

ユーザーガイド: e<sup>2</sup> studio 編

# RH850 スマート・コンフィグレータ

R20AN0739JC0100 Rev.1.00 Jan.20.2024

# 要旨

本アプリケーションノートでは、e<sup>2</sup> studio のプラグインツールである RH850 スマート・コンフィグレータ(以下、スマート・コンフィグレータと略す)の基本的な使用方法について説明します。

統合開発環境 e<sup>2</sup> studio の対象バージョンは以下の通りです。

• e<sup>2</sup> studio 2024-01 以降

# 対象デバイス/対応コンパイラ

サポートしているデバイス及びコンパイラは、以下の URL をご参照ください。

https://www.renesas.com/rh850-smart-configurator

# 目次

1.1 日的
1.1 日的
1.2 特長
2. プロジェクトの作成
3. スマート・コンフィグレータの操作方法1
3.1 スマート・コンフィグレータの表示1
3.2 操作手順1
3.3 プロジェクト情報の保存先1
3.4 ウィンドウ1
3.4.1 プロジェクト・エクスプローラー1
3.4.2 スマート・コンフィグレータビュー1
3.4.3 MCU/MPU パッケージビュー1
3.4.4 コンソールビュー
3.4.5 コンフィグレーションチェックビュー1
4. 周辺機能の設定1
4.1 ボード設定1
4.1.1 デバイス選択1
4.1.2 ボード選択
4.1.3 ボード設定のエクスポート1
4.1.4 ボード設定のインポート2
4.2 クロック設定2
4.3 システム設定(RH850/U2A のみ)2



4.4	コンポーネント設定	26
4.	4.1 コンポーネントー覧表示とハードウェアー覧表示の切り替え	26
4.	4.2 コード生成コンポーネントの追加方法	27
4.	4.3 ソフトウェアコンポーネントの削除	29
4.	4.4 ソフトウェアコンポーネントの設定	30
4.	4.5 ソフトウェアコンポーネントのリソース変更	31
4.	4.6 コンポーネントの設定のエクスポート	34
4.	4.7 コンポーネントの設定のインポート	35
4.	4.8 コンポーネントの基本設定	35
4.5	端子設定	37
4.	5.1 ソフトウェアコンポーネントの端子配置変更	38
4.	5.2 MCU/MPU パッケージの端子割り当て	39
4.	5.3 端子機能から端子番号の表示	40
4.	5.4 端子設定のエクスポート	41
4.	5.5 端子設定のインポート	41
4.	5.6 ボード端子設定情報を使用した端子設定	42
4.	5.7 端子のフィルタ機能	42
4.	5.8 端子エラー/警告設定	43
4.6	割り込み設定	44
4.	6.1 割り込み優先レベルと OS 管理の設定	44
4.	6.2 PE <i>n</i> の設定の変更(RH850/U2Aのみ)	45
4.	6.3 割り込み設定の変更	46
4.7	MCU マイグレーション機能	47
5. 5	競合の管理	.51
5.1	リソースの競合	51
5.2	端子の競合	52
•		50
6.		.53
6.1	生成 ソースの出力	53
6.2		54
6.3	生成ファイルの構成とファイル名	56
6.4	クロック設定	59
6.5	「「「」」で、「」」で、「」」で、「」」で、「」」で、「」」で、「」」で、「」	60
6.6	割り込み設定	61
7 .	ューザープログラムの作成	62
7		.02 62
7.1	コード主成の場合のガステムコード追加力法	02
1.2	ユーリーアフリリーションコードの使用	03
8 4	生成ソースのバックアップ	63
J		
9.	レポートの生成	64
9.1		64
9.2	ーニーー 端子機能リスト、端子番号リスト設定内容(csv 形式)	65
9.3	MCU/MPU パッケージ図(png 形式)	65



10. ユーザーコード保護機能	66
10.1 ユーザーコード保護機能の指定タグ	66
10.2 ユーザーコード保護機能の使用例	66
10.3 競合発生時の対応方法	67
10.3.1 競合の発生条件	67
10.3.2 競合の解決方法	68
	70
11. ヘルノ	70
11.1 ヘルフ	
12. 参考ドキュメント	71
ホームページとサポート容口	72
1	6
1. 1%安	6
1.2 特長	6
2. プロジェクトの作成	7
3. スマート・コンフィグレータの操作方法	11
3.1 スマート・コンフィグレータの表示	11
3.2 操作手順	12
3.3 プロジェクト情報の保存先	
3.4.3 MCU/MPU ハッケーシビュー	
$3.4.4  \exists \forall \forall \neg \nu \forall \neg \nu \forall = 1$	
3.4.5 コンフィクレーションチェックヒュー	16
4. 周辺機能の設定	17
4.1 ボード設定	17
4.1.1 デバイス選択	17
4.1.2 ボード選択	17
4.1.3 ボード設定のエクスポート	
4.1.4 ボード設定のインポート	20
4.2 クロック設定	21
4.3 システム設定(RH850/U2A のみ)	22
4.4 コンポーネント設定	
4.4.1 コンポーネントー覧表示とハードウェアー覧表示の切り替え	
4.4.2 コード生成コンポーネントの追加方法	27
4.4.3 ソフトウェアコンポーネントの削除	29
4.4.4 ソフトウェアコンポーネントの設定	
4.4.5 ソフトウェアコンポーネントのリソース変更	
4.4.6 コンポーネントの設定のエクスポート	
4.4.7 コンポーネントの設定のインポート	35



4.4.8 コンポーネントの基本設定	
4.5 「「」 おうした、フーンピーナントの地子町開本市	
4.5.1 ソノトワェアコンホーイントの「小士配直変更	
4.5.2 MCU/MPU ハッケーンの姉士割り当 (	
4.5.3 姉子機能から姉子奋亏の表示	
4.5.4 靖士設定のエクスホート	
4.5.5 「新士設定の1 ノホート	
4.5.0 小一下姉士設定情報を使用した姉士設定	
4.5.7 「「「サリノイルダ」 (版化	
4.5.8 姉ナエフー/ 言古設定	
4.6 刮り込み設定	
4.6.1 割り込み酸光レヘルとOS 官理の設定	
4.6.2 PENの設定の変更(RH850/U2Aのみ)	
4.0.3 割り込み設定の変更	
4.7 MCU マイクレーション 機能	
5	51
5. 脱口の官理	
5.1 リアースの脱音	
5.2 姉子の競音	
6. ソースの生成	53
6.1 生成ソースの出力	53
6.2 ソース生成先の設定	54
6.3 生成ファイルの構成とファイル名	56
6.4 クロック設定	59
6.5 端子設定	60
6.6 割り込み設定	61
7. ユーザープログラムの作成	62
7.1 コード生成の場合のカスタムコード追加方法	62
7.2 ユーザーアプリケーションコードの使用	63
8. 生成ソースのバックアップ	63
9. レポートの生成	64
9.1 全設定内容レポート	64
9.2 端子機能リスト、端子番号リスト設定内容(csv 形式)	65
9.3 MCU/MPU パッケージ図(png 形式)	65
10 ューザーコード保護機能	66
10.1 コーザーコード保護機能の指定なグ	00 AA
10.2 ユージーコード内域18.211年スノーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーー	00 aa
10.2	 67
10.3 1	07 67
10.3.1 成日の元工本日	۲0 وع
10.0.2 而口 97件(入力)公	
11. ヘルプ	70
11.1 ヘルプ	70 70



12.	参考ドキュメント	71
木-	ムページとサポート窓口	72



## 1. 概要

## 1.1 目的

本アプリケーションノートは、統合開発環境 e<sup>2</sup> studio でスマート・コンフィグレータを使用したプロ ジェクトの作成、基本的な使用方法について説明しています。

e<sup>2</sup> studio の使い方は、e<sup>2</sup> studio のユーザーズマニュアルを参照してください。

# 1.2 特長

スマート・コンフィグレータは、「ソフトウェアを自由に組み合わせられる」をコンセプトとしたユー ティリティです。ドライバコード生成、端子設定の機能でお客様のシステムへのルネサス製ドライバの組み 込みを容易にします。タイミング波形などのスマート・コンフィグレータのグラフィカルな表示により、ド ライバの設定が簡単になります。



2. プロジェクトの作成

スマート・コンフィグレータを使用して C/C++プロジェクトを生成する手順を、以下に説明します。 e<sup>2</sup> studio のプロジェクト作成ウィザードの詳細は、e<sup>2</sup> studio の関連ドキュメントを参照してください。

(1) e<sup>2</sup> studio を起動し、ワークスペースを指定します。起動後、[ファイル]→[新規]→[Renesas C/C++プロジェクト]→ [Renesas RH850] の順に選択してプロジェクト作成ウィザードを開きま す。

e2 poc workspace - test/test.scfg - e<sup>2</sup> studio

	ファイル(F)	編集(E)	ナビゲート(N)	検索(A)	プロジェクト(P)	Rene	esas	Views 実行(R)	Renesas AI	ウィン	ドウ(W) ヘルプ(H)
ſ	新規(	N)			Alt+シフト+N	N >		Renesas C/C++	Project	>	Renesas Debug
	ファイノ	レを開く(.).					c <sup>©</sup>	C/C++ Project			Renesas ASSP RISC-V
	🗀 דיל)	レ・システム	からプロジェクト	を開く			2	プロジェクト(R)…			Renesas RH850
	Recei	nt Files				>	9	サンプル(X)			Renesas RL78
	閉じる すべて	5(C) 閉じる(L)			Ctrl+V Ctrl+シフト+V	v v	3	その他(o)	Ctrl	+N	Renesas RX Renesas RA

図 2-1 新規プロジェクトの作成

(2) プロジェクト作成ウィザードで、 [Renesas RH850] → [Renesas CC-RH850 C/C++ Executable Project]を選択し、 [次へ] ボタンをクリックします。

New C/C++ Pr	roject		rolast			-		×
lemplates for i	Nellesas N	1650 P	roject					
All C/C++	RHBSO	Renesa A C Exec CC-RH t	<b>is CC-RH C</b> cutable Proje toolchain.	Executa ct for Rer	ible Proje nesas RH85	ct 50 usin <u>i</u>	g the	
	RHBSO	Renesa A C Libri toolchai	is CC-RH C ary Project fo in.	Library or Renesa:	Project s RH850 us	sing the	e CC-RH	
?	< 房	€る(B)	次へ(N)	>	終了(F)		キャンセノ	١

図 2-2 新規 C/C++プロジェクトのテンプレート



- (3) プロジェクト名を入力し、 [次へ] ボタンをクリックして次に進みます。
  - (例: CC-RH executable project, プロジェクト名: "Smart\_Configurator\_Example")

8			×
New Renesas CC-RH	Executable Project		<
New Renesas CC-RH Ex	ecutable Project		4
プロジェクト名(P) Smart	_Configurator_Example		
☑ デフォルト・ロケーション	の使用(D)		
ロケーション(L):	C:¥MyWork¥work¥RH850V1.10¥e2 poc workspace¥Smart_(	参照(R).	
ファイル・システムを選択(Y <b>ワーキング・セット</b>	<ul> <li>Create Directory for Project</li> <li>デフォルト 〜</li> </ul>		
ワーキング・セットにプロ	コジェクトを追加(T)	新規(W)	
ワーキング・セット(O):	~	選択(E)	
?	< 戻る(B) 次へ(N) > 終了(F)	キャンセ	l

図 2-3 New Renesas CC-RH executable project の作成

(4) デバイスおよびデバッグ設定を選択し、[次へ] をクリックします。(例: Target Device: RH850F1KM – 272 pins)

0	- 🗆 X
New Renesas CC-RH Executable Project	
Select toolchain, device & debug settings	
Toolchain Settings	
言語:	
ッールチェーン: Renesas CC-RH ~	[
ツールチェーン・バージョン: v2.04.01 ~	1
<u>ツールチェーンの管理…</u>	
Device Settings	Configurations
Target Board: Custom ~	✓ Hardware Debug 構成を生成
	E1 (RH850) ~
ターゲット・デバイス: R7F701653	□ Release 樺成を生成
<u>デバイスのアンロック</u>	I Release HAR 2 ± 10
エンディアン: Little ~	
プロジェクト・タイプ: デフォルト ~	
(ア) < 戻る(B) 次へ(N) >	終了(F) <b>キャンセル</b>

図 2-4 ツールチェーン、デバイス、デバッグ設定の選択



- (5) [コーディング・アシスタントツールの選択] ダイアログボックスで、[Use Smart Configurator]の チェックボックスを選び、[終了]をクリックします。
  - 【注】(4)で、スマート・コンフィグレータが対応しているデバイスを選択時のみ、[スマート・コン フィグレータを使用する]のチェックボックスが選択可能になります。

New Renesas CC-RH Executable Project
✓ Use Smart Configurator スマート・コンフィグレータは一つのユーザインタフェースでコード生成と FIT コンフィグレータの機能を併せ持ち、異なるタイブのデバイスドライバーやミドルウェアをインボート、設定、コードの生成を行います。また、スマート・コンフィグレータは統一されたクロック設定画面、割り込み設定画面、どン設定画面を持ちます。さらに、異なるタイブのデバイスドライバーやミドルウェアの使用による周辺機能、割り込み、ピン設定のハードウェア競合を検出します。(スマート・コンフィグレータはサポートデバイスでのみ使用可能です。)
Application Code Software Components Middleware & Drivers Device Drivers MCU Hardware
(?) < 戻る(B) 次へ(N) > 終了(F) キャンセル

図 2-5 コーディング・アシストツールの選択

(6) プロジェクト作成の完了を待ちます。

進行情報	
マートコンフィグレータ処理中	
スマートコンフィグレータエディタにデータをロード中	
	キャンセル

図 2-6 プロジェクト作成の処理



(7) 新規 C/C++プロジェクトの作成が成功すると、作成したプロジェクトがスマート・コンフィグレー タ・パースペクティブ上で開きます。

workspace - Smart_Configurator_Example/S	Smart_Configurator_Example.scfg - e <sup>2</sup> studio			- 🗆 ×
ファイル(F) 編集(E) ナビゲート(N) 検索(A) プロジ	ジェクト(P) Renesas Views 実行(R) ウィンドウ(W) ヘルプ(H	1)		
🗐 🛞 = 🍕 = 🛷 😒 📮 🎋 = 隆 =			Q 🛛 😰 🔤 c/c	++ 📓 スマート・コンフィグレータ
🔁 プロジェクト・エクスプローラ- 🗙 🖳 🗆	Smart_Configurator_Example.scfg ×	- 0	🐻 MCU/MPUパッケージ 🗙	- 0
😑 😫 🍸 🖇 🛩 🍜 Smart_Configurator_Example	概説	™ コードの生成 レポートの生成		フィルタ文字列を入力
> 🔊 Includes 🗸 😂 src	▼ 機能概要	0		
<ul> <li>&gt; is soc.gen</li> <li>&gt; is bootasm</li> <li>&gt; is cotarsm</li> <li>&gt; is cotarsm</li> <li>&gt; is idefine.h</li> <li>&gt; is output</li> <li>&gt; brash</li> <li>is smart_Configurator_Example.scfg</li> <li>is Smart_Configurator_Example Hardwa</li> </ul>		Application Code Software Components Middleware & Drivers Device Drivers MCU Hardware		
	● 確存の設定建設	~	5 00	
< >			P 7007	
	🐘 🔝 🐼   🖻	I □ ▼ □ □      I □		A 8 - 0
		↑ 記述/説明 ^	型	
		-		
<		>		ê1
	a 🗰 🕿 🖉 🕲			<u></u>

図 2-7 [スマート・コンフィグレータ] パースペクティブ



3. スマート・コンフィグレータの操作方法

3.1 スマート・コンフィグレータの表示

スマート・コンフィグレータの機能を十分に活用するためには、スマート・コンフィグレータ・パースペ クティブを確実に開いていることが必要です。開いていない場合は、e<sup>2</sup> studio ウィンドウ右上角のパース ペクティブを選択してください。

	クイック	・アクセス	B
e <sup>2</sup> パースペクティブを開く			×
100 C/C++ (デフォルト)			^
GIT Git			
🖏 Java			
<sup>と</sup> Java の型階層			
🔊 Java 参照			
🛼 LTTng Kernel			
E Remote System Explorer			
📓 Scripting			
🔜 SVN リポジトリー・エクスプローラー			
🔅 Synergy Configuration			
Target Explorer			
En Tracing			
『□-ド生成			
₽╹チーム同期化			
▶ リソース			~
			_
開<()	0)	キャンセノ	

図 3-1 [スマート・コンフィグレータ] パースペクティブを開く



## 3.2 操作手順

e<sup>2</sup> studio 上のスマート・コンフィグレータで周辺機能の設定し、ビルドするまでの手順を図 3-2 操作手順に示します。e<sup>2</sup> studio の操作については、e<sup>2</sup> studio の関連ドキュメントを参照してください。



図 3-2 操作手順

3.3 プロジェクト情報の保存先

スマート・コンフィグレータは、プロジェクトで使用するマイクロコントローラ、ビルド・ツール、周辺 機能、端子機能などの設定情報をプロジェクト・ファイル(\*.scfg)に保存し、参照します。

スマート・コンフィグレータのプロジェクト・ファイルは、e<sup>2</sup> studio のプロジェクト・ファイル(.project) と同階層にある"プロジェクト名.scfg"に保存します。

3.4 ウィンドウ

[スマート・コンフィグレータ]パースペクティブの構成を図 3-3 [スマート・コンフィグレータ]パー スペクティブに示します。

Sworkspace - Smart_Configurator_Example/S	mart_Configurat	or_Example.scfg - e <sup>2</sup> studio			- 🗆 X
ファイル(F) 編集(E) ナビゲート(N) 検索(A) プロシ	ゴクト(P) Renes	as Views 実行(R) ウィンドウ(W) ヘルプ(H)			
📓 🛞 = 🔦 = i 🖉 😂 i 🖨 i 🏘 = 💁 =				Q 🔡	🗟 C/C++ 🔮 גד-ריבטסרלע-פ
눱 プロジェクト・エクスプローラー 🗙 📃 🗖	Smart_Cont	figurator_Example.scfg $ imes$	- 0	🗗 MCU/MPUパッケージ 🗙	- 0
🖻 😫 🏹 🕴	概説				➡ フィルタ文字列を入力 >>
Smart_Configurator_Example			MENT NU MENT L		
✓ 🖉 src	▼ 機能概要		0		
> 🗁 smc_gen	$\sim$	概要	0		
> S boot.asm	$\square$	<u>概要</u> をクリックすると、スマート・コンフィグレータ	(2)		0000000000
> S cstart.asm		の機能を確認することができます。			
> 🇁 output		動画	Application Code		
> 🧽 trash		スマート・コンフィグレータの紹介			
Smart_Configurator_Example.scfg		関連動画	Software Components		
Smart_configurator_example Hardwa		最新情報	Middleware & O		
		<u>最新情報</u> をクリックすると、最新リリースの情	RTOS		
		報を確認することができます。	Device Drivers		
		すべての <u>リリースノート</u> を表示	ato	· • • • • • • • • • • • • • • • • • • •	1653 ee et et et
_	aIII	製品ドキュメント	MCU Hardware		
		ユーザーガイド		· • • • • • • • • • • • • •	<b>.</b>
		APIZLIPH			
		<u>アフリケーションノート</u> ツールニューフ			
	2				
	● 現在の設定 概要 ポード ク	2949館 ロック コンボーネント 端子 割り込み	•	▶ 凡例	
או-עיר 🗉		R. 🖬 🔂 🛹 🛙	マ P 日 日 P コンフィグレーションチェック ×		₩ 8 - 0
スマート・コンフィグレータ出力			0項目		00-
			△ 記述/説明 ^	型	
	C	<b>`</b>			
	(4	)		(5)	
<			>		
					C.F
	2	🕿 🏏 🔘			

図 3-3 [スマート・コンフィグレータ] パースペクティブ

- ① プロジェクト・エクスプローラー
- ② スマート・コンフィグレータビュー
- ③ MCU/MPU パッケージビュー
- ④ コンソールビュー
- ⑤ コンフィグレーションチェックビュー

3.4.1 プロジェクト・エクスプローラー

プロジェクトのフォルダ構成をツリーで表示します。



図 3-4 プロジェクト・エクスプローラー

ビューが開いていない場合は、e<sup>2</sup> studio メニュー上の [ウィンドウ] → [ビューの表示] → [その他] を選択し、開いた [ビューの表示] ダイアログボックスから [一般] → [プロジェクト・エクスプロー ラー] を選択してください。

3.4.2 スマート・コンフィグレータビュー

[概要],[ボード],[クロック],[システム],[コンポーネント],[端子],[割り込み]の7つのページから構成されます。タブをクリックして、ページを選択すると選択したタブに応じて内容が切り替わります。



図 3-5 スマート・コンフィグレータビュー

ビューが開いていない場合は、 [プロジェクト・エクスプローラー] からプロジェクト・ファイル(\*.scfg) を右クリックし、コンテキストメニューから [開く] を選択してください。 3.4.3 MCU/MPU パッケージビュー

MCU/MPU パッケージ図上に端子状態を表示します。端子設定を変更することもできます。

MCU/MPU パッケージビューは、[割り当てられた機能]、[ボード機能]、[シンボリック名]の3種類種類の切り替えが行えます。

・[割り当てられた機能]は、端子設定の割り当て状況を表示します。

- ・[ボード機能]は、ボードの初期端子設定情報を表示します。
- ・[シンボリック名]は、端子にユーザー定義したシンボリック名が表示されます。

シンボリック名のマクロ定義は、Pin.h ファイル内のポート読み取りまたは書き込み関数とともに生成 されます。

ボードの初期端子設定情報は、[ボード]ページの[ボード:]で選択したボードの端子情報です (「4.1.2 ボード選択」、「4.5.6 ボード端子設定情報を使用した端子設定」参照)。

【注】シンボリック名の機能は、RH850/F1KM および RH850/F1KH には適用されません。

シンボリック名の機能は、APORT、JPORT、および IPORT には適用されません。



図 3-6 MCU/MPU パッケージビュー

ビューが開いていない場合は、e<sup>2</sup> studio メニュー上の [Renesas Views] → [スマート・コンフィグレー タ] → [MCU/MPU パッケージ] を選択してください。



# 3.4.4 コンソールビュー

スマート・コンフィグレータビューまたは MCU/MPU パッケージビューでの設定変更内容が表示されます。

🖳 או-עעב 🖳 🖬 🔛 🛫	📑 🗕 🗖
スマート・コンフィグレータ出力	
M05000001: 端子 K25 に X1 の機能が害り当てられています M05000001: 端子 L25 に X2 の機能が害り当てられています	^
	$\sim$
<	>

#### 図 3-7 コンソールビュー

ビューが開いていない場合は、e<sup>2</sup> studio メニュー上の [ウィンドウ] → [ビューの表示] → [その他] を選択し、開いた [ビューの表示] ダイアログボックスから [一般] → [コンソール] を選択してくださ い。

## 3.4.5 コンフィグレーションチェックビュー

端子競合が発生した場合に、その内容を表示します。

ミンフィグレーションチェック ※ 0項目		7	000	
	型			

# 図 3-8 コンフィグレーションチェックビュー

ビューが開いていない場合は、 $e^2$  studio メニュー上の [Renesas Views]  $\rightarrow$  [スマート・コンフィグレータ]  $\rightarrow$  [コンフィグレーションチェック] を選択してください。



RH850 スマート・コンフィグレータ

4. 周辺機能の設定

周辺機能は、スマート・コンフィグレータビューから選択します。

4.1 ボード設定

ボードページでは、ボードおよび、デバイスの変更が可能です。

- 4.1.1 デバイス選択
  - [ … ] ボタンをクリックすると、デバイスが選択できます。

「4.7 MCU マイグレーション機能」の手順に従いデバイス変更を行ってください。

鬱 *Smart_C	onfigurator_Example.scfg ×		
デバイス達	訳	<b>じ</b> コードの生成	
デバイス選択	R		è 2
ボード:	カスタムユーザポード 〜 …		
デバイス:	R7F701653		
<u></u>	<u>ード情報をダウンロードする</u> =		
概要 ボード	クロック コンポーネント 端子 割り込み		

図 4-1 デバイス選択

4.1.2 ボード選択

[ … ] ボタンをクリックすると、ボードが選択できます。

ボード選択により、以下の一括変更が可能です。

- 端子割り当て(初期端子設定)
- メインクロック周波数
- サブクロック周波数
- デバイス
- オンチップ・デバッグ動作設定とエミュレータ設定

上記ボード設定情報は、Board Description File (.bdf) に定義されています。

Renesas 製ボード(Renesas Starter Kit 等)の.bdf ファイルを WEB からダウンロードし、インポートが可能です。

また、アライアンスパートナーが公開している.bdf ファイルを WEB からダウンロードし、インポートすることで、アライアンスパートナー製ボードの選択が可能となります。

選択したボードに応じて、デバイスが変更され、デバイスの変更は e<sup>2</sup> studio プロジェクトのターゲット・デバイスに反映されます。「4.7 MCU マイグレーション機能」の手順を参照してください。

Image: Smart_Configurator_Example.scfg ×	- 0
デバイス選択	「■
デバイス選択	24
ボード: カスタムユーザポード 〜	
デバイス: R7F701653	
ポード情報をダウンロードする	
概要   ホート   クロック   コンホーネント   端子   割り込み	

図 4-2 ボード選択

[検出された問題] に表示されたメッセージを確認して [次へ] をクリックします。

🔤 リファクタリング			×
Change Device 下のリストに示されている情報を確認してください。'次へ >' をクリックして次の項目を表示す	るか、'終了'		
をクリックします。 検出された問題 ④ This change cannot be undone. Please make sure you backup this project before	e continuin	a.	<b>₽</b> ↔
使用可能なコンテキスト情報はありません		5	
		++1/17	
、 大 (D) 八 (N) > 於 (r)		イャノビル	

図 4-3 検出された問題

表 4-1	デバイス変更の[検出された問題]の表示一	覧
-------	----------------------	---

メッセージ	説明
This change cannot be undone. Please make	デバイスを変更すると変更前に復元できませんので、
sure you backup this project before continuing.	ノロンエクトのハックアッノ俊に美行してくたさい。



[実行される変更] で、変更する項目を選択して [終了] をクリックします。

	_		×
<b>Change Device</b> The following changes to 4 files are necessary to perform the refactoring.			
実行される変更		<b>₽ 0</b>	7 -
<ul> <li>Change Device for SC_example</li> <li>Change Device for SC_example</li> <li>Change Device for SC_example</li> <li>Change Device for SC_example</li> <li>SC_example HardwareDebug</li> <li>SC_</li></ul>			
使用可能なプレビューはありません			
⑦ < 戻る(B) 次へ(N) > 終了(F)	)	キャンセル	

図 4-4 実行される変更項目確認

4.1.3 ボード設定のエクスポート

ボード設定のエクスポートは、以下の手順で行います。

- (1) ボードページで、 [ボードの設定をエクスポート] 🛂 ボタンをクリックします。
- (2) 出力場所を選択し、エクスポートするファイル名(表示名)を入力します。

Smart_Co	nfigurator_Example.scfg ×		- 0
デバイス通	訳	<b>じ</b> コードの生成	
デバイス選択	<b>२</b>		Ъď
ボード:	カスタムユーザポード 🗸 🛄		(1)
デバイス:	R7F701653		
	ポード情報をダウンロードする		
概要 ボード	クロック コンボーネント 端子 割り込み		

#### 図 4-5 ボード設定のエクスポート(bdf 形式)



# 4.1.4 ボード設定のインポート

ボード設定のインポートは、以下の手順で行います。

- (1) [ボードの設定をインポート] 🚵 ボタンをクリックし、bdf ファイルを選択してください。
- (2) インポートしたボード設定がボード選択の選択肢に追加されます。

鬱 Smart_Co	nfigurator_Example.scfg $ imes$	- 0
デバイス選	訳	💽 🤷 コードの生成 レポートの生成
デバイス選択	R	<u>≥</u> ∠
ボード:	カスタムユーザポード 〜	(1)
デバイス:	R7F701653 (2)	
	<u>ポード情報をダウンロードする</u>	
概要 ボード	クロック コンボーネント 端子 割り込み	

図 4-6 ボード設定のインポート(bdf 形式)

ー度インポートしたボード設定は、同じデバイスグループのプロジェクトにおいてボード選択の選択肢に 表示されます。



## 4.2 クロック設定

クロックページでは、システムクロックを設定することができます。クロックページで作成した設定は、 全てのドライバおよびミドルウェアで使用されます。

クロック設定を更新するには、以下の手順で行います。

- (1) ボード仕様に従って各クロックの周波数を指定します(ただし、一部の内部クロックは周波数が固定です)。
- (2) PLL 回路を使用する場合、PLL のクロックソースを選択します。
- (3) マルチプレクサの出力クロックのソースを選択します。
- (4) ドロップダウンリストで分周比を設定します。



図 4-7 クロック設定



# 4.3 システム設定(RH850/U2Aのみ)

[システム]タブで使用する CPUn(PEn)を選択します。 CPU0(PE0)は常にデフォルト設定として選択されます。 システム設定は、RH850/U2A のみサポートします。

システム設定	5 (1) (1) (1) (1) (1) (1) (1) (1) (1) (1)
CPUJ7選択	^
<ul> <li>CPU0(PE0)を使用(デフォルト)</li> </ul>	
r_cg_intvector_PE0.cおよびr_cg_intc_P <sup>1</sup> 0.cが "¥general"フォルダの下に生成され、WDT0ファイルが生成されます(WDT0が使用されている場合)。	
r_cg_intvector_PE1.cおよびr_cg_intc_PE1.cが "¥general"フォルダの下に生成され、WDT1ファイルが生成されます(WDT1が使用されている場合)。	
r_cg_intvector_PE2.cおよびr_cg_intc_P <sup>1</sup> 2.cが "¥general"フォルダの下に生成され、WDT2ファイルが生成されます(WDT2が使用されている場合)。	
r_cg_intvector_PE3.cおよびr_cg_intc_P <sup>I</sup> 3.cが "¥general"フォルダの下に生成され、WDT3ファイルが生成されます(WDT3が使用されている場合)。	
モ要  ポード   クロック <mark>システム</mark> コンポーネント   端子   割り込み	

図 4-8 システム設定

- たとえば、CPU0(PE0)と CPU1(PE1)を使用する場合、以下の設定が必要です。
- (1) スマート・コンフィグレータの[システム]ページをクリックし、CPU0(PE0)と CPU1(PE1)を選択しま す。 コードを生成します。

*Smart_Configurator_Example.scfg ×	
システム設定	○
CPUJア選択	í
✓ CPU0(PE0)を使用(デフォルト)	
r_cg_intvector_PE0.cおよびr_cg_intc_PE0.cが "¥general"フォルダの下に生成され、WDT0ファイルが生成されます(WDT0が使用されている場合)。 ☑ CPU1(PE1)を使用	
r_cg_intvector_PE1.cおよびr_cg_intc_PE1.cが "¥general"フォルダの下に生成され、WDT1ファイルが生成されます(WDT1が使用されている場合)。	
□ CPU2(PE2)を使用	
r_cg_intvector_PE2.cおよびr_cg_intc_PE2.cが "¥general"フォルダの下に生成され、WDT2ファイルが生成されます(WDT2が使用されている場合)。 □ CPU3(PE3)を使用	
r_cg_intvector_PE3.cおよびr_cg_intc_PE3.cが "¥general"フォルダの下に生成され、WDT3ファイルが生成されます(WDT3が使用されている場合)。	
概要  ボード   クロック   システム   コンポーネント   端子   割り込み	

図 4-9 CPU コア選択

(2) 「cstart\_pm0.asm」ファイルを新しいファイル「cstart\_pm1.asm」としてコピーします。





- 図 4-10 cstart\_pm0.asm のコピー
- (3) 「cstart\_pm1.asm」ファイルの内容を以下のように変更します。



#### 図 4-11 cstart\_pm1.asm の内容変更

(4) 「boot.asm」ファイルの内容を以下のように変更します。



#### 図 4-12 boot.asm の内容変更



- 💽 プロパティ: Smart\_Configurator\_Example  $\times$ フィルタ入力 設定 > リソース ✓ C/C++ ビルド Configuration: HardwareDebug [アクティブ] ~ 構成の管理.. ツールチェイン・エディター ビルド変数 ロギング 🔞 ツール設定 Toolchain Device 🎤 ビルド・ステップ 😤 ビルド成果物 🔜 バイナリー・パーサー 😣 エラー・パーサー 環境 設定 実行開始アドレス (-entry) ✓ 
  Some Common > C/C++ 一般 Code generation セクション配置 (-start) RESET\_A,EIINTTBL\_A/0,RESET\_B,EIINTTBL\_E Renesas OE Miscellaneous PIC/PID Task Tags > Validation 🗸 🛞 Compiler × ビルダー Preprocess セクション・ビューアー: プロジェクト・ネーチャー 🖄 Output アドレス セクション名 プロジェクト参照 Optimization 0x0000000 RESET\_A 実行/デバッグ設定 Code generation EIINTTBL\_A セクションの追加 MISRA C rule check 0x00010000 Miscellaneous セクション・オーバーレイの追加 EIINTTBL\_PE1.const 실 User ヤクションの除去 0x00020000 ✓ 🛞 Assembler .const Preprocess .INIT DSEC.const 上へ移動 .INIT\_BSEC.const 🖄 Output 下へ移動 🖄 User .text インポート... ✓ 🛞 Linker .data エクスポート... 🖄 Input 0x00800000 RESET B Input advanced EIINTTBL B 🖄 Output 0xFDE00000 .data.R 🖄 Output advanced □ リンカースクリプトの上書き: 🖄 List Optimization 参照 Section 再適用 Section advanced Verify OK キャンセル ? 適用して閉じる キャンセル
- (5) PE1 のセクションを設定します。

図 4-13 セクションの設定

(6) 「r\_cg\_intc\_PE1.c」ファイルにある API 名「R\_Interrupt\_Initialize\_ForPE」を 「R\_Interrupt\_Initialize\_ForPE1」に変更します。

r_cg_intc_PE1	l.c ×
2	* DISCLAIMER.
19	
21	⊕ * File Name : r_cg_intc_PE1.c.
27	⊕ Pragma directive.
29	⊖ /* Start user code for pragma. Do not edit comment generated here */
30	/* End user code. Do not edit comment generated here */
31	
33	Includes.
35	<pre>#include "r_cg_macrodriver.h"</pre>
36	<pre>#include "r_cg_userdefine.h"</pre>
37	$\odot$ /* Start user code for include. Do not edit comment generated here */
38	/* End user code. Do not edit comment generated here */
39	
41	⊕ Global variables and functions.
₿ 43	<pre>void R_Interrupt_Initialize_ForPE1(void)</pre>
44	{
45	}
46	

図 4-14 API 名変更

(7) API「R\_Interrupt\_Initialize\_ForPE1」の関数宣言を追加します。



▶ r_cg_macrodriver.h × 🖻 r_cg_intc_PE1.c	
178 #define PCR ALT OUT SETTING	(0xEEECEEEEUL) /* ALT Output setting */
179 #define PCR_DIRECT_ALT_MODE_SETTING	(0xFFFCFFFFUL) /* Direct ALT mode setting */
180 /* Write protected macro definition */	(
181 #define WRITE PROTECT ENABLE	(0xA5A5A501UL) /* Write protected */
182 #define WRITE PROTECT DISABLE	(0xA5A5A500UL) /* Write protected */
183 #define PORT WRITE PROTECT DISABLE	(0x0000000UL) /* Write protected */
184 /* Specify the interrupt bind (request) de	stination */
185 #define INT CPU PE0	(0x0000000UL) /* Bound to PE0 */
186 #define_INT_CPU_PE1	(0x0000001UL) /* Bound to PE1 */
187 #define_INT_CPU_PE2	(0x00000002UL) /* Bound to PE2 */
188 #define _INT_CPU_PE3	(0x0000003UL) /* Bound to PE3 */
189	
191 ⊕ Typedef definitions.	
193	
194 typedef signed char int8_t;	
195 typedef unsigned char uint8_t;	
<pre>196 typedef signed short int16_t;</pre>	
197 typedef unsigned short uint16_t	;
198 typedef signed long int32_t;	
199 typedef unsigned long uint32_t	;
200 typedef signed long long int64_t;	
201 typedef unsigned long long uint64_t	;
202 typedef unsigned short MD_STATU	5;
203 #detineIYPEDEF	
204 #endit	
205 207 @ Global functions	
200 woid B Systeminit(woid):	
209 Void R_Systeminic(Void);	
211 void R Interrupt Initialize ForPE1(void):	
212 97. STAFL USER CORE TOR TUNCTION, US NOT EN	it comment generated here */
213 /* End user code. Do not edit comment gene	rated here */
214 #endif	
図 4-15 関数国	宣言の追加

(8) API「R\_Interrupt\_Initialize\_ForPE1」の呼び出し記述を追加します。

.c

r_cg_systemin	it.c ×
2	* DISCLAIMER.
19	
21	<pre>  * File Name : r_cg_systeminit.c. </pre>
27	Pragma directive.
29	⊖ /* Start user code for pragma. Do not edit comment generated here */
30	<pre>/* End user code. Do not edit comment generated here */</pre>
31	
33	Includes.
35	<pre>#include "r_cg_macrodriver.h"</pre>
36	<pre>#include "r_cg_userdefine.h"</pre>
37	<pre>#include "Config_ADCJ0.h"</pre>
38	<pre>#include "r_cg_cgc.h"</pre>
39	⊖ /* Start user code for include. Do not edit comment generated here */
40	/* End user code. Do not edit comment generated here */
41	
43	Global variables and functions
45	> /* Start user code for global. Do not edit comment generated here */
40	7° End user code. Do not edit comment generated here "7
47	@* Eurotion Names R Systeminit
+9 E /	evold D Sustaminit(void)
55	/
56	
57	R Interrupt Initialize ForPE1():
58	/* Set peripheral settings */
59	R CGC Create():
50	R Config ADCJ0 (reate():
51	}
52	•
	凶 4-10 肉奴叶の山し記述の迫加



# 4.4 コンポーネント設定

コンポーネントページは、ドライバやミドルウェアをソフトウェアコンポーネントとして組み合わせま す。追加したコンポーネントは、左側のコンポーネントツリーに表示されます。



図 4-17 コンポーネントページ

#### 4.4.1 コンポーネントー覧表示とハードウェアー覧表示の切り替え

スマート コンフィグレータには、コンポーネント一覧表示とハードウェア一覧表示の 2 つのツリー ビューが用意されています。 ユーザーは、次のアイコンをクリックして 2 つのビューを切り替えることが できます。

・コンポーネント一覧表示

ツリー ビューには、コンポーネントのカテゴリごとにコンポーネントが表示されます。

・ハードウェアビューで表示

ツリー ビューには、ハードウェアリソース階層内のコンポーネントが表示されます。

<ul> <li>*Smart_Configurator_Example.scfg ×</li> <li>ソフトウェアコンポーネント設定</li> <li>コンポーネント 辿 込 長 巨 正 寺 マ 設定</li> </ul>
ソフトウェアコンポーネント設定 コンポーネント 🚵 🚵 尾 🗉 🏦
コンポーネント 🚵 🛃 🖳 🖽 📫 🕶 設定
■ コンポーネント一覧表示
ハードウェアー覧表示
71ルタ入力
柳葉 ボード クロック システム コンポーネント 端子 割り込み
1% &   小     ノロノノ   ノハノム   コノ小   イノト   新 ]   割り心の

図 4-18 ビューの切り替え



- 4.4.2 コード生成コンポーネントの追加方法 コポーネントの追加は、以下の手順で行います。
  - (1) [コンポーネントの追加] 🍾 アイコンをクリックします。

🏟 *Smart_Configurator_Example.scfg 🔀	
ソフトウェアコンポーネント設定	じまた 100 - ドの生成 レポートの生成
コンポーネ… ↓ 』 □ □ → ▼ 設定	١
(1) マーク マーク マーク マーク マーク マーク マーク マーク	
概要  ボード  クロック  システム コンポーネント  端子   割り込み	

図 4-19 コンポーネントの追加

(2) [コンポーネントの追加] ダイアログボックスの [ソフトウェアコンポーネントの選択] のリストからコンポーネントを選択し、[次へ] をクリックします。(例: A/D コンバータ)

📴 コンポーネントの追加			_		×
ソフトウェアコンポーネントの選択					-
使用可能なコンポーネントの一覧から選択してくださ	LA				
	•				
カテコリ 全(					~
機能全て					$\sim$
71119					
<u>コンポーネント ^ </u>	//ํ_>ั่ว`≀	•			^
➡ A/Dコンパータ	1.6.0				
ATOM Signal Output Mode Compare	1.1.1				
HATOM Signal Output Mode Immediate	1.1.1				
H ATOM Signal Output Mode PWM	1.1.1				
ATOM Signal Output Mode Serial	1.1.1				
Dead Time Module	1.1.0				
	1.5.0				
	1.3.0				
GTM Clock	1.1.0				
	1.3.1				
MSPI 729	1.3.1				~
✓ 最新バージョンのみ表示					
説明					
アナログ-デジタル(A/D)変換回路は、アナログ入ナ	」をデジタル信号に変換する機能	きです。			~
					$\sim$
基本設定					
(?) < 戻	る(B) 次へ(N) >	終了(F)		キャン	セル

図 4-20 コード生成コンポーネントの追加

 (3) [コンポーネントの追加] ダイアログボックスの [選択したコンポーネントのコンフィグレーション を追加します] ページで、適切なコンフィグレーション名を入力、またはデフォルト名を使用しま す。(例: Config\_ADCJ0)

リソースを選択し、[終了] をクリックします。(例: ADCJO)

🔯 コンポーネントの追加			
選択したコンポーネントの 加します	Dコンフィ <b>グレ</b> ーションを追		
A/Dコンパータ コンフィグレーション名: リソース:	Config_ADCJ0		~
<b>?</b> < 戻る(B)	次へ(N) > 終了	7 (F)	キャンセル

図 4-21 コンポーネントのコンフィグレーション追加



4.4.3 ソフトウェアコンポーネントの削除

プロジェクトからソフトウェアコンポーネントを削除するには、以下の手順で行います。

- (1) コンポーネントツリーからソフトウェアコンポーネントを選択します。
- (2) [コンポーネントの削除] 🦥 アイコンをクリックします。

コンポーネント	è d la		⊞ ≱	•
			1	
フィルタ入力				
🗸 🗁 ドライバ				
🗸 🌽 ALS				
💣 Co	nfig_TAUD0_0	)		
💕 Co	nfig_TAUJ0_0			
🚽 Co	nfig_TAUJ0_1			
∽ 🧁 A/D :	コンバータ			
💕 Co	nfig_ADCJ0			
概要 ボード クロック	システム コン	ポーネ	ント 端	子輩
図 4-22 ソフトワ	ウェアコンポ	ーネ	ントのi	削除

コンポーネントツリーから、選択したソフトウェアコンポーネントが削除されます。

この操作では、プロジェクトに登録されているソースファイルは削除されません。[コード生成] 🚺 ボタ ンでソースファイル出力を行うと、削除したソフトウェアコンポーネントのソースファイルが削除されま す。



4.4.4 ソフトウェアコンポーネントの設定

ソフトウェアコンポーネントを設定するには、以下の手順で行います。

- (1) コンポーネントツリーにある ソフトウェアコンポーネントを選択します。(例: A/D コンバータ)
- (2) 右側の設定パネルでドライバを設定します。図は例です。
  - a. [サンプリング制御設定] で [24 サイクル] を選択します。
  - b. [スキャングループ選択] で [スキャングループ2を使用する] を選択します。
  - c. [仮想チャンネルのエンドポインタ] に [3] を入力します。

フトウェアコンポーネント設定				💽 📟 コードの生成 レポートの生成
レポーネント 迠 🛃 📙 🕀 🛱 🤅		共通動作設定		^
ت 😮		中断方法	同期サスペンド	~
<b>ጋ</b> ィルタ入力		12/10ビット選択モード	12ビットモード	$\sim$
> ► ドライバ (1)		アライメント制御	右詰め	$\sim$
		サンプリング制御の設定		
Coning_ADCAO		サンプリング時間	24サイクル	✓ (2) a.
		スキャングループ選択		
		(複数のスキャングループの仮想チャネルを認	没定することはできません)	
(	2) b.	☑ スキャングループ1を使用		
		開始仮想チャネルポインタ	0	
		終了仮想チャネルポインタ	0	
		☑ スキャングループ2を使用		
		開始仮想チャネルポインタ	0	
		終了仮想チャネルポインタ	3	(2) c.
		□ スキャングループ3を使用		
		開始仮想チャネルポインタ	0	
		終了仮想チャネルポインタ	0	
		□ スキャングループ4(PWM-Diag)を使用		
	<			>

図 4-23 ソフトウェアコンポーネントの設定

ソフトウェアコンポーネントのコード生成は、デフォルトで生成する設定になっています。

ソフトウェアコンポーネントを右クリックし、 [ ✓ □-ド生成 ] をクリックすると、 [ □-ド生成 ] に変わりコードを生成しません。

[ <sup>□-ド生成</sup>]をクリックすると、 [ ✓ <sup>□-ド生成</sup> ]に変わりコードを生成します。

4.4.5 ソフトウェアコンポーネントのリソース変更

スマート・コンフィグレータでは、ユーザーは ソフトウェアコンポーネントのリソースを変更すること ができます(例: ADCA0 から ADCA1 に変更)。互換性のある設定は、現在のリソースから新しく選択し たリソースへ移行することができます。

現在のソフトウェアコンポーネント用にリソースを変更するには、以下の手順で行います。

- (1) ソフトウェアコンポーネントを右クリックします(例: Config\_ADCA0)。
- (2) コンテキストメニューから [リソースの変更]を選択します。

<ul> <li> <i>→</i> ドライバ</li></ul>	コンポーネ	ント 🛛 🚵 🛃 🖻 🖽 🚔 🔻
フィルタ入力 <ul> <li></li></ul>		10 T
<ul> <li>✓ ⇒ ドライバ</li> <li>✓ A/D コンパータ</li> <li>✓ コード生成</li> <li>(1)</li> <li>✓ コード生成</li> <li>リソースの変更 (2)</li> <li>※ 削除</li> <li>複製</li> <li>リネーム</li> <li>リセット時のデフォルト</li> </ul>	フィルタン	入力
<ul> <li>◇ A/D コンパータ</li> <li>Config_ADCA0 (1)</li> <li>✓ コード生成 リソースの変更 (2)</li> <li>※ 削除 複製 リネーム リセット時のデフォルト</li> </ul>	<ul> <li>✓ </li> </ul>	ドライバ
<ul> <li>Config_ADCA0 (1)</li> <li>✓ コード生成 リソースの変更 (2)</li> <li>※ 削除 複製 リネーム リセット時のデフォルト</li> </ul>	~ 🤅	→ A/D コンバータ
<ul> <li>✓ コード生成</li> <li>リソースの変更(2)</li> <li>業 削除</li> <li>複製</li> <li>リネーム</li> <li>リセット時のデフォルト</li> </ul>		Config_ADCA0 (1)
<ul> <li>✓ コード生成         <ul> <li>リソースの変更(2)</li> <li>業 削除             <ul></ul></li></ul></li></ul>		
リソースの変更 (2)	~	コード生成
<ul> <li>         ・ 削除         <ul> <li></li></ul></li></ul>	L	リソースの変更 (2)
複製 リネーム リセット時のデフォルト	*	削除
リネーム リセット時のデフォルト		複製
リセット時のデフォルト		リネーム
リセット時のデフオルト		
		リセット時のテノオルト
+ コンフィグレーションの追加 >	+	コンフィグレーションの追加 >

図 4-24 リソースの変更

- (3) [リソースの選択] ダイアログボックスにある新しいリソースを選択します(例: ADCA1)。
- (4) [次へ] ボタンが有効になるので、クリックします。

ג-עע 📴	の選択				×
リソースの	選択				
使用可能	なリソースの一覧だ	から選択してください			
リソース:	ADCA0				$\sim$
	ADCAO				
	ADCA1				
	(3)				
		(4)			
?	< 戻る(B)	次へ(N) >	終了(F)	キャンセル	ŀ



RH850 スマート・コンフィグレータ

- (5) コンフィグレーション設定は、[コンフィグレーション設定の選択]ダイアログボックスに表示され ます。
- (6) 設定が変更可能であるかを確認します。
- (7) テーブル内の設定を使用するか、デフォルト設定を使用するか選択します。
- (8) [終了]をクリックします。

	3)7イグレーション設定一覧 (7) (7)			
	設定項目	値	· · · · · · · · · · · · · · · · · · ·	^
)	中断方法	同期サスペンド	なん う 能	
	12/10ビット選択モード	12ビットモード	111	
	アライメント制御	ち詰め	111	
	サンプリング時間	18サイクル	(11)	
	スキャングループ1を使用	チェック	はい	
	開始仮想チャネルポインタ	0	はい	
	終了仮想チャネルポインタ	0	はい	
	スキャングループ2を使用	チェックを外す	はい	
	開始仮想チャネルポインタ	0	はい	

図 4-26 新しいリソース設定の確認

リソースは、自動的に更新されます。 (例:ADCA0 から ADCA1)



Image: Smart_Configurator_Example.scfg ×				- 0
ソフトウェアコンポーネント設定				🐻 📑 コードの生成 レポートの生成
コンポーネント 🚵 🛃 🗦 🗉 🕀 🚔 🔻	✓ スキャングループ1を使用 開始仮想チャネルポインタ	0		^
tu Tu	終了仮想チャネルポインタ	0		
7ィルタ入力	□ スキャングループ2を使用			
	開始仮想チャネルポインタ	0		
Config_ADCA0	終了仮想チャネルポインタ	0		
	□ スキャングループ3を使用			
	開始仮想チャネルポインタ	0		
	終了仮想チャネルポインタ	0		
	□ スキャングループ4(PWM-Diag)	を使用		
	ノイズフィルタ設定			
	サンプリングクロック周波数	サンプリングクロック供給/1	~	
	サンプリング数	2	~	
	仮想チャネル設定			
	仮想チャネル00	物理チャンネル選択	ADCA1I0	~
		自己診断チャネル	ANI100	~
		□ 内蔵プルダウン抵抗を有効		~
<	1.1			>
概要 ボード クロック コンポーネント 端子 割り込み				

図 4-27 自動的に更新されるリソース

コンポーネント名を変更する場合は、以下の手順で行います。

(9) コード ジェネレーター構成を右クリックします。

(10) [名前の変更] を選択して、構成の名前を変更します (例: Config\_ADCA0 を Config\_ADCA1 に変更 します)。



コンフィグレーション名を変更する場合は、以下の手順で行います。

- (9) ソフトウェアコンポーネントを右クリックします。
- (10) [リネーム]を選択して、コンポーネントに再度名前をつけます(例: Config\_ADCA0 を Config\_ADCA1)。

コンポーネント 🛛 🚵 🛃 🕞 🕀 静 🔻
10 To
フィルタ入力
🗸 🗁 ドライバ
✓ → A/D コンバータ
Config_ADCA0 (9)
✓ コード生成
リソースの変更
😫 削除
複製
リネーム (10)
リセット時のデフォルト
+ コンフィグレーションの追加 >

図 4-28 コンフィグレーションに再度名前をつける

# 4.4.6 コンポーネントの設定のエクスポート

[コンポーネント]ページに[コンフィグレーションのエクスポート]<sup>44</sup> ボタンをクリックすると、現在の設定を\*.xml ファイルとしてエクスポートできます。

フィルタ入力	⇒ →
フィルタ入力 、	
<ul> <li>✓ ⇒ ドライバ</li> <li>✓ ⇒ タイマ</li> <li>Config_TAUD0_0</li> <li>Config_TAUJ0_0</li> <li>Config_TAUJ0_1</li> <li>✓ ⇒ Δ(D_T)// - ダ</li> </ul>	
✓	
Config_TAUD0_0 Config_TAUJ0_0 Config_TAUJ0_1	
Config_TAUJ0_0 Config_TAUJ0_1	
💣 Config_TAUJ0_1	
Config_ADCJ0	
概要 ボード クロック システム コンポーネント	端子 割
[] <b>4-29</b> コンフィグレーションのエク	1 - 12



# 4.4.7 コンポーネントの設定のインポート

[コンフィグレーションのインポート] 🏜 ボタンをクリックし、エクスポートした\*.xml ファイルを選択 すると、\*.xml ファイルの内容をインポートします。

コンポーネント 🔤 🖄 🖳 🕀 🛱 🔻
10 T
7ィルタ入力
🗸 🧁 ドライバ
✓ 🧁 ୨イマ
Config_TAUD0_0
Config_TAUJ0_0
Config_TAUJ0_1
✓ → A/D コンバータ
Config_ADCJ0
概要 ボード クロック システム コンポーネント 端子 書

図 4-30 コンフィグレーションのインポート

4.4.8 コンポーネントの基本設定

モジュールの保存先、依存関係などのコンポーネントの基本設定を変更できます。変更するには、[コン ポーネントの追加]ダイアログ(図 4-20 コード生成コンポーネントの追加)に表示される[ソフトウェア コンポーネントの選択]ページの[基本設定]リンクをクリックし、[設定]ダイアログを表示させます。

📴 設定			— 🗆 X
<u>フィルタ入力</u>	コンポーネント		<
Support Folders Synergy Configuration Editor Synergy License Tracealyzer TraceX マスマート・コンフィグレータ MCU/MPUパッケージ外観 MMU コンボーネント 端子エラー/警告 スマート・ブラウザー デパイス・アドイン・サポート 起動設定 Scripting Toolchains Tracing Validation > Version Control (Team)	▲ コード生成設定 生成条件: trashへの/(ツクアップ API閣数出力: 依存コンポーネントの 依存関係の追加: 依存関係の追認: ファルダ設定 モジュールの保存先付	コンポーネントが存在する           ブ数 (0-20):         5           設定に合わせてすべては           設造加方法を選択してください           依存コンポーネントを追加する           より新しいパージョンの依存コンポーネン           はModule Downloadページで設定で           デファ	2場合は何もしない > は力する > に力する > ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・
? 🎽 🖆 🔘		適	用して閉じるキャンセル

図 4-31 コポーネント基本設定

【注】1. ユーザーは、[trash へのバックアップ数(1~20)] オプション(図 4-32 に示す)を設定することで、 バックアップのためにトラッシュフォルダに生成されるフォルダの数を制限できます。 制限を超 えると、最も古いフォルダが新しいフォルダに置き換わります。

バックアップの設定	
☑ バックアップ設定を有効にする	
trashへのバックアップ数 (1-20):	5

 初期化 API 関数のみを生成したい場合は、[API 関数出力:] リストボックスで「初期化関数のみ出 力する」に変更してください。".h"、".c"ファイルの voidR\_ {ConfigurationName} \_Create (void)、void R\_ {ConfigurationName} \_Create\_UserInit(void)のみが生成されます。 デフォル トのオプション設定「設定に合わせてすべて出力する」に変更するとすべての API 関数が再度生 成されます。

コード生成コンポ-	ネントの設定
API関数出力:	設定に合わせてすべて出力する ~
APIコードスタイル	設定に合わせてすべて出力する
	初期化関数のみ出力する

図 4-33 [API 関数出力:] の変更

図 4-32 バックアップ数の設定
### 4.5 端子設定

端子ページは、端子機能の割り当てに使用します。周辺機能別に端子機能を表示する[端子機能]リスト と、端子番号順に全ての端子を表示する[端子番号]リストの2つの表示があり、タブを切り替えることで 切り替えることができます。

• 2 000	71197	、 人力 (* = any si	tring, ? = any character)
~		機能 AOVREF AOVSS A1VREF A1VSS ADCA0I0 ADCA0I1 ADCA0I2 ADCA0I3 ADCA0I3 ADCA0I5	端子割り当て       設定されていません       設定されていません

図 4-34 端子ページ (端子機能)

[ボード]ページでボードを選択すると、[ボード機能]に初期端子設定情報が表示されます。 また、[機能]の選択リストに表示される[**W**]アイコンは、ボードの初期端子機能を示します。

									:811
Smart_Co	nfigurator_Example.scfg ×								
<b>耑子設</b> 分	È						⊐-×0	 の生成 レポー	トの生成
儲子番号									ill in d
フィルタ	入力 (* = any string, ? = any characte	r)	_					すべて	~
端子	端子名	ボード機能	機能		方向	備考	シンボリック名	コメント	^
L25	X2	X2	X2		0		# C		
K25	X1	X1	X1		1				
AK10	P1_1/RLIN312TX/CAN8RX/INTP8/I	TAUD001	設定されていません		なし				
AJ10	P1_0/RLIN312RX/INTP28/CAN8TX/.	TAUD000	設定されていません		なし				
C1	P11_0/MSPI6SC/TAUD0I0/PWGC49	TAUD0I0	TAUD0I0		1				
G18	P2_10/GTM117/GTMAT1O3/CAN5R	RLIN30TX	設定されていません		なし				
H6	P10_0/ICUMGPIO2/GTM1I0/GTMA.	RIICOSDA	RIICOSDA	~	10				
H7	P10_1/ICUMGPIO3/GTM0I2/GTMA.	RIICOSCL	FLXAORXDB	^	10				
F17	P2_13/GTM112/GTMAT1O1/GTM21.	MSPI0SO	<b>RLIN30TX</b>		0				
G17	P2_12/GTM0I4/GTMAT1O3N/GTM.	MSPIOSC	TSG3100		10				
J21	P2_3/GTM1I4/GTMAT1O5/GTM0I5/	MSPI0CSS0	TAUD118		なし				
AD9	AP0_0/ADCJ010	ADCJ010	CXP10TX		1		-		
AE18	P4_5/GTM1I0/GTMAT0O4/MSPI0D.	ADCJ0CNV0	RIICOSDA		0				
A1	VSS		PWGC550	-		読み取り	-		
A2	VSS		VSS		-	読み取り	-		

図 4-35 端子ページ (端子番号)

### 4.5.1 ソフトウェアコンポーネントの端子配置変更

スマート・コンフィグレータは、プロジェクトに追加されるソフトウェアコンポーネントに端子を配置し ます。端子の配置は端子ページで変更可能です。

このページでは、端子機能と端子番号のリストを表示します。

端子機能リストにあるソフトウェアコンポーネントの端子配置を変更するには、以下の手順で行います。

- (1) [ハードウェアリソース表示とソフトウェアコンポーネント表示の切り替え] 🚣 をクリックして、 ソフトウェアコンポーネントによって表示するように変更します。
- (2) ソフトウェアコンポーネントを選択します。(例: Config\_ICU)
- (3) [使用する] タブをクリックし、使用した端子でソートします。
- (4) 端子機能リストの端子割り当て欄で、端子配置を変更します。(例: P10\_0 から P0\_1)
- (5) 同じ周辺チャネルに属する1つの端子または複数の端子の配置は、一度[選択されたリソースの次の 端子割り当て先] ボタンをクリックするだけで変更することができます。

ハードウェアリソース	⊞ ⊟ 🗗 🚣	端子機能							20
フィルタ文字列を入力	(1)	端子機能名	を入力						
≛ すべて	^	使用する	機能	端子割り当て	(4)	端子番号	方向	備考	^
■ クロック発生回路		(3)	INTPO	P10_0/TAUD0I1/TAUE	001/CANORX/INTPO/CSCXFOU	Г A2	1		
			INTP1	P10_6/TAUD0I13/TAU	D0O13/CSIG0SO/ENCA0TIN0/A.	A9	1		
🔛 割り込み (二)			INTP2	P11_15/CAN2RX/INTE	2/CSIH2CSS4/PWGA55O/TAUB	I D5	1		
			INTP3	P1 2/CAN3RX/INTP3/	DPIN19/TAUJ2l2/TAUJ2O2	N1	1		

| 図 4−36 端子設定-[端子機能]リストの端子配置設定

スマート・コンフィグレータでは、ユーザーは他のソフトウェアコンポーネントにリンクすることなく、 端子ページで端子機能を有効にすることができます。それらの端子をソフトウェアコンポーネントが使用す る他の端子と区別するため、表の中に"この端子を使用するコンポーネントはない"という注意書きがつけら れます。

【注】 現在、ソフトウェアコンポーネント表示は未サポートです。端子配置変更はハードウェアリソース表 示から行ってください。



### 4.5.2 MCU/MPU パッケージの端子割り当て

スマート・コンフィグレータでは、 MCU パッケージビューでの端子設定を視覚化します。ユーザーは MCU/MPU パッケージビューをイメージファイルにキャプチャーでき、回転や拡大、縮小ができます。

MCU/MPU パッケージビューで端子を設定するには、以下の手順で行います。

- (1) [拡大] 🏓 ボタンをクリックるすか、マウスをスクロールして、ビュー内を拡大します。
- (2) 端子の上で右クリックします。
- (3) 割り当てを選択します。
- (4) [設定の変更...]で、端子の色をカスタマイズすることができます。





### 4.5.3 端子機能から端子番号の表示

端子機能に関連付けられた端子番号を表示できます。

端子機能から端子番号にジャンプするには以下の手順で行います。

- (1) [端子機能]タブで対象を右クリックし、ポップアップメニューを表示します。
- (2) [端子番号タブヘジャンプ]を選択します。
- (3) [端子番号]タブが開き、端子番号が選択された状態になります。

אין די		端子機能			
フィルタ文字列を入力		フィルタス	、力 (* = any strir	ng, ? = any character)	
📥 すべて	^	使用	機能	端子割り当て	端子番号
🇱 クロック発生回路			A0VCC	A0VCC	Y4
≴⊯ I/Oポート			A0V		/ Y3
🇱 割り込み			A0V 端子	番号タブにジャンプ	T5/T6/U4/\
🇱 System Control			A1V コメン	トを端子番号タブのコメントに追加	U1
🇱 ロウパワーサンプラ			A1V コメン	ト削除	〃 設定され
~ *欉 シリアルフラッシュメモリインタフェースA			A1V 端子	割り当て	W1/W2/Y1
SFMA0			A2V 端子	割り当て解除	U19
∽ ≇ll Multi Media Card Interface A			A2VREFH	〃 設定されていません	〃 設定されて
MMCA0			A2VSS	A2VSS	T16/U17/W
🗸 📲 Multichannel Serial Peripheral Inte	rfa 🔤	$\checkmark$	ADCJ0CNV0	P4_5/GTM1I0/GTMAT0O4/MSPI0DCS/MS	P / Y13
	. T			and an end of the end of the first state of the	- 20.000 V.L



<b></b> 子設定						
子番号						
端子番号	端子名	ボード機能	機能	方向	備考	コメント
W14	P4_8/GTM0I7/GTMAT0O3/RLIN30TX/EXTC		設定されていまt	なし		
W15	P4_10/GTM0I6/GTMAT1O1/EXTCLK0O/CA		設定されていませ	なし		
W16	P4_12/GTM1I1/GTMAT1O2/CAN5RX/INTP5		設定されていませ	なし		
W17	P4_14/GTM1I2/GTMAT0O2/CAN6TX/RLIN3		設定されていませ	なし		
W18	P3_4/GTM0I6/GTMAT0O2/RLIN37RX/INTP		設定されていませ	なし		
W19	P3_5/GTM0I3/GTMAT0O3N/CAN7RX/INTP		設定されていまt	なし		
W20	A2VSS		A2VSS	-	読み取り専用	
Y1	A1VSS		A1VSS	-	読み取り専用	
Y2	A0VSS		A0VSS	-	読み取り専用	
Y3	A0VREFH		A0VREFH	I		
Y4	A0VCC		A0VCC	-	読み取り専用	
Y5	AP0 1/ADCJ0I1		設定されていまた	なし		



### 4.5.4 端子設定のエクスポート

端子設定をエクスポートして、参照することができます。端子設定のエクスポートは、以下の手順で行います。

- (1) {ProjName}.scfg ファイルをセーブします.
- (2) 端子ページで、 [ボードの設定をエクスポート] ⊿ ボタンをクリックします。
- (3) 出力場所を選択し、エクスポートするファイル名を入力します。

XML フォーマットでエクスポートしたファイルは、同じデバイスの型名がある他のプロジェクトにイン ポートすることができます。

子								(2) 🖥 🖗
ハードウェアリソース 田 🖽	↓ª <sub>z</sub> 品	端子機能						<u>୧</u>
フィルタ文字列を入力		端子機能名	を入力					
▲ すべて 毎 クロック発生同路	^	使用する	機能	端子割り当て	端子番号	方向	備考	^
		$\checkmark$	TAUB1O1	P11_3/CSIH2SC/CAN3RX	B6	0		
			TAUB1O2	P11_12/RLIN25RX/PWG	E1	0		
1111日 111日 11日 11日 11日 11日 11日 11日 11日			TAUB1O3	P11_4/CSIH2SI/CAN3TX/	B5	0		
129F			TAUB1I0	設定されていません	設定されていません	なし		
🏝 電源			TAUB1I1	設定されていません	設定されていません	なし		
5 Uワハワーサンフラ	74		TAUB1I2	設定されていません	設定されていません	なし		
<ul> <li>▲ 電源</li> <li>▲ ロウパワーサンプラ</li> <li>▲ 外部メモリアクセスコントローラ</li> <li>▲ シリアリンフラッシュメモリインタフェースA</li> </ul>			TAUB1I3	設定されていません	設定されていません	なし		
> 🚣 シリアルフラッシュメモリインタフェースA			TAUB1I4	設定されていません	設定されていません	なし		
> 20日 クロック同期シリアルインタフェースG	~		TAUB115	設定されていません	設定されていません	なし		~
子機能 端子番号								

図 4-38 端子設定を XML ファイルヘエクスポートする

端子ページの [.csv ファイルにリストを保存] 📑 ボタンをクリックすることで、スマート・コンフィグ レータは CSV エクスポートをサポートすることができます。

#### 4.5.5 端子設定のインポート

現在のプロジェクトに端子設定をインポートするには、 [ボードの設定をインポート] 🔤 ボタンをク リックし、端子設定を含む XML ファイルを選択してください。設定がプロジェクトにインポートされる と、このファイルに指定された設定は、端子設定ページに反映されます。

設定								۱			
די ד	la 🖁	端子機能						<b>२</b> 🖪 🔤			
(ルタ文字列を入力		端子機能名	を入力								
📥 すべて	^	使用する	機能	端子割り当て	端子番号	方向	備考				
📕 クロック発生回路			TAUB1O1	P11_3/CSIH2SC/CAN3RX	B6	0					
\$₩ I/Oポート			TAUB1O2	P11_12/RLIN25RX/PWG	E1	0					
副 割り込み						TAUB1O3	P11_4/CSIH2SI/CAN3TX/	B5	0		
11 リセット						TAUB1I0	設定されていません	設定されていません	なし		
<ul> <li>● リセット</li> <li>▲ ロウパワーサンプラ</li> <li>▲ ロウパワーサンプラ</li> <li>▲ 外部メモリアクセスコントローラ</li> <li>&gt; ▲ シリアルフラッシュメモリインタフェースA</li> </ul>			TAUB1I1	設定されていません	設定されていません	なし					
			TAUB112	設定されていません	設定されていません	なし					
			TAUB1I3	設定されていません	設定されていません	なし					
			TAUB1I4	設定されていません	設定されていません	なし					
*************************************	~		TAUB115	設定されていません	設定されていません	なし					

図 4-39 端子設定を XML ファイルからインポートする

【注】 端子設定は反映されますが、コンポーネント設定には反映されません。



### RH850 スマート・コンフィグレータ

### 4.5.6 ボード端子設定情報を使用した端子設定

ルネサス製ボードの初期端子構成を設定できます。選択したボードは[ボード]タブで確認できます。 以下に端子を一括設定する手順を説明します。

- (1) [MCU/MPU パッケージ]の[ボード機能]を選択します。
- (2) [端子設定]ページを開き、[ボードの初期端子割り当ての設定]ボタンをクリックします。
- (3) [ボードの初期端子割り当て]ダイアログが開くので、[すべて選択]をクリックします。
- (4) [OK]をクリックします。



図 4-40 端子の初期設定

### 4.5.7 端子のフィルタ機能

[端子]ページの[端子設定]画面で、端子機能及び端子番号をフィルタリングすることが出来ます。

端子設定				<b>う</b> コードの生成 レポートの生成
ハードウェアリソース 🕀 🖻 🖡	z 🕹 🗴	端子機能		것 📕 🖬 🗠 🏼
フィルタ文字列を入力		フィルタ入力 (* = any st	ring, ? = any character)	すべて 🗸
▲ すべて ■ クロック発生回路	^	使用 機能	橋丁割り当( * 部中されていませく	<sup>痛</sup> 丁香 <sup>。</sup> 機能
↓ I/Oポート		A0VREF	<ul><li> </li></ul> <li> <ul><li> </li></ul></li>	✓ 設定さ端子割り当て ✓ 設定さ端子番号
■ 割り込み ■ リセット		A1VREF	<ul> <li>         ・設定されていません         ・         ・         ・</li></ul>	<ul> <li>✓ 設定さ方向</li> <li>✓ 設定さ備者</li> </ul>
■ 電源		ADCA0I0	AP0_0/ADCA0I0	/ 106 JXント
■ ロワバリーサンフラ *機 外部メモリアクヤスコントロー	-	ADCA011	AP0_1/ADCA0I1	/ 105

図 4-41 端子設定のフィルタリング

### 4.5.8 端子エラー/警告設定

[端子エラー/警告] 設定を使用して、[コンフィグレーションチェック] ビューのメッセージのエラーレベルを制御できます。

エラーレベルを変更は、[設定]ダイアログを開き、[スマート・コンフィグレータ]>[端子エラー/警告] ページで行います。

🚺 設定			×
	端子Iラ-/警告	⇔ ◄ ⇔	<b>▼</b> 8
<ul> <li>× スマート・コンフィグレータ MCU/MPUパッケージ コンボーネント 端子エラー/警告 &gt; ハルフ</li> </ul>	<ul> <li>端子競合         <ul> <li>つの端子番号に複数の端子機能が割り当てられている</li> <li>端子割り当て</li></ul></li></ul>	Error Error Error Info Warnir	~ ~ ~
	割り当てられた端子機能が、ボードの端子機能と一致していない	Warnir	ng ~
< >>	デフォルトの復元(T)	適用(l	_)
	適用して閉じる	キャンセル	þ

🗵 4-42 端子エラー/警告の設定

例えば、「ソフトウェアコンポーネント」の「割り当てられた端子機能を使用するソフトウェアコン ポーネントがない」の項目を「Info」から「Error」に変更すると、メッセージが図 4-43 のようになりま す。

🔝 コンフィグレーションチェック		
1 項目		
記述/説明	A	1
🔝 コンフィグレーションチェック		
1 error, 0 warnings, 0 other	rs	
記述/説明	^	큨
🗸 🥺 端子 (1 項目)		
I05000012: A0VC	Cを使用するコンポーネントのコンフィグレーションがありません。生成コード以外で端子の初期化を行う場合は、このメッセージを無視してください。	榨

図 4-43 端子エラー/警告の設定の変更例



## 4.6 割り込み設定

割り込みページは、 [コンポーネント] ページで設定した周辺機能の割り込みの確認および設定を行いま す。ベクタ番号別に割り込みが表示されます。割り込み優先レベル、OS 管理、割り込みハンドラ、エン ティティの生成、関数の生成の有効化/無効化などの共通設定を設定できます。 RH850/U2A の場合、PEn を設定して、PEn に割り込みを適用するかどうかを決定できます。。

しの設定								
育み割り込みべく	79							
マルタ文字列を,	入力							
ベクタ番号	例外要因コード	割り込み	割り込み要求元	周辺機能	優先レベル	状態	OS管理	
256	1100H	INTTAUB1I0	Interrupt for TAUB1 channel 0	TAUB1	最低	使用中		
257	1101H	INTTAUB111	Interrupt for TAUB1 channel 1	TAUB1	最低	使用中		
258	1102H	INTTAUB112	Interrupt for TAUB1 channel 2	TAUB1	景低	使用中		
259	1103H	INTTAUB113	Interrupt for TAUB1 channel 3	TAUB1	最低	使用中		

図 4-44 割り込みページ

#### 4.6.1 割り込み優先レベルと OS 管理の設定

コンポーネントページでのコンフィグレーションに割り込みを使用する場合、割り込みの状態は"使用 中"に変わります。使用中の割り込みだけを表示する場合は、[設定した割り込みのみ表示] 🐼 ボタンを クリックしてください。

- (1) 割り込みページで、割り込み優先レベルを変えることができます。
- (2) RTOS (RI850V4)を使用するプロジェクト時、OS管理カラムが有効になります。チェックをする と、割り込み関数がOS管理の割り込み書式で出力されます。

への設定								
育み割り込みべく	79							
マルタ文字列を	入力							
ベクタ番号	例外要因]-ド	割り込み	割り込み要求元	周辺機能	優先レベル	状態	OS管理	
256	1100H	INTTAUB110	Interrupt for TAUB1 channel 0	TAUB1	最低	使用中		(2)
257	1101H	INTTAUB111	Interrupt for TAUB1 channel 1	TAUB1	最低	使用中		(-)
258	1102H	INTTAUB112	Interrupt for TAUB1 channel 2	TAUB1	最低	使用中		
259	1103H	INTTAUB113	Interrupt for TAUB1 channel 3	TAUB1	最低	(使用中		
					レベル1 レベル2 レベル3 レベル4 レベル5 レベル5 レベル5 レベル5 レベル5 レベル5 レベル5 レベル5			

#### 図 4-45 割り込み設定



### 4.6.2 PEnの設定の変更(RH850/U2Aのみ)

RH850/U2A デバイスでは、[割り込み] ページで各割り込みに応答する PEn を選択することができます。 PEn は以下の手順で設定できます。

(1) [システム]ページで使用する PEn を選択します(4.3 システム設定(RH850/U2A のみ)を参照)。





- (2) [割り込み] ページの PEn 列のチェックボックスをオンまたはオフにして、割り込みに応答する PE を決定します。 割り込みには次の 2 種類があります。
  - (a) 各 PE の INTC1 に接続され、PEn 列で選択された各 PE が応答可能。
  - (b) 複数の PE で共有される INTC2 に接続され、PEn 列の1 だけが応答可能。

同り込みベクタ											
フィルタ文字列	を入力									ベクタ番	号
ベクタ番号	例外要因]-ド	割り込み	割り込み要求元	周辺機能	優先レベル	状態	OS管理	PE0	PE1	PE2	PE
26	101AH	INTTAUD0I14	TAUD0 Channel 14 interrupt	TAUD	最低			1	1		1
27	101BH	<b>INTTAPAOIPEKO</b>	TAPA0 peak interrupt 0	TAPA	最低			12	(V)	1	4
28	101CH	INTTAPA0IVLY0	TAPA0 valley interrupt 0	TAPA	最低			17	[27]	120	17
29	101DH	INTSDMACERR	sDMAC0 address error or sD	sDMAC	最低	使用中		0	10	V	V
30	101EH	INTDTSERR	DTS transfer error	DTS	最低	使用中	<u>iii</u>	V		10	6
21	10154	INITTRTMO	TRTM0 Timing Protection Vio	TDTM	要低		m .	100			
31	101FH	INTTPTM1	TPTM1 Timing Protection Vio	TPTM	最低				(W)		
31	101FH	INTTPTM2	TPTM2 Timing Protection Vio	TPTM	最低					127	
31	101FH	INTTPTM3	TPTM3 Timing Protection Vio	TPTM	最低						1
34	1022H	INTFLOENDNM	FPSYS0 Flash sequencer proc	FACI0	最低			0	0	0	0
36	1024H	INTFL1ENDNM	FPSYS1 Flash sequencer proc	FACI1	最低		E1	۲	0	0	0
38	1026H	INTFL2ENDNM	FPSYS2 Flash sequencer proc	FACI2	最低			(3)	0	0	0
39	1027H	INTDTS31TO0	DTS ch31-0 transfer end	DTS	最低	使用中		0	۲	0	0
40	1028H	INTDTS63T032	DTS ch63-32 transfer end	DTS	最低		E1	6	0	0	0
41	1029H	INTDTS95TO64	DTS ch95-64 transfer end	DTS	最低			0	0	0	0
42	102AH	INTDTS127TO96	DTS ch127-96 transfer end	DTS	最低			0	0	0	0
43	102BH	INTDTSCT31TO0	DTS ch31-0 transfer count m	DTS	最低	使用中		0	0	0	0
44	102CH	INTDTSCT63TO32	DTS ch63-32 transfer count	DTS	最低		E3	(0)	0	0	0

図 4-47 [割り込み]ページの PE 設定



#### 4.6.3 割り込み設定の変更

[割り込み]ページで、各割り込みハンドラの編集および、割り込みハンドラのエンティティを生成する かどうかを設定できます。

- (1) ユーザーは、コンポーネントで使用されていない割り込みハンドラ名を手動で入力できます。
- (2) [エンティティを生成する]にチェックを入れると、割り込みハンドラのエンティティを生成します。 デフォルトではチェックが入っています。 チェックを外すと割り込みハンドラコードが生成されな くなり、ユーザーが独自のハンドラコードを使用できるようになります。
- (3) [有効化/無効化関数を生成する]にチェックを入れると、割り込みを有効/無効にする関数を生成します。デフォルトではチェックは入っていません。

チェックを入れると、「r\_smc\_interrupt.c」ファイルに割り込み有効/無効の関数が生成されます。

ユーザーはこれらの API を直接呼び出すことで簡単に割り込みを使用できます。

割り込み有効/無効の関数のコードの例については、「図 4-49 割り込み有効/無効関数のコード例」を 参照してください。

マート・コンフィグ	レータ									_	
D1250 /	(ルプ										
*test.scfg ×											
り込み設定	Ê									つードの主成	レポートの
明込みベクタ											
									バクタ番号		
ベクタ番号	例外要因コード	割り込み	割り込み要求元	周辺機能	優先レベル	状態	OS管理	割り込みハンドラ	エンティティを生成	有効化/無効化機能を生成する	
558	122EH	INTDFE1FEND1	CH1 Filter Processing End Interrupt Request	DFE1	最低			eiint558	12	<b></b>	
559	122FH	INTDFE1FEND2	CH2 Filter Processing End Interrupt Request	DFE1	景低			eiint559	1971	8	
560	1230H	INTDFE1FEND3	CH3 Filter Processing End Interrupt Request	DFE1	量低			eiint560	E7.	10	
561	1231H	INTDFEFIFOOUTA	Buffer-A capture finished interrupt	DFE	最低			eiint561	12	10 I	
562	1232H	INTDFEFIFOOUTB	Buffer-B capture finished interrupt	DFE	最低			eiint562	121	10	
563	1233H	INTDFEFIFOERR	Error interrupt	DFE	量低			eiint563	12	10	
564	1234H	INTMSPIOTX0	MSPI0 Transmit status interrupt for channel 0	MSPIO	景低	使用中		r_Config_MSPI00_channel00_interrupt_send	1	(2)	
565	1235H	INTMSPIOTX1	MSPI0 Transmit status interrupt for channel 1	MSPIO	量低			eiint565	1	13	
566	1236H	INTMSPIOTX2	MSPI0 Transmit status interrupt for channel 2	MSPI0	景低			eiint566	1	10	
567	1237H	INTMSPIORXO	MSPI0 Receive status interrupt for channel 0	MSPIO	最低			eiint567	1	8	
568	1238H	INTMSPIORX1	MSPI0 Receive status interrupt for channel 1	MSPIO	量低			eiint568	1	13	
569	1239H	INTMSPIORX2	MSPI0 Receive status interrupt for channel 2	MSPI0	爱低			eiint569	12	8	
570	123AH	INTMSPI1TX0	MSPI1 Transmit status interrupt for channel 0	MSPI1	爱低			eiint570	192	8	
571	123BH	INTMSPI1TX1	MSPI1 Transmit status interrupt for channel 1	MSPI1	景低			eiint571	1	12	
572	123CH	INTMSPI1TX2	MSPI1 Transmit status interrupt for channel 2	MSPI1	最低			eiint572	120	8	
573	123DH	INTMSPI1RX0	MSPI1 Receive status interrupt for channel 0	MSPI1	量低			eiint573	621	12	
574	123EH	INTMSPI1RX1	MSPI1 Receive status interrupt for channel 1	MSPI1	最低			eiint574	2	10	
575	123FH	INTMSPI1RX2	MSPI1 Receive status interrupt for channel 2	MSPI1	景低			eiint575	192	8	
> 576	1240H	INTMSPIOTX/INTRC			景低	使用中		r_mspi0_interrupt_send	1971	23	
> 577	1241H	INTMSPIORX/INTRC			景低			eiint577	(F)	10	
> 578	1242H	INTMSPIOFE/INTRC			爱低	使用中		r_mspi0_interrupt_frameend	1	10	
> 579	1243H	INTMSPIOERR/INTR_			景低	使用中		r_mspi0_interrupt_error	191	E	
> 580	1244H	INTMSPI1TX/INTRC			量低			eiint580	192	12	
> 581	1245H	INTMSPI1RX/INTRC			爱低			eiint581	(2)	6	
> 582	1246H	INTMSPI1FE/INTRC			最低			eiint582	197	6	
> 583	1247H	INTMSPI1ERR/INTR			最低			eiint583	12	10	
> 584	1248H	INTMSPI2TX/INTRC			景低			eiint584	120	10	
> 585	1249H	INTMSPI2RX/INTRC			爱低			eiint585	(F)	8	
> 586	124AH	INTMSPI2FE/INTRC			婚纸			eiint586	121	8	

#### 図 4-48 割り込みハンドラとエンティティの設定







図 4-49 割り込み有効/無効関数のコード例

4.7 MCU マイグレーション機能

MCU マイグレーション機能は、異なるデバイス間でプロジェクト設定の移行を行います。プロジェクト 設定の変換は、同一ファミリ内で可能で、e<sup>2</sup> studioの[プロジェクト]メニューから以下の手順で行いま す。

- 【注】 デバイスの変更により、プロジェクトの設定が変わる場合があります。 デバイス変更を実行する前にプロジェクトのバックアップを行ってください。
  - (1) プロジェクトを選択し、[プロジェクト]メニューから [Change Device] を選択します。



workspace - Smart_Configu	urator_Example	e/Smart_Configurator_Example.s	cfg - e <sup>2</sup> studio					-	- 🗆 X
ファイル(F) 編集(E) ナビゲート(N	) 検索(A プ	ロジェクト(P) Renesas Views 実	行(R) ウィンドウ(W) ヘルブ	(H)					
📓 🛞 🕶 🔦 🕶 🥥 🕼	⊒ : † + <b>•</b>	ノロシェクトを開く(E)					c	🔪 🗄 🔛 🖓 🖓 🖓	スマート・コンフィグレータ
🔓 ЛОЎІР 🗙 🗖 🗖 🖡	*Smart_(	プロジェクトを閉じる(S)						パッケージ ×	- 0
		すべてビルド(A)	Ctrl+Alt+B			<b>1</b>			
🖌 😂 Smart Configurator	做記	ビルド構成	>			コードの生成 レポートの生	成 🛄 🚢		>>
> Ep includes	_ 188 (the imp)	プロジェクトのビルド(B)	Ctrl+B			٢	^		
🗸 🔁 src	▼ 030 RE 0403	リーキング・セットのビルト(W)	>			Ø			
Y 🗁 smc_gen	$\sim$	り シン(N) 自動的にビルド(M)							
> 🗁 general	W	(パー++ インデックフリ)							
> 🗁 r_pincfg	a <sup>2</sup>	オペアの依存関係を更新	Alt+D						
> S crtart arm		Change Device		Appli	cation Code				
> indefine h		Change Toolchain Version		Арри		2 V		RENESAS	0000
Smart_Configurate	1	C/C++ Project Settings	Ctrl+Alt+P	Softwar	e Components				
Smart_Configurate		プロパティ(P)			Middleware &				
		最新情報			Drivers	Q T			
	S	<u>最新情報</u> をクリックすると、最新し	リースの情報を確認すること	RTOS				DUDS/TE1 VM/S4	
		ができます。			Device Drivers	E S		R7F701653	
		すべての <u>リリースノート</u> を表示				ato			
	a	製品ドキュメント		MCL	l Hardware	-			
		ユーザーガイド							
		ΑΡΙマニュアル							
		<u>アプリケーションノート</u>							
		ツールニュース					V		
< >	概要 ホート 9	ロック コンホーネント  瑞子   割り込	<b>み</b>				► 7°L199		
ערב 🖳 א-עעב 📮			🖹 🚺 🛃 🖬	- 📑 - 🗖	🔝 コンフィグレーションチェッ	ל X			78-0
スマート・コンフィグレータ出力					0項目	^			
M05000001: 端子 P20 に A M05000002: 端子番号 P20	DCA0I0 の機能 の ADCA0I0	能が割り当てられています 設定が解除されました		^	記述/説明		型		
M05000001: 端子 C20 に A	DCA110 の機能	能が割り当てられています							
<				>					
						6	Smart_Configurator	Example	81
		: 🦡 m 🕿 🖉 🚳							
4									

図 4-50 [Change Device] 選択

 (2) デバイス選択リストから対象のデバイスを手動で選択し、「OK」をクリックします。 (ワイルド カード検索対応) (例: RH850 - C1M 252pin 部品番号: R7F701275 に変更)。

๑๑ มวะชรมวช	- 0	×	8			_		×
Change Device Select the new device for Smart_Configurator_Example	3		Device Selection You can filter devices by regular ex	pression				
Current Device: R7F701653			Search Device					
Target Board:         Custom           ターゲット・デバイス:         R7F701653	<u>71(12079</u> 1		Device > RH850 - C1H > RH850 - C1M - A1 > RH850 - C1M-A1 > RH850 - C1M-A2 > RH850 - C1M-A2 > RH850 - D1M-A2 > RH850 - D1L1 > RH850 - D1L2 > RH850 - D1M1 > RH850 - D1M1 > RH850 - D1M2 > RH850 - D1M2 > RH850 - E1M-S2 > RH850 - F1H > RH850 - F1H > RH850 - F1H > RH850 - F1H > RH850 - F1H	RAM 192 KB	ROM 4.0386 MB	Pin 252		
⑦ < 戻る(6) 次へ(N) > 終了(	(F) キャンセ	IL	> RH850 - F1KM > RH850 - F1L			ок	キャン1	~ 2Л

図 4-51 ターゲット・デバイス選択

(3) [検出された問題] に表示されたメッセージを確認して [次へ] をクリックします。

<ul> <li>๑๑ มาวะ</li> <li>๑๑ มาวะ</li> </ul>			×
Change Device 下のリストに示されている情報を確認してください。 '次へ >' をクリックして次の項目を表示するか ックします。	、'終了' をク	<sub>7</sub> ,	
検出された問題			\$ {}
A This change cannot be undone. Please make sure you backup this project before of	ontinuing.		
使用可能なコンテキスト情報はありません			
(ア)         (N) >         終了((N) >	F)	キャンセ	ll.

図 4-52 検出された問題

メッセージ	説明
ターゲット・デバイスはスマート・コン フィグレータでサポートされていません	スマート・コンフィグレータがサポートしていない デバイスへの変更時に表示されます。スマート・コ ンフィグレータの変換は実行できませんが、プロ ジェクト、ビルダー、リンカー、デバッカーは変換 できます。
This change cannot be undone. Please make sure you backup this project before continuing.	デバイスを変更すると変更前に戻すことができませ んので、プロジェクトのバックアップ後に実行して ください。

#### (4) [実行される変更] で、変更する項目を選択してマイグレーションを実行します。

e <sup>2</sup> リファクタリング	—		×
Change Device リファクタリングを実行するには、以下の変更が必要です。			
実行される変更 V 図		<b>₽</b> 0	+0+ +10+
<ul> <li>✓ ▲ Smart_Configurator_Example HardwareDebug</li> <li>✓ ▲ Build Settings</li> <li>✓ ▲ Project Files</li> <li>✓ ▲ Smart Configurator</li> </ul>			
使用可能なブレビューはありません			
? < 戻る(B) 次∧(N) > 終了(F)		キャンセ	zll

図 4-53 実行される変更

(5) デバイスの変更が完了すると、「概要」ページのデバイス名が更新されます。

Smart_Cont	figurator_Example.scfg $ imes$		- 0
概説			🚺 値 コードの生成 レポートの生成
	動画 スマート・コンフィグレータの紹介 関連動画 最新情報 たクリックすると、最新リリースの情報を確認すること ができます。 すべての <u>リリースノート</u> を表示	Application Code Software Components Middleware & Drivers Drivers Device Drivers	^ Configurate
	<b>製品ドキュメント</b> ユ <u>ーザーガイド</u> <u>APIマニュアル</u> アプリケーションノート ツールニュース	MCU Hardware	
<ul> <li>▼ 現在の設定</li> <li>使用しているが</li> <li>生成先ロケー</li> </ul>	E状態 ポード/デパイス: R7F701275 (ROM size: 4 MB , RAM size: 128 K ション (PROJECT_LOC¥): src¥smc_gen	B, Pin count: 252) 編集	
使用している	コンポーネント: ロック コンポーネント 端子 割り込み		~

### 図 4-54 デバイス更新確認

(6) コンソールにデバイス変更結果レポートが出力されます。

א-עעב 🖳 🛪 🖬 🖬 🖬 🖛 🖬 🗸 און א א א א א א א א א א א א א א א א א א	- 0	J
スマート・コンフィグレータ出力		
TICS00002. mill 田で L2 0 07 DECATA の総約が使用していたっていた。	-	~
NOSOBODO1: 端子 USV に X1 の機能が割り当ていています MOSOBOD01: 端子 USV に X1 の機能が割り当ていています	- 1	
10500001 - 端子 44/ に 23 の機能性度的光子をわています		
M00000003: レポート生成:C:\Users\cg\e2 studio\workspace\Smart Configurator Example\output\migration report 2024-01-11-09-24-17.html	1	
	-	-
	>	

図 4-55 設定変換ステータスレポート

【注】 デバイス変更結果レポートが正しく表示されない場合は、右メニューより「エンコード」の設定を "Unicode(UTF-8)"に変更してください。



### 5. 競合の管理

コンポーネントの追加、端子や割り込みの設定をすると、リソースの不一致や失われた依存関係にあるモ ジュールに関連する問題が起こる可能性があります。この情報はコンフィグレーションチェックビューに表 示されます。表示された情報を参照して、競合問題を解決してください。

### 5.1 リソースの競合

同じリソース(例:TAUB1)を使うために、二つのソフトウェアのコンフィグレーションを設定した場合、コンポーネントツリーにエラーマーク 🖪 が表示されます。

コンフィグレーションチェックビューに周辺機能の競合に関するメッセージが表示され、ユーザーに周辺 機能に競合が見つかったソフトウェア設定を知らせます。

🏶 Smart_Configurator_Example.scfg 🛛					- 8
ソフトウェアコンポーネント設定					🐻 🖨
	設定				^
20日日 10日 21川タ入力	クロック設定 動作クロック クロックソース	CK0 ~ PCLK/32768 ~ (実際	eの値:0.12207kHz)		
✓      ↔      ØAR	PWMマスターセレクト設定 マスターチャンネル選択	0 ~			1
	PWMスレーブ設定 □チャネル1スレーブ □チャネル4スレーブ □チャネル4スレーブ	□ チャネル2スレーブ □ チャネル5スレーブ □ チャネル5スレーブ	□チャネル3スレ □チャネル6スレ □チャネル6スレ	-ブ -ブ -ブ	
	□ チャネル10スレーブ □ チャネル13スレーブ <	□ チャネル11スレ−ブ □ チャネル14スレ−ブ	□ Fャネル12ス □ Fャネル15ス	レーブ レーブ	>
概要   ボード   クロック   コンポーネント  端子   割り込み					
■ コンフィグレーションチェック 窓					~ - 8
10 errors, 0 warnings, 0 others 記述/説明	^			タイプ	
	_Duplicateで設定)と次の割り込みの で設定)と次の割り込みのベクタが鏡 _Duplicateで設定)と次の割り込みの で設定)と次の割り込みのヘクタが鏡	Dベクタが競合しています : INTTAUB1IO (C 含しています : INTTAUB1IO (Config_TAU Dベクタが競合しています : INTTAUB1I1 (C 合しています : INTTAUB1I1 (Config_TAU	onfig_TAUB1で設定)。 B1_Duplicateで設定)。 onfig_TAUB1で設定)。 B1_Duplicateで設定)。	割り込み 割り込み 割り込み 割り込み	
<ul> <li>● E04010001:周辺機能 TAUB10 (Config_1)</li> <li>● E04010001:周辺機能 TAUB10 (Config_1)</li> <li>● E04010001:周辺機能 TAUB11 (Config_1)</li> <li>● E04010001:周辺機能 TAUB11 (Config_1)</li> <li>● E04010001:周辺機能 TAUB11 (Config_1)</li> <li>● © 端子 (2項目)</li> </ul>	AUB1_Duplicateで設定) は次の設定 (AUB1で設定) は次の設定で使用され (AUB1_Duplicateで設定) は次の設定 (AUB1_Duplicateで設定) は次の設定 (AUB1で設定) は次の設定で使用され	をで使用されています: Config_TAUB1. いています: Config_TAUB1_Duplicate. をで使用されています: Config_TAUB1. いています: Config_TAUB1_Duplicate.	<ol> <li>TALIDA 750年)</li> </ol>	周辺機能 周辺機能 周辺機能 周辺機能	
<ul> <li>E04010003: IAUB1O1 (Config_TAUB1_D</li> <li>E04010003: TAUB1O1 (Config_TAUB1C)</li> </ul>	Duplicateで設定)が使用する端子と2 設定)が使用する端子と次の端子が算	火の喃子が競合しています:IAUB1O1 (Co 競合しています:TAUB1O1 (Config_TAUB	onfig_IAUB1で設定). 81_Duplicateで設定).	) 师子 端子	

図 5-1 リソースの競合



### 5.2 端子の競合

端子の競合がある場合、エラーマーク 🔇 がツリーと端子機能リストに表示されます。

-ドウェアリソース 🗉 🖻	↓42 品 端子機	能				2 📓 🔤
パルタ文字列を入力	端子板	機能名を入力				
🏂 すべて	▲ 使用	する 機能	建之則147	洋고풍묘	士向	<del>使</del> 来
100/100/1000 回路		TAUB101	P11_3/CSIH2SC/CAN3RX	B6	0	複数の端子機能が同一端子に割り当てられています
(副) 利り込み						
🧱 リセット						
▲ 電源						
2010/10/00/00/00/00/00/00/00/00/00/00/00/						
> *# クロック同期シリアルインタフェースG						
> * # クロック同期シリアルインタフェースH						
NE LIN/UARI1 J9JI-X						
A CANFD12971-2						
- FlexRay						
📥 Ethernet AVB						
> 🛓 Single Edge Nibble Transmission						
> 💑 91マ・アレイ・ユニット BO						
TAURIO						
TAUB11						
TAUBI2						
TAUB13						
TAUB14						
TAUB15						
TAUB16 TAUB17						
TAUDIO						

図 5-2 端子の競合

競合情報の詳細は、コンフィグレーションチェックビューに表示されます。

3/27/1/2/23/25/2010	⊉ ▽ □ □
2 errors, 1 warning, 0 others	
記述/説明	タイプ
✓ ⑧ 端子 (3 項目)	
😣 E04010003: TAUB1O1 (Config_TAUB1で設定) が使用する端子と次の端子が競合しています : P11_3 (Pin Allocatorで設定).	端子
🚱 E05000010: 端子 B6を複数の機能で使用できません。端子 B6 に P11_3, TAUB1O1の機能が割り当てられています。	端子

図 5-3 端子競合のメッセージ

エラーマークのあるツリーノードを右クリックし、 [競合の解決]を選択して競合を解決してください。



選択されたノードの端子は、他の端子に再度割り当てられます。



### 6. ソースの生成

6.1 生成ソースの出力

スマート・コンフィグレータビューの [コードの生成] 🐻 ボタンをクリックすると、設定した内容に応じたソースファイルを出力します。

③ *Smart_Configura ソフトウェアコンス	ator_Example.scfg ☆ ポーネント設定			□ □-ドの生成 レポートの生成
コンポーネント		設定	í	
フィルタ入力	te te	ブロパティ ✔ 拳 Configurations	値	^



スマート・コンフィグレータは、<PeojectDir>¥src¥smc\_gen にファイルを生成し、プロジェクト・エク スプローラー内のソースファイルを更新します。すでにスマート・コンフィグレータでファイルを生成して いる場合、バックアップも生成します。(「8 生成ソースのバックアップ」を参照ください。)

【注】ユーザー作成のソースファイルを sms\_gen フォルダに置くと、ソースコード生成時に消去されます。

🚹 プロジェクト・エクスプローラー 🗙		E (	<b>\$</b> 7	000	
✓	mple [Hardware	Deb	ug]		
> 🗊 Includes					
🗸 📴 src					
∽ 🗁 smc_gen					
✓					
> 🗟 Config_ADCA	\0_user.c				
> 🗟 Config_ADCA	\0.c				
> h Config_ADCA	\0.h				
Y 🗁 general					
> 🖻 r_cg_ad_com	mon.c				
> h r_cg_ad_com	mon.h				
> 庙 r_cg_ad.h					
> 🖻 r_cg_cgc_use	r.c				
> 💽 r_cg_cgc.c					
> 🔓 r_cg_cgc.h					
> 🚺 r_cg_intc_PE1	.с				
> cg_intvecto	r_PE1.c				
> h r_cg_macrod	river.h				
> c r_cg_main.c					
> C r_cg_systemir	hit.c				
> h r_cg_userdefi	ne.h				
> h r_smc_entry.h	l .				
> 🔝 r_smc_interru	pt.c				
> h r_smc_interru	pt.h				
> C r_smc_intprg.	с				
> 🔁 r_pinctg					
> S boot.asm					
> S cstart_pm1.asm					
> In Iodefine.n					
> b output					
🖉 🗁 trasn	vample offe				
Smart_Configurator_E	vample.scig	oDel	bug law	ch	
Smart_Configurator_E	kample Hardwar	eve	oug.iaur	ich	

図 6-2 プロジェクト・エクスプローラー内のソースファイル



## 6.2 ソース生成先の設定

ソースの生成先は、スマート・コンフィグレータの[概要]ページから設定できます。

(1) [概要]ページの[現在の設定状態]の下にある[編集]ボタンをクリックします。

Smart_Co	nfigurator_Example.scfg $\times$	
既説		🕲 🤖 コードの生成 レポートの生成
▼ 機能概要		٢
	<b>概要</b> <u>概要</u> をクリックすると、スマート・コンフィグレータの 機能を確認することができます。	
	<b>動画</b> <u>スマート・コンフィグレータの紹介</u> 関連動画	Application Code
	<b>最新情報</b> <u>最新情報</u> をクリックすると、最新リリースの情報を 確認することができます。 すべての <u>リリースノート</u> を表示	RTOS Middleware & Configuration
ļ	<b>製品ドキュメント</b> ユ <u>ーザーガイド</u> <u>APIマニュアル</u> アプリケーションノート ツールニュース	MCU Hardware
・現在の設	定状態	
使用している 生成先ロケ- 使用している	5ボード/デバイス: R7F701275 (ROM size: 4 MB , RAM si ーション (PROJECT_LOC¥): src¥smc_gen 5コンポーネント:	ze: 128 KB, Pin count: 252)
要 ポード	クロック コンポーネント 端子 割り込み	

図 6-3 ソース生成先の変更

(2) [フォルダの選択] ダイアログで、生成先のフォルダを選択し、 [OK] ボタンをクリックします。

🔯 フォルダの選択		×
コードの生成先フォルダを選択してください:		
🔯 新規フォルダ		$\times$
フォルダ名: New_Folder	 	
OK	キャンセル	,
😢 選択したフォルダが空ではありません		
新規フォル/ダ OK	キャンセル	ŀ

図 6-4 フォルダ選択

(3) [コード生成 🚺 ] ボタンをクリックすると、(2)で選択したフォルダにソースファイルが生成され ます。 [概要] ページで現在の生成コードの場所を確認することもできます。



図 6-5 プロジェクト・エクスプローラー内のソースファイル



6.3 生成ファイルの構成とファイル名

スマート・コンフィグレータが出力するフォルダとファイルを下図に示します。なお、main()関数は e<sup>2</sup> studio でプロジェクト作成時に生成する {*Project name*}.c に含まれます。

"ConfigName"はコンポーネント設定で設定したコンフィグレーション名を示します。





フォルダ	ファイル	説明
general		このフォルダは常時生成されます。同じ周辺機能の CG ドライバ
		で共通に使用される、ヘッダファイルとソースファイルを含みま
		す。
	$r_cg_xxx.h^{(*l)}$	これらのファイルは常時生成されます。SFR レジスタを設定する
		ためのマクロ定義を含みます。
	r_cg_cgc.c	このファイルは常時生成されます。
		クロックページの設定を基にしたクロックソースの初期化を含み
		ます。
	r_cs_cgc.h	このファイルは常時生成されます。
		このヘッダファイルは、クロックを初期化するマクロ定義を含み
		ます。
	r_cs_cgc_user.c	このファイルは、CGC 初期化後にユーザーがコードを
		R_CGC_Create に追加する関数を含みます。
		ユーザーは、コードと関数を専用のユーザーコード領域に追加す
		ることができます。
	r_cg_intvector.c	このファイルは常時生成されます。割り込みベクタ・テーブルの
	r_cg_intvector_PEn.c	定義を含みます。
	r_cg_macrodriver.h	このファイルは常時生成されます。
		このヘッダファイルは、ドライバで使用される共通のマクロ定義
		を含みます。
	r_cg_main.c	このファイルは常時生成されます。main()関数を定義します。
	<i>r_cg_</i> systeminit. <i>c</i>	このファイルは常時生成されます。
		R_ConfigName_Createの名前を持つ全てのドライバの初期化関数を
		呼び出す <b>R_Systeminit</b> を含みます。 <b>R_Systeminit</b> は、クロックの初
		期化も呼び出します。
	r_cg_userdefine.h	このファイルは常時生成されます。
		ユーザーは、専用のユーザーコード領域にマクロ定義を追加する
		ことができます。
	r_smc_interrupt.c	このファイルは常時生成されます。
	r_smc_interrupt.h	このファイルは常時生成されます。
		[割り込み] ページで設定されたすべての割り込みの優先レベルの
		定義が含まれます。 ユーザーはこれらのマクロ定義をアプリケー
		ション・コードで使用できます。
	r_smc_entry.h	このファイルは常時生成されます。
		このファイルは以下のファイルをインクルードします。
		r_cg_xxx_common.h
		r_cg_macrodriver.h
		r_cg_userdefine.h
		r_cg_cgc.h
		{ConfigName}.h
		このファイルは、r_cg_main.c にインクルードされます。



フォルダ	ファイル	説明
	r_cg_intc_PEn.c	このファイルは以下に対してのみ生成されます。
		RH850/U2A :
		PE PEn(n=0~3) の使用が選択された場合のみ
		r_cg_intc_PEn.c(n=0~3)が生成されます。
		RH850/C1M:
		r_cg_intc_PEI.c が生成されます。
		DH820/112D ·
		r cg into PFO c が生成されます
		このファイルは割り込み初期化 API 定義を含みます。
	$r_cg_xxx_common.c^{(*l)}$	このファイルは、すべてのコンポーネントで共有される共通設定 を持つコンポーネントに対してのみ生成されます。
		通常、複数の共有 API が含まれており、ユーザーによって呼び出 されます。
	r_cg_xxx_common.h <sup>(*1)</sup>	r_cg_xxx_common.c および r_cg_xxx_common_user.c のヘッダファ イルです。
		このファイルは、すべてのコンポーネントで共有される共通設定 を持つコンポーネントに対してのみ生成されます。
	r ca xxx common user c <sup>(*1)</sup>	通常、後数の共有APIの亘台が占まれより。 このファイルは すべてのコンポーネントで共有されろ共通設定
	T_cg_xxx_common_user.c	を持つコンポーネントに対してのみ生成されます。
		通常、複数の共有割り込みサービスルーチンが含まれています。
		ユーザーは、専用のユーザーコード領域にコードと関数を追加で
		きます。
	r_smc_clock_info.h	RH850/U2B 専用に生成されます。
		クロックソースのマクロ定義と、[Clock]ページのモジュールク
		ロック設定が含まれています。
		このファイルをインクルードすることで、クロック設定マクロを 毎日オスニトができます
r pipefa	Pin c	このファイルけ堂時生成されます
1_pineig	1 111.0	「端子」タブで設定される全周辺機能に使用する端子機能初期化
		のリファレンスです(I/Oポート以外)。
	Pin.h	このファイルは常時生成されます。
		このファイルは、
		Pin.c での端子設定の関数プロトタイプ
		シンボル名の定義
		シンボル名ユーザーガイド
		シンボル名 API
		を含みます。
{ConfigName}		このフォルタは、フロジェクトに追加されるコンホーネント用に 生成されます。
		このフォルダの API 関数は、ConfigName(設定名)の後に命名し
		to the state of th
	{ConfigName}.c	このファイルは、ドライバ( <i>R_ConfigName_Create</i> )を初期化する関 数 ドライバに特有な操作 例えばスタート( <i>R_ConfigName_Start</i> )
		やストップ( $R_ConfigName_Stop$ )を実行する関数を含みます。
	{ConfigName} user.c	ドライバの初期化(R_ConfigName Create)の後に追加することがで
		きる割り込みサービスルーチンと関数を含みます。
		ユーザーは、専用のユーザーコード領域にコードと関数を追加す
		ることができます。
	{ConfigName}.h	{ConfigName}.c と {ConfigName}_user.cのヘッダファイルです。

\*1:xxx は周辺機能名を意味します。



# 6.4 クロック設定

クロックページにあるクロックソースの設定は、¥src¥smc\_gen¥general フォルダに生成されます。



図 6-7 メインクロックをクロックソースとして選択した場合のクロック設定

No	フォルダ	ファイル	マクロ/関数	説明			
general		r_cg_cgc.c	R_CGC_Create	この API 関数は、クロックを初期化します。			
				r_cg_systeminit.c の R_Systeminit は、main()関数 から、この関数を呼び出します。			
		r_cg_cgc.h	クロックに関連するマク ロ	これらのマクロは、R_CGC_Createのクロックの初期化に使用されます。			
		r_cg_cgc_user.c	R_CGC_Create_UserInit	<b>CGC</b> 初期化後に、ユーザーが <i>R_CGC_Create</i> に コードを追加する際にこの API 関数を使用しま す。			



## 6.5 端子設定

端子設定は、コンポーネントにより下記(1)から(2)に示すようなソースファイルに生成されます。

(1) {ConfigName}を使用したドライバの端子初期化

端子機能は¥src¥smc\_gen¥{ConfigName}¥{ConfigName}.cのR\_ConfigName\_Create で初期化されます。 端子初期化コードは、main()で処理されます。\_

フトウェアコンポーネ 🕀 🖻 🖧 品	端子機能				- 3   E	<b>  </b>   2-1
フィルタ文字列を入力	フィルタ入	.力 (* = any str	ing, ? = any character)		すべて	,
✓ ▲ A/Dコンバータ	使用	機能	端子割り当て	端子	番号	方向
Config_ADCA0	$\checkmark$	TAUB1O1	P11_3/CSIH2SC/CAN3RX/INTP3/PWGA28C	/ 16	3	0
🗸 🚣 CSI スレーブ	$\checkmark$	TAUB1O2	P11_12/RLIN25RX/PWGA52O/TAUB1I2/TA	/ 12		0
Config_CSIG0	$\checkmark$	TAUB1O3	P11_4/CSIH2SI/CAN3TX/INTP21/PWGA290	/ 16	4	0
✓ 益 データ CRC		TAUB1O4	〃 設定されていません	〃設	定されてい	なし
Config_DCRA0		TAUB1O5	✓ 設定されていません	〃設	定されてい	なし
🗸 🚣 入力期間カウント検出		TAUB1O6	〃 設定されていません	〃設	定されてい	なし
💣 Config_TAUB0_0		TAUB107	✓ 設定されていません	〃設	定されてい	なし
∽ 🊣 PWM出力		TAUB1O8	✓ 設定されていません	〃設	定されてい	なし
Config_TAUB1		TAUB1O9	✓ 設定されていません	〃設	定されてい	なし
		TAUB1O10	✓ 設定されていません	〃設	定されてい	なし
		TAUB1011	✓ 設定されていません	≠設	定されてい	なし
		TAUB1O12	✓ 設定されていません	〃設	定されてい	なし
		TAUB1O13	✓ 設定されていません	≠設	定されてい	なし
		TAUB1O14	✓ 設定されていません	〃設	定されてい	なし
		TAUB1O15	✓ 設定されていません	〃設	定されてい	なし
	<					

図 6-8 Config\_TAUB1 の端子設定

フォルダ	ファイル	関数	説明
{ConfigName}	{ConfigName}.c	<i>R_ConfigName_Create</i>	このドライバが使用した端子を API 関数が初期 化します。 <i>main()</i> 関数から、 <i>r_cg_systeminit.c</i> の <i>R_Systeminit</i> はこの関数を呼び出します。

(2) 端子初期化コードの参照

プロジェクトで使用されるすべての周辺端子機能については、¥src¥smc\_gen¥r\_pincfg フォルダにある Pin.c を参照してください(I/O ポート以外)。

フォルダ	ファイル	関数	説明
r_pincfg	Pin.c	<i>R_Pins_Create</i>	[端子] ページで設定された、I/O ポート 以外の全端子機能の初期化コードを含みま す。



## 6.6 割り込み設定

割り込みページの設定は、いくつかのソースファイルに生成されます。

RH85	RH850/C1M, F1KM, F1KH and U2B:																	
ペクタ番号 29	例外要因コード 101D	割り込み要求元 DMA transfer erro	or (DMAERR)	周辺機能 DMA	優先レベル 最低	状態 使用中	OS管理	割り込みハン r_dmac_err	パラ or_interru	pt_pe1	エンティティを生成する <sup>2]</sup>	有効化/無効1	と機能を生成する					
					(1)		(2)			(3)			(4)					
RH85	0/U2A:																	
ベクタ番号 29	例外要因コード 101DH	割り込み INTSDMACERR	割り込み要求元 sDMAC0 addre	ss error or s	DMAC1 addre	ess erro	周辺機能 sDMAC	優先レベル 最低	状態 使用中	OS管理	割り込みハンドラ r_sdmac_address_	error_interrupt	エンティティを生成 ジ	有効化/無効化機能を生成する □	PE0	PE1	PE2	PE3
								(1)		(2)	)	(3)		(4)			(5)	
							义	6-9	割	り込	み設定							

### RH850/F1KM, F1KH and U2B:

No	項目	フォルダ	ファイル	説明
(1)	優先レベル	{ConfigName}	{ConfigName}.c	割り込み優先レベル設定は、
				このファイルの R_ConfigName_Create で初期化さ
				れます。この関数は、main()関数の中から
				r_cg_systeminit.c の R_Systeminit によって呼ばれま
				す。
(2)	OS 管理	{ConfigName}	{ConfigName}_user.c	このファイルで定義されている割り込み関数を OS
		or	or	管理の割り込み書式で出力します。
		general	r_cg_xxx_common_user.c	
(3)	割り込みハン	general	r_smc_intprg.c	[エンティティを生成する]にチェックを入れる
	ドラ		r_cg_intvector_PE0.c	と、[割り込みハンドラ]に表示される割り込みハ
	エンティティ			ンドラのエンティティが「r_smc_intprg.c」に生成
	を生成する			されます。
(4)	有効化/無効	general	r_smc_interrupt.c	[有効化/無効化関数を生成する]にチェックを入れ
	化関数を生成		r_smc_interrupt.h	ると、有効化/無効化関数が r_smc_interrupt.c に生
	する			成されます。

### RH850/C1M, U2A:

No	項目	フォルダ	ファイル	説明		
(1)	優先レベル	general	r_cg_intc_PEn.c	割り込み優先レベル設定は、		
				このファイルの R_Interrupt_Initialize_ForPE()関数		
				で設定されます。この関数は、main()関数の中か		
				ら r_cg_systeminit.c の R_Systeminit()関数によって		
				呼ばれます。		
(2)	OS 管理	{ConfigName}	{ConfigName}_user.c	このファイルで定義されている割り込み関数を OS		
		または	または	管理の割り込み書式で出力します。		
		general	r_cg_xxx_common_user.c			
(3)	割り込みハ	general	r_smc_intprg.c	[エンティティを生成する]にチェックを入れる		
	ンドラ		r_cg_intvector_PE0.c	と、[割り込みハンドラ]に表示される割り込みハ		
	エンティ			ンドラのエンティティが「r_smc_intprg.c」に生成		
	ティを生成			されます。		
	する					
(4)	有効化/無効	general	r_smc_interrupt.c	[有効化/無効化関数を生成する]にチェックを入れ		
	化関数を生		r_smc_interrupt.h	ると、有効化/無効化関数が r_smc_interrupt.c に生		
	成する			成されます。		
(5)	PEn	general	r_cg_intc_PEn.c	PE 選択の設定は、このファイルの		
	(RH850/U2A			<b>R_Interrupt_Initialize_ForPE</b> で初期化されます。こ		
	のみ)			の関数は、main()関数の中から r_cg_systeminit.c の		
				R_Systeminit()関数によって呼ばれます。		



## 7. ユーザープログラムの作成

スマート・コンフィグレータのコンポーネントタイプには[コード生成]があります。ここでは、 [コード生成]のカスタムコード追加方法について説明します。

## 7.1 コード生成の場合のカスタムコード追加方法

コンポーネントのタイプで[コード生成]を選択した場合、ソースコード出力の際、同一ファイルが存在 する場合には、以下のコメントで囲まれた部分に限り、該当ファイルをマージします。

/\* Start user code for xxxx. Do not edit comment generated here \*/

/\* End user code. Do not edit comment generated here \*/

図 7-1 コード追加用のコメント

[コード生成]の場合、特定の周辺機能ごとに3つのファイルを生成します。デフォルトのファイル名 は、「Config\_xxx.h」、「Config\_xxx.c」、「Config\_xxx\_user.c」となり、xxx は周辺機能を表します。(例 えば、A/D コンバータ(リソース ADCA0)の場合、xxx は "ADCA0"になります。)カスタムコードを追加 するためのコメントは、3つのファイルそれぞれの先頭と最後に設けられる他、「Config\_xxx\_user.c」にあ る周辺機能の割り込み関数内にも追加されます。以下に ADCA0 の例 (Config\_ADCA0\_user.c)を示しま す。



図 7-2 生成コードの例



7.2 ユーザーアプリケーションコードの使用

生成されたコードを使用するには、以下の手順で行います。

r\_cg\_main.c ファイルを開き、main 関数で生成された関数を呼び出します。

コード生成の場合、端子初期化を含むドライバ初期化関数(R\_ConfigName\_Create)は、デフォルトで r\_cg\_hardware\_setup.cの R\_Systeminit 関数で呼び出されます。

ドライバ固有の処理を実行するには、アプリケーションコードを追加する必要があります。 例えば、開始(R\_ConfigName\_Start)および停止(R\_ConfigName\_Stop)。



図 7-3 コード生成関数コール

8. 生成ソースのバックアップ

スマート・コンフィグレータには、ソースコードのバックアップ機能があります。

[コードの生成] 🔂 ボタンをクリックしてコードの生成を行うとき、スマート・コンフィグレータは生成 ソースのバックアップを生成します。<Date-and-Time>は、コード生成を実行しバックアップフォルダを作 成した日時です。

<ProjectDir>¥trash¥<Date-and-Time>



### 9. レポートの生成

スマート・コンフィグレータは、設定情報をレポートで提供します。レポートを生成するには、以下の手 順で行います。

### 9.1 全設定内容レポート

スマート・コンフィグレータビューで [レポートの生成] ៉ ボタンをクリックし、レポートを出力します。出力形式は 、PDF とテキストの 2 種類が選択可能です。



図 9-1 全設定内容レポート出力(txt 形式)

(雪) レポート生成		_		×
設定内容のレポートを出力します				
◉ 全ての項目を出力する				
○ 選択した項目を出力する				
> ☑ コンポーネント				
> 🔳 端子				
□ 割り込み				
- L ☑ PDFで出力			<u>フォント運</u>	択
□テキストで出力				
C:¥Report			参照.	
?	OK		キャンセル	,

#### 図 9-2 レポート出力ダイアログ



9.2 端子機能リスト、端子番号リスト設定内容(csv 形式)

スマート・コンフィグレータビューの [端子] ページで [.csv ファイルにリストを保存] 📊 ボタンをク リックし、端子機能リスト、端子番号リスト設定内容を出力します。



9.3 MCU/MPU パッケージ図(png 形式)

MCU/MPU パッケージビューの[端子配置図を保存] 🎆 ボタンをクリックし、MCU/MPU パッケージ図を出力します。



図 9-4 MCU/MPU パッケージ図出力(png 形式)



10. ユーザーコード保護機能

図 10-1 の指定タグを追加することで、任意の位置にユーザーコードを追加できます。追加されたユー ザーコードはコード生成時に保護されます。

ユーザーコード保護機能は「コード生成コンポーネント」が生成したファイル、クロック生成ファイル、 割り込み生成ファイルのみサポート対象となります。

10.1 ユーザーコード保護機能の指定タグ

ユーザーコード保護機能を使用する場合、図 10-1 のように、/\* Start user code \*/ と /\* End user code \*/ を挿入し、このタグの間にユーザーコードを追加してください。指定タグが完全に一致しない場合は、保護 されません。

> /\* Start user code \*/ コメントの間にユーザーコードを追加 /\* End user code \*/ 図 10-1 ユーザーコード保護機能の指定タグ

10.2 ユーザーコード保護機能の使用例

図 10-2 に示すように、図 10-1 の指定タグを使用し、PWM 出力モジュールの Create 関数の中に新しい ユーザーコードを挿入します。その後、PWM 出力の GUI での設定を更新し、再びコード生成すると、挿入 されたユーザーコードが新たに生成されたファイルに自動的にマージされます。



R20AN0739JC0100 Rev.1.00 Jan.20.2024



10.3 競合発生時の対応方法

10.3.1 競合の発生条件

挿入したユーザーコードの前後にある生成コードに変更がある場合(GUIの設定変更、スマート・コン フィグレータのバージョンアップなど)、図 10-3 のように生成コードに競合が発生します。



### 図 10-3 生成された競合コード

競合が発生した場合、コンソールに図 10-4 のようなメッセージが表示されます。



#### 図 10-4 競合のメッセージ



### 10.3.2 競合の解決方法

競合を解決するには、競合が発生したファイルを開いて、下記の手順に従って手動でコードを修正してく ださい。

- 1) コンソールで競合しているファイルをクリックし、[File Compare]ビューを開きます。(図 10-5)
- 2) 「左から右へ現在の変更をコピー」をクリックします。(図 10-5)
- 3) 未使用のコードを削除します。(図 10-6)
- 4) 変更後のコードを保存します。(図 10-7)
- 【注】左側パネルのコードを右側パネルにコピーするか、右側パネルのコードを直接編集することで、競 合を手動で解決することもできます

競合が解決された後も、競合メッセージをクリックすると、[File Compare]ビューを開くことができます。



図 10-5 生成された競合コード



🔯 workspace - File Compare - e <sup>2</sup> studio									
ファイル(F) 編集(E) ナビゲート(N) 検索(A) プロジェクト(P) Renesas Views 実行(R) ウィンドウ(W) ヘルプ(H)									
· 🔛 ) ७ र 🗞 र   ८४ / । 🔍 । 🕸 र 🗛 र 🔝 ୩ ) 🗉 💩 💁 🔍 🔍									
🖉 🛞 Smart_Configurator_Example.scfg 🕼 Config_TAUB0_0.c 🔓 Hile Compare X									
🔓 🕂 C 比較				ā					
✓ ▶ 翻訳単位				8					
R_Config_TAUB0_0_Create				2					
↓ C 比較ビューアー 8		M 📰 🛱 😥 🗞 🏇	9 BA						
Existing code	٠	New code							
<pre>54 54 54 54 54 54 554 554 554 554 554 5</pre>		<pre>Sa Sa void R_Config_TAUB0_0_Create(void) S5{ Sa void R_Config_TAUB0_0_Create(void) S5{ Sa void R_Config_TAUB0_0_Create(void) Sa void R_Config_TAUB0_0_Counter_STOP; Sa void R_Config_TAUB0_10 operation and clear request */ Sa void R_CONFIG_CONTER_STOP; Sa void R_CONFIG_CONTER_STOP; Sa void R_CONFIG_CONTER_STOP; Sa void R_CONFIG_CONTER_STOP; Sa void R_CONFIG_CONTER_STOP; Sa void R_CONFIG_CONFIG_CONFIG_CONFIG_COUR; Sa void R_CONFIG_CONFIG_CONFIG_CONFIG_COUR; Sa void R_CONFIG_CONF</pre>							
				-					
NO00000002: コード生成の終了:       1-ビ生加応開始         N00000002: コード生加応開始       1-ビ生加応開始         N00000002: コード生加応開始       1-ビ生加応開始         N04000001: プーバル生成::src\smc_gen\Config TAUB0 0\Config TAUB0 0.h         N04000002: プーバル生成::src\smc_gen\Config TAUB0 0\Config TAUB0 0.c         N00000005: 赤色でハイライトされている上記のファイルには、ユーザコードのマージが競合しています。ファイルを開き、手動で競合を解決してください         N000000062: コード生成の終了:       Cillersignation         V00000002: コード生成の終了:       Studio\workspace\Smart Configurator Example\src\smc_gen									
< 左: 65: 1、 左: 65: 1、 鼓台 変更 #1 (左: 65: 68、右: 65: 69)									

### 図 10-6 「左から右へ現在の変更をコピー」後のコード



図 10-7 競合を解決した後のコード



# 11. ヘルプ

## 11.1 ヘルプ

スマート・コンフィグレータの詳細情報は、e<sup>2</sup> studio メニュー上のヘルプを参照ください。

図 11-1 ヘルプ表示

# 「概要」ページの⑦アイコンクリックにより、ヘルプを参照できます。



図 11-2 クイックスタート



12. 参考ドキュメント

ユーザーズマニュアル:ハードウェア

最新版をルネサスエレクトロニクスホームページから入手してください。

テクニカルアップデート/テクニカルニュース

最新の情報をルネサスエレクトロニクスホームページから入手してください。

ユーザーズマニュアル:開発環境

統合開発環境 e2 studio ユーザーズマニュアル 入門ガイド (R20UT2858)

CC-RH コンパイラ ユーザーズマニュアル (R20UT3516)

スマート・コンフィグレータ ユーザーズマニュアル RH850 API リファレンス編 (R20UT4361) (最新版をルネサスエレクトロニクスホームページから入手してください。)



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://www.renesas.com

## お問合せ先 <u>http://www.renesas.com/contact/</u>

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。


改訂記録

		改訂内容		
Rev.	発行日	ページ		ポイント
1.00	2024.01.20	—	新規作成	

# 製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテク ニカルアップデートを参照してください。

1. 静電気対策

CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。

2. 電源投入時の処置

電源投入時は、製品の状態は不定です。電源投入時には、LSIの内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部 リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオン リセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。

3. 電源オフ時における入力信号

当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入に より、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」について の記載のある製品は、その内容を守ってください。

4. 未使用端子の処理

未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。

5. クロックについて

リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した 後に切り替えてください。リセット時、外部発振子(または外部発振回路)を用いたクロックで動作を開始するシステムでは、クロックが十分安定 した後、リセットを解除してください。また、プログラムの途中で外部発振子(または外部発振回路)を用いたクロックに切り替える場合は、切り 替え先のクロックが十分安定してから切り替えてください。

#### 6. 入力端子の印加波形

入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V<sub>L</sub>(Max.)から V<sub>IH</sub>(Min.)までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V<sub>L</sub>(Max.)から V<sub>I</sub>(Min.)までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。

7. リザーブアドレス(予約領域)のアクセス禁止

リザーブアドレス(予約領域)のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス(予約領 域)があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。

8. 製品間の相違について

型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合が あります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

#### ご注意書き

- 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。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、お客様の責任において、お客様の機器・システムを設計ください。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。
- 当社製品または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. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要と なる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
- 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。
- 6. 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。

- 7. あらゆる半導体製品は、外部攻撃からの安全性を100%保証されているわけではありません。当社ハードウェア/ソフトウェア製品にはセキュリ ティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害(当社製品または当社製品が使用されてい るシステムに対する不正アクセス・不正使用を含みますが、これに限りません。)から生じる責任を負うものではありません。当社は、当社製品ま たは当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行 為(「脆弱性問題」といいます。)によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害に ついて、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア/ソフトウェア製品について、商品 性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
- 8. 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。
- 9. 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。
- 10. 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。
- 11. 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。
- 12. 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。
- 13. 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。
- 14. 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.5.0-1 2020.10)

### 本社所在地

〒135-0061 東京都江東区豊洲 3-2-24(豊洲フォレシア) www.renesas.com

# お問合せ窓口

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

## 商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の 商標です。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。