お客様各位

カタログ等資料中の旧社名の扱いについて

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

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

2010年4月1日
ルネサスエレクトロニクス株式会社

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

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

ご注意書き

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

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

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

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

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

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

ユーザーズ・マニュアル

RX850 Pro Ver.3.30

リアルタイム・オペレーティング・システム

コーディング編（CubeSuite）

対象ツール

RX850 Pro Ver.3.30

資料番号　U19429JJ1V0UM00（第1版）
発行年月　December 2008

〔メ　モ〕

目次要約

WindowsおよびWindows Vistaは，米国Microsoft Corporationの米国およびその他の国における登録商標または商標です。

TRON は，"The Real-time Operating system Nucleus" の略称です。

ITRON は，"Industrial TRON" の略称です。

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

TRON, ITRON, およびμITRON は，特定の商品ないし商品群を指すものではありません。

μITRON4.0 仕様は，(社)トロン協会が策定したオープンなリアルタイムカーネル仕様です。

μITRON4.0 仕様の仕様書は，(社)トロン協会Web サイト（http://www.assoc.tron.org/）から入手が可能です。

μITRON仕様の著作権は(社)トロン協会に属しています。

- **本資料に記載されている内容は2008年12月現在のもので，今後，予告なく変更することがあります。**
  量産設計の際には最新の個別データ・シート等をご参照ください。
- 文書による当社の事前の承諾なしに本資料の転載複製を禁じます。当社は，本資料の誤りに関し，一切その責を負いません。
- 当社は，本資料に記載された当社製品の使用に関連し発生した第三者の特許権，著作権その他の知的財産権の侵害等に関し，一切その責を負いません。当社は，本資料に基づき当社または第三者の特許権，著作権その他の知的財産権を何ら許諾するものではありません。
- 本資料に記載された回路，ソフトウエアおよびこれらに関連する情報は，半導体製品の動作例，応用例を説明するものです。お客様の機器の設計において，回路，ソフトウエアおよびこれらに関連する情報を使用する場合には，お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し，当社は，一切その責を負いません。
- 当社は，当社製品の品質，信頼性の向上に努めておりますが，当社製品の不具合が完全に発生しないことを保証するものではありません。また，当社製品は耐放射線設計については行っておりません。当社製品をお客様の機器にご使用の際には，当社製品の不具合の結果として，生命，身体および財産に対する損害や社会的損害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を行ってください。
- 当社は，当社製品の品質水準を「標準水準」，「特別水準」およびお客様に品質保証プログラムを指定していただく「特定水準」に分類しております。また，各品質水準は，以下に示す用途に製品が使われることを意図しておりますので，当社製品の品質水準をご確認ください。

  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

  特別水準：輸送機器（自動車，電車，船舶等），交通用信号機器，防災・防犯装置，各種安全装置，生命維持を目的として設計されていない医療機器

  特定水準：航空機器，航空宇宙機器，海底中継機器，原子力制御システム，生命維持のための医療機器，生命維持のための装置またはシステム等

当社製品のデータ・シート，データ・ブック等の資料で特に品質水準の表示がない場合は，標準水準製品であることを表します。意図されていない用途で当社製品の使用をお客様が希望する場合には，事前に当社販売窓口までお問い合わせください。

（注）

（1）本事項において使用されている「当社」とは，NECエレクトロニクス株式会社およびNECエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいう。

（2）本事項において使用されている「当社製品」とは，（1）において定義された当社の開発，製造製品をいう。

M8E0710J

〔メ　モ〕

はじめに

対 象 者　このマニュアルは，V850マイクロコントローラ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，RX850 Pro のコーディング機能とその操作方法を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。

- 概　　説
- システム構築
- ニュークリアス
- タスク管理機能
- 同期通信機能
- 割り込み処理管理機能
- メモリ・プール管理機能
- 時間管理機能
- スケジューラ
- システム初期化処理
- インタフェース・ライブラリ
- システム・コール
- システム・コンフィギュレーション・ファイル
- コンフィギュレータCF850 Pro

読 み 方　このマニュアルの読者には，電気，論理回路，マイクロコンピュータ，C言語，アセンブリ言語に関する一般知識が必要です。

V850マイクロコントローラのハードウエア機能，命令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	：本文中につけた注の説明
注意	：気をつけて読んでいただきたい内容
備考	：本文の補足説明
数の表記	：2進数 … XXXXまたはB’XXXX
	10進数 … XXXX
	16進数 … 0xXXXXまたはH’XXXX

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
G（ギガ）　：2^{30}＝1024^3

関連資料　**このマニュアルを使用する場合は，次の資料もあわせてご覧ください。**
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。

開発ツールに関する資料（ユーザーズ・マニュアル）

資　料　名		資料番号	
		和　文	英　文
RX**シリーズ**	**起動編**(CubeSuite)	U19428J	U19428E
	メッセージ編(CubeSuite)	U19433J	U19433E
RX850 Pro Ver.3.30	**コーディング編**(CubeSuite)	**このマニュアル**	U19429E
	デバッグ編(CubeSuite)	U19431J	U19431E
	解析編(CubeSuite)	U19432J	U19432E
	内部構造編(CubeSuite)	U19434J	U19434E
CubeSuite**統合開発環境**	**起動編**	U19549J	U19549E
	プログラミング編	U19390J	U19390E
	メッセージ編	U19550J	U19550E
	V850**コーディング編**	U19383J	U19383E
	V850**ビルド編**	U19386J	U19386E
	V850**デバッグ編**	U19389J	U19389E
	V850**設計編**	U19380J	U19380E

目　　次

図の目次

表の目次

第 1 章　概　　説

マイクロプロセッサは半導体技術の進歩にしたがって急速に普及し，今日では，あらゆる分野で利用されるようになってきました。しかし，マイクロプロセッサを取り巻く処理プログラム量は増大し，各種ハードウエアにあわせた固有のプログラムをそのつど作成することが困難となりました。

そこで，高性能，多機能化へと進むマイクロプロセッサの能力を完全に引き出すために，オペレーティング・システム（Operating System：OS）の重要性が高まってきました。

厳密な分け方ではありませんが，OS にはプログラム開発用と制御用の 2 種類があります。プログラム開発用の OS は，開発に使用するハードウエア構成をある程度固定（パーソナル・コンピュータなど）できるため，標準的な OS（Windows® など）が流通しやすい環境にあります。

これに対し，制御用の OS は，制御機器に組み込まれて使用します。つまり，各々のシステムによってハードウエア構成が異なり，しかも，用途に応じた効率の良い動作が要求されるため，標準的な OS が流通しにくい環境にあります。

NEC エレクトロニクスでは，V850 マイクロコントローラを開発，発売し，より強力なマイクロプロセッサを提供する一方で，このような市場状況を考慮し，高機能なマイクロプロセッサが持つ機能を十分に引き出すため，また，将来にわたっての体系的なソフトウエアの構築を支援するために，RX850 Pro を開発，発売しました。

RX850 Pro は高性能，高機能なマイクロプロセッサの応用範囲を拡大し，いっそうの汎用性を持たせるために開発されたリアルタイム，マルチタスク処理を実現する制御用 OS です。

1.1　概　　要

RX850 Pro は，効率の良いリアルタイム，マルチタスク処理環境を提供するとともに，対象プロセッサ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，組み込み型制御用リアルタイム・マルチタスク OS です。

また，ターゲット・システムに組み込んで使用することを前提として開発されているため，ROM 化を意識し，高速かつコンパクトな OS となっています。

1.1.1　リアルタイム OS

制御機器分野におけるシステムでは，内外の事象変化に対する即応性が要求されます。しかし，従来のシステムでは，このような要求を単純な割り込み処理で対処してきたため，制御機器が高性能化，多機能化するにつれ，単純な割り込み処理だけでの対処が難しくなってきています。

つまり，システムの複雑化，処理プログラム量の増大により，内外の事象変化に対する処理を，どのような順序で実行させるかを管理することが困難になってきたということです。

そこで，このような問題に対処するために考えられたのがリアルタイム OS です。

リアルタイム OS は，内外の事象変化に対して即応し，最適な処理プログラムを，最適な順序で実行させることを主な目的としています。

1.1.2　マルチタスク OS

OS の管理下で実行する処理プログラムの最小単位を，「タスク」と呼びます。また，1 つのプロセッサ上で複数のタスクを時間的に同時実行させることを，「マルチタスキング」と呼びます。

実際には，プロセッサ自体は，1 度に 1 つの処理プログラム（命令）しか実行することができません。しかし，複数のタスクの実行を何らかの基準（きっかけ）を利用して切り替えることにより，あたかも複数のタスクが同時実行しているかのようになります。

このように，システム内で定めた何らかの基準を利用してタスクを切り替え，タスクの並列処理を可能にした OS がマルチタスク OS です。

マルチタスク OS は，タスクを並列に実行させることにより，システム全体の処理能力を向上させることを主な目的としています。

1.2　適用分野

RX850 Pro は，次のような装置に適しています。

- モータ制御を利用したシステム
  例）　PPC，プリンタ，FAX
- 低消費電力が必要なシステム
  例）　携帯電話，PHS，ディジタル・スチル・カメラ

第 2 章　システム構築

この章では，RX850 Pro を使用したアプリケーション・システムの構築手順について説明します。

2.1　概　　要

システム構築とは，RX850 Pro の提供媒体からユーザの開発環境（ホスト・マシン）上に転送したファイル群を用いて，ロード・モジュールを作成したあと，ターゲット・システムへ組み込むことです。
次に，システムを構築する際の手順を示します。

1）システム・コンフィギュレーション・ファイルの作成

2）システム初期化部の作成
- ブート処理
- ハードウエア初期化部
- 初期化ハンドラ
- 割り込みエントリ

3）処理プログラムの作成
- タ ス ク
- 直接起動割り込みハンドラ
- 間接起動割り込みハンドラ
- 周期起動ハンドラ
- 拡張 SVC ハンドラ

備考　　処理プログラムは，C 言語やアセンブリ言語を用いて作成します。

4）初期化データの退避領域の作成

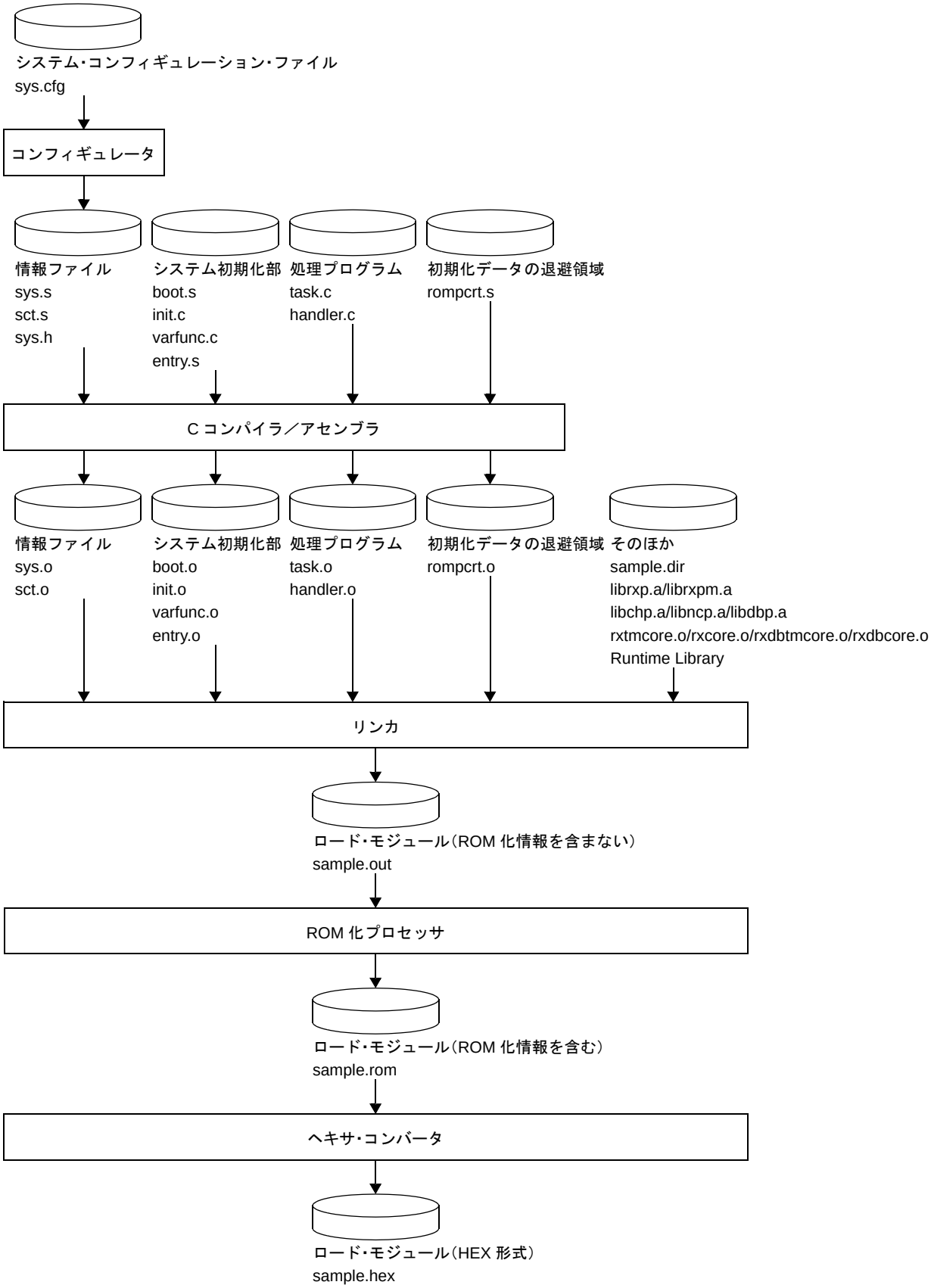
5）リンク・ディレクティブ・ファイルの作成

6）ロード・モジュールの作成

7）システムへの組み込み

図 2 － 1 に，システム構築手順の例を示します。

図 2 － 1　システム構築手順



次に，CA850 に添付されているサンプル・プログラムを基に，システム構築の流れを説明します。
なお，サンプル・プログラムの格納フォルダは，RX850 Pro のインストール・フォルダが <rx_root> とした場合，次のようになります。

<rx_root>¥smp850e¥rx85p¥src

2.2　システム・コンフィギュレーション・ファイルの作成

RX850 Pro で使用する各種データを保持した「システム・コンフィギュレーション・ファイル」という情報テーブルを作成します。
このファイルは，CF850 Pro を使用し，次のものを作成するために必要となります。

- システム情報テーブル・ファイル
- システム・コール・テーブル・ファイル
- システム情報ヘッダ・ファイル

システム情報テーブル・ファイルは，タスク，セマフォ，メモリ・プール等の RX850 Pro の資源情報，システム・コール・テーブル・ファイルは，アプリケーションで使用しているシステム・コールの一覧を保持するテーブルです。システム情報ヘッダ・ファイルは，システム情報テーブル・ファイルで作成したタスク，セマフォ等，資源 ID として指定されたシンボル名と実際の ID 番号を，#define 命令で対応させる記述があります。
サンプルのシステム・コンフィギュレーション・ファイルは

- sys.cfg

です。システム・コンフィギュレーション・ファイルの内容と書式については，「第 13 章　システム・コンフィギュレーション・ファイル」を参照してください。

2.3　システム初期化部の作成

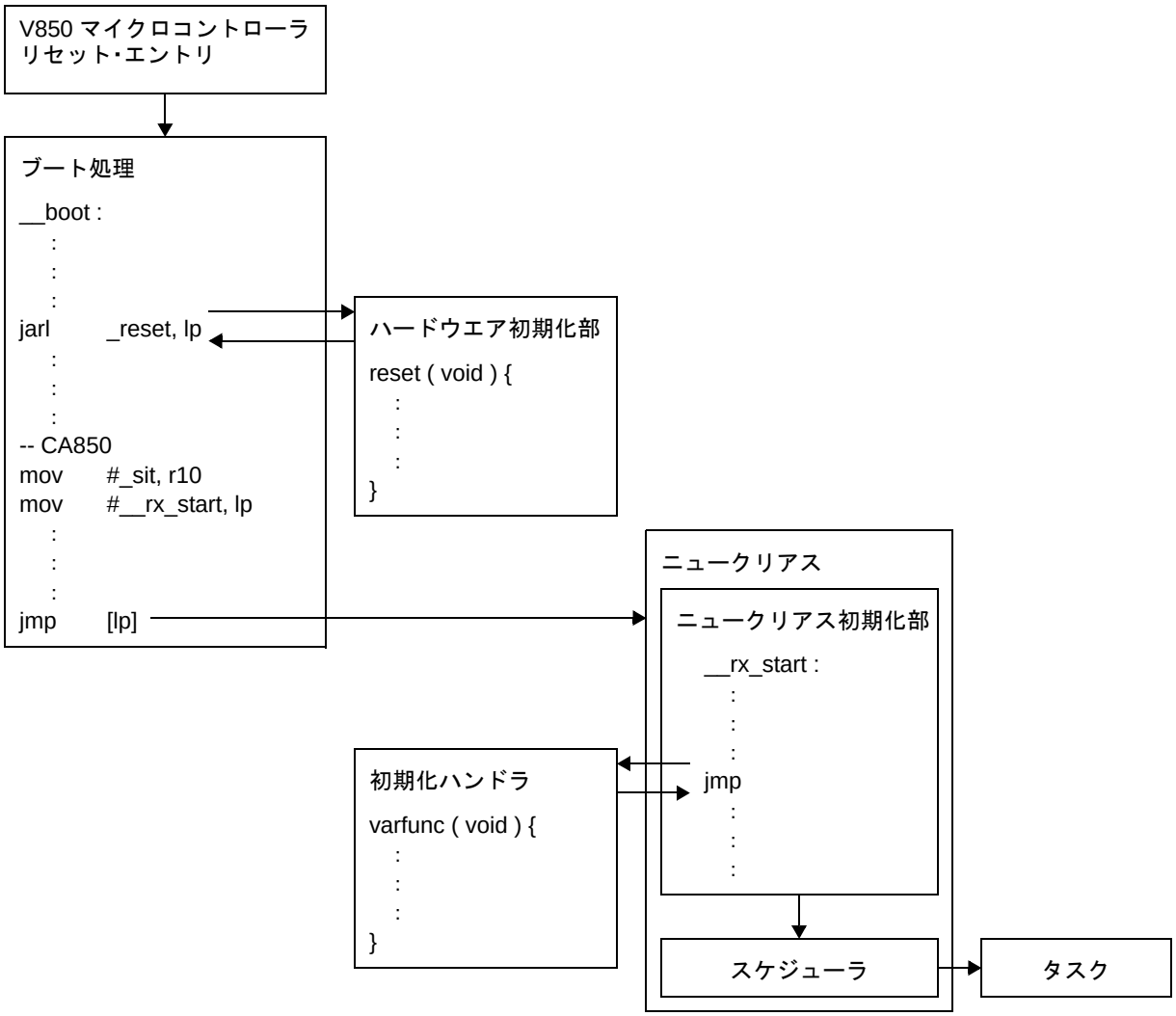
システム初期化部とは，ユーザのターゲット・システムに依存する部分を切り出し，移植性の向上をはかるとともに，カスタマイズを容易にするために用意された関数です。
システム初期化部の構成は，次のとおりです。

表 2 － 1　システム初期化部の構成

サンプル・ファイル名	種別	関数名	機能
boot.s	ブート処理	_boot	システムのブート処理
init.c	ハードウエア初期化部	reset	ハードウエアの初期化処理
varfunc.c	初期化ハンドラ	varfunc	ソフトウエアの初期化処理
entry.s	割り込みエントリ	無し	割り込み処理への分岐処理

また，システム初期化部のおおまかな流れは次のようになります。

図 2 － 2　システム初期化部の流れ



2.3.1　ブート処理

ブート処理は，V850 マイクロコントローラのリセット・エントリ（ハンドラ・アドレス：0x0 番地）に割りつけられ，システム初期化処理の中で最初に実行されます。
サンプルでは，boot.s 内の _boot というラベル以降が，ブート処理の本体になります。リセット・エントリからこのラベルにジャンプさせている部分は，entry.s 内にある次の命令に当たります。

```
.section    "RESET"
.extern     __boot

mov         #__boot, lp
jmp         [lp]
```

それぞれ下 2 行分が，ハンドラ・アドレス「0x0 番地」に割りつけられます。つまり，リセットされると，この命令が実行され，_boot へジャンプしてブート処理が実行されます。
ブート処理内では，次のことをする必要があります。

1） ブート処理内で使う sp（スタック・ポインタ）の設定
2） tp（テキスト・ポインタ），gp（グローバル・ポインタ）の設定
3） _sit シンボルの r10 アドレスに設定
4） __rx_start シンボルを lp レジスタに設定
5） jmp 命令を発行して，ニュークリアス初期化部に制御を移行

このほか，サンプルでは，ハードウエア初期化部 reset にジャンプする処理（jarl _reset, lp）が 3）と 4）の間にあります。
1）で設定するスタック・ポインタは，タスクや割り込みハンドラのスタックとは無関係です。RX850 Pro 起動後は，タスクや割り込みハンドラ等が使用するスタックは，システム情報テーブル・ファイルにより RX850 Pro 自身が管理しており，タスク・スイッチや割り込みにより自動的にスタック・ポインタを切り替えます。
つまり，ブート処理内で指定するスタック・ポインタは RX850 Pro 起動前に使用するもので，たとえば関数にジャンプし，その関数内でスタックに保存すべきデータがある場合等に使用されます。サンプルの reset 関数でスタックを使用する必要があるときには，このスタック・ポインタが使用されます。
また，サンプルでは，スタック・サイズとして 0x28000 バイトを取っていますが，これほど必要ないことがほとんどです。ただ，この領域は RX850 Pro で使用されるシステム・メモリ領域「システム・メモリ情報」で定義されたサイズにあわせてあり，RX850 Pro 起動後は，その領域として使用されるようになっています。
また，サンプルではブート処理内で RAM 上の bss 領域の初期化（0 クリア）が行われています。
なお，「初期値データのコピー処理」は，初期値データの退避領域の作成 rompcrt.s と _rcopy 関数の使用によって行われます。詳細については，「CubeSuite V850 ビルド編」のユーザーズ・マニュアルを参照してください。
そしてブート処理の最後では，4）から 6）の処理が必要になります。次の処理を行ってください。

```
.extern     _sit
mov         #_sit, r10

.extern     __rx_start
mov         #__rx_start, lp
jmp         [lp]
```

RX850 Pro 内部にあるシンボル __rx_start 以降が RX850 Pro のニュークリアス初期化処理に当たります。ブート処理が完了した際は jmp 命令によってニュークリアス初期化処理に移行します。その際に r10 レジスタに _sit シンボルのアドレスを代入してから移行してください。初期化処理では，この r10 レジスタに代入されたアドレスと，システム・コンフィギュレーション・ファイルから作ったシステム情報テーブル・ファイルを基に，資源生成や初期化を行っているためです。
ブート処理の記述は，サンプルのブート処理を基に，ユーザにあわせた環境へ変更していくことを推奨します。

2.3.2　ハードウエア初期化部

ハードウエア初期化部は，サンプルでは，ブート処理から呼び出される関数です。ブート処理の一環としてターゲット・システム上のハードウエアの初期化を行うために用意しているものです。

サンプルでは，reset 関数が，このハードウエア初期化処理を行っています。ブート処理から，この reset 関数を呼び出していますが，ハードウエア初期化処理が特に必要ない場合，または，ほかで行うという場合は，この関数を使用しなくても問題ありません。

サンプルのハードウエア初期化部では，次のような処理を行っています。

1）割り込みコントローラ，クロック・コントローラの初期化
2）周辺 I/O レジスタ，コントローラの初期化
3）ブート処理へ制御を戻す

2.3.3　ニュークリアス初期化部

ニュークリアス初期化部は，ブート処理終了後に実行される RX850 Pro 内部のルーチンです。ここでは，システム・コンフィギュレーション・ファイルより作成されたシステム情報テーブル・ファイルより，RX850 Pro のシステム管理ブロックの作成，タスクやセマフォ，メモリ・プール等の情報を作成，初期化を行っています。

RX850 Pro では，このニュークリアス初期化部において，初期化が正常に行われなかった場合，CPU を HALT 状態にします。ブート処理から初期化処理にジャンプしたあと，RX850 Pro が起動せずに HALT 状態になる場合は，RX850 Pro の管理ブロック作成等に使用するシステム・メモリ領域「システム・メモリ情報」が不足していることが考えられますので，メモリが十分に確保されているか確認してください。

ニュークリアス初期化部において初期化が終了すると，初期化ハンドラが呼び出されます。これは，「初期化ハンドラ情報」で指定されたもので，サンプルでは，varfunc 関数となっています。この関数については，「2.3.4　初期化ハンドラ」を参照してください。

初期化ハンドラから制御が戻ると，スケジューラが起動され，RX850 Pro が起動します。

2.3.4　初期化ハンドラ

初期化ハンドラは，ニュークリアス初期化部から呼び出される関数です。ここでは，RX850 Pro 起動直前にやっておきたい処理を記述します。

初期化ハンドラは，初期化ハンドラ情報で指定された関数を呼び出します。サンプルでは，varfunc 関数となっています。この関数は必ず指定する必要がありますので，特に必要ない場合でも，何もしない関数として生成してください。また，ハンドラの最後では，return 命令によってニュークリアス初期化処理へ戻してください。

なお，初期値データのコピー処理を，この初期化ハンドラで行うことも可能です。初期値データのコピー処理については，「CubeSuite V850 ビルド編」のユーザーズ・マニュアルを参照してください。

備考 1　RX850 Pro では，ニュークリアス初期化部から本処理部に制御を移す際，スタックをコンフィギュレーション時にシステム情報で指定されたシステム・スタックへの切り替え処理を実行しています。

備考 2　RX850 Pro では，ニュークリアス初期化部から本処理部に制御を移す際，tp（テキスト・ポインタ），gp（グローバル・ポインタ）をコンフィギュレーション時に初期化ハンドラ情報で定義された値への切り替え処理を実行しています。

備考 3　RX850 Pro では，ep（エレメント・ポインタ）に対する操作を行いません。したがって，本処理部を実行する際の ep は，ブート処理で設定された値となります。

備考 4　本処理部は，RX850 Pro の初期化処理がすべて完了する以前に呼び出されます。したがって，本処理部を割り込み許可状態とした場合，および，本処理部でシステム・コールを発行した場合，動作の保証はありません。

2.3.5　割り込みエントリ

割り込みエントリは，割り込みが発生した際に実行される命令を記述したもので，V850 マイクロコントローラが持つ割り込みハンドラ・アドレスに割りつけられます。ユーザが使用するすべての割り込みに対し，割り込みエントリを定義する必要があります。これらはアセンブリ言語で記述する必要があり，サンプルでは entry.s に記述されています。

RX850 Pro の割り込み処理には，直接起動割り込みハンドラ，間接起動割り込みハンドラの 2 種類があり，それぞれエントリの記述方法が違います。

直接起動割り込みハンドラの場合，通常の割り込みエントリと同様に分岐命令を記述します。サンプル・プログラム (V850ES/V850E1/V850E2 コア) では，割り込み INTP110（ハンドラ・アドレス：0x180 番地）を直接起動割り込みハンドラの例としています。

section 疑似命令を使用します。命令については，「CubeSuite V850 コーディング編」のユーザーズ・マニュアルを参照してください。直接起動割り込みハンドラのエントリは，次のようになります。

```
.section    "INTP110"
jr          _intp110_entry
```

なお，飛び先のラベル _intp110_entry も同じファイルで定義されており，直接起動割り込みハンドラの前処理，後処理の記述（マクロ記述）をしたあと，ハンドラ本体 intp130 へジャンプしています。

間接起動割り込みハンドラの場合，RX850 Pro で用意しているマクロを使用します。つまり，マクロ内容をハンドラ・アドレスに割りつける必要があります。マクロ名は RTOS_IntEntry_Indirect です。サンプル・プログラム (V850ES/V850E1/V850E2 コア) では，割り込み INTP120（ハンドラ・アドレス：0x1c0 番地）を間接起動割り込みハンドラの例としています。

.section 疑似命令を使用します。命令については，「CubeSuite V850 コーディング編」のユーザーズ・マニュアルを参照してください。間接起動割り込みハンドラのエントリは，次のようになります。

```
.section     "INTP120"
RTOS_IntEntry_Indirect
```

また，RX850 Pro で使用するクロック割り込みに関しても，間接起動割り込みハンドラと同じように割り込みエントリを登録する必要があります。サンプル・プログラム（V850ES/V850E1/V850E2 コア）では，INTCMD0（ハンドラ・アドレス：0x240 番地）をクロック割り込みとして使用しているので，次のように記述します。

```
.section    "INTCMD0"
RTOS_IntEntry_Indirect
```

備考　間接起動割り込みハンドラ情報で定義された割り込みハンドラ，および，システム情報で定義されたタイマの割り込み要因番号に対応したクロック・ハンドラについては，コンフィギュレータが該当割り込みエントリをシステム情報テーブルに自動出力しているため，ユーザが該当割り込みエントリを記述する必要がありません。
ただし，コンフィギュレータの起動オプションに -ne が指定された際には，システム情報テーブルに対する割り込みエントリの出力が抑制されます。

2.4　処理プログラムの作成

処理プログラム（アプリケーション本体）を作成します。
RX850 Pro で必要となるアプリケーションの処理単位は，大きく分けて次のようになります。

- タ ス ク
- 直接起動割り込みハンドラ
- 間接起動割り込みハンドラ
- 周期起動ハンドラ
- 拡張 SVC ハンドラ

サンプルでは，拡張 SVC ハンドラ以外の 4 つを使用しています。次の表がサンプルの内容です。

表 2 － 2　処理プログラムの構成

サンプル・ファイル名	種別	関数名	機能
task.c	タスク	task1 task2	タスク本体
handler.c	周期起動ハンドラ 間接起動割り込みハンドラ 直接起動割り込みハンドラ	cychdr1 cychdr2 inthdr1 inthdr2	各種ハンドラ処理

C 言語で記述された処理プログラムで，システム・コールを発行している場合には，RX850 Pro が提供しているヘッダ・ファイル stdrx85p.h をインクルードしてください。システム・コールを使用するうえで必要な定義が含まれています。また，サンプルにあるヘッダ・ファイル usr.h は，必要に応じて関数が使用する定数等を定義し，プログラム中でインクルードします。サンプルでは定数のマクロ定義のみです。

2.5　初期化データの退避領域の作成

この初期化データの退避領域を作成する必要があります。
これは，初期化データを ROM に格納しておき，プログラム実行前にその初期値を RAM にコピーする必要があるためです。格納元である ROM 領域を確保するのが，この初期化データの退避領域の作成に当たります。
この作成方法については，「CubeSuite V850 ビルド編」のユーザーズ・マニュアルの「ROM 化プロセッサ」の項を参照してください。

2.6　リンク・ディレクティブ・ファイルの作成

リンカがリンク時に参照するセクション情報，アドレス情報が記述されたリンク・ディレクティブ・ファイルを作成します。サンプルでは，次のファイルがこれに当たります。

- sample.dir

RX850 Pro において必須のセクションが存在し，それは次のようになっています。

表 2 － 3　RX850 Pro の必須セクション

セクション名	領域の種類	備考
.sit	システム情報領域	必須セクション
.system	RX850 Pro システム・コール配置領域	必須セクション
.system_cmn	RX850 Pro スケジューラ関連配置領域	必須セクション
.system_int	RX850 Pro 割り込み処理関連配置領域	必須セクション
.text	RX850 Pro インタフェース・ライブラリ配置領域	必須セクション
任意	システム･メモリ領域 System Memory Pool 0 配置領域	必須セクション
任意	システム･メモリ領域 System Memory Pool 1 配置領域	省略可能セクション
任意	システム・メモリ領域 User Memory Pool 0 配置領域	省略可能セクション
任意	システム・メモリ領域 User Memory Pool 1 配置領域	省略可能セクション

これらの属性は .sit が const 属性，システム・メモリ領域 4 つが bss 属性，残りは text 属性になっています。これらのセクション情報を，リンク・ディレクティブ・ファイルに定義する必要があります。サンプルのファイルに記述してあるように，これらは位置情報を定義します。次がサンプルの該当部分です。

```
     :
     :
CONST   : !LOAD ?R      V0x00002000 {
        .sit            = $PROGBITS ?A  .sit;
        .const          = $PROGBITS ?A  .const;
};
TEXT    :!LOAD       ?RX {
        .pro_epi_rutime = $PROGBITS ?AX .pro_epi_runtime;
        .system         = $PROGBITS ?AX .system;
        .system_cmn     = $PROGBITS ?AX .system_cmn;
        .system_int     = $PROGBITS ?AX .system_int;
        .text           = $PROGBITS ?AX .text;
};
S0MEMA  :!LOAD  ?RW     V0xffffc000 {
        .spol0area      = $NOBITS   ?AW .spol0area;
};
S1MEMA  :!LOAD  ?RW     V0xffffcc00 {
        .spol1area      = $NOBITS   ?AW .spol1area;
};
U0MEMA  :!LOAD  ?RW     V0xffffd200 {
        .upol0area      = $NOBITS   ?AW .upol0area;
};
     :
     :
```

そのほか，.data / .bss セクション等の RAM 領域に関するセクション，const セクション等の ROM 領域に関するセクションについて，必要なものも定義していきます。リンク・ディレクティブ・ファイルの記述は，サンプルを基に，ユーザにあわせた環境へ変更していくことを推奨します。

なお，リンク・ディレクティブ・ファイルの記述方法については，「CubeSuite V850 コーディング編」のユーザーズ・マニュアルを参照してください。

2.7　ロード・モジュールの作成

「2.2　システム・コンフィギュレーション・ファイルの作成」から「2.6　リンク・ディレクティブ・ファイルの作成」で作成されたファイル群，および，RX850 Pro，CA850 が提供しているライブラリ・ファイルに対して，CubeSuite 上でビルドを実行し，ロード・モジュールを生成します。

1）プロジェクトの作成／読み込み

プロジェクトの新規作成，または既存のプロジェクトの読み込みを行います。

備考　プロジェクトの新規作成，および既存のプロジェクトの読み込みについての詳細は，「RX シリーズ 起動編」，および「CubeSuite 起動編」のユーザーズ・マニュアルを参照してください。

2）ビルド対象プロジェクトの設定

ビルドの設定や実行を行う場合は，アクティブ・プロジェクトを設定します。
なお，サブプロジェクトがない場合，プロジェクトは常にアクティブになります。

備考　アクティブ・プロジェクトの設定についての詳細は，「CubeSuite V850 ビルド編」のユーザーズ・マニュアルを参照してください。

3）ビルド対象ファイルの設定

プロジェクトへのビルド対象ファイルの追加／削除，依存関係の更新などを行います。

備考　プロジェクトへのビルド対象ファイルの追加／削除，依存関係の更新についての詳細は，「CubeSuite V850 ビルド編」のユーザーズ・マニュアルを参照してください。

以下に，ロード・モジュール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。

- 「2.2　システム・コンフィギュレーション・ファイルの作成」で作成されたシステム・コンフィギュレーション・ファイル

  備考　システム・コンフィギュレーション・ファイル名の拡張子は，“cfg”を指定してください。
  拡張子が異なる場合は，“cfg”が自動的に付加されます（例えば，ファイル名に“aaa.c”を指定した場合，“aaa.c.cfg”となります）。

- 「2.3　システム初期化部の作成」で作成された C 言語／アセンブリ言語ソース・ファイル
- 「2.4　処理プログラムの作成」で作成された C 言語／アセンブリ言語ソース・ファイル
- 「2.5　初期化データの退避領域の作成」で作成された C 言語／アセンブリ言語ソース・ファイル
- 「2.6　リンク・ディレクティブ・ファイルの作成」で作成されたリンク・ディレクティブ・ファイル

備考 1　プロジェクト・ツリー パネルにシステム・コンフィギュレーション・ファイルを追加すると，リアルタイム OS 生成ファイル・ノードが表示されます。
リアルタイム OS 生成ファイル・ノードには，以下の情報ファイルが表示されます。ただし，この時点では，これらのファイルは生成されません。

- システム情報テーブル・ファイル
- システム情報ヘッダ・ファイル
- システム・コール・テーブル・ファイル

図 2 － 3　プロジェクト・ツリー パネル（sys.cfg 追加後）

備考 2　システム・コンフィギュレーション・ファイルを差し替える場合は，追加しているシステム・コンフィギュレーション・ファイルを一旦プロジェクトから外したのち，再度ファイルを追加してください。

備考 3　システム・コンフィギュレーション・ファイルは，プロジェクトに複数追加することができますが，有効となるのは最初に追加したファイルです。有効なファイルをプロジェクトから外しても，追加済みのファイルは有効にならないため，再度ファイルを追加してください。

4）ライブラリ，および，オブジェクトのリンク指定

RX850 Pro を使用したアプリケーションをリンクする際には，次のライブラリの参照，および，オブジェクトのリンクが必要になります。

- ニュークリアス・ライブラリ

librxp.a：　rel_blk の発行直前で対象メモリ・ブロックの先頭 4 バイトをゼロ・クリアする必要あり
librxpm.a：　rel_blk の発行直前で対象メモリ・ブロックの先頭 4 バイトをゼロ・クリアする必要なし

- インタフェース・ライブラリ

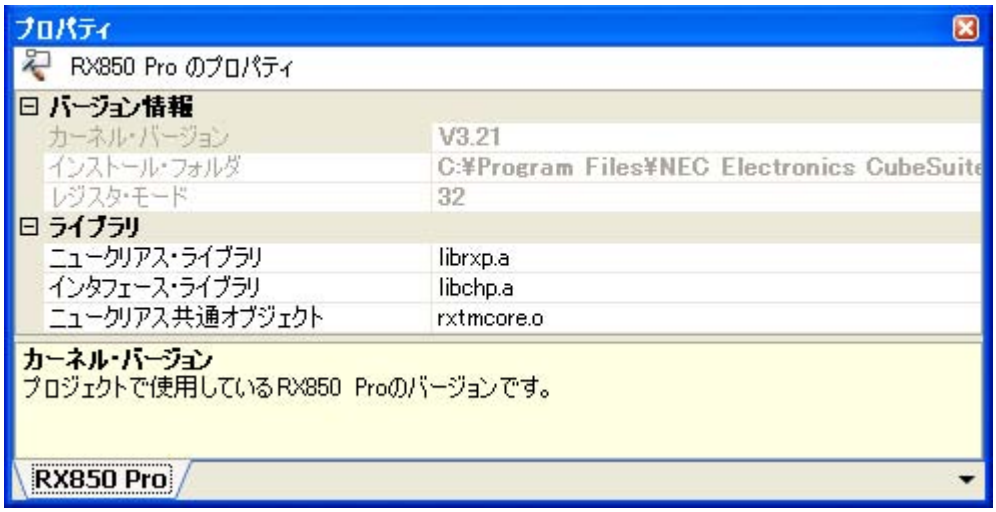
libchp.a：　パラメータ・チェック機能あり
libncp.a：　パラメータ・チェック機能なし

- ニュークリアス共通オブジェクト

rxtmcore.o：　周期起動ハンドラ内ではクロック割り込みよりも優先度の高いマスカブル割り込みを受け付け許可
rxcore.o：　周期起動ハンドラ内ではすべてのマスカブル割り込みを受け付け許可

プロジェクト・ツリーでリアルタイム OS ノードを選択し，プロパティ パネルの［RX850 Pro］タブをオープンします。
各プロパティにおいて，ニュークリアス・ライブラリ，インタフェース・ライブラリ，ニュークリアス共通オブジェクトをリンクすることを設定します。

図 2 － 4　プロパティ パネル：［RX850 Pro］タブ



ニュークリアス・ライブラリ，インタフェース・ライブラリ，ニュークリアス共通オブジェクトは複数の種類が用意されています。それぞれの用途に応じて使用するライブラリを決定してください。なお，インタフェース・ライブラリの詳細については，「第 11 章　インタフェース・ライブラリ」を，ニュークリアス共通オブジェクトについては，「第 8 章　時間管理機能」を参照してください。

5） 情報ファイルの出力指定

プロジェクト・ツリーでシステム・コンフィギュレーション・ファイルを選択し，プロパティ パネルをオープンします。
［システム・コンフィギュレーション・ファイル関連情報］タブにおいて，情報ファイル（システム情報テーブル・ファイル，システム情報ヘッダ・ファイル，システム・コール・テーブル・ファイル）を出力することを設定します。

図 2 － 5　プロパティ パネル：［システム・コンフィギュレーション・ファイル情報］タブ

6）ロード・モジュール・ファイルの出力指定

ビルドの生成物として，ロード・モジュール・ファイルを出力することを設定します。

備考　ロード・モジュールの出力指定についての詳細は，「CubeSuite V850 ビルド編」のユーザーズ・マニュアルを参照してください。

7）ビルド・オプションの設定

コンパイラ，アセンブラ，リンカなどに対するオプションを設定します。

備考 1　ビルド・オプションの設定についての詳細は，「CubeSuite V850 ビルド編」のユーザーズ・マニュアルを参照してください。

備考 2　以下に，コンパイル・オプションとして指定することが禁止されているものを示します。

表 2 － 4　禁止オプション

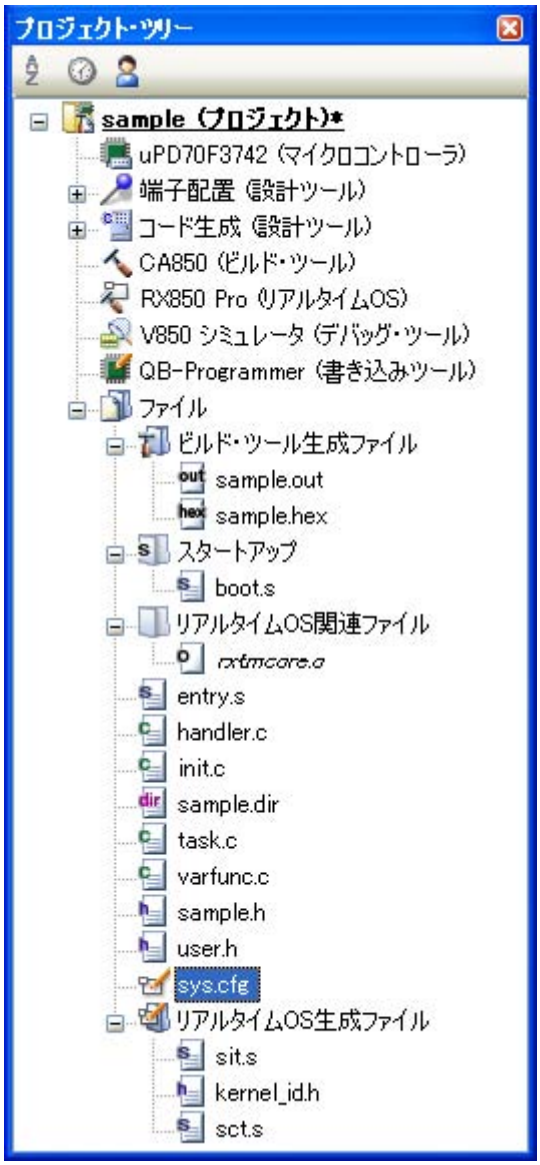
コンパイル・オプション	備考
-reg22 -reg26	RX850 Pro では，32 レジスタ・モードにのみ対応しているいため，左記オプションの指定が禁止されています。
-Xpack=1 -Xpack=2	RX850 Pro では，データ構造体が 4 バイト・アラインされた領域に割り付けられることを前提とした処理が行われるため，左記オプションの指定が禁止されています。

8）ビルドの実行

ビルドを実行し，ロード・モジュール・ファイルを生成します。

備考　ビルドの実行についての詳細は，「CubeSuite V850 ビルド編」のユーザーズ・マニュアルを参照してください。

図 2 － 6　プロジェクト・ツリー パネル（ビルド実行後）



リンカにより作成されたロード・モジュールは，初期化データに関しては RAM に正しく配置された状態になっています。そのため，初期化データがアプリケーションに存在する場合，初期化データ退避領域を確保し，コピー・ルーチンを組み込んだモジュールを作成する必要があります。この場合はリンカが生成したロード・モジュールに対し，ROM 化プロセッサを通したロード・モジュールを作成する必要があります。
ROM 化プロセッサの使用方法，および，コピー・ルーチンについては，「CubeSuite V850 ビルド編」のユーザーズ・マニュアルの「ROM 化プロセッサ」の項を参照してください。

9）プロジェクトの保存

プロジェクトの設定情報をプロジェクト・ファイルに保存します。

備考　プロジェクトの保存についての詳細は，「CubeSuite 起動編」のユーザーズ・マニュアルを参照してください。

2.8　システムへの組み込み

「2.7　ロード・モジュールの作成」の 6) でヘキサ・ファイルの出力を設定すると，ヘキサ・ファイルも生成します。その後，ROM ライタなどを用いてシステムへの組み込みを行います。

第 3 章　ニュークリアス

この章では，RX850 Pro の核であるニュークリアスについて説明します。

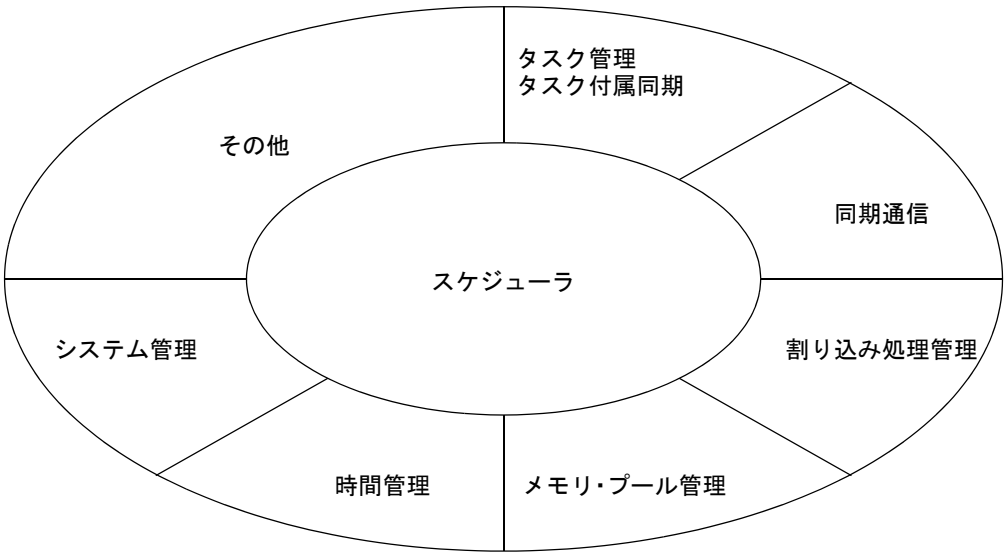
3.1　概　　要

ニュークリアスとは，リアルタイム，マルチタスク制御を行う RX850 Pro の核となる部分であり，次に示す機能を提供しています。

- 管理オブジェクトの生成，初期化処理
- 処理プログラム（タスク，非タスク）から発行されたシステム・コールに対応した処理
- ターゲット・システムの内外で発生した事象に対応した，次に実行すべき処理プログラム（タスク，非タスク）の選択処理

なお，管理オブジェクトの生成，初期化処理とシステム・コールに対応した処理は各種管理モジュールで，処理プログラムの選択処理はスケジューラで行われます。
次に，RX850 Pro のニュークリアスの構成を示します。

図 3 － 1　ニュークリアスの構成



3.2　機　　能

ニュークリアスは，各種管理モジュールとスケジューラから構成されています。
次に，各種管理モジュールとスケジューラの機能概要を示します。
なお，これらの機能に関する詳細は，「第 4 章　タスク管理機能」～「第 9 章　スケジューラ」を参照してください。

- タスク管理機能
  RX850 Pro の処理単位である，「タスクの生成，起動，実行，停止，終了，削除」などといったタスクの状態操作とタスクの状態管理を行います。

- 同期通信機能
  「排他制御，待ち合わせ，通信」といったタスク間の同期通信機能として，次の 3 種類の機能を提供します。

  排他制御機能：　セマフォ
  待ち合わせ機能：　イベントフラグ
  通信機能：　メールボックス

- 割り込み処理管理機能
  「間接起動割り込みハンドラの登録，直接起動割り込みハンドラからの復帰，割り込み許可レベルの変更／獲得」などといったマスカブル割り込みに関連した処理を行います。

- メモリ・プール管理機能
  コンフィギュレーション時に指定されたメモリ領域を，次の 2 種類の領域に分けて管理します。

  - RX850 Pro 用の領域
    各種管理オブジェクト
    メモリ・プール

  - 処理プログラム（タスク，非タスク）用の領域
    処理プログラムのテキスト領域
    処理プログラムのデータ領域
    処理プログラムのスタック領域

  なお，RX850 Pro では，ダイナミックなメモリ・プール管理も行っており，作業用のメモリ領域が必要になった際に獲得し，不要になった際には返却できるといった機能が用意されています。
  ユーザは，このようなダイナミックなメモリ操作を行うことにより，限りあるメモリ領域を効率良く使用することができます。

- 時間管理機能
  ハードウエア（クロック・コントローラなど）により一定周期で発生するクロック割り込みを利用した，タイマ・オペレーション機能（タスクの遅延起床，周期起動ハンドラの起動）などを提供します。

- スケジューラ
  ダイナミックに変化するタスクの状態を観察することにより，タスクの実行順序を管理，決定し，最適なタスクにプロセッサの使用権を与えます。
  なお，RX850 Pro では，タスクの実行順序を決定する方法として，優先度方式と FCFS 方式を採用しています。このため，スケジューラは，駆動された時点で，各タスクに付けられている優先度を参照し，実行可能な状態（run 状態，または ready 状態）にあるタスクの中から最適なものを選び出し，プロセッサの使用権を与えます。

  備考　　RX850 Pro におけるタスクの優先度は，その値が小さいほど高い優先度であることを示します。

第 4 章　タスク管理機能

この章では，RX850 Pro が行うタスク管理機能について説明します。

4.1　概　　要

タスクは実行実体であり，サイズなどが一様ではなく，直接管理することが困難です。そこで，RX850 Pro では，タスクと 1 対 1 に対応した管理オブジェクトを用いることにより，タスクが取り得る状態の管理，タスク自体の管理を行っています。

備考　タスクが処理を実行するうえで必要となるプログラム･カウンタや作業用レジスタなどの実行環境情報は,「タスク・コンテキスト」と呼ばれ，タスクの実行が切り替わる際には，現在実行中のタスク・コンテキストがセーブされ，次に実行されるタスクのタスク・コンテキストがロードされます。

4.2　タスクの状態

タスクは，処理を実行するのに必要な資源の獲得状況や事象発生の有無などにより，さまざまな状態へ遷移します。
そこで，RX850 Pro では，タスクの遷移する状態を，次の 7 種類に分けて管理しています。

- 未登録（non_existent）状態
  タスクとして生成されていない状態，または削除された際に遷移する状態です。
  なお，non_existent 状態のタスクは，その実行実体がメモリ上に配置されていても，RX850 Pro が管理していない状態です。

- 休止（dormant）状態
  タスクとして生成されたときの状態，またはタスクとしての処理を終了したときに遷移する状態です。
  なお，dormant 状態のタスクは，RX850 Pro のスケジューリング対象から除外されています。
  また，wait 状態とは次のような相違点があります。

  - すべての資源を解放している
  - タスク・コンテキストが処理再開時に初期化される
  - 状態操作を伴うシステム・コール（ter_tsk，chg_pri など）がエラーとなる

- 実行可能（ready）状態
  タスクとして処理を実行するうえで必要な準備は整っているが，より高い優先度（同じ優先度の場合もある）を持つほかのタスクが実行されているため，プロセッサの実行権が割り当てられるのを待っている状態です。
  なお，ready 状態のタスクは，RX850 Pro のスケジューリング対象となります。

- 実行（run）状態
  プロセッサの実行権が割り当てられ，現在タスクとして処理を実行中の状態です。
  なお，run 状態のタスクは，システム全体で 1 タスクしか存在しません。

- 待ち（wait）状態
  処理を実行するために必要な条件が整わないため，処理の実行が中断した状態です。
  なお，wait 状態からの処理再開は，処理の実行が中断した箇所からとなります。したがって，処理を再開するうえで必要となるタスク・コンテキストは，中断直前の値が復元されます。
  また，RX850 Pro では，wait 状態へと遷移する原因となった条件の種類により，次の 6 つの状態に分けて管理しています。

起床待ち状態：	slp_tsk，tslp_tsk を発行した際，自タスクのカウンタ（起床要求の発行回数を保持）が 0x0 の場合に遷移する状態です。
資源待ち状態：	wai_sem，twai_sem を発行した際，対象セマフォから資源を獲得できなかった場合に遷移する状態です。
イベントフラグ待ち状態：	wai_flg，twai_flg を発行した際，対象イベントフラグが要求した条件を満たしていなかった場合に遷移する状態です。
メッセージ待ち状態：	rcv_msg，trcv_msg を発行した際，対象メールボックスからメッセージを受信できなかった場合に遷移する状態です。

メモリ・ブロック待ち状態：　get_blk，tget_blk を発行した際，対象メモリ・プールからメモリ・ブロックを獲得できなかった場合に遷移する状態で。
時間経過待ち状態：　dly_tsk を発行した際，遷移する状態です。

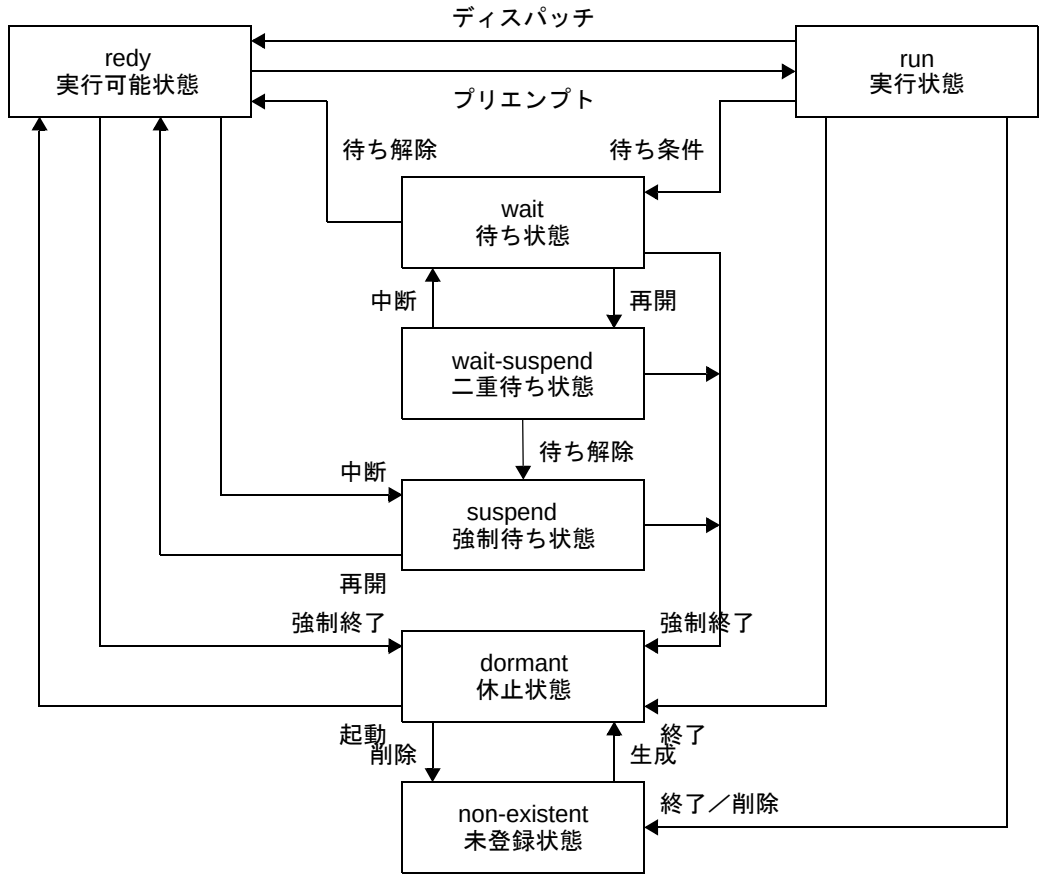
- 強制待ち（suspend）状態
  他タスクから強制的に実行を中断させられた状態です。
  なお，suspend 状態からの処理再開は，処理の実行が中断した箇所からとなります。したがって，処理を再開するうえで必要となるタスク・コンテキストは，中断直前の値が復元されます。

  備考　　RX850 Pro では，suspend 状態をネストさせることも可能です（127 回まで）。

- 二重待ち（wait_suspend）状態
  wait 状態と suspend 状態が複合した状態です。
  wait 状態が解除されると suspend 状態へ，suspend 状態が解除されると wait 状態へと遷移します。

図 4 － 1 に，タスクの状態遷移を示します。

図 4 － 1　タスクの状態遷移



4.3　タスクの生成

RX850 Pro では，タスクの生成において，「システム初期化処理（ニュークリアス初期化部）において，スタティックに生成する」，「処理プログラム内からシステム・コールを発行し，ダイナミックに生成する」の 2 種類のインタフェースを用意しています。
なお，RX850 Pro におけるタスクの生成とは，タスクを管理するための領域（管理オブジェクト）をシステム・メモリ領域に確保したあと，初期化し，タスクの状態を non_existent 状態から dormant 状態へと遷移させることです。

- スタティックに登録する場合
  タスクをスタティックに登録する場合，コンフィギュレーション時にタスク情報として指定します。
  RX850 Pro では，システム初期化処理時，情報ファイル（システム情報テーブル，システム情報ヘッダ・ファイル）に定義された情報を基にタスクの生成処理を行い，管理対象とします。

- ダイナミックに登録する場合
  タスクをダイナミックに登録する場合，処理プログラム（タスク）内から cre_tsk を発行します。
  RX850 Pro では，cre_tsk 発行時，パラメータで指定された情報を基にタスクの生成処理を行い，管理対象とします。

4.4　タスクの起動

RX850 Pro におけるタスクの起動は，タスクを dormant 状態から ready 状態へ遷移させ，スケジューリング対象とすることです。
なお，タスクの起動は，パラメータでタスクを指定して sta_tsk を発行することにより行います。

- sta_tsk
  パラメータで指定されたタスクを dormant 状態から ready 状態へと遷移させます。

4.5　タスクの終了

RX850 Pro におけるタスクの終了は，各種状態（ready 状態，run 状態，wait 状態，suspend 状態，wait_suspend 状態）のタスクを dormant 状態に遷移させ，RX850 Pro のスケジューリング対象から除外することです。
RX850 Pro では，タスクの終了形態として次に示す 2 種類を用意しています。

正常終了：　すべての処理が順調に完了し，スケジューリング対象である必要がなくなったとき，タスク自身で終了する形態です。
強制終了：　処理途中で何らかの異常が発生し，緊急に処理を終了しなければならないとき，他タスクから終了させられる形態です。

タスクの終了は，次に示すシステム・コールの発行により行います。

- ext_tsk
  自タスクを run 状態から dormant 状態へと遷移させます。

- exd_tsk
  自タスクを run 状態から non_existent 状態へと遷移させます。

- ter_tsk
  パラメータで指定されたタスクを強制的に dormant 状態へと遷移させます。

4.6　タスクの削除

RX850 Pro におけるタスクの削除は, run 状態, または dormant 状態のタスクを non_existent 状態へと遷移させ, RX850 Pro の管理下から除外することです。
タスクの削除は，次に示すシステム・コールの発行により行います。

- exd_tsk
  自タスクを run 状態から non_existent 状態へと遷移させます。

- del_tsk
  パラメータで指定されたタスクを dormant 状態から non_existent 状態へと遷移させます。

4.7　タスク内での処理

RX850 Pro 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。
したがって，タスクの処理を記述する際には，次の点に注意してください。

- レジスタの退避と復帰
  RX850 Pro 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，復帰処理を行っています。したがって，タスクの開始部分で作業用レジスタの退避処理を，終了部分で作業用レジスタの復帰処理を記述する必要はありません。
  ただし，タスクをアセンブリ言語で記述し，レジスタ変数用レジスタを使用する場合は，タスクの開始部分でレジスタ変数用レジスタの退避処理を，終了部分でレジスタ変数用レジスタの復帰処理を記述する必要があります。

- スタックの切り替え
  RX850 Pro では，タスクを切り替える際，各タスク専用のタスク用スタックへの切り替え処理を行っています。したがって，タスクの開始部分と終了部分でスタックの切り替え処理を記述する必要はありません。

- システム・コールの発行制限
  RX850 Pro のシステム・コールのなかには，タスク内で発行できないものもあります。
  次に，タスク内で発行可能なシステム・コールの一覧を示します。

  - タスク管理機能
    cre_tsk, del_tsk, sta_tsk, ext_tsk, exd_tsk, ter_tsk, dis_dsp, ena_dsp, chg_pri, rot_rdq, rel_wai, get_tid, ref_tsk, vget_tid

  - タスク付属同期機能
    sus_tsk, rsm_tsk, frsm_tsk, slp_tsk, tslp_tsk, wup_tsk, can_wup

  - 同期通信機能
    cre_sem, del_sem, sig_sem, wai_sem, preq_sem, twai_sem, ref_sem, vget_sid, cre_flg, del_flg, set_flg, clr_flg, wai_flg, pol_flg, twai_flg, ref_flg, vget_fid, cre_mbx, del_mbx, snd_msg, rcv_msg, prcv_msg, trcv_msg, ref_mbx, vget_mid

  - 割り込み処理管理機能
    def_int, ena_int, dis_int, loc_cpu, unl_cpu, chg_icr, ref_icr

  - メモリ・プール管理機能
    cre_mpl, del_mpl, get_blk, pget_blk, tget_blk, rel_blk, ref_mpl, vget_pid

  - 時間管理機能
    set_tim, get_tim, dly_tsk, def_cyc, act_cyc, ref_cyc

  - システム管理機能
    get_ver, ref_sys, def_svc, viss_svc

4.7.1　タスク情報の獲得

タスク情報の獲得は，ref_tsk を発行することにより行われます。

- ref_tsk
  パラメータで指定されたタスクのタスク情報（拡張情報，現在の優先度など）を獲得します。
  次に，タスク情報の詳細を示します。

  - 拡張情報
  - 現在の優先度
  - タスクの状態
  - wait 状態の種類
  - 待ち対象オブジェクト（セマフォ，イベントフラグなど）の ID 番号
  - 起床要求数
  - サスペンド要求数
  - キー ID 番号

4.7.2　ID 番号の獲得

タスクの ID 番号の獲得は，vget_tid を発行することにより行われます。

- vget_tid
  パラメータで指定されたタスクの ID 番号を獲得します。
  システム・コールでタスクに対する操作をするとき，タスクの ID 番号が必要です。タスクを生成する際，ID 番号をユーザが一意に決定するか，自動的に振り当てるかを指定できます。しかし，自動的に振り当てると指定した場合，ユーザはタスクの ID 番号を知ることができません。そのために必要となるものが「キー ID 番号」です。キー ID 番号はタスク生成の際に一意に指定します。
  このキー ID 番号をパラメータとして本システム・コールを発行すると，そのキー ID 番号を持つタスクの ID 番号を取得できます。

第 5 章　同期通信機能

この章では，RX850 Pro が行う同期通信機能について説明します。

5.1　概　　要

複数のタスクが並行に実行されている環境（マルチタスク処理）では，あるタスクの処理結果により，次に実行されるタスクが選択されたり，タスクの処理内容に違いが出るなど，タスク間で処理の実行条件を制限しあったり，処理内容が相互に関係しているという場合があります。

このため，あるタスクが他タスクの実行結果が出るまで実行を中断したり，処理を継続するうえで必要な条件が整うまで待つといった，タスク間の連絡機能が必要となります。

RX850 Pro では，このような機能を，「同期機能」と呼びます。なお，同期機能には，「排他制御機能」と「待ち合わせ機能」があり，RX850 Pro では，排他制御機能としてセマフォを，待ち合わせ機能としてイベントフラグを提供しています。

また，マルチタスク処理では，他タスクから処理結果を通知してもらうといった，タスク間の通信機能も必要となります。

RX850 Pro では，このような機能を，「通信機能」と呼びます。なお，RX850 Pro では，通信機能としてメールボックスを提供しています。

5.2　セマフォ

マルチタスク処理では，並行に動作する複数のタスクが，限られた数の資源（A/D コンバータ，コプロセッサ，ファイル，プログラムなど）を同時に使用するといった，資源の競合を防ぐ機能が必要となります。RX850 Pro では，このような機能を実現するために，非負数の計数型セマフォを提供しています。

なお，セマフォに対するダイナミックな操作は，次に示すセマフォ関連のシステム・コールを用いて行います。

cre_sem：　セマフォを生成する
del_sem：　セマフォを削除する
sig_sem：　資源を返却する
wai_sem：　資源を獲得する
preq_sem：　資源を獲得する（ポーリング）
twai_sem：　資源を獲得する（タイムアウトあり）
ref_sem：　セマフォ情報を獲得する
vget_sid：　セマフォの ID 番号を獲得する

備考　タスクの実行に必要となる各種要素を「資源」と呼びます。つまり，資源とは，A/D コンバータ，コプロセッサなどといったハードウエアや，ファイル，プログラムなどといったソフトウエアのすべてを指します。

5.2.1　セマフォの生成

RX850 Pro では，セマフォの生成において，「システム初期化処理（ニュークリアス初期化部）において，スタティックに生成する」，「処理プログラム内からシステム・コールを発行し，ダイナミックに生成する」の 2 種類のインタフェースを用意しています。

RX850 Pro におけるセマフォの生成とは，セマフォを管理するための領域（管理オブジェクト）をシステム・メモリ領域から確保したあと，初期化することです。

- スタティックに登録する場合
  セマフォをスタティックに登録する場合，コンフィギュレーション時にセマフォ情報として指定します。
  RX850 Pro では，システム初期化処理時，情報ファイル（システム情報テーブル，システム情報ヘッダ・ファイル）に定義された情報を基にセマフォの生成処理を行い，管理対象とします。

- ダイナミックに登録する場合
  セマフォをダイナミックに登録する場合，処理プログラム（タスク）内から cre_sem を発行します。
  RX850 Pro では，cre_sem 発行時，パラメータで指定された情報を基にセマフォの生成処理を行い，管理対象とします。

5.2.2　セマフォの削除

セマフォの削除は，del_sem を発行することにより行います。

- del_sem
  パラメータで指定されたセマフォを削除します。
  これにより，対象セマフォは RX850 Pro の管理対象から除外されます。
  ただし，本システム・コールを発行した際，対象セマフォの待ちキューにタスクがキューイングされていた場合には，該当タスクを待ちキューから外すとともに，wait 状態（資源待ち状態）から ready 状態へと遷移させます。
  なお，wait 状態を解除されたタスクには，wait 状態へと遷移するきっかけとなったシステム・コール（wai_sem，twai_sem）の戻り値として E_DLT が返されます。

5.2.3　資源の返却

資源の返却は，sig_sem を発行することにより行います。

- sig_sem
  パラメータで指定されたセマフォに資源を返却（セマフォ・カウンタに 0x1 を加算）します。
  ただし，本システム・コールを発行した際，対象セマフォの待ちキューにタスクがキューイングされていた場合には，資源の返却処理（セマフォ・カウンタの加算）は行わず，該当タスク（待ちキューの先頭タスク）に資源を渡します。
  これにより，該当タスクは待ちキューから外れ，wait 状態（資源待ち状態）から ready 状態へ，または wait_suspend 状態から suspend 状態へと遷移します。

5.2.4　資源の獲得

資源の獲得は，wai_sem，preq_sem，または twai_sem を発行することにより行います。

- wai_sem
  パラメータで指定されたセマフォから資源を獲得（セマフォ・カウンタから 0x1 を減算）します。
  ただし，本システム・コールを発行した際，対象セマフォから資源を獲得できなかった（空き資源が存在しなかった）場合には，自タスクを対象セマフォの待ちキューにキューイングしたあと，run 状態から wait 状態（資源待ち状態）へと遷移させます。
  なお，資源待ち状態は次の場合に解除され，ready 状態へ遷移します。

  - sig_sem が発行された
  - del_sem が発行され，対象セマフォが削除された
  - rel_wai が発行され，強制的に待ち状態が解除された

  備考　自タスクを対象セマフォの待ちキューにキューイングする際は，対象セマフォ生成時（コンフィギュレーション時，または cre_sem 発行時）に指定された順（FIFO 順，または優先度順）に行われます。

- preq_sem
  パラメータで指定されたセマフォから資源を獲得（セマフォ・カウンタから 0x1 を減算）します。
  ただし，本システム・コールを発行した際，対象セマフォから資源を獲得できなかった（空き資源が存在しなかった）場合には，戻り値として E_TMOUT が返されます。

- twai_sem
  パラメータで指定されたセマフォから資源を獲得（セマフォ・カウンタから 0x1 を減算）します。
  ただし，本システム・コールを発行した際，対象セマフォから資源を獲得できなかった（空き資源が存在しなかった）場合には，自タスクを対象セマフォの待ちキューにキューイングしたあと，run 状態から wait 状態（資源待ち状態）へと遷移させます。
  なお，資源待ち状態は次の場合に解除され，ready 状態へ遷移します。

  - パラメータで指定された待ち時間が経過した
  - sig_sem が発行された
  - del_sem が発行され，対象セマフォが削除された
  - rel_wai が発行され，強制的に待ち状態が解除された

備考　自タスクを対象セマフォの待ちキューにキューイングする際は，対象セマフォ生成時（コンフィギュレーション時，または cre_sem 発行時）に指定された順（FIFO 順，または優先度順）に行われます。

5.2.5　セマフォ情報の獲得

セマフォ情報の獲得は，ref_sem を発行することにより行われます。

- ref_sem
  パラメータで指定されたセマフォのセマフォ情報（拡張情報，待ちタスクの有無など）を獲得します。
  次に，セマフォ情報の詳細を示します。

  - 拡張情報
  - 待ちタスクの有無
  - 現在の資源数
  - 生成時に指定した最大資源数
  - キー ID 番号

5.2.6　ID 番号の獲得

セマフォの ID 番号の獲得は，vget_sid を発行することにより行われます。

- vget_sid
  パラメータで指定されたセマフォの ID 番号を獲得します。
  システム・コールでセマフォに対する操作をするとき，セマフォの ID 番号が必要です。セマフォを生成する際，ID 番号をユーザが一意に決定するか，自動的に振り当てるかを指定できます。しかし，自動的に振り当てると指定した場合，ユーザはセマフォの ID 番号を知ることができません。そのために必要となるものが「キー ID 番号」です。
  キー ID 番号はセマフォ生成の際に一意に指定します。
  このキー ID 番号をパラメータとして本システム・コールを発行すると，そのキー ID 番号を持つセマフォの ID 番号を取得できます。

5.2.7　セマフォによる排他制御

次に，セマフォを使用してタスク間の排他制御を行った場合の動作例を示します。

【 条件 】

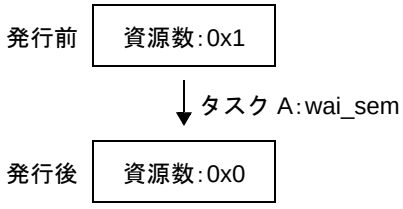
- タスクの優先度
  タスク A ＞ タスク B

- タスクの状態
  タスク A：　run 状態
  タスク B：　ready 状態

- セマフォの属性
  セマフォの初期資源数：　0x1
  セマフォの最大資源数：　0x5
  タスクのキューイング順序：　FIFO 順

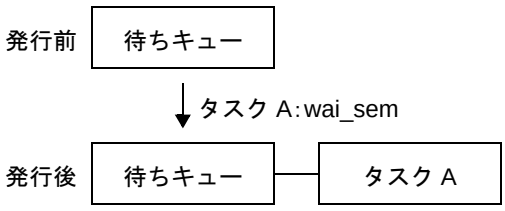
（1）タスク A が wai_sem を発行します。
現在 RX850 Pro が管理している対象セマフォの資源数は「0x1」です。したがって，RX850 Pro は対象セマフォのカウンタから 0x1 を減算します。
このとき，タスク A は wait 状態（資源待ち状態）へと遷移することはなく，run 状態のままとなります。
なお，対象セマフォのカウンタは図 5 － 1 のようになります。

図 5 － 1　セマフォ・カウンタの状態



（2）タスク A が wai_sem を発行します。
現在 RX850 Pro が管理している対象セマフォの資源数は「0x0」です。したがって，RX850 Pro はタスク A を run 状態から wait 状態（資源待ち状態）へと遷移させ，対象セマフォの待ちキューの最後尾にキューイングします。
このとき，対象セマフォの待ちキューは図 5 － 2 のようになります。

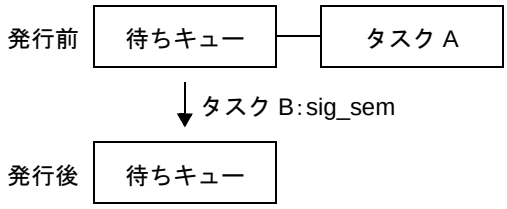
図 5 － 2　待ちキューの状態（wai_sem 発行時）



（3）タスク A の資源待ち状態への遷移にともない，タスク B が ready 状態から run 状態へと遷移します。

（4）タスク B が sig_sem を発行します。
これにより，対象セマフォの待ちキューにキューイングされているタスク A が資源待ち状態から ready 状態へと遷移します。
このとき，対象セマフォの待ちキューは，図 5 － 3 のようになります。

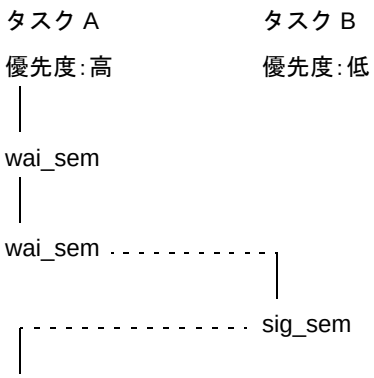
図 5 － 3　待ちキューの状態（sig_sem 発行時）



（5）優先度の高いタスク A が ready 状態から run 状態へと遷移します。
　　また，タスク B は run 状態から ready 状態へと遷移します。

図 5 － 4 に，（1）から（5）で説明した排他制御の流れを示します。

図 5 － 4　セマフォによるタスクの排他制御



5.3　イベントフラグ

マルチタスク処理では，あるタスクの処理結果が出るまで，他タスクが処理の実行再開を待つといった，タスク間の待ち合わせ機能が必要となります。このような場合，「処理結果が出た」という事象（イベント）が発生したか否かを，他タスクで判断できるような機能があればよく，RX850 Pro ではこのような機能を実現するために，イベントフラグを提供しています。

イベントフラグは，事象の有無を表す 1 ビットのフラグから構成されるデータの集合体です。RX850 Pro のイベントフラグは，32 ビットを一塊の情報として扱っており，各ビットに意味を持たせたり，数ビットの組み合わせで意味を持たせたりできます。

なお，イベントフラグに対するダイナミックな操作には，次に示すイベントフラグ関連のシステム・コールを使用します。

cre_flg：　イベントフラグを生成する
del_flg：　イベントフラグを削除する
set_flg：　ビット・パターンをセットする
clr_flg：　ビット・パターンをクリアする
wai_flg：　ビット・パターンをチェックする
pol_flg：　ビット・パターンをチェックする（ポーリング）
twai_flg：　ビット・パターンをチェックする（タイムアウトあり）
ref_flg：　イベントフラグ情報を獲得する
vget_fid：　イベントフラグの ID 番号を獲得する

5.3.1　イベントフラグの生成

RX850 Pro では，イベントフラグの生成において，「システム初期化処理（ニュークリアス初期化部）において，スタティックに生成する」，「処理プログラム内からシステム・コールを発行し，ダイナミックに生成する」の 2 種類のインタフェースを用意しています。

なお，RX850 Pro におけるイベントフラグの生成とは，イベントフラグを管理するための領域（管理オブジェクト）をシステム・メモリ領域に確保したあと，初期化することです。

- スタティックに登録する場合
  イベントフラグをスタティックに登録する場合，コンフィギュレーション時にイベントフラグ情報として指定します。
  RX850 Pro では，システム初期化処理時，情報ファイル（システム情報テーブル，システム情報ヘッダ・ファイル）に定義された情報を基にイベントフラグの生成処理を行い，管理対象とします。

- ダイナミックに登録する場合
  イベントフラグをダイナミックに登録する場合，処理プログラム（タスク）内から cre_flg を発行します。
  RX850 Pro では，cre_flg 発行時，パラメータで指定された情報を基にイベントフラグの生成処理を行い，管理対象とします。

5.3.2　イベントフラグの削除

イベントフラグの削除は，del_flg を発行することにより行います。

- del_flg
  パラメータで指定されたイベントフラグを削除します。
  これにより，対象イベントフラグは RX850 Pro の管理対象から除外されます。
  ただし，本システム・コールを発行した際，対象イベントフラグの待ちキューにタスクがキューイングされていた場合には，該当タスクを待ちキューから外すとともに，wait 状態（イベントフラグ待ち状態）から ready 状態へと遷移させます。
  なお，wait 状態を解除されたタスクには，wait 状態へと遷移するきっかけとなったシステム・コール（wai_flg, twai_flg）の戻り値として E_DLT が返されます。

5.3.3　ビット・パターンのセット

イベントフラグのビット・パターンのセットは，set_flg の発行により行います。

- set_flg
  パラメータで指定されたイベントフラグにビット・パターンをセットします。
  ただし，本システム・コールを発行した際，対象イベントフラグの待ちキューにキューイングされているタスクの待ち条件を満足した場合には，該当タスクを待ちキューから外します。これにより，該当タスクは wait 状態（イベントフラグ待ち状態）から ready 状態へ，または wait_suspend 状態から suspend 状態へと遷移します。

5.3.4　ビット・パターンのクリア

イベントフラグのビット・パターンのクリアは，clr_flg の発行により行います。

- clr_flg
  パラメータで指定されたイベントフラグのビット・パターンをクリアします。
  ただし，本システム・コールを発行した際，すでに対象イベントフラグのビット・パターンが 0 にクリアされていても，エラーとして扱われないので注意が必要です。

5.3.5　ビット・パターンのチェック

イベントフラグのビット・パターンのチェックは，wai_flg，pol_flg，または twai_flg の発行により行われます。

- wai_flg
  パラメータで指定されたイベントフラグに要求する待ち条件を満足するビット･パターンがセットされているかどうかをチェックします。
  ただし，本システム・コールを発行した際，対象イベントフラグのビット・パターンが要求する待ち条件を満足していなかった場合には，自タスクを対象イベントフラグの待ちキューの最後尾にキューイングしたあと，run 状態から wait 状態（イベントフラグ待ち状態）へと遷移させます。
  なお，イベントフラグ待ち状態は次の場合に解除され，ready 状態へ遷移します。

  - set_flg が発行され，要求する待ち条件がセットされた
  - del_flg が発行され，対象イベントフラグが削除された
  - rel_wai が発行され，強制的に待ち状態が解除された

- pol_flg
  パラメータで指定されたイベントフラグに要求する待ち条件を満足するビット･パターンがセットされているかどうかをチェックします。
  ただし，本システム・コールを発行した際，対象イベントフラグのビット・パターンが要求する待ち条件を満足していなかった場合は，戻り値として E_TMOUT が返されます。

- twai_flg
  パラメータで指定されたイベントフラグに要求する待ち条件を満足するビット･パターンがセットされているかどうかをチェックします。
  ただし，本システム・コールを発行した際，対象イベントフラグのビット・パターンが要求する待ち条件を満足していなかった場合には，自タスクを対象イベントフラグの待ちキューの最後尾にキューイングしたあと，run 状態から wait 状態（イベントフラグ待ち状態）へと遷移させます。
  なお，イベントフラグ待ち状態は次の場合に解除され，ready 状態へ遷移します。

  - パラメータで指定された待ち時間が経過した
  - set_flg が発行され，要求する待ち条件がセットされた
  - del_flg が発行され，対象イベントフラグが削除された
  - rel_wai が発行され，強制的に待ち状態が解除された

  また，RX850 Pro では，イベントフラグの待ち条件や条件成立時の処理を次のように指定できます。

- 待ち条件

  - AND 待ち
    要求ビット・パターンで「1」が設定されている全ビットが，対象イベントフラグに設定されるまで待ちます。

  - OR 待ち
    要求ビット・パターンで「1」が設定されているいずれかのビットが，対象イベントフラグに設定されるまで待ちます。

- 条件成立時

  - ビット・パターンのクリア
    待ち条件を満足した際，対象イベントフラグのビット・パターンをクリアします。

5.3.6　イベントフラグ情報の獲得

イベントフラグ情報の獲得は，ref_flg を発行することにより行われます。

- ref_flg
  パラメータで指定されたイベントフラグのイベントフラグ情報（拡張情報，待ちタスクの有無など）を獲得します。
  次に，イベントフラグ情報の詳細を示します。

  - 拡張情報
  - 待ちタスクの有無
  - 現在のビット・パターン
  - キー ID 番号

5.3.7　ID 番号の獲得

イベントフラグの ID 番号の獲得は，vget_fid を発行することにより行われます。

- vget_fid
  パラメータで指定されたイベントフラグの ID 番号を獲得します。
  システム・コールでイベントフラグに対する操作をするとき，イベントフラグの ID 番号が必要です。
  イベントフラグを生成する際，ID 番号をユーザが一意に決定するか，自動的に振り当てるかを指定できます。
  しかし，自動的に振り当てると指定した場合，ユーザはイベントフラグの ID 番号を知ることができません。
  そのために必要となるものが「キー ID 番号」です。キー ID 番号はイベントフラグ生成の際に一意に指定します。
  このキー ID 番号をパラメータとして本システム・コールを発行すると，そのキー ID 番号を持つイベントフラグの ID 番号を取得できます。

5.3.8　イベントフラグによる待ち合わせ

次に，イベントフラグを使用してタスク間の待ち合わせを行った場合の動作例を示します。

【 条件 】

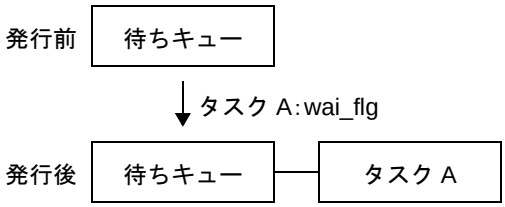
- タスクの優先度
  タスク A ＞ タスク B

- タスクの状態
  タスク A :　run 状態
  タスク B :　ready 状態

- イベントフラグの属性
  初期ビット・パターン :　0x0
  待ちキューにキューイング可能なタスク数 :　1 タスクのみ

（1）タスク A が wai_flg を発行します。要求ビット・パターンは「0x1」，待ち条件は「TWF_ANDW | TWF_CLR」とします。
現在，RX850 Pro が管理している対象イベントフラグのビット・パターンは「0x0」です。したがって，RX850 Pro はタスク A を run 状態から wait 状態（イベントフラグ待ち状態）へと遷移させ，対象イベントフラグの待ちキューの最後尾にキューイングします。
このとき，対象イベントフラグの待ちキューは，図 5 － 5 のようになります。

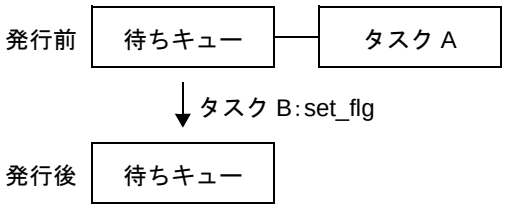
図 5 － 5　待ちキューの状態（wai_flg 発行時）



（2）タスク A のイベントフラグ待ち状態への遷移にともない，タスク B が ready 状態から run 状態へと遷移します。

（3）タスク B が set_flg を発行します。設定ビット・パターンは「0x1」とします。
これにより，対象イベントフラグの待ちキューにキューイングされているタスク A の待ち条件を満足したため，タスク A がイベントフラグ待ち状態から ready 状態へと遷移します。また，タスク A は，wai_flg を発行した際「TWF_CLR」を指定していたため，対象イベントフラグのビット・パターンはクリアされます。
このとき，対象イベントフラグの待ちキューは，図 5 － 6 のようになります。

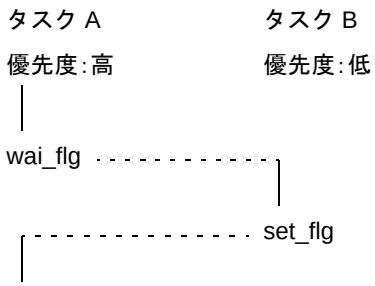
図 5 － 6　待ちキューの状態（set_flg 発行時）



（4）優先度の高いタスク A が ready 状態から run 状態へと遷移します。
なお，タスク B は run 状態から ready 状態へと遷移します。

図 5 － 7 に，（1）から（4）で説明した待ち合わせの流れを示します。

図 5 － 7　イベントフラグによるタスクの待ち合わせ



5.4　メールボックス

マルチタスク処理では，他タスクから処理結果を通知してもらうといった，タスク間の通信機能が必要となります。RX850 Pro では，このような機能を実現するためにメールボックスを提供しています。

RX850 Pro におけるメールボックスは，タスク専用の待ちキューとメッセージ専用の待ちキューを持ち，タスク間のメッセージ通信機能として使用するほかに，タスク間の待ち合わせ機能としても使用できます。

なお，メールボックスに対するダイナミックな操作は，次に示すメールボックス関連のシステム・コールを用いて行います。

cre_mbx：　メールボックスを生成する
del_mbx：　メールボックスを削除する
snd_msg：　メッセージを送信する
rcv_msg：　メッセージを受信する
prcv_msg：　メッセージを受信する（ポーリング）
trcv_msg：　メッセージを受信する（タイムアウトあり）
ref_mbx：　メールボックス情報を獲得する
vget_mid：　メールボックスの ID 番号を獲得する

5.4.1　メールボックスの生成

RX850 Pro では，メールボックスの生成において，「システム初期化処理（ニュークリアス初期化部）において，スタティックに生成する」，「処理プログラム内からシステム・コールを発行し，ダイナミックに生成する」の 2 種類のインタフェースを用意しています。

RX850 Pro におけるメールボックスの生成とは，メールボックスを管理するための領域（管理オブジェクト）をシステム・メモリ領域から確保したあと，初期化することです。

- スタティックに登録する場合
  メールボックスをスタティックに登録する場合，コンフィギュレーション時にメールボックス情報として指定します。
  RX850 Pro では，システム初期化処理時，情報ファイル（システム情報テーブル，システム情報ヘッダ・ファイル）に定義された情報を基にメールボックスの生成処理を行い，管理対象とします。

- ダイナミックに登録する場合
  メールボックスをダイナミックに登録する場合，処理プログラム（タスク）内から cre_mbx を発行します。
  RX850 Pro では，cre_mbx 発行時，パラメータで指定された情報を基にメールボックスの生成処理を行い，管理対象とします。

5.4.2　メールボックスの削除

メールボックスの削除は，del_mbx を発行することにより行われます。

- del_mbx
  パラメータで指定されたメールボックスを削除します。
  これにより，対象メールボックスは RX850 Pro の管理対象から除外されます。
  ただし，本システム・コールを発行した際，対象メールボックスのタスク用待ちキューにタスクがキューイングされていた場合には，該当タスクをタスク用待ちキューから外すとともに，wait 状態（メッセージ待ち状態）から ready 状態へと遷移させます。
  なお，wait 状態を解除されたタスクには，wait 状態へと遷移するきっかけとなったシステム・コール（rcv_msg, trcv_msg）の戻り値として E_DLT が返されます。
  また，本システム・コールを発行した際，対象メールボックスのメッセージ用待ちキューにメッセージがキューイングされていた場合は，該当メッセージをメッセージ用待ちキューから外すとともに，メッセージ用領域を獲得したメモリ・プールに返却します。このため，メモリ・プールから獲得したメモリ・ブロック以外の領域をメッセージ用領域として使用していた場合に，メールボックスを削除するような動作については，動作保証の対象となりません。本システム・コールによって削除される可能性のあるメールボックスについては，必ずメモリ・プールから獲得したメモリ・ブロックをメッセージ用領域として使用してください。

5.4.3　メッセージの送信

メッセージの送信は，snd_msg を発行することにより行われます。

- snd_msg
  パラメータで指定されたメールボックスにメッセージを送信します。
  ただし，本システム・コールを発行した際，対象メールボックスのタスク用待ちキューにタスクがキューイングされていた場合には，メッセージのキューイング操作は行わず，該当タスク（タスク用待ちキューの先頭タスク）にメッセージを渡します。
  これにより，該当タスクは待ちキューから外れ，wait 状態（メッセージ待ち状態）から ready 状態へ，または wait_suspend 状態から suspend 状態へと遷移します。
  対象メールボックスのタスク用待ちキューにタスクがキューイングされていない場合には，メッセージは対象メールボックスのメッセージ用待ちキューにキューイングされます。

  備考　メッセージを対象メールボックスのメッセージ用待ちキューにキューイングする際は，対象メールボックス生成時（コンフィギュレーション時，または cre_mbx 発行時）に指定された順（FIFO 順，または優先度順）に行われます。

5.4.4　メッセージの受信

メッセージの受信は，rcv_msg，prcv_msg，または trcv_msg を発行することにより行われます。

- rcv_msg
  パラメータで指定されたメールボックスからメッセージを受信します。
  ただし，本システム・コールを発行した際，対象メールボックスからメッセージを受信できなかった（メッセージ用待ちキューにメッセージが存在しなかった）場合には，自タスクを対象メールボックスのタスク用待ちキューの最後尾にキューイングしたあと，run 状態から wait 状態（メッセージ待ち状態）へと遷移させます。
  なお，メッセージ待ち状態は次の場合に解除され，ready 状態へ遷移します。

  - snd_msg が発行された
  - del_mbx が発行され，対象メールボックスが削除された
  - rel_wai が発行され，強制的に待ち状態が解除された

  備考　自タスクを対象メールボックスのタスク用待ちキューにキューイングする際は，対象メールボックス生成時（コンフィギュレーション時，または cre_mbx 発行時）に指定された順（FIFO 順，または優先度順）に行われます。

- prcv_msg
  パラメータで指定されたメールボックスからメッセージを受信します。
  ただし，本システム・コールを発行した際，対象メールボックスからメッセージを受信できなかった（メッセージ用待ちキューにメッセージが存在しなかった）場合には，戻り値として E_TMOUT が返されます。

- trcv_msg
  パラメータで指定されたメールボックスからメッセージを受信します。
  ただし，本システム・コールを発行した際，対象メールボックスからメッセージを受信できなかった（メッセージ用待ちキューにメッセージが存在しなかった）場合には，自タスクを対象メールボックスのタスク用待ちキューの最後尾にキューイングしたあと，run 状態から wait 状態（メッセージ待ち状態）へと遷移します。
  なお，メッセージ待ち状態は次の場合に解除され，ready 状態へ遷移します。

  - パラメータで指定された待ち時間が経過した
  - snd_msg が発行された
  - del_mbx が発行され，対象メールボックスが削除された
  - rel_wai が発行され，強制的に待ち状態が解除された

  備考　自タスクを対象メールボックスのタスク用待ちキューにキューイングする際は，対象メールボックス生成時（コンフィギュレーション時，または cre_mbx 発行時）に指定された順（FIFO 順，または優先度順）に行われます。

5.4.5　メッセージ

RX850 Pro では，メールボックスを介してタスク間でやり取りされる情報を「メッセージ」と呼びます。
なお，メッセージは，メールボックスを介して任意のタスクに送ることができます。ただし，RX850 Pro におけるタスク間通信では，メッセージの先頭アドレスを受信側タスクに渡すだけであり，メッセージの内容がほかの領域にコピーされるわけではないので注意が必要です。

- メッセージ領域の確保
  メッセージ用の領域は，get_blk，pget_blk，または tget_blk を発行し，RX850 Pro が管理しているメモリ・プールから確保することを推奨しています。
  また，メッセージの先頭 4 バイトは，メッセージ用待ちキューにキューイングする際のリンク領域として使用されます。
  したがって，メッセージはメッセージ用領域の先頭から 4 バイト以降に格納してください。

- メッセージ内容の作成
  RX850 Pro では，メールボックスに対して送信するメッセージの長さ，内容についての既定はありません。先頭 4 バイト以降のメッセージの長さ，内容については，メールボックスを使用して通信を行うタスク間で既定します。

  備考　RX850 Pro では，メッセージの先頭 4 バイトをメッセージ用待ちキューにキューイングする際のリンク領域として使用します。したがって，メッセージを対象メールボックスに送信する場合は，snd_msg を発行する前にメッセージの先頭 4 バイトに「0x0」を設定してください。
  なお，snd_msg を発行した際，メッセージの先頭 4 バイトに「0x0」以外の値が設定されていた場合には，RX850 Pro は「対象メッセージはすでにメッセージ用待ちキューにキューイングされている」と判断し，メッセージの送信処理は行わず，戻り値として E_OBJ を返します。

- メッセージの優先度
  RX850 Pro では，メッセージのキューイング方法として優先度順を指定できます。メッセージの優先度を指定する場合は，メッセージ用待ちキューにキューイングする際のリンク領域の 4 バイトとは別に 2 バイト使用します。したがって，メッセージはメッセージ用領域の先頭から 6 バイト以降に格納してください。なお，メッセージの優先度は 1 ～ 0x7fff の正の整数値で，値が小さいほど優先度が高くなります。

5.4.6　メールボックス情報の獲得

メールボックス情報の獲得は，ref_mbx を発行することにより行います。

- ref_mbx
  パラメータで指定されたメールボックスのメールボックス情報（拡張情報，待ちタスクの有無など）を獲得します。
  次に，メールボックス情報の詳細を示します。

  - 拡張情報
  - 待ちタスクの有無
  - 待ちメッセージの有無
  - キー ID 番号

5.4.7　ID 番号の獲得

メールボックスの ID 番号の獲得は，vget_mid を発行することにより行われます。

- vget_mid
  パラメータで指定されたメールボックスの ID 番号を獲得します。
  システム・コールでメールボックスに対する操作をするとき，メールボックスの ID 番号が必要です。メールボックスを生成する際，ID 番号をユーザが一意に決定するか，自動的に振り当てるかを指定できます。
  しかし，自動的に振り当てると指定した場合，ユーザはメールボックスの ID 番号を知ることができません。
  そのために必要となるものが「キー ID 番号」です。キー ID 番号はメールボックス生成の際に一意に指定します。
  このキー ID 番号をパラメータとして本システム・コールを発行すると，そのキー ID 番号を持つメールボックスの ID 番号を取得できます。

5.4.8　メールボックスによるタスク間通信

次に，メールボックスを使用してタスク間通信を行った場合の動作例を示します。

【 条件 】

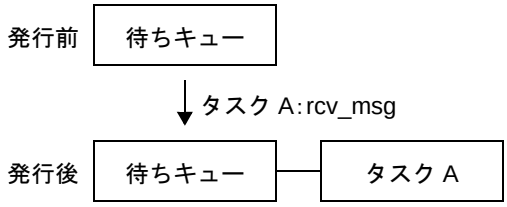
- タスクの優先度
  タスク A ＞ タスク B

- タスクの状態
  タスク A：　run 状態
  タスク B：　ready 状態

- メールボックスの属性
  タスクのキューイング順序：　FIFO 順
  メッセージのキューイング順序：　FIFO 順

（1）タスク A が rcv_msg を発行します。
現在，RX850 Pro が管理している対象メールボックスのメッセージ用待ちキューにはメッセージがキューイングされていません。したがって，RX850 Pro はタスク A を run 状態から wait 状態（メッセージ待ち状態）へと遷移させ，対象メールボックスのタスク用待ちキューの最後尾にキューイングします。
このとき，対象メールボックスのタスク用待ちキューは，図 5 － 8 のようになります。

図 5 － 8　タスク用待ちキューの状態（rcv_msg 発行時）



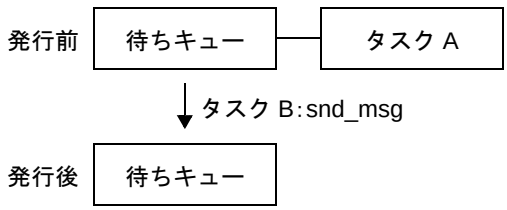
（2）タスク A のメッセージ待ち状態への遷移にともない，タスク B が ready 状態から run 状態へと遷移します。

（3）タスク B が get_blk を発行します。
これにより，メモリ・プールからメッセージ用の領域（メモリ・ブロック）が確保されます。

（4）タスク B がメモリ・ブロックにメッセージを書き込みます。

（5）タスク B が snd_msg を発行します。
これにより，対象メールボックスのタスク用待ちキューにキューイングされていたタスク A が，メッセージ待ち状態から ready 状態へと遷移します。
このとき，対象メールボックスのタスク用待ちキューは，図 5 － 9 のようになります。

図 5 － 9　待ちキューの状態（snd_msg 発行時）

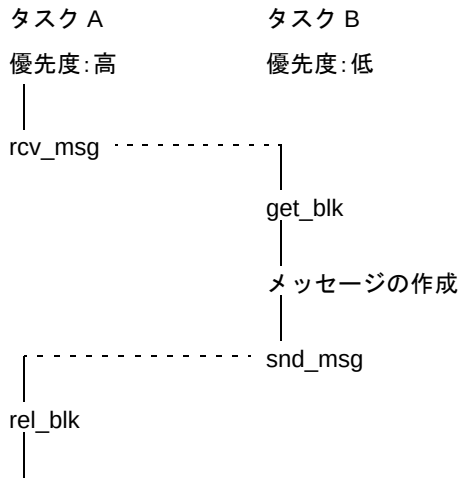


（6）優先度の高いタスク A が ready 状態から run 状態へと遷移します。
なお，タスク B は run 状態から ready 状態へと遷移します。

（7）タスク A が rel_blk を発行します。
　これにより，メッセージ用の領域として確保されていたメモリ・ブロックがメモリ・プールへ解放されます。

図 5 － 10 に，（1）から（7）で説明したタスク間通信の流れを示します。

図 5 － 10　メールボックスによるタスク間通信



第 6 章　割り込み処理管理機能

この章では，RX850 Pro が行う割り込み処理管理機能について説明します。

6.1　概　　要

RX850 Pro が行う割り込み処理管理では，次のような処理が行われます。

- 割り込みハンドラの登録
- 割り込みハンドラの起動
- 割り込みハンドラからの復帰
- 割り込み許可レベルの変更／獲得

6.2　割り込みハンドラ

割り込みハンドラは，割り込みが発生した際ただちに起動される割り込み処理専用ルーチンで，タスクとは独立したものとして扱われます。したがって，システム内で最高優先度を持つタスクが実行中であっても，その処理は中断され，割り込みハンドラに制御が移ります。
なお，割り込みハンドラ内で RX850 Pro の機能（システム・コールの発行など）が使用できないもの（つまり，通常の割り込み処理）を“直接起動割り込みハンドラ”，RX850 Pro の機能が使用できるものを“間接起動割り込みハンドラ”と呼び，区別しています。

- 直接起動割り込みハンドラ
  RX850 Pro が管理していない割り込み処理専用ルーチンです。
  このハンドラ上では，システム・コール実行，間接起動割り込みハンドラの多重割り込みは不可能です。直接起動割り込みハンドラの多重割り込みは可能です。
  RX850 Pro 機能を管理するオーバヘッドがないため，間接起動割り込みハンドラと比べて応答性が高いです。

- 間接起動割り込みハンドラ
  RX850 Pro が管理する割り込み処理専用ルーチンです。
  このハンドラ上では，システム・コール実行，直接起動割り込みハンドラ／間接起動割り込みハンドラの多重割り込みが可能です。
  RX850 Pro 機能を管理するオーバ・ヘッドがあるため，直接起動割り込みハンドラと比べて応答性が低いです。

また，RX850 Pro では，割り込みハンドラ内でシステム・コールが発行された際のスケジューリング処理に特異性を持たせています。
つまり，RX850 Pro では，割り込みハンドラ内でタスクのスケジューリング処理が必要となるシステム・コール（chg_pri，sig_sem など）が発行された際には，待ちキューの操作などといった処理を行うだけであり，実際のスケジューリング処理は，割り込みハンドラからの復帰処理（return 命令の発行など）まで遅延され，一括して行われます。

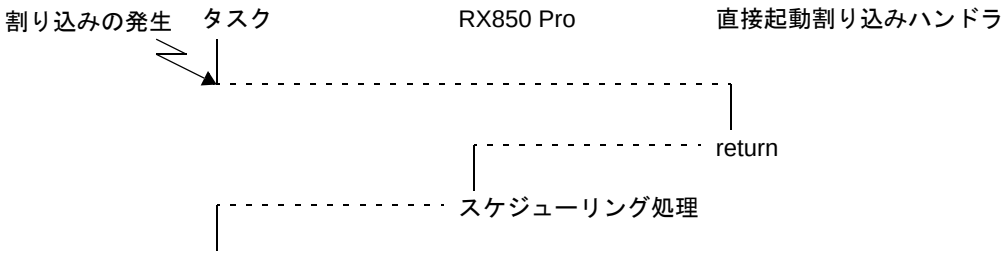
6.2.1　割り込み要因番号

割り込み要因番号は，“(例外コード - 0x80) / 0x10”で算出される割り込み要因を示す番号であり，システム・コンフィギュレーション・ファイルでシステム情報 clkhdr/ システム最大値情報 maxintfactor/ 間接起動割り込みハンドラ情報 inthdr を定義する際，および，処理プログラムから def_int，chg_icr，ref_icr を発行する際に使用されます。

6.3　直接起動割り込みハンドラ

直接起動割り込みハンドラは，割り込みが発生した際，RX850 Pro を介在させることなく起動される割り込み処理専用ルーチンです。このため，ハードウエアの限界に近い高速な応答性が期待されます。
図 6 － 1 に，割り込みハンドラの動作の流れを示します。

図 6 － 1　直接起動割り込みハンドラの動作の流れ



6.3.1　直接起動割り込みハンドラの登録

直接起動割り込みハンドラの登録は，割り込みが発生した際にプロセッサが制御を移すハンドラ・アドレスに直接起動割り込みハンドラを割り付ける，または直接起動割り込みハンドラへの分岐命令を設定することにより行われます。具体的な設定方法については，「B.6　直接起動割り込みハンドラ」を参照してください。

6.3.2　直接起動割り込みハンドラ内での処理

直接起動割り込みハンドラの処理を記述する際には，次の点に注意してください。

- レジスタの退避と復帰
  RX850 Pro では，直接起動割り込みハンドラの起動，および，復帰に際して，関与していません。したがって，直接起動割り込みハンドラ内で作業用レジスタを使用する場合は，直接起動割り込みハンドラの開始部分で作業用レジスタの退避処理を，終了部分で作業用レジスタの復帰処理を記述する必要があります。

- スタックの切り替え
  直接起動割り込みハンドラが使用する sp（スタック・ポインタ）は割り込みが発生した時点における値となります。

- システム・コールの発行制限
  RX850 Pro では，直接起動割り込みハンドラ内でシステム・コールを発行することを禁止しています。

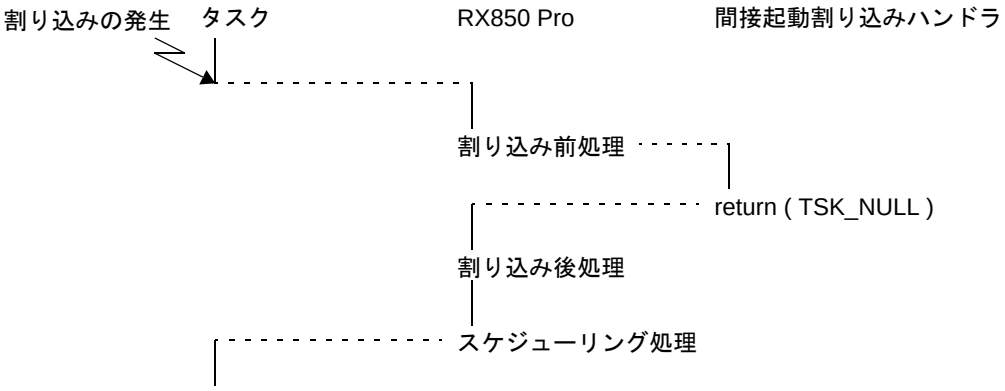
- 直接起動割り込みハンドラからの復帰処理
  直接起動割り込みハンドラを C 言語で記述した場合は，ハンドラの終了部分で return 命令を，アセンブリ言語で記述した場合は，reti 命令を発行することにより行われます。

  備考　直接起動割り込みハンドラが使用する gp（グローバル・ポインタ），tp（テキスト・ポインタ）は，割り込みが発生した時点における値となります。

6.4　間接起動割り込みハンドラ

間接起動割り込みハンドラは，割り込みが発生した際，RX850 Pro による割り込み前処理（レジスタの退避処理，スタックの切り替え処理など）を行わせたあと起動される割り込み処理専用ルーチンです。
このため，直接起動割り込みハンドラに比べて応答性の面では多少劣りますが，RX850 Pro による割り込み前処理が行われているため，ハンドラ内でシステム・コールを発行できるといった利点を有しています。
図 6 － 2 に，間接起動割り込みハンドラの動作の流れを示します。

図 6 － 2　間接起動割り込みハンドラの動作の流れ



6.4.1　間接起動割り込みハンドラの登録

RX850 Pro では，間接起動割り込みハンドラの登録において，「システム初期化処理（ニュークリアス初期化部）において，スタティックに登録する」，「処理プログラム内からシステム・コールを発行して，ダイナミックに登録する」の 2 種類のインタフェースを用意しています。
RX850 Pro における間接起動割り込みハンドラの登録とは，間接起動割り込みハンドラを管理するための領域（管理オブジェクト）をシステム・メモリ領域から確保したあと，初期化することです。

- スタティックに登録する場合
  間接起動割り込みハンドラをスタティックに登録する場合，コンフィギュレーション時に間接起動割り込みハンドラ情報として指定します。
  RX850 Pro では，システム初期化処理時，情報ファイル（システム情報テーブル，システム情報ヘッダ・ファイル）に定義された情報を基に間接起動割り込みハンドラの登録処理を行い，管理対象とします。

- ダイナミックに登録する場合
  間接起動割り込みハンドラをダイナミックに登録する場合，処理プログラム（タスク，非タスク）内から def_int を発行します。
  RX850 Pro では，def_int 発行時，パラメータで指定された情報を基に間接起動割り込みハンドラの登録処理を行い，管理対象とします。

備考　間接起動割り込みハンドラ情報で定義された割り込みハンドラ，および，システム情報で定義されたタイマの割り込み要因番号に対応したクロック・ハンドラについては，コンフィギュレータが該当割り込みエントリをシステム情報テーブルに自動出力しているため，ユーザが該当割り込みエントリを記述する必要がありません。
ただし，コンフィギュレータの起動オプションに -ne が指定された際には，システム情報テーブルに対する割り込みエントリの出力が抑制されるため，ユーザが独自に割り込みエントリを記述する必要があります。
なお，割り込みエントリについての詳細は，「B.7　間接起動割り込みハンドラ」を参照してください。

6.4.2　間接起動割り込みハンドラ内での処理

間接起動割り込みハンドラの処理を記述する際には，次の点に注意してください。

- レジスタの退避と復帰
  RX850 Pro では，間接起動割り込みハンドラに制御を移すときと，間接起動割り込みハンドラから復帰するときに C コンパイラの関数呼び出し規約に従った，作業レジスタの退避処理，復帰処理を行っています。したがって，間接起動割り込みハンドラの開始部分で作業用レジスタの退避処理を，終了部分で作業用レジスタの復帰処理を記述する必要がありません。

- スタックの切り替え
  RX850 Pro では，間接起動割り込みハンドラに制御を移すときと割り込みハンドラから復帰するときに，スタックの切り替え処理を行います。したがって，間接起動割り込みハンドラの開始部分で割り込みハンドラ用スタックへの切り替え処理を，終了部分で割り込み発生時のスタックへの切り替え処理を記述する必要はありません。
  ただし，コンフィギュレーション時に割り込みハンドラ用スタックを定義していない場合は，スタックの切り替え処理は行われず，割り込み発生時のスタックが使用されます。

- システム・コールの発行制限
  次に，間接起動割り込みハンドラ内で発行可能なシステム・コールの一覧を示します。

  - タスク管理機能
    sta_tsk，chg_pri，rot_rdq，rel_wai，get_tid，ref_tsk，vget_tid

  - タスク付属同期機能
    sus_tsk，rsm_tsk，frsm_tsk，wup_tsk，can_wup

  - 同期通信機能
    sig_sem, preq_sem, ref_sem, vget_sid, set_flg, clr_flg, pol_flg, ref_flg, vget_fid, snd_msg, prcv_msg, ref_mbx, vget_mid

  - 割り込み処理管理機能
    def_int，ena_int，dis_int，chg_icr，ref_icr

  - メモリ・プール管理機能
    pget_blk，rel_blk，ref_mpl，vget_pid

  - 時間管理機能
    set_tim，get_tim，def_cyc，act_cyc，ref_cyc

  - システム管理機能
    get_ver，ref_sys，def_svc，viss_svc

- 間接起動割り込みハンドラからの復帰処理
  間接起動割り込みハンドラからの復帰処理は，ハンドラの終了部分で return 命令を発行することにより行われます。

  - return (TSK_NULL) 命令
    間接起動割り込みハンドラから復帰します。

  - return (ID tskid) 命令
    パラメータで指定されたタスクに起床要求を発行したあと，間接起動割り込みハンドラから復帰します。

  RX850 Pro では，間接起動割り込みハンドラ内でタスクのスケジューリング処理が必要となるシステム・コール（chg_pri，sig_sem など）が発行された場合，待ちキューの操作などの処理が行われるだけであり，実際のスケジューリング処理は，間接起動割り込みハンドラからの復帰処理（return 命令の発行）まで遅延され，一括して行われます。

  備考　return 命令では，外部割り込みコントローラに対する処理終了通知（EOI コマンドの発行）は行いません。したがって，外部割り込み要求によって起動した間接起動割り込みハンドラから復帰する場合は，これらのシステム・コールを発行する前に，外部割り込みコントローラに対し処理終了を通知してください。

6.5　マスカブル割り込みの受け付け禁止／再開

RX850 Pro では，ユーザの処理プログラムからマスカブル割り込みの受け付け状態を操作し，マスカブル割り込みの受け付けを禁止／再開する機能が提供されています。
この機能は，次に示すシステム・コールをタスク，またはハンドラ内から発行することにより使用できます。

- loc_cpu
  マスカブル割り込みの受け付けを禁止したあと，ディスパッチ処理（タスクのスケジューリング処理）を禁止します。これにより，本システム・コールが発行されてから unl_cpu が発行されるまでの間は，ほかのタスクやハンドラに制御が移ることはありません。

- unl_cpu
  マスカブル割り込みの受け付けを許可したあと，ディスパッチ処理（タスクのスケジューリング処理）を再開します。これにより，loc_cpu の発行により禁止されていたマスカブル割り込みの受け付けが許可されるとともに，ディスパッチ処理も再開されます。

図 6 － 3 に割り込みをマスクしない場合（通常時）の動作の流れを，図 6 － 4 に loc_cpu を発行した場合の動作の流れを示します。

図 6 － 3　割り込みのマスク処理をしない場合（通常時）の動作の流れ

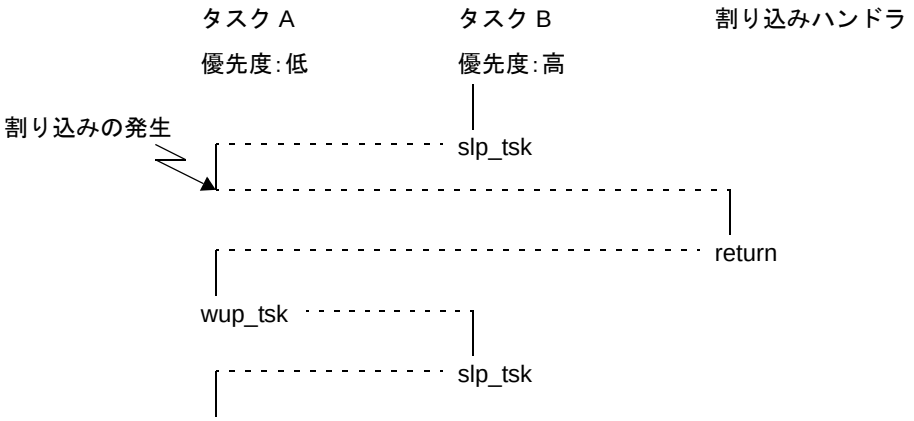
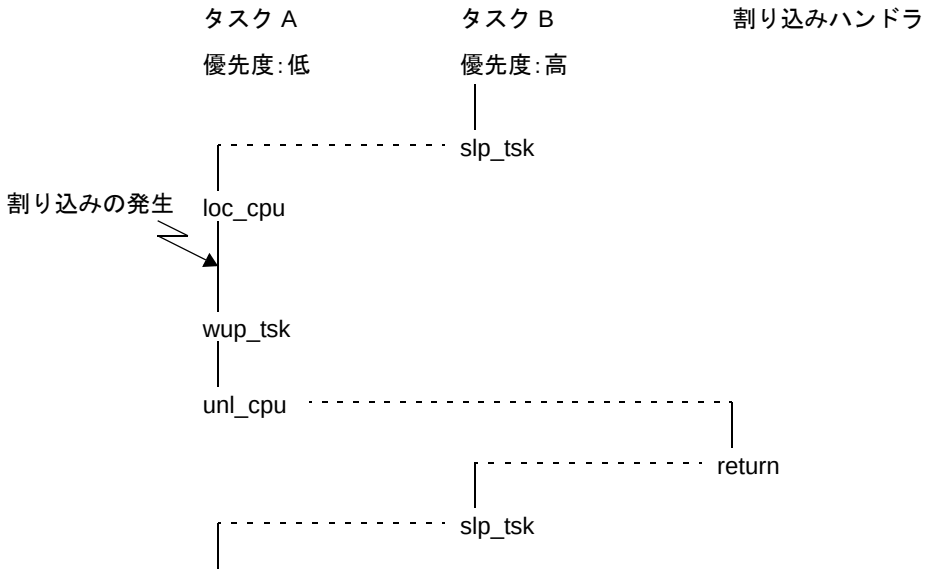


図 6 － 4　loc_cpu を発行した場合の動作の流れ



6.6　割り込み制御レジスタの変更／獲得

割り込み制御レジスタの変更／獲得は，chg_icr，または ref_icr を発行することにより行われます。

- chg_icr
  パラメータで指定された割り込み制御レジスタの内容を変更します。

- ref_icr
  パラメータで指定された割り込み制御レジスタの内容を獲得します。

備考　V850ES/V850E1/V850E2 コアで動作させている場合，割り込み制御レジスタ関連のシステム・コールである chg_icr，および ref_icr を発行すると，期待した割り込み制御レジスタが操作されない場合があります。RX850 Pro では，割り込み要因番号から割り込み制御レジスタのアドレスを算出しています。しかし，V850ES/V850E1/V850E2 コアの場合，割り込み要因番号と割り込み制御レジスタの並びが，他の V850 マイクロコントローラと異なっているため，正しいレジスタ・アドレスが得られなくなっています。そのため，V850ES/V850E1/V850E2 コアにおける chg_icr，および ref_icr の使用を制限します。割り込み制御レジスタをアプリケーション上から操作したい場合は，これらのシステム・コールは使わずに，直接レジスタを操作してください。

6.7　ノンマスカブル割り込み

ノンマスカブル割り込みは，割り込み優先順位の対象外となっており，すべての割り込みに優先して受け付けられます。また，プロセッサを割り込み禁止（psw の ID フラグをセット）状態にしても受け付けられます。したがって，RX850 Pro の処理中や割り込みハンドラの処理中であっても，ノンマスカブル割り込みは受け付けられます。

RX850 Pro では，ノンマスカブル割り込みに対応した割り込みハンドラ内でシステム・コールを発行した場合，その動作は保証していません。

6.8　クロック割り込み

RX850 Pro では，ハードウエア（クロック・コントローラなど）により一定周期で発生するクロック割り込みを利用して時間管理を行っています。

つまり，クロック割り込みが発生した場合には，RX850 Pro のシステム・クロック処理が呼び出され，タスクの時間経過待ち，周期起動ハンドラの起動などといった時間に関連した処理が行われます。

なお，時間管理についての詳細は，「第 8 章　時間管理機能」を参照してください。

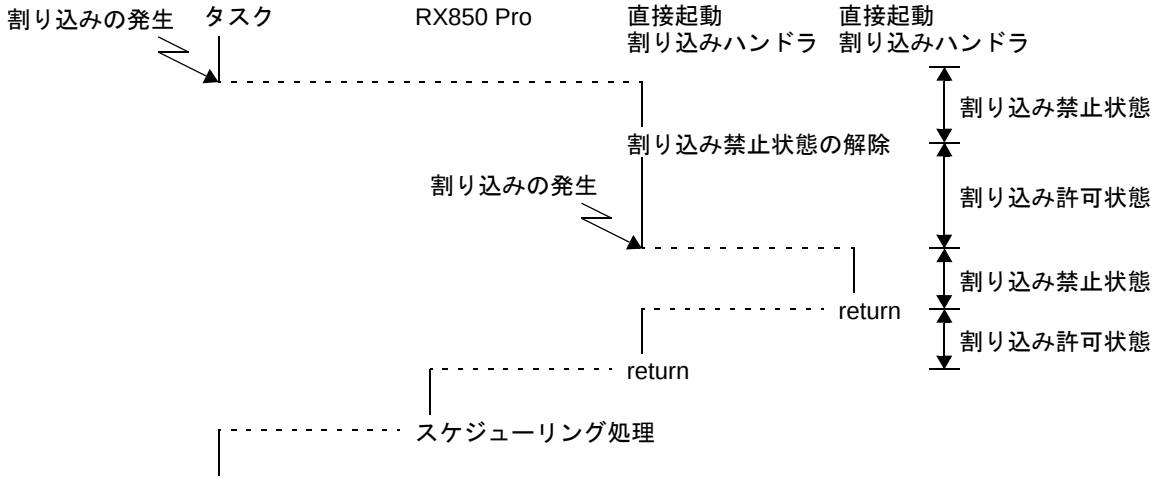
6.9　多重割り込み

割り込みハンドラ内でさらに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。RX850 Pro は多重割り込みにも対応します。

ただし，割り込みハンドラは，割り込み禁止（psw の ID フラグをセット）状態で処理を開始するため，多重割り込みを受け付けるには，割り込みハンドラ内で割り込み禁止状態の解除処理を記述する必要があります。

図 6 － 5 に，多重割り込み発生時の動作の流れを示します。

図 6 － 5　多重割り込み発生時の動作の流れ



第 7 章　メモリ・プール管理機能

この章では，RX850 Pro が行うメモリ・プール管理機能について説明します。

7.1　概　　要

RX850 Pro では，システムを管理するための情報テーブル，各種機能を実現するための管理ブロックを配置するためのメモリ領域，そしてメモリ・プールを使用するためのメモリ領域が必要となります。
これらを配置するメモリ領域には，次にあげる 4 種類があります。

- System Memory Pool 0（キーワード：SPOL0）
- System Memory Pool 1（キーワード：SPOL1）
- User Memory Pool 0（キーワード：UPOL0）
- User Memory Pool 1（キーワード：UPOL1）

これらのメモリ領域には「各種資源管理ブロック」,「タスク・スタック」,「割り込みハンドラ・スタック」,「メモリ・プール用のメモリ」の 4 種類が配置されます。また，配置できる領域の組み合わせは次のとおりです。

表 7 － 1　メモリ情報の配置組み合わせ

各種資源管理ブロック	タスク・スタック	割り込みスタック	メモリ・プール
SPOL0	SPOL0，または SPOL1	SPOL0，または SPOL1	UPOL0，または UPOL1

それぞれのメモリ領域の先頭アドレスとサイズは，コンフィギュレーション・ファイルにて設定します。SPOL0 は必ず作成する必要があります。SPOL1 はタスク・スタック，割り込みスタックを SPOL0 以外に配置したい場合に作成します。UPOL0 と UPOL1 はメモリ・プール管理機能を使用したい場合に作成します。なお，UPOL1 は UPOL0 を作成している場合に作成可能です。

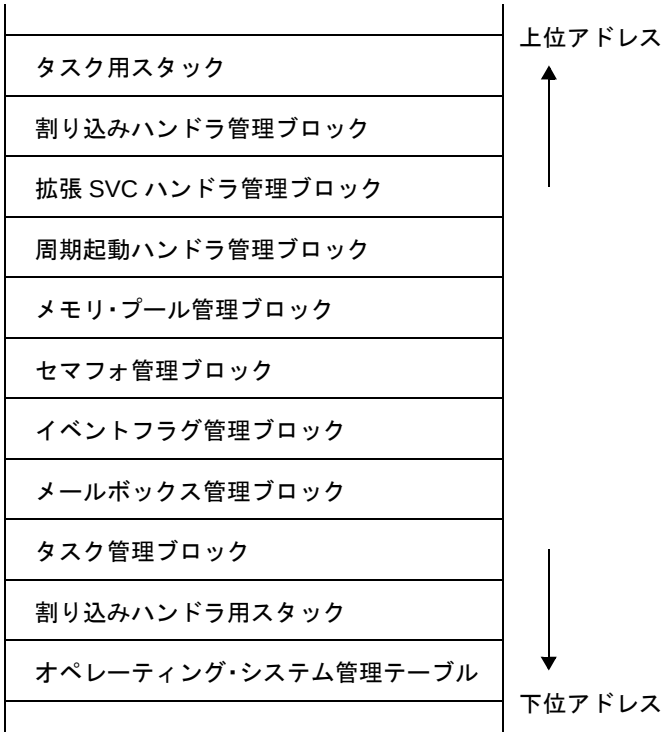
7.2　管理オブジェクト

RX850 Pro が提供する機能を実現するうえで必要となる管理オブジェクトを示します。これらの管理オブジェクトは，コンフィギュレーション時に指定した情報をもとに，システム初期化処理時に生成／初期化されます。
また，これらの管理オブジェクトは SPOL0（タスク用スタックと割り込みハンドラ用スタックは SPOL1 でも可）に配置されます。

- オペレーティング・システム管理テーブル
- タスク管理ブロック
- セマフォ管理ブロック
- イベントフラグ管理ブロック
- メールボックス管理ブロック
- メモリ・プール管理ブロック
- メモリ・ブロック管理ブロック
- 周期起動ハンドラ管理ブロック
- 拡張 SVC ハンドラ管理ブロック
- メモリ・プール
- タスク用スタック
- 割り込みハンドラ用スタック
- 割り込みハンドラ管理ブロック

図 7 － 1 に，管理オブジェクトの配置例を示します。

図 7 － 1　管理オブジェクトの配置例



7.3　メモリ・プールとメモリ・ブロック

RX850 Pro では，アプリケーション中でメモリ領域を獲得，解放するダイナミックなメモリ・プール管理機能を行っています。作業用のメモリ領域が必要となったときに獲得し，不要になったときに解放できるといった機能が用意されています。この機能により，限りあるメモリ領域を効率良く使用できます。

メモリ・プールとして使用できるメモリ領域は，UPOL0，または UPOL1 です。コンフィギュレーション時にメモリ・プール情報を定義して，または cre_mpl を発行してメモリ・プールを作成するときに，UPOL0，UPOL1 のどちらの領域を使うものかを指定します。

RX850 Pro が提供しているメモリ・プールは可変長メモリ・プールです。固定長メモリ・プールは搭載していません。

メモリ・プールは複数のメモリ・ブロックから構成されており，メモリ・プールに対する操作はメモリ・ブロックを単位として行います。

メモリ・プールに対するダイナミックな操作は，次に示すメモリ・プール関連のシステム・コールを使用します。

cre_mpl：　メモリ・プールを生成する
del_mpl：　メモリ・プールを削除する
get_blk：　メモリ・ブロックを獲得する
pget_blk：　メモリ・ブロックを獲得する（ポーリング）
tget_blk：　メモリ・ブロックを獲得する（タイムアウトあり）
rel_blk：　メモリ・ブロックを解放する
ref_mpl：　メモリ・プール情報を獲得する
vget_pid：　メモリ・プールの ID 情報を獲得する

7.3.1　メモリ・プールの生成

RX850 Pro では，メモリ・プールの生成において，「システム初期化処理（ニュークリアス初期化部）において，スタティックに生成する」，「処理プログラム内からシステム・コールを発行し，ダイナミックに生成する」の 2 種類のインタフェースを用意しています。
RX850 Pro におけるメモリ・プールの生成とは，メモリ・プールを管理するための領域（管理オブジェクト）とメモリ・プールの実体を，システム・メモリ領域から確保したあと，初期化することです。

- スタティックに登録する場合
  メモリ・プールをスタティックに登録する場合，コンフィギュレーション時にメモリ・プール情報として指定します。
  RX850 Pro では，システム初期化処理時，情報ファイル（システム情報テーブル，システム情報ヘッダ・ファイル）に定義された情報を基にメモリ・プールの生成処理を行い，管理対象とします。

- ダイナミックに登録する場合
  メモリ・プールをダイナミックに登録する場合，処理プログラム（タスク）内から cre_mpl を発行します。
  RX850 Pro では，cre_mpl 発行時，パラメータで指定された情報を基にメモリ・プールの生成処理を行い，管理対象とします。

備考　メモリ・プールを生成すると，指定サイズ以外に RX850 Pro がメモリ・プール管理領域として，メモリ・プールの先頭から 8 バイトを使用します。したがって，生成したメモリ・プールのサイズは「指定サイズ + 8 バイト」となります。

7.3.2　メモリ・プールの削除

メモリ・プールの削除は，del_mpl を発行することにより行われます。

- del_mpl
  パラメータで指定されたメモリ・プールを削除します。
  これにより，対象メモリ・プールは，RX850 Pro の管理対象から除外されます。
  ただし，本システム・コールを発行した際，対象メモリ・プールの待ちキューにタスクがキューイングされていた場合には，該当タスクを待ちキューから外すとともに，wait 状態（メモリ・ブロック待ち状態）から ready 状態へと遷移させます。
  なお，wait 状態を解除されたタスクには，wait 状態へと遷移するきっかけとなったシステム・コール（get_blk，tget_blk）の戻り値として E_DLT が返されます。
  また，本システム・コールを発行すると，対象メモリ・プールが管理しているメモリ・ブロックも RX850 Pro の管理対象から除外されます。本システム・コールを発行する前にすでにタスクが対象メモリ・プールからメモリ・ブロックを獲得していた場合は，そのメモリ・ブロックは動作保証外となるので，メモリ・プールの削除については注意が必要です。

7.3.3　メモリ・ブロックの獲得

メモリ・ブロックの獲得は，get_blk，pget_blk，または tget_blk を発行することにより行います。

備考　RX850 Pro では，メモリ・ブロックを獲得するときにメモリ・クリアを行いません。したがって，獲得したメモリ・ブロックの内容は不定です。

メモリ・ブロックを獲得すると，要求サイズ以外に RX850 Pro がメモリ管理領域としてメモリ・プールから 8 バイト使用します。また，RX850 Pro は要求サイズを自動的に 4 バイトでアラインするので，残りの空きメモリ・ブロック・サイズには注意してください。
次に，獲得したメモリ・ブロックのサイズを求める式を示します。

メモリ・ブロックのサイズ（blk_siz）= align 4 (ユーザ要求サイズ) + 8

- get_blk
  パラメータで指定されたメモリ・プールからメモリ・ブロックを獲得します。
  ただし，本システム・コールを発行した際，対象メモリ・プールからメモリ・ブロックを獲得できなかった（要求した空き領域が存在しなかった）場合には，自タスクを対象メモリ・プールの待ちキューにキューイングしたあと，run 状態から wait 状態（メモリ・ブロック待ち状態）へと遷移させます。
  なお，メモリ・ブロック待ち状態は次の場合に解除され，ready 状態へと遷移します。

- rel_blk が発行され，要求サイズのメモリ・ブロックが返却された
- del_mpl が発行され，対象メモリ・プールが削除された
- rel_wai が発行され，強制的に待ち状態が解除された

備考　自タスクを対象メモリ・プールの待ちキューにキューイングする際は，対象メモリ・プール生成時（コンフィギュレーション時，または cre_mpl 発行時）に指定された順（FIFO 順，または優先度順）に行われます。

- pget_blk
パラメータで指定されたメモリ・プールからメモリ・ブロックを獲得します。
ただし，本システム・コールを発行した際，対象メモリ・プールからメモリ・ブロックを獲得できなかった（要求したサイズの空き領域が存在しなかった）場合には，戻り値として E_TMOUT が返されます。

- tget_blk
パラメータで指定されたメモリ・プールからメモリ・ブロックを獲得します。
ただし，本システム・コールを発行した際，対象メモリ・プールからメモリ・ブロックを獲得できなかった（要求したサイズの空き領域が存在しなかった）場合には，自タスクを対象メモリ・プールの待ちキューにキューイングしたあと，run 状態から wait 状態（メモリ・ブロック待ち状態）へと遷移させます。
なお，メモリ・ブロック待ち状態は次の場合に解除され，ready 状態へと遷移します。

- パラメータで指定された待ち時間が経過した
- rel_blk が発行され，要求サイズのメモリ・ブロックが返却された
- del_mpl が発行され，対象メモリ・プールが削除された
- rel_wai が発行され，強制的に待ち状態が解除された

備考　自タスクを対象メモリ・プールの待ちキューにキューイングする際は，対象メモリ・プール生成時（コンフィギュレーション時，または cre_mpl 発行時）に指定された順（FIFO 順，または優先度順）に行われます。

7.3.4　メモリ・ブロックの返却

メモリ・ブロックの返却は，rel_blk を発行することにより行われます。

- rel_blk
パラメータで指定されたメモリ・プールにメモリ・ブロックを返却します。
ただし，本システム・コールを発行した際，対象メモリ・プールの待ちキューにキューイングされているタスク（待ちキューの先頭タスク）が要求したサイズを満足するようなメモリ・ブロックであった場合には，該当タスクにメモリ・ブロックを渡します。
これにより，該当タスクは待ちキューから外れ，wait 状態（メモリ・ブロック待ち状態）から ready 状態へ，または wait_suspend 状態から suspend 状態へと遷移します。

備考 1　RX850 Pro では，メモリ・ブロックを返却するときにメモリ・クリアを行いません。したがって，返却したメモリ・ブロックの内容は不定です。

備考 2　RX850 Pro には 2 つの仕様の rel_blk があります。

（1）rel_blk でメモリ・ブロックを返却する際，メモリ・ブロックの先頭 4 バイトが 0 でなかった場合に，メモリ・ブロックを返却せずに戻り値 E_OBJ で終了する。

（2）rel_blk を発行すると，メモリ・ブロックの先頭 4 バイトが 0 でなくてもメモリ・ブロックを返却する（戻り値 E_OK）。

（1）はメモリ・ブロックがメールボックスのメッセージ領域として使用されていた場合を考慮したもので，これまでの RX850 Pro に搭載されていた rel_blk と同じ仕様のものです。
メモリ・ブロックがメールボックスのメッセージ領域として使用されていた場合，先頭 4 バイトがメッセージの待ちキューのリンク領域になります。つまりメッセージが，メールボックスにキューイングされているときに rel_blk を発行し，必ずメモリ・ブロックが返却する仕様にすると，キューにつながれていたメッセージ領域が返却されてしまうことになります。
これを防ぐため，リンク領域である先頭 4 バイトが 0 でなければ，メッセージ領域として使用されているメモリ・ブロックと判断し，メモリ・ブロックを返却せずに戻り値 E_OBJ で終了します。
この仕様の rel_blk を使用してメモリ・ブロックを返却するときは，必ず先頭 4 バイトを 0 クリアする必要があります。
これらの仕様の rel_blk は，別々のライブラリに搭載されていますので，どちらか一方の rel_blk を使用することになります。使用する方のライブラリをリンクしてください。

（1）メモリ・ブロックの先頭 4 バイトを 0 でクリアする必要のある rel_blk が搭載されたライブラリ
→ librxp.a

（2）メモリ・ブロックの先頭 4 バイトを 0 でクリアしなくてもよい rel_blk が搭載されたライブラリ
→ librxpm.a

備考 3　メモリ・ブロックを返却するメモリ・プールは，get_blk，pget_blk，tget_blk を発行した際に指定したメモリ・プールと同じにしてください。

7.3.5　メモリ・プール情報の獲得

メモリ・プール情報の獲得は，ref_mpl を発行することにより行われます。

- ref_mpl
  パラメータで指定されたメモリ・プールのメモリ・プール情報（拡張情報，待ちタスクの有無など）を獲得します。
  次に，メモリ・プール情報の詳細を示します。

  - 拡張情報
  - 待ちタスクの有無
  - 空き領域の合計サイズ
  - 獲得可能な最大メモリ・ブロック・サイズ
  - キー ID 番号

7.3.6　ID 番号の獲得

メモリ・プールの ID 番号の獲得は，vget_pid を発行することにより行われます。

- vget_pid
  パラメータで指定されたメモリ・プールの ID 番号を獲得します。
  システム・コールでメモリ・プールに対する操作をするとき，メモリ・プールの ID 番号が必要です。メモリ・プールを生成する際，ID 番号をユーザが一意に決定するか，自動的に振り当てるかを指定できます。
  しかし，自動的に振り当てると指定した場合，ユーザはメモリ・プールの ID 番号を知ることができません。
  そのために必要となるものが「キー ID 番号」です。キー ID 番号はメモリ・プール生成の際に一意に指定します。
  このキー ID 番号をパラメータとして本システム・コールを発行すると，そのキー ID 番号を持つメモリ・プールの ID 番号を取得できます。

7.3.7　メモリ・プールによるメモリ・ブロックの動的管理

次に，メモリ・プールを使用してタスクでのメモリの動的使用を行った場合の動作例を示します。

【 条件 】

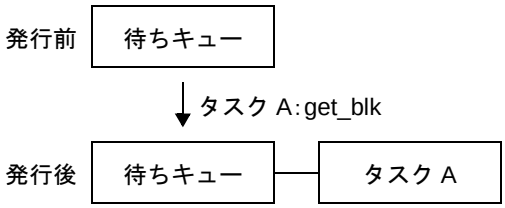
- タスクの優先度
  タスク A ＞ タスク B

- タスクの状態
  タスク A：　run 状態
  タスク B：　ready 状態

- メモリ・プールの属性
  空きメモリ・ブロック・サイズ：　0x20
  タスクのキューイング順序：　FIFO 順

（1）タスク A が get_blk を発行します。
要求メモリ・ブロック・サイズは「0x30」とします。
現在，RX850 Pro が管理している対象メモリ・プールの空きメモリ・ブロック・サイズは「0x20」です。
したがって，RX850 Pro はタスク A を run 状態から wait 状態（メモリ・ブロック待ち状態）へと遷移させ，対象メモリ・プールの待ちキューの最後尾にキューイングします。
このとき，対象メモリ・プールの待ちキューは，図 7 － 2 のようになります。

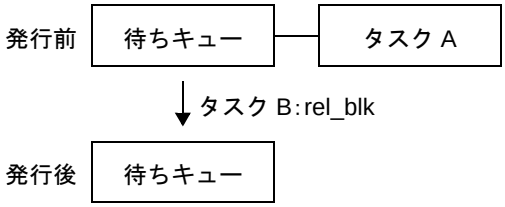
図 7 － 2　待ちキューの状態（get_blk 発行時）



（2）タスク A のメモリ・プール待ち状態への遷移にともない，タスク B が ready 状態から run 状態へと遷移します。

（3）タスク B が rel_blk を発行します。
返却メモリ・ブロック・サイズは，「0x16」とします。
これにより，対象メモリ・プールの待ちキューにキューイングされているタスク A の要求メモリ・ブロック・サイズを満たしたため，タスク A がメモリ・ブロック待ちから ready 状態へと遷移します。
このとき，対象メモリ・プールの待ちキューは，図 7 － 3 のようになります。

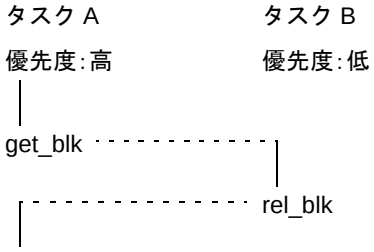
図 7 － 3　待ちキューの状態（rel_blk 発行時）



（4）優先度の高いタスク A が ready 状態から run 状態へと遷移します。
なお，タスク B は run 状態から ready 状態へと遷移します。

図 7 － 4 に，（1）から（4）で説明したメモリ・プールによるメモリの動的使用の流れを示します。

図 7 － 4　メモリ・プールによるメモリの動的使用



第 8 章　時間管理機能

この章では，RX850 Pro が行う時間管理機能について説明します。

8.1　概　　要

RX850 Pro における時間管理は，ハードウエア（クロック・コントローラなど）により一定周期で発生するクロック割り込みを利用して行います。

クロック割り込みが発生すると，RX850 Pro のシステム・クロック処理が呼び出され，システム・クロックの更新やタスクの遅延起床，周期起動ハンドラの起動などといった，時間に関連した処理が行われます。

8.2　システム・クロック

システム・クロックは，RX850 Pro が時間管理の際に使用する時刻（48 ビット幅，単位：ミリ秒）を保持したソフトウエア・タイマです。

システム・クロックは，システム初期化処理で「0x0」に設定されたあと，システム・クロック処理で基本クロック周期（コンフィギュレーション時にシステム情報で指定）を単位として更新されます。

備考　RX850 Pro が管理するシステム・クロックは，48 ビット幅で構成されています。このため，RX850 Pro では，桁あふれした数値（48 ビットでは表せない数値）は無視されます。

8.2.1　システム・クロックの設定と読み出し

システム・クロックの設定は set_tim を，読み出しは get_tim を発行することにより行います。

- set_tim
  システム・クロックにパラメータで指定された時刻を設定します。

- get_tim
  システム・クロックの現時刻をパラメータで指定されるパケットに格納します。

8.3　タイマ・オペレーション

リアルタイム処理では，あるタスクの処理を一定時間だけ中断させたり，あるハンドラの処理を一定周期で実行させたりといった，時間と同期した機能（タイマ・オペレーション機能）が必要となります。そこで，RX850 Pro では，タイマ・オペレーション機能として，タスクの遅延起床，タイムアウト，周期起動ハンドラの起動といった機能を提供しています。

8.4　タスクの遅延起床

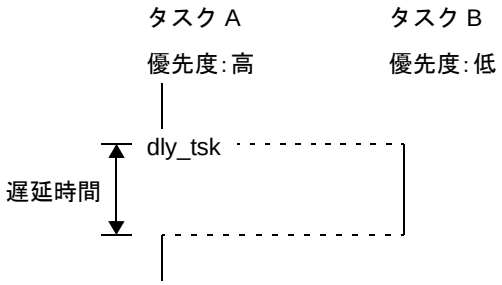
タスクの遅延起床とは，一定の時間が経過するまでの間，タスクを run 状態から wait 状態（時間経過待ち状態）へと遷移させ，時間が経過した時点で wait 状態を解除し，ready 状態へ遷移させるものです。
タスクの遅延起床は，dly_tsk を発行することにより行います。

- dly_tsk
  パラメータで指定された遅延時間だけ，自タスクを run 状態から wait 状態（時間経過待ち状態）へと遷移させます。
  なお，時間経過待ち状態は次の場合に解除され，ready 状態へ遷移します。

  - パラメータで指定された遅延時間が経過した
  - rel_wai が発行され，強制的に待ち状態が解除された

図 8 － 1 に，本システム・コールを発行した際の動作の流れを示します。

図 8 － 1　dly_tsk 発行時の動作の流れ



8.5　タイムアウト

タイムアウトは，要求する条件が即時に成立しなかった場合，一定の時間が経過するまでの間，タスクを run 状態から wait 状態（起床待ち状態，資源待ち状態など）へと遷移させ，時間が経過したときには wait 状態を解除し，ready 状態へと遷移させるものです。
タイムアウトは，tslp_tsk，twai_sem，twai_flg，trcv_msg，tget_blk を発行することにより行われます。

- tslp_tsk
  自タスクに発行されている起床要求を 1 回分だけ解除（起床要求カウンタから 0x1 を減算）します。
  ただし，本システム・コールを発行した際，自タスクの起床要求カウンタが 0x0 であった場合には，起床要求の解除（起床要求カウンタの減算）は行わず，自タスクを run 状態から wait 状態（起床待ち状態）へと遷移させます。
  なお，起床待ち状態は次の場合に解除され，ready 状態へ遷移します。

  - パラメータで指定された時間が経過した
  - wup_tsk が発行された
  - rel_wai が発行され，強制的に待ち状態が解除された

- twai_sem
  パラメータで指定されたセマフォから資源を獲得（セマフォ・カウンタから 0x1 を減算）します。
  ただし，本システム・コールを発行した際，対象セマフォから資源を獲得できなかった（空き資源が存在しなかった）場合には，自タスクを対象セマフォの待ちキューにキューイングしたあと，run 状態から wait 状態（資源待ち状態）へと遷移させます。
  なお，資源待ち状態は次の場合に解除され，ready 状態へ遷移します。

  - パラメータで指定された待ち時間が経過した
  - sig_sem が発行された
  - del_sem が発行され，対象セマフォが削除された
  - rel_wai が発行され，強制的に待ち状態が解除された

- twai_flg
  パラメータで指定されたイベントフラグに，要求する待ち条件を満足するビット・パターンがセットされているかどうかをチェックします。
  ただし，本システム・コールを発行した際，対象イベントフラグのビット・パターンが待ち条件を満足していなかった場合には，自タスクを対象イベントフラグの待ちキューの最後尾にキューイングしたあと，run 状態から wait 状態（イベントフラグ待ち状態）へと遷移させます。
  なお，イベントフラグ待ち状態は次の場合に解除され，ready 状態へ遷移します。

  - パラメータで指定された待ち時間が経過した
  - set_flg が発行され，要求する待ち条件がセットされた
  - del_flg が発行され，対象イベントフラグが削除された
  - rel_wai が発行され，強制的に待ち状態が解除された

- trcv_msg
  パラメータで指定されたメールボックスからメッセージを受信します。
  ただし，本システム・コールを発行した際，対象メールボックスからメッセージを受信できなかった（メッセージ用待ちキューにメッセージが存在しなかった）場合には，自タスクを対象メールボックスのタスク用待ちキューにキューイングしたあと，run 状態から wait 状態（メッセージ待ち状態）へと遷移させます。
  なお，メッセージ待ち状態は次の場合に解除され，ready 状態へ遷移します。

  - パラメータで指定された待ち時間が経過した
  - snd_msg が発行された
  - del_mbx が発行され，対象メールボックスが削除された
  - rel_wai が発行され，強制的に待ち状態が解除された

- tget_blk
  パラメータで指定されたメモリ・プールからメモリ・ブロックを獲得します。
  ただし，本システム・コールを発行した際，対象メモリ・プールからメモリ・ブロックを獲得できなかった（要求したサイズの空き領域が存在しなかった）場合には，自タスクを対象メモリ・プールの待ちキューにキューイングしたあと，run 状態から wait 状態（メモリ・ブロック待ち状態）へと遷移させます。
  なお，メモリ・ブロック待ち状態は次の場合に解除され，ready 状態へ遷移します。

  - パラメータで指定された待ち時間が経過した
  - rel_blk が発行され，要求するサイズのメモリ・ブロックが返却された
  - del_mpl が発行され，対象メモリ・プールが削除された
  - rel_wai が発行され，強制的に待ち状態が解除された

8.6　周期起動ハンドラ

周期起動ハンドラは，一定の起動時間に達した際，ただちに起動される周期処理専用ルーチンで，ユーザが記述する周期的な処理プログラムの中で，実行開始までのオーバヘッドが最も小さな処理プログラムです。
周期起動ハンドラはタスクとは独立したものとして扱われます。このため，ハンドラの起動時間に達したときには，システム内で最高優先度を持つタスクが実行中であっても，その処理が中断されて周期起動ハンドラに制御が移ります。
周期起動ハンドラに対するダイナミックな操作には，次に示す周期起動ハンドラ関連のシステム・コールや命令を使用します。

def_cyc：　周期起動ハンドラを登録する
act_cyc：　周期起動ハンドラの活性状態を制御する
ref_cyc：　周期起動ハンドラ情報を獲得する
return：　周期起動ハンドラから復帰する

8.6.1　周期起動ハンドラの登録

RX850 Pro では，周期起動ハンドラの登録において，「システム初期化処理（ニュークリアス初期化部）において，スタティックに登録する」，「処理プログラム内からシステム・コールを発行し，ダイナミックに登録する」の 2 種類のインタフェースを用意しています。
RX850 Pro における周期起動ハンドラの登録とは，周期起動ハンドラを管理するための領域（管理オブジェクト）をシステム・メモリ領域から確保したあと，初期化することです。

- スタティックに登録する場合
  周期起動ハンドラをスタティックに登録する場合，コンフィギュレーション時に周期起動ハンドラ情報として指定します。
  RX850 Pro では，システム初期化処理時，情報ファイル（システム情報テーブル，システム情報ヘッダ・ファイル）に定義された情報を基に周期起動ハンドラの登録処理を行い，管理対象とします。

- ダイナミックに登録する場合
  周期起動ハンドラをダイナミックに登録する場合，処理プログラム（タスク，非タスク）内から def_cyc を発行します。
  RX850 Pro では，def_cyc 発行時，パラメータで指定された情報を基に周期起動ハンドラの登録処理を行い，管理対象とします。

8.6.2　周期起動ハンドラの活性状態

周期起動ハンドラの活性状態は，RX850 Pro が周期起動ハンドラを起動するかどうかを判定する際の基準の 1 つです。

活性状態は，周期起動ハンドラ登録時（コンフィギュレーション時，または def_cyc 発行時）に設定されますが，RX850 Pro では，この活性状態をユーザの処理プログラムからダイナミックに変更する機能を提供しています。

- act_cyc
  周期起動ハンドラの活性状態をパラメータで指定された状態に変更します。

  TCY_OFF：　周期起動ハンドラの活性状態を OFF 状態に変更する
  TCY_ON：　周期起動ハンドラの活性状態を ON 状態に変更する
  TCY_INI：　周期起動ハンドラの周期カウンタを初期化する

RX850 Pro では，周期起床ハンドラの活性状態が OFF 状態であっても，周期カウンタのカウント処理を行っています。このため，act_cyc を発行し，活性状態を OFF 状態から ON 状態に変更した場合，1 回目の起動要求が発行されるまでの間隔は，周期起動ハンドラ登録時（コンフィギュレーション時，または def_cyc 発行時）に指定した起動時間間隔よりも短くなる可能性があります。したがって，1 回目の起動要求を周期起動ハンドラ登録時に指定した時間間隔で発行するには，act_cyc を発行する際，周期起動ハンドラの起動再開（TCY_ON）のほかに，周期カウンタの初期化（TCY_INI）をあわせて指定してください。

図 8 － 2，図 8 － 3 に，処理プログラムから act_cyc を発行して周期起動ハンドラの活性状態を OFF から ON に変更した場合の動作の流れを示します。

なお，図 8 － 2，図 8 － 3 における ΔT は，周期起動ハンドラ登録時に指定したハンドラの起動時間間隔を表します。また，図 8 － 2 における Δt と ΔT の関係は，Δt ≦ ΔT であるものとします。

図 8 － 2　act_cyc（TCY_ON）発行時の動作の流れ

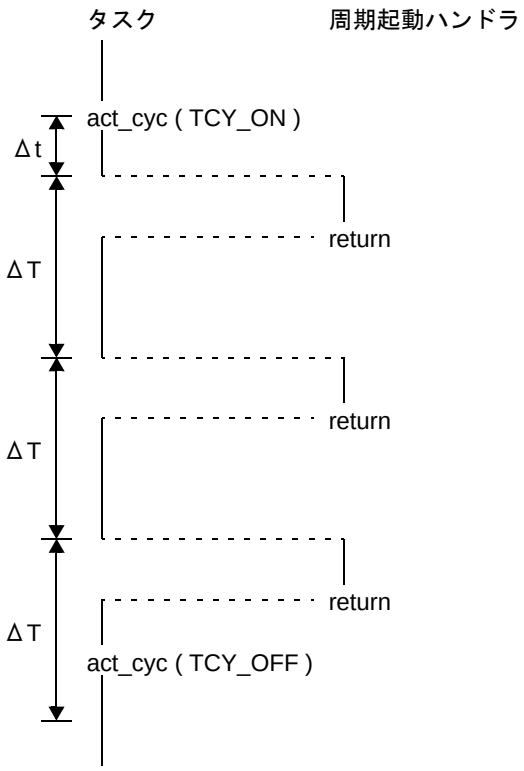
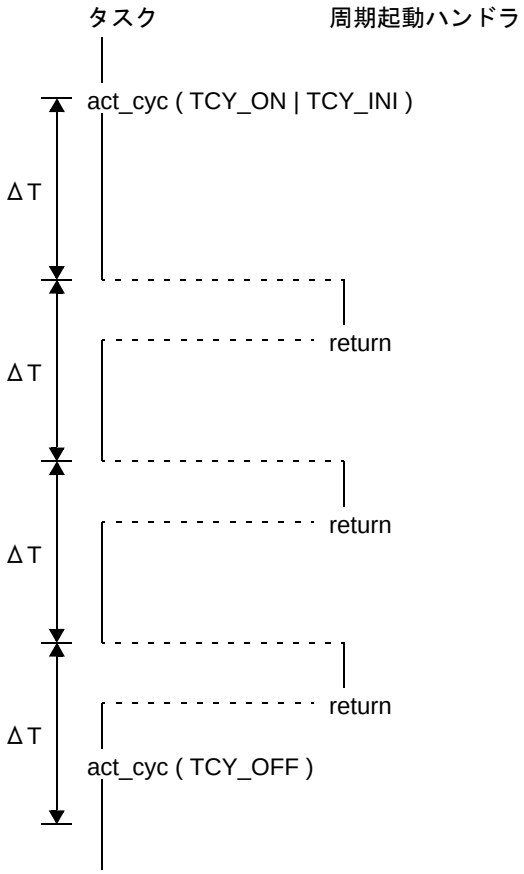


図 8 － 3　act_cyc（TCY_ON | TCY_INI）発行時の動作の流れ



8.6.3　周期起動ハンドラ内での処理

RX850 Pro では，クロック割り込みが発生してから周期起動ハンドラに制御を移す際，独自の割り込み前処理を行っています。また，周期起動ハンドラから制御を戻す際にも，独自の割り込み後処理を行っています。
したがって，周期起動ハンドラの処理を記述する際には，次の点に注意してください。

- レジスタの退避と復帰
  RX850 Pro では，周期起動ハンドラに制御を移すときと周期起動ハンドラから復帰するときに，C コンパイラの関数呼び出し規約に従って作業用レジスタの退避処理，復帰処理を行っています。したがって，周期起動ハンドラの開始部分で作業用レジスタの退避処理を，終了部分で作業用レジスタの復帰処理を記述する必要はありません。

- スタックの切り替え
  RX850 Pro では，周期起動ハンドラに制御を移すときと周期起動ハンドラから復帰するときに，スタックの切り替えを行います。したがって，周期起動ハンドラの開始部分で割り込みハンドラ用スタックへの切り替え処理を，終了部分で割り込み発生時のスタックへの切り替え処理を記述する必要はありません。
  ただし，コンフィギュレーション時に割り込みハンドラ用スタックを定義していない場合は，スタックの切り替え処理は行われず，割り込み発生時のスタックが使用されます。

- システム・コールの発行制限
  次に，周期起動ハンドラ内で発行可能なシステム・コールの一覧を示します。

  - タスク管理機能
    sta_tsk，chg_pri，rot_rdq，rel_wai，get_tid，ref_tsk，vget_tid

  - タスク付属同期機能
    sus_tsk，rsm_tsk，frsm_tsk，wup_tsk，can_wup

  - 同期通信機能
    sig_sem, preq_sem, ref_sem, vget_sid, set_flg, clr_flg, pol_flg, ref_flg, vget_fid, snd_msg, prcv_msg, ref_mbx, vget_mid

  - 割り込み処理管理機能
    def_int，ena_int，dis_int，chg_icr，ref_icr

  - メモリ・プール管理機能
    pget_blk，rel_blk，ref_mpl，vget_pid

  - 時間管理機能
    set_tim，get_tim，def_cyc，act_cyc，ref_cyc

  - システム管理機能
    get_ver，ref_sys，def_svc，viss_svc

- 周期起動ハンドラからの復帰処理
  周期起動ハンドラからの復帰処理は，ハンドラの終了部分で return 命令を発行することにより行われます。
  RX850 Pro では，周期起動ハンドラ内でタスクのスケジューリング処理が必要となるシステム・コール（chg_pri，sig_sem など）が発行された場合，待ちキューの操作などの処理が行われるだけであり，実際のスケジューリング処理は，周期起動ハンドラからの復帰処理（return 命令の発行）まで遅延され，一括して行われます。

8.6.4　周期起動ハンドラ情報の獲得

周期起動ハンドラ情報の獲得は，ref_cyc を発行することにより行われます。

- ref_cyc
  パラメータで指定された周期起動ハンドラの周期起動ハンドラ情報（拡張情報，残り時間など）を獲得します。
  次に，周期起動ハンドラ情報の詳細を示します。

  - 拡張情報
  - 次に周期起動ハンドラを起動するまでの残り時間
  - 現在の活性状態

8.6.5　周期起動ハンドラ中の割り込み

周期起動ハンドラは，割り込み禁止状態で開始されます。周期起動ハンドラ中では割り込みを許可したい場合，ハンドラの先頭で割り込みを許可してください。
RX850 Pro では，2 種類のニュークリアス共通部（rxtmcore.o，rxcore.o）を提供しており，使用するニュークリアス共通部によって，周期起動ハンドラ内で受け付け可能な割り込みが異なります。

- rxtmcore.o 使用時
  周期起動ハンドラは，クロック・ハンドラから呼び出されますが，クロック・ハンドラ実行中に割り込み終了処理を行っていませんので，クロック割り込みよりも優先順位の高い割り込みのみ受け付けることが可能です。なお，割り込みを許可してもクロック割り込みは保留されるため，周期起動ハンドラ内で時間のかかる処理を行う場合は，実際に経過した時間と RX850 Pro が管理している時間とにずれが生じる可能性があるため，注意が必要です。

- rxcore.o 使用時
  周期起動ハンドラは，クロック・ハンドラから呼び出されますが，クロック・ハンドラ実行中に割り込み終了処理を行っていますので，すべての割り込みレベルの割り込みを受け付けることができます。

また，周期起動ハンドラは間接起動割り込みハンドラとして実現されているので，実行時はハンドラ・スタック上で動作します。

第 9 章　スケジューラ

この章では，RX850 Pro が行うスケジューリング処理について説明します。

9.1　概　　要

RX850 Pro のスケジューラでは，ダイナミックに変化するタスクの状態を観察することによりタスクの実行順序を管理，決定し，最適なタスクにプロセッサの使用権を与えます。

9.2　駆動方式

RX850 Pro のスケジューラは，何らかの事象が発生した際に駆動される，事象駆動方式を採用しています。

なお，RX850 Pro における「何らかの事象」とは，タスクの状態遷移を引き起こす可能性のあるシステム・コールの発行，ハンドラからの復帰命令の発行，およびクロック割り込みの発生を意味します。これらの事象が発生するとスケジューラが駆動し，タスクのスケジューリング処理を行います。

次に，スケジューラを駆動するシステム・コールの一覧を示します。

- タスク管理機能
  sta_tsk，ext_tsk，exd_tsk，ena_dsp，chg_pri，rot_rdq，rel_wai

- タスク付属同期機能
  rsm_tsk，frsm_tsk，slp_tsk，tslp_tsk，wup_tsk

- 同期通信機能
  del_sem，sig_sem，wai_sem，twai_sem，del_flg，set_flg，wai_flg，twai_flg，del_mbx，snd_msg，rcv_msg，trcv_msg

- 割り込み処理管理機能
  unl_cpu

- メモリ・プール管理機能
  del_mpl，get_blk，tget_blk，rel_blk

- 時間管理機能
  dly_tsk

9.3　スケジューリング方式

RX850 Pro のスケジューリング方式は，優先度方式と FCFS（First Come First Service）方式を採用しています。

このため，スケジューラは，実行可能な状態（run 状態，ready 状態）にあるタスクの優先度を参照し，その中から最適なタスクを選び出し，プロセッサの使用権を与えます。

9.3.1　優先度方式

各タスクには，処理を実行するうえでの優先順位を決定する優先度が付けられています。

したがって，スケジューラは，実行可能な状態（run 状態，ready 状態）にあるタスクの優先度を参照し，その中から最も高い優先度（最高優先度）を持つタスクを選び出し，プロセッサの使用権を与えます。

備考　　RX850 Pro におけるタスクの優先度は，その値が小さいほど高い優先度であることを示します。

9.3.2　FCFS 方式

RX850 Pro では，複数のタスクに同一の優先度を割り付けることが可能です。このため，優先度方式におけるタスクを選び出す基準である，「最も高い優先度（最高優先度）を持つタスク」が複数存在する場合があります。

そこで，スケジューラは，最高優先度を持つタスクが複数存在する場合には，先に実行可能状態になったタスク（ready 状態になってから最も時間が経過しているタスク）を選び出し，プロセッサの使用権を与えます。

9.4　ラウンドロビン方式の実現

RX850 Pro では，優先度方式，または FCFS 方式のスケジューリングを行っていますが，これらの方式では，タスクが複数存在した場合，最初にプロセッサの使用権を与えられたタスクがほかの状態へ遷移するか，またはプロセッサの使用権を放棄しないかぎり，他タスクが「同一の優先度でありながら処理を実行することができない」といった事態が生じてきます。

そこで，RX850 Pro では，このような事態を回避するスケジューリング方式（ラウンドロビン方式）を実現するために，rot_rdq などのようなシステム・コールを提供しています。

次に，ラウンドロビン方式の実現例を示します。

【 条件 】

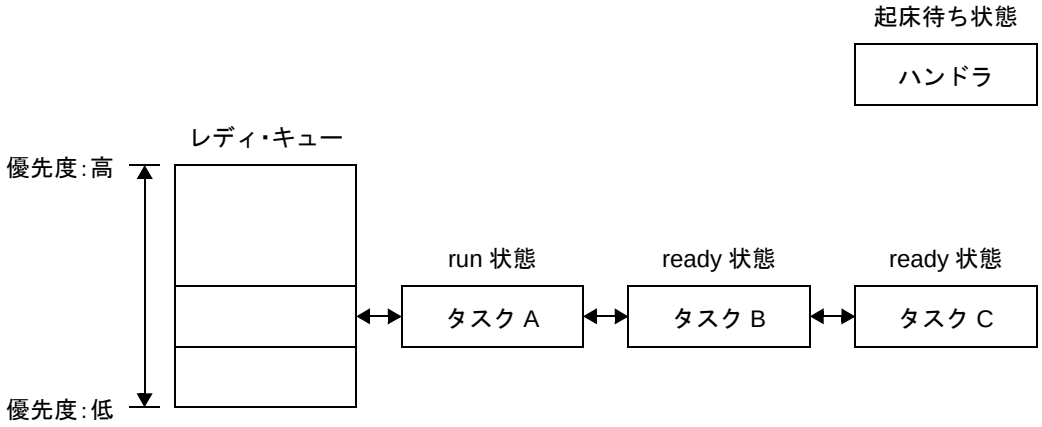
- タスクの優先度
  タスク A = タスク B = タスク C

- タスクの状態
  タスク A :　run 状態
  タスク B :　ready 状態
  タスク C :　ready 状態

- 周期起動ハンドラの属性
  活性状態 :　ON 状態
  起動時間間隔 :　ΔT（単位 : 基本クロック周期）
  処理内容 :　レディ・キューの回転（rot_rdq の発行）

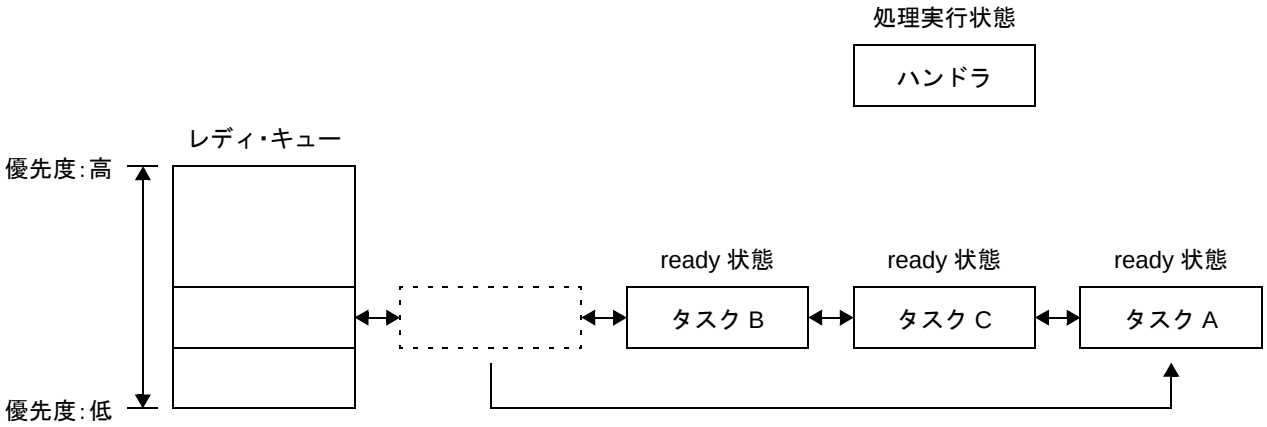
（1）現在，タスク A が処理を実行しています。
　　他タスク（タスク B，タスク C）は，タスク A と同一の優先度を持ちますが，タスク A がほかの状態へ遷移するか，またはプロセッサの使用権を放棄しないかぎり，処理を実行できません。
　　図 9 － 1 に，このときのレディ・キューの状態を示します。

図 9 － 1　レディ・キューの状態（その 1）



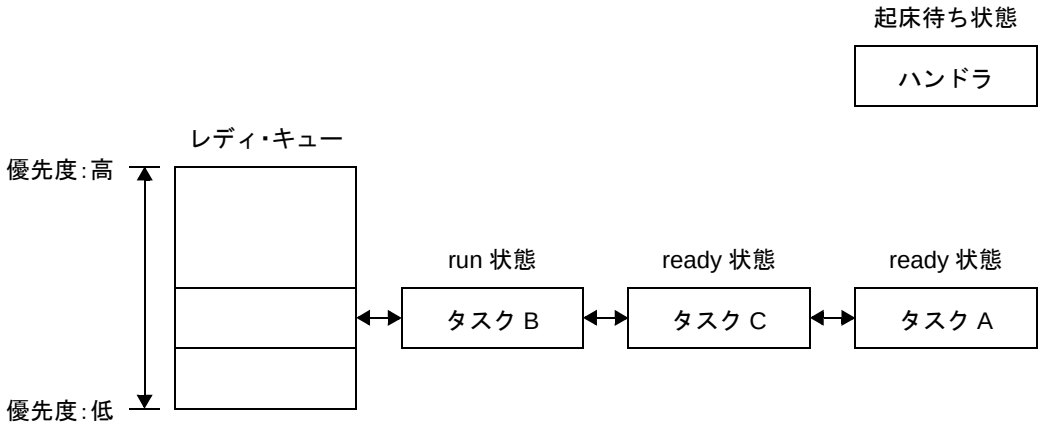
（2）時間の経過により周期起動ハンドラが起動し，rot_rdq を発行します。
これにより，タスク A は優先度に応じたレディ・キューの最後尾にキューイングされるとともに，run 状態から ready 状態へと遷移します。
図 9 － 2 に，このときのレディ・キューの状態を示します。

図 9 － 2　レディ・キューの状態（その 2）



（3）タスク A が run 状態から ready 状態へと遷移し，タスク B が ready 状態から run 状態へと遷移します。
図 9 － 3 に，このときのレディ・キューの状態を示します。

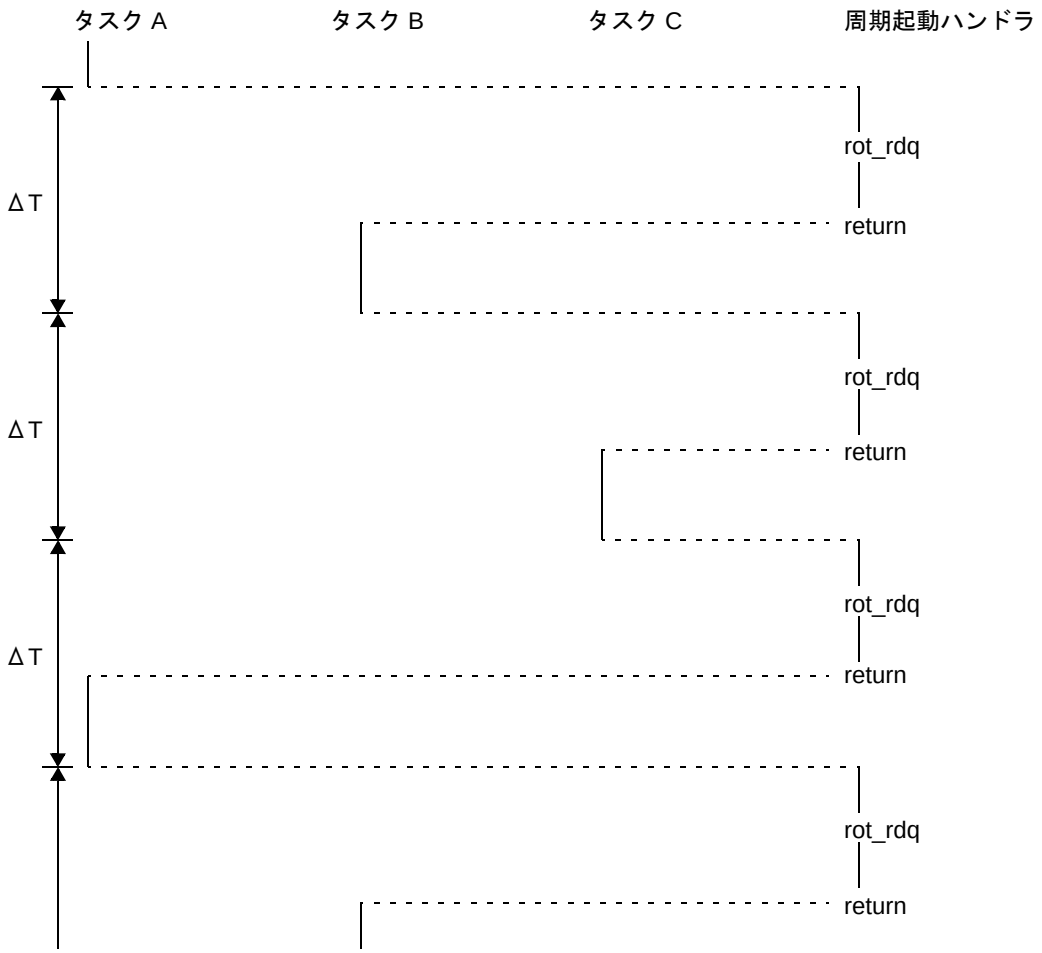
図 9 － 3　レディ・キューの状態（その 3）



（4）このようにして，一定周期で起動される周期起動ハンドラから rot_rdq を発行することにより，一定の時間（ΔT）が経過したときに必ずタスクが切り替わるスケジューリング方式（ラウンドロビン方式）を実現できます。

図９－４に，ラウンドロビン方式による動作の流れを示します。

図９－４　ラウンドロビン方式による動作の流れ



9.5　スケジューリングのロック機能

RX850 Pro では，ユーザの処理プログラム（タスク）からスケジューラの駆動を操作し，ディスパッチ処理（タスクのスケジューリング処理）を禁止／再開する機能が提供されています。
なお，これらの機能は，次に示すシステム・コールをタスク内から発行することにより実現されます。

- dis_dsp
  ディスパッチ処理（タスクのスケジューリング処理）を禁止します。
  これにより，本システム・コールの発行から ena_dsp が発行されるまでの間は，他タスクに制御が移ることはありません。

- ena_dsp
  ディスパッチ処理（タスクのスケジューリング処理）を再開します。
  なお，RX850 Pro では，dis_dsp の発行から本システム・コールの発行までの間に，タスクのスケジューリング処理が必要なシステム・コール（chg_pri，sig_sem など）が発行された場合，待ちキューの操作などの処理が行われるだけであり，実際のスケジューリング処理は，本システム・コールが発行されるまで遅延され，一括して行われます。

- loc_cpu
  マスカブル割り込みの受け付けを禁止したあと，ディスパッチ処理（タスクのスケジューリング処理）を禁止します。
  これにより，本システム・コールが発行されてから unl_cpu が発行されるまでの間は，ほかのタスクやハンドラに制御が移ることはありません。

- unl_cpu
  マスカブル割り込みの受け付けを許可したあと，ディスパッチ処理（タスクのスケジューリング処理）を再開します。
  なお，RX850 Pro では，loc_cpu の発行から本システム・コールの発行までの間にマスカブル割り込みが発生した場合，該当する割り込み処理（割り込みハンドラ）への移行は本システム・コールが発行されるまで遅延されます。また，loc_cpu の発行から本システム・コールの発行までの間にタスクのスケジューリング処理が必要なシステム・コール（chg_pri，sig_sem など）が発行された場合，待ちキューの操作などの処理が行われるだけであり，実際のスケジューリング処理は本システム・コールが発行されるまで遅延され，一括して行われます。

図 9 － 5 にスケジューリング処理が遅延されない場合（通常時）の動作の流れを，図 9 － 6，図 9 － 7 に，dis_dsp，loc_cpu を発行した場合の動作の流れを示します。

図 9 － 5　スケジューリング処理が遅延されない場合（通常時）の動作の流れ

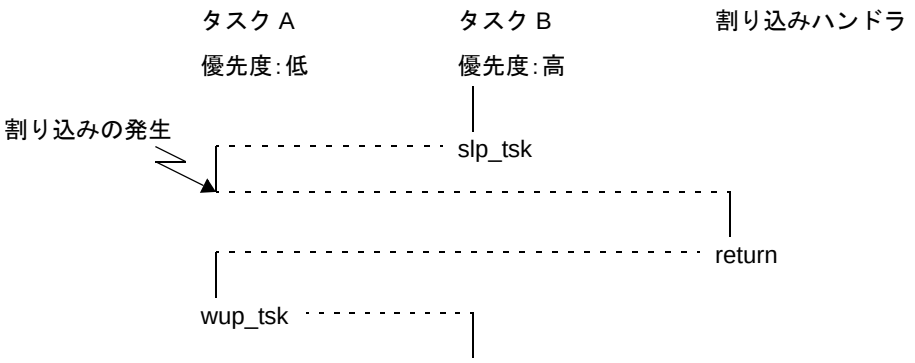


図 9 － 6　dis_dsp を発行した場合の動作の流れ

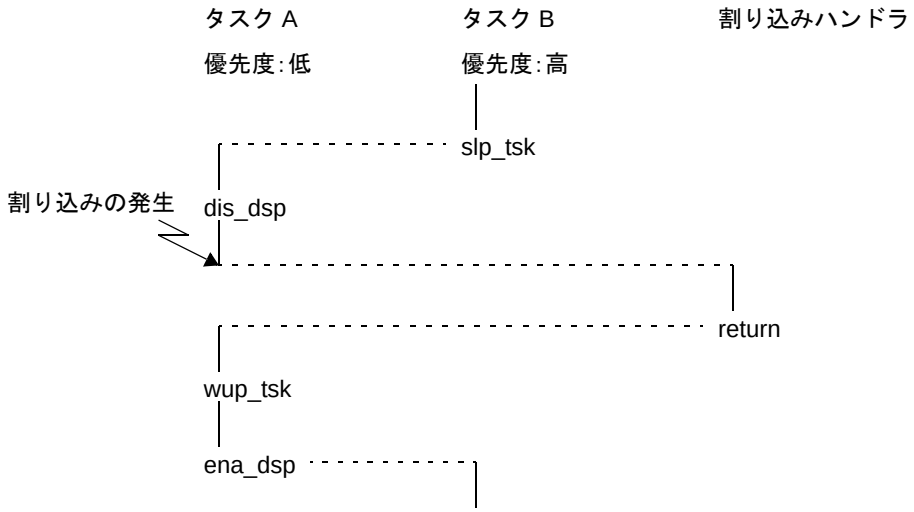
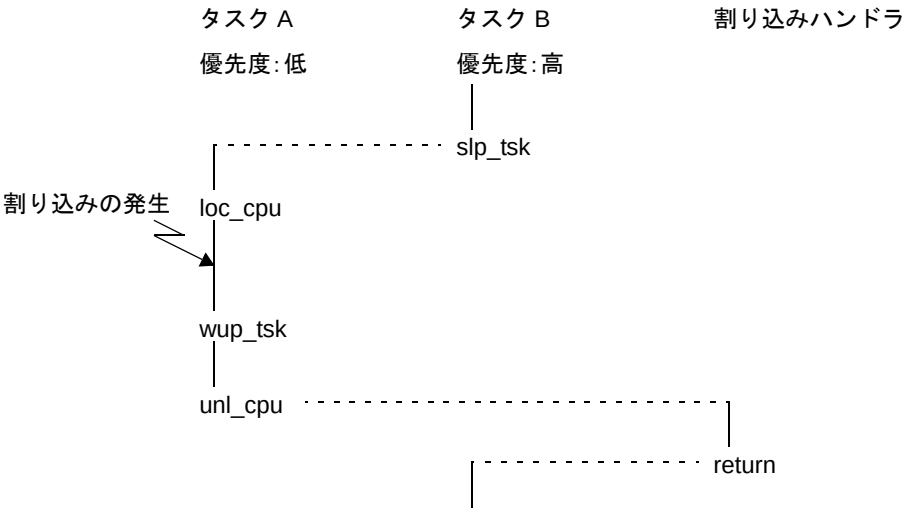


図 9 － 7　loc_cpu を発行した場合の動作の流れ



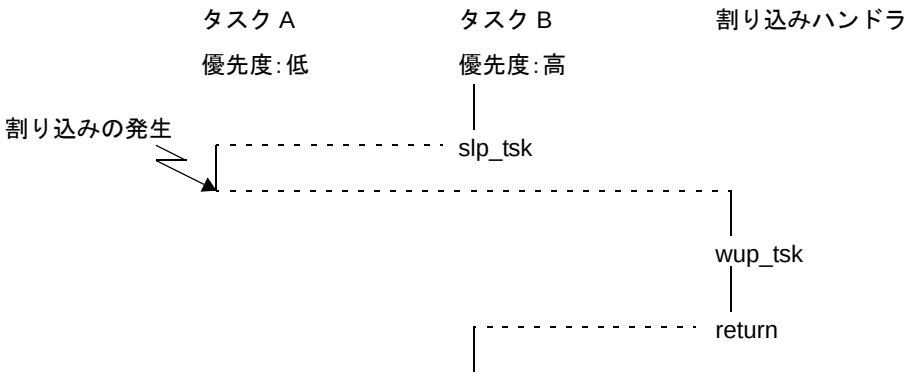
9.6　ハンドラ内でのスケジューリング

RX850 Pro では，各種ハンドラ（割り込みハンドラ，周期起動ハンドラ）を高速に終了させる目的で，ハンドラ内の処理が終了するまでの間，スケジューラの駆動を遅延します。

したがって，ハンドラ内でタスクのスケジューリング処理が必要になるシステム・コール（chg_pri，sig_sem など）が発行された場合，待ちキューの操作などの処理が行われるだけであり，実際のスケジューリング処理はハンドラからの復帰処理（return 命令などの発行）まで遅延され，一括して行われます。

図 9 － 8 に，ハンドラ内でスケジューリング処理が必要なシステム･コールが発行された場合の動作の流れを示します。

図 9 － 8　wup_tsk を発行した場合の動作の流れ



9.7　アイドル・ハンドラ

アイドル・ハンドラは，すべてのタスク（ユーザの定義したタスク）が run 状態，および ready 状態でなくなったとき，すなわち RX850 Pro のスケジューリング対象となるタスクがシステム内に 1 つも存在しなくなったときに処理を実行します。

アイドル・ハンドラの処理とは，CPU を HALT 状態にすることです。したがって，タスクがシステム内に 1 つも存在しなくなると，RX850 Pro は CPU を HALT 状態にします。

ただし，このアイドル・ハンドラは，CPU を IDLE 状態，STOP 状態にすることはできません。IDLE 状態，STOP 状態にしたい場合，またはアイドル処理を自分で記述したい場合は，一番優先度の低いタスクを作成し，アイドル・タスクとしてください。これにより，アイドル・ハンドラと同様の処理を実現できます。ただし，HALT や IDLE，STOP 状態は割り込みによって解除されるため，アイドル・タスク内では割り込みを禁止状態のままにしないでください。

第 10 章　システム初期化処理

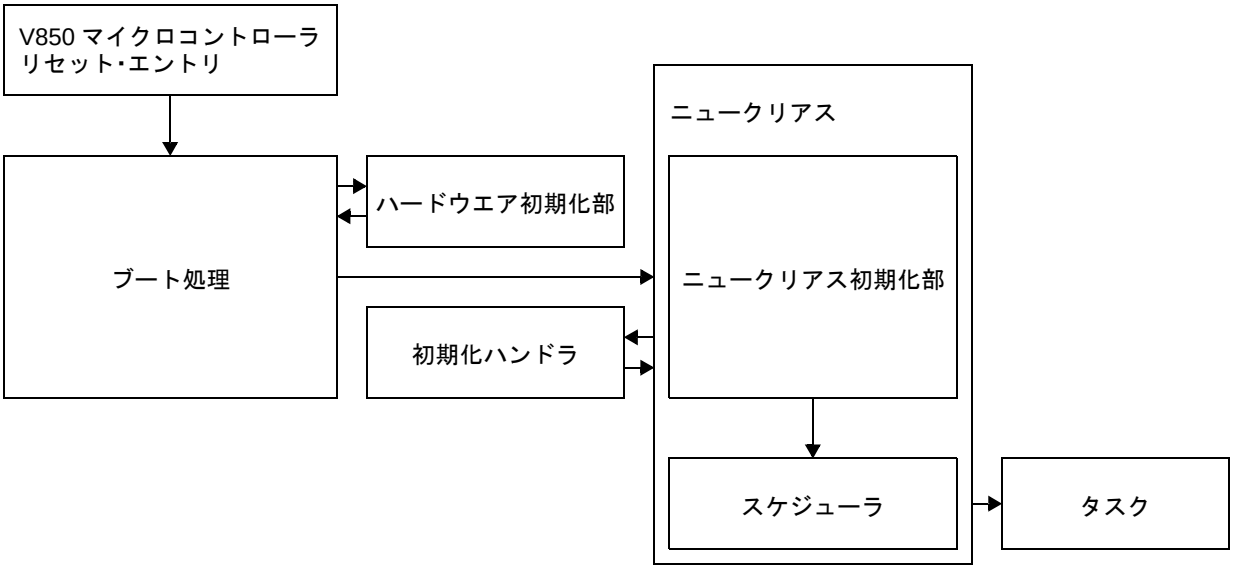
この章では，RX850 Pro が行うシステム初期化処理について説明します。

10.1　概　　要

システム初期化処理は，RX850 Pro が動作するうえで必要となるハードウエアの初期化処理とソフトウエアの初期化処理から構成されています。つまり，システムが起動したとき，RX850 Pro で最初に行われる処理が，システム初期化処理となります。

図 10 － 1 に，システム初期化処理の流れを示します。

図 10 － 1　システム初期化処理の流れ



10.2　ブート処理

ブート処理は，V850 マイクロコントローラのリセット・エントリ（ハンドラ・アドレス：0x0）に割り付けられた機能であり，システム初期化処理の中でも最初に実行されます。サンプルでは，boot.s というファイルで行われています（関数名：__boot）。

ブート処理内では，次のことをする必要があります。

- ブート処理内で使う sp（スタック・ポインタ）の設定
- tp（テキスト・ポインタ），gp（グローバル・ポインタ）の設定
- jarl 命令を発行して，ハードウエア初期化部に制御を移行
- _sit シンボルの r10 アドレスに設定
- __rx_start シンボルを lp レジスタに設定
- jmp 命令を発行して，ニュークリアス初期化部に制御を移行

サンプルのブート処理は，その処理内容を，ユーザのニーズにあわせて書き換えます。

10.3　ハードウエア初期化部

ハードウエア初期化部は，ブート処理から呼び出される関数であり，実行環境（ターゲット・システム）上のハードウエアを初期化するために用意されたものです。サンプルでは，init.c というファイルで行われています（関数名：reset）。

ハードウエア初期化部では，次のような処理を行います。

- 内部ユニットの初期化
  - 割り込みコントローラの初期化
  - クロック・コントローラの初期化
- 周辺コントローラの初期化
- ブート処理に制御を戻す

ハードウエア初期部は，実行環境のハードウエア構成に依存します。

この部分を切り出すことにより，さまざまなターゲット・システムへの移植性を向上させるとともに，カスタマイズ化を容易なものにしています。ユーザの実行環境にあわせて書き換えてください。

10.4　ニュークリアス初期化部

ニュークリアス初期化部は，ブート処理終了後に呼び出されます。情報ファイル（システム情報テーブル，システム情報ヘッダ・ファイル）に記述された情報（タスク情報，セマフォ情報など）をもとに，各種管理オブジェクトを生成／初期化します。この処理の終了後に，RX850 Pro が起動されます。なお，この処理部はニュークリアス・ライブラリに含まれています。

ニュークリアス初期化部では，次のような処理を行います。

- 管理オブジェクトの生成／初期化
  - タスクの生成
  - セマフォの生成／初期化
  - イベントフラグの生成／初期化
  - メモリ・プールの生成／初期化
  - 間接起動割り込みハンドラの登録
  - 周期起動ハンドラの登録
  - 拡張 SVC ハンドラの登録
- 初期タスクの起動
- システム・タスク（アイドル・タスク）の起動
- 初期化ハンドラの呼び出し
- スケジューラに制御を移す

10.5　初期化ハンドラ

初期化ハンドラは，ニュークリアス初期化部から呼び出される関数です。RX850 Pro 起動前にしておきたい処理がある場合などに利用できます。サンプルでは，varfunc.c というファイルで行われています（関数名：varfunc）。

初期化ハンドラでは，次のような処理を行います。

- 初期化データのコピー
- ニュークリアス初期化部に制御を戻す

備考 1　RX850 Pro では，ニュークリアス初期化部から本処理部に制御を移す際，スタックをコンフィギュレーション時にシステム情報で指定されたシステム・スタックへの切り替え処理を実行しています。

備考 2　RX850 Pro では，ニュークリアス初期化部から本処理部に制御を移す際，tp（テキスト・ポインタ），gp（グローバル・ポインタ）をコンフィギュレーション時に初期化ハンドラ情報で定義された値への切り替え処理を実行しています。

備考 3　RX850 Pro では，ep（エレメント・ポインタ）に対する操作を行いません。したがって，本処理部を実行する際の ep は，ブート処理で設定された値となります。

備考 4　本処理部は，RX850 Pro の初期化処理がすべて完了する以前に呼び出されます。したがって，本処理部を割り込み許可状態とした場合，および，本処理部でシステム・コールを発行した場合，動作の保証はありません。

10.6　割り込みエントリ

割り込みエントリは，割り込みが発生した際に実行される命令を記述したもので，V850 マイクロコントローラが持つ割り込みハンドラ・アドレスに割りつけられます。ユーザが使用するすべての割り込みに対し，割り込みエントリを定義する必要があります。これらはアセンブリ言語で記述する必要があり，サンプルでは，entry.s というファイルで行われています。

第 11 章　インタフェース・ライブラリ

この章では，RX850 Pro が提供するインタフェース・ライブラリについて説明します。

11.1　概　　要

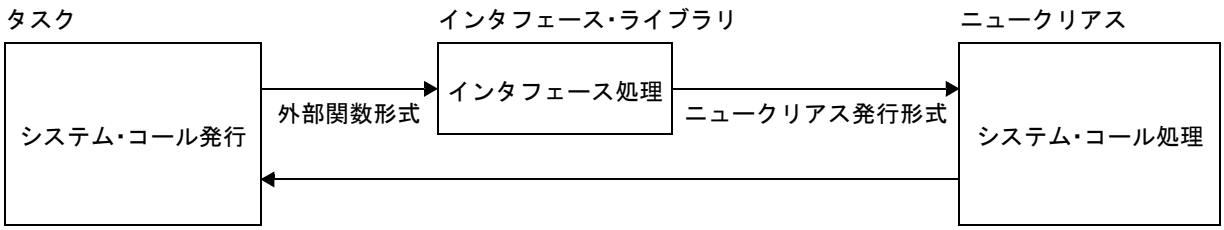
RX850 Pro では，ユーザの処理プログラムと RX850 Pro のニュークリアスの中間に位置するインタフェース・ライブラリを提供しています。インタフェース・ライブラリは，ニュークリアスが処理を行ううえで必要な各種情報の設定等を行ったあとに制御を移す機能を持っています。

処理プログラム（タスク，非タスク）を C 言語で記述した場合，システム・コールの発行形式や拡張 SVC ハンドラの呼び出し形式は外部関数形式となります。しかし，ニュークリアスが理解可能な発行形式（ニュークリアス発行形式）と外部関数形式には相違があります。

そこで，システム・コールの発行形式や拡張 SVC ハンドラの呼び出し形式を外部関数形式からニュークリアス発行形式に変換する手続き（インタフェース）が必要になります。このような，処理プログラムとニュークリアスの仲介役を行うインタフェースは各システム・コール毎にあり，これを集めたものがインタフェース・ライブラリです。

図 11 － 1 に，インタフェース・ライブラリの位置づけを示します。

図 11 － 1　インタフェース・ライブラリの位置づけ



11.2　インタフェース・ライブラリ内での処理

インタフェース・ライブラリでは，次のような処理を行っています。

- ニュークリアスが管理しているテーブルに必要な情報を設定する
- 必要になるデータをレジスタに設定する
- システム・コールのエラー値を設定（ただし，ニュークリアス内で設定されるエラーは除く）したあと，処理プログラムへ処理を戻す

インタフェース・ライブラリを用意することにより，ニュークリアスとユーザの処理プログラムの分離が容易になります。インタフェース・ライブラリはユーザ・アプリケーション側にリンクされることになるので，ニュークリアス本体を ROM 化したあとでユーザの処理プログラムを変更する必要が生じた場合でも，ニュークリアス本体が格納されている ROM を変更する必要がなくなります。また，ロード・モジュールを分割して作成するといったことも可能になります。

「11.5　システム・コール用インタフェース・ライブラリ」では，システム・コール用インタフェース・ライブラリの，「11.6　拡張 SVC ハンドラ用インタフェース・ライブラリ」では，拡張 SVC ハンドラのインタフェース・ライブラリの具体的な書式を掲載していますので，詳細はそちらを参照してください。

11.3　インタフェース・ライブラリの種類

RX850 Pro が提供するインタフェース・ライブラリには，システム・コールのパラメータをチェックする機能を持つものと持たないものに大別されます。システムにどちらを組み込むかは，リンク時に指定します。

パラメータ・チェック機能のあるライブラリを使用すると，システム・コール発行時のパラメータの指定が不正だった場合，必ず戻り値が返されます。一方，パラメータ・チェック機能のないライブラリを使用すると，システム・コール発行時のパラメータの指定が不正だった場合，戻り値が返されない場合があります。

これらのライブラリは，用途に応じて使い分けることができます。たとえば，デバッグ時はパラメータ・チェック機能のあるものを使用し，実際の組み込み時はパラメータ･チェック機能のないものを使用することで，プログラムのパフォーマンス向上や容量の節減等を実現します。

備考 1　パラメータ・チェック機能のないライブラリで戻り値が返されるエラーは，「第 12 章　システム・コール」の各システム・コールの“戻り値”欄で，「*」を付けて示しています。

備考 2　パラメータ・チェック機能のないライブラリを使用したとき，戻り値が返されないエラーが発生した場合は，アプリケーション・システムの動作は保証されません。

11.4　インタフェース・ライブラリの変更

ユーザが必要に応じてインタフェース・ライブラリを変更したい場合は，インタフェース・ライブラリを書き換える必要があります。

変更したインタフェース・ライブラリは，アセンブルし，あらためてライブラリ化する必要があります。

11.5　システム・コール用インタフェース・ライブラリ

システム・コール用インタフェース・ライブラリでは，次のような処理を行います。
ただし，システム・コールのパラメータの引き渡し方法については，使用する C コンパイラに準じます。

- システム・コールの機能コードを r10 レジスタに設定
- システム・コールからの戻り番地を lp レジスタに設定
- システム・コールのパラメータをチェック
- システム・コールの入り口のアドレス（hp レジスタ + 0x100 番地）を獲得
- システム・コールの入り口にジャンプ

また，システム・コールのパラメータをチェックした際にエラーを検出した場合，次の処理を行います。

- 検出したエラーに対応したエラー・コードを r10 レジスタに設定

図 11 － 2 に，システム・コール用インタフェース・ライブラリの記述例を示します。

図 11 － 2　システム・コール用インタフェース・ライブラリの記述例

```
    .text
    .globl  _syscall_name
    .align  2
_syscall_name :
    -- 機能コード設定
    mov     func_code, r10

    -- パラメータをチェック
    ........................
    ........................

    jz      _syscall_err

    -- システム・コールの入り口のアドレスを獲得
    ld.w    0x100 [hp], r12

    -- システム・コールの入り口にジャンプ
    jmp     [r12]
```

11.6　拡張 SVC ハンドラ用インタフェース・ライブラリ

拡張 SVC ハンドラ用インタフェース・ライブラリでは，次のような処理を行います。
ただし，拡張 SVC ハンドラのパラメータの引き渡し方法については，使用する C コンパイラに準じます。

- 拡張 SVC ハンドラの機能コードを r10 レジスタに設定
- 拡張 SVC ハンドラからの戻り番地を lp レジスタに設定
- 拡張 SVC ハンドラのパラメータをチェック
- 拡張 SVC ハンドラのパラメータ領域のサイズを r11 レジスタに設定

  例　　int 型のパラメータが 4 個の場合，0x10 を r11 レジスタに設定

- 拡張 SVC ハンドラの入り口のアドレスを獲得（hp レジスタ + 0x108 番地）を獲得
- 拡張 SVC ハンドラの入り口にジャンプ

また，拡張 SVC ハンドラのパラメータをチェックした際にエラーを検出した場合，次のような処理を行います。

- 検出したエラーに対応したエラー・コードを r10 レジスタに設定

図 11 － 3 に，拡張 SVC ハンドラ用インタフェース・ライブラリの記述例を示します。

図 11 － 3　拡張 SVC ハンドラ用インタフェース・ライブラリの記述例

```
    .text
    .globl  _svchdr_name
    .align  2
_svchdr_name :
    -- 機能コード設定
    mov     func_code, r10

    -- パラメータをチェック
    ........................
    ........................

    jz      _svchdr_err

    -- パラメータ領域のサイズを設定
    mov     prm_siz, r11

    -- 拡張 SVC ハンドラの入り口のアドレスを獲得
    ld.w    0x108[hp], r12

    -- 拡張 SVC ハンドラの入り口にジャンプ
    jmp     [r12]
```

第 12 章　システム・コール

この章では，RX850 Pro が提供するシステム・コールについて説明します。

12.1　概　　要

システム・コールとは，ユーザの処理プログラム（タスク，非タスク）から RX850 Pro のサービス・ルーチンを呼び出すための手続き／機能です。システム・コールを利用すると，RX850 Pro が直接管理している資源（カウンタ，待ちキューなど）を間接的に操作できます。

RX850 Pro では，μITRON3.0 仕様で規定されているシステム・コールのほかに，RX850 Pro オリジナルのシステム・コールも提供し，アプリケーション・システムの汎用性を高めています。

システム・コールは，その機能により次に示す 7 グループに分類できます。

- タスク管理機能（14 種類）
  タスクの状態操作を行うシステム・コールのグループです。
  また，このグループには，タスクを生成，起動，終了，または削除する機能，ディスパッチ処理を禁止，または再開する機能，タスクの優先度を変更する機能，タスクのレディ・キューを回転する機能，タスクの wait 状態を強制的に解除する機能，タスクの状態を参照する機能のシステム・コールも含まれます。

  cre_tsk，del_tsk，sta_tsk，ext_tsk，exd_tsk，ter_tsk，dis_dsp，ena_dsp，chg_pri，rot_rdq，rel_wai，get_tid，ref_tsk，vget_tid

- タスク付属同期機能（7 種類）
  タスクに従属した同期操作を行うシステム・コールのグループです。
  また，このグループには，タスクを suspend 状態に移行する，または suspend 状態のタスクを再開する機能，タスクを起床待ち状態に移行する，または起床待ち状態のタスクを起床させる機能，タスクの起床要求を無効にする機能などのシステム・コールも含まれます。

  sus_tsk，rsm_tsk，frsm_tsk，slp_tsk，tslp_tsk，wup_tsk，can_wup

- 同期通信機能（25 種類）
  タスク間の同期（排他制御，待ち合わせ）と通信を行うシステム・コールのグループです。
  また，このグループには，セマフォを操作する機能，イベントフラグを操作する機能，メールボックスを操作する機能のシステム・コールも含まれます。

  cre_sem，del_sem，sig_sem，wai_sem，preq_sem，twai_sem，ref_sem，vget_sid，cre_flg，del_flg，set_flg，clr_flg，wai_flg，pol_flg，twai_flg，ref_flg，vget_fid，cre_mbx，del_mbx，snd_msg，rcv_msg，prcv_msg，trcv_msg，ref_mbx，vget_mid

- 割り込み処理管理機能（7 種類）
  マスカブル割り込みに依存した処理を行うシステム・コールのグループです。
  また，このグループには，間接起動割り込みハンドラを登録，または登録解除する機能，直接起動割り込みハンドラから復帰する機能，割り込み許可レベルを変更，または参照する機能のシステム・コールも含まれます。

  def_int，ena_int，dis_int，loc_cpu，unl_cpu，chg_icr，ref_icr

- メモリ・プール管理機能（8 種類）
  メモリの割り当てを行うシステム・コールのグループです。
  また，このグループには，メモリ・プールを生成，または削除する機能，メモリ・ブロックを獲得，または返却する機能，メモリ・プールの状態を参照する機能のシステム・コールも含まれます。

  cre_mpl，del_mpl，get_blk，pget_blk，tget_blk，rel_blk，ref_mpl，vget_pid

- 時間管理機能（6 種類）
  時間に依存した処理を行うシステム・コールのグループです。
  また，このグループには，システム・クロックの時刻を設定，または参照する機能，タスクを時間経過待ち状態に遷移させる機能，周期起動ハンドラを登録，または登録解除する機能，周期起動ハンドラの活性状態を制御，または参照する機能のシステム・コールも含まれます。

  set_tim，get_tim，dly_tsk，def_cyc，act_cyc，ref_cyc

- システム管理機能（4 種類）
  システムに依存した処理を行うシステム・コールのグループです。
  また，このグループには，バージョン情報を獲得する機能，システムの状態を参照する機能，拡張 SVC ハンドラを登録，または登録解除する機能，拡張 SVC ハンドラを呼び出す機能のシステム・コールも含まれます。

  get_ver，ref_sys，def_svc，viss_svc

12.2　システム・コールの呼び出し

C 言語で記述された処理プログラム（タスク，非タスク）からシステム・コールを発行する場合，C 言語の関数として呼び出し，そのパラメータを引き数として渡します。
また，アセンブリ言語で記述された処理プログラムからシステム・コールを発行する場合は，使用する C コンパイラの関数呼び出し規約に従ってパラメータと戻り番地の設定を行ったあと，jarl 命令により呼び出します。

備考　RX850 Pro では，システム・コールのプロトタイプ宣言を stdrx85p.h ファイルで行っています。したがって，処理プログラムからシステム・コールを発行する際には，次のようにヘッダ・ファイルのインクルード処理を記述してください。

```
#include    <stdrx85p.h>
```

12.3　システム・コールの機能コード

RX850 Pro が提供するシステム・コールには，μITRON3.0 仕様に準拠した機能コードが割り当てられています。
表 12 － 1 に，システム・コールに割り当てられている機能コード一覧を示します。
なお，RX850 Pro では，1 以上の値については，ユーザが記述した拡張 SVC ハンドラを登録する際に指定する拡張機能コードとしています。

表 12 － 1　システム・コールの機能コード一覧

機能コード	分類
-256 ～ -225	RX850 Pro オリジナルのシステム・コール
-224 ～ -5	μITRON3.0 仕様に準拠したシステム・コール
-4 ～ 0	システム予約
1 ～	拡張 SVC ハンドラ

12.4　パラメータのデータ・タイプ

RX850 Pro が提供するシステム・コールのパラメータは，μITRON3.0 仕様に準拠したデータ・タイプを基本に定義されています。

表 12 － 2 に，システム・コールを発行する際に指定する各種パラメータのデータ・タイプ一覧を示します。

表 12 － 2　パラメータのデータ・タイプ一覧

マクロ	データ・タイプ	意味
B	signed char	符号付き 8 ビット整数
H	signed short	符号付き 16 ビット整数
INT	signed int	符号付き 32 ビット整数
W	signed long	符号付き 32 ビット整数
UB	unsigned char	符号なし 8 ビット整数
UH	unsigned short	符号なし 16 ビット整数
UINT	unsigned int	符号なし 32 ビット整数
UW	unsigned long	符号なし 32 ビット整数
VB	signed char	データ・タイプが一定しない値（8 ビット）
VH	signed short	データ・タイプが一定しない値（16 ビット）
VW	signed long	データ・タイプが一定しない値（32 ビット）
*VP	void	データ・タイプが一定しない値（ポインタ）
(*FP) ( )	void	処理プログラムの起動アドレス
BOOL	signed short	ブール値
FN	signed short	機能コード
ID	signed short	オブジェクト ID 番号
BOOL_ID	signed short	待ちタスクの有無
HNO	signed short	周期起動ハンドラの指定番号
ATR	unsigned short	オブジェクトの属性
ER	signed long	エラー・コード
PRI	signed short	タスクの優先度
TMO	signed long	待ち時間
CYCTIME	signed long	周期起動時間間隔（残り時間）
DLYTIME	signed long	遅延時間

12.5　パラメータ値の範囲

RX850 Pro が提供するシステム・コールのパラメータには，指定可能な値に範囲のあるもの，特定の値をシステムで予約しているものなどがあります。

表 12 － 3 に，システム・コールを発行する際に指定する各種パラメータの値域一覧を示します。

表 12 － 3　パラメータの値域一覧

パラメータの種類	値
オブジェクトの ID 番号	0x0 ～ max_cnt 注 1
オブジェクトのキー ID 番号	-0x8000 ～ 0x7FFF 注 2
割り込みハンドラの割り込みレベル	0x0 ～ 0xF
周期起動ハンドラの指定番号	0x1 ～ max_cnt
拡張 SVC ハンドラの拡張機能コード	0x1 ～ max_cnt
オブジェクトの優先度	0x1 ～ max_cnt
セマフォの最大資源数	0x1 ～ max_cnt
マスカブル割り込みの割り込み許可レベル	0x0 ～ 0xF
システム・クロックの時刻	0x0 ～ 0x7FFF FFFF FFFF
待ち時間	-0x1 ～ 0x7FFF FFFF
遅延時間	0x0 ～ 0x7FFF FFFF
周期起動ハンドラの起動時間間隔	0x1 ～ 0x7FFF FFFF
タスクのスタック・サイズ	0x0 ～ 0x7FFF FFFF
メモリ・プール・サイズ	0x1 ～ 0x7FFF FFFF
メモリ・ブロック・サイズ	0x1 ～ 0x7FFF FFFF
メッセージの優先度	0x1 ～ 0x7FFF

注 1　　max_cnt：システム・コンフィギュレーション時にシステム最大値情報で指定した最大オブジェクト数

注 2　　オブジェクトのキー ID 番号に“0x0”を指定することはできません。

12.6　システム・コールからの戻り値

RX850 Pro が提供するシステム・コールの戻り値は，μITRON3.0 仕様に準拠した戻り値を基本に定義されています。表 12 － 4 に，システム・コールからの戻り値一覧を示します。

表 12 － 4　システム・コールからの戻り値一覧

マクロ	数値	意味
E_OK	0	正常終了
E_NOMEM	-10	オブジェクト用の領域が確保できない
E_NOSPT	-17	CF 定義されていないシステム・コール，または登録されていない拡張 SVC ハンドラを呼び出した
E_RSATR	-24	オブジェクト属性の指定が不正である
E_PAR	-33	パラメータの指定が不正である
E_ID	-35	ID 番号の指定が不正である
E_NOEXS	-52	対象オブジェクトが存在していない
E_OBJ	-63	指定したオブジェクトの状態が不適当である
E_OACV	-66	アクセス権のない ID 番号を指定した
E_CTX	-69	システム・コールを発行する状態が不適当である
E_QOVR	-73	カウント値が 127 を越えた
E_DLT	-81	対象オブジェクトが削除された
E_TMOUT	-85	タイムアウト
E_RLWAI	-86	rel_wai により，wait 状態を強制的に解除された

12.7　システム・コールの拡張

RX850 Pro では，システム・コールを拡張（ユーザが記述した関数を拡張システム・コールとしてニュークリアスに登録）することができます。

なお，拡張システム・コールとして登録する関数に制限はなく，標準システム・コール（RX850 Pro が提供するシステム・コール）を含ませることも可能です。ただし，タスク状態からのみ発行可能な標準システム・コールを含ませた場合には，拡張システム・コールの発行状態が「タスク状態からのみ発行可能」に制限されます。

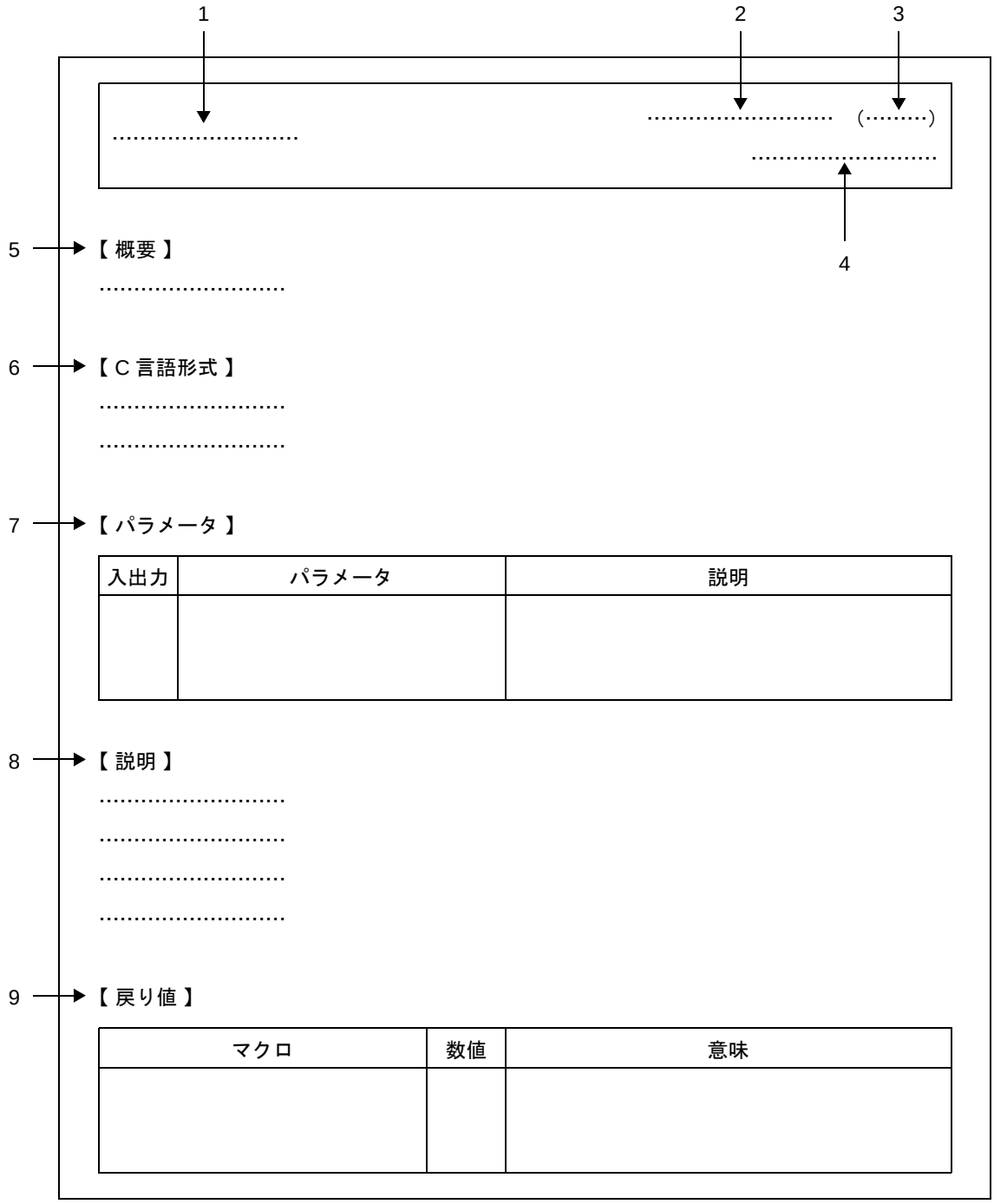
また，拡張システム・コールは，ユーザ定義のシステム・コールとして位置づけられる一方で，タスクに準じた部分を持ちます。つまり，標準システム・コールと同様に，処理が終了した際にはスケジューラが起動され，最適なタスクへの選択処理が行われます。

ただし，拡張システム・コール内に標準システム・コールを含ませた場合には，標準システム・コールの処理が終了した際にもスケジューラが起動されるため，拡張システム・コールの処理途中で，他タスクに制御が移ることがあります。

12.8　システム・コールの解説

この節では，RX850 Pro が提供するシステム・コールについて，次の記述フォーマットに従って解説します。

図 12 － 1　システム・コールの記述フォーマット



(1) 名称
システム・コールの名称を示しています。

(2) 名称の由来
システム・コール名称の由来を示しています。

(3) 機能コード
システム・コールの機能コードを示しています。

(4) 発行可能な状態
システム・コールの発行が可能な状態を示しています。

タスク：	タスクからのみ発行可能
非タスク：	非タスク（直接起動割り込みハンドラ，間接起動割り込みハンドラ，周期起動ハンドラ）からのみ発行可能
タスク／非タスク：	タスク，非タスクのどちらからも発行可能
直接起動割り込みハンドラ：	直接起動割り込みハンドラからのみ発行可能

(5)【 概要 】
システム・コールの機能概要を示しています。

(6)【 C 言語形式 】
システム・コールを C 言語で発行する際の記述形式を示しています。

(7)【 パラメータ 】
システム・コールのパラメータを次の形式で示しています。

入出力	パラメータ	説明

A： パラメータの種類

入 … RX850 Pro への入力パラメータ
出 … RX850 Pro からの出力パラメータ

B： パラメータのデータ・タイプ

C： パラメータの説明

(8)【 説明 】
システム・コールの機能について示しています。

(9)【 戻り値 】
システム・コールからの戻り値をマクロと数値で示しています。

* 付きの戻り値 … 「パラメータ・チェック機能あり／なし」の両方の場合に返す値
* なしの戻り値 … 「パラメータ・チェック機能あり」の場合にだけ返す値

12.8.1　タスク管理機能

ここでは，タスクの状態操作を行うシステム・コールのグループ（タスク管理機能）について説明します。
表 12 － 5 に，タスク管理機能の一覧を示します。

表 12 － 5　タスク管理機能

システム・コール	機能
cre_tsk	他タスクを生成する
del_tsk	他タスクを削除する
sta_tsk	他タスクを起動する
ext_tsk	自タスクを終了する
exd_tsk	自タスクを終了したあと，削除する
ter_tsk	他タスクを強制的に終了する
dis_dsp	ディスパッチ処理を禁止する
ena_dsp	ディスパッチ処理を許可する
chg_pri	タスクの優先度を変更する
rot_rdq	タスクのレディ・キューを回転する
rel_wai	他タスクの wait 状態を強制的に解除する
get_tid	自タスクの ID 番号を獲得する
ref_tsk	タスク情報を獲得する
vget_tid	タスクの ID 番号を獲得する

create task（-17）

cre_tsk

タスク

【 概要 】

他タスクを生成する。

【 C 言語形式 】

- ID 番号を指定する場合

```
#include    <stdrx85p.h>
ER          ercd = cre_tsk ( ID tskid, T_CTSK *pk_ctsk );
```

- ID 番号を指定しない場合

```
#include    <stdrx85p.h>
ER          ercd = cre_tsk ( ID_AUTO, T_CTSK *pk_ctsk, ID *p_tskid );
```

【 パラメータ 】

入出力	パラメータ	説明
入	ID　*tskid*;	タスクの ID 番号
入	T_CTSK　**pk_ctsk*;	タスク生成情報を格納したパケットの先頭アドレス
出	ID　**p_tskid*;	ID 番号を格納する領域のアドレス

【 タスク生成情報 T_CTSK の構造 】

```
typedef struct  t_ctsk {
        VP      exinf;          /* 拡張情報 */
        ATR     tskatr;         /* タスクの属性 */
        FP      task;           /* タスクの起動アドレス */
        PRI     itskpri;        /* タスクの起動時優先度（初期優先度）*/
        INT     stksz;          /* タスクのスタック・サイズ */
        VP      gp;             /* タスクの固有 gp レジスタ値 */
        VP      tp;             /* タスクの固有 tp レジスタ値 */
        ID      keyid;          /* タスクのキー ID 番号 */
} T_CTSK;
```

【 説明 】

RX850 Pro では，タスクの生成において「ID 番号を指定して生成する」，「ID 番号を指定しないで生成する」の 2 種類のインタフェースを用意しています。

- ID 番号を指定する場合
  pk_ctsk で指定された情報を基に，*tskid* で指定された ID 番号を持つタスクを生成します。これにより，対象タスクは non-existent 状態から dormant 状態へと遷移し，RX850 Pro の管理対象となります。

- ID 番号を指定しない場合
  pk_ctsk で指定された情報を基に，タスクを生成します。
  これにより，対象タスクは non-existent 状態から dormant 状態へと遷移し，RX850 Pro の管理対象となります。
  ID 番号の割り付けは RX850 Pro により行われ，割り付けられた ID 番号は *p_tskid* で指定される領域に格納されます。

次に，タスク生成情報の詳細を示します。

exinf …　拡張情報
対象タスクに関する“ユーザ独自の情報”を格納するための領域で，ユーザが自由に利用できます。
exinf に設定された情報は，処理プログラム（タスク，非タスク）から ref_tsk を発行することにより，ダイナミックに獲得できます。

tskatr …　タスクの属性
ビット 0 …　タスクの記述言語
TA_ASM（0）：　アセンブリ言語
TA_HLNG（1）：　C 言語

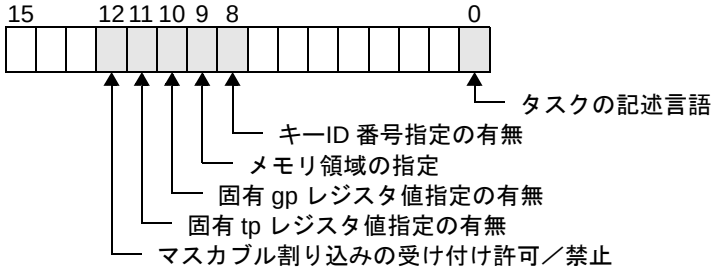
ビット 8 …　キー ID 番号指定の有無
TA_KEYID（1）：　キー ID 番号を指定

ビット 9 …　メモリ領域の指定
TA_SPOL0（0）：　システム・メモリ領域 0 番からスタックの領域を確保
TA_SPOL1（1）：　システム・メモリ領域 1 番からスタックの領域を確保

ビット 10 …　固有 gp レジスタ値指定の有無
TA_DPID（1）：　固有 gp レジスタ値を指定

ビット 11 …　固有 tp レジスタ値指定の有無
TA_DPIC（1）：　固有 tp レジスタ値を指定

ビット 12 …　マスカブル割り込みの受け付け許可／禁止
TA_ENAINT（0）：　タスク起動時，マスカブル割り込みの受け付けは許可状態
TA_DISINT（1）：　タスク起動時，マスカブル割り込みの受け付けは禁止状態



task …　タスクの起動アドレス

itskpri …　タスクの起動時優先度（初期優先度）

stksz …　タスクのスタック・サイズ（単位：バイト）

gp …　タスクの固有 gp レジスタ値

tp …　タスクの固有 tp レジスタ値

keyid …　タスクのキー ID 番号

備考　tskatr のビット 8 の値が 1（TA_KEYID）以外の場合，keyid の内容は意味を持ちません。
tskatr のビット 10 の値が 1（TA_DPID）以外の場合，gp の内容は意味を持ちません。
tskatr のビット 11 の値が 1（TA_DPIC）以外の場合，tp の内容は意味を持ちません。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOMEM	-10	タスク管理ブロックの領域が確保できない
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_RSATR	-24	属性 tskatr の指定が不正である

マクロ	数値	意味
E_PAR	-33	パラメータの指定が不正である - タスク生成情報を格納したパケットの先頭アドレスが不正（*pk_ctsk* = 0）である - 起動アドレスの指定が不正（task = 0）である - 起動時優先度の指定が不正（itskpri ≦ 0，最大優先度 ＜ itskpri）である - キー ID 番号の指定が不正（keyid = 0）である（TA_KEYID 属性指定時） - ID 番号を格納する領域のアドレスが不正（*p_tskid* = 0）である（ID 番号を指定しないで生成する場合）
E_ID	-35	ID 番号の指定が不正（最大タスク生成数 ＜ *tskid*）である
*E_OBJ	-63	指定した ID 番号を持つタスクがすでに生成されている
E_OACV	-66	アクセス権のない ID 番号（*tskid* ≦ 0）を指定した
E_CTX	-69	非タスクから本システム・コールを発行した

delete task（-18）

del_tsk

タスク

【概要】

他タスクを削除する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = del_tsk ( ID tskid );
```

【パラメータ】

入出力	パラメータ	説明
入	ID　*tskid*;	タスクの ID 番号

【説明】

tskid で指定されたタスクを dormant 状態から non-existent 状態へと遷移させます。
これにより，対象タスクは RX850 Pro の管理下から除外されます。
なお，自タスクを削除する場合には，exd_tsk を発行します。

備考　本システム・コールでは，削除要求のキューイングが行われません。このため，対象タスクが dormant 状態以外の場合には，戻り値として E_OBJ を返します。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_ID	-35	ID 番号の指定が不正（最大タスク生成数 < *tskid*）である
*E_NOEXS	-52	対象タスクが存在していない
*E_OBJ	-63	対象タスクが dormant 状態でない
E_OACV	-66	アクセス権のない ID 番号（*tskid* ≦ 0）を指定した
E_CTX	-69	非タスクから本システム・コールを発行した

sta_tsk

start task（-23）

タスク／非タスク

【概要】

他タスクを起動する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = sta_tsk ( ID tskid, INT stacd );
```

【パラメータ】

入出力	パラメータ	説明
入	ID　*tskid*;	タスクの ID 番号
入	INT　*stacd*;	起動コード

【説明】

tskid で指定されたタスクを dormant 状態から ready 状態へと遷移させます。
これにより，対象タスクは RX850 Pro のスケジューリング対象となります。
stacd には，対象タスクに引き渡す起動コードを指定します。対象タスクは，起動コードを関数パラメータと同様に取り扱うことで操作可能となります。

備考　本システム・コールでは，起動要求のキューイングが行われません。このため，対象タスクが dormant 状態以外の場合には，戻り値として E_OBJ を返します。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_ID	-35	ID 番号の指定が不正（最大タスク生成数 ＜ *tskid*）である
*E_NOEXS	-52	対象タスクが存在していない
*E_OBJ	-63	対象タスクが dormant 状態でない
E_OACV	-66	アクセス権のない ID 番号（*tskid* ≦ 0）を指定した

exit task（-21）

ext_tsk

タスク

【 概要 】

自タスクを終了する。

【 C 言語形式 】

```
#include    <stdrx85p.h>
void        ext_tsk ( void );
```

【 パラメータ 】

なし

【 説明 】

自タスクを run 状態から dormant 状態へと遷移させます。
これにより，自タスクは RX850 Pro のスケジューリング対象から除外されます。

備考 1　本システム・コールでは，タスク生成時（コンフィギュレーション時，または cre_tsk 発行時）に指定された“タスク生成情報”は初期化されます。

備考 2　タスクをアセンブリ言語で記述する場合，自タスクの終了は次のように記述してください。

```
jr          _ext_tsk
```

備考 3　本システム・コールを非タスク，またはディスパッチ禁止状態から発行した場合，動作は保証されません。

備考 4　本システム・コールでは，自タスクが終了する以前に獲得していた資源（メモリ・ブロック，セマフォ・カウントなど）の解放処理が行われません。したがって，本システム・コールを発行する前に，ユーザ側で資源の解放処理を行ってください。

【 戻り値 】

なし

exit and delete task（-22）

exd_tsk

タスク

【概要】

タスクを終了したあと，削除する。

【C 言語形式】

```
#include    <stdrx85p.h>
void        exd_tsk ( void );
```

【パラメータ】

なし

【説明】

自タスクを run 状態から non-existent 状態へと遷移させます。
これにより，自タスクは RX850 Pro の管理下から除外されます。

備考 1　タスクをアセンブリ言語で記述する場合，自タスクの終了，および削除は次のように記述してください。

```
        jr          _exd_tsk
```

備考 2　本システム・コールを非タスク，またはディスパッチ禁止状態から発行した場合，動作は保証されません。

備考 3　本システム・コールでは，自タスクが終了する以前に獲得していた資源（メモリ・ブロック，セマフォ・カウントなど）の解放処理が行われません。したがって，本システム・コールを発行する前に，ユーザ側で資源の解放処理を行ってください。

【戻り値】

なし

terminate task（-25）

ter_tsk

タスク

【 概要 】

他タスクを強制的に終了する。

【 C 言語形式 】

```
#include    <stdrx85p.h>
ER          ercd = ter_tsk ( ID tskid );
```

【 パラメータ 】

入出力	パラメータ	説明
入	ID　*tskid*;	タスクの ID 番号

【 説明 】

tskid で指定されたタスクを強制的に dormant 状態へと遷移させます。

備考 1　本システム・コールでは，タスク生成時（コンフィギュレーション時，または cre_tsk 発行時）に指定された“タスク生成情報”は初期化されます。

備考 2　本システム・コールでは，終了要求のキューイングが行われません。このため，対象タスクが ready 状態，wait 状態，suspend 状態，wait_suspend 状態以外の場合には，戻り値として E_NOEXS，または E_OBJ を返します。

備考 3　本システム・コールでは，自タスクが終了する以前に獲得していた資源（メモリ・ブロック，セマフォ・カウントなど）の解放処理が行われません。したがって，本システム・コールを発行する前にユーザ側で，資源の解放処理を行ってください。

【 戻り値 】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_ID	-35	ID 番号の指定が不正（最大タスク数 ＜ *tskid*）である
*E_NOEXS	-52	対象タスクが存在していない
*E_OBJ	-63	対象タスクが dormant 状態，または自タスクである
E_OACV	-66	アクセス権のない ID 番号（*tskid* ≦ 0）を指定した
E_CTX	-69	非タスクから本システム・コールを発行した

disable dispatch（-30）

dis_dsp

タスク

【概要】

ディスパッチ処理を禁止する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = dis_dsp ( void );
```

【パラメータ】

なし

【説明】

ディスパッチ処理（タスクのスケジューリング処理）を禁止します。
これにより，ena_dsp が発行されるまでの間，ディスパッチ処理が禁止されます。

RX850 Pro では，本システム・コールの発行から ena_dsp の発行までの間に，タスクのスケジューリング処理が必要なシステム・コール（chg_pri，sig_sem など）が発行された場合は，待ちキューのキュー操作などの処理を行うだけで，実際のスケジューリング処理は ena_dsp が発行されるまで遅延され，一括して行われます。

備考 1　本システム・コールでは，禁止要求のキューイングが行われません。このため，すでに本システム・コールが発行され，ディスパッチ処理が禁止されていた場合には，何も処理は行わず，エラーとしても扱いません。

備考 2　RX850 Pro では，本システム・コールの発行から ena_dsp の発行までの間に自タスクを wait 状態へと遷移させる可能性のあるシステム・コール（wai_sem，wai_flg など）を発行した場合は，待ち条件の即時成立／不成立にかかわらず，戻り値として E_CTX を返します。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
*E_CTX	-69	コンテキスト・エラー - 非タスクから本システム・コールを発行した - loc_cpu を発行後，本システム・コールを発行した

ena_dsp

enable dispatch（-29）

タスク

【概要】

ディスパッチ処理を許可する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = ena_dsp ( void );
```

【パラメータ】

なし

【説明】

ディスパッチ処理（タスクのスケジューリング処理）を許可します。
これにより，dis_dsp の発行により禁止されていたディスパッチ処理が再開されます。

RX850 Pro では，dis_dsp の発行から本システム・コールの発行までの間に，タスクのスケジューリング処理が必要なシステム・コール（chg_pri，sig_sem など）が発行された場合は，待ちキューのキュー操作などの処理を行うだけで，実際のスケジューリング処理は本システム・コールが発行されるまで遅延され，一括して行われます。

備考　本システム・コールでは，再開要求のキューイングが行われません。このため，すでに本システム・コールが発行され，ディスパッチ処理が再開されていた場合には，何も処理は行わず，エラーとしても扱いません。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
*E_CTX	-69	コンテキスト・エラー - 非タスクから本システム・コールを発行した - loc_cpu を発行後，本システム・コールを発行した

chg_pri

change priority（-27）

タスク／非タスク

【概要】

タスクの優先度を変更する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = chg_pri ( ID tskid, PRI tskpri );
```

【パラメータ】

入出力	パラメータ	説明
入	ID　*tskid*;	タスクの ID 番号 TSK_SELF（0）：　自タスク 数値：　タスクの ID 番号
入	PRI　*tskpri*;	タスクの優先度 TPRI_INI（0）：　タスクの起動時優先度 数値：　タスクの優先度

【説明】

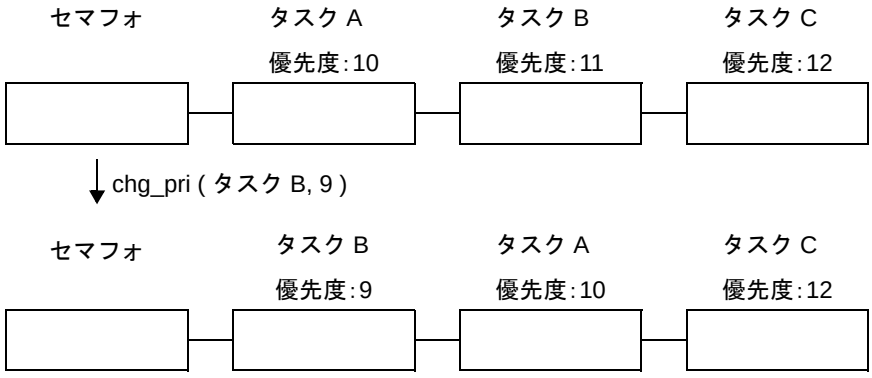
tskid で指定されたタスクの優先度を *tskpri* で指定された値に変更します。

対象タスクが run 状態，または ready 状態であった場合には，優先度の変更処理とともに，対象タスクを優先度に応じたレディ・キューの最後尾にキューイングしなおします。

備考 1　対象タスクが何らかの待ちキューに優先度順でキューイングされている場合には，本システム・コールの発行により，待ち順序が変わることがあります。

【例】

セマフォの待ちキューに 3 つのタスク（タスク A：優先度 10，タスク B：優先度 11，タスク C：優先度 12）が優先度順でキューイングされているとき，タスク B の優先度を“11”から“9”に変更した場合，待ちキューの待ち順序は次のように変更されます。



備考 2　*tskpri* で指定された値は，次に本システム・コールが発行されるまで，または対象タスクが dormant 状態へと遷移するまで有効になります。

備考 3　RX850 Pro におけるタスクの優先度は，その値が小さいほど高い優先度であることを示します。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_PAR	-33	優先度の指定が不正（*tskpri* ＜ 0，最大優先度 ＜ *tskpri*）である
E_ID	-35	ID 番号の指定が不正である - 最大タスク生成数 ＜ *tskid* - 非タスクから本システム・コールを発行したとき，*tskid* に TSK_SELF を指定した
*E_NOEXS	-52	対象タスクが存在していない
*E_OBJ	-63	対象タスクが dormant 状態である
E_OACV	-66	アクセス権のない ID 番号（*tskid* ＜ 0）を指定した

rot_rdq

rotate ready queue（-28）

タスク／非タスク

【概要】

タスクのレディ・キューを回転する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = rot_rdq ( PRI tskpri );
```

【パラメータ】

入出力	パラメータ	説明
入	PRI　*tskpri*;	タスクの優先度 TPRI_RUN（0）：　run 状態のタスクの優先度 数値：　タスクの優先度

【説明】

tskpri で指定された優先度に応じたレディ・キューの先頭タスクを最後尾にキューイングしなおします。

備考 1　対象優先度のレディ・キューにタスクが 1 つもキューイングされていない場合には，何も処理は行わず，エラーとしても扱いません。

備考 2　本システム・コールを一定周期で発行することにより，ラウンドロビン・スケジューリングを実現できます。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_PAR	-33	優先度の指定が不正（*tskpri* ＜ 0，最大優先度 ＜ *tskpri*）である

rel_wai

release wait（-31）

タスク／非タスク

【概要】

他タスクの wait 状態を強制的に解除する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = rel_wai ( ID tskid );
```

【パラメータ】

入出力	パラメータ	説明
入	ID　*tskid*;	タスクの ID 番号

【説明】

tskid で指定されたタスクの wait 状態を強制的に解除します。

これにより，対象タスクは待ちキューから外れ，wait 状態から ready 状態へ，または wait_suspend 状態から suspend 状態へと遷移します。

本システム・コールの発行により wait 状態を解除されたタスクには，wait 状態へと遷移するきっかけとなったシステム・コール（slp_tsk，wai_sem など）の戻り値として E_RLWAI が返されます。

備考　本システム・コールでは，suspend 状態の解除は行われません。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_ID	-35	ID 番号の指定が不正（最大タスク生成数 ＜ *tskid*）である
*E_NOEXS	-52	対象タスクが存在していない
*E_OBJ	-63	対象タスクが wait 状態，または wait_suspend 状態でない
E_OACV	-66	アクセス権のない ID 番号（*tskid* ≦ 0）を指定した

get_tid

get task identifier（-24）

タスク／非タスク

【概要】

自タスクの ID 番号を獲得する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = get_tid ( ID *p_tskid );
```

【パラメータ】

入出力	パラメータ	説明
出	ID　　*p_tskid;	ID 番号を格納する領域のアドレス

【説明】

自タスクの ID 番号を *p_tskid* で指定される領域に格納します。

備考　　本システム・コールを非タスクから発行した場合，*p_tskid* で指定される領域には FALSE（0）が格納されます。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_PAR	-33	ID 番号を格納する領域のアドレスが不正（*p_tskid* = 0）である

ref_tsk

refer task status（-20）

タスク／非タスク

【 概要 】

タスク情報を獲得する。

【 C 言語形式 】

```
#include    <stdrx85p.h>
ER          ercd = ref_tsk ( T_RTSK *pk_rtsk, ID tskid );
```

【 パラメータ 】

入出力	パラメータ	説明
出	T_RTSK　*pk_rtsk;	タスク情報を格納するパケットの先頭アドレス
入	ID　　tskid;	タスクの ID 番号 TSK_SELF（0）：　自タスク 数値：　　タスクの ID 番号

【 タスク情報 T_RTSK の構造 】

```
typedef struct  t_rtsk {
        VP      exinf;          /* 拡張情報 */
        PRI     tskpri;         /* 現在の優先度 */
        UINT    tskstat;        /* タスク状態 */
        UINT    tskwait;        /* 待ち要因 */
        ID      wid;            /* 待ちオブジェクト ID 番号 */
        INT     wupcnt;         /* 起床要求数 */
        INT     suscnt;         /* サスペンド要求数 */
        ID      keyid;          /* キー ID 番号 */
} T_RTSK;
```

【 説明 】

tskid で指定されたタスクの情報（拡張情報，現在の優先度など）を *pk_rtsk* で指定されるパケットに格納します。
次に，タスク情報の詳細を示します。

exinf …　拡張情報

tskpri …　現在の優先度

tskstat …　タスクの状態
- TTS_RUN（0x1）：　run 状態
- TTS_RDY（0x2）：　ready 状態
- TTS_WAI（0x4）：　wait 状態
- TTS_SUS（0x8）：　suspend 状態
- TTS_WAS（0xc）：　wait_suspend 状態
- TTS_DMT（0x10）：　dormant 状態

tskwait …　wait 状態の種類
- TTW_SLP（0x1）：　起床待ち状態
- TTW_DLY（0x2）：　時間経過待ち状態
- TTW_FLG（0x10）：　イベントフラグ待ち状態

TTW_SEM（0x20）:　資源待ち状態
TTW_MBX（0x40）:　メッセージ待ち状態
TTW_MPL（0x1000）: メモリ・ブロック待ち状態

wid …　待ちオブジェクト（セマフォ，イベントフラグなど）の ID 番号

wupcnt …　起床要求数

suscnt …　サスペンド要求数

keyid …　キー ID 番号
FALSE（0）:　生成時にキー ID 番号が指定されていない
数値 :　キー ID 番号

備考 1　tskstat の値が TTS_WAI，TTS_WAS 以外の場合，tskwait の内容は不定です。

備考 2　tskwait の値が TTW_FLG，TTW_SEM，TTW_MBX，TTW_MPF 以外の場合，wid の内容は不定です。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_PAR	-33	タスク情報を格納するパケットの先頭アドレスが不正（*pk_rtsk* = 0）である
E_ID	-35	ID 番号の指定が不正である - 最大タスク生成数 < *tskid* - 非タスクから本システム・コールを発行したとき，*tskid* に TSK_SELF を指定した
*E_NOEXS	-52	対象タスクが存在していない
E_OACV	-66	アクセス権のない ID 番号（*tskid* < 0）を指定した

vget_tid

get task Identifier（-248）

タスク／非タスク

【概要】

タスクの ID 番号を獲得する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = vget_tid ( ID *p_tskid, ID keyid );
```

【パラメータ】

入出力	パラメータ	説明
出	ID　*p_tskid;	ID 番号を格納する領域のアドレス
入	ID　keyid;	タスクのキー ID 番号

【説明】

keyid で指定されたタスクの ID 番号を *p_tskid* で指定される領域に格納します。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_PAR	-33	パラメータの指定が不正である - ID 番号を格納する領域のアドレスが不正（*p_tskid* = 0）である - キー ID 番号の指定が不正（*keyid* = 0）である
*E_NOEXS	-52	対象タスクが存在していない

12.8.2　タスク付属同期機能

ここでは，タスクに従属した同期操作を行うシステム・コールのグループ（タスク付属同期機能）について説明します。
表 12 － 6 に，タスク付属同期機能の一覧を示します。

表 12 － 6　タスク付属同期機能

システム・コール	機能
sus_tsk	他タスクを suspend 状態へ移行する
rsm_tsk	suspend 状態のタスクを再開する
frsm_tsk	suspend 状態のタスクを強制的に再開する
slp_tsk	自タスクを起床待ち状態へ移行する
tslp_tsk	自タスクを起床待ち状態へ移行する（タイムアウトあり）
wup_tsk	他タスクを起床する
can_wup	タスクの起床要求を無効化する

sus_tsk

suspend task（-33）

タスク／非タスク

【概要】

他タスクを suspend 状態へ移行する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = sus_tsk ( ID tskid );
```

【パラメータ】

入出力	パラメータ	説明
入	ID　*tskid*;	タスクの ID 番号

【説明】

tskid で指定されたタスクにサスペンド要求を発行（サスペンド要求カウンタに 0x1 を加算）します。

本システム・コールを発行した際，対象タスクが ready 状態，または wait 状態であった場合には，サスペンド要求の発行（サスペンド要求カウンタの加算処理）とともに，対象タスクを ready 状態から suspend 状態へ，または wait 状態から wait-suspend 状態へと遷移させます。

備考　RX850 Pro が管理するサスペンド要求カウンタは，7 ビット幅で構成されています。このため，本システム・コールの発行でサスペンド要求数が 127 回を越えた場合には，サスペンド要求カウンタの加算処理は行わず，戻り値として E_QOVR を返します。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_ID	-35	ID 番号の指定が不正（最大タスク生成数 ＜ *tskid*）である
*E_NOEXS	-52	対象タスクが存在していない
*E_OBJ	-63	指定したタスクの状態が不適当である - 対象タスクが dormant 状態である - タスクから本システム・コールを発行した際，対象タスクに自タスクを指定した
E_OACV	-66	アクセス権のない ID 番号（*tskid* ≦ 0）を指定した
*E_QOVR	-73	サスペンド要求数が 127 回を越えた

rsm_tsk

resume task（-35）

タスク／非タスク

【概要】

suspend 状態のタスクを再開する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = rsm_tsk ( ID tskid );
```

【パラメータ】

入出力	パラメータ	説明
入	ID　*tskid*;	タスクの ID 番号

【説明】

tskid で指定されたタスクに発行されているサスペンド要求を 1 回分だけ解除（サスペンド要求カウンタから 0x1 を減算）します。

本システム･コールの発行により対象タスクのサスペンド要求カウンタが0x0となった場合には, 対象タスクをsuspend状態から ready 状態へ，または wait_suspend 状態から wait 状態へと遷移させます。

備考　本システム・コールでは，解除要求のキューイングが行われません。このため，対象タスクが suspend 状態，またはwait_suspend状態でない場合には, サスペンド要求カウンタの減算処理は行わず, 戻り値としてE_OBJを返します。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_ID	-35	ID 番号の指定が不正（最大タスク生成数 ＜ *tskid*）である
*E_NOEXS	-52	対象タスクが存在していない
*E_OBJ	-63	対象タスクが suspend 状態，または wait_suspend 状態でない
E_OACV	-66	アクセス権のない ID 番号（*tskid* ≦ 0）を指定した

frsm_tsk

force resume task（-36）

タスク／非タスク

【概要】

suspend 状態のタスクを強制的に再開する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = frsm_tsk ( ID tskid );
```

【パラメータ】

入出力	パラメータ	説明
入	ID　*tskid*;	タスクの ID 番号

【説明】

tskid で指定されたタスクに発行されているサスペンド要求をすべて解除（サスペンド要求カウンタに 0x0 を設定）します。

これにより，対象タスクは suspend 状態から ready 状態へ，または wait_suspend 状態から wait 状態へと遷移します。

備考　本システム・コールでは，解除要求のキューイングが行われません。このため，対象タスクが suspend 状態，またはwait_suspend状態でない場合には，サスペンド要求カウンタの設定処理は行わず，戻り値としてE_OBJを返します。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_ID	-35	ID 番号の指定が不正（最大タスク生成数 ＜ *tskid*）である
*E_NOEXS	-52	対象タスクが存在していない
*E_OBJ	-63	対象タスクが suspend 状態，または wait_suspend 状態でない
E_OACV	-66	アクセス権のない ID 番号（*tskid* ≦ 0）を指定した

sleep task（-38）

slp_tsk

タスク

【 概要 】

自タスクを起床待ち状態へ移行する。

【 C 言語形式 】

```
#include    <stdrx85p.h>
ER          ercd = slp_tsk ( void );
```

【 パラメータ 】

なし

【 説明 】

自タスクに発行されている起床要求を 1 回分だけ解除（起床要求カウンタから 0x1 を減算）します。

ただし，本システム・コールを発行した際，自タスクの起床要求カウンタが 0x0 であった場合には，起床要求の解除（起床要求カウンタの減算処理）は行わず，自タスクを run 状態から wait 状態（起床待ち状態）へと遷移させます。

なお，起床待ち状態は wup_tsk，rel_wai が発行されると解除され，自タスクは ready 状態へと遷移します。

【 戻り値 】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_CTX	-69	コンテキスト・エラー - 非タスクから本システム・コールを発行した - ディスパッチ禁止状態から本システム・コールを発行した
*E_RLWAI	-86	rel_wai により起床待ち状態を強制的に解除された

tslp_tsk

sleep task with timeout（-37）

タスク

【 概要 】

自タスクを起床待ち状態へ移行する（タイムアウトあり）。

【 C 言語形式 】

```
#include    <stdrx85p.h>
ER          ercd = tslp_tsk ( TMO tmout );
```

【 パラメータ 】

入出力	パラメータ	説明
入	TMO　*tmout*;	待ち時間（単位：ms） TMO_POL（0）：　即時リターン TMO_FEVR（-1）：　永久待ち 数値：　待ち時間

【 説明 】

自タスクに発行されている起床要求を 1 回分だけ解除（起床要求カウンタから 0x1 を減算）します。

ただし，本システム・コールを発行した際，自タスクの起床要求カウンタが 0x0 であった場合には，起床要求の解除（起床要求カウンタの減算処理）は行わず，自タスクを run 状態から wait 状態（起床待ち状態）へと遷移させます。

なお，起床待ち状態は *tmout* で指定された待ち時間が経過するか，または wup_tsk，rel_wai が発行されると解除され，自タスクは ready 状態へと遷移します。

【 戻り値 】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_PAR	-33	待ち時間の指定が不正（*tmout* ＜ TMO_FEVR）である
E_CTX	-69	コンテキスト・エラー - 非タスクから本システム・コールを発行した - ディスパッチ禁止状態から本システム・コールを発行した
*E_TMOUT	-85	待ち時間が経過した
*E_RLWAI	-86	rel_wai により起床待ち状態を強制的に解除された

wup_tsk

wakeup task（-39）

タスク／非タスク

【概要】

他タスクを起床する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = wup_tsk ( ID tskid );
```

【パラメータ】

入出力	パラメータ	説明
入	ID　*tskid*;	タスクの ID 番号

【説明】

tskid で指定されたタスクに起床要求を発行（起床要求カウンタに 0x1 を加算）します。

ただし，本システム・コールを発行した際，対象タスクが wait 状態（起床待ち状態）であった場合には，起床要求の発行（起床要求カウンタの加算処理）は行わず，対象タスクを起床待ち状態から ready 状態へと遷移させます。

備考　RX850 Pro が管理する起床要求カウンタは，7 ビット幅で構成されています。このため，本システム・コールの発行により起床要求数が 127 回を越えた場合には，起床要求カウンタの加算処理は行わず，戻り値として E_QOVR を返します。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_ID	-35	ID 番号の指定が不正（最大タスク生成数 ＜ *tskid*）である
*E_NOEXS	-52	対象タスクが存在していない
*E_OBJ	-63	指定したタスクの状態が不適当である - 対象タスクが dormant 状態である - タスクから本システム・コールを発行した際，対象タスクに自タスクを指定した
E_OACV	-66	アクセス権のない ID 番号（*tskid* ≦ 0）を指定した
*E_QOVR	-73	起床要求数が 127 回を越えた

can_wup

cancel wakeup task（-40）

タスク／非タスク

【概要】

タスクの起床要求を無効化する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = can_wup ( INT *p_wupcnt, ID tskid );
```

【パラメータ】

入出力	パラメータ	説明
出	INT　*p_wupcnt;	起床要求数を格納する領域のアドレス
入	ID　tskid;	タスクの ID 番号 TSK_SELF（0）：　自タスク 数値：　タスクの ID 番号

【説明】

tskid で指定されたタスクに発行されている起床要求をすべて解除（起床要求カウンタに 0x0 を設定）します。
なお，本システム・コールの発行により解除された起床要求数は，*p_wupcnt* で指定される領域に格納されます。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_PAR	-33	起床要求数を格納する領域のアドレスが不正（*p_wupcnt* = 0）である
E_ID	-35	ID 番号の指定が不正である - 最大タスク生成数 ＜ *tskid* - 非タスクから本システム・コールを発行した際，*tskid* に TSK_SELF を指定した
*E_NOEXS	-52	対象タスクが存在していない
*E_OBJ	-63	対象タスクが dormant 状態である
E_OACV	-66	アクセス権のない ID 番号（*tskid* ＜ 0）を指定した

12.8.3　同期通信機能

ここでは，タスク間の同期（排他制御，待ち合わせ），および通信を行うシステム・コールのグループ（同期通信機能）について説明します。
表 12 － 7 に，同期通信機能の一覧を示します。

表 12 － 7　同期通信機能

システム・コール	機能
cre_sem	セマフォを生成する
del_sem	セマフォを削除する
sig_sem	資源を返却する
wai_sem	資源を獲得する
preq_sem	資源を獲得する（ポーリング）
twai_sem	資源を獲得する（タイムアウトあり）
ref_sem	セマフォ情報を獲得する
vget_sid	セマフォの ID 番号を獲得する
cre_flg	イベントフラグを生成する
del_flg	イベントフラグを削除する
set_flg	ビット・パターンをセットする
clr_flg	ビット・パターンをクリアする
wai_flg	ビット・パターンをチェックする
pol_flg	ビット・パターンをチェックする（ポーリング）
twai_flg	ビット・パターンをチェックする（タイムアウトあり）
ref_flg	イベントフラグ情報を獲得する
vget_fid	イベントフラグの ID 番号を獲得する
cre_mbx	メールボックスを生成する
del_mbx	メールボックスを削除する
snd_msg	メッセージを送信する
rcv_msg	メッセージを受信する
prcv_msg	メッセージを受信する（ポーリング）
trcv_msg	メッセージを受信する（タイムアウトあり）
ref_mbx	メールボックス情報を獲得する
vget_mid	メールボックスの ID 番号を獲得する

cre_sem

create semaphore（-49）

タスク

【概要】

セマフォを生成する。

【C 言語形式】

- ID 番号を指定する場合

```
#include    <stdrx85p.h>
ER          ercd = cre_sem ( ID semid, T_CSEM *pk_csem );
```

- ID 番号を指定しない場合

```
#include    <stdrx85p.h>
ER          ercd = cre_sem ( ID_AUTO, T_CSEM *pk_csem, ID *p_semid );
```

【パラメータ】

入出力	パラメータ	説明
入	ID　*semid*;	セマフォの ID 番号
入	T_CSEM　**pk_csem*;	セマフォ生成情報を格納したパケットの先頭アドレス
出	ID　**p_semid*;	ID 番号を格納する領域のアドレス

【セマフォ生成情報 T_CSEM の構造】

```
typedef struct  t_csem {
        VP      exinf;          /* 拡張情報 */
        ATR     sematr;         /* セマフォの属性 */
        INT     isemcnt;        /* セマフォの初期資源数 */
        INT     maxsem;         /* セマフォの最大資源数 */
        ID      keyid;          /* セマフォのキー ID 番号 */
} T_CSEM;
```

【説明】

RX850 Pro では，セマフォの生成において「ID 番号を指定して生成する」，「ID 番号を指定しないで生成する」の 2 種類のインタフェースを用意しています。

- ID 番号を指定する場合
  pk_csem で指定された情報を基に，*semid* で指定された ID 番号を持つセマフォを生成します。

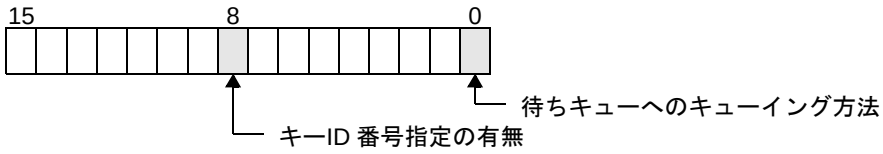
- ID 番号を指定しない場合
  pk_csem で指定された情報を基に，セマフォを生成します。
  ID番号の割り付けはRX850 Proにより行われ，割り付けられたID番号は，*p_semid*で指定される領域に格納されます。

次に，セマフォ生成情報の詳細を示します。

exinf …　拡張情報
対象セマフォに関する“ユーザ独自の情報”を格納するための領域で，ユーザが自由に利用できます。
exinf に設定された情報は，処理プログラム（タスク，非タスク）から ref_sem を発行することにより，ダイナミックに獲得できます。

sematr … セマフォの属性
ビット 0 … 待ちキューへのキューイング方法
TA_TPRI（0）:　優先度順
TA_TFIFO（1）:　FIFO 順

ビット 8 … キー ID 番号指定の有無
TA_KEYID（1）:　キー ID 番号を指定



isemcnt … セマフォの初期資源数

maxsem …セマフォの最大資源数

keyid …　セマフォのキー ID 番号

備考　sematr のビット 8 の値が TA_KEYID 以外の場合，keyid の内容は意味を持ちません。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOMEM	-10	セマフォ管理ブロックの領域が確保できない
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_RSATR	-24	属性 sematr の指定が不正である
E_PAR	-33	パラメータの指定が不正である - セマフォ生成情報を格納したパケットの先頭アドレスが不正（*pk_csem* = 0）である - 初期資源数の指定が不正（isemcnt ＜ 0）である - 最大資源数の指定が不正（maxsem ≦ 0，maxsem ＜ isemcnt）である - キー ID 番号の指定が不正（keyid = 0）である（TA_KEYID 属性指定時） - ID 番号を格納する領域のアドレスが不正（*p_semid* = 0）である（ID 番号を指定しないで生成する場合）
E_ID	-35	ID 番号の指定が不正（最大セマフォ生成数 ＜ *semid*）である
*E_OBJ	-63	指定した ID 番号を持つセマフォがすでに生成されている
E_OACV	-66	アクセス権のない ID 番号（*semid* ≦ 0）を指定した
E_CTX	-69	非タスクから本システム・コールを発行した

del_sem

delete semaphore（-50）

タスク

【概要】

セマフォを削除する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = del_sem ( ID semid );
```

【パラメータ】

入出力	パラメータ	説明
入	ID　*semid*;	セマフォの ID 番号

【説明】

semid で指定されたセマフォを削除します。

これにより，対象セマフォは RX850 Pro の管理下から除外されます。

本システム・コールの発行により wait 状態（資源待ち状態）を解除されたタスクには，wait 状態へと遷移するきっかけとなったシステム・コール（wai_sem，twai_sem）の戻り値として E_DLT が返されます。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_ID	-35	ID 番号の指定が不正（最大セマフォ生成数 ＜ *semid*）である
*E_NOEXS	-52	対象セマフォが存在していない
E_OACV	-66	アクセス権のない ID 番号（*semid* ≦ 0）を指定した
E_CTX	-69	非タスクから本システム・コールを発行した

sig_sem

signal semaphore（-55）

タスク／非タスク

【概要】

資源を返却する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = sig_sem ( ID semid );
```

【パラメータ】

入出力	パラメータ	説明
入	ID　*semid*;	セマフォの ID 番号

【説明】

semid で指定されたセマフォに資源を返却（セマフォ・カウンタに 0x1 を加算）します。

ただし，本システム・コールを発行した際，対象セマフォの待ちキューにタスクがキューイングされていた場合には，資源の返却処理（セマフォ・カウンタの加算処理）は行わず，該当タスク（待ちキューの先頭タスク）に資源を渡します。

これにより，該当タスクは待ちキューから外れ，wait 状態（資源待ち状態）から ready 状態へ，または wait_suspend 状態から suspend 状態へと遷移します。

備考　RX850 Pro が管理するセマフォ・カウンタは，生成時に指定された資源数としてとりうる最大資源数までしかカウントしません。したがって，本システム・コールの発行により，資源数が最大資源数を越えるような場合には，セマフォ・カウンタの加算処理は行われず，戻り値として E_QOVR が返されます。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_ID	-35	ID 番号の指定が不正（最大セマフォ生成数 ＜ *semid*）である
*E_NOEXS	-52	対象セマフォが存在していない
E_OACV	-66	アクセス権のない ID 番号（*semid* ≦ 0）を指定した
*E_QOVR	-73	資源数が生成時に指定した最大資源数を越えた

wai_sem

wait on semaphore（-53）

タスク

【概要】

資源を獲得する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = wai_sem ( ID semid );
```

【パラメータ】

入出力	パラメータ	説明
入	ID　*semid*;	セマフォの ID 番号

【説明】

semid で指定されたセマフォから資源を獲得（セマフォ・カウンタから 0x1 を減算）します。

ただし，本システム・コールを発行した際，対象セマフォから資源を獲得できなかった（空き資源が存在しなかった）場合には，自タスクを対象セマフォの待ちキューにキューイングしたあと，run 状態から wait 状態（資源待ち状態）へと遷移させます。

なお，資源待ち状態は sig_sem，del_sem，rel_wai が発行されると解除され，自タスクは ready 状態へと遷移します。

備考　自タスクを対象セマフォの待ちキューにキューイングする際は，対象セマフォ生成時（コンフィギュレーション時，または cre_sem 発行時）に指定された順（FIFO 順，または優先度順）に行われます。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_ID	-35	ID 番号の指定が不正（最大セマフォ生成数 ＜ *semid*）である
*E_NOEXS	-52	対象セマフォが存在していない
E_OACV	-66	アクセス権のない ID 番号（*semid* ≦ 0）を指定した
E_CTX	-69	コンテキスト・エラー - 非タスクから本システム・コールを発行した - ディスパッチ禁止状態から本システム・コールを発行した
*E_DLT	-81	del_sem により対象セマフォが削除された
*E_RLWAI	-86	rel_wai により資源待ち状態が強制的に解除された

preq_sem

poll and request semaphore（-107）

タスク／非タスク

【概要】

資源を獲得する（ポーリング）。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = preq_sem ( ID semid );
```

【パラメータ】

入出力	パラメータ	説明
入	ID　*semid*;	セマフォの ID 番号

【説明】

semid で指定されたセマフォから資源を獲得（セマフォ・カウンタから 0x1 を減算）します。

ただし，本システム・コールを発行した際，対象セマフォから資源を獲得できなかった（空き資源が存在しなかった）場合には，戻り値として E_TMOUT を返します。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_ID	-35	ID 番号の指定が不正（最大セマフォ生成数 ＜ *semid*）である
*E_NOEXS	-52	対象セマフォが存在していない
E_OACV	-66	アクセス権のない ID 番号（*semid* ≦ 0）を指定した
*E_TMOUT	-85	対象セマフォの資源数が 0x0 である

twai_sem

wait on semaphore with timeout（-171）

タスク

【概要】

資源を獲得する（タイムアウトあり）。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = twai_sem ( ID semid, TMO tmout );
```

【パラメータ】

入出力	パラメータ	説明
入	ID　*semid*;	セマフォの ID 番号
入	TMO　*tmout*;	待ち時間（単位：ms） TMO_POL（0）：　即時リターン TMO_FEVR（-1）：　永久待ち 数値：　待ち時間

【説明】

semid で指定されたセマフォから資源を獲得（セマフォ・カウンタから 0x1 を減算）します。

ただし，本システム・コールを発行した際，対象セマフォから資源を獲得できなかった（空き資源が存在しなかった）場合には，自タスクを対象セマフォの待ちキューにキューイングしたあと，run 状態から wait 状態（資源待ち状態）へと遷移させます。

なお，資源待ち状態は *tmout* で指定された時間が経過したとき，または sig_sem，del_sem，rel_wai が発行されたときに解除され，自タスクは ready 状態へと遷移します。

備考　自タスクを対象セマフォの待ちキューにキューイングする際は，対象セマフォ生成時（コンフィギュレーション時，または cre_sem 発行時）に指定された順（FIFO 順，または優先度順）に行われます。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_PAR	-33	待ち時間の指定が不正（*tmout* ＜ TMO_FEVR）である
E_ID	-35	ID 番号の指定が不正（最大セマフォ生成数 ＜ *semid*）である
*E_NOEXS	-52	対象セマフォが存在していない
E_OACV	-66	アクセス権のない ID 番号（*semid* ≦ 0）を指定した
E_CTX	-69	コンテキスト・エラー - 非タスクから本システム・コールを発行した - ディスパッチ禁止状態から本システム・コールを発行した
*E_DLT	-81	del_sem により対象セマフォが削除された
*E_TMOUT	-85	待ち時間が経過した

マクロ	数値	意味
*E_RLWAI	-86	rel_wai により資源待ち状態が強制的に解除された

ref_sem

refer semaphore status（-52）

タスク／非タスク

【概要】

セマフォ情報を獲得する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = ref_sem ( T_RSEM *pk_rsem, ID semid );
```

【パラメータ】

入出力	パラメータ	説明
出	T_RSEM *pk_rsem;	セマフォ情報を格納するパケットの先頭アドレス
入	ID semid;	セマフォの ID 番号

【セマフォ情報 T_RSEM の構造】

```
typedef struct  t_rsem {
        VP      exinf;          /* 拡張情報 */
        BOOL_ID wtsk;           /* 待ちタスクの有無 */
        INT     semcnt;         /* 現在の資源数 */
        INT     maxsem;         /* 最大資源数 */
        ID      keyid;          /* キー ID 番号 */
} T_RSEM;
```

【説明】

semid で指定されたセマフォのセマフォ情報（拡張情報，待ちタスクの有無など）を *pk_rsem* で指定されるパケットに格納します。

次に，セマフォ情報の詳細を示します。

exinf …　拡張情報

wtsk …　待ちタスクの有無
- FALSE（0）：　待ちタスクなし
- 数値：　待ちキューの先頭タスクの ID 番号

semcnt … 現在の資源数

maxsem …生成時に指定した最大資源数

keyid …　キー ID 番号
- FALSE（0）：　生成時にキー ID 番号が指定されていない
- 数値：　キー ID 番号

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない

マクロ	数値	意味
E_PAR	-33	セマフォ情報を格納するパケットの先頭アドレスが不正（*pk_rsem* = 0）である
E_ID	-35	ID 番号の指定が不正（最大セマフォ生成数 ＜ *semid*）である
*E_NOEXS	-52	対象セマフォが存在していない
E_OACV	-66	アクセス権のない ID 番号（*semid* ≦ 0）を指定した

vget_sid

get semaphore identifier（-246）

タスク／非タスク

【概要】

セマフォの ID 番号を獲得する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = vget_sid ( ID *p_semid, ID keyid );
```

【パラメータ】

入出力	パラメータ	説明
出	ID　*p_semid;	ID 番号を格納する領域のアドレス
入	ID　keyid;	セマフォのキー ID 番号

【説明】

keyid で指定されたセマフォの ID 番号を *p_semid* で指定される領域に格納します。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_PAR	-33	パラメータの指定が不正である - ID 番号を格納する領域のアドレスが不正（*p_semid* = 0）である - キー ID 番号の指定が不正（*keyid* = 0）である
*E_NOEXS	-52	対象セマフォが存在していない

cre_flg

create eventflag（-41）

タスク

【 概要 】

イベントフラグを生成する。

【 C 言語形式 】

- ID 番号を指定する場合

```
#include    <stdrx85p.h>
ER          ercd = cre_flg ( ID flgid, T_CFLG *pk_cflg );
```

- ID 番号を指定しない場合

```
#include    <stdrx85p.h>
ER          ercd = cre_flg ( ID_AUTO, T_CFLG *pk_cflg, ID *p_flgid );
```

【 パラメータ 】

入出力	パラメータ	説明
入	ID　*flgid*;	イベントフラグの ID 番号
入	T_CFLG　**pk_cflg*;	イベントフラグ生成情報を格納したパケットの先頭アドレス
出	ID　**p_flgid*;	ID 番号を格納する領域のアドレス

【 イベントフラグ生成情報 T_CFLG の構造 】

```
typedef struct  t_cflg {
        VP      exinf;          /* 拡張情報 */
        ATR     flgatr;         /* イベントフラグの属性 */
        UINT    iflgptn;        /* イベントフラグの初期ビット・パターン */
        ID      keyid;          /* イベントフラグのキー ID 番号 */
} T_CFLG;
```

【 説明 】

RX850 Pro では，イベントフラグの生成において「ID 番号を指定して生成する」，「ID 番号を指定しないで生成する」の 2 種類のインタフェースを用意しています。

- ID 番号を指定する場合
  pk_cflg で指定された情報を基に，*flgid* で指定された ID 番号を持つイベントフラグを生成します。

- ID 番号を指定しない場合
  pk_cflg で指定された情報を基に，イベントフラグを生成します。
  ID 番号の割り付けは RX850 Pro により行われ，割り付けられた ID 番号は，*p_flgid* で指定される領域に格納されます。

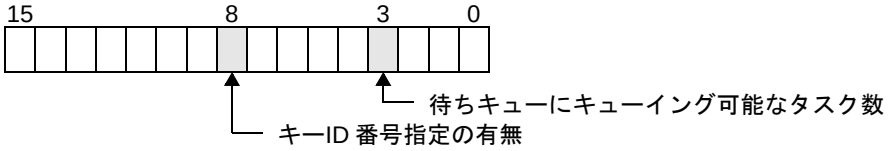
次に，イベントフラグ生成情報の詳細を示します。

exinf …　拡張情報
対象イベントフラグに関する“ユーザ独自の情報”を格納するための領域で，ユーザが自由に利用できます。
exinf に設定された情報は，処理プログラム（タスク，非タスク）から ref_flg を発行することにより，ダイナミックに獲得できます。

flgatr …　イベントフラグの属性

ビット 3 …　待ちキューにキューイング可能なタスク数
TA_WSGL（0）：　1 タスクのみ
TA_WMUL（1）：　複数タスク

ビット 8 …　キー ID 番号指定の有無
TA_KEYID（1）：　キー ID 番号を指定



iflgptn …　イベントフラグの初期ビット・パターン

keyid …　イベントフラグのキー ID 番号

備考　　flgatr のビット 8 の値が TA_KEYID 以外の場合，keyid の内容は意味を持ちません。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOMEM	-10	イベントフラグ管理ブロックの領域が確保できない
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_RSATR	-24	属性 flgatr の指定が不正である
E_PAR	-33	パラメータの指定が不正である - イベントフラグ生成情報を格納したパケットの先頭アドレスが不正（*pk_cflg* = 0）である - キー ID 番号の指定が不正（keyid = 0）である（TA_KEYID 属性指定時） - ID 番号を格納する領域のアドレスが不正（*p_flgid* = 0）である（ID 番号を指定しないで生成する場合）
E_ID	-35	ID 番号の指定が不正（最大イベントフラグ生成数 ＜ *flgid*）である
*E_OBJ	-63	指定した ID 番号を持つイベントフラグがすでに生成されている
E_OACV	-66	アクセス権のない ID 番号（*flgid* ≦ 0）を指定した
E_CTX	-69	非タスクから本システム・コールを発行した

del_flg

delete eventflag（-42）

タスク

【概要】

イベントフラグを削除する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = del_flg ( ID flgid );
```

【パラメータ】

入出力	パラメータ	説明
入	ID　*flgid*;	イベントフラグの ID 番号

【説明】

flgid で指定されたイベントフラグを削除します。

これにより，対象イベントフラグは RX850 Pro の管理下から除外されます。

なお，本システム・コールの発行により wait 状態（イベントフラグ待ち）を解除されたタスクには，wait 状態へと遷移するきっかけとなったシステム・コール（wai_flg，twai_flg）の戻り値として E_DLT が返されます。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_ID	-35	ID 番号の指定が不正（最大イベントフラグ生成数 ＜ *flgid*）である
*E_NOEXS	-52	対象イベントフラグが存在していない
E_OACV	-66	アクセス権のない ID 番号（*flgid* ≦ 0）を指定した
E_CTX	-69	非タスクから本システム・コールを発行した

set_flg

set eventflag（-48）

タスク／非タスク

【概要】

ビット・パターンをセットする。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = set_flg ( ID flgid, UINT setptn );
```

【パラメータ】

入出力	パラメータ	説明
入	ID　*flgid*;	イベントフラグの ID 番号
入	UINT　*setptn*;	セットするビット・パターン（32 ビット幅）

【説明】

flgid で指定されたイベントフラグのビット・パターンと *setptn* で指定されたビット・パターンの論理和（OR）をとり，その結果を対象イベントフラグに設定します。

たとえば，本システム・コールを発行するとき，対象イベントフラグのビット・パターンが B’1100，*setptn* で指定されたビット・パターンが B’1010 の場合，対象イベントフラグのビット・パターンは B’1110 となります。

なお，本システム・コールを発行した際，対象イベントフラグの待ちキューにキューイングされているタスクの待ち条件を満足した場合には，該当タスクを待ち状態から外します。

これにより，該当タスクは wait 状態（イベントフラグ待ち）から ready 状態へ，または wait_suspend 状態から suspend 状態へと遷移します。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_ID	-35	ID 番号の指定が不正（最大イベントフラグ生成数 < *flgid*）である
*E_NOEXS	-52	対象イベントフラグが存在していない
E_OACV	-66	アクセス権のない ID 番号（*flgid* ≦ 0）を指定した

clr_flg

clear eventflag（-47）

タスク／非タスク

【概要】

ビット・パターンをクリアする。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = clr_flg ( ID flgid, UINT clrptn );
```

【パラメータ】

入出力	パラメータ	説明
入	ID　*flgid*;	イベントフラグの ID 番号
入	UINT　*clrptn*;	クリアするビット・パターン（32 ビット幅）

【説明】

flgid で指定されたイベントフラグのビット・パターンと *clrptn* で指定されたビット・パターンの論理積（AND）をとり，その結果を対象イベントフラグに設定します。

たとえば，本システム・コールを発行するとき，対象イベントフラグのビット・パターンが B’1100，*clrptn* で指定されたビット・パターンが B’1010 の場合，対象イベントフラグのビット・パターンは B’1000 となります。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_ID	-35	ID 番号の指定が不正（最大イベントフラグ生成数 ＜ *flgid*）である
*E_NOEXS	-52	対象イベントフラグが存在していない
E_OACV	-66	アクセス権のない ID 番号（*flgid* ≦ 0）を指定した

wai_flg

wait eventflag（-46）

タスク

【概要】

ビット・パターンをチェックする。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = wai_flg ( UINT *p_flgptn, ID flgid, UINT waiptn, UINT wfmode );
```

【パラメータ】

入出力	パラメータ	説明
出	UINT *p_flgptn;	条件成立時のビット・パターンを格納する領域のアドレス
入	ID flgid;	イベントフラグの ID 番号
入	UINT waiptn;	要求するビット・パターン（32 ビット幅）
入	UINT wfmode;	待ち条件／条件成立時の指定 TWF_ANDW（0）：　AND 待ち TWF_ORW（2）：　OR 待ち TWF_CLR（1）：　ビット・パターンをクリアする

【説明】

waiptn で指定された要求ビット・パターンと *wfmode* で指定された待ち条件を満足するビット・パターンが，*flgid* で指定されたイベントフラグに設定されているかどうかをチェックします。

待ち条件を満足するビット・パターンが対象イベントフラグに設定されていた場合には，対象イベントフラグのビット・パターンを *p_flgptn* で指定される領域に格納します。

ただし，本システム・コールを発行した際，対象イベントフラグのビット・パターンが待ち条件を満足していなかった場合には，自タスクを対象イベントフラグの待ちキューの最後尾にキューイングしたあと，run 状態から wait 状態（イベントフラグ待ち状態）へと遷移させます。

なお，イベントフラグ待ちの状態は，set_flg の発行により待ち条件を満足するようなビット・パターンが設定されたとき，または del_flg，rel_wai が発行されたときに解除され，自タスクは ready 状態へと遷移します。

次に，*wfmode* の指定形式を示します。

- *wfmode* = TWF_ANDW
  waiptn で“1”を設定している全ビットが対象イベントフラグに設定されているかどうかをチェックします。

- *wfmode* = (TWF_ANDW | TWF_CLR)
  waiptn で“1”を設定している全ビットが対象イベントフラグに設定されているかどうかをチェックします。
  なお，待ち条件が満足されたときには，対象イベントフラグのビット・パターンがクリア（B’0000 を設定）されます。

- *wfmode* = TWF_ORW
  waiptn で“1”を設定しているビットのうち，いずれかのビットが対象イベントフラグに設定されているかどうかをチェックします。

- *wfmode* = (TWF_ORW | TWF_CLR)
  waiptn で“1”を設定しているビットのうち，いずれかのビットが対象イベントフラグに設定されているかどうかをチェックします。
  なお，待ち条件が満足されたときには，対象イベントフラグのビット・パターンがクリア（B’0000 を設定）されます。

備考 1　RX850 Pro では，イベントフラグの待ちキューにキューイング可能なタスク数を，生成時（コンフィギュレーション時，または cre_flg 発行時）に次のいずれかに指定します。

TA_WSGL 属性：　1 タスクのみキューイング可能
TA_WMUL 属性：　複数タスクをキューイング可能

このため，すでに待ちタスクがキューイングされている TA_WSGL 属性のイベントフラグに対して本システム・コールを発行した場合には，ビット・パターンのチェック処理は行わず，戻り値として E_OBJ を返します。

備考 2　del_flg，rel_wai の発行によりイベントフラグ待ち状態が強制的に解除された場合には，*p_flgptn* で指定される領域の内容は不定になります。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_PAR	-33	パラメータの指定が不正である - 条件成立時のビット・パターンを格納する領域のアドレスが不正（*p_flgptn* = 0）である - 要求するビット・パターンの指定が不正（*waiptn* = 0）である - 待ち条件／条件成立時 *wfmode* の指定が不正である
E_ID	-35	ID 番号の指定が不正（最大イベントフラグ生成数 ＜ *flgid*）である
*E_NOEXS	-52	対象イベントフラグが存在していない
*E_OBJ	-63	すでに待ちタスクがキューイングされているTA_WSGL属性のイベントフラグに対して，本システム・コールを発行した
E_OACV	-66	アクセス権のない ID 番号（*flgid* ≦ 0）を指定した
E_CTX	-69	コンテキスト・エラー - 非タスクから本システム・コールを発行した - ディスパッチ禁止状態から本システム・コールを発行した
*E_DLT	-81	del_flg により対象イベントフラグが削除された
*E_RLWAI	-86	rel_wai によりイベントフラグ待ち状態が強制的に解除された

pol_flg

poll eventflag（-106）

タスク／非タスク

【概要】

ビット・パターンをチェックする（ポーリング）。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = pol_flg ( UINT *p_flgptn, ID flgid, UINT waiptn, UINT wfmode );
```

【パラメータ】

入出力	パラメータ	説明
出	UINT *p_flgptn;	条件成立時のビット・パターンを格納する領域のアドレス
入	ID flgid;	イベントフラグの ID 番号
入	UINT waiptn;	要求するビット・パターン（32 ビット幅）
入	UINT wfmode;	待ち条件／条件成立時の指定 TWF_ANDW（0）：　AND 待ち TWF_ORW（2）：　OR 待ち TWF_CLR（1）：　ビット・パターンをクリアする

【説明】

waiptn で指定された要求ビット・パターンと *wfmode* で指定された待ち条件が *flgid* で指定されたイベントフラグに設定されているかどうかをチェックします。

待ち条件を満足するピット・パターンが対象イベントフラグに設定されていた場合には，対象イベントフラグのビット・パターンを *p_flgptn* で指定される領域に格納します。

ただし，本システム・コールを発行した際，対象イベントフラグのビット・パターンが待ち条件を満していなかった場合には，戻り値として E_TMOUT が返されます。

次に，*wfmode* の指定形式を示します。

- *wfmode* = TWF_ANDW
  waiptn で“1”を設定している全ビットが対象イベントフラグに設定されているかどうかをチェックします。

- *wfmode* = (TWF_ANDW | TWF_CLR)
  waiptn で“1”を設定している全ビットが対象イベントフラグに設定されているかどうかをチェックします。
  なお，待ち条件が満足されていたときには，対象イベントフラグのビット・パターンがクリア（B’0000 を設定）されます。

- *wfmode* = TWF_ORW
  waiptn で“1”を設定しているビットのうち，いずれかのビットが対象イベントフラグに設定されているかどうかをチェックします。

- *wfmode* = (TWF_ORW | TWF_CLR)
  waiptn で“1”を設定しているビットのうち，いずれかのビットが対象イベントフラグに設定されているかどうかをチェックします。
  なお，待ち条件が満足されていたときには，対象イベントフラグのビット・パターンがクリア（B’0000 を設定）されます。

備考　RX850 Pro では，イベントフラグの待ちキューにキューイング可能なタスク数を，生成時（コンフィギュレーション時，または cre_flg 発行時）に次のいずれかに指定します。

TA_WSGL 属性：　1 タスクのみキューイング可能
TA_WMUL 属性：　複数タスクをキューイング可能

このため，すでに待ちタスクがキューイングされている TA_WSGL 属性のイベントフラグに対して本システム・コールを発行した場合には，ビット・パターンのチェック処理は行わず，戻り値として E_OBJ を返します。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_PAR	-33	パラメータの指定が不正である - 条件成立時のビット・パターンを格納する領域のアドレスが不正（*p_flgptn* = 0）である - 要求するビット・パターンの指定が不正（*waiptn* = 0）である - 待ち条件／条件成立時 *wfmode* の指定が不正である
E_ID	-35	ID 番号の指定が不正（最大イベントフラグ生成数 ＜ *flgid*）である
*E_NOEXS	-52	対象イベントフラグが存在していない
*E_OBJ	-63	すでに待ちタスクがキューイングされているTA_WSGL属性のイベントフラグに対して本システム・コールを発行した
E_OACV	-66	アクセス権のない ID 番号（*flgid* ≦ 0）を指定した
*E_TMOUT	-85	対象イベントフラグのビット・パターンが待ち条件を満足していない

twai_flg

wait eventflag with timeout（-170）

タスク

【概要】

ビット・パターンをチェックする（タイムアウトあり）。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = twai_flg ( UINT *p_flgptn, ID flgid, UINT waiptn, UINT wfmode, TMO
tmout );
```

【パラメータ】

入出力	パラメータ	説明
出	UINT *p_flgptn;	条件成立時のビット・パターンを格納する領域のアドレス
入	ID flgid;	イベントフラグの ID 番号
入	UINT waiptn;	要求するビット・パターン（32 ビット幅）
入	UINT wfmode;	待ち条件／条件成立時の指定 TWF_ANDW（0）：　AND 待ち TWF_ORW（2）：　OR 待ち TWF_CLR（1）：　ビット・パターンをクリアする
入	TMO tmout;	待ち時間（単位：ms） TMO_POL（0）：　即時リターン TMO_FEVR（-1）：　永久待ち 数値：　待ち時間

【説明】

waiptn で指定された要求ビット・パターンと *wfmode* で指定された待ち条件が *flgid* で指定されたイベントフラグに設定されているかどうかをチェックします。

待ち条件を満足するピット・パターンが対象イベントフラグに設定されていた場合には，対象イベントフラグのビット・パターンを *p_flgptn* で指定される領域に格納します。

ただし，本システム・コール発行した際，対象イベントフラグのビット・パターンが待ち条件を満たしていなかった場合には，自タスクを対象イベントフラグの待ちキューの最後尾にキューイングしたあと，run 状態から wait 状態（イベントフラグ待ち状態）へと遷移させます。

なお，イベントフラグ待ちの状態は，*tmout* で指定された待ち時間が経過したとき，set_flg の発行により待ち条件を満足するようなビット・パターンが設定されたとき，または del_flg，rel_wai が発行されたときに解除され，自タスクは ready 状態へと遷移します。

次に，*wfmode* の指定形式を示します。

- *wfmode* = TWF_ANDW
  waiptn で“1”を設定している全ビットが対象イベントフラグに設定されているかどうかをチェックします。

- *wfmode* = (TWF_ANDW | TWF_CLR)
  waiptn で“1”を設定している全ビットが対象イベントフラグに設定されているかどうかをチェックします。
  なお，待ち条件が満足されたときには，対象イベントフラグのビット・パターンがクリア（B’0000 を設定）されます。

- *wfmode* = TWF_ORW
  waiptn で“1”を設定しているビットのうち，いずれかのビットが対象イベントフラグに設定されているかどうかをチェックします。

- *wfmode* = (TWF_ORW | TWF_CLR)
  waiptn で“1”を設定しているビットのうち，いずれかのビットが対象イベントフラグに設定されているかどうかをチェックします。
  なお，待ち条件が満足されたときには，対象イベントフラグのビット・パターンがクリア（B’0000 を設定）されます。

備考 1　RX850 Pro では，イベントフラグの待ちキューにキューイング可能なタスク数を，生成時（コンフィギュレーション時，または cre_flg 発行時）に次のいずれかに指定します。

TA_WSGL 属性：　1 タスクのみキューイング可能
TA_WMUL 属性：　複数タスクをキューイング可能

このため，すでに待ちタスクがキューイングされている TA_WSGL 属性のイベントフラグに対して本システム・コールを発行した場合には，ビット・パターンのチェック処理は行わず，戻り値として E_OBJ を返します。

備考 2　del_flg，rel_wai の発行によりイベントフラグ待ち状態が強制的に解除された場合には，*p_flgptn* で指定される領域の内容は不定になります。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_PAR	-33	パラメータの指定が不正である - 条件成立時のビット・パターンを格納する領域のアドレスが不正（*p_flgptn* = 0）である - 要求するビット・パターンの指定が不正（*waiptn* = 0）である - 待ち条件／条件成立時 *wfmode* の指定が不正である - 待ち時間の指定が不正（*tmout* ＜ TMO_FEVR）である
E_ID	-35	ID 番号の指定が不正（最大イベントフラグ生成数 ＜ *flgid*）である
*E_NOEXS	-52	対象イベントフラグが存在していない
*E_OBJ	-63	すでに待ちタスクがキューイングされているTA_WSGL属性のイベントフラグに対して本システム・コールを発行した
E_OACV	-66	アクセス権のない ID 番号（*flgid* ≦ 0）を指定した
E_CTX	-69	コンテキスト・エラー - 非タスクから本システム・コールを発行した - ディスパッチ禁止状態から本システム・コールを発行した
*E_DLT	-81	del_flg により対象イベントフラグが削除された
*E_TMOUT	-85	待ち時間が経過した
*E_RLWAI	-86	rel_wai によりイベントフラグ待ち状態が強制的に解除された

ref_flg

refer eventflag status（-44）

タスク／非タスク

【概要】

イベントフラグ情報を獲得する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = ref_flg ( T_RFLG *pk_rflg, ID flgid );
```

【パラメータ】

入出力	パラメータ	説明
出	T_RFLG *pk_rflg;	イベントフラグ情報を格納するパケットの先頭アドレス
入	ID flgid;	イベントフラグの ID 番号

【イベントフラグ情報 T_RFLG の構造】

```
typedef struct  t_rflg {
        VP      exinf;          /* 拡張情報 */
        BOOL_ID wtsk;           /* 待ちタスクの有無 */
        UINT    flgptn;         /* 現在のビット・パターン */
        ID      keyid;          /* キー ID 番号 */
} T_RFLG;
```

【説明】

flgid で指定されたイベントフラグのイベントフラグ情報（拡張情報，待ちタスクの有無など）を *pk_rflg* で指定されるパケットに格納します。
次に，イベントフラグ情報の詳細を示します。

exinf … 拡張情報

wtsk … 待ちタスクの有無
FALSE（0）： 待ちタスクなし
数値： 待ちキューの先頭タスクの ID 番号

flgptn … 現在のビット・パターン

keyid … キー ID 番号
FALSE（0）： 生成時にキー ID 番号が指定されていない
数値： キー ID 番号

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_PAR	-33	イベントフラグ情報を格納するパケットの先頭アドレスが不正（*pk_rflg* = 0）である

マクロ	数値	意味
E_ID	-35	ID 番号の指定が不正（最大イベントフラグ生成数 ＜ *flgid*）
*E_NOEXS	-52	対象イベントフラグが存在していない
E_OACV	-66	アクセス権のない ID 番号（*flgid* ≦ 0）を指定した

vget_fid

get eventflag identifier（-247）

タスク／非タスク

【概要】

イベントフラグの ID 番号を獲得する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = vget_fid ( ID *p_flgid, ID keyid );
```

【パラメータ】

入出力	パラメータ	説明
出	ID　*p_flgid;	ID 番号を格納する領域のアドレス
入	ID　keyid;	イベントフラグのキー ID 番号

【説明】

keyid で指定されたイベントフラグの ID 番号を *p_flgid* で指定される領域に格納します。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_PAR	-33	パラメータの指定が不正である - ID 番号を格納する領域のアドレスが不正（*p_flgid* = 0）である - キー ID 番号の指定が不正（*keyid* = 0）である
*E_NOEXS	-52	対象イベントフラグが存在していない

create mailbox（-57）

cre_mbx

タスク

【 概要 】

メールボックスを生成する。

【 C 言語形式 】

- ID 番号を指定する場合

```
#include    <stdrx85p.h>
ER          ercd = cre_mbx ( ID mbxid, T_CMBX *pk_cmbx );
```

- ID 番号を指定しない場合

```
#include    <stdrx85p.h>
ER          ercd = cre_mbx ( ID_AUTO, T_CMBX *pk_cmbx, ID *p_mbxid );
```

【 パラメータ 】

入出力	パラメータ	説明
入	ID　*mbxid*;	メールボックスの ID 番号
入	T_CMBX　**pk_cmbx*;	メールボックス生成情報を格納したパケットの先頭アドレス
出	ID　**p_mbxid*;	ID 番号を格納する領域のアドレス

【 メールボックス生成情報 T_CMBX の構造 】

```
typedef struct  t_cmbx {
        VP      exinf;          /* 拡張情報 */
        ATR     mbxatr;         /* メールボックスの属性 */
        ID      keyid;          /* メールボックスのキー ID 番号 */
} T_CMBX;
```

【 説明 】

RX850 Pro では，メールボックスの生成において「ID 番号を指定して生成する」，「ID 番号を指定しないで生成する」の 2 種類のインタフェースを用意しています。

- ID 番号を指定する場合
  pk_cmbx で指定された情報を基に，*mbxid* で指定された ID 番号を持つメールボックスを生成します。

- ID 番号を指定しない場合
  pk_cmbx で指定された情報を基に，メールボックスを生成します。
  ID番号の割り付けはRX850 Proにより行われ，割り付けられたID番号は，*p_mbxid*で指定される領域に格納されます。

次に，メールボックス生成情報の詳細を示します。

exinf …　拡張情報
exinf は対象メールボックスに関する“ユーザ独自の情報”を格納するための領域で，ユーザが自由に利用できます。
exinf に設定された情報は，処理プログラム（タスク，非タスク）から ref_mbx を発行することにより，ダイナミックに獲得できます。

mbxatr …　メールボックスの属性

ビット 0 …　タスク用待ちキューへのキューイング方法

TA_TPRI（0）：　優先度順

TA_TFIFO（1）：　FIFO 順

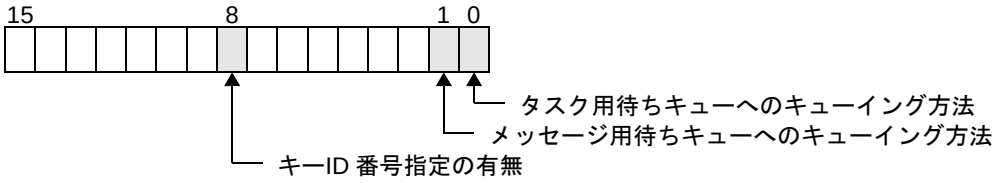
ビット 1 …　メッセージ用待ちキューへのキューイング方法

TA_MPRI（0）：　優先度順

TA_MFIFO（1）：　FIFO 順

ビット 8 …　キー ID 番号指定の有無

TA_KEYID（1）：　キー ID 番号を指定



keyid …　メールボックスのキー ID 番号

備考　mbxatr のビット 8 の値が TA_KEYID 以外の場合，keyid の内容は意味を持ちません。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOMEM	-10	メールボックス管理ブロックの領域が確保できない
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_RSATR	-24	属性 mbxatr の指定が不正である
E_PAR	-33	パラメータの指定が不正である - メールボックス生成情報を格納したパケットの先頭アドレスが不正（*pk_cmbx* = 0）である - キーID 番号の指定が不正（keyid = 0）である（TA_KEYID 指定時） - ID 番号を格納する領域のアドレスが不正（*p_mbxid* = 0）である（ID 番号を指定しないで生成する場合）
E_ID	-35	ID 番号の指定が不正（最大メールボックス生成数 ＜ *mbxid*）である
*E_OBJ	-63	指定した ID 番号を持つメールボックスがすでに生成されている
E_OACV	-66	アクセス権のない ID 番号（*mbxid* ≦ 0）を指定した
E_CTX	-69	非タスクから本システム・コールを発行した

del_mbx

delete mailbox（-58）

タスク

【概要】

メールボックスを削除する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = del_mbx ( ID mbxid );
```

【パラメータ】

入出力	パラメータ	説明
入	ID　*mbxid*;	メールボックスの ID 番号

【説明】

mbxid で指定されたメールボックスを削除します。

これにより，対象メールボックスは RX850 Pro の管理下から除外されます。

なお，本システム・コールの発行により wait 状態（メッセージ待ち状態）を解除されたタスクには，wait 状態へと遷移するきっかけとなったシステム・コール（rcv_msg，trcv_msg）の戻り値として E_DLT が返されます。

備考　本システム・コールを発行したときに，対象メールボックスのメッセージ用待ちキューにメモリ・プールから獲得したメモリ・ブロックを使用したメッセージがキューイングされていた場合，メッセージ（メモリ・ブロック）は該当メモリ・プールに返却されます。
このため，メモリ・プールから獲得したメモリ・ブロック以外の領域をメッセージとして使用していた場合には，動作保証外となりますので，本システム・コールは発行しないでください。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_ID	-35	ID 番号の指定が不正（最大メールボックス生成数 ＜ *mbxid*）である
*E_NOEXS	-52	対象メールボックスが存在していない
E_OACV	-66	アクセス権のない ID 番号（*mbxid* ≦ 0）を指定した
E_CTX	-69	非タスクから本システム・コールを発行した

snd_msg

send message（-63）

タスク／非タスク

【概要】

メッセージを送信する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = snd_msg ( ID mbxid, T_MSG *pk_msg );
```

【パラメータ】

入出力	パラメータ	説明
入	ID　*mbxid*;	メールボックスの ID 番号
入	T_MSG　**pk_msg*;	メッセージを格納したパケットの先頭アドレス

【メッセージ T_MSG の構造】

```
typedef struct  t_msg {
        VW      msgrfu;         /* メッセージ管理領域 */
        PRI     msgpri;         /* メッセージの優先度 */
        VB      msgcont[];      /* メッセージ本体 */
} T_MSG;
```

【説明】

mbxid で指定されたメールボックスに *pk_msg* で指定されたメッセージを送信（メッセージ用待ちキューにメッセージをキューイング）します。

ただし，本システム・コールを発行した際，対象メールボックスのタスク用待ちキューにタスクがキューイングされていた場合には，メッセージのキューイング操作は行われず，該当タスク（タスク用待ちキューの先頭タスク）にメッセージを渡します。

これにより，該当タスクはタスク用待ちキューから外れ，wait 状態（メッセージ待ち状態）から ready 状態へ，または wait_suspend 状態から suspend 状態へと遷移します。

備考 1　メッセージを対象メールボックスのメッセージ用待ちキューにキューイングする際は，対象メールボックス生成時（コンフィギュレーション時，または cre_mbx 発行時）に指定された順（FIFO 順，または優先度順）に行われます。

備考 2　RX850 Pro では，メッセージの先頭 4 バイト（メッセージ管理領域 msgrfu）をメッセージ用待ちキューにキューイングする際のリンク領域として使用します。このため，メッセージを対象メールボックスに送信する場合は，本システム・コールを発行する前に msgrfu に“0x0”を設定する必要があります。
本システム・コールを発行した際，msgrfu に“0x0”以外の値が設定されていた場合には，RX850 Pro が「対象メッセージはすでにメッセージ用待ちキューにキューイングされている」と判断し，メッセージの送信処理は行わず，戻り値として E_OBJ を返します。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない

マクロ	数値	意味
E_PAR	-33	メッセージを格納したパケットの先頭アドレスが不正（*pk_msg* = 0）である
E_ID	-35	ID 番号の指定が不正（最大メールボックス生成数 < *mbxid*）である
*E_NOEXS	-52	対象メールボックスが存在していない
E_OBJ	-63	指定したメッセージ用の領域がすでにメッセージとして使用されている
E_OACV	-66	アクセス権のない ID 番号（*mbxid* ≦ 0）を指定した

rcv_msg

receive message from mailbox（-61）

タスク

【概要】

メッセージを受信する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = rcv_msg ( T_MSG **ppk_msg, ID mbxid );
```

【パラメータ】

入出力	パラメータ	説明
出	T_MSG **ppk_msg;	メッセージの先頭アドレスを格納する領域のアドレス
入	ID mbxid;	メールボックスの ID 番号

【説明】

mbxid で指定されたメールボックスからメッセージを受信し，その先頭アドレスを *ppk_msg* で指定される領域に格納します。

ただし，本システム・コールを発行した際，対象メールボックスからメッセージを受信できなかった（メッセージ用待ちキューにメッセージが存在しなかった）場合には，自タスクを対象メールボックスのタスク待ち用待ちキューにキューイングしたあと，run 状態から wait 状態（メッセージ待ち状態）へと遷移させます。

なお，メッセージ待ち状態は snd_msg，del_mbx，rel_wai が発行されると解除され，自タスクは ready 状態へと遷移します。

備考　自タスクを対象メールボックスのタスク用待ちキューにキューイングする際は，対象メールボックス生成時（コンフィギュレーション時，または cre_mbx 発行時）に指定された順（FIFO 順，または優先度順）に行われます。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_PAR	-33	メッセージの先頭アドレスを格納する領域のアドレスが不正（*ppk_msg* = 0）である
E_ID	-35	ID 番号の指定が不正（最大メールボックス生成数 < *mbxid*）である
*E_NOEXS	-52	対象メールボックスが存在していない
E_OACV	-66	アクセス権のない ID 番号（*mbxid* ≦ 0）を指定した
E_CTX	-69	コンテキスト・エラー - 非タスクから本システム・コールを発行した - ディスパッチ禁止状態から本システム・コールを発行した
*E_DLT	-81	del_mbx により対象メールボックスが削除された
*E_RLWAI	-86	rel_wai によりメッセージ待ち状態が強制的に解除された

poll and receive message from mailbox（-108）

prcv_msg

タスク／非タスク

【概要】

メッセージを受信する（ポーリング）。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = prcv_msg ( T_MSG **ppk_msg, ID mbxid );
```

【パラメータ】

入出力	パラメータ	説明
出	T_MSG **ppk_msg;	メッセージの先頭アドレスを格納する領域のアドレス
入	ID mbxid;	メールボックスの ID 番号

【説明】

mbxid で指定されたメールボックスからメッセージを受信し，その先頭アドレスを *ppk_msg* で指定される領域に格納します。

ただし，本システム・コールを発行した際，対象メールボックスからメッセージを受信できなかった（メッセージ用待ちキューにメッセージが存在しなかった）場合には，戻り値として E_TMOUT が返されます。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_PAR	-33	メッセージの先頭アドレスを格納する領域アドレスが不正（*ppk_msg* = 0）である
E_ID	-35	ID 番号の指定が不正（最大メールボックス生成数 < *mbxid*）である
*E_NOEXS	-52	対象メールボックスが存在していない
E_OACV	-66	アクセス権のない ID 番号（*mbxid* ≦ 0）を指定した
*E_TMOUT	-85	対象メールボックスにメッセージが存在していない

trcv_msg

receive message from mailbox with timeout（-172）

タスク

【概要】

メッセージを受信する（タイムアウトあり）。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = trcv_msg ( T_MSG **ppk_msg, ID mbxid, TMO tmout );
```

【パラメータ】

入出力	パラメータ	説明
出	T_MSG **ppk_msg;	メッセージの先頭アドレスを格納する領域のアドレス
入	ID mbxid;	メールボックスの ID 番号
入	TMO tmout;	待ち時間（単位：ms） TMO_POL（0）：　即時リターン TMO_FEVR（-1）：　永久待ち 数値：　待ち時間

【説明】

mbxid で指定されたメールボックスからメッセージを受信し，その先頭アドレスを *ppk_msg* で指定される領域に格納します。

ただし，本システム・コールを発行した際，対象メールボックスからメッセージを受信できなかった（メッセージ用待ちキューにメッセージが存在しなかった）場合には，自タスクを対象メールボックスのタスク用待ちキューにキューイングしたあと，run 状態から wait 状態（メッセージ待ち状態）へと遷移させます。

なお，メッセージ待ち状態は *tmout* で指定された待ち時間が経過したとき，または snd_msg，del_mbx，rel_wai が発行発行されたときに解除され，自タスクは ready 状態へ遷移します。

備考　自タスクを対象メールボックスのタスク用待ちキューにキューイングする際は，対象メールボックス生成時（コンフィギュレーション時，または cre_mbx 発行時）に指定された順（FIFO 順，または優先度順）に行われます。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_PAR	-33	メッセージの先頭アドレスを格納する領域のアドレスが不正（*ppk_msg* = 0）である
E_ID	-35	ID 番号の指定が不正（最大メールボックス生成数 < *mbxid*）である
*E_NOEXS	-52	対象メールボックスが存在していない
E_OACV	-66	アクセス権のない ID 番号（*mbxid* ≦ 0）を指定した

マクロ	数値	意味
E_CTX	-69	コンテキスト・エラー - 非タスクから本システム・コールを発行した - ディスパッチ禁止状態から本システム・コールを発行した
*E_DLT	-81	del_mbx によりメールボックスが削除された
*E_TMOUT	-85	待ち時間が経過した
*E_RLWAI	-86	rel_wai によりメッセージ待ち状態が強制的に解除された

ref_mbx

refer mailbox status（-60）

タスク／非タスク

【概要】

メールボックス情報を獲得する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = ref_mbx ( T_RMBX *pk_rmbx, ID mbxid );
```

【パラメータ】

入出力	パラメータ	説明
出	T_RMBX *pk_rmbx;	メールボックス情報を格納するパケットの先頭アドレス
入	ID mbxid;	メールボックスの ID 番号

【メールボックス情報 T_RMBX の構造】

```
typedef struct  t_rmbx {
        VP      exinf;          /* 拡張情報 */
        BOOL_ID wtsk;           /* 待ちタスクの有無 */
        T_MSG   *pk_msg;        /* 待ちメッセージの有無 */
        ID      keyid;          /* キー ID 番号 */
} T_RMBX;
```

【説明】

mbxid で指定されたメールボックスのメールボックス情報（拡張情報，待ちタスクの有無など）を *pk_rmbx* で指定されたパケットに格納します。
次に，メールボックス情報の詳細を示します。

exinf … 拡張情報

wtsk … 待ちタスクの有無
- FALSE（0）： 待ちタスクなし
- 数値： 待ちキューの先頭タスクの ID 番号

pk_msg … 待ちメッセージの有無
- NADR（-1）： 待ちメッセージなし
- 数値： 待ちキューの先頭メッセージのアドレス

keyid … キー ID 番号
- FALSE（0）： 生成時にキー ID 番号が指定されていない
- 数値： キー ID 番号

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない

マクロ	数値	意味
E_PAR	-33	メールボックス情報を格納するパケットの先頭アドレスが不正（*pk_rmbx* = 0）である
E_ID	-35	ID 番号指定が不正（最大メールボックス生成数 < *mbxid*）である
*E_NOEXS	-52	対象メールボックスが存在していない
E_OACV	-66	アクセス権のない ID 番号（*mbxid* ≦ 0）を指定した

vget_mid

get mailbox identifier（-245）

タスク／非タスク

【概要】

メールボックスの ID 番号を獲得する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = vget_mid ( ID *p_mbxid, ID keyid );
```

【パラメータ】

入出力	パラメータ	説明
出	ID　*p_mbxid;	ID 番号を格納する領域のアドレス
入	ID　keyid;	メールボックスのキー ID 番号

【説明】

keyid で指定されたメールボックスの ID 番号を *p_mbxid* で指定される領域に格納します。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_PAR	-33	パラメータの指定が不正である - ID 番号を格納する領域のアドレスが不正（*p_mbxid* = 0）である - キー ID 番号の指定が不正（*keyid* = 0）である
*E_NOEXS	-52	対象メールボックスが存在していない

12.8.4　割り込み処理管理機能

ここでは，マスカブル割り込みに依存した処理を行うシステム・コールのグループ（割り込み処理管理機能）について説明します。

表 12 － 8 に，割り込み処理管理機能の一覧を示します。

表 12 － 8　割り込み処理管理機能

システム・コール	機能
def_int	間接起動割り込みハンドラを登録／登録解除する
ena_int	マスカブル割り込みの受け付けを再開する
dis_int	マスカブル割り込みの受け付けを禁止する
loc_cpu	マスカブル割り込みの受け付けとディスパッチ処理を禁止する
unl_cpu	マスカブル割り込みの受け付けとディスパッチ処理を再開する
chg_icr	割り込み制御レジスタの内容を変更する
ref_icr	割り込み制御レジスタの内容を獲得する

def_int

define interrupt handler（-65）

タスク／非タスク

【概要】

間接起動割り込みハンドラを登録／登録解除する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = def_int ( UINT eintno, T_DINT *pk_dint );
```

【パラメータ】

入出力	パラメータ	説明
入	UINT　*eintno*;	間接起動割り込みハンドラの割り込み要因番号
入	T_DINT　**pk_dint*;	間接起動割り込みハンドラ登録情報を格納したパケットの先頭アドレス

【間接起動割り込みハンドラ登録情報 T_DINT の構造】

```
typedef struct  t_dint {
        ATR     intatr;         /* 間接起動割り込みハンドラの属性 */
        FP      inthdr;         /* 間接起動割り込みハンドラの起動アドレス */
        VP      gp;             /* 間接起動割り込みハンドラの固有 gp レジスタ値 */
        VP      tp;             /* 間接起動割り込みハンドラの固有 tp レジスタ値 */
} T_DINT;
```

【説明】

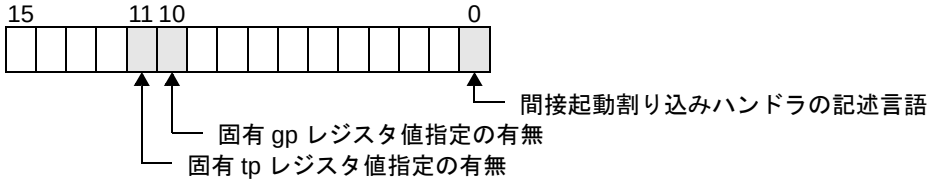
pk_dint で指定された情報を基に，*eintno* で指定された割り込み要因番号のマスカブル割り込みが発生した際に起動する間接起動割り込みハンドラを登録します。
次に，間接起動割り込みハンドラ登録情報の詳細を示します。

intatr …　間接起動割り込みハンドラの属性

ビット 0 …　間接起動割り込みハンドラの記述言語
- TA_ASM（0）:　アセンブリ言語
- TA_HLNG（1）:　C 言語

ビット 10 … 固有 gp レジスタ値指定の有無
- TA_DPID（1）:　固有 gp レジスタ値を指定

ビット 11 … 固有 tp レジスタ値指定の有無
- TA_DPIC（1）:　固有 tp レジスタ値を指定



inthdr …　間接起動割り込みハンドラの起動アドレス

gp …　間接起動割り込みハンドラの固有 gp レジスタ値

tp …　間接起動割り込みハンドラの固有 tp レジスタ値

本システム・コールを発行した際，対象割り込み要因番号に対応した間接起動割り込みハンドラがすでに登録されていた場合には，エラーとしては扱わず，本システム・コールで指定された間接起動割り込みハンドラを新規に登録します。
また，本システム・コールを発行した際，*pk_dint* で指定される領域に NADR（-1）を設定した場合，*eintno* で指定された間接起動割り込みハンドラの登録が解除されます。

備考 1　intatr のビット 10 の値が 1（TA_DPID）以外の場合，gp の内容は意味を持ちません。

備考 2　intatr のビット 11 の値が 1（TA_DPIC）以外の場合，tp の内容は意味を持ちません。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOMEM	-10	割り込みハンドラ管理ブロックの領域が確保できない
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_RSATR	-24	属性 intatr の指定が不正である
E_PAR	-33	パラメータの指定が不正である - 割り込み要因番号の指定が不正（*eintno* ＜ 0，最大割り込み要因番号 ＜ *eintno*）である - 間接起動割り込みハンドラ登録情報を格納したパケットの先頭アドレスが不正（*pk_dint* = 0）である - 起動アドレスの指定が不正（inthdr = 0）である

enable interrupt（-71）

ena_int

タスク／非タスク

【概要】

マスカブル割り込みの受け付けを再開する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = ena_int ( void );
```

【パラメータ】

なし

【説明】

dis_int の発行により禁止されていたマスカブル割り込みの受け付けを再開します。

なお，RX850 Pro では，dis_int の発行から本システム・コールの発行までの間にマスカブル割り込みが発生していた場合，該当する割り込み処理（直接起動割り込みハンドラ，間接起動割り込みハンドラ）への移行は本システム・コールが発行されるまで遅延されます。

備考　本システム・コールでは，再開要求のキューイングは行われません。したがって，すでに本システム・コールが発行され，マスカブル割り込みの受け付けが許可されていた場合には，なにも処理は行わず，エラーとしても扱いません。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない

dis_int

disable interrupt（-72）

タスク／非タスク

【概要】

マスカブル割り込みの受け付けを禁止する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = dis_int ( void );
```

【パラメータ】

なし

【説明】

マスカブル割り込みの受け付けを禁止します。
これにより，ena_int が発行されるまでの間，マスカブル割り込みの受け付けが禁止されます。
RX850 Pro では，本システム・コールの発行から ena_int の発行までの間に，マスカブル割り込みが発行していた場合，該当する割り込み処理（直接起動割り込みハンドラ，間接起動割り込みハンドラ）への移行は ena_int が発行されるまで遅延されます。

備考　本システム・コールでは，禁止要求のキューイングが行われません。このため，すでに本システム・コールが発行され，マスカブル割り込みの受け付けが禁止されていた場合には，何も処理は行わず，エラーとしても扱いません。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない

loc_cpu

lock CPU（-8）

タスク

【概要】

マスカブル割り込みの受け付けとディスパッチ処理を禁止する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = loc_cpu ( void );
```

【パラメータ】

なし

【説明】

マスカブル割り込みの受け付けとディスパッチ処理（タスクのスケジューリング処理）を禁止します。

したがって，本システム・コールの発行から unl_cpu が発行されるまでの間は，ほかのハンドラやタスクに制御が移ることはありません。

なお，RX850 Pro では，本システム・コールの発行から unl_cpu の発行までの間にマスカブル割り込みが発生していた場合，該当する割り込み処理（割り込みハンドラへの移行）は unl_cpu が発行されるまで遅延されます。また，タスクのスケジューリング処理が必要なシステム・コール（chg_pri，sig_sem など）が発行された場合，待ちキューのキュー操作などの処理を行うだけであり，実際のスケジューリング処理は unl_cpu が発行されるまで遅延され，一括して行われます。

備考　本システム・コールでは，禁止要求のキューイングは行われません。したがって，すでに本システム・コールが発行され，マスカブル割り込みの受け付けとディスパッチ処理が禁止されていた場合には，何も処理は行わず，エラーとしても扱いません。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_CTX	-69	非タスクから本システム・コールを発行した

unl_cpu

unlock CPU（-7）

タスク

【概要】

マスカブル割り込みの受け付けとディスパッチ処理を許可する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = unl_cpu ( void );
```

【パラメータ】

なし

【説明】

loc_cpu の発行により禁止されていたマスカブル割り込みの受け付けとディスパッチ処理（タスクのスケジューリング処理）を再開します。

なお，RX850 Pro では，loc_cpu の発行から本システム・コールの発行までの間にマスカブル割り込みが発生していた場合，該当する割り込み処理（割り込みハンドラ）への移行は本システム・コールが発行されるまで遅延されます。また，タスクのスケジューリング処理が必要なシステム・コール（chg_pri，sig_sem など）が発行された場合，待ちキューのキュー操作などの処理を行うだけであり，実際のスケジューリング処理は本システム・コールが発行されるまで遅延され，一括して行われます。

備考 1　dis_dsp の発行により禁止されたディスパッチ処理は，本システム・コールの発行により再開できます。

備考 2　本システム・コールでは，再開要求のキューイングは行われません。したがって，すでに本システム・コールが発行され，マスカブル割り込みの受け付けとディスパッチ処理が再開されていた場合には，何も処理は行わず，エラーとしても扱いません。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_CTX	-69	非タスクから本システム・コールを発行した

chg_icr

change interrupt control register（-67）

タスク／非タスク

【概要】

割り込み制御レジスタの内容を変更する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = chg_icr ( UINT eintno, UB icrcmd );
```

【パラメータ】

入出力	パラメータ	説明
入	UINT　*eintno*;	割り込み要因番号
入	UB　*icrcmd*;	割り込み要求フラグの指定 ICR_CLRINT（0x20）:　割り込み要求なし 割り込みマスク・フラグの指定 ICR_CLRMSK（0x10）:　割り込み処理を許可 ICR_SETMSK（0x40）:　割り込み処理を禁止 割り込み優先順位変更の指定 ICR_CHGLVL（0x08）:　割り込み優先順位を変更 割り込み優先順位の指定 数値（0 ～ 7）:　割り込み優先順位

【説明】

eintno で指定された割り込み制御レジスタの内容を *icrcmd* で指定された値に変更します。
次に，*icrcmd* の指定形式を示します。

- *icrcmd* = ICR_CLRINT
  割り込み制御レジスタの割り込み要求フラグを“0”に変更します。

- *icrcmd* = ICR_CLRMSK
  割り込み制御レジスタの割り込みマスク・フラグを“0”に変更します。

- *icrcmd* = ICR_SETMSK
  割り込み制御レジスタの割り込みマスク・フラグを“1”に変更します。

- *icrcmd* = (ICR_CHGLVL | 数値)
  割り込み制御レジスタの割り込み優先順位を“数値”で指定された値に変更します。
  なお，数値は“0”がレベル 0 に，“7”がレベル 7 に対応しています。

備考 1　割り込み要因番号 *eintno* には，「(例外コード - 0x80) / 0x10」で算出される値を指定します。

備考 2　V850ES/V850E1/V850E2 コアで動作させる場合，本システム・コールを発行しても，期待した割り込み制御レジスタが操作されない場合があります。RX850 Pro では，割り込み要因番号から割り込み制御レジスタのアドレスを算出しています。しかし，V850ES/V850E1/V850E2 コアの場合，割り込み要因番号と割り込み制御レジスタの並びが，他の V850 マイクロコントローラと異なっているため，正しいレジスタ・アドレスが得られなくなっています。そのため，V850ES/V850E1/V850E2 コアにおける本システム・コールの使用を制御

します。割り込み制御レジスタをアプリケーションから操作したい場合は，本システム・コールを使わず，直接レジスタを操作してください。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_PAR	-33	パラメータの指定が不正である - 割り込み要因番号の指定が不正（*eintno* ＜ 0，最大割り込み要因番号 ＜ *eintno*）である - 割り込み要求フラグの指定が不正（*eintno* ＜ 0，最大割り込み要因番号 ＜ *eintno*）である - 割り込み要求フラグとして（ICR_CLRMSK | ICR_SETMSK）が指定されている

ref_icr

refer interrupt control register status（-68）

タスク／非タスク

【概要】

割り込み制御レジスタの内容を獲得する。

【C 言語形式】

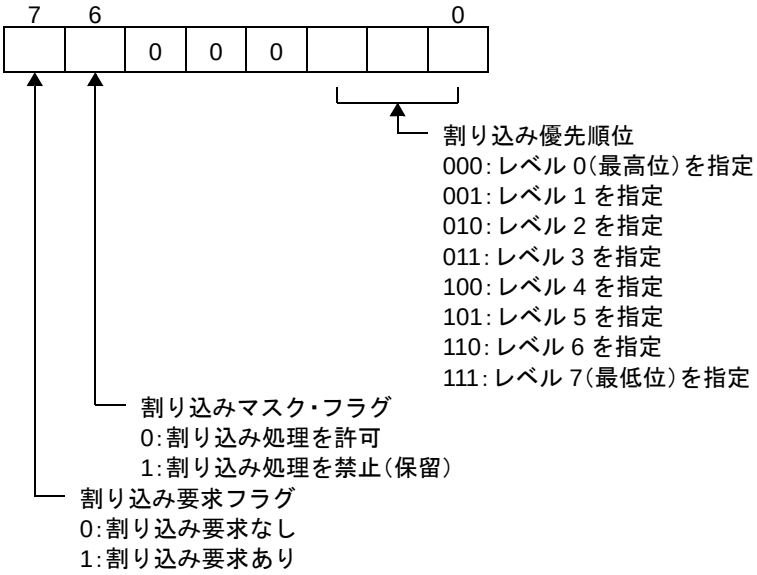
```
#include    <stdrx85p.h>
ER          ercd = ref_icr ( UB *p_regptn, UINT eintno );
```

【パラメータ】

入出力	パラメータ	説明
出	UB　*p_regptn;	割り込み制御レジスタの内容を格納する領域のアドレス
入	UINT　eintno;	割り込み要因番号

【説明】

eintno で指定された割り込み制御レジスタの内容を *p_regptn* で指定される領域に格納します。
次に，獲得した割り込み制御レジスタの内容を示します。



備考 1　割り込み要因番号 *eintno* には，「(例外コード - 0x80) / 0x10」で算出される値を指定します。

備考 2　V850ES/V850E1/V850E2 コアで動作させる場合，本システム・コールを発行しても，期待した割り込み制御レジスタが操作されない場合があります。RX850 Pro では，割り込み要因番号から割り込み制御レジスタのアドレスを算出しています。しかし，V850ES/V850E1/V850E2 コアの場合，割り込み要因番号と割り込み制御ジスタの並びが，他の V850 マイクロコントローラと異なっているため，正しいレジスタ・アドレスが得られなくなっています。そのため，V850ES/V850E1/V850E2 コアにおける本システム・コールの使用を制御します。割り込み制御レジスタをアプリケーションから操作したい場合は，本システム・コールを使わず，直接レジスタを操作してください。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_PAR	-33	パラメータの指定が不正である - 割り込み制御レジスタの内容を格納する領域のアドレスが不正（*p_regptn* = 0）である - 割り込み要因番号の指定が不正（*eintno* ＜ 0，最大割り込み要因番号 ＜ *eintno*）である

12.8.5　メモリ・プール管理機能

ここでは，メモリ・ブロックの割り当てを行うシステム・コールのグループ（メモリ・プール管理機能）について説明します。
表 12 － 9 に，メモリ・プール管理機能の一覧を示します。

表 12 － 9　メモリ・プール管理機能

システム・コール	機能
cre_mpl	メモリ・プールを生成する
del_mpl	メモリ・プールを削除する
get_blk	メモリ・ブロックを獲得する
pget_blk	メモリ・ブロックを獲得する（ポーリング）
tget_blk	メモリ・ブロックを獲得する（タイムアウトあり）
rel_blk	メモリ・ブロックを返却する
ref_mpl	メモリ・プール情報を獲得する
vget_pid	メモリ・プールの ID 番号を獲得する

cre_mpl

create variable-size memory pool（-137）

タスク

【概要】

メモリ・プールを生成する。

【C 言語形式】

- ID 番号を指定する場合

```
#include    <stdrx85p.h>
ER          ercd = cre_mpl ( ID mplid, T_CMPL *pk_cmpl );
```

- ID 番号を指定しない場合

```
#include    <stdrx85p.h>
ER          ercd = cre_mpl ( ID_AUTO, T_CMPL *pk_cmpl, ID *p_mplid );
```

【パラメータ】

入出力	パラメータ	説明
入	ID　*mplid*;	メモリ・プールの ID 番号
入	T_CMPL　**pk_cmpl*;	メモリ・プール生成情報を格納したパケットの先頭アドレス
出	ID　**p_mplid*;	ID 番号を格納する領域のアドレス

【メモリ・プール生成情報 T_CMPL の構造】

```
typedef struct  t_cmpl {
        VP      exinf;          /* 拡張情報 */
        ATR     mplatr;         /* メモリ・プールの属性 */
        INT     mplsz;          /* メモリ・プールのサイズ */
        ID      keyid;          /* メモリ・プールのキー ID 番号 */
} T_CMPL;
```

【説明】

RX850 Pro では，メモリ・プールの生成において「ID 番号を指定して生成する」，「ID 番号を指定しないで生成する」の 2 種類のインタフェースを用意しています。

- ID 番号を指定する場合
  pk_cmpl で指定された情報を基に，*mplid* で指定された ID 番号を持つメモリ・プールを生成します。

- ID 番号を指定しない場合
  pk_cmpl で指定された情報を基に，メモリ・プールを生成します。
  ID番号の割り付けはRX850 Proにより行われ，割り付けられたID番号は，*p_mplid* で指定される領域に格納されます。

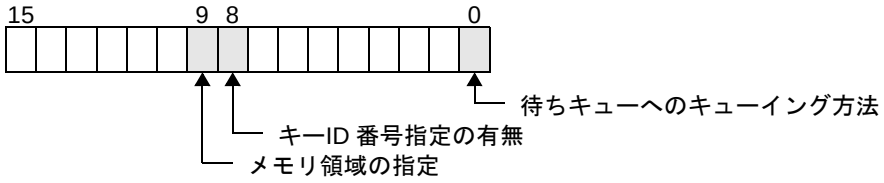
次に，メモリ・プール生成情報の詳細を示します。

exinf …　拡張情報
exinf は対象メモリ・プールに関する“ユーザ独自の情報”を格納するための領域で，ユーザが自由に利用できます。
exinf に設定された情報は，処理プログラム（タスク，非タスク）から ref_mpl を発行することにより，ダイナミックに獲得できます。

mplatr …　メモリ・プールの属性
ビット 0 …　待ちキューへのキューイング方法
TA_TPRI（0）：　優先度順
TA_TFIFO（1）：　FIFO 順

ビット 8 …　キー ID 番号指定の有無
TA_KEYID（1）：　キー ID 番号を指定

ビット 9 …　メモリ領域の指定
TA_UPOL0（0）：　ユーザ・メモリ領域 0 番からメモリ・プール領域を確保
TA_UPOL1（1）：　ユーザ・メモリ領域 1 番からメモリ・プール領域を確保



mplsz …　メモリ・プールのサイズ（単位：バイト）

keyid …　メモリ・プールのキー ID 番号

備考　　mplatr のビット 8 の値が 1（TA_KEYID）以外の場合，keyid の内容は意味を持ちません。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOMEM	-10	メモリ・プール管理ブロック，またはメモリ・プール領域が確保できない
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_RSATR	-24	属性 mplatr の指定が不正である
E_PAR	-33	パラメータの指定が不正である - メモリ・プール生成情報を格納したパケットの先頭アドレスが不正（*pk_cmpl* = 0）である - サイズの指定が不正（mplsz ≦ 0）である - キー ID 番号の指定が不正（keyid = 0）である（TA_KEYID 属性指定時） - ID 番号を格納する領域のアドレスが不正（*p_mplid* = 0）である（ID 番号を指定しないで生成する場合）
E_ID	-35	ID 番号の指定が不正（最大メモリ・プール生成数 < *mplid*）である
*E_OBJ	-63	指定した ID 番号を持つメモリ・プールがすでに生成されている
E_OACV	-66	アクセス権のない ID 番号（*mplid* ≦ 0）を指定した
E_CTX	-69	非タスクから本システム・コールを発行した

del_mpl

delete variable-size memory pool（-138）

タスク

【概要】

メモリ・プールを削除する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = del_mpl ( ID mplid );
```

【パラメータ】

入出力	パラメータ	説明
入	ID　　*mplid*;	メモリ・プールの ID 番号

【説明】

mplid で指定されたメモリ・プールを削除します。

これにより，対象メモリ・プールは RX850 Pro の管理下から除外されます。

なお，本システム・コールの発行により wait 状態（メモリ・ブロック待ち状態）を解除されたタスクは，wait 状態へと遷移するきっかけとなったシステム・コール（get_blk，tget_blk）の戻り値として E_DLT が返されます。

また，対象メモリ・プールが管理していたメモリ・ブロックをタスクが獲得していた場合に，本システム・コールを発行すると，メモリ・ブロックについても RX850 Pro の管理下から除外されます。したがって，獲得していたメモリ・ブロックの内容については不定となります。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_ID	-35	ID 番号の指定が不正（最大メモリ・プール生成数 ＜ *mplid*）である
*E_NOEXS	-52	対象メモリ・プールが存在していない
E_OACV	-66	アクセス権のない ID 番号（*mplid* ≦ 0）を指定した
E_CTX	-69	非タスクから本システム・コールを発行した

get_blk

get variable-size memory block（-141）

タスク

【 概要 】

メモリ・ブロックを獲得する。

【 C 言語形式 】

```
#include    <stdrx85p.h>
ER          ercd = get_blk ( VP *p_blk, ID mplid, INT blksz );
```

【 パラメータ 】

入出力	パラメータ	説明
出	VP *p_blk;	メモリ・ブロックの先頭アドレスを格納する領域のアドレス
入	ID mplid;	メモリ・プールの ID 番号
入	INT blksz;	メモリ・ブロックのサイズ（単位：バイト）

【 説明 】

mplid で指定されたメモリ・プールから *blksz* で指定されたサイズのメモリ・ブロックを獲得し，その先頭アドレスを *p_blk* で指定される領域に格納します。

ただし，本システム・コールを発行した際，対象メモリ・プールからメモリ・ブロックを獲得できなかった（要求したサイズの空き領域が存在しなかった）場合には，自タスクを対象メモリ・プールの待ちキューにキューイングしたあと，run 状態から wait 状態（メモリ・ブロック待ち状態）へと遷移させます。

なお，メモリ・ブロック待ち状態は rel_blk の発行により要求したサイズを満足するようなメモリ・ブロックが返却されるか，または del_mpl，rel_wai が発行されると解除され，自タスクは ready 状態へと遷移します。

備考 1　RX850 Pro では，メモリ・ブロックを獲得する際，メモリ・クリアを行いません。したがって，獲得したメモリ・ブロックの内容は不定となります。

備考 2　自タスクを対象メモリ・プールの待ちキューにキューイングする際は，対象メモリ・プール生成時（コンフィギュレーション時，または cre_mpl 発行時）に指定された順（FIFO 順，または優先度順）に行われます。

【 戻り値 】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_PAR	-33	パラメータの指定が不正である - メモリ・ブロックの先頭アドレスを格納する領域のアドレスが不正（*p_blk* = 0）である - メモリ・ブロック・サイズの指定が不正（*blksz* ≦ 0）である
E_ID	-35	ID 番号の指定が不正（最大メモリ・プール生成数 ＜ *mplid*）である
*E_NOEXS	-52	対象メモリ・プールが存在していない
E_OACV	-66	アクセス権のない ID 番号（*mplid* ≦ 0）を指定した

マクロ	数値	意味
E_CTX	-69	コンテキスト・エラー - 非タスクから本システム・コールを発行した - ディスパッチ禁止状態から本システム・コールを発行した
*E_DLT	-81	del_mpl により対象メモリ・プールが削除された
*E_RLWAI	-86	rel_wai によりメモリ・ブロック待ち状態が強制的に解除された

pget_blk

poll and get variable-size memory block（-104）

タスク／非タスク

【概要】

メモリ・ブロックを獲得する（ポーリング）。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = pget_blk ( VP *p_blk, ID mplid, INT blksz );
```

【パラメータ】

入出力	パラメータ	説明
出	VP *p_blk;	メモリ・ブロックの先頭アドレスを格納する領域のアドレス
入	ID mplid;	メモリ・プールの ID 番号
入	INT blksz;	メモリ・ブロックのサイズ（単位：バイト）

【説明】

mplid で指定されたメモリ・プールから，*blksz* で指定されたサイズのメモリ・ブロックを獲得し，その先頭アドレスを *p_blk* で指定される領域に格納します。

ただし，本システム・コールを発行した際，対象メモリ・プールからメモリ・ブロックを獲得できなかった（要求したサイズの空き領域が存在しなかった）場合には，戻り値として E_TMOUT が返されます。

備考　RX850 Pro では，メモリ・ブロックを獲得する際，メモリ・クリアを行いません。したがって，獲得したメモリ・ブロックの内容は不定となります。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_PAR	-33	パラメータの指定が不正である - メモリ・ブロックの先頭アドレスを格納する領域のアドレスが不正（*p_blk* = 0）である - メモリ・ブロック・サイズの指定が不正（*blksz* ≦ 0）である
E_ID	-35	ID 番号の指定が不正（最大メモリ・プール生成数 ＜ *mplid*）である
*E_NOEXS	-52	対象メモリ・プールが存在していない
E_OACV	-66	アクセス権のない ID 番号（*mplid* ≦ 0）を指定した
*E_TMOUT	-85	対象メモリ・プールに空き領域が存在しない

tget_blk

get variable-size memory block with timeout（-168）

タスク

【概要】

メモリ・ブロックを獲得する（タイムアウトあり）。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = tget_blk ( VP *p_blk, ID mplid, INT blksz, TMO tmout );
```

【パラメータ】

入出力	パラメータ	説明
出	VP *p_blk;	メモリ・ブロックの先頭アドレスを格納する領域のアドレス
入	ID mplid;	メモリ・プールの ID 番号
入	INT blksz;	メモリ・ブロックのサイズ（単位：バイト）
入	TMO tmout;	待ち時間（単位：ms） TMO_POL（0）：　即時リターン TMO_FEVR（-1）：　永久待ち 数値：　待ち時間

【説明】

mplid で指定されたメモリ・プールから，*blksz* で指定されたサイズのメモリ・ブロックを獲得し，その先頭アドレスを *p_blk* で指定される領域に格納します。

ただし，本システム・コールを発行した際，対象メモリ・プールからメモリ・ブロックを獲得できなかった（要求したサイズの空き領域が存在しなかった）場合には，自タスクを対象メモリ・プールの待ちキューにキューイングしたあと，run 状態から wait 状態（メモリ・ブロック待ち状態）へと遷移させます。

なお，メモリ・ブロック待ち状態は *tmout* で指定された待ち時間が経過するか，rel_blk の発行により要求したサイズを満足するようなメモリ・ブロックが返却されるか，または del_mpl，rel_wai が発行されると解除され，自タスクは ready 状態へと遷移します。

備考 1　RX850 Pro では，メモリ・ブロックを獲得する際，メモリ・クリアを行いません。したがって，獲得したメモリ・ブロックの内容は不定となります。

備考 2　自タスクを対象メモリ・プールの待ちキューにキューイングする際は，対象メモリ・プール生成時（コンフィギュレーション時，または cre_mpl 発行時）に指定された順（FIFO 順，または優先度順）に行われます。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_PAR	-33	パラメータの指定が不正である - メモリ・ブロックの先頭アドレスを格納する領域のアドレスが不正（*p_blk* = 0）である - メモリ・ブロック・サイズの指定が不正（*blksz* ≦ 0）である

マクロ	数値	意味
E_ID	-35	ID 番号の指定が不正（最大メモリ・プール生成数 ＜ *mplid*）である
*E_NOEXS	-52	対象メモリ・プールが存在していない
E_OACV	-66	アクセス権のない ID 番号（*mplid* ≦ 0）を指定した
E_CTX	-69	コンテキスト・エラー - 非タスクから本システム・コールを発行した - ディスパッチ禁止状態から本システム・コールを発行した
*E_DLT	-81	del_mpl によりメモリ・プールが削除された
*E_TMOUT	-85	待ち時間が経過した
*E_RLWAI	-86	rel_wai によりメモリ・ブロック待ち状態が強制的に解除された

rel_blk

release variable-size memory block（-143）

タスク／非タスク

【概要】

メモリ・ブロックを返却する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = rel_blk ( ID mplid, VP blk );
```

【パラメータ】

入出力	パラメータ	説明
入	ID　*mplid*;	メモリ・プールの ID 番号
入	VP　*blk*;	メモリ・ブロックの先頭アドレス

【説明】

blk で指定されたメモリ・ブロックを *mplid* で指定されたメモリ・プールに返却します。

本システム・コールを発行した際，返却するメモリ・ブロックのサイズが対象メモリ・プールの待ちキューにキューイングされているタスク（待ちキューの先頭タスク）が要求したサイズを満足するような場合には，該当タスク（待ちキューの先頭タスク）にメモリ・ブロックを渡します。

これにより，該当タスクは待ちキューから外れ，wait 状態（メモリ・ブロック待ち状態）から ready 状態へ，または wait_suspend 状態から suspend 状態へと遷移します。

備考 1　RX850 Pro では，メモリ・ブロックを返却するときにメモリ・クリアを行いません。したがって，返却したメモリ・ブロックの内容は不定です。

備考 2　RX850 Pro には 2 つの仕様の本システム・コールがあります。

（1）本システム・コールでメモリ・ブロックを返却する際，メモリ・ブロックの先頭 4 バイトが 0 でなかった場合に，メモリ・ブロックを返却せずに戻り値 E_OBJ で終了する。

（2）本システム・コールを発行すると，メモリ・ブロックの先頭 4 バイトが 0 でなくてもメモリ・ブロックを返却する（戻り値 E_OK）。

（1）はメモリ・ブロックがメールボックスのメッセージ領域として使用されていた場合を考慮したもので，これまでの RX850 Pro に搭載されていた本システム・コールと同じ仕様のものです。

メモリ・ブロックがメールボックスのメッセージ領域として使用されていた場合，先頭 4 バイトがメッセージの待ちキューのリンク領域になります。つまりメッセージが，メールボックスにキューイングされているときに本システム・コールを発行し，必ずメモリ・ブロックが返却する仕様にすると，キューにつながれていたメッセージ領域が返却されてしまうことになります。これを防ぐため，リンク領域である先頭 4 バイトが 0 でなければ，メッセージ領域として使用されているメモリ・ブロックと判断し，メモリ・ブロックを返却せずに戻り値 E_OBJ で終了します。

この仕様の本システム・コールを使用してメモリ・ブロックを返却するときは，必ず先頭 4 バイトを 0 クリアする必要があります。

これらの仕様の本システム・コールは，別々のライブラリに搭載されていますので，どちらか一方の本システム・コールを使用することになります。使用する方のライブラリをリンクしてください。

（1）メモリ・ブロックの先頭 4 バイトを 0 でクリアする必要のある本システム・コールが搭載されたライブラリ
→ librxp.a

（2）メモリ・ブロックの先頭 4 バイトを 0 でクリアしなくてもよい本システム・コールが搭載されたライブラリ
→ librxpm.a

備考 3　メモリ・ブロックを返却するメモリ・プールは，get_blk，pget_blk，tget_blk　を発行した際に指定したメモリ・プールと同じにしてください。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
*E_PAR	-33	パラメータの指定が不正である - メモリ・ブロックの先頭アドレスの指定が不正（*blk* = 0）である - 獲得時に指定したメモリ・プールと本システム・コール発行時に指定したメモリ・プールが異なっている
E_ID	-35	ID 番号の指定が不正（最大メモリ・プール生成数 < *mplid*）である
*E_NOEXS	-52	対象メモリ・プールが存在していない
*E_OBJ	-63	指定したメモリ・ブロックの状態が不適当である - 返却するメモリ・ブロックの先頭 4 バイトに 0x0 以外の値が入っている - メッセージ領域として使用されているメモリ・ブロックを返却しようとした
E_OACV	-66	アクセス権のない ID 番号（*mplid* ≦ 0）を指定した

ref_mpl

refer variable-size memory pool status（-140）

タスク／非タスク

【概要】

メモリ・プール情報を獲得する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = ref_mpl ( T_RMPL *pk_rmpl, ID mplid );
```

【パラメータ】

入出力	パラメータ	説明
出	T_RMPL *pk_rmpl;	メモリ・プール情報を格納するパケットの先頭アドレス
入	ID mplid;	メモリ・プールの ID 番号

【メモリ・プール情報 T_RMPL の構造】

```
typedef struct  t_rmpl {
        VP      exinf;          /* 拡張情報 */
        BOOL_ID wtsk;           /* 待ちタスクの有無 */
        INT     frsz;           /* 空き領域の合計サイズ */
        INT     maxsz;          /* 獲得可能な最大メモリ・ブロック・サイズ */
        ID      keyid;          /* キー ID 番号 */
} T_RMPL;
```

【説明】

mplid で指定されたメモリ・プールのメモリ・プール情報（拡張情報，待ちタスクの有無など）を *pk_rmpl* で指定されるパケットに格納します。
次に，メモリ・プール情報の詳細を示します。

exinf … 拡張情報

wtsk … 待ちタスクの有無
FALSE（0）： 待ちタスクなし
数値： 待ちキューの先頭タスクの ID 番号

frsz … 空き領域の合計サイズ（単位：バイト）

maxsz … 獲得可能な最大メモリ・ブロック・サイズ（単位：バイト）

keyid … キー ID 番号
FALSE（0）： 生成時にキー ID 番号が指定されていない
数値： キー ID 番号

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない

マクロ	数値	意味
E_PAR	-33	メモリ・プール情報を格納するパケットの先頭アドレスが不正（*pk_rmpl* = 0）である
E_ID	-35	ID 番号指定が不正（最大メモリ・プール生成数 < *mplid*）である
*E_NOEXS	-52	対象メモリ・プールが存在していない
E_OACV	-66	アクセス権のない ID 番号（*mplid* ≦ 0）を指定した

vget_pid

get variable-size memory pool identifier（-242）

タスク／非タスク

【概要】

メモリ・プールの ID 番号を獲得する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = vget_pid ( ID *p_mplid, ID keyid );
```

【パラメータ】

入出力	パラメータ	説明
出	ID　*p_mplid;	ID 番号を格納する領域のアドレス
入	ID　keyid;	メモリ・プールのキー ID 番号

【説明】

keyid で指定されたメモリ・プールールの ID 番号を *p_mplid* で指定される領域に格納します。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_PAR	-33	パラメータの指定が不正である - ID 番号を格納する領域のアドレスが不正（*p_mplid* = 0）である - キー ID 番号の指定が不正（*keyid* = 0）である
*E_NOEXS	-52	対象メモリ・プールが存在していない

12.8.6　時間管理機能

ここでは，時間に依存した処理を行うシステム・コールのグループ（時間管理機能）について説明します。
表 12 － 10 に，時間管理機能の一覧を示します。

表 12 － 10　時間管理機能

システム・コール	機能
set_tim	システム・クロックに時刻を設定する
get_tim	システム・クロックの時刻を獲得する
dly_tsk	自タスクを時間経過待ち状態へ移行させる
def_cyc	周期起動ハンドラを登録／登録解除する
act_cyc	周期起動ハンドラの活性状態を制御する
ref_cyc	周期起動ハンドラ情報を獲得する

set_tim

set time（-83）

タスク／非タスク

【概要】

システム・クロックに時刻を設定する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = set_tim ( SYSTIME *pk_tim );
```

【パラメータ】

入出力	パラメータ	説明
入	SYSTIME *pk_tim;	時刻を格納したパケットの先頭アドレス

【時刻 SYSTIME の構造】

```
typedef struct  t_systime {
        UW      ltime;          /* 時刻（下位 32 ビット）*/
        H       utime;          /* 時刻（上位 16 ビット）*/
} SYSTIME;
```

【説明】

システム・クロックに *pk_tim* で指定された時刻を設定します。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_PAR	-33	時刻を格納したパケットの先頭アドレスが不正（*pk_tim* = 0）である

get_tim

get time（-84）

タスク／非タスク

【 概要 】

システム・クロックの時刻を獲得する。

【 C 言語形式 】

```
#include    <stdrx85p.h>
ER          ercd = get_tim ( SYSTIME *pk_tim );
```

【 パラメータ 】

入出力	パラメータ	説明
出	SYSTIME *pk_tim;	時刻を格納するパケットの先頭アドレス

【 時刻 SYSTIME の構造 】

```
typedef struct  t_systime {
        UW      ltime;          /* 時刻（下位 32 ビット）*/
        H       utime;          /* 時刻（上位 16 ビット）*/
} SYSTIME;
```

【 説明 】

システム・クロックの現時刻を *pk_tim* で指定されるパケットに格納します。

【 戻り値 】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_PAR	-33	時刻を格納するパケットの先頭アドレスが不正（*pk_tim* = 0）である

dly_tsk

delay task（-85）

タスク

【概要】

自タスクを時間経過待ち状態へ移行させる。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = dly_tsk ( DLYTIME dlytim );
```

【パラメータ】

入出力	パラメータ	説明
入	DLYTIME *dlytim*;	遅延時間（単位：ms）

【説明】

自タスクを *dlytim* で指定された遅延時間だけ，run 状態から wait 状態（時間経過待ち状態）へと遷移させます。

なお，時間経過待ち状態は *dlytim* で指定された遅延時間が経過するか，または rel_wai が発行されると解除され，自タスクは ready 状態へ遷移します。

備考　　時間経過待ち状態は，wup_tsk では解除されません。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_PAR	-33	遅延時間の指定が不正（*dlytim* ＜ 0）である
E_CTX	-69	コンテキスト・エラー - 非タスクから本システム・コールを発行した - ディスパッチ禁止状態から本システム・コールを発行した
*E_RLWAI	-86	rel_wai により時間経過待ち状態が強制的に解除された

def_cyc

define cyclic handler（-90）

タスク／非タスク

【 概要 】

周期起動ハンドラを登録／登録解除する。

【 C 言語形式 】

```
#include    <stdrx85p.h>
ER          ercd = def_cyc ( HNO cycno, T_DCYC *pk_dcyc );
```

【 パラメータ 】

入出力	パラメータ	説明
入	HNNO　*cycno*;	周期起動ハンドラの指定番号
入	T_DCYC　**pk_dcyc*;	周期起動ハンドラ登録情報を格納したパケットの先頭アドレス

【 周期起動ハンドラ登録情報 T_DCYC の構造 】

```
typedef struct  t_dcyc {
        VP      exinf;          /* 拡張情報 */
        ATR     cycatr;         /* 周期起動ハンドラの属性 */
        FP      cychdr;         /* 周期起動ハンドラの起動アドレス */
        UINT    cycact;         /* 周期起動ハンドラの初期活性状態 */
        CYCTIME cyctim;         /* 周期起動ハンドラの起動時間間隔 */
        VP      gp;             /* 周期起動ハンドラの固有 gp レジスタ値 */
        VP      tp;             /* 周期起動ハンドラの固有 tp レジスタ値 */
} T_DCYC;
```

【 説明 】

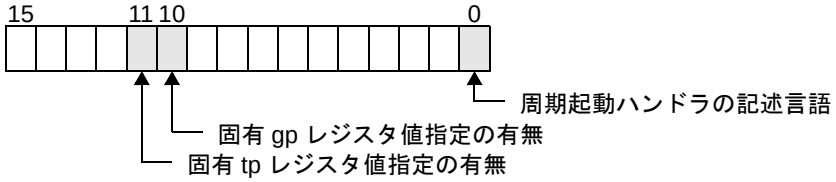
pk_dcyc で指定された情報を基に，*cycno* で指定された指定番号を持つ周期起動ハンドラを登録します。
次に，周期起動ハンドラ登録情報の詳細を示します。

exinf …　拡張情報
exinf は対象周期起動ハンドラに関する“ユーザ独自の情報”を格納するための領域で，ユーザが自由に利用できます。
exinf に設定された情報は，処理プログラム（タスク，非タスク）から ref_cyc を発行することにより，ダイナミックに獲得できます。

cycatr …　周期起動ハンドラの属性
ビット 0 …　周期起動ハンドラの記述言語
TA_ASM（0）：　アセンブリ言語
TA_HLNG（1）：　C 言語

ビット 10 …　固有 gp レジスタ値指定の有無
TA_DPID（1）：　固有 gp レジスタ値を指定

ビット 11 …　固有 tp レジスタ値指定の有無
TA_DPIC（1）：　固有 tp レジスタ値を指定



cychdr … 周期起動ハンドラの起動アドレス

cycact … 周期起動ハンドラの初期活性状態
TCY_OFF（0）： 初期活性状態は OFF 状態
TCY_ON（1）： 初期活性状態は ON 状態

cyctim … 周期起動ハンドラの起動時間間隔（単位：基本クロック周期）

gp … 周期起動ハンドラの固有 gp レジスタ値

tp … 周期起動ハンドラの固有 tp レジスタ値

本システム・コールを発行した際，対象指定番号に対応した周期起動ハンドラがすでに登録されていた場合には，エラーとしては扱わず，本システム・コールで指定された周期起動ハンドラを新規に登録します。
また，本システム・コールを発行した際，*pk_dcyc* で指定される領域に NADR（-1）を設定した場合，*cycno* で指定された周期起動ハンドラの登録が解除されます。

備考 1　cycatr のビット 10 の値が 1（TA_DPID）以外の場合，gp の内容は意味を持ちません。

備考 2　cycatr のビット 11 の値が 1（TA_DPIC）以外の場合，tp の内容は意味を持ちません。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_RSATR	-24	属性 cycatr の指定が不正である
E_PAR	-33	パラメータの指定が不正である - 指定番号の指定が不正（*cycno* ≦ 0，最大周期起動ハンドラ登録数 ＜ *cycno*）である - 周期起動ハンドラ登録情報を格納したパケットの先頭アドレスの指定が不正（*pk_dcyc* = 0）である - 起動アドレスの指定が不正（cychdr = 0）である - 初期活性状態 cycact の指定が不正である - 起動時間間隔の指定が不正（cyctim ≦ 0）である

act_cyc

activate cyclic handler（-94）

タスク／非タスク

【概要】

周期起動ハンドラの活性状態を制御する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = act_cyc ( HNO cycno, UINT cycact );
```

【パラメータ】

入出力	パラメータ	説明
入	HNO　*cycno*;	周期起動ハンドラの指定番号
入	UINT　*cycact*;	活性状態／周期カウンタの指定 TCY_OFF（0）：活性状態を OFF 状態へ変更 TCY_ON（1）：活性状態を ON 状態へ変更 TCY_INI（2）：周期カウンタを初期化

【説明】

cycno で指定され周期起動ハンドラの活性状態を *cycact* で指定された状態に変更します。
次に，*cycact* の指定形式を示します。

- *cycact* = TCY_OFF
  対象周期起動ハンドラの活性状態を OFF 状態に変更します。
  これにより，起動時間に達しても，対象周期起動ハンドラは起動されません。

  備考　RX850 Proでは, 周期起動ハンドラの活性状態がOFF状態でも, 周期カウンタのカウント処理は行われます。

- *cycact* = TCY_ON
  対象周期起動ハンドラの活性状態を ON 状態に変更します。
  これにより，起動時間に達した際には，対象周期起動ハンドラが起動されます。

- *cycact* = TCY_INI
  対象周期起動ハンドラの周期カウンタを初期化します。

- *cycact* = (TCY_ON | TCY_INI)
  対象周期起動ハンドラの活性状態を ON 状態に変更したあと，周期カウンタを初期化します。
  これにより，起動時間に達した際には，対象周期起動ハンドラが起動されます。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない

マクロ	数値	意味
E_PAR	-33	パラメータの指定が不正である - 周期起動ハンドラの指定番号が不正（*cycno* ≦ 0，最大周期起動ハンドラ登録数 ＜ *cycno*）である - 活性状態／周期カウンタ *cycact* の指定が不正である
*E_NOEXS	-52	対象周期起動ハンドラが登録されていない

refer cyclic handler status（-92）

ref_cyc

タスク／非タスク

【概要】

周期起動ハンドラ情報を獲得する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = ref_cyc ( T_RCYC *pk_rcyc, HNO cycno );
```

【パラメータ】

入出力	パラメータ	説明
出	T_RCYC *pk_rcyc;	周期起動ハンドラ情報を格納するパケットの先頭アドレス
入	HNO cycno;	周期起動ハンドラの指定番号

【周期起動ハンドラ情報 T_RCYC の構造】

```
typedef struct  t_rcyc {
        VP      exinf;          /* 拡張情報 */
        CYCTIME lfttim;         /* 残り時間 */
        UINT    cycact;         /* 現在の活性状態 */
} T_RCYC;
```

【説明】

cycno で指定された周期起動ハンドラの周期起動ハンドラ情報（拡張情報，残り時間など）を *pk_rcyc* で指定されるパケットに格納します。
次に，周期起動ハンドラ情報の詳細を示します。

exinf …　拡張情報

lfttim …　次に周期起動ハンドラを起動するまでの残り時間（単位：基本クロック周期）

cycact …　現在の活性状態
TCY_OFF（0）：　活性状態は OFF 状態
TCY_ON（1）：　活性状態は ON 状態

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_PAR	-33	パラメータの指定が不正である - 周期起動ハンドラ情報を格納するパケットの先頭アドレスが不正（*pk_rcyc* = 0）である - 周期起動ハンドラの指定番号が不正（*cycno* ≦ 0，最大周期起動ハンドラ登録数 < *cycno*）である

マクロ	数値	意味
*E_NOEXS	-52	対象周期起動ハンドラが登録されていない

12.8.7　システム管理機能

ここでは，システムに依存した処理を行うシステム・コールのグループ（システム管理機能）について説明します。
表 12 － 11 に，システム管理機能の一覧を示します。

表 12 － 11　システム管理機能

システム・コール	機能
get_ver	RX850 Pro のバージョン情報を獲得する
ref_sys	システム情報を獲得する
def_svc	拡張 SVC ハンドラを登録／登録解除する
viss_svc	拡張 SVC ハンドラを呼び出す

get version information（-16）

get_ver

タスク／非タスク

【 概要 】

RX850 Pro のバージョン情報を獲得する。

【 C 言語形式 】

```
#include    <stdrx85p.h>
ER          ercd = get_ver ( T_VER *pk_ver );
```

【 パラメータ 】

入出力	パラメータ	説明
出	T_VER　*pk_ver;	バージョン情報を格納するパケットの先頭アドレス

【 バージョン情報 T_VER の構造 】

```
typedef struct  t_ver {
        UH      maker;          /*OS メーカ */
        UH      id;             /*OS 形式番号 */
        UH      spver;          /* 仕様書バージョン番号 */
        UH      prver;          /*OS 製品バージョン番号 */
        UH      prno[4];        /* 製品番号／製品管理情報 */
        UH      cpu;            /*CPU 情報 */
        UH      var;            /* バリエーション記述子 */
} T_VER;
```

【 説明 】

RX850 Pro のバージョン情報（OS メーカ，OS 形式番号など）を *pk_ver* で指定されるパケットに格納します。
次に，バージョン情報の詳細を示します。

maker …　OS メーカ
　　0x0117：　NEC エレクトロニクス

id …　OS 形式番号
　　0x0000：　未使用

spver …　仕様書バージョン番号
　　0x5302：　μITRON3.0 Ver.3.02

prver …　OS 製品バージョン番号
　　0x032*x*：　RX850 Pro Ver.3.2*x*

prno[4] …　製品番号／製品管理情報
　　不定：　出荷製品のシリアル番号（出荷製品ごとに異なります）

cpu …　CPU 情報
　　0x0d33：　V850 コア
　　0x0d37：　V850ES/V850E1/V850E2 コア

var …　バリエーション記述子
　　0xc000：　μITRON レベル E，ファイル・サポートなし

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_PAR	-33	バージョン情報を格納するパケットの先頭アドレスが不正（*pk_ver* = 0）である

refer system status（-12）

ref_sys

タスク／非タスク

【概要】

システム情報を獲得する。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = ref_sys ( T_RSYS *pk_rsys );
```

【パラメータ】

入出力	パラメータ	説明
出	T_RSYS　*pk_rsys;	システム情報を格納するパケットの先頭アドレス

【システム情報 T_RSYS の構造】

```
typedef struct  t_rsys {
        INT     sysstat;        /* システムの状態 */
} T_RSYS;
```

【説明】

ダイナミックに変化するシステム情報（システムの状態）の現在値を *pk_rsys* で指定されるパケットに格納します。次に，システム情報の詳細を示します。

sysstat … システムの状態

TSS_TSK（0）:　タスクの処理を実行中（ディスパッチ処理は許可状態）
TSS_DDSP（1）:　タスクの処理を実行中（ディスパッチ処理は禁止状態）
TSS_LOC（3）:　タスクの処理を実行中（マスカブル割り込みの受け付けとディスパッチ処理は禁止状態）
TSS_INDP（4）:　非タスク（割り込みハンドラ，周期起動ハンドラなど）の処理を実行中

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
*E_PAR	-33	システム情報を格納するパケットの先頭アドレスが不正（*pk_rsys* = 0）である

def_svc

define supervisor call handler（-9）

タスク／非タスク

【 概要 】

拡張 SVC ハンドラを登録／登録解除する。

【 C 言語形式 】

```
#include    <stdrx85p.h>
ER          ercd = def_svc ( FN s_fncd, T_DSVC *pk_dsvc );
```

【 パラメータ 】

入出力	パラメータ	説明
入	FN　　*s_fncd*;	拡張 SVC ハンドラの拡張機能コード
入	T_DSVC　**pk_dsvc*;	拡張 SVC ハンドラ登録情報を格納したパケットの先頭アドレス

【 拡張 SVC ハンドラ登録情報 T_DSVC の構造 】

```
typedef struct  t_dsvc {
        ATR     svcatr;         /* 拡張 SVC ハンドラの属性 */
        FP      svchdr;         /* 拡張 SVC ハンドラの起動アドレス */
        VP      gp;             /* 拡張 SVC ハンドラの固有 gp レジスタ値 */
        VP      tp;             /* 拡張 SVC ハンドラの固有 tp レジスタ値 */
} T_DSVC;
```

【 説明 】

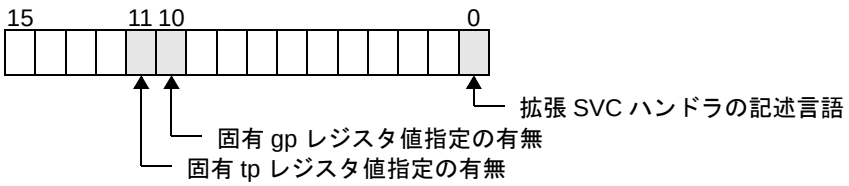
pk_dsvc で指定された情報を基に，*s_fncd* で指定された拡張機能コードを持つ拡張 SVC ハンドラを登録します。
次に，拡張 SVC ハンドラ登録情報の詳細を示します。

svcatr …　拡張 SVC ハンドラの属性

ビット 0 …　拡張 SVC ハンドラの記述言語
- TA_ASM（0）：　アセンブリ言語
- TA_HLNG（1）：　C 言語

ビット 10 …　固有 gp レジスタ値指定の有無
- TA_DPID（1）：　固有 gp レジスタ値を指定

ビット 11 …　固有 tp レジスタ値指定の有無
- TA_DPIC（1）：　固有 tp レジスタ値を指定



svchdr …　拡張 SVC ハンドラの起動アドレス

gp …　拡張 SVC ハンドラの固有 gp レジスタ値

tp …　　拡張 SVC ハンドラの固有 tp レジスタ値

本システム・コールを発行した際，すでに対象拡張機能コードに対応した拡張 SVC ハンドラが登録されていた場合には，エラーとしては扱わず，本システム・コールで指定された拡張 SVC ハンドラを新規に登録します。
また，本システム・コールを発行した際，*pk_dsvc* で指定される領域に NADR（－ 1）を設定した場合，*s_fncd* で指定された拡張 SVC ハンドラの登録が解除されます。

備考 1　svcatr のビット 10 の値が 1（TA_DPID）以外の場合，gp の内容は意味を持ちません。

備考 2　svcatr のビット 11 の値が 1（TA_DPIC）以外の場合，tp の内容は意味を持ちません。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない
E_RSATR	-24	属性 svcatr の指定が不正である
E_PAR	-33	パラメータの指定が不正である - 拡張機能コードの指定が不正（*s_fncd* ≦ 0，最大拡張 SVC ハンドラ登録数 ＜ *s_fncd*）である拡張 SVC ハンドラ登録情報を格納したパケットの先頭アドレスが不正（*pk_dsvc* = 0）である - 起動アドレスの指定が不正（svchdr = 0）である

issued supervisor call handler（-250）

viss_svc

タスク／非タスク

【概要】

拡張 SVC ハンドラを呼び出す。

【C 言語形式】

```
#include    <stdrx85p.h>
ER          ercd = viss_svc ( FN s_fncd, VW prm1, VW prm2, VW prm3 );
```

【パラメータ】

入出力	パラメータ	説明
入	FN　*s_fncd*;	拡張 SVC ハンドラの拡張機能コード
入	VW　*prm1*;	拡張 SVC ハンドラへの引き渡しパラメータ 1
入	VW　*prm2*;	拡張 SVC ハンドラへの引き渡しパラメータ 2
入	VW　*prm3*;	拡張 SVC ハンドラへの引き渡しパラメータ 3

【説明】

s_fncd で指定された拡張機能コードを持つ拡張 SVC ハンドラを呼び出します。

備考　本システム・コールを利用して拡張 SVC ハンドラを呼び出す場合には，拡張 SVC ハンドラ用インタフェース・ライブラリの記述が不要となります。

【戻り値】

マクロ	数値	意味
*E_OK	0	正常終了
*E_NOSPT	-17	本システム・コールが CF 定義されていない，または登録されていない拡張 SVC ハンドラを呼び出した
E_PAR	-33	拡張機能コードの指定が不正（*s_fncd* ≦ 0，最大拡張 SVC ハンドラ登録数＜ *s_fncd*）である
その他	－	拡張 SVC ハンドラからの戻り値

第 13 章　システム・コンフィギュレーション・ファイル

この章では，システム・コンフィギュレーション・ファイルと，その記述方法について説明します。

13.1　概　　要

RX850 Pro を使ったシステムを構築する場合，必要とする各種データ（システム情報や資源情報など）と使用するシステム・コールの種類を保持した情報が必要となります。この情報は，システム情報テーブル・ファイル，および，システム・コール・テーブル・ファイルと呼ばれます。

システム情報テーブル・ファイル，および，システム・コール・テーブル・ファイルはアセンブリ言語で記述されており，既定された形式のデータ羅列となっています。これらは，各種エディタで記述することは可能ですが，変更や追加が非常に難しいものであるため，記述にはかなりの労力を必要とします。

そこで，コンフィギュレータ CF850 Pro というアプリケーションを提供しています。

これは，システム・コンフィギュレーション・ファイルという，RX850 Pro のシステムや資源，使用するシステム・コールに関する情報を記述した，独自のフォーマットのファイルを，システム情報テーブル・ファイルとシステム・コール・テーブル・ファイルに変換するアプリケーションです。つまり，ユーザはシステム・コンフィギュレーション・ファイルを作成し，CF850 Pro によって，システム情報テーブル・ファイルとシステム・コール・テーブル・ファイルを得ることができます。

CF850 Pro は，システム・コンフィギュレーション・ファイルからシステム情報テーブル・ファイル，システム・コール・テーブル・ファイル，システム情報ヘッダ・ファイルの 3 つのファイルを出力します。システム情報ヘッダ・ファイルは，作成したタスク，セマフォ等，資源 ID として指定されたシンボル名と実際の ID 番号を，#define 命令で対応させる記述があるファイルです。

CF850 Pro の起動方法に関しては，「第 14 章　コンフィギュレータ CF850 Pro」を参照してください。

次に，システム・コンフィギュレーション・ファイルの記述方法について説明します。

13.2　表記方法

次に，システム・コンフィギュレーション・ファイルを記述する際の表記方法を示します。

1） 構成要素と文字コード

a） 文字コード
システム・コンフィギュレーション・ファイルは，ASCII コードで記述します。
なお，漢字コード（SJIS，EUC のみ）はコメント中にのみ使用できます。

b） 単語
システム・コンフィギュレーション・ファイルの中では，空白（スペース・コード，タブ・コード，改行コード）を含まない一連の文字列を単語と呼びます。以降の説明中の数値，シンボル名，キー・ワードはすべて単語の一種です。
CF850 Pro は単語内の大文字と小文字を区別します。したがって，“ABC”，“Abc”，“abc”はそれぞれ別の単語として扱われます。

c） 文
システム・コンフィギュレーション・ファイルの中では，一つ以上の空白で区切られた一連の単語を文と呼びます。文の区切りは改行で行います。
なお，システム・コンフィギュレーション・ファイルでは，行末に“¥”を記述すると，次の行との併合を意味します。ただし，“¥”の直前 1 文字は，“スペース”，または，“タブ”でなければなりません。

2） 数値
単語の中で，先頭が数字コードで始まる単語は数値として扱われます。先頭の数字コードにより，表 13 － 1 に示すように数値が分類されます。
なお，数値は指定のない限り 32 ビット幅（0x0 ～ 0xFFFFFFFF）の範囲で記述できます。

表 13 － 1　数値の種別

種別	先頭の数字コード	含まれる文字	例
8 進数	0	0 ～ 7	0123，0，056712

種別	先頭の数字コード	含まれる文字	例
10 進数	0 以外の数字	0 ～ 9	123，0，689525
16 進数	0x，または 0X	0 ～ 9，a ～ f（，または，A ～ F），x，X	0x12C，0X0，0xabcdef

3） シンボル名
名前とシンボル名は文脈によって区別します。シンボル名はユーザ・プログラム上の関数名や変数名を示します。ただし，シンボル名の先頭 1 文字は，“_”，または，“英字”でなければなりません。

備考　　シンボル名として指定可能な文字数は，255 文字以内に限られます。

4） コメント
システム・コンフィギュレーション・ファイルでは，“--”以降行末までがコメントとして扱われます。

5） キー・ワード
CF850 Pro では，次に示す文字列をキー・ワードとして予約しています。
したがって，これらの文字列をほかの用途には使用できません。

```
clkhdr，clktim，cyc，defstk，flg，flgsvc，ini，inthdr，intstk，intsvc，maxcyc，maxflg，
maxint，maxintfactor，maxmbx，maxmpl，maxpri，maxsem，maxsvc，maxtsk，mbx，mbxsvc，
mem，mpl，mplsvc，no_use，prtflg，prtmbx，prtmpl，prtsem，prttsk，RX850PRO，rxsers，
sct_def，sem，semsvc，ser_def，sit_def，SPOL0，SPOL1，svc，syssvc，TA_ASM，TA_DISINT，
TA_ENAINT，TA_HLNG，TA_MFIFO，TA_MPRI，TA_TFIFO，TA_TPRI，TA_WMUL，TA_WSGL，TCY_OFF，
TCY_ON，timsvc，tsk，tsksvc，TTS_DMT，TTS_RDY，UPOL0，UPOL1，V850，V850ES，V850E1，
V850E2
```

13.3　コンフィギュレーション情報

システム･コンフィギュレーション･ファイルに記述するコンフィギュレーション情報は，次の 3 種類に大別されます。

- リアルタイム OS 情報
  使用するリアルタイム OS に関するデータ

- SIT（System Information Table）情報
  RX850 Pro が動作するうえで必要となるデータ

- SCT（System Call Table）情報
  システムで使用する機能（システム・コール）であるか否かを選択したデータ

13.3.1　リアルタイム OS 情報

システム・コンフィギュレーション・ファイルに記述するリアルタイム OS 情報は，次に示す 1 項目です。

1） RX シリーズ情報
RX シリーズ情報として，次のデータを定義します。

- リアルタイム OS 名
- バージョン番号

13.3.2　SIT（System Information Table）情報

システム・コンフィギュレーション・ファイルに記述する SIT 情報は，次に示す 12 項目です。

1）システム情報
システム情報として，次のデータを定義します。

- プロセッサ種別
- 基本クロック周期
- タイマの割り込み要因番号
- デフォルト・スタック・サイズ
- 割り込みハンドラ用スタック情報
  割り込みハンドラ用スタックのサイズ
  割り込みハンドラ用スタックを割り付けるシステム・メモリ領域の種類
- ID 番号の保護範囲
  タスクの ID 番号保護範囲
  セマフォの ID 番号保護範囲
  イベントフラグの ID 番号保護範囲
  メールボックスの ID 番号保護範囲
  メモリ・プールの ID 番号保護範囲

2）システム最大値情報
システム最大値情報として，次のデータを定義します。

- タスクの優先度範囲
- 管理オブジェクトの最大生成数／最大登録数
  タスクの最大生成数
  セマフォの最大生成数
  イベントフラグの最大生成数
  メールボックスの最大生成数
  メモリ・プールの最大生成数
  周期起動ハンドラの最大登録数
  拡張 SVC ハンドラの最大登録数
  間接起動割り込みハンドラの最大登録数
  割り込み要因番号の最大値

3）システム・メモリ情報
個々のシステム･メモリ領域（System Memory Pool 0, System Memory Pool 1, User Memory Pool 0, User Memory Pool 1）に対して，次のデータを定義します。

- システム・メモリ領域の種類
- セクション名
- サイズ

4）タスク情報
個々のタスクに対して，次のデータを定義します。

- ID 番号
- 初期状態
- 起動コード
- 拡張情報
- 記述言語
- 起動アドレス
- 初期優先度
- 割り込み状態
- タスク用スタック情報
  タスク用スタックのサイズ
  タスク用スタックを割り付けるシステム・メモリ領域の種類
- 固有 gp レジスタ値
- 固有 tp レジスタ値
- キー ID 番号

5) セマフォ情報
個々のセマフォに対して，次のデータを定義します。

- ID 番号
- 拡張情報
- タスクのキューイング方式
- 初期資源数
- 最大資源数
- キー ID 番号

6) イベントフラグ情報
個々のイベントフラグに対して，次のデータを定義します。

- ID 番号
- 拡張情報
- 複数タスクの待ち許可／禁止
- 初期ビット・パターン
- キー ID 番号

7) メールボックス情報
個々のメールボックスに対して，次のデータを定義します。

- ID 番号
- 拡張情報
- タスクのキューイング方式
- メッセージのキューイング方式
- キー ID 番号

8) 間接起動割り込みハンドラ情報
個々の間接起動割り込みハンドラに対して，次のデータを定義します。

- 割り込み要因番号
- 記述言語
- 起動アドレス
- 固有 gp レジスタ値
- 固有 tp レジスタ値

9) メモリ・プール情報
個々のメモリ・プールに対して，次のデータを定義します。

- ID 番号
- 拡張情報
- タスクのキューイング方式
- メモリ・プール情報
  メモリ・プールのサイズ
  メモリ・プールを割り付けるシステム・メモリ領域の種類
- キー ID 番号

10) 周期起動ハンドラ情報
個々の周期起動ハンドラに対して，次のデータを定義します。

- 指定番号
- 拡張情報
- 記述言語
- 起動アドレス
- 初期活性状態
- 起動時間間隔
- 固有 gp レジスタ値
- 固有 tp レジスタ値

11) 拡張 SVC ハンドラ情報
個々の拡張 SVC ハンドラに対して，次のデータを定義します。

- 拡張機能コード
- 記述言語
- 起動アドレス
- 固有 gp レジスタ値
- 固有 tp レジスタ値

12) 初期化ハンドラ情報
個々の初期化ハンドラに対して，次のデータを定義します。

- 記述言語
- 起動アドレス
- 固有 gp レジスタ値
- 固有 tp レジスタ値

13.3.3　SCT（System Call Table）情報

システム・コンフィギュレーション・ファイルに記述する SCT 情報は，次に示す 8 項目です。

1) タスク管理／タスク付属同期機能システム・コール情報
タスク管理／タスク付属同期機能システム・コール情報として，ユーザ処理プログラム内で使用するタスク管理／タスク付属同期機能システム・コールを定義します。
次に，RX850 Pro が提供しているタスク管理／タスク付属同期機能システム・コールを示します。

```
cre_tsk, del_tsk, sta_tsk, ext_tsk, exd_tsk, ter_tsk, dis_dsp, ena_dsp, chg_pri,
rot_rdq, rel_wai, get_tid, ref_tsk, vget_tid, sus_tsk, rsm_tsk, frsm_tsk, slp_tsk,
tslp_tsk, wup_tsk, can_wup
```

2) 同期通信（セマフォ）機能システム・コール情報
同期通信（セマフォ）機能システム・コール情報として，ユーザ処理プログラム内で使用する同期通信（セマフォ）機能システム・コールを定義します。
次に，RX850 Pro が提供している同期通信（セマフォ）機能システム・コールを示します。

```
cre_sem, del_sem, sig_sem, wai_sem, preq_sem, twai_sem, ref_sem, vget_sid
```

3) 同期通信（イベントフラグ）機能システム・コール情報
同期通信（イベントフラグ）機能システム・コール情報として，ユーザ処理プログラム内で使用する同期通信（イベントフラグ）機能システム・コールを定義します。
次に，RX850 Pro が提供している同期通信（イベントフラグ）機能システム・コールを示します。

```
cre_flg, del_flg, set_flg, clr_flg, wai_flg, pol_flg, twai_flg, ref_flg, vget_fid
```

4) 同期通信（メールボックス）機能システム・コール情報
同期通信（メールボックス）機能システム・コール情報として，ユーザ処理プログラム内で使用する同期通信（メールボックス）機能システム・コールを定義します。
次に，RX850 Pro が提供している同期通信（メールボックス）機能システム・コールを示します。

```
cre_mbx, del_mbx, snd_msg, rcv_msg, prcv_msg, trcv_msg, ref_mbx, vget_mid
```

5) 割り込み処理管理機能システム・コール情報
割り込み処理管理機能システム・コール情報として，ユーザ処理プログラム内で使用する割り込み処理管理システム・コールを定義します。
次に，RX850 Pro が提供している割り込み処理管理機能システム・コールを示します。

```
def_int, ena_int, dis_int, loc_cpu, unl_cpu, chg_icr, ref_icr
```

6）メモリ・プール管理機能システム・コール情報

メモリ・プール管理機能システム・コール情報として，ユーザ処理プログラム内で使用するメモリ・プール管理機能システム・コールを定義します。
次に，RX850 Pro が提供しているメモリ・プール管理機能システム・コールを示します。

```
cre_mpl，del_mpl，get_blk，pget_blk，tget_blk，rel_blk，ref_mpl，vget_pid
```

7）時間管理機能システム・コール情報

時間管理機能システム・コール機能情報として，ユーザ処理プログラム内で使用する時間処理管理システム・コール機能を定義します。
次に，RX850 Pro が提供している時間管理機能システム・コールを示します。

```
set_tim，get_tim，dly_tsk，def_cyc，act_cyc，ref_cyc
```

8）システム管理機能システム・コール情報

システム管理機能システム・コール情報として，ユーザ処理プログラム内で使用するシステム管理機能システム・コールを定義します。
次に，RX850 Pro が提供しているシステム管理機能システム・コールを示します。

```
get_ver，ref_sys，def_svc，viss_svc
```

13.4　リアルタイム OS 情報の記述形式

システム・コンフィギュレーション・ファイルに記述するリアルタイム OS 情報の記述形式を示します。
表記中の太字は予約語であることを，イタリック書体はユーザが該当する数値，シンボル名，キー・ワードを記述する部分であることを表しています。

13.4.1　RX シリーズ情報

RX シリーズ情報では，使用するリアルタイム OS の「リアルタイム OS 名，バージョン番号」を定義します。
なお，システム・コンフィギュレーション・ファイルに RX シリーズ情報を記述することは必須です。
以下に，RX シリーズ情報の記述形式を示します。

```
rxsers      rtos_nam    rtos_ver
```

次に，RX シリーズ情報で記述する各項目について示します。

- *rtos_nam*
  リアルタイム OS の名前（リアルタイム OS 名）を指定します。
  なお，*rtos_nam* として指定可能なキー・ワードは，RX850PRO に限られます。

- *rtos_ver*
  リアルタイム OS のバージョン番号を指定します。
  なお，*rtos_ver* として指定可能なキー・ワードは，V32*x*（*x* は 0 ～ 9 の任意の数字）に限られます。

13.5　SIT 情報の記述形式

システム・コンフィギュレーション・ファイルに記述する SIT 情報の記述形式を示します。
表記中の太字は予約語であることを，イタリック書体はユーザが該当する数値，シンボル名，キー・ワードを記述する部分であることを表しています。

13.5.1　システム情報

システム情報では，「プロセッサ種別，基本クロック周期，タイマの割り込み要因番号，デフォルト・スタック・サイズ，割り込みハンドラ用スタック情報，ID 番号の保護範囲」を定義します。
なお，システム・コンフィギュレーション・ファイルにシステム情報を記述することは必須です。
以下に，システム情報の記述形式を示します。

```
[ cputype       chip_type ]
clktim          time
clkhdr          clk_intno
defstk          stk_siz
intstk          intstk_siz:mem_nam
prttsk          tsk_idlmt
prtsem          sem_idlmt
prtflg          flg_idlmt
prtmbx          mbx_idlmt
prtmpl          mpl_idlmt
```

次に，システム情報で記述する各項目について示します。

- *chip_type*
  ターゲット・デバイスのプロセッサ種別を指定します。
  なお，*chip_type* として指定可能なキー・ワードは，V850ES，V850E1，V850E2 に限られます。

  V850ES：　V850ES コア
  V850E1：　V850E1 コア
  V850E2：　V850E2 コア

  省略時　ターゲット・デバイスのプロセッサ種別は，“V850E1”となります。

- *time*
  使用するタイマの基本クロック周期（単位：ms）を指定します。
  なお，*time* として指定可能な値は，0x1 ～ 0x7fff に限られます。

  備考 1　基本クロック周期とは，RX850 Pro が時間管理機能（周期起床，遅延起床，タイムアウトなど）を実現するうえで必要になるクロック割り込みの発生周期を意味します。

  備考 2　RX850 Pro が時間管理用に使用するタイマについては，1 ミリ秒周期で割り込みが発生するように初期化を行う必要があります。

- *clk_intno*
  タイマの割り込み要因番号を指定します。
  なお，*clk_intno* として指定可能な値は，デバイス・ファイルで既定されている割り込み要因名，または，“(例外コード - 0x80) / 0x10”で計算される値に限られます。

- *stk_siz*
  デフォルト・スタック・サイズ（単位：バイト）を指定します。
  なお，*stk_siz* として指定可能な値は，0x0 ～ 0x7ffffffc の 4 バイト境界値に限られます。

  備考　デフォルト・スタック・サイズとは，システム内で存在可能なタスク用スタックの最小サイズを意味します。したがって，システム起動時に行われる静的なタスクの生成や，cre_tsk の発行により行われる動的なタスクの生成において，デフォルト・スタック・サイズ以下のスタック・サイズが指定された場合には，そのサイズは無視され，デフォルト・スタック・サイズがスタック・サイズとして使用されます。

- *intstk_siz* : *mem_nam*
  割り込みハンドラ用スタックのサイズ（単位：バイト），および，割り込みハンドラ用スタックを割り付けるシステム・メモリ領域の種類を指定します。
  なお，*intstk_siz* として指定可能な値は，0x0 ～ 0x7ffffffc の 4 バイト境界値に限られます。
  また，*mem_nam* として指定可能なキー・ワードは，SPOL0，SPOL1 に限られます。

  SPOL0：　System Memory Pool 0 に割り込みハンドラ用スタックを割り付けます。
  SPOL1：　System Memory Pool 1 に割り込みハンドラ用スタックを割り付けます。

- *tsk_idlmt*
  タスクの ID 番号無指定生成が行われた際に保護する ID 番号の範囲を指定します。
  なお，*tsk_idlmt* として指定可能な範囲は，0x0 ～ *tsk_cnt* に限られます。

  備考　*tsk_idlmt* に 0x0 を指定した場合，ID 番号を保護しません。*tsk_cnt* はシステム最大値情報のタスクの最大生成数 maxtsk に定義された値となります。

- *sem_idlmt*
  セマフォの ID 番号無指定生成が行われた際に保護する ID 番号の範囲を指定します。
  なお，*sem_idlmt* として指定可能な範囲は，0x0 ～ *sem_cnt* に限られます。

  備考　*sem_idlmt* に 0x0 を指定した場合，ID 番号を保護しません。*sem_cnt* はシステム最大値情報のセマフォの最大生成数 maxsem に定義された値となります。

- *flg_idlmt*
  イベントフラグの ID 番号無指定生成が行われた際に保護する ID 番号の範囲を指定します。
  なお，*flg_idlmt* として指定可能な範囲は，0x0 ～ *flg_cnt* に限られます。

  備考　*flg_idlmt* に 0x0 を指定した場合，ID 番号を保護しません。*flg_cnt* はシステム最大値情報のイベントフラグの最大生成数 maxflg に定義された値となります。

- *mbx_idlmt*
  メールボックスの ID 番号無指定生成が行われた際に保護する ID 番号の範囲を指定します。
  なお，*mbx_idlmt* として指定可能な範囲は，0x0 ～ *mbx_cnt* に限られます。

  備考　*mbx_idlmt* に 0x0 を指定した場合，ID 番号を保護しません。*mbx_cnt* はシステム最大値情報のメールボックスの最大生成数 maxmbx に定義された値となります。

- *mpl_idlmt*
  メモリ・プールの ID 番号無指定生成が行われた際に保護する ID 番号の範囲を指定します。
  なお，*mpl_idlmt* として指定可能な範囲は，0x0 ～ *mpl_cnt* に限られます。

  備考　*mpl_idlmt* に 0x0 を指定した場合，ID 番号を保護しません。*mpl_cnt* はシステム最大値情報のメモリ・プールの最大生成数 maxmpl に定義された値となります。

13.5.2　システム最大値情報

システム最大値情報では，「タスクの優先度範囲，管理オブジェクトの最大生成数／最大登録数」を定義します。
なお，システム・コンフィギュレーション・ファイルにシステム最大値情報を記述することは必須です。
以下に，システム最大値情報の記述形式を示します。

```
maxpri                  pri_lvl
maxtsk                  tsk_cnt
maxsem                  sem_cnt
maxflg                  flg_cnt
maxmbx                  mbx_cnt
maxmpl                  mpl_cnt
maxcyc                  cyc_cnt
maxsvc                  svc_cnt
maxint                  ith_cnt
maxintfactor            itf_cnt
```

次に，システム最大値情報で記述する各項目について示します。

- *pri_lvl*
  タスクの優先度範囲（優先度数）を指定します。
  なお，*pri_lvl* として指定可能な値は，0x1 ～ 0xfc に限られます。

- *tsk_cnt*
  タスクの最大生成数を指定します。
  なお，*tsk_cnt* として指定可能な値は，0x1 ～ 0x7fff に限られます。

- *sem_cnt*
  セマフォの最大生成数を指定します。
  なお，*sem_cnt* として指定可能な値は，0x0 ～ 0x7fff に限られます。

- *flg_cnt*
  イベントフラグの最大生成数を指定します。
  なお，*flg_cnt* として指定可能な値は，0x0 ～ 0x7fff に限られます。

- *mbx_cnt*
  メールボックスの最大生成数を指定します。
  なお，*mbx_cnt* として指定可能な値は，0x0 ～ 0x7fff に限られます。

- *mpl_cnt*
  メモリ・プールの最大生成数を指定します。
  なお，*mpl_cnt* として指定可能な値は，0x0 ～ 0x7fff に限られます。

- *cyc_cnt*
  周期起動ハンドラの最大登録数を指定します。
  なお，*cyc_cnt* として指定可能な値は，0x0 ～ 0x7fff に限られます。

- *svc_cnt*
  拡張 SVC ハンドラの最大登録数を指定します。
  なお，*svc_cnt* として指定可能な値は，0x0 ～ 0x7fff に限られます。

- *ith_cnt*
  割り込みハンドラの最大登録数を指定します。
  なお，*ith_cnt* として指定可能な値は，0x0 ～ *itf_cnt* + 1 に限られます。

- *itf_cnt*
  間接起動割り込みハンドラ情報で定義した割り込み要因番号の最大値を指定します。
  なお, *itf_cnt* として指定可能な値は, “(間接起動割り込みハンドラに使用する割り込みの例外コードの最大値 - 0x80) / 0x10” で計算される値（0x0 ～ 0x7fff，デバイス・ファイルの指定時には 0x0 ～ “対象プロセッサで既定されている例外コードの最大値 - 0x80) / 0x10”））に限られます。

13.5.3　システム・メモリ情報

システム・メモリ情報では，「システム・メモリ領域の種類，セクション名，サイズ」を個々のシステム・メモリ領域（System Memory Pool 0，System Memory Pool 1，User Memory Pool 0，User Memory Pool 1）に対して定義します。

なお，システム・コンフィギュレーション・ファイルに System Memory Pool 0 に関するデータを記述することは必須です。

また，システム・コンフィギュレーション・ファイルに User Memory Pool 1 に関するデータを記述する場合，User Memory Pool 0 に関するデータを記述することは必須です。

以下に，システム・メモリ情報の記述形式を示します。

```
mem          mem_id       sec_nam      mem_siz
```

次に，システム・メモリ情報で記述する各項目について示します。

- *mem_id*

  システム・メモリ領域の種類を指定します。
  なお，*mem_id* として指定可能なキー・ワードは，SPOL0，SPOL1，UPOL0，UPOL1 のいずれかに限られます。

  SPOL0：　システム・メモリ領域の種類は，System Memory Pool 0 となります。
  SPOL1：　システム・メモリ領域の種類は，System Memory Pool 1 となります。
  UPOL0：　システム・メモリ領域の種類は，User Memory Pool 0 となります。
  UPOL1：　システム・メモリ領域の種類は，User Memory Pool 1 となります。

- *sec_nam*

  システム・メモリ領域が割り付けられるメモリ領域のセクション名を指定します。
  なお，*sec_nam* として指定可能な値は“リンク・ディレクティブ・ファイルで定義されているセクション名 *.sec_nam* から . を除いた名前”に限られます。

  備考　リンク・ディレクティブ・ファイルについての詳細は，「2.6　リンク・ディレクティブ・ファイルの作成」を参照してください。

- *mem_siz*

  システム・メモリ領域のサイズ（単位：バイト）を指定します。
  なお，*mem_siz* として指定可能な値は，0x100 ～ 0x7ffffffc の 4 バイト境界値に限られます。

13.5.4　タスク情報

タスク情報では，「ID 番号，初期状態，起動コード，拡張情報，記述言語，起動アドレス，初期優先度，割り込み状態，タスク用スタック情報，固有 gp レジスタ値，固有 tp レジスタ値，キー ID 番号」を個々のタスクに対して定義します。
なお，システム・コンフィギュレーション・ファイルにタスク情報は最低 1 個必要です。
また，タスク情報を定義可能な数は，1 ～システム最大値情報で指定したタスクの最大生成数 *tsk_cnt* の範囲に限られます。
以下に，タスク情報の記述形式を示します。

```
tsk         tsk_id      sts         sta_code    ext_inf     lang        ¥
            sta_adr     pri         intr        stk_siz:mem_nam         ¥
            data        text        key_id
```

次に，タスク情報で記述する各項目について示します。

- *tsk_id*
  タスクの ID 番号を指定します。
  なお，*tsk_id* として指定可能な値は，0x0 ～ *tsk_cnt*，シンボル名に限られます。
  tsk_id に 0x0，または，シンボル名を指定した場合，CF850 Pro が *tsk_idlmt* ～ *tsk_cnt* の範囲で未使用の ID 番号を自動的に割り当てます。
  なお，*tsk_idlmt* は，システム情報のタスクの ID 番号保護範囲 prttsk に定義された値となります。
  また，*tsk_cnt* は，システム最大値情報のタスクの最大生成数 maxtsk に定義された値となります。

- *sts*
  タスクの初期状態を指定します。
  なお，*sts* として指定可能なキー・ワードは，TTS_DMT，TTS_RDY のいずれかに限られます。

  TTS_DMT：　システム起動時，dormant 状態へと遷移します。
  TTS_RDY：　システム起動時，ready 状態へと遷移します。

  備考　すべての静的タスクの初期状態を TTS_DMT にすると，システム起動時に動作しているタスクがないことになってしまいます。この場合，初期化ハンドラで適当なタスクを起動する必要があります。

- *sta_code*
  タスクの起動コードを指定します。
  なお，*sta_code* として指定可能な値は，0x0 ～ 0xffffffff，シンボル名に限られます。

  備考　*sta_code* は，sts に TTS_RDY を指定した場合にのみ有効となり，TTS_DMT を指定した場合には無効となります。

- *ext_inf*
  タスクの拡張情報を指定します。
  なお，*ext_inf* として指定可能な値は，0x0 ～ 0xffffffff，シンボル名に限られます。

  備考　*ext_inf* は，対象タスクに関する“ユーザ独自の情報”を指定するためのものであり，ユーザが自由に指定できます。
  なお，*ext_inf* に指定された値は，処理プログラム（タスク／非タスク）から ref_tsk を発行することにより，動的に獲得できます。

- *lang*
  タスクの記述言語を指定します。
  なお，*lang* として指定可能なキー・ワードは，TA_HLNG，TA_ASM のどちらかに限られます。

  TA_HLNG：　タスクを C 言語で記述します。
  TA_ASM：　タスクをアセンブリ言語で記述します。

- *sta_adr*
  タスクの起動アドレスを指定します。
  なお，*sta_adr* として指定可能な値は，0x0 ～ 0xfffffffe の 2 バイト境界値，または，シンボル名に限られます。

- *pri*
  タスクの初期優先度を指定します。
  なお，*pri* として指定可能な値は，0x1 ～ *pri_lvl* に限られます。
  また，pri_lvl は，システム最大値情報のタスクの優先度範囲 maxpri に定義された値となります。

- *intr*
  タスク起動時の割り込み状態を指定します。
  なお，*intr* として指定可能なキー・ワードは，TA_ENAINT，TA_DISINT のどちらかに限られます。

  TA_ENAINT：　タスク起動時，すべてのマスカブル割り込みを許可します。
  TA_DISINT：　タスク起動時，すべてのマスカブル割り込みを禁止します。

- *stk_siz* : *mem_nam*
  タスク用スタックのサイズ（単位：バイト），および，タスク用スタックを割り付けるシステム・メモリ領域の種類を指定します。
  なお，*stk_siz* として指定可能な値は，0x0 ～ 0x7ffffffc の 4 バイト境界値に限られます。
  また，*mem_nam* として指定可能なキー・ワードは，SPOL0，SPOL1 のいずれかに限られます。

  SPOL0：　システム・メモリ領域の種類は，System Memory Pool 0 となります。
  SPOL1：　システム・メモリ領域の種類は，System Memory Pool 1 となります。

- *data*
  タスクの固有 gp レジスタ値を指定します。
  なお，*data* として指定可能な値／キー・ワードは，0x0 ～ 0xffffffff，シンボル名，no_use に限られます。

  no_use：　固有 gp レジスタ値の設定は行われません。

- *text*
  タスクの固有 tp レジスタ値を指定します。
  なお，*text* として指定可能な値／キー・ワードは，0x0 ～ 0xffffffff，シンボル名，no_use に限られます。

  no_use：　固有 tp レジスタ値の設定は行われません。

- *key_id*
  タスクのキー ID 番号を指定します。
  なお，*key_id* として指定可能な値は，0x0 ～ 0x7fff に限られます。

  備考　*key_id* に 0x0 を設定した場合，CF850 Pro はキー ID 番号の割り付けを行いません。

13.5.5　セマフォ情報

セマフォ情報では，「ID 番号，拡張情報，タスクのキューイング方式，初期資源数，最大資源数，キー ID 番号」を個々のセマフォに対して定義します。
なお，セマフォ情報として定義可能な数は，0 ～システム最大値情報で登録したセマフォの最大生成数 *sem_cnt* の範囲に限られます。
以下に，セマフォ情報の記述形式を示します。

```
sem         sem_id      ext_inf     twai_opt    init_cnt    max_cnt     key_id
```

次に，セマフォ情報で記述する各項目について示します。

- *sem_id*
  セマフォの ID 番号を指定します。
  なお，*sem_id* として指定可能な値は，0x0 ～ *sem_cnt*，シンボル名に限られます。
  sem_id に 0x0，または，シンボル名を指定した場合，CF850 Pro が *sem_idlmt* ～ *sem_cnt* の範囲で未使用の ID 番号を自動的に割り当てます。
  なお，*sem_idlmt* は，システム情報のセマフォの ID 番号保護範囲 prtsem に定義された値となります。
  また，*sem_cnt* は，システム最大値情報のセマフォの最大生成数 maxsem に定義された値となります。

- *ext_inf*
  セマフォの拡張情報を指定します。
  なお，*ext_inf* として指定可能な値は，0x0 ～ 0xffffffff，シンボル名に限られます。

  備考　*ext_inf* は，対象セマフォに関する“ユーザ独自の情報”を指定するためのものであり，ユーザが自由に指定できます。
  なお，*ext_inf* に指定された値は，処理プログラム（タスク／非タスク）から ref_sem を発行することにより，動的に獲得できます。

- *twai_opt*
  タスクのキューイング方式を指定します。
  なお，*twai_opt* として指定可能なキー・ワードは，TA_TFIFO，TA_TPRI のいずれかに限られます。

  TA_TFIFO：　タスクのキューイング方式は，資源の獲得要求を行った順になります。
  TA_TPRI：　タスクのキューイング方式は，タスクの優先度順になります。

- *init_cnt*
  セマフォの初期資源数を指定します。
  なお，*init_cnt* として指定可能な数値は，0x0 ～ *max_cnt* に限られます。

- *max_cnt*
  セマフォの最大資源数を指定します。
  なお，*max_cnt* として指定可能な数値は，0x1 ～ 0x7fffffff に限られます。

- *key_id*
  セマフォのキー ID 番号を指定します。
  なお，*key_id* として指定可能な値は，0x0 ～ 0x7fff に限られます。

  備考　*key_id* に 0x0 を設定した場合，CF850 Pro はキー ID 番号の割り付けを行いません。

13.5.6　イベントフラグ情報

イベントフラグ情報では，「ID 番号，拡張情報，複数タスク待ちの許可／禁止，初期ビット・パターン，キー ID 番号」を個々のイベントフラグに対して定義します。

なお，イベントフラグ情報として定義可能な数は，0 ～システム最大値情報で登録したイベントフラグの最大生成数 *flg_cnt* に限られます。

以下に，イベントフラグ情報の記述形式を示します。

```
flg        flg_id      ext_inf     twai_opt    init_ptn    key_id
```

次に，イベントフラグ情報で記述する各項目について示します。

- *fig_id*
  イベントフラグの ID 番号を指定します。
  なお，*fig_id* として指定可能な値は，0x0 ～ *flg_cnt*，シンボル名に限られます。
  flg_id に 0x0，または，シンボル名を指定した場合，CF850 Pro が *flg_idlmt* ～ *flg_cnt* の範囲で未使用の ID 番号を自動的に割り当てます。
  なお，*flg_idlmt* は，システム情報のイベントフラグの ID 番号保護範囲 prtflg に定義された値となります。
  また，*flg_cnt* は，システム最大値情報のイベントフラグの最大生成数 maxflg に定義された値となります。

- *ext_inf*
  イベントフラグの拡張情報を指定します。
  なお，*ext_inf* として指定可能な値は，0x0 ～ 0xffffffff，シンボル名に限られます。

  備考　*ext_inf* は，対象イベントフラグに関する“ユーザ独自の情報”を指定するためのものであり，ユーザが自由に指定できます。
  なお，*ext_inf* に指定された値は，処理プログラム（タスク／非タスク）から ref_flg を発行することにより，動的に獲得できます。

- *twai_opt*
  複数タスク待ちの許可／禁止を指定します。
  なお，*twai_opt* として指定可能なキー・ワードは，TA_WSGL，TA_WMUL のいずれかに限られます。

  TA_WSGL：　複数タスク待ちは禁止になります。
  TA_WMUL：　複数タスク待ちは許可になります。

- *init_ptn*
  イベントフラグの初期ビット・パターン（32 ビット幅）を指定します。
  なお，*init_ptn* として指定可能な値は，0x0 ～ 0xffffffff に限られます。

- *key_id*
  イベントフラグのキー ID 番号を指定します。
  なお，*key_id* として指定可能な値は，0x0 ～ 0x7fff に限られます。

  備考　*key_id* に 0x0 を設定した場合，CF850 Pro はキー ID 番号の割り付けを行いません。

13.5.7　メールボックス情報

メールボックス情報では，「ID 番号，拡張情報，タスクのキューイング方式，メッセージのキューイング方式，キー ID 番号」を個々のメールボックスに対して定義します。
なお，メールボックス情報として定義可能な数は，0 ～システム最大値情報で登録したメールボックスの最大生成数 *mbx_cnt* に限られます。
以下に，メールボックス情報の記述形式を示します。

```
mbx         mbx_id      ext_inf     twai_opt    mwai_opt    key_id
```

次に，メールボックス情報で記述する各項目について示します。

- *mbx_id*
  メールボックスの ID 番号を指定します。
  なお，*mbx_id* として指定可能な値は，0x0 ～ *mbx_cnt*，シンボル名に限られます。
  mbx_id に 0x0，または，シンボル名を指定した場合，CF850 Pro が *mbx_idlmt* ～ *mbx_cnt* の範囲で未使用の ID 番号を自動的に割り当てます。
  なお，*mbx_idlmt* は，システム情報のメールボックスの ID 番号保護範囲 prtmbx に定義された値となります。
  また，*mbx_cnt* は，システム最大値情報のメールボックスの最大生成数 maxmbx に定義された値となります。

- *ext_inf*
  メールボックスの拡張情報を指定します。
  なお，*ext_inf* として指定可能な値は，0x0 ～ 0xffffffff，シンボル名に限られます。

  備考　*ext_inf* は，対象メールボックスに関する“ユーザ独自の情報”を指定するためのものであり，ユーザが自由に指定できます。
  なお，*ext_inf* に指定された値は，処理プログラム（タスク／非タスク）から ref_mbx を発行することにより，動的に獲得できます。

- *twai_opt*
  タスクのキューイング方式を指定します。
  なお，*twai_opt* として指定可能なキー・ワードは，TA_TFIFO，TA_TPRI のいずれかに限られます。

  TA_TFIFO：　タスクのキューイング方式は，メッセージの受信要求を行った順になります。
  TA_TPRI：　タスクのキューイング方式は，タスクの優先度順になります。

- *mwai_opt*
  メッセージのキューイング方式を指定します。
  なお，*mwai_opt* として指定可能なキー・ワードは，TA_MFIFO，TA_MPRI のいずれかに限られます。

  TA_MFIFO：　メッセージのキューイング方式は，メッセージの送信を行った順になります。
  TA_MPRI：　メッセージのキューイング方式は，メッセージの優先度順になります。

- *key_id*
  メールボックスのキー ID 番号を指定します。
  なお，*key_id* として指定可能な値は，0x0 ～ 0x7fff に限られます。

  備考　*key_id* に 0x0 を設定した場合，CF850 Pro はキー ID 番号の割り付けを行いません。

13.5.8　間接起動割り込みハンドラ情報

間接起動割り込みハンドラ情報では，「割り込み要因番号，記述言語，起動アドレス，固有 gp レジスタ値，固有 tp レジスタ値」を個々の間接起動割り込みハンドラに対して定義します。

なお，間接起動割り込みハンドラ情報として定義可能な数は，0 ～システム最大値情報で登録した間接起動割り込みハンドラの最大生成数 *ith_cnt* に限られます。

以下に，間接起動割り込みハンドラ情報の記述形式を示します。

```
inthdr      int_no      lang        hdr_adr     data        text
```

次に，間接起動割り込みハンドラ情報で記述する各項目について示します。

- *int_no*

  間接起動割り込みハンドラの割り込み要因番号を指定します。
  なお，*int_no* として指定可能な値は，デバイス・ファイルで既定されている割り込み要因名，または，“(例外コード - 0x80) / 0x10”で計算される値に限られます。

  備考　*int_no* に *clk_intno* と同じ割り込み要因番号を指定することはできません。
  なお，*clk_intno* は，システム情報のタイマの割り込み要因番号 clkhdr に定義された値となります。

- *lang*

  間接起動割り込みハンドラの記述言語を指定します。
  なお，*lang* として指定可能なキー・ワードは，TA_HLNG，TA_ASM のどちらかに限られます。

  TA_HLNG：　間接起動割り込みハンドラを C 言語で記述します。
  TA_ASM：　間接起動割り込みハンドラをアセンブリ言語で記述します。

- *hdr_adr*

  間接起動割り込みハンドラの起動アドレスを指定します。
  なお，*hdr_adr* として指定可能な値は，0x0 ～ 0xfffffffe の 2 バイト境界値，シンボル名に限られます。

- *data*

  間接起動割り込みハンドラの固有 gp レジスタ値を指定します。
  なお，*data* として指定可能な値／キー・ワードは，0x0 ～ 0xffffffff，シンボル名，no_use に限られます。

  no_use：　固有 gp レジスタ値の設定は行われません。

- *text*

  間接起動割り込みハンドラの固有 tp レジスタ値を指定します。
  なお，*text* として指定可能な値／キー・ワードは，0x0 ～ 0xffffffff，シンボル名，no_use に限られます。

  no_use：　固有 tp レジスタ値の設定は行われません。

13.5.9　メモリ・プール情報

メモリ・プール情報では，「ID 番号，拡張情報，タスクのキューイング方式，メモリ・プール情報，キー ID 番号」を個々のメモリ・プールに対して定義します。

なお，メモリ・プール情報として定義可能な数は，0 ～システム最大値情報で登録したメモリ・プールの最大生成数 *mpl_cnt* に限られます。

以下に，メモリ・プール情報の記述形式を示します。

```
mpl         mpl_id      ext_inf     twai_opt    mpl_siz:mem_nam         key_id
```

次に，メモリ・プール情報で記述する各項目について示します。

- *mpl_id*

  メモリ・プールの ID 番号を指定します。

  なお，*mpl_id* として指定可能な値は，0x0 ～ *mpl_cnt*，シンボル名に限られます。

  mpl_id に 0x0，または，シンボル名を指定した場合，CF850 Pro が *mpl_idlmt* ～ *mpl_cnt* の範囲で未使用の ID 番号を自動的に割り当てます。

  なお，*mpl_idlmt* は，システム情報のメモリ・プールの ID 番号保護範囲 prtmpl に定義された値となります。

  また，*mpl_cnt* は，システム最大値情報のメモリ・プールの最大生成数 maxmpl に定義された値となります。

- *ext_inf*

  メモリ・プールの拡張情報を指定します。

  なお，*ext_inf* として指定可能な値は，0x0 ～ 0xffffffff，シンボル名に限られます。

  備考　*ext_inf* は，対象メモリ・プールに関する“ユーザ独自の情報”を指定するためのものであり，ユーザが自由に指定できます。
  なお，*ext_inf* に指定された値は，処理プログラム（タスク／非タスク）から ref_mpl を発行することにより，動的に獲得できます。

- *twai_opt*

  タスクのキューイング方式を指定します。

  なお，*twai_opt* として指定可能なキー・ワードは，TA_TFIFO，TA_TPRI のいずれかに限られます。

  TA_TFIFO：　タスクのキューイング方式は，メモリ・ブロックの要求を行った順になります。
  TA_TPRI：　タスクのキューイング方式は，タスクの優先度順になります。

- *mpl_siz* : *mem_nam*

  メモリ・プールのサイズ（単位：バイト），および，メモリ・プールを割り付けるシステム・メモリ領域の種類を指定します。

  なお，*mpl_siz* として指定可能な値は，0x4 ～ 0x7ffffffc の 4 バイト境界値に限られます。

  また，*mem_nam* として指定可能なキー・ワードは，UPOL0，UPOL1 のいずれかに限られます。

  UPOL0：　システム・メモリ領域の種類は，User Memory Pool 0 となります。
  UPOL1：　システム・メモリ領域の種類は，User Memory Pool 1 となります。

- *key_id*

  メモリ・プールのキー ID 番号を指定します。

  なお，*key_id* として指定可能な値は，0x0 ～ 0x7fff に限られます。

  備考　*key_id* に 0x0 を設定した場合，CF850 Pro はキー ID 番号の割り付けを行いません。

13.5.10 周期起動ハンドラ情報

周期起動ハンドラ情報では，「指定番号，拡張情報，記述言語，起動アドレス，初期活性状態，起動時間間隔，固有 gp レジスタ値，固有 tp レジスタ値」を個々の周期起動ハンドラに対して定義します。

なお，周期起動ハンドラ情報として定義可能な数は，0 ～システム最大値情報で登録した周期起動ハンドラの最大登録数 *cyc_cnt* の範囲に限られます。

以下に，周期起動ハンドラ情報の記述形式を示します。

```
cyc         cyc_no      ext_inf     lang        hdr_adr     act         ¥
            intvl       data        text
```

次に，周期起動ハンドラ情報で記述する各項目について示します。

- *cyc_no*
  周期起動ハンドラの指定番号を指定します。
  なお，*cyc_no* として指定可能な値は，0x1 ～ *cyc_cnt*，シンボル名に限られます。
  *cyc_no*にシンボル名を指定した場合, CF850 Proが0x1～*cyc_cnt*の範囲で未使用のID番号を自動的に割り当てます。
  なお，*cyc_cnt* は，システム最大値情報の周期起動ハンドラの最大登録数 maxcyc に定義された値となります。

- *ext_inf*
  周期起動ハンドラの拡張情報を指定します。
  なお，*ext_inf* として指定可能な値は，0x0 ～ 0xffffffff，シンボル名に限られます。

  備考　*ext_inf* は，対象周期起動ハンドラに関する“ユーザ独自の情報”を指定するためのものであり，ユーザが自由に指定できます。
  なお，*ext_inf* に指定された値は，処理プログラム（タスク／非タスク）から ref_cyc を発行することにより，動的に獲得できます。

- *lang*
  周期起動ハンドラの記述言語を指定します。
  なお，*lang* として指定可能なキー・ワードは，TA_HLNG，TA_ASM のどちらかに限られます。

  TA_HLNG：　周期起動ハンドラを C 言語で記述します。
  TA_ASM：　周期起動ハンドラをアセンブリ言語で記述します。

- *hdr_adr*
  周期起動ハンドラの起動アドレスを指定します。
  なお，*hdr_adr* として指定可能な値は，0 ～ 0xfffffffe の 2 バイト境界値，シンボル名に限られます。

- *act*
  周期起動ハンドラの初期活性状態を指定します。
  なお，*act* として指定可能なキー・ワードは，TCY_ON，TCY_OFF のどちらかに限られます。

  TCY_ON：　システム起動時，活性状態は ON 状態になります。
  TCY_OFF：　システム起動時，活性状態は OFF 状態になります。

- *intvl*
  周期起動ハンドラの起動時間間隔（単位：基本クロック周期）を指定します。
  なお，*intvl* として指定可能な値は，0x1 ～ 0xffffffff に限られます。

- *data*
  周期起動ハンドラの固有 gp レジスタ値を指定します。
  なお，*data* として指定可能な値／キー・ワードは，0x0 ～ 0xffffffff，シンボル名，no_use に限られます。

  no_use：　固有 gp レジスタ値の設定は行われません。

- *text*
  周期起動ハンドラの固有 tp レジスタ値を指定します。
  なお，*text* として指定可能な値／キー・ワードは，0x0 ～ 0xffffffff，シンボル名，no_use に限られます。

  no_use：　固有 tp レジスタ値の設定は行われません。

13.5.11 拡張 SVC ハンドラ情報

拡張 SVC ハンドラ情報では，「拡張機能コード，記述言語，起動アドレス，固有 gp レジスタ値，固有 tp レジスタ値」を個々の拡張 SVC ハンドラに対して定義します。

なお，拡張 SVC ハンドラ情報として定義可能な数は，0 ～システム最大値情報で登録した拡張 SVC ハンドラの最大登録数 *scv_cnt* に限られます。

以下に，拡張 SVC ハンドラ情報の記述形式を示します。

```
svc        svc_no      lang        hdr_adr     data        text
```

次に，拡張 SVC ハンドラ情報で記述する各項目について示します。

- *svc_no*
  拡張 SVC ハンドラの拡張機能コードを指定します。
  なお，*svc_no* として指定可能な値は，0x1 ～ *svc_cnt*，シンボル名に限られます。
  *svc_no*にシンボル名を指定した場合, CF850 Proが0x1～*svc_cnt*の範囲で未使用のID番号を自動的に割り当てます。
  なお，*svc_cnt* は，システム最大値情報の拡張 SVC ハンドラの最大登録数 maxsvc に定義された値となります。

- *lang*
  拡張 SVC ハンドラの記述言語を指定します。
  なお，*lang* として指定可能なキー・ワードは，TA_HLNG，TA_ASM のどちらかに限られます。

  TA_HLNG：　拡張 SVC ハンドラを C 言語で記述します。
  TA_ASM：　拡張 SVC ハンドラをアセンブリ言語で記述します。

- *hdr_adr*
  拡張 SVC ハンドラの起動アドレスを指定します。
  なお，*hdr_adr* として指定可能な値は，0x0 ～ 0xfffffffe の 2 バイト境界値，シンボル名に限られます。

- *data*
  拡張 SVC ハンドラの固有 gp レジスタ値を指定します。
  なお，*data* として指定可能な値／キー・ワードは，0x0 ～ 0xffffffff，シンボル名，no_use に限られます。

  no_use：　固有 gp レジスタ値の設定は行われません。

- *text*
  拡張 SVC ハンドラの固有 tp レジスタ値を指定します。
  なお，*text* として指定可能な値／キー・ワードは，0x0 ～ 0xffffffff，シンボル名，no_use に限られます。

  no_use：　固有 tp レジスタ値の設定は行われません。

13.5.12 初期化ハンドラ情報

初期化ハンドラ情報では，初期化ハンドラの「記述言語，起動アドレス，固有 gp レジスタ値，固有 tp レジスタ値」を定義します。

なお，システム・コンフィギュレーション・ファイルにて，初期化ハンドラの情報を省略することが可能です。省略した場合，RX850 Pro は初期化ハンドラがないものと判断し，処理を続けます。

以下に，初期化ハンドラ情報の記述形式を示します。

```
ini         lang        hdr_adr     data        text
```

次に，初期化ハンドラ情報で記述する各項目について示します。

- *lang*

  初期化ハンドラの記述言語を指定します。
  なお，*lang* として指定可能なキー・ワードは，TA_HLNG，TA_ASM のどちらかに限られます。

  TA_HLNG：　初期化ハンドラを C 言語で記述します。
  TA_ASM：　初期化ハンドラをアセンブリ言語で記述します。

- *hdr_adr*

  初期化ハンドラの起動アドレスを指定します。
  なお，*hdr_adr* として指定可能な値は，0x0 ～ 0xfffffffe の 2 バイト境界値，シンボル名に限られます。

- *data*

  初期化ハンドラの固有 gp レジスタ値を指定します。
  なお，*data* として指定可能な値／キー・ワードは，0x0 ～ 0xffffffff，シンボル名，no_use に限られます。

  no_use：　固有 gp レジスタ値の設定は行われません。

- *text*

  初期化ハンドラの固有 tp レジスタ値を指定します。
  なお，*text* として指定可能な値／キー・ワードは，0x0 ～ 0xffffffff，シンボル名，no_use に限られます。

  no_use：　固有 tp レジスタ値の設定は行われません。

13.6　SCT 情報の記述形式

システム・コンフィギュレーション・ファイルに記述する SCT 情報の記述形式を示します。

表記中の太字は予約語であることを，イタリック書体はユーザが該当する数値，シンボル名，キー・ワードを記述する部分であることを表しています。

13.6.1　タスク管理／タスク付属同期機能システム・コール情報

タスク管理／タスク付属同期機能システム・コール情報では，ユーザ処理プログラム内で使用する「タスク管理／タスク付属同期機能システム・コール」を個々のシステム・コールに対して定義します。

ここで定義せずにアプリケーションでシステム・コールを使用した場合，システム・コールの戻り値として E_NOSPT（-17）が入ります。

以下に，タスク管理／タスク付属同期機能システム・コール情報の記述形式を示します。

```
tsksvc      svc_nam
```

次に，タスク管理／タスク付属同期機能システム・コールで記述する各項目について示します。

- *svc_nam*

  タスク管理機能システム・コール名を指定します。

  なお，*svc_nam* として指定可能なキー・ワードは，次のものに限られます。

  cre_tsk，del_tsk，sta_tsk，ext_tsk，exd_tsk，ter_tsk，dis_dsp，ena_dsp，chg_pri，rot_rdq，rel_wai，get_tid，ref_tsk，vget_tid，sus_tsk，rsm_tsk，frsm_tsk，slp_tsk，tslp_tsk，wup_tsk，can_wup

13.6.2　同期通信（セマフォ）機能システム・コール情報

同期通信（セマフォ）機能システム・コール情報では，ユーザ処理プログラム内で使用する「同期通信（セマフォ）機能システム・コール」を個々のシステム・コールに対して定義します。

ここで定義せずにアプリケーションでシステム・コールを使用した場合，システム・コールの戻り値として E_NOSPT（-17）が入ります。

以下に，同期通信（セマフォ）機能システム・コール情報の記述形式を示します。

semsvc	*svc_nam*

次に，同期通信（セマフォ）機能システム・コールで記述する各項目について示します。

- *svc_nam*

  同期通信（セマフォ）機能システム・コール名を指定します。
  なお，*svc_nam* として指定可能なキー・ワードは，次のものに限られます。

  ```
  cre_sem，del_sem，sig_sem，wai_sem，preq_sem，twai_sem，ref_sem，vget_sid
  ```

13.6.3　同期通信（イベントフラグ）機能システム・コール情報

同期通信（イベントフラグ）機能システム・コール情報では，ユーザ処理プログラム内で使用する「同期通信（イベントフラグ）機能システム・コール」を個々のシステム・コールに対して定義します。
ここで定義せずにアプリケーションでシステム・コールを使用した場合，システム・コールの戻り値として E_NOSPT（-17）が入ります。
以下に，同期通信（イベントフラグ）機能システム・コール情報の記述形式を示します。

```
flgsvc      svc_nam
```

次に，同期通信（イベントフラグ）機能システム・コールで記述する各項目について示します。

- *svc_nam*
  同期通信（イベントフラグ）機能システム・コール名を指定します。
  なお，*svc_nam* として指定可能なキー・ワードは，次のものに限られます。

    cre_flg，del_flg，set_flg，clr_flg，wai_flg，pol_flg，twai_flg，ref_flg，vget_fid

13.6.4　同期通信（メールボックス）機能システム・コール情報

同期通信（メールボックス）機能システム・コール情報では，ユーザ処理プログラム内で使用する「同期通信（メールボックス）機能システム・コール」を個々のシステム・コールに対して定義します。

ここで定義せずにアプリケーションでシステム・コールを使用した場合，システム・コールの戻り値として E_NOSPT（-17）が入ります。

以下に，同期通信（メールボックス）機能システム・コール情報の記述形式を示します。

mbxsvc　　*svc_nam*

次に，同期通信（メールボックス）機能システム・コールで記述する各項目について示します。

- *svc_nam*

  同期通信（メールボックス）機能システム・コール名を指定します。

  なお，*svc_nam* として指定可能なキー・ワードは，次のものに限られます。

  ```
  cre_mbx，del_mbx，snd_msg，rcv_msg，prcv_msg，trcv_msg，ref_mbx，vget_mid
  ```

13.6.5　割り込み処理管理機能システム・コール情報

割り込み処理管理機能システム・コール情報では，ユーザ処理プログラム内で使用する「割り込み処理管理機能システム・コール」を個々のシステム・コールに対して定義します。

ここで定義せずにアプリケーションでシステム・コールを使用した場合，システム・コールの戻り値として E_NOSPT（-17）が入ります。

以下に，割り込み処理管理機能システム・コール情報の記述形式を示します。

intsvc　　*svc_nam*

次に，割り込み処理管理機能システム・コールで記述する各項目について示します。

- *svc_nam*

  割り込み処理管理機能システム・コール名を指定します。

  なお，*svc_nam* として指定可能なキー・ワードは，次のものに限られます。

  ```
  def_int，ena_int，dis_int，loc_cpu，unl_cpu，chg_icr，ref_icr
  ```

13.6.6　メモリ・プール管理機能システム・コール情報

メモリ・プール管理機能システム・コール情報では，ユーザ処理プログラム内で使用する「メモリ・プール管理機能システム・コール」を個々のシステム・コールに対して定義します。

ここで定義せずにアプリケーションでシステム・コールを使用した場合，システム・コールの戻り値として E_NOSPT（-17）が入ります。

以下に，メモリ・プール管理機能システム・コール情報の記述形式を示します。

mplsvc　　*svc_nam*

次に，メモリ・プール管理機能システム・コールで記述する各項目について示します。

- *svc_nam*

  メモリ・プール管理機能システム・コール名を指定します。

  なお，*svc_nam* として指定可能なキー・ワードは，次のものに限られます。

  `cre_mpl`，`del_mpl`，`get_blk`，`pget_blk`，`tget_blk`，`rel_blk`，`ref_mpl`，`vget_pid`

13.6.7　時間管理機能システム・コール情報

時間管理機能システム・コール情報では，ユーザ処理プログラム内で使用する「時間管理機能システム・コール」を個々のシステム・コールに対して定義します。

ここで定義せずにアプリケーションでシステム・コールを使用した場合，システム・コールの戻り値として E_NOSPT（-17）が入ります。

以下に，時間管理機能システム・コール情報の記述形式を示します。

timsvc　　*svc_nam*

次に，時間管理機能システム・コールで記述する各項目について示します。

- *svc_nam*
  時間管理機能システム・コール名を指定します。
  なお，*svc_nam* として指定可能なキー・ワードは，次のものに限られます。

    `set_tim`，`get_tim`，`dly_tsk`，`def_cyc`，`act_cyc`，`ref_cyc`

13.6.8　システム管理機能システム・コール情報

システム管理機能システム・コール情報では，ユーザ処理プログラム内で使用する「システム管理機能システム・コール」を個々のシステム・コールに対して定義します。

ここで定義せずにアプリケーションでシステム・コールを使用した場合，システム・コールの戻り値として E_NOSPT（-17）が入ります。

以下に，システム管理機能システム・コール情報の記述形式を示します。

syssvc　　*svc_nam*

次に，システム管理機能システム・コールで記述する各項目について示します。

- *svc_nam*

  システム管理機能システム・コール名を指定します。

  なお，*svc_nam* として指定可能なキー・ワードは，次のものに限られます。

  ```
  get_ver，ref_sys，def_svc，viss_svc
  ```

13.7　記述上の注意点

システム・コンフィギュレーション・ファイルにコンフィギュレーション情報（リアルタイム OS 情報，SIT 情報，SCT 情報）を記述する場合，次の順序で行います。

1）リアルタイム OS 情報の記述が開始されることの宣言

2）リアルタイム OS 情報の記述

3）SIT（System Information Table）情報の記述が開始されることの宣言

4）SIT（System Information Table）情報の記述

5）SCT（System Call Table）情報の記述が開始されることの宣言

6）SCT（System Call Table）情報の記述

図 13 － 1 に，システム・コンフィギュレーション・ファイルの記述イメージを示します。

図 13 － 1　システム・コンフィギュレーション・ファイルの記述イメージ

```
-- リアルタイム OS 情報の記述が開始されることの宣言
ser_def

-- リアルタイム OS 情報の記述
.....
.....
.....

-- SIT（System Information Table）情報の記述が開始されることの宣言
sit_def

-- SIT（System Information Table）情報の記述
.....
.....
.....

-- SCT（System Call Table）情報の記述が開始されることの宣言
sct_def

-- SCT（System Call Table）情報の記述
.....
.....
.....
```

13.8　記 述 例

以下に，システム・コンフィギュレーション・ファイルの記述例を示します。
なお，記述例では，次に示すデータが記述されています。

【 リアルタイム OS 情報 】

1) RX シリーズ情報

リアルタイム OS 名：	RX850PRO
バージョン番号：	V321

【 SIT（System Information Table）情報 】

1) システム情報

プロセッサ種別：	V850E1 コア
基本クロック周期：	0x1 ms
タイマの割り込み要因番号：	0x1c（INTCMD0）
デフォルト・スタック・サイズ：	0x100 バイト
割り込みハンドラ用スタック情報：	System Memory Pool 0 から 0x100 バイトのメモリ領域を確保
タスクの ID 番号保護範囲：	0x1
セマフォの ID 番号保護範囲：	0x1
イベントフラグの ID 番号保護範囲：	0x1
メールボックスの ID 番号保護範囲：	0x1
メモリ・プールの ID 番号保護範囲：	0x1

2) システム最大値情報

タスクの優先度範囲：	0xf
タスクの最大生成数：	0x2
セマフォの最大生成数：	0x1
イベントフラグの最大生成数：	0x2
メールボックスの最大生成数：	0x3
メモリ・プールの最大生成数：	0x2
周期起動ハンドラの最大登録数：	0x1
拡張 SVC ハンドラの最大登録数：	0x1
割り込みハンドラの最大登録数：	0x5
割り込み要因番号の最大値：	0x30

3) システム・メモリ情報

System Memory Pool 0：	.syspol0 セクションから 0x2000 バイトのメモリ領域を確保
System Memory Pool 1：	.syspol1 セクションから 0x1000 バイトのメモリ領域を確保
User Memory Pool 0：	.usrpol0_0 セクションから 0x7000 バイトのメモリ領域を確保
User Memory Pool 0：	.usrpol0_1 セクションから 0x2500 バイトのメモリ領域を確保
User Memory Pool 1：	.usrpol1 セクションから 0x1500 バイトのメモリ領域を確保

4) タスク情報

ID 番号：	0x1
初期状態：	ready 状態
起動コード：	0x0
拡張情報：	0x0
記述言語：	アセンブリ言語
起動アドレス：	_task01
初期優先度：	0x8
割り込み状態：	すべてのマスカブル割り込みを許可
タスク用スタック情報：	System Memory Pool 0 から 0x100 バイトのメモリ容量を確保
固有 gp レジスタ値：	設定しない
固有 tp レジスタ値：	設定しない
キー ID 番号：	0x1

ID 番号：	TASK02
初期状態：	dormant 状態
起動コード：	0x0
拡張情報：	0x0
記述言語：	C 言語
起動アドレス：	_task02

初期優先度：　0xf
割り込み状態：　すべてのマスカブル割り込みを禁止
タスク用スタック情報：　System Memory Pool 0 から 0x100 バイトのメモリ容量を確保
固有 gp レジスタ値：　設定しない
固有 tp レジスタ値：　設定しない
キー ID 番号：　0x2

5) セマフォ情報
ID 番号：　0x1
拡張情報：　0x0
タスクのキューイング方式：　資源の獲得要求を行った順（FIFO 順）
初期資源数：　0xff
最大資源数：　0xff
キー ID 番号：　0x1

6) イベントフラグ情報
ID 番号：　0x1
拡張情報：　0x0
複数タスクの待ち許可／禁止：　禁止
初期ビット・パターン：　0x0
キー ID 番号：　0x1

7) メールボックス情報
ID 番号：　0x1
拡張情報：　0x0
タスクのキューイング方式：　メッセージの受信要求を行った順（FIFO 順）
メッセージのキューイング方式：　メッセージの送信を行った順（FIFO 順）
キー ID 番号：　0x1

8) 間接起動割り込みハンドラ情報
割り込み要因番号：　0x14（INTP120）
記述言語：　アセンブリ言語
起動アドレス：　_inthdr01
固有 gp レジスタ値：　設定しない
固有 tp レジスタ値：　設定しない

9) メモリ・プール情報
ID 番号：　0x1
拡張情報：　0x0
タスクのキューイング方式：　タスクの優先度順
メモリ・プール情報：　User Memory Pool 0 から 0x2000 バイトのメモリ領域を確保
キー ID 番号：　0x1

10) 周期起動ハンドラ情報
指定番号：　0x1
拡張情報：　0x0
記述言語：　C 言語
起動アドレス：　_cychdr01
初期活性状態：　OFF 状態
起動時間間隔：　0x100 基本クロック周期
固有 gp レジスタ値：　設定しない
固有 tp レジスタ値：　設定しない

11) 拡張 SVC ハンドラ情報
拡張機能コード：　0x1
記述言語：　C 言語
起動アドレス：　_svchdr01
固有 gp レジスタ値：　設定しない
固有 tp レジスタ値：　設定しない

12) 初期化ハンドラ情報
記述言語：　C 言語
起動アドレス：　_varfunc
固有 gp レジスタ値：　設定しない
固有 tp レジスタ値：　設定しない

【 SCT（System Call Table）情報 】

1） タスク管理／タスク付属同期機能システム・コール情報
ユーザ処理プログラムで使用するタスク管理／タスク付属同期機能システム・コール情報として，次のシステム・コールを定義

```
sta_tsk，exd_tsk
```

2） 同期通信（セマフォ）機能システム・コール情報
ユーザ処理プログラムで使用する同期通信（セマフォ）機能システム・コール情報として，次のシステム・コールを定義

```
sig_sem，wai_sem
```

3） 同期通信（イベントフラグ）機能システム・コール情報
ユーザ処理プログラムで使用する同期通信（イベントフラグ）機能システム・コール情報として，次のシステム・コールを定義

```
cre_flg，del_flg，set_flg，wai_flg
```

4） 同期通信（メールボックス）機能システム・コール情報
ユーザ処理プログラムで使用する同期通信（メールボックス）機能システム・コール情報として，次のシステム・コールを定義

```
cre_mbx，del_mbx，snd_msg，rcv_msg
```

5） 割り込み処理管理機能システム・コール情報
ユーザ処理プログラムで使用する割り込み処理管理機能システム･コール情報として，次のシステム･コールを定義

```
ena_int
```

6） メモリ・プール管理機能システム・コール情報
ユーザ処理プログラムで使用するメモリ・プール管理機能システム・コール情報として，次のシステム・コールを定義

```
cre_mpl，del_mpl，get_blk，rel_blk
```

7） 時間管理機能システム・コール情報
ユーザ処理プログラムで使用する時間管理機能システム・コール情報として，次のシステム・コールを定義

```
act_cyc，ref_cyc
```

8） システム管理機能システム・コール情報
ユーザ処理プログラムで使用するシステム管理機能システム・コール情報として，次のシステム・コールを定義

```
viss_svc
```

図 13 － 2　システム・コンフィギュレーション・ファイルの記述例

```
------------------------------
-- リアルタイム OS 情報の記述が開始されることの宣言
------------------------------
ser_def

------------------------------
-- リアルタイム OS 情報の記述
------------------------------

-- RX シリーズ情報
rxsers      RX850PRO    V321

------------------------------
-- SIT（System Information Table）情報の記述が開始されることの宣言
------------------------------
sit_def

------------------------------
-- SIT（System Information Table）情報の記述
------------------------------

-- システム情報
cputype         V850E1
clktim          0x1
clkhdr          0x1c
defstk          0x100
intstk          0x100:SPOL0
prtstk          0x1
prtsem          0x1
prtflg          0x1
prtmbx          0x1
prtmpl          0x1

-- システム最大値情報
maxpri          0xf
maxtsk          0x2
maxsem          0x1
maxflg          0x2
maxmbx          0x3
maxmpl          0x2
maxcyc          0x1
maxsvc          0x1
maxint          0x5
maxintfactor    0x30

-- システム・メモリ情報
mem         SPOL0       syspol0     0x2000
mem         SPOL1       syspol1     0x1000
mem         UPOL0       usrpol0_0   0x7000
mem         UPOL0       usrpol0_1   0x2500
mem         UPOL1       usrpol1     0x1500

- タスク情報
tsk         0x1         TTS_RDY     0x0         0x0         TA_ASM      ¥
            _task01     0x8         TA_ENAINT   0x100:SPOL0 no_use      ¥
            no_use      0x1
tsk         TASK02      TTS_DMT     0x0         0x0         TA_HLNG     ¥
            _task02     0xf         TA_DISINT   0x100:SPOL0 no_use      ¥
            no_use      0x2

-- セマフォ情報
```

```
sem         0x1         0x0         TA_TFIFO    0xff        0xff        0x1

-- イベントフラグ情報
flg         0x1         0x0         TA_WSGL     0x0         0x1

-- メールボックス情報
mbx         0x1         0x0         TA_TFIFO    TA_MFIFO    0x1

-- 間接起動割り込みハンドラ情報
inthdr      0x14        TA_ASM      _inthdr01   no_use      no_use

-- メモリ・プール情報
mpl         0x1         0x0         TA_TPRI     0x2000:UPOL0            0x1

-- 周期起動ハンドラ情報
cyc         0x1         0x0         TA_HLNG     _cychdr01   TCY_OFF     ¥
            0x100       no_use      no_use

-- 拡張 SVC ハンドラ情報
svc         0x1         TA_HLNG     _svchdr01   no_use      no_use

-- 初期化ハンドラ情報
ini         TA_HLNG     _varfunc    no_use      no_use

------------------------------
-- SCT（System Call Table）情報の記述が開始されることの宣言
------------------------------
sct_def

------------------------------
-- SCT（System Call Table）情報の記述
------------------------------

-- タスク管理／タスク付属同期機能システム・コール情報
tsksvc      sta_tsk
tsksvc      exd_tsk

-- 同期通信（セマフォ）機能システム・コール情報
semsvc      sig_sem
semsvc      wai_sem

-- 同期通信（イベントフラグ）機能システム・コール情報
flgsvc      cre_flg
flgsvc      del_flg
flgsvc      set_flg
flgsvc      wai_flg

-- 同期通信（メールボックス）機能システム・コール情報
mbxsvc      cre_mbx
mbxsvc      del_mbx
mbxsvc      snd_msg
mbxsvc      rcv_msg

-- 割り込み処理管理機能システム・コール情報
intsvc      ena_int

-- メモリ・プール管理機能システム・コール情報
mplsvc      cre_mpl
mplsvc      del_mpl
mplsvc      get_blk
mplsvc      rel_blk

-- 時間管理機能システム・コール情報
```

```
timsvc      act_cyc
timsvc      ref_cyc

-- システム管理機能システム・コール情報
syssvc      viss_svc
```

第 14 章　コンフィギュレータ CF850 Pro

この章では，コンフィギュレータ CF850 Pro を操作して，システム・コンフィギュレーション・ファイルから情報ファイル（システム情報テーブル・ファイル，システム・コール・テーブル・ファイル，システム情報ヘッダ・ファイル）を生成する方法について解説しています。

14.1　概　　要

RX850 Pro 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）を構築をする場合，RX850 Pro に提供するデータを保持した情報が必要となります。

基本的に情報ファイルは，既定された形式のデータ羅列であるため，各種エディタを用いて記述することは可能です。しかし，情報ファイルは，記述性／可読性の面で劣ったものとなっているため，記述に際してはかなりの時間と労力を必要としてます。

そこで，RX850 Pro では，記述性／可読性の面で優れたシステム・コンフィギュレーション・ファイルから情報ファイルへと変換するユーティリティ・ツール（コンフィギュレータ CF850 Pro）を提供しています。

CF850 Pro は，システム・コンフィギュレーション・ファイルを入力ファイルとして読み込んだあと，情報ファイルを出力します。

以下に，CF850 Pro が出力する情報ファイルについて示します。

- システム情報テーブル・ファイル
  RX850 Pro が動作するうえで必要となるデータ（タスク，セマフォ，イベントフラグ等の RX850 Pro の資源情報）を保持した情報ファイルです。

- システム・コール・テーブル・ファイル
  ユーザの処理プログラム内で使用するシステム・コールの種類に関するデータを保持した情報ファイルです。

- システム情報ヘッダ・ファイル
  システム・コンフィギュレーション・ファイルに記述されたオブジェクト名（タスク名，セマフォ名，イベントフラグ名など）と ID 番号の対応付けを保持した情報ファイルです。

14.2　起動方法

14.2.1　コマンド・ラインからの起動

以下に，CF850 Pro をコマンド・ラインから起動する際の起動方法を示します。
ただし，入力例中の“C>”はコマンド・プロンプトを，“△”はスペース・キーの入力を表しています。
また，“[　]”で囲まれた起動オプションは，省略可能な起動オプションであることを表しています。

```
C>  cf850pro △ [@cmd_file] △ [-cpu △ name] △ [-devpath=path] △ [-i △ sit_file] △ [-c △ sct_file] △ [-d △ h_file]
    △ [-ni] △ [-nc] △ [-nd] △ [-ne] △ [-V] △ [-help] △ cf_file
```

次に，CF850 Pro の起動オプションを示します。

- *@cmd_file*
  コマンド・ファイル名を指定します。

  省略時　コマンド・ライン上で指定された起動オプションが有効となります。

  備考 1　コマンド・ファイル名 *cmd_file* として指定可能な文字数は，パス名を含めて 255 文字以内に限られます。

  備考 2　コマンド・ファイルについての詳細は，「14.2.3　コマンド・ファイル」を参照してください。

- -cpu △ *name*
  ターゲット・デバイスの品種指定名を指定します。

  省略時　CF850 Pro はデバイス・ファイルの読み込みを行いません。このため，システム・コンフィギュレーション･ファイル内で“デバイス･ファイルで既定されている割り込み要因名”を用いた定義が行えなくなります。

- -devpath=*path*
  -cpu △ *name* で指定されたターゲット・デバイスに対応したデバイス・ファイルを *path* フォルダから検索します。

  省略時　カレント・フォルダに対して検索処理を行います。

- -i △ *sit_file*
  CF850 Pro からの出力ファイル（システム情報テーブル・ファイル名）を指定します。

  省略時　以下に示した起動オプションが指定されていたものとして処理を行います。

  ```
  -i △ sit.s
  ```

  備考 1　出力ファイル名 *sit_file* として指定可能な文字数は，パス名を含めて 255 文字以内に限られます。

  備考 2　本起動オプションと -ni を同時に指定した場合，前者に入力した起動オプションは無効となり，後者に入力した起動オプションが有効となります。

- -c △ *sct_file*
  CF850 Pro からの出力ファイル（システム・コール・テーブル・ファイル名）を指定します。

  省略時　以下に示した起動オプションが指定されていたものとして処理を行います。

  ```
  -c △ sct.s
  ```

  備考 1　出力ファイル名 *sct_file* として指定可能な文字数は，パス名を含めて 255 文字以内に限られます。

  備考 2　本起動オプションと -nc を同 時に指定した場合，前者に入力した起動オプションは無効となり，後者に入力した起動オプションが有効となります。

- -d △ *h_file*
  CF850 Pro からの出力ファイル（システム情報ヘッダ・ファイル名）を指定します。

  省略時　*cf_file* で指定されたシステム・コンフィギュレーション・ファイル名の拡張子を“.h”に置き換えたシステム情報ヘッダ・ファイルを出力します。

  備考 1　出力ファイル名 *h_file* として指定可能な文字数は，パス名を含めて 255 文字以内に限られます。

  備考 2　本起動オプションと -nd を同時に指定した場合，前者に入力した起動オプションは無効となり，後者に入力した起動オプションが有効となります。

- -ni
  システム情報テーブル・ファイルを出力しません。

  省略時　以下に示した起動オプションが指定されていたものとして処理を行います。

  -i △ sit.s

  備考　本起動オプションと -i △ *sit_file* を同時に指定した場合，前者に入力した起動オプションは無効となり，後者に入力した起動オプションが有効となります。

- -nc
  システム・コール・テーブル・ファイルを出力しません。

  省略時　以下に示した起動オプションが指定されていたものとして処理を行います。

  -c △ sct.s

  備考　本起動オプションと -c △ *sct_file* を同時に指定した場合，前者に入力した起動オプションは無効となり，後者に入力した起動オプションが有効となります。

- -nd
  システム情報ヘッダ・ファイルを出力しません。

  省略時　*cf_file* で指定されたシステム・コンフィギュレーション・ファイル名の拡張子を“.h”に置き換えたシステム情報ヘッダ・ファイルを出力します

  備考　本起動オプションと -d △ *h_file* を同時に指定した場合，前者に入力した起動オプションは無効となり，後者に入力した起動オプションが有効となります。

- -ne
  システム情報テーブル・ファイルに対する“割り込みエントリ”の出力を抑制します。

  省略時　システム情報テーブル・ファイルに“割り込みエントリ”を出力します。

- -V
  CF850 Pro のバージョン情報を標準出力に出力します。

  省略時　バージョン情報を出力しません。

  備考　本起動オプションを指定すると，ほかの起動オプションはすべて無効となります。

- -help
  CF850 Pro の起動オプションの使い方を標準出力に出力します。

  省略時　CF850 Pro の起動オプションの使い方を出力しません。

  備考　本起動オプションを指定すると，ほかの起動オプションはすべて無効となります。

- *cf_file*
  CF850 Pro の入力ファイル（システム・コンフィギュレーション・ファイル名）を指定します。

  省略時　入力ファイルの指定は，省略できません。

  備考　入力ファイル名 *cf_file* として指定可能な文字数は，パス名を含めて 255 文字以内に限られます。

14.2.2　CubeSuite からの起動

プロパティ パネルの[システム・コンフィギュレーション・ファイル関連情報]タブで設定した内容に基づき，CubeSuite のビルド時に起動されます。

14.2.3　コマンド・ファイル

CF850 Pro 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）コメント行
行頭に # が記述された行については，コメント行として扱われます。

2）起動オプション
起動オプションと起動オプションは，スペース・コード，または，改行コードで区切って記述します。
なお，-cpu，-i，-c，-d 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は，*-xxx* 部とパラメータ部をスペース・コード，または，改行コードで区切って記述します。

備考　-devpath のパラメータ部（パス名 *path*）にスペース・コードを含むフォルダ名が存在する場合は，パラメータ部をダブル・クォーテーション " でくくる必要があります。

-devpath="Program Files¥DEV"

3）文字数制限
コマンド・ファイル内の 1 行に対する文字数制限は 4096 文字となっています。

図 14 － 1 に，コマンド・ファイルの記述例を示します。
なお，図 14 － 1 の記述例では，次に示す起動オプションが記述されています。

ターゲット・デバイス：	μPD70F3742
デバイス・ファイルの検索フォルダ：	C:¥Program Files¥NEC Electronics CubeSuite¥CubeSuite¥Device¥V850¥Devicefile
システム情報テーブル・ファイル：	sit.s（“割り込みエントリ”を含まない）
システム・コール・テーブル・ファイル：	sct.s
システム情報ヘッダ・ファイル：	sys.h
システム・コンフィギュレーション・ファイル：	sys.cfg

図 14 － 1　コマンド・ファイルの記述例

```
# Command File
-cpu
f3742
-devpath="C:¥Program Files¥NEC Electronics
CubeSuite¥CubeSuite¥Device¥V850¥Devicefile"
-i
sit.s
-c
sct.s
-d
sys.h
-ne
sys.cfg
```

14.3　コマンド入力例

以下に，CF850 Pro のコマンド入力例を示します。
なお，この例では，ターゲット・デバイスとして μPD70F3742 を指定しています。

- cf850pro -cpu f3742 -devpath=”C:¥Program Files¥NEC Electronics CubeSuite¥CubeSuite¥Device¥V850¥Devicefile” -i sitfile.s -c sctfile.s -d hfile.h -ne cffile.cfg
  システム・コンフィギュレーション・ファイル cffile.cfg をカレント・フォルダから，品種指定名 f3742 に対応したデバイス・ファイルを C:¥Program Files¥NEC Electronics CubeSuite¥CubeSuite¥Device¥V850¥Devicefile フォルダから入力ファイルとして読み込んだあと，システム情報テーブル・ファイル（割り込みエントリを含まない）sitfile.s，システム・コール・テーブル・ファイル sctfile.s，システム情報ヘッダ・ファイル hfile.h を出力します。

- cf850pro -cpu f3742 -devpath=”C:¥Program Files¥NEC Electronics CubeSuite¥CubeSuite¥Device¥V850¥Devicefile” -i sitfile.s -ne cffile.cfg
  システム・コンフィギュレーション・ファイル cffile.cfg をカレント・フォルダから，品種指定名 f3742 に対応したデバイス・ファイルを C:¥Program Files¥NEC Electronics CubeSuite¥CubeSuite¥Device¥V850¥Devicefile フォルダから入力ファイルとして読み込んだあと，システム情報テーブル・ファイル（割り込みエントリを含まない）sit.s，システム・コール・テーブル・ファイル sct.s，システム情報ヘッダ・ファイル cffile.h を出力します。

- cf850pro -cpu f3742 -devpath=”C:¥Program Files¥NEC Electronics CubeSuite¥CubeSuite¥Device¥V850¥Devicefile” -c sctfile.s -ne cffile.cfg
  システム・コンフィギュレーション・ファイル cffile.cfg をカレント・フォルダから，品種指定名 f3742 に対応したデバイス・ファイルを C:¥Program Files¥NEC Electronics CubeSuite¥CubeSuite¥Device¥V850¥Devicefile フォルダから入力ファイルとして読み込んだあと，システム情報テーブル・ファイル（割り込みエントリを含まない）sit.s，システム・コール・テーブル・ファイル sctfile.s，システム情報ヘッダ・ファイル cffile.h を出力します。

- cf850pro -cpu f3742 -devpath=”C:¥Program Files¥NEC Electronics CubeSuite¥CubeSuite¥Device¥V850¥Devicefile” -d hfile.h -ne cffile.cfg
  システム・コンフィギュレーション・ファイル cffile.cfg をカレント・フォルダから，品種指定名 f3742 に対応したデバイス・ファイルを C:¥Program Files¥NEC Electronics CubeSuite¥CubeSuite¥Device¥V850¥Devicefile フォルダから入力ファイルとして読み込んだあと，システム情報テーブル・ファイル（割り込みエントリを含まない）sit.s，システム・コール・テーブル・ファイル sct.s，システム情報ヘッダ・ファイル hfile.h を出力します。

- cf850pro -cpu f3742 -devpath=”C:¥Program Files¥NEC Electronics CubeSuite¥CubeSuite¥Device¥V850¥Devicefile” -ne cffile.cfg
  システム・コンフィギュレーション・ファイル cffile.cfg をカレント・フォルダから，品種指定名 f3742 に対応したデバイス・ファイルを C:¥Program Files¥NEC Electronics CubeSuite¥CubeSuite¥Device¥V850¥Devicefile フォルダから入力ファイルとして読み込んだあと，システム情報テーブル・ファイル（割り込みエントリを含まない）sit.s，システム・コール・テーブル・ファイル sct.s，システム情報ヘッダ・ファイル cffile.h を出力します。

- cf850pro -V
  CF850 Pro のバージョン情報を標準出力に出力します。

- cf850pro -help
  CF850 Pro の起動オプションの使い方を標準出力に出力します。

付録 A　ウインドウ・リファレンス

本付録では，CF850 Pro の起動オプションを統合開発環境プラットホーム CubeSuite から設定する際に必要となるウインドウ／パネルについて説明しています。

A.1　説　　明

以下に，ウインドウ／パネルの一覧を示します。

表 A－1　ウインドウ／パネルの一覧

ウインドウ／パネル名	機能概要
メイン・ウインドウ	CubeSuite を起動した際，最初にオープンするウインドウ
プロジェクト・ツリー パネル	プロジェクトの構成要素のツリー表示
プロパティ パネル	プロジェクト・ツリー パネルで選択しているリアルタイム OS ノード，システム・コンフィギュレーション・ファイル等について，詳細情報の表示，および設定の変更

メイン・ウインドウ

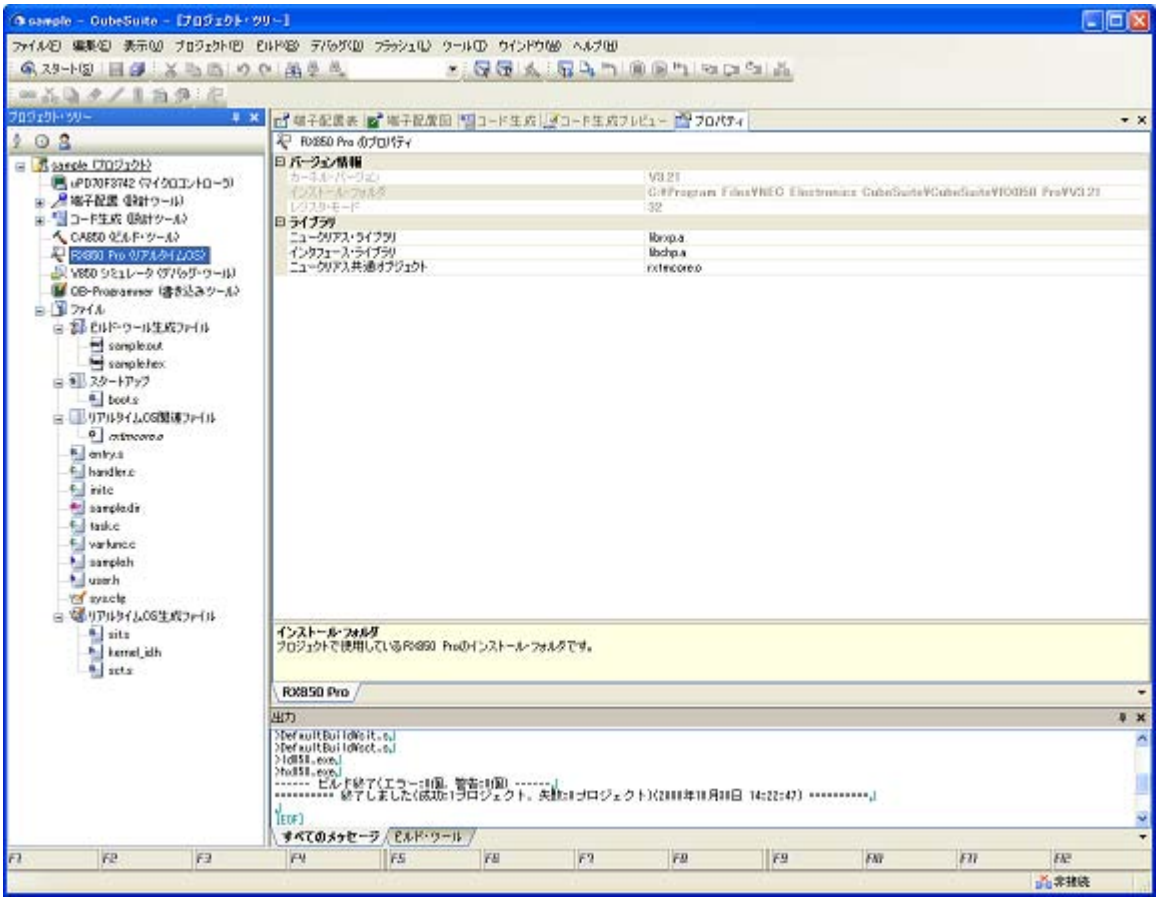
概要

CubeSuite を起動した際，最初にオープンするウインドウです。
ビルドを行う際は，本ウインドウからユーザ・プログラムの実行制御，および各パネルのオープンを行います。

なお，本ウインドウは，次の方法でオープンすることができます。

- Windows® の［スタート］→［すべてのプログラム］→［NEC Electronics CubeSuite］→［CubeSuite］を選択

表示イメージ



機能

1）メニューバー

リアルタイム OS 関連のメニューを示します。
なお，各メニューから引き出される項目は，ユーザ設定 ダイアログでカスタマイズすることができます。

- ［表示］

リアルタイム OS	リアルタイム OS の各ツールを起動するためのカスケード・メニューを表示します。
リソース情報	RD850Pro ウインドウをオープンします。 なお，本メニューは，デバッグ・ツールと切断時は無効となります。
実行解析	AZ850 ウインドウをオープンします。 なお，本メニューは，デバッグ・ツールと切断時は無効となります。

2）ツールバー

リアルタイム OS 関連のボタン群を示します。
なお，ツールバー上のボタンは，ユーザ設定 ダイアログでカスタマイズすることができます。また，同ダイアログにより，新規にツールバーを作成することもできます。

- リアルタイム OS ツールバー

	RD850Pro ウインドウをオープンします。 なお，本ボタンは，デバッグ・ツールと切断時は無効となります。

3）パネル表示エリア

以下のパネルを表示するエリアです。

- プロジェクト・ツリー パネル
- プロパティ パネル
- 出力 パネル

表示内容の詳細については，各パネルの項を参照してください。

備考　出力 パネルについての詳細は，「CubeSuite V850 ビルド編」のユーザーズ・マニュアルを参照してください。

プロジェクト・ツリー パネル

概要

プロジェクトを構成するリアルタイム OS ノード，システム・コンフィギュレーション・ファイル等の構成要素をツリー表示します。

なお，本パネルは，次の方法でオープンすることができます。

- ［表示］メニュー→［プロジェクト・ツリー］を選択

表示イメージ

機能

1）プロジェクト・ツリー エリア

プロジェクトの構成要素を以下のノードでツリー表示します。

ノード	説明
RX850 Pro(リアルタイム OS) （“リアルタイム OS ノード”と呼びます。）	使用するリアルタイム OS です。
xxx.cfg	システム・コンフィギュレーション・ファイルです。
リアルタイム OS 関連ファイル （“リアルタイム OS 関連ファイル・ノード”と呼びます。）	プロジェクト作成時に作成されるノードで，以下のオブジェクトが直下に表示されます。 - ニュークリアス共通オブジェクト（.o） 本ノード，および本ノードに表示されているオブジェクトを削除することはできません。
リアルタイム OS 生成ファイル （“リアルタイム OS 生成ファイル・ノード”と呼びます。）	システム・コンフィギュレーション・ファイル追加時に作成されるノードで，以下の情報ファイルが直下に表示されます。 - システム情報テーブル・ファイル（.s） - システム情報ヘッダ・ファイル（.h） - システム・コール・テーブル・ファイル（.s） 本ノード，および本ノードに表示されているファイルを直接削除することはできません。 システム・コンフィギュレーション・ファイルをプロジェクトから外した場合，本ノード，および本ノードに表示されているファイルは表示されなくなります。

コンテキスト・メニュー

1）リアルタイム OS ノード，リアルタイム OS 関連ファイル・ノード，リアルタイム OS 生成ファイル・ノードを選択している場合

プロパティ	選択しているノードのプロパティをプロパティ パネルに表示します。

2）ニュークリアス共通オブジェクト，システム・コンフィギュレーション・ファイル，情報ファイルを選択している場合

アセンブル	選択しているアセンブラ・ソース・ファイルをアセンブルします。 なお，本メニューは，システム情報テーブル・ファイル，システム・コール・テーブル・ファイルを選択している場合のみ表示されます。 ただし，ビルド・ツールが実行中の場合は無効となります。
開く	ファイルの拡張子に割り当てられたアプリケーションで選択しているファイルをオープンします。 なお，本メニューは，複数のファイルを選択している場合は無効となります。
内部エディタで開く ...	エディタ パネルで選択しているファイルをオープンします。 なお，本メニューは，複数のファイルを選択している場合は無効となります。
アプリケーションを指定して開く ...	プログラムから開く ダイアログをオープンし，指定したアプリケーションで選択しているファイルをオープンします。 ただし，複数のファイルを選択している場合は無効となります。
エクスプローラでフォルダを開く	選択しているファイルが存在しているフォルダをエクスプローラでオープンします。

追加	プロジェクトにファイル，カテゴリ・ノードを追加するためのカスケード・メニューを表示します。
既存のファイルを追加 ...	既存のファイルを追加 ダイアログをオープンし，選択したファイルをプロジェクトに追加します。
新しいファイルを追加 ...	ファイル追加 ダイアログをオープンし，選択した種類でファイルを作成し，プロジェクトに追加します。
新しいカテゴリを追加	選択しているファイルと同じレベルにカテゴリ・ノードを追加し，カテゴリ名が編集可能な状態になります。 なお，本メニューは，ビルド・ツールが実行中の場合，およびカテゴリのネスト数が 20 の場合は無効となります。
プロジェクトから外す	選択しているファイルをプロジェクトから外します。 ファイル自体はファイル・システム上からは削除されません。 なお，本メニューは，ビルド・ツールが実行中の場合は無効となります。
コピー	選択しているファイルをクリップ・ボードにコピーします。 ファイル名を編集中の場合は，選択している文字列をクリップ・ボードにコピーします。
貼り付け	本メニューは常に無効です。
名前の変更	選択しているファイルの名前が編集可能な状態になります。 実際のファイル名も変更されます。
プロパティ	選択しているファイルのプロパティをプロパティ パネルに表示します。

プロパティ パネル

概要

プロジェクト・ツリー パネルで選択しているリアルタイム OS ノード，システム・コンフィギュレーション・ファイル等について，カテゴリ別に詳細情報の表示，および設定の変更を行います。

なお，本パネルは，次の方法でオープンすることができます。

- プロジェクト・ツリー パネル上において，リアルタイム OS ノード，システム・コンフィギュレーション・ファイル等を選択したのち，［表示］メニュー→［プロパティ］を選択，またはコンテキスト・メニュー→［プロパティ］を選択

備考　すでにプロパティ パネルがオープンしている場合，プロジェクト・ツリー パネル上において，リアルタイム OS ノード，システム・コンフィギュレーション・ファイル等を選択することで，選択した項目の詳細情報を表示します。

表示イメージ

機能

1）対象名エリア

プロジェクト・ツリー パネルで選択しているノードの名称を表示します。
複数のノードを選択している場合，本エリアは空欄となります。

2）詳細情報表示／変更エリア

プロジェクト・ツリー パネルで選択しているリアルタイム OS ノード，システム・コンフィギュレーション・ファイル等の詳細情報を，カテゴリ別のリスト形式で表示し，設定の変更を直接行うことができるエリアです。
⊟マークは，そのカテゴリ内に含まれているすべての項目が展開表示されていることを示し，また，⊞マークは，カテゴリ内の項目が折りたたみ表示されていることを示します。展開／折りたたみ表示の切り替えは，このマークのクリック，またはカテゴリ名のダブルクリックにより行うことができます。
カテゴリ，およびそれに含まれる項目の表示内容／設定方法についての詳細は，該当するタブの項を参照してください。

3）プロパティの説明エリア

詳細情報表示／変更エリアで選択したカテゴリや項目の簡単な説明を表示します。

4）タブ選択エリア

タブを選択することにより，詳細情報を表示するカテゴリが切り替わります。
本パネルには，次のタブが存在します（各タブ上における表示内容／設定方法についての詳細は，該当するタブの項を参照してください）。

- プロジェクト・ツリー パネルでリアルタイム OS ノードを選択している場合
  - ［RX850 Pro］タブ
- プロジェクト・ツリー パネルでシステム・コンフィギュレーション・ファイルを選択している場合
  - ［システム・コンフィギュレーション・ファイル関連情報］タブ
  - ［ファイル情報］タブ
- プロジェクト・ツリー パネルでリアルタイム OS 関連ファイル・ノード，リアルタイム OS 生成ファイル・ノードを選択している場合
  - ［カテゴリ情報］タブ
- プロジェクト・ツリー パネルでニュークリアス共通オブジェクトを選択している場合
  - ［ビルド設定］タブ
  - ［ファイル情報］タブ
- プロジェクト・ツリー パネルでシステム情報テーブル・ファイル，システム・コール・テーブル・ファイルを選択している場合
  - ［ビルド設定］タブ
  - ［個別アセンブル・オプション］タブ
  - ［ファイル情報］タブ
- プロジェクト・ツリー パネルでシステム情報ヘッダ・ファイルを選択している場合
  - ［ファイル情報］タブ

備考 1　［ファイル情報］タブ，［カテゴリ情報］タブ，［ビルド設定］タブ，［個別アセンブル・オプション］タブについての詳細は，「CubeSuite V850 ビルド編」のユーザーズ・マニュアルを参照してください。

備考 2　プロジェクト・ツリー パネルで複数の構成要素を選択している場合は，その構成要素に共通するタブのみ表示されます。プロパティの値の変更は，選択している複数の構成要素に共通に反映されます。

［編集］メニュー（プロパティ パネル専用部分）

元に戻す	直前に行ったプロパティの値の編集作業を取り消します。
切り取り	プロパティの値を編集中の場合，選択している文字列を切り取ってクリップ・ボードに移動します。
コピー	選択しているプロパティの値文字列をクリップ・ボードにコピーします。
貼り付け	プロパティの値を編集中の場合，クリップ・ボードの内容を挿入します。
削除	プロパティの値を編集中の場合，選択している文字列を削除します。
すべて選択	プロパティの値を編集中の場合，選択しているプロパティの値文字列をすべて選択します。

コンテキスト・メニュー

元に戻す	直前に行ったプロパティの値の編集作業を取り消します。
切り取り	プロパティの値を編集中の場合，選択している文字列を切り取ってクリップ・ボードに移動します。

コピー	選択しているプロパティの値文字列をクリップ・ボードにコピーします。
貼り付け	プロパティの値を編集中の場合，クリップ・ボードの内容を挿入します。
削除	プロパティの値を編集中の場合，選択している文字列を削除します。
すべて選択	プロパティの値を編集中の場合，選択しているプロパティの値文字列をすべて選択します。
デフォルトに戻す	選択している項目の設定値をプロジェクトに設定しているデフォルト値に戻します。 ただし，［個別アセンブル・オプション］タブにおいては，全体オプションの設定値に戻します。
すべてデフォルトに戻す	現在表示しているタブの設定値をすべてプロジェクトに設定しているデフォルト値に戻します。 ただし，［個別アセンブル・オプション］タブにおいては，全体オプションの設定値に戻します。

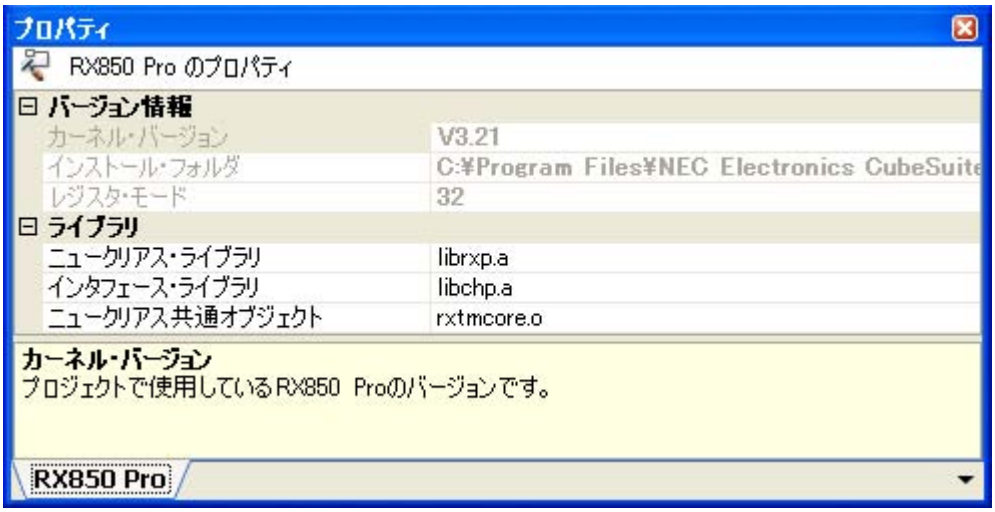
［RX850 Pro］タブ

概要

本タブでは，使用する RX850 Pro に対して，次に示すカテゴリごとに詳細情報の表示を行います。

- バージョン情報
- ライブラリ

表示イメージ



機能

1）［バージョン情報］

RX850 Pro のバージョンに関する詳細情報の表示を行います。

カーネル・バージョン	使用する RX850 Pro のバージョンを表示します。 なお，バージョンはプロジェクト作成時に確定するため，変更することはできません。	
	デフォルト	*使用する RX850 Pro のバージョン*
	変更方法	変更不可
インストール・フォルダ	使用する RX850 Pro がインストールされているフォルダを絶対パスで表示します。	
	デフォルト	*使用する RX850 Pro がインストールされているフォルダ*
	変更方法	変更不可
レジスタ・モード	プロジェクトで設定しているレジスタ・モードを表示します。 ビルド・ツールの［レジスタ・モードの選択］プロパティで選択しているレジスタ・モードと同じ値が表示されます。	
	デフォルト	*ビルド・ツールでのプロパティで選択しているレジスタ・モード*
	変更方法	変更不可

備考　ビルド・ツールの［レジスタ・モードの選択］プロパティで，26 レジスタ・モード，または 22 レジスタ・モードを選択した場合は，エラーとなります。そのままビルドを実行し，ロード・モジュールを作成することは可能ですが，ワーニングが発生します。

2）［ライブラリ］

ライブラリに関する詳細情報の表示，および設定の変更を行います。

ニュークリアス・ライブラリ	アプリケーションのリンク時に参照するニュークリアス・ライブラリを選択します。		
	デフォルト	librxp.a	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	librxp.a	librxp.a（rel_blk 発行時にメモリ・ブロックの先頭 4 バイトのゼロクリアが必要なニュークリアス・ライブラリ）を参照します。
		librxpm.a	librxpm.a（rel_blk 発行時にメモリ・ブロックの先頭4バイトのゼロクリアが不要なニュークリアス・ライブラリ）を参照します。
インタフェース・ライブラリ	アプリケーションのリンク時に参照するインタフェース・ライブラリを選択します。		
	デフォルト	libchp.a	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	libchp.a	libchp.a（パラメータ・チェックありのライブラリ）を参照します。すべてのエラー・コードを検出します。
		libncp.a	libncp.a（パラメータ・チェックなしのライブラリ）を参照します。必要なエラー・コードのみ検出します。
ニュークリアス共通オブジェクト	ユーザ・アプリケーションにリンクするニュークリアス共通オブジェクトを選択します。 オブジェクトを変更すると，プロジェクト・ツリーに表示しているオブジェクト名も変更します。		
	デフォルト	rxtmcore.o	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	rxtmcore.o	rxtmcore.o（周期ハンドラ内でタイマ割り込みより優先度の高い割り込みのみ受け付け可能）をリンクします。
		rxcore.o	rxcore.o（周期ハンドラ内ですべての割り込みを受け付け可能）をリンクします。

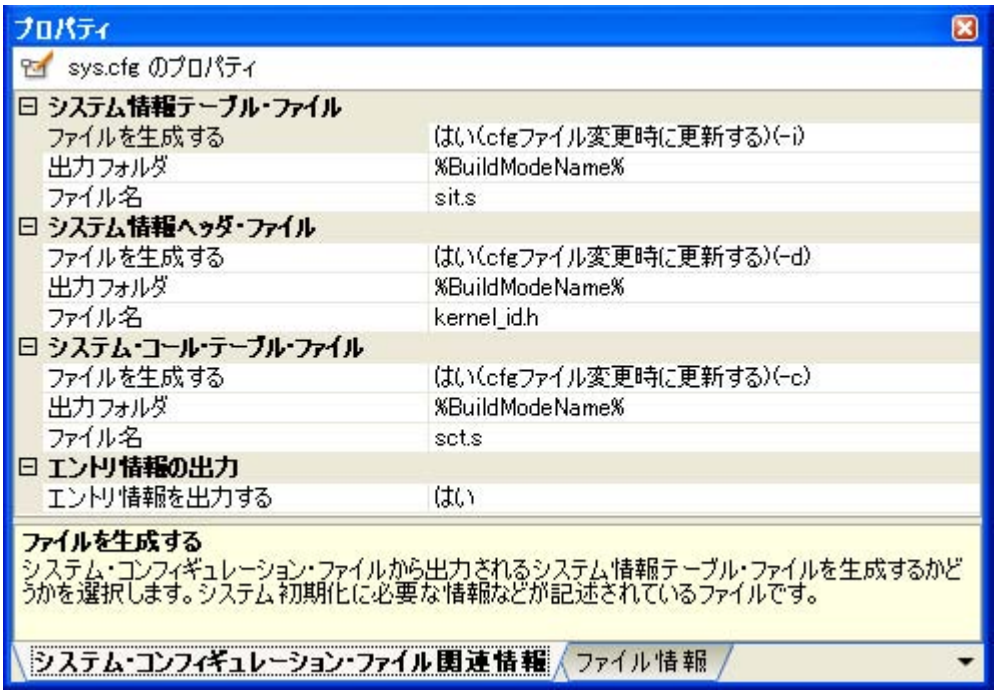
［システム・コンフィギュレーション・ファイル関連情報］タブ

概要

本タブでは，使用するシステム・コンフィギュレーション・ファイルに対して，次に示すカテゴリごとに詳細情報の表示，および設定の変更を行います。

- システム情報テーブル・ファイル
- システム情報ヘッダ・ファイル
- システム・コール・テーブル・ファイル
- エントリ情報の出力

表示イメージ



機能

1)［システム情報テーブル・ファイル］

システム情報テーブル・ファイルに関する詳細情報の表示，および設定の変更を行います。

<table>
<tr><td rowspan="6">ファイルを生成する</td><td colspan="3">システム情報テーブル・ファイルを生成するかどうか，およびシステム・コンフィギュレーション・ファイルを変更した場合にシステム情報テーブル・ファイルを更新するかどうかを選択します。</td></tr>
<tr><td>デフォルト</td><td colspan="2">はい (.cfg ファイル変更時に更新する)(-i)</td></tr>
<tr><td>変更方法</td><td colspan="2">ドロップダウン・リストによる選択</td></tr>
<tr><td rowspan="3">指定可能値</td><td>はい (.cfg ファイル変更時に更新する)(-i)</td><td>システム情報テーブル・ファイルを新規生成し，プロジェクト・ツリーに表示します。
システム情報テーブル・ファイルがすでに存在するときにシステム・コンフィギュレーション・ファイルを変更した場合は，システム情報テーブル・ファイルを更新します。</td></tr>
<tr><td>はい (.cfg ファイル変更時に更新しない)(-ni)</td><td>システム・コンフィギュレーション・ファイルを変更しても，システム情報テーブル・ファイルを更新しません。
システム情報テーブル・ファイルが存在しないときに本項目を選択した場合は，ビルド時にエラーとなります。</td></tr>
<tr><td>いいえ (プロジェクトに登録しない)(-ni)</td><td>システム情報テーブル・ファイルの生成を行わず，プロジェクト・ツリーにも表示しません。
システム情報テーブル・ファイルが存在するときに本項目を選択しても，ファイル自体の削除は行いません。</td></tr>
<tr><td rowspan="4">出力フォルダ</td><td colspan="3">システム情報テーブル・ファイルを出力するフォルダを指定します。
相対パスで指定した場合は，プロジェクト・フォルダを基点とします。
絶対パスで指定した場合は，プロジェクト・フォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。
埋め込みマクロとして次のマクロ名があります。
%BuildModeName% : ビルド・モード名に置換します。
空欄にした場合は，マクロ名“%BuildModeName%”が表示されます。
なお，本プロパティは，［ファイルを生成する］プロパティで［いいえ (プロジェクトに登録しない)(-ni)］を選択した場合は表示されません。</td></tr>
<tr><td>デフォルト</td><td colspan="2">%BuildModeName%</td></tr>
<tr><td>変更方法</td><td colspan="2">テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，フォルダの参照 ダイアログによる編集</td></tr>
<tr><td>指定可能値</td><td colspan="2">247 文字までの文字列</td></tr>
<tr><td rowspan="4">ファイル名</td><td colspan="3">システム情報テーブル・ファイル名を指定します。
ファイル名を変更すると，プロジェクト・ツリーに表示しているファイル名も変更します。
拡張子は“.s”を指定してください。拡張子が異なる場合や省略した場合は，“.s”が自動的に付加されます。
なお，本プロパティは，［ファイルを生成する］プロパティで［いいえ (プロジェクトに登録しない)(-ni)］を選択した場合は表示されません。</td></tr>
<tr><td>デフォルト</td><td colspan="2">sit.s</td></tr>
<tr><td>変更方法</td><td colspan="2">テキスト・ボックスによる直接入力</td></tr>
<tr><td>指定可能値</td><td colspan="2">259 文字までの文字列</td></tr>
</table>

2)［システム情報ヘッダ・ファイル］

システム情報ヘッダ・ファイルに関する詳細情報の表示，および設定の変更を行います。

<table>
<tr><td rowspan="6">ファイルを生成する</td><td colspan="3">システム情報ヘッダ・ファイルを生成するかどうか，およびシステム・コンフィギュレーション・ファイルを変更した場合にシステム情報ヘッダ・ファイルを更新するかどうかを選択します。</td></tr>
<tr><td>デフォルト</td><td colspan="2">はい (.cfg ファイル変更時に更新する)(-d)</td></tr>
<tr><td>変更方法</td><td colspan="2">ドロップダウン・リストによる選択</td></tr>
<tr><td rowspan="3">指定可能値</td><td>はい (.cfg ファイル変更時に更新する)(-d)</td><td>システム情報ヘッダ・ファイルを生成し，プロジェクト・ツリーに表示します。
システム情報ヘッダ・ファイルがすでに存在するときにシステム・コンフィギュレーション・ファイルを変更した場合は，システム情報ヘッダ・ファイルを更新します。</td></tr>
<tr><td>はい (.cfg ファイル変更時に更新しない)(-nd)</td><td>システム・コンフィギュレーション・ファイルを変更しても，システム情報ヘッダ・ファイルを更新しません。
システム情報ヘッダ・ファイルが存在しないときに本項目を選択した場合は，ビルド時にエラーとなります。</td></tr>
<tr><td>いいえ (プロジェクトに登録しない)(-nd)</td><td>システム情報ヘッダ・ファイルの生成を行わず，プロジェクト・ツリーにも表示しません。
システム情報ヘッダ・ファイルが存在するときに本項目を選択しても，ファイル自体の削除は行いません。</td></tr>
<tr><td rowspan="4">出力フォルダ</td><td colspan="3">システム情報ヘッダ・ファイルを出力するフォルダを指定します。
相対パスで指定した場合は，プロジェクト・フォルダを基点とします。
絶対パスで指定した場合は，プロジェクト・フォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。
埋め込みマクロとして次のマクロ名があります。
%BuildModeName% : ビルド・モード名に置換します。
空欄にした場合は，マクロ名“%BuildModeName%”が表示されます。
なお，本プロパティは，［ファイルを生成する］プロパティで［いいえ (プロジェクトに登録しない)(-nd)］を選択した場合は表示されません。</td></tr>
<tr><td>デフォルト</td><td colspan="2">%BuildModeName%</td></tr>
<tr><td>変更方法</td><td colspan="2">テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，フォルダの参照 ダイアログによる編集</td></tr>
<tr><td>指定可能値</td><td colspan="2">247 文字までの文字列</td></tr>
<tr><td rowspan="4">ファイル名</td><td colspan="3">システム情報ヘッダ・ファイル名を指定します。
ファイル名を変更すると，プロジェクト・ツリーに表示しているファイル名も変更します。
拡張子は“.h”を指定してください。拡張子が異なる場合や省略した場合は，“.h”が自動的に付加されます。
なお，本プロパティは，［ファイルを生成する］プロパティで［いいえ (プロジェクトに登録しない)(-nd)］を選択した場合は表示されません。</td></tr>
<tr><td>デフォルト</td><td colspan="2">kernel_id.h</td></tr>
<tr><td>変更方法</td><td colspan="2">テキスト・ボックスによる直接入力</td></tr>
<tr><td>指定可能値</td><td colspan="2">259 文字までの文字列</td></tr>
</table>

3）［システム・コール・テーブル・ファイル］

システム・コール・テーブル・ファイルに関する詳細情報の表示，および設定の変更を行います。

<table>
<tr><td rowspan="6">ファイルを生成する</td><td colspan="3">システム・コール・テーブル・ファイルを生成するかどうか，およびシステム・コンフィギュレーション・ファイルを変更した場合にシステム・コール・テーブル・ファイルを更新するかどうかを選択します。</td></tr>
<tr><td>デフォルト</td><td colspan="2">はい (.cfg ファイル変更時に更新する)(-c)</td></tr>
<tr><td>変更方法</td><td colspan="2">ドロップダウン・リストによる選択</td></tr>
<tr><td rowspan="3">指定可能値</td><td>はい (.cfg ファイル変更時に更新する)(-c)</td><td>システム・コール・テーブル・ファイルを生成し，プロジェクト・ツリーに表示します。
システム・コール・テーブル・ファイルがすでに存在するときにシステム･コンフィギュレーション･ファイルを変更した場合は，システム・コール・テーブル・ファイルを更新します。</td></tr>
<tr><td>はい (.cfg ファイル変更時に更新しない)(-nc)</td><td>システム・コンフィギュレーション・ファイルを変更しても，システム・コール･テーブル･ファイルを更新しません。
システム・コール・テーブル・ファイルが存在しないときに本項目を選択した場合は，ビルド時にエラーとなります。</td></tr>
<tr><td>いいえ (プロジェクトに登録しない)(-nc)</td><td>システム・コール・テーブル・ファイルの生成を行わず，プロジェクト・ツリーにも表示しません。
システム・コール・テーブル・ファイルが存在するときに本項目を選択しても，ファイル自体の削除は行いません。</td></tr>
<tr><td rowspan="4">出力フォルダ</td><td colspan="3">システム・コール・テーブル・ファイルを出力するフォルダを指定します。
相対パスで指定した場合は，プロジェクト・フォルダを基点とします。
絶対パスで指定した場合は，プロジェクト・フォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。
埋め込みマクロとして次のマクロ名があります。
%BuildModeName%：ビルド・モード名に置換します。
空欄にした場合は，マクロ名“%BuildModeName%”が表示されます。
なお，本プロパティは，［ファイルを生成する］プロパティで［いいえ (プロジェクトに登録しない)(-nc)］を選択した場合は表示されません。</td></tr>
<tr><td>デフォルト</td><td colspan="2">%BuildModeName%</td></tr>
<tr><td>変更方法</td><td colspan="2">テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，フォルダの参照 ダイアログによる編集</td></tr>
<tr><td>指定可能値</td><td colspan="2">247 文字までの文字列</td></tr>
<tr><td rowspan="4">ファイル名</td><td colspan="3">システム・コール・テーブル・ファイルを指定します。
ファイル名を変更すると，プロジェクト・ツリーに表示しているファイル名も変更します。
拡張子は“.s”を指定してください。拡張子が異なる場合や省略した場合は，“.s”が自動的に付加されます。
なお，本プロパティは，［ファイルを生成する］プロパティで［いいえ (プロジェクトに登録しない)(-nc)］を選択した場合は表示されません。</td></tr>
<tr><td>デフォルト</td><td colspan="2">sct.s</td></tr>
<tr><td>変更方法</td><td colspan="2">テキスト・ボックスによる直接入力</td></tr>
<tr><td>指定可能値</td><td colspan="2">259 文字までの文字列</td></tr>
</table>

4)［エントリ情報の出力］

エントリ情報の出力に関する詳細情報の表示，および設定の変更を行います。

<table>
<tr><td rowspan="4">エントリ情報を出力する</td><td colspan="3">エントリ情報をシステム情報テーブル・ファイルに出力するかどうかを選択します。</td></tr>
<tr><td>デフォルト</td><td colspan="2">はい</td></tr>
<tr><td>変更方法</td><td colspan="2">ドロップダウン・リストによる選択</td></tr>
<tr><td>指定可能値</td><td>はい</td><td>エントリ情報をシステム情報テーブル・ファイルに出力します。
ただし，［システム情報テーブル・ファイル］カテゴリの［ファイルを生成する］プロパティで［いいえ (プロジェクトに登録しない)(-ni)］を選択した場合は，出力しません。</td></tr>
<tr><td>いいえ (-ne)</td><td>エントリ情報のシステム情報テーブル・ファイルへの出力を行いません。</td></tr>
</table>

付録 B　プログラミングのために

本付録では, NEC エレクトロニクス製 V850 マイクロコントローラ用 C コンパイラ CA850 を使用した際の処理プログラムの記述方法について説明します。

B.1　概　　要

RX850 Pro では，処理プログラムを用途別に次のように区別しています。

- タスク
  RX850 Pro の管理下で実行可能な処理プログラムの最小単位です。

- 直接起動割り込みハンドラ
  割り込みが発生した際，RX850 Pro を介在させることなく起動される割り込み処理専用ルーチンです。
  RX850 Pro が介在しないため，ハンドラ内でシステム・コールを発行することはできませんが，高速な応答性が期待されます。

- 間接起動割り込みハンドラ
  割り込みが発生した際，RX850 Pro に割り込み前処理（レジスタの退避処理，スタックの切り替え処理など）を行わせたあと起動される割り込み処理専用ルーチンです。
  RX850 Pro による割り込み前処理が行われているため，直接起動割り込みハンドラに比べて応答性の面では劣りますが，ハンドラ内でシステム・コールを発行できるといった利点を有しています。

- 周期起動ハンドラ
  一定の起動時間に達した際，ただちに起動される周期処理専用ルーチンであり，タスクとは独立したものとして扱われます。このため，起動時間に達した際には，システム内で最高優先度を持つタスクが実行中であっても，その処理は中断され，周期起動ハンドラに制御が移ります。
  なお，周期起動ハンドラは，ユーザが記述する周期的な処理プログラムの中で，実行開始までのオーバヘッドが最も小さい処理プログラムです。

- 拡張 SVC ハンドラ
  ユーザが拡張システム・コールとして登録した関数です。
  なお，これらの処理プログラムは，一般的に，または RX850 Pro を使用する上での約束ごとなどにより，それぞれに基本型があります。

B.2　キー・ワード

コンフィギュレータでは，次に示す文字列をキー・ワードとして予約しています。したがって，これらの文字列をほかの用途に使用することは禁止されています。

```
clkhdr，clktim，cyc，defstk，flg，flgsvc，ini，inthdr，intstk，intsvc，maxcyc，maxflg，
maxint，maxintfactor，maxmbx，maxmpl，maxpri，maxsem，maxsvc，maxtsk，mbx，mbxsvc，mem，
mpl，mplsvc，no_use，prtflg，prtmbx，prtmpl，prtsem，prttsk，RX850PRO，rxsers，sct_def，
sem，semsvc，ser_def，sit_def，SPOL0，SPOL1，svc，syssvc，TA_ASM，TA_DISINT，TA_ENAINT，
TA_HLNG，TA_MFIFO，TA_MPRI，TA_TFIFO，TA_TPRI，TA_WMUL，TA_WSGL，TCY_OFF，TCY_ON，timsvc，
tsk，tsksvc，TTS_DMT，TTS_RDY，UPOL0，UPOL1
```

B.3　予 約 語

RX850 Pro では，次に示す文字列を外部シンボルとして予約しています。したがって，これらの文字列をほかの用途に使用することは禁止されています。

```
_x_，_f_，_e_，_rx_
```

備考　これらの文字列を使用することが禁止されるのは，単一のロード・モジュールを作成した場合です。
RX850 Pro とアプリケーションが分離しているロード・モジュールを作成している場合は，これらの文字列から始まるシンボルを使用しても問題ありません。

B.4　処理プログラム起動時のハードウエア状態

以下に，処理プログラム起動時のハードウエア状態（sp，tp，gp，ep，psw の ID ビット）を示します。

表 B － 1　ハードウエア状態（タスク）

	タスク
スタック・ポインタ（sp）	タスク・スタック （タスク生成時に指定したプール領域内の値）
テキスト・ポインタ（tp）	不定 （_rx_start 呼び出し時の値）
グローバル・ポインタ（gp）	タスク生成時に指定した値 （no_use の場合は不定値）
エレメント・ポインタ（ep）	タスク生成時に指定した値 （no_use の場合は不定値）
割り込み状態（psw の ID ビット）	タスク生成時に指定した値 （デフォルトは割り込み許可）

表 B － 2　ハードウエア状態（直接起動割り込みハンドラ）

	直接起動割り込みハンドラ
スタック・ポインタ（sp）	割り込み発生時のスタック
テキスト・ポインタ（tp）	不定
グローバル・ポインタ（gp）	割り込み発生時の値
エレメント・ポインタ（ep）	割り込み発生時の値
割り込み状態（psw の ID ビット）	割り込み禁止

表 B － 3　ハードウエア状態（間接起動割り込みハンドラ）

	間接起動割り込みハンドラ
スタック・ポインタ（sp）	システム・スタック （システム・スタック定義時に指定したプール領域内の値）
テキスト・ポインタ（tp）	不定 （__rx_start 呼び出し時の値）
グローバル・ポインタ（gp）	間接起動割り込みハンドラ生成時に指定した値 （no_use の場合は不定）
エレメント・ポインタ（ep）	間接起動割り込みハンドラ生成時に指定した値 （no_use の場合は不定）
割り込み状態（psw の ID ビット）	割り込み禁止

表 B － 4　ハードウエア状態（周期起動ハンドラ）

	周期起動ハンドラ
スタック・ポインタ（sp）	システム・スタック （システム・スタック定義時に指定したプール領域内の値）
テキスト・ポインタ（tp）	不定 （__rx_start 呼び出し時の値）
グローバル・ポインタ（gp）	周期起動ハンドラ生成時に指定した値 （no_use の場合は不定）
エレメント・ポインタ（ep）	周期起動ハンドラ生成時に指定した値 （no_use の場合は不定）
割り込み状態（psw の ID ビット）	割り込み禁止

表 B － 5　ハードウエア状態（拡張 SVC ハンドラ）

	拡張 SVC ハンドラ
スタック・ポインタ（sp）	拡張 SVC ハンドラ呼び出し時のスタック
テキスト・ポインタ（tp）	不定 （__rx_start 呼び出し時の値）
グローバル・ポインタ（gp）	拡張 SVC ハンドラ生成時に指定した値 （no_use の場合は不定）
エレメント・ポインタ（ep）	拡張 SVC ハンドラ生成時に指定した値 （no_use の場合は不定）
割り込み状態（psw の ID ビット）	拡張 SVC ハンドラ呼び出し時の状態

表 B － 6　ハードウエア状態（初期化ハンドラ）

	初期化ハンドラ
スタック・ポインタ（sp）	システム・スタック （システム・スタック定義時に指定したプール領域内の値）
テキスト・ポインタ（tp）	不定 （__rx_start 呼び出し時の値）
グローバル・ポインタ（gp）	初期化ハンドラ生成時に指定した値 （no_use の場合は不定）
エレメント・ポインタ（ep）	初期化ハンドラ生成時に指定した値 （no_use の場合は不定）
割り込み状態（psw の ID ビット）	割り込み禁止

B.5　タ ス ク

タスクを C 言語で記述する場合，プラグマ指令による関数宣言を行ったのち，INT 型の引数を 1 つ持った void 型の関数として記述します。
なお，引数（stacd）には，コンフィギュレーション時にタスク情報で指定された起動コード，または sta_tsk 発行時に指定された起動コードが設定されます。
次に，タスクの記述形式（C 言語）を示します。

図 B － 1　タスクの記述形式（C 言語）

```
#include    <stdrx85p.h>

#pragma     rtos_task   func_task

void
func_task ( INT stacd )
{
            /* タスク func_task の本体処理 */
            ..........................
            ..........................

            /* タスク func_task の終了 */
            ext_tsk ( );
}
```

備考　プラグマ指令による関数宣言についての詳細は，「CubeSuite V850 コーディング編」のユーザーズ・マニュアルを参照してください。

また，タスクをアセンブリ言語で記述する場合は，CA850 の関数呼び出し規約に従った関数として記述します。
なお，引数（r6 レジスタ）には，コンフィギュレーション時にタスク情報で指定された起動コード，または sta_tsk 発行時に指定された起動コードが設定されます。
次に，タスクの記述形式（アセンブリ言語）を示します。

図 B － 2　タスクの記述形式（アセンブリ言語）

```
.include    "stdrx85p.inc"

            .text
            .align      4
            .globl      _func_task
_func_task :
            # タスク func_task の本体処理
            ..........................
            ..........................

            # タスク func_task の終了
            jr          _ext_tsk
```

B.6　直接起動割り込みハンドラ

B.6.1　推奨

直接起動割り込みハンドラを記述する場合は, C 言語（プラグマを使用するなど）, またはアセンブリ言語で記述します。
詳細は V850 マイクロコントローラ ハードウエア編や,「CubeSuite V850 コーディング編」のユーザーズ・マニュアルを参照してください。

B.6.2　間接起動割り込みハンドラと同等機能を実現する場合

直接起動割り込みハンドラで間接起動割り込みハンドラと同等機能を実現する（システム・コール呼び出しが可能など）ことも可能です。
直接起動割り込みハンドラで間接起動割り込みハンドラと同等機能を実現する場合は，アセンブリ言語を使用します。ただし，処理本体を C 言語で記述し，Jarl 命令で呼び出す形を使うこともできます。
直接起動割り込みハンドラで間接起動割り込みハンドラと同等機能を実現する場合は，その処理前にレジスタの退避，処理後に復帰の処理を行う必要があります。
しかし，RX850 Pro ではレジスタの退避，復帰処理を記述したマクロを用意しており，アセンブリ言語でのハンドラの記述におけるユーザ負担を軽減しています。
ただし，直接起動割り込みハンドラで間接起動割り込みハンドラと同等機能を実現した場合，機能は間接起動割り込みハンドラと同等になりますが，応答性も間接起動割り込みハンドラと同等になります。また，定義方法が間接起動割り込みハンドラより煩雑です。
つまり，間接起動割り込みハンドラと比べて，機能的，性能的なメリットがなく，定義が煩雑というデメリットがあります。
このため，割り込みハンドラ上でシステム・コールなどの RX850 Pro 機能を使用したい場合，通常は間接起動割り込みハンドラを使用することを推奨します。
次に，直接起動割り込みハンドラの記述形式（アセンブリ言語）を示します。

図 B － 3　直接起動割り込みハンドラの記述形式（アセンブリ言語）

```
.include    "stdrx85p.inc"

            /* 割り込みエントリ */
            .section    "int_name", text
            jr          _func_inthdr

            .text
            .align      4
            .globl      _func_inthdr
_func_inthdr :
            /* レジスタの退避，スタックの切り替え */
            RTOS_IntEntry

            /* 直接起動割り込みハンドラの本体処理 */
            .extern     _inthdr_body
            jarl        _inthdr_body, lp

            /*r10：ハンドラ復帰後に起床するタスクの ID*/
            /* スタックの切り替え，レジスタの復帰 */
            /* 直接起動割り込みハンドラからの復帰と指定したタスクの起床 */
            RTOS_IntReturnWakeup    r10
```

```
#include    <stdrx85p.h>

ID
inthdr_body ( void )
{
            __asm ( "mov #__tp_TEXT, tp" );
            __asm ( "mov #__gp_DATA, gp" );

            /* 直接起動割り込みハンドラ func_inthdr の本体処理 */

            ..........................
            ..........................

            /* 直接起動割り込みハンドラ func_inthdr 本体からの復帰 */
            return ( tskid );
}
```

まず，ハンドラ・アドレスに割り込みハンドラのエントリ処理（jr 命令）を記述します。この例では，2，3 行目がこれにあたります。

次に，割り込みハンドラ処理の本体についてです。

マクロ RTOS_IntEntry では，RX850 Pro に対してハンドラ開始の通知，テンポラリ・レジスタと lp の退避，そしてスタックの切り替えなどを行います。そのあと，これら以外のレジスタ（r20 ～ r30）の退避を行い，ハンドラ本体部へと処理が移ります。記述例では，ハンドラ本体部である C の関数 inthdr_body を呼び出しています。ハンドラ本体処理を行う前に，ハンドラで使用する tp（テキスト・ポインタ）と gp（グローバル・ポインタ）の設定を行います。「6.3　直接起動割り込みハンドラ」に説明があるように，直接起動割り込みハンドラで使用する tp と gp の値が不定になるためです。この設定は，アセンブリ言語で書かなくてはならないので，C 言語でハンドラ本体を書くときは，例のように __asm 命令を使用するか，# pragma asm ～ pragma endasm 指令を使って記述してください。ハンドラ本体部では，ユーザーズ・マニュアルに記載されている「ハンドラから発行可能なシステム・コール」が発行可能です。

ハンドラの発行処理が終了したら，ユーザが退避したレジスタの復帰と割り込みハンドラからの復帰を行います。割り込み復帰後に指定したタスクを起床させるときは，レジスタ r10 に起床させるタスクの ID をセットする必要があります。記述例では，inthdr_body から復帰するときに戻り値としてタスク ID を返しており，その値が r10 にコピーされます。これは，CA850 がこのようなコードを出力しています。

簡単な処理のみを行って復帰する場合，reti 命令を使うこともできます。そのときは，命令を発行する前に，レジスタの復帰処理を行う必要があります。

備考　割り込みが発生した際にプロセッサが制御を移すハンドラ･アドレスに対して直接起動割り込みハンドラへの分岐命令などを設定する必要があります。図 B － 3 内の .section 疑似命令がこれにあたります。.section 疑似命令についての詳細は，「CubeSuite V850 コーディング編」のユーザーズ・マニュアルを参照してください。なお，“int_name”には，デバイス・ファイルで定義されている割り込み要求名を指定します。

B.7　間接起動割り込みハンドラ

間接起動割り込みハンドラを C 言語で記述する場合，引数を持たない ID 型の関数として記述します。
次に，間接起動割り込みハンドラの記述形式（C 言語）を示します。

図 B － 4　間接起動割り込みハンドラの記述形式（C 言語）

```
#include    <stdrx85p.h>

ID
func_inthdr ( void )
{
            /* 間接起動割り込みハンドラ func_inthdr の本体処理 */
            ..........................
            ..........................

            /* 間接起動割り込みハンドラ func_inthdr からの復帰 */
            return ( TSK_NULL );
}
```

備考　間接起動割り込みハンドラは，ニュークリアス内の割り込み前処理から呼び出されるサブルーチンですが，間接起動割り込みハンドラを記述する場合，割り込みが発生した際にプロセッサへ制御を移すハンドラ・アドレスに対して，間接起動割り込みハンドラへの分岐命令などを設定する必要があります。これはアセンブリ言語で記述する必要があります。
ただし，RX850 Pro では，この分岐命令として記述すべき処理をマクロで提供しているので，それを使用します。たとえば，INTP100（アドレス：0x100）というマスカブル割り込みを間接起動割り込みハンドラとして使用するには，

```
  .section    "INTP100"
  RTOS_IntEntry_Indirect
```

と記述します。
なお，クロック割り込みも間接起動割り込みハンドラとして扱われるので，同様の記述が必要です。

また，間接起動割り込みハンドラをアセンブリ言語で記述する場合は，CA850 の関数呼び出し規約に従った関数として記述します。

次に，間接起動割り込みハンドラの記述形式（アセンブリ言語）を示します。

図 B － 5　間接起動割り込みハンドラの記述形式（アセンブリ言語）

```
.include    "stdrx85p.inc"

            .text
            .align      4
            .globl      _func_inthdr

_func_inthdr :
            # 間接起動割り込みハンドラ func_inthdr の本体処理
            ......................
            ......................

            # 間接起動割り込みハンドラ func_inthdr からの復帰
            mov         TSK_NULL, r10
            jmp         [lp]
```

備考　間接起動割り込みハンドラは，ニュークリアス内の割り込み前処理から呼び出されるサブルーチンですが，間接起動割り込みハンドラを記述する場合，割り込みが発生した際にプロセッサへ制御を移すハンドラ・アドレスに対して，間接起動割り込みハンドラへの分岐命令などを設定する必要があります。これはアセンブリ言語で記述する必要があります。
ただし，RX850 Pro では，この分岐命令として記述すべき処理をマクロで提供しているので，それを使用します。たとえば，INTP100（アドレス：0x100）というマスカブル割り込みを間接起動割り込みハンドラとして使用するには，

```
.section    "INTP100"
RTOS_IntEntry_Indirect
```

と記述します。
なお，クロック割り込みも間接起動割り込みハンドラとして扱われるので，同様の記述が必要です。

B.8　周期起動ハンドラ

周期起動ハンドラを C 言語で記述する場合，引数を持たない void 型の関数として記述します。
次に，周期起動ハンドラの記述形式（C 言語）を示します。

図 B － 6　周期起動ハンドラの記述形式（C 言語）

```
#include    <stdrx85p.h>

void
func_cychdr ( void )
{
            /* 周期起動ハンドラ func_cychdr の本体処理 */
            ..........................
            ..........................

            /* 周期起動ハンドラ func_cychdr からの復帰 */
            return;
}
```

備考　　周期起動ハンドラは，ニュークリアス内のシステム・クロック処理から呼び出されるサブルーチンです。

また，周期起動ハンドラをアセンブリ言語で記述する場合は，CA850 の関数呼び出し規約に従った関数として記述します。
次に，周期起動ハンドラの記述形式（アセンブリ言語）を示します。

図 B － 7　周期起動ハンドラの記述形式（アセンブリ言語）

```
.include    "stdrx85p.inc"

            .text
            .align      4
            .globl      _func_cychdr
_func_cychdr :
            # 周期起動ハンドラ func_cychdr の本体処理
            ..........................
            ..........................

            # 周期起動ハンドラ func_cychdr からの復帰
            jmp         [lp]
```

備考　　周期起動ハンドラは，ニュークリアス内のシステム・クロック処理から呼び出されるサブルーチンです。

B.9　拡張 SVC ハンドラ

拡張 SVC ハンドラを C 言語で記述する場合，INT 型の関数として記述します。
次に，拡張 SVC ハンドラの記述形式（C 言語）を示します。

図 B － 8　拡張 SVC ハンドラの記述形式（C 言語）

```
#include    <stdrx85p.h>

INT
func_svchdr ( VW prm1, VW prm2, VW prm3 )
{
            int         ret;

            /* 拡張 SVC ハンドラ func_svchdr の本体処理 */
            ..........................
            ..........................

            /* 拡張 SVC ハンドラ func_svchdr からの復帰 */
            return ( INT ret );
}
```

また，拡張 SVC ハンドラをアセンブリ言語で記述する場合は，CA850 の関数呼び出し規約に従った関数として記述します。
次に，拡張 SVC ハンドラの記述形式（アセンブリ言語）を示します。

図 B － 9　拡張 SVC ハンドラの記述形式（アセンブリ言語）

```
.include    "stdrx85p.inc"

            .text
            .align      4
            .globl      _func_svchdr
_func_svchdr :
            # 拡張 SVC ハンドラ func_svchdr の本体処理
            ..........................
            ..........................

            # 拡張 SVC ハンドラ func_svchdr からの復帰
            mov         ret, r10
            jmp         [lp]
```

付録 C　メモリとその容量の見積もり

本付録では，RX850 Pro におけるメモリ（RAM）の管理と，使用する容量について説明します。

C.1　SPOL と UPOL

RX850 Pro で使用する RAM 領域は，次の 4 つです。

SPOL0　…　System Memory Pool 0
SPOL1　…　System Memory Pool 1
UPOL0　…　User Memory Pool 0
UPOL1　…　User Memory Pool 1

これらの配置情報は，システム・コンフィギュレーション・ファイルにて「先頭アドレス」と「サイズ」を指定することによって決定されます。つまり，使用できる RAM 領域上のアドレスとそのサイズを指定することになります。
また，これらのメモリ・プールは用途が決まっており，次のような分類になります。

表 C － 1　メモリ・プールの種類と割り付けられるもの

メモリ・プール名	割りつけられるもの
SPOL0	オペレーティング・システム管理テーブル レディ・キュー 各種管理ブロック タスク・スタック 割り込みハンドラ用スタック
SPOL1	タスク・スタック 割り込みハンドラ用スタック 可変長メモリ・プール
UPOL0	可変長メモリ・プール
UPOL1	可変長メモリ・プール

SPOL0 は，RX850 Pro のシステムに関する情報が配置されるため，生成が必須なメモリ・プールです。SPOL1 は SPOL0 だけでまかなえる場合は，作成しなくても良いメモリ・プールです。また，管理ブロックが配置される SPOL0 を内蔵 RAM に配置し，比較的大きいサイズを必要とするスタックを SPOL1 として外部 RAM に配置することによって，システムのパフォーマンスを向上させる使用方法もあります。
UPOL0 は RX850 Pro のメモリ管理機能を使用する場合に必要となります。その場合は UPOL0 から作成してください。UPOL1 だけの作成はできません。

C.2　管理領域のメモリ容量

RX850 Pro における，オペレーティング・システム管理テーブル，各種管理ブロックが使用するサイズについて説明します。

これらは SPOL0 より確保されます。表 C － 2 に，各オブジェクト 1 つ当たりが使用する管理領域のサイズとその算出方法を示します。

表 C － 2　オブジェクト管理領域のサイズ

オブジェクト名	1 オブジェクト当たりの管理領域サイズ（単位：バイト）	サイズの算出方法（単位：バイト）
オペレーティング・システム管理テーブル，レディ・キュー	520 ～ 1048	504 + align 32 (優先度数 + 4) / 8 + align 4 ((優先度数 + 4) * 2) 優先度数は，システム最大値情報 *pri_lvl* と同値。
システム・メモリ領域管理ブロック	8	8 * 4 = 32 SPOL0，SPOL1，UPOL0，UPOL1 それぞれにつき 8 バイト。4 つすべてを生成しなくても，必ず 4 つ分確保されるため，常に 32 バイト確保される。
タスク管理ブロック	56	56 * タスクの最大生成数 タスクの最大生成数は，システム最大値情報 *maxtsk* と同値。
セマフォ管理ブロック	20	20 * セマフォの最大生成数 セマフォの最大生成数は，システム最大値情報 *maxsem* と同値。
イベントフラグ管理ブロック	20	20 * イベントフラグの最大生成数 イベントフラグの最大生成数は，システム最大値情報 *maxflg* と同値。
メールボックス管理ブロック	20	20 * メールボックスの最大生成数 メールボックスの最大生成数は，システム最大値情報 *maxmbx* と同値。
割り込みハンドラ管理ブロック	16	16 * 割り込みハンドラの最大生成数 + allign4 (最大割り込み要因番号) 割り込みハンドラの最大生成数は，システム最大値情報 *maxint* と同値。 最大割り込み要因番号は，システム最大値情報 *maxintfactor* と同値。
周期起動ハンドラ管理ブロック	40	40 * 周期起動ハンドラの最大生成数 周期起動ハンドラの最大生成数は，システム最大値情報 *maxcyc* と同値。
メモリ・プール管理ブロック	24	24 * メモリ・プールの最大生成数 メモリ・プールの最大生成数は，システム最大値情報 *maxmpl* と同値。
拡張 SVC ハンドラ管理ブロック	16	16 * 拡張 SVC ハンドラの最大生成数 拡張 SVC ハンドラの最大生成数は，システム最大値情報 *maxsvc* と同値。

C.3　タスク・スタックの容量

タスク・スタック領域は，次の 4 つに分類されます。

- スタック管理ブロック
- コンテキスト領域
- 割り込みスタック・フレーム
- タスク使用領域

タスク生成時（コンフィギュレーション時，または cre_tsk 発行時）に，タスク・スタックのサイズを指定しますが，その値は「割り込みスタック・フレーム」と「タスク使用領域」の合計サイズとなります。実際にメモリに確保されるサイズは，上記の「スタック管理テーブル」と「コンテキスト領域」のサイズがさらに加算されたサイズになります（さらに，その値を 4 バイトでアラインした値）。
「タスク使用領域」はユーザのアプリケーションによって変動しますが，「スタック管理テーブル」「コンテキスト領域」，および，「割り込みスタック・フレーム」のサイズは決まっており，そのサイズは次のようになっています。

表 C － 3　タスクで使用するタスク・スタックのサイズ

タスク・スタック領域	サイズ（単位：バイト）
スタック管理テーブル	28
コンテキスト領域 （割り込みスタック・フレームを含む：72 バイト）	148
タスク使用領域	アプリケーション依存

タスク生成時（コンフィギュレーション時，または cre_tsk 発行時）に，タスク・スタックのサイズを 100 バイトと指定した場合，実際にメモリに確保されるスタック・サイズは，次のようになります。

100 + 28 + 148 = 276 バイト

また，タスクから拡張 SVC ハンドラを起動している場合，ハンドラ実行のためのレジスタ退避領域と，拡張 SVC ハンドラ自身が消費するスタック領域が必要となります。これらのサイズは次のようになります。

表 C － 4　拡張 SVC ハンドラで使用するタスク・スタックのサイズ

タスク・スタック領域	サイズ（単位：バイト）
拡張 SVC ハンドラ用レジスタ退避領域	28
拡張 SVC ハンドラ使用領域	アプリケーション依存

タスク・スタック領域は，タスク生成時に SPOL0，または，SPOL1 から確保され，タスク削除（del_tsk，exd_tsk，ter_tsk）によって解放されます。したがって，すべてのタスクが同時に生成される可能性のある場合は，すべてのタスクにおいてサイズを計算し，それらを合計したサイズが確保可能となるように，SPOL0，SPOL1 のサイズを決定する必要があります。すべてのタスクが同時に生成されない場合は，同時に生成された状態となるタスクの組み合わせの中で，スタックのサイズの合計が最大となる組み合わせのサイズを計算し，そのサイズを基に，SPOL0，SPOL1 のサイズを決定します。
次に，タスク・スタック・サイズの計算方法をまとめたものを示します。全サイズの合計がメモリ領域として確保する必要のあるサイズ，網掛け部分のサイズの合計が，タスク生成時（コンフィギュレーション時，または cre_tsk 発行時）に指定するタスク・スタック・サイズです。なお，メモリ領域に確保される値は 4 バイトでアラインされた値です。

表Ｃ－５　タスク・スタックで使用されるサイズのまとめ

タスク・スタック領域	サイズ（単位：バイト）	備考
スタック管理テーブル	28	
コンテキスト領域 （割り込みスタック・フレームを含む：72 バイト）	148	
タスク使用領域	アプリケーション依存	タスクがプッシュ・ポップするスタック・サイズを算出して指定。 使用している変数の数等を考慮。 タスクからシステム・コールが発行された際に RX850 Pro は lp（r31）をプッシュするので 4 バイト加算。
拡張SVCハンドラ用レジスタ退避領域	28	拡張 SVC ハンドラを使用していない場合は不要。
拡張 SVC ハンドラ使用領域	アプリケーション依存	拡張 SVC ハンドラを使用していない場合は不要。 拡張 SVC ハンドラからシステム・コールが発行された際に RX850 Pro は lp（r31）をプッシュするので 4 バイト加算。

C.4　割り込みハンドラ用スタックの容量

割り込みハンドラ用スタック領域は，次の 4 つにおいて使用されます。

- 初期化ハンドラ
- アイドル・タスク
- 割り込みハンドラ
- 周期起動ハンドラ

それぞれ，使用されるサイズについて説明します。

1） 初期化ハンドラ
初期化ハンドラとして記述した関数が消費（プッシュ・ポップ）するサイズ分の領域を確保します。初期化ハンドラ実行中は，タスクや割り込みハンドラが起動することがないため，「2) アイドル・タスク」以降で説明している領域サイズよりも，その消費サイズが小さい場合は，初期化ハンドラで消費するスタック・サイズを考慮する必要はありません。

2） アイドル・タスク
アイドル・タスク実行中，および，タスク終了後（del_tsk，exd_tsk，ter_tsk 発行後）に，次タスクが実行されるまでの間に入る割り込み用のスタック消費分です。このサイズとして 72 バイトが必要です。スタックがさらに伸長するケースがありますが，その場合については，「3) 割り込みハンドラ」の説明以降を参照してください。
なお，この 72 バイト分の領域は，初期化処理時に加算されていますので，コンフィギュレーション時に考慮に入れる必要はありません。つまり，コンフィギュレーション時に指定する割り込みハンドラ用スタックのサイズに 72 バイト加えた値が，実際に確保されるメモリ・サイズとなります。

3） 割り込みハンドラ
割り込みが発生する場合，初回割り込み発生時（タスクに割り込みが入ったとき）は，タスク・スタック，または，「2) アイドル・タスク」で示したアイドル・タスク用の領域に生成されます。以後，多重割り込みが発生する可能性がある場合は，最大ネスト回数分の割り込みスタック・フレームのサイズを加算する必要があります。
割り込みハンドラが起動する際は，割り込みスタック・フレームとは別に，レジスタ退避領域として，28 バイトが必要となります。たとえば，タスク・スタックに割り込みスタック・フレーム分の情報を積んだあと，sp（スタック・ポインタ）を割り込みハンドラ用スタックに切り替えて，さらに積まれる分に当たります。多重割り込みを許可するシステムの場合，このサイズを考慮に入れる必要があります。よって，「割り込みの最大ネスト回数 + 1（初回割り込み分）」に，28 を掛けた値を，割り込みハンドラ用スタックのサイズに加算します。
さらに，割り込みハンドラとして記述した関数が，プッシュ・ポップによって消費するスタック・サイズを，割り込みが最大にネストした場合を考慮して加算します。これには，割り込みハンドラ内でシステム・コールを発行する場合 4 バイトを，拡張 SVC ハンドラを発行する場合は，28 バイトを加算します。また，拡張 SVC ハンドラ内でシステム・コールを発行している場合は，さらに 4 バイトを加算します。
クロック・ハンドラは，プッシュ・ポップによってスタックを消費しない割り込みハンドラとして扱います。これを考慮したサイズを算出します。

4） 周期起動ハンドラ
周期起動ハンドラは，クロック・ハンドラから呼び出されるサブルーチンとして実装されています。
また，周期起動ハンドラ実行中に，新たなクロック割り込みが発生し，別の周期起動ハンドラが起動するような態になっても，実行中だった周期起動ハンドラの処理が優先されます。したがって，周期起動ハンドラとして記述した関数の中で，その関数自身が消費するスタック・サイズの一番大きいものを，ハンドラ用スタックのサイズに加算します。

次に，割り込みハンドラ用スタックのサイズの計算方法をまとめたものを示します。全サイズの合計がメモリ領域として確保する必要のあるサイズ，網掛け部分のサイズの合計が，コンフィギュレーション時に指定する割り込みハンドラ用スタックのサイズです。なお，メモリ領域に確保される値は 4 バイトでアラインされた値です。

表Ｃ－6　多重割り込みを受け付けないシステムにおける割り込みハンドラ用スタック・サイズ

割り込みハンドラ用スタック領域	サイズ（単位：バイト）	備考
アイドル・タスク用領域	144	
割り込みハンドラ用レジスタ退避領域	28	
割り込みハンドラ使用領域	アプリケーション依存	割り込みハンドラがプッシュ･ポップするスタック・サイズを算出して指定。 使用している変数の数等を考慮。 割り込みハンドラからシステム･コールが発行された際に RX850 Pro は lp（r31）をプッシュするので 4 バイト加算。 使用する割り込みハンドラ中で最もスタックを使用するハンドラ（周期起動ハンドラを含む）の使用スタック・サイズを指定。
拡張SVCハンドラ用レジスタ退避領域	28	拡張 SVC ハンドラを割り込みハンドラが呼び出していない場合は不要。
拡張 SVC ハンドラ使用領域	アプリケーション依存	拡張 SVC ハンドラを割り込みハンドラが呼び出していない場合は不要。 拡張 SVC ハンドラからシステム・コールが発行された際に RX850 Pro は lp（r31）をプッシュするので 4 バイト加算。

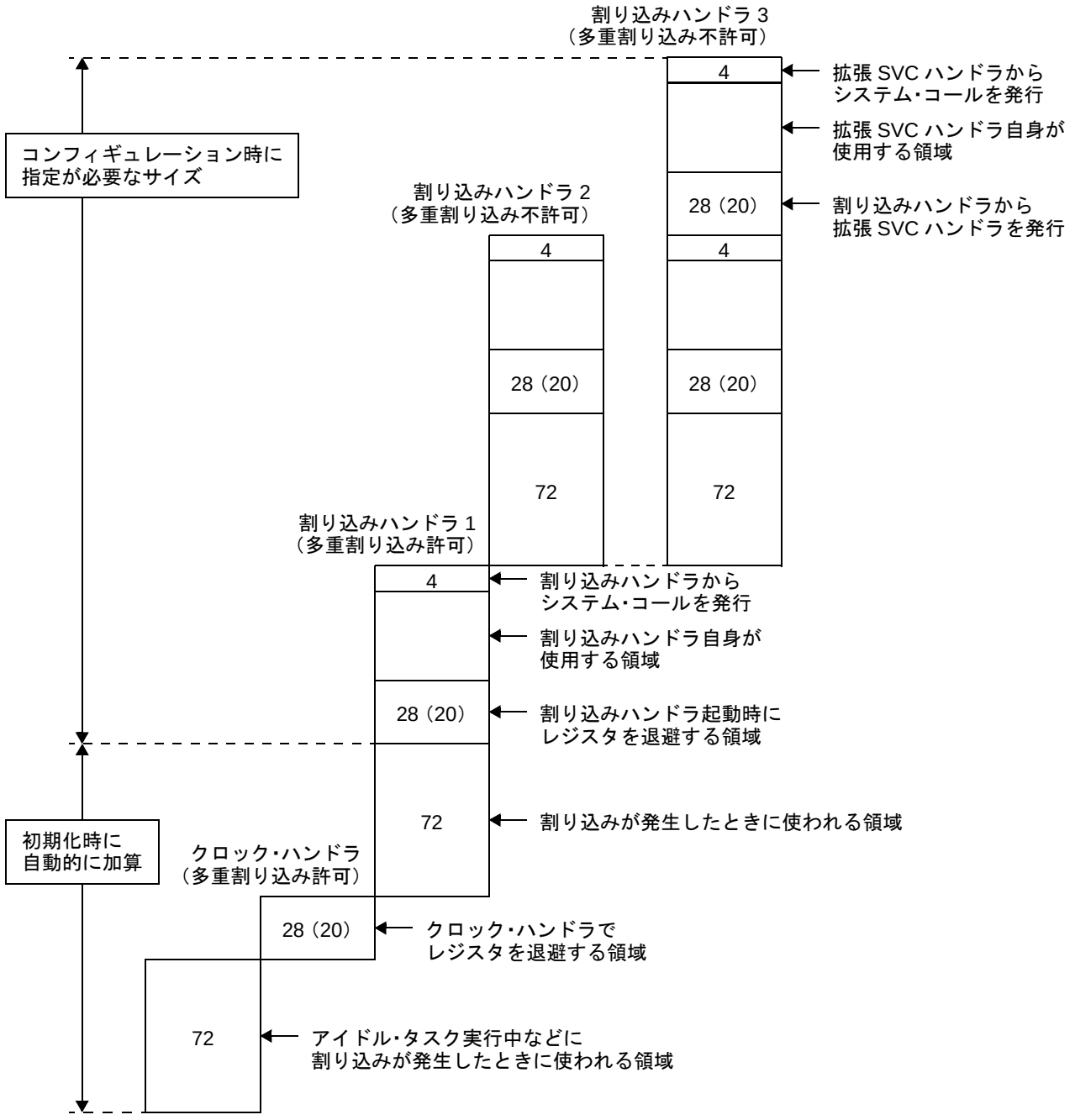
この表は多重割り込みを受け付けない場合のものですが，周期起動ハンドラに対して割り込みが入り，その割り込みを受け付けてしまう場合は，多重割り込みと同等になります。つまり，この表に該当するケースとしては，ニュークリアス共通部オブジェクトとして rxtmcore.o（周期起動ハンドラ内でクロック割り込みより高い割り込み受け付け可能版）を使用し，クロック割り込みより高い割り込みを使用していないこと，および，割り込みハンドラはすべて割り込み禁止状態で動作するアプリケーションのときにのみ該当します。

表Ｃ－7　多重割り込みを受け付けるシステムにおける割り込みハンドラ用スタック・サイズ

割り込みハンドラ用スタック領域	サイズ（単位：バイト）	備考
アイドル・タスク用領域	144	
割り込みスタック・フレーム	72 * n	
割り込みハンドラ用レジスタ退避領域	28 * (n+1)	
割り込みハンドラ使用領域	アプリケーション依存	割り込みハンドラがプッシュ･ポップするスタック・サイズを算出して指定。 使用している変数の数等を考慮。 割り込みハンドラからシステム･コールが発行された際に RX850 Pro は lp（r31）をプッシュするので 4 バイト加算。 使用する割り込みハンドラ毎（周期起動ハンドラを含む）に使用スタック･サイズを算出し，その合計を使用スタック･サイズを指定。
拡張SVCハンドラ用レジスタ退避領域	28 * m	拡張 SVC ハンドラを割り込みハンドラが呼び出していない場合は不要。
拡張 SVC ハンドラ使用領域	アプリケーション依存	拡張 SVC ハンドラを割り込みハンドラが呼び出していない場合は不要。 拡張 SVC ハンドラからシステム・コールが発行された際に RX850 Pro は lp（r31）をプッシュするので 4 バイト加算。

上記の表で，n は割り込みの最大ネスト回数を，m は拡張 SVC ハンドラを使用している割り込みハンドラの数を指します。

図Ｃ－１　割り込みハンドラ用スタック領域の見積もり



C.5　メモリ・プールの容量

メモリ領域（システム・メモリ，メモリ・プール，メモリ・ブロック）の容量について説明します。
メモリ領域は，

- システム・メモリ（UPOL0，または UPOL1）を確保（コンフィギュレーション時）
- システム・メモリからメモリ・プールを確保（コンフィギュレーション時，または cre_mpl 発行時）
- メモリ・プールからメモリ・ブロックを獲得（get_blk，pget_blk，tget_blk の発行）

といった手順で確保します。
なお，各メモリ領域（システム・メモリ，メモリ・プール，メモリ・ブロック）の容量については，実際にアプリケーションが使用するサイズに 8 バイト（メモリ領域管理用）を加算し，さらに 4 バイト・アラインした値である必要があります。

表 C － 8　メモリ・プールのサイズ

オブジェクト名	サイズの算出方法（単位：バイト）
メモリ・プール	align 4 (メモリ・プールのサイズ + 8) メモリ・プールのサイズは，cre_mpl 発行時，または，メモリ・プール情報 *mpl_siz* と同値。

C.6　メモリ容量見積もりの例

次に，RX850 Pro の管理領域 SPOL，UPOL として使用されるメモリ容量の見積もり方法の例を示します。
ここでは，対象CPUがV850E1コアであり，システム・コールcre_tsk, cre_mplは発行されないものとして見積もります。

【 アプリケーション情報 】

情報	値（単位：バイト）
割り込みハンドラ用スタック領域 *intstk_siz*	SPOL0 より 256
タスクの優先度範囲 *pri_lvl*	15
タスクの最大生成数 *maxtsk*	2
セマフォの最大生成数 *maxsem*	1
イベントフラグの最大生成数 *maxflg*	2
メールボックスの最大生成数 *maxmbx*	3
割り込みハンドラの最大生成数 *maxint*	4
メモリ・プールの最大生成数 *maxmpl*	2
周期起動ハンドラの最大登録数 *maxcyc*	1
拡張 SVC ハンドラの最大登録数 *maxsvc*	1
割り込み要因番号の最大値 *maxintfactor*	56
タスク・スタック情報	SPOL0 より 256 SPOL1 より 0
メモリ・プール情報	UPOL0 より 4096 UPOL1 より 8192

【 見積もり方法 】

オブジェクト情報	サイズ（単位：バイト）
オペレーティング・システム管理テーブル	SPOL0 より 504 + align 32 (15 + 4) / 8 + align 4 ((15 + 4) * 2) = 548
システム・メモリ管理ブロック	SPOL0 より 8 * 4 = 32
タスク管理ブロック	SPOL0 より 56 * 2 = 112
セマフォ管理ブロック	SPOL0 より 20 * 1 = 20
イベントフラグ管理ブロック	SPOL0 より 20 * 2 = 40
メールボックス管理ブロック	SPOL0 より 20 * 3 = 60
割り込みハンドラ管理ブロック	SPOL0 より 16 * 4 + align 4 (56) = 120
メモリ・プール管理ブロック	SPOL0 より 24 * 2 = 48
周期起動ハンドラ管理ブロック	SPOL0 より 40 * 1 = 40

オブジェクト情報	サイズ（単位：バイト）
拡張 SVC ハンドラ管理ブロック	SPOL0 より 16 * 1 = 16
タスク・スタック	SPOL0 より align 4 (28 + 148 + 256) + align 4 (28 + 148 + 256) = 864
割り込みハンドラ用スタック	SPOL0 より align 4 (144 + 28 + 256) = 428
メモリ・プール	UPOL0 より 4096 + 8 = 4104 UPOL1 より 8192 + 8 = 8200

上記の計算結果より

SPOL0：　548 + 32 + 112 + 20 + 40 + 60 + 120 + 48 + 40 + 16 + 864 + 428 = 2328 バイト
SPOL1：　0 バイト
UPOL0：　4104 バイト
UPOL1：　8200 バイト

がそれぞれ必要であることになります。

付録 D　索　　引

A

C

D

E

F

G

L

P

R

S

T

U

V

W

い

う

か

き

こ

し

す

せ

た

ち

と

に

は

ひ

ふ

み

め

り

ろ

わ

〔メ　モ〕

【発　行】
NECエレクトロニクス株式会社
〒211-8668　神奈川県川崎市中原区下沼部1753
電話（代表）：044(435)5111

お問い合わせ先

【ホームページ】

NECエレクトロニクスの情報がインターネットでご覧になれます。
URL(アドレス)　http://www.necel.co.jp/

【営業関係，技術関係お問い合わせ先】

半導体ホットライン
（電話：午前 9:00～12:00，午後 1:00～5:00）
電　話　：044−435−9494
E-mail　：info@necel.com

【資料請求先】

NECエレクトロニクスのホームページよりダウンロードいただくか，NECエレクトロニクスの販売特約店へお申し付けください。