.1 にリソースを操作するためのシステムサービスを示します。

表 7.1 リソースシステムサービス

システムサービス	機能
GetResource	リソースの獲得
ReleaseResource	リソースの解放

8 イベント管理

8.1 概要

イベントは拡張タスクで使用することができます。イベントを使用することにより拡張タスクを同期させることが可能です。

8.2 イベント操作

複数のイベントを拡張タスクに割り当てることができます。イベントのセットは全てのタスクから行うことが可能です。イベントの待ちおよびクリアは拡張タスクのみ行うことが可能です。複数のイベントを待つことができます。

待ち状態の拡張タスクは、少なくとも一つイベントがセットされた場合、実行可能状態に遷移します。実行中の拡張タスクが、すでにセットされているイベントに対して待ちを試みた場合、タスクは実行状態のままです。

セットされているイベントはタスク起動時にクリアされます。

図 8.1と図 8.2に イベントセットによる拡張タスクの同期を示します。

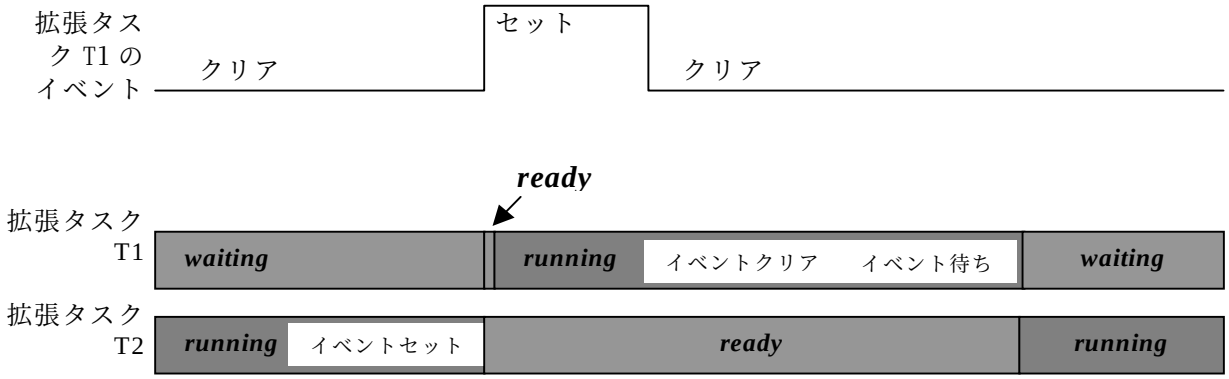

図 8.1 フルプリエンプティブ拡張タスクの同期

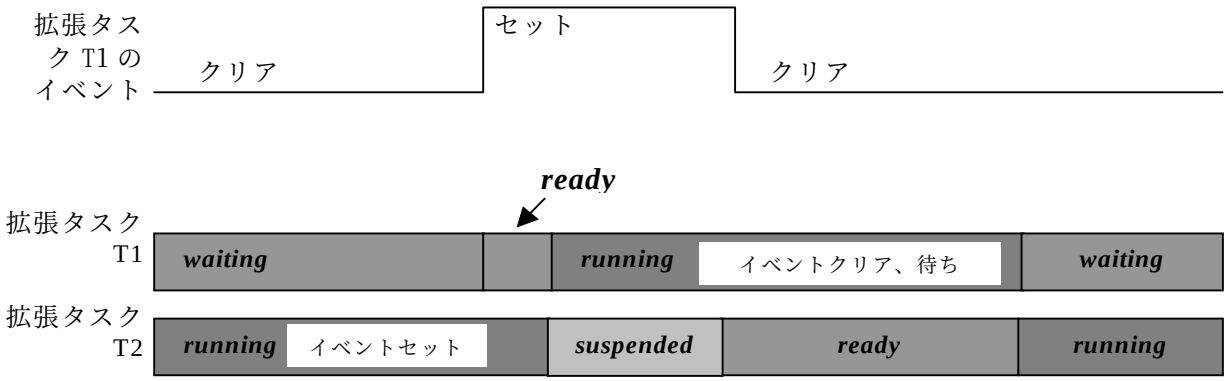

図 8.2 非プリエンプティブ拡張タスクの同期

この例では、拡張タスク T1 が最高優先度です。タスク T1 はイベントを待ちます。タスク T2 は T1 にイベントをセットします。続いて、T1 は待ち状態から実行可能状態へ遷移します。図 8.1のフルプリエンプティブスケジュールの場合、T1 が最高優先度であるため、タスク切り換えが発生し、T2 は T1 によってプリエンプトされます。その後、T1 はこのイベントを再び待ち、T2 は実行を継続します。図 8.2の非プリエンプティブスケジュールの場合、次のスケジューリングポイント(T2 が休止状態に遷移する)までタスク切り換えが発生しません。

8.3 イベント ID

拡張タスクは 8 ビットで表される 8 要因のイベントを持つことができます。各ビットはイベントの状態を表します。0 はイベントがセットされていないかクリアされたことを示します。1 はイベントがセットされたことを示します。図 8.3に各イベントの ID を示します。

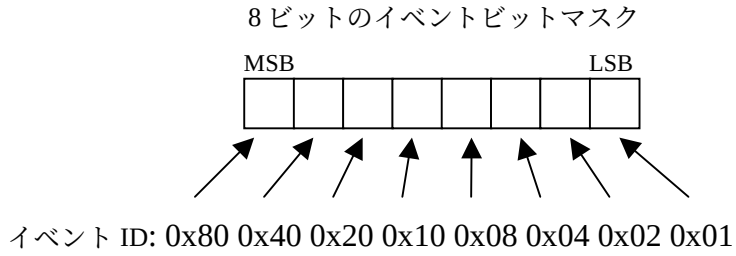

図 8.3 イベント ID

イベント ID はイベント操作時に使用できます。論理和により一度に複数のイベント ID を使用することができます。例えば、ClearEvent(0x01|0x80)はイベント ID 0x01 および 0x80 をクリアします。

イベント ID はユーザが指定するか、あるいはコンフィギュレータによって自動的に決定されます。コンフィギュレータを使用して各タスクで使用するイベントを定義してください。定義されていないイベントをタスクの中で使用しないでください。

8.4 イベントのシステムサービス

表 8.1 にイベントを操作するためのシステムサービスを示します。

表 8.1 イベントシステムサービス

システムサービス	機能
SetEvent	イベントのセット
WaitEvent	イベントの待ち
ClearEvent	イベントのクリア
GetEvent	イベント状態の参照

9 アラーム、カウンタ管理

9.1 概要

OS は、繰り返しイベントを発生させることができる機能を提供します。それは 図 9.1に示されるようにアラームとカウンタという 2 つの概念を与えます。

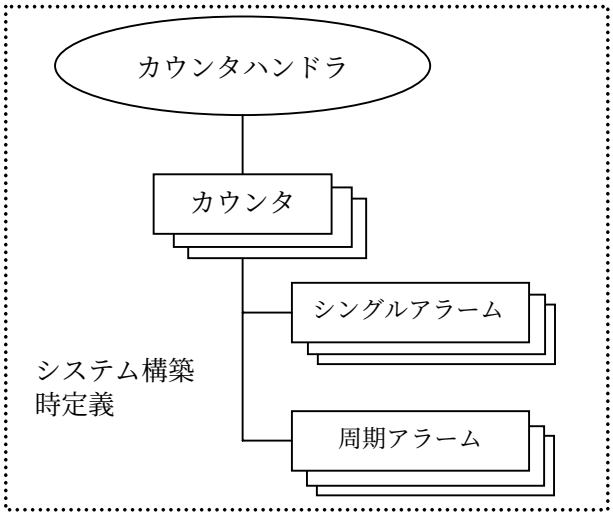

図 9.1 アラーム管理階層

カウンタハンドラは等間隔割り込みタイマのような再発するハンドラです。各カウンタは、”tick”と呼ぶカウント値で示されます。カウンタのハンドラを実行し、IncrementCounter システムサービスによりカウント値を増加させます。OS は、カウンタに基づいたアラームを提供します。アラーム満了時、タスクの起動またはイベントをセットすることができます。

9.2 カウンタ

9.2.1 カウンタハンドラ

カウンタハンドラは動的（可変）アラーム用カウンタをインクリメントするために使用します。IncrementCounter システムサービスによりカウンタをインクリメントしてください。カウンタハンドラの例としてタイマ割り込みサービスルーチンなどがあります。これらの ISR はアプリケーションで用意してください。

例えば、タイマ割り込みサービスルーチンでカウンタをインクリメントするシステムサービスを使用し、カウンタをカウントアップさせます。

9.2.2 システムタイマ

OS は動的（可変）アラームのために、システムタイマカウンタ（カウンタ名：SYSTEM_TIMER）を提供します。 OS は、システムタイマ用に内蔵コンペアマッチタイマチャネル 0 を予約します。システムタイマ用 ISR は、システムタイマをインクリメントします。

システムタイマで使用される ISR 名については11.9 節を参照してください。

9.2.3 非可変アラームタイマ

ユーザは静的（非可変）アラームを定義することができます。これらのアラームは OS が提供する非可変アラームタイマカウンタ（カウンタ名：NonVariantAlarmTimer）を使用します。 OS は、非可変アラームタイマ用に内蔵コンペアマッチタイマチャネル 1 を予約します。非可変アラームタイマ用 ISR は、非可変アラームタイマをインクリメントします。

非可変アラームタイマで使用される ISR 名については11.9 節を参照してください。

9.2.4 カウンタ属性

各カウンタはシステム構築時に設定された次の属性を持っています。

- **maxallowedvalue**
- **ticksperbase**
- **mincycle**

以下にこれらの属性を説明します。

9.2.4.1 maxallowedvalue

maxallowedvalue 属性はカウント値の範囲を決定するために使用されます。

> **留意事項**: **maxallowedvalue** カウント値に到達することはできません。カウンタが、(**maxallowedvalue** – 1)に到達すると、次のインクリメントは 0 から始まります。

カウンタが 8 の **maxallowedvalue** の場合、図 9.2のようにカウント値は 0 から 7 の範囲です。

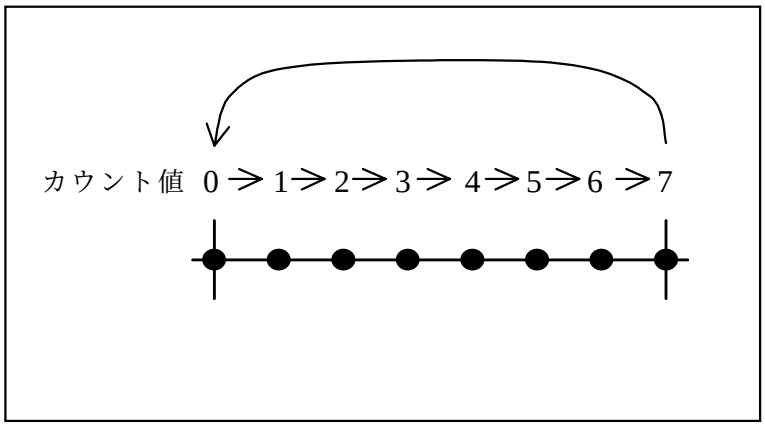

図 9.2 8 の'maxallowedvalue'を持つカウント値の範囲

カウンタがカウント値 7 に到達すると、次のインクリメントによりカウンタは 0 になります。周期アラームの周期カウント値が **maxallowedvalue** にセットされた場合、アラームは常にその同じカウント値で満了します。

カウント値は、符号なし 32 ビットで表されます。したがって、カウンタの周期が 100 µs であれば約 5 日に相当します。

9.2.4.2 ticksperbase

ticksperbase 属性は OS で管理されない（未使用である）ため、ユーザが自由に定義できます。

9.2.4.3 mincycle

mincycle 値は周期アラームの最小周期を設定するときに使用されます。周期アラームの周期が制限範囲内であるかどうか **mincycle** 値と比較されます。

9.3 アラーム

9.3.1 概要

OS は、アラーム満了時、タスク起動あるいはイベントセットが可能な機能を提供します。あらかじめ設定したカウント値に達した場合、アラームは満了となります。図 9.3 にアラームの種類を示します。

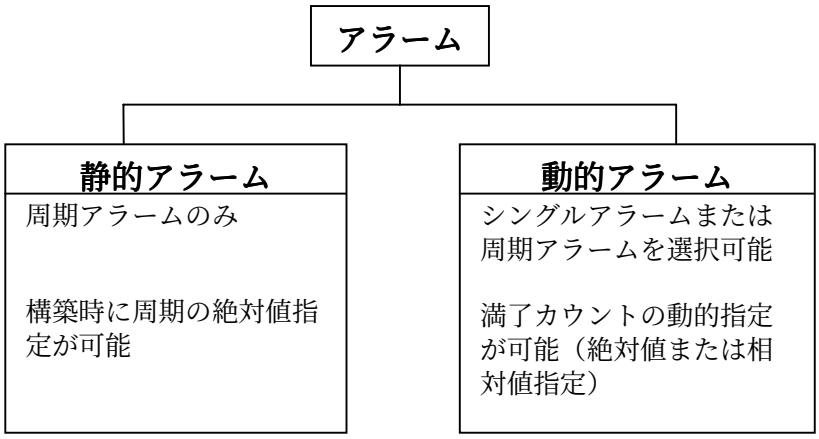

図 9.3 アラーム分類

アラームは静的アラームと動的アラームに分類されます。

動的アラーム: 動的（可変）アラームの設定はアプリケーション実行時に行います。システムサービスを使用することによってアラーム満了や周期のカウント値を変更することができます。動的アラームは任意のカウンタを使用す

ることができます。使用するカウンタはシステム構築時に指定します。動的アラームの有効期間は SetAbsAlarm または SetRelAlarm システムサービスによるアラーム開始から CancelAlarm システムサービスによるアラームキャンセルまでの期間です。
動的アラームは、以下の特長を持っています。

- 絶対値または相対値による満了カウントの指定により動的なアラーム設定が可能
- アラーム開始、キャンセル可能
- シングルアラームまたは周期アラームが可能
- システムタイマカウンタ、ユーザ定義カウンタを使用可能

静的アラーム: 静的（非可変）アラームは、周期的なタスク起動または周期的なイベントセットを行うことが出来ます。周期はシステム構築時に定義し、変更することはできません。静的アラームはキャンセルすることはできません。静的アラームは非可変アラームタイマを使用します。静的アラームは OS 初期化時に自動的に開始されます。静的アラームを停止することはできません。
以下の点に注意してください。

- アラーム満了時間はシステム構築時に静的に設定（変更不可）
- アラーム開始、キャンセル不可
- 周期アラームのみ
- 非可変アラームタイマカウンタのみ使用可能
- オーバヘッド小（動的アラームと比較）

9.3.2 アラームのパラメータ

9.3.2.1 満了カウント

動的アラームの満了カウントは、絶対または相対で指定することができます。このパラメータは、システムサービスを使用することにより実行時に指定することができます。

絶対カウントを指定した場合、そのカウントに達したときにアラームは満了します。

相対カウントを指定した場合、現在のカウントに対する相対カウントに達したときにアラームは満了します。

以下の点に注意してください。

- 絶対アラームの満了カウント値を現在のカウント値と同じカウントに設定した場合、アラームはすぐに満了しません。代わりに、カウンタが 1 回転して満了カウントに達した場合に満了します。

- 相対アラームの相対カウント値を 0 に設定した場合、満了カウントは現在のカウント値に設定されます。カウンタが 1 回転して満了カウントに達した場合に満了します。

- 絶対アラームの満了カウント値を **maxallowedvalue** に設定した場合、満了カウントは 0 に設定されます。

- 相対アラームの満了カウント値を **maxallowedvalue** に設定した場合、満了カウントは現在のカウント値に設定されます。

9.3.2.2 シングルアラームと周期アラーム

システムサービスを使用することにより、実行時にシングルアラームまたは周期アラームを選択することができます。

シングルアラームは設定されたカウントで満了し、自動的に停止します。周期アラームは現在のカウントに対する相対カウントで満了し、リスタートします。シングルアラームと周期アラームをキャンセルすることができるシステムサービスが提供されます。

9.4 カウンタ、アラームの使用例

図 9.4にカウンタとアラームの使用例を示します。

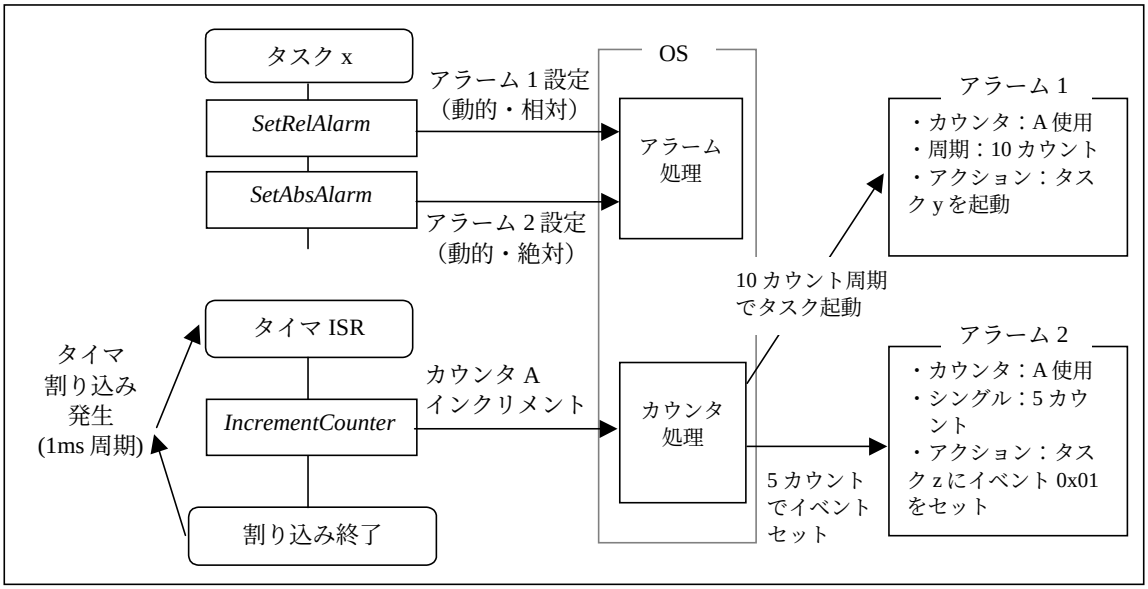

図 9.4 カウンタ、アラーム使用例

アラーム 1 とアラーム 2 はカウンタ A を共用します。タスク y はカウンタ A の 10 カウント（10ms）周期で起動されます。タスク z はカウンタ A の 5 カウント（5ms）経過後イベントセットされます。アラーム 2 はシングルアラームのため、タスク z は一回だけイベントがセットされます。

9.5 カウンタ、アラームのシステムサービス

表 9.1 にカウンタ、アラームを操作するためのシステムサービスを示します。

表 9.1 カウンタ、アラームシステムサービス

システムサービス	機能
InitialiseCounter	カウンタ初期化
IncrementCounter	カウンタ増加
GetAlarmBase	カウンタ属性の取得
GetAlarm	アラーム満了までの残りカウント値
SetRelAlarm	相対アラームの設定
SetAbsAlarm	絶対アラームの設定
CancelAlarm	アラームのキャンセル

10 システム制御

10.1 システム開始

図 10.1 に OS の開始方法を示します。

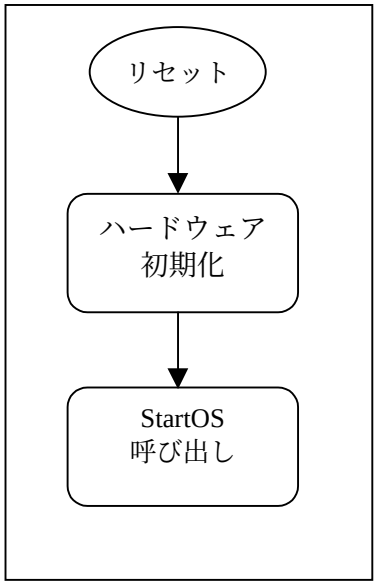

図 10.1 OS 開始

ユーザは必要に応じてハードウェアを初期化する必要があります。

StartOS システムサービスは OS の初期化と開始を行います。引数としてアプリケーションモードを指定してください。引数が不正である場合、システムサービスから復帰し、正常な場合、OS が開始されます。*StartOS* 実行中、カテゴリ 1 を除く全ての割り込みがマスクされます。スケジューラが開始されると全ての割り込みが受け付けられます。自動起動あるいは *StartupHook* フックルーチンによって起動されたタスクにスケジューリングされます。*StartOS* システムサービスを発行するプログラムは 8 バイトのスタック領域を確保してください。

10.2 システムシャットダウン

ShutdownOS システムサービスによりシステムをシャットダウンすることができます。これはアプリケーションから呼び出したり、CPU 例外によって OS が呼び出します。*ShutdownOS* システムサービスは、正常なエラーコードが渡されたかどうかに関わらずシャットダウンします。シャットダウン実行中は、NMI を除く全ての割り込みがマスクされます。シャットダウン処理が終了すると、*StartOS* システムサービスを発行したプログラムに復帰します。復帰時、R15 と SR は回復されますが、その他のレジスタは不定となります。

10.3 フックルーチン

フックルーチンは OS 内部処理内にアプリケーションコードを定義できます。

- システム開始
  OS が開始され、スケジューラが呼び出される前に *StartupHook* フックルーチンが呼び出されます。タスクの自動起動よりも後に呼ばれます。

- システムシャットダウン
  システムシャットダウン時に *ShutdownHook* フックルーチンが呼び出されます。*ShutdownHook* フックルーチンから必ずしも復帰する必要はありません。例えば、StartOS システムサービスを発行し、再び OS を開始することも可能です。

- タスク切り換え
  PreTaskHook, PostTaskHook フックルーチンはタスクコンテキスト切り換え時に呼び出されます。*PreTaskHook* フックルーチンはタスクが実行状態に移る前に呼び出されます。ただし、呼び出された時のタスクの状態はすでに実行状態へ遷移しています。 *PostTaskHook* フックルーチンはタスクが実行状態から実行可能状態、待ち状態または休止状態に遷移するときに呼び出されます。ただし、呼び出された時のタスクの状態は実行状態です。

- エラー処理
  ErrorHook フックルーチンはシステムサービスが正常に呼び出されなかった（E_OK 以外のエラーコードが返る）場合に呼び出されます。

以下の点に注意してください。

1. フックルーチンはオプションです。各フックルーチンを自由に組み込むことが可能です。

2. フックルーチンから使用できるシステムサービスは制限されています。付録 “A.3 ”を参照してください。

3. *ShutdownHook* フックルーチンを除くフックルーチン実行中はカテゴリ 1 割り込みのみ受け付けられます。したがって、割り込みマスク時間を短くするため、フックルーチンの実行時間はできるだけ短くしてください。*ShutdownHook* フックルーチン実行中は NMI 割り込みのみ受け付けられます。

4. フックルーチン実行中、割り込みマスクレベルは下げないでください。**割り込みマスクレベルを下げた場合、システムの動作は保証されません。**

10.4 エラー処理

10.4.1 エラーステータス

二つのエラーモードから選択することが可能です。

標準エラーステータス
– 最小パラメータチェック
– 高速動作
– メモリ消費小

拡張エラーステータス
– より徹底したエラーチェック
– 低速動作
– メモリ消費大

拡張エラーステータスはアプリケーションの開発、デバッグに有効です。標準エラーステータスはデバッグされたシステムに有効です。 システムサービスによって返されるエラーコードは“付録 A.1 ”を参照してください。

10.4.2 シャットダウンエラー

表 10.1 にシャットダウンエラーを示します。

表 10.1 シャットダウンエラー

エラー	**エラーコード**
不当命令	E_OS_SYS_ILLEGAL
未定義例外	E_OS_SYS_EXCEPTION
不当スロット命令	E_OS_SYS_ILLEGAL_SLOT
CPU アドレスエラー	E_OS_SYS_CPU_ADDRESS

シャットダウンエラーが発生した場合、*ShutdownOS* システムサービスが実行されます。ただし、この場合エラーフックは呼ばれません。

10.5 エラーフック再入

ErrorHook フックルーチンから不正なシステムサービスが発行されても ２重に *ErrorHook* フックルーチンが呼び出されることはありません。この場合、*ErrorHook* フックルーチンは呼ばれず処理は続行されます（エラーコードは返ります）。したがって、*ErrorHook* フックルーチンレベルから発行するシステムサービスのエラー処理を行うことはできません。これは独自機能です。

11 プログラミング

以下に OS アプリケーションのプログラミング方法を示します。

11.1 レジスタ

表 11.1に、レジスタの初期値と使用法を示します。

表 11.1 レジスタ初期値と使用法

レジスタ	処理 レベル	内容
PC	タスク	タスクの開始アドレスに設定されます。
	ISR	ISR 開始アドレスに設定されます。
	フック ルーチン	フックルーチン開始アドレスに設定されます。
SR	タスク	0 に設定されます。自由に使用できます。
	ISR	発生した割り込みマスクレベルに設定されます。割り込みマスクレベルは下げないでください。
	フック ルーチン	OS マスクレベルに設定されます。割り込みマスクレベルは下げないでください。
R15	タスク	タスク用スタック領域に設定されます。他のスタック領域に変更しないでください。
	ISR	ISR 用スタック領域に設定されます。他のスタック領域に変更しないでください。
	フック ルーチン	OS 用スタック領域に設定されます。他のスタック領域に変更しないでください。
R0～R7, FR0～FR11, FPUL, FPSCR[4]	タスク	不定。自由に使用できます。
	ISR	不定。自由に使用できます。
	フック ルーチン	不定。自由に使用できます。
R8～R14, MACH, MACL, PR, GBR, FR12～FR15	タスク	不定。自由に使用できます。
	ISR	割り込み発生前の値に設定されます。使用するレジスタはコンパイラが生成するコードによって退避、復帰されます。
	フック ルーチン	不定。使用するレジスタはコンパイラが生成するコードによって退避、復帰されます。

StartOS システムサービスを発行するプログラムは、そのプログラムに #pragma noregalloc を指定し、gbr 組み込み関数および#pragma gbr_base, #pragma gbr_base1 を使用しないで下さい。

表 11.1の内容は、バージョンアップなどにより改変される場合があります。現状と異なる場合は、現状仕様優先で設計してください。

[4] FR0～FR15,FPUL,FPSCR レジスタは浮動小数点演算ユニット（FPU）を搭載しているプロセッサのみ有効です。

11.2 プロセスの宣言

11.2.1 OS 起動

以下の構文にしたがって OS を起動するプログラムを定義してください。

```
#include "sysserv.h"

#pragma noregalloc(関数名)
型 関数名(引数)
{
   :
   StartOS(アプリケーションモード ID);
   :
}
```

11.2.2 タスク

以下の構文にしたがってタスクを定義してください。

```
#include “osekos.h”

TASK (タスク名)
{
   :
   :
   TerminateTask(); または ChainTask(タスク名);
}
```

11.2.3 ISR

以下の構文にしたがって ISR を定義してください。“osekos.h”ファイルをインクルードする必要があります。

```
#include “osekos.h”
:
ISR (ISR 名)
{
   :
   :
}
```

TASK,ISR マクロは“OSEKtype.h” ファイルで定義されています。

11.3 システム構築ファイル

コンフィギュレータによって生成される主なヘッダファイルを以下に示します。

11.3.1 ヘッダファイル

表 11.2 ヘッダファイル

ファイル名	説明
size.h	マシンワードサイズ
defsapp.h	OSEK 状態定義
sysobjid.h	システムオブジェクト ID
OSEKtype.h	型情報
errcodes.h	エラーコード
intdecl.h	割り込み宣言
sysserv.h	システムサービス宣言
Api_id.h	システムサービス ID
osekos.h	上記全てをインクルードしたヘッダファイル

OS を起動するプログラム、タスク、ISR は “osekos.h”をインクルードしてください。

11.4 システムサービス定義

表 11.3に示すシステムサービスを使用する必要はありません。これらのシステムサービスを定義するマクロは、"sysserv.h"ファイルにあります。

表 11.3 システムサービス定義

システムサービス	説明
DeclareTask	タスクオブジェクト宣言
DeclareResource	リソースオブジェクト宣言
DeclareEvent	イベントオブジェクト宣言
DeclareAlarm	アラームオブジェクト宣言
EnterISR	割り込み処理レベルへ移行
LeaveISR	割り込み処理レベルから復帰

11.5 システムオブジェクト参照

システムオブジェクト ID は“sysobjid.h”ファイルに列挙型で定義されます。ISR システムオブジェクト ID はシステムオブジェクト名と同一です。ただし、ISR の ID は ISR 名に”_ID”を付加した名称となります。

以下のようにオブジェクト名称を指定することによって参照できます。

```
#include “osekos.h”
:
:
TASK(Task2)
{
        ActivateTask(Task1);
        DisableInterrupt(Isr1_ID);
}
```

11.6 アセンブラルーチンからのシステムサービスコール

アセンブラルーチンからシステムサービスを呼び出す場合、JSR 命令によって各システムサービスの先頭アドレスに分岐してください。
その際、以下のパラメータルールに従ってください。パラメータの型については、表 A.6 データ型および構築ファイルの C 言語ヘッダファイルを参照してください。

表 11.4 引数

レジスタ	引数
R4	第 1 引数
R5	第 2 引数
R6	第 3 引数
R7	第 4 引数

戻り値は R0 レジスタに格納されます。

システムサービス呼び出し前後では R0～R7,PR,FR0～FR11,FPUL,FPSCR レジスタは保証されません。したがって、これらのレジスタを使用する場合、システムサービス呼び出し前に保存し、呼び出し後に復帰してください。また、StartOS システムサービスを発行するプログラムは R8～R14,MACH, MACL,GBR,FR12～FR15,FPUL, FPSCR を使用しないで下さい。

11.7 アセンブラの ISR

11.7.1 割り込みカテゴリ 1

(1)NMI の ISR

ISR 内で使用するレジスタは、入口で退避し、出口で復帰してください。ISR の終了は RTE 命令を使用してください。

(2)その他の割り込み

・ 割り込みスタックの変更
ISR の先頭でスタックを切り替えてください。対応する割り込みマスクレベルのスタックを使用してください。

・ ISR 復帰
ISR の終了は RTE 命令を使用してください。

・ 使用レジスタ

ISR 内で使用するレジスタは、入口で退避し、出口で復帰してください。

・ ISR 例

図 11.1に割り込みマスクレベル 4 の ISR を示します。なお、C 言語の場合はコンフィギュレータが自動的に生成しますのでアプリケーションで意識する必要はありません。

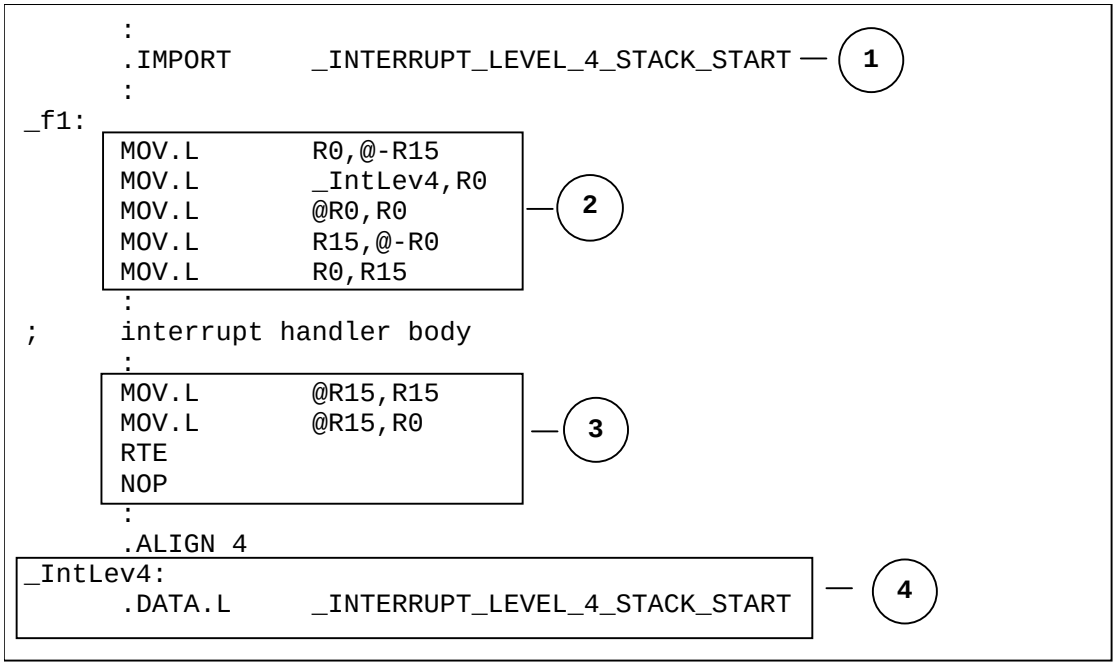

図 11.1 アセンブラの ISR

説明:

1. 割り込みマスクレベル 4 の割り込みスタックの開始を参照します。参照可能なシンボルを以下に示します。

表 11.5 割り込みカテゴリ 1 のスタック開始シンボル

Symbol	Meaning
_INTERRUPT_LEVEL_1_STACK_START	割込みマスクレベル 1 のスタック
_INTERRUPT_LEVEL_2_STACK_START	割込みマスクレベル 2 のスタック
_INTERRUPT_LEVEL_3_STACK_START	割込みマスクレベル 3 のスタック
_INTERRUPT_LEVEL_4_STACK_START	割込みマスクレベル 4 のスタック
_INTERRUPT_LEVEL_5_STACK_START	割込みマスクレベル 5 のスタック
_INTERRUPT_LEVEL_6_STACK_START	割込みマスクレベル 6 のスタック
_INTERRUPT_LEVEL_7_STACK_START	割込みマスクレベル 7 のスタック
_INTERRUPT_LEVEL_8_STACK_START	割込みマスクレベル 8 のスタック
_INTERRUPT_LEVEL_9_STACK_START	割込みマスクレベル 9 のスタック
_INTERRUPT_LEVEL_10_STACK_START	割込みマスクレベル 10 のスタック
_INTERRUPT_LEVEL_11_STACK_START	割込みマスクレベル 11 のスタック
_INTERRUPT_LEVEL_12_STACK_START	割込みマスクレベル 12 のスタック
_INTERRUPT_LEVEL_13_STACK_START	割込みマスクレベル 13 のスタック
_INTERRUPT_LEVEL_14_STACK_START	割込みマスクレベル 14 のスタック
_INTERRUPT_LEVEL_15_STACK_START	割込みマスクレベル 15 のスタック

2. ISR 実行前にスタックを割り込みスタックへ切り替えます。

3. スタックを元に戻し、ISR を終了します。

4. スタックの先頭アドレスを参照します。

11.7.2 割り込みカテゴリ 2

ISR 内で R8～R14,MACH,MACL,PR,GBR,FR12～FR15 レジスタを使用する場合は入口で退避し、出口で復帰してください。ISR の終了は RTS 命令を使用してください。ISR 内でスタックを切り替える必要はありません。

11.8 ISR 登録

ベクタテーブルに ISR の開始アドレスを登録してください。図 11.2に、カテゴリ 1 のベクタテーブルと ISR の関係を示します。

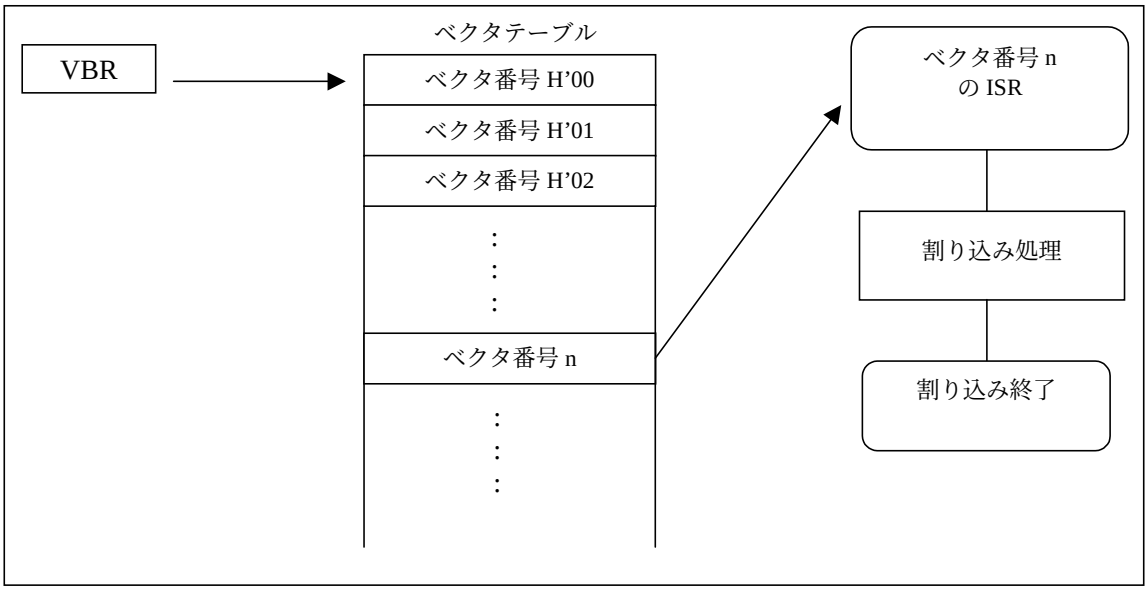

図 11.2 カテゴリ 1 のベクタテーブルと ISR の関係

カテゴリ 2 は割り込み発生時、OS に制御が移ります。ISR の前に割り込み前処理が実行されます。図 11.3 に前処理の動作を示します。

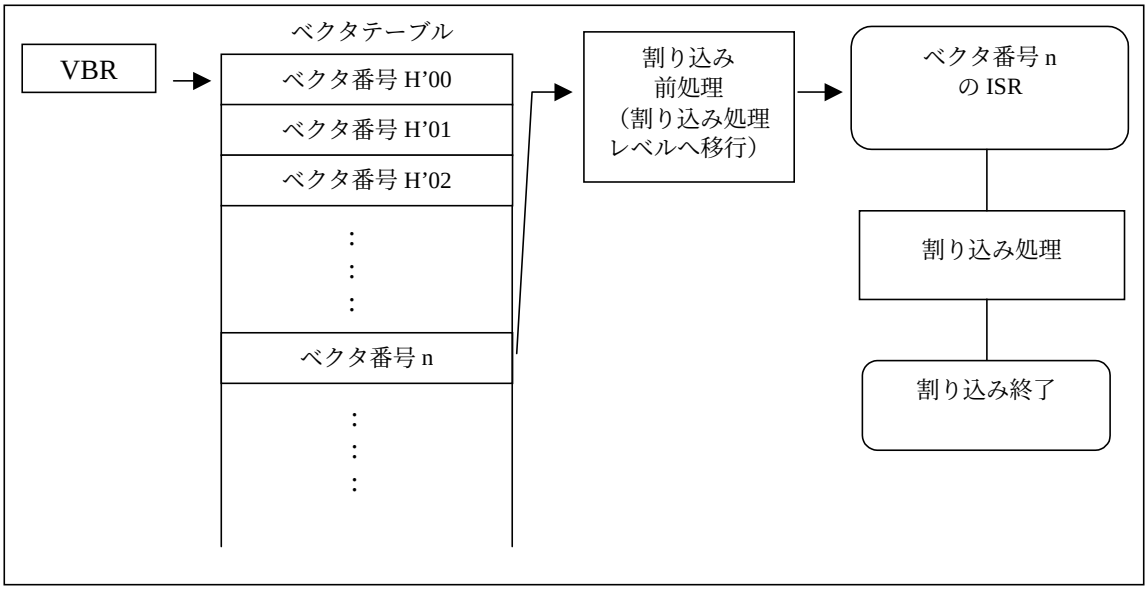

図 11.3 カテゴリ 2 のベクタテーブルと ISR の関係

カテゴリ 2 の各割り込みは、前処理をベクタテーブルに登録してください。これらの前処理はコンフィギュレータで自動的に生成されます。前処理名は、ISR 名に“_ISR”が付加されます。たとえば、“ATUTimer1”は“ATUTimer1_ISR”になります。

また、OS コンフィギュレータで指定した割り込みマスクレベルと同じ値を CPU の割り込み優先レベル設定レジスタ（IPR）に設定してください。その後、IPR を変更しないでください。

11.9 OS 割り込み

OS は、表 11.6の割込みを予約します。

表 11.6 OS 割り込み

割り込み	**ISR (アセンブラ名)**
コンペアマッチタイマ 割り込み 0	_SysTimer_ISR
コンペアマッチタイマ割り込み 1	_NonVarTimer_ISR

12 システムサービス

12.1 概要

各システムサービスの機能詳細を以下の表にしたがって説明します。アセンブラインタフェースについては11.6節を参照してください。

システムサービス名

構文	インタフェース
パラメータ(入力):	入力パラメータ一覧
パラメータ(出力):	出力パラメータ一覧
説明:	システムサービス機能の説明
詳細:	システムサービス機能の詳細説明
エラーステータス: 標準 拡張	リターン値一覧 ● 標準エラーステータスのエラーコード ● 拡張エラーステータスで追加されるエラーコード
コンフォーマンスクラス:	使用可能なコンフォーマンスクラス

システムサービスのデータ型は以下の書き方にしたがいます。

…Type:　　各データの値
…RefType:　…Type へのポインタ

12.2 タスク管理

12.2.1 データ型

- **StatusType**
  リターン値

- **TaskType**
  タスク ID

- **TaskRefType**
  TaskType へのポインタ

- **TaskStateType**
  タスク状態

- **TaskStateRefType**
  TaskStateType へのポインタ

12.2.2 システムサービス

12.2.2.1 ActivateTask

構文	StatusType **ActivateTask**(TaskType <TaskID>)
パラメータ(入力): TaskID	起動タスク ID（または名称）
パラメータ(出力):	なし
説明:	<TaskID>は、休止状態から実行可能状態に遷移します[5]。タスクの先頭アドレスから実行が開始されます。
詳細:	タスク、割り込み、*StartupHook* フックルーチンから呼び出すことができます。
エラーステータス: 標準 拡張	• エラー無し： **E_OK** • 起動超過： **E_OS_LIMIT（独自機能）** • 無効レベルからの呼び出し： **E_OS_CALLEVEL** • <TaskID> が無効： **E_OS_ID**
コンフォーマンスクラス:	BCC1, BCC2, ECC1, ECC2

[5] 多重起動の場合、ActivateTask システムサービスは、タスクの状態をすぐに変更しません。タスクが休止状態でない場合、タスクの起動が記録され、後で実行されます。

12.2.2.2 TerminateTask

構文	StatusType **TerminateTask**(void)
パラメータ(入力):	なし
パラメータ(出力):	なし
説明:	呼び出しタスクを終了します。実行状態から休止状態に遷移します[6]。
詳細:	タスクがリソースを占有している場合、*TerminateTask* システムサービスを呼び出す前にリソースを解放する必要があります。標準エラーステータスでは、リソース占有時にタスクを終了した場合、動作は保証されません。 標準エラーステータスでは、呼び出しもとに復帰しません。拡張エラーステータスでは、エラーが発生した場合、呼び出しもとに復帰します。エラーが発生しなかった場合、呼び出しもとに復帰しません。 タスクのみ呼び出すことができます。 タスクを終了する場合は本システムサービスまたは ChainTask システムサービスを必ず発行してください。
エラーステータス: 標準 拡張	● なし ● タスク以外からの呼び出し： **E_OS_CALLEVEL** ● リソース占有中： **E_OS_RESOURCE**
コンフォーマンスクラス:	BCC1, BCC2, ECC1, ECC2

[6] タスクが多重起動されている場合、休止状態から実行可能状態に遷移します。

12.2.2.3 ChainTask

構文	StatusType **ChainTask**(TaskType <TaskID>)
パラメータ(入力): TaskID	次に起動するタスク ID（または名称）
パラメータ(出力):	なし
説明:	呼び出しタスクを終了します。その後、<TaskID>を連続的に起動します。
詳細:	次に起動するタスクが現在のタスクと同じ場合、多重起動は起こりません。タスク終了後に起動されます。 タスクがリソースを占有している場合、*ChainTask* システムサービスを呼び出す前にリソースを解放する必要があります。標準エラーステータスでは、リソース占有時にタスクを終了した場合、動作は保証されません。 標準エラーステータスでは、呼び出しもとに復帰しません。拡張エラーステータスでは、エラーが発生した場合、呼び出しもとに復帰します。エラーが発生しなかった場合、呼び出しもとに復帰しません。 タスクのみ呼び出すことができます。 タスクを終了する場合は本システムサービスまたは TerminateTask システムサービスを必ず発行してください。
エラーステータス: 標準 拡張	● なし ● 起動超過： **E_OS_LIMIT（独自機能）** ● タスク以外からの呼び出し： **E_OS_CALLEVEL** ● <TaskID>が無効, **E_OS_ID** ● リソース占有中： **E_OS_RESOURCE**
コンフォーマンスクラス:	BCC1, BCC2, ECC1, ECC2

12.2.2.4 Schedule

構文	StatusType **Schedule**(void)
パラメータ(入力):	なし
パラメータ(出力):	なし
説明:	現在のタスクより優先度の高いタスクが実行可能状態であれば、現在のタスクは実行可能状態に遷移し、より高いタスクは実行状態に遷移します。そうでなければ現在のタスクは実行を続けます。
詳細:	非プリエンプティブタスクを強制的に再スケジューリングすることができます。 RES_SCHEDULER リソースが占有されている場合、再スケジューリングは起こりません。 タスクのみ呼び出すことができます。
エラーステータス: 標準 拡張	• 正常終了： **E_OK** • タスク以外からの呼び出し： **E_OS_CALLEVEL**
コンフォーマンスクラス:	BCC1, BCC2, ECC1, ECC2

12.2.2.5 GetTaskID

構文	StatusType **GetTaskID**(TaskRefType <TaskID>)
パラメータ(入力): TaskID	タスク ID 格納領域およびそのアドレス
パラメータ(出力): *TaskID	現在実行中のタスク ID
説明:	現在実行中のタスク ID を取得することができます。
詳細:	タスク、いくつかのフックルーチンから呼び出すことができます。 実行中のタスクがない場合、**ID_INVALID_TASK** が返ります。
エラーステータス: 標準 拡張	• 正常終了： **E_OK** • 無効レベルからの呼び出し： **E_OS_CALLEVEL**
コンフォーマンスクラス:	BCC1, BCC2, ECC1, ECC2

12.2.2.6 GetTaskState

構文	StatusType **GetTaskState**(TaskType <TaskID>, TaskStateRefType <State>)
パラメータ(入力): TaskID State	 参照するタスク ID（または名称） 状態格納領域およびそのアドレス
パラメータ(出力): *State	 <TaskID>の状態
説明:	システムサービスを呼び出したときの<TaskID>の状態を取得することができます。
詳細:	タスク、割り込み、いくつかのフックルーチンから呼び出すことができます。
エラーステータス: 標準 拡張	 • 正常終了：**E_OK** • 無効レベルからの呼び出し：**E_OS_CALLEVEL** • <TaskID>が無効 **E_OS_ID**
コンフォーマンスクラス:	BCC1, BCC2, ECC1, ECC2

12.2.3 TaskStateType データ型の定数

定数	**説明**
SUSPENDED (D'0)	休止状態
WAITING (D'1)	待ち状態
READY (D'2)	実行可能状態
RUNNING (D'3)	実行状態

12.3 割り込み管理

12.3.1 データ型

- **IntDescriptorType**
  割り込みマスクレベルを持ったステータスレジスタ値

12.3.2 システムサービス

12.3.2.1 EnableInterrupt

(a) 割り込みマスクレベル（OS コンフィギュレータの OS ページで Use mask level を選択した場合）

構文	void **EnableInterrupt**(IntDescriptorType <Descriptor>) または void **EnableInterruptMask**(IntDescriptorType <Descriptor>)
パラメータ(入力): Descriptor	有効にする割り込みマスクレベル-1 を持ったステータスレジスタ値
パラメータ(出力):	なし
説明:	<Descriptor>で指定したステータスレジスタの割り込みマスクレベルより高いレベルの割り込みを有効にします。
詳細:	<Descriptor>の値の範囲は 16 の倍数で H'00000000 から H'000000f0 です。それ以外の値を指定した場合、動作は保証されません。H'00000000 を指定したとき OS は再スケジューリングを行います。 タスク、割り込みから呼び出すことができます。フックルーチンから呼び出した場合、エラーは返りません。その場合、動作は保証されません。 **注：本システムサービスの動作は独自機能です。**
エラーステータス: 標準 拡張	なし**（独自機能）** なし**（独自機能）**
コンフォーマンスクラス:	BCC1, BCC2, ECC1, ECC2

(b) 割り込み要因（OS コンフィギュレータの OS ページで Use source を選択した場合）

構文	StatusType **EnableInterrupt**(IntDescriptorType <Descriptor>) または StatusType **EnableInterruptSource**(IntDescriptorType <Descriptor>)
パラメータ(入力): Descriptor	割り込みを有効にする ISR ID（ISR 名の最後に"_ID"を付加した名称）
パラメータ(出力):	なし
説明:	<Descriptor>で指定した ISR ID に対応する割り込み要因を有効にします。
詳細:	CPU の割り込み要求許可ビット、あるいは割り込み優先レベル設定レジスタ(IPR)を有効値に設定します。各割り込み要因に対する設定値は、12.3.4を参照して下さい。ユーザ定義要因（CUSTOM0 ～ CUSTOM31）については12.3.5を参照してください。 タスク、割り込みから呼び出すことができます。フックルーチンから呼び出した場合、エラーは返りません。その場合、動作は保証されません。 本システムサービスでは、エラーが発生してもエラーフックルーチンは呼び出されません。 **注：本システムサービスの動作は独自機能です。Ho7047 は本システムサービスをサポートしていません。**
エラーステータス: 標準 拡張	• 正常終了： **E_OK** • 指定した割り込みが有効： **E_OS_NOFUNC**
コンフォーマンスクラス:	BCC1, BCC2, ECC1, ECC2

12.3.2.2 DisableInterrupt

(a) 割り込みマスクレベル（OS コンフィギュレータの OS ページで Use mask level を選択した場合）

構文	IntDescriptorType **DisableInterrupt**(IntDescriptorType <Descriptor>) または IntDescriptorType **DisableInterruptMask**(IntDescriptorType <Descriptor>)
パラメータ(入力): Descriptor	無効にする割り込みマスクレベルを持ったステータスレジスタ値
パラメータ(出力): R0	DisableInterrupt 発行前の割り込みマスクレベルを持ったステータスレジスタ値
説明:	<Descriptor>で指定したステータスレジスタの割り込みマスクレベル以下の割り込みを無効にします。
詳細:	<Descriptor>の値の範囲は 16 の倍数で H'00000000 から H'000000f0 です。それ以外の値を指定した場合、動作は保証されません。H'00000000 を指定した場合、すべての割り込みが有効になります。指定した割り込みマスクレベルがすでに無効になっている場合（指定した割り込みマスクレベルが無効前の割り込みマスクレベル以下の場合）、指定した値はステータスレジスタには設定されません。 タスク、割り込みから呼び出すことができます。フックルーチンから呼び出した場合、エラーは返りません。その場合、動作は保証されません。 **注：本システムサービスの動作は独自機能です。**
エラーステータス: 標準 拡張	なし**（独自機能）** なし**（独自機能）**
コンフォーマンスクラス:	BCC1, BCC2, ECC1, ECC2

(b) 割り込み要因（OS コンフィギュレータの OS ページで Use source を選択した場合）

構文	StatusType **DisableInterrupt**(IntDescriptorType <Descriptor>) または StatusType **DisableInterruptSource**(IntDescriptorType <Descriptor>)
パラメータ(入力): Descriptor	割り込みを無効にする ISR ID（ISR 名の最後に"_ID"を付加した名称）
パラメータ(出力):	なし
説明:	<Descriptor>で指定した ISR ID に対応する割り込み要因を無効にします。
詳細:	CPU の割り込み要求許可ビット、あるいは割り込み優先レベル設定レジスタ(IPR)を無効値に設定します。各割り込み要因に対する設定値は、12.3.4を参照して下さい。ユーザ定義要因（CUSTOM0 ～ CUSTOM31）については12.3.5を参照してください。 タスク、割り込みから呼び出すことができます。フックルーチンから呼び出した場合、エラーは返りません。その場合、動作は保証されません。 本システムサービスでは、エラーが発生してもエラーフックルーチンは呼び出されません。 **注：本システムサービスの動作は独自機能です。Ho7047 は本システムサービスをサポートしていません。**
エラーステータス: 標準 拡張	• 正常終了： **E_OK** • 指定した割り込みが無効： **E_OS_NOFUNC**
コンフォーマンスクラス:	BCC1, BCC2, ECC1, ECC2

12.3.2.3 GetInterruptDescriptor

(a) 割り込みマスクレベル（OS コンフィギュレータの OS ページで Use mask level を選択した場合）

構文	IntDescriptorType **GetInterruptDescriptor**(void) または IntDescriptorType **GetInterruptDescriptorMask**(void)
パラメータ(入力):	なし
パラメータ(出力): R0	現在の割り込みマスクレベルを持ったステータスレジスタ値
説明:	現在の割込みマスクレベルを取得します。
詳細:	タスク、割り込み、いくつかのフックルーチンから呼び出すことができます。無効なフックルーチンから呼び出した場合、エラーは返りません。その場合、動作は保証されません。 **注：本システムサービスの動作は独自機能です。**
エラーステータス: 標準 拡張	なし**（独自機能）** なし
コンフォーマンスクラス:	BCC1, BCC2, ECC1, ECC2

(b) 割り込み要因（OS コンフィギュレータの OS ページで Use source を選択した場合）

構文	StatusType **GetInterruptDescriptor**(IntDescriptorType <Descriptor>) または StatusType **GetInterruptDescriptorSource**(IntDescriptorType <Descriptor>)
パラメータ(入力): Descriptor	割り込み状態を取得する ISR ID（ISR 名の最後に”_ID”を付加した名称）
パラメータ(出力):	なし
説明:	<Descriptor>で指定した ISR ID に対応する割り込み要因の状態を取得します。
詳細:	CPU の割り込み要求許可ビット、あるいは割り込み優先レベル設定レジスタ(IPR)の値を参照します。各割り込み要因に対する有効値、無効値は、12.3.4を参照して下さい。IPR を参照する割り込み要因の場合、OS コンフィギュレータで指定した割り込みレベルのみ有効値となります。ユーザ定義要因（CUSTOM0 ～ CUSTOM31）については12.3.5を参照してください。 タスク、割り込み、いくつかのフックルーチンから呼び出すことができます。無効なフックルーチンから呼び出した場合、エラーは返りません。その場合、動作は保証されません。 本システムサービスでは、エラーが発生してもエラーフックルーチンは呼び出されません。 **注：本システムサービスの動作は独自機能です。Ho7047 は本システムサービスをサポートしていません。**
エラーステータス: 標準 拡張	● 割り込みが有効：**1** ● 割り込みが無効：**0** なし
コンフォーマンスクラス:	BCC1, BCC2, ECC1, ECC2

12.3.3 IntDescriptorType データ型の定数

割り込みマスクレベル方式（OS コンフィギュレータの OS ページで Use mask level）を選択した場合に使用します。

表 12.1 IntDescriptorType データ

定数	説明
SR_IMS00 (H’00000000)	割り込みマスクレベル 0
SR_IMS01 (H’00000010)	割り込みマスクレベル 1
SR_IMS02 (H’00000020)	割り込みマスクレベル 2
SR_IMS03 (H’00000030)	割り込みマスクレベル 3
SR_IMS04 (H’00000040)	割り込みマスクレベル 4
SR_IMS05 (H’00000050)	割り込みマスクレベル 5
SR_IMS06 (H’00000060)	割り込みマスクレベル 6
SR_IMS07 (H’00000070)	割り込みマスクレベル 7
SR_IMS08 (H’00000080)	割り込みマスクレベル 8
SR_IMS09 (H’00000090)	割り込みマスクレベル 9
SR_IMS10 (H’000000a0)	割り込みマスクレベル 10
SR_IMS11 (H’000000b0)	割り込みマスクレベル 11
SR_IMS12 (H’000000c0)	割り込みマスクレベル 12
SR_IMS13 (H’000000d0)	割り込みマスクレベル 13
SR_IMS14 (H’000000e0)	割り込みマスクレベル 14
SR_IMS15 (H’000000f0)	割り込みマスクレベル 15

12.3.4 割り込み要因と設定値

割り込み要因方式（OS コンフィギュレータの OS ページで Use source）を選択した場合に使用します。Ho7047 は割り込み要因方式をサポートしていません。

表 12.2 割り込み要因と設定値

No.	要因名	ベクタ番号	レジスタ	ビット	無効値	有効値
1	IRQ0	64	IPRA	ビット 15～12	0	1～15 *1
2	IRQ1	65	IPRA	ビット 11～8	0	1～15 *1
3	IRQ2	66	IPRA	ビット 7～4	0	1～15 *1
4	IRQ3	67	IPRA	ビット 3～0	0	1～15 *1
5	IRQ4	68	IPRB	ビット 15～12	0	1～15 *1
6	IRQ5	69	IPRB	ビット 11～8	0	1～15 *1
7	IRQ6	70	IPRB	ビット 7～4	0	1～15 *1
8	IRQ7	71	IPRB	ビット 3～0	0	1～15 *1
9	DMAC0 DEI0	72	CHCR0	IE（ビット 2）	0	1
10	DMAC1 DEI1	74	CHCR1	IE（ビット 2）	0	1
11	DMAC2 DEI2	76	CHCR2	IE（ビット 2）	0	1
12	DMAC3 DEI3	78	CHCR3	IE（ビット 2）	0	1
13	ATU0 ITV1(ITVE6)	80	ITVRR1	ITVE6（ビット 0）	0	1
14	ATU0 ITV1(ITVE7)	80	ITVRR1	ITVE7（ビット 1）	0	1
15	ATU0 ITV1(ITVE8)	80	ITVRR1	ITVE8（ビット 2）	0	1
16	ATU0 ITV1(ITVE9)	80	ITVRR1	ITVE9（ビット 3）	0	1
17	ATU0 ITV2A(ITVE10A)	80	ITVRR2A	ITVE10A（ビット 0）	0	1
18	ATU0 ITV2A(ITVE11A)	80	ITVRR2A	ITVE11A（ビット 1）	0	1
19	ATU0 ITV2A(ITVE12A)	80	ITVRR2A	ITVE12A（ビット 2）	0	1
20	ATU0 ITV2A(ITVE13A)	80	ITVRR2A	ITVE13A（ビット 3）	0	1
21	ATU0 ITV2B(ITVE10B)	80	ITVRR2B	ITVE10B（ビット 0）	0	1
22	ATU0 ITV2B(ITVE11B)	80	ITVRR2B	ITVE11B（ビット 1）	0	1
23	ATU0 ITV2B(ITVE12B)	80	ITVRR2B	ITVE12B（ビット 2）	0	1
24	ATU0 ITV2B(ITVE13B)	80	ITVRR2B	ITVE13B（ビット 3）	0	1
25	ATU0 ICI0A	84	TIER0	ICE0A（ビット 0）	0	1
26	ATU0 ICI0B	86	TIER0	ICE0B（ビット 1）	0	1
27	ATU0 ICI0C	88	TIER0	ICE0C（ビット 2）	0	1
28	ATU0 ICI0D	90	TIER0	ICE0D（ビット 3）	0	1
29	ATU0 OVI0	92	TIER0	OVE0（ビット 4）	0	1
30	ATU1 IMI1A	96	TIER1A	IME1A（ビット 0）	0	1
31	ATU1 CMI1	96	TIER1B	CME1（ビット 0）	0	1
32	ATU1 IMI1B	97	TIER1A	IME1B（ビット 1）	0	1
33	ATU1 IMI1C	98	TIER1A	IME1C（ビット 2）	0	1
34	ATU1 IMI1D	99	TIER1A	IME1D（ビット 3）	0	1
35	ATU1 IMI1E	100	TIER1A	IME1E（ビット 4）	0	1
36	ATU1 IMI1F	101	TIER1A	IME1F（ビット 5）	0	1
37	ATU1 IMI1G	102	TIER1A	IME1G（ビット 6）	0	1
38	ATU1 IMI1H	103	TIER1A	IME1H（ビット 7）	0	1

表 12.2 割り込み要因と設定値（続き）

No.	要因名	ベクタ番号	レジスタ	ビット	無効値	有効値
39	ATU1 OVI1A	104	TIER1A	OVE1A（ビット 8）	0	1
40	ATU1 OVI1B	104	TIER1B	OVE1B（ビット 8）	0	1
41	ATU2 IMI2A	108	TIER2A	IME2A（ビット 0）	0	1
42	ATU2 CMI2A	108	TIER2B	CME2A（ビット 0）	0	1
43	ATU2 IMI2B	109	TIER2A	IME2B（ビット 1）	0	1
44	ATU2 CMI2B	109	TIER2B	CME2B（ビット 1）	0	1
45	ATU2 IMI2C	110	TIER2A	IME2C（ビット 2）	0	1
46	ATU2 CMI2C	110	TIER2B	CME2C（ビット 2）	0	1
47	ATU2 IMI2D	111	TIER2A	IME2D（ビット 3）	0	1
48	ATU2 CMI2D	111	TIER2B	CME2D（ビット 3）	0	1
49	ATU2 IMI2E	112	TIER2A	IME2E（ビット 4）	0	1
50	ATU2 CMI2E	112	TIER2B	CME2E（ビット 4）	0	1
51	ATU2 IMI2F	113	TIER2A	IME2F（ビット 5）	0	1
52	ATU2 CMI2F	113	TIER2B	CME2F（ビット 5）	0	1
53	ATU2 IMI2G	114	TIER2A	IME2G（ビット 6）	0	1
54	ATU2 CMI2G	114	TIER2B	CME2G（ビット 6）	0	1
55	ATU2 IMI2H	115	TIER2A	IME2H（ビット 7）	0	1
56	ATU2 CMI2H	115	TIER2B	CME2H（ビット 7）	0	1
57	ATU2 OVI2A	116	TIER2A	OVE2A（ビット 8）	0	1
58	ATU2 OVI2B	116	TIER2B	OVE2B（ビット 8）	0	1
59	ATU3 IMI3A	120	TIER3	IME3A（ビット 0）	0	1
60	ATU3 IMI3B	121	TIER3	IME3B（ビット 1）	0	1
61	ATU3 IMI3C	122	TIER3	IME3C（ビット 2）	0	1
62	ATU3 IMI3D	123	TIER3	IME3D（ビット 3）	0	1
63	ATU3 OVI3	124	TIER3	OVE3（ビット 4）	0	1
64	ATU4 IMI4A	128	TIER3	IME4A（ビット 5）	0	1
65	ATU4 IMI4B	129	TIER3	IME4B（ビット 6）	0	1
66	ATU4 IMI4C	130	TIER3	IME4C（ビット 7）	0	1
67	ATU4 IMI4D	131	TIER3	IME4D（ビット 8）	0	1
68	ATU4 OVI4	132	TIER3	OVE4（ビット 9）	0	1
69	ATU5 IMI5A	136	TIER3	IME5A（ビット 10）	0	1
70	ATU5 IMI5B	137	TIER3	IME5B（ビット 11）	0	1
71	ATU5 IMI5C	138	TIER3	IME5C（ビット 12）	0	1
72	ATU5 IMI5D	139	TIER3	IME5D（ビット 13）	0	1
73	ATU5 OVI5	140	TIER3	OVE5（ビット 14）	0	1
74	ATU6 CMI6A	144	TIER6	CME6A（ビット 0）	0	1
75	ATU6 CMI6B	145	TIER6	CME6B（ビット 1）	0	1
76	ATU6 CMI6C	146	TIER6	CME6C（ビット 2）	0	1
77	ATU6 CMI6D	147	TIER6	CME6D（ビット 3）	0	1
78	ATU7 CMI7A	148	TIER7	CME7A（ビット 0）	0	1
79	ATU7 CMI7B	149	TIER7	CME7B（ビット 1）	0	1
80	ATU7 CMI7C	150	TIER7	CME7C（ビット 2）	0	1
81	ATU7 CMI7D	151	TIER7	CME7D（ビット 3）	0	1
82	ATU8 OSI8A	152	TIER8	OSE8A（ビット 0）	0	1
83	ATU8 OSI8B	153	TIER8	OSE8B（ビット 1）	0	1
84	ATU8 OSI8C	154	TIER8	OSE8C（ビット 2）	0	1

表 12.2 割り込み要因と設定値（続き）

No.	要因名	ベクタ番号	レジスタ	ビット	無効値	有効値
85	ATU8 OSI8D	155	TIER8	OSE8D（ビット 3）	0	1
86	ATU8 OSI8E	156	TIER8	OSE8E（ビット 4）	0	1
87	ATU8 OSI8F	157	TIER8	OSE8F（ビット 5）	0	1
88	ATU8 OSI8G	158	TIER8	OSE8G（ビット 6）	0	1
89	ATU8 OSI8H	159	TIER8	OSE8H（ビット 7）	0	1
90	ATU8 OSI8I	160	TIER8	OSE8I（ビット 8）	0	1
91	ATU8 OSI8J	161	TIER8	OSE8J（ビット 9）	0	1
92	ATU8 OSI8K	162	TIER8	OSE8K（ビット 10）	0	1
93	ATU8 OSI8L	163	TIER8	OSE8L（ビット 11）	0	1
94	ATU8 OSI8M	164	TIER8	OSE8M（ビット 12）	0	1
95	ATU8 OSI8N	165	TIER8	OSE8N（ビット 13）	0	1
96	ATU8 OSI8O	166	TIER8	OSE8O（ビット 14）	0	1
97	ATU8 OSI8P	167	TIER8	OSE8P（ビット 15）	0	1
98	ATU9 CMI9A	168	TIER9	CME9A（ビット 0）	0	1
99	ATU9 CMI9B	169	TIER9	CME9B（ビット 1）	0	1
100	ATU9 CMI9C	170	TIER9	CME9C（ビット 2）	0	1
101	ATU9 CMI9D	171	TIER9	CME9D（ビット 3）	0	1
102	ATU9 CMI9E	172	TIER9	CME9E（ビット 4）	0	1
103	ATU9 CMI9F	174	TIER9	CME9F（ビット 5）	0	1
104	ATU10 CMI10A	176	TIER10	CME10A（ビット 0）	0	1
105	ATU10 CMI10B	178	TIER10	CME10B（ビット 2）	0	1
106	ATU10 ICI10A	180	TIER10	ICE10A（ビット 1）	0	1
107	ATU10 CMI10G	180	TIER10	CME10G（ビット 3）	0	1
108	ATU11 IMI11A	184	TIER11	IME11A（ビット 0）	0	1
109	ATU11 IMI11B	186	TIER11	IME11B（ビット 1）	0	1
110	ATU11 OVI11	187	TIER11	OVE11（ビット 8）	0	1
111	CMT0 CMTI0	188	CMCSR0	CMIE（ビット 6）	0	1
112	A/D0 ADI0	190	ADCSR0	ADIE（ビット 6）	0	1
113	CMT1 CMTI1	192	CMCSR1	CMIE（ビット 6）	0	1
114	A/D1 ADI1	194	ADCSR1	ADIE（ビット 6）	0	1
115	A/D2 ADI2	196	ADCSR2	ADIE（ビット 6）	0	1
116	SCI0 ERI0	200	SCR0	RIE（ビット 6）	0	1
117	SCI0 RXI0	201	SCR0	RIE（ビット 6）	0	1
118	SCI0 TXI0	202	SCR0	TIE（ビット 7）	0	1
119	SCI0 TEI0	203	SCR0	TEIE（ビット 2）	0	1
120	SCI1 ERI1	204	SCR1	RIE（ビット 6）	0	1
121	SCI1 RXI1	205	SCR1	RIE（ビット 6）	0	1
122	SCI1 TXI1	206	SCR1	TIE（ビット 7）	0	1
123	SCI1 TEI1	207	SCR1	TEIE（ビット 2）	0	1
124	SCI2 ERI2	208	SCR2	RIE（ビット 6）	0	1
125	SCI2 RXI2	209	SCR2	RIE（ビット 6）	0	1
126	SCI2 TXI2	210	SCR2	TIE（ビット 7）	0	1
127	SCI2 TEI2	211	SCR2	TEIE（ビット 2）	0	1
128	SCI3 ERI3	212	SCR3	RIE（ビット 6）	0	1
129	SCI3 RXI3	213	SCR3	RIE（ビット 6）	0	1
130	SCI3 TXI3	214	SCR3	TIE（ビット 7）	0	1

表 12.2 割り込み要因と設定値（続き）

No.	要因名	ベクタ番号	レジスタ	ビット	無効値	有効値
131	SCI3 TEI3	215	SCR3	TEIE（ビット 2）	0	1
132	SCI4 ERI4	216	SCR4	RIE（ビット 6）	0	1
133	SCI4 RXI4	217	SCR4	RIE（ビット 6）	0	1
134	SCI4 TXI4	218	SCR4	TIE（ビット 7）	0	1
135	SCI4 TEI4	219	SCR4	TEIE（ビット 2）	0	1
136	HCAN0 ERS0(IRR5)	220	IMR0	IMR5（ビット 5）	1	0
137	HCAN0 ERS0(IRR6)	220	IMR0	IMR6（ビット 6）	1	0
138	HCAN0 ERS0(IRR3)	220	IMR0	IMR3（ビット 3）	1	0
139	HCAN0 ERS0(IRR4)	220	IMR0	IMR4（ビット 4）	1	0
140	HCAN0 OVR0(IRR0)	221	IMR0	IMR0（ビット 0）	1	0
141	HCAN0 OVR0(IRR7)	221	IMR0	IMR7（ビット 7）	1	0
142	HCAN0 OVR0(IRR9)	221	IMR0	IMR9（ビット 9）	1	0
143	HCAN0 OVR0(IRR10)	221	IMR0	IMR10（ビット 10）	1	0
144	HCAN0 OVR0(IRR11)	221	IMR0	IMR11（ビット 11）	1	0
145	HCAN0 OVR0(IRR12)	221	IMR0	IMR12（ビット 12）	1	0
146	HCAN0 OVR0(IRR13)	221	IMR0	IMR13（ビット 13）	1	0
147	HCAN0 OVR0(IRR14)	221	IMR0	IMR14（ビット 14）	1	0
148	HCAN0 OVR0(IRR15)	221	IMR0	IMR15（ビット 15）	1	0
149	HCAN0 RM0(IRR1)	222	IMR0	IMR1（ビット 1）	1	0
150	HCAN0 RM0(IRR2)	222	IMR0	IMR2（ビット 2）	1	0
151	HCAN0 SLE0(IRR8)	223	IMR0	IMR8（ビット 8）	1	0
152	WDT ITI	224	TCSR	TME（ビット 5）	0	1
153	HCAN1 ERS1(IRR5)	228	IMR1	IMR5（ビット 5）	1	0
154	HCAN1 ERS1(IRR6)	228	IMR1	IMR6（ビット 6）	1	0
155	HCAN1 ERS1(IRR3)	228	IMR1	IMR3（ビット 3）	1	0
156	HCAN1 ERS1(IRR4)	228	IMR1	IMR4（ビット 4）	1	0
157	HCAN1 OVR1(IRR0)	229	IMR1	IMR0（ビット 0）	1	0
158	HCAN1 OVR1(IRR7)	229	IMR1	IMR7（ビット 7）	1	0
159	HCAN1 OVR1(IRR9)	229	IMR1	IMR9（ビット 9）	1	0
160	HCAN1 OVR1(IRR10)	229	IMR1	IMR10（ビット 10）	1	0
161	HCAN1 OVR1(IRR11)	229	IMR1	IMR11（ビット 11）	1	0
162	HCAN1 OVR1(IRR12)	229	IMR1	IMR12（ビット 12）	1	0
163	HCAN1 OVR1(IRR13)	229	IMR1	IMR13（ビット 13）	1	0
164	HCAN1 OVR1(IRR14)	229	IMR1	IMR14（ビット 14）	1	0
165	HCAN1 OVR1(IRR15)	229	IMR1	IMR15（ビット 15）	1	0
166	HCAN1 RM1(IRR1)	230	IMR1	IMR1（ビット 1）	1	0
167	HCAN1 RM1(IRR2)	230	IMR1	IMR2（ビット 2）	1	0
168	HCAN1 SLE1(IRR8)	231	IMR1	IMR8（ビット 8）	1	0

【注】*1 OS コンフィギュレータの Isr ページで指定した割り込みレベル

12.3.5 ユーザ定義要因関数

ユーザ定義要因（CUSTOM0 ～ CUSTOM31）は、割り込みの有効、無効、状態取得を行う処理が標準提供されていません。したがって、ユーザ定義要因を使用する場合、そのユーザ定義要因の EnableInterruptSource（または EnableInterrupt）システムサービスに対する割り込みを有効にする処理を、Example\OSEKisc.c ファイル内の_OSEKInterrupt_CUSTOMxx 関数割り込みに追加して下さい。また、DisableInterruptSource（または DisableInterrupt）システムサービスに対する割り込み無効処理、GetInterruptDescriptorSource（または GetInterruptDescriptor）システムサービスに対する割り込み状態取得処理を追加してください。
ユーザ定義要因関数の入力パラメータは OS が渡しますが、出力パラメータはユーザ側で設定する必要があります。EnableInterruptSource（または EnableInterrupt）、DisableInterruptSource（または DisableInterrupt）システムサービスの場合はエラーコード（E_OK または E_OS_NOFUNC）を、GetInterruptDescriptorSource（または GetInterruptDescriptor）システムサービスの場合は割り込み状態値（1 または 0）を設定してください。設定された出力パラメータは EnableInterruptSource（または EnableInterrupt）、DisableInterruptSource（または DisableInterrupt）、GetInterruptDescriptorSource（または GetInterruptDescriptor）システムサービスそれぞれのリターンコードとしてシステムサービス呼び出しプログラムへ返ります。

表 12.3 ユーザ定義要因関数

要因名	関数名
CUSTOM0	_OSEKInterrupt_CUSTOM0
CUSTOM1	_OSEKInterrupt_CUSTOM1
CUSTOM2	_OSEKInterrupt_CUSTOM2
CUSTOM3	_OSEKInterrupt_CUSTOM3
CUSTOM4	_OSEKInterrupt_CUSTOM4
CUSTOM5	_OSEKInterrupt_CUSTOM5
CUSTOM6	_OSEKInterrupt_CUSTOM6
CUSTOM7	_OSEKInterrupt_CUSTOM7
CUSTOM8	_OSEKInterrupt_CUSTOM8
CUSTOM9	_OSEKInterrupt_CUSTOM9
CUSTOM10	_OSEKInterrupt_CUSTOM10
CUSTOM11	_OSEKInterrupt_CUSTOM11
CUSTOM12	_OSEKInterrupt_CUSTOM12
CUSTOM13	_OSEKInterrupt_CUSTOM13
CUSTOM14	_OSEKInterrupt_CUSTOM14
CUSTOM15	_OSEKInterrupt_CUSTOM15
CUSTOM16	_OSEKInterrupt_CUSTOM16
CUSTOM17	_OSEKInterrupt_CUSTOM17
CUSTOM18	_OSEKInterrupt_CUSTOM18
CUSTOM19	_OSEKInterrupt_CUSTOM19
CUSTOM20	_OSEKInterrupt_CUSTOM20
CUSTOM21	_OSEKInterrupt_CUSTOM21
CUSTOM22	_OSEKInterrupt_CUSTOM22
CUSTOM23	_OSEKInterrupt_CUSTOM23
CUSTOM24	_OSEKInterrupt_CUSTOM24
CUSTOM25	_OSEKInterrupt_CUSTOM25
CUSTOM26	_OSEKInterrupt_CUSTOM26
CUSTOM27	_OSEKInterrupt_CUSTOM27
CUSTOM28	_OSEKInterrupt_CUSTOM28
CUSTOM29	_OSEKInterrupt_CUSTOM29
CUSTOM30	_OSEKInterrupt_CUSTOM30
CUSTOM31	_OSEKInterrupt_CUSTOM31

以下にユーザ定義要因関数の仕様を示します。

構文	StatusType **_OSEKInterrupt_CUSTOM*xx*(** IntDescriptorType <Descriptor>, DWORD <Service>)
パラメータ(入力*): Descriptor Service	割り込みサービスに指定した ISR ID システムサービス ID（EnableInterruptSource_ID, DisableInterruptSource_ID または GetInterruptDescriptorSource_ID）
パラメータ(出力):	なし
説明:	ユーザ定義要因に対する割り込みの有効、無効、状態取得処理を行います。
詳細:	割り込み要因方式の割り込みシステムサービスでユーザ定義要因の ISR ID を指定した場合に呼び出されます。 本ルーチンではシステムサービスを発行しないでください。 本ルーチン実行中はカテゴリ 1 割り込みのみ受け付けられます。したがって、割り込みマスク時間を短くするため、本ルーチンの実行時間はできるだけ短くしてください。本ルーチン実行中、割り込みマスクレベルは下げないでください。 **ユーザ定義要因に対応する ISR ID を指定した割り込みシステムサービスを呼び出す場合、呼び出しプログラム（タスク、ISR、フックルーチン）のスタックサイズに、本ルーチンで使用するスタックサイズを加算して下さい。** **Ho7047 は本関数をサポートしていません。**
エラーステータス: 標準 拡張	● 正常終了：**E_OK**（EnableInterruptSource_ID, DisableInterruptSource_ID の場合） ● 割り込みが有効：**1** ● 割り込みが無効：**0** （GetInterruptDescriptorSource_ID の場合） ● 指定した割り込みが有効または無効： **E_OS_NOFUNC** （EnableInterruptSource_ID, DisableInterruptSource_ID の場合）
コンフォーマンスクラス:	BCC1, BCC2, ECC1, ECC2

【注】* 入力パラメータは OS が設定します。

12.4 リソース管理

12.4.1 データ型

- **ResourceType**
  リソース ID

12.4.2 システムサービス

12.4.2.1 GetResource

構文	StatusType **GetResource**(ResourceType <ResID>)
パラメータ(入力): ResID	リソース ID（または名称）
パラメータ(出力):	なし
説明:	<ResID>を獲得し、タスクの優先度を上げます。
詳細:	リソース占有中、休止または待ち状態になるシステムサービス（*TerminateTask, ChainTask, WaitEvent*）は使用できません。 タスクのみ呼び出すことができます。 リソース占有のネストは LIFO 規則に従ってください。 **注：ネストされるリソースが外側のリソースの優先度より低い場合、エラーが返ります（RES_SCHDULER を除く）。この動作は独自機能です。**ただし、この場合もシステムサービスは実行されます。
エラーステータス: 標準 拡張	• 正常終了：**E_OK** • タスクレベル以外からの呼び出し：**E_OS_CALLEVEL** • <ResID>が無効：**E_OS_ID** • <ResID>占有中、<ResID>が使用定義外：**E_OS_ACCESS** • リソースネスト不正：**E_OS_INVALID_NEST（独自機能）**
コンフォーマンスクラス:	BCC1, BCC2, ECC1, ECC2

12.4.2.2 ReleaseResource

構文	StatusType **ReleaseResource**(ResourceType <ResID>)
パラメータ(入力): ResID	リソース ID（または名称）
パラメータ(出力):	なし
説明:	<ResID>を解放し、もとの優先度に戻します。
詳細:	リソースのネストに関しては、*GetResource* の詳細を参照してください。 タスクのみ呼び出すことができます。
エラーステータス: 標準 拡張	● エラー無し： **E_OK** ● タスクレベル以外からの呼び出し： **E_OS_CALLEVEL** ● <ResID>が無効： **E_OS_ID** ● <ResID>が獲得されてない、またはリソース解放順序不正（他のリソースを先に解放しなければならない）：**E_OS_NOFUNC**
コンフォーマンスクラス:	BCC1, BCC2, ECC1, ECC2

12.5 イベント管理

12.5.1 データ型

- **EventMaskType**
  イベントマスク値

- **EventMaskRefType**
  EventMaskType へのポインタ

12.5.2 システムサービス

12.5.2.1 SetEvent

構文	StatusType **SetEvent**(TaskType <TaskID>, EventMaskType <Mask>)
パラメータ(入力): TaskID Mask	 イベントをセットするタスク ID（または名称） イベントマスク値（または名称）
パラメータ(出力):	なし
説明:	<TaskID>にイベントをセットします。<Mask>で指定されたイベントを１つ以上待っていた場合、<TaskID>は実行可能状態に遷移します。 タスクは１つ以上のイベントをセットできます。 例えば、タスクがイベント ID の 0x01 および 0x80 をセットしたい場合、*SetEvent*(<TaskID>, 0x01 | 0x80)を実行します。
詳細:	セットされないイベントは変化（クリア）しません。 <TaskID>は拡張タスクでなければなりません。 タスク、割り込みから呼び出すことができます。
エラーステータス: 標準 拡張	 • 正常終了： **E_OK** • 無効レベルからの呼び出し：**E_OS_CALLEVEL** • <TaskID>が無効： **E_OS_ID** • <TaskID>が拡張タスクでない： **E_OS_ACCESS** • <TaskID>が休止状態： **E_OS_STATE**
コンフォーマンスクラス:	ECC1, ECC2

12.5.2.2 ClearEvent

構文	StatusType **ClearEvent**(EventMaskType <Mask>)
パラメータ(入力): Mask	クリアするイベントマスク値（または名称）
パラメータ(出力):	なし
説明:	呼び出しタスクに対して、<Mask>で指定されたイベントをクリアします。 タスクは１つ以上のイベントをクリアできます。 例えば、タスクがイベント ID の 0x01 および 0x80 をクリアしたい場合、*ClearEvent*(0x01 | 0x80) を実行します。
詳細:	本システムサービスは拡張タスクのみ呼び出すことができます。
エラーステータス: 標準 拡張	● 正常終了： **E_OK** ● タスクレベル以外からの呼び出し： **E_OS_CALLEVEL** ● .基本タスクからの呼び出し： **E_OS_ACCESS**
コンフォーマンスクラス:	ECC1, ECC2

12.5.2.3 GetEvent

構文	StatusType **GetEvent**(TaskType <TaskID>, EventMaskRefType <Mask>)
パラメータ(入力): TaskID Mask	 イベント参照タスク ID（または名称） イベントマスク値格納領域およびそのアドレス
パラメータ(出力): *Mask	 現在のイベントマスク値
説明:	<TaskID>のイベントマスク値を取得します。
詳細:	タスク、割り込み、いくつかのフックルーチンから呼び出すことができます。 <TaskID>は拡張タスクでなければなりません。
エラーステータス: 標準 拡張	 • 正常終了： **E_OK** • 無効レベルからの呼び出し：**E_OS_CALLEVEL** • <TaskID>が無効：**E_OS_ID** • <TaskID>が拡張タスクでない：**E_OS_ACCESS** • <TaskID>が休止状態：**E_OS_STATE**
コンフォーマンスクラス:	ECC1, ECC2

12.5.2.4 WaitEvent

構文	StatusType **WaitEvent**(EventMaskType <Mask>)
パラメータ(入力): Mask	待ちイベントマスク値（または名称）
パラメータ(出力):	なし
説明:	呼び出しタスクのイベントに対して、<Mask>で指定されたイベントが１つもセットされてなかった場合、待ち状態に遷移します。 イベントがすでにセットされていた場合、状態は変化しません。 タスクは１つ以上のイベントを待つことができます。例えば、タスクがイベント ID の 0x01 および 0x80 を待つ場合、*WaitEvent*(0x01 | 0x80)を実行します。
詳細:	待ち状態に遷移する場合、再スケジューリングされます。 タスクがリソースを占有している場合、*WaitEvent* システムサービスを呼び出す前にリソースを解放する必要があります。標準エラーステータスでは、リソース占有時にイベント待ちした場合、動作は保証されません。 本システムサービスは拡張タスクのみ呼び出すことができます。
エラーステータス: 標準 拡張	● 正常終了： **E_OK** ● タスクレベル以外からの呼び出し：**E_OS_CALLEVEL** ● 基本タスクからの呼び出し：**E_OS_ACCESS** ● リソース占有中： **E_OS_RESOURCE**
コンフォーマンスクラス:	ECC1, ECC2

12.6 カウンタ、アラーム管理

12.6.1 データ型

- **TickType**
  カウンタ値

- **TickRefType**
  TickType へのポインタ

- **AlarmBaseType**
  カウンタ特性の要素

  - **TickType maxallowedvalue**: 最大カウント値
  - **TickType ticksperbase**: カウンタ仕様（OS 未使用）
  - **TickType mincycle**: 最小周期

- **AlarmBaseRefType**
  AlarmBaseType へのポインタ

- **AlarmType**
  アラーム ID

12.6.2 システムサービス

12.6.2.1 GetAlarmBase

構文	StatusType **GetAlarmBase**(AlarmType <AlarmID>, AlarmBaseRefType <Info>)
パラメータ(入力): AlarmID Info	 参照アラーム ID（または名称） カウンタ特性格納領域およびそのアドレス
パラメータ(出力): *Info	 カウンタ特性
説明:	カウンタ特性を取得します。 <Info> にデータ型 AlarmBaseType の情報（最大カウント値、カウンタ仕様、最小周期）が格納されます。
詳細:	タスク、割り込み、いくつかのフックルーチンから呼び出すことができます。 非可変アラームには使用できません。
エラーステータス: 標準 拡張	 • 正常終了： **E_OK** • 無効レベルからの呼び出し： **E_OS_CALLEVEL** • <AlarmID>が無効： **E_OS_ID**
コンフォーマンスクラス:	BCC1,BCC2,ECC1, ECC2

12.6.2.2 GetAlarm

構文	StatusType **GetAlarm**(AlarmType <AlarmID>, TickRefType <Tick>)
パラメータ(入力): AlarmID Tick	 参照アラーム ID（または名称） カウント値格納領域およびそのアドレス
パラメータ(出力): *Tick	 <AlarmID>が満了するまでの相対カウント値
説明:	<AlarmID>が満了するまでの相対カウント値を取得します。
詳細:	<AlarmID>が未使用の場合、 <Tick>は不定です。 タスク、割り込み、いくつかのフックルーチンから呼び出すことができます。 非可変アラームには使用できません。
エラーステータス: 標準 拡張	 • 正常終了： **E_OK** • <AlarmID>未使用： **E_OS_NOFUNC** • 無効レベルからの呼び出し： **E_OS_CALLEVEL** • <AlarmID>が無効： **E_OS_ID**
コンフォーマンスクラス:	BCC1,BCC2,ECC1, ECC2

12.6.2.3 SetRelAlarm

構文	StatusType **SetRelAlarm**(AlarmType <AlarmID>, TickType <increment>, TickType <cycle>)
パラメータ(入力): AlarmID increment cycle	 アラーム ID （または名称） 最初のアラーム満了の相対カウント値 周期カウント値
パラメータ(出力):	なし
説明:	<AlarmID>を活性化します。<increment>カウント値が経過した後、関連するアクション(タスク起動またはイベントセット)が実行されます。
詳細:	シングルアラームの場合、<cycle>に 0 を設定して下さい。 <increment>が 0 の場合、満了カウントは現在のカウント値に設定されます。 周期アラーム（<cycle>が 0 以外）の場合、<cycle>にしたがって周期的に満了します。 アラームがすでに使用されている場合、無視され、**E_OS_STATE** が返ります。 タスク、割り込みから呼び出すことができます。 非可変アラームには使用できません。
エラーステータス: 標準 拡張	 • 正常終了： **E_OK** • <AlarmID>使用中： **E_OS_STATE** • 無効レベルからの呼び出し： **E_OS_CALLEVEL** • <AlarmID>が無効： **E_OS_ID** • <increment>不正(0 より小さいまたは **maxallowedvalue** より大きい)： **E_OS_VALUE** • <cycle>不正 (**mincycle** より小さいまたは **maxallowedvalue** より大きい)： **E_OS_VALUE**
コンフォーマンスクラス:	BCC1,BCC2, ECC1, ECC2（アラームがイベントをセットする場合、ECC1,ECC2 のみ使用可能）

12.6.2.4 SetAbsAlarm

構文	StatusType **SetAbsAlarm**(AlarmType <AlarmID>, TickType <start>, TickType <cycle>)
パラメータ(入力): AlarmID start cycle	 アラーム ID（または名称） 最初のアラーム満了の絶対カウント値 周期カウント値
パラメータ(出力):	なし
説明:	<AlarmID>を活性化します。<start>カウント値に達したとき、関連するアクション(タスク起動またはイベントセット)が実行されます。
詳細:	シングルアラームの場合、<cycle>に 0 を設定して下さい。 周期アラーム（<cycle>が 0 以外）の場合、<cycle>にしたがって周期的に満了します。 アラームがすでに使用されている場合、無視され、**E_OS_STATE** が返ります。 タスク、割り込みから呼び出すことができます。 非可変アラームには使用できません。
エラーステータス: 標準 拡張	 • 正常終了： **E_OK** • <AlarmID>使用中： **E_OS_STATE** • 無効レベルからの呼び出し： **E_OS_CALLEVEL** • <AlarmID>が無効： **E_OS_ID** • <start>不正(0 より小さいまたは **maxallowedvalue** より大きい)： **E_OS_VALUE** • <cycle>不正 (**mincycle** より小さいまたは **maxallowedvalue** より大きい)： **E_OS_VALUE**
コンフォーマンスクラス:	BCC1,BCC2, ECC1, ECC2（アラームがイベントをセットする場合、ECC1,ECC2 のみ使用可能）

12.6.2.5 CancelAlarm

構文	StatusType **CancelAlarm**(AlarmType <AlarmID>)
パラメータ(入力): AlarmID	アラーム ID（または名称）
パラメータ(出力):	なし
説明:	<AlarmID>をキャンセルします。
詳細:	タスク、割り込みから呼び出すことができます。 非可変アラームには使用できません。
エラーステータス: 標準 拡張	• 正常終了： **E_OK** • <AlarmID>未使用： **E_OS_NOFUNC** • 無効レベルからの呼び出し： **E_OS_CALLEVEL** • <AlarmID>が無効： **E_OS_ID**
コンフォーマンスクラス:	BCC1,BCC2,ECC1, ECC2

12.6.2.6 InitialiseCounter

構文	StatusType **InitialiseCounter**(CounterType <CounterID>, TickType <tick>)
パラメータ(入力): CounterID Tick	 カウンタ ID（または名称） カウンタ初期値
パラメータ(出力):	なし
説明:	カウンタの<tick>カウント値を初期化します。
詳細:	タスク、割り込みから呼び出すことができます。 カウンタが初期化されない場合、カウント値は増加されず、アラームは動作しません。 カウンタは再度初期化することが可能です。 非可変アラームタイマカウンタには使用できません。 **注：本システムサービスは独自機能です。**
エラーステータス: 標準 拡張	• 正常終了： **E_OK** • 無効レベルからの呼び出し：**E_OS_CALLEVEL** • <CounterID>が無効 **E_OS_ID** • <tick>不正(0 より小さいまたは **maxallowedvalue** より大きい)： **E_OS_VALUE**
コンフォーマンスクラス:	BCC1,BCC2,ECC1, ECC2

12.6.2.7 IncrementCounter

構文	StatusType **IncrementCounter**(CounterType <CounterID>)
パラメータ(入力): CounterID	カウンタ ID（または名称）
パラメータ(出力):	なし
説明:	<CounterID>のカウント値を１つ進めます。本システムサービスによってカウント値がアラームの満了時間に達した場合、アラームに関連されたアクションが実行されます。
詳細:	タスク、割り込みから呼び出すことができます。それ以外から呼び出さないで下さい。 拡張エラーステータスにおいて、複数のエラーが発生した場合、**E_OS_MULTIPLE** エラーが返ります。エラーが発生したアクションは取り消されますが、システムサービスの実行は継続されます。 カウンタが初期化されてない場合、カウント値は増加しません。 非可変アラームタイマカウンタには使用できません。 タスクと ISR から同じカウンタ ID に対して本システムサービスを発行しないでください。また、異なる割り込みマスクレベルから同じカウンタ ID に対して本システムサービスを発行しないでください。発行した場合、動作は保証されません。 **注：本システムサービスは独自機能です。**
エラーステータス: 標準 拡張	• 正常終了： **E_OK** • タスクの多重起動： **E_OS_LIMIT** • **E_OS_STATE** または **E_OS_LIMIT** が 2 回以上発生：**E_OS_MULTIPLE** • 無効レベルからの呼び出し： **E_OS_CALLEVEL** • <CounterID>が無効： **E_OS_ID** • イベントセットするタスクの状態が休止状態：**E_OS_STATE**
コンフォーマンスクラス:	BCC1,BCC2,ECC1, ECC2

12.7 システム管理

12.7.1 データ型

- **AppModeType**
  アプリケーションモード ID

12.7.2 システムサービス

12.7.2.1 GetActiveApplicationMode

構文	AppModeType **GetActiveApplicationMode**(void)
パラメータ(入力):	なし
パラメータ(出力): リターン値	アプリケーションモード ID
説明:	現在のアプリケーションモード ID を取得します。
コンフォーマンスクラス:	BCC1,BCC2,ECC1, ECC2

12.7.2.2 StartOS

構文	void **StartOS**(AppModeType <Mode>)
パラメータ(入力): Mode	アプリケーションモード ID（または名称）
パラメータ(出力):	なし
説明:	指定したアプリケーションモードで OS を開始します。
詳細:	不正なアプリケーションモードが指定された場合、OS の開始を取り消し、呼び出しもとに復帰します。 CPU 初期化を行った後、本システムサービスを呼び出してください。
コンフォーマンスクラス:	BCC1,BCC2,ECC1, ECC2

12.7.2.3 ShutdownOS

構文	StatusType **ShutdownOS**(StatusType <Error>)
パラメータ(入力): Error	エラー種別
パラメータ(出力):	なし
説明:	OS をシャットダウンします。
詳細:	StartOS 呼び出し元へ復帰します。タスクからのみ呼び出すことができます。
エラーステータス: 標準 拡張	● 呼び出しもとには復帰しません ● 無効レベルからの呼び出し： **E_OS_CALLEVEL（独自機能）**
コンフォーマンスクラス:	BCC1,BCC2,ECC1, ECC2

12.8 フックルーチン

12.8.1 システムサービス

12.8.1.1 ErrorHook

構文	void **ErrorHook** (StatusType <Error>, APICall <APICallId>, ContextType <ContextId>)
パラメータ(入力*): Error APICall ContextID	発生エラーコード エラーが発生したシステムサービス ID**（独自機能。表 A.3参照）** 呼び出しプログラムのコンテキスト ID**（独自機能。表 A.4参照）**
パラメータ(出力):	なし
説明:	システムサービスエラーが発生した場合、呼び出しもとに戻る前に、システムサービス最後で本フックルーチンが呼び出されます。
詳細:	10 章を参照してください。
コンフォーマンスクラス:	BCC1, BCC2, ECC1, ECC2

12.8.1.2 PreTaskHook

構文	void **PreTaskHook** (void)
パラメータ(入力*):	なし
パラメータ(出力):	なし
説明:	タスクが実行状態に遷移する前に OS によって呼び出されます。ただし、本フックルーチンが呼び出された時のタスクの状態は実行状態です。(*GetTaskID* システムサービスによってタスク ID を得ることができます)
詳細:	10 章を参照してください。
コンフォーマンスクラス:	BCC1, BCC2, ECC1, ECC2

12.8.1.3 PostTaskHook

構文	void **PostTaskHook** (TaskStateType <State>)
パラメータ(入力*): State	タスク状態（**独自機能。12.2.3参照**）
パラメータ(出力):	なし
説明:	タスクが実行状態から実行可能状態、待ち状態または休止状態に遷移するとき、OS によって呼び出されます。ただし、本フックルーチンが呼び出された時のタスクの状態は実行状態です。(*GetTaskID* システムサービスによってタスク ID を得ることができます)
詳細:	10 章を参照してください。
コンフォーマンスクラス:	BCC1, BCC2, ECC1, ECC2

12.8.1.4 StartupHook

構文	void **StartupHook** (void)
パラメータ(入力*):	なし
パラメータ(出力):	なし
説明:	スケジューリングされる前の OS 初期化の最後に呼び出されます。タスクを起動したり、ハードウェアの設定などが可能です。
詳細:	10 章を参照して下さい。
コンフォーマンスクラス:	BCC1, BCC2, ECC1, ECC2

12.8.1.5 ShutdownHook

構文	void **ShutdownHook** (StatusType <Error>)
パラメータ(入力*): Error	エラー種別
パラメータ(出力):	なし
説明:	*ShutdownOS* システムサービスまたは OS がシャットダウンした場合に呼び出されます。
詳細:	10.3節を参照してください。
コンフォーマンスクラス:	BCC1, BCC2, ECC1, ECC2

【注】* 入力パラメータは OS が設定します。

A 付録

A.1 システムサービスリターンコード

表 A.1 システムサービスリターンコード

システムサービス	標準 エラーステータス	拡張 エラーステータスで追加されるコード
ActivateTask	E_OK, E_OS_LIMIT*	E_OS_ID, E_OS_CALLEVEL
TerminateTask	---	E_OS_RESOURCE, E_OS_CALLEVEL
ChainTask	E_OS_LIMIT*	E_OS_ID, E_OS_RESOURCE, E_OS_CALLEVEL
Schedule	E_OK	E_OS_CALLEVEL
GetTaskID	E_OK	E_OS_CALLEVEL
GetTaskState	E_OK	E_OS_ID, E_OS_CALLEVEL
EnterISR	---	---
LeaveISR	---	---
EnableInterrupt または EnableInterruptMask（割り込みマスクレベル方式）*	---	---
DisableInterrupt または DisableInterruptMask（割り込みマスクレベル方式）*	---	---
GetInterruptDescriptor または GetInterruptDescriptorMask（割り込みマスクレベル方式）*	---	---
EnableInterrupt または EnableInterruptSource（割り込み要因方式）*	E_OK	E_OS_NOFUNC
DisableInterrupt または DisableInterruptSource（割り込み要因方式）*	E_OK	E_OS_NOFUNC
GetInterruptDescriptor または GetInterruptDescriptorSource（割り込み要因方式）*	---	---
GetResource	E_OK	E_OS_ID, E_OS_ACCESS, E_OS_CALLEVEL, E_OS_INVALID_NEST*
ReleaseResource	E_OK	E_OS_ID, E_OS_NOFUNC, E_OS_CALLEVEL
SetEvent	E_OK	E_OS_ID, E_OS_ACCESS, E_OS_STATE, E_OS_CALLEVEL

表 A.1 システムサービスリターンコード（続き）

システムサービス	標準 エラーステータス	拡張 エラーステータスで追加されるコード
ClearEvent	E_OK	E_OS_ACCESS, E_OS_CALLEVEL
GetEvent	E_OK	E_OS_ID, E_OS_ACCESS, E_OS_STATE, E_OS_CALLEVEL
WaitEvent	E_OK	E_OS_ACCESS, E_OS_RESOURCE, E_OS_CALLEVEL
GetAlarmBase	E_OK	E_OS_ID, E_OS_CALLEVEL
GetAlarm	E_OK, E_OS_NOFUNC	E_OS_ID, E_OS_CALLEVEL
SetRelAlarm	E_OK, E_OS_STATE	E_OS_ID, E_OS_VALUE, E_OS_CALLEVEL
SetAbsAlarm	E_OK, E_OS_STATE	E_OS_ID, E_OS_VALUE, E_OS_CALLEVEL
CancelAlarm	E_OK, E_OS_NOFUNC	E_OS_ID, E_OS_CALLEVEL
InitialiseCounter*	E_OK	E_OS_ID, E_OS_CALLEVEL, E_OS_VALUE
IncrementCounter*	E_OK, E_OS_LIMIT, E_OS_MULTIPLE	E_OS_ID, E_OS_STATE, E_OS_CALLEVEL
GetActiveApplicationMode	---	---
StartOS	---	---
ShutdownOS	---	E_OS_CALLEVEL*

【注】* 独自機能

A.2 ID

A.2.1 リターンコード ID

表 A.2 リターンコード

エラー	**説明**	**ID**
E_OK	正常終了	0 (H’0)
E_OS_ACCESS	・リソース占有中、リソースが使用定義外 ・拡張タスクでない ・基本タスクからの呼び出し	1 (H’1)
E_OS_CALLEVEL	無効レベルからの呼び出し	2 (H’2)
E_OS_ID	ID 無効	3 (H’3)
E_OS_LIMIT*	多重起動超過	4 (H’4)
E_OS_NOFUNC	・オブジェクト未使用 ・リソースが獲得されてない、またはリソース開放順序不正（他のリソースを先に開放しなければならない） ・割り込みがすでに有効または無効	5 (H’5)
E_OS_RESOURCE	リソース占有中	6 (H’6)
E_OS_STATE	・タスクが休止状態 ・アラーム使用中	7 (H’7)
E_OS_VALUE	・カウント値不正(0 より小さいまたは **maxallowedvalue** より大きい) ・周期不正 (**mincycle** より小さいまたは **maxallowedvalue** より大きい)	8 (H’8)
E_OS_INVALID_NEST*	リソースネスト不正	9 (H’9)
E_OS_MULTIPLE*	**E_OS_STATE** または **E_OS_LIMIT** が 2 回以上発生	10 (H’a)
E_OS_SYS_EXCEPTION*	未定義例外	100 (H’64)
E_OS_SYS_ILLEGAL*	一般不当命令	101 (H’65)
E_OS_SYS_ILLEGAL_SLOT*	スロット不当命令	102 (H’66)
E_OS_SYS_CPU_ADDRESS*	CPU アドレスエラー	103 (H’67)

【注】* 独自機能

A.2.2 システムサービス ID

表 A.3 システムサービス ID

システムサービス	ID
DeclareTask	0 (H’0)
ActivateTask	1 (H’1)
TerminateTask	2 (H’2)
ChainTask	3 (H’3)
Schedule	4 (H’4)
GetTaskID	5 (H’5)
GetTaskState	6 (H’6)
EnterISR	7 (H’7)
LeaveISR	8 (H’8)
EnableInterrupt	9 (H’9)
DisableInterrupt	10 (H’a)
GetInterruptDescriptor	11 (H’b)
DeclareResourc	12 (H’c)
GetResource	13 (H’d)
ReleaseResource	14 (H’e)
DeclareEvent	15 (H’f)
SetEvent	16 (H’10)
ClearEvent	17 (H’11)
GetEvent	18 (H’12)
WaitEvent	19 (H’13)
DeclareAlarm	20 (H’14)
GetAlarmBase	21 (H’15)
GetAlarm	22 (H’16)
SetRelAlarm	23 (H’17)
SetAbsAlarm	24 (H’18)
CancelAlarm	25 (H’19)
GetActiveApplicationMode	26 (H’1a)
StartOS	27 (H’1b)
ShutdownOS	28 (H’1c)
InitialiseCounter*	29 (H’1d)
IncrementCounter*	30 (H’1e)
NonVariant (非可変アラームタイマカウンタ)	31 (H’1f)
EnableInterruptMask*	32 (H’20)
DisableInterruptMask*	33 (H’21)
GetInterruptDescriptorMask*	34 (H’22)
EnableInterruptSource*	35 (H’23)
DisableInterruptSource*	36 (H’24)
GetInterruptDescriptorSource*	37 (H’25)

【注】* 独自機能

A.2.3 コンテキスト ID

表 A.4 コンテキスト ID

コンテキスト	**ID**
OS	0 (H’0)
OS_IDLE	1 (H’1)
TASK	2 (H’2)
ISR	3 (H’3)
ERRORHOOK	4 (H’4)
PRETASKHOOK	5 (H’5)
POSTTASKHOOK	6 (H’6)
SHUTDOWNHOOK	7 (H’7)
STARTUPHOOK	8 (H’8)

A.3 システムサービス呼び出し

タスク、ISR、フックルーチンにおいて発行可能なシステムサービスを✔で示します。

表 A.5 システムサービス

システムサービス	タスク	ISR	Error フック	PreTask フック	PostTask フック	Startup フック	Shutdown フック	OS 呼出しプログラム
ActivateTask	✔	✔	---	---	---	✔	---	---
TerminateTask	✔	---	---	---	---	---	---	---
ChainTask	✔	---	---	---	---	---	---	---
Schedule	✔	---	---	---	---	---	---	---
GetTaskID	✔	---	✔	✔	✔	---	---	---
GetTaskState	✔	✔	✔	✔	✔	---	---	---
EnableInterrupt	✔	✔	---	---	---	---	---	---
DisableInterrupt	✔	✔	---	---	---	---	---	---
GetInterruptDescriptor	✔	✔	✔	✔	✔	---	---	---
GetResource	✔	---	---	---	---	---	---	---
ReleaseResource	✔	---	---	---	---	---	---	---
SetEvent	✔	✔	---	---	---	---	---	---
ClearEvent	✔	---	---	---	---	---	---	---
GetEvent	✔	✔	✔	✔	✔	---	---	---
WaitEvent	✔	---	---	---	---	---	---	---
GetAlarmBase	✔	✔	✔	✔	✔	---	---	---
GetAlarm	✔	✔	✔	✔	✔	---	---	---
SetRelAlarm	✔	✔	---	---	---	---	---	---
SetAbsAlarm	✔	✔	---	---	---	---	---	---
CancelAlarm	✔	✔	---	---	---	---	---	---
InitialiseCounter	✔	✔	---	---	---	---	---	---
IncrementCounter	✔	✔	---	---	---	---	---	---
GetActiveApplicationMode	✔	✔	✔	✔	✔	✔	✔	---
StartOS	---	---	---	---	---	---	---	✔
ShutdownOS	✔	---	---	---	---	---	---	---
EnableInterruptMask	✔	✔	---	---	---	---	---	---
DisableInterruptMask	✔	✔	---	---	---	---	---	---
GetInterruptDescriptorMask	✔	✔	✔	✔	✔	---	---	---
EnableInterruptSource	✔	✔	---	---	---	---	---	---
DisableInterruptSource	✔	✔	---	---	---	---	---	---
GetInterruptDescriptorSource	✔	✔	✔	✔	✔	---	---	---

A.4 データ型

表 A.6 データ型

名称	型（サイズ）	説明
StatusType	unsigned long （4バイト）	リターン値
TaskType	signed short（2バイト）	タスク ID
TaskRefType	TaskType *（4バイト）	TaskType へのポインタ
TaskStateType	signed char （1バイト）	タスク状態
TaskStateRefType	TaskStateType *（4バイト）	TaskStateType へのポインタ
IntDescriptorType	unsigned long （4バイト）	ステータスレジスタ値
IntDescriptorRefType	IntDescriptorType *（4バイト）	IntDescriptorType へのポインタ
ResourceType	signed char （1バイト）	リソース ID
EventMaskType	signed char （1バイト）	イベントマスク値
EventMaskRefType	EventMaskType *（4バイト）	EventMaskType へのポインタ
TickType	unsigned long（4バイト）	カウンタ値
TickRefType	TickType *（4バイト）	TickType へのポインタ
AlarmBaseType	struct AlarmBase（12バイト）	カウンタ特性の要素
AlarmBaseRefType	AlarmBaseType *（4バイト）	AlarmBaseType へのポインタ
AlarmType	signed char （1バイト）	アラーム ID
AppModeType	signed char （1バイト）	アプリケーションモード ID
CounterType	signed char （1バイト）	カウンタ ID
APICall	unsigned long（4バイト）	システムサービス ID
ContextType	unsigned long（4バイト）	コンテキスト ID

Ho7000 シリーズ
オペレーティングシステムマニュアル

ルネサス エレクトロニクス株式会社
神奈川県川崎市中原区下沼部1753 〒211-8668

ADJ-702-392