RX ファミリ リアルタイム OS RI600V4 V1.05.00 リリースノート

R20UT4376JJ0100
Rev.1.00
2018.10.01

この度は、RX リアルタイム OS RI600V4 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

1. 製品構成

RI600V4 は型名により、以下のように契約形態と提供物が異なります。

- トライアル版

型 名	契約形態
RTRRX0000TR01ERRZZ	トライアル版、インストール可能な PC は 1 台

提供物は、リアルタイム OS RI600V4 カーネル オブジェクトのトライアル版（使用制限あり）、および、コマンドライン・コンフィギュレータ CFG600 で、Web サイトからのダウンロードによる提供となります。

なお、CS+ for CC プラグインや e^2 studio を使用する際は、Web サイトから個別にダウンロードしてください。

- 評価契約・量産契約

型 名	契約形態	提供物
R0R5RX00TCW011	評価契約、インストール可能な PC は 1 台	A
R0R5RX00TCW01A	評価契約、インストール可能な PC は無制限	A
R0R5RX00TCW01K	量産契約、量産数は 3000 台まで	A
R0R5RX00TCW01U	量産契約、量産数は無制限	A
R0R5RX00TCW01Z	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 RI600V4 カーネル オブジェクト	V1.05.00
		コマンドライン・コンフィギュレータ CFG600	V1.03.00.002
		CS+ for CC 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）	V3.02.01.01
		リアルタイム OS ビルド設定プラグイン（RI600V4 依存部）	V3.00.00.06
		リアルタイム OS 解析制御プラグイン（共通部）	V3.00.00.03
		リアルタイム OS 解析制御プラグイン（uITRON4 依存部）	V3.00.00.02
		リアルタイム OS 解析制御プラグイン（RI600V4 依存部）	V3.00.00.03
		リアルタイム OS リソース情報表示プラグイン（共通部）	V3.01.00.01
		リアルタイム OS リソース情報表示プラグイン（uITRON4 依存部）	V3.00.00.06
		リアルタイム OS タスク・アナライザ・プラグイン（共通部）	V3.00.02.01
		リアルタイム OS タスク・アナライザ・プラグイン（解析結果パネル）	V3.01.00.08
		リアルタイム OS タスク・アナライザ・プラグイン（RI600V4 依存部）	V3.00.00.02
		GUI コンフィギュレータ GUI600	V 1.01.00.004
		リアルタイム OS RI600V4 カーネル ソース・コード	V1.05.00

2. 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を以下に示します。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	R20UT0751JJ0106
RI600V4 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	R20UT0711JJ0104
RI600V4 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	R20UT0775JJ0101
RI600V4 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 解析編	R20UT2185JJ0101
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	R20UT0756JJ0105

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。本製品をインストール後、Windows のスタートメニューからユーザーズマニュアルを参照できます。

また、ルネサスエレクトロニクスのホームページからユーザーズマニュアルを入手することができます。なお、提供媒体のないトライアル版はルネサスエレクトロニクスのホームページから入手してください。

2.1. ユーザーズマニュアル正誤表

ユーザーズマニュアルの正誤情報を以下に示します。

マニュアル名	資料番号	箇所	誤	正
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	R20UT0751JJ0106	Page 31 表 1—1	RI600V4 では EZ Emulator を 使用できます。	RI600V4 では EZ Emulator を 使用できません。
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	R20UT0751JJ0106	Page 32 表 1—3	RI600V4 では EZ Emulator を 使用できます。	RI600V4 では EZ Emulator を 使用できません。

3. 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。

- RX700 シリーズ MCU
- RX600 シリーズ MCU
- RX200 シリーズ MCU
- RX100 シリーズ MCU

4. 動作環境

本製品を使用するには、以下の環境が必要になります。

4.1. ハードウェア環境

以下のハードウェア環境に対応しています。

- 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®では最低 2GB）
- 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

以下の OS に対応しています。

- Windows 7（32 ビット版、64 ビット版） : CS+, e^2 studio
- Windows 8.1（32 ビット版、64 ビット版） : CS+, e^2 studio
- Windows Vista（32 ビット版、64 ビット版） : CS+
- Windows 10（32 ビット版、64 ビット版） : CS+, e^2 studio

※何れの OS 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。

以下のランタイム・ライブラリが必要です。

- .NET Framework 4.5.2
- Microsoft Visual C++ 2010 SP1 ランタイム・ライブラリ

4.3. 対応ツール

本製品は以下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 CS+ for CC	ルネサス エレクトロニクス	V3.02.00 以降
統合開発環境 e^2 studio	ルネサス エレクトロニクス	V4.2 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 V2.04.01 以降を推奨

4.4. リアルタイム OS タスク・アナライザの動作環境に関する注意

リアルタイム OS タスク・アナライザを「ハードウェア・トレース・モードでトレース・チャートを取得」で使用するには、以下のいずれかのデバッグ・ツールが必要です。

- シミュレータ
- トレース機能を有するエミュレータ

  CS+ for CC の場合であれば、トレースの［タイム・スタンプ出力］を［はい］に設定可能なエミュレータがトレース機能を有するエミュレータです。

  ［トレース・データ種別］は［データアクセス］にしてください。

  なお、［タイム・スタンプ出力］および［トレース・データ種別］は、使用するデバッグ・ツールの［プロパティ］パネルの［デバッグ・ツール設定］タブ上の［トレース］カテゴリ内にあります。

  RX100 シリーズで E1 エミュレータを使用する場合は、［タイム・スタンプ出力］は［いいえ］固定のため、リアルタイム OS タスク・アナライザは使用できません。

5. インストールに関する注意事項

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. CS+と e2 studio 環境で共通

5.1.1. インストール時の注意事項

5.1.1.1. 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。

5.1.1.2. 実行環境に関する注意事項

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（CS+ for CC を実行するために必要です）。

5.1.1.3. 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。

5.1.1.4. 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。

5.1.1.5. 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。

コントロールパネルの「プログラムと機能」の［変更］ボタンから行うとエラーになります。

5.1.1.6. 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。

5.1.1.7. 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。

5.1.2. アンインストール時の注意事項

5.1.2.1. 管理者権限に関する注意事項

アンインストール（フォルダ／ファイル削除）するには、Windows®の管理者権限が必要です。

5.1.2.2. 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。

5.1.2.3. 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。

5.2. CS+環境の場合

5.2.1. インストール時の注意事項

5.2.1.1. 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 32 ビット版で、システムドライブが C:の場合
  C:¥Program Files¥Common Files¥Renesas Electronics CubeSuite+¥
- Windows®が 64 ビット版で、システムドライブが C:の場合
  C:¥Program Files (x86)¥Common Files¥Renesas Electronics CubeSuite+¥

5.2.1.2. プラグインの有効化

本製品のインストール直後など、本製品のプラグインが無効になっている場合があります。「8.2.1 プラグインの有効化」にしたがって本製品のプラグインを有効にしてください。

5.2.2. アンインストール時の注意事項

5.2.2.1. 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（CS+ for CC 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

- 「プログラムと機能」

から、以下を削除してください。

- CS+ Realtime OS Common Plugins
- CS+ Realtime OS RI600V4 Plugins
- CS+ Realtime OS RI600V4 Object Release（量産契約、ソース・コード付き「以外」の場合）
- CS+ Realtime OS RI600V4 Source Release（量産契約、ソース・コード付きの場合）
- CS+ Realtime OS RI600V4 Trial（トライアル版）

5.3. e^2 studio 環境の場合

5.3.1. インストール時の注意事項

なし

5.3.2. アンインストール時の注意事項

5.3.2.1. 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、コントロールパネルの

- 「プログラムと機能」

から、以下を削除してください。

- Renesas Realtime OS RI600V4 Object Release（量産契約、ソース・コード付き「以外」の場合）
- Renesas Realtime OS RI600V4 Source Release（量産契約、ソース・コード付きの場合）
- Renesas Realtime OS RI600V4 Trial（トライアル版）

6. 過去リリース版との相違点

本章では、本製品の各バージョンでの変更点を説明します。

6.1. RI600V4 V1.04.00 での変更点

6.1.1. カーネルの相違点

(1) サービス・コール呼び出し方法の変更

e2 studio 対応に伴い、テーブル生成ユーティリティの起動を廃止したため、サービス・コールの呼び出し方法をテーブルジャンプから、通常関数コールに変更しました。ただし、サービス・コールの使い方は従来と変わりません。これに伴い以前の版で構築したビルド環境を変更する必要があります。詳細は「8.1.10 アプリケーションビルド時の注意事項」を参照してください。

また、サービス・コールのスタック使用量が変わります。詳細は「8.1.6 スタック使用量について」を参照してください。

(2) カーネルのバージョン情報

バージョンの変更は、以下の通りです。

項 目	変更前	変更後
TKERNEL_PRVER、 ref_ver および iref_ver で返る T_RVER prver	0x130	0x140

6.1.2. コンフィギュレータの相違点

(1) kernel_id.h に出力する下記の #pragma 記述と、その記述に対応する関数のプロトタイプ宣言記述の順番が逆であった不具合を修正。

- #pragma task
- #pragma cychandler
- #pragma almhandler

6.1.3. リアルタイム OS ビルド設定プラグインの相違点

(1) CS+ for CC に対応

CS+ for CC に対応しました。なお、本プラグインは CubeSuite+上では動作しません。

(2) ［リアルタイム OS］タブ、および［システムコンフィギュレーションファイル関連情報］タブからのヘルプジャンプするように変更しました。

6.1.4. リアルタイム OS リソース情報表示プラグインの相違点

（1） CS+ for CC に対応
CS+ for CC に対応しました。なお、本プラグインは CubeSuite+上では動作しません。

（2） 待ち要因で表示される資源を、ID 番号から名称に変更
待ち要因で表示される資源を、今までは ID 番号で表示していましたが、今版では名称に変更して判別しやすくしました。

（3） リソース選択タブの視認性を向上
リソースを選択するタブを二段にし、さらにリソース名の前にアイコンを付加することで、視認性を向上しました。

（4） メッセージを一部改善
エラー時などに表示されるメッセージを一部改善しました。

（5） 表示メニュー、または、ツールバーのボタンを選択してリアルタイム OS リソース情報パネルを開いても、パネルがアクティブにならない制限を解除しました。

6.1.5. リアルタイム OS タスク・アナライザ・プラグインの相違点

（1） CS+ for CC に対応
CS+ for CC に対応しました。なお、本プラグインは CubeSuite+上では動作しません。

（2） オブジェクト情報エリアのオブジェクトの入れ替え操作（ドラック＆ドロップ）ができないことがある制限を解除しました。

6.1.6. CS+用サンプル・プログラムの相違点

（1） Firmware Integration Technology モジュールを組み込んだサンプル・プログラムを新規に追加
FIT モジュールを組み込んだ CS+用のサンプル・プログラムを新規に追加しました。詳細は、「10.1 Firmware Integration Technology モジュールを組み込んだサンプル・プログラム」を参照してください。

6.1.7. システム・ダウン・ルーチンの相違点

（1） システム・ダウン・ルーチンで発生しなくなったエラー
RI600V4 V1.04.00 以降では、システム・ダウン・ルーチン（_RI_sys_dwn__）で type = -3（組み込まれていないサービス・コールの呼び出し）のエラーは内部処理の変更により発生しなくなりました。

6.2. RI600V4 V1.05.00 での変更点

6.2.1. カーネルの相違点

（1） RXv3 アーキテクチャのサポート
RXv3 アーキテクチャのサポートに伴い、RXv3 アーキテクチャ利用時に RXv2 用カーネル・ライブラリとリンクするよう変更しました。RXv3 アーキテクチャは RXv2 アーキテクチャと互換性があります。
RI600V4 ユーザーズマニュアル コーディング編（R20UT0711JJ0104）における 2.6.3 カーネル・ライブラリの表 2−1 カーネル・ライブラリは以下に示す表の通りに読み替えてください。

	フォルダ	対応コンパイラ・バージョン	対応 CPU コア	ファイル名	説明
1	<ri_root>¥library¥rxv1	V1.02.01 以降	・RXv1 アーキテクチャ	ri600lit.lib	リトル・エンディアン用
				ri600big.lib	ビッグ・エンディアン用
2	<ri_root>¥library¥rxv2	V2.01.00 以降	・RXv1 アーキテクチャ ・RXv2 アーキテクチャ ・RXv3 アーキテクチャ	ri600lit.lib	リトル・エンディアン用
				ri600big.lib	ビッグ・エンディアン用

（2） カーネルのバージョン情報

バージョンの変更は、以下の通りです。

項 目	変更前	変更後
TKERNEL_PRVER、 ref_ver および iref_ver で返る T_RVER prver	0x140	0x150

6.2.2. CS+用サンプル・プログラムの相違点

（1） RX66T 用サンプル・プログラムを新規に追加

RXv3 アーキテクチャのサポートに伴い、RX66T デバイスを利用した CS+用のサンプル・プログラムを新規に追加しました。

7. 最新版のプラグインについて

プラグインは予告なくアップデートされることがあります。CS+のアップデート機能により最新のプラグインへアップデートを行ってください。

アップデートした製品は e^2 studio ではご利用頂けませんので、ご注意ください。

8. 注意事項

8.1. CS+と e^2 studio で共通の注意事項

8.1.1. カーネル・バージョンの区別について

以下の変数を参照することで、カーネル・バージョンを区別することができます。

```
const UW _RI600V4_VERSION = <設定値>;
```

カーネルのバージョンは、X.YY.ZZ.aa の形式で表されます。設定値のビット 31～24 が X、ビット 23～16 が YY、ビット 15～8 が ZZ、ビット 7～0 が aa を表します。

実際のバージョンは以下になります。

カーネル・バージョン （製品バージョン）	_RI600V4_VERSION 値	備考
V1.01.00（V1.01.00）	(変数の定義なし)	過去のバージョン
V1.02.00（V1.02.00）	(変数の定義なし)	過去のバージョン
V1.02.01（V1.02.01）	(変数の定義なし)	過去のバージョン
V1.02.02（V1.02.02）	(変数の定義なし)	過去のバージョン
V1.03.00.03（V1.03.00）	0x01030003	過去のバージョン
V1.04.00.00（V1.04.00）	0x01040000	過去のバージョン
V1.05.00.00（V1.05.00）	0x01050000	本バージョン

8.1.2. 以前のバージョンからの移行

RI600V4 の以前のバージョンから移行した場合は、必ずリビルドを行ってください。

8.1.3. GUI コンフィギュレータ

8.1.3.1. 概要

GUI コンフィギュレータは、GUI 画面上で各種カーネル・コンフィギュレーション情報を入力することで、システム・コンフィギュレーション・ファイルを生成するツールです。GUI コンフィギュレータを使用すれば、システム・コンフィギュレーション・ファイルの記法を習得しなくてもカーネルを構築することができます。

GUI コンフィギュレータの使用方法については、オンライン・ヘルプ、または、GUI コンフィギュレータに実装されているヘルプ機能を参照してください。

GUI コンフィギュレータを起動するには、Windows のスタートメニューから「GUI コンフィギュレータ」を実行してください。

- GUI コンフィギュレータ実行ファイルのファイルパス

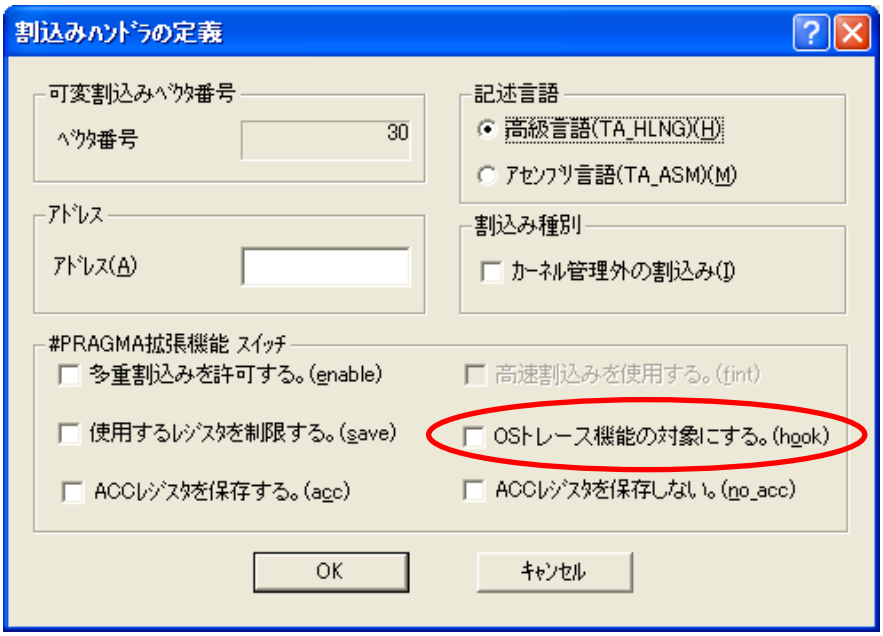
  <インストール・フォルダ>¥bin600¥Guiconfig_RI600.exe

8.1.3.2. ［割り込みハンドラの定義］ダイアログ・ボックス

［割り込みハンドラの定義］ダイアログ・ボックスの［OS トレース機能の対象にする（hook）］チェック・ボックスの設定は無視されます。

図 8

-1 割り込みハンドラの定義



8.1.4. タイマ・テンプレート・ファイル

以下に、RI600V4 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。

ターゲット・デバイスに対応したタイマ・テンプレート・ファイルの情報は、ウェブ上の RI600V4 製品ページにて随時更新しています。最新の情報についてはウェブ上でご確認ください。

表 8-1 タイマ・テンプレート・ファイル

テンプレート・ファイル	対応 MCU
rx610.tpl	RX600 シリーズ RX610 グループ
rx62t.tpl	RX600 シリーズ RX62T グループ
rx62n.tpl	RX600 シリーズ RX62G グループ RX600 シリーズ RX62N グループ RX600 シリーズ RX621 グループ
rx630.tpl	RX700 シリーズ RX71M グループ *1 RX600 シリーズ RX66T グループ *1 RX600 シリーズ RX65N グループ *1 RX600 シリーズ RX651 グループ *1 RX600 シリーズ RX64M グループ *1 RX600 シリーズ RX630 グループ RX600 シリーズ RX63N グループ RX600 シリーズ RX631 グループ RX600 シリーズ RX634 グループ RX600 シリーズ RX63T グループ RX200 シリーズ RX21A グループ RX200 シリーズ RX230 グループ RX200 シリーズ RX231 グループ RX200 シリーズ RX23T グループ RX200 シリーズ RX24T グループ RX200 シリーズ RX24U グループ
rx210.tpl	RX200 シリーズ RX210 グループ RX200 シリーズ RX220 グループ RX100 シリーズ RX110 グループ *1 RX100 シリーズ RX111 グループ *1 RX100 シリーズ RX113 グループ *1 RX100 シリーズ RX130 グループ *1

*1 システム・コンフィギュレーション・ファイルで、clock.timer に”CMT2”および”CMT3”を指定してはなりません。

8.1.5. カーネル・ソース・コードのビルド方法

RI600V4 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。ソース・コードが付属するのは、ソース付き量産契約版(R0R5RX00TCW01Z)のみです。

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、以下のように”nmake.exe”[1]を実行してください。これにより、<インストール・フォルダ>¥library”下にライブラリが生成されます。

- ”<インストール・フォルダ>¥library¥rxv1”フォルダのライブラリ生成コマンド
  `nmake release_install(RET)`
  備考：製品添付のライブラリは、CC-RX V1.02.01 でビルドされています。
- ”<インストール・フォルダ>¥library¥rxv2”フォルダのライブラリ生成コマンド
  `nmake –f make_rxv2.mak release_install(RET)`
  備考：製品添付のライブラリは、CC-RX V2.01.00 でビルドされています。

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、インストール・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”library¥rxv1”または”library¥rxv2” フォルダにコピーしてください。

1 “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。

8.1.6. スタック使用量について

8.1.6.1. 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*)

「RI600V4 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*=136
- *clocksz2*=136
- *clocksz3*=204

8.1.6.2. サービス・コールのスタック使用量(*svcsz*)

サービス・コールでは、以下のようにスタックを使用します。

（1） タスク・コンテキストから呼び出された場合
タスク・コンテキスト実行中のスタックはユーザ・スタックです。サービス・コールでは、
（a）ユーザ・スタック（呼び出し元スタック）
（b）システム・スタック
を使用します。

（2） 非タスク・コンテキストから呼び出された場合
非タスク・コンテキスト実行中のスタックはシステム・スタックです。サービス・コールでは、
（c）システム・スタック（呼び出し元スタック）
を使用します。

サービス・コールが使用する呼び出し元のスタック((a),(c))の使用量は、Call Walker によって表示されます。

また、(b)および(c)のサイズは、「RI600V4 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 節に記載のようにシステム・スタックの使用量を算出するために必要となります（付録 D.4 節では *svcsz* と表記しています）。以下に、各サービス・コールの(a)～(c)のサイズを示します。

表 8-2 サービス・コールのスタック使用量

	サービス・コール	ユーザ・スタック使用サイズ（a）	システム・スタック使用サイズ（b）（c）	
タスク管理機能				
1	act_tsk	4	44	
2	iact_tsk	0	52	
3	can_act	4	44	
4	ican_act	0	48	
5	sta_tsk	4	44	
6	ista_tsk	0	48	
7	ext_tsk	0	60	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。
8	ter_tsk	4	116	
9	chg_pri	4	44	

	サービス・コール	ユーザ・スタック使用サイズ（a）	システム・スタック使用サイズ（b）（c）	
タスク管理機能				
10	ichg_pri	0	60	
11	get_pri	4	44	
12	iget_pri	0	48	
13	ref_tsk	4	44	
14	iref_tsk	0	48	
15	ref_tst	4	44	
16	iref_tst	0	48	
タスク付属同期機能				
17	slp_tsk	4	44	
18	tslp_tsk	4	44	
19	wup_tsk	4	44	
20	iwup_tsk	0	52	
21	can_wup	4	44	
22	ican_wup	0	48	
23	rel_wai	4	112	
24	irel_wai	0	132	
25	sus_tsk	4	44	
26	isus_tsk	0	48	
27	rsm_tsk	4	44	
28	irsm_tsk	0	48	
29	frsm_tsk	4	44	
30	ifrsm_tsk	0	48	
31	dly_tsk	4	44	
セマフォ				
32	sig_sem	4	44	
33	isig_sem	0	60	
34	wai_sem	4	44	
35	pol_sem	4	44	
36	ipol_sem	0	48	
37	twai_sem	4	44	
38	ref_sem	4	44	
39	iref_sem	0	48	

	サービス・コール	ユーザ・スタック使用サイズ（a）	システム・スタック使用サイズ（b）（c）	
イベントフラグ				
40	set_flg	4	48	
41	iset_flg	0	68	
42	clr_flg	4	44	
43	iclr_flg	0	48	
44	wai_flg	4	48	
45	pol_flg	4	44	
46	ipol_flg	0	48	
47	twai_flg	4	48	
48	ref_flg	4	44	
49	iref_flg	0	48	
データキュー				
50	snd_dtq	4	44	
51	psnd_dtq	4	44	
52	ipsnd_dtq	0	60	
53	tsnd_dtq	4	44	
54	fsnd_dtq	4	44	
55	ifsnd_dtq	0	60	
56	rcv_dtq	4	44	
57	prcv_dtq	4	44	
58	iprcv_dtq	0	64	
59	trcv_dtq	4	44	
60	ref_dtq	4	44	
61	iref_dtq	0	48	
メールボックス				
62	snd_mbx	4	44	
63	isnd_mbx	0	60	
64	rcv_mbx	4	44	
65	prcv_mbx	4	44	
66	iprcv_mbx	0	48	
67	trcv_mbx	4	44	
68	ref_mbx	4	44	
69	iref_mbx	0	48	

	サービス・コール	ユーザ・スタック使用サイズ（a）	システム・スタック使用サイズ（b）（c）	
ミューテックス				
70	loc_mtx	4	44	
71	ploc_mtx	4	44	
72	tloc_mtx	4	44	
73	unl_mtx	4	52	
74	ref_mtx	4	44	
メッセージ・バッファ				
75	snd_mbf	4	44	
76	psnd_mbf	4	44	
77	ipsnd_mbf	0	64	
78	tsnd_mbf	4	44	
79	rcv_mbf	4	56	
80	prcv_mbf	4	56	
81	trcv_mbf	4	56	
82	ref_mbf	4	44	
83	iref_mbf	0	48	
固定長メモリ・プール				
84	get_mpf	4	48	
85	pget_mpf	4	44	
86	ipget_mpf	0	48	
87	tget_mpf	4	48	
88	rel_mpf	20	44	
89	irel_mpf	0	64	
90	ref_mpf	4	44	
91	iref_mpf	0	48	
可変長メモリ・プール				
92	get_mpl	28	88	
93	pget_mpl	4	104	
94	ipget_mpl	0	108	
95	tget_mpl	28	88	
96	rel_mpl	4	104	
97	ref_mpl	4	44	
98	iref_mpl	0	48	

	サービス・コール	ユーザ・スタック使用サイズ（a）	システム・スタック使用サイズ（b）（c）	
時間管理機能				
99	set_tim	4	44	
100	iset_tim	0	48	
101	get_tim	4	44	
102	iget_tim	0	48	
周期ハンドラ				
103	sta_cyc	4	44	
104	ista_cyc	0	48	
105	stp_cyc	4	44	
106	istp_cyc	0	48	
107	ref_cyc	4	44	
108	iref_cyc	0	48	
アラームハンドラ				
109	sta_alm	4	44	
110	ista_alm	0	48	
111	stp_alm	4	44	
112	istp_alm	0	48	
113	ref_alm	4	44	
114	iref_alm	0	48	
システム状態管理機能				
115	rot_rdq	4	44	
116	irot_rdq	0	48	
117	get_tid	4	44	
118	iget_tid	0	48	
119	loc_cpu	4	44	
120	iloc_cpu	0	48	
121	unl_cpu	4	44	
122	iunl_cpu	0	48	
123	dis_dsp	4	44	
124	ena_dsp	4	44	
125	sns_ctx	0	48	
126	sns_loc	0	48	
127	sns_dsp	0	48	
128	sns_dpn	0	48	
129	vsta_knl	0	40	システム・スタック・ポインタを初期化後に使用します。
130	ivsta_knl	0	40	
131	vsys_dwn	4	44	
132	ivsys_dwn	0	48	

	サービス・コール	ユーザ・スタック使用サイズ（a）	システム・スタック使用サイズ（b）（c）	
割り込み管理機能				
133	chg_ims	4	44	
134	ichg_ims	0	48	
135	get_ims	4	4	
136	iget_ims	4	4	
137	カーネル管理割り込みハンドラ	0	52	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 52 バイトを使用します。
システム構成管理機能				
138	ref_ver	4	44	
139	iref_ver	0	48	
オブジェクト・リセット機能				
140	vrst_dtq	4	44	
141	vrst_mbx	4	44	
142	vrst_mbf	4	44	
143	vrst_mpf	4	44	
144	vrst_mpl	4	72	

8.1.6.3. 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。

8.1.7. RX610 グループ使用時の注意事項

RX610 グループの PSW.IPL は 3 ビット構成のため、以下は必ず 8 未満としてください。

- chg_ims、ichg_ims で指定する割り込みマスク値
- システム・コンフィギュレーション・ファイルの system.system_IPL 設定値
- システム・コンフィギュレーション・ファイルの clock.IPL 設定値

8.1.8. 大域最適化コンパイル・オプションの注意事項

RI600V4 を組み込んだプログラムでは、大域最適化オプション（-ip_optimize、-merge_files、-whole_program）は利用できません。

8.1.9. OS データ初期化の注意事項

RI600V4 V1.04.00 で OS データの初期化方法が変わりました。これに伴い、RI600V4 V1.04.00 より前に作成したユーザプログラム側に以下の変更が必要です。

・リンカのオプション設定

変更前：”-rom=DRI_ROM=PRI_ROM”オプションあり

変更後：”-rom=DRI_ROM=PRI_ROM”オプションなし

・B,R セクションの初期化設定（サンプル・プログラムの dbsct.c）

変更前：

```
#pragma section C C$DSEC
extern const struct {
      _UBYTE *rom_s;
      _UBYTE *rom_e;
      _UBYTE *ram_s;
}     _DTBL[] = {
      { __sectop("D"), __secend("D"), __sectop("R") },
      { __sectop("D_2"), __secend("D_2"), __sectop("R_2") },
      { __sectop("D_1"), __secend("D_1"), __sectop("R_1") },
      { __sectop("DRI_ROM"), __secend("DRI_ROM"), __sectop("RRI_RAM") }
};
#pragma section C C$BSEC
extern const struct {
      _UBYTE *b_s;
      _UBYTE *b_e;
}     _BTBL[] = {
      { __sectop("B"), __secend("B") },
      { __sectop("B_2"), __secend("B_2") },
      { __sectop("B_1"), __secend("B_1") }
};
```

変更後：

```
#pragma section C C$DSEC
extern const struct {
      _UBYTE *rom_s;
      _UBYTE *rom_e;
      _UBYTE *ram_s;
}     _DTBL[] = {
      { __sectop("D"), __secend("D"), __sectop("R") },
      { __sectop("D_2"), __secend("D_2"), __sectop("R_2") },
      { __sectop("D_1"), __secend("D_1"), __sectop("R_1") }
};

#pragma section C C$BSEC
extern const struct {
      _UBYTE *b_s;
      _UBYTE *b_e;
}     _BTBL[] = {
      { __sectop("B"), __secend("B") },
      { __sectop("B_2"), __secend("B_2") },
      { __sectop("B_1"), __secend("B_1") },
      { __sectop("BRI_RAM"), __secend("BRI_RAM") }
};
```

8.1.10. アプリケーションビルド時の注意事項

RI600V4 V1.04.00 以降、mkritbl.exe（テーブル生成ユーティリティ）の起動を廃止し、サービス・コール呼び出し方法をテーブルジャンプから、通常関数コールに変更しました。

これに伴い、V1.04.00 より前にアプリケーションをビルドしていた環境は、以下の手順で mkritbl.exe が出力していた ritable.src を、cfg600.exe（コンフィギュレータ）が出力する ritable.src に変更する必要があります。

- ビルド環境を CubeSuite+のプロジェクトで構築していた場合
  - CS+ V3.00.00 以降でプロジェクトを読み直し、リビルドする。

- ビルド環境を自作（makefile など）で構築していた場合
  - mkritbl.exe の起動を削除
  - リンクする ritable.src を mkritbl.exe が出力するものから cfg600.exe が出力するものに変更

なお、mkritble.exe の入力ファイルであった mrc ファイルは不要となりましたが、V1.04.00 以降も出力されます。このファイルは動作に影響を与えないため無視してください。

8.2. CS+使用時の注意事項

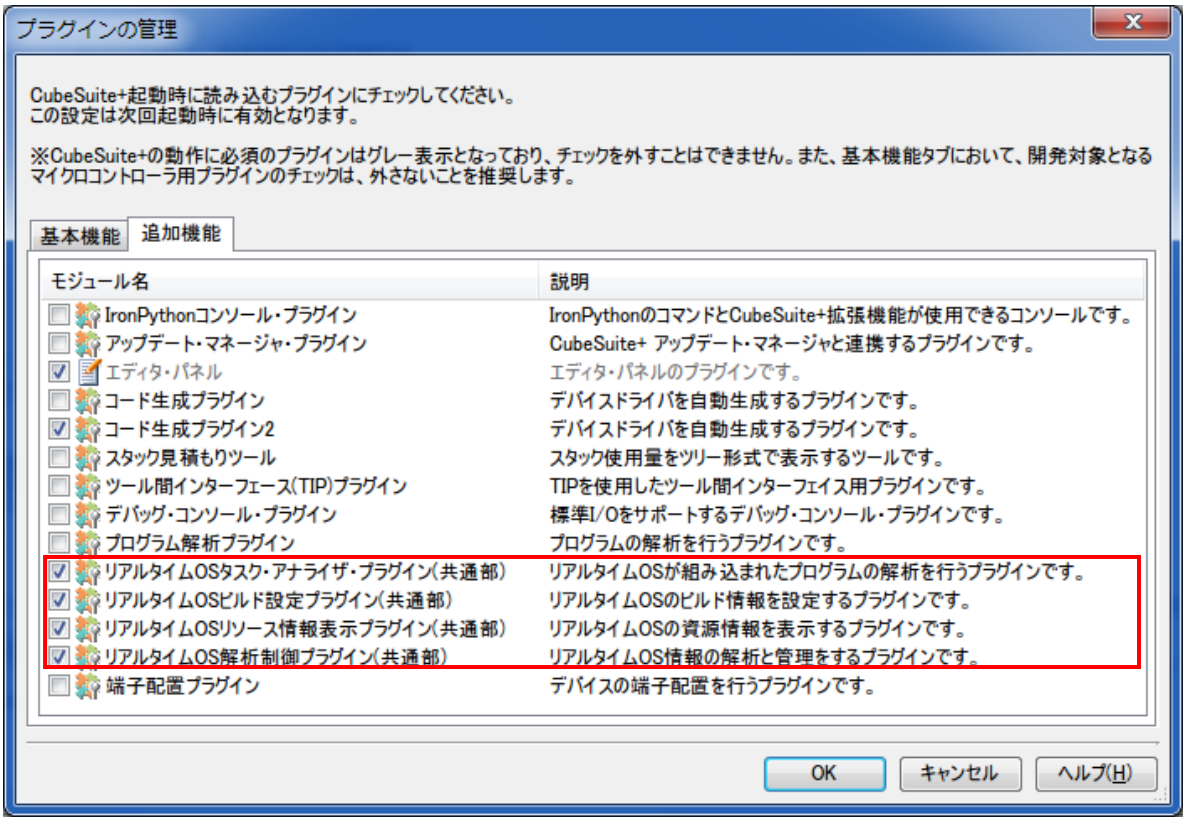
8.2.1. プラグインの有効化

本製品のインストール直後は、本製品のプラグインが CS+ for CC に読み込まれず、無効になっている場合があります。本製品のプラグインが無効になっていると、ビルドできないなどの問題が生じます。

CS+ for CC の［プラグインの管理］ダイアログの［追加機能］タブで、以下のプラグインを有効にしてください。

- リアルタイム OS タスク・アナライザ・プラグイン（共通部）
- リアルタイム OS ビルド設定プラグイン（共通部）
- リアルタイム OS リソース情報表示プラグイン（共通部）
- リアルタイム OS 解析制御プラグイン（共通部）

図 8-2 プラグイン管理



8.2.2. CS+のプロジェクト作成

本製品を使用したプロジェクトを作成するには、以下の 2 つの方法があります。

- 本製品添付のサンプル・プロジェクトを流用する
- 新しいプロジェクトを作成する

8.2.2.1. 本製品添付のサンプル・プロジェクトを流用する

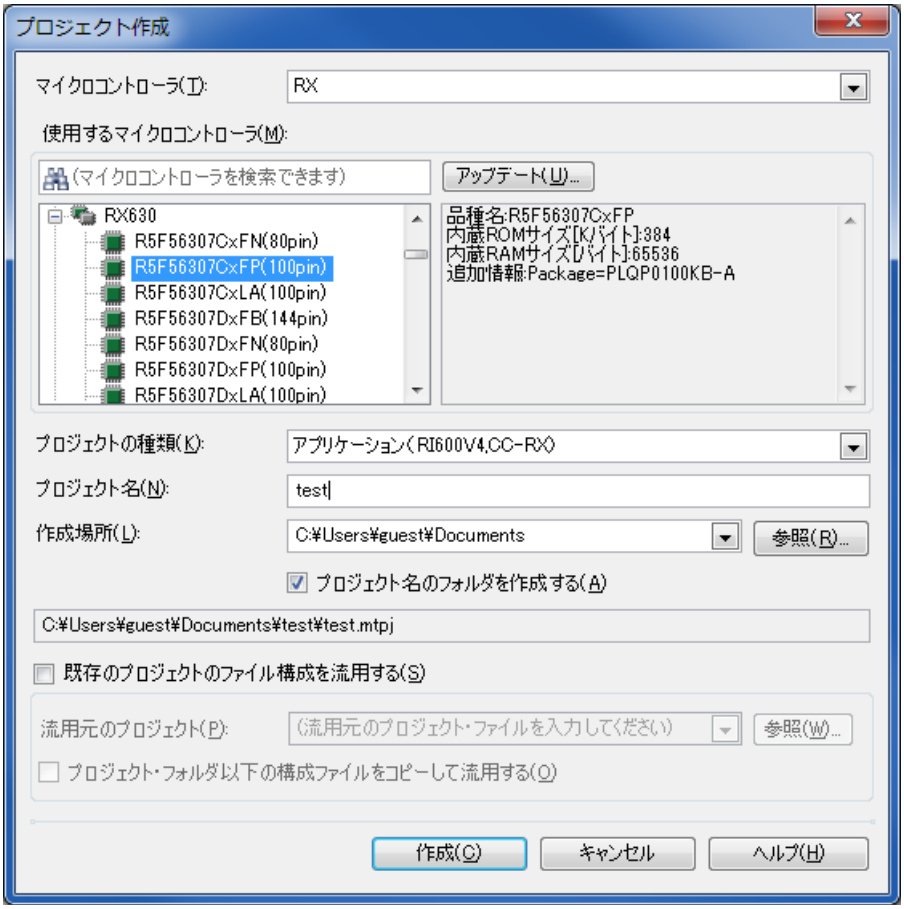
CS+のスタートパネルの［サンプル・プロジェクトを読み込む］エリアで［RX］タブを選択し、”RX???_RI600V4”という名称のプロジェクトを選択してください。

8.2.2.2. 新しいプロジェクトを作成する

（1） プロジェクトの作成

CS+のスタートパネルの[新しいプロジェクトを作成する]エリアの[GO]ボタンを押し、[プロジェクト作成]ダイアログをオープンします。

図 8-3 プロジェクト作成ダイアログ（プロジェクト新規作成）



- ［マイクロコントローラ］：”RX”を選択してください。
- ［プロジェクトの種類］：”アプリケーション（RI600V4, CC-RX）”を選択してください。

［作成］ボタンを押すと、プロジェクトが作成されます。

(2) ファイルの登録

プロジェクト作成直後は、何もファイルが登録されていません。「RI600V4 コーディング編」の「第 2 章 システム構築」を参考に、以下のようなファイルを登録してください。

- タスクやハンドラなどの処理プログラム・ファイル（「RI600V4 コーディング編」の 2.2 節を参照）
- システム・コンフィギュレーション・ファイル（「RI600V4 コーディング編」の 2.3 節を参照）
- ユーザ・オウン・コーディング部（「RI600V4 コーディング編」の 2.4 節を参照）

(3) ビルド・オプションの設定

「RI600V4 コーディング編」の「2.5 ロード・モジュールの生成」および「2.6 ビルド・オプション」を参考に、適切なビルド・オプションを設定してください。

8.2.3. サンプル・プログラム

提供するサンプル・プログラムは、リアルタイム OS タスク・アナライザを「ハードウェア・トレース・モードでトレース・チャートを取得」で使用する設定になっています。

「ソフトウェア・トレース・モードでトレース・チャートを取得」または「ソフトウェア・トレース・モードで長時間統計を取得」に設定を変更した場合は、システム・コンフィギュレーション・ファイルに以下の追記が必要です。詳細は、「RI600V4 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「15.3 ソフトウェア・トレース・モードのユーザ・オウン・コーディング部」を参照してください。なお、出荷時のシステム・コンフィギュレーション・ファイルは、これらの記述がコメントアウトされています。

(1) 「ソフトウェア・トレース・モードでトレース・チャートを取得」

```
interrupt_vector[29]{           // CMT CH1
    os_int          = NO;
    entry_address   = _RIUSR_trcSW_interrupt();         // in trcSW_cmt.src
};
```

(2) 「ソフトウェア・トレース・モードで長時間統計を取得」

```
interrupt_vector[29]{           // CMT CH1
    os_int          = NO;
    entry_address   = _RIUSR_trcLONG_interrupt();       // in trcLONG_cmt.src
};
```

8.2.4. リアルタイム OS リソース情報パネルに関する注意事項

8.2.4.1. 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 の初期化完了前は、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。

8.2.4.2. 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。

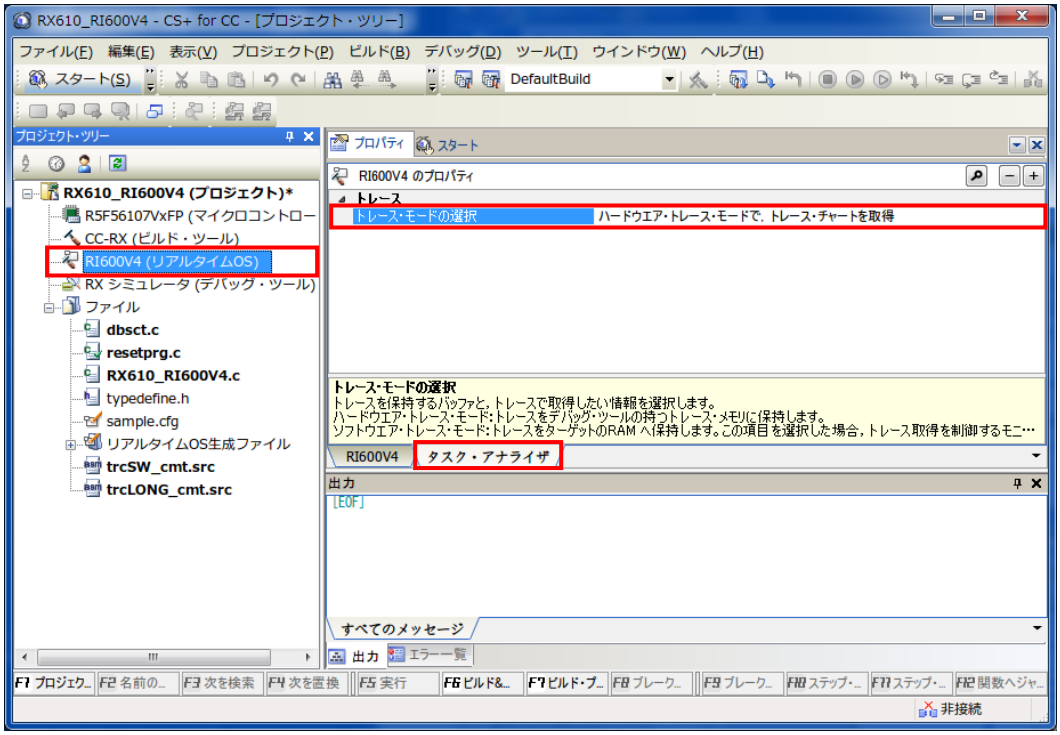
デバッグ情報を生成するには「ビルド・ツール」の「リンク・オプション」のプロパティで「デバッグ情報を出力する」を「はい」に設定してください。

8.2.5. リアルタイム OS タスク・アナライザに関する注意事項

8.2.5.1. トレース・モード変更

RI600V4 のプロパティの「タスク・アナライザ」タブで、「トレース・モードの選択」を選択します。また「トレース・モードの選択」を変更した場合は、必ずビルドを行ってください。トレース・モードごとに使用するモニタが違うため、ビルドを行うことで正しいモニタを組み込みます。

図 8-4 トレース・モードの選択

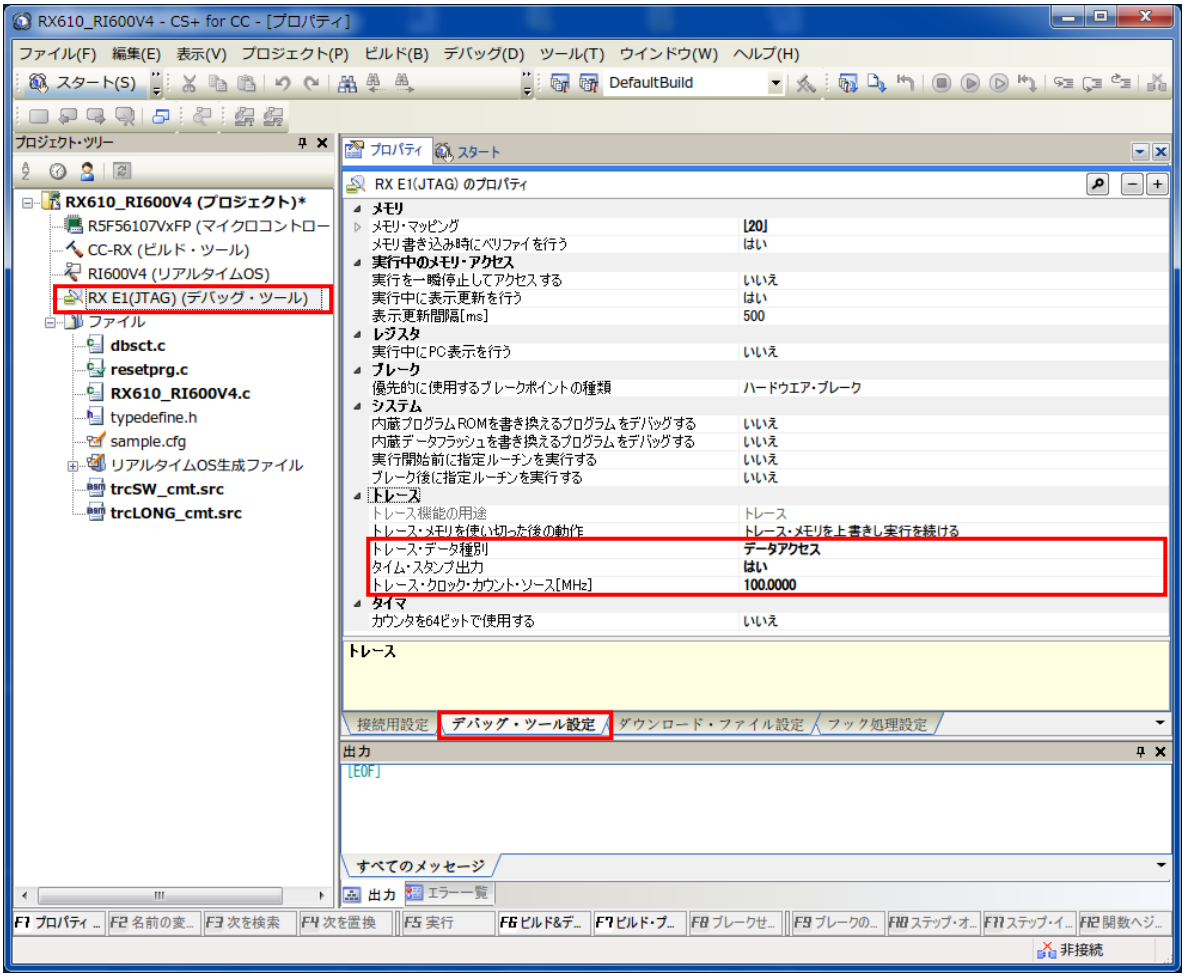


8.2.5.2. E1／E20 エミュレータを使用して「ハードウェア・トレース・モードでトレース・チャートを取得」する場合

デバッグ・ツールのプロパティで［デバッグ・ツール設定］タブの［トレース］カテゴリを以下のように設定してください。

- ［トレース・データ種別］：データアクセス
- ［タイム・スタンプ出力］：はい
- ［トレース・クロック・カウント・ソース［MHz］］：適切な値
  例：製品添付の RX610 用サンプル・プログラムでは「100.000」

図 8-5 E1／E20 エミュレータのトレース設定

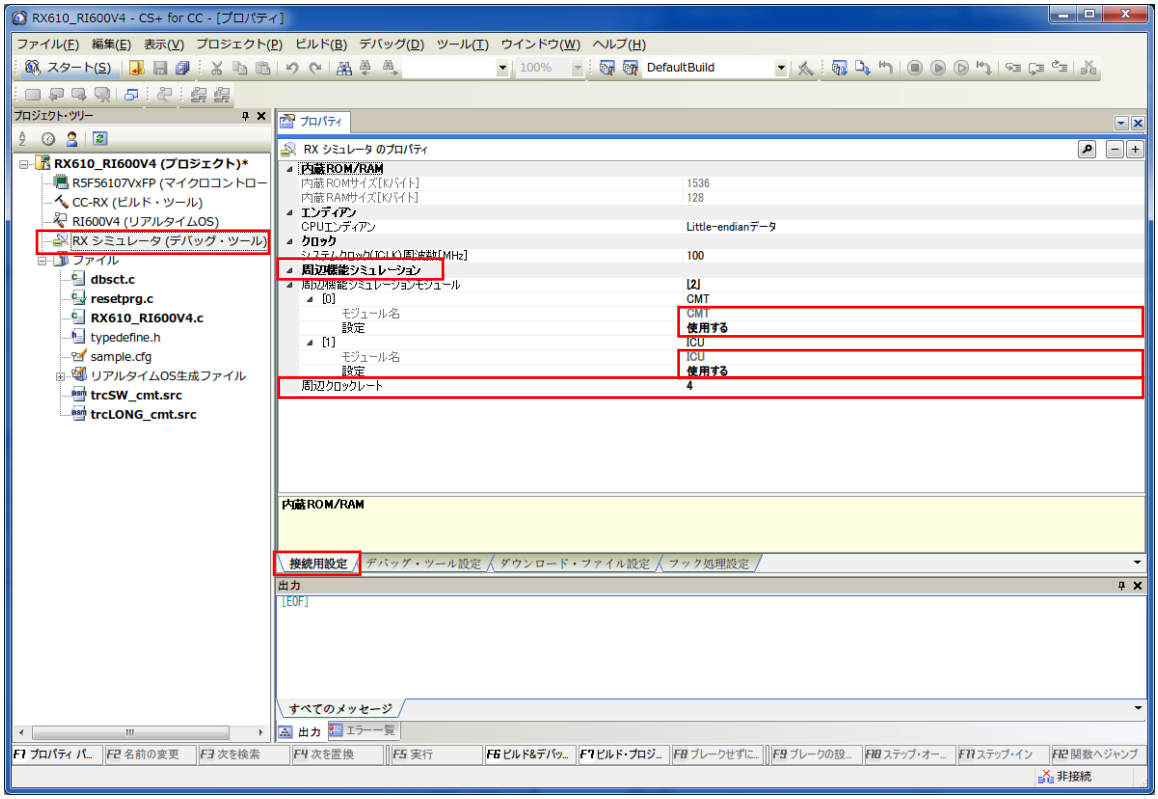


8.2.5.3. シミュレータを使用して「ハードウェア・トレース・モードで、トレース・チャートを取得」する場合

RX のシミュレータを使用してアプリケーションを動作させる場合、タスク・アナライザで表示する時間を正しくするために、以下の設定値を適切に設定する必要があります

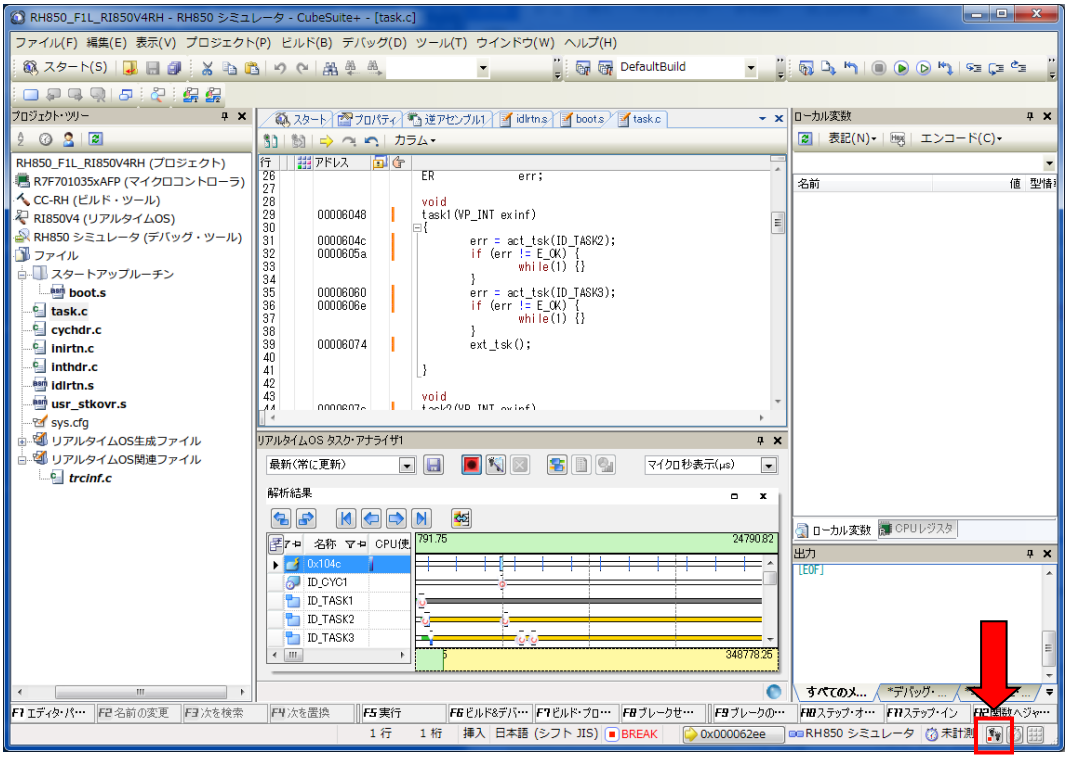
- システムクロック（ICLK）周波数［MHz］
  RX シミュレータ（デバッグ・ツール）プロパティの「接続用設定」タブ内の「クロック」カテゴリ（参照）を「適切な値」に設定する。
  例：製品添付の RX610 用サンプル・プログラムでは「100.000」
- 周辺機能シミュレーションモジュール「CMT」
  RX シミュレータ（デバッグ・ツール）プロパティの「接続用設定」タブ内の「周辺機能シミュレーション」カテゴリの「周辺機能シミュレーション」の「CMT」（参照）を「使用する」に設定する
- 周辺機能シミュレーションモジュール「ICU」
  RX シミュレータ（デバッグ・ツール）プロパティの「接続用設定」タブ内の「周辺機能シミュレーション」カテゴリの「周辺機能シミュレーション」の「ICU」（参照）を「使用する」に設定する
- 周辺機能シミュレーションの「周辺クロックレート」
  RX シミュレータ（デバッグ・ツール）プロパティの「接続用設定」タブ内の「周辺機能シミュレーション」カテゴリの「周辺クロックレート」（参照）を「適切な値」に設定する
  例：製品添付の RX610 用サンプル・プログラムでは「4」

図 8-6　RX シミュレータ（デバッグ・ツール）プロパティの「接続用設定」タブ



そして、デバッガのトレーススイッチを ON にしてください（足跡マークのアイコン）。

図 8-7　トレーススイッチ



8.2.5.4. デバッグ・ツールの設定

デバッグ・ツールのプロパティにおいて「デバッグ・ツール」タブ内の「トレース」カテゴリを以下の組み合わせの設定にしないでください。

- 実行前にトレース・メモリをクリアする：いいえ
- トレース・タイム・タグを積算する：はい

8.2.5.5. ソフトウェア・トレースのタイム・スタンプについて

ソフトウェア・トレースのタイム・スタンプは、カーネルのタイマ機能を使用して実現しています。カーネルのタイマは OS タイマ割り込みを使用して実現しているため、割り込み禁止状態の場合は、タイマ割り込み処理が保留されます。タスクなどで割り込み禁止にし、その期間が 1ms 以上であった場合は正しい時間を表示できません（処理順は正しく表示します）。

8.3. e^2 studio 使用時の注意事項

詳細は e^2 studio ヘルプを参照してください。

9. 制限事項

9.1. CS+と e^2 studio で共通の制限事項

9.1.1. リアルタイム OS リソース情報表示プラグイン

(1) **待ちタスク表示（子ノード表示）で表示リセットを選択すると、タスク・タブの表示がリセットされる制限**

待ちタスクのカラム情報をリセットすると、タスクのカラム情報もリセットします。ただし、表示情報の内容としては問題ありません。

(2) **タスク、周期ハンドラ、アラームハンドラにおける「残り時間」表示で、実際の表示値よりも 1 多い値が表示されることがある**

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］

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

9.1.2. リアルタイム OS タスク・アナライザ・プラグイン

(1) **CPU 使用率カラムに対してフィルタリング操作を行ったとき、正しい結果が得られないことがある制限**

フィルタリング内容によって、正しくフィルタリングされた結果が表示されないことがあります。例えば CPU 使用率を「80%以上の表示」を指定したとき、フィルタリングされずにすべてのカラムが表示されてしまいます。

9.2. CS+使用時の制限事項

9.2.1. リアルタイム OS ビルド設定プラグイン

下記に現状の制限事項を記載します。

(1) **ビルド・モード未対応の制限事項**

下記の制限により、複数のビルド・モードを使用しないでください。

- ビルド・モードごとにコンフィギュレータのオプションを保存しません。そのため、複数のビルド・モードを作成しても、すべてのビルド・モードで同じコンフィギュレータ・オプションで起動します。
- ビルド・モードを切り替えるたびに、ビルド・ツールの「追加のインクルード・パス」に kernel_id.h へのパスが追加されてしまいます。正しいパスはリアルタイム OS ビルド設定プラグインが「システム・インクルード・パス」に設定していますが、IDE が「追加のインクルード・パス」に、ビルド・

モードを切り替える前のパスを設定してしまい、ビルド時に IDE が設定したパスを先行して参照します。ビルド・モードを切り替えた後に kernel_id.h が変更されるようなコンフィギュレーション・ファイル編集を行った場合、その変更がビルドに反映されないことになります。

（2） **流用プロジェクト機能に関する制限**

流用元のプロジェクトに sit.s などのコンフィギュレータが生成するファイルが存在しない（クリーンされている状況）かつ、流用元のファイルを「コピーして流用プロジェクトを作成する」という操作が行われた場合、本来グレー表示でプロジェクト・ツリーに登録されている sit.s ファイルなどがプロジェクト・ツリーから削除されてしまいます。

（3） **High-performance Embedded Workshop プロジェクトの変換に関する制限**

High-performance Embedded Workshop の RI600/4 プロジェクトを CS+プロジェクトに変換したとき、以下の High-performance Embedded Workshop プロジェクトの設定が CS+プロジェクトに反映されません。

- RX Standard Toolchain の［RI600/4］タブの［コンフィギュレーション］カテゴリの［その他のオプション］のうち、「［-v］コマンドのオプションの説明と詳細なバージョンを表示する」を除くオプションの設定
- RX Standard Toolchain の［RI600/4］タブの［コンフィギュレーション］カテゴリの［ユーザ指定オプション］の設定
- RX Standard Toolchain の［RI600/4］タブの［テーブル生成］カテゴリのうち、［MRC ファイル検索フォルダ］の設定

プロジェクトへの変換後、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、必要な設定を行ってください。

9.3. e2 studio 使用時の制限事項

9.3.1. リアルタイム OS タスク・アナライザ・プラグイン

9.3.1.1. 対応デバッグ・ツールとトレース・モードの制限

制限により、デバッグ・ツールとトレース・モードの組み合わせで使用できないものがあります。

対応デバッグ・ツール	ハードウェア・トレース・モード	ソフトウェア・トレース・モード
E1	制限(1)	制限(3)
E20	制限(1)	制限(3)
シミュレータ	制限(2)	制限(3)

(1) **エミュレータ接続でハードウェア・トレース時に RTOS タスク・アナライザが利用できない制限**

デバッグ・ツールが E1、E20 の場合、RTOS アナライザの表示する情報は不正となり利用できません。

(2) **シミュレータ接続でハードウェア・トレース時に時間が取得できない制限**

デバッグ・ツールがシミュレータの場合、タイム・スタンプは取得できません。これはシミュレータのトレースへはタイム・スタンプを出力しないためです。このためトレース・チャートでイベント順を追うことはできますが、イベント間の時間は正しくありません（全て 1 です）。

(3) **ソフトウェア・トレース時で指定トレースバッファサイズが大きいとデータ取得できない場合がある制限**

大きいトレースバッファサイズを指定した場合、プログラム停止後のトレース取得時に 1 分程度処理が返らず、処理が返ってきてもデータが取得できない場合があります。この現象が起きた場合は、トレースバッファサイズを小さくすることで、この問題を回避することができます。

10. サンプル・プログラム

10.1. Firmware Integration Technology モジュールを組み込んだサンプル・プログラム

本章では、Firmware Integration Technology（以下 FIT と称す）モジュールを組み込んだサンプル・プログラムについて説明します。

10.1.1. 概 要

本サンプル・プログラムは、Renesas Starter Kits ボードで動作するように構成されています。本サンプル・プログラムをご利用頂くことにより、アプリケーションをより迅速かつ容易に開発することが可能となります。 本サンプル・プログラムは、既存の「RX64M_RI600V4」などに必要最小限の FIT モジュールを追加したものです。 ハードウェアの初期化に FIT モジュールを使用していることを除いて、基本的な動作は同じです。

10.1.2. FIT 対応サンプル・プログラムの構成

FIT に対応したサンプル・プログラムは、CS+のサンプル・プロジェクトとして提供します。
今回提供する FIT 対応 RI600V4 サンプル・プロジェクト、および組み込んだ FIT モジュールを以下に示します。

- FIT 対応 RI600V4 サンプル・プロジェクト

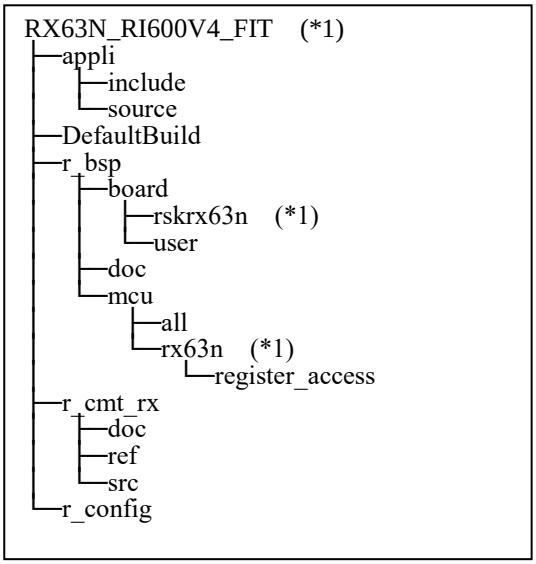
サンプル・プロジェクト名	Renesas Starter Kits ボード名
RX71M_RI600V4_FIT	Renesas Starter Kit+ for RX71M
RX65N_RI600V4_FIT	Renesas Starter Kit+ for RX65N
RX64M_RI600V4_FIT	Renesas Starter Kit+ for RX64M
RX63N_RI600V4_FIT	Renesas Starter Kit+ for RX63N
RX210_RI600V4_FIT	Renesas Starter Kit for RX210 (B 版)
RX113_RI600V4_FIT	Renesas Starter Kit for RX113

- 使用した FIT モジュール

FIT モジュール名	FIT モジュール名	リビジョン
ボードサポートパッケージ（BSP）	r_bsp	Dec.01.15 Rev.3.10 Oct.01.16 Rev.3.40（RX65N 用）
コンペアマッチタイマ（CMT）	r_cmt_rx	Jun.30.15 Rev.2.60 Oct.01.16 Rev.3.00（RX65N 用）

10.1.3. FIT 対応 RI600V4 サンプル・プロジェクトのディレクトリ構成

RX63N 用 FIT 対応 RI600V4 サンプル・プロジェクトのディレクトリ構成を以下に示します。



*1 その他のサンプル・プロジェクトでは、各々対応した MCU 名と RSK ボード名に置き換わります。

FIT モジュール（r_bsp と r_cmt_rx）はプロジェクトのルートディレクトリに配置しています。
FIT モジュールの設定変更用ヘッダファイルは r_config フォルダにまとめて格納します。

10.1.4. FIT 対応 RI600V4 サンプル・プロジェクトの変更点

FIT 対応 RI600V4 サンプル・プロジェクトでは、FIT モジュール及び基にしたサンプル・プログラムを一部変更しています。 FIT モジュールは基本的に RTOS あり／なしのプロジェクトで使えるように変更しています。

本章では主な変更点について説明します。

（1） FIT モジュールによる RSK ボードの初期化

【対象ファイル】

r_bsp¥board¥<RSK ボード名>¥resetprg.c

r_bsp¥board¥<RSK ボード名>¥dbsct.c

※RSK ボード名 ：rskrx113, rskrx210, rskrx63n, rskrx64m, rskrx65n, rskrx71m

【変更内容】

FIT モジュールのボードサポートパッケージ（r_bsp）を使って RSK ボードを初期化します。

そのため元のサンプル・プログラムから重複する以下のファイルを削除しました。

appli¥source¥reset¥resetprg.c

appli¥source¥reset¥dbsct.c

スタートアップ・ルーチン（resetprg.c の PowerON_Reset_PC 関数）では BSP_CFG_RTOS_USED マクロを使って、OS 無し又は RI600V4 の場合で処理を切り替えるようにしています。

割り込みベクタ及び固定／例外ベクタの先頭アドレスは OS 無し又は RI600V4 の場合で異なります。

OS 無しの場合にはユーザーモードで main 関数を呼び出し、RI600V4 の場合にはスーパーバイザ・モードで vsta_knl を呼び出します。

```
void PowerON_Reset_PC(void)
{
#if BSP_CFG_RTOS_USED == 0       /* Non-OS */
      set_intb((void *)__sectop("C$VECT"));
#ifdef __RXV2
      set_extb((void *)__sectop("EXCEPTVECT"));/* RXv2 command */
#endif/* __RXV2 */
#elif BSP_CFG_RTOS_USED == 1    /* FreeRTOS */
#elif BSP_CFG_RTOS_USED == 2    /* SEGGER embOS */
#elif BSP_CFG_RTOS_USED == 3    /* Micrium MicroC/OS */
#elif BSP_CFG_RTOS_USED == 4    /* Renesas RI600V4 & RI600PX */
      set_intb((void *)__sectop("INTERRUPT_VECTOR"));
#ifdef __RXV2
      set_extb((void *)__sectop("FIX_INTERRUPT_VECTOR"));/* RXv2 command */
#endif/* __RXV2 */
#endif/* BSP_CFG_RTOS_USED */

      （省略）

#if BSP_CFG_RTOS_USED == 0       /* Non-OS */
      nop();
      set_psw(PSW_init);
#if BSP_CFG_RUN_IN_USER_MODE==1
      chg_pmusr() ;
#endif
      main();
#if BSP_CFG_IO_LIB_ENABLE == 1
      _CLOSEALL();
#endif
      while(1)
      {
            /* Infinite loop. Put a breakpoint here if you want to catch an exit of main(). */
      }
#elif BSP_CFG_RTOS_USED == 1    /* FreeRTOS */
#elif BSP_CFG_RTOS_USED == 2    /* SEGGER embOS */
#elif BSP_CFG_RTOS_USED == 3    /* Micrium MicroC/OS */
#elif BSP_CFG_RTOS_USED == 4    /* Renesas RI600V4 */
      /* Lock a timer resource by r_bsp, if using time function on RTOS. */
      if(R_BSP_HardwareLock((mcu_lock_t)(BSP_LOCK_CMT0 + _RI_CLOCK_TIMER)) == false){
            while(1);
      }
      /* Initialize CMT for RI600V4 */
      _RI_init_cmt();
      /* Make sure to disable interrupt. */
      clrpsw_i();

      vsta_knl();          /* Start RI600V4 and never return */
      brk();
#endif/* BSP_CFG_RTOS_USED */
}
```

RX シミュレータによるデバッグの際、無限待ちを防止するため、USE_SIM_DEBUG マクロでクロック初期化ルーチンをスキップできるようにしています。

詳細は「10.1.5 FIT 対応 RI600V4 サンプル・プロジェクトの注意点」の「（7）RX シミュレータによるデバッグ」を参照してください。

dbsct.c では BSP_CFG_RTOS_USED マクロを使うことにより、初期化セクション設定を OS 無し又は RI600V4 の場合で切り替えています。

（2） 割り込みベクタの集約

【対象ファイル】

r_bsp¥board¥<RSK ボード名>¥vecttbl.c

r_bsp¥mcu¥<MCU 名>¥mcu_interrupts.c

r_cmt_rx¥src¥r_cmt_rx.c

appli¥source¥kernel¥sample.cfg

※RSK ボード名：rskrx113, rskrx210, rskrx63n, rskrx64m, rskrx65n rskrx71m

※MCU 名： rx64m, rx65n, rx71m

【変更内容】

FIT モジュールで定義されている割り込みベクタを RTOS 側のシステム・コンフィギュレーション・ファイル（sample.cfg）へ集約します。

vecttbl.c では BSP_CFG_RTOS_USED マクロを使って、以下の記述を除外しています。

・割り込みハンドラ関数の#pragma interrupt

・割り込みベクタテーブルの定義（#pragma section C FIXEDVECT 以降）

```
#if BSP_CFG_RTOS_USED == 0       /* Non-OS */
#pragma interrupt (non_maskable_isr)
#elif BSP_CFG_RTOS_USED == 1    /* FreeRTOS */
#elif BSP_CFG_RTOS_USED == 2    /* SEGGER embOS */
#elif BSP_CFG_RTOS_USED == 3    /* Micrium MicroC/OS */
#elif BSP_CFG_RTOS_USED == 4    /* Renesas RI600V4 & RI600PX */
#endif
void non_maskable_isr(void)
{
        （省略）
}

#if BSP_CFG_RTOS_USED == 0       /* Non-OS */
#pragma section C FIXEDVECT

void * const Fixed_Vectors[] =
{
      （省略）
      (void *) non_maskable_isr,         /* 0xffffffff8    NMI */
      (void *) PowerON_Reset_PC      /* 0xfffffffc    RESET */
};
#elif BSP_CFG_RTOS_USED == 1    /* FreeRTOS */
#elif BSP_CFG_RTOS_USED == 2    /* SEGGER embOS */
#elif BSP_CFG_RTOS_USED == 3    /* Micrium MicroC/OS */
#elif BSP_CFG_RTOS_USED == 4    /* Renesas RI600V4 & RI600PX */
#endif
```

RX64M, RX65N, RX71M では mcu_interrupts.c の以下のグループ割り込みハンドラから#pragma interrupt を除外しています。

group_al0_handler_isr

group_al1_handler_isr

group_bl0_handler_isr

group_bl1_handler_isr

group_bl2_handler_isr （RX65N のみ）

上記の割り込みハンドラ関数はすべて sample.cfg ファイルに登録します。

```
// BSP Interrupt Handler Definition (VECT_ICU_GROUPBL0)
interrupt_vector[110]{
        os_int                  = YES;
        entry_address       = group_bl0_handler_isr();
        pragma_switch       = E,ACC;
};
```

r_cmt_rx.c では、以下の FIT タイマ API 用の割り込みハンドラから#pragma interrupt 行と static 宣言を除外しています。

cmt0_isr

cmt1_isr

cmt2_isr

cmt3_isr

上記の割り込みハンドラ関数は sample.cfg ファイルで以下のように登録しています。

【RX71M, RX65N, RX64M の場合】

割り込みベクタ 128：cmt2_isr

割り込みベクタ 129：cmt3_isr

【RX63N, RX210, RX113 の場合】

割り込みベクタ 30：cmt2_isr

割り込みベクタ 31：cmt3_isr

（3） RTOS 用ヘッダファイルを FIT 側でインクルード

【対象ファイル】

r_bsp¥board¥<RSK ボード名>¥r_bsp.h

※RSK ボード名：rskrx113, rskrx210, rskrx63n, rskrx64m, rskrx65n, rskrx71m

【変更内容】

以下のヘッダファイルを r_bsp.h でインクルードします。

kernel.h

kernel_id.h

```
#if BSP_CFG_RTOS_USED == 0       /* Non-OS */
#elif BSP_CFG_RTOS_USED == 1    /* FreeRTOS */
#elif BSP_CFG_RTOS_USED == 2    /* SEGGER embOS */
#elif BSP_CFG_RTOS_USED == 3    /* Micrium MicroC/OS */
#elif BSP_CFG_RTOS_USED == 4    /* Renesas RI600V4 & RI600PX */
#include       "kernel.h"
#include       "kernel_id.h"
#endif/* BSP_CFG_RTOS_USED */
```

r_bsp.h は platform.h 内でインクルードされているので、RTOS のソースでは platform.h のみをインクルードします。

（4） r_cmt_rx モジュールで RI600V4 が使用するタイマーリソースを除外

【対象ファイル】

r_cmt_rx¥src¥r_cmt_rx.c

r_config¥r_cmt_rx_config.h

【変更内容】

本モジュールのタイマ API では、CMT チャンネルが_RI_CLOCK_TIMER または_RI_TRACE_TIMER の場合にエラーを返します。

※_RI_TRACE_TIMER マクロは r_cmt_rx_config.h で定義しています。

```
bool R_CMT_Stop (uint32_t channel)
{
    /* Make sure valid channel number was input. */
    if (channel >= CMT_RX_NUM_CHANNELS)
    {
        /* Invalid channel number was used. */
        return false;
    }

#if BSP_CFG_RTOS_USED == 0      /* Non-OS */
#elif BSP_CFG_RTOS_USED == 1    /* FreeRTOS */
#elif BSP_CFG_RTOS_USED == 2    /* SEGGER embOS */
#elif BSP_CFG_RTOS_USED == 3    /* Micrium MicroC/OS */
#elif BSP_CFG_RTOS_USED == 4    /* Renesas RI600V4 & RI600PX */
    /* Exclude RTOS timers */
    if (channel == _RI_CLOCK_TIMER || channel == _RI_TRACE_TIMER)
    {
        return false;
    }
#endif/* BSP_CFG_RTOS_USED */

    /* Stop counter. */
    cmt_counter_stop(channel);
```

（5） r_cmt_rx モジュールで RTOS 予約チャンネルの初期状態を設定

【対象ファイル】

r_cmt_rx¥src¥r_cmt_rx.c

【変更内容】

CMT チャンネルの利用状況は g_cmt_modes 配列に格納されますが、そこに RTOS 側で使用する CMT チャンネルの初期値（CMT_RX_MODE_PERIODIC）を設定しています。

※ペアとなる CMT チャンネルのパワーダウン防止のため。

（6） RSK ボード上の LED 制御

【対象ファイル】

appli¥include¥hw_control.h

appli¥source¥common¥hw_control.c

【変更内容】

RSK ボード上の LED の点灯／消灯を制御する set_LED 関数を作成しました。

（7） デバッグ用メッセージ出力

【対象ファイル】

appli¥include¥rtos_sample_config.h

【変更内容】

本サンプルでは RX シミュレータ／E1 エミュレータでデバッグ中に、printf 関数を使ってデバッグ・コンソールへ任意のメッセージを出力できます。 ただし、サンプル・プログラムでは printf 関数は直接呼び出さずに、rtos_sample_config.h で定義した DEBUG_print マクロを使用します。
DEBUG_print マクロは同じヘッダファイルで定義した以下のマクロにより有効／無効を制御できます。

USE_DEBUG_MESSAGE （定義ありの場合、デバッグ・コンソールにメッセージを出力）

（8） サンプル・プログラムの変更

【対象ファイル】

appli¥source¥task.c
appli¥source¥sysdwn.c

【変更内容】

・すべての対象 C ソースで kernel.h と kernel_id.h の代わりに platform.h をインクルードします。
・タスク内に DEBUG_print マクロを使ったメッセージ出力を追加。
・タスク task1 と task2 にタイミング調整用の dly_tsk を追加。
・タスク task1（task.c）の書き換え権限取得時に LED2 を点灯、LED3 を消灯。
・タスク task2（task.c）の書き換え権限取得時に LED2 を消灯、LED3 を点灯。
・周期ハンドラ cyh1（handler.c）の呼び出し回数を分周して LED1 の点灯／消灯を切り替え。
分周には rtos_sample_config.h で定義した LED_BLINK_DIV_RATIO マクロを使用。
・システム・ダウン・ルーチン_RI_sys_dwn__（sysdwn.c）でエラーメッセージを
デバッグ・コンソールに出力します。
※Trial 版で 1 時間の使用制限に達した場合のエラーメッセージを追加していますが、
製品版及び評価版では不要なメッセージになります。
・PowerON_Reset_PC 関数（resetprg.c）で r_bsp の API を使い CMT チャンネルをロックします。

（9） 個別コンパイル・オプションの設定

【対象ファイル】

r_bsp¥board¥<RSK ボード名> ¥resetprg.c
※RSK ボード名：rskrx113, rskrx210, rskrx63n, rskrx64m, rskrx65n rskrx71m

【変更内容】

スタック領域を 4 バイト境界に配置するため、個別コンパイル・オプションを設定しています。
設定内容は、[その他]-[その他の追加オプション] に”-nostuff”を追加しています。

（10） CC-RX（ビルド・ツール）のオプション変更

【コンパイル・オプション】

・[ソース]-[C ソース・ファイルの言語]を C89 から C99 に変更しています。
・[最適化]-[最適化レベル]を 2 から 0 に変更しています。

【リンク・オプション】

・[リスト]-[シンボル情報を出力する] を「はい」に変更しています。

【ライブラリ・ジェネレート・オプション】

・[標準ライブラリ]-[ライブラリ構成]を C89 から C99 に変更しています。

・[標準ライブラリ]-[構築対象のライブラリ]を「カスタム」に変更しています。

10.1.5. FIT 対応 RI600V4 サンプル・プロジェクトの注意点

本章では FIT 対応 RI600V4 サンプル・プロジェクトご使用時の主な注意点について説明します。

（1） CMT チャンネルの制約

RI600V4 ではデフォルトで CMT0 をシステム時刻の更新に使用しています。
また、ソフトウェア・トレース・モード使用時に CMT1（固定）を使用します。
その他 CMT2 以降のチャンネルは FIT のタイマ API（R_CMT_CreateOneShot 等）で動的に利用されます。

（2） トレース・モードの変更

本サンプルではタスク・アナライザをすぐ使えるようにソフトウェア・トレース・モードがデフォルトで有効になっています。
トレース・モードを変更する場合は以下の設定を変更してください。

1.トレース・モードの選択

「RI600V4（リアルタイム OS）」の「タスク・アナライザ」タブの中の
以下でトレース・モードを選択します。
[トレース]-[トレース・モードの選択]

2. 割り込みハンドラの変更

sample.cfg でトレース･モードに応じた割り込みハンドラをベクタ 29 番（CMT1）に登録します。

【ハードウェア・トレース・モードでトレース・チャートを取得する場合】

```
entry_address = cmt1_isr();
```

※FIT のタイマ割り込みハンドラ（ダミー）を登録します。

【ソフトウェア・トレース・モードでトレース・チャートを取得する場合】

```
entry_address = _RIUSR_trcSW_interrupt ();
```

【ソフトウェア・トレース・モードで長時間統計を取得する場合】

```
entry_address = _RIUSR_trcLONG_interrupt();
```

（3） ハードウェア・トレース・モードの制約

RX113 には「タイム・スタンプ出力」機能がないため、ハードウェア・トレース・モードでタスク・アナライザを利用できません。
RX71M はハードウェア・トレース・モードのタスク・アナライザに未対応です。
次版以降で対応予定です。

（4） タイマーリソース CMT1 の再利用

ソフトウェア・トレース・モードを使用しない場合、CMT1 は使われません。
CMT1 を FIT のタイマ API で再利用する場合は以下を設定してください。

１．(２)のハードウェア・トレース・モードの設定をする。
２．以下のファイルで定義されている_RI_TRACE_TIMER マクロに_RI_CLOCK_TIMER マクロと同じ値を設定します。

r_config¥ r_cmt_rx_config.h

（5） エミュレータからの電源供給

本サンプルではデフォルトでエミュレータ（USB）から電源供給する設定にしていますが、

開発の段階で消費電流が増加して USB 電源では供給能力が不足する可能性があるため、RX64M 以降の RSK ボードでは外部電源を使用して開発を行ってください。
「RX E1(JTAG)（デバッグ・ツール）」のプロパティの「接続用設定」タブで以下を変更してください。
[ターゲット・ボードとの接続]-[エミュレータから電源供給をする（最大 200mA）]

（6） FIT モジュールの更新

本サンプルに付属する FIT モジュール（r_bsp と r_cmt_rx）は、RTOS 用にカスタマイズされています。そのため、それらを該当 FIT モジュールの最新バージョンで上書きしないでください。

（7） RX シミュレータによるデバッグ

RX71M, RX65N, RX64M, RX113 のクロック初期化ルーチンはエミュレータを前提としているため、RX シミュレータでデバッグするとレジスタの読み出しで無限待ちループに入ります。
この問題を回避するため、コンパイル・オプションの[ソース]-[マクロ定義]で USE_SIM_DEBUG マクロを定義するか、resetprg.c の中の以下のコメントを外してビルドしてください。

```
//#define USE_SIM_DEBUG
```

マクロの定義によりクロック初期化ルーチンがスキップされます。

（8） デバッグ時のメッセージ出力機能

DEBUG_print マクロ関数によりデバッグ実行中のログやエラーメッセージの出力が可能になります。
ただし、本サンプルでは DEBUG_print はデフォルトで無効にしています。
CS+のデバッグ・コンソール・プラグインがデフォルトで無効になっているためです。
以下の手順で DEBUG_print を有効に出来ます。

１．デバッグ・コンソール・プラグインの有効化
CS+のメニューから[ツール]-[プラグインの管理]で「プラグインの管理」ダイアログを開き、「追加機能」タブの中の「デバッグ・コンソール・プラグイン」にチェックを入れます。

２．USE_DEBUG_MESSAGE マクロを定義してビルド
コンパイル時に USE_DEBUG_MESSAGE マクロを定義するか、rtos_sample_config.h の中の以下のコメントを外してビルドしてください。

```
//#define USE_DEBUG_MESSAGE
```

（9） タスク・スタック・サイズ

本サンプルは printf 標準関数の利用を前提としています。
printf 標準関数はスタックを多く消費します。（400 バイト以上）
そのため、タスク・スタック・サイズは想定よりも多く確保してください。

（10） R_CMT_Control の制限

FIT タイマ API の R_CMT_Control 関数で、CMT チャンネルに_RI_CLOCK_TIMER マクロ（0） 又は _RI_TRACE_TIMER マクロ（1）と同じ値を指定した場合は、エラーを返します。
RTOS 予約チャンネルを内部処理で除外しているためです。

（11） FIT API の制約

FIT モジュールでは様々な API を提供していますが、ご使用に際して以下の制約があります。

表 10-1 RI600V4 での r_bsp API の使用可否

r_bsp API 名	カーネル開始前	カーネル開始後	
	スタートアップ・ルーチン（PowerON_Reset_PC）	タスク（ユーザ・モード）	非タスク（スーパーバイザ・モード）
R_BSP_GetVersion	✓	✓	✓
R_BSP_InterruptsDisable	✓	(効果なし)	✓
R_BSP_InterruptsEnable	✓	(効果なし)	✓
R_BSP_CpuInterruptLevelRead	✓	✓	✓
R_BSP_CpuInterruptLevelWrite	✓	(効果なし) ***1**	(非推奨) ***1**
R_BSP_RegisterProtectEnable	✓	✓	✓
R_BSP_RegisterProtectDisable	✓	✓	✓
R_BSP_SoftwareLock	✓	✓	✓
R_BSP_SoftwareUnlock	✓	✓	✓
R_BSP_HardwareLock	✓	✓	✓
R_BSP_HardwareUnlock	✓	✓	✓
R_BSP_InterruptWrite	✓	✓	✓
R_BSP_InterruptRead	✓	✓	✓
R_BSP_InterruptControl	✓	✓	✓
R_BSP_SoftwareDelay	✓	✓	✓

*1 RI600V4 のサービス・コール chg_ims 又は ichg_ims をご使用ください。

表 10-2 RI600V4 での r_cmt_rx API の使用可否

r_cmt_rx API 名	カーネル開始前	カーネル開始後	
	スタートアップ・ルーチン（PowerON_Reset_PC）	タスク（ユーザ・モード）	非タスク（スーパーバイザ・モード）
R_CMT_CreatePeriodic	✓	✓ *1	✓ *1
R_CMT_CreateOneShot	✓	✓ *1	✓ *1
R_CMT_Control	✓	✓	✓
R_CMT_Stop	✓	✓	✓
R_CMT_GetVersion	✓	✓	✓

*1 RI600V4 の周期ハンドラ／アラームハンドラのご使用を推奨します。

（12） 未定義割り込みについて

本サンプルで未定義割り込みが発生した場合、カーネル内部のハンドラからシステム・ダウン・ルーチン（_RI_sys_dwn__）が呼ばれます。

r_bsp モジュールの未定義割り込みハンドラ（undefined_interrupt_source_isr）は呼び出されません。そのため、未定義割り込みのコールバック関数を登録しても、そのコールバック関数が呼び出されることはありません。

未定義割り込み発生時の処理はシステム・ダウン・ルーチンに記述してください。

（13） 基本クロック周期の設定について

RI600V4 では基本クロック用タイマ割り込みの周期を 1 ミリ秒に設定することを推奨しています。
FIT 対応 RI600V4 サンプル・プロジェクトでは、以下のファイルで設定した PCLKB のクロック周波数を、sample.cfg の clock.timer_clock に設定してください。

r_config¥r_bsp_config.h

これによりシステム時刻の単位が 1 ミリ秒になります。

```
// RX64M の場合の sample.cfg 設定例
clock{
    template        = rx630.tpl;
    timer           = CMT0;
    timer_clock     = 60MHz; // PCLKB の周波数
    IPL             = 13;
};
```

確認の方法は、周期ハンドラの起動周期を 125 に設定して、以下のファイルで
LED_BLINK_DIV_RATIO マクロに 2 を設定します。

appli¥include¥rtos_sample_config.h

これにより、約 1 秒間隔で LED1 が点滅します。

10.1.6. FIT モジュールの追加方法

FIT 対応 RI600V4 サンプルに新たな FIT モジュールを追加する手順について説明します。

（1） FIT モジュールのソースを格納した ZIP ファイルを入手します。

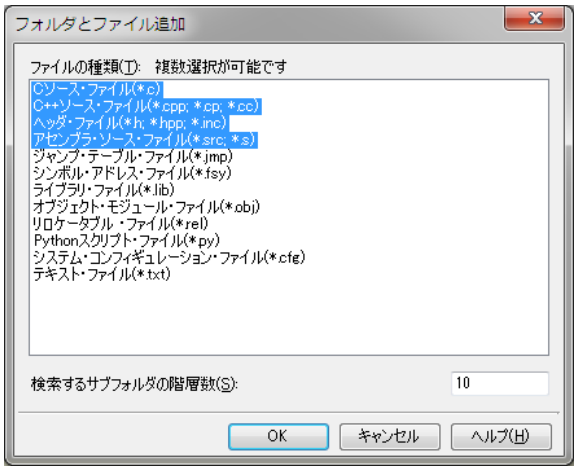
（2） CS+プロジェクトのルートディレクトリに FIT モジュールのソースを解凍します。

（3） r_config フォルダに FIT モジュールのコンフィギュレーション・ファイルを作成します。
通常リファレンス・ファイルがあるので、それをコピーしてファイル名を変更します。
コピー元： ref¥<FIT モジュール名>_config_reference.h
コピー先： r_config¥< FIT モジュール名>_config.h

（4） Windows エクスプローラで解凍した FIT モジュールのトップディレクトリを
CS+のプロジェクト・ツリーの「ファイル」へドラッグ＆ドロップします。

（5） 「フォルダとファイル追加」ダイアログで以下を設定して「OK」ボタンを押します。
・プロジェクトへ追加するファイルの種類（*.c, *.h など）を選択します。
・「検索するサブフォルダの階層数」を最大の階層数以上（例：10）に変更します。

図 10-1 「フォルダとファイル追加」ダイアログ



（6） 「CC-RX（ビルド・ツール）」の「コンパイル・オプション」タブで、
[ソース]-[追加のインクルード・パス]を確認します。

（7） CS+では上記の方法により、追加したディレクトリの全相対パスが「追加のインクルード・パス」へ登録されます。

（8） FIT モジュール内の割り込みハンドラをシステム・コンフィギュレーション・ファイルで登録します。
以下の手順でソースを変更してください。
１．FIT モジュール側のソースで”#pragma interrupt” の行をコメントアウトします。
※割り込みハンドラのソースでは platform.h 又は、kernel.h 及び kernel_id.h をインクルードする必要があります。確認してください。

２．割り込みハンドラ関数の static 宣言を削除します。

３．sample.cfg で割り込みハンドラを登録します。

以 上

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2018.10.1	—	新規発行

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/contact/

すべての商標および登録商標は，それぞれの所有者に帰属します。

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

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

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

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

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

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

(Rev.4.0-1　2017.11)

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 エレクトロニクス株式会社　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

■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：https://www.renesas.com/contact/

© 2018 Renesas Electronics Corporation. All rights reserved.
Colophon 6.0