

RL78 スマート・コンフィグレータ

R20AN0579JJ0104 Rev.1.04 2025.04.21

ユーザーガイド: e² studio 編

要旨

本アプリケーションノートでは、e² studio のプラグインツールである RL78 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 e² studio の対象バージョンは以下の通りです。

• e² studio 2025-04 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。 <u>https://www.renesas.com/rl78-smart-configurator</u>

目次

1	概要	5
11	目的	5
1.2	日	5
1.3	ソフトウェア・コンポーネント	5
2.	プロジェクトの作成	6
3.	スマート・コンフィグレータレータの操作方法	10
3.1	スマート・コンフィグレータの表示	.10
3.2	操作手順	.11
3.3	プロジェクト情報のの保存先	12
3.4	ウィンドウ	12
3.4.1	プロジェクト・エクスプローラー	.13
3.4.2	! スマート・コンフィグレータビュー	13
3.4.3	3 MCU/MPU パッケージビュー	.14
3.4.4	コンソールビュー	15
3.4.5	。 コンフィグレーションチェックビュー	15
3.4.6	Developer Assist Browser	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	3 ボード設定のエクスポート	19
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	22
4.4	コンポーネント設定	.24
4.4.1	コンポーネント・ビューとハードウェア・ビューの切り替え	.24
4.4.2	2 コード生成コンポーネントの追加方法	25
4.4.3	3 ソフトウェア・コンポーネントの削除	27
4.4.4	└── CG ドライバの設定	28
4.4.5	う CG コンフィグレーションのリソース変更	29
4.4.6	う SNOOZE モード・シーケンサの設定	31
4.4.7	′ SMS データファイルの更新	.34
4.4.8	3 ELCL 固定機能モジュールのダウンロード	35
4.4.9) 固定機能 ELCL コンポーネントの設定	36
4.4.1	0 ELCL Flexible Circuit の作成と編集	37
4.4.1	1 RL78 Software Integration System モジュールのダウンロード	42
4.4.1	2 RL78 Software Integration System モジュールの追加	43
4.4.1	3 RL78 Software Integration System モジュールの設定の設定	.44
4.4.1	4 BSP コンフィグレーションのバージョン変更	45
4.4.1	5 コンポーネントのコンフィグレーションのエクスポート	47
4.4.1	6 コンポーネントのコンフィグレーションのインポート	47
4.4.1	7 コンポーネントの基本設定	48



4.5	端子設定	51
4.5.1	PIOR 機能による端子割り当ての変更	52
4.5.2	ソフトウェア・コンポーネントの端子配置変更	53
4.5.3	MCU/MPU パッケージを使用した端子の設定	54
4.5.4	端子機能から端子番号の表示	55
4.5.5	端子設定のエクスポート	56
4.5.6	端子設定のインポート	56
4.5.7	ボード端子設定情報を使用した端子設定	57
4.5.8	端子のフィルタ機能	57
4.5.9	端子エラー/警告の設定	58
4.6	割り込み設定	59
4.6.1	割り込み優先レベルの設定	59
4.6.2	割り込みバンクの設定	60
4.7	MCU マイグレーション機能	61
5.	競合の管理	64
5.1	リソースの競合	64
5.2	端子の競合	64
c		66
0. 6 1		60
0.1 6.2	生成ノースの山力	67
0.Z	コートエル场別の友史 生成ファイルの構成とファイルタ	60
6.J	<u> </u>	72
6.5	ッロック設定	73
6.6	割り込み設定	74
0.0		
7.	ユーザープログラムの生成	75
7.1	コード生成のカスタムコード追加方法	75
7.2	ユーザーアプリケーションコードの使用方法	77
-		
8.	生成ソースのバックアップ	78
a	しポートの生成	70
0. 0.1	ク 空 内 の 上 ポ ー ト (PDE またけ tvt 形式)	70
9.1		80
9.2 Q 3	「加」版化リスト、加」由ラリストの設定内容	80
0.0		00
10.	Developer Assistance	81
11.	ユーザーコード保護機能	82
11.1	ユーザーコード保護機能の指定タグ	82
11.2	ユーザーコード保護機能の使用例	82
11.3	競合発生時の対応方法	83
11.3.	1 競合の発生条件	83
11.3.	2 競合の解決方法	84
10		00
12.	ヘルノ	бŊ



13. ‡	参考ドキュメント8	7
改訂	2録8	8



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 e² 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。

e² studio の使い方は、e² 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。SW 統合機能を使用したミドルウェアのインポート、ドライバコード生成、端子設定の3 つ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。

1.3 ソフトウェア・コンポーネント

スマート・コンフィグレータは、3種類のソフトウェア・コンポーネント(コード生成(CG)、 Graphical Configurator と RL78 Software Integration System)に対応します。それぞれのソフトウェアが対応するドライバとミドルウェアは、以下の通りです。

- <u>コード生成(DTC、A/D コンバータ、割り込みコントローラなど)</u> CG ドライバは、DTC、AD コンバータ、割り込みコントローラなどのマイコン周辺機能の制御プロ グラムです。コード生成機能を使用したソフトウェア・コンポーネントの組み込みが便利です。
- グラフィカル・コンフィグレータ(SMS、ELCL) グラフィカル・コンフィグレータ・モジュールは、他のドライバ設定に比べてグラフィカルな GUI を 提供することで、複雑な構成の設定を容易にします。SNOOZE モード・シーケンサ(SMS)とロ ジック&イベント・リンク・コントローラ(ELCL)のソフトウェア・コンポーネントを提供してい ます。
- <u>RL78 Software Integration System(静電容量センサユニット(CTSU2L)など)</u>
 RL78 Software Integration System モジュールは、ドライバ、ミドルウェア、アプリケーション SW の ソフトウェア・コンポーネントで、コードを生成するための簡単な GUI を提供します。



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た C/C++プロジェクトの生成手順を、以下に説明します。 e² studio のプロジェクト作成ウィザードの詳細は、e² studio の関連ドキュメントを参照してください。

 e² 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++ Project] - [Renesas RL78] 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。

C	📓 workspace - e² studio										
ファイ	′ル(F) 編算	Ę(E)	ソース(S)	リファクタリング(T)	ナビゲート(N)	検索(4	A) プロジェクト(P)	Renesas Views	実行(R)	ウィンドウ(W)	ヘルプ(H)
	新規(N)				Alt+シフト+N >		Renesas C/C++ Pr	oject		>	Renesas Debug
ファイルを開く(.)			Makefile Project with Existing Code			Renesas RL78					
È,	ファイル・シス	ステムカ)らプロジェ	クトを開く		Ċ	C/C++ Project				Renesas RX

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 CC-RL C Executable Project] または [LLVM for Renesas RL78 C/C++ Executable Project] を選択し、[次へ] ボタンをクリックします。

📴 New C/C++ Project					\times		
Templates for New O	emplates for New C/C++ Project						
All CMake Make	LLVM for Renesas A C/C++ Executable for Renesas RL78 Toc	RL78 C/C++ Executabl Project for Renesas RL78 o	e Project		^		
Renesas Debug Renesas RL78	LLVM for Renesas A C/C++ Library Proj Renesas RL78 Toolch	RL78 C/C++ Library Pr ect for Renesas RL78 using ain.	r oject g LLVM for				
•	Renesas CC-RL C/C	C++ Executable Project Project for Renesas RL78 0	t using the CC-R	PL toolchail	n.		
•	Renesas CC-RL C/C	C++ Library Project ect for Renesas RL78 usin	g the CC-RL to	olchain.	<		
?	< Back	lext > Finis	h	Cancel			

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-RL executable project, プロジェクト名: "Smart_Configurator_Example")

0		_		×
New Renesas CC-RL Ex New Renesas CC-RL Exe	xecutable Project cutable Project			\$
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:¥e2_studio¥workspace¥Smart_Configurator_	Example	参照(R)	
	Create Directory for Project			
ファイル・システムを選択(Y)	デフォルト ~			
ワーキング・セット				
ワーキング・セットにプロ	リジェクトを追加(T)		新規(W)	
ワーキング・セット(ロ):		\sim	選択(E)	
?	< 戻る(B) 次へ(N) > 終	了(F)	キャンセ	μ

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

(4) ツールチェーン、デバイス、デバッグ設定を選択します。[次へ] をクリックします。
 (例:ターゲット・デバイス:RL78/G23 - 128pin(型名:R7F100GSNxFB))

8						×
New Renesas CC-RL Select toolchain, devic	Executable Proje	ect				Ŷ
Toolchain Settings 言語: ツールチェーン: ツールチェーン・パージョン:	C O C++ Renesas CC-RL v1.12.00		~			
Device Settings Target Board: Cu ターゲット・デバイス: R7	stom F100GSNxFB	ソールナエーンの官ま Download additional board	¥里 ▼ rds	Configurations Hardware E2 Lite (R	Debug 構成 L78)	【を生成 ~
エンディアン: Litt プロジェクト・タイプ: デフ	le オルト	<u> デバイスのアンロッ</u>	<u>ック</u> ~	Debug 稱 RL78 Sim Release 權	成を生成 ulator i成を生成	~
?	< Back	Next >		Finish	Cane	cel

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、[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を選び、[終了] をクリックします。

3			×
New Renesas CC-RL Executable Project Select Coding Assistant settings			*
✓ Use Smart Configurator Use 周辺コート生成 ● 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 RL78 SIS コンフィグレータの機能を併せ持ち、異 バーやらドルウェアをインボート、設定、ユ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影 画面、ビ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 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。)	なるタイプ 安定画面、 り込み、1	のデバイス 割り込み ごン設定の	ドライ 設定 ハード
Application Code Software Components Middleware & Drivers Device Drivers MCU Hardware			
? < 戻る(B) 次へ(N) > 終了(F)		キャンセ	IL

図 2-5 コーディング・アシストツールの選択

- 【注】 (4)のデバイス設定で、スマート・コンフィグレータが対応しているデバイス選択時のみ、 [スマート・コンフィグレータを使用する]チェックボックスが選択可能になります。
- (6) [プロジェクトテンプレートの選択] ダイアログで、[Bare Metal Minimal] または [Bare Metal Blinky]
 を選択し、[終了] をクリックします。

6	— D X	
New Renesas C プロジェクト テンプ	C-RL Executable Project レートを選択します。	
プロジェクトテンプレ	ノートの選択	
•	Bare Metal - Minimal BSP を含むベアメタル プロジェクト。このプロジェクトは、 クロック、ピン、ドライバー、および C ランタイム環境を初期化し ます。	
0 🔯	Bare Metal - Blinky BSP を含むペアメタル プロジェクト。利用可能な場合は LED が点滅します。 このプロジェクトは、クロック、ピン、ドライ パー、および C ランタイム環境を初期化します。	
?	< 戻る(B) 次へ(N) > 終了(F) キャンセル]



【注】 [Bare Metal - Blinky] は、(4) の [Target Board] で「Custom」以外の LED リソースを持つ ボードが選択された場合に表示されます。「Custom」ボード選択時は、[Bare Metal -Minimal] のみが表示されます。



(7) プロジェクト作成の完了を待ちます。

進行情報	
ス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 2-7 プロジェクト作成の処理

(8) 新規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。



図 2-8 [スマート・コンフィグレータ] パースペクティブ



RL78 スマート・コンフィグレータ

3. スマート・コンフィグレータレータの操作方法

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、e² 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。

國 パースペクティブを開く	-		×
IIIII C/C++ (デフォルト) よ Java の型階層 、Java の型階層 Scripting Scripting コード生成 ポスマート・コンフィグレータ ビーチーム同期化 茶デバッグ IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII			
開<(c))	キャンセ	رال

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

e² 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。e² studio の操作については、e² 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、e² 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。



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
- (6) Developer Assist Browser



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。

৳ プロジェクト・エクスプロ−ラ− 🗙	E	\$ 7	000		
✓ 😂 Smart_Configurator_Exam	ple				
> 🗊 Includes					
🗸 📴 src					
✓ 🔁 smc_gen					
> 📂 general					
> 🗁 r_bsp					
> 📂 r_config					
> 10 Smart_Configurator	Example.c				
Smart_Configurator_Ex	ample.scfg				
📄 Smart_Configurator_Ex	ample Hard	wareDebu	.ig.la	unch	n I
⑦ Developer Assistance					

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、e² 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6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。

Smart_Confi	gurator_Example.scfg \times			-
既説			で コードの生成 レ7	/── ぱートの生♬
機能概要				?
	概要 <u>概要</u> をクリックすると、スマート・コンフィグレー ができます。	タの機能を確認すること		
	動画 <u>スマート・コンフィグレータの紹介</u> 開まままで		Application Code	
	国法判正 最新情報 最新情報 ます。 ます。 まず、 の リリースノート を表示	青報を確認することができ	Software Components Middleware & Drivers Device Drivers	
	現在のパージョン: VI.13.0 製品ドキュメントとよくあるご質問 ユーザーガイド <u>APIでニュアル</u> アプリケーションノート ツールニース	Ĵ(FAQ)	MCU Hardware	
現在の設定 E:コード生成	FAQ: <u>Smart Configurator</u> 状態 の動作は、 <u>設定</u> ページでカスタマイズできまま	ŧ. ()		
使用しているボ・ + ポキロケーシ	ード/デバイス: R7F100GSNxFB (ROM size:	768KB, RAM size: 48KB, I	in count: 128)	
更用しているコ)	ンポーネント:			
	^	バージョン	設定	
コンポーネント		170	Config ADC(ADC: 使用中)	
コンポーネント ○ A/Dコンバ	-9	1.7.0	conng_no c(no c) b(n n)	
コンポーネント ② A/Dコンバ ③ Board Su	ータ pport Packages v1.90 (r_bsp)	1.90	r_bsp(使用中)	

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種類を切り替えることができます。

- [割り当てられた機能]は、端子設定の割り当て状態を表示します。
- [ボード機能]は、ボードの初期設定情報を表示します。
 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 ボード設定、4.5.7 ボード端子設定情報を使用した端子設定を参照ください。)
- [シンボリック名]は、ユーザーが端子に定義したシンボル名を表示します。シンボリック名のマクロ 定義は、Pin.h ファイルにポート読み出しまたは書き込みの関数とともに生成します。



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、e² studio メニュー上の [Renesas Views] - [スマート・コンフィグレータ] - [MCU 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。

ערב 🖳	🕒 🗟 🔐 🖻 🚽 🔂 🕶 🖻	
スマート・コンフィグレータ出力		
M05000001: 端子 3 (こ PCLBUZ0 の機能が割り当てられています M05000001: 端子 117 (こ ANIO の機能が割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、e² 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

コンフィグレーションチェックビューには、ドライバが使用する割り込み、周辺機器、端子設定で競合が 発生した場合の詳細が表示されます。

🔝 Configuration Problems 🛛		*	∇	
8 errors, 0 warnings, 0 others				
	タイプ			
> 😢 割り込み (2 項目)				
> 🔕 周辺機能 (2 項目)				
> 🔇 端子 (2 項目)				
> 🔞 設定 (2 項目)				

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、e² studio メニュー上の [Renesas Views] - [スマート・コンフィグレータ] - [コンフィグレーション・チェック] を選択してください。



3.4.6 Developer Assist Browser

[Developer Assist Browser] ビューは、スマート・コンフィグレータ開発者支援機能のビューサービスで す。API 情報をナビゲートおよび参照し、[Copy] コンテキスト・メニューを使用して、コード生成コンポー ネント API 使用例のコードスニペットを C/C++エディタに貼り付けることができます。

このビューが開かれていない場合、e² studio メニュー上の [Renesas Views] - [スマート・コンフィグレー タ] - [Developer Assist Browser] を選択してください。

-	🗊 MCU/MPUパッケージ 👩 Develope	r Assist Browser $ imes$			
1	General				^
	Below is a list of API functions of	output by the Sm	art Configurator for common use.		1
	API Function Name	Peripheral Name	Description		
	main	-	Main function.		
	R_Systeminit	-	Executes initialization processing that is required before controlling various peripheral modules.	re	
	R_DTC_Set_PowerOn	Data	Starts the clock supply for DTC.		
	R_DTC_Set_PowerOff	Controller	Stops the clock supply for DTC.		
	R_TAUm_Create		Executes initialization processing that is required before controlling TAU <i>m</i> (enables TAU <i>m</i> input clock supply and initializes TAU <i>m</i> module).	re	
	R_TAUm_Set_PowerOn	Timer Arrav	Starts the clock supply for TAUm.		
	R_TAUm_Set_PowerOff	Unit	Stops the clock supply for TAUm.		
	R_TAUm_Set_Reset		Sets TAU <i>m</i> module in reset state.		
	R_TAUm_Release_Reset		Releases TAUm module from reset state.		
	R_ITL_Create		Executes initialization processing that is required before controlling the 32-bits IT (enables input clock supply and initializes ITL <i>m</i> module).	re	~

図 3-9 Developer Assist Browser ビュー



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。

🔅 *Smart_Co	nfigurator_Example.scfg $ imes$		- 8
デバイス邊	劉 沢	じ ⊐−ドの生成	 レポートの生成
デバイス選択	5		è 2
ボード:	カスタムユーザボード 🗸 🛄		
デバイス:	R7F100GSNxFB		
	<u>小ートI音報をグリノロートする</u>		
概要 ボード /	フロック システム 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

[] をクリックすると、リストから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。ルネサス製ボード(ファストプロトタイピングボード等)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 e² studio プロジェクトのターゲット・ デバイスに反映されます。詳細は 4.7MCU マイグレーション機能を参照してください。

🔅 Smart_Configurator_Example.scfg 🗙		- 8	図 リファクタリング	— 🗆	×
テバイス選択	℃ コードの生成		Change Device Select the new device for Smart_Configurator_Example		
デバイス選択		2 2	Current Device: R7F100GSNxFB Current Board: RL78G23-128p_FastPrototypingBoard_LED		
ボード: カスタムユーザボード	~		Target Board: RL78G23-128p_FastPrototypingBoard	additional boa	v ards
デバイス: R7F100GSNxFB ボード情報をダウンロードする			ターゲット・デバイス: R7F100GSN×FB	<u>デバイスのアン</u> [<u>コック</u>
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		(ア) < 戻る(B) 次へ(N) > 終了(F)	キャンセ	zll

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			\times
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; をクリックします。	か、'終了'	40072	E
検出された問題			00
(d) This change cannot be undone. Please make sure you backup this project before c	ontinuing	g.	
使用可能なコンテキスト情報はありません			
マヘ(N) > 終了(F)		キャンセル	/

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.	プロジェクトのバックアップ後に実行してください。



[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。

			×
Change Device The following changes to 4 files are necessary to perform the refactoring.			
実行される変更 ◆ ② 会 Change Device for SC_example ◆ ② 会 Launch Configurations ② 会 SC_example HardwareDebug > ② 会 Build Settings ② 会 Project Files ② 会 Smart Configurator		设 ①	ए •
使用可能なプレビューはありません			
? < 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

ボード設定後に今後の参考としてボード設定のエクスポートができます。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 ×		- 8
デバイス達	劉 沢	じ コードの生成	 レポートの生成
デバイス選択	ł		
ボード:	カスタムユーザボード 🗸 📈		
デバイス:	R7F100GSNxFB		
	ボード情報をダウンロードする		
			~
概要ボード!	7ロック システム コンポーネント 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。

🌼 *Smart_Configurator_Example.scfg 🗙	- 8
デバイス選択	😇 値 コードの生成 レポートの生成
デバイス選択	(1) (1) (1) (1) (1) (1) (1) (1)
(2) ボード: カスタムユーザボード 〜	
デバイス: R7F100GSNxFB	
<u>ホート情報をダワンロートする</u>	
	¥

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他のプロジェクトでも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) 動作モードと EVDD 設定を指定します。
- (2) デバイス操作に必要なクロックを選択します(デフォルトは、高速オンチップ・オシレータが選択されています)。
- (3) ボードの仕様に従って各クロックの周波数を指定します(一部の内部クロックでは周波数が固定され ていいますので注意してください)
- (4) マルチプレクサ・シンボルで、出力クロックのためのクロック・ソースを選択します。



図 4-7 [クロック] ページ

4.3 システム設定

[システム] ページでは、オンチップ・デバッグを設定できます。スマート・コンフィグレータは、コード 生成時に [システム] ページの設定に応じて、リンカ・オプションを設定します。リンカ・オプションの設定 は、プロジェクト・プロパティの [C/C++ ビルド] - [設定] - [Linker] - [デバイス] から確認できます。



図 4-8 e² studio リンカ・オプション画面(デフォルト)

スマート・コンフィグレータの [システム] ページで、以下のように設定します。

ステム設定				(4)	こードの生成レポートの当
2					
オンチップ・デバッグ設定					
オンチップ・デバッグ動作設定 〇 使用しない	(1)	◉ エミュレータを使う			() сомポ−ト
Iミュレータ設定 ○ E20		○ E2	(2)	\square	• E2 Lite
疑似RRM/DMM機能設定 〇 使用しない		◉ 使用する			
Start/Stop関数機能設定 ○使用しない	3)	◉ 使用する			
 通過ポイント機能設定 ● 使用しない 		○使用する			
トレース機能設定 ○使用しない		◉ 使用する			
セキュリティID設定 □ セキュリティIDを設定する					
セキュリティID		0x000000000000000000000000000000000000			
セキュリティID認証失敗時の設定 〇 フラッシュ・メモリのデータを消去しない ④ フラッシュ・メモリのデータを消去する					

図 4-9 スマート・コンフィグレータ [システム] ページ設定



[システム] タブで、上図の (1) から (3) の設定を行なってから (4) の [コード生成] ボタンをクリックする と、e²studio のリンカ・オプション更新を確認する下図のダイアログが表示されます。

e	コンパイラ変更の確認			Х
	設定項目	古い値 PFF00 PFFFF	新しい値	
	デバック・モージ領域の範囲 デバッグ・モニタ領域を設定する オンチップ・デバッグ・オプション・バイト制	false 04	true 84	
	٢		>	
		ОК	キャンセル	

図 4-10 リンカ・オプション確認ダイアログ

[OK] ボタンをクリックし、リンカ・オプション設定を確認すると、以下のように更新されます。

・・・ SON Compilation Datab Con	figuration: HardwareDeb			
タック解析 リールチェイン・エディター ルド変数 1000000000000000000000000000000000000	ツール設定 Toolchain Devi	ug [アッティン] ce] ※ ビルド・ステップ 🥮 ビルド成果物 🔛 バイナリー・パーサー 😝 I	Eラー・パーサー	~ 構成の管注
 時度 時度 ⇒ ⇒ ⇒ ⇒ ⇒ ⇒ ⇒ ⇒ ⇒ ⇒ ⇒ ⇒ ⇒	 SMS Assembler Common Compiler Compiler Assembler Linker リスト 漫道北 リスト 漫道化 セクション デバイス 出力 ゴボー ビbrary Generator 漫成 アド/(ソスト コーザー Cometer コーザー Cometer ロボーマット その他 コーザー 	セキュリティID値 (-security_id) シリアル・プログラミング・セキュリティID値 (-flash_security_id) 図RRM/DMM機能用ワーク領域を確保する(-rrm) 開始アドレス (-rrm= <value>) ②OCDモニタのメモリ領域を確保する(-debug_monitor) メモリ領域 (-debug_monitor= <start address=""> - <end address="">) ②オプション・バイト領域のユーザ・オブション・バイトに値を設定する(-us ユーザ・オブション・バイト領域のスーザ・オブション・バイトに値を設定する(-us ユーザ・オブション・バイト領域のスンチップ・デバッグ・オブション・バイトに値を設定する オブジョン・バイト領域のセキュリティ・オブション・バイトに値を設定する セキュリティ・オブション・バイト制御値 (-ocdbg= <value>) ③オブション・バイト領域のセキュリティ・オブション・バイトに値を設定する セキュリティ・オブション・バイト目標域のセキュリティ・オブション・バイトに値を設定する セキュリティ・オブション・バイト制御値 (-security_opt_byte= <value>) RAM領域から除外する領域 (-self/-ocdtr/-ocdtpi) RAM領域から除外する領域にをRAM領域から除外する(-stride_sc ③ドレースRAM領域をRAM領域から除外する(-stride_sc ④ドレースRAM領域をRAM領域から除外する(-stride_sc ④オブシェクト・ファイル/作成時に指定したデバイス・ファイルの情報と気をするが: セキュリアの表別ソ付けアドレスがデバイス・ファイルの情報を置合するが:</value></value></end></start></value>	000000000000000000000000000000000000	

図 4-11 e² studio リンカ・オプション画面(更新後)

【注】 MCU タイプの選択またはチップ部品番号に応じて、これらの設定値は異なります。 詳細設定の構成については、デバイスユーザーズマニュアル:ハードウェア編を参照してください。



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。

🔅 *Smart_Configurator_Example.scfg X		- 0
ソフトウェアコンポーネント設定	じ コードの生成	
コンポーネ 🚵 🖾 🖾 🖂 🕞 🕕 設定		^
■ 次		*
概要 ボード クロック システム コンボーネント 端子 割り込み		

図 4-12 コンポーネント・ページ

4.4.1 コンポーネント・ビューとハードウェア・ビューの切り替え

コンポーネント・ツリーでは、コンポーネント・ビューとハードウェア・ビューの2つのツリー表示を提供しています。以下のアイコンをクリックすることで、表示を切り替えることができます。

(1) [コンポーネント・ビュー] 🎭 アイコン :

コンポーネント・ツリーに、コンポーネントのカテゴリごとにコンポーネントを表示します。

(2) [ハードウェア・ビュー] 🔣 アイコン :

コンポーネント・ツリーに、ハードウェア・リソース階層でコンポーネントを表示します。

コンポーネント	↓ª_ 🖻 🕀
	تو 🔪
フィルタ入力	

図 4-13 コンポーネント・ビューとハードウェア・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

- コンポーネントを追加するには、以下の2つの方法があります。
- (a) [コンポーネントの追加] 🍆 アイコンからのコンポーネント追加
- (b) ハードウェア・リソース・ノードからのコンポーネント追加

[コンポーネントの追加] 🍆アイコンからのコンポーネント追加 (a) について説明します。

a-1. [コンポーネントの追加] *****アイコンをクリック、またはハードウェア・ビューに切り替えて、 ハードウェア・リソース・ノードをダブルクリックします。

#Smart_Configurator_Example.scfg ×		- 0
ソフトウェアコンポーネント設定	じ コードの生成	
コンポー 🚵 🛂 📲 🕕 設定		^
 ■ 歳 21ルタ入力 > (⇒ スタートアップ 、 (⇒ ジェネリック 、 (⇒ jェネリック 、 (⇒ jェ > p) 		
<		>
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 4-14 コンポーネントの追加

- a-2. [コンポーネントの追加] ダイアログの [ソフトウェア・コンポーネントの選択] ページの リストからコンポーネントを選択します(例: A/D コンバータ)。
- a-3. [タイプ]は [コード生成] であることを確認してください。
- a-4. [次へ] をクリックします。

📴 コンポーネントの追加				×
ソフトウェアコンボーネントの選択 使用可能なコンポーネントの一覧から選択!	してください			₿
カテゴリ 全て 機能 全て フイルタ				~
_{ביאל a-2}	Short Name	91 ,a-3	バージョン	^
⊕ A/Dコンパータ		コード生成	1.4.0	
Board Support Packages, - v1.60	r bsp	RL78 Software In	1.60	
⊕ D/Aコンパータ		コード生成	1.3.0	
 ELCL エッジ検出の間引き機能 		グラフィカル・コンフィ	1.0.0	
 ELCL スレーブセレクト端子機能(4 		グラフィカル・コンフィ	1.0.0	
【1 ELCL チャタリング防止機能		グラフィカル・コンフィ	1.0.0	
■ ELCL マンチェスタコード復調機能		グラフィカル・コンフィ	1.0.0	
11日にの 地球パニス 万断知識化		M57/68 757/	100	~
✓ 最新パージョンのみ表示 説明				
アナログ-デジタル(A/D)変換回路は、アナ	ログ入力をデジタ	7ル信号に変換する機能	どです。	<u> </u>
RL78 Software Integration Systemモジ	ュールをダウンロー	- <u>ドする</u>		
ELCLモジュールをダウンロードする				
基本設定				
	a-4			
(?) < 戻る(8) 次へ	(<u>N</u>) >	終了(E)	キャン	セル

図 4-15 コード生成のコンポーネントの追加



- a-5. [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_ADC)
- a-6. リソースを選択、またはデフォルトのリソースを使用します。(例: ADC)
- a-7. [終了]をクリックします。

🕲 コンポーネントの追加	
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	
A/Dコンバータ a-5	
コンフィグレーション名: Config_ADC	
Jy-2: a-0 ADC	~
a-7	
? <戻る(B) 次へ(N) > 終了(F)	キャンセル

図 4-16 [コンポーネントの追加] ダイアログ

- ハードウェア・リソース・ノードからのコンポーネント追加(b)について説明します。
- b-1. [🔀 (ハードウェア・ビュー)]アイコンをクリックし、ツリー・ビューをハードウェア・リソース 階層表示にします。
- b-2. ハードウェア・リソース・ノードをダブルクリックします(例: A/D コンバータ)。
- b-3. [コンポーネントの追加] ダイアログのリストに選択したハードウェア・リソース・ノードのコンポー ネントが表示されます。
- b-4. これ以降は、[コンポーネントの追加] 🝗 アイコンからのコンポーネント追加(a-3)と同じ手順です。



図 4-17 ハードウェア・リソース・ノードからの追加



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。

- コンポーネント・ツリーから1項または複数項のソフトウェア・コンポーネントを選択します。
 (Shift または Ctrl キーを押下しながらクリックすると複数のソフトウェア・コンポーネントを選択できます。)
- (2) [コンポーネントの削除] 🖥 アイコンをクリックします。



図 4-18 ソフトウェア・コンポーネントの削除

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。

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をコード生成するため、[コード生成] ³ ボタンを押してください。



4.4.4 CG ドライバの設定

CG コンフィグレーションを設定するには、以下の手順で行います。

- コンポーネント・ツリーにある CG コンフィグレーションをクリックし、選択します(例: Config_ADC)。
- (2) 右側の設定パネルでドライバを設定します。以下に手順と画面の例を示します
 - a. [分解能設定] で [10 ビット] を選択します。
 - b. [トリガ・モード設定]で[ソフトウエア・トリガ・ノーウエイト・モード]を選択します。
 - c. [A/D チャネルの選択] で [ANI0] を選択します。
 - d. [変換時間] で [2112/fCLK] を選択します。

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
፲ンポーネット 🚵 🛃 🖧 🕀 🕀	設定			í) ^
 ・ ・	コンパレータ動作設定 ④ 停止	○許可		
 	 → 分解能設定 ● 10ビット	🔾 ৪৫ঁ৬৮	O 12Ľット	
✓ ≥ Kライバ A/D IV/(-9)	VREF(+) 設定 ● VDD	OAVREFP	○ 内部基準電圧	
(I) Config_ADC	VREF(-) 設定 ● VSS	OAVREFM		
(2) b	 トリガ・モード設定 ● ソフトウエア・トリガ・ノーウエイト・モード ● ソフトウエア・トリガ・ウエイト・モード ● ハードウェア・トリガ・ノーウエイト・モード ● ハードウェア・トリガ・ウエイト・モード ● ハードウェア・トリガ・ウエイト・モード ■ INTTM01 			
	動作モード設定 連続セレクト・モード ワソショット・セレクト・モード A/Dチャネルの選択 (2) C. 	○ 連続スキャン・モード ○ ワンショット・スキャン・モード ANIO	~	
	変換時間設定 変換時間モード 変換時間 (2) d	標準1 2112/fCLK	~ (66 µs)	
	変換結果上限/下限値設定 ● ADLL≦ADCRn≦ADULで割り込み要 ○ ADUL < ADCRnまたはADLL > ADCRn	求信号(INTAD)を発生 で割り込み要求信号(INTAD)を発生		
	上限值(ADUL)	255		
	下限值(ADLL)	0		¥

図 4-19 コンポーネントの追加

CG コンフィグレーションのコード生成は、デフォルトで生成する設定になっています。

CG コンフィグレーションを右クリックし、 [✓ コート生成 クリックすると、 [□-ト生成 コードを生成しません。

[コード±๙)リックすると、 [✓ コード±๙ コードを生成します。



4.4.5 CG コンフィグレーションのリソース変更

スマート・コンフィグレータでは、ユーザーは CG コンフィグレーションのリソースを変更することがで きます(例: TAU0_1 から TAU0_3 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) CGコンフィグレーションを右クリックします(例: Config_TAU0_1)。
- (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。



図 4-20 リソースの追加

- (3) [リソースの選択] ダイアログにある新しいリソースを選択します(例: TAU0_3)。
- (4) [次へ] ボタンが有効になるので、クリックします。

📴 YY-20	D選択	_		×
リソースの	選択			
使用可能な	リソースの一覧から選択してください			
動作モード:	8 ビット・カウンタ・モード			~
וע-ג:	TAU0_3 (3)			~
	TAU0_1			
	TAU0_3			
	TAU1_3			
	ITL000 ITL001			
	ITL012			
	ITL013			_
	(4)			
?	< 戻る(B) 次へ (N) > 終了	7(F)	キャンセ	!JL

図 4-21 [コンポーネント] ページ-新しいリソースの選択



- (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。

💁 リソースの選択		—		×
コンフィグレーション設定の選択				
コンフィグレーション設定一覧 (7)				
使用する設定の選択? 🖲 以下の設定を	吏用する ○ デフォルト設定を修	使用する		
設定項目 (6)	値		変更可能	^
動作クロック	CK02		はい	
クロック・ソース	fCLK/2		はい	
動作モード設定	上位8ビット		はい	
インターバル時間(上位8ビット)	10		はい	
間隔単位	μs		はい	
インターバル時間(下位8ビット)	10		はい	
間隔単位	μs		はい	
カウント開始時にINTTM01割り込みを到	発生する チェックを外す		はい	×
<			>	
	(8)			
? < 戻る(B)	次へ(N) > 終了(F)	キャンセル	

図 4-22 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: INTTM01 から INTTM03 へ)。

			コードの生成 レボートの生
ンポーネント 🛛 🚵 🗳 📮 🖻	設定		(
 編 課 ンイルタスカ ン (2, 29-トアップ) 	クロック設定 動作クロック クロック・ソース	CK02	✓ (クロック周波教:16000 kHz)
 ◇ (⇒ ジェネリック ごネリック ごネリック ご r_bsp ◇ (⇒ ドライパ) 	動作モード設定 ・上位8ビット 	○下位8ビット	○上位と下位8ピット
✓	インターバル・タイマ設定 インターバル時間(上位8ビット) インターバル時間(下位8ビット)	10 μ 10 μ 込みを発生する	us ~ (実際の値:10) us ~
	割り込み設定 ✓ タイマ・チャネル3のカウント完了で 優先順位 ✓ タイマ・チャネル3のカウント完了で	割り込み発き <mark>(INTTM03)</mark> レベル3(低優先順位) *割り込み発生(INTTM03H)	×
	優先順位	レベル3(低優先順位)	~

図 4-23 自動的に更新されるリソース



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) CG コンフィグレーションを右クリックします。
- (10) [リネーム] を選択して、コンフィグレーションに再度名前をつけます(例: Config_TAU0_1 から Config_TAU0_3 へ)。

コンポーネント		è 🕹 🖓	
		1	6 5
フィルタ入力			
🗸 🗁 ଅଧି–ዞ	・アップ		
🗸 🗁 🖓 I	ネリック		
e	r_bsp		
V 🗁 F51/			
(9)	Config TAU0 1	1	
	✓ □−ド生成)	
	初期化ムロ	のみ出力	
	10,4010,401	が面	
	97 103	e.c	
	😫 削除		
	複製		
(10)	リネーム		
	リセット時の	のデフォルト	
	+ コンフィグレ	ーションの追加	>

図 4-24 コンフィグレーション・リネーム

4.4.6 SNOOZE モード・シーケンサの設定

SNOOZE モード・シーケンサ(SMS)は、グラフィカル・コンフィグレータ・タイプのコンポーネントで、[ソフトウェア・コンポーネントの選択]から追加できます。

📴 コンポーネントの追加				\times
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	尺 してください		ŧ	
カテゴリ 全て 機能 全て フィルタ				~
コンポーネント 日IIC通信 (スレーブ・モード) 日IIC通信 (マスタ・モード) 単 PWMHサカ	Short Name	タイプ コード生成 コード生成 コード生成	バージ 1.7.0 1.8.0 1.9.0	^
ASSNOOZEモード・シーケンサ # SPI(CSI)通信 日 Touch middleware. # UART通信	rm_touch	グラフィカル・コンフィグレータ コート生成 RL78 Software Integration System コード生成	1.4.0 1.7.0 1.40 1.9.0)
 ・	状態でも他の周辺機 イ中に処理を実行する	能の信号をトリガとして起動できるため、CF ることで低消費電力を実現できます。	りから独	^
RL78 Software Integration Systemモジ ELCLモジュールをダウンロードする 基本設定	ユールをダウンロードする	2		~
?	< 戻る(B)	次へ(N) > 終了(F)	キャンセノ	ŀ

図 4-25 SNOOZE モード・シーケンサ (SMS) の追加



SNOOZE モード・シーケンサ(SMS)の GUI ルック・アンド・フィールは、以下の「図 4-26 SNOOZE モード・シーケンサ(SMS)GUI」表示となり、コード生成と比較してよりグラフィカルに表示します。 ブロックをドラッグ・アンド・ドロップすることで構成できます。



図 4-26 SNOOZE モード・シーケンサ (SMS) GUI

表 4-2 S	SMS GUI	エリア	の説明
---------	---------	-----	-----

エリア		説明
(1) SMS ブロック		SMS で使用できるブロックを表示します。
		ブロックはシーケンス(機能)を形成するためのパーツで、A/D 電圧取
		得、比較&分岐、1byte 転送などがあります。
(2) ツールバー	(+	キャンパスを拡大します。
	P	キャンパスを縮小します。
	I	SMS データ管理ダイアログを表示し、使用する変数などを管理します。
	è	SMS シーケンスをインポートします。このアイコンをクリックすると幾
		つかのサンプルシーケンスをご使用いただけます。
	4	SMS シーケンスをエクスポートします。
	~	SMS データファイルを更新します。
	i	SMS データファイルの情報を表示します。
(3) 起動トリガ選択		起動トリガを選択します。
(4) 使用リソース		キャンパスで使用しているレジスタ、命令数を表示します。
(5) キャンパス		SMS ブロックを配置して、シーケンスを作成します。
(6) コンソール		SMS で使用できないコンポーネント設定時にメッセージを表示します。



以下のように SMS ブロックを設定します。

- (1) ブロック・リストからブロックを選択します(例: CSI マスタ受信)
- (2) [CSI マスタ受信] ブロックを、キャンバスの Start ブロックと Finish ブロック間 🚫 のインジケー ターが表示されない位置) にドラッグ&ドロップします。
- (3) [CSI マスタ受信] ブロックをダブルクリックして、CSI マスタ受信設定ダイアログを開きます。
- (4) CSI マスタ受信設定ダイアログのプロパティを設定します。
- (5) [データ管理]を開くと、受信データを編集できます。
- (6) 設定が必要なブロックは、右下が赤く表示されます。正しく設定されると緑に変わります。
- (7) 同じようにいくつかのブロックを追加して、シーケンスを作成します。



図 4-27 SMS ブロック設定

[注] SMS (Snooze Mode Sequencer) モジュールを正常にビルドするには、ルネサスエレクトロニクス ホームページからから最新の SMSASM_Vxxx_setup.exe ツールをダウンロードしてインストールす る必要があります。 ツールバーの 図 ボタンをクリックして、[Renesas ツールチェーン管理] に確 認します。

ルタ入力		Renesas ツールチェーン管理	← - <>
Module Download		VI.11.00	L:#Program Files (X00)#Kenesas Electronics#LS+#LL#LL-KL#V1.11.00#
My Renesas		Renesas DSP Assembler	
Reality AI Authentication		KPIT GNURL78-ELF Toolchai	n
Renesas OF		✓ ✓ Renesas SMS Assembler	
Renesas ツールチェーン管理		✓ v1.00.00	C:¥Program Files (x86)¥Renesas¥SMS¥
Smart Manual		GCC for Renesas RL/8	
Support Folders		4.9.2.202201	C:¥ProgramData¥GCC for Renesas RL78 4.9.2.202201-GNURL78-ELF¥rl78-elf¥rl78-e
Tracealyzer			
TraceX	\sim	<	

図 4-28 SMS アセンブラ



4.4.7 SMS データファイルの更新

以下の手順で、SMS データファイル (ブロック、シーケンス) の更新が行えます。更新することで新しい ブロック、シーケンスをご使用いただけます。

- (1) [SMS データファイルを更新] 🕎 ボタンをクリックして、SMS データファイルの更新を行います。
- (2) SMS データファイルの更新を確認します。
- (3) 新しいバージョンが存在した場合、自動的にダウンロードして更新します。

設定	(1)
 ▶ 算術 ● 1byte転送 ▲ 1byte転送 > 分岐 ● 分岐 ● 小比較&分岐 ▲ 1 再加50.255 ● 特殊 ▼ Wait 	* * * * * * * * * * * * * * * * * * *
▲ valt ● デバッグ ● ボート ● タイマ TAU	Start Finish
🗀 A/D	v
<	> >
(2)	進行情報
(3)	③ 情報 × ① 新しいパージョンのデータはありません。 OK
	図 4-29 SMS データファイルのダウンロード



4.4.8 ELCL 固定機能モジュールのダウンロード

ELCL(ロジック&イベント・リンク・コントローラ)のソフトウェアコンポーネントタイプは、グラ フィカル・コンフィグレータです。ELCLコンポーネントには2種類あり、1種類は「スレーブセレクトピ ン機能」、「チャタリング防止機能」などの固定機能 ELCLコンポーネントで、もう1種類は ELCL Flexible Circuit で、フレキシブルに ELCL 回路を作成できます。ELCL 固定機能モジュールは、[コンポーネ ントの追加] ダイアログから追加できます。コンポーネントリストに含まれていない ELCL 固定機能モ ジュールを使用したい場合は、[ELCL モジュールをダウンロードする] のリンクより、ダウンロードできま す。

カテゴリ 機能	全て 全て		CI RL78	ELCLモジュールのダウンロード			
741/29		<u></u> 90:	ダウンロードするRL78 ELCLモジュールを選択してください (2)				
コンボー	-ネント ^	Short Name	9	9111	パージョン	すべて選択	
 	D27/(-タ ard Support Packages v1.90 pacitive Sensing Unit driver. pacitive Sensing Unit driver. A27/(-タ 1. AND 1. D flip flop 1. EXOR 2. Flexible Circuit パージョンのみ表示 - デジタル(A/D)変換回路は、アナログ入:	r_bsp r_ctsu r_tkbo カモデジタル信号に変換する機能		マナチエスタコード復調機能 チャタリング防止機能 スレーブセレクト端子機能(4線式SPI) エッジ検出の間引き機能 フリップフロップ AND 複数パラメータ監視機能 RL78/G23 Common ELCL Module EXOR Through OR or NOR セレクタ	2.00 2.00 2.00 2.00 1.1.0 1.1.0 2.00 2.0	選択をすべて解除	
RL78 Software Integration Systemモジュールをダウンロードする ELCLモジュールをダウンロードする			1		(3) ダウンロード	キャンセル	
基本設定	<u>E</u>						

ダウンロードした ELCL 固定機能モジュールは、コンポーネン選択トリストに自動的に追加させます。

דעב 🕲	ーネントの追加		_		×			
ソフトウ 使用可能	ェ アコンポーネントの選択 となコンポーネントの一覧から選択してください				8 8 8 8			
カテゴリ	全 て				~			
機能	全て				\sim			
71119								
コンポー	-ネント ^	Short Name	タイプ	バージョン	^			
i i i i i i i i i i i i i i i i i i i	L Flexible Circuit		グラフィカル・コンフィグレータ	1.0.0				
🕄 ELC	CL OR		グラフィカル・コンフィグレータ	1.1.0				
🕼 ELC	L selector		グラフィカル・コンフィグレータ	1.1.0				
🕼 ELC	L Through		グラフィカル・コンフィグレータ	1.1.0				
🕼 ELC	LL エッジ検出の間引き機能		グラフィカル・コンフィグレータ	1.0.0				
Call ELC	ニL スレーブセレクト端子機能 (4線式SPI)		グラフィカル・コンフィグレータ	2.0.0				
Call ELC	L チャタリング防止機能		グラフィカル・コンフィグレータ	1.0.0	~			
 ✓ 最新 説明 アナロク 	 2 最新パージョンのみ表示 説明 アナログ-デジタル(A/D)変換回路は、アナログ入力をデジタル信号に変換する機能です。 							
<u>RL78 So</u> <u>ELCL王3</u> 基本設7	<u>RL78 Software Integration Systemモジュールをグウンロードする ELCLモジュールをグウソロードする 基本設定</u>							
?	< 戻る	(B) 次へ(N) > 終了(F)	キャン	セル			

図 4-31 ELCL 固定機能モジュールの追加



4.4.9 固定機能 ELCL コンポーネントの設定

以下の手順で、固定機能 ELCL コンポーネントを設定します。

- (1) [ソフトウェア・コンポーネントの選択]から固定機能 ELCL コンポーネントを選択します。
 (例: ELCL スレーブセレクト端子機能(4線式 SPI))
- (2) [構成] パネルでドライバを構成します。
 - a. [Input signal selector] :入力信号を選択します。
 - b. [Event control (link processor)]:論理セルブロックを選択します。
 - c. [Output signal selector] : 出力信号を選択します。
- (3) リンクをクリックすることで、アプリケーションノートを開くことができます。



図 4-32 固定機能 ELCL コンポーネントの設定


4.4.10 ELCL Flexible Circuit の作成と編集

ELCL (ロジック&イベント・リンク・コントローラ) Flexible Circuit コンポーネントは、グラフィカル・ コンフィギュレータの新しいコンポーネント・タイプで、コンポーネントリストから選択して使用すること ができます。

📴 コンポーネントの追加		—		×	
ソフトウェアコンボーネントの選択 使用可能なコンポーネントの一覧から選択してください					
カテゴリ 全て 機能 全て フィルタ				~	
コンポーネント 中 D/Aコンバータ 開 ELCL AND 開 ELCL D flip flop 開 ELCL EXOR 4	Short Name	タイプ コード生成 グラフィカル・コンフィグレータ グラフィカル・コンフィグレータ グラフィカル・コンフィグレータ グラフィカル・コンフィグレータ	パージ 1.5.0 1.1.0 1.2.0 1.1.0 1.0.0		
は ELCL OR グラフィカル・コンフィグレータ 1.1.0 は ELCL selector グラフィカル・コンフィグレータ 1.1.0 く ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・					
The logic and event link controller (E peripheral functions through interna peripheral functions without CPU int RL78 Software Integration Systemモジニ ELCLモジュールをダウンロードする 基本設定	LCL) links signals output b l logic cell blocks, allowing ervention. <u>1ールをダウンロードする</u>	y peripheral functions to other 9 direct communications betwe	specified een	~ k	
?	< 戻る(B) 次へ(I	N) > 終了(F)	キャンセ	л	

図 4-33 ELCL Flexible Circuit コンポーネントの追加



ELCL フレキシブル回路コンポーネントは、ELCL 回路の作成と編集のためのドラッグアンドドロップ操作をサポートする直感的な GUI を提供し、回路設計後に ELCL レジスタ設定を自動的に生成できます。



図 4-34 ELCL Flexible Circuit コンポーネント

エリア	説明
(1)ELCL 要素	ELCL で使用できる要素を表示します。
(2) プロパティ	選択した ELCL 要素の設定を表示します。
(3) ツールバー 🏓	キャンパスを拡大します。

表 4-3	ELCL Flexible Circuit GUI エリア説明

(3) ツールバー	Æ	キャンパスを拡大します。
	P	キャンパスを縮小します。
	?	ヘルプを表示します。
(4) キャンバス	入力	ELCL 入力要素を配置する領域です。
	論理セルブロック	ELCL 論理セルブロック要素を配置する領域です。
	出力	ELCL 出力要素を配置する領域です。



以下の手順に従って、ELCL Flexible Circuit を作成します。

- (1) パネルから入力/論理セル/出力の ELCL 要素をキャンバスにドラッグ&ドロップします。
- (2) キャンバスで入力/論理セル/出力の ELCL 要素を選択し、プロパティを設定します。
- (3) 始点を終点にドラッグ&ドロップして接続します。
- (4) ELCL 回路作成後、[コードの生成] ボタンをクリックすると、ELCL レジスタ設定が生成されます。
- 【注】 手順 (1)、(2)、(3) は決まった操作ではなく、各ステップを自由に操作し ELCL 回路を作成または編 集できます。



図 4-35 ELCL Flexible Circuit 作成手順



以下に、ELCL 回路を簡単に作成し、正しい設計に導くのに役立つ GUI 操作の詳細を示します。

(1) 始点を終点にドラッグ&ドロップして接続します。



【注】 ポイントをドラッグした時、接続可能な終点だけを薄緑色でハイライト表示します。



(2) ハードウェアの制限を超えた接続や設定を行った場合、各部にエラーが表示されます。 ユーザーは、示されたメッセージに基づいてエラーを解決できます。



図 4-38 ELCL Flexible Circuit エラー表示

表 4-4	ELCL Flexible Circuit エラーメッセージー覧	
-------	----------------------------------	--

No.	エラーメッセージ
1	現在選択されている入力イベント信号は、pass-through/AND/OR/EX-OR 回路の入力 0 に接続でき
	ません。
2	信号選択レジスタ xxx はすべて使用されており、割り当てられません。
2	ELL1SEL4 と ELL1SEL5 は両方とも使用されています。同じ論理セルブロック内の別のフリップ
3	フロップで使用している信号と同じ信号を使用してください。
1	このクロック設定を使用する場合は、別のフリップフロップで使用している信号と同じ信号を設
4	定してください。
5	フリップフロップのセットとリセットは、異なる信号を選択する必要があります。
6	ELCL 回路には割り当てられていないリソースが存在するか、ライン接続が完了していないため、
0	一部のコードが生成されないか、正しくありません。
7	入力信号に割り込み要求信号を接続した場合、出力信号に周辺機能のハードウェアトリガ以外は
	接続できません。
8	xxx はすでに使用されています。利用可能な他のリソースを選択してください。



4.4.11 RL78 Software Integration System モジュールのダウンロード

RL78 Software Integration System モジュールは、ドライバ、ミドルウェア、アプリケーション SW のソ フトウェア・コンポーネントで、コードを生成するための簡単な GUI を提供し、[コンポーネントの追加] ダ イアログからダウンロードできます。

- (1) [図 4-14 コンポーネントの追加] ¹ アイコンをクリックして [コンポーネントの追加] ダイアログを開きます。
- (2) [RL78 Software Integration System モジュールをダウンロードする] リンクをクリックして、使用する RL78 Software Integration System モジュールをダウンロードします。

カテゴリ 全て 機能 全て フィルタ コンポーネント へ Si # A/Dコンパーク 聞 Board Support Packages v1.60 # CApacitive Sensing Unit driver. # O/Aコンパーク 図 ELCL AND 図 最新パージョンのみ表示 説明 アナログ-デジタル(A/D)変換回路は、アナログ入力を	hort Name bsp _ctsu	タイプ コード生成 RL78 Software Integration RL78 Software Integration コード生成 グラフィカル・コンフィグレータ	パージ 1.4.0 1.60 1.30 1.3.0 1.1.0	~
フノポーネント Si コンポーネント Si # A/Dコンパータ Board Support Packages v1.60 # D/Aコンパータ r. 世/Aコンパータ Bi 型 人力ンパータ Bi ジーク Bi ブーク Bi ジーク Bi ジーク Bi アナログ・デジタル(A/D) S換回路は、アナログへ力を	hort Name _bsp _ctsu	タイプ コード生成 RL78 Software Integration RL78 Software Integration コード生成 グラフィカル・コンフィグレータ	パージ 1.4.0 1.60 1.30 1.3.0 1.1.0	~
コンポーネント ^ SJ # A/Dコンパーク # Board Support Packages v1.60 # Capacitive Sensing Unit driver. # D/AJンパーク I ELCL AND	hort Name _bsp _ctsu	タイプ コード生成 RL78 Software Integration RL78 Software Integration コード生成 グラフィカル・コンフィグレータ	パージ 1.4.0 1.60 1.30 1.3.0 1.1.0	^
 	_bsp _ctsu	コード生成 RL78 Software Integration RL78 Software Integration コード生成 グラフィカル・コンフィグレータ	1.4.0 1.60 1.30 1.3.0 1.1.0	~
 	_bsp _ctsu	RL78 Software Integration RL78 Software Integration コード生成 グラフィカル・コンフィグレータ	1.60 1.30 1.3.0 1.1.0	~
 	ctsu	RL78 Software Integration コード生成 グラフィカル・コンフィグレータ	1.30 1.3.0 1.1.0	~
 ■ D/Aコンパータ 【図 ELCL AND □ 量新パージョンのみ表示 説明 アナログ・デジタル(A/D)変換回路は、アナログ入力を 		コード生成 グラフィカル・コンフィグレータ	1.3.0 1.1.0	~
は ELCL AND 「最新パージョンのみ表示 説明 アナログ・デジタル(A/D)変換回路は、アナログ入力を		グラフィカル・コンフィグレータ	1.1.0	~
✓ 最新パージョンのみ表示 説明 アナログ-デジタル(A/D)変換回路は、アナログ入力を				
	デジタル信号に素	変換する機能です。		^ ~
<u>RL78 Software Integration Systemモジュールをダウ ELL1モンユールをグリフロートする 基本設定…</u>	<u>>D-F\$33</u>			

図 4-39 RL78 Software Integration System モジュールのダウンロード

【注】 ダウンロードには、My Renesas へのログインが必要です。ログインしていない場合は、次のダイア ログボックスでログインを求められます。

My Renesas ロ グイン My Renesasアカウントの情報を入力してください。	× My Renesas
メール・アドレス: [パスワード:	パスワードをお忘れの方? Proxy Settings
e ² studioは使用状況を記録し、ルネサスに送信することができます。送信されたデータは、製品・サービスの改善や、より良いサポ 用されます。 〇 下記リンクの「プライバシーポリシー」と「収集する情報について」を理解し、ルネサスに使用状況のデータが送信されることに同 ク ルネサスに使用状況のデータが送信されることに同意しない。 <u>プライパシーポリシー」収集する情報について</u> この設定は、後で[設定]ウィンドウで変更することができます。	ート・メンテナンスのために使 食する。
<u>My Renesas</u> へご登録いただくと、ツール製品などのダウンロードサービスやメールニュースなどの各種サービスをご利用いただけるよう	になります。 ログイン キャンセル

図 4-40 My Renesas ログイン



- (3) [RL78 Software Integration System モジュールのダウンロード] ダイアログで必要なモジュールを選択します。
- (4) [参照] をクリックして、ダウンロードしたモジュールを保存する場所を選択します。
- (5) [ダウンロード]をクリックすると、モジュールのダウンロードが開始されます。

	, ,				(3)
	タイトル	ドキュメントNo.	リビジョン	発行百 ^	すべて選
	RL78 Family FS2012 Sensor Control Mo	R01AN6196EJ	Rev.1.12	2023-04-2	課題たすべて
	RL78 Family HS400X Sensor Control M	R01AN6446EJ	Rev.1.02	2023-04-2	MEDIC 3. (
	RL78 Family HS300x Sensor Control M	R01AN6194EJ	Rev. 1.22	2023-03-0:	
	RL78 Family ZMOD4410, ZMOD4450 a	R01AN6197EJ	Rev. 1.21	2023-03-0	
	RL78 Family Sensor I2C Communicatio	R01AN6193EJ	Rev.1.11	2023-03-0	
\square	RL78 Family OB1203 Sensor Control M	R01AN6379EJ	Rev. 1.02	2023-03-0	
	RL78 Family FS1015 Sensor Control Mo	R01AN6198EJ	Rev.1.01	2023-03-0:	
\square	RL78 Family FS3000 Sensor Control Mo	R01AN6195EJ	Rev. 1.01	2023-03-0:	
	RL78ファミリ CTSUモジュール Software Inte	R11AN0484JJ	Rev.1.30	2023-02-1: 🗸	
<	-			>	6
モジ	ュール・フォルダー・パス:				È
	C:¥Users¥ ¥.eclipse¥com.renesas.	platform_download	d¥RL78_Mo	dules¥GenericM	odules #

図 4-41 RL78 Software Integration System モジュールのダウンロード

4.4.12 RL78 Software Integration System モジュールの追加

RL78 Software Integration System モジュールを追加する手順は、以下の通りです。

- (1) [図 4-14 コンポーネントの追加] 놓 アイコンをクリックして [コンポーネントの追加] ダイアログを 開きます。
- (2) コンポーネントリストから RL78 Software Integration System タイプのコンポーネントを選択します。複数のモジュールを選択したい場合は、Ctrl キーを押下しながらクリックします。
- (3) [終了]をクリックします。

😰 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	てください		ŧ	
カテゴリ 全て 機能 全て フイルク				~
コンボーネント ■ A/Dコンバータ (2)	Short Name	タイプ コード生成	パージ 1.4.0	^
 ■ Board Support Ackages: Visio ■ Capacitive Sensing Unit driver. ✓ 最新バージョンのみ表示 	r_ctsu	RL78 Software Integration	1.30	v
説明 依存モジュール: r_bsp パージョン 1.13 The CTSU2L module is a CTSU2L driver configured assuming access via the To application.	for the Capacitive Se uch middleware laye	nsing Unit. The CTSU2L module r, but can also be accessed from	is the user	<
RL78 Software Integration Systemモジュー ELCLモジュールをグウンロードする 基本設定	<u>ルをダウンロードする</u>			
		(3)		
?	< 戻る(B) り	ζ∧(N) > 終了(F)	キャンセ	IL

図 4-42 RL78 Software Integration System モジュールの追加



4.4.13 RL78 Software Integration System モジュールの設定

RL78 Software Integration System を使用するには、構成オプションを設定し、設定方法はコンポーネントによって異なります。

構成パネルで構成オプションを設定しコード生成を行うと、RL78 Software Integration System の構成ファイルに自動的に設定が反映されます。

【注】RL78 Software Integration System モジュールの構成ファイルは、r_config フォルダに生成されます。

コンボーネント 盗 込 に 正 設定 ① ・ ()	ソフトウェアコンポーネント設定	Ĕ		🔋 📄 コードの生成 レポートの生成
Rel は TUI/9入力 TUI/9入力 Charlen Configurations Parameter check Use system default Parameter check Use system default Parameter check Use system default Parameter check Use system default Parameter check Use system default Parameter check Use system default Parameter check Use system default Parameter check Par	コンポーネント 🚵 🛃 📮 🕀	設定		i
		プロパティ ▼ ② Configurations # Parameter check # Data transfer of INTCTSUWR and INTCTSURD # DTC setting # Auto-judgment function in Snooze mode using SMS # Data storage address setting for CTSURD # Data storage address setting for CTSUWR # Interrupt level for INTCTSUWR # Interrupt level for INTCTSURD # Interrupt level for INTCTSUFN ▼ 10 UV-ス ▼ 10 CTSU ■ TSCAP端子 ■ TSO的端子 ■ TSO1端子	値 Use system default Interrupt handler Setting in r_ctsu Disable 0xFF800 0xFF800 Level 2 Level 2 Level 2 Level 2 Level 2 回 使用する 回 使用する	
				~ ~

図 4-43 RL78 Software Integration System モジュールの設定



4.4.14 BSP コンフィグレーションのバージョン変更

BSP コンフィグレーションのバージョン変更は、以下の手順で行います。

(1) コンポーネント・ツリーから、バージョンを変更する r_bsp コンポーネントを右クリックします。

Smart_Configurator_Example.scfg ソフトウェアコンポーネント ポ マント マン	9 X 定		じ コードの生成	 □ □ □ レポートの生成
コンポーネント 」 ^a _Z 🕞 🕀	設定			í
 ・ ・	プロパティ ✔ 傪 Configurations		値	^
✓ (⇒ スタートアップ)	# Start up select # Control of invalid mem	ory access detection	Enable (use BSP startup) Disable	
	# RAM guard space(GRA	M0-1)	Disabled	
	の変更	of port function(GPORT)	Disabled	
	·02.2	rupt function(GINT)	Disabled	
😫 削除		of clock control function, voltac	Disabled	
リセット時	きのデフォルト	DFLEN)	Disables	
		 functions by Code Generator/Sn 	Enable	
サンプルフ	ブロジェクトのダウンロードとインボート		Enable	
	# Parameter check enabl	e	Enable	
	# Setting for starting the	high-speed on-chip oscillator at th	High-speed	~
	<			>
				~ ~
概要 ボード クロック システム コンポージ	ネント 端子 割り込み			

図 4-44 BSP コンフィグレーションのバージョン変更

- (2) コンテキスト・メニューから [バージョンの変更...] を右クリックします。
- (3) [バージョンの変更] ダイアログボックスで変更したいバージョンを選択します。デバイスが対応していないバージョンを選択した場合、[選択されたバージョンはターゲット・デバイスまたはツールチェーンをサポートしていません。] と表示されますので、対応しているバージョンを選択してください。
- (4) [次へ]をクリックします。

 パージョンの変更 パージョンの選 変更後のパージョンの 	快		_		×
コンポーネント名: 現在のバージョン: 変更後のバージョン:	r_bsp 1.00				
	1.01				
?	< 戻る(B)	次へ(N) >	終了(F)	キャン	ะม

図 4-45 BSP コンポーネントのバージョン選択



(5) バージョン変更により、変更する設定項目の一覧が表示されますので、問題ないことを確認し、[終 了] をクリックします。

🕲 バージョンの変更	—		×
設定の変更内容			
以下の設定が追加または削除されます。			
設定項目		状態	
変更はありません。			
<			>
(P) < 戻る(B) 次へ(N) > 終了(F)		キャンセ	μ

図 4-46 設定変更項目の確認

(6) [バージョンを変更し、コードを生成しますか。] と表示されますので、問題なければ [はい] をクリックします。

📴 バージョンの変更		×
バージョンを変更し、コードを生成しますか。	はい(Y) いいえ(N)

図 4-47 バージョンの変更確認

(7) BSP コンポーネントのバージョンが変更され、自動的にコード生成が実行されます。



4.4.15 コンポーネントのコンフィグレーションのエクスポート

[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]²²イ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。

コンポーネント	è 🖾 📮 🕀
⇒E	ت ن
type filter text	
 ✓ (⇒ スタートア) ✓ (⇒ ジェネ) 	ップ Jック
▼	isp
🗸 🗁 ७१८	
💕 Co	nfig_TAU0_0

図 4-48 コンフィグレーションのエクスポート

4.4.16 コンポーネントのコンフィグレーションのインポート

[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	
⇒"⊟ ដំ	1
type filter text	
✓	。 ク

図 4-49 コンフィグレーションのインポート



4.4.17 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 4-15 コード生成の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					×
フィルタ入力	コンポーネント			⇔ - ⇒	▼ 000
 一般 一般 C/C++ Java Language Servers Library Hover Oomph Remote Development Renesas Breakpoints Lite UI Mode Logging Module Download My Renesas Reality AI Authentication Renesas QE Renesas QE Renesas QE Renesas QE Renesas QE Renesas V=/J.F.IY管理 Smart Manual Support Folders TraceAX Xマート・コンフィグレータ MCU/MPU/(ッケージ外観	バックアップの設定 「バックアップ設定 trashへのバックアップ コード生成コンボーネ API関数出力: APIコードスタイル: FIT(RX) / SIS(RL78 コード生成時の動作 依存コンポーネントの 依存間係の追加: 依存関係の違加: 依存関係の違認: ファルダ設定 モジュールの保存先(E有効にする 数 (1-20): 5 小の設定 設定に合わせてすべて出力する 7クロで値を表記 (RISC-V MCU)コンポーネントの設定 設定ファイルを更新する 確認と追加 追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トはチェッ tModule Downloadページで設定できます	/クから除外する /クから除外する	適用	> > > > > >
? ù 🖌 🔘		ž	適用して閉じる	キャンセ	.IL

図 4-50 コンポーネントの基本設定

【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 4-51 に示す)を設定することで、バックアップのため、トラッシュフォルダに生成したフォルダの数を制限できます。 制限を 超えると、新しいフォルダが古いフォルダを置き換えます。

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	





【注】 2. コード生成時の動作には、「設定ファイルを更新する」と「すべてのコンポーネントファイルを 再生成する」の2つのオプションがあります。デフォルトの設定は、「設定ファイルを更新す る」です。「設定ファイルを更新する」を選択し、コード生成する場合、スマート・コンフィグ レータはプロジェクト内にファイルが存在するかどうかをチェックします。ファイルが存在する 場合、そのファイルは書きされません。ただし、設定ファイル(例:xxx_config.h)はコード生成 するたびに更新されます。「すべてのコンポーネントファイルを再生成する」を選択しコード生 成する場合、ファイルが常に上書きされます。

- FIT(RX) / SIS(RL78 /	RISC-V MCU)コンポーネントの設定
コード生成時の動作:	設定ファイルを更新する ~
−依存コンポーネントの確	設定ファイルを更新する すべてのコンポーネントファイルを再生成する

図 4-52 [コード生成時の動作:]の変更

【注】 3. 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 カする」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。

コード生成コンポース	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 4-53 [API 関数出力:]の変更

e² studio 2022-10 から、初期化 API のみ出力する機能を個別の構成 (コード生成コンポーネント) に適用できるようになりました。 選択したコンポーネントを右クリックし、コンテキスト・メ ニューから [初期化 API のみを出力] を選択してください。



図 4-54 [初期化 API のみ出力:] の変更

【注】 4. HEX 値でコードを生成するには、下図の [マクロを使用せず即値(16 進数)で表記] オプションに変 更してください。[マクロで値を表記] に戻すと、マクロ記述ですべての API が生成されます。

コード生成コンポーキ	やントの設定
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:	マクロで値を表記 〜
FIT(RX) / SIS(RL7	マクロで値を表記 マクロを使用せず即値(16進数)で表記

図 4-55 [APIコードスタイル:]の変更



【注】 5. モジュールのバージョンとその依存関係が不一致の場合に、警告メッセージ W04020011 を表示し ます。モジュールとその依存関係の改訂履歴を確認し、使用しているモジュールに変更が不要な 場合は、この警告を無視してかまいません。この警告を消すには、コンポーネント基本設定の[依 存関係の確認:] リストボックスで「依存コポーネントのバージョンをチェックしない」を選択 し、[OK]をクリックします。

依存関係の確認:	より新しいバージョンの依存コンポーネントはチェックから除外する 🗸 🗸 🗸
コールガシーテ	依存コンポーネントをチェックしない
ノアルグ設定	より新しいバージョンの依存コンボーネントはチェックから除外する
モジュールの保存先	依存コンボーネントのバージョンを厳密にチェックする

図 4-56 [依存関係の確認:]の変更



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。

子設定							で コードの生	。 成 レポート
ドウェアリソース 🕀 🖻 🖧 💑	端子機能						२ ।	
ィルタ文字列を入力	フィルタス	力 (* = any strir	ng, ? = any characte	r)			すべて	(
<u>*</u> すべて ^	使用する	機能	PIOR	端子割り当て	端子番号	方向	備考	コメント
\$@ 1/0ボート		ANIO		P20/ANI0/AVREFP/EI20	/ 117	1	この端子を初期化するコンポーネントが	
2019/0 発生回路		ANI1		✓ 設定されていません	/ 設定されていま	せなし		
(91 7· 7/1· 1= yh		ANI2		✓ 設定されていません	/ 設定されていま	せ なし		
V TAUO		ANI3		✓ 設定されていません	/ 設定されていま	せ なし		
VIAU00		ANI4		✓ 設定されていません	/ 設定されていま	せ なし		
IAU01		ANI5		✓ 設定されていません	/ 設定されていま	せ なし		
TAU02		ANI6		✓ 設定されていません	/ 設定されていま	せ なし		
		ANI7		🥒 設定されていません	〃 設定されていま	せ なし		
TAU04		ANI8		🥒 設定されていません	〃 設定されていま	せ なし		
TAU05		ANI9		🥒 設定されていません	🥒 設定されていま	せ なし		
TAU05 TAU07		ANI10			🥒 設定されていま	せ なし		
TAU11		ANI11		✓ 設定されていません	/ 設定されていま	せ なし		
		ANI12		✓ 設定されていません	/ 設定されていま	せ なし		
		ANI13			🥒 設定されていま	せ なし		
		ANI14		✓ 設定されていません	〃 設定されていま	せ なし		
		ANI16		✓ 設定されていません	〃 設定されていま	せ なし		
		ANI17		✓ 設定されていません	〃 設定されていま	せ なし		
衣示切り笛	1							

図 4-57 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]にボードの初期端子機能情報が表示されます。 また、[機能] 選択リストに表示される []] アイコンは、ボードの初期端子機能を示します。

子設定	gurator_cxample.scrg ×							
							コードの生	成 レポートの生
子番号							2	l 🖬 ès e
フィルタ入力 ('	* = any string, ? = any character)						すべ	τ `
端子番号	端子名	ボード機能	機能	方向	備考	シンボリック名		1
1	P142/SCK30/SCL30	P142	設定されていません	なし			Pmod1-4[RTS]	
2	P141/PCLBUZ1/INTP7	P141	設定されていません	なし			Pmod1-1[CTS]	
3	P140/PCLBUZ0/INTP6	INTP6	設定されていません	なし			Pmod1-7[INT]	
4	P120/ANI19/IVCMP1/EI120		設定されていません	なし			J4-12	
5	P37/ANI21		設定されていません	なし			J4-11	
6	P36/ANI22		設定されていません	なし			J4-10	
7	P35/ANI23		設定されていません	なし			J4-9	
8	P34/TxDA1	TxDA1	設定されていません	なし			J9-2[IO1/TX]	
9	P33/RxDA1	RxDA1	定されていません 🗸	なし			J9-1[IO0/RX]	
10	P32/CLKA1		設定されていませ	the			J4-8	
11	P106/TI17/TO17		P33	_			J9-4[IO~3]	
12	P105/TI16/TO16		RxDA1				J9-5[IO~4]	
13	P104/TI15/TO15		記たされていません	140			J9-6[IO~5]	
14	P103/TI14/TO14		設定されていません	なし			J9-7[IO~6]	
15	P47/INTP2	P47	設定されていません	なし			Pmod2-7[IO1]	
16	P46/INTP1/TI05/TO05		設定されていません	なし			J9-8[IO~7]	
17	P45/SO01		設定されていません	なし			J4-7	
18	P44/SI01/SDA01	SDA01	設定されていません	なし			Grove3-2[SDA]	
19	P43/SCK01/SCL01	SCI 01	設定されていません	なし			Grove3-1[SCL]	

図 4-58 端子ページ (端子番号)



4.5.1 PIOR 機能による端子割り当ての変更

PIOR「フィルタ機能」は、端子機能設定の管理、端子機能設定の再構成、端子機能の競合のチェックに 便利な機能です。PIOR機能の割り当てを変更するには、以下の手順で行います。

- (1) ツールテキスト入力ボックスに「pior1」と入力し、PIOR1 に関連するすべての端子機能を表示させ ます。
- (2) 端子割り当ての1つを変更すると、PIOR1に関連するすべての端子機能割り当てが自動的に再割り 当てされます。
- (3) 端子エラーメッセージは、[備考] 欄と [構成問題ビュー] に表示されます。
- (4) 端子エラーメッセージが表示されたら、端子割り当てを再構成する必要があります。

ハードウェアリソース 🕀 🖻 💐 🚟	端子機能	(1)					2 🗉 🖬 🕹 🕹
フィルタ文字列を入力	pior1						× すべて ~
 ♥増 シリアル・インタフェースIICA ● IICA0 ● IICA1 ● UB シリアル・インタフェースUARTA ● UARTA0 ● UARTA1 ● UARTA1 ● UARTA1 ● ロジック&イベンドリンクコントローラ ● 割り込み機能 ● キー割り込み ● 幸客電式タッチセンサ ● オンチップ・デバッグ機能 ● 電源 	使用する 機能	PIOR 2 PIOR1 CK00 PIOR1 CK20 PIOR1 20 PIOR1	 場子割り当て (2) り音を為れていません。 りちがPCBUZ/SCK00 りまたれていません。 り設定れていません。 り設定れていません。 り設定れていません。 り設定れていません。 りたたれていません。 りまれていません。 	 端子番号 ジラ定されていません ジラ定されていません ジラ定されていません ジラ定されていません ジラごされていません ジラごされていません ジラごされていません ジラごされていません ジラごされていません ジラごされていません ジラごされていません ジラごされていません ジラごされていません 	方	□ 考 (3) 複数の端子機能が同一端子に割り ソフトウェアコンポーネントにより使用さ ソフトウェアコンポーネントにより使用さ	当てられています れていますが、…
端子機能 端子番号 概要 ボード クロック システム コンボーネント 端子 目	則り込み						
 ユンフィグレーションチェック × 5 errors, 0 warnings, 0 others 	(3)						₹ 8 - E
記述/説明		^			型		
 ◆ ● 様子(5項目) ◆ Eo4010003: PCLBUZ1(Config_PCLBUZ0で) ◆ Eo4010003: SCK00(Config_CSL00で防疫): ◆ Eo500010: 端子 72 を複数の機能で使用で) ◆ Eo5000011: SCX20を*端子*パージで割り当て ◆ Eo5000011: SCX20を*端子*パージで割り当て 	(設定)が使用する端 が使用する端子と次) ?きません。端子 72 (ててください。 てください。	端子と次の端子が競合し この端子が競合しています に PCLBUZ1, SCK00の	、ています:SCK00 (Pin Allocatorで設定)、SCK00 (Confi :PCLBUZ1 (Pin Allocatorで設定)、PCLBUZ1 (Config 機能が割り当てられています。	g_CSlooで設定). g_PCLBUZoで設定).	端子 端子 端子 端子 端子 端子		

図 4-59 PIOR フィルタ機能

PIOR 設定コードは、bsp ファイル: \ProjectDir\src\smc_gen\r_bsp\r_config\ r_bsp_config.h に生成されま す。 PIOR 設定コード値を変更したい場合は、関連するピンの割り当てを変更して、再度コードを生成して ください。

🔅 *Smart_Configu	rator_Example.scfg ×							- 0
端子設定							じ コードの生成	□ レポートの生成
ソフトウェアコンポー	차가 🗉 🖻 🎝 😹	端子機能					3	a 22 23
フィルタ文字列を	入力	フィルタ入び	力 (* = any string	g, ? = any characte	r)		すべて	~
🗸 💑 r_bsp		使用する	機能	PIOR	端子割り当て	端子番号	方向	備考
📦 r_bsp			PCLBUZO	PIOR3	P140/PCLBUZ0/INTP6	/ 3	0	
	ロリアフリーエリ利御回路 ia PCLBUZO							
	· <u>j_</u> ·					_		
		< .						,
端子機能 端子番	·뮥							
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						
#Smart_Config	urator_Example.scfg	h r_bsp_con	fig.h 🔀					
506								^
507	128/100-pin 0 : PCLBUZ0 -	P140						
509	1 : PCLBUZ0 -	P51						
510	*/	. DTOD2 (0)	18 5					
511	#detine BSP_CFC	1_HTOR3 (0)	/~ Generat	ed vaiue. Do	not edit this manually */			× .
	<							>

図 4-60 PIOR のコード生成



4.5.2 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割り 当てます。端子の割り当て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割り当てを変更するには、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] 品 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_INTC)。
- (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、または端子番号欄で、端子配置を変更します(例: P46 から P56)。
- (5) または、[選択されたリソースの次の端子割り当て先] 💙 ボタンをクリックし、端子配置を変更しま す。クリックするごとに、機能を持つ端子が表示されます。

	(1)						(5)	
ソフトウェアコンポーネント 🕀 🖻 📲	品 ;	耑子機能					્ર	
		フィルタ入ナ] (* = any s	string, ? = any	r character)			すべて ~
✓ 💑 r_bsp	(3)	使用する	幾能	PIOR (2	端子割り当て	端子番号	方向	備考
r_bsp			INTP1	PIOR5	P46/INTP1/TI05/TO05	/ 16	1	
◇ 売割り込みコントローフ			INTPO		/ 設定されていません	∥ 設定されていません	なし	
(2) Config_INIC			INTP2		〃 設定されていません	∥ 設定されていません	なし	
> 🐴 179-100.918			INTP3	PIOR5	∥ 設定されていません	∥ 設定されていません	なし	
			INTP4	PIOR5	∥ 設定されていません	∥ 設定されていません	なし	
			INTP5	PIOR4	∥ 設定されていません	∥ 設定されていません	なし	
			INTP6	PIOR5	∥ 設定されていません	🥒 設定されていません	なし	
			INTP7	PIOR5	∥ 設定されていません	∥ 設定されていません	なし	
			INTP8	PIOR5	∥ 設定されていません	∥ 設定されていません	なし	
			INTP9	PIOR5	∥ 設定されていません	∥ 設定されていません	なし	
			INTP10	PIOR1	∥ 設定されていません	∥ 設定されていません	なし	
			INTP11	PIOR1	∥ 設定されていません	∥ 設定されていません	なし	
		<						>

図 4-61 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"この端子を初期化するコンポーネントがありません"とい う注意書きがつけられます。



4.5.3 MCU/MPU パッケージを使用した端子の設定

スマート・コンフィグレータでは、 MCU/MPU パッケージビューで端子設定を視覚化します。MCU/MPU パッケージビューを画像ファイルに保存し、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、マウスホイールをスクロールして、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。



図 4-62 MCU/MPU パッケージを使用した端子設定



4.5.4 端子機能から端子番号の表示

端子機能に関連付けられている端子番号に移動できます。端子機能か端子番号に移動するには、以下の手順で移動します。

- (1) [端子機能] タブで、使用する端子機能を右クリックしポップアップメニューを開きます。
- (2) [端子番号タブにジャンプ]を選択します。
- (3) [端子番号] タブに移動し、(1) で選択した端子機能の端子番号を表示します。

							-د	ドの生成 レポート	ወ ያ
トウェアコン	# 🕀 🕞 ↓ª₂ 👪	端子機能					3	📕 📑 🖻	3 (
ィルタ文字列	列を入力	フィルタス	力 (* = any string, ? =	any character)				すべて	
🗸 📥 r_bsp	0	使用する	5 機能	PIOR	端子割り当て		端子番号 方向	備者	
🔍 r_	bsp		INTRO		/ D127/EI127/INITD	•	/ 20 L	1 mar - 2	
/ 🚣 割り返	込みコントローラ		INTPU	DIODE	/ PIS//EIIS//INTP // 30字さねていませ/	0	 29 1 4 記字されていませた! 		
🚽 Ca	onfig_INTC		INTPL	PIUKS	/ 設定されていません	v	/ 設定されていませんし		
			INTP2	DIOPE	/ 設定されていません	v	/ 設定されていませんし		
			INTDA	PIORS	/ 設定されていません		 一 設定されていません! / 設定されていません! 		
			INTP4	PIORA	/ P16/E016/CCD0		1/IN / 76		
			INTRG	PIORS	/ P140/PCLBUZ0/I	NTP6	従子番号タブにジャンプ		
			INTP7	PIOR5	/ 設定されていません		加丁面オノノにノマノノ		_
			INTP8	PIOR5	/ 設定されていません		コメノトを端す番号ツノのコメ	ノトに迫加	
			INTP9	PIOR5	/ 設定されていません		コメント削除		
			INTP10	PIOR1	/ 設定されていません		端子割り当て		
			INTP11	PIOR1	/ 設定されていません		端子割り当て解除		
					- BARECIVEVIOLEN	-			_
		1							
root 用記 ボード クロ mart_Confi	Iック システム コンポーネン igurator_Example.scfg >		り込み						
^{成 能} 称 70 ボード 70 mart_Confi 子設定	リック システム コンボーネン igurator_Example.scfg 〉		以込み		ŀ		7-	でし ぶの生成 レポート	
^{成用E} *m ³ ボード クロ mart_Confi ご設定 番号	リック システム コンボーネン igurator_Example.scfg ン		以込み		•]-	で ドの生成 レポート 1日 1日 1日 1日 1日 1日 1日 1日	-0
^{(成肥 Mi]} ボード クロ nart_Confi ご設定 番号	ック システム コンポーネン igurator_Example.scfg ン * = any string, ? = any ch	ト 端子 割り ×			•]-	 でします。 ドの生成 レポート 日本 マベて 	-0
^{12歳距} ¹⁴⁶ 7 ポード クロ mart_Confi ご設定 番号 ルタ入力(* 子番号	ック システム コンボーネン igurator_Example.scfg ン * = any string, ? = any ch 端子名	ト 端子 割り Aracter)	以込み	ポード機能	機能	方向	[倍考	 で」 ドの生成 レポート 単一二 単一二 すべて シンボリック名 	-0
¹²² 版 NM 7 ポード クロ mart_Confi ?設定 番号 ルタ入力(* 子番号 72	igurator_Example.scfg > * = any string, ? = any ch 端子名 P55/PCLBUZ1/SCK00	ト 端子 割り Aracter)	リ込み	ボード機能	機能 設定されていません。	方向なし	[備考	 「の生成レポート 」 すべて シンボリック名 	- 0
^{(x)版 米市子 クロ mart_Confir 子設定 番号 ルタ入力(* 子番号 72 73}	igurator_Example.scfg > igurator_Example.scfg > * = any string, ? = any ch 端子名 P55/PCLBUZ1/SCK00 P56/INTP1	ト端子 割り Aracter)	リ込み	ポード機能	機能 設定されていません 設定されていません 設定されていません	方向なしなし]-	 「の生成 レポート 「「の生成 レポート 「」「」」」 すべて シンボリック名 	- 0
⁽¹⁾⁽¹⁾⁽¹⁾⁽¹⁾⁽¹⁾⁽¹⁾⁽¹⁾⁽¹⁾⁽¹⁾⁽¹⁾⁽¹⁾⁽¹⁾⁽¹⁾⁽	igurator_Example.scfg > igurator_Example.scfg > * = any string, ? = any ch 端子名 P55/PCLBUZ1/SCK00 P56/INTP1 P57/INTP3	ト端子 割り Aracter)		<i>朮</i> −ド機能	機能 設定されていません 設定されていません 設定されていません	方向なしなし]- 	 「の生成 レポート 「すべて シンボリック名 	
^{(成形 米市 7} クロ ボード クロ mart_Confi 子設定 番号 ルタ入力(*) 子番号 72 73 74 75	igurator_Example.scfg > igurator_Example.scfg > * = any string, ? = any ch 端子名 P55/PCLBUZ1/SCK00 P56/INTP1 P57/INTP3 P17/EQ17/CCD01/Tio2	ト端子 割り aracter)	ジ込み 	ポ−ド機能	機能 設定されていません 設定されていません 設定されていません 設定されていません	方向なしなしなし	 值考	 でします。 ドの生成 レポート 「「」」 「」」 すべて シンボリック名 	
^{(成肥 米市子 クロ ボード クロ mart_Confi 子設定 番号 ルタ入力(* 子番号 72 73 74 75 76}	igurator_Example.scfg ン igurator_Example.scfg ン * = any string, ? = any ch 端子名 P55/PCLBUZ1/SCK00 P56/INTP1 P57/INTP3 P17/EO17/CCD01/TI02 P16/EO16/CCD00/TI0	ト 端子 割り aracter)	ジンみ リシンみ の/TxD0 シンプンD0 シングンクロー シングン シングンクロー シングン シングンクロー シングン シング シング	ポード機能 INTP5	機能 設定されていません 設定されていません 設定されていません 設定されていません 設定されていません 1NTP5	方向 なし なし なし レ レ	一 備考	 でします。 ドの生成レポート 一一 すべて シンボリック名 	
^{成肥 米市子} クロ ボード クロ mart_Confi 2 設定 番号 ルタ入力(* 子番号 72 73 74 75 76 77	igurator_Example.scfg ン igurator_Example.scfg ン 端子名 P55/PCLBUZ1/SCK00 P56/INTP1 P57/INTP3 P17/EO17/CCD01/Tl02 P16/EO16/CCD00/Tl0 P15/EO15/SCK20/SCL	ト 端子 割り aracter) //T002/S000 //T001/INTP 20/102/T00	リ込み 0/TxD0 25/SI00/RxD0 12	ポード機能 INTP5	機能 設定されていません 設定されていません 設定されていません 設定されていません 認定されていません 認定されていません 認定されていません 認定されていません	方向 なしななし なし]- 【借考	 でしたしていた。 ドの生成レポート 単一 単二 単二 単二 すべて シンボリック名 	
¹⁰⁰ 1000	igurator_Example.scfg > igurator_Example.scfg > igurator_Example.scfg > 端子名 P55/PCLBUZ1/SCK00 P56/INTP1 P57/INTP3 P17/E017/CCD01/T02 P16/E016/CCD00/T02 P15/VCQUT1/E014/R3	ト 端子 割り aracter) 2/TO02/SOO 1/TO01/INTF 20/TI02/TO00 D2/SI20/SD	ジ込み の/TxD0 25/SI00/RxD0 12 3420/SCLA0/TI03/T003	ポード機能 INTP5 RxD2	機能 設定されていません 設定されていません 設定されていません 設定されていません 認定されていません INTP5 設定されていません RxD2	方向 なし なし なし なし なし よし し て し	(備考)	 で」 ドの生成 レポート 単一二 すべて シンボリック名 	
^{(成形 米市子 クロ ボード クロ mart_Confi 子設定 番号 ルタ入力 (* 子番号 72 73 74 75 76 77 78 79}	igurator_Example.scfg > igurator_Example.scfg > igurator_Example.scfg > 端子名 P55/PCLBUZ1/SCK00 P56/INTP1 P57/INTP3 P17/E017/CCD01/TI02 P16/E016/CCD00/TI02 P16/E016/CCD00/TI02 P15/E015/SCK20/SCL P14//VRE1/E013/TxE P13//WRE1/E013/TxE	ト 端子 割り aracter) 2/T002/S000 //T001/INTF 20/T02/S1200 CD2/S120/SD	ジ込み 0/TxD0 25/Sl00/RxD0 12 1420/SCLA0/Tl03/TO03 AA0/Tl04/TO04	ボード機能 INTP5 RxD2 TxD2	機能 設定されていません 設定されていません 設定されていません 設定されていません INTP5 設定されていません RxD2 TxD2	方向 なななし なし しなし 0		 ドの生成 レポート 単 100 すべて シンボリック名 	
(xt)配 xt)T (xt)T	igurator_Example.scfg ン igurator_Example.scfg ン igurator_Example.scfg ン 端子名 P55/PCLBUZ1/SCK00 P56/INTP1 P57/INTP3 P17/E015/SCK20/SCL P16/E016/CCD00/TI0 P15/E015/SCK20/SCL P13/IVREF1/E013/TxE P13/IVREF1/E013/TxE P12/E112/E012/S000/	ト 端子 割り aracter) //T002/S000 //T001/INTF 20/T02/S020/SD/ 7/S020/SD/ 7/S020/SD/	ジ込み シンス シンス シンス シンス シンス シンス シンス シンス	ボード機能 INTP5 RxD2 TxD2	機能 設定されていません 設定されていません 設定されていません 設定されていません 認定されていません NTP5 設定されていません RxD2 TxD2 設定されていません	方向 なし なし よし よし 日 なし こ なし るし るし	コー 「 「 「 「 「 「 「 「 」 この端子を初期化するコン. この端子を初期化するコン.	 ドの生成 レポート すべて シンボリック名 ・・ ・・ 	
(ct) RE (大) (ct) (ct) (ct) (ct) (ct) (ct) (ct) (ct	igurator_Example.scfg >> igurator_Example.scfg >> igurator_Example	ト 端子 割り aracter) 2/T002/S000 //T001/INTP 20/T02/S020/SD/ 12/S020/SD/ 1xD0/T00L1	0/TxD0 0/TxD0 25/SI00/RxD0 12 AA0/TI04/T004 TxD/INTP5/TI05/T005 kD/SDA0/TI06/T005	ポード機能 INTP5 RxD2 TxD2	機能 設定されていません 設定されていません 設定されていません 設定されていません INTPS 認定されていません RxD2 TxD2 設定されていません 設定されていません のたつまれん	方 向 な し な し な し し る し し て し し る し し て し し る し し て し し る し し こ し し こ し し こ し し こ し し こ し し こ し し こ し し こ し し こ し し こ し し こ し こ し し こ し し こ し し こ こ こ こ こ こ こ こ こ こ こ こ こ	コー	 「の生成 レポート 「「の生成 レポート 「すべて シンボリック名 ・・ 	
witter witter ボード クロ mart_Confi 不設定 番号 ルタ入力(* 7番号 72 73 74 75 76 77 78 79 80 81 82	igurator_Example.scfg > igurator_Example.scfg > igurator_Example.scfg > # = any string, ? = any ch 端子名 P55/PCLBUZ1/SCK00 P56/INTP1 P57/INTP3 P17/E015/SCK20/SCL P14/VCOUT1/E014/R: P13/IVREF1/E013/TxE P12/E112/E012/SC00/ P11/E111/E011/SI00/F P10/E110/E010/SCK00	ト 端子 割り aracter) 2/TO02/SO00 1/TO01/INTF 20/TI02/TO0 D2/S02/SD/ TxD0/TO0LT ixD0/TO0LR ixD0/TO0LR	り込み の/TxD0 25/Sl00/RxD0 12 12 12 12 12 12 12 12 12 12	ポード機能 INTP5 RxD2 TxD2	 機能 設定されていません 設定されていません 設定されていません 設定されていません INTPS 設定されていません RxD2 TxD2 設定されていません 設定されていません 設定されていません 設定されていません 	方 なななし 方 なななし レ し の なな し なし し るし し るし し るし し るし し るし し るし し るし し るし し るし し るし し う む し し ろ し るし し う し ろ し う し ろ し し ろ し し ろ し ろ し し ろ し ろ し し ろ し し ろ し し ろ ろ し ろ し ろ し ろ ろ ろ ろ ろ ろ ろ ろ ろ ろ ろ ろ ろ		 ドの生成 レポート 重 1 1 すべて シンボリック名 	
(xtille) ※11 づ / 1 / 1 / 1 / 1 / 1 / 1 / 1 / 1 / 1	igurator_Example.scfg ン igurator_Example.scfg ン igurator_Example.scfg ン 端子名 P55/PCLBUZ1/SCK00 P56/INTP1 P57/INTP3 P17/E017/CCD01/Tl02 P16/E016/CCD00/Tl02 P15/E015/SCK20/SCL P14/VCOUT1/E014/R3 P13/I/REF1/E013/TxE P12/E112/E012/SO00/ P11/E111/E011/Sl00/F P10/E110/E010/SCK00 P90	ト 端子 割り aracter) 2/TO02/SO00 1/TO01/INTF 20/TI02/TO0 c/D2/SI20/SD 2/SO20/SD/ TxD0/TOOLT xD0/TOOLR ySCL00/TI07	り込み り込み の/TxD0 P5/SI00/RxD0 12 12 12 12 12 12 12 12 12 12	ポード機能 INTP5 RxD2 TxD2	機能 設定されていません 設定されていません 設定されていません 設定されていません 認定されていません 認定されていません 設定されていません 設定されていません 設定されていません 設定されていません	方 ななななし なし なし なし なし なし し なし なし なし	コー 備考 この端子を初期化するコン. この端子を初期化するコン.	 「の生成 レポート 「「の生成 レポート 「すべて シンボリック名 ・・ 	
(x限 メポゴ / クロ ボード クロ (オポード クロ (オポード クロ (オポード) クロ (オポー)	igurator_Example.scfg > igurator_Example.scfg > igurator_Example.sc	ト 端子 割り aracter) //TO02/SO00 //TO01/INTF 20/TI02/TO0 /2/SO20/SDA TxD0/TOOLT xD0/TOOLR /SCL00/TI07	リ込み り込み 0/TxD0 25/SI00/RxD0 12 1420/SCLA0/TI03/T003 AA0/TI04/T004 TxD/INTP5/TI05/T005 1xD/SDA00/TI06/T006 //T007	ポード機能 INTP5 RxD2 TxD2	 機能 設定されていません 設定されていません 設定されていません 設定されていません RxD2 TxD2 設定されていません 設定されていません 設定されていません 設定されていません 設定されていません 設定されていません 設定されていません 設定されていません 	方 なななな 向 し なななな し なし なななななな し な ななななな ななななな な し	コー 信考 この端子を初期化するコン、 この端子を初期化するコン、	 「の生成レポート 「」 すべて シンボリック名 ・・ 	
KRE ■ MT 7 KT F 2 T	igurator_Example.scfg > igurator_Example.scfg > igurator_Example.scfg > 端子名 P55/PCLBUZ1/SCK00 P56/INTP1 P57/INTP3 P17/EO15/SCK20/SCL P14/VCOUT1/EO14/R3 P13/IVREF1/EO13/TxC P12/E112/EO12/SO00/ P11/E111/EO11/SI00/F P10/E110/EO10/SCK00 P90 P91 P92	ト 端子 割り aracter) //T002/S000 //T001/INTF 20/T001/INTF 20/T02/S020/SD4 TxD0/T00L x>D0/T00LR ySCL00/T107	り込み り込み り込み シタン ない 大しの 大しの 大しの 大しの 大しの 大しの 大しの 大しの	ボード機能 INTP5 RxD2 TxD2	機能 設定されていません 設定されていません 設定されていません 設定されていません 認定されていません 認定されていません 設定されていません 設定されていません 設定されていません 設定されていません 設定されていません 設定されていません 設定されていません	方向 方 な な な な し な し な し し ひ し ひ な な な な な な な な な な な な		 で」 ドの生成 レポート ゴイて シンボリック名 ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	

図 4-63 端子番号タブにジャンプ



4.5.5 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) 端子ページで、[ボードの設定をエクスポート] 崎 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。

*Smart_Configurator_Example.scfg	×						•	
Smart_Configurator_Example/Smart_C	onfigurator_Exa	ample.scfg					じょうしょう いっぽう いっぽう いっぽう いっぽう いっぽう いっぽう いっぽう いっぽ	±
ソフトウェアコンポ 🕞 📄 🖧 趧	端子機能						🥺 🖪 🔛 🗠	4
フィルタ文字列を入力	フィルタ入力) (* = any strir	ng, ? = any characte	r)			<u>इ</u> र्रर ५	1
🗸 💑 r_bsp	使用する	機能	PIOR	端子割り当て	端子番号	方向	備考	•
💟 r_bsp		INTPO		P137/EI137/INTP0	/ 29	1	この端子を初期化す	
		INTP1	PIOR5	P46/INTP1/TI05/TO05	/ 16	1		
		INTP2			〃 設定されていませ	なし		
> A/D_J//9		INTP3	PIOR5		〃 設定されていませ	なし		
> 🔤 1.29-7.70.91 V		INTP4	PIOR5		〃 設定されていませ	なし		
		INTP5	PIOR4	✓ 設定されていません	✓ 設定されていませ	なし		1
	<						>	
端子機能端子番号	×.1 48 → (†)11							
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 4-64 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、端子設定を CSV 形式で 保存します。

4.5.6 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。

t_Configurator_Example/Smart_	Configurator_Exa	ample.scfg					○ コードの生成 レポートの
アトウェアコンポ 🕀 🖻 ↓ª 🖁	端子機能						- 🔁 🖪 🔛
イルタ文字列を入力	フィルタ入力) (* = any strin	ng, ? = any characte	r)			すべて
🗸 💑 r_bsp	使用する	機能	PIOR	端子割り当て	端子番号	方向	備考
💓 r_bsp		INTPO		/ P137/EI137/INTP0	/ 29	1	この端子を初期化す
/ 査 割り込みコントローフ		INTP1	PIOR5	P46/INTP1/TI05/TO05	/ 16	1	
Config_INIC		INTP2			/ 設定されていませ	なし	
		INTP3	PIOR5		/ 設定されていませ	なし	
🐴 129-Λμ•91 Υ		INTP4	PIOR5		∥ 設定されていませ	なし	
		INTP5	PIOR4		/ 設定されていませ	なし	
	<						>

図 4-65 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。



4.5.7 ボード端子設定情報を使用した端子設定

ボードの初期端子設定を一括で行えます。端子を一括で設定するには、以下の手順で行います。

- (1) [ボード] ページで、[カスタムユーザーボード] 以外のボード設定情報を選択します。
 (4.1.2 ボード選択错误!未找到引用源。参照)
- (2) MCU/MPU パッケージで [ボード機能] を選択します。 (ボードの初期端子設定が参照できます)
- (3) [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。
- (4) [ボードの初期端子割り当ての設定] ダイアログが開いたら [すべて選択] をクリックしてください。
- (5) [OK] ボタンをクリックします。

#Smart_Configurator_Example.scfg ×	(2) -	🖥 MCU/MPUパッケージ × 💿 Developer Assist Browser (2)-ー 🗖
端子設定	◎ ボードの初期様子割り当て × □ トの生成	🔚 🔺 🔎 🖻 🖻 フィルタ文字列を入力 🔤 ボード機能 👻
ハードウェアリソ ① □ 見 品 端子機能 フィルク文型のあるカ	ポードの初期端子割り当ての設定 (4)	
	割り当て 第子機能 すべて選択 「方向 方向 ANI4 「方向 方向 「方向 ANI5 「道沢解除 「れてしませ なし ANI6 「 「 ANI7 「 「 ANI8 「 「 ANI9 「 「 ANI10 「 「 ANI11 「 「 ANI13 「 「 ANI14 「 「 ANI2 「 「 ANI2 「 「	
	ΔNDS > (5) ОК 4+УУЕШ	× D@I

図 4-66 ボードの初期端子設定

端子設定を一度に設定しない場合は、手順(4)で個別に設定してください。

4.5.8 端子のフィルタ機能

「端子」ページの[端子機能]タブ、[端子番号]タブでフィルタ範囲を指定し、より簡単に参照することができます。

ソフトウェアコンポーネント 🗈 🖻 🔩 🔜	端子機能					રા		<u>n</u> 2
フィルタ文字列を入力	フィルタ入力 (* =	any string, ? = any	y character)			すり	×7	~
	使用する 機能	발 PIOR	端子割り当て	端子番号	方向	備 <mark>す/</mark> 備 - 機i PIC	べて 能 DR	
Config_ADC V	<					·····································	子割り当し 子番号 向	
端子機能 端子番号 概要 ボード クロック システム コンポーネント 端子	そ割り込み					備: 	考 いト	

図 4-67 [端子機能] タブのフィルタ

(በአልታ (* = any string, ? = any character)					すべて
行番号	端子名	デフォルト機能	機能	方向	備考	すべて コン 端子番号
1	P142/SCK30/SCL30		設定されていません	なし		端子名
2	P141/PCLBUZ1/INTP7		設定されていません	なし		デフォルト機能
						機能

図 4-68 [端子番号] タブのフィルタ



4.5.9 端子エラー/警告の設定

[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端子設定問題の表示方法を制御できます。制御を変更したい場合は、[新規コンポーネント] ダイアログで [基本設定] リンクをクリックし、[設定] ダイアログを表示します。次に、[スマート・コンフィグレータ] の [端子エラー/警告] を選択し、設定を変更します。



図 4-69 端子エラー/警告の設定

例:[ソフトウェアコンポーネント]の設定を「情報」から「エラー」に変更。



図 4-70 ソフトウェアコンポーネントのエラー

4.6 割り込み設定

[割り込み]ページには、各ベクタ番号によりすべての割り込みが表示され、 [コンポーネント] ページで選択した周辺モジュールの割り込みを確認・設定できます。 コード生成タイプのコンポーネントで割り込みを使用すると、割り込みの状態が「使用中」に変わります。

- (1) 使用中の割り込みのみを表示するには、[図設定した割り込みの表示)] ボタンをクリックしてください。
- (2) グループ割り込みは、割り込みテーブルでは折りたたまれます。グループ割り込みリストの割り込み を見るには、[展開] ボタンをクリックしてください。

割り	mart_Configurat 込み設定	or_Example.sctg ×						じ コードの生成	<u>。</u> レポートの生!
設知	官済み割り込みべ	79						(1) 💌
	フィルタ文字列を	入力				ベクタ番号			~
	ベクタ番号	ベクトルテーブルアドレス	割り込み	割り込み要求元	周辺機能	優先レベル	状態	バンク指定	備考
	3	0000AH	INTP1	Pin input edge detection	INTC	レベル3(低優先順位)	使用中	なし	
2)	> 18	00028H	INTSRE1/INTTM03H			レベル3(低優先順位)	使用中	なし	
1	24	00034H	INTAD	End of A/D conversion	ADC	レベル3(低優先順位)	使用中	なし	
	<								>

図 4-71 [割り込み] ページ

4.6.1 割り込み優先レベルの設定

[割り込み]ページの割り込み優先レベルは、以下の手順で変更できます。

- (1) 優先レベルを変更したい割り込みを表示します。
- (2) 優先レベルセルをクリックし、ドロップダウンリストから割り込み優先レベルを設定します。

🔅 *Smart_	_Configurate	or_Example.scfg \times							- 0
割り込み	み設定							じ コードの生成 レ	
設定済み	り割り込みべく	79							×
71	ルタ文字列を	入力			Ň	クタ番号			\sim
ベク	79番号	ベクトルテーブルアドレス	割り込み	割り込み要求元	周辺機能	優先レベル	状態	バンク指定	備考
	3	0000AH	INTP1	Pin input edge detection	INTC	レベル3(低優先順位)	使用中	なし	
>	18	00028H	INTSRE1/INTTM03H		(2)	レベル3(低優先順位)	使用中	なし	
	24	00034H	INTAD	End of A/D conversion	ADC	レベル3(低優先順位) ∨	使用中	なし	
						レベル0(高優先順位)			
						レベル1			
<						レベル2(低優先順位)			>
		_							
概要 ボート	ド クロック シ	マステム コンポーネント 端子 割	り込み						

図 4-72 割り込み設定



4.6.2 割り込みバンクの設定

[割り込み] ページの割り込みバンクは、以下の手順で変更できます。

- (1) バンクを変更したい割り込みを表示します。
- (2) バンク指定セルをクリックし、ドロップダウンリストからバンク(None / 1 / 2 / 3)を設定します。
- (3) 優先度の異なる複数の割り込みに対して同じバンクを指定すると警告マークが表示され、[備考] と [Configuration Problems] に警告メッセージが表示されます。必要に応じて再設定してください。

みべクタ									
イルタ文字列	川を入力					(1)		ベクタ番号	
ベクタ番号	ベクトルテーブルアドレス	割り込み	割り込み要求元	周辺機能	優先レベル	状態	バンク指定	備考 (3)	
0	00004H	INTWDTI	Watchdog timer interval	WDT	レベル0(高優先順位)	使用中	701.	(-)	
1	00006H	INTLVI	Voltage detection	LVD	レベルコ	使用中	A 1	優先度の異なる複数の割り込み機能に対して、1つのレジスタバンクを指	旨定することはできません。
2	00008H	INTPO	Pin input edge detection	INTC	レベル2	使用中	14U		
3	0000AH	INTP1	Pin input edge detection	INTC	レベル3(低優先順位)	使用中	tr!		
4	0000CH	INTP2	Pin input edge detection	INTC	レベル3(低優先順位)	使用中	<u>a</u> 1	優先度の異なる複数の割り込み機能に対して、1つのレジスタバンクを指	旨定することはできません。
5	0000EH	INTP3	Pin input edge detection	INTC	レベル3(低優先順位)	使用中	711		
6	00010H	INTP4	Pin input edge detection	INTC	レベル3(低優先順位)	使用中	なし、い		
7	00012H	INTP5	Pin input edge detection	INTC	レベル3(低優先順位)	(2)	なし		
8	00014H	INTST2/INTCSI20/IN			レベル3(低優先順位)	(2)	1		
> 9	00016H	INTSR2/INTCSI21/IN			レベル3(低優先順位)		2		
> 10	00018H	INTSRE2/INTTM11H			レベル3(低優先順位)		3		
ード クロック ィグレーション	システム コンポーネント 茸 チェック ×	岩子 <u>割り込み</u>							7 8
12 warning:	s, 45 others		^					71	
明								<u>22</u>	
明込み (1 エ	頁目)								

図 4-73 割り込みバンク設定

割り込みバンクの設定は、以下のようにコードに反映されます。

(1) CCRLの場合、コンポーネントの{ConfigurationName}_user.cファイルに反映されます。



図 4-74 CCRL 割り込みバンクコード

(2) LLVMの場合、<ProjectDir>\src\smc_gen\general\r_cg_interrupt_handler.h ファイルに反映されます。

```
/*
 * INT_P0 (0x8)
 */
void r_Config_INTC_intp0_interrupt(void) __attribute__ ((interrupt(bank=RB1)));
```

図 4-75 LLVM 割り込みバンクコード

実際に生成されるコード仕様は、コンパイラによって異なります。 詳細については、対応する IDE の ユーザーガイドを参照してください。



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_Configurator_Example/Sma	rt_Configurator_Example.scfg - e² studio	0		- 🗆	×
ファイル(F) 編集(E) ナビゲート(N) 検索(A) プロジ	バェクト(P) Renesas Views 実行(R) ウ	ィンドウ(W) ヘルプ(H)			
🔚 🛞 • 🔦 • 🕸 • 💁 •	プロジェクトを開く(E)		Q 🛛 🔛 🖬 C/C++ 📓	🛛 スマート・コンフィ	ィグレータ
 	プロジェクトを閉じる(S)	cample.scfg ×		-	
📄 🗸 👺 Smart_Configurator_Example [H	すべてビルド(A) Ctrl+A	lt+B		a	
> 🛐 Includes	ビルド構成	>	コードの生き	成 レポートの生き	戎
> 🗁 output	プロジェクトのビルド(B) Ct	rl+B		(?)	^ <u>_</u>
> 🧀 trash	ワーキング・セットのビルド(W)	>		0	(?)
Smart_Configurator_Example.s	クリーン(N)				
Smart_Configurator_Example F	自動的にビルド(M)	ックすると、スマート・コンフィグレータの機能を			
> (7) Developer Assistance	C/C++ インデックス(i)	とかできます。	Application Code		L
62	すべての依存関係を更新 AI	t+D		2 B	
	Change Device	- <u>フィグレータの紹介</u>	Software Components		
	Change Toolchain Version		Middleware &		
(B)	C/C++ Project Settings Ctrl+A	It+P tc	RTOS		
	プロパティ(P)	トレート ション・ 「「「「「」」 「「」」 「「」」 「「」」 「「」」 「「」」 「「」」	Device Drivers	<u>e</u> .	
	ु व	ることかできます。		5	
		ロロドキョイント		ġ.	
			NICU Hardware		
		プリケーションノート			
	2				
		*			
	▼ 現在の設定状態				¥
<		7 システム コンボーネント 端子 割り込み			

図 4-76 e² studioの[Change Device]

(2) Target Board からボードを選択すると、デバイスは自動的に選択されます。



図 4-77 Target Board の選択

デバイスを直接選択したい場合は、ターゲット・デバイスからデバイスを選択します。

(例: RL78/G23 R7F1000GPGxFBを選択)



					~		
hange Device Select the new de	evice for Smart_Co	nfigurator_Example					
Current Device: R	7F100GSN×FB						
Current Board: RL	78G23-128p_FastP	rototypingBoard_LED					
Target Board:	Custom				~		
ターゲット・デバイス	R7F100GSNxFB						
			7/11	スのアンビ	<u>99</u>		
		Vou can filter devices by regular	expression				
		Vou can filter devices by regular Search Device	expression				
		Vevice Selection You can filter devices by regular: Search Device Device 	expression RAM	RC	DM	Pin	
		Device Selection You can filter devices by regular. Search Device Device RL78 - 023 80pin ▼RL78 - 023 100pin R77100GP6xFB	RAM 16 KB	RC	DM B KB	Pin 100	
		Device Selection You can filter devices by regular. Search Device Device RL78 - G23 80pin ▼ RL78 - G23 00pin R7F100GPGxFB R7f100GPHxFB	RAM 16 KB 20 KB	RC 124 195	DM B K B 2 KB	Pin 100 100	
		Device Selection You can filter devices by regular. Search Device Device No RL78 - G23 80bin ▼ RL78 - G23 80bin R7F 100GPGxFB R7F 100GPJxFB	RAM RAM 16 KB 20 KB 24 KB	RC 12/ 19/ 25/	DM 8 KB 2 KB 5 KB	Pin 100 100 100	
		Device Selection You can filter devices by regular. Search Device Device > RL78 - G23 80pin ▼ RL78 - G23 100pin R7F 100GPKxFB R7F 100GPKxFB R7F 100GPKxFB R7F 100GPKxFB	RAM 16 KB 20 KB 24 KB 32 KB	RC 12/ 19/ 25/ 38	DM B KB 2 KB 5 KB 4 KB	Pin 100 100 100 100	
		Device Selection You can filter devices by regular. Search Device Device R178 - 023 80pin R178 - 023 100pin R7F 1005P6xF8 R7F 1005P6xF8 R7F 1005P1xF8 R7F 1005P1xF8 R7F 1005P1xF8 R7F 1005P1xF8 R7F 1005P1xF8	RAM 16 KB 20 KB 24 KB 32 KB 48 KB 49 KB 40 KB 4	RC 12/ 19/ 25/ 38	DM B KB 2 KB 5 KB 4 KB 2 KB	Pin 100 100 100 100 100	
		Device Selection You can filter devices by regular. Search Device Device R178 - 623 80pin ▼ R178 - 621 00pin R7F 100GP4xF8 R7F 100GP4xF8	RAM 16 KB 20 KB 24 KB 22 KB 32 KB 48 KB 4	RC 12/ 19/ 25/ 38- 51/ 76/	2 KB 2 KB 5 KB 4 KB 2 KB 8 KB	Pin 100 100 100 100 100 100	
		Device selection You can filter devices by regular. Search Device Device RT8 - G23 80pin RT60GPGxFB R7F100GPLxFB	RAM RAM 16 KB 20 KB 24 KB 32 KB 48 KB 48 KB	RC 124 199 255 384 511 766	DM 8 KB 2 KB 5 KB 4 KB 2 KB 3 KB	Pin 100 100 100 100 100 100	

図 4-78 ターゲット・デバイスの 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。

 ๑ มวะกริยมชั 	_		×
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、' クリックします。	終了'	* - 2 ~	
検出された問題		ł	J 🗘
This change cannot be undone. Please make sure you backup this project before con Section 2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.	ntinui	ng.	
使用可能なコンテキスト情報はありません			
(ア) (N) > 終了(F)		キャンセ	IL

図 4-79 検出された問題

表 4-1 検出された問題のメッセージ

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	スマート・コンフィグレータがサポートしていな
グレータでサポートされていません	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	デバイスを変更すると変更前に戻すことができま
sure you backup this project before	せんので、プロジェクトのバックアップ後に実行
continuing.	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。

圆 ปวะวรปาวช์	—		×
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		₽ 0	7 -
 ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ <p< td=""><td></td><td></td><td></td></p<>			
> ☑ 🔮 Build Settings ☑ 🛃 Project Files ☑ 🍨 Smart Configurator			
使用可能なプレビューはありません			
(?) < 戻る(B) 次へ(N) > 終了(F)		キャンセ	zll

図 4-80 実行される変更項目確認

(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_Configurator_Example.scfg 🗙			
概説		で コードの生成	└ぱートの生成
製品ドキュメント ユーザーマニュアルとリリースノート 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 使用しているボードノデバイス: R7F100GPGxFB (ROM size: 1)	28KB. RAM size: 16KB	Pin count: 100)	
生成先ロケーション (PROJECT_LOC¥): src¥smc_gen 使用しているコンポーネント:		編集	
コンポーネント ^	バージョン	設定	
Board Support Packages v1.60 (r_bsp)	1.60	r_bsp(使用中)	
	6		

図 4-81 デバイス更新確認

(6) コンソースにデバイス変更結果レポートが出力されます。

עעב 🗐	🖹 📑 📴 📑 🖻 🗕 🖬 🗖 🗖 🗖
スマート・コンフィグレータ出力	
M03000003: クロック設定の <u>R7F100GSNxFB</u>	^
M00000003: レポート生成:output\migration report 2021-03-10-16-05-34.html	
	~
κ	>
図 4_92 記字亦換フラークフレギー	_ L

図 4-82 設定変換ステータスレポート



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、リソースの競合に関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。なお、競合が発生していてもコードは生成できます。

5.1 リソースの競合

同じリソース(例: S12AD1)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				○ 2-ドの生成 レポートの生成	
コンポーネント 🚵 🛃 🖧 🗄 🕀	設定			(i) ^	
 ・ ・	コンパレ-タ動作設定 ● 停止	○許可			
	分解能設定 ① 10ビット	○ 8ビット	○ 12분ット		
Config_ADC_Duplicate	VREF(+) 設定 ● VDD	○ AVREFP	○ 内部基準電圧	~	
概要 ボード クロック システム コンポーネント 端子	子 割り込み				
🔝 Ξンフィグレーションチェック 🗙				78 - 8	
6 errors, 0 warnings, 0 others					
記述/説明		^		型	
✔ 🔕 割り込み (2 項目)					
8 E04010005: INTAD (Config_ADC_Dup	licateで設定)と次の割り込みのべ	クタが競合しています : INTAD (Config_ADCで	?設定).	割り込み	
◎ E04010005: INTAD (Config_ADCで設定) と次の割り込みのパクタが競合しています: INTAD (Config_ADC_Duplicateで設定).					
✔ 🔕 周辺機能 (2 項目)					
6 E04010001: 周辺機能 A/Dコンパータ (Config_ADC_Duplicateで設定) は次の設定で使用されています: Config_ADC.					
◇ ◎ 端子 (2項目)					
E04010003: ANIO (Config_ADC_Duplic	cate ("設定) か使用する端子と次の 、 が使用する端子と次の	り端子か競合しています:ANIO(Config_ADC	(設定).	「「「」	
V E04010003: ANI0 (Config_ADCで設定))か1史用する端子と次の端子が競合	雪しています:ANI0 (Config_ADC_Duplicate	(設定).	师子	

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。

*Smart_Configurator_Example.scfg ×	_							=	' 🗆
端子設定								^で コードの生成 レポートの:	生成
ソフトウェアコンポーネント 🗉 🖻 🖧 🔜	端子機能							🤣 😐 📑 🔤	4
フィルタ文字列を入力	<u>7</u> ィルタスプ	り (* = any stri	ng, ? = any o	character)				すべて	\sim
> 📩 r bsp	使用する	機能	PIOR	端子割り当て	端子番号	方向	備考		^
		🔕 ANIO		P20/ANI0/AVREFP/EI20	/ 117	1	複数の端子機能が同一端	子に割り当てられています	
Config_ADC		ANI1		🥖 設定されていません	〃 設定されていません	なし			
		ANI2		✓ 設定されていません	✓ 設定されていません	なし			
& Config_PORT		ANI3		/ 設定されていません		なし			~
	<							>	
端子機能 端子番号 暦要 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 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。

INDATUP Say Find Say Say		7 🕴 🗖
3 errors, 0 warnings, 0 others		
記述/說明	型	
▼ 🔞 端子 (3 項目)		
😢 E04010003: ANIO (Config_ADCで設定) が使用する端子と次の端子が競合しています : P20 (Pin Allocatorで設定)、P20 (Config_PORTで設	定). 端子	
😣 E04010003: P20 (Config_PORTで設定) が使用する端子と次の端子が競合しています : ANIO (Pin Allocatorで設定)、ANIO (Config_ADCで割	定). 端子	
😢 E05000010: 端子 117を複数の機能で使用できません。端子 117 に P20, ANIO 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

エラーマークが表示されているツリー・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。

ソフトウェアコンポーネント	🕀 🕒 🞝 👪	端子機能	
フィルタ文字列を入力		フィルタ入力] (* = any string, ? =
 ✓ 益 r_bsp ● r_bsp ✓ 益 A/Dコンパータ ⑥ Config_ADC 		使用する 	機能 ※ P20 CCD00 CCD01
マ 赤 ボート Config_PORT			CCD02
	すべて割り当て	47 P.	.CD03 .CD04
	競合の解決	肝防	CD05 CD06
		<	CDort
端子機能 端子番号			
概要 ボード クロック システレ	」 コンポーネント 端子	割り込み	

図 5-4 端子競合の解決

選択されたノードの端子機能は、他の端子に再度割り当てられます。



6. ソースの生成

ソースの生成は、コンフィグレーションチェックビューで競合が発生していても生成できます。

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。

🌼 *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

図 6-1 ソースファイルの生成

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】作成したソースファイルを sms_gen フォルダに入れると、ソースコード生成時に消去されます。



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 コード生成場所の変更

(1) コードの生成場所を変更するには、[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。



図 6-3 生成先ロケーションの編集

(2) [フォルダの選択] ダイアログで、コード生成用の空のフォルダを選択するか、新しいフォルダを作成 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:				
Smart_Configurator_Example				
→ Goutput				
> ≽ smc_gen				
🗿 新規フォルダ				×
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	JL	

図 6-4 生成先フォルダの選択



(3) [コードの生成] 🔂 ボタンをクリックすると、ソース コードは新しい場所に生成されます。 [概要] ページで、現在のコード生成先の場所を確認できます。



図 6-5 新しいコード生成場所に生成



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図 6-6 生成ファイルの構成とファイル名 に示します。なお、main() 関数は e² 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。

"r_xxx" は RL78 Software Integration System Modules 名、 "*ConfigName*" はコンポーネント設定で形成 されたコンフィグレーション名、 "*Project name*" は e² studio で設定されたプロジェクト名を示します。



図 6-6 生成ファイルの構成とファイル名



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
general		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。
	$r ca xxx.h^{(*1)}$	このファイルは SFR レジスタを設定するためのマクロ
		定義を含みます。
	r smc entry.h	このファイルは常時生成されます。
		このファイルには、プロジェクトに追加される CG ド
		ライバのヘッダファイルが含まれます。
		ユーザーが追加するソースファイルで CG ドライバの
		関数を使用する場合、このファイルのインクルードが
		必要です。
	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_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_cg_systeminit.c	このファイルは常時生成されます。
		全コンボーネントの Create() 関数を含みます。周辺機
	(*4)	能の初期化に使用しまり。
	r_cg_xxx_common_user.c ⁱ	このファイルは使用する周辺機能の共通の割り込み API 関数を含みます。
	r_cg_xxx_common.c ^(*1)	このファイルは対応する周辺機能を使用する場合に生 成されます。
	r_cg_xxx_common.h ^(*1)	このファイルは対応する周辺機能を使用する場合に生
		成されます。
	r_cg_interrupt_handlers.h(*2)	このファイルには、すべての割り込みルーチンの宣言
		が含まれています。構成が作成されていない場合、す
		べての割り込みルーチンがデフォルトのルーチンで
		す。特定の構成が作成された場合、この構成の対応す
		る割り込みルーナンの宣言かテフォルトのルーナン宣 ミに罢き施わります
	(*2)	「「「「」」」」「「」」」」」」」」」
	r_cg_inthandier.c ⁽²⁾	レーチン定義が含まれています。
	r cg vect table c ^(*2)	このファイルには、すべての割り込みルーチンエント
		リアドレスを含む割り込みベクタテーブルが含まれて
		います。構成が作成されていない場合、すべての割り
		込みルーチンエントリアドレスがデフォルトです。 特
		定の構成が作成された場合、この構成の対応する割り
		込みルーチンエントリアドレスがデフォルトのルーチ
		ンエントリアトレスに直き換わります。
r_bsp		このフォルダは常時生成されます。
		以下を含む複数のサフフォルタ(board, doc, mcu)か ら構成されます。
		main() 実行前に MCU を起動する初期化コード
		(例:スタックのセットアップ、メモリの初期化)
		・ iodefine.h (mcu フォルダ) にあるすべての SFR レ
		ジスタの定義
		・ r_bspのアブリケーションノート(doc フォルダ)
		フロジェクトで使用されるデバイスの r_bsp.h を含む、
		platform.n もこのノオルダに生成されます。



フォルダ	ファイル	説明
r_xxx/		このフォルダは、プロジェクトに追加された RL78
rm_ <i>xxx</i> ^(*1)		Software Integration System モジュール用に生成されま
		す。
		・ doc フォルダ : アプリケーションノート
		・ r_xxx.c/rm_xxx.c ^(Note*1) : ソースファイル
		・ r_xxx.c/rm_xxx.h ^{(Note*1):} ヘッダファイル
		・ r_xxx_api.h/rm_xxx_api.h ^{(Note*1) :} すべての API とイ
		ンターフェース定義のリスト
r config		このフォルダは常時生成されます。
		MCU パッケージ、クロック、割り込み等のコンフィグ
		レーションヘッダファイルを含みます。
	r bsp config.h	このファイルは常時生成されます。
		クロック初期化と他の MCU に関連する r_bsp の設定を
		含みます。いくつかの MCU 関連の設定はスマート・コ
		ンフィグレータが生成し(例:パッケージタイプ)、
		他の設定(例:スタックサイズ)はユーザーが手動で
		設定します。
	r_bsp_config.inc	このファイルは常時生成されます。
		構成ヘッダファイルを生成します。
	r_xxx_config.h/rm_xxx_config.h	このファイルは、プロジェクトに追加されるすべての
	(*1)	RL78 Software Integration System ドライバ/ミドル
		ウェアの構成ヘッダファイルです。
r_pincfg	Pin.h	このファイルは常時生成されます。
		端子のシンボルをサポートするために生成され、
		smc_entry.h に含まれています。
	Pin.c	このファイルは常時生成されます。
		[端子] ページで有効になっている端子設定が生成されま
		す。PIOR を設定する必要のない端子設定を生成するだ
		す。PIOR を設定する必要のない端子設定を生成するだけです。
	r_xxx_pinset.c	す。PIOR を設定する必要のない端子設定を生成するだけです。 このファイルは、RL78 Software Integration System の
	r_ <i>xxx</i> _pinset.c	す。PIOR を設定する必要のない端子設定を生成するだけです。 このファイルは、RL78 Software Integration System の 端子設定ソースファイルです。
	r_xxx_pinset.c r_xxx_pinset.h	す。PIOR を設定する必要のない端子設定を生成するだけです。 このファイルは、RL78 Software Integration Systemの 端子設定ソースファイルです。 このファイルは、RL78 Software Integration Systemの
	r_xxx_pinset.c r_xxx_pinset.h	す。PIOR を設定する必要のない端子設定を生成するだけです。 このファイルは、RL78 Software Integration System の 端子設定ソースファイルです。 このファイルは、RL78 Software Integration System の 端子設定ヘッダファイルです。
{ConfigName}	r_ <i>xxx</i> _pinset.c r_ <i>xxx</i> _pinset.h	す。PIOR を設定する必要のない端子設定を生成するだけです。 このファイルは、RL78 Software Integration System の 端子設定ソースファイルです。 このファイルは、RL78 Software Integration System の 端子設定ヘッダファイルです。 このフォルダは、プロジェクトに追加される CG ドラ
{ConfigName}	r_xxx_pinset.c r_xxx_pinset.h	す。PIOR を設定する必要のない端子設定を生成するだけです。 このファイルは、RL78 Software Integration Systemの 端子設定ソースファイルです。 このファイルは、RL78 Software Integration Systemの 端子設定ヘッダファイルです。 このフォルダは、プロジェクトに追加される CG ドラ イバ用に生成されます。
{ConfigName}	r_xxx_pinset.c r_xxx_pinset.h	す。PIOR を設定する必要のない端子設定を生成するだけです。 このファイルは、RL78 Software Integration System の 端子設定ソースファイルです。 このファイルは、RL78 Software Integration System の 端子設定ヘッダファイルです。 このフォルダは、プロジェクトに追加される CG ドラ イバ用に生成されます。 このフォルダ内の API 関数には、 <i>ConfigName</i> (設定
{ConfigName}	r_xxx_pinset.c r_xxx_pinset.h	す。PIOR を設定する必要のない端子設定を生成するだけです。 このファイルは、RL78 Software Integration System の 端子設定ソースファイルです。 このファイルは、RL78 Software Integration System の 端子設定ヘッダファイルです。 このフォルダは、プロジェクトに追加される CG ドラ イバ用に生成されます。 このフォルダ内の API 関数には、 <i>ConfigName</i> (設定 名)を含んだ名称がつけられています。
{ConfigName}	r_ <i>xxx</i> _pinset.c r_ <i>xxx</i> _pinset.h { <i>ConfigName</i> }.c	す。PIOR を設定する必要のない端子設定を生成するだけです。 このファイルは、RL78 Software Integration System の 端子設定ソースファイルです。 このファイルは、RL78 Software Integration System の 端子設定ヘッダファイルです。 このフォルダは、プロジェクトに追加される CG ドラ イバ用に生成されます。 このフォルダ内の API 関数には、 <i>ConfigName</i> (設定 名)を含んだ名称がつけられています。 このファイルは、ドライバを初期化する関数
{ConfigName}	r_ <i>xxx</i> _pinset.c r_ <i>xxx</i> _pinset.h { <i>ConfigName</i> }.c	す。PIOR を設定する必要のない端子設定を生成するだけです。 このファイルは、RL78 Software Integration System の 端子設定ソースファイルです。 このファイルは、RL78 Software Integration System の 端子設定ヘッダファイルです。 このフォルダは、プロジェクトに追加される CG ドラ イバ用に生成されます。 このフォルダ内の API 関数には、 <i>ConfigName</i> (設定 名)を含んだ名称がつけられています。 このファイルは、ドライバを初期化する関数 (R_ConfigName_Create)、ドライバに特有な操作、
{ConfigName}	r_xxx_pinset.c r_xxx_pinset.h { <i>ConfigName</i> }.c	す。PIOR を設定する必要のない端子設定を生成するだ けです。 このファイルは、RL78 Software Integration System の 端子設定ソースファイルです。 このファイルは、RL78 Software Integration System の 端子設定ヘッダファイルです。 このフォルダは、プロジェクトに追加される CG ドラ イバ用に生成されます。 このフォルダ内の API 関数には、 <i>ConfigName</i> (設定 名)を含んだ名称がつけられています。 このファイルは、ドライバを初期化する関数 (R_ConfigName_Create)、ドライバに特有な操作、 例えばスタート(R_ConfigName_Start)やストップ
{ConfigName}	r_xxx_pinset.c r_xxx_pinset.h { <i>ConfigName</i> }.c	す。PIOR を設定する必要のない端子設定を生成するだけです。 このファイルは、RL78 Software Integration System の 端子設定ソースファイルです。 このファイルは、RL78 Software Integration System の 端子設定ヘッダファイルです。 このフォルダは、プロジェクトに追加される CG ドラ イバ用に生成されます。 このフォルダ内の API 関数には、 <i>ConfigName</i> (設定 名)を含んだ名称がつけられています。 このファイルは、ドライバを初期化する関数 (R_ConfigName_Create)、ドライバに特有な操作、 例えばスタート(R_ConfigName_Start)やストップ (R_ConfigName_Stop)を実行する関数を含みます。
{ConfigName}	r_xxx_pinset.c r_xxx_pinset.h { <i>ConfigName</i> }.c { <i>ConfigName</i> }_user.c	す。PIOR を設定する必要のない端子設定を生成するだけです。 このファイルは、RL78 Software Integration System の 端子設定ソースファイルです。 このファイルは、RL78 Software Integration System の 端子設定ヘッダファイルです。 このフォルダは、プロジェクトに追加される CG ドラ イバ用に生成されます。 このフォルダ内の API 関数には、 <i>ConfigName</i> (設定 名)を含んだ名称がつけられています。 このファイルは、ドライバを初期化する関数 (R_ConfigName_Create)、ドライバに特有な操作、 例えばスタート(R_ConfigName_Start)やストップ (R_ConfigName_Stop)を実行する関数を含みます。 ドライバの初期化(R_ConfigName_Create)の後に追
{ConfigName}	r_xxx_pinset.c r_xxx_pinset.h { <i>ConfigName</i> }.c { <i>ConfigName</i> }_user.c	す。PIOR を設定する必要のない端子設定を生成するだけです。 このファイルは、RL78 Software Integration System の 端子設定ソースファイルです。 このファイルは、RL78 Software Integration System の 端子設定ヘッダファイルです。 このフォルダは、プロジェクトに追加される CG ドラ イバ用に生成されます。 このフォルダ内の API 関数には、 <i>ConfigName</i> (設定 名)を含んだ名称がつけられています。 このファイルは、ドライバを初期化する関数 (R_ConfigName_Create)、ドライバに特有な操作、 例えばスタート(R_ConfigName_Start)やストップ (R_ConfigName_Stop)を実行する関数を含みます。 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
{ConfigName}	r_xxx_pinset.c r_xxx_pinset.h { <i>ConfigName</i> }.c { <i>ConfigName</i> }_user.c	す。PIOR を設定する必要のない端子設定を生成するだけです。 このファイルは、RL78 Software Integration System の 端子設定ソースファイルです。 このファイルは、RL78 Software Integration System の 端子設定ヘッダファイルです。 このフォルダは、プロジェクトに追加される CG ドラ イバ用に生成されます。 このフォルダ内の API 関数には、 <i>ConfigName</i> (設定 名)を含んだ名称がつけられています。 このファイルは、ドライバを初期化する関数 (R_ConfigName_Create)、ドライバに特有な操作、 例えばスタート(R_ConfigName_Start)やストップ (R_ConfigName_Stop)を実行する関数を含みます。 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。
{ConfigName}	r_xxx_pinset.c r_xxx_pinset.h { <i>ConfigName</i> }.c { <i>ConfigName</i> }_user.c	す。PIOR を設定する必要のない端子設定を生成するだけです。 このファイルは、RL78 Software Integration System の 端子設定ソースファイルです。 このファイルは、RL78 Software Integration System の 端子設定ヘッダファイルです。 このフォルダは、プロジェクトに追加される CG ドラ イバ用に生成されます。 このフォルダ内の API 関数には、 <i>ConfigName</i> (設定 名)を含んだ名称がつけられています。 このファイルは、ドライバを初期化する関数 (R_ConfigName_Create)、ドライバに特有な操作、 例えばスタート(R_ConfigName_Start)やストップ (R_ConfigName_Stop)を実行する関数を含みます。 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
{ConfigName}	r_xxx_pinset.c r_xxx_pinset.h { <i>ConfigName</i> }.c { <i>ConfigName</i> }_user.c	す。PIOR を設定する必要のない端子設定を生成するだけです。 このファイルは、RL78 Software Integration System の 端子設定ソースファイルです。 このファイルは、RL78 Software Integration System の 端子設定ヘッダファイルです。 このフォルダは、プロジェクトに追加される CG ドラ イバ用に生成されます。 このフォルダ内の API 関数には、 <i>ConfigName</i> (設定 名)を含んだ名称がつけられています。 このファイルは、ドライバを初期化する関数 (R_ConfigName_Create)、ドライバに特有な操作、 例えばスタート(R_ConfigName_Start)やストップ (R_ConfigName_Stop)を実行する関数を含みます。 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。

L_____ 【注】 *1. xxx は周辺機能名を示します。

*2. LLVM ツールチェーンで生成されるファイルです。

6.4 クロック設定

[クロック] ページで選択したクロック・ソースの設定は、\src\smc_gen\r_config フォルダにある r_bsp_config.h ファイルのマクロに生成されます。main()を実行する前に r_bsp によって、クロック初期化 コードは処理されます。r_bsp_config.h ファイルには、他の MCU 関連の設定(パッケージ、スタックサイ ズなど)も含みます。

クロック設定 ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●	○回 コードの生成 レポートの生成
動作モード: 高速メイン・モード4.0(V)*5.5(V)	*
◎ 高速オンチップ・オシレータ	
MIDER: 32 * (MHz)	fihp
THUCO INITER 2: 18末 (STOPE~ Khoke JU) ~ 2時あたび SNO07Eモードへの称行時に高速オンチップ	32 (MHz)
発展著を組めするための設定があります。)	fMAIN 32 (MHz)
	FCLK
	32000 (kHz)
✓ 中連オンチゥブ・オシレータ	4 (MHz)
周波教: 4 v (MHz)	
· · · · · · · · · · · · · · · · · · ·	
	fMXP (MHz)
mea. 5 (Alto)	(minac)
周波波. ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・	
	fiL
修道オンデップ・オシレータ	32.768 (kHz)
周波教: 32.768 (kHz)	
11.の数件条件はウォッチドッグ・ライマが数件、またはISXPが修道オンチップ・オシ レータを遺訳	fSXP 32.768 (kHz)
	15XR 32.768 (kHz)
E/FC-F. A I 開始	
XT1発展モード: 00:100 00:67	-
世紀モード: STOP HAI TE- kmin mikite 面	
	v
_< 標準 ポード クロック システム コンポーネント 博子 劇り込み	•
HINE TO I CANTA PARTICI LA TRA MARMAN	

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

フォルダ	ファイル	マクロ/関数	説明
r_config	r_bsp_config.h	クロックに関連するマク ロ	これらの設定は、クロック・ソースの [クロッ ク]ページにあるユーザーの選択を基に、ス マート・コンフィグレータによって生成され ます。 <i>main()</i> を実行する前に、r_bsp はクロッ クの初期化を処理します。
		MCU 設定に関連するマ クロ	MCU 関連の設定は、スマート・コンフィグ レータによってマクロが生成されます(例: パッケージタイプ)。マクロの詳細は、r_bsp フォルダのアプリケーションノート (\src\smc_gen\r_bsp\doc)を参照してください。

【注】コードの生成実行前の r_bsp_config.h は trash フォルダにバックアップされます(8 章を参照)。


6.5 端子設定

[端子]ページの設定は、コンポーネントにより下記に示すソースファイルに生成されます。

1) <u>{ConfigName}を使用したドライバの端子初期化</u>

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()を実行する前に処理されます。

🔅 *Smart_Configurator_Example.scfg 🔾	X						- 0
端子設定						じ コードの生成	└ポートの生成
ארבעבעדיזי דעריזיעייי	端子機能					3	1 è 2
フィルタ文字列を入力	フィルタ入力	(* = any string	, ? = any ch	aracter)		すべて	\sim
> 📥 r_bsp	使用する	機能	PIOR	端子割り当て	端子番号	方向	備考
✓ 击 万形波用刀 ✓ Config TAU0 1	\checkmark	TO01		P16/EO16/CCD00/TI01/TO01/INTP5/SI00/RxD0	/ 76	0	
	<						>
端子機能 端子番号							
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 6-8 Config_TAU0_0の端子設定

フォルダ	ファイル	関数	ドライバ	説明
{ConfigName}	{ConfigName}.c	R_ConfigName_ Create	CG	このドライバが使用する端子を API 関数が初期化します。main() 関数を 実行する前に、r_cg_systeminit はこ の関数を呼び出します。

2) <u>RL78 Software Integration System コンポーネントの端子初期化</u>

端子機能は\src\smc_gen\r_pincfg\{*ConfigName*}_pinset.cのR_{*PeripheralName*}_PinSetInitで初期化 されます。このファイルの API 関数は、アプリケーションコードからユーザーによって呼び出されま す。

🔅 *Smart_Configurator_Example.scfg	×							- 0
端子設定						じ コードの生成	 レポートの))生成
יייייייייייייייייייייייייייייייייייייי	端子機能					3	, 24	⊿
フィルタ文字列を入力	フィルタ入力 (* = any string, ? = any character) すべて 🗸 🗸						\sim	
> 💑 r_bsp	使用する	機能	PIOR	端子割り当て	端子番号	方向	備考	^
×.⊕. r ctsu		TS00		P50/TS00/EI50/EO50/CCD03	/ 67	ю		
💓 r_ctsu		TS01		P31/TS01/EI31/TI03/TO03/INTP4/PCLBUZ0	/ 41	IO		~
	<						>	
端子機能 端子番号 概要 ポード クロック システム コンポーネ:	기 사 端子 割り	込み						

図 6-9 r_ctsu の端子設定

フォルダ	ファイル	関数	ドライバ	説明
r_pincfg	{ConfigName} _pinset.c	R_{ <i>PeripheralName</i> } _PinSetInit	RL78 Software Integration System	このドライバが使用する端子を API 関数が初期化します。ユーザーは、 main() 関数でこの関数を呼び出す必 要があります。



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。

割り込み関数は、ファイル\src\smc_gen\{*ConfigName*}\{*ConfigName*}.cのR_ConfigName_Createで初期 化されます。

り込み設	定) コードの	 生成 レポ-	_ −トの生
り込みベクタ									Ĩ
					ベクタ番号				~
ベクタ番号	ベクトルテーブルアドレス	割り込み	割り込み要求元	周辺機能	優先レベル	状態	バンク指定	備考	^
0	00004H	INTWDTI	Watchdog timer interval	WDT	レベル0(高優先順位)		1		
1	00006H	INTLVI	Voltage detection	LVD	レベル3(低優先順位)		なし		
2	00008H	INTPO	Pin input edge detection	INTC	レベル3(低優先順位)		なし		
3	0000AH	INTP1	Pin input edge detection	INTC	レベル3(低優先順位)		なし		~
/									>

図 6-10 割り込み設定

項目	フォルダ	ファイル	ドライバ	説明
Priority	{ConfigName}	{ConfigName}.c	Code Generator	優先レベルは、このファイルの R_ConfigName_Create で初期化されま す。main() 関数を実行する前に、 r_cg_systeminit はこの関数を呼び出し ます。
Bank (CCRL Project)	{ConfigName}	{ConfigName}_user .c	Code Generator	割り込みを次のように宣言します。 #pragma interrupt "Interrupt API Name"(vect="Interrupt Name", bank=RBbankNumber) 図 4-74 CCRL 割り込みバンクコードを 参照してください。
Bank (LLVM Project)	{\projectDIR\src\ smc_gen\general}	r_cg_interrupt_han dler.h		割り込みを次のように宣言します。 Void "Interrupt API Name" (void) attribute ((interrupt(bank=RBbankNumber))); 図 4-75 LLVM 割り込みバンクコードを 参照してください。



7. ユーザープログラムの生成

ここでは、スマート・コンフィグレータが出力したソースファイルへ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 <u>xxx</u>. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */

[コード生成]の場合、指定した周辺機能ごとに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 ADC)の場合、xxx は "ADC"と名付けられます。)カスタムコード を追加するためのコメントは、「*.c」ファイルの先頭と最後および「*.h」ファイルの最後に設けられる 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 ADC の例 (Config_ADC_user.c)を示します。

Includes #include "r cg macrodriver.h" #include "r cg userdefine.h" #include "Config_ADC.h" /* Start user code for include. Do not edit comment generated here */ /* End user code. Do not edit comment generated here */ Pragma directive #pragma interrupt r Config ADC interrupt(vect=INTAD) /* Start user code for pragma. Do not edit comment generated here */ /* End user code. Do not edit comment generated here */ Global variables and functions /* Start user code for global. Do not edit comment generated here */ /* End user code. Do not edit comment generated here */



```
* Function Name: R Config ADC Create UserInit
* Description : This function adds user code after initializing the AD converter.
* Arguments : None
* Return Value : None
       *****
void R_Config_ADC_Create_UserInit(void)
{
  /* Start user code for user init. Do not edit comment generated here */
  /* End user code. Do not edit comment generated here */
}
* Function Name: r_Config_ADC_interrupt
* Description : This function is INTAD interrupt service routine.
* Arguments : None
* Return Value : None
static void __near r Config ADC interrupt(void)
{
  /* Start user code for r_Config_ADC_interrupt. Do not edit comment generated here */
  /* End user code. Do not edit comment generated here */
}
/* Start user code for adding. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
```



7.2 ユーザーアプリケーションコードの使用方法

RL78 Software Integration System Modules およびコード生成のコードを使用するには、以下の手順で行います。

(1) {*Project name*}.c ファイルを開き、使用するモジュールのヘッダファイルをインクルードコード に追加します。

RL78 Software Integration System Modules の場合は、r_xxx.h です。 コード生成の場合は、自動的に r_smc_entry.h に追加されます。



図 7-1 ヘッダファイルの追加

(2) main() 関数で生成された関数を呼び出し、アプリケーションコードを追加します。

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_systeminit.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) と停止 (R_ConfigName_Stop) です。



図 7-2 コード生成関数コール

RL78 Software Integration System Modules の場合は、対応するアプリケーションノートの 「API 機能」の章に記載されている例を参照してください。

詳細については、「13. 参考ドキュメント」のスマート・コンフィグレータのアプリケーション例を参照 してください。



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、以下の場所にソースコードをバックアップする機能があります。

<ProjectDir>\trash\<Date-and-Time>

[コード生成] - 「 ボタンをクリックしてコード生成を行うと、スマート・コンフィグレータはコード 生成前のソースのバックアップを作成します。<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。



9. レポートの生成

スマート・コンフィグレータは、ユーザー設定のレポートを提供します。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(PDF または txt 形式)

スマート・コンフィグレータビューで[レポートの生成] _{レポーや生成} ボタンをクリックし、レポートを出力 します。

Smart_Cont	figurator_Example.scfg $ imes$					
概説		⁽⁾ コードの生成 レポートの生成				
▼ 機能概要		0				
	概要 盛要をクリックすると、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。					
	動画 <u>スマート・コンフィグレータの紹介</u> 関連動画	Application Code				
	最新情報 <u>最新情報</u> <u>気新情報</u> を ができます。 すべての <u>リリースノート</u> を表示 現在のバージョン: V1.13.0	RTOS Device Drivers				
	製品ドキュメントとよくあるご質問(FAQ) ユ <u>ーザーガイド</u> <u>APIマニュアル</u> アブリケーションノート ツールニュース	MCU Hardware				
	FAQ : Smart Configurator					
▼ 現在の設定	2状態					
注:コード生成の動作は、 <u>設定</u> ページでカスタマイズできます。 ① 使用しているボード/デバイス: R7F100GSLxFB (ROM size: 512KB, RAM size: 48KB, Pin count: 128)						
生成先ロケー	生成先ロケーション (PROJECT_LOC¥): New_Folder 編集					
概要 ボード ク	ロック システム コンボーネント 端子 割り込み					

図 9-1 全設定内容レポート出力(PDF または txt 形式)

レポート生成		Х
設定内容のレポートを出力します		
- オブション		
◉ 全ての項目を出力する		
○選択した項目を出力する		
→ □ 200 A > □ 200 A		
> ■ 端子		
□割り込み		
✓ PDFで出力	フォント	<u>選択</u>
」 テキストで出力		
C:¥e2_studio¥workspace¥Smart_Configurator_Example¥output	参照	R
ОК	キャンセ	μ

図 9-2 レポート出力ダイアログ



9.2 端子機能リスト、端子番号リストの設定内容

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、表示中の端子リスト(端子機能リストまたは端子番号リスト)の設定内容を出力します。

🕸 Smart_Configurator_Example.scfg 🗙								- 0
端子設定						じ コードの生き	ば レポート	<u>)</u> の生成
א-גאזעגעאזערע װעראַדערעערערערערערערערערערערערערערערערערערע	端子機能					- Q 🔳	I] 2	14
フィルタ文字列を入力	フィルタ入力] (* = any s	tring, ? = any	character)		すべて		\sim
👗 すべて 🔨	使用する	機能	PIOR	端子割り当て	端子番号	方向	備考	^
\$₩ 1/0ボート		ANIO		P20/ANI0/AVREFP/EI20	/ 117	1		
		ANI1		✓ 設定されていません	∥ 設定されていません	なし		_
> (2. 917·7/1·1=>>		ANI2		✓ 設定されていません	∥ 設定されていません	なし		_
		ANI3		✓ 設定されていません	/ 設定されていません	なし		
> 30) クロック田川/ ノリー田川制御回路		ANI4		🥒 設定されていません	🥖 設定されていません	なし		
		ANI5		🥒 設定されていません	🥒 設定されていません	なし		
		ANI6		🥒 設定されていません	🥒 設定されていません	なし		
		ANI7		🥒 設定されていません	🥒 設定されていません	なし		
		ANI8		🥒 設定されていません	🥒 設定されていません	なし		
		ANI9		🥒 設定されていません	🥒 設定されていません	なし		\sim
< >	<						2	*
····································								
概要 ホート クロック システム コンホーネント [端子	割り込み							

図 9-3 端子機能リスト、端子番号リスト出力(csv 形式)

9.3 MCU/MPU パッケージ図

[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. Developer Assistance

Developer Assistance 機能は、プロジェクト作成時にスマート・コンフィグレータを選択すると、プロ ジェクト・ツリーに [Developer Assistance] という仮想ルート・ノードを提供します。

コード生成された API は [Developer Assistance] ツリーに表示され、API を選択すると [Developer Assist Browser] ビューに API 情報をナビゲートします。コーディング中に API テンプレート・ノードを C/C++エ ディタにドラッグ&ドロップすることもできます。



図 10-1 API 情報のナビゲートと API のドラッグ&ドロップ

[Developer Assist Browser] ビューでテキストを選択し、[Copy] コンテキスト・メニューを使用して、 コード生成コンポーネント API 使用例のコードスニペットを C/C++エディタに貼り付けることができま す。



図 10-2 Usage example のコピー&ペースト



11. ユーザーコード保護機能

e² studio 2023-01 Smart Configurator for RL78 プラグイン以降のバージョンより、新たなユーザーコード 保護機能をサポートしました。図 11-1 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 できるようになりました。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のみサポートします。

11.1 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 11-1 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。



11.2 ユーザーコード保護機能の使用例

図 11-2 に示すように、図 11-1 の指定タグを使用し、A/D コンバータモジュールの 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、A/D コンバータの GUI 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。



図 11-2 ユーザーコードの保護機能



11.3 競合発生時の対応方法

11.3.1 競合の発生条件

GUIの設定変更やスマート・コンフィグレータのバージョンアップにより、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、生成コードに競合が発生します。

競合が発生した場合、図 11-3 生成コードのメッセージのようにコンソールに競合メッセージが表示されます。

א א-עיב 🖳 🖬 🛃 🚽 🖻 🗸 😭	
スマート・コンフィグレータ出力	
M0000001: コード生成を開始	~
M04000001: ファイルを生成: <u>src\smc_gen\Config_RTC\Config_RTC.h</u>	
M04000001: ファイルを生成: <u>src\smc_gen\Config_RTC\Config_RTC.c</u>	
M0000005: 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	
M0000002: コード生成の終了: <u>C:\Users\\\e2_studio\workspace\Smart_Configurator_Example\src\smc_gen</u>	
	\sim
<	>

図 11-3 生成コードのメッセージ

ユーザーは、コンソール メッセージに競合ファイルをクリックし、[ファイル比較]ビューを開き、次の章 11.3.2 競合の解決方法ように競合を解決できます。



11.3.2 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) コンソールメッセージの競合ファイルをクリックし、[File Compare] ビューを開きます。(図 11-4)
- (2) 図 11-4 のように、矢印 🕟 をクリックし左側パネルのコードを右側パネルにコピーします。



図 11-4 生成コードの競合解決前

(3) 図 11-56 のように、適切な位置にコードを追加し、不要なコードを削除します。



図 11-56 生成コードの競合解決後



(4) コードのマージ後、保存 🔚 アイコンをクリックし保存します。

B • % • \$ • 9 • B # B & 3	Q 18 QC++ 27-1-377
Smart_Configurator_Example.scfg	8
<mark>₿</mark> C比較	
● 肥厚单位 ●● R_Config_RTC_Create	
6 C比較ビューア− 8	M 📰 😤 👿 🔗 📣 强 43
isting code	New code
<pre>62 RTCIF = 00; /* Set FATCK = / 00; /* Set L2/24-hour system and period of Constant-period interrupt (INTATC) */ /* Set L2/24-hour system and period of Constant-period interrupt (INTATC) */ /* Start User code */ POUS &-007CU; /* Start User code */ 70 R_Config_RTC_Create_UserInit(); 72 74 75 * Function Name: #_Config_RTC_Start 76 * Return Value : None 78 * Return Value : None 79 78 * Return Value : None 79 79 70 70 * Return Value : None 70 70 * Return Value : None 70 70 * Return Value : None 70 70 * Return Value : None 71 70 * Return Value : None 71 71 72 * Seturn Value : None 72 75 * Return Value : None 75 75 * Return Value : None 75 75 75 75 75 75 75 75 75 75</pre>	<pre>66 RTCC0 C 400 RTC RTCHC DTABLE 40 RTC_2440UR_MODE 40 RTC_INTRTC_NOT_GENERA 67 /* Start user code */ 68 RTC_COUNTER CTCLTD_CTABLE 40 RTC_2440UR_MODE 40 RTC_INTRTC_NOT_GENERA 67 /* Start user code */ 68 RTC_COUNTER SEC; 79 /* End user code */ 71 SEC = 00 RTC_COUNTER SEC; 72 RTCC 00 RTER MOUR; 73 MOUR = 12 RTC_COUNTER MOUR; 74 MUE = 00 RTC_COUNTER MOUR; 75 DATA = 00 RTC_COUNTER MOUR; 76 NOTH = 10 RTC_COUNTER NOUR; 77 YVAR = 00 RTC_COUNTER NOUR; 78 RTCC0 = (-90 RTC_RTCHT2; DISABLE -90 RTC_12HOUR_MODE _00 RTC_INTRTC_NOT_GENERA 78 RTCC0 = (-90 RTC_RTCHT2; DISABLE _00 RTC_12HOUR_MODE _00 RTC_INTRTC_NOT_GENERA 74 RTCC0 = (-90 RTC_RTCHT2; DISABLE _00 RTC_12HOUR_MODE _00 RTC_INTRTC_NOT_GENERA 75 RTCC0 = (-90 RTC_COUNTER VIE); 76 RTCC0 = (-90 RTC_RTCHT2; DISABLE _00 RTC_12HOUR_MODE _00 RTC_INTRTC_NOT_GENERA 74 RTCC0 = (-90 RTC_RTCHT2; DISABLE _00 RTC_12HOUR_MODE _00 RTC_INTRTC_NOT_GENERA 75 RTCC0 = (-90 RTC_RTCHT2; DISABLE _00 RTC_12HOUR_MODE _00 RTC_INTRTC_NOT_GENERA 75 RTCC0 = (-90 RTC_RTCHT2; DISABLE _00 RTC_12HOUR_MODE _00 RTC_INTRTC_NOT_GENERA 75 RTCC0 = (-90 RTC_RTCHT2; DISABLE _00 RTC_12HOUR_MODE _00 RTC_INTRTC_NOT_GENERA 75 RTCC0 = (-90 RTC_RTCHT2; DISABLE _00 RTC_12HOUR_MODE _00 RTC_INTRTC_NOT_GENERA 75 RTCC0 = (-90 RTC_RTCHT2; DISABLE _00 RTC_12HOUR_MODE _00 RTC_INTRTC_NOT_GENERA 75 RTCC0 = (-90 RTC_RTCHT2; DISABLE _00 RTC_12HOUR_MODE _00 RTC_INTRTC_NOT_GENERA 75 RTCC0 = (-90 RTC_RTCHT2; DISABLE _00 RTC_12HOUR_MODE _00 RTC_INTRTC_NOT_GENERA 75 RTCC0 = (-90 RTC_RTCHT2; DISABLE _00 RTC_12HOUR_MODE _00 RTC_INTRTC_NOT_GENERA 75 RTCC0 = (-90 RTC_RTCHT2; DISABLE _00 RTC_12HOUR_MODE _00 RTC_INTRTC_NOT_GENERA 75 RTCC0 = (-90 RTC_RTCHT2; RTCHT2; RTCR_RTC_RTC_RTC_RTC_RTC_RTCHT2; RTCR_RTC_RTC_RTCHT2; RTCR_RTC_RTCHT2; RTCR_RTC_RTC_RTC_RTCHT2; RTCR_RTC_RTC_RTC_RTC_RTC_RTC_RTC_RTC_RTC</pre>

図 11-7 コードの削除と保存

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。

【注】 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、[ファイル比較]ビューを開くことができます。



12. ヘルプ

スマート・コンフィグレータの詳細については、e² studio メニューのヘルプを参照してください。 ヘル プをメニューから選択すると、ヘルプの目次が表示されます。



図 12-1 ヘルプ表示

ヘルプは、[概要] ページの 🕐 アイコンからも参照できます。



図 12-2 クイックスタート

どちらの方法でも同じヘルプを参照できます。



13. 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 (R20UT4374)

CC-RL コンパイラ ユーザーズマニュアル (R20UT3123)

スマート・コンフィグレータ ユーザーズマニュアル RL78 API リファレンス (R20UT4852)

(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。)

【アプリケーションノート:SMS & ELCL】

(最新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。



改訂記録

Rev.	セクション	改訂内容
1.00	-	新規作成
1.01	第2章 プロジェクの作成	2.1 プロジェクの作成:図 2-5,2-7 の変更
	第3章 スマート・コンフィ	3.4 ウィンドウ:図 3-3,3-4,3-5 の変更
	グレータレータの操作方法	3.4.6 Developer Assist Browser:3.4.6 章の追加
	第 10 章 Developer	10.Developer Assistance: 10章の追加
1.02	Assistance 要旨	
1.02	<u>スロ</u> 筆 4 音 周辺機能の設定	411 デバイス選択・図 4-2 表 4-1 及び表 4-1 の注を削除
		4.1.2 ボード選択:図 4-4.表 4-2 及び表 4-2 の注を削除
		4.4.12 BSP コンフィグレーションのバージョン変更の注を削除
		4.4.13 コンポーネントの基本設定:図 13-1 コンポーネントの基 本設定の変更
		4413 コンポーネントの基本設定・注12の変更
		4.4.13 コンポーネントの基本設定:注3の追加
		4.6.2 割り込みバンクの設定:手順(3)の変更、図 4-57 割り込
		みバンク設定の変更
	第 12 章 参考ドキュメント	SMS & ELCL アプリケーションノート:参考先を削除
1.03	第2章 プロジェクの作成	e² studio プロジェクト作成ウィザード 更新
	第3章 スマート・コンフィ	3.4.3 MCU/MPC パッケージビュー 更新
	グレータの操作	
	第4章 周辺機能の設定	4.1.2 ボード選択 更新
		4.4.3 ソフトウェア・コンポーネントの削除 複数コンポーネント の指定方法を追加
		4.4.10 RL78 Software Integration System モジュールのダウン
		ロート 史利 4 4 11 PL 78 Software Integration System エジュールの追加 追加
		4.4.12 RI 78 Software Integration System モジュールの設定 再新
		4.5 端子設定 図 4-50 図 4-51 更新
		453 MCU/MPU パッケージビューを使用した端子の設定 更新
		4.5.4 端子機能から端子番号の表示 追加
		4.5.9 端子エラー/端子警告の設定 追加
		4.7 MCU マイグレーション機能 更新
	第6章 ソースの生成	6.2 コード生成場所の変更 追加
		6.3 生成ファイルの構成とファイル名 更新
	第 7 章 ユーザープログラム の作成	7.2 ユーザーアプリケーションコードの使用方法 追加
	第 11 章 ユーザーコード保護 機能	11. ユーザーコード保護機能 追加
1.04	第4章 周辺機能の設定	4.4.8 ELCL 固定機能モジュールのダウンロード 更新
		4.4.9 固定機能 ELCL コンポーネントの設定 更新
		4.4.10 ELCL Flexible Circuit の作成と編集 追加
	第6章 ソースの生成	6.3 生成ファイルの構成とファイル名

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_{IL}(Max.)か ら V_{IH}(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_{IL}(Max.)から V_{IH} (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。

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。
- 5. 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

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。

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/