RX ファミリ リアルタイム OS RI600PX V1.01.01 リリースノート

R20UT2478JJ0100
Rev.1.00
2013.03.15

この度は, RX ファミリ リアルタイム OS RI600PX をご使用いただきまして, 誠にありがとうございます。

本資料では, 本製品をお使いいただく上での制限事項および注意事項を記載しております。ご使用の前に, 必ずお読みくださいますようお願い申し上げます。

目次

第 1 章 製品構成 3

第 2 章 ユーザーズマニュアルについて 4

第 3 章 対象デバイスについて 5

第 4 章 動作環境 6
4.1 ハードウェア環境 6
4.2 ソフトウェア環境 6
4.3 対応ツール 6

第 5 章 インストール時の注意事項 7
5.1 インストール時の注意事項 7
5.1.1 管理者権限に関する注意事項 7
5.1.2 実行環境に関する注意事項 7
5.1.3 ネットワーク・ドライブに関する注意事項 7
5.1.4 インストール先フォルダ名に関する注意事項 7
5.1.5 インストール後の必要ファイルに関する注意事項 7
5.1.6 機能の変更や修復に関する注意事項 7
5.1.7 インストール・フォルダの変更に関する注意事項 8
5.1.8 インストールするバージョンに関する注意事項 8
5.1.9 インストーラの起動に関する注意事項 8
5.1.10 インストールの順序に関する注意事項 8
5.2 アンインストール時の注意事項 8
5.2.1 管理者権限に関する注意事項 8
5.2.2 アンインストールのフォルダに関する注意事項 8
5.2.3 インストーラ以外での追加／修正に関する注意事項 8

第 6 章 アンインストール時の選択キーワード 9

第 7 章 前バージョン（V1.01.00＋2012 年 11 月のオンライン・アップデート）からの変更点について 10
7.1 追加・更新されたツール 10
7.2 CubeSuite+ V2 に対応 10
7.3 CC-RX V2 に対応 10
7.4 機能追加 10
7.4.1 リソース情報表示プラグイン 10
7.5 制限事項の解除 11
7.5.1 カーネル 11
7.5.2 CubeSuite+用プラグイン 11

第 8 章 オンライン・アップデートでの変更点について 12

第 9 章 注意事項 13
9.1 ビルド時の注意 13
9.1.1 ブート処理ファイルの”nostuff”コンパイルオプション 13
9.1.2 ROM から RAM にマップするセクション 14
9.2 タイマ・テンプレート・ファイル 17
9.3 カーネル・ソース・コードのビルド方法 17
9.4 スタック使用量について 18
9.4.1 基本クロック割り込みハンドラのスタック使用量(*clocksz1*，*clocksz2*，*clocksz3*) 18
9.4.2 サービス・コールのスタック使用量(*svcsz*) 18
9.4.3 カーネル・ライブラリをビルドした場合 20
9.5 リアルタイム OS リソース情報パネルに関する注意事項 21
9.5.1 参照はリアルタイム OS 初期化後に行う 21
9.5.2 デバッグ情報を生成したプログラムを使用する 21
9.6 RI600/PX をご使用されていたお客様へ 22
9.6.1 制限事項の解除 22
9.6.2 カーネルのバージョン情報 22
9.6.3 CubeSuite+に関する注意事項 22

第 10 章 制限事項 23
10.1 CubeSuite+使用時の制限事項 23
10.1.1 サービス・コール情報ファイル格納パス 23
10.1.2 リアルタイム OS リソース情報パネル 23
10.1.3 リアルタイム OS リソース情報パネルの「残り時間」 24

第 11 章 ドキュメント訂正 25

第 12 章 サンプル・プログラム 26
12.1 概要 26
12.2 ファイル構成 27
12.3 メモリ・マップ 28
12.3.1 RAM 領域 28
12.3.2 ROM 領域 28
12.3.3 メモリ・オブジェクト 29
12.3.4 ユーザ・スタック 31
12.4 セクションに関するビルド・ツールの設定 32
12.4.1 標準ライブラリ構築ツール 32
12.4.2 C/C++コンパイラ 32
12.4.3 リンカ 32
12.5 アクセス違反の対処例 33

第1章 製品構成

RI600PX は型名により，契約形態と提供物が異なります。

型名	契約形態	提供物
R0R5RX00PCW011	評価契約，インストール可能な PC は 1 台	A
R0R5RX00PCW01A	評価契約，インストール可能な PC は無制限	A
R0R5RX00PCW01K	量産契約，量産数は 3000 台まで	A
R0R5RX00PCW01U	量産契約，量産数は無制限	A
R0R5RX00PCW01Z	量産契約，量産数は無制限，ソース・コード付き	B

提供物は以下となります。

提供物		ツール名		バージョン
B	A	リアルタイム OS RI600PX カーネル オブジェクト		V1.01.00
		コマンドライン・コンフィギュレータ cfg600px		V1.01.01
		テーブル生成ユーティリティ mkritblpx		V1.01.00
		CubeSuite+用プラグイン		
			リアルタイム OS ビルド設定プラグイン（共通部）	V2.00.00
			リアルタイム OS ビルド設定プラグイン（RI600PX 依存部）	V2.00.00
			リアルタイム OS 解析制御プラグイン（共通部）	V2.00.00
			リアルタイム OS 解析制御プラグイン（μITRON4 依存部）	V2.00.00
			リアルタイム OS 解析制御プラグイン（RI600PX 依存部）	V2.00.00
			リアルタイム OS リソース情報表示プラグイン（共通部）	V2.00.00
			リアルタイム OS リソース情報表示プラグイン（μITRON4 依存部）	V2.00.00
		リアルタイム OS RI600PX カーネル ソース・コード		V1.01.00

第2章 ユーザーズマニュアルについて

本製品に対応したユーザーズマニュアルは，次のようになります。本文書と合わせてお読みください。

マニュアル名	資料番号
RI シリーズ リアルタイム・オペレーティング・システム ユーザーズマニュアル 起動編	R20UT0751JJ0102
RI600PX リアルタイム・オペレーティング・システム ユーザーズマニュアル コーディング編	R20UT0964JJ0100
RI600PX リアルタイム・オペレーティング・システム ユーザーズマニュアル デバッグ編	R20UT0950JJ0100
RI シリーズ リアルタイム・オペレーティング・システム ユーザーズマニュアル メッセージ編	R20UT0756JJ0102

なおユーザーズマニュアルは PDF ファイルで提供媒体にパッケージされています。

また，ルネサス エレクトロニクスのホームページから入手することができます。

第3章　対象デバイスについて

本製品は，以下のデバイスに対応しています。

- MPU（Memory Protection Unit）を搭載した RX600 シリーズ MCU
- MPU（Memory Protection Unit）を搭載した RX200 シリーズ MCU

第4章 動作環境

本製品を使用するには，次の環境が必要になります。

4.1 ハードウェア環境

- プロセッサ：1GHz 以上（ハイパー・スレッディング，マルチ・コア CPU に対応）
- メモリ容量：推奨 2GB 以上。最低 1GB 以上（64 ビット版 Windows® 7 および Windows® 8 では 2GB 以上）
- ディスプレイ：1024×768 以上の解像度，65536 色以上

4.2 ソフトウェア環境

次のソフトウェア環境に対応しています。

- Windows® XP（32bit 版のみ）
- Windows Vista®（32bit 版，64bit 版）
- Windows® 7（32bit 版，64bit 版）
- Windows® 8（32bit 版，64bit 版）
- .NET Framework 4 + 言語パック
- Microsoft Visual C++ 2010 SP1 ランタイム・ライブラリ
- Internet Explorer 6.0 以上

いずれの場合も，最新の Service Pack がインストールされていることを推奨します。

4.3 対応ツール

本製品は次の開発ツールに対応しています。

ツール名	提供元	バージョン
統合開発環境 CubeSuite+	ルネサス エレクトロニクス	V2.00.00 以降
C/C++コンパイラ CC-RX	ルネサス エレクトロニクス	V1.02.01 以降

第5章 インストール時の注意事項

本章では，インストール，アンインストール時の注意事項について説明します。

5.1 インストール時の注意事項

5.1.1 管理者権限に関する注意事項

インストールするには，Windows®の管理者権限が必要です。

5.1.2 実行環境に関する注意事項

インストールを実行する Windows®には，.NET Framework と Visual C++ のランタイム・ライブラリがインストールされている必要があります。

5.1.3 ネットワーク・ドライブに関する注意事項

ネットワーク・ドライブからのインストールはできません。

また，ネットワーク・ドライブへのインストールもできません。

5.1.4 インストール先フォルダ名に関する注意事項

インストール先フォルダ名に指定可能な文字は，Windows®に準じます。 / * : < > ? | " ¥ ; , の 11 文字は使用できません。また，空白文字ではじまるものと空白文字で終わるものは指定できません。

指定する際に，絶対パスで指定し，相対パスでは指定しないでください。

また，インストール先フォルダの区切り子には ¥ を使用してください。/ は使用しないでください。

5.1.5 インストール後の必要ファイルに関する注意事項

インストール後にできる次のフォルダ（含むフォルダ以下のファイル）には，ツールが動作するために必要なファイル類がありますので削除しないでください。

（Windows®が 32bit 版で，システムドライブが C:の場合）

C:¥Program Files¥Common Files¥Renesas Electronics CubeSuite+¥

（Windows®が 64bit 版で，システムドライブが C:の場合）

C:¥Program Files (x86)¥Common Files¥Renesas Electronics CubeSuite+¥

5.1.6 機能の変更や修復に関する注意事項

インストール済みのツールに対して，機能の変更や修復を行う場合は，そのツールのインストール・パッケージを用意し，インストール用プログラムを実行すると起動する，プログラムの保守画面で，「変更」または「修復」を実行してください。

「プログラムの追加と削除」(Windows® XP の場合)，「プログラムと機能」(Windows Vista® / Windows® 7 / Windows® 8 の場合)の[変更]ボタンから行うとエラーになります。

5.1.7 インストール・フォルダの変更に関する注意事項

インストールしたツールのフォルダを変更したい場合には，一度全ての CubeSuite+関連ツールをアンインストールしてから，再度インストールしてください。

全ての CubeSuite+関連ツールをアンインストールするには，統合アンインストーラを起動して，表示されているツール類を全て削除した後，「プログラムの追加と削除」(Windows® XP の場合)，「プログラムと機能」(Windows Vista® / Windows® 7 / Windows® 8 の場合)で「CubeSuite+ Utilities」を削除してください。

5.1.8 インストールするバージョンに関する注意事項

新しいバージョンがインストールされている場合には，古いバージョンがインストールされない可能性があります。

5.1.9 インストーラの起動に関する注意事項

日本語版以外の Windows®で，インストーラを起動するパスに多バイト文字が含まれているとエラーとなりインストールを実行することができません。

5.1.10 インストールの順序に関する注意事項

本パッケージをインストールする前に CubeSuite+をインストールしてください。なお，本パッケージがインストールされるフォルダは，CubeSuite+がインストールされているフォルダと同じとなります。

5.2 アンインストール時の注意事項

5.2.1 管理者権限に関する注意事項

アンインストールするには，Windows®の管理者権限が必要です。

5.2.2 アンインストールのフォルダに関する注意事項

ツールのアンインストールの実行順序によっては，フォルダが完全に削除されない場合があります。この場合，アンインストールした後に残ったフォルダは，エクスプローラ等で削除してください。

5.2.3 インストーラ以外での追加／修正に関する注意事項

ツール，および，マニュアル類をインストールしたフォルダに，本製品のインストーラ以外の手段によって，追加または修正されたファイルは，アンインストール時に削除できません。

第6章 アンインストール時の選択キーワード

本製品をアンインストールする場合は，2 つの方法があります。

- 統合アンインストーラを使用する(CubeSuite+自体をアンインストールする)
- 個別にアンインストールする(本製品のみをアンインストールする)

個別にアンインストールを行なう場合，コントロールパネルの

- 「プログラムの追加と削除」(Windows® XP の場合)
- 「プログラムと機能」(Windows Vista® / Windows® 7/ Windows® 8 の場合)

から，以下を削除してください。

- 「CubeSuite+ Realtime OS Common Plugins」
- 「CubeSuite+ Realtime OS RI600PX Plugins」
- 「CubeSuite+ Realtime OS RI600PX Object Release」または「CubeSuite+ Realtime OS RI600PX Source Release」

第7章 前バージョン（V1.01.00＋2012 年 11 月のオンライン・アップデート）からの変更点について

7.1 追加・更新されたツール

ツール名	更新前	更新後
リアルタイム OS RI600PX カーネル オブジェクト	V1.01.00	変更なし
コマンドライン・コンフィギュレータ cfg600px	V1.01.00	V1.01.01
テーブル生成ユーティリティ mkritblpx	V1.01.00	変更なし
CubeSuite+用プラグイン		
リアルタイム OS ビルド設定プラグイン（共通部）	V1.03.00	V2.00.00
リアルタイム OS ビルド設定プラグイン（RI600PX 依存部）	V1.00.00	V2.00.00
リアルタイム OS 解析制御プラグイン（共通部）	V1.01.01	V2.00.00
リアルタイム OS 解析制御プラグイン（μITRON4 依存部）	V1.01.01	V2.00.00
リアルタイム OS 解析制御プラグイン（RI600PX 依存部）	V1.00.00	V2.00.00
リアルタイム OS リソース情報表示プラグイン（共通部）	V1.02.00	V2.00.00
リアルタイム OS リソース情報表示プラグイン（μITRON4 依存部）	V1.02.00	V2.00.00
リアルタイム OS RI600PX カーネル ソース・コード	V1.01.00	変更なし
RI シリーズ リアルタイム・オペレーティング・システム ユーザーズマニュアル 起動編	Rev.1.02	変更なし
RI600PX リアルタイム・オペレーティング・システム ユーザーズマニュアル コーディング編	Rev.1.00	変更なし
RI600PX リアルタイム・オペレーティング・システム ユーザーズマニュアル デバッグ編	Rev.1.00	変更なし
RI シリーズ リアルタイム・オペレーティング・システム ユーザーズマニュアル メッセージ編	Rev.1.02	変更なし

7.2 CubeSuite+ V2 に対応

対応する CubeSuite+のバージョンを V2.00.00 に変更しました。

7.3 CC-RX V2 に対応

対応ツールに C/C++コンパイラ CC-RX V2.00.00 を追加しました。

7.4 機能追加

7.4.1 リソース情報表示プラグイン

リアルタイム OS リソース情報パネルの[タスク], [周期ハンドラ]および[アラーム・ハンドラ]タブにおいて，ダブルクリックによって選択されたプログラムのソースファイルを開く機能を追加しました。

7.5 制限事項の解除

以下の制限事項を解除しました。

7.5.1 カーネル

(1) リセットベクタに数値を指定する場合の問題

ツールニュース URL：http://tool-support.renesas.com/jpn/toolnews/121216/tn4.htm

7.5.2 CubeSuite+用プラグイン

(1) 「外部変数アクセス最適化」コンパイラ・オプション

「CC-RX(ビルド・ツール)」の[プロパティ・パネル]で以下の設定を行った場合，ビルド時に下記のエラーが発生する場合があります。

- 「CC-RX(ビルド・ツール)」の[プロパティ・パネル]の設定

タブ	カテゴリ	項目	設定値
コンパイラ・オプション	最適化	外部変数アクセス最適化を行う	はい(モジュール間で最適化) (-map)

- エラー

```
(O) : A3001 (F) Can't open file 'DefaultBuild¥ritable.src'
```

第8章 オンライン・アップデートでの変更点について

オンライン・アップデートはありません。

第9章　注意事項

本章では，RI600PX V1.01.01の注意事項について説明します。

9.1　ビルド時の注意

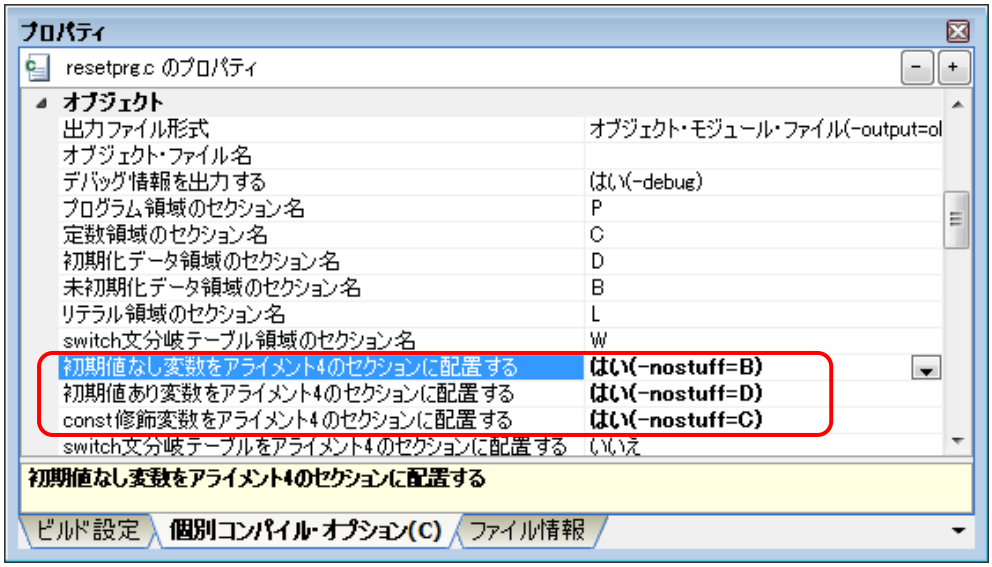
9.1.1　ブート処理ファイルの”nostuff”コンパイルオプション

「RI600PX リアルタイム・オペレーティング・システム　ユーザーズマニュアル コーディング編（R20UT0964JJ Rev.1.00）」の「17.2.3 ブート処理ファイルのコンパイル・オプション」に記載のように，ブート処理ファイル（サンプルプロジェクトでは”resetprg.c”）に”nostuff”オプションを設定する必要があります。そうでない場合，RI600PX は正常に動作しません。

“nostuff”オプションをブート処理ファイルのみに設定するにはブート処理ファイルの[プロパティ]パネルの[個別コンパイル・オプション]タブで，”nostuff”オプションを全ファイルに設定するに[CC-RX（ビルド・ツール）]の[プロパティ] パネルの[コンパイル・オプション]タブで，以下のいずれかを設定してください。

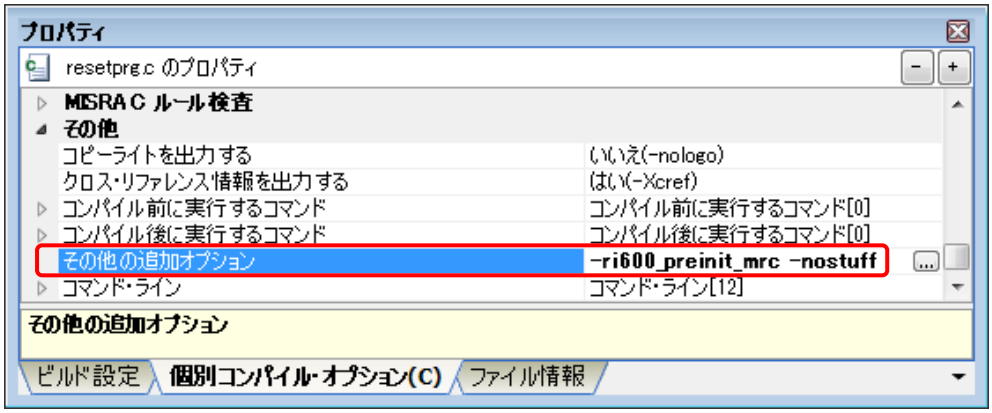
(1)　[オブジェクト]カテゴリで設定する

下図のように，[初期値なし変数をアライメント 4 のセクションに配置する]，[初期値あり変数をアライメント 4 のセクションに配置する]，および[const 修飾変数をアライメント 4 のセクションに配置する]を”はい”に設定してください。



(2)　[その他]カテゴリで設定する

下図のように，[その他の追加オプション]に”-nostuff”を追加してください。



9.1.2 ROM から RAM にマップするセクション

初期化データセクションは，ROM から RAM にマップする必要があります。RI600PX の初期化データセクション”DRI_ROM”セクションも，RAM 上の”RRI_RAM”セクションにマップする必要があります。そのためには，セクション情報ファイルの設定とリンケージエディタの設定が必要です。

(1) セクション情報ファイル

ブート処理関数（PowerON_Reset_PC( )）から呼び出されるセクション初期化関数”_INITSCT( )”は，コンパイラの標準ライブラリとして提供されます。

_INITSCT( )により，未初期化データセクションはゼロ初期化され，初期化データセクションは ROM 領域の初期値が RAM 領域へコピーされます。

初期化対象のセクションは、ユーザ・オウン・コーディング部としてセクション情報ファイルのセクション初期化用テーブル(DTBL,BTBL)へ記述する必要があります。_INITSCT()が使用するセクションの先頭アドレスおよび最終アドレスを、セクションアドレス演算子を用いて設定します。セクション初期化用テーブルのセクション名は、未初期化データ領域を C$BSEC、初期化データ領域を C$DSEC で宣言します。

備考：RI600PX の現在の実装では，”DRI_ROM”セクションの初期化は不要です。

```
#include "typedefine.h"

#pragma unpack

#pragma section C C$DSEC
extern const struct {
    _UBYTE *rom_s;      /* Start address of the initialized data section in ROM */
    _UBYTE *rom_e;      /* End address of the initialized data section in ROM   */
    _UBYTE *ram_s;      /* Start address of the initialized data section in RAM */
}   _DTBL[] = {
    { __sectop("DRI_ROM"), __secend("DRI_ROM"), __sectop("RI_RAM") },
    { __sectop("D"), __secend("D"), __sectop("R") },
    { __sectop("D_2"), __secend("D_2"), __sectop("R_2") },
    { __sectop("D_1"), __secend("D_1"), __sectop("R_1") },
    { __sectop("DS"), __secend("DS"), __sectop("RS") },
    { __sectop("DS_2"), __secend("DS_2"), __sectop("RS_2") },
    { __sectop("DS_1"), __secend("DS_1"), __sectop("RS_1") },
    { __sectop("DU_SH"), __secend("DU_SH"), __sectop("RU_SH") },
    { __sectop("DU_SH_2"), __secend("DU_SH_2"), __sectop("RU_SH_2") },
    { __sectop("DU_SH_1"), __secend("DU_SH_1"), __sectop("RU_SH_1") },
    { __sectop("DU_MASTERDOM"), __secend("DU_MASTERDOM"), __sectop("RU_MASTERDOM") },
    { __sectop("DU_MASTERDOM_2"), __secend("DU_MASTERDOM_2"), __sectop("RU_MASTERDOM_2") },
    { __sectop("DU_MASTERDOM_1"), __secend("DU_MASTERDOM_1"), __sectop("RU_MASTERDOM_1") },
    { __sectop("DU_DOM_A"), __secend("DU_DOM_A"), __sectop("RU_DOM_A") },
    { __sectop("DU_DOM_A_2"), __secend("DU_DOM_A_2"), __sectop("RU_DOM_A_2") },
    { __sectop("DU_DOM_A_1"), __secend("DU_DOM_A_1"), __sectop("RU_DOM_A_1") },
    { __sectop("DU_DOM_B"), __secend("DU_DOM_B"), __sectop("RU_DOM_B") },
    { __sectop("DU_DOM_B_2"), __secend("DU_DOM_B_2"), __sectop("RU_DOM_B_2") },
    { __sectop("DU_DOM_B_1"), __secend("DU_DOM_B_1"), __sectop("RU_DOM_B_1") }
};
```

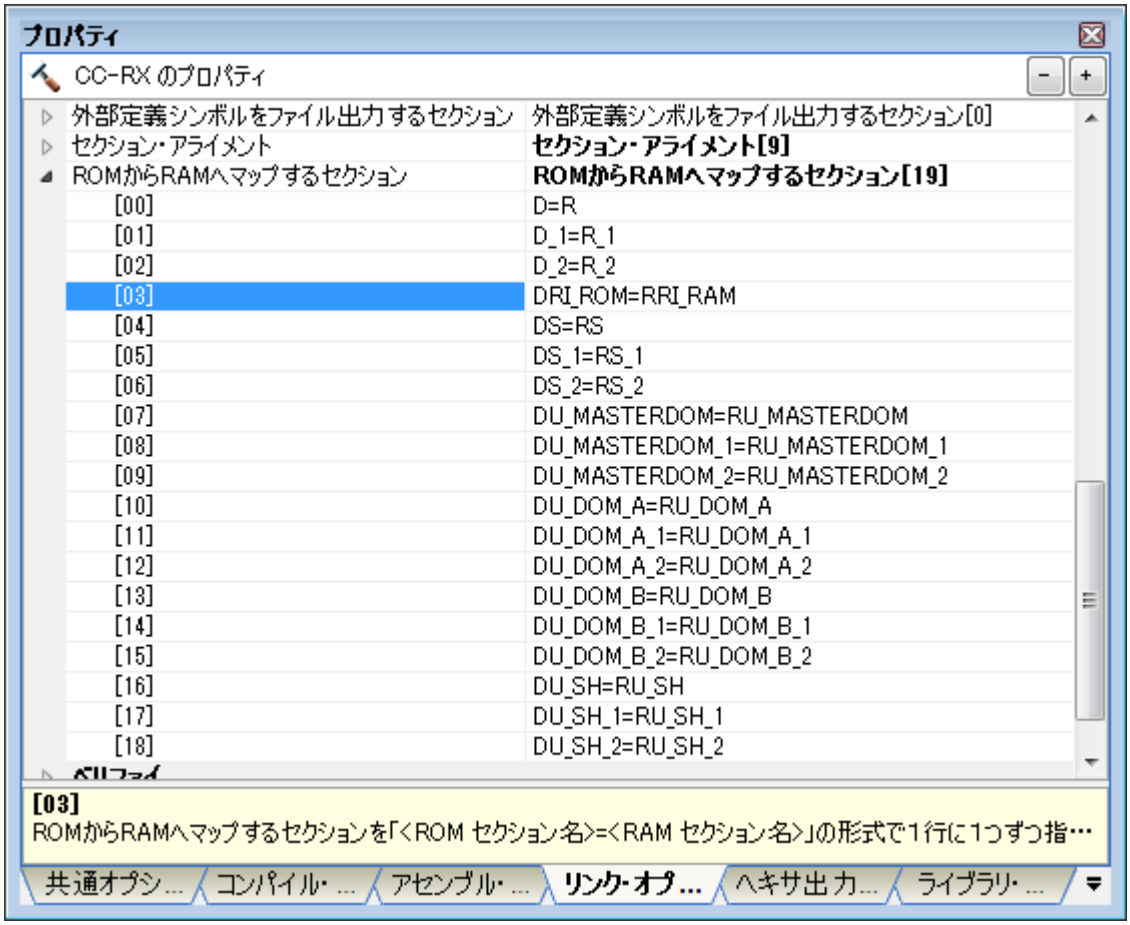
```
#pragma section C C$BSEC
extern const struct {
    _UBYTE *b_s;        /* Start address of non-initialized data section */
    _UBYTE *b_e;        /* End address of non-initialized data section */
}   _BTBL[] = {
    { __sectop("B"), __secend("B") },
    { __sectop("B_2"), __secend("B_2") },
    { __sectop("B_1"), __secend("B_1") },
    { __sectop("BS"), __secend("BS") },
    { __sectop("BS_2"), __secend("BS_2") },
    { __sectop("BS_1"), __secend("BS_1") },
    { __sectop("BU_SH"), __secend("BU_SH") },
    { __sectop("BU_SH_2"), __secend("BU_SH_2") },
    { __sectop("BU_SH_1"), __secend("BU_SH_1") },
    { __sectop("BU_MASTERDOM"), __secend("BU_MASTERDOM") },
    { __sectop("BU_MASTERDOM_2"), __secend("BU_MASTERDOM_2") },
    { __sectop("BU_MASTERDOM_1"), __secend("BU_MASTERDOM_1") },
    { __sectop("BU_DOM_A"), __secend("BU_DOM_A") },
    { __sectop("BU_DOM_A_2"), __secend("BU_DOM_A_2") },
    { __sectop("BU_DOM_A_1"), __secend("BU_DOM_A_1") },
    { __sectop("BU_DOM_B"), __secend("BU_DOM_B") },
    { __sectop("BU_DOM_B_2"), __secend("BU_DOM_B_2") },
    { __sectop("BU_DOM_B_1"), __secend("BU_DOM_B_1") }
};

#pragma section C CS

/*
** CTBL prevents excessive output of L1100 messages when linking.
** Even if CTBL is deleted, the operation of the program does not change.
*/
_UBYTE * const _CTBL[] = {
    __sectop("C_1"), __sectop("C_2"), __sectop("C"),
    __sectop("W_1"), __sectop("W_2"), __sectop("W"),
    __sectop("CU_MASTERDOM_1"), __sectop("CU_MASTERDOM_2"), __sectop("CU_MASTERDOM"),
    __sectop("CU_DOM_A_1"), __sectop("CU_DOM_A_2"), __sectop("CU_DOM_A"),
    __sectop("CU_DOM_B_1"), __sectop("CU_DOM_B_2"), __sectop("CU_DOM_B"),
    __sectop("WU_SH_1"), __sectop("WU_SH_2"), __sectop("WU_SH"),
    __sectop("LU_SH"),
    __sectop("CU_SH_1"), __sectop("CU_SH_2"), __sectop("CU_SH"),
    __sectop("CS_1"), __sectop("CS_2"), __sectop("CS"), __sectop("P")
};

#pragma packoption
```

(2) リンケージエディタの設定

セクション情報ファイルの_DTBL[]で指定したセクションと”DRI_ROM”，”RRI_RAM”セクションは，リンケージエディタの”rom”オプションを使って， ROM に配置されたセクションを RAM 上のセクションにマップする設定を行う必要があります。

下図のように，[CC-RX（ビルド・ツール）]の[プロパティ] パネルの[リンク・オプション]タブを設定してください。

なお，RI600PX が提供するサンプルプロジェクトでは，”DRI_ROM”セクションの”RRI_RAM”セクションへのマップを設定済みです。



9.2 タイマ・テンプレート・ファイル

以下に，RI600PX が提供するタイマ・テンプレート・ファイルと，対応している MCU を示します。

なお，タイマ・テンプレート・ファイルは，システム・コンフィギュレーション・ファイルの"clock.template"に指定するファイルです。

テンプレート・ファイル	対応 MCU
rx62n.tpl	RX600 シリーズ RX62N グループ RX600 シリーズ RX621 グループ RX600 シリーズ RX62T グループ
rx630.tpl	RX600 シリーズ RX630 グループ RX600 シリーズ RX631 グループ RX600 シリーズ RX63N グループ RX600 シリーズ RX63T グループ RX200 シリーズ RX21A グループ

9.3 カーネル・ソース・コードのビルド方法[1]

RI600PX カーネルはライブラリで提供されているため，通常はカーネル・ソース・コードをビルドしてカーネル・ライブラリを再生成する必要はありません。

カーネルのソース・コードは，”<インストール・フォルダ>¥src600”に格納されます。カーネルをビルドするためには，カレント・フォルダをこのフォルダとし，”nmake.exe”[2]を実行します。

ビルド時には，コンパイラが必要とする環境変数の設定が必要です。

ビルド例：

```
C:¥Program Files¥Renesas Electronics¥CubeSuite+¥RI600PX¥src600> nmake(RET)
```

カーネル・ライブラリは以下のフォルダに生成されます。

カーネル・ライブラリ名	内容
product¥big¥debug¥ri600big.lib	デバッグ情報付きビッグ・エンディアン・ライブラリ
product¥big¥release¥ri600big.lib	デバッグ情報なしビッグ・エンディアン・ライブラリ
product¥little¥debug¥ri600lit.lib	デバッグ情報付きリトル・エンディアン・ライブラリ
product¥little¥release¥ri600lit.lib	デバッグ情報なしリトル・エンディアン・ライブラリ

インストール・フォルダに対する書き込み権限がない場合，”src600”フォルダを書き込み可能なフォルダにコピーしてビルドしてください。ビルド後，インストール・フォルダに対する書き込み権限のあるユーザにて，生成されたライブラリをインストール・フォルダの”lib600” フォルダにコピーしてください。

1 ソース・コードが付属するのは，ソース付き量産契約版(R0R5RX0PCW01Z)のみです。

2 “nmake.exe”は，米国 Microsoft Corporation により提供されるプロジェクトをビルドするためのツールです。”nmake.exe”は，Microsoft Visual Studio 2008 等に含まれています。

9.4 スタック使用量について

9.4.1 基本クロック割り込みハンドラのスタック使用量(*clocksz1*，*clocksz2*，*clocksz3*)

「RI600PX リアルタイム・オペレーティング・システム ユーザーズマニュアル コーディング編」の付録 D.4 節に記載の *clocksz1*，*clocksz2* および *clocksz3* の値は，以下の通りです。

- *clocksz1*= 104
- *clocksz2*= 104
- *clocksz3* = 160

9.4.2 サービス・コールのスタック使用量(*svcsz*)

カーネルは，システム・スタックを使用します。

「RI600PX リアルタイム・オペレーティング・システム ユーザーズマニュアル コーディング編」の付録 D.4 節に記載の *svcsz* は，システムで使用しているサービス・コールの使用サイズ（下記表を参照）と以下の算出式の最大値としてください。

- アクセス例外ハンドラ（_RI_sys_access_exception()）を基点とする関数ツリーでの使用量＋16
- タイマ初期化コールバック関数（_RI_init_cmt_knl()）を基点とする関数ツリーでの使用量＋8

	サービス・コール	使用量	備考
タスク管理機能			
1	cre_tsk	44	
2	acre_tsk	44	
3	del_tsk	28	
4	act_tsk	32	
5	iact_tsk	24	
6	can_act	24	
7	ican_act	24	
8	sta_tsk	32	
9	ista_tsk	28	
10	ext_tsk	36	タスク開始関数からのリターン時にも ext_tsk が呼び出されます。
11	exd_tsk	40	
12	ter_tsk	104	
13	chg_pri	56	
14	ichg_pri	60	
15	get_pri	36	
16	iget_pri	36	
17	ref_tsk	44	
18	iref_tsk	44	
19	ref_tst	36	
20	iref_tst	36	
タスク付属同期機能			
21	slp_tsk	32	
22	tslp_tsk	40	
23	wup_tsk	52	
24	iwup_tsk	56	
25	can_wup	24	
26	ican_wup	24	
27	rel_wai	100	
28	irel_wai	104	
29	sus_tsk	32	
30	isus_tsk	28	
31	rsm_tsk	32	
32	irsm_tsk	28	
33	frsm_tsk	32	
34	ifrsm_tsk	28	
35	dly_tsk	40	
タスク例外処理機能			
36	def_tex	40	
37	ras_tex	28	
38	iras_tex	24	
39	dis_tex	24	
40	ena_tex	28	
41	sns_tex	24	
42	ref_tex	32	
43	iref_tex	32	
セマフォ			
44	cre_sem	44	
45	acre_sem	48	
46	del_sem	68	
47	sig_sem	52	
48	isig_sem	56	
49	wai_sem	48	

	サービス・コール	使用量	備考
50	pol_sem	24	
51	ipol_sem	24	
52	twai_sem	52	
53	ref_sem	36	
54	iref_sem	36	
イベントフラグ			
55	cre_flg	44	
56	acre_flg	48	
57	del_flg	68	
58	set_flg	68	
59	iset_flg	72	
60	clr_flg	24	
61	iclr_flg	24	
62	wai_flg	52	
63	pol_flg	36	
64	ipol_flg	36	
65	twai_flg	56	
66	ref_flg	36	
67	iref_flg	36	
データ・キュー			
68	cre_dtq	44	
69	acre_dtq	48	
70	del_dtq	68	
71	snd_dtq	52	
72	psnd_dtq	52	
73	ipsnd_dtq	56	
74	tsnd_dtq	56	
75	fsnd_dtq	52	
76	ifsnd_dtq	60	
77	rcv_dtq	52	
78	prcv_dtq	52	
79	iprcv_dtq	56	
80	trcv_dtq	52	
81	ref_dtq	40	
82	iref_dtq	40	
メールボックス			
83	cre_mbx	44	
84	acre_mbx	48	
85	del_mbx	68	
86	snd_mbx	52	
87	isnd_mbx	60	
88	rcv_mbx	48	
89	prcv_mbx	36	
90	iprcv_mbx	36	
91	trcv_mbx	52	
92	ref_mbx	36	
93	iref_mbx	36	
ミューテックス			
94	cre_mtx	44	
95	acre_mtx	48	
96	del_mtx	72	
97	loc_mtx	48	
98	ploc_mtx	32	
99	tloc_mtx	52	
100	unl_mtx	64	
101	ref_mtx	36	
メッセージ・バッファ			
102	cre_mbf	44	
103	acre_mbf	48	
104	del_mbf	68	
105	snd_mbf	56	
106	psnd_mbf	56	
107	ipsnd_mbf	64	
108	tsnd_mbf	56	
109	rcv_mbf	76	
110	prcv_mbf	76	
111	trcv_mbf	76	
112	ref_mbf	36	
113	iref_mbf	36	
固定長メモリ・プール			
114	cre_mpf	44	
115	acre_mpf	48	
116	del_mpf	68	
117	get_mpf	48	
118	pget_mpf	36	
119	ipget_mpf	36	
120	tget_mpf	52	
121	rel_mpf	52	
122	irel_mpf	48	
123	ref_mpf	36	
124	iref_mpf	36	
可変長メモリ・プール			
125	cre_mpl	80	
126	acre_mpl	80	
127	del_mpl	68	
128	get_mpl	92	
129	pget_mpl	96	
130	ipget_mpl	96	
131	tget_mpl	92	
132	rel_mpl	96	
133	ref_mpl	36	
134	iref_mpl	36	
時間管理機能			
135	set_tim	36	
136	iset_tim	36	
137	get_tim	36	

	サービス・コール	使用量	備考
138	iget_tim	36	
周期ハンドラ			
139	cre_cyc	40	
140	acre_cyc	44	
141	del_cyc	28	
142	sta_cyc	24	
143	ista_cyc	24	
144	stp_cyc	24	
145	istp_cyc	24	
146	ref_cyc	36	
147	iref_cyc	36	
アラーム・ハンドラ			
148	cre_alm	40	
149	acre_alm	44	
150	del_alm	28	
151	sta_alm	28	
152	ista_alm	28	
153	stp_alm	24	
154	istp_alm	24	
155	ref_alm	36	
156	iref_alm	36	
システム状態管理機能			
157	rot_rdq	28	
158	irot_rdq	24	
159	get_tid	36	
160	iget_tid	36	
161	loc_cpu	16	
162	iloc_cpu	16	
163	unl_cpu	28	
164	iunl_cpu	24	
165	dis_dsp	16	
166	ena_dsp	28	
167	sns_ctx	24	
168	sns_loc	24	
169	sns_dsp	24	
170	sns_dpn	24	
171	vsta_knl	84	ISP を初期化後に使用します。
172	ivsta_knl	84	
173	vsys_dwn	32	
174	ivsys_dwn	32	
割込み管理機能			
175	chg_ims	36	
176	ichg_ims	24	
177	get_ims	36	
178	iget_ims	36	
179	カーネル管理割り込みハンドラ	28	カーネル管理割り込みハンドラ終了時に，割り込み発生前のシステム・スタック・ポインタから 28 バイトを使用します。
システム構成管理機能			
180	ref_ver	36	
181	iref_ver	36	
オブジェクト・リセット機能			
182	vrst_dtq	60	
183	vrst_mbx	28	
184	vrst_mbf	32	
185	vrst_mpf	32	
186	vrst_mpl	32	
メモリブジェクト管理機能			
187	ata_mem	64	
188	det_mem	60	
189	sac_mem	76	
190	vprb_mem	40	
191	ref_mem	68	

9.4.3 カーネル・ライブラリをビルドした場合

コンパイラのバージョンやオプション設定を変更してカーネル・ライブラリをビルドした場合，カーネルのスタック使用量が変わる場合があるので，注意してください。

9.5 リアルタイム OS リソース情報パネルに関する注意事項

リアルタイム OS リソース情報パネルにおける注意事項について説明します。

9.5.1 参照はリアルタイム OS 初期化後に行う

リアルタイム OS リソース情報パネルを参照する場合は，リアルタイム OS 初期化後に参照してください。リアルタイム OS が初期化される前は，リアルタイム OS リソース情報パネルの表示が不定となります。

9.5.2 デバッグ情報を生成したプログラムを使用する

リアルタイム OS リソース情報パネルを使用する際は, デバッグ情報を生成したプログラムをダウンロードしてください。デバッグ情報がないプログラムをダウンロードして，リアルタイム OS リソース情報パネルを表示しようとした場合，エラーが発生します。

デバッグ情報を生成するには，“ビルド・ツール”“リンク・オプション”のプロパティで，“デバッグ情報を出力する”を“はい”に設定してください。

9.6 RI600/PX をご使用されていたお客様へ

RI600PX は，RI600/PX を CutbeSuite+対応させたリアルタイム OS 製品です。ここでは，RI600/PX V.1.00 Release 00 からRI600PX V1.01.01への変更点等を示します。

9.6.1 制限事項の解除

システム・コンフィギュレーション・ファイルの interrupt_vector[]および interrupt_fvector[]の pragma_switch に"NOACC"を指定できない制限事項を解除しました。

9.6.2 カーネルのバージョン情報

項目	変更前 (RI600/PX)	変更後 (RI600PX)
TKERNEL_PRVER, ref_ver および iref_ver で返る T_RVER.prver	0x0100	0x0111

9.6.3 CubeSuite+に関する注意事項

CubeSuite+で RI600/PX の High-performance Embedded Workshop のプロジェクトを変換した場合は, 必ず「リビルド」を行ってください。

第10章　制限事項

本章では，RI600PX V1.01.01の制限事項について説明します。

10.1　CubeSuite+使用時の制限事項

CubeSuite+ V2.00.00 使用時には，以下の制限があります。

10.1.1　サービス・コール情報ファイル格納パス

(1)　内容

システム・コンフィギュレーション・ファイルの[プロパティ・パネル]で以下の設定を行った場合，ビルドで生成されるロードモジュールが不正となる場合があります。その場合，一部のサービス・コールがエラーE_NOSPT となります。

- システム・コンフィギュレーション・ファイルの[プロパティ・パネル]の設定

タブ	カテゴリ	項目	設定値
システム・コンフィギュレーション・ファイル関連情報	サービス・コール情報ファイル	サービス・コール情報ファイル格納パス	空白を含む文字列

(2)　回避策

[サービス・コール情報ファイル格納パス]には，空白を含まないパスを設定してください。

または，ビルド前に必要なサービス・コール情報ファイルをビルド・モードのフォルダにコピーしてください。

10.1.2　リアルタイム OS リソース情報パネル

(1)　内容

「RI600PX を使用したアプリケーションのロードモジュール」と，「RI600PX を使用していないアプリケーションのロードモジュール」を同時にダウンロードした場合，リアルタイム OS リソース情報パネルに誤った情報が表示される場合があります。

(2)　発生条件

実行停止時の PC(プログラム・カウンタ)が「RI600PX を使用したアプリケーションのロードモジュール」の範囲外の場合に，発生します。

(3)　回避策

実行停止時の PC(プログラム・カウンタ)が「RI600PX を使用したアプリケーションのロードモジュール」の範囲外の場合は，リアルタイム OS リソース情報パネルに表示される情報は無視してください。

10.1.3 リアルタイム OS リソース情報パネルの「残り時間」

(1) 内容

リアルタイム OS リソース情報パネルの以下の項目に表示される値が，本来の値より最大で TIC_NUME だけ大きくなる場合があります。

- [タスク]タブの[残り時間]
- [周期ハンドラ]タブの[残り時間]
- [アラーム・ハンドラ]タブの[残り時間]

(2) 回避策

本来の値は，以下のように算出してください。

- 表示された値＞TIC_NUME の場合
  本来の値＝([残り時間]に表示された値) - TIC_NUME
- 表示された値≦TIC_NUME の場合
  本来の値＝0

第11章　　ドキュメント訂正

RI600PX V1.01.01にドキュメントの訂正はありません。

第12章　サンプル・プログラム

本章では，RI600PX V1.01.01が提供するサンプル・プログラム「RX630_RI600PX」について説明します。

12.1　概要

ドメインは，「マスタ・ドメイン」，「ドメイン A」，「ドメイン B」の 3 つです。

マスタ・ドメイン（domid #1）は，「信頼されたドメイン」です。マスタ・ドメインは，ドメイン A・B の実行に必要な各種オブジェクトを動的に生成します。マスタ・ドメインに属するタスク（MasterDom_Task）は，システム・コンフィギュレーション・ファイルによって静的に生成・起動されます。

ドメイン A（domid #2）とドメイン B（domid #3）は，「信頼されたドメイン」ではありません。

ドメイン A に属するタスクは AppDomA_Task，ドメイン B に属するタスクは AppDomB_Task です。

AppDomA_Task と AppDomB_Task は，セマフォ（ID_SEM1）を使って排他制御しながら共有変数 g_ulSharedData をアクセスします。

また，AppDomA_Task は，データ・キュー（ID_DTQ1）にデータを送信し，AppDomB_Task はそれを受信します。

表12.1　オブジェクト一覧

種別	IDなど	解説
ドメイン	1	マスタ・ドメイン 信頼されたドメイン 所属タスク：MasterDom_Task
	2	ドメイン A 信頼されていないドメイン 所属タスク：AppDomA_Task
	3	ドメイン B 信頼されていないドメイン 所属タスク：AppDomB_Task
タスク	ID_MASTERDOMTASK	システム・コンフィギュレーション・ファイルで生成・起動される。
	ID_DOM_A_TASK	MasterDom_Taskによって生成・起動される。
	ID_DOM_B_TASK	MasterDom_Taskによって生成・起動される。
セマフォ	ID_SEM1	MasterDom_Task によって，生成される。 AppDomA_Task と AppDomB_Task からの変数"g_ulSharedData"へのアクセスの排他制御に使用。
データ・キュー	ID_DTQ1	MasterDom_Task によって，生成される。 AppDomA_Task と AppDomB_Task の間での通信に使用。 データ・キュー領域は BS セクションとして生成される。BS セクションは非メモリ・オブジェクト。
可変長メモリ・プール	ID_MPL1	MasterDom_Task によって，生成される。 単なるダミー プール領域は BU_SH セクションであり，memory_object[4]内。
周期ハンドラ	ID_CYC1	システム・コンフィギュレーション・ファイルで生成・開始される。 AppDomA_Task と AppDomB_Task のレディキューを回転する。
アラーム ハンドラ	ID_ALM1	システム・コンフィギュレーション・ファイルで生成される。 単なるダミー
割り込み ハンドラ	可変ベクタ#64	システム・コンフィギュレーション・ファイルで定義される。 単なるダミー

12.2 ファイル構成

「RX630_RI600PX」サンプル・プログラムは，以下のフォルダに格納されています。

```
<CubeSuite+_root>¥SampleProjects¥RX¥RX630_RI600PX
```

- <CubeSuite+_root>
  CubeSuite+ のインストール・フォルダを表しています。
  デフォルトでは，” C:¥Program Files¥Renesas Electronics¥CubeSuite+”となります。

(1) appli¥source¥reset フォルダ

- resetprg.c
  ブート処理ファイルです。ブート処理ファイルについては，「RI600PX リアルタイム・オペレーティング・システム ユーザーズマニュアル コーディング編」の 17.2 節を参照してください。
- dbsct.c
  セクション情報ファイルです。セクション情報ファイルについては，「RI600PX リアルタイム・オペレーティング・システム ユーザーズマニュアル コーディング編」の 17.4 節を参照してください。

(2) appli¥source¥kernel フォルダ

- sample.cfg
  システム・コンフィギュレーション・ファイルです。システム・コンフィギュレーション・ファイルについては，「RI600PX リアルタイム・オペレーティング・システム ユーザーズマニュアル コーディング編」の第 20 章を参照してください。
- access_exc.c
  アクセス例外ハンドラです。アクセス例外ハンドラについては，「RI600PX リアルタイム・オペレーティング・システム ユーザーズマニュアル コーディング編」の 3.10 節を参照してください。
- init_cmt.c
  基本クロック用タイマ初期化ルーチンです。基本クロック用タイマ初期化ルーチンについては，「RI600PX リアルタイム・オペレーティング・システム ユーザーズマニュアル コーディング編」の 10.9 節を参照してください。
- sysdwn.c
  システム・ダウン・ルーチンです。システム・ダウン・ルーチンについては，「RI600PX リアルタイム・オペレーティング・システム ユーザーズマニュアル コーディング編」の 15.2 節を参照してください。
- handler.c
  各種ハンドラです。

(3) appli¥source¥master_dom フォルダ

- master_dom.c
  マスタ・ドメインに所属するタスクです。

(4) appli¥source¥dom_A フォルダ

- dom_A.c
  ドメイン A に所属するタスクです。

(5) appli¥source¥dom_B フォルダ

- dom_B.c
  ドメイン B に所属するタスクです。

(6) appli¥source¥common フォルダ

- common.c
  ドメイン間で共有する関数とデータです。

12.3 メモリ・マップ

[ ]で示したセクションは，16 バイト境界に配置するためにリンカの aligned_section オプションを指定しています。詳細は，「RI600PX リアルタイム・オペレーティング・システム ユーザーズマニュアル コーディング編」マニュアルの 2.6.4 節を参照してください。

12.3.1 RAM 領域

表12.2 RAM 領域

アドレス	セクション並び（リンカ設定）	解説	メモリ・オブジェクト
0～0x0001FFFF	SI	システム・スタック	非メモリ・オブジェクト
	BRI_RAM,RRI_RAM	RI600PXデータ	
	BS,BS_1,BS_2,RS,RS_1,RS_2	ハンドラ専用データ	
	[SURI_STACK]	ユーザ・スタック	
	[BU_MASTERDOM],BU_MASTERDOM_1, BU_MASTERDOM_2, RU_MASTERDOM,RU_MASTERDOM_1, RU_MASTERDOM_2	マスタ・ドメイン専用データ	memory_object[1]
	[BU_DOM_A],BU_DOM_A_1,BU_DOM_A_2, RU_DOM_A,RU_DOM_A_1,RU_DOM_A_2	ドメインA専用データ	memory_object[2]
	[BU_DOM_B],BU_DOM_B_1,BU_DOM_B_2, RU_DOM_B,RU_DOM_B_1,RU_DOM_B_2	ドメインB専用データ	memory_object[3]
	[BURI_HEAP],BU_SH,BU_SH_1,BU_SH_2, RU_SH,RU_SH_1,RU_SH_2	ドメイン間共有データ	memory_object[4]

12.3.2 ROM 領域

表12.3 ROM 領域

アドレス	セクション並び（リンカ設定）	解説	メモリ・オブジェクト
0xFFFF0000～ 0xFFFFFF7F	[PU_MASTERDOM],CU_MASTERDOM, CU_MASTERDOM_1,CU_MASTERDOM_2, DU_MASTERDOM, DU_MASTERDOM_1,DU_MASTERDOM_2	マスタ・ドメイン専用コード・定数	memory_object[5]
	[PU_DOM_A],CU_DOM_A,CU_DOM_A_1, CU_DOM_A_2,DU_DOM_A,DU_DOM_A_1, DU_DOM_A_2	ドメインA専用コード・定数	memory_object[6]
	[PU_DOM_B],CU_DOM_B,CU_DOM_B_1, CU_DOM_B_2,DU_DOM_B,DU_DOM_B_1, DU_DOM_B_2	ドメインB専用コード・定数	memory_object[7]
	[PU_SH],WU_SH,WU_SH_1,WU_SH_2,LU_SH, CU_SH,CU_SH_1,CU_SH_2, DU_SH,DU_SH_1,DU_SH_2	ドメイン間共有コード・定数	memory_object[8]
	INTERRUPT_VECTOR	可変ベクタ・テーブル	非メモリ・オブジェクト
	PRI_KERNEL	RI600PXコード	
	CRI_ROM,DRI_ROM	RI600PX定数	
	C$*,PS,CS,CS_1,CS_2,DS,DS_1,DS_2	ハンドラ専用コード・定数	
0xFFFFFF80～ 0xFFFFFFFF	FIX_INTERRUPT_VECTOR	固定ベクタ・テーブル	

12.3.3 メモリ・オブジェクト

メモリ・オブジェクトは全部で 8 個あり，システム・コンフィギュレーション・ファイルで登録します。
以下に，システム・コンフィギュレーション・ファイルでのメモリ・オブジェクト登録内容を示します。

(1) memory_object[1]：マスタ・ドメイン専用データ

```
memory_object[1]{
    start_address = BU_MASTERDOM;
    end_address   = RU_MASTERDOM_2;
    acptn1        = 0x0001;          マスタドメインのみオペランド・リード・アクセス許可
    acptn2        = 0x0001;          マスタドメインのみオペランド・ライト・アクセス許可
    acptn3        = 0;               全ドメインの実行アクセス禁止
};
```

(2) memory_object[2]：ドメイン A 専用データ

```
memory_object[2]{
    start_address = BU_DOM_A;
    end_address   = RU_DOM_A_2;
    acptn1        = 0x0002;          ドメイン A のみオペランド・リード・アクセス許可
    acptn2        = 0x0002;          ドメイン A のみオペランド・ライト・アクセス許可
    acptn3        = 0;               全ドメインの実行アクセス禁止
};
```

(3) memory_object[3]：ドメイン B 専用データ

```
memory_object[3]{
    start_address = BU_DOM_B;
    end_address   = RU_DOM_B_2;
    acptn1        = 0x0004;          ドメイン B のみオペランド・リード・アクセス許可
    acptn2        = 0x0004;          ドメイン B のみオペランド・ライト・アクセス許可
    acptn3        = 0;               全ドメインの実行アクセス禁止
};
```

(4) memory_object[4]：ドメイン間共有データ

```
memory_object[4]{
    start_address = BURI_HEAP;
    end_address   = RU_SH_2;
    acptn1        = TACP_SHARED;     全ドメインのオペランド・リード・アクセス許可
    acptn2        = TACP_SHARED;     全ドメインのオペランド・ライト・アクセス許可
    acptn3        = 0;               全ドメインの実行アクセス禁止
};
```

(5) memory_object[5]：マスタ・ドメイン専用コード・定数

```
memory_object[5]{
    start_address = PU_MASTERDOM;
    end_address   = DU_MASTERDOM_2;
    acptn1        = 0x0001;          マスタドメインのみオペランド・リード・アクセス許可
    acptn2        = 0;               全ドメインのオペランド・ライト・アクセス禁止
    acptn3        = 0x0001;          マスタドメインのみ実行アクセス許可
};
```

(6) memory_object[6]：ドメイン A 専用コード・定数

```
memory_object[6]{
    start_address = PU_DOM_A;
    end_address   = DU_DOM_A_2;
    acptn1        = 0x0002;          ドメイン A のみオペランド・リード・アクセス許可
    acptn2        = 0;               全ドメインのオペランド・ライト・アクセス禁止
    acptn3        = 0x0002;          ドメイン A のみ実行アクセス許可
};
```

(7) memory_object[7]：ドメイン B 専用コード・定数

```
memory_object[7]{
    start_address = PU_DOM_B;
    end_address   = DU_DOM_B_2;
    acptn1        = 0x0004;          ドメイン B のみオペランド・リード・アクセス許可
    acptn2        = 0;               全ドメインのオペランド・ライト・アクセス禁止
    acptn3        = 0x0004;          ドメイン B のみ実行アクセス許可
};
```

(8) memory_object[8]：ドメイン間共有コード・定数

```
memory_object[8]{
    start_address = PU_SH;
    end_address   = DU_SH_2;
    acptn1        = TACP_SHARED;     全ドメインのオペランド・リード・アクセス許可
    acptn2        = 0;               全ドメインのオペランド・ライト・アクセス禁止
    acptn3        = TACP_SHARED;     全ドメインの実行アクセス許可
};
```

12.3.4 ユーザ・スタック

ユーザ・スタックはメモリ・オブジェクト外とする必要があります。本サンプルでは，全タスクのユーザ・スタックをデフォルトと同じ SURI_STACK セクションとしています。

(1) MasterDom_Task のユーザ・スタック

MasterDom_Task は，システム・コンフィギュレーション・ファイルで静的に生成されます。

```
task[]{
    name           = ID_MASTERDOMTASK;
    entry_address   = MasterDom_Task();
    initial_start   = ON;
    stack_size      = 256;
    priority        = 1;
//  stack_section   = SURI_STACK;
    exinf          = 1;
    domain_num      = 1;
};
```

"stack_section"を省略すると，ユーザ・スタックは SURI_STACK セクションに生成されます。

(2) AppDomA_Task と AppDomB_Task のユーザ・スタック

AppDomA_Task と AppDomB_Task は，MasterDom_Task が呼び出す acre_tsk サービス・コールによって動的に生成されます。acre_tsk には，それぞれのタスクのユーザ・スタックの先頭アドレスとサイズを渡します。

AppDomA_Task と AppDomB_Task のスタック領域は，共に master_dom.c にて#pragma section ディレクティブを使って SURI_STACK セクションに生成しています。

```
/////////////////////////////////////////////////////////////////////////
// Stack for AppDomA_Task and AppDomB_Task
/////////////////////////////////////////////////////////////////////////
#pragma section B SURI_STACK
static UW   s_ulDomA_Stk[DOM_A_STKSZ/sizeof(UW)];   // Stack area for AppDomA_Task
static UW   s_ulDomB_Stk[DOM_B_STKSZ/sizeof(UW)];   // Stack area for AppDomB_Task
```

12.4 セクションに関するビルド・ツールの設定

12.4.1 標準ライブラリ構築ツール

標準ライブラリのセクションは全ドメインからアクセス可能なメモリ・オブジェクトとしています。

表12.4 標準ライブラリのセクション

領域	セクション	メモリ・オブジェクト
コード	PU_SH	memory_object[8]
定数	CU_SH	
リテラル	LU_SH	
switch文分岐テーブル	WU_SH,WU_SH_1,WU_SH_2	
初期化データ	DU_SH,DU_SH_1,DU_SH_2	
未初期化データ	BU,BU_SH_1,BU_SH_2	memory_object[4]
初期化データ（RAM） （リンカで指定）	RU_SH,RU_SH_1,RU_SH_2	

12.4.2 C/C++コンパイラ

デフォルトのセクションは表 12.4と同じとし，これ以外のセクションとする場合には，個別にソース・コードに#pragma section ディレクティブを記述することでセクションを切り替えることとしています。

12.4.3 リンカ

「RI600PX リアルタイム・オペレーティング・システム ユーザーズマニュアル コーディング編」マニュアルの 2.6.4 節に記載のように，以下のセクションについて，aligned_section オプションを指定する必要があります。

- システム・コンフィギュレーション・ファイルで，memory_object[].start_address に指定したセクション
- システム・コンフィギュレーション・ファイルで，task[].stack_section に指定したセクション（本サンプルでは指定していません）
- SURI_STACK セクション

このため，メモリ・オブジェクトの先頭セクションと SURI_STACK に aligned_section を指定しています。

12.5　アクセス違反の対処例

本サンプルでは，以下の例を実装しています。詳細はサンプル・コードを参照してください。

- AppDomA_Task：タスク例外を発生させ，タスク例外処理ルーチンでタスクを再起動する。
- AppDomB_Task：longjmp( )を使って，正常な時点からやり直す。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して、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。
2. 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。
3. 本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害に関し、当社は、何らの責任を負うものではありません。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。
4. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、
   各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   標準水準： コンピュータ、OA機器、通信機器、計測機器、AV機器、
   家電、工作機械、パーソナル機器、産業用ロボット等
   高品質水準：輸送機器（自動車、電車、船舶等）、交通用信号機器、
   防災・防犯装置、各種安全装置等
   当社製品は、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。たとえ、意図し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損害が生じても、当社は一切その責任を負いません。なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の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。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は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、社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様 の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。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。
9. 本資料に記載されている当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。また、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に使用しないでください。当社製品または技術を輸出する場合は、「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。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。

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。

RENESAS

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

※営業お問合せ窓口の住所・電話番号は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス販売株式会社 〒100-0004 千代田区大手町 2-6-2（日本ビル） (03)5201-5307

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/contact/