文書番号：NGKA_SPEC-01-320

NCES

次世代車載システム向け RTOS
外部仕様書

Ver.3.2.0

2014/3/12

名古屋大学大学院情報科学研究科附属 組込みシステム研究センター

＜目次＞

1. 概要

1.1　本文書の目的

本文書は「次世代車載システム向け RTOS 要求仕様書」に記述された要求事項を満たすリアルタイム OS(以降，OS と略す)の機能仕様を規定するものである．

本仕様は AUTOSAR Specification of Operating System 仕様(以降，AUTOSAR 仕様と略す)をベースに，必要な拡張，修正を行ったものである．また，本仕様は AUTOSAR 仕様が OSEK/VDX Operating System 仕様(以降，OSEK 仕様と略す)をベースにしていることから，間接的に OSEK 仕様もベースにしている【OS001】.

本文書は，OS に関する一般的な知識を持ったソフトウェア技術者が読むことを想定して記述している．OSEK 仕様や AUTOSAR 仕様に関する知識があることが望ましいが，それを前提とせず記述している．

1.2　関連文書

1.2.1　ベースとした文書

以下の表は，本文書のベースとする文書であり，その内容は本文書内に包含されている．

文書名	バージョン
OSEK/VDX Operating System	Ver.2.2.3
AUTOSAR Specification of Operating System	V5.0.0 (R4.0 Rev 3)

1.2.2　参考文書

以下の表は，本文書から参照している文書，または本文書を理解するために必要な文書である．内容は本文書に包含されていない．

文書名	バージョン
次世代車載システム向け RTOS 要求仕様書	Ver.3.0.0
次世代車載システム向け RTOS ハードウェア要求仕様書	Ver.3.0.0
次世代車載システム向け RTOS 用語集	Ver.3.0.0
AUTOSAR Specification of Standard Types	V1.3.0 (R4.0 Rev 1)
AUTOSAR List of Basic Software Modules	V1.6.0 (R4.0 Rev 3)
Software Component Template	V4.2.0 (R4.0 Rev 3)

1.3　凡例

本文書では，OSEK 仕様と，AUTOSAR 仕様と，名古屋大学大学院情報科学研究科附属組込みシステム研究センター(NCES)を中心とする次世代車載システム向け RTOS の仕様検討及び開発に関するコンソーシアム型共同研究(ATK2 コンソーシアム)で，新規に規定した仕様が混在しているため，以下に示す仕様番号を用いてこれらの仕様を区別して管理を行う．仕様番号は，要求事項にのみ付与することを基本とする．ただし，AUTOSAR 仕様において，概念の説明や補足事項についても仕様番号が付与されているものに関しては，そのまま付与する．また，本文書から ATK2 コンソーシアムで開発した OS である ATK2 の実装に依存する仕様を，参考情報として本文書に記載するため，仕様番号を付与する．

仕様番号	内容
【COS*xxxx*】	関連文書 OSEK/VDX Operating System で規定された仕様．AUTOSAR 仕様では CoreOS として扱われるため，COS と表記を行う．
【OS*xxx*】	関連文書 AUTOSAR Specification of Operating System で規定された仕様．AUTOSAR 仕様で記述されている OS 仕様番号を用いる．本仕様に採用しなかった AUTOSAR 仕様についても，【OS*xxx*】で記載している．
【OS_Conf*xxx*】 【MCOS_Conf*xxxx*】	関連文書 AUTOSAR Specification of Operating System で規定された仕様．AUTOSAR 仕様でコンフィギュレーション情報に関して記述されている OS 仕様番号を用いる．AUTOSAR 仕様では，【OS*xxx*_Conf】という表記であるが，プレフィックスに統一するため，本仕様では左記の表記とする．本仕様に採用しなかった AUTOSAR 仕様についても，【OS_Conf*xxx*】で記載している．同様に，マルチコア対応 OS におけるコンフィギュレーション情報に関する OS 仕様番号は【MCOSx*xxx*_Conf】という表記であるが，本仕様では左記の表記とする．
【OSa*xxx*】	関連文書 AUTOSAR Specification of Operating System に記述されているが，仕様番号表記がないもの．AUTOSAR 仕様の要求事項であるため，本仕様では，仕様番号を付与する．
【NOS*xxxx*】	ATK2 コンソーシアムで新規に規定した仕様．
【IOS*xxx*】	ATK2 の実装において規定した仕様．【COS*xxxx*】【OS*xxx*】【NOS*xxxx*】において，実装定義と規定されている仕様や，その他の仕様だけでは実装が不明確である場合に，参考情報として本文書に記載する．
〔COS*xxx*〕 〔OS*xxx*〕 〔NOS*xxx*〕	本文中で上記の仕様番号を内部参照する際に使用する． 仕様定義である【COS*xxxx*】【OS*xxx*】【NOS*xxxx*】と区別するため，〔COS*xxxx*〕〔OS*xxx*〕〔NOS*xxxx*〕とする．
≪COS*xxx*≫ ≪OS*xxx*≫ ≪NOS*xxx*≫	本文中で上記の仕様番号を引用する際に使用する． 仕様定義である【COS*xxxx*】【OS*xxx*】【NOS*xxxx*】と区別するため，≪COS*xxxx*≫≪OS*xxx*≫≪NOS*xxxx*≫とする．

AUTOSAR 仕様との違い

削除や改変を行った AUTOSAR 仕様に対して，どのような差分があるかを説明する．

OSEK 仕様との違い

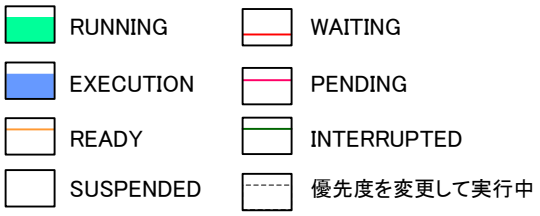
削除や改変を行った OSEK 仕様に対して，どのような差分があるかを説明する．ただし，本書だけで理解可能である改変に関しては記載しない．

使用上の注意

本仕様に準拠した OS の開発者ではなく，本 OS を使用してアプリケーション開発を行うユーザに対する，注意事項もしくは推奨事項を説明する．

図中で用いる処理単位の状態

本書の図の中で用いる，タスクなどの処理単位の状態の意味は以下の通りである．



1.4　本仕様書に対応する ATK2 のバージョン

本仕様書に対応する ATK2 のバージョンは，以下である．

- ATK2-SC1 Release 1.2.0
- ATK2-SC3FL2 Release 1.2.0
- ATK2-SC1-MC Release 1.1.0
- ATK2-SC3FL2-MC Release 1.1.0

2. 概念

2.1　機能一覧

本 OS が提供する機能の概略を以下に示す.

タスク管理

- タスクの起動，終了.
- タスクディスパッチ機構.

割込み処理

- 外部割込みに対する割込みサービスルーチン(以降，ISR と略す)の起動.
- OS が管理しないカテゴリ 1ISR(以降, C1ISR と略す)と OS が管理するカテゴリ 2ISR(以降, C2ISR と略す)のサポート.
- 割込みの禁止，許可操作.

イベント同期処理

- イベントを利用したタスクの同期機能.

リソース管理

- タスク，C2ISR 間の排他制御機能.

カウンタ管理

- システム外部のハードウェアで発生する事象，およびシステム内部のソフトウェアで発生する事象をカウント(カウンタの駆動)する機能.

アラーム管理

- カウンタによって駆動される繰り返し処理(タスク起動，イベントセット，コールバックルーチン起動，他のカウンタに対するカウント)を行うための機能.

スケジュールテーブル管理

- カウンタによって駆動される一連の繰り返し処理(タスク起動，イベントセットの複数の組み合わせ)を静的なテーブルに記載されたタイミングで実行する機能.

タイミング保護

- タスク，ISR に与えられた時間制約の監視.

OS アプリケーション(以降，OSAP と略す)管理

- OSAP の強制終了，再起動.

- 信頼関数を実行する機能.

メモリ保護

- OSAP によって分割されたプログラム間の不正なメモリアクセスの防止.

サービス保護

- OS を危険な状態にする振る舞いや OSAP が起こすエラーからの保護.
- OSEK 仕様のエラーチェックの強化.

保護違反時処理

- タイミング保護，メモリ保護，CPU 例外で発生した保護違反のアプリケーションへの通知.
- 違反したプログラムに対する強制終了処置.

エラー処理

- OS が提供するサービス内で発生したエラーに対する処理.

IOC(Inter-OS-Application Communicator)

- コアやメモリ保護の境界をまたいで通信するために用意された OSAP 間の通信機能.

マルチコア対応

- コアを跨いだシステムサービス.
- OS の起動，終了のマルチコア対応.
- スピンロック管理.
- コア間割込み処理.
- ミューテックス管理(参考仕様)

2.2　主要概念

2.2.1　処理単位

本 OS の処理単位を以下に示す.

2.2.1.1　タスク

タスクは OS によって実行されるプログラムの処理単位である．複数のタスクが存在する場合，タスクはそれぞれ並行に実行され, 実行順序は OS によってスケジューリングされる(詳細は 2.5 節を参照).

2.2.1.2　ISR

ISR は割込みコントローラからの割込み要求をトリガとして起動されるプログラムの処理単位である(詳細は 2.7 節を参照).

2.2.1.3　フックルーチン

フックルーチンは OS 処理内の特定のタイミングに，OS によって実行される処理単位である(詳細は 2.14 節を参照).

2.2.1.4　アラームコールバック

アラームコールバックはアラーム満了時に, OS によって実行される処理単位である(詳細は 2.12.4 節を参照).

2.2.2　プロセッサコア

ソフトウェアの視点で 1 つの処理単位のみを同時に実行できるハードウェアの単位を「プロセッサコア」または単に「コア」と呼ぶ. マルチスレッドプロセッサにおけるスレッドも 1 つのコアとして扱う.

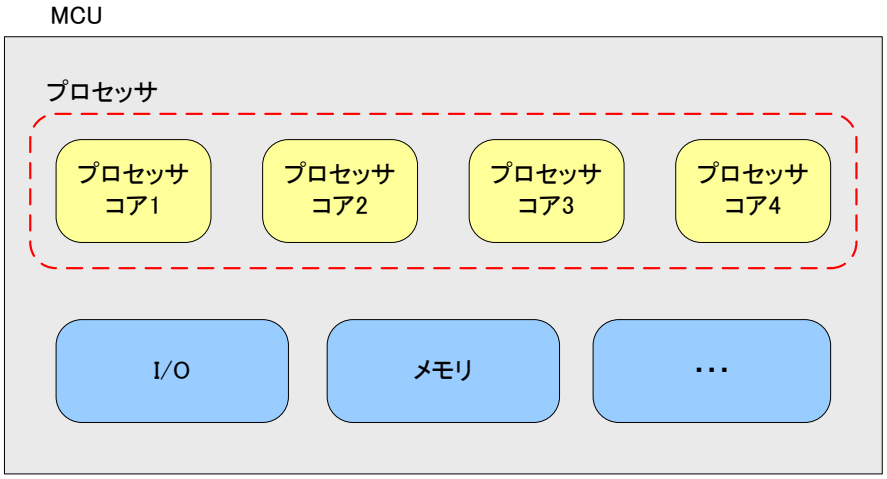


図 2-1 プロセッサコアの位置づけ

シングルコアとマルチコアの定義

コアの数が 1 つのシステムをシングルコアシステムと呼び，コアが複数存在するシステムをマルチコアシステムと呼ぶ.

2.2.3　メモリリージョン

本節は，メモリ保護機能をサポートする SC3，SC4 に対する説明である．

一般に，ターゲットハードウェアには性質の異なる複数のメモリ領域が存在する．OS オブジェクトのコンフィギュレーションにおいて，OS の内部データの配置や，アクセス保護を行うメモリ領域(以降，保護領域と略す)の指定など，複数のメモリデバイスから特定のメモリ領域を意識してコンフィギュレーションすることが必要になる．ターゲットハードウェアに存在するメモリ領域の中で，同じ性質を持つ連続したメモリ領域のことをメモリリージョンと呼ぶ．ハードウェアにどのような性質のメモリリージョンが存在するかは実装定義である【NOS0010】．ATK2 では，ハードウェアにどのような性質のメモリリージョンが存在するかは，ターゲット定義とする【IOS136】.

<u>メモリリージョンの性質</u>

一般に，ターゲットに存在するメモリがもつ性質として，以下の性質が挙げられる．マルチコアシステムのハードウェアにおけるメモリの性質に関しての詳細は「次世代車載システム向け RTOS ハードウェア要求仕様書」を参照.

- 書込み可否(ROM，RAM)
- 命令フェッチの可否(フェッチ可能，データ専用)
- アクセス速度(内部メモリ，外部メモリ)
- キャッシュ(キャッシャブル，ノンキャッシャブル)
- 揮発性(電源 OFF 後のデータ保持可否)
- マルチコアシステムにおける共有アクセス可否(プライベート，グローバル)
- マルチコアシステムにおけるアクセス速度(ローカルバス接続，グローバルバス接続)

<u>メモリリージョンの命名規則</u>

メモリリージョン名として統一された意味づけをするために，メモリリージョンの命名規則を定める【NOS0011】．命名規則は以下のとおりである．

規定の無いものは，実装定義とする【NOS0907】.

[<メモリ共有種別>]_<ROM/RAM>_[<フェッチ可否>]_[<その他>]_[<番号>]
※[　]内は省略可を示す

<メモリ共有種別>
　PRIVATE：プライベートメモリ
　LOCAL：ローカルメモリ
　GLOBAL：グローバルメモリ
<ROM/RAM>
　ROM：ROM

RAM：RAM

<フェッチ可否>

INST：命令専用

DATA：データ専用

<その他>

CACHE：キャッシュ有効

EXT：外部メモリ

<番号>

同じ性質のメモリリージョンがシステム中に複数存在する場合の識別のための番号

シングルコアシステムにおける例：

ROM_INST_1, ROM_DATA_1, RAM_1, RAM_2

マルチコアシステムにおける例：

PRIVATE_ROM_INST_1, ROM_DATA_1, GLOBAL_RAM_EXT_1, LOCAL_RAM_CACHE_2

<フェッチ可否> 省略時は命令・データフェッチに制限が無いことを意味する【NOS0012】.

<メモリ共有種別> 省略時に，ローカル，グローバルのどちらを意味するかは，実装定義である【NOS0013】. ATK2 では，メモリ共有種別を省略時にローカルもしくはグローバルのどちらかを意味するかは，ターゲット定義とする【IOS137】.

メモリリージョンの管理

ターゲットがサポートするすべてのメモリリージョンをジェネレータが管理する【NOS0014】. ジェネレータはコンフィギュレーションファイルにある記述とメモリリージョン管理情報を元に，ロードモジュールのリンクに必要なファイル(リンカスクリプト等)を出力する【NOS0015】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，メモリ領域を意識したコンフィギュレーションの仕組みが定義されていないため，本仕様で規定を行った.

2.2.4　処理レベル

異なる処理単位の間において，処理が実行される優先順位を処理レベルという．OS の持つ処理レベルは以下の 6 種類で，記載順に高い処理レベルを持つ【NOS0242】.

- 全割込み禁止時間監視保護違反時処理レベル(タイミング保護を行う場合のみ)
- C1ISR 処理レベル
- OS 処理レベル
- C2ISR 処理レベル
- スケジューラ処理レベル
- タスク処理レベル

全割込み禁止時間監視保護違反時処理レベル

タイミング保護機能における全割込み禁止時間監視により保護違反が発生した際に，保護違反時処理が動作する処理レベルである．最も高い処理レベルで動作する．

C1ISR 処理レベル

C1ISR が動作する処理レベルである．C1ISR は OS の管理外であり，OS よりも高い処理レベルで動作する．

OS 処理レベル

OS の機能のうち，システムサービス，フックルーチン，アラームコールバック，ハードウェアカウンタ制御関数，保護違反時処理が動作する処理レベルである．

C2ISR 処理レベル

C2ISR が動作する処理レベルである．C2ISR は OS の管理下にあり，OS よりも低い処理レベルで動作する．

スケジューラ処理レベル

OS 機能におけるスケジューラが動作する概念的な処理レベルである．

タスク処理レベル

タスクが動作する処理レベルである．

2.2.5　優先度

タスクや割込みが動作する処理レベルでは，複数の処理単位が入れ換わり動作する．これらの処理単位において，処理が実行される優先順位を優先度と呼ぶ．タスクが実行される優先順位をタスク優先度と呼び，割込みが実行される優先順位を割込み優先度と呼ぶ．

タスク優先度の詳細は 2.5.6 節，割込み優先度の詳細は 2.7.4 節を参照．

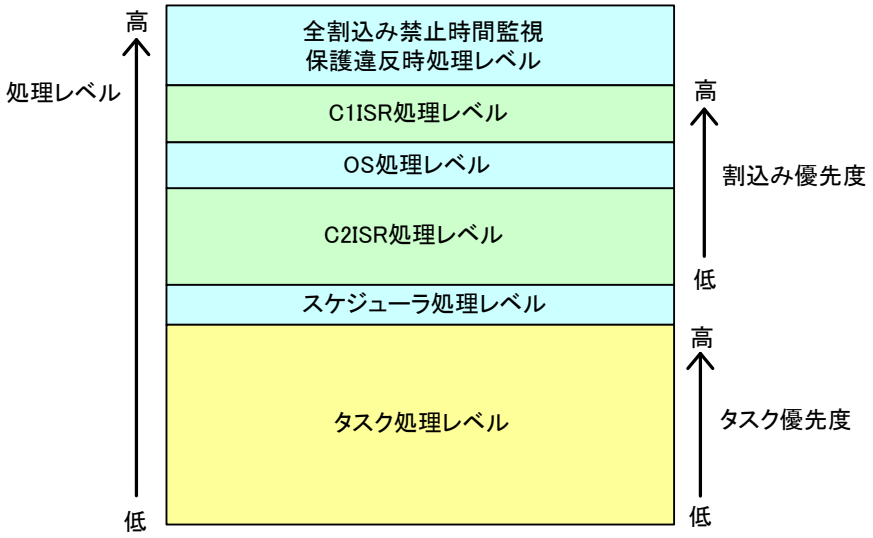


図 2-2 処理レベルと優先度

OSEK 仕様との違い

OSEK 仕様では，割込みレベル，スケジューラレベル，タスクレベルの 3 つの処理レベルを規定している【COS0301】．しかし，AUTOSAR におけるフックルーチンやタイミング保護の仕様を明確にするために，6 つの処理レベルを規定した≪NOS0242≫．

2.2.6　スタック

本 OS で管理するスタックを以下に示す.

2.2.6.1　タスク用スタック

SC1，SC2 において，タスクが使用するスタックである．タスク用スタックは，スタック共有する場合を除いて，タスク毎に確保される【NOS0826】．各タスク用スタックのサイズは，コンフィギュレーション時に静的に指定する≪NOS0282≫.

2.2.6.2　C1ISR 用スタック

C1ISR が使用するスタックである．OS では，C1ISR 用スタックを管理しない【NOS0889】．割込み発生時の処理単位のスタックをそのまま使用するか，C1ISR 用スタックを別途用意して，割込み発生時に切り替えるかなどの取り扱いはアプリケーション依存である【NOS0890】.

使用上の注意

C1ISR 用スタックとして，割込み発生時の処理単位のスタックをそのまま使用する場合，C1ISR が使用するスタックサイズも考慮して，各処理単位用のスタックサイズを決定する必要がある.

2.2.6.3　C2ISR 用スタック

SC1，SC2 において，C2ISR が使用するスタックである．C2ISR 用スタックの確保方法は，実装定義である【NOS0827】．ATK2 の SC1，SC2 では，C2ISR 用スタックは，フック用スタックと合わせて 1 つのスタックで確保される【IOS083】．各 C2ISR 用スタックのサイズは，コンフィギュレーション時に静的に指定する≪NOS0284≫.

2.2.6.4　フック用スタック

SC1，SC2 において，すべてのフックルーチンが使用するスタックである．フック用スタックの確保方法は，実装定義である【NOS0828】．ATK2 の SC1，SC2 では，フック用スタックは，C2ISR 用スタックと合わせて 1 つのスタックで確保される≪IOS083≫．フック用スタックのサイズは，コンフィギュレーション時に静的に指定する≪NOS0825≫.

2.2.6.5　信頼タスク用スタック

SC3，SC4 において，信頼タスクが使用するスタックである．信頼タスク用スタックは，スタック共有する場合を除いて，信頼タスク毎に確保される【NOS0829】．各信頼タスク用スタックのサイズは，コンフィギュレーション時に静的に指定する≪NOS0282≫.

2.2.6.6　非信頼タスク用スタック

SC3，SC4 において，非信頼タスクが使用するメモリ保護対象のスタックである．非信頼タスク用スタックは，スタック共有する場合を除いて，非信頼タスク毎に確保される【NOS0830】．各非信頼タス

ク用スタックのサイズは，コンフィギュレーション時に静的に指定する≪NOS0282≫.

2.2.6.7　非信頼タスク用システムスタック

SC3，SC4 において，非信頼タスクからシステムサービス，信頼関数が呼び出された際に使用するスタックである．非信頼タスク用システムスタックは，スタック共有する場合を除いて，非信頼タスク毎に確保される【NOS0831】．各非信頼タスク用スタックのサイズは，コンフィギュレーション時に静的に指定する≪NOS0283≫.

2.2.6.8　信頼 C2ISR 用スタック

SC3, SC4 において, 信頼 C2ISR が使用するスタックである. 信頼 C2ISR 用スタックの確保方法は, 実装定義である【NOS0832】. ATK2 の SC3, SC4 のメモリ保護機能の機能レベル 1, 2 では, 信頼 C2ISR 用スタックは，信頼フック用スタックと合わせて 1 つのスタックで確保される【IOS084】．SC3，SC4 のメモリ保護機能の機能レベル 3 では，スタック共有する場合を除いて，信頼 C2ISR 毎にスタック領域が確保される(現状，機能レベル 3 は未実装)【IOS085】．信頼 C2ISR 用スタックのサイズは，コンフィギュレーション時に静的に指定する≪NOS0284≫.

2.2.6.9　非信頼 C2ISR 用スタック

SC3，SC4 のメモリ保護機能の機能レベル 3 において，非信頼 C2ISR が使用するメモリ保護対象のスタックである．非信頼 C2ISR 用スタックは，スタック共有する場合を除いて，非信頼 C2ISR 毎に確保される【NOS0833】．非信頼 C2ISR 用スタックのサイズは，コンフィギュレーション時に静的に指定する≪NOS0284≫.

2.2.6.10 非信頼 C2ISR 用システムスタック

SC3，SC4 のメモリ保護機能の機能レベル 3 において，非信頼 C2ISR からシステムサービス，信頼関数が呼び出された際に使用するスタックである．非信頼 C2ISR 用システムスタックは，スタック共有する場合を除いて，非信頼 C2ISR 毎に確保される【NOS0834】．非信頼 C2ISR 用スタックのサイズは，コンフィギュレーション時に静的に指定する≪NOS0285≫.

2.2.6.11 信頼フック用スタック

SC3，SC4 において，システム定義のフックルーチン，信頼フック，非信頼フックから呼び出されたシステムサービスが使用するスタックである．信頼フック用スタックの確保方法は，実装定義である【NOS0835】．ATK2 の SC3，SC4 のメモリ保護機能の機能レベル 1，2 では，信頼フック用スタックは，信頼 C2ISR 用スタックと合わせて 1 つのスタックで確保される≪IOS084≫．SC3，SC4 のメモリ保護機能の機能レベル 3 では，信頼フック用スタックはシステムに 1 つ確保される(現状，機能レベル 3 は未実装)【IOS086】．信頼フック用スタックのサイズは，コンフィギュレーション時に静的に指定する≪NOS0825≫.

2.2.6.12 非信頼フック用スタック

SC3，SC4 において，非信頼フックが使用するスタックである．非信頼フック用スタックは，システムに 1 つ確保される【NOS0836】．非信頼フック用スタックのサイズは，コンフィギュレーション時に静的に指定する≪NOS0780≫．

2.2.6.13 スタック領域の確保方法

スタック領域の確保方法は実装定義である【NOS0844】．ATK2 では，スタック領域の確保方法を，ターゲット定義とする【IOS133】．

2.3　OS 機能のクラス・レベル分け

本 OS では，OS 機能のクラス・レベル分けとして，スケーラビリティクラス，機能レベル，コンフォーマンスクラスの概念を持つ．それぞれの関係を図 2-3 に示す．なお，スケーラビリティクラスとコンフォーマンスクラスは独立した概念であるため，依存関係はない．

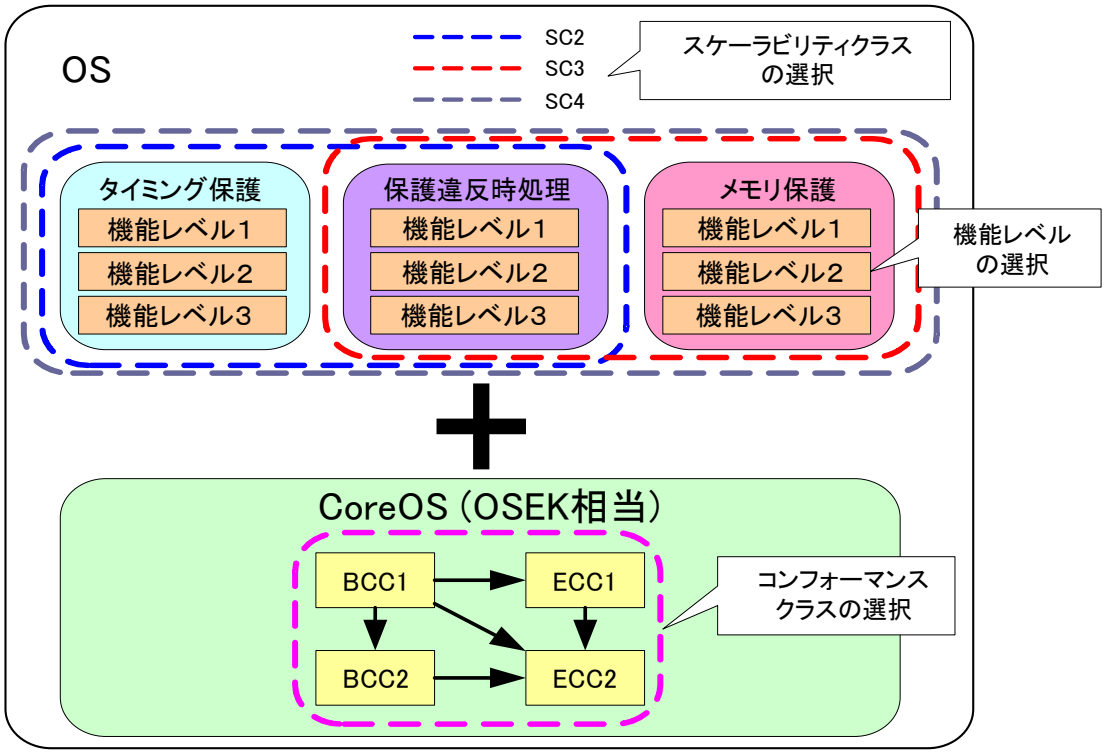


図 2-3 OS 機能のクラス・レベルの関係

2.3.1　スケーラビリティクラス

スケーラビリティクラスはユーザに必要な機能とターゲットの持つ機能によって OS 機能を定義した機能セットである【OS241】. スケーラビリティクラスごとの OS 機能の一覧を 表 2-1 に示す.

表 2-1 スケーラビリティクラスごとの機能一覧

機能	SC1	SC2	SC3	SC4
OSEK OS(全コンフォーマンスクラス)	○	○	○	○
アラームコールバック	○			
カウンタ管理システムサービス	○	○	○	○
ソフトウェアフリーランタイマサービス	○	○	○	○
スケジュールテーブル	○	○	○	○
スタックモニタリング	○	○	○	○
保護違反時処理(プロテクションフック)	○	○	○	○
CPU 例外に対する処理	○	○	○	○
タイミング保護		○		○
グローバル時間同期		○		○
メモリ保護			○	○
OSAP			○	○
サービス保護	※1	※1	○	○
信頼関数			○	○

※1：サービス保護の一部をサポート

各 SC で最低限サポートすべき OS オブジェクト数および OSAP 数を表 2-2 に示す.

表 2-2 各スケーラビリティクラスでサポートするべき最小 OS オブジェクト

機能	SC1	SC2	SC3	SC4
スケジュールテーブル数	2	8	2	8
ソフトウェアカウンタ数	8	8	8	8
OSAP 数	0	0	2	2

スケーラビリティクラス間の互換性

本仕様を準拠したインターフェースであれば，下位のスケーラビリティクラスの OS に，上位のスケーラビリティクラスの機能を追加実装してもよい【OS240】.

コンフォーマンスクラスとの関係

OSEK OS はコンフォーマンスクラスと呼ぶ詳細な機能セットを持つ(詳細は 2.3.3 節を参照). OSEK OS で定義されているシステムサービスは，すべてのスケーラビリティクラスから利用可能である【NOS0319】.

ATK2 では，表 2-3 に示す OS オブジェクト数および OSAP 数で実装する.

表 2-3 ATK2 における OS オブジェクト数および OSAP 数

機能	SC1, SC2	SC3, SC4
スケジュールテーブル数	ターゲット定義(※1) 【IOS089】	
ソフトウェアカウンタ数	ターゲット定義(※1) 【IOS090】	
OSAP 数	－	信頼 OSAP : ターゲット定義(※1) 【IOS091】 非信頼 OSAP : 32【IOS092】

※1　unsigned int の上限値

2.3.2　機能レベル

本 OS は，各スケーラビリティクラスの持つ機能を段階的に選択可能とするための機能レベルを提供する【NOS0117】〔NOS0273〕〔NOS0274〕〔NOS0275〕.

本 OS の機能と機能レベルを 表 2-4 に示す.

表 2-4 本 OS の機能と機能レベル数

機能	機能レベル数
保護違反時処理	3
タイミング保護	3
メモリ保護	3

AUTOSAR 仕様との違い

本仕様では，各スケーラビリティクラスに対して機能レベルという概念を導入した.

2.3.2.1　保護違反時処理の機能レベル

保護違反時処理の機能レベルを表 2-5 に示す(詳細は 2.22 節を参照)【NOS0263】. 表 2-5 には，保護違反時処理として指定するプロテクションフックの返り値を示す.

表 2-5 保護違反時処理の機能レベル

機能		Lvl.1	Lvl.2	Lvl.3
保護違反時処理	何もしない(PRO_IGNORE)	○	○	○
	OS シャットダウン(PRO_SHUTDOWN)	○	○	○
	違反したタスクの強制終了(PRO_TERMINATETASKISR)		○	○
	違反した C2ISR の強制終了(PRO_TERMINATETASKISR)			○
	違反した処理単位が所属する非信頼 OSAP の強制終了 (PRO_TERMINATEAPPL)		○	○
	違反した処理単位が所属する非信頼 OSAP の強制終了した後，リスタートタスクを起動する (PRO_TERMINATEAPPL_RESTART).		○	○
	違反した処理単位が所属する信頼 OSAP の強制終了 (PRO_TERMINATEAPPL)			○
	違反した処理単位が所属する信頼 OSAP の強制終了した後，リスタートタスクを起動する (PRO_TERMINATEAPPL_RESTART).			○

2.3.2.2　タイミング保護の機能レベル

タイミング保護の機能レベルを表 2-6 に示す(詳細は 2.17 節を参照)【NOS0209】.

表 2-6 タイミング保護の機能レベル

機能			Lvl.1	Lvl.2	Lvl.3
タイミング保護	タスクに対する機能	実行時間の監視	○	○	○
		起動および待ち解除時間の監視			○
		リソース占有時間の監視		※1	○
		OS 割込み禁止時間の監視		○	○
		全割込み禁止時間の監視			○
	C2ISR に対する機能	実行時間の監視			○
		割込み発生間隔の監視		○	○
		リソース占有時間の監視		※1	○
		OS 割込み禁止時間の監視			○
		全割込み禁止時間の監視			○

※1：リソース毎に監視時間を定義する(詳細は 2.17.2.3 節を参照)

2.3.2.3　メモリ保護の機能レベル

メモリ保護機能の機能レベルを表 2-7 に示す(詳細は 2.19 節を参照)【NOS0264】.

表 2-7 メモリ保護機能の機能レベル

機能		Lvl.1	Lvl.2	Lvl.3
OSAP	信頼関数	○	○	○
	非信頼 OSAP に所属するタスクのサポート	○	○	○
	非信頼 OSAP に所属するフックルーチンのサポート		○	○
	非信頼 OSAP に所属する C2ISR のサポート			○
メモリ保護	書込みアクセス保護	○	○	○
	読出し(実行)アクセス保護		○	○
	周辺デバイスに対するアクセス保護	※1	※1	○

※1：周辺デバイスに対する保護領域のサポートは必須ではない

2.3.2.4　機能レベルの指定

機能レベルは，コンフィギュレーション時に静的に指定する【NOS0234】〔NOS0273〕〔NOS0274〕〔NOS0275〕．また，システム動作中に機能レベルを変更することはできない【NOS0235】．ATK2 では，すべての機能を，機能レベル 2 で実装する【IOS098】.

2.3.3　コンフォーマンスクラス

コンフォーマンスクラスは，様々なアプリケーションからの要求やハードウェアの制約からの要求に対応するため機能セットである．

コンフォーマンスクラスにより異なる OS 機能を以下に示す．

- タスクに対する多重起動要求の許可，禁止【COS0314】
- 使用できるタスクの種別(詳細は 2.5.2 節を参照)【COS0315】
- タスク優先度ごとのタスク数【COS0316】

2.3.3.1　コンフォーマンスクラスの種別

コンフォーマンスクラスは以下の 4 種類である．それぞれのコンフォーマンスクラスが持つ最小の機能の組み合わせを表 2-8 に示す．

表 2-8 各コンフォーマンスクラスでサポートすべき最小機能セット

	コンフォーマンスクラス			
	BCC1 【COS0317】	BCC2 【COS0318】	ECC1 【COS0319】	ECC2 【COS0320】
基本タスクの多重起動要求	不要	要	不要	要
使用できるタスクの種別	基本タスク	基本タスク	基本タスク および 拡張タスク	基本タスク および 拡張タスク
同時に実行可能状態，実行状態になることができるタスクの最大数	8	8	16	16
1 優先度への複数タスクの割当て	不要	要	不要	要
1 タスクごとのイベント数	－	－	8	8
タスク優先度数	8	8	16	16
使用できるリソース数	1 (※1)	8	8	8
使用できる内部リソース数	2	2	2	2
使用できるアラーム数	1	1	1	1
使用できるアプリケーションモード	1	1	1	1

※1：すべてのタスクに関連付いたリソースを 1 つ用意する(詳細は 2.9.6 節を参照)

ATK2 では，表 2-9 に示す機能セットで実装する．

表 2-9 ATK2 における機能セット

	ATK2
タスクの多重起動要求	基本タスク：可【IOS015】 拡張タスク：不可【IOS016】
同時に実行可能状態，実行状態になることができるタスクの最大数	ターゲット定義(※1) 【IOS017】
1 優先度への複数タスクの割当て	可【IOS018】
1 タスクごとのイベント数	32≪IOS026≫
タスク優先度数	16≪IOS020≫
使用できるリソース数	ターゲット定義(※2) 【IOS021】
使用できる内部リソース数	ターゲット定義(※2) 【IOS022】
使用できるアラーム数	ターゲット定義(※1) 【IOS023】
使用できるアプリケーションモード	32【IOS024】

※1　unsigned int の上限値

※2　リソース数と内部リソース数の合計値が unsigned int の上限値

さらに ATK2 では，表 2-10 に示す機能制約がある．

表 2-10 ATK2 における機能制約

	ATK2
タスクの最大起動要求回数	256≪IOS025≫
SuspendAllInterrupts 最大ネスト回数	255【IOS062】 ※全割込み禁止スピンロック獲得との ネスト回数との合計値とする【IOS202】
SuspendOSInterrupts 最大ネスト回数	255【IOS063】 ※OS 割込み禁止スピンロック獲得との ネスト回数との合計値とする【IOS203】
スケジュールテーブルの最大満了点数	254【IOS093】
メモリセクション情報(OsMemorySection)の上限数	ターゲット定義【IOS160】
アクセスを許可するメモリ領域(OsMemoryArea)の上限数	ターゲット定義【IOS161】

2.3.3.2　コンフォーマンスクラス間の互換性

低機能なコンフォーマンスクラスで動作していたアプリケーションは，アプリケーションを変更することなく，高機能なコンフォーマンスクラスで動作させることができる【COS0312】．OS の動作中にコンフォーマンスクラスを動的に変更することはできない【COS0313】．コンフォーマンスクラス間の互換性を図 2-4 に示す．

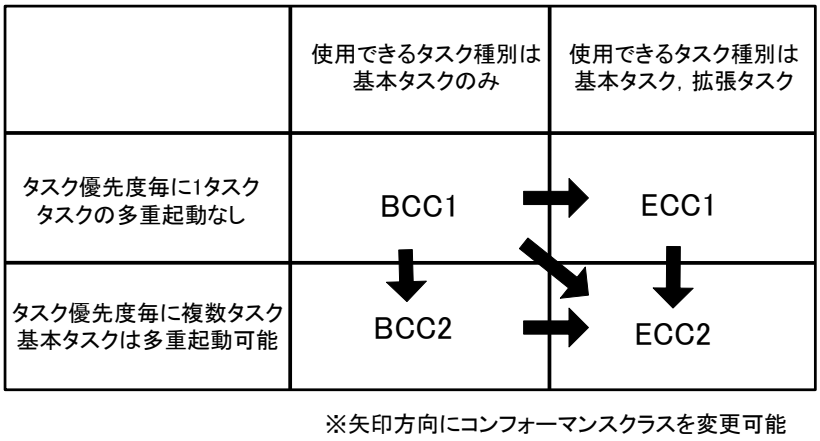


図 2-4 コンフォーマンスクラス間の互換性

2.4　マルチコア対応 OS

マルチコアシステム上での動作に対応した OS を，マルチコア対応 OS と呼ぶ．マルチコア対応 OS は，シングルコア OS に対して，以下の機能を追加したものである．本節では，マルチコア対応 OS の仕様を規定する．

- コアを跨いだシステムサービス
- OS の起動，終了のマルチコア対応
- スピンロック
- コア間割込み
- ミューテックス(参考仕様)

2.4.1　対象とするマルチコアシステムのハードウェア

マルチコアシステムは，複数のコアにより構成されているシステムであり，複数の処理単位を並行して同時に実行することが可能である【OS568】．本 OS が対象とするマルチコアシステムは，システムを構成する各コアはハードウェア的に均質(ホモジニアス)であり，少なくとも命令セット，エンディアンは同一であるものとする．詳細は「次世代車載システム向け RTOS ハードウェア要求仕様書」を参照．

2.4.2　OS オブジェクト

コアへの OS オブジェクトの割付け

マルチコア対応 OS では，SC1，SC2 であっても，OSAP のサポートが必須である【OS764】．リソース，IOC，スピンロックを除く OS オブジェクトは，いずれかの OSAP に所属しているため，OSAP を割付けるコアを指定することで，OSAP に所属する各 OS オブジェクトが割付けられるコアが決定する．タスクは，タスクが所属する OSAP が割付けられたコアで実行される【OS570】．同様に ISR は，ISR が所属する OSAP が割付けられたコアで実行される【OS571】．

OSAP へのコアの割付けは，コンフィギュレーション時に静的に指定する【OS573】〔MCOS_Conf1020〕．OS は，割込み発生時，ISR が起動するコアが静的に決定されるように，ハードウェアの初期化を行う【OS572】．

リソースは，OSAP に所属しないため，そのリソースを獲得するように指定されたタスク，C2ISR が割付けられるコアに割付く【NOS0914】．リソースは，同じコア内のタスク，C2ISR からのみ獲得可能であるので，リソースが割付くコアは一意に定まる．

スピンロックはコア間での排他制御に用いるため，OSAP にも所属せず，どのコアにも割付けられない【NOS0915】．IOC は，OSAP 間の通信であるので OSAP に所属せず，コアを跨いだ通信が可能であるので，どのコアにも割付けられない【NOS0916】．

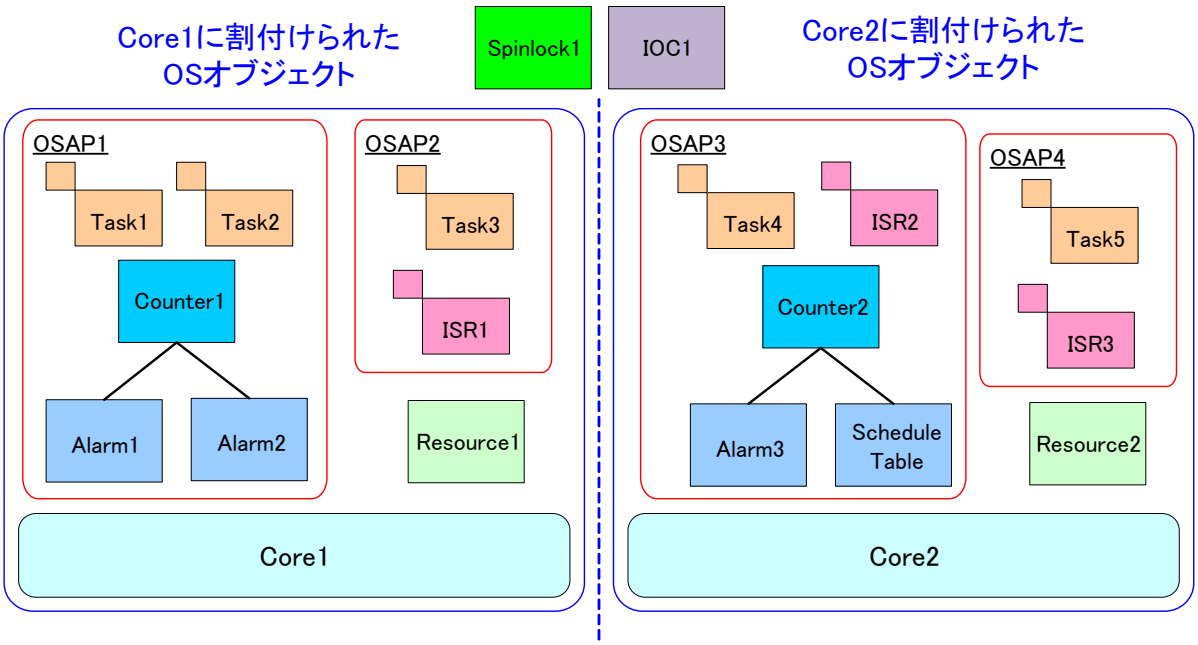


図 2-5 OS オブジェクトのコア割付け

OS は，各コアのタスクや ISR のスケジューリングを独立して実行する【OS569】．OS は，タスクや ISR が割付いているコアを動的に変更することはできない【OSa117】．

このように，タスクが動作するコアを静的に決定しておく方式は，機能分散型マルチプロセッサ(Function Distributed Multiprocessor: FDMP)，もしくは非対称型マルチプロセッサ(Asymmetric

Multiprocessor: AMP)と呼ばれる．対称型マルチプロセッサ(Symmetric Multiprocessor: SMP)と呼ばれる方式で行われるような, 負荷分散を目的とした, OS による動的なタスクの移動はサポートしない.

OS オブジェクトの ID

マルチコア対応 OS においても，シングルコア OS におけるオブジェクト ID の規定を維持するものとし，すべてのコアにおいて，オブジェクト ID は一意とする【OS567】．OS オブジェクトのコアへの割付けは，コンフィギュレーション時に決定されるため，C 言語で記述されたソースコード上の OS オブジェクト ID はコア割付けの影響を受けることはない.

2.4.3　コア ID

マルチコアシステムにおいてコアを一意に識別するために，各コアに ID を割り振り，これをコア ID と呼ぶ．ハードウェアレベルのコアを識別する方法に関係なく，本 OS におけるコア ID は，論理的に識別できるようにする【OSa118】〔OS627〕〔OS628〕．つまり，各コア ID を示す定数を用意し，アプリケーションからは，これらの定数を用いることで，コアを判別する(詳細は 3.6.2 節を参照).

2.4.4　シングルコア OS 機能のマルチコア拡張

2.4.4.1　コアを跨ぐシステムサービス

システムサービスを呼び出した処理単位が割付けられたコアと，システムサービスのパラメータに指定された OS オブジェクトが割付けられたコアが異なる場合，システムサービスが「コアを跨ぐ」と表現する．

シングルコア OS で提供されるシステムサービスが，マルチコア OS においてどのように振る舞うかは，システムサービス毎に規定する．コアを跨いで呼び出すことができないシステムサービスを，コアを跨いで呼び出した場合，OS は，拡張エラーとして E_OS_CORE を返す【OS589】.

他のコアに対して，ディスパッチを伴うシステムサービスを呼び出した場合，呼出し先のコアに対して，割込み(コア間割込み)を発生させることで，ディスパッチを実行する【NOS0917】．したがって，割込み禁止状態となっているコアに対して，ディスパッチを伴うシステムサービスを発行した場合，その割込み禁止状態が解除されるまで，ディスパッチが保留される【NOS1068】.

なお，マルチコア対応 OS においては，OS 処理が実行されているコア以外のコアから，OS 処理の途中の状態が観測できる場合がある【NOS1056】．具体的には，1 つのシステムサービスにより複数の OS オブジェクトの状態が変化する場合に，一部の OS オブジェクトの状態のみが変化し，残りの OS オブジェクトの状態が変化していない過渡的な状態が観測できる場合がある．

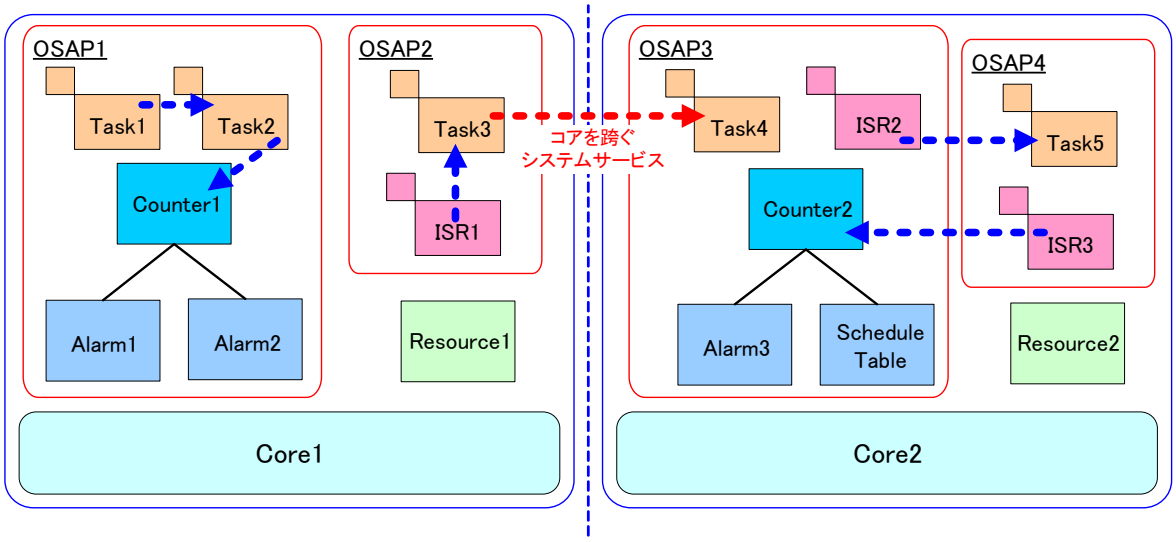


図 2-6 コアを跨ぐシステムサービス

2.4.4.2　OS の起動と終了

マルチコアシステムにおけるすべてのコアで，OS の起動および終了シーケンスにおけるいくつかのフェーズで，同期を行う(詳細は 2.24.4 節を参照).

すべてのコアで OS シャットダウンを行うシステムサービスを呼び出した場合，他のすべてのコアに対して，割込み(コア間割込み)を発生させることで，OS シャットダウンを実行する【NOS0918】.

2.4.5　マルチコア追加機能

マルチコア対応 OS では，コアを跨いだアプリケーション間の連携動作に役立つ以下の機能を追加する. マルチコア対応 OS で追加されたシステムサービスは, すべての SC から利用可能である【NOS0339】.

2.4.5.1　コア ID の取得

本 OS は，任意の処理単位からその処理単位が動作しているコアの ID を取得するシステムサービスとして GetCoreID を提供する(詳細は 3.9.63 節を参照).

2.4.5.2　起動中のコア数の取得

マルチコアシステムのおいて，本 OS が起動しているコアの数を取得するシステムサービスとして GetNumberOfActivatedCores を提供する(詳細は 3.9.62 節を参照).

2.4.5.3　スピンロック

本 OS は，異なるコアに割付けられているタスク，C2ISR 間の排他制御機能として，スピンロックを提供する(詳細は 2.26 節を参照)【OS648】〔OS686〕〔OS695〕〔OS703〕.

2.4.5.4　コア間割込み処理

本 OS は，あるコアに割付けられている処理単位から，任意の他のコアに対して割込みを発生させるコア間割込み機能を提供する．割込み先のコアではユーザ定義の処理単位(コア間 ISR)を起動する(詳細は 2.27 節を参照).

2.4.5.5　ミューテックス(参考仕様)

本 OS は，異なるコアに割付けられているタスク間で共有資源の排他制御機能を行うミューテックス機能を提供する．スピンロックでは，タスクがスピンしてスピンロック解放を待つのに対し，ミューテックスでは，タスクがブロッキング(待ち状態へ移行)してミューテックス解放を待つ(詳細は 2.28 節を参照).

2.4.6　想定する実装方式

本節では，マルチコア対応 OS を効率的に実装するために，想定する実装方式を規定する．なお，これらの実装方式は OS の仕様には影響を与えないが，ハードウェアに対する要件に影響を与えるため，本文書において規定する．この実装ができない場合，マニュアル等に明記する．

OS 実装方式

マルチコア対応 OS の実装方式として，1 リンクモデルを採用する【NOS0017】．これは本仕様にて規定されるすべての機能が，同一のメモリ空間で動作するように実装されることを意味する．

システムサービス実装方式

コアを跨いで実行するシステムサービスの実装は，直接操作法と呼ばれる方式を採用する【NOS0018】．これは他のコアに割付けられている OS オブジェクトに対するシステムサービスを実装する際に，他のコアの OS 内部データを直接操作して実現する方式である．

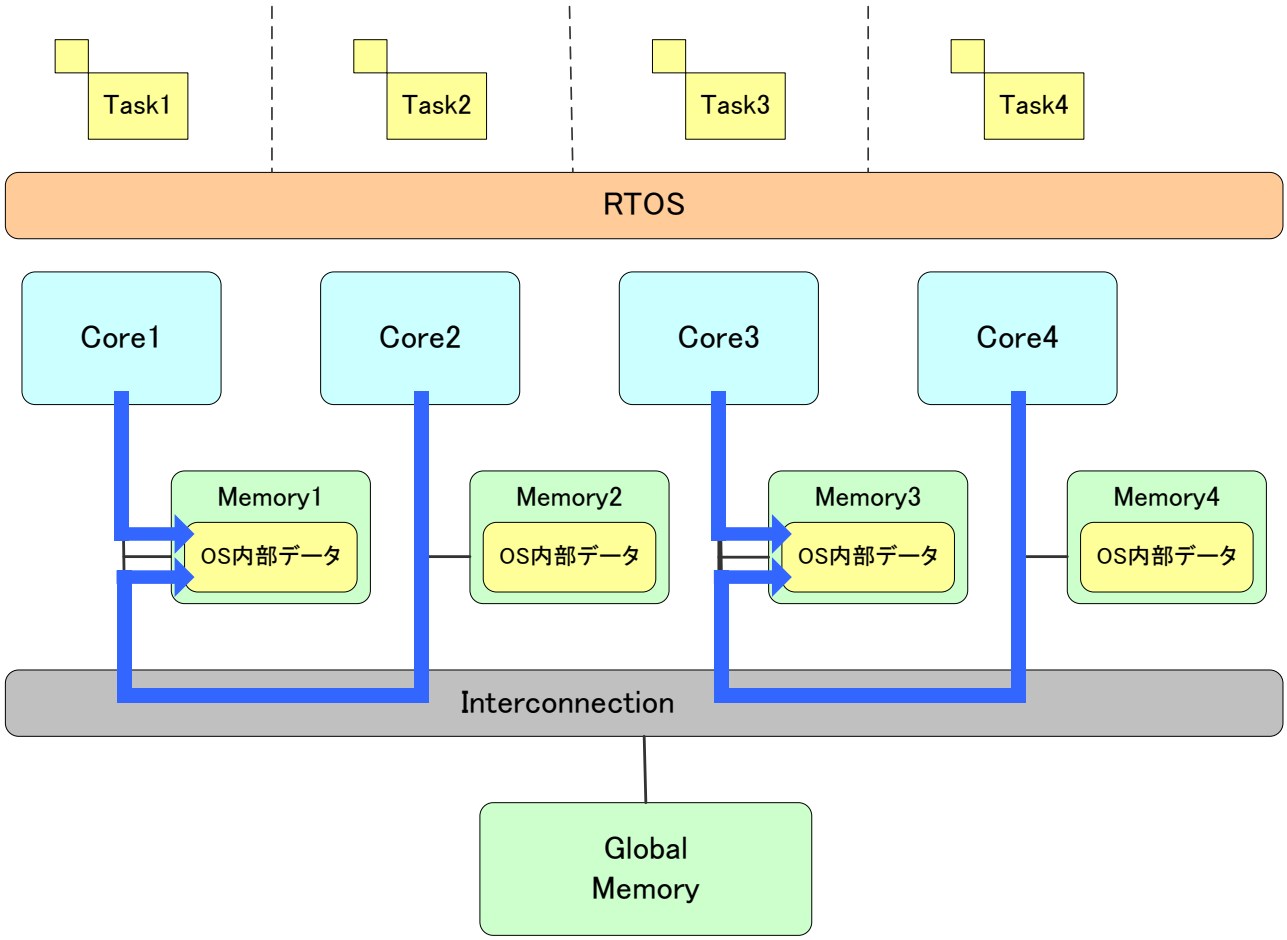


図 2-7 直接操作法によるコア間システムサービス実装

OS 実行コードの動作形態

マルチコアシステムを構成する各コアにおいて，同じ機能を持つ OS が動作する【NOS0019】．また，各コアで独立した OS の実行コードを持つか，すべてのコアで同一の実行コードを共有するかは実装定義とする【NOS0020】．ATK2 では，すべてのコアで同一の実行コードを共有する【IOS094】．OS 実行コードの持ち方によるメリットとデメリットを以下に示す．

表 2-11 OS 実行コードの持ち方によるメリットとデメリット

OS 実行コード	メリット	デメリット
各コアで独立した実行コードを持つ	コア毎に実行するコードが静的に決まっているため，コア番号をランタイムで取得する必要がなく，処理性能への影響が少ない．	コア数に比例して，OS の実行コードが増加するため，コア数が 2 倍になると OS のメモリ使用量も 2 倍となる．
すべてのコアで同一の実行コードを共有する	OS の実行コード 1 本であるため，コア数が変化しても，OS のメモリ使用量への影響が少ない．	動作中のコア番号をランタイムで取得する処理が必要なため，処理性能が低下する可能性がある．

2.5　タスク

タスクは OS によって実行されるプログラムの処理単位である．複数のタスクが存在する場合，タスクはそれぞれ並列に実行され，実行順序は OS によってスケジューリングされる．本 OS では，タスクを制御するための機能を提供する【COS0401】．タスクは，タスク ID によって識別する．

2.5.1　タスク状態

本 OS では，同時に実行状態になることができるタスクは 1 つのみ(マルチコア対応 OS の場合はコアごとに 1 つ)であり，他の状態には複数のタスクがなることができる【COS0416】.

本 OS で，タスクが遷移する状態を以下に示す.

実行状態(RUNNING)

タスクがコアに割付けられ，タスクの処理を実行している状態.

実行可能状態(READY)

タスクを実行できる条件はすべて満たされているが，より高い優先度のタスクが実行中であるため，コアに割付けられておらず，タスクの処理を実行できない状態．スケジューラは，実行可能状態(実行状態も含まれる)のタスクの中から次に実行状態となるタスクを決定する.

待ち状態(WAITING)

1 つ以上のイベントを待っており，タスクの処理を継続できない状態(詳細は 2.8 節を参照).

休止状態(SUSPENDED)

タスクが起動されておらず，タスクの処理が行われない状態.

2.5.2　タスクの種別

本 OS では，基本タスクと拡張タスクを定義する．

基本タスク

タスクはシステムサービスによって起動され，実行可能状態となる．さらに，他に高い優先度のタスクがなければコアに割付けられ，実行状態となる．基本タスクは以下の条件でコアを解放する．なお，基本タスクは待ち状態に入ることはできない≪COS3549≫.

- システムサービスによってタスク処理を終了した場合，コアを解放し休止状態となる【COS0408】.
- タスクの処理を実行中に，より高い優先度のタスクが起動した場合，コアを解放し実行可能状態となる【COS0409】.
- タスクの処理を実行中に割込みが発生した場合，タスクの状態は変更せずにコアを解放し，ISR の処理を実行する【COS0410】.

拡張タスク

拡張タスクは基本タスクの機能に加え，待ち状態に入ることができる【COS0412】．拡張タスクはシステムサービスによって待ち状態に入りコアを解放することで，より低い優先度のタスクをコアに割付けることができる．なお，本 OS では，休止状態から直接待ち状態となることはできない.

基本タスクと拡張タスクの比較

基本タスクは待ち状態に入ることがないため，拡張タスクに比べてタスク管理のためのメモリ領域として使用する RAM 容量を少なく実装することができる．一方，拡張タスクは待ち状態に入ることにより，1 つのタスクで処理を一貫して実行することができ，実行条件が整わない場合には待ち状態に入ることでデータやイベント待ちの同期を取ることができる.

また，コンフォーマンスクラスによっては，1 つの基本タスクに対して複数回の起動要求を行うことができるが，拡張タスクへはコンフォーマンスクラスによらず，複数回の起動要求を行うことはできない≪COS0314≫≪COS0483≫

2.5.3　タスクの状態遷移

2.5.3.1　基本タスクのタスク状態遷移

基本タスクの状態遷移を図 2-8 に示す.

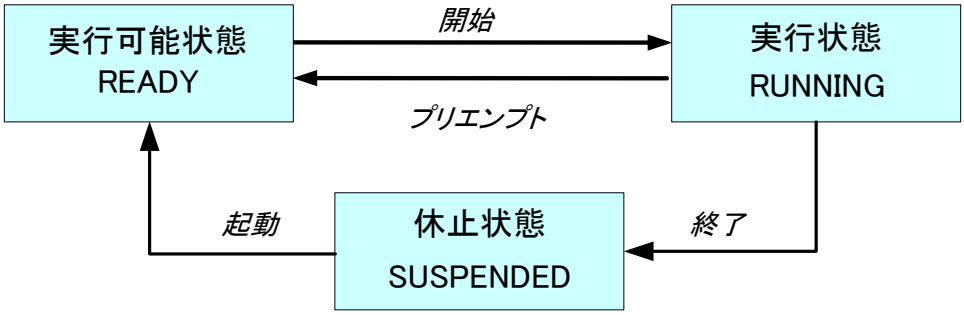


図 2-8 基本タスクのタスク状態遷移図

起動

システムサービスによってタスクが休止状態から実行可能状態となる状態遷移

開始

スケジューラによってタスクが実行可能状態から実行状態となる状態遷移

プリエンプト

スケジューラによってタスクが実行状態から実行可能状態となる状態遷移

終了

システムサービスによってタスクが実行状態から休止状態となる状態遷移

2.5.3.2　拡張タスクのタスク状態遷移

拡張タスクの状態遷移を図 2-9 に示す.

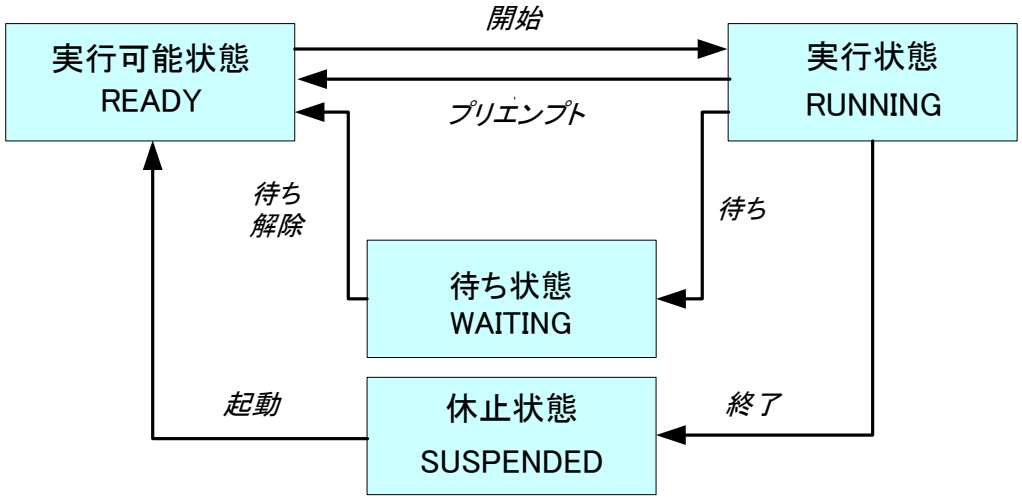


図 2-9 拡張タスクのタスク状態遷移図

起動

システムサービスによってタスクが休止状態から実行可能状態となる状態遷移

開始

スケジューラによってタスクが実行可能状態から実行状態となる状態遷移

プリエンプト

スケジューラによってタスクが実行状態から実行可能状態となる状態遷移

終了

システムサービスによってタスクが実行状態から休止状態となる状態遷移

待ち

システムサービスによって実行状態から待ち状態となる状態遷移

待ち解除

システムサービスによって待ち状態から実行可能状態となる状態遷移

2.5.4　タスクの操作

2.5.4.1　タスクの起動

本 OS は，タスクを起動するシステムサービスとして ActivateTask，ChainTask を提供する(詳細は 3.9.4 節，3.9.6 節を参照)．ActivateTask または ChainTask によってタスクが起動されると，タスクのプログラムコードの先頭から実行を開始できる状態となる【COS0432】．なお，本 OS ではタスクの起動時にパラメータを渡すことはできない【COS0482】.

タスクの多重起動

コンフォーマンスクラスによっては，1 つの基本タスクに対して複数回の起動要求を行うことができる【COS0483】．起動要求時に最大起動要求回数に達していない場合，起動要求回数がタスクごとにインクリメントされ，多重に起動要求されているタスクが終了した後，再度起動される【NOS0364】.

多重に起動要求されているタスクが不正終了，強制終了した場合も，再度起動される【NOS0380】．多重に起動要求されているタスクが所属する OSAP が強制終了した場合は，再度起動されず，起動要求はクリアされる【NOS0381】.

最大起動要求回数は，コンフィギュレーション時に静的に指定する【COS0434】.

OSEK 仕様との違い

OSEK 仕様では, 起動要求時に最大起動要求回数に達していない場合, 起動要求はタスク優先度ごと, 要求順にキューイングされると規定されている【COS0435】．しかし，タスクの起動順序は優先度によって確定できるため，起動要求の要求順を保持する必要性が不明確である．また，要求順を保持するには実装が複雑となり，メモリを消費することから，本仕様では要求順を保持する仕様を削除した≪NOS0364≫.

2.5.4.2　タスクの終了

本 OS は，タスク自身を終了するシステムサービスとして TerminateTask を提供する(詳細は 3.9.5 節を参照)．タスクはタスク自身からのみ終了することができる．また，タスクの終了と同時に他のタスクを起動するシステムサービスとして ChainTask を提供する(詳細は 3.9.6 節を参照).

タスクは，タスクのプログラムコード末で，TerminateTask もしくは ChainTask を呼び出さなければならない. TerminateTask, ChainTask を呼び出すことで, OS はタスクを正常に終了する【COS1427】. TerminateTask，ChainTask を呼び出さずに終了した場合，タスクの不正終了となる(詳細は 2.20.1.1 節を参照).

SC3，SC4 においては，保護違反時処理によりタスクが強制終了されることがある(詳細は 2.22.4.1 節を参照)．また，タスクが所属する OSAP が強制終了されることで，タスクが強制終了されることがある(詳細は 2.22.4.3 節を参照).

2.5.4.3　実行中のタスク ID の取得

本 OS は，現在実行中のタスクの ID を取得するシステムサービスとして GetTaskID を提供する(詳

細は 3.9.8 節を参照).

2.5.4.4　タスク状態の取得

本 OS は，タスクの状態を取得するシステムサービスとして GetTaskState を提供する(詳細は 3.9.9 節を参照).

2.5.4.5　タスクのイベント待ち

拡張タスクでは，イベント管理機能における WaitEvent というシステムサービスによって待ち状態に入ることができる(詳細は 2.8 節を参照).

2.5.5　タスクの実装方法

タスクの本体記述

OS はタスク ID とタスクのエントリアドレスの対応情報を保持し, エントリアドレスを使用してタスクを呼び出す【COS3293】.

タスクの本体記述を以下に示す【COS3294】〔COS0482〕.

```
TASK(<タスク ID>)
{
}
```

タスク ID の生成方法

タスク ID はコンフィギュレーション時にジェネレータによって割り当てる【COS3295】.

タイミング保護機能の適用

SC2，SC4 ではタスクにタイミング保護機能が適用される(詳細は 2.17.2 節を参照)．タスクに対する時間制約はコンフィギュレーション時に静的に指定する.

2.5.6　タスク優先度

タスク優先度とは，実行可能状態のタスクの中から次に実行状態となるタスクを決定するために，タスクに割当てられる優先順位である．タスクは，タスク優先度に従ってコアに割付けられ，終了時にコアを解放する．

2.5.6.1　タスク優先度の範囲と優先順位

タスク優先度の最小値は 0 である【COS0441】．タスク優先度は値が大きいほど高い優先順位を持つ【COS0442】.

タスク優先度の指定

本 OS では，タスク優先度はコンフィギュレーション時に静的に割当てられ，システムサービスによる動的なタスク優先度の変更はサポートしない【COS0443】．コンフィギュレーション時に割当てられたタスク優先度を初期優先度と呼び，タスク実行中のタスク優先度を現在優先度と呼ぶ.

ただし，リソース管理機能における GetResource を使用した場合，OS によって一時的に優先度が引き上げられる(詳細は 2.9.2 節を参照)．現在優先度は，リソース獲得，解放によって変化する．

タスク状態遷移時のタスクの優先順位

プリエンプトされたタスクは同一優先度を持つ実行可能状態のタスクの中で最も高い優先順位となる【COS0446】．また，待ち解除されたタスクは同一優先度を持つ実行可能状態のタスクの中で最も低い優先順位となる【COS0447】．同様に，起動されたタスクは，同一優先度を持つ実行可能状態のタスクの中で最も低い優先順位となる≪COS0450≫.

2.5.7　スケジューラ

実行状態または実行可能状態であるタスクの中から，次に実行状態とすべきタスクを決定する処理をスケジューリングと呼び，スケジューリングを行う OS 機能をスケジューラと呼ぶ.

スケジューラは，実行状態のタスクと実行可能状態のタスクの中から，最も優先度の高いタスクを検索し，そのタスクを実行状態にする【COS0448】．同じ優先度のタスクが複数存在する場合，最初に実行可能状態もしくは実行状態に遷移したタスクを，次の実行タスクとする【COS0450】.

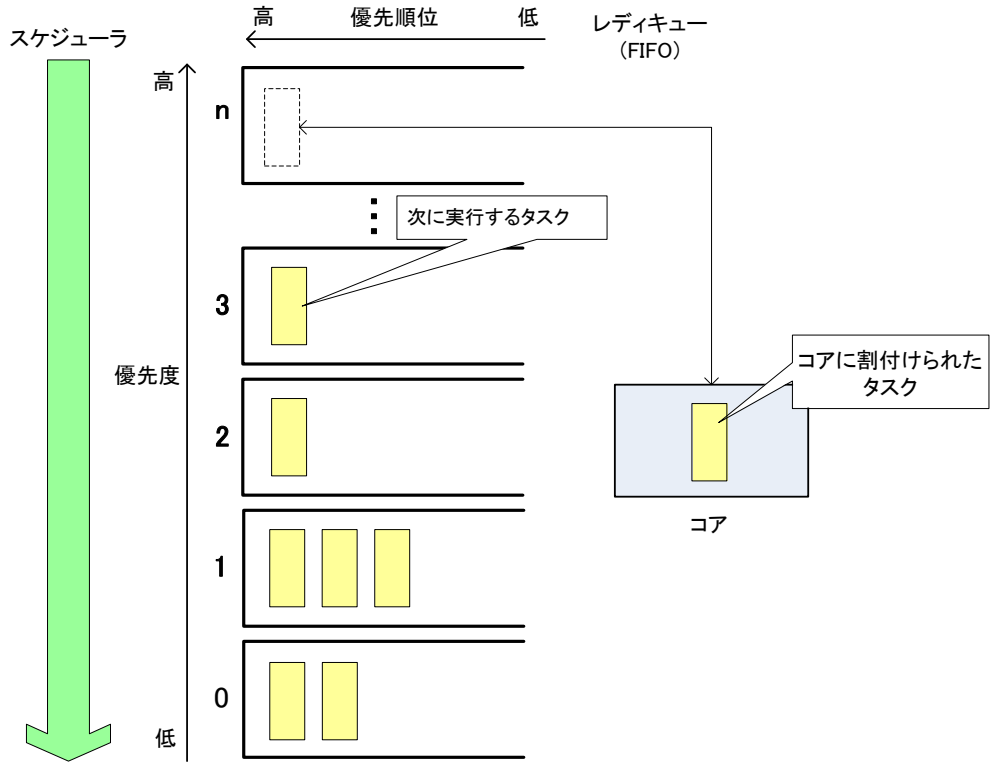


図 2-10 スケジューラによる実行タスクの検索

2.5.7.1　スケジューラの操作

本 OS は，明示的なスケジューリングを要求するシステムサービスとして Schedule を提供する(詳細は 3.9.7 節を参照).

2.5.7.2　タスクディスパッチ機構

本 OS は，スケジューリングにより実行中のタスクを中断して，別のタスクに切り換えるプリエンプト機能を持つ【COS0413】．中断したタスクの処理を，中断した箇所から再開するため，OS はタスク切換え時にタスクコンテキストの保存と復帰を行う【COS0414】.

2.5.8　スケジューリングポリシ

本 OS では，3 種類のスケジューリングポリシをサポートする．スケジューラは，スケジューリングポリシで規定されたタイミングで起動され，タスクの再スケジューリングを行う【COS0438】.

スケジューラ起動時に，DisableAllInterrupts による割込み禁止状態であった場合，OS は割込み許可状態に戻す【NOS0648】．例えば，タスク終了時に DisableAllInterrupts による割込み禁止状態であった場合，次に起動するタスクが実行状態となった時点で，割込み許可状態となっている．

2.5.8.1　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，タスクディスパッチを伴うシステムサービスの他に，タスク実行中の任意のタイミングに，OS によって再スケジューリングが行われ実行中タスクがプリエンプトされる可能性がある【COS0451】. フルプリエンプティブスケジューリングで動作するタスクを，プリエンプティブタスクと呼ぶ.

フルプリエンプティブスケジューリングを使用する場合，タスクがユーザの意図しないタイミングでプリエンプトされる可能性がある．そのため，共有資源へのアクセスなど，処理の実行を不可分に行う場合，ユーザは OS が提供する機能を用いて排他制御を行う必要がある.

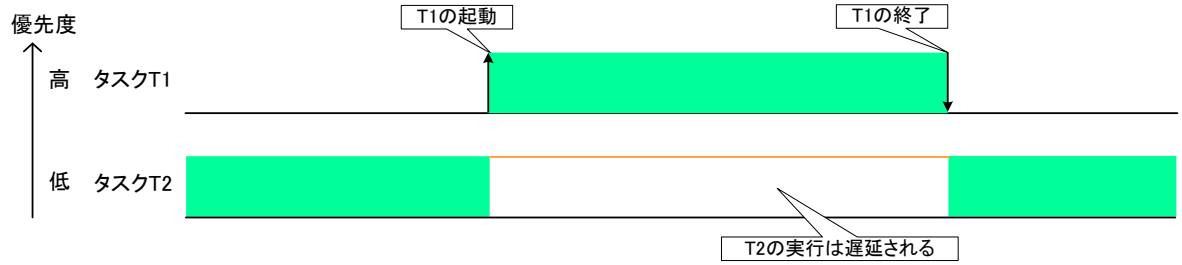


図 2-11 フルプリエンプティブスケジューリングの例

フルプリエンプティブスケジューリングの再スケジューリングタイミング

フルプリエンプティブスケジューリングで，再スケジューリングが行われる可能性があるのは以下のタイミングである．

- タスクの終了時（TerminateTask 発行時)≪COS3232≫
- タスク終了とタスク起動要求の実行時（ChainTask 発行時)≪COS3248≫
- タスク起動処理要求の実行時（ActivateTask 発行時)≪COS3216≫
- 待ち状態への遷移時（WaitEvent 発行時)≪COS3546≫
- 待ち状態のタスクに対するイベントのセット時（SetEvent 発行時)≪COS3553≫
- リソース解放時（ReleaseResource 発行時)≪COS3424≫
- C2ISR 処理レベルからタスク処理レベルへの復帰時【COS0460】
- タスクの不正終了時(詳細は 2.20.1.1 節を参照)
- タスクの強制終了時(詳細は 2.22.4.1 節を参照)
- タスクが所属する OSAP の強制終了時(詳細は 2.22.4.3 節を参照)

アプリケーション作成時の注意事項

フルプリエンプティブスケジューリングを使用する場合は，システムサービスによって明示的なスケジューリングタイミングを OS に通知する必要はない．ただし，ノンプリエンプティブスケジューリング向けに開発したアプリケーションを，フルプリエンプティブスケジューリングで使用する場合などにおいて，プリエンプティブタスクから Schedule を呼び出しても OS は何も処理をしないため，影響はない．

2.5.8.2　ノンプリエンプティブスケジューリング

ノンプリエンプティブスケジューリングは，実行状態のタスクがシステムサービスによって明示的にスケジューリングの要求を行わない限り，スケジューリングが発生しないスケジューリングである【COS0477】．ノンプリエンプティブスケジューリングで動作するタスクを，ノンプリエンプティブタスクと呼ぶ．

ノンプリエンプティブスケジューリングの再スケジューリングタイミング

ノンプリエンプティブスケジューリングで，再スケジューリングが行われる可能性があるのは以下のタイミングである．

- タスクの終了時（TerminateTask 発行時）【COS0462】
- タスク終了とタスク起動要求の実行時（ChainTask 発行時）【COS0463】
- 明示的なスケジューラ呼出し時（Schedule 発行時）≪COS3261≫
- 待ち状態への遷移時（WaitEvent 発行時）【COS0465】
- タスクの不正終了時(詳細は 2.20.1.1 節を参照)
- タスクの強制終了時(詳細は 2.22.4.1 節を参照)
- タスクが所属する OSAP の強制終了時(詳細は 2.22.4.3 節を参照)

アプリケーション作成時の注意事項

ノンプリエンプティブスケジューリングを使用する場合は，明示的なスケジューリング要求があるまでタスクがプリエンプトされないため，実行状態のタスクよりも高い優先度のタスクが実行可能状態になっても，ユーザが配置した次の再スケジューリングタイミングまで，高い優先度のタスクの実行が遅延される．

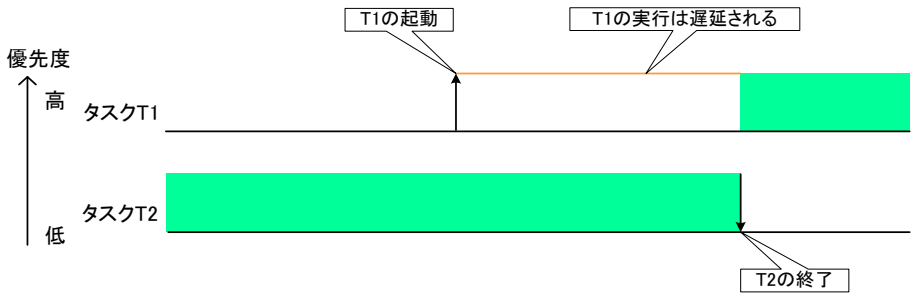


図 2-12 ノンプリエンプティブスケジューリングの例

2.5.8.3　混合プリエンプティブスケジューリング

プリエンプティブタスクと，ノンプリエンプティブタスクが混合している場合のスケジューリングを，混合プリエンプティブスケジューリングと呼ぶ.

OS は，各タスクのスケジューリングポリシに従ってスケジューリングを行う.

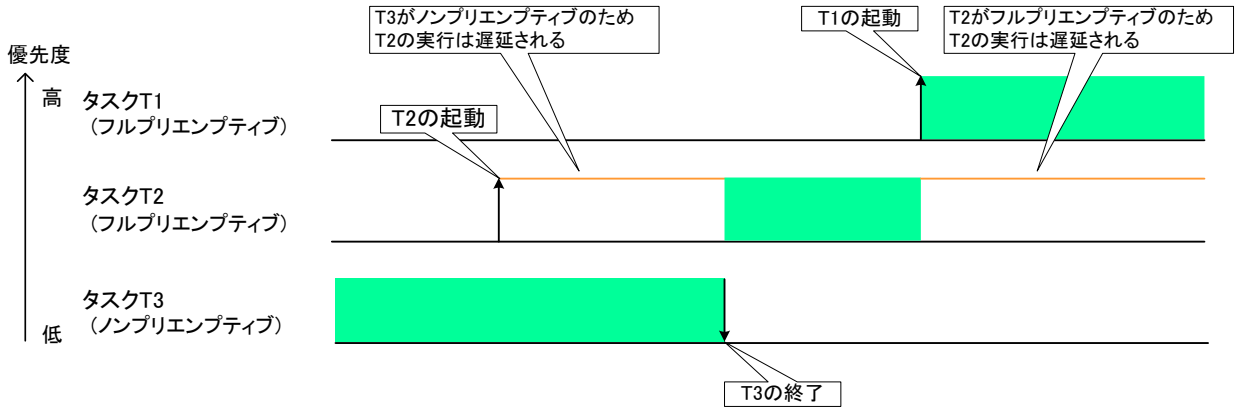


図 2-13 混合プリエンプティブスケジューリングの例

2.5.8.4　スケジューリングポリシの指定

スケジューリングポリシは，タスク毎にコンフィギュレーション時に静的に指定する【COS0470】. つまり，プリエンプティブタスクと，ノンプリエンプティブタスクが，混在した場合が，混合プリエンプティブスケジューリングとなる.

2.5.9　グループタスク

グループタスクは複数のタスクの関連付けを行い，関連付けられたタスク間ではノンプリエンプティブなスケジューリングを行う仕組みである．グループタスクは，内部リソース機能により実現する．

本 OS ではグループタスクの考え方を用いて，プリエンプティブスケジューリングとノンプリエンプティブスケジューリングを組み合わせてタスクをスケジューリングすることが可能である．

2.5.9.1　グループタスク内タスクのスケジューリングポリシ

グループタスクに含まれるタスクのスケジューリングは，通常のタスクと同じく，タスク優先度とスケジューリングポリシに沿って行われる．グループタスクに含まれるタスクが通常のタスクと異なる点は，スケジューラにより実行状態となった際に，グループタスク内のタスクはグループ内の他のタスクに対してノンプリエンプティブなタスクとして動作することである【COS0478】．一方，グループタスクに含まれるタスクの最高優先度より高いタスクに対してはプリエンプティブなタスクとして動作する【COS0479】．

2.5.9.2　内部リソースによるグループタスクの実現方法

図 2-14 の例では，タスク T2 とタスク T3 がグループタスクとして定義されている．タスク T4 がタスク T3 を起動するとタスク T3 は内部リソースを自動的に獲得し，内部リソースの優先度までタスク優先度が引き上げられる．そのため，タスク T2 を起動してもタスク T2 にはタスクディスパッチしない．一方，グループタスクに含まれず内部リソースより優先度の高いタスク T1 を起動するとタスクディスパッチが行われ，タスク T1 が実行される．

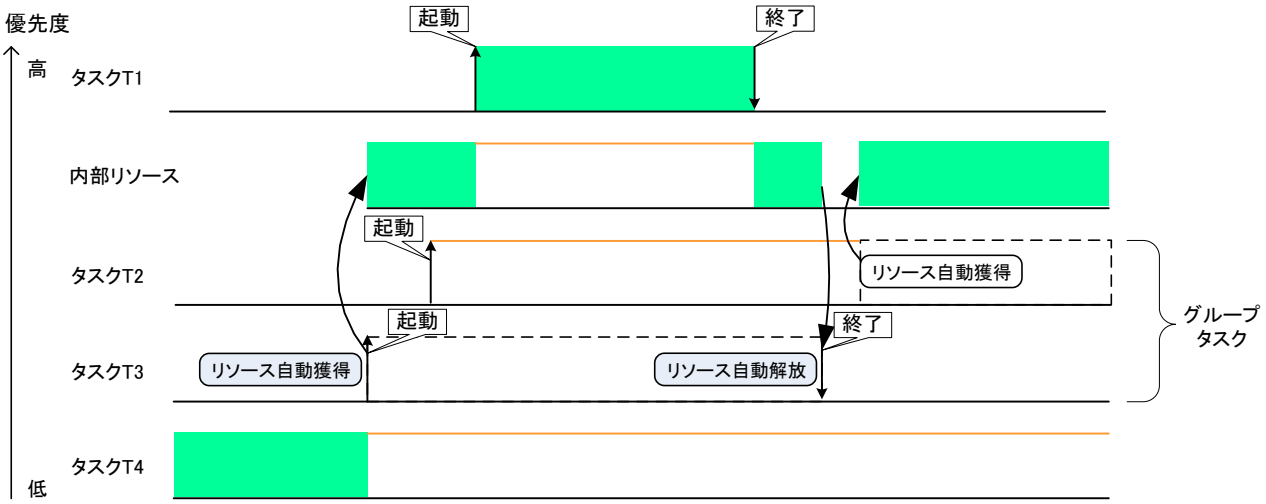


図 2-14 グループタスクの動作例

2.5.10 マルチコア対応 OS におけるスケジューリング

マルチコア対応 OS におけるスケジューリングは各々のコアで独立に行われる≪OS569≫．つまり，各々のコアが個別にスケジューラとタスクディスパッチ機構を持っており，前述のスケジューリングポリシが適用される単位は，同じコアに割付けられているタスクの集合となる．

割込み禁止状態となっているコアに対して，ディスパッチを伴うシステムサービスを発行した場合，その割込み禁止状態が解除されるまで，ディスパッチが保留される【NOS0919】.

2.5.11 スタック共有

基本タスクは，待ち状態となることがないため，1 つの基本タスクが実行状態となってから終了するまでに，他の同一タスク優先度の基本タスクが実行状態となることがない．したがって，同一タスク優先度の基本タスクが使用するスタックは，ジェネレータによって共有される【NOS0624】．同一タスク優先度の基本タスクに設定したスタックサイズが異なる場合は，最も大きいサイズでスタック領域が確保され，それぞれのタスクのスタックサイズは最も大きいサイズに拡張される【NOS0630】.

ATK2 では，タスク毎にスタックの先頭番地を指定可能であるので，タスクの種別やタスク優先度に関わらず，同一番地を指定することで複数のタスクのスタックを共有することができる≪IOS047≫≪IOS048≫.

2.6　アプリケーションモード

アプリケーションモードは，OS がアプリケーションプログラムに対して，異なる制御モードでの起動を提供するための機能である．

OS は 1 つ以上のアプリケーションモードをサポートしなければならない【COS0501】〔COS0507〕．アプリケーションモードは OS の動作開始以降，変更することはできない【COS0502】．アプリケーションモードはすべてのコンフォーマンスクラスで使用できる【COS0507】.

2.6.1　アプリケーションモードによる OS 起動処理の制御

アプリケーションモードを使用することで，ユーザは OS 起動時に自動起動する OS オブジェクトを変更することができる【COS0505】〔NOS0670〕．一方，アプリケーションモードは OS 終了時の処理には影響しない【COS0508】.

ユーザは，アプリケーションモードを，OS 起動前に決定する【COS0504】．アプリケーションモードは，StartOS 実行時の引数によって決定する≪COS3709≫.

2.6.2　自動起動が可能な OS オブジェクト

OS が自動起動を行うことができる OS オブジェクトを以下に示す.

タスクの自動起動

コンフィギュレーション時に指定されたタスクを, OS 起動時に自動起動する【COS0506】〔COS0505〕.

アラームの自動起動

コンフィギュレーション時に指定されたアラームを, OS 起動時に自動起動する【OS476】〔COS0505〕.

スケジュールテーブルの自動起動

コンフィギュレーション時に指定されたスケジュールテーブルを，OS 起動時に自動起動する【NOS0670】〔COS0505〕.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，自動起動するタスク，アラームを起動した後で，スケジュールテーブルを起動すると規定されている【OS510】．しかし，オブジェクトの起動順序がユーザから観測できないため，順序に関する規定を削除した.

2.6.3　指定されたアプリケーションモードの取得

本 OS は，OS 起動時に指定されたアプリケーションモードを取得するシステムサービスとして GetActiveApplicationMode を提供する(詳細は 3.9.1 節を参照).

2.6.4　自動起動オブジェクトの指定

タスク，アラーム，スケジュールテーブルが自動起動するアプリケーションモードを，コンフィギュレーション時に静的に指定する≪COS0506≫≪OS476≫≪NOS0670≫．1 つのオブジェクトを，複数のアプリケーションモードに指定することが可能である【NOS0370】.

2.6.5　マルチコア対応 OS におけるアプリケーションモード

マルチコアシステムでは，ユーザは，OS を起動するすべてのコアで StartOS を実行する【OS607】. アプリケーションモードを選択せず，他コアで指定されたアプリケーションモードに従う場合は，StartOS に DONOTCARE を指定する【OS609】. ユーザは，OS を起動する少なくとも 1 つのコアで，DONOTCARE 以外のアプリケーションモードを指定する必要がある【OS610】〔NOS0920〕. また，複数のコアにおいて，DONOTCARE 以外の異なるアプリケーションモードを指定して，StartOS が呼び出された場合，OS は起動せず，すべてのコアで無限ループする【OS608】. すべてのコアにおいて，DONOTCARE を指定して，StartOS が呼び出された場合も，OS は起動せず，すべてのコアで無限ループする【NOS0920】. つまり，マルチコアシステムを構成するすべてのコアで，共通のアプリケーションモードが適用されるため，コアごとに個別のアプリケーションモードを指定して OS を動作させることはできない.

本 OS は，各コアにおいて，StartOS で指定された共通のアプリケーションモードで自動起動するように，コンフィギュレーション時に指定されたタスクを，最初のスケジューリングの前に起動する【OS668】. 本 OS は，各コアにおいて，StartOS で指定された共通のアプリケーションモードで自動起動するように，コンフィギュレーション時に指定されたアラームを，最初のスケジューリングの前に起動する【OS669】. 本 OS は，各コアにおいて，StartOS で指定された共通のアプリケーションモードで自動起動するように，コンフィギュレーション時に指定されたスケジュールテーブルを，最初のスケジューリングの前に起動する【OS670】.

2.7　割込み処理

割込み処理は割込みコントローラからの割込み要求をトリガとして起動される処理である．割込み処理は，スケジューリングポリシに関係なく，タスクの処理を中断することができる【COS0610】．割込み処理の処理レベルはタスクより高いため，割込み処理中にタスクは動作しない【COS1429】.

ISR は ISR ID によって識別する.

2.7.1　ISR の種別

ISR には，C1ISR と C2ISR の 2 種類がある【COS0601】.

なお，ISR 中では使用できるシステムサービスが制限される(詳細は表 3-1 を参照).

C1ISR

C1ISR は OS が管理しない ISR である．C1ISR 実行に必要な割込み出入り口処理は実装定義である【NOS0642】. ATK2 では, C1ISR 実行に必要な割込み出入り口処理を, ターゲット定義とする【IOS095】. また，C1ISR の処理内では，OS のシステムサービスは一部を除き使用してはならない【COS0602】.

C1ISR は OS が管理しない ISR であるため，C1ISR によりタスクの再スケジューリングが発生することはない【COS0603】．また，C1ISR は，C2ISR より割込み出入り口処理時間が短く，割込みが禁止される時間も短いため，応答性がよい.

C1ISR 実行中に，以下のシステムサービスを対で呼び出さなかった場合の動作は保証されない【NOS0378】.

・DisableAllInterrupts と EnableAllInterrupts
・SuspendOSInterrupts と ResumeOSInterrupts
・SuspendAllInterrupts と ResumeAllInterrupts

また，C1ISR も，C2ISR と同様に，ISR ID によって識別する【NOS0398】．C1ISR の ISR ID は，SC3，SC4 において，システムサービスの引数に，C1ISR を指定した場合に，エラーを検出する際に使用される.

ATK2 では，C1ISR が動作していることを OS が管理するために，C1ISR の出入り口処理で，OS が管理する処理レベルを C1ISR 処理レベルへ変更することができる【IOS045】．C1ISR の出入り口処理で C1ISR 処理レベルへ変更した場合，C1ISR から使用できないシステムサービスを呼び出すと，OS は E_OS_CALLEVEL を返す【IOS151】.

C2ISR

C2ISR は OS が管理する ISR である．OS は C2ISR 実行に必要な割込み出入り口処理を提供する【COS0604】．C2ISR の処理内では OS のシステムサービスを使用することができるため，C2ISR の出口処理ではタスクの再スケジューリングが発生する可能性がある.

2.7.2　再スケジューリングタイミング

OS は ISR 実行中に呼び出されたシステムサービスでは, 再スケジューリングを行わない【COS0605】. C2ISR 実行中に再スケジューリングが必要となった場合，OS はすべての C2ISR の実行が終了した後に，再スケジューリングを行う【COS0615】.

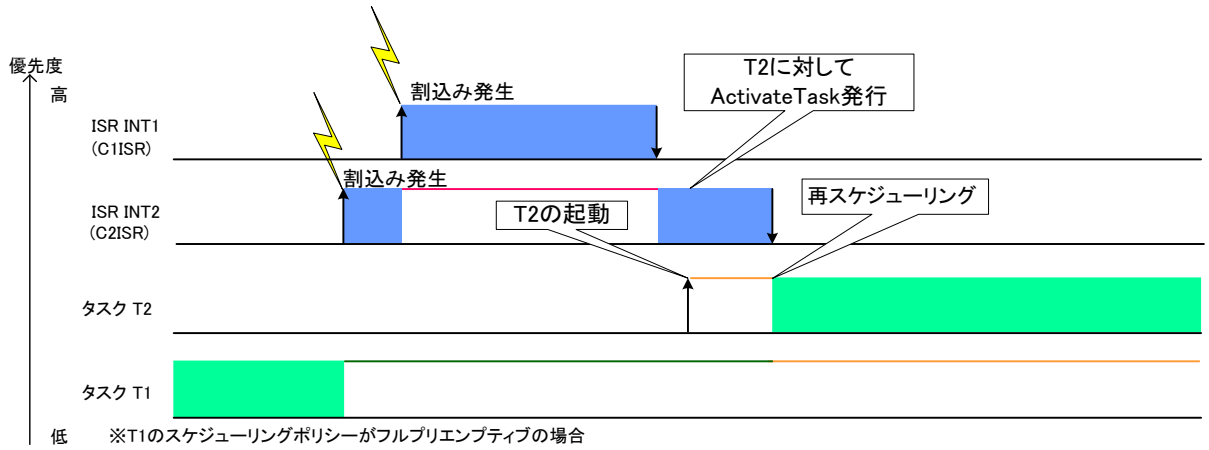


図 2-15 ISR 終了後の再スケジューリング動作の例

2.7.3　割込み管理の実装方法

C2ISR の本体記述

C2ISR の本体記述を以下に示す【COS3374】.

```
ISR(<ISR ID>)
    {
    }
```

C1ISR の本体記述

C1ISR の本体記述方法は実装定義である【COS3376】. ATK2 では，C2ISR と同様に，C1ISR 用の本体記述を用いて，C1ISR を宣言する．ATK2 における C1ISR の本体記述を以下に示す【IOS031】.

```
C1ISR(<ISR ID>)
    {
    }
```

また，C1ISR は，C 言語とアセンブリ言語のどちらでも定義可能とする【IOS032】. アセンブリ言語で定義する場合のルーチン名は，以下である【IOS033】.

```
C1ISRMain<ISR ID>
```

タイミング保護機能の適用

SC2，SC4 では C2ISR にタイミング保護機能が適用される(詳細は 2.17.2 節を参照)．C2ISR に対する時間制約はコンフィギュレーション時に静的に指定する必要がある.

OSAP の適用

SC3，SC4 の時に，C1ISR が信頼 OSAP に所属していない場合，ジェネレータはエラーを検出する【OS361】.

OS 自身が使用する割込みに対する制約

OS 自身が使用する割込み要因が, ISR のコンテナで指定された場合, エラーを検出する【NOS0363】. また，SC3，SC4 の時に，OS 自身が使用する割込み要因が，OSAP のコンテナで指定された場合もエラーを検出する【OS177】. なお，OS 自身が使用する割込みとは，タイミング保護違反を検出するために使用する割込みなどのことである.

2.7.4　割込み優先度

割込み優先度とは，割込み処理中に発生した割込み要求を受け付けるかを決定するために，割込みに割当てられる優先順位である．処理中の割込み優先度より高い優先度を持つ割込みであれば要求は受け付けられ，同じか，もしくは低い割込み優先度をもつ割込みであれば，要求は受け付けられない．

2.7.4.1　割込み優先度の範囲

割込み優先度の段階数は，1 以上である【COS0303】．割込み優先度は値が大きいほど高い優先順位を持つ【NOS0897】. 割込み優先度の最大値はターゲットの割込みコントローラに依存する【COS0607】.

ATK2 では，割込み優先度には，1 以上の値が割当てられる【IOS001】．コンフィギュレーション時のタスク優先度と割込み優先度は独立しており，割込み優先度は，タスク優先度の最大値より高くなるように解釈される【IOS002】．これは，タスクの上限優先度が変更された際に，割込み優先度を変更する必要をなくすためである．

OSEK 仕様との違い

OSEK 仕様では，ISR に対する割込み優先度の割当ては，実装とハードウェアに依存すると規定している【COS0305】．本仕様では，コンフィギュレーション時に割込み優先度を設定できるよう規定した≪NOS0665≫.

2.7.4.2　割込み優先度の制約

C1ISR の割込み優先度は，すべての C2ISR の割込み優先度より高く指定しなければならない【NOS0369】．マルチコア対応 OS では，すべてのコアを通して C1ISR の割込み優先度が，C2ISR の割込み優先度より高くなるように指定しなければならない【NOS1055】.

C1ISR より高い割込み優先度を持つ C2ISR がある場合，図 2-16 のように C1ISR に対して C2ISR が割込むケースが発生する．このとき，C2ISR 中のシステムサービスでタスク切換えの必要が生じた場合，C1ISR の出口処理では再スケジューリングが行われないため，再スケジューリング点が不定となる可能性がある．

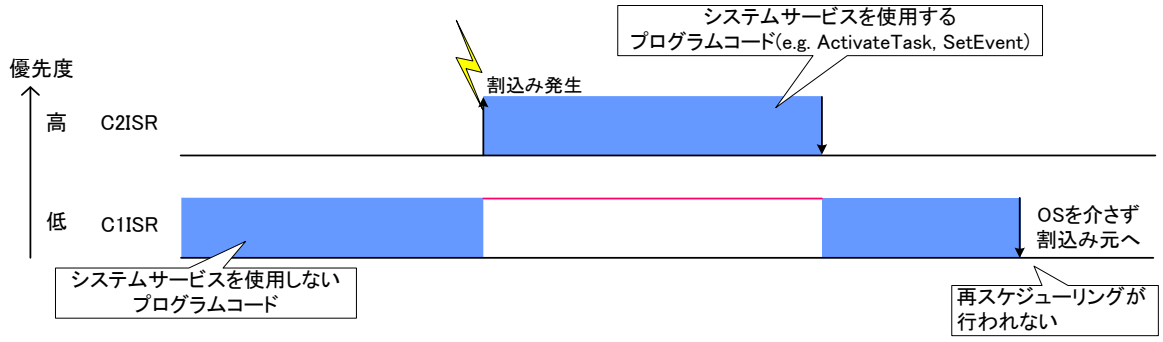


図 2-16 C1ISR に C2ISR が割込んだ場合の問題点

2.7.5　実行中の ISRID の取得

本 OS は, 現在実行中の ISR の ID を取得するシステムサービスとして GetISRID を提供する(詳細は 3.9.16 節を参照).

2.7.6　割込み禁止・許可状態の操作

本 OS は，割込みを禁止する機能と，許可する機能を提供する【COS0612】. 割込みの禁止と許可は組で使用しなければならないが, 割込みを禁止した状態で, タスクもしくは C2ISR を終了した場合は, サービス保護機能により割込みが許可される≪OS239≫≪OS368≫. ただし, 割込みを禁止した状態で, C1ISR を終了した場合の動作は保証されない【NOS0346】.

割込み禁止中に，割込みが発生した場合，割込みを許可した時点で，まだ割込み要求が保持されていれば，その時点で割込み要求を受け付ける【NOS1162】.

2.7.6.1　全割込みの禁止・許可状態の操作

本 OS は，すべての割込みを禁止するシステムサービスとして DisableAllInterrupts を提供する(詳細は 3.9.11 節を参照). また，DisableAllInterrupts によって禁止された割込みを許可するシステムサービスとして EnableAllInterrupts を提供する(詳細は 3.9.10 節を参照).

DisableAllInterrupts/EnableAllInterrupts は，割込み禁止状態をネストして操作することはできない.

本 OS は, すべての割込みを禁止するシステムサービスとして SuspendAllInterrupts を提供する. (詳細は 3.9.13 節を参照). また，SuspendAllInterrupts によって禁止された割込みを許可するシステムサービスとして ResumeAllInterrupts を提供する(詳細は 3.9.12 節を参照).

SuspendAllInterrupts/ResumeAllInterrupts は，割込み禁止状態をネストして操作することができる.

2.7.6.2　OS 割込みの禁止・許可状態の操作

本 OS は，すべての C2ISR を禁止するシステムサービスとして SuspendOSInterrupts を提供する．(詳細は 3.9.15 節を参照)．また，SuspendOSInterrupts によって禁止された割込みを許可するシステムサービスとして ResumeOSInterrupts を提供する(詳細は 3.9.14 節を参照)．

SuspendOSInterrupts/ResumeOSInterrupts は，割込み禁止状態をネストして操作することができる．

2.7.6.3　個別割込みの禁止・許可状態の操作

本 OS は，割込みを個別に禁止するシステムサービスとして DisableInterruptSource を提供する(詳細は 3.9.18 節を参照)．また，DisableInterruptSource によって禁止された割込みを許可するシステムサービスとして EnableInterruptSource を提供する(詳細は 3.9.17 節を参照)．

2.7.6.4　マルチコア対応 OS における割込み禁止・許可システムサービス

割込み禁止，許可を行うシステムサービスの対象となる割込み要因は，システムサービスを呼び出した処理単位が割付いているコアに接続されている割込み要因のみとする≪OS590≫≪OS591≫≪OS592≫≪OS593≫≪OS594≫≪OS595≫．

2.7.6.5　未使用の割込み番号の取り扱い

OS は，コンフィギュレーション時に指定されなかった割込み番号に対する割込みを，割込み禁止状態として起動し，以後割込みを許可することはない【NOS0857】．ただし，ターゲットによっては，未定義の割込みが発生した場合に，CPU 例外として扱うことがある【NOS1138】．

2.7.7　スタック共有

C2ISR は, 待ち状態となることがないため, 1 つの C2ISR が実行状態となってから終了するまでに, 他の同一割込み優先度の C2ISR が実行状態となることがない. したがって, 同一割込み優先度の C2ISR が使用するスタックは，ジェネレータによって共有される【NOS0651】．同一割込み優先度の C2ISR に設定したスタックサイズが異なる場合は，最も大きいサイズで確保され，それぞれの C2ISR のスタックサイズは最も大きいサイズに拡張される【NOS0660】.

ATK2 の SC3，SC4 のメモリ保護機能の機能レベル 3 では，C2ISR 毎にスタック領域を確保する≪IOS085≫．この場合，C2ISR 毎にスタックの先頭番地を指定可能であるので，割込み優先度に関わらず, 同一番地を指定することで複数の C2ISR のスタックを共有することができる≪IOS056≫≪IOS057≫．

2.8　イベント

イベントはタスクが同期処理を行うための OS オブジェクトである．

2.8.1　イベントとタスクの関係

イベントは拡張タスクでのみ使用することができる【COS0702】．イベントは単独の OS オブジェクトではなく，タスクが所有する OS オブジェクトである【COS0705】．タスクは複数のイベントを持つことができるが，タスクが持つことのできるイベント数には制限がある【COS0706】．

イベントは，所有するタスクとイベントマスク値によって識別される【COS0708】．

イベントの操作とタスク状態

イベント管理機能として，OS は以下の機能を提供する．

- イベントのセットおよび待ち解除
  タスクの持つイベントに対してイベントを通知すること．および，イベントの通知によってイベント待ちから実行可能状態とすること．

- イベント待ち
  イベントの通知を待つこと．

- イベントのクリア
  通知済みのイベントをクリアすること．

- イベントマスク値の取得
  通知済みのイベントマスク値を取得すること．

これらの操作によって，タスクは，実行状態から待ち状態，待ち状態から実行可能状態へと遷移する．

イベントの表現形式

イベントはビットマスクで表現される【COS0710】．イベントの各ビットが持つ意味はアプリケーション依存である．

イベントの初期状態

タスクのイベントはタスク起動時に OS によってクリアされる【COS0709】．

2.8.2　イベントの操作

2.8.2.1　イベントのセット

本 OS は，休止状態でない拡張タスクに対してイベントのセットを行うシステムサービスとして SetEvent を提供する(詳細は 3.9.19 節を参照).

2.8.2.2　イベントのクリア

本 OS は，イベントのクリアを行うシステムサービスとして ClearEvent を提供する(詳細は 3.9.20 節を参照)．イベントのクリアは ClearEvent を発行するタスク自身に対してのみ，行うことができる【COS0719】〔COS3520〕.

2.8.2.3　イベント待ち

本 OS は，イベント待ちを行うためのシステムサービスとして WaitEvent を提供する(詳細は 3.9.22 節を参照)．タスク種別が拡張タスクの場合のみ，イベント待ちを行うことができる.

拡張タスクがイベント待ちを要求すると，指定されたイベントがセットされるまで待ち状態となる【COS0722】．指定されたイベントのうち，少なくとも 1 つがセットされた場合，拡張タスクは待ち状態から実行可能状態に遷移する【COS0720】〔COS3553〕.

イベント待ちを要求した時点で既にイベントがセットされていた場合，タスクは実行状態を継続する≪COS0721≫.

2.8.2.4　イベントマスク値の取得

本 OS は，タスクが保持しているイベントマスク値を取得するシステムサービスとして GetEvent を提供する(詳細は 3.9.21 節を参照).

2.8.3　イベントの動作例

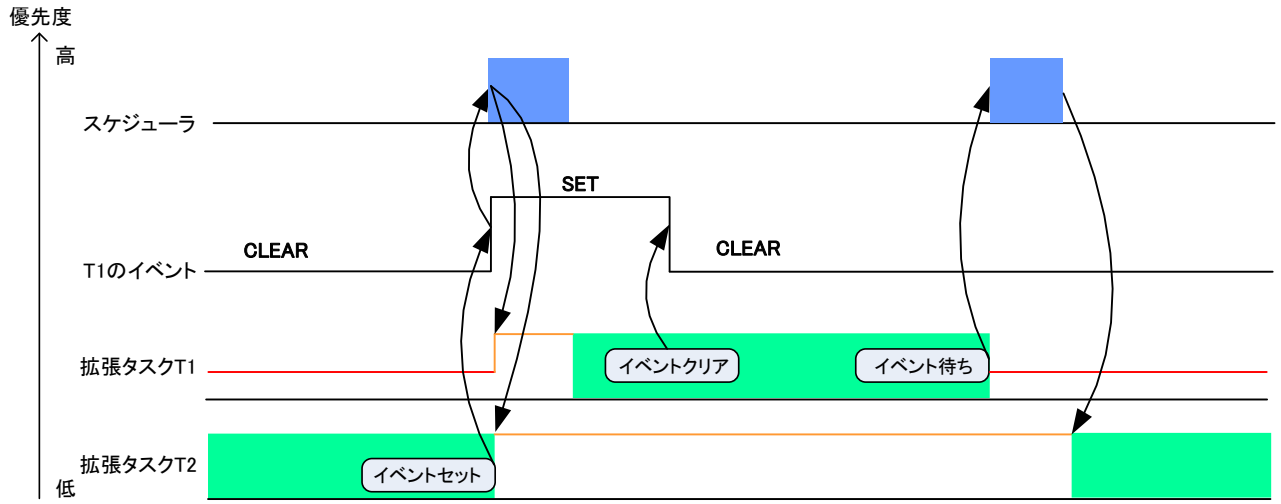


図 2-17 プリエンプティブタスクの同期例

図 2-17 の例では，待ち状態の T1 に対して T2 がイベントをセットすることにより，スケジューラが起動され T1 が実行可能状態へ遷移する．T1 の優先度は T2 より高いため，T2 は T1 にプリエンプトされる．その後，T1 がイベントクリアしてからイベント待ちすることにより再び待ち状態に遷移し，スケジューラが起動して T2 が実行される．

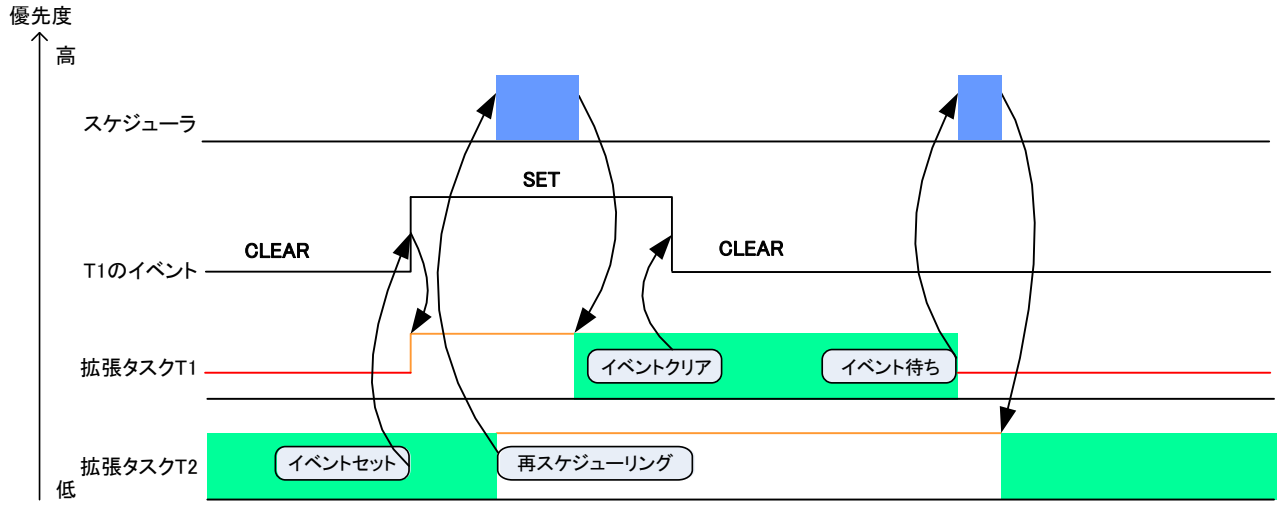


図 2-18 ノンプリエンプティブタスクの同期例

図 2-18 の例では，図 2-17 と同様に T1 に対して T2 がイベントをセットするが，ノンプリエンプティブタスクであるため再スケジューリングがすぐには行われず，T2 が再スケジューリングを明示的に行うまで T2 はプリエンプトされない．

2.8.4　イベントマスクの設定

イベントマスクの設定方法は，実装によっていくつかの方法が考えられるが，本節では ATK2 における，イベントマスクの設定について説明する．

ATK2 では，タスクごとに定義できるイベントの上限数を 32 とする【IOS026】．また，イベントの各ビットが持つ意味をタスク間で共有する使用方法を想定して，イベントは複数のタスクで使用可能とする【IOS027】．イベントマスク値は，コンフィギュレーション時に指定するか，ジェネレータによって自動的に割り当てることで決定する≪OS_Conf034≫．

なお，イベントはビットマスクで実現するため，異なるタスクで同じイベントマスクを使用する場合，組み合わせによっては，使用できないイベントマスクが発生する可能性がある．例えば，タスク A にイベント A が 0x01 で割り当てられ，タスク B にイベント B が 0x01 で割り当てられた場合，タスク C はイベント A とイベント B を，同時に定義することはできない．このようなコンフィギュレーションを行った場合，ジェネレータはエラーを検出する【IOS028】．ただし，本エラーチェックは，ジェネレータによって自動的に割り当てられたイベントマスク値に対してのみ行い，ユーザがイベントマスク値を指定した場合は，エラーチェックを行わない【IOS219】．

また，コンフィギュレーション時に指定するイベントマスク値に，複数のビットが立った値を指定することができる【IOS169】．ユーザがイベントマスク値を指定する場合は，複数のビットが立っているイベントマスクのビットが，他のイベントマスクのビットと重複してもよい【IOS170】．イベントマスクを自動的に割り当てる際，設定済みのイベントマスクと重複しないイベントマスクを用意できない場合，ジェネレータはエラーを検出する【IOS171】．

2.9　リソース

リソースは処理単位間で排他制御を行うための OS オブジェクトである．リソースはリソース ID によって識別する．

リソースを使用して排他区間を開始することをリソースの獲得と呼び，排他区間を終了することをリソースの解放と呼ぶ．また，処理単位がリソースを獲得して，排他状態にあることを，リソースの占有と呼ぶ．リソースはすべてのコンフォーマンスクラスで使用できる【COS0801】.

本 OS ではリソースによる排他制御の方法として優先度上限プロトコルを採用する．

2.9.1　リソースの種別

リソースオブジェクトに対しては，標準リソース，内部リソース，リンクリソースのいずれかを指定することができる．どのリソースを指定しても排他制御の振る舞いは同様である【COS0830】.

標準リソース

処理単位がシステムサービスを呼び出すことにより獲得，解放するリソース．

内部リソース

タスクが実行状態へ遷移する時，実行状態となったタスクに対して OS が自動的に獲得するリソース【COS0833】．ノンプリエンプティブスケジューリングにおける再スケジューリング時に，実行状態でなくなったタスクから OS によって自動的に解放される【COS0834】．GetResource，ReleaseResource により獲得，解放を行うことはできない【COS0829】.

リンクリソース

標準リソースにリンクされ，リンクされた標準リソースと同じ振る舞いをするリソース(処理単位がシステムサービスを呼び出すことにより獲得，解放する).

2.9.2　優先度上限プロトコルと上限優先度

本 OS では，優先度上限プロトコルを用いた排他制御を行う．

上限優先度

リソースは属性として上限優先度を持つ．リソースの上限優先度は，そのリソースを獲得するすべてのタスクの中で最も高いタスク優先度となる優先度である【COS0826】.

上限優先度の決定

ジェネレータはリソースを獲得するすべてのタスク優先度を解釈し，リソースの上限優先度を決定する【COS0825】．リソースを用いて排他制御を行うタスクに対して，コンフィギュレーション時に獲得するリソースを指定する必要がある．

2.9.3　リソースの操作

2.9.3.1　リソースの獲得

本 OS は, リソースを獲得するためのシステムサービスとして GetResource を提供する(詳細は 3.9.23 節を参照). タスクがリソースを獲得した時，OS はタスクの現在優先度を，リソースの上限優先度まで引き上げる【COS0827】〔COS3410〕. リソースを獲得しタスク優先度を引き上げることで，タスク優先度がリソースの上限優先度以下のタスクが実行されることを防止し，共有資源の排他制御を行う【COS0805】. したがって，複数のタスクや C2ISR が，同時に同じリソースを占有することはない.

リソースを獲得するタスクの初期優先度がリソースの上限優先度より高い場合は，リソースを獲得することはできない【COS0804】. 逆に，リソースを獲得するタスクの初期優先度がリソースの上限優先度以下の場合は, コンフィギュレーションに関係なく, リソースを獲得することができる【NOS0367】. また，タスク，C2ISR は同一のリソースをネストして獲得することはできない【COS0810】.

同一リソースをネストして獲得する要求に対応するため，定義済みの標準リソースと全く同じ振る舞いをするリンクリソースを，コンフィギュレーションで定義可能とする【COS0839】〔OS_Conf052〕. リンクリソースは，リンク元の標準リソースとは別のリソース ID が振られ，リンクリソースとリンク元の標準リソースはネストして獲得可能である【COS0840】. 標準リソースと同様に，同じリンクリソースは，ネストして獲得できない【COS0841】.

2.9.3.2　リソースの解放

本 OS は，GetResource によって獲得したリソースを解放するためのシステムサービスとして ReleaseResource を提供する(詳細は 3.9.24 節を参照).

リソースを解放した時点で，OS は再スケジューリングを行う【COS0821】〔COS3424〕. この時，リソースを解放したタスクがシステムの中で最高優先度のタスクでない場合，リソースを解放したタスクはプリエンプトされ，OS は最高優先順位を持つタスクを開始する.

2.9.3.3　リソース占有中の状態

タスクはリソースを占有した状態で TerminateTask，ChainTask，Schedule，WaitEvent のシステムサービスを呼出してはならない【COS0811】〔COS3235〕〔COS3255〕〔COS3266〕〔COS3550〕. また，C2ISR はリソースを占有した状態で処理を終了してはならない≪OS369≫. ただし，これらの制約は内部リソース占有中には適用されない【COS0835】〔NOS0718〕.

先に獲得したリソースの占有中に, 異なるリソースをネストして獲得することができる【COS3412】. リソースはネストして獲得することができるが，LIFO の順序で解放を行わなくてはならない【COS0813】〔COS3429〕.

<u>使用上の注意</u>

リソースの獲得と解放は，同一の関数内で行うことを推奨する.

2.9.3.4　リソースの獲得と解放による排他例

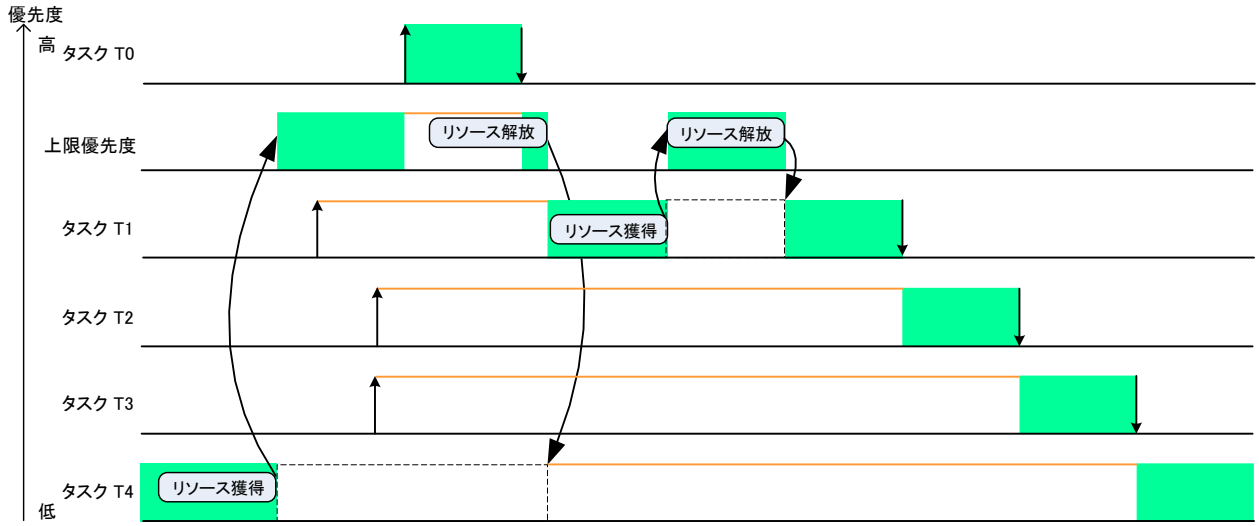


図 2-19 リソースによるタスク間の排他例

図 2-19 では，T1 と T4 がリソースを獲得可能であり，T4 がリソースを占有している間は T1 が実行状態にならない．また，優先度が引き上げられることにより T2，T3 も実行状態にならず，優先度逆転も起こらないことを示している．

2.9.4　優先度上限プロトコルの C2ISR 処理レベルへの拡張

本 OS は，オプションとして優先度上限プロトコルを C2ISR 処理レベルまで拡張し，タスク－ISR 間，ISR－ISR 間でリソースによる排他制御が可能な実装としてもよい【COS0802】〔IOS168〕．これを優先度上限プロトコルの C2ISR 処理レベル拡張と呼ぶ．ただし，内部リソースに対しては，優先度上限プロトコルの C2ISR 処理レベル拡張はできない【NOS0718】．C2ISR 処理レベルへ拡張した優先度上限プロトコルと通常の優先度上限プロトコルの違いを以下に示す．

ATK2 では，優先度上限プロトコルの C2ISR 処理レベルへの拡張をサポートする【IOS168】．

2.9.4.1　ISR によるリソースの獲得

タスク－ISR 間，ISR－ISR 間の場合，リソースを獲得するすべての ISR の中で最も高い割込み優先度を持つ ISR の優先度を，リソースの仮想的な上限優先度とする【COS0823】．ISR が獲得するリソースをタスクが獲得した場合，OS は，タスク優先度を仮想的な上限優先度まで引き上げる【NOS0719】．ISR が獲得するリソースをタスクが獲得した場合，仮想的な上限優先度は，実際の割込み優先度と等しくなり，仮想的な上限優先度以下の割込み要求は受け付けられない【NOS0898】．

OSEK 仕様との違い

OSEK 仕様では，仮想的な上限優先度と割込み優先度の対応関係は実装定義であると規定されている【COS0824】．しかし，ISR が獲得するリソースをタスクが獲得した場合，引き上げられた仮想的な優先度以下の割込み要求を受け付けない必要があるので，本仕様では，実装定義ではなく，割込み優先度は仮想的な割込み優先度と等しくなると規定した≪NOS0898≫．

2.9.4.2　ISR によるリソースの解放

リソース獲得により ISR との排他を行う場合，リソース解放時に再スケジューリングが行われるだけでなく，保留されていた ISR が起動されることがある【COS0809】．

2.9.4.3　ISR によるリソースの獲得と解放による排他例

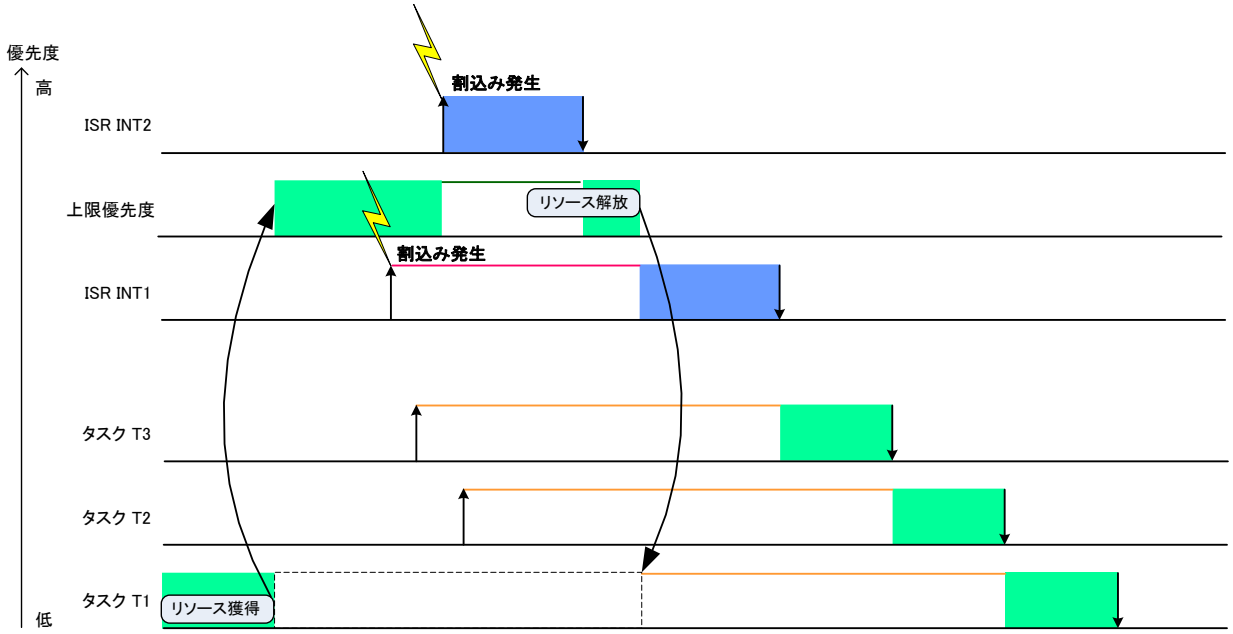


図 2-20 タスクーC2ISR 間の排他例

図 2-20 の例では, T1 と INT1 がリソースを獲得可能であり, T1 がリソースを占有している間に T2, T3 を起動してもリソースの獲得により優先度が引き上げられているため実行状態を継続する．また, INT1 が起動しても INT1 の実行がペンディングされ，T1 は実行状態を継続する．一方，INT2 の優先度は上限優先度より高いため，T1 実行中に割込みが発生すると INT2 が実行される．

その後 T1 がリソースを解放すると優先度が下がるため, ペンディングされていた INT1 が実行され, さらに実行可能状態となった T2 と T3 が実行，終了してから T1 の実行が行われる．

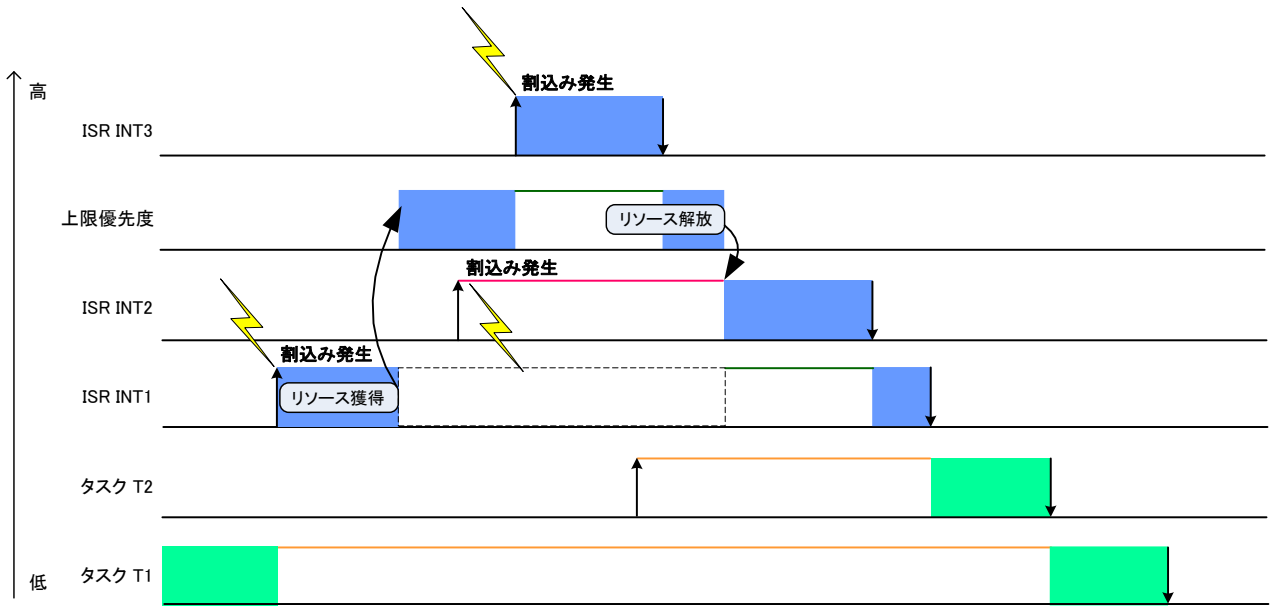


図 2-21 C2ISR－C2ISR 間の排他例

図 2-21 の例では，INT1 と INT2 がリソースを獲得可能であり，INT1 実行中に INT2 が発生してもリソースの獲得により優先度が引き上げられているため，INT2 はペンディングされ INT 1 の実行が継続される．一方, INT3 は上限優先度より優先度が高いため, INT1 に割込んで実行され T2 を起動する．INT3 の終了後に INT1 がリソースを解放するとペンディングしていた INT2 が実行され, INT2 の実行を終了後に INT1 の残りの処理を実行する．また，割込み処理が終了した時点で，T2 の優先度が T1 より高いため，T2 が実行され，T2 の終了後に T1 が実行される．

2.9.5　内部リソースとグループタスクとの関係

同一の内部リソースを持つタスク群はグループタスクとして振舞う(詳細は 2.5.9 節を参照).

2.9.6　スケジューラリソース

AUTOSAR 仕様では，スケジューラリソースをサポートしないため，本仕様においてもサポートしない. 本節では, スケジューラリソースをサポートしないことによる影響と, その対応について説明する.

OSEK 仕様との互換性

OSEK 仕様では，スケジューラリソースは「RES_SCHEDULER」という ID を持つ．本 OS では，OSEK 仕様で開発されたアプリケーションとの互換性のために，標準リソースを使用して，同様の振る舞いを実現することが可能である．具体的には，コンフィギュレーション時に，「RES_SCHEDULER」という ID の標準リソースを生成し，すべてのタスクから獲得するように指定する．C2ISR からは，「RES_SCHEDULER」を獲得しない.

SC3，SC4 においては，「RES_SCHEDULER」に対して，すべての OSAP からのアクセス権を付与する.

BCC1 における取扱い

OSEK 仕様では，BCC1 におけるリソースの最小機能セットは，「スケジューラリソースのみ」と規定されている≪COS0317≫．これは，スケジューラリソースへの GetResource により，タスクディスパッチを禁止する状態を実現できることを想定していると考えられる.

AUTOSAR 仕様では，スケジューラリソースをサポートしないことから，表 2-8 における BCC1 の使用できる内部リソース数を 1 と変更してもよいが，タスクディスパッチを禁止するだけの実装であれば，リソースの実装は不要であるため，本仕様では「すべてのタスクに関連付いたリソースを 1 つ」とした.

2.9.7　マルチコア対応 OS におけるリソース

リソースは，優先度上限プロトコルによって排他制御を行うが，スケジューリングは各コアで独立して実行するため，コア間でのリソースによる排他制御はできない．コア間での排他制御には，スピンロックを使用する(詳細は 2.26 節を参照).

同じリソースに，異なるコアに割付くタスク，C2ISR を関連付けした場合，ジェネレータはエラーを検出する【OS662】.

2.10 カウンタ

カウンタは，何らかの事象をカウントし，接続された他の OS オブジェクトに対して処理タイミングの通知を行うための OS オブジェクトである．カウンタはカウンタ ID によって識別する．

接続対象となる OS オブジェクトはアラームとスケジュールテーブルで，カウンタによって両者を駆動する．

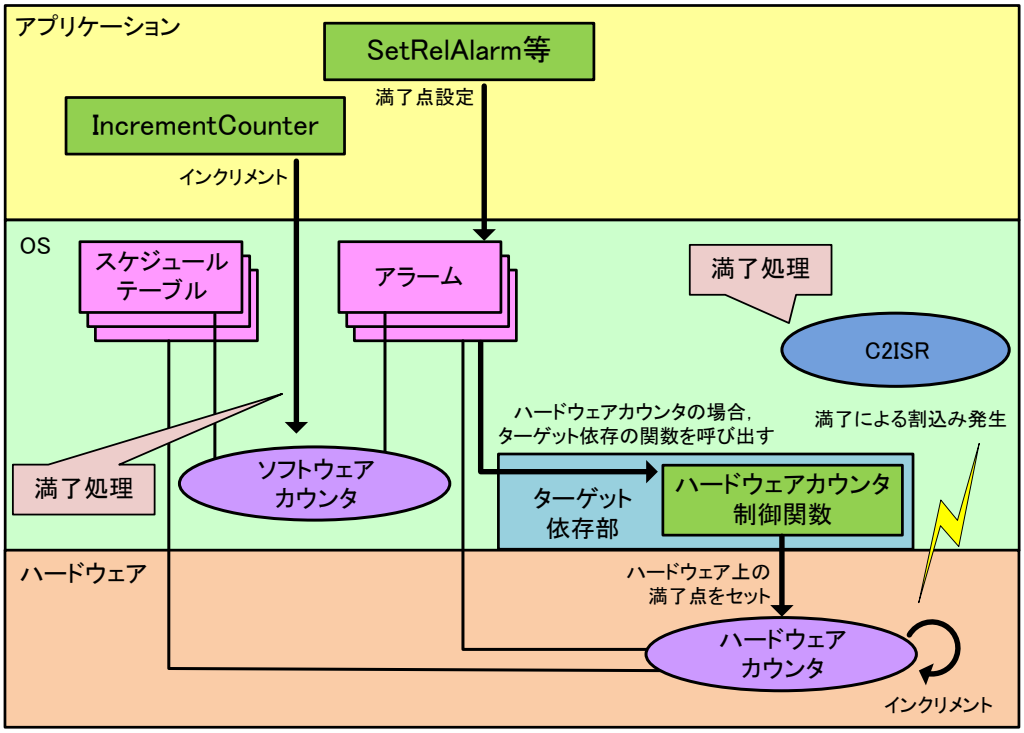


図 2-22 カウンタの概念図

2.10.1 ティック

カウンタはティックという単位で事象をカウントする．ティックが示すものはアプリケーション依存でカウンタ毎に異なる【COS0904】．カウンタは，カウントしたティックを保持し，この値をカウント値と呼ぶ．

2.10.2 カウンタの種別

カウンタは以下の 2 種類に分類される．

- ソフトウェアカウンタ

  ティックが OS によって管理され，システムサービスによりティックがインクリメントされるカウンタ

・ ハードウェアカウンタ
　　タイマなどのハードウェアによってティックが管理されるカウンタ

OSEK 仕様との違い

OSEK 仕様では，システムカウンタは少なくとも 1 つ常に存在すると規定している【COS3686】．本仕様では，システムカウンタは必要に応じてアプリケーションで用意するものとして削除した．

2.10.3 カウンタの属性

カウンタは以下の属性を持つ．

・ カウンタ最大値
　　カウンタティックの最大値

・ 最小周期
　　カウンタに接続する OS オブジェクトが指定できる最小周期

2.10.4 カウンタの設定

カウンタに接続する OS オブジェクトの指定やカウンタの属性の設定は，コンフィギュレーション時に静的に指定する【COS0927】．カウンタは複数の OS オブジェクトと接続することが可能である【COS0914】.

オブジェクトのコア割付けに関する制約

アラームが接続されるカウンタは，アラームと同一のコアに割付けられていなければならない【OS631】〔OS663〕．スケジュールテーブルが接続されるカウンタは，スケジュールテーブルと同一のコアに割付けられていなければならない【OS630】〔OS665〕.

2.10.5 カウンタの操作

2.10.5.1 カウンタのインクリメントによるアクションの起動

OS はソフトウェアカウンタをインクリメントするシステムサービスとして IncrementCounter を提供する(詳細は 3.9.25 節を参照)．カウンタがインクリメントされ，カウンタに接続する OS オブジェクトで指定されたティックになったとき，カウンタは OS オブジェクトに指定されたアクションを実行する【COS0906】.

カウンタのティックは，カウンタ最大値(OsCounterMaxAllowedValue)までインクリメントされ，カウント値がカウンタ最大値であるカウンタに対して，IncrementCounter を発行すると，カウント値は 0 となる【OSa001】.

マルチコア対応 OS において，異なるコアに割付いているカウンタは，インクリメントできない

【OS629】．したがって，アラーム，スケジュールテーブルの満了処理を実行するコアは，IncrementCounter を呼び出したコアとなる．

2.10.6 ハードウェアカウンタ

2.10.6.1 C2ISR による満了処理

ハードウェアカウンタに接続されたアラーム，スケジュールテーブルの満了処理は，ハードウェアカウンタに対応する C2ISR によって実行される【NOS0671】．したがって，アラーム，スケジュールテーブルをハードウェアカウンタに接続して使用するには，満了時に起動する C2ISR が必要である【NOS0672】．ハードウェアカウンタに対応する C2ISR は，コンフィギュレーション時に静的に指定する【NOS0673】．

複数アラーム，スケジュールテーブルが同時に満了する場合，各満了処理の間に，ハードウェアカウンタより割込み優先度の高い C2ISR が起動する可能性がある【NOS0685】．

2.10.6.2 ハードウェアカウンタ制御関数

ハードウェアカウンタに対する満了点の設定や，現在ティックの取り出しなどの処理はハードウェアに依存するため，ターゲット毎に用意する【NOS0674】．これらのハードウェアカウンタを制御する関数を，ハードウェアカウンタ制御関数と呼ぶ．ハードウェアカウンタ制御関数の仕様を表 2-12 に示す．<counter id>は，コンフィギュレーション時にハードウェアカウンタに付与するカウンタ ID を示す．

ハードウェアカウンタ制御関数は，OS 処理レベルで呼び出される【NOS0691】．

AUTOSAR 仕様との違い

AUTOSAR 仕様では，ハードウェアカウンタの具体的な実装方法が定義されていないため，本仕様で規定を行った．

表 2-12 ハードウェアカウンタ制御関数仕様

関数定義	仕様
void init_hwcounter_<counter id> (TickType maxval, TimeType nspertick) 【NOS0675】	<counter id>が示すハードウェアカウンタで必要な初期化処理を実行する．maxval で指定された値を最大値としてカウンタが循環するようにハードウェアへ設定する．nspertick は，OsSecondsPerTick で指定した値をナノ秒で表現した値である． 本関数は，OS の初期化処理によって実行される．
void start_hwcounter_<counter id> (void) 【NOS0676】	<counter id>が示すハードウェアカウンタを開始する． 本関数は，OS のハードウェアカウンタ初期化後に実行される．
void stop_hwcounter_<counter id>(void) 【NOS0677】	<counter id>が示すハードウェアカウンタを停止する． 本関数は，OS の終了処理によって実行される．
void set_hwcounter_<counter id> (TickType exprtick)【NOS0678】	<counter id>が示すハードウェアカウンタに対して満了点を設定する．exprtick は次の満了点の絶対時刻を示す値である． 本関数は，アラーム，スケジュールテーブルの次回満了点を設定する OS 内部処理から実行される．
TickType get_hwcounter_<counter id> (void)【NOS0679】	<counter id>が示すハードウェアカウンタの現在カウント値を取得して返す．ダウンカウンタの場合，アップカウンタとしての値を返す． 本関数は，GetCounterValue，GetElapsedValue が呼び出された際に，OS 内部処理から実行される．
void cancel_hwcounter_<counter id> (void)【NOS0680】	<counter id>が示すハードウェアカウンタに設定されている満了点を削除する． 本関数は，アラーム，スケジュールテーブルの次回満了点をキャンセルする際に，OS 内部処理から実行される．
void trigger_hwcounter_<counter id> (void)【NOS0681】	<counter id>が示すハードウェアカウンタに対応する C2ISR を起動する割込みを発生させる． 本関数は，満了点の間隔が小さい場合などに，1 つ目の満了点に対するキャンセル処理を実行中に，次の満了時刻が経過してしまった場合に，強制的に満了処理を実行するために，OS 内部処理から実行される．

void int_clear_hwcounter_<counter id>(void)【NOS0682】	<counter id>が示すハードウェアカウンタに対応する C2ISR が起動した際に, 割込み要求のクリアなどのハードウェアに対して行う処理を実行する．処理の必要がないハードウェアは空の関数とする． 本関数は，ハードウェアカウンタに対応する C2ISR が起動した直後に実行される．
void int_cancel_hwcounter_<counter id>(void)【NOS0683】	<counter id>が示すハードウェアカウンタに対応する C2ISR に対する割込み要求がペンディングされている場合に，割込み要求をキャンセルする．一度発生した割込み要求をキャンセルできないハードウェアの場合は，空の関数とする． 本関数は，満了点の間隔が小さい場合などに，1 つ目の満了処理実行中に，次の満了時刻が経過してしまった場合に，不要な C2ISR の起動を抑止するために，OS 内部処理から実行される．前述の例で，空の関数とした場合，何も処理を実行しない C2ISR が起動する．
void increment_hwcounter_<counter id>(void)【NOS0684】	本関数はデバッグ用であるので，空の関数とする． 本関数を実装する場合は，<counter id>が示すハードウェアカウンタのティック値をインクリメントする．ただし，stop_hwcounter_<counter id>によってハードウェアカウンタが停止している前提での使用を想定している．

2.11 ソフトウェアフリーランタイマ

ソフトウェアフリーランタイマは，物理的なタイマを複数のアプリケーションから共有するために OS が提供する機能である．ソフトウェアフリーランタイマは，カウンタにより実現する．

タイマの初期化

OS が直接タイマの操作を行い，GPT ドライバが使用しないすべてのタイマの初期化と設定を行う【OS374】．ソフトウェアカウンタの初期値は 0 とし，ハードウェアカウンタの初期値は init_hwcounter_<counter id>でユーザが設定する【NOS0396】.

タイマ機能の抽象化

OS が扱うタイマを，0 からカウントを開始し，カウント値が単調増加して一定値に達した後，0 に戻るタイマとして振舞うように抽象化する【OS384】.

カウンタの読出し

OS はカウンタの現在値を読み出すシステムサービスとして GetCounterValue を提供する(詳細は 3.9.26 節を参照)．読み出される値は，カウンタがハードウェアカウンタである場合はタイマのカウントの値，ソフトウェアカウンタである場合はソフトウェアで管理されたティックの値となる．

また，OS はカウンタの過去の値と現在値の差分を求めるシステムサービスとして GetElapsedValue を提供する(詳細は 3.9.27 節を参照).

2.12 アラーム

アラームは，繰り返し処理を行うための OS オブジェクトである．アラームはアラーム ID によって識別する．

アラームの動作を開始することをアラームのセット，アラームの動作を停止することをアラームの停止，アラームが処理するタイミングになったことをアラームの満了と呼ぶ．

アラームは別の OS オブジェクトであるカウンタに接続し，カウンタの駆動によって動作する【COS0903】.

マルチコア対応 OS においては，アラームに対する処理を，アラームが所属する OSAP が割付けられているコアにて実行する【OS634】.

2.12.1 アラームの満了タイミング

アラームの満了は，接続されたカウンタのティックが，あらかじめ設定したティックに到達した時に発生する【COS0910】.

2.12.2 アラーム満了時に実行させるアクション

アラーム満了時に実行させるアクション(1 つのみ)は，コンフィギュレーション時に指定を行う．アラームに指定できるアクションを以下に示す【COS0908】．なお，これらの処理内容はユーザが呼び出すシステムサービスの処理内容と同じである【COS0920】.

タスクの起動

コンフィギュレーション時に指定されたタスクを起動する．

マルチコア対応 OS においては，他のコアに割付いたタスクを起動することもできる【OS632】.

イベントのセット

コンフィギュレーション時に指定されたタスクに対して，指定されたイベントをセットする．

マルチコア対応 OS においては，他のコアに割付いたタスクのイベントをセットすることもできる【OS633】.

アラームコールバックの実行

コンフィギュレーション時に指定された関数をアラームコールバックとして実行する．

マルチコア対応 OS においては，アラームが割付いているコアにてアラームコールバックを実行する【OS635】．アラームコールバックは SC1 でのみ使用できる≪OS242≫.

カウンタの駆動

コンフィギュレーション時に指定されたカウンタのティックをインクリメントする【OS301】.

マルチコア対応 OS においては，他のコアに割付いたカウンタをインクリメントすることはできない

≪OS629≫.

アラームの動作例

図 2-23 にアラームの動作例を示す．以下の動作例では，アラーム 1 はカウンタが 5 の時に単発でコールバック呼出しを行い，アラーム 2 はカウンタが 4 の時から 3 カウント周期でタスクの起動を行っている．

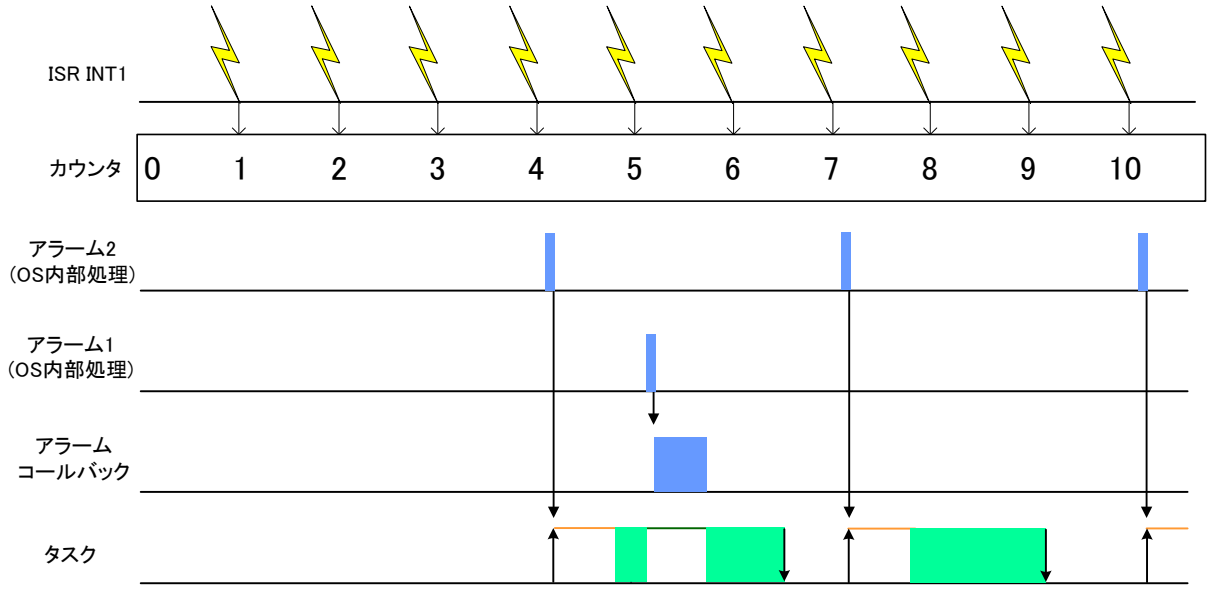


図 2-23 アラームの動作例

2.12.3 アラームの操作

2.12.3.1 アラームのセット

本 OS は，アラームに対して絶対時刻による満了時刻のセットを行うシステムサービスとして SetAbsAlarm を提供する(詳細は 3.9.31 節を参照)．また，現在時刻からの相対時刻による満了時刻のセットを行うシステムサービスとして SetRelAlarm を提供する(詳細は 3.9.30 節を参照)．

アラームのセットは，以下のパラメータの組み合わせで動的に指定する．

起動回数の指定

アラームにセットするアクションの起動回数の種別として，以下の 2 つを提供する【COS0912】．

- 単発アラーム
- 周期アラーム

初回起動タイミングの指定

アラームのセット時に，初回の起動タイミングを指定する．

周期の指定

初回起動後，2 回目以降の起動周期を指定する【COS0922】．2 回目以降の起動周期を 0 とすることで，単発アラームとして指定することが可能である．

2.12.3.2 アラームの停止

本 OS は，アラームの停止を行うシステムサービスとして CancelAlarm を提供する(詳細は 3.9.32 節を参照)．

2.12.3.3 アラームの情報取得

本 OS は，アラームの情報を取得するシステムサービスとして GetAlarmBase を提供する(詳細は 3.9.28 節を参照)．また，アラームが満了するまでのティック数を取得するシステムサービスとして GetAlarm を提供する(詳細は 3.9.29 節を参照)．

2.12.4 アラームコールバック

アラームコールバックはアラーム満了時に，OS によって実行される処理単位である．
アラームコールバックは SC1 でのみ使用することができる【OS242】．

補足事項

アラームコールバックが SC2，SC3，SC4 で使用できない理由は，アラームコールバックに対するタイミング保護やメモリ保護の実現が非効率であるからである．アラームコールバックに対する時間保護

やメモリ保護をサポートしなくても，アラームからタスクを起動することで，同等の処理が可能である．

2.12.4.1 アラームコールバックの処理レベル

アラームコールバックは OS 処理レベルで動作する【COS0925】．そのため，C2ISR が禁止された状態で実行される．アラームコールバックは呼出し可能なシステムサービスが制限される(詳細は表 3-1 を参照)．

2.12.4.2 アラームコールバックの記述方法

アラームコールバックの記述方法を示す．

アラームコールバックのプロトタイプ宣言

アラームコールバックのプロトタイプ宣言を以下に示す【COS0924】．

```
ALARMCALLBACK(AlarmCallbackRoutineName);
```

アラームコールバックの本体記述

アラームコールバックの本体記述を以下に示す【COS3691】．アラームコールバックは引数，返り値を持たない【COS0923】．

```
ALARMCALLBACK(<アラームコールバック名>)
{
}
```

2.12.5 アラームの設定

アラームはコンフィギュレーション時に以下の情報を指定する．

- アラームに接続するカウンタ【COS0915】
- アラーム満了時に起動するタスク，イベント，アラームコールバック，カウンタ【COS0916】

オブジェクトの OSAP 所属に関する制約

OSAP が再起動する場合，OSAP に所属するカウンタは初期化されてから再スタートするので，他の OSAP に所属するアラームが接続されていると，正しいタイミングで満了処理を実行できない可能性がある．したがって，保護違反時処理の機能レベルが 2 の場合，非信頼 OSAP に所属するカウンタに対して，他の OSAP に所属するアラームを接続することはできない【NOS1129】．保護違反時処理の機能レベルが 3 の場合，すべてのカウンタに対して，他の OSAP に所属するアラームを接続することはできない【NOS1130】．

オブジェクトのコア割付けに関する制約

アラームは，接続するカウンタと同一のコアに割付けられていなければならない≪OS631≫.

2.13 スケジュールテーブル

スケジュールテーブルとは，タスク起動やイベントセットなどの一連の処理を，静的なテーブルに記述されたタイミングで実行するための OS オブジェクトである．スケジュールテーブルはスケジュールテーブル ID によって識別する．

スケジュールテーブルは，別の OS オブジェクトであるカウンタに接続し，カウンタの駆動によって動作する．また，スケジュールテーブル制御は接続されたカウンタのティックと，異なる供給源を持つティックを同期させる同期機能を提供する．

スケジュールテーブルを使用する利点

スケジュールテーブルを使うことにより，アラームでは停止とセットの組み合わせで実現しなければならないタスク起動やイベントセットなどの一連の処理を，静的な定義により実現することができる．

2.13.1 スケジュールテーブルの構成要素

スケジュールテーブルは以下の要素から構成される．

スケジュールテーブル周期

スケジュールテーブルの処理が一回りするティック数．

満了点

スケジュールテーブル内で，タスク起動，イベントセットを実行する時刻．スケジュールテーブルは少なくとも 1 つの満了点を持つ【OS401】〔OS_Conf143〕．1 つの満了点には，少なくとも 1 つの満了点処理を持つ【OS407】〔NOS0852〕．すべての満了点は，一意なオフセットで満了時刻を指定する【OS442】〔NOS0851〕．満了時刻は，スケジュールテーブルの先頭からのオフセットで指定する【OS404】〔OS_Conf062〕．

初期オフセット

スケジュールテーブルの先頭から先頭満了点までのティック数．初期オフセットは 0 か，もしくはスケジュールテーブルが接続されたカウンタの最小周期(OsCounterMinCycle)以上，カウンタの最大値(OsCounterMaxAllowedValue)以下でなくてはならない【OS443】.

遅延

満了点間のティック数．遅延はスケジュールテーブルが接続されたカウンタの最小周期(OsCounterMinCycle)以上，カウンタの最大値(OsCounterMaxAllowedValue)以下でなくてはならない【OS408】.

最終遅延

スケジュールテーブルの最終満了点からスケジュールテーブル終端までのティック数．最終遅延はスケジュールテーブルが接続されたカウンタの最小周期(OsCounterMinCycle)以上，カウンタ最大値(OsCounterMaxAllowedValue)以下でなくてはならない【OS444】．例外として，単発動作指定時のみ，最終遅延を 0 以上，カウンタ最大値(OsCounterMaxAllowedValue)以下としてもよい【OS427】．

スケジュールテーブル時刻

スケジュールテーブルの現在のティック．

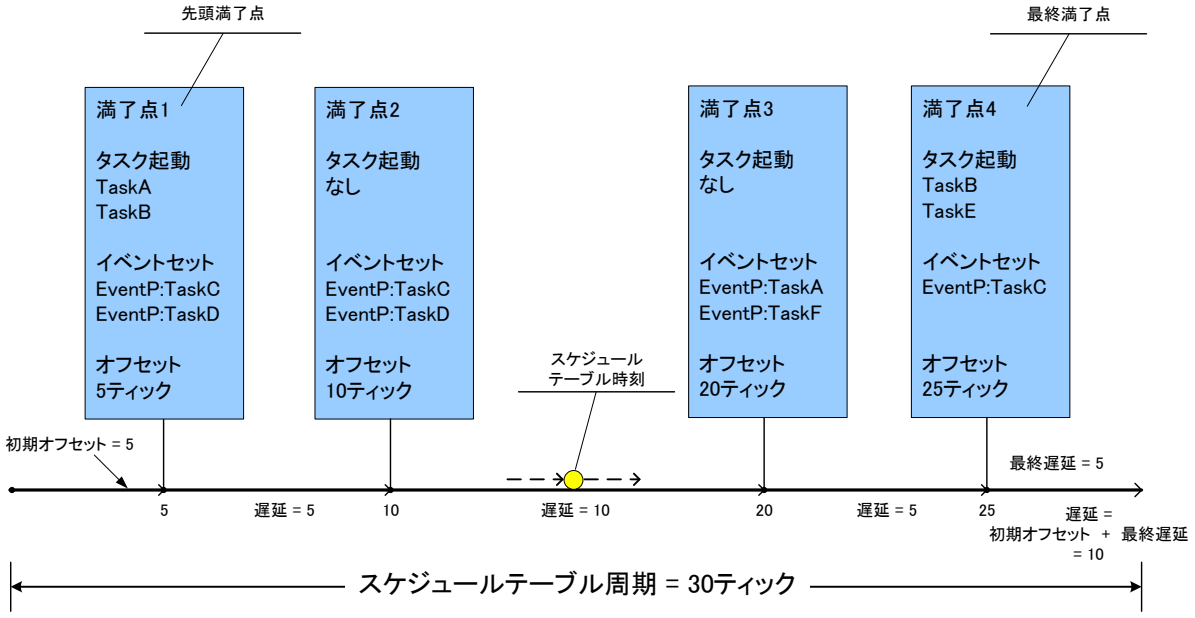


図 2-24 スケジュールテーブルの構成

2.13.2 スケジュールテーブルを制御するカウンタ

本 OS は 2 種類のカウンタによってスケジュールテーブルを制御する.

駆動カウンタ

スケジュールテーブルと接続されたカウンタ．駆動カウンタは本 OS で定義された OS オブジェクトのカウンタである．スケジュールテーブルは 1 つのカウンタで制御される【OS409】〔OS_Conf145〕. 1 つのカウンタは，少なくとも 1 つのスケジュールテーブルを駆動することができる【OS410】.

同期カウンタ

明示同期スケジュールテーブルで使用する絶対時刻の供給源．同期カウンタは OS オブジェクトのカウンタではない.

2.13.3 スケジュールテーブルの種別

スケジュールテーブルは以下の 3 種類に分類される.

同期なしスケジュールテーブル

同期機能を使用しないスケジュールテーブル.

暗黙同期スケジュールテーブル

駆動カウンタと常に同期しているスケジュールテーブル．OS は暗黙同期のスケジュールテーブルに対して特別な機能を提供しない.

明示同期スケジュールテーブル

ユーザが同期カウンタの絶対時刻を OS に供給し，OS が駆動カウンタと同期カウンタを同期させるスケジュールテーブル.

同期機能のユースケース

スケジュールテーブル時刻と，スケジュールテーブルに接続されたカウンタの周期(OsCounterMaxAllowedValue+1)が異なるため，スケジュールテーブルの先頭満了点処理が実行される絶対時刻は一定でない．そこで，絶対時刻とスケジュールテーブル時刻を同期させるため，OS は同期機能を提供する.

2.13.4 スケジュールテーブルの状態

スケジュールテーブルは以下の状態を持つ.

停止状態(SCHEDULETABLE_STOPPED)

スケジュールテーブルが停止している状態．満了点処理は実行されない.

動作状態(SCHEDULETABLE_RUNNING)

カウンタのティックに従って満了点処理を実行する状態．同期なしスケジュールテーブルにおいては，満了点処理は実行されるが，同期カウンタとは非同期である.

切換え待ち状態(SCHEDULETABLE_NEXT)

スケジュールテーブルの切換え待ち要求を受けた状態．具体的には，現在動作しているスケジュールテーブルが，スケジュールテーブルの切換え要求を受けた際に，切換え後のスケジュールテーブルに指定され，現在動作しているスケジュールテーブルの終了を待っている状態．満了点処理は実行されない.

同期待ち状態(SCHEDULETABLE_WAITING)

明示同期スケジュールテーブルにおいて同期カウンタのティック供給を待っている状態．満了点処理は実行されない.

同期動作状態(SCHEDULETABLE_RUNNING_AND_SYNCHRONOUS)

明示同期スケジュールテーブルにおいて駆動カウンタと同期カウンタが同期しており，駆動カウンタのティックに従って満了点処理を実行する状態．暗黙同期スケジュールテーブルにおいては，カウンタのティックに従って満了点処理を実行する状態.

さらに，タイミングによって動作状態は，以下の状態に区別される．以下の状態は，本文書における便宜上の呼称であり，スケジュールテーブルの状態として OS で管理する際に区別するものではない.

動作開始待ち状態

スケジュールテーブルの開始要求を受けた後，指定されたティックとなるまでの間の状態．状態は，SCHEDULETABLE_RUNNING であるが，満了点処理は実行されない．OS 管理上は，動作状態である.

切換え受付け済み動作状態

動作状態のスケジュールテーブルが，切換え要求を受けた状態．切換え要求を受けたスケジュールテーブルは，周期起動であっても，切換えと同時に停止する≪NOS0661≫．OS 管理上は，動作状態である.

切換え受付け済み同期待ち状態

同期待ち状態のスケジュールテーブルが，切換え要求を受けた状態．切換え要求を受けたスケジュールテーブルは，周期起動であっても，切換えと同時に停止する≪NOS0661≫．OS 管理上は，同期待ち状態である．

切換え受付け済み同期動作状態

同期動作状態のスケジュールテーブルが，切換え要求を受けた状態．切換え要求を受けたスケジュールテーブルは，周期起動であっても，切換えと同時に停止する≪NOS0661≫．OS 管理上は，同期動作状態である．

2.13.5 スケジュールテーブルの状態遷移

2.13.5.1 同期なしスケジュールテーブルの状態遷移

同期なしテーブルの状態遷移を図 2-25 に示す．

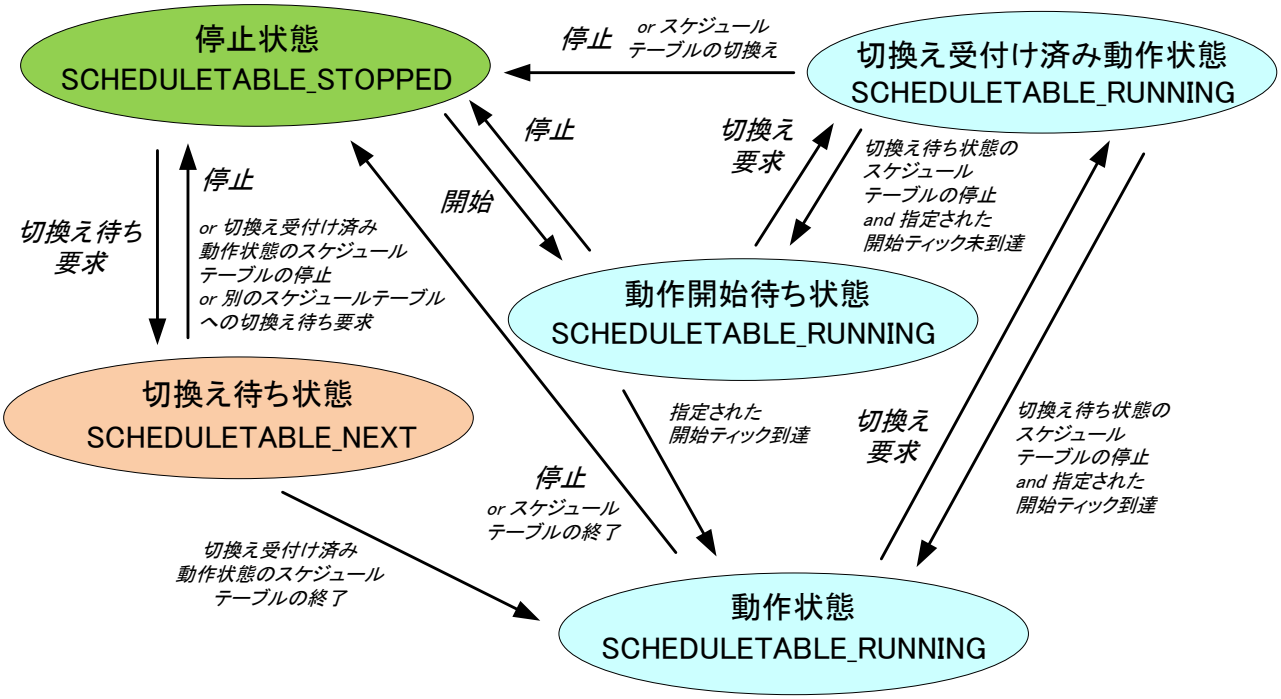


図 2-25 同期なしスケジュールテーブルの状態遷移図

2.13.5.2 暗黙同期スケジュールテーブルの状態遷移

暗黙同期スケジュールテーブルの状態遷移を図 2-26 に示す.

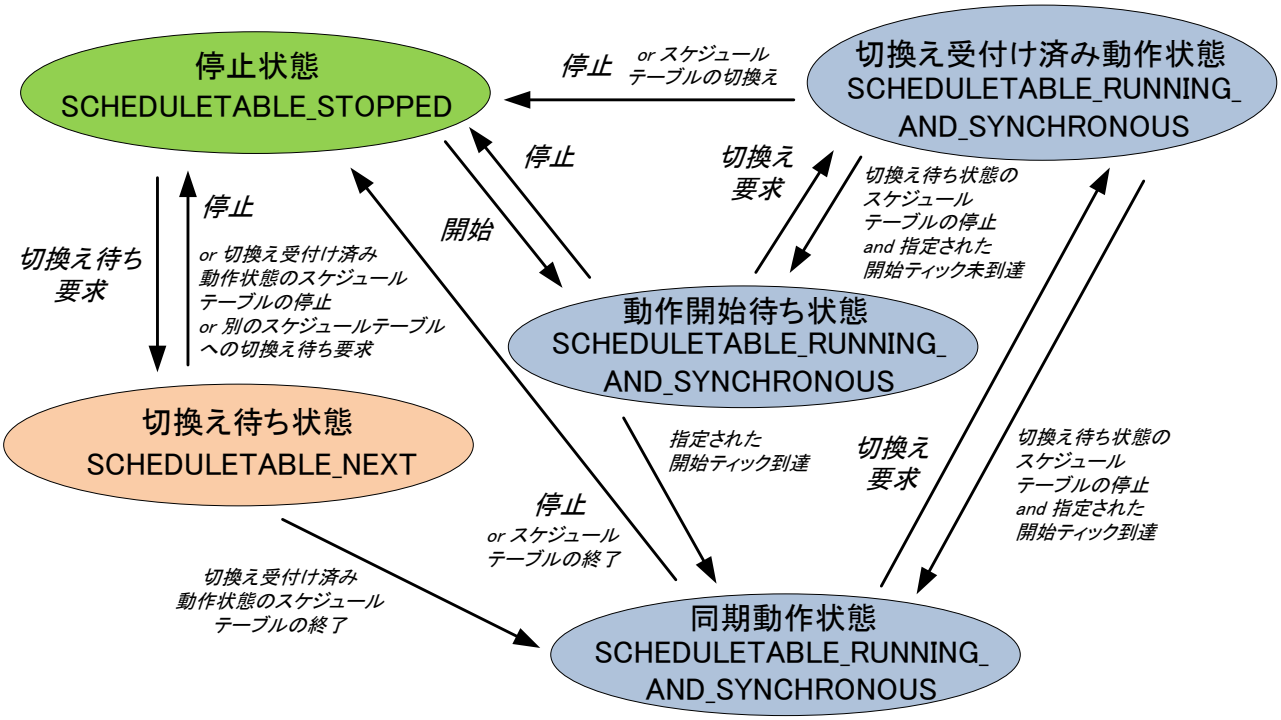


図 2-26 暗黙同期スケジュールテーブルの状態遷移図

2.13.5.3 明示同期スケジュールテーブルの状態遷移

明示同期スケジュールテーブルの状態遷移を図 2-27 に示す.

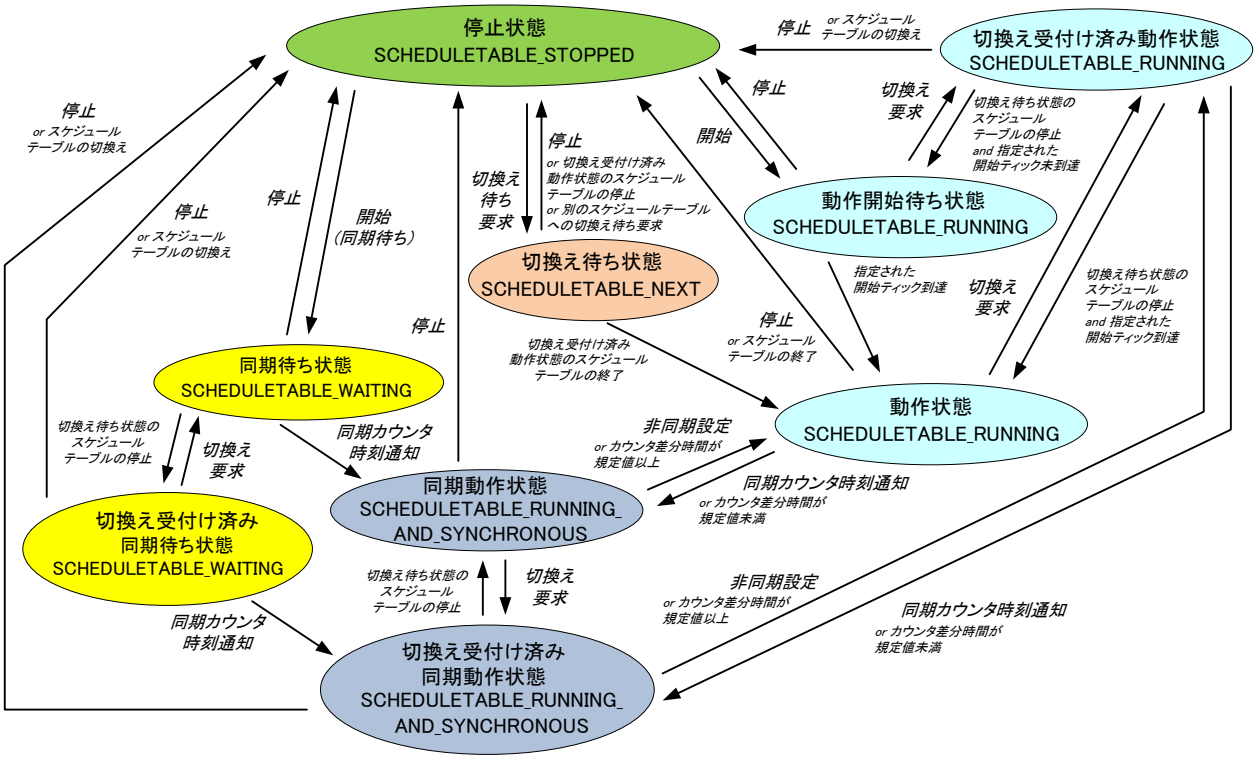


図 2-27 明示同期スケジュールテーブルの状態遷移図

2.13.6 スケジュールテーブルの操作

2.13.6.1 スケジュールテーブルの開始

本 OS は，スケジュールテーブルを開始する機能を提供する．開始方法には以下の 4 種類がある．

マルチコア対応 OS においては，スケジュールテーブルに対する処理を，スケジュールテーブルが所属する OSAP が割付けられているコアにて実行する【OS643】.

絶対時刻での開始

本 OS は，駆動カウンタが指定されたティックと一致した時にスケジュールテーブルを開始するためのシステムサービスとして StartScheduleTableAbs を提供する(詳細は 3.9.34 節を参照). 先頭満了点は駆動カウンタの値が(初期オフセット＋指定されたティック)に達した時である.

相対時刻での開始

本 OS は，駆動カウンタが現在のティックから指定されたティック分進んだ時に，スケジュールテーブルを開始するためのシステムサービスとして StartScheduleTableRel を提供する(詳細は 3.9.33 節を参照)．先頭満了点は駆動カウンタの値が(現在の駆動カウンタのティック＋指定されたティック＋初期オフセット)に達した時である.

スケジュールテーブルの切換え

本 OS は，現在動作中のスケジュールテーブルの停止と，指定されたスケジュールテーブルの開始を不可分に行うためのシステムサービスとして NextScheduleTable を提供する(詳細は 3.9.36 節を参照). OS は現在動作中のスケジュールテーブルを最終遅延まで動作させた後，次に開始するスケジュールテーブルを先頭から開始する【OS414】.

同期待ちでの開始

本 OS は，同期カウンタの値が供給された後，同期カウンタの値が 0 となった時にスケジュールテーブルを開始するためのシステムサービスとして StartScheduleTableSynchron を提供する(詳細は 3.9.37 節を参照).

先頭満了点は駆動カウンタの値が同期カウンタ値通知時から(スケジュールテーブル周期－通知された同期カウンタ値＋初期オフセット)経過した時である.

2.13.6.2 スケジュールテーブルの繰り返し

スケジュールテーブルには 2 種類の動作モードがある．スケジュールテーブルの動作モードはコンフィギュレーション時に静的に指定する【OS413】〔OS_Conf144〕.

単発動作

スケジュールテーブルを 1 周期だけ動作させる．最終満了点の後，最終遅延経過してから停止状態となる【OS009】.

周期動作

スケジュールテーブルを繰り返し動作させる．最終満了点の後，(最終遅延＋初期オフセット)経過してから先頭満了点に戻る【OS194】.

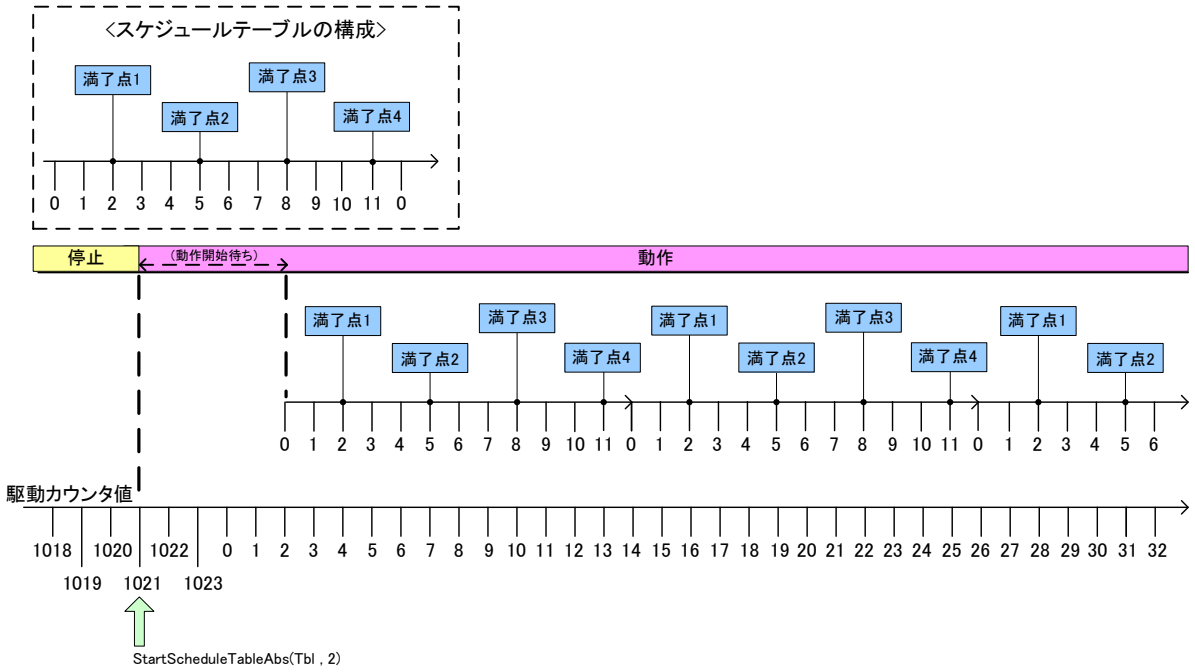


図 2-28 絶対時刻でのスケジュールテーブル動作開始

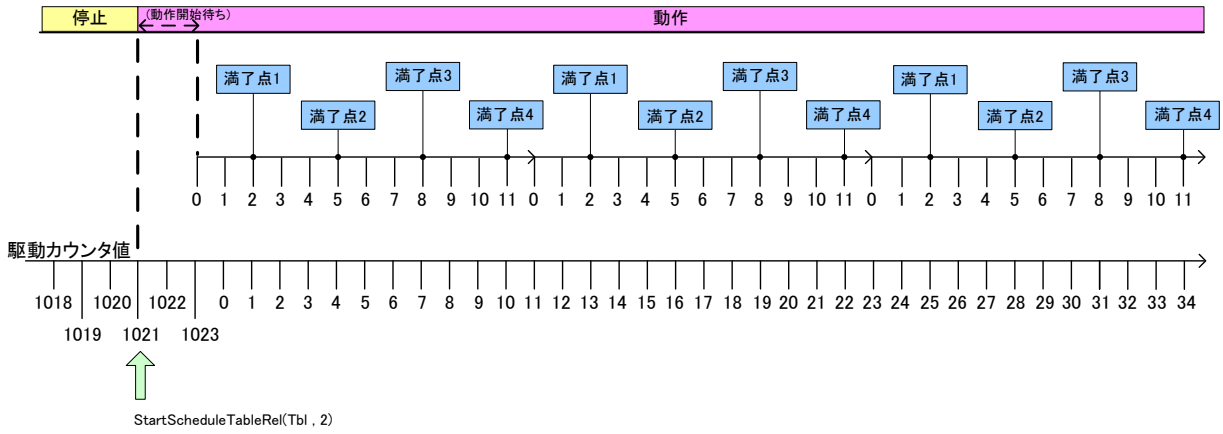


図 2-29 相対時刻でのスケジュールテーブル動作開始

2.13.6.3 開始されたスケジュールテーブルの動作

スケジュールテーブルが開始され，スケジュールテーブル状態が動作状態，同期動作状態の場合，OS は駆動カウンタが 1 ティック進むごとに，スケジュールテーブル時刻を 1 ティック進める【OS411】.

スケジュールテーブルの満了は，接続されたカウンタのティックが，あらかじめ設定した満了点のオフセットに到達した時に発生する．OS は先頭満了点から最終満了点までオフセットが小さいものから順にこれを繰り返す【OS002】.

上記の処理を，OS は複数のスケジュールテーブルを並行して処理する【OS007】.

2.13.6.4 満了点処理

満了点処理では以下のアクションを 1 つ以上実行できる【NOS0884】．これらの処理内容は通常のシステムサービスと同じである【NOS0885】．同一満了点でのアクションは，タスクの起動をすべて行なってからイベントのセットを行う【OS412】.

タスクの起動

満了点処理として，コンフィギュレーション時に指定されたタスクを起動する【OS402】.

マルチコア対応 OS においては，他のコアに割付いたタスクを起動することもできる【OS641】.

イベントのセット

満了点処理として，コンフィギュレーション時に指定されたタスクに対してイベントをセットする【OS403】.

マルチコア対応 OS においては，他のコアに割付いたタスクのイベントをセットすることもできる【OS642】.

2.13.6.5 スケジュールテーブルの停止

OS はスケジュールテーブルを即座に停止するシステムサービスである StopScheduleTable を提供する(詳細は 3.9.35 節を参照).

スケジュールテーブルを停止すると，停止したスケジュールテーブル時刻が最終満了点の前であっても，次回起動時には先頭から実行される【OS428】.

2.13.6.6 スケジュールテーブル状態の取得

OS はスケジュールテーブル状態を取得するシステムサービスである GetScheduleTableStatus 機能を提供する(詳細は 3.9.40 節を参照).

2.13.7 スケジュールテーブルの同期

2.13.7.1 暗黙同期の同期方法

OS は暗黙同期のスケジュールテーブルのために特別な機能を用意しない.

暗黙同期スケジュールテーブルは，常に駆動カウンタと同期していなければならない．OS は以下の 2 つの制約によって暗黙同期スケジュールテーブルと駆動カウンタの同期を保証する.

暗黙同期スケジュールテーブルの周期に対する制約

暗黙同期スケジュールテーブルのスケジュールテーブル周期は接続される駆動カウンタの OsCounterMaxAllowedValue+1 でなくてはならない【OS429】〔OS440〕.

暗黙同期スケジュールテーブルの開始方法に対する制約

暗黙同期スケジュールテーブルは相対時刻での開始は行えない【OS430】〔OS452〕.

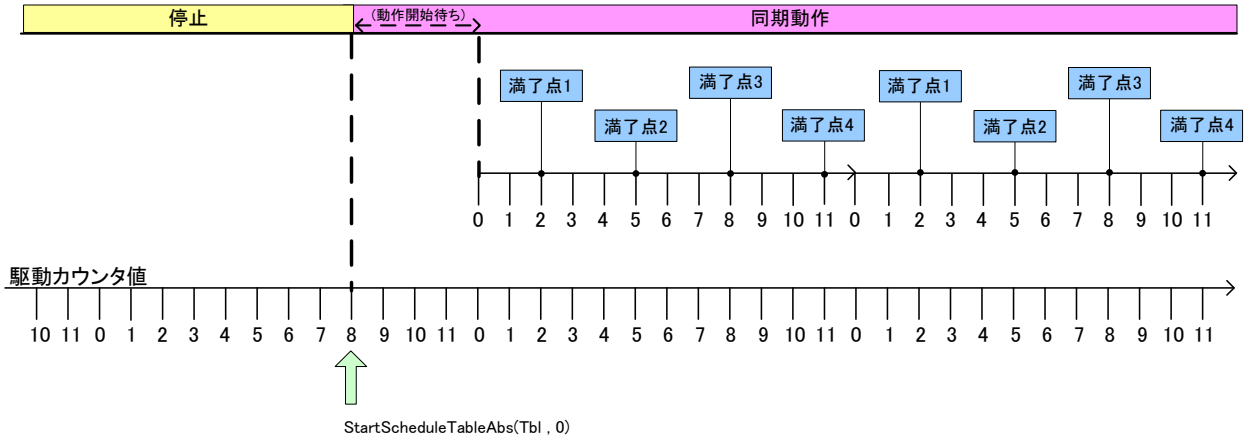


図 2-30 暗黙同期スケジュールテーブルの動作

2.13.7.2 明示同期の同期方法

明示同期では OS オブジェクトとして定義される駆動カウンタと，OS オブジェクトでない同期カウンタの 2 つを同期させるため，OS が同期を行う機能を提供する．

<u>明示同期の前提条件</u>

明示同期スケジュールテーブルでは，以下の条件を満たすことを前提とする．なお，(*)が付く条件は，ユーザが保証すべき条件である．

- 明示同期スケジュールテーブルは駆動カウンタより大きな周期を持たない【OS431】
- 明示同期スケジュールテーブルは同期カウンタと同じ周期を持つ(*)【OS462】
- 同期カウンタは駆動カウンタと同じ分解能を持つ(*)【OS463】

<u>明示同期スケジュールテーブルの開始機能</u>

OS は明示同期スケジュールに対して 2 種類の開始方法を提供する．

- 非同期スタート

  非同期スタートでは同期カウンタ値が不定な状態でスケジュールテーブルの動作を開始する．非同期スタートは，絶対時刻，相対時刻で開始することができ，駆動カウンタの値のみでスケジュールテーブルを開始する．

- 同期スタート

  同期スタートでは同期カウンタ値が供給された状態でスケジュールテーブルの動作を開始する．同期スタートは，OS が提供する同期待ちでの開始により行うことができ，開始後に通知される同期カウンタ値と駆動カウンタ値によってスケジュールテーブルを開始する．この時のスケジュールテーブル状態は同期待ち状態である【OS435】.

<u>同期カウンタ時刻の通知</u>

OS はスケジュールテーブルに対して同期カウンタ時刻を通知するシステムサービスとして SyncScheduleTable を提供する(詳細は 3.9.38 節を参照)【OS013】.

OS は通知された同期カウンタ値によってスケジュールテーブル時刻と同期カウンタの差分を求め，必要であれば同期制御を開始する．

同期制御方式と補正パラメータ

同期制御では各満了点の遅延値を増加，減少させることにより満了点タイミングを変化させ，同期カウンタと補正カウンタの差分を 0 に近づけていくことで同期を行う．

同期制御のために，ユーザは同期制御に必要な補正パラメータを，明示同期スケジュールテーブルのコンフィギュレーション時に静的に指定する．

・ 遅延値の最大増加量

ユーザは満了点に対して，遅延値の最大増加量をティックで指定する【OS416】．(遅延値の最大増加量＋前の満了点からの遅延値)は，スケジュールテーブルが接続されたカウンタのティックの最大値より小さくなければならない【OS437】．遅延値の最大増加量は，スケジュールテーブル周期より小さくなければならない【OS559】．

・ 遅延値の最大減少量

ユーザは満了点に対して，遅延値の最大減少量をティックで指定する【OS415】．(遅延値の初期オフセットー最大減少量)は(初期オフセット＋接続されたカウンタの最小周期)より大きくなければならない【OS436】．

・ 許容誤差値

ユーザは明示同期スケジュールテーブルに許容誤差値(同期状態とみなすスケジュールテーブル時刻－同期カウンタ値差分の絶対値)を 0 からスケジュールテーブル周期の範囲で指定する【OS438】．

図 2-31 補正パラメータの適用例

同期処理

明示同期スケジュールテーブルに同期カウンタ値が通知された後，OS は明示同期スケジュールテーブルの同期制御を開始する【OS417】.

同期カウンタ値が通知されると，OS は現在のスケジュールテーブル時刻と同期カウンタの差分値を算出する【OS206】. 差分値が許容誤差値以下の場合，OS はスケジュールテーブルを同期動作状態に遷移させる【OS418】. 差分値が許容誤差値より大きい場合，OS はスケジュールテーブルを動作状態に遷移させる【OS419】.

スケジュールテーブル時刻が同期カウンタに対して遅れている場合，OS は次の満了点の遅延値を遅延値－MIN (遅延値の最大減少量，進み量)に補正する【OS420】. 一方，スケジュールテーブル時刻が同期カウンタに対して進んでいる場合，OS は次の満了点の遅延値を遅延値＋MIN(遅延値の最大増加量，遅れ量)に補正する【OS421】.

図 2-32 に，最大増加量 2，最大減少量 1 に設定されている場合における，明示同期スケジュールテーブルの動作例を示す.

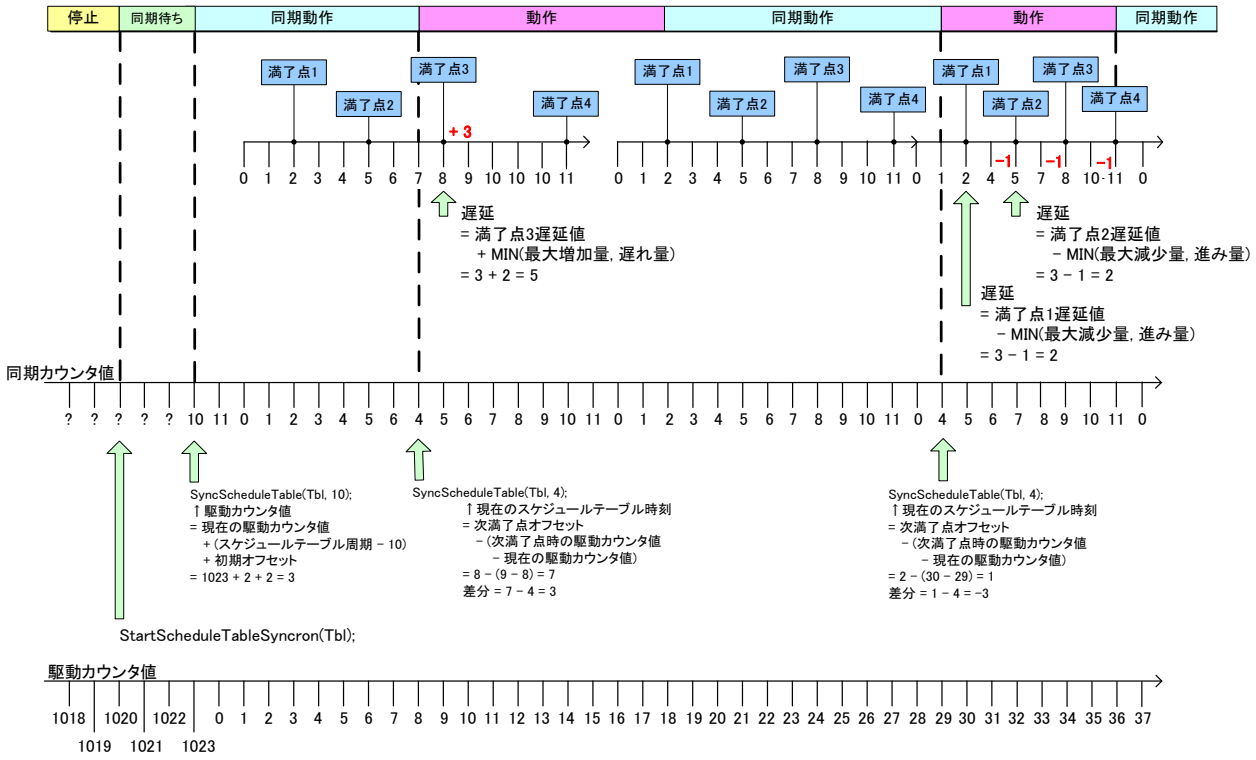


図 2-32 明示同期スケジュールテーブルの動作例

2.13.7.3 同期制御の停止

OS はスケジュールテーブルの同期制御を即座に停止するシステムサービスとして

SetScheduleTableAsync を提供する(詳細は 3.9.39 節を参照).

2.13.8 スケジュールテーブルの設定

スケジュールテーブルに接続するカウンタの指定や満了点の設定は，コンフィギュレーション時に静的に指定する≪OS_Conf141≫.

オブジェクトの OSAP 所属に関する制約

OSAP が再起動する場合，OSAP に所属するカウンタは初期化されてから再スタートするので，他の OSAP に所属するスケジュールテーブルが接続されていると，正しいタイミングで満了処理を実行できない可能性がある．したがって，保護違反時処理の機能レベルが 2 の場合，非信頼 OSAP に所属するカウンタに対して，他の OSAP に所属するスケジュールテーブルを接続することはできない【NOS1131】. 保護違反時処理の機能レベルが 3 の場合，すべてのカウンタに対して，他の OSAP に所属するスケジュールテーブルを接続することはできない【NOS1132】.

オブジェクトのコア割付けに関する制約

スケジュールテーブルは，接続するカウンタと同一のコアに割付けられていなければならない≪OS630≫.

2.14 フックルーチン

フックルーチンは OS 処理内の特定のタイミングで，ユーザ定義の処理を実行する処理単位である．フックルーチンの種別と，それぞれが定義可能な単位(システム，OSAP)の一覧を表 2-13 に示す．

表 2-13 フックルーチンが定義可能な単位

フックルーチン種別	システム定義	OSAP 固有
スタートアップフック	○	○
シャットダウンフック	○	○
プレタスクフック	○	×
ポストタスクフック	○	×
エラーフック	○	○
プロテクションフック	○	×

○：定義可能，×：定義不可能

2.14.1 フックルーチンの種別

2.14.1.1 システム定義のフックルーチン

システム定義のフックルーチンには以下の 6 つの種別がある．

<u>システム定義のスタートアップフック</u>

OS 起動処理におけるスケジューラ起動前に，システム定義のスタートアップフックが呼び出される【COS1110】.

マルチコア対応 OS においては，すべてのコアでシステム定義のスタートアップフックが呼び出される≪OS581≫.

<u>システム定義のシャットダウンフック</u>

OS シャットダウン時もしくは OS が何らかのエラーで異常終了する際に，システム定義のシャットダウンフックが呼び出される【COS1111】.

マルチコア対応 OS においては，すべてのコアでシステム定義のシャットダウンフックが呼び出される≪OS588≫.

<u>システム定義のプレタスクフック</u>

タスクディスパッチによるタスク切換え時において，次に実行されるタスクが動作する前にシステム定義のプレタスクフックが呼び出される．なお，システム定義のプレタスクフックの次に実行されるタスクの状態は実行状態である【COS1163】.

マルチコア対応 OS においては，タスク切換えが発生したコアでのみシステム定義のプレタスクフッ

クが呼び出される【NOS0921】.

システム定義のポストタスクフック

タスクディスパッチによるタスク切換え時において，実行中タスクがプリエンプト，休止，待ち状態に入る前にシステム定義のポストタスクフックが呼び出される．なお，システム定義のポストタスクフック実行中のタスクの状態は実行状態である【COS1162】.

マルチコア対応 OS においては，タスク切換えが発生したコアでのみシステム定義のポストタスクフックが呼び出される【NOS0922】.

システム定義のエラーフック

ユーザが返り値のデータ型が StatusType のシステムサービスを呼出し，実行の結果，返り値が E_OK でない場合に，システム定義のエラーフックが呼び出される【COS1126】．返り値のデータ型が StatusType でないシステムサービスであっても，状況によってエラーフックが呼び出される場合がある【NOS0409】.

また，エラーフックから呼び出されたシステムサービスでエラーが発生した場合にはエラーフックは呼び出されない【COS1127】．そのため，エラーフックでは，システムサービスの返り値でしかエラーを検知することはできない.

ユーザが明示的にシステムサービスを呼び出す以外に，アラーム満了時のアクションでエラーが発生した場合にもエラーフックは呼び出される【COS1129】．また，スケジュールテーブル満了時のアクションでエラーが発生した場合もエラーフックは呼び出される【NOS0655】.

マルチコア対応 OS においては，エラーが発生したコアでのみシステム定義のエラーフックが呼び出される【NOS0923】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，返り値のデータ型が StatusType でないシステムサービスはエラーフックを呼び出さないと規定されている【OS367】．しかし，返り値が無効値となる場合など，エラーフックが呼び出されるべき状況があることから，本仕様では返り値のデータ型が StatusType でないシステムサービスであっても，エラーフックが呼び出される場合があると規定した≪NOS0409≫．どの状況でエラーフックが呼び出されるかどうかは，システムサービス毎に規定した(詳細は 3.9 節を参照).

システム定義のプロテクションフック

保護違反が発生した時に，違反の区別を示すエラーコードをパラメータとして，システム定義のプロテクションフックが呼び出される【NOS0125】.

プロテクションフック実行中に保護違反が発生した場合には，プロテクションフックは呼び出されずに，E_OS_PROTECTION_FATAL を終了要因のエラーコードとして OS シャットダウンを行う【NOS0126】.

マルチコア対応 OS においては，保護違反が発生したコアでのみシステム定義のプロテクションフッ

クが呼び出される【NOS0924】.

AUTOSAR 仕様との違い

AUTOSAR 仕様におけるシステム定義のプロテクションフックの仕様で曖昧だった内容を，本仕様で規定した.

使用上の注意

AUTOSAR 仕様では，シャットダウンフック中の保護違反によりプロテクションフックが起動した場合，シャットダウン処理とプロテクションフックが無限ループする可能性がある．これに対し，本仕様では，シャットダウンフック実行中の保護違反に関する振る舞いを規定したため，1 回目のシャットダウンフック以降に発生したシャットダウン処理では，シャットダウンフックが呼び出されず，無限ループは発生しない≪NOS0631≫≪NOS0879≫≪NOS0633≫.

2.14.1.2 OSAP 固有のフックルーチン

OSAP が有効な場合，システム定義のフックルーチンに加え，以下の 3 種類のフックルーチンが使用できる(詳細は表 2-1 を参照).

OSAP 固有のスタートアップフック

OS 起動処理におけるスケジューラ起動前，システム定義のスタートアップフックの後に，OSAP 固有のスタートアップフックが呼び出される【OS236】．OSAP 固有のスタートアップフックは，スタートアップフックが所属する OSAP のアクセス権を持った状態で実行される【OS226】.

OSAP が複数定義されている場合，OSAP 固有のスタートアップフックの呼出し順は実装定義とする【NOS0128】．ATK2 では，OSAP 固有のスタートアップフックの呼出し順は，コンフィギュレーションファイルの記載順とする【IOS096】.

マルチコア対応 OS においては，すべてのコアでシステム定義のスタートアップフックが呼び出された後に，OSAP 固有のスタートアップフックが，スタートアップフックが所属する OSAP が割付けられいるコアでのみ，それぞれ呼び出される≪OS582≫.

OSAP 固有のシャットダウンフック

OS シャットダウン時もしくは OS が何らかのエラーで異常終了する際に，OSAP 固有のシャットダウンフックの後に，システム定義のシャットダウンフックが呼び出される【OS237】．OSAP 固有のシャットダウンフックは，シャットダウンフックが所属する OSAP のアクセス権を持った状態で実行される【OS225】.

OSAP が複数定義されている場合，OSAP 固有のシャットダウンフックの呼出し順は実装定義とする【NOS0127】．ATK2 では，OSAP 固有のシャットダウンフックの呼出し順は，コンフィギュレーションファイルの記載順とする【IOS097】.

マルチコア対応 OS においては，すべてのコアでシステム定義のシャットダウンフックが呼び出され

る前に，OSAP 固有のシャットダウンフックが，シャットダウンフックが所属する OSAP が割付けられいるコアでのみ，それぞれ呼び出される≪OS586≫.

OSAP 固有のエラーフック

システム定義のエラーフックが呼び出された場合，システム定義のエラーフックの後に，エラーが発生した OS オブジェクトが所属する OSAP の，OSAP 固有のエラーフックが呼び出される【OS246】. OSAP 固有のエラーフックは, エラーフックが所属する OSAP のアクセス権を持った状態で実行される【OS085】.

アラーム満了時のアクションでエラーが発生した場合，システム定義のエラーフックが呼び出された後に，アラームが所属する OSAP の，OSAP 固有のエラーフックが呼び出される【NOS0903】. また，スケジュールテーブル満了時のアクションでエラーが発生した場合も，システム定義のエラーフックが呼び出された後に，スケジュールテーブルが所属する OSAP の，OSAP 固有のエラーフックが呼び出される【NOS0904】.

マルチコア対応 OS においては，エラーが発生した処理単位が割付けられいるコアで，エラーが発生した処理単位が所属する OSAP 固有のエラーフックが呼び出される【NOS0925】.

OSAP 固有のフックルーチンの命名規則

OSAP ごとに定義できるフックルーチンと命名規則を以下に示す．<App>は OSAPID である.

- スタートアップフック：StartupHook_<App>【OS060】
- シャットダウンフック：ShutdownHook_<App>【OS112】
- エラーフック：ErrorHook_< App >≪OS246≫

OSEK 仕様との違い

OSEK 仕様では，フックルーチンは特殊なコンテキストで実行されると規定している．本仕様では，OS 処理レベルを定義したため, フックルーチンは OS 処理レベルで動作すると規定した≪NOS0243≫. また，OSEK 仕様では，フックルーチンの機能は標準化されないので，移植性がないと規定されている【COS1107】．本仕様では，処理レベルや実行タイミングを規定しているので，削除した.

2.14.2 フックルーチンの処理レベル

フックルーチンは OS から呼びだされ，OS 処理レベルで動作する【NOS0243】．プロテクションフックの処理レベルは，保護違反を発生する要因ごとに定義される(詳細は 2.22.1 節を参照).

2.14.3 フックルーチンで行う処理

フックルーチン内ではユーザの任意の処理を実行することができる【COS1106】．ただし，フックルーチン内で使用できるシステムサービスは制限されている(詳細は表 3-1 を参照).

2.14.4 フックルーチンの指定

フックルーチンの動作の有無は，コンフィギュレーション時に静的に指定する【COS1109】．フックルーチンを有効としているにも関わらず，フックルーチンのアドレスが不正(NULL など)の場合は，フックルーチンを無効に設定した場合と同等の処理を行う【IOS177】.

2.15 エラー処理

エラー処理は，OS が提供するシステムサービスにおける異常を検出する機能である．
OS はエラーを起こした処理単位に対し，2 種類のエラー処理方法を提供する【COS1167】.

- 集中エラー処理
  OS が提供するシステムサービスでエラーが発生した時に，OS が提供するエラーフックを用いてエラー処理を行う方法を，集中エラー処理と呼ぶ.

- 分散エラー処理
  OS が提供するシステムサービスでエラーが発生した時に，システムサービスからの返り値を用いてエラー処理を行う方法を，分散エラー処理と呼ぶ.

発生したエラーに対する処理方法はユーザによって決定される【COS1120】.

2.15.1 エラー種別

本 OS ではエラーを 2 つに分類する.

アプリケーションエラー

OS は要求されたサービスを正常に処理しないが, OS の内部状態は正常である場合をアプリケーションエラーと呼ぶ.
この場合，OS はユーザに集中エラー通知を行った後，分散エラー処理のためのエラーコードをアプリケーションに返す．この時の出力パラメータの値は保証されない【COS1125】.

フェイタルエラー

OS が OS の内部状態が正常であることを保証しない場合をフェイタルエラーと呼ぶ．この場合，OS は OS シャットダウンを行う【COS1220】．この場合，E_OS_SYS_ASSERT_FATAL を終了要因のエラーコードとして OS シャットダウンを行う【NOS0690】.
マルチコア対応 OS において，いずれかのコアでフェイタルエラーが発生した場合，同期 OS シャットダウンを行う【OS762】.

2.15.2 エラーコード種別

OS は，エラーが発生した際にエラー原因を特定するためのエラーコードを，アプリケーションに提供する【COS1122】. エラー発生時に返すエラーコードを, 本 OS では以下の 2 つに区分する【COS1123】.

標準エラー

常に検出するエラーを標準エラーと呼ぶ．デバッグ済みのアプリケーションでも起こりうるエラーである．

拡張エラー

開発中にのみ検出する追加エラーを拡張エラーと呼ぶ．テスト時やデバッグされていないアプリケーション向けに使われるエラーである．

使用するエラーコード種別の選択基準

拡張エラーはテスト段階で検出されることが多いため，発生したすべての拡張エラーを取り除いた後は標準エラーのみの検出に変更することにより，実行速度の向上と省メモリ化を図ることができる．

ただし，SC3，SC4 では常に拡張エラーの検出を行う【OS327】．マルチコア対応した SC1，SC2 では，OSAP がサポートされるが，常に拡張エラーの検出を行う必要はなく，標準エラーのみ検出するよう指定できる【OS763】〔OS_Conf046〕.

2.15.2.1 エラーコード種別の指定

拡張エラーを検出するかどうかは，コンフィギュレーション時に静的に指定する≪OS_Conf046≫. 拡張エラーを検出しないコンフィギュレーション時に，拡張エラーが発生する要因となる処理を実施した場合の動作は保証されない【NOS0348】.

2.15.3 システムサービスにおけるエラーチェック

ほとんどのシステムサービスは分散エラー処理のために返り値を返す．システムサービスが正常に終了した場合は E_OK を返し，何らかのエラーが検出された場合は E_OK 以外の値を返す【COS1208】.

本 OS がエラーチェックを行う順序は実装定義である【COS1212】．また，複数のエラーが検出された場合に返るエラーコードは実装定義である【COS1213】.

C1ISR から発行したシステムサービスにおいて，エラーが発生した場合の動作は保証されない【NOS0868】.

使用上の注意

システムサービスが返す E_OK でないエラーコードの中には，OS が正常に動作中でも発生し，警告とし扱うことができるものがある(具体的なエラーコード名は各システムサービスの返り値欄に記載).

2.15.4 エラーフック

システムサービスの返り値が E_OK でない場合，システム定義のエラーフックが有効になっていれば，OS は集中エラー処理のためにシステム定義のエラーフックを呼び出す【COS1214】．また，返り値のデータ型が StatusType でないシステムサービスであっても，状況によってエラーフックが呼び出される場合がある≪NOS0409≫．SC3，SC4 で，OSAP 固有のエラーフックが有効な場合，そのシステム定義のエラーフックから戻った後 OSAP 固有のエラーフックを呼び出す≪OS246≫．

2.15.4.1 エラーを起こしたシステムサービスの情報の取得

エラーフックでの効率的なエラー処理のために，OS はエラーを起こしたシステムサービスの情報を取得する機能を提供する【COS1219】．なお，効率的な実装を許容するためにエラー情報構造の詳細は定義しない【COS1218】.

- システムサービス ID の取得
  OS はエラーを引き起こしたシステムサービスの ID をエラーフック内で取得する機能を提供する.

- システムサービスパラメータの取得
  OS はエラーを引き起こしたシステムサービスのパラメータをエラーフック内で取得する機能を提供する.

<u>エラーフック内でのシステムサービス ID の取得方法</u>

システム定義のエラーフック，OSAP 固有のエラーフック内では，OSErrorGetServiceId()マクロにより，エラーを引き起こしたシステムサービス ID を取得することができる【COS1130】〔COS1219〕．システムサービス ID は OSServiceIdType 型で，OSServiceId_*xxxx* の形で定義される(*xxxx* には各システムサービス名が入る)【COS3835】.

例えば ActivateTask でエラーが発生した場合 OSErrorGetServiceId()マクロは OSServiceId_ActivateTask を返す.

<u>エラーフック内でのシステムサービスパラメータの取得方法</u>

システム定義のエラーフック，OSAP 固有のエラーフック内では，OSError_Name1_Name2 マクロによりエラーを引き起こしたシステムサービスのパラメータを取得することができる【COS3836】〔COS1219〕．Name1 はシステムサービス名である【COS1131】．Name2 はシステムサービスのパラメータ名である【COS1132】.

例えば SetRelAlarm の場合，

- OSError_SetRelAlarm_AlarmID()
- OSError_SetRelAlarm_increment()
- OSError_SetRelAlarm_cycle()

によってシステムサービスパラメータを取得することができる．

システムサービスパラメータの中で，第 2 引数以降や，オブジェクト ID でないパラメータが，取得できるかは実装定義である【COS1168】．ATK2 では，第 2 引数以降や，オブジェクト ID でないパラメータも含めた，すべてのシステムサービスパラメータを取得可能とする【IOS030】．

2.15.4.2 エラーフックの指定

エラーフックの動作の有無は, コンフィギュレーション時に静的に指定する【OS439】〔OS_Conf036〕.

システムサービス ID の取得機能とシステムサービスパラメータの取得機能の使用有無は，コンフィギュレーション時に静的に指定する【COS1169】.

2.15.4.3 エラーフックの実装例

エラーフックの実装例を図 2-33 に示す.

```
void
ErrorHook(StatusType Error)
{
      switch(Error){
      case E_OS_ACCESS:
         ・・・
      case E_OS_STATE:
         ・・・
      case E_OS_ID:
         ・・・
      }

      switch(OSErrorGetServiceId()){
      case OSServiceId_ActivateTask:
         ・・・
      case OSServiceId_SetRelAlarm:
         ・・・
           CallingTask = GetTaskID();
           Param1 = OSError_SetRelAlarm_AlarmID();
           Param2 = OSError_SetRelAlarm_increment();
           Param3 = OSError_SetRelAlarm_cycle();
              ・・・
      case OSServiceId_StartScheduleTableRel:
              ・・・
      }
         ・・・
}
```

図 2-33 エラーフック実装例

2.16 スタックモニタリング

スタックモニタリングはタスク, C2ISR のスタックオーバーフローを検出する機能である【OS067】. スタックモニタリングはターゲットが持つハードウェア機能の可能な範囲で，スタックポインタを監視する.

2.16.1 スタックオーバーフローの検出方法

スタックオーバーフローの検出方法の代表的なものとして，スタックポインタチェック方式，マジックナンバーチェック方式，スタック残量チェック方式がある．実装としてどの方式を使用するか，あるいは別の方式を使用するかは実装定義とする【NOS0358】.

2.16.1.1 スタックポインタチェック方式

ソフトウェアのみによるスタックポインタチェック方式は，スタックポインタが，対象処理単位のスタック領域の範囲を超えていないかチェックする方式である【NOS0356】. スタックポインタをチェックする時点で，スタックオーバーフローが発生している場合に検出可能である【NOS0365】.

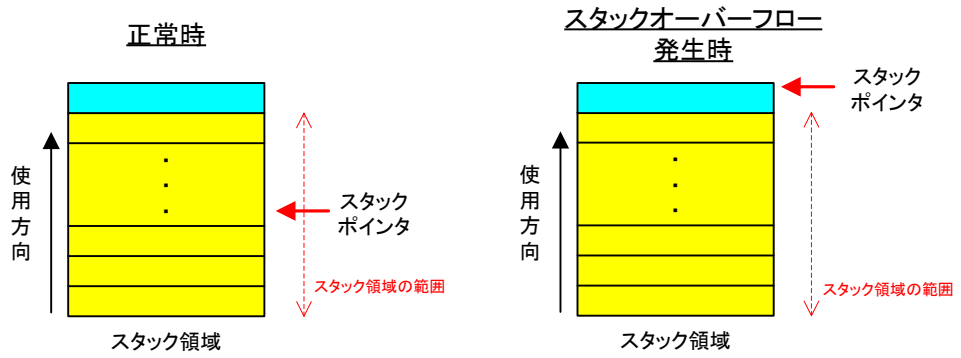


図 2-34 スタックポインタチェック方式

2.16.1.2 マジックナンバーチェック方式

マジックナンバーチェック方式は，OS 起動時に各処理単位のスタック領域の終端，もしくはスタック領域に隣接するメモリ領域に, 特殊な値(マジックナンバー)を書き込んでおき, 処理単位切換え時に, マジックナンバーが書き変わっていないかチェックする方式である【NOS0357】. 処理単位実行時にスタックオーバーフローが発生して，スタックの領域の境界に書き込んだ場合に検出可能である【NOS0366】.

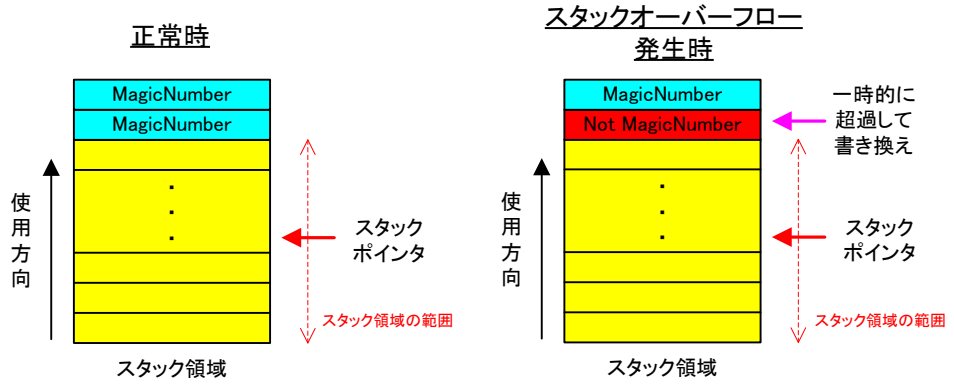


図 2-35 マジックナンバーチェック方式

マジックナンバー領域のサイズは実装定義とする【NOS0359】. マジックナンバー領域を，コンフィギュレーション時に設定するタスク，C2ISR のスタックサイズに含めるかは，実装定義とする【NOS0360】. マジックナンバーの値は，実装定義とする【NOS0361】.

2.16.1.3 スタック残量チェック方式

スタック残量チェック方式は，多重割込みによる C2ISR 起動時に，現在のスタックポインタと，コンフィギュレーション時に指定した C2ISR のスタックサイズを使用して，残りのスタックサイズが足りているかをチェックする方式である【NOS1069】. ある低割込み優先度の C2ISR を実行中に，中割込み優先度の CI2SR が起動した場合，中割込み優先度以上の CI2SR が使用するスタックサイズの合計値が，C2ISR 用スタック領域に残っていない場合，低割込み優先度の C2ISR がスタックを指定した以上に使用したと判断し，スタックオーバーフローとする(図 2-36 参照).

異なる割込み優先度の C2ISR 用スタックを 1 つのスタック領域で確保する場合，スタックポインタチェック方式とマジックナンバー方式では，確保したスタック領域全体に対してスタックオーバーフローした場合しか検出することができない. そこで，スタック残量チェックにより，多重割込み発生時，それまで実行していた C2ISR がスタックを使い過ぎていないかのチェックが可能となる.

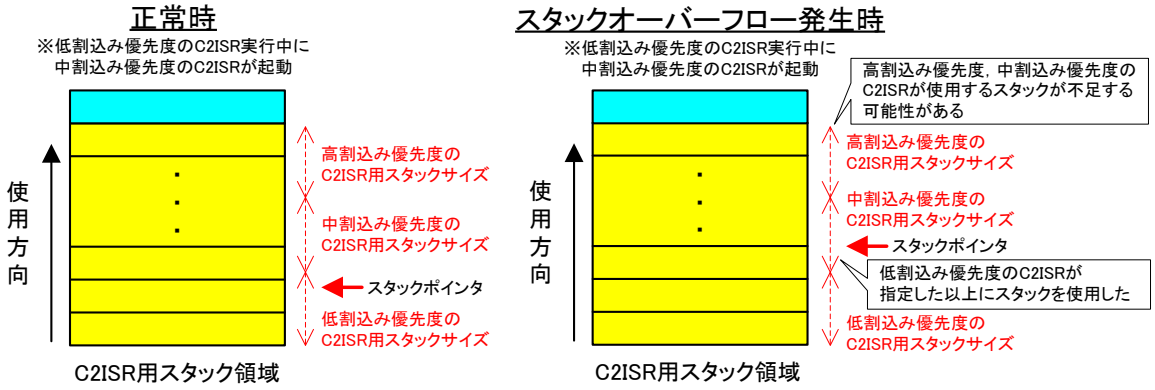


図 2-36 スタック残量チェック方式

ATK2 では，各方式によってスタックオーバーフローを検出可能な状況が異なるため，スタックポインタチェック方式，マジックナンバーチェック方式，スタック残量チェック方式の 3 つを併用する【IOS007】．ただし，併用した場合でも以下のケースでは，スタックオーバーフローを検出できない．

- スタックオーバーフローしたが，マジックナンバー領域を書き換えずに，スタック領域の範囲内に戻った場合．
- マジックナンバー領域とは異なる領域を書き換えた場合．

マジックナンバーチェック方式におけるマジックナンバー領域のサイズは 4byte とする【IOS008】．マジックナンバーチェック方式におけるマジックナンバー領域は，各処理単位に割り当てられたスタックの最後に確保する．すなわち，各処理単位が使用可能なスタックのサイズは，割り当てサイズ - 4byte となる【IOS009】．マジックナンバーの値は，ターゲット定義とする【IOS130】．

2.16.2 スタックモニタリングの動作タイミング

OS は，実装定義のチェック可能な箇所でスタックオーバーフローが発生していないかのチェックを行う【NOS1070】．そのため，チェック時に既にスタックポインタが範囲内に収まっている場合などに，エラーの検知が行えない可能性がある．

SC3，SC4 では，非信頼タスク用スタック，非信頼 C2ISR 用スタックに対してスタックオーバーフローが発生した場合は，メモリ保護機能により，即座にメモリ保護違反が発生する．信頼タスク用スタック，非信頼タスク用システムスタック，信頼 C2ISR 用スタック，非信頼 C2ISR 用システムスタックに対しては，SC１，SC2 と同様のスタックモニタリングを行う【NOS1071】．

ATK2 では以下のタイミングでスタックモニタリングを行う．

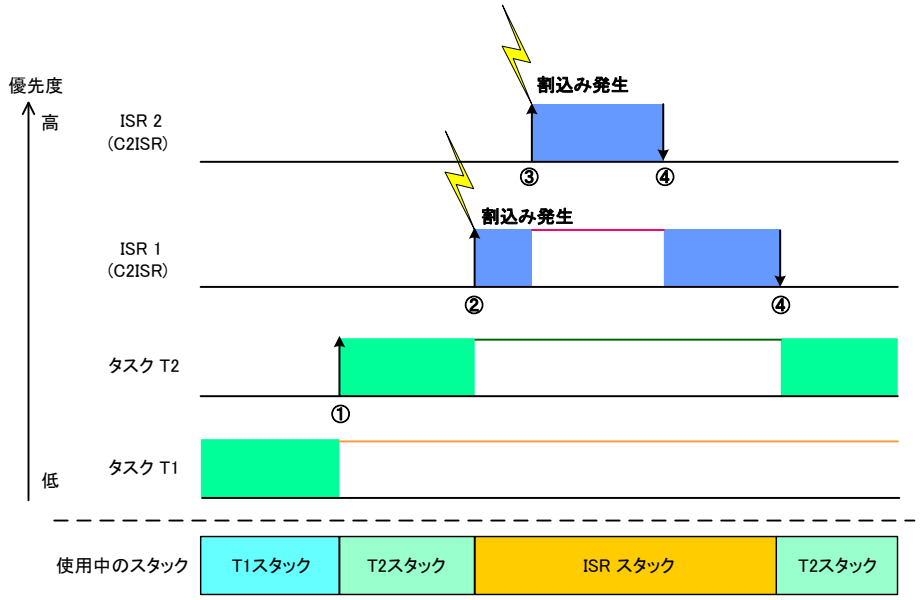


図 2-37 ATK2 におけるスタックモニタリングのタイミング

①： タスクディスパッチによりタスクが切り換わるタイミングで，実行状態であったタスク(タスク T1)が，設定したスタックサイズを超過していないかチェックする【IOS010】．①は，スタックポインタチェック方式，マジックナンバーチェック方式を使用する【IOS220】．

②： 割込み発生時，実行状態であったタスク(タスク T2)が，設定したスタックサイズを超過していないかチェックする【IOS011】．②は，スタックポインタチェック方式，マジックナンバーチェック方式を使用する【IOS221】．

③： 多重割込み発生時，実行状態であった C2ISR(ISR 1)が，設定したスタックサイズを超過していないかチェックする【IOS012】．③は，スタック残量チェック方式，マジックナンバーチェック方式を使用する【IOS222】．

④： 割込み終了時，実行状態であった C2ISR(ISR 2/ISR 1)が，設定したスタックサイズを超過していないかチェックする【IOS013】．C2ISR の終了は return によって行うので，スタックポインタが元に戻ることから，④は，マジックナンバーチェック方式のみ使用する【IOS014】．

③，④のタイミングに対しては，SC1，SC2 および SC3，SC4 のメモリ保護機能の機能レベル 1，2 では，システム定義のフックルーチン，信頼フック，および C2ISR が使用するスタックは 1 つの領域で確保するため，これらすべての処理単位で使用するスタック領域に対してスタックモニタリングを実行する【IOS060】．したがって，システム定義のフックルーチンがスタックオーバーフローを起こした場合にも，③，④のマジックナンバーチェック方式によって，スタックオーバーフローを検出する可能性がある．

SC3，SC4 のメモリ保護機能の機能レベル 3 では，C2ISR 毎にスタック領域を確保するため，C2ISR

毎にスタックモニタリングを実行する(現状，機能レベル 3 は未実装)【IOS061】.

また，SC3，SC4 では，信頼関数終了時に，信頼タスク用スタック，非信頼タスク用システムスタック，信頼 C2ISR 用スタック，非信頼 C2ISR 用システムスタックに対して，スタックモニタリングを行う【IOS142】．ただし，信頼関数終了時のスタックモニタリングは，マジックナンバーチェック方式のみとする【IOS162】.

実行状態のタスクが TerminateTask もしくは ChainTask を発行することにより，休止状態へと遷移する場合も，①のタイミングと同様にスタックモニタリングを実行する【IOS163】．プロテクションフックが有効の場合で，タスク終了時にスタックオーバーフローを検出した場合，ポストタスクフックは呼び出されずに，プロテクションフックが呼び出される【IOS164】．このプロテクションフックで，タスクを強制終了した場合，ポストタスクフックは呼び出されない【IOS165】.

2.16.3 スタックオーバーフロー時の動作

プロテクションフックが有効な場合，スタックオーバーフローを検出すると，OS は E_OS_STACKFAULT をパラメータとしてプロテクションフックを呼び出す【OS396】．プロテクションフックが無効な場合，スタックオーバーフローを検出すると，OS は E_OS_STACKFAULT をパラメータとして ShutdownOS を呼び出す【OS068】.

2.16.4 スタックモニタリングの指定

スタックモニタリングの使用有無はコンフィギュレーション時に静的に指定する.

2.17 タイミング保護

タイミング保護機能は，タスク，C2ISR が指定された時間制約を守れているかを監視する機能である．

本仕様におけるタイミング保護機能は，今後 AUTOSAR 仕様から大きく変更する可能性がある．そのため，本仕様におけるタイミング保護機能は，ベースとしたバージョンの AUTOSAR 仕様に追従していない部分が含まれる．

2.17.1 監視対象となる処理単位

SC2 の場合，指定されたタスクと C2ISR がタイミング保護の対象となる【OS397】.

SC4 の場合，非信頼 OSAP のすべてのタスク，C2ISR が保護対象となる【OS028】. また，信頼 OSAP の中で指定されたタスク，C2ISR も保護対象となる【OS089】.

C1ISR を実行中に保護違反が発生した場合，OS の一貫性を保つことができないため，C1ISR はタイミング保護の対象としない．また，OS 起動前の処理単位はタイミング保護の対象としない.

2.17.2 監視対象となる時間

タイミング保護機能の対象となる時間を以下に示す.

2.17.2.1 実行時間

それぞれの処理単位に許された実行時間の最大値を実行時間バジェットと呼ぶ.

タスク実行時間

タスクが実行状態となってから休止状態もしくは待ち状態になるまでの時間をタスク実行時間と呼ぶ．タスク実行時間には，エラー処理(エラーフックを含む)，プレタスクフック，ポストタスクフック，システムサービスの実行時間を含む．タスクが実行可能状態となっている時間，C2ISR に割込まれている時間，プロテクションフックの実行時間はタスク実行時間に含まれない．C1ISR は OS の管理外であり，実行時間を OS で管理できないため，C1ISR に割込まれた場合の時間は，割込まれたタスクの実行時間に含まれる．(図 2-38 を参照)

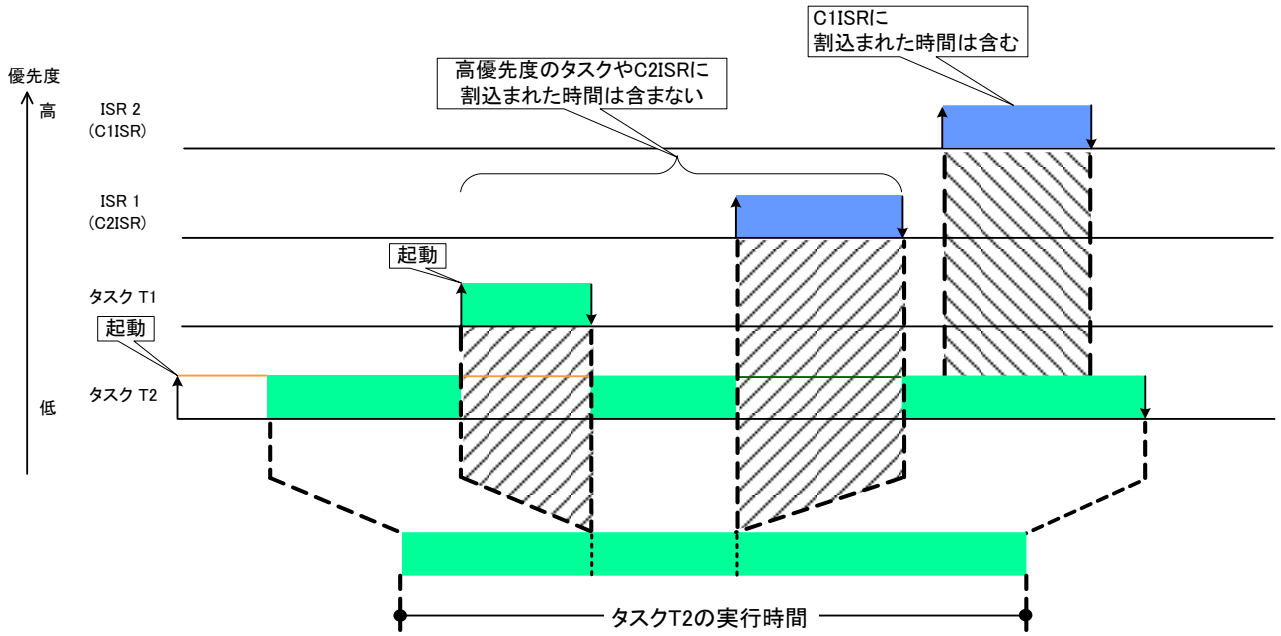


図 2-38 タスク実行時間

C2ISR 実行時間

C2ISR が発生してから終了するまでの時間を C2ISR 実行時間と呼ぶ．C2ISR 実行時間にはエラー処理，システムサービスの実行時間を含む．より高い優先度の C2ISR に割込まれた場合の時間，プロテクションフックの実行時間は，C2ISR 実行時間に含まれない．C1ISR は OS の管理外であり，実行時間を OS で管理できないため，C1ISR に割込まれている時間は，割込まれた C2ISR の実行時間に含む．(図 2-39 を参照)

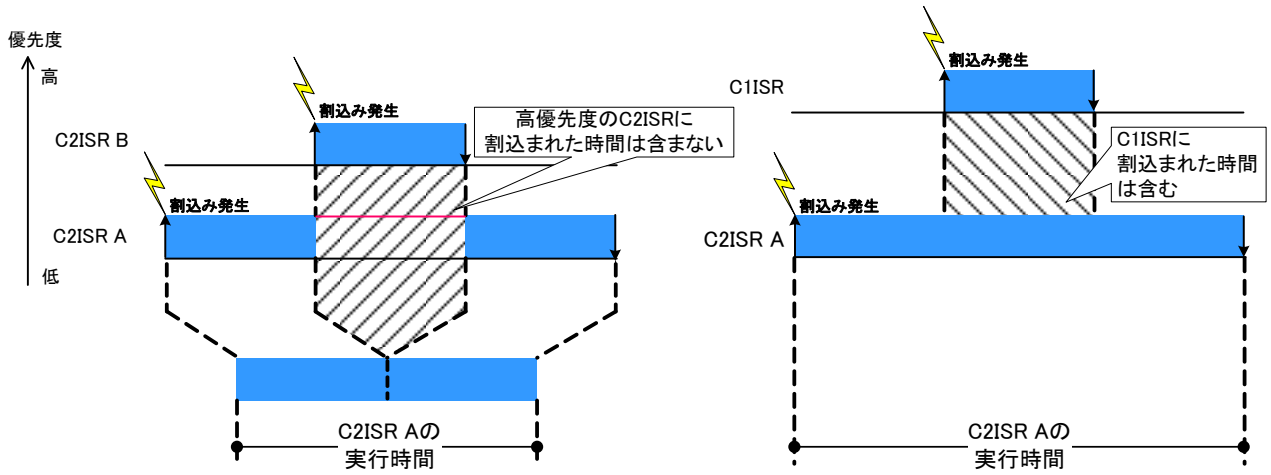


図 2-39 C2ISR 実行時間

2.17.2.2 到着時間

それぞれの処理単位に許された到着時間の最小値をタイムフレームと呼ぶ.

基本タスク到着時間

基本タスク到着時間は基本タスクの起動間隔である．起動には休止状態から実行可能状態への遷移とタスク実行中の多重起動の双方が含まれる．(図 2-40 を参照)

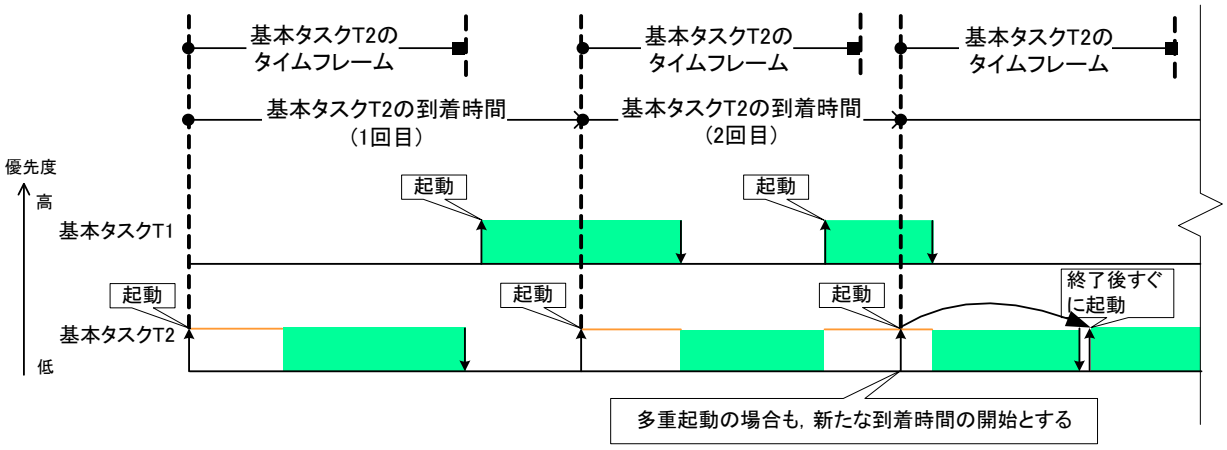


図 2-40 基本タスクの到着時間

拡張タスク到着時間

拡張タスク到着時間は拡張タスクの起動もしくは待ち解除間隔である．(図 2-41 を参照)

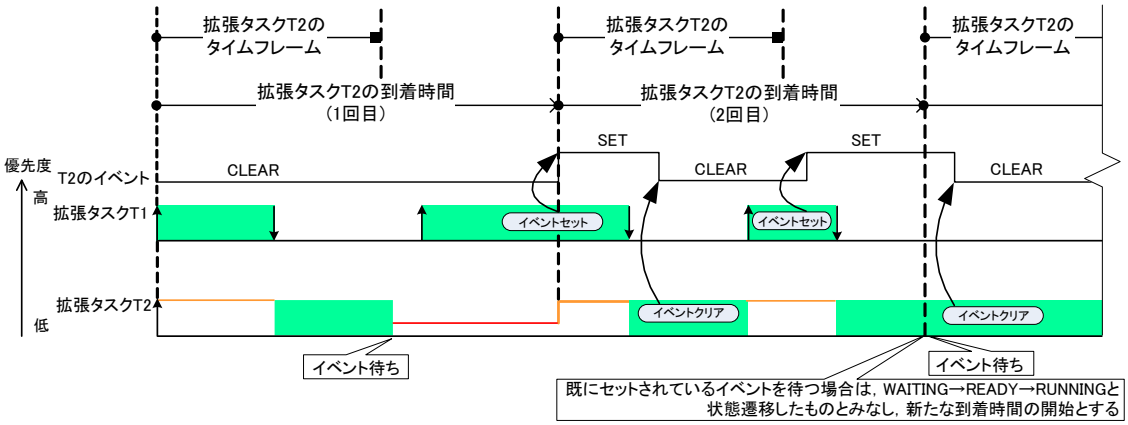


図 2-41 拡張タスクの到着時間

C2ISR 到着時間

C2ISR の到着時間は C2ISR の起動間隔である．(図 2-42 を参照)

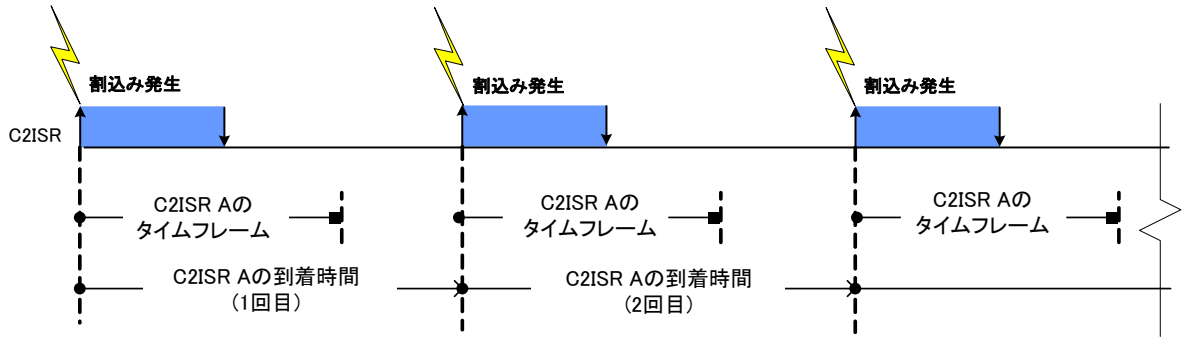


図 2-42 C2ISR 到着時間

2.17.2.3 ロック時間

リソース占有時間

タスク，C2ISR がリソースを占有する時間をリソース占有時間と呼ぶ．ただし，より高い優先度のタスクや C2ISR が実行されている時間は，リソース占有時間に含まれない．(図 2-43 を参照)

タスク，C2ISR のリソース占有時間の最大値をリソース占有時間バジェットと呼ぶ．

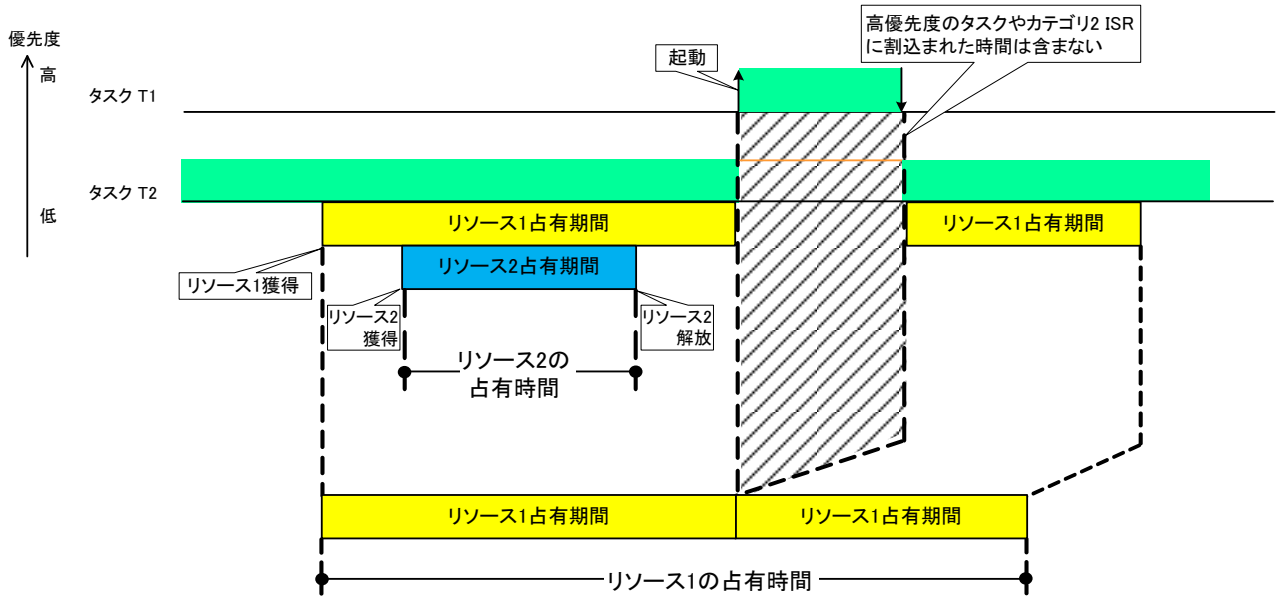


図 2-43 リソース占有時間

タイミング保護の機能レベル 3 では，リソースを獲得する処理単位毎に，リソース占有時間を定義可能であるが，機能レベル 2 では，処理単位に関係なくリソース毎に決まったリソース占有時間を定義する≪NOS0209≫．

割込み禁止時間

タスク，C2ISR が割込みを禁止してから許可するまでの時間を割込み禁止時間と呼ぶ．割込み禁止時間には，C2ISR の割込みを禁止する OS 割込み禁止時間と，すべての割込みを禁止する全割込み禁止時間がある．（図 2-44 を参照）

タスク，C2ISR の OS 割込み禁止時間の最大値を OS 割込み禁止時間バジェットと呼び，全割込み禁止時間の最大値を全割込み禁止時間バジェットと呼ぶ．

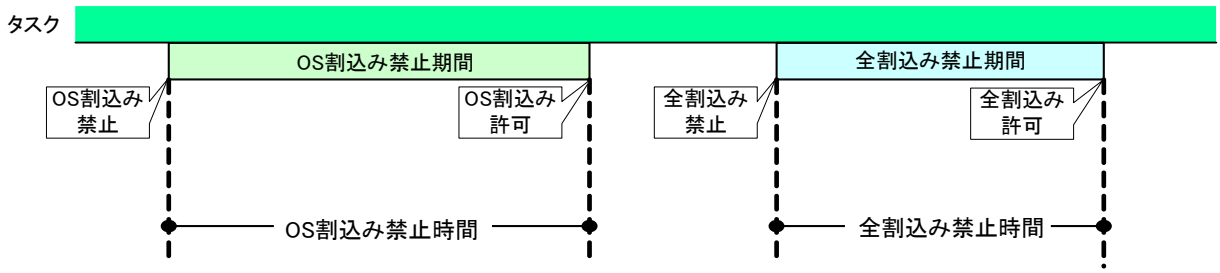


図 2-44 割込み禁止時間

2.17.3 タスクに対するタイミング保護機能

タスク実行時間の監視

タスクの実行時間バジェットを超えてタスクが実行状態を維持し続けていないか，OS が監視する．タスクが休止状態か待ち状態になった時，OS はタスク実行時間を 0 にリセットする【OS473】．タスクの実行時間がタスク実行時間バジェットを越えた時，OS は E_OS_PROTECTION_TIME_TASK をパラメータとして保護違反時処理を呼び出す【NOS0181】．

到着時間(起動および待ち時間解除)間隔の監視

タスクが起動された時を，タイムフレームの始点とする【OS469】．同様に，タスクが待ち解除された時も，タイムフレームの始点とする【OS472】．タスクの到着時間がタスクのタイムフレーム以上であることを，OS が監視する【OS465】．タスクの起動もしくは待ち解除間隔がタイムフレーム未満である場合，OS はタスクの起動もしくは待ち解除は行なわず，E_OS_PROTECTION_ARRIVAL_TASK をパラメータとして保護違反時処理を呼び出す【NOS0182】．このとき，タスクは起動もしくは待ち解除されないため，タイムフレームの始点は更新しない【NOS0297】．

タスクの状態遷移とタイミング保護の対応を図 2-45 に示す．

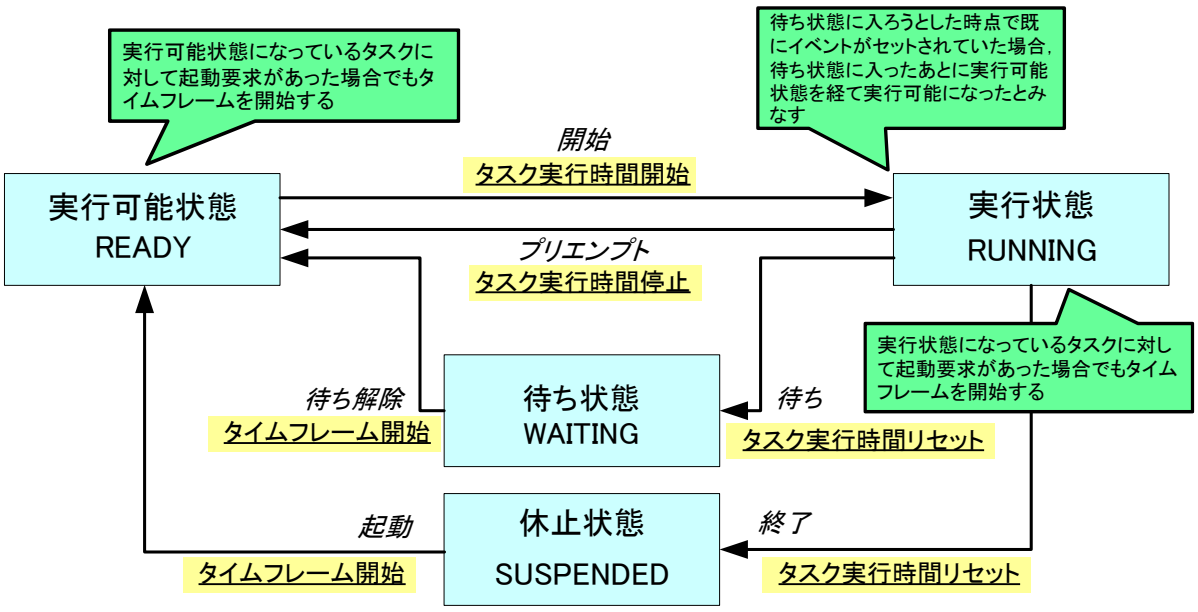


図 2-45 タスクの状態遷移とタイミング保護の対応

リソース占有時間の監視

タスクがリソース占有時間バジェットを超えてリソースを占有し続けないか，OS が監視する．タスクがリソース占有時間バジェットを越えてリソースを占有した時，OS は E_OS_PROTECTION_LOCKED_RESOURCE をパラメータとして保護違反時処理を呼び出す

【NOS0183】.

OS 割込み禁止時間の監視

タスクが OS 割込み禁止時間バジェットを超えて OS 割込みを禁止し続けないか，OS が監視する．タスクが OS 割込み禁止時間バジェットを超えて OS 割込みを禁止した時，OS は E_OS_PROTECTION_LOCKED_OSINT をパラメータとして保護違反時処理を呼び出す【NOS0184】.

全割込み禁止時間の監視

タスクが全割込み禁止時間バジェットを超えてすべての割込みを禁止し続けないか，OS が監視する．タスクが全割込み禁止時間バジェットを超えてすべての割込みを禁止した時，OS は E_OS_PROTECTION_LOCKED_ALLINT をパラメータとして保護違反時処理を呼び出す【NOS0185】.

保護違反時の動作

タスクがタスク実行時間，起動および待ち解除間隔，リソース占有時間，割込み禁止時間の時間制約に違反した場合に，保護違反時処理に渡すパラメータの一覧を表 2-14 に示す.

表 2-14 タスクの違反処理一覧

タスクの違反処理	保護違反時処理に渡すパラメータ
タスクの実行時間が実行時間バジェットを超えた.	E_OS_PROTECTION_TIME_TASK ≪NOS0181≫
タスクの到着時間(起動もしくは待ち時間解除間隔)が，タイムフレーム未満である.	E_OS_PROTECTION_ARRIVAL_TASK ≪NOS0182≫
タスクがリソース占有時間バジェットを越えてリソースを占有した.	E_OS_PROTECTION_LOCKED_RESOURCE ≪NOS0183≫
タスクが OS 割込み禁止時間バジェットを超えて OS 割込みを禁止した.	E_OS_PROTECTION_LOCKED_OSINT ≪NOS0184≫
タスクが全割込み禁止時間バジェットを超えてすべての割込みを禁止した.	E_OS_PROTECTION_LOCKED_ALLINT ≪NOS0185≫

保護違反タスク

タスクが実行時間，リソース占有時間，割込み禁止時間の時間制約に違反した場合の保護違反タスクはその時間制約を越えて実行したタスクとし，タスクの起動要求もしくは待ち解除要求が時間制約に違反した場合の保護違反タスクは起動要求もしくは待ち解除要求を受けたタスクとする【NOS0244】.

AUTOSAR 仕様との違い

本仕様では，AUTOSAR 仕様で規定されていたエラーコードに対して，より詳細な情報が取得できるよう細分化を行った．AUTOSAR 仕様から変更を行ったタイミング保護違反時のエラーコードを，表 2-15 に示す.

表 2-15 タスクに対するタイミング保護違反時のエラーコードの定義の違い

AUTOSAR 仕様	本仕様
E_OS_PROTECTION_TIME【OS064】	E_OS_PROTECTION_TIME_TASK≪NOS0181≫
E_OS_PROTECTION_TIME【OS210】	E_OS_PROTECTION_TIME_ISR≪NOS0186≫
E_OS_PROTECTION_ARRIVAL【OS466】	E_OS_PROTECTION_ARRIVAL_TASK≪NOS0182≫
E_OS_PROTECTION_ARRIVAL【OS467】	
E_OS_PROTECTION_ARRIVAL【OS048】	E_OS_PROTECTION_ARRIVAL_ISR≪NOS0187≫
E_OS_PROTECTION_LOCKED【OS033】	E_OS_PROTECTION_LOCKED_RESOURCE≪NOS0183≫
E_OS_PROTECTION_LOCKED【OS037】	E_OS_PROTECTION_LOCKED_OSINT≪NOS0184≫
	E_OS_PROTECTION_LOCKED_ALLINT≪NOS0185≫

2.17.4 C2ISR に対するタイミング保護機能

実行時間の監視

C2ISR の実行時間バジェットを超えて C2ISR が処理を実行し続けていないか，OS が監視する．C2ISR が終了した場合，OS は C2ISR の実行時間を 0 にリセットする【OS474】．C2ISR の実行時間が実行時間バジェットを越えた時，OS は E_OS_PROTECTION_TIME_ISR をパラメータとして保護違反時処理を呼び出す【NOS0186】.

割込み発生間隔の監視

C2ISR の処理が開始された時をタイムフレームの始点とする【OS471】．C2ISR の到着時間がタイムフレーム以上であることを，OS が監視する【OS470】．C2ISR の発生間隔がタイムフレーム未満である場合，OS は C2ISR を実行せず，E_OS_PROTECTION_ARRIVAL_ISR をパラメータとして保護違反時処理を呼び出す【NOS0187】．このとき，C2ISR は実行されないため，タイムフレームの始点は更新しない【NOS0298】.

リソース占有時間の監視

C2ISR がリソース占有時間バジェットを超えてリソースを占有し続けないか, OS が監視する. C2ISR がリソース占有時間バジェットを越えてリソースを占有した時，OS は E_OS_PROTECTION_LOCKED_RESOURCE をパラメータとして保護違反時処理を呼び出す【NOS0188】.

OS 割込み禁止時間の監視

C2ISR が OS 割込み禁止時間バジェットを超えて OS 割込みを禁止し続けないか，OS が監視する．C2ISR が OS 割込み禁止時間バジェットを越えて OS 割込みを禁止した時，OS は E_OS_PROTECTION_LOCKED_OSINT をパラメータとして保護違反時処理を呼び出す【NOS0189】.

全割込み禁止時間の監視

C2ISR が全割込み禁止時間バジェットを超えてすべての割込みを禁止し続けないか, OS が監視する．C2ISR が全割込み禁止時間バジェットを越えてすべての割込みを禁止した時，OS は E_OS_PROTECTION_LOCKED_ALLINT をパラメータとして保護違反時処理を呼び出す【NOS0190】.

保護違反時の動作

C2ISR が実行時間，割込み発生間隔，リソース占有時間，割込み禁止時間の時間制約に違反した場合に，保護違反時処理に渡すパラメータの一覧を表 2-16 に示す.

表 2-16 C2ISR の違反処理一覧

C2ISR の違反処理	保護違反時処理に渡すパラメータ
C2ISR の実行時間が実行時間バジェットを超えた.	E_OS_PROTECTION_TIME_ISR ≪NOS0186≫
C2ISR の発生間隔がタイムフレーム未満である.	E_OS_PROTECTION_ARRIVAL_ISR ≪NOS0187≫
C2ISR がリソース占有時間バジェットを越えてリソースを占有した.	E_OS_PROTECTION_LOCKED_RESOURCE ≪NOS0188≫
C2ISR が OS 割込み禁止時間バジェットを超えて OS 割込みを禁止した.	E_OS_PROTECTION_LOCKED_OSINT ≪NOS0189≫
C2ISR が全割込み禁止時間バジェットを超えてすべての割込みを禁止した.	E_OS_PROTECTION_LOCKED_ALLINT ≪NOS0190≫

保護違反 C2ISR

C2ISR が実行時間，リソース占有時間，割込み禁止時間の時間制約に違反した場合の保護違反 C2ISR は時間制約を越えて実行した C2ISR とし，C2ISR の発生間隔が時間制約に違反した場合の保護違反 C2ISR は発生した C2ISR とする【NOS0245】.

AUTOSAR 仕様との違い

本仕様では，AUTOSAR 仕様で規定されていたエラーコードに対して，より詳細な情報が取得できるよう細分化を行った．AUTOSAR 仕様から変更を行ったタイミング保護違反時のエラーコードを，表 2-17 に示す.

表 2-17 C2ISR に対するタイミング保護違反時のエラーコードの定義の違い

AUTOSAR 仕様	本仕様
E_OS_PROTECTION_TIME≪OS064≫≪OS210≫	E_OS_PROTECTION_TIME_TASK≪NOS0181≫
	E_OS_PROTECTION_TIME_ISR≪NOS0186≫
E_OS_PROTECTION_ARRIVAL≪OS466≫≪OS467≫≪OS048≫	E_OS_PROTECTION_ARRIVAL_TASK≪NOS0182≫
	E_OS_PROTECTION_ARRIVAL_ISR≪NOS0187≫
E_OS_PROTECTION_LOCKED≪OS033≫≪OS037≫	E_OS_PROTECTION_LOCKED_RESOURCE≪NOS0183≫
	E_OS_PROTECTION_LOCKED_OSINT≪NOS0184≫
	E_OS_PROTECTION_LOCKED_ALLINT≪NOS0185≫

2.18 OS アプリケーション(OSAP)

2.18.1 OSAP の概要

本 OS は，複数の OS オブジェクトの集合として OSAP を定義する．OSAP は，SC3，SC4 でサポートされる機能である．OSAP が有効な場合，リソースを除く OS オブジェクトはいずれかの OSAP に所属しなければならない．リソースは OSAP に所属しない．

OSAP は，複数定義することが可能であり，OSAPID によって識別する．

2.18.2 OSAP の種別

OSAP には，信頼 OSAP と非信頼 OSAP の 2 種類が存在する【OS446】〔OS_Conf115〕.

信頼 OSAP

信頼 OSAP とは, ハードウェアへのアクセスやシステムサービスの使用に制限がない OSAP である．ターゲットのハードウェアが特権モードをサポートする場合は，特権モード下での動作も許可される．

また，信頼 OSAP は他の OSAP に対して信頼関数を提供することができる(詳細は 2.18.8 節を参照)【OS464】.

非信頼 OSAP

非信頼 OSAP とは，ハードウェアへのアクセスやシステムサービスの使用が制限される OSAP である．ターゲットのハードウェアが特権モードをサポートする場合，特権モードでの動作は許可されない【OS058】．ただし，ターゲットのハードウェアが特権モードでもメモリ保護を実現できる場合，メモリ保護が有効かつ特権モードの状態で非信頼 OSAP を動作させることを許可する【NOS0237】.

OS の振る舞い

OS 自体は信頼 OSAP と同様の振る舞いをすることができる．ハードウェアでレジスタの保護がサポートされている場合には，OS が管理する制御レジスタに対して，非信頼 OSAP がアクセスすることを禁止する【OS096】.

また，システム定義のフックルーチンは，信頼 OSAP と同様の振る舞いをする【NOS0340】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，ハードウェアが特権モードをサポートする場合は，非信頼 OSAP は非特権モードで動作することが義務付けられている．本仕様では，特権モードでもメモリ保護が実現できるハードウェア上で動作させる場合には，非信頼 OSAP を特権モードで動作させることを許可する≪NOS0237≫.

表 2-18 信頼 OSAP と非信頼 OSAP の動作モードの違い

アプリケーションの種別	プロセッサ動作モード	メモリ保護機能	AUTOSAR仕様	本仕様
信頼 OSAP	特権モード	有効	○	○
		無効	○	○
	非特権モード	有効	×	×
		無効	×	×
非信頼 OSAP	特権モード	有効	×	○
		無効	×	×
	非特権モード	有効	○	○
		無効	×	×

○：設定可能，×：設定不可能

2.18.3 OSAP の構成

本 OS は，OSAP として，信頼関数，タスク，ISR，アラーム，スケジュールテーブル，カウンタと，OSAP 固有のスタートアップフック，エラーフック，シャットダウンフックから構成される【OS445】〔OS_Conf114〕〔OS_Conf020〕.

2.18.4 OSAP の指定

OS オブジェクトが所属する OSAP をコンフィギュレーション時に静的に指定する．また，OSAP が保護違反時処理に使用するリスタートタスクを，コンフィギュレーション時に静的に指定する(詳細は 2.22 節を参照)．リスタートタスクは，OSAP に所属するタスクの 1 つであり，OSAP 再起動時に OS によって起動されること以外は，通常のタスクと同じ振る舞いをする【NOS1139】.

SC3，SC4 で，タスク，C2ISR，カウンタ，アラーム，スケジュールテーブルがどの OSAP にも所属していない場合，ジェネレータはエラーを検出する【OS311】.

2.18.5 OSAP の状態

本 OS では，OSAP は以下の 3 つの状態を持つ．各 OSAP は以下の状態の中から，常に 1 つの状態をとる.

アプリケーション利用可能状態(APPLICATION_ACCESSIBLE)

OSAP に所属するオブジェクトを他の OSAP から利用できる状態．OS は，StartOS が呼び出された後，StartupHook を呼び出す前に全ての OSAP の状態を APPLICATION_ACCESSIBLE に初期化する【OS500】.

アプリケーション再起動状態(APPLICATION_RESTARTING)

OSAP が再起動中であり，OSAP に所属するオブジェクトを他の OSAP から利用できない状態. TerminateApplication や ProtectionHook によって OSAP が終了し，リスタート要求がある場合，OS は OSAP の状態を APPLICATION_RESTARTING とする【OS503】.

本状態は，他の OSAP から利用できないだけであり，自身の OSAP からのタスクの起動やアラーム，スケジュールテーブルの開始は可能であり，満了処理も実行される【NOS1140】.

アプリケーション終了状態(APPLICATION_TERMINATED)

OSAP が終了され，OSAP に所属するオブジェクトを他の OSAP から利用できない状態. TerminateApplication や ProtectionHook によって OSAP が終了し，リスタート要求がない場合，OS は OSAP の状態を APPLICATION_TERMINATED とする【OS502】. OS は本状態から他の状態へ遷移を行わない.

OSAP 状態ごとのアクセス権限

OSAP の状態がアプリケーション利用可能状態以外である場合，他の OSAP に所属する OS オブジェクトからのアクセスを禁止する【OS504】. アプリケーション利用可能状態ではない OSAP に対して，他の OSAP に所属する OS オブジェクトがシステムサービスの発行を行った場合，OS モジュールは E_OS_ACCESS を返す【OS509】.

OSAP の状態遷移

OSAP の状態遷移を図 2-46 に示す.

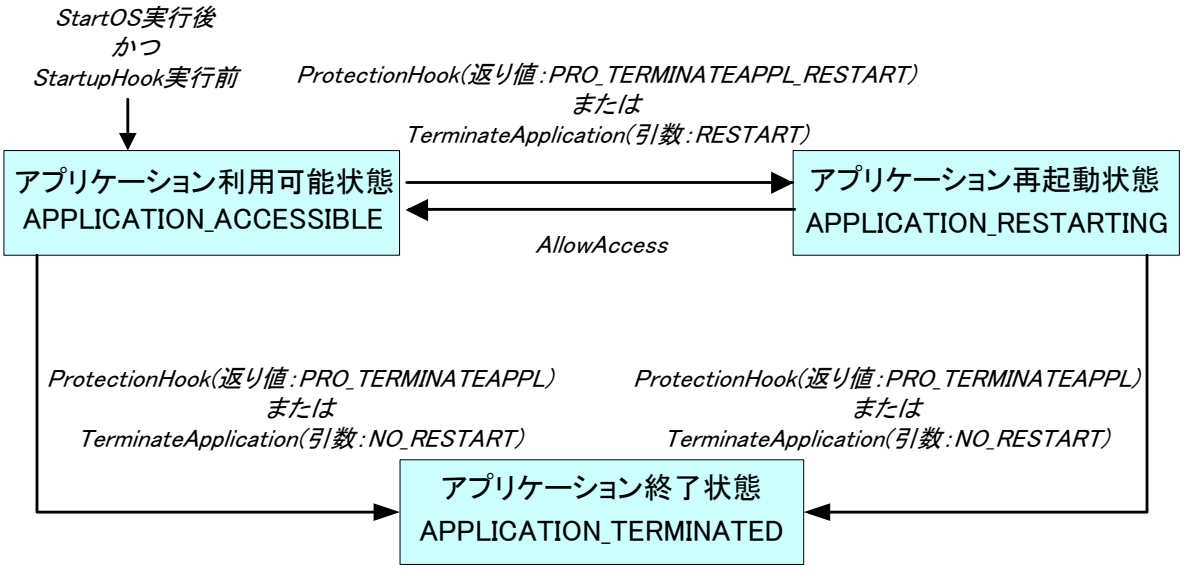


図 2-46 OSAP の状態遷移図

2.18.6 OSAP の操作

2.18.6.1 動作中の OSAPID の取得

本 OS は, 動作中の OSAP の ID を取得するシステムサービスとして GetApplicationID を提供する(詳細は 3.9.41 節を参照). なお，動作中の OSAP とは，実行中のタスク，ISR，フックルーチンが所属する OSAP を示す.

2.18.6.2 OS オブジェクトが所属する OSAPID の取得

本 OS は，タスク，ISR，カウンタ，アラーム，スケジュールテーブルが所属する OSAPID を取得するシステムサービスとして，CheckTaskOwnership，CheckISROwnership，CheckAlarmOwnership，CheckCounterOwnership，CheckScheduleTableOwnership を提供する(詳細は 3.9.53 節～3.9.57 節を参照).

2.18.6.3 OS オブジェクトに対するアクセス権の取得

本 OS は，指定した OSAP が，指定したタスク，ISR，リソース，カウンタ，アラーム，スケジュールテーブルにアクセス可能かを取得するシステムサービスとして CheckTaskAccess，CheckISRAccess，CheckAlarmAccess，CheckResourceAccess，CheckCounterAccess，CheckScheduleTableAccess を提供する(詳細は 3.9.45 節～3.9.50 節を参照).

2.18.6.4 OSAP の終了

本 OS は，OSAP の終了を要求するシステムサービスとして TerminateApplication を提供する(詳細は 3.9.59 節を参照).

OSAP の終了で行う処理を以下に示す【OS447】.

- 終了する OSAP に所属するタスクと ISR の終了
- 終了する OSAP に所属する ISR に対応する割込み要因の禁止
- 終了する OSAP に所属するアラーム，スケジュールテーブルの停止

OSAP 終了時，OSAP 固有のシャットダウンフックは呼び出されない【OSa178】.

2.18.6.5 OSAP の利用許可

本 OS は，再起動状態の OSAP を利用可能状態へ遷移させるシステムサービスとして AllowAccess を提供する(詳細は 3.9.60 節を参照).

OSAP 再起動時，OSAP 固有のスタートアップフックは呼び出されない【OSa179】.

2.18.6.6 動作中の OSAP 状態の取得

本 OS は, OSAP の状態を取得するシステムサービスとして GetApplicationState を提供する(詳細は

3.9.61 節を参照).

2.18.7 OSAP に所属する OS オブジェクトの保護

OSAP に所属する OS オブジェクトは，他の非信頼 OSAP から不正なアクセスをされないよう，OS によって保護される．なお，信頼 OSAP は他の OSAP に対して自由にアクセスすることが許される．

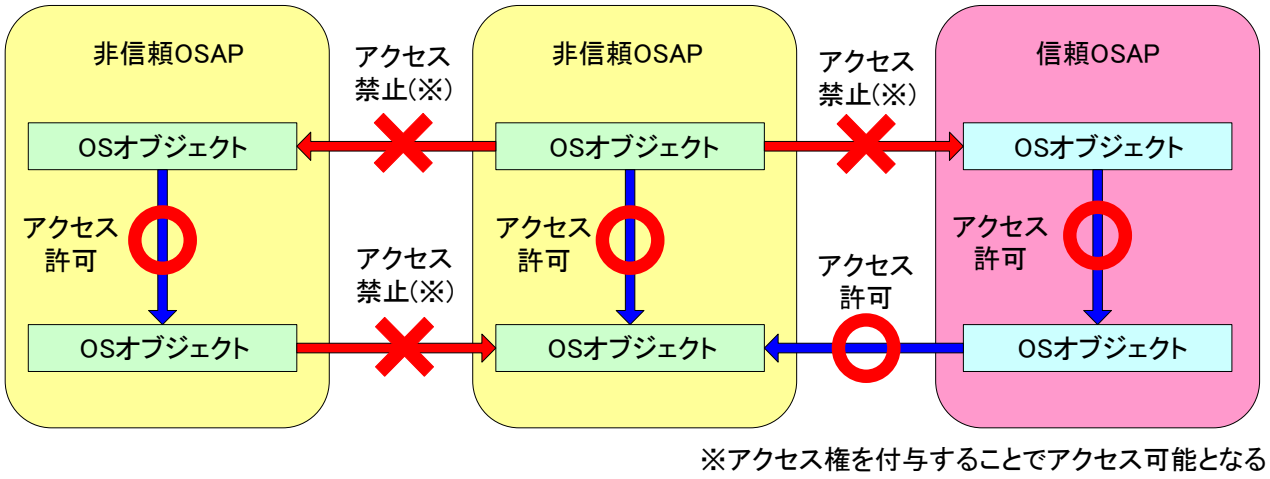


図 2-47 OSAP に所属する OS オブジェクトの保護

2.18.7.1 OS オブジェクトに対するアクセス権

OSAP に所属する処理単位は，同じ OSAP に所属するすべての OS オブジェクトに対してアクセスすることができる．しかし，非信頼 OSAP の場合，他の OSAP に所属する，アクセス権が付与されていない OS オブジェクトに対してアクセスすることは禁止されている．非信頼 OSAP から他の OSAP に所属する OS オブジェクトに対してアクセスを行うためには，コンフィギュレーション時にアクセス権を付与する必要がある【OS448】．リソースは OSAP に所属しないため，リソースへアクセス可能とする OSAP をコンフィギュレーション時に指定する必要がある【OSa013】〔OS_Conf051〕.

2.18.8 信頼関数

信頼関数は信頼 OSAP が他の非信頼 OSAP に対して提供するサービスルーチンである．OS は信頼 OSAP が信頼関数を提供するための機能を提供する【OS451】〔OS097〕〔OS_Conf021〕．また，提供された信頼関数を呼び出すためのシステムサービスとして CallTrustedFunction を提供する(詳細は 3.9.42 節を参照).

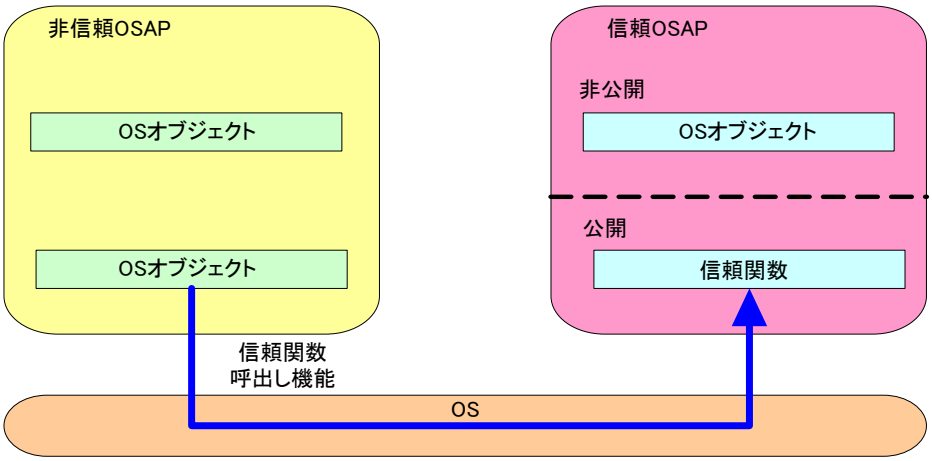


図 2-48 信頼関数を呼び出すための機能

2.18.8.1 信頼関数のパラメータのチェック

本 OS は，信頼関数のパラメータとして指定されたメモリ領域に対して，信頼関数の呼出し元のタスク，C2ISR が，読出し，書込み，実行のアクセス権を持つかをチェックするための機能を持つ．また，渡されたパラメータが呼び出し元のタスク，C2ISR のスタック領域であるかをチェックするための機能を持つ.

なお，本 OS は，タスクが，指定された領域のアクセス権を持つか，また，スタック領域であるかをチェックするシステムサービスとして CheckTaskMemoryAccess を提供する(詳細は 3.9.44 節を参照).また，C2ISR が，指定された領域のアクセス権を持つか，また，スタック領域であるかをチェックするシステムサービスとして CheckISRMemoryAccess を提供する(詳細は 3.9.43 節を参照).

2.18.8.2 信頼関数のアクセス権と優先度

信頼関数は，信頼関数が所属する信頼 OSAP と同じアクセス権で動作する≪NOS0326≫.

信頼関数がタスクから呼ばれた場合，信頼関数は呼び出されたタスクと同じタスク優先度で動作し，タスクで許可されたシステムサービスを使用することができる【OS365】.

信頼関数が C2ISR から呼ばれた場合, 信頼関数は呼び出された C2ISR と同じ割込み優先度で動作し，C2ISR で許可されたシステムサービスを使用することができる【OS364】.

2.18.8.3 信頼関数の記載方法

信頼関数の記載方法を示す.

信頼関数のプロトタイプ宣言

信頼関数のプロトタイプ宣言を以下に示す【NOS0374】.

```
StatusType TRUSTED_<信頼関数名>(TrustedFunctionIndexType FunctionIndex,
                                TrustedFunctionParameterRefType FunctionParams);
```

信頼関数の本体記述

信頼関数の本体記述を以下に示す【NOS0375】. 信頼関数はユーザ定義のエラーコードを返す【NOS0179】. したがって, 信頼関数がエラーを返した場合, OS はエラーフックを呼び出す【NOS0373】.

```
StatusType TRUSTED_<信頼関数名>(TrustedFunctionIndexType FunctionIndex,
                                TrustedFunctionParameterRefType FunctionParams)
{
}
```

AUTOSAR 仕様との違い

AUTOSAR 仕様では，信頼関数の返り値は void 型と定義されていた≪OS312≫. 本仕様では，信頼関数を StatusType 型とし，ユーザが定義した信頼関数の返り値を CallTrustedFunction の返り値とするよう規定した≪NOS0374≫≪NOS0375≫≪NOS0179≫. これにより，ユーザ定義の信頼関数がエラーを返した場合も，エラーフックを呼び出すよう規定した≪NOS0373≫.

ATK2 では，信頼関数のプロトタイプ宣言，本体記述のために，TRUSTEDFUNCTION マクロを提供する【IOS068】. TRUSTEDFUNCTION マクロを用いた信頼関数の記載方法を示す. 信頼関数の引数 FunctionIndex のデータ型は TrustedFunctionIndexType，FunctionParams のデータ型は TrustedFunctionParameterRefType である【IOS069】.

信頼関数のプロトタイプ宣言

TRUSTEDFUNCTION(TRUSTED_<信頼関数名>, FunctionIndex, FunctionParams);

信頼関数の本体記述

```
TRUSTEDFUNCTION(TRUSTED_<信頼関数名>, FunctionIndex, FunctionParams)
{
}
```

2.18.8.4 信頼関数の指定

信頼関数は OSAP のコンフィギュレーションと共に，コンフィギュレーション時に静的に指定する．

2.18.8.5 信頼関数呼び出し時のスタックチェック

信頼関数の呼出し時に，信頼関数が使用するスタックの残り量と，信頼関数の実行時に使用するスタックサイズ(OsTrustedFunctionStackSize)を比較し，オーバーフローが起きないことをチェックする【NOS0399】．非信頼タスクから呼び出された信頼関数は，非信頼タスク用システムスタックを使用する【NOS0720】．同様に，非信頼 C2ISR から呼び出された信頼関数は，非信頼 C2ISR 用システムスタックを使用する【NOS0839】．スタックの残り量が，信頼関数が使用するスタックサイズ(OsTrustedFunctionStackSize)より少ない場合，CallTrustedFunction は，信頼関数を呼び出さずに，E_OS_STACKINSUFFICIENT を返す【NOS0400】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，信頼関数呼び出し時のスタックチェックについて規定されていない．信頼関数の実行は特権モードで行われるため，メモリ保護の対象外となる．しかし，非信頼 OSAP に所属する処理単位から，スタックが残り少ない状態で信頼関数が呼び出された場合，信頼関数がスタックオーバーフローを起こし，メモリ保護違反が検知できない場合がある．そこで，信頼関数の呼び出し時には，スタックの残り量と信頼関数が使用するスタック量を比較し，オーバーフローが起きないことをチェックするよう規定した≪NOS0399≫．また，スタックが不足した状態で信頼関数が呼び出されることを避けるため, スタックの残り量が, 信頼関数が使用するスタック量より少ない場合，CallTrustedFunction は，信頼関数を呼び出さずに，E_OS_STACKINSUFFICIENT を返すよう規定した≪NOS0400≫.

2.18.8.6 信頼関数呼出しの記述例

```
/* 引数の構造体をユーザが定義することにより，信頼関数に引数を渡すことができる */
struct my_trusted_function_parameter
{
        type1 parameter1;
        type2 parameter2;
};

/* この関数をユーザが呼び出すことにより，信頼関数が使用できる */
StatusType my_trusted_function_stub(
        type1 parameter1,
        type2 parameter2)
{
        /* 引数を構造体に格納 */
        struct my_trusted_function_parameter st;
        st.parameter1 = parameter1;
        st.parameter2 = parameter2;

        /* MY_TRUSTED_FUNCTION で指定した信頼関数を呼び出す */
        return(CallTrustedFunction(MY_TRUSTED_FUNCTION, &st));
}

/* CallTrustedFunction() システムサービスの実装概念例 */
StatusType CallTrustedFunction(
        TrustedFunctionIndexType idx,
        TrustedFunctionParameterRefType ref)
{
        StatusType return_value;

        /* 信頼関数が登録されているかをチェック */
        if(idx > MAX_EX_SVC)
                return(E_OS_SERVICEID);

        /* 信頼 OSAP が実行できる状態に移行 */
        …

        /* idx で指定された信頼関数を呼び出す */
        return_value = (*(ex_svc_table[idx]))(idx, ref);

        /* 呼出し元の OSAP が実行できる状態に復帰 */
        …

        /* 返り値を返す */
        return(return_value);
}
```

```
/* CallTrustedFunction() から呼び出されるユーザ記述の信頼関数 */
TRUSTEDFUNCTION(TRUSTED_my_function, idx, param_ref)
{
	TaskType task;
	type1 parameter1;
	type2 parameter2;
	StatusType return_value = E_OK;
	struct my_trusted_function_parameter *ref = param_ref

	if (GetTaskID(&task) != E_OK)
		task = INVALID_TASK;

	if(!OSMEMORY_IS_READABLE(
		SVC_CALL(CheckTaskMemoryAccess)(task, ref, sizeof(*ref))))
	{
		/* アクセス権がない場合はエラー */
		return_value = E_OS_ACCESS;
	}
	else
	{
		parameter1 = ref->parameter1;
		parameter2 = ref->parameter2;

		/* ISR から呼び出されたか？ */
		if(GetISRID() != INVALID_ISR)
		{
			/* ISR から呼び出された場合はエラー */
			return_value = E_OS_ACCESS;
		}
		else if(OSMEMORY_IS_WRITEABLE(
			SVC_CALL(CheckTaskMemoryAccess)(task, parameter1, parameter2)))
		{
			/* 信頼関数の本体を呼び出す */
			return_value = my_trusted_function_body(parameter1, parameter2);
		}
		else
		{
			/* アクセス権がない場合はエラー */
			return_value = E_OS_ACCESS;
		}
	}
	/* 返り値を返す */
	return(return_value);
}
```

2.18.8.7 実装上の注意

信頼関数実行中は，ユーザが排他制御を行わない限り，途中で他の処理単位へ切り替わることがあるので，以下の状況が発生する可能性がある．

(1) 他の処理単位が，実行中の信頼関数を呼び出す(リエントラントに使用する).
(2) 信頼関数実行中の処理単位が所属する OSAP に所属する別の処理単位が，保護違反を発生し，その OSAP が強制終了する.
(3) 他の処理単位が，信頼関数実行中の処理単位が所属する OSAP を終了する.

したがって，信頼関数の中で，リエントラントに使用されたり，途中で終了されると，データ構造が一貫性のない状態になるなどの理由で問題が発生する処理においては，以下の処理単位との間で排他制御しなければならない.

(1) 同じデータ構造をアクセスする信頼関数(自分自身を含む)を呼び出す他の処理単位.
(2) 信頼関数を呼び出した処理単位と同一の OSAP 内の他の処理単位.
(3) 信頼関数を呼び出した処理単位が属する OASP を終了する(TerminateApplication を発行する)処理単位.

この排他制御は，SuspendOSInterrupts/ ResumeOSInterrupts を使って実現するのが最も簡単であるが，割込み禁止時間を短くしたい場合には，リソースなどのより適切な方法を用いるべきである.
また,「信頼関数を呼び出した処理単位が属する OASP を終了する処理単位」に C2ISR が含まれていると，この排他制御は，少なくとも一部の割込みを禁止することになる．そのため，排他制御区間が長い場合，割込み禁止時間が長くなってしまう.

2.18.9 マルチコア対応 OS における OSAP

本 OS ではコアを跨ぐシステムサービスを直接操作法で実装することを前提としているが，コアを跨いだ OSAP の終了処理に関しては，遠隔呼出し法によって実装することを想定する．OSAP 終了処理において，OSAP に所属する各々の OS オブジェクトの終了処理を担うのは，終了する OSAP が割付けられているコアの OS である【NOS0112】．マルチコア対応 OS における OSAP の終了は，以下の手順に従う【NOS0113】.

(1) 終了する OSAPID を引数に与えて TerminateApplication()を呼び出す.
(2) 終了する OSAP が割付けられているコアに対して，コア間割込みによって OSAP の終了処理を要求する．直ちにシステムサービスの呼出し元にリターンする.
(3) コア間割込みによる終了処理要求を受理した各コアは，シングルコア OS と同様の OSAP 終了処理を実行する.

2.18.10システムサービスの呼出し方法

本 OS の SC3，SC4 において，システムサービスは，ソフトウェア割込みによって呼出すのが基本である．システムサービス呼出しを通常の方法で記述した場合，ソフトウェア割込みによって呼出すことが生成される【NOS1153】．一般に，ソフトウェア割込みによるシステムサービス呼出しはオーバヘッドが大きい．そのため，信頼 OSAP に属する処理単位からは，関数呼出しによってシステムサービスを呼出すことで，オーバヘッドを削減することができる．そこで，信頼 OSAP に属する処理単位から関数呼出しによってシステムサービスを呼出す機能を提供する．

信頼 OSAP に属する処理単位が実行する関数のみを含んだソースファイルでは，OS ヘッダファイル(Os.h)をインクルードする前に，信頼 OSAP であることを示すマクロを定義することで，システムサービス呼出しを通常の方法で記述した場合に，関数呼出しによって呼出すコードが生成される【NOS1154】．また，信頼 OSAP に属する処理単位が実行する関数と，非信頼 OSAP に属する処理単位が実行する関数の両方を含んだソースファイルでは，関数呼出しによってシステムサービスを呼出すための名称を作るマクロ(SVC_CALL)を用いることで，関数呼出しによって呼出すコードが生成される【NOS1155】．

例えば，ActivateTask を関数呼出しによって呼出す場合には，次のように記述すればよい．

```
ercd = SVC_CALL(ActivateTask)(tskid);
```

一方，信頼 OSAP であることを示すマクロを定義したソースファイルであっても，ソフトウェア割込みによるシステムサービス呼出しを呼出すための名称マクロ(SVC_TRAP)を用いることで，ソフトウェア割込みによる呼出すコードが生成される【NOS1156】．

```
ercd = SVC_TRAP(ActivateTask)(tskid);
```

また，拡張性のために，非信頼 OSAP に属する処理単位が実行する関数のみを含んだソースファイルでは，OS ヘッダファイル(Os.h)をインクルードする前に，非信頼 OSAP であることを示すマクロを定義する【NOS1157】．それぞれのマクロ名は実装依存とする【NOS1158】．

ATK2 では，信頼 OSAP，非信頼 OSAP を示すマクロを以下とする【IOS233】．

- 信頼 OSAP：TOPPERS_TRUSTED
- 非信頼 OSAP：TOPPERS_NON_TRUSTED

信頼 OSAP，非信頼 OSAP を示すマクロを，どちらも定義して OS ヘッダファイルをインクルードした場合は，エラーとする【IOS234】．

2.19 メモリ保護

メモリ保護機能は，OSAP のメモリ領域や周辺デバイスが，不正なアクセスをされないよう，OS によって保護する機能である．

2.19.1 保護対象の OSAP

メモリ保護機能は，非信頼 OSAP がメモリ領域に対して不正なメモリアクセス(読出し，書込み，実行)を行ったときに，メモリ領域を不正なメモリアクセスから保護する機能である．

信頼 OSAP がメモリ領域に対してメモリアクセスを行うことに対しては保護を行わない【NOS0870】.

2.19.2 メモリ領域に対する保護の概要

メモリ保護機能では，アクセス対象となるメモリ領域に対して書込みアクセス保護と読出し(実行)アクセス保護を行う．

書込みアクセス保護はメモリに対する書込みアクセスの制御を行う．読出し(実行)アクセス保護はメモリからの読出しアクセスの制御を行う．実行アクセス時にはメモリに格納されたコードを読み出す必要があるため，読出し(実行)アクセス保護で保護を行う．ハードウェアでサポートされている場合は，読出しアクセスと実行アクセスに対して別々に保護を行ってもよい．

なお，メモリ保護の機能レベルが 1 の場合，読出し(実行)アクセス保護は行わない．

アクセス対象のメモリ領域に対する保護設定の一覧を表 2-19 に示す．

表 2-19 アクセス対象のメモリ領域に対する保護設定の一覧

<table>
<tr><td colspan="2" rowspan="3"></td><td colspan="4">アクセス元の処理単位</td></tr>
<tr><td colspan="2">非信頼 OSAP</td><td colspan="2">信頼 OSAP</td></tr>
<tr><td>W</td><td>R(X)</td><td>W</td><td>R(X)</td></tr>
<tr><td rowspan="5">アクセス対象のメモリ領域</td><td>同じ OSAP</td><td colspan="2" rowspan="2">表 2-20 参照</td><td>○</td><td>○</td></tr>
<tr><td>異なる OSAP</td><td>○</td><td>○</td></tr>
<tr><td>共有メモリ領域</td><td colspan="2">表 2-22 参照</td><td>○</td><td>○</td></tr>
<tr><td>OS のメモリ領域</td><td>×</td><td>×</td><td>○</td><td>○</td></tr>
<tr><td>保護領域が配置されていない領域</td><td>×</td><td>×</td><td>※1</td><td>※1</td></tr>
</table>

W：書込みアクセス　　○　：アクセス許可
R(X)：読出し(実行)アクセス　　×　：アクセス禁止
※1：実装定義

2.19.2.1 OSAP のメモリ領域

OSAP のメモリ領域に対する保護設定の一覧を表 2-20 に示す．なお，メモリ保護の機能レベルが 1 の場合，読出し(実行)アクセス保護は行われない．

表 2-20 OSAP のメモリ領域に対する保護設定の一覧

			アクセス元の処理単位	
			非信頼 OSAP	
			W	R(X)
アクセス対象のメモリ領域	同じ非信頼 OSAP 内	同じ処理単位のスタック	○	○
		異なる処理単位のスタック	×(※1)	×(※1)
		コード	×	○
		専有リードオンリーデータ	×	○
		専有リードライトデータ	○	○
		共有リード専有ライト	○	○
		専有周辺デバイス	○	○
	異なる非信頼 OSAP 間	スタック	×	×
		コード	×	×
		専有リードオンリーデータ	×	×
		専有リードライトデータ	×	×
		共有リード専有ライト	×	○
		専有周辺デバイス	×	×

W　　：書込みアクセス　　○　：アクセス許可
R(X)：読出し(実行)アクセス　　×　：アクセス禁止
※1：同じ非信頼フック同士は，同一スタックを使用するためアクセス許可となる

各メモリ領域に対する保護内容の詳細を以下に示す．

<u>タスクと C2ISR のスタック領域の保護</u>

OS は，タスク，C2ISR のスタック領域に対して，タスク，C2ISR 自身が，読出し，書込みアクセスすることを許可する【OS196】.

タスク，C2ISR のスタック領域に対して，同一の非信頼 OSAP に所属する他のタスク，C2ISR が，読出し，書込みアクセスすることを禁止する【NOS0260】.

タスク，C2ISR のスタック領域に対して，他の非信頼 OSAP に所属するタスク，C2ISR が，読出し，書込みアクセスすることを禁止する【NOS0295】.

ハードウェアでサポートされている場合には，タスク，C2ISR のスタック領域に対して，実行アクセ

スすることを禁止する【NOS0213】.

OS は，タスク，C2ISR のスタック領域に対して，すべての非信頼フックが，読出し，書込みアクセスすることを禁止する【NOS0871】.

フックルーチンのスタック領域の保護

OS は，非信頼フックのスタック領域に対して，同一の非信頼 OSAP に所属するタスク，C2ISR が，読出し，書込みアクセスすることを禁止する【NOS0872】.

非信頼フックのスタック領域は，システムに 1 つ確保されるため，すべての非信頼フックが，読出し，書込みアクセスすることを許可する【NOS0873】.

ハードウェアでサポートされている場合には，フックルーチンのスタック領域に対して，実行アクセスすることを禁止する【NOS0874】.

OSAP の専有コード領域の保護

OS は，非信頼 OSAP の専有コード領域に対して，非信頼 OSAP 自身が実行アクセスすることを許可する【NOS0148】. 非信頼 OSAP の専有コード領域に対して，非信頼 OSAP 自身が書込みアクセスをすることは禁止とする【NOS0251】.

非信頼 OSAP の専有コード領域に対して，非信頼 OSAP 自身が読出しアクセスすることを禁止するかは実装定義とする【NOS0265】. ATK2 では，非信頼 OSAP の専有コード領域に対して，非信頼 OSAP 自身が読出しアクセスすることを禁止するかを，ターゲット定義とする【IOS099】.

非信頼 OSAP の専有コード領域に対して，他の非信頼 OSAP が読出し，書込み，実行アクセスすることを禁止する【NOS0252】.

OSAP の専有リードオンリーデータ領域の保護

OS は，非信頼 OSAP の専有リードオンリーデータ領域に対して，非信頼 OSAP 自身が読出しアクセスすることを許可し，書込みアクセスすることは禁止する【NOS0305】.

非信頼 OSAP の専有リードオンリーデータ領域に対して，非信頼 OSAP 自身が実行アクセスすることを禁止するかは実装定義とする【NOS0307】. ATK2 では，非信頼 OSAP の専有リードオンリーデータ領域に対して，非信頼 OSAP 自身が実行アクセスすることを禁止するかを，ターゲット定義とする【IOS199】.

非信頼 OSAP の専有リードオンリーデータ領域に対して，他の非信頼 OSAP が，読出し，書込みアクセスすることを禁止する【NOS0306】.

OSAP の専有リードライトデータ領域の保護

OS は，非信頼 OSAP の専有リードライトデータ領域に対して，非信頼 OSAP 自身が読出し，書込みアクセスすることを許可する【OS086】.

非信頼 OSAP の専有リードライトデータ領域に対して，他の非信頼 OSAP が，読出し，書込みアクセスすることを禁止する【NOS0294】.

ハードウェアでサポートされている場合には，非信頼 OSAP の専有リードライトデータ領域に対して，実行アクセスすることを禁止する【NOS0214】.

OSAP の共有リード専有ライトデータ領域の保護

OS は，非信頼 OSAP の共有リード専有ライトデータ領域に対して，すべての非信頼 OSAP が，読出しアクセスすることを許可する【NOS0250】.

非信頼 OSAP の共有リード専有ライトデータ領域に対して，非信頼 OSAP 自身が書込みアクセスすることを許可するが，他の非信頼 OSAP が書込みアクセスすることを禁止する【NOS0296】

ハードウェアでサポートされている場合には，非信頼 OSAP の共有リード専有ライトデータ領域に対して，実行アクセスすることを禁止する【NOS0216】.

OSAP の専有周辺デバイスの保護

OS は，信頼 OSAP が周辺デバイスに対して，読出し，書込みアクセスすることを許可する【OS209】.

非信頼 OSAP による，OSAP の専有周辺デバイスに対するアクセスを制御する機能は，機能レベル 3 では必須とする．本機能を持つ場合，OS は専有周辺デバイスに対して OSAP のメモリ領域と同様の保護を行う【NOS0266】．本機能を持たない場合，OS は専有周辺デバイスに対して非信頼 OSAP が読出し，書込み，実行アクセスすることを一律禁止する【NOS0255】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，メモリ領域に対する書込みアクセスの制御は必須であるのに対して，読出しアクセスの制御はオプションまたは未規定となっている．本仕様では，機能レベルという概念を導入し，機能レベル 1 では，読出しアクセスの制御は行わず，機能レベル 2 および機能レベル 3 では，読出しアクセスの制御を必須と規定した．これにより，AUTOSAR 仕様との互換を維持している．

AUTOSAR 仕様から変更を行った OSAP のメモリ領域に対する保護設定の違いを，表 2-21 に示す.

表 2-21 OSAP のメモリ領域に対する保護設定の違い

AUTOSAR 仕様	本仕様	変更内容
【OS208】	≪NOS0260≫	タスク，C2ISR のスタック領域に対して，同一の非信頼 OSAP に所属する他のタスク，C2ISR がアクセスする場合の規定が記述されている． AUTOSAR 仕様では，書込みアクセスのみを禁止している≪OS208≫. これに対して，書込み，読出しアクセスを禁止するよう規定した≪NOS0260≫.

AUTOSAR 仕様	本仕様	変更内容
【OS355】	≪NOS0295≫	タスク，C2ISR のスタック領域に対して，他の非信頼 OSAP に所属するタスク，C2ISR がアクセスする場合の規定が記述されている． AUTOSAR 仕様では，書込みアクセスのみを禁止している≪OS355≫． これに対して，書込み，読出しアクセスを禁止するよう規定した≪NOS0295≫．
【OS027】	≪NOS0252≫	OSAP の専有コード領域に対して，他の非信頼 OSAP がアクセスする場合の規定が記述されている． AUTOSAR 仕様では，実行アクセスを禁止してもよいと規定している≪OS027≫． これに対して，書込み，読出し，実行アクセスを禁止と明記した≪NOS0252≫．
【OS087】 【OS195】 【OS356】	－	AUTOSAR 仕様では，タスク，C2ISR 毎の専有データ領域をサポートしているのに対して，本仕様ではサポートしない．
【OS026】 【OS207】	≪NOS0294≫	OSAP の専有リードライトデータ領域に対して，他の非信頼 OSAP がアクセスする場合の規定が記述されている． AUTOSAR 仕様では，読出しアクセスを禁止してもよいと規定している≪OS026≫． また，書込みアクセスを禁止すると規定としている≪OS207≫． これに対して，書込み，読出しアクセスの禁止を明記した≪NOS0294≫．
【OS083】	≪NOS0266≫	専有周辺デバイス領域に対して，非信頼 OSAP がアクセスする場合の規定が記述されている． AUTOSAR 仕様では，書込み許可に関してのみ規定されている≪OS083≫． しかし，本仕様では，より一般化して，OSAP のメモリ領域と同様の保護を行うと規定した≪NOS0266≫．

2.19.2.2 共有メモリ領域

共有メモリ領域に対する保護設定の一覧を表 2-22 に示す.

表 2-22 共有メモリ領域に対する保護設定の一覧

			アクセス元の処理単位	
			非信頼 OSAP	
			W	R(X)
アクセス対象のメモリ領域	共有メモリ領域	コード	×	○
		共有リードオンリーデータ	×	○
		共有リードライトデータ	○	○
		共有周辺デバイス	○	○

W　　：書込みアクセス　　○　：アクセス許可
R(X)：読出し(実行)アクセス　　×　：アクセス禁止

各メモリ領域に対する保護内容の詳細を以下に示す.

共有コード領域の保護

OS は共有コード(ライブラリ)領域に対して，すべての非信頼 OSAP が実行アクセスすることを許可する【OS081】. すべての非信頼 OSAP が書込みアクセスすることは禁止とする【NOS0217】.

共有コード領域に対して，すべての非信頼 OSAP が，読出しアクセスすることを禁止するかは，実装定義とする【NOS0267】. ATK2 では，共有コード領域に対して，すべての非信頼 OSAP が，読出しアクセスすることを禁止するかを，ターゲット定義とする【IOS100】.

共有リードオンリーデータ領域の保護

OS は共有リードオンリーデータ領域に対して，すべての非信頼 OSAP が読出しアクセスすることを許可し，書込みアクセスすることを禁止する【NOS0308】.

共有リードオンリーデータ領域に対して，実行アクセスすることを禁止するかは，実装定義とする【NOS0309】. ATK2 では，共有リードオンリーデータ領域に対して，実行アクセスすることを禁止するかを，ターゲット定義とする【IOS200】.

共有リードライトデータ領域の保護

OS は共有リードライトデータ領域に対して，すべての非信頼 OSAP が読出し，書込みアクセスすることを許可する【NOS0141】.

ハードウェアでサポートされている場合には，共有リードライトデータ領域に対して，実行アクセスすることを禁止する【NOS0215】.

共有周辺デバイスの保護

OS は，信頼 OSAP が周辺デバイスに対して，読出し，書込みアクセスすることを許可する≪OS209≫.

非信頼 OSAP による，共有周辺デバイスに対するアクセスを制御する機能は，機能レベル 3 では必須とする．本機能を持つ場合，OS は共有周辺デバイスに対して共有メモリ領域と同様の保護を行う【NOS0256】．本機能を持たない場合，OS は共有周辺デバイスに対して非信頼 OSAP が読出し，書込み，実行アクセスすることを一律禁止する【NOS0153】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，OSAP のメモリ領域と共有コード領域に関してのみ規定しているが，本仕様では，その他の共有メモリ領域に関する規定を追加した.

2.19.2.3 OS のメモリ領域

OS のメモリ領域の保護

OS は，OS のメモリ領域に対して非信頼 OSAP が読出し，書込み，実行アクセスすることを禁止する【NOS0253】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，OS のデータとスタック領域に対するアクセス保護の規定がされていた【OS198】．本仕様では，非信頼 OSAP が OS の全てのメモリ領域に対してアクセス禁止であることを明確に規定した≪NOS0253≫.

2.19.2.4 保護領域が配置されていない領域

保護領域が配置されていない領域の保護

保護領域が配置されていない領域に対して信頼 OSAP が読出し，書込み，実行アクセスできるかどうかは，実装定義とする【NOS0911】．ATK2 では，保護領域が配置されていない領域に対して信頼 OSAP が読出し，書込み，実行アクセスできるかどうかを，ターゲット定義とする【IOS159】

OS は，保護領域が配置されていない領域に対して非信頼 OSAP が読出し，書込み，実行アクセスすることを禁止する【NOS0281】.

AUTOSAR 仕様との違い

本仕様では，非信頼 OSAP が保護領域として設定されていない領域に対してアクセス禁止であることを明確に規定した≪NOS0281≫.

2.19.3 保護違反時の動作

OS は，メモリ保護違反を検出した場合，E_OS_PROTECTION_MEMORY をパラメータとして保護違反時処理を呼び出す【OS044】.

2.19.4 メモリ保護の設定における前提

メモリ領域の保護に関する設定は，コンフィギュレーション時に静的に指定する【NOS0220】．システム動作中にメモリ保護に関する設定を変更する機能は提供しない．

オブジェクトモジュール

プログラムのオブジェクトコードとデータを含むファイルをオブジェクトモジュールと呼ぶ．オブジェクトファイルとライブラリファイルは，オブジェクトモジュールである．

標準のセクション

コンパイラに特別な指定をしないで出力されるセクションを，標準のセクションと呼ぶ．
標準のセクションには，以下のセクションが含まれるが，詳細はターゲット毎に異なる．

- コードセクション
- 定数データセクション
- データセクション
  - 初期値を持つデータセクション
  - 初期値を持たないデータセクション

OS が提供する標準のセクション

ATK2 では，OS は以下のセクションを提供する【IOS143】．これらのセクションは，コンフィギュレーション時に定義できない【IOS144】．OS が提供するセクション名は，ターゲット定義とする【IOS145】．

- 非信頼 OSAP 毎の標準のセクションを配置するためのセクション
- OS および信頼 OSAP のメモリ領域の標準のセクションを配置するためのセクション
- 共有領域の標準のセクションを配置するためのセクション

メモリリージョン

オブジェクトモジュールの配置対象となる同じ性質を持った一連のメモリ領域をメモリリージョンと呼ぶ．使用できるメモリリージョンは，実装毎に定める(詳細は 2.2.3 節を参照)．

メモリリージョンとメモリセクションの関係

図 2-49 にメモリリージョンとメモリセクションの関係を示す．ジェネレータは，コンフィギュレーションファイルに記述されたメモリセクションを，指定されたパラメータに従ってメモリリージョンに配置する．配置結果はリンカの入力として使われる情報ファイル(リンカスクリプト等)として出力される．

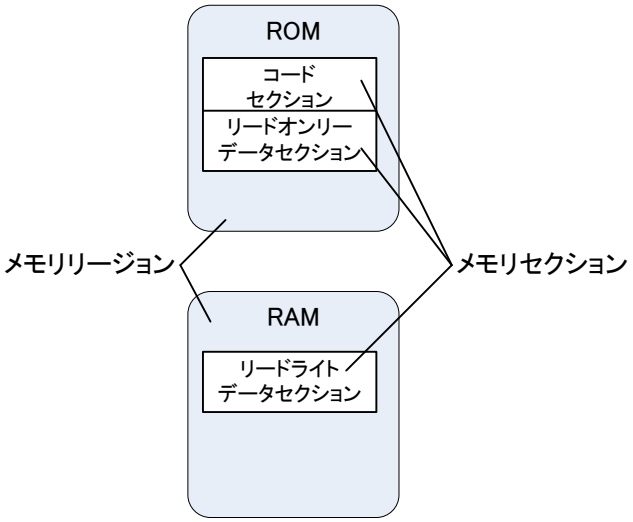


図 2-49 メモリリージョンとセクションの関係

開発環境に関する前提

メモリ保護の設定を行うために，開発環境(コンパイラやリンカ)が，以下に挙げる機能を持つことが前提となる．

- オブジェクトコードおよびデータが出力されるセクションを指定された名称のセクションに変更できること．

- 各セクションを配置するメモリリージョンと各セクションの配置順序を指定できること．

- 各セクションを配置するアドレスをハードウェアが持つメモリ保護機構の制約に合致するようにアラインできること．

また，次の機能を持つことが望ましいが，必須とはしない．

- 標準のセクションを配置するメモリリージョンと，そのメモリリージョン内での標準のセクションの配置順序を，オブジェクトモジュールごとに指定できること

AUTOSAR 仕様との違い

AUTOSAR 仕様では，メモリ保護のコンフィギュレーション方法が規定されていないため，本仕様で新たに規定した．

2.19.5 メモリ保護の設定方法

セクション名指定によるメモリ保護設定

ユーザは OsMemorySection コンテナを用いることで，指定したセクションを指定したメモリリージョンに配置し，メモリ保護の設定として，書込み，読出し，実行アクセスの許可，禁止の指定ができる(詳細は 3.8.40 節を参照)【NOS0224】.

後述のオブジェクトモジュールごとのメモリ保護設定(OsMemoryModule コンテナ)がサポートされている場合には，ユーザは，標準のセクションのメモリ保護設定を，OsMemorySection コンテナで行うことはできない【NOS0908】.

オブジェクトモジュール指定によるメモリ保護設定

ユーザは OsMemoryModule コンテナを用いることで，指定したオブジェクトモジュールに含まれる標準のセクションを，セクションの種類別に定められたデフォルトのメモリリージョンに配置することができる【NOS0257】．なお，デフォルトのメモリリージョンは実装定義である(例：データセクションは RAM でなければならない)【NOS0299】．ATK2 では，デフォルトのメモリリージョンを，ターゲット定義とする【IOS101】．また，指定したオブジェクトモジュールに含まれる標準のセクションのメモリ保護属性は，実装定義とする(詳細は 3.8.41 節を参照)【NOS0300】．ATK2 では，標準のセクションのメモリ保護属性をターゲット定義とする【IOS178】.

この機能を実現するためには，開発環境(コンパイラやリンカ)が，標準のセクションを配置するメモリリージョンとセクションの配置順序をオブジェクトモジュールごとに指定する機能を持っていることが必要である．

アドレス指定によるメモリ保護設定

ユーザは OsMemoryArea コンテナを用いることで，先頭アドレスとサイズで指定したメモリ領域に対して，メモリ保護属性の設定を指定できる(詳細は 3.8.42 節を参照)【NOS0226】.

この機能は，周辺デバイスが配置されたメモリ領域に対してメモリ保護設定を行うためのものであるが，コードやデータを配置する通常のメモリに対しても用いることができる．

スタック領域のメモリ保護設定

OS は，コンフィギュレーション時にタスク，C2ISR が使用するスタック領域を自動的に確保するため，ユーザがスタック領域のメモリ保護設定を行う必要はない【NOS0259】．ATK2 において，ユーザがスタック領域を指定する場合のメモリ保護設定方法はターゲット定義とする【IOS201】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，メモリ領域のコンフィギュレーションの仕組みが定義されていないため，本仕様で新たに規定した．

2.20 サービス保護

サービス保護機能は，不正な動作から OS を保護する機能である．システムを危険な状態にする振る舞いや，OSAP が起こすエラーに対する保護機能を提供する．

各 SC におけるサービス保護機能の適用一覧を表 2-23 に示す【NOS0002】.

表 2-23 サービス保護機能の適用一覧

サービス保護機能の内容	SC1	SC2	SC3	SC4
処理単位の不正な処理に対する保護	※以下の分類を参照			
タスクが不正終了した場合の保護	○	○	○	○
C2ISR が不正終了した場合の保護	○	○	○	○
フックルーチンが不正終了した場合の保護	○	○	○	○
非信頼 OSAP のシャットダウン要求からの保護			○	○
システムサービスでの不正な処理への保護	※以下の分類を参照			
不正なアドレスパラメータに対する書込みの検知			○	○
不正な処理単位からのシステムサービス呼出しの検知	○	○	○	○
割込み許可状態での割込み許可システムサービス呼出しに対する保護	○	○	○	○
割込み禁止状態でのシステムサービス呼出し	○	○	○	○
システムサービスの引数への NULL ポインタ指定に対する保護	○	○	○	○
不正な ID を指定したシステムサービス呼出しの検知	○	○	○	○
C1ISR を指定したシステムサービス呼出しの検知	○	○	○	○
内部リソースを指定したシステムサービス呼出しの検知	○	○	○	○
異なる OSAP からのアクセス保護	※以下の分類を参照			
アクセス権を持たない処理単位からの OS オブジェクトアクセス保護			○	○
メモリ領域に対するアクセス権チェックサービスの提供			○	○
OS オブジェクトに対するアクセス権チェックサービスの提供			○	○
OS シャットダウン時のポストタスクフックの呼出し	○	○	○	○

AUTOSAR 仕様との違い

AUTOSAR 仕様におけるサービス保護機能には，OSEK 仕様に対する補完の仕様とメモリ保護に関する仕様が含まれており SC3，SC4 でのみサポートする仕様である．本仕様では OSEK 仕様に対する補完の仕様を SC1，SC2 でもサポートするよう拡張した≪NOS0002≫.

2.20.1 処理単位の不正な処理に対する保護

2.20.1.1 タスクが不正終了した場合の保護

タスクがシステムサービスによるタスクの終了を行わずにタスク処理を終了することを，タスクの不正終了と呼ぶ．タスクの不正終了時に，OS は以下の処理を順に行う【NOS0443】.

(1) マルチコア対応 OS において，不正終了したタスクがスピンロックを占有していた場合，OS は占有していたスピンロックを解放する【NOS1115】.

(2) 不正終了したタスクが，SuspendAllInterrupts，SuspendOSInterrupts による割込み禁止状態だった場合，OS は割込み許可状態に戻す【OS239】．DisableAllInterrupts による割込み禁止状態は，スケジューラ起動時に許可状態に戻される≪NOS0648≫.

(3) 不正終了したタスクがリソースを占有していた場合，OS は占有していたリソースを解放する【OS070】.

(4) システム定義のエラーフックが有効の場合，不正終了したタスクが実行状態から休止状態に遷移する前に E_OS_MISSINGEND をパラメータとして，システム定義のエラーフックを呼び出す【OS069】．OSAP 固有のエラーフックが有効の場合，システム定義のエラーフックの後に，E_OS_MISSINGEND をパラメータとして，不正終了したタスクが所属する OSAP の，OSAP 固有のエラーフックを呼び出す≪OS246≫.

(5) スタックモニタリングが有効の場合，不正終了したタスクのスタックモニタリングを行う【IOS166】．プロテクションフックが有効の場合で，スタックオーバーフローを検出した場合，ポストタスクフックは呼び出されずに，プロテクションフックが呼び出される≪IOS164≫．このプロテクションフックで，タスクを強制終了した場合，ポストタスクフックは呼び出されない≪IOS165≫.

(6) OS はタスクの終了を行い，ポストタスクフックが有効な場合はポストタスクフックを呼び出す【OS052】.

2.20.1.2 C2ISR が不正終了した場合の保護

C2ISR が，割込み禁止状態もしくはリソースを占有した状態で C2ISR を終了した場合を，C2ISR の不正終了と呼ぶ．C2ISR の不正終了時に，OS は以下の処理を順に行う【NOS0445】.

(1) スタックモニタリングが有効の場合，不正終了した C2ISR のスタックモニタリングを行う【IOS167】.

(2) マルチコア対応 OS において，不正終了した C2ISR がスピンロックを占有していた場合，OS は占有していたスピンロックを解放した後で，システム定義のエラーフックが有効であれば E_OS_SPINLOCK をパラメータとしてシステム定義のエラーフックを呼び出す【NOS1116】. OSAP 固有のエラーフックが有効の場合，システム定義のエラーフックの後に，E_OS_SPINLOCK をパラメータとして，不正終了したタスクが所属する OSAP の，OSAP 固有のエラーフックを呼び出す≪OS246≫.

(3) C2ISR が割込み禁止状態で処理を終了した場合，OS は割込み許可状態に戻した後で，システム定義のエラーフックが有効であれば E_OS_DISABLEDINT をパラメータとしてシステム定義のエラーフックを呼び出す【OS368】. OSAP 固有のエラーフックが有効の場合，システム定義のエラーフックの後に，E_OS_DISABLEDINT をパラメータとして，不正終了したタスクが所属する OSAP の，OSAP 固有のエラーフックを呼び出す≪OS246≫.

(4) C2ISR が獲得したリソースを解放せずに処理を終了した場合，OS は解放されていないリソースを解放し，システム定義のエラーフックが有効であれば E_OS_RESOURCE をパラメータとしてシステム定義のエラーフックを呼び出す【OS369】. OSAP 固有のエラーフックが有効の場合，システム定義のエラーフックの後に，E_OS_RESOURCE をパラメータとして，不正終了したタスクが所属する OSAP の，OSAP 固有のエラーフックを呼び出す≪OS246≫.

2.20.1.3 フックルーチンが不正終了した場合の保護

フックルーチンが，SuspendAllInterrupts，SuspendOSInterrupts による割込み禁止状態もしくはスピンロックを占有した状態(プロテクションフックのみ)で処理を終了した場合を，フックルーチンの不正終了と呼ぶ．フックルーチンの不正終了時に，OS は以下の処理を順に行う【NOS1125】.

(1) マルチコア対応 OS において，不正終了したプロテクションフックがスピンロックを占有していた場合，OS は占有していたスピンロックを解放した後で，システム定義のエラーフックが有効であれば E_OS_SPINLOCK をパラメータとしてシステム定義のエラーフックを呼び出す【NOS1124】.

(2) フックルーチンが，SuspendAllInterrupts，SuspendOSInterrupts による割込み禁止状態で処理を終了した場合，OS は割込み許可状態に戻した後で，エラーフックが有効であれば E_OS_DISABLEDINT をパラメータとしてエラーフックを呼び出す【NOS0645】. DisableAllInterrupts による割込み禁止状態が，フックルーチン終了後に維持されるかどうかは，実装定義とする【NOS0646】. ATK2 では，DisableAllInterrupts による割込み禁止状態が，フックルーチン終了後に維持されるかどうかは，ターゲット依存とする【IOS232】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，タスクの不正終了時に OS が行う処理の順序が規定されていなかったため，本仕様では，まず不正終了を検知したことによるエラー処理であるエラーフックを行い，エラーの原因である割込み禁止の解除，リソースの解放を行った後，正常な状態でポストタスクフックを呼び出すよう規定した≪NOS0443≫．C2ISR も同様に順序を明確に規定した．≪NOS0445≫

また，フックルーチンが不正終了した場合の保護が規定されていなかったため，本仕様で規定した≪NOS0645≫≪NOS0646≫．

2.20.1.4 非信頼 OSAP のシャットダウン要求からの保護

使用できるシステムサービスは呼出し元の処理単位によって制限されているが，非信頼 OSAP では，加えてシステム全体に影響を与えるシステムサービスの使用を制限する．

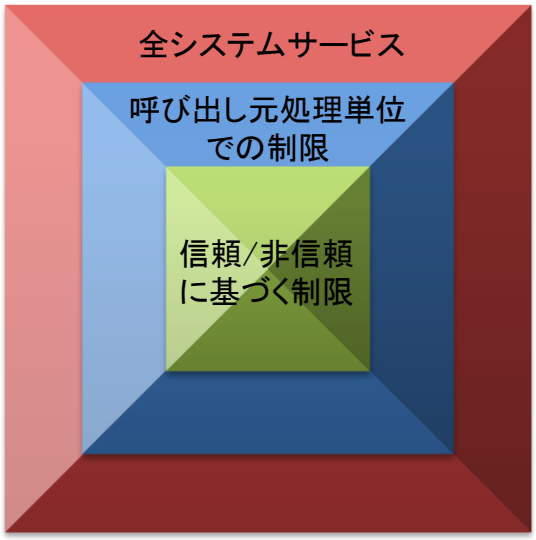


図 2-50 システムサービス使用の制限

OS は非信頼 OSAP に所属する処理単位からの OS シャットダウン要求を無視する【OS054】．ただし，非信頼 OSAP のタスク，C2ISR から呼び出された信頼関数からの OS シャットダウンは可能である≪NOS0326≫．

2.20.2 システムサービスでの不正な処理への保護

2.20.2.1 不正なアドレスパラメータに対する書込み

OSAP が有効な場合, システムサービスに渡されたアドレスが呼出し元の OSAP から書込みできないメモリ領域を指している場合，OS は E_OS_ILLEGAL_ADDRESS を返す【OS051】. CheckTaskMemoryAccess，CheckISRMemoryAccess を除いて，E_OS_ILLEGAL_ADDRESS は，非信頼 OSAP からの呼び出しでのみ返す【NOS0869】.

2.20.2.2 不正な処理単位からのシステムサービス呼出し

実行中の処理単位から呼び出せないシステムサービスを呼び出した場合，返り値の型が StatusType のシステムサービスは E_OS_CALLEVEL を返し，返り値の型が StatusType でないシステムサービスは無効値を返す【OS088】. 返り値の型が StatusType でないシステムサービスの場合，エラーコードを E_OS_CALLEVEL としてエラーフックを呼び出すことでエラーを検知する【NOS0411】〔NOS0409〕.

返り値の型が，AccessType であるシステムサービスにおける無効値は，アクセス不可能を指す【NOS0520】. 返り値の型が，ObjectAccessType であるシステムサービスにおける無効値は，NO_ACCESS を指す【NOS0521】.

なお，C1ISR から呼び出せないシステムサービスを呼び出した場合の動作は保証しない【NOS0320】〔COS0602〕.

2.20.2.3 割込み禁止状態でのシステムサービス呼出し

タスク，C2ISR，フックルーチンが割込み禁止の状態で，返り値の型が StatusType のシステムサービスを呼び出した場合，別に規定がない限りは，OS はシステムサービスの処理を行わず, E_OS_DISABLEDINT を返す【OS093】. 返り値の型が StatusType でないシステムサービスの場合，別に規定がない限りは，OS はシステムサービスの処理を行わず，エラーコードを E_OS_DISABLEDINT としてエラーフックを呼び出すことでエラーを検知する【NOS0412】〔NOS0409〕.

DisableAllInterrupts と EnableAllInterrupts によるクリティカルセクション中に，SuspendAllInterrupts，ResumeAllInterrupts，SuspendOSInterrupts，ResumeOSInterrupts を呼び出した場合，エラーコードを E_OS_DISABLEDINT として，エラーフックを呼び出す【NOS0712】〔NOS0409〕.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，C1ISR からの不正なシステムサービス呼出しの規定が記述されていなかったため，本仕様で明確に規定した≪NOS0320≫.

本仕様では，返り値の型が StatusType でないシステムサービスの場合でも，エラーフックが呼ばれると規定したことから，該当するシステムサービスにおいて，エラーフックが呼ばれた場合に渡されるエラーコードを規定した≪NOS0411≫≪NOS0412≫.

AUTOSAR 仕様では，割込み許可状態で割込み許可システムサービスを呼び出した場合，OS は処理

を行わないと規定されていた【OS092】．しかし，本仕様では，割込み許可システムサービスもエラーフックを呼び出すよう規定したため，この仕様を削除した．

2.20.2.4 システムサービスの引数への NULL ポインタ指定に対する保護

引数にポインタを指定するシステムサービスにおいて，値が NULL であるポインタを指定した場合，OS は E_OS_PARAM_POINTER を返す【OS566】．

AUTOSAR 仕様との違い

AUTOSAR 仕様 V5.0.0 (R4.0 Rev 3)では，NULL ポインタ指定に対するエラーコードが OS_E_PARAM_POINTER として規定されている≪OS566≫．しかし，他のエラーコードとプレフィックスが逆であり，誤記と思われるため，本仕様では，E_OS_PARAM_POINTER とした．なお，AUTOSAR 仕様 V5.1.0 (R4.1 Rev 1)では，E_OS_PARAM_POINTER へ修正されている．

ATK2 では，互換性のために，E_OS_PARAM_POINTER と同じ値で，OS_E_PARAM_POINTER を定義する【IOS228】．

2.20.2.5 返り値が StatusType でないシステムサービスに対するエラー

返り値の型が StatusType でなく，オブジェクト ID を指定するシステムサービスに対して，不正な ID を指定して呼び出した場合，エラーコードを E_OS_ID としてエラーフックを呼び出す【NOS0519】〔NOS0409〕．

返り値の型が StatusType でなく，引数に ISRID を指定するシステムサービスに対して，C1ISR を指定して呼び出した場合，エラーコードを E_OS_ID としてエラーフックを呼び出す【NOS0699】〔NOS0409〕．

返り値の型が StatusType でなく，引数にリソース ID を指定するシステムサービスに対して，内部リソースを指定して呼び出した場合，エラーコードを E_OS_ID としてエラーフックを呼び出す【NOS0700】〔NOS0409〕．

2.20.2.6 OS 起動前と OS 終了後のシステムサービス呼出し

通常のシステムサービスは OS 起動前と OS シャットダウン後に使用してはならないが，DisableAllInterrupts/EnableAllInterrupts と SuspendAllInterrupts/ResumeAllInterrupts の組は，例外として使用することができる(ただし，スタティック変数が初期化されていることを前提とする)【OS299】．また，マルチコア対応 OS においては，OS 起動前であっても，StartCore，StartNonAutosarCore，GetCoreID は，使用することができる【NOS1126】．

OS 起動前に，DisableAllInterrupts，EnableAllInterrupts，SuspendAllInterrupts，ResumeAllInterrupts，StartCore，StartNonAutosarCore，GetCoreID，StartOS 以外のシステムサービスを呼び出した場合の動作は保証されない【NOS0850】．OS 起動前に呼び出すことができるシステムサービスにおいて，エラーが発生した場合は，エラーフックは呼び出されない【NOS1127】．

2.20.2.7 起動していないコアに対する呼出し

マルチコア対応 OS において，本 OS が起動していないコアに対して，返り値の型が StatusType のシステムサービスを呼び出した場合，OS はシステムサービスの処理を行わず，E_OS_CORE を返す【OSa120】．返り値の型が StatusType でないシステムサービスの場合，別に規定がない限りは，OS はシステムサービスの処理を行わず，エラーコードを E_OS_CORE としてエラーフックを呼び出すことでエラーを検知する【NOS0926】〔NOS0409〕．

2.20.3 異なる OSAP からのアクセス保護

OSAP が有効な場合，アクセス権のない OS オブジェクトやメモリへのアクセスを不正なアクセスとして検出する．

2.20.3.1 アクセス権を持たない処理単位からの OS オブジェクトアクセス保護

システムサービスのパラメータにアクセス権を持たない OS オブジェクトの ID を指定された場合，OS は E_OS_ACCESS を返す【OS056】．

2.20.3.2 メモリ領域に対するアクセス権チェックサービスの提供

すべての非信頼 OSAP に対して，OS は特定のタスク，C2ISR のメモリ領域へのアクセス権があるかをチェックするシステムサービスとして CheckTaskMemoryAccess(詳細は 3.9.44 節を参照)と CheckISRMemoryAccess(詳細は 3.9.43 節を参照)を提供する【OS449】〔OS512〕〔OS513〕．

2.20.3.3 OS オブジェクトに対するアクセス権チェックサービスの提供

すべての非信頼 OSAP に対して，OS は特定の OS オブジェクトへのアクセス権があるかをチェックするシステムサービスとして CheckTaskAccess，CheckISRAccess，CheckAlarmAccess，CheckResourceAccess，CheckCounterAccess，CheckScheduleTableAccess を提供する(詳細は 3.9.45 節～3.9.50 節を参照)．

2.21 CPU 例外発生に対する処理

0 による除算，不正命令実行などの CPU 例外が発生した場合，OS は E_OS_PROTECTION_EXCEPTION をパラメータとしてプロテクションフックを呼び出す【OS245】. CPU 例外発生時に，プロテクションフックから CPU 例外要因を取得できる方法を，実装定義で用意することを推奨する【NOS0838】. ATK2 では，CPU 例外要因を取得できる方法を，ターゲット定義とする【IOS131】.

2.22 保護違反時処理

保護違反時処理は，保護違反が発生した際に保護違反に対するユーザ定義のエラー処理を実行する機能である．保護違反時処理を発生する要因はタイミング保護機能，メモリ保護機能，スタックモニタリング，CPU 例外の 4 つがある．

2.22.1 保護違反時処理が動作する処理レベル

保護違反を発生する要因ごとの保護違反時処理が動作する処理レベルを以下に示す．

- タイミング保護

  OS 処理レベルで動作する【NOS0635】．ただし，DisableAllInterrupts を発行した場合，システムは C1ISR 処理レベルがマスクされている状態となるので，この割込み禁止期間を監視するためには，C1ISR 処理レベルより高い処理レベル(全割込み禁止時間監視保護違反時処理レベル)を OS が管理する必要がある【NOS0640】．

- メモリ保護

  OS 処理レベルで動作する【NOS0637】．

- スタックモニタリング

  OS 処理レベルで動作する【NOS0906】．

- CPU 例外
  - C1ISR 以外の処理単位における CPU 例外

    OS 処理レベルで動作する【NOS0638】．

  - C1ISR における CPU 例外

    C1ISR 処理レベルで動作する【NOS0639】．

2.22.2 プロテクションフック

保護違反が発生した時に，プロテクションフックが有効であれば OS はプロテクションフックを呼び出す．プロテクションフックは OS の一部として実行される【OS211】．プロテクションフックには保護違反の要因を示すエラーコードがパラメータとして指定される．

2.22.2.1 プロテクションフックの指定

プロテクションフックの使用有無はコンフィギュレーション時に静的に指定する.

プロテクションフックがコンフィギュレーション時に無効とされている場合，OS は保護違反時処理として OS シャットダウンを行う【OS107】. このとき，OS シャットダウンのパラメータとして，違反の区別を示すエラーコードを指定する【NOS0271】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では保護違反時処理から OS シャットダウンされた場合のエラーコードが規定されていないため，本仕様で規定した≪NOS0271≫.

2.22.3 保護違反に対して OS が提供する機能

プロテクションフック内でユーザ定義の処理を実行した後，OS は，プロテクションフックの返り値に応じて，保護違反を起こしたタスク，C2ISR，もしくは保護違反を起こしたタスク，C2ISR が所属する OSAP に対する処理を行う. これを保護違反時要求と呼ぶ. 保護違反に対して OS が提供する機能を以下に示す.

表 2-24 保護違反に対して OS が提供する機能一覧

保護違反に対して OS が提供する機能	SC1	SC2	SC3	SC4
何もしない(PRO_IGNORE)【NOS0909】 ※保護違反要因が以下の場合のみ使用可能.				
E_OS_PROTECTION_EXCEPTION ※CPU 例外要因によっては，PRO_IGNORE を返しても OS シャットダウンを行なってもよい【NOS0910】.	○	○	○	○
E_OS_PROTECTION_ARRIVAL_TASK				
E_OS_PROTECTION_ARRIVAL_ISR				
保護違反を起こしたタスク，C2ISR の強制終了(PRO_TERMINATETASKISR)【OS553】		○	○	○
保護違反を起こしたタスク，C2ISR が所属する OSAP の強制終了(PRO_TERMINATEAPPL)【OS554】			○	○
保護違反を起こしたタスク，C2ISR が所属する OSAP の強制終了と，コンフィギュレーションで指定されたリスタートタスクの起動(PRO_TERMINATEAPPL_RESTART)【OS555】			○	○
OS シャットダウン(PRO_SHUTDOWN)【OS556】	○	○	○	○

AUTOSAR 仕様との違い

AUTOSAR 仕様では，プロテクションフックから PRO_IGNORE を返すことができる保護違反要因は，E_OS_PROTECTION_ARRIVAL のみと規定されている【OS106】．本仕様では，E_OS_PROTECTION_ARRIVAL を拡張したので，PRO_IGNORE を返すことができる保護違反要因も拡張した≪NOS0182≫≪NOS0187≫≪NOS0909≫．また，CPU 例外発生時も，CPU 例外要因によっては無視してよい状況が考えられることから，PRO_IGNORE を返すことができると規定した≪NOS0909≫.

2.22.3.1 保護違反時処理から OS シャットダウンを行う場合のエラーコード

保護違反時処理から OS シャットダウンを行う状況と，終了要因のエラーコードに指定するエラーコードの一覧を表 2-25 に示す.

表 2-25 保護違反時処理から OS シャットダウンを行う場合のパラメータの一覧

保護違反時処理から OS シャットダウンを行う状況	パラメータ
プロテクションフックが無効と指定されている場合	違反の区別を示す エラーコード【NOS0379】
プロテクションフックからの返り値として渡される保護違反時要求が PRO_SHUTDOWN であった場合	
プロテクションフックからの返り値として渡される保護違反時要求に誤りがあった場合	E_OS_PROTECTION_FATAL
プロテクションフック実行中に保護違反が発生した場合	

2.22.3.2 システム定義のシャットダウンフック実行中の保護違反時処理

システム定義のシャットダウンフック実行中に保護違反が発生して呼び出されたプロテクションフックの返り値が, PRO_IGNORE 以外の場合は, システム定義のシャットダウンフックを即座に終了し，シャットダウンを行う【NOS0631】．プロテクションフックが無効で，システム定義のシャットダウンフック実行中に保護違反が発生した場合も，システム定義のシャットダウンフックを即座に終了し，シャットダウンを行う【NOS0912】.

2.22.3.3 OSAP 固有のスタートアップフック実行中の保護違反時処理

OSAP 固有のスタートアップフック実行中に保護違反が発生して呼び出されたプロテクションフックの返り値が，PRO_TERMINATEAPPL の場合は，実行中の OSAP 固有のスタートアップフックを即座に終了し，スタートアップフックが所属する OSAP を終了状態とする【NOS0875】．この後，後続のスタートアップフックは実行され，OS が起動する【NOS0876】.

2.22.3.4 OSAP 固有のエラーフック実行中の保護違反時処理

OSAP 固有のエラーフック実行中に保護違反が発生して呼び出されたプロテクションフックの返り

値が，PRO_TERMINATEAPPL の場合は，実行中の OSAP 固有のエラーフックが所属する OSAP を強制終了する【NOS0877】．プロテクションフックの返り値が，PRO_TERMINATEAPPL_RESTART の場合は，実行中の OSAP 固有のエラーフックが所属する OSAP を強制終了し，リスタートタスクを起動する【NOS0878】．

2.22.3.5 OSAP 固有のシャットダウンフック実行中の保護違反時処理

OSAP 固有のシャットダウンフック実行中に保護違反が発生して呼び出されたプロテクションフックの返り値が，PRO_IGNORE 以外の場合は，実行中の OSAP 固有のシャットダウンフックを即座に終了し，シャットダウンフックが所属する OSAP を終了状態とする【NOS0879】．この後，後続のシャットダウンフックは実行される【NOS0633】．後続のシャットダウンフックに渡されるエラーコードは，保護違反を発生したシャットダウンフックに渡されたエラーコードのまま，変更しない【NOS0882】．プロテクションフックが無効で，OSAP 固有のシャットダウンフック実行中に保護違反が発生した場合も，実行中の OSAP 固有のシャットダウンフックを即座に終了後，シャットダウンフックが所属する OSAP を終了状態とし，後続のシャットダウンフックが実行される【NOS0913】．

AUTOSAR 仕様との違い

AUTOSAR 仕様では，フックルーチン実行中に呼び出されたプロテクションフックの返り値に対する規定はない．本仕様では，フックルーチン毎に，プロテクションフックの返り値によって実行する処理を規定した≪NOS0631≫≪NOS0875≫≪NOS0876≫≪NOS0877≫≪NOS0878≫≪NOS0879≫≪NOS0633≫．

2.22.3.6 信頼関数実行中の保護違反時処理

保護違反時処理の機能レベルが 3 でない場合，かつ信頼関数実行中に保護違反が発生して呼び出されたプロテクションフックの返り値が，PRO_IGNORE 以外の場合は，信頼関数を即座に終了し，シャットダウンを行う【NOS1141】．

保護違反時処理の機能レベルが 3 である場合，かつ信頼関数実行中に保護違反が発生して呼び出されたプロテクションフックの返り値が，PRO_TERMINATEAPPL の場合は，信頼関数を即座に終了し，実行中の信頼関数が所属する OSAP を終了する【NOS1142】．また，プロテクションフックの返り値が，PRO_TERMINATEAPPL_RESTART の場合は，信頼関数を即座に終了し，実行中の信頼関数が所属する OSAP を再起動する【NOS1143】．プロテクションフックの返り値が，PRO_IGNORE，PRO_TERMINATEAPPL，PRO_TERMINATEAPPL_RESTART 以外の場合は，信頼関数を即座に終了し，シャットダウンを行う【NOS1144】．

すべての機能レベルにおいて，プロテクションフックが無効で，信頼関数実行中に保護違反が発生した場合，信頼関数を即座に終了し，シャットダウンを行う【NOS1145】．

2.22.3.7 保護違反を起こした処理単位の取得

保護違反発生時に，プロテクションフックから保護違反を起こした処理単位を取得する方法を，実装

定義で用意することを推奨する【NOS0905】．ATK2 では，保護違反を起こした処理単位を取得するシステムサービスとして，GetFaultyContext を提供する(詳細は 3.9.73 節を参照)【IOS181】〔NOS0905〕．

2.22.3.8 誤った保護違反時要求に対する処理

プロテクションフックから返り値として渡される保護違反時要求に，OS が処理不可能となるような誤りがある場合の OS 処理を表 2-26 に示す．なお，保護違反時要求に誤りがあり，OS シャットダウンを行う場合は E_OS_PROTECTION_FATAL をパラメータに設定する【NOS0272】.

表 2-26 保護違反時要求に誤りがある場合の OS 処理一覧

保護違反時要求	OS 処理
C1ISR，システム定義のスタートアップフック，システム定義のエラーフック，プレタスクフック，ポストタスクフック実行中に発生したプロテクションフックの返り値が以下のもの以外である【NOS0643】.	OS シャットダウン エラーコード： E_OS_PROTECTION_FATAL ≪NOS0272≫
PRO_IGNORE	
PRO_SHUTDOWN	
OSAP 固有のスタートアップフック実行中に発生したプロテクションフックの返り値が以下のもの以外である【NOS0880】.	
PRO_IGNORE	
PRO_SHUTDOWN	
PRO_TERMINATEAPPL	
OSAP 固有のエラーフック，信頼関数実行中に発生したプロテクションフックの返り値が以下のもの以外である【NOS0881】.	
PRO_IGNORE	
PRO_SHUTDOWN	
PRO_TERMINATEAPPL	
PRO_TERMINATEAPPL_RESTART	
プロテクションフックの返り値が PRO_TERMINATEAPPL または PRO_TERMINATEAPPL_RESTART であるにも関わらず， OSAP が定義されていない【OS244】.	
保護違反の要因が E_OS_PROTECTION_ARRIVAL_TASK または E_OS_PROTECTION_ARRIVAL_ISR で， プロテクションフックの返り値が以下のもの以外である【OS506】.	
PRO_IGNORE	
PRO_SHUTDOWN	
プロテクションフックの返り値が PRO_IGNORE で， 保護違反要因が以下のもの以外である【NOS0003】.	
E_OS_PROTECTION_EXCEPTION	
E_OS_PROTECTION_ARRIVAL_TASK	
E_OS_PROTECTION_ARRIVAL_ISR	
保護違反時処理の機能レベルが 1 である場合に， プロテクションフックの返り値が以下である【NOS0625】.	
PRO_TERMINATEAPPL	
PRO_TERMINATEAPPL_RESTART	

保護違反時要求	OS 処理
保護違反時処理の機能レベルが 3 でない場合に，C2ISR 実行中のプロテクションフックの返り値が，PRO_TERMINATETASKISR である【NOS0627】.	
保護違反時処理の機能レベルが 3 でない場合に，信頼 OSAP による保護違反に対するプロテクションフックの返り値が，以下である【NOS1159】.	
PRO_TERMINATEAPPL	
PRO_TERMINATEAPPL_RESTART	
コンフィギュレーション時に，リスタートタスクを設定していないにも関わらず，プロテクションフックの返り値が PRO_TERMINATEAPPL_RESTART である【OS557】.	
プロテクションフックの返り値が未定義である【NOS0164】.	

AUTOSAR 仕様との違い

AUTOSAR 仕様では，保護違反時の OS 処理として何もしないを選択できるのは，保護違反要因が E_OS_PROTECTION_ARRIVAL の場合のみであった【OS475】．本仕様では，プロテクションフックの返り値として，PRO_IGNORE を指定できる保護違反要因を拡張しているため，何もしないを選択できないにも関わらず選択した場合の保護違反要因も変更した≪NOS0003≫.

AUTOSAR 仕様では，プロテクションフックからの返り値が未定義であった場合，プロテクションフックがないものとして処理を行うと記述されている【OS308】．AUTOSAR 仕様では，プロテクションフックの返り値が未定義であった場合の規定が記述されていなかったため，本仕様で明確に規定した≪NOS0164≫.

AUTOSAR 仕様では，機能レベルが規定されていないため，本仕様では機能レベルによって発生する誤った保護違反要求に対する処理を規定した≪NOS0625≫.

AUTOSAR 仕様では，プロテクションフックの返り値が PRO_TERMINATETASKISR であるにも関わらず，対象のタスク，C2ISR が存在しない場合，動作中の OSAP を強制終了し，動作中の OSAP が存在しない場合シャットダウンを行うと規定されている【OS243】．しかし，保護違反を起こした処理単位を特定する方法を用意することで，誤った保護違反時処理要求に対する処理を一律シャットダウンとしたほうが仕様が明確であることから，本規定を削除した． 本仕様では，保護違反を起こした処理単位を取得する方法を，実装定義で用意することを推奨すると規定した≪NOS0905≫.

2.22.4 強制終了処理

2.22.4.1 タスクの強制終了処理

OS はタスクの強制終了処理として，タスクが占有していたリソースをすべて解放し，割込み禁止状態になっていた場合は割込み許可状態に戻す【OS108】．なお，タスクの強制終了時には，ポストタスクフックの呼出しは行わない【NOS0323】．タスクの強制終了後，次にスケジューラが起動するまで，実行状態のタスクが存在しない状態となる【NOS0653】.

マルチコア対応 OS において, タスクがスピンロックを占有していた場合は, すべて解放する【OS613】.

2.22.4.2 C2ISR の強制終了処理

OS は C2ISR の強制終了処理として，C2ISR が占有していたリソースをすべて解放し，割込み禁止状態になっていた場合には割込み許可状態に戻す【OS109】．なお，C2ISR の強制終了時にはフックの呼出しは行わない【NOS0324】.

マルチコア対応 OS において，C2ISR がスピンロックを占有していた場合は，すべて解放する【NOS0927】.

2.22.4.3 OSAP の強制終了処理

OS は，OSAP の強制終了処理として，以下の処理を行う【OS110】.

- OSAP に所属するすべてのタスクと C2ISR の強制終了
- OSAP に所属するすべてのアラームの停止
- OSAP に所属するすべてのスケジュールテーブルの停止
- OSAP に所属するすべての C2ISR の割込み要因を禁止

リスタートオプションが指定され，リスタートタスクが定義されている場合，OS はリスタートタスクを起動する【OS111】.

マルチコア対応 OS においても，強制終了する OSAP が割付けられているコアで，シングルコア OS と同様の強制終了処理を行う【NOS0928】.

2.23 ネストして呼び出された処理単位の振る舞い

本 OS では，タスクや ISR から呼び出されるフックルーチン実行中に，さらに別のフックルーチンが呼び出される場合がある．本節では，本 OS で発生しうるすべての状況における，処理単位の振る舞いについて規定する．

2.23.1 使用可能なシステムサービス

処理単位がネストして呼び出されている状態では，ネストされている処理単位におけるシステムサービスの使用制限が適用される【NOS0506】．例えば，タスク実行中に呼び出されたエラーフックでは，GetTaskState を使用できるが，プロテクションフック実行中に呼び出されたエラーフックにおいては，GetTaskState は使用できない．なお，C1ISR 実行中に発生したフックルーチンから，C1ISR からの発行が動作保証されないシステムサービスを呼び出した場合も，動作保証されない【NOS0644】．

使用上の注意

エラーフック，プロテクションフックからは，GetTaskID，GetISRID を発行できるが，スタートアップ，シャットダウンフックからは発行できない．したがって，スタートアップ，シャットダウンフック実行中に起動したエラーフック，プロテクションフックから，GetTaskID，GetISRID を発行した場合，不正な処理単位からの呼び出しとなる．

2.23.2 実行中のタスクおよび C2ISR

タスクもしくは C2ISR を実行中に，システム定義のエラーフック，プロテクションフック，シャットダウンフック，および OSAP 固有のエラーフック，シャットダウンフックが相互にネストして呼び出されても，GetTaskID，GetISRID で取得する ID は，フックが呼び出される前に実行していたタスク，C2ISR の ID となる【NOS0507】.

ただし，システム定義および OSAP 固有のスタートアップフックと，StartOS 呼出し時に指定された Mode が不正であった場合に呼び出される，システム定義および OSAP 固有のシャットダウンフックにおいては，実行中のタスク，C2ISR は存在しないので，GetTaskID，GetISRID で取得する ID は，INVALID_TASK，INVALID_ISR となる【NOS0508】.

また，タスクもしくは C2ISR から，信頼関数を呼び出した場合，GetTaskID，GetISRID で取得する ID は，信頼関数を呼び出したタスク，C2ISR の ID となる【NOS0509】.

2.23.3 動作中の OSAP

タスク，C2ISR，OSAP 固有のフックルーチンを実行中に，システム定義のエラーフック，プロテクションフック，シャットダウンフックが相互にネストして呼び出されても，GetApplicationID で取得する ID は，フックが呼び出される前に実行していたタスク，C2ISR，OSAP 固有のフックルーチンが所属する OSAP の ID となる【OSa005】.

また，システム定義のスタートアップフックと，StartOS 呼出し時に指定された Mode が不正であっ

た場合に呼び出される，システム定義のシャットダウンフックにおいては，実行中のタスク，C2ISR は存在しないので, GetApplicationID で取得する ID は, INVALID_OSAPPLICATION となる【NOS0511】.

ただし， OSAP 固有のスタートアップフックと，StartOS 呼出し時に指定された Mode が不正であった場合に呼び出される， OSAP 固有のシャットダウンフックにおいては，その OSAP が動作中であるので，GetApplicationID で取得する ID は，呼び出されたフックが所属する OSAP の ID となる【NOS0512】.

2.23.4 OS オブジェクトのアクセス保護

実行中のタスク，C2ISR の有無に関わらず，最後に呼び出されたフックルーチンが，システム定義のフックルーチンである場合は，オブジェクトへのアクセス保護は適用されない【NOS0513】. 最後に呼び出されたフックルーチンが，OSAP 固有のフックルーチンである場合は，そのフックルーチンが所属する OSAP のアクセス保護が適用される【NOS0514】.

プレタスクフック，ポストタスクフックは，システム定義のみであるため，アクセス保護は適用されない【OSa006】.

2.23.5 メモリ保護

実行中のタスク，C2ISR の有無に関わらず，最後に呼び出されたフックルーチンが，システム定義のフックルーチンである場合は，メモリ保護は行わない【NOS0516】. 最後に呼び出されたフックルーチンが，OSAP 固有のフックルーチンである場合は，そのフックルーチンが所属する OSAP のメモリ保護を行う【NOS0517】.

プレタスクフック，ポストタスクフックは，システム定義のみであるため，メモリ保護は行わない【OSa007】.

以上の仕様をまとめた一覧を表 2-28 に示す.（表 2-28 の凡例を表 2-27 に示す）
"or"で複数の処理単位を規定している箇所は，いずれの処理単位でも同じ結果であることを指し，かつ次にネストする処理単位とは異なる処理単位の組み合わせであることを示す. 例えば，2 行目から 4 行目の意味は，システム定義のスタートアップフック実行中に，システム定義のエラーフック，プロテクションフック，シャットダウンフックのいずれかが呼び出され，さらにそこから，システム定義のエラーフック，プロテクションフック，シャットダウンフックのいずれかが呼び出された状態を示すが，この場合，エラーフック→エラーフックや，プロテクションフック→プロテクションフックなど，同じ処理単位がネストして呼び出される組み合わせは含まないという意味である.

また, 表中のアルファベットは, その仕様の根拠となる仕様番号への参照であり, 表 2-27 の凡例に, 各アルファベットが対応する仕様番号を合わせて示す.

表 2-27 表 2-28 の凡例

表記	意味
①	処理レベル（網掛け：OS 処理レベル，無地：呼び出し元の処理単位の処理レベル）
②	実行中タスク/C2ISR の有無（網掛け：有り，無地：無し）
③	動作中 OSAP の有無（網掛け：有り，無地：無し）
④	OS オブジェクトのアクセス保護（網掛け：有り，無地：無し）
⑤	メモリ保護（網掛け：有り，無地：無し）
[SD]	システム定義
[OS]	OSAP 固有
ERR	エラーフック
PRO	プロテクションフック
STA	スタートアップフック
SHU	シャットダウンフック
PPT	プレタスクフック，ポストタスクフック
ACB	アラームコールバック
TRF	信頼関数

表記	対応する仕様番号
A	≪NOS0243≫
B	≪NOS0242≫
C	≪COS0925≫
D	≪OS365≫
E	≪OS364≫
F	≪NOS0508≫
G	≪COS3271≫≪OS263≫
H	≪NOS0507≫
I	≪COS1163≫
J	≪OS263≫
K	≪NOS0509≫
L	≪NOS0511≫
M	≪NOS0512≫

表記	対応する仕様番号
N	≪OSa005≫
O	≪OSa006≫
P	≪NOS0395≫
Q	≪NOS0513≫
R	≪NOS0514≫
S	≪OS056≫
T	≪OSa007≫
U	≪NOS0325≫
V	≪NOS0516≫
W	≪NOS0517≫
X	≪OS265≫
Y	≪OS361≫

表 2-28 ネストして呼び出された処理単位に関する規定

ネスト 3 ネスト 2 ネスト 1 実行中の処理単位				①	②	③	④	⑤
[SD]STA				A	F	L	Q	V
	[SD]ERR or [SD]PRO or [SD]SHU	[SD]ERR						
		[SD]PRO						
		[SD]SHU						
	[SD]ERR or [SD]PRO or [OS]SHU	[SD]ERR				N		
		[SD]PRO						
		[OS]ERR				M	R	W
		[OS]SHU						
[OS]STA						M	R	W
	[SD]ERR or [SD]PRO or [SD]SHU or [OS]ERR or [OS]SHU	[SD]ERR				N	Q	V
		[SD]PRO						
		[SD]SHU						
		[OS]ERR				M	R	W
		[OS]SHU						
[SD]SHU ※1						L	Q	V
	[SD]ERR or [SD]PRO	[SD]ERR						
		[SD]PRO						
[OS]SHU ※1						M	R	W
	[SD]ERR or [SD]PRO or [OS]ERR	[SD]ERR				M	Q	V
		[SD]PRO						
		[OS]ERR				M	R	W
タスク				B	G	N	S	※5
	[SD]ERR or [SD]PRO or [SD]SHU or [OS]ERR or	[SD]ERR		A	H		Q	V
		[SD]PRO						
		[SD]SHU						
		[OS]ERR					R	W

ネスト 3 ネスト 2 ネスト 1 実行中の 処理単位				①	②	③	④	⑤
	[OS]SHU	[OS]SHU						
	PPT(※6)				I	N	O	T
		[SD]ERR or [SD]PRO or [SD]SHU or [OS]ERR or [OS]SHU	[SD]ERR		H		Q	V
			[SD]PRO					
			[SD]SHU					
			[OS]ERR				R	W
			[OS]SHU					
	ACB			C	※2	※3		
		[SD]ERR or [SD]PRO or [SD]SHU	[SD]ERR	A				
			[SD]PRO					
			[SD]SHU					
	TRF			D	K	P	U	X
		[SD]ERR or [SD]PRO or [SD]SHU or [OS]ERR or [OS]SHU	[SD]ERR	A	H	N	Q	V
			[SD]PRO					
			[SD]SHU					
			[OS]ERR				R	W
			[OS]SHU					
C2ISR				B	G	N	S	※5
	[SD]ERR or [SD]PRO or [SD]SHU or [OS]ERR or [OS]SHU	[SD]ERR		A	J		Q	V
		[SD]PRO						
		[SD]SHU						
		[OS]ERR					R	W
		[OS]SHU						
	ACB			C	※2	※3		
		[SD]ERR or [SD]PRO or [SD]SHU	[SD]ERR	A				
			[SD]PRO					
			[SD]SHU					
	TRF			E	K	P	U	X

<table>
<tr><td colspan="4">ネスト 3
ネスト 2
ネスト 1
実行中の
処理単位</td><td>①</td><td>②</td><td>③</td><td>④</td><td>⑤</td></tr>
<tr><td rowspan="5"></td><td rowspan="5"></td><td rowspan="5">[SD]ERR or
[SD]PRO or
[SD]SHU or
[OS]ERR or
[OS]SHU</td><td>[SD]ERR</td><td rowspan="5">A</td><td rowspan="5">J</td><td rowspan="5">N</td><td rowspan="2">Q</td><td rowspan="2">V</td></tr>
<tr><td>[SD]PRO</td></tr>
<tr><td>[SD]SHU</td></tr>
<tr><td>[OS]ERR</td><td rowspan="2">R</td><td rowspan="2">W</td></tr>
<tr><td>[OS]SHU</td></tr>
<tr><td>C1ISR</td><td></td><td></td><td></td><td>B</td><td colspan="3">※4</td><td>Y</td></tr>
</table>

※1：StartOS 呼出し時に Mode が不正であった場合に呼び出されるシャットダウンフックを指す
※2：GetTaskID，GetISRID は，アラームコールバックから呼び出し不可
※3：SC3，SC4 では，アラームコールバックをサポートしない
※4：C1ISR から呼出し可能なシステムサービスとは無関係である
※5：タスク，C2ISR に対するメモリ保護機能である
※6：信頼関数から TerminateTask を発行した場合のポストタスクフックも同様の振る舞いである

2.24 システムの初期化と終了

2.24.1 システムの初期化手順

ターゲットリセット後の初期化処理はターゲットに依存するため，ユーザは OS を起動する前にターゲットハードウェアに依存した初期化処理を行う必要がある【COS1142】.

本 OS では，OS を起動するシステムサービスとして StartOS を提供する(詳細は 3.9.2 節を参照)．ユーザはターゲット初期化処理が終了した後，StartOS に指定するアプリケーションモードを決定し，StartOS の呼び出しを行う．

本 OS が提供するシステムの初期化手順を以下に示す．（＜＞は図 2-51 の処理に対応する）

＜OS 起動要求＞

(1)　ユーザはアプリケーションモードを指定して，OS の起動処理である StartOS を呼び出す.

＜OS 内部初期化処理＞

(2)　OS は，OS の内部初期化処理を実行する【COS1146】.

＜スタートアップフック実行＞

(3)　OS はコンフィギュレーションでシステム定義のスタートアップフックが有効な場合, スタートアップフックを呼び出す【COS1147】．スタートアップフック実行中は OS 割込み禁止状態である【COS1149】.

(4)　SC3，SC4 で OSAP 固有のスタートアップフックが有効な場合， OSAP 固有のスタートアップフックを呼び出す≪OS236≫．OSAP が複数定義されている場合，OSAP 固有のスタートアップフックの呼出し順は実装定義とする≪NOS0128≫.

(5)　ユーザは，任意の初期化処理をシステム定義のスタートアップフック，OSAP 固有のスタートアップフック内で実行することができる【COS1148】．スタートアップフック内で，アプリケーションモードによって異なった処理を実行するために，OS はアプリケーションモードを取得する機能として GetActiveApplicationMode を提供する(詳細は 3.9.1 節を参照).

＜OS 実行開始＞

(6)　OS は，指定されたアプリケーションモードに従って自動起動するタスク，アラーム，スケジュールテーブルを起動状態とする【NOS0669】．スケジューリングを開始すると共に OS 割込み許可状態にする【COS1150】．同一のタスク優先度に指定されたタスクの起動順序は規定しない【COS1152】.

＜タスク，ISR 実行＞

(7)　OS は OS 起動処理を終了し，タスク，ISR の実行を開始する．ユーザは OS 初期化後に動作する特別なタスクを定義する必要はない【COS1134】.

OSEK 仕様との違い

OSEK 仕様では，自動起動するタスクを起動した後で，アラームを起動すると規定されている【COS1151】．しかし，オブジェクトの起動順序がユーザから観測できないため，順序に関する規定を削除した≪NOS0669≫.

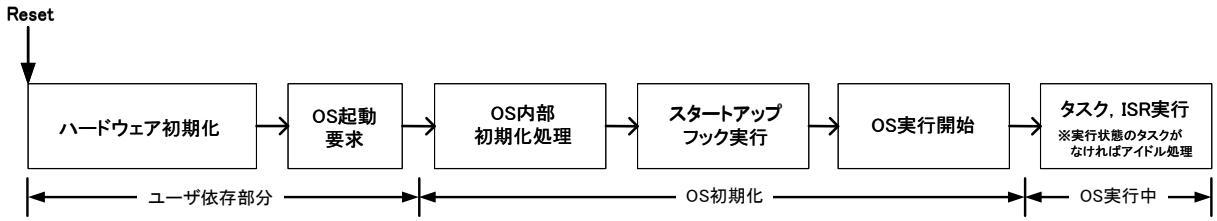


図 2-51 システム初期化シーケンス

2.24.2 オブジェクトの自動起動

本 OS では，タスク，アラーム，スケジュールテーブルの自動起動指定によって，OS 初期化後に直ちに動作するタスク，アラーム，スケジュールテーブルを指定することができる【COS1135】.

2.24.3 システムの終了手順

本 OS では, OS を終了するためのシステムサービスとして ShutdownOS を提供する(詳細は 3.9.3 節を参照)．またフェイタルエラーが発生した場合，OS は OS 内部からの要求でシャットダウンする可能性がある【COS1155】.

ポストタスクフックが有効な場合でも，シャットダウン処理では OS はポストタスクフックを呼び出さない【OS071】.

OS がシャットダウンを要求された場合，シャットダウンフックを呼び出した後に OS の終了を行う【COS1156】．SC3，SC4 では OSAP 固有のシャットダウンフックが有効な場合，システム定義のシャットダウンフックの前に OSAP 固有のシャットダウンフックを呼び出す≪OS237≫．OSAP が複数定義されている場合，OSAP 固有のシャットダウンフックの呼出し順は実装定義とする≪NOS0127≫.

システム定義のシャットダウンフック内ではユーザはいかなる振る舞いを実装してもよく，システム定義のシャットダウンフックからリターンしなくてもよい【COS1157】．OS はシステム定義のシャットダウンフックがリターンした後，割込みをすべて禁止し，無限ループする【OS425】.

2.24.4 マルチコアシステムの初期化と終了

コアが動作を開始することを単に「起動する」と呼び，OS が動作する状態にすることを「OS を起動する」と呼ぶ.

マスタ／スレーブ構成

マルチコアシステムは，ハードウェアに依らず，システム起動時に唯一起動するマスタコアと，他のコアから起動される 1 つ以上のスレーブコアが存在するマスタ／スレーブ構成とする．ハードウェアでマスタ／スレーブ構成がサポートされていなくても，ソフトウェアでこの構成をエミュレートする【OS577】．ただし，エミュレートされたスレーブコアは，他のコアから起動される前に main 関数を実行してはならない【OS578】.

マスタコアは，他のコアを起動することができる【OS574】〔OS676〕〔OS682〕．同様にスレーブコアも，他のコアを起動することができる【OS575】〔OS676〕〔OS682〕．ハードウェア上のすべてのコアを，本 OS を使用するコアとして起動しなくてもよい【OS576】〔OS682〕．本 OS を使用可能とするコアの上限数は，コンフィギュレーション時に静的に指定する．また，指定しない場合は，コアの上限数を 1 とする【OS583】.

バリア同期

OS 上で動作するアプリケーション(タスク，ISR)の初期状態における振る舞いの再現性を確保するために，システム初期化，終了処理では必要なタイミングで同期を行う．具体的には，マルチコアシステムを構成するすべてのコアで，所定の処理が完了するのを待ち合わせる方法により同期する．このような同期方法を「バリア同期」と呼ぶ.

OS 管理外のコア

本 OS を使用しないが，他の OS や OS レスのアプリケーションを使用するなどの目的で起動するコアを，「OS 管理外のコア」と呼ぶ．本 OS は，OS 管理外のコアを起動するシステムサービスとして，StartNonAutosarCore を提供する(詳細は 3.9.65 節を参照)【OS584】〔OS682〕．StartOS を呼び出す前のコアからでも，呼び出した後のコアからでも，OS 管理外のコアを起動することができる【OS585】．OS 管理外のコアから，本 OS が提供するシステムサービスを呼び出し場合の動作は保証しない【OSa121】.

2.24.4.1 マルチコアシステムの初期化手順

本 OS が提供するマルチコアシステムの初期化手順を以下に示す.

(1) マスタコアが，本 OS を使用するスレーブコアを StartCore により起動する【OSa122】.
(2) マスタコアから StartOS を呼び出し，マスタコアの OS を起動する【OSa123】.
(3) 起動したスレーブコアは，他に本 OS を使用するスレーブコアがあれば StartCore により起動し，なければ，StartOS によりスレーブコアの OS を起動する【OSa124】.

(4) StartOS の呼び出しにより，マスタコアと，起動されたすべてのスレーブコアにおいて，OS オブジェクトの初期化を行う【OSa125】．IOC を使用する場合，IOC の初期化も行う【OS611】.
(5) バリア同期(同期 1)を行う【OS580】.
(6) システム定義のスタートアップフックが有効な場合，すべてのコアでシステム定義のスタートアップフックを呼び出す【OS581】.
(7) スタートアップフックを呼び出した後，引き続き，各コアに割付けられた OSAP の OSAP 固有のスタートアップフックが有効な場合，OSAP 固有のスタートアップフックを，それぞれのコアで非同期に呼び出す【OS582】.
(8) バリア同期(同期 2)を行う【OS579】.
(9) 各コアでタスク，ISR の実行を開始する【OSa126】.

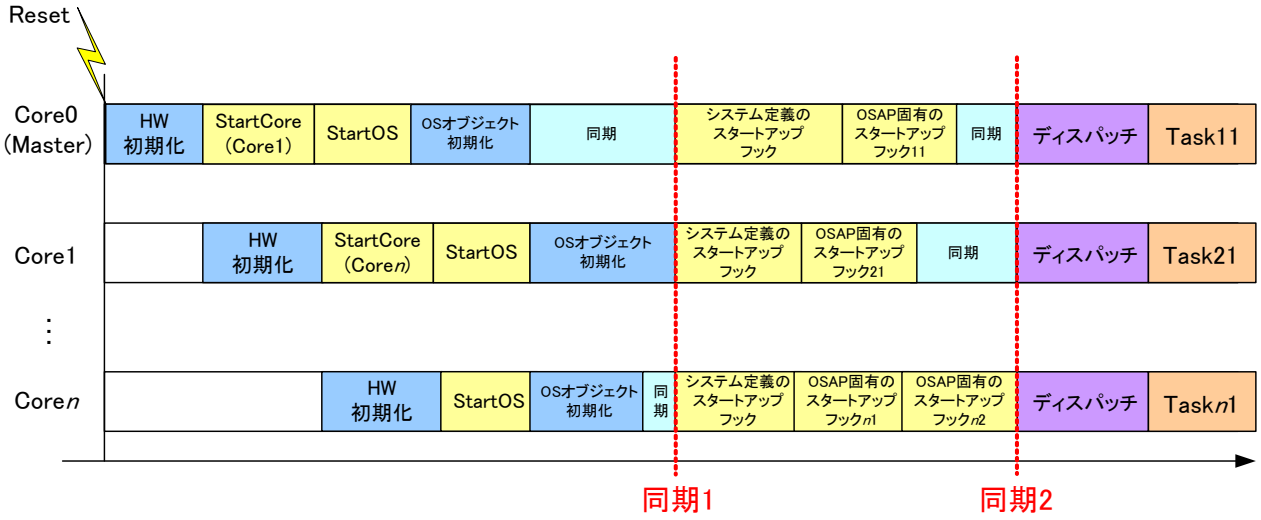


図 2-52 マルチコアシステムの初期化手順

2.24.4.2 マルチコアシステムの終了手順

本 OS では，2 種類のマルチコアシステムの終了手順を提供する．1 つは，ShutdownAllCores によりすべてのコアを同時に終了する手順で,「同期 OS シャットダウン」と呼ぶ. もう 1 つは, ShutdownOS によりそれぞれのコアを終了する手順で,「個別 OS シャットダウン」と呼ぶ．別に規定がない限り，本仕様の「OS シャットダウン」に関する仕様は，マルチコア対応 OS においては，「同期 OS シャットダウン」を指す.

同期 OS シャットダウン

本 OS が提供するマルチコアシステムの同期 OS シャットダウン手順を以下に示す.

(1) 本 OS が起動しているいずれかのコア(マスタコアとは限らない)で, ShutdownAllCores を呼び出す【OS621】.

(2) ShutdownAllCores が呼び出されたコアから，他のすべてのコアに OS 終了要求が通知される【OSa127】〔OS714〕.
(3) ShutdownAllCores を呼び出したコア，および OS 終了要求を受理したコアで，各コアに割付けられた OSAP の OSAP 固有のシャットダウンフックが有効な場合, OSAP 固有のシャットダウンフックを，それぞれのコアで非同期に呼び出す【OS586】.
(4) バリア同期を行う【OS587】.
(5) システム定義のシャットダウンフックが有効な場合，すべてのコアでシステム定義のシャットダウンフックを呼び出す【OS588】.
(6) すべてのコアで OS オブジェクトクリーンアップを行う【NOS0929】.
(7) 必要に応じて，すべてのコアでハードウェアの終了処理を行う【NOS0930】.
(8) すべてのコアを無限ループへ移行する【OSa128】.

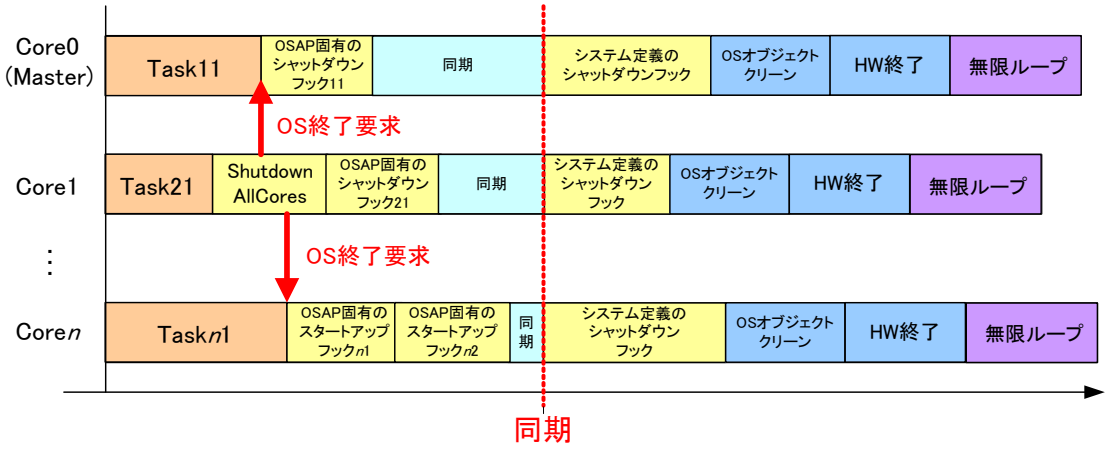


図 2-53 同期 OS シャットダウンによるマルチコアシステム終了手順

個別 OS シャットダウン

本 OS が提供するマルチコアシステムの個別 OS シャットダウン手順を以下に示す.

(1) いずれかのコア(マスタコアとは限らない)で，ShutdownOS を呼び出す【OS616】.
(2) ShutdownOS を呼び出したコアのみ，シングルコアと同様に OS シャットダウンを行う(詳細は 2.24.3 節を参照)【OS617】.
(3) すべてのコアで ShutdownOS を呼び出された時点で，システム終了となる【NOS0931】.

一度，ShutdonwOS を発行したコアに割付いているタスクや，OSAP は起動することはできない【OS618】. ShutdonwOS は，同時に複数のコアで呼び出すことができる【OS619】. ShutdownOS を呼び出したコアに，スピンロックを占有している処理単位が存在する場合，スピンロックを解放する【OS620】.

2.25 IOC(Inter-OS-Application Communicator)

IOC は，コアやメモリ保護の境界を跨いで通信するために用意された OSAP 間のデータ通信機能である．したがって，マルチコア対応 OS，もしくはメモリ保護機能をサポートしたシングルコア OS で使用することができる【OSa151】．メモリ保護機能をサポートしないシングルコア OS では，IOC はサポートしなくてもよい【OSa152】．

IOC は，OS の機能として提供され，RTE による OSAP 内通信および，COM-Stack による ECU 間通信とは別のデータ通信機能である【OS671】．

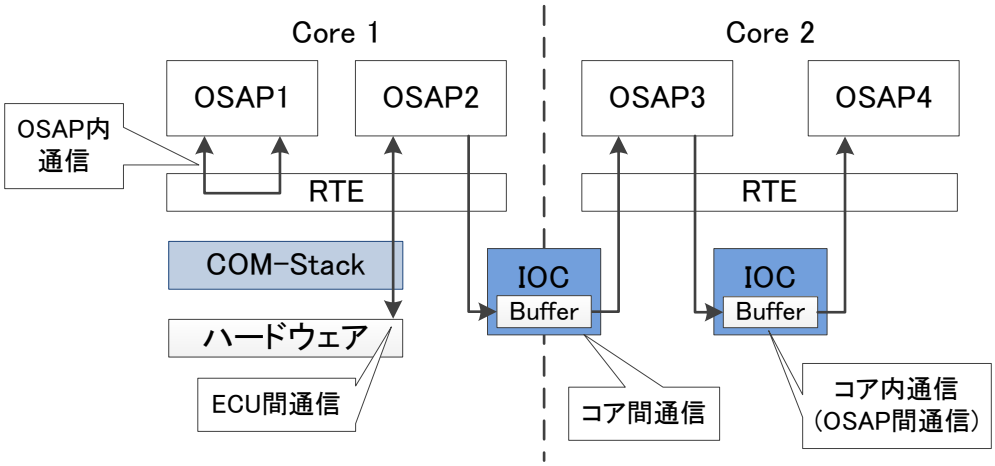


図 2-54 IOC によるデータ通信

2.25.1 IOC の構成

IOC は，センダ，レシーバ，IOC バッファで構成される．IOC は，IOC ID で識別する．

センダ

センダは，IOC によるデータ通信の送信側となる処理単位の集合である．センダは，OSAP 単位で管理され，1 つのセンダは 1 つの OSAP に所属する≪MCOS_Conf1014≫．センダは，1 つの IOC に 1 つ以上存在するため，センダ ID で識別する．

レシーバ

レシーバは，IOC によるデータ通信の受信側となる処理単位の集合である．レシーバは，OSAP 単位で管理され，1 つのレシーバは 1 つの OSAP に所属する≪MCOS_Conf1012≫．レシーバは，1 つの IOC に 1 つのみ存在するため，識別する必要はない．

IOC バッファ

IOC バッファは，センダが送信データを格納するメモリ領域である．レシーバは，IOC バッファから

データを取得することで，センダが送信したデータを受信する【OSa153】. IOC によるデータ通信では，センダとレシーバが，異なるコアや異なる OSAP であっても，センダとレシーバからアクセス可能なメモリ領域を確保する【OSa154】. IOC によるデータ通信では，センダがデータを送信してから，レシーバがデータを受信するまで，データの一貫性が保証される【OSa155】.

IOC では，コア間のエンディアンの変換はサポートしない【OSa156】.

2.25.2 IOC の種別

2.25.2.1 センダの数

1 つの IOC にはセンダは 1 つ以上存在する≪MCOS_Conf1015≫. 1 つの IOC に 1 つのセンダのみ存在するデータ通信を，1:1 通信と呼ぶ. 1 つの IOC に 2 つ以上のセンダが存在するデータ通信を，N:1 通信と呼ぶ.

なお，1 つの IOC にはレシーバは 1 つのみ存在する≪MCOS_Conf1017≫. 複数のレシーバで受信する 1:N 通信は，RTE によってサポートされる【OSa157】.

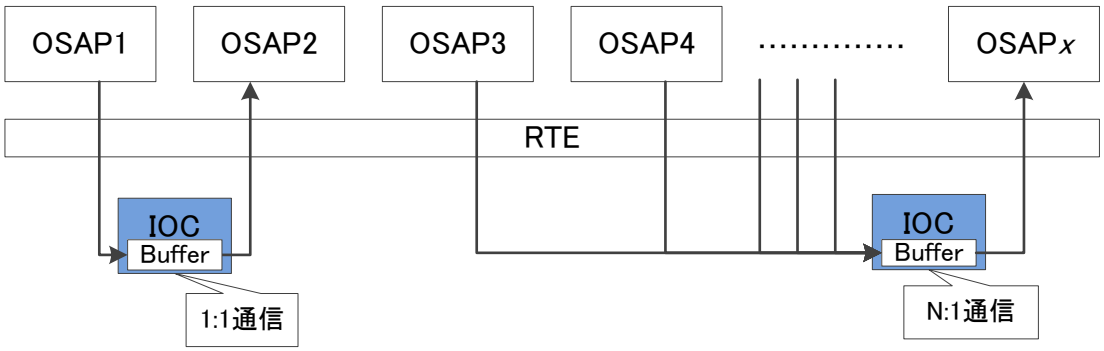


図 2-55 1:1 通信と N:1 通信

1:1 通信において，センダとレシーバが同じ OSAP に所属する場合，RTE による OSAP 内通信を使用したほうが効率的であるので，1:1 通信の場合は，センダとレシーバを同じ OSAP に所属させることはできない【OSa158】. ただし，N:1 通信の場合は，複数のセンダの中に，レシーバと同じ OSAP に所属するセンダが存在してもよい【OSa159】.

2.25.2.2 キューの有無

IOC は，キューあり通信と，キューなし通信をサポートする【OSa160】〔MCOS_Conf1001〕.

キューあり通信の IOC では，IOC バッファにキュー管理領域を持ち，通信データを FIFO で送受信する【OSa161】. キュー管理領域のキューサイズは，コンフィギュレーション時に静的に指定する≪MCOS_Conf1001≫.

キューなし通信の IOC では，キュー管理領域を持たず，常に最後に送信されたデータが，IOC バッファに格納される【OSa162】.

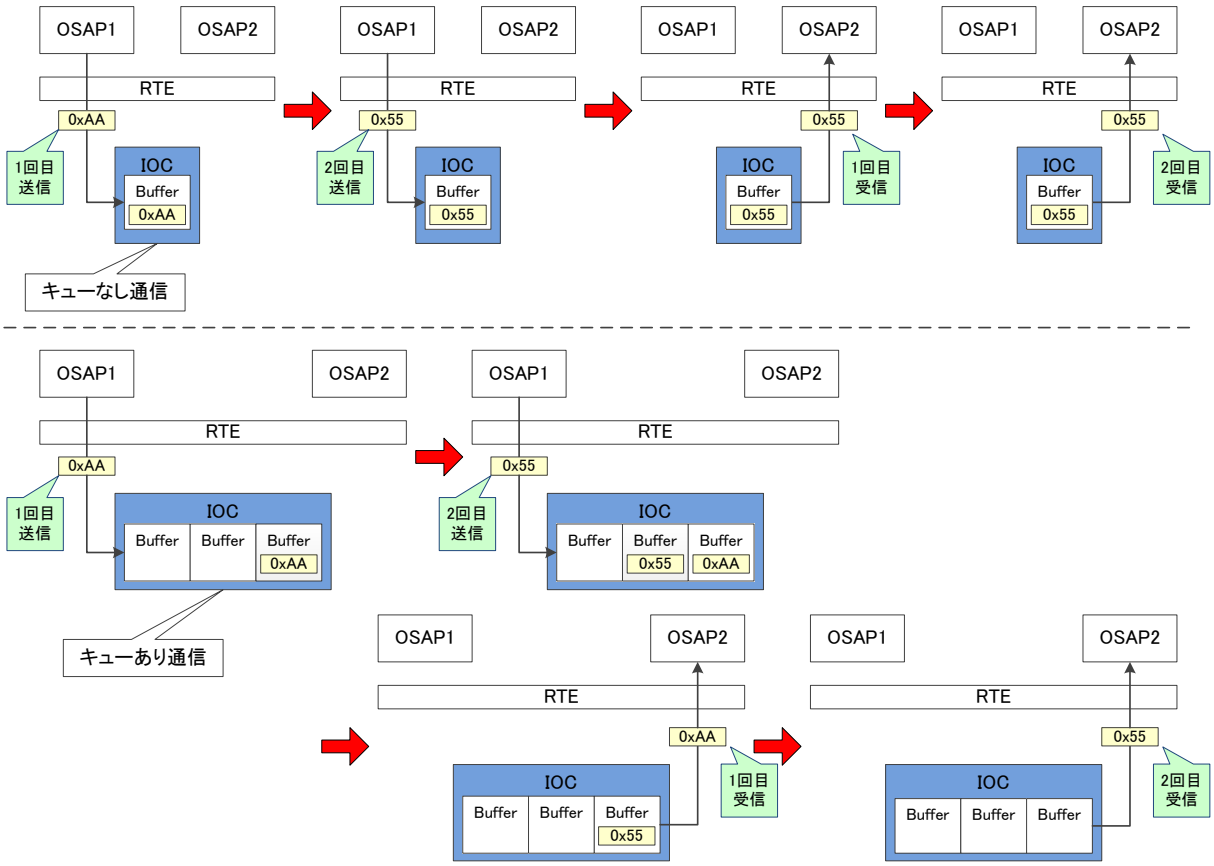


図 2-56 キューなし通信とキューあり通信

2.25.2.3 データの数

IOC は，単一通信と，グループ通信をサポートする【OSa163】〔MCOS_Conf1023〕.

単一通信は，1 回の送受信処理にて，1 つのデータのみを送受信処理する【OSa164】〔OS718〕〔OS738〕. 単一通信は，1:1 通信，N:1 通信のどちらでも使用できる≪MCOS_Conf1015≫.

グループ通信は，1 回の送受信処理にて，2 つ以上のデータを送受信処理する【OSa165】〔OS728〕〔OS746〕. グループ通信は，1:1 通信でのみ使用できる≪MCOS_Conf1015≫.

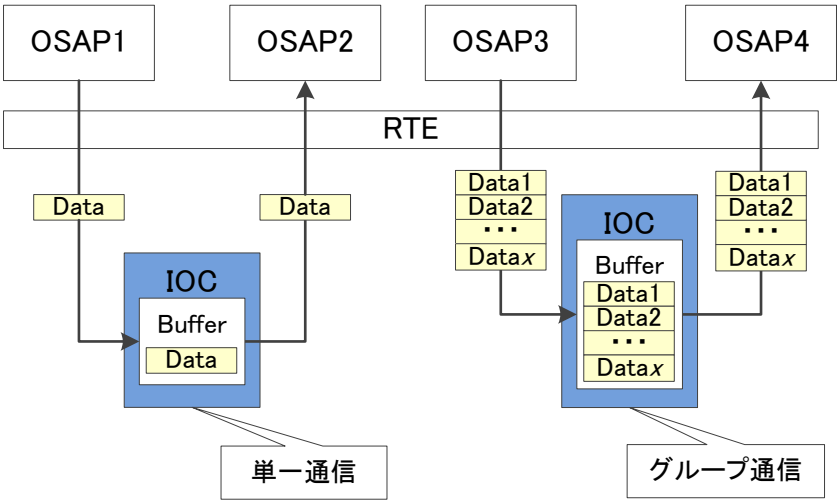


図 2-57 単一通信とグループ通信

2.25.3 IOC の操作

本 OS は，IOC を操作するための API として，IOC 用システムサービスを提供する．IOC 用システムサービスは，IOC 用システムサービス名に IOC ID やセンダ ID が含まれ，IOC 用システムサービス自体が，ジェネレータによって生成される【OSa166】．生成した IOC 用システムサービスを使用するために必要な定義や宣言は，IOC ヘッダファイル(Ioc.h)として出力する【OSa167】.

IOC 用システムサービスは，OS 処理レベルで実行される【OSa168】.

2.25.3.1 データ送信

本 OS は, IOC に対してデータを送信するための IOC 用システムサービスとして, IOC の種別毎に, 以下を提供する．<IocId>には IOC ID が入り，<SenderId>にはセンダ ID が入る．1:1 通信の場合，[_<SenderId>]は省略される.

キューあり通信，単一通信

IocSend_<IocId>[_<SenderId>] （詳細は 3.10.1 節を参照）

キューなし通信，単一通信

IocWrite_<IocId>[_<SenderId>] （詳細は 3.10.2 節を参照）

キューあり通信，グループ通信

IocSendGroup_<IocId> （詳細は 3.10.3 節を参照）

キューなし通信，グループ通信

IocWriteGroup_<IocId> （詳細は 3.10.4 節を参照）

2.25.3.2 データ受信

本 OS は，IOC からデータを受信するための IOC 用システムサービスとして，IOC の種別毎に，以下を提供する．<IocId>には IOC ID が入る．

キューあり通信，単一通信

IocReceive_<IocId> （詳細は 3.10.5 節を参照）

キューなし通信，単一通信

IocRead_<IocId> （詳細は 3.10.6 節を参照）

キューあり通信，グループ通信

IocReceiveGroup_<IocId> （詳細は 3.10.7 節を参照）

キューなし通信，グループ通信

IocReadGroup_<IocId> （詳細は 3.10.8 節を参照）

2.25.3.3 キューの初期化

キューあり通信の IOC において，キューに格納されたすべてのデータを削除し，キューを空の状態にすることを，キューの初期化と呼ぶ．本 OS は，キューあり通信の IOC に対して，キューを初期化するための IOC 用システムサービスとして，IocEmptyQueue_<IocId>を提供する(詳細は 3.10.9 節を参照)．<IocId>には IOC ID が入る．

キューの初期化は，レシーバのみが実行することができる【NOS1075】.

2.25.4 通信データ

IOC では，任意のデータ型のデータを通信することができる【OSa169】．IOC で通信するデータ型は，コンフィギュレーション時に静的に指定する≪MCOS_Conf1005≫.

コンフィギュレーション時に指定したデータ型が，引数のデータ型となるように，IOC 用システムサービスが，ジェネレータによって生成される≪OSa166≫.

2.25.4.1 汎整数型

汎整数型を IOC のデータ型として指定した場合，データ送信用の IOC 用システムサービスの引数は値渡しとなる≪OS723≫≪OS733≫.

2.25.4.2 汎整数型以外のデータ型

汎整数型以外のデータ型を IOC のデータ型として指定した場合，データ送信用の IOC 用システムサービスの引数は参照渡しとなる≪OS723≫≪OS733≫．なお，汎整数型かどうかは，IOC のデータ型の

参照先(OsIocDataTypeRef)の IMPLEMENTATION-DATA-TYPE で決定する．(詳細は，Software Component Template 仕様を参照)

2.25.5 IOC 用システムサービスに対するサービス保護

ユーザが返り値の IOC 用システムサービスを呼出し，実行の結果，返り値が IOC_E_OK でない場合，かつシステム定義のエラーフックが有効な場合，システム定義のエラーフックが呼び出される【NOS1163】．また，OSAP 固有のエラーフックが有効な場合，エラーが発生した OS オブジェクトが所属する OSAP の，OSAP 固有のエラーフックが呼び出される【NOS1164】.

2.25.5.1 不正な処理単位からのシステムサービス呼出し

IOC 用システムサービスは，タスクと C2ISR からのみ呼び出すことができる【NOS1077】．タスクと C2ISR 以外の処理単位から呼び出した場合，IOC 用システムサービスは IOC_E_NOK を返す【NOS1078】.

2.25.5.2 割込み禁止状態でのシステムサービス呼出し

IOC 用システムサービスは，割込み禁止状態では，呼び出すことができない【NOS1080】．割込み禁止状態で呼び出した場合，IOC 用システムサービスは IOC_E_NOK を返す【NOS1081】.

2.25.5.3 センダ，レシーバが所属する OSAP 以外からの呼出し

IOC 用システムサービスは，IOC 用システムサービス名に含まれる IOC のセンダ，レシーバが所属する処理単位からのみ呼び出すことができる【NOS1083】．N:1 通信におけるデータ送信の IOC 用システムサービスに関しては，IOC 用システムサービス名に含まれるセンダが所属する処理単位からのみ呼び出すことができる【NOS1084】.

各センダ，レシーバ用に生成された IOC 用システムサービスを，センダ，レシーバが所属する OSAP と異なる OSAP の処理単位から呼び出した場合，IOC 用システムサービスは IOC_E_NOK を返す【NOS1085】.

2.25.5.4 エラーを起こした IOC 用システムサービスの情報の取得

通常のシステムサービス同様，IOC 用システムサービスの呼出しによって起動したエラーフックからも，システムサービス ID の取得およびシステムサービスパラメータの取得が可能である【NOS1087】.

システムサービス ID は，OSEK 仕様で規定された命名規則とは異なり，AUTOSAR 仕様で規定されている(詳細は 3.6.9 節を参照).

IOC 用システムサービスでは，コンフィギュレーションによって引数が変化するので，システムサービスパラメータとして，IOC 用システムサービス名に含まれる IOC ID およびセンダ ID を取得するマクロを提供する(詳細は 3.5.4 節を参照)【NOS1088】.

2.25.6 IOC 用システムサービスに対するメモリ保護

IOC 用システムサービスに渡されたアドレスが，呼出し元の OSAP からアクセスできないメモリ領域を指している場合の振る舞いは実装定義である【NOS1109】．ATK2 では，IOC 用システムサービスに渡されたアドレスのメモリ領域へアクセスする際に，メモリ保護違反が発生する【IOS225】．なお，キューあり通信の IOC において，データ受信用の IOC 用システムサービスを呼び出しにより，メモリ保護違反が発生した場合，データの受信(書込み)に失敗するが，キューからは 1 つデータが取り出される【IOS226】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，IOC にデータが送信されたタイミングで，レシーバ側にコールバック関数を呼び出す，データ受信通知機能をオプションとして提供すると規定されている【OSa170】.

- ReceiverPullCB の概要【OS757】
- 関数名のコンフィギュレーション方法【OS758】
- コールバック関数名【OS759】
- コールバック関数の処理レベル【OS760】
- データ受信通知のサポート範囲【OS761】

しかし，本仕様では，以下の理由からデータ受信通知機能はサポートしない.

- データ受信通知機能に関する仕様は未完成である【OSa171】.
- ActivateTask によるレシーバ側のタスク起動や，コア間割込みによる通知で同等の機能の実現が可能である.

2.26 スピンロック

スピンロックは異なるコアに割付けられた処理単位間で排他制御を行うための OS オブジェクトであり，マルチコア対応 OS でのみ使用する．スピンロックはスピンロック ID によって識別する．

スピンロックを使用して排他区間を開始することをスピンロックの獲得と呼び，排他区間を終了することをスピンロックの解放と呼ぶ. また, 処理単位がスピンロックを獲得して, 排他状態にあることを, スピンロックの占有と呼ぶ．スピンロックはすべてのコンフォーマンスクラスで使用できる【NOS0932】.

2.26.1 スピンロックの種別

スピンロックオブジェクトに対しては，標準スピンロック，OS 割込み禁止スピンロック，全割込み禁止スピンロックのいずれかを指定することができる．スピンロック獲得後に，スピンロックを獲得した処理単位が割り付けられたコアの割込み禁止状態が変化すること以外は，どのスピンロックを指定しても排他制御の振る舞いは同様である．

標準スピンロック

スピンロック獲得後，スピンロックを獲得した処理単位が割り付けられたコアの割込み禁止状態を変更しないスピンロック．

OS 割込み禁止スピンロック

スピンロック獲得と同時に，スピンロックを獲得した処理単位が割り付けられたコアを，OS 割込み禁止状態とするスピンロック．

全割込み禁止スピンロック

スピンロック獲得と同時に，スピンロックを獲得した処理単位が割り付けられたコアを，全割込み禁止状態とするスピンロック．

2.26.2 スピンロックの操作

2.26.2.1 スピンロックの獲得

本 OS は，スピンロックを獲得するためのシステムサービスとして GetSpinlock を提供する(詳細は 3.9.66 節を参照)．GetSpinlock によって獲得しようとしたスピンロックが，他のコアの処理単位によって既に獲得されている場合，スピンロックが獲得できるまで，スピン(実行状態のままビジーウェイト)する【OS649】〔OS687〕.

本 OS は，スピンロックを獲得するためのシステムサービスとして TryToGetSpinlock を提供する(詳細は 3.9.68 節を参照)．TryToGetSpinlock によって獲得しようとしたスピンロックが，他のコアの処理単位によって既に獲得されている場合，スピンせずに即リターンする【OS652】.

本 OS は，タスクが獲得しようとしたスピンロックが，既に同じコアに割付けられている処理単位に

よって占有されている場合，エラーとする【OS658】〔OS690〕．同様に C2ISR 獲得しようとしたスピンロックが，既に同じコアに割付けられている処理単位によって占有されている場合も，エラーとする【OS659】〔OS690〕．これは，図 2-58 に示すような，デッドロックが発生するのを避けるためである．

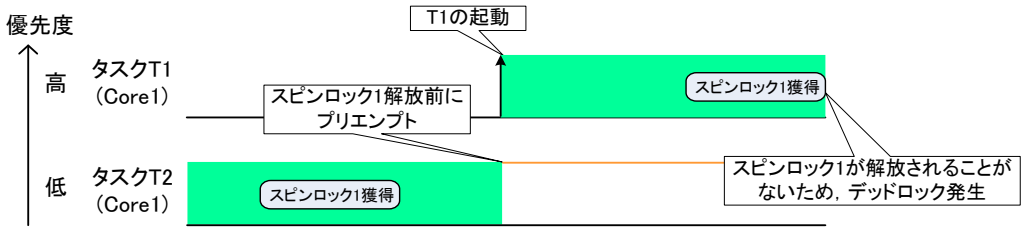


図 2-58 スピンロックによるデッドロックの例 1

また，複数のスピンロックをネストして獲得する場合，すべてのコアで特定の獲得順序で獲得しないと，図 2-59 に示すようなデッドロックが発生する可能性がある．

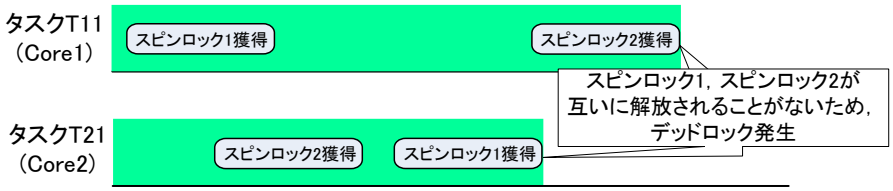


図 2-59 スピンロックによるデッドロックの例 2

このデッドロックを防止するために，本 OS は，スピンロック獲得順序を管理する機能を提供する【OS660】〔MCOS_Conf1022〕．同一の処理単位において，スピンロックをネストして獲得する際，コンフィギュレーション時に獲得順序を指定しておき，指定された獲得順序と異なる順序でスピンロックを獲得しようとした場合，エラーとする．また，獲得順序が指定されていないスピンロックを獲得している場合に，他のスピンロックを獲得しようとした場合も，エラーとする【OS661】．

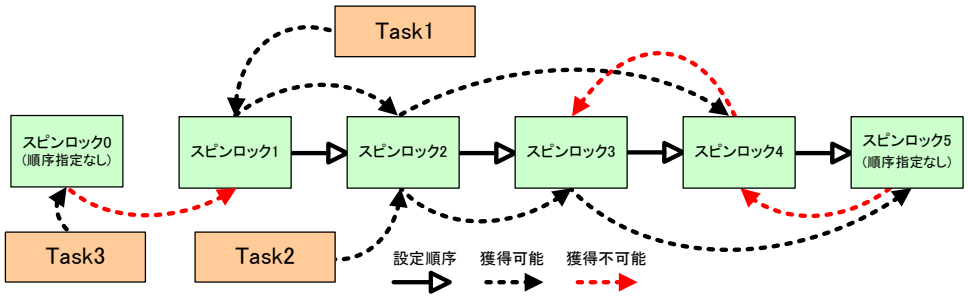


図 2-60 スピンロックの獲得可否

使用上の注意

獲得順序のチェックは，処理単位毎に行うため，図 2-61 のような操作を行った場合，デッドロックが発生する可能性がある．このデッドロックは，標準スピンロックの場合でのみ発生する．

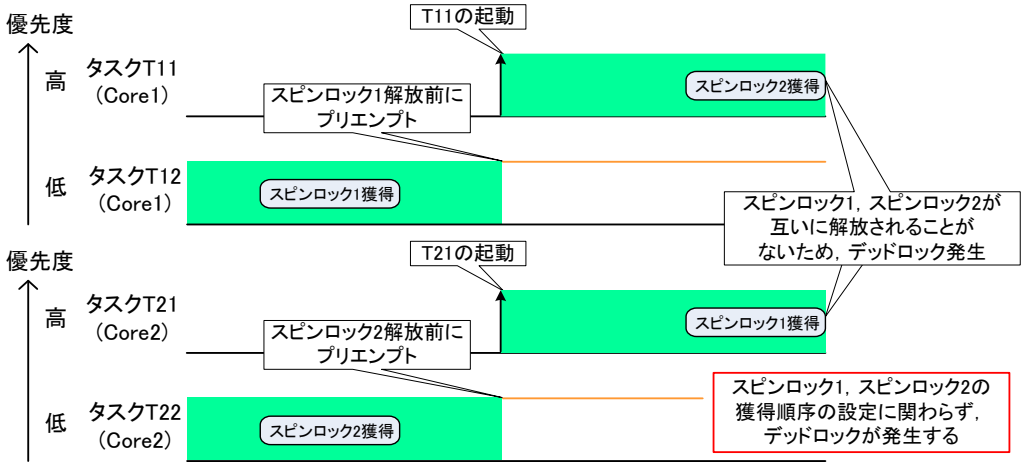


図 2-61 スピンロックによるデッドロックの例 3

解放されたスピンロックを獲得する順序

あるコアが占有中のスピンロックを獲得しようとするコアが 3 つ以上存在する場合，スピンロック獲得待ちのコアの到着順を OS が管理しないと，スピンロック獲得までの最悪実行時間が見積れなくなる．そのため，リアルタイム性を考慮し，スピンロック獲得待ちのコアを FIFO で管理して，その先頭コアが獲得する【NOS0054】．ただし，ハードウェアによって FIFO で管理できない場合は，マニュアル等に明記する．

2.26.2.2 スピンロックの解放

本 OS は，スピンロックを解放するためのシステムサービスとして ReleaseSpinlock を提供する(詳細は 3.9.67 節を参照)．ReleaseSpinlock によって，占有していないスピンロックを解放しようとした場

合，エラーとする【OS655】〔OS699〕．スピンロックは，LIFO の順序で解放を行わなくてはならない≪OS701≫．

解放されたスピンロックの獲得を待っている他のコアの処理単位が存在する場合，最初に獲得待ちとなっていたコアの処理単位がスピンロックを獲得する≪NOS0054≫．

2.26.2.3 スピンロック占有中の状態

タスクはスピンロックを占有した状態で TerminateTask，ChainTask，Schedule，WaitEvent のシステムサービスを呼出してはならない≪OS612≫≪OS624≫≪OS622≫．また，C2ISR はスピンロックを占有した状態で処理を終了してはならない【NOS0933】.

OS 割込み禁止スピンロック占有中の処理単位が割付いているコアは，OS 割込み禁止状態となる【NOS0934】．全割込み禁止スピンロック占有中の処理単位が割付いているコアは，全割込み禁止状態となる【NOS0935】．したがって，これらのスピンロックによる割込み禁止状態においても，割込み禁止状態から呼び出すことができないシステムサービスを呼び出した場合，E_OS_DISABLEDINT エラーとなる≪OS093≫≪NOS0412≫．

しかし，これらのスピンロックによる割込み禁止状態と，SuspendOSInterrupts，SuspendAllInterrupts，DisableAllInterrupts による割込み禁止状態は，互いに影響を与えないものとする【NOS0936】．例えば，OS 割込み禁止スピンロックを獲得して OS 割込み禁止状態となったコアで，ResumeOSInterrupts を発行しても，OS 割込み禁止スピンロックによる OS 割込み禁止状態は解除されない.

使用上の注意

あるコアの処理単位がスピンロックを占有している間，同じスピンロックを獲得しようとした他のコアの処理単位はスピンするため，スピンロックによる排他区間が長大化するに従い，スピンによるプロセッサ時間消費も長大化し，システム全体性能に影響を及ぼす要因になりうる．そのため，スピンロックによる排他区間は極力短くする.

2.27 コア間割込み

コア間割込み機能は，マルチコアシステムにおける他の任意のコアに対し割込みを発生させ，ユーザ定義のコア間割込みサービスルーチン(以降，ICISR と略す)を実行する機能である【NOS0060】.

本 OS は，ディスパッチやシャットダウン要求のために，コア間割込みを使用する≪NOS0917≫≪NOS0918≫．このコア間割込みに，ユーザ定義の ICISR を実行する処理を追加し，コア間割込みを発生させるシステムサービスを用意することで，自由度の高いコア間同期・通信をユーザに提供する.

ICISR は，C2ISR と同等の機能を有する OS 管理下の処理単位である【NOS0061】．定義可能なユーザ定義の ICISR の上限数は実装依存である【NOS1117】．ATK2 では，定義可能なユーザ定義の ICISR の上限数はコア毎に 32 個である【IOS229】.

ICISR の ID

ICISR は ISR ID によって識別する【NOS0062】．ISR ID は，C1ISR，C2ISR と合わせて，すべてのコアにおいて，一意とする【NOS0063】．ICISR 実行中に，GetISRID を呼び出すと，ICISR の ID が返る【NOS0937】.

ICISR の割込み優先度

ICISR の割込み優先度は実装定義であり，ユーザは指定できない【NOS0068】．ATK2 では，ICISR の割込み優先度を，ターゲット定義とする【IOS102】.

2.27.1 ICISR の起動

OS は，指定されたコアに指定されたコア間割込みを発生させるシステムサービスとして RaiseInterCoreInterrupt を提供する(詳細は 3.9.70 節を参照)．割込みが発生したコアでは，ISR ID に対応した ICISR を起動する.

指定された ICISR が割付いているコアが，割込み禁止状態などの理由で対象の ICISR を起動できない状態の場合，指定した ICISR に対する割込み要求は最大 1 回まで保留可能である【NOS0067】.

2.27.2 ICISR の生成

ICISR の本体記述

ICISR の本体記述を以下に示す【NOS0064】.

```
ICISR(<ISR ID>)
{
}
```

2.27.3 ICISR の機能

システムサービス

ICISR から呼出し可能なシステムサービスは，C2ISR と同一である【NOS0938】.

リソースによる排他制御

ICISR は，タスク－ICISR 間，C2ISR－ICISR 間，ICISR－ICISR 間でリソースによる排他制御が可能である【NOS0939】.

スタックモニタリング

ICISR がスタックオーバーフローした場合，C2ISR と同等のスタックモニタリングによって検出される【NOS0940】．1 つのコアに複数の ICISR が割付いていて，同時に複数の ICISR に対する起動要求が発生した場合，1 つの ICISR を終了する毎にスタックモニタリングを行う【IOS227】.

タイミング保護

SC2，SC4 では，ICISR に対して C2ISR と同等のタイミング保護機能が適用される【NOS0942】. ICISR に対する時間制約の指定はコンフィギュレーション時にユーザが静的に指定する【NOS0069】.

OSAP

ICISR は，C2ISR 同様，いずれかの OSAP に所属する【NOS0943】．1 つの OSAP に複数の ICISR を所属させることも可能である【NOS0944】. ICISR 実行中に, GetApplicationID を呼び出すと, ICISR が所属する OSAP ID が返る【NOS0945】．CheckISROwnership により，ICISR が所属する OSAP ID を取得できる【NOS0946】.

メモリ保護

SC3，SC4 では，ICISR に対して C2ISR と同等のメモリ保護機能が適用される【NOS0947】.

サービス保護

ICISR が不正終了する場合，C2ISR と同等のサービス保護が適用される【NOS0948】.

個別割込みの禁止・許可状態の操作

ICISR に対しても, 割込みを個別に禁止, 許可するシステムサービスである DisableInterruptSource, EnableInterruptSource を使用することができる【NOS1044】.

2.28 ミューテックス管理(参考仕様)

ミューテックスは異なるコアに割付けられたタスク間で排他制御を行うための OS オブジェクトであり，スピンロックに類似した機能である．

ミューテックスを使用して排他区間を開始することをミューテックスの獲得，排他区間を終了することをミューテックスの解放と呼ぶ. また, タスクがミューテックスを獲得して, 排他状態にあることを, ミューテックスの占有と呼ぶ.

ミューテックスはミューテックス ID を持つ．排他制御の方法として優先度上限プロトコルを採用する【NOS0083】.

スピンロックとの違い

ミューテックスはスピンロックに類似した機能であるが，以下の点でスピンロックと大きく異なる．

- ミューテックス獲得待ちをブロッキングで実現
  Waiting 状態へ移行し，他タスクに実行権を譲る．

- ミューテックスを獲得できる処理単位は拡張タスクのみ
  基本タスクと ISR から獲得することはできない．

2.28.1 優先度上限プロトコルと上限優先度

ミューテックス機能では優先度上限プロトコルによる排他制御を行う．

上限優先度

ミューテックスは属性として上限優先度を持つ．ミューテックスの上限優先度は，そのミューテックスを共有しているすべてのタスクの中で最も高い優先度である【NOS0084】．ミューテックスの上限優先度は，コンフィギュレーション時に静的に決定する【NOS0085】.

2.28.2 ミューテックスの操作

2.28.2.1 ミューテックスの獲得

本 OS は，ミューテックスを獲得するためのシステムサービスとして GetMutex を提供する(詳細は 3.9.71 節を参照).

タスクがミューテックスを獲得した時，OS はタスクの現在優先度をミューテックスの上限優先度まで引き上げる【NOS0086】．ミューテックスを獲得しタスク優先度を引き上げることで，同じコア内においてミューテックスの上限優先度以下の優先度を持つタスクが実行されることを防止し，コアに閉じた排他制御を行う．既に獲得中のミューテックスを他のコアのタスクが獲得しようとした場合，そのタスクを Waiting 状態に移行することで，コアを跨った排他制御を行う【NOS0087】.

ミューテックスを獲得するタスクの初期優先度がミューテックスの上限優先度より高い場合は，ミューテックスを獲得することはできない【NOS0088】.

タスクはリソースを獲得した状態でミューテックスを獲得することはできない.

2.28.2.2 ミューテックスの解放

本 OS は，GetMutex によって取得したミューテックスを解放するためのシステムサービスとして ReleaseMutex を提供する(詳細は 3.9.72 節を参照).

タスクがミューテックスを解放した時，OS はタスクの現在優先度をそのミューテックスを獲得した際の優先度に戻す【NOS0089】．ミューテックスを解放した時点で，OS はミューテックスの獲得を待っている待ち状態のタスクを実行可能状態に移行させる【NOS0090】．複数のミューテックス待ちタスクが存在した場合，タスクの優先度が高いものから先に待ち解除させる【NOS0091】.

また，ミューテックスを解放した時点で，ミューテックスを解放したタスクが動作しているコアにおいて OS は再スケジューリングを行う【NOS0092】．この時，ミューテックスを解放したタスクがシステムの中で最高優先度のタスクでない場合，ミューテックスを解放したタスクはプリエンプトされ，OS は最高優先順位を持つタスクへのタスクディスパッチを行う．

タスクは獲得していないミューテックスの解放操作をすることはできない≪NOS0106≫.

2.28.2.3 ミューテックス占有中の状態

タスクはミューテックスを占有した状態で TerminateTask，ChainTask，Schedule，WaitEvent のシステムサービスを呼出すことはできない【NOS0096】.

ミューテックスの獲得と解放はネストさせることができるが，LIFO の順序で獲得と解放を行わなくてはならない【NOS0093】．また，ミューテックスの獲得と解放は，同一の関数内で行うことを推奨する．なお，タスクは同一のミューテックスをネストして獲得することはできない.

2.28.3 ミューテックスの生成

ジェネレータは，ミューテックスを共有しているすべてのタスク優先度を解釈し，ミューテックスの上限優先度を決定する【NOS0290】．ユーザはコンフィギュレーション時に，ミューテックスを用いて排他制御を行うタスクに対して，共有するミューテックスの ID を指定する必要がある.

2.29 コンフィギュレーション方法

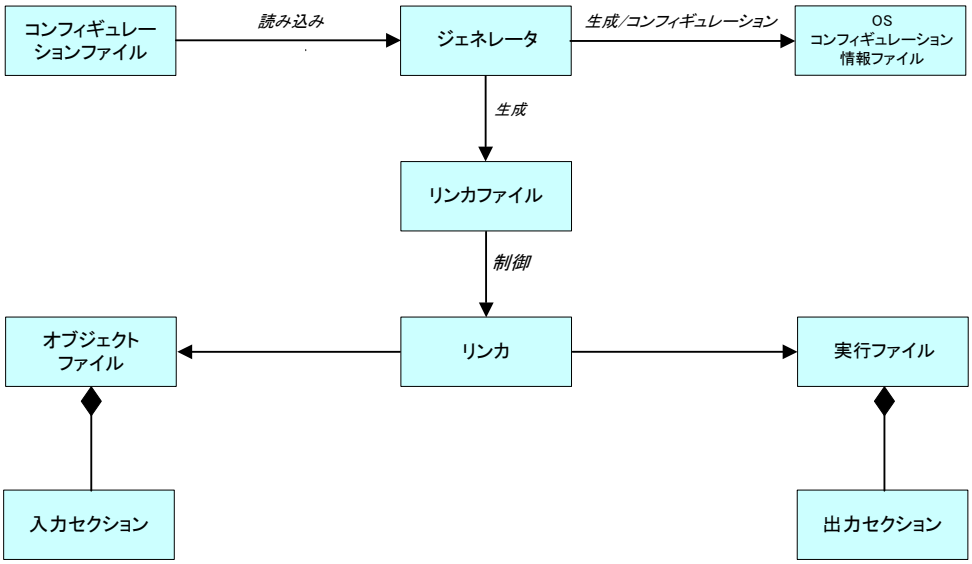


図 2-62 コンフィギュレーション手順

2.29.1 コンフィギュレーションファイルの記述

本 OS ではコンフィギュレーションファイルと呼ぶファイルに OS オブジェクトの定義を記述する．

ユーザは OS が提供する定義方法を用いて，OS オブジェクトを静的に定義する【COS1201】．本 OS では動的に OS オブジェクトを生成することはできない【COS1225】．

2.29.2 コンフィギュレーション処理

コンフィギュレーション情報ファイルの生成には外部ツールが必要である(以降，ジェネレータと呼ぶ)【COS1226】．ジェネレータはコンフィギュレーション情報に追加や修正を加えることができる．

ジェネレータは以下の処理を行う．

- コンフィギュレーションファイルを読み込む【OS172】.
- コンフィギュレーションファイルの整合性をチェックする【OS173】.
- エラーを検出した場合は，コンフィギュレーション情報ファイルを生成しない【OS179】.

ジェネレータは，指定されたスケーラビリティクラスと一致しないコンフィギュレーション情報がある場合(例：SC1 を選択して，タスクや C2ISR の実行時間バジェットを指定する)，警告を出力する【OS320】.

コンフィギュレーション情報に浮動小数点型のパラメータがある場合，以下のパラメータを除いては，OS で扱える値へ丸める【OSa010】.

- タイミング保護に使用する実行時間は切り捨てる【OSa011】.
- タイムフレームは切り捨てる【OSa012】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，サービス保護を使用する場合に拡張エラーを選択しないとエラーとすると規定している【OS050】．サービス保護には，OSAP が起こすエラーに対する保護が含まれるが，SC1，SC2 のマルチコア対応 OS では，拡張エラーを使用しないことが可能である≪OS763≫．したがって，本規定を削除した．

2.29.3 コンフィギュレーション情報の出力

ジェネレータは，ベクタテーブルを含む再配置可能なコンフィギュレーション情報を出力する【OS336】．また，OS 内部で使用するタイマの情報をコンソールやリストファイルの形式で出力する【OS370】.

2.29.4 OS オブジェクトのアクセス方法

オブジェクトを識別するためのデータ型の定義は実装依存である【COS1203】．ユーザは，OS オブジェクトへの参照の宣言無しに，コンフィギュレーション時に静的に指定した名称を用いて，OS オブジェクトにアクセスできる【OSa004】.

OSEK 仕様との違い

OSEK 仕様では，ジェネレータによって生成された OS オブジェクトにアクセスするためにはアプリケーションプログラムで OS オブジェクトへの参照を宣言すると規定されている【COS1232】. しかし，AUTOSAR 仕様では，移植性のために，宣言の有無に関係なくアクセスできると説明されているため，本仕様では宣言に対する仕様を削除した≪OSa004≫．ただし，OSEK 仕様に準拠して開発されたアプリケーションとの互換性のため，OS オブジェクトへの参照の宣言があってもエラーとしない(詳細は 3.7

節を参照).

2.30 ファイル構成

2.30.1 OS ヘッダファイル

本 OS を使用するために必要な定義や宣言は，OS ヘッダファイル(Os.h)および OS ヘッダファイルからインクルードされるファイルに含まれる．本 OS を使用する場合には，OS ヘッダファイルをインクルードする【OS546】〔COS1205〕.

本 OS を使用するために必要な定義の中で，コンフィギュレーションによって生成される定義は，OS ヘッダファイルとは別ファイル(Os_Cfg.h)に定義し，OS ヘッダファイルからインクルードする【OSa008】．また，本 OS と他の関連モジュールで共通に用いる定義も同様に，OS ヘッダファイルとは別ファイル(Rte_Os_Type.h，Std_Types.h，MemMap.h)に定義し，OS ヘッダファイルからインクルードする【OSa009】．Rte_Os_Type.h は，OS ヘッダファイルからインクルードされ，システム全体で共通的に使用するデータ型が定義された Rte_Type.h をインクルードする【OS765】.

ATK2 では，プリコンパイルが必要な情報のみを Os_Cfg.h に定義し，アプリケーションとのリンク時に必要な情報を Os_Lcfg.h へ定義する【IOS046】.

2.30.2 ファイルの互換性チェック

本 OS では，プレコンパイル時のチェックにて，ファイルの互換性のチェックを行う．OS が他のモジュールのバージョンに依存する場合は，インクルードするヘッダファイル(Os.h)に以下のバージョン情報を記述し，他のモジュールのバージョンをチェックし，未サポートのバージョンである場合，本 OS はエラーを返す【OS552】.

- OS_AR_RELEASE_MAJOR_VERSION
- OS_AR_RELEASE_MINOR_VERSION

他のモジュールのヘッダファイルにも，以下の命名規則でバージョン情報が記述される.

- <MODULENAME>_AR_RELEASE_MAJOR_VERSION
- <MODULENAME>_AR_RELEASE_MINOR_VERSION

2.30.3 デバッグサポート

AUTOSAR 仕様との違い

AUTOSAR 仕様では，アプリケーション実装時のデバッグをサポートするため，デバッグに使用する情報を OS ヘッダファイルより公開すると規定している．

- グローバル変数での定義【OS551】
- Os.h による変数アクセス【OS550】
- sizeof 演算子によるサイズ取得【OS549】

しかし，ヘッダファイルを 1 つにまとめることでデバッグが容易になるという根拠が不明確であるため，本仕様では規定しない．なお，AUTOSAR 仕様 V5.1.0 (R4.1 Rev 1)では，これらの規定は削除されている．

2.31 ポートインターフェイス

ポートインターフェイスの仕様については，現在検討中である．

- OsService ポートインターフェイス定義【OS560】
- Os ポート定義【OS561】

3. API 仕様

3.1　API の実装ルール

API のインターフェース記述と実装

システムサービスのインターフェースは ISO/ANSI-C に基づいて記載する．通常の実装では ISO/ANSI-C であるが，必要に応じて C 言語マクロなどで実装してもよい【COS1228】.

データ型の実装

データ型の実装方法は特に規定しない【COS1234】.

3.2　各処理単位が使用できるシステムサービス

システムサービスは各処理単位から呼び出すことができる【COS1205】．しかし，処理単位によって使用することのできるシステムサービスは制限される．さらに，コンフォーマンスクラスによっても制限される【COS1229】．詳細は各システムサービスの仕様を参照.

各処理単位から呼び出すことのできるシステムサービスを表 3-1 に示す.

表 3-1 各処理単位から呼出し可能なシステムサービス

System Service	Task	C1 ISR	C2 ISR	ErrorHook	PreTaskHook	PostTaskHook	StartupHook	ShutdownHook	AlarmCallback	ProtectionHook
ActivateTask【OSa061】	○		○							
TerminateTask【OSa062】	○									
ChainTask【OSa063】	○									
Schedule【OSa064】	○									
GetTaskID【OSa065】	○		○	○	○	○				○
GetTaskState【OSa066】	○		○	○	○	○				
DisableAllInterrupts【OSa067】	○	○	○	○	○	○	○	○	○	○
EnableAllInterrupts【OSa068】	○	○	○	○	○	○	○	○	○	○
SuspendAllInterrupts【OSa069】	○	○	○	○	○	○	○	○	○	○
ResumeAllInterrupts【OSa070】	○	○	○	○	○	○	○	○	○	○
SuspendOSInterrupts【OSa071】	○	○	○	○	○	○	○	○	○	○
ResumeOSInterrupts【OSa072】	○	○	○	○	○	○	○	○	○	○

System Service	Task	C1 ISR	C2 ISR	ErrorHook	PreTaskHook	PostTaskHook	StartupHook	ShutdownHook	AlarmCallback	ProtectionHook
GetResource【OSa073】	○		※1							
ReleaseResource【OSa074】	○		※1							
SetEvent【OSa075】	○		○							
ClearEvent【OSa076】	○									
GetEvent【OSa077】	○		○	○	○	○				
WaitEvent【OSa078】	○									
GetAlarmBase【OSa079】	○		○	○	○	○				
GetAlarm【OSa080】	○		○	○	○	○				
SetRelAlarm【OSa081】	○		○							
SetAbsAlarm【OSa082】	○		○							
CancelAlarm【OSa083】	○		○							
GetActiveApplicationMode【OSa084】	○		○	○	○	○	○	○		
StartOS【OSa085】										
ShutdownOS【OSa086】	○		○	○			○			
GetApplicationID【OSa087】	○		○	○	○	○	○	○		○
GetISRID【OSa088】	○		○	○						○
CallTrustedFunction【OSa089】	○		○							
CheckISRMemoryAccess【OSa090】	○		○	○						○
CheckTaskMemoryAccess【OSa091】	○		○	○						○
CheckTaskAccess【OSa092】	○		○	○						○
CheckISRAccess【OSa093】	○		○	○						○
CheckAlarmAccess【OSa094】	○		○	○						○
CheckResourceAccess【OSa095】	○		○	○						○
CheckCounterAccess【OSa096】	○		○	○						○
CheckScheduleTableAccess【OSa097】	○		○	○						○
CheckTaskOwnership【OSa098】	○		○	○						○
CheckISROwnership【OSa099】	○		○	○						○
CheckAlarmOwnership【OSa100】	○		○	○						○
CheckCounterOwnership【OSa101】	○		○	○						○
CheckScheduleTableOwnership【OSa102】	○		○	○						○

System Service	Task	C1 ISR	C2 ISR	ErrorHook	PreTaskHook	PostTaskHook	StartupHook	ShutdownHook	AlarmCallback	ProtectionHook
StartScheduleTableRel【OSa103】	○		○							
StartScheduleTableAbs【OSa104】	○		○							
StartScheduleTableSynchron【OSa105】	○		○							
StopScheduleTable【OSa106】	○		○							
NextScheduleTable【OSa107】	○		○							
SyncScheduleTable【OSa108】	○		○							
GetScheduleTableStatus【OSa109】	○		○							
SetScheduleTableAsync【OSa110】	○		○							
IncrementCounter【OSa111】	○		○							
GetCounterValue【OSa112】	○		○							
GetElapsedValue【OSa113】	○		○							
TerminateApplication【OSa114】	○		○	※2						
AllowAccess【OSa115】	○		○							
GetApplicationState【OSa116】	○		○	○	○	○	○	○		○
DisableInterruptSource【NOS0500】	○		○	○	○	○	○	○		○
EnableInterruptSource【NOS0501】	○		○	○	○	○	○	○		○
GetNumberOfActivatedCores【NOS0949】	○	○	○	○	○	○	○	○	○	○
GetCoreID【OSa129】	○	○	○	○	○	○	○	○	○	○
StartCore【OSa130】										
StartNonAutosarCore【OSa131】										
GetSpinlock【OS650】 【OS651】 【NOS1121】	○		○							○
ReleaseSpinlock【OS656】 【OS657】 【NOS1122】	○		○							○
TryToGetSpinlock【OS653】 【OS654】 【NOS1123】	○		○							○
ShutdownAllCores【OSa132】	○		○	○			○			

System Service	Task	C1 ISR	C2 ISR	ErrorHook	PreTaskHook	PostTaskHook	StartupHook	ShutdownHook	AlarmCallback	ProtectionHook
RaiseInterCoreInterrupt【NOS0502】	○		○	○	○	○				○
GetMutex【NOS0504】	○									
ReleaseMutex【NOS0505】	○									
GetFaultyContext【IOS192】										○

○：呼出し可能

※1：オプションにより呼出し可能≪IOS168≫

※2：OSAP 固有エラーフックが，自身の所属する OSAP を終了する場合にのみ呼出し可能【OSa180】

AUTOSAR 仕様との違い

AUTOSAR 仕様では，GetNumberOfActivatedCores を呼出し可能な処理単位は，タスクと C2ISR と規定されている【OSa133】．しかし，GetNumberOfActivatedCores をタスクと C2ISR 以外の処理単位から呼び出すことによる不都合が想定できないため，本仕様では呼び出すことができると規定した≪NOS0949≫.

AUTOSAR 仕様では，GetSpinlock，ReleaseSpinlock，TryToGetSpinlock はタスクと C2ISR からのみ呼出し可能と規定されている≪OS650≫≪OS651≫≪OS656≫≪OS657≫≪OS653≫≪OS654≫. しかし，RTE によって生成されるコードでは，プロテクションフックからもスピンロックを使用する可能性があるので，本仕様では，プロテクションフックからも GetSpinlock，ReleaseSpinlock，TryToGetSpinlock を呼出し可能と規定した≪NOS1121≫≪NOS1122≫≪NOS1123≫.

3.3　API 仕様記載凡例

API 仕様記載の凡例を示す.

システムサービス名

C 言語 I/F		＜C 言語 I/F を記載する＞
パラメータ[in]		＜システムサービスパラメータで入力となるものを記載する＞
パラメータ[in/out]		＜システムサービスパラメータで入力と出力となるものを記載する＞
パラメータ[out]		＜システムサービスパラメータで出力となるものを記載する＞
返り値	標準エラー	＜システムサービスの返り値で標準エラーとなるものと，該当エラーが発生する条件を記載する．(W)と付いている場合は警告として扱ってもよい＞
	拡張エラー	＜システムサービスの返り値で拡張エラーとなるものと，該当エラーが発生する条件を記載する＞
エラーフックに渡されるエラーコード	標準エラー	＜返り値の型が StatusType でないシステムサービスにおいて，標準エラーとしてエラーフックが呼び出されるものと，該当するエラーが発生する条件を記載する＞
	拡張エラー	＜返り値の型が StatusType でないシステムサービスにおいて，拡張エラーとしてエラーフックが呼び出されるものと，該当するエラーが発生する条件を記載する＞
コンフォーマンスクラス		＜使用できるコンフォーマンスクラスを記載する＞
スケーラビリティクラス		＜使用できるスケーラビリティクラスを記載する＞ ＜マルチコア対応 OS の場合にサポートされるスケーラビリティクラスを「マルチコア：」に続けて記載する＞
機能		＜システムサービスの機能を記載する＞

データ型名

データ型名	＜データ型の ID 名を記載する＞
概要	＜データ型が示す内容を記載する＞

マクロ名

マクロ名	＜マクロの ID 名を記載する＞
概要	＜マクロの定義内容を記載する＞

定数名

定数名	＜定数の ID 名を記載する＞
概要	＜定数が示す内容を記載する＞

宣言記述名

宣言記述名	＜宣言記述の名称と，引数を記載する＞
概要	＜宣言記述の内容を記載する＞
コンフォーマンスクラス	＜使用できるコンフォーマンスクラスを記載する＞
スケーラビリティクラス	＜使用できるスケーラビリティクラスを記載する＞

コンテナ名＜コンテナ名称の末尾のみを記載する＞

コンテナ名	＜コンテナ名称をフルパスで記載する＞
概要	＜コンテナの概要を記載する＞
多重度	＜コンテナの多重度を記載する＞
パラメータ	＜コンテナが含むパラメータ名を列挙する＞
	:
サブコンテナ	＜コンテナが含むサブコンテナを列挙する＞
	:

コンテナパラメータ名＜コンテナパラメータの末尾のみを記載する＞

パラメータ名	＜コンテナパラメータ名称をフルパスで記載する＞
概要	＜コンテナパラメータの概要を記載する＞
型	＜コンテナパラメータのデータ型を記載する＞
値の範囲	＜コンテナパラメータの値範囲を記載する＞
多重度	＜コンテナパラメータの多重度を記載する＞
制限事項	＜制限事項がある場合，コンテナパラメータへの制約を記載する＞

3.4　データ型

3.4.1　StatusType

データ型名	StatusType【COS3101】
概要	システムサービス要求に対する結果

StatusType に対する命名規則

システムサービスのエラーコードは「E_」から始まる名称をつける【COS3102】.

OS による予約語

「E_OS_」で始まる ID は OS の予約語である.

システムサービスエラーコード定義

システムサービスが正常に終了した場合の StatusType の値は E_OK(=0)である【COS3104】〔COS1208〕.

それ以外の場合はエラーコードで，以下の値を定義する.

- E_OS_ACCESS = 1【COS3105】〔COS3102〕〔COS1208〕
- E_OS_CALLEVEL = 2【COS3106】〔COS3102〕〔COS1208〕
- E_OS_ID = 3【COS3107】〔COS3102〕〔COS1208〕
- E_OS_LIMIT = 4 【COS3108】〔COS3102〕〔COS1208〕
- E_OS_NOFUNC = 5【COS3109】〔COS3102〕〔COS1208〕
- E_OS_RESOURCE = 6【COS3110】〔COS3102〕〔COS1208〕
- E_OS_STATE = 7【COS3111】〔COS3102〕〔COS1208〕
- E_OS_VALUE = 8【COS3112】〔COS3102〕〔COS1208〕
- E_OS_SERVICEID (値は実装定義) = 9【IOS103】〔COS3102〕〔COS1208〕
- E_OS_ILLEGAL_ADDRESS (値は実装定義) = 10【IOS104】〔COS3102〕〔COS1208〕
- E_OS_MISSINGEND (値は実装定義) = 11【IOS105】〔COS3102〕〔COS1208〕
- E_OS_DISABLEDINT (値は実装定義) = 12【IOS106】〔COS3102〕〔COS1208〕
- E_OS_STACKFAULT (値は実装定義) = 13【IOS107】〔COS3102〕〔COS1208〕
- E_OS_PROTECTION_MEMORY (値は実装定義) = 14【IOS108】〔COS3102〕〔COS1208〕
- E_OS_PROTECTION_TIME_TASK (値は実装定義) = 15【IOS109】〔COS3102〕〔COS1208〕
- E_OS_PROTECTION_TIME_ISR (値は実装定義) = 16【IOS110】〔COS3102〕〔COS1208〕
- E_OS_PROTECTION_ARRIVAL_TASK（値は実装定義） = 17【IOS111】〔COS3102〕〔COS1208〕
- E_OS_PROTECTION_ARRIVAL_ISR (値は実装定義) = 18【IOS112】〔COS3102〕〔COS1208〕
- E_OS_PROTECTION_LOCKED_RESOURCE（値は実装定義） = 19【IOS113】〔COS3102〕〔COS1208〕

- E_OS_PROTECTION_LOCKED_OSINT（値は実装定義）　= 20【IOS114】〔COS3102〕〔COS1208〕
- E_OS_PROTECTION_LOCKED_ALLINT（値は実装定義）　= 21【IOS115】〔COS3102〕〔COS1208〕
- E_OS_PROTECTION_EXCEPTION（値は実装定義）　= 22【IOS116】〔COS3102〕〔COS1208〕
- E_OS_PROTECTION_FATAL（値は実装定義）　= 23【IOS117】〔COS3102〕〔COS1208〕
- E_OS_MODE（値は実装定義）　= 24【IOS120】〔COS3102〕〔COS1208〕
- E_OS_SHUTDOWN_FATAL（値は実装定義）　= 25【IOS121】〔COS3102〕〔COS1208〕
- E_OS_PARAM_POINTER（値は実装定義）　= 26【IOS122】〔COS3102〕〔COS1208〕
- E_OS_SYS_ASSERT_FATAL（値は実装定義）　= 27【IOS123】〔COS3102〕〔COS1208〕
- E_OS_STACKINSUFFICIENT（値は実装定義）　= 28【IOS124】〔COS3102〕〔COS1208〕
- E_OS_CORE（値は実装定義）　= 29【IOS204】〔COS3102〕〔COS1208〕
- E_OS_SPINLOCK（値は実装定義）　= 30【IOS205】〔COS3102〕〔COS1208〕
- E_OS_INTERFERENCE_DEADLOCK（値は実装定義）　= 31【IOS119】〔COS3102〕〔COS1208〕
- E_OS_NESTING_DEADLOCK（値は実装定義）　= 32【IOS118】〔COS3102〕〔COS1208〕
- E_OS_SHUTDOWN_OTHER_CORE（値は実装定義）　= 33【IOS206】〔COS3102〕〔COS1208〕

OS 内エラーコード定義

本 OS では OS 内で発生したエラーに対して「E_OS_SYS_」から始まるエラーコードを実装定義で定義してもよい【COS3113】. ただし，これらのエラーコードは移植性がないため，注意が必要である.

OS 内エラーコードは，システムサービスと同様に，アプリケーションから使用することができる【COS3114】.

ATK2 では，E_OS_SYS_ASSERT_FATAL，E_OS_STACKINSUFFICIENT を規定した≪IOS123≫≪IOS124≫.

3.4.2　AppModeType

データ型名	AppModeType【COS3701】
概要	アプリケーションモード型

3.4.3　OSServiceIdType

データ型名	OSServiceIdType【COS3801】
概要	システムサービス ID 型

3.4.4 TaskType

データ型名	TaskType【COS3201】
概要	タスク ID

3.4.5 TaskRefType

データ型名	TaskRefType【COS3202】
概要	タスク ID へのポインタ型

3.4.6 TaskStateType

データ型名	TaskStateType【COS3203】
概要	タスク状態

3.4.7 TaskStateRefType

データ型名	TaskStateRefType【COS3204】
概要	タスク状態へのポインタ型

3.4.8 ISRType

データ型名	ISRType【OSa014】
概要	ISRID

3.4.9 EventMaskType

データ型名	EventMaskType【COS3501】
概要	イベントマスク型

3.4.10 EventMaskRefType

データ型名	EventMaskRefType【COS3502】
概要	イベントマスクへのポインタ型

3.4.11 ResourceType

データ型名	ResourceType【COS3401】
概要	リソース ID

3.4.12 CounterType

データ型名	CounterType【OSa018】
概要	カウンタ ID

3.4.13 TickType

データ型名	TickType【COS3601】
概要	ティック

3.4.14 TickRefType

データ型名	TickRefType【COS3602】
概要	ティックへのポインタ型

3.4.15 AlarmType

データ型名	AlarmType【COS3609】
概要	アラーム ID

3.4.16 AlarmBaseType

データ型名	AlarmBaseType【COS3603】		
概要	アラーム情報		
メンバ変数	データ型	メンバ名	メンバ変数の概要
	TickType	maxallowedvalue	システムサービスで指定できるティックの最大値
	TickType	ticksperbase	カウンタ固有の値(OS は不使用)
	TickType	mincycle	アラームのセットに指定できる最小のサイクル数

3.4.17 AlarmBaseRefType

データ型名	AlarmBaseRefType【COS3608】
概要	アラーム情報へのポインタ型

3.4.18 ScheduleTableType

データ型名	ScheduleTableType【OSa015】
概要	スケジュールテーブル ID

3.4.19 ScheduleTableStatusType

データ型名	ScheduleTableStatusType【OSa016】
概要	スケジュールテーブル状態型

3.4.20 ScheduleTableStatusRefType

データ型名	ScheduleTableStatusRefType【OSa017】
概要	スケジュールテーブル状態へのポインタ型

3.4.21 PhysicalTimeType

データ型名	PhysicalTimeType【OSa019】
概要	ティックから時間に換算するマクロで使用される型

3.4.22 ApplicationType

データ型名	ApplicationType【OSa020】
概要	OSAPID

3.4.23 ApplicationStateType

データ型名	ApplicationStateType【OSa021】
概要	OSAP の状態を示すデータ型

3.4.24 ApplicationStateRefType

データ型名	ApplicationStateRefType【OSa022】
概要	OSAP の状態を示すデータへのポインタ型

3.4.25 TrustedFunctionIndexType

データ型名	TrustedFunctionIndexType【OSa023】
概要	信頼関数 ID

3.4.26 TrustedFunctionParameterRefType

データ型名	TrustedFunctionParameterRefType【OSa024】
概要	信頼関数パラメータへのポインタ型. TrustedFunctionParameterRefType は任意のデータ型へのポインタをキャストせずに渡せるように，void*で定義する.

3.4.27 AccessType

データ型名	AccessType【OSa025】
概要	メモリ領域アクセス権

3.4.28 ObjectAccessType

データ型名	ObjectAccessType【OSa026】
概要	OS オブジェクトアクセス権

3.4.29 ObjectTypeType

データ型名	ObjectTypeType【OSa027】
概要	OS オブジェクト種別

3.4.30 MemoryStartAddressType

データ型名	MemoryStartAddressType【OSa028】
概要	メモリ領域先頭へのポインタ型

3.4.31 MemorySizeType

データ型名	MemorySizeType【OSa029】
概要	メモリ領域サイズ

3.4.32 RestartType

データ型名	RestartType【OSa030】
概要	OSAP 強制終了時のリスタートタスク起動有無

3.4.33 ProtectionReturnType

データ型名	ProtectionReturnType【OSa031】
概要	保護違反に対する OS の処理種別

3.4.34 Std_ReturnType

データ型名	Std_ReturnType【OSa172】
概要	IOC 用システムサービス要求に対する結果

IOC 用システムサービスエラーコード定義

IOC 用システムサービスが正常に終了した場合の Std_ReturnType の値は IOC_E_OK (=0)である【OSa173】．それ以外の場合はエラーコードで，以下の値を定義する．

- IOC_E_NOK = 1【OSa174】
- IOC_E_LIMIT = 130【OSa175】
- IOC_E_LOST_DATA = 64【OSa176】
- IOC_E_NO_DATA = 131 【OSa177】

3.4.35 IocType

データ型名	IocType【NOS1089】
概要	IOC ID

3.4.36 SenderIdType

データ型名	SenderIdType【NOS1090】
概要	センダ ID

3.4.37 CoreIdType

データ型名	CoreIdType【OSa134】
概要	コア ID

3.4.38 SpinlockIdType

データ型名	SpinlockIdType【OSa135】
概要	スピンロック ID

3.4.39 TryToGetSpinlockType

データ型名	TryToGetSpinlockType【OSa136】
概要	TryToGetSpinlock 実行結果情報

3.4.40 MutexType

データ型名	MutexType【NOS0098】
概要	ミューテックス ID

3.4.41 StackType

データ型名	StackType【NOS0841】
概要	スタック領域を用意するためのデータ型

3.4.42 TimeType

データ型名	TimeType【NOS0859】
概要	ハードウェアカウンタにおける 1 ティック当たりの実時間をナノ秒単位で管理するためのデータ型

3.4.43 FaultyContextType

データ型名	FaultyContextType【IOS182】
概要	保護違反を起こした処理単位情報を管理するためのデータ型

3.5　マクロ

3.5.1　メモリアクセス

OSMEMORY_IS_READABLE

マクロ名	OSMEMORY_IS_READABLE(AccessType)【OSa032】
概要	指定されたメモリが読出し可能である場合，0 以外の値を返す

OSMEMORY_IS_WRITEABLE

マクロ名	OSMEMORY_IS_WRITEABLE(AccessType)　【OSa033】
概要	指定されたメモリが書込み可能である場合，0 以外の値を返す

OSMEMORY_IS_EXECUTABLE

マクロ名	OSMEMORY_IS_EXECUTABLE(AccessType)　【OSa034】
概要	指定されたメモリが実行可能である場合，0 以外の値を返す

OSMEMORY_IS_STACKSPACE

マクロ名	OSMEMORY_IS_STACKSPACE(AccessType)　【OSa035】
概要	指定されたメモリがスタック領域である場合，0 以外の値を返す

3.5.2　システムサービス呼出し

SVC_CALL

マクロ名	SVC_CALL(SystemServiceName)(Parameter)【NOS0292】
概要	Parameter で指定された引数で，SystemServiceName で指定されたシステムサービスを，明示的に関数呼出しにて発行する.

SVC_TRAP

マクロ名	SVC_TRAP(SystemServiceName)(Parameter)【NOS0293】
概要	Parameter で指定された引数で，SystemServiceName で指定されたシステムサービスを，明示的にソフトウェア割込みにて発行する.

AUTOSAR 仕様との違い

OS のシステムサービスを，関数呼出しで行うか，ソフトウェア割込みで行うかを明示的に区別するため，本仕様では，SVC_CALL と SVC_TRAP の 2 つのインターフェースを定義する≪NOS0292≫≪NOS0293≫．なお，信頼 OSAP から，関数呼出しでシステムサービスを発行できるかは実装定義である．ATK2 では，信頼 OSAP から，関数呼出しでシステムサービスを発行可能とする【IOS132】.

3.5.3　システムサービス ID 取得

OSErrorGetServiceId

マクロ名	OSErrorGetServiceId()【COS3834】〔COS1130〕
概要	システムサービスエラー発生時のシステムサービス ID の取得.

3.5.4　システムサービスパラメータ取得

OSError_StartOS_Mode

マクロ名	OSError_StartOS_Mode()≪COS3836≫
概要	システムサービスエラー発生時の StartOS 実行第 1 引数の取得.

OSError_ShutdownOS_Error

マクロ名	OSError_ShutdownOS_Error()≪COS3836≫
概要	システムサービスエラー発生時の ShutdownOS 実行第 1 引数の取得.

OSError_ActivateTask_TaskID

マクロ名	OSError_ActivateTask_TaskID()≪COS3836≫
概要	システムサービスエラー発生時の ActivateTask 実行第 1 引数の取得.

OSError_ChainTask_TaskID

マクロ名	OSError_ChainTask_TaskID()≪COS3836≫
概要	システムサービスエラー発生時の ChainTask 実行第 1 引数の取得.

OSError_GetTaskID_TaskID

マクロ名	OSError_GetTaskID_TaskID()≪COS3836≫
概要	システムサービスエラー発生時の GetTaskID 実行第 1 引数の取得.

OSError_GetTaskState_TaskID

マクロ名	OSError_GetTaskState_TaskID()≪COS3836≫
概要	システムサービスエラー発生時の GetTaskState 実行第 1 引数の取得.

OSError_GetTaskState_State

マクロ名	OSError_GetTaskState_State()≪COS3836≫
概要	システムサービスエラー発生時の GetTaskState 実行第 2 引数の取得.

OSError_EnableInterruptSource_EnableISR

マクロ名	OSError_EnableInterruptSource_EnableISR()≪COS3836≫
概要	システムサービスエラー発生時の EnableInterruptSource 実行第 1 引数の取得.

OSError_DisableInterruptSource_DisableISR

マクロ名	OSError_DisableInterruptSource_DisableISR()≪COS3836≫
概要	システムサービスエラー発生時の DisableInterruptSource 実行第 1 引数の取得.

OSError_SetEvent_TaskID

マクロ名	OSError_SetEvent_TaskID()≪COS3836≫
概要	システムサービスエラー発生時の SetEvent 実行第 1 引数の取得.

OSError_SetEvent_Mask

マクロ名	OSError_SetEvent_Mask()≪COS3836≫
概要	システムサービスエラー発生時の SetEvent 実行第 2 引数の取得.

OSError_ClearEvent_Mask

マクロ名	OSError_ClearEvent_Mask()≪COS3836≫
概要	システムサービスエラー発生時の ClearEvent 実行第 1 引数の取得.

OSError_GetEvent_TaskID

マクロ名	OSError_GetEvent_TaskID()≪COS3836≫
概要	システムサービスエラー発生時の GetEvent 実行第 1 引数の取得.

OSError_GetEvent_Event

構文	OSError_GetEvent_Event()≪COS3836≫
記述	システムサービスエラー発生時の GetEvent 実行第 2 引数の取得.

OSError_WaitEvent_Mask

マクロ名	OSError_WaitEvent_Mask()≪COS3836≫
概要	システムサービスエラー発生時の WaitEvent 実行第 1 引数の取得.

OSError_GetResource_ResID

マクロ名	OSError_GetResource_ResID()≪COS3836≫
概要	システムサービスエラー発生時の GetResource 実行第 1 引数の取得.

OSError_ReleaseResource_ResID

マクロ名	OSError_ReleaseResource_ResID()≪COS3836≫
概要	システムサービスエラー発生時の ReleaseResource 実行第 1 引数の取得.

OSError_IncrementCounter_CounterID

マクロ名	OSError_IncrementCounter_CounterID()≪COS3836≫
概要	システムサービスエラー発生時の IncrementCounter 実行第 1 引数の取得.

OSError_GetCounterValue_CounterID

マクロ名	OSError_GetCounterValue_CounterID()≪COS3836≫
概要	システムサービスエラー発生時の GetCounterValue 実行第 1 引数の取得.

OSError_GetCounterValue_Value

マクロ名	OSError_GetCounterValue_Value()≪COS3836≫
概要	システムサービスエラー発生時の GetCounterValue 実行第 2 引数の取得.

OSError_GetElapsedValue_CounterID

マクロ名	OSError_GetElapsedValue_CounterID()≪COS3836≫
概要	システムサービスエラー発生時の GetElapsedValue 実行第 1 引数の取得.

OSError_GetElapsedValue_Value

マクロ名	OSError_GetElapsedValue_Value()≪COS3836≫
概要	システムサービスエラー発生時の GetElapsedValue 実行第 2 引数の取得.

OSError_GetElapsedValue_ElapsedValue

マクロ名	OSError_GetElapsedValue_ElapsedValue()≪COS3836≫
概要	システムサービスエラー発生時の GetElapsedValue 実行第 3 引数の取得.

OSError_GetAlarmBase_AlarmID

マクロ名	OSError_GetAlarmBase_AlarmID()≪COS3836≫
概要	システムサービスエラー発生時の GetAlarmBase 実行第 1 引数の取得.

OSError_GetAlarmBase_Info

マクロ名	OSError_GetAlarmBase_Info()≪COS3836≫
概要	システムサービスエラー発生時の GetAlarmBase 実行第 2 引数の取得.

OSError_GetAlarm_AlarmID

マクロ名	OSError_GetAlarm_AlarmID()≪COS3836≫
概要	システムサービスエラー発生時の GetAlarm 実行第 1 引数の取得.

OSError_GetAlarm_Tick

マクロ名	OSError_GetAlarm_Tick()≪COS3836≫
概要	システムサービスエラー発生時の GetAlarm 実行第 2 引数の取得.

OSError_SetRelAlarm_AlarmID

マクロ名	OSError_SetRelAlarm_AlarmID()≪COS3836≫
概要	システムサービスエラー発生時の SetRelAlarm 実行第 1 引数の取得

OSError_SetRelAlarm_increment

マクロ名	OSError_SetRelAlarm_increment()≪COS3836≫
概要	システムサービスエラー発生時の SetRelAlarm 実行第 2 引数の取得

OSError_SetRelAlarm_cycle

マクロ名	OSError_SetRelAlarm_cycle()≪COS3836≫
概要	システムサービスエラー発生時の SetRelAlarm 実行第 3 引数の取得.

OSError_SetAbsAlarm_AlarmID

マクロ名	OSError_SetAbsAlarm_AlarmID()≪COS3836≫
概要	システムサービスエラー発生時の SetAbsAlarm 実行第 1 引数の取得.

OSError_SetAbsAlarm_start

マクロ名	OSError_SetAbsAlarm_start()≪COS3836≫
概要	システムサービスエラー発生時の SetAbsAlarm 実行第 2 引数の取得.

OSError_SetAbsAlarm_cycle

マクロ名	OSError_SetAbsAlarm_cycle()≪COS3836≫
概要	システムサービスエラー発生時の SetAbsAlarm 実行第 3 引数の取得.

OSError_CancelAlarm_AlarmID

マクロ名	OSError_CancelAlarm_AlarmID()≪COS3836≫
概要	システムサービスエラー発生時の CancelAlarm 実行第 1 引数の取得.

OSError_StartScheduleTableRel_ScheduleTableID

マクロ名	OSError_StartScheduleTableRel_ScheduleTableID()≪COS3836≫
概要	システムサービスエラー発生時の StartScheduleTableRel 実行第 1 引数の取得.

OSError_StartScheduleTableRel_Offset

マクロ名	OSError_StartScheduleTableRel_Offset()≪COS3836≫
概要	システムサービスエラー発生時の StartScheduleTableRel 実行第 2 引数の取得.

OSError_StartScheduleTableAbs_ScheduleTableID

マクロ名	OSError_StartScheduleTableAbs_ScheduleTableID()≪COS3836≫
概要	システムサービスエラー発生時の StartScheduleTableAbs 実行第 1 引数の取得.

OSError_StartScheduleTableAbs_Start

マクロ名	OSError_StartScheduleTableAbs_Start()≪COS3836≫
概要	システムサービスエラー発生時の StartScheduleTableAbs 実行第 2 引数の取得.

OSError_StopScheduleTable_ScheduleTableID

マクロ名	OSError_StopScheduleTable_ScheduleTableID()≪COS3836≫
概要	システムサービスエラー発生時の StopScheduleTable 実行第 1 引数の取得.

OSError_NextScheduleTable_ScheduleTableID_From

マクロ名	OSError_NextScheduleTable_ScheduleTableID_From()≪COS3836≫
概要	システムサービスエラー発生時の NextScheduleTable 実行第 1 引数の取得.

OSError_NextScheduleTable_ScheduleTableID_To

マクロ名	OSError_NextScheduleTable_ScheduleTableID_To()≪COS3836≫
概要	システムサービスエラー発生時の NextScheduleTable 実行第 2 引数の取得.

OSError_StartScheduleTableSynchron_ScheduleTableID

マクロ名	OSError_StartScheduleTableSynchron_ScheduleTableID()≪COS3836≫
概要	システムサービスエラー発生時の StartScheduleTableSynchron 実行第 1 引数の取得.

OSError_SyncScheduleTable_ScheduleTableID

マクロ名	OSError_SyncScheduleTable_ScheduleTableID()≪COS3836≫
概要	システムサービスエラー発生時の SyncScheduleTable 実行第 1 引数の取得.

OSError_SyncScheduleTable_Value

マクロ名	OSError_SyncScheduleTable_Value()≪COS3836≫
概要	システムサービスエラー発生時の SyncScheduleTable 実行第 2 引数の取得.

OSError_SetScheduleTableAsync_ScheduleTableID

マクロ名	OSError_SetScheduleTableAsync_ScheduleTableID()≪COS3836≫
概要	システムサービスエラー発生時の SetScheduleTableAsync 実行第 1 引数の取得.

OSError_GetScheduleTableStatus_ScheduleTableID

マクロ名	OSError_GetScheduleTableStatus_ScheduleTableID()≪COS3836≫
概要	システムサービスエラー発生時の GetScheduleTableStatus 実行第 1 引数の取得.

OSError_GetScheduleTableStatus_ScheduleStatus

マクロ名	OSError_GetScheduleTableStatus_ScheduleStatus()≪COS3836≫
概要	システムサービスエラー発生時の GetScheduleTableStatus 実行第 2 引数の取得.

OSError_CallTrustedFunction_FunctionIndex

マクロ名	OSError_CallTrustedFunction_FunctionIndex()≪COS3836≫
概要	システムサービスエラー発生時の CallTrustedFunction 実行第 1 引数の取得.

OSError_CallTrustedFunction_FunctionParams

マクロ名	OSError_CallTrustedFunction_FunctionParams()≪COS3836≫
概要	システムサービスエラー発生時の CallTrustedFunction 実行第 2 引数の取得.

OSError_CheckISRMemoryAccess_ISRID

マクロ名	OSError_CheckISRMemoryAccess_ISRID()≪COS3836≫
概要	システムサービスエラー発生時の CheckISRMemoryAccess 実行第 1 引数の取得.

OSError_CheckISRMemoryAccess_Address

マクロ名	OSError_CheckISRMemoryAccess_Address()≪COS3836≫
概要	システムサービスエラー発生時の CheckISRMemoryAccess 実行第 2 引数の取得.

OSError_CheckISRMemoryAccess_Size

マクロ名	OSError_CheckISRMemoryAccess_Size()≪COS3836≫
概要	システムサービスエラー発生時の CheckISRMemoryAccess 実行第 3 引数の取得.

OSError_CheckTaskMemoryAccess_TaskID

マクロ名	OSError_CheckTaskMemoryAccess_TaskID()≪COS3836≫
概要	システムサービスエラー発生時の CheckTaskMemoryAccess 実行第 1 引数の取得.

OSError_CheckTaskMemoryAccess_Address

マクロ名	OSError_CheckTaskMemoryAccess_Address()≪COS3836≫
概要	システムサービスエラー発生時の CheckTaskMemoryAccess 実行第 2 引数の取得.

OSError_CheckTaskMemoryAccess_Size

マクロ名	OSError_CheckTaskMemoryAccess_Size()≪COS3836≫
概要	システムサービスエラー発生時の CheckTaskMemoryAccess 実行第 3 引数の取得.

OSError_CheckTaskAccess_ApplID

マクロ名	OSError_CheckTaskAccess_ApplID()≪COS3836≫
概要	システムサービスエラー発生時の CheckTaskAccess 実行第 1 引数の取得.

OSError_CheckTaskAccess_TaskID

マクロ名	OSError_CheckTaskAccess_TaskID()≪COS3836≫
概要	システムサービスエラー発生時の CheckTaskAccess 実行第 2 引数の取得.

OSError_CheckISRAccess_ApplID

マクロ名	OSError_CheckISRAccess_ApplID()≪COS3836≫
概要	システムサービスエラー発生時の CheckISRAccess 実行第 1 引数の取得.

OSError_CheckISRAccess_ISRID

マクロ名	OSError_CheckISRAccess_ISRID()≪COS3836≫
概要	システムサービスエラー発生時の CheckISRAccess 実行第 2 引数の取得.

OSError_CheckAlarmAccess_ApplID

マクロ名	OSError_CheckAlarmAccess_ApplID()≪COS3836≫
概要	システムサービスエラー発生時の CheckAlarmAccess 実行第 1 引数の取得.

OSError_CheckAlarmAccess_AlarmID

マクロ名	OSError_CheckAlarmAccess_AlarmID()≪COS3836≫
概要	システムサービスエラー発生時の CheckAlarmAccess 実行第 2 引数の取得.

OSError_CheckResourceAccess_ApplID

マクロ名	OSError_CheckResourceAccess_ApplID()≪COS3836≫
概要	システムサービスエラー発生時の CheckResourceAccess 実行第 1 引数の取得.

OSError_CheckResourceAccess_ResID

マクロ名	OSError_CheckResourceAccess_ResID()≪COS3836≫
概要	システムサービスエラー発生時の CheckResourceAccess 実行第 2 引数の取得.

OSError_CheckCounterAccess_ApplID

マクロ名	OSError_CheckCounterAccess_ApplID()≪COS3836≫
概要	システムサービスエラー発生時の CheckCounterAccess 実行第 1 引数の取得.

OSError_CheckCounterAccess_CounterID

マクロ名	OSError_CheckCounterAccess_CounterID()≪COS3836≫
概要	システムサービスエラー発生時の CheckCounterAccess 実行第 2 引数の取得.

OSError_CheckScheduleTableAccess_ApplID

マクロ名	OSError_CheckScheduleTableAccess_ApplID()≪COS3836≫
概要	システムサービスエラー発生時の CheckScheduleTableAccess 実行第 1 引数の取得.

OSError_CheckScheduleTableAccess_ScheduleTableID

マクロ名	OSError_CheckScheduleTableAccess_ScheduleTableID()≪COS3836≫
概要	システムサービスエラー発生時の CheckScheduleTableAccess 実行第 2 引数の取得.

OSError_CheckSpinlockAccess_ApplID

マクロ名	OSError_CheckSpinlockAccess_ApplID()≪COS3836≫
概要	システムサービスエラー発生時の CheckSpinlockAccess 実行第 1 引数の取得.

OSError_CheckSpinlockAccess_SpinlockId

マクロ名	OSError_CheckSpinlockAccess_SpinlockId()≪COS3836≫
概要	システムサービスエラー発生時の CheckSpinlockAccess 実行第 2 引数の取得.

OSError_CheckTaskOwnership_TaskID

マクロ名	OSError_CheckTaskOwnership_TaskID()≪COS3836≫
概要	システムサービスエラー発生時の CheckTaskOwnership 実行第 1 引数の取得.

OSError_CheckISROwnership_ISRID

マクロ名	OSError_CheckISROwnership_ISRID()≪COS3836≫
概要	システムサービスエラー発生時の CheckISROwnership 実行第 1 引数の取得.

OSError_CheckAlarmOwnership_AlarmID

マクロ名	OSError_CheckAlarmOwnership_AlarmID()≪COS3836≫
概要	システムサービスエラー発生時の CheckAlarmOwnership 実行第 1 引数の取得.

OSError_CheckCounterOwnership_CounterID

マクロ名	OSError_CheckCounterOwnership_CounterID()≪COS3836≫
概要	システムサービスエラー発生時の CheckCounterOwnership 実行第 1 引数の取得.

OSError_CheckScheduleTableOwnership_ScheduleTableID

マクロ名	OSError_CheckScheduleTableOwnership_ScheduleTableID()≪COS3836≫
概要	システムサービスエラー発生時の CheckScheduleTableOwnership 実行第 1 引数の取得.

OSError_TerminateApplication_Application

マクロ名	OSError_TerminateApplication_Application()≪COS3836≫
概要	システムサービスエラー発生時の TerminateApplication 実行第 1 引数の取得.

OSError_TerminateApplication_RestartOption

マクロ名	OSError_TerminateApplication_RestartOption()≪COS3836≫
概要	システムサービスエラー発生時の TerminateApplication 実行第 2 引数の取得.

OSError_GetApplicationState_Application

マクロ名	OSError_GetApplicationState_Application()≪COS3836≫
概要	システムサービスエラー発生時の GetApplicationState 実行第 1 引数の取得.

OSError_GetApplicationState_Value

マクロ名	OSError_GetApplicationState_Value()≪COS3836≫
概要	システムサービスエラー発生時の GetApplicationState 実行第 2 引数の取得.

OSError_IocSend_IocId

マクロ名	OSError_IocSend_IocId()≪NOS1088≫
概要	IOC 用システムサービスエラー発生時の IocSend_<IocId>[_<SenderId>], IocSendGroup_<IocId>で実行された<IocId>の取得.

OSError_IocSend_SenderId

マクロ名	OSError_IocSend_SenderId()≪NOS1088≫
概要	IOC 用システムサービスエラー発生時の IocSend_<IocId>_<SenderId>で実行された<SenderId>の取得.

OSError_IocWrite_IocId

マクロ名	OSError_IocWrite_IocId()≪NOS1088≫
概要	IOC 用システムサービスエラー発生時の IocWrite_<IocId>[_<SenderId>]，IocWriteGroup_<IocId>で実行された<IocId>の取得.

OSError_IocWrite_SenderId

マクロ名	OSError_IocWrite_SenderId()≪NOS1088≫
概要	IOC 用システムサービスエラー発生時の IocWrite_<IocId>_<SenderId>で実行された<SenderId>の取得.

OSError_IocReceive_IocId

マクロ名	OSError_IocReceive_IocId()≪NOS1088≫
概要	IOC 用システムサービスエラー発生時の IocReceive_<IocId>, IocReceiveGroup_<IocId>で実行された<IocId>の取得.

OSError_IocRead_IocId

マクロ名	OSError_IocRead_IocId()≪NOS1088≫
概要	IOC 用システムサービスエラー発生時の IocRead_<IocId>, IocReadGroup_<IocId>で実行された<IocId>の取得.

OSError_IocEmptyQueue_IocId

マクロ名	OSError_IocEmptyQueue_IocId()≪NOS1088≫
概要	IOC 用システムサービスエラー発生時の IocEmptyQueue_<IocId> で実行された<IocId>の取得.

OSError_StartCore_CoreID

マクロ名	OSError_StartCore_CoreID()≪COS3836≫
概要	システムサービスエラー発生時の StartCore 実行第 1 引数の取得.

OSError_StartCore_Status

マクロ名	OSError_StartCore_Status()≪COS3836≫
概要	システムサービスエラー発生時の StartCore 実行第 2 引数の取得.

OSError_StartNonAutosarCore_CoreID

マクロ名	OSError_StartNonAutosarCore_CoreID()≪COS3836≫
概要	システムサービスエラー発生時の StartNonAutosarCore 実行第 1 引数の取得.

OSError_StartNonAutosarCore_Status

マクロ名	OSError_StartNonAutosarCore_Status()≪COS3836≫
概要	システムサービスエラー発生時の StartNonAutosarCore 実行第 2 引数の取得.

OSError_GetSpinlock_SpinlockId

マクロ名	OSError_GetSpinlock_SpinlockId()≪COS3836≫
概要	システムサービスエラー発生時の GetSpinlock 実行第 1 引数の取得.

OSError_ReleaseSpinlock_SpinlockId

マクロ名	OSError_ReleaseSpinlock_SpinlockId()≪COS3836≫
概要	システムサービスエラー発生時の ReleaseSpinlock 実行第 1 引数の取得.

OSError_TryToGetSpinlock_SpinlockId

マクロ名	OSError_TryToGetSpinlock_SpinlockId()≪COS3836≫
概要	システムサービスエラー発生時の TryToGetSpinlock 実行第 1 引数の取得.

OSError_TryToGetSpinlock_Success

マクロ名	OSError_TryToGetSpinlock_Success()≪COS3836≫
概要	システムサービスエラー発生時の TryToGetSpinlock 実行第 2 数の取得.

OSError_ShutdownAllCores_Error

マクロ名	OSError_ShutdownAllCores_Error()≪COS3836≫
概要	システムサービスエラー発生時の ShutdownAllCores 実行第 1 引数の取得.

OSError_RaiseInterCoreInterrupt_ISRID

マクロ名	OSError_RaiseInterCoreInterrupt_ISRID ()≪COS3836≫
概要	システムサービスエラー発生時の RaiseInterCoreInterrupt 実行第 1 引数の取得.

OSError_GetMutex_MtxID

マクロ名	OSError_GetMutex_MtxID()≪COS3836≫
概要	システムサービスエラー発生時の GetMutex 実行第 1 引数の取得.

OSError_ReleaseMutex_MtxID

マクロ名	OSError_ReleaseMutex_MtxID()≪COS3836≫
概要	システムサービスエラー発生時の ReleaseMutex 実行第 1 引数の取得.

3.6　定数

3.6.1　タスク状態

RUNNING

定数名	RUNNING【COS3288】
概要	タスクが実行状態であることを示す定数

WAITING

定数名	WAITING【COS3289】
概要	タスクが待ち状態であることを示す定数

READY

定数名	READY【COS3290】
概要	タスクが実行可能状態であることを示す定数

SUSPENDED

定数名	SUSPENDED【COS3291】
概要	タスクが休止状態であることを示す定数

3.6.2　ID

INVALID_TASK

定数名	INVALID_TASK【COS3292】
概要	いずれのタスクの ID でもないことを示す定数

INVALID_ISR

定数名	INVALID_ISR【OSa036】
概要	いずれの ISR の ID でもないことを示す定数

INVALID_OSAPPLICATION

定数名	INVALID_OSAPPLICATION【OSa037】
概要	いずれの OSAPID でもないことを示す定数

OS_CORE_ID_MASTER

定数名	OS_CORE_ID_MASTER【OS628】
概要	マスタコアのコア ID を示す定数. マスタコアとなるコア ID を示す OS_CORE_ID_<No>と同じ値となる.

OS_CORE_ID_<No>

定数名	OS_CORE_ID_<No>【OS627】
概要	各コアのコア ID を示す定数. <No>は，0 から OsNumberOfCores-1 までの連番となる. 本マクロの値は実装定義である.

3.6.3　アラーム設定

以下3つの定数はすべてのカウンタに対してGetAlarmBaseで取得できるパラメータを定数値として定義するものである．よって，カウンタの名前が自明な場合はシステムサービスを使用してアラーム情報を取得する必要がない【COS3685】.

OSMAXALLOWEDVALUE_<Cnt>

定数名	OSMAXALLOWEDVALUE_<Cnt>【COS3682】
概要	カウンタオブジェクト<Cnt>に対してシステムサービスで指定できるティックの最大値. すべてのカウンタに対して定義する.

OSTICKSPERBASE_<Cnt>

定数名	OSTICKSPERBASE_<Cnt>【COS3683】
概要	カウンタオブジェクト<Cnt>のカウンタ固有の値(OS は不使用). すべてのカウンタに対して定義する.

OSMINCYCLE_<Cnt>

定数名	OSMINCYCLE_<Cnt>【COS3684】
概要	カウンタオブジェクト<Cnt>に接続されたアラームのセットに指定できるサイクルの最小値. すべてのカウンタに対して定義する.

3.6.4　スケジュールテーブル状態

SCHEDULETABLE_STOPPED

定数名	SCHEDULETABLE_STOPPED【OSa038】
概要	ScheduleTableStatusType 型のスケジュールテーブルが停止状態であることを示す定数

SCHEDULETABLE_NEXT

定数名	SCHEDULETABLE_NEXT【OSa039】
概要	ScheduleTableStatusType 型のスケジュールテーブルが切換え待ち状態であることを示す定数

SCHEDULETABLE_WAITING

定数名	SCHEDULETABLE_WAITING【OSa040】
概要	ScheduleTableStatusType 型のスケジュールテーブルが同期待ち状態であることを示す定数

SCHEDULETABLE_RUNNING

定数名	SCHEDULETABLE_RUNNING【OSa041】
概要	ScheduleTableStatusType 型のスケジュールテーブルが動作状態であることを示す定数

SCHEDULETABLE_RUNNING_AND_SYNCHRONOUS

定数名	SCHEDULETABLE_RUNNING_AND_SYNCHRONOUS【OSa042】
概要	ScheduleTableStatusType 型のスケジュールテーブルが同期動作状態であることを示す定数

3.6.5　OSAP 状態

APPLICATION_ACCESSIBLE

定数名	APPLICATION_ACCESSIBLE【OSa043】
概要	ApplicationStateType 型の OSAP が利用可能状態であることを示す定数

APPLICATION_RESTARTING

定数名	APPLICATION_RESTARTING【OSa044】
概要	ApplicationStateType 型の OSAP が再起動状態であることを示す定数

APPLICATION_TERMINATED

定数名	APPLICATION_TERMINATED【OSa045】
概要	ApplicationStateType 型の OSAP が終了状態であることを示す定数

3.6.6　アクセス情報

ACCESS

定数名	ACCESS【OSa046】
概要	ObjectAccessType 型の OS オブジェクトにアクセスできることを示す定数

NO_ACCESS

定数名	NO_ACCESS【OSa047】
概要	ObjectAccessType 型の OS オブジェクトにアクセスできないことを示す定数

3.6.7　オブジェクト情報

OBJECT_TASK

定数名	OBJECT_TASK【OSa048】
概要	ObjectTypeType 型の OS オブジェクトがタスクであることを示す定数

OBJECT_ISR

定数名	OBJECT_ISR【OSa049】
概要	ObjectTypeType 型の OS オブジェクトが ISR であることを示す定数

OBJECT_ALARM

定数名	OBJECT_ALARM【OSa050】
概要	ObjectTypeType 型の OS オブジェクトがアラームであることを示す定数

OBJECT_RESOURCE

定数名	OBJECT_RESOURCE【OSa051】
概要	ObjectTypeType 型の OS オブジェクトがリソースであることを示す定数

OBJECT_COUNTER

定数名	OBJECT_COUNTER【OSa052】
概要	ObjectTypeType 型の OS オブジェクトがカウンタであることを示す定数

OBJECT_SCHEDULETABLE

定数名	OBJECT_SCHEDULETABLE【OSa053】
概要	ObjectTypeType 型の OS オブジェクトがスケジュールテーブルであることを示す定数

OBJECT_SPINLOCK

定数名	OBJECT_SPINLOCK【NOS0950】
概要	ObjectTypeType 型の OS オブジェクトがスピンロックであることを示す定数

3.6.8　リスタート情報

RESTART

定数名	RESTART【OSa054】
概要	RestartType 型のリスタートタスクを起動することを示す定数

NO_RESTART

定数名	NO_RESTART【OSa055】
概要	RestartType 型のリスタートタスクを起動しないことを示す定数

3.6.9　サービス ID

OSServiceId_GetActiveApplicationMode

定数名	OSServiceId_GetActiveApplicationMode≪COS3835≫
概要	GetActiveApplicationMode のサービス ID

OSServiceId_StartOS

定数名	OSServiceId_StartOS≪COS3835≫
概要	StartOS のサービス ID

OSServiceId_ShutdownOS

定数名	OSServiceId_ShutdownOS≪COS3835≫
概要	ShutdownOS のサービス ID

OSServiceId_ActivateTask

定数名	OSServiceId_ActivateTask≪COS3835≫
概要	ActivateTask のサービス ID

OSServiceId_TerminateTask

定数名	OSServiceId_TerminateTask≪COS3835≫
概要	TerminateTask のサービス ID

OSServiceId_ChainTask

定数名	OSServiceId_ChainTask≪COS3835≫
概要	ChainTask のサービス ID

OSServiceId_Schedule

定数名	OSServiceId_Schedule≪COS3835≫
概要	Schedule のサービス ID

OSServiceId_GetTaskID

定数名	OSServiceId_GetTaskID≪COS3835≫
概要	GetTaskID のサービス ID

OSServiceId_GetTaskState

定数名	OSServiceId_GetTaskState≪COS3835≫
概要	GetTaskState のサービス ID

OSServiceId_EnableAllInterrupts

定数名	OSServiceId_EnableAllInterrupts≪COS3835≫
概要	EnableAllInterrupts のサービス ID

OSServiceId_DisableAllInterrupts

定数名	OSServiceId_DisableAllInterrupts≪COS3835≫
概要	DisableAllInterrupts のサービス ID

OSServiceId_ResumeAllInterrupts

定数名	OSServiceId_ResumeAllInterrupts≪COS3835≫
概要	ResumeAllInterrupts のサービス ID

OSServiceId_SuspendAllInterrupts

定数名	OSServiceId_SuspendAllInterrupts≪COS3835≫
概要	SuspendAllInterrupts のサービス ID

OSServiceId_ResumeOSInterrupts

定数名	OSServiceId_ResumeOSInterrupts≪COS3835≫
概要	ResumeOSInterrupts のサービス ID

OSServiceId_SuspendOSInterrupts

定数名	OSServiceId_SuspendOSInterrupts≪COS3835≫
概要	SuspendOSInterrupts のサービス ID

OSServiceId_GetISRID

定数名	OSServiceId_GetISRID≪COS3835≫
概要	GetISRID のサービス ID

OSServiceId_EnableInterruptSource

定数名	OSServiceId_EnableInterruptSource≪COS3835≫
概要	EnableInterruptSource のサービス ID

OSServiceId_DisableInterruptSource

定数名	OSServiceId_DisableInterruptSource≪COS3835≫
概要	DisableInterruptSource のサービス ID

OSServiceId_SetEvent

定数名	OSServiceId_SetEvent≪COS3835≫
概要	SetEvent のサービス ID

OSServiceId_ClearEvent

定数名	OSServiceId_ClearEvent≪COS3835≫
概要	ClearEvent のサービス ID

OSServiceId_GetEvent

定数名	OSServiceId_GetEvent≪COS3835≫
概要	GetEvent のサービス ID

OSServiceId_WaitEvent

定数名	OSServiceId_WaitEvent≪COS3835≫
概要	WaitEvent のサービス ID

OSServiceId_GetResource

定数名	OSServiceId_GetResource≪COS3835≫
概要	GetResource のサービス ID

OSServiceId_ReleaseResource

定数名	OSServiceId_ReleaseResource≪COS3835≫
概要	ReleaseResource のサービス ID

OSServiceId_IncrementCounter

定数名	OSServiceId_IncrementCounter≪COS3835≫
概要	IncrementCounter のサービス ID

OSServiceId_GetCounterValue

定数名	OSServiceId_GetCounterValue≪COS3835≫
概要	GetCounterValue のサービス ID

OSServiceId_GetElapsedValue

定数名	OSServiceId_GetElapsedValue≪COS3835≫
概要	GetElapsedValue のサービス ID

OSServiceId_GetAlarmBase

定数名	OSServiceId_GetAlarmBase≪COS3835≫
概要	GetAlarmBase のサービス ID

OSServiceId_GetAlarm

定数名	OSServiceId_GetAlarm≪COS3835≫
概要	GetAlarm のサービス ID

OSServiceId_SetRelAlarm

定数名	OSServiceId_SetRelAlarm≪COS3835≫
概要	SetRelAlarm のサービス ID

OSServiceId_SetAbsAlarm

定数名	OSServiceId_SetAbsAlarm≪COS3835≫
概要	SetAbsAlarm のサービス ID

OSServiceId_CancelAlarm

定数名	OSServiceId_CancelAlarm≪COS3835≫
概要	CancelAlarm のサービス ID

OSServiceId_StartScheduleTableRel

定数名	OSServiceId_StartScheduleTableRel≪COS3835≫
概要	StartScheduleTableRel のサービス ID

OSServiceId_StartScheduleTableAbs

定数名	OSServiceId_StartScheduleTableAbs≪COS3835≫
概要	StartScheduleTableAbs のサービス ID

OSServiceId_StopScheduleTable

定数名	OSServiceId_StopScheduleTable≪COS3835≫
概要	StopScheduleTable のサービス ID

OSServiceId_NextScheduleTable

定数名	OSServiceId_NextScheduleTable≪COS3835≫
概要	NextScheduleTable のサービス ID

OSServiceId_StartScheduleTableSynchron

定数名	OSServiceId_StartScheduleTableSynchron≪COS3835≫
概要	StartScheduleTableSynchron のサービス ID

OSServiceId_SyncScheduleTable

定数名	OSServiceId_SyncScheduleTable≪COS3835≫
概要	SyncScheduleTable のサービス ID

OSServiceId_SetScheduleTableAsync

定数名	OSServiceId_SetScheduleTableAsync≪COS3835≫
概要	SetScheduleTableAsync のサービス ID

OSServiceId_GetScheduleTableStatus

定数名	OSServiceId_GetScheduleTableStatus≪COS3835≫
概要	GetScheduleTableStatus のサービス ID

OSServiceId_GetApplicationID

定数名	OSServiceId_GetApplicationID≪COS3835≫
概要	GetApplicationID のサービス ID

OSServiceId_CallTrustedFunction

定数名	OSServiceId_CallTrustedFunction≪COS3835≫
概要	CallTrustedFunction のサービス ID

OSServiceId_CheckISRMemoryAccess

定数名	OSServiceId_CheckISRMemoryAccess≪COS3835≫
概要	CheckISRMemoryAccess のサービス ID

OSServiceId_CheckTaskMemoryAccess

定数名	OSServiceId_CheckTaskMemoryAccess≪COS3835≫
概要	CheckTaskMemoryAccess のサービス ID

OSServiceId_CheckTaskAccess

定数名	OSServiceId_CheckTaskAccess≪COS3835≫
概要	CheckTaskAccess のサービス ID

OSServiceId_CheckISRAccess

定数名	OSServiceId_CheckISRAccess≪COS3835≫
概要	CheckISRAccess のサービス ID

OSServiceId_CheckAlarmAccess

定数名	OSServiceId_CheckAlarmAccess≪COS3835≫
概要	CheckAlarmAccess のサービス ID

OSServiceId_CheckResourceAccess

定数名	OSServiceId_CheckResourceAccess≪COS3835≫
概要	CheckResourceAccess のサービス ID

OSServiceId_CheckCounterAccess

定数名	OSServiceId_CheckCounterAccess≪COS3835≫
概要	CheckCounterAccess のサービス ID

OSServiceId_CheckScheduleTableAccess

定数名	OSServiceId_CheckScheduleTableAccess≪COS3835≫
概要	CheckScheduleTableAccess のサービス ID

OSServiceId_CheckSpinlockAccess

定数名	OSServiceId_CheckSpinlockAccess≪COS3835≫
概要	CheckSpinlockAccess のサービス ID

OSServiceId_CheckTaskOwnership

定数名	OSServiceId_CheckTaskOwnership≪COS3835≫
概要	CheckTaskOwnership のサービス ID

OSServiceId_CheckISROwnership

定数名	OSServiceId_CheckISROwnership≪COS3835≫
概要	CheckISROwnership のサービス ID

OSServiceId_CheckAlarmOwnership

定数名	OSServiceId_CheckAlarmOwnership≪COS3835≫
概要	CheckAlarmOwnership のサービス ID

OSServiceId_CheckCounterOwnership

定数名	OSServiceId_CheckCounterOwnership≪COS3835≫
概要	CheckCounterOwnership のサービス ID

OSServiceId_CheckScheduleTableOwnership

定数名	OSServiceId_CheckScheduleTableOwnership≪COS3835≫
概要	CheckScheduleTableOwnership のサービス ID

OSServiceId_TerminateApplication

定数名	OSServiceId_TerminateApplication≪COS3835≫
概要	TerminateApplication のサービス ID

OSServiceId_AllowAccess

定数名	OSServiceId_AllowAccess≪COS3835≫
概要	AllowAccess のサービス ID

OSServiceId_GetApplicationState

定数名	OSServiceId_GetApplicationState≪COS3835≫
概要	GetApplicationState のサービス ID

IOCServiceId_IOC_Send

定数名	IOCServiceId_IOC_Send≪OS718≫
概要	IocSend のサービス ID

IOCServiceId_IOC_Write

定数名	IOCServiceId_IOC_Write≪OS718≫
概要	IocWrite のサービス ID

IOCServiceId_IOC_SendGroup

定数名	IOCServiceId_IOC_SendGroup≪OS728≫
概要	IocSendGroup のサービス ID

IOCServiceId_IOC_WriteGroup

定数名	IOCServiceId_IOC_WriteGroup≪OS728≫
概要	IocWriteGroup のサービス ID

IOCServiceId_IOC_Receive

定数名	IOCServiceId_IOC_Receive≪OS738≫
概要	IocReceive のサービス ID

IOCServiceId_IOC_Read

定数名	IOCServiceId_IOC_Read≪OS738≫
概要	IocRead のサービス ID

IOCServiceId_IOC_ReceiveGroup

定数名	IOCServiceId_IOC_ReceiveGroup≪OS746≫
概要	IocReceiveGroup のサービス ID

IOCServiceId_IOC_ReadGroup

定数名	IOCServiceId_IOC_ReadGroup≪OS746≫
概要	IocReadGroup のサービス ID

IOCServiceId_IOC_EmptyQueue

定数名	IOCServiceId_IOC_EmptyQueue≪OS754≫
概要	IocEmptyQueue のサービス ID

OSServiceId_StartCore

定数名	OSServiceId_StartCore≪COS3835≫
概要	StartCore のサービス ID

OSServiceId_StartNonAutosarCore

定数名	OSServiceId_StartNonAutosarCore≪COS3835≫
概要	StartNonAutosarCore のサービス ID

OSServiceId_GetSpinlock

定数名	OSServiceId_GetSpinlock≪COS3835≫
概要	GetSpinlock のサービス ID

OSServiceId_ReleaseSpinlock

定数名	OSServiceId_ReleaseSpinlock≪COS3835≫
概要	ReleaseSpinlock のサービス ID

OSServiceId_TryToGetSpinlock

定数名	OSServiceId_TryToGetSpinlock≪COS3835≫
概要	TryToGetSpinlock のサービス ID

OSServiceId_ShutdownAllCores

定数名	OSServiceId_ShutdownAllCores≪COS3835≫
概要	ShutdownAllCores のサービス ID

OSServiceId_RaiseInterCoreInterrupt

定数名	OSServiceId_RaiseInterCoreInterrupt≪COS3835≫
概要	RaiseInterCoreInterrupt のサービス ID

OSServiceId_GetMutex

定数名	OSServiceId_GetMutex≪COS3835≫
概要	GetMutex のサービス ID

OSServiceId_ReleaseMutex

定数名	OSServiceId_ReleaseMutex≪COS3835≫
概要	ReleaseMutex のサービス ID

OSServiceId_TaskMissingEnd

定数名	OSServiceId_TaskMissingEnd【NOS0155】
概要	タスクの不正終了時のサービス ID

OSServiceId_ISRMissingEnd

定数名	OSServiceId_ISRMissingEnd【NOS0158】
概要	C2ISR，ICISR の不正終了時のサービス ID

OSServiceId_HookMissingEnd

定数名	OSServiceId_HookMissingEnd【NOS0647】
概要	フックルーチンの不正終了時のサービス ID

3.6.10 プロテクションフックの返り値

PRO_IGNORE

定数名	PRO_IGNORE【OSa056】
概要	ProtectionReturnType 型の保護違反時処理を行わないことを示す定数

PRO_TERMINATETASKISR

定数名	PRO_TERMINATETASKISR【OSa057】
概要	ProtectionReturnType 型の保護違反時処理としてタスク，C2ISR を強制終了することを示す定数

PRO_TERMINATEAPPL

定数名	PRO_TERMINATEAPPL【OSa058】
概要	ProtectionReturnType 型の保護違反時処理として OSAP を強制終了することを示す定数

PRO_TERMINATEAPPL_RESTART

定数名	PRO_TERMINATEAPPL_RESTART【OSa059】
概要	ProtectionReturnType 型の保護違反時処理として OSAP を強制終了し，リスタートタスクを起動することを示す定数

PRO_SHUTDOWN

定数名	PRO_SHUTDOWN【OSa060】
概要	ProtectionReturnType 型の保護違反時処理として OS をシャットダウンすることを示す定数

3.6.11 処理単位情報

FC_TASK

定数名	FC_TASK【IOS193】
概要	タスクが保護違反を起こしたことを示す定数

FC_C2ISR

定数名	FC_C2ISR【IOS194】
概要	C2ISR が保護違反を起こしたことを示す定数

FC_SYSTEM_HOOK

定数名	FC_SYSTEM_HOOK【IOS195】
概要	システム定義のフックルーチンが保護違反を起こしたことを示す定数

FC_OSAP_HOOK

定数名	FC_OSAP_HOOK【IOS196】
概要	OSAP 固有のフックルーチンが保護違反を起こしたことを示す定数

FC_TRUSTED_FUNC

定数名	FC_TRUSTED_FUNC【IOS197】
概要	信頼関数が保護違反を起こしたことを示す定数

FC_INVALID

定数名	FC_INVALID【IOS198】
概要	保護違反を起こした処理単位を特定できないことを示す定数

3.6.12 無効値

INVALID_APPMODETYPE

定数名	INVALID_APPMODETYPE【NOS0344】
概要	GetActiveApplicationMode がエラーを検出したときに返る定数

3.7　宣言記述

AUTOSAR 仕様では，移植性のために，ジェネレータが生成する OS オブジェクトの実現方法に関係なく，ユーザは OS オブジェクトにアクセスできると説明されている≪OSa004≫．しかし，OSEK 仕様に準拠して開発されたアプリケーションとの互換性のため，以下に示す OS オブジェクトへの参照の宣言があってもエラーとしない【NOS0451】.

3.7.1　DeclareTask

宣言記述名	DeclareTask(TaskIdentifier)【COS3205】
概要	TaskIdentifier で指定されたタスクオブジェクトの外部参照宣言
コンフォーマンスクラス	BCC1，BCC2，ECC1，ECC2
スケーラビリティクラス	SC1，SC2，SC3，SC4

3.7.2　DeclareEvent

宣言記述名	DeclareEvent(EventIdentifier)【COS3503】
概要	EventIdentifier で指定されたイベントオブジェクトの外部参照宣言
コンフォーマンスクラス	ECC1，ECC2
スケーラビリティクラス	SC1，SC2，SC3，SC4

3.7.3　DeclareResource

宣言記述名	DeclareResource(ResourceIdentifier)【COS3402】
概要	ResourceIdentifier で指定されたリソースオブジェクトの外部参照宣言
コンフォーマンスクラス	BCC1，BCC2，ECC1，ECC2
スケーラビリティクラス	SC1，SC2，SC3，SC4

3.7.4 DeclareAlarm

宣言記述名	DeclareAlarm(AlarmIdentifier)【COS3610】
概要	AlarmIdentifier で指定されたアラームオブジェクトの外部参照宣言
コンフォーマンスクラス	BCC1，BCC2，ECC1，ECC2
スケーラビリティクラス	SC1，SC2，SC3，SC4

3.8　オブジェクトコンテナ

OS で設定するパラメータのコンフィギュレーションクラスは，すべてプレコンパイルタイムである【OS558】〔COS1225〕〔COS1201〕．したがって，各パラメータに対するコンフィギュレーションクラスの説明は省略する．

ATK2 では，以下のパラメータのみプレコンパイルタイムとして，他のパラメータはすべてリンクタイムとする【IOS172】.

- /AUTOSAR/EcucDefs/Os/OsOS/OsScalabilityClass
- /AUTOSAR/EcucDefs/Os/OsOS/OsViolationHandlingLevel
- /AUTOSAR/EcucDefs/Os/OsOS/OsTimingProtectionLevel
- /AUTOSAR/EcucDefs/Os/OsOS/OsMemoryProtectionLevel
- /AUTOSAR/EcucDefs/Os/OsOS/OsStackMonitoring
- /AUTOSAR/EcucDefs/Os/OsOS/OsStatus
- /AUTOSAR/EcucDefs/Os/OsOS/OsUseGetServiceId
- /AUTOSAR/EcucDefs/Os/OsOS/OsUseParameterAccess
- /AUTOSAR/EcucDefs/Os/OsOS/OsHooks/OsErrorHook
- /AUTOSAR/EcucDefs/Os/OsOS/OsHooks/OsPostTaskHook
- /AUTOSAR/EcucDefs/Os/OsOS/OsHooks/OsPreTaskHook
- /AUTOSAR/EcucDefs/Os/OsOS/OsHooks/OsProtectionHook
- /AUTOSAR/EcucDefs/Os/OsOS/OsHooks/OsShutdownHook
- /AUTOSAR/EcucDefs/Os/OsOS/OsHooks/OsStartupHook
- /AUTOSAR/EcucDefs/Os/OsApplication/OsApplicationHooks/OsAppErrorHook (※1)
- /AUTOSAR/EcucDefs/Os/OsApplication/OsApplicationHooks/OsAppShutdownHook (※1)
- /AUTOSAR/EcucDefs/Os/OsApplication/OsApplicationHooks/OsAppStartupHook (※1)

※1：すべての OSAP において false となるか，1 つでも true である OSAP が存在するかの切り替わりで，プレコンパイルタイムとなる．

3.8.1　Os

コンテナ名	/AUTOSAR/EcucDefs/Os【NOS0892】
概要	OS トップコンテナ
多重度	0 .. 1
パラメータ	－
サブコンテナ	OsOS
	OsAppMode
	OsTask
	OsIsr
	OsEvent
	OsResource
	OsCounter
	OsAlarm
	OsScheduleTable
	OsApplication
	OsMemoryRegion
	OsMemorySection
	OsMemoryModule
	OsMemoryArea
	OsLinkSection
	OsStandardMemoryRegion
	OsIoc
	OsSpinlock
	OsInterCoreInterrupt
	OsInclude
	OsMutex
制限事項	本コンテナが存在しない場合，ジェネレータはエラーを検出する【NOS0893】.

3.8.2　OsOS

コンテナ名	/AUTOSAR/EcucDefs/Os/OsOS【OS_Conf044】
概要	OS 設定コンテナ
多重度	1
パラメータ	OsScalabilityClass
	OsViolationHandlingLevel
	OsTimingProtectionLevel
	OsMemoryProtectionLevel
	OsStackMonitoring
	OsStatus
	OsUseGetServiceId
	OsUseParameterAccess
	OsNumberOfCores
	OsMasterCoreId
サブコンテナ	OsHooks
	OsHookStack
	OsNonTrustedHookStack
	OsOsStack
	OsTimingProtection

OsScalabilityClass

パラメータ名	/AUTOSAR/EcucDefs/Os/OsOS/ OsScalabilityClass【OS_Conf259】
概要	スケーラビリティクラス. 本パラメータが省略された場合，実装定義のデフォルト値が選択される【NOS0446】.
型	列挙型
値の範囲	適用する SC1，SC2，SC3，SC4
多重度	0 .. 1
制限事項	ATK2 では，スケーラビリティクラス毎にソースコードを分けているため，OsScalabilityClass を省略した場合，使用したソースコードのスケーラビリティクラスとなる【IOS125】.

OsViolationHandlingLevel

パラメータ名	/AUTOSAR/EcucDefs/Os/OsOS/ OsViolationHandlingLevel【NOS0273】
概要	保護違反時処理機能の機能レベル. 本パラメータが省略された場合，実装定義のデフォルト値が選択される【NOS0447】.
型	列挙型
値の範囲	Level1：機能レベル 1 Level2：機能レベル 2 Level3：機能レベル 3
多重度	0 .. 1
制限事項	SC2，SC3，SC4 のみで使用可能. ATK2 では，OsViolationHandlingLevel を省略した場合，Level2 が選択される【IOS126】.

OsTimingProtectionLevel

パラメータ名	/AUTOSAR/EcucDefs/Os/OsOS/ OsTimingProtectionLevel【NOS0274】
概要	タイミング保護機能の機能レベル. 本パラメータが省略された場合，実装定義のデフォルト値が選択される【NOS0448】.
型	列挙型
値の範囲	Level1：機能レベル 1 Level2：機能レベル 2 Level3：機能レベル 3
多重度	0 .. 1
制限事項	SC2，SC4 のみで使用可能. ATK2 では，OsTimingProtectionLevel を省略した場合，Level2 が選択される【IOS127】.

OsMemoryProtectionLevel

パラメータ名	/AUTOSAR/EcucDefs/Os/OsOS/ OsMemoryProtectionLevel【NOS0275】
概要	メモリ保護機能の機能レベル. 本パラメータが省略された場合，実装定義のデフォルト値が選択される【NOS0449】.
型	列挙型
値の範囲	Level1：機能レベル 1 Level2：機能レベル 2 Level3：機能レベル 3
多重度	0 .. 1
制限事項	SC3，SC4 のみで使用可能. ATK2 では, OsMemoryProtectionLevel を省略した場合, Level2 が選択される【IOS128】.

OsStackMonitoring

パラメータ名	/AUTOSAR/EcucDefs/Os/OsOS/ OsStackMonitoring【OS_Conf307】
概要	スタックモニタリングの使用有無
型	ブール型
値の範囲	true：スタックモニタリング有効 false：スタックモニタリング無効
多重度	1
制限事項	－

OsStatus

パラメータ名	/AUTOSAR/EcucDefs/Os/OsOS/OsStatus【OS_Conf046】 〔COS1120〕〔COS1123〕〔COS1167〕
概要	エラーコード種別の指定
型	列挙型
値の範囲	STANDARD：標準エラーのみ検出 EXTENDED：標準エラーと拡張エラーを検出
多重度	1
制限事項	SC3，SC4 で本パラメータが EXTENDED でない場合，ジェネレータはエラーを検出する【OS328】.

OsUseGetServiceId

パラメータ名	/AUTOSAR/EcucDefs/Os/OsOS/ OsUseGetServiceId【OS_Conf047】〔COS1169〕
概要	OSErrorGetServiceId()の使用有無
型	ブール型
値の範囲	true：OSErrorGetServiceId 有効 false：OSErrorGetServiceId 無効
多重度	1
制限事項	－

OsUseParameterAccess

パラメータ名	/AUTOSAR/EcucDefs/Os/OsOS/ OsUseParameterAccess【OS_Conf048】
概要	エラーが発生したシステムサービスの引数取得の使用有無
型	ブール型
値の範囲	true：エラーが発生したシステムサービスの引数取得有効 false：エラーが発生したシステムサービスの引数取得無効
多重度	1
制限事項	－

OsNumberOfCores

パラメータ名	/AUTOSAR/EcucDefs/Os/OsOS/ OsNumberOfCores【MCOS_Conf1019】
概要	本 OS を起動するコア数の最大値
型	整数型
値の範囲	1 .. 65535
多重度	0 .. 1
制限事項	マルチコア対応 OS のみで使用可能. 本パラメータに，ハードウェア上に存在するコアの数より大きい値を指定した場合，ジェネレータはエラーを検出する【NOS0951】. 本パラメータを省略した場合，1 とする≪OS583≫.

OsMasterCoreId

パラメータ名	/AUTOSAR/EcucDefs/Os/OsOS/OsMasterCoreId【NOS1151】
概要	マスタコアのコア ID
型	整数型
値の範囲	0 .. 65534
多重度	0 .. 1
制限事項	マルチコア対応 OS のみで使用可能. 本パラメータに，ターゲット定義のマスタコア ID と異なる ID を指定した場合，ジェネレータはエラーを検出する【NOS1152】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，OsUseResScheduler というコンテナで，スケジューラリソースを有効とするか無効とするかを指定している【OS_Conf049】．しかし，AUTOSAR 仕様では，スケジューラリソースをサポートしないため，OsUseResScheduler は使用されない．そのため，OsUseResScheduler コンテナは削除する．

機能レベルの導入に伴い, OsOS コンテナに機能レベルを選択するパラメータを追加した≪NOS0273≫≪NOS0274≫≪NOS0275≫.

一部の BSW は，マスタコアでしか使用できないため，RTE が OS のコンフィギュレーションファイルを読み込んだ際に，どの OSAP がマスタコアに割付いているかを判断できる必要がある．よって，本仕様では，OsMasterCoreId を新設した≪NOS1151≫.

3.8.3　OsHooks

コンテナ名	/AUTOSAR/EcucDefs/Os/OsOS/OsHooks【OS_Conf035】〔COS1109〕
概要	フック設定コンテナ
多重度	1
パラメータ	OsErrorHook
	OsPostTaskHook
	OsPreTaskHook
	OsProtectionHook
	OsShutdownHook
	OsStartupHook
サブコンテナ	－

OsErrorHook

パラメータ名	/AUTOSAR/EcucDefs/Os/OsOS/OsHooks/ OsErrorHook【OS_Conf036】〔COS1109〕〔COS1120〕〔COS1167〕
概要	エラーフックの呼出し有無
型	ブール型
値の範囲	true：システム定義のエラーフック有効 false：システム定義のエラーフック無効
多重度	1
制限事項	－

OsPostTaskHook

パラメータ名	/AUTOSAR/EcucDefs/Os/OsOS/OsHooks/ OsPostTaskHook【OS_Conf037】〔COS1109〕
概要	ポストタスクフックの呼出し有無
型	ブール型
値の範囲	true：ポストタスクフック有効 false：ポストタスクフック無効
多重度	1
制限事項	タスク実行時間には，プレタスクフック，ポストタスクフックの実行時間も含まれるので，SC2，SC4 において，本パラメータに true を指定した場合，ジェネレータは警告を表示する【OS562】.

OsPreTaskHook

パラメータ名	/AUTOSAR/EcucDefs/Os/OsOS/OsHooks/ OsPreTaskHook【OS_Conf038】〔COS1109〕
概要	プレタスクフックの呼出し有無
型	ブール型
値の範囲	true：プレタスクフック有効 false：プレタスクフック無効
多重度	1
制限事項	タスク実行時間には，プレタスクフック，ポストタスクフックの実行時間も含まれるので，SC2，SC4 において，本パラメータに true を指定した場合，ジェネレータは警告を表示する≪OS562≫.

OsProtectionHook

パラメータ名	/AUTOSAR/EcucDefs/Os/OsOS/OsHooks/ OsProtectionHook【OS_Conf214】〔COS1109〕
概要	プロテクションフックの呼出し有無
型	ブール型
値の範囲	true：プロテクションフック有効 false：プロテクションフック無効
多重度	0 .. 1
制限事項	本仕様では，SC1 でもプロテクションフックをサポートするが，AUTOSAR 仕様では SC1 でプロテクションフックをサポートしていないため，本パラメータの多重度は 0..1 である．本パラメータを省略した場合，プロテクションフックを無効とする【NOS0692】.

OsShutdownHook

パラメータ名	/AUTOSAR/EcucDefs/Os/OsOS/OsHooks/ OsShutdownHook【OS_Conf039】〔COS1109〕
概要	シャットダウンフックの呼出し有無
型	ブール型
値の範囲	true：システム定義のシャットダウンフック有効 false：システム定義のシャットダウンフック無効
多重度	1
制限事項	－

OsStartupHook

パラメータ名	/AUTOSAR/EcucDefs/Os/OsOS/OsHooks/ OsStartupHook【OS_Conf040】〔COS1109〕
概要	スタートアップフックの呼出し有無
型	ブール型
値の範囲	true：システム定義のスタートアップフック有効 false：システム定義のスタートアップフック無効
多重度	1
制限事項	－

3.8.4　OsHookStack

コンテナ名	/AUTOSAR/EcucDefs/Os/OsOS/OsHookStack【NOS0824】
概要	SC1，SC2 の場合は，フック用スタック設定コンテナ. SC3，SC4 の場合は，信頼フック用スタック設定コンテナ.
多重度	0 .. *
パラメータ	OsHookStackSize
	OsHookStackStartAddress
	OsHookStackCoreAssignment
サブコンテナ	－
制限事項	OsHookStackCoreAssignment が同じ値の本コンテナを複数指定した場合，ジェネレータはエラーを検出する【NOS1052】. ATK2 では，SC1，SC2 で，フックルーチンを使用する場合に本コンテナを省略した場合, ターゲット定義のサイズでフック用スタックが確保される【IOS078】. ATK2 では，SC3，SC4 で，信頼フック用スタックを使用する場合に本コンテナを省略した場合, ターゲット定義のサイズで信頼フック用スタックが確保される【IOS087】.

OsHookStackSize

パラメータ名	/AUTOSAR/EcucDefs/Os/OsOS/OsHookStack/ OsHookStackSize【NOS0825】
概要	SC1，SC2 の場合は，フック用スタックのサイズ. SC3，SC4 の場合は，信頼フック用スタックのサイズ.
型	整数型
値の範囲	－
多重度	1
制限事項	ATK2 では，OsHookStackStartAddress が指定されている場合，かつ本パラメータで指定された値が, ターゲット定義のアライメント制約を満たしていない場合，ジェネレータはエラーを検出する【IOS079】.

OsHookStackStartAddress

パラメータ名	/AUTOSAR/EcucDefs/Os/OsOS/OsHookStack/ OsHookStackStartAddress【IOS080】
概要	システム定義のフックルーチンおよび信頼フック用スタックの先頭番地. 本パラメータを省略した場合，ジェネレータが， OsHookStackSize で指定したサイズのスタック領域を確保する.
型	整数型
値の範囲	－
多重度	0 .. 1
制限事項	SC3，SC4 のメモリ保護機能の機能レベル 3 のみで使用可能(現状，機能レベル 3 は未実装). 本パラメータで指定された値が，ターゲット定義のアライメント制約を満たしていない場合，ジェネレータはエラーを検出する【IOS081】.

OsHookStackCoreAssignment

パラメータ名	/AUTOSAR/EcucDefs/Os/OsOS/OsHookStack/ OsHookStackCoreAssignment【NOS0952】
概要	SC1，SC2 の場合は，フック用スタックを使用するコア ID. SC3，SC4 の場合は，信頼フック用スタックを使用するコア ID.
型	整数型
値の範囲	0 .. 65534
多重度	0 .. 1
制限事項	マルチコア対応 OS のみで使用可能

3.8.5　OsNonTrustedHookStack

コンテナ名	/AUTOSAR/EcucDefs/Os/OsOS/OsNonTrustedHookStack【NOS0779】
概要	非信頼フック用スタック設定コンテナ
多重度	0 .. *
パラメータ	OsNonTrustedHookStackSize
	OsNonTrustedHookStackStartAddress
	OsNonTrustedHookStackCoreAssignment
サブコンテナ	－
制限事項	SC3，SC4 のみで使用可能. OsNonTrustedHookStackCoreAssignment が同じ値の本コンテナを複数指定した場合，ジェネレータはエラーを検出する【NOS1053】. ATK2 では, 非信頼フック用スタックを使用する場合に本コンテナを省略した場合, ターゲット定義のサイズで非信頼フック用スタックが確保される【IOS088】.

OsNonTrustedHookStackSize

パラメータ名	/AUTOSAR/EcucDefs/Os/OsOS/OsNonTrustedHookStack/ OsNonTrustedHookStackSize【NOS0780】
概要	非信頼フック用スタックのサイズ
型	整数型
値の範囲	－
多重度	1
制限事項	SC3，SC4 のみで使用可能. ATK2 では，OsNonTrustedHookStackStartAddress が指定されている場合，かつ本パラメータで指定された値が，ターゲット定義のアライメント制約を満たしていない場合, ジェネレータはエラーを検出する【IOS076】.

OsNonTrustedHookStackStartAddress

パラメータ名	/AUTOSAR/EcucDefs/Os/OsOS/OsNonTrustedHookStack/ OsNonTrustedHookStackStartAddress【IOS066】
概要	非信頼フック用スタックの先頭番地. 本パラメータを省略した場合，ジェネレータが， OsNonTrustedHookStackSize で指定したサイズのスタック領域を確保する.
型	整数型
値の範囲	－
多重度	0 .. 1
制限事項	SC3，SC4 のみで使用可能. 本パラメータで指定された値が，ターゲット定義のアライメント制約を満たしていない場合，ジェネレータはエラーを検出する【IOS077】.

OsNonTrustedHookStackCoreAssignment

パラメータ名	/AUTOSAR/EcucDefs/Os/OsOS/ OsNonTrustedHookStack/ OsNonTrustedHookStackCoreAssignment【NOS0953】
概要	非信頼フック用スタックを使用するコア ID
型	整数型
値の範囲	0 .. 65534
多重度	0 .. 1
制限事項	SC3，SC4 のマルチコア対応 OS のみで使用可能

3.8.6　OsOsStack

コンテナ名	/AUTOSAR/EcucDefs/Os/OsOS/OsOsStack【IOS052】
概要	SC1，SC2 の場合は，C2ISR 用スタックとフック用スタックを 1 つのスタックで確保するための設定コンテナ. SC3，SC4 のメモリ保護機能の機能レベル 1，2 の場合は，信頼 C2ISR 用スタックと信頼フック用スタックを 1 つのスタックで確保するための設定コンテナ. 本コンテナが省略された場合, C2ISR に対して OsIsrStackSize で指定された値の，割込み優先度毎の最大値を合計した値と，OsHookStackSize で指定された値(省略された場合は 0)の合計値を，ターゲット定義の最小値に足した値のスタックを確保する.
多重度	0 .. *
パラメータ	OsOsStackSize
	OsOsStackStartAddress
	OsOsStackCoreAssignment
サブコンテナ	－
制限事項	OsOsStackCoreAssignment が同じ値の本コンテナを複数指定した場合，ジェネレータはエラーを検出する【NOS1054】. SC3，SC4 のメモリ保護機能の機能レベル 3 の場合に，本コンテナを指定した場合，ジェネレータはエラーを検出する(現状，機能レベル 3 は未実装)【IOS082】.

OsOsStackSize

パラメータ名	/AUTOSAR/EcucDefs/Os/OsOS/OsOsStack/ OsOsStackSize【IOS053】
概要	SC1，SC2 の場合は，C2ISR 用スタックとフック用スタックを 1 つのスタックで確保する場合のサイズ. SC3, SC4 のメモリ保護機能の機能レベル 1, 2 の場合は, 信頼 C2ISR 用スタックと信頼フック用スタックを 1 つのスタックで確保する場合のサイズ.
型	整数型
値の範囲	－
多重度	1
制限事項	SC1，SC2 ，および SC3，SC4 のメモリ保護機能の機能レベル 2 以下のみで使用可能. C2ISR に対して OsIsrStackSize で指定された値の, 割込み優先度毎の最大値を合計した値と，OsHookStackSize で指定された値の合計値を，ターゲット定義の最小値に足した値より，小さい値を指定した場合，ジェネレータはエラーを検出する【IOS054】. OsOsStackStartAddress が指定されている場合，かつ本パラメータで指定された値が, ターゲット定義のアライメント制約を満たしていない場合，ジェネレータはエラーを検出する【IOS070】.

OsOsStackStartAddress

パラメータ名	/AUTOSAR/EcucDefs/Os/OsOS/OsOsStack/ OsOsStackStartAddress【IOS055】
概要	SC1，SC2 の場合は，C2ISR 用スタックとフック用スタックを 1 つのスタックで確保する場合の先頭番地. SC3，SC4 のメモリ保護機能の機能レベル 1，2 の場合は，信頼 C2ISR 用スタックと信頼フック用スタックを 1 つのスタックで確保する場合の先頭番地. 本パラメータを省略した場合，ジェネレータが， OsOsStackSize で指定したサイズのスタック領域を確保する.
型	整数型
値の範囲	－
多重度	0 .. 1
制限事項	SC1，SC2 ，および SC3，SC4 のメモリ保護機能の機能レベル 2 以下のみで使用可能. 本パラメータで指定された値が，ターゲット定義のアライメント制約を満たしていない場合，ジェネレータはエラーを検出する【IOS071】.

OsOsStackCoreAssignment

パラメータ名	/AUTOSAR/EcucDefs/Os/OsOS/ OsOsStack/ OsOsStackCoreAssignment【NOS0954】
概要	SC1，SC2 の場合は，1 つの領域で確保される C2ISR 用スタックとフック用スタックを使用するコア ID. SC3，SC4 のメモリ保護機能の機能レベル 1，2 の場合は，1 つの領域で確保される信頼 C2ISR 用スタックと信頼フック用スタックを使用するコア ID.
型	整数型
値の範囲	0 .. 65534
多重度	0 .. 1
制限事項	マルチコア対応 OS のみで使用可能

AUTOSAR 仕様との違い

ATK2 の SC1, SC2 および SC3, SC4 のメモリ保護機能の機能レベル 2 以下では, 信頼フック, C2ISR が使用するスタックを 1 つの領域で確保して使用するため，このスタックのサイズ，先頭番地を指定するパラメータをそれぞれ追加した≪IOS053≫≪IOS055≫.

3.8.7　OsTimingProtection

コンテナ名	/AUTOSAR/EcucDefs/Os/OsOS/OsTimingProtection【NOS0201】
概要	タイミング保護設定コンテナ
多重度	0 .. 1
パラメータ	OsAllInterruptLockBudget
	OsOsInterruptLockBudget
サブコンテナ	－
制限事項	SC2，SC4 のみで使用可能

OsAllInterruptLockBudget

パラメータ名	/AUTOSAR/EcucDefs/Os/OsOS/OsTimingProtection/ OsAllInterruptLockBudget【NOS0202】
概要	全割込み禁止バジェット(単位：秒)
型	浮動小数点型
値の範囲	0 .. 無限大
多重度	0 .. 1
制限事項	SC2，SC4 においてタイミング保護の機能レベル 2 の場合のみで使用可能

OsOsInterruptLockBudget

パラメータ名	/AUTOSAR/EcucDefs/Os/OsOS/OsTimingProtection/ OsOsInterruptLockBudget【NOS0203】
概要	OS 割込み禁止バジェット(単位：秒)
型	浮動小数点型
値の範囲	0 .. 無限大
多重度	0 .. 1
制限事項	SC2，SC4 においてタイミング保護の機能レベル 2 の場合のみで使用可能

AUTOSAR 仕様との違い

本仕様では，機能レベルを導入することにより，タスク，C2ISR の割込み禁止時間バジェットの設定方法を選択可能とした．タイミング保護の機能レベルが 2 の場合，割込み禁止時間バジェットの設定は本コンテナで行う≪NOS0202≫≪NOS0203≫.

3.8.8　OsAppMode

コンテナ名	/AUTOSAR/EcucDefs/Os/OsAppMode【OS_Conf022】
概要	アプリケーションモードコンテナ
多重度	1 .. *
パラメータ	－
サブコンテナ	－
制限事項	ATK2 では，使用可能なアプリケーションモードの上限数は，32 である≪IOS024≫.

OSEK 仕様との違い

OSEK 仕様では，OSDEFAULTAPPMODE はデフォルトのアプリケーションモードで，StartOS に対して常に有効なパラメータであると規定している【COS3725】. 本仕様では, OSDEFAULTAPPMODE は，必要に応じてユーザが用意するものとして削除した．OSDEFAULTAPPMODE を使用する場合，本コンテナで定義する.

3.8.9　OsTask

コンテナ名	/AUTOSAR/EcucDefs/Os/OsTask【OS_Conf073】
概要	タスクオブジェクトコンテナ
多重度	0 .. *
パラメータ	OsTaskActivation
	OsTaskPriority
	OsTaskSchedule
	OsTaskAccessingApplication
	OsTaskEventRef
	OsTaskResourceRef
	OsTaskStackSize
	OsTaskStackStartAddress
	OsTaskSystemStackSize
	OsTaskSystemStackStartAddress
	OsTaskMutexRef
サブコンテナ	OsTaskAutostart
	OsTaskTimingProtection

OsTaskActivation

パラメータ名	/AUTOSAR/EcucDefs/Os/OsTask/OsTaskActivation 【OS_Conf074】〔COS0434〕
概要	最大起動要求回数
型	整数型
値の範囲	1 .. 256【IOS025】
多重度	1
制限事項	対象のタスクが拡張タスクである場合に，本パラメータに 1 以外を設定すると，ジェネレータはエラーを検出する【NOS0802】.

OsTaskPriority

パラメータ名	/AUTOSAR/EcucDefs/Os/OsTask/OsTaskPriority【OS_Conf075】
概要	タスク優先度
型	整数型
値の範囲	0 .. 15【IOS020】
多重度	1
制限事項	－

OsTaskSchedule

パラメータ名	/AUTOSAR/EcucDefs/Os/OsTask/OsTaskSchedule【OS_Conf076】 〔COS0470〕
概要	タスクのスケジューリングポリシ
型	列挙型
値の範囲	FULL：フルプリエンプティブスケジューリング NON：ノンプリエンプティブスケジューリング
多重度	1
制限事項	本パラメータに NON を設定し，OsTaskResourceRef に内部リソースを指定した場合，ジェネレータはエラーを検出する≪OS_Conf076≫.

OsTaskAccessingApplication

パラメータ名	/AUTOSAR/EcucDefs/Os/OsTask/ OsTaskAccessingApplication【OS_Conf077】
概要	タスクにアクセス可能な OSAP
型	/AUTOSAR/EcucDefs/Os/OsApplication への参照
値の範囲	－
多重度	0 .. *
制限事項	SC3，SC4 のみで使用可能

OsTaskEventRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsTask/ OsTaskEventRef【OS_Conf078】
概要	タスクの持つイベント
型	/AUTOSAR/EcucDefs/Os/OsEvent への参照
値の範囲	－
多重度	0 .. *
制限事項	－

OsTaskResourceRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsTask/ OsTaskResourceRef【OS_Conf079】
概要	タスクが獲得するリソース
型	/AUTOSAR/EcucDefs/Os/OsResource への参照
値の範囲	－
多重度	0 .. *
制限事項	内部リソースは，1 つのみ指定することができる【COS0832】. SC3，SC4 で，本パラメータに内部リソースを指定した場合，かつタスクが所属する OSAP が，指定した内部リソースのアクセス権を持たない場合，ジェネレータはエラーを検出する【NOS0654】. 本パラメータで指定したリソースがリンクリソースの場合，ジェネレータはエラーを検出する【NOS0703】. マルチコア対応 OS において，同じリソースに異なるコアに割付くタスク，C2ISR を関連付けした場合，ジェネレータはエラーを検出する≪OS662≫.

OsTaskStackSize

パラメータ名	/AUTOSAR/EcucDefs/Os/OsTask/OsTaskStackSize【NOS0282】
概要	SC1，SC2 の場合は，タスク用スタックのサイズ． SC3，SC4 の場合は以下となる． 信頼タスクの場合，本パラメータで指定したサイズと，OsTaskSystemStackSize に指定されたサイズとの合計値を，信頼タスク用スタックのサイズとして確保する． 非信頼タスクの場合，非信頼タスク用スタックのサイズとなる．
型	整数型
値の範囲	－
多重度	0 .. 1
制限事項	本パラメータに 0 を指定した場合，ジェネレータはエラーを検出する【NOS0444】． SC1，SC2 もしくは SC3，SC4 における非信頼タスクの場合，本パラメータに，実装定義の最小値より小さい値を指定した場合，ジェネレータはエラーを検出する【NOS0818】． SC3，SC4 における信頼タスクの場合，本パラメータで指定したサイズと，OsTaskSystemStackSize に指定されたサイズとの合計値が，実装定義の最小値より小さい値を指定した場合，ジェネレータはエラーを検出する【NOS0820】． 本パラメータを省略した場合，ターゲット定義のサイズで対象のスタックが確保される【NOS0896】． ATK2 では，スタックサイズの最小値をターゲット定義とする【IOS140】． ATK2 では，OsTaskStackStartAddress が指定されている場合，かつ本パラメータで指定された値が，ターゲット定義のアライメント制約を満たしていない場合，ジェネレータはエラーを検出する【IOS074】．

OsTaskStackStartAddress

パラメータ名	/AUTOSAR/EcucDefs/Os/OsTask/ OsTaskStackStartAddress【IOS047】
概要	SC1，SC2 の場合は，タスク用スタックの先頭番地. SC3，SC4 における信頼タスクの場合は，信頼タスク用スタックの先頭番地．非信頼タスクの場合は，非信頼タスク用スタックの先頭番地. 本パラメータを省略した場合，ジェネレータが， OsTaskStackSize で指定したサイズのスタック領域を確保する.
型	整数型
値の範囲	－
多重度	0 .. 1
制限事項	本パラメータで指定された値が，ターゲット定義のアライメント制約を満たしていない場合，ジェネレータはエラーを検出する【IOS072】.

OsTaskSystemStackSize

パラメータ名	/AUTOSAR/EcucDefs/Os/OsTask/ OsTaskSystemStackSize【NOS0283】
概要	信頼タスクの場合，本パラメータで指定したサイズと，OsTaskStackSize に指定されたサイズとの合計値を，タスクのスタックサイズとして確保する． 非信頼タスクの場合，非信頼タスク用システムスタックのサイズとなる．非信頼タスクで，本パラメータが省略された場合，実装定義の最小値を指定されたものとする．ATK2 では，スタックサイズの最小値をターゲット定義とする≪IOS140≫．
型	整数型
値の範囲	－
多重度	0 .. 1
制限事項	SC3，SC4 のみで使用可能． SC3，SC4 における非信頼タスクの場合で，本パラメータに，実装定義の最小値より小さい値もしくは 0 を指定した場合，ジェネレータはエラーを検出する【NOS0819】． ATK2 では，スタックサイズの最小値をターゲット定義とする≪IOS140≫． ATK2 では，信頼タスクに対して， OsTaskStackStartAddress が指定されているとき，本パラメータを省略しない，もしくは 0 以外を指定した場合，ジェネレータはエラーを検出する【IOS051】． ATK2 では，OsTaskSystemStackStartAddress が指定されている場合，かつ本パラメータで指定された値が，ターゲット定義のアライメント制約を満たしていない場合，ジェネレータはエラーを検出する【IOS075】．

OsTaskSystemStackStartAddress

パラメータ名	/AUTOSAR/EcucDefs/Os/OsTask/ OsTaskSystemStackStartAddress【IOS048】
概要	非信頼タスク用システムスタックのスタック先頭番地. 本パラメータを省略した場合，ジェネレータが， OsTaskSystemStackSize で指定したサイズのスタック領域を確保する.
型	整数型
値の範囲	－
多重度	0 .. 1
制限事項	SC3，SC4 のみで使用可能. 信頼タスクに対して，本パラメータを指定した場合，ジェネレータはエラーを検出する【IOS049】. 非信頼タスクに対して， OsTaskSystemStackSize を指定せずに，本パラメータを指定した場合，ジェネレータはエラーを検出する【IOS050】. 本パラメータで指定された値が，ターゲット定義のアライメント制約を満たしていない場合，ジェネレータはエラーを検出する【IOS073】. 本パラメータで, 非信頼 OSAP からアクセス可能な領域を指定した場合，ジェネレータはエラーを検出する【IOS139】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，タスク用に確保するスタックサイズを指定するパラメータが存在しなかった．そのため，本仕様ではスタックサイズのパラメータを追加した．また，OSAP に対応するために，非信頼 OSAP 用と信頼 OSAP 用のスタックサイズのパラメータをそれぞれ追加した≪NOS0282≫≪NOS0283≫.

ATK2 では，タスクのスタック共有をサポートするため，タスクスタックの先頭番地を指定するパラメータを追加した．また，OSAP に対応するために，非信頼 OSAP 用と信頼 OSAP 用のタスクスタックの先頭番地を指定するパラメータをそれぞれ追加した≪IOS047≫≪IOS048≫.

OsTaskMutexRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsTask/OsTaskMutexRef【NOS0291】
概要	タスクが共有するミューテックス
型	/AUTOSAR/EcucDefs/Os/OsMutex への参照
値の範囲	－
多重度	0 .. *
制限事項	－

3.8.10 OsTaskAutostart

コンテナ名	/AUTOSAR/EcucDefs/Os/OsTask/ OsTaskAutostart【OS_Conf080】〔COS1134〕〔COS1135〕
概要	タスクオブジェクトの自動起動指定コンテナ
多重度	0 .. 1
パラメータ	OsTaskAppModeRef
サブコンテナ	－
制限事項	リスタートタスクに対して，本コンテナが指定された場合，ジェネレータは警告を表示する【NOS1136】.

OsTaskAppModeRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsTask/OsTaskAutostart/ OsTaskAppModeRef【OS_Conf081】
概要	自動起動するアプリケーションモード
型	/AUTOSAR/EcucDefs/Os/OsAppMode への参照
値の範囲	－
多重度	1 .. *
制限事項	－

3.8.11 OsTaskTimingProtection

コンテナ名	/AUTOSAR/EcucDefs/Os/OsTask/ OsTaskTimingProtection【OS_Conf325】
概要	タスクオブジェクトタイミング保護設定コンテナ
多重度	0 .. 1
パラメータ	OsTaskAllInterruptLockMonitor
	OsTaskAllInterruptLockBudget
	OsTaskExecutionBudget
	OsTaskOsInterruptLockMonitor
	OsTaskOsInterruptLockBudget
	OsTaskTimeFrame
	OsTaskResourceLockMonitor
サブコンテナ	OsTaskResourceLock
制限事項	SC2，SC4 のみで使用可能

OsTaskAllInterruptLockMonitor

パラメータ名	/AUTOSAR/EcucDefs/Os/OsTask/OsTaskTimingProtection/ OsTaskAllInterruptLockMonitor【NOS0191】
概要	全割込み禁止時間監視設定
型	ブール型
値の範囲	true：全割込み禁止時間を監視する false：全割込み禁止時間を監視しない
多重度	0 .. 1
制限事項	SC2，SC4 においてタイミング保護の機能レベル 2 の場合のみで使用可能

OsTaskAllInterruptLockBudget

パラメータ名	/AUTOSAR/EcucDefs/Os/OsTask/OsTaskTimingProtection/ OsTaskAllInterruptLockBudget【OS_Conf085】
概要	全割込み禁止時間バジェット(単位：秒)
型	浮動小数点型
値の範囲	0 .. 無限大
多重度	0 .. 1
制限事項	SC2，SC4 においてタイミング保護の機能レベル 3 の場合のみ使用可能

OsTaskExecutionBudget

パラメータ名	/AUTOSAR/EcucDefs/Os/OsTask/OsTaskTimingProtection/ OsTaskExecutionBudget【OS_Conf185】
概要	タスク実行時間バジェット(単位：秒)
型	浮動小数点型
値の範囲	0 .. 無限大
多重度	0 .. 1
制限事項	SC2，SC4 のみで使用可能

OsTaskOsInterruptLockMonitor

パラメータ名	/AUTOSAR/EcucDefs/Os/OsTask/OsTaskTimingProtection/ OsTaskOsInterruptLockMonitor【NOS0192】
概要	OS 割込み禁止監視設定
型	ブール型
値の範囲	true：OS 割込み禁止時間を監視する false：OS 割込み禁止時間を監視しない
多重度	0 .. 1
制限事項	SC2，SC4 においてタイミング保護の機能レベル 2 の場合のみ使用可能

OsTaskOsInterruptLockBudget

パラメータ名	/AUTOSAR/EcucDefs/Os/OsTask/OsTaskTimingProtection/ OsTaskOsInterruptLockBudget【OS_Conf086】
概要	OS 割込み禁止バジェット(単位：秒)
型	浮動小数点型
値の範囲	0 .. 無限大
多重度	0 .. 1
制限事項	SC2，SC4 においてタイミング保護の機能レベル 3 の場合のみ使用可能

OsTaskTimeFrame

パラメータ名	/AUTOSAR/EcucDefs/Os/OsTask/OsTaskTimingProtection/ OsTaskTimeFrame【OS_Conf391】
概要	タスクタイムフレーム(単位：秒)
型	浮動小数点型
値の範囲	0 .. 無限大
多重度	0 .. 1
制限事項	SC2，SC4 のみで使用可能

OsTaskResourceLockMonitor

パラメータ名	/AUTOSAR/EcucDefs/Os/OsTask/OsTaskTimingProtection/ OsTaskResourceLock/ OsTaskResourceLockMonitor【NOS0193】
概要	リソース占有時間監視設定
型	ブール型
値の範囲	true：リソース占有時間を監視する false：リソース占有時間を監視しない
多重度	0 .. 1
制限事項	SC2，SC4 においてタイミング保護の機能レベル 2 の場合のみ使用可能

AUTOSAR 仕様との違い

本仕様では，機能レベルを導入することにより，タスクがリソースを獲得する際のリソース占有時間バジェットの設定方法を選択可能とした．タイミング保護の機能レベルが 2 の場合，リソース占有時間バジェットの設定はリソースオブジェクトで行い，タスクオブジェクトでは獲得するリソースの占有時間の監視を行うかの設定を行う≪NOS0193≫．タイミング保護の機能レベルが 3 の場合，AUTOSAR 仕様と同様の設定方法とした．

また，タスクが割込み禁止状態にする際の割込み禁止バジェットの設定方法も同様である≪NOS0191≫≪NOS0192≫．

3.8.12 OsTaskResourceLock

コンテナ名	/AUTOSAR/EcucDefs/Os/OsTask/OsTaskTimingProtection/ OsTaskResourceLock【OS_Conf082】
概要	タスクのリソース占有時間バジェット設定コンテナ
多重度	0 .. *
パラメータ	OsTaskResourceLockBudget
	OsTaskResourceLockResourceRef
サブコンテナ	－

OsTaskResourceLockBudget

パラメータ名	/AUTOSAR/EcucDefs/Os/OsTask/OsTaskTimingProtection/ OsTaskResourceLock/ OsTaskResourceLockBudget【OS_Conf083】
概要	リソース占有時間バジェット(単位：秒)
型	浮動小数点型
値の範囲	0 .. 無限大
多重度	1
制限事項	SC2，SC4 においてタイミング保護の機能レベル 3 の場合のみ

OsTaskResourceLockResourceRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsTask/OsTaskTimingProtection/ OsTaskResourceLock/ OsTaskResourceLockResourceRef【OS_Conf084】
概要	リソース占有時間バジェットを監視するリソース
型	/AUTOSAR/EcucDefs/Os/OsResource への参照
値の範囲	－
多重度	1
制限事項	SC2，SC4 においてタイミング保護の機能レベル 3 の場合のみ

3.8.13 OsIsr

コンテナ名	/AUTOSAR/EcucDefs/Os/OsIsr【OS_Conf041】
概要	ISR オブジェクトコンテナ
多重度	0 .. *
パラメータ	OsIsrCategory
	OsIsrInterruptNumber
	OsIsrInterruptPriority
	OsIsrInterruptSource
	OsIsrAccessingApplication
	OsIsrResourceRef
	OsIsrStackSize
	OsIsrStackStartAddress
	OsIsrSystemStackSize
	OsIsrSystemStackStartAddress
	OsIsrInterruptTrigger
サブコンテナ	OsIsrTimingProtection

OsIsrCategory

パラメータ名	/AUTOSAR/EcucDefs/Os/OsIsr/OsIsrCategory【OS_Conf042】〔COS0601〕
概要	ISR のカテゴリ
型	列挙型
値の範囲	CATEGORY_1：C1ISR CATEGORY_2：C2ISR
多重度	1
制限事項	SC2，SC4 で本パラメータが CATEGORY_1 の場合，ジェネレータは警告を出力する【OS045】.

OsIsrInterruptNumber

パラメータ名	/AUTOSAR/EcucDefs/Os/OsIsr/ OsIsrInterruptNumber【NOS0663】
概要	ISR が使用する割込み番号
型	整数型
値の範囲	定義なし
多重度	1
制限事項	他の ISR で指定された割込み番号を，本パラメータに指定した場合，ジェネレータはエラーを検出する【NOS0799】. 実装定義の有効値以外を指定した場合，ジェネレータはエラーを検出する【NOS0664】. ATK2 では，割込み番号の有効値はターゲット定義とする【IOS146】.

OsIsrInterruptPriority

パラメータ名	/AUTOSAR/EcucDefs/Os/OsIsr/ OsIsrInterruptPriority【NOS0665】
概要	ISR の割込み優先度
型	整数型
値の範囲	定義なし
多重度	1
制限事項	C2ISR に指定した割込み優先度以下の割込み優先度を，C1ISR に指定した場合，ジェネレータはエラーを検出する【NOS0666】. 実装定義の有効値以外を指定した場合，ジェネレータはエラーを検出する【NOS0856】. ATK2 では，割込み優先度の有効値はターゲット定義とする【IOS147】.

OsIsrInterruptSource

パラメータ名	/AUTOSAR/EcucDefs/Os/OsIsr/ OsIsrInterruptSource【NOS0667】
概要	ISR の割込み要因の初期状態
型	列挙型
値の範囲	ENABLE：有効 DISABLE：無効
多重度	0 .. 1
制限事項	SC3，SC4 のみで使用可能. OsIsrCategory に CATEGORY_1 を指定した場合に，本パラメータに DISABLE を指定した場合，ジェネレータはエラーを検出する【NOS0668】. SC3，SC4 で本パラメータを省略した場合，ジェネレータはエラーを検出する【NOS0891】.

OsIsrAccessingApplication

パラメータ名	/AUTOSAR/EcucDefs/Os/OsIsr/ OsIsrAccessingApplication【NOS0694】
概要	ISR にアクセス可能な OSAP
型	/AUTOSAR/EcucDefs/Os/OsApplication への参照
値の範囲	－
多重度	0 .. *
制限事項	SC3，SC4 のみで使用可能. OsIsrCategory に CATEGORY_1 を指定した場合に，本パラメータを指定した場合，ジェネレータはエラーを検出する【NOS0708】.

OsIsrResourceRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsIsr/OsIsrResourceRef【OS_Conf043】
概要	ISR が獲得するリソース
型	/AUTOSAR/EcucDefs/Os/OsResource への参照
値の範囲	－
多重度	0 .. *
制限事項	本パラメータに，内部リソースを指定した場合，ジェネレータはエラーを検出する【COS0831】〔NOS0718〕. OsIsrCategory に CATEGORY_1 を指定した場合に，本パラメータを指定した場合，ジェネレータはエラーを検出する【NOS0687】. 本パラメータで指定したリソースがリンクリソースの場合，ジェネレータはエラーを検出する【NOS0704】. マルチコア対応 OS において，同じリソースに異なるコアに割付くタスク，C2ISR を関連付けした場合，ジェネレータはエラーを検出する≪OS662≫.

OsIsrStackSize

パラメータ名	/AUTOSAR/EcucDefs/Os/OsIsr/OsIsrStackSize【NOS0284】
概要	SC1，SC2 の場合は，C2ISR 用スタックのサイズ. SC3，SC4 の場合は以下となる. 信頼 C2ISR の場合，本パラメータで指定したサイズと，OsIsrSystemStackSize に指定されたサイズとの合計値を，信頼 C2ISR スタックのサイズとして確保する．非信頼 C2ISR の場合，非信頼 C2ISR 用スタックのサイズとなる.
型	整数型
値の範囲	－
多重度	0 .. 1
制限事項	本パラメータに 0 を指定した場合，ジェネレータはエラーを検出する【NOS0442】. 対象の ISR が，C1ISR である場合に，本パラメータを指定すると，ジェネレータはエラーを検出する【NOS0743】. 対象の ISR が，C2ISR である場合に，本パラメータを省略した場合，ターゲット定義のサイズで対象のスタックが確保される【NOS0745】.

OsIsrStackStartAddress

パラメータ名	/AUTOSAR/EcucDefs/Os/OsIsr/ OsIsrStackStartAddress【IOS056】
概要	信頼 C2ISR の場合，信頼 C2ISR 用スタックの先頭番地．非信頼 C2ISR の場合，非信頼 C2ISR 用スタックの先頭番地． 本パラメータを省略した場合，ジェネレータが，OsIsrStackSize で指定したサイズのスタック領域を確保する．
型	整数型
値の範囲	－
多重度	0 .. 1
制限事項	SC3，SC4 のメモリ保護機能の機能レベル 3 のみで使用可能(現状，機能レベル 3 は未実装)． 本パラメータで指定された値が，ターゲット定義のアライメント制約を満たしていない場合，ジェネレータはエラーを検出する【IOS211】． 対象の ISR が，C1ISR である場合に，本パラメータを指定すると，ジェネレータはエラーを検出する【IOS064】．

OsIsrSystemStackSize

パラメータ名	/AUTOSAR/EcucDefs/Os/OsIsr/ OsIsrSystemStackSize【NOS0285】
概要	信頼 C2ISR の場合，本パラメータで指定したサイズと，OsIsrStackSize に指定されたサイズとの合計値を，信頼 C2ISR 用スタックのサイズとして確保する． 非信頼 C2ISR の場合，非信頼 C2ISR 用システムスタックのサイズとなる．非信頼 C2ISR で，本パラメータが省略された場合，実装定義の最小値を指定されたものとする．ATK2 では，スタックサイズの最小値をターゲット定義とする≪IOS140≫．
型	整数型
値の範囲	－
多重度	0 .. 1
制限事項	SC3，SC4 のメモリ保護機能の機能レベル 3 のみで使用可能． 対象の ISR が，C1ISR である場合に，本パラメータを指定すると，ジェネレータはエラーを検出する【NOS0744】．

OsIsrSystemStackStartAddress

パラメータ名	/AUTOSAR/EcucDefs/Os/OsIsr/ OsIsrSystemStartAddress【IOS057】
概要	非信頼 C2ISR 用システムスタックの先頭番地. 本パラメータを省略した場合，ジェネレータが，OsIsrSystemStackSize で指定したサイズのスタック領域を確保する.
型	整数型
値の範囲	－
多重度	0 .. 1
制限事項	SC3，SC4 のメモリ保護機能の機能レベル 3 のみで使用可能(現状，機能レベル 3 は未実装). 信頼 C2ISR に対して，本パラメータを指定した場合，ジェネレータはエラーを検出する【IOS058】. 非信頼 C2ISR に対して，OsIsrSystemStackSize を指定せずに，本パラメータを指定した場合，ジェネレータはエラーを検出する【IOS059】. 本パラメータで指定された値が，ターゲット定義のアライメント制約を満たしていない場合，ジェネレータはエラーを検出する【IOS212】. 対象の ISR が，C1ISR である場合に，本パラメータを指定すると，ジェネレータはエラーを検出する【IOS065】.

OsIsrInterruptTrigger

パラメータ名	/AUTOSAR/EcucDefs/Os/OsIsr/ OsIsrInterruptTrigger【NOS0777】
概要	割込み要因のトリガ設定. 本パラメータの使用可否は実装定義である．ATK2 では，使用可否をターゲット定義とする【IOS148】.
型	－
値の範囲	－
多重度	0 .. 1
制限事項	－

AUTOSAR 仕様との違い

AUTOSAR 仕様では, C2ISR 用に確保するスタックサイズを指定するパラメータが存在しなかった.

そのため，本仕様ではスタックサイズのパラメータを追加した．また，OSAP に対応するために，非信頼 OSAP 用と信頼 OSAP 用の C2ISR スタックのサイズをそれぞれ追加した≪NOS0284≫≪NOS0285≫.

また，ターゲットシステムによっては，割込み要因に対する属性を指定できる場合があるので，予約するパラメータを規定した≪NOS0777≫.

ATK2 の SC1，SC2 では，C2ISR が使用するスタックを 1 つの領域で確保して使用するが，SC3，SC4 では，メモリ保護を実現するために C2ISR 毎に使用するスタック領域を分ける必要があるので，C2ISR が使用するスタックの先頭番地を指定するパラメータを追加した．また，OSAP に対応するために，非信頼 OSAP 用と信頼 OSAP 用の C2ISR が使用するスタックの先頭番地を指定するパラメータをそれぞれ追加した≪IOS056≫≪IOS057≫.

3.8.14 OsIsrTimingProtection

コンテナ名	/AUTOSAR/EcucDefs/Os/OsIsr/OsIsrTimingProtection【OS_Conf326】
概要	ISR オブジェクトタイミング保護設定コンテナ
多重度	0 .. 1
パラメータ	OsIsrAllInterruptLockMonitor
	OsIsrAllInterruptLockBudget
	OsIsrExecutionBudget
	OsIsrOsInterruptLockMonitor
	OsIsrOsInterruptLockBudget
	OsIsrTimeFrame
	OsIsrResourceLockMonitor
サブコンテナ	OsIsrResourceLock
制限事項	SC2，SC4 のみで使用可能

OsIsrAllInterruptLockMonitor

パラメータ名	/AUTOSAR/EcucDefs/Os/OsIsr/OsIsrTimingProtection/ OsIsrAllInterruptLockMonitor【NOS0198】
概要	全割込み禁止時間監視設定
型	ブール型
値の範囲	true：全割込み禁止時間を監視する false：全割込み禁止時間を監視しない
多重度	0 .. 1
制限事項	SC2，SC4 においてタイミング保護の機能レベル 2 の場合のみで使用可能

OsIsrAllInterruptLockBudget

パラメータ名	/AUTOSAR/EcucDefs/Os/OsIsr/OsIsrTimingProtection/ OsIsrAllInterruptLockBudget【OS_Conf229】
概要	全割込み禁止時間バジェット(単位：秒)
型	浮動小数点型
値の範囲	0 .. 無限大
多重度	0 .. 1
制限事項	SC2，SC4 においてタイミング保護の機能レベル 3 の場合のみで使用可能

OsIsrExecutionBudget

パラメータ名	/AUTOSAR/EcucDefs/Os/OsIsr/OsIsrTimingProtection/ OsIsrExecutionBudget【OS_Conf222】
概要	C2ISR 実行時間バジェット(単位：秒)
型	浮動小数点型
値の範囲	0 .. 無限大
多重度	0 .. 1
制限事項	SC2，SC4 のみで使用可能

OsIsrOsInterruptLockMonitor

パラメータ名	/AUTOSAR/EcucDefs/Os/OsIsr/OsIsrTimingProtection/ OsIsrOsInterruptLockMonitor【NOS0199】
概要	OS 割込み禁止時間監視設定
型	ブール型
値の範囲	true：OS 割込み禁止時間を監視する false：OS 割込み禁止時間を監視しない
多重度	0 .. 1
制限事項	SC2，SC4 においてタイミング保護の機能レベル 2 の場合のみで使用可能

OsIsrOsInterruptLockBudget

パラメータ名	/AUTOSAR/EcucDefs/Os/OsIsr/OsIsrTimingProtection/ OsIsrOsInterruptLockBudget【OS_Conf387】
概要	OS 割込み禁止バジェット(単位：秒)
型	浮動小数点型
値の範囲	0 .. 無限大
多重度	0 .. 1
制限事項	SC2，SC4 においてタイミング保護の機能レベル 3 の場合のみで使用可能

OsIsrTimeFrame

パラメータ名	/AUTOSAR/EcucDefs/Os/OsIsr/OsIsrTimingProtection/ OsIsrTimeFrame【OS_Conf223】
概要	ISR タイムフレーム(単位：秒)
型	浮動小数点型
値の範囲	0 .. 無限大
多重度	0 .. 1
制限事項	SC2，SC4 のみで使用可能

OsIsrResourceLockMonitor

パラメータ名	/AUTOSAR/EcucDefs/Os/OsIsr/OsIsrTimingProtection/ OsIsrResourceLockMonitor【NOS0200】
概要	リソース占有時間監視設定
型	ブール型
値の範囲	true：リソース占有時間を監視する false：リソース占有時間を監視しない
多重度	0 .. 1
制限事項	SC2，SC4 においてタイミング保護の機能レベル 2 の場合のみで使用可能

3.8.15 OsIsrResourceLock

コンテナ名	/AUTOSAR/EcucDefs/Os/OsIsr/OsIsrTimingProtection/ OsIsrResourceLock【OS_Conf388】
概要	ISR のリソース占有時間バジェット設定コンテナ
多重度	0 .. *
パラメータ	OsIsrResourceLockBudget
	OsIsrResourceLockResourceRef
サブコンテナ	－
制限事項	SC2，SC4 のみで使用可能

OsIsrResourceLockBudget

パラメータ名	/AUTOSAR/EcucDefs/Os/OsIsr/OsIsrTimingProtection/ OsIsrResourceLock/ OsIsrResourceLockBudget【OS_Conf389】
概要	リソース占有時間バジェット(単位：秒)
型	浮動小数点型
値の範囲	0 .. 無限大
多重度	1
制限事項	SC2，SC4 においてタイミング保護の機能レベル 3 の場合のみ

OsIsrResourceLockResourceRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsIsr/OsIsrTimingProtection/ OsIsrResourceLock/ OsIsrResourceLockResourceRef【OS_Conf390】
概要	リソース占有時間バジェットを監視するリソース
型	/AUTOSAR/EcucDefs/Os/OsResource への参照
値の範囲	－
多重度	1
制限事項	SC2，SC4 においてタイミング保護の機能レベル 3 の場合のみ

AUTOSAR 仕様との違い

本仕様では，機能レベルを導入することにより，C2ISR がリソースを獲得する際のリソース占有時間バジェットの設定方法を選択可能とした．タイミング保護の機能レベルが 2 の場合，リソース占有時間バジェットの設定はリソースオブジェクトで行い，割込みオブジェクトでは獲得するリソースの占有時間の監視を行うかの設定を行う≪NOS0200≫．タイミング保護の機能レベルが 3 の場合，AUTOSAR 仕様と同様の設定方法とした．

また，C2ISR が割込み禁止状態にする際の割込み禁止バジェットの設定方法も同様である≪NOS0198≫≪NOS0199≫.

3.8.16 OsEvent

コンテナ名	/AUTOSAR/EcucDefs/Os/OsEvent【OS_Conf033】
概要	イベントオブジェクトコンテナ
多重度	0 .. *
パラメータ	OsEventMask
サブコンテナ	－

OsEventMask

パラメータ名	/AUTOSAR/EcucDefs/Os/OsEvent/OsEventMask【OS_Conf034】
概要	イベントマスク値
型	整数型
値の範囲	1 .. 0xffffffff【IOS019】
多重度	0 .. 1
制限事項	本パラメータを省略した場合，イベントマスク値は自動的に設定される≪OS_Conf034≫. 本パラメータに 0 を指定した場合，ジェネレータはエラーを検出する【NOS0800】. 本パラメータを省略した場合に，設定可能なビットマスクが存在しない場合，ジェネレータはエラーを検出する【NOS0801】.

3.8.17 OsResource

コンテナ名	/AUTOSAR/EcucDefs/Os/OsResource【OS_Conf252】
概要	リソースオブジェクトコンテナ
多重度	0 .. *
パラメータ	OsResourceProperty
	OsResourceAccessingApplication
	OsResourceLinkedResourceRef
サブコンテナ	OsResourceTimingProtection
制限事項	ATK2 では，いずれの OsTaskResourceRef，OsIsrResourceRef にも指定されていないリソースが存在する場合，ジェネレータは警告を出力し，該当リソースの生成は行わない【IOS036】.

OsResourceProperty

パラメータ名	/AUTOSAR/EcucDefs/Os/OsResource/ OsResourceProperty【OS_Conf050】
概要	リソース種別
型	列挙型
値の範囲	STANDARD：標準リソース INTERNAL：内部リソース LINKED：リンクリソース
多重度	1
制限事項	本パラメータに LINKED を指定した場合に， OsResourceLinkedResourceRef 以外のパラメータ，サブコンテナを指定すると，ジェネレータはエラーを検出する【NOS0705】.

OsResourceAccessingApplication

パラメータ名	/AUTOSAR/EcucDefs/Os/OsResource/ OsResourceAccessingApplication【OS_Conf051】
概要	リソースにアクセス可能な OSAP
型	/AUTOSAR/EcucDefs/Os/OsApplication への参照
値の範囲	－
多重度	0 .. *
制限事項	SC3，SC4 のみで使用可能

OsResourceLinkedResourceRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsResource/ OsResourceLinkedResourceRef【OS_Conf052】 〔COS0839〕
概要	リンクリソースにおけるリンク先リソース
型	/AUTOSAR/EcucDefs/Os/OsResource への参照
値の範囲	－
多重度	0 .. 1
制限事項	OsResourceProperty が LINKED 以外で，本パラメータを指定した場合，ジェネレータは無視する≪OS_Conf052≫. OsResourceProperty が LINKED の場合は，本パラメータを省略すると，ジェネレータはエラーを検出する【NOS0784】. 本パラメータで指定するリソースが，標準リソースでない場合，ジェネレータはエラーを検出する【NOS0707】.

3.8.18 OsResourceTimingProtection

コンテナ名	/AUTOSAR/EcucDefs/Os/OsResource/ OsResourceTimingProtection【NOS0207】
概要	リソースオブジェクトタイミング保護設定コンテナ
多重度	0 .. 1
パラメータ	OsResourceLockBudget
サブコンテナ	－
制限事項	SC2，SC4 のみで使用可能

OsResourceLockBudget

パラメータ名	/AUTOSAR/EcucDefs/Os/OsResource/OsResourceTimingProtection/ OsResourceLockBudget【NOS0208】
概要	リソース占有時間バジェット(単位：秒)
型	浮動小数点型
値の範囲	0 .. 無限大
多重度	0 .. 1
制限事項	SC2，SC4 においてタイミング保護の機能レベル 2 の場合のみ

AUTOSAR 仕様との違い

本仕様では，機能レベルを導入することにより，タスク，C2ISR のリソース占有時間バジェットの設定方法を選択可能とした．タイミング保護の機能レベルが 2 の場合，リソース占有時間バジェットの設

定は本コンテナで行う≪NOS0208≫.

3.8.19 OsCounter

コンテナ名	/AUTOSAR/EcucDefs/Os/OsCounter【OS_Conf026】〔COS0927〕
概要	カウンタオブジェクトコンテナ
多重度	0 .. *
パラメータ	OsCounterMaxAllowedValue
	OsCounterTicksPerBase
	OsCounterMinCycle
	OsCounterType
	OsSecondsPerTick
	OsCounterIsrRef
	OsCounterAccessingApplication
サブコンテナ	OsDriver
	OsTimeConstant

OsCounterMaxAllowedValue

パラメータ名	/AUTOSAR/EcucDefs/Os/OsCounter/ OsCounterMaxAllowedValue【OS_Conf027】
概要	カウンタのティックの最大値
型	整数型
値の範囲	1 .. 0x7fffffff【IOS152】
多重度	1
制限事項	－

OsCounterTicksPerBase

パラメータ名	/AUTOSAR/EcucDefs/Os/OsCounter/ OsCounterTicksPerBase【OS_Conf029】
概要	カウンタ固有の値(OS は不使用)
型	整数型
値の範囲	1 .. 0x7fffffff【IOS156】
多重度	1
制限事項	－

OsCounterMinCycle

パラメータ名	/AUTOSAR/EcucDefs/Os/OsCounter/ OsCounterMinCycle【OS_Conf028】
概要	接続されたアラームがカウンタに指定できる最小周期値
型	整数型
値の範囲	1 .. 0x7fffffff【IOS153】
多重度	1
制限事項	本パラメータに，OsCounterMaxAllowedValue より大きい値を設定すると，ジェネレータはエラーを検出する【NOS0786】.

OsCounterType

パラメータ名	/AUTOSAR/EcucDefs/Os/OsCounter/ OsCounterType【OS_Conf255】
概要	カウンタのタイプ
型	列挙型
値の範囲	HARDWARE：ハードウェアカウンタ SOFTWARE：ソフトウェアカウンタ
多重度	1
制限事項	－

OsSecondsPerTick

パラメータ名	/AUTOSAR/EcucDefs/Os/OsCounter/ OsSecondsPerTick【OS_Conf030】〔OSa010〕
概要	ハードウェアカウンタにおける 1 ティック当たりの実時間(単位：秒) ジェネレータは，ユーザが実時間を計算するために，変換マクロを生成する【OS393】. 生成されるマクロの形式は， OS_TICKS2<Unit>_<Counter>(ticks) である. <Unit>には NS, US, MS, SEC が入り, 4 種類のマクロを出力する. <Counter>は本コンテナを含むカウンタの名称である.
型	浮動小数点型
値の範囲	0 .. 無限大
多重度	0 .. 1
制限事項	－

OsCounterIsrRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsCounter/ OsCounterIsrRef【NOS0688】〔NOS0673〕
概要	ハードウェアカウンタの満了処理を行う C2ISR
型	/AUTOSAR/EcucDefs/Os/OsIsr への参照
値の範囲	－
多重度	0 .. 1
制限事項	参照する ISR が，C1ISR であった場合，ジェネレータはエラーを検出する【NOS0689】. OsCounterType が HARDWARE の場合は，本パラメータを省略すると，ジェネレータはエラーを検出する【NOS0785】〔NOS0672〕. SC3，SC4 で，参照する C2ISR が，ハードウェアカウンタと異なる OSAP に所属している場合，ジェネレータはエラーを検出する【NOS0722】.

OsCounterAccessingApplication

パラメータ名	/AUTOSAR/EcucDefs/Os/OsCounter/ OsCounterAccessingApplication【OS_Conf031】
概要	カウンタにアクセス可能な OSAP
型	/AUTOSAR/EcucDefs/Os/OsApplication への参照
値の範囲	－
多重度	0 .. *
制限事項	SC3，SC4 のみで使用可能

3.8.20 OsDriver

AUTOSAR 仕様に記載がないため，本仕様においても記載しない.

3.8.21 OsTimeConstant

AUTOSAR 仕様では，OsTimeConstant コンテナが規定されている【OS_Conf386】．本コンテナの OsTimeValue パラメータで，1 ティック当たりの時間を秒単位の定数で定義するように規定している【OS_Conf002】．しかし，OsSecondsPerTick で同様の値を定義するため，本仕様では，本コンテナを規定しない.

3.8.22 OsAlarm

コンテナ名	/AUTOSAR/EcucDefs/Os/OsAlarm【OS_Conf003】
概要	アラームオブジェクトコンテナ
多重度	0 .. *
パラメータ	OsAlarmAccessingApplication
	OsAlarmCounterRef
サブコンテナ	OsAlarmAction
	OsAlarmAutostart

OsAlarmAccessingApplication

パラメータ名	/AUTOSAR/EcucDefs/Os/OsAlarm/ OsAlarmAccessingApplication【OS_Conf004】
概要	アラームにアクセス可能な OSAP
型	/AUTOSAR/EcucDefs/Os/OsApplication への参照
値の範囲	－
多重度	0 .. *
制限事項	SC3，SC4 のみで使用可能

OsAlarmCounterRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsAlarm/ OsAlarmCounterRef【OS_Conf005】〔COS0903〕 〔COS0914〕〔COS0915〕〔COS0927〕
概要	アラームに接続するカウンタ
型	/AUTOSAR/EcucDefs/Os/OsCounter への参照
値の範囲	－
多重度	1
制限事項	マルチコア対応 OS において，アラームと異なるコアに割付くカウンタを本パラメータに指定した場合，ジェネレータはエラーを検出する【OS663】. 保護違反時処理の機能レベル 2 で，本パラメータに他の非信頼 OSAP に所属するカウンタを指定した場合，ジェネレータはエラーを検出する≪NOS1129≫. 保護違反時処理の機能レベル 3 で，本パラメータに他の OSAP に所属するカウンタを指定した場合，ジェネレータはエラーを検出する≪NOS1130≫.

3.8.23 OsAlarmAction

コンテナ名	/AUTOSAR/EcucDefs/Os/OsAlarm/OsAlarmAction【OS_Conf006】〔COS0908〕
概要	アラームアクション指定
多重度	1
パラメータ	－
サブコンテナ	OsAlarmActivateTask
	OsAlarmCallback
	OsAlarmIncrementCounter
	OsAlarmSetEvent
制限事項	SC1 以外で OsAlarmCallback がサブコンテナに含まれる場合，ジェネレータはエラーを検出する【OS461】. サブコンテナが 1 つでない場合，ジェネレータはエラーを検出する【NOS0789】.

3.8.24 OsAlarmActivateTask

コンテナ名	/AUTOSAR/EcucDefs/Os/OsAlarm/ OsAlarmAction/OsAlarmActivateTask【OS_Conf007】 〔COS0916〕
概要	アラームアクションのタスク起動指定
多重度	0 .. 1
パラメータ	OsAlarmActivateTaskRef
サブコンテナ	－

OsAlarmActivateTaskRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsAlarm/OsAlarmAction/OsAlarmActivateTask /OsAlarmActivateTaskRef【OS_Conf008】
概要	アラームのアクションで起動するタスク
型	/AUTOSAR/EcucDefs/Os/OsTask への参照
値の範囲	－
多重度	1
制限事項	SC3，SC4 で，アラームが所属する OSAP が参照先タスクのアクセス権を持たない場合，ジェネレータはエラーを検出する【OS344】.

3.8.25 OsAlarmCallback

コンテナ名	/AUTOSAR/EcucDefs/Os/OsAlarm/ OsAlarmAction/OsAlarmCallback【OS_Conf014】 〔COS0916〕
概要	アラームアクションのアラームコールバック指定
多重度	0 .. 1
パラメータ	OsAlarmCallbackName
サブコンテナ	－

OsAlarmCallbackName

パラメータ名	/AUTOSAR/EcucDefs/Os/OsAlarm/OsAlarmAction/OsAlarmCallback /OsAlarmCallbackName【OS_Conf087】
概要	アラームのアクションで呼び出すアラームコールバックの名称
型	関数名
値の範囲	－
多重度	1
制限事項	－

3.8.26 OsAlarmIncrementCounter

コンテナ名	/AUTOSAR/EcucDefs/Os/OsAlarm/OsAlarmAction/ OsAlarmIncrementCounter【OS_Conf302】〔COS0916〕
概要	アラームアクションのカウンタインクリメント指定
多重度	0 .. 1
パラメータ	OsAlarmIncrementCounterRef
サブコンテナ	－

OsAlarmIncrementCounterRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsAlarm/OsAlarmAction/OsAlarmIncrementCounter /OsAlarmIncrementCounterRef【OS_Conf015】
概要	アラームのアクションでインクリメントするカウンタ
型	/AUTOSAR/EcucDefs/Os/OsCounter への参照
値の範囲	－
多重度	1
制限事項	本パラメータが直接もしくは間接的に本アラームの接続されているカウンタを指定した場合，ジェネレータはエラーを検出する【NOS0350】. 本パラメータで指定するカウンタが，ハードウェアカウンタである場合，ジェネレータはエラーを検出する【NOS0788】. SC3，SC4 で，アラームが所属する OSAP が参照先カウンタのアクセス権を持たない場合，ジェネレータはエラーを検出する【NOS0659】. マルチコア対応 OS において，アラームと異なるコアに割付くカウンタを本パラメータに指定した場合，ジェネレータはエラーを検出する【OS664】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，本アラームのアクションでインクリメントするカウンタが，直接もしくは間接的に本アラームの接続されているカウンタを指定した場合，ジェネレータは警告を出力すると記述され

ている【OS303】．本仕様では，アラームのアクションで直接もしくは間接的に，アラームの接続されているカウンタにインクリメント処理を行うと，エラーとなる≪NOS0058≫．したがって，ジェネレータは，警告ではなく，エラーを検出することにした．

3.8.27 OsAlarmSetEvent

コンテナ名	/AUTOSAR/EcucDefs/Os/OsAlarm/OsAlarmAction/ OsAlarmSetEvent【OS_Conf016】〔COS0916〕
概要	アラームアクションのイベントセット指定
多重度	0 .. 1
パラメータ	OsAlarmSetEventRef
	OsAlarmSetEventTaskRef
サブコンテナ	－

OsAlarmSetEventRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsAlarm/OsAlarmAction/ OsAlarmSetEvent/OsAlarmSetEventRef【OS_Conf017】
概要	アラームのアクションでセットするイベント
型	/AUTOSAR/EcucDefs/Os/OsEvent への参照
値の範囲	－
多重度	1
制限事項	－

OsAlarmSetEventTaskRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsAlarm/OsAlarmAction/ OsAlarmSetEvent/OsAlarmSetEventTaskRef【OS_Conf018】
概要	アラームのアクションでイベントをセットするタスク
型	/AUTOSAR/EcucDefs/Os/OsTask への参照
値の範囲	－
多重度	1
制限事項	本パラメータで指定するタスクが，基本タスクである場合，ジェネレータはエラーを検出する【NOS0787】. SC3，SC4 で，アラームが所属する OSAP が参照先タスクのアクセス権を持たない場合，ジェネレータはエラーを検出する≪OS344≫.

3.8.28 OsAlarmAutostart

コンテナ名	/AUTOSAR/EcucDefs/Os/OsAlarm/OsAlarmAutostart【OS_Conf009】〔COS1135〕
概要	アラーム自動起動設定
多重度	0 .. 1
パラメータ	OsAlarmAlarmTime
	OsAlarmAppModeRef
	OsAlarmCycleTime
	OsAlarmAutostartType
サブコンテナ	－

OsAlarmAlarmTime

パラメータ名	/AUTOSAR/EcucDefs/Os/OsAlarm/OsAlarmAutostart/ OsAlarmAlarmTime【OS_Conf010】
概要	アラーム自動起動時の初回満了時刻
型	整数型
値の範囲	0 .. 0x7fffffff【IOS154】
多重度	1
制限事項	OsAlarmAutostartType が RELATIVE の場合，値が 0 であってはならない≪OS_Conf010≫. 本パラメータに，アラームが接続されたカウンタの最大値(OsCounterMaxAllowedValue)より大きい値を指定すると，ジェネレータはエラーを検出する【NOS0790】.

OsAlarmAppModeRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsAlarm/OsAlarmAutostart/ OsAlarmAppModeRef【OS_Conf013】
概要	自動起動するアプリケーションモード
型	/AUTOSAR/EcucDefs/Os/OsAppMode への参照
値の範囲	－
多重度	1 .. *
制限事項	－

OsAlarmCycleTime

パラメータ名	/AUTOSAR/EcucDefs/Os/OsAlarm/OsAlarmAutostart/ OsAlarmCycleTime【OS_Conf012】〔COS0912〕
概要	アラーム自動起動時の周期時間，0 の場合は単発アラームとなる
型	整数型
値の範囲	0 .. 0x7fffffff【IOS155】
多重度	1
制限事項	本パラメータに，アラームが接続されたカウンタの最大値(OsCounterMaxAllowedValue)より大きい値を指定すると，ジェネレータはエラーを検出する【NOS0845】. 本パラメータに，アラームが接続されたカウンタの最小周期(OsCounterMinCycle)より小さい値を指定すると，ジェネレータはエラーを検出する【NOS0846】.

OsAlarmAutostartType

パラメータ名	/AUTOSAR/EcucDefs/Os/OsAlarm/OsAlarmAutostart/ OsAlarmAutostartType【OS_Conf011】
概要	アラームの動作種別
型	列挙型
値の範囲	ABSOLUTE：絶対アラームとして自動起動する RELATIVE：相対アラームとして自動起動する
多重度	1
制限事項	－

3.8.29 OsScheduleTable

コンテナ名	/AUTOSAR/EcucDefs/Os/OsScheduleTable【OS_Conf141】
概要	スケジュールテーブルオブジェクトコンテナ
多重度	0 .. *
パラメータ	OsScheduleTableDuration
	OsScheduleTableRepeating
	OsSchTblAccessingApplication
	OsScheduleTableCounterRef
サブコンテナ	OsScheduleTableAutostart
	OsScheduleTableExpiryPoint
	OsScheduleTableSync

OsScheduleTableDuration

パラメータ名	/AUTOSAR/EcucDefs/Os/OsScheduleTable/ OsScheduleTableDuration【OS_Conf053】
概要	スケジュールテーブル周期(ティック数)
型	整数型
値の範囲	0 .. 0xffffffff【IOS157】
多重度	1
制限事項	OsScheduleTblSyncStrategy に IMPLICIT を指定し，本パラメータに，駆動カウンタのカウンタ最大値(OsCounterMaxAllowedValue)+1 以外を指定した場合，ジェネレータはエラーを出力する【OS440】.

OsScheduleTableRepeating

パラメータ名	/AUTOSAR/EcucDefs/Os/OsScheduleTable/ OsScheduleTableRepeating【OS_Conf144】
概要	スケジュールテーブルの繰り返し指定
型	ブール型
値の範囲	true：周期動作 false：単発動作
多重度	1
制限事項	－

OsSchTblAccessingApplication

パラメータ名	/AUTOSAR/EcucDefs/Os/OsScheduleTable/ OsSchTblAccessingApplication【OS_Conf054】
概要	スケジュールテーブルにアクセス可能な OSAP
型	/AUTOSAR/EcucDefs/Os/OsApplication への参照
値の範囲	－
多重度	0 .. *
制限事項	SC3，SC4 のみで使用可能

OsScheduleTableCounterRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsScheduleTable/ OsScheduleTableCounterRef【OS_Conf145】〔COS0914〕〔COS0927〕
概要	スケジュールテーブルに接続するカウンタ
型	/AUTOSAR/EcucDefs/Os/OsCounter への参照
値の範囲	－
多重度	1
制限事項	マルチコア対応 OS において，スケジュールテーブルと異なるコアに割付くカウンタを本パラメータに指定した場合，ジェネレータはエラーを検出する【OS665】. 保護違反時処理の機能レベル 2 で，本パラメータに他の非信頼 OSAP に所属するカウンタを指定した場合，ジェネレータはエラーを検出する≪NOS1131≫. 保護違反時処理の機能レベル 3 で，本パラメータに他の OSAP に所属するカウンタを指定した場合，ジェネレータはエラーを検出する≪NOS1132≫.

3.8.30 OsScheduleTableAutostart

コンテナ名	/AUTOSAR/EcucDefs/Os/OsScheduleTable/ OsScheduleTableAutostart【OS_Conf335】〔COS1135〕
概要	スケジュールテーブル自動起動設定コンテナ
多重度	0 .. 1
パラメータ	OsScheduleTableStartValue
	OsScheduleTableAutostartType
	OsScheduleTableAppModeRef
サブコンテナ	－

OsScheduleTableStartValue

パラメータ名	/AUTOSAR/EcucDefs/Os/OsScheduleTable/ OsScheduleTableAutostart/ OsScheduleTableStartValue【OS_Conf057】
概要	OsScheduleTableAutostartType が ABSOLUTE の場合，自動起動するスケジュールテーブルに対して設定するティックの絶対値として使用される． OsScheduleTableAutostartType が RELATIVE の場合，自動起動するスケジュールテーブルに対して設定するティックの相対値として使用される．
型	整数型
値の範囲	0 .. 0x7fffffff【IOS158】
多重度	0 .. 1
制限事項	OsScheduleTableAutostartType が ABSOLUTE か RELATIVE の場合に，本パラメータを省略すると，ジェネレータはエラーを検出する【NOS0782】． OsScheduleTableAutostartType が RELATIVE の場合に，本パラメータに 0 を設定すると，ジェネレータはエラーを検出する【NOS0783】． 本パラメータに，スケジュールテーブルが接続されたカウンタの最大値(OsCounterMaxAllowedValue)より大きい値を指定すると，ジェネレータはエラーを検出する【NOS0849】．

OsScheduleTableAutostartType

パラメータ名	/AUTOSAR/EcucDefs/Os/OsScheduleTable/OsScheduleTableAutostart/OsScheduleTableAutostartType【OS_Conf056】
概要	自動起動種別の指定
型	列挙型
値の範囲	ABSOLUTE：ティックの絶対値で自動起動する RELATIVE：ティックの相対値で自動起動する SYNCHRON：同期待ち状態で自動起動する
多重度	1
制限事項	対象のスケジュールテーブルが, 明示同期スケジュールテーブルでない場合に，本パラメータに SYNCHRON を指定すると，ジェネレータはエラーを検出する【NOS0741】. 対象のスケジュールテーブルが, 暗黙同期スケジュールテーブルである場合に，本パラメータに RELATIVE を指定すると，ジェネレータはエラーを検出する【NOS0742】.

OsScheduleTableAppModeRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsScheduleTable/OsScheduleTableAppModeRef【OS_Conf058】
概要	自動起動するアプリケーションモード
型	/AUTOSAR/EcucDefs/Os/OsAppMode への参照
値の範囲	－
多重度	1 .. *
制限事項	－

3.8.31 OsScheduleTableExpiryPoint

コンテナ名	/AUTOSAR/EcucDefs/Os/OsScheduleTable/ OsScheduleTableExpiryPoint【OS_Conf143】〔NOS0884〕
概要	スケジュールテーブル満了点設定コンテナ
多重度	1 .. *
パラメータ	OsScheduleTblExpPointOffset
サブコンテナ	OsScheduleTableEventSetting
	OsScheduleTableTaskActivation
	OsScheduleTableAdjustableExpPoint
制限事項	サブコンテナに OsScheduleTableEventSetting もしくは OsScheduleTableTaskActivation が 1 つも指定されていない場合，ジェネレータはエラーを検出する【NOS0852】. ATK2 では，1 つのスケジュールテーブルに対する満了点の上限数は，254 である≪IOS093≫.

OsScheduleTblExpPointOffset

パラメータ名	/AUTOSAR/EcucDefs/Os/OsScheduleTable/OsScheduleTableExpiryPoint/OsScheduleTblExpPointOffset【OS_Conf062】
概要	スケジュールテーブル満了点オフセット(スケジュールテーブル先頭からのティック数)
型	整数型
値の範囲	－
多重度	1
制限事項	初期オフセットを設定する場合，本パラメータに，0，もしくはスケジュールテーブルが接続されたカウンタの最小周期(OsCounterMinCycle)以上，カウンタの最大値(OsCounterMaxAllowedValue)以下ではない値を指定すると，ジェネレータはエラーを検出する≪OS443≫. 本パラメータによって，遅延が設定される場合，遅延が，スケジュールテーブルが接続されたカウンタの最小周期(OsCounterMinCycle)以上，カウンタの最大値(OsCounterMaxAllowedValue)以下ではなくなる値を指定すると，ジェネレータはエラーを検出する≪OS408≫. 同様に，最終遅延で不正な値が設定された場合も，ジェネレータはエラーを検出する≪OS444≫. ただし，単発動作指定時は最終遅延が 0 以上，カウンタの最大値(OsCounterMaxAllowedValue)以下でも設定可能である≪OS427≫. 本パラメータに，スケジュールテーブル周期(OsScheduleTableDuration)より大きい値を指定すると，ジェネレータはエラーを検出する【NOS0847】. 同一スケジュールテーブルの他の満了点で設定した値を，本パラメータに指定した場合，ジェネレータはエラーを検出する【NOS0851】.

3.8.32 OsScheduleTableEventSetting

コンテナ名	/AUTOSAR/EcucDefs/Os/OsScheduleTable/OsScheduleTableExpiryPoint/OsScheduleTableEventSetting【OS_Conf059】
概要	スケジュールテーブル満了点セットするイベントの設定コンテナ
多重度	0 .. *
パラメータ	OsScheduleTableSetEventRef
	OsScheduleTableSetEventTaskRef
サブコンテナ	－

OsScheduleTableSetEventRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsScheduleTable/OsScheduleTableExpiryPoint/ OsScheduleTableEventSetting/ OsScheduleTableSetEventRef【OS_Conf060】
概要	満了点アクションでセットするイベント
型	/AUTOSAR/EcucDefs/Os/OsEvent への参照
値の範囲	－
多重度	1
制限事項	－

OsScheduleTableSetEventTaskRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsScheduleTable/OsScheduleTableExpiryPoint/ OsScheduleTableEventSetting/ OsScheduleTableSetEventTaskRef【OS_Conf061】
概要	満了点アクションでイベントをセットするタスク
型	/AUTOSAR/EcucDefs/Os/OsTask への参照
値の範囲	－
多重度	1
制限事項	本パラメータで指定するタスクが，基本タスクである場合，ジェネレータはエラーを検出する【NOS0781】. SC3，SC4 で，スケジュールテーブルが所属する OSAP が参照先タスクのアクセス権を持たない場合，ジェネレータはエラーを検出する≪OS343≫.

3.8.33 OsScheduleTableTaskActivation

コンテナ名	/AUTOSAR/EcucDefs/Os/OsScheduleTable/OsScheduleTableExpiryPoint/ OsScheduleTableTaskActivation【OS_Conf066】
概要	スケジュールテーブル満了点で起動するタスクの設定コンテナ
多重度	0 .. *
パラメータ	OsScheduleTableActivateTaskRef
サブコンテナ	－

OsScheduleTableActivateTaskRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsScheduleTable/OsScheduleTableExpiryPoint/OsScheduleTableTaskActivation/OsScheduleTableActivateTaskRef【OS_Conf067】
概要	満了点アクションで起動するタスク
型	/AUTOSAR/EcucDefs/Os/OsTask への参照
値の範囲	－
多重度	1
制限事項	SC3，SC4 でスケジュールテーブルが所属する OSAP が参照先タスクのアクセス権を持たない場合，ジェネレータはエラーを検出する【OS343】.

3.8.34 OsScheduleTblAdjustableExpPoint

コンテナ名	/AUTOSAR/EcucDefs/Os/OsScheduleTable/OsScheduleTableExpiryPoint/OsScheduleTblAdjustableExpPoint【OS_Conf068】
概要	スケジュールテーブル満了点同期設定コンテナ
多重度	0 .. 1
パラメータ	OsScheduleTableMaxLengthen
	OsScheduleTableMaxShorten
サブコンテナ	－
制限事項	SC2，SC4 のみで使用可能

OsScheduleTableMaxLengthen

パラメータ名	/AUTOSAR/EcucDefs/Os/OsScheduleTable/OsScheduleTableExpiryPoint/OsScheduleTblAdjustableExpPoint/OsScheduleTableMaxLengthen【OS_Conf069】
概要	同期補正で満了点の遅延を伸ばしてもよい最大値(ティック数)
型	整数型
値の範囲	－
多重度	1
制限事項	－

OsScheduleTableMaxShorten

パラメータ名	/AUTOSAR/EcucDefs/Os/OsScheduleTable/OsScheduleTableExpiryPoint/ OsScheduleTblAdjustableExpPoint/ OsScheduleTableMaxShorten【OS_Conf070】
概要	同期補正で満了点の遅延を縮めてもよい最大値(ティック数)
型	整数型
値の範囲	－
多重度	1
制限事項	－

3.8.35 OsScheduleTableSync

コンテナ名	/AUTOSAR/EcucDefs/Os/OsScheduleTable/ OsScheduleTableSync【OS_Conf063】
概要	スケジュールテーブル同期設定コンテナ. 本コンテナを省略した場合，同期なしスケジュールテーブルとなる≪OS_Conf065≫.
多重度	0 .. 1
パラメータ	OsScheduleTblExplicitPrecision
	OsScheduleTblSyncStrategy
サブコンテナ	－

OsScheduleTblExplicitPrecision

パラメータ名	/AUTOSAR/EcucDefs/Os/OsScheduleTable/OsScheduleTableSync/ OsScheduleTblExplicitPrecision【OS_Conf064】
概要	スケジュールテーブルを同期状態とみなす最大差分(ティック数)
型	整数型
値の範囲	－
多重度	0 .. 1
制限事項	－

OsScheduleTblSyncStrategy

パラメータ名	/AUTOSAR/EcucDefs/Os/OsScheduleTable/OsScheduleTableSync/OsScheduleTblSyncStrategy【OS_Conf065】
概要	スケジュールテーブルの同期方式
型	列挙型
値の範囲	EXPLICIT：明示同期 IMPLICIT：暗黙同期 NONE：同期なし(デフォルト)
多重度	1
制限事項	－

3.8.36 OsApplication

コンテナ名	/AUTOSAR/EcucDefs/Os/OsApplication【OS_Conf114】
概要	OSAP オブジェクトコンテナ
多重度	0 .. *
パラメータ	OsTrusted
	OsAppAlarmRef
	OsAppCounterRef
	OsAppEcucPartitionRef
	OsAppIsrRef
	OsAppScheduleTableRef
	OsAppTaskRef
	OsRestartTask
	OsAppMemorySectionRef
	OsAppMemoryModuleRef
	OsAppMemoryAreaRef
	OsApplicationCoreAssignment
	OsAppInterCoreInterruptRef
	OsAppStandardMemoryRomRegionRef
	OsAppStandardMemoryRamRegionRef
サブコンテナ	OsApplicationHooks
	OsApplicationTrustedFunction
制限事項	SC3，SC4 もしくはマルチコア対応 OS のみで使用可能

OsTrusted

パラメータ名	/AUTOSAR/EcucDefs/Os/OsApplication/ OsTrusted【OS_Conf115】
概要	信頼 OSAP か非信頼 OSAP かを表す
型	ブール型
値の範囲	true：信頼 OSAP false：非信頼 OSAP
多重度	1
制限事項	SC3，SC4 のみで使用可能

OsAppAlarmRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsApplication/ OsAppAlarmRef【OS_Conf231】
概要	OSAP に所属するアラーム
型	/AUTOSAR/EcucDefs/Os/OsAlarm への参照
値の範囲	－
多重度	0 .. *
制限事項	SC3，SC4 もしくはマルチコア対応 OS のみで使用可能. 他の OSAP に所属するアラームを，本パラメータに指定した場合，ジェネレータはエラーを検出する【NOS0792】. いずれの OSAP にも所属しないアラームが存在する場合，ジェネレータはエラーを検出する≪OS311≫.

OsAppCounterRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsApplication/ OsAppCounterRef【OS_Conf234】
概要	OSAP に所属するカウンタ
型	/AUTOSAR/EcucDefs/Os/OsCounter への参照
値の範囲	－
多重度	0 .. *
制限事項	SC3，SC4 もしくはマルチコア対応 OS のみで使用可能. 他の OSAP に所属するカウンタを，本パラメータに指定した場合，ジェネレータはエラーを検出する【NOS0793】. いずれの OSAP にも所属しないカウンタが存在する場合，ジェネレータはエラーを検出する≪OS311≫. 本パラメータで指定するカウンタにアラームが接続されている場合で，そのアラームが所属する OSAP が割付くコアと，OsApplicationCoreAssignment で指定するコアが異なる場合，ジェネレータはエラーを検出する≪OS631≫. 本パラメータで指定するカウンタにスケジュールテーブルが接続されている場合で，そのスケジュールテーブルが所属する OSAP が割付くコアと，OsApplicationCoreAssignment で指定するコアが異なる場合，ジェネレータはエラーを検出する≪OS630≫.

OsAppEcucPartitionRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsApplication/ OsAppEcucPartitionRef【OS_Conf392】
概要	OSAP に対応する EcucPartition
型	/AUTOSAR/EcucDefs/EcuC/EcucPartitionCollection/EcucPartition への参照
値の範囲	－
多重度	0 .. 1
制限事項	OS では，本パラメータを使用しない.

OsAppIsrRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsApplication/ OsAppIsrRef【OS_Conf221】
概要	OSAP に所属する ISR
型	/AUTOSAR/EcucDefs/Os/OsIsr への参照
値の範囲	－
多重度	0 .. *
制限事項	SC3，SC4 もしくはマルチコア対応 OS のみで使用可能. 他の OSAP に所属する ISR を，本パラメータに指定した場合，ジェネレータはエラーを検出する【NOS0794】. OsTrusted に false が指定されている場合に，本パラメータに C1ISR を指定すると，ジェネレータはエラーを検出する≪OS361≫. いずれの OSAP にも所属しない ISR が存在する場合，ジェネレータはエラーを検出する≪OS311≫. SC3，SC4 のメモリ保護機能の機能レベル 1，2 で，OsTrusted に false が指定されている場合に，本パラメータに C2ISR を指定すると，ジェネレータはエラーを検出する【NOS0855】.

OsAppScheduleTableRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsApplication/ OsAppScheduleTableRef【OS_Conf230】
概要	OSAP に所属するスケジュールテーブル
型	/AUTOSAR/EcucDefs/Os/OsScheduleTable への参照
値の範囲	－
多重度	0 .. *
制限事項	SC3，SC4 もしくはマルチコア対応 OS のみで使用可能. 他の OSAP に所属するスケジュールテーブルを，本パラメータに指定した場合，ジェネレータはエラーを検出する【NOS0796】. いずれの OSAP にも所属しないスケジュールテーブルが存在する場合，ジェネレータはエラーを検出する≪OS311≫.

OsAppTaskRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsApplication/ OsAppTaskRef【OS_Conf116】
概要	OSAP に所属するタスク
型	/AUTOSAR/EcucDefs/Os/OsTask への参照
値の範囲	－
多重度	0 .. *
制限事項	SC3，SC4 もしくはマルチコア対応 OS のみで使用可能. 他の OSAP に所属するタスクを，本パラメータに指定した場合，ジェネレータはエラーを検出する【NOS0797】. いずれの OSAP にも所属しないタスクが存在する場合，ジェネレータはエラーを検出する≪OS311≫.

OsRestartTask

パラメータ名	/AUTOSAR/EcucDefs/Os/OsApplication/OsRestartTask【OS_Conf120】
概要	OSAP 再起動時に起動するタスク (本パラメータを指定しない場合，リスタートタスクの起動は行われない)
型	/AUTOSAR/EcucDefs/Os/OsTask への参照
値の範囲	－
多重度	0 .. 1
制限事項	SC3，SC4 の保護違反時処理機能の機能レベル 2，3 で使用可能. 本パラメータに，異なる OSAP に所属するタスクを指定した場合，ジェネレータはエラーを検出する【NOS1135】. ATK2 では，本パラメータに指定したタスクが，OsAppTaskRef に指定されていなくても，該当 OSAP に所属されるものとして扱う【IOS231】.

OsAppMemorySectionRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsApplication/ OsAppMemorySectionRef【NOS0176】
概要	メモリセクション(OsMemorySection)の ID
型	/AUTOSAR/EcucDefs/Os/OsMemorySection への参照
値の範囲	－
多重度	0 .. *
制限事項	SC3，SC4 のみで使用可能. 他の OSAP に所属するメモリセクション(OsMemorySection)の ID を，本パラメータに指定した場合，ジェネレータはエラーを検出する【NOS0899】.

OsAppMemoryModuleRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsApplication/ OsAppMemoryModuleRef【NOS0177】
概要	メモリセクション情報(オブジェクトファイル単位指定)(OsMemoryModule)の ID
型	/AUTOSAR/EcucDefs/Os/OsMemoryModule への参照
値の範囲	－
多重度	0 .. *
制限事項	SC3，SC4 のみで使用可能. 他の OSAP に所属するメモリセクション情報(オブジェクトファイル単位指定)(OsMemoryModule)の ID を，本パラメータに指定した場合，ジェネレータはエラーを検出する【NOS0900】.

OSAppMemoryAreaRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsApplication/ OsAppMemoryAreaRef【NOS0178】
概要	メモリ領域(OsMemoryArea)の ID
型	/AUTOSAR/EcucDefs/Os/OsMemoryArea への参照
値の範囲	－
多重度	0 .. *
制限事項	SC3，SC4 のみで使用可能. 他の OSAP に所属するメモリ領域(OsMemoryArea)の ID を，本パラメータに指定した場合，ジェネレータはエラーを検出する【NOS0901】.

OsApplicationCoreAssignment

パラメータ名	/AUTOSAR/EcucDefs/Os/OsApplication/ OsApplicationCoreAssignment【MCOS_Conf1020】
概要	OSAP を割付けるコア ID
型	整数型
値の範囲	0 .. 65534
多重度	0 .. 1
制限事項	マルチコア対応 OS のみで使用可能. マルチコアシステム上で識別できない不正なコア ID を本パラメータに指定した場合，ジェネレータはエラーを検出する【OS667】.

OsAppInterCoreInterruptRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsApplication/ OsAppInterCoreInterruptRef【NOS1111】
概要	OSAP に所属する ICISR
型	/AUTOSAR/EcucDefs/Os/OsInterCoreInterrupt への参照
値の範囲	－
多重度	0 .. *
制限事項	マルチコア対応 OS のみで使用可能. 他の OSAP に所属する ICISR を，本パラメータに指定した場合，ジェネレータはエラーを検出する【NOS1112】. いずれの OSAP にも所属しない ICISR が存在する場合，ジェネレータはエラーを検出する【NOS1113】. SC3，SC4 のメモリ保護機能の機能レベル 1，2 で，OsTrusted に false が指定されている場合に，本パラメータを指定すると，ジェネレータはエラーを検出する【NOS1114】.

OsAppStandardMemoryRomRegionRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsApplication / OsAppStandardMemoryRomRegionRef【NOS1057】
概要	OSAP の標準 ROM リージョンの指定. 本パラメータを省略した場合，OsStandardMemoryRegion で指定した標準 ROM リージョンが使用される.
型	/AUTOSAR/EcucDefs/Os/OsMemoryRegion への参照
値の範囲	－
多重度	0 .. 1
制限事項	SC3，SC4 のみで使用可能. 本パラメータで指定したメモリリージョンの OsMemoryRegionWriteable が true である場合，ジェネレータはエラーを検出する【NOS1058】.

OsAppStandardMemoryRamRegionRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsApplication / OsAppStandardMemoryRamRegionRef【NOS1059】
概要	OSAP の標準 RAM リージョンの指定. 本パラメータを省略した場合，OsStandardMemoryRegion で指定した標準 RAM リージョンが使用される.
型	/AUTOSAR/EcucDefs/Os/OsMemoryRegion への参照
値の範囲	－
多重度	0 .. 1
制限事項	SC3，SC4 のみで使用可能. 本パラメータで指定したメモリリージョンの OsMemoryRegionWriteable が false である場合，ジェネレータはエラーを検出する【NOS1060】.

3.8.37 OsApplicationHooks

コンテナ名	/AUTOSAR/EcucDefs/Os/OsApplication/ OsApplicationHooks【OS_Conf020】
概要	OSAP 固有フックコンテナ
多重度	1
パラメータ	OsAppErrorHook
	OsAppShutdownHook
	OsAppStartupHook
サブコンテナ	－
制限事項	SC3，SC4 もしくはマルチコア対応 OS のみで使用可能

OsAppErrorHook

パラメータ名	/AUTOSAR/EcucDefs/Os/OsApplication/OsApplicationHooks/ OsAppErrorHook【OS_Conf213】
概要	OSAP 固有のエラーフック呼出し有無
型	ブール型
値の範囲	true：OSAP 固有のエラーフック有効 false：OSAP 固有のエラーフック無効
多重度	1
制限事項	SC3，SC4 もしくはマルチコア対応 OS のみで使用可能

OsAppShutdownHook

パラメータ名	/AUTOSAR/EcucDefs/Os/OsApplication/OsApplicationHooks/ OsAppShutdownHook【OS_Conf125】
概要	OSAP 固有のシャットダウンフック呼出し有無
型	ブール型
値の範囲	true：OSAP 固有のシャットダウンフック有効 false：OSAP 固有のシャットダウンフック無効
多重度	1
制限事項	SC3，SC4 もしくはマルチコア対応 OS のみで使用可能

OsAppStartupHook

パラメータ名	/AUTOSAR/EcucDefs/Os/OsApplication/OsApplicationHooks/ OsAppStartupHook【OS_Conf124】
概要	OSAP 固有のスタートアップフック呼出し有無
型	ブール型
値の範囲	true：OSAP 固有のスタートアップフック有効 false：OSAP 固有のスタートアップフック無効
多重度	1
制限事項	SC3，SC4 もしくはマルチコア対応 OS のみで使用可能

3.8.38 OsApplicationTrustedFunction

コンテナ名	/AUTOSAR/EcucDefs/Os/OsApplication/ OsApplicationTrustedFunction【OS_Conf021】
概要	OSAP 信頼関数定義
多重度	0 .. *
パラメータ	OsTrustedFunctionName
	OsTrustedFunctionStackSize
サブコンテナ	－
制限事項	SC3，SC4 のみで使用可能. 非信頼 OSAP に対して，本コンテナを指定した場合，ジェネレータはエラーを検出する【NOS0791】.

OsTrustedFunctionName

パラメータ名	/AUTOSAR/EcucDefs/Os/OsApplication/OsApplicationTrustedFunction/OsTrustedFunctionName【OS_Conf254】
概要	OSAP 信頼関数名
型	関数名型
値の範囲	－
多重度	1
制限事項	SC3，SC4 のみで使用可能. 本パラメータに"TRUSTED_"から始まらない関数名を指定した場合，ジェネレータはエラーを検出する【NOS0823】.

OsTrustedFunctionStackSize

パラメータ名	/AUTOSAR/EcucDefs/Os/OsApplication/OsApplicationTrustedFunction/OsTrustedFunctionStackSize【NOS0721】
概要	OSAP 信頼関数が使用するスタックサイズ
型	整数型
値の範囲	－
多重度	1
制限事項	SC3，SC4 のみで使用可能

3.8.39 OsMemoryRegion

コンテナ名	/AUTOSAR/EcucDefs/Os/OsMemoryRegion【NOS0693】
概要	メモリリージョン情報コンテナ
多重度	0 .. *
パラメータ	OsMemoryRegionName
	OsMemoryRegionWriteable
	OsMemoryRegionSize
	OsMemoryRegionStartAddress
制限事項	SC3，SC4 のみで使用可能. 標準 ROM リージョンと標準 RAM リージョンが，1 つずつ定義されていない場合，ジェネレータはエラーを検出する【NOS0747】.

OsMemoryRegionName

パラメータ名	/AUTOSAR/EcucDefs/Os/OsMemoryRegion/ OsMemoryRegionName【NOS0746】
概要	メモリリージョン名称
型	文字列型
値の範囲	－
多重度	1
制限事項	SC3，SC4 のみで使用可能. 他のメモリリージョンで指定されたメモリリージョン名称を，本パラメータに指定した場合，ジェネレータはエラーを検出する【NOS0803】.

OsMemoryRegionWriteable

パラメータ名	/AUTOSAR/EcucDefs/Os/OsMemoryRegion/ OsMemoryRegionWriteable【NOS0749】
概要	メモリリージョンに対する書込みアクセス設定
型	ブール型
値の範囲	true：書込み可能 false：書込み不可
多重度	1
制限事項	SC3，SC4 のみで使用可能

OsMemoryRegionSize

パラメータ名	/AUTOSAR/EcucDefs/Os/OsMemoryRegion/ OsMemoryRegionSize【NOS0751】
概要	メモリリージョンサイズ
型	整数型
値の範囲	－
多重度	1
制限事項	SC3，SC4 のみで使用可能. 本パラメータに 0 を指定すると，ジェネレータはエラーを検出する【NOS0814】. 本パラメータで指定された値が，実装定義のアライメント制約を満たしていない場合，ジェネレータはエラーを検出する【NOS1160】. 本パラメータで指定された領域が，他のメモリ領域と重なる場合，ジェネレータはエラーを検出する【NOS1161】.

OsMemoryRegionStartAddress

パラメータ名	/AUTOSAR/EcucDefs/Os/OsMemoryRegion/ OsMemoryRegionStartAddress【NOS0750】
概要	メモリリージョン開始アドレス
型	整数型
値の範囲	－
多重度	1
制限事項	SC3，SC4 のみで使用可能. 本パラメータで指定された値が，実装定義のアライメント制約を満たしていない場合，ジェネレータはエラーを検出する【NOS0804】. ATK2 では，アライメント制約はターゲット定義とする【IOS141】. 本パラメータと OsMemoryRegionSize とで定義されるメモリリージョンが，他のメモリリージョンと重なる場合，ジェネレータはエラーを検出する【NOS0805】.

3.8.40 OsMemorySection

コンテナ名	/AUTOSAR/EcucDefs/Os/OsMemorySection【NOS0301】
概要	メモリセクション情報コンテナ. OsAppMemorySectionRef に指定された場合は，OSAP の専有領域，指定されなかった場合は共有領域となる.
多重度	0 .. *
パラメータ	OsMemorySectionName
	OsMemorySectionMemoryRegionRef
	OsMemorySectionWriteable
	OsMemorySectionReadable
	OsMemorySectionExecutable
	OsMemorySectionShort
	OsMemorySectionInitialize
	OsMemorySectionCacheable
	OsMemorySectionDevice
	OsMemorySectionExport
制限事項	SC3，SC4 のみで使用可能

OsMemorySectionName

パラメータ名	/AUTOSAR/EcucDefs/Os/OsMemorySection/ OsMemorySectionName【NOS0760】
概要	セクション名称
型	文字列型
値の範囲	－
多重度	1
制限事項	SC3，SC4 のみで使用可能. 他のメモリセクション情報で指定されたセクション名称を，本パラメータに指定した場合，ジェネレータはエラーを検出する【NOS0809】. 本パラメータに，標準のセクションの名称を指定した場合，ジェネレータはエラーを検出する【NOS0810】.

OsMemorySectionMemoryRegionRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsMemorySection/ OsMemorySectionMemoryRegionRef【NOS0761】
概要	セクションを配置するメモリリージョン
型	/AUTOSAR/EcucDefs/Os/OsMemoryRegion への参照
値の範囲	－
多重度	1
制限事項	SC3，SC4 のみで使用可能

OsMemorySectionWriteable

パラメータ名	/AUTOSAR/EcucDefs/Os/OsMemorySection/ OsMemorySectionWriteable【NOS0762】
概要	メモリセクションに対する書込みアクセス設定
型	ブール型
値の範囲	true：書込み可能 false：書込み不可
多重度	1
制限事項	SC3，SC4 のみで使用可能

OsMemorySectionReadable

パラメータ名	/AUTOSAR/EcucDefs/Os/OsMemorySection/ OsMemorySectionReadable【NOS0763】
概要	メモリセクションに対する読込みアクセス設定
型	ブール型
値の範囲	true：読込み可能 false：読込み不可
多重度	1
制限事項	SC3，SC4 のみで使用可能

OsMemorySectionExecutable

パラメータ名	/AUTOSAR/EcucDefs/Os/OsMemorySection/ OsMemorySectionExecutable【NOS0764】
概要	メモリセクションに対する実行アクセス設定
型	ブール型
値の範囲	true：実行可能 false：実行不可
多重度	1
制限事項	SC3，SC4 のみで使用可能

OsMemorySectionShort

パラメータ名	/AUTOSAR/EcucDefs/Os/OsMemorySection/ OsMemorySectionShort【NOS0765】
概要	メモリセクションに対するショートデータ領域設定
型	ブール型
値の範囲	true：ショートデータ領域に配置する false：ショートデータ領域に配置しない
多重度	1
制限事項	SC3，SC4 のみで使用可能

OsMemorySectionInitialize

パラメータ名	/AUTOSAR/EcucDefs/Os/OsMemorySection/ OsMemorySectionInitialize【NOS0766】
概要	メモリセクションに対するメモリ初期化設定
型	列挙型
値の範囲	DATA：初期化する BSS：ゼロクリアする NO_INITIALIZE：初期化しない
多重度	0 .. 1
制限事項	SC3，SC4 のみで使用可能. 書込み不可のメモリリージョンもしくは，書込み不可のメモリセクションに対して，本パラメータを指定した場合，ジェネレータはエラーを検出する【NOS0858】.

OsMemorySectionCacheable

パラメータ名	/AUTOSAR/EcucDefs/Os/OsMemorySection/ OsMemorySectionCacheable【NOS0767】
概要	メモリセクションに対するキャッシュ設定
型	ブール型
値の範囲	true：キャッシュ可能 false：キャッシュ不可
多重度	1
制限事項	SC3，SC4 のみで使用可能

OsMemorySectionDevice

パラメータ名	/AUTOSAR/EcucDefs/Os/OsMemorySection/ OsMemorySectionDevice【NOS0768】
概要	メモリセクションに対するデバイス領域設定
型	ブール型
値の範囲	true：デバイス領域である false：デバイス領域でない
多重度	1
制限事項	SC3，SC4 のみで使用可能

OsMemorySectionExport

パラメータ名	/AUTOSAR/EcucDefs/Os/OsMemorySection/ OsMemorySectionExport【NOS0769】
概要	メモリセクションに対する専有ライト共有リード領域の設定
型	ブール型
値の範囲	true：専有ライト共有リード false：専有リードライト
多重度	1
制限事項	SC3，SC4 のみで使用可能. OsMemorySectionWritable が true，かつ OsMemorySectionReadable が true，かつ OsMemorySectionExecutable が false の場合以外に，本パラメータに true を指定すると，ジェネレータはエラーを検出する【NOS0770】.

3.8.41 OsMemoryModule

コンテナ名	/AUTOSAR/EcucDefs/Os/OsMemoryModule【NOS0302】
概要	メモリセクション情報(オブジェクトファイル単位指定)コンテナ. OsAppMemoryModuleRef に指定された場合は, OSAP の専有領域となり，指定されなかった場合はカーネル専有領域となる．更に, OsMemoryModuleExport に true を指定した場合は，共有リード領域となる.
多重度	0 .. *
パラメータ	OsMemoryModuleName
	OsMemoryModuleExport
制限事項	SC3，SC4 のみで使用可能. 本コンテナの使用可否は実装定義である．ATK2 では，使用可否をターゲット定義とする【IOS180】.

OsMemoryModuleName

パラメータ名	/AUTOSAR/EcucDefs/Os/OsMemoryModule/ OsMemoryModuleName【NOS0772】
概要	オブジェクトファイル名称
型	文字列型
値の範囲	－
多重度	1
制限事項	SC3，SC4 のみで使用可能. 他のメモリセクション情報(オブジェクトファイル単位指定)で指定されたオブジェクトファイル名称を，本パラメータに指定した場合，ジェネレータはエラーを検出する【NOS0808】.

OsMemoryModuleExport

パラメータ名	/AUTOSAR/EcucDefs/Os/OsMemoryModule/ OsMemoryModuleExport【NOS0773】
概要	オブジェクトファイルに対する専有ライト共有リード領域の設定． 本パラメータに true を指定した場合，リードオンリーセクションは共有リードオンリーになり，実行セクションは共有実行になる．
型	ブール型
値の範囲	true：専有ライト共有リード false：専有リードライト
多重度	1
制限事項	SC3，SC4 のみで使用可能

3.8.42 OsMemoryArea

コンテナ名	/AUTOSAR/EcucDefs/Os/OsMemoryArea【NOS0303】
概要	アクセスを許可するメモリ領域の設定コンテナ． OsAppMemoryAreaRef に指定された場合は，OSAP の専有領域，指定されなかった場合は共有領域となる．
多重度	0 .. *
パラメータ	OsMemoryAreaWriteable
	OsMemoryAreaReadable
	OsMemoryAreaExecutable
	OsMemoryAreaCacheable
	OsMemoryAreaDevice
	OsMemoryAreaSize
	OsMemoryAreaStartAddress
制限事項	SC3，SC4 のみで使用可能

OsMemoryAreaWriteable

パラメータ名	/AUTOSAR/EcucDefs/Os/OsMemoryArea/ OsMemoryAreaWriteable【NOS0752】
概要	アクセスを許可するメモリ領域に対する書込みアクセス設定
型	ブール型
値の範囲	true：書込み可能 false：書込み不可
多重度	1
制限事項	SC3，SC4 のみで使用可能

OsMemoryAreaReadable

パラメータ名	/AUTOSAR/EcucDefs/Os/OsMemoryArea/ OsMemoryAreaReadable【NOS0753】
概要	アクセスを許可するメモリ領域に対する読込みアクセス設定
型	ブール型
値の範囲	true：読込み可能 false：読込み不可
多重度	1
制限事項	SC3，SC4 のみで使用可能

OsMemoryAreaExecutable

パラメータ名	/AUTOSAR/EcucDefs/Os/OsMemoryArea/ OsMemoryAreaExecutable【NOS0754】
概要	アクセスを許可するメモリ領域に対する実行アクセス設定
型	ブール型
値の範囲	true：実行可能 false：実行不可
多重度	1
制限事項	SC3，SC4 のみで使用可能

OsMemoryAreaCacheable

パラメータ名	/AUTOSAR/EcucDefs/Os/OsMemoryArea/ OsMemoryAreaCacheable【NOS0755】
概要	アクセスを許可するメモリ領域に対するキャッシュ設定
型	ブール型
値の範囲	true：キャッシュ可能 false：キャッシュ不可
多重度	1
制限事項	SC3，SC4 のみで使用可能

OsMemoryAreaDevice

パラメータ名	/AUTOSAR/EcucDefs/Os/OsMemoryArea/ OsMemoryAreaDevice【NOS0756】
概要	アクセスを許可するメモリ領域に対するデバイス領域設定
型	ブール型
値の範囲	true：デバイス領域である false：デバイス領域でない
多重度	1
制限事項	SC3，SC4 のみで使用可能

OsMemoryAreaSize

パラメータ名	/AUTOSAR/EcucDefs/Os/OsMemoryArea/ OsMemoryAreaSize【NOS0758】
概要	アクセスを許可するメモリ領域のサイズ
型	整数型
値の範囲	－
多重度	1
制限事項	SC3，SC4 のみで使用可能. 本パラメータに 0 を指定すると，ジェネレータはエラーを検出する【NOS0815】. 本パラメータで指定された値が，実装定義のアライメント制約を満たしていない場合，ジェネレータはエラーを検出する【NOS0853】. ATK2 では，アライメント制約はターゲット定義とする≪IOS141≫. 本パラメータで指定された領域が，他のメモリ領域と重なる場合，ジェネレータはエラーを検出する【NOS0854】.

OsMemoryAreaStartAddress

パラメータ名	/AUTOSAR/EcucDefs/Os/OsMemoryArea/ OsMemoryAreaStartAddress【NOS0757】
概要	アクセスを許可するメモリ領域の先頭アドレス
型	整数型
値の範囲	－
多重度	1
制限事項	SC3，SC4 のみで使用可能. 本パラメータで指定された値が，実装定義のアライメント制約を満たしていない場合，ジェネレータはエラーを検出する【NOS0813】. ATK2 では，アライメント制約はターゲット定義とする≪IOS141≫.

3.8.43 OsLinkSection

コンテナ名	/AUTOSAR/EcucDefs/Os/OsLinkSection【NOS0774】
概要	非信頼タスクのスタックを配置するメモリセクション情報コンテナ
多重度	0 .. *
パラメータ	OsLinkSectionName
	OsLinkSectionMemoryRegionRef
制限事項	SC3，SC4 のみで使用可能

OsLinkSectionName

パラメータ名	/AUTOSAR/EcucDefs/Os/OsLinkSection/ OsLinkSectionName【NOS0775】
概要	セクション名称
型	文字列型
値の範囲	－
多重度	1
制限事項	SC3，SC4 のみで使用可能. 他のメモリセクション情報で指定されたセクション名称を，本パラメータに指定した場合，ジェネレータはエラーを検出する【NOS0811】. 本パラメータに，標準のセクションの名称を指定した場合，ジェネレータはエラーを検出する【NOS0812】.

OsLinkSectionMemoryRegionRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsLinkSection/ OsLinkSectionMemoryRegionRef【NOS0776】
概要	セクションを配置するメモリリージョン
型	/AUTOSAR/EcucDefs/Os/OsMemoryRegion への参照
値の範囲	－
多重度	1
制限事項	SC3，SC4 のみで使用可能

3.8.44 OsStandardMemoryRegion

コンテナ名	/AUTOSAR/EcucDefs/Os/ OsStandardMemoryRegion【NOS1061】
概要	OsStandardMemoryCoreAssignment を省略した場合, システムの標準メモリリージョン情報コンテナ. OsStandardMemoryCoreAssignment を指定した場合, コアの標準メモリリージョン情報コンテナ.
多重度	0 .. *
パラメータ	OsStandardMemoryRomRegionRef
	OsStandardMemoryRamRegionRef
	OsStandardMemoryCoreAssignment
制限事項	SC3，SC4 のみで使用可能. OsStandardMemoryCoreAssignment が同じ値の本コンテナを複数指定した場合，ジェネレータはエラーを検出する【NOS1062】.

OsStandardMemoryRomRegionRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsStandardMemoryRegion/ OsStandardMemoryRomRegionRef【NOS1063】
概要	標準 ROM リージョンの指定
型	/AUTOSAR/EcucDefs/Os/OsMemoryRegion への参照
値の範囲	－
多重度	1
制限事項	SC3，SC4 のみで使用可能. 本パラメータで指定したメモリリージョンの OsMemoryRegionWriteable が true である場合，ジェネレータはエラーを検出する【NOS1064】.

OsStandardMemoryRamRegionRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsStandardMemoryRegion/ OsStandardMemoryRamRegionRef【NOS1065】
概要	標準 RAM リージョンの指定
型	/AUTOSAR/EcucDefs/Os/OsMemoryRegion への参照
値の範囲	－
多重度	1
制限事項	SC3，SC4 のみで使用可能. 本パラメータで指定したメモリリージョンの OsMemoryRegionWriteable が false である場合，ジェネレータは エラーを検出する【NOS1066】.

OsStandardMemoryCoreAssignment

パラメータ名	/AUTOSAR/EcucDefs/Os/OsStandardMemoryRegion/ OsStandardMemoryCoreAssignment【NOS1067】
概要	標準メモリリージョンを使用するコアの指定. 本パラメータで指定したコアで, OsStandardMemoryRomRegionRef および OsStandardMemoryRamRegionRef で指定した標準 ROM，RAM リージョンが使用される.
型	整数型
値の範囲	0 .. 65534
多重度	0 .. 1
制限事項	SC3，SC4 のマルチコア対応 OS のみで使用可能

3.8.45 OsIoc

コンテナ名	/AUTOSAR/EcucDefs/Os/OsIoc【MCOS_Conf1000】
概要	IOC 情報コンテナ
多重度	0 .. 1
パラメータ	－
サブコンテナ	OsIocCommunication
制限事項	SC3，SC4 もしくはマルチコア対応 OS のみで使用可能

3.8.46 OsIocCommunication

コンテナ名	/AUTOSAR/EcucDefs/Os/OsIoc/ OsIocCommunication【MCOS_Conf1003】
概要	IOC オブジェクトコンテナ
多重度	0 .. *
パラメータ	OsIocBufferLength
サブコンテナ	OsIocSenderProperties
	OsIocReceiverProperties
	OsIocDataProperties
制限事項	SC3，SC4 もしくはマルチコア対応 OS のみで使用可能

OsIocBufferLength

パラメータ名	/AUTOSAR/EcucDefs/Os/OsIoc/OsIocCommunication/ OsIocBufferLength【MCOS_Conf1001】
概要	IOC のキューサイズ. 本パラメータを指定した場合，IOC はキューありとなり，本パラメータを省略した場合，IOC はキューなしとなる.
型	整数型
値の範囲	0 .. 0xffffffff
多重度	0 .. 1
制限事項	SC3，SC4 もしくはマルチコア対応 OS のみで使用可能. 本パラメータに 0 を指定した場合，ジェネレータはエラーを検出する【NOS1092】

3.8.47 OsIocSenderProperties

コンテナ名	/AUTOSAR/EcucDefs/Os/OsIoc/OsIocCommunication/ OsIocSenderProperties【MCOS_Conf1015】
概要	センダ情報コンテナ. 本コンテナが 1 つ定義された場合，1:1 通信となり，2 つ以上定義された場合，N:1 通信となる.
多重度	1 .. *
パラメータ	OsIocFunctionImplementationKind
	OsIocSenderId
	OsIocSendingOsApplicationRef
サブコンテナ	－
制限事項	SC3，SC4 もしくはマルチコア対応 OS のみで使用可能. グループ通信の IOC に対して，本コンテナを 2 つ以上定義した場合，ジェネレータはエラーを検出する≪MCOS_Conf1015≫.

OsIocFunctionImplementationKind

パラメータ名	/AUTOSAR/EcucDefs/Os/OsIoc/OsIocCommunication/OsIocSenderProperties/ OsIocFunctionImplementationKind【MCOS_Conf1036】
概要	IOC 用システムサービスの実装方法の選択
型	列挙型
値の範囲	FUNCTION：関数 MACRO：マクロ DO_NOT_CARE：指定しない(デフォルト)
多重度	0 .. 1
制限事項	SC3，SC4 もしくはマルチコア対応 OS のみで使用可能. ATK2 では，本パラメータが省略された場合と，DO_NOT_CARE が指定された場合は，FUNCTION として扱う【IOS223】

OsIocSenderId

パラメータ名	/AUTOSAR/EcucDefs/Os/OsIoc/OsIocCommunication/OsIocSenderProperties/ OsIocSenderId【MCOS_Conf1016】
概要	N:1 通信時におけるセンダ ID
型	整数型
値の範囲	0 .. 255
多重度	0 .. 1
制限事項	SC3，SC4 もしくはマルチコア対応 OS のみで使用可能. 1:1 通信の IOC に対して，本パラメータを指定した場合，ジェネレータはエラーを検出する≪MCOS_Conf1016≫.

OsIocSendingOsApplicationRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsIoc/OsIocCommunication/OsIocSenderProperties/ OsIocSendingOsApplicationRef【MCOS_Conf1014】
概要	センダが所属する OSAP
型	/AUTOSAR/EcucDefs/Os/OsApplication への参照
値の範囲	－
多重度	1
制限事項	SC3，SC4 もしくはマルチコア対応 OS のみで使用可能

3.8.48 OsIocReceiverProperties

コンテナ名	/AUTOSAR/EcucDefs/Os/OsIoc/OsIocCommunication/ OsIocReceiverProperties【MCOS_Conf1017】
概要	レシーバ情報コンテナ
多重度	1
パラメータ	OsIocFunctionImplementationKind
	OsIocReceivingOsApplicationRef
サブコンテナ	－
制限事項	SC3，SC4 もしくはマルチコア対応 OS のみで使用可能

OsIocFunctionImplementationKind

パラメータ名	/AUTOSAR/EcucDefs/Os/OsIoc/OsIocCommunication/OsIocReceiverProperties/ OsIocFunctionImplementationKind≪MCOS_Conf1036≫
概要	IOC 用システムサービスの実装方法の選択
型	列挙型
値の範囲	FUNCTION：関数 MACRO：マクロ DO_NOT_CARE：指定しない(デフォルト)
多重度	0 .. 1
制限事項	SC3，SC4 もしくはマルチコア対応 OS のみで使用可能. ATK2 では，本パラメータが省略された場合と，DO_NOT_CARE が指定された場合は，FUNCTION として扱う【IOS224】

OsIocReceivingOsApplicationRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsIoc/OsIocCommunication/ OsIocReceiverProperties/ OsIocReceivingOsApplicationRef【MCOS_Conf1012】
概要	レシーバが所属する OSAP
型	/AUTOSAR/EcucDefs/Os/OsApplication への参照
値の範囲	－
多重度	1
制限事項	SC3，SC4 もしくはマルチコア対応 OS のみで使用可能

AUTOSAR 仕様との違い

AUTOSAR 仕様では，受信通知機能を使用する場合のコールバック関数の指定パラメータとして，OsIocReceiverPullCB が規定されている【MCOS_Conf1010】．しかし，本仕様では，受信通知機能をサポートしないため，削除した．

3.8.49 OsIocDataProperties

コンテナ名	/AUTOSAR/EcucDefs/Os/OsIoc/OsIocCommunication/ OsIocDataProperties【MCOS_Conf1023】
概要	通信データ情報コンテナ. 本コンテナが 1 つ定義された場合, 単一通信となり, 2 つ以上定義された場合, グループ通信となる.
多重度	1 .. *
パラメータ	OsIocDataPropertyIndex
	OsIocDataTypeRef
	OsIocInitValue
サブコンテナ	－
制限事項	SC3，SC4 もしくはマルチコア対応 OS のみで使用可能. N:1 通信の IOC に対して，本コンテナを 2 つ以上定義した場合，ジェネレータはエラーを検出する≪MCOS_Conf1015≫.

OsIocDataPropertyIndex

パラメータ名	/AUTOSAR/EcucDefs/Os/OsIoc/OsIocCommunication/OsIocDataProperties OsIocDataPropertyIndex【MCOS_Conf1035】
概要	生成される IOC 用システムサービスの引数の順序. 小さい値のデータから順に，引数に与えられる.
型	整数型
値の範囲	0 .. 255
多重度	0 .. 1
制限事項	SC3，SC4 もしくはマルチコア対応 OS のみで使用可能. グループ通信の IOC に対して，本パラメータを省略した場合，ジェネレータはエラーを検出する【NOS1093】. 単一通信の IOC に対して，本パラメータを指定した場合，ジェネレータはエラーを検出する【NOS1094】. 他の通信データに指定した値を，本パラメータに指定した場合，ジェネレータはエラーを検出する【NOS1095】. 1 つの IOC に対する本パラメータの値が，0 からの連番となっていない場合，ジェネレータはエラーを検出する【NOS1096】.

OsIocDataTypeRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsIoc/OsIocCommunication/OsIocDataProperties OsIocDataTypeRef【MCOS_Conf1005】
概要	通信データのデータ型として使用する IMPLEMENTATION-DATA-TYPE
型	IMPLEMENTATION-DATA-TYPE への参照
値の範囲	－
多重度	1
制限事項	SC3，SC4 もしくはマルチコア対応 OS のみで使用可能.
	本パラメータで指定したデータ型は，Rte_Type.h に定義されるものであり，Rte_Type.h は RTE ジェネレータが生成するものである．本 OS を単体で使用する場合は，Rte_Type.h をユーザが用意する必要がある≪OS765≫.

OsIocInitValue

パラメータ名	/AUTOSAR/EcucDefs/Os/OsIoc/OsIocCommunication/OsIocDataProperties OsIocInitValue【MCOS_Conf1024】
概要	通信データの初期値
型	文字列型
値の範囲	－
多重度	0 .. 1
制限事項	SC3，SC4 もしくはマルチコア対応 OS のみで使用可能. OsIocDataTypeRef で指定したデータ型に格納可能な初期値であるかどうかは，ユーザ責任とする【NOS1097】. キューなし通信の IOC に対して，本パラメータを省略した場合，ジェネレータはエラーを検出する【NOS1098】.

3.8.50 OsSpinlock

コンテナ名	/AUTOSAR/EcucDefs/Os/OsSpinlock【MCOS_Conf0258】
概要	スピンロックオブジェクトコンテナ
多重度	0 .. *
パラメータ	OsSpinlockAccessingApplication
	OsSpinlockSuccessor
	OsSpinlockLockMethod
サブコンテナ	－
制限事項	マルチコア対応 OS のみで使用可能

OsSpinlockAccessingApplication

パラメータ名	/AUTOSAR/EcucDefs/Os/OsSpinlock/ OsSpinlockAccessingApplication【MCOS_Conf1021】
概要	スピンロックにアクセス可能な OSAP
型	/AUTOSAR/EcucDefs/Os/OsApplication への参照
値の範囲	－
多重度	1 .. *
制限事項	マルチコア対応 OS のみで使用可能. 本パラメータで指定した OSAP が, すべて同じコアに割付けられている場合, コンフィギュレーションはエラーを検出する【OSa137】. SC1，SC2 においては，本パラメータをエラーチェックのみに使用し, 本パラメータに指定されていない OSAP からでもアクセス可能とする【NOS0955】.

OsSpinlockSuccessor

パラメータ名	/AUTOSAR/EcucDefs/Os/OsSpinlock/ OsSpinlockSuccessor【MCOS_Conf1022】
概要	ネストして獲得可能なスピンロック指定. 獲得するスピンロックの順に本パラメータを記述することで，スピンロック獲得順序を示すリストを管理し，チェックすることができる.
型	/AUTOSAR/EcucDefs/Os/OsSpinlock への参照
値の範囲	－
多重度	0 .. 1
制限事項	マルチコア対応 OS のみで使用可能. 本パラメータで作成されるスピンロック獲得順序のリストが循環している場合，ジェネレータはエラーを検出する【OS666】.

OsSpinlockLockMethod

パラメータ名	/AUTOSAR/EcucDefs/Os/OsSpinlock/ OsSpinlockLockMethod【NOS0956】
概要	スピンロック種別
型	列挙型
値の範囲	LOCK_NOTHING：標準スピンロック LOCK_CAT2_INTERRUPTS：OS 割込み禁止スピンロック LOCK_ALL_INTERRUPTS：全割込み禁止スピンロック
多重度	1
制限事項	マルチコア対応 OS のみで使用可能

AUTOSAR 仕様との違い

本仕様では，AUTOSAR 仕様 V5.0.0 (R4.0 Rev 3)では規定されていない，割込み禁止を行うスピンロックを規定した≪NOS0934≫≪NOS0935≫．しかし，AUTOSAR 仕様 V5.1.0 (R4.1 Rev 1)では，本仕様と同等の仕様が追加されたため，独自に追加したパラメータ名を AUTOSAR 仕様 V5.1.0 (R4.1 Rev 1)に合わせた≪NOS0956≫．ただし，タスクとの排他のみを行う LOCK_WITH_RES_SCHEDULER には未対応である．

3.8.51 OsInterCoreInterrupt

コンテナ名	/AUTOSAR/EcucDefs/Os/OsInterCoreInterrupt【NOS0071】
概要	ICISR オブジェクトコンテナ
多重度	0 .. *
パラメータ	OsInterCoreInterruptInterruptSource
	OsInterCoreInterruptAccessingApplication
	OsInterCoreInterruptResourceRef
	OsInterCoreInterruptStackSize
	OsInterCoreInterruptStackStartAddress
	OsInterCoreInterruptSystemStackSize
	OsInterCoreInterruptSystemStackStartAddress
サブコンテナ	OsInterCoreInterruptTimingProtection
制限事項	マルチコア対応 OS のみで使用可能

OsInterCoreInterruptInterruptSource

パラメータ名	/AUTOSAR/EcucDefs/Os/OsInterCoreInterrupt/ OsInterCoreInterruptInterruptSource【NOS1047】
概要	ICISR の初期状態
型	列挙型
値の範囲	ENABLE：有効 DISABLE：無効
多重度	0 .. 1
制限事項	SC3，SC4 のマルチコア対応 OS のみで使用可能. SC3，SC4 のマルチコア対応 OS で本パラメータを省略した場合，ジェネレータはエラーを検出する【NOS1048】.

OsInterCoreInterruptAccessingApplication

パラメータ名	/AUTOSAR/EcucDefs/Os/OsInterCoreInterrupt/ OsInterCoreInterruptAccessingApplication【NOS0957】
概要	ICISR にアクセス可能な OSAP
型	/AUTOSAR/EcucDefs/Os/OsApplication への参照
値の範囲	－
多重度	0 .. *
制限事項	SC3，SC4 のマルチコア対応 OS のみで使用可能

OsInterCoreInterruptResourceRef

パラメータ名	/AUTOSAR/EcucDefs/Os/OsInterCoreInterrupt/ OsInterCoreInterruptResourceRef【NOS0958】
概要	ICISR が獲得するリソース
型	/AUTOSAR/EcucDefs/Os/OsResource への参照
値の範囲	－
多重度	0 .. *
制限事項	マルチコア対応 OS のみで使用可能. 同じリソースに，異なるコアに割付くタスク，C2ISR，ICISR を関連付けした場合，ジェネレータはエラーを検出する【NOS0959】.

OsInterCoreInterruptStackSize

パラメータ名	/AUTOSAR/EcucDefs/Os/OsInterCoreInterrupt/ OsInterCoreInterruptStackSize【NOS0960】
概要	SC1，SC2 の場合は，ICISR 用スタックのサイズ． SC3，SC4 の場合は以下となる． 信頼 ICISR の場合，本パラメータで指定したサイズと，OsInterCoreInterruptSystemStackSize に指定されたサイズとの合計値を，信頼 ICISR スタックのサイズとして確保する．非信頼 ICISR の場合，非信頼 ICISR 用スタックのサイズとなる．
型	整数型
値の範囲	－
多重度	1
制限事項	マルチコア対応 OS のみで使用可能． 本パラメータに 0 を指定した場合，ジェネレータはエラーを検出する【NOS0961】.

OsInterCoreInterruptStackStartAddress

パラメータ名	/AUTOSAR/EcucDefs/Os/OsInterCoreInterrupt/ OsInterCoreInterruptStackStartAddress【IOS213】
概要	信頼 ICISR の場合，信頼 ICISR 用スタックの先頭番地．非信頼 ICISR の場合，非信頼 ICISR 用スタックの先頭番地． 本パラメータを省略した場合，ジェネレータが，OsInterCoreInterruptStackSize で指定したサイズのスタック領域を確保する．
型	整数型
値の範囲	－
多重度	0 .. 1
制限事項	SC3，SC4 のメモリ保護機能の機能レベル 3 のマルチコア対応 OS のみで使用可能(現状，機能レベル 3 は未実装)． 本パラメータで指定された値が，ターゲット定義のアライメント制約を満たしていない場合，ジェネレータはエラーを検出する【IOS214】.

OsInterCoreInterruptSystemStackSize

パラメータ名	/AUTOSAR/EcucDefs/Os/OsInterCoreInterrupt/ OsInterCoreInterruptSystemStackSize【NOS1049】
概要	信頼 ICISR の場合，本パラメータで指定したサイズと，OsInterCoreInterruptStackSize に指定されたサイズとの合計値を，信頼 ICISR 用スタックのサイズとして確保する． 非信頼 ICISR の場合，非信頼 ICISR 用システムスタックのサイズとなる．非信頼 ICISR で，本パラメータが省略された場合，実装定義の最小値を指定されたものとする．ATK2 では，スタックサイズの最小値をターゲット定義とする≪IOS140≫．
型	整数型
値の範囲	－
多重度	0 .. 1
制限事項	SC3，SC4 のメモリ保護機能の機能レベル 3 のマルチコア対応 OS のみで使用可能(現状，機能レベル 3 は未実装)

OsInterCoreInterruptSystemStackStartAddress

パラメータ名	/AUTOSAR/EcucDefs/ Os/OsInterCoreInterrupt/ OsInterCoreInterruptSystemStackStartAddress【IOS215】
概要	非信頼 ICISR 用システムスタックの先頭番地. 本パラメータを省略した場合，ジェネレータが，OsInterCoreInterruptSystemStackSize で指定したサイズのスタック領域を確保する.
型	整数型
値の範囲	－
多重度	0 .. 1
制限事項	SC3, SC4 のメモリ保護機能の機能レベル 3 のマルチコア対応 OS のみで使用可能(現状，機能レベル 3 は未実装). 信頼 ICISR に対して，本パラメータを指定した場合，ジェネレータはエラーを検出する【IOS216】. 非信頼 ICISR に対して，OsInterCoreInterruptSystemStackSize を指定せずに，本パラメータを指定した場合，ジェネレータはエラーを検出する【IOS217】. 本パラメータで指定された値が，ターゲット定義のアライメント制約を満たしていない場合，ジェネレータはエラーを検出する【IOS218】.

3.8.52 OsInterCoreInterruptTimingProtection

コンテナ名	/AUTOSAR/EcucDefs/Os/OsInterCoreInterrupt/ OsInterCoreInterruptTimingProtection【NOS0072】
概要	ICISR オブジェクトタイミング保護設定コンテナ
多重度	0 .. 1
パラメータ	OsInterCoreInterruptAllInterruptLockMonitor
	OsInterCoreInterruptAllInterruptLockBudget
	OsInterCoreInterruptExecutionBudget
	OsInterCoreInterruptOsInterruptLockMonitor
	OsInterCoreInterruptOsInterruptLockBudget
	OsInterCoreInterruptTimeFrame
	OsInterCoreInterruptResourceLockMonitor
サブコンテナ	－
制限事項	SC2，SC4 のマルチコア対応 OS のみで使用可能

OsInterCoreInterruptAllInterruptLockMonitor

パラメータ名	/AUTOSAR/EcucDefs/Os/OsInterCoreInterrupt/ OsInterCoreInterruptTimingProtection/ OsInterCoreInterruptAllInterruptLockMonitor 【NOS1102】
概要	全割込み禁止時間監視設定
型	ブール型
値の範囲	true：全割込み禁止時間を監視する false：全割込み禁止時間を監視しない
多重度	0 .. 1
制限事項	SC2，SC4 のマルチコア対応 OS のみで使用可能

3.8.53 OsInclude

コンテナ名	/AUTOSAR/EcucDefs/Os/OsInclude【IOS235】
概要	インクルードファイルコンテナ
多重度	0 .. *
パラメータ	OsIncludeFileName
サブコンテナ	－
制限事項	－

OsIncludeFileName

パラメータ名	/AUTOSAR/EcucDefs/Os/ OsInclude/OsIncludeFileName【IOS236】
概要	Os_Lcfg.h からインクルードするヘッダファイル名
型	文字列型
値の範囲	－
多重度	1 .. *
制限事項	－

OsInterCoreInterruptAllInterruptLockBudget

パラメータ名	/AUTOSAR/EcucDefs/Os/OsInterCoreInterrupt/ OsInterCoreInterruptTimingProtection/ OsInterCoreInterruptAllInterruptLockBudget 【NOS1103】
概要	全割込み禁止時間バジェット(単位：秒)
型	浮動小数点型
値の範囲	定義なし
多重度	0 .. 1
制限事項	SC2，SC4 のマルチコア対応 OS のみで使用可能

OsInterCoreInterruptExecutionBudget

パラメータ名	/AUTOSAR/EcucDefs/Os/OsInterCoreInterrupt/ OsInterCoreInterruptTimingProtection/ OsInterCoreInterruptExecutionBudget【NOS1104】
概要	ICISR 実行時間バジェット(単位：秒)
型	浮動小数点型
値の範囲	定義なし
多重度	0 .. 1
制限事項	SC2，SC4 のマルチコア対応 OS のみで使用可能

OsInterCoreInterruptOsInterruptLockMonitor

パラメータ名	/AUTOSAR/EcucDefs/Os/OsInterCoreInterrupt/ OsInterCoreInterruptTimingProtection/ OsInterCoreInterruptOsInterruptLockMonitor 【NOS1105】
概要	OS 割込み禁止時間監視設定
型	ブール型
値の範囲	true：OS 割込み禁止時間を監視する false：OS 割込み禁止時間を監視しない
多重度	0 .. 1
制限事項	SC2，SC4 のマルチコア対応 OS のみで使用可能

OsInterCoreInterruptOsInterruptLockBudget

パラメータ名	/AUTOSAR/EcucDefs/Os/OsInterCoreInterrupt/ OsInterCoreInterruptTimingProtection/ OsInterCoreInterruptOsInterruptLockBudget 【NOS1106】
概要	OS 割込み禁止時間バジェット(単位：秒)
型	浮動小数点型
値の範囲	定義なし
多重度	0 .. 1
制限事項	SC2，SC4 のマルチコア対応 OS のみで使用可能

OsInterCoreInterruptTimeFrame

パラメータ名	/AUTOSAR/EcucDefs/Os/OsInterCoreInterrupt/ OsInterCoreInterruptTimingProtection/ OsInterCoreInterruptTimeFrame【NOS1107】
概要	ICISR タイムフレーム(単位：秒)
型	浮動小数点型
値の範囲	定義なし
多重度	0 .. 1
制限事項	SC2，SC4 のマルチコア対応 OS のみで使用可能

OsInterCoreInterruptResourceLockMonitor

パラメータ名	/AUTOSAR/EcucDefs/Os/OsInterCoreInterrupt/ OsInterCoreInterruptTimingProtection/ OsInterCoreInterruptResourceLockMonitor【NOS1108】
概要	リソース占有時間監視設定
型	ブール型
値の範囲	true：リソース占有時間を監視する false：リソース占有時間を監視しない
多重度	0 .. 1
制限事項	SC2，SC4 のマルチコア対応 OS のみで使用可能

3.8.54 OsMutex

コンテナ名	/AUTOSAR/EcucDefs/Os/OsMutex【NOS0100】
概要	ミューテックスオブジェクトコンテナ
多重度	0 .. *
パラメータ	OsMutexAccessingApplication
	OsMutexCoreAssignment
サブコンテナ	－

OsMutexAccessingApplication

パラメータ名	/AUTOSAR/EcucDefs/Os/OsMutex/ OsMutexAccessingApplication【NOS0962】
概要	ミューテックスにアクセス可能な OSAP
型	/AUTOSAR/EcucDefs/Os/OsApplication への参照
値の範囲	－
多重度	0 .. *
制限事項	SC3，SC4 のマルチコア対応 OS のみで使用可能

OsMutexCoreAssignment

パラメータ名	/AUTOSAR/EcucDefs/Os/OsMutex/ OsMutexCoreAssignment【NOS0963】
概要	ミューテックスの割付けコア
型	整数型
値の範囲	－
多重度	1
制限事項	マルチコア対応 OS のみで使用可能

3.9　システムサービス

3.9.1　GetActiveApplicationMode

<table>
<tr><td colspan="2">C 言語 I/F</td><td colspan="2">AppModeType GetActiveApplicationMode(void)【COS3702】</td></tr>
<tr><td colspan="2">パラメータ[in]</td><td colspan="2">－</td></tr>
<tr><td colspan="2">パラメータ[in/out]</td><td colspan="2">－</td></tr>
<tr><td colspan="2">パラメータ[out]</td><td colspan="2">－</td></tr>
<tr><td colspan="2" rowspan="3">返り値</td><td>StartOS で指定されたアプリケーションモード</td><td>アプリケーションモード取得成功≪COS3703≫</td></tr>
<tr><td>INVALID_APPMODETYPE</td><td>不正な処理単位からの呼出し≪NOS0413≫</td></tr>
<tr><td></td><td>割込み禁止状態からの呼出し≪NOS0414≫</td></tr>
<tr><td rowspan="3">エラーフックに渡されるエラーコード</td><td>標準エラー</td><td colspan="2">－</td></tr>
<tr><td rowspan="2">拡張エラー</td><td>E_OS_CALLEVEL</td><td>不正な処理単位からの呼出し≪NOS0411≫</td></tr>
<tr><td>E_OS_DISABLEDINT</td><td>割込み禁止状態からの呼出し≪NOS0412≫</td></tr>
<tr><td colspan="2">コンフォーマンスクラス</td><td colspan="2">BCC1，BCC2，ECC1，ECC2</td></tr>
<tr><td colspan="2">スケーラビリティクラス</td><td colspan="2">SC1，SC2，SC3，SC4≪NOS0319≫</td></tr>
<tr><td colspan="4">機能</td></tr>
<tr><td colspan="4">本システムサービスは，OS 起動時に指定されたアプリケーションモードを取得する．</td></tr>
</table>

機能仕様

GetActiveApplicationMode は，現在のアプリケーションモードを返す【COS3703】.

不正な処理単位から GetActiveApplicationMode を呼び出した場合，INVALID_APPMODETYPE を返す【NOS0413】．また，割込み禁止状態で呼び出された場合も，INVALID_APPMODETYPE を返す【NOS0414】〔NOS0383〕〔NOS0384〕〔NOS0385〕.

StartOS 呼出し時に Mode が不正であった場合，StartOS 呼出し時に指定された Mode の値を返す【NOS0390】.

3.9.2　StartOS

C 言語 I/F		void StartOS(AppModeType Mode)【COS3706】	
パラメータ[in]		Mode	アプリケーションモード
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値		－	
エラーフックに渡されるエラーコード	標準エラー	－	
	拡張エラー	E_OS_ACCESS	OS が起動しているコアでの呼出し≪NOS0170≫≪NOS1051≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC1，SC2，SC3，SC4≪NOS0319≫	
機能			
本システムサービスは，指定されたアプリケーションモードで OS を起動する．			

機能仕様

StartOS は，Mode で指定されたアプリケーションモードで OS を起動する【COS3709】〔COS0504〕．なお，OS 起動前に呼び出した StartOS からはリターンしない【OS424】．

StartOS 呼出し時に指定された Mode が不正であった場合，E_OS_MODE を終了要因のエラーコードとして，OS シャットダウンを行う【NOS0371】．このとき，システム定義のスタートアップフック，および OSAP 固有のスタートアップフックは呼び出されずに，OSAP 固有のシャットダウンフック，システム定義のシャットダウンフックが順番に呼び出される【NOS0394】．

DisableAllInterrupts または SuspendAllInterrupts による割込み禁止状態で，StartOS を呼び出した場合，EnableAllInterrupts または ResumeAllInterrupts の発行と同等の処理を行なって，割込み許可状態へ遷移した上で，OS を起動する【NOS0372】．

既に OS が起動している状態で，StartOS が呼び出された場合は，エラーコードを E_OS_ACCESS として，エラーフックを呼び出す【NOS0170】〔NOS0409〕．

StartOS は，ユーザが OS 外から呼び出す必要がある≪OSa085≫．

マルチコア対応 OS 仕様

既に OS が起動している状態のコアにおいて，StartOS が呼び出された場合は，エラーコードを E_OS_ACCESS として，エラーフックを呼び出す【NOS1051】〔NOS0409〕．

3.9.3　ShutdownOS

C 言語 I/F		void ShutdownOS(StatusType Error)【COS3712】	
パラメータ[in]		Error	終了要因のエラーコード (OS 定義のものであること)
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値		－	
エラーフックに渡されるエラーコード	標準エラー	E_OS_ACCESS	非信頼 OSAP からの呼出し≪NOS0709≫
	拡張エラー	－	
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC1，SC2，SC3，SC4≪NOS0319≫	
機能			
本システムサービスは，すべての OS サービスを終了する．			

機能仕様

ShutdownOS は, すべての OS サービスを終了しシステム全体を停止する【COS3721】. ShutdownOS で OS サービスを終了するために行う処理は実装定義である【COS1222】. Error に設定されるエラーコードは OS で定義されたものでなければならない【COS3719】. OS で定義されていないエラーコードが指定された場合，Error を E_OS_SHUTDOWN_FATAL として ShutdownOS を呼び出したものとして，シャットダウン処理を行う【NOS0382】. また，不正な処理単位から呼び出した場合も，Error を E_OS_SHUTDOWN_FATAL として ShutdownOS を呼び出したものとして, シャットダウン処理を行う【NOS0362】.

コンフィギュレーション時にシャットダウンフックを有効と設定した場合，シャットダウンを行う前にシャットダウンフックを呼び出す【COS3716】. シャットダウンフックから制御が返ってきた場合，すべての割込みを禁止し無限ループを行う≪OS425≫. 割込み禁止状態で，ShutdownOS が呼ばれた場合は，割込み禁止状態のまま OS サービスの終了処理を行い，無限ループを行う【NOS0710】.

OS の内部状態が不定となり，処理を続行することが不可能となった場合には，OS 内部から ShutdownOS を呼び出すこともある【COS3715】. なお，ShutdownOS がアプリケーションから呼ばれたか OS 内部から呼ばれたかは，シャットダウンフックに渡されるエラーコードによって識別することができる.

OS は，非信頼 OSAP に所属する処理単位からの OS シャットダウン要求を無視する≪OS054≫. ただし，エラーコードを E_OS_ACCESS として，エラーフックを呼び出す【NOS0709】〔NOS0409〕.

ATK2 では，ShutdownOS で OS サービスを終了するために行う処理を，ターゲット定義とする【IOS129】.

マルチコア対応 OS 仕様

ATK2 では，マルチコア対応 OS におけるシステムの終了は，同期 OS シャットダウンのみ対応するため，ShutdownOS はサポートしない【IOS207】.

3.9.4　ActivateTask

C 言語 I/F		StatusType ActivateTask(TaskType TaskID)【COS3210】	
パラメータ[in]		TaskID	タスク ID
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	E_OK	タスク起動成功≪COS3104≫
		E_OS_LIMIT	タスクの起動要求数が最大起動要求回数より多い(W)≪COS3220≫
		E_OS_PROTECTION_ARRIVAL_TASK	タスクの到着間隔がタイムフレーム未満(SC2，SC4 のみ)≪NOS0194≫
	拡張エラー	E_OS_ID	TaskID が不正≪COS3221≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_ACCESS	アクセスが許可されていないタスクが指定された(SC3，SC4 のみ)≪OS056≫
			タスクが所属する OSAP が利用可能でない(SC3，SC4 のみ)≪OS509≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪OS093≫
		E_OS_CORE	起動していないコアへの呼出し≪OSa120≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC1，SC2，SC3，SC4≪NOS0319≫	
機能			
本システムサービスは，TaskID で指定されたタスクを起動する．起動されたタスクはタスクの最初の命令から実行される．			

機能仕様

ActivateTask は，TaskID で指定されたタスクの状態を，休止状態から実行可能状態へ遷移させる【COS3213】．起動されたタスクは OS によってタスクの最初の命令から実行される【COS3214】．

再スケジューリングのタイミングは呼出し元の処理単位やスケジューリングポリシによって異なる【COS3216】．起動されたタスクが拡張タスクの場合，保持していたイベントはタスク起動時にクリアされる【COS3218】．休止状態でない基本タスクに対して発行した場合，ActivateTask では，タスクの状態を変化させず，起動要求回数をインクリメントするのみである【COS3296】．

ActivateTask 呼出し時に指定された TaskID が不正であった場合，E_OS_ID を返す【COS3221】．TaskID で指定されたタスクの起動要求数が最大起動要求回数より多い場合，E_OS_LIMIT を返す

【COS3220】. E_OS_LIMIT を返す場合，タスク起動要求は無視される【COS3217】.

TaskID で指定されたタスクの到着間隔がタイムフレーム未満であった場合，E_OS_PROTECTION_ARRIVAL_TASK を返す【NOS0194】. E_OS_PROTECTION_ARRIVAL_TASK を返す場合，タスクの起動要求は無視される(SC2，SC4 のみ)【NOS0195】.

マルチコア対応 OS 仕様

ActivateTask は，TaskID で指定されたタスクに対してアクセス権があれば，異なるコアに割付いている OSAP に所属するタスクを起動することができる【OS596】. コアを跨いで呼び出す場合，ActivateTask によって対象のタスクが起動した，あるいはエラーが発生して起動しなかったかの結果が確定した上で，呼び出し元の処理単位へリターンする【OS598】. ActivateTask をコアを跨いで呼び出し，エラーが発生した場合は，呼び出し元のコアでエラーフックが呼び出される【OS599】.

3.9.5　TerminateTask

C 言語 I/F		StatusType TerminateTask(void)【COS3223】〔COS1427〕	
パラメータ[in]		－	
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	なし	－	タスク正常終了≪COS3230≫
	拡張エラー	E_OS_RESOURCE	リソースを占有した状態での呼出し≪COS3235≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪OS093≫
		E_OS_SPINLOCK	スピンロックを占有した状態での呼出し≪OS612≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC1，SC2，SC3，SC4≪NOS0319≫	
機能			
本システムサービスは，本システムサービスを呼び出したタスクを終了する．			

機能仕様

TerminateTask は，TerminateTask を呼び出したタスクの状態を，実行状態から休止状態へ遷移させる【COS3226】．タスクが正常に終了した場合，呼出し元に制御は返らない【COS3230】．また，タスクが正常に終了した場合は再スケジューリングが行われる【COS3232】〔COS0462〕．

呼出し元タスクが内部リソースを占有していた場合は内部リソースを解放する【COS3227】．タスクが，標準リソースを占有したまま TerminateTask を呼び出した場合，E_OS_RESOURCE を返す【COS3235】〔COS0835〕．拡張エラー無効時に標準リソースを占有したままタスクを終了した場合の動作は保証されない【COS3229】．

マルチコア対応 OS 仕様

タスクが，スピンロックを占有したまま TerminateTask を呼び出した場合，E_OS_SPINLOCK を返す【OS612】．OS 割込み禁止スピンロック，全割込み禁止スピンロックの獲得により，コアが割込み禁止状態となっていても，E_OS_SPINLOCK を返す【NOS0964】．

3.9.6　ChainTask

C 言語 I/F		StatusType ChainTask(TaskType TaskID)【COS3238】〔COS1427〕	
パラメータ[in]		TaskID	タスク ID
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	なし	－	タスク起動と呼び出したタスク正常終了≪COS3246≫
	標準エラー	E_OS_LIMIT	タスクの起動要求数が最大起動要求回数より多い(W)≪COS3253≫
		E_OS_PROTECTION_ARRIVAL_TASK	タスクの到着間隔がタイムフレーム未満(SC2，SC4 のみ)≪NOS0196≫
	拡張エラー	E_OS_ID	TaskID が不正≪COS3254≫
		E_OS_RESOURCE	リソースを占有した状態での呼出し≪COS3255≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_ACCESS	アクセスが許可されていないタスクが指定された(SC3，SC4 のみ)≪OS056≫
			タスクが所属する OSAP が利用可能でない(SC3，SC4 のみ)≪OS509≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪OS093≫
		E_OS_SPINLOCK	スピンロックを占有した状態での呼出し≪OS612≫
		E_OS_CORE	起動していないコアへの呼出し≪OSa120≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC1，SC2，SC3，SC4≪NOS0319≫	
機能			
本システムサービスは，本システムサービスを呼び出したタスクを終了した後，TaskID で指定されたタスクを起動する．			

機能仕様

ChainTask は，ChainTask を呼び出したタスクの状態を実行状態から休止状態へ遷移させた後，TaskID で指定されたタスクの状態を休止状態から実行可能状態へ遷移させる【COS3241】．ChainTask によって起動されたタスクはそのタスクの初期優先度で最後に起動要求をされたものとして扱う【COS0484】．起動されたタスクが拡張タスクの場合，保持していたイベントはタスク起動時にクリア

される【COS3251】. ChainTask を使用することにより，タスク終了直後に他のタスクを起動することを保証できる【COS3839】.

タスクが正常に終了した場合，呼出し元に制御は返らない【COS3246】. また，タスクが正常に終了した場合は再スケジューリングが行われる【COS3248】〔COS0463〕. 呼出し元のタスクに対して ChainTask を呼び出した場合，多重要求にはならず，すぐに実行可能状態へ遷移させる【COS3242】. 呼出し元のタスクと休止状態の基本タスク以外のタスクに対して発行した場合，ChainTask では，タスクの状態を変化させず，起動要求回数をインクリメントするのみである≪COS3296≫.

呼出し元タスクが内部リソースを占有していた場合は内部リソースを解放する【COS3243】.
タスクが，標準リソースを占有したまま ChainTask を呼び出した場合，E_OS_RESOURCE を返す【COS3255】〔COS0835〕. 拡張エラー無効時に標準リソースを占有したままタスクを終了した場合の動作は保証されない【COS3245】.

ChainTask 呼出し時に指定された TaskID が不正であった場合，E_OS_ID を返す【COS3254】. TaskID で指定されたタスクの起動要求数が最大起動要求回数より多い場合，E_OS_LIMIT を返す【COS3253】. E_OS_LIMIT を返す場合，タスク起動要求は無視される【COS3250】.

TaskID で指定されたタスクの到着間隔がタイムフレーム未満であった場合，E_OS_PROTECTION_ARRIVAL_TASK を返す【NOS0196】. E_OS_PROTECTION_ARRIVAL_TASK を返す場合，タスク起動要求は無視される【NOS0197】.

マルチコア対応 OS 仕様

ChainTask は，TaskID で指定されたタスクに対してアクセス権があれば，異なるコアに割付いている OSAP に所属するタスクを起動することができる【OS600】.
ChainTask をコアを跨いで呼び出す場合，OS は以下の手順で処理を行う必要がある【NOS0046】.

- エラーチェックの後，始めに後続タスクの起動処理を行う.
- 後続タスクが起動した後に ChainTask を呼び出したタスクの終了処理を行う.

この振る舞いは, ChainTask を呼び出したタスクと後続タスクが一時的に同時に実行されるという過渡的な状態が生ずるという点でシングルコアと異なる.

タスクが，スピンロックを占有したまま ChainTask を呼び出した場合，E_OS_SPINLOCK を返す≪OS612≫. OS 割込み禁止スピンロック，全割込み禁止スピンロックの獲得により，コアが割込み禁止状態となっていても，E_OS_SPINLOCK を返す【NOS0965】.

3.9.7　Schedule

C 言語 I/F		StatusType Schedule(void)【COS3258】	
パラメータ[in]		－	
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	E_OK	再スケジューリング成功≪COS3104≫
	拡張エラー	E_OS_RESOURCE	リソースを占有した状態での呼出し≪COS3266≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪OS093≫
		E_OS_SPINLOCK	呼出し元タスクがスピンロックを占有している≪OS624≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC1，SC2，SC3，SC4≪NOS0319≫	
機能			
本システムサービスは，明示的な再スケジューリングを行う．			

機能仕様

Schedule は，Schedule を呼び出したタスクより優先度の高いタスクがある場合に再スケジューリングを行い，それ以外の場合はそのまま呼び出したタスクの処理を継続する【COS3261】.

Schedule による再スケジューリングは，呼出し元タスクが内部リソースを占有しているか，ノンプリエンプティブタスクの場合にのみ行う【COS3263】. Schedule は，内部リソースを占有しているタスクが，より優先度の低いタスクをコアに割付ける際に使用する.

Schedule でプリエンプトが発生した際には内部リソースを解放し，その後，呼出し元タスクが再び実行状態となった場合は自動的に内部リソースを獲得する【COS3262】. タスクが標準リソースを占有したまま Schedule を呼び出した場合，E_OS_RESOURCE を返す【COS3266】〔COS0835〕.

マルチコア対応 OS 仕様

タスクがスピンロックを占有したまま Schedule を呼び出した場合，E_OS_SPINLOCK を返す【OS624】. OS 割込み禁止スピンロック，全割込み禁止スピンロックの獲得により，コアが割込み禁止状態となっていても，E_OS_SPINLOCK を返す【NOS0966】.

3.9.8　GetTaskID

C 言語 I/F		StatusType GetTaskID(TaskRefType TaskID)【COS3268】	
パラメータ[in]		－	
パラメータ[in/out]		－	
パラメータ[out]		TaskID	タスク ID を格納する領域へのポインタ
返り値	標準エラー	E_OK	タスク ID 取得成功≪COS3104≫
	拡張エラー	E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_PARAM_POINTER	ポインタ引数が NULL≪OS566≫
		E_OS_ILLEGAL_ADDRESS	TaskID に不正なアドレスが指定された(SC3，SC4 のみ)≪OS051≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC1，SC2，SC3，SC4≪NOS0319≫	
機能			
本システムサービスは，現在実行中のタスクの ID 情報を取得する．			

機能仕様

現在，実行状態となっているタスクの ID を TaskID に格納する【COS3271】．実行しているタスクが存在しない場合，TaskID に INVALID_TASK を格納する【COS3274】．GetTaskID は，割込み禁止状態であっても，呼び出すことができる【NOS1146】．

マルチコア対応 OS 仕様

GetTaskID を呼び出した処理単位が割付いているコアにおける，現在実行中のタスクの ID 情報を取得する【OSa138】．GetTaskID を呼び出した処理単位が割付いているコアに，現在しているタスクが存在しない場合，TaskID に INVALID_TASK を格納する【NOS0967】．

AUTOSAR 仕様との違い

AUTOSAR 仕様では，割込み禁止の状態で，返り値の型が StatusType のシステムサービスを呼び出した場合，OS はシステムサービスの処理を行わず，E_OS_DISABLEDINT を返すと規定されている≪OS093≫．しかし，割込み禁止の状態で呼び出されたエラーフックやプロテクションフックから，GetTaskID を使用することが想定されるため，本仕様では，割込み禁止状態でも呼び出すことができると規定した≪NOS1146≫．

3.9.9　GetTaskState

C 言語 I/F		StatusType GetTaskState (TaskType TaskID, TaskStateRefType State)【COS3278】	
パラメータ[in]		TaskID	タスク ID
パラメータ[in/out]		－	
パラメータ[out]		State	タスク状態を格納する領域へのポインタ
返り値	標準エラー	E_OK	タスク状態取得成功≪COS3104≫
	拡張エラー	E_OS_ID	TaskID が不正≪COS3286≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_ACCESS	アクセスが許可されていないタスクが指定された(SC3，SC4 のみ)≪OS056≫
			タスクが所属する OSAP が利用可能でない(SC3，SC4 のみ)≪OS509≫
		E_OS_PARAM_POINTER	ポインタ引数が NULL≪OS566≫
		E_OS_ILLEGAL_ADDRESS	State に不正なアドレスが指定された(SC3，SC4 のみ)≪OS051≫
		E_OS_CORE	起動していないコアへの呼出し≪OSa120≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC1，SC2，SC3，SC4≪NOS0319≫	
機能			
本システムサービスは，TaskID で指定されたタスクの状態を取得する.			

機能仕様

GetTaskState は，TaskID で指定されたタスクの状態を取得し，State に格納する【COS3281】. プリエンプティブタスクの場合，取得結果を評価するときには既に別のタスク状態へ遷移している可能性がある【COS3283】. TaskID で指定されたタスクが，多重に起動要求されていても，取得結果に影響はない【COS3284】. GetTaskState は, 割込み禁止状態であっても, 呼び出すことができる【NOS1147】.

GetTaskState 呼出し時に指定された TaskID が不正であった場合，E_OS_ID を返す【COS3286】.

マルチコア対応 OS における仕様

GetTaskState は，コアを跨いで呼び出すことができる【OSa139】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，割込み禁止の状態で，返り値の型が StatusType のシステムサービスを呼び出した場合，OS はシステムサービスの処理を行わず，E_OS_DISABLEDINT を返すと規定されている≪OS093≫．しかし，割込み禁止の状態で呼び出されたエラーフックやプロテクションフックから，GetTaskState を使用することが想定されるため，本仕様では，割込み禁止状態でも呼び出すことができると規定した≪NOS1147≫．

3.9.10 EnableAllInterrupts

C 言語 I/F		void EnableAllInterrupts(void)【COS3301】〔COS0612〕
パラメータ[in]		－
パラメータ[in/out]		－
パラメータ[out]		－
返り値	標準エラー	－
	拡張エラー	－
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2
スケーラビリティクラス		SC1，SC2，SC3，SC4≪NOS0319≫
機能		
本システムサービスは，DisableAllInterrupts によって設定された割込み禁止状態を，割込み許可状態に戻す.		

機能仕様

EnableAllInterrupts は，DisableAllInterrupts によって設定された割込み禁止状態を割込み許可状態に戻す【COS3304】．EnableAllInterrupts は DisableAllInterrupts と対に呼び出さなければならない.

EnableAllInterrupts はハードウェアの特性に合わせて最小限のオーバーヘッドで実装すべきである【COS3838】.

DisableAllInterrupts と EnableAllInterrupts によるクリティカルセクション中は他のシステムサービスを呼出してはならない【COS3306】．他のシステムサービスを呼び出した場合，返り値の型が StatusType のシステムサービスは E_OS_DISABLEDINT を返す≪OS093≫．返り値の型が StatusType でないシステムサービスは無効値を返す【NOS0383】．返り値がないシステムサービスを呼び出した場合は，システムサービス毎に規定する．ただし，SC3，SC4 においては，DisableInterruptSource，EnableInterruptSource を呼び出すことが可能である【NOS0347】.

DisableAllInterrupts と EnableAllInterrupts によるクリティカルセクション中に呼び出された処理単位から，EnableAllInterrupts を呼び出した場合，元の処理単位が呼び出した DisableAllInterrupts による割込み禁止状態が解除される【NOS0636】.

SuspendAllInterrupts と ResumeAllInterrupts，SuspendOSInterrupts と ResumeOSInterrupts によるクリティカルセクション中に EnableAllInterrupts を呼び出した場合，OS は何もしない【NOS0711】．また，DisableAllInterrupts を呼び出していない状態で，EnableAllInterrupts を呼び出した場合，OS は何もしない【NOS0866】.

ATK2 では，オーバーヘッド削減のために，DisableAllInterrupts と EnableAllInterrupts の実装をユーザが再定義することができる【IOS150】.

マルチコア対応 OS 仕様

対象となる割込み要因は，EnableAllInterrupts を呼び出した処理単位が割付いているコアに接続されている割込み要因のみとする【OS591】.

3.9.11 DisableAllInterrupts

C 言語 I/F		void DisableAllInterrupts(void)【COS3310】〔COS0612〕
パラメータ[in]		－
パラメータ[in/out]		－
パラメータ[out]		－
返り値	標準エラー	－
	拡張エラー	－
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2
スケーラビリティクラス		SC1，SC2，SC3，SC4≪NOS0319≫
機能		
本システムサービスは，ターゲットの割込みをすべて禁止し，クリティカルセクションに入る．		

機能仕様

DisableAllInterrupts は，ターゲットの割込みをすべて禁止しクリティカルセクションに入る【COS3313】．DisableAllInterrupts は EnableAllInterrupts と対に呼び出さなければならない．

DisableAllInterrupts はハードウェアの特性に合わせて最小限のオーバーヘッドで実装すべきである【COS3318】.

DisableAllInterrupts と EnableAllInterrupts によるクリティカルセクション中は他のシステムサービスを呼出してはならない【COS3317】．他のシステムサービスを呼び出した場合，返り値の型が StatusType のシステムサービスは E_OS_DISABLEDINT を返す≪OS093≫．返り値の型が StatusType でないシステムサービスは無効値を返す≪NOS0383≫．返り値がないシステムサービスを呼び出した場合は，システムサービス毎に規定する．ただし，SC3，SC4 においては，DisableInterruptSource，EnableInterruptSource を呼び出すことが可能である≪NOS0347≫．

DisableAllInterrupts はネストして呼出してはならない【COS3320】．ネストして呼び出した場合，OS は何もしない【NOS0353】〔COS3320〕．ネストが必要な場合は SuspendAllInterupts/ResumeAllInterrupts または SuspendOSInterrupts/ResumeOSInterrupts を使用すること．

DisableAllInterrupts と EnableAllInterrupts によるクリティカルセクション中に呼び出された処理単位から，EnableAllInterrupts を呼び出した場合，元の処理単位が呼び出した DisableAllInterrupts による割込み禁止状態が解除される≪NOS0636≫.

SuspendAllInterrupts と ResumeAllInterrupts，SuspendOSInterrupts と ResumeOSInterrupts によるクリティカルセクション中に DisableAllInterrupts を呼び出した場合，OS は何もしない【NOS0837】.

ATK2 では，オーバーヘッド削減のために，DisableAllInterrupts と EnableAllInterrupts の実装をユーザが再定義することができる≪IOS150≫.

タイミング保護有効時の注意事項

タイミング保護における全割込み禁止時間の監視が有効である場合，DisableAllInterrupts によるクリティカルクセション中でも，保護違反時処理の割込みが発生する可能性がある.

マルチコア対応 OS 仕様

対象となる割込み要因は，DisableAllInterrupts を呼び出した処理単位が割付いているコアに接続されている割込み要因のみとする【OS590】.

3.9.12 ResumeAllInterrupts

C 言語 I/F		void ResumeAllInterrupts(void)【COS3324】〔COS0612〕	
パラメータ[in]		－	
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	－	
	拡張エラー	－	
エラーフックに渡されるエラーコード	標準エラー	E_OS_STATE	SuspendAllInterrupts を呼び出していない≪NOS0821≫
		E_OS_DISABLEDINT	DisableAllInterrupts 割込み禁止状態からの呼出し≪NOS0712≫
	拡張エラー	－	
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC1，SC2，SC3，SC4≪NOS0319≫	
機能			
本システムサービスは，SuspendAllInterrupts によって設定された割込み禁止状態を，割込み許可状態に戻す.			

機能仕様

ResumeAllInterrupts は，SuspendAllInterrupts によって設定された割込み禁止状態を割込み許可状態に戻す【COS3327】．SuspendAllInterrupts がネストして呼び出されている場合，ResumeAllInterrupts は割込み状態を SuspendAllInterrupts が最後に呼び出された時の状態へ戻す【COS3332】．ResumeAllInterrupts は SuspendAllInterrupts と対に呼び出さなければならない．SuspendAllInterrupts を呼び出していない状態で，ResumeAllInterrupts を呼び出した場合，エラーコードを E_OS_STATE として，エラーフックを呼び出す【NOS0821】〔NOS0409〕〔NOS0657〕.

ResumeAllInterrupts はハードウェアの特性に合わせて最小限のオーバーヘッドで実装すべきである【COS3331】.

SuspendAllInterrupts と ResumeAllInterrupts によるクリティカルセクション中は，SuspendAllInterrupts，ResumeAllInterrupts，SuspendOSInterrupts，ResumeOSInterrupts 以外のシステムサービスを呼出してはならない【COS3330】．他のシステムサービスを呼び出した場合，返り値の型が StatusType のシステムサービスは E_OS_DISABLEDINT を返す≪OS093≫．返り値の型が StatusType でないシステムサービスは無効値を返す【NOS0384】．返り値がないシステムサービスを呼び出した場合は，システムサービス毎に規定する．ただし，SC3，SC4 においては，

DisableInterruptSource，EnableInterruptSource を呼び出すことが可能である≪NOS0347≫.

SuspendAllInterrupts と ResumeAllInterrupts によるクリティカルセクション中に呼び出された処理単位から，ResumeAllInterrupts を呼び出した場合，元の処理単位が呼び出した SuspendAllInterrupts による割込み禁止状態は解除されない【NOS0657】.

マルチコア対応 OS 仕様

対象となる割込み要因は，ResumeAllInterrupts を呼び出した処理単位が割付いているコアに接続されている割込み要因のみとする【OS593】.

3.9.13 SuspendAllInterrupts

C 言語 I/F		void SuspendAllInterrupts(void)【COS3336】〔COS0612〕	
パラメータ[in]		－	
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	－	
	拡張エラー	－	
エラーフックに渡されるエラーコード	標準エラー	E_OS_LIMIT	ネスト回数の上限値超過≪NOS0714≫
		E_OS_DISABLEDINT	DisableAllInterrupts 割込み禁止状態からの呼出し≪NOS0712≫
	拡張エラー	－	
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC1，SC2，SC3，SC4≪NOS0319≫	
機能			
本システムサービスは，ターゲットの割込み状態を保存した後，ターゲットの割込みをすべて禁止し，クリティカルセクションに入る．			

機能仕様

ターゲットの割込み状態を保存した後，ターゲットの割込みをすべて禁止しクリティカルセクションに入る【COS3339】．SuspendAllInterrupts は ResumeAllInterrupts と対に呼び出さなければならない．

SuspendAllInterrupts はハードウェアの特性に合わせて最小限のオーバーヘッドで実装すべきである【COS3344】.

SuspendAllInterrupts をネストして呼び出す場合の，ネスト回数の上限値は実装定義である【NOS0713】．上限値を超えて呼び出した場合，エラーコードを E_OS_LIMIT として，エラーフックを呼び出す【NOS0714】〔NOS0409〕.

SuspendAllInterrupts と ResumeAllInterrupts によるクリティカルセクション中は，SuspendAllInterrupts，ResumeAllInterrupts，SuspendOSInterrupts，ResumeOSInterrupts 以外のシステムサービスを呼出してはならない【COS3343】．他のシステムサービスを呼び出した場合，返り値の型が StatusType のシステムサービスは E_OS_DISABLEDINT を返す≪OS093≫．返り値の型が StatusType でないシステムサービスは無効値を返す≪NOS0384≫．返り値がないシステムサービスを呼び出した場合は，システムサービス毎に規定する．ただし，SC3，SC4 においては，DisableInterruptSource，EnableInterruptSource を呼び出すことが可能である≪NOS0347≫.

SuspendAllInterrupts と ResumeAllInterrupts によるクリティカルセクション中に呼び出された処理単位から，ResumeAllInterrupts を呼び出した場合，元の処理単位が呼び出した SuspendAllInterrupts による割込み禁止状態は解除されない≪NOS0657≫.

ATK2 では，SuspendAllInterrupts のネスト回数の上限値を 255 と規定した≪IOS062≫.

タイミング保護有効時の注意事項

タイミング保護における全割込み禁止時間の監視が有効である場合，SuspendAllInterrupts によるクリティカルクセション中でも，保護違反時処理の割込みが発生する可能性がある.

マルチコア対応 OS 仕様

対象となる割込み要因は，SuspendAllInterrupts を呼び出した処理単位が割付いているコアに接続されている割込み要因のみとする【OS592】.

3.9.14 ResumeOSInterrupts

C 言語 I/F		void ResumeOSInterrupts(void)【COS3348】〔COS0612〕	
パラメータ[in]		－	
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	－	
	拡張エラー	－	
エラーフックに渡されるエラーコード	標準エラー	E_OS_STATE	SuspendOSInterrupts を呼び出していない≪NOS0822≫
		E_OS_DISABLEDINT	DisableAllInterrupts 割込み禁止状態からの呼出し≪NOS0712≫
	拡張エラー	－	
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC1，SC2，SC3，SC4≪NOS0319≫	
機能			
本システムサービスは，SuspendOSInterrupts によって設定された割込み禁止状態を，割込み許可状態に戻す.			

機能仕様

SuspendOSInterrupts によって設定された割込み禁止状態を割込み許可状態に戻す【COS3351】. SuspendOSInterrupts がネストして呼び出されている場合，ResumeOSInterrupts は割込み状態を SuspendOSInterrupts が最後に呼び出された時の状態へ戻す【COS3357】. ResumeOSInterrupts は SuspendOSInterrupts と対に呼び出さなければならない. SuspendOSInterrupts を呼び出していない状態で，ResumeOSInterrupts を呼び出した場合，エラーコードを E_OS_STATE として，エラーフックを呼び出す【NOS0822】〔NOS0409〕〔NOS0658〕.

ResumeOSInterrupts はハードウェアの特性に合わせて最小限のオーバーヘッドで実装すべきである【COS3355】.

SuspendOSInterrupts と ResumeOSInterrupts によるクリティカルセクション中は，SuspendAllInterrupts，ResumeAllInterrupts，SuspendOSInterrupts，ResumeOSInterrupts 以外のシステムサービスを呼出してはならない【COS3354】. 他のシステムサービスを呼び出した場合，返り値の型が StatusType のシステムサービスは E_OS_DISABLEDINT を返す≪OS093≫. 返り値の型が StatusType でないシステムサービスは無効値を返す【NOS0385】. 返り値がないシステムサービスを呼び出した場合は，システムサービス毎に規定する. ただし，SC3，SC4 においては，DisableInterruptSource，EnableInterruptSource を呼び出すことが可能である≪NOS0347≫.

SuspendOSInterrupts と ResumeOSInterrupts によるクリティカルセクション中に呼び出された処理単位から，ResumeOSInterrupts を呼び出した場合，元の処理単位が呼び出した SuspendOSInterrupts による割込み禁止状態は解除されない【NOS0658】.

マルチコア対応 OS 仕様

対象となる割込み要因は，ResumeOSInterrupts を呼び出した処理単位が割付いているコアに接続されている割込み要因のみとする【OS595】.

3.9.15 SuspendOSInterrupts

C 言語 I/F		void SuspendOSInterrupts(void)【COS3361】〔COS0612〕	
パラメータ[in]		－	
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	－	
	拡張エラー	－	
エラーフックに渡されるエラーコード	標準エラー	E_OS_LIMIT	ネスト回数の上限値超過≪NOS0716≫
		E_OS_DISABLEDINT	DisableAllInterrupts 割込み禁止状態からの呼出し≪NOS0712≫
	拡張エラー	－	
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC1，SC2，SC3，SC4≪NOS0319≫	
機能			
本システムサービスは，ターゲットの割込み状態を保存した後，C2ISR をすべて禁止し，クリティカルセクションに入る．			

機能仕様

ターゲットの割込み状態を保存した後，C2ISR をすべて禁止しクリティカルセクションに入る【COS3364】．SuspendOSInterrupts は ResumeOSInterrupts と対に呼び出さなければならない．

SuspendOSInterrupts はハードウェアの特性に合わせて最小限のオーバーヘッドで実装すべきである【COS3369】．

SuspendOSInterrupts をネストして呼び出す場合の，ネスト回数の上限値は実装定義である【NOS0715】．上限値を超えて呼び出した場合，エラーコードを E_OS_LIMIT として，エラーフックを呼び出す【NOS0716】〔NOS0409〕．

SuspendOSInterrupts と ResumeOSInterrupts によるクリティカルセクション中は SuspendAllInterrupts，ResumeAllInterrupts，SuspendOSInterrupts，ResumeOSInterrupts 以外のシステムサービスを呼出してはならない【COS3368】．他のシステムサービスを呼び出した場合，返り値の型が StatusType のシステムサービスは E_OS_DISABLEDINT を返す≪OS093≫．返り値の型が StatusType でないシステムサービスは無効値を返す≪NOS0385≫．返り値がないシステムサービスを呼び出した場合は，システムサービス毎に規定する．ただし，SC3，SC4 においては，DisableInterruptSource，EnableInterruptSource を呼び出すことが可能である≪NOS0347≫．

SuspendOSInterrupts はすべての C2ISR を禁止するが，効率的な実装方法ができない場合はそれ以外の割込みを禁止してもよい【COS3370】.

SuspendOSInterrupts と ResumeOSInterrupts によるクリティカルセクション中に呼び出された処理単位から，ResumeOSInterrupts を呼び出した場合，元の処理単位が呼び出した SuspendOSInterrupts による割込み禁止状態は解除されない≪NOS0658≫.

ATK2 では，SuspendOSInterrupts のネスト回数の上限値を 255 と規定した≪IOS063≫.

マルチコア対応 OS 仕様

対象となる割込み要因は，SuspendOSInterrupts を呼び出した処理単位が割付いているコアに接続されている割込み要因のみとする【OS594】.

3.9.16 GetISRID

C 言語 I/F		ISRType GetISRID(void)【OS511】	
パラメータ[in]		－	
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値		現在実行中の ISR ID	ISR ID 取得成功≪OS263≫
		INVALID_ISR	実行中の ISR が存在しない≪OS264≫
			不正な処理単位からの呼出し≪NOS0416≫
エラーフックに渡されるエラーコード	標準エラー	－	
	拡張エラー	E_OS_CALLEVEL	不正な処理単位からの呼出し≪NOS0411≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC1，SC2，SC3，SC4【OS515】	
機能			
本システムサービスは，現在実行中の C2ISR の ID 情報を取得する．			

機能仕様

GetISRID は，C2ISR，および，C2ISR 実行中に発生したフックルーチンから呼び出された場合，実行中の ISR ID を返す【OS263】．それ以外の場合は INVALID_ISR を返す【OS264】．GetISRID は，割込み禁止状態であっても，呼び出すことができる【NOS1148】．

不正な処理単位から GetISRID を呼び出した場合，INVALID_ISR を返す【NOS0416】．

マルチコア対応 OS 仕様

GetISRID を呼び出した処理単位が割付いているコアにおける，現在実行中の C2ISR の ID 情報を取得する【NOS0968】．GetISRID を呼び出した処理単位が割付いているコアに，現在している C2ISR が存在しない場合，INVALID_ISR を返す【NOS0969】．

3.9.17 EnableInterruptSource

C 言語 I/F		StatusType EnableInterruptSource(ISRType EnableISR) 【NOS0238】〔COS0612〕	
パラメータ[in]		EnableISR	割込み要因を有効にする C2ISR の ID
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	E_OK	割込み要因有効成功≪COS3104≫
	拡張エラー	E_OS_ID	EnableISR が不正≪NOS0406≫
			C1ISR の指定≪NOS0695≫
		E_OS_CALLEVEL	C1ISR からの呼出し≪IOS151≫
		E_OS_ACCESS	アクセスが許可されていない ISR が指定された≪OS056≫
			ISR が所属する OSAP が利用可能でない≪OS509≫
		E_OS_CORE	コアを跨いだ呼出し≪NOS0970≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC3，SC4【NOS0887】	
機能			
本システムサービスは，EnableISR で指定された C2ISR の割込み要因を有効にする.			

機能仕様

EnableInterruptSource は，EnableISR で指定された C2ISR の割込み要因を有効にする.

既に割込み要因が有効の場合，何もせず E_OK を返す【NOS0894】.

EnableInterruptSource 呼出し時に指定された EnableISR が不正であった場合，E_OS_ID を返す【NOS0406】. EnableISR に C1ISR が指定された場合も，E_OS_ID を返す【NOS0695】.

DisableInterruptSource と EnableInterruptSource によるクリティカルセクション中に呼び出された処理単位から，EnableInterruptSource を呼び出した場合，元の処理単位が呼び出した DisableInterruptSource による割込み禁止状態が解除される【NOS0634】.

マルチコア対応 OS 仕様

EnableInterruptSource を，コアを跨いで呼び出した場合，E_OS_CORE を返す【NOS0970】.

EnableISR に ICISR を指定した際に, 割込み要因が無効となっている間に RaiseInterCoreInterrupt によって対象の ICISR に対する割込み要求を受け付けていた場合，指定した ICISR を実行する【NOS1045】.

AUTOSAR 仕様との違い

本仕様では割込み要因ごとの有効処理を追加した．インターフェースとして AUTOSAR 仕様 V2.0.1 で規定されていた EnableInterruptSource を採用している≪NOS0238≫．C1ISR は OS 管理外であるため，EnableInterruptSource は使用できないように規定した≪NOS0695≫．

3.9.18 DisableInterruptSource

C 言語 I/F		StatusType DisableInterruptSource(ISRType DisableISR) 【NOS0239】〔COS0612〕	
パラメータ[in]		DisableISR	割込み要因を無効にする C2ISR の ID
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	E_OK	割込み要因無効成功≪COS3104≫
	拡張エラー	E_OS_ID	DisableISR が不正≪NOS0407≫
			C1ISR の指定≪NOS0696≫
		E_OS_CALLEVEL	C1ISR からの呼出し≪IOS151≫
		E_OS_ACCESS	アクセスが許可されていない ISR が指定された≪OS056≫
			ISR が所属する OSAP が利用可能でない≪OS509≫
		E_OS_CORE	コアを跨いだ呼出し≪NOS0971≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC3，SC4【NOS0888】	
機能			
本システムサービスは，DisableISR で示される C2ISR の割込み要因を無効にする.			

機能仕様

DisableInterruptSource は，DisableISR で示される C2ISR の割込み要因を無効にする.

既に割込み要因が無効の場合，何もせず E_OK を返す【NOS0895】.

DisableInterruptSource 呼出し時に指定された DisableISR が不正であった場合，E_OS_ID を返す【NOS0407】. DisableISR に C1ISR が指定された場合も，E_OS_ID を返す【NOS0696】.

DisableInterruptSource と EnableInterruptSource によるクリティカルセクション中に呼び出された処理単位から，EnableInterruptSource を呼び出した場合，元の処理単位が呼び出した DisableInterruptSource による割込み禁止状態が解除される≪NOS0634≫.

マルチコア対応 OS 仕様

DisableInterruptSource を，コアを跨いで呼び出した場合，E_OS_CORE を返す【NOS0971】.

DisableISR に ICISR を指定した場合, RaiseInterCoreInterrupt を呼び出されても, 指定した ICISR が起動しない状態となる【NOS1046】. したがって，本 OS がディスパッチやシャットダウン要求のために使用するコア間割込み自身は禁止されない.

AUTOSAR 仕様との違い

本仕様では割込み要因ごとの無効処理を追加した．インターフェースとして AUTOSAR 仕様 V2.0.1 で規定されていた DisableInterruptSource を採用している≪NOS0239≫．C1ISR は OS 管理外であるため，DisableInterruptSource は使用できないように規定した≪NOS0696≫．

3.9.19 SetEvent

C 言語 I/F		StatusType SetEvent(TaskType TaskID, EventMaskType Mask)【COS3508】	
パラメータ[in]		TaskID	タスク ID
		Mask	イベントマスク値
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	E_OK	イベントセット成功≪COS3104≫
		E_OS_PROTECTION_ARRIVAL_TASK	イベントセットにより TaskID で指定されたタスクの待ち解除が，そのタスクのタイムフレーム未満(SC2，SC4 のみ)≪NOS0204≫
	拡張エラー	E_OS_ID	TaskID が不正≪COS3516≫
		E_OS_ACCESS	TaskID で指定されたタスクが拡張タスクでない≪COS3517≫
			アクセスが許可されていないタスクが指定された(SC3，SC4 のみ)≪OS056≫
			タスクが所属する OSAP が利用可能でない(SC3，SC4 のみ)≪OS509≫
		E_OS_STATE	TaskID で指定されたタスクが休止状態である≪COS3518≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪OS093≫
		E_OS_CORE	起動していないコアへの呼出し≪OSa120≫
コンフォーマンスクラス		ECC1，ECC2	
スケーラビリティクラス		SC1，SC2，SC3，SC4≪NOS0319≫	
機能			
本システムサービスは，TaskID で指定されたタスクに Mask で指定されたイベントを設定する．			

機能仕様

SetEvent は，TaskID で指定されたタスクのイベントマスク値に，Mask で指定されたイベントを設定する【COS3513】．TaskID で指定されたタスクが，Mask で指定されたイベントのうち，少なくとも 1 つのイベントを待っている場合，TaskID で指定されたタスクは待ち状態から実行可能状態へ遷移し，再スケジューリングが行われる【COS3553】． Mask で指定されていないイベントは変化しない

【COS3514】.

SetEvent 呼出し時に指定された TaskID が不正であった場合, E_OS_ID を返す【COS3516】. TaskID で指定されたタスクが拡張タスクでない場合, E_OS_ACCESS を返す【COS3517】〔COS0702〕. TaskID で指定されたタスクの状態が休止状態であった場合，E_OS_STATE を返す【COS3518】.

TaskID で指定されたタスクの到着間隔がタイムフレーム未満であった場合，E_OS_PROTECTION_ARRIVAL_TASK を返す(SC2，SC4 のみ)【NOS0204】. E_OS_PROTECTION_ARRIVAL_TASK を返す場合，イベントはセットせず TaskID で指定されたタスクを待ち状態から実行可能状態へ遷移させない(SC2，SC4 のみ)【NOS0205】.

なお，SetEvent によって TaskID で指定されたタスクが待ち解除された場合でも，タスクが保持するイベントマスク値のクリアは行わない【NOS0368】.

マルチコア対応 OS 仕様

SetEvent は，TaskID で指定されたタスクに対してアクセス権があれば，異なるコアに割付いている OSAP に所属するタスクのイベントをセットすることができる【OS602】. コアを跨いで呼び出す場合，SetEvent によって対象のタスクが起動した，あるいはエラーが発生して起動しなかったかの結果が確定した上で，呼び出し元の処理単位へリターンする【OS604】. SetEvent をコアを跨いで呼び出し，エラーが発生した場合は，呼び出し元のコアでエラーフックが呼び出される【OS605】.

3.9.20 ClearEvent

C 言語 I/F		StatusType ClearEvent(EventMaskType Mask)【COS3520】	
パラメータ[in]		Mask	イベントマスク値
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	E_OK	イベントクリア成功≪COS3104≫
	拡張エラー	E_OS_ACCESS	呼出し元タスクが拡張タスクでない≪COS3526≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪OS093≫
コンフォーマンスクラス		ECC1，ECC2	
スケーラビリティクラス		SC1，SC2，SC3，SC4≪NOS0319≫	
機能			
本システムサービスは，Mask で指定されたイベントをクリアする.			

機能仕様

ClearEvent は，ClearEvent を呼び出したタスクの現在のイベントマスク値から，Mask で指定されたイベントをクリアする【COS3523】.

呼出し元タスクが拡張タスクでない場合，E_OS_ACCESS を返す【COS3526】〔COS0702〕.

3.9.21 GetEvent

C 言語 I/F		StatusType GetEvent(TaskType TaskID, EventMaskRefType Event)【COS3529】	
パラメータ[in]		TaskID	タスク ID
パラメータ[in/out]		－	
パラメータ[out]		Event	イベント状態を格納する領域へのポインタ
返り値	標準エラー	E_OK	イベント状態取得成功≪COS3104≫
	拡張エラー	E_OS_ID	TaskID が不正≪COS3538≫
		E_OS_ACCESS	TaskID で指定されたタスクが拡張タスクでない≪COS3539≫
			アクセスが許可されていないタスクが指定された(SC3，SC4 のみ)≪OS056≫
			タスクが所属する OSAP が利用可能でない(SC3，SC4 のみ)≪OS509≫
		E_OS_STATE	TaskID で指定されたタスクが休止状態≪COS3540≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪OS093≫
		E_OS_PARAM_POINTER	ポインタ引数が NULL≪OS566≫
		E_OS_ILLEGAL_ADDRESS	Event に不正なアドレスが指定された(SC3，SC4 のみ)≪OS051≫
		E_OS_CORE	起動していないコアへの呼出し≪OSa120≫
コンフォーマンスクラス		ECC1，ECC2	
スケーラビリティクラス		SC1，SC2，SC3，SC4≪NOS0319≫	
機能			
本システムサービスは，TaskID で指定されたタスクが保持しているイベントマスク値を取得する．			

機能仕様

GetEvent は，GetEvent が呼び出された時点の TaskID で指定されたタスクが保持するイベントマス

ク値を Event で指定された領域にコピーする【COS3535】.

GetEvent 呼出し時に指定された TaskID が不正であった場合, E_OS_ID を返す【COS3538】. TaskID で指定されたタスクが拡張タスクでない場合, E_OS_ACCESS を返す【COS3539】〔COS0702〕. TaskID で指定されたタスクの状態が休止状態であった場合，E_OS_STATE を返す【COS3540】.

マルチコア対応 OS における仕様

GetEvent は，コアを跨いで呼び出すことができる【NOS0972】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，GetEvent は，コアを跨いで呼び出すことができないと規定されている【OSa140】．しかし，GetEvent をコアを跨いで呼び出すことによる不都合が想定できないため，本仕様では呼び出すことができると規定した≪NOS0972≫.

3.9.22 WaitEvent

C 言語 I/F		StatusType WaitEvent(EventMaskType Mask)【COS3542】	
パラメータ[in]		Mask	イベント待ちを行うイベントマスク値
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	E_OK	イベント待ちが完了≪COS3104≫
		E_OS_PROTECTION_ARRIVAL_TASK	既にセットされているイベントに対し，タイムフレーム未満で呼出し(SC2，SC4 のみ)≪NOS0206≫
	拡張エラー	E_OS_ACCESS	呼出し元タスクが拡張タスクでない≪COS3549≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_RESOURCE	呼出し元タスクがリソースを占有している≪COS3550≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪OS093≫
		E_OS_SPINLOCK	呼出し元タスクがスピンロックを占有している≪OS622≫
コンフォーマンスクラス		ECC1，ECC2	
スケーラビリティクラス		SC1，SC2，SC3，SC4≪NOS0319≫	
機能			
本システムサービスは，本システムサービスを呼び出したタスクを待ち状態とする．			

機能仕様

WaitEvent は，Mask で指定されたイベントマスク値がセットされるまで，WaitEvent を呼び出したタスクを待ち状態に遷移させる【COS3545】〔COS0722〕．待ち状態に入る際には，内部リソースを解放し再スケジューリングが行われる【COS3546】〔COS0465〕．Mask で指定されたイベントマスク値が既にセットされた状態であった場合は，タスクは実行状態を継続する【COS0721】.

タスクが標準リソースを占有したまま WaitEvent を呼び出した場合，E_OS_RESOURCE を返す【COS3550】〔COS0835〕．呼出し元タスクが拡張タスクでない場合，E_OS_ACCESS を返す【COS3549】〔COS0702〕〔COS0412〕.

Mask で指定されたイベントマスク値が既にセットされた状態，かつ，呼出し元タスクの到着間隔がタイムフレーム未満であった場合，E_OS_PROTECTION_ARRIVAL_TASK を返す(SC2，SC4 のみ)【NOS0206】.

マルチコア対応 OS 仕様

タスクがスピンロックを占有したまま WaitEvent を呼び出した場合，E_OS_SPINLOCK を返す【OS622】．OS 割込み禁止スピンロック，全割込み禁止スピンロックの獲得により，コアが割込み禁止状態となっていても，E_OS_SPINLOCK を返す【NOS0973】.

3.9.23 GetResource

C 言語 I/F		StatusType GetResource(ResourceType ResID)【COS3407】	
パラメータ[in]		ResID	リソース ID
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	E_OK	リソース獲得成功≪COS3104≫
	拡張エラー	E_OS_ID	ResID が不正≪COS3418≫
			内部リソースの指定≪COS0829≫
		E_OS_ACCESS	ResID で指定されたリソースが既に占有されている≪COS3837≫
			ResID の上限優先度が呼出し元の初期優先度より低い≪COS3419≫
			アクセスが許可されていないリソースが指定された(SC3，SC4 のみ)≪OS056≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪OS093≫
		E_OS_CORE	コアを跨いだ呼出し≪OS589≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC1，SC2，SC3，SC4≪NOS0319≫	
機能			
本システムサービスは，ResID で指定されたリソースを獲得する.			

機能仕様

GetResource は，ResID で指定されたリソースを獲得する．ResID で指定されたリソースの属性が標準リソースであった場合，リソースを獲得した処理単位は上限優先度プロトコルにより現在優先度が引き上げられ，ReleaseResource を呼び出すまで排他区間となる【COS3410】〔COS0805〕.

GetResource 呼出し時に指定された ResID が不正であった場合, E_OS_ID を返す【COS3418】. ResID で指定されたリソースが, 既に占有されている場合, E_OS_ACCESS を返す【COS3837】〔COS0810〕. ResID で指定されたリソースの上限優先度が呼出し元の初期優先度よりも低い場合，E_OS_ACCESS を返す【COS3419】〔COS0804〕.

マルチコア対応 OS 仕様

スピンロックを占有している処理単位であっても，リソースを獲得することができる【OSa141】. GetResource を，コアを跨いで呼び出した場合，E_OS_CORE を返す≪OS589≫.

3.9.24 ReleaseResource

C 言語 I/F		StatusType ReleaseResource(ResourceType ResID)【COS3421】	
パラメータ[in]		ResID	リソース ID
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	E_OK	リソース解放成功≪COS3104≫
	拡張エラー	E_OS_ID	ResID が不正≪COS3428≫
			内部リソースの指定≪COS0829≫
		E_OS_ACCESS	アクセスが許可されていないリソースが指定された(SC3，SC4 のみ)≪OS056≫
		E_OS_NOFUNC	ResID が呼出し元で占有されていないか，解放順序が適切でない≪COS3429≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪OS093≫
		E_OS_CORE	コアを跨いだ呼出し≪OS589≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC1，SC2，SC3，SC4≪NOS0319≫	
機能			
本システムサービスは，ResID で指定されたリソースを解放する.			

機能仕様

ReleaseResource は，ResID で指定されたリソースを解放し，排他区間を終了し，再スケジューリングが行われる【COS3424】．タスクがリソースを解放した時，OS はタスクの現在優先度を，そのリソースを獲得する前のタスク優先度に戻す【COS0828】．ReleaseResource は GetResource と対に呼び出さなければならない.

ReleaseResource 呼出し時に指定された ResID が不正であった場合, E_OS_ID を返す【COS3428】.
ResID で指定されたリソースが呼出し元の処理単位で占有されていない，またはリソースの解放順序が LIFO となっていない場合，E_OS_NOFUNC を返す【COS3429】.

マルチコア対応 OS 仕様

ReleaseResource を，コアを跨いで呼び出した場合，E_OS_CORE を返す≪OS589≫.

OSEK 仕様との違い

OSEK 仕様では, ResID で指定されたリソースの上限優先度が処理単位の初期優先度よりも低い場合,

E_OS_ACCESS を返すと規定されている【COS3430】．しかし，リソースの上限優先度が処理単位の初期優先度よりも低い場合，そのリソースを獲得することができないため，本エラーが発生することはありえない．したがって，本仕様では，本規定を削除した．

3.9.25 IncrementCounter

C 言語 I/F		StatusType IncrementCounter(CounterType CounterID)【OS399】	
パラメータ[in]		CounterID	カウンタ ID
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	E_OK	ティックインクリメント成功≪COS3104≫
	拡張エラー	E_OS_ID	CounterID が不正か，カウンタがハードウェアカウンタである≪OS285≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_STATE	他処理単位によって操作中のカウンタが指定された≪NOS0058≫
		E_OS_ACCESS	アクセスが許可されていないカウンタが指定された(SC3，SC4 のみ)≪OS056≫
			カウンタが所属する OSAP が利用可能でない(SC3，SC4 のみ)≪OS509≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪OS093≫
		E_OS_CORE	コアを跨いだ呼出し≪OS629≫≪OS589≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC1，SC2，SC3，SC4【OS530】	
機能			
本システムサービスは，CounterID で指定されたソフトウェアカウンタをインクリメントする.			

機能仕様

IncrementCounter は，CounterID で指定されたソフトウェアカウンタをインクリメントし，E_OK を返す【OS286】. カウンタをインクリメントした結果，カウンタに接続された OS オブジェクト(アラーム，スケジュールテーブル)が満了した場合，アクションを実行する．アクションの結果，何らかのエラーが発生した場合でも IncrementCounter は E_OK を返す【OS321】〔COS0906〕.

IncrementCounter 呼出し時に指定された CounterID が不正であった場合，また，CounterID で指定されたカウンタがハードウェアカウンタであった場合，E_OS_ID を返す【OS285】.

複数のアラーム，あるいは複数のスケジュールテーブルが同時に満了する場合，各満了処理が終了するごとに，再スケジューリングを行う【NOS0342】. したがって，IncrementCounter がタスクから呼び出された場合，再スケジューリングを行う可能性がある【OS529】〔NOS0342〕. また，C2ISR から IncrementCounter を実行している間に，より割込み優先度の高い C2ISR が起動する可能性がある【NOS0351】. これらの場合，CounterID で指定されたカウンタは操作中となり，他の処理単位から操

作中のカウンタに対して IncrementCounter を発行した場合，エラーとして E_OS_STATE を返す【NOS0058】.

使用上の注意

低優先度のタスクから IncrementCounter を呼び出した際に満了処理が発生し，同じティックで複数のアクションの処理を実行する場合に，それらの処理が不可分に実行されるとは限らない.

例えば，低優先度のタスクで IncrementCounter を呼び出した際に満了処理が発生し，同じティックのアクションとして中優先度タスクと高優先度タスクに対して ActivateTask を行った場合，先に中優先度のタスクに対して ActivateTask が呼び出され中優先度タスクが起動した後に，高優先度のタスクに対して ActivateTask が呼び出される可能性がある.

上記のような優先度逆転が発生する場合があるため，IncrementCounter は高優先度のタスクから呼び出すことを推奨する.

IncrementCounter 実行中に発生した割込みによって起動した C2ISR から CancelAlarm を発行した場合，CancelAlarm 後に，1 回満了処理が実行される可能性がある.

IncrementCounter 実行による満了処理によってタスクディスパッチが発生するので，IncrementCounter 実行中にスタックオーバーフローを検知する場合がある．SC3，SC4 では，このスタックオーバーフローによるプロテクションフックから，IncrementCounter を呼び出したタスク，C2ISR を強制終了することができるが，この場合，対象のカウンタは操作中のままとなる．ただし，信頼タスク，非信頼タスクともに，実装定義の最小値以上のスタックを確保することから，この最小値をすべでのシステムサービスで使用するスタックサイズより大きくすることで，この問題は回避可能である．信頼関数経由で IncrementCounter を呼び出す場合も，信頼関数が使用するスタックサイズを指定するので，見積もりが正しければ，スタックオーバーフローすることはない.

マルチコア対応 OS における仕様

CounterID で指定されたカウンタが，IncrementCounter を呼び出した処理単位と異なるコアに割付けられている場合，E_OS_CORE を返す≪OS629≫≪OS589≫.

3.9.26 GetCounterValue

C 言語 I/F		StatusType GetCounterValue(CounterType CounterID, TickRefType Value)【OS383】	
パラメータ[in]		CounterID	カウンタ ID
パラメータ[in/out]		－	
パラメータ[out]		Value	ティックを格納する領域へのポインタ
返り値	標準エラー	E_OK	ティック取得成功≪COS3104≫
	拡張エラー	E_OS_ID	CounterID が不正≪OS376≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_ACCESS	アクセスが許可されていないカウンタが指定された(SC3，SC4 のみ)≪OS056≫
			カウンタが所属する OSAP が利用可能でない(SC3，SC4 のみ)≪OS509≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪OS093≫
		E_OS_PARAM_POINTER	ポインタ引数が NULL≪OS566≫
		E_OS_ILLEGAL_ADDRESS	Value に不正なアドレスが指定された(SC3，SC4 のみ)≪OS051≫
		E_OS_CORE	起動していないコアへの呼出し≪OSa120≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC1，SC2，SC3，SC4【OS532】	
機能			
本システムサービスは，カウンタの現在のカウント値を取得する．			

機能仕様

GetCounterValue は，CounterID で指定されたカウンタの現在のティックを Value で指定された領域に格納する【OS377】.

GetCounterValue 呼出し時に指定された CounterID が不正であった場合, E_OS_ID を返す【OS376】.

ATK2 では，操作中のカウンタに対して，GetCounterValue を発行した場合，操作中の要因となった IncrementCounter によるティックのインクリメントが行われた後の値が Value に格納される

【IOS003】.

使用上の注意

CounterID で指定されたカウンタがハードウェアカウンタであった場合，タイマの実時間を返し，カウンタがソフトウェアカウンタであった場合，現在のティックを返す【OS531】.

マルチコア対応 OS における仕様

GetCounterValue は，コアを跨いで呼び出すことができる【OSa142】.

3.9.27 GetElapsedValue

C 言語 I/F		StatusType GetElapsedValue(CounterType CounterID, TickRefType Value, TickRefType ElapsedValue)【OS392】	
パラメータ[in]		CounterID	カウンタ ID
パラメータ[in/out]		Value	前回のティックが格納され，現在のティックを上書きする領域へのポインタ
パラメータ[out]		ElapsedValue	ティックの差分を格納する領域へのポインタ
返り値	標準エラー	E_OK	差分取得成功≪COS3104≫
	拡張エラー	E_OS_ID	CounterID が不正≪OS381≫
		E_OS_VALUE	Value がカウンタのカウンタ最大値を超えている≪OS391≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_ACCESS	アクセスが許可されていないカウンタが指定された(SC3，SC4 のみ)≪OS056≫
			カウンタが所属する OSAP が利用可能でない(SC3，SC4 のみ)≪OS509≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪OS093≫
		E_OS_PARAM_POINTER	ポインタ引数が NULL≪OS566≫
		E_OS_ILLEGAL_ADDRESS	Value または ElapsedValue に不正なアドレスが指定された(SC3，SC4 のみ)≪OS051≫
		E_OS_CORE	起動していないコアへの呼出し≪OSa120≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC1，SC2，SC3，SC4【OS534】	
機能			
本システムサービスは，指定されたティックと現在のティックの差分を取得する．			

機能仕様

GetElapsedValue は，Value で指定されたティックから現在のティックの差分を ElapsedValue に格納する【OS382】．また，現在のティックを Value に格納する【OS460】.

GetElapsedValue 呼出し時に指定された CounterID が不正であった場合, E_OS_ID を返す【OS381】. Value が指定されたカウンタの最大値より大きい場合，E_OS_VALUE を返す【OS391】.

ATK2 では，操作中のカウンタに対して，GetElapsedValue を発行した場合，操作中の要因となった IncrementCounter によるティックのインクリメントが行われた後のティックが Value に格納され，インクリメントが行われた後におけるティックの差分が ElapsedValue に格納される【IOS004】.

使用上の注意

GetElapsedValue は，Value に設定する前回のティックを取得した時点から，対象カウンタが，カウンタの最大値(OsCounterMaxAllowedValue)の回数より多くインクリメントされていた場合，ElapsedValue に格納される値は不正となる【OS533】.

マルチコア対応 OS における仕様

GetElapsedValue は，コアを跨いで呼び出すことができる【OSa143】.

3.9.28 GetAlarmBase

C 言語 I/F		StatusType GetAlarmBase(AlarmType AlarmID, AlarmBaseRefType Info)【COS3614】	
パラメータ[in]		AlarmID	アラーム ID
パラメータ[in/out]		－	
パラメータ[out]		Info	アラーム情報を格納する領域へのポインタ
返り値	標準エラー	E_OK	アラーム情報取得成功≪COS3104≫
	拡張エラー	E_OS_ID	AlarmID が不正≪COS3621≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_ACCESS	アクセスが許可されていないアラームが指定された(SC3，SC4 のみ)≪OS056≫
			アラームが所属する OSAP が利用可能でない(SC3，SC4 のみ)≪OS509≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪OS093≫
		E_OS_PARAM_POINTER	ポインタ引数が NULL≪OS566≫
		E_OS_ILLEGAL_ADDRESS	Info に不正なアドレスが指定された(SC3，SC4 のみ)≪OS051≫
		E_OS_CORE	起動していないコアへの呼出し≪OSa120≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC1，SC2，SC3，SC4≪NOS0319≫	
機能			
本システムサービスは，AlarmID で指定されたアラームの情報を取得する.			

機能仕様

GetAlarmBase は，AlarmID で指定されたアラーム情報を取得する【COS3617】．アラーム情報は，Info で示す構造体(AlarmBaseType)に格納される【COS3618】.

GetAlarmBase 呼出し時に指定された AlarmID が不正であった場合, E_OS_ID を返す【COS3621】.

マルチコア対応 OS における仕様

GetAlarmBase は，コアを跨いで呼び出すことができる【OS639】.

3.9.29 GetAlarm

C 言語 I/F		StatusType GetAlarm(AlarmType AlarmID, TickRefType Tick)【COS3623】	
パラメータ[in]		AlarmID	アラーム ID
パラメータ[in/out]		－	
パラメータ[out]		Tick	アラーム満了までのティック数を格納する領域へのポインタ
返り値	標準エラー	E_OK	アラーム満了までのティック数取得成功≪COS3104≫
		E_OS_NOFUNC	アラームがセットされていない(W)≪COS3631≫
	拡張エラー	E_OS_ID	AlarmID が不正≪COS3632≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_ACCESS	アクセスが許可されていないアラームが指定された(SC3，SC4 のみ)≪OS056≫
			アラームが所属する OSAP が利用可能でない(SC3，SC4 のみ)≪OS509≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪OS093≫
		E_OS_PARAM_POINTER	ポインタ引数が NULL≪OS566≫
		E_OS_ILLEGAL_ADDRESS	Tick に不正なアドレスが指定された(SC3，SC4 のみ)≪OS051≫
		E_OS_CORE	起動していないコアへの呼出し≪OSa120≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC1，SC2，SC3，SC4≪NOS0319≫	
機能			
本システムサービスは, AlarmID で指定されたアラームが満了するまでのティック数を取得する.			

機能仕様

GetAlarm は，AlarmID で指定されたアラームが満了するまでのティック数を取得し，Tick の領域に格納する【COS3626】．AlarmID が不正な場合，Tick の領域には何も格納されない【COS3628】.

GetAlarm 呼出し時に指定された AlarmID が不正であった場合，E_OS_ID を返す【COS3632】. AlarmID で指定されたアラームの動作がまだセットされていない場合，E_OS_NOFUNC を返す【COS3631】.

ATK2 では，操作中のカウンタに接続されたアラームに対して，GetAlarm を発行した場合，操作中の要因となった IncrementCounter によるティックのインクリメントが行われた後における，アラームが満了するまでのティック数が Tick に格納される【IOS005】.

マルチコア対応 OS における仕様

GetAlarm は，コアを跨いで呼び出すことができる【OS640】.

3.9.30 SetRelAlarm

<table>
<tr><td colspan="2">C 言語 I/F</td><td colspan="2">StatusType SetRelAlarm(AlarmType AlarmID,
TickType increment, TickType cycle)【COS3634】</td></tr>
<tr><td colspan="2" rowspan="3">パラメータ[in]</td><td>AlarmID</td><td>アラーム ID</td></tr>
<tr><td>increment</td><td>アラーム満了までのティックの相対値</td></tr>
<tr><td>cycle</td><td>アラーム満了周期(単発アラームの場合は 0 を指定する)</td></tr>
<tr><td colspan="2">パラメータ[in/out]</td><td colspan="2">－</td></tr>
<tr><td colspan="2">パラメータ[out]</td><td colspan="2">－</td></tr>
<tr><td rowspan="11">返り値</td><td rowspan="2">標準エラー</td><td>E_OK</td><td>アラームセット成功≪COS3104≫</td></tr>
<tr><td>E_OS_STATE</td><td>アラームが既にセットされている(W)≪COS3648≫</td></tr>
<tr><td rowspan="9">拡張エラー</td><td>E_OS_ID</td><td>AlarmID が不正≪COS3649≫</td></tr>
<tr><td rowspan="3">E_OS_VALUE</td><td>increment がアラームに関連付けられたカウンタの最大値より大きいか，0 より小さい≪COS3650≫</td></tr>
<tr><td>increment が 0 である≪NOS0405≫</td></tr>
<tr><td>cycle に指定された値が，カウンタの最小周期より小さい(0 を除く)か，アラームに関連付けられたカウンタの最大値より大きい場合≪COS3651≫</td></tr>
<tr><td rowspan="2">E_OS_CALLEVEL</td><td>不正な処理単位からの呼出し≪OS088≫</td></tr>
<tr><td>C1ISR からの呼出し≪IOS151≫</td></tr>
<tr><td rowspan="2">E_OS_ACCESS</td><td>アクセスが許可されていないアラームが指定された(SC3，SC4 のみ)≪OS056≫</td></tr>
<tr><td>アラームが所属する OSAP が利用可能でない(SC3，SC4 のみ)≪OS509≫</td></tr>
<tr><td>E_OS_DISABLEDINT</td><td>割込み禁止状態からの呼出し≪OS093≫</td></tr>
<tr><td>E_OS_CORE</td><td>起動していないコアへの呼出し≪OSa120≫</td></tr>
<tr><td colspan="2">コンフォーマンスクラス</td><td colspan="2">BCC1，BCC2，ECC1，ECC2(アラーム満了時のアクションが SetEvent の場合は ECC1，ECC2 のみ)</td></tr>
<tr><td colspan="2">スケーラビリティクラス</td><td colspan="2">SC1，SC2，SC3，SC4≪NOS0319≫</td></tr>
<tr><td colspan="4">機能</td></tr>
<tr><td colspan="4">本システムサービスは，AlarmID で指定されたアラームを，現在のティックから increment で指定された相対時刻が経過した後に満了するよう設定する．初回の満了後，cycle が 0 でない場合は，cycle の周期でアラームを満了させる．</td></tr>
</table>

機能仕様

SetRelAlarm は，AlarmID で指定されたアラームが現在のティックから increment で指定された相対時刻が経過した後に満了するよう設定する【COS3639】. 初回の満了後, cycle が 0 でない場合は, cycle の周期でアラームを満了させる【COS3642】〔COS0922〕.

SetRelAlarm 呼出し時に指定された AlarmID が不正であった場合，E_OS_ID を返す【COS3649】. increment で指定された値がアラームに関連付けられたカウンタの最大値より大きいか，0 より小さい場合, E_OS_VALUE を返す【COS3650】. increment で指定された値が 0 であった場合, E_OS_VALUE を返す【NOS0405】. cycle に指定された値が，カウンタの最小周期より小さい(0 を除く)か，アラームに関連付けられたカウンタの最大値より大きい場合，E_OS_VALUE を返す【COS3651】.

AlarmID で指定されたアラームが, 既にセットされている場合, E_OS_STATE を返す【COS3648】.

使用上の注意

increment が非常に小さい場合，SetRelAlarm の終了前にアラームが満了する可能性がある【COS3641】. アラームの動作パラメータを変更する場合，再設定する前に対象のアラームを停止する必要がある【COS3644】.

マルチコア対応 OS における仕様

SetRelAlarm は，コアを跨いで呼び出すことができる【OS636】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，increment で指定された値が 0 であった場合に，E_OS_VALUE を返すエラーを標準エラーと規定している【OS304】. しかし，OSEK 仕様では，E_OS_VALUE を返すエラーは拡張エラーと規定されているため，本仕様では，E_OS_VALUE を返すエラーをすべて拡張エラーと規定した≪NOS0405≫.

ATK2 では，操作中のカウンタに接続されたアラームに対して，SetRelAlarm を発行した場合，操作中の要因となった IncrementCounter によるティックのインクリメントが行われた後のティックから increment で指定された相対時刻が経過した後に満了するよう設定する【IOS006】.

3.9.31 SetAbsAlarm

C 言語 I/F		StatusType SetAbsAlarm(AlarmType AlarmID, TickType start, TickType cycle)【COS3653】	
パラメータ[in]		AlarmID	アラーム ID
		start	アラーム満了のティックの絶対値
		cycle	アラーム満了周期 (単発アラームの場合は 0 を指定する)
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	E_OK	アラームセット成功≪COS3104≫
		E_OS_STATE	アラームが既にセットされている(W)≪COS3667≫
	拡張エラー	E_OS_ID	AlarmID が不正≪COS3668≫
		E_OS_VALUE	start がアラームに関連付けられたカウンタの最大値より大きいか，0 より小さい≪COS3669≫
			cycle に指定された値が，カウンタの最小周期より小さい(0 を除く)か，アラームに関連付けられたカウンタの最大値より大きい場合≪COS3670≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_ACCESS	アクセスが許可されていないアラームが指定された(SC3，SC4 のみ)≪OS056≫
			アラームが所属する OSAP が利用可能でない(SC3，SC4 のみ)≪OS509≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪OS093≫
		E_OS_CORE	起動していないコアへの呼出し≪OSa120≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2 (アラーム満了時のアクションが SetEvent の場合は ECC1，ECC2 のみ)	
スケーラビリティクラス		SC1，SC2，SC3，SC4≪NOS0319≫	
機能			
本システムサービスは，AlarmID で指定されたアラームを，start で指定された絶対時刻に達した際に満了するよう設定する．初回の満了後，cycle が 0 でない場合は，cycle の周期でアラームを満了させる．			

機能仕様

SetAbsAlarm は，AlarmID で指定されたアラームが start で指定された絶対時刻に達した際に満了

するよう設定する【COS3658】．初回の満了後，cycle が 0 でない場合は cycle の周期でアラームを満了させる【COS3661】〔COS0922〕．

SetAbsAlarm 呼出し時に指定された AlarmID が不正であった場合，E_OS_ID を返す【COS3668】．start で指定された値がアラームに関連付けられたカウンタの最大値より大きいか，0 より小さい場合，E_OS_VALUE を返す【COS3669】．cycle に指定された値が，カウンタの最小周期より小さい(0 を除く)か，アラームに関連付けられたカウンタの最大値より大きい場合，E_OS_VALUE を返す【COS3670】．
AlarmID で指定されたアラームが，既にセットされている場合，E_OS_STATE を返す【COS3667】．

使用上の注意点

start と，接続されているカウンタの現在値の差が非常に小さい場合，SetAbsAlarm の終了前にアラームが満了する可能性がある【COS3659】．また，start が，接続されているカウンタの現在のティックを過ぎていた場合，アラームが満了するのはカウンタがラップアラウンドした後となる【COS3660】．アラームの動作パラメータを変更する場合，再設定する前に対象のアラームを停止する必要がある【COS3663】．

ATK2 では，start で指定された値と接続されているカウンタの現在値が同じ場合，即座に満了せず，次の周期で start に達した際に満了する【IOS029】．

マルチコア対応 OS における仕様

SetAbsAlarm は，コアを跨いで呼び出すことができる【OS637】．

3.9.32 CancelAlarm

C 言語 I/F		StatusType CancelAlarm(AlarmType AlarmID)【COS3672】	
パラメータ[in]		AlarmID	アラーム ID
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	E_OK	アラーム停止成功≪COS3104≫
		E_OS_NOFUNC	アラームが既に停止されている(W) ≪COS3678≫
	拡張エラー	E_OS_ID	AlarmID が不正≪COS3679≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_ACCESS	アクセスが許可されていないアラームが指定された(SC3，SC4 のみ)≪OS056≫
			アラームが所属する OSAP が利用可能でない(SC3，SC4 のみ)≪OS509≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪OS093≫
		E_OS_CORE	起動していないコアへの呼出し≪OSa120≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC1，SC2，SC3，SC4≪NOS0319≫	
機能			
本システムサービスは，AlarmID で指定されたアラームを停止する.			

機能仕様

CancelAlarm は，AlarmID で指定されたアラームを停止する【COS3675】.

CancelAlarm 呼出し時に指定された AlarmID が不正であった場合，E_OS_ID を返す【COS3679】.

AlarmID で指定されたアラームが, 既に停止されている場合, E_OS_NOFUNC を返す【COS3678】.

マルチコア対応 OS における仕様

CancelAlarm は，コアを跨いで呼び出すことができる【OS638】.

3.9.33 StartScheduleTableRel

C 言語 I/F		StatusType StartScheduleTableRel (ScheduleTableType ScheduleTableID, TickType Offset)【OS347】	
パラメータ[in]		ScheduleTableID	スケジュールテーブル ID
		Offset	起動オフセット
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	E_OK	スケジュールテーブル起動成功≪COS3104≫
		E_OS_STATE	スケジュールテーブルが既に開始されている≪OS277≫
	拡張エラー	E_OS_ID	ScheduleTableID が不正≪OS275≫
			ScheduleTableID が暗黙同期スケジュールテーブルを指定している≪OS452≫
		E_OS_VALUE	Offset が 0≪OS332≫
			またはOffsetが(OsCounterMaxAllowedValue－初期オフセット)より大きい≪OS276≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_ACCESS	アクセスが許可されていないスケジュールテーブルが指定された(SC3，SC4 のみ)≪OS056≫
			スケジュールテーブルが所属する OSAP が利用可能でない(SC3，SC4 のみ)≪OS509≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪OS093≫
		E_OS_CORE	起動していないコアへの呼出し≪OSa120≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC1，SC2，SC3，SC4【OS521】	
機能			
本システムサービスは，現在のティックから Offset で指定された相対時刻が経過した後にスケジュールテーブルを開始する．			

機能仕様

StartScheduleTableRel は, ScheduleTableID で指定された SCHEDULETABLE_STOPPED 状態のスケジュールテーブルを開始する．先頭満了点はスケジュールテーブルに接続されたカウンタの現在のティック値から(Offset＋スケジュールテーブルの初期オフセット)を経過した時である．システムサービスからリターンする前に，スケジュールテーブル状態を SCHEDULETABLE_RUNNING 状態へ遷

移させる【OS278】.

StartScheduleTableRel の呼出し時に指定された ScheduleTableID が不正であった場合，E_OS_ID を返す【OS275】．また，ScheduleTableID で指定されたスケジュールテーブルが暗黙同期スケジュールテーブルであった場合，E_OS_ID を返す【OS452】.

StartScheduleTableRel の呼出し時に指定された Offset が 0 であった場合，E_OS_VALUE を返す【OS332】．また，StartScheduleTableRel の呼出し時に指定された Offset が，接続されているカウンタの OsCounterMaxAllowedValue からスケジュールテーブルの初期オフセットを引いたものより大きい場合，E_OS_VALUE を返す【OS276】.

StartScheduleTableRel の呼出し時に，スケジュールテーブル状態が SCHEDULETABLE_STOPPED 以外の場合，E_OS_STATE を返す【OS277】.

マルチコア対応 OS における仕様

StartScheduleTableRel は，コアを跨いで呼び出すことができる【OS645】.

3.9.34 StartScheduleTableAbs

C 言語 I/F		StatusType StartScheduleTableAbs (ScheduleTableType ScheduleTableID, TickType Start)【OS358】	
パラメータ[in]		ScheduleTableID	スケジュールテーブル ID
		Start	開始時刻
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	E_OK	スケジュールテーブル起動成功≪COS3104≫
		E_OS_STATE	スケジュールテーブル状態が SCHEDULETABLE_STOPPED 以外≪OS350≫
	拡張エラー	E_OS_ID	SheduleTableID が不正≪OS348≫
		E_OS_VALUE	Start が接続しているカウンタのカウンタ最大値より大きい≪OS349≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_ACCESS	アクセスが許可されていないスケジュールテーブルが指定された(SC3，SC4 のみ)≪OS056≫
			スケジュールテーブルが所属する OSAP が利用可能でない(SC3，SC4 のみ)≪OS509≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪OS093≫
		E_OS_CORE	起動していないコアへの呼出し≪OSa120≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC1，SC2，SC3，SC4【OS522】	
機能			
本システムサービスは，Start で指定された絶対時刻に達した際にスケジュールテーブルを開始する．			

機能仕様

StartScheduleTableAbs は，ScheduleTableID で指定された SCHEDULETABLE_STOPPED 状態のスケジュールテーブルを開始する．開始タイミングは接続されたカウンタが次に Start で指定された時刻と一致した時である．スケジュールテーブルの同期方式が明示同期か同期なしの場合は SCHEDULETABLE_RUNNING に遷移し，暗黙同期の場合は SCHEDULETABLE_RUNNING_AND_SYNCHRONOUS へ遷移する．先頭満了点はスケジュールテーブルに接続されたカウンタが(Start＋スケジュールテーブルの初期オフセット)に達した時である

【OS351】.

StartScheduleTableAbs の呼出し時に指定された ScheduleTableID が不正であった場合，E_OS_ID を返す【OS348】.

StartScheduleTableAbs の呼出し時に指定された Start が，接続されているカウンタの OsCounterMaxAllowedValue より大きい場合，E_OS_VALUE を返す【OS349】.

StartScheduleTableAbs の呼出し時にスケジュールテーブル状態が SCHEDULETABLE_STOPPED 以外の場合，E_OS_STATE を返す【OS350】.

ATK2 では, Start で指定された値と接続されているカウンタの現在値が同じ場合, 即座に開始せず, 次の周期で Start に達した際に開始する【IOS174】.

マルチコア対応 OS における仕様

StartScheduleTableAbs は，コアを跨いで呼び出すことができる【OS644】.

3.9.35 StopScheduleTable

C 言語 I/F		StatusType StopScheduleTable (ScheduleTableType ScheduleTableID)【OS006】	
パラメータ[in]		ScheduleTableID	スケジュールテーブル ID
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	E_OK	スケジュールテーブル停止成功≪COS3104≫
		E_OS_NOFUNC	スケジュールテーブル状態が SCHEDULETABLE_STOPPED≪OS280≫
	拡張エラー	E_OS_ID	SheduleTableID が不正≪OS279≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_ACCESS	アクセスが許可されていないスケジュールテーブルが指定された(SC3，SC4 のみ)≪OS056≫
			スケジュールテーブルが所属する OSAP が利用可能でない(SC3，SC4 のみ)≪OS509≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪OS093≫
		E_OS_CORE	起動していないコアへの呼出し≪OSa120≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC1，SC2，SC3，SC4【OS523】	
機能			
本システムサービスは，ScheduleTableID で指定されたスケジュールテーブルを停止する.			

機能仕様

StopScheduleTable は，ScheduleTableID で指定されたスケジュールテーブルを停止し，SCHEDULETABLE_STOPPED 状態へ遷移させる【OS281】.

StopScheduleTable の呼出し時に指定された ScheduleTableID が不正であった場合，E_OS_ID を返す【OS279】.

StopScheduleTable の呼出し時に，スケジュールテーブル状態が SCHEDULETABLE_STOPPED の場合，E_OS_NOFUNC を返す【OS280】.

マルチコア対応 OS における仕様

StopScheduleTable は，コアを跨いで呼び出すことができる【OS646】.

3.9.36 NextScheduleTable

<table>
<tr><td colspan="2">C 言語 I/F</td><td colspan="2">StatusType NextScheduleTable
(ScheduleTableType ScheduleTableID_From,
ScheduleTableType ScheduleTableID_To)【OS191】</td></tr>
<tr><td colspan="2" rowspan="2">パラメータ[in]</td><td>ScheduleTableID_From</td><td>切換え前スケジュールテーブル ID</td></tr>
<tr><td>ScheduleTableID_To</td><td>切換え後スケジュールテーブル ID</td></tr>
<tr><td colspan="2">パラメータ[in/out]</td><td colspan="2">－</td></tr>
<tr><td colspan="2">パラメータ[out]</td><td colspan="2">－</td></tr>
<tr><td rowspan="12">返り値</td><td rowspan="3">標準エラー</td><td>E_OK</td><td>スケジュールテーブル切換え成功≪COS3104≫</td></tr>
<tr><td>E_OS_NOFUNC</td><td>ScheduleTableID_From のスケジュールテーブル状態が SCHEDULETABLE_STOPPED か，SCHEDULETABLE_NEXT≪OS283≫</td></tr>
<tr><td>E_OS_STATE</td><td>ScheduleTableID_To のスケジュールテーブル状態が SCHEDULETABLE_STOPPED でない≪OS309≫</td></tr>
<tr><td rowspan="9">拡張エラー</td><td rowspan="3">E_OS_ID</td><td>ScheduleTableID_From または ScheduleTableID_To が不正≪OS282≫</td></tr>
<tr><td>2 つのスケジュールテーブルが同一のカウンタに接続されていない≪OS330≫</td></tr>
<tr><td>2 つのスケジュールテーブルの同期方式が等しくない≪OS484≫</td></tr>
<tr><td rowspan="2">E_OS_CALLEVEL</td><td>不正な処理単位からの呼出し≪OS088≫</td></tr>
<tr><td>C1ISR からの呼出し≪IOS151≫</td></tr>
<tr><td rowspan="2">E_OS_ACCESS</td><td>アクセスが許可されていないスケジュールテーブルが指定された(SC3，SC4 のみ)≪OS056≫</td></tr>
<tr><td>スケジュールテーブルが所属する OSAP が利用可能でない(SC3，SC4 のみ)≪OS509≫</td></tr>
<tr><td>E_OS_DISABLEDINT</td><td>割込み禁止状態からの呼出し≪OS093≫</td></tr>
<tr><td>E_OS_CORE</td><td>起動していないコアへの呼出し≪OSa120≫</td></tr>
<tr><td colspan="2">コンフォーマンスクラス</td><td colspan="2">BCC1，BCC2，ECC1，ECC2</td></tr>
<tr><td colspan="2">スケーラビリティクラス</td><td colspan="2">SC1，SC2，SC3，SC4【OS524】</td></tr>
<tr><td colspan="4">機能</td></tr>
<tr><td colspan="4">本システムサービスは，動作中のスケジュールテーブルから，他のスケジュールテーブルへの切換えを行う．</td></tr>
</table>

機能仕様

NextScheduleTable は，ScheduleTableID_To で指定されたスケジュールテーブルを ScheduleTableID_From で指定されたスケジュールテーブルの最終遅延経過後に開始する．ScheduleTableID_To で指定されたスケジュールテーブルの先頭満了点は，ScheduleTableID_From で指定されたスケジュールテーブルの最終満了点処理後，ScheduleTableID_From の最終遅延＋ScheduleTableID_To の初期オフセット経過後である【OS284】.

NextScheduleTable 呼出し時に ScheduleTableID_From で指定されたスケジュールテーブルが，前の NextScheduleTable によって既に次のスケジュールテーブルを指定されまだスケジュールテーブルが切り換わっていない場合，前の NextScheduleTable 呼出し時に ScheduleTableID_To で指定されたスケジュールテーブルの状態を SCHEDULETABLE_STOPPED に変更し現在の呼出し時に ScheduleTableID_To で指定されたスケジュールテーブルを次に起動するスケジュールテーブルとする【OS324】.

NextScheduleTable の呼出し時に, ScheduleTableID_From と ScheduleTableID_To で指定された同期方式が明示同期であり，かつ，OS モジュールが既に ScheduleTableID_From と同期していた場合，ScheduleTableID_To が開始した後も同期状態を保たなければならない【OS505】.

NextScheduleTable 終了後，スケジュールテーブルが切り換わる前に ScheduleTableID_From で指定されたスケジュールテーブルが停止された場合，ScheduleTableID_To で指定されたスケジュールテーブルは起動せず，SCHEDULETABLE_STOPPED 状態となる【OS453】.

NextScheduleTable の呼出し時に指定された ScheduleTableID_From と ScheduleTableID_To のうち，どちらかが不正であった場合，E_OS_ID を返す【OS282】．NextScheduleTable の呼出し時に指定された 2 つのスケジュールテーブルが異なるカウンタに接続されている場合, E_OS_ID を返す【OS330】.
NextScheduleTable の呼出し時に ScheduleTableID_To で指定されたスケジュールテーブルの同期方式が ScheduleTableID_From で指定されたスケジュールテーブルの同期方式と等しくない場合，E_OS_ID を返す【OS484】.

NextScheduleTable の呼出し時に, ScheduleTableID_From で指定されたスケジュールテーブル状態が SCHEDULETABLE_STOPPED か，SCHEDULETABLE_NEXT の場合，E_OS_NOFUNC を返す【OS283】．NextScheduleTable の呼出し時に ScheduleTableID_To で指定されたスケジュールテーブル状態が SCHEDULETABLE_STOPPED でない場合，E_OS_STATE を返す【OS309】.

ScheduleTableID_From で指定されたスケジュールテーブルは，周期動作，単発動作に関わらず，ScheduleTableID_To で指定されたスケジュールテーブルへの切換えと同時に停止する【NOS0661】.

ただし，ScheduleTableID_From で指定されたスケジュールテーブルが周期動作で，スケジュールテーブルが切り換わる前に ScheduleTableID_To で指定されたスケジュールテーブルが停止された場合，ScheduleTableID_From で指定されたスケジュールテーブルは繰り返し動作する【NOS0662】.

マルチコア対応 OS における仕様

NextScheduleTable は，コアを跨いで呼び出すことができる【NOS0974】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，NextScheduleTable は，コアを跨いで呼び出すことができないと規定されている【OSa144】．しかし，NextScheduleTable をコアを跨いで呼び出すことによる不都合が想定できないため，本仕様では呼び出すことができると規定した≪NOS0974≫.

3.9.37 StartScheduleTableSynchron

C 言語 I/F		StatusType StartScheduleTableSynchron (ScheduleTableType ScheduleTableID)【OS201】	
パラメータ[in]		ScheduleTableID	スケジュールテーブル ID
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	E_OK	スケジュールテーブル同期開始成功≪COS3104≫
		E_OS_STATE	スケジュールテーブル状態が SCHEDULETABLE_STOPPED 以外≪OS388≫
	拡張エラー	E_OS_ID	スケジュールテーブル ID が不正，または指定されたスケジュールテーブルが明示同期スケジュールテーブルでない≪OS387≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_ACCESS	アクセスが許可されていないスケジュールテーブルが指定された(SC4 のみ)≪OS056≫
			スケジュールテーブルが所属する OSAP が利用可能でない(SC4 のみ)≪OS509≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪OS093≫
		E_OS_CORE	コアを跨いだ呼出し≪OS589≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC2，SC4【OS525】	
機能			
本システムサービスは，明示同期スケジュールテーブルを開始する．			

機能仕様

StartScheduleTableSynchron は，ScheduleTableID で指定されたスケジュールテーブルを SCHEDULETABLE_WAITING 状態に設定して動作を開始する．先頭満了点はスケジュールテーブルに接続されたカウンタが"(Duration-SyncValue)+InitialOffset ticks"を経過した時である【OS389】.

- DuratioNOS0ScheduleTableDuration(スケジュールテーブル周期)
- SyncValue：SyncScheduleTable の Value で渡された現在時刻
- InitialOffset　：スケジュールテーブルの初期オフセット

StartScheduleTableSynchron の呼出し時に指定された ScheduleTableID が不正であった場合，また

は，ScheduleTableID で指定されたスケジュールテーブルが明示同期でなかった場合，E_OS_ID を返す【OS387】. スケジュールテーブル状態が SCHEDULETABLE_STOPPED 以外の場合, E_OS_STATE を返す【OS388】.

マルチコア対応 OS における仕様

StartScheduleTableSynchron を，コアを跨いで呼び出した場合，E_OS_CORE を返す≪OS589≫.

3.9.38 SyncScheduleTable

C 言語 I/F		StatusType SyncScheduleTable (ScheduleTableType ScheduleTableID, TickType Value)【OS199】	
パラメータ[in]		ScheduleTableID	スケジュールテーブル ID
		Value	現在の同期カウンタの時刻
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	E_OK	時刻通知成功≪COS3104≫
		E_OS_STATE	スケジュールテーブル状態が SCHEDULETABLE_STOPPED か SCHEDULETABLE_NEXT≪OS456≫
	拡張エラー	E_OS_ID	スケジュールテーブル ID が不正，または指定されたスケジュールテーブルが明示同期スケジュールテーブルでない≪OS454≫
		E_OS_VALUE	Value がスケジュールテーブル周期より大きい≪OS455≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_ACCESS	アクセスが許可されていないスケジュールテーブルが指定された(SC4 のみ)≪OS056≫
			スケジュールテーブルが所属する OSAP が利用可能でない(SC4 のみ)≪OS509≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪OS093≫
		E_OS_CORE	コアを跨いだ呼出し≪OS589≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC2，SC4【OS526】	
機能			
本システムサービスは，スケジュールテーブルに同期カウンタの値を通知し，同期を開始する．			

機能仕様

SyncScheduleTable は，現在の同期カウンタの値をスケジュールテーブルに通知し，スケジュールテーブルは同期カウンタに同期して動作を行う【OS457】.

SyncScheduleTable の呼出し時に指定された ScheduleTableID が不正であった場合，また，ScheduleTableID で指定されたスケジュールテーブルが明示同期でなかった場合，E_OS_ID を返す【OS454】. SyncScheduleTable の呼出し時に指定された Value が OsScheduleTableDuration より大

きい場合，E_OS_VALUE を返す【OS455】.

スケジュールテーブル状態が SCHEDULETABLE_STOPPED か SCHEDULETABLE_NEXT の場合，E_OS_STATE を返す【OS456】.

マルチコア対応 OS における仕様

SyncScheduleTable を，コアを跨いで呼び出した場合，E_OS_CORE を返す≪OS589≫.

3.9.39 SetScheduleTableAsync

C 言語 I/F		StatusType SetScheduleTableAsync (ScheduleTableType ScheduleTableID)【OS422】	
パラメータ[in]		ScheduleTableID	スケジュールテーブル ID
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	E_OK	非同期設定成功≪COS3104≫
	拡張エラー	E_OS_ID	スケジュールテーブル ID が不正，または指定されたスケジュールテーブルが明示同期スケジュールテーブルでない≪OS458≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_ACCESS	アクセスが許可されていないスケジュールテーブルが指定された(SC4 のみ)≪OS056≫
			スケジュールテーブルが所属する OSAP が利用可能でない(SC4 のみ)≪OS509≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪OS093≫
		E_OS_CORE	コアを跨いだ呼出し≪OS589≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC2，SC4【OS527】	
機能			
本システムサービスは，スケジュールテーブルの同期を停止する．			

機能仕様

スケジュールテーブルが動作中に SetScheduleTableAsync が呼び出された場合, SyncScheduleTable が呼ばれるまで同期処理を停止する【OS362】．スケジュールテーブルが動作中に SetScheduleTableAsync が呼び出された場合，OS モジュールは満了点の実行を継続する【OS323】．

ScheduleTableID で指定されたスケジュールテーブルの状態が SCHEDULETABLE_RUNNING_AND_SYNCHRONOUS であれば，状態を SCHEDULTABLE_RUNNING 状態に遷移させる【OS300】．

SetScheduleTableAsync の呼出し時に指定された ScheduleTableID が不正であった場合，また，ScheduleTableID で指定されたスケジュールテーブルが明示同期でなかった場合，E_OS_ID を返す【OS458】．ScheduleTableID で指定されたスケジュールテーブル状態が SCHEDULETABLE_STOPPED，SCHEDULETABLE_NEXT，SCHEDULETABLE_WAITING であった場合，E_OS_STATE を返す【OS483】．

マルチコア対応 OS における仕様

SetScheduleTableAsync を，コアを跨いで呼び出した場合，E_OS_CORE を返す≪OS589≫.

3.9.40 GetScheduleTableStatus

C 言語 I/F		StatusType GetScheduleTableStatus (ScheduleTableType ScheduleTableID, ScheduleTableStatusRefType ScheduleStatus)【OS227】	
パラメータ[in]		ScheduleTableID	スケジュールテーブル ID
パラメータ[in/out]		－	
パラメータ[out]		ScheduleStatus	スケジュールテーブル状態を格納する領域へのポインタ
返り値	標準エラー	E_OK	スケジュールテーブル状態取得成功≪COS3104≫
	拡張エラー	E_OS_ID	ScheduleTableID が不正≪OS293≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_ACCESS	アクセスが許可されていないスケジュールテーブルが指定された(SC3，SC4 のみ)≪OS056≫
			スケジュールテーブルが所属する OSAP が利用可能でない(SC3，SC4 のみ)≪OS509≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪OS093≫
		E_OS_PARAM_POINTER	ポインタ引数が NULL≪OS566≫
		E_OS_ILLEGAL_ADDRESS	ScheduleStatus に不正なアドレスが指定された(SC3，SC4 のみ)≪OS051≫
		E_OS_CORE	起動していないコアへの呼出し≪OSa120≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC1，SC2，SC3，SC4【OS528】	
機能			
本システムサービスは，スケジュールテーブルの状態を取得する．			

機能仕様

GetScheduleTableStatus は，ScheduleTableID で指定されたスケジュールテーブルの状態を ScheduleStatus に格納する．格納される状態を以下に示す．

- スケジュールテーブルが停止状態の場合：SCHEDULETABLE_STOPPED【OS289】
- スケジュールテーブルが切換え待ち状態の場合：SCHEDULETABLE_NEXT【OS353】
- スケジュールテーブルが同期待ち状態の場合：SCHEDULETABLE_WAITING【OS354】
- スケジュールテーブルが同期動作状態の場合：
  SCHEDULETABLE_RUNNING_AND_SYNCHRONOUS【OS290】

・　スケジュールテーブルが動作状態の場合：SCHEDULETABLE_RUNNING【OS291】

GetScheduleTableStatus の呼出し時に，ScheduleTableID が不正であった場合，E_OS_ID を返す【OS293】.

マルチコア対応 OS における仕様

GetScheduleTableStatus は，コアを跨いで呼び出すことができる【OS647】.

3.9.41 GetApplicationID

C 言語 I/F		ApplicationType GetApplicationID(void)【OS016】	
パラメータ[in]		－	
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値		現在動作中の OSAPID	OSAPID 取得成功≪OS261≫
		INVALID_OSAPPLICATION	どの OSAP も動作していない≪OS262≫
			不正な処理単位からの呼出し≪NOS1119≫
エラーフックに渡されるエラーコード	標準エラー	－	
	拡張エラー	E_OS_CALLEVEL	不正な処理単位からの呼出し≪NOS0411≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC3，SC4【OS514】 マルチコア：SC1，SC2【NOS0975】	
機能			
本システムサービスは，現在動作中の OSAPID を取得する．			

機能仕様

GetApplicationID は，現在実行中のタスク，C2ISR，フックルーチンが所属する OSAPID を返す【OS261】．どの OSAP も動作していない場合，INVALID_OSAPPLICATION を返す【OS262】．GetApplicationID は，割込み禁止状態であっても，呼び出すことができる【NOS1149】．

不正な処理単位から GetApplicationID を呼び出した場合，INVALID_OSAPPLICATION を返す【NOS1119】．

マルチコア対応 OS における仕様

GetApplicationID を呼び出した処理単位が割付いているコアにおける，現在動作中の OSAPID を取得する【NOS0976】．GetApplicationID を呼び出した処理単位が割付いているコアに，どの OSAP も動作していない場合，INVALID_OSAPPLICATION を返す【NOS0977】．

3.9.42 CallTrustedFunction

<table>
<tr><td colspan="2">C 言語 I/F</td><td colspan="2">StatusType CallTrustedFunction(
TrustedFunctionIndexType FunctionIndex,
TrustedFunctionParameterRefType FunctionParams)
【OS097】</td></tr>
<tr><td colspan="2" rowspan="2">パラメータ[in]</td><td>FunctionIndex</td><td>信頼関数 ID</td></tr>
<tr><td>FunctionParams</td><td>信頼関数のパラメータへのポインタ，パラメータをとらない場合は NULL を渡す</td></tr>
<tr><td colspan="2">パラメータ[in/out]</td><td colspan="2">－</td></tr>
<tr><td colspan="2">パラメータ[out]</td><td colspan="2">－</td></tr>
<tr><td rowspan="7">返り値</td><td rowspan="4">標準エラー</td><td>E_OK</td><td>信頼関数呼出し成功≪COS3104≫</td></tr>
<tr><td>E_OS_STACKINSUFFICIENT</td><td>スタックの残り量が，信頼関数が使用するスタック量より少ない≪NOS0400≫</td></tr>
<tr><td>E_OS_SERVICEID</td><td>FunctionIndex が不正≪OS292≫≪OS100≫</td></tr>
<tr><td>ユーザ定義エラー</td><td>ユーザ定義信頼関数のエラーコード≪NOS0179≫</td></tr>
<tr><td rowspan="3">拡張エラー</td><td rowspan="2">E_OS_CALLEVEL</td><td>不正な処理単位からの呼出し≪OS088≫</td></tr>
<tr><td>C1ISR からの呼出し≪IOS151≫</td></tr>
<tr><td rowspan="2">E_OS_ACCESS</td><td>信頼関数が所属する OSAP が利用可能でない≪OS509≫</td></tr>
<tr><td>異なるコアに割付いている OSAP に所属する信頼関数の呼び出し≪OS623≫</td></tr>
<tr><td colspan="2">E_OS_DISABLEDINT</td><td>割込み禁止状態からの呼出し≪OS093≫</td></tr>
<tr><td colspan="2">コンフォーマンスクラス</td><td colspan="2">BCC1，BCC2，ECC1，ECC2</td></tr>
<tr><td colspan="2">スケーラビリティクラス</td><td colspan="2">SC3，SC4【OS516】</td></tr>
<tr><td colspan="4">機能</td></tr>
<tr><td colspan="4">本システムサービスは，FunctionIndex で指定された信頼関数を，プロセッサコアを特権モードに変更して呼び出す≪OS265≫.</td></tr>
</table>

機能仕様

CallTrustedFunction はプロセッサコアを特権モードに変更し，コンフィギュレーション時に定義した信頼関数のリストから FunctionIndex で指定された信頼関数を呼び出す. 信頼関数実行中はメモリ保護を無効とする【OS265】. ターゲットのハードウェアが特権モードでもメモリ保護を実現できる場合はモード変更を行う必要はないが，MPU の設定を変更する必要がある【NOS0280】.

CallTrustedFunction で呼び出した信頼関数は，信頼関数が所属する信頼 OSAP と同じアクセス権で実行する．しかし，信頼関数に対しては，CallTrustedFunction を呼び出したタスク，ISR が所属する OSAP のタイミング保護が適用される【NOS0325】. また，信頼関数に対しては，所属している信頼 OSAP のサービス保護が適用される【NOS0326】. ただし，信頼関数を実行している状態において，動作中とする OSAP は，信頼関数を呼び出した処理単位が所属する OSAP となる【NOS0395】.

信頼関数が C2ISR から呼ばれた場合, 信頼関数は呼び出された C2ISR と同じ割込み優先度で動作し，C2ISR で許可されたシステムサービスを使用することができる≪OS364≫.
信頼関数がタスクから呼ばれた場合，信頼関数は呼び出されたタスクと同じタスク優先度で動作し，タスクで許可されたシステムサービスを使用することができる≪OS365≫.

CallTrustedFunction 呼出し時に指定された FunctionIndex が不正な場合, E_OS_SERVICEID を返す【OS292】. FunctionIndex で指定された信頼関数が，信頼関数のリストに正しくコンフィギュレートされていない場合，OS は E_OS_SERVICEID を返す【OS100】.

CallTrustedFunction 呼出し時に，呼び出し元の処理単位が使用可能なスタックの残り量が，信頼関数が使用するスタック量より少ない場合， CallTrustedFunction は，信頼関数を呼び出さずに，E_OS_STACKINSUFFICIENT を返す≪NOS0400≫.

使用上の注意

CallTrustedFunction は StandardI/O などの標準インターフェースの呼び出しを行う際に呼び出されることを想定している．呼び出された信頼関数は渡されたパラメータのチェックを行わなければならない.
CallTrustedFunction の呼び出しにより，タイミング保護が無効とならないように注意する．信頼 OSAP でも，タイミング保護による保護違反時処理を実行される可能性がある．したがって，内部状態を持たない関数のみ CallTrustedFunction を呼び出すことを推奨する【OS312】.

マルチコア対応 OS における仕様

CallTrustedFunction を，コアを跨いで呼び出した場合，E_OS_ACCESS を返す【OS623】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，信頼関数の返り値は void 型と定義されていた≪OS312≫．本仕様では，信頼関数を StatusType 型とし，ユーザが定義した信頼関数の返り値を CallTrustedFunction の返り値とするよう規定した≪NOS0179≫.

AUTOSAR 仕様では，信頼関数には CallTrustedFunction を呼び出したタスク，ISR が所属する OSAP のサービス保護が適用されると定義されていた【OS266】．本仕様では，信頼関数には，信頼関数が所属している信頼 OSAP のサービス保護を適用するよう規定した≪NOS0325≫≪NOS0326≫．ただし，信頼関数を実行している状態において，動作中とする OSAP は，信頼関数を呼び出した処理単位が所属する OSAP となるよう規定した≪NOS0395≫.

AUTOSAR 仕様では，信頼関数実行中は，同じ OSAP に所属する処理単位への切り替えを行わないと規定されている【OS563】．しかし，同じ OSAP の処理単位への切り替えだけを禁止しても，他の OSAP の処理単位から OSAP を終了される可能性があり，優先度逆転が発生する可能性もあるので，本仕様では規定しない．この問題への対応策は，2.18.8.7 節を参照.

AUTOSAR 仕様では，タイミング保護に対する信頼関数の仕様が規定されている.

- 信頼関数実行中のタイミング保護違反処理【OS565】
- ネストして実行された信頼関数におけるタイミング保護違反処理【OS564】

タイミング保護に関する仕様は大きく変更する可能性があるため，本仕様では現状規定しない.

3.9.43 CheckISRMemoryAccess

<table>
<tr><td colspan="2">C 言語 I/F</td><td colspan="3">AccessType CheckISRMemoryAccess(
ISRType ISRID, MemoryStartAddressType Address,
MemorySizeType Size)【OS512】</td></tr>
<tr><td colspan="2" rowspan="3">パラメータ[in]</td><td>ISRID</td><td colspan="2">ISR ID</td></tr>
<tr><td>Address</td><td colspan="2">チェックするメモリ領域の先頭アドレス</td></tr>
<tr><td>Size</td><td colspan="2">チェックするメモリ領域のサイズ</td></tr>
<tr><td colspan="2">パラメータ[in/out]</td><td colspan="3">－</td></tr>
<tr><td colspan="2">パラメータ[out]</td><td colspan="3">－</td></tr>
<tr><td colspan="2" rowspan="8">返り値</td><td rowspan="8">アクセス可能か
不可能か</td><td colspan="2">メモリアクセス情報取得成功≪OS267≫</td></tr>
<tr><td colspan="2">ISRID が不正≪OS268≫</td></tr>
<tr><td colspan="2">C1ISR の指定≪NOS0408≫</td></tr>
<tr><td colspan="2">Size が 0 である≪NOS0862≫</td></tr>
<tr><td colspan="2">不正な処理単位からの呼出し≪NOS0417≫</td></tr>
<tr><td colspan="2">割込み禁止状態からの呼出し≪NOS0426≫</td></tr>
<tr><td colspan="2">Address と Size で指定されたメモリ領域が，メモリ保護属性が異なる領域を含む場合≪NOS0418≫</td></tr>
<tr><td colspan="2">起動していないコアへの呼出し≪NOS0979≫</td></tr>
<tr><td rowspan="9">エラーフックに渡されるエラーコード</td><td>標準エラー</td><td colspan="3">－</td></tr>
<tr><td rowspan="8">拡張エラー</td><td rowspan="2">E_OS_ID</td><td colspan="2">ISRID が不正≪NOS0440≫</td></tr>
<tr><td colspan="2">C1ISR の指定≪NOS0699≫</td></tr>
<tr><td>E_OS_VALUE</td><td colspan="2">Size が 0 である≪NOS0863≫</td></tr>
<tr><td>E_OS_CALLEVEL</td><td colspan="2">不正な処理単位からの呼出し≪NOS0411≫</td></tr>
<tr><td>E_OS_
DISABLEDINT</td><td colspan="2">割込み禁止状態からの呼出し≪NOS0412≫</td></tr>
<tr><td>E_OS_ILLEGAL_
ADDRESS</td><td colspan="2">Address と Size で指定されたメモリ領域が，メモリ保護属性が異なる領域を含む場合≪NOS0419≫</td></tr>
<tr><td>E_OS_CORE</td><td colspan="2">起動していないコアへの呼出し≪NOS0926≫</td></tr>
<tr><td colspan="2">コンフォーマンスクラス</td><td colspan="3">BCC1，BCC2，ECC1，ECC2</td></tr>
<tr><td colspan="2">スケーラビリティクラス</td><td colspan="3">SC3，SC4【OS517】</td></tr>
<tr><td colspan="5">機能</td></tr>
<tr><td colspan="5">本システムサービスは，ISRID で指定された ISR が，Address と Size で指定されたメモリ領域に対</td></tr>
</table>

して，書込み，読出し，実行が可能か，またスタックの一部であるかを示す情報を返す≪OS267≫.

機能仕様

CheckISRMemoryAccess は，ISRID で指定された ISR が，Address と Size で指定されたメモリ領域に対して，書込み，読出し，実行が可能か，またスタックの一部であるかを示す情報を返す【OS267】. CheckISRMemoryAccess の返り値と，OS マクロの OSMEMORY_IS_READABLE, OSMEMORY_IS_WRITEABLE，OSMEMORY_IS_EXECUTABLE, OSMEMORY_IS_STACKSPACE を使用して，アクセス可能かを確認することができる【OSa002】

CheckISRMemoryAccess 呼出し時に指定された ISRID が不正な場合はアクセス不可能を返す【OS268】. ISRID で指定された ISR が，C1ISR の場合も，アクセス不可能を返す【NOS0408】. これらの ISRID が不正な場合，エラーコードを E_OS_ID として，エラーフックを呼び出す【NOS0440】〔NOS0409〕.

Size に 0 を指定された場合，アクセス不可能を返す【NOS0862】. この場合，エラーコードを E_OS_VALUE として，エラーフックを呼び出す【NOS0863】〔NOS0409〕.

不正な処理単位から CheckISRMemoryAccess を呼び出した場合，アクセス不可能を返す【NOS0417】. また，割込み禁止状態で呼び出された場合も，アクセス不可能を返す【NOS0426】〔NOS0383〕〔NOS0384〕〔NOS0385〕.

Address と Size で指定されたメモリ領域が，メモリ保護属性が異なる領域を含む場合，アクセス不可能を返す【NOS0418】. この場合，エラーコードを E_OS_ILLEGAL_ADDRESS として，エラーフックを呼び出す【NOS0419】〔NOS0409〕. なお，ISRID で指定された C2ISR のスタック領域は，他のメモリ領域とメモリ保護属性が異なるものとして扱う【NOS0860】. ATK2 では，メモリ保護機能の機能レベル 3 において，ISR のスタック領域を共有した場合に，ISRID で指定した ISR のスタック領域の境界をまたぐメモリ領域に対して CheckISRMemoryAccess を呼び出しても，エラーフックが呼び出されず，アクセス不可能を返すことがある【IOS173】.

ISRID で指定された ISR が，信頼 OSAP に所属している場合は，一律書込み，読出し，実行可能とし，OsIsrSystemStackSize でサイズを定義したスタックの一部であるかを示す情報を返す【NOS0401】. ISRID で指定された ISR が，非信頼 OSAP に所属している場合，OsIsrStackSize でサイズを指定したスタックの一部であるかを示す情報を返す【NOS0402】.

マルチコア対応 OS における仕様

CheckISRMemoryAccess は，コアを跨いで呼び出すことができる【NOS0978】.

本 OS が起動していないコアに対して CheckISRMemoryAccess を呼び出した場合，アクセス不可能を返す【NOS0979】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，CheckISRMemoryAccess は，コアを跨いで呼び出すことができないと規定されている【OSa145】. しかし，CheckISRMemoryAccess をコアを跨いで呼び出すことによる不都合が

想定できないため，本仕様では呼び出すことができると規定した≪NOS0978≫.

AUTOSAR 仕様では，指定されたメモリ領域全体に対してアクセスできない場合，アクセス不可能とみなすと規定されている【OS313】．しかし，本仕様では指定されたメモリ領域が，メモリ保護属性が異なる領域を含む場合，アクセス不可能を返すと規定したため削除した≪NOS0418≫.

3.9.44 CheckTaskMemoryAccess

C 言語 I/F		AccessType CheckTaskMemoryAccess(TaskType TaskID, MemoryStartAddressType Address, MemorySizeType Size)【OS513】	
パラメータ[in]		TaskID	タスク ID
		Address	チェックするメモリ領域の先頭アドレス
		Size	チェックするメモリ領域のサイズ
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値		アクセス可能か不可能か	メモリアクセス情報取得成功≪OS269≫
			TaskID が不正≪OS270≫
			Size が 0 である≪NOS0864≫
			不正な処理単位からの呼出し≪NOS0420≫
			割込み禁止状態からの呼出し≪NOS0427≫
			Address と Size で指定されたメモリ領域が，メモリ保護属性が異なる領域を含む場合≪NOS0421≫
			起動していないコアへの呼出し≪NOS0981≫
エラーフックに渡されるエラーコード	標準エラー	－	
	拡張エラー	E_OS_ID	TaskID が不正≪NOS0441≫
		E_OS_VALUE	Size が 0 である≪NOS0865≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪NOS0411≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪NOS0412≫
		E_OS_ILLEGAL_ADDRESS	Address と Size で指定されたメモリ領域が，メモリ保護属性が異なる領域を含む場合≪NOS0422≫
		E_OS_CORE	起動していないコアへの呼出し≪NOS0926≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC3，SC4【OS518】	

機能

本システムサービスは，TaskID で指定されたタスクが，Address と Size で指定されたメモリ領域に対して，書込み，読出し，実行が可能か，またスタックの一部であるかを示す情報を返す.

機能仕様

CheckTaskMemoryAccess は，TaskID で指定されたタスクが，Address と Size で指定されたメモリ領域に対して，書込み，読出し，実行が可能か，またスタックの一部であるかを示す情報を返す【OS269】．CheckTaskMemoryAccess の返り値と，OS マクロの OSMEMORY_IS_READABLE，OSMEMORY_IS_WRITEABLE，OSMEMORY_IS_EXECUTABLE，OSMEMORY_IS_STACKSPACE を使用して，アクセス可能かを確認することができる【OSa003】．

CheckTaskMemoryAccess 呼出し時に指定された TaskID が不正な場合はアクセス不可能を返す【OS270】．この場合，エラーコードを E_OS_ID として，エラーフックを呼び出す【NOS0441】．

Size に 0 を指定された場合，アクセス不可能を返す【NOS0864】．この場合，エラーコードを E_OS_VALUE として，エラーフックを呼び出す【NOS0865】〔NOS0409〕．

不正な処理単位から CheckTaskMemoryAccess を呼び出した場合，アクセス不可能を返す【NOS0420】．また，割込み禁止状態で呼び出された場合も，アクセス不可能を返す【NOS0427】〔NOS0383〕〔NOS0384〕〔NOS0385〕．

Address と Size で指定されたメモリ領域が，メモリ保護属性が異なる領域を含む場合も，アクセス不可能を返す【NOS0421】．この場合，エラーコードを E_OS_ILLEGAL_ADDRESS として，エラーフックを呼び出す【NOS0422】〔NOS0409〕．なお，TaskID で指定されたタスクのスタック領域は，他のメモリ領域とメモリ保護属性が異なるものとして扱う【NOS0861】．ATK2 では，タスクのスタック領域を共有した場合に，TaskID で指定したタスクのスタック領域の境界をまたぐメモリ領域に対して CheckTaskMemoryAccess を呼び出しても，エラーフックが呼び出されず，アクセス不可能を返すことがある【IOS179】．

TaskID で指定されたタスクが，信頼 OSAP に所属している場合は，一律書込み，読出し，実行可能とし，OsTaskSystemStackSize でサイズを定義したスタックの一部であるかを示す情報を返す【NOS0403】．TaskID で指定されたタスクが，非信頼 OSAP に所属している場合，OsTaskStackSize でサイズを指定したスタックの一部であるかを示す情報を返す【NOS0404】．

マルチコア対応 OS における仕様

CheckTaskMemoryAccess は，コアを跨いで呼び出すことができる【NOS0980】．

本 OS が起動していないコアに対して CheckTaskMemoryAccess を呼び出した場合，アクセス不可能を返す【NOS0981】．

AUTOSAR 仕様との違い

AUTOSAR 仕様では，CheckTaskMemoryAccess は，コアを跨いで呼び出すことができないと規定されている【OSa146】．しかし，CheckTaskMemoryAccess をコアを跨いで呼び出すことによる不都合が想定できないため，本仕様では呼び出すことができると規定した≪NOS0980≫．

AUTOSAR 仕様では，指定されたメモリ領域全体に対してアクセスできない場合，アクセス不可能とみなすと規定されている【OS314】．しかし，本仕様では指定されたメモリ領域が，メモリ保護属性が異なる領域を含む場合，アクセス不可能を返すと規定したため削除した≪NOS0421≫．

3.9.45 CheckTaskAccess

<table>
<tr><td colspan="2">C 言語 I/F</td><td colspan="3">ObjectAccessType CheckTaskAccess(
ApplicationType ApplID,
TaskType TaskID)【NOS0522】</td></tr>
<tr><td colspan="2" rowspan="2">パラメータ[in]</td><td>ApplID</td><td colspan="2">OSAPID</td></tr>
<tr><td>TaskID</td><td colspan="2">タスク ID</td></tr>
<tr><td colspan="2">パラメータ[in/out]</td><td colspan="3">－</td></tr>
<tr><td colspan="2">パラメータ[out]</td><td colspan="3">－</td></tr>
<tr><td colspan="2" rowspan="6">返り値</td><td>ACCESS</td><td colspan="2">ApplID が示す OSAP が，TaskID が示すタスクにアクセス可能≪NOS0524≫</td></tr>
<tr><td rowspan="5">NO_ACCESS</td><td colspan="2">ApplID が示す OSAP が，TaskID が示すタスクにアクセス不可能≪NOS0525≫</td></tr>
<tr><td colspan="2">ApplID，TaskID が不正≪NOS0526≫</td></tr>
<tr><td colspan="2">不正な処理単位からの呼出し≪NOS0527≫</td></tr>
<tr><td colspan="2">割込み禁止状態からの呼出し≪NOS0528≫</td></tr>
<tr><td colspan="2">起動していないコアへの呼出し≪NOS0983≫</td></tr>
<tr><td rowspan="5">エラーフックに渡されるエラーコード</td><td>標準エラー</td><td colspan="3">－</td></tr>
<tr><td rowspan="4">拡張エラー</td><td>E_OS_ID</td><td colspan="2">ApplID，TaskID が不正≪NOS0519≫</td></tr>
<tr><td>E_OS_CALLEVEL</td><td colspan="2">不正な処理単位からの呼出し≪NOS0411≫</td></tr>
<tr><td>E_OS_DISABLEDINT</td><td colspan="2">割込み禁止状態からの呼出し≪NOS0412≫</td></tr>
<tr><td>E_OS_CORE</td><td colspan="2">起動していないコアへの呼出し≪NOS0926≫</td></tr>
<tr><td colspan="2">コンフォーマンスクラス</td><td colspan="3">BCC1，BCC2，ECC1，ECC2</td></tr>
<tr><td colspan="2">スケーラビリティクラス</td><td colspan="3">SC3，SC4【NOS0523】</td></tr>
<tr><td colspan="5">機能</td></tr>
<tr><td colspan="5">本システムサービスは，ApplID で指定された OSAP が，TaskID で指定されたタスクに対してアクセス可能かを返す.</td></tr>
</table>

機能仕様

CheckTaskAccess は, ApplID が示す OSAP が, TaskID で指定されたタスクにアクセス可能な場合，ACCESS を返す【NOS0524】．ApplID が示す OSAP が，TaskID で指定されたタスクにアクセス不可能な場合，NO_ACCESS を返す【NOS0525】.

CheckTaskAccess 呼出し時に指定された ApplID や TaskID が不正な場合，NO_ACCESS を返す【NOS0526】.

不正な処理単位から CheckTaskAccess を呼び出した場合, NO_ACCESS を返す【NOS0527】. また, 割込み禁止状態で呼び出された場合も，NO_ACCESS を返す【NOS0528】〔NOS0383〕〔NOS0384〕〔NOS0385〕.

CheckTaskAccess は，ApplID で指定された OSAP の状態に関わらず，指定されたタスクに対してアクセス可能かを返す【NOS0529】.

マルチコア対応 OS における仕様

CheckTaskAccess は，コアを跨いで呼び出すことができる【NOS0982】.

本 OS が起動していないコアに対して CheckTaskAccess を呼び出した場合，NO_ACCESS を返す【NOS0983】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，CheckObjectAccess は，コアを跨いで呼び出すことができないと規定されている【OSa147】. しかし，CheckObjectAccess (CheckTaskAccess)をコアを跨いで呼び出すことによる不都合が想定できないため，本仕様では呼び出すことができると規定した≪NOS0982≫.

3.9.46 CheckISRAccess

<table>
<tr><td colspan="2">C 言語 I/F</td><td colspan="3">ObjectAccessType CheckISRAccess(
ApplicationType ApplID,
ISRType ISRID)【NOS0530】</td></tr>
<tr><td colspan="2" rowspan="2">パラメータ[in]</td><td>ApplID</td><td colspan="2">OSAPID</td></tr>
<tr><td>ISRID</td><td colspan="2">ISRID</td></tr>
<tr><td colspan="2">パラメータ[in/out]</td><td colspan="3">－</td></tr>
<tr><td colspan="2">パラメータ[out]</td><td colspan="3">－</td></tr>
<tr><td colspan="2" rowspan="8">返り値</td><td>ACCESS</td><td colspan="2">ApplID が示す OSAP が，ISRID が示す ISR にアクセス可能≪NOS0532≫</td></tr>
<tr><td rowspan="7">NO_ACCESS</td><td colspan="2">ApplID が示す OSAP が，ISRID が示す ISR にアクセス不可能≪NOS0533≫</td></tr>
<tr><td colspan="2">ApplID，ISRID が不正≪NOS0534≫</td></tr>
<tr><td colspan="2">C1ISR の指定≪NOS0697≫</td></tr>
<tr><td colspan="2">不正な処理単位からの呼出し≪NOS0535≫</td></tr>
<tr><td colspan="2">割込み禁止状態からの呼出し≪NOS0536≫</td></tr>
<tr><td colspan="2">起動していないコアへの呼出し≪NOS0985≫</td></tr>
<tr><td rowspan="6">エラーフックに渡されるエラーコード</td><td>標準エラー</td><td colspan="3">－</td></tr>
<tr><td rowspan="5">拡張エラー</td><td rowspan="2">E_OS_ID</td><td>ApplID，ISRID が不正≪NOS0519≫</td></tr>
<tr><td>C1ISR の指定≪NOS0699≫</td></tr>
<tr><td>E_OS_CALLEVEL</td><td>不正な処理単位からの呼出し≪NOS0411≫</td></tr>
<tr><td>E_OS_DISABLEDINT</td><td>割込み禁止状態からの呼出し≪NOS0412≫</td></tr>
<tr><td>E_OS_CORE</td><td>起動していないコアへの呼出し≪NOS0926≫</td></tr>
<tr><td colspan="2">コンフォーマンスクラス</td><td colspan="3">BCC1，BCC2，ECC1，ECC2</td></tr>
<tr><td colspan="2">スケーラビリティクラス</td><td colspan="3">SC3，SC4【NOS0531】</td></tr>
<tr><td colspan="5">機能</td></tr>
<tr><td colspan="5">本システムサービスは，ApplID で指定された OSAP が，ISRID で指定された ISR に対してアクセス可能かを返す.</td></tr>
</table>

機能仕様

CheckISRAccess は，ApplID が示す OSAP が，ISRID で指定された ISR にアクセス可能な場合，ACCESS を返す【NOS0532】．ApplID が示す OSAP が，ISRID で指定された ISR にアクセス不可能な場合，NO_ACCESS を返す【NOS0533】.

CheckISRAccess 呼出し時に指定された ApplID や ISRID が不正な場合，NO_ACCESS を返す【NOS0534】．ISRID で指定された ISR が，C1ISR の場合も，アクセス不可能を返す【NOS0697】.

不正な処理単位から CheckISRAccess を呼び出した場合，NO_ACCESS を返す【NOS0535】．また，割込み禁止状態で呼び出された場合も，NO_ACCESS を返す【NOS0536】〔NOS0383〕〔NOS0384〕〔NOS0385〕.

CheckISRAccess は，ApplID で指定された OSAP の状態に関わらず，指定された ISR に対してアクセス可能かを返す【NOS0537】.

マルチコア対応 OS における仕様

CheckISRAccess は，コアを跨いで呼び出すことができる【NOS0984】.

本 OS が起動していないコアに対して CheckISRAccess を呼び出した場合，NO_ACCESS を返す【NOS0985】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，CheckObjectAccess は，コアを跨いで呼び出すことができないと規定されている≪OSa147≫．しかし，CheckObjectAccess (CheckISRAccess)をコアを跨いで呼び出すことによる不都合が想定できないため，本仕様では呼び出すことができると規定した≪NOS0984≫.

3.9.47 CheckAlarmAccess

<table>
<tr><td colspan="2">C 言語 I/F</td><td colspan="3">ObjectAccessType CheckAlarmAccess(
ApplicationType ApplID,
AlarmType AlarmID)【NOS0538】</td></tr>
<tr><td colspan="2" rowspan="2">パラメータ[in]</td><td>ApplID</td><td colspan="2">OSAPID</td></tr>
<tr><td>AlarmID</td><td colspan="2">アラーム ID</td></tr>
<tr><td colspan="2">パラメータ[in/out]</td><td colspan="3">－</td></tr>
<tr><td colspan="2">パラメータ[out]</td><td colspan="3">－</td></tr>
<tr><td colspan="2" rowspan="6">返り値</td><td>ACCESS</td><td colspan="2">ApplID が示す OSAP が，AlarmID が示すアラームにアクセス可能≪NOS0540≫</td></tr>
<tr><td rowspan="5">NO_ACCESS</td><td colspan="2">ApplID が示す OSAP が，AlarmID が示すアラームにアクセス不可能≪NOS0541≫</td></tr>
<tr><td colspan="2">ApplID，AlarmID が不正≪NOS0542≫</td></tr>
<tr><td colspan="2">不正な処理単位からの呼出し≪NOS0543≫</td></tr>
<tr><td colspan="2">割込み禁止状態からの呼出し≪NOS0544≫</td></tr>
<tr><td colspan="2">起動していないコアへの呼出し≪NOS0987≫</td></tr>
<tr><td rowspan="5">エラーフックに渡されるエラーコード</td><td>標準エラー</td><td colspan="3">－</td></tr>
<tr><td rowspan="4">拡張エラー</td><td>E_OS_ID</td><td colspan="2">ApplID，AlarmID が不正≪NOS0519≫</td></tr>
<tr><td>E_OS_CALLEVEL</td><td colspan="2">不正な処理単位からの呼出し≪NOS0411≫</td></tr>
<tr><td>E_OS_DISABLEDINT</td><td colspan="2">割込み禁止状態からの呼出し≪NOS0412≫</td></tr>
<tr><td>E_OS_CORE</td><td colspan="2">起動していないコアへの呼出し≪NOS0926≫</td></tr>
<tr><td colspan="2">コンフォーマンスクラス</td><td colspan="3">BCC1，BCC2，ECC1，ECC2</td></tr>
<tr><td colspan="2">スケーラビリティクラス</td><td colspan="3">SC3，SC4【NOS0539】</td></tr>
<tr><td colspan="5">機能</td></tr>
<tr><td colspan="5">本システムサービスは，ApplID で指定された OSAP が，AlarmID で指定されたアラームに対してアクセス可能かを返す.</td></tr>
</table>

機能仕様

CheckAlarmAccess は，ApplID が示す OSAP が，AlarmID で指定されたアラームにアクセス可能な場合，ACCESS を返す【NOS0540】．ApplID が示す OSAP が，AlarmID で指定されたアラームにアクセス不可能な場合，NO_ACCESS を返す【NOS0541】.

CheckAlarmAccess 呼出し時に指定された ApplID や AlarmID が不正な場合，NO_ACCESS を返す【NOS0542】.

不正な処理単位から CheckAlarmAccess を呼び出した場合，NO_ACCESS を返す【NOS0543】．また，割込み禁止状態で呼び出された場合も，NO_ACCESS を返す【NOS0544】〔NOS0383〕〔NOS0384〕〔NOS0385〕．

CheckAlarmAccess は，ApplID で指定された OSAP の状態に関わらず，指定されたアラームに対してアクセス可能かを返す【NOS0545】．

マルチコア対応 OS における仕様

CheckAlarmAccess は，コアを跨いで呼び出すことができる【NOS0986】．

本 OS が起動していないコアに対して CheckAlarmAccess を呼び出した場合，NO_ACCESS を返す【NOS0987】．

AUTOSAR 仕様との違い

AUTOSAR 仕様では，CheckObjectAccess は，コアを跨いで呼び出すことができないと規定されている≪OSa147≫．しかし，CheckObjectAccess (CheckAlarmAccess)をコアを跨いで呼び出すことによる不都合が想定できないため，本仕様では呼び出すことができると規定した≪NOS0986≫．

3.9.48 CheckResourceAccess

C 言語 I/F		ObjectAccessType CheckResourceAccess(ApplicationType ApplID, ResourceType ResID)【NOS0546】	
パラメータ[in]		ApplID	OSAPID
		ResID	リソース ID
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値		ACCESS	ApplID が示す OSAP が，ResID が示すリソースにアクセス可能≪NOS0548≫
		NO_ACCESS	ApplID が示す OSAP が，ResID が示すリソースにアクセス不可能≪NOS0549≫
			ApplID，ResID が不正≪NOS0550≫
			内部リソースの指定≪NOS0701≫
			不正な処理単位からの呼出し≪NOS0551≫
			割込み禁止状態からの呼出し≪NOS0552≫
			起動していないコアへの呼出し≪NOS0989≫
エラーフックに渡されるエラーコード	標準エラー	－	
	拡張エラー	E_OS_ID	ApplID，ResID が不正≪NOS0519≫
			内部リソースの指定≪NOS0700≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪NOS0411≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪NOS0412≫
		E_OS_CORE	起動していないコアへの呼出し≪NOS0926≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC3，SC4【NOS0547】	
機能			
本システムサービスは，ApplID で指定された OSAP が，ResID で指定されたリソースに対してアクセス可能かを返す.			

機能仕様

CheckResourceAccess は，ApplID が示す OSAP が，ResID で指定されたリソースにアクセス可能な場合，ACCESS を返す【NOS0548】．ApplID が示す OSAP が，ResID で指定されたリソースにアクセス不可能な場合，NO_ACCESS を返す【NOS0549】.

CheckResourceAccess 呼出し時に指定された ApplID や ResID が不正な場合，NO_ACCESS を返す【NOS0550】．また，ResID に内部リソースが指定された場合も，NO_ACCESS を返す【NOS0701】．

不正な処理単位から CheckResourceAccess を呼び出した場合，NO_ACCESS を返す【NOS0551】．また，割込み禁止状態で呼び出された場合も，NO_ACCESS を返す【NOS0552】〔NOS0383〕〔NOS0384〕〔NOS0385〕．

CheckResourceAccess は，ApplID で指定された OSAP の状態に関わらず，指定されたリソースに対してアクセス可能かを返す【NOS0553】．

マルチコア対応 OS における仕様

CheckResourceAccess は，コアを跨いで呼び出すことができる【NOS0988】．

本 OS が起動していないコアに対して CheckResourceAccess を呼び出した場合，NO_ACCESS を返す【NOS0989】．

AUTOSAR 仕様との違い

AUTOSAR 仕様では，CheckObjectAccess は，コアを跨いで呼び出すことができないと規定されている≪OSa147≫．しかし，CheckObjectAccess (CheckResourceAccess)をコアを跨いで呼び出すことによる不都合が想定できないため，本仕様では呼び出すことができると規定した≪NOS0988≫．

3.9.49 CheckCounterAccess

<table>
<tr><td colspan="2">C 言語 I/F</td><td colspan="3">ObjectAccessType CheckCounterAccess(
ApplicationType ApplID,
CounterType CounterID)【NOS0554】</td></tr>
<tr><td colspan="2" rowspan="2">パラメータ[in]</td><td>ApplID</td><td colspan="2">OSAPID</td></tr>
<tr><td>CounterID</td><td colspan="2">カウンタ ID</td></tr>
<tr><td colspan="2">パラメータ[in/out]</td><td colspan="3">－</td></tr>
<tr><td colspan="2">パラメータ[out]</td><td colspan="3">－</td></tr>
<tr><td colspan="2" rowspan="6">返り値</td><td>ACCESS</td><td colspan="2">ApplID が示す OSAP が，CounterID が示すカウンタにアクセス可能≪NOS0556≫</td></tr>
<tr><td rowspan="5">NO_ACCESS</td><td colspan="2">ApplID が示す OSAP が，CounterID が示すカウンタにアクセス不可能≪NOS0557≫</td></tr>
<tr><td colspan="2">ApplID，CounterID が不正≪NOS0558≫</td></tr>
<tr><td colspan="2">不正な処理単位からの呼出し≪NOS0559≫</td></tr>
<tr><td colspan="2">割込み禁止状態からの呼出し≪NOS0560≫</td></tr>
<tr><td colspan="2">起動していないコアへの呼出し≪NOS0991≫</td></tr>
<tr><td rowspan="5">エラーフックに渡されるエラーコード</td><td>標準エラー</td><td colspan="3">－</td></tr>
<tr><td rowspan="4">拡張エラー</td><td>E_OS_ID</td><td colspan="2">ApplID，CounterID が不正≪NOS0519≫</td></tr>
<tr><td>E_OS_CALLEVEL</td><td colspan="2">不正な処理単位からの呼出し≪NOS0411≫</td></tr>
<tr><td>E_OS_DISABLEDINT</td><td colspan="2">割込み禁止状態からの呼出し≪NOS0412≫</td></tr>
<tr><td>E_OS_CORE</td><td colspan="2">起動していないコアへの呼出し≪NOS0926≫</td></tr>
<tr><td colspan="2">コンフォーマンスクラス</td><td colspan="3">BCC1，BCC2，ECC1，ECC2</td></tr>
<tr><td colspan="2">スケーラビリティクラス</td><td colspan="3">SC3，SC4【NOS0555】</td></tr>
<tr><td colspan="5">機能</td></tr>
<tr><td colspan="5">本システムサービスは，ApplID で指定された OSAP が，CounterID で指定されたカウンタに対してアクセス可能かを返す.</td></tr>
</table>

機能仕様

CheckCounterAccess は，ApplID が示す OSAP が，CounterID で指定されたカウンタにアクセス可能な場合，ACCESS を返す【NOS0556】．ApplID が示す OSAP が，CounterID で指定されたカウンタにアクセス不可能な場合，NO_ACCESS を返す【NOS0557】.

CheckCounterAccess 呼出し時に指定された ApplID や CounterID が不正な場合，NO_ACCESS を返す【NOS0558】.

不正な処理単位から CheckCounterAccess を呼び出した場合，NO_ACCESS を返す【NOS0559】. また, 割込み禁止状態で呼び出された場合も, NO_ACCESS を返す【NOS0560】〔NOS0383〕〔NOS0384〕〔NOS0385〕.

CheckCounterAccess は，ApplID で指定された OSAP の状態に関わらず，指定されたカウンタに対してアクセス可能かを返す【NOS0561】.

マルチコア対応 OS における仕様

CheckCounterAccess は，コアを跨いで呼び出すことができる【NOS0990】.

本 OS が起動していないコアに対して CheckCounterAccess を呼び出した場合，NO_ACCESS を返す【NOS0991】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，CheckObjectAccess は，コアを跨いで呼び出すことができないと規定されている≪OSa147≫．しかし，CheckObjectAccess (CheckCounterAccess)をコアを跨いで呼び出すことによる不都合が想定できないため，本仕様では呼び出すことができると規定した≪NOS0990≫.

3.9.50 CheckScheduleTableAccess

<table>
<tr><td colspan="2">C 言語 I/F</td><td colspan="3">ObjectAccessType CheckScheduleTableAccess(
ApplicationType ApplID,
ScheduleTableType ScheduleTableID)【NOS0562】</td></tr>
<tr><td colspan="2" rowspan="2">パラメータ[in]</td><td>ApplID</td><td colspan="2">OSAPID</td></tr>
<tr><td>ScheduleTableID</td><td colspan="2">スケジュールテーブル ID</td></tr>
<tr><td colspan="2">パラメータ[in/out]</td><td colspan="3">－</td></tr>
<tr><td colspan="2">パラメータ[out]</td><td colspan="3">－</td></tr>
<tr><td colspan="2" rowspan="6">返り値</td><td>ACCESS</td><td colspan="2">ApplID が示す OSAP が，ScheduleTableID が示すスケジュールテーブルにアクセス可能≪NOS0564≫</td></tr>
<tr><td rowspan="5">NO_ACCESS</td><td colspan="2">ApplID が示す OSAP が，ScheduleTableID が示すスケジュールテーブルにアクセス不可能≪NOS0565≫</td></tr>
<tr><td colspan="2">ApplID，ScheduleTableID が不正≪NOS0566≫</td></tr>
<tr><td colspan="2">不正な処理単位からの呼出し≪NOS0567≫</td></tr>
<tr><td colspan="2">割込み禁止状態からの呼出し≪NOS0568≫</td></tr>
<tr><td colspan="2">起動していないコアへの呼出し≪NOS0993≫</td></tr>
<tr><td rowspan="5">エラーフックに渡されるエラーコード</td><td>標準エラー</td><td colspan="3">－</td></tr>
<tr><td rowspan="4">拡張エラー</td><td>E_OS_ID</td><td colspan="2">ApplID，ScheduleTableID が不正≪NOS0519≫</td></tr>
<tr><td>E_OS_CALLEVEL</td><td colspan="2">不正な処理単位からの呼出し≪NOS0411≫</td></tr>
<tr><td>E_OS_DISABLEDINT</td><td colspan="2">割込み禁止状態からの呼出し≪NOS0412≫</td></tr>
<tr><td>E_OS_CORE</td><td colspan="2">起動していないコアへの呼出し≪NOS0926≫</td></tr>
<tr><td colspan="2">コンフォーマンスクラス</td><td colspan="3">BCC1，BCC2，ECC1，ECC2</td></tr>
<tr><td colspan="2">スケーラビリティクラス</td><td colspan="3">SC3，SC4【NOS0563】</td></tr>
<tr><td colspan="5">機能</td></tr>
<tr><td colspan="5">本システムサービスは，ApplID で指定された OSAP が，ScheduleTableID で指定されたスケジュールテーブルに対してアクセス可能かを返す.</td></tr>
</table>

機能仕様

CheckScheduleTableAccess は，ApplID が示す OSAP が，ScheduleTableID で指定されたスケジュ

ールテーブルにアクセス可能な場合，ACCESS を返す【NOS0564】．ApplID が示す OSAP が，ScheduleTableID で指定されたスケジュールテーブルにアクセス不可能な場合，NO_ACCESS を返す【NOS0565】.

CheckScheduleTableAccess 呼出し時に指定された ApplID や ScheduleTableID が不正な場合，NO_ACCESS を返す【NOS0566】.

不正な処理単位から CheckScheduleTableAccess を呼び出した場合，NO_ACCESS を返す【NOS0567】．また，割込み禁止状態で呼び出された場合も，NO_ACCESS を返す【NOS0568】〔NOS0383〕〔NOS0384〕〔NOS0385〕.

CheckScheduleTableAccess は，ApplID で指定された OSAP の状態に関わらず，指定されたスケジュールテーブルに対してアクセス可能かを返す【NOS0569】.

マルチコア対応 OS における仕様

CheckScheduleTableAccess は，コアを跨いで呼び出すことができる【NOS0992】.

本 OS が起動していないコアに対して CheckScheduleTableAccess を呼び出した場合, NO_ACCESS を返す【NOS0993】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，CheckObjectAccess は，コアを跨いで呼び出すことができないと規定されている≪OSa147≫．しかし，CheckObjectAccess (CheckScheduleTableAccess)をコアを跨いで呼び出すことによる不都合が想定できないため，本仕様では呼び出すことができると規定した≪NOS0992≫.

3.9.51 CheckSpinlockAccess

<table>
<tr><td colspan="2">C 言語 I/F</td><td colspan="3">ObjectAccessType CheckSpinlockAccess(
ApplicationType ApplID,
SpinlockIdType SpinlockId)【NOS1004】</td></tr>
<tr><td colspan="2" rowspan="2">パラメータ[in]</td><td>ApplID</td><td colspan="2">OSAPID</td></tr>
<tr><td>SpinlockId</td><td colspan="2">スピンロック ID</td></tr>
<tr><td colspan="2">パラメータ[in/out]</td><td colspan="3">－</td></tr>
<tr><td colspan="2">パラメータ[out]</td><td colspan="3">－</td></tr>
<tr><td colspan="2" rowspan="5">返り値</td><td>ACCESS</td><td colspan="2">ApplID が示す OSAP が，SpinlockId が示すスピンロックにアクセス可能≪NOS1005≫</td></tr>
<tr><td rowspan="4">NO_ACCESS</td><td colspan="2">ApplID が示す OSAP が，SpinlockId が示すスピンロックにアクセス不可能≪NOS1006≫</td></tr>
<tr><td colspan="2">ApplID，SpinlockId が不正≪NOS1007≫</td></tr>
<tr><td colspan="2">不正な処理単位からの呼出し≪NOS1008≫</td></tr>
<tr><td colspan="2">割込み禁止状態からの呼出し≪NOS1009≫</td></tr>
<tr><td rowspan="4">エラーフックに渡されるエラーコード</td><td>標準エラー</td><td colspan="3">－</td></tr>
<tr><td rowspan="3">拡張エラー</td><td>E_OS_ID</td><td colspan="2">ApplID，SpinlockId が不正≪NOS0519≫</td></tr>
<tr><td>E_OS_CALLEVEL</td><td colspan="2">不正な処理単位からの呼出し≪NOS0411≫</td></tr>
<tr><td>E_OS_DISABLEDINT</td><td colspan="2">割込み禁止状態からの呼出し≪NOS0412≫</td></tr>
<tr><td colspan="2">コンフォーマンスクラス</td><td colspan="3">BCC1，BCC2，ECC1，ECC2</td></tr>
<tr><td colspan="2">スケーラビリティクラス</td><td colspan="3">マルチコア：SC1，SC2，SC3，SC4≪NOS0339≫</td></tr>
<tr><td colspan="5">機能</td></tr>
<tr><td colspan="5">本システムサービスは，ApplID で指定された OSAP が，SpinlockId で指定されたスピンロックに対してアクセス可能かを返す.</td></tr>
</table>

機能仕様

CheckSpinlockAccess は，ApplID が示す OSAP が，SpinlockId で指定されたスピンロックにアクセス可能な場合，ACCESS を返す【NOS1005】．ApplID が示す OSAP が，SpinlockId で指定されたスピンロックにアクセス不可能な場合，NO_ACCESS を返す【NOS1006】.

CheckSpinlockAccess 呼出し時に指定された ApplID や SpinlockId が不正な場合，NO_ACCESS を返す【NOS1007】.

不正な処理単位から CheckSpinlockAccess を呼び出した場合，NO_ACCESS を返す【NOS1008】. また，割込み禁止状態で呼び出された場合も，NO_ACCESS を返す【NOS1009】〔NOS0383〕〔NOS0384〕〔NOS0385〕.

CheckSpinlockAccess は，ApplID で指定された OSAP の状態に関わらず，指定された OS オブジェクトに対してアクセス可能かを返す【NOS1010】.

3.9.52 CheckObjectAccess

AUTOSAR 仕様では，OS オブジェクトに対する OSAP のアクセス権の有無を，取得するシステムサービスとして，CheckObjectAccess を規定している．

- CheckObjectAccess を使用可能なスケーラビリティクラス【OS519】
- CheckObjectAccess の使用条件【OS450】
- CheckObjectAccess の概要【OS256】
- CheckObjectAccess のアクセス権有りの場合の処理【OS271】
- CheckObjectAccess のアクセス権無しの場合の処理【OS272】
- CheckObjectAccess におけるエラー発生時の処理【OS423】

しかし，CheckObjectAccess は，すべての OS オブジェクトに対応するため，可変長引数を使用している．可変長引数を使用すると，処理速度が遅くなり，引数の型チェックもできないため，本仕様では，OS オブジェクト毎に，OSAP のアクセス権の有無を取得するシステムサービスを規定した≪NOS0522≫≪NOS0530≫≪NOS0538≫≪NOS0546≫≪NOS0554≫≪NOS0562≫．

ATK2 では, AUTOSAR 仕様との互換性のために, CheckObjectAccess をライブラリとして提供する．

<table>
<tr><td>C 言語 I/F</td><td colspan="2">ObjectAccessType CheckObjectAccess(
ApplicationType ApplID,
ObjectTypeType ObjectType,
void …) 【IOS034】</td></tr>
<tr><td rowspan="3">パラメータ[in]</td><td>ApplID</td><td>OSAPID</td></tr>
<tr><td>ObjectType</td><td>…の OS オブジェクト型</td></tr>
<tr><td>…</td><td>OS オブジェクトの ID</td></tr>
<tr><td>パラメータ[in/out]</td><td colspan="2">－</td></tr>
<tr><td>パラメータ[out]</td><td colspan="2">－</td></tr>
<tr><td rowspan="2">返り値</td><td>ACCESS</td><td>ApplID が示す OSAP が…で示す OS オブジェクトにアクセス可能≪IOS038≫</td></tr>
<tr><td>NO_ACCESS</td><td>ApplID が示す OSAP が…で示す OS オブジェクトにアクセス不可能≪IOS038≫
または ObjectType が不正≪IOS040≫</td></tr>
<tr><td>コンフォーマンスクラス</td><td colspan="2">BCC1，BCC2，ECC1，ECC2</td></tr>
<tr><td>スケーラビリティクラス</td><td colspan="2">SC3，SC4【IOS037】</td></tr>
<tr><td colspan="3">機能</td></tr>
<tr><td colspan="3">本ライブラリは，ApplID で指定された OSAP が指定された OS オブジェクトに対してアクセス可能かを返す．</td></tr>
</table>

機能仕様

CheckObjectAccess は，与えられた引数から OS オブジェクト毎に規定したシステムサービス，CheckTaskAccess ， CheckISRAccess ， CheckAlarmAccess ， CheckResourceAccess ，CheckCounterAccess，CheckScheduleTableAccess，CheckSpinlockAccess を呼ぶ実装とし，返り値はこれらのシステムサービスからの返り値をそのまま使用する【IOS038】．そのため，エラーチェックおよびエラー検出時のエラーフックは，ライブラリから呼び出した各システムサービスによるエラーフックが呼び出されるものとする【IOS039】.

ObjectType で指定した OS オブジェクトが不正で呼び出すシステムサービスを特定できない場合，本ライブラリは NO_ACCESS を返す【IOS040】．この場合，エラーフックは呼び出されない【IOS175】.

3.9.53 CheckTaskOwnership

<table>
<tr><td colspan="2">C 言語 I/F</td><td colspan="3">ApplicationType CheckTaskOwnership
(TaskType TaskID)【NOS0570】</td></tr>
<tr><td colspan="2">パラメータ[in]</td><td>TaskID</td><td colspan="2">タスク ID</td></tr>
<tr><td colspan="2">パラメータ[in/out]</td><td colspan="3">－</td></tr>
<tr><td colspan="2">パラメータ[out]</td><td colspan="3">－</td></tr>
<tr><td colspan="2" rowspan="5">返り値</td><td colspan="2">所属する OSAP の OSAPID</td><td>OSAPID 取得成功≪NOS0572≫</td></tr>
<tr><td rowspan="4">INVALID_OSAPPLICATION</td><td>TaskID が不正≪NOS0573≫</td></tr>
<tr><td>不正な処理単位からの呼出し≪NOS0574≫</td></tr>
<tr><td>割込み禁止状態からの呼出し≪NOS0575≫</td></tr>
<tr><td>起動していないコアへの呼出し≪NOS0995≫</td></tr>
<tr><td rowspan="5">エラーフックに渡されるエラーコード</td><td>標準エラー</td><td colspan="3">－</td></tr>
<tr><td rowspan="4">拡張エラー</td><td>E_OS_ID</td><td>TaskID が不正≪NOS0519≫</td></tr>
<tr><td>E_OS_CALLEVEL</td><td>不正な処理単位からの呼出し≪NOS0411≫</td></tr>
<tr><td>E_OS_DISABLEDINT</td><td>割込み禁止状態からの呼出し≪NOS0412≫</td></tr>
<tr><td>E_OS_CORE</td><td>起動していないコアへの呼出し≪NOS0926≫</td></tr>
<tr><td colspan="2">コンフォーマンスクラス</td><td colspan="3">BCC1，BCC2，ECC1，ECC2</td></tr>
<tr><td colspan="2">スケーラビリティクラス</td><td colspan="3">SC3，SC4【NOS0571】
マルチコア：SC1，SC2【NOS1011】</td></tr>
<tr><td colspan="5">機能</td></tr>
<tr><td colspan="5">本システムサービスは，TaskID で指定されたタスクが所属する OSAPID を返す.</td></tr>
</table>

機能仕様

CheckTaskOwnership は，TaskID で指定されたタスクが所属する OSAPID を返す【NOS0572】.

CheckTaskOwnership 呼出し時に，引数で指定された TaskID が不正な場合，INVALID_OSAPPLICATION を返す【NOS0573】.

不正な処理単位から CheckTaskOwnership を呼び出した場合，INVALID_OSAPPLICATION を返す【NOS0574】．また，割込み禁止状態で呼び出された場合も，INVALID_OSAPPLICATION を返す【NOS0575】〔NOS0383〕〔NOS0384〕〔NOS0385〕.

CheckTaskOwnership は，引数で指定されたタスクが所属する OSAP の状態に関わらず，タスクが所属する OSAPID を返す【NOS0576】.

マルチコア対応 OS における仕様

CheckTaskOwnership は，コアを跨いで呼び出すことができる【NOS0994】.

本 OS が起動していないコアに対して CheckTaskOwnership を呼び出した場合，INVALID_OSAPPLICATION を返す【NOS0995】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，CheckObjectOwnership は，コアを跨いで呼び出すことができないと規定されている【OSa148】．しかし，CheckObjectAccess (CheckTaskOwnership)をコアを跨いで呼び出すことによる不都合が想定できないため，本仕様では呼び出すことができると規定した≪NOS0994≫.

3.9.54 CheckISROwnership

C 言語 I/F		ApplicationType CheckISROwnership (ISRType ISRID)【NOS0577】	
パラメータ[in]		ISRID	ISR ID
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値		所属する OSAP の OSAPID	OSAPID 取得成功≪NOS0579≫
		INVALID_OSAPPLICATION	ISRID が不正≪NOS0580≫
			C1ISR の指定≪NOS0698≫
			不正な処理単位からの呼出し≪NOS0581≫
			割込み禁止状態からの呼出し≪NOS0582≫
			起動していないコアへの呼出し≪NOS0997≫
エラーフックに渡されるエラーコード	標準エラー	－	
	拡張エラー	E_OS_ID	ISRID が不正≪NOS0519≫
			C1ISR の指定≪NOS0699≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪NOS0411≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪NOS0412≫
		E_OS_CORE	起動していないコアへの呼出し≪NOS0926≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC3，SC4【NOS0578】 マルチコア：SC1，SC2【NOS1012】	
機能			
本システムサービスは，ISRID で指定された ISR が所属する OSAPID を返す.			

機能仕様

CheckISROwnership は，ISRID で指定された ISR が所属する OSAPID を返す【NOS0579】.

CheckISROwnership 呼出し時に，引数で指定された ISRID が不正な場合，INVALID_OSAPPLICATION を返す【NOS0580】．ISRID で指定された ISR が，C1ISR の場合も，INVALID_OSAPPLICATION を返す【NOS0698】.

不正な処理単位から CheckISROwnership を呼び出した場合，INVALID_OSAPPLICATION を返す【NOS0581】．また，割込み禁止状態で呼び出された場合も，INVALID_OSAPPLICATION を返す【NOS0582】〔NOS0383〕〔NOS0384〕〔NOS0385〕．

CheckISROwnership は，引数で指定された ISR が所属する OSAP の状態に関わらず，ISR が所属する OSAPID を返す【NOS0583】．

マルチコア対応 OS における仕様

CheckISROwnership は，コアを跨いで呼び出すことができる【NOS0996】．

本 OS が起動していないコアに対して CheckISROwnership を呼び出した場合，INVALID_OSAPPLICATION を返す【NOS0997】．

AUTOSAR 仕様との違い

AUTOSAR 仕様では，CheckObjectOwnership は，コアを跨いで呼び出すことができないと規定されている≪OSa148≫．しかし，CheckObjectAccess (CheckISROwnership)をコアを跨いで呼び出すことによる不都合が想定できないため，本仕様では呼び出すことができると規定した≪NOS0996≫．

3.9.55 CheckAlarmOwnership

<table>
<tr><td colspan="2">C 言語 I/F</td><td colspan="3">ApplicationType CheckAlarmOwnership
(AlarmType AlarmID)【NOS0584】</td></tr>
<tr><td colspan="2">パラメータ[in]</td><td>AlarmID</td><td colspan="2">アラーム ID</td></tr>
<tr><td colspan="2">パラメータ[in/out]</td><td colspan="3">－</td></tr>
<tr><td colspan="2">パラメータ[out]</td><td colspan="3">－</td></tr>
<tr><td colspan="2" rowspan="5">返り値</td><td colspan="2">所属する OSAP の OSAPID</td><td>OSAPID 取得成功≪NOS0586≫</td></tr>
<tr><td colspan="2" rowspan="4">INVALID_OSAPPLICATION</td><td>AlarmID が不正≪NOS0587≫</td></tr>
<tr><td>不正な処理単位からの呼出し≪NOS0588≫</td></tr>
<tr><td>割込み禁止状態からの呼出し≪NOS0589≫</td></tr>
<tr><td>起動していないコアへの呼出し≪NOS0999≫</td></tr>
<tr><td rowspan="5">エラーフックに渡されるエラーコード</td><td>標準エラー</td><td colspan="3">－</td></tr>
<tr><td rowspan="4">拡張エラー</td><td>E_OS_ID</td><td colspan="2">AlarmID が不正≪NOS0519≫</td></tr>
<tr><td>E_OS_CALLEVEL</td><td colspan="2">不正な処理単位からの呼出し≪NOS0411≫</td></tr>
<tr><td>E_OS_DISABLEDINT</td><td colspan="2">割込み禁止状態からの呼出し≪NOS0412≫</td></tr>
<tr><td>E_OS_CORE</td><td colspan="2">起動していないコアへの呼出し≪NOS0926≫</td></tr>
<tr><td colspan="2">コンフォーマンスクラス</td><td colspan="3">BCC1，BCC2，ECC1，ECC2</td></tr>
<tr><td colspan="2">スケーラビリティクラス</td><td colspan="3">SC3，SC4【NOS0585】
マルチコア：SC1，SC2【NOS1013】</td></tr>
<tr><td colspan="5">機能</td></tr>
<tr><td colspan="5">本システムサービスは，AlarmID で指定されたアラームが所属する OSAPID を返す.</td></tr>
</table>

機能仕様

CheckAlarmOwnership は, AlarmID で指定されたアラームが所属する OSAPID を返す【NOS0586】.

CheckAlarmOwnership 呼出し時に，引数で指定された AlarmID が不正な場合，INVALID_OSAPPLICATION を返す【NOS0587】.

不正な処理単位から CheckAlarmOwnership を呼び出した場合，INVALID_OSAPPLICATION を返す【NOS0588】. また，割込み禁止状態で呼び出された場合も，INVALID_OSAPPLICATION を返す【NOS0589】〔NOS0383〕〔NOS0384〕〔NOS0385〕.

CheckAlarmOwnership は，引数で指定されたアラームが所属する OSAP の状態に関わらず，アラームが所属する OSAPID を返す【NOS0590】.

マルチコア対応 OS における仕様

CheckAlarmOwnership は，コアを跨いで呼び出すことができる【NOS0998】.

本 OS が起動していないコアに対して CheckAlarmOwnership を呼び出した場合，INVALID_OSAPPLICATION を返す【NOS0999】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，CheckObjectOwnership は，コアを跨いで呼び出すことができないと規定されている≪OSa148≫．しかし，CheckObjectAccess (CheckAlarmOwnership)をコアを跨いで呼び出すことによる不都合が想定できないため，本仕様では呼び出すことができると規定した≪NOS0998≫.

3.9.56 CheckCounterOwnership

<table>
<tr><td colspan="2">C 言語 I/F</td><td colspan="3">ApplicationType CheckCounterOwnership
(CounterType CounterID)【NOS0598】</td></tr>
<tr><td colspan="2">パラメータ[in]</td><td>CounterID</td><td colspan="2">カウンタ ID</td></tr>
<tr><td colspan="2">パラメータ[in/out]</td><td colspan="3">－</td></tr>
<tr><td colspan="2">パラメータ[out]</td><td colspan="3">－</td></tr>
<tr><td colspan="2" rowspan="5">返り値</td><td colspan="2">所属する OSAP の OSAPID</td><td>OSAPID 取得成功≪NOS0600≫</td></tr>
<tr><td colspan="2" rowspan="4">INVALID_OSAPPLICATION</td><td>CounterID が不正≪NOS0601≫</td></tr>
<tr><td>不正な処理単位からの呼出し≪NOS0602≫</td></tr>
<tr><td>割込み禁止状態からの呼出し≪NOS0603≫</td></tr>
<tr><td>起動していないコアへの呼出し≪NOS1001≫</td></tr>
<tr><td rowspan="5">エラーフックに渡されるエラーコード</td><td>標準エラー</td><td colspan="3">－</td></tr>
<tr><td rowspan="4">拡張エラー</td><td>E_OS_ID</td><td>CounterID が不正≪NOS0519≫</td></tr>
<tr><td>E_OS_CALLEVEL</td><td>不正な処理単位からの呼出し≪NOS0411≫</td></tr>
<tr><td>E_OS_DISABLEDINT</td><td>割込み禁止状態からの呼出し≪NOS0412≫</td></tr>
<tr><td>E_OS_CORE</td><td>起動していないコアへの呼出し≪NOS0926≫</td></tr>
<tr><td colspan="2">コンフォーマンスクラス</td><td colspan="3">BCC1，BCC2，ECC1，ECC2</td></tr>
<tr><td colspan="2">スケーラビリティクラス</td><td colspan="3">SC3，SC4【NOS0599】
マルチコア：SC1，SC2【NOS1014】</td></tr>
<tr><td colspan="5">機能</td></tr>
<tr><td colspan="5">本システムサービスは，CounterID で指定されたカウンタが所属する OSAPID を返す.</td></tr>
</table>

機能仕様

CheckCounterOwnership は，CounterID で指定されたカウンタが所属する OSAPID を返す【NOS0600】.

CheckCounterOwnership 呼出し時に，引数で指定された CounterID が不正な場合，INVALID_OSAPPLICATION を返す【NOS0601】.

不正な処理単位から CheckCounterOwnership を呼び出した場合，INVALID_OSAPPLICATION を

返す【NOS0602】．また，割込み禁止状態で呼び出された場合も，INVALID_OSAPPLICATION を返す【NOS0603】〔NOS0383〕〔NOS0384〕〔NOS0385〕．

CheckCounterOwnership は，引数で指定されたカウンタが所属する OSAP の状態に関わらず，カウンタが所属する OSAPID を返す【NOS0604】．

マルチコア対応 OS における仕様

CheckCounterOwnership は，コアを跨いで呼び出すことができる【NOS1000】．

本 OS が起動していないコアに対して CheckCounterOwnership を呼び出した場合，INVALID_OSAPPLICATION を返す【NOS1001】．

AUTOSAR 仕様との違い

AUTOSAR 仕様では，CheckObjectOwnership は，コアを跨いで呼び出すことができないと規定されている≪OSa148≫．しかし，CheckObjectAccess (CheckCounterOwnership)をコアを跨いで呼び出すことによる不都合が想定できないため，本仕様では呼び出すことができると規定した≪NOS1000≫．

3.9.57 CheckScheduleTableOwnership

<table>
<tr><td colspan="2">C 言語 I/F</td><td colspan="3">ApplicationType CheckScheduleTableOwnership
(ScheduleTableType ScheduleTableID)【NOS0605】</td></tr>
<tr><td colspan="2">パラメータ[in]</td><td>ScheduleTableID</td><td colspan="2">スケジュールテーブル ID</td></tr>
<tr><td colspan="2">パラメータ[in/out]</td><td colspan="3">－</td></tr>
<tr><td colspan="2">パラメータ[out]</td><td colspan="3">－</td></tr>
<tr><td colspan="2" rowspan="5">返り値</td><td colspan="2">所属する OSAP の OSAPID</td><td>OSAPID 取得成功≪NOS0607≫</td></tr>
<tr><td colspan="2" rowspan="4">INVALID_OSAPPLICATION</td><td>ScheduleTableID が不正≪NOS0608≫</td></tr>
<tr><td>不正な処理単位からの呼出し≪NOS0609≫</td></tr>
<tr><td>割込み禁止状態からの呼出し≪NOS0610≫</td></tr>
<tr><td>起動していないコアへの呼出し≪NOS1003≫</td></tr>
<tr><td rowspan="5">エラーフックに渡されるエラーコード</td><td>標準エラー</td><td colspan="3">－</td></tr>
<tr><td rowspan="4">拡張エラー</td><td>E_OS_ID</td><td colspan="2">ScheduleTableID が不正≪NOS0519≫</td></tr>
<tr><td>E_OS_CALLEVEL</td><td colspan="2">不正な処理単位からの呼出し≪NOS0411≫</td></tr>
<tr><td>E_OS_DISABLEDINT</td><td colspan="2">割込み禁止状態からの呼出し≪NOS0412≫</td></tr>
<tr><td>E_OS_CORE</td><td colspan="2">起動していないコアへの呼出し≪NOS0926≫</td></tr>
<tr><td colspan="2">コンフォーマンスクラス</td><td colspan="3">BCC1，BCC2，ECC1，ECC2</td></tr>
<tr><td colspan="2">スケーラビリティクラス</td><td colspan="3">SC3，SC4【NOS0606】
マルチコア：SC1，SC2【NOS1015】</td></tr>
<tr><td colspan="5">機能</td></tr>
<tr><td colspan="5">本システムサービスは，ScheduleTableID で指定されたスケジュールテーブルが所属する OSAPID を返す.</td></tr>
</table>

機能仕様

CheckScheduleTableOwnership は，ScheduleTableID で指定されたスケジュールテーブルが所属する OSAPID を返す【NOS0607】.

CheckScheduleTableOwnership 呼出し時に，引数で指定された ScheduleTableID が不正な場合，

INVALID_OSAPPLICATION を返す【NOS0608】.

不正な処理単位から CheckScheduleTableOwnership を呼び出した場合，INVALID_OSAPPLICATION を返す【NOS0609】．また，割込み禁止状態で呼び出された場合も，INVALID_OSAPPLICATION を返す【NOS0610】〔NOS0383〕〔NOS0384〕〔NOS0385〕.

CheckScheduleTableOwnership は，引数で指定されたスケジュールテーブルが所属する OSAP の状態に関わらず，スケジュールテーブルが所属する OSAPID を返す【NOS0611】.

マルチコア対応 OS における仕様

CheckScheduleTableOwnership は，コアを跨いで呼び出すことができる【NOS1002】.

本 OS が起動していないコアに対して CheckScheduleTableOwnership を呼び出した場合，INVALID_OSAPPLICATION を返す【NOS1003】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，CheckObjectOwnership は，コアを跨いで呼び出すことができないと規定されている≪OSa148≫．しかし，CheckObjectAccess (CheckScheduleTableOwnership)をコアを跨いで呼び出すことによる不都合が想定できないため，本仕様では呼び出すことができると規定した≪NOS1002≫.

3.9.58 CheckObjectOwnership

AUTOSAR 仕様では，OS オブジェクトが所属する OSAPID を取得するシステムサービスとして，CheckObjectOwnership を規定している．

- CheckObjectOwnership を使用可能なスケーラビリティクラス【OS520】
- CheckObjectOwnership の概要【OS017】
- CheckObjectOwnership の処理【OS273】
- CheckObjectOwnership におけるエラー発生時の処理【OS274】

しかし，CheckObjectOwnership は，すべての OS オブジェクトに対応するため，可変長引数を使用している．可変長引数を使用すると，処理速度が遅くなり，引数の型チェックもできないため，本仕様では，OS オブジェクト毎に，所属する OSAPID を取得するシステムサービスを規定した≪NOS0570≫≪NOS0577≫≪NOS0584≫≪NOS0598≫≪NOS0605≫．

ATK2 では，AUTOSAR 仕様との互換性のために，CheckObjectOwnership をライブラリとして提供する．

<table>
<tr><td>C 言語 I/F</td><td colspan="3">ApplicationType CheckObjectOwnership(
ObjectTypeType ObjectType,
void …) 【IOS035】</td></tr>
<tr><td rowspan="2">パラメータ[in]</td><td>ObjectType</td><td colspan="2">…の OS オブジェクト型</td></tr>
<tr><td>…</td><td colspan="2">OS オブジェクトの ID</td></tr>
<tr><td>パラメータ[in/out]</td><td colspan="3">－</td></tr>
<tr><td>パラメータ[out]</td><td colspan="3">－</td></tr>
<tr><td rowspan="2">返り値</td><td>所属する OSAP の OSAPID</td><td colspan="2">OSAPID 取得成功≪IOS042≫</td></tr>
<tr><td>INVALID_OSAPPLICATION</td><td colspan="2">ObjectType が不正≪IOS044≫</td></tr>
<tr><td>コンフォーマンスクラス</td><td colspan="3">BCC1，BCC2，ECC1，ECC2</td></tr>
<tr><td>スケーラビリティクラス</td><td colspan="3">SC3，SC4【IOS041】
マルチコア：SC1，SC2【IOS208】</td></tr>
<tr><td colspan="4">機能</td></tr>
<tr><td colspan="4">本ライブラリは，ObjectType で指定された OS オブジェクトが所属する
OSAPID を返す．</td></tr>
</table>

機能仕様

CheckObjectOwnership は，与えられた引数から OS オブジェクト毎に規定したシステムサービス，CheckTaskOwnership，CheckISROwnership，CheckAlarmOwnership，CheckCounterOwnership，CheckScheduleTableOwnership を呼ぶ実装とし，返り値はこれらのシステムサービスからの返り値をそのまま使用する【IOS042】．そのため，エラーチェックおよびエラー検出時のエラーフックは，ライブラリから呼び出した各システムサービスによるエラーフックが呼び出されるものとする【IOS043】．

ObjectType で指定した OS オブジェクトが不正で呼び出すシステムサービスを特定できない場合，本ライブラリは INVALID_OSAPPLICATION を返す【IOS044】．この場合，エラーフックは呼び出されない【IOS176】.

3.9.59 TerminateApplication

C 言語 I/F		StatusType TerminateApplication(ApplicationType Application, RestartType RestartOption)【OS258】	
パラメータ[in]		Application	終了を行う OSAP の ID.
		RestartOption	OSAP のリスタートタスクを起動するか RESTART：リスタートタスクを起動する NO_RESTART：リスタートタスクを起動しない
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	E_OK	OSAP 終了成功≪COS3104≫
		E_OS_STATE	Application で指定された OSAP が既に利用可能な状態でない≪OS507≫≪OS508≫≪OS548≫
	拡張エラー	E_OS_ID	Application が不正≪OS493≫
			信頼 OSAP の指定≪NOS1133≫
			OSAP 固有のエラーフックによる 他の OSAP の指定≪NOS1134≫
		E_OS_VALUE	RestartOption が不正≪OS459≫≪NOS0886≫
		E_OS_ACCESS	他の非信頼 OSAP からの呼出し≪OS494≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			システム定義のエラーフックからの呼出し≪NOS1137≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪OS093≫
		E_OS_CORE	起動していないコアへの呼出し≪OSa120≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC3，SC4【OS536】	
機能			
Application で指定された OSAP を終了する.			

機能仕様

TerminateApplication は，Application で指定された OSAP を停止(OSAP に所属するすべてのタスク，C2ISR を強制終了し，OSAP に所属するすべての C1ISR，C2ISR の割込み要因を禁止)する.

RestartOption が RESTART の場合は，コンフィギュレーション時に指定されたリスタートタスクを起動する．OSAP が再起動される場合は状態を APPLICATION_RESTARTING に，再起動されない場合は状態を APPLICATION_TERMINATED に設定する．呼出し元の処理単位が Application で指定された OSAP に所属している場合は，TerminateApplication からリターンしない【OS287】.

TerminateApplication の呼出し時に指定された Application が不正であった場合，E_OS_ID を返す【OS493】. RestartOption が不正であった場合, E_OS_VALUE を返す【OS459】. TerminateApplication の呼出し元が非信頼 OSAP であり, Application で指定された OSAP に所属しない場合, E_OS_ACCESS を返す【OS494】.

TerminateApplication 呼出し時に，Application で指定された OSAP の状態が APPLICATION_TERMINATED であった場合，E_OS_STATE を返す【OS507】．Application で指定された OSAP の状態が APPLICATION_RESTARTING であり，呼出し元の処理単位が Application で指定された OSAP に所属しない場合, E_OS_STATE を返す【OS508】. Application で指定された OSAP の状態が APPLICATION_RESTARTING であり, 呼び出し元の処理単位が Application に属しており，かつ，RestartOption が RESTART の場合，E_OS_STATE を返す【OS548】.

コンフィギュレーション時に，Application で指定された OSAP に対して，リスタートタスクを設定していないにも関わらず，RestartOption に RESTART を指定した場合，E_OS_VALUE を返す【NOS0886】.

TerminateApplication は，OSAP 固有のエラーフックから呼出し可能で，自身の OSAP に対してのみ呼出し可能である≪OSa180≫．OSAP 固有のエラーフックからの呼出しで，Application に自身以外の OSAP が指定された場合，E_OS_ID を返す【NOS1134】．システム定義のエラーフックから呼び出した場合，E_OS_CALLEVEL を返す【NOS1137】.

TerminateApplication は，保護違反時処理の機能レベルが 2 および 3 の場合のみサポートする【NOS0389】．保護違反時処理の機能レベルが 2 で，Application に信頼 OSAP を指定して呼び出した場合，E_OS_ID を返す【NOS1133】.

使用上の注意

OSAP が 1 つも生成されていない場合，このサービスは利用できない．信頼 OSAP に所属する処理単位は任意の OSAP を停止することができる．非信頼 OSAP に属する処理単位は自身が所属する OSAP のみを停止できる【OS535】.

マルチコア対応 OS における仕様

TerminateApplication は，コアを跨いで呼び出すことができる【OSa149】．Application で指定した OSAP に所属する処理単位が，スピンロックを占有していた場合，OS は，占有されているスピンロックを解放する【OS614】.

TerminateApplication をコアを跨いで呼び出した場合，OSAP の状態変更と，OSAP 終了処理を開始するのみとし，OSAP 終了処理の完了を待たずに，呼び出し元にリターンする【NOS1050】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，TerminateApplication が複数のコアから同時に呼び出された場合，最初に受理した終了要求によって，OSAP 終了処理を実行し，終了処理が完了するまでは，その他の終了要求に対しては，一律 E_OK を返すと規定されている【OS615】. しかし，最初の TerminateApplication によって, 排他的に OSAP の状態を, APPLICATION_TERMINATED か APPLICATION_RESTARTING へと遷移し，以後の TerminateApplication 呼出し時は，既に利用可能な状態ではなくなっている実装しか想定されないことから，本仕様では削除した．つまり，本規定の状況では，2 回目以降の TerminateApplication に対しては，E_OS_STATE が返る≪OS507≫≪OS508≫.

3.9.60 AllowAccess

C 言語 I/F		StatusType AllowAccess(void)【OS501】	
パラメータ[in]		－	
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	E_OK	OSAP の再起動成功≪COS3104≫
		E_OS_STATE	不正な OSAP からの呼出し≪OS497≫
	拡張エラー	E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪OS093≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC3，SC4【OS547】	
機能			
本システムサービスは，OSAP を利用可能な状態とする．			

機能仕様

AllowAccess は，呼出し元の処理単位が所属する OSAP の状態が APPLICATION_RESTARTING の場合，その OSAP の状態を APPLICATION_ACCESSIBLE に設定し他の OSAP から利用可能な状態とする【OS498】.

呼び出し元の OSAP の状態が，APPLICATION_RESTARTING 以外の場合は，E_OS_STATE を返す【OS497】.

AllowAccess は，保護違反時処理の機能レベルが 2 および 3 の場合のみサポートする【NOS0391】.

3.9.61 GetApplicationState

C 言語 I/F		StatusType GetApplicationState(ApplicationType Application, ApplicationStateRefType Value)【OS499】	
パラメータ[in]		Application	OSAPID
パラメータ[in/out]		－	
パラメータ[out]		Value	OSAP 状態を格納するポインタ
返り値	標準エラー	E_OK	OSAP 状態の取得成功≪COS3104≫
	拡張エラー	E_OS_ID	Application が不正≪OS495≫
		E_OS_CALLEVEL	C1ISR からの呼出し≪IOS151≫
		E_OS_PARAM_POINTER	ポインタ引数が NULL≪OS566≫
		E_OS_ILLEGAL_ADDRESS	Value に不正なアドレスが指定された≪OS051≫
		E_OS_CORE	起動していないコアへの呼出し≪OSa120≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC3，SC4【OS537】	
機能			
本システムサービスは，Application で指定された OSAP の現在の状態を取得する．			

機能仕様

GetApplicationState は，Application で指定された OSAP 状態を Value に格納する【OS496】. GetApplicationState は，割込み禁止状態であっても，呼び出すことができる【NOS1150】.

GetApplicationState の呼出し時に指定された Application が不正であった場合，E_OS_ID を返す【OS495】.

マルチコア対応 OS における仕様

GetApplicationState は，コアを跨いで呼び出すことができる【NOS1016】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，割込み禁止の状態で，返り値の型が StatusType のシステムサービスを呼び出した場合，OS はシステムサービスの処理を行わず，E_OS_DISABLEDINT を返すと規定されている≪OS093≫．しかし，割込み禁止の状態で呼び出されたエラーフックやプロテクションフックから，GetApplicationState を使用することが想定されるため，本仕様では，割込み禁止状態でも呼び出すことができると規定した≪NOS1150≫.

3.9.62 GetNumberOfActivatedCores

C 言語 I/F	uint32 GetNumberOfActivatedCores(void)【OS672】	
パラメータ[in]	－	
パラメータ[in/out]	－	
パラメータ[out]	－	
返り値	OS が起動中のコア数	コア数取得成功≪OS626≫
コンフォーマンスクラス	BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス	マルチコア：SC1，SC2，SC3，SC4≪NOS0339≫	
機能		
本システムサービスは，現在の本 OS が起動中のコアの数を取得する．		

機能仕様

GetNumberOfActivatedCores は，本 OS が起動しているコアの数を返す【OS626】.

GetNumberOfActivatedCores の返り値は，コンフィギュレーション時に OsNumberOfCores で指定したコア数以下の値となる【OS673】. GetNumberOfActivatedCores は，割込み禁止状態でも呼び出すことができる【NOS1017】.

3.9.63 GetCoreID

C 言語 I/F	CoreIdType GetCoreID(void)【OS674】	
パラメータ[in]	－	
パラメータ[in/out]	－	
パラメータ[out]	－	
返り値	実行中のコア ID	コア ID 取得成功≪OS675≫
コンフォーマンスクラス	BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス	マルチコア：SC1，SC2，SC3，SC4≪NOS0339≫	
機能		
本システムサービスは，現在の処理単位を実行中のコアの ID 情報を取得する.		

機能仕様

GetCoreID は，GetCoreID を呼び出した処理単位が割付いているコア ID を返す【OS675】.

GetCoreID は，本 OS が起動する前でも呼び出すことができる【OS625】. GetCoreID は，割込み禁止状態でも呼び出すことができる【NOS1018】.

3.9.64 StartCore

<table>
<tr><td colspan="2">C 言語 I/F</td><td colspan="4">void StartCore(CoreIdType CoreID, StatusType* Status)【OS676】</td></tr>
<tr><td colspan="2">パラメータ[in]</td><td>CoreID</td><td colspan="3">起動するコア ID</td></tr>
<tr><td colspan="2">パラメータ[in/out]</td><td colspan="4">－</td></tr>
<tr><td rowspan="4">パラメータ[out]</td><td>標準エラー</td><td rowspan="4">Status</td><td>E_OK</td><td colspan="2">OS 管理のコアの起動成功≪OS677≫</td></tr>
<tr><td rowspan="3">拡張エラー</td><td>E_OS_ID</td><td colspan="2">コア ID が不正≪OSa150≫</td></tr>
<tr><td>E_OS_ACCESS</td><td colspan="2">OS 起動後のコアからの呼出し≪OS678≫</td></tr>
<tr><td>E_OS_STATE</td><td colspan="2">起動済みのコア ID の指定≪OS679≫≪OS680≫</td></tr>
<tr><td colspan="2">返り値</td><td colspan="4">－</td></tr>
<tr><td rowspan="6">エラーフックに渡されるエラーコード</td><td>標準エラー</td><td colspan="4">－</td></tr>
<tr><td rowspan="5">拡張エラー</td><td>E_OS_ID</td><td colspan="3">コア ID が不正≪NOS1022≫</td></tr>
<tr><td>E_OS_ACCESS</td><td colspan="3">OS 起動後のコアからの呼出し≪NOS1019≫</td></tr>
<tr><td>E_OS_STATE</td><td colspan="3">起動済みのコア ID の指定≪NOS1020≫≪NOS1021≫</td></tr>
<tr><td>E_OS_PARAM_POINTER</td><td colspan="3">ポインタ引数が NULL≪NOS1023≫</td></tr>
<tr><td>E_OS_ILLEGAL_ADDRESS</td><td colspan="3">Status に不正なアドレスが指定された(SC3, SC4 のみ)≪NOS1024≫</td></tr>
<tr><td colspan="2">コンフォーマンスクラス</td><td colspan="4">BCC1，BCC2，ECC1，ECC2</td></tr>
<tr><td colspan="2">スケーラビリティクラス</td><td colspan="4">マルチコア：SC1，SC2，SC3，SC4≪NOS0339≫</td></tr>
<tr><td colspan="6">機能</td></tr>
<tr><td colspan="6">本システムサービスは，CoreID で指定したコアを OS 管理のコアとして起動する．</td></tr>
</table>

機能仕様

StartCore によって起動されたコアは，本 OS で管理するコアとなる【OS677】.

本 OS は，OS 起動後のコアの起動はサポートしない【OS606】．OS 起動後のコアから，StartCore を呼び出した場合，Status に E_OS_ACCESS を格納する【OS678】．この場合，エラーコードを E_OS_ACCESS として，エラーフックを呼び出す【NOS1019】〔NOS0409〕.

CoreID に，既に StartCore によって起動されたコア ID を指定して呼び出した場合，Status に E_OS_STATE を格納する【OS679】．この場合，エラーコードを E_OS_STATE として，エラーフックを呼び出す【NOS1020】〔NOS0409〕.

CoreID に，既に StartNonAutosarCore によって起動されたコア ID を指定して呼び出した場合も，Status に E_OS_STATE を格納する【OS680】．この場合，エラーコードを E_OS_STATE として，エラーフックを呼び出す【NOS1021】〔NOS0409〕.

コンフィギュレーション時に OsNumberOfCores で指定した値以上の数値か, マルチコアシステム上で識別できない不正なコア ID を CoreID に指定した場合, Status に E_OS_ID を格納する【OSa150】. この場合，エラーコードを E_OS_ID として，エラーフックを呼び出す【NOS1022】〔NOS0409〕.

Status で与えたアドレスが，NULL であるポインタを指定した場合，エラーコードを E_OS_PARAM_POINTER として，エラーフックを呼び出す【NOS1023】〔NOS0409〕.

SC3，SC4 において，OS 起動後に StartCore を呼び出した際，Status で与えたアドレスが，呼出し元の OSAP から書込みできないメモリ領域を指している場合，エラーコードを E_OS_ILLEGAL_ADDRESS として，エラーフックを呼び出す【NOS1024】〔NOS0409〕.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，StartCore でエラーが発生しても，エラーフックは呼び出されないと規定されている【OS681】. しかし，本仕様では，返り値のデータ型が StatusType でないシステムサービスであっても，状況によってエラーフックが呼び出される場合があると規定したため，StartCore でエラーが発生した場合は, エラーフックが呼び出されると規定した≪NOS1019≫≪NOS1020≫≪NOS1021≫≪NOS1022≫≪NOS1023≫≪NOS1024≫.

3.9.65 StartNonAutosarCore

<table>
<tr><td colspan="2">C 言語 I/F</td><td colspan="4">void StartNonAutosarCore(
CoreIdType CoreID, StatusType* Status)【OS682】</td></tr>
<tr><td colspan="2">パラメータ[in]</td><td>CoreID</td><td colspan="3">起動するコア ID</td></tr>
<tr><td colspan="2">パラメータ[in/out]</td><td colspan="4">－</td></tr>
<tr><td rowspan="4">パラメータ
[out]</td><td>標準エラー</td><td rowspan="4">Status</td><td>E_OK</td><td colspan="2">OS 管理外のコアの起動成功≪OS683≫</td></tr>
<tr><td rowspan="3">拡張エラー</td><td>E_OS_ID</td><td colspan="2">コア ID が不正≪OS685≫</td></tr>
<tr><td>E_OS_ACCESS</td><td colspan="2">OS 起動後のコアに対する呼出し≪NOS1128≫</td></tr>
<tr><td>E_OS_STATE</td><td colspan="2">起動済みのコア ID の指定≪OS684≫≪NOS1025≫</td></tr>
<tr><td colspan="2">返り値</td><td colspan="4">－</td></tr>
<tr><td rowspan="6">エラーフックに渡されるエラーコード</td><td>標準エラー</td><td colspan="4">－</td></tr>
<tr><td rowspan="5">拡張エラー</td><td>E_OS_ID</td><td colspan="3">コア ID が不正≪NOS1030≫</td></tr>
<tr><td>E_OS_ACCESS</td><td colspan="3">OS 起動後のコアからの呼出し≪NOS1029≫</td></tr>
<tr><td>E_OS_STATE</td><td colspan="3">起動済みのコア ID の指定≪NOS1027≫≪NOS1028≫</td></tr>
<tr><td>E_OS_PARAM_POINTER</td><td colspan="3">ポインタ引数が NULL≪NOS1031≫</td></tr>
<tr><td>E_OS_ILLEGAL_ADDRESS</td><td colspan="3">Status に不正なアドレスが指定された(SC3，SC4 のみ)≪NOS1032≫</td></tr>
<tr><td colspan="2">コンフォーマンスクラス</td><td colspan="4">BCC1，BCC2，ECC1，ECC2</td></tr>
<tr><td colspan="2">スケーラビリティクラス</td><td colspan="4">マルチコア：SC1，SC2，SC3，SC4≪NOS0339≫</td></tr>
<tr><td colspan="6">機能</td></tr>
<tr><td colspan="6">本システムサービスは，CoreID で指定したコアを OS 管理外のコアとして起動する．</td></tr>
</table>

機能仕様

StartNonAutosarCore によって起動されたコアは，本 OS の管理外のコアとなる【OS683】.

CoreID に，既に StartNonAutosarCore によって起動されたコア ID を指定して呼び出した場合，Status に E_OS_STATE を格納する【OS684】．この場合，エラーコードを E_OS_STATE として，エラーフックを呼び出す【NOS1027】〔NOS0409〕.

CoreID に，既に StartCore によって起動されたコア ID を指定して呼び出した場合も，Status に E_OS_STATE を格納する【NOS1025】．この場合，エラーコードを E_OS_STATE として，エラーフックを呼び出す【NOS1028】〔NOS0409〕.

OS 起動後のコアを指定して，StartNonAutosarCore を呼び出した場合，Status に E_OS_ACCESS を格納する【NOS1128】．この場合，エラーコードを E_OS_ACCESS として，エラーフックを呼び出

す【NOS1029】〔NOS0409〕.

マルチコアシステム上で識別できない不正なコア ID を CoreID に指定した場合, Status に E_OS_ID を格納する【OS685】. この場合, エラーコードを E_OS_ID として, エラーフックを呼び出す【NOS1030】〔NOS0409〕.

Status で与えたアドレスが，NULL であるポインタを指定した場合，エラーコードを E_OS_PARAM_POINTER として，エラーフックを呼び出す【NOS1031】〔NOS0409〕.

SC3，SC4 において，OS 起動後に StartNonAutosarCore を呼び出した際，Status で与えたアドレスが，呼出し元の OSAP から書込みできないメモリ領域を指している場合，エラーコードを E_OS_ILLEGAL_ADDRESS として，エラーフックを呼び出す【NOS1032】〔NOS0409〕.

AUTOSAR 仕様との違い

本仕様では，返り値のデータ型が StatusType でないシステムサービスであっても，状況によってエラーフックが呼び出される場合があると規定したため, OS 起動後に StartNonAutosarCore を呼び出した場合は，エラーフックが呼び出されると規定した≪NOS1027≫≪NOS1028≫≪NOS1029≫≪NOS1030≫≪NOS1031≫≪NOS1032≫.

AUTOSAR 仕様では，StartNonAutosarCore における E_OS_ID，E_OS_STATE は標準エラーであると規定されているが，StartCore は拡張エラーとなっている．StartNonAutosarCore の誤記と思われるため，本仕様では，拡張エラーとして規定した.

3.9.66 GetSpinlock

C 言語 I/F		StatusType GetSpinlock(SpinlockIdType SpinlockId)【OS686】	
パラメータ[in]		SpinlockId	スピンロック ID
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	E_OK	スピンロック獲得成功≪OS688≫
		E_OS_LIMIT	ネスト回数の上限値超過≪NOS1035≫
	拡張エラー	E_OS_ID	SpinlockId が不正≪OS689≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_ACCESS	アクセスが許可されていないスピンロックが指定された(SC3，SC4 のみ)≪OS692≫
		E_OS_DISABLEDINT	DisableAllInterrupts による割込み禁止状態からの呼出し≪NOS1072≫
		E_OS_INTERFERENCE_DEADLOCK	呼び出し元のコアの処理単位によって既に獲得済み≪OS690≫
		E_OS_NESTING_DEADLOCK	指定した順序に反する順序での獲得≪OS691≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		マルチコア：SC1，SC2，SC3，SC4≪NOS0339≫	
機能			
本システムサービスは，SpinlockId で指定されたスピンロックを獲得する.			

機能仕様

GetSpinlock は，SpinlockId で指定されたスピンロックを獲得する. SpinlockId で指定されたスピンロックが他のコアの処理単位によって既に占有されている場合, 獲得できるまでスピンする【OS687】. スピンロックを正常に獲得できた場合, E_OK を返す【OS688】. GetSpinlock は, SuspendAllInterrupts, SuspendOSInterrupts による割込み禁止状態でも呼び出すことができる【OS693】. リソースを占有している処理単位であっても，GetSpinlock によってスピンロックを獲得することができる【OS694】.

GetSpinlock 呼出し時，割込み禁止状態でなければ，スピン中に割込みが発生した場合，割込みを受け付ける【NOS0338】.

GetSpinlock 呼出し時に指定された SpinlockId が不正であった場合，E_OS_ID を返す【OS689】. アクセスが許可されていないスピンロックが指定された場合，E_OS_ACCESS を返す【OS692】. SpinlockId で指定されたスピンロックが, 呼び出し元の処理単位と同じコアの処理単位によって既に占

有されている場合，E_OS_INTERFERENCE_DEADLOCK を返す【OS690】．コンフィギュレーション時に指定した獲得順序に反する順序でスピンロックを獲得しようとした場合，E_OS_NESTING_DEADLOCK を返す【OS691】． また，獲得順序が指定されていないスピンロックを獲得している場合に，他のスピンロックを獲得しようとした場合も，E_OS_NESTING_DEADLOCK を返す≪ OS661 ≫． DisableAllInterrupts による割込み禁止状態で呼び出した場合，E_OS_DISABLEDINT を返す【NOS1072】.

OS 割込み禁止スピンロックをネストして獲得する場合の，ネスト回数の上限値は実装定義である【NOS1033】．全割込み禁止スピンロックをネストして獲得する場合の，ネスト回数の上限値は実装定義である【NOS1034】. ネスト回数の上限値を超えて呼び出した場合, E_OS_LIMIT を返す【NOS1035】.

ATK2 では，OS 割込み禁止スピンロック獲得のネスト回数の上限値を，SuspendOSInterrupts のネスト回数との合計値で，255 と規定した≪IOS203≫．また，全割込み禁止スピンロック獲得のネスト回数の上限値を，SuspendAllInterrupts のネスト回数との合計値で，255 と規定した≪IOS202≫.

3.9.67 ReleaseSpinlock

C 言語 I/F		StatusType ReleaseSpinlock(SpinlockIdType SpinlockId)【OS695】	
パラメータ[in]		SpinlockId	スピンロック ID
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	E_OK	スピンロック獲得成功≪OS697≫
	拡張エラー	E_OS_ID	SpinlockId が不正≪OS698≫
		E_OS_ACCESS	アクセスが許可されていないスピンロックが指定された(SC3，SC4 のみ)≪OS700≫
		E_OS_STATE	SpinlockId が呼出し元で占有されていない≪OS699≫
		E_OS_DISABLEDINT	DisableAllInterrupts による割込み禁止状態からの呼出し≪NOS1073≫
		E_OS_NOFUNC	先に解放するべき別のスピンロックを占有している≪OS701≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		マルチコア：SC1，SC2，SC3，SC4≪NOS0339≫	
機能			
本システムサービスは，SpinlockId で指定されたスピンロックを解放する．解放されたスピンロックは他の処理単位から獲得できるようになる．			

機能仕様

ReleaseSpinlock は，呼び出し元の処理単位が占有しているスピンロックを解放する【OS696】．スピンロックを正常に解放できた場合，E_OK を返す【OS697】．ReleaseSpinlock は，SuspendAllInterrupts，SuspendOSInterrupts による割込み禁止状態でも呼び出すことができる≪OS693≫≪OS711≫．

ReleaseSpinlock 呼出し時に指定された SpinlockId が不正であった場合，E_OS_ID を返す【OS698】．呼び出し元の処理単位が占有していないスピンロックを指定した場合，E_OS_STATE を返す【OS699】．アクセスが許可されていないスピンロックが指定された場合，E_OS_ACCESS を返す【OS700】．呼び出し元の処理単位が，SpinlockId で指定したスピンロックを獲得した後に，別のスピンロックを獲得し，解放していない場合，E_OS_NOFUNC を返す【OS701】．DisableAllInterrupts による割込み禁止状態で呼び出した場合，E_OS_DISABLEDINT を返す【NOS1073】．

AUTOSAR 仕様との違い

AUTOSAR 仕様では，スピンロックとリソースの獲得，解放は LIFO で実行しなければならないと規定されている【OS702】．しかし，本要求の必要性が不明であることから，本仕様では削除した．

3.9.68 TryToGetSpinlock

C 言語 I/F		StatusType TryToGetSpinlock(SpinlockIdType SpinlockId, TryToGetSpinlockType* Success)【OS703】	
パラメータ[in]		SpinlockId	スピンロック ID
パラメータ[in/out]		－	
パラメータ[out]		Success	スピンロック獲得結果を格納する領域へのポインタ
返り値	標準エラー	E_OK	スピンロック獲得成功もしくは失敗≪OS705≫
		E_OS_LIMIT	ネスト回数の上限値超過≪NOS1036≫
	拡張エラー	E_OS_ID	SpinlockId が不正≪OS707≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_ACCESS	アクセスが許可されていないスピンロックが指定された(SC3，SC4 のみ)≪OS710≫
		E_OS_DISABLEDINT	DisableAllInterrupts による割込み禁止状態からの呼出し≪NOS1074≫
		E_OS_INTERFERENCE_DEADLOCK	呼び出し元のコアの処理単位によって既に獲得済み≪OS708≫
		E_OS_NESTING_DEADLOCK	指定した順序に反する順序での獲得≪OS709≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		マルチコア：SC1，SC2，SC3，SC4≪NOS0339≫	
機能			
本システムサービスは，SpinlockId で指定されたスピンロックを獲得する．			

機能仕様

TryToGetSpinlock は，SpinlockId で指定されたスピンロックを獲得する．SpinlockId で指定されたスピンロックを獲得できれば獲得し，他の処理単位が占有していることにより，獲得できなければすぐにリターンする【OS704】．スピンロックを正常に獲得できた場合，Success に TRYTOGETSPINLOCK_SUCCESS を格納し，他の処理単位が占有していることにより，獲得できなかった場合，Success に TRYTOGETSPINLOCK_NOSUCCESS を格納し，E_OK を返す【OS705】．返り値が E_OK でない場合に，Success に何を格納するかは規定しない【OS706】．TryToGetSpinlock は，SuspendAllInterrupts，SuspendOSInterrupts による割込み禁止状態でも呼び出すことができる

【OS711】．リソースを占有している処理単位であっても，TryToGetSpinlock によってスピンロックを獲得することができる【OS712】.

TryToGetSpinlock 呼出し時に指定された SpinlockId が不正であった場合, E_OS_ID を返す【OS707】. アクセスが許可されていないスピンロックが指定された場合，E_OS_ACCESS を返す【OS710】. SpinlockId で指定されたスピンロックが, 呼び出し元の処理単位と同じコアの処理単位によって既に占有されている場合，E_OS_INTERFERENCE_DEADLOCK を返す【OS708】．コンフィギュレーション時に指定した獲得順序に反する順序でスピンロックを獲得しようとした場合，E_OS_NESTING_DEADLOCK を返す【OS709】．また，獲得順序が指定されていないスピンロックを獲得している場合に，他のスピンロックを獲得しようとした場合も，E_OS_NESTING_DEADLOCK を返す≪OS661≫．DisableAllInterrupts による割込み禁止状態で呼び出した場合，E_OS_DISABLEDINT を返す【NOS1074】.

OS 割込み禁止スピンロックをネストして獲得する場合の，ネスト回数の上限値は実装定義である≪NOS1033≫．全割込み禁止スピンロックをネストして獲得する場合の，ネスト回数の上限値は実装定義である≪NOS1034≫．ネスト回数の上限値を超えて呼び出した場合，E_OS_LIMIT を返す【NOS1036】.

ATK2 では，OS 割込み禁止スピンロック獲得のネスト回数の上限値を，SuspendOSInterrupts のネスト回数との合計値で，255 と規定した≪IOS203≫．また，全割込み禁止スピンロック獲得のネスト回数の上限値を，SuspendAllInterrupts のネスト回数との合計値で，255 と規定した≪IOS202≫.

3.9.69 ShutdownAllCores

C 言語 I/F		void ShutdownAllCores(StatusType Error)【OS713】	
パラメータ[in]		Error	終了要因のエラーコード (OS 定義のものであること)
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値		－	
エラーフックに渡されるエラーコード	標準エラー	E_OS_ACCESS	非信頼 OSAP からの呼出し ≪NOS1037≫
	拡張エラー	－	
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		マルチコア：SC1，SC2，SC3，SC4≪NOS0339≫	
機能			
本システムサービスは，すべてのコアにおけるすべての OS サービスを終了する．			

機能仕様

ShutdownAllCores は同期 OS シャットダウンにより，すべてのコアにおけるすべての OS サービスを終了しシステム全体を停止する【OS714】．ShutdownAllCores からはリターンしない【OS715】．

OS で定義されていないエラーコードが指定された場合，Error を E_OS_SHUTDOWN_FATAL として ShutdownAllCores を呼び出したものとして，シャットダウン処理を行う【NOS1038】．また，不正な処理単位から呼び出した場合も，Error を E_OS_SHUTDOWN_FATAL として ShutdownAllCores を呼び出したものとして，シャットダウン処理を行う【NOS1039】．

ShutdownAllCores を呼び出していないコアでは，シャットダウンフックに E_OS_SHUTDOWN_OTHER_CORE が渡される【NOS1040】．

OS は，非信頼 OSAP に所属する処理単位からの OS シャットダウン要求を無視する【OS716】．ただし，エラーコードを E_OS_ACCESS として，エラーフックを呼び出す【NOS1037】．

ATK2 では，ShutdownAllCores で OS サービスを終了するために行う処理を，ターゲット定義とする【IOS209】．

3.9.70 RaiseInterCoreInterrupt

C 言語 I/F		StatusType RaiseInterCoreInterrupt(ISRType ISRID)【NOS0316】	
パラメータ[in]		ISRID	ICISR の ID
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	E_OK	正常終了≪COS3104≫
	拡張エラー	E_OS_ID	ISRID が不正≪NOS0074≫
			C1ISR，C2ISR の指定≪NOS1120≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_ACCESS	アクセスが許可されていない ISR が指定された≪OS056≫
			ICISR が所属する OSAP が利用可能でない(SC3, SC4 のみ)≪OS509≫
		E_OS_CORE	起動していないコアへの呼出し≪OSa120≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		マルチコア：SC1，SC2，SC3，SC4≪NOS0339≫	
機能			
本システムサービスは，ISRID で指定された ICISR を起動させる.			

機能仕様

RaiseInterCoreInterrupt は，ISRID で指定された ICISR が所属する OSAP が割り付けられたコアで，ISRID で指定された ICISR を起動させる【NOS0066】. RaiseInterCoreInterrupt は，割込み禁止状態でも，呼び出すことができる【NOS1110】.

RaiseInterCoreInterrupt 呼出し時に指定された ISRID が不正であった場合，E_OS_ID を返す【NOS0074】. ISRID に，C1ISR，C2ISR が指定された場合も，E_OS_ID を返す【NOS1120】.

ISRID で指定された ICISR が所属する OSAP が割付いているコアが，RaiseInterCoreInterrupt を呼び出した処理単位が所属する OSAP が割付いているコアと同一であっても，自身のコアに対してコア間割込みを発生させることができる【NOS1118】.

3.9.71 GetMutex

C 言語 I/F		StatusType GetMutex(MutexType MtxID) 【NOS0317】	
パラメータ[in]		MtxID	ミューテックス ID
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	E_OK	ミューテックス獲得成功≪COS3104≫
	拡張エラー	E_OS_ID	MtxID が不正≪NOS0101≫
		E_OS_ACCESS	MtxID で指定されたミューテックスが呼出し元タスクによって既に獲得済み≪NOS0094≫
			タスクがリソースを獲得した状態での呼出し≪NOS0097≫
			MtxID の上限優先度が呼出し元タスクの初期優先度より低い≪NOS0088≫
			アクセスが許可されていないミューテックスが指定された(SC3，SC4 のみ)≪OS056≫
			ミューテックスが所属する OSAP が利用可能でない(SC3，SC4 のみ)≪OS509≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪OS093≫
コンフォーマンスクラス		ECC1，ECC2(拡張タスクのみ)	
スケーラビリティクラス		マルチコア：SC1，SC2，SC3，SC4≪NOS0339≫	
機能			
本システムサービスは，MtxID で指定されたミューテックスを獲得する.			

機能仕様

GetMutex は，MtxID で指定されたミューテックスを獲得する【NOS0102】. ミューテックスを獲得した処理単位は上限優先度プロトコルにより現在優先度が引き上げられ，ReleaseMutex を呼び出すまでクリティカルセクションとなる【NOS0103】.

GetMutex 呼出し時に指定された MtxID が不正であった場合, E_OS_ID を返す【NOS0101】. MtxID で指定されたミューテックスが呼出し元の処理単位で既に獲得済みである場合，E_OS_ACCESS を返す【NOS0094】. また，呼出し元の処理単位がリソースを獲得した状態で呼び出した場合，E_OS_ACCESS を返す【NOS0097】. MtxID で指定されたミューテックスの上限優先度が，呼出し元の初期優先度よりも低い場合，E_OS_ACCESS を返す≪NOS0088≫.

3.9.72 ReleaseMutex

C 言語 I/F		StatusType ReleaseMutex(MtxType MtxID) 【NOS0318】	
パラメータ[in]		MtxID	ミューテックス ID
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	E_OK	ミューテックス解放成功≪COS3104≫
	拡張エラー	E_OS_ID	MtxID が不正≪NOS0105≫
		E_OS_NOFUNC	MtxID が呼出し元で獲得されていない≪NOS0106≫
			解放順序が獲得した LIFO 順でない≪NOS0093≫
		E_OS_ACCESS	アクセスが許可されていないミューテックスが指定された(SC3，SC4 のみ)≪OS056≫
			ミューテックスが所属する OSAP が利用可能でない(SC3，SC4 のみ)≪OS509≫
		E_OS_CALLEVEL	不正な処理単位からの呼出し≪OS088≫
			C1ISR からの呼出し≪IOS151≫
		E_OS_DISABLEDINT	割込み禁止状態からの呼出し≪OS093≫
コンフォーマンスクラス		ECC1，ECC2(拡張タスクのみ)	
スケーラビリティクラス		マルチコア：SC1，SC2，SC3，SC4≪NOS0339≫	
機能			
本システムサービスは，MtxID で指定されたミューテックスを解放する.			

機能仕様

ReleaseMutex は，MtxID で指定されたミューテックスを解放する【NOS0104】．ミューテックスを解放した時，OS はタスクの現在優先度をそのミューテックスを獲得した際の優先度に戻し，クリティカルセクションを終了する≪NOS0089≫.

ReleaseMutex 呼出し時に指定された MtxID が不正であった場合，E_OS_ID を返す【NOS0105】．MtxID で指定されたミューテックスが呼出し元の処理単位で獲得されていない場合，E_OS_NOFUNC を返す【NOS0106】．また，ミューテックスの解放順序が LIFO となっていない場合，E_OS_NOFUNC を返す≪NOS0093≫.

3.9.73 GetFaultyContext

C 言語 I/F	FaultyContextType GetFaultyContext(void)≪IOS181≫	
パラメータ[in]	－	
パラメータ[in/out]	－	
パラメータ[out]	－	
返り値	保護違反を起こした処理単位情報	取得成功≪IOS184≫
	FC_INVALID	処理単位を特定できない≪IOS191≫
		不正な処理単位からの呼出し≪IOS190≫
コンフォーマンスクラス	BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス	SC1，SC2，SC3，SC4【IOS183】	
機能		
本システムサービスは，保護違反を起こした処理単位情報を取得する．		

機能仕様

GetFaultyContext は，保護違反発生時に，プロテクションフックから呼び出すことで，保護違反を起こした処理単位情報を返す【IOS184】.

保護違反を起こした処理単位がタスクの場合，FC_TASK を返す【IOS185】．保護違反を起こした処理単位が C2ISR の場合，FC_C2ISR を返す【IOS186】．保護違反を起こした処理単位が ICISR の場合も，FC_C2ISR を返す【IOS230】．保護違反を起こした処理単位がシステム定義のフックルーチンの場合，FC_SYSTEM_HOOK を返す【IOS187】．保護違反を起こした処理単位が OSAP 固有のフックルーチンの場合，FC_OSAP_HOOK を返す【IOS188】．保護違反を起こした処理単位が信頼関数の場合，FC_TRUSTED_FUNC を返す【IOS189】.

プロテクションフック以外の処理単位から呼び出した場合，FC_INVALID を返す【IOS190】．C1ISR が保護違反を起こした場合など，何らかの理由で保護違反を起こした処理単位を特定できない場合も，FC_INVALID を返す【IOS191】.

マルチコア対応 OS における仕様

GetFaultyContext を呼び出した処理単位が割付いているコアにおける，保護違反を起こした処理単位情報を返す【IOS210】.

3.10 IOC 用システムサービス

3.10.1 IocSend_<IocId>[_<SenderId>]

C 言語 I/F		Std_ReturnType IocSend_<IocId>[_<SenderId>] (<Data> IN)【OS718】	
パラメータ[in]		IN	送信データ
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	IOC_E_OK	送信成功≪OS725≫
		IOC_E_LIMIT	キューフル≪OS726≫
	拡張エラー	IOC_E_NOK	不正な処理単位からの呼出し≪NOS1078≫
			割込み禁止状態からの呼出し≪NOS1081≫
			<IocId>で指定された IOC のセンダが所属する OSAP からの呼び出しでない≪NOS1085≫
			<SenderId>で指定されたセンダが所属する OSAP からの呼び出しでない≪NOS1085≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC3，SC4≪OSa151≫ マルチコア：SC1，SC2≪OSa151≫	
機能			
本システムサービスは，IOC(キューあり通信，単一通信)の送信処理を行う．1:1 通信の場合，<IocId>を IOC ID とし，_<SenderId>を省略したシステムサービスが，IOC 毎に生成される．N:1 通信の場合，<IocId>を IOC ID ，<SenderId>をセンダ ID としたシステムサービスが，IOC とセンダの組み合わせ毎に生成される．<Data>は送信するデータのデータ型である．			

機能仕様

IocSend は，データが正常に送信された場合，IOC_E_OK を返す【OS725】．IocSend は，キューにデータを格納するスペースがなく，データを送信出来ない場合，IOC_E_LIMIT を返す【OS726】．

IocSend は，送信したデータが受信されるのを待たずに，呼出し元にリターンする【OS719】．IocSend は，送信データの IOC バッファへの書込みを完了した後で，呼出し元にリターンする【OS720】．IocSend によるデータ送信では，データの送信順序を保証する【OS721】．IocSend によるデータ送信では，データ完全性を保証する【OS722】．

IN は，<Data>が汎整数型の場合は値渡しとし，汎整数型以外のデータ型は参照渡しとする【OS723】．IN が参照渡しである IocSend によりデータを送信した場合に，IN が示すデータに変更がないことを保証する【OS724】．

キューのサイズは，コンフィギュレーション時に静的に指定する【OS727】〔MCOS_Conf1001〕．

AUTOSAR 仕様との違い

AUTOSAR 仕様では，IocSend，および IocWrite によるデータの送信順序は保障され，異なるコアに割り当てられた複数のセンダが存在する場合，データを受信する順序は実装依存とすると規定されている≪OS721≫．しかし，マルチコアシステムにおいて，共有メモリを使用する際には，コア間の排他を行う必要があり，これによりデータ送信順序は保証されることは明確であるため，本規定は削除した．

3.10.2 IocWrite_<IocId>[_<SenderId>]

C 言語 I/F		Std_ReturnType IocWrite_<IocId>[_<SenderId>] (<Data> IN)≪OS718≫	
パラメータ[in]		IN	送信データ
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	IOC_E_OK	送信成功≪OS725≫
	拡張エラー	IOC_E_NOK	不正な処理単位からの呼出し≪NOS1078≫
			割込み禁止状態からの呼出し≪NOS1081≫
			<IocId>で指定された IOC のセンダが所属する OSAP からの呼び出しでない≪NOS1085≫
			<SenderId>で指定されたセンダが所属する OSAP からの呼び出しでない≪NOS1085≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC3，SC4≪OSa151≫ マルチコア：SC1，SC2≪OSa151≫	
機能			
本システムサービスは, IOC(キューなし通信, 単一通信)の送信処理を行う. 1:1 通信の場合, <IocId>を IOC ID とし, _<SenderId>を省略したシステムサービスが, IOC 毎に生成される. N:1 通信の場合, <IocId>を IOC ID ，<SenderId>をセンダ ID としたシステムサービスが，IOC とセンダの組み合わせ毎に生成される．<Data>は送信するデータのデータ型である.			

機能仕様

IocWrite は，データが正常に送信された場合，IOC_E_OK を返す≪OS725≫.

IocWrite は，送信したデータが受信されるのを待たずに，呼出し元にリターンする≪OS719≫. IocWrite は, 送信データの IOC バッファへの書込みを完了した後で, 呼出し元にリターンする≪OS720≫． IocWrite によるデータ送信では，データ完全性を保証する≪OS722≫.

IN は，<Data>が汎整数型の場合は値渡しとし，汎整数型以外のデータ型は参照渡しとする≪OS723≫．IN が参照渡しである IocSend によりデータを送信した場合に，IN が示すデータに変更がないことを保証する≪OS724≫.

3.10.3 IocSendGroup_<IocId>

C 言語 I/F		Std_ReturnType IocSendGroup_<IocId> (<Data1> IN1, <Data1> IN2, ...)【OS728】	
パラメータ[in]		IN*x*	送信データ
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	IOC_E_OK	送信成功≪OS735≫
		IOC_E_LIMIT	キューフル≪OS736≫
	拡張エラー	IOC_E_NOK	不正な処理単位からの呼出し≪NOS1078≫
			割込み禁止状態からの呼出し≪NOS1081≫
			<IocId>で指定された IOC のセンダが所属する OSAP からの呼び出しでない≪NOS1085≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC3，SC4≪OSa151≫ マルチコア：SC1，SC2≪OSa151≫	
機能			
本システムサービスは，IOC(キューあり通信，グループ通信)の送信処理を行う．<IocId>を IOC ID としたシステムサービスが，IOC 毎に生成される． <Data*x*>は送信するデータのデータ型である．			

機能仕様

IocSendGroup は，データが正常に送信された場合，IOC_E_OK を返す【OS735】．IocSendGroup は，キューにデータを格納するスペースがなく，データを送信出来ない場合，IOC_E_LIMIT を返す【OS736】.

IocSendGroup は，送信したデータが受信されるのを待たずに，呼出し元にリターンする【OS729】．IocSendGroup は，送信データの IOC バッファへの書込みを完了した後で，呼出し元にリターンする【OS730】．IocSendGroup によるデータ送信では，データの送信順序を保証する【OS731】．IocSendGroup によるデータ送信では，データ完全性を保証する【OS732】.

IN*x* は，<Data*x*>が汎整数型の場合は値渡しとし，汎整数型以外のデータ型は参照渡しとする【OS733】．IN*x* が参照渡しである IocSendGroup によりデータを送信した場合に，IN*x* が示すデータに変更がないことを保証する【OS734】.

キューのサイズは，コンフィギュレーション時に静的に指定する【OS737】〔MCOS_Conf1001〕.

3.10.4 IocWriteGroup_<IocId>

C 言語 I/F		Std_ReturnType IocWriteGroup_<IocId> (<Data1> IN1, <Data1> IN2, ...)≪OS728≫	
パラメータ[in]		IN*x*	送信データ
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	IOC_E_OK	送信成功≪OS735≫
	拡張エラー	IOC_E_NOK	不正な処理単位からの呼出し≪NOS1078≫
			割込み禁止状態からの呼出し≪NOS1081≫
			<IocId>で指定された IOC のセンダが所属する OSAP からの呼び出しでない≪NOS1085≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC3，SC4≪OSa151≫ マルチコア：SC1，SC2≪OSa151≫	
機能			
本システムサービスは，IOC(キューなし通信，グループ通信)の送信処理を行う．<IocId>を IOC ID としたシステムサービスが，IOC 毎に生成される． <Data*x*>は送信するデータのデータ型である．			

機能仕様

IocWriteGroup は，データが正常に送信された場合，IOC_E_OK を返す≪OS735≫.

IocWriteGroup は，送信したデータが受信されるのを待たずに，呼出し元にリターンする≪OS729≫. IocWriteGroup は，送信データの IOC バッファへの書込みを完了した後で，呼出し元にリターンする≪OS730≫．IocWriteGroup によるデータ送信では，データ完全性を保証する≪OS732≫.

IN*x* は，<Data*x*>が汎整数型の場合は値渡しとし，汎整数型以外のデータ型は参照渡しとする≪OS733≫．IN*x* が参照渡しである IocWriteGroup によりデータを送信した場合に，IN*x* が示すデータに変更がないことを保証する≪OS734≫.

3.10.5 IocReceive_<IocId>

C 言語 I/F		Std_ReturnType IocReceive_<IocId>(<Data> OUT)【OS738】	
パラメータ[in]		－	
パラメータ[in/out]		－	
パラメータ[out]		OUT	受信データ
返り値	標準エラー	IOC_E_OK	受信成功≪OS743≫
		IOC_E_NO_DATA	キューに受信データがない≪OS744≫
		IOC_E_LOST_DATA	キューフルが発生した≪OS745≫
	拡張エラー	IOC_E_NOK	不正な処理単位からの呼出し≪NOS1078≫
			割込み禁止状態からの呼出し≪NOS1081≫
			<IocId>で指定された IOC のレシーバが所属する OSAP からの呼び出しでない≪NOS1085≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC3，SC4≪OSa151≫ マルチコア：SC1，SC2≪OSa151≫	
機能			
本システムサービスは，IOC(キューあり通信，単一通信)の受信処理を行う．<IocId>を IOC ID としたシステムサービスが，IOC 毎に生成される．<Data>は受信するデータのデータ型である．			

機能仕様

IocReceive は，データを正常に受信した場合，IOC_E_OK を返す【OS743】．IocReceive は，キューにデータが存在しない場合，IOC_E_NO_DATA を返す【OS744】．IocReceive に対応する IocSend 呼出し時に，キューにデータを格納するスペースがなく，IOC_E_LIMIT が返った後で，IocReceive を呼び出した場合，データ受信を行った上で，IOC_E_LOST_DATA を返す【OS745】.

IocReceive は，正常に受信した場合，受信したデータを OUT に格納する【OS739】．IocReceive は，参照渡しした OUT へのデータ書込みを完了した上でリターンすることを保証する【OS742】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，受信通知機能を使用する場合，データを受信するコアでコールバック関数から IocReceive，および IocRead を呼び出すと規定されている【OS740】．しかし，本仕様では，受信通知機能をサポートしないため，削除した.

3.10.6 IocRead_<IocId>

C 言語 I/F		Std_ReturnType IocRead_<IocId>(<Data> OUT)≪OS738≫	
パラメータ[in]		－	
パラメータ[in/out]		－	
パラメータ[out]		OUT	受信データ
返り値	標準エラー	IOC_E_OK	受信成功≪OS743≫
	拡張エラー	IOC_E_NOK	不正な処理単位からの呼出し≪NOS1078≫
			割込み禁止状態からの呼出し≪NOS1081≫
			<IocId>で指定された IOC のレシーバが所属する OSAP からの呼び出しでない≪NOS1085≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC3，SC4≪OSa151≫ マルチコア：SC1，SC2≪OSa151≫	
機能			
本システムサービスは，IOC(キューなし通信，単一通信)の受信処理を行う．<IocId>を IOC ID としたシステムサービスが，IOC 毎に生成される．<Data>は受信するデータのデータ型である．			

機能仕様

IocRead は，データを正常に受信した場合，IOC_E_OK を返す≪OS743≫．

IocRead は，正常に受信した場合，受信したデータを OUT に格納する≪OS739≫．IocRead は，参照渡しした OUT へのデータ書込みを完了した上でリターンすることを保証する≪OS742≫．IocRead は，常に最後に送信されたデータを受信する【OS741】.

3.10.7 IocReceiveGroup_<IocId>

C 言語 I/F		Std_ReturnType IocReceiveGroup_<IocId> (<Data1> OUT1, <Data1> OUT2, ...)【OS746】	
パラメータ[in]		－	
パラメータ[in/out]		－	
パラメータ[out]		OUT*x*	受信データ
返り値	標準エラー	IOC_E_OK	受信成功≪OS751≫
		IOC_E_NO_DATA	キューに受信データがない≪OS752≫
		IOC_E_LOST_DATA	キューフルが発生した≪OS753≫
	拡張エラー	IOC_E_NOK	不正な処理単位からの呼出し≪NOS1078≫
			割込み禁止状態からの呼出し≪NOS1081≫
			<IocId>で指定された IOC のレシーバが所属する OSAP からの呼び出しでない≪NOS1085≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC3，SC4≪OSa151≫ マルチコア：SC1，SC2≪OSa151≫	
機能			
本システムサービスは，IOC(キューあり通信，グループ通信)の受信処理を行う．<IocId>を IOC ID としたシステムサービスが，IOC 毎に生成される．<Data*x*>は受信するデータのデータ型である．			

機能仕様

IocReceiveGroup は，データを正常に受信した場合，IOC_E_OK を返す【OS751】．IocReceiveGroup は，キューにデータが存在しない場合，IOC_E_NO_DATA を返す【OS752】．IocReceiveGroup に対応する IocSendGroup 呼出し時に，キューにデータを格納するスペースがなく，IOC_E_LIMIT が返った後で，IocReceiveGroup を呼び出した場合，データ受信を行った上で，IOC_E_LOST_DATA を返す【OS753】.

IocReceiveGroup は，正常に受信した場合，受信したデータを OUT*x* に格納する【OS747】. IocReceiveGroup は, 参照渡しした OUT*x* へのデータ書込みを完了した上でリターンすることを保証する【OS750】.

AUTOSAR 仕様との違い

AUTOSAR 仕様では，受信通知機能を使用する場合，データを受信するコアでコールバック関数から IocReceiveGroup，および IocReadGroup を呼び出すと規定されている【OS748】．しかし，本仕様では，受信通知機能をサポートしないため，削除した.

3.10.8 IocReadGroup_<IocId>

C 言語 I/F		Std_ReturnType IocReadGroup_<IocId> (<Data1> OUT1, <Data1> OUT2, ...)≪OS746≫	
パラメータ[in]		－	
パラメータ[in/out]		－	
パラメータ[out]		OUT*x*	受信データ
返り値	標準エラー	IOC_E_OK	受信成功≪OS751≫
	拡張エラー	IOC_E_NOK	不正な処理単位からの呼出し≪NOS1078≫
			割込み禁止状態からの呼出し≪NOS1081≫
			<IocId>で指定された IOC のレシーバが所属する OSAP からの呼び出しでない≪NOS1085≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC3，SC4≪OSa151≫ マルチコア：SC1，SC2≪OSa151≫	
機能			
本システムサービスは，IOC(キューなし通信，グループ通信)の受信処理を行う．<IocId>を IOC ID としたシステムサービスが，IOC 毎に生成される．<Data*x*>は受信するデータのデータ型である．			

機能仕様

IocReadGroup は，データを正常に受信した場合，IOC_E_OK を返す≪OS751≫．

IocReadGroup は，正常に受信した場合，受信したデータを OUT*x* に格納する≪OS747≫．IocReadGroup は，参照渡しした OUT*x* へのデータ書込みを完了した上でリターンすることを保証する≪OS750≫．IocReadGroup は，常に最後に送信されたデータを受信する【OS749】.

3.10.9 IocEmptyQueue_<IocId>

C 言語 I/F		Std_ReturnType IocEmptyQueue_<IocId>(void)【OS754】	
パラメータ[in]		－	
パラメータ[in/out]		－	
パラメータ[out]		－	
返り値	標準エラー	IOC_E_OK	キュー初期化成功≪OS755≫
	拡張エラー	IOC_E_NOK	不正な処理単位からの呼出し≪NOS1078≫
			割込み禁止状態からの呼出し≪NOS1081≫
			<IocId>で指定された IOC のレシーバが所属する OSAP からの呼び出しでない≪NOS1085≫
コンフォーマンスクラス		BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス		SC3，SC4≪OSa151≫ マルチコア：SC1，SC2≪OSa151≫	
機能			
本システムサービスは，IOC のキューの初期化を行う．<IocId>を IOC ID としたシステムサービスが，IOC 毎に生成される．			

機能仕様

IocEmptyQueue は，IOC のキューの初期化を行う【OS756】．キューに格納されたデータをすべて受信して初期化するよりも，IocEmptyQueue を使用するほうが効率的である．IocEmptyQueue は，レシーバのみが実行することができる≪NOS1075≫．

本 OS は，すべてのキューありの IOC に対して，IocEmptyQueue を提供する【OS755】.

3.11 フックルーチン

3.11.1 ErrorHook

C 言語 I/F	void ErrorHook(StatusType Error)【COS3802】〔COS1106〕	
パラメータ[in]	Error	発生したエラーのエラーコード
パラメータ[in/out]	－	
パラメータ[out]	－	
返り値	なし	
コンフォーマンスクラス	BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス	SC1，SC2，SC3，SC4≪NOS0319≫	
機能		
本フックルーチンは，エラーが発生した際に呼び出される．		

機能仕様

ErrorHook は，以下の時に呼び出される．

- システムサービスの返り値が E_OK 以外で，システムサービスが終了する時 【COS3805】〔COS1214〕.
- StatusType 以外の返り値を返すシステムサービスで，エラーが発生してシステムサービスが終了する時≪NOS0409≫
- アラーム満了時のアクションにて，タスク起動またはイベントのセットでエラーが発生した時【COS3806】.
- アラーム満了時のアクションにて，操作中のカウンタに対して，カウンタのティックをインクリメントする時【NOS0867】.
- スケジュールテーブル満了時のアクションにて，タスク起動またはイベントのセットでエラーが発生した時 【NOS0656】.

例外として，本フックルーチン内で呼び出したシステムサービスが E_OK 以外の値を返す場合，本フックルーチンは呼び出されず，返り値によってしかエラー検出できない【COS3807】.

3.11.2 PreTaskHook

C 言語 I/F	void PreTaskHook(void)【COS3810】〔COS1106〕
パラメータ[in]	－
パラメータ[in/out]	－
パラメータ[out]	－
返り値	なし
コンフォーマンスクラス	BCC1，BCC2，ECC1，ECC2
スケーラビリティクラス	SC1，SC2，SC3，SC4≪NOS0319≫
機能	
本フックルーチンは，新しいタスクを実行状態に切換えた後 OS から呼び出される.	

機能仕様

PreTaskHook は，新しいタスクを実行状態に切換えた後 OS から呼び出される【COS3813】.

3.11.3 PostTaskHook

C 言語 I/F	void PostTaskHook(void)【COS3816】〔COS1106〕
パラメータ[in]	－
パラメータ[in/out]	－
パラメータ[out]	－
返り値	なし
コンフォーマンスクラス	BCC1，BCC2，ECC1，ECC2
スケーラビリティクラス	SC1，SC2，SC3，SC4≪NOS0319≫
機能	
本フックルーチンは，実行されていたタスクが実行状態から他の状態に切り換わる前に OS から呼び出される.	

機能仕様

PostTaskHook は，実行されていたタスクが実行状態から他の状態に切り換わる前に OS から呼び出される【COS3819】.

3.11.4 StartupHook

C 言語 I/F	void StartupHook(void)【COS3822】〔COS1106〕〔COS1148〕
パラメータ[in]	－
パラメータ[in/out]	－
パラメータ[out]	－
返り値	なし
コンフォーマンスクラス	BCC1，BCC2，ECC1，ECC2
スケーラビリティクラス	SC1，SC2，SC3，SC4≪NOS0319≫
機能	
本フックルーチンは，OS の初期化完了後，スケジューラが動作する前に呼び出される．	

機能仕様

StartupHook は，OS の初期化完了後，スケジューラが動作する前に呼び出される【COS3825】.
本フックルーチンではデバイスドライバの初期化などの処理を行うことができる．

3.11.5 ShutdownHook

C 言語 I/F	void ShutdownHook(StatusType Error)【COS3828】〔COS1106〕	
パラメータ[in]	Error	発生したエラーのエラーコード
パラメータ[in/out]	－	
パラメータ[out]	－	
返り値	なし	
コンフォーマンスクラス	BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス	SC1，SC2，SC3，SC4≪NOS0319≫	
機能		
本フックルーチンは OS 処理終了中に呼び出される.		

機能仕様

ShutdownHook は，OS 処理終了中に呼び出される【COS3831】.

3.11.6 ProtectionHook

C 言語 I/F	ProtectionReturnType ProtectionHook(StatusType FatalError) 【OS538】〔COS1106〕	
パラメータ[in]	FatalError	発生した保護違反要因のエラーコード
パラメータ[in/out]	－	
パラメータ[out]	－	
返り値	プロテクションフック実行後の OS に対する処理要求．以下の 5 つのいずれかを返す必要がある． ・ PRO_IGNORE ・ PRO_TERMINATETASKISR ・ PRO_TERMINATEAPPL ・ PRO_TERMINATEAPPL_RESTART ・ PRO_SHUTDOWN	
コンフォーマンスクラス	BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス	SC1【NOS0341】　　SC2，SC3，SC4【OS542】	
機能		
本フックルーチンは，メモリ保護違反などの深刻なエラーが発生した際に呼び出される．		

機能仕様

ProtectionHook は，メモリ保護違反などの深刻なエラーが発生した際に呼び出される．
本フックルーチンの返り値によって OS は以下の処理を行う．

- PRO_IGNORE：処理を行わない
- PRO_TERMINATETASKISR：エラーを起こしたタスクまたは C2ISR の強制終了
- PRO_TERMINATEAPPL：エラーを起こしたタスクまたは C2ISR が所属する OSAP の強制終了
- PRO_TERMINATEAPPL_RESTART：エラーを起こしたタスクまたは C2ISR が所属する OSAP の強制終了と OSAP の再起動
- PRO_SHUTDOWN：OS シャットダウン

ProtectionHook からの返り値が未定義であった場合，OS は E_OS_PROTECTION_FATAL を終了要因のエラーコードとして OS シャットダウンを行う≪NOS0164≫．

3.11.7 StartupHook_<App>

C 言語 I/F	void StartupHook_<App>(void)【OS539】〔COS1106〕〔COS1148〕
パラメータ[in]	－
パラメータ[in/out]	－
パラメータ[out]	－
返り値	なし
コンフォーマンスクラス	BCC1，BCC2，ECC1，ECC2
スケーラビリティクラス	SC3，SC4【OS543】 マルチコア：SC1，SC2【NOS1041】
機能	
本フックルーチンは OS の初期化中，システムのスタートアップフック実行後に呼び出される．	

機能仕様

OSAP 固有の StartupHook は，OS の初期化中，システムのスタートアップフック実行後に呼び出される．複数の OSAP が本フックルーチンを定義している場合の呼出し順序は実装定義である≪NOS0128≫．

3.11.8 ErrorHook_<App>

C 言語 I/F	void ErrorHook_<App>(StatusType Error)【OS540】〔COS1106〕	
パラメータ[in]	Error	発生したエラーのエラーコード
パラメータ[in/out]	－	
パラメータ[out]	－	
返り値	なし	
コンフォーマンスクラス	BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス	SC3，SC4【OS544】 マルチコア：SC1，SC2【NOS1042】	
機能		
本フックルーチンは本フックルーチンが所属する OSAP のタスクまたは C2ISR がエラーを起こした際に呼び出される． システムのエラーフックが有効な場合は，システムのエラーフック実行後に呼び出される．		

機能仕様

OSAP 固有の ErrorHook は，本フックルーチンが所属する OSAP のタスク，C2ISR が，エラーを起こした際に呼び出される．システムのエラーフックが有効な場合は，システムのエラーフック実行後に呼び出される．

3.11.9 ShutdownHook_<App>

C 言語 I/F	void ShutdownHook_<App>(StatusType Error)【OS541】 〔COS1106〕	
パラメータ[in]	Error	発生したエラーのエラーコード
パラメータ[in/out]	－	
パラメータ[out]	－	
返り値	なし	
コンフォーマンスクラス	BCC1，BCC2，ECC1，ECC2	
スケーラビリティクラス	SC3，SC4【OS545】 マルチコア：SC1，SC2【NOS1043】	
機能		
本フックルーチンは，OS の終了時に呼び出される．システム定義のシャットダウンフックが有効な場合，システム定義のシャットダウンフックより前に呼び出される．		

機能仕様

OSAP 固有の ShutdownHook は，OS の終了時に呼び出される．システム定義のシャットダウンフックが有効な場合，システム定義のシャットダウンフックより前に呼び出される．複数の OSAP が本フックルーチンを定義している場合の呼出し順序は実装定義である≪NOS0127≫．

4. リファレンス

4.1　システムサービス一覧

OS 管理

AppModeType GetActiveApplicationMode(void)
void StartOS(AppModeType Mode)
void ShutdownOS(StatusType Error)
void StartCore(CoreIdType CoreID, StatusType* Status)
void StartNonAutosarCore(CoreIdType CoreID, StatusType* Status)
void ShutdownAllCores(StatusType Error)
CoreIdType GetCoreID(void)
uint32 GetNumberOfActivatedCores(void)

タスク管理

StatusType ActivateTask(TaskType TaskID)
StatusType TerminateTask(void)
StatusType ChainTask(TaskType TaskID)
StatusType Schedule(void)
StatusType GetTaskID(TaskRefType TaskID)
StatusType GetTaskState(TaskType TaskID, TaskStateRefType State)

割込み管理

void EnableAllInterrupts(void)
void DisableAllInterrupts(void)
void ResumeAllInterrupts(void)
void SuspendAllInterrupts(void)
void ResumeOSInterrupts(void)
void SuspendOSInterrupts(void)
ISRType GetISRID(void)
StatusType EnableInterruptSource(ISRType EnableISR)
StatusType DisableInterruptSource(ISRType DisableISR)

イベント管理

StatusType SetEvent(TaskType TaskID, EventMaskType Mask)
StatusType ClearEvent(EventMaskType Mask)
StatusType GetEvent(TaskType TaskID, EventMaskRefType Event)
StatusType WaitEvent(EventMaskType Mask)

リソース管理

StatusType GetResource(ResourceType ResID)

StatusType ReleaseResource(ResourceType ResID)

カウンタ制御

StatusType IncrementCounter(CounterType CounterID)

ソフトウェアフリーランタイマ制御

StatusType GetCounterValue(CounterType CounterID, TickRefType Value)

StatusType GetElapsedValue(CounterType CounterID,
TickRefType Value, TickRefType ElapsedValue)

アラーム制御

StatusType GetAlarmBase(AlarmType AlarmID, AlarmBaseRefType Info)

StatusType GetAlarm(AlarmType AlarmID, TickRefType Tick)

StatusType SetRelAlarm(AlarmType AlarmID, TickType increment, TickType cycle)

StatusType SetAbsAlarm(AlarmType AlarmID, TickType start, TickType cycle)

StatusType CancelAlarm(AlarmType AlarmID)

スケジュールテーブル制御

StatusType StartScheduleTableRel(ScheduleTableType ScheduleTableID, TickType Offset)

StatusType StartScheduleTableAbs(ScheduleTableType ScheduleTableID, TickType Start)

StatusType StopScheduleTable(ScheduleTableType ScheduleTableID)

StatusType NextScheduleTable(ScheduleTableType ScheduleTableID_From,
ScheduleTableType ScheduleTableID_To)

StatusType StartScheduleTableSynchron(ScheduleTableType ScheduleTableID)

StatusType SyncScheduleTable(ScheduleTableType ScheduleTableID, TickType Value)

StatusType SetScheduleTableAsync(ScheduleTableType ScheduleTableID)

StatusType GetScheduleTableStatus(ScheduleTableType ScheduleTableID,
ScheduleTableStatusRefType ScheduleStatus)

OSAP 制御

ApplicationType GetApplicationID(void)

StatusType CallTrustedFunction(TrustedFunctionIndexType FunctionIndex,
TrustedFunctionParameterRefType FunctionParams)

AccessType CheckISRMemoryAccess(ISRType ISRID,
MemoryStartAddressType Address, MemorySizeType Size)

AccessType CheckTaskMemoryAccess(TaskType TaskID,
MemoryStartAddressType Address, MemorySizeType Size)
ObjectAccessType CheckTaskAccess(ApplicationType ApplID, TaskType TaskID)
ObjectAccessType CheckISRAccess(ApplicationType ApplID, ISRType ISRID)
ObjectAccessType CheckAlarmAccess(ApplicationType ApplID, AlarmType AlarmID)
ObjectAccessType CheckResourceAccess(ApplicationType ApplID, ResourceType ResID)
ObjectAccessType CheckCounterAccess(ApplicationType ApplID, CounterType CounterID)
ObjectAccessType CheckScheduleTableAccess
(ApplicationType ApplID, ScheduleTableType ScheduleTableID)
ApplicationType CheckTaskOwnership(TaskType TaskID)
ApplicationType CheckISROwnership(ISRType ISRID)
ApplicationType CheckAlarmOwnership(AlarmType AlarmID)
ApplicationType CheckCounterOwnership(CounterType CounterID)
ApplicationType CheckScheduleTableOwnership(ScheduleTableType ScheduleTableID)
StatusType TerminateApplication(ApplicationType Application, RestartType RestartOption)
StatusType AllowAccess(void)
StatusType GetApplicationState(ApplicationType Application, ApplicationStateRefType Value)
FaultyContextType GetFaultyContext(void)

IOC

Std_ReturnType IocSend_<IocId>[_<SenderId>](<Data> IN)
Std_ReturnType IocWrite_<IocId>[_<SenderId>](<Data> IN)
Std_ReturnType IocSendGroup_<IocId> (<Data1> IN1, <Data2> IN2, …)
Std_ReturnType IocWriteGroup_<IocId> (<Data1> IN1, <Data2> IN2, …)
Std_ReturnType IocReceive_<IocId> (<Data> OUT)
Std_ReturnType IocRead_<IocId> (<Data> OUT)
Std_ReturnType IocReceiveGroup_<IocId> (<Data1> OUT1, <Data2> OUT2, …)
Std_ReturnType IocReadGroup_<IocId> (<Data1> OUT1, <Data2> OUT2, …)
Std_ReturnType IocEmptyQueue_<IocId> (void)

スピンロック管理

StatusType GetSpinlock(SpinlockIdType SpinlockId)
StatusType ReleaseSpinlock(SpinlockIdType SpinlockId)
StatusType TryToGetSpinlock(SpinlockIdType SpinlockId, TryToGetSpinlockType* Success)

コア間割込み管理

StatusType RaiseInterCoreInterrupt(ISRType ISRID)

ミューテックス管理

StatusType GetMutex(MutexType MtxID)

StatusType ReleaseMutex(MtxType MtxID)

4.2　フックルーチン

void ErrorHook(StatusType Error)

void PreTaskHook(void)

void PostTaskHook(void)

void StartupHook(void)

void ShutdownHook(StatusType Error)

ProtectionReturnType ProtectionHook(StatusType FatalError)

void StartupHook_<App>(void)

void ErrorHook_<App>(StatusType Error)

void ShutdownHook_<App>(StatusType Error)

4.3　データ型一覧

表 4-1 データ型一覧

データ型名	概要
StatusType	システムサービス要求に対する結果
AppModeType	アプリケーションモード型
OSServiceIdType	システムサービス ID 型
TaskType	タスク ID
TaskRefType	タスク ID へのポインタ型
TaskStateType	タスク状態
TaskStateRefType	タスク状態へのポインタ型
ISRType	ISRID
EventMaskType	イベントマスク型
EventMaskRefType	イベントマスクへのポインタ型
ResourceType	リソース ID
CounterType	カウンタ ID
TickType	ティック
TickRefType	ティックへのポインタ型
AlarmType	アラーム ID
AlarmBaseType	アラーム情報
AlarmBaseRefType	アラーム情報へのポインタ型
ScheduleTableType	スケジュールテーブル ID
ScheduleTableStatusType	スケジュールテーブル状態型
ScheduleTableStatusRefType	スケジュールテーブル状態へのポインタ型
PhysicalTimeType	ティックから時間に換算するマクロで使用される型
ApplicationType	OSAPID
ApplicationStateType	OSAP の状態を示すデータ型
ApplicationStateRefType	OSAP の状態を示すデータへのポインタ型
TrustedFunctionIndexType	信頼関数 ID
TrustedFunctionParameterRefType	信頼関数パラメータへのポインタ型
AccessType	メモリ領域アクセス権
ObjectAccessType	OS オブジェクトアクセス権
ObjectTypeType	OS オブジェクト種別
MemoryStartAddressType	メモリ領域先頭へのポインタ型
MemorySizeType	メモリ領域サイズ
RestartType	OSAP 強制終了時のリスタートタスク起動有無

データ型名	概要
ProtectionReturnType	保護違反に対する OS の処理種別
Std_ReturnType	IOC 用システムサービス要求に対する結果
IocType	IOC ID
SenderIdType	センダ ID
CoreIdType	コア ID
SpinlockIdType	スピンロック ID
TryToGetSpinlockType	TryToGetSpinlock 実行結果情報
MutexType	ミューテックス ID
StackType	スタック領域を用意するためのデータ型
TimeType	ハードウェアカウンタにおける 1 ティック当たりの実時間をナノ秒単位で管理するためのデータ型
FaultyContextType	保護違反を起こした処理単位情報を管理するためのデータ型

4.4　定数とマクロ一覧

4.4.1　OS 定数一覧

表 4-2 OS 定数一覧

定数名	概要
RUNNING	タスクが実行状態
WAITING	タスクが待ち状態である
READY	タスクが実行可能状態である
SUSPENDED	タスクが休止状態である
INVALID_TASK	いずれのタスクの ID でもないタスク ID
INVALID_ISR	いずれの ISR の ID でもない ISR ID
INVALID_OSAPPLICATION	いずれの OSAP でもない OSAPID
OS_CORE_ID_MASTER	マスタコアのコア ID を示す定数
OS_CORE_ID_<No>	各コアのコア ID を示す定数
OSMAXALLOWEDVALUE_<Cnt>	カウンタオブジェクト<Cnt>に対してシステムサービスで指定できるティックの最大値
OSTICKSPERBASE_<Cnt>	カウンタオブジェクト<Cnt>のカウンタ固有の値(OS は不使用).
OSMINCYCLE_<Cnt>	カウンタオブジェクト<Cnt>のアラームのセットに指定できるサイクルの最小値
SCHEDULETABLE_STOPPED	ScheduleTableStatusType 型のスケジュールテーブルが停止状態であることを示す定数
SCHEDULETABLE_NEXT	ScheduleTableStatusType 型のスケジュールテーブルが切換え待ち状態であることを示す定数
SCHEDULETABLE_WAITING	ScheduleTableStatusType 型のスケジュールテーブルが同期待ち状態であることを示す定数
SCHEDULETABLE_RUNNING	ScheduleTableStatusType 型のスケジュールテーブルが動作状態であることを示す定数
SCHEDULETABLE_ RUNNING_AND_SYNCHRONOUS	ScheduleTableStatusType 型のスケジュールテーブルが同期動作状態であることを示す定数
APPLICATION_ACCESSIBLE	ApplicationStateType 型の OSAP が利用可能状態であることを示す定数
APPLICATION_RESTARTING	ApplicationStateType 型の OSAP が再起動状態であることを示す定数
APPLICATION_TERMINATED	ApplicationStateType 型の OSAP が終了状態であることを示す定数

定数名	概要
ACCESS	ObjectAccessType 型の OS オブジェクトにアクセスできることを示す定数
NO_ACCESS	ObjectAccessType 型の OS オブジェクトにアクセスできないことを示す定数
OBJECT_TASK	ObjectTypeType 型の OS オブジェクトがタスクであることを示す定数
OBJECT_ISR	ObjectTypeType 型の OS オブジェクトが ISR であることを示す定数
OBJECT_ALARM	ObjectTypeType 型の OS オブジェクトがアラームであることを示す定数
OBJECT_RESOURCE	ObjectTypeType 型の OS オブジェクトがリソースであることを示す定数
OBJECT_COUNTER	ObjectTypeType 型の OS オブジェクトがカウンタであることを示す定数
OBJECT_SCHEDULETABLE	ObjectTypeType 型の OS オブジェクトがスケジュールテーブルであることを示す定数
RESTART	RestartType 型のリスタートタスクを起動することを示す定数
NO_RESTART	RestartType 型のリスタートタスクを起動しないことを示す定数
PRO_IGNORE	ProtectionReturnType 型の保護違反時処理を行わないことを示す定数
PRO_TERMINATETASKISR	ProtectionReturnType 型の保護違反時処理としてタスク，C2ISR を強制終了することを示す定数
PRO_TERMINATEAPPL	ProtectionReturnType 型の保護違反時処理として OSAP を強制終了することを示す定数
PRO_TERMINATEAPPL_RESTART	ProtectionReturnType 型の保護違反時処理として OSAP を強制終了し，リスタートタスクを起動することを示す定数
PRO_SHUTDOWN	ProtectionReturnType 型の保護違反時処理として OS をシャットダウンすることを示す定数
FC_TASK	タスクが保護違反を起こしたことを示す定数
FC_C2ISR	C2ISR が保護違反を起こしたことを示す定数
FC_SYSTEM_HOOK	システム定義のフックルーチンが保護違反を起こしたことを示す定数

定数名	概要
FC_OSAP_HOOK	OSAP 固有のフックルーチンが保護違反を起こしたことを示す定数
FC_TRUSTED_FUNC	信頼関数が保護違反を起こしたことを示す定数
FC_INVALID	保護違反を起こした処理単位を特定できないことを示す定数
INVALID_APPMODETYPE	GetActiveApplicationMode がエラーを検出したときに返る定数

4.4.2　システムサービス ID 一覧

表 4-3 システムサービス ID 一覧

システムサービス ID 名	対応するシステムサービス名
OSServiceId_GetActiveApplicationMode	GetActiveApplicationMode
OSServiceId_StartOS	StartOS
OSServiceId_ShutdownOS	ShutdownOS
OSServiceId_ActivateTask	ActivateTask
OSServiceId_TerminateTask	TerminateTask
OSServiceId_ChainTask	ChainTask
OSServiceId_Schedule	Schedule
OSServiceId_GetTaskID	GetTaskID
OSServiceId_GetTaskState	GetTaskState
OSServiceId_EnableAllInterrupts	EnableAllInterrupts
OSServiceId_DisableAllInterrupts	DisableAllInterrupts
OSServiceId_ResumeAllInterrupts	ResumeAllInterrupts
OSServiceId_SuspendAllInterrupts	SuspendAllInterrupts
OSServiceId_ResumeOSInterrupts	ResumeOSInterrupts
OSServiceId_SuspendOSInterrupts	SuspendOSInterrupts
OSServiceId_GetISRID	GetISRID
OsServiceId_EnableInterruptSource	EnableInterruptSource
OsServiceId_DisableInterruptSource	DisableInterruptSource
OSServiceId_SetEvent	SetEvent
OSServiceId_ClearEvent	ClearEvent
OSServiceId_GetEvent	GetEvent
OSServiceId_WaitEvent	WaitEvent
OSServiceId_GetResource	GetResource
OSServiceId_ReleaseResource	ReleaseResource
OSServiceId_IncrementCounter	IncrementCounter
OSServiceId_GetCounterValue	GetCounterValue
OSServiceId_GetElapsedValue	GetElapsedValue
OSServiceId_GetAlarmBase	GetAlarmBase
OSServiceId_GetAlarm	GetAlarm
OSServiceId_SetRelAlarm	SetRelAlarm
OSServiceId_SetAbsAlarm	SetAbsAlarm
OSServiceId_CancelAlarm	CancelAlarm

システムサービス ID 名	対応するシステムサービス名
OSServiceId_StartScheduleTableRel	StartScheduleTableRel
OSServiceId_StartScheduleTableAbs	StartScheduleTableAbs
OSServiceId_StopScheduleTable	StopScheduleTable
OSServiceId_NextScheduleTable	NextScheduleTable
OSServiceId_StartScheduleTableSynchron	StartScheduleTableSynchron
OSServiceId_SyncScheduleTable	SyncScheduleTable
OSServiceId_SetScheduleTableAsync	SetScheduleTableAsync
OSServiceId_GetScheduleTableStatus	GetScheduleTableStatus
OSServiceId_GetApplicationID	GetApplicationID
OSServiceId_CallTrustedFunction	CallTrustedFunction
OSServiceId_CheckISRMemoryAccess	CheckISRMemoryAccess
OSServiceId_CheckTaskMemoryAccess	CheckTaskMemoryAccess
OSServiceId_CheckTaskAccess	CheckTaskAccess
OSServiceId_CheckISRAccess	CheckISRAccess
OSServiceId_CheckAlarmAccess	CheckAlarmAccess
OSServiceId_CheckResourceAccess	CheckResourceAccess
OSServiceId_CheckCounterAccess	CheckCounterAccess
OSServiceId_CheckScheduleTableAccess	CheckScheduleTableAccess
OSServiceId_CheckSpinlockAccess	CheckSpinlockAccess
OSServiceId_CheckTaskOwnership	CheckTaskOwnership
OSServiceId_CheckISROwnership	CheckISROwnership
OSServiceId_CheckAlarmOwnership	CheckAlarmOwnership
OSServiceId_CheckCounterOwnership	CheckCounterOwnership
OSServiceId_CheckScheduleTableOwnership	CheckScheduleTableOwnership
OSServiceId_TerminateApplication	TerminateApplication
OSServiceId_AllowAccess	AllowAccess
OSServiceId_GetApplicationState	GetApplicationState
IOCServiceId_IOC_Send	IocSend
IOCServiceId_IOC_Write	IocWrite
IOCServiceId_IOC_SendGroup	IocSendGroup
IOCServiceId_IOC_WriteGroup	IocWriteGroup
IOCServiceId_IOC_Receive	IocReceive
IOCServiceId_IOC_Read	IocRead
IOCServiceId_IOC_ReceiveGroup	IocReceiveGroup

システムサービス ID 名	対応するシステムサービス名
IOCServiceId_IOC_ReadGroup	IocReadGroup
IOCServiceId_IOC_EmptyQueue	IocEmptyQueue
OSServiceId_StartCore	StartCore
OSServiceId_StartNonAutosarCore	StartNonAutosarCore
OSServiceId_GetSpinlock	GetSpinlock
OSServiceId_ReleaseSpinlock	ReleaseSpinlock
OSServiceId_TryToGetSpinlock	TryToGetSpinlock
OSServiceId_ShutdownAllCores	ShutdownAllCores
OSServiceId_GetMutex	GetMutex
OSServiceId_ReleaseMutex	ReleaseMutex
OSServiceId_TaskMissingEnd	タスク不正終了
OSServiceId_ISRMissingEnd	C2ISR 不正終了
OSServiceId_HookMissingEnd	フックルーチン不正終了

4.4.3　OS マクロ一覧

表 4-4 OS マクロ一覧

マクロ名	概要
OSMEMORY_IS_READABLE(AccessType)	指定されたメモリが読出し可能である場合，0 以外の値を返す.
OSMEMORY_IS_WRITEABLE(AccessType)	指定されたメモリが書込み可能である場合，0 以外の値を返す.
OSMEMORY_IS_EXECUTABLE(AccessType)	指定されたメモリが実行可能である場合，0 以外の値を返す.
OSMEMORY_IS_STACKSPACE(AccessType)	指定されたメモリがスタック領域である場合，0 以外の値を返す.
SVC_CALL(SystemServiceName)(Parameter)	Parameter で指定された引数で，SystemServiceName で指定されたシステムサービスを，明示的に関数呼出しにて発行する.
SVC_TRAP(SystemServiceName)(Parameter)	Parameter で指定された引数で，SystemServiceName で指定されたシステムサービスを，明示的にソフトウェア割込みにて発行する.

4.4.4　システムサービス ID 取得マクロ一覧

表 4-5 システムサービス ID 取得マクロ一覧

マクロ名	概要
OSErrorGetServiceId()	システムサービスエラー発生時のシステムサービス ID の取得

4.4.5　システムサービスパラメータ取得マクロ一覧

表 4-6 システムサービスパラメータ取得マクロ一覧

マクロ名	システムサービス名	取得引数名
OSError_StartOS_Mode()	StartOS	Mode
OSError_ShutdownOS_Error()	ShutdownOS	Error
OSError_ActivateTask_TaskID()	ActivateTask	TaskID
OSError_ChainTask_TaskID()	ChainTask	TaskID
OSError_GetTaskID_TaskID()	GetTaskID	TaskID
OSError_GetTaskState_TaskID()	GetTaskState	TaskID
OSError_GetTaskState_State()	GetTaskState	State
OSError_EnableInterruptSource_EnableISR()	EnableInterruptSource	EnableISR
OSError_DisableInterruptSource_DisableISR()	DisableInterruptSource	DisableISR
OSError_SetEvent_TaskID()	SetEvent	TaskID
OSError_SetEvent_Mask()	SetEvent	Mask
OSError_ClearEvent_Mask()	ClearEvent	Mask
OSError_GetEvent_TaskID()	GetEvent	TaskID
OSError_GetEvent_Event()	GetEvent	Event
OSError_WaitEvent_Mask()	WaitEvent	Mask
OSError_GetResource_ResID()	GetResource	ResID
OSError_ReleaseResource_ResID()	ReleaseResource	ResID
OSError_IncrementCounter_CounterID()	IncrementCounter	CounterID
OSError_GetCounterValue_CounterID()	GetCounterValue	CounterID
OSError_GetCounterValue_Value()	GetCounterValue	Value
OSError_GetElapsedValue_CounterID()	GetElapsedValue	CounterID
OSError_GetElapsedValue_Value()	GetElapsedValue	Value
OSError_GetElapsedValue_ElapsedValue()	GetElapsedValue	ElapsedValue
OSError_GetAlarmBase_AlarmID()	GetAlarmBase	AlarmID
OSError_GetAlarmBase_Info()	GetAlarmBase	Info
OSError_GetAlarm_AlarmID()	GetAlarm	AlarmID
OSError_GetAlarm_Tick()	GetAlarm	Tick
OSError_SetRelAlarm_AlarmID()	SetRelAlarm	AlarmID
OSError_SetRelAlarm_increment()	SetRelAlarm	increment
OSError_SetRelAlarm_cycle()	SetRelAlarm	cycle
OSError_SetAbsAlarm_AlarmID()	SetAbsAlarm	AlarmID
OSError_SetAbsAlarm_start()	SetAbsAlarm	start

マクロ名	システムサービス名	取得引数名
OSError_SetAbsAlarm_cycle()	SetAbsAlarm	cycle
OSError_CancelAlarm_AlarmID()	CancelAlarm	AlarmID
OSError_StartScheduleTableRel_ScheduleTableID()	StartScheduleTableRel	ScheduleTableID
OSError_StartScheduleTableRel_Offset()	StartScheduleTableRel	Offset
OSError_StartScheduleTableAbs_ScheduleTableID()	StartScheduleTableAbs	ScheduleTableID
OSError_StartScheduleTableAbs_Start()	StartScheduleTableAbs	Start
OSError_StopScheduleTable_ScheduleTableID()	StopScheduleTable	ScheduleTableID
OSError_NextScheduleTable_ScheduleTableID_From()	NextScheduleTable	ScheduleTableID_From
OSError_NextScheduleTable_ScheduleTableID_To()	NextScheduleTable	ScheduleTableID_To
OSError_StartScheduleTableSynchron_ScheduleTableID()	StartScheduleTableSynchron	ScheduleTableID
OSError_SyncScheduleTable_ScheduleTableID()	SyncScheduleTable	ScheduleTableID
OSError_SyncScheduleTable_Value()	SyncScheduleTable	Value
OSError_SetScheduleTableAsync_ScheduleTableID()	SetScheduleTableAsync	ScheduleTableID
OSError_GetScheduleTableStatus_ScheduleTableID()	GetScheduleTableStatus	ScheduleTableID
OSError_GetScheduleTableStatus_ScheduleStatus()	GetScheduleTableStatus	ScheduleStatus
OSError_CallTrustedFunction_FunctionIndex()	CallTrustedFunction	FunctionIndex
OSError_CallTrustedFunction_FunctonParams()	CallTrustedFunction	FunctonParams
OSError_CheckISRMemoryAccess_ISRID()	CheckISRMemoryAccess	ISRID
OSError_CheckISRMemoryAccess_Address()	CheckISRMemoryAccess	Address
OSError_CheckISRMemoryAccess_Size()	CheckISRMemoryAccess	Size
OSError_CheckTaskMemoryAccess_TaskID()	CheckTaskMemoryAccess	TaskID
OSError_CheckTaskMemoryAccess_Address()	CheckTaskMemoryAccess	Address
OSError_CheckTaskMemoryAccess_Size()	CheckTaskMemoryAccess	Size
OSError_CheckTaskAccess_ApplID()	CheckTaskAccess	ApplID
OSError_CheckTaskAccess_TaskID()	CheckTaskAccess	TaskID
OSError_CheckISRAccess_ApplID()	CheckISRAccess	ApplID
OSError_CheckISRAccess_ISRID()	CheckISRAccess	ISRID
OSError_CheckAlarmAccess_ApplID()	CheckAlarmAccess	ApplID
OSError_CheckAlarmAccess_AlarmID()	CheckAlarmAccess	AlarmID
OSError_CheckResourceAccess_ApplID()	CheckResourceAccess	ApplID
OSError_CheckResourceAccess_ResID()	CheckResourceAccess	ResID
OSError_CheckCounterAccess_ApplID()	CheckCounterAccess	ApplID
OSError_CheckCounterAccess_CounterID()	CheckCounterAccess	CounterID
OSError_CheckScheduleTableAccess_ApplID()	CheckScheduleTableAccess	ApplID

マクロ名	システムサービス名	取得引数名
OSError_CheckScheduleTableAccess_ScheduleTableID()	CheckScheduleTableAccess	ScheduleTableID
OSError_CheckSpinlockAccess_ApplID()	CheckSpinlockAccess	ApplID
OSError_CheckSpinlockAccess_SpinlockId()	CheckSpinlockAccess	SpinlockId
OSError_CheckTaskOwnership_TaskID()	CheckTaskOwnership	TaskID
OSError_CheckISROwnership_ISRID()	CheckISROwnership	ISRID
OSError_CheckAlarmOwnership_AlarmID()	CheckAlarmOwnership	AlarmID
OSError_CheckCounterOwnership_CounterID()	CheckCounterOwnership	CounterID
OSError_CheckScheduleTableOwnership_ScheduleTableID()	CheckScheduleTableOwnership	ScheduleTableID
OSError_TerminateApplication_Application()	TerminateApplication	Application
OSError_TerminateApplication_RestartOption()	TerminateApplication	RestartOption
OSError_GetApplicationState_Application()	GetApplicationState	Application
OSError_GetApplicationState_Value()	GetApplicationState	Value
OSError_IocSend_IocId()	IocSend，IocSendGroup	<IocId>
OSError_IocSend_SenderId()	IocSend	<SenderId>
OSError_IocWrite_IocId()	IocWrite，IocWriteGroup	<IocId>
OSError_IocWrite_SenderId()	IocWrite	<SenderId>
OSError_IocReceive_IocId()	IocReceive，IocReceiveGroup	<IocId>
OSError_IocRead_IocId()	IocRead，IocReadGroup	<IocId>
OSError_IocEmptyQueue_IocId()	IocEmptyQueue	<IocId>
OSError_StartCore_CoreID()	StartCore	CoreID
OSError_StartCore_Status()	StartCore	Status
OSError_StartNonAutosarCore_CoreID()	StartNonAutosarCore	CoreID
OSError_StartNonAutosarCore_Status()	StartNonAutosarCore	Status
OSError_GetSpinlock_SpinlockId()	GetSpinlock	SpinlockId
OSError_ReleaseSpinlock_SpinlockId()	ReleaseSpinlock	SpinlockId
OSError_TryToGetSpinlock_SpinlockId()	TryToGetSpinlock	SpinlockId
OSError_TryToGetSpinlock_Success()	TryToGetSpinlock	Success
OSError_ShutdownAllCores_Error()	ShutdownAllCores	Error
OSError_RaiseInterCoreInterrupt_ISRID()	RaiseInterCoreInterrupt	ISRID
OSError_GetMutex_MtxID()	GetMutex	MtxID
OSError_ReleaseMutex_MtxID()	ReleaseMutex	MtxID

4.5　エラーコード一覧

表 4-7 エラーコード一覧

エラーコード	値
E_OS_ACCESS	1
E_OS_CALLEVEL	2
E_OS_ID	3
E_OS_LIMIT	4
E_OS_NOFUNC	5
E_OS_RESOURCE	6
E_OS_STATE	7
E_OS_VALUE	8
E_OS_SERVICEID	実装定義
E_OS_ILLEGAL_ADDRESS	実装定義
E_OS_MISSINGEND	実装定義
E_OS_DISABLEDINT	実装定義
E_OS_STACKFAULT	実装定義
E_OS_PROTECTION_MEMORY	実装定義
E_OS_PROTECTION_TIME_TASK	実装定義
E_OS_PROTECTION_TIME_ISR	実装定義
E_OS_PROTECTION_ARRIVAL_TASK	実装定義
E_OS_PROTECTION_ARRIVAL_ISR	実装定義
E_OS_PROTECTION_LOCKED_RESOURCE	実装定義
E_OS_PROTECTION_LOCKED_OSINT	実装定義
E_OS_PROTECTION_LOCKED_ALLINT	実装定義
E_OS_PROTECTION_EXCEPTION	実装定義
E_OS_PROTECTION_FATAL	実装定義
E_OS_MODE	実装定義
E_OS_SHUTDOWN_FATAL	実装定義
E_OS_PARAM_POINTER	実装定義
E_OS_SYS_ASSERT_FATAL	実装定義
E_OS_STACKINSUFFICIENT	実装定義
E_OS_CORE	実装定義
E_OS_SPINLOCK	実装定義
E_OS_INTERFERENCE_DEADLOCK	実装定義
E_OS_NESTING_DEADLOCK	実装定義

E_OS_SHUTDOWN_OTHER_CORE	実装定義

4.6　IOC 用システムサービスエラーコード一覧

表 4-8 IOC 用システムサービスエラーコード一覧

エラーコード	値
IOC_E_OK	0
IOC_E_NOK	1
IOC_E_LIMIT	130
IOC_E_LOST_DATA	64
IOC_E_NO_DATA	131

4.7　対象外とした AUTOSAR 仕様

OS 以外のモジュールとの関連を示した規定等で, OS に対する要求仕様として取り扱う必要がない仕様に関しては，本仕様では対象外とした．具体的には以下の仕様である．

- ヘッダファイルによる公開パラメータ(＊)【OS766】
- 関連仕様情報【OS767】

※AUTOSAR 仕様 V5.1.0 (R4.1 Rev 1)では，(＊)の付く仕様は削除されている．

変更履歴

Version	Date	Detail	Editor
1.0.0	2010/3/29	NCES 内部リリース	NCES
1.1.0	2010/12/1	TOPPERS 会員向け早期リリース	NCES
1.2.0	2011/12/28	ATK2 コンソーシアム向けリリース	NCES
1.2.1	2012/3/30	・ファイル名から文書番号を削除 ・コピーライトを記載 ・未規定である仕様，曖昧な仕様の明確化 ・誤字，脱字，説明不足等を修正	NCES
2.0.0	2013/1/22	・未規定である仕様，曖昧な仕様の明確化 ・誤字，脱字，説明不足等を修正 ・AUTOSAR に記載されている仕様番号が無い 要求事項への付番(OSa*xxx*) ・一般向けリリース	NCES
3.0.0	2013/6/28	・IOC，マルチコアに対応 ・コア間割込み機能を追加 ・「コンフィギュレータ」を「ジェネレータ」へ呼称変更	NCES
3.0.1	2013/7/10	本仕様書に対応する ATK2 のバージョンを記載	NCES
3.1.0	2013/10/10	・仕様番号の凡例をプレフィックスに統一 ・1 つの文章に複数付与していた仕様番号を 1 文章につき 1 つの仕様番号となるように分離 ・保護違反時処理の機能レベル 2 の内容を拡張	NCES
3.1.1	2013/12/21	・一部のシステムサービスを割込み禁止状態からでも 呼び出せるように変更 ・誤字，脱字の修正	NCES
3.2.0	2014/3/12	・引用タグの記号を変更 ・設計書へのトレーサビリティ対応 ・IOC で使用するデータ型を IMPLEMENTATION-DATA-TYPE へ変更	NCES