User's Manual

RV850

リアルタイム・オペレーティング・システム
ユーザーズマニュアル 機能編

対象デバイス

RH850 ファミリ（RH850G3K）

RH850 ファミリ（RH850G3M）

RH850 ファミリ（RH850G3KH）

RH850 ファミリ（RH850G3MH）

本資料に記載の全ての情報は発行時点のものであり、ルネサス エレクトロニクスは、
予告なしに、本資料に記載した製品または仕様を変更することがあります。
ルネサス エレクトロニクスのホームページなどにより公開される最新情報をご確認ください。

ルネサス エレクトロニクス
www.renesas.com

Rev.1.20　2019.09

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品、本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

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

   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
6. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
10. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
12. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

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

(Rev.4.0-1　2017.11)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。

© 2019 Renesas Electronics Corporation. All rights reserved.

このマニュアルの使い方

対象者

このマニュアルは，RH850 ファミリの各製品の応用システムを設計，開発するユーザを対象としています。

目的

このマニュアルは，次の構成に示すルネサス エレクトロニクス製リアルタイム OS RV850 の機能をユーザに理解していただくことを目的としています。

構成

このマニュアルは，大きく分けて次の内容で構成しています。

1. 概　　説
2. システム構築
3. タスク管理
4. 割り込み管理
5. リソース管理
6. イベント管理
7. カウンタ管理
8. アラーム管理
9. スケジュール・テーブル管理
10. OS アプリケーション管理
11. OS 実行管理
12. スケジュール管理
13. システム初期化処理
14. システム・サービス
A. コンフィギュレータ
B. CF ファイル（OIL）
C. メモリ使用量

読み方

このマニュアルを読むにあたっては，電気，論理回路，マイクロコンピュータ，C 言語，アセンブラの一般知識が必要となります。

RH850 ファミリのハードウエア機能を知りたいとき
→　各製品のユーザーズ・マニュアルを参照してください。

凡例

データ表記の重み	：	左が上位桁，右が下位桁
注	：	本文中についた注の説明
注意	：	気をつけて読んでいただきたい内容
備考	：	本文中の補足説明
数の表記	：	10 進数 ... XXXX 16 進数 ... 0xXXXX
2 のべき数を示す接頭語（アドレス空間，メモリ容量）：		
		K（キロ）　$2^{10} = 1024$
		M（メガ）　$2^{20} = 1024^2$

関連資料

関連資料は暫定版の場合がありますが，この資料では「暫定」の表示をしておりません。あらかじめご了承ください。

資料名		資料番号	
		和文	英文
RV850	機能編	このマニュアル	R20UT2768E

注意　上記資料は，予告なしに内容を変更することがあります。設計などには，必ず最新の資料を使用してください。

この資料に記載されている会社名，製品名などは，各社の商標または登録商標です。

目次

1. 概　説 9
1.1 概　要 9
1.2 特　長 9
1.3 構　成 9
1.4 実行環境 10
1.5 フォルダ構成 11
1.5.1 オブジェクト・ファイル提供形式 11
1.5.2 ソース・ファイル提供形式 13
2. システム構築 15
2.1 概　要 15
2.2 ユーザ・オウン・コーディング部の記述 15
2.2.1 ユーザ・オウン・ライブラリの生成 16
2.3 処理プログラムの記述 16
2.4 CF ファイルの記述 18
2.4.1 情報ファイルの生成 18
2.5 リンカ・ディレクティブ・ファイルの記述 18
2.6 ロード・モジュールの生成 19
3. タスク管理 20
3.1 概　要 20
3.1.1 状　態 20
3.1.2 スタック・モニタリング機能 21
3.1.3 タ ス ク 21
3.1.4 タスクでの処理 21
3.1.5 タスクの生成 22
3.1.6 タスクの終了 22
3.2 システム・サービス 23
4. 割り込み管理 24
4.1 概　要 24
4.1.1 スタック・モニタリング機能 24
4.2 ブート処理 24
4.2.1 ブート処理での処理 25
4.3 割り込みサービス・ルーチン 26
4.3.1 割り込みサービス・ルーチンでの処理 27
4.3.2 割り込みサービス・ルーチンの登録 28
4.3.3 割り込みサービス・ルーチンの終了 28
4.4 システム・サービス 30

4.5 ユーザ・オウン・コーディング部 30
4.5.1 エントリ処理（直接分岐方式の例外ベクタ） 30
4.5.2 例外／割り込み安全対策処理 32
4.6 多重割り込み 33
5. リソース管理 34
5.1 概 要 34
5.1.1 シーリング値 34
5.1.2 スケジューラ・リソース 34
5.2 リソースの生成 35
5.3 システム・サービス 35
6. イベント管理 36
6.1 概 要 36
6.2 イベントの生成 36
6.3 システム・サービス 36
7. カウンタ管理 37
7.1 概 要 37
7.1.1 システム・カウンタ 37
7.2 カウンタの生成 37
7.3 システム・サービス 37
8. アラーム管理 38
8.1 概 要 38
8.2 アラーム・コールバック 38
8.2.1 アラーム・コールバックでの処理 38
8.2.2 アラーム・コールバックの登録 39
8.3 アラームの生成 39
8.4 システム・サービス 39
9. スケジュール・テーブル管理 40
9.1 概 要 40
9.2 スケジュール・テーブル 40
9.2.1 状 態 40
9.2.2 スケジュール・テーブルの生成 40
9.3 システム・サービス 40
10. OS アプリケーション管理 41
10.1 概 要 41
10.1.1 信 頼 性 41
10.1.2 状 態 41
10.1.3 メモリ・アクセス保護機能 42
10.1.4 周辺 I/O 保護機能 42

10.2 Trusted 関数 . . . 44
10.2.1 Trusted 関数の処理 . . . 44
10.2.2 Trusted 関数の登録 . . . 45
10.2.3 Trusted 関数の引き継ぎデータ . . . 45
10.3 固有フック・ルーチン . . . 47
10.3.1 固有フック・ルーチンでの処理 . . . 48
10.3.2 固有フック・ルーチンの登録 . . . 49
10.4 OS アプリケーションの生成 . . . 49
10.5 システム・サービス . . . 49

11. OS 実行管理 . . . 50
11.1 概　要 . . . 50
11.2 共通フック・ルーチン . . . 50
11.2.1 共通フック・ルーチンでの処理 . . . 53
11.2.2 共通フック・ルーチンの登録 . . . 54
11.2.3 システム・サービス . . . 54

12. スケジュール管理 . . . 55
12.1 概　要 . . . 55
12.2 フック・ルーチン . . . 55
12.3 アイドル・ハンドラ . . . 56
12.3.1 アイドル・ハンドラでの処理 . . . 56

13. システム初期化処理 . . . 57
13.1 概　要 . . . 57
13.2 エントリ処理（直接分岐方式の例外ベクタ） . . . 58
13.3 ブート処理 . . . 58
13.4 カーネル初期化部 . . . 58
13.5 フック・ルーチン . . . 58

14. システム・サービス . . . 59
14.1 概　要 . . . 59
14.1.1 システム・サービスの呼び出し . . . 60
14.2 データ・マクロ . . . 61
14.2.1 データ・タイプ . . . 61
14.2.2 エラー・ステータス . . . 63
14.2.3 不正なタスク識別子 . . . 63
14.2.4 タスクの状態 . . . 64
14.2.5 スケジュール・テーブルの状態 . . . 64
14.2.6 異常終了 . . . 64
14.2.7 アクセス権の種類 . . . 65
14.2.8 オブジェクトの種類 . . . 65
14.2.9 アクセス権の有無 . . . 65

14.2.10 再起動オプション . . . 66
14.2.11 OS アプリケーションの状態 . . . 66
14.2.12 システム・サービス識別子 . . . 67
14.2.13 カウンタ情報 . . . 69
14.2.14 アクセス権の有無 . . . 70
14.2.15 アイドル・ハンドラの動作モード . . . 70
14.2.16 コアの識別子 . . . 70
14.3 データ構造体 . . . 71
14.3.1 アラーム・ベース情報 . . . 71
14.4 システム・サービス・リファレンス . . . 72
14.4.1 タスク管理 . . . 75
14.4.2 割り込み管理 . . . 85
14.4.3 リソース管理 . . . 95
14.4.4 イベント管理 . . . 100
14.4.5 カウンタ管理 . . . 108
14.4.6 アラーム管理 . . . 114
14.4.7 スケジュール・テーブル管理 . . . 125
14.4.8 OS アプリケーション管理 . . . 137
14.4.9 OS 実行管理 . . . 158
14.4.10 ユーティリティ関数 . . . 165

A. コンフィギュレータ . . . 175
A.1 概 要 . . . 175
A.2 起動方法 . . . 175
A.2.1 コマンド・ファイル . . . 177
A.3 コマンド入力例 . . . 178
A.4 メッセージ . . . 179
A.4.1 フェイタル・エラー . . . 179
A.4.2 アボート・エラー . . . 183
A.4.3 ワーニング . . . 185

B. CF ファイル（OIL） . . . 186
B.1 概 要 . . . 186
B.2 コンフィギュレーション情報 . . . 187
B.3 インクルード・ファイル . . . 188
B.4 CPU セクション . . . 189
B.4.1 アラーム情報 . . . 190
B.4.2 アプリケーション・モード情報 . . . 194
B.4.3 OS アプリケーション情報 . . . 195
B.4.4 カウンタ情報 . . . 202
B.4.5 イベント情報 . . . 205
B.4.6 割り込みサービス・ルーチン情報 . . . 207
B.4.7 OS 情報 . . . 209

B.4.8 リソース情報 214
B.4.9 スケジュール・テーブル情報 217
B.4.10 タスク情報 222
B.4.11 システム情報 226

C. メモリ使用量 229
C.1 概 要 229
C.2 標準処理領域（.kernel_system） 229
C.3 インタフェース領域（.kernel_interface） 230
C.4 定数データ領域（.kernel_const） 231
C.5 定数データ領域（.kernel_identifier） 235
C.6 変数データ領域（.kernel_work） 237
C.6.1 優先度バッファ 239
C.7 スタック領域（.kernel_stack） 240
C.7.1 システム・スタック 241
C.7.2 OS アプリケーション・スタック 244
C.7.3 タスク・スタック（拡張タスク） 246
C.8 割り込みハンドラ・アドレス・テーブル（.kernel_address） 248

改訂記録 249

1. 概　　説

1.1 概　　要

RV850 は，効率の良いリアルタイム処理環境，およびマルチタスク処理環境を提供すると伴に，対象デバイスの制御機器分野における応用範囲を拡大することを目的として開発されたリアルタイム・マルチタスク OS です。

1.2 特　　長

以下に，RV850 の特徴を示します。

(1) OSEK/VDX 仕様，AUTOSAR 仕様に準拠
車載分散制御用のオープン・エンド・アーキテクチャの業界標準規格を目的とした OSEK/VDX 仕様（OSEK/VDX Operating System Ver.2.2.3，OSEK/VDX OSEK Implementation Language Ver.2.5），および AUTOSAR 仕様（AUTOSAR Specification of Operating System R4.2 Rev2）に準拠した設計が行われており，AUTOSAR 仕様における以下の機能を提供しています。

- スケーラビリティ・クラス SC1 でサポートする機能
- スケーラビリティ・クラス SC3 でサポートする機能

RV850 では，スケーラビリティ・クラス SC2 および SC4，マルチコア機能，IOC 機能はサポートしておりません。

(2) 高い移植性
様々な実行環境に対応するために，RV850 が処理を実行するうえで必要となるハードウエア依存処理を RV850 依存部（ユーザ・オウン・コーディング部）として切り出し，サンプル・ソース・ファイルを提供しています。これにより，様々な実行環境への移植性を向上させると伴に，カスタマイズ化を容易にしています。

(3) ROM 化の実現
実行環境に組み込んで使用することを前提としているため，ROM 化を意識し，コンパクトな設計が行われています。
また，RV850 では，スケーラビリティ・クラスが SC1 の際には，ユーザの記述したアプリケーション・システムで使用されるシステム・サービスのみがシステム構築時にリンクされるため，コンパクトでありながら，ユーザのニーズに最適なリアルタイム・マルチタスク環境を構築することが可能となっています。

(4) ユーティリティ・ツールの提供
システム構築時に有益なユーティリティ・ツールを提供しています。

- コンフィギュレータ
  CF ファイルを入力ファイルとして読み込むことにより，RV850 に提供するコンフィギュレーション情報を保持した情報ファイル（SIT ファイル，ENTRY ファイル，カーネル・マクロ・ファイル）を出力します。

1.3 構　　成

RV850 は，以下に示した 2 種類のモジュールから構成されています。

(1) RV850 共通部
RV850 の核となる処理部分であり，以下に示す機能を提供しています。

- オブジェクト用の領域確保処理
- オブジェクトの初期化処理
- システム・サービスに対応した処理
- 事象の変化に対応した処理プログラムの選択処理

(2) RV850 依存部
RV850 が処理を実行するうえで必要となるハードウエア依存処理です。
なお，本処理部は，処理プログラムの実行環境に依存しているため，ユーザ・オウン・コーディング部として切り出され，サンプル・ソース・ファイルを提供しています。

1.4 実行環境

RV850 が処理を実行するうえで必要となる実行環境を示します。

(1) デバイス
以下に，RV850 の対象デバイスを示します。

- RH850 ファミリ（G3K コア，G3M コア，G3KH コア，G3MH コア）

備考 1. RV850 では，デバイスが提供している各種レジスタ（PSW，PMR）を利用し，割り込み管理を実現しています。

備考 2. スケーラビリティ・クラス SC3 に対応した RV850 は，デバイスが提供しているメモリ保護機能（MPU）を利用し，不正なメモリ・アクセスの抑制を実現しています。

備考 3. RV850 では，デバイスが提供している SIMD 演算機能に対応していません。

(2) 周辺コントローラ
RV850 では，様々な実行環境に対応する目的から，実行環境に依存した処理については，ユーザ・オウン・コーディング部として切り出すとともに，サンプル・ソース・ファイルを提供し，移植性の向上を図っています。

備考 1. RV850 では，割り込みハンドラ・アドレスの選択方式として，テーブル参照方式を採用しています。
したがって，RV850 は，テーブル参照方式が非搭載のハードウエアでは動作しません。

備考 2. RV850 では，割り込みコントローラ INTC1，および INTC2 を利用し，割り込み管理を実現しています。

備考 3. RV850 では，AUTOSAR 仕様のスケーラビリティ・クラス SC3（G3M，G3KH，G3MH 限定）で規定された周辺保護を実現する際，マシン・コンフィギュレーション（MCFG0）の SPID ビット（システム・プロテクション番号）と連動した安全機能を利用しています。
したがって，RV850 は，安全機能が非搭載のハードウエアでは，周辺保護を行いません。

以下に，RV850 が占有し，処理プログラムからの操作が禁止されている OS 予約資源を示します。

表 1.1　RV850 が占有している OS 予約資源

資源名	スケーラビリティ・クラス
汎用レジスタ（r2)	SC1 ／ SC3
マシン・コンフィギュレーション（MCFG0）の SPID ビット	SC3（G3M，G3KH，G3MH 限定）
割り込みハンドラ・アドレス・テーブルのベース・アドレス（INTBP）	SC1 ／ SC3
SYSCALL の動作設定（SCCFG）	SC3
SYSCALL ベース・ポインタ（SCBP）	SC3
プログラム・ステータス・ワード（PSW）の UM ビット	SC1 ／ SC3
割り込み優先度マスク（PMR）	SC1 ／ SC3
割り込み機能の設定（INTCFG）	SC1 ／ SC3
例外コード “OsIsrExceptionCode” ／例外コード “OsCounterExceptionCode” で定義された割り込みに対応した割り込みコントローラ（INTC1，INTC2）の EI レベル割り込み制御レジスタ（EIC*n*）の EITB*n* ビット，および EIP3*n*-0*n* ビット	SC1 ／ SC3
メモリ保護機能（MPU）の領域 0 ～領域 3	SC3
メモリ保護機能（MPU）の領域 4 ～領域 11	SC3（G3M，G3KH，G3MH 限定）

備考　プログラム・ステータス・ワード（PSW）の ID ビットは，RV850 は占有しませんが，RV850 では，処理プログラムから PSW の ID ビットを操作する際の注意事項を規定しています。詳細は，「2.3 処理プログラムの記述」を参照してください。

1.5 フォルダ構成

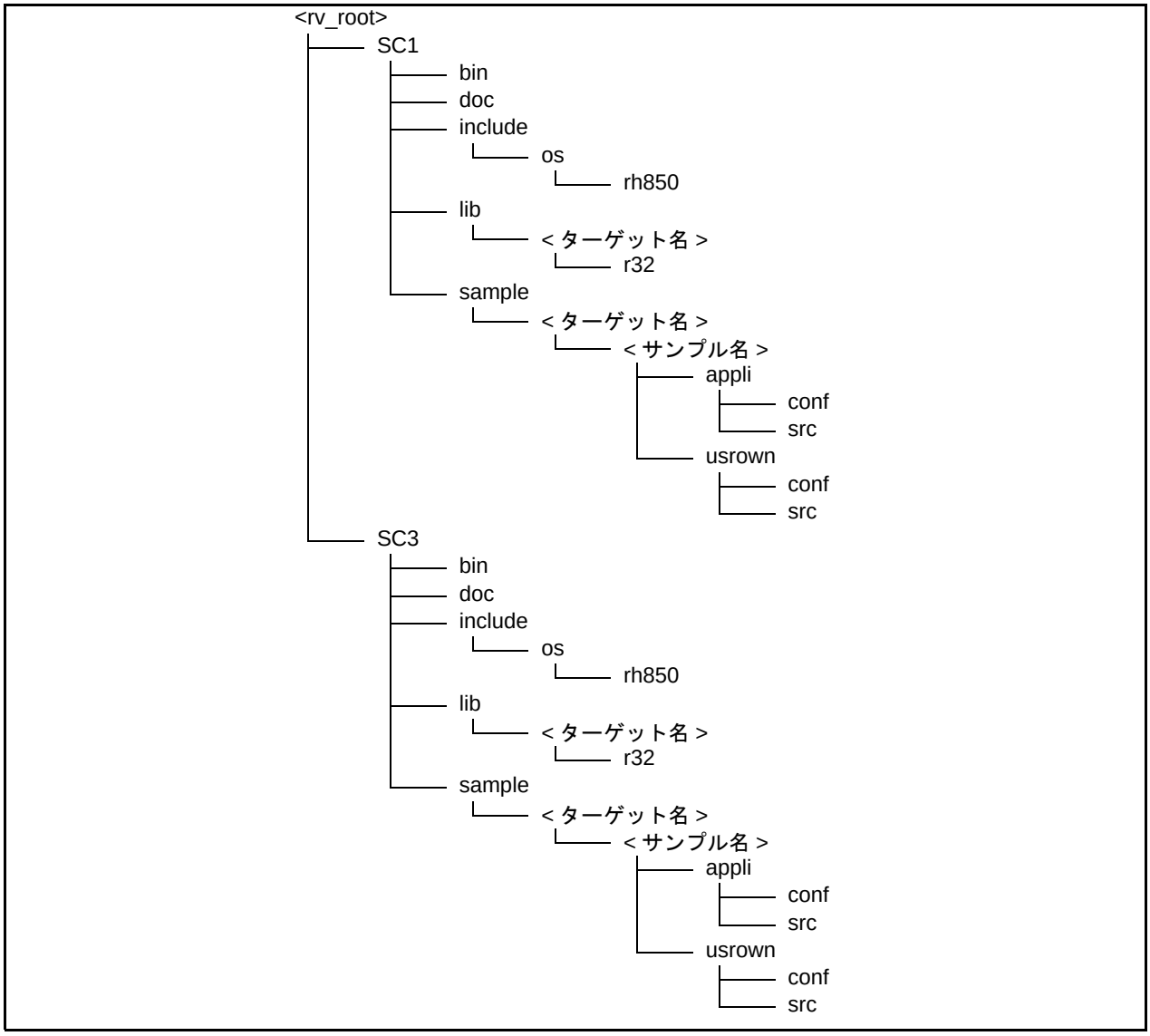
ホスト・マシン上に展開されたファイル群のフォルダ構成は，提供形式の種類により異なります。

- オブジェクト・ファイル提供形式
- ソース・ファイル提供形式

1.5.1 オブジェクト・ファイル提供形式

以下に，RV850 が提供しているファイル群（オブジェクト・ファイル提供形式）をホスト・マシン上に展開した際に生成されるフォルダ構成を示します。

図 1.1　　フォルダ構成（オブジェクト・ファイル提供形式）



以下に，各フォルダの詳細を示します。

(1)　<rv_root>
パッケージの展開時に指定した「RV850 の展開先」です。

(2)　<rv_root>￥SC1
RV850 のファイル（スケーラビリティ・クラス SC1 用）が格納されているフォルダです。

(3)　<rv_root>￥SC3
RV850 のファイル（スケーラビリティ・クラス SC3 用）が格納されているフォルダです。

(4) <rv_root>￥{SC1，SC3}￥bin
RV850 のユーティリティ・ツールが格納されているフォルダです。
Os_Configurator.exe： コンフィギュレータ
AUTOSAR_RENESAS_OS_ECUConfigurationParameters.arxml：
OS モジュール定義ファイル（コンフィギュレーション・エディタ用）

(5) <rv_root>￥{SC1，SC3}￥doc
RV850 のドキュメント・ファイルが格納されているフォルダです。

(6) <rv_root>￥{SC1，SC3}￥include
RV850 の標準ヘッダ・ファイルが格納されているフォルダです。
Os.h： 標準ヘッダ・ファイル
MemMap.h： AUTOSAR 標準ヘッダ・ファイル

(7) <rv_root>￥{SC1，SC3}￥include￥os
RV850 のヘッダ・ファイルが格納されているフォルダです。

(8) <rv_root>￥{SC1，SC3}￥include￥os￥rh850
RV850 のヘッダ・ファイル（デバイス依存：RH850 ファミリ）が格納されているフォルダです。

(9) <rv_root>￥{SC1，SC3}￥lib￥<ターゲット名>￥r32
RV850 のカーネル・ライブラリ（32 レジスタ・モード）が格納されているフォルダです。
libecc2extsc1.a： ECC2，拡張ステータス，SC1，FPU 対応なし
libecc2extsc1_fpu.a： ECC2，拡張ステータス，SC1，FPU 対応あり
libecc2extsc3.a： ECC2，拡張ステータス，SC3，FPU 対応なし，G3M，G3KH，G3MH 限定
libecc2extsc3_fpu.a： ECC2，拡張ステータス，SC3，FPU 対応あり，G3M，G3KH，G3MH 限定
libecc2extsc3_g3k.a： ECC2，拡張ステータス，SC3，FPU 対応なし，G3K 限定

(10) <rv_root>￥{SC1，SC3}￥sample￥<ターゲット名>￥<サンプル名>￥appli￥conf
サンプル・ロード・モジュールを生成するためのファイルが格納されているフォルダです。

(11) <rv_root>￥{SC1，SC3}￥sample￥<ターゲット名>￥<サンプル名>￥appli￥src
サンプル・プログラムのソース・ファイル，およびヘッダ・ファイルが格納されているフォルダです。

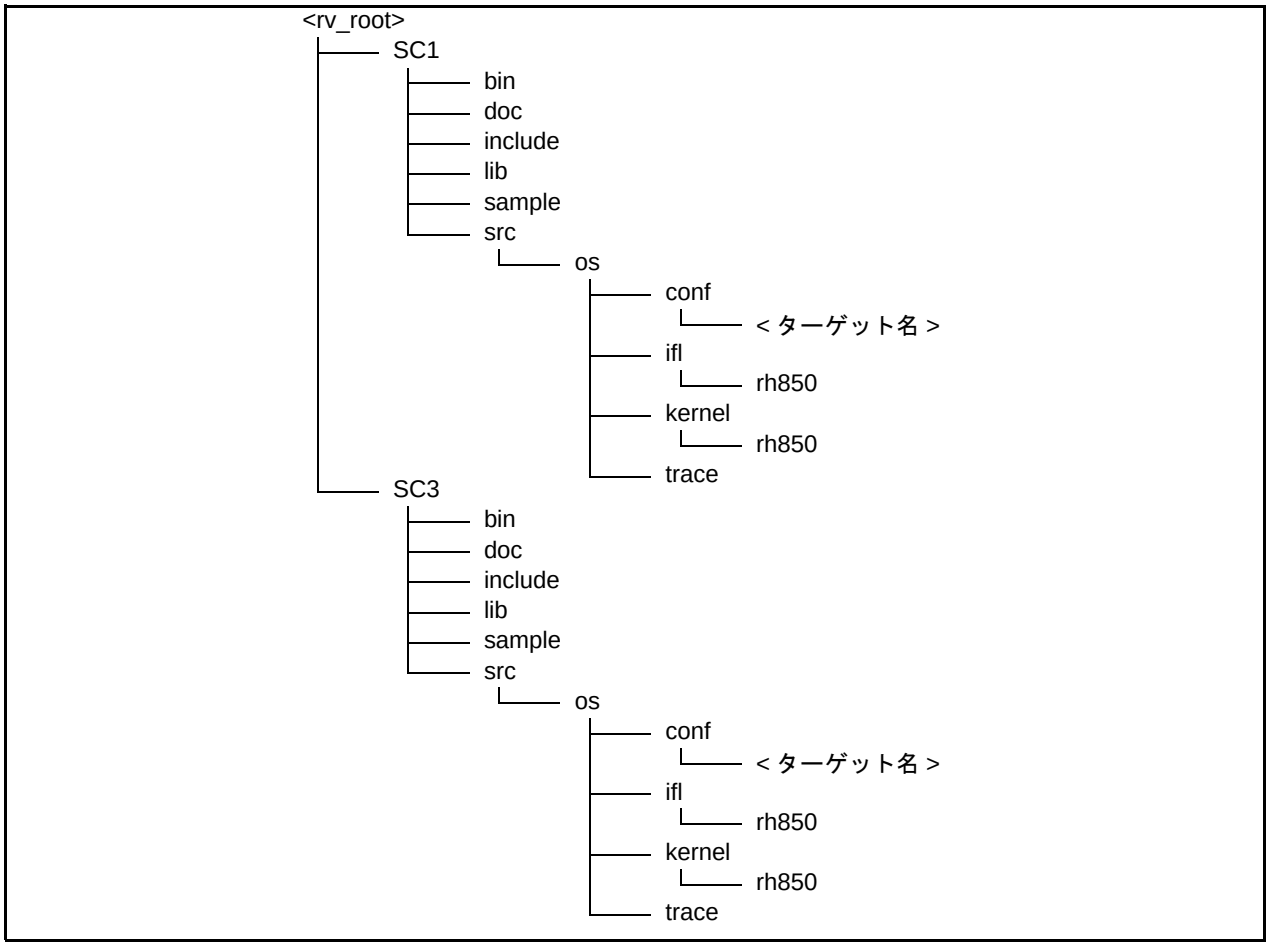
(12) <rv_root>￥{SC1，SC3}￥sample￥<ターゲット名>￥<サンプル名>￥usrown￥conf
ユーザ・オウン・ライブラリ（32 レジスタ・モード），およびユーザ・オウン・ライブラリ（32 レジスタ・モード）を生成するためのファイルが格納されているフォルダです。
libusr.a： ユーザ・オウン・ライブラリ

(13) <rv_root>￥{SC1，SC3}￥sample￥<ターゲット名>￥<サンプル名>￥usrown￥src
ユーザ・オウン・コーディング部のソース・ファイルが格納されているフォルダです。

1.5.2 ソース・ファイル提供形式

以下に，RV850 が提供しているファイル群（ソース・ファイル提供形式）をホスト・マシン上に展開した際に生成されるフォルダ構成を示します。

図 1.2　フォルダ構成（ソース・ファイル提供形式）



以下に，各フォルダの詳細を示します。

(1) <rv_root>
パッケージの展開時に指定した「RV850 の展開先」です。

(2) <rv_root>￥SC1
RV850 のファイル（スケーラビリティ・クラス SC1 用）が格納されているフォルダです。

(3) <rv_root>￥SC3
RV850 のファイル（スケーラビリティ・クラス SC3 用）が格納されているフォルダです。

(4) <rv_root>￥{SC1，SC3}￥{bin，doc，include，lib，sample}
「1.5.1 オブジェクト・ファイル提供形式」と同様のフォルダ構成，ファイル配置です。

(5) <rv_root>￥{SC1，SC3}￥src￥os￥conf￥< ターゲット名 >
カーネル・ライブラリ（32 レジスタ・モード）を生成するためのファイルが格納されているフォルダです。本フォルダに配置されているファイルを利用して生成されたカーネル・ライブラリ（32 レジスタ・モード）は，<rv_root>￥{SC1，SC3}￥lib￥< ターゲット名 >￥r32 フォルダに格納されます。

(6) <rv_root>￥{SC1，SC3}￥src￥os￥ifl
RV850 のインタフェース・ファイルが格納されているフォルダです。

(7) <rv_root>￥{SC1，SC3}￥src￥os￥ifl￥rh850
RV850 のインタフェース・ファイル（デバイス依存：RH850 ファミリ）が格納されているフォルダです。

(8) <rv_root>￥{SC1，SC3}￥src￥os￥kernel
カーネル・ライブラリ（32 レジスタ・モード）のソース・ファイルが格納されているフォルダです。

(9) <rv_root>￥{SC1，SC3}￥src￥os￥kernel￥rh850
カーネル・ライブラリ（32 レジスタ・モード）のソース・ファイル（デバイス依存：RH850 ファミリ）が格納されているフォルダです。

(10) <rv_root>￥{SC1，SC3}￥src￥os￥trace
トレース・ルーチンのソース・ファイルが格納されているフォルダです。

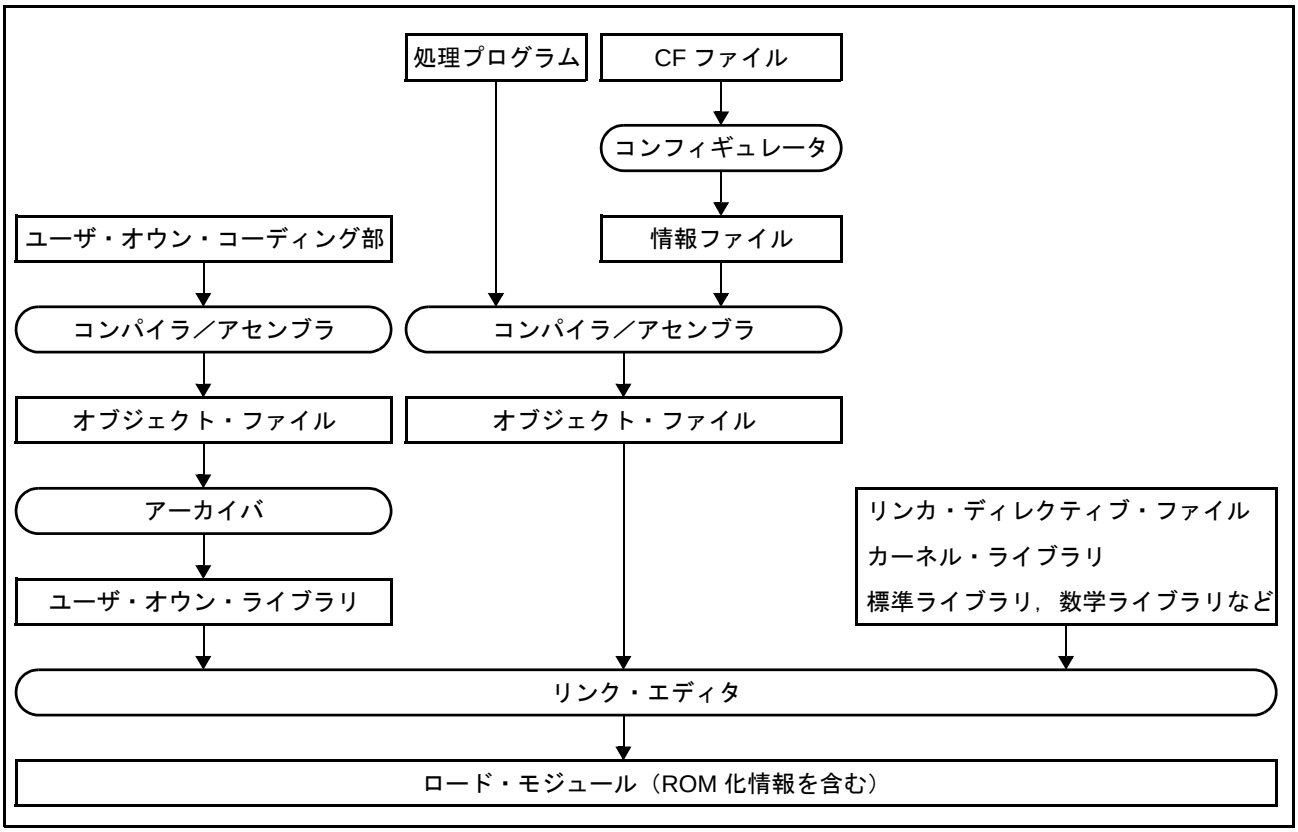
2. システム構築

本章では，RV850 が提供している機能を利用したシステム（ロード・モジュール）の構築手順について解説しています。

2.1 概　　要

以下に，システム構築の手順を示します。

図 2.1　　システム構築の手順



備考　　RV850 では，_kernel，および _KERNEL で始まるシンボル名を OS 予約シンボルとしているため，指定された用途以外での使用は禁止されています。

2.2 ユーザ・オウン・コーディング部の記述

RV850 では，様々な実行環境に対応するために，RV850 が処理を実行するうえで必要となるハードウエア依存処理を RV850 依存部（ユーザ・オウン・コーディング部）として切り出し，サンプル・ソース・ファイル（direct_vector.850，excent.850）を提供しています。
これにより，様々な実行環境への移植性を向上させると伴に，カスタマイズ化を容易にしています。
以下に，機能別に切り出されているユーザ・オウン・コーディング部の一覧を示します。

(1)　エントリ処理（直接分岐方式の例外ベクタ）
リセット（RESET），FE レベル割り込み（FENMI，TRAP など），EI レベル割り込み（例外コード “OsIsrExceptionCode” ／例外コード “OsCounterExceptionCode” で未定義）などが発生した際，対応する処理（ブート処理，例外／割り込み安全対策処理など）への分岐処理を割り付けるために切り出されたエントリ処理専用ルーチンです。

(2)　例外／割り込み安全対策処理
FE レベル割り込み（FENMI，TRAP など），EI レベル割り込み（例外コード “OsIsrExceptionCode” ／例外コード “OsCounterExceptionCode” で未定義）などが発生した際，エントリ処理から呼び出される安全対策処理専用ルーチンです。

備考　　ユーザ・オウン・コーディング部についての詳細は，「4.5.1 エントリ処理（直接分岐方式の例外ベクタ）」，「4.5.2 例外／割り込み安全対策処理」を参照してください。

2.2.1 ユーザ・オウン・ライブラリの生成

「2.2 ユーザ・オウン・コーディング部の記述」で作成されたソース・ファイル（例外／割り込み安全対策処理：excent.850）に対してコンパイラ／アセンブラ／アーカイバを実行し，ライブラリ・ファイル（ユーザ・オウン・ライブラリ）を生成します。

備考 コンパイラ／アセンブラ／アーカイバについての詳細は，使用するコンパイラ・パッケージのユーザーズ・マニュアルを参照してください。

2.3 処理プログラムの記述

システムとして実現すべき処理を記述します。
なお，RV850 では，処理プログラムを実現すべき処理の種類に合わせて，以下のように大別しています。

(1) ブート処理
RV850 が処理を実行するうえで必要となる最低限のハードウエアを初期化するために切り出された初期処理専用ルーチンです。

(2) タスク
システム・サービスを発行して状態操作を行わない限り，または CF ファイルで定義した条件が成立しない限り実行されることのない処理ルーチンです。

(3) 割り込みサービス・ルーチン
EI レベル割り込み（例外コード “OsIsrExceptionCode” ／例外コード “OsCounterExceptionCode” で定義）が発生した際に呼び出される割り込み処理専用ルーチンです。
なお，RV850 では，割り込みの発生から割り込みサービス・ルーチンを呼び出すまでの応答性を考慮し，2 種類のカテゴリをサポートしています。

- カテゴリ 1
- カテゴリ 2

(4) アラーム・コールバック（SC1 限定）
アラームが満了した際に呼び出される満了処理専用ルーチンです。

(5) Trusted 関数（SC3 限定）
信頼性が確保されている OS アプリケーションについては，OS アプリケーション単位に固有な Trusted 関数を所属させることができます。

(6) 共通フック・ルーチン
RV850 では，共通なフック・ルーチンとして，用途別に 6 種類のフック・ルーチンをサポートしています。

- StartupHook
- ShutdownHook
- PostTaskHook
- PreTaskHook
- ErrorHook
- ProtectionHook（SC3 限定）

(7) 固有フック・ルーチン（SC3 限定）
OS アプリケーション単位に固有なフック・ルーチンとして，用途別に 3 種類のフック・ルーチンをサポートしています。

- StartupHook_*OsApplication*
- ShutdownHook_*OsApplication*
- ErrorHook_*OsApplication*

(8) アイドル・ハンドラ
デバイスが提供している低消費電力対応機能を有効活用するために切り出されたアイドル処理専用ルーチンです。

備考 1. 処理プログラムについての詳細は，「4.2 ブート処理」，「3.1.3 タスク」，「4.3 割り込みサービス・ルーチン」，「8.2 アラーム・コールバック」，「10.2 Trusted 関数」，「11.2 共通フック・ルーチン」，「10.3 固有フック・ルーチン」，「12.3 アイドル・ハンドラ」を参照してください。

備考 2. 処理プログラムで，システム・サービスを発行せずに直接プログラム・ステータス・ワード（PSW）の ID ビットを操作する場合は，処理プログラムが切り替わる前に，操作前の状態に戻すようにしてください。ただし，処理プログラムに対し，PSW の ID ビットに関する注意事項が存在する場合は，注意事項に従った上で可能な場合のみ，PSW の ID ビットを操作するようにしてください。

2.4 CF ファイルの記述

RV850 が動作するうえで必要なデータを保持した情報ファイル（SIT ファイル，ENTRY ファイル，カーネル・マクロ・ファイル）を生成する際に必要となる CF ファイルを記述します。

備考　CF ファイルについての詳細は，「B. CF ファイル（OIL）」を参照してください。

2.4.1 情報ファイルの生成

「2.4 CF ファイルの記述」で作成された CF ファイルに対してコンフィギュレータを実行し，情報ファイル（SIT ファイル，ENTRY ファイル，カーネル・マクロ・ファイル）を生成します。

備考　コンフィギュレータについての詳細は，「A. コンフィギュレータ」を参照してください。

2.5 リンカ・ディレクティブ・ファイルの記述

リンク・エディタが行うメモリ配置を固定化するためのリンカ・ディレクティブ・ファイルを記述します。
なお，RV850 では，機能単位にモジュール化されたオブジェクトの割り付け先（セクション名，属性，配置先）を規定しています。したがって，ユーザが記述したプログラムの他に，規定された割り付け先をリンカ・ディレクティブ・ファイルに定義することは必須となります。
以下に，RV850 が規定している割り付け先の一覧を示します。

表 2.1　RV850 によって規定されるオブジェクト割り付け先

セクション名	属性	配置先	説明
.kernel_system	AX	ROM	標準処理領域
.kernel_interface	AX	ROM	インタフェース領域
.kernel_const	A	ROM	定数データ領域（ROM）
.kernel_identifier	A	ROM	定数データ領域
.kernel_work	ABW	RAM	変数データ領域（RAM）
.kernel_stack	ABW	RAM	スタック領域
.kernel_*address*	A	ROM	割り込みハンドラ・アドレス・テーブル

備考 1.　“ 属性 ” 列のキーワードは，以下の意味を持ちます。
A：　アクセス可
B：　初期値なし
W：　書き込み可
X：　実行可

備考 2.　RV850 が規定しているセクションのうち，.kernel_interface，および .kernel_identifier については，すべての OS アプリケーションからアクセス可能な領域に配置する必要があります。

備考 3.　RV850 が規定しているセクションのうち，.kernel_system については，対象デバイスが G3MH の場合は，0x00000100 未満のアドレスに配置しないでください。

備考 4.　割り込みハンドラ・アドレス・テーブル .kernel_*addresss* の割り付け先については，コンフィギュレータが ENTRY ファイルに出力しているため，ユーザがリンカ・ディレクティブ・ファイルに .kernel_*address* に関する定義を記述する必要はありません。
なお，.kernel_*address* における “*address*” は，「ベース・アドレス “OsInterruptBaseAddress” + 4 * 割り込みチャネル番号」で算出された値となります。

備考 5.　リンカ・ディレクティブ・ファイルについての詳細は，使用するコンパイラ・パッケージのユーザーズ・マニュアルを参照してください。

2.6 ロード・モジュールの生成

「2.2 ユーザ・オウン・コーディング部の記述」から「2.5 リンカ・ディレクティブ・ファイルの記述」で作成されたファイル群，および RV850 ／コンパイラ・パッケージが提供しているライブラリ・ファイルに対してコンパイラ／アセンブラ／リンク・エディタを実行し，ロード・モジュール（ROM 化情報を含む）を生成します。
以下に，ロード・モジュール（ROM 化情報を含む）を生成する際に必要となるファイル群の一覧を示します。

(1) 「2.2 ユーザ・オウン・コーディング部の記述」で作成されたライブラリ・ファイル

- ユーザ・オウン・ライブラリ

(2) 「2.3 処理プログラムの記述」で作成されたソース・ファイル

- 処理プログラム（ブート処理，タスク，割り込みサービス・ルーチン，アラーム・コールバック，Trusted 関数，共通フック・ルーチン，固有フック・ルーチン，アイドル・ハンドラ）

(3) 「2.4 CF ファイルの記述」で作成された情報ファイル

- 情報ファイル（SIT ファイル，ENTRY ファイル，カーネル・マクロ・ファイル）

(4) 「2.5 リンカ・ディレクティブ・ファイルの記述」で作成されたリンカ・ディレクティブ・ファイル

- リンカ・ディレクティブ・ファイル

(5) RV850 が提供しているライブラリ・ファイル

- カーネル・ライブラリ

(6) コンパイラ・パッケージが提供しているライブラリ・ファイル

- 標準ライブラリ，数学ライブラリなど

備考 1. RV850 では，プログラム・レジスタ（r2）を占有しています。このため，処理プログラムをコンパイル／アセンブルする際には，-reserve_r2 オプションを指定する必要があります。

備考 2. RV850 では，微小データ・エリア（TDA）を無効としています。このため，処理プログラムをコンパイル／アセンブルする際には，-notda オプションを指定する必要があります。

備考 3. RV850 では，callt 命令使用時の動作を保証していません。このため，処理プログラムをコンパイル／アセンブルする際には，-no_callt オプションを指定する必要があります。

備考 4. RV850 の標準ヘッダ・ファイル Os.h では，AUTOSAR 標準ヘッダ・ファイル Std_Types.h，および MemMap.h のインクルードが行われます。このため，該当ヘッダ・ファイルのインクルードを行う必要がない場合には，各種ファイルをコンパイル／アセンブルする際に -D__WITHOSONLY__ オプションを指定する必要があります。
その際，構築するシステムで浮動小数点演算を全く用いない場合（-fnone オプション指定時）は，-D__WITHOSONLY__ オプションに加えて -D__NOFLOAT__ オプションを指定する必要があります。

備考 5. SIT ファイル（Os_Cfg.c）をコンパイルする際には -sda=0 オプションを指定することを推奨します。-sda=0 を指定しない場合には，表 2.1 の .kernel_const セクションと .kernel_work セクションに格納されることを想定している RV850 のデータが，それぞれ .sbss セクションと .rosdata セクションに配置される可能性があります。
特に，スケーラビリティ・クラスが SC3 の SIT ファイルに -sda=0 を指定しない際には，.sbss セクションと .rosdata セクションを，全ての Non-Trusted な OS アプリケーションからの書き込みアクセスが不可能な領域に配置する必要が生じます。

備考 6. RV850 では，共通フック・ルーチンのライブラリ化を禁止しています。

備考 7. フック・ルーチン，アイドル・ハンドラ，ユーザ・オウン・ライブラリについては，カーネル・ライブラリよりも先にリンクする必要があります。

備考 8. RV850 では，浮動小数点演算コプロセッサ（FPU）が非搭載のハードウエアにおけるロード・モジュール（“FPU 対応あり ” のカーネル・ライブラリ libecc2extsc1_fpu.a，libecc2extsc3_fpu.a がリンクされたロード・モジュール）の動作を保証していません。

備考 9. コンパイラ／アセンブラ／リンク・エディタについての詳細は，使用するコンパイラ・パッケージのユーザーズ・マニュアルを参照してください。

3. タスク管理

本章では，RV850 が提供しているタスク管理について解説しています。

3.1 概　　要

RV850 では，タスクの状態操作を実現するためのメカニズムとして “ タスク管理機能 ” を提供しています。
なお，RV850 では，タスクを実現すべき処理の用途に合わせて，OSEK/VDX 仕様で要求されている 2 種類のタスクをサポートしています。

(1) 基本タスク
イベントの割り付けが行われていないタスク（イベント識別子 “OsTaskEventRef” が未定義のタスク）であり，SUSPENDED 状態，READY 状態，RUNNING 状態といった 3 種類の状態に遷移可能です。基本タスクの動作中に使用するスタックは，以下の通りです。

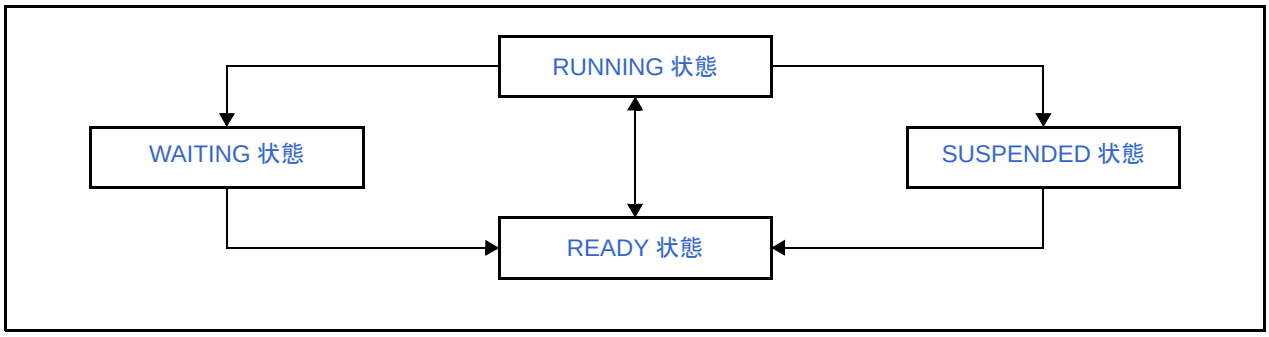
- スケーラビリティ・クラスが SC1 の場合
システム・スタック・サイズ “OsStackSize” で定義された「システム・スタック」
- スケーラビリティ・クラスが SC3 の場合
OS アプリケーション・スタック・サイズ “OsAppStackSize” で定義された「OS アプリケーション・スタック」

(2) 拡張タスク
イベントの割り付けが行われたタスク（イベント識別子 “OsTaskEventRef” が定義されたタスク）であり，SUSPENDED 状態，READY 状態，RUNNING 状態，WAITING 状態といった 4 種類の状態に遷移可能です。拡張タスクの動作中に使用するスタックは，タスク・スタック・サイズ “OsTaskStackSize” で定義された「タスク・スタック」です。

3.1.1 状　　態

タスクは，処理を実行するうえで必要となる資源の獲得状況，および事象の発生有無などにより，様々な状態へと遷移していきます。
なお，RV850 では，タスクが取り得る状態を 4 種類に大別し，管理しています。

図 3.1　　タスクの状態



(1) SUSPENDED 状態
RV850 の管理下にありながらもスケジューリング対象からは除外されている状態です。

(2) READY 状態
RV850 のスケジューリング対象でありながら，他タスクが処理を実行中のため，デバイスの使用権が割り当てられるのを待っている状態です。

(3) RUNNING 状態
デバイスの使用権が割り当てられ，タスクとしての処理を実行中の状態です。
なお，RUNNING 状態のタスクは，システム全体を通して同時に複数存在することはありません。

(4) WAITING 状態
タスクとしての処理を継続するうえで必要な条件が整わなかった際に遷移する状態です。

3.1.2 スタック・モニタリング機能

RV850 では，スタック・オーバーフローの発生を検知するための機能として，スタック・モニタリング機能を提供しています。この機能は，タスク動作中に RV850 の処理に移行したタイミングにおいて，RV850 の処理に必要なスタック (タスク・スタック，システム・スタック，OS アプリケーション・スタック) の残量が存在するかどうかをチェックします。具体的には，下記のタイミングでスタック残量のチェックを行います。

- タスクから発行したシステム・サービス処理の開始時
- タスク実行中に発生したカテゴリ 2 割り込みの前処理の開始時

タスク動作中にスタック・オーバーフローを検知した場合，スケーラビリティ・クラスが SC1 の際には ShutdownOS の発行が，スケーラビリティ・クラスが SC3 の際には ProtectionHook“OsProtectionHook” の定義内容にしたがった処理（共通フック・ルーチン ProtectionHook の呼び出し，または ShutdownOS の発行）が行われます。

備考 1. スタック・モニタリング機能の使用有無は，スタック・モニタリング機能 “OsStackMonitoring” で行います。

備考 2. RV850 のスタック・モニタリング機能では，タスクから RV850 の処理に移行しない限りは，スタック・オーバーフローの検出ができません。タスク動作中の任意のタイミングで発生しうるスタック・オーバーフローを検知する必要がある場合には，スケーラビリティ・クラス SC3 の Non-Trusted な OS アプリケーションで提供されるメモリ・アクセス保護機能を使用してください。

3.1.3 タ ス ク

システム・サービスを発行して状態操作を行わない限り，または CF ファイルで定義した条件が成立しない限り実行されることのない処理ルーチンです。
以下に，タスクを C 言語で記述する際の基本型を示します。

```
TASK ( OsTask ) {
      ……
      ……
}
```

備考 *OsTask* には，識別子 “OsTask” で定義したタスクの識別子を記述します。

3.1.4 タスクでの処理

RV850 では，タスクを切り替える際，独自のスケジューリング処理を行っています。このため，タスクを記述する際には，以下に示す注意点があります。

(1) レジスタの退避／復帰
RV850 では，タスクを切り替える際，C コンパイラの関数呼び出し規約に従った作業用レジスタの退避処理／復帰処理を行います。
したがって，“ レジスタの退避処理／復帰処理 ” は，記述する必要がありません。

(2) FPSR の退避／復帰
RV850 では，タスクを起動する際，浮動小数点設定／状態レジスタ（FPSR）の内容を FPSR“OsAppDefaultFPSRValue”，または FPSR のデフォルト値 “OsDefaultFPSRValue” で定義された値へと変更します。
したがって，FPSR の退避／復帰 “OsSaveFpuReg” で TRUE を定義している場合，“FPSR の退避処理／復帰処理 ” は，記述する必要がありません。

備考 1. “FPSR の退避処理／復帰処理 ” は，“FPU 対応あり ” のカーネル・ライブラリ libecc2extsc1_fpu.a，libecc2extsc3_fpu.a をリンクした場合に限り行われます。

備考 2. タスク内でインプレサイス例外モードの浮動小数点演算を使用する場合で，RV850 による FPSR の復帰処理前に演算結果を確定させる必要がある場合は，タスクの終了（TerminateTask ならびに ChainTask の発行）の直前に syncp 命令と synce 命令を発行してください。

(3) スタックの切り替え
RV850 では，タスクを切り替える際，タスク・スタック・サイズ “OsTaskStackSize” で定義されたタスク・スタック，システム・スタック・サイズ “OsStackSize” で定義されたシステム・スタック，または OS アプリケーション・スタック・サイズ “OsAppStackSize” で定義された OS アプリケーション・スタックへと切り替えます。
したがって，“ スタックの切り替え処理 ” を記述する必要がありません。

備考　システム・スタック・サイズ “OsStackSize” で定義されたシステム・スタックへの切り替えは，切り替え先のタスクが基本タスク（SC1）の場合に限られます。
また，OS アプリケーション・スタック・サイズ “OsAppStackSize” で定義された OS アプリケーション・スタックへの切り替えは，切り替え先のタスクが基本タスク（SC3）の場合に限られます。

(4) 割り込みの受け付け
RV850 では，タスクを切り替える際，割り込みの受け付けに関する操作を行いません。
したがって，割り込みの受け付けを明示的に変更する場合，タスクで “EnableAllInterrupts，DisableAllInterrupts などの発行処理 ” を記述する必要があります。

(5) システム・サービスの発行
“ タスクから発行可能なシステム・サービス ” のみが発行可能となります。

備考　各システム・サービスの発行有効範囲についての詳細は，「14.4 システム・サービス・リファレンス」を参照してください。

3.1.5 タスクの生成

RV850 では，タスクの生成を “ タスク情報の定義 ” による静的な生成に限定しています。したがって，処理プログラムからシステム・サービスを発行するなどの動的な生成はできません。

(1) 静的な生成
タスクの静的な生成は，CF ファイルにタスク情報を定義することにより実現されます。
なお，RV850 では，カーネル初期化部において，タスク情報の定義内容を情報ファイルから読み出し，該当初期化処理を行い，管理対象としています。

3.1.6 タスクの終了

タスクの終了（SUSPENDED 状態への遷移）は，TerminateTask，または ChainTask を発行することにより実現されます。
なお，RV850 では，タスクが TerminateTask，または ChainTask を発行することなく終了した場合，以下の操作を行います。

【スケーラビリティ・クラス SC1 の場合】

(1) タスクが “ インターナル・リソース ” を獲得していた場合，インターナル・リソースを解放
(2) タスクが “ インターナル・リソース ” を獲得していた場合，現在優先度を変更（タスクの現在優先度を初期優先度 “OsTaskPriority” に戻す）
(3) 共通フック・ルーチン（PostTaskHook）が登録されていた場合，PostTaskHook を呼び出す
(4) タスクを RUNNING 状態から SUSPENDED 状態へと遷移
(5) タスクを優先度に応じたレディ・キューから外す
(6) タスクが基本タスクの場合，起動要求カウンタを減算（起動要求カウンタから 0x1 を減算）
(7) 起動要求カウンタの減算結果が 0x0 以外の場合，タスクを SUSPENDED 状態から READY 状態へと遷移
(8) スケジューラを起動

【スケーラビリティ・クラス SC3 の場合】

(1) タスクが DisableAllInterrupts，SuspendAllInterrupts，SuspendOSInterrupts を発行したにも関わらず，対応する EnableAllInterrupts，ResumeAllInterrupts，ResumeOSInterrupts を非発行の場合，割り込みの受け付けを許可
(2) タスクが “ 通常のリソース ” を獲得していた場合，通常のリソースを解放
(3) タスクが “ 通常のリソース ” を獲得していた場合，現在優先度を変更（タスクの現在優先度を初期優先度 “OsTaskPriority” に戻す）
(4) タスクが “ インターナル・リソース ” を獲得していた場合，インターナル・リソースを解放
(5) タスクが “ インターナル・リソース ” を獲得していた場合，現在優先度を変更（タスクの現在優先度を初期優先度 “OsTaskPriority” に戻す）
(6) 共通フック・ルーチン（ErrorHook）が登録されていた際には，タスクが DisableAllInterrupts，SuspendAllInterrupts，SuspendOSInterrupts を発行したにも関わらず，対応する EnableAllInterrupts，

ResumeAllInterrupts，ResumeOSInterrupts を非発行の場合，またはタスクが “ 通常のリソース ” を獲得していた場合，E_OS_MISSINGEND（0x14）を引数とした ErrorHook を呼び出す

(7) タスクが所属している OS アプリケーションに固有フック・ルーチン（ErrorHook_*OsApplication*）が登録されていた際には，タスクが DisableAllInterrupts，SuspendAllInterrupts，SuspendOSInterrupts を発行したにも関わらず，対応する EnableAllInterrupts，ResumeAllInterrupts，ResumeOSInterrupts を非発行の場合，またはタスクが “ 通常のリソース ” を獲得していた場合，E_OS_MISSINGEND（0x14）を引数とした ErrorHook_*OsApplication* を呼び出す

(8) 共通フック・ルーチン（PostTaskHook）が登録されていた場合，PostTaskHook を呼び出す

(9) タスクを RUNNING 状態から SUSPENDED 状態へと遷移

(10) タスクを優先度に応じたレディ・キューから外す

(11) タスクが基本タスクの場合，起動要求カウンタを減算（起動要求カウンタから 0x1 を減算）

(12) 起動要求カウンタの減算結果が 0x0 以外の場合，タスクを SUSPENDED 状態から READY 状態へと遷移

(13) スケジューラを起動

備考 1. RV850 では，スケーラビリティ・クラスが SC1 の場合，“ 通常のリソース ” を獲得しているタスクが TerminateTask，または ChainTask を発行することなく終了した場合の動作を保証していません。

備考 2. RV850 では，スケーラビリティ・クラスが SC1 の場合，DisableAllInterrupts，SuspendAllInterrupts，SuspendOSInterrupts を発行したにも関わらず，対応する EnableAllInterrupts，ResumeAllInterrupts，ResumeOSInterrupts を非発行のタスクが TerminateTask，または ChainTask を発行することなく終了した場合の動作を保証していません。

備考 3. AUTOSAR 仕様では，スケーラビリティ・クラス “OsScalabilityClass” に SC1 を定義し，TerminateTask，または ChainTask を発行することなくタスクが終了した場合の動作が規定されていません。
したがって，上記に示した SC1 の場合における動作は，RV850 独自のものとなります。

3.2 システム・サービス

処理プログラムからタスクに関する動的な操作は，「14.4.1 タスク管理」で示したシステム・サービスを用いて行います。

4. 割り込み管理

本章では，RV850 が提供している割り込み管理について解説しています。

4.1 概　　要

RV850 では，デバイスが発生させる割り込みに対応した処理を実現するためのメカニズムとして “ 割り込み管理機能 ” を提供しています。
RV850 では，デバイスが発生させる割り込みのうち，EI レベル割り込み（例外コード “OsIsrExceptionCode” ／例外コード “OsCounterExceptionCode” で定義）に対応した処理のみを割り込みサービス・ルーチンとして扱います。
したがって，リセット （RESET），FE レベル割り込み （FENMI，TRAP），EI レベル割り込み （例外コード “OsIsrExceptionCode” ／例外コード “OsCounterExceptionCode” で未定義）などに対応した処理 （ブート処理，例外／割り込み安全対策処理など）は，割り込みサービス・ルーチンとして扱われず，特別な扱いとなります。

4.1.1 スタック・モニタリング機能

RV850 では，スタック・オーバーフローの発生を検知するための機能として，スタック・モニタリング機能を提供しています。この機能は，割り込みサービス・ルーチン（カテゴリ 2）の動作中に RV850 の処理に移行したタイミングにおいて，RV850 の処理に必要なスタック (システム・スタック，OS アプリケーション・スタック) の残量が存在するかどうかをチェックします。具体的には，下記のタイミングでスタック残量のチェックを行います。

- 割り込みサービス・ルーチン（カテゴリ 2）から発行したシステム・サービス処理の開始時
- 割り込みサービス・ルーチン（カテゴリ 2）実行中に発生したカテゴリ 2 割り込みの前処理の開始時

割り込みサービス・ルーチン（カテゴリ 2）動作中にスタック・オーバーフローを検出した場合，スケーラビリティ・クラスが SC1 の際には ShutdownOS の発行が，スケーラビリティ・クラスが SC3 の際には ProtectionHook“OsProtectionHook” の定義内容にしたがった処理（共通フック・ルーチン ProtectionHook の呼び出し，または ShutdownOS の発行）が行われます。

備考 1. RV850 では，カテゴリ 2 の割り込みサービス・ルーチンに限り，スタック・モニタリング機能を適用しています。

備考 2. スタック・モニタリング機能の使用有無は，スタック・モニタリング機能 “OsStackMonitoring” で行います。

備考 3. RV850 のスタック・モニタリング機能では，割り込みサービス・ルーチン（カテゴリ 2）から RV850 の処理に移行しない限りは，スタック・オーバーフローの検出ができません。
割り込みサービス・ルーチン動作中の任意のタイミングで発生しうるスタック・オーバーフローを検知する必要がある場合には，スケーラビリティ・クラス SC3 の Non-Trusted な OS アプリケーションで提供されるメモリ・アクセス保護機能を使用してください。

4.2 ブート処理

RV850 が処理を実行するうえで必要となる最低限のハードウエアを初期化するために切り出された初期処理専用ルーチンであり，ハードウエア・リセットが発生した際にデバイスが強制的に制御を移すハンドラ・アドレスに割り付けられた分岐処理（エントリ処理（直接分岐方式の例外ベクタ））から呼び出されます。
なお，ブート処理では，最低限のハードウエア初期化処理として，以下の操作が必要となります。

(1) 基本システム・レジスタの初期化

- プログラム・ステータス・ワード（PSW）の EBV ビット，CU*n* ビットに対する操作
- リセット・ベクタ・ベース・アドレス（RBASE）の初期化
- 例外ハンドラ・ベクタ・アドレス（EBASE）の初期化
- CPU の制御（MCTL）の UIC ビットに対する操作

(2) 割り込み機能レジスタの初期化 (G3K，G3M，G3KH 限定)

- FPI 例外割り込み優先度設定（FPIPR）の初期化

(3) 割り込み制御レジスタの初期化

- EI レベル割り込みバインド・レジスタ（EIBD*n*）の初期化

(4) マシン・コンフィギュレーション（MCFG0）の SPID ビット（システム・プロテクション番号）と連動した安全機能に関する初期化（G3M，G3KH，G3MH 限定）

(5) 周辺コントローラ（タイマなど）の初期化

備考 1. RV850 では，FPU 機能レジスタ（FPSR）に対する操作を行う際に限り PSW の CU*n* ビットを操作（CU0 ビットに 1 を設定）し，該当操作完了時には，CU*n* ビットの値を操作前の値に戻しています。したがって，RV850 は，CU*n* ビットについては，ブート処理で設定された値を引き継いで動作を開始し，処理プログラムから動的に CU*n* ビットに対する操作が行われた際にも，変更後の値を引き継いで動作を行うことになります。

備考 2. RV850 では，割り込みハンドラ・アドレスの選択方式として，テーブル参照方式を採用しているため，RBASE の RINT ビット，EBASE の RINT ビットには，0 を設定してください。

備考 3. MCTL の UIC ビットに対する操作は，処理プログラムで EI 命令／DI 命令を発行する場合に限り必要となります。

備考 4. FPIPR の FPIPR ビットには，割り込みサービス・ルーチン情報で定義したカテゴリ 2 の割り込みサービス・ルーチンよりも高い優先度を設定してください。

備考 5. I/O 領域に対するアクセス権に関する操作は，スケーラビリティ・クラス “OsScalabilityClass” で SC3 を定義した場合に限り必要となります。

備考 6. MCFG0 レジスタの SPID ビットがシステム SPID 値 “OsSystemSystemSPID” で指定した値の際には，I/O 領域に対する全アクセスが可能となる設定を行ってください。
SPID についての詳細は，対象デバイスのユーザーズ・マニュアルを参照してください。

備考 7. AUTOSAR 仕様では，タイマの初期化をリアルタイム OS で行う旨の規定（OS374）が行われていますが，RV850 では，StartOS の発行前（RV850 の起動前）に行うようにしています。

以下に，ブート処理をアセンブリ言語で記述する際の基本型を示します。

```
    .tex
    .align  4
    .globl  _boot
_boot:
    ……
    ……
```

4.2.1 ブート処理での処理

ブート処理は，RV850 が起動する以前に呼び出される初期化処理専用ルーチンです。このため，ブート処理を記述する際には，以下に示す注意点があります。

(1) レジスタの退避／復帰
ブート処理を実行するうえで退避／復帰を必要とするレジスタは存在しません。
したがって，“ レジスタの退避処理／復帰処理 ” は，記述する必要がありません。

(2) FPSR の退避／復帰
ブート処理が開始された時点では，浮動小数点設定／状態レジスタ（FPSR）の設定が行われていません。
したがって，“FPSR の退避処理／復帰処理 ” は，記述する必要がありません。

(3) スタックの切り替え
ブート処理が開始された時点では，スタック・ポインタ（SP）の設定が行われていません。
したがって，ブート処理専用スタックを使用する場合，“SP の設定処理 ” を記述する必要があります。

(4) システム・サービスの発行
“ ブート処理から発行可能なシステム・サービス ” のみが発行可能となります。

備考 各システム・サービスの発行有効範囲についての詳細は，「14.4 システム・サービス・リファレンス」を参照してください。

4.3 割り込みサービス・ルーチン

EI レベル割り込み（例外コード “OsIsrExceptionCode” ／例外コード “OsCounterExceptionCode” で定義）が発生した際に呼び出される割り込み処理専用ルーチンです。

なお，RV850 では，割り込みサービス・ルーチンを “ タスクとは独立したもの ” として位置付けています。このため，割り込みが発生した際には，システムで最高位の優先度を持つタスクが処理を実行中であっても，その処理は中断され，割り込みサービス・ルーチンに制御が移行します。

また，RV850 では，割り込みの発生から割り込みサービス・ルーチンを呼び出すまでの応答性を考慮し，2 種類のカテゴリをサポートしています。

(1) カテゴリ 1

割り込みが発生した際，RV850 を介在させることなく呼び出される割り込み処理専用ルーチンです。

なお，カテゴリ 1 の割り込みサービス・ルーチンは，割り込みが発生した際にデバイスが強制的に制御を移すハンドラ・アドレスから直接呼び出されるため，ハードウエアの限界に近い応答性を有しています。

以下に，カテゴリ 1 の割り込みサービス・ルーチンを C 言語で記述する際の基本型を示します。

【#pragma ghs interrupt 指令を使用】

```
#pragma ghs interrupt
ISR ( OsIsr ) {
    ……
    ……
    return;
}
```

【#pragma ghs interrupt (enable) 指令を使用】

```
#pragma ghs interrupt (enable)
ISR ( OsIsr ) {
    ……
    ……
    return;
}
```

以下に，カテゴリ 1 の割り込みサービス・ルーチンをアセンブリ言語で記述する際の基本型を示します。

```
    .text
    .align  4
    .globl  _ISROsIsr
_ISROsIsr:
    addi    -16, sp, sp
    st.w    r1, 0[sp]
    ……
    ……
    ld.w    0[sp], r1
    addi    16, sp, sp
    eiret
```

備考　カテゴリ 1 の割り込みサービス・ルーチンは，スーパバイザ・モードで動作するため，デバイスが提供しているメモリ保護機能（MPU）は適用外となります。

(2) カテゴリ 2
割り込みが発生した際，RV850 が提供している割り込み前処理（レジスタの退避，スタックの切り替えなど）を実行したあとに呼び出される割り込み処理専用ルーチンです。
なお，カテゴリ 2 の割り込みサービス・ルーチンは，割り込みが発生した際にデバイスが強制的に制御を移すハンドラ・アドレスから RV850 が提供している割り込み前処理を経由して呼び出されるため，割り込みサービス・ルーチン内の処理を簡素化することが可能となります。
以下に，カテゴリ 2 の割り込みサービス・ルーチンを C 言語で記述する際の基本型を示します。

```
ISR ( OsIsr ) {
    ……
    ……
    return;
}
```

備考 *OsIsr* には，識別子 “OsIsr” で定義した割り込みサービス・ルーチンの識別子を記述します。

4.3.1 割り込みサービス・ルーチンでの処理

RV850 では，割り込みの発生から割り込みサービス・ルーチンを起動するまでの応答性を考慮し，2 種類のカテゴリをサポートしています。このため，割り込みサービス・ルーチンを記述する際には，カテゴリ別に以下に示す注意点があります。

(1) カテゴリ 1

(a) レジスタの退避／復帰
RV850 では，カテゴリ 1 の割り込みサービス・ルーチンの呼び出しに関与していません。
したがって，“ 処理内容に合わせたレジスタの退避処理／復帰処理 ” を記述する必要があります。

(b) FPSR の退避／復帰
RV850 では，カテゴリ 1 の割り込みサービス・ルーチンの呼び出しに関与していません。
したがって，FPSR の内容を明示的に変更する場合，“FPSR の退避処理／復帰処理 ” を記述する必要があります。

備考 カテゴリ 1 の割り込みサービス・ルーチン内でインプレサイス例外モードの浮動小数点演算を使用する場合で，割り込み発生時点の処理に移行する前に演算結果を確定させる必要がある場合は，カテゴリ 1 の割り込みサービス・ルーチン終了直前に syncp 命令と synce 命令を発行してください。

(c) スタックの切り替え
RV850 では，カテゴリ 1 の割り込みサービス・ルーチンの呼び出しに関与していません。
したがって，カテゴリ 1 の割り込みサービス・ルーチン専用スタックを使用する場合，“ スタック・ポインタ（SP）の設定処理 ” を記述する必要があります。

(d) 割り込みの受け付け
RV850 では，カテゴリ 1 の割り込みサービス・ルーチンの呼び出しに関与していません。
このため，該当割り込みの発生時にデバイスによるプログラム・ステータス・ワード（PSW）の ID ビット操作が行われ，割り込みの受け付けが禁止されています。
したがって，割り込みの受け付けを明示的に変更する場合，“EI 命令／ DI 命令の発行処理 ” を記述する必要があります。

(e) システム・サービスの発行
“ 割り込みサービス・ルーチンから発行可能なシステム・サービス ” のみが発行可能となります。

備考 1. RV850 では，割り込みサービス・ルーチンから発行不可なシステム・サービスが発行された場合の動作を保証していません。

備考 2. 各システム・サービスの発行有効範囲についての詳細は，「14.4 システム・サービス・リファレンス」を参照してください。

(2) カテゴリ 2

(a) レジスタの退避／復帰
RV850 では，カテゴリ 2 の割り込みサービス・ルーチンに制御を移す際，C コンパイラの関数呼び出し規約に従った作業用レジスタの退避処理／復帰処理を行います。加えて，スケーラビリティ・クラスが SC3 の際には，TerminateApplication による OS アプリケーションの終了処理を考慮したレジスタの退避処理／復帰処理を行います。
したがって，“ レジスタの退避処理／復帰処理 ” は，記述する必要がありません。

(b) FPSR の退避／復帰
RV850 では，カテゴリ 2 の割り込みサービス・ルーチンに制御を移す際，浮動小数点設定／状態レジスタ（FPSR）の内容を FPSR“OsAppDefaultFPSRValue”，または FPSR のデフォルト値 “OsDefaultFPSRValue” で定義された値へと変更します。
したがって，FPSR の退避／復帰 “OsSaveFpuReg” で TRUE を定義している場合，“FPSR の退避処理／復帰処理 ” は，記述する必要がありません。

備考 1. “FPSR の退避処理／復帰処理 ” は，“FPU 対応あり ” のカーネル・ライブラリ libecc2extsc1_fpu.a，libecc2extsc3_fpu.a をリンクした場合に限り行われます。

備考 2. カテゴリ 2 の割り込みサービス・ルーチン内でインプレサイス例外モードの浮動小数点演算を使用する場合で，RV850 による FPSR の復帰処理前に演算結果を確定させる必要がある場合は，カテゴリ 2 の割り込みサービス・ルーチン終了直前に syncp 命令と synce 命令を発行してください。

(c) スタックの切り替え
RV850 では，カテゴリ 2 の割り込みサービス・ルーチンに制御を移す際，システム・スタック・サイズ “OsStackSize” で定義されたシステム・スタック，または OS アプリケーション・スタック・サイズ “OsAppStackSize” で定義された OS アプリケーション・スタックへと切り替えます。
したがって，“ スタックの切り替え処理 ” を記述する必要がありません。

(d) 割り込みの受け付け
RV850 では，カテゴリ 2 の割り込みサービス・ルーチンに制御を移す際，割り込みの受け付けを許可状態（PSW の ID ビットに対する操作）としています。
したがって，割り込みの受け付けを明示的に変更する場合，“DisableAllInterrupts，SuspendAllInterrupts などの発行処理 ” を記述する必要があります。

備考 割り込みの受け付けを PSW の ID ビット操作によって変更する場合は，カテゴリ 2 の割り込みサービス・ルーチンが終了する前には元に戻すようにしてください。

(e) システム・サービスの発行
“ 割り込みサービス・ルーチン，または割り込みサービス・ルーチン（カテゴリ 2）から発行可能なシステム・サービス ” のみが発行可能となります。

備考 各システム・サービスの発行有効範囲についての詳細は，「14.4 システム・サービス・リファレンス」を参照してください。

4.3.2 割り込みサービス・ルーチンの登録

RV850 では，割り込みサービス・ルーチンの登録を “ 割り込みサービス・ルーチン情報の定義 ” による静的な登録に限定しています。したがって，処理プログラムからシステム・サービスを発行するなどの動的な生成はできません。

(1) 静的な登録
割り込みサービス・ルーチンの静的な登録は，CF ファイルに割り込みサービス・ルーチン情報を定義することにより実現されます。
なお，RV850 では，カーネル初期化部において，割り込みサービス・ルーチン情報の定義内容を情報ファイルから読み出し，該当初期化処理を行い，管理対象としています。

RV850 では，デバイスが発生させる割り込みのうち，例外コード “OsIsrExceptionCode” ／例外コード “OsCounterExceptionCode” で定義された EI レベル割り込みに対応した割り込みサービス・ルーチンのみを管理対象としています。
したがって，EI レベル割り込み以外の割り込みに対応した割り込み処理専用ルーチンは，コンパイラが提供している #pragma 指令を利用し，RV850 の管理対象外処理プログラムとして登録を行ってください。

4.3.3 割り込みサービス・ルーチンの終了

割り込みサービス・ルーチンの終了（割り込みの発生した処理プログラムに制御を戻す）は，return 命令（アセンブリ言語の場合は eiret 命令）を発行することにより実現されます。
なお，RV850 では，割り込みサービス・ルーチンが return 命令を発行した場合，以下の操作を行います。

(1) カテゴリ 1，スケーラビリティ・クラス SC1 ／ SC3 の場合

- 割り込みの発生した処理プログラムに制御を戻す

(2) カテゴリ 2，スケーラビリティ・クラス SC1 の場合

- 作業用レジスタを復帰

- スタックを切り替える
- 浮動小数点設定／状態レジスタ（FPSR）を復帰
- スケジューラを起動

(3) カテゴリ 2，スケーラビリティ・クラス SC3 の場合

- 作業用レジスタを復帰
- スタックを切り替える
- 浮動小数点設定／状態レジスタ（FPSR）を復帰
- 割り込みサービス・ルーチンが DisableAllInterrupts，SuspendAllInterrupts などを発行したにも関わらず，対応する EnableAllInterrupts，ResumeAllInterrupts などを非発行の場合，割り込みの受け付けを許可
- 割り込みサービス・ルーチンが所属している OS アプリケーションに固有フック・ルーチン（ErrorHook_*OsApplication*）が登録されていた際には，割り込みサービス・ルーチンが DisableAllInterrupts，SuspendAllInterrupts などを発行したにも関わらず，対応する EnableAllInterrupts，ResumeAllInterrupts などを非発行の場合，E_OS_DISABLEDINT（0x15）を引数とした ErrorHook_*OsApplication* を呼び出す
- 共通フック・ルーチン（ErrorHook）が登録されていた際には，割り込みサービス・ルーチンが DisableAllInterrupts，SuspendAllInterrupts などを発行したにも関わらず，対応する EnableAllInterrupts，ResumeAllInterrupts などを非発行の場合，E_OS_DISABLEDINT（0x15）を引数とした ErrorHook を呼び出す
- 割り込みサービス・ルーチンが “ 通常のリソース ” を獲得していた場合，通常のリソースを解放
- 割り込みサービス・ルーチンが “ 通常のリソース ” を獲得していた場合，現在優先度を変更（割り込みサービス・ルーチンの現在優先度を初期優先度 “OsIsrPriority” に戻す）
- 割り込みサービス・ルーチンが “ 通常のリソース ” を獲得していた場合，割り込みの受け付けを許可（優先度 “INTPRI0 ～シーリング値 ” に対応した割り込み要因の受け付けを許可）
- 割り込みサービス・ルーチンが所属している OS アプリケーションに固有フック・ルーチン（ErrorHook_*OsApplication*）が登録されていた際には，割り込みサービス・ルーチンが “ 通常のリソース ” を獲得していた場合，E_OS_RESOURCE（0x6）を引数とした ErrorHook_*OsApplication* を呼び出す
- 共通フック・ルーチン（ErrorHook）が登録されていた際には，割り込みサービス・ルーチンが “ 通常のリソース ” を獲得していた場合，E_OS_RESOURCE（0x6）を引数とした ErrorHook を呼び出す
- スケジューラを起動

備考 1. RV850 では，カテゴリ 1 の場合，DisableAllInterrupts，SuspendAllInterrupts などを発行したにも関わらず，対応する EnableAllInterrupts，ResumeAllInterrupts などを非発行の割り込みサービス・ルーチンが return 命令を発行した場合の動作を保証していません。

備考 2. FPSR の復帰は，“FPU 対応あり ” のカーネル・ライブラリ libecc2extsc1_fpu.a，libecc2extsc3_fpu.a をリンクした場合に限り行われます。

4.4 システム・サービス

処理プログラムから割り込みに関する動的な操作は，「14.4.2 割り込み管理」で示したシステム・サービスを用いて行います。

4.5 ユーザ・オウン・コーディング部

RV850 では，様々な実行環境に対応するために，RV850 が処理を実行するうえで必要となるハードウエア依存処理を RV850 依存部（ユーザ・オウン・コーディング部）として切り出し，サンプル・ソース・ファイルを提供しています。

これにより，様々な実行環境への移植性を向上させると伴に，カスタマイズ化を容易にしています。

4.5.1 エントリ処理（直接分岐方式の例外ベクタ）

リセット（RESET），FE レベル割り込み（FENMI，TRAP など），EI レベル割り込み（例外コード “OsIsrExceptionCode” ／例外コード “OsCounterExceptionCode” で未定義）などが発生した際，対応する処理（ブート処理，例外／割り込み安全対策処理など）への分岐処理を割り付けるために切り出されたエントリ処理専用ルーチンです。

以下に，エントリ処理をアセンブリ言語で記述する際の基本型を示します。

【SC1，RBASE ／ EBASE：0x0 の場合】

```
    .globl  _entry0000
    .globl  _entry0010
    ……
    .globl  _entry0100
    ……

    .org    0x00000000
_entry0000:
    jr      _boot

    .org    0x00000010
_entry0010:
    jr      __kernel_e_IllegalExcEntry

    ……

    .org    0x00000100
_entry0100:
    jr      __kernel_e_IllegalExcEntry

    ……
```

【SC3，RBASE ／ EBASE：0x0 の場合】

```
    .globl  _entry0000
    .globl  _entry0010
    .globl  _entry0020
    ……
    .globl  _entry0090
    .globl  _entry00A0
    .globl  _entry00B0
    ……
    .globl  _entry0100
    ……

    .org    0x00000000
_entry0000:
    jr      _boot

    .org    0x00000010
_entry0010:
    jr      __kernel_e_ProtectEntry

    .org    0x00000020
_entry0020:
    jr      __kernel_e_IllegalExcEntry

    ……

    .org    0x00000090
_entry0090:
    jr      __kernel_e_ProtectEntry

    .org    0x000000A0
_entry00A0:
    jr      __kernel_e_ProtectEntry

    .org    0x000000B0
_entry00B0:
    jr      __kernel_e_IllegalExcEntry

    ……

    .org    0x00000100
_entry0100:
    jr      __kernel_e_IllegalExcEntry

    ……
```

備考 1. 例外コード “OsIsrExceptionCode” ／例外コード “OsCounterExceptionCode” で定義された EI レベル割り込みに関するエントリ処理については，コンフィギュレータが ENTRY ファイルに出力しているため，ユーザが該当エントリ処理を記述する必要はありません。

備考 2. スケーラビリティ・クラスが SC3 の場合，システム・エラー例外（+0x10 番地），メモリ・プロテクション例外（+0x90 番地），特権命令例外（+0xA0 番地）については，保護例外に対応した処理 _kernel_e_ProtectEntry への分岐処理を記述してください。

4.5.2 例外／割り込み安全対策処理

FE レベル割り込み（FENMI，TRAP など），EI レベル割り込み（例外コード “OsIsrExceptionCode” ／例外コード “OsCounterExceptionCode” で未定義）などが発生した際，エントリ処理から呼び出される安全対策処理専用ルーチンです。

以下に，例外／割り込み安全対策処理を C 言語で記述する際の基本型を示します。

```
#pragma ghs interrupt

void
_kernel_e_IllegalExcEntry ( void ) {
     ……
     ……
}
```

例外／割り込み安全対策処理を記述する際には，以下に示す注意点があります。

(1) レジスタの退避／復帰
RV850 は，例外／割り込み安全対策処理ルーチンの呼び出しに関与していません。したがって，ルーチンの処理内容に合わせたレジスタの退避処理／復帰処理を記述する必要があります。

(2) FPSR の退避／復帰
RV850 は，例外／割り込み安全対策処理ルーチンの呼び出しに関与していません。したがって，FPSR の内容を明示的に変更する場合，"FPSR の退避処理／復帰処理 " を記述する必要があります。

(3) スタックの切り替え
RV850 は，例外／割り込み安全対策処理ルーチンの呼び出しに関与していません。したがって，例外／割り込み安全対策処理ルーチン専用に別途スタックを確保して使用する場合，" スタック・ポインタ（SP）の設定処理 " を記述する必要があります。

(4) 割り込みの受け付け
例外／割り込み安全対策処理ルーチンが呼び出された時点では，デバイスによるプログラム・ステータス・ワード（PSW）の ID ビット操作が行われ，割り込みの受け付けが禁止されています。

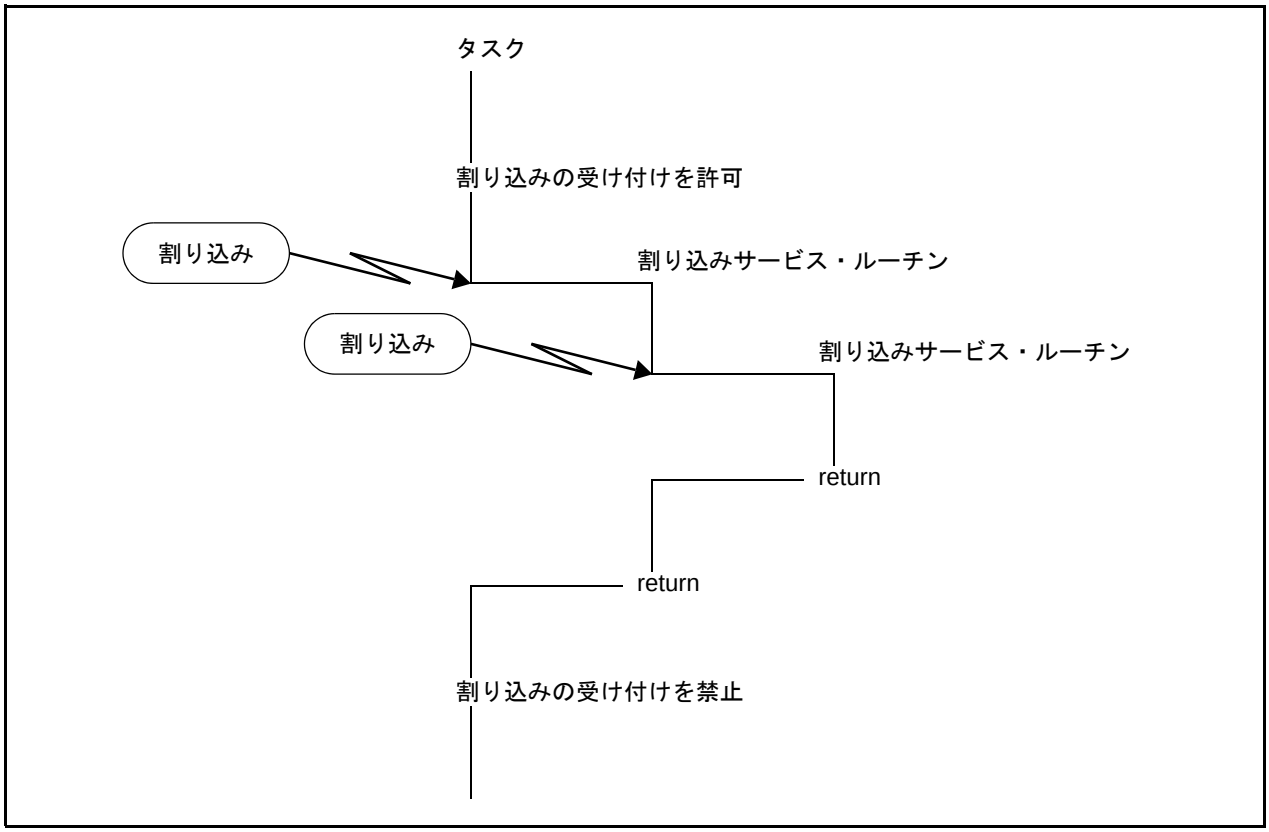
(5) システム・サービスの発行
例外／割り込み安全対策処理ルーチンからシステム・サービスを呼び出すことはできません。

備考 RV850 では，デフォルトの例外／割り込み安全対策処理を用意しています。
このため，_kernel_e_IllegalExcEntery の記述が行われていない場合であっても，エントリ処理に例外／割り込み安全対策処理への分岐処理が割り付けられていた際，および SYSCALL 命令の発行時に不正な値が指定された際には，デフォルトの例外／割り込み安全対策処理（引数 *Error* に E_OS_SYS_ILLEGAL_EXCEPTION を指定した ShutdownOS の発行処理）が呼び出されます。

4.6 多重割り込み

RV850 では，割り込みサービス・ルーチンが処理を実行中に再び割り込みが発生することを “ 多重割り込み ” と呼んでいます。

図 4.1 多重割り込み



備考 カテゴリ 1 の割り込みサービス・ルーチンの初期優先度 “OsIsrPriority” には，カテゴリ 2 のどの割り込みサービス・ルーチンよりも高い優先度（ハードウエア・カウンタに指定された優先度 “OsCounterPriority” を含む）が指定されているため，カテゴリ 1 の割り込みサービス・ルーチンが処理を実行中に受け付けられる割り込みは，カテゴリ 1 割り込みとなります。

5. リソース管理

本章では，RV850 が提供しているリソース管理について解説しています。

5.1 概　　要

RV850 では，排他制御のためのメカニズムとして “ リソース管理機能 ” を提供しています。リソースは “ 優先度上限プロトコル (Priority Ceiling Plotocols)” によって排他制御を実現し，限られた数の共有資源（データ，周辺機器，共通関数等）を使用するタスクや割り込みサービス・ルーチン（カテゴリ 2）による資源競合やデッドロックを防止できます。
RV850 では，3 種類のリソースをサポートしています。

(1) 通常のリソース
処理プログラムから GetResource，ReleaseResource を発行することにより，動的に獲得／解放することが可能なリソースです。

(2) インターナル・リソース
タスクが READY 状態から RUNNING 状態へと遷移した際に RV850 により自動的に割り付けられるリソースです。

(3) リンクト・リソース
他リソースから属性（シーリング値，OS アプリケーション識別子）を引き継いだリソースです。

5.1.1 シーリング値

シーリング値は，処理プログラム（タスク，割り込みサービス・ルーチン（カテゴリ 2））がリソースを獲得してから解放するまでの区間に与えられる優先度となります。
したがって，優先度 3 のタスクがシーリング値 10 のリソースを獲得した際には，該当リソースを解放するまでの区間は，優先度 10 のタスクとして動作するため，優先度 0 ～ 10 のタスクへと制御が移ることがありません。
また，優先度 3 のタスクがシーリング値 INTPRI5 のリソースを獲得した際には，該当リソースを解放するまでの区間は，最高優先度 29 のタスクとして動作すると伴に，INTPRI0 ～ INTPRI5 の割り込みの受け付けも禁止状態となるため，優先度 0 ～ 29 のタスク，および INTPRI0 ～ INTPRI5 の割り込みサービス・ルーチンへと制御が移ることがありません。

備考　　INTPRI0 ～ INTPRI15 のシーリング値のリソースの獲得区間中に保留された割り込みは，該当リソースの解放時に受け付けられます。

5.1.2 スケジューラ・リソース

OSEK/VDX 仕様では，タスクからスケジューラの起動を動的に禁止／許可するための手段として，シーリング値：29，識別子：RES_SCHEDULER を持つリソースを規定しています。
タスクから GetResource（引数 *ResID* に RES_SCHEDULER）を発行することにより，該当システム・サービスの発行から ReleaseResource（引数 *ResID* に RES_SCHEDULER）を発行するまでの区間がスケジューラの起動は禁止されます。

備考 1.　　RV850 は AUTOSAR 仕様に準拠するため，スケジューラ・リソースの自動的な定義は行われません。RV850 においてスケジューラ・リソース RES_SCHEDULER を有効にする方法については，「5.2 リソースの生成」を参照してください。

備考 2.　　スケジューラ・リソースのシーリング値は 29 であり，割り込みサービス・ルーチンが関連する排他制御には使用できません。

5.2 リソースの生成

RV850 では，リソースの生成を “ リソース情報の定義 ” による静的な生成に限定しています。したがって，処理プログラムからシステム・サービスを発行するなどの動的な生成はできません。

(1) 静的な生成
リソースの静的な生成は，CF ファイルにリソース情報を定義することにより実現されます。
なお，RV850 では，カーネル初期化部において，リソース情報の定義内容を情報ファイルから読み出し，該当初期化処理を行い，管理対象としています。

備考 OSEK/VDX 仕様では，スケジューラ・リソースの生成は，スケジューラ・リソース “OsUseResScheduler” で TRUE を定義することで自動的に行われることとなっていますが，AUTOSAR 仕様では自動的な生成は行わないとしています。
そのため，RV850 においてスケジューラ・リソースを生成する際には，以下のように CF ファイルに定義してください。

- リソース情報の識別子 “OsResource” で RES_SCHEDULER を指定
- リソース情報のシーリング値 “OsResourcePriority” で 29 を指定

5.3 システム・サービス

処理プログラムからリソースに関する動的な操作は，「14.4.3 リソース管理」で示したシステム・サービスを用いて行います。

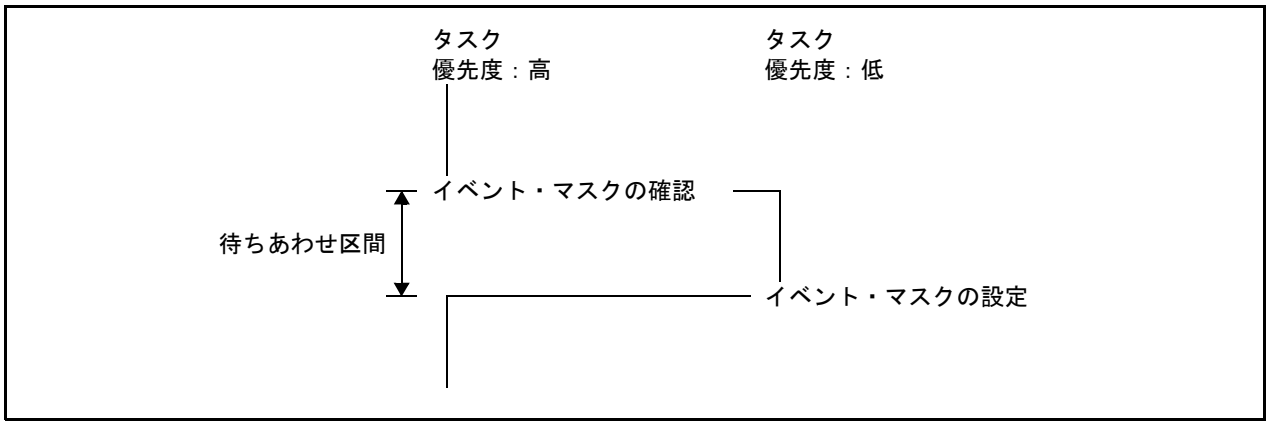
6. イベント管理

本章では，RV850 が提供しているイベント管理について解説しています。

6.1 概　　要

RV850 では，“ あるタスクの処理結果が出るまでの間，処理の実行を待つ ” といったタスク間の待ちあわせを実現するためのメカニズムとして “ イベント管理機能 ” を提供しています。

図 6.1　　タスク間の待ちあわせ



備考　　RV850 では，イベントが割り付けられているか否か（イベント識別子 “OsTaskEventRef” の定義が行われているか否か）により，基本タスク，拡張タスクの区別を行っています。
なお，RV850 では，拡張タスクに割り付け可能なイベントの最大数を 32 個としています。

6.2 イベントの生成

RV850 では，イベントの生成を “ イベント情報の定義 ” による静的な生成に限定しています。したがって，処理プログラムからシステム・サービスを発行するなどの動的な生成はできません。

(1)　静的な生成
イベントの静的な生成は，CF ファイルにイベント情報を定義することにより実現されます。
なお，RV850 では，カーネル初期化部において，イベント情報の定義内容を情報ファイルから読み出し，該当初期化処理を行い，管理対象としています。

6.3 システム・サービス

処理プログラムからイベントに関する動的な操作は，「14.4.4 イベント管理」で示したシステム・サービスを用いて行います。

7. カウンタ管理

本章では，RV850 が提供しているカウンタ管理について解説しています。

7.1 概　　要

RV850 では，事象の発生回数に対応した処理を行うためのメカニズムとして “ カウンタ管理機能 ” を提供し，2 種類のカウンタをサポートしています。

(1) ソフトウエア・カウンタ
カウント値の更新は，処理プログラムからシステム・サービス “IncrementCounter” が呼び出された際に行われます。

(2) ハードウエア・カウンタ
カウント値の更新は，カウンタ情報で定義した例外コード “OsCounterExceptionCode” に対応した割り込みが発生した際に行われます。

7.1.1 システム・カウンタ

RV850 では，OSEK/VDX 仕様で要求されているシステム・カウンタをサポートしていますが，識別子 “OsCounter” に規定されたキーワード “SYS_COUNTER” を定義する以外，通常のカウンタと取り扱い方法，および利用方法に差異を設けていません。

7.2 カウンタの生成

RV850 では，カウンタの生成を “ カウンタ情報の定義 ” による静的な生成に限定しています。したがって，処理プログラムからシステム・サービスを発行するなどの動的な生成はできません。

(1) 静的な生成
カウンタの静的な生成は，CF ファイルにカウンタ情報を定義することにより実現されます。
なお，RV850 では，カーネル初期化部において，カウンタ情報の定義内容を情報ファイルから読み出し，該当初期化処理を行い，管理対象としています。

備考　RV850 では，対象カウンタが “ ハードウエア・カウンタ ” の場合，生成処理の一環として，カウント値の更新を行うための割り込みサービス・ルーチン（カテゴリ 2）の登録処理もあわせて行います。

7.3 システム・サービス

処理プログラムからカウンタに関する動的な操作は，「14.4.5 カウンタ管理」で示したシステム・サービスを用いて行います。

8. アラーム管理

本章では，RV850 が提供しているアラーム管理について解説しています。

8.1 概　　要

RV850 では，カウンタのもつカウント値の変化に同期した処理を行うための機能として “ アラーム管理機能 ” を提供しています。
なお，アラーム管理機能では，カウンタ識別子 “OsAlarmCounterRef” において関連付けを行ったカウンタのカウント値が満了条件（満了カウント値，周期カウント値）を満足した際には，タスクの起動，イベント・マスクの設定，カウント値の更新，アラーム・コールバックの呼び出し（SC1 限定）といった満了処理を実行します。

8.2 アラーム・コールバック

アラームが満了した際に呼び出される満了処理専用ルーチンです。
なお，RV850 では，アラーム・コールバックを “ タスクとは独立したもの ” として位置付けています。このため，アラームが満了した際には，システムで最高位の優先度を持つタスクが処理を実行中であっても，その処理は中断され，アラーム・コールバックに制御が移行します。
以下に，アラーム・コールバックを C 言語で記述する際の基本型を示します。

```
ALARMCALLBACK ( OsAlarmCallbackName ) {
    ……
    ……
    return;
}
```

備考　　*OsAlarmCallbackName* には，アラーム情報で定義したアラーム・コールバックの識別子を記述します。

8.2.1 アラーム・コールバックでの処理

RV850 では，各種処理プログラムからアラーム・コールバックに制御を移す際，独自の前処理を行っています。
また，アラーム・コールバックから各種処理プログラムに制御を戻す際にも，独自の後処理を行っています。このため，アラーム・コールバックを記述する際には，以下に示す注意点があります。

(1) レジスタの退避／復帰
RV850 では，アラーム・コールバックに制御を移す際，C コンパイラの関数呼び出し規約に従った作業用レジスタの退避処理／復帰処理を行います。
したがって，“ レジスタの退避処理／復帰処理 ” を記述する必要がありません。

(2) FPSR の退避／復帰
RV850 では，アラーム・コールバックに制御を移す際，浮動小数点設定／状態レジスタ（FPSR）の退避処理／復帰処理を行いません。
したがって，FPSR の内容を明示的に変更する場合，“FPSR の退避処理／復帰処理 ” を記述する必要があります。

備考　　アラーム・コールバック内でインプレサイス例外モードの浮動小数点演算を使用する場合で，RV850 の処理に移行する前に演算結果を確定させる必要がある場合は，アラーム・コールバック終了直前に syncp 命令と synce 命令を発行してください。

(3) スタックの切り替え
RV850 では，アラーム・コールバックに制御を移す際，システム・スタック・サイズ “OsStackSize” で定義されたシステム・スタックへと切り替えます。
したがって，“ スタックの切り替え処理 ” を記述する必要がありません。

(4) 割り込みの受け付け
RV850 では，アラーム・コールバックに制御を移す際，カテゴリ 2 割り込みの受け付けを禁止状態（PMR の PM*n* ビットに対する操作）へと変更しています。

(5) システム・サービスの発行
“ アラーム・コールバックから発行可能なシステム・サービス ” のみが発行可能となります。

備考　　各システム・サービスの発行有効範囲についての詳細は，「14.4 システム・サービス・リファレンス」を参照してください。

8.2.2 アラーム・コールバックの登録

RV850 では，アラーム・コールバックの登録を “ アラーム・コールバック識別子 “OsAlarmCallbackName” の定義 ” による静的な登録に限定しています。したがって，処理プログラムからシステム・サービスを発行するなどの動的な生成はできません。

(1) 静的な登録
アラーム・コールバックの静的な登録は，アラーム・コールバック識別子 “OsAlarmCallbackName” を定義することにより実現されます。
なお，RV850 では，カーネル初期化部において，アラーム・コールバック識別子 “OsAlarmCallbackName” の定義内容を情報ファイルから読み出し，該当初期化処理を行い，管理対象としています。

8.3 アラームの生成

RV850 では，アラームの生成を “ アラーム情報の定義 ” による静的な生成に限定しています。したがって，処理プログラムからシステム・サービスを発行するなどの動的な生成はできません。

(1) 静的な生成
アラームの静的な生成は，CF ファイルにアラーム情報を定義することにより実現されます。
なお，RV850 では，カーネル初期化部において，アラーム情報の定義内容を情報ファイルから読み出し，該当初期化処理を行い，管理対象としています。

8.4 システム・サービス

処理プログラムからアラームに関する動的な操作は，「14.4.6 アラーム管理」で示したシステム・サービスを用いて行います。

9. スケジュール・テーブル管理

本章では，RV850 が提供しているスケジュール・テーブル管理について解説しています。

9.1 概　　要

RV850 では，カウンタのもつカウント値の変化に同期した処理を行うための機能として “ スケジュール・テーブル管理機能 ” を提供しています。
なお，スケジュール・テーブル管理機能では，カウンタ識別子 “OsScheduleTableCounterRef” において関連付けを行ったカウンタのカウント値が満了条件（満了カウント値）を満足した際には，タスクの起動，イベント・マスクの設定といった満了処理を実行します。

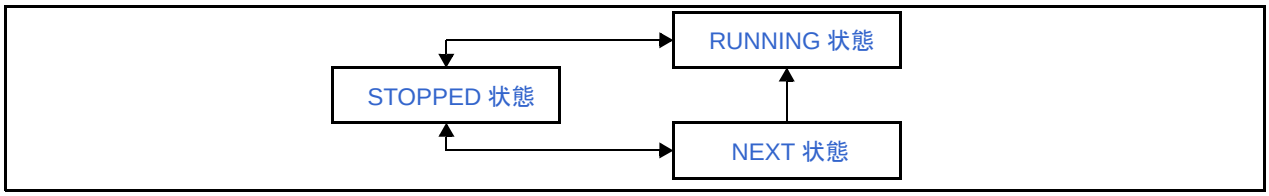
9.2 スケジュール・テーブル

アラームでは，1 個の満了カウント値，1 個の満了処理しか定義できないのに対し，スケジュール・テーブルでは，複数の満了カウント値，複数の満了処理を定義することができます。

9.2.1 状　　態

RV850 では，スケジュール・テーブルが取り得る状態を 3 種類に大別し，管理しています。

図 9.1　　スケジュール・テーブルの状態



(1) STOPPED 状態
RV850 の管理下にありながらもスケジュール・カウント対象からは除外されている状態です。

(2) NEXT 状態
NextScheduleTable の発行に伴い，引数 *ScheduleTableID_To* で指定されたスケジュール・テーブルが STOPPED 状態から遷移する状態です。
なお，NEXT 状態から RUNNING 状態への遷移は，RUNNING 状態のスケジュール・テーブル（引数 *ScheduleTableID_From* で指定されたスケジュール・テーブル）がスケジュール・カウントを完了したタイミングとなります。

(3) RUNNING 状態
スケジュール・カウントを実行中の状態です。

9.2.2 スケジュール・テーブルの生成

RV850 では，スケジュール・テーブルの生成を “ スケジュール・テーブル情報の定義 ” による静的な生成に限定しています。したがって，処理プログラムからシステム・サービスを発行するなどの動的な生成はできません。

(1) 静的な生成
スケジュール・テーブルの静的な生成は，CF ファイルにスケジュール・テーブル情報を定義することにより実現されます。
なお，RV850 では，カーネル初期化部において，スケジュール・テーブル情報の定義内容を情報ファイルから読み出し，該当初期化処理を行い，管理対象としています。

9.3 システム・サービス

処理プログラムからスケジュール・テーブルに関する動的な操作は，「14.4.7 スケジュール・テーブル管理」で示したシステム・サービスを用いて行います。

10. OS アプリケーション管理

本章では，RV850 が提供している OS アプリケーション管理（SC3 限定）について解説しています。

10.1 概　　要

RV850 では，システムで使用する資源（タスク，カウンタなど）をグルーピングし，アクセス保護（システム・サービスの発行に伴う資源の操作を禁止／許可）するためのメカニズムとして “OS アプリケーション管理機能 ” を提供しています。

なお，資源のグルーピングは，OS アプリケーションを単位に行い，同一 OS アプリケーションに所属している資源については，無条件にアクセスを許可，他 OS アプリケーションに所属している資源については，OS アプリケーション識別子 “OsTaskAccessingApplication”，OS アプリケーション識別子 “OsCounterAccessingApplication” などで個別にアクセス許可を与えることでアクセス保護を実現しています。

10.1.1 信 頼 性

RV850 では，OS アプリケーションを以下に示した 2 種類に分類しています。

(1) Trusted な OS アプリケーション
RV850 では，信頼性 “OsTrusted” で TRUE の定義された OS アプリケーションを Trusted な OS アプリケーション（信頼性が確保された OS アプリケーション）として扱います。
なお，RV850 では，Trusted な OS アプリケーションに所属している処理プログラムについては，信頼性が確保されているため，スーパバイザ・モード（PSW の UM ビットに 0 を設定），およびシステム・プロテクション番号（MCFG0 レジスタの SPID ビット）をシステム SPID 値 “OsSystemSystemSPID” で指定した値に設定して動作させ，メモリ・アクセス保護機能，および周辺 I/O 保護機能を非適用としています。

備考　AUTOSAR 仕様では，Trusted な OS アプリケーションにおいて，Trusted な OS アプリケーションの保護 “OsTrustedApplicationWithProtection” による保護機能の適用有無設定が可能である旨の規定が行われていますが，RV850 では，Trusted な OS アプリケーションに対する保護機能の適用はできません。

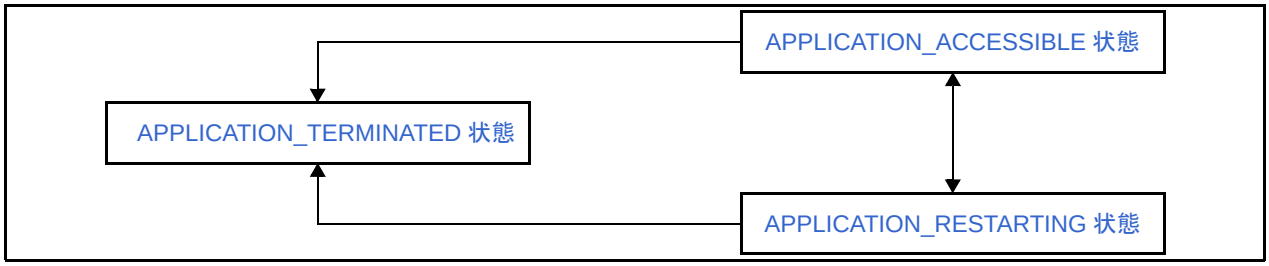
(2) Non-Trusted な OS アプリケーション
RV850 では，信頼性 “OsTrusted” で FALSE の定義された OS アプリケーションを Non-Trusted な OS アプリケーション（信頼性が確保されていない OS アプリケーション）として扱います。
なお，RV850 では，Non-Trusted な OS アプリケーションに所属している処理プログラムについては，信頼性が確保されていないため，ユーザ・モード（PSW の UM ビットに 1 を設定），および SPID“OsApplicationSPID” で指定したシステム・プロテクション番号（MCFG0 レジスタの SPID ビット）で動作させ，メモリ・アクセス保護機能，および周辺 I/O 保護機能を適用可能としています。

10.1.2 状　　態

OS アプリケーションは，処理プログラムからシステム・サービス（StartOS，TerminateApplication，AllowAccess）を発行する，または共通フック・ルーチン（ProtectionHook）の戻り値に設定された値（APPLICATION_TERMINATED，APPLICATION_RESTARTING）により，様々な状態へと遷移していきます。

なお，RV850 では，OS アプリケーションが取り得る状態を 3 種類に大別し，管理しています。

図 10.1　OS アプリケーションの状態



(1) APPLICATION_ACCESSIBLE 状態
ブート処理から StartOS が発行された際，CF ファイルに定義された全 OS アプリケーションが最初に遷移する状態です。

また，処理プログラムから AllowAccess が発行された際にも，AllowAccess を発行した処理プログラムの所属している OS アプリケーションが APPLICATION_RESTARTING 状態から本状態へと遷移します。

(2) APPLICATION_RESTARTING 状態
処理プログラムから TerminateApplication（再起動オプション：RESTART）が発行された際，APPLICATION_ACCESIBLE 状態の対象 OS アプリケーションが本状態へと遷移します。
また，共通フック・ルーチン（ProtectionHook）の戻り値に APPLICATION_RESTARTING が設定された際にも，ProtectionHook の呼び出し元の処理プログラム（保護違反 “ スタック・オーバーフロー，不正なメモリ・アクセス，例外の発生 ” の発生した処理プログラム）の所属している OS アプリケーションが本状態へと遷移します。

(3) APPLICATION_TERMINATED 状態
処理プログラムから TerminateApplication（再起動オプション：NO_RESTART）が発行された際，APPLICATION_ACCESIBLE 状態の対象 OS アプリケーションが本状態へと遷移します。
また，共通フック・ルーチン（ProtectionHook）の戻り値に APPLICATION_TERMINATED が設定された際にも，ProtectionHook の呼び出し元の処理プログラム（保護違反 “ スタック・オーバーフロー，不正なメモリ・アクセス，例外の発生 ” の発生した処理プログラム）の所属している OS アプリケーションが本状態へと遷移します。

10.1.3 メモリ・アクセス保護機能

RV850 では，Non-Trusted な OS アプリケーションに所属している処理プログラム（タスク，割り込みサービス・ルーチン，フック・ルーチン）が不正なメモリ・アクセスを行うことを抑制するための機能として，メモリ・アクセス保護機能を提供しています。

これにより，該当処理プログラムが不正なメモリ・アクセスを行った際には，ProtectionHook“OsProtectionHook” の定義内容にしたがった処理（共通フック・ルーチン ProtectionHook の呼び出し，または ShutdownOS の発行）が行われます。

以下に，RV850 がメモリ・アクセス保護機能で監視対象としているメモリ領域を示します。

なお，監視対象となるメモリ領域についてはメモリ領域識別子 “OsSystemMemoryArea” で，メモリ領域の種類（OS アプリケーション固有，システム共通）についてはメモリ領域識別子 “OsAppMemoryAreaNameRef”，メモリ領域識別子 “OsMemoryAreaNameRef” で定義し，未定義のメモリ領域はアクセス禁止なメモリ領域として扱われます。

表 10.1 監視対象としているメモリ領域

メモリ領域の種類		アクセスの種類
コード領域	OS アプリケーション固有	リード／実行
	システム共通	
データ領域	OS アプリケーション固有	リード／ライト
	システム共通	
スタック領域	OS アプリケーション固有	リード／ライト
	拡張タスク固有	
I/O 領域		リード／ライト

備考 1. スタック領域については，メモリ領域識別子 “OsSystemMemoryArea”，メモリ領域識別子 “OsAppMemoryAreaNameRef”，メモリ領域識別子 “OsMemoryAreaNameRef” での定義が不要です。また，" スタック・モニタリング機能 “OsStackMonitoring” の指定に関わらず，Non-Trusted な OS アプリケーションが使用するスタック領域の監視を常時行います。

備考 2. スタック領域のオーバーフローを検出するために考慮するスタック・ポインタの閾値 (アドレス) は，タスクや割り込みサービス・ルーチン（カテゴリ 2）が使用中のスタック領域のトップ・アドレスに対し，RV850 のシステム・サービスが使用するスタック使用量の最大値を加味したアドレスです。これは，Non-Trusted な OS アプリケーションであっても，RV850 のシステム・サービス実行中はスーパバイザ・モードへの移行により，メモリ・アクセス保護機能が動作しないことに対する対策です。

10.1.4 周辺 I/O 保護機能

RV850 では，Non-Trusted な OS アプリケーションに所属している処理プログラム（タスク，割り込みサービス・ルーチン，フック・ルーチン）が不正な周辺 I/O アクセスを行うことを抑制するための機能として，マシン・コンフィ

ギュレーション（MCFG0）の SPID ビット（システム・プロテクション番号）を利用した周辺 I/O 保護機能を提供しています。

なお，RV850 では，OS アプリケーションの切り替えが発生した際，SPID ビットの内容を SPID“OsApplicationSPID” で定義された値に書き換えることにより，周辺 I/O 保護機能を実現しています。

備考 1. RV850 が周辺 I/O 保護機能で監視対象とする I/O 領域については，メモリ領域識別子 “OsAppMemoryAreaNameRef” の定義を行い，アクセス許可なメモリ領域とする必要があります。

備考 2. SPID“OsApplicationSPID” に定義する値は，Non-Trusted な OS アプリケーションに 1 対 1 である必要はありません。
したがって，複数の Non-Trusted な OS アプリケーションに同一の SPID“OsApplicationSPID” を定義することが可能です。

10.2 Trusted 関数

Trusted な OS アプリケーションについては，OS アプリケーション単位に固有な Trusted 関数を所属させることができます。

Trusted 関数は，処理プログラムから CallTrustedFunction および CallTrustedFunctionRestricted を発行することにより呼び出されます。CallTrustedFunction および CallTrustedFunctionRestricted は他の Non-Trusted な OS アプリケーションからも呼び出すことが可能なので，Non-Trusted な OS アプリケーションから一時的に保護機能非適用の処理をさせたい場合に Trusted 関数を使用することができます。

以下に，Trusted 関数を C 言語で記述する際の基本型を示します。

```
TRUSTED ( OsTrustedFunctionName ) {
      .....
      .....
}
```

備考 1. OsTrustedFunctionName には，識別子 “OsTrustedFunctionName” で定義した Trusted 関数の識別子を記述します。

備考 2. Trusted 関数がアクセス可能なオブジェクト（タスク，リソースなど）は，Trusted 関数が所属している OS アプリケーションと同様です。

備考 3. Trusted 関数はスーパバイザ・モードで動作するため，CallTrustedFunction および CallTrustedFunctionRestricted を Non-Trusted な OS アプリケーションに所属している処理プログラムから発行した場合，モード切り替え処理（ユーザ・モードからスーパバイザ・モードへと移行），ならびにシステム・プロテクション番号（MCFG0 レジスタの SPID ビット）の切り替え（Trusted 関数実行中はシステム SPID 値 “OsSystemSystemSPID” で指定した値）が行われます。

備考 4. Trusted 関数はスーパバイザ・モードで動作するため，メモリ・アクセス保護機能は適用外となります。また，システム・プロテクション番号（MCFG0 レジスタの SPID ビット）がシステム SPID 値 “OsSystemSystemSPID” に指定した値で動作するため，システム・プロテクション番号と連動した周辺 I/O 保護機能は適用外となります。

備考 5. AUTOSAR 仕様では，Trusted 関数から同一 OS アプリケーションの他タスクへのディスパッチは行わない旨の規定が行われていますが，RV850 では，他タスクへのディスパッチを行います。

備考 6. AUTOSAR 仕様では，CallTrustedFunction を発行した処理プログラムから Trusted 関数へと制御を移す場合，カテゴリ 2 割り込みの受け付けを禁止する旨の規定が行われていますが，RV850 では，割り込みの受け付けに関する操作を行いません。
ただし，CallTrustedFunctionRestricted を発行した処理プログラムから Trusted 関数へと制御を移す場合は，RV850 は割り込みの受け付け禁止処理（PSW の ID ビットに対する操作）を行います。

備考 7. CallTrustedFunctionRestricted の発行によって機能制限付きで呼び出される Trusted 関数では，割り込みの受け付け許可処理（PSW の ID ビットに対する操作）を行うことを禁止します。割り込みの受け付け許可処理を行った場合，その後の動作を保証しません。

備考 8. CallTrustedFunctionRestricted の発行によって機能制限付きで呼び出される Trusted 関数では，システム・サービスおよび RV850 が提供しているユーティリティ関数の発行を禁止します。システム・サービスおよび RV850 が提供しているユーティリティ関数を発行した場合，その後の動作を保証しません。

10.2.1 Trusted 関数の処理

RV850 では，各種処理プログラムから Trusted 関数に制御を移す際，独自の前処理を行っています。また，Trusted 関数から各種処理プログラムに制御を戻す際にも，独自の後処理を行っています。このため，Trusted 関数を記述する際には，以下に示す注意点があります。

(1) レジスタの退避／復帰
RV850 では，Trusted 関数に制御を移す際，C コンパイラの関数呼び出し規約に従った作業用レジスタの退避処理／復帰処理を行います。
したがって，“ レジスタの退避処理／復帰処理 ” を記述する必要がありません。

(2) FPSR の退避／復帰
FPSR“OsAppDefaultFPSRValue”，または FPSR のデフォルト値 “OsDefaultFPSRValue” で定義された値へと変更します。
したがって，FPSR の退避／復帰 “OsSaveFpuReg” で TRUE を定義している場合，“FPSR の退避処理／復帰処理 ” は，記述する必要がありません。

備考 1. “FPSR の退避処理／復帰処理 ” は，“FPU 対応あり ” のカーネル・ライブラリ libecc2extsc1_fpu.a，libecc2extsc3_fpu.a をリンクした場合に限り行われます。

備考 2. Trusted 関数内でインプレサイス例外モードの浮動小数点演算を使用する場合で，RV850 による FPSR の復帰処理前に演算結果を確定させる必要がある場合は，Trusted 関数終了直前に syncp 命令と sync 命令を発行してください。

備考 3. CallTrustedFunctionRestricted の発行により Trusted 関数を呼び出す場合は，“FPSR の退避処理／復帰処理 ” は行いません。

(3) スタックの切り替え
RV850 では，Trusted 関数を CallTrustedFunction を発行した処理プログラムの延長線として位置付けているため，スタックの切り替えを行いません。
したがって，CallTrustedFunction を発行する処理プログラムでは，該当処理プログラムのスタック・サイズを見積もる際，Trusted 関数が必要とするサイズを考慮する必要があります。

(4) 割り込みの受け付け
RV850 では，Trusted 関数に制御を移す際，割り込みの受け付けに関する操作を行いません。
したがって，割り込みの受け付けを明示的に変更する場合，“EnableAllInterrupts，DisableAllInterrupts などの発行処理 ” を記述する必要があります。
ただし，CallTrustedFunctionRestricted を発行した処理プログラムから Trusted 関数に制御を移す場合は，RV850 は割り込みの受け付け禁止処理（PSW の ID ビットに対する操作）を行います。

(5) システム・サービスの発行
RV850 では，Trusted 関数を CallTrustedFunction を発行した処理プログラムの延長線として位置付けているため，CallTrustedFunction の発行により呼び出された Trusted 関数で発行可能なシステム・サービスは，CallTrustedFunction を発行した処理プログラムの種類に依存します。
ただし，CallTrustedFunctionRestricted の発行により呼び出された Trusted 関数では，システム・サービスおよび RV850 が提供しているユーティリティ関数の発行を禁止します。システム・サービスおよび RV850 が提供しているユーティリティ関数を発行した場合，その後の動作を保証しません。

備考 各システム・サービスの発行有効範囲についての詳細は，「14.4 システム・サービス・リファレンス」を参照してください。

10.2.2 Trusted 関数の登録

RV850 では，Trusted 関数の登録を “ 識別子 “OsTrustedFunctionName” の定義 ” による静的な登録に限定しています。したがって，処理プログラムからシステム・サービスを発行するなどの動的な生成はできません。

(1) 静的な登録
Trusted 関数の静的な登録は，識別子 “OsTrustedFunctionName” を定義することにより実現されます。
なお，RV850 では，カーネル初期化部において，識別子 “OsTrustedFunctionName” の定義内容を情報ファイルから読み出し，該当初期化処理を行い，管理対象としています。

10.2.3 Trusted 関数の引き継ぎデータ

本節では，Trusted 関数の呼び出し元と Trusted 関数本体との間のデータの引き継ぎ方法について説明します。Trusted 関数「TFUNC1」を以下のように基本形式に従って記述したとします。

```
TRUSTED (TFUNC1) {
    ……
    return;
}
```

この場合，コンパイル時に下記の形式にマクロ展開されます。

```
void
TRUSTED_TFUNC1(TrustedFunctionIndexType FunctionIndex,
               TrustedFunctionParameterRefType FunctionParams) {
    ……
    return;
}
```

Trusted 関数「TFUNC1」の本体からは CallTrustedFunction システム・サービスで指定した第 1 引数と第 2 引数が参照可能となります。

- 第 1 引数 FunctionIndex
  TrustedFunctionIndexType 型の FunctionIndex という引数の形式で参照可能
- 第 2 引数 FunctionParams が指し示す引き継ぎデータ
  TrustedFunctionParameterRefType 型の FunctionParams という引数のポインタを参照する形式で参照可能

以下に，タスク TASK1 から，Trusted 関数 TFUNC1 を呼び出す際の引き継ぎデータの参照の例を示します。

```
/* Trusted 関数 TFUNC1 の引き継ぎデータ構造体 */
struct TFUNC1_parameter_struct {
    int param1;/* 入力データ 1 */
    int param2;/* 入力データ 2 */
    int ret_code; /* 出力データ */
};

/* Trusted 関数 TFUNC1 の引継ぎデータ領域の確保 */
struct TFUNC1_parameter_struct Tfunc1_paramaters;

/* タスク */
TASK(TASK1) {
    ……
    ……
     /* Trusted 関数への入力データ */
    local_struct.param1 = 1;
    local_struct.param2 = 2;

    /* Trusted 関数呼び出し */
    CallTrustedFunction(TFUNC1,\\
                        (TrustedFunctionParameterRefType)&Tfunc1_paramaters);
    /* Trusted 関数からの出力データ取得 */
    ret_code = Tfunc1_paramaters.value_return;
    ……
    ……
    TerminateTask();
}

/* Trusted 関数 */
TRUSTED (TFUNC1) {
    int p1, p2;

    /* 入力データ取得 */
    p1 = FunctionParams->param1;
    p2 = FunctionParams->param2;
    ……
    ……
    /* 呼び出し元への出力データ */
    FunctionParams->value_return = 0x100;

    return;
}
```

10.3 固有フック・ルーチン

OS アプリケーション単位に固有なフック・ルーチンとして，用途別に 3 種類のフック・ルーチンをサポートしています。

備考 Non-Trusted な OS アプリケーションの固有フック・ルーチンは，ユーザ・モード（PSW の UM ビットに 1 を設定），ならびに SPID“OsApplicationSPID” で指定したシステム・プロテクション番号（MCFG0 レジスタの SPID ビット）で動作し，メモリ・アクセス保護機能，およびシステム・プロテクション番号と連動した周辺 I/O 保護機能を適用可能としています。
Trusted な OS アプリケーションの固有フック・ルーチンは，スーパバイザ・モード（PSW の UM ビットに 0 を設定），ならびにシステム・プロテクション番号（MCFG0 レジスタの SPID ビット）がシステム SPID 値 “OsSystemSystemSPID” に指定した値で動作し，メモリ・アクセス保護機能，およびシステム・プロテクション番号と連動した周辺 I/O 保護機能は適用外となります。

(1) StartupHook_*OsApplication*
処理プログラムから StartOS が発行された際に呼び出される初期化処理専用フック・ルーチンです。
以下に，StartupHook_*OsApplication* を C 言語で記述する際の基本型を示します。

```
void
StartupHook_OsApplication ( void ) {
    ……
    ……
}
```

備考 1. *OsApplication* には，識別子 “OsApplication” で定義した OS アプリケーションの識別子を記述します。

備考 2. StartupHook_OsApplication“OsAppStartupHook” で TRUE の指定が行われている OS アプリケーション情報が複数存在する場合，StartupHook_*OsApplication* の呼び出しは，CF ファイルでの記述順に行われます。

(2) ShutdownHook_*OsApplication*
処理プログラムから ShutdownOS が発行された際に呼び出される終了処理専用フック・ルーチンです。
以下に，ShutdownHook_*OsApplication* を C 言語で記述する際の基本型を示します。

```
void
ShutdownHook_OsApplication ( StatusType Fatalerror ) {
    ……
    ……
}
```

備考 1. *OsApplication* には，識別子 “OsApplication” で定義した OS アプリケーションの識別子を記述します。

備考 2. 引数 *Fatalerror* には，ShutdownOS の引数 *Error* に指定された引き継ぎデータが設定されます。なお，本フック・ルーチンを呼び出した処理プログラムがデフォルトの例外／割り込み安全対策処理であった場合，引数 *Fatalerror* には，E_OS_SYS_ILLEGAL_EXCEPTION が設定されます。

備考 3. 引数 *Fatalerror* が E_OS_SYS_ILLEGAL_EXCEPTION の際には，本フック・ルーチン内で OSIllegalException_SystemRegister_ExcCode を発行することによりレジスタ値（EIIC，または FEIC）を，OSIllegalException_SystemRegister_ExcPC を発行することにより，レジスタ値（EIPC，または FEPC）を獲得することができます。

備考 4. ShutdownHook_OsApplication“OsAppShutdownHook” で TRUE の指定が行われている OS アプリケーション情報が複数存在する場合，ShutdownHook_*OsApplication* の呼び出しは，CF ファイルでの記述順に行われます。

(3) ErrorHook_*OsApplication*
処理プログラムから発行したシステム・サービスが異常終了した際に呼び出されるエラー処理専用フック・ルーチンです。
以下に，ErrorHook_*OsApplication* を C 言語で記述する際の基本型を示します。

```
void
ErrorHook_OsApplication ( StatusType Error ) {
    ……
    ……
```

備考 1. *OsApplication* には，識別子 “OsApplication” で定義した OS アプリケーションの識別子を記述します。

備考 2. 引数 *Error* には，異常終了したシステム・サービスのエラー・ステータスが設定されます。

備考 3. 本フック・ルーチンは，システム・サービスが異常終了した処理プログラムが所属している OS アプリケーション固有のものとなります。

備考 4. ErrorHook_*OsApplication* から発行したシステム・サービスが異常終了した際，ErrorHook，および ErrorHook_*OsApplication* の再呼び出しは行われません。

10.3.1 固有フック・ルーチンでの処理

RV850 では，各種処理プログラムから固有フック・ルーチンに制御を移す際，独自の前処理を行っています。また，固有フック・ルーチンから各種処理プログラムに制御を戻す際にも，独自の後処理を行っています。このため，固有フック・ルーチンを記述する際には，以下に示す注意点があります。

(1) レジスタの退避／復帰
RV850 では，固有フック・ルーチンに制御を移す際，C コンパイラの関数呼び出し規約に従った作業用レジスタの退避処理／復帰処理を行います。
したがって，“ レジスタの退避処理／復帰処理 ” を記述する必要がありません。

(2) FPSR の退避／復帰
RV850 では，固有フック・ルーチンに制御を移す際，浮動小数点設定／状態レジスタ（FPSR）の退避／復帰に関する操作を行いません。
したがって，FPSR の内容を明示的に変更する場合は，“FPSR の退避処理／復帰処理 ” を記述する必要があります。

備考 固有フック・ルーチン内でインプレサイス例外モードの浮動小数点演算を使用する場合で，RV850 の処理に移行する前に演算結果を確定させる必要がある場合は，固有フック・ルーチン終了直前に syncp 命令と synce 命令を発行してください。

(3) スタックの切り替え
RV850 では，固有フック・ルーチンに制御を移す際，OS アプリケーション・スタック・サイズ “OsAppStackSize” で定義された OS アプリケーション・スタックへと切り替えます。
したがって，“ スタックの切り替え処理 ” を記述する必要がありません。

(4) 割り込みの受け付け
RV850 では，固有フック・ルーチンに制御を移す際，カテゴリ 2 割り込みの受け付けを禁止状態（PMR の PM*n* ビットに対する操作）へと変更しています。

備考 1. 固有フック・ルーチンでカテゴリ 2 割り込みの受け付けを明示的に操作することは禁止されています。

備考 2. RV850 では，ShutdownHook_*OsApplication* に制御を移す際，割り込みの受け付け禁止処理として，PMR の PM*n* ビットに対する操作の他，PSW の ID ビットに対するする操作も行っています。

(5) システム・サービスの発行
“ 固有フック・ルーチンから発行可能なシステム・サービス ” のみが発行可能となります。

備考 各システム・サービスの発行有効範囲についての詳細は，「14.4 システム・サービス・リファレンス」を参照してください。

10.3.2 固有フック・ルーチンの登録

RV850 では，固有フック・ルーチンの登録を以下の定義による静的な登録に限定しています。したがって，処理プログラムからシステム・サービスを発行するなどの動的な生成はできません。

- StartupHook_OsApplication“OsAppStartupHook”
- ShutdownHook_OsApplication“OsAppShutdownHook”
- ErrorHook_OsApplication“OsAppErrorHook”

(1) 静的な登録
固有フック・ルーチンの静的な登録は，以下の定義で TRUE を定義することにより実現されます。

- StartupHook_OsApplication“OsAppStartupHook”
- ShutdownHook_OsApplication“OsAppShutdownHook”
- ErrorHook_OsApplication“OsAppErrorHook”

なお，RV850 では，カーネル初期化部において，定義内容を情報ファイルから読み出し，該当初期化処理を行い，管理対象としています。

10.4 OS アプリケーションの生成

RV850 では，OS アプリケーションの生成を “OS アプリケーション情報の定義 ” による静的な生成に限定しています。したがって，処理プログラムからシステム・サービスを発行するなどの動的な生成はできません。

(1) 静的な生成
OS アプリケーションの静的な生成は，CF ファイルに OS アプリケーション情報を定義することにより実現されます。
なお，RV850 では，カーネル初期化部において，OS アプリケーション情報の定義内容を情報ファイルから読み出し，該当初期化処理を行い，管理対象としています。

10.5 システム・サービス

処理プログラムから OS アプリケーションに関する動的な操作は，「14.4.8 OS アプリケーション管理」で示したシステム・サービスを用いて行います。

11. OS 実行管理

本章では，RV850 が提供している OS 実行管理について解説しています。

11.1 概　　要

RV850 では，RV850 の起動／終了といった処理を実現するためのメカニズムとして “OS 実行管理機能 ” を提供しています。

11.2 共通フック・ルーチン

RV850 では，共通なフック・ルーチンとして，用途別に 6 種類のフック・ルーチンをサポートしています。

備考　　共通フック・ルーチンは，スーパバイザ・モードで動作するため，メモリ・アクセス保護機能は適用外となります。また，システム・プロテクション番号（MCFG0 レジスタの SPID ビット）がシステム SPID 値 “OsSystemSystemSPID” に指定した値で動作するため，システム・プロテクション番号と連動した周辺 I/O 保護機能は適用外となります。

(1) StartupHook

処理プログラムから StartOS が発行された際に呼び出される初期化処理専用フック・ルーチンです。
以下に，StartupHook を C 言語で記述する際の基本型を示します。

```
void
StartupHook ( void ) {
    ……
    ……
}
```

(2) ShutdownHook

ShutdownOS が発行された際に呼び出される終了処理専用フック・ルーチンです。
以下に，ShutdownHook を C 言語で記述する際の基本型を示します。

```
void
ShutdownHook ( StatusType Fatalerror ) {
    ……
    ……
}
```

備考 1.　　引数 *Fatalerror* には，ShutdownOS の引数 *Error* に指定された引き継ぎデータが設定されます。なお，本フック・ルーチンを呼び出した処理プログラムがデフォルトの例外／割り込み安全対策処理であった場合，引数 *Fatalerror* には，E_OS_ILLEGAL_EXCEPTION が設定されます。

備考 2.　　引数 *Fatalerror* が E_OS_SYS_ILLEGAL_EXCEPTION の際には，本フック・ルーチン内で OSIllegalException_SystemRegister_ExcCode を発行することによりレジスタ値（EIIC，または FEIC）を，OSIllegalException_SystemRegister_ExcPC を発行することによりレジスタ値（EIPC，または FEPC）を獲得することができます。

(3) PostTaskHook

スケジューラから呼び出されるスケジューリング前処理専用フック・ルーチンです。
以下に，PostTaskHook を C 言語で記述する際の基本型を示します。

```
void
PostTaskHook ( void ) {
    ……
    ……
}
```

(4) PreTaskHook

スケジューラから呼び出されるスケジューリング後処理専用フック・ルーチンです。
以下に，PreTaskHook を C 言語で記述する際の基本型を示します。

```
void
PreTaskHook ( void ) {
    ……
    ……
}
```

(5) ErrorHook

処理プログラムから発行したシステム・サービスが異常終了した際に呼び出されるエラー処理専用フック・ルーチンです。
以下に，ErrorHook を C 言語で記述する際の基本型を示します。

```
void
ErrorHook ( StatusType Error ) {
    ……
    ……
}
```

備考 1. 引数 *Error* には，異常終了したシステム・サービスのエラー・ステータスが設定されます。
エラー・ステータスについての詳細は，「14.2.2 エラー・ステータス」を参照してください。

備考 2. ErrorHook から発行したシステム・サービスが異常終了した際，ErrorHook，および ErrorHook_*OsApplication* の再呼び出しは行われません。

(6) ProtectionHook

RV850 が保護違反（スタック・オーバーフロー，不正なメモリ・アクセス，例外の発生）を検出した際に呼び出される保護違反対応処理専用フック・ルーチンです。
以下に，ProtectionHook を C 言語で記述する際の基本型を示します。

```
ProtectionReturnType
ProtectionHook ( StatusType Fatalerror, void *adr ) {
    ……
    ……
    return (ProtectionReturnType ReturnType );
}
```

備考 1. 引数 *Fatalerror* には保護違反の種類が設定されます。
以下に，*Fatalerror* に設定される値を示します。

【E_OS_STACKFAULT（0x16）】
スタックのオーバーフローを検出した

【E_OS_PROTECTION_MEMORY（0x17）】
不正なメモリ・アクセスを検出した

備考 2. 引数 *adr* で指定された領域には，引数 *Fatalerror* が E_OS_STACKFAULT の場合は “0” が，E_OS_PROTECTION_MEMORY の場合は “ 保護違反を検出した時点における FE レベル割り込み受け付け時の状態退避レジスタ（FEPC）の値 ” が設定されます。

備考 3. 戻り値 *ReturnType* には，本共通フック・ルーチンの処理完了後に行う RV850 の処理を指定します。
以下に，*ReturnType* に設定可能な値を示します。

【PRO_TERMINATETASKISR（0x1）】
保護違反を行った処理プログラムの種類により，処理内容は以下のように異なります。

- タスクの場合
  保護違反が発生した箇所でタスクの実行を強制的に終了し，SUSPENDED 状態へと遷移
  リソースを獲得していた際には，該当リソースを解放
  DisableAllInterrupts を発行していた際には，EnableAllInterrupts を発行
  SuspendAllInterrupts を発行していた際には，ResumeAllInterrupts を発行
  SuspendOSInterrupts を発行していた際には，ResumeOSInterrupts を発行
  スケジューラを起動
- 割り込み・サービス・ルーチンの場合
  リソースを獲得していた際には，該当リソースを解放
  DisableAllInterrupts を発行していた際には，EnableAllInterrupts を発行
  SuspendAllInterrupts を発行していた際には，ResumeAllInterrupts を発行
  SuspendOSInterrupts を発行していた際には，ResumeOSInterrupts を発行
  保護違反が発生した箇所で割り込みサービス・ルーチンの実行を強制的に終了し，スケジューラを起動
- 上記以外の場合
  PRO_TERMINATEAPPL と同様の処理

【PRO_TERMINATEAPPL（0x2）】
保護違反を行った処理プログラムが所属している OS アプリケーションに対し，
TerminateApplication（再起動オプション：NO_RESTART）相当の処理を実行
なお，APPLICATION_ACCESSIBLE 状態の OS アプリケーションが存在しない際には，
ShutdownOS（引き継ぎデータ ：*Fatalerror*）を発行

【PRO_SHUTDOWN（0x4）】
ShutdownOS（引き継ぎデータ：*Fatalerror*）を発行

【PRO_TERMINATEAPPL_RESTART（0x12）】タスク識別子 “OsRestartTask”
保護違反を行った処理プログラムが所属している OS アプリケーションに対し，
TerminateApplication（再起動オプション：RESTART）相当の処理を実行
なお，APPLICATION_ACCESSIBLE 状態の OS アプリケーションが存在しない際，およびタスク識別子 “OsRestartTask” が未定義の際には，ShutdownOS（引き継ぎデータ ：*Fatalerror*）を発行

【その他】
ShutdownOS（引き継ぎデータ：*Fatalerror*）を発行

備考 4. AUTOSAR 仕様では，引数 *adr* の規定が行われていません。
本引数は，RV850 が独自に追加した引数です。

11.2.1 共通フック・ルーチンでの処理

RV850 では，各種処理プログラムから共通フック・ルーチンに制御を移す際，独自の前処理を行っています。また，共通フック・ルーチンから各種処理プログラムに制御を戻す際にも，独自の後処理を行っています。このため，共通フック・ルーチンを記述する際には，以下に示す注意点があります。

(1) レジスタの退避／復帰
RV850 では，共通フック・ルーチンに制御を移す際，C コンパイラの関数呼び出し規約に従った作業用レジスタの退避処理／復帰処理を行います。
したがって，“ レジスタの退避処理／復帰処理 ” を記述する必要がありません。

備考 RV850 では，ProtectionHook に制御を移す際，FE レベル例外用作業レジスタ（FEWR）を退避／復帰することなく使用します。
したがって，ProtectionHook に制御が移った際には，FEWR の値が不定値となります。

(2) FPSR の退避／復帰
RV850 では，共通フック・ルーチンに制御を移す際，浮動小数点設定／状態レジスタ（FPSR）の退避／復帰に関する操作を行いません。
したがって，FPSR の内容を明示的に変更する場合，“FPSR の退避処理／復帰処理 ” を記述する必要があります。

備考 共通フック・ルーチン内でインプレサイス例外モードの浮動小数点演算を使用する場合で，RV850 の処理に移行する前に演算結果を確定させる必要がある場合は，共通フック・ルーチン終了直前に syncp 命令と synce 命令を発行してください。

(3) スタックの切り替え
RV850 では，共通フック・ルーチンに制御を移す際，システム・スタック・サイズ “OsStackSize” で定義されたシステム・スタックへと切り替えます。
したがって，“ スタックの切り替え処理 ” を記述する必要がありません。

備考 スケーラビリティ・クラスが SC3 の場合，RV850 では，PostTaskHook，PreTaskHook，ErrorHook に制御を移す際，OS アプリケーション・スタック・サイズ “OsAppStackSize” で定義された OS アプリケーション・スタックへと切り替えます。

(4) 割り込みの受け付け
RV850 では，共通フック・ルーチンに制御を移す際，カテゴリ 2 割り込みの受け付けを禁止状態（PMR の PM*n* ビットに対する操作）へと変更しています。

備考 1. 共通フック・ルーチンでカテゴリ 2 割り込みの受け付けを明示的に操作することは禁止されています。

備考 2. RV850 では，ShutdownHook，ProtectionHook に制御を移す際，割り込みの受け付け禁止処理として，PMR の PM*n* ビットに対する操作の他，PSW の ID ビットに対するする操作も行っています。

(5) システム・サービスの発行
“ 共通フック・ルーチンから発行可能なシステム・サービス ” のみが発行可能となります。

備考 各システム・サービスの発行有効範囲についての詳細は，「14.4 システム・サービス・リファレンス」を参照してください。

11.2.2 共通フック・ルーチンの登録

RV850 では，共通フック・ルーチンの登録を以下の定義による静的な登録に限定しています。したがって，処理プログラムからシステム・サービスを発行するなどの動的な生成はできません。

- StartupHook“OsStartupHook”
- ShutdownHook“OsShutdownHook”
- PostTaskHook“OsPostTaskHook”
- PreTaskHook“OsPreTaskHook”
- ErrorHook“OsErrorHook”
- ProtectionHook“OsProtectionHook”（SC3 限定）

(1) 静的な登録
共通フック・ルーチンの静的な登録は，以下の定義で TRUE を定義することにより実現されます。

- StartupHook“OsStartupHook”
- ShutdownHook“OsShutdownHook”
- PostTaskHook“OsPostTaskHook”
- PreTaskHook“OsPreTaskHook”
- ErrorHook“OsErrorHook”
- ProtectionHook“OsProtectionHook”（SC3 限定）

なお，RV850 では，カーネル初期化部において，定義内容を情報ファイルから読み出し，該当初期化処理を行い，管理対象としています。

11.2.3 システム・サービス

処理プログラムから OS 実行に関する動的な操作は，「14.4.9 OS 実行管理」で示したシステム・サービスを用いて行います。

12. スケジュール管理

本章では，RV850 が提供しているスケジュール管理について解説しています。

12.1 概　　要

RV850 では，動的に変化していくタスクの状態を直接参照することにより，タスクの実行順序を管理／決定し，最適なタスクにデバイスの利用権を与えるためのメカニズムとして “ スケジュール管理機能 ” を提供しています。
なお，RV850 では，以下に示した 2 種類のスケジューリング方式をサポートしています。

(1) 非プリエンプティブ属性
処理を実行中のタスクが明示的にデバイスの使用権を放棄した際にスケジューラの起動が行われます。
以下に，“ 非プリエンプティブ属性 ” におけるスケジューラの起動条件を示します。

- TerminateTask の発行
- ChainTask の発行
- Schedule の発行
- WaitEvent の発行
- TerminateApplication の発行
- 保護例外（システム・エラー例外，メモリ・プロテクション例外，特権命令例外）の発生

備考　“WaitEvent の発行 ” によるスケジューラの起動は，要求条件を満足するイベント・マスクが対象イベントに設定されていなかった場合に限定されます。

(2) プリエンプティブ属性
何らかの事象（きっかけ）の発生に伴い，スケジューラの起動が行われます。
以下に，“ プリエンプティブ属性 ” におけるスケジューラの起動条件を示します。

- ActivateTask の発行
- TerminateTask の発行
- ChainTask の発行
- Schedule の発行
- ReleaseResource の発行
- SetEvent の発行
- WaitEvent の発行
- TerminateApplication の発行
- 保護例外（システム・エラー例外，メモリ・プロテクション例外，特権命令例外）の発生
- カテゴリ 2 の割り込みサービス・ルーチンからの復帰命令の発行
- アラーム／スケジュール・テーブルの満了

12.2 フック・ルーチン

RV850 では，スケジューリング処理の前処理／後処理として，フックルーチンの呼び出しを行っています。

(1) PostTaskHook
スケジューラから呼び出されるスケジューリング前処理専用フック・ルーチンです。

(2) PreTaskHook
スケジューラから呼び出されるスケジューリング後処理専用フック・ルーチンです。

備考　PostTaskHook，および PreTaskHook についての詳細は，「11.2 共通フック・ルーチン」を参照してください。

12.3 アイドル・ハンドラ

デバイスが提供している低消費電力対応機能を有効活用するために切り出されたアイドル処理専用ルーチンであり，RV850 のスケジューリング対象となるタスク（READY 状態，または RUNNING 状態のタスク）がシステム内に 1 つも存在しなくなった際にスケジューラから呼び出されます。

以下に，アイドル・ハンドラを C 言語で記述する際の基本型を示します。

```
void
IdleHandler ( IdleModeType idlemode ) {
    ……
    ……
}
```

備考 1. アイドル・ハンドラは，スーパバイザ・モードで動作するため，メモリ・アクセス保護機能は適用外となります。また，システム・プロテクション番号（MCFG0 レジスタの SPID ビット）がシステム SPID 値 “OsSystemSystemSPID” に指定した値で動作するため，システム・プロテクション番号と連動した周辺 I/O 保護機能は適用外となります。

備考 2. RV850 ではデフォルト・アイドル・ハンドラが用意されています。
したがって，IdleHandler の記述が行われていない場合には，デフォルト・アイドル・ハンドラ（空の無限ループ処理）が呼び出されます。

備考 3. 引数 *idlemode* には，アイドル・ハンドラ呼び出しの直前に発行された ControlIdle の引数 *IdleMode* に指定された引継ぎデータが設定されます。

備考 4. RV850 では，引数 *idlemode* に対応する処理内容については規定しません。引数 *idlemode* に対応する処理は，ユーザが記述する必要があります。

備考 5. StartOS を発行してから一度も ControlIdle を発行していない場合は，引数 *idlemode* には IDLE_NO_HALT が設定されます。

備考 6. アイドル・ハンドラの処理が終了した場合，「12.3.1 アイドル・ハンドラでの処理」に示す前処理は実行せずに，再びアイドル・ハンドラの先頭から処理を開始します。

備考 7. アイドル・ハンドラ処理中に割り込みサービス・ルーチン（カテゴリ 2）が呼び出された場合，アイドル・ハンドラの割り込み発生箇所に戻らず，「12.3.1 アイドル・ハンドラでの処理」に示す前処理を実行した後，再びアイドル・ハンドラの先頭から処理を開始します。

12.3.1 アイドル・ハンドラでの処理

RV850 では，アイドル・ハンドラに制御を移す際，独自の前処理を行っています。このため，アイドル・ハンドラを記述する際には，以下に示す注意点があります。

(1) レジスタの退避／復帰
RV850 では，アイドル・ハンドラに制御を移す際，C コンパイラの関数呼び出し規約に従った作業用レジスタの退避処理／復帰処理を行います。
したがって，“ レジスタの退避処理／復帰処理 ” を記述する必要がありません。

(2) FPSR の退避／復帰
RV850 では，アイドル・ハンドラに制御を移す際，浮動小数点設定／状態レジスタ（FPSR）の退避処理／復帰処理を行いません。
したがって，FPSR の内容を明示的に変更する場合，アイドル・ハンドラで “FPSR の退避処理／復帰処理 ” を記述する必要があります。

(3) スタックの切り替え
RV850 では，アイドル・ハンドラに制御を移す際，システム・スタック・サイズ “OsStackSize” で定義されたシステム・スタックへと切り替えます。
したがって，“ スタックの切り替え処理 ” を記述する必要がありません。

(4) 割り込みの受け付け
RV850 では，アイドル・ハンドラに制御を移す際，割り込みの受け付けを許可状態（PSW の ID ビットに対する操作）へと変更しています。

(5) システム・サービスの発行
システム・サービスの発行が禁止されています。

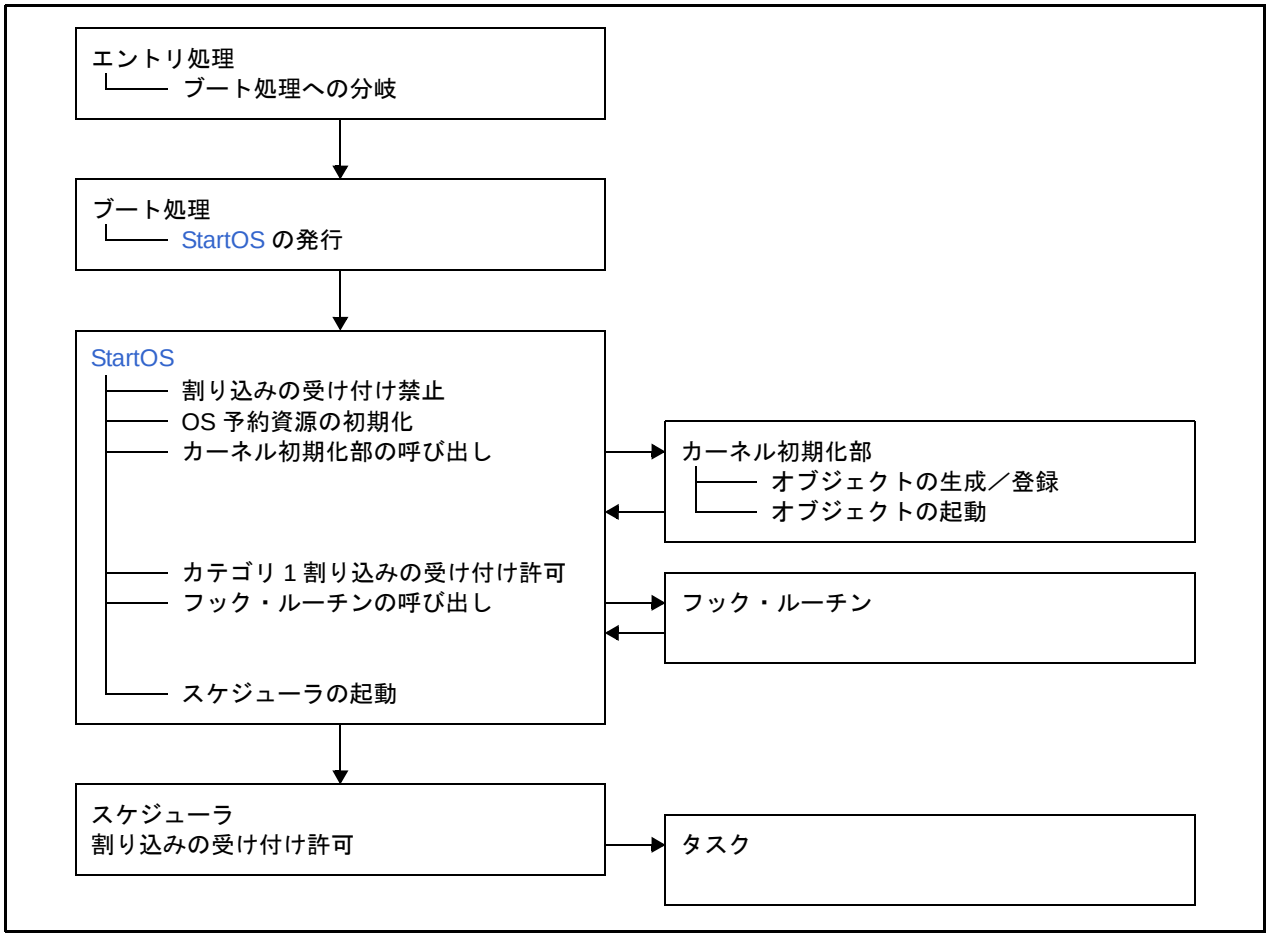
13. システム初期化処理

本章では，RV850 が提供しているシステム初期化処理について解説しています。

13.1 概　　要

RV850 では，ハードウエア・リセットの発生から処理プログラム（タスク）に制御を移すまでに必要となる“RV850 が処理を実行するうえで必要となるハードウエア／ソフトウエアの初期化処理 ” を提供しています。
以下に，ハードウエア・リセットの発生から処理プログラム（タスク）に制御が移るまでに実行される処理の流れを示します。

図 13.1　　システム初期化処理の流れ



備考　　RV850 では，StartOS でフック・ルーチンの呼び出しが行われる以前にユーザが EI レベル割り込みマスク・レジスタ（IMR*m*）に対する操作を行い，例外コード “OsIsrExceptionCode” ／例外コード “OsCounterExceptionCode” で定義された EI レベル割り込みの受け付けを許可した場合の動作を保証していません。

13.2 エントリ処理（直接分岐方式の例外ベクタ）

リセット（RESET），FE レベル割り込み（FENMI，TRAP など），EI レベル割り込み（例外コード“OsIsrExceptionCode”／例外コード“OsCounterExceptionCode”で未定義）などが発生した際，対応する処理（ブート処理，例外／割り込み安全対策処理など）への分岐処理を割り付けるために切り出されたエントリ処理専用ルーチンです。

備考 エントリ処理についての詳細は，「4.5.1 エントリ処理（直接分岐方式の例外ベクタ）」を参照してください。

13.3 ブート処理

RV850 が処理を実行するうえで必要となる最低限のハードウエアを初期化するために切り出された初期処理専用ルーチンであり，ハードウエア・リセットが発生した際にデバイスが強制的に制御を移すハンドラ・アドレスに割り付けられた分岐処理（エントリ処理）から呼び出されます。

備考 ブート処理についての詳細は，「4.2 ブート処理」を参照してください。

13.4 カーネル初期化部

カーネル初期化部は，RV850 が処理を実行するうえで必要となる最低限のソフトウエアを初期化するために切り出された初期化処理専用ルーチンであり，ブート処理で StartOS を発行することにより呼び出されます。
なお，カーネル初期化部では，以下に示した処理が行われます。

(1) オブジェクトの生成／登録
タスク情報，割り込みサービス・ルーチン情報，リソース情報などで定義された情報をもとに，オブジェクト（タスク，割り込みサービス・ルーチン，リソースなど）の生成／登録を行います。

(2) オブジェクトの起動
タスク情報，アラーム情報，スケジュール・テーブル情報などで定義された情報をもとに，オブジェクト（タスク，アラーム，スケジュール・テーブルなど）の起動を行います。

備考 カーネル初期化部は，RV850 が提供している機能の一部であるため，ユーザがカーネル初期化部を記述する必要はありません。

13.5 フック・ルーチン

フック・ルーチンは，カーネル初期化部から呼び出される初期化処理専用フック・ルーチンです。

備考 1. RV850 では，スケーラビリティ・クラス“OsScalabilityClass”がスケーラビリティ・クラス 3（SC3）の場合，共通フック・ルーチン StartupHook，固有フック・ルーチン StartupHook_*OsApplication* の順に呼び出しを行います。

備考 2. StartupHook についての詳細は「11.2 共通フック・ルーチン」を，StartupHook_*OsApplication* についての詳細は「10.3 固有フック・ルーチン」を参照してください。

14. システム・サービス

本章では，RV850 が提供しているシステム・サービスについて解説しています。

14.1 概　　要

RV850 が提供するシステム・サービスは，ユーザが記述した処理プログラムから RV850 が直接的に管理する資源（オブジェクト，メモリ領域など）を間接的に操作するために用意されたサービス・ルーチンです。
以下に，RV850 が提供するシステム・サービスの一覧を示します。

(1) タスク管理
ActivateTask，TerminateTask，ChainTask，Schedule，GetTaskID，GetTaskState

(2) 割り込み管理
EnableAllInterrupts，DisableAllInterrupts，ResumeAllInterrupts，SuspendAllInterrupts，ResumeOSInterrupts，SuspendOSInterrupts

(3) リソース管理
GetResource，ReleaseResource

(4) イベント管理
SetEvent，ClearEvent，GetEvent，WaitEvent

(5) カウンタ管理
IncrementCounter，GetCounterValue，GetElapsedValue

(6) アラーム管理
GetAlarmBase，GetAlarm，SetRelAlarm，SetAbsAlarm，CancelAlarm

(7) スケジュール・テーブル管理
StartScheduleTableRel，StartScheduleTableAbs，StopScheduleTable，NextScheduleTable，GetScheduleTableStatus

(8) OS アプリケーション管理
GetApplicationID，GetCurrentApplicationID，GetISRID，CallTrustedFunction，CallTrustedFunctionRestricted，CheckISRMemoryAccess，CheckTaskMemoryAccess，CheckObjectAccess，CheckObjectOwnership，TerminateApplication，AllowAccess，GetApplicationState

(9) OS 実行管理
StartOS，ShutdownOS，GetActiveApplicationMode，ControlIdle

(10) ユーティリティ関数
InitApplicationInterrupts，_kernel_fv0_InitializeIntService，OSIllegalException_SystemRegister_ExcCode，OSIllegalException_SystemRegister_ExcPC，OSErrorGetServiceId，OSError_SystemService_Parameter

備考　AUTOSAR 仕様では，CallTrustedFunctionRestricted，ユーティリティ関数 InitApplicationInterrupts，_kernel_fv0_InitializeIntService，OSIllegalException_SystemRegister_ExcCode，OSIllegalException_SystemRegister_ExcPC の規定が行われていません。
CallTrustedFunctionRestricted および該当ユーティリティ関数は，RV850 が独自に追加したユーティリティ関数です。

14.1.1 システム・サービスの呼び出し

RV850 が提供しているシステム・サービスは C 言語関数として実装されているため，C 言語で記述された処理プログラムから発行する場合は，通常の C 言語関数と同様の方法で呼び出しを行うことにより該当処理が行われます。

また，システム・サービスをアセンブリ言語で記述された処理プログラムから発行する場合は，システム・サービスを発行する直前で使用する C コンパイラ・パッケージの関数呼び出し規約に従った引数，および戻り番地の設定を行うことにより該当処理が行われます。

なお，システム初期化処理が完了する以前に割り込み管理のシステム・サービスを発行する際には，該当システム・サービスの発行前に _kernel_fv0_InitializeIntService の発行が必要となります。

備考 1. システム・サービスを発行する処理プログラムでは，以下に示したヘッダ・ファイルの定義（インクルード）が必要となります。

Os.h： 標準ヘッダ・ファイル

備考 2. RV850 では，システム・サービスが発行された際，システム・サービスを発行した処理プログラムの種類に応じたスタック（タスク用スタック，システム用スタックなど）への切り替え処理を行います。したがって，処理プログラム内でシステム・サービスの発行に伴うスタックの切り替え処理を記述する必要はありません。

備考 3. SYSCALL 命令の発行時に不正な値が指定された際には，例外／割り込み安全対策処理 _kernel_e_IllegalExcEntry の呼び出しが行われます。

14.2 データ・マクロ

RV850 が提供しているシステム・サービスを発行する際に使用するデータ・マクロについて以下に示します。

14.2.1 データ・タイプ

以下に，システム・サービスを発行する際に使用するデータ・タイプを示します。
なお，データ・タイプは，標準ヘッダ・ファイル “Os.h” から呼び出されるヘッダ・ファイル “Os_types.h” に定義されています。

表 14.1 データ・タイプ

マクロ	型	説明
AccessType	unsigned short	アクセス権
AlarmBaseRefType	AlarmBaseType *	アラーム・ベース情報を格納する領域へのポインタ
AlarmBaseType	struct _AlarmBaseType	アラーム・ベース情報
AlarmType	signed short	アラーム識別子
ApplicationStateRefType	ApplicationStateType *	OS アプリケーションの状態を格納する領域へのポインタ
ApplicationStateType	signed short	OS アプリケーションの状態
ApplicationType	signed short	OS アプリケーション識別子
AppModeType	signed short	アプリケーション・モード
boolean	unsigned char	真偽値（TRUE，または FALSE）
CoreIdType	unsigned short	コアの識別子
CounterType	signed short	カウンタ識別子
EventMaskRefType	EventMaskType *	イベント・マスクを格納する領域へのポインタ
EventMaskType	unsigned long	イベント・マスク
float32	float	32 ビット浮動小数点数
float64	double	64 ビット浮動小数点数
IdleModeType	unsigned char	アイドル・ハンドラの動作モード
ISRType	signed short	割り込みサービス・ルーチン識別子
MemorySizeType	unsigned long	メモリ領域のサイズ
MemoryStartAddressType	unsigned long *	メモリ領域の先頭アドレス
ObjectAccessType	signed short	アクセス権
ObjectTypeType	signed short	オブジェクトの種類
OSServiceIdType	signed char	システム・サービス識別子
PhysicalTimeType	unsigned long	時間
ProtectionReturnType	signed short	共通フック・ルーチン ProtectionHook の戻り値
RestartType	signed short	再起動オプション
ResourceType	signed short	リソース識別子

マクロ	型	説明
ScheduleTableStatusRefType	ScheduleTableStatusType *	スケジュール・テーブルの状態を格納する領域へのポインタ
ScheduleTableStatusType	signed short	スケジュール・テーブルの状態
ScheduleTableType	signed short	スケジュール・テーブル識別子
sint8	signed char	符号付き 8 ビット整数
sint8_least	signed long	符号付き整数（最小：8 ビット）
sint16	signed short	符号付き 16 ビット整数
sint16_least	signed long	符号付き整数（最小：16 ビット）
sint32	signed long	符号付き 32 ビット整数
sint32_least	signed long	符号付き整数（最小：32 ビット）
sint64	signed long long	符号付き 64 ビット整数
StatusType	unsigned char	システム・サービスの戻り値
SystemRegisterType	unsigned long	システム・レジスタの値
TaskRefType	TaskType *	タスク識別子を格納する領域へのポインタ
TaskType	signed short	タスク識別子
TaskStateRefType	TaskStateType *	タスクの状態を格納する領域へのポインタ
TaskStateType	signed short	タスクの状態
TickRefType	TickType *	カウント値を格納する領域／カウント値を格納した領域へのポインタ
TickType	unsigned long	カウント値
TrustedFunctionIndexType	signed short	Trusted 関数識別子
TrustedFunctionParameterRefType	unsigned long *	引き継ぎデータを格納した領域へのポインタ
uint8	unsigned char	符号なし 8 ビット整数
uint8_least	unsigned long	符号なし整数（最小：8 ビット）
uint16	unsigned short	符号なし 16 ビット整数
uint16_least	unsigned long	符号なし整数（最小：16 ビット）
uint32	unsigned long	符号なし 32 ビット整数
uint32_least	unsigned long	符号なし整数（最小：32 ビット）
uint64	unsigned long long	符号なし 64 ビット整数

備考　AUTOSAR 仕様では，EventMaskType，および TickType を 64 ビット値とする旨の規定が行われていますが，RV850 では，EventMaskType，および TickType を 32 ビット値としています。

14.2.2 エラー・ステータス

以下に，システム・サービスからの戻り値（エラー・ステータス）に対応したマクロを示します。
なお，エラー・ステータスは，標準ヘッダ・ファイル “Os.h” から呼び出されるヘッダ・ファイル “Os_error.h” に定義されています。

表 14.2 エラー・ステータス

マクロ	数値	説明
E_OK	0x0	正常終了
E_OS_ACCESS	0x1	対象オブジェクトにアクセスすることができない
E_OS_CALLEVEL	0x2	発行有効範囲外の処理プログラムから発行した
E_OS_ID	0x3	識別子の指定が不正である
E_OS_LIMIT	0x4	起動要求数が最大起動要求数を越えた
E_OS_NOFUNC	0x5	対象オブジェクトを操作することができない
E_OS_RESOURCE	0x6	リソースの獲得中に禁止されている操作を行った
E_OS_STATE	0x7	対象オブジェクトの状態が不正である
E_OS_VALUE	0x8	引数の指定が不正である
E_OS_SERVICEID	0x11	Trusted 関数識別子の指定が不正である
E_OS_PARAM_POINTER	0x12	引数の指定が不正（NULL ポインタ）である
E_OS_ILLEGAL_ADDRESS	0x13	引数で指定された領域に対するアクセス権がない
E_OS_MISSINGEND	0x14	TerminateTask，ChainTask を発行することなくタスクが終了した
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した
E_OS_STACKFAULT	0x16	スタックのオーバーフローを検出した
E_OS_PROTECTION_MEMORY	0x17	不正なメモリ・アクセスを検出した
E_OS_SYS_ILLEGAL_EXCEPTION	0x1F	割り込みサービス・ルーチン情報で未定義の割り込みの発生を検出した

備考 1. AUTOSAR 仕様では，スケーラビリティ・クラス “OsScalabilityClass” がスケーラビリティ・クラス 3（SC3）／スケーラビリティ・クラス 4（SC4）の場合，エラー・ステータス E_OS_CALLEVEL，および E_OS_DISABLEDINT を返却する旨の規定（OS088，OS093）が行われていますが，RV850 では，スケーラビリティ・クラスが SC3 の場合に加え，スケーラビリティ・クラスが SC1，ステータス種別が拡張ステータスの場合にも本エラー・ステータスを返却するようにしています。

備考 2. AUTOSAR 仕様では，エラー・ステータス E_OS_SYS_ILLEGAL_EXCEPTION の規定が行われていません。
本マクロは，RV850 が独自に追加したマクロです。

14.2.3 不正なタスク識別子

以下に，GetTaskID を発行した際，引数 *TaskID* に格納される数値に対応したマクロを示します。
なお，不正なタスク識別子は，標準ヘッダ・ファイル “Os.h” から呼び出されるヘッダ・ファイル “Os_constant.h” に定義されています。

表 14.3 不正なタスク識別子

マクロ	数値	説明
INVALID_TASK	0x7FFF	対象タスクが存在しない

14.2.4 タスクの状態

以下に，GetTaskState を発行した際，引数 *State* に格納される数値に対応したマクロを示します。
なお，タスクの状態は，標準ヘッダ・ファイル “Os.h” から呼び出されるヘッダ・ファイル “Os_constant.h” に定義されています。

表 14.4 タスクの状態

マクロ	数値	説明
SUSPENDED	0x0	SUSPENDED 状態
READY	0x1	READY 状態
RUNNING	0x2	RUNNING 状態
WAITING	0x4	WAITING 状態

14.2.5 スケジュール・テーブルの状態

以下に，GetScheduleTableStatus を発行した際，引数 *ScheduleStatus* に格納される数値に対応したマクロを示します。
なお，スケジュール・テーブルの状態は，標準ヘッダ・ファイル “Os.h” から呼び出されるヘッダ・ファイル “Os_constant.h” に定義されています。

表 14.5 スケジュール・テーブルの状態

マクロ	数値	説明
SCHEDULETABLE_STOPPED	0x0	STOPPED 状態
SCHEDULETABLE_NEXT	0x1	NEXT 状態
SCHEDULETABLE_RUNNING	0x4	RUNNING 状態

14.2.6 異常終了

以下に，GetApplicationID，GetCurrentApplicationID，GetISRID，CheckObjectOwnership からの戻り値（異常終了）に対応したマクロを示します。
なお，異常終了は，標準ヘッダ・ファイル “Os.h” から呼び出されるヘッダ・ファイル “Os_constant.h” に定義されています。

表 14.6 異常終了

マクロ	数値	説明
INVALID_ISR	0x7FFF	異常終了
INVALID_OSAPPLICATION	0x7FFF	異常終了

14.2.7 アクセス権の種類

以下に，CheckISRMemoryAccess，CheckTaskMemoryAccess からの戻り値（アクセス権の種類）に対応したマクロを示します。
なお，アクセス権の種類は，標準ヘッダ・ファイル “Os.h” から呼び出されるヘッダ・ファイル “Os_constant.h” に定義されています。

表 14.7 アクセス権の種類

マクロ	数値	説明
T_u2_NOACCESS	0x0	アクセス権なし
T_u2_EXECUTABLE	0x2	実行可
T_u2_READABLE	0x4	読み込み可
T_u2_WRITEABLE	0x8	書き込み可
T_u2_STACKSPACE	0x10	スタック領域

備考 複数のアクセス権を有する場合，戻り値（アクセス権の種類）は各数値の和となります。

14.2.8 オブジェクトの種類

以下に，CheckObjectAccess，CheckObjectOwnership を発行する際，引数 *ObjectType* に設定可能なマクロを示します。
なお，オブジェクトの種類は，標準ヘッダ・ファイル “Os.h” から呼び出されるヘッダ・ファイル “Os_constant.h” に定義されています。

表 14.8 オブジェクトの種類

マクロ	数値	説明
OBJECT_TASK	0x1	タスク
OBJECT_ISR	0x2	割り込みサービス・ルーチン
OBJECT_ALARM	0x3	アラーム
OBJECT_RESOURCE	0x4	リソース
OBJECT_COUNTER	0x5	カウンタ
OBJECT_SCHEDULETABLE	0x6	スケジュール・テーブル

14.2.9 アクセス権の有無

以下に，CheckObjectAccess からの戻り値（アクセス権の有無）に対応したマクロを示します。
なお，アクセス権の有無は，標準ヘッダ・ファイル “Os.h” から呼び出されるヘッダ・ファイル “Os_constant.h” に定義されています。

表 14.9 アクセス権の有無

マクロ	数値	説明
NO_ACCESS	0x0	アクセス権なし
ACCESS	0x1	アクセス権あり

14.2.10 再起動オプション

以下に，TerminateApplication を発行する際，引数 *RestartOption* に設定可能なマクロを示します。
なお，再起動オプションは，標準ヘッダ・ファイル “Os.h” から呼び出されるヘッダ・ファイル “Os_constant.h” に定義されています。

表 14.10 再起動オプション

マクロ	数値	説明
NO_RESTART	0x0	終了処理を行います。
RESTART	0x1	終了処理を行ったのち，以下の操作を行います。 - タスク識別子 “OsRestartTask” で定義されたタスクを SUSPENDED 状態から READY 状態へと遷移 - 対象 OS アプリケーションを APPLICATION_ACCESSIBLE 状態から APPLICATION_RESTARTING 状態へと遷移

14.2.11 OS アプリケーションの状態

以下に，GetApplicationState を発行した際，引数 *Value* に格納される数値に対応したマクロを示します。
なお，OS アプリケーションの状態は，標準ヘッダ・ファイル “Os.h” から呼び出されるヘッダ・ファイル “Os_constant.h” に定義されています。

表 14.11 OS アプリケーションの状態

マクロ	数値	説明
APPLICATION_ACCESSIBLE	0x0	APPLICATION_ACCESSIBLE 状態
APPLICATION_RESTARTING	0x1	APPLICATION_RESTARTING 状態
APPLICATION_TERMINATED	0x2	APPLICATION_TERMINATED 状態

14.2.12 システム・サービス識別子

以下に，OSErrorGetServiceId からの戻り値（システム・サービス識別子）に対応したマクロを示します。
なお，システム・サービス識別子は，標準ヘッダ・ファイル “Os.h” から呼び出されるヘッダ・ファイル “Os_service.h” に定義されています。

表 14.12 システム・サービス識別子

マクロ	数値	説明
OSServiceId_GetApplicationID	0x0	GetApplicationID のシステム・サービス識別子
OSServiceId_GetISRID	0x1	GetISRID のシステム・サービス識別子
OSServiceId_CallTrustedFunction	0x2	CallTrustedFunction のシステム・サービス識別子
OSServiceId_CheckISRMemoryAccess	0x3	CheckISRMemoryAccess のシステム・サービス識別子
OSServiceId_CheckTaskMemoryAccess	0x4	CheckTaskMemoryAccess のシステム・サービス識別子
OSServiceId_CheckObjectAccess	0x5	CheckObjectAccess のシステム・サービス識別子
OSServiceId_CheckObjectOwnership	0x6	CheckObjectOwnership のシステム・サービス識別子
OSServiceId_StartScheduleTableRel	0x7	StartScheduleTableRel のシステム・サービス識別子
OSServiceId_StartScheduleTableAbs	0x8	StartScheduleTableAbs のシステム・サービス識別子
OSServiceId_StopScheduleTable	0x9	StopScheduleTable のシステム・サービス識別子
OSServiceId_NextScheduleTable	0xA	NextScheduleTable のシステム・サービス識別子
OSServiceId_GetScheduleTableStatus	0xE	GetScheduleTableStatus のシステム・サービス識別子
OSServiceId_IncrementCounter	0xF	IncrementCounter のシステム・サービス識別子
OSServiceId_GetCounterValue	0x10	GetCounterValue のシステム・サービス識別子
OSServiceId_GetElapsedValue	0x11	GetElapsedValue のシステム・サービス識別子
OSServiceId_TerminateApplication	0x12	TerminateApplication のシステム・サービス識別子
OSServiceId_AllowAccess	0x13	AllowAccess のシステム・サービス識別子
OSServiceId_GetApplicationState	0x14	GetApplicationState のシステム・サービス識別子
OSServiceId_ControlIdle	0x1D	ControlIdle のシステム・サービス識別子
OSServiceId_GetCurrentApplicationID	0x27	GetCurrentApplicationID のシステム・サービス識別子
OSServiceId_StartOS	0x40	StartOS のシステム・サービス識別子
OSServiceId_ShutdownOS	0x41	ShutdownOS のシステム・サービス識別子
OSServiceId_GetActiveApplicationMode	0x42	GetActiveApplicationMode のシステム・サービス識別子
OSServiceId_ActivateTask	0x43	ActivateTask のシステム・サービス識別子
OSServiceId_TerminateTask	0x44	TerminateTask のシステム・サービス識別子

マクロ	数値	説明
OSServiceId_ChainTask	0x45	ChainTask のシステム・サービス識別子
OSServiceId_Schedule	0x46	Schedule のシステム・サービス識別子
OSServiceId_GetTaskID	0x47	GetTaskID のシステム・サービス識別子
OSServiceId_GetTaskState	0x48	GetTaskState のシステム・サービス識別子
OSServiceId_EnableAllInterrupts	0x49	EnableAllInterrupts のシステム・サービス識別子
OSServiceId_DisableAllInterrupts	0x4A	DisableAllInterrupts のシステム・サービス識別子
OSServiceId_ResumeAllInterrupts	0x4B	ResumeAllInterrupts のシステム・サービス識別子
OSServiceId_SuspendAllInterrupts	0x4C	SuspendAllInterrupts のシステム・サービス識別子
OSServiceId_ResumeOSInterrupts	0x4D	ResumeOSInterrupts のシステム・サービス識別子
OSServiceId_SuspendOSInterrupts	0x4E	SuspendOSInterrupts のシステム・サービス識別子
OSServiceId_GetResource	0x4F	GetResource のシステム・サービス識別子
OSServiceId_ReleaseResource	0x50	ReleaseResource のシステム・サービス識別子
OSServiceId_SetEvent	0x51	SetEvent のシステム・サービス識別子
OSServiceId_ClearEvent	0x52	ClearEvent のシステム・サービス識別子
OSServiceId_GetEvent	0x53	GetEvent のシステム・サービス識別子
OSServiceId_WaitEvent	0x54	WaitEvent のシステム・サービス識別子
OSServiceId_GetAlarmBase	0x55	GetAlarmBase のシステム・サービス識別子
OSServiceId_GetAlarm	0x56	GetAlarm のシステム・サービス識別子
OSServiceId_SetRelAlarm	0x57	SetRelAlarm のシステム・サービス識別子
OSServiceId_SetAbsAlarm	0x58	SetAbsAlarm のシステム・サービス識別子
OSServiceId_CancelAlarm	0x59	CancelAlarm のシステム・サービス識別子
OSServiceId_ExitTask	0x62	TerminateTask を発行せずタスクを終了
OSServiceId_ExitIsr	0x65	リソースを獲得したままカテゴリ 2 の割り込みサービス・ルーチンを終了，およびクリティカル・セクションを終了せずカテゴリ 2 の割り込みサービス・ルーチンを終了

14.2.13 カウンタ情報

以下に，カウンタ情報に対応したマクロを示します。
なお，カウンタ情報は，CF ファイルに対してコンフィギュレータを実行した際に生成される情報ファイル（カーネル・マクロ・ファイル）に定義されています。

表 14.13 カウンタ情報

マクロ	説明
OSMAXALLOWEDVALUE_*OsCounter*	最大カウント値 “OsCounterMaxAllowedValue”（単位：tick）
OSMINCYCLE_*OsCounter*	最小周期 “OsCounterMinCycle”（単位：tick）
OSTICKSPERBASE_*OsCounter*	基本カウント値 “OsCounterTicksPerBase”（単位：tick）
OSMAXALLOWEDVALUE	システム・カウンタ（識別子 “OsCounter” に SYS_COUNTER が指定された場合）の最大カウント値 “OsCounterMaxAllowedValue”（単位：tick）
OSMINCYCLE	システム・カウンタ（識別子 “OsCounter” に SYS_COUNTER が指定された場合）の最小周期 “OsCounterMinCycle”（単位：tick）
OSTICKSPERBASE	システム・カウンタ（識別子 “OsCounter” に SYS_COUNTER が指定された場合）の基本カウント値 “OsCounterTicksPerBase”（単位：tick）
OSTICKDURATION	システム・カウンタ（識別子 “OsCounter” に SYS_COUNTER が指定された場合）の基本時間（単位：ns） 「109 * 1 tick 当たりの秒数 “OsSecondsPerTick” * 基本カウント値 “OsCounterTicksPerBase”」の算出結果
OS_TICK2NS_*OsCounter* (*Value*)	引数 *Value*（単位：tick）を ns 単位の値に変換 「*Value* * 109 * 1 tick 当たりの秒数 “OsSecondsPerTick”」の算出結果
OS_TICK2US_*OsCounter* (*Value*)	引数 *Value*（単位：tick）を ms 単位の値に変換 「*Value* * 106 * 1 tick 当たりの秒数 “OsSecondsPerTick”」の算出結果
OS_TICK2MS_*OsCounter* (*Value*)	引数 *Value*（単位：tick）を ms 単位の値に変換 「*Value* * 103 * 1 tick 当たりの秒数 “OsSecondsPerTick”」の算出結果
OS_TICK2SEC_*OsCounter* (*Value*)	引数 *Value*（単位：tick）を秒単位の値に変換 「*Value* * 1 tick 当たりの秒数 “OsSecondsPerTick”」の算出結果

14.2.14 アクセス権の有無

以下に，アクセス権の有無に対応したマクロを示します。
なお，アクセス権の有無は，標準ヘッダ・ファイル “Os.h” から呼び出されるヘッダ・ファイル “Os_constant.h” に定義されています。

表 14.14 アクセス権の有無

マクロ	説明
OSMEMORY_IS_EXECUTEABLE (AccessType *Type*)	引数 *Type* に CheckISRMemoryAccess，または CheckTaskMemoryAccess の戻り値を指定し，実行可否を確認
OSMEMORY_IS_READABLE (AccessType *Type*)	引数 *Type* に CheckISRMemoryAccess，または CheckTaskMemoryAccess の戻り値を指定し，読み込み可否を確認
OSMEMORY_IS_WRITEABLE (AccessType *Type*)	引数 *Type* に CheckISRMemoryAccess，または CheckTaskMemoryAccess の戻り値を指定し，書き込み可否を確認
OSMEMORY_IS_STACKSPACE (AccessType *Type*)	引数 *Type* に CheckISRMemoryAccess，または CheckTaskMemoryAccess の戻り値を指定し，スタック領域であるか否かを確認

14.2.15 アイドル・ハンドラの動作モード

以下に，アイドル・ハンドラの動作モードに対応したマクロを示します。
なお，アイドル・ハンドラの動作モードは，標準ヘッダ・ファイル “Os.h” から呼び出されるヘッダ・ファイル “Os_constant.h” に定義されています。

表 14.15 アイドル・ハンドラの動作モード

マクロ	数値	説明
IDLE_NO_HALT	0x0	StartOS 発行直後の動作モード
IDLEMODE_n	n	ユーザが任意に定義した n 番目の動作モード

備考 n の取りうる値の範囲は，1 ～ 31 となります。

14.2.16 コアの識別子

以下に，コアの識別子に対応したマクロを示します。
なお，コアの識別子は，標準ヘッダ・ファイル “Os.h” から呼び出されるヘッダ・ファイル “Os_constant.h” に定義されています。

表 14.16 コアの識別子

マクロ	数値	説明
OS_CORE_ID_n	n	コアの識別子

備考 AUTOSAR 仕様では，n の取りうる値の範囲は 0 ～ OsNumberOfCores - 1 である旨の規定が行われていますが，RV850 では，n の取りうる値の範囲は，1 ～ 8 となります。

14.3 データ構造体

RV850 が提供しているシステム・サービスを処理プログラムから発行する際に使用するデータ構造体について以下に示します。

14.3.1 アラーム・ベース情報

以下に，GetAlarmBase を発行した際，引数 *Info* に格納されるアラーム・ベース情報を示します。
なお，アラーム・ベース情報は，標準ヘッダ・ファイル “Os.h” から呼び出されるヘッダ・ファイル “Os_types.h” に定義されています。

図 14.1 アラーム・ベース情報

```
struct _AlarmBaseType {
    TickType maxallowedvalue;  /* 最大カウント値 “OsCounterMaxAllowedValue” */
    TickType mincycle;         /* 最小周期 “OsCounterMinCycle” */
    TickType ticksperbase;     /* 基本カウント値 “OsCounterTicksPerBase” */
};

typedef struct _AlarmBaseType AlarmBaseType;
typedef AlarmBaseType *AlarmBaseRefType;
```

(1) maxallowedvalue
GetAlarmBase の引数 *AlarmID* で指定したアラームと関連付けられているカウンタの最大カウント値 “OsCounterMaxAllowedValue” が格納されます。

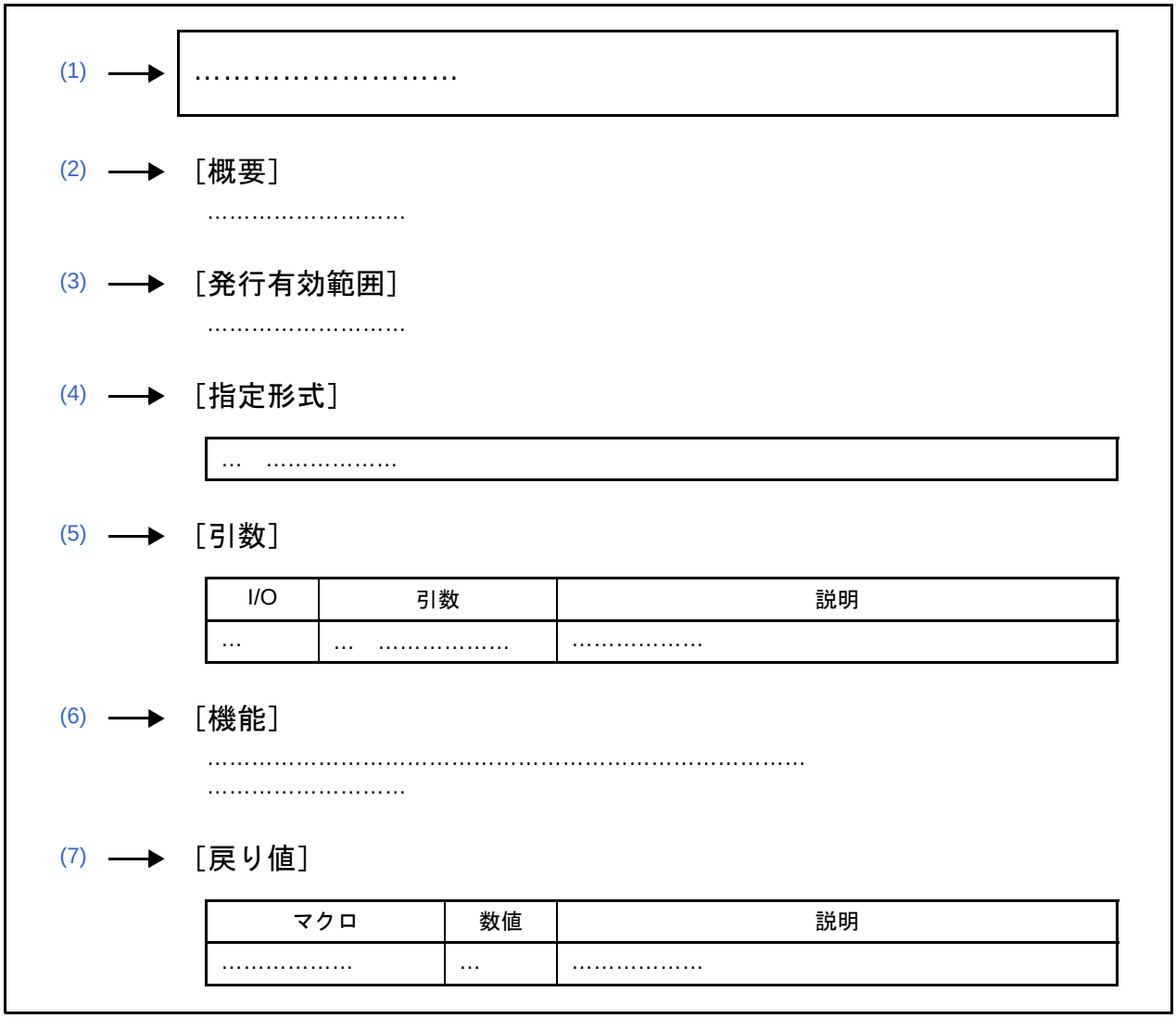
(2) mincycle
GetAlarmBase の引数 *AlarmID* で指定したアラームと関連付けられているカウンタの最小周期 “OsCounterMinCycle” が格納されます。

(3) ticksperbase
GetAlarmBase の引数 *AlarmID* で指定したアラームと関連付けられているカウンタの基本カウント値 “OsCounterTicksPerBase” が格納されます。

14.4 システム・サービス・リファレンス

本節では，RV850 が提供するシステム・サービスについて，次の記述フォーマットに従って説明します。

図 14.2 システム・サービスの記述フォーマット



(1) 名称
システム・サービスの名称を示しています。

(2) ［概要］
システム・サービスの概要を示しています。

(3) ［発行有効範囲］
システム・サービスの発行が可能な処理プログラムの種類，および区間を示しています。

(a) ブート処理
RV850 が処理を実行するうえで必要となる最低限のハードウエアを初期化するために切り出された初期処理専用ルーチンです。

(b) タスク
システム・サービスを発行して状態操作を行わない限り，または CF ファイルで定義した条件が成立しない限り実行されることのない処理ルーチンです。
なお，タスクの種類により発行の可否が異なる場合には，発行可能なタスクを以下のように表記しています。

- 拡張タスクからのみ発行可能な場合
  タスク（拡張）

(c) 割り込みサービス・ルーチン

割り込みが発生した際に呼び出される割り込み処理専用ルーチンです。
なお，割り込みサービス・ルーチンのカテゴリにより発行の可否が異なる場合には，発行可能な割り込みサービス・ルーチンを以下のように表記しています。

- カテゴリ 2 の割り込みサービス・ルーチンからのみ発行可能な場合
  割り込みサービス・ルーチン（カテゴリ 2）

(d) アラーム・コールバック
アラームが満了した際に呼び出される満了処理専用ルーチンです。

(e) 共通フック・ルーチン
すべての OS アプリケーションに共通なフック・ルーチンです。
共通フック・ルーチン（StartupHook，ShutdownHook，PostTaskHook，PreTaskHook，ErrorHook，ProtectionHook）の種類により発行の可否が異なる場合には，発行可能な共通フック・ルーチンを以下のように表記しています。

- PostTaskHook，PreTaskHook，ErrorHook，ProtectionHook からのみ発行可能な場合
  共通フック・ルーチン（PostTaskHook，PreTaskHook，ErrorHook，ProtectionHook）

(f) 固有フック・ルーチン
OS アプリケーション単位に固有なフック・ルーチンです。
固有フック・ルーチン（StartupHook_*OsApplication*，ShutdownHook_*OsApplication*，ErrorHook_*OsApplication*）の種類により発行の可否が異なる場合には，発行可能な固有フック・ルーチンを以下のように表記しています。

- StartupHook_*OsApplication*，ErrorHook_*OsApplication* からのみ発行可能な場合
  固有フック・ルーチン（StartupHook_*OsApplication*，ErrorHook_*OsApplication*）

(g) クリティカル・セクション
以下に示した割り込みの受け付け禁止区間を意味します。

- DisableAllInterrupts の発行から EnableAllInterrupts の発行まで
- SuspendAllInterrupts の発行から ResumeAllInterrupts の発行まで
- SuspendOSInterrupts の発行から ResumeOSInterrupts の発行まで

(4) ［指定形式］
システム・サービスを C 言語で記述された処理プログラムから発行する際の指定形式を示しています。

(5) ［引数］
システム・サービスの引数を次の形式で示しています。

I/O	引数	説明
(a)	(b)	(c)

(a) “I/O” 列
引数の種類

- I … RV850 への入力引数
- O … RV850 からの出力引数

(b) “ 引数 ” 列
引数のデータ・タイプ

(c) “ 説明 ” 列
引数の説明

(6) ［機能］
システム・サービスの機能を示しています。

(7) ［戻り値］
システム・サービスからの戻り値を次の形式で示しています。

マクロ	数値	説明
(a)	(b)	(c)

(a) “ マクロ ” 列
戻り値のマクロ

(b) “数値”列
戻り値の値

(c) “説明”列
戻り値の説明

14.4.1 タスク管理

以下に，RV850 が提供しているタスク管理用システム・サービスを示します。

表 14.17　タスク管理用システム・サービス

システム・サービス名	機能概要
ActivateTask	タスクの起動
TerminateTask	タスクの終了
ChainTask	タスクの終了／起動
Schedule	スケジューラの起動
GetTaskID	RUNNING 状態へと遷移しているタスクのタスク識別子を獲得
GetTaskState	タスクの現在状態を獲得

ActivateTask

[概要]

タスクの起動

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2）

[指定形式]

```
StatusType ActivateTask ( TaskType TaskID );
```

[引数]

I/O	引数	説明
I	TaskType *TaskID*;	タスク識別子

[機能]

対象タスク（引数 *TaskID* で指定されたタスク）を SUSPENDED 状態から READY 状態へと遷移させます。
対象タスクは，SUSPENDED 状態から READY 状態への遷移に伴い，優先度に応じたレディ・キューの最後尾にキューイングされます。

備考 1. 本システム・サービスをスケジューリング属性 “OsTaskSchedule” がプリエンプティブ属性（FULL）のタスクから発行した場合，タスクの状態操作処理（対象タスクを SUSPENDED 状態から READY 状態へと遷移）が完了した際には，スケジューラの起動が行われます。

備考 2. 対象タスクの種別が基本タスクの場合，タスクの状態操作処理（対象タスクを SUSPENDED 状態から READY 状態へと遷移）と伴に，起動要求カウンタの加算処理（起動要求カウンタに 0x1 を加算）が行われます。
対象タスクが SUSPENDED 状態以外の状態（READY 状態，または RUNNING 状態）へと遷移していた際には，タスクの状態操作処理は行われず，優先度に応じたレディ・キューの最後尾へのキューイング処理，および起動要求カウンタの加算処理が行われます。

[戻り値]

マクロ	数値	説明
E_OK	0x0	正常終了
E_OS_ACCESS	0x1	異常終了 - 本システム・サービスを発行した処理プログラムが所属しているOSアプリケーションは，対象タスクに対するアクセス権がない（SC3 限定） - 対象タスクの所属しているOSアプリケーションがAPPLICATION_RESTARTING 状態，またはAPPLICATION_TERMINATED 状態である（SC3 限定）
E_OS_CALLEVEL	0x2	発行有効範囲外の処理プログラムから発行した
E_OS_ID	0x3	引数 *TaskID* の指定が不正である

マクロ	数値	説明
E_OS_LIMIT	0x4	異常終了 - 対象タスクが READY 状態，RUNNING 状態，または WAITING 状態の拡張タスクである - 起動要求数が対象タスクの最大起動要求数 “OsTaskActivation” を越えた
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した

TerminateTask

[概要]

タスクの終了

[発行有効範囲]

タスク

[指定形式]

```
StatusType TerminateTask ( void );
```

[引数]

なし

[機能]

自タスク（本システム・サービスを発行したタスク）を RUNNING 状態から SUSPENDED 状態へと遷移させます。
自タスクは，RUNNING 状態から SUSPENDED 状態への遷移に伴い，優先度に応じたレディ・キューから外れます。
タスクの状態操作処理（自タスクを RUNNING 状態から SUSPENDED 状態へと遷移）が完了した際には，スケジューラの起動が行われます。

備考 1. 自タスクの種別が基本タスクの場合，タスクの状態操作処理（自タスクを RUNNING 状態から SUSPENDED 状態へと遷移）と伴に，起動要求カウンタの減算処理（起動要求カウンタから 0x1 を減算）が行われます。
起動要求カウンタの減算結果が 0x0 以外の場合，タスクの状態操作処理（自タスクを SUSPENDED 状態から READY 状態へと遷移）が行われ，自タスクは再び READY 状態へと遷移します。
タスクの状態操作処理（自タスクを SUSPENDED 状態から READY 状態へと遷移）が完了した際には，スケジューラの起動が行われます。

備考 2. 自タスクが “ インターナル・リソース ” を獲得していた場合，タスクの状態操作処理（自タスクを RUNNING 状態から SUSPENDED 状態へと遷移）と伴に，インターナル・リソースの解放処理，および現在優先度の変更処理（自タスクの現在優先度を初期優先度 “OsTaskPriority” に戻す）が行われます。
インターナル・リソースのシーリング値が INTPRI*x* の際には，割り込みの受け付け許可処理（PMR の PM*n* ビットに対する操作）も行われます。

備考 3. スケーラビリティ・クラス “OsScalabilityClass” にスケーラビリティ・クラス 3（SC3）を定義し，本システム・サービス，または ChainTask を発行することなくタスクが終了した場合，該当タスクに共通フック・ルーチン（ErrorHook），固有フック・ルーチン（ErrorHook_*OsApplication*）が登録されていた際には，E_OS_MISSINGEND（0x14）を引数としたフック・ルーチンの呼び出しが行われます。

[戻り値]

マクロ	数値	説明
E_OS_CALLEVEL	0x2	発行有効範囲外の処理プログラムから発行した
E_OS_RESOURCE	0x6	自タスクが “ 通常のリソース ” を獲得している
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した

ChainTask

[概要]

タスクの終了／起動

[発行有効範囲]

タスク

[指定形式]

```
StatusType ChainTask ( TaskType TaskID );
```

[引数]

I/O	引数	説明
I	TaskType TaskID;	タスク識別子

[機能]

自タスク（本システム・サービスを発行したタスク）を RUNNING 状態から SUSPENDED 状態へ，対象タスク（引数 *TaskID* で指定されたタスク）を SUSPENDED 状態から READY 状態へと遷移させます。

自タスクは，RUNNING 状態から SUSPENDED 状態への遷移に伴い，優先度に応じたレディ・キューから外れ，対象タスクは，SUSPENDED 状態から READY 状態への遷移に伴い，優先度に応じたレディ・キューの最後尾にキューイングされます。

タスクの状態操作処理（自タスクを RUNNING 状態から SUSPENDED 状態へ，対象タスクを SUSPENDED 状態から READY 状態へと遷移）が完了した際には，スケジューラの起動が行われます。

備考 1. 引数 *TaskID* に自タスクを指定した場合，自タスクは SUSPENDED 状態へと遷移することなく，RUNNING 状態から READY 状態へと遷移します。
タスクの状態操作処理（自タスクを RUNNING 状態から READY 状態へと遷移）が完了した際には，スケジューラの起動が行われます。

備考 2. 自タスクの種別が基本タスクの場合，タスクの状態操作処理（自タスクを RUNNING 状態から SUSPENDED 状態へと遷移）と伴に，起動要求カウンタの減算処理（起動要求カウンタから 0x1 を減算）が行われます。
起動要求カウンタの減算結果が 0x0 以外の場合，タスクの状態操作処理（自タスクを SUSPENDED 状態から READY 状態へと遷移）が行われ，自タスクは再び READY 状態へと遷移します。
タスクの状態操作処理（自タスクを SUSPENDED 状態から READY 状態へと遷移）が完了した際には，スケジューラの起動が行われます。

備考 3. 対象タスクの種別が基本タスクの場合，タスクの状態操作処理（対象タスクを SUSPENDED 状態から READY 状態へと遷移）と伴に，起動要求カウンタの加算処理（起動要求カウンタに 0x1 を加算）が行われます。
対象タスクが SUSPENDED 状態以外の状態（READY 状態，または RUNNING 状態）へと遷移していた際には，タスクの状態操作処理は行われず，優先度に応じたレディ・キューの最後尾へのキューイング処理，および起動要求カウンタの加算処理が行われます。

備考 4. 自タスクが “ インターナル・リソース ” を獲得していた場合，タスクの状態操作処理（自タスクを RUNNING 状態から SUSPENDED 状態へ，または自タスクを RUNNING 状態から READY 状態へと遷移）と伴に，インターナル・リソースの解放処理，および現在優先度の変更処理（自タスクの現在優先度を初期優先度 “OsTaskPriority” に戻す）が行われます。
インターナル・リソースのシーリング値が INTPRI*x* 際には，割り込みの受け付け許可処理（PMR の PM*n* ビットに対する操作）も行われます。

備考 5. スケーラビリティ・クラス “OsScalabilityClass” にスケーラビリティ・クラス 3（SC3）を定義し，TerminateTask，または本システム・サービスを発行することなくタスクが終了した場合，該当タスク

に共通フック・ルーチン（ErrorHook），固有フック・ルーチン（ErrorHook_*OsApplication*）が登録されていた際には，E_OS_MISSINGEND（0x14）を引数としたフック・ルーチンの呼び出しが行われます。

［戻り値］

マクロ	数値	説明
E_OS_ACCESS	0x1	異常終了 - 自タスクが所属している OS アプリケーションは，対象タスクに対するアクセス権がない（SC3 限定） - 対象タスクの所属している OS アプリケーションが APPLICATION_RESTARTING 状態，または APPLICATION_TERMINATED 状態である（SC3 限定）
E_OS_CALLEVEL	0x2	発行有効範囲外の処理プログラムから発行した
E_OS_ID	0x3	引数 *TaskID* の指定が不正である
E_OS_LIMIT	0x4	異常終了 - 対象タスクが READY 状態，RUNNING 状態，または WAITING 状態の拡張タスクである - 起動要求数が対象タスクの最大起動要求数 “OsTaskActivation” を越えた
E_OS_RESOURCE	0x6	自タスクが “ 通常のリソース ” を獲得している
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した

Schedule

[概要]

スケジューラの起動

[発行有効範囲]

タスク

[指定形式]

```
StatusType Schedule ( void );
```

[引数]

なし

[機能]

スケジューラを起動します。

備考　自タスクが“インターナル・リソース”を獲得していた場合，インターナル・リソースの解放処理，および現在優先度の変更処理（自タスクの現在優先度を，該当リソースを獲得する以前に設定されていた優先度に戻す）を行ったのち，スケジューラを起動します。
インターナル・リソースのシーリング値が INTPRI*x* の際には，割り込みの受け付け許可処理（PMR の PM*n* ビットに対する操作）も行われます。

[戻り値]

マクロ	数値	説明
E_OK	0x0	正常終了
E_OS_CALLEVEL	0x2	発行有効範囲外の処理プログラムから発行した
E_OS_RESOURCE	0x6	自タスクが“通常のリソース”を獲得している
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した

GetTaskID

[概要]

RUNNING 状態へと遷移しているタスクのタスク識別子を獲得

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2），共通フック・ルーチン（PostTaskHook，PreTaskHook，ErrorHook，ProtectionHook），固有フック・ルーチン（ErrorHook_*OsApplication*）

[指定形式]

```
StatusType GetTaskID ( TaskRefType TaskID );
```

[引数]

I/O	引数	説明
O	`TaskRefType` *TaskID*;	獲得したタスク識別子を格納する領域へのポインタ

[機能]

対象タスク（本システム・サービスを発行した際，RUNNING 状態へと遷移しているタスク）のタスク識別子を獲得し，引数 *TaskID* で指定された領域に格納します。

備考 1. 本システム・サービスを発行した際，対象タスクが存在しなかった場合には，引数 *TaskID* で指定された領域に INVALID_TASK（0x7FFF）を格納します。

備考 2. 本システム・サービスが戻り値として返すタスク識別子は数値であり，タスク情報の識別子 “OsTask” に指定された名前との対応は，コンフィギュレータが出力した SIT ファイルに定義されています。

[戻り値]

マクロ	数値	説明
E_OK	0x0	正常終了
E_OS_CALLEVEL	0x2	発行有効範囲外の処理プログラムから発行した
E_OS_PARAM_POINTER	0x12	引数 *TaskID* の指定が不正（NULL ポインタ）である
E_OS_ILLEGAL_ADDRESS	0x13	本システム・サービスを発行した処理プログラムは，引数 *TaskID* で指定された領域に対するアクセス権がない（SC3 限定）
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した

GetTaskState

[概要]

タスクの現在状態を獲得

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2），共通フック・ルーチン（PostTaskHook，PreTaskHook，ErrorHook），固有フック・ルーチン（ErrorHook_*OsApplication*）

[指定形式]

```
StatusType GetTaskState ( TaskType TaskID, TaskStateRefType State );
```

[引数]

I/O	引数	説明
I	`TaskType` *TaskID*;	タスク識別子
O	`TaskStateRefType` *State*;	獲得した現在状態を格納する領域へのポインタ

[機能]

対象タスク（引数 *TaskID* で指定されたタスク）の現在状態を獲得し，引数 *State* で指定された領域に格納します。
なお，引数 *State* には，現在状態の種類により，以下の値が格納されます。

マクロ	数値	説明
SUSPENDED	0x0	SUSPENDED 状態
READY	0x1	READY 状態
RUNNING	0x2	RUNNING 状態
WAITING	0x4	WAITING 状態

備考 1. 引数 *State* で指定された領域に格納される現在状態は，対象タスクの起動要求カウンタに保持されている値に影響を受けません。

備考 2. WaitEvent の発行により呼び出された共通フック・ルーチン（PostTaskHook）で本システム・サービスを発行した場合，引数 *State* には WAITING が格納されます。

[戻り値]

マクロ	数値	説明
E_OK	0x0	正常終了

マクロ	数値	説明
E_OS_ACCESS	0x1	異常終了 - 本システム・サービスを発行した処理プログラムが所属しているOSアプリケーションは，対象タスクに対するアクセス権がない（SC3限定） - 対象タスクの所属しているOSアプリケーションがAPPLICATION_RESTARTING状態，またはAPPLICATION_TERMINATED状態である（SC3限定）
E_OS_CALLEVEL	0x2	発行有効範囲外の処理プログラムから発行した
E_OS_ID	0x3	引数 *TaskID* の指定が不正である
E_OS_PARAM_POINTER	0x12	引数 *State* の指定が不正（NULLポインタ）である
E_OS_ILLEGAL_ADDRESS	0x13	本システム・サービスを発行した処理プログラムは，引数 *State* で指定された領域に対するアクセス権がない（SC3限定）
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した

14.4.2 割り込み管理

以下に，RV850 が提供している割り込み管理用システム・サービスを示します。

表 14.18　割り込み管理用システム・サービス

システム・サービス名	機能概要
EnableAllInterrupts	割り込みの受け付けを許可（ネスト管理なし）
DisableAllInterrupts	割り込みの受け付けを禁止（ネスト管理なし）
ResumeAllInterrupts	割り込みの受け付けを許可（ネスト管理あり）
SuspendAllInterrupts	割り込みの受け付けを禁止（ネスト管理あり）
ResumeOSInterrupts	カテゴリ 2 割り込みの受け付けを許可（ネスト管理あり）
SuspendOSInterrupts	カテゴリ 2 割り込みの受け付けを禁止（ネスト管理あり）

EnableAllInterrupts

[概要]

割り込みの受け付けを許可（ネスト管理なし）

[発行有効範囲]

ブート処理，タスク，割り込みサービス・ルーチン，アラーム・コールバック，共通フック・ルーチン，固有フック・ルーチン，クリティカル・セクション

[指定形式]

```
void EnableAllInterrupts ( void );
```

[引数]

なし

[機能]

DisableAllInterrupts の発行により禁止されていた割り込みの受け付けを許可します。

備考 1. 本システム・サービスでは，割り込みの受け付け許可処理として，プログラム・ステータス・ワード（PSW）の ID ビットに対する操作を行います。本システム・サービスを発行した際には，PSW の ID ビットを，DisableAllInterrupts 発行前の値に戻します。

備考 2. 本システム・サービスを発行した際には，DisableAllInterrupts の発行により開始されたクリティカル・セクションが終了します。

備考 3. 本システム・サービスを共通フック・ルーチン（ProtectionHook）から発行した場合，何も処理は行われず，エラーとしても扱われません。

備考 4. 本システム・サービスを SuspendAllInterrupts の発行から ResumeAllInterrupts が発行されるまでの間に発行した場合，PSW の ID ビットに対する操作は行われません。

備考 5. 本システム・サービスをブート処理（StartOS の発行前）から発行する場合，本システム・サービスの発行前に _kernel_fv0_InitializeIntService の発行が必要となります。

備考 6. AUTOSAR 仕様では，スケーラビリティ・クラス “OsScalabilityClass” がスケーラビリティ・クラス 3（SC3）／スケーラビリティ・クラス 4（SC4）の場合，本システム・サービスの発行前に DisableAllInterrupts が発行されていなかった際には，何も処理は行わず，エラーとしても扱わない旨の規定（OS092）が行われていますが，RV850 では，スケーラビリティ・クラス “OsScalabilityClass” の定義内容に依存することなく，本システム・サービスの発行前に DisableAllInterrupts が発行されていなかった際には，何も処理は行わず，エラーとしても扱いません。

[戻り値]

なし

DisableAllInterrupts

[概要]

割り込みの受け付けを禁止（ネスト管理なし）

[発行有効範囲]

ブート処理，タスク，割り込みサービス・ルーチン，アラーム・コールバック，共通フック・ルーチン，固有フック・ルーチン，クリティカル・セクション

[指定形式]

```
void DisableAllInterrupts ( void );
```

[引数]

なし

[機能]

割り込みの受け付けを禁止します。

備考 1. 本システム・サービスでは，割り込みの受け付け禁止処理として，プログラム・ステータス・ワード（PSW）の ID ビットに対する操作を行います。

備考 2. 本システム・サービスを発行した際には，クリティカル・セクションが開始します。

備考 3. 本システム・サービスを共通フック・ルーチン（ProtectionHook）から発行した場合，何も処理は行われず，エラーとしても扱われません。

備考 4. 本システム・サービスを SuspendAllInterrupts の発行から ResumeAllInterrupts が発行されるまでの間に発行した場合，PSW の ID ビットに対する操作は行われません。

備考 5. 本システム・サービスは EnableAllInterrupts と対で使用することを想定しています。
本システム・サービスの発行後には，同一処理プログラムにおいて，EnableAllInterrupts を発行してください。
本システム・サービスを EnableAllInterrupts の発行前に再発行した場合，何も処理は行われず，エラーとしても扱われません。

備考 6. 処理プログラムから本システム・サービスを発行したにも関わらず，EnableAllInterrupts を発行することなく終了した場合の動作は，スケーラビリティ・クラス “OsScalabilityClass” の定義内容により，以下のように異なります。

- スケーラビリティ・クラス 1（SC1）
  動作は，保証されません。
- スケーラビリティ・クラス 3（SC3）
  処理プログラムの種類により，以下のように異なります。

【タスクの場合】
割り込みの受け付け許可処理（PSW の ID ビットに対する操作），クリティカル・セクションの終了を行います。
共通フック・ルーチン（ErrorHook），固有フック・ルーチン（ErrorHook_*OsApplication*）が登録されていた場合，E_OS_MISSINGEND（0x14）を引数としたフック・ルーチンの呼び出しが行われます。

【割り込みサービス・ルーチン（カテゴリ 2）の場合】
割り込みの受け付け許可処理（PSW の ID ビットに対する操作），クリティカル・セクションの終了を行います。

共通フック・ルーチン（ErrorHook），固有フック・ルーチン（ErrorHook_*OsApplication*）が登録されていた場合，E_OS_DISABLEDINT（0x15）を引数としたフック・ルーチンの呼び出しが行われます。

【固有フック・ルーチン（StartupHook_*OsApplication*）の場合】
割り込みの受け付け許可処理（PSW の ID ビットに対する操作），クリティカル・セクションの終了を行います。

【固有フック・ルーチン（ShutdownHook_*OsApplication*）の場合】
クリティカル・セクションの終了を行います。

【上記以外の場合】
動作は，保証されません。

AUTOSAR 仕様では，処理プログラムが固有フック・ルーチンの場合の動作が規定されていませんが，RV850 では，処理プログラムが StartupHook_*OsApplication*，ShutdownHook_*OsApplication* の場合には上記操作を行います。

備考 7. 本システム・サービスをブート処理（StartOS の発行前）から発行する場合，本システム・サービスの発行前に _kernel_fv0_InitializeIntService の発行が必要となります。

［戻り値］

なし

ResumeAllInterrupts

[概要]

割り込みの受け付けを許可（ネスト管理あり）

[発行有効範囲]

ブート処理，タスク，割り込みサービス・ルーチン，アラーム・コールバック，共通フック・ルーチン，固有フック・ルーチン，クリティカル・セクション

[指定形式]

```
void ResumeAllInterrupts ( void );
```

[引数]

なし

[機能]

禁止要求カウンタ（本システム・サービス／ SuspendAllInterrupts 専用）から 0x1 を減算します。
禁止要求カウンタの減算結果が 0x0 の場合，禁止要求カウンタの減算処理（禁止要求カウンタから 0x1 を減算）と伴に，SuspendAllInterrupts の発行により禁止されていた割り込みの受け付けを許可します。

備考 1. 本システム・サービスでは，割り込みの受け付け許可処理として，プログラム・ステータス・ワード（PSW）の ID ビットに対する操作を行います。本システム・サービスを発行した際には，PSW の ID ビットを，SuspendAllInterrupts 発行前の値に戻します。

備考 2. 禁止要求カウンタの減算結果が 0x0 の場合，SuspendAllInterrupts の発行により開始されたクリティカル・セクションが終了します。

備考 3. 本システム・サービスを共通フック・ルーチン（ProtectionHook）から発行した場合，何も処理は行われず，エラーとしても扱われません。

備考 4. 本システム・サービスを DisableAllInterrupts の発行から EnableAllInterrupts が発行されるまでの間に発行した場合，PSW の ID ビットに対する操作は行われません。

備考 5. 本システム・サービスをブート処理（StartOS の発行前）から発行する場合，本システム・サービスの発行前に _kernel_fv0_InitializeIntService の発行が必要となります。

備考 6. AUTOSAR 仕様では，スケーラビリティ・クラス “OsScalabilityClass” がスケーラビリティ・クラス 3（SC3）／スケーラビリティ・クラス 4（SC4）の場合，本システム・サービスの発行前に SuspendAllInterrupts が発行されていなかった際には，何も処理は行わず，エラーとしても扱わない旨の規定（OS092）が行われていますが，RV850 では，スケーラビリティ・クラス “OsScalabilityClass” の定義内容に依存することなく，本システム・サービスの発行前に SuspendAllInterrupts が発行されていなかった際には，何も処理は行わず，エラーとしても扱いません。

[戻り値]

なし

SuspendAllInterrupts

[概要]

割り込みの受け付けを禁止（ネスト管理あり）

[発行有効範囲]

ブート処理，タスク，割り込みサービス・ルーチン，アラーム・コールバック，共通フック・ルーチン，固有フック・ルーチン，クリティカル・セクション

[指定形式]

```
void SuspendAllInterrupts ( void );
```

[引数]

なし

[機能]

禁止要求カウンタ（ResumeAllInterrupts ／本システム・サービス専用，最大禁止要求数：127）に 0x1 を加算します。
本システム・サービスを発行した時点における禁止要求カウンタの値が 0x0 の場合，禁止要求カウンタの加算処理（禁止要求カウンタに 0x1 を加算）と伴に，割り込みの受け付けを禁止します。

備考 1. 本システム・サービスでは，割り込みの受け付け禁止処理として，プログラム・ステータス・ワード（PSW）の ID ビットに対する操作を行います。

備考 2. 本システム・サービスを発行した時点における禁止要求カウンタの値が 0x0 の場合，本システム・サービスの発行によりクリティカル・セクションが開始します。

備考 3. 本システム・サービスを共通フック・ルーチン（ProtectionHook）から発行した場合，何も処理は行われず，エラーとしても扱われません。

備考 4. 本システム・サービスを DisableAllInterrupts の発行から EnableAllInterrupts が発行されるまでの間に発行した場合，PSW の ID ビットに対する操作は行われません。

備考 5. 本システム・サービスは，ResumeAllInterrupts と対で使用することを想定しています。
本システム・サービスの発行後には，同一処理プログラムにおいて，ResumeAllInterrupts を発行してください。
本システム・サービスを ResumeAllInterrupts の発行前に再発行した場合，禁止要求カウンタの加算処理（禁止要求カウンタに 0x1 を加算）のみを行い，割り込みの受け付け禁止処理（PSW の ID ビットに対する操作）は行いません。
本システム・サービスの再発行に伴い，禁止要求カウンタ（最大禁止要求数：127）がオーバフローした場合，何も処理は行われず，エラーとしても扱われません。

備考 6. 処理プログラムから本システム・サービスを発行したにも関わらず，ResumeAllInterrupts を発行することなく終了した場合の動作は，スケーラビリティ・クラス “OsScalabilityClass” の定義内容により，以下のように異なります。

- スケーラビリティ・クラス 1（SC1）
  動作は，保証されません。
- スケーラビリティ・クラス 3（SC3）
  処理プログラムの種類により，以下のように異なります。

【タスクの場合】
割り込みの受け付け許可処理（PSW の ID ビットに対する操作），禁止要求カウンタのクリア処理（禁止要求カウンタに 0x0 を設定），クリティカル・セクションの終了を行います。

共通フック・ルーチン（ErrorHook），固有フック・ルーチン（ErrorHook_*OsApplication*）が登録されていた場合，E_OS_MISSINGEND（0x14）を引数としたフック・ルーチンの呼び出しが行われます。

【割り込みサービス・ルーチン（カテゴリ 2）の場合】
割り込みの受け付け許可処理（PSW の ID ビットに対する操作），禁止要求カウンタのクリア処理（禁止要求カウンタに 0x0 を設定），クリティカル・セクションの終了を行います。
共通フック・ルーチン（ErrorHook），固有フック・ルーチン（ErrorHook_*OsApplication*）が登録されていた場合，E_OS_DISABLEDINT（0x15）を引数としたフック・ルーチンの呼び出しが行われます。

【固有フック・ルーチン（StartupHook_*OsApplication*）の場合】
割り込みの受け付け許可処理（PSW の ID ビットに対する操作），禁止要求カウンタのクリア処理（禁止要求カウンタに 0x0 を設定），クリティカル・セクションの終了を行います。

【固有フック・ルーチン（ShutdownHook_*OsApplication*）の場合】
クリティカル・セクションの終了を行います。

【上記以外の場合】
動作は，保証されません。

AUTOSAR 仕様では，処理プログラムが固有フック・ルーチンの場合の動作が規定されていませんが，RV850 では，処理プログラムが StartupHook_*OsApplication*，ShutdownHook_*OsApplication* の場合には上記操作を行います。

備考 7. 本システム・サービスをブート処理（StartOS の発行前）から発行する場合，本システム・サービスの発行前に _kernel_fv0_InitializeIntService の発行が必要となります。

［戻り値］

なし

ResumeOSInterrupts

[概要]

カテゴリ 2 割り込みの受け付けを許可（ネスト管理あり）

[発行有効範囲]

タスク，割り込みサービス・ルーチン，アラーム・コールバック，共通フック・ルーチン，固有フック・ルーチン，クリティカル・セクション

[指定形式]

```
void ResumeOSInterrupts ( void );
```

[引数]

なし

[機能]

禁止要求カウンタ（本システム・サービス／ SuspendOSInterrupts 専用）から 0x1 を減算します。
禁止要求カウンタの減算結果が 0x0 の場合，禁止要求カウンタの減算処理（禁止要求カウンタから 0x1 を減算）と伴に，SuspendOSInterrupts の発行により禁止されていた割り込みの受け付けを許可します。

備考 1. 本システム・サービスでは，割り込みの受け付け許可処理として，プライオリティ・マスク・レジスタ（PMR）の PM*n* ビットに対する操作を行います。
操作対象となる PM*n* ビットは，以下に示した優先度に対応したビット，および以下に示した優先度よりも低い優先度に対応した対応したビットとなります。

- 割り込みサービス・ルーチン（カテゴリ 2）用として定義された初期優先度 “OsIsrPriority”
- ハードウエア・カウンタ用として定義された優先度 “OsCounterPriority”

備考 2. 禁止要求カウンタの減算結果が 0x0 の場合，SuspendOSInterrupts の発行により開始されたクリティカル・セクションが終了します。

備考 3. 本システム・サービスを共通フック・ルーチン（ProtectionHook）から発行した場合，何も処理は行われず，エラーとしても扱われません。

備考 4. 本システム・サービスをアラーム・コールバック，共通フック・ルーチン（StartupHook，ShutdownHook，PostTaskHook，PreTaskHook，ErrorHook），固有フック・ルーチンから発行した場合，禁止要求カウンタの減算処理（禁止要求カウンタから 0x1 を減算）のみを行い，割り込みの受け付け許可処理（PMR の PM*n* ビットに対する操作）は行いません。

備考 5. SuspendOSInterrupts を発行する以前に GetResource（シーリング値：INTPRI*x*）の発行が行われていた場合，本システム・サービスを発行した際のプライオリティ・マスク・レジスタ（PMR）の内容は，SuspendOSInterrupts を発行する以前の値（INTPRI0 ～ INTPRI*x* に対応した割り込み要因の受け付けが禁止された状態）となります。

備考 6. AUTOSAR 仕様では，スケーラビリティ・クラス “OsScalabilityClass” がスケーラビリティ・クラス 3（SC3）／スケーラビリティ・クラス 4（SC4）の場合，本システム・サービスの発行前に SuspendOSInterrupts が発行されていなかった際には，何も処理は行わず，エラーとしても扱わない旨の規定（OS092）が行われていますが，RV850 では，スケーラビリティ・クラス “OsScalabilityClass” の定義内容に依存することなく，本システム・サービスの発行前に SuspendOSInterrupts が発行されていなかった際には，何も処理は行わず，エラーとしても扱いません。

[戻り値]

なし

SuspendOSInterrupts

[概要]

カテゴリ 2 割り込みの受け付けを禁止（ネスト管理あり）

[発行有効範囲]

タスク，割り込みサービス・ルーチン，アラーム・コールバック，共通フック・ルーチン，固有フック・ルーチン，クリティカル・セクション

[指定形式]

```
void SuspendOSInterrupts ( void );
```

[引数]

なし

[機能]

禁止要求カウンタ（ResumeOSInterrupts ／本システム・サービス専用，最大禁止要求数：127）に 0x1 を加算します。
本システム・サービスを発行した時点における禁止要求カウンタの値が 0x0 の場合，禁止要求カウンタの加算処理（禁止要求カウンタに 0x1 を加算）と伴に，割り込みの受け付けを禁止します。

備考 1. 本システム・サービスでは，割り込みの受け付け禁止処理として，プライオリティ・マスク・レジスタ（PMR）の PM*n* ビットに対する操作を行います。
操作対象となる PM*n* ビットは，以下に示した優先度に対応したビット，および以下に示した優先度よりも低い優先度に対応したビットとなります。

- 割り込みサービス・ルーチン（カテゴリ 2）用として定義された初期優先度 “OsIsrPriority”
- ハードウエア・カウンタ用として定義された優先度 “OsCounterPriority”

備考 2. 本システム・サービスを発行した時点における禁止要求カウンタの値が 0x0 の場合，本システム・サービスの発行によりクリティカル・セクションが開始します。

備考 3. 本システム・サービスを共通フック・ルーチン（ProtectionHook）から発行した場合，何も処理は行われず，エラーとしても扱われません。

備考 4. 本システム・サービスをアラーム・コールバック，共通フック・ルーチン（StartupHook，ShutdownHook，PostTaskHook，PreTaskHook，ErrorHook），固有フック・ルーチンから発行した場合，禁止要求カウンタの加算処理（禁止要求カウンタに 0x1 を加算）のみを行い，割り込みの受け付け禁止処理（PMR の PM*n* ビットに対する操作）は行いません。

備考 5. 本システム・サービスは，ResumeOSInterrupts と対で使用することを想定しています。
本システム・サービスの発行後には，ResumeOSInterrupts を発行してください。
本システム・サービスを ResumeOSInterrupts の発行前に再発行した場合，禁止要求カウンタの加算処理（禁止要求カウンタに 0x1 を加算）のみを行い，割り込みの受け付け禁止処理（PMR の PM*n* ビットに対する操作）は行いません。
本システム・サービスの再発行に伴い，禁止要求カウンタ（最大禁止要求数：127）がオーバフローした場合，何も処理は行われず，エラーとしても扱われません。

備考 6. 処理プログラムから本システム・サービスを発行したにも関わらず，ResumeOSInterrupts を発行することなく終了した場合の動作は，スケーラビリティ・クラス “OsScalabilityClass” の定義内容により，以下のように異なります。

- スケーラビリティ・クラス 1（SC1）
  動作は，保証されません。

- スケーラビリティ・クラス 3（SC3）
  処理プログラムの種類により，以下のように異なります。

  【タスクの場合】
  割り込みの受け付け許可処理（PMR の PM*n* ビットに対する操作），禁止要求カウンタのクリア処理（禁止要求カウンタに 0x0 を設定），クリティカル・セクションの終了を行います。
  共通フック・ルーチン（ErrorHook），固有フック・ルーチン（ErrorHook_*OsApplication*）が登録されていた場合，E_OS_MISSINGEND（0x14）を引数としたフック・ルーチンの呼び出しが行われます。

  【割り込みサービス・ルーチン（カテゴリ 2）の場合】
  割り込みの受け付け許可処理（PMR の PM*n* ビットに対する操作），禁止要求カウンタのクリア処理（禁止要求カウンタに 0x0 を設定），クリティカル・セクションの終了を行います。
  共通フック・ルーチン（ErrorHook），固有フック・ルーチン（ErrorHook_*OsApplication*）が登録されていた場合，E_OS_DISABLEDINT（0x15）を引数としたフック・ルーチンの呼び出しが行われます。

  【固有フック・ルーチン（StartupHook_*OsApplication*）の場合】
  割り込みの受け付け許可処理（PMR の PM*n* ビットに対する操作），禁止要求カウンタのクリア処理（禁止要求カウンタに 0x0 を設定），クリティカル・セクションの終了を行います。

  【固有フック・ルーチン（ShutdownHook_*OsApplication*）の場合】
  クリティカル・セクションの終了を行います。

  【上記以外の場合】
  動作は，保証されません。

AUTOSAR 仕様では，処理プログラムが固有フック・ルーチンの場合の動作が規定されていませんが，RV850 では，処理プログラムが StartupHook_*OsApplication*，ShutdownHook_*OsApplication* の場合には上記操作を行います。

［戻り値］

なし

14.4.3 リソース管理

以下に，RV850 が提供しているリソース管理用システム・サービスを示します。

表 14.19 リソース管理用システム・サービス

システム・サービス名	機能概要
GetResource	リソースの獲得
ReleaseResource	リソースの解放

GetResource

[概要]

リソースの獲得

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2）

[指定形式]

```
StatusaType GetResource ( ResourceType ResID );
```

[引数]

I/O	引数	説明
I	`ResourceType ResID;`	リソース識別子

[機能]

対象リソース（引数 *ResID* で指定されたリソース）を獲得します。
なお，本システム・サービスでは，リソースの獲得処理（本システム・サービスの発行から ReleaseResource を発行するまでの間，対象リソースを占有）と伴に，現在優先度の変更処理が行われます。

備考 1. 現在優先度の変更処理は，本システム・サービスを発行した処理プログラムの種別により，以下のように異なります。

- タスク
  自タスクの現在優先度を該当リソースのシーリング値 “OsResourcePriority” に変更します。
  自タスクは，現在優先度の変更に伴い，変更後の優先度に応じたレディ・キューの先頭へと再キューイングされます。
  リソースのシーリング値が INTPRIx の場合，自タスクの現在優先度を最高優先度（29）へと変更すると伴に，割り込みの受け付け禁止処理（優先度 “INTPRI0 ～シーリング値 ” に対応した割り込み要因の受け付けを禁止）が行われます。
- 割り込みサービス・ルーチン（カテゴリ 2）
  割り込みサービス・ルーチンの現在優先度を該当リソースのシーリング値 “OsResourcePriority” に変更すると伴に，割り込みの受け付け禁止処理（優先度 “INTPRI0 ～シーリング値 ” に対応した割り込み要因の受け付けを禁止）が行われます。

備考 2. 処理プログラムが複数のリソースを獲得した際には，リソースの解放を last-in-first-out（最後に獲得したリソースを最初に解放）の順序で行う必要があります。

備考 3. 獲得したリソースのシーリング値が本システム・サービスを発行した処理プログラムの初期優先度（初期優先度 “OsTaskPriority”，または初期優先度 “OsIsrPriority”）よりも高く，本システム・サービスを発行した処理プログラムの現在優先度よりも低い場合，現在優先度の変更処理，および割り込みの受け付け禁止処理は行われません。
したがって，シーリング値が 10 のリソースを獲得済みのタスク（初期優先度：1）は，シーリング値が 5 のリソースを獲得しても，現在優先度は 10 のままとなります。

備考 4. 割り込みサービス・ルーチン（カテゴリ 2）から本システム・サービスを発行したにも関わらず，ReleaseResource を発行することなく割り込みサービス・ルーチンの処理が終了した場合の動作は，スケーラビリティ・クラス “OsScalabilityClass” の定義内容により，以下のように異なります。

- スケーラビリティ・クラス 1（SC1）
  動作は，保証されません。

- スケーラビリティ・クラス 3（SC3）
  割り込みサービス・ルーチンからの復帰処理時にリソースの解放処理（本システム・サービスの発行から ReleaseResource を発行するまでの間，占有していた対象リソースを解放）と伴に，現在優先度の変更処理（割り込みサービス・ルーチンの現在優先度を初期優先度 “OsIsrPriority” に戻す）が行われます。
  割り込みサービス・ルーチンに共通フック・ルーチン（ErrorHook），固有フック・ルーチン（ErrorHook_*OsApplication*）が登録されていた場合，リソースの解放処理，現在優先度の変更処理と伴に，E_OS_RESOURCE（0x6）を引数としたフック・ルーチンの呼び出しが行われます。

［戻り値］

マクロ	数値	説明
E_OK	0x0	正常終了
E_OS_ACCESS	0x1	異常終了 - 既に他処理プログラムが対象リソースを獲得している - 対象リソースのシーリング値が本システム・サービスを発行した処理プログラムの初期優先度（初期優先度 “OsTaskPriority”，または初期優先度 “OsIsrPriority”）よりも低い - 本システム・サービスを発行した処理プログラムが所属している OS アプリケーションは，対象リソースに対するアクセス権がない（SC3 限定）
E_OS_CALLEVEL	0x2	発行有効範囲外の処理プログラムから発行した
E_OS_ID	0x3	引数 *ResID* の指定が不正である
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した

ReleaseResource

[概要]

リソースの解放

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2）

[指定形式]

```
StatusaType ReleaseResource ( ResourceType ResID );
```

[引数]

I/O	引数	説明
I	`ResourceType` *ResID*`;`	リソース識別子

[機能]

対象リソース（引数 *ResID* で指定されたリソース）を解放します。
なお，本システム・サービスでは，リソースの解放処理（GetResource の発行から本システム・サービスを発行するまでの間，占有していた対象リソースを解放）と伴に，現在優先度の変更処理が行われます。

備考 1. 現在優先度の変更処理は，本システム・サービスを発行した処理プログラムの種別により，以下のように異なります。

- タスク
  自タスクの現在優先度を該当リソースを獲得する以前に設定されていた優先度に戻します。
  自タスクは，現在優先度の変更に伴い，変更後の優先度に応じたレディ・キューの先頭へと再キューイングされます。
  リソースのシーリング値が INTPRIx の際には，自タスクの現在優先度を，該当リソースを獲得する以前に設定されていた優先度へと戻すと伴に，割り込みの受け付け許可処理（優先度 “INTPRI0 ～シーリング値 ” に対応した割り込み要因の受け付けを許可）が行われます。
  本システム・サービスをスケジューリング属性 “OsTaskSchedule” がプリエンプティブ属性（FULL）のタスクから発行した場合，現在優先度の変更処理が完了した際には，スケジューラの起動が行われます。
- 割り込みサービス・ルーチン（カテゴリ 2）
  割り込みサービス・ルーチンの現在優先度を該当リソースを獲得する以前に設定されていた優先度に戻すと伴に，割り込みの受け付け許可処理（優先度 “INTPRI0 ～シーリング値 ” に対応した割り込み要因の受け付けを許可）が行われます。

備考 2. 処理プログラムが複数のリソースを獲得した際には，リソースの解放を last-in-first-out（最後に獲得したリソースを最初に解放）の順序で行う必要があります。

[戻り値]

マクロ	数値	説明
E_OK	0x0	正常終了

マクロ	数値	説明
E_OS_ACCESS	0x1	異常終了 - 他処理プログラムが対象リソースを獲得している - 本システム・サービスを発行した処理プログラムが所属しているOSアプリケーションは，対象リソースに対するアクセス権がない（SC3限定）
E_OS_CALLEVEL	0x2	発行有効範囲外の処理プログラムから発行した
E_OS_ID	0x3	引数 *ResID* の指定が不正である
E_OS_NOFUNC	0x5	異常終了 - リソースの解放順序が不正である - 引数 *ResID* に指定したリソースは，いずれの処理プログラムも未獲得である
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した

14.4.4 イベント管理

以下に，RV850 が提供しているイベント管理用システム・サービスを示します。

表 14.20 イベント管理用システム・サービス

システム・サービス名	機能概要
SetEvent	イベント・マスクの設定
ClearEvent	イベント・マスクのクリア
GetEvent	イベント・マスクの獲得
WaitEvent	イベント・マスクの確認（WAITING 状態への遷移あり）

SetEvent

[概要]

イベント・マスクの設定

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2）

[指定形式]

```
StatusType SetEvent ( TaskType TaskID, EventMaskType Mask );
```

[引数]

I/O	引数	説明
I	TaskType *TaskID*;	タスク識別子
I	EventMaskType *Mask*;	設定するイベント・マスク

[機能]

対象タスク（引数 *TaskID* で指定された拡張タスク）に割り付けられているイベント（32 ビット幅）に引数 *Mask* で指定されたイベント・マスクを設定します。

備考 1. 本システム・サービスを発行した際，対象タスクに割り付けられているイベントの現在パターンが “B'1100”，引数 *Mask* で指定された設定パターンが “B'1010” であった場合には，イベント・マスクの設定処理に伴い，該当イベントの現在パターンは “B'1110” となります。

備考 2. 対象タスクに複数のイベントが割り付けられていた場合，各イベントに対してイベント・マスクの設定処理が行われます。

備考 3. イベント・マスクの設定結果が対象タスク（WAITING 状態）の要求パターンを満足した場合，イベント・マスクの設定処理と伴に，タスクの状態操作処理（対象タスクを WAITING 状態から READY 状態へと遷移）が行われます。
要求パターンの条件成立は，“ イベント・マスクの設定結果 ” と “ 対象タスクの要求パターン（対象タスクが WaitEvent を発行した際，引数 *Mask* で指定したイベント・マスク）” の比較を行い，1 ビットでも一致（設定結果が B'0001，要求パターンが B'1111 など）した場合となります。
対象タスクは，WAITING 状態から READY 状態への遷移に伴い，優先度に応じたレディ・キューの最後尾にキューイングされます。
本システム・サービスをスケジューリング属性 “OsTaskSchedule” がプリエンプティブ属性（FULL）のタスクから発行した場合，タスクの状態操作処理（対象タスクを WAITING 状態から READY 状態へと遷移）が完了した際には，スケジューラの起動が行われます。

[戻り値]

マクロ	数値	説明
E_OK	0x0	正常終了

マクロ	数値	説明
E_OS_ACCESS	0x1	異常終了 - 対象タスクが基本タスクである - 本システム・サービスを発行した処理プログラムが所属しているOSアプリケーションは，対象タスクに対するアクセス権がない（SC3限定） - 対象タスクの所属しているOSアプリケーションがAPPLICATION_RESTARTING状態，またはAPPLICATION_TERMINATED状態である（SC3限定）
E_OS_CALLEVEL	0x2	発行有効範囲外の処理プログラムから発行した
E_OS_ID	0x3	引数 *TaskID* の指定が不正である
E_OS_STATE	0x7	対象タスクがSUSPENDED状態である
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した

ClearEvent

[概要]

イベント・マスクのクリア

[発行有効範囲]

タスク（拡張）

[指定形式]

```
StatusType ClearEvent ( EventMaskType Mask );
```

[引数]

I/O	引数	説明
I	`EventMaskType` *Mask*;	クリアするイベント・マスク

[機能]

自タスク（本システム・サービスを発行した拡張タスク）に割り付けられているイベント（32 ビット幅）の現在パターンを引数 *Mask* で指定されたイベント・マスクでクリアします。

備考 1. 本システム・サービスを発行した際，自タスクに割り付けられているイベントの現在パターンが “B'1100”，引数 *Mask* で指定されたクリア・パターンが “B'1010” であった場合には，イベント・マスクのクリア処理に伴い，該当イベントの現在パターンは “B'0100” となります。

備考 2. 自タスクに複数のイベントが割り付けられていた場合，各イベントに対してイベント・マスクのクリア処理が行われます。

[戻り値]

マクロ	数値	説明
E_OK	0x0	正常終了
E_OS_ACCESS	0x1	基本タスクから発行した
E_OS_CALLEVEL	0x2	発行有効範囲外の処理プログラムから発行した
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した

GetEvent

[概要]

イベント・マスクの獲得

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2），共通フック・ルーチン（PostTaskHook，PreTaskHook，ErrorHook），固有フック・ルーチン（ErrorHook_*OsApplication*）

[指定形式]

```
StatusType GetEvent ( TaskType TaskID, EventMaskRefType Event );
```

[引数]

I/O	引数	説明
I	`TaskType` *TaskID*;	タスク識別子
O	`EventMaskRefType` *Event*;	獲得したイベント・マスクを格納する領域へのポインタ

[機能]

対象タスク（引数 *TaskID* で指定されたタスク）に割り付けられているイベントの現在パターンを獲得し，引数 *Event* で指定された領域に格納します。

備考　対象タスクに複数のイベントが割り付けられていた場合，各イベントの現在パターンの論理和を引数 *Event* で指定された領域に格納します。

[戻り値]

マクロ	数値	説明
E_OK	0x0	正常終了
E_OS_ACCESS	0x1	異常終了 - 対象タスクが基本タスクである - 本システム・サービスを発行した処理プログラムが所属しているOSアプリケーションは，対象タスクに対するアクセス権がない（SC3限定） - 対象タスクの所属しているOSアプリケーションがAPPLICATION_RESTARTING状態，またはAPPLICATION_TERMINATED状態である（SC3限定）
E_OS_CALLEVEL	0x2	発行有効範囲外の処理プログラムから発行した
E_OS_ID	0x3	引数 *TaskID* の指定が不正である
E_OS_STATE	0x7	対象タスクSUSPENDED状態である
E_OS_PARAM_POINTER	0x12	引数 *Event* の指定が不正（NULLポインタ）である

マクロ	数値	説明
E_OS_ILLEGAL_ADDRESS	0x13	本システム・サービスを発行した処理プログラムは，引数 *Event* で指定された領域に対するアクセス権がない（SC3 限定）
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した

WaitEvent

[概要]

イベント・マスクの確認（WAITING 状態への遷移あり）

[発行有効範囲]

タスク（拡張）

[指定形式]

```
StatusType WaitEvent ( EventMaskType Mask );
```

[引数]

I/O	引数	説明
I	EventMaskType *Mask*;	要求パターン

[機能]

自タスク（本システム・サービスを発行したタスク）に割り付けられているイベントの現在パターンが引数 *Mask* で指定された要求パターンの条件を満足しているか否かを確認します。

備考 1. 自タスクに複数のイベントが割り付けられていた場合，各イベントの現在パターンの論理和に対してイベント・マスクの確認処理が行われます。

備考 2. 要求パターンの条件成立は，“ 自タスクに割り付けられているイベントの現在パターン ” と “ 引数 *Mask* で指定した要求パターン ” の比較を行い，1 ビットでも一致（現在パターンが B'0001，要求パターンが B'1111 など）した場合となります。
したがって，自タスクのイベントの現在パターンが B'1000，要求パターンが B'0111 の際には，条件不成立となります。

備考 3. イベント・マスクの確認結果に伴い，本システム・サービスの処理内容が以下のように異なります。

- 条件成立
  本システム・サービスの戻り値として，E_OK（0x0）が返され，自タスクは RUNNING 状態のままとなります。
- 条件不成立
  自タスクを RUNNING 状態から WAITING 状態へと遷移させます。
  自タスクは，RUNNING 状態から WAITING 状態への遷移に伴い，優先度に応じたレディ・キューから外れます。
  タスクの状態操作処理（自タスクを RUNNING 状態から WAITING 状態へと遷移）が完了した際には，スケジューラの起動が行われます。

備考 4. 自タスクが “ インターナル・リソース ” を獲得していた際には，イベント・マスクの確認結果が条件不成立の場合に限り，タスクの状態操作処理（自タスクを RUNNING 状態から WAITING 状態へと遷移）と伴に，インターナル・リソースの解放処理，および現在優先度の変更処理（自タスクの現在優先度を，該当リソースを獲得する以前に設定されていた優先度に戻す）を行ったのち，スケジューラを起動します。インターナル・リソースのシーリング値が INTPRI*x* の際には，割り込みの受け付け許可処理（優先度 “INTPRI0 ～シーリング値 ” に対応した割り込み要因の受け付けを許可）も行われます。

［戻り値］

マクロ	数値	説明
E_OK	0x0	正常終了
E_OS_ACCESS	0x1	基本タスクから発行した
E_OS_CALLEVEL	0x2	発行有効範囲外の処理プログラムから発行した
E_OS_RESOURCE	0x6	自タスクが “ 通常のリソース ” を獲得している
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した

14.4.5 カウンタ管理

以下に，RV850 が提供しているカウンタ管理用システム・サービスを示します。

表 14.21 カウンタ管理用システム・サービス

システム・サービス名	機能概要
IncrementCounter	カウント値の更新
GetCounterValue	現在カウント値の獲得
GetElapsedValue	現在カウント値／相対カウント値の獲得

IncrementCounter

[概要]

カウント値の更新

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2）

[指定形式]

```
StatusType IncrementCounter ( CounterType CounterID );
```

[引数]

I/O	引数	説明
I	`CounterType` *CounterID*`;`	カウンタ識別子

[機能]

対象カウンタ（引数 *CounterID* で指定されたソフトウエア・カウンタ，単位：tick）に 0x1 を加算します。

備考 1. カウンタの加算結果が対象カウンタに関連付けられているアラーム／スケジュール・テーブルの満了条件が成立した場合，カウンタの加算処理（カウンタに 0x1 を加算）と伴に，アラーム情報／スケジュール・テーブル情報で定義された満了処理（タスクの起動など）が行われます。

備考 2. 本システム・サービスの発行に伴い行われた満了処理（タスクの起動など）でエラー（“ タスクに対する起動要求数が最大起動要求数 “OsTaskActivation” を越えた ” など）が発生した場合，共通フック・ルーチン（ErrorHook），固有フック・ルーチン（ErrorHook_*OsApplication*）の呼び出しが行われますが，本システム・サービスの戻り値は E_OK（0x0）となります。

備考 3. カウンタの加算結果がオーバフローする（最大カウント値 “OsCounterMaxAllowedValue” を越える）場合，エラーとして扱わず，対象カウンタに 0x0 を設定します。

[戻り値]

マクロ	数値	説明
E_OK	0x0	正常終了
E_OS_ACCESS	0x1	異常終了 - 本システム・サービスを発行した処理プログラムが所属している OS アプリケーションは，対象カウンタに対するアクセス権がない（SC3 限定） - 対象カウンタの所属している OS アプリケーションが APPLICATION_RESTARTING 状態，または APPLICATION_TERMINATED 状態である（SC3 限定）
E_OS_CALLEVEL	0x2	発行有効範囲外の処理プログラムから発行した
E_OS_ID	0x3	異常終了 - 引数 *CounterID* の指定が不正である - 対象カウンタがハードウエア・カウンタである

マクロ	数値	説明
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した

GetCounterValue

[概要]

現在カウント値の獲得

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2）

[指定形式]

```
StatusType GetCounterValue ( CounterType CounterID, TickRefType Value );
```

[引数]

I/O	引数	説明
I	`CounterType` *CounterID*;	カウンタ識別子
O	`TickRefType` *Value*;	獲得した現在カウント値を格納する領域へのポインタ

[機能]

対象カウンタ（引数 *CounterID* で指定されたカウンタ）の現在カウント値（単位：tick）を獲得し，引数 *Value* で指定された領域に格納します。

備考　AUTOSAR 仕様では，引数 *Value* で指定された領域に格納する値は，引数 *CounterID* で指定されたカウンタの種別（ソフトウエア・カウンタ，ハードウエア・カウンタ）に準じた単位とする旨の規定（OS531）が行われていますが，RV850 では，種別に関係なく，単位を tick で統一しています。

[戻り値]

マクロ	数値	説明
E_OK	0x0	正常終了
E_OS_ACCESS	0x1	異常終了 - 本システム・サービスを発行した処理プログラムが所属している OS アプリケーションは，対象カウンタに対するアクセス権がない（SC3 限定） - 対象カウンタの所属している OS アプリケーションが APPLICATION_RESTARTING 状態，または APPLICATION_TERMINATED 状態である（SC3 限定）
E_OS_CALLEVEL	0x2	発行有効範囲外の処理プログラムから発行した
E_OS_ID	0x3	引数 *CounterID* の指定が不正である
E_OS_PARAM_POINTER	0x12	引数 *Value* の指定が不正（NULL ポインタ）である
E_OS_ILLEGAL_ADDRESS	0x13	本システム・サービスを発行した処理プログラムは，引数 *Value* で指定された領域に対するアクセス権がない（SC3 限定）
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した

GetElapsedValue

[概要]

現在カウント値／相対カウント値の獲得

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2）

[指定形式]

```
StatusType GetElapsedValue ( CounterType CounterID, TickRefType Value, TickRefType
ElapsedValue );
```

[引数]

I/O	引数	説明
I	`CounterType` *CounterID*`;`	カウンタ識別子
I/O	`TickRefType` *Value*`;`	起点カウント値を格納した領域へのポインタ，および現在カウント値を格納する領域へのポインタ
O	`TickRefType` *ElapsedValue*`;`	獲得した相対カウント値を格納する領域へのポインタ

[機能]

対象カウンタ（引数 *CounterID* で指定されたカウンタ）の現在カウント値（単位：tick）を獲得し，引数 *Value* で指定された起点カウント値（単位：tick）と現在カウント値の差分（相対カウント値，単位：tick）を引数 *ElapsedValue* で指定された領域に格納すると伴に，獲得した現在カウント値を引数 *Value* で指定された領域に格納します。

備考 1. 本システム・サービスを発行した際，対象カウンタの現在カウント値が “0x5”，引数 *Value* で指定された起点カウント値が “0x3” であった場合には，引数 *ElapsedValue* で指定された領域に格納される相対カウント値は “0x2”，引数 *Value* で指定された領域に格納される現在カウント値は “0x5” となります。

備考 2. RV850 では，カウンタがオーバフローした（最大カウント値 “OsCounterMaxAllowedValue” を越えた）回数を保持していません。したがって，引数 *Value* で指定された起点カウント値から本システム・サービスを発行するまでの間に対象カウンタがオーバフローした際には，引数 *ElapsedValue* で指定された領域に格納される値は，不正値となります。

備考 3. AUTOSAR 仕様では，引数 *Value*，*ElapsedValue* で指定された領域に格納する値は，引数 *CounterID* で指定されたカウンタの種別（ソフトウエア・カウンタ，ハードウエア・カウンタ）に準じた単位とする旨の規定（OS531）が行われていますが，RV850 では，種別に関係なく，単位を tick で統一しています。

[戻り値]

マクロ	数値	説明
E_OK	0x0	正常終了

マクロ	数値	説明
E_OS_ACCESS	0x1	異常終了 - 本システム・サービスを発行した処理プログラムが所属しているOSアプリケーションは，対象カウンタに対するアクセス権がない（SC3限定） - 対象カウンタの所属しているOSアプリケーションがAPPLICATION_RESTARTING状態，またはAPPLICATION_TERMINATED状態である（SC3限定）
E_OS_CALLEVEL	0x2	発行有効範囲外の処理プログラムから発行した
E_OS_ID	0x3	引数 *CounterID* の指定が不正である
E_OS_VALUE	0x8	引数 *Value* の指定が不正（*Value* ＞ 最大カウント値 “OsCounterMaxAllowedValue”）である
E_OS_PARAM_POINTER	0x12	引数 *Value*，または引数 *ElapsedValue* の指定が不正（NULLポインタ）である
E_OS_ILLEGAL_ADDRESS	0x13	本システム・サービスを発行した処理プログラムは，引数 *Value* で指定された領域，または引数 *ElapsedValue* で指定された領域に対するアクセス権がない（SC3限定）
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した

14.4.6 アラーム管理

以下に，RV850 が提供しているアラーム管理用システム・サービスを示します。

表 14.22 アラーム管理用システム・サービス

システム・サービス名	機能概要
GetAlarmBase	アラーム・ベース情報の獲得
GetAlarm	残りカウント値の獲得
SetRelAlarm	相対アラームの起動
SetAbsAlarm	絶対アラームの起動
CancelAlarm	アラームの終了

GetAlarmBase

[概要]

アラーム・ベース情報の獲得

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2），共通フック・ルーチン（PostTaskHook，PreTaskHook，ErrorHook），固有フック・ルーチン（ErrorHook_*OsApplication*）

[指定形式]

```
StatusType GetAlarmBase ( AlarmType AlarmID, AlarmBaseRefType Info );
```

【アラーム・ベース情報 AlarmBaseRefType】

```
struct _AlarmBaseType {
    TickType maxallowedvalue;  /* 最大カウント値 “OsCounterMaxAllowedValue” */
    TickType mincycle;         /* 最小周期 “OsCounterMinCycle” */
    TickType ticksperbase;     /* 基本カウント値 “OsCounterTicksPerBase” */
};

typedef struct _AlarmBaseType AlarmBaseType;
typedef AlarmBaseType *AlarmBaseRefType;
```

[引数]

I/O	引数	説明
I	`AlarmType` *AlarmID*;	アラーム識別子
O	`AlarmBaseRefType` *Info*;	獲得したアラーム・ベース情報を格納する領域へのポインタ

[機能]

対象アラーム（引数 *AlarmID* で指定されたアラーム）のアラーム・ベース情報（対象アラームが関連付けられているカウンタの最大カウント値 “OsCounterMaxAllowedValue”，最小周期 “OsCounterMinCycle”，基本カウント値 “OsCounterTicksPerBase”）を獲得し，引数 *Info* で指定された領域に格納します。

備考　アラーム・ベース情報 AlarmBaseRefType についての詳細は，「14.3.1 アラーム・ベース情報」を参照してください。

[戻り値]

マクロ	数値	説明
E_OK	0x0	正常終了

マクロ	数値	説明
E_OS_ACCESS	0x1	異常終了 - 本システム・サービスを発行した処理プログラムが所属しているOSアプリケーションは，対象アラームに対するアクセス権がない（SC3限定） - 対象アラームの所属しているOSアプリケーションがAPPLICATION_RESTARTING状態，またはAPPLICATION_TERMINATED状態である（SC3限定）
E_OS_CALLEVEL	0x2	発行有効範囲外の処理プログラムから発行した
E_OS_ID	0x3	引数 *AlarmID* の指定が不正である
E_OS_PARAM_POINTER	0x12	引数 *Info* の指定が不正（NULLポインタ）である
E_OS_ILLEGAL_ADDRESS	0x13	本システム・サービスを発行した処理プログラムは，引数 *Info* で指定された領域に対するアクセス権がない（SC3限定）
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した

GetAlarm

[概要]

残りカウント値の獲得

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2），共通フック・ルーチン（PostTaskHook，PreTaskHook，ErrorHook），固有フック・ルーチン（ErrorHook_*OsApplication*）

[指定形式]

```
StatusType GetAlarm ( AlarmType AlarmID, TickRefType Tick );
```

[引数]

I/O	引数	説明
I	`AlarmType` *AlarmID*;	アラーム識別子
O	`TickRefType` *Tick*;	獲得した残りカウント値を格納する領域へのポインタ

[機能]

対象アラーム（引数 *AlarmID* で指定されたアラーム）の満了条件が成立するまでの残りカウント値（単位：tick）を獲得し，引数 *Tick* で指定された領域に格納します。

備考 1. 本システム・サービスの発行後，引数 *Tick* で指定された領域に格納された値の数だけ対象アラームが関連付けられているカウンタのカウント値が更新された際，対象アラームの満了条件は成立し，満了処理 “OsAlarmAction” で定義された満了処理（タスクの起動など）が行われます。
対象アラームが関連付けられているカウンタがソフトウエア・カウントの際には，本システム・サービスの発行後，IncrementCounter を引数 *Tick* に格納された数だけ発行することにより，対象アラームの満了条件は成立します。

備考 2. 本システム・サービスを割り込みサービス・ルーチン（RV850 が IncrementCounter に関する処理を行っている最中に発生した割り込みにより呼び出された割り込みサービス・ルーチンなど）から発行した際には，獲得した残りカウント値が不正な値となる場合があります。

[戻り値]

マクロ	数値	説明
E_OK	0x0	正常終了
E_OS_ACCESS	0x1	異常終了 - 本システム・サービスを発行した処理プログラムが所属している OS アプリケーションは，対象アラームに対するアクセス権がない（SC3 限定） - 対象アラームの所属している OS アプリケーションが APPLICATION_RESTARTING 状態，または APPLICATION_TERMINATED 状態である（SC3 限定）
E_OS_CALLEVEL	0x2	発行有効範囲外の処理プログラムから発行した
E_OS_ID	0x3	引数 *AlarmID* の指定が不正である

マクロ	数値	説明
E_OS_NOFUNC	0x5	対象アラームが非アクティブ状態である
E_OS_PARAM_POINTER	0x12	引数 *Tick* の指定が不正（NULL ポインタ）である
E_OS_ILLEGAL_ADDRESS	0x13	本システム・サービスを発行した処理プログラムは，引数 *Tick* で指定された領域に対するアクセス権がない（SC3 限定）
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した

SetRelAlarm

[概要]

相対アラームの起動

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2）

[指定形式]

```
StatusType SetRelAlarm ( AlarmType AlarmID, TickType increment, TickType cycle );
```

[引数]

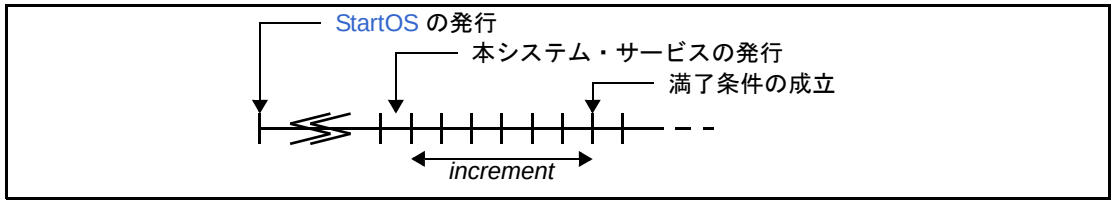
I/O	引数	説明
I	`AlarmType` *AlarmID*;	アラーム識別子
I	`TickType` *increment*;	相対カウント値（単位：tick）
I	`TickType` *cycle*;	周期カウント値（単位：tick）

[機能]

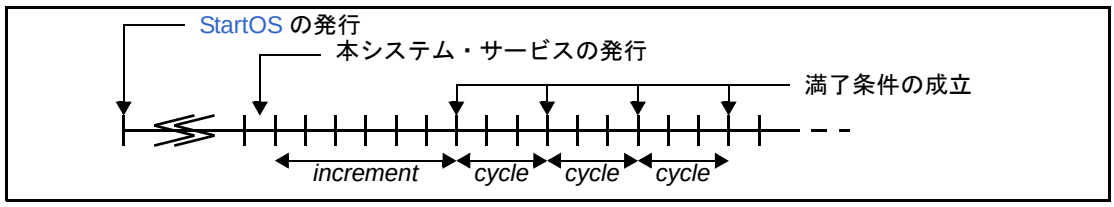
対象アラーム（引数 *AlarmID* で指定された相対アラーム）に引数 *increment*，および *cycle* で指定された満了条件を設定すると伴に，対象アラームを非アクティブ状態からアクティブ状態へと遷移させます。

備考 1. 対象アラームは，引数 *cycle* に 0x0 が指定された場合は“ワンショット・アラーム”として，0x0 以外の値が指定された場合は“周期アラーム”として動作します。

備考 2. 対象アラームがワンショット・アラームとして動作する場合，本システム・サービスの発行後，「“引数 *increment* で指定された値の数”＋1」だけカウント値（対象アラームが関連付けられているカウンタのカウント値）が更新された際に満了処理を実行し，対象アラームはアクティブ状態から非アクティブ状態へと遷移します。
以下に，*increment* = 6 の場合の流れを示します。



備考 3. 対象アラームが周期アラームとして動作する場合，本システム・サービスの発行後，「“引数 *increment* で指定された値の数”＋1」だけカウント値（対象アラームが関連付けられているカウンタのカウント値）が更新された際に 1 回目の満了処理を実行し，その後，引数 *cycle* で指定された値の数だけカウント値（対象アラームが関連付けられているカウンタのカウント値）が更新されるたびに満了処理が繰り返し実行されます。
以下に，*increment* = 6，*cycle* = 3 の場合の流れを示します。



［戻り値］

マクロ	数値	説明
E_OK	0x0	正常終了
E_OS_ACCESS	0x1	異常終了 - 本システム・サービスを発行した処理プログラムが所属している OS アプリケーションは，対象アラームに対するアクセス権がない（SC3 限定） - 対象アラームの所属している OS アプリケーションが APPLICATION_RESTARTING 状態，または APPLICATION_TERMINATED 状態である（SC3 限定）
E_OS_CALLEVEL	0x2	発行有効範囲外の処理プログラムから発行した
E_OS_ID	0x3	引数 *AlarmID* の指定が不正である
E_OS_STATE	0x7	対象アラームがアクティブ状態である
E_OS_VALUE	0x8	異常終了 - 引数 *increment* の指定が不正（*increment* = 0x0）である - 引数 *increment* の指定が不正（*increment* > 最大カウント値 “OsCounterMaxAllowedValue”）である - 引数 *cycle* の指定が不正（*cycle* < 最小周期 “OsCounterMinCycle”）である - 引数 *cycle* の指定が不正（*cycle* > 最大カウント値 “OsCounterMaxAllowedValue”）である
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した

SetAbsAlarm

[概要]

絶対アラームの起動

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2）

[指定形式]

```
StatusType SetAbsAlarm ( AlarmType AlarmID, TickType start, TickType cycle );
```

[引数]

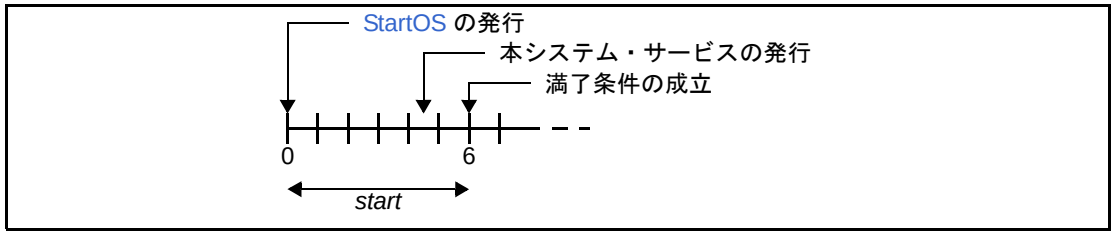
I/O	引数	説明
I	`AlarmType` *AlarmID*;	アラーム識別子
I	`TickType` *start*;	絶対カウント値（単位：tick）
I	`TickType` *cycle*;	周期カウント値（単位：tick）

[機能]

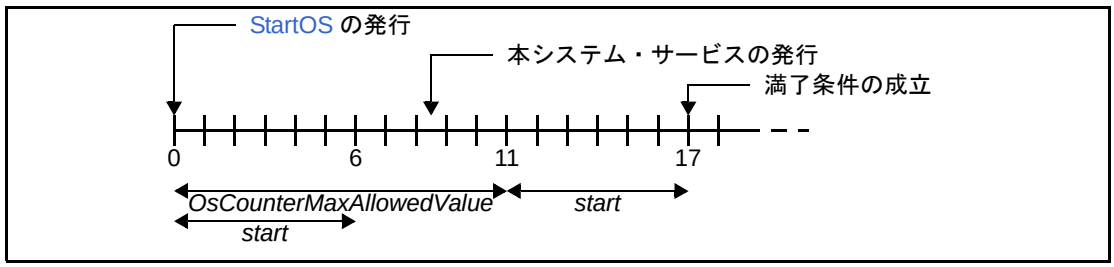
対象アラーム（引数 *AlarmID* で指定された絶対アラーム）に引数 *start*，および *cycle* で指定された満了条件を設定すると伴に，対象アラームを非アクティブ状態からアクティブ状態へと遷移させます。

備考 1. 対象アラームは，引数 *cycle* に 0x0 が指定された場合は “ ワンショット・アラーム ” として，0x0 以外の値が指定された場合は “ 周期アラーム ” として動作します。

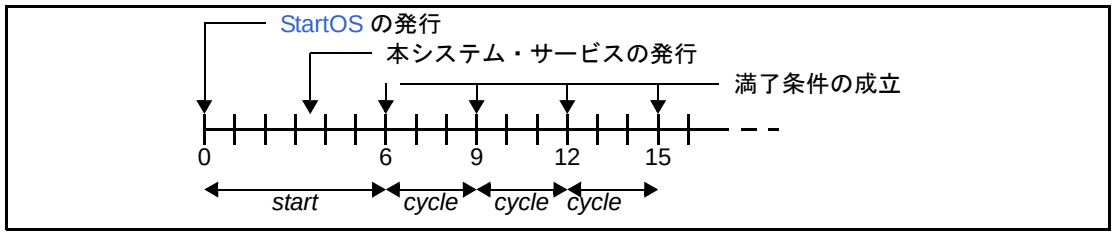
備考 2. 対象アラームがワンショット・アラームとして動作する場合，本システム・サービスの発行後，引数 *start* で指定された絶対カウント値がカウント値（対象アラームが関連付けられているカウンタのカウント値）と一致した際に満了処理を実行し，対象アラームはアクティブ状態から非アクティブ状態へと遷移します。
以下に，*start*= 6 の場合の流れを示します。



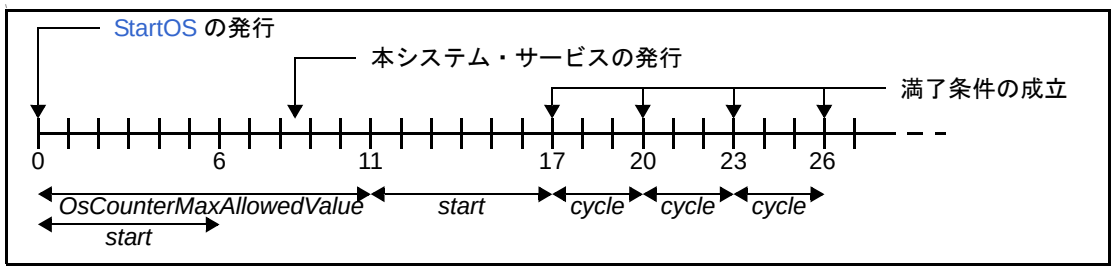
RV850 では，カウンタの加算処理（カウンタに 0x1 を加算）を行った際，カウンタの加算結果がオーバフローする（最大カウント値 “OsCounterMaxAllowedValue” を越える）場合，エラーとして扱わず，対象カウンタに 0x0 を設定します。
したがって，引数 *start* に指定された値がカウント値（対象アラームが関連付けられているカウンタのカウント値）よりも小さい場合，ワンショット・アラームの動作は以下のようになります。
以下に，*start* = 6，*OsCounterMaxAllowedValue* = 11 の場合の流れを示します。



備考 3. 対象アラームが周期アラームとして動作する場合，本システム・サービスの発行後，引数 *start* で指定された絶対カウント値が現在カウント値（対象アラームが関連付けられているカウンタのカウント値）と一致した際に 1 回目の満了処理を実行し，その後，引数 *cycle* で指定された値の数だけカウント値（対象アラームが関連付けられているカウンタのカウント値）が更新されるたびに満了処理が繰り返し実行されます。
以下に，*start* = 6，*cycle* = 3 の場合の流れを示します。



RV850 では，カウンタの加算処理（カウンタに 0x1 を加算）を行った際，カウンタの加算結果がオーバフローする（最大カウント値 “OsCounterMaxAllowedValue” を越える）場合，エラーとして扱わず，対象カウンタに 0x0 を設定します。
したがって，引数 *start* に指定された値がカウント値（対象アラームが関連付けられているカウンタのカウント値）よりも小さい場合，周期アラームの動作は以下のようになります。
以下に，*start* = 6，*cycle* = 3，*OsCounterMaxAllowedValue* = 11 の場合の流れを示します。



[戻り値]

マクロ	数値	説明
E_OK	0x0	正常終了
E_OS_ACCESS	0x1	異常終了 - 本システム・サービスを発行した処理プログラムが所属している OS アプリケーションは，対象アラームに対するアクセス権がない（SC3 限定） - 対象アラームの所属している OS アプリケーションが APPLICATION_RESTARTING 状態，または APPLICATION_TERMINATED 状態である（SC3 限定）
E_OS_CALLEVEL	0x2	発行有効範囲外の処理プログラムから発行した
E_OS_ID	0x3	引数 *AlarmID* の指定が不正である
E_OS_STATE	0x7	対象アラームがアクティブ状態である

マクロ	数値	説明
E_OS_VALUE	0x8	異常終了 - 引数 *start* の指定が不正（*start* > 最大カウント値 “OsCounterMaxAllowedValue”）である - 引数 *cycle* の指定が不正（*cycle* < 最小周期 “OsCounterMinCycle”）である - 引数 *cycle* の指定が不正（*cycle* > 最大カウント値 “OsCounterMaxAllowedValue”）である
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した

CancelAlarm

[概要]

アラームの終了

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2）

[指定形式]

```
StatusType CancelAlarm ( AlarmType AlarmID );
```

[引数]

I/O	引数	説明
I	`AlarmType` *AlarmID*;	アラーム識別子

[機能]

対象アラーム（引数 *AlarmID* で指定されたアラーム）をアクティブ状態から非アクティブ状態へと遷移させます。

備考 1. 非アクティブ状態へと遷移したアラームでは，関連付けられているカウンタのカウント値が更新されても満了条件が成立するまでの残りカウント値の減算処理，および満了条件が成立したか否かの判定処理は行われません。
したがって，非アクティブ状態へと遷移したアラームの満了条件が成立することはありません。

備考 2. 非アクティブ状態へと遷移したアラームは，処理プログラムから SetRelAlarm，または SetAbsAlarm を発行することにより，アクティブ状態へと遷移します。

[戻り値]

マクロ	数値	説明
E_OK	0x0	正常終了
E_OS_ACCESS	0x1	異常終了 - 本システム・サービスを発行した処理プログラムが所属している OS アプリケーションは，対象アラームに対するアクセス権がない（SC3 限定） - 対象アラームの所属している OS アプリケーションが APPLICATION_RESTARTING 状態，または APPLICATION_TERMINATED 状態である（SC3 限定）
E_OS_CALLEVEL	0x2	発行有効範囲外の処理プログラムから発行した
E_OS_ID	0x3	引数 *AlarmID* の指定が不正である
E_OS_NOFUNC	0x5	対象アラームが非アクティブ状態である
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した

14.4.7 スケジュール・テーブル管理

以下に，RV850 が提供しているスケジュール・テーブル管理用システム・サービスを示します。

表 14.23 スケジュール・テーブル管理用システム・サービス

システム・サービス名	機能概要
StartScheduleTableRel	相対スケジュール・テーブルの起動
StartScheduleTableAbs	絶対スケジュール・テーブルの起動
StopScheduleTable	スケジュール・テーブルの終了
NextScheduleTable	スケジュール・テーブルの切り替え
GetScheduleTableStatus	現在状態の獲得

StartScheduleTableRel

[概要]

相対スケジュール・テーブルの起動

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2）

[指定形式]

```
StatusType StartScheduleTableRel ( ScheduleTableType ScheduleTableID, TickType Offset );
```

[引数]

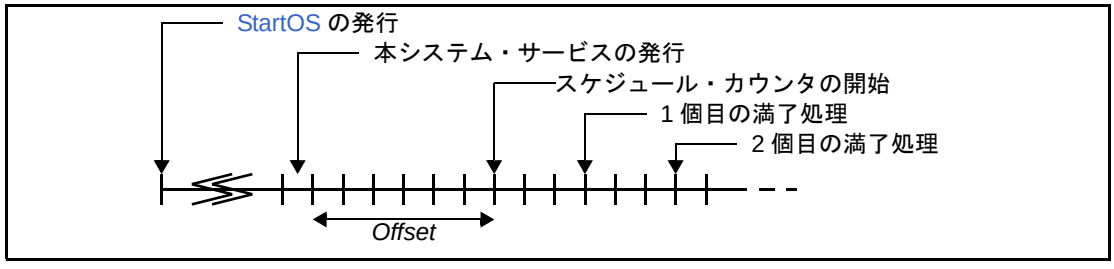
I/O	引数	説明
I	`ScheduleTableType` *ScheduleTableID*;	スケジュール・テーブル識別子
I	`TickType` *Offset*;	相対カウント値

[機能]

対象スケジュール・テーブル（引数 *ScheduleTableID* で指定された相対スケジュール・テーブル）を STOPPED 状態から RUNNING 状態へと遷移させます。

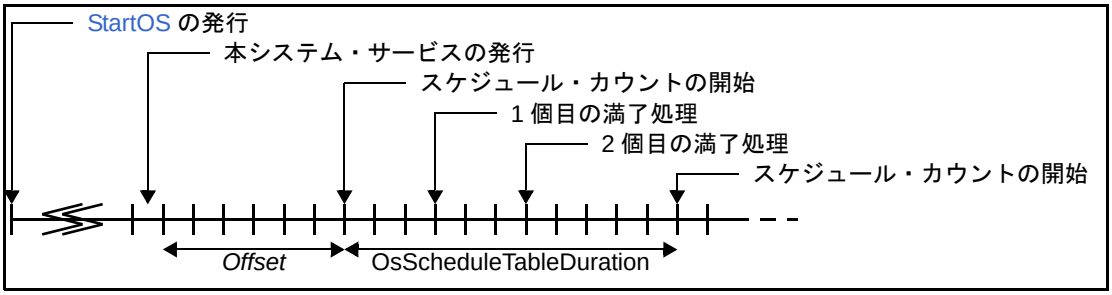
備考 1. 本システム・サービスを周期属性 “OsScheduleTableRepeating” が非周期属性（FALSE）のスケジュール・テーブルに対して発行した場合は “ ワンショット・スケジュール・テーブル ” として，周期属性（TRUE）のスジュール・テーブルに対して発行場合は “ 周期スケジュール・テーブル ” として動作します。

備考 2. 対象スケジュール・テーブルがワンショット・スケジュール・テーブルとして動作する場合，本システム・サービスの発行後，「“ 引数 *Offset* で指定された値の数 ” + 1」だけカウント値（対象スケジュール・テーブルが関連付けられているカウンタのカウント値）が更新された際にスケジュール・カウントが開始され，最終満了処理を実行後，対象スケジュール・テーブルは RUNNING 状態から STOPPED 状態へと遷移します。
以下に，*Offset* = 6，満了処理が 2 個（1 個目の OsScheduleTblExpPointOffset = 3，2 個目の OsScheduleTblExpPointOffset = 6）の場合の流れを示します。



備考 3. 対象スケジュール・テーブルが周期スケジュール・テーブルとして動作する場合，本システム・サービスの発行後，「“ 引数 *Offset* で指定された値の数 ” + 1」だけカウント値（対象スケジュール・テーブルが関連付けられているカウンタのカウント値）が更新された際にスケジュール・カウントが開始され，スケジュール・カウント値 “OsScheduleTableDuration” で定義された値の数だけカウント値が更新されるたびにスケジュール・カウントが繰り返し実行されます。

以下に，*Offset* = 6，OscheduleTableDuration = 11，満了処理が 2 個（1 個目の OsScheduleTblExpPointOffset = 3，2 個目の OsScheduleTblExpPointOffset = 6）の場合の流れを示します。



［戻り値］

マクロ	数値	説明
E_OK	0x0	正常終了
E_OS_ACCESS	0x1	異常終了 - 本システム・サービスを発行した処理プログラムが所属している OS アプリケーションは，対象スケジュール・テーブルに対するアクセス権がない（SC3 限定） - 対象スケジュール・テーブルの所属している OS アプリケーションが APPLICATION_RESTARTING 状態，または APPLICATION_TERMINATED 状態である（SC3 限定）
E_OS_CALLEVEL	0x2	発行有効範囲外の処理プログラムから発行した
E_OS_ID	0x3	引数 *ScheduleTableID* の指定が不正である
E_OS_STATE	0x7	対象スケジュール・テーブルが RUNNING 状態，または NEXT 状態であるスケジュール・カウントの開始
E_OS_VALUE	0x8	異常終了 - 引数 *Offset* の指定が不正（*Offset* ≦ 0x0）である - 引数 *Offset* の指定が不正（*Offset* > 最大カウント値 “OsCounterMaxAllowedValue” - 満了カウント値 “OsScheduleTblExpPointOffset” 注）である
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した

注　対象スケジュール・テーブルに複数の満了カウント値 “OsScheduleTblExpPointOffset” が定義されている場合，上記計算式の満了カウント値 “OsScheduleTblExpPointOffset” は，その最小値となります。

StartScheduleTableAbs

[概要]

絶対スケジュール・テーブルの起動

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2）

[指定形式]

```
StatusType StartScheduleTableAbs ( ScheduleTableType ScheduleTableID, TickType Start );
```

[引数]

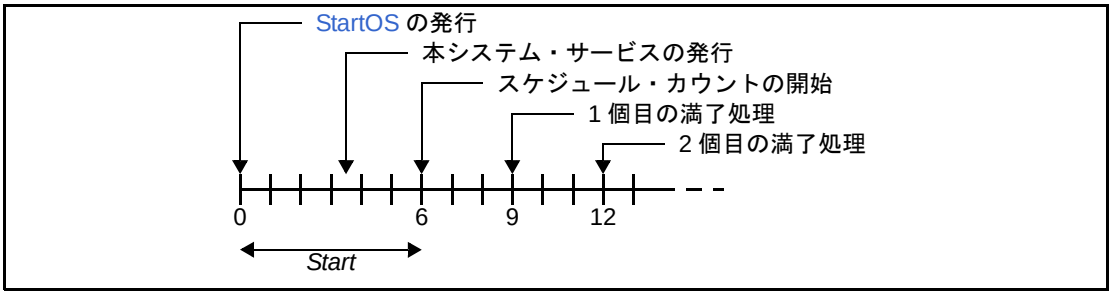
I/O	引数	説明
I	`ScheduleTableType` *ScheduleTableID*;	スケジュール・テーブル識別子
I	`TickType` *Start*;	絶対カウント値

[機能]

対象スケジュール・テーブル（引数 *ScheduleTableID* で指定された絶対スケジュール・テーブル）を STOPPED 状態から RUNNING 状態へと遷移させます。

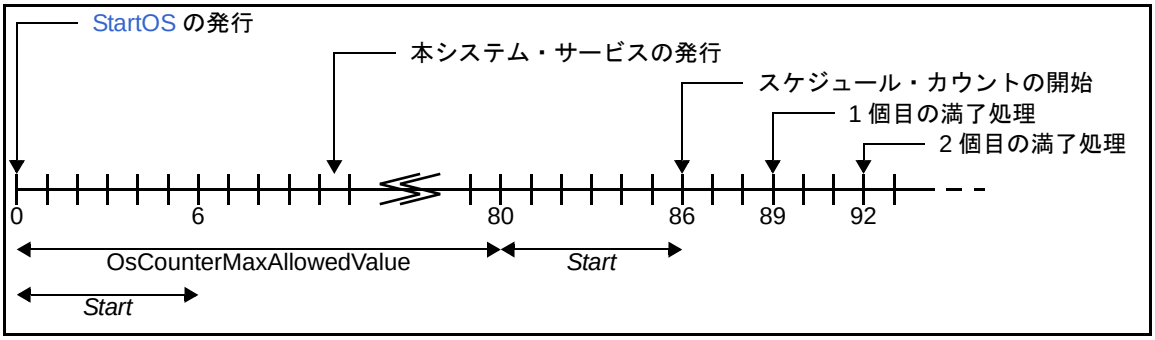
備考 1. 本システム・サービスを周期属性 “OsScheduleTableRepeating” が非周期属性（FALSE）のスケジュール・テーブルに対して発行した場合は “ ワンショット・スケジュール・テーブル ” として，周期属性（TRUE）のスジュール・テーブルに対して発行場合は “ 周期スケジュール・テーブル ” として動作します。

備考 2. 対象スケジュール・テーブルがワンショット・スケジュール・テーブルとして動作する場合，本システム・サービスの発行後，引数 *Start* で指定された絶対カウント値がカウント値（対象スケジュール・テーブルが関連付けられているカウンタのカウント値）と一致した際にスケジュール・カウントが開始され，最終満了処理を実行後，対象スケジュール・テーブルは RUNNING 状態から STOPPED 状態へと遷移します。
以下に，*Start* = 6，満了処理が 2 個（1 個目の OsScheduleTblExpPointOffset = 3，2 個目の OsScheduleTblExpPointOffset = 6）の場合の流れを示します。

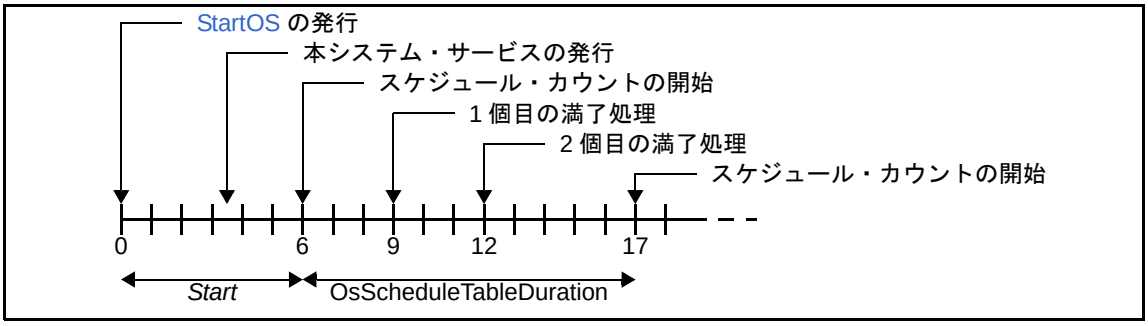


RV850 では，カウンタの加算処理（カウンタに 0x1 を加算）を行った際，カウンタの加算結果がオーバフローする（最大カウント値 “OsCounterMaxAllowedValue” を越える）場合，エラーとして扱わず，対象カウンタに 0x0 を設定します。
したがって，引数 *Start* に指定された値がカウント値（対象スケジュール・テーブルが関連付けられているカウンタのカウント値）よりも小さい場合，ワンショット・スケジュール・テーブルの動作は，以下のようになります。

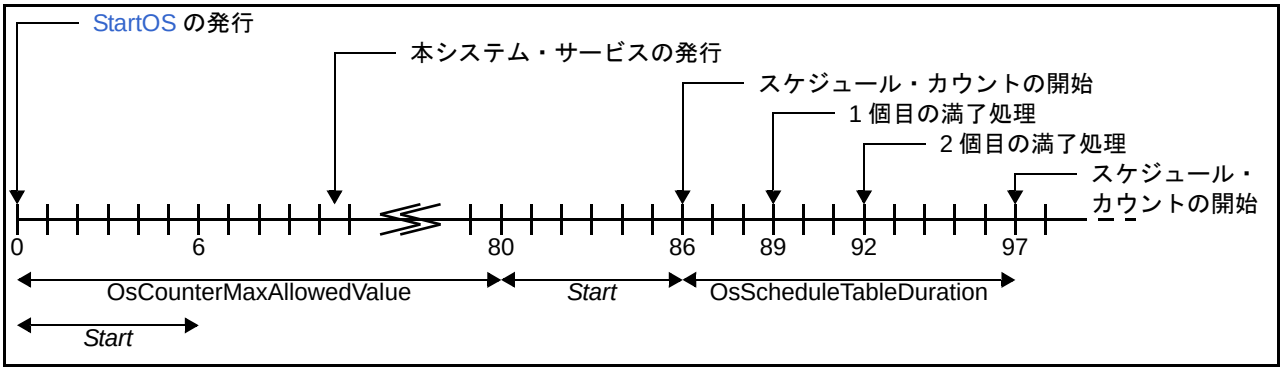
以下に，*Start* = 6，OsCounterMaxAllowedValue = 80，満了処理が 2 個（1 個目の OsScheduleTblExpPointOffset = 3，2 個目の OsScheduleTblExpPointOffset = 6）の場合の流れを示します。



備考 3. 対象スケジュール・テーブルが周期スケジュール・テーブルとして動作する場合，本システム・サービスの発行後，引数 *Start* で指定された絶対カウント値がカウント値（対象スケジュール・テーブルが関連付けられているカウンタのカウント値）と一致した際にスケジュール・カウントが開始され，スケジュール・カウント値 “OsScheduleTableDuration” で定義された値の数だけカウント値が更新されるたびにスケジュール・カウントが繰り返し実行されます。
以下に，*Start* = 6，OsScheduleTableDuration = 11，満了処理が 2 個（1 個目の OsScheduleTblExpPointOffset = 3，2 個目の OsScheduleTblExpPointOffset = 6）の場合の流れを示します。



RV850 では，カウンタの加算処理（カウンタに 0x1 を加算）を行った際，カウンタの加算結果がオーバフローする（最大カウント値 “OsCounterMaxAllowedValue” を越える）場合，エラーとして扱わず，対象カウンタに 0x0 を設定します。
したがって，引数 *Start* に指定された値がカウント値（対象スケジュール・テーブルが関連付けられているカウンタのカウント値）よりも小さい場合，周期スケジュール・テーブルの動作は，以下のようになります。
以下に，*Start* = 6，OsCounterMaxAllowedValue = 80，OsScheduleTableDuration = 11，満了処理が 2 個（1 個目の OsScheduleTblExpPointOffset = 3，2 個目の OsScheduleTblExpPointOffset = 6）の場合の流れを示します。



［戻り値］

マクロ	数値	説明
E_OK	0x0	正常終了
E_OS_ACCESS	0x1	異常終了 - 本システム・サービスを発行した処理プログラムが所属しているOSアプリケーションは，対象スケジュール・テーブルに対するアクセス権がない（SC3限定） - 対象スケジュール・テーブルの所属しているOSアプリケーションがAPPLICATION_RESTARTING状態，またはAPPLICATION_TERMINATED状態である（SC3限定）
E_OS_CALLEVEL	0x2	発行有効範囲外の処理プログラムから発行した
E_OS_ID	0x3	引数 *ScheduleTableID* の指定が不正である
E_OS_STATE	0x7	対象スケジュール・テーブルがRUNNING状態，またはNEXT状態である
E_OS_VALUE	0x8	引数 *Start* の指定が不正（*Start* > 最大カウント値 “OsCounterMaxAllowedValue”）である
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した

StopScheduleTable

[概要]

スケジュール・テーブルの終了

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2）

[指定形式]

```
StatusType StopScheduleTable ( ScheduleTableType ScheduleTableID );
```

[引数]

I/O	引数	説明
I	ScheduleTableType *ScheduleTableID*;	スケジュール・テーブル識別子

[機能]

対象スケジュール・テーブル（引数 *ScheduleTableID* で指定されたスケジュール・テーブル）を RUNNING 状態，または NEXT 状態から STOPPED 状態へと遷移させます。

備考　対象スケジュール・テーブルが切り替え前のスケジュール・テーブル（NextScheduleTable の発行時に引数 *ScheduleTableID_From* で指定されたスケジュール・テーブル）であった場合，対象スケジュール・テーブルと伴に，NextScheduleTable の発行時に引数 *ScheduleTableID_To* で指定されたスケジュール・テーブルについても STOPPED 状態へと遷移させます。

[戻り値]

マクロ	数値	説明
E_OK	0x0	正常終了
E_OS_ACCESS	0x1	異常終了 - 本システム・サービスを発行した処理プログラムが所属しているOSアプリケーションは，対象スケジュール・テーブルに対するアクセス権がない（SC3 限定） - 対象スケジュール・テーブルの所属しているOSアプリケーションがAPPLICATION_RESTARTING状態，またはAPPLICATION_TERMINATED状態である（SC3 限定）
E_OS_CALLEVEL	0x2	発行有効範囲外の処理プログラムから発行した
E_OS_ID	0x3	引数 *ScheduleTableID* の指定が不正である
E_OS_NOFUNC	0x5	対象スケジュール・テーブルが STOPPED 状態である
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した

NextScheduleTable

[概要]

スケジュール・テーブルの切り替え

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2）

[指定形式]

```
StatusType NextScheduleTable ( ScheduleTableType ScheduleTableID_From, ScheduleTableType ScheduleTableID_To );
```

[引数]

I/O	引数	説明
I	ScheduleTableType *ScheduleTableID_From*;	スケジュール・テーブル識別子
I	ScheduleTableType *ScheduleTableID_To*;	スケジュール・テーブル識別子

[機能]

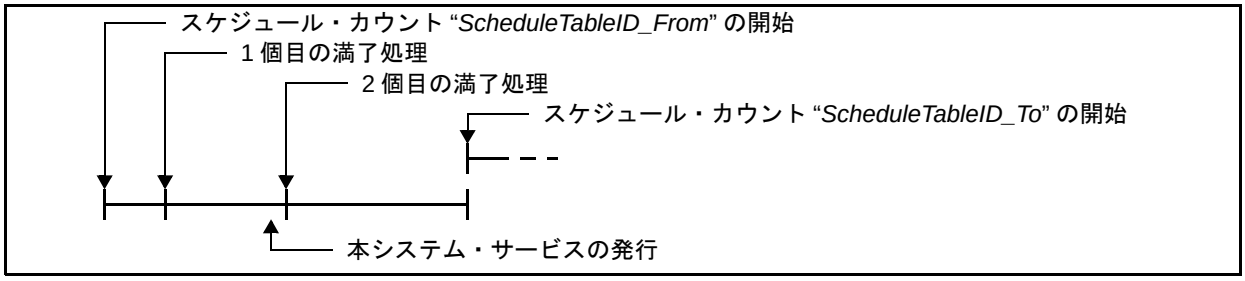
引数 *ScheduleTableID_From* で指定されたスケジュール・テーブルと引数 *ScheduleTableID_To* で指定されたスケジュール・テーブルの切り替えを行います。

なお，スケジュール・テーブルの切り替えタイミングは，本システム・サービスの発行タイミングに依存しています。

- 最終満了処理の実行前に本システム・サービスを発行

  スケジュール・テーブル *ScheduleTableID_From* のスケジュール・カウントが完了した時点で切り替え処理が実行されます。

  したがって，下図のように，スケジュール・テーブル *ScheduleTableID_From* に 2 個の満了処理が定義されており，2 個目の満了処理（最終満了処理）を実行する以前に本システム・サービスが発行された際には，スケジュール・テーブル *ScheduleTableID_From* のスケジュール・カウントが完了した時点でスケジュール・テーブル *ScheduleTableID_To* への切り替え処理が実行されます。

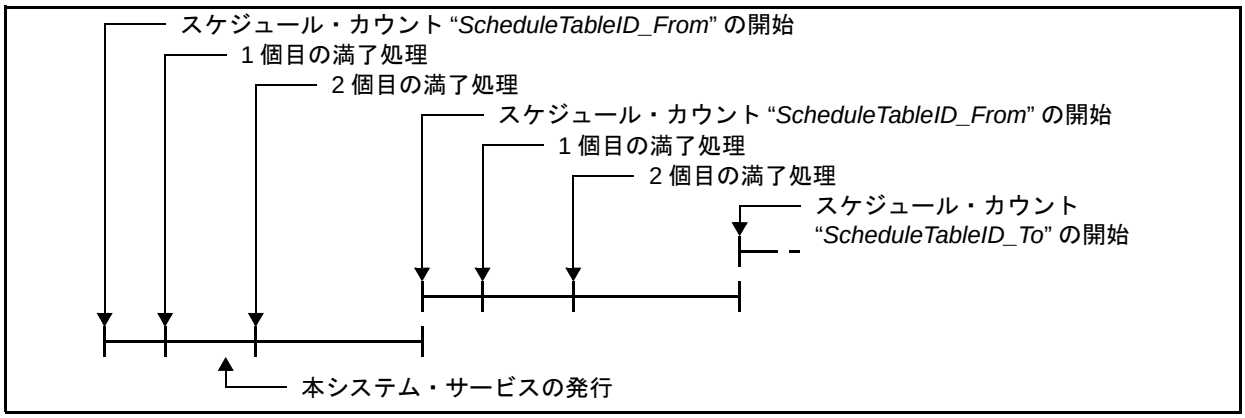


- 最終満了処理の実行後に本システム・サービスを発行

  スケジュール・テーブル *ScheduleTableID_From* のスケジュール・カウントが完了した時点で再度スケジュール・テーブル *ScheduleTableID_From* のスケジュール・カウントが開始され，最終満了処理を実行後，スケジュール・テーブル *ScheduleTableID_To* への切り替え処理が実行されます。

  したがって，下図のように，スケジュール・テーブル *ScheduleTableID_From* に 2 個の満了処理が定義されており，2 個目の満了処理（最終満了処理）を実行後に本システム・サービスが発行された際には，スケジュール・テーブル *ScheduleTableID_From* のスケジュール・カウントが完了した時点で再度スケジュール・テーブルのスケジュール・カウント *ScheduleTableID_From* が開始され，スケジュール・テーブル *ScheduleTableID_From* の再スケ

ジュール・カウントが完了した時点でスケジュール・テーブル *ScheduleTableID_To* への切り替え処理が実行されます。



備考 1. 引数 *ScheduleTableID_From* で指定されたスケジュール・テーブルは，スケジュール・テーブルの切り替え処理後，STOPPED 状態へと遷移します。
また，引数 *ScheduleTableID_To* で指定されたスケジュール・テーブルは，本システム・サービスの発行に伴い，NEXT 状態へと遷移したのち，スケジュール・テーブルの切り替え処理後，RUNNING 状態へと遷移します。

備考 2. 本システム・サービスの発行からスケジュール・テーブルの切り替え処理が実行されるまでの間に，本システム・サービスが再発行された場合，先のシステム・サービス発行時に引数 *ScheduleTableID_To* で指定されたスケジュール・テーブルは NEXT 状態から STOPPED 状態へと遷移します。

備考 3. AUTOSAR 仕様では，引数 *ScheduleTableID_From*，*ScheduleTableID_To* で指定可能なスケジュール・テーブルの周期属性 “OsScheduleTableRepeating” に関する規定が行われていません。
そこで，RV850 では，該当引数に指定可能なスケジュール・テーブルを，周期属性 “OsScheduleTableRepeating” を付与（TRUE）したものに限定し，該当引数に周期属性以外のスケジュール・テーブルが指定された際には，エラー・ステータス E_OS_ID を返却しています。

［戻り値］

マクロ	数値	説明
E_OK	0x0	正常終了
E_OS_ACCESS	0x1	異常終了 - 本システム・サービスを発行した処理プログラムが所属しているOSアプリケーションは，対象スケジュール・テーブルに対するアクセス権がない（SC3 限定） - 対象スケジュール・テーブルの所属しているOSアプリケーションがAPPLICATION_RESTARTING状態，またはAPPLICATION_TERMINATED状態である（SC3 限定）
E_OS_CALLEVEL	0x2	発行有効範囲外の処理プログラムから発行した

マクロ	数値	説明
E_OS_ID	0x3	異常終了 - 引数 *ScheduleTableID_From* の指定が不正である - 引数 *ScheduleTableID_To* の指定が不正である - 引数 *ScheduleTableID_From* で指定されたスケジュール・テーブルの周期属性 “OsScheduleTableRepeating” が “ 周期属性を付与しない（FALSE）” である - 引数 *ScheduleTableID_To* で指定されたスケジュール・テーブルの周期属性 “OsScheduleTableRepeating” が “ 周期属性を付与しない（FALSE）” である - 引数 *ScheduleTableID_From* で指定されたスケジュール・テーブルに関連付けられているカウンタと引数 *ScheduleTableID_To* で指定されたスケジュール・テーブルに関連付けられているカウンタが異なる
E_OS_NOFUNC	0x5	対象スケジュール・テーブル *ScheduleTableID_From* が STOPPED 状態，または NEXT 状態である
E_OS_STATE	0x7	対象スケジュール・テーブル *ScheduleTableID_To* が NEXT 状態，または RUNNING 状態である
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した

GetScheduleTableStatus

[概要]

現在状態の獲得

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2）

[指定形式]

```
StatusType GetScheduleTableStatus ( ScheduleTableType ScheduleTableID,
ScheduleTableStatusRefType ScheduleStatus );
```

[引数]

I/O	引数	説明
I	`ScheduleTableType` *ScheduleTableID*`;`	スケジュール・テーブル識別子
O	`ScheduleTableStatusRefType` *ScheduleStatus*`;`	獲得した現在状態を格納する領域へのポインタ

[機能]

対象スケジュール・テーブル（引数 *ScheduleTableID* で指定されたスケジュール・テーブル）の現在状態を獲得し，引数 *ScheduleStatus* で指定された領域に格納します。

なお，引数 *ScheduleStatus* には，現在状態の種類により，以下の値が格納されます。

マクロ	数値	Description
SCHEDULETABLE_STOPPED	0x0	STOPPED 状態
SCHEDULETABLE_NEXT	0x1	NEXT 状態
SCHEDULETABLE_RUNNING	0x4	RUNNING 状態

[戻り値]

マクロ	数値	説明
E_OK	0x0	正常終了
E_OS_ACCESS	0x1	異常終了 - 本システム・サービスを発行した処理プログラムが所属している OS アプリケーションは，対象スケジュール・テーブルに対するアクセス権がない（SC3 限定） - 対象スケジュール・テーブルの所属している OS アプリケーションが APPLICATION_RESTARTING 状態，または APPLICATION_TERMINATED 状態である（SC3 限定）
E_OS_CALLEVEL	0x2	発行有効範囲外の処理プログラムから発行した
E_OS_ID	0x3	引数 *ScheduleTableID* の指定が不正である

マクロ	数値	説明
E_OS_PARAM_POINTER	0x12	引数 *ScheduleStatus* の指定が不正（NULL ポインタ）である
E_OS_ILLEGAL_ADDRESS	0x13	本システム・サービスを発行した処理プログラムは，引数 *ScheduleStatus* で指定された領域に対するアクセス権がない（SC3 限定）
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した

14.4.8 OS アプリケーション管理

以下に，RV850 が提供している OS アプリケーション管理用システム・サービスを示します。

表 14.24 OS アプリケーション管理用システム・サービス

システム・サービス名	機能概要
GetApplicationID	OS アプリケーション識別子の獲得
GetCurrentApplicationID	現在動作中 OS アプリケーション識別子の獲得
GetISRID	割り込みサービス・ルーチン識別子の獲得
CallTrustedFunction	Trusted 関数の呼び出し
CallTrustedFunctionRestricted	Trusted 関数の機能制限付き呼び出し
CheckISRMemoryAccess	アクセス権の確認（割り込みサービス・ルーチン）
CheckTaskMemoryAccess	アクセス権の確認（タスク）
CheckObjectAccess	アクセス権の確認（オブジェクト）
CheckObjectOwnership	OS アプリケーション識別子の獲得
TerminateApplication	OS アプリケーションの終了
AllowAccess	OS アプリケーションの起動
GetApplicationState	現在状態の獲得

備考 AUTOSAR 仕様では，システム・サービス CallTrustedFunctionRestricted の規定が行われていません。CallTrustedFunctionRestricted は，RV850 が独自に追加したシステム・サービスです。

GetApplicationID

[概要]

OS アプリケーション識別子の獲得

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2），共通フック・ルーチン（StartupHook，ShutdownHook，PostTaskHook，PreTaskHook，ErrorHook，ProtectionHook），固有フック・ルーチン（StartupHook_*OsApplication*，ShutdownHook_*OsApplication*，ErrorHook_*OsApplication*）

[指定形式]

```
ApplicationType GetApplicationID ( void );
```

[引数]

なし

[機能]

対象 OS アプリケーション（本システム・サービスを発行した処理プログラムが所属している OS アプリケーション）の OS アプリケーション識別子を獲得し，戻り値として返します。

備考 1. 本システム・サービスの発行は，スケーラビリティ・クラス “OsScalabilityClass” で “ スケーラビリティ・クラス 3（SC3）” を定義した場合に限られます。

備考 2. 本システム・サービスを共通フック・ルーチン（PostTaskHook，PreTaskHook，ErrorHook，ProtectionHook），固有フック・ルーチン（StartupHook_*OsApplication*，ErrorHook_*OsApplication*，ShutdownHook_*OsApplication*）から発行した場合，該当フック・ルーチンを呼び出した処理プログラムが所属している OS アプリケーションの識別子を戻り値として返します。

備考 3. 本システム・サービスを共通フック・ルーチン（StartupHook，ShutdownHook）から発行した場合，INVALID_OSAPPLICATION を戻り値として返します。

備考 4. 本システム・サービスを Trusted 関数から発行した場合，該当 Trusted 関数を呼び出した処理プログラムが所属する OS アプリケーションの識別子を戻り値として返します。

備考 5. 本システム・サービスが戻り値として返す OS アプリケーション識別子は数値であり，識別子 “OsApplication” に指定された名前との対応は，コンフィギュレータが出力した SIT ファイルに定義されています。

[戻り値]

マクロ	数値	説明
－	0x7FFF 以外	獲得した OS アプリケーション識別子
INVALID_OSAPPLICATION	0x7FFF	発行有効範囲外の処理プログラムから発行した

備考 本システム・サービスをタスク／割り込みサービス・ルーチン（カテゴリ 2）以外の処理プログラムから呼び出された共通フック・ルーチン（ProtectionHook）で発行した場合，戻り値として INVALID_OSAPPLICATION（0x7FFF）を返します。

GetCurrentApplicationID

[概要]

現在動作中 OS アプリケーション識別子の獲得

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2），共通フック・ルーチン（StartupHook，ShutdownHook，PostTaskHook，PreTaskHook，ErrorHook，ProtectionHook），固有フック・ルーチン（StartupHook_*OsApplication*，ShutdownHook_*OsApplication*，ErrorHook_*OsApplication*）

[指定形式]

```
ApplicationType GetCurrentApplicationID ( void );
```

[引数]

なし

[機能]

対象 OS アプリケーション（本システム・サービスを発行した時に動作中の OS アプリケーション）の OS アプリケーション識別子を獲得し，戻り値として返します。GetApplicationID との違いは，Trusted 関数から発行した場合の戻り値のみです。

備考 1. 本システム・サービスの発行は，スケーラビリティ・クラス “OsScalabilityClass” で “ スケーラビリティ・クラス 3（SC3）” を定義した場合に限られます。

備考 2. 本システム・サービスを共通フック・ルーチン（PostTaskHook，PreTaskHook，ErrorHook，ProtectionHook），固有フック・ルーチン（StartupHook_*OsApplication*，ErrorHook_*OsApplication*，ShutdownHook_*OsApplication*）から発行した場合，該当フック・ルーチンを呼び出した処理プログラムが所属している OS アプリケーションの識別子を戻り値として返します。

備考 3. 本システム・サービスを共通フック・ルーチン（StartupHook，ShutdownHook）から発行した場合，INVALID_OSAPPLICATION を戻り値として返します。

備考 4. 本システム・サービスを Trusted 関数から発行した場合，該当 Trusted 関数が所属する OS アプリケーションの識別子を戻り値として返します。

備考 5. 本システム・サービスが戻り値として返す OS アプリケーション識別子は数値であり，識別子 “OsApplication” に指定された名前との対応は，コンフィギュレータが出力した SIT ファイルに定義されています。

[戻り値]

マクロ	数値	説明
―	0x7FFF 以外	獲得した OS アプリケーション識別子
INVALID_OSAPPLICATION	0x7FFF	発行有効範囲外の処理プログラムから発行した

備考 本システム・サービスをタスク／割り込みサービス・ルーチン（カテゴリ 2）以外の処理プログラムから呼び出された共通フック・ルーチン（ProtectionHook）で発行した場合，戻り値として INVALID_OSAPPLICATION（0x7FFF）を返します。

GetISRID

[概要]

割り込みサービス・ルーチン識別子の獲得

[発行有効範囲]

割り込みサービス・ルーチン（カテゴリ 2），共通フック・ルーチン（ErrorHook，ProtectionHook），固有フック・ルーチン（ErrorHook_*OsApplication*）

[指定形式]

```
ISRType GetISRID ( void );
```

[引数]

なし

[機能]

対象割り込みサービス・ルーチン（本システム・サービスを発行した割り込みサービス・ルーチン，または本システム・サービスを発行したフック・ルーチンを呼び出した割り込みサービス・ルーチン）の割り込みサービス・ルーチン識別子を獲得し，戻り値として返します。

備考 1. 本システム・サービスをタスクから呼び出されたフック・ルーチンで発行した場合，戻り値として INVALID_ISR（0x7FFF）を返します。

備考 2. 本システム・サービスが戻り値として返す割り込みサービス・ルーチン識別子は数値であり，識別子 “OsIsr” に指定された名前との対応は，コンフィギュレータが出力した SIT ファイルに定義されています。

[戻り値]

マクロ	数値	説明
－	0x7FFF 以外	獲得した割り込みサービス・ルーチン識別子
INVALID_ISR	0x7FFF	発行有効範囲外の処理プログラムから発行した

備考 本システム・サービスをタスクから呼び出された共通フック・ルーチン（ErrorHook，ProtectionHook），固有フック・ルーチン（ErrorHook_*OsApplication*）で発行した場合，戻り値として INVALID_ISR（0x7FFF）を返します。

CallTrustedFunction

[概要]

Trusted 関数の呼び出し

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2）

[指定形式]

```
StatusType CallTrustedFunction ( TrustedFunctionIndexType FunctionIndex,
TrustedFunctionParameterRefType FunctionParams );
```

[引数]

I/O	引数	説明
I	`TrustedFunctionIndexType` *FunctionIndex*;	Trusted 関数識別子
I	`TrustedFunctionParameterRefType` *FunctionParams*;	引き継ぎデータを格納した領域へのポインタ

[機能]

対象 Trusted 関数（引数 *FunctionIndex* で指定された Trusted 関数）を呼び出します。
なお，本システム・サービス処理から Trusted 関数の本体を呼び出す際には，引数 *FunctionIndex* で指定された値を Trusted 関数の第 1 引数として，引数 *FunctionParams* で指定された値（ポインタ）を Trusted 関数の第 2 引数として渡しています。
Trusted 関数への引き継ぎデータ (パラメータ) の詳細は 10.2.3 Trusted 関数の引き継ぎデータ を参照してください。

備考 1. 本システム・サービスの発行は，スケーラビリティ・クラス “OsScalabilityClass” で “ スケーラビリティ・クラス 3（SC3）” を定義した場合に限られます。

備考 2. 引数を持たない Trusted 関数を呼び出す場合，引数 *FunctionParams* には NULL を指定します。

備考 3. Trusted 関数はスーパバイザ・モードで動作するため，本システム・サービスを Non-Trusted な OS アプリケーションに所属している処理プログラムから発行した場合，モード切り替え処理（ユーザ・モードからスーパバイザ・モードへと移行）が行われます。

備考 4. 本システム・サービスでは引数 *FunctionParams* の指し示す引継ぎデータの正当性を確認できません。したがって，引継ぎデータの正当性を確認する必要がある場合は，本システム・サービスの発行により呼び出された Trusted 関数本体で行ってください。

備考 5. Trusted 関数についての詳細は，「10.2 Trusted 関数」を参照してください。

［戻り値］

マクロ	数値	説明
E_OK	0x0	正常終了
E_OS_ACCESS	0x1	対象 Trusted 関数の所属している OS アプリケーションが APPLICATION_RESTARTING 状態，または APPLICATION_TERMINATED 状態である
E_OS_CALLEVEL	0x2	発行有効範囲外の処理プログラムから発行した
E_OS_SERVICEID	0x11	引数 *FunctionIndex* の指定が不正である
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した

CallTrustedFunctionRestricted

[概要]

Trusted 関数の機能制限付き呼び出し

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2），共通フック・ルーチン（StartupHook，ShutdownHook，PostTaskHook，PreTaskHook，ErrorHook，ProtectionHook），固有フック・ルーチン（StartupHook_*OsApplication*，ShutdownHook_*OsApplication*，ErrorHook_*OsApplication*），クリティカル・セクション

[指定形式]

```
StatusType CallTrustedFunctionRestricted ( TrustedFunctionIndexType FunctionIndex,
TrustedFunctionParameterRefType FunctionParams );
```

[引数]

I/O	引数	説明
I	`TrustedFunctionIndexType` *FunctionIndex*;	Trusted 関数識別子
I	`TrustedFunctionParameterRefType` *FunctionParams*;	引き継ぎデータを格納した領域へのポインタ

[機能]

対象 Trusted 関数（引数 *FunctionIndex* で指定された Trusted 関数）を呼び出します。本システム・サービスは，CallTrustedFunction と比較すると，以下に示す機能的な制限が設けられていますが，Trusted 関数の呼び出しオーバーヘッドを削減できます。

- スタック・モニタリング機能 “OsStackMonitoring” で TRUE を指定した場合においても，RV850 は本システム・サービス発行時にスタック・オーバーフローの発生を検知しません。スタックが不足している状況で本システム・サービスを発行した場合，その後の動作を保証しません。
- RV850 では，本システム・サービスを発行した処理プログラムから Trusted 関数に制御を移す際，割り込みの受け付け禁止処理（PSW の ID ビットに対する操作）を行います。このため，すべての割り込みを受け付けることができません。Trusted 関数から本システム・サービスを発行した処理プログラムに制御を移す際，PSW の ID ビットを，本システム・サービス発行前の値に戻します。
- 本システム・サービスの発行により Trusted 関数を呼び出す場合は，“FPSR の退避処理／復帰処理 ” は行いません。
- 本システム・サービスは，発行有効範囲および引数 *FunctionIndex* の正当性の確認を行いません。発行有効範囲外の処理プログラムから本システム・サービスを発行した場合や，引数 *FunctionIndex* の指定が不定である場合，その後の動作を保証しません。

なお，本システム・サービス処理から Trusted 関数の本体を呼び出す際には，引数 FunctionIndex で指定された値を Trusted 関数の第 1 引数として，引数 FunctionParams で指定された値（ポインタ）を Trusted 関数の第 2 引数として渡しています。

Trusted 関数への引き継ぎデータ (パラメータ) の詳細は 10.2.3 Trusted 関数の引き継ぎデータ を参照してください。

備考 1. 本システム・サービスの発行は，スケーラビリティ・クラス “OsScalabilityClass” で “ スケーラビリティ・クラス 3（SC3）” を定義した場合に限られます。

備考 2. 引数を持たない Trusted 関数を呼び出す場合，引数 *FunctionParams* には NULL を指定します。

備考 3. Trusted 関数はスーパバイザ・モードで動作するため，本システム・サービスを Non-Trusted な OS アプリケーションに所属している処理プログラムから発行した場合，モード切り替え処理（ユーザ・モードからスーパバイザ・モードへと移行）が行われます。

備考 4. 本システム・サービスでは引数 *FunctionParams* の指し示す引継ぎデータの正当性を確認できません。したがって，引継ぎデータの正当性を確認する必要がある場合は，本システム・サービスの発行により呼び出された Trusted 関数本体で行ってください。

備考 5. Trusted 関数についての詳細は，「10.2 Trusted 関数」を参照してください。本システム・サービスで呼び出す Trusted 関数には，CallTrustedFunction で呼び出す場合に比べて多くの制約が存在するため，Trusted 関数の記述内容には注意してください。

［戻り値］

マクロ	数値	説明
E_OK	0x0	正常終了

CheckISRMemoryAccess

[概要]

アクセス権の確認（割り込みサービス・ルーチン）

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2），共通フック・ルーチン（ErrorHook，ProtectionHook），固有フック・ルーチン（ErrorHook_*OsApplication*）

[指定形式]

```
AccessType CheckISRMemoryAccess ( ISRType ISRID, MemoryStartAddressType Address,
MemorySizeType Size );
```

[引数]

I/O	引数	説明
I	`ISRType` *ISRID*;	割り込みサービス・ルーチン識別子
I	`MemoryStartAddressType` *Address*;	メモリ領域の先頭アドレス
I	`MemorySizeType` *Size*;	メモリ領域のサイズ（単位：バイト）

[機能]

対象割り込みサービス・ルーチン（引数 *ISRID* で指定された割り込みサービス・ルーチン）が該当メモリ領域（引数 *Address*，および *Size* で指定されたメモリ領域）に対してアクセス権を有しているか否か，および該当メモリ領域がスタック領域であるか否かを確認し，戻り値として対応する値を返します。

備考 1. 本システム・サービスの発行は，スケーラビリティ・クラス “OsScalabilityClass” で “ スケーラビリティ・クラス 3（SC3）” を定義した場合に限られます。

備考 2. 本システム・サービスの戻り値は，該当メモリ領域の全体に対して有効なアクセス権を返します。
したがって，一部のメモリ領域が書き込み不可の場合，該当メモリ領域に対する書き込みは不可であるものとして戻り値を返します。

備考 3. 処理プログラムは，対象割り込みサービス・ルーチンの所属している OS アプリケーションに対するアクセス権の有無に関係なく，本システム・サービスを発行することができます。

備考 4. 処理プログラムは，対象割り込みサービス・ルーチンの所属している OS アプリケーションの状態に関係なく，本システム・サービスを発行することができます。

［戻り値］

マクロ	数値	説明
T_u2_NOACCESS	0x0	正常終了 - アクセス権なし 異常終了 - 発行有効範囲外の処理プログラムから発行した - 引数 *ISRID* の指定が不正である - クリティカル・セクションから発行した
T_u2_EXECUTABLE	0x2	実行可
T_u2_READABLE	0x4	読み込み可
T_u2_WRITEABLE	0x8	書き込み可
T_u2_STACKSPACE	0x10	スタック領域

備考　戻り値は確認結果（有しているアクセス権の種別，およびスタック領域であるか否か）の和となります。
対象割り込みサービス・ルーチンが有しているアクセス権の種別が “ 実行可／読み込み可／書き込み否 ”，該当メモリ領域が “ 非スタック領域 ” の場合，戻り値は “0x6” となります。

CheckTaskMemoryAccess

[概要]

アクセス権の確認（タスク）

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2），共通フック・ルーチン（ErrorHook，ProtectionHook），固有フック・ルーチン（ErrorHook_*OsApplication*）

[指定形式]

```
AccessType CheckTaskMemoryAccess ( TaskType TaskID, MemoryStartAddressType Address,
MemorySizeType Size );
```

[引数]

I/O	引数	説明
I	`TaskType` *TaskID*;	タスク識別子
I	`MemoryStartAddressType` *Address*;	メモリ領域の先頭アドレス
I	`MemorySizeType` *Size*;	メモリ領域のサイズ（単位：バイト）

[機能]

対象タスク（引数 *TaskID* で指定されたタスク）が該当メモリ領域（引数 *Address*，および *Size* で指定されたメモリ領域）に対してアクセス権を有しているか否か，および該当メモリ領域がスタック領域であるか否かを確認し，戻り値として対応する値を返します。

備考 1. 本システム・サービスの発行は，スケーラビリティ・クラス “OsScalabilityClass” で “ スケーラビリティ・クラス 3（SC3）” を定義した場合に限られます。

備考 2. 本システム・サービスの戻り値は，該当メモリ領域の全体に対して有効なアクセス権を返します。
したがって，一部のメモリ領域が書き込み不可の場合，該当メモリ領域に対する書き込みは不可であるものとして戻り値を返します。

備考 3. 処理プログラムは，対象タスクの所属している OS アプリケーションに対するアクセス権の有無に関係なく，本システム・サービスを発行することができます。

備考 4. 処理プログラムは，対象タスクの所属している OS アプリケーションの状態に関係なく，本システム・サービスを発行することができます。

［戻り値］

マクロ	数値	説明
T_u2_NOACCESS	0x0	正常終了 - アクセス権なし 異常終了 - 発行有効範囲外の処理プログラムから発行した - 引数 *TaskID* の指定が不正である - クリティカル・セクションから発行した
T_u2_EXECUTABLE	0x2	実行可
T_u2_READABLE	0x4	読み込み可
T_u2_WRITEABLE	0x8	書き込み可
T_u2_STACKSPACE	0x10	スタック領域

備考　戻り値は確認結果（有しているアクセス権の種別，およびスタック領域であるか否か）の和となります。
対象タスクが有しているアクセス権の種別が “ 実行否／読み込み可／書き込み可 ”，該当メモリ領域が “ スタック領域 ” の場合，戻り値は “0x1C” となります。

CheckObjectAccess

[概要]

アクセス権の確認（オブジェクト）

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2），共通フック・ルーチン（ErrorHook，ProtectionHook），固有フック・ルーチン（ErrorHook_*OsApplication*）

[指定形式]

```
ObjectAccessType CheckObjectAccess ( ApplicationType ApplID, ObjectTypeType ObjectType,
<Object>Type ObjectID );
```

備考　引数 *ObjectID* の変数型 <*Object*>Type は，指定するオブジェクトの種類により異なります（TaskType，ISRType，AlarmType など）。

[引数]

I/O	引数	説明
I	`ApplicationType` *ApplID*;	OS アプリケーション識別子
I	`ObjectTypeType` *ObjectType*;	オブジェクトの種類
I	<*Object*>`Type` *ObjectID*;	オブジェクト識別子

[機能]

対象 OS アプリケーション（引数 *ApplID* で指定された OS アプリケーション）に所属しているオブジェクト（タスク，割り込みサービス・ルーチン，アラームなど）が引数 *ObjectType*，および *ObjectID* で指定されたオブジェクトに対してアクセス権を有しているか否かを確認し，戻り値として対応する値を返します。

なお，引数 *ObjectType* には，オブジェクトの種類により，以下の値を指定します。

マクロ	数値	説明
OBJECT_TASK	0x1	タスク
OBJECT_ISR	0x2	割り込みサービス・ルーチン
OBJECT_ALARM	0x3	アラーム
OBJECT_RESOURCE	0x4	リソース
OBJECT_COUNTER	0x5	カウンタ
OBJECT_SCHEDULETABLE	0x6	スケジュール・テーブル

備考 1.　本システム・サービスの発行は，スケーラビリティ・クラス “OsScalabilityClass” で “ スケーラビリティ・クラス 3（SC3）” を定義した場合に限られます。

備考 2.　処理プログラムは，対象オブジェクトの所属している OS アプリケーションに対するアクセス権の有無に関係なく，本システム・サービスを発行することができます。

備考 3.　処理プログラムは，対象オブジェクトの所属している OS アプリケーションの状態に関係なく，本システム・サービスを発行することができます。

［戻り値］

マクロ	数値	説明
NO_ACCESS	0x0	正常終了 - アクセス権なし 異常終了 - 発行有効範囲外の処理プログラムから発行した - 引数 *AppIID* の指定が不正である - 引数 *ObjectType* の指定が不正である - 引数 *ObjectID* の指定が不正である - クリティカル・セクションから発行した
ACCESS	0x1	アクセス権あり

CheckObjectOwnership

[概要]

OS アプリケーション識別子の獲得

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2），共通フック・ルーチン（ErrorHook，ProtectionHook），固有フック・ルーチン（ErrorHook_*OsApplication*）

[指定形式]

```
ApplicationType CheckObjectOwnership ( ObjectTypeType ObjectType, <Object>Type ObjectID );
```

備考　引数 *ObjectID* の変数型 <*Object*>Type は，指定するオブジェクトの種類により異なります（TaskType，ISRType，AlarmType など）。

[引数]

I/O	引数	説明
I	`ObjectTypeType` *ObjectType*;	オブジェクトの種類
I	`<Object>Type` *ObjectID*;	オブジェクト識別子

[機能]

対象オブジェクト（引数 *ObjectType*，および *ObjectID* で指定されたオブジェクト）が所属している OS アプリケーションの識別子を獲得し，戻り値として返します。

なお，引数 *ObjectType* には，オブジェクトの種類により，以下の値を指定します。

マクロ	数値	説明
OBJECT_TASK	0x1	タスク
OBJECT_ISR	0x2	割り込みサービス・ルーチン
OBJECT_ALARM	0x3	アラーム
OBJECT_RESOURCE	0x4	リソース
OBJECT_COUNTER	0x5	カウンタ
OBJECT_SCHEDULETABLE	0x6	スケジュール・テーブル

備考 1.　本システム・サービスの発行は，スケーラビリティ・クラス “OsScalabilityClass” で “ スケーラビリティ・クラス 3（SC3）” を定義した場合に限られます。

備考 2.　処理プログラムは，対象オブジェクトの所属している OS アプリケーションの状態に関係なく，本システム・サービスを発行することができます。

［戻り値］

マクロ	数値	説明
−	0x7FFF 以外	獲得した OS アプリケーション識別子
INVALID_OSAPPLICATION	0x7FFF	異常終了 - 本システム・サービスを発行した処理プログラムは，対象オブジェクトの所属している OS アプリケーションに対するアクセス権がない - 発行有効範囲外の処理プログラムから発行した - 引数 *ObjectType* の指定が不正である - 引数 *ObjectID* の指定が不正である - クリティカル・セクションから発行した

TerminateApplication

[概要]

OS アプリケーションの終了

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2），固有フック・ルーチン（ErrorHook_*OsApplication*）

[指定形式]

```
StatusType TerminateApplication ( ApplicationType Application, RestartType RestartOption );
```

[引数]

I/O	引数	説明
I	`ApplicationType` *Application*;	OS アプリケーション識別子
I	`RestartType` *RestartOption*;	再起動オプション NO_RESTART（0x0）： OS アプリケーションの終了 RESTART（0x1）： OS アプリケーションの終了／再起動

[機能]

対象 OS アプリケーション（引数 *Application* で指定された OS アプリケーション）に対して，引数 *RestartOption* で指定された操作を行います。

備考 1. 本システム・サービスの発行は，スケーラビリティ・クラス “OsScalabilityClass” で “ スケーラビリティ・クラス 3（SC3）” を定義した場合に限られます。

備考 2. 対象 OS アプリケーションに対する操作は，引数 *RestartOption* の指定内容により，以下のように異なります。

- NO_RESTART（0x0）
  対象 OS アプリケーションの終了処理として，以下の操作を行います。
  - 対象 OS アプリケーションに所属しているタスクを SUSPENDED 状態へと遷移
  - 対象 OS アプリケーションに所属している RUNNING 状態のタスクを強制的に終了
  - 対象 OS アプリケーションに所属している実行中の割り込みサービス・ルーチンを強制的に終了
  - 対象 OS アプリケーションに所属している，多重割り込みにより中断している割り込みサービス・ルーチンを強制的に終了
  - 対象 OS アプリケーションに所属している割り込みサービス・ルーチンに対応した割り込みの受け付けを禁止（EI レベル割り込みマスク・レジスタ IMR*m* に対する操作）
  - 対象 OS アプリケーションに所属している処理プログラムが獲得しているリソースを解放
  - 対象 OS アプリケーションに所属しているアラームを非アクティブ状態へと遷移
  - 対象 OS アプリケーションに所属しているスケジュール・テーブルを STOPPED 状態へと遷移
  - 対象 OS アプリケーションを APPLICATION_TERMINATED 状態へと遷移
- RESTART（0x1）
  対象 OS アプリケーションに対して，NO_RESTART（0x0）が指定された際と同様の操作を行ったのち，以下の操作を行います。

- タスク識別子 “OsRestartTask” の指定が行われている際には，該当タスクを SUSPENDED 状態から READY 状態へと遷移
- 対象 OS アプリケーションを APPLICATION_TERMINATED 状態から APPLICATION_RESTARTING 状態へと遷移

本システム・サービスの発行にともない APPLICATION_RESTARTING 状態へと遷移した OS アプリケーションについては，タスク識別子 “OsRestartTask” で指定されたタスクから AllowAccess を発行することにより，APPLICATION_RESTATING 状態から APPLICATION_ACCESSIBLE 状態へと遷移します。
また，本システム・サービスの発行にともない禁止された割り込みの受け付けについては，タスク識別子 “OsRestartTask” で指定されたタスクから InitApplicationInterrupts を発行することにより，割り込みの受け付けが許可されます。

備考 3. 本システム・サービスを発行した際，RUNNING 状態のタスクが存在していた場合，OS アプリケーションの終了処理によって該当タスクが SUSPENDED 状態へと遷移する以前に共通フック・ルーチン（PostTaskHook）の呼び出しが行われます。

備考 4. Non-Trusted な OS アプリケーションに所属している処理プログラムから本システム・サービスを発行した場合，引数 *Application* に “ 本システム・サービスを発行した処理プログラムが所属している OS アプリケーション以外 ” を指定した際には，戻り値として E_OS_ACCESS（0x1）を返します。

備考 5. ErrorHook_*OsApplication* から本システム・サービスを発行した場合，引数 *Application* に “ 本システム・サービスを発行した ErrorHook_*OsApplication* が所属している OS アプリケーション以外 ” を指定した際には，戻り値として E_OS_CALLEVEL（0x2）を返します。

備考 6. 本システム・サービスでは，受け付けが保留されている割り込みのクリアを行いません。
したがって，OS アプリケーションを再起動した際，受け付けが保留されている割り込みのクリアが必要な場合には，割り込みの受け付けを許可する以前にユーザが割り込みのクリア処理を行う必要があります。

[戻り値]

マクロ	数値	説明
E_OK	0x0	正常終了
E_OS_ACCESS	0x1	本システム・サービスを発行した処理プログラムが所属している OS アプリケーションが Non-Trusted である
E_OS_CALLEVEL	0x2	発行有効範囲外の処理プログラムから発行した
E_OS_ID	0x3	引数 *Application* の指定が不正である
E_OS_STATE	0x7	異常終了 - 対象 OS アプリケーションが本システム・サービスを発行した処理プログラムの所属していない APPLICATION_RESTARTING 状態の OS アプリケーションである - 本システム・サービスを発行した処理プログラムの所属している APPLICATION_RESTARTING 状態の OS アプリケーションに対して，不正な再起動オプション RESTART を指定した - 対象 OS アプリケーションが APPLICATION_TERMINATED 状態である
E_OS_VALUE	0x8	引数 *RestartOption* の指定が不正である
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した

AllowAccess

[概要]

OS アプリケーションの起動

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2）

[指定形式]

```
StatusType AllowAccess ( void );
```

[引数]

なし

[機能]

対象 OS アプリケーション（本システム・サービスを発行した時に動作中の OS アプリケーション）を APPLICATION_RESTATING 状態から APPLICATION_ACCESSIBLE 状態へと遷移させます。

備考　本システム・サービスの発行は，スケーラビリティ・クラス “OsScalabilityClass” で “ スケーラビリティ・クラス 3（SC3）” を定義した場合に限られます。

[戻り値]

マクロ	数値	説明
E_OK	0x0	正常終了
E_OS_CALLEVEL	0x2	発行有効範囲外の処理プログラムから発行した
E_OS_STATE	0x7	本システム・サービスを発行した処理プログラムの所属している OS アプリケーションが APPLICATION_ACCESSIBLE 状態，または APPLICATION_TERMINATED 状態である
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した

GetApplicationState

[概要]

現在状態の獲得

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2），共通フック・ルーチン（StartupHook，ShutdownHook，PostTaskHook，PreTaskHook，ErrorHook，ProtectionHook），固有フック・ルーチン（StartupHook_*OsApplication*，ShutdownHook_*OsApplication*，ErrorHook_*OsApplication*）

[指定形式]

```
StatusType GetApplicationState ( ApplicationType Application, ApplicationStateRefType
Value );
```

[引数]

I/O	引数	説明
I	`ApplicationType` *Application*;	OS アプリケーション識別子
O	`ApplicationStateRefType` *Value*;	獲得した現在状態を格納する領域へのポインタ

[機能]

対象 OS アプリケーション（引数 *Application* で指定された OS アプリケーション）の現在状態を獲得し，引数 *Value* で指定された領域に格納します。

なお，引数 *Value* には，現在状態の種類により，以下の値が格納されます。

マクロ	数値	説明
APPLICATION_ACCESSIBLE	0x0	APPLICATION_ACCESSIBLE 状態
APPLICATION_RESTARTING	0x1	APPLICATION_RESTARTING 状態
APPLICATION_TERMINATED	0x2	APPLICATION_TERMINATED 状態

備考　本システム・サービスの発行は，スケーラビリティ・クラス “OsScalabilityClass” で “ スケーラビリティ・クラス 3（SC3）” を定義した場合に限られます。

[戻り値]

マクロ	数値	説明
E_OK	0x0	正常終了
E_OS_ID	0x3	引数 *Application* の指定が不正である
E_OS_PARAM_POINTER	0x12	引数 *Value* の指定が不正（NULL ポインタ）である
E_OS_ILLEGAL_ADDRESS	0x13	本システム・サービスを発行した処理プログラムは，引数 *Value* で指定された領域に対するアクセス権がない

マクロ	数値	説明
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した

14.4.9 OS 実行管理

以下に，RV850 が提供している OS 実行管理用システム・サービスを示します。

表 14.25 OS 実行管理用システム・サービス

システム・サービス名	機能概要
StartOS	RV850 の起動
ShutdownOS	RV850 の終了
GetActiveApplicationMode	アプリケーション・モードの獲得
ControlIdle	アイドル・ハンドラの動作モードの変更

StartOS

[概要]

RV850 の起動

[発行有効範囲]

ブート処理

[指定形式]

```
void StartOS ( AppModeType Mode );
```

[引数]

I/O	引数	説明
I	AppModeType *Mode*;	アプリケーション・モード 名前： 通常のアプリケーション・モード OSDEFAULTAPPMODE： デフォルト・アプリケーション・モード

[機能]

RV850 の起動処理を行います。

備考 1. 本システム・サービスでは，起動処理として，以下の操作を行います。

- 割り込みの受け付け禁止
  プログラム・ステータス・ワード（PSW）の ID ビットに対する操作
- OS 予約資源の初期化
- カーネル初期化部の呼び出し
  オブジェクトの生成／登録
  オブジェクトの起動
- カテゴリ 1 割り込みの受け付け許可
  プライオリティ・マスク・レジスタ（PMR）の PM*n* ビットに対する操作
  プログラム・ステータス・ワード（PSW）の ID ビットに対する操作
- フック・ルーチンの呼び出し
  StartupHook の呼び出し
  StartupHook_*OsApplication* の呼び出し
- スケジューラの起動
- 割り込みの受け付け許可
  プライオリティ・マスク・レジスタ（PMR）の PM*n* ビットに対する操作

備考 2. 本システム・サービスでは，システム・レジスタに対する操作を行うため，スーパバイザ・モード（PSW の UM ビットが 0）での発行が必須となります。また，システム・プロテクション番号（MCFG0 レジスタの SPID ビット）の設定（4.2 ブート処理 備考 6.）が完了している状態での発行が必須となります。

備考 3. 引数 *Mode* で指定された値がアプリケーション・モード “OsTaskAppModeRef” で定義された値と一致したタスクは，本システム・サービスの発行に伴い，SUSPENDED 状態から READY 状態へと遷移します。

備考 4. 引数 *Mode* で指定された値がアプリケーション・モード “OsAlarmAppModeRef” で定義された値と一致したアラームは，本システム・サービスの発行に伴い，非アクティブ状態からアクティブ状態へと遷移します。

備考 5. 引数 *Mode* で指定された値がアプリケーション・モード “OsScheduleTableAppModeRef” で定義された値と一致したスケジュール・テーブルは，本システム・サービスの発行に伴い，STOPPED 状態から RUNNING 状態へと遷移します。

備考 6. 本システム・サービスでは，EI レベル割り込みマスク・レジスタ（IMR*m*）に対する操作を行いません。
したがって，ユーザは，StartupHook 以降の処理プログラムで IMR*m* に対する操作を行い，割り込みの受け付けを許可する必要があります。

備考 7. ブート処理でクリティカル・セクションを開始したまま，本システム・サービスを発行した場合，クリティカル・セクションを終了し，備考 1 の起動処理を行います。

［戻り値］

なし

ShutdownOS

[概要]

RV850 の終了

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2），共通フック・ルーチン（StartupHook，ErrorHook），固有フック・ルーチン（StartupHook_*OsApplication*，ErrorHook_*OsApplication*）

[指定形式]

```
void ShutdownOS ( StatusType Error );
```

[引数]

I/O	引数	説明
I	`StatusType` *Error*;	引き継ぎデータ

[機能]

RV850 の終了処理を行います。

備考 1. 本システム・サービスでは，終了処理として，以下の操作を行います。

- 割り込みの受け付け禁止
  プログラム・ステータス・ワード（PSW）の ID ビットに対する操作
  プライオリティ・マスク・レジスタ（PMR）の PM*n* ビットに対する操作
  EI レベル割り込みマスク・レジスタ（IMR*m*）に対する操作
- フック・ルーチンの呼び出し
  引数 *Error* で指定された値を引き継ぎデータとした ShutdownHook_*OsApplication* の呼び出し
  引数 *Error* で指定された値を引き継ぎデータとした ShutdownHook の呼び出し
- 空処理の呼び出し
  空処理の無限実行（RV850 を擬似的な HALT 状態へと移行）

備考 2. 空処理からの復帰は，ハードウエア・リセットの発生に限られます。

備考 3. 本システム・サービスを Non-Trusted な OS アプリケーションに所属している処理プログラムから発行した場合，何も処理は行われず，エラーとしても扱われません。

備考 4. 本システム・サービスを発行有効範囲外の処理プログラムから発行した場合，何も処理は行われず，エラーとしても扱われません。

備考 5. RV850 では，共通フック・ルーチン（ProtectionHook）が未定義の場合，スタック・オーバフローを検知した際には，引数 *Error* に E_OS_STACKFAULT（0x16）を指定した本システム・サービスの発行を行います。

備考 6. RV850 では，割り込みサービス・ルーチン情報で未定義の割り込みが発生した場合，引数 *Error* に E_OS_SYS_ILLEGAL_EXCEPTION（0x1F）を指定した本システム・サービスの発行を行います。

備考 7. AUTOSAR 仕様では，スケーラビリティ・クラス “OsScalabilityClass” がスケーラビリティ・クラス 1（SC1）の場合，共通フック・ルーチン PostTaskHook を呼び出すのか否かの規定が行われていません。そこで，RV850 では，スケーラビリティ・クラス “OsScalabilityClass” の定義内容に依存することなく，本システム・サービスから共通フック・ルーチン PostTaskHook の呼び出しは行いません。

［戻り値］

なし

GetActiveApplicationMode

[概要]

アプリケーション・モードの獲得

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2），アラーム・コールバック，共通フック・ルーチン，固有フック・ルーチン，クリティカル・セクション

[指定形式]

```
AppModeType GetActiveApplicationMode ( void );
```

[引数]

なし

[機能]

StartOS の引数 *OsAppMode* で指定したアプリケーション・モードを獲得し，戻り値として返します。

[戻り値]

マクロ	数値	説明
−	0x0 ～ 0x7E	獲得したアプリケーション・モード

StartOS の引数 *OsAppMode* に不正な値が指定された場合，上記以外の値が返されることがあります。

ControlIdle

[概要]

アイドル・ハンドラの動作モードの変更

[発行有効範囲]

タスク，割り込みサービス・ルーチン（カテゴリ 2）

[指定形式]

```
StatusType ControlIdle ( CoreIdType CoreID, IdleModeType IdleMode );
```

[引数]

I/O	引数	説明
I	`CoreIdType` *CoreID*`;`	コア識別子
I	`IdleModeType` *IdleMode*`;`	アイドル・ハンドラの動作モード

[機能]

本システム・サービスを発行したコアで動作するアイドル・ハンドラの動作モードを，本システム・サービスの引数 *IdleMode* に指定された値に変更します。本システム・サービスで指定したアイドル・ハンドラの動作モードは，アイドル・ハンドラの引数 *idlemode* に設定されます。

備考 1. RV850 では，引数 *CoreID* の指定を無視し，常に本システム・サービスを発行した処理プログラムが配置されたコアのアイドル・ハンドラを対象とします。

備考 2. StartOS を発行してから，アイドル・ハンドラが起動する間に，本システム・サービスを複数回発行した場合，最後に発行した本システム・サービスで指定されたアイドル・ハンドラの動作モードが，アイドル・ハンドラの引数 *idlemode* に設定されます。

備考 3. AUTOSAR 仕様では，デバイス依存の省電力動作モードを設定できる旨の規定が行われていますが，RV850 では，デバイス依存の省電力動作モードを規定していません。RV850 では，引数 *idlemode* はアイドル・ハンドラの動作モードと規定しています。アイドル・ハンドラの動作モードに対応する処理は，ユーザがアイドル・ハンドラに記述してください。

[戻り値]

マクロ	数値	説明
E_OK	0x0	正常終了
E_OS_CALLEVEL	0x2	発行有効範囲外の処理プログラムから発行した
E_OS_ID	0x3	引数 *IdleMode* の指定が不正である
E_OS_DISABLEDINT	0x15	クリティカル・セクションから発行した

14.4.10 ユーティリティ関数

以下に，RV850 が提供しているユーティリティ関数を示します。

表 14.26 ユーティリティ関数

ユーティリティ関数名	機能概要
InitApplicationInterrupts	割り込みの受け付けを許可
_kernel_fv0_InitializeIntService	割り込み管理用システム・サービスの発行許可
OSIllegalException_SystemRegister_ExcCode	レジスタ値（EIIC，または FEIC）の獲得
OSIllegalException_SystemRegister_ExcPC	レジスタ値（EIPC，または FEPC）の獲得
OSErrorGetServiceId	システム・サービス識別子の獲得
OSError_SystemService_Parameter	引数の獲得

備考 AUTOSAR 仕様では，ユーティリティ関数 InitApplicationInterrupts，_kernel_fv0_InitializeIntService，OSIllegalException_SystemRegister_ExcCode，OSIllegalException_SystemRegister_ExcPC の規定が行われていません。
該当ユーティリティ関数は，RV850 が独自に追加したユーティリティ関数です。

InitApplicationInterrupts

[概要]

割り込みの受け付けを許可

[発行有効範囲]

タスク

[指定形式]

```
void InitApplicationInterrupts ( void );
```

[引数]

なし

[機能]

本ユーティリティ関数を発行した処理プログラムの所属している OS アプリケーションに対する割り込みの受け付けを許可します。

備考 1. 本ユーティリティ関数の発行は，スケーラビリティ・クラス “OsScalabilityClass” で “ スケーラビリティ・クラス 3（SC3）” を定義した場合に限られます。

備考 2. 本ユーティリティ関数では，割り込みの受け付け許可処理として，EI レベル割り込みマスク・レジスタ（IMR*m*）に対する操作を行います。
操作対象となる IMR*m* は，該当 OS アプリケーションに登録されている割り込みサービス・ルーチンに対応したものとなります。

備考 3. 本ユーティリティ関数は，引数 *RestartOption* に RESTART（0x1）を指定して TerminateApplication を発行した際，SUSPENDED 状態から READY 状態へと遷移するタスク（タスク識別子 “OsRestartTask”）で発行することを想定しています。

備考 4. 本ユーティリティ関数を発行有効範囲外の処理プログラムから発行した場合，何も処理は行われず，エラーとしても扱われません。

備考 5. RV850 では，OS アプリケーションを終了する際，および OS アプリケーションを再起動する際，受け付けが保留されている割り込みのクリアを行いません。
したがって，OS アプリケーションを再起動した際，受け付けが保留されている割り込みのクリアが必要な場合には，本ユーティリティ関数の発行以前にユーザが割り込みのクリア処理を行う必要があります。

[戻り値]

なし

_kernel_fv0_InitializeIntService

[概要]

割り込み管理用システム・サービスの発行許可

[発行有効範囲]

ブート処理

[指定形式]

```
void _kernel_fv0_InitializeIntService ( void );
```

[引数]

なし

[機能]

StartOS の発行前に RV850 が提供している割り込み管理用システム・サービスを発行可能とするための初期化処理を行います。

備考 1. 本ユーティリティ関数の発行は，StartOS を発行する以前に限られます。

備考 2. 本ユーティリティ関数では，RV850 の内部データに対する操作を行うため，スーパバイザ・モード（PSW の UM ビットが 0）での発行が必須となります。

[戻り値]

なし

OSIllegalException_SystemRegister_ExcCode

[概要]

レジスタ値（EIIC，または FEIC）の獲得

[発行有効範囲]

共通フック・ルーチン（ShutdownHook），固有フック・ルーチン（ShutdownHook_*OsApplication*）

[指定形式]

```
SystemRegisterType OSIllegalException_SystemRegister_ExcCode ( void );
```

[引数]

なし

[機能]

レジスタ値（EIIC，または FEIC）を獲得し，戻り値として返します。

備考　　本ユーティリティ関数の発行は，処理プログラムでシステム・サービスを発行する以前に限られます。

[戻り値]

マクロ	数値	説明
－	0x0 ～ 0xFFFFFFFF	獲得したシステム・レジスタ値

OSIllegalException_SystemRegister_ExcPC

[概要]

レジスタ値（EIPC，または FEPC）の獲得

[発行有効範囲]

共通フック・ルーチン（ShutdownHook），固有フック・ルーチン（ShutdownHook_*OsApplication*）

[指定形式]

```
SystemRegisterType OSIllegalException_SystemRegister_ExcPC ( void );
```

[引数]

なし

[機能]

レジスタ値（EIPC，または FEPC）を獲得し，戻り値として返します。

備考　　本ユーティリティ関数の発行は，処理プログラムでシステム・サービスを発行する以前に限られます。

[戻り値]

マクロ	数値	説明
－	0x0 ～ 0xFFFFFFFF	獲得したシステム・レジスタ値

OSErrorGetServiceId

[概要]

システム・サービス識別子の獲得

[発行有効範囲]

共通フック・ルーチン（ErrorHook），固有フック・ルーチン（ErrorHook_*OsApplication*）

[指定形式]

```
OSServiceIdType OSErrorGetServiceId ( void );
```

[引数]

なし

[機能]

対象システム・サービス（本ユーティリティ関数を発行した処理プログラムを呼び出す要因となったシステム・サービス）のシステム・サービス識別子を獲得し，戻り値として返します。

備考 1. 本ユーティリティ関数の発行は，処理プログラムでシステム・サービスを発行する以前に限られます。

備考 2. 本ユーティリティ関数を発行する処理プログラム（ErrorHook，ErrorHook_Application）の呼び出し要因が “ アラームおよびスケジュール・テーブルの満了 ” の場合，本ユーティリティ関数の戻り値は OSServiceId_IncrementCounter となります。

備考 3. 本ユーティリティ関数を発行する処理プログラム（ErrorHook，ErrorHook_Application）の呼び出し要因が “TerminateTask を発行せずタスクを終了 ” の場合，本ユーティリティ関数の戻り値は OSServiceId_ExitTask となります。

備考 4. 本ユーティリティ関数を発行する処理プログラム（ErrorHook，ErrorHook_Application）の呼び出し要因が “ リソースを獲得したままカテゴリ 2 の割り込みサービス・ルーチンを終了，およびクリティカル・セクションを終了せずカテゴリ 2 の割り込みサービス・ルーチンを終了 ” の場合，本ユーティリティ関数の戻り値は OSServiceId_ExitIsr となります。

[戻り値]

マクロ	数値	説明
OSServiceId_GetApplicationID	0x0	GetApplicationID のシステム・サービス識別子
OSServiceId_GetISRID	0x1	GetISRID のシステム・サービス識別子
OSServiceId_CallTrustedFunction	0x2	CallTrustedFunction のシステム・サービス識別子
OSServiceId_CheckISRMemoryAccess	0x3	CheckISRMemoryAccess のシステム・サービス識別子
OSServiceId_CheckTaskMemoryAccess	0x4	CheckTaskMemoryAccess のシステム・サービス識別子
OSServiceId_CheckObjectAccess	0x5	CheckObjectAccess のシステム・サービス識別子
OSServiceId_CheckObjectOwnership	0x6	CheckObjectOwnership のシステム・サービス識別子
OSServiceId_StartScheduleTableRel	0x7	StartScheduleTableRel のシステム・サービス識別子
OSServiceId_StartScheduleTableAbs	0x8	StartScheduleTableAbs のシステム・サービス識別子

マクロ	数値	説明
OSServiceId_StopScheduleTable	0x9	StopScheduleTable のシステム・サービス識別子
OSServiceId_NextScheduleTable	0xA	NextScheduleTable のシステム・サービス識別子
OSServiceId_GetScheduleTableStatus	0xE	GetScheduleTableStatus のシステム・サービス識別子
OSServiceId_IncrementCounter	0xF	IncrementCounter のシステム・サービス識別子
OSServiceId_GetCounterValue	0x10	GetCounterValue のシステム・サービス識別子
OSServiceId_GetElapsedValue	0x11	GetElapsedValue のシステム・サービス識別子
OSServiceId_TerminateApplication	0x12	TerminateApplication のシステム・サービス識別子
OSServiceId_AllowAccess	0x13	AllowAccess のシステム・サービス識別子
OSServiceId_GetApplicationState	0x14	GetApplicationState のシステム・サービス識別子
OSServiceId_ControlIdle	0x1D	ControlIdle のシステム・サービス識別子
OSServiceId_GetCurrentApplicationID	0x27	GetCurrentApplicationID のシステム・サービス識別子
OSServiceId_StartOS	0x40	StartOS のシステム・サービス識別子
OSServiceId_ShutdownOS	0x41	ShutdownOS のシステム・サービス識別子
OSServiceId_GetActiveApplicationMode	0x42	GetActiveApplicationMode のシステム・サービス識別子
OSServiceId_ActivateTask	0x43	ActivateTask のシステム・サービス識別子
OSServiceId_TerminateTask	0x44	TerminateTask のシステム・サービス識別子
OSServiceId_ChainTask	0x45	ChainTask のシステム・サービス識別子
OSServiceId_Schedule	0x46	Schedule のシステム・サービス識別子
OSServiceId_GetTaskID	0x47	GetTaskID のシステム・サービス識別子
OSServiceId_GetTaskState	0x48	GetTaskState のシステム・サービス識別子
OSServiceId_EnableAllInterrupts	0x49	EnableAllInterrupts のシステム・サービス識別子
OSServiceId_DisableAllInterrupts	0x4A	DisableAllInterrupts のシステム・サービス識別子
OSServiceId_ResumeAllInterrupts	0x4B	ResumeAllInterrupts のシステム・サービス識別子
OSServiceId_SuspendAllInterrupts	0x4C	SuspendAllInterrupts のシステム・サービス識別子
OSServiceId_ResumeOSInterrupts	0x4D	ResumeOSInterrupts のシステム・サービス識別子
OSServiceId_SuspendOSInterrupts	0x4E	SuspendOSInterrupts のシステム・サービス識別子
OSServiceId_GetResource	0x4F	GetResource のシステム・サービス識別子
OSServiceId_ReleaseResource	0x50	ReleaseResource のシステム・サービス識別子
OSServiceId_SetEvent	0x51	SetEvent のシステム・サービス識別子
OSServiceId_ClearEvent	0x52	ClearEvent のシステム・サービス識別子
OSServiceId_GetEvent	0x53	GetEvent のシステム・サービス識別子
OSServiceId_WaitEvent	0x54	WaitEvent のシステム・サービス識別子
OSServiceId_GetAlarmBase	0x55	GetAlarmBase のシステム・サービス識別子
OSServiceId_GetAlarm	0x56	GetAlarm のシステム・サービス識別子
OSServiceId_SetRelAlarm	0x57	SetRelAlarm のシステム・サービス識別子

マクロ	数値	説明
OSServiceId_SetAbsAlarm	0x58	SetAbsAlarm のシステム・サービス識別子
OSServiceId_CancelAlarm	0x59	CancelAlarm のシステム・サービス識別子
OSServiceId_ExitTask	0x62	TerminateTask を発行せずタスクを終了
OSServiceId_ExitIsr	0x65	リソースを獲得したままカテゴリ 2 の割り込みサービス・ルーチンを終了，およびクリティカル・セクションを終了せずカテゴリ 2 の割り込みサービス・ルーチンを終了

OSError_*SystemService_Parameter*

[概要]

引数の獲得

[発行有効範囲]

共通フック・ルーチン（ErrorHook），固有フック・ルーチン（ErrorHook_*OsApplication*）

[指定形式]

```
TaskType OSError_ActivateTask_TaskID ( void );
TaskType OSError_ChainTask_TaskID ( void );
TaskRefType OSError_GetTaskID_TaskID ( void );
TaskType OSError_GetTaskState_TaskID ( void );
TaskStateRefType OSError_GetTaskState_State ( void );
ResourceType OSError_GetResource_ResID ( void );
ResourceType OSError_ReleaseResource_ResID ( void );
TaskType OSError_SetEvent_TaskID ( void );
EventMaskType OSError_SetEvent_Mask ( void );
EventMaskType OSError_ClearEvent_Mask ( void );
TaskType OSError_GetEvent_TaskID ( void );
EventMaskRefType OSError_GetEvent_Event ( void );
EventMaskType OSError_WaitEvent_Mask ( void );
CounterType OSError_IncrementCounter_CounterID ( void );
CounterType OSError_GetCounterValue_CounterID ( void );
TickRefType OSError_GetCounterValue_Value ( void );
CounterType OSError_GetElapsedValue_CounterID ( void );
TickRefType OSError_GetElapsedValue_Value ( void );
TickRefType OSError_GetElapsedValue_ElapsedValue ( void );
AlarmType OSError_GetAlarmBase_AlarmID ( void );
AlarmBaseRefType OSError_GetAlarmBase_Info ( void );
AlarmType OSError_GetAlarm_AlarmID ( void );
TickRefType OSError_GetAlarm_Tick ( void );
AlarmType OSError_SetRelAlarm_AlarmID ( void );
TickType OSError_SetRelAlarm_increment ( void );
TickType OSError_SetRelAlarm_cycle ( void );
AlarmType OSError_SetAbsAlarm_AlarmID ( void );
```

```
TickType OSError_SetAbsAlarm_start ( void );
TickType OSError_SetAbsAlarm_cycle ( void );
AlarmType OSError_CancelAlarm_AlarmID ( void );
ScheduleTableType OSError_StartScheduleTableRel_ScheduleTableID ( void );
TickType OSError_StartScheduleTableRel_Offset ( void );
ScheduleTableType OSError_StartScheduleTableAbs_ScheduleTableID ( void );
TickType OSError_StartScheduleTableAbs_Start ( void );
ScheduleTableType OSError_StopScheduleTable_ScheduleTableID ( void );
ScheduleTableType OSError_NextScheduleTable_ScheduleTableID_From ( void );
ScheduleTableType OSError_NextScheduleTable_ScheduleTableID_To ( void );
ScheduleTableType OSError_GetScheduleTableStatus_ScheduleTableID ( void );
ScheduleTableStatusRefType OSError_GetScheduleTableStatus_ScheduleStatus ( void );
TrustedFunctionIndexType OSError_CallTrustedFunction_FunctionIndex ( void );
TrustedFunctionParameterRefType OSError_CallTrustedFunction_FunctionParams ( void );
ApplicationType OSError_TerminateApplication_Application ( void );
RestartType OSError_TerminateApplication_RestartOption ( void );
ApplicationType OSError_GetApplicationState_Application ( void );
ApplicationStateRefType OSError_GetApplicationState_Value ( void );
CoreIdType OSError_ControlIdle_CoreID ( void );
IdleModeType OSError_ControlIdle_IdleMode ( void );
```

備考 1. ユーティリティ関数名の *SystemService* にはシステム・サービス名を，*Parameter* には引数名を指定します。

備考 2. ユーティリティ関数名の *SystemService* に指定可能なシステム・サービス名は，引数を持つ StatusType 型のシステム・サービスに限られます。

[引数]

なし

[機能]

対象システム・サービス（本ユーティリティ関数を発行した処理プログラムを呼び出す要因となったシステム・サービス）の引数に指定した値，またはポインタのアドレスを獲得し，戻り値として返します。

備考 本ユーティリティ関数を発行する処理プログラム（ErrorHook，ErrorHook_Application）の呼び出し要因が “ システム・サービスの異常終了 ” 以外の場合，本ユーティリティ関数の戻り値は不定値となります。

[戻り値]

戻り値についての詳細は，「14.4 システム・サービス・リファレンス」に記載されている各システム・サービスの “ [引数] ” 欄を参照してください。

A. コンフィギュレータ

本付録では，コンフィギュレータについて解説しています。

A.1 概　　要

コンフィギュレータは，CF ファイルを入力ファイルとして読み込んだのち，情報ファイル（SIT ファイル，ENTRY ファイル，カーネル・マクロ・ファイル）を出力するユーティリティ・ツールです。

A.2 起動方法

以下に，コンフィギュレータをコマンド・プロンプトから起動する際の方法を示します。
なお，“C:￥>” はコマンド・プロンプトを，“ △ ” は Space キーの入力を，“ [　] ” で囲まれた部分は省略可能な起動オプションであることを示しています。

```
C:￥> Os_Configurator.exe △ [cf_file] △ [@cmd_file] △ [-o △ sit_file] △ [-no] △ [-e △ entry_file] △ [-ne] △ [-nk] △ [-I △ path_name] △ [-nsc] △ [-V] △ [-help]
```

以下に，各起動オプションの詳細を示します。

(1) *cf_file*
cf_file で指定された CF ファイル（ARXML 形式，OIL 形式）を入力ファイルとして扱います。
なお，*cf_file* として指定可能な文字数は，255 文字までに限られます。

備考 1. *cf_file* に “Program Files” などといった半角スペースを含む文字列を指定する際には，*cf_file* をダブルクォーテーションでくくる必要があります。

備考 2. コンフィギュレータは，拡張子が .arxml，.xml の場合は ARXML 形式として，拡張子が .oil の場合は OIL 形式として *cf_file* を扱います。
なお，*cf_file* に指定可能な拡張子は，.arxml，.xml，.oil に限られます。

備考 3. *cf_file* は，以下のように複数の指定が可能ですが，最後に指定した CF ファイルのみが有効な CF ファイルとして扱われ，他ファイルは無効なファイルとして扱われます。
したがって，以下の例では，cf_file3.oil が有効な CF ファイルとなり，他ファイル cf_file1.arxml，cf_file2.xml は無視されます。

```
C:￥> Os_Configurator.exe △ cf_file1.arxml △ cf_file2.xml △ cf_file3.oil
```

備考 4. コンフィギュレータは，*cf_file* が ARXML 形式の場合，その記述内容（設定項目）が AUTOSAR_RENESAS_OS_ECUConfigurationParameters.arxml に沿った内容（OIL 形式における設定項目と同等）であるものとして動作します。

備考 5. AUTOSAR 仕様では，OSEK Implementation Language Version 2.5 の仕様に則って記述された OIL 形式のファイルを入力ファイルとして指定可能とする旨の規定が行われていませんが，RV850 では，該当形式のファイルを入力ファイルとして指定可能としています。

【省略時解釈】
Os_Configurator.exe を起動したフォルダに対して以下のファイルが存在しているか否かの検索を行い，該当ファイルが存在していた場合には，該当ファイルを CF ファイルとして扱います。

- Os_Config.arxml（検索順位：第 1 位）
- Os_Config.xml（検索順位：第 2 位）
- Os_Config.oil（検索順位：第 3 位）

(2) *@cmd_file*
cmd_file で指定されたファイルをコマンド・ファイルとして扱います。
なお，*cmd_file* として指定可能な文字数は，255 文字までに限られます。

備考 *cmd_file* に “Program Files” などといった半角スペースを含む文字列を指定する際には，*cmd_file* をダブルクォーテーションでくくる必要があります。

(3) -o △ *sit_file*
sit_file というファイル名で SIT ファイルを出力します。
なお，*sit_file* として指定可能な文字数は，255 文字までに限られます。

備考 1. *sit_file* に “Program Files” などといった半角スペースを含む文字列を指定する際には，*sit_file* をダブルクォーテーションでくくる必要があります。

備考 2. -o △ *sit_file* は，以下のように複数の指定が可能ですが，最初に指定した -o △ *sit_file* のみが有効な起動オプションとして扱われ，他の -o △ *sit_file* は無効な起動オプションとして扱われます。
したがって，以下の例では，-o △ sit_file1.c が有効な起動オプションとなり，-o △ sit_file2.c，-o △ sit_file3.c は無視されます。

```
C:￥> Os_Configurator.exe △ -o △ sit_file1.c △ -o △ sit_file2.c △ -o △
sit_file3.c △ cf_file.oil
```

【省略時解釈】
“-o △ *sit_file*” の指定を省略した場合は “-o △ Os_Cfg.c” を指定したものとして処理が行われます。

(4) -no
SIT ファイルを出力しません。

(5) -e △ *entry_file*
entry_file というファイル名で ENTRY ファイルを出力します。
なお，*entry_file* として指定可能な文字数は，255 文字までに限られます。

備考 1. *entry_file* に “Program Files” などといった半角スペースを含む文字列を指定する際には，*entry_file* をダブルクォーテーションでくくる必要があります。

備考 2. -e △ *entry_file* は，以下のように複数の指定が可能ですが，最初に指定した -e △ *entry_file* のみが有効な起動オプションとして扱われ，他の -e △ *entry_file* は無効な起動オプションとして扱われます。
したがって，以下の例では，-e △ entry_file1.850 が有効な起動オプションとなり，-e △ entry_file2.850，-e △ entry_file3.850 は無視されます。

```
C:￥> Os_Configurator.exe △ -e △ entry_file1.850 △ -e △ entry_file2.850 △
-e △ entry_file3.850 △ cf_file.oil
```

【省略時解釈】
“-e △ *entry_file*” の指定を省略した場合は “-e △ Os_CfgEntry.850” を指定したものとして処理が行われます。

(6) -ne
ENTRY ファイルを出力しません。

(7) -nk
カーネル・マクロ・ファイルを出力しません。

【省略時解釈】
カーネル・マクロ・ファイル “Os_Cfg.h” を出力します。

(8) -I △ *path_name*
CF ファイル（OIL）内に記述したインクルード・ファイルの検索対象フォルダを指定します。
なお，*path_name* として指定可能な文字数は，255 文字までに限られます。

備考 1. *path_name* に “Program Files” などといった半角スペースを含む文字列を指定する際には，*path_name* をダブルクォーテーションでくくる必要があります。

備考 2. 本起動オプションは，以下のように複数の指定（最大：255）が可能です。

```
C:￥> Os_Configurator.exe △ -I △ path_name1 △ -I △ path_name2 △ -I △
path_name3 △ cf_file.oil
```

備考 3. インクルード・ファイルの検索順序は，CF ファイル（OIL）での記述形式により，以下のように異なります。

【“#include △ <*inc_file*>” 形式の場合】

- *path_name* で指定されたフォルダ
- *cf_file* を格納しているフォルダ
- Os_Configurator.exe を起動したフォルダ

【“#include △ "*inc_file*"” 形式の場合】

- *cf_file* を格納しているフォルダ
- Os_Configurator.exe を起動したフォルダ
- *path_name* で指定されたフォルダ

【省略時解釈】
インクルード・ファイルの検索は，以下の順序で行われます。

- *cf_file* を格納しているフォルダ
- Os_Configurator.exe を起動したフォルダ

(9) -nsc
CF ファイル（OIL）内に記述された C++ スタイルのコメント “//” をエラーとして扱います。

【省略時解釈】
CF ファイル（OIL）内に記述された C++ スタイルのコメント “//” をエラーとして扱いません。

(10) -V
コンフィギュレータのバージョン情報を出力します。

備考 本起動オプションを指定した場合，他起動オプションはすべて無効となり，情報ファイルの出力が抑制されます。

【省略時解釈】
バージョン情報を出力しません。

(11) -help
コンフィギュレータの起動オプションに関する説明（種類，用途など）を出力します。

備考 本起動オプションを指定した場合，他起動オプションはすべて無効となり，情報ファイルの出力が抑制されます。

【省略時解釈】
起動オプションに関する説明を出力しません。

A.2.1 コマンド・ファイル

コンフィギュレータでは，コマンド・ライン上で指定可能な起動オプションの文字数制限を解消する目的からコマンド・ファイル対応を行っています。
以下に，コマンド・ファイルの記述形式を示します。

(1) コメント
“#” から行末までがコメントとして扱われます。
なお，コメントについては，SJIS コード，または EUC コードでの記述が可能です。

(2) 起動オプションの区切り
各起動オプションについては，改行，半角スペース，タブのいずれかを用いて区切ります。
なお，“-o △ *sit_file*”，“-e △ *entry_file*”，“-I △ *path_name*” といった *-xxx* 部とパラメータ部から構成される起動オプションについても，*-xxx* 部とパラメータ部を改行，半角スペース，タブのいずれかを用いて区切ります。

(3) 半角スペースを含む文字列
“*cf_file*”，“*cmd_file*”，“-o △ *sit_file*”，“-e △ *entry_file*”，“-I △ *path_name*” といった起動オプションにおいて，半角スペースを含む文字列（Program Files など）を指定する際には，該当部をダブルクォーテーションでくくる必要があります。

図 A.1 コマンド・ファイルの記述例

```
# コマンド・ファイル
C:￥sample￥src￥cf_file.oil          # CF ファイル
-o C:￥sample￥src￥sit_file.c        # SIT ファイル
-e C:￥sample￥src￥entry_file.850    # ENTRY ファイル
-I "C:￥Program Files￥sample￥inc"   # インクルード・パス
```

A.3 コマンド入力例

以下に，コンフィギュレータのコマンド入力例を示します。
なお，“C:￥>” はコマンド・プロンプトを，“ △ ” は Space キーの入力を示しています。

(1) C:￥Program Files￥sample のフォルダに格納されている cf_file.oil を CF ファイルとして読み込んだのち，情報ファイル（SIT ファイル：Os_Cfg.c，ENTRY ファイル：Os_CfgEntry.850，カーネル・マクロ・ファイル：Os_Cfg.h）を Os_Configurator.exe を起動したフォルダに出力します。
なお，cf_file.oil 内に定義されたインクルード・ファイルの検索は，以下の順序で行われます。

- cf_file.oil を格納しているフォルダ
- Os_Configurator.exe を起動したフォルダ

```
C:￥> Os_Configurator.exe △ "C:￥Program Files￥sample￥cf_file.oil"
```

(2) Os_Configurator.exe を起動したフォルダから相対パスで ..￥sample に格納されている cf_file.oil を入力ファイルとして読み込んだのち，情報ファイル（SIT ファイル：sit_file.c）を Os_Configurator.exe を起動したフォルダから相対パスで ..￥sample のフォルダに出力します。
なお，cf_file.oil 内に定義されたインクルード・ファイルの検索は，以下の順序で行われます。

- cf_file.oil を格納しているフォルダ
- Os_Configurator.exe を起動したフォルダ

```
C:￥> Os_Configurator.exe△..￥sample￥cf_file.oil△-o△..￥sample￥sit_file.c△-ne△
-nk
```

(3) Os_Configurator.exe を起動したフォルダに格納されている cf_file.oil を CF ファイルとして読み込んだのち，情報ファイル（SIT ファイル：Os_Cfg.c，ENTRY ファイル：Os_CfgEntry.850，カーネル・マクロ・ファイル：Os_Cfg.h）を Os_Configurator.exe を起動したカレント・フォルダに出力します。
なお，cf_file.oil 内に定義されたインクルード・ファイルの検索は，以下の順序で行われます。

【“#include △ <*inc_file*>” 形式の場合】

- Os_Configurator.exe を起動したフォルダから相対パスで ..￥sample￥inc のフォルダ
- cf_file.oil を格納しているフォルダ
- Os_Configurator.exe を起動したフォルダ

【“#include △ "*inc_file*"” 形式の場合】

- cf_file.oil を格納しているフォルダ
- Os_Configurator.exe を起動したフォルダ
- Os_Configurator.exe を起動したフォルダから相対パスで ..￥sample￥inc のフォルダ

```
C:￥> Os_Configurator.exe △ cf_file.oil △ -I △ ..￥sample￥inc
```

(4) コンフィギュレータのバージョン情報を出力します。

```
C:￥> Os_Configurator.exe △ -V
```

A.4 メッセージ

メッセージは，コンフィギュレータが処理を実行中に“ユーザに通知すべき情報（起動オプションの指定が不正，数値の指定が有効範囲外など）”を検出した際に生成され，出力されます。
なお，メッセージは，通知すべき情報の重要度により，以下の 3 種類に分類されます。

(1) E：フェイタル・エラー
一定数以上のフェイタル・エラーが発生した際に処理を中止します。
情報ファイルを出力しません。

(2) F：アボート・エラー
処理を中止します。
情報ファイルを出力しません。

(3) W：ワーニング
処理を継続します。
情報ファイルを出力します。
出力された情報ファイルの内容は，ユーザが意図したものとは異なる可能性があります。

A.4.1 フェイタル・エラー

以下に，フェイタル・エラーを検出した際に出力されるメッセージを示します。
なお，メッセージ内のイタリック書体については，フェイタル・エラーを検出した際に決定します。

表 A.1 フェイタル・エラー

E2001	［メッセージ］	Nesting of include is greater than 8.
	［説明］	インクルード・ファイルのネスト数が 8 を越えています。
	［対処方法］	複数のインクルード・ファイルをまとめるなどして，インクルード・ファイルのネスト数を 8 以下に抑えてください。
E2002	［メッセージ］	Object name is longer than 255 characters.
	［説明］	オブジェクト名が制限文字数“255 文字 ”を越えています。
	［対処方法］	オブジェクト名を 255 文字以下に変更してください。
E2003	［メッセージ］	Syntax error.
	［説明］	文法上の不正があります。
	［対処方法］	文法的な記述ミスを確認してください。
E2005	［メッセージ］	*information* is expected.
	［説明］	省略不可の情報 *information* が記述されていません。
	［対処方法］	情報 *information* の記述を行ってください。
E2006	［メッセージ］	Illegal description (*string*).
	［説明］	無効な文字列 *string* が指定されています。
	［対処方法］	有効な文字列を指定してください。
E2007	［メッセージ］	Out of range (*value*).
	［説明］	数値 *value* の指定が不正です。
	［対処方法］	有効範囲内の数値を指定してください。
E2008	［メッセージ］	*xxx* sub part is expected when *yyy* is *zzz*.
	［説明］	*yyy* に *zzz* を指定した場合，*xxx* の記述を省略することはできません。
	［対処方法］	*xxx* の記述を行ってください。

E2009	［メッセージ］	*xxx* sub part is unnecessary when *yyy* is *zzz*.
	［説明］	*yyy* に *zzz* を指定した場合，*xxx* を指定することはできません。
	［対処方法］	*xxx* を削除してください。
E2010	［メッセージ］	There is no required item in *xxx*.
	［説明］	*xxx* で省略不可の項目が定義されていません。
	［対処方法］	該当項目を追記してください。
E3000	［メッセージ］	Identifier is conflict (*name*).
	［説明］	名前 *name* が重複しています。
	［対処方法］	どちらか一方の名前を変更してください。
E3001	［メッセージ］	*xxx* is multiple defined (*yyy*).
	［説明］	*yyy* で *xxx* が多重定義されています。
	［対処方法］	*xxx* の多重定義を回避してください。
E3002	［メッセージ］	Too many objects (*xxx*).
	［説明］	オブジェクト *xxx* の定義総数が制限定義数を越えています。
	［対処方法］	制限定義数内でオブジェクト *xxx* を定義してください。
E3005	［メッセージ］	*information* is not aligned.
	［説明］	情報 *information* がアラインされていません。
	［対処方法］	規定された数でアラインされた数値を指定してください。
E3007	［メッセージ］	RV850 doesn't support a value larger than 32 bit size.
	［説明］	0xFFFFFFFF より大きな値が指定されています。
	［対処方法］	0xFFFFFFFF 以下の値を指定してください。
E3102	［メッセージ］	*xxx* is not defined in *yyy*.
	［説明］	識別子 *yyy* に対応した情報 *xxx* が定義されていません。
	［対処方法］	情報 *xxx* の定義してください。
E3103	［メッセージ］	*xxx* can not defined in *yyy*.
	［説明］	*yyy* に *xxx* を定義することはできません。
	［対処方法］	*xxx* を削除してください。
E3104	［メッセージ］	Timing Protection is defined for Category 1 ISR (*xxx*).
	［説明］	カテゴリ 1 の割り込みサービス・ルーチンに対してタイミング保護に関する定義が行われています。
	［対処方法］	割り込みサービス・ルーチンのカテゴリを 1 から 2 へ変更，またはタイミング保護に関する定義を削除してください。
E3105	［メッセージ］	Setting in a memory area is the illegal value.
	［説明］	メモリ領域の指定範囲が不正です。
	［対処方法］	適切なアドレスを指定してください。

E3106	［メッセージ］	The size and the termination address can't be specified at the same time.
	［説明］	サイズと終端アドレスの両方が記述されています。
	［対処方法］	サイズ，または終端アドレスのどちらか一方を使用してメモリ領域を指定してください。
E3107	［メッセージ］	*xxx* is lower than *yyy*.
	［説明］	*yyy* より小さな値が *xxx* に指定されています。
	［対処方法］	*xxx* には，*yyy* 以上の値を指定してください。
E3108	［メッセージ］	*xxx* is higher than *yyy*.
	［説明］	*yyy* より大きな値が *xxx* に指定されています。
	［対処方法］	*xxx* には，*yyy* 以下の値を指定してください。
E3109	［メッセージ］	There are a lot of specification of *xxx* (*yyy*).
	［説明］	*yyy* における *xxx* の定義総数が多すぎます。
	［対処方法］	*xxx* の定義数を減らしてください。
E3110	［メッセージ］	Internal resource is multiple defined (*xxx*).
	［説明］	インターナル・リソースが多重定義されています。
	［対処方法］	インターナル・リソースの定義を 1 個にしてください。
E3112	［メッセージ］	*xxx* is OS reserved.
	［説明］	*xxx* は予約語のため，使用することができません。
	［対処方法］	予約語を避けた名前に変更してください。
E3113	［メッセージ］	*xxx* doesn't belong to *yyy*.
	［説明］	オブジェクト *xxx* が OS アプリケーション *yyy* に所属していません。
	［対処方法］	オブジェクト *xxx* を OS アプリケーション *yyy* に所属させてください。
E4002	［メッセージ］	*xxx* must be *yyy* for *zzz*.
	［説明］	*zzz* の条件では *xxx* に *yyy* を定義する必要があります。
	［対処方法］	*xxx* には，*zzz* の条件に沿った値を定義してください。
E4003	［メッセージ］	*xxx* cannot be defined by *yyy*.
	［説明］	*yyy* の条件では，*xxx* を定義することができません。
	［対処方法］	*xxx* を定義するには，条件 *yyy* を見直す必要があります。
E4006	［メッセージ］	No one use this *object_type* (*object_name*).
	［説明］	オブジェクト種別 *object_type* のオブジェクト *object_name* はどこにも使われていません。
	［対処方法］	*object_name* に対応したオブジェクト情報を削除してください。
E4007	［メッセージ］	Not enabled to access (*xxx*) (*yyy*).
	［説明］	*xxx* は *yyy* に対するアクセス権がありません。
	［対処方法］	*xxx* に *yyy* に対するアクセス権を与えてください。

E4008	［メッセージ］	Invalid event name.
	［説明］	タスクへの割り付けが行われていないイベントを指定しています。
	［対処方法］	イベント識別子を見直してください。
E4009	［メッセージ］	Category 1 ISR's priority must be set to a higher priority than Category 2 ISR's priority.
	［説明］	カテゴリ 1 の割り込みサービス・ルーチンに，カテゴリ 2 の割り込みサービス・ルーチンよりも低い優先度が定義されています。
	［対処方法］	カテゴリ 1 の割り込みサービス・ルーチンには，カテゴリ 2 の割り込みサービス・ルーチンよりも高い優先度を定義してください。
E4010	［メッセージ］	There is no space in event mask (*xxx*).
	［説明］	未使用のイベント・ビットが存在しないため，AUTO を指定することができません。
	［対処方法］	イベント・マスクを 0x1 ～ 0xFFFFFFFF の数値で指定してください。
E4011	［メッセージ］	TASK or ISR is expected in OS application (*xxx*).
	［説明］	OS アプリケーション情報にタスク，および割り込みサービス・ルーチンが 1 個も所属していません。
	［対処方法］	OS アプリケーション情報では，最低でも 1 個のタスク，または割り込みサービス・ルーチンを所属させてください。
E4013	［メッセージ］	*object_type* (*object_name*) doesn't belong to application.
	［説明］	オブジェクト種別 *object_type* のオブジェクト *object_name* がいずれの OS アプリケーションにも所属していません。
	［対処方法］	オブジェクト *object_name* をいずれかの OS アプリケーションに所属させてください。
E4014	［メッセージ］	Cyclic chain of alarm actions with Increment Counter (*xxx*).
	［説明］	操作対象となるカウンタ *xxx* がループしています。
	［対処方法］	関連付けられているカウンタ，および満了処理で操作を行うカウンタの正当性を確認してください。
E4015	［メッセージ］	*object_name* is defined in multiple OS application.
	［説明］	複数の OS アプリケーション情報で同一のオブジェクト *object_name* が定義されています。
	［対処方法］	識別子を見直してください。
E4017	［メッセージ］	The offset value of action is the same value.
	［説明］	同一の満了カウント値が他箇所で定義されています。
	［対処方法］	どちらか一方の満了カウント値を変更してください。
E4019	［メッセージ］	The same exception code as other ISR is specified.
	［説明］	同一の例外コードが他箇所で定義されています。
	［対処方法］	どちらか一方の例外コードを変更してください。
E4020	［メッセージ］	When *xxx* option is used, *yyy* should be defined.
	［説明］	起動オプションに *xxx* を指定した場合，*yyy* の記述を省略することはできません。
	［対処方法］	起動オプション *xxx* の指定を取りやめるか，*yyy* の記述を行ってください。

A.4.2 アボート・エラー

以下に，アボート・エラーを検出した際に出力されるメッセージを示します。
なお，メッセージ内のイタリック書体については，アボート・エラーを検出した際に決定します。

表 A.2 アボート・エラー

F1000	［メッセージ］	Can not open file (*file_name*).
	［説明］	ファイル *file_name* をオープンすることができません。
	［対処方法］	該当ファイルが存在しているか否かを確認してください。
F1002	［メッセージ］	Unknown device file format.
	［説明］	デバイス・ファイルのフォーマットが不正です。
	［対処方法］	デバイス・ファイルを再インストールしてください。
F1003	［メッセージ］	Too long file name.
	［説明］	ファイル名が制限文字数 “255 文字 ” を越えています。
	［対処方法］	ファイル名を 255 文字以下に変更してください。
F1004	［メッセージ］	Too long folder name.
	［説明］	フォルダ名が制限文字数 “255 文字 ” を越えています。
	［対処方法］	フォルダ名を 255 文字以下に変更してください。
F1005	［メッセージ］	Output file names are the same (*file_name*).
	［説明］	出力ファイル名 *file_name* が競合しています。
	［対処方法］	起動オプション -o で指定された SIT ファイル名と起動オプション -e で指定された ENTRY ファイル名の競合を回避してください。
F1006	［メッセージ］	Unauthorized use of option (*option*).
	［説明］	起動オプション *option* の指定形式が不正です。
	［対処方法］	起動オプションの指定形式を確認してください。
F1007	［メッセージ］	Illegal option (*option*).
	［説明］	起動オプション *option* の指定が不正です。
	［対処方法］	起動オプションの指定形式を確認してください。
F1008	［メッセージ］	Illegal format in command file.
	［説明］	コマンド・ファイルの記述形式が不正です。
	［対処方法］	コマンド・ファイルの記述形式を確認してください。
F1009	［メッセージ］	Out of memory.
	［説明］	メモリが足りません。
	［対処方法］	常駐プログラムを終了するなどして，メモリ領域の解放を行ってください。
F1010	［メッセージ］	*option* option is expected.
	［説明］	*option* は省略不可の起動オプションです。
	［対処方法］	起動オプション *option* を指定してください。

F1011	［メッセージ］	Too many include path.
	［説明］	インクルード・パスの定義数が 255 個を越えています。
	［対処方法］	複数のインクルード・ファイルをまとめるなどして，インクルード・パスの定義数を 255 個以下に抑えてください。
F1012	［メッセージ］	Too many File lines.
	［説明］	CF ファイル（OIL）の行数が 32767 行を越えています。
	［対処方法］	省略時解釈を利用するなどして，CF ファイル（OIL）の記述行数を 32767 行以下に抑えてください。
F1013	［メッセージ］	Configuration file is not specified.
	［説明］	CF ファイルが指定されていません。
	［対処方法］	CF ファイルを指定してください。
F6002	［メッセージ］	XML file syntax error.
	［説明］	XML 文法エラーが発生しました。
	［対処方法］	CF ファイル（XML）の記述を確認してください。
F6003	［メッセージ］	Unknown error.
	［説明］	未知のエラーが発生しました。
	［対処方法］	CF ファイル（XML）の記述を確認してください。
F6004	［メッセージ］	XML file syntax error by an AUTOSAR schema.
	［説明］	AUTOSAR スキーマによる文法エラーが発生しました。
	［対処方法］	CF ファイル（XML）の記述を確認してください。

A.4.3 ワーニング

以下に，ワーニングを検出した際に出力されるメッセージを示します。
なお，メッセージ内のイタリック書体については，ワーニングを検出した際に決定します。

表 A.3 ワーニング

W1000	［メッセージ］	Nested command file.
	［説明］	コマンド・ファイルがネストしています。 コマンド・ファイルのネストを無視して処理を行います。
	［対処方法］	コマンド・ファイルからコマンド・ファイルを呼び出すことは禁止されています。呼び出し先のコマンド・ファイルの内容を呼び出し元コマンド・ファイル内に記述してください。
W4001	［メッセージ］	Priority is lower than *process*'s priority. (Rise to value of process's priory)
	［説明］	シーリング値の指定が不正です。 処理プログラム *process* の優先度と同値が定義されていたものとして処理を行います。
	［対処方法］	シーリング値を，リソースを使用している処理プログラムの優先度よりも高い優先度に変更してください。

備考 AUTOSAR 仕様では，W4001 に相当したワーニングの規定が行われていません。
本ワーニングは，RV850 が独自に追加したワーニング項目です。

B. CF ファイル（OIL）

本付録では，CF ファイル（OIL）を OSEK Implementation Language Version 2.5 の仕様に則って記述する際の方法について解説しています。

B.1 概　　要

CF ファイル（OIL）は，RV850 に提供するコンフィギュレーション情報を保持したファイル（SIT ファイル，ENTRY ファイル，カーネル・マクロ・ファイル）を生成する際に必要となるファイルであり，ユーザがテキスト・エディタを利用して記述するものです。
以下に，CF ファイル（OIL）を記述する際の表記方法を示します。

(1) 文字コード
CF ファイル（OIL）は，ASCII コードで記述します。
なお，コメントについては，SJIS コード，または EUC コードでの記述が可能です。

(2) コメント
“/*” から “*/” で囲まれた部位，および “//” から行末までがコメントとして扱われます。

(3) 数値
数字 “0 ～ 9” で始まる単語が数値として扱われます。
なお，小数点を含む数値は，10 進数での表記に限られます。

0 ～ 9 の数字で始まる単語：　10 進数
0x，または 0X で始まる単語：　16 進数

備考　CF ファイル（OIL）に 2 進数，および 8 進数を記述することはできません。

(4) 名前
英字 “a ～ z，A ～ Z”，またはアンダーバー “_” で始まる単語が名前として扱われます。
なお，名前として指定可能な文字数は，255 文字以内に限られます。

備考 1.　名前とシンボル名の区別は，CF ファイル（OIL）の文脈から判断されます。

備考 2.　CF ファイル（OIL）では，英小文字 “a ～ z” と英大文字 “A ～ Z” の区別が行われます。

(5) シンボル名
英字 “a ～ z，A ～ Z”，またはアンダーバー “_” で始まる単語がシンボル名として扱われます。
なお，シンボル名として指定可能な文字数は，4095 文字以内に限られます。

備考 1.　名前とシンボル名の区別は，CF ファイル（OIL）の文脈から判断されます。

備考 2.　CF ファイル（OIL）では，英小文字 “a ～ z” と英大文字 “A ～ Z” の区別が行われます。

備考 3.　RV850 では，_kernel，および _KERNEL で始まるシンボル名を OS 予約シンボルとしているため，指定された用途以外での使用は禁止されています。

(6) キーワード
以下に示した単語は，CF ファイル（OIL）用のキーワードとして予約されているため，指定された用途以外で使用することは禁止されています。

ABSOLUTE，ACCESSING_APPLICATION，ACTION，ACTIVATETASK，ACTIVATION，
ADJUSTABLEEXPPOINT，ALARM，ALARMCALLBACK，ALARMCALLBACKNAME，ALARMTIME，
APPLICATION，APPMODE，ATTRIBUTE，AUTO，AUTOSTART，CATEGORY，CODE，CONST，
CONSTNAME，CORE，CORENUMBER，COUNTER，CPU，CPUCORE，CYCLETIME，DATA，
DEFAULTFPSRVALUE，DRIVER，ECUCPARTITION，ENDADDRESS，ERRORHOOK，EVENT，
EXCEPTIONCODE，EXECUTIONBUDGET，EXPLICIT，EXTENDED，FALSE，FULL，G3K，G3M，
G3KH，G3MH，HARDWARE，IMPLICIT，INCREMENTCOUNTER，INTC1EICTRL，INTC1EIMASK，
INTC2EICTRL，INTC2EIMASK，INTERNAL，INTERRUPTBASE，INTPRI0 ～ INTPRI15，ISR，IOC，
LENGTH，LINKED，LINKEDRESOURCE，LOCAL_TO_GLOBAL_TIME_SYNCHRONIZATION，
LOCKINGTIME，MASK，MAXADVANCE，MAXALLINTERRUPTLOCKTIME，MAXALLOWEDVALUE，
MAXEXCEPTIONCODE，MAXOSINTERRUPTLOCKTIME，MAXRESOURCELOCKTIME，MAXRETARD，
MEMORYAREA，MINCYCLE，NAME，NON，NONE，OFFSET，OS，OSDEFAULTAPPMODE，
OSTMCNT，PRECISION，PERIODIC，POSTTASKHOOK，PRETASKHOOK，PRIORITY，PROTECT，
PROTECTIONHOOK，READONLY，READWRITE，RELATIVE，RESOURCE，RESOURCEPROPERTY，
RESOURCELOCK，RES_SCHEDULER，RESTARTTASK，SAVEFPUREG，SC1 ～ SC4，
SCALABILITYCLASS，SCHEDULE，SCHEDULETABLE，SECONDSPERTICK，SETEVENT，
SHUTDOWNHOOK，SIZE，SOFTWARE，

, SPINLOCK, STANDARD, STACKMONITORING, STACKSIZE, STARTADDRESS, STARTUPHOOK, STARTVALUE, STATUS, SYNCHRON, SYNCSTRATEGY, SYSTEM, SYSTEM_CLOCK, SYSTEM_SPID, TASK, TICKSPERBASE, TIMECONSTANTS, TIMEFRAME, TIMEVALUE, TIMING_PROTECTION, TRACESYSTEMENTRY, TRACESYSTEMEXIT, TRACETASKSTATUS, TRUE, TRUSTED, TRUSTED_FUNCTION, TRUSTEDDELAYTIMINGVIOLATIONCALL, TRUSTEDWITHPROTECTION, TYPE, USEGETSERVICEID, USEPARAMETERACCESS, USERESSCHEDULER

B.2 コンフィギュレーション情報

CF ファイル（OIL）に記述するデータ（コンフィギュレーション情報）は，以下に示した項目に大別されます。

(1) インクルード・ファイル

(2) CPU セクション

(a) アラーム情報

(b) アプリケーション・モード情報

(c) OS アプリケーション情報

(d) カウンタ情報

(e) イベント情報

(f) 割り込みサービス・ルーチン情報

(g) OS 情報

(h) リソース情報

(i) スケジュール・テーブル情報

(j) タスク情報

(k) システム情報

以下に，CF ファイル（OIL）の記述イメージを示しますが，“ [ ] ”で囲まれた部分は省略可能な項目であることを示しています。

図 B.1 CF ファイル（OIL）の記述イメージ

```
// インクルード・ファイル
[ #include inc_file ]

// CPU セクション
CPU Os {
    [ アラーム情報 ]
    [ アプリケーション・モード情報 ]
    [ OS アプリケーション情報 ]
    [ カウンタ情報 ]
    [ イベント情報 ]
    [ 割り込みサービス・ルーチン情報 ]
    OS 情報
    [ リソース情報 ]
    [ スケジュール・テーブル情報 ]
    [ タスク情報 ]
    システム情報
};
```

B.3 インクルード・ファイル

他 CF ファイル（OIL）を利用するうえで必要となる情報として，以下の項目を定義します。

- インクルード・ファイル “inc_file”

以下に，インクルード・ファイルの記述形式を示しますが，“ [ ] ” で囲まれた部分は省略可能な項目であることを示しています。

図 B.2 インクルード・ファイルの記述形式

```
#include inc_file
```

(1) インクルード・ファイル “*inc_file*”
インクルード・ファイルのファイル名を指定します。
なお，*inc_file* として指定可能な値は，ファイル名（パスを含め 255 文字以内）に限られます。

備考 1. インクルード・ファイルの検索順序は，*inc_file* の記述形式により，以下のように異なります。

【“#include <*inc_file*>” 形式の場合】

- 起動オプション “-I △ *path_name*” の *path_name* で指定されたフォルダ
- 起動オプション “*cf_file*” で指定された CF ファイル（OIL）の格納フォルダ
- コンフィギュレータ Os_Configurator.exe を起動したフォルダ

【“#include "*inc_file*"” 形式の場合】

- 起動オプション “*cf_file*” で指定された CF ファイル（OIL）の格納フォルダ
- コンフィギュレータ Os_Configurator.exe を起動したフォルダ
- 起動オプション “-I △ *path_name*” の *path_name* で指定されたフォルダ

備考 2. インクルード・ファイルの最大ネスト数は，8 に限られます。

備考 3. 本項目は，以下のように複数の定義が可能です。

```
#include "inc_file1.oil"
#include "inc_file2.oil"
#include "inc_file3.oil"
```

B.4 CPU セクション

RV850 が提供している機能を実現するうえで必要となる情報として，以下の項目を定義します。

- 識別子 “Os”
- アラーム情報～システム情報

なお，CPU セクションとして定義可能な数は，1 個に限られます。
以下に，CPU セクションの記述形式を示しますが，“ [ ] ” で囲まれた部分は省略可能な項目であることを示しています。

図 B.3 CPU セクションの記述形式

```
CPU Os {
    [ アラーム情報 ]
    [ アプリケーション・モード情報 ]
    [ OS アプリケーション情報 ]
    [ カウンタ情報 ]
    [ イベント情報 ]
    [ 割り込みサービス・ルーチン情報 ]
    OS 情報
    [ リソース情報 ]
    [ スケジュール・テーブル情報 ]
    [ タスク情報 ]
    システム情報
};
```

(1) 識別子 “*Os*”
CPU セクションの識別子を指定します。
なお，*Os* として指定可能な値は，名前に限られます。

(2) アラーム情報～システム情報
“ アラーム情報～システム情報 ” についての詳細は，「B.4.1 アラーム情報」から「B.4.11 システム情報」を参照してください。

B.4.1 アラーム情報

RV850 が提供しているアラーム管理を実現するうえで必要となる情報として，以下の項目を定義します。

- 識別子 “OsAlarm”
- OS アプリケーション識別子 “OsAlarmAccessingApplication”
- カウンタ識別子 “OsAlarmCounterRef”
- 満了処理 “OsAlarmAction”
    - タスク識別子 “OsAlarmActivateTaskRef”
    - イベント識別子 “OsAlarmSetEventRef”
    - タスク識別子 “OsAlarmSetEventTaskRef”
    - カウンタ識別子 “OsAlarmIncrementCounterRef”
    - アラーム・コールバック識別子 “OsAlarmCallbackName”
- 初期状態 “OsAlarmAutostart”
    - 満了カウント値 “OsAlarmAlarmTime”
    - カウント方法 “OsAlarmAutostartType”
    - 周期カウント値 “OsAlarmCycleTime”
    - アプリケーション・モード “OsAlarmAppModeRef”

なお，アラーム情報として定義可能な数は，スケジュール・テーブル情報との合計で 0 ～ 1023 個に限られます。

備考　RV850 では，スケーラビリティ・クラス “OsScalabilityClass” で SC3 を定義，本情報で定義したカウンタを OS アプリケーション情報で未定義の場合，フェイタル・エラー E4013 としています。

以下に，アラーム情報の記述形式を示しますが，“ [　] ” で囲まれた部分は省略可能な項目であることを示しています。

図 B.4　アラーム情報の記述形式

```
ALARM OsAlarm {
    [ ACCESSING_APPLICATION = OsAlarmAccessingApplication; ]
    COUNTER = OsAlarmCounterRef;
    ACTION = OsAlarmAction {
        [ TASK = OsAlarmActivateTaskRef; ]
        [ EVENT = OsAlarmSetEventRef; ]
        [ TASK = OsAlarmSetEventTaskRef; ]
        [ COUNTER = OsAlarmIncrementCounterRef; ]
        [ ALARMCALLBACKNAME = OsAlarmCallbackName; ]
    };
    [ AUTOSTART = OsAlarmAutostart {
        [ ALARMTIME = OsAlarmAlarmTime; ]
        [ TYPE = OsAlarmAutostartType; ]
        [ CYCLETIME = OsAlarmCycleTime; ]
        [ APPMODE = OsAlarmAppModeRef; ]
    }; ]
};
```

(1)　識別子 “*OsAlarm*”
アラームの識別子を指定します。
なお，*OsAlarm* として指定可能な値は，名前に限られます。

(2)　OS アプリケーション識別子 “*OsAlarmAccessingApplication*”
アラームへのアクセス権を与えるオブジェクト（タスク，割り込みサービス・ルーチン，カウンタ）が定義されている OS アプリケーションの識別子を指定します。
なお，*OsAlarmAccessingApplication* として指定可能な値は，識別子 “OsApplication” に限られます。

備考 1.　本項目は，スケーラビリティ・クラス “OsScalabilityClass” で SC3 を定義した場合に限り指定可能となります。

AUTOSAR 仕様では，スケーラビリティ・クラス “OsScalabilityClass” で SC3，SC4 以外の値が定義された場合の動作をワーニングとする旨の規定が行われていますが，RV850 では，フェイタル・エラー E4003 としています。

備考 2. 本アラームが所属している OS アプリケーションの識別子を *OsAlarmAccessingApplication* として指定する必要はありません。

備考 3. 本項目は，以下のように複数の定義（最大：31 個）が可能です。

```
ACCESSING_APPLICATION = OsApplication1;
ACCESSING_APPLICATION = OsApplication2;
ACCESSING_APPLICATION = OsApplication3;
```

【省略時解釈】
アラームへのアクセス権を与えるオブジェクト（タスク，割り込みサービス・ルーチン，カウンタ）は，本アラームが所属している OS アプリケーションに定義されているオブジェクトに限定されているものとして処理が行われます。

(3) カウンタ識別子 “*OsAlarmCounterRef*”
関連付けを行うカウンタ（満了条件が成立したか否かの確認対象となるカウント値を保有しているカウンタ）の識別子を指定します。
なお，*OsAlarmCounterRef* として指定可能な値は，識別子 “OsCounter” に限られます。

備考 スケーラビリティ・クラス “OsScalabilityClass” で SC3 を定義し，本項目に “ 本アラームが所属している OS アプリケーションに未所属のカウンタ（他 OS アプリケーションに所属しているカウンタ）” を指定する場合は，OS アプリケーション識別子 “OsAlarmAccessingApplication” に該当カウンタが所属している OS アプリケーションの識別子を定義する必要があります。

(4) 満了処理 “*OsAlarmAction*”
満了条件が成立した際に実行する処理の種別を指定します。
なお，*OsAlarmAction* として指定可能な値は，ACTIVATETASK，SETEVENT，INCREMENTCOUNTER，または ALARMCALLBACK に限られます。

ACTIVATETASK：	タスクの起動（ActivateTask と同等の処理）
SETEVENT：	イベント・マスクの設定（SetEvent と同等の処理）
INCREMENTCOUNTER：	カウント値の更新（IncrementCounter と同等の処理）
ALARMCALLBACK：	アラーム・コールバックの呼び出し（SC1 限定）

備考 ALARMCALLBACK は，スケーラビリティ・クラス “OsScalabilityClass” で SC1 を定義した場合に限り指定可能となります。

(a) タスク識別子 “*OsAlarmActivateTaskRef*”
満了条件が成立した際に起動するタスクの識別子を指定します。
なお，*OsAlarmActivateTaskRef* として指定可能な値は，識別子 “OsTask” に限られます。

備考 1. 本項目は，満了処理 “OsAlarmAction” で ACTIVATETASK を定義した場合に限り指定可能となります。

備考 2. スケーラビリティ・クラス “OsScalabilityClass” で SC3 を定義し，本項目に “ 本アラームが所属している OS アプリケーションに未所属のタスク（他 OS アプリケーションに所属しているタスク）” を指定する場合は，OS アプリケーション識別子 “OsTaskAccessingApplication” に本アラームが所属している OS アプリケーションの識別子を定義する必要があります。

(b) イベント識別子 “*OsAlarmSetEventRef*”
満了条件が成立した際に設定するイベント・マスクを保持しているイベントの識別子を設定します。
なお，*OsAlarmSetEventRef* として指定可能な値は，識別子 “OsEvent” に限られます。

備考 1. 本項目は，満了処理 “OsAlarmAction” で SETEVENT を定義した場合に限り指定可能となります。

備考 2. 本項目で指定するイベントは，タスク識別子 “OsAlarmSetEventTaskRef” で定義したタスクに割り付けられている必要があります。

(c) タスク識別子 “*OsAlarmSetEventTaskRef*”
満了条件が成立した際にイベント・マスクを設定するタスクの識別子を指定します。
なお，*OsAlarmSetEventTaskRef* として指定可能な値は，識別子 “OsTask” に限られます。

備考 1. 本項目は，満了処理 “OsAlarmAction” で SETEVENT を定義した場合に限り指定可能となります。

備考 2. スケーラビリティ・クラス “OsScalabilityClass” で SC3 を定義し，本項目に “ 本アラームが所属している OS アプリケーションに未所属のタスク（他 OS アプリケーションに所属しているタスク）” を指定する場合は，OS アプリケーション識別子 “OsTaskAccessingApplication” に本アラームが所属している OS アプリケーションの識別子を定義する必要があります。

(d) カウンタ識別子 “*OsAlarmIncrementCounterRef*”
満了条件が成立した際にカウント値を更新するカウンタの識別子を指定します。
なお，*OsAlarmIncrementCounterRef* として指定可能な値は，識別子 “OsCounter” に限られます。

備考 1. 本項目は，満了処理 “OsAlarmAction” で INCREMENTCOUNTER を定義した場合に限り指定可能となります。

備考 2. 本項目での指定対象カウンタは，種別 “OsCounterType” で SOFTWARE を定義したカウンタに限られます。

備考 3. スケーラビリティ・クラス “OsScalabilityClass” で SC3 を定義し，本項目に “ 本アラームが所属している OS アプリケーションに未所属のカウンタ（他 OS アプリケーションに所属しているカウンタ）” を指定する場合は，OS アプリケーション識別子 “OsCounterAccessingApplication” に本アラームが所属している OS アプリケーションの識別子を定義する必要があります。

備考 4. AUTOSAR 仕様では，本項目とカウンタ識別子 “OsAlarmCounterRef” の値が同じ場合の動作（ループ動作）をワーニングとする旨の規定が行われていますが，RV850 では，フェイタル・エラー E4014 としています。

(e) アラーム・コールバック識別子 “*OsAlarmCallbackName*”
満了条件が成立した際に呼び出すアラーム・コールバックの識別子を指定します。
なお，*OsAlarmCallbackName* として指定可能な値は，シンボル名（4095 文字以内）に限られます。

備考 1. 本項目は，満了処理 “OsAlarmAction” で ALARMCALLBACK を定義した場合に限り指定可能となります。

備考 2. アラーム・コールバックを以下のように記述した際には，*OsAlarmCallbackName* に設定する値は，“OsAlarmCallbackName1” となります。
なお，アラーム・コールバックについての詳細は，「8.2 アラーム・コールバック」を参照してください。

```
ALARMCALLBACK ( OsAlarmCallbackName1 ) {
    ……
    ……
    return;
}
```

(5) 初期状態 “*OsAlarmAutostart*”
アラームの初期状態を指定します。
なお，*OsAlarmAutostart* として指定可能な値は，TRUE，または FALSE に限られます。

TRUE： StartOS の引数 *Mode* に依存
FALSE： 非アクティブ状態

備考 1. 本項目で “TRUE” を指定した場合，アラームの初期状態は，StartOS の引数 *Mode* に設定する値により以下のように変化します。

【引数 *Mode* がアプリケーション・モード “OsAlarmAppModeRef” の値と一致】

- アクティブ状態

【引数 *Mode* がアプリケーション・モード “OsAlarmAppModeRef” の値と不一致】

- 非アクティブ状態

備考 2. FALSE を指定する際の記述形式は，以下のようになります。

```
AUTOSTART = FALSE;
```

備考 3. ARXML 形式の CF ファイルでは，本項目に 1 を指定することにより TRUE と同様の扱いが，本項目に 0 を指定することにより FALSE と同様の扱いが行われます。

【省略時解釈】
FALSE を指定したものとして処理が行われます。

(a) 満了カウント値 “*OsAlarmAlarmTime*”
アラームの満了カウント値（相対カウント値，または絶対カウント値）を指定します。
なお，*OsAlarmAlarmTime* として指定可能な値は，カウンタ識別子 “OsAlarmCounterRef” で関連付けたカウンタの種別 “OsCounterType” の定義内容に依存します。

- ABSOLUTE の場合
  0x0 ～最大カウント値 “OsCounterMaxAllowedValue” に限られます。
- RELATIVE の場合
  0x1 ～最大カウント値 “OsCounterMaxAllowedValue” に限られます。

備考 本項目に指定する値は，カウント方法 “OsAlarmAutostartType” で RELATIVE を定義する場合は相対カウント値，ABSOLUTE を定義する場合は絶対カウント値となります。

(b) カウント方法 “*OsAlarmAutostartType*”
満了カウント値 “OsAlarmAlarmTime” までのカウント方法を指定します。
なお，*OsAlarmAutostartType* として指定可能な値は，RELATIVE，または ABSOLUTE に限られます。

RELATIVE： 相対カウント
ABSOLUTE： 絶対カウント

(c) 周期カウント値 “*OsAlarmCycleTime*”
アラームの周期カウント値を指定します。
なお，*OsAlarmCycleTime* として指定可能な値は，0x0，または最小周期 “OsCounterMinCycle” ～最大カウント値 “OsCounterMaxAllowedValue” に限られます。

備考 アラームは，*OsAlarmCycleTime* に 0x0 が指定された場合は “ ワンショット・アラーム ” として，0x0 以外の値が指定された場合は “ 周期アラーム ” として動作します。

(d) アプリケーション・モード “*OsAlarmAppModeRef*”
アラームのアプリケーション・モードを指定します。
なお，*OsAlarmAppModeRef* として指定可能な値は，アプリケーション・モード “OsAppMode” に限られます。

備考 1. デフォルト・アプリケーション・モード OSDEFAULTAPPMODE については，定義の有無に関わらず，定義されていたものとして処理が行われます。

備考 2. 本項目は，以下のように複数の定義（最大：127 個）が可能です。

```
APPMODE = OSDEFAULTAPPMODE;
APPMODE = OsAppMode1;
APPMODE = OsAppMode2;
```

【省略時解釈】
AUTOSAR 仕様では，本項目は省略不可とする旨の規定が行われていますが，RV850 では，本項目が省略された際には，OSDEFAULTAPPMODE を指定したものとして処理が行われます。

B.4.2 アプリケーション・モード情報

RV850 が提供している OS 実行管理を実現するうえで必要となる情報として，以下の項目を定義します。

- アプリケーション・モード “OsAppMode”

なお，アプリケーション・モード情報として定義可能な数は，0 ～ 127 個に限られます。
以下に，アプリケーション・モード情報の記述形式を示しますが，“ [ ] ” で囲まれた部分は省略可能な項目であることを示しています。

図 B.5 アプリケーション・モード情報の記述形式

```
APPMODE OsAppMode { };
```

(1) アプリケーション・モード “*OsAppMode*”
アプリケーション・モードを指定します。
なお，*OsAppMode* として指定可能な値は，名前，または OSDEFAULTAPPMODE に限られます。

名前： 通常のアプリケーション・モード
OSDEFAULTAPPMODE： デフォルト・アプリケーション・モード

備考 デフォルト・アプリケーション・モード OSDEFAULTAPPMODE については，定義の有無に関わらず，定義されていたものとして処理が行われます。

【省略時解釈】
AUTOSAR 仕様では，本項目は省略不可とする旨の規定が行われていますが，RV850 では，本項目が省略された際には，以下の記述を行ったものとして処理が行われます。

```
APPMODE OSDEFAULTAPPMODE { };
```

B.4.3 OS アプリケーション情報

RV850 が提供している OS アプリケーション管理を実現するうえで必要となる情報として，以下の項目を定義します。

- 識別子 “OsApplication”
- 信頼性 “OsTrusted”
- OS アプリケーション・スタック・サイズ “OsAppStackSize”
- FPSR“OsAppDefaultFPSRValue”
- コア識別子 “OsApplicationCoreRef”
- SPID“OsApplicationSPID”
- アラーム識別子 “OsAppAlarmRef””
- カウンタ識別子 “OsAppCounterRef”
- 割り込みサービス・ルーチン識別子 “OsAppIsrRef”
- スケジュール・テーブル識別子 “OsAppScheduleTableRef”
- タスク識別子 “OsAppTaskRef”
- タスク識別子 “OsRestartTask”
- StartupHook_OsApplication“OsAppStartupHook”
- ShutdownHook_OsApplication“OsAppShutdownHook”
- ErrorHook_OsApplication“OsAppErrorHook”
- Trusted 関数 “OsApplicationTrustedFunction”
    - 識別子 “OsTrustedFunctionName”
- メモリ領域識別子 “OsAppMemoryAreaNameRef”
    - 属性 “OsAppMemoryAreaAttribute”
- 時間保護違反の遅延発生 “OsTrustedApplicationDelayTimingViolationCall”
- Trusted な OS アプリケーションの保護 “OsTrustedApplicationWithProtection”

なお，OS アプリケーション情報として定義可能な数は，0 ～ 31 個に限られます。

備考　AUTOSAR 仕様では，OS アプリケーション情報として，項目 ECUCPARTITION が規定されていますが，RV850 では，該当項目をサポート外としています。

以下に，OS アプリケーション情報の記述形式を示しますが，“ [　] ” で囲まれた部分は省略可能な項目であることを示しています。

図 B.6　OS アプリケーション情報の記述形式

```
APPLICATION OsApplication {
    TRUSTED = OsTrusted;
    STACKSIZE = OsAppStackSize;
    [ DEFAULTFPSRVALUE = OsAppDefaultFPSRValue; ]
    [ CORE = OsApplicationCoreRef; ]
    [ SPID = OsApplicationSPID; ]
    [ ALARM = OsAppAlarmRef; ]
    [ COUNTER = OsAppCounterRef; ]
    [ ISR = OsAppIsrRef; ]
    [ SCHEDULETABLE = OsAppScheduleTableRef; ]
    [ TASK = OsAppTaskRef; ]
    [ RESTARTTASK = OsRestartTask; ]
    STARTUPHOOK = OsAppStartupHook;
    SHUTDOWNHOOK = OsAppShutdownHook;
    ERRORHOOK = OsAppErrorHook;
    [ TRUSTED_FUNCTION = OsApplicationTrustedFunction {
        NAME = OsTrustedFunctionName;
```

```
    }; ]
    [ MEMORYAREA = OsAppMemoryAreaNameRef {
        ATTRIBUTE = OsAppMemoryAreaAttribute;
    }; ]
    [ TRUSTEDDELAYTIMINGVIOLATIONCALL = OsTrustedApplicationDelayTimingViolationCall; ]
    [ TRUSTEDWITHPROTECTION = OsTrustedApplicationWithProtection; ]
};
```

(1) 識別子 “*OsApplication*”
OS アプリケーションの識別子を指定します。
なお，*OsApplication* として指定可能な値は，名前に限られます。

備考 本項目は，スケーラビリティ・クラス “OsScalabilityClass” で SC3 を定義した場合に限り指定可能となります。
AUTOSAR 仕様では，スケーラビリティ・クラス “OsScalabilityClass” で SC3，SC4 以外の値が定義された場合の動作をワーニングとする旨の規定が行われていますが，RV850 では，フェイタル・エラー E4003 としています。

(2) 信頼性 “*OsTrusted*”
OS アプリケーションの信頼性を指定します。
なお，*OsTrusted* として指定可能な値は，TRUE，または FALSE に限られます。

TRUE： Trusted な OS アプリケーション
FALSE： Non-Trusted な OS アプリケーション

備考 ARXML 形式の CF ファイルでは，本項目に 1 を指定することにより TRUE と同様の扱いが，本項目に 0 を指定することにより FALSE と同様の扱いが行われます。

(3) OS アプリケーション・スタック・サイズ “*OsAppStackSize*”
本 OS アプリケーションが使用するスタックのサイズ（単位：バイト）を指定します。
なお，*OsAppStackSize* として指定可能な値は，0x4 ～ 0x0FFFFFFC の 0x4 バイト・アライン値に限られます。

備考 1. 本項目で指定するサイズについての詳細は，「C.7.2 OS アプリケーション・スタック」を参照してください。

備考 2. AUTOSAR 仕様では，本項目の規定が行われていません。
本項目は，RV850 が独自に追加した項目です。

(4) FPSR“*OsAppDefaultFPSRValue*”
本 OS アプリケーションに所属している処理プログラム（タスク，割り込みサービス・ルーチン，フック・ルーチン）を起動する際，浮動小数点設定／状態レジスタ（FPSR）に設定する値を指定します。
なお，*OsAppDefaultFPSRValue* として指定可能な値は，0x0 ～ 0xFFFFFFFF に限られます。

備考 1. 本項目は，FPSR の退避／復帰 “OsSaveFpuReg” で TRUE を定義した場合に限り指定可能となります。

備考 2. AUTOSAR 仕様では，本項目の規定が行われていません。
本項目は，RV850 が独自に追加した項目です。

【省略時解釈】
FPSR のデフォルト値 “OsDefaultFPSRValue” と同値を指定したものとして処理が行われます。

(5) コア識別子 “*OsApplicationCoreRef*”
RV850 が制御対象とするコア名を指定します。
なお，*OsApplicationCoreRef* として指定可能な値は，名前に限られます。

備考 RV850 では，本項目にいかなる値が指定されたとしても，RV850 の動作に影響しません。

(6) SPID“*OsApplicationSPID*”
I/O 領域に対するアクセス権（システム・プロテクション番号）を指定します。
なお，*OsApplicationSPID* として指定可能な値は，0x0 ～ 0x1F に限られます。

備考 1. 本項目は，コア識別子 “OsSystemCpuCore” で G3M，G3KH，G3MH のいずれかを指定し，かつ信頼性 “OsTrusted” で FALSE を定義した場合に限り指定可能となります。

備考 2. 信頼性 “OsTrusted” で TRUE を定義した場合，または本項目にシステム SPID 値 “OsSystemSystemSPID” で指定した値を指定した場合，I/O 領域に対する全アクセスが可能となります。

備考 3. *OsAppicationSPID* に指定した値と I/O 領域に対するアクセス権の対応は，ユーザ・オウン・コーディング部として切り出されたブート処理の内容に依存します。

備考 4. SPID についての詳細は，対象デバイスのユーザーズ・マニュアルを参照してください。

備考 5. AUTOSAR 仕様では，本項目の規定が行われていません。
本項目は，RV850 が独自に追加した項目です。

(7) アラーム識別子 "*OsAppAlarmRef*""
本 OS アプリケーションに所属させるアラームの識別子を指定します。
なお，*OsAppAlarmRef* として指定可能な値は，識別子 "OsAlarm" に限られます。

備考 本項目は，以下のように複数の定義（最大：1023 個）が可能です。

```
ALARM = OsAlarm1;
ALARM = OsAlarm2;
ALARM = OsAlarm3;
```

(8) カウンタ識別子 "*OsAppCounterRef*"
本 OS アプリケーションに所属させるカウンタの識別子を指定します。
なお，*OsAppCounterRef* として指定可能な値は，識別子 "OsCounter" に限られます。

備考 本項目は，以下のように複数の定義（最大：1023 個）が可能です。

```
COUNTER = OsCounter1;
COUNTER = OsCounter2;
COUNTER = OsCounter3;
```

(9) 割り込みサービス・ルーチン識別子 "*OsAppIsrRef*"
本 OS アプリケーションに所属させる割り込みサービス・ルーチンの識別子を指定します。
なお，*OsAppIsrRef* として指定可能な値は，識別子 "OsIsr" に限られます。

備考 1. 信頼性 "OsTrusted" が Trusted な OS アプリケーション（TRUE）の場合に限り，カテゴリ 1 の割り込みサービス・ルーチンの定義が可能です。

備考 2. 本項目は，以下のように複数の定義（最大：1023 個）が可能です。

```
ISR = OsIsr1;
ISR = OsIsr2;
ISR = OsIsr3;
```

備考 3. AUTOSAR 仕様では，タスク識別子 "OsAppTaskRef" と本項目の両項目を未定義可能とする旨の規定が行われていますが，RV850 では，両項目が未定義の際には，フェイタル・エラー E4011 としています。

(10) スケジュール・テーブル識別子 "*OsAppScheduleTableRef*"
本 OS アプリケーションに所属させるスケジュール・テーブルの識別子を指定します。
なお，*OsAppScheduleTableRef* として指定可能な値は，識別子 "OsScheduleTable" に限られます。

備考 本項目は，以下のように複数の定義（最大：1023 個）が可能です。

```
SCHEDULETABLE = OsScheduleTable1;
SCHEDULETABLE = OsScheduleTable2;
SCHEDULETABLE = OsScheduleTable3;
```

(11) タスク識別子 "*OsAppTaskRef*"
本 OS アプリケーションに所属させるタスクの識別子を指定します。
なお，*OsAppTaskRef* として指定可能な値は，識別子 "OsTask" に限られます。

備考 1. 本項目は，以下のように複数の定義（最大：1023 個）が可能です。

```
TASK = OsTask1;
TASK = OsTask2;
TASK = OsTask3;
```

備考 2. AUTOSAR 仕様では，本項目と割り込みサービス・ルーチン識別子 "OsAppIsrRef" の両項目を未定義可能とする旨の規定が行われていますが，RV850 では，両項目が未定義の際には，フェイタル・エラー E4011 としています。

(12) タスク識別子 “*OsRestartTask*”
TerminateApplication（引数 *RestartOption* に RESTART を設定）の発行時に起動要求を発行するタスクの識別子を指定します。
なお，*OsRestartTask* として指定可能な値は，タスク識別子 “OsAppTaskRef” に限られます。

【省略時解釈】
TerminateApplication を発行する際，引数 *RestartOption* に RESTART は設定されないものとして処理が行われます。

(13) StartupHook_*OsApplication*“*OsAppStartupHook*”
本 OS アプリケーションに固有フック・ルーチン（StartupHook_*OsApplication*）を所属させるか否かを指定します。
なお，*OsAppStartupHook* として指定可能な値は，TRUE，または FALSE に限られます。

TRUE： 固有フック・ルーチンを所属させる
FALSE： 固有フック・ルーチンを所属させない

備考 1. TRUE を指定した際には，以下の固有フック・ルーチンが本 OS アプリケーションに所属することになります。

```
void
StartupHook_OsApplication ( void ) {
    ……
    ……
}
```

備考 2. ARXML 形式の CF ファイルでは，本項目に 1 を指定することにより TRUE と同様の扱いが，本項目に 0 を指定することにより FALSE と同様の扱いが行われます。

(14) ShutdownHook_*OsApplication*“*OsAppShutdownHook*”
本 OS アプリケーションに固有フック・ルーチン（ShutdownHook_*OsApplication*）を所属させるか否かを指定します。
なお，*OsAppShutdownHook* として指定可能な値は，TRUE，または FALSE に限られます。

TRUE： 固有フック・ルーチンを所属させる
FALSE： 固有フック・ルーチンを所属させない

備考 1. TRUE を指定した際には，以下の固有フック・ルーチンが本 OS アプリケーションに所属することになります。

```
void
ShutdownHook_OsApplication ( StatusType Fatalerror ) {
    ……
    ……
}
```

備考 2. ARXML 形式の CF ファイルでは，本項目に 1 を指定することにより TRUE と同様の扱いが，本項目に 0 を指定することにより FALSE と同様の扱いが行われます。

(15) ErrorHook_*OsApplication*“*OsAppErrorHook*”
本 OS アプリケーションに固有フック・ルーチン（ErrorHook_*OsApplication*）を所属させるか否かを指定します。
なお，*OsAppErrorHook* として指定可能な値は，TRUE，または FALSE に限られます。

TRUE： 固有フック・ルーチンを所属させる
FALSE： 固有フック・ルーチンを所属させない

備考 1. TRUE を指定した際には，以下の固有フック・ルーチンが本 OS アプリケーションに所属することになります。

```
void
ErrorHook_OsApplication ( StatusType Error ) {
    ……
    ……
}
```

備考 2. ARXML 形式の CF ファイルでは，本項目に 1 を指定することにより TRUE と同様の扱いが，本項目に 0 を指定することにより FALSE と同様の扱いが行われます。

(16) Trusted 関数 “*OsApplicationTrustedFunction*”
本 OS アプリケーションに Trusted 関数を所属させるか否かを指定します。
なお，*OsApplicationTrustedFunction* として指定可能な値は，TRUE，または FALSE に限られます。

TRUE： Trusted 関数を所属させる
FALSE： Trusted 関数を所属させない

備考 1. 本項目は，信頼性 “OsTrusted” で TRUE を定義した場合に限り指定可能となります。

備考 2. 本項目は，以下のように複数の定義（最大：1023 個）が可能です。

```
TRUSTED_FUNCTION = TRUE {
    NAME = OsTrustedFunctionName1;
};
TRUSTED_FUNCTION = TRUE {
    NAME = OsTrustedFunctionName2;
};
TRUSTED_FUNCTION = TRUE {
    NAME = OsTrustedFunctionName3;
};
```

備考 3. FALSE を指定する際の記述形式は，以下のようになります。

```
TRUSTED_FUNCTION = FALSE;
```

備考 4. ARXML 形式の CF ファイルでは，本項目に 1 を指定することにより TRUE と同様の扱いが，本項目に 0 を指定することにより FALSE と同様の扱いが行われます。

【省略時解釈】
FALSE を指定したものとして処理が行われます。

(a) 識別子 “*OsTrustedFunctionName*”
Trusted 関数の識別子を指定します。
なお，*OsTrustedFunctionName* として指定可能な値は，名前に限られます。

備考 1. Trusted 関数を以下のように記述した際には，*OsTrustedFunctionName* に設定する値は “OsTrustedFunctionName1” となります。

```
TRUSTED ( OsTrustedFunctionName1 ) {
    ……
    ……
}
```

備考 2. AUTOSAR 仕様では，本項目に指定可能な値を関数名とする旨の規定が行われていますが，RV850 では，本項目に指定可能な値を識別子としています。

(17) メモリ領域識別子 “*OsAppMemoryAreaNameRef*”
本 OS アプリケーションでアクセス保護対象とするメモリ領域の識別子を指定します。
なお，*OsAppMemoryAreaNameRef* として指定可能な値は，メモリ領域識別子 “OsSystemMemoryArea” に限られます。

備考 1. 本項目は，コア識別子 “OsSystemCpuCore” で G3M，G3KH，G3MH のいずれかを指定し，かつ信頼性 “OsTrusted” で FALSE を定義した場合に限り指定可能となります。

備考 2. 本項目は，以下のように複数の定義（最大：4 個）が可能です。

```
MEMORYAREA = OsSystemMemoryArea11 {
    ATTRIBUTE = CODE;
};
MEMORYAREA = OsSystemMemoryArea12 {
    ATTRIBUTE = CONST;
};
MEMORYAREA = OsSystemMemoryArea13 {
    ATTRIBUTE = DATA;
};
```

備考 3. アクセス保護対象が全 OS アプリケーション共通となるメモリ領域については，メモリ領域識別子 “OsMemoryAreaNameRef” で定義を行います。

備考 4. AUTOSAR 仕様では，本項目の規定が行われていません。
本項目は，RV850 が独自に追加した項目です。

(a) 属性 “*OsAppMemoryAreaAttribute*”
メモリ領域に対して付与する属性を指定しています。
なお，*OsAppMemoryAreaAttribute* として指定可能な値は，CODE，CONST，または DATA に限られます。

CODE： 読み込み可，および実行可
CONST： 読み込み可
DATA： 読み込み可，および書き込み可

備考 1. 本 OS アプリケーションに所属している処理プログラム（タスク，割り込みサービス・ルーチン，フック・ルーチン）が本項目で指定した以外のメモリ・アクセスを行った際には，ProtectionHook“OsProtectionHook” の定義内容にしたがった処理（ProtectionHook の呼び出し，または ShutdownOS の発行）が行われます。

備考 2. AUTOSAR 仕様では，本項目の規定が行われていません。
本項目は，RV850 が独自に追加した項目です。

(18) 時間保護違反の遅延発生 “*OsTrustedApplicationDelayTimingViolationCall*”
Trusted な OS アプリケーションで時間保護違反が発生したときに，ProtectionHook の起動の遅延を許容するかどうかを指定します。
なお，*OsTrustedApplicationDelayTimingViolationCall* として指定可能な値は，TRUE，または FALSE に限られます。

備考 RV850 では，時間保護機能をサポートしていないため，本項目にいかなる値が指定されたとしても，RV850 の動作に影響しません。

【省略時解釈】
TRUE を指定したものとして処理が行われます。

(19) Trusted な OS アプリケーションの保護 “*OsTrustedApplicationWithProtection*”
Trusted な OS アプリケーションがメモリ保護等を必要とする OS アプリケーションであるかどうかを指定します。
なお，*OsTrustedApplicationWithProtection* として指定可能な値は，TRUE，または FALSE に限られます。

備考 RV850 では，Trusted な OS アプリケーションに対するメモリ保護機能をサポートしていないため，本項目にいかなる値が指定されたとしても，RV850 の動作に影響しません。

【省略時解釈】
FALSE を指定したものとして処理が行われます。

B.4.4 カウンタ情報

RV850 が提供しているカウンタ管理を実現するうえで必要となる情報として，以下の項目を定義します。

- 識別子 “OsCounter”
- 最大カウント値 “OsCounterMaxAllowedValue”
- 最小周期 “OsCounterMinCycle”
- 基本カウント値 “OsCounterTicksPerBase”
- 種別 “OsCounterType”
- 1 tick 当たりの秒数 “OsSecondsPerTick”
- OS アプリケーション識別子 “OsCounterAccessingApplication”
- ハードウエア・カウンタ情報
    - 例外コード “OsCounterExceptionCode”
    - 優先度 “OsCounterPriority”
- データ・マクロ情報
    - マクロ名 “OsConstName”
    - カウント値 “OsTimeValue”

なお，カウンタ情報として定義可能な数は，0 ～ 1023 個に限られます。

備考 1. RV850 では，本情報で定義したカウンタがアラーム情報，またはスケジュール・テーブル情報で利用されていない場合，フェイタル・エラー E4006 としています。

備考 2. RV850 では，スケーラビリティ・クラス “OsScalabilityClass” で SC3 を定義，本情報で定義したカウンタを OS アプリケーション情報で未定義の場合，フェイタル・エラー E4013 としています。

以下に，カウンタ情報の記述形式を示しますが，“ [ ] ” で囲まれた部分は省略可能な項目であることを示しています。

図 B.7 カウンタ情報の記述形式

```
COUNTER OsCounter {
    MAXALLOWEDVALUE = OsCounterMaxAllowedValue;
    MINCYCLE = OsCounterMinCycle;
    TICKSPERBASE = OsCounterTicksPerBase;
    TYPE = OsCounterType;
    [ SECONDSPERTICK = OsSecondsPerTick; ]
    [ ACCESSING_APPLICATION = OsCounterAccessingApplication; ]
    [ DRIVER {
        EXCEPTIONCODE = OsCounterExceptionCode;
        [ PRIORITY = OsCounterPriority; ]
    }; ]
    [ TIMECONSTANTS {
        CONSTNAME = OsConstName;
        TIMEVALUE = OsTimeValue;
    }; ]
};
```

(1) 識別子 “*OsCounter*”
カウンタの識別子を指定します。
なお，*OsCounter* として指定可能な値は，名前，または SYS_COUNTER に限られます。

名前： 通常のカウンタ
SYS_COUNTER： システム・カウンタ

備考 システム・カウンタについての詳細は，「7.1.1 システム・カウンタ」を参照してください。

(2) 最大カウント値 “*OsCounterMaxAllowedValue*”
カウント値としてカウント可能な最大カウント値を指定します。

なお，*OsCounterMaxAllowedValue* として指定可能な値は，最小周期 “OsCounterMinCycle” ～ 0x7FFFFFFF に限られます。

備考　AUTOSAR 仕様では，本項目に指定可能な最大値を 0xFFFFFFFFFFFFFFFF とする旨の規定が行われていますが，RV850 では，本項目に指定な最大値を 0x7FFFFFFF としています。

(3) 最小周期 “*OsCounterMinCycle*”
カウンタを利用した周期処理（アラーム，スケジュール・テーブル）で定義可能な最小周期を指定します。
なお，*OsCounterMinCycle* として指定可能な値は，0x1 ～最大カウント値 “OsCounterMaxAllowedValue” に限られます。

備考　AUTOSAR 仕様では，本項目に指定可能な最大値を 0xFFFFFFFFFFFFFFFF とする旨の規定が行われていますが，RV850 では，本項目に指定な最大値を 0x7FFFFFFF としています。

(4) 基本カウント値 “*OsCounterTicksPerBase*”
カウンタの基本カウント値を指定します。
なお，*OsCounterTicksPerBase* として指定可能な値は，0x1 ～ 0x7FFFFFFF に限られます。

備考　AUTOSAR 仕様では，本項目に指定可能な最大値を 0xFFFFFFFF とする旨の規定が行われていますが，RV850 では，本項目に指定な最大値を 0x7FFFFFFF としています。

(5) 種別 “*OsCounterType*”
カウンタの種別を指定します。
なお，*OsCounterType* として指定可能な値は，SOFTWARE，または HARDWARE に限られます。

SOFTWARE：　ソフトウエア・カウンタ
HARDWARE：　ハードウエア・カウンタ

備考　ソフトウエア・カウンタ，およびハードウエア・カウンタについての詳細は，「7.1 概　　要」を参照してください。

(6) 1 tick 当たりの秒数 “*OsSecondsPerTick*”
1 tick 当たりの秒数（単位：秒）を指定します。
なお，*OsSecondsPerTick* として指定可能な値は，0.000001 ～ 1.000000 に限られます。

備考 1.　本項目は，種別 “OsCounterType” で HARDWARE を定義した場合に限り指定可能となります。

備考 2.　AUTOSAR 仕様では，本項目に指定可能な値を 0 ～ INF とする旨の規定が行われていますが，RV850 では，本項目に指定な値を 0.000001 ～ 1.000000 としています。

(7) OS アプリケーション識別子 “*OsCounterAccessingApplication*”
カウンタへのアクセス権を与えるオブジェクト（タスク，割り込みサービス・ルーチン，アラーム）が定義されている OS アプリケーションの識別子を指定します。
なお，*OsCounterAccessingApplication* として指定可能な値は，識別子 “OsApplication” に限られます。

備考 1.　本項目は，スケーラビリティ・クラス “OsScalabilityClass” で SC3 を定義した場合に限り指定可能となります。
AUTOSAR 仕様では，スケーラビリティ・クラス “OsScalabilityClass” で SC3，SC4 以外の値が定義された場合の動作をワーニングとする旨の規定が行われていますが，RV850 では，フェイタル・エラー E4003 としています。

備考 2.　本カウンタが所属している OS アプリケーションの識別子を *OsCounterAccessingApplication* として指定する必要はありません。

備考 3.　本項目は，以下のように複数の定義（最大：31 個）が可能です。

```
ACCESSING_APPLICATION = OsApplication1;
ACCESSING_APPLICATION = OsApplication2;
ACCESSING_APPLICATION = OsApplication3;
```

【省略時解釈】
カウンタへのアクセス権を与えるオブジェクト（タスク，割り込みサービス・ルーチン，アラーム）は，本カウンタが所属している OS アプリケーションに定義されているオブジェクトに限定されているものとして処理が行われます。

(8) ハードウエア・カウンタ情報
ハードウエア・カウンタの更新処理を行う割り込みサービス・ルーチン（カテゴリ 2）に関する情報を定義します。

備考　本項目は，種別 “OsCounterType” で HARDWARE を定義した場合に限り指定可能となります。

(a) 例外コード “*OsCounterExceptionCode*”
割り込みサービス・ルーチンの起動要因となる EI レベル割り込みの例外コードを指定します。
なお，*OsCounterExceptionCode* として指定可能な値は，0x1000 ～最大例外コード “OsSystemMaxExceptionCode” に限られます。

備考 AUTOSAR 仕様では，本項目の規定が行われていません。
本項目は，RV850 が独自に追加した項目です。

(b) 優先度 “*OsCounterPriority*”
例外コード “OsCounterExceptionCode” で定義された割り込みの優先度を指定します。
なお，*OsCounterPriority* として指定可能な値は，INTPRI0 ～ INTPRI15（対象デバイスが G3K の場合は INTPRI0 ～ INTPRI7）に限られます。

備考 1. *OsCounterPriority* に指定する値は，INTPRI0 が最低位の優先度を，INTPRI15 が最高位の優先度を意味しています。

備考 2. *OsCounterPriority* には，カテゴリ “OsIsrCategory” で 0x1 が定義されている割り込みサービス・ルーチンの初期優先度 “OsIsrPriority” 以上の優先度を指定することはできません。

備考 3. AUTOSAR 仕様では，本項目の規定が行われていません。
本項目は，RV850 が独自に追加した項目です。

【省略時解釈】
INTPRI0 を指定したものとして処理が行われます。

(9) データ・マクロ情報
時間の長さを秒単位から tick 単位へと変換した結果（「カウント値 “OsTimeValue” / 1 tick 当たりの秒数 “OsSecondsPerTick”」で算出された値）をデータ・マクロとして情報ファイル（カーネル・マクロ・ファイル）に出力する場合に限り定義を行います。

備考 1. 本項目は，種別 “OsCounterType” で HARDWARE を定義した場合に限り指定可能となります。

備考 2. 本項目は，1 tick 当たりの秒数 “OsSecondsPerTick” を定義した場合に限り指定可能となります。

備考 3. 本項目は，以下のように複数の定義（最大：1023 個）が可能です。

```
TIMECONSTANTS {
    CONSTNAME = OSConstMs;
    TIMEVALUE = 0.1;
};
TIMECONSTANTS {
    CONSTNAME = OSConstSec;
    TIMEVALUE = 1;
};
TIMECONSTANTS {
    CONSTNAME = OSConstMin;
    TIMEVALUE = 60;
};
```

(a) マクロ名 “*OsConstName*”
処理プログラムからデータ参照を行う際に使用するマクロ名を指定します。
なお，*OsConstName* として指定可能な値は，名前に限られます。

(b) カウント値 “*OsTimeValue*”
時間の長さを秒単位から tick 単位へと変換する値（単位：秒）を指定します。
なお，*OsTimeValue* として指定可能な値は，0.000001 ～ 999.999999 に限られます。

備考 AUTOSAR 仕様では，本項目に指定可能な値を 0 ～ INF とする旨の規定が行われていますが，RV850 では，本項目に指定な値を 0.000001 ～ 999.999999 としています。

B.4.5 イベント情報

RV850 が提供しているイベント管理を実現するうえで必要となる情報として，以下の項目を定義します。

- 識別子 “OsEvent”
- イベント・マスク “OsEventMask”

なお，イベント情報として定義可能な数は，0 ～ 1023 個に限られます。
以下に，イベント情報の記述形式を示しますが，“ [ ] ” で囲まれた部分は省略可能な項目であることを示しています。

備考 RV850 では，本情報で定義したイベントをタスク情報で未定義の場合，フェイタル・エラー E4006 としています。

図 B.8 イベント情報の記述形式

```
EVENT OsEvent {
    [ MASK = OsEventMask; ]
};
```

(1) 識別子 “*OsEvent*”
イベントの識別子を指定します。
なお，*OsEvent* として指定可能な値は，名前に限られます。

(2) イベント・マスク “*OsEventMask*”
イベントのイベント・マスクを指定します。
なお，*OsEventMask* として指定可能な値は，0x1 ～ 0xFFFFFFFF，または AUTO に限られます。

備考 1. AUTO を指定した場合，本イベントが割り付けられている全タスクの中から未使用の最下位イベント・ビットを検出し，該当ビットに “1” を設定した値がイベント・マスクとして割り付けられます。
したがって，以下のような定義が行われていた際には，3 ビット目が未使用の最下位イベント・ビットとなるため，OsEvent3 のイベント・マスクは “0x4” となります。

```
TASK OsTask1 {
    ……
    ……
    EVENT = OsEvent1;
    EVENT = OsEvent3;
    ……
    ……
};
TASK OsTask2 {
    ……
    ……
    EVENT = OsEvent2;
    EVENT = OsEvent3;
    ……
    ……
};
EVENT OsEvent1 {
    MASK = 0x3;
};
EVENT OsEvent2 {
    MASK = 0x8;
};
EVENT OsEvent3 {
    MASK = AUTO;
};
```

備考 2. AUTOSAR 仕様では，本項目に指定可能な最大値を 0xFFFFFFFFFFFFFFFF とする旨の規定が行われていますが，RV850 では，本項目に指定な最大値を 0xFFFFFFFF としています。

備考 3. ARXML 形式の CF ファイルでは，本項目に 0 を指定することにより，AUTO と同様の扱いが行われます。

【省略時解釈】
AUTO を指定したものとして処理が行われます。

B.4.6 割り込みサービス・ルーチン情報

RV850 が提供している割り込み管理を実現するうえで必要となる情報として，以下の項目を定義します。

- 識別子 “OsIsr”
- カテゴリ “OsIsrCategory”
- 例外コード “OsIsrExceptionCode”
- 初期優先度 “OsIsrPriority”
- リソース識別子 “OsIsrResourceRef”
- タイミング保護 “OsIsrTimingProtection”

なお，割り込みサービス・ルーチン情報として定義可能な数は，0 ～ 1023 個に限られます。

備考 1. AUTOSAR 仕様では，タスク情報と本情報が未定義の場合における動作が規定されていませんが，RV850 では，フェイタル・エラー E4011 としています。

備考 2. RV850 では，スケーラビリティ・クラス “OsScalabilityClass” で SC3 を定義，本情報で定義した割り込みサービス・ルーチンを OS アプリケーション情報で未定義の場合，フェイタル・エラー E4013 としています。

以下に，割り込みサービス・ルーチン情報の記述形式を示しますが，“ [ ] ” で囲まれた部分は省略可能な項目であることを示しています。

図 B.9 割り込みサービス・ルーチン情報の記述形式

```
ISR OsIsr {
    CATEGORY = OsIsrCategory;
    EXCEPTIONCODE = OsIsrExceptionCode;
    [ PRIORITY = OsIsrPriority; ]
    [ RESOURCE = OsIsrResourceRef; ]
    [ TIMING_PROTECTION = OsIsrTimingProtection; ]
};
```

(1) 識別子 “*OsIsr*”
割り込みサービス・ルーチンの識別子を指定します。
なお，*OsIsr* として指定可能な値は，名前に限られます。

備考 割り込みサービス・ルーチンを以下のように記述した際には，*OsIsr* に設定する値は “OsIsr1” となります。

```
ISR ( OsIsr1 ) {
    ..................
    ..................
}
```

(2) カテゴリ “*OsIsrCategory*”
割り込みサービス・ルーチンのカテゴリを指定します。
なお，*OsIsrCategory* として指定可能な値は，0x1，または 0x2 に限られます。

0x1： カテゴリ 1
0x2： カテゴリ 2

備考 スケーラビリティ・クラス “OsScalabilityClass” で SC3 を定義した場合，0x1 の指定は，信頼性 “OsTrusted” で TRUE を定義した場合に限られます。

(3) 例外コード “*OsIsrExceptionCode*”
割り込みサービス・ルーチンの起動要因となる EI レベル割り込みの例外コードを指定します。
なお，*OsIsrExceptionCode* として指定可能な値は，0x1000 ～最大例外コード “OsSystemMaxExceptionCode” に限られます。

備考 1. 本項目は，以下のように複数の定義（最大：512 個）が可能です。

```
EXCEPTIONCODE = 0x1000;
EXCEPTIONCODE = 0x1010;
```

```
EXCEPTIONCODE = 0x1020;
```

備考 2. AUTOSAR 仕様では，本項目の規定が行われていません。
本項目は，RV850 が独自に追加した項目です。

(4) 初期優先度 “*OsIsrPriority*”
例外コード “OsIsrExceptionCode” で定義された割り込みの初期優先度を指定します。
なお，*OsIsrPriority* として指定可能な値は，対象デバイスの種類により異なり，INTPRI0 ～ INTPRI15（対象デバイスが G3K の場合は INTPRI0 ～ INTPRI7）に限られます。

備考 1. *OsIsrPriority* に指定する値は，INTPRI0 が最低位の優先度を，INTPRI15 が最高位の優先度を意味しています。

備考 2. カテゴリ 1 の割り込みサービス・ルーチンの *OsIsrPriority* には，カテゴリ 2 のどの割り込みサービス・ルーチンよりも高い優先度（ハードウエア・カウンタに指定された優先度 “OsCounterPriority” を含む）を指定する必要があります。

備考 3. カテゴリ “OsIsrCategory” で 0x1 を定義した場合，本項目には，優先度 “OsCounterPriority” と同値を指定することはできません。

備考 4. AUTOSAR 仕様では，本項目の規定が行われていません。
本項目は，RV850 が独自に追加した項目です。

【省略時解釈】
INTPRI0 を指定したものとして処理が行われます。

(5) リソース識別子 “*OsIsrResourceRef*”
割り込みサービス・ルーチンが操作（獲得／解放）を行うリソースの識別子を指定します。
なお，*OsIsrResourceRef* として指定可能な値は，識別子 “OsResource” に限られます。

備考 1. 本項目は，カテゴリ “OsIsrCategory” で 0x2 を定義した場合に限り指定可能となります。

備考 2. 本項目に “ インターナル・リソースのリソース識別子 ” を指定することはできません。

備考 3. スケーラビリティ・クラス “OsScalabilityClass” で SC3 を定義し，本項目に “ 本割り込みサービス・ルーチンが所属している OS アプリケーションに未所属のリソース（他 OS アプリケーションに所属しているリソース）” を指定する場合は，OS アプリケーション識別子 “OsResourceAccessingApplication” に本割り込みサービス・ルーチンが所属している OS アプリケーションの識別子を定義する必要があります。

備考 4. 本項目は，以下のように複数の定義（最大：1023）が可能です。

```
RESOURCE = OsResource1;
RESOURCE = OsResource2;
RESOURCE = OsResource3;
```

【省略時解釈】
本割り込みサービス・ルーチンは，リソースに対する操作（獲得／解放）は行わないものとして処理が行われます。

(6) タイミング保護 “*OsIsrTimingProtection*”
割り込みサービス・ルーチンが処理を実行する際，タイミング保護機能を使用するか否かを指定します。
なお，*OsIsrTimingProtection* として指定可能な値は，FALSE に限られます。

FALSE： 使用しない

備考 1. AUTOSAR 仕様では，スケーラビリティ・クラス “OsScalabilityClass” で SC1，SC3 が定義された場合，タイミング保護機能の使用を禁止する旨の規定が行われています。
そこで，RV850 では，本項目に不正な値が指定された場合，フェイタル・エラー E4003 としています。

備考 2. ARXML 形式の CF ファイルでは，本項目に 0 を指定することにより FALSE と同様の扱いが行われます。

【省略時解釈】
FALSE を指定したものとして処理が行われます。

B.4.7 OS 情報

RV850 が動作するうえで必要となる基本情報として，以下の項目を定義します。

- 識別子 “OsOS”
- 最大コア数 “OsNumberOfCores”
- スケーラビリティ・クラス “OsScalabilityClass”
- スタック・モニタリング機能 “OsStackMonitoring”
- ステータス種別 “OsStatus”
- OSErrorGetServiceId“OsUseGetServiceId”
- OSError_SystemService_Parameter“OsUseParameterAccess”
- スケジューラ・リソース “OsUseResScheduler”
- FPSR の退避／復帰 “OsSaveFpuReg”
- ベース・アドレス “OsInterruptBaseAddress”
- システム・スタック・サイズ “OsStackSize”
- FPSR のデフォルト値 “OsDefaultFPSRValue”
- StartupHook“OsStartupHook”
- ShutdownHook“OsShutdownHook”
- PostTaskHook“OsPostTaskHook”
- PreTaskHook“OsPreTaskHook”
- ErrorHook“OsErrorHook”
- ProtectionHook“OsProtectionHook”
- メモリ領域識別子 “OsMemoryAreaNameRef”
    - 属性 “OsMemoryAreaAttribute”

なお，OS 情報として定義可能な数は，1 個に限られます。
以下に，OS 情報の記述形式を示しますが，“ [ ] ” で囲まれた部分は省略可能な項目であることを示しています。

図 B.10 OS 情報の記述形式

```
OS OsOS {
    [ CORENUMBER = OsNumberOfCores; ]
    [ SCALABILITYCLASS = OsScalabilityClass; ]
    STACKMONITORING = OsStackMonitoring;
    STATUS = OsStatus;
    USEGETSERVICEID = OsUseGetServiceId;
    USEPARAMETERACCESS = OsUseParameterAccess;
    USERESSCHEDULER = OsUseResScheduler;
    SAVEFPUREG = OsSaveFpuReg;
    INTERRUPTBASE = OsInterruptBaseAddress;
    STACKSIZE = OsStackSize;
    [ DEFAULTFPSRVALUE = OsDefaultFPSRValue; ]
    STARTUPHOOK = OsStartupHook;
    SHUTDOWNHOOK = OsShutdownHook;
    POSTTASKHOOK = OsPostTaskHook;
    PRETASKHOOK = OsPreTaskHook;
    ERRORHOOK = OsErrorHook;
    PROTECTIONHOOK = OsProtectionHook;
    [ MEMORYAREA = OsMemoryAreaNameRef {
        ATTRIBUTE = OsMemoryAreaAttribute;
    }; ]
};
```

(1) 識別子 “*OsOS*”
OS の識別子を指定します。
なお，*OsOS* として指定可能な値は，名前に限られます。

(2) 最大コア数 “*OsNumberOfCores*”
RV850 が制御対象とするコアの最大数を指定します。
なお，*OsNumberOfCores* として指定可能な値は，0x1 に限られます。

備考 AUTOSAR 仕様では，本項目に指定可能な最大値を 65535 とする旨の規定が行われていますが，RV850 では，本項目に指定な値を 0x1 に限定しています。

【省略時解釈】
0x1 を指定したものとして処理が行われます。

(3) スケーラビリティ・クラス “*OsScalabilityClass*”
RV850 のスケーラビリティ・クラスを指定します。
なお，*OsScalabilityClass* として指定可能な値は，SC1，SC3，または AUTO に限られます。

SC1： スケーラビリティ・クラス 1
SC3： スケーラビリティ・クラス 3
AUTO： コンフィギュレータが CF ファイルの記述内容から最適なスケーラビリティ・クラスを選択

備考 1. AUTOSAR 仕様では，本項目に定義可能なキーワードとして，SC1，SC2，SC3，SC4 を規定していますが，RV850 では，SC2，SC4 をサポート外としています。

備考 2. AUTOSAR 仕様では，キーワード AUTO の規定が行われていません。
本キーワードは，RV850 が独自に追加したキーワードです。

【省略時解釈】
AUTO を指定したものとして処理が行われます。

(4) スタック・モニタリング機能 “*OsStackMonitoring*”
処理プログラム（タスク，割り込みサービス・ルーチン）が処理を実行する際，スタック・モニタリング機能を使用するか否を指定します。
なお，*OsStackMonitoring* として指定可能な値は，TRUE，FALSE に限られます。

TRUE： 使用する
FALSE： 使用しない

備考 1. スタック・モニタリング機能についての詳細は，「3.1.2 スタック・モニタリング機能」，および「4.1.1 スタック・モニタリング機能」を参照してください。

備考 2. ARXML 形式の CF ファイルでは，本項目に 1 を指定することにより TRUE と同様の扱いが，本項目に 0 を指定することにより FALSE と同様の扱いが行われます。

(5) ステータス種別 “*OsStatus*”
RV850 のステータス種別を指定します。
なお，*OsStatus* として指定可能な値は，EXTENDED に限られます。

EXTENDED： 拡張ステータス

(6) OSErrorGetServiceId“*OsUseGetServiceId*”
ユーティリティ関数 OSErrorGetServiceId を使用するか否かを指定します。
なお，*OsUseGetServiceId* として指定可能な値は，TRUE，または FALSE に限られます。

TRUE： 使用する
FALSE： 使用しない

備考 1. AUTOSAR 仕様では，ユーティリティ関数 OSErrorGetServiceId の使用有無を本項目で指定する旨の規定が行われていますが，RV850 では，本項目の定義内容に依存することなく，TRUE を指定したものとして処理が行われます。

備考 2. ARXML 形式の CF ファイルでは，本項目に 1 を指定することにより TRUE と同様の扱いが，本項目に 0 を指定することにより FALSE と同様の扱いが行われます。

(7) OSError_*SystemService_Parameter*“*OsUseParameterAccess*”
ユーティリティ関数 OSError_SystemService_Parameter を使用するか否かを指定します。
なお，*OsUseParameterAccess* として指定可能な値は，TRUE，または FALSE に限られます。

TRUE： 使用する

FALSE： 使用しない

備考 1. AUTOSAR 仕様では，ユーティリティ関数 OSError_SystemService_Parameter の使用有無を本項目で指定する旨の規定が行われていますが，RV850 では，本項目の定義内容に依存することなく，TRUE を指定したものとして処理が行われます。

備考 2. ARXML 形式の CF ファイルでは，本項目に 1 を指定することにより TRUE と同様の扱いが，本項目に 0 を指定することにより FALSE と同様の扱いが行われます。

(8) スケジューラ・リソース “*OsUseResScheduler*”
スケジューラ・リソース（識別子：RES_SCHEDULER）を使用するか否かを指定します。
なお，*OsUseResSchedule* として指定可能な値は，TRUE，または FALSE に限られます。

TRUE： 使用する
FALSE： 使用しない

備考 1. スケジューラ・リソースについての詳細は，「5.1.1 シーリング値」を参照してください。

備考 2. AUTOSAR 仕様では，スケジューラ・リソースの使用有無を本項目で指定する旨の規定が行われていますが，RV850 では，本項目の定義内容に依存することなく，識別子 “OsResource” において識別子 RES_SCHEDULER が定義されているか否かでスケジューラ・リソースの使用有無を判断しています。

備考 3. ARXML 形式の CF ファイルでは，本項目に 1 を指定することにより TRUE と同様の扱いが，本項目に 0 を指定することにより FALSE と同様の扱いが行われます。

(9) FPSR の退避／復帰 “*OsSaveFpuReg*”
処理プログラム（タスク，割り込みサービス・ルーチン，フック・ルーチンなど）の切り替えが発生した際，浮動小数点設定／状態レジスタ（FPSR）の退避／復帰を RV850 に行わせるか否かを指定します。
なお，*OsSaveFpuReg* として指定可能な値は，TRUE，または FALSE に限られます。

TRUE： RV850 に FPSR の退避／復帰を行わせる
FALSE： RV850 に FPSR の退避／復帰を行わせない

備考 1. AUTOSAR 仕様では，本項目の規定が行われていません。
本項目は，RV850 が独自に追加した項目です。

備考 2. ARXML 形式の CF ファイルでは，本項目に 1 を指定することにより TRUE と同様の扱いが，本項目に 0 を指定することにより FALSE と同様の扱いが行われます。

(10) ベース・アドレス “*OsInterruptBaseAddress*”
割り込みハンドラ・アドレス・テーブルのベース・アドレスを指定します。
なお，*OsInterruptBaseAddress* として指定可能な値は，0x0 ～ 0xFFFFFE00 の 0x200 バイト・アライン値に限られます。

備考 1. RV850 では，割り込みハンドラ・アドレスの選択方式として，テーブル参照方式を採用しています。

備考 2. AUTOSAR 仕様では，本項目の規定が行われていません。
本項目は，RV850 が独自に追加した項目です。

(11) システム・スタック・サイズ “*OsStackSize*”
システムが使用するスタックのサイズを指定します。
なお，*OsStackSize* として指定可能な値は，0x4 ～ 0xFFFFFFC の 0x4 バイト・アライン値に限られます。

備考 1. 本項目で指定するサイズについての詳細は，「C.7.1 システム・スタック」を参照してください。

備考 2. AUTOSAR 仕様では，本項目の規定が行われていません。
本項目は，RV850 が独自に追加した項目です。

(12) FPSR のデフォルト値 “*OsDefaultFPSRValue*”
処理プログラム（タスク，割り込みサービス・ルーチン，フック・ルーチンなど）を起動する際，浮動小数点設定／状態レジスタ（FPSR）に設定するデフォルト値を指定します。
なお，*OsDefaultFPSRValue* として指定可能な値は，0x0 ～ 0xFFFFFFFF に限られます。

備考 1. 本項目は，FPSR の退避／復帰 “OsSaveFpuReg” で TRUE を定義した場合に限り指定可能となります。

備考 2. OS アプリケーションに所属している処理プログラム（タスク，割り込みサービス・ルーチン，フック・ルーチン）については，FPSR“OsAppDefaultFPSRValue” で定義した値が優先的に設定されます。

備考 3. AUTOSAR 仕様では，本項目の規定が行われていません。

本項目は，RV850 が独自に追加した項目です。

【省略時解釈】
0x20000 を指定したものとして処理が行われます。

(13) StartupHook“*OsStartupHook*”
共通フック・ルーチン（StartupHook）を使用するか否かを指定します。
なお，*OsStartupHook* として指定可能な値は，TRUE，または FALSE に限られます。

TRUE： 使用する
FALSE： 使用しない

備考 ARXML 形式の CF ファイルでは，本項目に 1 を指定することにより TRUE と同様の扱いが，本項目に 0 を指定することにより FALSE と同様の扱いが行われます。

(14) ShutdownHook“*OsShutdownHook*”
共通フック・ルーチン（ShutdownHook）を使用するか否かを指定します。
なお，*OsShutdownHook* として指定可能な値は，TRUE，または FALSE に限られます。

TRUE： 使用する
FALSE： 使用しない

備考 ARXML 形式の CF ファイルでは，本項目に 1 を指定することにより TRUE と同様の扱いが，本項目に 0 を指定することにより FALSE と同様の扱いが行われます。

(15) PostTaskHook“*OsPostTaskHook*”
共通フック・ルーチン（PostTaskHook）を使用するか否かを指定します。
なお，*OsPostTaskHook* として指定可能な値は，TRUE，または FALSE に限られます。

TRUE： 使用する
FALSE： 使用しない

備考 ARXML 形式の CF ファイルでは，本項目に 1 を指定することにより TRUE と同様の扱いが，本項目に 0 を指定することにより FALSE と同様の扱いが行われます。

(16) PreTaskHook“*OsPreTaskHook*”
共通フック・ルーチン（PreTaskHook）を使用するか否かを指定します。
なお，*OsPreTaskHook* として指定可能な値は，TRUE，または FALSE に限られます。

TRUE： 使用する
FALSE： 使用しない

備考 ARXML 形式の CF ファイルでは，本項目に 1 を指定することにより TRUE と同様の扱いが，本項目に 0 を指定することにより FALSE と同様の扱いが行われます。

(17) ErrorHook“*OsErrorHook*”
共通フック・ルーチン（ErrorHook）を使用するか否かを指定します。
なお，*OsErrorHook* として指定可能な値は，TRUE，または FALSE に限られます。

TRUE： 使用する
FALSE： 使用しない

備考 ARXML 形式の CF ファイルでは，本項目に 1 を指定することにより TRUE と同様の扱いが，本項目に 0 を指定することにより FALSE と同様の扱いが行われます。

(18) ProtectionHook“*OsProtectionHook*”
共通フック・ルーチン（ProtectionHook）を使用するか否かを指定します。
なお，*OsProtectionHook* として指定可能な値は，TRUE，または FALSE に限られます。

TRUE： 使用する
FALSE： 使用しない

備考 1. 本項目で FALSE を指定した場合，保護違反（スタック・オーバーフロー，不正なメモリ・アクセス，例外の発生）の検出時には ShutdownOS の発行が行われます。

備考 2. 本項目は，スケーラビリティ・クラス “OsScalabilityClass” で SC3 を定義した場合に限り TRUE の指定が可能となります。
AUTOSAR 仕様では，スケーラビリティ・クラス “OsScalabilityClass” で SC1 が，本項目で TRUE が定義された場合の動作をワーニングとする旨の規定が行われていますが，RV850 では，フェイタル・エラー E4003 としています。

備考 3. ARXML 形式の CF ファイルでは，本項目に 1 を指定することにより TRUE と同様の扱いが，本項目に 0 を指定することにより FALSE と同様の扱いが行われます。

(19) メモリ領域識別子 “*OsMemoryAreaNameRef*”
全 OS アプリケーションでアクセス保護対象とするメモリ領域の識別子を指定します。
なお，*OsMemoryAreaNameRef* として指定可能な値は，メモリ領域識別子 “OsSystemMemoryArea” に限られます。

備考 1. 本項目は，スケーラビリティ・クラス “OsScalabilityClass” で SC3 を定義した場合に限り指定可能となります。

備考 2. 本項目は，以下のように複数の定義（コア識別子 “OsSystemCpuCore” で G3K を定義した場合：最大 3 個，コア識別子 “OsSystemCpuCore” で G3M，G3KH，G3MH のいずれかを定義した場合：最大 7 個）が可能です。

```
MEMORYAREA = OsSystemMemoryArea11 {
    ATTRIBUTE = CODE;
};
MEMORYAREA = OsSystemMemoryArea12 {
    ATTRIBUTE = CONST;
};
MEMORYAREA = OsSystemMemoryArea13 {
    ATTRIBUTE = DATA;
};
```

備考 3. アクセス保護対象が OS アプリケーション固有となるメモリ領域については，メモリ領域識別子 “OsAppMemoryAreaNameRef” で定義を行います。

備考 4. AUTOSAR 仕様では，本項目の規定が行われていません。
本項目は，RV850 が独自に追加した項目です。

(a) 属性 “*OsMemoryAreaAttribute*”
メモリ領域に対して付与する属性を指定しています。
なお，*OsMemoryAreaAttribute* として指定可能な値は，CODE，CONST，または DATA に限られます。

CODE： 読み込み可，および実行可
CONST： 読み込み可
DATA： 読み込み可，および書き込み可

備考 1. 処理プログラム（タスク，割り込みサービス・ルーチン，フック・ルーチン）が本項目で指定した以外のメモリ・アクセスを行った際には，ProtectionHook“OsProtectionHook” の定義内容にしたがった処理（ProtectionHook の呼び出し，または ShutdownOS の発行）が行われます。

備考 2. AUTOSAR 仕様では，本項目の規定が行われていません。
本項目は，RV850 が独自に追加した項目です。

B.4.8 リソース情報

RV850 が提供しているリソース管理を実現するうえで必要となる情報として，以下の項目を定義します。

- 識別子 “OsResource”
- シーリング値 “OsResourcePriority”
- 種別 “OsResourceProperty”
- OS アプリケーション識別子 “OsResourceAccessingApplication”
- リソース識別子 “OsResourceLinkedResourceRef”

なお，リソース情報として定義可能な数は，0 ～ 1023 個（スケジューラ・リソースを使用する場合は 1022 個）に限られます。

備考　RV850 では，本情報で定義されたカウンタがタスク情報，または割り込みサービス・ルーチン情報で利用されていない場合，フェイタル・エラー E4006 としています。

以下に，リソース情報の記述形式を示しますが，“ [　] ” で囲まれた部分は省略可能な項目であることを示しています。

図 B.11　リソース情報の記述形式

```
RESOURCE OsResource {
    [ PRIORITY = OsResourcePriority; ]
    RESOURCEPROPERTY = OsResourceProperty;
    [ ACCESSING_APPLICATION = OsResourceAccessingApplication; ]
    [ LINKEDRESOURCE = OsResourceLinkedResourceRef; ]
};
```

(1) 識別子 “*OsResource*”
リソースの識別子を指定します。
なお，*OsResource* として指定可能な値は，名前に限られます。

備考　スケジューラ・リソースを使用する際には，本項目に RES_SCHEDULER を指定してください。
なお，スケジューラ・リソースについての詳細は，「5.1.1 シーリング値」を参照してください。

(2) シーリング値 “*OsResourcePriority*”
リソースのシーリング値を指定します。
なお，*OsResourcePriority* として指定可能な値は，0 ～ 29，INTPRI0 ～ INTPRI15（対象デバイスが G3K の場合は INTPRI0 ～ INTPRI7），または AUTO に限られます。

備考 1.　本項目は省略可能で，省略時には備考 5. の通りにシーリング値を自動的に算出して割り当てます。スケジューラ・リソースを生成する場合以外は，本項目によるシーリング値の指定を省略し，その代わりにタスクが操作するリソース識別子 “OsTaskResourceRef”，割り込みサービス・ルーチン（カテゴリ 2）が操作するリソース識別子 “OsIsrResourceRef” を指定することを推奨します。
シーリング値についての詳細は，「5.1.1 シーリング値」を参照してください。

備考 2.　*OsResourcePriority* に指定する値は，本リソースを使用する処理プログラムの優先度以上，本リソースが所属している OS アプリケーションの全割り込みサービス・ルーチン（カテゴリ 2）の最高位の優先度以下とする必要があります。

備考 3.　スケジューラ・リソースを生成する際には，識別子 “OsResource” に RES_SCHEDULER を定義し，*OsResourcePriority* には，優先度 29 を指定する必要があります。

備考 4.　*OsResourcePriority* に指定する値 “0 ～ 29” は，0 が最低位の優先度であることを，29 が最高位の優先度であることを，また “INTPRIx” は，INTPRI0 が最低位の優先度であることを，INTPRI15 が最高位の優先度であることを意味しています。

備考 5.　AUTO を指定した場合，本リソースを使用すると CF ファイル上で定義された全タスク／全割り込みサービス・ルーチンの中から最高位の優先度を算出し，該当値がシーリング値として割り付けられます。
したがって，以下のような定義が行われていた場合には，OsResource1 のシーリング値は “INTPRI5” となります。

```
TASK OsTask1 {
    ……
    ……
    PRIORITY = 29;
    ……
    ……
    RESOURCE = OsResource1;
    ……
    ……
};
ISR OsISR1 {
    ……
    ……
    PRIORITY = INTPRI5;
    ……
    ……
    RESOURCE = OsResource1;
    ……
    ……
};
RESOURCE OsResource1 {
    PRIORITY = AUTO;
};
```

備考 6. AUTOSAR 仕様では，本項目の規定が行われていません。
本項目は，RV850 が独自に追加した項目です。

【省略時解釈】
AUTO を指定したものとして処理が行われます。

(3) 種別 “*OsResourceProperty*”
リソースの種別を指定します。
なお，*OsResourceProperty* として指定可能な値は，STANDARD，INTERNAL，または LINKED に限られます。

STANDARD： 通常のリソース
INTERNAL： インターナル・リソース
LINKED： リンクト・リソース

備考 通常のリソース，インターナル・リソース，およびリンクト・リソースについての詳細は，「5.1 概　要」を参照してください。

(4) OS アプリケーション識別子 “*OsResourceAccessingApplication*”
リソースへのアクセス権を与えるオブジェクト（タスク，割り込みサービス・ルーチン）が定義されている OS アプリケーションの識別子を指定します。
なお，*OsResourceAccessingApplication* として指定可能な値は，識別子 “OsApplication” に限られます。

備考 1. 本項目は，スケーラビリティ・クラス “OsScalabilityClass” で SC3 を定義した場合に限り指定可能となります。種別 “OsResourceProperty” で STANDARD，および INTERNAL を指定した場合は，本項目は省略できません。

備考 2. 種別 “OsResourceProperty” で LINKED を定義した場合，本リソースの OS アプリケーション識別子はリソース識別子 “OsResourceLinkedResourceRef” で定義したリソースの OS アプリケーション識別子と同値が割り付けられるため，本項目を定義する必要はありません。

備考 3. 本項目は，以下のように複数の定義（最大：31 個）が可能です。

```
ACCESSING_APPLICATION = OsApplication1;
ACCESSING_APPLICATION = OsApplication2;
ACCESSING_APPLICATION = OsApplication3;
```

【省略時解釈】
スケーラビリティ・クラス “OsScalabilityClass” で SC1 を定義した場合は，本項目を指定していないものとして処理が行われます。

(5) リソース識別子 “*OsResourceLinkedResourceRef*”
シーリング値を引き継ぐリソースの識別を指定します。

なお，*OsResourceLinkedResourceRef* として指定可能な値は，識別子 “OsResource” に限られます。

備考　本項目は，種別 “OsResourceProperty” で LINKED を定義した場合に限り指定可能となります。

B.4.9 スケジュール・テーブル情報

RV850 が提供しているスケジュール・テーブル管理を実現するうえで必要となる情報として，以下の項目を定義します。

- 識別子 “OsScheduleTable”
- スケジュール・カウント値 “OsScheduleTableDuration”
- 周期属性 “OsScheduleTableRepeating”
- OS アプリケーション識別子 “OsSchTblAccessingApplication”
- カウンタ識別子 “OsScheduleTableCounterRef”
- 初期状態 “OsScheduleTableAutostart”
  - 種別 “OsScheduleTableAutostartType”
  - オフセット・カウント値 “OsScheduleTableStartValue”
  - アプリケーション・モード “OsScheduleTableAppModeRef”
- 満了条件／満了処理
  - 満了カウント値 “OsScheduleTblExpPointOffset”
  - 満了処理（タスクの起動）
    - タスク識別子 “OsScheduleTableActivateTaskRef”
  - 満了処理（イベント・マスクの設定）
    - イベント識別子 “OsScheduleTableSetEventRef”
    - タスク識別子 “OsScheduleTableSetEventTaskRef”

なお，スケジュール・テーブル情報として定義可能な数は，アラーム情報との合計で 0 ～ 1023 個に限られます。

備考 1. RV850 では，スケーラビリティ・クラス “OsScalabilityClass” で SC3 を定義，本情報で定義したカウンタを OS アプリケーション情報で未定義の場合，フェイタル・エラー E4013 としています。

備考 2. AUTOSAR 仕様では，スケジュール・テーブル情報として，項目 ADJUSTABLEEXPPOINT，LOCAL_TO_GLOBAL_TIME_SYNCHRONIZATION が規定されていますが，RV850 では，該当項目をサポート外としています。

以下に，スケジュール・テーブル情報の記述形式を示しますが，“ [ ] ” で囲まれた部分は省略可能な項目であることを示しています。

図 B.12 スケジュール・テーブル情報の記述形式

```
SCHEDULETABLE OsScheduleTable {
    LENGTH = OsScheduleTableDuration;
    PERIODIC = OsScheduleTableRepeating;
    [ ACCESSING_APPLICATION = OsSchTblAccessingApplication; ]
    COUNTER = OsScheduleTableCounterRef;
    [ AUTOSTART = OsScheduleTableAutostart {
        TYPE = OsScheduleTableAutostartType;
        STARTVALUE = OsScheduleTableStartValue;
        [ APPMODE = OsScheduleTableAppModeRef; ]
    }; ]
    ACTION {
        OFFSET = OsScheduleTblExpPointOffset;
        [ ACTIVATETASK {
            TASK = OsScheduleTableActivateTaskRef;
        }; ]
        [ SETEVENT {
            EVENT = OsScheduleTableSetEventRef;
            TASK = OsScheduleTableSetEventTaskRef;
        }; ]
    };
};
```

(1) 識別子 “*OsScheduleTable*”
スケジュール・テーブルの識別子を指定します。
なお，*OsScheduleTable* として指定可能な値は，名前に限られます。

(2) スケジュール・カウント値 “*OsScheduleTableDuration*”
スケジュール・テーブルのスケジュール・カウント値を指定します。
なお，*OsScheduleTableDuration* として指定可能な値は，最小周期 “OsCounterMinCycle” ～最大カウント値 “OsCounterMaxAllowedValue” に 0x1 を加算した値までに限られます。

(3) 周期属性 “*OsScheduleTableRepeating*”
スケジュール・テーブルに周期属性を付与するか否かを指定します。
なお，*OsScheduleTableRepeating* として指定可能な値は，TRUE，または FALSE に限られます。

TRUE： 周期属性を付与
FALSE： 周期属性を付与しない

備考 1. スケジュール・テーブルは，*OsScheduleTableRepeating* に TRUE が指定された場合は “ 周期スケジュール・テーブル ” として，FALSE が指定された場合は “ ワンショット・スケジュール・テーブル ” として動作します。

備考 2. ARXML 形式の CF ファイルでは，本項目に 1 を指定することにより TRUE と同様の扱いが，本項目に 0 を指定することにより FALSE と同様の扱いが行われます。

(4) OS アプリケーション識別子 “*OsSchTblAccessingApplication*”
スケジュール・テーブルへのアクセス権を与えるオブジェクト（タスク，割り込みサービス・ルーチン，カウンタ）が定義されている OS アプリケーションの識別子を指定します。
なお，*OsSchTblAccessingApplication* として指定可能な値は，識別子 “OsApplication” に限られます。

備考 1. 本項目は，スケーラビリティ・クラス “OsScalabilityClass” で SC3 を定義した場合に限り指定可能となります。

備考 2. 本スケジュール・テーブルが所属している OS アプリケーションの識別子を *OsSchTblAccessingApplication* として指定する必要はありません。

備考 3. 本項目は，以下のように複数の定義（最大：31 個）が可能です。

```
ACCESSING_APPLICATION = OsApplication1
ACCESSING_APPLICATION = OsApplication2;
ACCESSING_APPLICATION = OsApplication3;
```

【省略時解釈】
スケジュール・テーブルへのアクセス権を与えるオブジェクト（タスク，割り込みサービス・ルーチン，カウンタ）は，本スケジュール・テーブルが所属している OS アプリケーションに定義されているオブジェクトに限定されているものとして処理が行われます。

(5) カウンタ識別子 “*OsScheduleTableCounterRef*”
関連付けを行うカウンタ（開始条件／満了条件が成立したか否かの確認対象となるカウント値を保有しているカウンタ）の識別子を指定します。
なお，*OsScheduleTableCounterRef* として指定可能な値は，識別子 “OsCounter” に限られます。

備考 スケーラビリティ・クラス “OsScalabilityClass” で SC3 を定義し，本項目に “ 本スケジュール・テーブルが所属している OS アプリケーションに未所属のカウンタ（他 OS アプリケーションに所属しているカウンタ）” を指定する場合は，OS アプリケーション識別子 “OsSchTblAccessingApplication” に該当カウンタが所属している OS アプリケーションの識別子を定義する必要があります。

(6) 初期状態 “*OsScheduleTableAutostart*”
スケジュール・テーブルの初期状態を指定します。
なお，*OsScheduleTableAutostart* として指定可能な値は，TRUE，または FALSE に限られます。

TRUE： StartOS の引数 *Mode* に依存
FALSE： STOPPED 状態

備考 1. 本項目で “TRUE” を指定した場合，スケジュール・テーブルの初期状態は，StartOS の引数 *Mode* に設定する値により以下のように変化します。

【引数 *Mode* がアプリケーション・モード “OsScheduleTableAppModeRef” の値と一致】

- RUNNING 状態（その他の組みあわせを定義）

【引数 *Mode* がアプリケーション・モード “OsScheduleTableAppModeRef” の値と不一致】

- STOPPED 状態

備考 2. FALSE を指定する際の指定形式は，以下のようになります。

```
AUTOSTART = FALSE;
```

備考 3. ARXML 形式の CF ファイルでは，本項目に 1 を指定することにより TRUE と同様の扱いが，本項目に 0 を指定することにより FALSE と同様の扱いが行われます。

【省略時解釈】
FALSE を指定したものとして処理が行われます。

(a) 種別 “*OsScheduleTableAutostartType*”
スケジュール・テーブルの種別を指定します。
なお，*OsScheduleTableAutostartType* として指定可能な値は，ABSOLUTE，または RELATIVE に限られます。

ABSOLUTE： 絶対スケジュール・テーブル
RELATIVE： 相対スケジュール・テーブル

備考 AUTOSAR 仕様では，本項目に不正な値が指定された場合の動作をワーニングとする旨の規定が行われていますが，RV850 では，フェイタル・エラー E4003 としています。

(b) オフセット・カウント値 “*OsScheduleTableStartValue*”
StartOS を発行してからスケジュール・カウントを開始するまでのオフセット・カウント値を指定します。
なお，*OsScheduleTableRelOffset* として指定可能な値は，種別 “OsScheduleTableAutostartType” の定義内容に依存します。

- ABSOLUTE の場合
  0x0 ～「最大カウント値 “OsCounterMaxAllowedValue” - *InitialOffset*」で算出された値に限られます。
- RELATIVE の場合
  0x1 ～「最大カウント値 “OsCounterMaxAllowedValue” - *InitialOffset*」で算出された値に限られます。

備考 *InitialOffset* は，本スケジュール・テーブルに定義されている満了カウント値 “OsScheduleTblExpPointOffset” の最小値を意味しています。

(c) アプリケーション・モード “*OsScheduleTableAppModeRef*”
スケジュール・テーブルのアプリケーション・モードを指定します。
なお，*OsScheduleTableAppModeRef* として指定可能な値は，アプリケーション・モード “OsAppMode” に限られます。

備考 1. デフォルト・アプリケーション・モード OSDEFAULTAPPMODE については，定義の有無に関わらず，定義されていたものとして処理が行われます。

備考 2. 本項目は，以下のように複数の定義（最大：127 個）が可能です。

```
APPMODE = OSDEFAULTAPPMODE;
APPMODE = OsAppMode1;
APPMODE = OsAppMode2;
```

【省略時解釈】
AUTOSAR 仕様では，本項目は省略不可とする旨の規定が行われていますが，RV850 では，本項目が省略された際には，OSDEFAULTAPPMODE を指定したものとして処理が行われます。

(7) 満了条件／満了処理
スケジュール・テーブルの満了条件，および満了処理を指定します。

備考 本項目は，以下のように複数の定義（最大：1023 個）が可能です。

```
ACTION {
    OFFSET = 0x7;
    ACTIVATETASK {
        TASK = OsTask2;
    };
};
ACTION {
    OFFSET = 0x3;
```

```
    SETEVENT {
        EVENT = OsEvent1;
        TASK = OsTask1;
    };
};
ACTION {
    OFFSET = 0xF;
    ACTIVATETASK {
        TASK = OsTask4;
    };
    SETEVENT {
        EVENT = OsEvent3;
        TASK = OsTask3;
    };
};
```

(a) 満了カウント値 “*OsScheduleTblExpPointOffset*”
スケジュール・テーブルの満了カウント値を指定します。
なお，*OsScheduleTblExpPointOffset* として指定可能な値は，周期属性 “OsScheduleTableRepeating” の定義内容に依存します。

- TRUE の場合
  0x0，または最小周期 “OsCounterMinCycle” ～「スケジュール・カウント値 “OsScheduleTableDuration” - 最小周期 “OsCounterMinCycle”」で算出された値に限られます。
- FALSE の場合
  0x0，または最小周期 “OsCounterMinCycle” ～スケジュール・カウント値 “OsScheduleTableDuration” に限られます。

備考 1. *OsScheduleTblExpPointOffset* は，オフセット・カウント値 “OsScheduleTableStartValue” から満了処理を実行するまでの相対カウント値となります。

備考 2. 複数の満了条件／満了処理を定義する場合，各満了ポイント（満了処理の実行タイミング）の間隔は，最小周期 “OsCounterMinCycle” 以上の間隔である必要があります。

(b) 満了処理（タスクの起動）
スケジュール・テーブルの満了処理として，タスクの起動処理（ActivateTask と同等の処理）を実行するか否かを指定します。

備考 本項目は，以下のように複数の定義（最大：1023 個）が可能です。
なお，1 満了条件に対して定義可能な満了処理の最大数は，“ タスクの起動 ACTIVATETASK” と “ イベント・マスクの設定 SETEVENT” の合計が 1023 個以下となります。

```
ACTIVATETASK {
    TASK = OsTask1;
};
ACTIVATETASK {
    TASK = OsTask2;
};
ACTIVATETASK {
    TASK = OsTask3;
};
```

<1> タスク識別子 “*OsScheduleTableActivateTaskRef*”
満了条件が成立した際に起動するタスクの識別子を指定します。
なお，*OsScheduleTableActivateTaskRef* として指定可能な値は，識別子 “OsTask” に限られます。

備考 スケーラビリティ・クラス “OsScalabilityClass” で SC3 を定義し，本項目に “ 本スケジュール・テーブルが所属している OS アプリケーションに未所属のタスク（他 OS アプリケーションに所属しているタスク）” を指定する場合は，OS アプリケーション識別子 “OsTaskAccessingApplication” に本スケジュール・テーブルが所属している OS アプリケーションの識別子を定義する必要があります。

(c) 満了処理（イベント・マスクの設定）
スケジュール・テーブルの満了処理として，イベント・マスクの設定処理（SetEvent と同等の処理）を実行するか否かを指定します。

備考　本項目は，以下のように複数の定義（最大：1023 個）が可能です。
なお，1 満了条件に対して定義可能な満了処理の最大数は，“ タスクの起動 ACTIVATETASK” と “ イベント・マスクの設定 SETEVENT” の合計が 1023 個以下となります。

```
SETEVENT {
    EVENT = OsEvent1;
    TASK = OsTask1;
};
SETEVENT {
    EVENT = OsEvent2;
    TASK = OsTask2;
};
SETEVENT {
    EVENT = OsEvent2;
    TASK = OsTask2;
};
SETEVENT {
    EVENT = OsEvent3;
    TASK = OsTask3;
};
```

<1> イベント識別子 “*OsScheduleTableSetEventRef*”
満了条件が成立した際に設定するイベント・マスクを保持しているイベントの識別子を指定します。
なお，*OsScheduleTableSetEventRef* として指定可能な値は，識別子 “OsEvent” に限られます。

備考　本項目で指定するイベントは，タスク識別子 “OsScheduleTableSetEventTaskRef” で指定するタスクに割り付けられている必要があります。

<2> タスク識別子 “*OsScheduleTableSetEventTaskRef*”
満了条件が成立した際にイベント・マスクを設定するタスクの識別子を指定します。
なお，*OsScheduleTableSetEventTaskRef* として指定可能な値は，識別子 “OsTask” に限られます。

備考　スケーラビリティ・クラス “OsScalabilityClass” で SC3 を定義し，本項目に “ 本スケジュール・テーブルが所属している OS アプリケーションに未所属のタスク（他 OS アプリケーションに所属しているタスク）” を指定する場合は，OS アプリケーション識別子 “OsTaskAccessingApplication” に本スケジュール・テーブルが所属している OS アプリケーションの識別子を定義する必要があります。

B.4.10 タスク情報

RV850 が提供しているタスク管理を実現するうえで必要となる情報として，以下の項目を定義します。

- 識別子 “OsTask”
- 最大起動要求数 “OsTaskActivation”
- 初期優先度 “OsTaskPriority”
- スケジューリング属性 “OsTaskSchedule”
- タスク・スタック・サイズ “OsTaskStackSize”
- OS アプリケーション識別子 “OsTaskAccessingApplication”
- イベント識別子 “OsTaskEventRef”
- リソース識別子 “OsTaskResourceRef”
- 初期状態 “OsTaskAutostart”
    - アプリケーション・モード “OsTaskAppModeRef”
- タイミング保護 “OsTaskTimingProtection”

なお，タスク情報として定義可能な数は，0 ～ 1023 個に限られます。

備考 1. AUTOSAR 仕様では，本情報と割り込みサービス・ルーチン情報が未定義の場合における動作が規定されていませんが，RV850 では，フェイタル・エラー E4011 としています。

備考 2. RV850 では，スケーラビリティ・クラス “OsScalabilityClass” で SC3 を定義，本情報で定義したタスクを OS アプリケーション情報で未定義の場合，フェイタル・エラー E4013 としています。

以下に，タスク情報の記述形式を示しますが，“ [ ] ” で囲まれた部分は省略可能な項目であることを示しています。

図 B.13 タスク情報の記述形式

```
TASK OsTask {
    ACTIVATION = OsTaskActivation;
    PRIORITY = OsTaskPriority;
    SCHEDULE = OsTaskSchedule;
    [ STACKSIZE = OsTaskStackSize; ]
    [ ACCESSING_APPLICATION = OsTaskAccessingApplication; ]
    [ EVENT = OsTaskEventRef; ]
    [ RESOURCE = OsTaskResourceRef; ]
    [ AUTOSTART = OsTaskAutostart {
        [ APPMODE = OsTaskAppModeRef; ]
    }; ]
    [ TIMING_PROTECTION = OsTaskTimingProtection; ]
};
```

(1) 識別子 “*OsTask*”
タスクの識別子を指定します。
なお，*OsTask* として指定可能な値は，名前に限られます。

備考 タスクを以下のように記述した際には，*OsTask* に設定する値は “OsTask1” となります。

```
TASK ( OsTask1 ) {
    ..................
    ..................
}
```

(2) 最大起動要求数 “*OsTaskActivation*”
起動要求数としてカウント可能な最大起動要求数を指定します。
なお，*OsTaskActivation* として指定可能な値は，0x1 ～ 0x7F に限られます。

備考 1. イベント識別子 “OsTaskEventRef” を定義した場合，*OsTaskActivation* に 0x1 以外の値を指定することができません。

備考 2. AUTOSAR 仕様では，本項目に指定可能な最大値を 0xFFFFFFFF とする旨の規定が行われていますが，RV850 では，本項目に指定な最大値を 0x7F としています。

(3) 初期優先度 “*OsTaskPriority*”
タスクの初期優先度を指定します。
なお，*OsTaskPriority* として指定可能な値は，0 ～ 29 に限られます。

備考 1. *OsTaskPriority* に指定する値 “0 ～ 29” は，0 が最低位の優先度であることを，29 が最高位の優先度であることを意味しています。

備考 2. AUTOSAR 仕様では，本項目に指定可能な最大値を 0xFFFFFFFF とする旨の規定が行われていますが，RV850 では，本項目に指定な最大値を 29 としています。

(4) スケジューリング属性 “*OsTaskSchedule*”
タスクのスケジューリング属性を指定します。
なお，*OsTaskSchedule* として指定可能な値は，NON，または FULL に限られます。

NON： 非プリエンプティブ属性
FULL： プリエンプティブ属性

備考 スケジューリング属性についての詳細は，「12.1 概　　要」を参照してください。

(5) タスク・スタック・サイズ “*OsTaskStackSize*”
タスクのスタック・サイズ（単位：バイト）を指定します。
なお，*OsTaskStackSize* として指定可能な値は，0x4 ～ 0xFFFFFFC の 0x4 バイト・アライン値に限られます。

備考 1. イベント識別子 “OsTaskEventRef” の定義を行った場合，本項目を指定することは必須となります。

備考 2. 本項目で指定するサイズについての詳細は，「C.7.3 タスク・スタック（拡張タスク）」を参照してください。

備考 3. AUTOSAR 仕様では，本項目の規定が行われていません。
本項目は，RV850 が独自に追加した項目です。

(6) OS アプリケーション識別子 “*OsTaskAccessingApplication*”
タスクへのアクセス権を与えるオブジェクト（タスク，割り込みサービス・ルーチン，アラーム，スケジュール・テーブル）が定義されている OS アプリケーションの識別子を指定します。
なお，*OsTaskAccessingApplication* として指定可能な値は，識別子 “OsApplication” に限られます。

備考 1. 本項目は，スケーラビリティ・クラス “OsScalabilityClass” で SC3 を定義した場合に限り指定可能となります。
AUTOSAR 仕様では，スケーラビリティ・クラス “OsScalabilityClass” で SC3，SC4 以外の値が定義された場合の動作をワーニングとする旨の規定が行われていますが，RV850 では，フェイタル・エラー E4003 としています。

備考 2. 本タスクが所属している OS アプリケーションの識別子を *OsTaskAccessingApplication* として指定する必要はありません。

備考 3. 本項目は，以下のように複数の定義（最大：31 個）が可能です。

```
ACCESSING_APPLICATION = OsApplication1;
ACCESSING_APPLICATION = OsApplication2;
ACCESSING_APPLICATION = OsApplication3;
```

【省略時解釈】
タスクへのアクセス権を与えるオブジェクト（タスク，割り込みサービス・ルーチン，アラーム，スケジュール・テーブル）は，本タスクが所属している OS アプリケーションに定義されているオブジェクトに限定されているものとして処理が行われます。

(7) イベント識別子 “OsTaskEventRef”
タスクに割り付けるイベントの識別子を指定します。
なお，*OsTaskEventRef* として指定可能な値は，識別子 “OsEvent” に限られます。

備考 本項目は，以下のように複数の定義（最大：32 個）が可能です。

```
EVENT = OsEvent1;
EVENT = OsEvent2;
EVENT = OsEvent3;
```

【省略時解釈】
タスクにイベントを割り付けないものとして処理が行われます。

(8) リソース識別子 “*OsTaskResourceRef*”
タスクが操作（獲得／解放）を行うリソースの識別子を指定します。
なお，*OsTaskResourceRef* として指定可能な値は，識別子 “OsResource” に限られます。

備考 1. スケーラビリティ・クラス “OsScalabilityClass” で SC3 を定義した場合は，OS アプリケーション識別子 “OsResourceAccessingApplication” に本タスクが所属している OS アプリケーションの識別子を定義する必要があります。

備考 2. 本項目は，以下のように複数の定義（最大：1023 個）が可能です。
ただし，種別 “OsResourceProperty” で INTERNAL を定義したリソースについては，1 個の指定に限定されます。

```
RESOURCE = OsResource1;
RESOURCE = OsResource2;
RESOURCE = OsResource3;
```

【省略時解釈】
本タスクは，リソースに対する操作（獲得／解放）は行わないものとして処理が行われます。

(9) 初期状態 “*OsTaskAutostart*”
タスクの初期状態を指定します。
なお，*OsTaskAutostart* として指定可能な値は，TRUE，または FALSE に限られます。

TRUE： StartOS の引数 *Mode* に依存
FALSE： SUSPENDED 状態

備考 1. 本項目で “TRUE” を指定した場合，タスクの初期状態は，StartOS の引数 *Mode* に設定する値により以下のように変化します。

【引数 *Mode* がアプリケーション・モード “OsTaskAppModeRef” の値と一致】

- READY 状態

【引数 *Mode* がアプリケーション・モード “OsTaskAppModeRef” の値と不一致】

- SUSPENDED 状態

備考 2. FALSE を指定する際の指定形式は，以下のようになります。

```
AUTOSTART = FALSE;
```

備考 3. ARXML 形式の CF ファイルでは，本項目に 1 を指定することにより TRUE と同様の扱いが，本項目に 0 を指定することにより FALSE と同様の扱いが行われます。

【省略時解釈】
FALSE を指定したものとして処理が行われます。

(a) アプリケーション・モード “*OsTaskAppModeRef*”
タスクのアプリケーション・モードを指定します。
なお，*OsTaskAppModeRef* として指定可能な値は，アプリケーション・モード “OsAppMode” に限られます。

備考 1. デフォルト・アプリケーション・モード OSDEFAULTAPPMODE については，定義の有無に関わらず，定義されていたものとして処理が行われます。

備考 2. 本項目は，以下のように複数の定義（最大：127 個）が可能です。

```
APPMODE = OSDEFAULTAPPMODE;
APPMODE = OsAppMode1;
APPMODE = OsAppMode2;
```

【省略時解釈】
AUTOSAR 仕様では，本項目は省略不可とする旨の規定が行われていますが，RV850 では，本項目が省略された際には，OSDEFAULTAPPMODE を指定したものとして処理が行われます。

(10) タイミング保護 “*OsTaskTimingProtection*”
タスクが処理を実行する際，タイミング保護機能を使用するか否かを指定します。

なお，*OsTaskTimingProtection* として指定可能な値は，FALSE に限られます。

FALSE： 使用しない

備考 1. AUTOSAR 仕様では，スケーラビリティ・クラス “OsScalabilityClass” で SC1，SC3 が定義された場合，タイミング保護機能の使用を禁止する旨の規定が行われています。
そこで，RV850 では，本項目に不正な値が指定された場合，フェイタル・エラー E4003 としています。

備考 2. ARXML 形式の CF ファイルでは，本項目に 1 を指定することにより TRUE と同様の扱いが，本項目に 0 を指定することにより FALSE と同様の扱いが行われます。

【省略時解釈】
FALSE を指定したものとして処理が行われます。

B.4.11 システム情報

RV850 が動作するうえで必要となる基本情報，および RV850 が提供しているメモリ・アクセス保護機能を実現するうえで必要となる情報として，以下の項目を定義します。

- 最大例外コード “OsSystemMaxExceptionCode”
- 動作周波数 “OsSystemSystemClock”
- システム SPID 値 “OsSystemSystemSPID”
- コア識別子 “OsSystemCpuCore”
- INTC1 用 EICn“OsSystemINTC1EIControlAddress”
- INTC1 用 IMRm“OsSystemINTC1EIMaskAddress”
- INTC2 用 EICn“OsSystemINTC2EIControlAddress”
- INTC2 用 IMRm“OsSystemINTC2EIMaskAddress”
- メモリ領域識別子 “OsSystemMemoryArea”
    - 先頭アドレス “AreaStartAddressValue”
    - サイズ “AreaSizeValue”
    - 末尾アドレス “AreaEndAddressValue”

なお，システム情報として定義可能な数は，1 個に限られます。
以下に，システム情報の記述形式を示しますが，“ [ ] ” で囲まれた部分は省略可能な項目であることを示しています。

図 B.14 システム情報の記述形式

```
SYSTEM {
    MAXEXCEPTIONCODE = OsSystemMaxExceptionCode;
    SYSTEM_CLOCK = OsSystemSystemClock;
    SYSTEM_SPID = OsSystemSystemSPID;
    CPUCORE = OsCpuCore;
    INTC1EICTRL = OsSystemINTC1EIControlAddress;
    INTC1EIMASK = OsSystemINTC1EIMaskAddress;
    INTC2EICTRL = OsSystemINTC2EIControlAddress;
    INTC2EIMASK = OsSystemINTC2EIMaskAddress;
    [ MEMORYAREA OsSystemMemoryArea {
        STARTADDRESS = AreaStartAddressValue;
        [ SIZE = AreaSizeValue; ]
        [ ENDADDRESS = AreaEndAddressValue; ]
    }; ]
};
```

(1) 最大例外コード “*OsSystemMaxExceptionCode*”
RV850 が管理対象とする EI レベル割り込みの例外コードの最大値を指定します。
なお，*OsSystemMaxExceptionCode* として指定可能な値は，0x1000 ～ 0x11FF に限られます。

備考 AUTOSAR 仕様では，本項目の規定が行われていません。
本項目は，RV850 が独自に追加した項目です。

(2) 動作周波数 “*OsSystemSystemClock*”
対象デバイスの動作周波数（CPU クロック，単位：kHz）を指定します。
なお，*OsSystemSystemClock* として指定可能な値は，0x1 ～ 0xFFFFFFFF に限られます。

備考 AUTOSAR 仕様では，本項目の規定が行われていません。
本項目は，RV850 が独自に追加した項目です。

(3) システム SPID 値 “*OsSystemSystemSPID*”
RV850 のカーネル処理および Trusted な OS アプリケーション処理のために予約する SPID を指定します。
なお，*OsSystemSystemSPID* として指定可能な値は，0x0 ～ 0x1F に限られます。

備考 本項目は，スケーラビリティ・クラス “OsScalabilityClass” で SC3 を指定した場合に限り指定可能となります。

(4) コア識別子 “*OsSystemCpuCore*”
RV850 が制御対象とするコアの識別子を指定します。
なお，*OsSystemCpuCore* として指定格納な値は，G3K，G3M，G3KH，G3MH のいずれかに限られます。

備考 AUTOSAR 仕様では，本項目の規定が行われていません。
本項目は，RV850 が独自に追加した項目です。

(5) INTC1 用 EIC*n*“*OsSystemINTC1EIControlAddress*”
INTC1 用 EI レベル割り込み制御レジスタ（EIC*n*）の先頭アドレスを指定します。
なお，*OsSystemINTC1EIControlAddress* として指定可能な値は，0x0 ～ 0xFFFFFFFF に限られます。

備考 1. *OsSystemINTC1EIControlAddress* に指定する値についての詳細は，対象デバイスのユーザーズ・マニュアルを参照してください。

備考 2. AUTOSAR 仕様では，本項目の規定が行われていません。
本項目は，RV850 が独自に追加した項目です。

(6) INTC1 用 IMRm“*OsSystemINTC1EIMaskAddress*”
INTC1 用 EI レベル割り込みマスク・レジスタ（IMRm）のアドレスを指定します。
なお，*OsSystemINTC1EIMaskAddress* として指定可能な値は，0x0 ～ 0xFFFFFFFF に限られます。

備考 1. *OsSystemINTC1EIMaskAddress* に指定する値についての詳細は，対象デバイスのユーザーズ・マニュアルを参照してください。

備考 2. AUTOSAR 仕様では，本項目の規定が行われていません。
本項目は，RV850 が独自に追加した項目です。

(7) INTC2 用 EIC*n*“*OsSystemINTC2EIControlAddress*”
INTC2 用 EI レベル割り込み制御レジスタ（EIC*n*）の先頭アドレスを指定します。
なお，*OsSystemINTC2EIControlAddress* として指定可能な値は，0x0 ～ 0xFFFFFFFF に限られます。

備考 1. *OsSystemINTC2EIControlAddress* に指定する値についての詳細は，対象デバイスのユーザーズ・マニュアルを参照してください。

備考 2. AUTOSAR 仕様では，本項目の規定が行われていません。
本項目は，RV850 が独自に追加した項目です。

(8) INTC2 用 IMRm“*OsSystemINTC2EIMaskAddress*”
INTC2 用 EI レベル割り込みマスク・レジスタ（IMRm）のアドレスを指定します。
なお，*OsSystemINTC2EIMaskAddress* として指定可能な値は，0x0 ～ 0xFFFFFFFF に限られます。

備考 1. *OsSystemINTC2EIMaskAddress* に指定する値についての詳細は，対象デバイスのユーザーズ・マニュアルを参照してください。

備考 2. AUTOSAR 仕様では，本項目の規定が行われていません。
本項目は，RV850 が独自に追加した項目です。

(9) メモリ領域識別子 “*OsSystemMemoryArea*”
RV850 がアクセス保護対象とするメモリ領域の識別子を指定します。
なお，*OsSystemMemoryArea* として指定可能な値は，名前に限られます。

備考 1. 本項目は，スケーラビリティ・クラス “OsScalabilityClass” で SC3 を定義した場合に限り指定可能となります。
AUTOSAR 仕様では，スケーラビリティ・クラス “OsScalabilityClass” で SC3，SC4 以外の値が定義された場合の動作をワーニングとする旨の規定が行われていますが，RV850 では，フェイタル・エラー E4003 としています。

備考 2. 本項目は，以下のように複数の定義（最大：131 個）が可能です。

```
MEMORYAREA OsSystemMemoryArea1 {
    STARTADDRESS = 0x15000000;
    SIZE = 0x50000000;
};
MEMORYAREA OsSystemMemoryArea2 {
    STARTADDRESS = 0x30000000;
    SIZE = 0x10000000;
};
MEMORYAREA OsSystemMemoryArea3 {
    STARTADDRESS = 0x60000000;
    ENDADDRESS = 0x70000000;
};
```

(a) 先頭アドレス “*AreaStartAddressValue*”
アクセス保護対象とするメモリ領域の先頭アドレスを指定します。
なお，*AreaStartAddressValue* として指定可能な値は，0x0 ～ 0xFFFFFFFC の 0x4 バイト・アライン値，またはシンボル名に限られます。

備考 AUTOSAR 仕様では，本項目の規定が行われていません。
本項目は，RV850 が独自に追加した項目です。

(b) サイズ “*AreaSizeValue*”
アクセス保護対象とするメモリ領域のサイズ（単位：バイト）を指定します。
なお，*AreaSizeValue* として指定可能な値は，0x4 ～ 0xFFFFFFFC の 0x4 バイト・アライン値に限られます。

備考 1. 末尾アドレス “AreaEndAddressValue” を定義した場合，本項目を指定することはできません。

備考 2. AUTOSAR 仕様では，本項目の規定が行われていません。
本項目は，RV850 が独自に追加した項目です。

(c) 末尾アドレス “*AreaEndAddressValue*”
アクセス保護対象とするメモリ領域の末尾アドレスを指定します。
なお，*AreaEndAddressValue* として指定可能な値は，0x4 ～ 0xFFFFFFFC の 0x4 バイト・アライン値，0xFFFFFFFF，またはシンボル名に限られます。

備考 1. サイズ “AreaSizeValue” を定義した場合，本項目を指定することはできません。

備考 2. AUTOSAR 仕様では，本項目の規定が行われていません。
本項目は，RV850 が独自に追加した項目です。

C. メモリ使用量

本付録では，メモリ使用量について解説しています。

C.1 概　　要

RV850 が使用／管理するメモリ領域は，その用途により，7 種類のセクションに大別されます。

表 C.1　　メモリ領域

セクション名	説明
.kernel_system	標準処理領域
.kernel_interface	インタフェース領域
.kernel_const	定数データ領域（ROM）
.kernel_identifier	定数データ領域
.kernel_work	変数データ領域（RAM）
.kernel_stack	スタック領域
.kernel_*address*	割り込みハンドラ・アドレス・テーブル

C.2 標準処理領域（.kernel_system）

標準処理領域（.kernel_system）のメモリ使用量は，ロード・モジュールを生成する際にリンクするカーネル・ライブラリの種類により，以下のように異なります。

カーネル・ライブラリ	サイズ（1K バイト = 1024 バイト）
libecc2extsc1.a	12.3 K バイト
libecc2extsc1_fpu.a	12.5 K バイト
libecc2extsc3.a	22.4 K バイト
libecc2extsc3_fpu.a	22.5 K バイト
libecc2extsc3_g3k.a	21.8 K バイト

C.3 インタフェース領域（.kernel_interface）

インタフェース領域（.kernel_interface）のメモリ使用量は，ロード・モジュールを生成する際にリンクするカーネル・ライブラリの種類により，以下のように異なります。

カーネル・ライブラリ	サイズ（1K バイト = 1024 バイト）
libecc2extsc1.a	0.89 K バイト
libecc2extsc1_fpu.a	0.89 K バイト
libecc2extsc3.a	0.94 K バイト
libecc2extsc3_fpu.a	0.94 K バイト
libecc2extsc3_g3k.a	0.94 K バイト

C.4 定数データ領域（.kernel_const）

定数データ領域（.kernel_const）のメモリ使用量は，スケーラビリティ・クラスの種類により，以下のように異なります。

(1) SC1 用定数データ領域（.kernel_const）

以下に，スケーラビリティ・クラスが SC1 の場合における定数データ領域（.kernel_const）のサイズを見積もる際の計算式（単位：バイト）を示します。
なお，下記計算式における “align 4 （*x*）” は，数値 “*x*” を 4 バイト・アラインした結果を意味しています。

```
KERNEL_CONST =
    align 4 (
        673
        + 40 * Alarm_Num
        + 12 * AlarmAutostart_Num
        + 12 * AppMode_Num
        + 16 * Counter_Num
        + 16 * CounterType_Num
        + 16 * Isr_Num
        + 4 * Resource_Num
        + 32 * ScheduleTable_Num
        + 12 * ScheduleTableAutostart_Num
        + 16 * ScheduleTblExpPointOffset_Num
        + 8 * ScheduleTableAction_Num
        + 20 * Task_Num
        + 2 * TaskAutostart_Num
        + 4 * (SystemMaxExceptionCode - 4095)
    )
```

上記計算式の各キーワードは，以下に示した意味を持ちます。

キーワード	意味
Alarm_Num	アラーム情報の総数
AlarmAutostart_Num	各アプリケーション・モードで自動起動するアラームの総数
AppMode_Num	アプリケーション・モード情報の総数 （OSDEFAULTAPPMODE に関する定義は除く）
Counter_Num	カウンタ情報の総数 なお，カウンタ情報の総数が 0 の場合は 1 を指定してください。
CounterType_Num	種別 “OsCounterType” で HARDWARE を定義した総数
Isr_Num	割り込みサービス・ルーチン情報の総数 なお，割り込みサービス・ルーチン情報の総数が 0 の場合は 1 を指定してください。
Resource_Num	リソース情報の総数 なお，リソース情報の総数が 0 の場合は 1 を指定してください。
ScheduleTable_Num	スケジュール・テーブル情報の総数 なお，スケジュール・テーブル情報の総数が 0 の場合は 1 を指定してください。
ScheduleTableAutostart_Num	各アプリケーション・モードで自動起動するスケジュール・テーブルの総数
ScheduleTblExpPointOffset_Num	スケジュール・テーブル情報で定義した満了カウント値の総数
ScheduleTableAction_Num	スケジュール・テーブル情報で定義した満了処理の総数

キーワード	意味
Task_Num	タスク情報の総数 なお，タスク情報の総数が 0 の場合は 1 を指定してください。
TaskAutostart_Num	各アプリケーション・モードで自動起動するタスクの総数
SystemMaxExceptionCode	最大例外コード “OsSystemMaxExceptionCode” で定義した値

(2) SC3 用定数データ領域（.kernel_const）

以下に，スケーラビリティ・クラスが SC3 の場合における定数データ領域（.kernel_const）のサイズを見積もる際の計算式（単位：バイト）を示します。

なお，下記計算式における “align 4 （*x*）” は，数値 “*x*” を 4 バイト・アラインした結果を意味しています。

```
KERNEL_CONST =
    align 4 (
        1253
        + 44 * Alarm_Num
        + 12 * AlarmAutostart_Num
        + 12 * AppMode_Num
        + 148 * OsApplication_Num
        + 8 * TrustedFunctionName_Num
        + 24 * Counter_Num
        + 20 * CounterType_Num
        + 20 * Isr_Num
        + 8 * Resource_Num
        + 36 * ScheduleTable_Num
        + 12 * ScheduleTableAutostart_Num
        + 16 * ScheduleTblExpPointOffset_Num
        + 8 * ScheduleTableAction_Num
        + 24 * Task_Num
        + 2 * TaskAutostart_Num
        + 4 * (SystemMaxExceptionCode - 4095)
    )
```

上記計算式の各キーワードは，以下に示した意味を持ちます。

キーワード	意味
Alarm_Num	アラーム情報の総数
AlarmAutostart_Num	各アプリケーション・モードで自動起動するアラームの総数
AppMode_Num	アプリケーション・モード情報の総数 （OSDEFAULTAPPMODE に関する定義は除く）
OsApplication_Num	OS アプリケーション情報の総数
TrustedFunctionName_Num	OS アプリケーション情報で定義した Trusted 関数の総数 なお，Trusted 関数の総数が 0 の場合は 1 を指定してください。
Counter_Num	カウンタ情報の総数 なお，カウンタ情報の総数が 0 の場合は 1 を指定してください。
CounterType_Num	種別 “OsCounterType” で HARDWARE を定義した総数
Isr_Num	割り込みサービス・ルーチン情報の総数 なお，割り込みサービス・ルーチン情報の総数が 0 の場合は 1 を指定してください。
Resource_Num	リソース情報の総数 なお，リソース情報の総数が 0 の場合は 1 を指定してください。
ScheduleTable_Num	スケジュール・テーブル情報の総数 なお，スケジュール・テーブル情報の総数が 0 の場合は 1 を指定してください。
ScheduleTableAutostart_Num	各アプリケーション・モードで自動起動するスケジュール・テーブルの総数
ScheduleTblExpPointOffset_Num	スケジュール・テーブル情報で定義した満了カウント値の総数
ScheduleTableAction_Num	スケジュール・テーブル情報で定義した満了処理の総数

キーワード	意味
Task_Num	タスク情報の総数 なお，タスク情報の総数が 0 の場合は 1 を指定してください。
TaskAutostart_Num	各アプリケーション・モードで自動起動するタスクの総数
SystemMaxExceptionCode	最大例外コード “OsSystemMaxExceptionCode” の総数

C.5 定数データ領域（.kernel_identifier）

定数データ領域（.kernel_identifier）のメモリ使用量は，スケーラビリティ・クラスの種類により，以下のように異なります。

(1) SC1 用定数データ領域（.kernel_identifier）

以下に，スケーラビリティ・クラスが SC1 の場合における定数データ領域（.kernel_identifier）のサイズを見積もる際の計算式（単位：バイト）を示します。

```
KERNEL_IDENTIFIER =
      2 * Alarm_Num
    + 2 * Counter_Num
    + 4 * Event_Num
    + 2 * Isr_Num
    + 2 * Resource_Num
    + 2 * ScheduleTable_Num
    + 2 * Task_Num
```

上記計算式の各キーワードは，以下に示した意味を持ちます。

キーワード	意味
Alarm_Num	アラーム情報の総数
Counter_Num	カウンタ情報の総数
Event_Num	イベント情報の総数
Isr_Num	割り込みサービス・ルーチン情報の総数
Resource_Num	リソース情報の総数
ScheduleTable_Num	スケジュール・テーブル情報の総数
Task_Num	タスク情報の総数

(2) SC3 用定数データ領域（.kernel_identifier）

以下に，スケーラビリティ・クラスが SC3 の場合における定数データ領域（.kernel_identifier）のサイズを見積もる際の計算式（単位：バイト）を示します。

```
KERNEL_IDENTIFIER =
      2 * Alarm_Num
    + 2 * OsApplication_Num
    + 2 * TrustedFunctionName_Num
    + 2 * Counter_Num
    + 4 * Event_Num
    + 2 * Isr_Num
    + 2 * Resource_Num
    + 2 * ScheduleTable_Num
    + 2 * Task_Num
```

上記計算式の各キーワードは，以下に示した意味を持ちます。

キーワード	意味
Alarm_Num	アラーム情報の総数
OsApplication_Num	OS アプリケーション情報の総数
TrustedFunctionName_Num	OS アプリケーション情報で定義した Trusted 関数の総数
Counter_Num	カウンタ情報の総数
Event_Num	イベント情報の総数
Isr_Num	割り込みサービス・ルーチン情報の総数
Resource_Num	リソース情報の総数
ScheduleTable_Num	スケジュール・テーブル情報の総数
Task_Num	タスク情報の総数

C.6 変数データ領域（.kernel_work）

変数データ領域（.kernel_work）のメモリ使用量は，スケーラビリティ・クラスの種類により，以下のように異なります。

(1) SC1 用変数データ領域（.kernel_work）

以下に，スケーラビリティ・クラスが SC1 の場合における変数データ領域（.kernel_work）のサイズを見積もる際の計算式（単位：バイト）を示します。
なお，下記計算式における “align 4 （*x*）” は，数値 “*x*” を 4 バイト・アラインした結果を意味しています。

```
KERNEL_WORK =
    align 4 (
        128
        + 32 * Alarm_Num
        + 16 * Counter_Num
        + 12 * CounterType_Num
        + 12 * IsrCategory_Num
        + 8 * Resource_Num
        + 32 * ScheduleTable_Num
        + 24 * Task_Num
        + PriorityBuf_Ttl
    )
```

上記計算式の各キーワードは，以下に示した意味を持ちます。

キーワード	意味
Alarm_Num	アラーム情報の総数
Counter_Num	カウンタ情報の総数
CounterType_Num	種別 “OsCounterType” で HARDWARE を定義した総数
IsrCategory_Num	カテゴリ “OsIsrCategory” で 2 を定義した総数
Resource_Num	リソース情報の総数
ScheduleTable_Num	スケジュール・テーブル情報の総数
Task_Num	タスク情報の総数
PriorityBuf_Ttl	「優先度バッファ」で算出した値の総和

(2) SC3 用変数データ領域（.kernel_work）

以下に，スケーラビリティ・クラスが SC3 の場合における変数データ領域（.kernel_work）のサイズを見積もる際の計算式（単位：バイト）を示します。

なお，下記計算式における “align 4 （*x*）” は，数値 “*x*” を 4 バイト・アラインした結果を意味しています。

```
KERNEL_WORK =
    align 4 (
        148
        + 32 * Alarm_Num
        + 8 * OsApplication_Num
        + 16 * Counter_Num
        + 32 * CounterType_Num
        + 32 * IsrCategory_Num
        + 8 * Resource_Num
        + 32 * ScheduleTable_Num
        + 32 * Task_Num
        + PriorityBuf_Ttl
    )
```

上記計算式の各キーワードは，以下に示した意味を持ちます。

キーワード	意味
Alarm_Num	アラーム情報の総数
OsApplication_Num	OS アプリケーション情報の総数
Counter_Num	カウンタ情報の総数
CounterType_Num	種別 “OsCounterType” で HARDWARE を定義した総数
IsrCategory_Num	カテゴリ “OsIsrCategory” で 2 を定義した総数
Resource_Num	リソース情報の総数
ScheduleTable_Num	スケジュール・テーブル情報の総数
Task_Num	タスク情報の総数
PriorityBuf_Ttl	「優先度バッファ」で算出した値の総和

C.6.1 優先度バッファ

以下に，タスク情報で定義した優先度，およびリソース情報で定義したシーリング値に対応した 1 優先度当たりのバッファ・サイズを見積もる際の計算式（単位：バイト）を示します。

なお，下記計算式における “align 4 （*x*）” は，数値 “*x*” を 4 バイト・アラインした結果を意味しています。

```
PriorityBuf =
    align 4 (
        8
        + 2 * (TaskActivation_Pri + Resource - 1)
    )
```

上記計算式の各キーワードは，以下に示した意味を持ちます。

キーワード	意味
TaskActivation_Pri	最大起動要求数 “OsTaskActivation” の総和
Resource	リソース情報で定義したシーリング値と一致している場合は 1，一致していない場合は 0

備考 “*TaskActivation_Pri* + *Resource* - 1” の計算結果が 0 となる場合，1 優先度当たりのバッファ・サイズ PriorityBuf は 0 となります。

C.7 スタック領域（.kernel_stack）

スタック領域（.kernel_stack）のメモリ使用量は，スケーラビリティ・クラスの種類により，以下のように異なります。

(1) SC1 用スタック領域（.kernel_stack）

以下に，スケーラビリティ・クラスが SC1 の場合におけるスタック領域（.kernel_stack）のサイズを見積もる際の計算式（単位：バイト）を示します。

```
KERNEL_STACK =
      SystemStack
    + TaskStack_Ttl
```

上記計算式の各キーワードは，以下に示した意味を持ちます。

キーワード	意味
SystemStack	「SC1 用システム・スタック」で算出した値
ExtTaskStack_Ttl	「SC1 用タスク・スタック（拡張タスク）」で算出した値の総和

(2) SC3 用スタック領域（.kernel_stack）

以下に，スケーラビリティ・クラスが SC3 の場合におけるスタック領域（.kernel_stack）のサイズを見積もる際の計算式（単位：バイト）を示します。

```
KERNEL_STACK =
      SystemStack
    + OsApplicationStack_Ttl
    + ExtTaskStack_Ttl
```

上記計算式の各キーワードは，以下に示した意味を持ちます。

キーワード	意味
SystemStack	「SC3 用システム・スタック」で算出した値
OsApplicationStack_Ttl	「OS アプリケーション・スタック」で算出した値の総和
ExtTaskStack_Ttl	「SC3 用タスク・スタック（拡張タスク）」で算出した値の総和

C.7.1 システム・スタック

システム・スタックのメモリ使用量は，スケーラビリティ・クラスの種類により，以下のように異なります。

(1) SC1 用システム・スタック

以下に，スケーラビリティ・クラスが SC1 の場合におけるシステム・スタックのサイズを見積もる際の計算式（単位：バイト）を示します。

なお，下記計算式における “align 4 （*x*）” は数値 “*x*” を 4 バイト・アラインした結果を，“Max （*x*，*y*，*z*）” は数値 “*x*”，“*y*”，“*z*” を比較した場合の最大値を意味しています。

```
SystemStack =
    align4 (
        44
        + ErrorHookStack_Siz
        + ShutdownHokkStack_Siz
        + AlarmCallbackStack_Siz
        + Max (
            TaskStack_Ttl + MAX (
                8 + PostTaskHookStackSiz,
                8 + PreTaskHookStack_Siz
            ),
            20 + StartupHookStack_Siz,
            IdleHandlerStack_Siz
        )
        + MAX (
            92 + IsrStack_Max * Nest_Count,
            SystemServiceStack_Max
        )
    )
```

上記計算式の各キーワードは，以下に示した意味を持ちます。

キーワード	意味
ErrorHookStack_Siz	共通フック・ルーチン ErrorHook の処理内容に応じたスタック・サイズ
ShutdownHookStack_Siz	共通フック・ルーチン ShutdownHook の処理内容に応じたスタック・サイズ
AlarmCallbackStack_Siz	アラーム・コールバックの処理内容に応じたスタック・サイズ
TaskStack_Ttl	「(a) SC1 における基本タスクのスタック・サイズ」で算出した値の総和
PostTaskHookStack_Siz	共通フック・ルーチン PostTaskHook の処理内容に応じたスタック・サイズ
PreTaskHookStack_Siz	共通フック・ルーチン PreTaskHook の処理内容に応じたスタック・サイズ
StartupHookStack_Siz	共通フック・ルーチン StartupHook の処理内容に応じたスタック・サイズ
IdleHandlerStack_Siz	アイドル・ハンドラの処理内容に応じたスタック・サイズ
IsrStackMax	「(b) SC1 における割り込みサービス・ルーチン（カテゴリ 2）のスタック・サイズ」で算出した値の最大値
Nest_Count	割り込みサービス・ルーチン（カテゴリ 2）の最大ネスト数
SystemServiceStack_Max	164（拡張タスクから発行したシステム・サービスが必要とするスタック・サイズ）

(a) SC1 における基本タスクのスタック・サイズ

以下に，スケーラビリティ・クラスが SC1 の場合における基本タスクが使用するシステム・スタックのサイズを見積もる際の計算式（単位：バイト）を示します。
なお，下記計算式における “align 4 （*x*）” は数値 “*x*” を 4 バイト・アラインした結果を意味しています。

```
TaskStack =
    align4 (
        164
        + TaskStack_Siz
        + AlarmCallbackStack_Siz
    )
```

上記計算式の各キーワードは，以下に示した意味を持ちます。

キーワード	意味
TaskStack_Siz	基本タスクの処理内容に応じたスタックのサイズ
AlarmCallbackStack_Siz	アラーム・コールバックの処理内容に応じたスタックのサイズ

(b) SC1 における割り込みサービス・ルーチン（カテゴリ 2）のスタック・サイズ

以下に，スケーラビリティ・クラスが SC1 の場合における割り込みサービス・ルーチン・スタック（カテゴリ 2）のサイズを見積もる際の計算式（単位：バイト）を示します。
なお，下記計算式における “align 4 （x）” は数値 “x” を 4 バイト・アラインした結果を意味しています。

```
IsrStack =
    align4 (
        256
        + IsrStack_Siz
        + AlarmCallbackStack_Siz
    )
```

上記計算式の各キーワードは，以下に示した意味を持ちます。

キーワード	意味
IsrStack_Siz	割り込みサービス・ルーチン（カテゴリ 2）の処理内容に応じたスタックのサイズ
AlarmCallbackStack_Siz	アラーム・コールバックの処理内容に応じたスタックのサイズ

(2) SC3 用システム・スタック

以下に，スケーラビリティ・クラスが SC3 の場合におけるシステム・スタックのサイズを見積もる際の計算式（単位：バイト）を示します。

なお，下記計算式における “align 4 （*x*）” は数値 “*x*” を 4 バイト・アラインした結果を，“Max （*x*，*y*，*z*）” は数値 “*x*”，“*y*”，“*z*” を比較した場合の最大値を意味しています。

```
SystemStack =
    align4 (
        Max (
            Max (
                320,
                264 + ProtectionHookStack_Siz + ErrorHookStack_Siz,
                104 + StartupHookStack_Siz + ErrorHookStack_Siz
            )
            + Max (
                104,
                44 + ShutdownHookStack_Siz
            ),
            IdleHandlerStack_Siz
        )
    )
```

上記計算式の各キーワードは，以下に示した意味を持ちます。

キーワード	意味
ProtectionHookStack_Siz	共通フック・ルーチン ProtectionHook の処理内容に応じたスタック・サイズ
ErrorHookStack_Siz	共通フック・ルーチン ErrorHook の処理内容に応じたスタック・サイズ
StartupHookStack_Siz	共通フック・ルーチン StartupHook の処理内容に応じたスタック・サイズ
ShutdownHookStack_Siz	共通フック・ルーチン ShutdownHook の処理内容に応じたスタック・サイズ
IdleHandlerStack_Siz	アイドル・ハンドラの処理内容に応じたスタック・サイズ

C.7.2 OS アプリケーション・スタック

以下に，OS アプリケーション・スタックのサイズを見積もる際の計算式（単位：バイト）を示します。
なお，下記計算式における “align 4 （*x*）” は数値 “x” を 4 バイト・アラインした結果を，“Max （*x*，*y*，*z*）” は数値 “*x*”，“*y*”，“*z*” を比較した場合の最大値を意味しています。

```
OsApplicationStack =
    align 4 (
        32
        + AppShutdownHookStack_Siz
        + Max (
            TaskStack_Ttl + MAX (
                48 + PostTaskHookStack_Siz,
                48 + PreTaskHookStack_Siz,
            ),
            32 + AppStartupHookStack_Siz
        )
        + MAX (
            104 + ErrorHookStack_Siz,
            216 + AppErrorHookStack_Siz
        )
        + MAX (
            140 + IsrStack_Max * Nest_Count,
            SystemServiceStack_Max
        )
    )
```

上記計算式の各キーワードは，以下に示した意味を持ちます。

キーワード	意味
AppShutdownStackHook_Siz	固有フック・ルーチン Shutdown_*OsApplication* の処理内容に応じたスタック・サイズ
TaskStack_Ttl	「(a) SC3 における基本タスクのスタック・サイズ」で算出した値の総和
PostTaskHookStack_Siz	共通フック・ルーチン PostTaskHook の処理内容に応じたスタック・サイズ
PreTaskHookStack_Siz	共通フック・ルーチン PreTaskHook の処理内容に応じたスタック・サイズ
AppStartupHookStack_Siz	固有フック・ルーチン StartupHook_*OsApplication* の処理内容に応じたスタック・サイズ
ErrorHookStack_Siz	共通フック・ルーチン ErrorHook の処理内容に応じたスタック・サイズ
AppErrorHookStack_Siz	固有フック・ルーチン ErrorHook_*OsApplication* の処理内容に応じたスタック・サイズ
IsrStackMax	「(b) SC3 における割り込みサービス・ルーチン（カテゴリ 2）のスタック・サイズ」で算出した値の最大値
Nest_Count	割り込みサービス・ルーチン（カテゴリ 2）の最大ネスト数
SystemServiceStack_Max	164（拡張タスクから発行したシステム・サービスが必要とするスタック・サイズ）

(a) SC3 における基本タスクのスタック・サイズ

以下に，スケーラビリティ・クラスが SC3 の場合におけるタスク・スタック（基本タスク）のサイズを見積もる際の計算式（単位：バイト）を示します。
なお，下記計算式における “align 4 （*x*）” は数値 “*x*” を 4 バイト・アラインした結果を意味しています。

```
TaskStack =
    align4 (
        164
        + TaskStack_Siz
        + 24 * TrustedFunctionNest_Max
    )
```

上記計算式の各キーワードは，以下に示した意味を持ちます。

キーワード	意味
TaskStack_Siz	基本タスクの処理内容に応じたスタックのサイズ
TrustedFunctionNest_Max	Trusted 関数の最大ネスト数

(b) SC3 における割り込みサービス・ルーチン（カテゴリ 2）のスタック・サイズ

以下に，スケーラビリティ・クラスが SC3 の場合における割り込みサービス・ルーチン・スタック（カテゴリ 2）のサイズを見積もる際の計算式（単位：バイト）を示します。
なお，下記計算式における “align 4 （x）” は数値 “x” を 4 バイト・アラインした結果を意味しています。

```
IsrStack =
    align4 (
        304
        + IsrStack_Siz
        + 24 * TrustedFunctionNest_Max
    )
```

上記計算式の各キーワードは，以下に示した意味を持ちます。

キーワード	意味
IsrStack_Siz	割り込みサービス・ルーチン（カテゴリ 2）の処理内容に応じたスタックのサイズ
TrustedFunctionNest_Max	Trusted 関数の最大ネスト数 なお，Non-Trusted な OS アプリケーションであり，最大ネスト数が 0 の場合は 1 を指定してください。

C.7.3 タスク・スタック（拡張タスク）

タスク・スタック（拡張タスク）のメモリ使用量は，スケーラビリティ・クラスの種類により，以下のように異なります。

(1) SC1 用タスク・スタック（拡張タスク）

以下に，スケーラビリティ・クラスが SC1 の場合におけるタスク・スタック（拡張タスク）のサイズを見積もる際の計算式（単位：バイト）を示します。
なお，下記計算式における “align 4 （*x*）” は数値 “*x*” を 4 バイト・アラインした結果を意味しています。

```
ExtTaskStack =
    align4 (
        TaskStack_Siz
        + TaskContext_Siz
        + IsrContext_Siz
        + SystemServiceFrame_Siz
    )
```

上記計算式の各キーワードは，以下に示した意味を持ちます。

キーワード	意味
TaskStack_Siz	拡張タスクの処理内容に応じたスタックのサイズ
TaskContext_Siz	スケジューリング属性 “OsTaskSchedule” で NON を定義した場合は 0，FULL を定義した場合は 48
IsrContext_Siz	124（拡張タスク実行中に発生した割り込み処理に必要なスタック・サイズ）
SystemServiceFrame_Siz	8（拡張タスクから発行したシステム・サービスが必要とするスタック・サイズ）

(2) SC3 用タスク・スタック（拡張タスク）
SC3 用タスク・スタック（拡張タスク）のメモリ使用量は，所属している OS アプリケーションの信頼性が Trusted であるか Non-Trusted であるかにより，以下のように異なります。

(a) Trusted な OS アプリケーション
以下に，OS アプリケーションの信頼性が Trusted の場合における SC3 用タスク・スタック（拡張タスク）のサイズを見積もる際の計算式（単位：バイト）を示します。
なお，下記計算式における “align 4 （*x*）” は数値 “*x*” を 4 バイト・アラインした結果を意味しています。

```
ExtTaskStack =
    align4 (
        TaskStack_Siz
        + TaskContext_Siz
        + IsrContext_Siz
        + SystemServiceFrame_Siz
        + 24 * TrustedFunctionNest_Max
    )
```

上記計算式の各キーワードは，以下に示した意味を持ちます。

キーワード	意味
TaskStack_Siz	拡張タスクの処理内容に応じたスタックのサイズ
TaskContext_Siz	スケジューリング属性 “OsTaskSchedule” で NON を定義した場合は 0，FULL を定義した場合は 48
IsrContext_Siz	128（拡張タスク実行中に発生した割り込み処理に必要なスタック・サイズ）
SystemServiceFrame_Siz	12（拡張タスクから発行したシステム・サービスが必要とするスタック・サイズ）
TrustedFunctionNest_Max	Trusted 関数の最大ネスト数

(b) Non-Trusted な OS アプリケーション
以下に，OS アプリケーションの信頼性が Non-Trusted の場合における SC3 用タスク・スタック（拡張タスク）のサイズを見積もる際の計算式（単位：バイト）を示します。
なお，下記計算式における “align 4 （x）” は数値 “x” を 4 バイト・アラインした結果を意味しています。

```
ExtTaskStack =
    align4 (
        140
        + TaskStack_Siz
        + 24 * TrustedFunctionNest_Max
    )
```

上記計算式の各キーワードは，以下に示した意味を持ちます。

キーワード	意味
TaskStack_Siz	拡張タスクの処理内容に応じたスタックのサイズ
TrustedFunctionNest_Max	Trusted 関数の最大ネスト数

C.8 割り込みハンドラ・アドレス・テーブル（.kernel_address）

以下に，割り込みハンドラ・アドレス・テーブル（.kernel_*address*）のサイズを見積もる際の計算式（単位：バイト）を示します。

```
IntHdrAdrTbl =
    4 * (SystemMaxExceptionCode - 4095)
```

上記計算式の各キーワードは，以下に示した意味を持ちます。

キーワード	意味
SystemMaxExceptionCode	最大例外コード “OsSystemMaxExceptionCode” で定義した値

改訂記録

Rev.	発行日	改定内容	
		ページ	ポイント
1.00	2013.11.01	-	初版発行
1.01	2014.09.30	10	［1.4　実行環境］ “(1) デバイス ” から以下の説明文章 “ 備考 1.” を削除。 スケーラビリティ・クラス SC3 に対応した RV850 では，対象デバイスが RH850 ファミリ（G3M コア）に限定されます。
		10	［1.4　実行環境］ “(2) 周辺コントローラ ” - “ 備考 3.” の説明文章を以下のように変更。 … のスケーラビリティ・クラス SC3 で規定された … 　→ … のスケーラビリティ・クラス SC3 （G3M コア限定）で規定された …
		10	［1.4　実行環境］ “ 表 1.1　RV850 が占有している OS 予約資源 ” - “ マシン・コンフィギュレーション（MCFG0）の SPID ビット ” の説明文章を以下のように変更。 SC3 　→ SC3（G3M コア限定）
		10	［1.4　実行環境］ “ 表 1.1　RV850 が占有している OS 予約資源 ” - “ 割り込み優先度マスク（PMR）” の説明文章を以下のように変更。 SC3 　→ SC1 ／ SC3
		10	［1.4　実行環境］ “ 表 1.1　RV850 が占有している OS 予約資源 ” - “ 割り込み機能の設定（INTCFG）” の説明文章を以下のように変更。 SC3 　→ SC1 ／ SC3
		10	［1.4　実行環境］ “ 表 1.1　RV850 が占有している OS 予約資源 ” に於ける 　割り込みコントローラ（INTC1）の … 　割り込みコントローラ（INTC2）の … を 1 行に統合すると伴に，記述内容を以下のように変更。 割り込みコントローラ（INTC1）の EI レベル割り込み制御レジスタ（EIC*n*）の EITB*n* ビット 割り込みコントローラ（INTC2）の EI レベル割り込み制御レジスタ（EIC*n*）の EITB*n* ビット 　→ 例外コード “OsIsrExceptionCode” ／例外コード “OsCounterExceptionCode” で定義された割り込みに対応した割り込みコントローラ（INTC1，INTC2）の EI レベル割り込み制御レジスタ（EIC*n*）の EITB*n* ビット，および EIP3*n*-0*n* ビット
		12	［1.5.1　オブジェクト・ファイル提供形式］ “(9) <rx_root>￥{SC1，SC3}￥lib￥< ターゲット名 >￥r32” の説明文章を以下のように変更。

Rev.	発行日	改定内容	
		ページ	ポイント
1.01	2014.09.30		libecc2extsc3.a： … , FPU 対応なし libecc2extsc3_fpu.a： … , FPU 対応あり → libecc2extsc3.a： … , FPU 対応なし，G3M 限定 libecc2extsc3_fpu.a： … , FPU 対応あり，G3M 限定
		12	［1.5.1　オブジェクト・ファイル提供形式］ “(9) <rx_root>￥{SC1，SC3}￥lib￥< ターゲット名 >￥r32” に以下の説明文章を追加。 libecc2extsc3_g3k.a： ECC2，拡張ステータス，SC3，FPU 対応なし，G3K 限定
		14	［1.5.2　ソース・ファイル提供形式］ “(10)” の箇条書きタイトルを以下のように変更。 <rx_root>￥src￥os￥trace → <rx_root>￥｛SC1，SC3｝￥src￥os￥trace
		14	［1.5.2　ソース・ファイル提供形式］ “(10) <rx_root>￥{SC1，SC3}￥src￥os￥trace” の説明文章を以下のように変更。 トレース・ルーチンのソース・ファイル（スケーラビリティ・クラス SC3 用）が … → トレース・ルーチンのソース・ファイルが …
		15	［2.2　ユーザ・オウン・コーディング部の記述］ 本節の説明文章を以下のように変更。 … として切り出し，サンプル・ソース・ファイル（excent.850）を … → … として切り出し，サンプル・ソース・ファイル（direct_vector.850, excent.850）を …
		15	［2.2　ユーザ・オウン・コーディング部の記述］ “(1)” の箇条書きタイトルを以下のように変更。 FE レベル例外エントリ処理 → エントリ処理（直接分岐方式の例外ベクタ）
		15	［2.2　ユーザ・オウン・コーディング部の記述］ “(1) エントリ処理（直接分岐方式の例外ベクタ）” の説明文章を以下のように変更。 FE レベル例外が発生した際にデバイスが強制的に制御を移すハンドラ・アドレスに該当処理（発生した FE レベル例外の種類に対応した処理：ブート処理，保護例外に対応した処理）への分岐処理を割り付けるために切り出されたエントリ処理専用ルーチンです。 → リセット（RESET），FE レベル割り込み（FENMI，TRAP など），EI レベル割り込み（例外コード “OsIsrExceptionCode” ／例外コード “OsCounterExceptionCode” で未定義）などが発生した際，対応する処理（ブート処理，例外／割り込み安全対策処理など）への分岐処理を割り付けるために切り出されたエントリ処理専用ルーチンです。
		15	［2.2　ユーザ・オウン・コーディング部の記述］ “(2)” の箇条書きタイトルを以下のように変更。

Rev.	発行日	改定内容	
		ページ	ポイント
1.01	2014.09.30		デフォルト割り込みサービス・ルーチン → 例外／割り込み安全対策処理
		15	［2.2　ユーザ・オウン・コーディング部の記述］ “(2) 例外／割り込み安全対策処理 ” の説明文章を以下のように変更。 デフォルト割り込みサービス・ルーチンは，例外コード “OsIsrExceptionCode” ／例外コード “OsCounterExceptionCode” で未定義の EI レベル割り込みが発生した際に呼び出される割り込み処理専用ルーチンです。 → FE レベル割り込み（FENMI，TRAP など），EI レベル割り込み（例外コード “OsIsrExceptionCode” ／例外コード “OsCounterExceptionCode” で未定義）などが発生した際，エントリ処理から呼び出される安全対策処理専用ルーチンです。
		16	［2.2.1　ユーザ・オウン・ライブラリの生成］ 本項の説明文章を以下のように変更。 … で作成されたソース・ファイルに対して … → … で作成されたソース・ファイル（例外／割り込み安全対策処理：excent.850）に対して …
		16	［2.3　処理プログラムの記述］ “(3) 割り込みサービス・ルーチン ” の説明文章を以下のように変更。 割り込みが発生した際に … → EI レベル割り込み（例外コード “OsIsrExceptionCode” ／例外コード “OsCounterExceptionCode” で定義）が発生した際に …
		17	［2.5　リンカ・ディレクティブ・ファイルの記述］ 本節の説明文章を以下のように変更。 リンク・エディタが行うアドレス割り付けをユーザが固定化するための情報（セクション情報，アドレス情報など）を保持したリンカ・ディレクティブ・ファイルを記述します。 → リンク・エディタが行うメモリ配置を固定化するためのリンカ・ディレクティブ・ファイルを記述します。
		17	［2.5　リンカ・ディレクティブ・ファイルの記述］ “ 表 2.1　規定された割り付け先 ” に以下の説明文章を追加。 .kernel_*address*　A　ROM　割り込みハンドラ・アドレス・テーブル
		17	［2.5　リンカ・ディレクティブ・ファイルの記述］ 本節に以下の説明文章を “ 備考 3.” として追加。 割り込みハンドラ・アドレス・テーブル .kernel_*addresss* の割り付け先については，コンフィギュレータが ENTRY ファイルに出力しているため，ユーザがリンカ・ディレクティブ・ファイルに .kernel_*address* に関する定義を記述する必要はありません。 なお，.kernel_*address* における “*address*” は，「ベース・アドレス “OsInterruptBaseAddress” + 4 * 割り込みチャネル番号」で算出された値となります。
		18	［2.6　ロード・モジュールの生成］ 本節に以下の説明文章を “ 備考 5.” として追加。 RV850 では，共通フック・ルーチンのライブラリ化を禁止しています。

Rev.	発行日	改定内容	
		ページ	ポイント
1.01	2014.09.30	18	[2.6　ロード・モジュールの生成] “ 備考 7.” の説明文章を以下のように変更。 … のカーネル・ライブラリ libecc2stdsc1_fpu.a，libecc2extsc1_fpu.a, libecc2extsc3_fpu.a が … 　→ … のカーネル・ライブラリ libecc2extsc1_fpu.a，libecc2extsc3_fpu.a が …
		20	[3.1.4　タスクでの処理] “(2) FPSR の退避／復帰 ” - “ 備考 ” の説明文章を以下のように変更。 … のカーネル・ライブラリ libecc2stdsc1_fpu.a，libecc2extsc1_fpu.a, libecc2extsc3_fpu.a を … 　→ … のカーネル・ライブラリ libecc2extsc1_fpu.a，libecc2extsc3_fpu.a を …
		20	[3.1.4　タスクでの処理] “(3) スタックの切り替え ” - “ 備考 ” の説明文章を以下のように変更。 また，スタック・サイズ “OsAppStackSize” で定義されたシステム・スタックへの切り替えは，… 　→ また，スタック・サイズ “OsAppStackSize” で定義された OS アプリケーション・スタックへの切り替えは，…
		23	[4.1　概　　要] 本節の説明文章を以下のように変更。 … のうち，例外コード “OsIsrExceptionCode” ／例外コード “OsCounterExceptionCode” で定義された EI レベル割り込みに対応した割り込みサービス・ルーチンのみを管理対象としています。 したがって，例外コード “OsIsrExceptionCode” ／例外コード “OsCounterExceptionCode” で未定義の割り込みに対応した処理（ブート処理，例外／割り込み安全対策処理など）は，RV850 の管理外の処理として動作します。 　→ … のうち，EI レベル割り込み（例外コード “OsIsrExceptionCode” ／例外コード “OsCounterExceptionCode” で定義）に対応した処理のみを割り込みサービス・ルーチンとして扱います。 したがって，リセット （RESET），FE レベル割り込み （FENMI，TRAP），EI レベル割り込み （例外コード “OsIsrExceptionCode” ／例外コード “OsCounterExceptionCode” で未定義）などに対応した処理 （ブート処理，例外／割り込み安全対策処理など）は，割り込みサービス・ルーチンとして扱われず，特別な扱いとなります。
		23	[4.2　ブート処理] “(4)” の箇条書きタイトルを以下のように変更。 … に関する初期化 　→ … に関する初期化（G3M コア限定）
		25	[4.3　割り込みサービス・ルーチン] 本節の説明文章を以下のように変更。 割り込みが発生した際に … 　→ EI レベル割り込み（例外コード “OsIsrExceptionCode” ／例外コード “OsCounterExceptionCode” で定義）が発生した際に …

Rev.	発行日	改定内容	
		ページ	ポイント
1.01	2014.09.30	26	[4.3.1　割り込みサービス・ルーチンでの処理] “(1) カテゴリ 1” - “(d) 割り込みの受け付け ” に以下の説明文章を追加。 このため，該当割り込みの発生時にデバイスによるプログラム・ステータス・ワード（PSW）の ID ビット操作が行われ，割り込みの受け付けが禁止されています。
		26	[4.3.1　割り込みサービス・ルーチンでの処理] “(1) カテゴリ 1” - “(e) システム・サービスの発行 ” に以下の説明文章を “ 備考 1.” として追加。 RV850 では，割り込みサービス・ルーチンから発行不可なシステム・サービスが発行された場合の動作を保証していません。
		27	[4.3.1　割り込みサービス・ルーチンでの処理] “(2) カテゴリ 2” - “(b) FPSR の退避／復帰 ” - “ 備考 ” の説明文章を以下のように変更。 … のカーネル・ライブラリ libecc2stdsc1_fpu.a，libecc2extsc1_fpu.a, libecc2extsc3_fpu.a を … 　→ … のカーネル・ライブラリ libecc2extsc1_fpu.a，libecc2extsc3_fpu.a を …
		27	[4.3.1　割り込みサービス・ルーチンでの処理] “(2) カテゴリ 2” - “(c) スタックの切り替え ” の説明文章を以下のように変更。 … で定義されたシステム・スタックへと切り替えます。 　→ … で定義されたシステム・スタック，またはスタック・サイズ “OsAppStackSize” で定義された OS アプリケーション・スタックへと切り替えます。
		27	[4.3.3　割り込みサービス・ルーチンの終了] “(2) カテゴリ 2，スケーラビリティ・クラス SC1 の場合 ” の説明文章を以下のように変更。 割り込みの発生した処理プログラムに制御を戻す 　→ スケジューラの起動
		27	[4.3.3　割り込みサービス・ルーチンの終了] “(3) カテゴリ 2，スケーラビリティ・クラス SC3 の場合 ” の説明文章を以下のように変更。 … を許可（対応する EnableAllInterrupts，ResumeAllInterrupts, ResumeOSInterrupts と同等の処理） 　→ … を許可（対応する EnableAllInterrupts，ResumeAllInterrupts と同等の処理）
		28	[4.3.3　割り込みサービス・ルーチンの終了] “(3) カテゴリ 2，スケーラビリティ・クラス SC3 の場合 ” の説明文章を以下のように変更。 … していた場合，E_OS_RESOURCE（0x15）を … 　→ … していた場合，E_OS_RESOURCE（0x6）を …
		28	[4.3.3　割り込みサービス・ルーチンの終了] “(3) カテゴリ 2，スケーラビリティ・クラス SC3 の場合 ” の説明文章を以下のように変更。

Rev.	発行日	改定内容	
		ページ	ポイント
1.01	2014.09.30		割り込みの発生した処理プログラムに制御を移す 　→ スケジューラを起動
		28	[4.3.3　割り込みサービス・ルーチンの終了] “ 備考 2.” に於ける説明文章を以下のように変更。 … のカーネル・ライブラリ libecc2stdsc1_fpu.a，libecc2extsc1_fpu.a, libecc2extsc3_fpu.a を … 　→ … のカーネル・ライブラリ libecc2extsc1_fpu.a，libecc2extsc3_fpu.a を …
		29	[4.5.1　エントリ処理（直接分岐方式の例外ベクタ）] 本項の説明文章を以下のように変更。 FE レベル例外が発生した際にデバイスが強制的に制御を移すハンドラ・アドレスに該当処理（発生した FE レベル例外の種類に対応した処理：ブート処理，保護例外に対応した処理など）への分岐処理を割り付けるために切り出されたエントリ処理専用ルーチンです。 以下に，FE レベル例外エントリ処理をアセンブリ言語で記述する際の基本型を示します。 　→ リセット（RESET），FE レベル割り込み（FENMI，TRAP など），EI レベル割り込み（例外コード “OsIsrExceptionCode” ／例外コード “OsCounterExceptionCode” で未定義）などが発生した際，対応する処理（ブート処理，例外／割り込み安全対策処理など）への分岐処理を割り付けるために切り出されたエントリ処理専用ルーチンです。 以下に，エントリ処理をアセンブリ言語で記述する際の基本型を示します。

		29	[4.5.1　エントリ処理（直接分岐方式の例外ベクタ）]
			“【SC1，RBASE ／ EBASE：0x0 の場合】” の基本型を以下と差し替え。
			
			`    .globl  _entry0000`
			`    .globl  _entry0010`
			`    ……`
			`    .globl  _entry0100`
			`    ……`
			
			`    .org    0x00000000`
			`_entry0000:`
			`    jr      _boot`
			
			`    .org    0x00000010`
			`_entry0010:`
			`    jr      __kernel_e_IllegalExcEntry`
			
			`    ……`
			
			`    .org    0x00000100`
			`_entry0100:`
			`    jr      __kernel_e_IllegalExcEntry`
			
			`    ……`

		30	[4.5.1　エントリ処理（直接分岐方式の例外ベクタ）]
			“【SC3，RBASE ／ EBASE：0x0 の場合】” の基本型を以下と差し替え。
			
			`    .globl  _entry0000`
			`    .globl  _entry0010`
			`    .globl  _entry0020`

Rev.	発行日	改定内容	
		ページ	ポイント
1.01	2014.09.30		…… .globl _entry0090 .globl _entry00A0 .globl _entry00B0 …… .globl _entry0100 …… .org 0x00000000 _entry0000: jr _boot .org 0x00000010 _entry0010: jr __kernel_e_ProtectEntry .org 0x00000020 _entry0020: jr __kernel_e_IllegalExcEntry …… .org 0x00000090 _entry0090: jr __kernel_e_ProtectEntry .org 0x000000A0 _entry00A0: jr __kernel_e_ProtectEntry .org 0x000000B0 _entry00B0: jr __kernel_e_IllegalExcEntry …… .org 0x00000100 _entry0100: jr __kernel_e_IllegalExcEntry ……
		30	[4.5.1 エントリ処理（直接分岐方式の例外ベクタ）] “ 備考 1.” の説明文章を以下のように変更。 EI レベル割り込みに関するエントリ処理は，ENTRY ファイルに出力されます。 → 例外コード “OsIsrExceptionCode” ／例外コード “OsCounterExceptionCode” で定義された EI レベル割り込みに関するエントリ処理については，コンフィギュレータが ENTRY ファイルに出力しているため，ユーザが該当エントリ処理を記述する必要はありません。
		31	[4.5.2 例外／割り込み安全対策処理] 本項の説明文章を以下のように変更。 デフォルト割り込みサービス・ルーチンは，例外コード “OsIsrExceptionCode” ／例外コード “OsCounterExceptionCode” で未定義の EI レベル割り込みが発生した際に呼び出される割り込み処理専用ルーチンです。 以下に，デフォルト割り込みサービス・ルーチンを C 言語で記述する際の基本

Rev.	発行日	改定内容	
		ページ	ポイント
1.01	2014.09.30		型を示します。 → FE レベル割り込み（FENMI，TRAP など），EI レベル割り込み（例外コード “OsIsrExceptionCode” ／例外コード “OsCounterExceptionCode” で未定義）などが発生した際，エントリ処理から呼び出される安全対策処理専用ルーチンです。 以下に，例外／割り込み安全対策処理を C 言語で記述する際の基本型を示します。
		31	［4.5.2　例外／割り込み安全対策処理］ “ 備考 ” の説明文章を以下のように変更。 RV850 ではデフォルト割り込みサービス・ルーチンが用意されています。 したがって，_kernel_e_IllegalExcEntery の記述が行われていない場合には，デフォルト割り込みサービス・ルーチン（引数 *Error* に E_OS_SYS_ILLEGAL_EXCEPTION を指定した ShutdownOS の発行処理）が呼び出されます。 → RV850 では，デフォルトの例外／割り込み安全対策処理を用意しています。 このため，_kernel_e_IllegalExcEntery の記述が行われていない場合であっても，エントリ処理に例外／割り込み安全対策処理への分岐処理が割り付けられていた際，および SYSCALL 命令の発行時に不正な値が指定された際には，デフォルトの例外／割り込み安全対策処理（引数 *Error* に E_OS_SYS_ILLEGAL_EXCEPTION を指定した ShutdownOS の発行処理）が呼び出されます。
		32	［5.1.1　スケジューラ・リソース］ 本項の説明文章を以下のように変更。 RV850 では，処理プログラムからスケジューラの起動を動的に禁止／許可するための手段として，スケジューラ・リソース（識別子：RES_SCHEDULER）を提供しています。 → OSEK/VDX 仕様では，処理プログラムからスケジューラの起動を動的に禁止／許可するための手段として，識別子：RES_SCHEDULER を持つリソースを規定しています。RV850 においてスケジューラ・リソース RES_SCHEDULER を有効にする方法については，「5.2　リソースの生成」を参照してください。
		35	［8.1　概　　要］ 本節の説明文章を以下のように変更。 RV850 では，カウンタの上位機能（カウント値の変化に対応するための機能）として “ アラーム管理機能 ” を提供しています。 → RV850 では，カウンタのもつカウント値の変化に同期した処理を行うための機能として “ アラーム管理機能 ” を提供しています。
		37	［9.1　概　　要］ 本節の説明文章を以下のように変更。 RV850 では，カウンタの上位機能（カウント値の変化に対応するための機能）として “ スケジュール・テーブル管理機能 ” を提供しています。 → RV850 では，カウンタのもつカウント値の変化に同期した処理を行うための機能として “ スケジュール・テーブル管理機能 ” を提供しています。
		39	［10.1.3　メモリ・アクセス保護機能］ 本項の説明文章を以下のように変更。 RV850 がメモリ・アクセス保護機能で監視対象としているメモリ領域を以下に示します。

Rev.	発行日	改定内容	
		ページ	ポイント
1.01	2014.09.30		なお，監視対象となるメモリ領域（アクセス許可なメモリ領域）は，メモリ領域識別子 “OsAppMemoryAreaNameRef” で定義し，該当項目で未定義のメモリ領域はアクセス禁止なメモリ領域として扱われます。 → 以下に，RV850 がメモリ・アクセス保護機能で監視対象としているメモリ領域を示します。 なお，監視対象となるメモリ領域についてはメモリ領域識別子 “OsSystemMemoryArea” で，メモリ領域の種類（OS アプリケーション固有，システム共通）についてはメモリ領域識別子 “OsAppMemoryAreaNameRef”，メモリ領域識別子 “OsMemoryAreaNameRef” で定義し，未定義のメモリ領域はアクセス禁止なメモリ領域として扱われます。
		39	［10.1.3　メモリ・アクセス保護機能］ 本項に以下の説明文章を “ 備考 ” として追加。 スタック領域については，メモリ領域識別子 “OsSystemMemoryArea”，メモリ領域識別子 “OsAppMemoryAreaNameRef”，メモリ領域識別子 “OsMemoryAreaNameRef” での定義が不要です。
		40	［10.2.1　Trusted 関数の処理］ “(2) FPSR の退避／復帰 ” の説明文章を以下のように変更。 RV850 では，Trusted 関数を CallTrustedFunction を発行した処理プログラムの延長線として位置付けているため，浮動小数点設定／状態レジスタ（FPSR）の退避処理／復帰処理を行いません。 したがって，FPSR の内容を明示的に変更する場合，“FPSR の退避処理／復帰処理 ” を記述する必要があります。 → FPSR“OsAppDefaultFPSRValue”，または FPSR のデフォルト値 “OsDefaultFPSRValue” で定義された値へと変更します。 したがって，FPSR の退避／復帰 “OsSaveFpuReg” で TRUE を定義している場合，“FPSR の退避処理／復帰処理 ” は，記述する必要がありません。
		40	［10.2.1　Trusted 関数の処理］ “(2) FPSR の退避／復帰 ” に以下の説明文章を “ 備考 ” として追加。 “FPSR の退避処理／復帰処理 ” は，“FPU 対応あり ” のカーネル・ライブラリ libecc2extsc1_fpu.a，libecc2extsc3_fpu.a をリンクした場合に限り行われます。
		41	［10.3　固有フック・ルーチン］ “(1) StartupHook_*OsApplication*" に以下の説明文章を “ 備考 2.” として追加。 StartupHook_OsApplication“OsAppStartupHook” で TRUE の指定が行われてる OS アプリケーション情報が複数存在する場合，StartupHook_*OsApplication* の呼び出しは，CF ファイルでの記述順に行われます。
		41	［10.3　固有フック・ルーチン］ “(2) ShutdownHook_*OsApplication*” - “ 備考 2.” の説明文章を以下のように変更。 … が RV850 の提供しているデフォルト割り込みサービス・ルーチンであった場合，… → … がデフォルトの例外／割り込み安全対策処理であった場合，…
		41	［10.3　固有フック・ルーチン］ “(2) ShutdownHook_*OsApplication*” に以下の説明文章を “ 備考 3.” として追加。 引数 *Fatalerror* が E_OS_SYS_ILLEGAL_EXCEPTION の際には，本フック・ルーチン内で OSIllegalException_SystemRegister_ExcCode を発行することによりレジスタ値（EIIC，または FEIC）を，

Rev.	発行日	改定内容	
		ページ	ポイント
1.01	2014.09.30		OSIllegalException_SystemRegister_ExcPC を発行することによりレジスタ値（EIPC，または FEPC）を獲得することができます。
		41	［10.3　固有フック・ルーチン］ “(2) ShutdownHook_*OsApplication*” に以下の説明文章を “ 備考 4.” として追加。 ShutdownHook_OsApplication“OsAppStartupHook” で TRUE の指定が行われている OS アプリケーション情報が複数存在する場合， ShutdownHook_*OsApplication* の呼び出しは，CF ファイルでの記述順に行われます。
		42	［10.3　固有フック・ルーチン］ “(3) ErrorHook_*OsApplication*” - “ 備考 4.” の説明文章を以下のように変更。 … した際，ErrorHook_*OsApplication* の再呼び出しは行われません。 → … した際，ErrorHook，および ErrorHook_*OsApplication* の再呼び出しは行われません。
		42	［10.3.1　固有フック・ルーチンでの処理］ “(4) 割り込みの受け付け ” - “ 備考 1.” の説明文章を以下のように変更。 RV850 では，固有フック・ルーチンでカテゴリ 2 割り込みの受け付けを許可状態へと変更することが禁止されています。 → 固有フック・ルーチンでカテゴリ 2 割り込みの受け付けを明示的に操作することは禁止されています。
		42	［10.3.1　固有フック・ルーチンでの処理］ “(4) 割り込みの受け付け ” に以下の説明文章を “ 備考 2.” として追加。 RV850 では，ShutdownHook_*OsApplication* に制御を移す際，割り込みの受け付け禁止処理として，PMR の PM*n* ビットに対する操作の他，PSW の ID ビットに対する操作も行っています。
		44	［11.2　共通フック・ルーチン］ “(2) ShutdownHook” - “ 備考 1.” の説明文章を以下のように変更。 … が RV850 の提供しているデフォルト割り込みサービス・ルーチンであった場合，… → … がデフォルトの例外／割り込み安全対策処理であった場合，…
		44	［11.2　共通フック・ルーチン］ “(2) ShutdownHook” に以下の説明文章を “ 備考 2.” として追加。 引数 *Fatalerror* が E_OS_SYS_ILLEGAL_EXCEPTION の際には，本フック・ルーチン内で OSIllegalException_SystemRegister_ExcCode を発行することによりレジスタ値（EIIC，または FEIC）を， OSIllegalException_SystemRegister_ExcPC を発行することによりレジスタ値（EIPC，または FEPC）を獲得することができます。
		45	［11.2　共通フック・ルーチン］ “(5) ErrorHook” - “ 備考 2.” の説明文章を以下のように変更。 … した際，ErrorHook の再呼び出しは行われません。 → … した際，ErrorHook，および ErrorHook_*OsApplication* の再呼び出しは行われません。

Rev.	発行日	改定内容	
		ページ	ポイント
1.01	2014.09.30	45	［11.2　共通フック・ルーチン］ “(6) ProtectionHook” - “ 備考 2.” の説明文章を以下のように変更。 引数 *adr* で指定された領域には，保護違反を検出した時点における FE レベル例外受け付け時の状態退避レジスタ（FEPC）の値が設定されます。 → 引数 *adr* で指定された領域には，引数 *Fatalerror* が E_OS_STACKFAULT の場合は “0” が，E_OS_PROTECTION_MEMORY の場合は “ 保護違反を検出した時点における FE レベル割り込み受け付け時の状態退避レジスタ（FEPC）の値 ” が設定されます。
		45	［11.2　共通フック・ルーチン］ “(6) ProtectionHook” - “ 備考 3.” - “【PRO_TERMINATETASKISR（0x1)】” の説明文章を以下のように変更。 タスクが保護違反を検出した場合，該当タスクを SUSPENDED 状態へと遷移させたのち，スケジューラを起動 タスク以外の処理プログラムが保護違反を検出した場合，該当処理プログラムを呼び出したタスクに制御を戻し，タスクの処理を再開 → 保護違反を行った処理プログラムの種類により，処理内容は以下のように異なります。 【タスクの場合】 - SUSPENDED 状態へと遷移 - リソースを獲得していた際には，該当リソースを解放 - DisableAllInterrupts を発行していた際には，EnableAllInterrupts を発行 - SuspendAllInterrupts を発行していた際には，ResumeAllInterrupts を発行 - SuspendOSInterrupts を発行していた際には，ResumeOSInterrupts を発行 - スケジューラを起動 【割り込みサービス・ルーチンの場合】 - リソースを獲得していた際には，該当リソースを解放 - DisableAllInterrupts を発行していた際には，EnableAllInterrupts を発行 - SuspendAllInterrupts を発行していた際には，ResumeAllInterrupts を発行 - SuspendOSInterrupts を発行していた際には，ResumeOSInterrupts を発行 - スケジューラを起動 【上記以外の場合】 - PRO_TERMINATEAPPL と同様の処理
		46	［11.2　共通フック・ルーチン］ “(6) ProtectionHook” - “ 備考 3.” - “【PRO_TERMINATEAPPL（0x2)】” の説明文章を以下のように変更。 … に対し，TerminateApplication を発行（再起動オプション：NO_RESTART） → … に対し，TerminateApplication（再起動オプション：NO_RESTART）相当の処理を実行 なお，APPLICATION_ACCESSIBLE 状態の OS アプリケーションが存在しない際には，ShutdownOS（引き継ぎデータ：*Fatalerror*）を発行

Rev.	発行日	改定内容	
		ページ	ポイント
1.01	2014.09.30	46	［11.2　共通フック・ルーチン］ “(6) ProtectionHook” - “ 備考 3.” - “【PRO_SHUTDOWN（0x4）】” の説明文章を以下のように変更。 保護違反を行った処理プログラムが所属している OS アプリケーションに対し，ShutdownOS を発行（引き継ぎデータ：*Fatalerror*） 　→ ShutdownOS（引き継ぎデータ：*Fatalerror*）を発行
		46	［11.2　共通フック・ルーチン］ “(6) ProtectionHook” - “ 備考 3.” - “【PRO_TERMINATEAPPL_RESTART（0x12）】” の説明文章を以下のように変更。 … に対し，TerminateApplication を発行（再起動オプション：RESTART） 　→ … に対し，TerminateApplication（再起動オプション：RESTART）相当の処理を実行 なお，APPLICATION_ACCESSIBLE 状態の OS アプリケーションが存在しない際，およびタスク識別子 “OsRestartTask” が未定義の際には，ShutdownOS（引き継ぎデータ：*Fatalerror*）を発行
		46	［11.2　共通フック・ルーチン］ “(6) ProtectionHook” - “ 備考 3.” に以下の説明文章を追加。 【その他】 ShutdownOS（引き継ぎデータ：*Fatalerror*）を発行
		46	［11.2.1　共通フック・ルーチンでの処理］ “(1) レジスタの退避／復帰 ” に以下の説明文章を “ 備考 ” として追加。 RV850 では，ProtectionHook に制御を移す際，FE レベル例外用作業レジスタ（FEWR）を退避／復帰することなく使用します。 したがって，ProtectionHook に制御が移った際には，FEWR の値が不定値となります。
		46	［11.2.1　共通フック・ルーチンでの処理］ “(4) 割り込みの受け付け ” - “ 備考 2.” の説明文章を以下のように変更。 RV850 では，ProtectionHook に制御を移す際，… 　→ RV850 では，ShutdownHook，ProtectionHook に制御を移す際，…
		48	［12.1　概　　要］ “(2) プリエンプティブ属性 ” の説明文章を以下のように変更。 タスクの状態遷移を引き起こす可能性があるシステム・サービスの発行 カテゴリ 2 の割り込みサービス・ルーチンからの復帰命令の発行 Schedule の発行 　→ ActivateTask の発行 TerminateTask の発行 ChainTask の発行 Schedule の発行 ReleaseResource の発行 SetEvent の発行 WaitEvent の発行 TerminateApplication の発行 保護例外（システム・エラー例外，メモリ・プロテクション例外，特権命令例外）の発生 カテゴリ 2 の割り込みサービス・ルーチンからの復帰命令の発行

Rev.	発行日	改定内容	
		ページ	ポイント
1.01	2014.09.30		アラーム／スケジュール・テーブルの満了
		50	［13.1　概　　要］ 本節に以下の説明文章を “ 備考 ” として追加。 RV850 では，StartOS でフック・ルーチンの呼び出しが行われる以前にユーザが EI レベル割り込みマスク・レジスタ（IMR*m*）に対する操作を行い，例外コード “OsIsrExceptionCode” ／例外コード “OsCounterExceptionCode” で定義された EI レベル割り込みの受け付けを許可した場合の動作を保証していません。
		51	［13.2　エントリ処理（直接分岐方式の例外ベクタ）］ 本節の説明文章を以下のように変更。 FE レベル例外が発生した際にデバイスが強制的に制御を移すハンドラ・アドレスに該当処理（発生した FE レベル例外の種類に対応した処理：ブート処理，保護例外に対応した処理など）への分岐処理を割り付けるために切り出されたエントリ処理専用ルーチンです。 備考 FE レベル例外エントリ処理についての詳細は，「4.5.1　FE レベル例外エントリ処理」を参照してください。 　→ リセット（RESET），FE レベル割り込み（FENMI，TRAP など），EI レベル割り込み（例外コード “OsIsrExceptionCode” ／例外コード “OsCounterExceptionCode” で未定義）などが発生した際，対応する処理（ブート処理，例外／割り込み安全対策処理など）への分岐処理を割り付けるために切り出されたエントリ処理専用ルーチンです。 備考 エントリ処理についての詳細は，「4.5.1　エントリ処理」を参照してください。
		52	［14.1　概　　要］ “(10) ユーティリティ関数 ” から以下のユーティリティ関数を削除 OSMEMORY_IS_Attribute
		52	［14.1　概　　要］ “(10) ユーティリティ関数 ” に以下のユーティリティ関数を追加。 OSIllegalException_SystemRegister_ExcCode, OSIllegalException_SystemRegister_ExcPC
		52	［14.1　概　　要］ “ 備考 ” の説明文章を以下のように変更。 … では，ユーティリティ関数 InitApplicationInterrupts, _kernel_fv0_InitializeIntService の規定が行われていません。 　→ … では，ユーティリティ関数 InitApplicationInterrupts, _kernel_fv0_InitializeIntService，OSIllegalException_SystemRegister_ExcCode, OSIllegalException_SystemRegister_ExcPC の規定が行われていません。
		53	［14.1.1　システム・サービスの呼び出し］ 本項の説明文章を以下のように変更。 なお，システム・サービスをアセンブリ言語で記述された処理プログラムから発行する場合は，システム・サービスを発行する直前で使用する C コンパイラ・パッケージの関数呼び出し規約に従った引数，および戻り番地の設定を行うことにより該当処理が行われます。 　→ また，システム・サービスをアセンブリ言語で記述された処理プログラムから発行する場合は，システム・サービスを発行する直前で使用する C コンパイラ・

Rev.	発行日	改定内容	
		ページ	ポイント
1.01	2014.09.30		パッケージの関数呼び出し規約に従った引数，および戻り番地の設定を行うことにより該当処理が行われます。 なお，システム初期化処理が完了する以前に割り込み管理のシステム・サービスを発行する際には，該当システム・サービスの発行前に _kernel_fv0_InitializeIntService の発行が必要となります。
		53	［14.1.1　システム・サービスの呼び出し］ 本項に以下の説明文章を “ 備考 3.” として追加。 SYSCALL 命令の発行時に不正な値が指定された際には，例外／割り込み安全対策処理 _kernel_e_IllegalExcEntry の呼び出しが行われます。
		54 - 55	［14.2.1　データ・タイプ］ “ 表 14.1　データ・タイプ ” に以下のマクロを追加。 boolean，float32，float64，sint8，sint8_least，sint16，sint16_least，sint32，sint32_least，sint64，uint8，uint8_least，uint16，uint16_least，uint32，uint32_least，uint64
		55	［14.2.1　データ・タイプ］ “ 表 14.1　データ・タイプ ” - “SystemRegisterType” の説明文章を以下のように変更。 システム・レジスタ（EIIC）の値 　→ システム・レジスタの値
		57	［14.2.6　異常終了］ “ 表 14.6　異常終了 ” の説明文章を以下のように変更。 INVALID_ISR　0x7FFF　発行有効範囲外の処理プログラムから発行した 　→ INVALID_OSAPPLICATION　0x7FFF　異常終了
		63	［14.2　データ・マクロ］ 本節に以下の項を追加。 14.2.14　アクセス権の有無
		72	［ChainTask］ “［機能］” - “ 備考 3.” の説明文章を以下のように変更。 対象タスクが基本タスクの場合，… は行われず，起動要求カウンタの加算処理のみが行われます。 　→ 対象タスクの種別が基本タスクの場合，… は行われず，優先度に応じたレディ・キューの最後尾へのキューイング処理，および起動要求カウンタの加算処理が行われます。
		80 - 81	［DisableAllInterrupts］ “［機能］” - “ 備考 6.” - “ スケーラビリティ・クラス 3（SC3）” の説明文章を以下のように変更。 処理プログラムを切り替える際，割り込みの受け付け許可処理（PSW の ID ビットに対する操作）を行います。 処理プログラムに共通フック・ルーチン（ErrorHook），固有フック・ルーチン（ErrorHook_*OsApplication*）が登録されていた場合，割り込みの受け付け許可処理と伴に，E_OS_DISABLEDINT（0x15）を引数としたフック・ルーチンの呼び出しが行われます。 　→ 処理プログラムの種類により，以下のように異なります。

Rev.	発行日	改定内容	
		ページ	ポイント
1.01	2014.09.30		【タスクの場合】 割り込みの受け付け許可処理（PSW の ID ビットに対する操作），クリティカル・セクションの終了を行います。 共通フック・ルーチン（ErrorHook），固有フック・ルーチン（ErrorHook_*OsApplication*）が登録されていた場合，E_OS_MISSINGEND（0x14）を引数としたフック・ルーチンの呼び出しが行われます。 【割り込みサービス・ルーチン（カテゴリ 2）の場合】 割り込みの受け付け許可処理（PSW の ID ビットに対する操作），クリティカル・セクションの終了を行います。 共通フック・ルーチン（ErrorHook），固有フック・ルーチン（ErrorHook_*OsApplication*）が登録されていた場合，E_OS_DISABLEDINT（0x15）を引数としたフック・ルーチンの呼び出しが行われます。 【固有フック・ルーチン（StartupHook_*OsApplication*）の場合】 割り込みの受け付け許可処理（PSW の ID ビットに対する操作），クリティカル・セクションの終了を行います。 【固有フック・ルーチン（ShutdownHook_*OsApplication*）の場合】 クリティカル・セクションの終了を行います。 【上記以外の場合】 動作は，保証されません。
		81	［DisableAllInterrupts］ “［機能］” - “ 備考 6.” の説明文章を以下のように変更。 … の場合には上記操作を，処理プログラムが ErrorHook_*OsApplication* の場合には何も処理は行わないようにしています。 → … の場合には上記操作を行います。
		83 - 84	［SuspendAllInterrupts］ “［機能］” - “ 備考 6.” - “ スケーラビリティ・クラス 3（SC3）” の説明文章を以下のように変更。 処理プログラムを切り替える際，割り込みの受け付け許可処理（PSW の ID ビットに対する操作），および禁止要求カウンタのクリア処理（禁止要求カウンタに 0x0 を設定）を行います。 処理プログラムに共通フック・ルーチン（ErrorHook），固有フック・ルーチン（ErrorHook_*OsApplication*）が登録されていた場合，割り込みの受け付け許可処理，禁止要求カウンタのクリア処理と伴に，E_OS_DISABLEDINT（0x15）を引数としたフック・ルーチンの呼び出しが行われます。 → 処理プログラムの種類により，以下のように異なります。 【タスクの場合】 割り込みの受け付け許可処理（PSW の ID ビットに対する操作），禁止要求カウンタのクリア処理（禁止要求カウンタに 0x0 を設定），クリティカル・セクションの終了を行います。 共通フック・ルーチン（ErrorHook），固有フック・ルーチン（ErrorHook_*OsApplication*）が登録されていた場合，E_OS_MISSINGEND（0x14）を引数としたフック・ルーチンの呼び出しが行われます。 【割り込みサービス・ルーチン（カテゴリ 2）の場合】 割り込みの受け付け許可処理（PSW の ID ビットに対する操作），禁止要求カウンタのクリア処理（禁止要求カウンタに 0x0 を設定），クリティカル・セクションの終了を行います。

Rev.	発行日	改定内容	
		ページ	ポイント
1.01	2014.09.30		共通フック・ルーチン（ErrorHook），固有フック・ルーチン（ErrorHook_*OsApplication*）が登録されていた場合，E_OS_DISABLEDINT（0x15）を引数としたフック・ルーチンの呼び出しが行われます。 【固有フック・ルーチン（StartupHook_*OsApplication*）の場合】 割り込みの受け付け許可処理（PSW の ID ビットに対する操作），禁止要求カウンタのクリア処理（禁止要求カウンタに 0x0 を設定），クリティカル・セクションの終了を行います。 【固有フック・ルーチン（ShutdownHook_*OsApplication*）の場合】 クリティカル・セクションの終了を行います。 【上記以外の場合】 動作は，保証されません。
		84	［SuspendAllInterrupts］ “［機能］” - “ 備考 6.” の説明文章を以下のように変更。 … の場合には上記操作を，処理プログラムが ErrorHook_*OsApplication* の場合には何も処理は行わないようにしています。 → … の場合には上記操作を行います。
		85	［ResumeOSInterrupts］ “［機能］” に以下の説明文章を “ 備考 5.” として追加。 SuspendedOSInterrupts を発行する以前に GetResource（シーリング値 : INTPRI*x*）の発行が行われていた場合，本システム・サービスを発行した際のプライオリティ・マスク・レジスタ（PMR）の内容は，SuspendedOSInterrupts を発行する以前の値（INTPRI0 ～ INTPRI*x* に対応した割り込み要因の受け付けが禁止された状態）となります。
		86	［SuspendOSInterrupts］ “［機能］” - “ 備考 4.” の説明文章を以下のように変更。 … の加算処理（禁止要求カウンタから 0x1 を減算）のみを行い，… → … の加算処理（禁止要求カウンタに 0x1 を加算）のみを行い，…
		87	［SuspendOSInterrupts］ “［機能］” - “ 備考 6.” - “ スケーラビリティ・クラス 3（SC3）” の説明文章を以下のように変更。 処理プログラムを切り替える際，割り込みの受け付け許可処理（PMR の PM*n* ビットに対する操作），および禁止要求カウンタのクリア処理（禁止要求カウンタに 0x0 を設定）を行います。 処理プログラムに共通フック・ルーチン（ErrorHook），固有フック・ルーチン（ErrorHook_*OSApplication*）が登録されていた場合，割り込みの受け付け許可処理，禁止要求カウンタのクリア処理と伴に，E_OS_DISABLEDINT（0x15）を引数としたフック・ルーチンの呼び出しが行われます。 → 処理プログラムの種類により，以下のように異なります。 【タスクの場合】 割り込みの受け付け許可処理（PMR の PM*n* ビットに対する操作），禁止要求カウンタのクリア処理（禁止要求カウンタに 0x0 を設定），クリティカル・セクションの終了を行います。 共通フック・ルーチン（ErrorHook），固有フック・ルーチン（ErrorHook_*OSApplication*）が登録されていた場合，E_OS_MISSINGEND（0x14）を引数としたフック・ルーチンの呼び出しが行われます。

Rev.	発行日	改定内容	
		ページ	ポイント
1.01	2014.09.30		【割り込みサービス・ルーチン（カテゴリ 2）の場合】 割り込みの受け付け許可処理（PMR の PM*n* ビットに対する操作），禁止要求カウンタのクリア処理（禁止要求カウンタに 0x0 を設定），クリティカル・セクションの終了を行います。 共通フック・ルーチン（ErrorHook），固有フック・ルーチン（ErrorHook_*OSApplication*）が登録されていた場合，E_OS_DISABLEDINT（0x15）を引数としたフック・ルーチンの呼び出しが行われます。 【固有フック・ルーチン（StartupHook_*OsApplication*）の場合】 割り込みの受け付け許可処理（PMR の PM*n* ビットに対する操作），禁止要求カウンタのクリア処理（禁止要求カウンタに 0x0 を設定），クリティカル・セクションの終了を行います。 【固有フック・ルーチン（ShutdownHook_*OsApplication*）の場合】 クリティカル・セクションの終了を行います。 【上記以外の場合】 動作は，保証されません。
		87	［SuspendOSInterrupts］ “［機能］” - “ 備考 6.” の説明文章を以下のように変更。 … の場合には上記操作を，処理プログラムが ErrorHook_*OsApplication* の場合には何も処理は行わないようにしています。 → … の場合には上記操作を行います。
		140	［CheckObjectAccess］ “［戻り値］” - 表 - “NO_ACCESS” の説明文章を以下のように変更。 クリティカル・セクションから発行した基本タスクから発行した → クリティカル・セクションから発行した
		143	［TerminateApplication］ “［機能］” - “ 備考 2.” - “RESTART（0x1）” の説明文章を以下のように変更。 タスク識別子 “OsRestartTask” で指定されたタスクを SUSPENDED 状態から READY 状態へと遷移 → タスク識別子 “OsRestartTask” の指定が行われている際には，該当タスクを SUSPENDED 状態から READY 状態へと遷移
		144	［TerminateApplication］ “［機能］” に以下の説明文章を “ 備考 5.” として追加。 ErrorHook_*OsApplication* から本システム・サービスを発行した場合，引数 *Application* に “ 本システム・サービスを発行した ErrorHook_*OsApplication* が所属している OS アプリケーション以外 ” を指定した際には，戻り値として E_OS_CALLEVEL（0x2）を返します。
		144	［TerminateApplication］ “［戻り値］” - “E_OS_ACCESS” の説明文章を以下のように変更。 本システム・サービスを発行した処理プログラムが所属している OS アプリケーションは，引数 *Application* で指定された OS アプリケーションに対するアクセス権がない → 本システム・サービスを発行した処理プログラムが所属している OS アプリケー

Rev.	発行日	改定内容	
		ページ	ポイント
1.01	2014.09.30		ションが Non-Trusted である
		153	［GetActiveApplicationMode］ “［戻り値］” に以下の説明文章を “ 備考 ” として追加。 StartOS の引数 *OsAppMode* に不正な値が指定された場合，上記以外の値が返されることがあります。
		154	［14.4.10　ユーティリティ関数］ “ 表 14.24　ユーティリティ関数 ” に以下のユーティリティ関数を追加。 OSIllegalException_SystemRegister_ExcCode, OSIllegalException_SystemRegister_ExcPC
		154	［14.4.10　ユーティリティ関数］ “ 備考 ” の説明文章を以下のように変更。 … では，ユーティリティ関数 InitApplicationInterrupts, _kernel_fv0_InitializeIntService の規定が行われていません。 　→ … では，ユーティリティ関数 InitApplicationInterrupts, _kernel_fv0_InitializeIntService，OSIllegalException_SystemRegister_ExcCode, OSIllegalException_SystemRegister_ExcPC の規定が行われていません。
		155	［InitiApplicationInterrupts］ “［機能］” の “ 備考 3.” を以下のように “ 備考 3.”，“ 備考 4.” に分割。 備考 3. 本ユーティリティ関数は，引数 *RestartOption* に RESTART（0x1）を指定して TerminateApplication を発行した際，SUSPENDED 状態から READY 状態へと遷移するタスク（タスク識別子 “OsRestartTask”）で発行することを想定しています。 本ユーティリティ関数を発行有効範囲外の処理プログラムから発行した場合，何も処理は行われず，エラーとしても扱われません。 　→ 備考 3. 本ユーティリティ関数は，引数 *RestartOption* に RESTART（0x1）を指定して TerminateApplication を発行した際，SUSPENDED 状態から READY 状態へと遷移するタスク（タスク識別子 “OsRestartTask”）で発行することを想定しています。 備考 4. 本ユーティリティ関数を発行有効範囲外の処理プログラムから発行した場合，何も処理は行われず，エラーとしても扱われません。
		157 - 158	［14.4.10　ユーティリティ関数］ 本項に以下のユーティリティ関数を追加。 OSIllegalException_SystemRegister_ExcCode, OSIllegalException_SystemRegister_ExcPC
		159	［OSErrorGetServiceId］ “［機能］” に以下の説明文章を “ 備考 2.” として追加。 本ユーティリティ関数を発行する処理プログラム（ErrorHook, ErrorHook_*OsApplication*）の呼び出し要因が “ システム・サービスの異常終了 ” 以外の場合，本ユーティリティ関数の戻り値は不定値となります。
		162	［OSError_*SystemService_Parameter*］ “［機能］” に以下の説明文章を “ 備考 ” として追加。 本ユーティリティ関数を発行する処理プログラム（ErrorHook,

Rev.	発行日	改定内容	
		ページ	ポイント
1.01	2014.09.30		ErrorHook_*OsApplication*）の呼び出し要因が “ システム・サービスの異常終了 ” 以外の場合，本ユーティリティ関数の戻り値は不定値となります。
		163	［A.2　起動方法］ “(1) *cf_file*” に以下の説明文章を “ 備考 2.” として追加。 コンフィギュレータは，拡張子が .arxml，.xml の場合は ARXML 形式として，拡張子が .oil の場合は OIL 形式として *cf_file* を扱います。 なお，*cf_file* に指定可能な拡張子は，.arxml，.xml，.oil に限られます。
		164	［A.2　起動方法］ “(3) -o △ *sit_file*” に以下の説明文章を “ 備考 2.” として追加。 -o △ *sit_file* は，以下のように複数の指定が可能ですが，最初に指定した -o △ *sit_file* のみが有効な起動オプションとして扱われ，他の -o △ *sit_file* は無効な起動オプションとして扱われます。 したがって，以下の例では，-o △ sit_file1.c が有効な起動オプションとなり，-o △ sit_file2.c，-o △ sit_file3.c は無視されます。 `C:\> Os_Configurator.exe △ -o △ sit_file1.c △ -o △ sit_file2.c △ -o △ sit_file3.c △ cf_file.oil`
		164	［A.2　起動方法］ “(5) -e △ *entry_file*” に以下の説明文章を “ 備考 2.” として追加。 -e △ *entry_file* は，以下のように複数の指定が可能ですが，最初に指定した -e △ *entry_file* のみが有効な起動オプションとして扱われ，他の -e △ *entry_file* は無効な起動オプションとして扱われます。 したがって，以下の例では，-e △ entry_file1.850 が有効な起動オプションとなり，-e △ entry_file2.850，-e △ entry_file3.850 は無視されます。 `C:\> Os_Configurator.exe △ -e △ entry_file1.850 △ -e △ entry_file2.850 △ -e △ entry_file3.850 △ cf_file.oil`
		164	［A.2　起動方法］ “(8) -I △ *path_name*” に以下の説明文章を “ 備考 2.” として追加。 本起動オプションは，以下のように複数の指定が（最大：255）が可能です。 `C:\> Os_Configurator.exe △ -I △ path_name1 △ -I △ path_name2 △ -I △ path_name3 △ cf_file.oil`
		168 - 170	［A.4.1　フェイタル・エラー］ “ 表 A.1　フェイタル・エラー ” に以下のフェイタル・エラーを追加。 E3007，E4020
		172	［A.4.2　アボート・エラー］ “ 表 A.2　アボート・エラー ” - “F1012” のメッセージを以下のように変更。 Too many lines. 　→ Too many File lines.
		174 - 175	［B.1　概　　要］ “(6) キーワード ” に以下のキーワードを追加。 OSTMCNT，TRACESYSTEMENTRY，TRACESYSTEMEXIT，TRACETASKSTATUS

Rev.	発行日	改定内容	
		ページ	ポイント
1.01	2014.09.30	184	［B.4.3　OS アプリケーション情報］ “(6) SPID “*OsApplicationSPID*” - “ 備考 1.” の説明文章を以下のように変更。 本項目は，信頼性 “OsTrusted” で FALSE を定義した場合に限り指定可能となります。 　→ 本項目は，コア識別子 “*OsSystemCpuCore*” で G3M を，信頼性 “OsTrusted” で FALSE を定義した場合に限り指定可能となります。
		187	［B.4.3　OS アプリケーション情報］ “(17) メモリ領域識別子 “*OsAppMemoryAreaNameRef*” - “ 備考 1.” の説明文章を以下のように変更。 本項目は，信頼性 “OsTrusted” で FALSE を定義した場合に限り指定可能となります。 　→ 本項目は，コア識別子 “*OsSystemCpuCore*” で G3M を，信頼性 “OsTrusted” で FALSE を定義した場合に限り指定可能となります。
		192	［B.4.5　イベント情報］ 本項に以下の説明文章を “ 備考 ” として追加。 RV850 では，本情報で定義したイベントをタスク情報で未定義の場合，フェイタル・エラー E4006 としています。
		197	［B.4.7　OS 情報］ “(5) ステータス種別 “*OsStatus*”” から以下の説明文章を削除。 STANDARD：標準ステータス
		200	［B.4.7　OS 情報］ “(19) メモリ領域識別子 “*OsMemoryAreaNameRef*”” - “ 備考 2.” の説明文章を以下のように変更。 本項目は，以下のように複数の定義（最大：7 個）が可能です。 　→ 本項目は，以下のように複数の定義（コア識別子 “*OsSystemCpuCore*” で G3K を定義した場合：最大 3 個，コア識別子 “*OsSystemCpuCore*” で G3M を定義した場合：最大 7 個）が可能です。
		215	［C.1　概　　要］ “ 表 C.1　メモリ領域 ” に以下の説明文章を追加。 .kernel_*address*　割り込みハンドラ・アドレス・テーブル
		215	［C.2　標準処理領域（.kernel_system）］ 表から以下のカーネル・ライブラリを削除追加。 libecc2stdsc1.a libecc2stdsc1_fpu.a
		215	［C.2　標準処理領域（.kernel_system）］ 表の説明文章を以下のように変更。 ECC2，拡張ステータス，SC1，FPU 対応なし libecc2extsc1.a　12.0K バイト ECC2，拡張ステータス，SC1，FPU 対応あり libecc2extsc1_fpu.a　12.1K バイト ECC2，拡張ステータス，SC3，FPU 対応なし libecc2extsc3.a　21.9K バイト ECC2，拡張ステータス，SCS，FPU 対応あり libecc2extsc3_fpu.a　22.1K バイト 　→

Rev.	発行日	改定内容	
		ページ	ポイント
1.01	2014.09.30		libecc2extsc1.a　12.2K バイト libecc2extsc1_fpu.a　12.3K バイト libecc2extsc3.a　22.1K バイト libecc2extsc3_fpu.a　22.3K バイト
		215	［C.2　標準処理領域（.kernel_system）］ 表に以下の説明文章を追加。 libecc2extsc3_g3k.a　21.5 K バイト
		216	［C.3　インタフェース領域（.kernel_interface）］ 表から以下のカーネル・ライブラリを削除。 libecc2stdsc1.a libecc2stdsc1_fpu.a
		216	［C.3　インタフェース領域（.kernel_interface）］ 表の説明文章を以下のように変更。 ECC2，拡張ステータス，SC1，FPU 対応なし libecc2extsc1.a ECC2，拡張ステータス，SC1，FPU 対応あり libecc2extsc1_fpu.a ECC2，拡張ステータス，SC3，FPU 対応なし libecc2extsc3.a ECC2，拡張ステータス，SCS，FPU 対応あり libecc2extsc3_fpu.a → libecc2extsc1.a libecc2extsc1_fpu.a libecc2extsc3.a libecc2extsc3_fpu.a
		216	［C.3　インタフェース領域（.kernel_interface）］ 表に以下の説明文章を追加 libecc2extsc3_g3k.a　0.92 K バイト
		217	［C.4　定数データ領域（.kernel_const）］ “(1) SC1 用定数データ領域（.kernel_const）” の計算式を以下のように変更。 KERNEL_CONST = 　align 4 (　　652 　　… 　　+ 4 * (*SystemMaxExceptionCode* - 4096) 　) 　→ KERNEL_CONST = 　align 4 (　　674 　　… 　　+ 4 * (*SystemMaxExceptionCode* - 4095) 　)
		217	［C.4　定数データ領域（.kernel_const）］ “(1) SC1 用定数データ領域（.kernel_const）” - 表 - “*AppMode_Num*” の説明文章を以下のように変更。 アプリケーション・モード情報の総数 → アプリケーション・モード情報の総数（OSDEFAULTAPPMODE に関する定義は除く）

Rev.	発行日	改定内容	
		ページ	ポイント
1.01	2014.09.30	219	［C.4　定数データ領域（.kernel_const)］ “(2) SC3 用定数データ領域（.kernel_const）” の計算式を以下のように変更。 KERNEL_CONST = 　align 4 (　　1232 　　… 　　+ 4 * (*SystemMaxExceptionCode* - 4096) 　) 　→ KERNEL_CONST = 　align 4 (　　1254 　　… 　　+ 4 * (*SystemMaxExceptionCode* - 4095) 　)
		219	［C.4　定数データ領域（.kernel_const)］ “(2) SC3 用定数データ領域（.kernel_const）” - 表 - “*AppMode_Num*” の説明文章を以下のように変更。 … の総数 　→ … の総数（OSDEFAULTAPPMODE に関する定義は除く）
		223	［C.6　定数データ領域（.kernel_work)］ “(1) SC1 用定数データ領域（.kernel_work）” の計算式を以下のように変更。 KERNEL_WORK = 　align 4 (　　132 　　… 　　+ 28 * *Task_Num* 　　… 　) 　→ KERNEL_CONST = 　align 4 (　　128 　　… 　　+ 24 * *Task_Num* 　　… 　)
		224	［C.6　定数データ領域（.kernel_work)］ “(2) SC3 用定数データ領域（.kernel_work）” の計算式を以下のように変更。 KERNEL_WORK = 　align 4 (　　144 　　… 　　+ 36 * *Task_Num* 　　… 　) 　→ KERNEL_CONST = 　align 4 (　　148 　　… 　　+ 32 * *Task_Num*

Rev.	発行日	改定内容	
		ページ	ポイント
1.01	2014.09.30		…)
		227	［C.7.1　システム・スタック］ “(1) SC1 用システム・スタック ” の計算式を以下のように変更。 SystemStack = 　align 4 (　　60 　　… 　　+ Max （ 　　… 　　36 + *StartupHookStack_Siz*, 　　… 　　) 　　… 　) 　→ SystemStack = 　align 4 (　　44 　　… 　　+ Max （ 　　… 　　20 + *StartupHookStack_Siz*, 　　… 　　) 　　… 　)
		228	［C.7.1　システム・スタック］ “(2) SC3 用システム・スタック ” の計算式を以下のように変更。 SystemStack = 　align 4 (　　Max （ 　　Max (　　324, 　　264 + *ProtectionHookStack_Siz* + *ErrorHookStack_Siz*, 　　108 + *StartupHookStack_Siz* + *ErrorHookStack_Siz* 　　) 　　… 　　) 　) 　→ SystemStack = 　align 4 (　　Max （ 　　Max (　　320, 　　260 + *ProtectionHookStack_Siz* + *ErrorHookStack_Siz,* 　　104 + *StartupHookStack_Siz* + *ErrorHookStack_Siz* 　　) 　　… 　　) 　)
		229	［C.7.2　OS アプリケーション・タスク］ 本項の計算式を以下のように変更。

Rev.	発行日	改定内容	
		ページ	ポイント
1.01	2014.09.30		OsApplicationStack = 　align 4 (　　… 　　+ Max (　　56 + *ErrorHookStack_Siz*, 　　… 　　) 　　… 　) 　→ OsApplicationStack = 　align 4 (　　… 　　+ Max (　　104 + *ErrorHookStack_Siz*, 　　… 　　) 　　… 　)
		230	［C.7.3　タスク・スタック（基本タスク）］ “(1) SC1 用タスク・スタック（基本タスク）” の計算式を以下のように変更。 TaskStack = 　align 4 (　　160 　　… 　) 　→ TaskStack = 　align 4 (　　164 　　… 　)
		233	［C.7.5　割り込みサービス・ルーチン・スタック（カテゴリ 2）］ “(1) SC1 用割り込みサービス・ルーチン・スタック（カテゴリ 2）” の計算式を以下のように変更。 IsrStack = 　align 4 (　　252 　　… 　) 　→ IsrStack = 　align 4 (　　256 　　… 　)
		234	［C.　メモリ容量］ 本付録に “C.8　割り込みハンドラ・アドレス・テーブル（.kernel_*address*）” を追加。

Rev.	発行日	改定内容	
		ページ	ポイント
1.02	2015.06.10	18	[2.5 リンカ・ディレクティブ・ファイルの記述] “ 表 2.1” の表題を以下のように変更。 規定された割り付け先 → RV850 によって規定されるオブジェクト割り付け先
		18	[2.5 リンカ・ディレクティブ・ファイルの記述] “ 表 2.1” 中の “.kernel_work” の説明を以下のように修正。 定数データ領域 (RAM) → 変数データ領域 (RAM)
		18	[2.5 リンカ・ディレクティブ・ファイルの記述] “ 備考 1.” を以下のように修正。 X：書き込み可 W：実行可 → W：書き込み可 X：実行可
		19	[2.6 ロード・モジュールの生成] 以下の説明文章を “ 備考 4.” に追加。 その際，構築するシステムで浮動小数点演算を全く用いない場合（-fnone オプション指定時）は，-D__WITHOSONLY__ オプションに加えて -D__NOFLOAT__ オプションを指定する必要があります。
		19	[2.6 ロード・モジュールの生成] 以下の説明文章を “ 備考 5.” に追加。 SIT ファイル（Os_Cfg.c）をコンパイルする際には -sda=0 オプションを指定することを推奨します。-sda=0 を指定しない場合には，表 2.1 の .kernel_const セクションと .kernel_work セクションに格納されることを想定している RV850 のデータが，それぞれ .sbss セクションと .rosdata セクションに配置される可能性があります。特に，スケーラビリティ・クラスが SC3 の SIT ファイルに -sda=0 を指定しない際には，.sbss セクションと .rosdata セクションを，全ての Non-Trusted な OS アプリケーションからの書き込みアクセスが不可能な領域に配置する必要が生じます。
		20	[3.1 概　　要] 以下の説明文章を “(1) 基本タスク ” に追加。 基本タスクが動作中に使用するスタックは，以下の通りです。 - スケーラビリティ・クラスが SC1 の場合 システム・スタック・サイズ “OsStackSize” で定義された「システム・スタック」 - スケーラビリティ・クラスが SC3 の場合 OS アプリケーション・スタック・サイズ “OsAppStackSize” で定義された「OS アプリケーション・スタック」
		20	[3.1 概　　要] 以下の説明文章を “(2) 拡張タスク ” に追加。 拡張タスクが動作中に使用するスタックは，タスク・スタック・サイズ “OsTaskStackSize” で定義された「タスク・スタック」です。

Rev.	発行日	改定内容	
		ページ	ポイント
1.02	2015.06.10	21	[3.1.2 スタック・モニタリング機能] 以下の説明文章を追加。 この機能は，タスク動作中に RV850 の処理に移行したタイミングにおいて，RV850 の処理に必要なスタック (タスク・スタック，システム・スタック，OS アプリケーション・スタック) の残量が存在するかどうかをチェックします。具体的には，下記のタイミングでスタック残量のチェックを行います。 - タスクから発行したシステム・サービス処理の開始時 - タスク実行中に発生したカテゴリ 2 割り込みの前処理の開始時
		21	[3.1.2 スタック・モニタリング機能] 説明文章を以下のように変更。 これにより，スタック・オーバーフローが発生した場合 → タスク動作中にスタック・オーバーフローを検知した場合
		21	[3.1.2 スタック・モニタリング機能] 以下の説明文章を “ 備考 2.” に追加。 RV850 のスタック・モニタリング機能では，タスクから RV850 の処理に移行しない限りは，スタック・オーバーフローの検出ができません。タスク動作中の任意のタイミングで発生しうるスタック・オーバーフローを検知する必要がある場合には，スケーラビリティ・クラス SC3 の Non-Trusted な OS アプリケーションで提供されるメモリ・アクセス保護機能を使用してください。
		24	[4.1.1 スタック・モニタリング機能] 以下の説明文章を追加。 この機能は，割り込みサービス・ルーチン（カテゴリ 2）の動作中に RV850 の処理に移行したタイミングにおいて，RV850 の処理に必要なスタック (システム・スタック，OS アプリケーション・スタック) の残量が存在するかどうかをチェックします。具体的には，下記のタイミングでスタック残量のチェックを行います。 - 割り込みサービス・ルーチン（カテゴリ 2）から発行したシステム・サービス処理の開始時 - 割り込みサービス・ルーチン（カテゴリ 2）実行中に発生したカテゴリ 2 割り込みの前処理の開始時
		24	[4.1.1 スタック・モニタリング機能] 説明文章を以下のように変更。 これにより，スタック・オーバーフローが発生した場合 → 割り込みサービス・ルーチン（カテゴリ 2）動作中にスタック・オーバーフローを検知した場合
		24	[4.1.1 スタック・モニタリング機能] 以下の説明文章を “ 備考 3.” に追加。 RV850 のスタック・モニタリング機能では，割り込みサービス・ルーチン（カテゴリ 2）から RV850 の処理に移行しない限りは，スタック・オーバーフローの検出ができません。割り込みサービス・ルーチン動作中の任意のタイミングで発生しうるスタック・オーバーフローを検知する必要がある場合には，スケーラビリティ・クラス SC3 の Non-Trusted な OS アプリケーションで提供されるメモリ・アクセス保護機能を使用してください。

Rev.	発行日	改定内容	
		ページ	ポイント
1.02	2015.06.10	32	[4.5.2 例外／割り込み安全対策処理] 例外／割り込み安全対策処理を C 言語で記述する際の記述例に，以下の記述を追加。 `#pragma ghs interrupt`
		32	[4.5.2 例外／割り込み安全対策処理] 以下の説明文章を追加。 例外／割り込み安全対策処理を記述する際には，以下に示す注意点があります。 (1) レジスタの退避／復帰 ... (5) システム・サービスの発行
		34	[5.1 概　　要] 説明文章を以下のように変更。 RV850 では，限られた数の資源が競合することを防止するためのメカニズムとして “ リソース管理機能 ” を提供し，3 種類のリソースをサポートしています。 　→ RV850 では，排他制御のためのメカニズムとして “ リソース管理機能 ” を提供しています。リソースは “ 優先度上限プロトコル (Priority Ceiling Plotocols)” によって排他制御を実現し，限られた数の共有資源（データ，周辺機器，共通関数等）を使用するタスクや割り込みサービス・ルーチン（カテゴリ 2）による資源競合やデッドロックを防止できます。 RV850 では，3 種類のリソースをサポートしています。
		34	[5.1.1 シーリング値] Rev.1.01 の ”5.1.2 シーリング値 ” から移動し，” 備考 ” を以下のように変更。 リソースの獲得区間中に保留された割り込みは，該当リソースの解放時に受け付けられます。 　→ INTPRI0 ～ INTPRI15 のシーリング値のリソースの獲得区間中に保留された割り込みは，該当リソースの解放時に受け付けられます。
		34	[5.1.2 スケジューラ・リソース] Rev.1.01 の ”5.1.1 スケジューラ・リソース ” から移動し，説明文章を以下のように変更。 OSEK/VDX 仕様では，処理プログラムからスケジューラの起動を動的に禁止／許可するための手段として，識別子：RES_SCHEDULER を持つリソースを規定しています。 　→ OSEK/VDX 仕様では，タスクからスケジューラの起動を動的に禁止／許可するための手段として，シーリング値：29，識別子：RES_SCHEDULER を持つリソースを規定しています。
		34	[5.1.2 スケジューラ・リソース] ” 備考 2.” を追加。

Rev.	発行日	改定内容	
		ページ	ポイント
1.02	2015.06.10	35	[5.2 リソースの生成] “ 備考 ” の説明文章を以下のように変更。 スケジューラ・リソースの生成は，スケジューラ・リソース “OsUseResScheduler” で TRUE を定義すると伴に，リソース情報の識別子 “OsResource” で RES_SCHEDULER を指定した定義を行うことにより実現されます。 → OSEK/VDX 仕様では，スケジューラ・リソースの生成は，スケジューラ・リソース “OsUseResScheduler” で TRUE を定義することで自動的に行われることとなっていますが，AUTOSAR 仕様では自動的な生成は行わないとしています。そのため，RV850 においてスケジューラ・リソースを生成する際には，以下のように CF ファイルに定義してください。
		41	[10.1.1 信 頼 性] “(1)Trusted な OS アプリケーション ” の説明文章を以下のように変更。 なお，RV850 では，Trusted な OS アプリケーションに所属している処理プログラムについては，信頼性が確保されているため，スーパバイザ・モード（PSW の UM ビットに 0 を設定）で動作させ，メモリ・アクセス保護機能，および周辺 I/O 保護機能を非適用としています。 → なお，RV850 では，Trusted な OS アプリケーションに所属している処理プログラムについては，信頼性が確保されているため，スーパバイザ・モード（PSW の UM ビットに 0 を設定），およびシステム・プロテクション番号（MCFG0 レジスタの SPID ビット）を 0 に設定して動作させ，メモリ・アクセス保護機能，および周辺 I/O 保護機能を非適用としています。
		41	[10.1.1 信 頼 性] “(2)Non-Trusted な OS アプリケーション ” の説明文章を以下のように変更。 なお，RV850 では，Non-Trusted な OS アプリケーションに所属している処理プログラムについては，信頼性が確保されていないため，ユーザ・モード（PSW の UM ビットに 1 を設定）で動作させ，メモリ・アクセス保護機能，および周辺 I/O 保護機能を適用可能としています。 → なお，RV850 では，Non-Trusted な OS アプリケーションに所属している処理プログラムについては，信頼性が確保されていないため，ユーザ・モード（PSW の UM ビットに 1 を設定），および SPID“OsApplicationSPID” で指定したシステム・プロテクション番号（MCFG0 レジスタの SPID ビット）で動作させ，メモリ・アクセス保護機能，および周辺 I/O 保護機能を適用可能としています。
		42	[10.1.3 メモリ・アクセス保護機能] 以下の説明文章を “ 備考 1.” に追加。 また，" スタック・モニタリング機能 “OsStackMonitoring” の指定に関わらず，Non-Trusted な OS アプリケーションが使用するスタック領域の監視を常時行います。
		42	[10.1.3 メモリ・アクセス保護機能] 以下の説明文章を “ 備考 2.” に追加。 スタック領域のオーバーフローを検出するために考慮するスタック・ポインタの閾値 (アドレス) は，タスクや割り込みサービス・ルーチン（カテゴリ 2）が使用中のスタック領域のトップ・アドレスに対し，RV850 のシステム・サービスが使用するスタック使用量の最大値を加味したアドレスです。これは，Non-Trusted な OS アプリケーションであっても，RV850 のシステム・サービス実行中はスーパバイザ・モードへの移行により，メモリ・アクセス保護機能が動作しないことに対する対策です。

Rev.	発行日	改定内容	
		ページ	ポイント
1.02	2015.06.10	44	[10.2Trusted 関数] 説明文書を以下のように変更。 信頼性が確保されている OS アプリケーションについては，OS アプリケーション単位に固有な Trusted 関数を所属させることができます。 → Trusted な OS アプリケーションについては，OS アプリケーション単位に固有な Trusted 関数を所属させることができます。 なお，Trusted 関数は，処理プログラムから CallTrustedFunction を発行することにより呼び出されます。 → Trusted 関数は，処理プログラムから CallTrustedFunction を発行することにより呼び出されます。CallTrustedFunction は他の Non-Trusted な OS アプリケーションからも呼び出すことが可能なので，Non-Trusted な OS アプリケーションから一時的に保護機能非適用の処理をさせたい場合に Trusted 関数を使用することができます。
		44	[10.2Trusted 関数] “ 備考 3.” の説明文章を以下のように変更。 Trusted 関数はスーパバイザ・モードで動作するため，CallTrustedFunction を Non-Trusted な OS アプリケーションに所属している処理プログラムから発行した場合，モード切り替え処理（ユーザ・モードからスーパバイザ・モードへと移行）が行われます。 → Trusted 関数はスーパバイザ・モードで動作するため，CallTrustedFunction を Non-Trusted な OS アプリケーションに所属している処理プログラムから発行した場合，モード切り替え処理（ユーザ・モードからスーパバイザ・モードへと移行），ならびにシステム・プロテクション番号（MCFG0 レジスタの SPID ビット）の切り替え（Trusted 関数実行中は 0）が行われます。
		44	[10.2Trusted 関数] “ 備考 4.” の説明文章を以下のように変更。 Trusted 関数は，スーパバイザ・モードで動作するため，デバイスが提供しているメモリ保護機能（MPU）は適用外となります。 → Trusted 関数はスーパバイザ・モードで動作するため，メモリ・アクセス保護機能は適用外となります。また，システム・プロテクション番号（MCFG0 レジスタの SPID ビット）が 0 で動作するため，システム・プロテクション番号と連動した周辺 I/O 保護機能は適用外となります。
		45	[10.2.3Trusted 関数の引き継ぎデータ] 本節を新規に追加。
		47	[10.3 固有フック・ルーチン] 以下の説明文章を ” 備考 ” として追加。 Non-Trusted な OS アプリケーションの固有フック・ルーチンは，ユーザ・モード（PSW の UM ビットに 1 を設定），ならびに SPID“OsApplicationSPID” で指定したシステム・プロテクション番号（MCFG0 レジスタの SPID ビット）で動作し，メモリ・アクセス保護機能，およびシステム・プロテクション番号と連動した周辺 I/O 保護機能を適用可能としています。Trusted な OS アプリケーションの固有フック・ルーチンは，スーパバイザ・モード（PSW の UM ビットに 0 を設定），ならびにシステム・プロテクション番号（MCFG0 レジスタの SPID ビット）が 0 で動作し，メモリ・アクセス保護機能，およびシステム・プロテクション番号と連動した周辺 I/O 保護機能は適用外となります。

Rev.	発行日	改定内容	
		ページ	ポイント
1.02	2015.06.10	50	[11.2 共通フック・ルーチン] 以下の説明文章を “ 備考 ” として追加。 共通フック・ルーチンは，スーパバイザ・モードで動作するため，メモリ・アクセス保護機能は適用外となります。また，システム・プロテクション番号（MCFG0 レジスタの SPID ビット）が 0 で動作するため，システム・プロテクション番号と連動した周辺 I/O 保護機能は適用外となります。
		56	[12.3 アイドル・ハンドラ] “ 備考 1.” の説明文書を以下のように変更。 アイドル・ハンドラは，スーパバイザ・モードで動作するため，デバイスが提供しているメモリ保護機能（MPU）は適用外となります。 → アイドル・ハンドラは，スーパバイザ・モードで動作するため，メモリ・アクセス保護機能は適用外となります。また，システム・プロテクション番号（MCFG0 レジスタの SPID ビット）が 0 で動作するため，システム・プロテクション番号と連動した周辺 I/O 保護機能は適用外となります。
		141	[14.4.8 OS アプリケーション管理] “CallTrustedFunction” の [機能] の説明文章を以下のように修正。 なお，Trusted 関数を呼び出す際には，引数 FunctionParams で指定された値を Trusted 関数の第 1 引数として渡しています。 → なお，本システム・サービス処理から Trusted 関数の本体を呼び出す際には，引数 FunctionIndex で指定された値を Trusted 関数の第 1 引数として，引数 FunctionParams で指定された値（ポインタ）を Trusted 関数の第 2 引数として渡しています。Trusted 関数への引き継ぎデータ (パラメータ) の詳細は 10.2.3 Trusted 関数の引き継ぎデータを参照してください。
		141	[14.4.8 OS アプリケーション管理] “CallTrustedFunction” の “ 備考 4.” の説明文章を以下のように修正。 本システム・サービスでは引数 FunctionParams の正当性を確認していません。したがって，引数 FunctionParams の正当性を確認する必要がある場合は，本システム・サービスの発行により呼び出された Trusted 関数で行ってください。 → 本システム・サービスでは引数 FunctionParams の指し示す引継ぎデータの正当性を確認できません。したがって，引継ぎデータの正当性を確認する必要がある場合は，本システム・サービスの発行により呼び出された Trusted 関数本体で行ってください。
		159	[14.4.9 OS 実行管理] “StartOS” の ” 備考 2.” の説明文章を以下のように修正。 本システム・サービスでは，システム・レジスタに対する操作を行うため，スーパバイザ・モード（PSW の UM ビットが 0）での発行が必須となります。 → 本システム・サービスでは，システム・レジスタに対する操作を行うため，スーパバイザ・モード（PSW の UM ビットが 0）での発行が必須となります。また，システム・プロテクション番号（MCFG0 レジスタの SPID ビット）の設定（4.2 ブート処理 備考 6.）が完了している状態での発行が必須となります。

Rev.	発行日	改定内容	
		ページ	ポイント
1.02	2015.06.10	167	[14.4.10 ユーティリティ関数] “_kernel_fv0_InitializeIntService” の ” 備考 2.” の説明文章を以下のように修正。 本ユーティリティ関数では，システム・レジスタに対する操作を行うため，スーパバイザ・モード（PSW の UM ビットが 0）での発行が必須となります。 → 本ユーティリティ関数では，RV850 の内部データに対する操作を行うため， スーパバイザ・モード（PSW の UM ビットが 0）での発行が必須となります。
		175	[A.2 起動方法] コマンド・プロンプトから起動する方法を以下のように修正。 OS_Configurator.exe → Os_Configurator.exe
		196	[B.4.3 OS アプリケーション情報] 以下のように変更。 (3) スタック・サイズ “*OsAppStackSize*” → (3)OS アプリケーション・スタック・サイズ “*OsAppStackSize*”
		196	[B.4.3 OS アプリケーション情報] (6)SPID“OsApplicationSPID” の説明文章を以下のように変更。 I/O 領域に対するアクセス権（SPID による保護）を指定します。 → I/O 領域に対するアクセス権（システム・プロテクション番号）を指定します。
		210	[B.4.7 OS 情報] “(4) スタック・モニタリング機能 “OsStackMonitoring”” の ” 備考 1.” を以下のように修正。 スタック・モニタリング機能についての詳細は，「3.1.2 スタック・モニタリング機能」，または「4.1.1 スタック・モニタリング機能」を参照してください。 → スタック・モニタリング機能についての詳細は，「3.1.2 スタック・モニタリング機能」，および「4.1.1 スタック・モニタリング機能」を参照してください。
		211	[B.4.7 OS 情報] 以下のように変更。 (11) スタック・サイズ “*OsStackSize*” → (11) システム・スタック・サイズ “*OsStackSize*”
		214	[B.4.8 リソース情報] “(2) シーリング値 “OsResourcePriority”” の ” 備考 1.” に以下の説明文章を追加。 本項目は省略可能で，省略時には備考 5. の通りにシーリング値を自動的に算出して割り当てます。スケジューラ・リソースを生成する場合以外は，本項目によるシーリング値の指定を省略し，その代わりにタスクが操作するリソース識別子 “OsTaskResourceRef” ，割り込みサービス・ルーチン（カテゴリ 2）が操作するリソース識別子 “OsIsrResourceRef” を指定することを推奨します。シーリング値についての詳細は，「5.1.1 シーリング値」を参照してください。

Rev.	発行日	改定内容	
		ページ	ポイント
1.02	2015.06.10	214	[B.4.8 リソース情報] “(2) シーリング値 “OsResourcePriority”” の ” 備考 3.” を以下のように修正。 識別子 “OsResource” に RES_SCHEDULER を定義した場合，OsResourcePriority には，どの初期優先度 “OsTaskPriority” よりも高い優先度を指定する必要があります。 → スケジューラ・リソースを生成する際には，識別子 “OsResource” に RES_SCHEDULER を定義し，OsResourcePriority には，優先度 29 を指定する必要があります。
		214	[B.4.8 リソース情報] “(2) シーリング値 “OsResourcePriority”” の ” 備考 5.” を以下のように修正。 AUTO を指定した場合，本リソースが所属している OS アプリケーションの全タスク／全割り込みサービス・ルーチンの中から最高位の優先度を検出し，該当値がシーリング値として割り付けられます。 → AUTO を指定した場合，本リソースを使用すると CF ファイル上で定義された全タスク／全割り込みサービス・ルーチンの中から最高位の優先度を算出し，該当値がシーリング値として割り付けられます。
		223	[B.4.10 タスク情報] 以下のように変更。 (5) スタック・サイズ “*OsTaskStackSize*” → (5) タスク・スタック・サイズ “*OsTaskStackSize*”
		241	[C.7.1 システム・スタック] ”(1)SC1 用システム・スタック ” の *SystemServiceStack_Max* の説明を以下のように変更。 拡張タスクからシステム・サービスの発行を行う場合は 164，行わない場合は 0 → 164（拡張タスクから発行したシステム・サービスが必要とするスタック・サイズ）
		242	[C.7.1 システム・スタック] Rev.1.01 の ”C.7.3 タスク・スタック（基本タスク）”，”C.7.5 割り込みサービス・ルーチン・スタック（カテゴリ 2）” にて記載していた下記の説明文章を ”(1)SC1 用システム・スタック ” に追加。 (a)SC1 における基本タスクのスタック・サイズ (b)SC1 における割り込みサービス・ルーチン（カテゴリ 2）のスタック・サイズ
		244	[C.7.2 OS アプリケーション・スタック] *SystemServiceStack_Max* の説明を以下のように変更。 拡張タスクからシステム・サービスの発行を行う場合は 164，行わない場合は 0 → 164（拡張タスクから発行したシステム・サービスが必要とするスタック・サイズ）

Rev.	発行日	改定内容	
		ページ	ポイント
1.02	2015.06.10	245	[C.7.2 OS アプリケーション・スタック] Rev.1.01 の ”C.7.3 タスク・スタック（基本タスク）”，”C.7.5 割り込みサービス・ルーチン・スタック（カテゴリ 2）” にて記載していた下記の説明文章を追加。 (a)SC3 における基本タスクのスタック・サイズ (b)SC3 における割り込みサービス・ルーチン（カテゴリ 2）のスタック・サイズ
		246	[C.7.3 タスク・スタック（拡張タスク）] “(1)SC1 用タスク・スタック（拡張タスク）” の *IsrContext_Siz* の説明を以下のように修正。 拡張タスクで割り込みが発生する場合は 124，発生しない場合は 0 → 124（拡張タスク実行中に発生した割り込み処理に必要なスタック・サイズ）
		246	[C.7.3 タスク・スタック（拡張タスク）] “(1)SC1 用タスク・スタック（拡張タスク）”*SystemServiceFrame_Siz* の説明を以下のように変更。 拡張タスクからシステム・サービスの発行を行う場合は 8，行わない場合は 0 → 8（拡張タスクから発行したシステム・サービスが必要とするスタック・サイズ）
		247	[C.7.3 タスク・スタック（拡張タスク）] “(2)SC3 用タスク・スタック（拡張タスク）” の *IsrContext_Siz* の説明を以下のように変更。 拡張タスクで割り込みが発生する場合は 128，発生しない場合は 0 → 128（拡張タスク実行中に発生した割り込み処理に必要なスタック・サイズ）
		247	[C.7.3 タスク・スタック（拡張タスク）] “(2)SC3 用タスク・スタック（拡張タスク）”*SystemServiceFrame_Siz* の説明を以下のように変更。 拡張タスクからシステム・サービスの発行を行う場合は 12，行わない場合は 0 → 12（拡張タスクから発行したシステム・サービスが必要とするスタック・サイズ）
		21	[3.1.4 タスクでの処理] “(2) FPSR の退避／復帰 ” に “ 備考 2.” を追加。
		27	[4.3.1 割り込みサービス・ルーチンでの処理] “(1) カテゴリ 1” に “ 備考 ” を追加。
		28	[4.3.1 割り込みサービス・ルーチンでの処理] “(2) カテゴリ 2” に “ 備考 2.” を追加。
		38	[8.2.1 アラーム・コールバックでの処理] “(2) FPSR の退避／復帰 ” に “ 備考 ” を追加。
		45	[10.2.1 Trusted 関数の処理] “(2) FPSR の退避／復帰 ” に “ 備考 2.” を追加。
		48	[10.3.1 固有フック・ルーチンでの処理] “(2) FPSR の退避／復帰 ” に “ 備考 ” を追加。
		53	[11.2.1 共通フック・ルーチンでの処理] “(2) FPSR の退避／復帰 ” に “ 備考 ” を追加。

Rev.	発行日	改定内容	
		ページ	ポイント
1.02	2015.06.10	133	[14.4.7 スケジュール・テーブル管理] “NextScheduleTable” の備考 3. を以下のように変更。 そこで，RV850 では，該当引数に指定可能なスケジュール・テーブルの周期属性 “OsScheduleTableRepeating” を周期属性を付与（TRUE）に限定し，該当周期属性以外のスケジュール・テーブルが指定された際には，エラー・ステータス E_OS_ID を返却しています。 → そこで，RV850 では，該当引数に指定可能なスケジュール・テーブルを，周期属性 “OsScheduleTableRepeating” を付与（TRUE）したものに限定し，該当引数に周期属性以外のスケジュール・テーブルが指定された際には，エラー・ステータス E_OS_ID を返却しています。
		229	[C.2 標準処理領域（.kernel_system）] V2.01.01 へのバージョン・アップ（カーネル・ライブラリ変更）に伴い，下記のカーネル・ライブラリ使用時のメモリ使用量を以下のように変更。 libecc2extsc1_fpu.a 12.3K バイト→ 12.4K バイト libecc2extsc3_g3k.a 21.5K バイト→ 21.6K バイト
		231	[C.4 定数データ領域（.kernel_const）] V2.01.01 へのバージョン・アップ（カーネル・ライブラリ変更）に伴い，”(1) SC1 用定数データ領域（.kernel_const）” のメモリ使用量計算式における固定値を以下のように変更。 674 → 673
		233	[C.4 定数データ領域（.kernel_const）] V2.01.01 へのバージョン・アップ（カーネル・ライブラリ変更）に伴い，”(2) SC3 用定数データ領域（.kernel_const）” のメモリ使用量計算式における固定値を以下のように変更。 1254 → 1253
		243	[C.7.1 システム・スタック] V2.01.01 へのバージョン・アップ（カーネル・ライブラリ変更）に伴い，”(2) SC3 用システム・スタック ” のメモリ使用量計算式における固定値を以下のように変更。 260 → 264
		-	[1.5 フォルダ構成] 「RV850 の展開先」を示すフォルダ名を以下のように変更。 <rx_root> → <rv_root>
		223	[B.4.10 タスク情報] “(5) タスク・スタック・サイズ “OsTaskStackSize”” の備考 2. 以下のように変更。 本項目で指定するサイズについての詳細は，「C.7.3 タスク・スタック（基本タスク）」，または，「C.7.4 タスク・スタック（拡張タスク）」を参照してください。 → 本項目で指定するサイズについての詳細は，「C.7.3 タスク・スタック（拡張タスク）」を参照してください。

Rev.	発行日	改定内容	
		ページ	ポイント
1.03	2015.12.17	10	[1.4 実行環境] 新規デバイス対応に伴い，”(1) デバイス ” の対象デバイスを以下のように修正 RH850 ファミリ（G3K コア，G3M コア） → RH850 ファミリ（G3K コア，G3M コア，G3KH コア，G3MH コア）
		10	[1.4 実行環境] 新規デバイス対応に伴い，”(2) 周辺コントローラ ” の備考 3. を以下のように修正 RV850 では，AUTOSAR 仕様のスケーラビリティ・クラス SC3（G3M 限定）で規定された周辺保護を実現する際 ... → RV850 では，AUTOSAR 仕様のスケーラビリティ・クラス SC3（G3M，G3KH，G3MH 限定）で規定された周辺保護を実現する際 ...
		10	[1.4 実行環境] 新規デバイス対応に伴い，” 表 1.1RV850 が占有している OS 予約資源 ” のマシン・コンフィギュレーション（MCFG0）の SPID ビットのスケーラビリティクラスを以下のように修正 SC3（G3M 限定） → SC3（G3M，G3KH，G3MH 限定）
		11	[1.5.1 オブジェクト・ファイル提供形式] 新規デバイス対応に伴い，”(9)” のカーネルライブラリ提供形式を以下のように修正 libecc2extsc3.a：　ECC2，拡張ステータス，SC3，FPU 対応なし，G3M 限定 libecc2extsc3_fpu.a：　ECC2，拡張ステータス，SC3，FPU 対応あり，G3M 限定 → libecc2extsc3.a：　ECC2，拡張ステータス，SC3，FPU 対応なし，G3M，G3KH，G3MH 限定 libecc2extsc3_fpu.a：　ECC2，拡張ステータス，SC3，FPU 対応あり，G3M，G3KH，G3MH 限定
		24	[4.2 ブート処理] 新規デバイス対応に伴い，”(2)” を以下のように修正 割り込み機能レジスタの初期化 → 割り込み機能レジスタの初期化 (G3K，G3M，G3KH 限定)
		24	[4.2 ブート処理] 新規デバイス対応に伴い，”(4)” を以下のように修正 マシン・コンフィギュレーション（MCFG0）の SPID ビット（システム・プロテクション番号）と連動した安全機能に関する初期化（G3M 限定） → マシン・コンフィギュレーション（MCFG0）の SPID ビット（システム・プロテクション番号）と連動した安全機能に関する初期化（G3M，G3KH，G3MH 限定）
		186	[B.1 概　　要] “(6) キーワード ” に G3KH，G3MH を追加。

Rev.	発行日	改定内容	
		ページ	ポイント
1.03	2015.12.17	196	[B.4.3 OS アプリケーション情報] 新規デバイス対応に伴い，”(6)SPID“OsApplicationSPID”” を以下のように修正 本項目は，コア識別子 “OsSystemCpuCore” で G3M を，信頼性 “OsTrusted” で FALSE を定義した場合に限り指定可能となります。 → 本項目は，コア識別子 “OsSystemCpuCore” で G3M，G3KH，G3MH のいずれかを指定し，かつ信頼性 “OsTrusted” で FALSE を定義した場合に限り指定可能となります。
		200	[B.4.3 OS アプリケーション情報] 新規デバイス対応に伴い， ”(17) メモリ領域識別子 “OsAppMemoryAreaNameRef”” を以下のように修正 本項目は，コア識別子 “OsSystemCpuCore” で G3M を，信頼性 “OsTrusted” で FALSE を定義した場合に限り指定可能となります。 → 本項目は，コア識別子 “OsSystemCpuCore” で G3M，G3KH，G3MH のいずれかを指定し，かつ信頼性 “OsTrusted” で FALSE を定義した場合に限り指定可能となります。
		213	[B.4.7 OS 情報] 新規デバイス対応に伴い，”(19) メモリ領域識別子 “OsMemoryAreaNameRef”” を以下のように修正 本項目は，以下のように複数の定義（コア識別子 “OsSystemCpuCore” で G3K を定義した場合：最大 3 個，コア識別子 “OsSystemCpuCore” で G3M を定義した場合：最大 7 個）が可能です → 本項目は，以下のように複数の定義（コア識別子 “OsSystemCpuCore” で G3K を定義した場合：最大 3 個，コア識別子 “OsSystemCpuCore” で G3M，G3KH，G3MH のいずれかを定義した場合：最大 7 個）が可能です
		227	[B.4.11 システム情報] 新規デバイス対応に伴い，”(4) コア識別子 “OsSystemCpuCore”” を以下のように修正 *OsSystemCpuCore* として指定格納な値は，G3K，または G3M に限られます。 → *OsSystemCpuCore* として指定格納な値は，G3K，G3M，G3KH，G3MH のいずれかに限られます。
1.04	2016.08.31	9	[1.2 特　　長] “(1)OSEK/VDX 仕様，AUTOSAR 仕様に準拠 ” の説明を以下のように変更。 …，および AUTOSAR 仕様（AUTOSAR Specification of Operating System Ver5.0.0 R4.0 Rev3）に準拠した設計が行われており，各種機能を提供しています。 → …，および AUTOSAR 仕様（AUTOSAR Specification of Operating System R4.2 Rev2）に準拠した設計が行われており，AUTOSAR 仕様における以下の機能を提供しています。 - スケーラビリティ・クラス SC1 でサポートする機能 - スケーラビリティ・クラス SC3 でサポートする機能 RV850 では，スケーラビリティ・クラス SC2 および SC4，マルチコア機能，IOC 機能はサポートしておりません。

Rev.	発行日	改定内容	
		ページ	ポイント
1.04	2016.08.31	10	[1.4 実行環境] “ 表 1.1” 中の “ メモリ保護機能（MPU）” の説明を以下のように変更。 メモリ保護機能（MPU） → メモリ保護機能（MPU）の領域 0 ～領域 3
		10	[1.4 実行環境] “ 表 1.1” に “ メモリ保護機能（MPU）の領域 4 ～領域 11　SC3（G3M，G3MH，G3KH 限定）” を追加。
		10	[1.4 実行環境] “ 表 1.1” に備考を追加。
		17	[2.3 処理プログラムの記述] 備考を備考 1. に変更し，備考 2. を追加。
		18	[2.5 リンカ・ディレクティブ・ファイルの記述] 備考 3 を備考 4. に変更，備考 4 を備考 5. に変更し，備考 3. を追加。
		22	[3.1.6 タスクの終了] “【スケーラビリティ・クラス SC3 の場合】” の説明を以下のように変更。 タスクが DisableAllInterrupts，SuspendAllInterrupts，SuspendOSInterrupts を発行したにも関わらず，対応する EnableAllInterrupts，ResumeAllInterrupts，ResumeOSInterrupts を非発行の場合，割り込みの受け付けを許可 （対応する EnableAllInterrupts，ResumeAllInterrupts，ResumeOSInterrupts と同等の処理） → タスクが DisableAllInterrupts，SuspendAllInterrupts，SuspendOSInterrupts を発行したにも関わらず，対応する EnableAllInterrupts，ResumeAllInterrupts，ResumeOSInterrupts を非発行の場合，割り込みの受け付けを許可
		25	[4.2 ブート処理] “(5) 周辺コントローラ（タイマなど）の初期化 ” の備考 6. を以下のように変更。 MCFG0 レジスタの SPID ビットが 0 の際には，I/O 領域に対する全アクセスが可能となる設定を行ってください。 SPID についての詳細は，対象デバイスのユーザーズ・マニュアルを参照してください。 → MCFG0 レジスタの SPID ビットが 0 の際には，I/O 領域に対する全アクセスが可能となる設定を行ってください。ただし，MCFG0 レジスタの SPID ビットのリセット後の値が 0 ではないデバイスの場合は，0 ではなくリセット後の値の際に，I/O 領域に対する全アクセスが可能となる設定を行ってください。（以降，このマニュアルで SPID ビットが 0 と説明している箇所については，同様に 0 をリセット後の値として読み替えてください。） SPID についての詳細は，対象デバイスのユーザーズ・マニュアルを参照してください。

Rev.	発行日	改定内容	
		ページ	ポイント
1.04	2016.08.31	27	[4.3.1 割り込みサービス・ルーチンでの処理] “(2) カテゴリ 2” の “(a) レジスタの退避／復帰 ” の説明を以下のように変更。 RV850 では，カテゴリ 2 の割り込みサービス・ルーチンに制御を移す際，C コンパイラの関数呼び出し規約に従った作業用レジスタの退避処理／復帰処理を行います。 したがって，“ レジスタの退避処理／復帰処理 ” は，記述する必要がありません。 → RV850 では，カテゴリ 2 の割り込みサービス・ルーチンに制御を移す際，C コンパイラの関数呼び出し規約に従った作業用レジスタの退避処理／復帰処理を行います。加えて，スケーラビリティ・クラスが SC3 の際には，TerminateApplication による OS アプリケーションの終了処理を考慮したレジスタの退避処理／復帰処理を行います。 したがって，“ レジスタの退避処理／復帰処理 ” は，記述する必要がありません。
		29	[4.3.3 割り込みサービス・ルーチンの終了] “(3) カテゴリ 2，スケーラビリティ・クラス SC3 の場合 ” の説明を以下のように変更。 割り込みサービス・ルーチンが DisableAllInterrupts，SuspendAllInterrupts などを発行したにも関わらず，対応する EnableAllInterrupts，ResumeAllInterrupts などを非発行の場合，割り込みの受け付けを許可（対応する EnableAllInterrupts，ResumeAllInterrupts と同等の処理） → 割り込みサービス・ルーチンが DisableAllInterrupts，SuspendAllInterrupts などを発行したにも関わらず，対応する EnableAllInterrupts，ResumeAllInterrupts などを非発行の場合，割り込みの受け付けを許可
		41	[10.1.1 信 頼 性] “(1)Trusted な OS アプリケーション ” に備考を追加。
		44	[10.2 Trusted 関数] 説明を以下のように変更。 Trusted 関数は，処理プログラムから CallTrustedFunction を発行することにより呼び出されます。CallTrustedFunction は他の Non-Trusted な OS アプリケーションからも呼び出すことが可能なので，Non-Trusted な OS アプリケーションから一時的に保護機能非適用の処理をさせたい場合に Trusted 関数を使用することができます。 → Trusted 関数は，処理プログラムから CallTrustedFunction および CallTrustedFunctionRestricted を発行することにより呼び出されます。CallTrustedFunction および CallTrustedFunctionRestricted は他の Non-Trusted な OS アプリケーションからも呼び出すことが可能なので，Non-Trusted な OS アプリケーションから一時的に保護機能非適用の処理をさせたい場合に Trusted 関数を使用することができます。
		44	[10.2 Trusted 関数] 備考 2. を以下のように変更。 Trusted 関数がアクセス可能なオブジェクト（タスク，リソースなど）は，CallTrustedFunction を発行した処理プログラムが所属している OS アプリケーションと同様です。 → Trusted 関数がアクセス可能なオブジェクト（タスク，リソースなど）は，Trusted 関数が所属している OS アプリケーションと同様です。

Rev.	発行日	改定内容	
		ページ	ポイント
1.04	2016.08.31	44	[10.2 Trusted 関数] 備考 3. を以下のように変更。 Trusted 関数はスーパバイザ・モードで動作するため，CallTrustedFunction を Non-Trusted な OS アプリケーションに所属している処理プログラムから発行した場合，モード切り替え処理（ユーザ・モードからスーパバイザ・モードへと移行），ならびにシステム・プロテクション番号（MCFG0 レジスタの SPID ビット）の切り替え（Trusted 関数実行中は 0）が行われます。 → Trusted 関数はスーパバイザ・モードで動作するため，CallTrustedFunction および CallTrustedFunctionRestricted を Non-Trusted な OS アプリケーションに所属している処理プログラムから発行した場合，モード切り替え処理（ユーザ・モードからスーパバイザ・モードへと移行），ならびにシステム・プロテクション番号（MCFG0 レジスタの SPID ビット）の切り替え（Trusted 関数実行中は 0）が行われます。
		44	[10.2 Trusted 関数] 備考 6. を以下のように変更。 AUTOSAR 仕様では，CallTrustedFunction を発行した処理プログラムから Trusted 関数へと制御を移す場合，カテゴリ 2 割り込みの受け付けを禁止する旨の規定が行われていますが，RV850 では，割り込みの受け付けに関する操作を行いません。 → AUTOSAR 仕様では，CallTrustedFunction を発行した処理プログラムから Trusted 関数へと制御を移す場合，カテゴリ 2 割り込みの受け付けを禁止する旨の規定が行われていますが，RV850 では，割り込みの受け付けに関する操作を行いません。 ただし，CallTrustedFunctionRestricted を発行した処理プログラムから Trusted 関数へと制御を移す場合は，RV850 は割り込みの受け付け禁止処理（PSW の ID ビットに対する操作）を行います。
		44	[10.2 Trusted 関数] 説明に備考 7. と備考 8. を追加。
		45	[10.2.1 Trusted 関数の処理] “(2)FPSR の退避／復帰 ” に備考 3. を追加。
		45	[10.2.1 Trusted 関数の処理] “(4) 割り込みの受け付け ” の説明を以下のように変更。 RV850 では，Trusted 関数に制御を移す際，割り込みの受け付けに関する操作を行いません。したがって，割り込みの受け付けを明示的に変更する場合，“EnableAllInterrupts，DisableAllInterrupts などの発行処理 ” を記述する必要があります。 → RV850 では，CallTrustedFunction を発行した処理プログラムから Trusted 関数に制御を移す際，割り込みの受け付けに関する操作を行いません。したがって，割り込みの受け付けを明示的に変更する場合，“EnableAllInterrupts，DisableAllInterrupts などの発行処理 ” を記述する必要があります。 ただし，CallTrustedFunctionRestricted を発行した処理プログラムから Trusted 関数に制御を移す場合は，RV850 は割り込みの受け付け禁止処理（PSW の ID ビットに対する操作）を行います。

Rev.	発行日	改定内容	
		ページ	ポイント
1.04	2016.08.31	45	[10.2.1 Trusted 関数の処理] “(5) システム・サービスの発行 ” の説明を以下のように変更。 RV850 では，Trusted 関数を CallTrustedFunction を発行した処理プログラムの延長線として位置付けているため，Trusted 関数で発行可能なシステム・サービスは，CallTrustedFunction を発行した処理プログラムの種類に依存します。 → RV850 では，Trusted 関数を CallTrustedFunction を発行した処理プログラムの延長線として位置付けているため，CallTrustedFunction の発行により呼び出された Trusted 関数で発行可能なシステム・サービスは，CallTrustedFunction を発行した処理プログラムの種類に依存します。 ただし，CallTrustedFunctionRestricted の発行により呼び出された Trusted 関数では，システム・サービスおよび RV850 が提供しているユーティリティ関数の発行を禁止します。システム・サービスおよび RV850 が提供しているユーティリティ関数を発行した場合，その後の動作を保証しません。
		52	[11.2 共通フック・ルーチン] “(6)ProtectionHook” の “ 備考 3.” の【PRO_TERMINATETASKISR（0x1）】のタスクの場合の説明を以下のように変更。 SUSPENDED 状態へと遷移 → 保護違反が発生した箇所でタスクの実行を強制的に終了し，SUSPENDED 状態へと遷移
		52	[11.2 共通フック・ルーチン] “(6)ProtectionHook” の “ 備考 3.” の【PRO_TERMINATETASKISR（0x1）】の割り込みサービス・ルーチンの場合の説明を以下のように変更。 ... スケジューラを起動 → 保護違反が発生した箇所で割り込みサービス・ルーチンの実行を強制的に終了し，スケジューラを起動

		56	[12.3 アイドル・ハンドラ]
			アイドル・ハンドラを C 言語で記述する際の基本型を以下のように変更。
			
			`void`
			`IdleHandler ( void ) {`
			`    ……`
			`    ……`
			`}`
			→
			`void`
			`IdleHandler ( IdleModeType idlemode ) {`
			`    ……`
			`    ……`
			`}`

		56	[12.3 アイドル・ハンドラ] 説明に備考 3. と備考 4. と備考 5. と備考 6. と備考 7. を追加。

Rev.	発行日	改定内容	
		ページ	ポイント
1.04	2016.08.31	59	[14.1 概　　要] “(8)OS アプリケーション管理 ” と “(9)OS 実行管理 ” を以下のように変更。 GetApplicationID，GetISRID，CallTrustedFunction，CheckISRMemoryAccess，CheckTaskMemoryAccess，CheckObjectAccess，CheckObjectOwnership，TerminateApplication，AllowAccess，GetApplicationState → GetApplicationID，GetCurrentApplicationID，GetISRID，CallTrustedFunction，CallTrustedFunctionRestricted，CheckISRMemoryAccess，CheckTaskMemoryAccess，CheckObjectAccess，CheckObjectOwnership，TerminateApplication，AllowAccess，GetApplicationState StartOS，ShutdownOS，GetActiveApplicationMode → StartOS，ShutdownOS，GetActiveApplicationMode，ControlIdle
		59	[14.1 概　　要] 備考を以下のように変更。 AUTOSAR 仕様では，ユーティリティ関数 InitApplicationInterrupts，_kernel_fv0_InitializeIntService，OSIllegalException_SystemRegister_ExcCode，OSIllegalException_SystemRegister_ExcPC の規定が行われていません。 該当ユーティリティ関数は，RV850 が独自に追加したユーティリティ関数です。 → AUTOSAR 仕様では，CallTrustedFunctionRestricted，ユーティリティ関数 InitApplicationInterrupts，_kernel_fv0_InitializeIntService，OSIllegalException_SystemRegister_ExcCode，OSIllegalException_SystemRegister_ExcPC の規定が行われていません。 CallTrustedFunctionRestricted および該当ユーティリティ関数は，RV850 が独自に追加したユーティリティ関数です。
		61	[14.2.1 データ・タイプ] “ 表 14.1 データ・タイプ ” の説明を以下のように変更。 MemoryStartAddressType　unsigned long　メモリ領域の先頭アドレス → MemoryStartAddressType　unsigned long *　メモリ領域の先頭アドレス
		61	[14.2.1 データ・タイプ] “ 表 14.1 データ・タイプ ” に以下のマクロを追加。 IdleModeType，CoreIdType
		64	[14.2.6 異常終了] 説明を以下のように変更。 以下に，GetApplicationID，GetISRID，CheckObjectOwnership からの戻り値（異常終了）に対応したマクロを示します。 → 以下に，GetApplicationID，GetCurrentApplicationID，GetISRID，CheckObjectOwnership からの戻り値（異常終了）に対応したマクロを示します。
		68	[14.2.12 システム・サービス識別子] “ 表 14.12 システム・サービス識別子 ” に以下のマクロを追加。 OSServiceId_ControlIdle，OSServiceId_GetCurrentApplicationID，OSServiceId_ExitTask，OSServiceId_ExitIsr

Rev.	発行日	改定内容	
		ページ	ポイント
1.04	2016.08.31	70	[14.2 データ・マクロ] 本節に以下の項を追加。 “14.2.15 アイドル・ハンドラの動作モード ”，“14.2.16 コアの識別子 ”
		82	[14.4.1 タスク管理] “GetTaskID” の備考 2. を以下のように変更。 獲得した数値とタスク情報の識別子 “OsTask” の対応は，コンフィギュレータが出力した SIT ファイルに定義されています。 → 本システム・サービスが戻り値として返すタスク識別子は数値であり，タスク情報の識別子 “OsTask” に指定された名前との対応は，コンフィギュレータが出力した SIT ファイルに定義されています。
		83	[14.4.1 タスク管理] “GetTaskState” の備考を備考 1. に変更し，備考 2. を追加。
		86	[14.4.2 割り込み管理] “EnableAllInterrupts” の備考 1. を以下のように変更。 本システム・サービスでは，割り込みの受け付け許可処理として，プログラム・ステータス・ワード（PSW）の ID ビットに対する操作を行います。 → 本システム・サービスでは，割り込みの受け付け許可処理として，プログラム・ステータス・ワード（PSW）の ID ビットに対する操作を行います。本システム・サービスを発行した際には，PSW の ID ビットを，DisableAllInterrupts 発行前の値に戻します。
		86	[14.4.2 割り込み管理] “EnableAllInterrupts” の備考 4. を以下のように変更。 本システム・サービスを SuspendAllInterrupts の発行から ResumeAllInterrupts が発行されるまでの間に発行した場合，何も処理は行われず，エラーとしても扱われません。 → 本システム・サービスを SuspendAllInterrupts の発行から ResumeAllInterrupts が発行されるまでの間に発行した場合，PSW の ID ビットに対する操作は行われません。
		86	[14.4.2 割り込み管理] “EnableAllInterrupts” の備考 5. を以下のように変更。 本システム・サービスをブート処理（StartOS の発行前）から，または ShutdownOS の発行後に発行した際には，割り込みの受け付け許可処理（PSW の ID ビットに対する操作）を強制的に行います。 本システム・サービスをブート処理（StartOS の発行前）から発行する場合，本システム・サービスの発行前に _kernel_fv0_InitializeIntService の発行が必要となります。 → 本システム・サービスをブート処理（StartOS の発行前）から発行する場合，本システム・サービスの発行前に _kernel_fv0_InitializeIntService の発行が必要となります。

Rev.	発行日	改定内容	
		ページ	ポイント
1.04	2016.08.31	87	[14.4.2 割り込み管理] “DisableAllInterrupts” の備考 4. を以下のように変更。 本システム・サービスを SuspendAllInterrupts の発行から ResumeAllInterrupts が発行されるまでの間に発行した場合，何も処理は行われず，エラーとしても扱われません。 → 本システム・サービスを SuspendAllInterrupts の発行から ResumeAllInterrupts が発行されるまでの間に発行した場合，PSW の ID ビットに対する操作は行われません。
		88	[14.4.2 割り込み管理] “DisableAllInterrupts” の備考 7. を以下のように変更。 本システム・サービスをブート処理（StartOS の発行前）から，または ShutdownOS の発行後に発行した際には，割り込みの受け付け禁止処理（PSW の ID ビットに対する操作）を強制的に行います。 本システム・サービスをブート処理（StartOS の発行前）から発行する場合，本システム・サービスの発行前に _kernel_fv0_InitializeIntService の発行が必要となります。 → 本システム・サービスをブート処理（StartOS の発行前）から発行する場合，本システム・サービスの発行前に _kernel_fv0_InitializeIntService の発行が必要となります。
		89	[14.4.2 割り込み管理] “ResumeAllInterrupts” の備考 1. を以下のように変更。 本システム・サービスでは，割り込みの受け付け許可処理として，プログラム・ステータス・ワード（PSW）の ID ビットに対する操作を行います。 → 本システム・サービスでは，割り込みの受け付け許可処理として，プログラム・ステータス・ワード（PSW）の ID ビットに対する操作を行います。本システム・サービスを発行した際には，PSW の ID ビットを，SuspendAllInterrupts 発行前の値に戻します。
		89	[14.4.2 割り込み管理] “ResumeAllInterrupts” の備考 4. を以下のように変更。 本システム・サービスを DisableAllInterrupts の発行から EnableAllInterrupts が発行されるまでの間に発行した場合，何も処理は行われず，エラーとしても扱われません。 → 本システム・サービスを DisableAllInterrupts の発行から EnableAllInterrupts が発行されるまでの間に発行した場合，PSW の ID ビットに対する操作は行われません。

Rev.	発行日	改定内容	
		ページ	ポイント
1.04	2016.08.31	89	[14.4.2 割り込み管理] “ResumeAllInterrupts” の備考 5. を以下のように変更。 本システム・サービスをブート処理（StartOS の発行前）から，または ShutdownOS の発行後に発行した際には，割り込みの受け付け許可処理（PSW の ID ビットに対する操作）のみを行い，禁止要求カウンタの減算処理（禁止要求カウンタから 0x1 を減算）は行いません。 本システム・サービスをブート処理（StartOS の発行前）から発行する場合，本システム・サービスの発行前に _kernel_fv0_InitializeIntService の発行が必要となります。 → 本システム・サービスをブート処理（StartOS の発行前）から発行する場合，本システム・サービスの発行前に _kernel_fv0_InitializeIntService の発行が必要となります。
		90	[14.4.2 割り込み管理] “SuspendAllInterrupts” の備考 4. を以下のように変更。 本システム・サービスを DisableAllInterrupts の発行から EnableAllInterrupts が発行されるまでの間に発行した場合，何も処理は行われず，エラーとしても扱われません。 → 本システム・サービスを DisableAllInterrupts の発行から EnableAllInterrupts が発行されるまでの間に発行した場合，PSW の ID ビットに対する操作は行われません。
		91	[14.4.2 割り込み管理] “SuspendAllInterrupts” の備考 7. を以下のように変更。 本システム・サービスをブート処理（StartOS の発行前）から，または ShutdownOS の発行後に発行した際には，割り込みの受け付け禁止処理（PSW の ID ビットに対する操作）のみを行い，禁止要求カウンタの加算処理（禁止要求カウンタに 0x1 を加算）は行いません。 本システム・サービスをブート処理（StartOS の発行前）から発行する場合，本システム・サービスの発行前に _kernel_fv0_InitializeIntService の発行が必要となります。 → 本システム・サービスをブート処理（StartOS の発行前）から発行する場合，本システム・サービスの発行前に _kernel_fv0_InitializeIntService の発行が必要となります。
		137	[14.4.8 OS アプリケーション管理] “ 表 14.24OS アプリケーション管理用システム・サービス ” に以下のシステム・サービスを追加。 GetCurrentApplicationID，CallTrustedFunctionRestricted
		137	[14.4.8 OS アプリケーション管理] “ 表 14.24OS アプリケーション管理用システム・サービス ” に備考を追加。
		138	[14.4.8 OS アプリケーション管理] “GetApplicationID” の備考 3. を備考 5. に変更し，説明を以下のように変更。 獲得した数値と識別子 “OsApplication” の対応は，コンフィギュレータが出力した SIT ファイルに定義されています。 → 本システム・サービスが戻り値として返す OS アプリケーション識別子は数値であり，識別子 “OsApplication” に指定された名前との対応は，コンフィギュレータが出力した SIT ファイルに定義されています。

Rev.	発行日	改定内容	
		ページ	ポイント
1.04	2016.08.31	138	[14.4.8 OS アプリケーション管理] “GetApplicationID” に備考 3. と備考 4. を追加。
		139	[14.4.8 OS アプリケーション管理] “GetCurrentApplicationID” を追加。
		140	[14.4.8 OS アプリケーション管理] “GetISRID” の備考 2. を以下のように変更。 獲得した数値と識別子 “OsIsr” の対応は，コンフィギュレータが出力した SIT ファイルに定義されています。 → 本システム・サービスが戻り値として返す割り込みサービス・ルーチン識別子は数値であり，識別子 “OsIsr” に指定された名前との対応は，コンフィギュレータが出力した SIT ファイルに定義されています。
		143	[14.4.8 OS アプリケーション管理] “CallTrustedFunctionRestricted” を追加。
		153	[14.4.8 OS アプリケーション管理] “TerminateApplication” の備考 2. の NO_RESTART（0x0）の説明に以下を追加。 - 対象 OS アプリケーションに所属している RUNNING 状態のタスクを強制的に終了 - 対象 OS アプリケーションに所属している実行中の割り込みサービス・ルーチンを強制的に終了 - 対象 OS アプリケーションに所属している，多重割り込みにより中断している割り込みサービス・ルーチンを強制的に終了
		155	[14.4.8 OS アプリケーション管理] “AllowAccess” の [機能] の説明を以下のように変更。 対象 OS アプリケーション（本システム・サービスを発行した処理プログラムの所属している OS アプリケーション）を APPLICATION_RESTATING 状態から APPLICATION_ACCESSIBLE 状態へと遷移させます。 → 対象 OS アプリケーション（本システム・サービスを発行した時に動作中の OS アプリケーション）を APPLICATION_RESTATING 状態から APPLICATION_ACCESSIBLE 状態へと遷移させます。
		156	[14.4.8 OS アプリケーション管理] “GetApplicationState” の [戻り値] から E_OS_CALLEVEL を削除。
		158	[14.4.9 OS 実行管理] “ 表 14.25OS 実行管理用システム・サービス ” に以下のシステム・サービスを追加。 ControlIdle
		160	[14.4.9 OS 実行管理] “StartOS” に備考 7. を追加。
		164	[14.4.9 OS 実行管理] “ControlIdle” を追加。

Rev.	発行日	改定内容	
		ページ	ポイント
1.04	2016.08.31	170	[14.4.10 ユーティリティ関数] “OSErrorGetServiceId” の備考 2. を以下のように変更。 本ユーティリティ関数を発行する処理プログラム（ErrorHook, ErrorHook_Application）の呼び出し要因が “ システム・サービスの異常終了 ” 以外の場合，本ユーティリティ関数の戻り値は不定値となります。 → 本ユーティリティ関数を発行する処理プログラム（ErrorHook, ErrorHook_Application）の呼び出し要因が “ アラームおよびスケジュール・テーブルの満了 ” の場合，本ユーティリティ関数の戻り値は OSServiceId_IncrementCounter となります。
		170	[14.4.10 ユーティリティ関数] “OSErrorGetServiceId” に備考 3. と備考 4. を追加。
		171	[14.4.10 ユーティリティ関数] “OSErrorGetServiceId” の [戻り値] に以下のマクロを追加。 OSServiceId_ControlIdle，OSServiceId_GetCurrentApplicationID, OSServiceId_ExitTask，OSServiceId_ExitISR
		174	[14.4.10 ユーティリティ関数] “OSError_SystemService_Parameter” の [指定形式] に以下のマクロを追加。 OSError_ControlIdle_CoreID，OSError_ControlIdle_IdleMode
		186	[B.1 概　　要] “(6) キーワード ” に TRUSTEDDELAYTIMINGVIOLATIONCALL，TRUSTED-WITHPROTECTION を追加。
		195	[B.4.3OS アプリケーション情報] 以下のように変更。 コア ID”OsApplicationCoreAssignment” → コア識別子 ”OsApplicationCoreRef”
		196	[B.4.3OS アプリケーション情報] 以下のように変更。 (5) コア ID”OsApplicationCoreAssignment” RV850 が制御対象とするコアの ID を指定します。 なお，*OsApplicationCoreAssignment* として指定可能な値は，0x1 に限られます。 備考　AUTOSAR 仕様では，本項目に指定可能な値を 0 ～ 65534 とする旨の規定が行われていますが，RV850 では，本項目に指定な値を 0x1 に限定しています。 【省略時解釈】 0x1 を指定したものとして処理が行われます。 → (5) コア識別子 ”OsApplicationCoreRef” RV850 が制御対象とするコア名を指定します。 なお，*OsApplicationCoreRef* として指定可能な値は，名前に限られます。 備考　RV850 では，本項目にいかなる値が指定されたとしても，RV850 の動作に影響しません。

Rev.	発行日	改定内容	
		ページ	ポイント
1.04	2016.08.31	201	[B.4.3OS アプリケーション情報] “ 時間保護違反の遅延発生 “OsTrustedApplicationDelayTimingViolationCall”” と “Trusted な OS アプリケーションの保護 “OsTrustedApplicationWithProtection”” を追加。

		214	[B.4.8 リソース情報]
			“ 図 B.11 リソース情報の記述形式 ” の ACCESSING_APPLICATION を以下のように修正。
			
			`ACCESSING_APPLICATION = OsResourceAccessingApplication;`
			→
			`[ ACCESSING_APPLICATION = OsResourceAccessingApplication; ]`

		215	[B.4.8 リソース情報] “(4)OS アプリケーション識別子 “OsResourceAccessingApplication”” の備考 1. を以下のように修正。 本項目は，スケーラビリティ・クラス “OsScalabilityClass” で SC3 を定義した場合に限り指定可能となります。 → 本項目は，スケーラビリティ・クラス “OsScalabilityClass” で SC3 を定義した場合に限り指定可能となります。種別 “OsResourceProperty” で STANDARD，および INTERNAL を指定した場合は，本項目は省略できません。
		215	[B.4.8 リソース情報] “(4)OS アプリケーション識別子 “OsResourceAccessingApplication”” に【省略時解釈】を追加。
		229	[C.2 標準処理領域（.kernel_system）] V2.02.00 へのバージョン・アップ（カーネル・ライブラリ変更）に伴い，下記のカーネル・ライブラリ使用時のメモリ使用量を以下のように変更。 libecc2extsc1.a 12.2K バイト→ 12.3K バイト libecc2extsc1_fpu.a 12.4K バイト→ 12.5K バイト libecc2extsc3.a 22.1K バイト→ 22.4K バイト libecc2extsc3_fpu.a 22.3K バイト→ 22.5K バイト libecc2extsc3_g3k.a 21.6K バイト→ 21.8K バイト
		230	[C.3 インタフェース領域（.kernel_interface）] V2.02.00 へのバージョン・アップ（カーネル・ライブラリ変更）に伴い，下記のカーネル・ライブラリ使用時のメモリ使用量を以下のように変更。 libecc2extsc1.a 0.85K バイト→ 0.89K バイト libecc2extsc1_fpu.a 0.85K バイト→ 0.89K バイト libecc2extsc3.a 0.92K バイト→ 0.94K バイト libecc2extsc3_fpu.a 0.92K バイト→ 0.94K バイト libecc2extsc3_g3k.a 0.92K バイト→ 0.94K バイト

Rev.	発行日	改定内容	
		ページ	ポイント
1.05	2016.11.30	27	[4.3.1 割り込みサービス・ルーチンでの処理] V2.02.00 へのバージョン・アップに伴い，カテゴリ 1ISR の割り込み受け付け方法に関する説明を以下のように修正 したがって，割り込みの受け付けを明示的に変更する場合，”EnableAllInterrupts, DisableAllInterrupts などの発行処理 ” を記述する必要があります。 → したがって，割り込みの受け付けを明示的に変更する場合，”EI 命令／ DI 命令の発行処理 ” を記述する必要があります。
		56	[12.3.1 アイドル・ハンドラでの処理] ”(4) 割り込みの受け付け ” の以下の備考を削除 備考　RV850 では，アイドル・ハンドラで割り込みの受け付けを禁止状態へと変更することが禁止されています。
		138	[14.4.8 OS アプリケーション管理] “GetApplicationID” の [機能] にある備考 4. を以下のように修正。 … 該当 Trusted 関数が所属する OS アプリケーションの識別子を戻り値として返します。 → … 該当 Trusted 関数を呼び出した処理プログラムが所属する OS アプリケーションの識別子を戻り値として返します。
		139	[14.4.8 OS アプリケーション管理] “GetCurrentApplicationID” の [機能] にある備考 4. を以下のように修正。 … 該当 Trusted 関数を呼び出した処理プログラムが所属する OS アプリケーションの識別子を戻り値として返します。 → … 該当 Trusted 関数が所属する OS アプリケーションの識別子を戻り値として返します。
		145	[14.4.8 OS アプリケーション管理] “CheckISRMemoryAccess” の [戻り値] にある T_u2_WRITEABLE に対する説明を以下のように修正。 読み込み可 → 書き込み可
1.20	2019.09.30	25	[4.2 ブート処理] (5) 周辺コントローラ（タイマなど）の初期化の備考 6. を以下のように変更。 MCFG0 レジスタの SPID ビットが 0 の際には，I/O 領域に対する全アクセスが可能となる設定を行ってください。ただし，MCFG0 レジスタの SPID ビットのリセット後の値が 0 ではないデバイスの場合は，0 ではなく，リセット後の値の際に I/O 領域に対する全アクセスが可能となる設定を行ってください。（以降，このマニュアルで SPID ビットが 0 と説明している箇所については，同様に 0 をリセット後の値として読み替えてください。） → MCFG0 レジスタの SPID ビットがシステム SPID 値 “OsSystemSystemSPID” で指定した値の際には，I/O 領域に対する全アクセスが可能となる設定を行ってください。

Rev.	発行日	改定内容	
		ページ	ポイント
1.20	2019.09.30	41	[10.1.1 信 頼 性] (1) に対する説明文書を以下のように変更。 およびシステム・プロテクション番号（MCFG0 レジスタの SPID ビット）を 0 に設定して動作させ，メモリ・アクセス保護機能，および周辺 I/O 保護機能を非適用としています。 → およびシステム・プロテクション番号（MCFG0 レジスタの SPID ビット）をシステム SPID 値 “OsSystemSystemSPID” で指定した値に設定して動作させ，メモリ・アクセス保護機能，および周辺 I/O 保護機能を非適用としています。
		44	[10.2 Trusted 関数] 備考 3. に対する説明文書を以下のように変更。 ならびにシステム・プロテクション番号（MCFG0 レジスタの SPID ビット）の切り替え（Trusted 関数実行中は 0）が行われます。 → ならびにシステム・プロテクション番号（MCFG0 レジスタの SPID ビット）の切り替え（Trusted 関数実行中はシステム SPID 値 “OsSystemSystemSPID” で指定した値）が行われます。
		44	[10.2 Trusted 関数] 備考 4. に対する説明文書を以下のように変更。 また，システム・プロテクション番号（MCFG0 レジスタの SPID ビット）が 0 で動作するため，システム・プロテクション番号と連動した周辺 I/O 保護機能は適用外となります。 → また，システム・プロテクション番号（MCFG0 レジスタの SPID ビット）がシステム SPID 値 “OsSystemSystemSPID” に指定した値で動作するため，システム・プロテクション番号と連動した周辺 I/O 保護機能は適用外となります。
		47	[10.3 固有フック・ルーチン] 備考に対する説明文書を以下のように変更。 ならびにシステム・プロテクション番号（MCFG0 レジスタの SPID ビット）が 0 で動作し，メモリ・アクセス保護機能，およびシステム・プロテクション番号と連動した周辺 I/O 保護機能は適用外となります。 → ならびにシステム・プロテクション番号（MCFG0 レジスタの SPID ビット）がシステム SPID 値 “OsSystemSystemSPID” に指定した値で動作し，メモリ・アクセス保護機能，およびシステム・プロテクション番号と連動した周辺 I/O 保護機能は適用外となります。
		50	[11.2 共通フック・ルーチン] 備考に対する説明文書を以下のように変更。 また，システム・プロテクション番号（MCFG0 レジスタの SPID ビット）が 0 で動作するため，システム・プロテクション番号と連動した周辺 I/O 保護機能は適用外となります。 → また，システム・プロテクション番号（MCFG0 レジスタの SPID ビット）がシステム SPID 値 “OsSystemSystemSPID” に指定した値で動作するため，システム・プロテクション番号と連動した周辺 I/O 保護機能は適用外となります。

Rev.	発行日	改定内容	
		ページ	ポイント
1.20	2019.09.30	56	[12.3 アイドル・ハンドラ] 備考 1. に対する説明文書を以下のように変更。 また，システム・プロテクション番号（MCFG0 レジスタの SPID ビット）が 0 で動作するため，システム・プロテクション番号と連動した周辺 I/O 保護機能は適用外となります。 → また，システム・プロテクション番号（MCFG0 レジスタの SPID ビット）がシステム SPID 値 “OsSystemSystemSPID” に指定した値で動作するため，システム・プロテクション番号と連動した周辺 I/O 保護機能は適用外となります。
		138	[GetApplicationID] 備考 2. に対する説明文書を以下のように変更。 本システム・サービスを共通フック・ルーチン（PostTaskHook，PreTaskHook，ErrorHook，ProtectionHook），固有フック・ルーチン（ErrorHook_*OsApplication*）から発行した場合，該当フック・ルーチンを呼び出した処理プログラムが所属している OS アプリケーションの識別子を戻り値として返します。 → 本システム・サービスを共通フック・ルーチン（PostTaskHook，PreTaskHook，ErrorHook，ProtectionHook），固有フック・ルーチン（StartupHook_*OsApplication*，ErrorHook_*OsApplication*，ShutdownHook_*OsApplication*）から発行した場合，該当フック・ルーチンを呼び出した処理プログラムが所属している OS アプリケーションの識別子を戻り値として返します。
		138	[GetApplicationID] 備考 3. に対する説明文書を以下のように変更。 本システム・サービスを共通フック・ルーチン（StartupHook，ShutdownHook），固有フック・ルーチン（StartupHook_*OsApplication*，ShutdownHook_*OsApplication*）から発行した場合，INVALID_OSAPPLICATION を戻り値として返します。 → 本システム・サービスを共通フック・ルーチン（StartupHook，ShutdownHook）から発行した場合，INVALID_OSAPPLICATION を戻り値として返します。
		139	[GetCurrentApplicationID] 備考 2. に対する説明文書を以下のように変更。 本システム・サービスを共通フック・ルーチン（PostTaskHook，PreTaskHook，ErrorHook，ProtectionHook），固有フック・ルーチン（ErrorHook_*OsApplication*）から発行した場合，該当フック・ルーチンを呼び出した処理プログラムが所属している OS アプリケーションの識別子を戻り値として返します。 → 本システム・サービスを共通フック・ルーチン（PostTaskHook，PreTaskHook，ErrorHook，ProtectionHook），固有フック・ルーチン（StartupHook_*OsApplication*，ErrorHook_*OsApplication*，ShutdownHook_*OsApplication*）から発行した場合，該当フック・ルーチンを呼び出した処理プログラムが所属している OS アプリケーションの識別子を戻り値として返します。

Rev.	発行日	改定内容	
		ページ	ポイント
1.20	2019.09.30	139	[GetCurrentApplicationID] 備考 3. に対する説明文書を以下のように変更。 本システム・サービスを共通フック・ルーチン（StartupHook, ShutdownHook），固有フック・ルーチン（StartupHook_*OsApplication*, ShutdownHook_*OsApplication*）から発行した場合，INVALID_OSAPPLICATION を戻り値として返します。 → 本システム・サービスを共通フック・ルーチン（StartupHook，ShutdownHook）から発行した場合，INVALID_OSAPPLICATION を戻り値として返します。
		186	[B.1 概　　要] (6) キーワードに SYSTEM_SPID を追加。
		196	[B.4.3 OS アプリケーション情報] (6)SPID“OsApplicationSPID” に対する説明文を以下のように変更。 I/O 領域に対するアクセス権（システム・プロテクション番号）を指定します。 なお，*OsApplicationSPID* として指定可能な値は，0x0 ～ 0x3 に限られます。 → I/O 領域に対するアクセス権（システム・プロテクション番号）を指定します。 なお，*OsApplicationSPID* として指定可能な値は，0x0 ～ 0x1F に限られます。
		226	[B.4.11 システム情報] (3) システム SPID 値 “OsSystemSystemSPID” を追加。

RV850 ユーザーズマニュアル
機能編

発行年月日　2013年 2月 15日　Rev.1.00
2019年 9月 30日　Rev.1.20

発行　ルネサス エレクトロニクス株式会社
〒135-0061 東京都江東区豊洲3-2-24(豊洲フォレシア)

RV850

RENESAS

ルネサス エレクトロニクス株式会社

R20UT2768JJ0120