# e<sup>2</sup> studio

統合開発環境

ユーザーズマニュアル 入門ガイド

対象デバイス RX, RL78, RH850, RZ ファミリ



Rev.4.00 2016.5

#### 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。お客様の機器・システムの設計において、回路、ソフトウェアおよ びこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの 使用に起因して、お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
- 2. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを 保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客 様に生じた場合においても、当社は、一切その責任を負いません。
- 3. 本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害に関し、当社は、何らの責任を負うものではありません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
- 4. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製等により生じた損害 に関し、当社は、一切その責任を負いません。
- 5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、
  - 各品質水準は、以下に示す用途に製品が使用されることを意図しております。
    - 標準水準: コンピュータ、OA 機器、通信機器、計測機器、AV 機器、
      - 家電、工作機械、パーソナル機器、産業用ロボット等

高品質水準: 輸送機器(自動車、電車、船舶等)、交通用信号機器、

防災・防犯装置、各種安全装置等

当社製品は、直接生命・身体に危害を及ぼす可能性のある機器・システム(生命維持装置、人体に埋め込み使用するもの等)、もしくは多大な物的損害を発生させるおそれのある機器・システム(原子力制御システム、軍事機器等)に使用されることを意図しておらず、使用することはできません。たとえ、意図しない用途に当社製品を使用したことによりお客様または第三者に損害が生じても、当社は一切その責任を負いません。なお、ご不明点がある場合は、当社営業にお問い合わせください。

- 6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
- 7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
- 8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せく ださい。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境 関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令 を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
- 9. 本資料に記載されている当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。また、当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途に使用しないでください。当社製品または技術を輸出する場合は、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。
- 10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
- 11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
- 注1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネ サス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する 会社をいいます。
- 注 2. 本資料において使用されている「当社製品」とは、注1において定義された当社の開発、製造 製品をいいます。

# このマニュアルの使い方

このマニュアルは、アプリケーション・システムを開発する際の統合開発環境である e<sup>2</sup> studio について説明します。

e<sup>2</sup> studio は RX ファミリ、RL78 ファミリ、RZ ファミリ、および Renesas Synergy 用の統合開発環境で、ソフトウェア開発における、設計、実装、デバッグなどの各開発フェーズに必要なツールを単一のプラットフォームに統合しています。 Synergy 開発プラットフォームについては、以下の Synergy Gallery のサイトにあるドキュメントを参照してください。 https://synergygallery.renesas.com/

統合することで、さまざまなツールを使い分けずにすべての開発を本製品のみで行うことができます。

対象者	このマニュアルは、e <sup>2</sup> studio を使用してソフトウェアやハードウェアのアプリケーション・システムを開発するユーザを対象としています。		
目的	このマニュアルは、ユーザカ ェアのシステム開発を始める	「ターゲットデバイスを使用してハードウェアやソフトウ る際の e <sup>2</sup> studio の機能を説明します。	
構成	このマニュアルは、大きく分	けて次の内容で構成しています。	
	第1章 概説		
	第2章 インストール		
	第3章 プロジェクトの生成	:	
	第4章 ビルド		
	第5章 デバッグ		
	第6章 ヘルプ		
読み方	このマニュアルを読むにあれる一般知識が必要となります	こっては、電気、論理回路、マイクロコンピュータに関す す。	
凡例	データ表記の重み	: 左が上位桁、右が下位桁	
	アクティブ・ロウの表記	:XXX(端子、信号名称に上線)	
	注	:本文中につけた注の説明	
	注意	:気をつけて読んでいただきたい内容	
	備考	:本文中の補足説明	
	数の表記	:10 進数 XXXX 16 進数 0xXXXX	

第1章	둘 概説	1
1.1.	システム構成	1
1.2.	動作環境	1
1.2.1.	システム要件	1
1.2.2.	サポートするツールチェーン	2
第2章	፪ インストール	3
2.1.	e <sup>2</sup> studioのインストール	3
2.2.	e <sup>2</sup> studioのアンインストール	4
2.3.	e <sup>2</sup> studioのバージョンアップ	4
2.4.	コンパイラパッケージのインストール	6
笙3音	ま プロジェクトの作成	7
31	- ッロッエットの17/30	7
3.2	「「「「」」」)」)」)」「「「「」」)」)」)」)」)」)」)」)」)」)」	11
3.2. 3.3	ノ ノ 、 、 、 、 、 、 、 、 、 、 、 、 、 、 、 、 、 、	
3.4	「12Wクロクエクトのインボート	17
0.4.		15
第4章	☞ ビルド	21
4.1.	ビルドオプションの設定	21
4.2.	サンプルプロジェクトのビルド	23
4.3.	ビルド構成のレポート	24
第5章	を デバッグ	25
5.1.	ー ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・	
5.2.	新規デバッグ構成の作成	30
5.3.	基本的なデバッグ機能	32
5.3.1.	ブレークポイントの使い方	33
5.3.2.	式ビュー	35
5.3.3.	レジスタービュー	36
5.3.4.	メモリービュー	37
5.3.5.	· 一 逆アセンブルビュー	39
5.3.6.	変数ビュー	41
5.3.7.	イベントポイントビュー	42
5.3.8.	IOレジスタビュー	45
5.3.9.	トレースビュー	46
たっぷ		40
- 第6章	ューヘルノ	49

# 目次

# 第1章 概説

ルネサス製 Eclipse ベースの組み込みスタジオ(通称 e<sup>2</sup> studio)は、ルネサス製マイクロコントローラをサポ ートする開発環境です。これは、ビルドフェーズ(エディタ、コンパイラ、リンカコントロールなど)及び拡張 GDB インタフェースをサポートするデバッグフェーズをサポートしたオープンソース Eclipse CDT (C/C++ 開発ツ ール)プロジェクトに基づき開発されています。

この章では、RX ファミリシリーズマイクロコントローラ用のアプリケーション開発を例として、e<sup>2</sup> studio のシス テム構成と動作環境を説明します。

本ドキュメントの e<sup>2</sup> studio に関する説明は、特に注意書きのない限り e<sup>2</sup> studio V5.0 に基づくものです。

#### 1.1. システム構成

一般的なシステム構成の例を以下に示します。



図 1-1 システム構成

#### 1.2. 動作環境

本製品に対するシステムの必要条件を以下に示します。

#### 1.2.1. システム要件

PC ハードウェア環境:

プロセッサ: 1GHz 以上(ハイパースレッディング及びマルチコア CPU をサポートする)

メインメモリ: 1GB 以上(特に Windows 64 ビット OS に関しては 2GB 以上が望ましい)

ディスプレイ: 解像度 1,024 x 768 ピクセル以上; 65,536 色以上

インタフェース: USB 2.0 (ハイスピードまたはフルスピード) ハイスピードが望ましい

PC ソフトウェア環境:

Windows 7 (32/64 ビット OS), Windows 8.1 (32/64 ビット OS), および Windows 10 (32/64 ビット OS)

## 1.2.2. サポートするツールチェーン

#### 1.2.2.1. サポートするコンパイラ

RX ファミリ用ルネサス C/C++コンパイラパッケージ RL78 ファミリ用ルネサス C/C++コンパイラパッケージ GCC for Renesas GNURX Windows ツールチェーン GCC for Renesas GNURL78 Windows ツールチェーン Renesas GNUARM-NONE Windows ツールチェーン

注: Renesas 製コンパイラパッケージには「統合開発環境付き」、「統合開発環境なし」がありますが、 e<sup>2</sup> studio にはいずれも適用可能です。

#### 1.2.2.2. サポートするエミュレータ

E2 エミュレータ Lite (RX, RL78), E1 (RX, RL78, RH850), E20 (RX), IECUBE (RL78), Segger J-Link (RX, RZ)

#### 1.2.2.3. サポートするシミュレータ

Renesas シミュレータ (RX, RL78)



# 第2章 インストール

この章では、e<sup>2</sup> studio のインストールおよびアンインストールについて説明します。

最新の e<sup>2</sup> studio 統合開発環境インストーラパッケージはルネサスウェブサイト (<u>http://www.renesas.com/e2studio\_download</u>)から無償でダウンロードできます。ダウンロードするに は無料で登録できる MyRenesas アカウントが必要です。

- 注意: 1. インストーラは二種類ありますが、Web インストーラ(サイズの小さな方)を使用するにはインター ネット接続が必要です。
  - e<sup>2</sup> studio ではメジャーバージョン間のアップデートをサポートしていません(例: V4.x(V4.0.0 から V4.3.1 のいずれか)から V5.x(V5.0.0 以降)へのアップデート)。古いバージョンをアンインストー ルしてから新しいバージョンをインストールしてください。あるいは、新しいバージョンの e<sup>2</sup> studio を新しいフォルダにインストールしてください。
  - 3. e<sup>2</sup> studio インストーラ(V4 以降)には、インストール済みの e<sup>2</sup> studio の機能を削除または追加 する [修正] 機能があります。
  - 4. [ヘルプ] メニューの [更新の検査] による更新機能はプラグインのアップデートを行うための機能です。e<sup>2</sup> studio 全体を更新するにはインストーラを御使用ください。

#### 2.1. e<sup>2</sup> studio のインストール

- (1) e<sup>2</sup> studio インストーラをダブルクリックして e<sup>2</sup> studio インストールウィザードページを開いてください。
   [Next] をクリックします。
- (2) Install Folder

デフォルトのインストール先は "C:\Renesas\e2\_studio" に設定されています。変更する場合は、イ ンストールするフォルダをテキストボックスに直接入力するか [Browse...] ボタンをクリックして指定 してください。 また e<sup>2</sup> studio をインストールする Windows ユーザを選択してください。 [Next] をクリックします。

- (3) **Device Families** インストールするデバイスファミリを選択してください。[Next] をクリックします。
- (4) Extra Components

インストールする追加コンポーネント(言語パック、SVN および Git サポート、Micrium、RTOS サポート)を選択してください。[Next] をクリックします。

- (5) Components必要なコンポーネントを選択し、[Next] をクリックします。
- (6) Additional Software
   追加ソフトウェア(コンパイラ、ユーティリティ)を選択し、[Next] をクリックします。
- (7) License Agreement
   ライセンス契約を読んで同意した後、[Next] をクリックします。
   ライセンス契約に同意しない場合、インストールは続行できません。

#### (8) Shortcuts

スタートメニューに表示するショートカット名を選択し、[Next] をクリックします。

(9) Summary

[Install] をクリックし、ルネサス e<sup>2</sup> studio をインストールしてください。

(10) Installing...

インストールが実行されます。Additional Software で選択した項目に応じて、ソフトウェアをインスト ールするためのダイアログボックスが開きます。

#### (11) Results

[Finish] ボタンをクリックしてインストールを完了してください。

#### 2.2. e<sup>2</sup> studio のアンインストール

e<sup>2</sup> studio のアンインストールは、Windows OS での通常のプログラムアンインストール手順で行えます。

- (1) [スタート] → [コントロールパネル] → [プログラムと機能]を選択します。
- (2) インストール済みプログラムのリストから、"e<sup>2</sup> studio" を選択し、[アンインストール(U)] ボタンをクリ ックします。
- (3) [アンインストール] ダイアログボックスの [アンインストール] ボタンをクリックして削除を確認してく ださい。

アンインストールの最後に、e<sup>2</sup> studio はインストール先から削除され、ショートカットメニューも削除されます。

#### 2.3. e<sup>2</sup> studio のバージョンアップ

本節では、インストーラのバージョンアップ機能を用いてバージョンアップする手順を説明します。

- 下記のルネサスページからインストールしたいバージョンの e<sup>2</sup> studio インストーラをダウンロードしてください。<u>http://www.renesas.com/e2studio\_download</u>
- 注: 差分アップデートプログラムを使用したバージョンアップは、バージョン 3.X 以下の e<sup>2</sup> studio での み可能です。
- (2) (1)でダウンロードしたインストーラファイルをダブルクリックして実行してください。インストーラがイン ストール済みのバージョンを検出します。インストール済みの e<sup>2</sup> studio を更新するか、あるいは別 のフォルダに新しいバージョンの e<sup>2</sup> studio をインストールするかを選択できます。
   更新する場合は[アップグレード] を選択し、[次へ(N)>] をクリックしてバージョンアップを開始してく ださい。表示されたフォルダに上書きしない場合は[インストール]を選択して別フォルダを指定してく ださい。

e <sup>²</sup> studio 5.1.0.022 セットアップ		RE	ΝΕΣΛΣ
e2 studio バージョン 5.0.0.043 は既に	インストールされています。		
	何を実施しますか?		
	アップグレード パージョン 5.1.0.022 へのア	ップグレード	
	インストール 異なる場所へのインストール		

図 2-1 インストーラを使用した e<sup>2</sup> studio のパージョンアップ

(3)「2.1 章 e<sup>2</sup> studio のインストール」の手順を実行してください。[アップグレード] を選択した場合は既存の e<sup>2</sup> studio と同じフォルダにインストールしますので、「2.1 章の(2) Install Folder」の手順は不要になります。

e2 studio について	
e2 studio Version: 5.1.0.022 Parts Copyright (C) 2010-2016 Renesas Electronics Corp. All rights reserved.	
e2 studio は eclipse.org. に対して開発された拡張されたソフトウェアです。	
? インストール詳細(i) OK	

図 2-2 e<sup>2</sup> studio – [e2 studio について] パネル

(4) [ヘルプ] メニューの [e2 studio について] で e<sup>2</sup> studio のバージョンが確認できます。

#### 2.4. コンパイラパッケージのインストール

V4.0 以降の e<sup>2</sup> studio インストーラでは、インターネット接続環境ではインストール時に自動的にコンパイラ パッケージをインストールできますが、インターネット接続のない環境、または V3.1 以前の e<sup>2</sup> studio インスト ーラを使用した場合にはコンパイラパッケージがインストールされません。

コンパイラパッケージを別途インストールする必要がある場合には、それぞれ下記のサイトからダウンロード してください。コンパイラパッケージのインストール方法については各パッケージのダウンロードページをご覧 ください。

Renesas 製コンパイラパッケージのダウンロードサイト:

RX ファミリ用 <u>http://www.renesas.com/rx\_c</u> RL78 ファミリ用 <u>http://www.renesas.com/rl78 c</u>

GCC for Renesas ツールチェーンのダウンロードサイト:

https://gcc-renesas.com/ja/

コンパイラがインストール済みであるかを確認するには、ツールバーの <sup>ど</sup> をクリックするか、[ヘルプ] メニ ューから[Renesas ツールチェーンの追加] をクリックして下図に示す [Renesas ツールチェーン管理] を開 きます。e<sup>2</sup> studio に統合したいツールチェーンを確認してください。

使用したいコンパイラがリストにない場合は、[追加] ボタンをクリックしてインストールされている場所を指定 してください。



図 2-3 ツールチェーン管理

# 第3章 プロジェクトの作成

この章では、新規プロジェクトの作成、および既存の e<sup>2</sup> studio プロジェクト、HEW 統合開発環境プロジェクト、CS+プロジェクトの e<sup>2</sup> studio へのインポートについて説明します。

- 注意: 1. ご利用の PC に e<sup>2</sup> studio をインストールして使用する場合は、別途提供されるコンパイラパッケー ジをインストールする必要があります。
  - 2. e<sup>2</sup> studio インストールフォルダ名、ワークスペースフォルダ名、プロジェクト名、ビルド構成名、ソースファイル名に多バイト文字は使用できません。ソースファイルを配置するファイルパスにも多バイト文字が含まれないように留意してください。

#### 3.1. 新規プロジェクトの作成

ルネサス RXC ツールチェーンで新規プロジェクトを作成するには、Windows の [スタート] メニューから e<sup>2</sup> studio を起動し、ワークスペースディレクトリを指定します。

(1) [ファイル] → [新規] → [C Project] の順にクリックして新しい C プロジェクトを作成します。下図に示 す新規プロジェクト作成ウィザードが起動します。

選択したタイプの C プロジェクトを作成します          プロジェクト名(P): Tutorial         グ デフォルト・ロケーションの使用(P)         ロケーション(L): D:¥Workspace¥Tutorial         ⑦ Create Directory for Project         グロジェクトの確認:         グロジェクトの確認:         ※ Executable (Renesas)         ● Sample Project         ※ Static Library (Renesas)         ● Sample Project         ▶ Debug-Only Project         ▶ Executable (IAR)         ▶ Executable (IAR)         ▶ E. Library (IAR)         ▶ E. Makefile プロジェクト	¢プロジェクト	
プロジェクト名(P): Tutoria ⑦ デフオルト・ロケーションの使用(D) □ケーション(L): D:¥Workspace¥Tutorial 参照(R) ⑦ Create Directory for Project プロジェクトの確認: ツールチェイン: ⑦ Executable (Renesas) ⑧ Sample Project ⑧ Sample Project ◎ Executable (IAR) ◎ Executable (IAR) ◎ Executable (IAR) ◎ Executable (IAR) ◎ Executable (IAR) ◎ Executable (IAR) ◎ Makefile プロジェクト	- ダー・ - ダー・ · · 選択したタイプの C プロジェクトを作成します	
<ul> <li>▼フォルト・ロケーションの使用(2)</li> <li>□ケーション(1): D:¥Workspace¥Tutorial 参照(2)</li> <li>② Create Directory for Project</li> <li>プロジェクトの種類: ソールチェイン:</li> <li>▲ Executable (Renesas)</li> <li>▲ Sample Project</li> <li>※ Static Library (Renesas)</li> <li>▲ Sample Project</li> <li>※ Executable (IAR)</li> <li>▲ Library (IAR)</li> <li>▲ Library (IAR)</li> <li>▲ Makefile プロジェクト</li> </ul>	プロジェクト名(P): Tutoria	
ロケーション(L): D:¥Workspace¥Tutorial 参照(R)  ⑦ Create Directory for Project  ⑦ Create Directory for Project  ③ Sample Project  ③ Sample Project  ⑤ Debug-Only Project  ⑤ Executable (IAR)  ⑤ ELibrary (IAR)  ⑥ E Library (IAR)  ⑥ Makefile プロジェクト  ⑦ Makefile プロジェクト	デフォルト・ロケーションの使用(D)	
Create Directory for Project       ツールチェイン:         グロジェクトの種類:       ツールチェイン:         ・ ⑤ Executable (Renesas)       ● Sample Project         ・ ⑤ Static Library (Renesas)       ● Sample Project         ● Debug-Only Project       KPIT GNURX-ELF Toolchain         ▶ ⑥ Debug-Only Project       Renesas CCRL Toolchain         ▶ ⑥ Library (IAR)       Renesas SHC Toolchain         ▶ @ Aukefile プロジェクト       Renesas SHC Toolchain	ロケーション(L): D:¥Workspace¥Tutorial	参照(民)
プロジェクトの種類: ツールチェイン: ・ Executable (Renesas) ・ Sample Project ・ Static Library (Renesas) ・ Sample Project ト Debug-Only Project Debug-Only Pr	Create Directory for Project	
<ul> <li>Executable (Renesas)</li> <li>Sample Project</li> <li>Static Library (Renesas)</li> <li>Sample Project</li> <li>Debug-Only Project</li> <li>Executable (IAR)</li> <li>Library (IAR)</li> <li>Library (IAR)</li> <li>Makefile プロジェクト</li> </ul>	プロジェクトの種類:	ツールチェイン:
	<ul> <li>Executable (Renesas)</li> <li>Sample Project</li> <li>Static Library (Renesas)</li> <li>Sample Project</li> <li>Debug-Only Project</li> <li>Executable (IAR)</li> <li>Excutable (IAR)</li> <li>Elibrary (IAR)</li> <li>Amkefile プロジェクト</li> </ul>	KPIT GNUARM-NONE-EABI Toolchain KPIT GNURL78-ELF Toolchain KPIT GNURX-ELF Toolchain KPIT GNUSH-ELF Toolchain Renesas CCRL Toolchain Renesas RXC Toolchain Renesas SHC Toolchain
	(?) < 戻る(目)	次へ(N) > 終了(E) キャンセル

図 3-1 新規プロジェクト作成ウィザード (1/4)

(2) プロジェクト名を入力し、"Renesas RXC Toolchain"を選択します。[次へ(N)>] をクリックしてください。 "Renesas RXC Toolchain"が表示されない場合は「2.5 コンパイラパッケージのインストール」 の手順にしたがってください。

e2 studio - ブ	ロジェクト生	成				
ターゲット固て	有の設定の選択					
ツールチュ	:ーン・バージョン	: v2.03.00	•			
デバッグ・	ハードウェア:	E1 (RX)	•			· VERE
データ・コ	ンディアン:	Little-endian data	•			250
ターゲット	∽の選択:	R5F564MLCxFC				
Renesas I	RTOS サポート:	None	•		31	
構成の選	R:					
<b>▼</b> /(-h	<sup>、</sup> ウェア・デバッグ	: 八一下	ウェアを使用して	デバッグする		
▼ シミコ	レーターを使用し	てデバッグする :シミュ	レーターを使用し	てデバッグする		
<u> </u>	-ス (テハックしな	ハ) : ナハッ	ク情報のないノロ	ジェクト		
ビルド構成 クトはア? ードウェブ 成されます	伐は選択したデバッ ウティブな構成(ク ♡ (E1 (RX)) 、お。 す。	グ・モード・オブションの ルーブから選択された最初 こびデバッグ・ターゲット (	プロジェクトにの の構成)用にビ川 R5F564MLCxFC	み作成されます。たた ドされます。選択した をもとに、ユーザーの	ビレデフォルトでは、 ニデバイス (RX600) Dためのデバッグ構成	そのプロジェ 、デバッグ・ハ 成が自動的に作
1	< 戻る( <u>B</u> )	次へ(N)	>	終了( <u>E</u> )		キャンセル

図 3-2 新規プロジェクト作成ウィザード (2/4)

- (3) [ツールチェーン・バージョン]、[デバッグ・ハードウェア]、[ターゲットの選択]を指定してください(例: ツールチェーン・バージョン: "v2.04.00"、デバッグ・ハードウェア: "E1"、ターゲットの選択: "RX64M (176 ピンデバイス,型番: R5F564MLCxFC)")。[次へ(N)>]をクリックして次へ進みます。
  - 注意: E1 と同様に "E2 Lite" も [デバッグ・ハードウェア] のプルダウンメニューから選択できます。

ー e2 studio - プロジェクト生居 コード生成と FITの設定	ŝ	
□ コード生成を使用する □ FITモジュールを使用する	「FITモジュールのダウンロード」	
e2 studio コード生成は GUI A/D コンバーター、 DMA れ、周辺地能の初期化に制限 従来、「マイコン初期設定」 ースの共通化1 などの情報が の変更が必要でした。 Firmware Integration Ted 局に組み込むことを可能にし エースが共通化されているた することができます。	で設定することにより、自動的に MCU 周辺機能(クロック、タイマー、 コントローラーはど)のプログラム(デ)(イス・ドライバー)を生成します。 を加えることはありません。 (ターグットボートの支援方法)「ファイル構成」「観察名」「ユーザア サンプルコードごとに異なっている場合が多く、ユーザアプリケーション nology (以下FIT)では、ごれら情報のルール化により、ユーザアプリケー ました。また、FITIに対応した周辺機能ドライバやミドルウェアは、ユー め、現在使用中のRXマイコンから別のRXマイコンへ移行する場合もユー	シリアル・インターフェース、 機能は、APIとして提供さ プリケーションとのインタフェ へ組み込む際にサンプルコード ションへ&サンプルコードを含 ザアプリケーションとされいを打 がアプリケーションを含易に移植
	User Application	
CG Device Drive	Middleware RTOS Device Driver (BSP)	
	MCU	
⑦ < 戻る( <u>B</u> )	次へ(N) > 終了(E)	キャンセル

図 3-3 新規プロジェクト作成ウィザード (3/4)

- (4) [コード生成を使用する] または[FIT モジュールを使用する]オプションが選択可能なデバイスでは、 この画面でチェックマークをつけることでそれぞれの機能が利用可能なプロジェクトが作成できます。 [FIT モジュールを使用する]をチェックした場合は、更にここでモジュールのダウンロード操作が行え ます(FIT は RX ファミリでのみ利用可能です)。
- 注: コード生成の利用可否は「インストール詳細」ダイアログの「Renesas デバイス・サポート」で確認できます。

要約	×
プロジェクトの要約:	
プロジェクト・ジェネレーター プロジェクト名: Tutorial プロジェクト・ディレクトリー: D:¥Workspac CPU シリーズ: RX600 CPU 型名: RX64M ツールチェーン名: Renesas RXC Toolchain ツールチェーン・パージョン: v2.03.00	Def.
生成ファイル:	
D:¥Workspace¥Tutorial¥src¥Tutorial.c Main Program D:¥Workspace¥Tutorial¥src¥dbsct.c Setting of B and R sections	<del></del>
'OK' をクリックしてプロジェクトを生成するか、 'キャン をクリックして中止してください。 OK キャ	ンセル' 'ンセル

図 3-4 新規プロジェクト作成ウィザード (4/4)

(5) 最後に[終了(E)]ボタンを押すとプロジェクトの要約が表示されます。更に[OK] をクリックするとプロジェクトが生成され、下記の画面例のように表示されます。

C - Tutorial/src/Tutorial.c - e2 studio	S 42/75 115/5		-> 1°±000		
ファイル(E) 編集(E) ソース(S) リファクタリ	リンク(T) ナヒケー	ト(N) 検索(A) プロジェクト(P) Renesas Views 美行(R) ワ	イントワ( <u>W</u> )	ヘルフ( <u>H</u> )	
📑 🛨 🔚 🕒   🗞 🔺 🗞 🖬 : 🖉	: "I : Ø : C : C	· • 🚳 • 🖻 • 6° • : 🎋 • 🜔 • 💁 • : 😂 🗁 🖋 • : 🗾	9 🗉 🗓 : 🖞	- 🖓 🔹 🏷 🤤	• => •
				クイック・ア	クセス 🛛 🖻 🔂 🖬
🎦 プロジェクト・エクスプロ 🙁 🔍 🗖	Interial.c ⋈		- 8	<b>₽ צ ⊚</b> м	I 🗐 Ø 🖓 🗖
🖻 🔄 👘 🗸	3	⊕ /* PROJECT NAME : Tutorial	<b>^</b>	1 E I	ª₂ ì≷ №° ● ¥€ ▽
a 😂 Tutorial [HardwareDebug]	13			🕂 main(voi	d) : void
Includes	14			<ul> <li>main(voi</li> </ul>	d) : void
4 📇 src	15	⊕ /* FILE :Main.c or Main.cpp		💉 abort(vo	id) : void
▷ c dbsct.c	27	⊖#ifdefcplusplus			
interrupt_handlers.c	28	⊖//#include <u><ios></ios></u> // Remove t	the con		
iodefine.h	30	#endif	ne con =		
reset_program.c	31				
⊳ 💽 sbrk.c	32	vold main(vold); ⊖#ifdef colusplus	_		
⊳ 🖻 sbrk.h	34	extern "C" {			
In stacksct.h	35	void abort(void);			
Tutorial.c	37	ر #endif			
In typedefine.h	38				
b ln vect.h	39	void main(void)			
vector_table.c	41	L.			
🗟 custom.bat	42	}	-		
📄 Tutorial Debug.launch		<	+		
Tutorial HardwareDebug.launch	🔝 問題 🛛 🧔	タスク 📃 コンソール 🔲 プロパティー 🔋 Memory Usage 🐚 ス	タック解析	쪶 スマート・ブラウ	ਸੁ– 🍃 ⊽ 🗖 🗖
	0 項目				
	記述/説明	*		リソース	パス
	•	III			
🔝 書き込	み可能	スマート挿入			1

図 3-5 作成された新しい C プロジェクト

(6) この例では「Tutorial」という新しい C プロジェクトが生成された様子を示します。

このプロジェクトは、"Tutorial.c" というアプリケーションファイルと、標準的なスタートアップファイル (dbsct.c, interrupt\_handlers.c, sbrk.c など) から構成されています。[プロジェクト・エクスプローラー] パネルではすべ てのプロジェクトおよびソースファイルを Windows エクスプローラーと同様のフォルダ階層として表示します。

#### プロジェクトをバックアップする際の注意事項:

- 「.」(ドット)で始まる名前のファイルやフォルダ(例:.project)にはプロジェクトの設定情報が含まれますので、バックアップを取る際にはこれらのファイルやフォルダを含めてプロジェクトのフォルダ全体を圧縮するなどしてください。
- 他のプロジェクトのファイルを参照する設定など、プロジェクト間で共有される設定を保存するためには、 ワークスペース全体をバックアップする必要があります。

Г

#### 3.2. ワークスペースへの既存プロジェクトのインポート

本節では、既存のプロジェクトをディレクトリまたはアーカイブからワークスペースへインポートする手順を説 明します。

選択 アーカイブ・ファイルまたはディレクトリーから新規プロジェクトを作成します。	
アーガイン・ファイルよんはノイレントシールの制成ノロシエクトでTFMしより。	
インポート・ソースの選択( <u>S</u> ):	
フィルタ入力	
▲ 🦢 ─般	<b>^</b>
Sconvert CCRX to GNURX Project	
DS-5 KPIT GNUARM-RZ/NONE Project	
W HEW フロンエクト Pename & Import Existing C/C++ Project into Workspace	E
■ Renesas共通プロジェクト・ファイル	
 ↓ アーカイブ・ファイル	
────────────────────────────────────	
1. 設定	
	-
(?) < 戻る(且) 次へ(N) > 終了(	(E) キャンセル

#### 図 3-6 既存プロジェクトインポートウィザード

(1) e<sup>2</sup> studio で、[ファイル] → [インポート] をクリックし、HEW プロジェクトインポートウィザードを開きます。「既存プロジェクトをワークスペースへ」を選択し、[次へ(N)>] ボタンをクリックして [プロジェクトのインポート] ウィンドウを開いてください。

プロジェクトのインポート 既存の Eclipse プロジェクトを検索するディレクトリーを選択します。		
◎ ルート・ディレクトリーの違訳(工): D:¥SampleProjects	•	参照( <u>R</u> )
◎ アーカイブ・ファイルの選択(A):	-	参照( <u>R</u> )
プロジェクト( <u>P</u> ):		
<ul> <li>図 33NRSK (D:¥SampleProjects¥63NRSK 直接アクセス)</li> <li>DC12_Oneshot (D:¥SampleProjects¥Samples¥ADC12_Oneshot 直接アクセス)</li> <li>DC12_Repeat (D:¥SampleProjects¥Samples¥ADC12_Repeat 直接アクセス)</li> <li>Application (D:¥SampleProjects¥Application 直接アクセス)</li> <li>Async_Serial (D:¥SampleProjects¥Samples¥ASync_Serial 直接アクセス)</li> <li>AN_API_Demo_RSKRX63N (D:¥SampleProjects¥Samples¥CASYnc_Serial 直接アクセス)</li> <li>Ch (D:¥SampleProjects¥Samples¥CBC12_B#20/202)</li> <li>Ch (D:¥SampleProjects¥Samples¥CBC12_B#20/202)</li> <li>Ch (D:¥SampleProjects¥Samples¥CBC12_B#20/202)</li> <li>Ch (D:¥SampleProjects¥Samples¥CBC12_B#20/202)</li> <li>AC (D:¥SampleProjects¥Samples¥DAC12_B#20/202)</li> <li>MAC (D:¥SampleProjects¥Samples¥DMAC 直接アクセス)</li> <li>ATŹS=&gt;&gt;</li> </ul>	× H	すべて選択( <u>S</u> ) 選択をすべて解除( <u>D</u> ) 更新( <u>E</u> )
<ul> <li>□ ネストしたプロジェクトを検索(L)</li> <li>□ プロジェクトをワークスペースにコピー(C)</li> <li>□ ワークスペースに罠に存在するプロジェクトを通す(L)</li> <li>ワーキング・セット</li> <li>□ ワーキング・セットにプロジェクトを追加(I)</li> <li>□ ワーキング・セット(Q):</li> </ul>	•	選択(E)
⑦ <戻る(B) 次へ(M) > 終了(E)		キャンセル

#### 図 3-7 e<sup>2</sup> studio の [プロジェクトのインポート] ウィンドウ

(2) プロジェクトが保管されているディレクトリまたはアーカイブを検索して選択します。既存のプロジェクトがすべて表示されますので、インポートするプロジェクトを選択してください。[終了(F)] をクリックしてプロジェクトのインポートを完了します。

C - Tutorial/main.c - e2 studio ファイル(F) 編集(E) ソース(S) リス	ファクタリング(T) ナ	ピゲート(N) 検索(A) プロジェクト(P) Renesas Views 車行(R) ウィンドウ(	W) ヘルプ(H)
± • = 6 ≥   ⊗ • 6 • 6 :	X 🖻 🖺 🖉	☐ [ @ • @ • ] • @ • ] <b>☆</b> • ] <b>⊅</b> • ] <b>⊅</b> ⊕ <i>A</i> • ] <b>≯</b> □ 1 2-	1   2 × 7 × ← ← × → (ック・アクセス   昭   ��0
<ul> <li>プロジェクト・エクス ※</li> <li>□</li> <li>○</li> <li>② Power_Off</li> <li>○</li> <li>○</li></ul>	<ul> <li>main.c x3</li> <li>699</li> <li>71</li> <li>72</li> <li>73</li> <li>75</li> <li>77</li> <li>78</li> <li>79</li> <li>80</li> <li>81</li> <li>82</li> <li>83</li> <li>84</li> <li>85</li> <li>87</li> <li>89</li> </ul>	<pre></pre>	E 7 ∞ 2 C C C C C C C C C C C C C C C C C C
	E 10 問題 23 20 0 errors, 12 wa	) タスク 💿 コンソール 🔝 ブロパティー 🔋 Memory Usage 🛐 スタック解析 ( mings, 0 others	

#### 図 3-8 e<sup>2</sup> studio でのプロジェクトのインポート

(3) e<sup>2</sup> studio へのプロジェクトのインポートが完了しました。

既存のプロジェクト名のままインポートするのでなく、プロジェクト名を変更してインポートすることもできます。 このオプションを使用する場合、一度にインポートできるプロジェクトは一つのみです。

選択			
Rename and Import and Existing C/C++ Project	t into the workspace		Ľ
インポート・ソースの選択( <u>S</u> ):			
フィルタ入力			
<ul> <li>★ KX</li> <li>★ Convert CCRX to GNURX Project</li> <li>★ DS-5 KPIT GNUARM-RZ/NONE Project</li> <li>★ HEW プロジェクト</li> <li>★ Rename &amp; Import Existing C/C++ Pro</li> <li>★ Renesas共通プロジェクト・ファイル</li> <li>▶ アーカイブ・ファイル</li> <li>● ファイル・システム</li> <li>※ 取存プロジェクトをワークスペースへ</li> <li>※ 設定</li> <li>▶ ● C</li> </ul>	t oject into Workspace		
? < 戻る(旦) 2	欠へ( <u>N</u> ) >	終了(E)	キャンセル

#### 図 3-9 名称変更とプロジェクトのインポートウィザード

 (1) e<sup>2</sup> studio で、[ファイル] → [インポート] をクリックし、HEW プロジェクトインポートウィザードを開きます。"Rename & Import Existing C/C++ Project into Workspace"を選択し、[次へ(N)>] ボタンを クリックして [名称変更とプロジェクトのインポート] ウィンドウを開いてください。

プロジェクト名(P): Tutorial_NewName         ● ルート・ディレクトリーの選択(T): D:¥SampleProjects         ● アーカイブ・ファイルの選択(A):         ● アーカイブ・ファイルの選択(A):         プロジェクト(P):         Timer_Capture (D:¥SampleProjects¥Samples¥Timer_Capture)         Timer_Compare (D:¥SampleProjects¥Samples¥Timer_Compare)         Timer_Levent (D:¥SampleProjects¥Samples¥Timer_Event)         Timer_Mode (D:¥SampleFrojects¥Samples¥Timer_Kode)         Tutorial (D:¥SampleFrojects¥Samples¥UImer_Mode)         UP_Demo (D:¥SampleFrojects¥Samples¥UIP_Demo)         USB_Host_HID_Demo (D:¥SampleProjects¥Samples¥USB_Host_HID_Demo)
<ul> <li>● ルート・ディレクトリーの選択(1): D:¥SampleProjects</li> <li>● アーカイブ・ファイルの選択(A):</li> <li>● アーカイブ・ファイルの選択(A):</li> <li>● 夢照(B)</li> <li>プロジェクト(P):</li> <li>Timer_Capture (D:¥SampleProjects¥Samples¥Timer_Capture)</li> <li>Timer_Event (D:¥SampleProjects¥Samples¥Timer_Compare)</li> <li>Timer_Event (D:¥SampleProjects¥Samples¥Timer_Event)</li> <li>Timer_Mode (D:¥SampleProjects¥Samples¥Timer_Levent)</li> <li>Timer_Mode (D:¥SampleProjects¥Samples¥Timer_Mode)</li> <li>Tutorial (D:¥SampleProjects¥Samples¥UIP_Demo)</li> <li>USB_Host_HID_Demo (D:¥SampleProjects¥Samples¥USB_Host_HID_Demo)</li> </ul>
● アーカイブ・ファイルの選択(A): ● 参照(B) プロジェクト(P): Timer_Capture (D:¥SampleProjects¥Samples¥Timer_Capture) Timer_Compare (D:¥SampleProjects¥Samples¥Timer_Compare) Timer_Event (D:¥SampleProjects¥Samples¥Timer_Event) Timer_Mode (D:¥SampleProjects¥Samples¥Timer_Mode) Tutorial (D:¥SampleProjects¥Samples¥UIP_Demo) USB_Host_HID_Demo (D:¥SampleProjects¥Samples¥UIP_Demo)
Timer_Capture (D:¥SampleProjects¥Samples¥Timer_Capture) Timer_Compare (D:¥SampleProjects¥Samples¥Timer_Compare) Timer_Event (D:¥SampleProjects¥Samples¥Timer_Event) Timer_Mode (D:¥SampleProjects¥Samples¥Timer_Mode) Tutorial (D:¥SampleProjects¥Tutorial) uIP_Demo (D:¥SampleProjects¥Samples¥uIP_Demo) USB_Host_HID_Demo (D:¥SampleProjects¥Samples¥USB_Host_HID_Demo)
USB_Host_MSC_Demo (D:¥SampleProjects¥Samples¥USB_Host_MSC_Demo) Voltage_Detect (D:¥SampleProjects¥Samples¥Voltage_Detect)

図 3-10 e<sup>2</sup> studio の [名称変更とプロジェクトのインポート] ウィザード

(2) プロジェクトが保管されているディレクトリまたはアーカイブを検索して選択します。既存のプロジェクトがすべて表示されますので、インポートするプロジェクトを選択し、新しい名称を入力してください。
 [終了(F)] をクリックしてプロジェクトのインポートを完了します。

3 ▼ II I	x 12 Ø 8		(₩) (₩) ■ ♥ ▼ ♦ クイック・アクセ	=) ▼ ♥= �= ▼ = 2ス   🖻   🖫
J□ジェクト・エクス S       □         Includes       > src         Includes       > flashLED.c         Includes       > flashLED.c         Includes       > intprg.c         Includes       > intprg.c         Interset       > interset         Interset       > interset	main.c ☆          69         71         72         73         75         77         78         79         80         81         82         83         84         85         87         89         90         91         93         45	<pre>* System Includes [] /* Following header file provides C99 type definitions. * #include <stdint.h> * User Includes (Project Level Includes)[] /* Defines MCU registers */ #include "iodefine.h" /* LCD controlling function prototypes &amp; macro definition #include "lod.h" /* Timer &amp; ADC function definitions */ #include "lod.h" /* LD flash function definitions */ #include "flashLED.h" * Macro Definitions[] /* Device nickname definition */ #define NICKNAME "RXG3N " * Function Prototypes[] /* Static Text function declaration */ </stdint.h></pre>	□         □	**2 atdint.h bdefine.h cd.h imeradc.h lashLED.h UICKNAME Static_Test(void) Timer_Delay(unin gConstStr : unin ReplaceStr : unin ReplaceStr : unin anin(void) : void Static_Test(void)
DefaultSession.hsf Description.txt flashLED.h h hwsetup.h	0項目 記述/説明 く		リソース	<u>р</u>

図 3-11 e<sup>2</sup> studio での名称変更とプロジェクトのインポート

(3) プロジェクトの名称変更と e<sup>2</sup> studio へのインポートが完了しました。

#### 3.3. HEW プロジェクトのインポート

本節では、既存のプロジェクトワークスペースを e<sup>2</sup> studio に移行する HEW プロジェクトのインポート機能を 説明します。これにより、HEW で作成した既存のアプリケーションプログラムやワークスペースを再利用する ことができます。

ローカル・ファイル・システムからワークスペースに HEW プロジェクトをインボートします。 インボート・ソースの選択(5): フィルタ入力 ② 一般 ③ Convert CCRX to GNURX Project ③ DS-5 KPIT GNUARM-RZ/NONE Project ④ HEW プロジェクト ④ Rename & Import Existing C/C++ Project into Workspace ④ Renaset 通プロジェクト・ファイル ④ アーカイブ・ファイル ④ ファイル・システム ④ 既存プロジェクトをワークスペースへ 圓 設定	選択	
インボート・ソースの選択( <u>S</u> ): フィルタ入力 ・ DS-5 KPIT GNUARM-RZ/NONE Project ・ DS-5 KPIT GNUARM-RZ/NONE Project ・ Reneas共通プロジェクト ・ アーカイブ・ファイル ・ アーカイブ・ファイル ・ ファイル・システム ※ 既存プロジェクトをワークスペースへ ■ 設定	ローカル・ファイル・システムからワークスペースに HEW プロジェクトをインポートします。	
フィルタ入力         ● 一般	インポート・ソースの違択( <u>S</u> ):	
<ul> <li>● 一般</li> <li>◆ Convert CCRX to GNURX Project</li> <li>◆ DS-5 KPIT GNUARM-RZ/NONE Project</li> <li>◆ HEW プロジェクト</li> <li>○ Rename &amp; Import Existing C/C++ Project into Workspace</li> <li>◆ Rename &amp; Import Existing C/C++ Project into Workspace</li> <li>◆ Rename &amp; Import Existing C/C++ Project into Workspace</li> <li>◆ Rename &amp; Import Existing C/C++ Project into Workspace</li> <li>◆ Rename &amp; Import Existing C/C++ Project into Workspace</li> <li>◆ Rename &amp; Import Existing C/C++ Project into Workspace</li> <li>◆ Rename &amp; Import Existing C/C++ Project into Workspace</li> <li>◆ Rename &amp; Import Existing C/C++ Project into Workspace</li> <li>◆ Rename &amp; Import Existing C/C++ Project into Workspace</li> <li>◆ Rename &amp; Import Existing C/C++ Project into Workspace</li> <li>◆ Rename &amp; Import Existing C/C++ Project into Workspace</li> <li>◆ Rename &amp; Import Existing C/C++ Project into Workspace</li> <li>◆ Rename &amp; Import Existing C/C++ Project into Workspace</li> <li>◆ Rename &amp; Import Existing C/C++ Project into Workspace</li> <li>◆ Rename &amp; Import Existing C/C++ Project into Workspace</li> <li>◆ Rename &amp; Import Existing C/C++ Project Into Workspace</li> <li>◆ Rename &amp; Import Existing C/C++ Project Into Workspace</li> <li>◆ Rename &amp; Import Existing C/C++ Project Into Workspace</li> <li>◆ Rename &amp; Import Existing C/C++ Project Into Workspace</li> <li>◆ Rename &amp; Import Existing C/C++ Project Into Workspace</li> <li>◆ Rename &amp; Import Existing C/C++ Project Into Workspace</li> <li>◆ Rename &amp; Import Existing C/C++ Project Into Workspace</li> <li>◆ Rename &amp; Import Existing C/C++ Project Into Workspace</li> <li>◆ Rename &amp; Import Existing C/C++ Project Into Workspace</li> <li>◆ Rename &amp; Import Existing C/C++ Project Into Workspace</li> <li>◆ Rename &amp; Import Existing C/C++ Project Into Workspace</li> </ul>	フィルタ入力	
	<ul> <li>Convert CCRX to GNURX Project</li> <li>DS-5 KPIT GNUARM-RZ/NONE Project</li> <li>W HEW プロジェクト</li> <li>Rename &amp; Import Existing C/C++ Project into Workspace</li> <li>Rensas共通プロジェクト・ファイル</li> <li>アーカイブ・ファイル</li> <li>ファイル・システム</li> <li>設定</li> </ul>	F

図 3-12 HEW プロジェクトインポートウィザード

- (1) e<sup>2</sup> studio で、[ファイル] → [インポート] をクリックし、HEW プロジェクトインポートウィザードを開きます。"HEW プロジェクト"を選択し、[次へ(N)>] ボタンをクリックして [HEW インポート] ウィンドウを開いてください。
- (2) HEW プロジェクトファイル (.hwp) を検索して選択し、[終了(F)] ボタンをクリックしてこのプロジェクトをインポートします。

ファイル( <u>F</u> ) 編集( <u>E</u> ) ソース( <u>S</u> )	リファク	フタリング(T) ナビ	<u>ビ</u> ゲート( <u>N</u> ) 検索( <u>A</u> ) プロジェクト( <u>P</u> ) Renesas <u>V</u> iews 実行( <u>R</u> ) ウィンドウ	(W) ヘルプ(H)
📑 • 🗏 🕞 🕭 🗍 🗞 • 🔨 •	K 📑	X 19 0 C	👌 • 🖄 • 👌 • 🎯 •   🎄 • 🔿 • 🎭 • 😂 🖨 • 🍞 🗐 🖷	
			<u>7</u> -	イック・アクセス 🔹 📴
🏠 プロジェクト・エクス 🛛	20	🝺 main.c 🛿	- 0	ד א איצע די בי די
n 🖘	59 V	69	⊕* System Includes	P □ 1 N N N ● ₩
🔺 😂 Tutorial [Debug]	<b>^</b>	71	<pre>/* Following header file provides C99 type definitions. ' #include catding header</pre>	▽
▷ 🖑 バイナリー		73	#Include (Scaline.in)	stdint h
Includes		75	⊕ * User Includes (Project Level Includes)	indefine b
		77	/* Defines MCU registers */	lodeline.n
> Debug		78	#include "loderine.n" /* LCD controlling function prototypes & macro definition	lica.n
		80	#include "lcd.h"	timeradc.h
▷ <u>c</u> flashLED.c		81	/* Timer & ADC function definitions */	flashLED.h
▷ In flashLED.h		82	<pre>#include "timeradc.h"</pre>	# NICKNAME
hwsetup.c	=	84	#include "flashLED.h"	+ Static_Test(void) :
hwsetup.h		85		+ Timer_Delay(uint32
id_code.c		87		C aConstStr : const u
intprg.c		89	/* Device nickname definition */	gReplaceStr : uint8
iodefine.h		90	#detine NICKNAME KAOSN	<ul> <li>grtepideesti - dirteg</li> <li>main(woid) - woid</li> </ul>
		93	* Function Prototypes.	Chable Test(usid)
Idd b		95	/* Static Test function declaration */	<ul> <li>Static_lest(void) :</li> </ul>
		96	<pre>void Static_Test(void); /* Times Delay function prototype */</pre>	
▷ In Iowsrc.n		98	void Timer Delay(uint32 t, int8 t):	
⊳ l.c main.c			4 III >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>	< III
Id resetprg.c		周期 尚 55     6		
▷ 🔝 rskrx63ndef.h		🔟 1928 🚈 '97	マロンフロン ID A E フロバワイ 『Methody Osage 国 スタック解析	
b c sbrk.c				🌺 🚮   🛃 🖳 🔻 📩 🕶 AC 👻
b In sbrk.h		Import wizard n	nessages	
In stacksct.h				
switch.c		ツールチェーン・バー:	ジョン 'RXC_v1.02.00' がブロジェクト 'Tutorial' にありません。プロジェクトはバージョン 'I	RXC_v2.03.00' に更新されます。
⊳ 🔚 switch.h				
b imeradc.c				
	<b>_</b>	<		•
81	書き込	み可能	スマート	

図 3-13 HEW から e<sup>2</sup> studio へのプロジェクトの移行

(3) HEW プロジェクトの e<sup>2</sup> studio へのインポートが完了しました。

変換の後、すべてのオリジナルプロジェクトとソースファイルは、新たに作成されたプロジェクトワークスペースとともに e<sup>2</sup> studio に保存されます。さらに、".cproject"、".\*linker"、".info"及び ".project" が作成され、追加されます。

HEW と e<sup>2</sup> studio プロジェクトのワークスペースは物理的に同じ場所のファイルを共有します。

HEW プロジェクトのインポートが失敗した場合、以下の2つの前提条件を確認してください。

- (i) HEW プロジェクトワークスペースはバージョン v4.07 以降でなければなりません。
- (ii) 再度 HEW プロジェクトをインポートする場合はその前に、".cproject"、".\*linker"、".info" 及び ".project" などのファイルを手動で削除しておく必要があります。
- 参考: v4.07 より古い HEW プロジェクトのワークスペースをインポートしたい場合は、まず HEW v4.07 以降を使用してワークスペースを更新し、その後にインポートしてください。

## 3.4. CS+プロジェクトのインポート

プログラムを再利用するために、本節では、既存のプロジェクトワークスペースを e<sup>2</sup> studio に移行する CS+のインポート機能を説明します。

選択 Renesas共通プロジェクト・ファイル・フォーマットで保管されたプロジェクトを Cubesuite+ からインボートします。          インボート・ソースの選択(S):         フィルタ入力         ● 一般         ● Convert CCRX to GNURX Project         ● DS-5 KPIT GNUARM-R2/NONE Project         ● HEW プロジェクト         ● Renesas共通プロジェクト・ファイル         ● Renesas共通プロジェクト・ファイル         ● TO-7 イフィイル         ● TO-7 イフィブル         ● ファイル・システム         ● 酸存プロジェクトをワークスペースへ         ● DS         ● C         ● C         ● C         ● C
インボート・ソースの選択(5):         フィルタ入力         ● 一般         ● Convert CCRX to GNURX Project         ● DS-5 KPIT GNUARM-R2/NONE Project         ● NHW プロジェクト         ● Remeas非通プロジェクト         ● アーカイブ・ファイル         ● アーカイブ・ファイル         ● アーカイブ・ファイル         ● アーカイブ・ファイル         ● アーカイブ・ファイル         ● アーカイブ・ファイル         ● 欧作プロジェクトをワークスペースへ         ● 設定         ● C
<pre>インボート・ソースの選択(S): フィルタスカ Convert CCRX to GNURX Project Convert CCRX to GNURX Proj</pre>
フイルタ入力
<ul> <li>         ・ 一般         <ul> <li></li></ul></li></ul>

#### 図 3-14 CS+プロジェクトインポートウィザード

e<sup>2</sup> studio で、[ファイル] → [インポート] をクリックします。 "Renesas 共通プロジェクト・ファイル" を選択し、 [次へ(N)>] ボタンをクリックしてください。[プロジェクトのインポート] ウィンドウが開きます。

(1) CS+プロジェクトファイル (.rcpe) を検索して選択し、[終了(F)] ボタンをクリックしてこのプロジェクト をインポートします。

ファイル(E) 編集(E) ソース(S) リ	ファク	タリング(T) ナト	ビゲート( <u>N</u> ) 検索( <u>A</u> ) プロジェクト( <u>P</u> ) Renesas <u>V</u> iews 実行( <u>R</u> ) ウィンドウ( <u>W</u> ) ヘルプ(	<u>H</u> )
📑 🕶 📄 💿 🗢 💧 🕶 🚳	<b>D</b>	x 🖭 🖉 🕿	1 🚓 + 🖄 + 🖻 + 🞯 + 🕸 + 💽 + 🏰 + 🙋 🖨 🖉 + 🍞 🗐 🖷 🕴 + 👘	• <=
				zス 🔡 🖬 🖬 🕻
🎦 プロジェクト・エクス 🛛 🖓	ÞΥ	💽 main.c 🛿	- 🗆 🔚 7 🛙	»2
	$\bigtriangledown$	2	DISCLAIMER[]	4. N N N N N
▲ States and a state of the		22	/* Copyright (C) 2012 Renesas Electronics Corporation. A.	
⊳ 🔊 Includes		65	* History : 13 Aug. 2012 Ver. 1.00 First Release. =	tdint h
b 🕞 STC		67		odofino h
b dbsct.c		69	⊕ * System Includes	Juenne.n
Fight ED c		71	#include <stdint.h></stdint.h>	bain .
b busctup c		73	u t	imeradc.n
b D id code e		75	* User Includes (Project Level Includes)	lashLED.h
▷ [c] Id_code.c		78	/* Defines MCD registers */ # N #include "iodefine.h"	VICKNAME
⊳ ie intprg.c		79	/* LCD controlling function prototypes & macro definition 🕂 🕂 S	<pre>static_Test(void) :</pre>
▷ [c] lcd.c		80	#include "lcd.h"	Timer_Delay(uint32
⊳ 💽 main.c		81	/* Timer & ADL function definitions */	JConstStr : const u
Resetprg.c		83	/* LED flash function definitions */	ReplaceStr : uint8
b 🖻 sbrk.c		84	<pre>#include "flashLED.h"</pre>	nain(void) : void
iswitch.c		85	⊕ * Macro Definitions∏ ● S	Static_Test(void) :
timeradc.c	=	89	/* Device nickname definition */	
k letter lett		90	#define NICKNAME "RX63N "	
DefaultSession.hsf		91		
Description.txt				
M flashLED.h		🛃 問題 🍃 タス	スク 📮 コンソール 😫 🔲 プロパティー 🔋 Memory Usage 🐚 スタック解析 虆 スマート・	・ブラウ 🎴 🗖
M hwsetup.h			R. 🖬 🚽	E → → ∧C →
Miodefine.h	,	Import wizard r	nessages	
R led b				
lowerc h				
BRK Header byt				
SK_Heddel.ckt	-			
<				Þ
A =	a=:1 7/	THE	77 h'	1.1

図 3-15 CS+から e<sup>2</sup> studio へのプロジェクトの移行

(2) CS+プロジェクトの e<sup>2</sup> studio へのインポートが完了しました。

変換の後、すべてのオリジナルプロジェクトソースファイルは、新たに作成されたプロジェクトワークスペース とともに e<sup>2</sup> studio に保存されます。さらに、".cproject"、".\*linker"、".info" 及び ".project" が作成され、追加 されます。

CS+と e<sup>2</sup> studio プロジェクトのワークスペースは物理的に同じ場所のファイルを共有します。

# 第4章 ビルド

この章では、e<sup>2</sup> studio 統合開発環境のためのビルド構成と主要なビルド機能について説明します。

# 4.1. ビルドオプションの設定

プロジェクトを作成するとデフォルトのビルドオプション設定が生成されます。通常このデフォルトオプションで プロジェクトをビルドできます。しかしビルドオプションの変更が必要な場合は(ツールチェーンのバージョンや 最適化オプションの変更など)、プロジェクトのビルド前に以下の手順で変更してください。



図 4-1 チュートリアルプロジェクトのプロパティと Tutorial.c ソースファイルのプロパティ

ビルドオプションは、プロジェクトのプロパティウィンドウまたはソースファイルのプロパティウィンドウで変更できます。

- (1) ① "Tutorial" プロジェクトまたは ② ソースファイル "Tutorial.c" を選択します。
- (2) アイコン 「「「をクリックして(あるいは右クリックして [プロパティ] を選択するか、ショートカットキーの [Alt]+[Enter] もしくは [Alt]+[T] を使って)プロパティダイアログを開いてください。
- (3) "C/C++ ビルド" オプションをクリックし、構成の設定を表示、編集してください。

プロパティウィンドウはワークスペース、プロジェクト、ソースの各レベル別に設定できますが、ワークスペース 内のファイル共通に適用できる項目の他に、個別のプロジェクトにはさらに多くの設定項目があります。

フィルタ入力	ツールチェーン・バージョンの変更	← ▼ ⇒ ▼
<ul> <li>&gt; リソース</li> <li>C/C++ ビルド Device Tool chain エディター</li> <li>ツールチェーン・バージョンの変更 ビルド変数 ロギング</li> </ul>	▲ プロジェクト名: Tutorial ツールチェーン名: Renesas RXC Toolchain 現在のバージョン: v2.03.00 有効なバージョン: v2.03.00 ▼	
依存関係スキャン 環境 設定	v2.03.00 v2.02.00 デフォルトの復元(T)	適用(L)
(?)	OK ‡†	ッンセル

図 4-2 ツールチェーン・バージョンの変更

(1) [C/C++ ビルド] → [ツールチェーン・バージョンの変更] をクリックして、ツールチェーン・バージョン を表示、変更してください。

[有効なバージョン]のドロップダウンリストからバージョンを選択できます。表示されるバージョンは 図 2-3 で示したツールチェーン管理の設定に依存します。

フィルタ入力	環境			¢	•
▶ UV-Z					
▲ C/C++ ピルド					
Device	構成: HardwareD	ebug [アクティブ]		-	構成の管理
Tool chain エディター					
ツールチェーン・パージョンの空	T T T T T T T T T T T T T T T T T T T				
デルド変数	設定する環境変数				ie to
ロギング	70.80	125	ф#		
依存関係スキャン	发数	18	田木	- Â	選択
環境	BIN_RX	C:¥PROGRA~1¥RENESA~2¥CS_~1	BUILD SYST		
	CONFIGDIR	<pre>\${workspace_loc:/\${ProjName}}*</pre>	BUILD SYST		編集
C/C++	CWD	D:¥Workspace¥Tutorial¥HardwareD	BUILD SYST		削除
タスク・リポジトリー	INC_RX	C:¥PROGRA~1¥RENESA~2¥CS_~1	BUILD SYST	=	
F1127-	PATH	C:¥PROGRA~1¥RENESA~2¥CS_~1	BUILD SYST		定義解除
プロジェクト参昭	PWD	D:¥Workspace¥Tutorial¥HardwareD	BUILD SYST		
実行/デバッグ設定	RXC_LIB	C:¥PROGRA~1¥RENESA~2¥CS_~1	BUILD SYST		
211/7/19782C	TCINSTALL	C:¥PROGRA~1¥RENESA~2¥CS_~1	BUILD SYST		
					]
	◎ ネイティフ環境/	へ変数を追加			
	◎ ネイティブ環境	を指定された環境と置換			
				1.000	
		テフォルトの	D復元(I)	週月	Ħ( <u>L</u> )
(?)		OK		キャン†	211
		UK			

図 4-3 コンパイラのビルド設定:環境

(2) [C/C++ビルド] → [環境] の順にクリックし、ビルドオプションを設定し、環境変数を追加、編集してく ださい。

ビルドオプションにより、環境変数を用いて指定したパス名を含むすべてのツールチェーンの構成設定を保存 することができます。現行のビルド構成は図 4-3 に示すように "HardwareDebug [アクティブ]"です。

ビルドオプションの詳細は、以下の場所にあるコンパイラユーザーズマニュアルを参照してください。 "{コンパイラをインストールしたディレクトリ}\doc"(例:C:\Program Files\Renesas\RX\2\_3\_0\doc\)

#### 4.2. サンプルプロジェクトのビルド

ァイル(F) 編集(E) ソース(S) リファクタリング(T)	ナビゲート(N) 検索(A) プロ	]ジェクト(P) 実行(R) Renesas \	iews ウィンドウ(V	/) ヘルプ(H)	
<ul> <li></li></ul>		プロジェクトを開く(E) プロジェクトを閉じる(S) すべてビルド(A) ビルド構成 プロジェクトのビルド(B)	Ctrl+B	▼ 初 ▼ や や ▼ → ○ □ □ 提 ア ⋈ 表示するフ	<ul> <li>クイック・アクセ</li> <li>の M 目 タ ロ I</li> <li>の M 目 タ ロ I</li> <li>い</li> </ul>
<ul> <li>) [1] Includes</li> <li>&gt; [2] src</li> <li>&gt; [2] dbsct.c</li> <li>&gt; [2] interrupt_handlers.c</li> <li>&gt; [4] reset_program.c</li> <li>&gt; [4] sbrk.h</li> <li>&gt; [6] sbrk.c</li> <li>&gt; [6] sbrk.h</li> <li>&gt; [6] sbrk.h</li> <li>&gt; [6] tutorial.c</li> <li>&gt; [6] typedefine.h</li> <li>&gt; [6] vector_table.c</li> </ul>	02 02 02	フーキング・セットのピルド(W)           クリーン(N)           目動的にピルド(M)           Make ターケット           Renesas Tool Settings           マイの依存閣係を更新           C/C++ Index           プロパティ(P)	Alt+T Alt+Q Alt+D	Λu	
Custom.bat     Tutorial Debug.launch     Tutorial HardwareDebug.launch	問題 22 🧔 タスク 🗐 コンソ 1日	ール 🔲 プロバティー 🔋 Memor	/ Usage 🔋 スタック	♪解析 QQ スマート・ブラウ	ff- 🎲 ⊽ 🗖
	2017年1月			<u> </u>	

図 4-4 "チュートリアル" プロジェクトのビルド例

- (1) e<sup>2</sup> studio 環境下において、"Tutorial" という名前で新規プロジェクトを作成します(あるいは既存の プロジェクトのいずれかを開きます)。
- (2) [プロジェクト・エクスプローラー] で、当該のプロジェクトをクリックします。
- (3) [プロジェクト] → [プロジェクトのビルド] の順でクリックするか、 <sup>6</sup> のアイコンをクリックして、本プロジェクトをビルドします。

[コンソール] に "ビルドが完了しました" というメッセージが表示されます。このビルドの最後に \${CONFIGDIR} ディレクトリに出力されたファイルは、"makefile"、"Tutorial.abs"、"Tutorial.map"、 "Tutorial.mot"、"Tutorial.x" などで構成されます。

"Tutorial.abs" はデバッグに使われる ELF/DWARF フォーマット(\*.abs)のルネサス標準ロードモジュールで す。GDB は別の ELF/DWARF 仕様(\*.x)のロードモジュールフォーマットをサポートしているため、e<sup>2</sup> studio のデバッグ用に "Tutorial.abs" を "Tutorial.x" に変換されます。

#### 4.3. ビルド構成のレポート

プロジェクトレポート機能により、プロジェクトとビルド構成を e<sup>2</sup> studio からファイルに出力して、プロジェクト やビルド環境の設定を容易にチェックし、比較することができます。



図 4-5 プロジェクトレポート

- (1) [プロジェクト・エクスプローラー] を右クリックしてコンテクスト・メニューを開きます。
- (2) [Save build setting report] を選択してビルド構成レポートを保存します。

# 第5章 デバッグ

この章では、e<sup>2</sup> studio 統合開発環境のデバッグ構成と主要なデバッグ機能の使い方について説明します。 以下の説明は、E1 エミュレータまたは E2 エミュレータ Lite と RSK RX64M ボードのハードウェア構成に基 づきビルドされた "Tutorial" プロジェクト(4.2 節)に関するものです。



図 5-1 [デバッグ] パースペクティブへの切り替え

(1) e<sup>2</sup> studio の "Tutorial" プロジェクトワークスペースを開き、[デバッグ] パースペクティブをクリックし ます。

パースペクティブはワークベンチの表示レイアウトを目的別に定義するものです。それぞれのパースペクティ ブは、ビュー、メニュー、ツールバーの組み合わせで構成され、パースペクティブを切り替えることで目的に適 した画面レイアウトに切り替えることができます。

例えば、[C/C++] パースペクティブには C/C++プログラム開発に必要なビューが定義され、[デバッグ] パー スペクティブにはプログラムのデバッグに必要なビューが定義されています。ユーザが [C/C++] パースペク ティブ表示中にデバッガに接続しようとする場合、e<sup>2</sup> studio は [デバッグ] パースペクティブに切り替えるよう ユーザを促します。

ワークベンチは複数のパースペクティブを持つことができ、それぞれカスタマイズしたり、新しいパースペクティブを追加することもできます。

注意:デバッグについての詳細は、6章で説明する「e<sup>2</sup> studio デバッグ・ヘルプ」を参照してください。

#### 5.1. 既存デバッグ構成の変更

初めてデバッグを行う際にはデバッグ構成(デバッグに必要な一連のオプション)を設定してください。デフォルトの設定内容は以下の操作で変更できます。



図 5-2 [デバッグ構成]ウィンドウを開く

(1) [プロジェクト・エクスプローラー] の "Tutorial" プロジェクトをクリックします。

[実行] → [デバッグの構成...] あるいは <sup>参</sup> <sup>\*</sup>アイコン(下向き矢印) → [デバッグの構成] の順にク リックし、[デバッグ構成] ウィンドウを開きます。

構成の作成、管理、および実行	ī		
3 🗎 🗙 🖻 ‡⊳ ▾	名前( <u>N</u> ): Tutorial HardwareDebug	共通( <u>C</u> ) 頃 ソース	
Debug-only GDB Simulator Debugging (5)	プロジェクト( <u>P</u> ):		
C GDB ハードウェア・デバッギ:	Tutorial		参照( <u>B</u> )
GHS Local C/C++ Launch Renesas GDB Hardware Atta	C/C++ アプリケーション: HardwareDebug¥Tutorial.x		
Tutorial HardwareDebug	変数	救(⊻) プロジェクトの検索( <u>H</u> )	参照( <u>R</u> )
💽 Renesas Simulator Debuggir	起動前に必要に応じてビルド		
	ビルド構成: Use Active		•
	◎ 自動ビルドを有効にする	◎ 自動ビルドを無効にする	
	◎ ワークスペース設定の使用	ワークスペース設定の構成	
< Ⅲ → フィルター一致: 8 / 12 項目		適用( <u>Y</u> )	前回保管した状態に戻す(⊻)
$(\mathbf{\hat{2}})$		デバッグ(D)	閉じる

図 5-3 ロードモジュールの選択

- (2) [デバッグ構成] ウィンドウで、"Renesas GDB Hardware Debugging" デバッグ構成の表示を展開し、既存のデバッグ構成をクリックしてください(例: "Tutorial HardwareDebug")。
- (3) [メイン] タブを選択し、プロジェクトビルドフォルダの中にあるロードモジュール "Tutorial.x" を追加 します。

名前(N): Tutorial HardwareDebug	
📄 メイン 🏇 Debugger 🕒 Sta	artup 🔲 共通( <u>C</u> ) 🦆 ソース
Debug hardware: E1 (RX)	▼ Target Device: R5F564ML

図 5-4 ターゲットデバイスの選択

- (4) [Debugger] タブに切り替え、Debug hardware を "E1"、Target Device を "R5F564ML" に設定し ます。
  - Debug Hardware: "E1"
  - Target Device: "R5F564ML"

📄 メイン 🕸 Debugger 🛛 🕨 Startup 🔲 共	(通(C) シース	
Debug hardware: E1 (RX)	Target Device: R5F564ML	
GDB Settings Connection Settings デバッ	ッグ・ツール設定	
▲ クロック		
メイン・クロック・ソース	EXTAL	-
EXTAL 周波数[MHz]	24.0000	
内部フラッシュメモリー書き換え時に	クロック・ソーはい	*
▲ ターゲット・ボードとの接続		
エミュレーター	(Auto)	
接続タイプ	JTag	-
JTag クロック周波数[MHz]	16.5	-
Fine ボーレート[Mbps]	2.00	~
ホット・プラグ	いいえ	-
▲ 電源		
エミュレーターから電源を供給する (М	1AX 200mA) いいえ	-
供給電圧	3.3V	+
▲ CPU 動作モード		
レジスター設定	シングルチップ	-
モード端子	シングルチップ・モード	-
⊿ 通信モード		
モード	デバッグ・モード	-

図 5-5 接続設定の変更

- (5) [Debugger] タブの中の [Connection Settings] サブタブを選択し、E1 エミュレータと RSK RX64M ボードの設定に基づいて、以下のように設定します。
  - クロック
     ・メイン・クロック・ソース = "EXTAL"
    - EXTAL 周波数[MHz] = "24.0000"
  - ターゲット・ボードとの接続
    - 接続タイプ = "JTag"
    - JTag クロック周波数[MHz] = "16.5"
  - 電源
    - エミュレータから電源を供給する(MAX 200mA) = "いいえ"
  - 通信モード

     モード = "デバッグ・モード"

"エミュレータから電源を供給する(MAX 200mA)"を "はい"に設定すると、エミュレータは外部 電源を使用することなくターゲットボードに電源を供給します(最大電流 200 mA)。

注意:図 5-5 のデバッグ構成は<u>例として示されています</u>。誤った設定は誤作動やハードウェアの故障の原因となりますので、接続の前にボードとエミュレータの設定を慎重に確認してください。



さらに e<sup>2</sup> studio には既存のプロジェクトデバッグ構成を新規プロジェクトに複製する機能があります。この機能は、同一のデバイスとデバッガ設定を用いるプロジェクトに使用できます。

Debug hardware: E1 (RX)   Target Devi	ce: R5F564ML
GDB Settings Connection Setting デバッグ・ツール	設定
⊿ IO	
デフォルト IO ファイル名を使用	(はい) ・
IO ファイル名	\${eclipse_home}¥internal¥IoFiles¥RX¥RX64M.sfp
▲ 中断	
ハードウェア・ブレークポイントの強制	(はい) ・
⊿ X∓U−	
エンディアン	リトル・エンディアン・
内部フラッシュメモリーの上書き	[0]
外部メモリー領域	[0]
ワーク RAM 開始アドレス	0x1000
ワーク RAM サイズ (Bytes)	0x500
▲ システム	
内蔵プログラムROMを書き換えるプログラムをデバ	いいえ・
内蔵データ・フラッシュを書き換えるプログラムを	いいえ・
▲ パフォーマンス・タイマー	
動作周波数 [MHz]	
▲ Start / Stop 機能設定	
ユーザー・プログラム実行前に関数を実行する	いいえ・
start 関数の開始アドレス	0x0
ユーザー・プログラム停止時に関数を実行する	いいえ
stop 関数の開始アドレス	0x0
ワーク RAM 開始アドレス	0x1000
ワーク RAM サイズ ( バイト )	0x230
▲ 外部フラッシュ	
ガウンロードな方がにする	11117

図 5-6 デバッグツール設定の変更

- (6) サブタブ [デバッグ・ツール設定] に切り替え、RSK RX64M ボードに基づいて以下を確認してください。
  - メモリ
     エンディアン = "リトル・エンディアン"
- (7) [適用] ボタンをクリックし、設定を確認します。
- (8) [デバッグ] をクリックし、デバッグ起動構成を実行して E1 または E2 Lite と RSK RX64M ボードに 接続します。

☆ デバッグ ☆	🍇 🔩 🔻 🎭 🌄 🕹 🕺   📩 🤜   i+   🍫 🔻 🗉 E
Tutorial HardwareDebug [Rene	esas GDB Hardware Debugging]
Thread #1 1 (single control of the second	e) (Suspended : シグナル : SIGINT:Interrupt) () at r_cg_resetprg.c:69 0xffc00000
📕 C:/Renesas/e2_studio_400	26a/DebugComp/rx-elf-gdbrx-force-64bit-double -rx-force-v2 (7.8.2)

図 5-7 [デバッグ] ビューのユーザターゲット接続

正しく接続できた場合は、図に示すような[デバッグ] ビュー画面が表示されます。プログラムのエントリポイントは、"r\_cg\_resetprg.c"の "PowerON\_Reset\_PC()" に設定されています。

#### 5.2. 新規デバッグ構成の作成

新規デバッグ構成を作成する簡単な方法は、既存の構成を複製する方法です。以下の手順で行います。

ノア11ル(F) 編集(E) ソース   ■ ■ ■ ■ ■ ■ ■ ■ ■		テビケート(N) 横	※(A) ノロジェクト(P) → = → 1 ◎31 / ペ1	) Kenesas	views 🧃	長行(K) ワイント	-: «:
	• 128 (010 ; « UP (1)		r to set let : W : 1	⊂ * 1	Tutorial H	HardwareDebug	7.
☆ デバッグ ☆	ial HardwareDebug [Renes	🍇 🔍 👻 🇞 🖏	ebuqqinq]		デバッグ(D) デバッグ の	) /構成(B)	
	·C:/Renesas/e2_studio_400 ·GDB server	026a/DebugComp/r	x-elf-gdbrx-force-64	bit-a d	う気に入りの	の編成(V) <sup>-1ユロリ</sup>	

図 5-8 [デバッグ構成]ウィンドウを開く

(1) [プロジェクト・エクスプローラー]の "Tutorial" プロジェクトをクリックします。

[実行] → [デバッグの構成...] あるいは 🌞 🔭 アイコン (下向き矢印) → [デバッグの構成] の順に クリックし、[デバッグ構成] ウィンドウを開きます。

	名前(N): Tutorial HardwareDebug (1 直接アクセス)	
イルタ入力	📄 メイン 🕸 Debugger 🍉 Startup 🔲 共通( <u>C</u> ) 🦤 ソース	
Debug-only GDB Simulator Debugging (SH, RL78, RH850)	プロジェクト( <u>P</u> ):	
■ GDB ハードウェア・デバッギング	Tutorial	参照( <u>B</u> )
GHS Local C/C++ Launch	C/C++ アプリケーション:	
Renesas GDB Hardware Attach	HardwareDebug¥Tutorial.x	
<ul> <li>Tutorial HardwareDebug</li> <li>Tutorial HardwareDebug</li> <li>Tutorial HardwareDebug (1 直接アクセス))</li> <li>Renesas Simulator Debugging (RX only)</li> </ul>	<u> </u>	参照( <u>B</u> )

図 5-9 選択したデバッグ起動構成の複製

- (2) [デバッグ構成] ウィンドウで、デバッグ構成 (例: "Tutorial HardwareDebug") を選択し
   コンをクリックします (現在選択しているデバッグ構成をコピーします)。新規デバッグ起動構成 (例: "Tutorial HardwareDebug (1)") が作成されます。
- (3) デバッグ構成は 5.1 節と同じ手順で変更できます。

#### RL78 デバッグ時の注意事項:

• ホット・プラグイン接続は、RL78/F1A, F13, F14, および F15 に対してのみサポートします。

#### 5.3. 基本的なデバッグ機能

本節では、e<sup>2</sup> studio がサポートする典型的なデバッグビューを説明します。

- 標準的な GDB デバッグ (Eclipse フレームワークによってサポートされている):ブレークポイント、式、 レジスタ、メモリ、逆アセンブル、変数
- 標準的な GDB デバッグの拡張:イベントポイント、IO レジスタ、トレース

[デバッグ・ツールバー] を開くには、プルダウンメニューボタンをクリックして [デバッグ・ツールバーを表示] を選択します。以下に示すのは、[デバッグ] ビューの便利なツールバーです。



図 5-10 デバッグビューの便利なツールバー

プログラムを実行するには 🕪 ボタンをクリックするか [F8] キーを入力します。

プログラムは、ブレークポイントで、あるいは 💷 ボタンをクリックすることで一時停止します。一時停止中 は以下の操作が可能です。

- 🐼 ボタンまたは [F6] キーは、現在実行中のプログラム行にある次の関数呼び出しをステップオ ーバ実行します(1行実行するが関数内には入らない)。
- 1> ボタンで、実行を再開します。

デバッグセッションの停止は、選択したデバッグセッション/プロセスを 📕 ボタンで終了するか、選択した プロセスとデバッガを 💦 ボタンで切断します。

他に以下のような操作が可能です。

- 🕹 ボタンは、エントリポイントからプログラムを再実行します(  $^{\infty}$   $\rightarrow$  🏴 の順にクリックするの と同じ)。
- 🏷 ボタンは、プログラムをパワーオンリセットのエントリポイントにリセットします。

#### 5.3.1. ブレークポイントの使い方

デバッガ接続中は、ソースコードまたは特定のアドレスにブレークポイントを設定することができます。エディタ や逆アセンブルビューには、下図に示すようにブレークポイントマーカを置くことができます。プログラムカウン タがブレークポイントの設定された行に対応する実行コードのアドレスに達すると、その位置の命令が実行さ れる前に中断(break)します。 e<sup>2</sup> studio ではソフトウェアブレークポイントとハードウェアブレークポイントを区 別して設定できます。エディタや逆アセンブルビューの左端でダブルクリックすると、デフォルトのブレークポイ ント型でブレークポイントが配置されます。ハードウェアブレークポイントのリソースが残っていない場合はデ バッグコンソールにブレークポイント型を「ソフトウェア」に変更した旨を警告するメッセージが表示されます。

ブレークポイントの設定方法は2通りあります。

_	<b>.</b>		
-	1	1	•
/J	ILLA.		

6			
7 ⊕#ifdefFPU[]			
1 fffe0017 _INITSCT();			
<pre>3 // _INIT_IOLIB();</pre>	// Use SIM I/O		
5 ⊕// ecrno=0;	// Remove the comme	ent	
8 9 0 0 Toggle Software Breakpoint Toggle Hardware Breakpoint			
1 ゴレークポイントの切り替え(K)	Ctrl+シフト+B		
ハードウェア ポイントを追加(A) ブレークポイント namic Printf	Ctrl+ダブル・クリック		
5 ブレークポイントを使用可能にする(E)	シフト+ダブル・クリック		
ハードウェアブレーク プロパティー(P)	Ctrl+ダブル・クリック		
ポイント設定エラー (R)	<b>&gt;</b>		C/C++ ブレークポイント
9 Switch Default e2 studio Breakpoint type to Software	vare		C/C++ Dynamic Printf
1 Toggle Relocated Breakpoint Enabled		۲	e2 studio Breakpoint
2 Delete Relocated Breakpoint			
אראיז איז איז איז איז איז איז איז איז איז			
ブレークポイント			

#### 図 5-11 [ブレークポイント] ビュー – ブレークポイント設定方法 1

ブレークポイント設定方法1では、

 ソースコードの左余白を右クリックして [Toggle Software Breakpoint] または [Toggle Hardware Breakpoint] を選択してハードウェアブレークポイント 20 またはソフトウェアブレークポイント 20 を 設定します。 方法 2:



#### 図 5-12 [ブレークポイント] ビュー – ブレークポイント設定方法 2

ブレークポイント設定方法2では、

- (1) 右クリックしてコンテクスト・メニューを開き、[ブレークポイント型] → [e2 studio Breakpoint](デフォルトのハードウェアブレークポイント)または [ブレークポイント型] → [C/C++ ブレークポイント](ソフトウェアブレークポイント)を選択します。
- (2) ソースコードの左余白をダブルクリックして、ハードウェアブレークポイントまたはソフトウェアブレーク ポイントを設定します。
- (3) [ビューの表示] → [ブレークポイント] またはアイコン <sup>1</sup> をクリックし(あるいはショートカットキー [ALT]+[Shift]+[Q], [B] を使い)、[ブレークポイント] ビューを開いて、設定したハードウェアブレークポイントまたはソフトウェアブレークポイントを表示します。

ブレークポイントを無効にする際は、選択したブレークポイントのみを無効にするか、あるいはすべてのブレークポイントを無効にするか選択できます。

- (1) [ブレークポイント] ビューで、ブレークポイントを有効または無効に切り替えることができます。無効になったソフトウェアブレークポイントは白い丸が表示されます。無効になったハードウェアブレークポイントは白い丸が表示されます。
- (2) すべてのブレークポイントを無効にするには、[ブレークポイント] ビューにあるアイコン 🌂 をクリック します。斜線を重ねた青い丸が [ブレークポイント] ビューとエディタ画面の両方に表示されます。

#### 5.3.2. 式ビュー

式ビューでは、デバッグ中のグローバル変数、静的変数、ローカル変数の値を監視できます。ブレークした時 に表示が更新され、前回表示した時と異なる値は背景が黄色になります。また「リアルタイム・リフレッシュ」を 有効にした変数("R"マーカが表示されたもの)はデバッガが実行されている間、設定された周期で表示が更 新されます。

C:/Renesas/e2_studio_40026a/DebugComp/rx-	<mark>e</mark> 2 ਿਓ ਜ਼ ਲ	ka ⇒	<u>≉ ⊨  <b>4</b> ×</u>	🍇 📑 r	*	□ X	
	式		タイプ	値			~
a r_cq_main.c ☆ a r_cq_resetprq.c	R gP	eriodic_Delay	volatile int	12		ו ר	
44	中 新	しいれを追加					-
46 ⊕ Global variables and functions.					+	- F	
48 /* Start user code for global d 49 extern volatile in gPeriodic De 50 extern volatile un signed char gP 51 /* End user code. Do not edit co	lay; ericaic_F	lag; erated here */	,				_

図 5-13 [式] ビュー

グローバル変数を見るには、

- (1) [ビューの表示] → [式] あるいはアイコン 🙀 をクリックし、[式] ビューを開きます。
- (2) "main.c"の49行目で、グローバル変数(例えば "gPeriodic\_Delay")を[式] ビューヘドラッグ& ドロップします。(または、グローバル変数を右クリックして "監視式を追加(A)…"メニューアイテムを 選択し、[式] ビューに追加します。)
- (3) [式] ビューで、"リアルタイム・リフレッシュ"メニューアイテムを右クリックして選択します。これで、プログラムの動作中、表示した値はリアルタイムに更新されます。"R"は、このグローバル変数がリアルタイムで更新されることを意味します。
- (4) "リアルタイム・リフレッシュ"を無効にするには、"リアルタイム・リフレッシュを無効にする"メニュー 項目を右クリックして選択します。

#### 5.3.3. レジスタービュー

レジスタービューは汎用レジスタについての情報を表示します。プログラムを停止すると、変化のあった値を 強調表示します。

出出 レジスター 🗯		🖾 📲 🕞 📑 🖆 🕸 🗸 🖓	
名前	値	記述/説明	-
4 👬 General Reg	gisten	General Purpose and FPU Register Group	
1010 0101 <b>rO</b>	0x400		
1919 <b>r1</b>	0x80020		
1111 r2	0x200		
1919 r3	0x0		
1010 r4	0x88002		
1919 <b>r5</b>	0xefff7fff		
1919 <b>r6</b>	0x0		
1010 r7	0x0		
1010 r8	0x0		
1010			
Alminio タブ10 進数:1024 タブ8 進:02000 タブバイナリー:10000 タブFloat:1.43492	00000 963e-042		

図 5-14 [レジスター] ビュー

汎用レジスタ "r0" を表示するには、

- (1) [ビューの表示] → [レジスター] あるいはアイコン 🔤 をクリックし、[レジスター] ビューを開きます。
- (2) "r0" をクリックすると、他の基数フォーマットで値を表示します。

プログラムを停止すると、変化のあった値は [レジスター] ビューの中で強調表示(黄色など)されます。

#### 5.3.4. メモリービュー

メモリービューでは、ユーザは "メモリーモニター" でメモリを表示し編集することができます。各モニターは "ベースアドレス" と呼ばれる格納位置によって特定される記憶場所を表します。各メモリーモニターの中のメ モリデータは異なる "メモリレンダリング" で表示することができます。メモリレンダリングはあらかじめ設定し たデータフォーマット(例えば、16 進数、符号付き整数、符号なし整数、ASCII、イメージなど)です。



図 5-15 [メモリー] ビュー (1/2)

変数 (例えば "Data1") のメモリを表示するには、

- (1) [ビューの表示] → [メモリー] あるいはアイコン 🕕 をクリックし、[メモリー] ビューを開きます。
- (2) アイコン 中 をクリックし、[モニター・メモリー] ダイアログボックスを開きます。変数 "Data1" のア ドレスを入力します。

	&Data1 <16 進	数の整数> 🔂	新規レンダリング	·		
	メモリー・モニ	ター: &Data1 : 0x	808			
	作成するレンダ	リングを選択してく	ださい:			
	<u>16 進数の整数</u>			^ L>	ッダリングの追加	
	16 進					
	ASCII		$\mathbf{i}$			
			$\rightarrow$	*		
			\			
	&Data1 <16 進数の書	餐数> &Data	a1:0x808 <16	5進> 🛛 🦂	新規レンダリン?	Ø)
	&Data1 <16 進数の書 アドレス	隆数> &Data 0 - 3	a1:0x808<16	5 進> 🛛 📢 8 - B	▶ 新規レンダリンク C - F	Ø)
	&Data1 <16 進数の書 アドレス	隆数> &Data 0 - 3 0 0000000	a1 : 0x808 <16 4 - 7 0000C92D	5進> 🛛 📢 8 - B 11040000	▶ 新規レンダリン C - F 11040000	Ø)
Data1:0x80 <mark>8 &lt;16                                   </mark>	&Data1 <16 進数の書 アドレス ecceccocooo0000080 載の整数> 🛛 🗣 新規	登数> &Data 0 - 3 0 00000000 レンダリング… 3	a1:0x808<16 4 - 7 0000C92D 55A945	5 進> 🛛 📢 8 - B 11040000 729FF6F2	<ul> <li>新規レンダリン</li> <li>C - F</li> <li>11040000</li> <li>058CBB90</li> </ul>	Ø)
Data1:0x80 <mark>3 &lt;16 道</mark> アドレス	&Data1 <16 進数の書 アドレス ecceccocoooooooooooooooooooooooooooooo	<ul> <li>         を数&gt;         <ul> <li>&amp;Data</li> <li>0 - 3</li> <li>0 0000000</li> <li>レンダリング… }</li> </ul> </li> </ul>	a1:0x808 <16 4 - 7 0000C92D 55A9450 068877	5 進> 3 - B 11040000 729FF6F2 00200010	<ul> <li>新規レンダリンク</li> <li>C - F</li> <li>11040000</li> <li>058CBB90</li> <li>3A78797E</li> </ul>	Ø)
Data1:0x80 <mark>8 &lt;16 減</mark> Pドレス 000000000000000000000000000000000000	&Data1 <16 進数の書 アドレス 000000000000000000000000000000000000	を数> & Data の - 3 の 00000000 レンダリング… 3 の 00000411	a1:0x808 <10 4 - 7 0000C92D 55A9450 068877 5DD5BB3F	5 進> 22 4 8 - B <u>11040000</u> 729FF6F2 00200010 2C80300D	<ul> <li>新規レンダリング</li> <li>C - F</li> <li>11040000</li> <li>058CBB90</li> <li>3A78797E</li> <li>AC4155CB</li> </ul>	Ø)
Data1:0x80 <mark>3 &lt;16 減</mark> アドレス 00000000000000000000000000 0000000000	&Data1 <16 進数の書 アドレス 000000000000000000000000000000000000	を数> & Data 0 - 3 0 0000000 レンダリング… 3 0 00000411 F2F69F72 3	a1:0x808 <10 4 - 7 0000C92D 55A94F 068B77 5DD5BB3F 64930BA4	5 進> 2 4 8 - B 11040000 729FF6F2 00200010 2C80300D 96731FBD	<ul> <li>新規レンダリン</li> <li>C - F</li> <li>11040000</li> <li>058CBB90</li> <li>3A78797E</li> <li>AC4155CB</li> <li>224B1E9C</li> </ul>	Ø)

図 5-16 [メモリー] ビュー (2/2)

変数 "Data1" 用の新しいレンダリングフォーマット(例えば 16 進)を追加するには、

(1) 🕆 新規レンダリング... タブをクリックし、"16 進"を選択してレンダリングを追加します。

これで、"&Data1<16 進数の整数>" タブの横に新しい "&Data1 <16 進>" タブが作成されます。

#### 5.3.5. 逆アセンブルビュー

逆アセンブルビューは、ロードしたプログラムのソースコードとアセンブラ命令を混在して表示します。現在実 行中の行は画面上で矢印のマーカで強調表示されます。逆アセンブルビューでは、アセンブラ命令へのブレ ークポイントの設定、ブレークポイントの有効化/無効化、逆アセンブル命令のステップ実行、プログラムの 特定の命令へのジャンプが可能です。

🖻 main.c 🔀		
109	<pre></pre>	
117	⊖void main(void)	
118	{	
119	/* Initialise the debug LCD */	
120 ffff89a4	<pre>Init_LCD();</pre>	
121		
122	/* Displays the <u>Renesas</u> splash screen */	
123 tttt89a8	Display_LCD(LCD_LINE1, "Renesas");	
124 11118964	<pre>Display_LCD(LCD_LINE2, NICKNAME);</pre>	
125		•
	4	P
🖉 逆アセンブル 🌮		
≝ 逆アセンブル 2 <sup>2</sup> ffff89a2: 0	ロケーションを入力 ・ ② 協議風 ご t オペコード 関数オフセット	
■ 逆アセンブル & ffff89a2: 0	ロケーションを入力 ・ ② 沿区 © 1 オペコード 関数オフセット 	
逆アセンブル 2 <sup>2</sup> ffff89a2: 0 ffff89a4: 0x	ロケーションを入力  ・  ②  る 「 ③  ③  ③  ③  ③  ③  ③  ③  ③  ③  ③  ③  ③	
逆アセンブル お ffff89a2: 0 ffff89a4: 0x 123	ロケーションを入力 ・ ② 企区 C r オペコード 関数オフセット 0000000596fdff main+0 bsr.a 0xffff873a <init_lcd> Display_LCD(LCD_LINE1, "Renesas");</init_lcd>	
逆アセンブル & ffff89a2: 0 ffff89a4: 0xi 123 ffff89a8: 0xi	ロケーションを入力 ・ ② 企区 © 1 オペコード 関数オフセット 0000000596fdff main+0 bsr.a 0xffff873a <init_lcd> Display_LCD(LCD_LINE1, "Renesas"); mov.l #0xffff8124, r2</init_lcd>	
逆アセンブル SP ffff89a2: 0 ffff89a4: 0x0 123 ffff89a8: 0x0 ffff89a8: 0x0 ffff89a8: 0x0 ffff89a8: 0x0	ロケーションを入力 ・ ② 企 ⑤ ⑥ ⑥ ⑥ ⑥ 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	
<ul> <li>逆アセンブル &amp; ffff89a2: 0</li> <li>ffff89a4: 0x</li> <li>ffff89a4: 0x</li> <li>ffff89a8: 0x</li> <li>ffff89a8: 0x</li> <li>ffff89ae: 0x</li> </ul>	オペコード	
<ul> <li>逆アセンブル デ ffff89a2: 0 ffff89a4: 0xt 123 ffff89a8: 0xt ffff89a8: 0xt ffff89a8: 0xt ffff89a8: 0xt ffff89a8: 0xt</li> </ul>	オペコード	
逆アセンブル ŷ ffff89a2: 0 ffff89a4: 0x 123 ffff89a8: 0x ffff89a8: 0x ffff89ae: 0x hにより、アセンブ c ソース(アクティ)	オペコード     関数オフセット     tsd #24     main:     bsr.a 0xffff873a <init_lcd>     Display_LCD(LCD_LINE1, "Renesas");     mov.l #0xffff87ba <display_lcd>     Display_LCD(LCD_LINE1, "Renesas");     mov.l #0, r1     bsr.a 0xffff87ba <display_lcd>     Display_LCD&gt;     Display_LCD(LCD_LINE2, NICKNAME);     mov.l #0xffff812c, r2</display_lcd></display_lcd></init_lcd>	
逆アセンブル 32 ffff89a2: 0 ffff89a2: 0 ffff89a4: 0x 123 ffff89a8: 0x ffff89ae: 0x れにより、アセンブ に C ソース(アクティン グコンテクスト)にリ	オペコード     関数オフセット     サtsd #24     main:     bsr.a 0xffff873a <init_lcd>     Display_LCD(LCD_LINE1, "Renesas");     mov.l #0xfff87ba <display_lcd>     Display_LCD(LCD_LINE2, NICKNAME);     mov.l #0xfff812c, r2     vクしま</display_lcd></init_lcd>	

図 5-17 [逆アセンブル] ビュー

混合モードでCコードとアセンブリコードの両方を表示するには、

- (1) [ビューの表示] → [逆アセンブル] あるいはアイコン 🏧 をクリックし、[逆アセンブル] ビューを開き ます。
- (2) アイコン な をクリックして、アセンブリソースとCソース(アクティブなデバッグコンテクスト)のリン クを有効にします。
- (3) [逆アセンブル] ビューのアドレス列で右クリックし、"オペコードを表示する" と "関数オフセットを表示する" を選択します。
- (4) コンテクスト・メニューを使用すればエディタ内で Source Addresses を有効にすることができます。



図 5-18 Source Addresses メニュー





#### 5.3.6. 変数ビュー

変数ビューは、現在実行中のスコープ内で表示可能な全てのローカル変数を表示します。

589 597 ffff8ecb 598 599 600 601 602 603 604 ⊕	<pre>* Outline void StartDebour {     /* Declare     initial     static bool     /* Check if     called).</pre>	: StartDebounceTin nceTimer(uint16_t) local static varia ised yet */ timer_initialised the CMT timer is */	<pre>mer[] compare_match) ble to track if th     = false; not initialised (f</pre>	e CMT timer has been first time function has be	▲ een
605 ffff8ecf ⊖	if(!timer_i	nitialised)			
606 607 608 ffff8ee2 609	{ /* Disal SYSTEM.	ble register prote PRCR.WORD = 0xA50B	ction */		
	*				- F
)= 変数 😒			<b>‱ ⇒t</b> : [=	) 🔷 🖉 🗙 🔆 📑 🖻	~
名前	タイプ	値			
(x)= compare_mat	ch uint16_t	2048			
(x)= timer_initialise	ed _Bool	true	J		
				4	•

図 5-20 [変数] ビュー

ローカル変数(例えば、関数 "StartDebounceTimer()" の "compare\_match" や "timer\_initialised")を 見るには、

- (1) [ビューの表示] → [変数] またはアイコン 🙌 をクリックし、[変数] ビューを開きます。
- (2) ステップ実行で関数 "StartDebounceTimer()"の中に入ると、[変数]ビュー内にローカル変数の 値が表示されます(上記の例では 2048 および true)。

#### 5.3.7. イベントポイントビュー

イベントは、プログラム実行中にブレークあるいはトレース機能を実行するために設定された条件の組み合わせです。ユーザはイベントポイントビューで、異なる種類の定義されたイベント、たとえば、トレース開始、トレース終了、トレース・レコード、イベント・ブレーク、実行前 PC ブレーク、タイマー開始、およびタイマー終了、などを設定、表示することができます。

設定できるイベント数や設定条件は MCU によって異なります。以下に挙げる2種類のイベントがあります。

- 実行アドレス:エミュレータは CPU が特定のアドレスの命令を実行しようとしたことを検出します。これが "実行前 PC" ブレーク(例えば、イベントにより、条件は指定アドレスで命令の実行直前に成立する)、あるいは他のイベント(例えば、イベントにより、条件は指定アドレスで命令の実行直後に成立する)となります。
- データ・アクセス:エミュレータは指定された条件での指定アドレスあるいは指定アドレス範囲へのア クセスを検出します。これにより、アドレスとデータを組み合わせた条件を設定することができます。

イベントの組み合わせは(OR, AND(およびその組み合わせ), およびシーケンシャル)は2つ以上のイベントに使用できます。

			· · · · ·				<u> </u>	
プ	アト	ドレス	データ	パス	タイ	チャネル	-	
🛾 ᢨ トレース開始								
🛛 🗃 トレース停止								
] 🖞 イベント・ブレーク 🛛 🥤	🖁 編集 イベント	・ブレーク						
● 美行前 PC ブレーク						28 2	sonter.	
💽 タイマー開始						н <u>п</u> 0.	Parente [	
👔 🔗 タイマー停止	タイプ	アドレス	データ	パス・フ	カウント	タイマー チャ	ネル コメント	<
マークト 保管されたテンプト								
9191 WEC1000 000								
			1					
	追加	<u>橋葉…</u>	PC: 0 OA: 0/1024 す^	べて: 0/1024				
					_			

図 5-21 [イベントポイント] ビュー (1/2)

アドレスまたはデータが一致する条件(例えば、gFlashCount = "0xB0" のとき)で、グローバル変数にイベン ト・ブレークを設定するには、

- (1) [ビューの表示] → [イベントポイント] あるいはアイコン

   をクリックし、[イベントポイント] ビューを 開きます。
- (2) "イベント・ブレーク" オプションをダブルクリックし、[編集イベント・ブレーク] ダイアログボックスを開きます。

(3) [追加...] ボタンをクリックし、以下の操作を行います。

G					
	e <sup>2</sup> イベントポイン	トの追加		x	
	イベントポイン	ト Data Access データ・アクセス設	定 一	3	
	開始アドレス:	&_\$gFla	ashCount	<b>→</b>	
	マドレフ設定	データ・アクセス設	定		
	データ・アクセン	ノンジェン ノンシュー ス設定:	~		
	リード/ライト:		Read/Writ	e 🔹	
	アクセス・サイン	ズ:	Not Specif	ied 🗸	
	Bus Master:		CPU 👻		
	☑ 比較条件:				
	比較値:		0×B0		
	マスク値:		0		
	比較条件:		Equals	<b>•</b>	
(x)= 変数 💊 ブI	ノークポ 60° C	-SPY Auto 👷 式	• 1×>	トポ 🛛 📄 IO Regist 🖳 🗆	
		×	× 🔆 🛙	🛛 📄 🖌 🦕 🛛 PC: 0/8 OA: 1/4 🛛 🞭	
タイプ	タイプ		λ χ	データ	
📃 🖉 📸 NL	🔽 釐 トレース開始				
	〕				
	ース・レコード				
	ース・レコード ント・ブレーク				

図 5-22 [イベントポイント] ビュー (2/2)

- (4) イベントポイントの種類に "Data Access" を選択します。
- (5) [アドレス設定] タブに進み、アイコン をクリックしてシンボル "\_\$gFlashCount" を検索します。 (このグローバル変数のアドレスは "&\_\$gFlashCount" です。)
- (6) 次に、[データ・アクセス設定] タブに切り替え、[比較条件] チェックボックスを有効にして比較値を "0xB0" に設定します。[OK] をクリックしてください。
- (7) [イベントポイント] ビューで、イベント・ブレークが "gFlashCount = 0xB0" に設定されて有効になっ ていることを確認してください。プログラムを最初から実行するためにリセットします。

εl,	タイプ	値	名前 :gFlashCount	
(X)= gFlashCount ♣ 新しい式を追加	uint16_t	176	タフDetails:176 タブデフォルト:176 タブ10 進数:176 タブ16 進:0xb0 タブバイナリー:10110000 タブ8 道:0260	ł
C HASHLED.C 23	resetprg.c	main.c		
72 73 74 fffc8002 ⊖ 75 76 fffc8025 ⊖ 77 78 79 80 81 fffc8020	<pre>resetprg.c [ /* Flash th while((gSwi</pre>	main.c me LEDs for 2 itchFlag == 0 LLed_Delay =	<pre>200 times or until a user switch is pressed */ )&amp;&amp;(gFlashCount &gt; 0)) 0; ulLed_Delay &lt; 4000000; ++ulLed_Delay) ; after a specific delay. */</pre>	

#### 図 5-23 イベント・ブレークの実行

図 5-23 は、gFlashCount が 176(0xB0)になったときにプログラムの 76 行目で実行が停止する様子を示します。

#### 5.3.8. IO レジスタビュー

IO レジスタは Special Function Register(SFR)のことです。IO レジスタビューは、ターゲット専用の IO ファ イルで定義された全レジスタセットの名前、アドレス、値(16 進数および 2 進数)を表示します。ユーザは、[選 択されたレジスター] に必要な IO レジスタを選択して追加することによって、IO レジスタビューをカスタマイ ズすることができます。

ろ前	値	アドレス	アクセス	*	
> POE					
PORT					
PORT0					
▷ ● PDR	0x28	0x0008c000	RW	=	
PODR	0x28	0x0008c020	RW		
PIDR	0x00	0x0008c040	RW		
⊳ PMR	0x00	0x0008c060	RW		
▷ ODR	0x00	0x0008c080	RW		
▷ ODR1	0x00	0x0008c081	RW		
▶ ● PCR	IO Registers 🛛		E E 🗲	\delta 🎓 🔍 🚍	) 💾 🔻 💾 [
	前	値	アドレス	アクセス	
	<ul> <li>PORT0</li> </ul>				
	PDR	0x28	0x0008c000	RW	
	PCR	0x00	0x0008c0c0	RW	

図 5-24 [IO Registers] ビュー

IO レジスタを選択して (例えばポート 0 の PDR や PCR) を表示するには、

- (1) [ウィンドウ] → [ビューの表示] → [その他...] の順でクリックします。[ビューの表示]ダイアログボック スで、[デバッグ] の下にある [IO Registers] をクリックするか、あるいはアイコン 
   定 [IO Registers] ビューを開きます。
- (2) [すべてのレジスター] タブ上で、[IO Registers] ビューで [PORT0] を探し、PORT0 の IO レジスタ リストを拡張します。
- (3) "PDR" および "PCR" を [選択されたレジスター] タブヘドラッグ&ドロップします。IO レジスタ横の 緑の丸いマーク●は、選択されたレジスタであることを意味しています。
- (4) [選択されたレジスター] タブに切り替え、"PORT0" IO レジスタの "PDR" および "PCR" を表示し ます。

拡張した IO レジスタリストを [すべてのレジスター] にロードするには時間がかかる場合があるため、必要な IO レジスタを選択して [選択されたレジスター] に表示することを推奨します。

#### 5.3.9. トレースビュー

トレースとは、ユーザプログラム実行中、1 サイクルごとのバス情報をトレースメモリから取得することを意味 します。取得されたトレース情報はトレースビューに表示されます。それによりユーザはプログラムの実行を 追跡し、問題が発生した箇所を探すことができます。

トレースバッファは有限(1~32M バイトのサイズ)なため、バッファが一杯になると、最も古いトレースデータを 新しいデータで上書きします。

🗞 Trace 🖇	×	<b>≈ 3</b> ‡	⇒Q, ¢°   (	∄ 🗳 🗣 🖨 🖗 🖡 🗍	₩∀ @ % 🕑 ~
No records					
Cycle	Label	Address	Source A	● トレース設定	
				トレース・モード: トレース出力: トレース・タイプ: トレース容量: Timestamp Frequency Divider: タイム・スタンプ表示を有効化: OK	<ul> <li>トレースを停止する</li> <li>トレースを停止する</li> <li>トレースメモリを上書きし実行を続ける</li> <li>分岐</li> <li>✓</li> <li>✓</li> <li>✓</li> <li>✓</li> <li>≠ャンセル</li> </ul>

図 5-25 [Trace] ビュー (1/2)

2つの関数の間 (例えば、"main()" と "sort()" 間のトレース) の point-to-point トレースを設定するには、

- (1) [ウィンドウ] → [ビューの表示] → [その他...] の順にクリックします。[ビューの表示] ダイアログボッ クスで、[デバッグ] の下にある [Trace] あるいはアイコン <sup>8</sup> をクリックし、[Trace] ビューを開き ます。
- (2) **ひ**アイコンを選択して[Trace] ビューをオンにします。
- (3) アイコン 🗾 (取得) をクリックし、以下を設定します。
  - トレース・モード: "トレースを停止する"
  - トレース·タイプ: "分岐"
- (4) [OK] をクリックします。

🗞 Trace 🛛			<b>\$</b>	⇒ 🔍 🐨		; 🛦 🖞	\$   <b>₩   ∀</b>	۵ 🍫 🗹	) ~
No record	S								
Cycle	Label	Address	Source	Destin	Data	Size	R/W	BUS M	Туре
	- (	-14 - 1							X
	ス・イベント	~ホイント							
● 開始	📔 停1	上 🥤 記錄	录						
					組	み合わせ条件	‡: OR		161
タイプ		アドレス		データ	パス・カ	ウントタ	イマー チャ	ネル コメン	
	DR	&main		J					
	ス・イベント	ポイント							×
● 開始	層   停止	L 🧉 記録	a x						
					組	み合わせ条(	#: OR	▼ <	5 순
タイプ		アド	レス		データ	パス	・カウント	タイマー チ	ヤネル:
☑ OR		&Display_LC	O (0xffff87ba	a)					0
<u> </u>									

図 5-26 [Trace] ビュー (2/2)

- (1) <sup>Sol</sup>(トレース・イベントポイントを編集)をクリックし、[トレース・イベントポイント] ダイアログボックスを 開きます。
- (2) [開始] タブの下の "main()" 関数に、最初のイベントポイント(実行アドレス "&main"、あるいは 0xFFFC8228)を追加します。
- (3) 次に [停止] タブに切り替え、"Display\_LCD()" 関数に、2 番目のイベントポイント(実行アドレス "&Display\_LCD", あるいは 0xFFFC816F)を追加します。
- (4) リセット後にプログラムを実行します。



図 5-27 2 つの関数間の point-to-point トレース

上に示した図は、関数 "main()" から "Display\_LCD ()" 間のトレース結果を示しています。トレース結果は トレースパラメータ(例えば、分岐タイプ、アドレス範囲)でフィルタし、.xml フォーマットで(バス、アセンブリ、ソ ース情報を含む)保存することができます。

# 第6章 ヘルプ

ヘルプシステムによって、ユーザはワークベンチ内の各ヘルプウィンドウやヘルプ画面から、ヘルプドキュメン トのブラウズ、検索、ブックマーク、印刷が可能です。また、ヘルプメニューから e<sup>2</sup> studio 専用のオンラインフ ォーラムにアクセスできます。

[ヘルプ] をクリックしてヘルプメニューをプルダウンしてください。

$\sim$	レプ(H)	
3	ようこそ(W) (1)	
(?)  _   %	ヘルプ目次(H) IAR Information Center 検索(E) ダイナミック・ヘルプ(D) 3	
æ	キー・アシスト(K) Ctrl+シフト+L ヒント(T) バグまたは拡張を報告 虎の巻の表示(C)	
<b>1</b> 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	ドキュメント検索 ツールニュース RenesasRulz コミュニティー・フォーラム Renesas ツールチェーンの追加 更新の検査 新規ソフトウェアのインストール インストール詳細	
*	Create IAR Embedded Workbench error report	
(8) (8)	e2 studio について(A)	

図 6-1 ヘルプメニュー

ヘルプメニューの使い方

- ① [ようこそ] をクリックすると、e<sup>2</sup> studio の概要、e<sup>2</sup> studio チュートリアルとサンプルプログラムへのリンク、 リリースノートを表示します。
- ② [ヘルプ目次] をクリックすると、新たにヘルプウィンドウが開きヘルプを検索できます。
- ③ [ダイナミック・ヘルプ] をクリックすると、ワークベンチにヘルプ画面を開きます。
- ④ [RenesasRulz コミュニティー・フォーラム] をクリックすると、e<sup>2</sup> studio 関連のディスカッション参加型オン ラインフォーラムにアクセスします。インターネット接続が必要です。

[ヘルプ] ウィンドウには、有用な情報が豊富にあります。例えば "e2 studio デバッグ・ヘルプ" では、デバッグ構成、設定できるブレークポイント数、エミュレータの使用法などの有用な情報を提供します。"e2 studio デバッグ・ヘルプ" にアクセスするには、[ヘルプ] メニュー  $\rightarrow$  [ヘルプ目次]  $\rightarrow$  "e2 studio デバッグ・ヘルプ" の順にクリックしてください。



## 改訂記録

			改訂内容					
Rev.	発行日	ページ	概要					
1.00	2013.11.16	-	初版発行					
2.00	2014.04.15	-	e <sup>2</sup> studio v3.0.0.022をサポート					
2.10	2014.12.08	-	e <sup>2</sup> studio v3.1.2.009をサポート					
3.00	2015.06.16	-	e <sup>2</sup> studio v4.0.0.023をサポート					
4.00	2016.05.27	-	e <sup>2</sup> studio v5.0.0.043をサポート					

e <sup>2</sup> studio ユーザー: 入門ガイド	ズマニュアル	
発行年月日	2013年11月16日 2014年 6月 2日 2015年 6月16日 2016年 5月27日	Rev.1.00 Rev.2.00 Rev.3.00 Rev.4.00
発行	ルネサス エレクトロ	コニクス株式会社



ルネサスエレクトロニクス株式会社

http://www.renesas.com

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

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

■営業お問合せ窓口

© 2015 Renesas Electronics Corporation. All rights reserved. Colophon 3.0

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

# e<sup>2</sup> studio



R20UT2771EJ0300