User's Manual

16

RENESAS

Embedded Target

ユーザーズマニュアル　操作編

対象デバイス
RX ファミリ
RL78 ファミリ
RA ファミリ

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　2024.08

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。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。
2. 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

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口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。

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。
7. リザーブアドレス（予約領域）のアクセス禁止
   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
8. 製品間の相違について
   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

このマニュアルの使い方

1. 目的

このマニュアルは、これらのデバイスを使用したシステムのハードウェアまたはソフトウェアを開発する際の参考として、モデルベース開発ツールの機能を理解することを目的としています。

2. 対象者

このマニュアルは、MATLAB®/Simulink®の機能を理解したい方、ソフトウェアおよびハードウェア・アプリケーション・システムを設計したい方を対象としています。

3. 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。

4. このマニュアルの読み方

このマニュアルを読むにあたっては，電気，論理回路，マイクロコンピュータに関する一般知識が必要となります。

5. 数や記号の表記

注:	本文中についた注の説明
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 ... XXXXH or 0xXXXX

6. 略語および略称の説明

略語／略称	英語名	日本語名
GUI	Graphical User Interface	グラフィカル・ユーザー・インターフェース
GDB	Standard GNU Debugger	GNU デバッガ—
IDE	Integrated Development Environment	統合開発環境
I/O	Input/Output	入力／出力
LM	Load Module	ロード・モジュール
MCU	Microcontroller Unit	マイクロ・コントローラー・ユニット
PC	Personal Computer	パーソナル・コンピュータ
PIL	Processor in the Loop	プロセッサー・イン・ザ・ループ

Licensing

This product uses the IronPython based on the following license.

目次

1. 概説

本章では、Processor in the Loop Simulation System である Embedded Target の機能概要について説明しています。

1.1 概要

Embedded Target は、PILS（Processor in the Loop Simulation system）において、検証環境の作成処理を自動化することにより、対象とする Simulink®モデルのアルゴリズム検証を容易にしています。

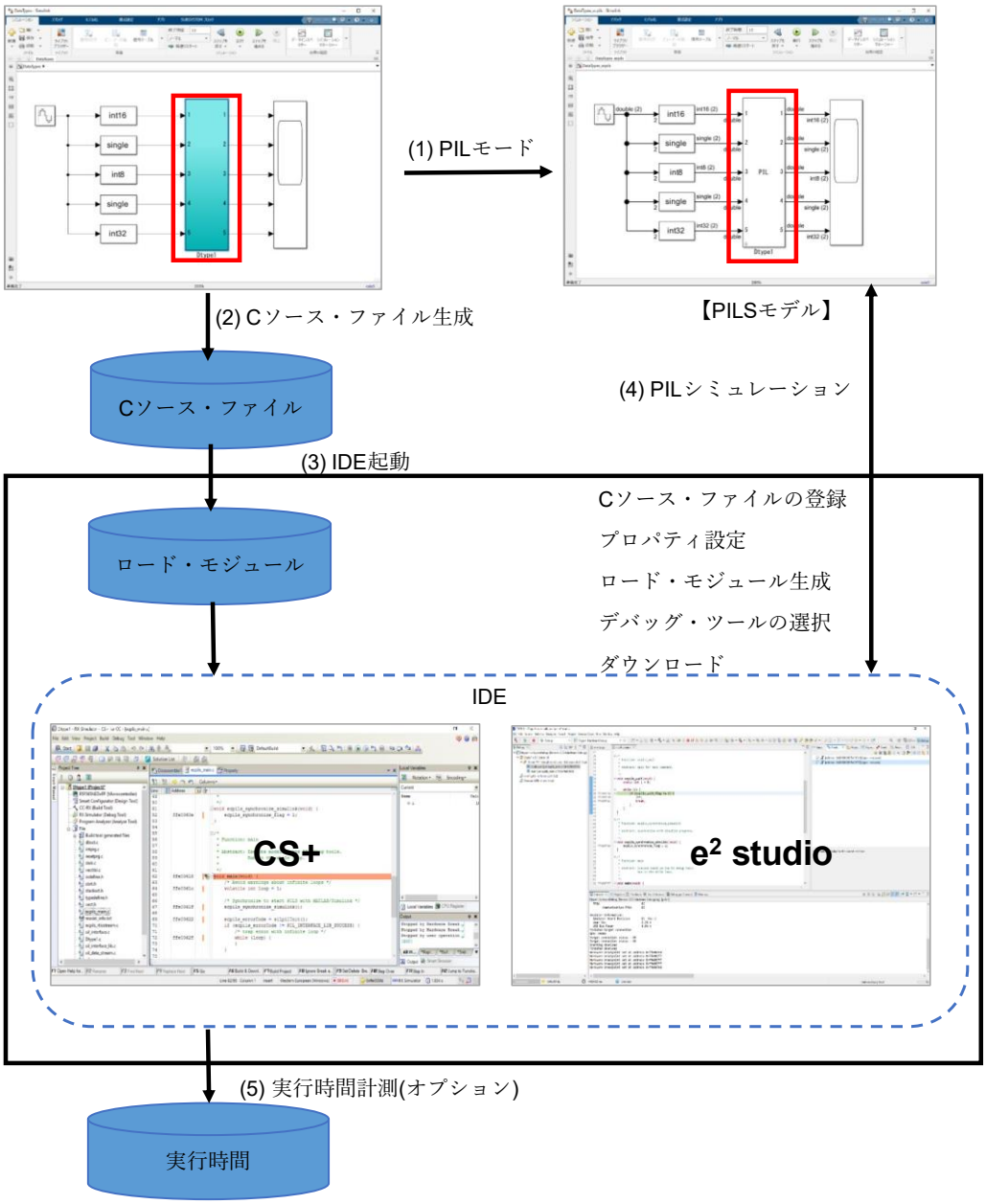

図 1-1 検証環境の作成処理の流れ

備考 Embedded Target では、上図における（1）～（5）を自動実行します。

1.2 特長

Embedded Target の特長を以下に示します。

1. 検証環境の作成

   PIL シミュレーションを行う際に必要となる下記の検証環境の作成処理を自動で実行することができます。コード生成対象となるブロックが、サブシステムの場合と、参照先モデル、最上位モデルの場合で、それぞれ検証環境の作成処理の動作が異なります。

   A. サブシステムブロック
      a. PIL連携用ブロックの生成と置き換え
      b. Cソース・ファイルの生成
      c. CS+/e² studioの起動
         - Cソース・ファイルの登録
         - プロパティ設定
         - ロード・モジュールの生成
         - デバッグツールの接続
         - ロード・モジュールのダウンロード
      d. PILシミュレーションは検証環境の作成後、手動で実行する必要があります。

   B. 参照先モデル／最上位モデル
      a. PILモードの指定（モデル作成時に、あらかじめ設定しておきます）
      b. Cソース・ファイルの生成
      c. CS+/e² studioの起動
         - Cソース・ファイルの登録
         - プロパティ設定
         - ロード・モジュールの生成
         - デバッグツールの接続
         - ロード・モジュールのダウンロード
      d. PILシミュレーション（検証環境の作成後、自動的に実行されます）

2. アルゴリズムの対象デバイスでの検証

   MATLAB®/Simulink®と CS+/e² studio の連携により逐次実行された PIL シミュレーションを行うことにより、Simulink®モデルから生成されたロード・モジュールのアルゴリズムを検証できます。ロード・モジュールは、対象となるデバイスのハードウェアの処理能力により、ひとつのコア（これ以降、シングルコア PIL シミュレーションとします）で実行可能です。

3. Simulink®モデルのアルゴリズム性能の測定

   Embedded Target は CS+/e² studio 上で Simulink®モデルによって生成されるロード・モジュールを実行することにより、PIL シミュレーションの実行時間測定（これ以降、パフォーマンス測定とします）を行うことができます。ファイルフォーマットに自動的に保存される測定結果は、PIL シミュレーションモードと測定方法によって異なる実行時間情報を提供します。

4. 複数のコード生成対象(ブロック、モデル)

   コード生成対象となるブロック、モデルは以下です。
   - サブシステムブロック
   - 参照先モデル
   - 最上位モデル

1.3 動作環境

Embedded Target の動作環境を以下に示します。

1. ハードウェア環境
   - オペレーティング・システム: Microsoft® Windows® 10 (64-bit) (推奨)
     Microsoft Windows® 11 (64-bit)
   - プロセッサー: 1GHz以上（ハイパースレッディング、マルチコアCPUに対応）
   - メイン・メモリー: 推奨：4Gバイト以上

2. MATLAB®、Simulink®製品（The MathWorks, Inc製）

MATLAB®	R2018b、R2021a	～	R2024a (R2024aを推奨)
Simulink®	同上		
Stateflow®(必要に応じて)	同上		
MATLAB® Coder™	同上		
Simulink® Coder®	同上		
Embedded Coder®	同上		

3. MEXファイル用コンパイラ

   MEXファイルは、MATLAB®からのCライブラリを起動するインターフェイスです。MEXファイルコンパイラは、MEXファイルのコンパイルに使用します。

   Embedded TargetはWindows® 10のMEXファイル用コンパイラとして以下のコンパラで動作確認を行いました。

   - Microsoft Visual C++ 2015 compiler (MATLAB® R2018b、R2021a)
   - Microsoft Visual C++ 2019 compiler (MATLAB® R2021b ～ R2024a)
   - Microsoft Visual C++ 2022 compiler (MATLAB® R2022a ～ R2024a)
   - MinGW 6.3 C/C++ (distributed by mingw-w64) (MATLAB® R2018b ～ R2024a)
   - MinGW 8.1 C/C++ (distributed by mingw-w64) (MATLAB® R2023a ～ R2024a)

   参考: System Requirements & Platform Availability
   https://www.mathworks.com/support/requirements/matlab-system-requirements.html

4. 統合開発環境（ルネサスエレクトロニクス製）

CS+	V8.12.00
e² studio	2024-04、2024-07

5. ビルド環境（ロード・モジュールを生成する環境）

CC-RX	CS+ V8.12.00以上、またはe² studio 2024-04以上に梱包（ルネサスエレクトロニクス製）
CC-RL	CS+ V8.12.00以上、またはe² studio 2024-04以上に梱包（ルネサスエレクトロニクス製）
GNU ARM Embedded	(バージョン: 13.2.1.arm-13-7以上)

備考1 MATLAB®/Simulink®製品については、使用するMATLAB®、およびSimulink®のバージョンに対応したオプション製品で該当環境を構築します。

備考2 MATLAB®をインストールする場合は、UAC(User Account Control)の対象となるフォルダ以外にインストール先を変更してインストールことを推奨します。ご使用のMATLAB®バージョンによっては、インストール先が"<システムドライブ>:¥Program Files"または"<システムドライブ>¥Program Files(x86)"などUAC対象のフォルダの場合、MEXビルドできない問題や、MATLAB®パスをセーブできない問題等が発生します。

備考3 統合開発環境は、CS+またはe² studioに限られます。

6. デバッグツール

エミュレータ (ルネサスエレクトロニクス製) E1、E2、E2 Lite、E20、EZ Emulator (RL78ファミリのみ)、COM Port (RL78/G23, RL78/G24シリーズのみ)

エミュレータ(他社製) J-Link (RAファミリまたは、いくつかのRXファミリのシリーズでe2 studio使用時のみ)

シミュレータ (ルネサスエレクトロニクス製、RAファミリを除く)

備考 シミュレータは、CS+／e2 studioに同梱されています。

7. MATLAB®、MATLAB® Coder™、Simulink® Coder™、Embedded Coder ®はMathWorks, Incの商標または登録商標です。
(https://www.mathworks.com/company/aboutus/policies_statements/trademarks.html)
Microsoft®、Windows®、Visual C++®はMicrosoft Corporationの米国およびその他の国における商標または登録商標です。
(https://www.microsoft.com/en-us/legal/intellectualproperty/trademarks/en-us.aspx)

1.4 機能仕様

Embedded Target では MATLAB®/Simulink®モデルのアルゴリズムを検証するために様々な機能を提供しています。いくつかの機能はルネサスエレクトロニクスが提供するライセンスを要求します。ここではライセンスの種類と使用できる機能を説明します。

1.4.1 ライセンスの種類と機能

以下はルネサスエレクトロニクスが登録した特定のライセンスを必要とする機能のリストです。

- RXデバイスでのシングルコアPILシミュレーション (Embedded Target for RX)
- RL78デバイスでのシングルコアPILシミュレーション (Embedded Target for RL78)
- RAデバイスでのシングルコアPILシミュレーション (Embedded Target for RA)

ルネサスエレクトロニクスは Embedded Target に対し、柔軟性のあるライセンス方針を提供しているので、必要に応じて上記の機能から選ぶことができます。
以下にライセンスの種類ごとの使用可能な操作を示します。

- RXデバイスでのシングルコアPILシミュレーション
- RL78デバイスでのシングルコアPILシミュレーション
- RAデバイスでのシングルコアPILシミュレーション
- ロード・モジュール生成
  RXデバイス・ファミリおよびRL78デバイス・ファミリ: ルネサスコンパイラによる生成
  RAデバイス・ファミリ: GNU ARM Embedded Compilerによる生成

1.4.2 機能説明

1.4.2.1 PILシミュレーションの対象デバイス

Embedded Target は、RX、RL78、RA を含むルネサスの様々な MCU ファミリによる Simulink®モデルのアルゴリズム検証をサポートします。
RX、RL78 および RA MCU ファミリでの PIL シミュレーションには、ルネサスが登録したライセンスが必要です。
ライセンスタイプは Embedded Target が提供する PIL シミュレーションモードです。PIL シミュレーションモードの詳細については、1.4.2.3 対象 MCU を参照してください。

1.4.2.2 対象デバイスのビルド・ツール

Simulink®モデルによって生成されたロード・モジュールは、サポートする RX デバイス・ファミリおよび RL78 デバイス・ファミリ用のルネサスコンパイラ、またはサポートする RA デバイス・ファミリ用の GNU ARM Embedded によって生成されます。この機能にはライセンスは不要です。

1.4.2.3 対象MCU

PIL シミュレーション用の対象デバイスのハードウェア仕様により、Embedded Target はシングルコアの対象 MCU を提供します。対象 MCU は、PIL シミュレーション中にロード・モジュールが実行できるコアの数を示します。

- シングルコアPILシミュレーション：ロード・モジュールは、対象デバイスの1つのコア上で実行できます。この機能は、Embedded Targetによってサポートされるすべてのファミリに対し使用可能です。

両方の対象 MCU でもシングルコア MCU の有効なライセンスが必要です。

- RXデバイスなら”Embedded Target for RX”ライセンス
- RL78デバイスなら”Embedded Target for RL78”ライセンス
- RAデバイスなら”Embedded Target for RA”ライセンス *

備考 Embedded Target は、Simulink®モデルでアルゴリズム性能をターゲットデバイスで測定する方法を提供します。それらの方法は対象 MCU に準じて含まれます。詳細については、3.5 コード生成対象となるアルゴリズムの対象デバイスでの検証を参照してください。

* RA ファミリのデバイスは e² studio 2023-10 でのみ使用可能です。

1.4.3 ライセンス管理の方法

Embedded Target のライセンスを管理するには、CS+に含まれる Renesas License Manager を使用します。ライセンスは、製品購入時に提供されたライセンスキーを Renesas License Manager に登録することにより有効になります。CS+をインストールしていない場合は、Renesas License Manager 単体をインストールしてください。

1.5 基本的な使い方

Embedded Target の使い方にはコード生成対象がサブシステムブロックか、参照先モデルか、最上位モデルかで違いがあります。それぞれの場合の使い方を以下に示します。ご使用の Simulink®モデルに合わせてご確認ください。

1.5.1 サブシステムブロック使用の場合

コード生成対象がサブシステムブロック使用の場合は、以下の手順で Embedded Target を利用します。

1. Simulink®モデルを作成します。コード生成対象のブロックは、サブシステム化して、1つのサブシステムブロックにまとめておきます。

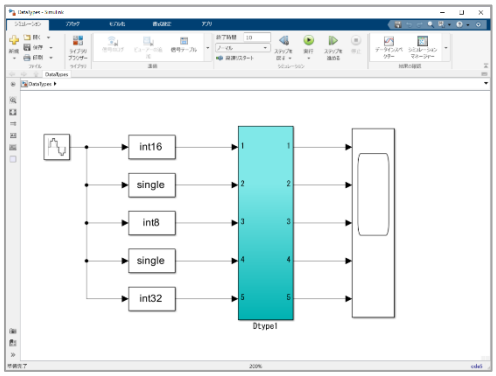

図 1-2 サブシステムモデル

2. コンフィギュレーション パラメーター ダイアログにおいて、コード生成や検証環境の作成に必要なパラメーターを設定します。

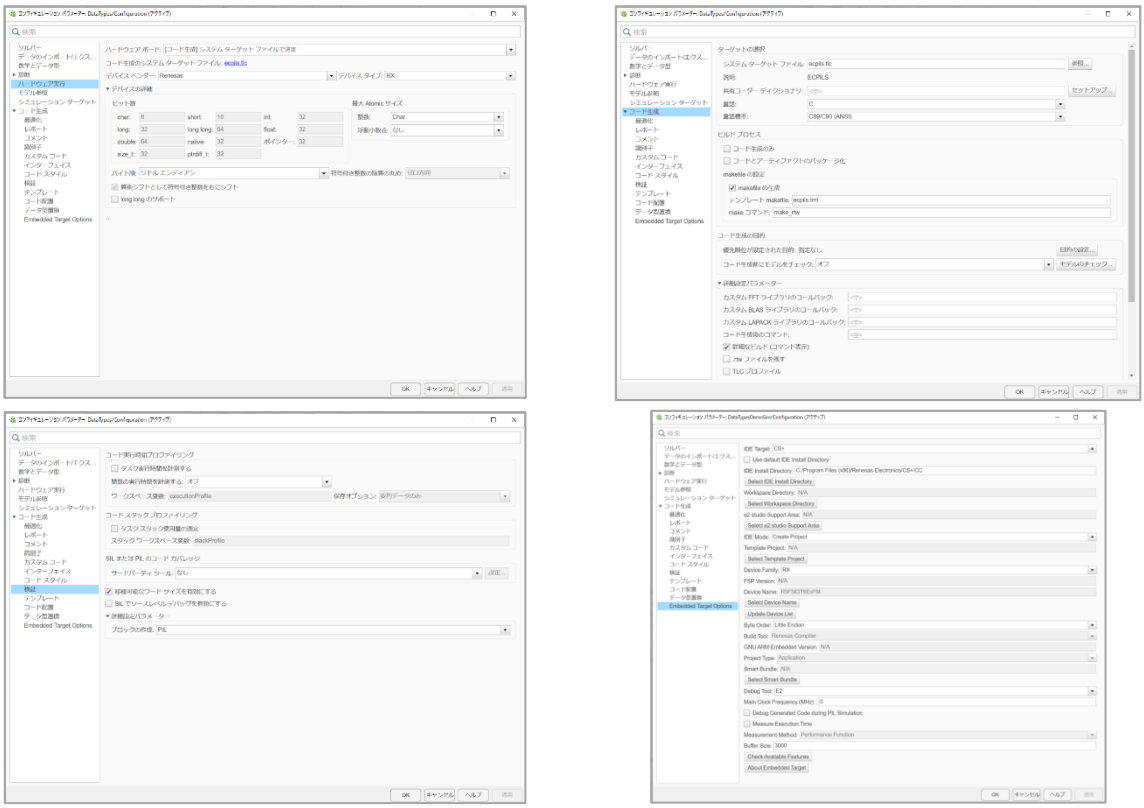

図 1-3 コンフィギュレーション・パラメーター設定

3. Simulink®モデルのサブシステムブロックを選択後、MATLAB®コマンド・ウィンドウにおいて、ecpils_buildコマンドを実行して、コード生成、PIL検証環境の作成を行います。Embedded TargetはCソースコードを生成し、CS+/e2 studioを起動します。また、Simulink®モデルは、生成されたPIL連携用ブロックとサブシステムブロックが置き換えられます。

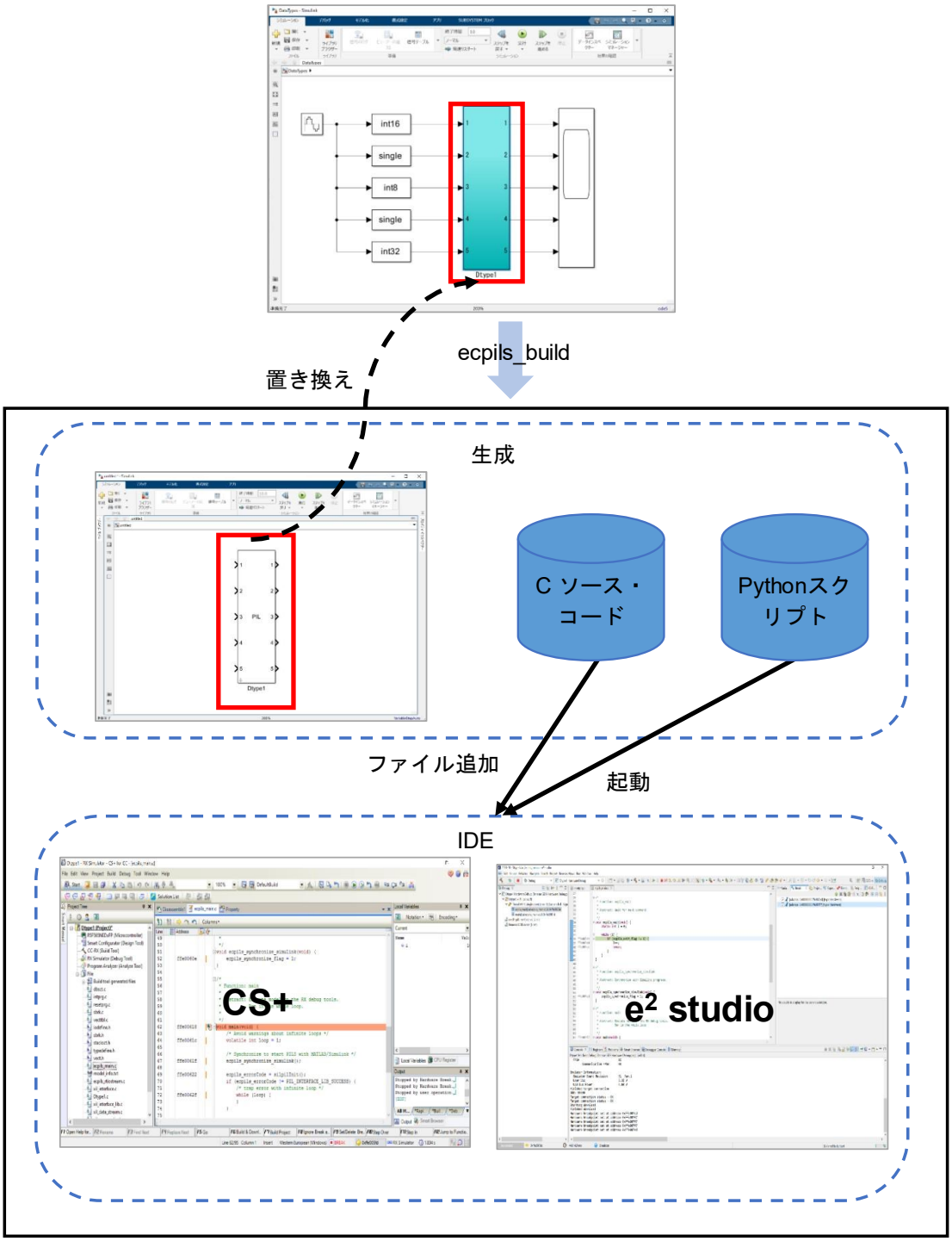

図 1-4 Embedded Targetの処理の流れ(サブシステムブロックの場合)

4. Simulink®モデルでシミュレーションを開始し、PILシミュレーションを実行します。

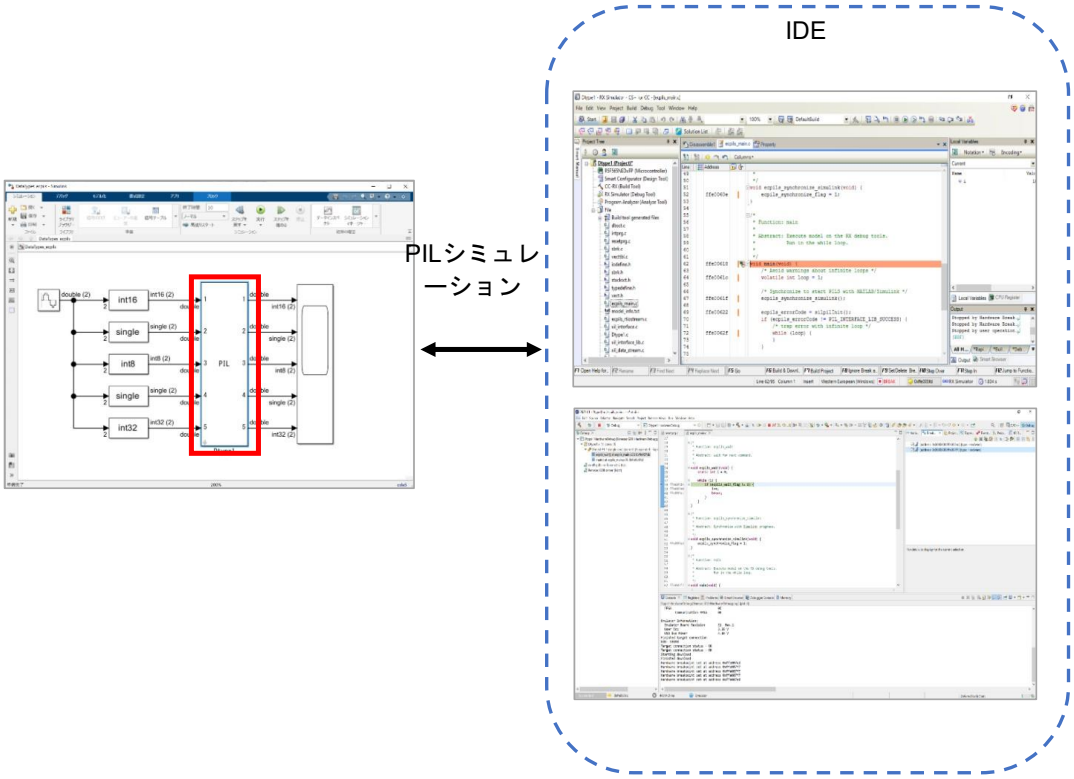

図 1-5 PILシミュレーションの流れ(サブシステムの場合)

1.5.2 参照先モデル使用の場合

コード生成対象が参照先モデルの場合は、以下の手順で Embedded Target を利用します。

1. Simulink®モデルを作成します。コード生成対象のブロックは、Modelブロックになります。

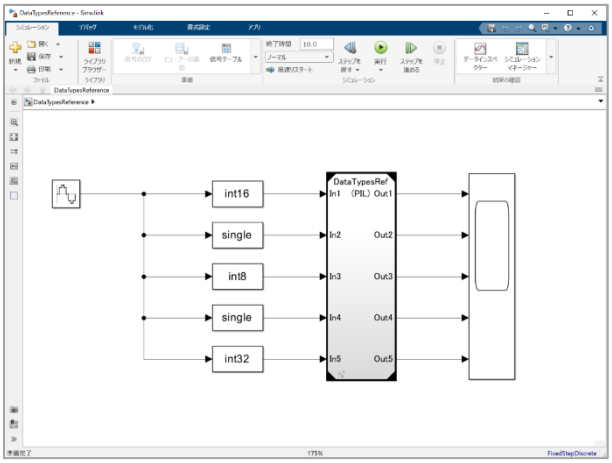

図 1-6 参照先モデル

2. コンフィギュレーション パラメーター ダイアログにおいて、コード生成や検証環境の作成に必要なパラメーターを設定します。

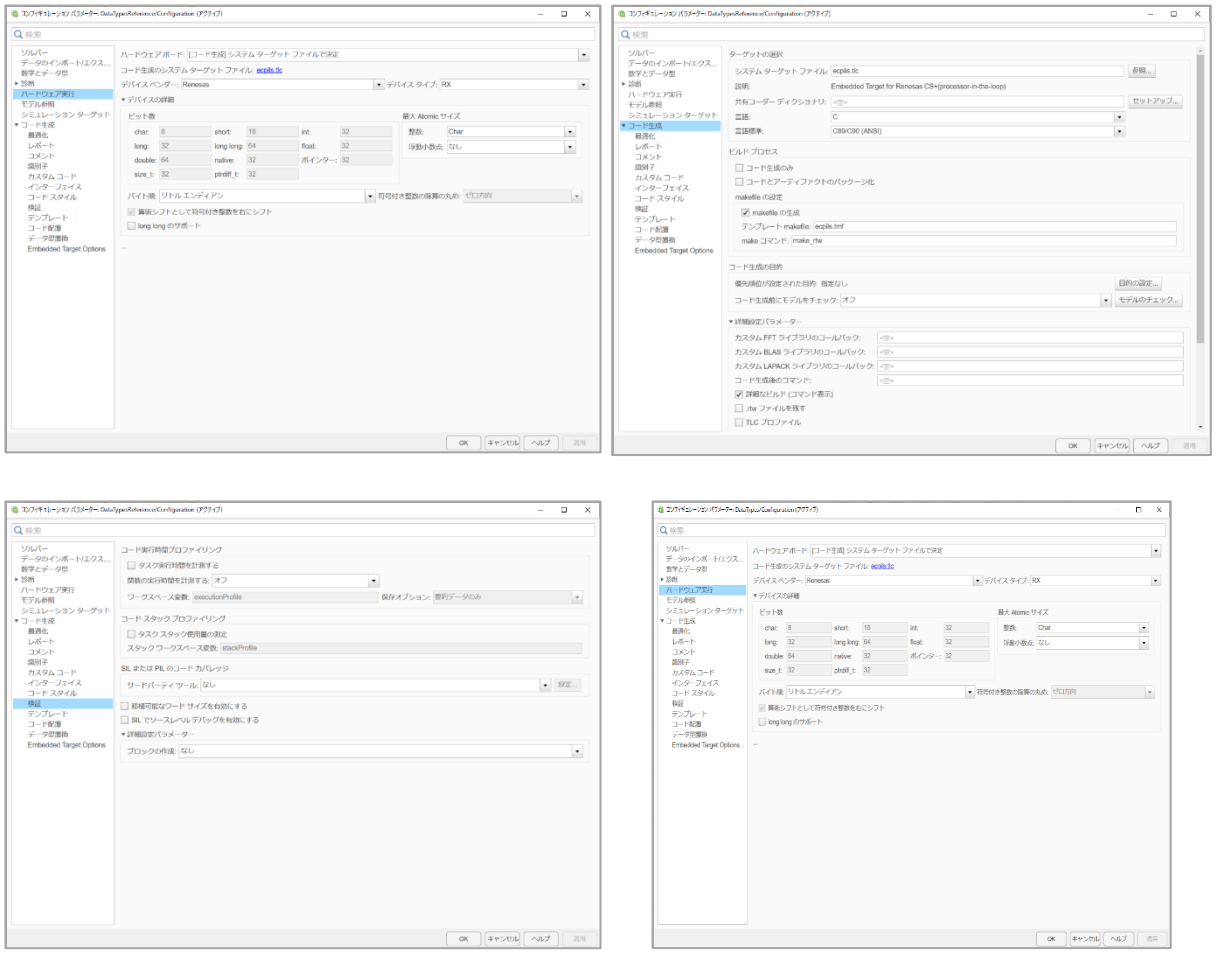

図 1-7 コンフィギュレーション パラメーター設定

3. Modelブロックに対してPILモードを設定します。

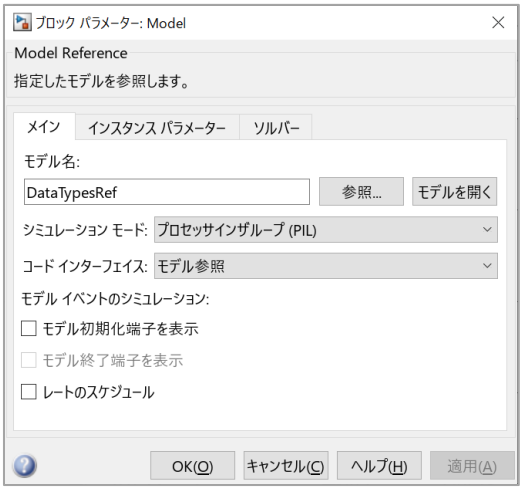

図 1-8 ブロック パラメーター設定

4. Simulink®モデルでシミュレーションを開始すると、コード生成から検証環境の作成、PILシミュレーションまでを自動で実行します。

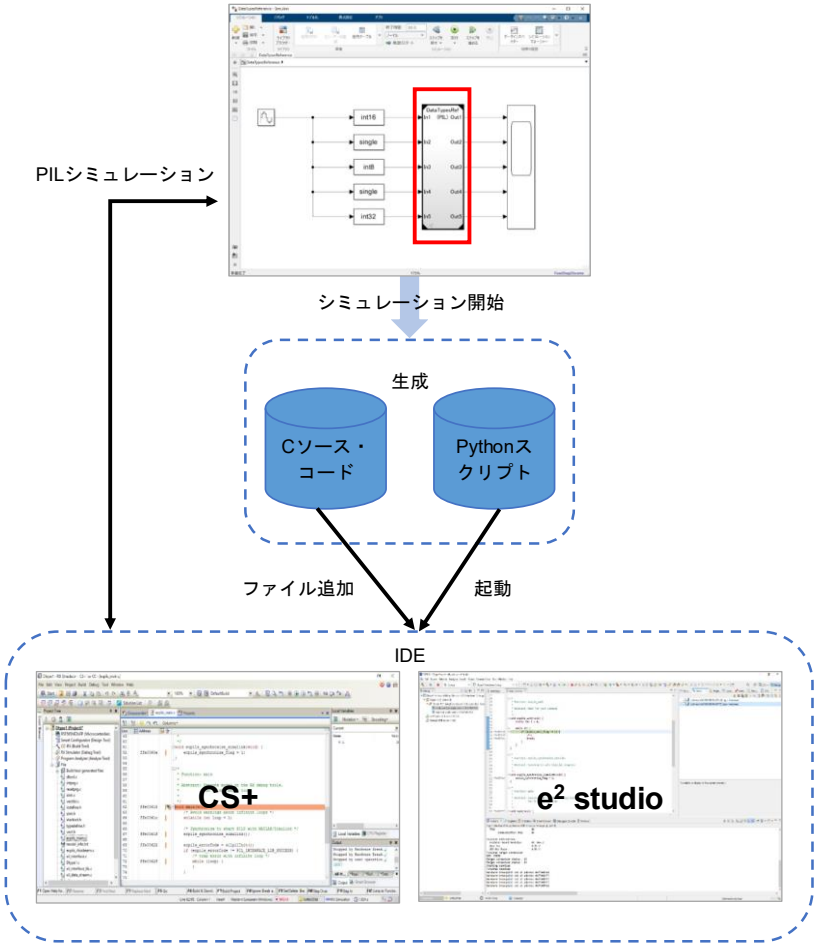

図 1-9 Embedded Targetの処理の流れ(参照先モデルの場合)

1.5.3 最上位モデル使用の場合

コード生成対象が最上位モデルの場合は、以下の手順で Embedded Target を利用します。

1. Simulink®モデルを作成します。コード生成対象は、最上位のモデルになります。

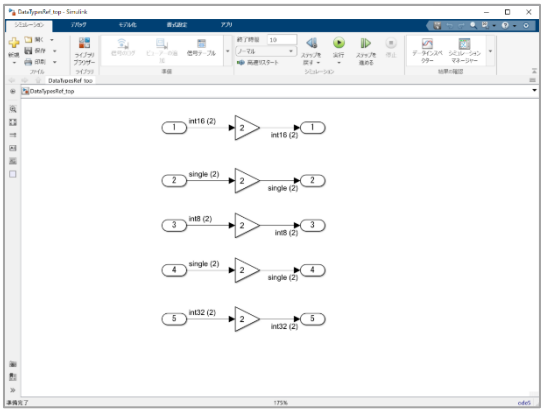

図 1-10 最上位モデル

2. コンフィギュレーション パラメーター ダイアログにおいて、コード生成や検証環境の作成に必要なパラメーターを設定します。

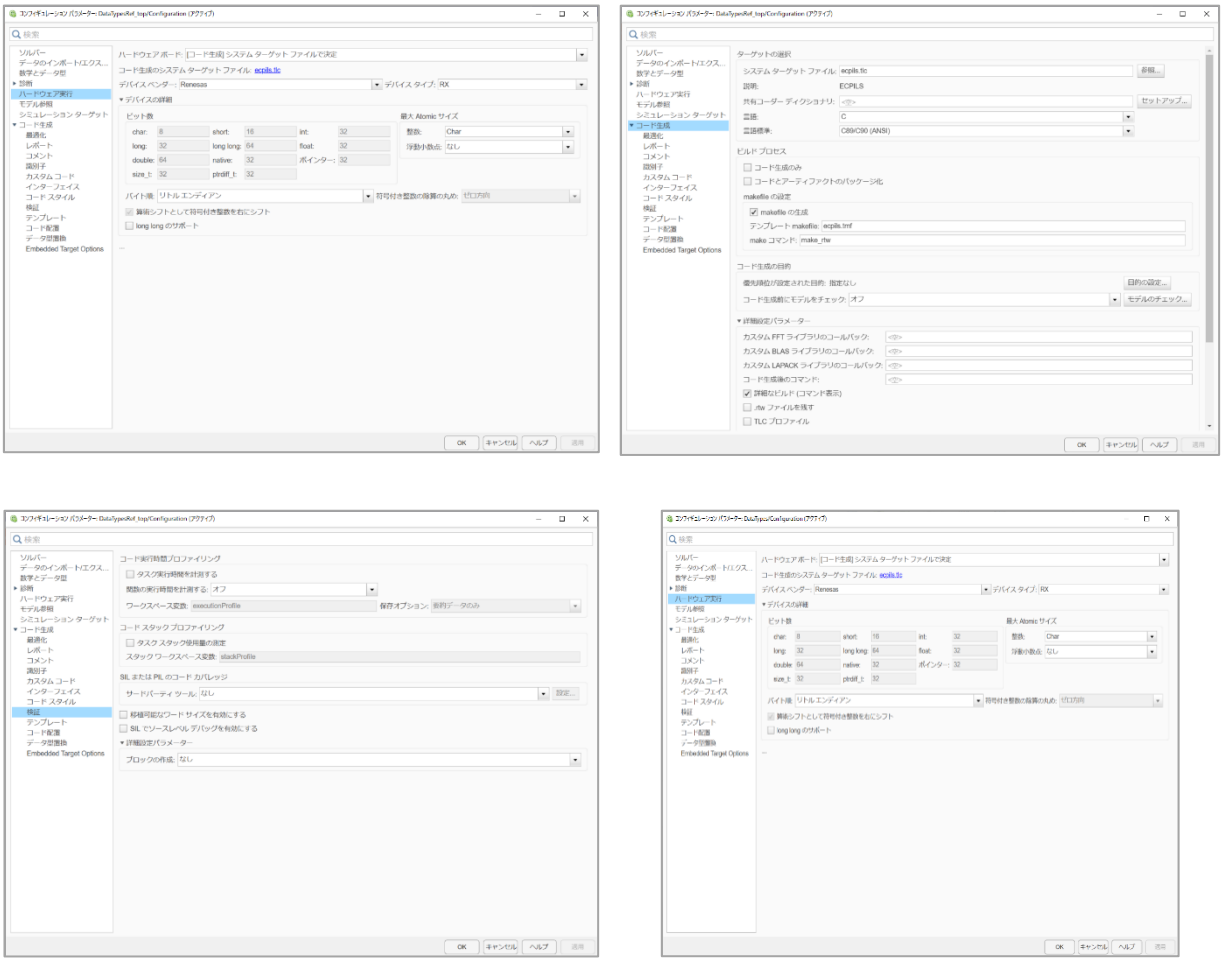

図 1-11 コンフィギュレーション・パラメーター設定

3. Simulink®モデルに対してPILモードを設定します。

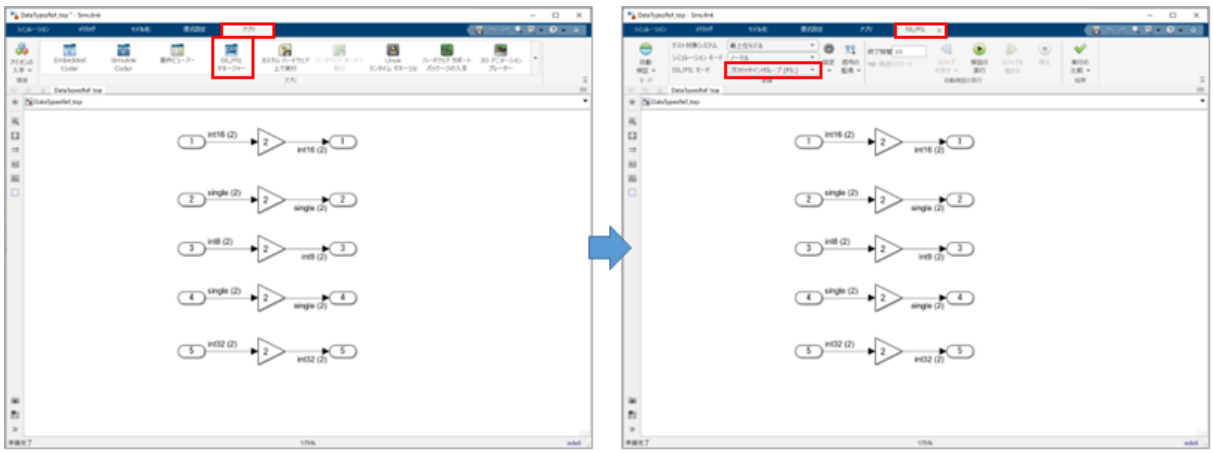

図 1-12 PILモデル設定

4. Simulink®モデルでシミュレーションを開始すると、コード生成から検証環境の作成、PILシミュレーションまでを自動で実行します。

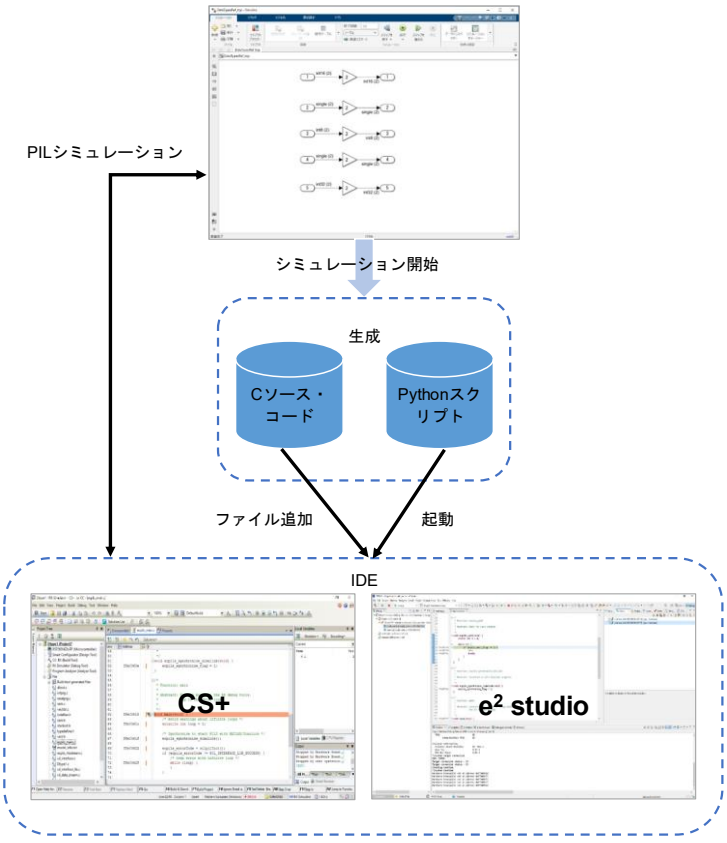

図 1-13 Embedded Targetの処理の流れ(最上位モデルの場合)

2. インストール

本章では、Embedded Target のインストール手順、およびアンインストール手順について説明しています。

2.1 Embedded Targetのインストール

以下に、Embedded Target のパッケージに関する情報と、インストール手順を示します。

2.1.1 パッケージ

Embedded Target をインストールするには、以下のインストーラ・ファイルが必要となります。

- Renesas_Embedded_Target_<バージョン情報>_Setup.exe

インストールを行うと、以下のフォルダ構成でプログラム、ドキュメント、およびサンプルが格納されます。

<バージョン情報>¥	et¥	Embedded Target プログラム一式
	et¥plugins¥IronPython	IronPython 2.7.4 パッケージ
	smp¥	サンプル・モデル

2.1.2 手順

Quick Start Guide を参照し、インストール、ライセンスの追加および初期設定を完了してください。

2.2 Embedded Targetのアンインストール

以下に、Embedded Target のアンインストール手順を示します。

1. MATLAB®を起動し、[パス設定]メニューで起動するパス設定ダイアログで<Embedded Targetインストール・フォルダ>¥<バージョン情報>¥EmbeddedTarget フォルダを削除してください。

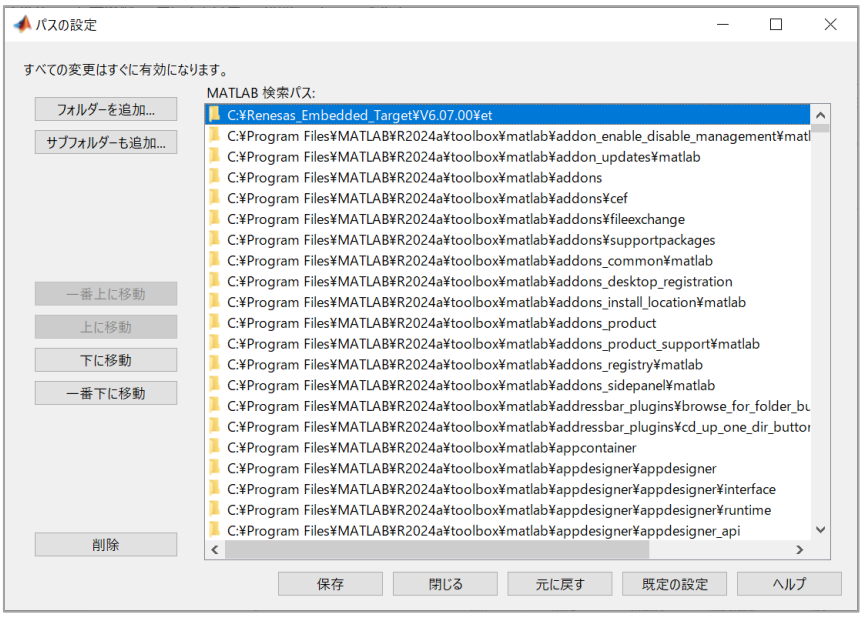

図 2-1 パスの設定ダイアログ

2. <Embedded Targetインストール・フォルダ>¥<バージョン情報>フォルダとインストール時にコピーされた<MATLAB®インストール・フォルダ>¥bin¥win64フォルダを削除してください。

2.3 ライセンスの削除

ライセンスは Renesas License Manager から削除できます。
使用する PC を変更する場合は、現在使用している PC からライセンスを削除した後に新しい PC でライセンスを登録してください。

3. 機能

本章では、Embedded Targetが提供している機能について説明しています。

3.1 概要

Embedded Target では、検証環境の作成機能と、アルゴリズム検証機能を提供しています。
Embedded Target は、Embedded Coder®と連携することにより、検証環境を作成します。

Embedded Target では、サブシステムブロック、参照先モデル、最上位モデルの 3 種をコード生成対象にすることができます。それぞれで検証環境の作成方法が異なります。

表 3-1 3種類のコード生成対象について

コード生成対象	サブシステムブロック	参照先モデル	最上位モデル
生成コードへの入出力	サブシステムブロックの入出力ポートを使用する。	Modelブロックの入出力ポートを使用する。	MATLAB®ワークスペース変数を使用する。
PIL連携	サブシステムブロックをPIL連携ブロックに置き換えてPIL連携を行う。	Modelブロックに対してPILモードを設定してPIL連携を行う。	モデルに対してPILモードを設定してPIL連携を行う。

1. コード生成対象がサブシステムブロックの場合
   サブシステムブロックから PIL 連携用ブロックを生成し、生成した PIL 連携用ブロックをサブシステムブロックと置き換えて検証を行います。

   a. Cコード生成ツール(Embedded Coder®)を使用し、Simulink®モデルからPIL連携用ブロックを生成します。
   b. PIL連携用ブロックとSimulink®モデルのサブシステム化されたブロックを置き換えます。
   c. Cコード生成ツール(Embedded Coder®)を使用し、Simulink®モデルからCソース・ファイルを生成します。
   d. CS+または、e² studioを起動します。
   e. 生成したCソース・ファイルをCS+または、e² studioのプロジェクトに登録します。
   f. PILシミュレーションを実行する際に使用するデバッグツールの種類を選択します。
   g. CS+または、e² studioのビルド・ツールのビルド機能を使用し、Cソース・ファイルからロード・モジュールを生成します。
   h. 生成されたロード・モジュールをデバッグツールにダウンロードします。
   i. PILシミュレーションを行うことにより得られる情報からSimulink®モデルに対するアルゴリズムの検証を行うことができます。

2. コード生成対象が最上位モデル／参照先モデルの場合
   コード生成対象が最上位モデル／参照先モデルの場合は、サブシステムブロックの場合とは異なり、PILモードを指定してシミュレーションを実行することで、検証を行うことができます。検証を開始すると、自動的に C ソース・ファイルと CS+/e2 studio のプロジェクト・ファイルを生成し、それらを使用して CS+/e2 studio と通信を行うことで、検証を行います。

   a. 最上位モデル／参照先モデルを用意します。
   b. コンフィギュレーション パラメーターの設定を行います。
   c. シミュレーションモードにPILモードを指定します。
   d. Cコード生成ツール(Embedded Coder®)を使用し、Simulink®モデルからCソース・ファイルを生成します。
   e. CS+または、e2 studioを起動します。
   f. 生成したCソース・ファイルをCS+または、e2 studioのプロジェクトに登録します。
   g. PILシミュレーションを実行する際に使用するデバッグツールの種類を選択します。
   h. CS+または、e2 studioのビルド・ツールのビルド機能を使用し、Cソース・ファイルからロード・モジュールを生成します。
   i. 生成されたロード・モジュールをデバッグツールにダウンロードします。
   j. PILシミュレーションを行うことにより得られる情報からSimulink®モデルに対するアルゴリズムの検証を行うことができます。

コード生成対象がサブシステムブロックの場合を 3.2 コード生成対象がサブシステムブロックの PIL シミュレーションの実行に、参照先モデルの場合を 3.3 コード生成対象が参照先モデルの PIL シミュレーションの実行に､最上位モデルの場合を 3.4 コード生成対象が最上位モデルの PIL シミュレーションの実行に、それぞれ記述します。

3.2 コード生成対象がサブシステムブロックのPILシミュレーションの実行

以下に、コード生成対象がサブシステムブロックの場合の、PIL シミュレーションを行う際に必要となる検証環境の作成方法を示します。

3.2.1 検証環境の作成

以下に、PIL シミュレーションを行う際に必要となる検証環境の作成方法を示します。この説明では、シングルコア PIL シミュレーションモードに対し、提供のサンプル・モデル DataTypes.slx を使用します。

3.2.1.1 RAファミリSecure Bundleを使用したNon Secureのデバッグ設定の準備

RA TrustZone® Non-Secure Project で PIL シミュレーションを正常に実行するには、RA TrustZone® Secure Project で作成した Smart Bundle ファイル(*.sbd)を参照する必要があります。そのため、本節では、RA TrustZone® Secure Project を生成する方法と、そのために必要な設定について説明します。

- Step 1:e² studioを起動し、次のリンク先の手順に従ってRA TrustZone® Secure Projectを生成する。
  Generating an RA Secure Project for e² studio.

- Step 2: ターゲットデバイスに接続する。
  その後、下図の様にメニューバーの[実行] > [Renesas Debug Tools] > [Renesas Device Partition Manager]を選択する。
  表示された[Renesas Device Partition Manager]ウィンドウの[Device Family]コンボボックスで“Renesas RA”選択し、 [Initialize device back to factory default]チェックボックスをチェックし “Run”を押下し、ウィンドウを閉じる。

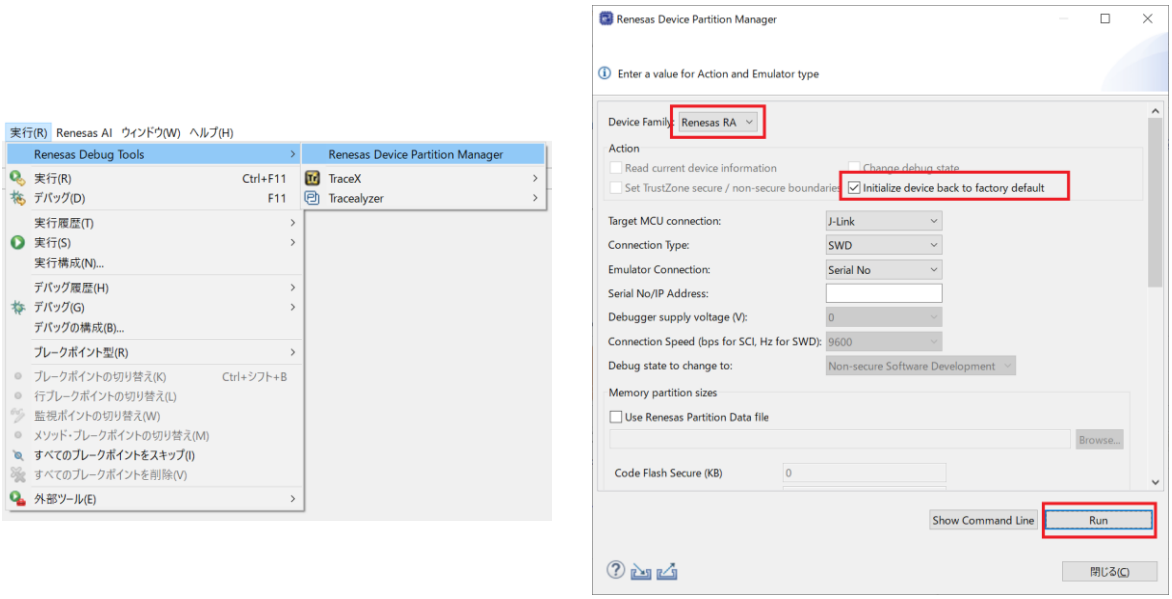

図 3-1 RA TrustZone® Secure用のRenesas Device Partition Manager設定

- Step 3: Step 1で生成したRA TrustZone® Secure projectをビルド＆ダウンロードする。
  ビルド時にビルド・ターゲットと同じフォルダにSmart Bundleファイル(*.sbd)が生成される。

- Step 4: 現在のデバッグセッションを終了する。

- Step 5: ターゲットデバイスから切断後、再度接続する。
  その後、下図の様にメニューバーの[実行] > [Renesas Debug Tools] > [Renesas Device Partition

Manager]を選択する。
表示された[Renesas Device Partition Manager]ウィンドウの[Device Family]コンボボックスで“Renesas RA”を選択し、 [Initialize device back to factory default]チェックボックスのチェックを外し、[Change debug state]チェックボックスをチェックし、[Debug state to change to]コンボボックスで“Non-secure Software Development”を選択し、“Run” を押下し、ウィンドウを閉じる。

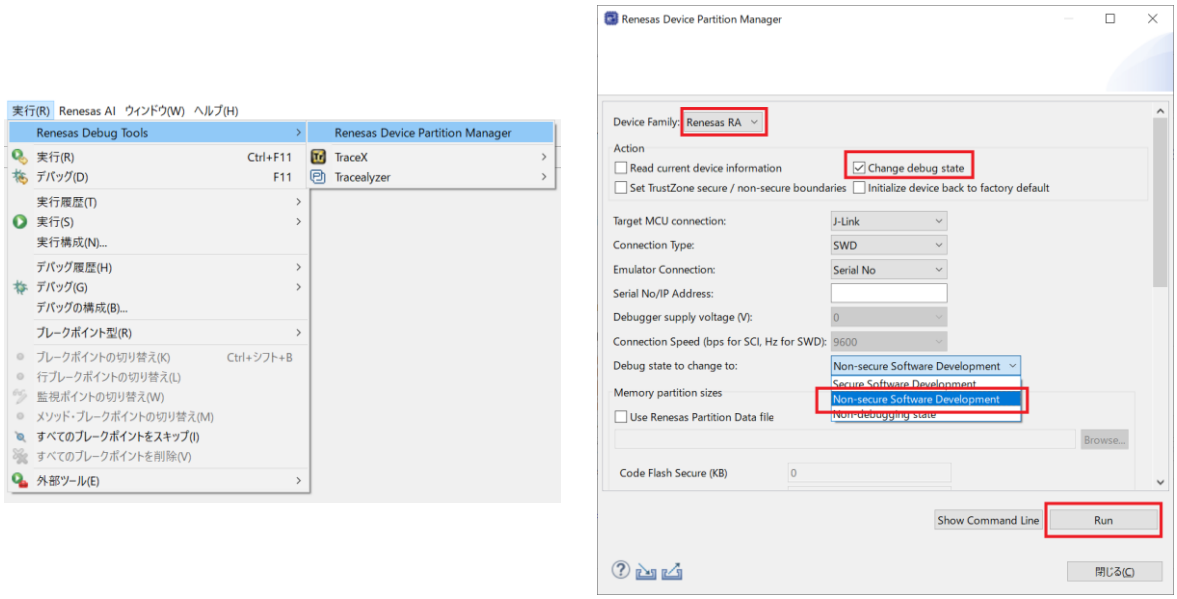

図 3-2 RA TrustZone® Non-Secure用のRenesas Device Partition Manager設定

- ·Step 6: e² studioを閉じる。

3.2.1.2 Simulink®モデルのサブシステム化

Simulink®モデルのブロック群をCソース･ファイルの生成対象となるブロック単位にサブシステム化します。

備考 Embedded Target が提供するサンプル・モデルでは既にサブシステム化されたブロックを用意しています。よって再度サブシステム化する必要はありません。

以下にサンプル・モデルとサブシステムブロック名を示します。

表 3-2 サブシステムブロック名

サンプル・モデル名	コード生成対象	サブシステム化されたブロック名
DataTypes.slx	Subsystem block	DataTypes_PIL

3.2.1.3 コンフィギュレーション パラメーターの設定

Embedded Taret は、Embedded Coder と連携することにより、検証環境の作成を実現しています。したがって、Embedded Target が提供している検証環境の作成処理を利用する際には、Embedded Coder のオプションを確認/設定する必要があります。

1. コンフィギュレーション パラメーター ダイアログのオープン
   DataTypes ウィンドウの[モデル化]メニューから[モデル設定]を選択し、コンフィギュレーション パラメーター ダイアログをオープンします。

2. [ハードウェア実行]オプションの設定
[選択]エリアの[ハードウェア実行]を選択し、[デバイス ベンダー]に[Renesas]を選択後、以下のように各項目の設定を行い、[適用]ボタンをクリックします。

[RX 使用の場合]
[デバイス タイプ]に[RX]を選択します。
また、[バイト順]に[リトルエンディアン]または[ビッグエンディアン]を選択します。

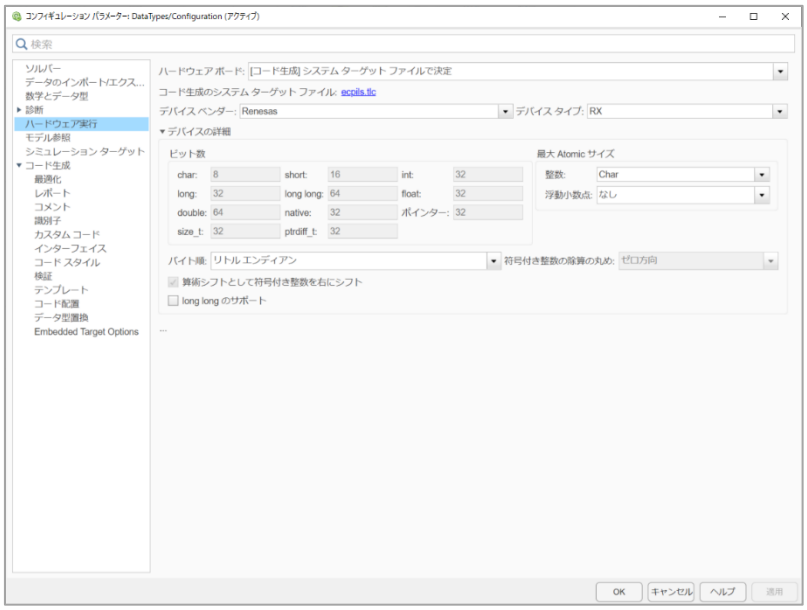

図 3-3 [ハードウェア実行]オプション (RX)

[RL78 または RA 使用の場合]
[デバイス タイプ]に[RL78]または[ARM Cortex]を選択します。
[バイト順]には[リトルエンディアン]が設定され変更はできません。

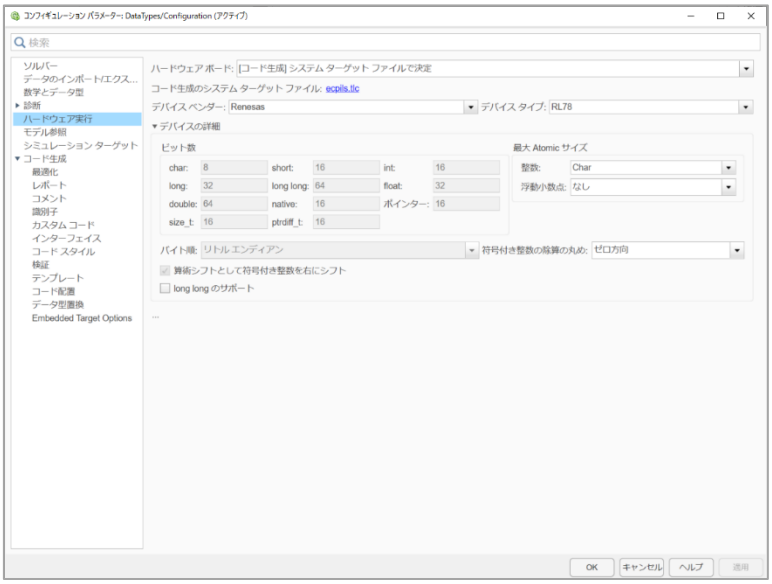

図 3-4 [ハードウェア実行]オプション (RL78)

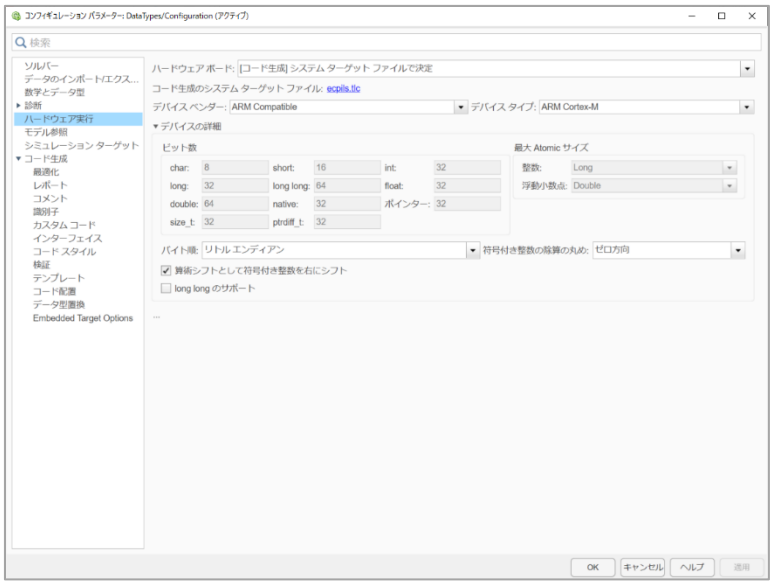

図 3-5 [ハードウェア実行]オプション (RA)

備考 1 [Embedded Target Options]パネルの[Device Family]値を設定することによって、[ハードウェア実行]パネルの[デバイスタイプ]の値を変更することができます。[Embedded Target Options]パネルの“OK”または“適用”ボタンを押してください。

備考 2 [Device Family]値の設定は、[IDE Mode]値が[Embedded Target Options]パネルの[Create Project]の場合に使用可能となります。

3. [コード生成]オプションの設定
[選択]エリアの[コード生成]を選択後、下図と同様の設定を行い、[適用]ボタンをクリックします。

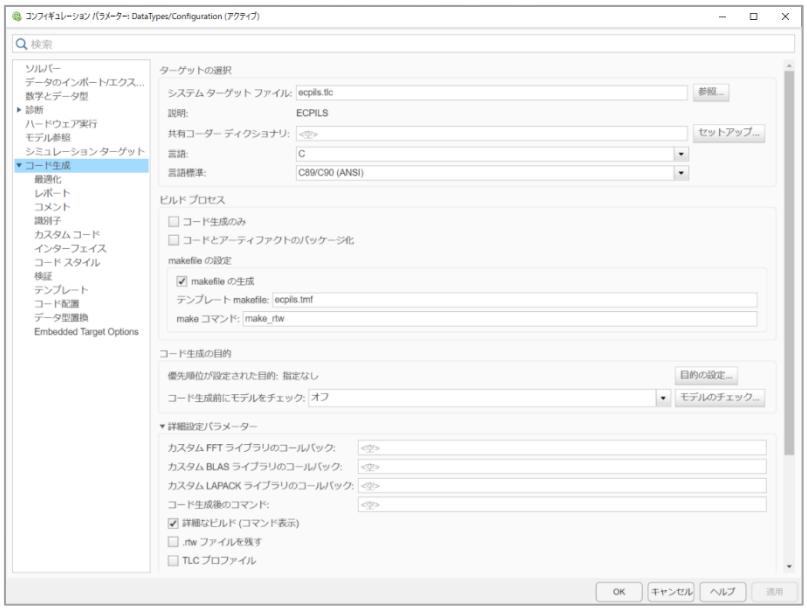

図 3-6 [コード生成]オプション

備考 テンプレート makefile(ecpils.tmf)は選択した Mex コンパイラ(>>mex -setup)によって上書きされます。

4. [検証]オプションの設定
   [選択]エリアの[コード生成] - [検証]を選択後、下図と同様の設定を行い、[適用]ボタンをクリックします。

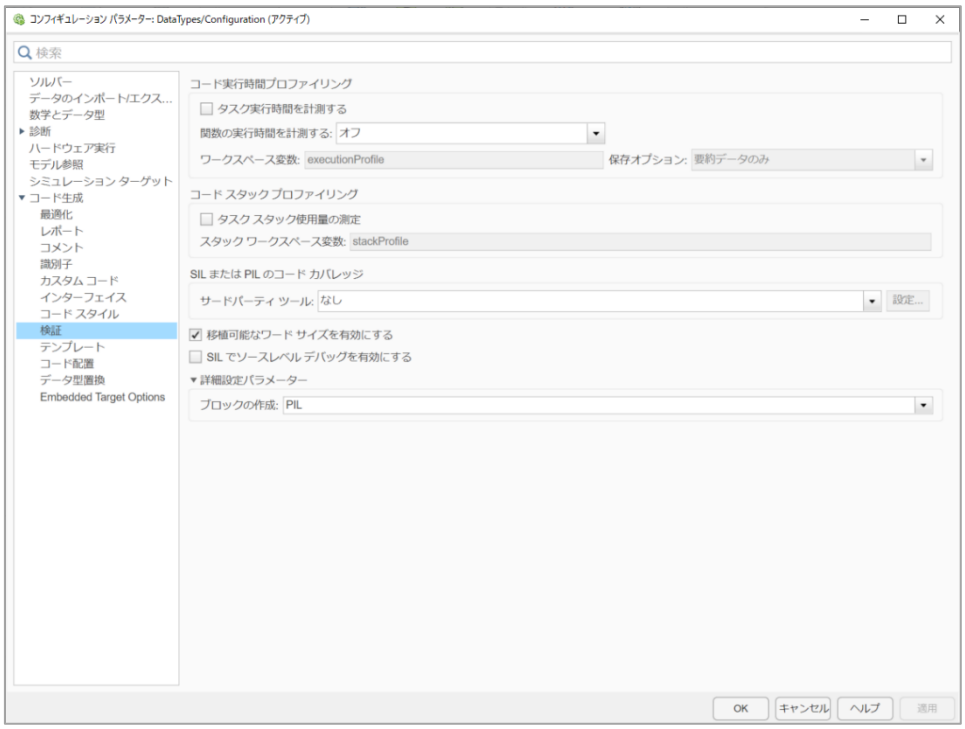

図 3-7 [検証]オプション

5. [Embedded Target Options]オプションの設定
   [選択]エリアの[コード生成] - [Embedded Target Options]を選択後、各項目の設定を行い、[適用]ボタンをクリックします。

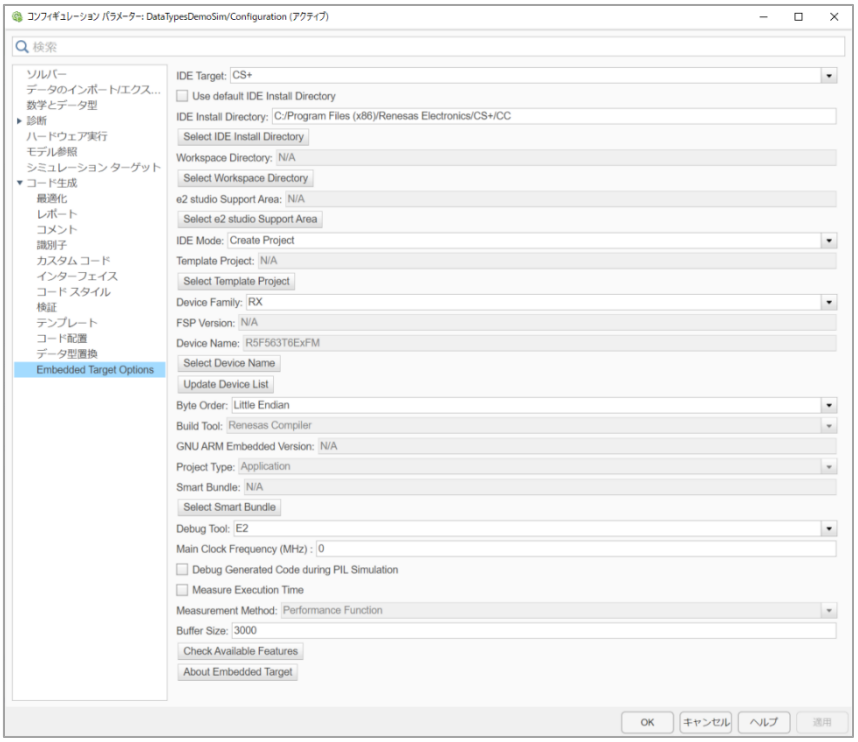

図 3-8 [Embedded Target Options]設定

以下に、[Embedded Target Options]オプションの構成項目を示します。

表 3-3 Embedded Target Options

項目名	説明
IDE Target *1	検証時に使用する IDE を選択します。 IDE は CS+(既定)または、e^2 studio のどちらかを選択可能です。
[Use default IDE Install Directory] チェックボックス	この項目は、選択された IDE Target の既定のインストール・フォルダを[IDE Install Directory]項目に設定します。 設定するインストール・フォルダは以下の通りです。 • CS+: "C:/Program Files (x86)/Renesas Electronics/CS+/CC" • e^2 studio: "C:/Renesas/e2_studio" 既定のインストール・フォルダが無効な場合、本項目は自動でチェックが外れ、[IDE Install Directory]項目が空欄になります。
IDE Install Directory *2	検証時に使用する IDE のインストール・フォルダを絶対パスで指定します。 インストール・フォルダは対象の IDE ごとに以下となります。 CS+： CubeSuiteW+.exe が格納されているフォルダ e^2 studio： e2studio.exe が格納されているフォルダ
[Select IDE Install Directory] ボタン *2 *3	CS+/e^2 studio のインストール・フォルダを絶対パスで指定するためのダイアログを表示します。なお、該当ダイアログにおける指定結果は[IDE Install Directory]に反映されます。
Workspace Directory	e^2 studio プロジェクトが格納されるワークスペース・フォルダを絶対パスで指定します。

[Select Workspace Directory] ボタン *4	e² studio のワークスペース・フォルダを絶対パスで指定するためのダイアログを表示します。なお、該当ダイアログにおける指定結果は[Workspace Directory]に反映されます。	
e2 studio Support Area *5	e² studio が Integration Service API を動作させるために使用する補助フォルダを絶対パスで指定します。	
[Select e² studio Support Area] ボタン *6	e² studio の補助フォルダを絶対パスで指定するためのダイアログを表示します。なお、該当ダイアログにおける指定結果は[e2 studio Support Area]に反映されます。	
IDE Mode *7	CS+/e² studio が起動時に読み込むプロジェクト・ファイルの種類と、起動後にロード・モジュールのダウンロードといった一連の処理の実行の可否を選択します。	
	Create Project	Embedded Target が提供しているデフォルトの CS+/e² studio プロジェクト・ファイルを読み込みます（既定）。
	Open Project	前回終了時の CS+/e² studio プロジェクト・ファイルを読み込みます。
	Open Project & LM Download	前回終了時の CS+/e² studio プロジェクト・ファイルを読み込んで起動した後、LM をデバッグツールにダウンロードします。
	Template Project & LM Download	[Template Project]で指定されている CS+/e² studio プロジェクト・ファイルをベースに生成した CS+/e² studio プロジェクト・ファイルを読み込んで起動した後、LM をデバッグツールにダウンロードします。
Template Project *8	[IDE Mode]に[Template Project & LM Download]を選択した場合、[Select Template Project]ボタンを利用して、テンプレートとして使用する CS+プロジェクト・ファイル名または e² studio のプロジェクト・フォルダ名を絶対パスで指定します。	
[Select Template Project] ボタン *9	テンプレートとして使用する CS+プロジェクト・ファイルまたは e² studio のプロジェクト・フォルダを絶対パスで指定するためのダイアログを表示します。選択した結果を[Template Project]に反映します。	
Device Family *10	使用するマイクロコントローラのシリーズ名を選択します。	
	RX	マイクロコントローラとして、RX デバイス・ファミリを使用します。
	RL78	マイクロコントローラとして、RL78 デバイス・ファミリを使用します。
	RA *11	マイクロコントローラとして、RA デバイス・ファミリを使用します。
FSP Version	RA ファミリのプロジェクト生成に使用する FSP のバージョンを指定します。	
Device Name *12	[Select Device Name]ボタンを利用して、使用するマイクロコントローラの名称を指定します。	
[Select Device Name] ボタン *13	使用するマイクロコントローラの名称を指定するための一覧を表示します。選択した結果を[Device Name]に反映します。	
[Update Device List] ボタン *14	[Select Device Name]ボタンで表示される一覧を最新の情報へ更新します。	
Byte Order *15	マイクロコントローラで使用されるエンディアンを選択します。	
	Little Endian	マイクロコントローラのエンディアンとしてリトルエンディアンを使用します。
	Big Endian	マイクロコントローラのエンディアンとしてビッグエンディアンを使用します。
Build Tool *16	CS+/e² studio プロジェクト用にビルド・ツールを選択します。これはコンパイラがロード・モジュールを作成するために使用されることを意味します。	
	Renesas Compiler	新しいプロジェクトを作成する際、CS+/e² studio によって決められるいずれかのルネサスコンパイラを選択します。

	GCC ARM Embedded Compiler	RA デバイス・ファミリの新しいプロジェクトを作成する際、e² studio によって決められるいずれかの GCC ARM Embedded コンパイラを選択します。
GNU ARM Embedded Version	RA ファミリのプロジェクト生成に使用する GNU ARM Embedded のバージョンを指定します。	
Project Type *17 *20	PIL シミュレーション時に生成する IDE のプロジェクトの種類を選択します。	
	Application	CS+で RX, RL78 ファミリのデバイスを扱う際の規定値です。
	Executable	e²studio で RX, RL78 ファミリのデバイスを扱う際の規定値です。
	Flat (Non-TrustZone®)	e²studio で RA ファミリのデバイスを扱う際の規定値です。RA ファミリのプロジェクトの種類として Flat (Non-TrustZone®)を指定します。
	TrustZone® Secure	RA ファミリのプロジェクトの種類として TrustZone® Secure を指定します。
	TrustZone® Non-Secure	RA ファミリのプロジェクトの種類として TrustZone® Non-secure を指定します。
Smart Bundle *18 *20	RA ファミリプロジェクトの Smart Bundle ファイル(*.sbd)の絶対パスを指定します。	
[Select Smart Bundle] ボタン *19 *20	Smart Bundle ファイル(*.sbd) の絶対パスを指定するためのダイアログを表示します。選択した結果を [Smart Bundle]に反映します。	
Debug Tool *21	対象デバイスに接続するデバッグツールと接続タイプを選択します。	
	E2 (既定)	E2 エミュレータを使用し、対象デバイスに接続します。
	E2Lite	E2 Lite エミュレータを使用し、対象デバイスに接続します。
	COM Port	COM Port を使用し、対象デバイスに接続します。(RL78/G23 シリーズのみ)
	Simulator *22	対象デバイスのシミュレータを使用します。
	E20(Serial)	シリアル接続で E20 エミュレータを使用し、対象デバイスに接続します。
	E20(JTAG)	JTAG 接続で E20 エミュレータを使用し、対象デバイスに接続します。
	EZ_Emulator *23	EZ emulator を使用し、対象デバイスに接続します。
	E1(JTAG)	JTAG 接続で E1 エミュレータを使用し、対象デバイスに接続します。
	E1(Serial)	シリアル接続で E1 エミュレータを使用し、対象デバイスに接続します。
	J-Link	J-Link エミュレータを使用し、対象デバイスに接続します。
Main Clock Frequency (MHz) *24	対象デバイスのメインクロックの周波数を設定します。	
Debug Generated Code during PIL Simulation	チェックあり	CS+/e² studio でのユーザの操作によって PIL シミュレーションを中断しロード・モジュールのデバッグが可能です。 実行時間測定機能は無効になり、[Measure Execution Time]チェックボックスを選択することはできません。
	チェックなし	PIL シミュレーションは、CS+/e2 studio でのユーザの操作によって中断できません。実行時間測定機能を有効にすることができます。

Measure Execution Time *22	チェックあり	アルゴリズムの検証において、実行時間を測定します。
	チェックなし	アルゴリズムの検証において、実行時間を測定せず、高速に PIL シミュレーションを実行します。（既定）
Measurement Method	使用中の時間測定方法を選択します。	
	Performance Function	PIL シミュレーションの間、CS+のロード・モジュールの実行時間は、CS+/e^2 studio デバッグツールのパフォーマンス機能で測定されます。
Buffer Size	バッファサイズを 1000～3000 の範囲で指定します。	
[Check Available Features] ボタン *25	Embedded Target システムの使用可能な必須機能のリストを表示します。	
[About Embedded Target] ボタン	Embedded Target のバージョン情報およびコピーライト情報を表示します。	

*1… [IDE Target]で CS+を選択したとき、[IDE Install Directory]は CS+のデフォルトのインストール・フォルダのパスに変更され、[Workspace Directory]と[e2 studio Support Area]は無効となり編集不可能となります。
また[IDE Target]で e2 studio を選択したときは[IDE Install Directory]は e² studio のデフォルトのインストール・フォルダのパスに変更され、[Workspace Directory]と[e2 studio Support Area]は有効となり編集可能となります。

*2… ダイアログで指定したフォルダに CS+/e² studio がインストールされていない（指定したフォルダに CubeSuiteW+.exe/e2studio.exe ファイルが存在しない）場合は、エラーを出力し、指定したフォルダの情報は[IDE Install Directory]に反映されません。

*3… [Use default IDE Install Directory]チェックボックスがチェックされている場合、[Select IDE Install Directory]ボタンを押下するとエラー・メッセージを表示します。

*4… [IDE Target]が“CS+”のとき、[Select Workspace Directory]ボタンを押下するとエラー・メッセージを表示します。

*5… [e2 studio Support Area]を指定するには以下の手順を行ってください。

1. e² studio で[ヘルプ] - [e2 studio について]メニュー -> [インストール詳細]ボタンで[e² studio のインストール詳細]ダイアログを表示する。
2. [Support Folders]タブで“e² studio support area”の絶対パスを確認できる

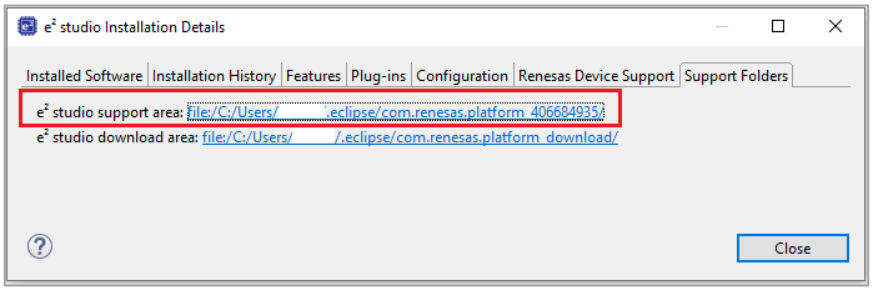

図 3-9 “e2 studio support area”の絶対パス

*6… [IDE Target]が“CS+”のとき、[Select e2 studio Support Area]ボタンを押下するとエラー・メッセージを表示します。

*7… “Open Project”、”Open Project & LM Download”選択時で、前回終了時のプロジェクト・ファイルがない場合は、“Create Project”選択時と同様、デフォルト・プロジェクト・ファイルを作成した後、CS+/e² studio を起動します。

*8… [IDE Mode]に“Template Project & LM Download”が選択されている場合のみ設定が有効になります。
Embedded Target で作成した CS+のプロジェクト・ファイルか e² studio のプロジェクト・フォルダを指定してください。Embedded Target 以外で作成された CS+プロジェクト・ファイルか e² studio のプロジェクト・フォルダを指定した場合は、動作保証しません。

*9… [IDE Mode]に“Template Project & LM Download”以外が選択されている場合に押下した場合は、エラーになります。

*10.. [IDE Mode]に“Create Project”が選択されているかつ、使用する IDE が選択したデバイスとコンパイラに対応している場合のみ設定が有効になります。
“RX”の選択肢は、“Embedded Target for RX”ライセンスが有効な場合のみ、[Device Family]リストで選択することが可能です。
“RL78”の選択肢は、“Embedded Target for RL78”ライセンスが有効な場合のみ、[Device Family]リストで選択することが可能です。
“RA”の選択肢は、“Embedded Target for RA”ライセンスが有効な場合のみ、[Device Family]リストで選択することが可能です。
[Check Available Features]ボタンをクリックして、有効性を確認してください。

*11.. e² studio で RA デバイス・ファミリのプロジェクト作成するとき、e² studio に含まれている FSP または GCC ARM Embedded のバージョンが異なる場合があります。FSP または GCC ARM Embedded の指定のバージョンが正しくない場合は、プロジェクトの作成に失敗します。正しいバージョンを指定するには“Embedded Target Options”の”FSP Version”または”GNU ARM Embedded Version”の値を変更してください。

*12.. 前回終了時の CS+プロジェクト・ファイルまたは e² studio プロジェクト・フォルダが存在しない場合は、前回指定された情報を使用してデバイス名を生成します。

*13.. [IDE Mode]に“Create Project”が選択されている場合のみ、ボタンが表示されます。
[Device Family]で選択されているマイクロコントローラのシリーズの一覧のみ表示されます。
[IDE Mode]に“Open Project”、“Open Project & LM Download”および “Template Project & LM Download”が選択されている場合に押下した場合は、一覧は表示されず、エラーになります。

*14.. CS+/e^2 studio から最新情報を取得するため、CS+/e^2 studio が起動しますが、自動的に終了します。CS+/e^2 studio が終了したタイミングで情報が更新されます。

*15.. [Byte Order]は[Device Family]で”RX”が選択されている場合に有効になります。
[Device Family]で”RL78”または”RA”が選択されている場合、[Byte Order]は”Little Endian”に固定され変更不能になります。

*16.. [Build Tool]が“Renesas Compiler”または”GCC ARM Embedded”に設定されている場合、実際に使用するビルド・ツールは、プロジェクト作成時に CS+/e^2 studio によって決定します。

*17.. [Project Type]は TrustZone®をサポートする RA デバイスを選択した場合のみ有効です。それ以外の場合は、[Project Type]の値が初期値に自動的に変更されます。

18.. [Smart Bundle]は、RA TrustZone® Secure プロジェクトタイプの Smart Bundle ファイル(.sbd)を参照する必要がある RA TrustZone® Non-Secure プロジェクトタイプでのみ使用されます。そのため、RA TrustZone® Non-Secure プロジェクトを正常に作成するには、RA TrustZone® Secure プロジェクトタイプの Smart Bundle ファイル(*.sbd)ファイルを[Smart Bundle]に指定する必要があります。
[Smart Bundle]を指定するには、”3.2.1.1 RA ファミリ Secure Bundle を使用した Non Secure のデバッグ設定の準備”で作成した RA TrustZone® Secure の Smart Bundle ファイル(*.sbd)を取得します。

*19.. [Device Family]が”RA”と異なる場合、または[Project Type]が「TrustZone® Non-Secure」と異なる場合、[Select Smart Bundle]ボタンをクリックすると、エラー・メッセージが表示されます。

*20.. これらの項目は、e^2 studio 2024-07 でのみ利用可能な RA TrustZone®プロジェクトタイプをサポートするために使用されます

*21.. Embedded Target を介したいずれかの E1 接続タイプの実際の対象デバイス（LSI デバイス）への接続は、CS+または e^2 studio の GUI での手動接続と同様です。効果的に使用するため、CS+/e^2 studio の同じ対象デバイス用にダミーのプロジェクトを作成してください。その後、使用可能な E1 エミュレータの接続タイプを介し、対象デバイスに接続します。成功したら、同じ E1 接続タイプとメインクロック周波数の値を PIL シミュレーション用 Embedded Target に適用します。
COM port は RL78/G23 シリーズでのみサポートします。
J-Link は RA デバイス・ファミリと RX デバイス・ファミリの一部のデバイスでかつ[IDE Target]が e^2 studio の場合のみサポートします。
RL78/F25 シリーズは CS+ V8.12.00 以上で Debug Tool は E2, E2 Lite, Simulator のみをサポートします。
[Device Family]リストが “RX”または”RL78”または”RA”に設定されているかつ[IDE Mode]が”Create Project”の場合、このリストを設定することができます。
デフォルトの E2 エミュレータのタイプは対応する CS+/e^2 studio パッケージが使用されます。

*22.. [Measure Execution Time]はいくつかのケースでサポートされません。詳細は制限事項「4.3 時間計測機能をサポートできない」を参照してください。

*23... CS+ V8.09.00 以上と e^2 studio 2023-01 以上では、RX デバイス・ファミリでの EZ emulator はサポートされません。

*24.. [Debug Tool]リストが“E1”(Jtag/Serial)、“E20”(Jtag/Serial)、“E2”、“E2 Lite”、“EZ emulator”接続タイプのいずれかに設定されている場合、このテキストボックスの変更が可能となります

*25.. このボタンをクリックすると、ダイアログボックスが表示され必要な機能のリストを示します。ライセンスが不要な機能はこのボックスに表示されません。これらのボタンは自由に使用することができます。

備考 各ツールのインストール・ディレクトリ・パスを直接入力する場合
1. ￥”は使用不可。代わりに“/”を使用してください。
2. インストール・ディレクトリ・パスの最後に “/”を置かないでください。

6. コンフィギュレーション パラメーター・ダイアログのクローズ
設定内容を確認後、[OK]ボタンをクリックします。

3.2.1.4 検証環境の作成

以下に、サブシステムブロックを使用して PIL シミュレーションを行う際に必要となる検証環境の作成方法を示します。
Embedded Target では、以下に示したコマンドを MATLAB®のコマンド・ウィンドウで利用可能なコマンドとして提供しています。サブシステムブロックを使用する場合に、環境構築の際の一連の処理を自動で行うことができます。

表 3-4 提供コマンド

Command nameコマンド名	Description説明
ecpils_build	検証環境の作成処理の実行(サブシステムブロックを使用する場合のみ)

検証環境の作成の実行には、以下の方法があります。

DataTypes ウィンドウのサブシステム化されたブロックを選択し、以下の方法で検証環境の作成を実行してください。

コマンド・ウィンドウからの実行。
以下の形式で Embedded Target が提供している ecpils_build コマンドを MATLAB®のコマンド・ウィンドウから発行し、検証環境の作成をします。
なお、”>>”はコマンド・プロンプトを、”[Enter]”はエンター・キーの入力を表しています。

```
>> ecpils_build [Enter]
```

備考 本方法で“検証環境の作成”を行った場合、以下の操作も合わせて行われます。
したがって、”3.2.1.5 PIL シミュレーション用 PIL 連携ブロックの置き換え”の実施は必要ありません。

- 選択されたサブシステム化されたブロックを含むモデル・ファイルをコピー（コピー先のファイル名には、元のファイル名にサフィックスとして”_ecpils” を付与）

- コピー先のモデル・ファイルに対して、サブシステム化されたブロックと PIL 連携用ブロックの置き換えを実施

ecpils_build コマンド実行前に、編集したモデル・ファイルは保存してください。

3.2.1.5 PILシミュレーション用PIL連携ブロックの置き換え

検証環境の作成を MATLAB®のコマンド・ウィンドウから実行した場合、“ブロックの書き換え”は検証環境の作成の一連の処理として行われるため、ユーザが明示的な操作を行う必要がありません。
以下に、検証環境の作成を DataTypes ウィンドウから実行した際に必要となるブロックの置き換え方法を示します。

- ブロックの置き換え方法
  DataTypes ウィンドウのサブシステム化されたブロック“DataTypes”ブロックを削除し、untitled ウィンドウの PIL 連携用ブロック” DataTypes ブロック”をドラッグし、DataTypes ウィンドウの該当部位にドロップし、PIL シミュレーション・モデルを完成させます。この際、DataTypes ウィンドウは別名にて保存してください。そのまま保存すると元のサブシステムブロックが上書きされてしまいます。

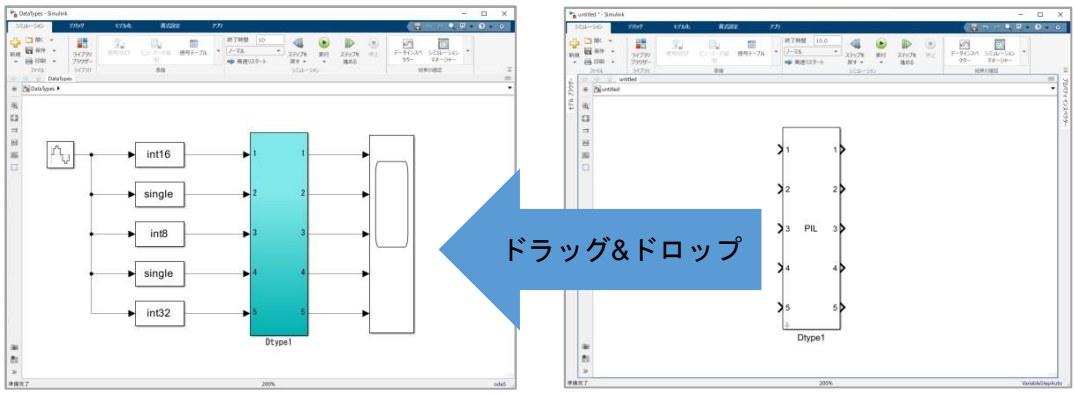

図 3-10 PIL連携用ブロックの置き換え

備考 1 モデル・ファイル上に作成された、 PIL 連携用ブロックを元のサブシステムと入れ替えます。untitled モデルは、保存せずに閉じても問題ありません。

備考 2 PIL 連携用ブロックをモデル内に、複数配置することはできません。

3.2.1.6 ロード・モジュールの生成

コンフィギュレーション パラメーター ダイアログの[Embedded Target options]オプションで、[IDE Mode]で“Open Project & LM Download”や“Template Project & LM Download”を設定した場合、ロード・モジュールの生成、およびダウンロードは、検証環境の作成の一連の処理とし自動的に行われるため、ユーザが明示的な操作を行う必要がありません。
以下に、[IDE Mode]に“Create Project”や”Open Project”を設定した際のロード・モジュールの生成方法を示します。

- ロード・モジュールの生成方法

  [IDE に CS+を使用する場合]

  (1) CS+のプロパティ・パネルにおいてビルド・ツール（コンパイラ、アセンブラなど）のオプション設定を行います。特に、以下の場合は、CS+デフォルト設定から変更する必要があります。

  [RX においてビッグエンディアンを使用する場合]
  CS+プロジェクト・ツリーで[ビルド・ツール]を選択し、[共通オプション]-[CPU]-[データのエンディアン]に[Big-endian データ(-endian=big)]を設定します。
  この時、「デバッグ・ツールのエンディアン設定も変更しますか？」というメッセージ・ダイアログが出力されるので、「はい」を選択します。

  [RL78 を使用する場合]
  CS+プロジェクト・ツリーで[ビルド・ツール]を選択し、* [コンパイル・オプション]-[出力コード]-[double 型／long double 型を float 型として処理する]を” いいえ(-dbl_size=8)”に設定してください。
  そして、[リンク・オプション]-[デバイス]-[オンチップ・デバッグ・オプション・バイト制御値]と[ユーザ・オプション・バイト値]に値を設定します。そして、[デバッグ・モニタ領域を設定する]に[はい(範囲指定)(-DEBUG_MONITOR=<アドレス範囲>)]に設定し、[デバッグ・モニタ領域の範囲]に値を設定します。これらのプロパティの設定値は各デバイスのハードウェア・ユーザ・マニュアルを参照してください。
  コンフィギュレーション パラメーター ダイアログの[Embedded Target Options]オプションで[Debug Generated Code during PIL Simulation]がチェックされている場合、[コンパイル・オプション]-[最適化(詳細)]-[関数のインライン展開を行う]に[いいえ(-Oinline_level=0)]を設定してください。

  * このオプションは RL78 S3 コアのデバイスで 2 バイト以上のデータ型を使用するときにのみ設定してください。

  (2) CS+プロジェクトを上書き保存します。

  (3) CS+の［ビルド］メニューから［ビルド・プロジェクト］を選択します。

[IDE に e2 studio を使用する場合]

(1) e² studio のプロパティ・パネルにおいてビルド・ツールのオプション設定を行います。

[RL78 を使用する場合]

プロジェクト上での右クリックから[プロパティ]メニュー選択し、ビルド・ツールの設定ダイアログの[C/C++ビルド]-[設定]画面でツール設定タブを開きます。* [Compiler]-[出力コード]カテゴリで[double 型／long double 型を float 型として処理する(-dbl_size)]のチェックを外し、[Linker]-[デバイス]カテゴリで[オプション・バイト領域のユーザ・オプション・バイトに値を設定する(-user_opt_byte)], [オプション・バイト領域のオンチップ・デバッグ・オプションバイトに値を設定する(-ocdbg)]および[OCD モニタのメモリ領域を確保する(-debug_monitor)]をチェックし値を設定します。これらのプロパティの設定値は各デバイスのハードウェア・ユーザ・マニュアルを参照してください。

コンフィギュレーション パラメーター ダイアログの[Embedded Target Options]オプションで[Debug Generated Code during PIL Simulation]がチェックされている場合、[Compiler]-[最適化]-[関数の自動インライン展開を行う(-Oinline_level)]に[いいえ]を設定してください。

[RA8T1 シリーズを使用する場合]

クロックがデバイスに合っていることを確認するには、以下の手順に従ってください。

(1) [プロジェクト・エクスプローラー]パネルで”configuration.xml”をダブルクリックし、FSP Configuration を開く

(2) [BSP]タブで[Board]、[Device]で適切なボードとデバイスを選択する

* このオプションの設定は RL78 S3 コアのデバイスで 2 バイト以上のデータ型を使用するときにのみ行ってください。

(2) 以下のいずれかの方法でビルドします。

- プロジェクト上での右クリックからの[プロジェクトのビルド]メニュー選択
- プロジェクトにフォーカスしてからの[プロジェクト] □ [プロジェクトのビルド]メニュー選択
- プロジェクトにフォーカスしてからの アイコンのクリック
- プロジェクトにフォーカスしてからの[Ctrl] + [B]キー押下

備考 ロード・モジュールの生成方法についての詳細は、CS+/e² studio のユーザーズマニュアルを参照してください。

3.2.1.7 ロード・モジュールのダウンロード

コンフィギュレーション パラメーター ダイアログの[Embedded Target Options]オプションで[IDE Mode]に“Open Project & LM Download” や“Template Project & LM Download”を設定した場合、“ロード・モジュールのダウンロード”は検証環境の作成の一連の処理として自動的に行われるため、ユーザが明示的な操作を行う必要はありません。
以下に、[IDE Mode]]に”Create Project”、または”Open Project”を設定した際に必要となるデバッグツールの設定、および、ロード・モジュールのダウンロード方法を示します。

- デバッグツールの設定方法

  [IDE に CS+、デバイスに RX または RL78、エミュレータに E1/E2/E2 Lite/E20 (Jtag/Serial)/EZ Emulator または COM Port を使用の場合]

  CS+プロジェクト・ツリーで[デバッグ・ツール]を選択し、プロパティ設定の[デバッグ・ツール設定]-[実行中のメモリ・アクセス]-[実行中にアクセスする]（[実行を一瞬停止してアクセスする]の場合あり）を[はい]に設定してください。

[IDE に e2 studio、デバイスに RX、エミュレータに E1/E2/E2 Lite/E20 (Jtag/Serial)/EZ Emulator、ビッグエンディアンを使用の場合]

(1) [プロジェクト・エクスプローラ]ペインの“Tutorial”プロジェクトをクリックしてフォーカスを移動します。

(2) [実行] → [デバッグの構成...]メニューもしくは アイコンのドロップダウン → [デバッグの構成...]メニューから[デバッグ構成]ウインドウを開きます。

(3) [デバッグ構成]ウインドウで[Renesas GDB Hardware Debugging] □ [Tutorial HardwareDebug]に移動し、[Debugger]タブに切り替えます。
[Debugger]タブ下の[デバッグ・ツール設定]サブ・タブに移動し、[メモリ] – [エンディアン]プロパティの値を“ビッグ・エンディアン”に設定します。

- ロード・モジュールのダウンロード方法
  [IDE に CS+を使用の場合]
  CS+のプロパティ・パネルにおいてデバッグ・ツール（E1/E2/E2 Lite/E20(JTAG/シリアル)/EZ emulator、J-Link もしくは COM port、シミュレータ）のオプション設定を行ったのち、CS+の［デバッグ］メニューから［デバッグ・ツールへダウンロード］を選択することによりロード・モジュールのダウンロードが行われます。

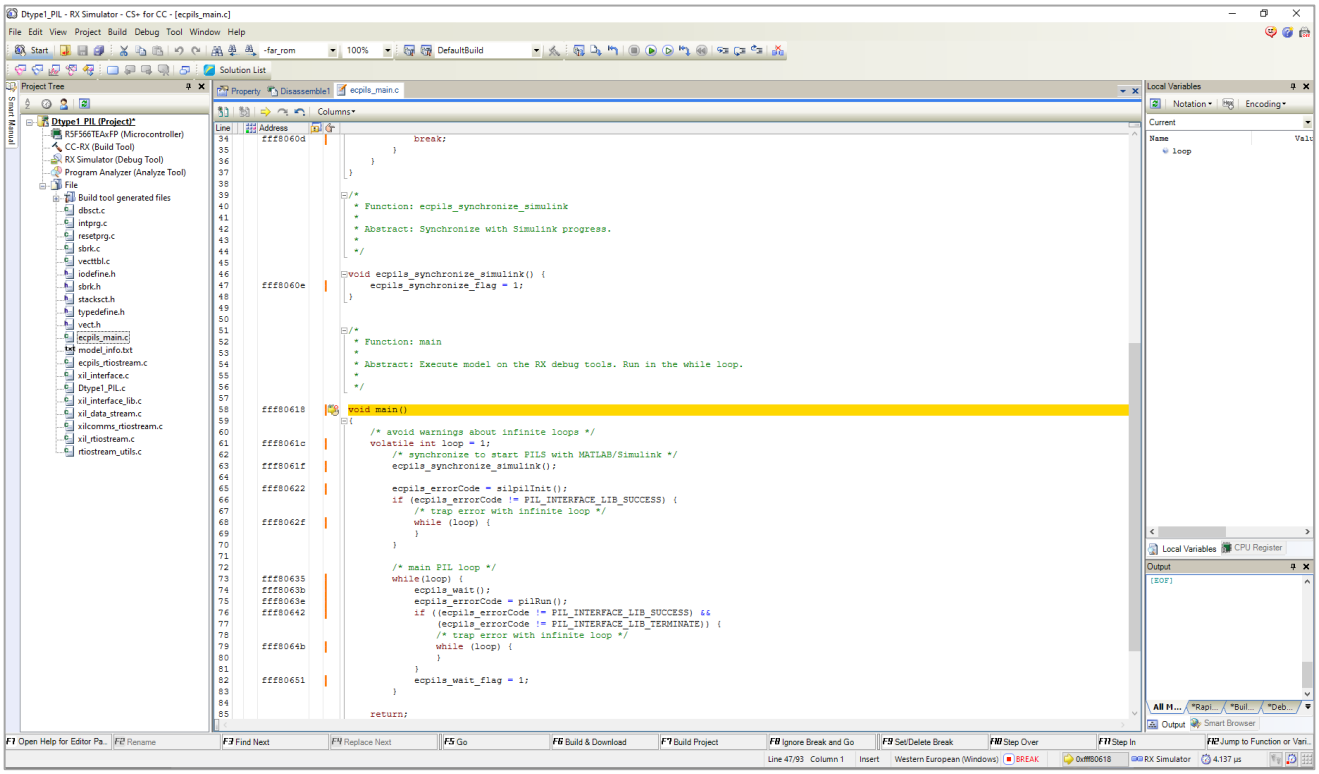

図 3-11 デバッガへのダウンロードが完了した状態のCS+

[IDE に e2 studio を使用の場合]
[プロジェクト・エクスプローラ]パネル上で対象プロジェクトをクリックしフォーカスを移動する。その後、  アイコンをクリックしデバッガ・セッションの開始とロードモジュールのダウンロードを行う。

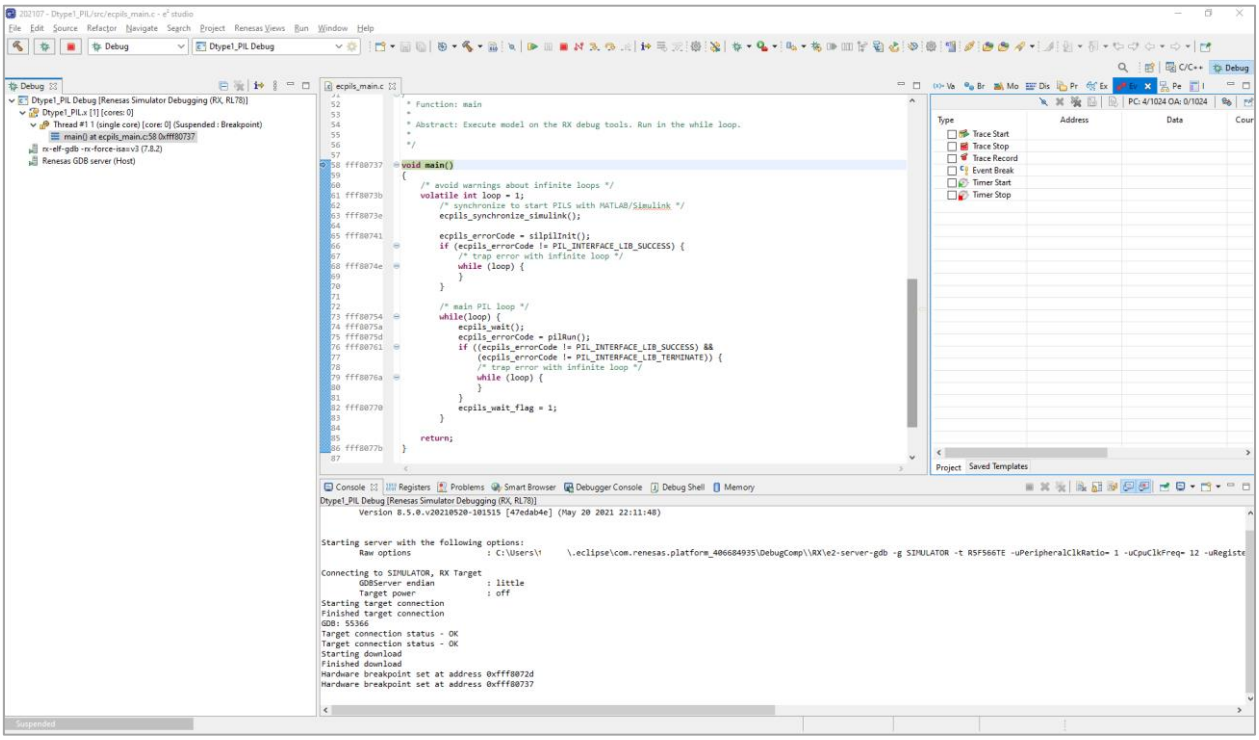

図 3-12 デバッガへのダウンロードが完了した状態のe2 studio

3.2.2 PILシミュレーションの実行

以下に、検証環境の作成を DataTypes ウィンドウから実行した際に必要となる、PIL シミュレーションの実行方法を示します。

- PILシミュレーションの実行方法
  DataTypes ウィンドウからの内容が PIL シミュレーション・モデルへと変わっていることを確認後、[シミュレーション]メニューから[実行]を選択し、PIL シミュレーションを開始します。

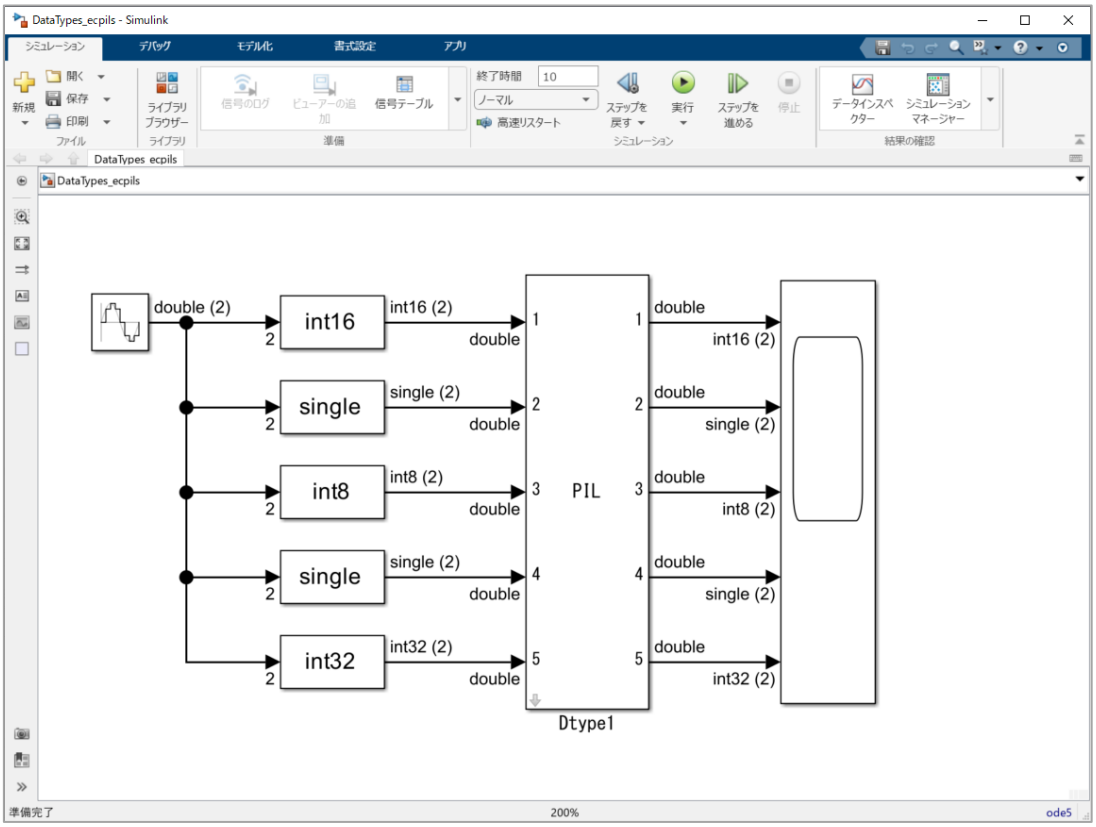

図 3-13 PILシミュレーションの実行

備考 本操作を実行後、モデル・ファイルの保存を行った場合、元のモデル・ファイルに対する上書き保存が行われます。

Embedded Target がダウンロード完了を確認するまでは、確認ダイアログが表示されます。確認ダイアログは、ダウンロード完了後に自動的に閉じます。ダウンロードを中止して一連の動作を中断したい場合は、確認ダイアログの OK ボタンを押下してください。

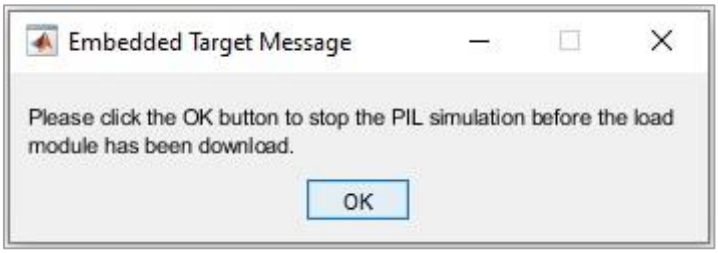

図 3-14 確認ダイアログ

3.2.3 PILシミュレーション中の生成コードのデバッグ

このセクションでは PIL シミュレーション中にモデルから生成したコードをデバッグする方法について説明します。
以下の手順で CS+/e2 studio のデバッグツールが提供するすべての機能を使用することができます。

- Step 1: モデルを開き、コード生成対象のサブシステムを選択して[コンフィギュレーション パラメーター]ウィンドウを表示します。
- Step 2: 必要な条件を設定し、[Debug Generated Code during PIL Simulation] チェックボックスをチェックします。
- Step 3: Save model and execute MATLAB® command: “>> ecpils_build [Enter]” モデルを保存し、次のMATLAB®コマンドを実行します。
  “>> ecpils_build [Enter]”
- Step 4: CS+/e² studioプロジェクトをビルド＆ダウンロードします。
  “ecpils_main.c”の関数main()と*ecpils_synchronize_simulink® ()*に自動的にブレーク・ポイントが設定されます。
- Step 5: Simulink®モデルの [Run]ボタンを押下し、デバッグを開始します。
  - ステップ実行
  - コードブロック、関数へのステップイン
  - 停止
  - その他

このモードを有効にするには、 コンフィギュレーション パラメーターの設定時に[Debug Generated Code during PIL Simulation] チェックボックスをチェックしてください。下図はその例です。

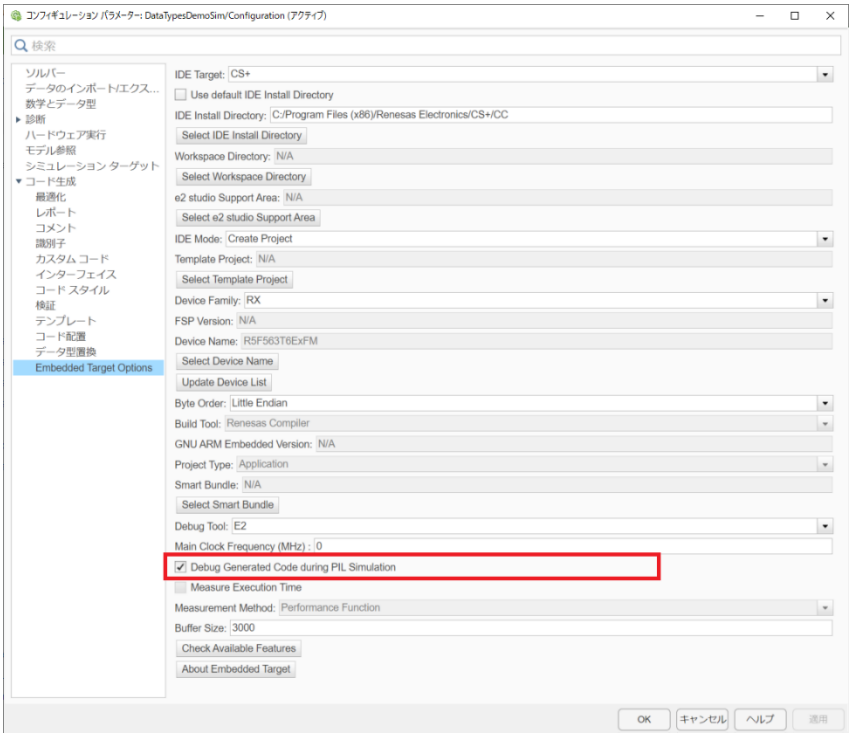

図 3-15 PILシミュレーション中の生成コードデバッグの有効化

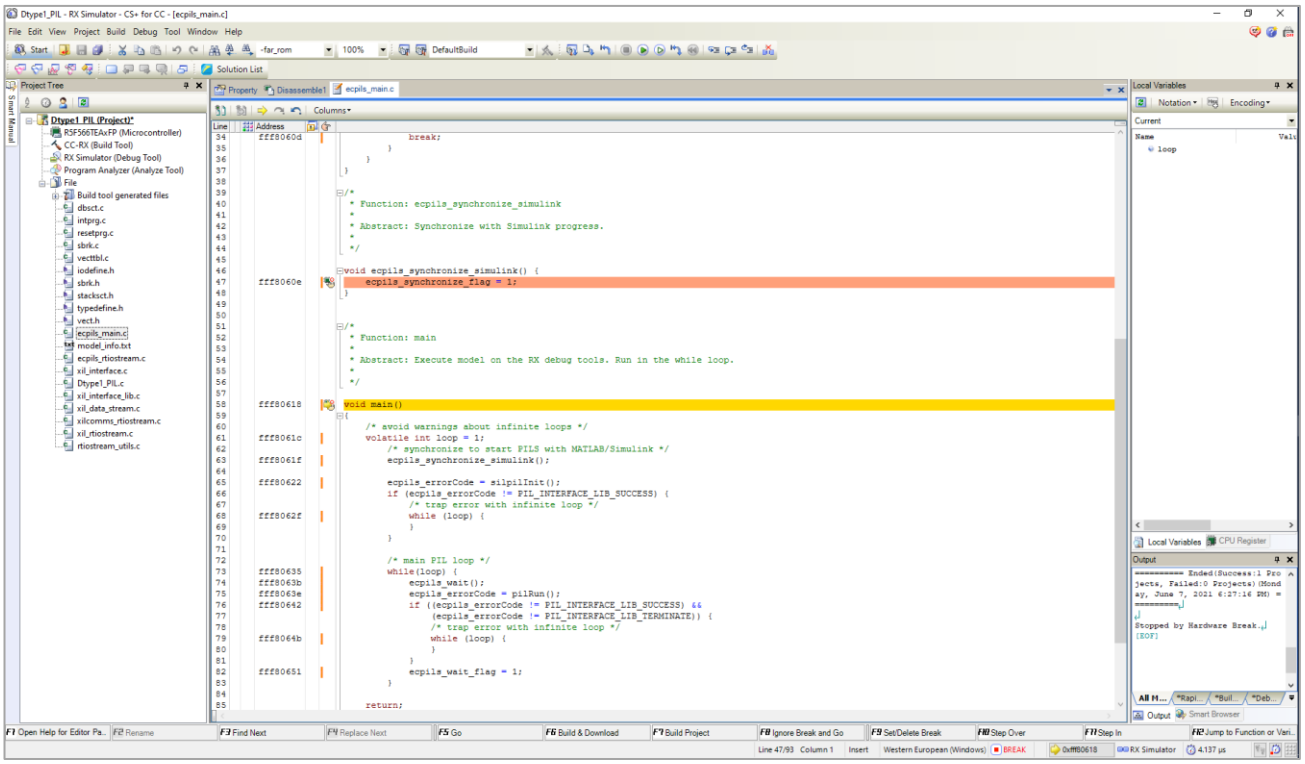

図 3-16 ecpils_main.cに自動的に設定される2つのブレーク・ポイント(CS+)

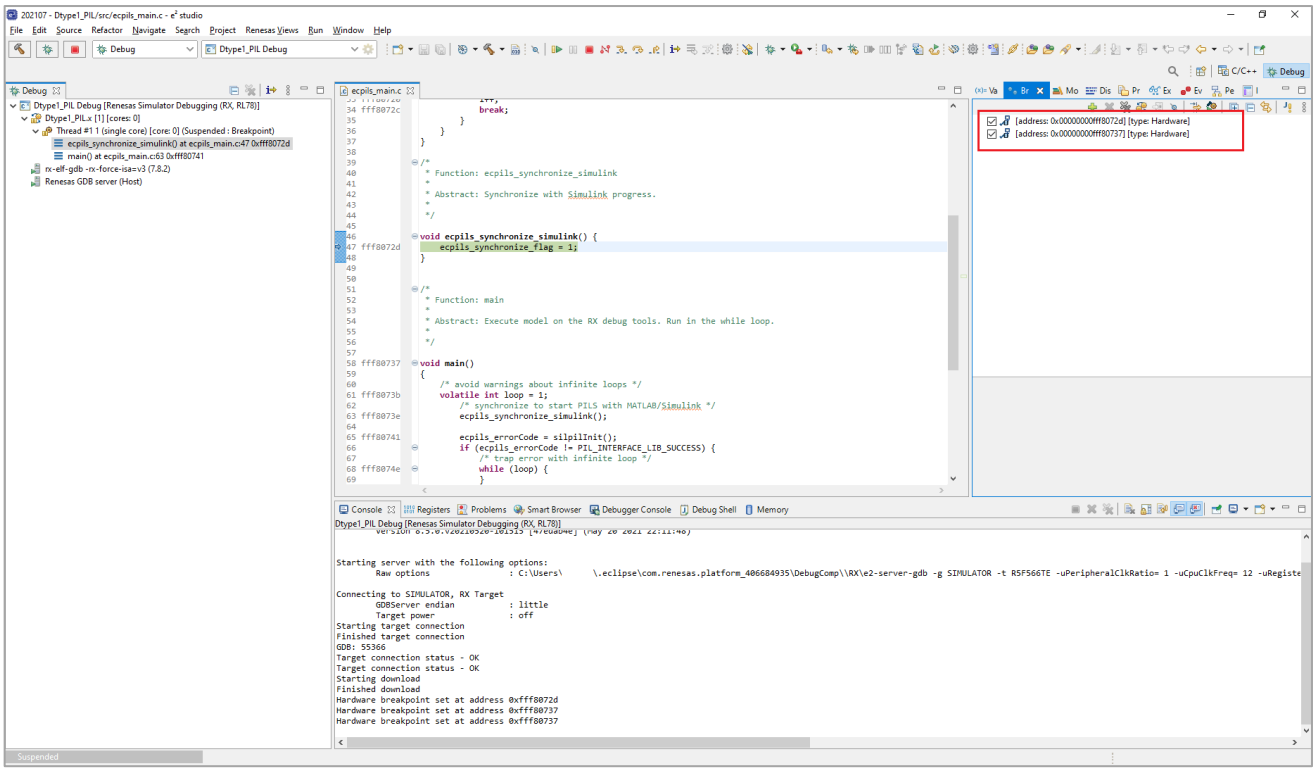

図 3-17 ecpils_main.cに自動的に設定される2つのブレーク・ポイント(e² studio)

備考 1. この機能を使用する場合、実行時間は計測できません。[Measure Execution Time] チェックボックスは無効になります。
備考 2. デバッグ中のステップ実行のタイムアウト時間は最大 24 時間です。タイムアウトを検出すると PIL シミュレーションプロセスは停止します。
備考 3. ターゲット・プログラム(CS+/e² studio)が停止した場合、Simulink®の画面は固まります。Simulink®上の GUI から一時停止や停止はできません。これは MATLAB®/Simulink®の制限となります。PIL シミュレーションを停止するには、デバッグ中に設定したブレーク・ポイントを削除し、プログラムが終了するまで実行してください。
備考 4. デバッグ中にデバッグツールから切断したり、CS+/e² studio のプロセスを終了したりしないでください。予期しない動作の原因と成ります。
備考 5. Embedded Target で生成したコードの変更や、PC の値の変更他、PIL シミュレーションのワークフローを変えるような変更を CS+/e² studio 側で行わないでください。これらの操作は PIL シミュレーションプロセスにとって不正でエラーの原因となります。
備考 6. RL78 デバイス・ファミリでこの機能を使用する場合、[関数のインライン展開を行う]を設定してください。詳しくは 3.2.1.6 ロード・モジュールの生成を参照していください。

3.2.4 Embedded Targetの再実行

Embedded Target の再実行を行う場合は、以下の方法があります。それぞれの手順で実行してください。

- ecpils_buildコマンドによる再実行（Simulink®モデルを更新した場合や、再度コード生成を行う場合）
  (1) 前回実行した Simulink®モデルである<モデル名>_ecpils.slx ウィンドウと CS+/e² studio を終了してください。
  (2) 前回の実行で生成された slprj フォルダを削除してください。
  (3) MATLAB®コマンド・ウィンドウで ecpils_build コマンドを実行してください。
  (4) Simulink®ウィンドウからシミュレーションを実行してください。

- PILシミュレーションのみの再実行（Simulink®モデルの更新がなく、コード生成処理を省略する場合）
  (1) 前回実行した Simulink®モデルである<モデル名>_ecpils.slx ウィンドウと CS+/e² studio を終了してください。
  (2) <モデル名>_ecpils.slx ウィンドウを起動しシミュレーションを開始してください。
  (3) CS+/e² studio が起動します。コンフィグレーション・ダイアログの[Embedded Target Options]の[IDE Mode]に[Create Project]または[Open Project]が設定されている場合は、ダウンロードを行ってください。
  (4) CS+/e² studio ダウンロード完了後、自動で PIL シミュレーションの実行が開始されます。

3.2.5 PILシミュレーション後のクリーンアップ

Embedded Target が生成したフォルダ、ファイルをクリーンアップするのは以下の方法です。

- 手動または、"ecpils_cleanup"コマンドを使用した自動で以下のフォルダとファイルを削除します。
  — フォルダ: +ecpils、 < モデル名 >_ecpils、 slprj
  — モデル: < モデル名 >_ecpils.slx

3.3 コード生成対象が参照先モデルのPILシミュレーションの実行

以下に、コード生成対象が参照先モデルの場合の、PIL シミュレーションを行う際に必要となる検証環境の作成方法を示します。

3.3.1 検証環境の作成

以下に、PIL シミュレーションを行う際に必要となる検証環境の作成方法を示します。この説明では提供のサンプル・モデル DataTypesReference.slx と DataTypesRef.slx を使用します。

3.3.1.1 RAファミリSecure Bundleを使用したNon Secureのデバッグ設定の準備

“3.2.1.1RA ファミリ Secure Bundle を使用した Non Secure のデバッグ設定の準備”を参照してください。

3.3.1.2 参照先モデルを使用したモデルの準備

サンプル・モデル DataTypesReference.slx と参照先モデル DataTypesRef.slx を使用します。DataTypesReference.slx は DataTypesRef.slx を参照します。DataTypesRef.slx は直接使用されません。

3.3.1.3 コンフィギュレーション パラメーターの設定

コード生成のオプションを確認/設定します。DataTypesReference.slx と DataTypesRef.slx の組み合わせで同じ設定を行う必要があります。

1. コンフィギュレーション パラメーター ダイアログのオープン
   DataTypesReference ウィンドウの[モデル化]メニューから[モデル設定]を選択し、コンフィギュレーション パラメーター ダイアログをオープンします。

2. [ハードウェア実行]オプションの設定
   [選択]エリアの[ハードウェア実行]を選択し、以下のように各項目の設定を行い、[適用]ボタンをクリックします。

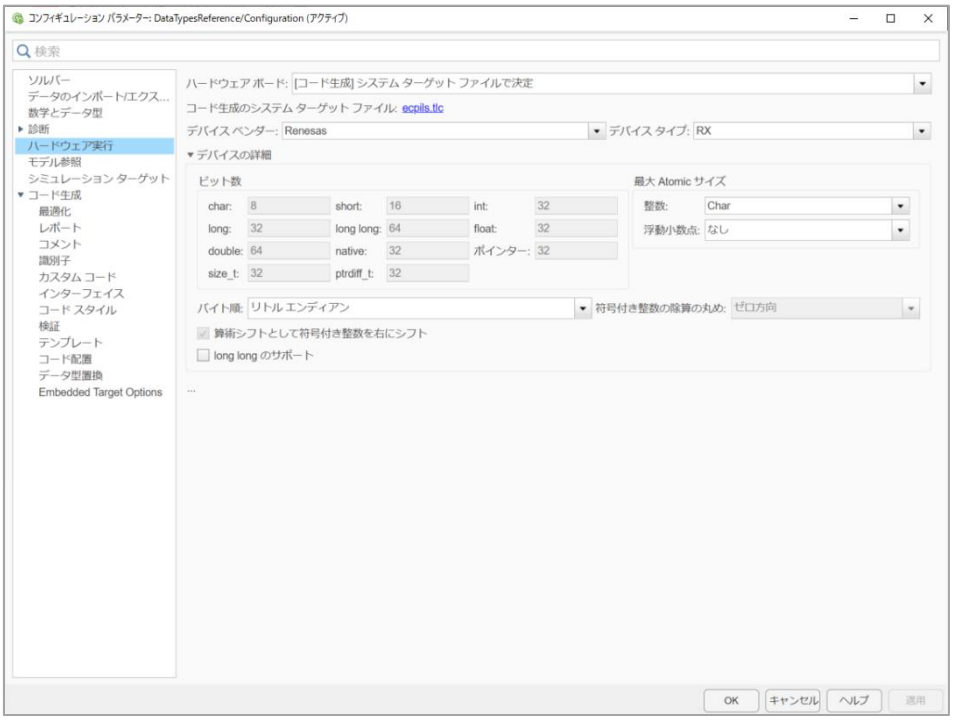

図 3-18 [ハードウェア実行]オプション

備考 1. [Embedded Target options]パネルの[Device Family]値を設定することによって、[ハードウェア実行]パネルの[デバイスタイプ]の値を変更することができます。[Embedded Target Options]の“OK”または“適用”ボタンを押してください。

備考 2. [Device Family]値の設定は、[IDE Mode]値が[Embedded Target options]パネルの[Create Project]の場合に使用可能となります。

3. [モデル参照]オプションの設定

[選択]エリアの[モデル参照]を選択後、[リビルド]を設定し、[適用]ボタンをクリックします。

[常に行う]に設定した場合は、Simulink®モデルの変更に関わらずコード生成が行われますが、[任意の変更を検出]、または、[既知の依存関係で任意の変更が検出された場合]に設定した場合は、Simulink®モデルの変更が検出されなかった場合には、コード生成の処理が省略されます。なお、[行わない]は設定しないでください。

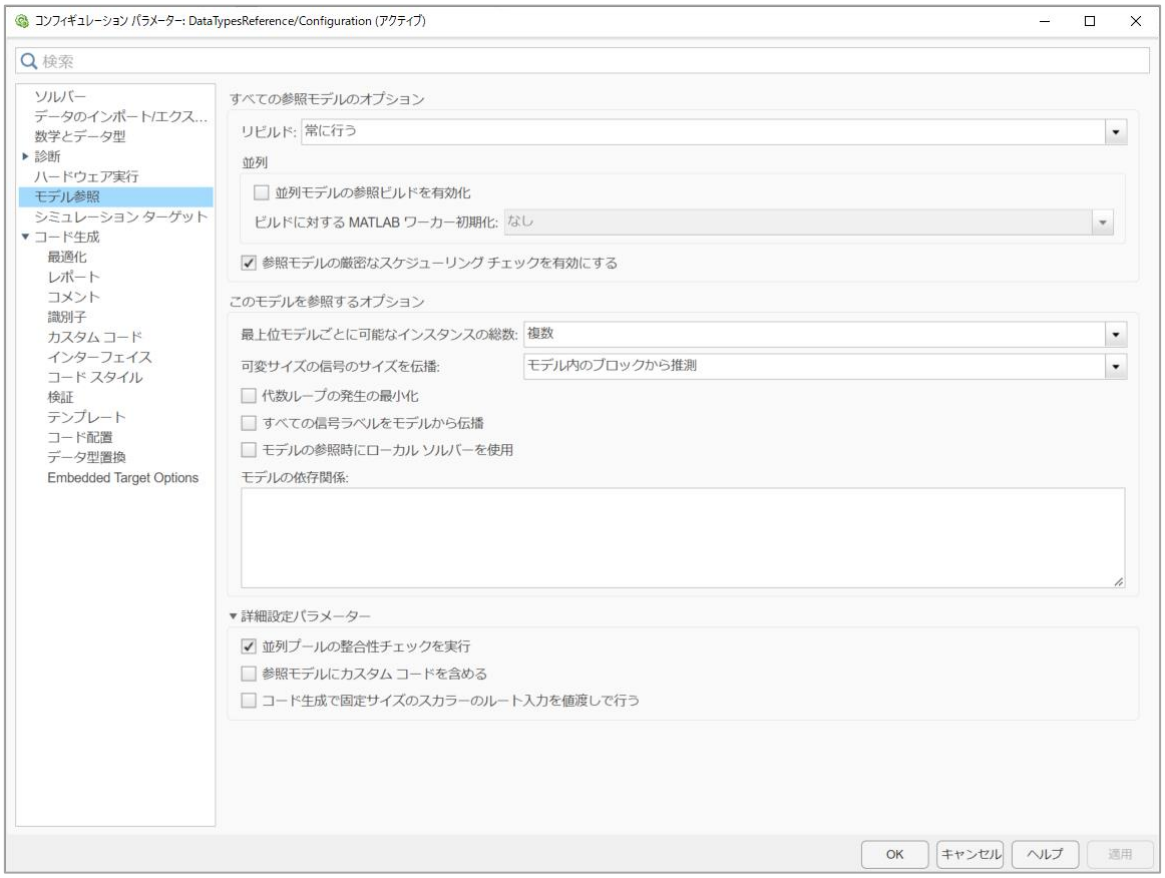

図 3-19 [モデル参照]オプション

4. [コード生成]オプションの設定
   [選択]エリアの[コード生成]を選択後、下図と同様の設定を行い、[適用]ボタンをクリックします。

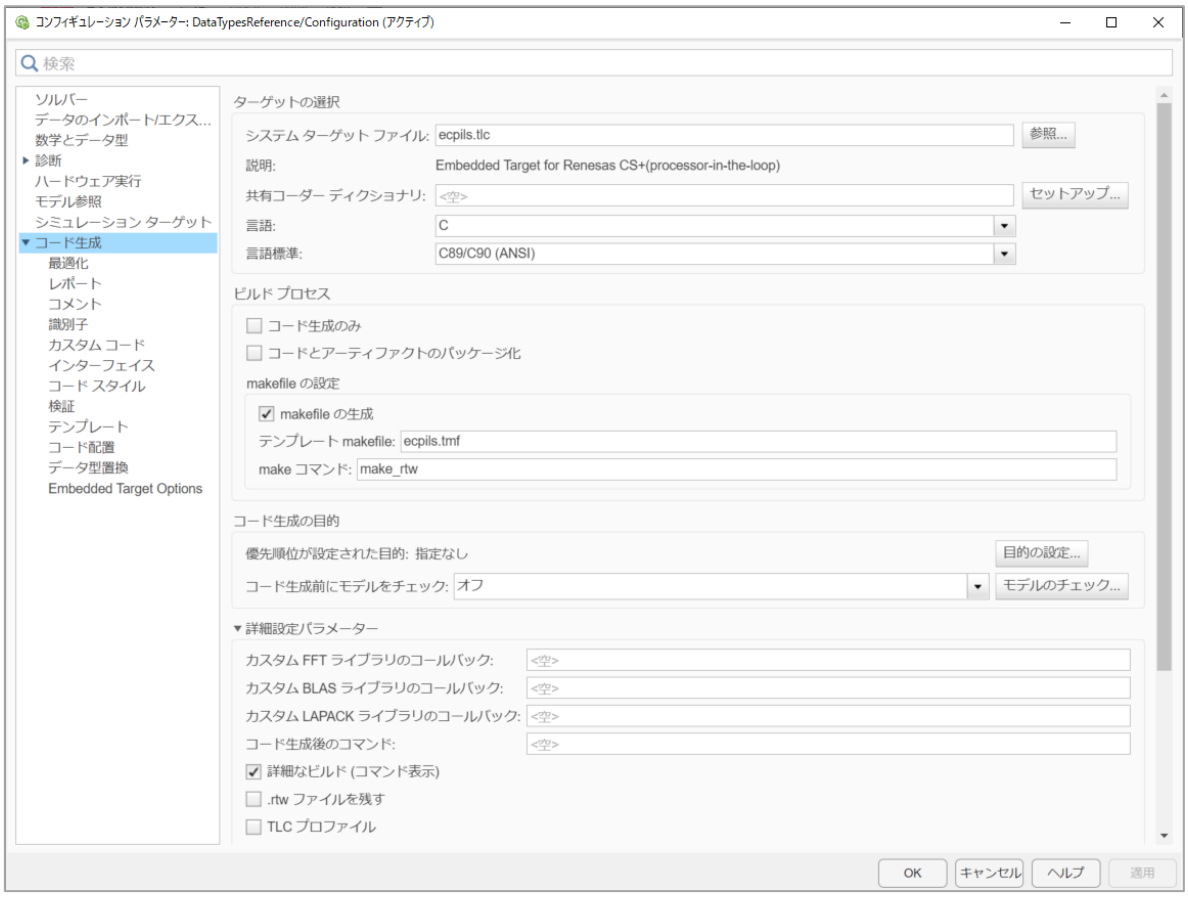

図 3-20 [コード生成]オプション

備考　テンプレート**makefile(ecpils.tmf)**は選択した**MEX**コンパイラ**(>> mex -setup)**によって上書きされます。

5. [検証]オプションの設定

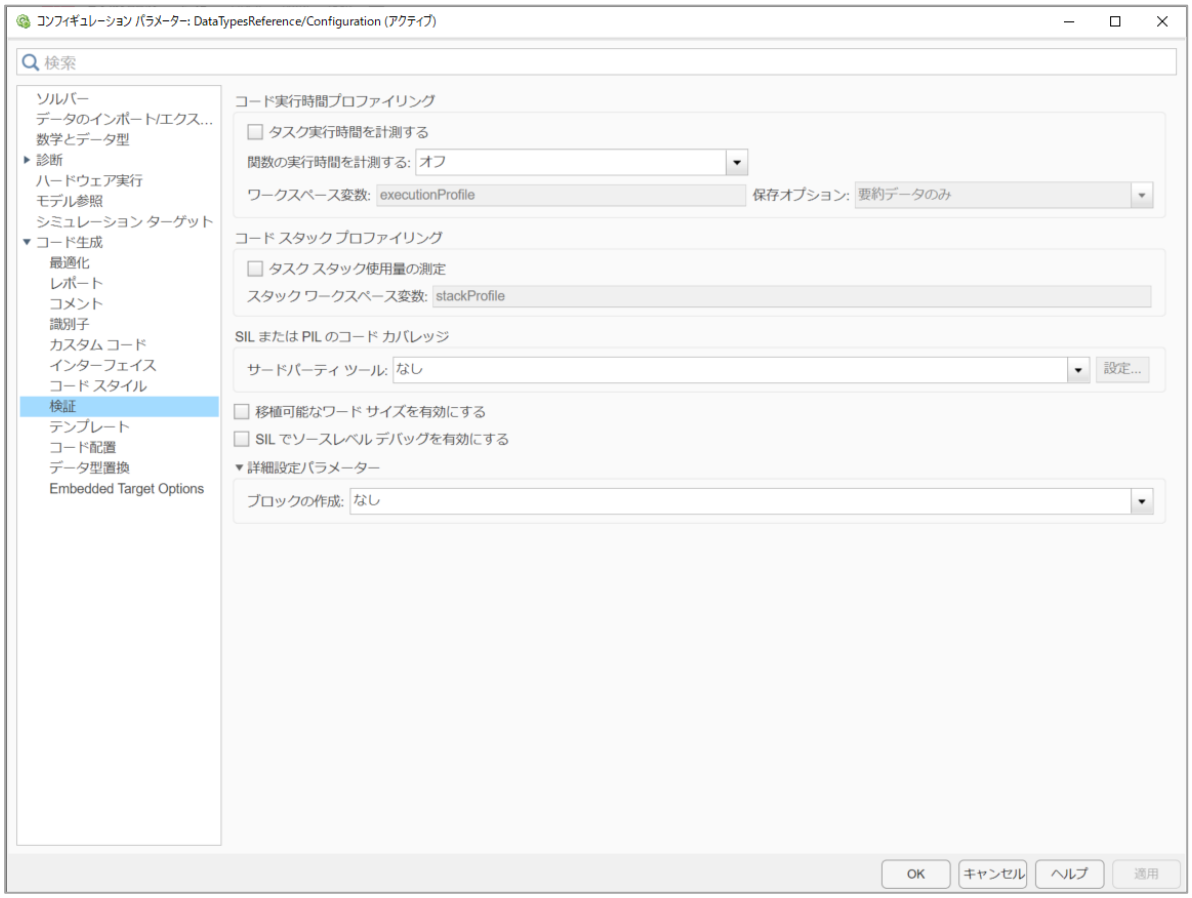

図 3-21 [検証]オプション

6. [Embedded Target Options]オプションの設定
   サブシステムブロック使用時と同じ設定を使用する。 “3.2.1.3 コンフィギュレーション パラメーターの設定” / ”5. [Embedded Target Options]オプションの設定”を参照のこと。

7. コンフィギュレーション パラメーター・ダイアログのクローズ
   設定内容を確認後、[OK]ボタンをクリックします。

8. DataTypesRef モデルのコンフィギュレーション パラメーターの設定
   コード生成対象となる Model ブロックをダブルクリックし、DataTypesRef ウィンドウを開きます。
   DataTypesRef ウィンドウの[ツール]メニューから[コード生成] - [オプション]を選択し、コンフィギュレーション パラメーター ダイアログをオープンし、(2)から(6)で行った設定を同様に行います。

3.3.1.4 PILモードの指定

コード生成対象となるモデルの PIL モードを指定します。

1. ブロック パラメーター ダイアログのオープン
   コード生成対象となる Model ブロックを選択し、右クリックで[ブロックパラメーター(Model Reference)]を選択し、ブロック パラメーター ダイアログをオープンします。

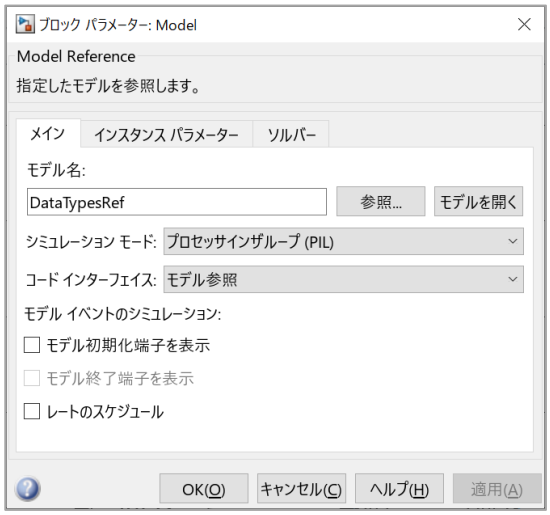

図 3-22 ブロック パラメーター ダイアログ

2. PILの選択
   シミュレーションモードにプロセッサインザループ(PIL)を選択します。

3.3.1.5 PILシミュレーションの実行

DataTypesReference ウィンドウの情報が PIL シミュレーションモデルの情報に変更されていることを確認した後、[シミュレーション]メニューから[実行]を選択し、PIL シミュレーションを開始します。PIL シミュレーションの準備として、コードの生成と CS+/e^2 studio の起動が行われます。

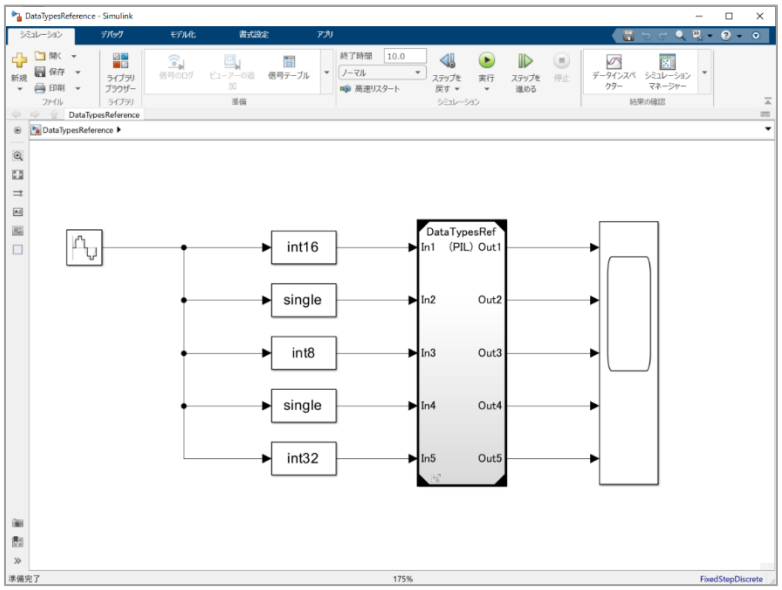

図 3-23 PILシミュレーションの実行

備考 本操作を実行後、コード生成を行った場合、CS+/e2 studio の起動が行われます。また、既存の実行時間計測結果保管ファイルは削除されます。

3.3.1.6 ロード・モジュールの生成

サブシステムブロックの場合と同じ手順を行います。“3.2.1.6 ロード･モジュールの生成”を参照してください。

3.3.1.7 ロード・モジュールのダウンロード

CS+/e2 studio の起動が行われます。コンフィグレーション・ダイアログの Embedded Target Options において、IDE Mode が“Create Project”モード、および、“Open Project”モードに設定されている場合は、以下の手順に従って、ロード・モジュールのビルドとダウンロードを行ってください。“Open Project & LM Download”モード、および、“Template Project & LM Download”モードに設定されている場合は、ロード・モジュールのビルドと、デバッガへのダウンロードが自動的に行われます。ダウンロードが完了しましたら、 PIL シミュレーションを開始します。

- デバッグツールの設定方法
  [IDE に CS+、デバイスに RX または RL78、エミュレータに E1/E2/E2 Lite/E20 (Jtag/Serial)/EZ Emulator または COM Port を使用の場合]
  CS+プロジェクト・ツリーで[デバッグ・ツール]を選択し、プロパティ設定の[デバッグ・ツール設定]-[実行中のメモリ・アクセス]-[実行中にアクセスする]（[実行を一瞬停止してアクセスする]の場合あり）を[はい]に設定してください。

  [IDE に e2 studio、デバイスに RX、エミュレータに E1/E2/E2 Lite/E20 (Jtag/Serial)/EZ Emulator、ビッグエンディアンを使用の場合]
  (1) [プロジェクト・エクスプローラ]ペインの“Tutorial”プロジェクトをクリックしてフォーカスを移動します。
  (2) [実行] →[デバッグの構成…]メニューもしくは アイコンのドロップダウン →[デバッグの構成…]メニューから[デバッグ構成]ウインドウを開きます。
  (3) [デバッグ構成]ウインドウで[Renesas GDB Hardware Debugging] □ [Tutorial HardwareDebug]に移動し、[Debugger]タブに切り替えます。
  [Debugger]タブ下の[デバッグ・ツール設定]サブ・タブに移動し、[メモリ] – [エンディアン]プロパティの値を“ビッグ・エンディアン”に設定します。

- ロード・モジュールのダウンロード方法
  [IDE に CS+を使用の場合]
  CS+のプロパティ・パネルにおいてデバッグ・ツール（E1/E2/E2 Lite/E20(JTAG/シリアル)/EZ emulator、COM Port、シミュレータ）のオプション設定を行ったのち、CS+の［デバッグ］メニューから［デバッグ・ツールへダウンロード］を選択することによりロード・モジュールのダウンロードが行われます。

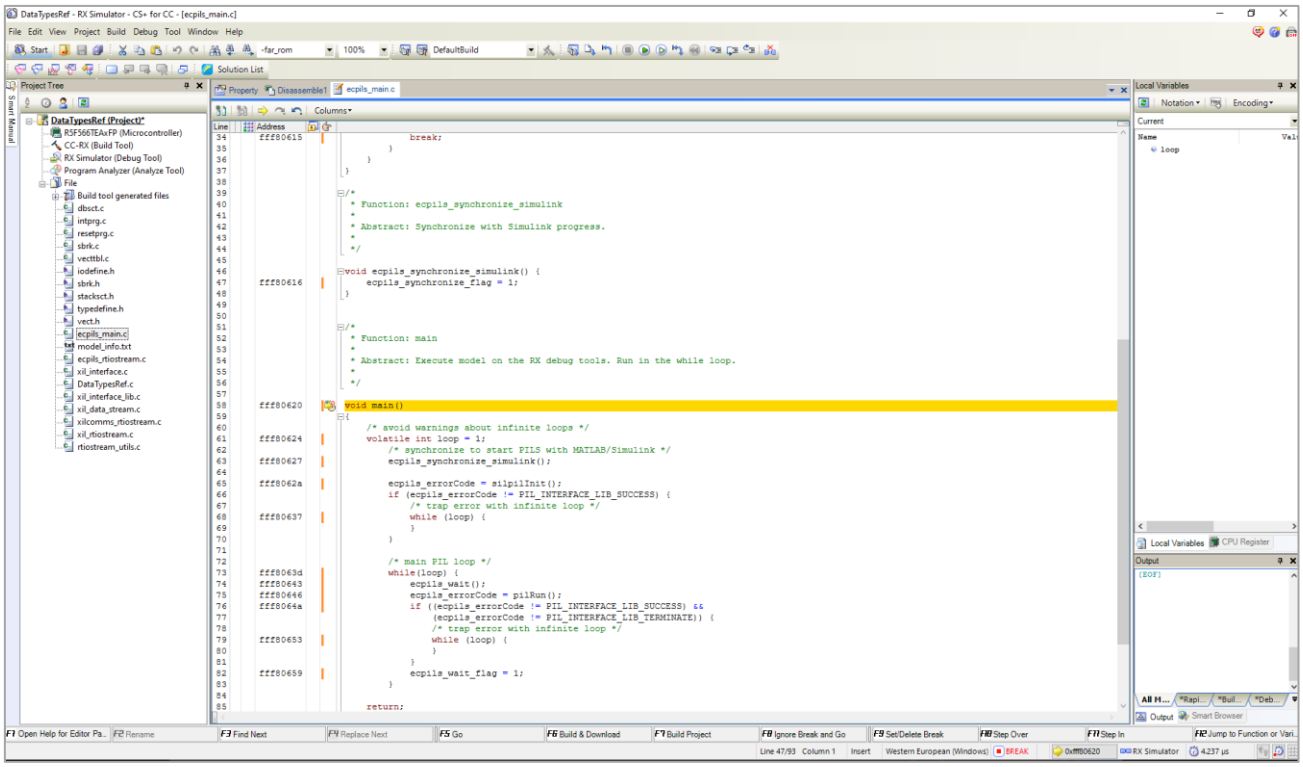

図 3-24 デバッガへのダウンロードが完了した状態のCS+

[IDE に e² studio を使用の場合]

[プロジェクト・エクスプローラ]パネル上で対象プロジェクトをクリックしフォーカスを移動する。その後、 アイコンをクリックしデバッガ・セッションの開始とロードモジュールのダウンロードを行う。

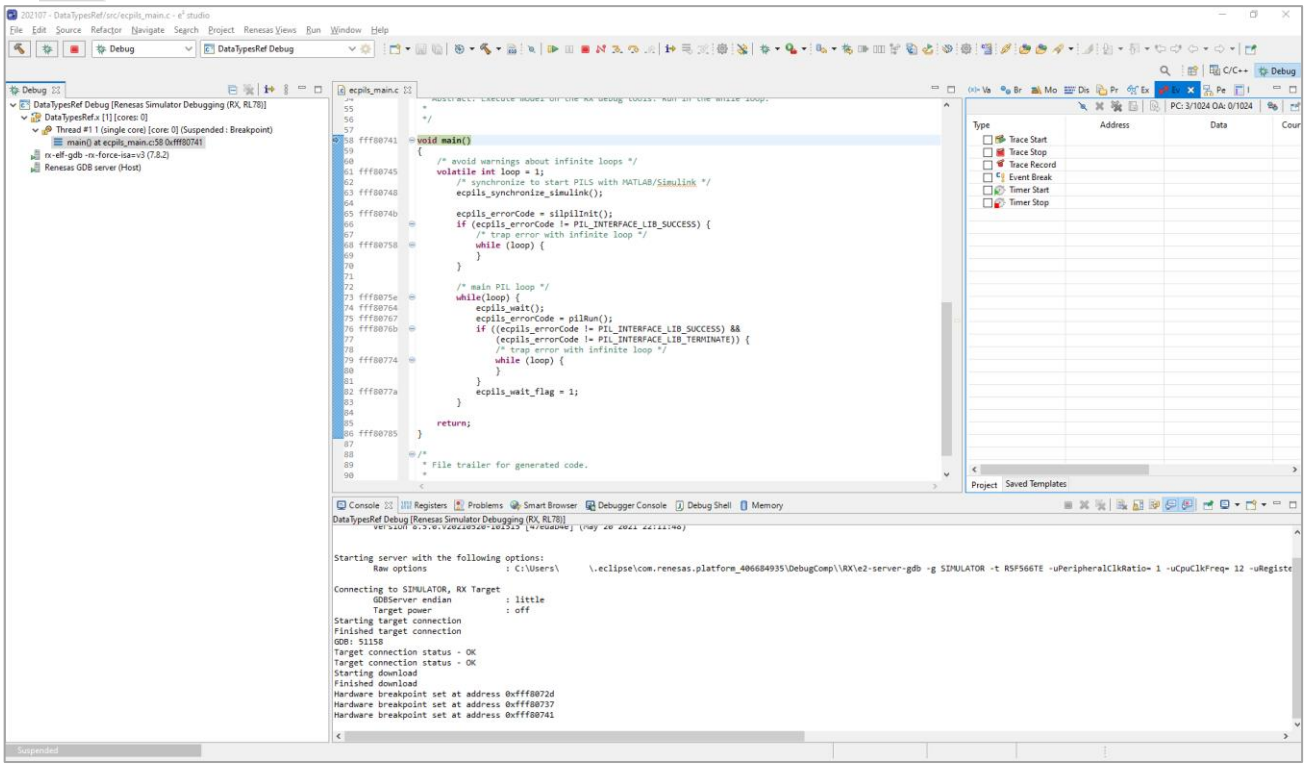

図 3-25 デバッガへのダウンロードが完了した状態のe² studio

Embedded Target がダウンロード完了を確認するまでは、確認ダイアログが表示されます。確認ダイアログは、ダウンロード完了後に自動的に閉じます。ダウンロードを中止して一連の動作を中断したい場合は、確認ダイアログの OK ボタンを押下してください。

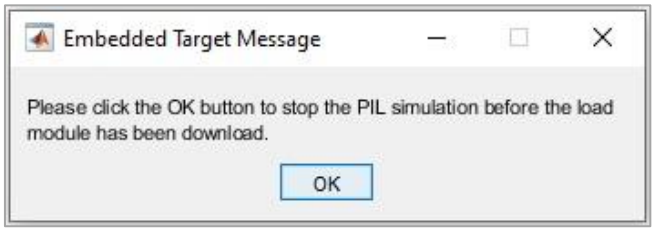

図 3-26 確認ダイアログ

3.3.2 PILシミュレーション中の生成コードのデバッグ

このセクションでは PIL シミュレーション中にモデルから生成したコードをデバッグする方法について説明します。
以下の手順で CS+/e^2 studio のデバッグツールが提供するすべての機能を使用することができます。

- Step 1: モデルを開き、コード生成対象のサブシステムを選択して[コンフィギュレーション パラメーター]ウィンドウを表示します。
- Step 2: 必要な条件を設定し(参照先モデル、参照モデル両方の設定を行います)、[Debug Generated Code during PIL Simulation] チェックボックスをチェックします。
- Step 3: モデルを保存し、Simulink®モデルの[実行]ボタンを押下します。
- Step 4: CS+/e2 studioプロジェクトをビルド＆ダウンロードします。 “ecpils_main.c”の関数*main()*と*ecpils_synchronize_simulink®()*に自動的にブレーク・ポイントが設定されます。
- Step 5: デバッグを開始します。
  - ステップ実行
  - コードブロック、関数へのステップイン
  - 停止
  - その他

このモードを有効にするには、 コンフィギュレーション パラメーターの設定時に[Debug Generated Code during PIL Simulation] チェックボックスをチェックしてください。下図はその例です。

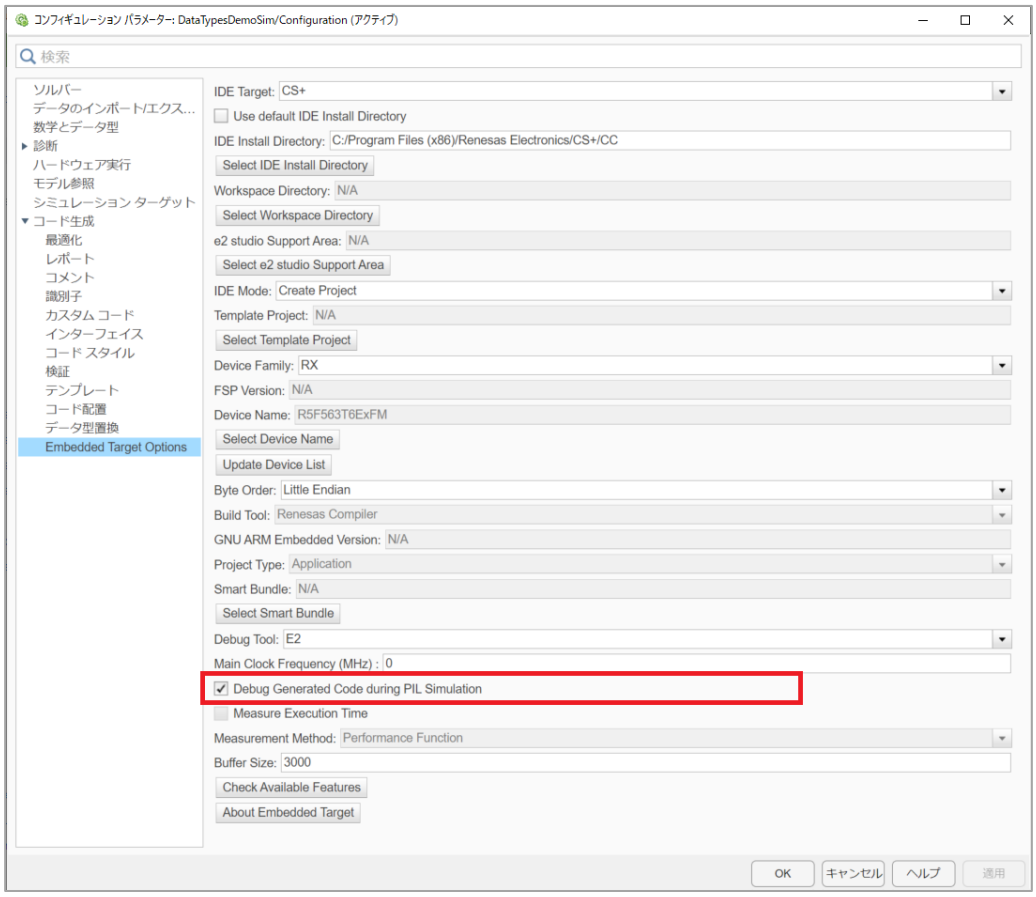

図 3-27 PILシミュレーション中の生成コードデバッグの有効化

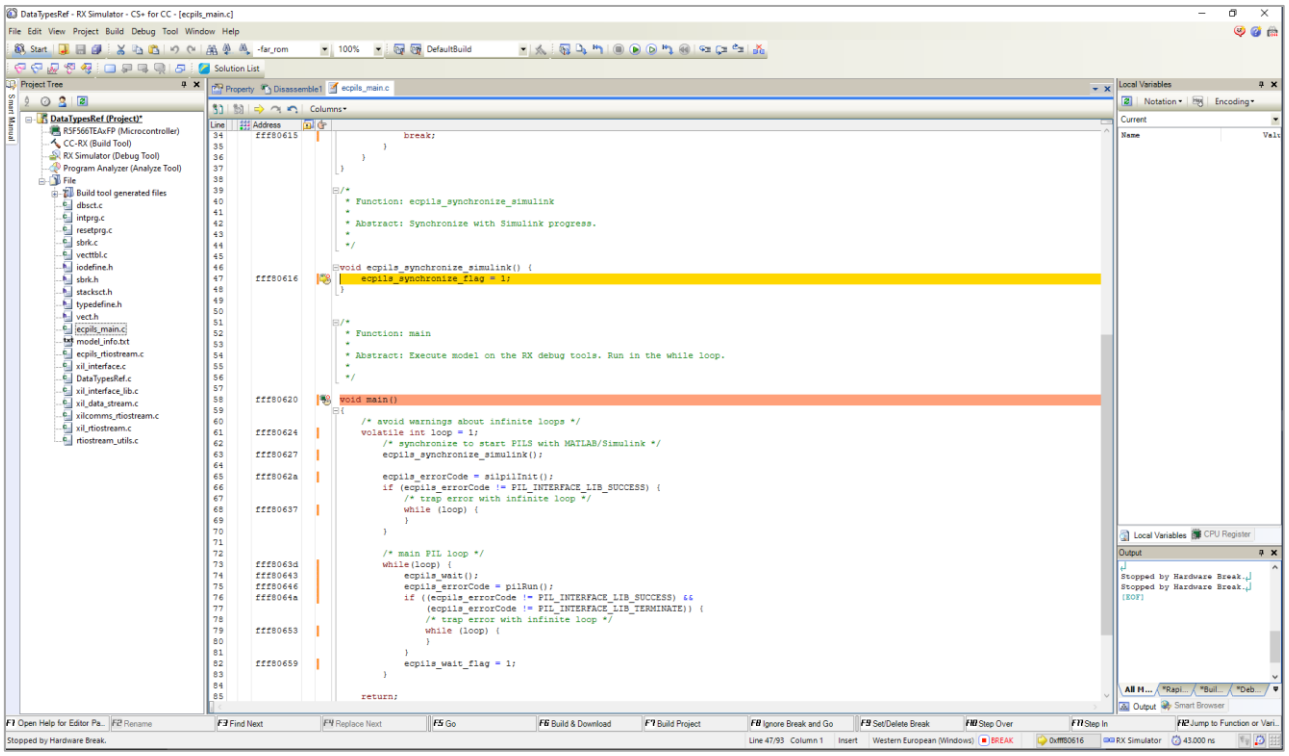

図 3-28 ecpils_main.cに自動的に設定される2つのブレーク・ポイント(CS+)

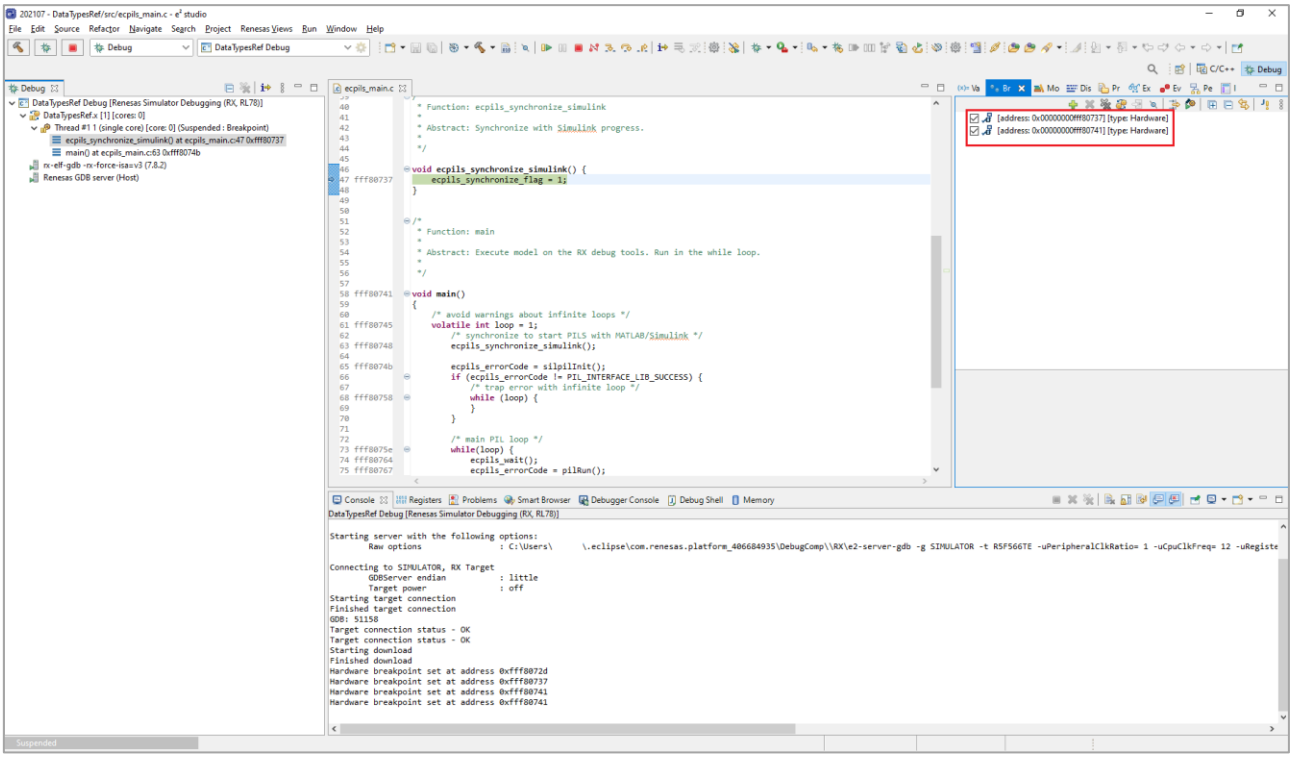

図 3-29 ecpils_main.cに自動的に設定される2つのブレーク・ポイント(e² studio)

備考 1. この機能を使用する場合、実行時間は計測できません。[Measure Execution Time] チェックボックスは無効になります。

備考 2. デバッグ中のステップ実行のタイムアウト時間は最大 24 時間です。タイムアウトを検出すると PIL シミュレーションプロセスは停止します。

備考 3. ターゲット・プログラム(CS+/e^2 studio)が停止した場合、Simulink®の画面は固まります。Simulink®上の GUI から一時停止や停止はできません。これは MATLAB®/Simulink®の制限となります。PIL シミュレーションを停止するには、デバッグ中に設定したブレーク・ポイントを削除し、プログラムが終了するまで実行してください。

備考 4. デバッグ中にデバッグツールから切断したり、CS+/e^2 studio のプロセスを終了したりしないでください。予期しない動作の原因と成ります。

備考 5. Embedded Target で生成したコードの変更や、PC の値の変更他、PIL シミュレーションのワークフローを変えるような変更を CS+/e^2 studio 側で行わないでください。これらの操作は PIL シミュレーションプロセスにとって不正でエラーの原因となります。

備考 6. RL78 デバイス・ファミリでこの機能を使用する場合、[関数のインライン展開を行う]を設定してください。詳しくは 3.2.1.6 ロード・モジュールの生成を参照してください。

3.3.3 Embedded Targetの再実行

前回実行した Simulink®モデルと CS+/e^2 studio を終了します。その後、Simulink®モデルを起動して、シミュレーションを開始してください。
コンフィグレーション パラメーター ダイアログの[モデル参照]の[リビルド]の設定によって動作が異なります。[常に行う]が設定されている場合は、必ずコード生成が行われます。[任意の変更を検出]または[既知の依存関係で任意の変更が検出された場合]が設定されている場合は、Simulink®モデルに変更がないと、コード生成の処理が省略されます。

3.3.4 シミュレーション後のクリーンアップ

Embedded Target のワークスペースをクリーンアップする方法は以下です。

- 手動または、"ecpils_cleanup"コマンドを使用した自動で以下のフォルダとファイルを削除します。
  - フォルダ: +ecpils、 slprj

3.4 コード生成対象が最上位モデルのPILシミュレーションの実行

以下に、コード生成対象が最上位モデルの場合の、PIL シミュレーションを行う際に必要となる検証環境の作成方法を示します。

3.4.1 検証環境の作成

以下に、PIL シミュレーションを行う際に必要となる検証環境の作成方法を示します。この説明では提供のサンプル・モデル DataTypesRef_Top.slx を使用します。

3.4.1.1 RAファミリSecure Bundleを使用したNon Secureのデバッグ設定の準備

“3.2.1.1RA ファミリ Secure Bundle を使用した Non Secure のデバッグ設定の準備”を参照してください。

3.4.1.2 最上位モデルを使用したモデルの準備

サンプル・モデル DataTypes_Top.slx を使用します。

3.4.1.3 コンフィギュレーション パラメーターの設定

サブシステムブロック使用時と同じ設定を使用します。3.2.1.3 コンフィギュレーション パラメーターの設定を参照してください。

3.4.1.4 PILモードの指定

コード生成対象となるモデルの PIL モードを指定します。

1. プロセッサインザループ(PIL)の選択
   モデル・ウィンドウで[アプリ] – [SIL/PIL マネージャー] を選択し、[SIL/PIL]メニューの[SIL/PIL モード]で”Processor-in-the-Loop (PIL)”を選択します。

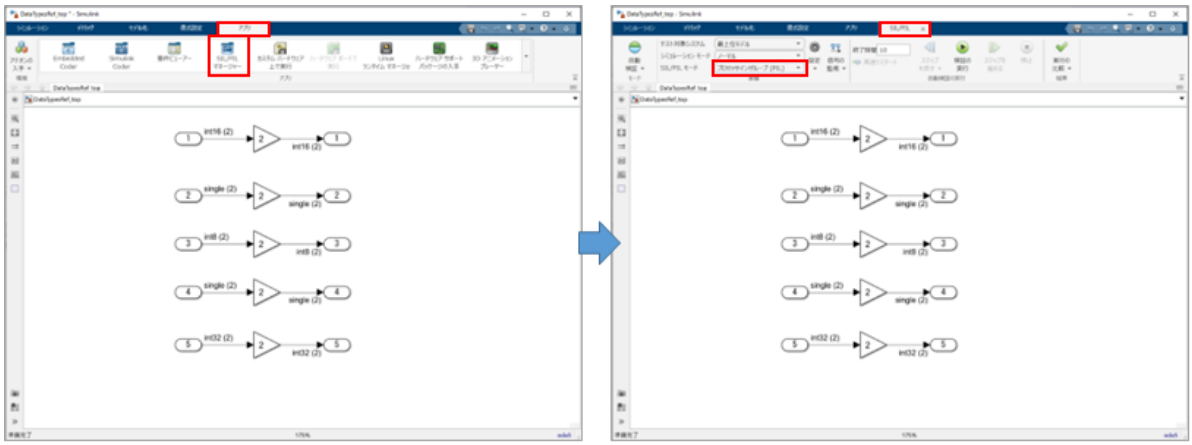

図 3-30 シミュレーションメニュー(MATLAB® R2021a以上)

2. 引数値の設定
   モデルの引数値を設定します。ワークスペース変数の設定を行います。サンプル・モデルではワークスペース変数の設定を行う.m ファイル(DataTypesRef_Top_input.m)を用意しています。

3.4.1.5 PILシミュレーションの実行

ワークスペース変数の設定が行われていることを確認後、[SIL/PIL]メニューから[実行]を選択し、PIL シミュレーションを開始します。PIL シミュレーションの準備として、コードの生成と CS+/e^2 studio の起動が行われます。

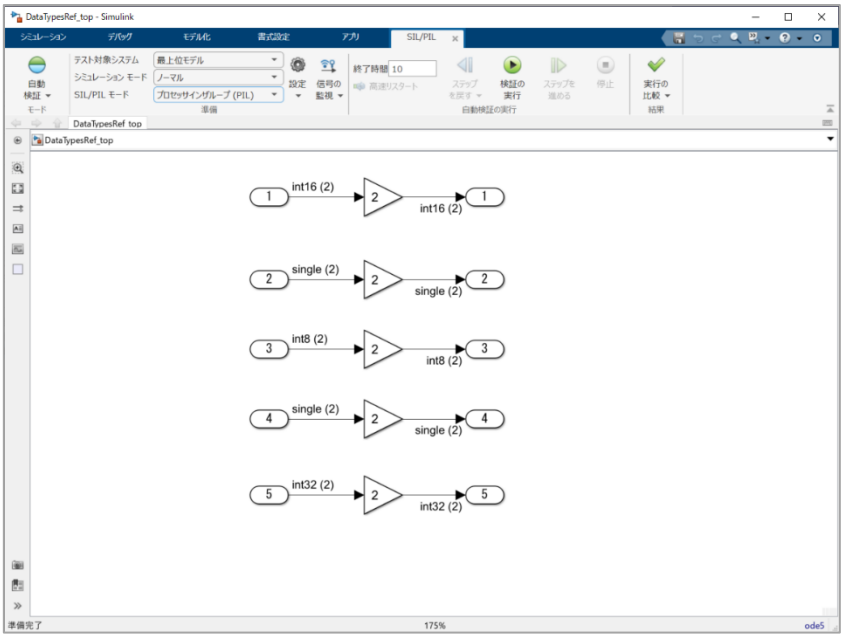

図 3-31 PILシミュレーションの実行(MATLAB® R2021a以上)

備考 本操作を実行後、コード生成を行った場合、CS+/e^2 studio の起動が行われます。また、既存の実行時間計測結果保管ファイルは削除されます。

3.4.1.6 ロード・モジュールの生成

サブシステムブロックを使用時と同じ手順で行います。3.2.1.6 ロード・モジュールの生成を参照してください。

3.4.1.7 ロード・モジュールのダウンロード

参照先モデル使用時と同じ手順で行います。3.3.1.7 ロード･モジュールのダウンロードを参照してください。

3.4.2 PILシミュレーション中の生成コードのデバッグ

参照先モデル使用時と同じ手順で行います。3.3.2 PIL シミュレーション中の生成コードのデバッグを参照してください。

3.4.3 Embedded Targetの再実行

前回実行した Simulink®モデルと CS+/e^2 studio を終了します。その後、Simulink®モデルを起動して、シミュレーションを開始してください。
Simulink®モデルに変更がないと、コード生成の処理が省略されます。Simulink®モデルに変更がある場合、コード生成処理は省略されません。

3.4.4 PILシミュレーション後のクリーンアップ

Embedded Target のワークスペースをクリーンアップする方法は以下です。

- 手動または、"ecpils_cleanup"コマンドを使用した自動で以下のフォルダとファイルを削除します。
  - フォルダ: +ecpils、 < モデル名 >_ecpils、 slprj

3.5 コード生成対象となるアルゴリズムの対象デバイスでの検証

PIL シミュレーションを実行したことによって得られた情報（Simulink®モデルと実行結果から生成されたロード・モジュールの実行時間）を使用して、アルゴリズムを検証することができます。
コンフィギュレーション パラメーター ダイアログボックスの[Embedded Target Options]にある[Measure Execution Time]チェックボックスを選んだ場合にのみ、実行時間の測定が有効になります。
PIL シミュレーションモードや時間計測方法によって、実行時間の計測手順は異なる可能性があります。詳細については、PIL シミュレーションモードと時間計測方法に応じて、以下を参照してください。

備考 1. “Performance Function”を使った時間測定はサポートされているすべて MCU ファミリのシングルコア MCU で使用出来ます。

備考 2. “Performance Function”を使った時間測定は実行時間の供給元として CS+/e^2 studio タイマ機能を使用します。そのため、デバッグツールによる実行時間測定前に CS+/e^2 studio のタイマ機能を有効にする必要があります。タイマ機能を有効にするには CS+/e^2 studio のステータスバーの右にある[Timer]ボタンを押下します。

備考 3. 実行時間計測の結果は、ナノ秒単位で示されます。

以下で “Performance Function”を使った時間計測法を示します。これにより、PIL シミュレーションプロセス全体の合計実行時間が提供されます。これはシングルコア PIL シミュレーションでのみ使用できます。
この機能を有効にするためには、[Measure Execution Time]チェックボックスをチェックし、[Measurement Method]に“Performance Function”を選択してください。
コード生成対象によって以下のファイルに計測結果が保存されます。

1. コード生成対象がサブシステムブロックの場合

コード生成対象がサブシステムブロックの場合の時間計測結果保存ファイル
<*モデル・ファイルを含むフォルダ*> ¥<*サブシステム名*>_ecpils¥ <*サブシステム名*>.txt

2. コード生成対象が参照先モデルの場合

Modelコード生成対象が参照先モデルの場合の時間計測結果保存ファイル
<*モデル・ファイルを含むフォルダ*> ¥slprj¥ecpils¥<*参照先モデル名*>¥<*参照先モデル名*>.txt

3. コード生成対象が最上位モデルの場合

コード生成対象が最上位モデルの場合の時間計測結果保存ファイル
<*モデル・ファイルを含むフォルダ*> ¥<*最上位モデル名*>_ecpils¥<*最上位モデル名*>.txt

4. 計測結果は以下のフォーマットで保存されます。

通常時	Total: <*総実行時間*>ns, Pass Count: <*パスカウント*>, Average: <*平均実行時間*>ns, Max: <*最大実行時間*>ns, Min: <*最小実行時間*>ns
ステップ毎の時間計測でオーバーフロー発生時	Total: Overflow, Pass Count: <*パスカウント*>, Average: N/A, Max: N/A, Min: <*最小実行時間*>ns
総実行時間計算でオーバーフロー発生時	Total: Overflow, Pass Count: <*パスカウント*>, Average: N/A, Max: <*最大実行時間*>ns, Min: <*最小実行時間*>ns

4. エラー・メッセージ

本章では、Embedded Target によって出力するエラー・メッセージについて説明します。

4.1 概要

エラー・メッセージは、コンフィギュレーション パラメーター ダイアログで [Embedded Target Options] オプションを設定している際、および Embedded Target が PIL シミュレーション処理を実行中に“ユーザに通知すべき情報”を検出した際に生成され、通知されます。

備考 Embedded Target が出力するエラー・メッセージは、CS+/e^2 studio と連携していません。このため、Embedded Target がエラー・メッセージを出力した際、F1 キーを押下しても該当ヘルプは起動しません

4.2 コンフィギュレーション パラメーター ダイアログのエラー

以下に、コンフィギュレーション パラメーター ダイアログで検出されるエラー・メッセージの一覧を示します。
なお、本エラー・メッセージは、Embedded Target エラー・ダイアログに出力します。

表 4-1 コンフィギュレーション パラメーター ダイアログにおけるエラー・メッセージ

[メッセージ]	E0101 The <*Embedded Target option*> is invalid. Please check the entered value.
[説明]	項目の値が正しくない値を指定した場合に表示されるエラーです。
[対処方法]	エラーが表示された項目に正しい値を指定してください。
[メッセージ]	E0102 This button cannot currently be pressed. Please change the value of <*Embedded Target option*>.
[説明]	エラーメッセージに示されたオプションの値が一致しないためにボタンが押せない場合に表示されます。
[対処方法]	エラーメッセージに示されたオプションの値を変更してください。
[メッセージ]	E0105 <*Embedded Target license*> is not registered.
[説明]	選択したデバイスファミリに対応するライセンスが登録されていない場合に表示されます。
[対処方法]	• サポートデバイスのPILシミュレーションのライセンスをお持ちでない場合は、ルネサスエレクトロニクスから入手してください。 • サポートデバイスのPILシミュレーションのライセンスファイルをすでに入手している場合は、それがEmbedded Targetのインストールに適用できるか確認してください。その機能の有無を確認するため、[Embedded Target Options]パネルの”Check Available Features”で確認を行ってください。

[メッセージ]	E0106 The selected debug tool is not supported. Please choose another debug tool.
[説明]	選択したデバッグツールが指定されたデバイス・ファミリまたはデバイスでサポートされていない場合に表示されるエラーです。
[対処方法]	[Device Family]オプションと[Device Name]オプションで指定したデバイスに対応する別のデバッグツールを選択してください。
[メッセージ]	E0108 No devices are available. Please press [Update Device List] button to update the list of devices corresponding to the selected device family.
[説明]	選択したデバイスファミリのデバイスファイルにデバイスがない場合に表示されます
[対処方法]	現在使用しているIDEを確認し、[Update Device List]ボタンを押下しデバイスファイルを更新してください。
[メッセージ]	E0110 No license is registered. Please register a valid license.
[説明]	ご使用のシステムでライセンスが登録されていないか、期限切れのときに表示されるエラーです。
[対処方法]	有効な Embedded Target ライセンスをシステムに登録してください。
[メッセージ]	E0115 File <target file> has been changed.
[説明]	パッケージ中の ecpils ファイルが変更されたか、カレント・ワークスペース中の同名のファイルが存在しています。
[対処方法]	• Embedded Targetを再インストールしてください。 • カレント・ワークスペースにエラー・メッセージ中の<target file>と同名のファイルが存在するかどうか確認し、存在する場合はワークスペース側のファイル名を変更してください。

4.3 ビルド時のエラー

以下に、ビルドで検出されるエラー・メッセージの一覧を示します。
なお、本エラー・メッセージは、Embedded Target エラー・ダイアログに出力します。

表 4-2 ビルドにおけるエラー・メッセージ

[メッセージ]	E0201 Please exit <IDE Target>, which has started.
[説明]	CS+または e2 studio が既に起動されています。
[対処方法]	**(1)** 起動しているCS+/e^2 studioを終了してから、やり直してください。 **(2)** Windows®タスクマネージャーでCS+/e^2 studioのプロセスを終了してください。 **(3)** CS+のラピッド・スタート機能が使用されていないことを確認してください。
[メッセージ]	E0202 The current mex compiler configuration is not supported.
[説明]	現在の MATLAB®のコンパイラ・ツールチェインに mingw64、 MSVC(Microsoft Visual C)が含まれていません。
[対処方法]	使用する MATLAB®バージョンに対応した MinGW、MSVC をインストールしてください。
[メッセージ]	E0203 The selected template project file does not exist.
[説明]	Embedded Target Options で指定されたテンプレート・プロジェクト・ファイルのアドレスは正しくありません。
[対処方法]	正しいテンプレート・プロジェクトのアドレスを指定してください。
[メッセージ]	E0204 The current working directory does not contain model <ModelName>.
[説明]	コード生成中のビルドでカレント・ディレクトリがプロジェクト・ディレクトリと異なることでエラーが発生しました。よって Embedded Target はコード生成前に警告を発しました。
[対処方法]	カレント・ディレクトリをプロジェクト・ディレクトリに変更してください。
[メッセージ]	E0205 Opening the PIL simulation communications channel was not possible.
[説明]	確認ダイアログの[OK]ボタンのクリックで PIL シミュレーションを停止しようとしたときに表示されるエラーです。
[対処方法]	Embedded Target でビルド中は、PIL シミュレーションを停止するための確認ダイアログの[OK]ボタンのクリックを避けてください。
[メッセージ]	E0206 The GenCodeOnly option is not supported.
[説明]	[Generate code only]チェックボックスがチェックされているときに表示されるエラーです。
[対処方法]	[Generate code only]チェックボックスのチェックを外してください。
[メッセージ]	E0207 The Create Block option is not set to "PIL".
[説明]	コンフィギュレーション ダイアログで[Create Block]に[PIL]が設定されていません。
[対処方法]	対象モデルのコンフィギュレーション ダイアログを開き、[すべてのパラメータ]タブで[Create Block]オプションを検索し、[Create Block]に[PIL]を指定してください。

4.4 CS+/e2 studio起動からダウンロードまでのエラー

以下に、CS+/e2 studio 起動からダウンロードまでの Embedded Target による処理で検出されるエラー・メッセージの一覧を示します。

表 4-3 CS+/e2 studioにおけるエラー・メッセージ

[メッセージ]	E0300 Creating the <IDE Target> project was not possible (project.Create error). [Direct Cause] <エラーの直接原因>
[説明]	CS+/e2 studio プロジェクト・ファイルが生成できませんでした。
[対処方法]	• Embedded Targetがサポートしているe2 studioバージョンであることを確認してください。 • CS+ Pythonプラングインが有効になっていることを確認してください。 • e2 studioでSupport Areaが正しく設定されていることを確認してください。
[メッセージ]	E0302 Adding the source file was not possible (project.File.Add error).
[説明]	CS+プロジェクト・ファイルにソースを登録できませんでした。
[対処方法]	Embedded Target.がサポートしている MATLAB®バージョンであることを確認してください。
[メッセージ]	E0303 Removing the source file was not possible (project.File.Remove error).
[説明]	IDE のプロジェクト・ファイルからソースを削除できませんでした。
[対処方法]	IDE Mode が“Template Project & LM Download”の場合、Embedded Target で作成した IDE プロジェクトが指定されていることを確認してください。
[メッセージ]	E0304 Setting the debug tool was not possible (debugger.DebugTool.Change error).
[説明]	[Embedded Target Options]パネルで設定した対象となるデバッグ・ツールに変更することはできません。
[対処方法]	CS+プロジェクトで使用可能な E1/E2 エミュレータ接続タイプを確認してください。 検証環境を再度生成してください。
[メッセージ]	E0312 Building was not possible (build.All error).
[説明]	ビルド時にエラーが発生しました。
[対処方法]	以下の確認を行い、検証環境の作成からやり直してください。 1. CS+/e2 studio プロパティ設定を見直してください。 2. CS+/e2 studio の出力パネルに表示されているエラー・メッセージを確認してください。 3. デバイスのメモリサイズが小さい場合は、メモリサイズの大きいデバイスの利用を検討してください。
[メッセージ]	E0320 Connecting the debug tool was not possible (debugger.Connect error).
[説明]	デバッグ・ツールへの接続時にエラーが発生しました。
[対処方法]	• CS+/e2 studioプロパティ設定を確認してください。 • E1/E2/E2 Lite/E20 (Jtag/Serial)/EZ emulator/COM Port/J-Linkが正しく接続されているか確認してください。
[メッセージ]	E0321 Downloading the load module was not possible (debugger.Download.LoadModule error).
[説明]	ダウンロード時にエラーが発生しました。
[対処方法]	• CS+/e2 studioプロパティ設定を確認してください。 • CS+/e2 studioビルド時にエラーが発生していないか確認してください。

[メッセージ]	E0322 Setting the timer event was not possible (debugger.Timer.Set error).
[説明]	タイマイベントを設定できない場合、エラーが発生します。
[対処方法]	タイマイベントの数を減らすため、Simulink®モデルのコアを再配置してください。 Simulink®モデルからロード・モジュールを再生成、再実行してください。
[メッセージ]	E0323 Opening the e^2 studio project was not possible (project.Open error).
[説明]	e^2 studio のプロジェクト・ファイルをワークスペースにインポートできませんでした。
[対処方法]	• Embedded Targetがサポートしているe^2 studioバージョンであることを確認してください。 • e^2 studioでSupport Areaが正しく設定されていることを確認してください。

4.5 PILシミュレーション時のエラー

以下に、PIL シミュレーション時のエラーを示します。なお、本エラー・メッセージは、PIL シミュレーション中に MATLAB®/Simulink®のエラー・ダイアログに出力します。

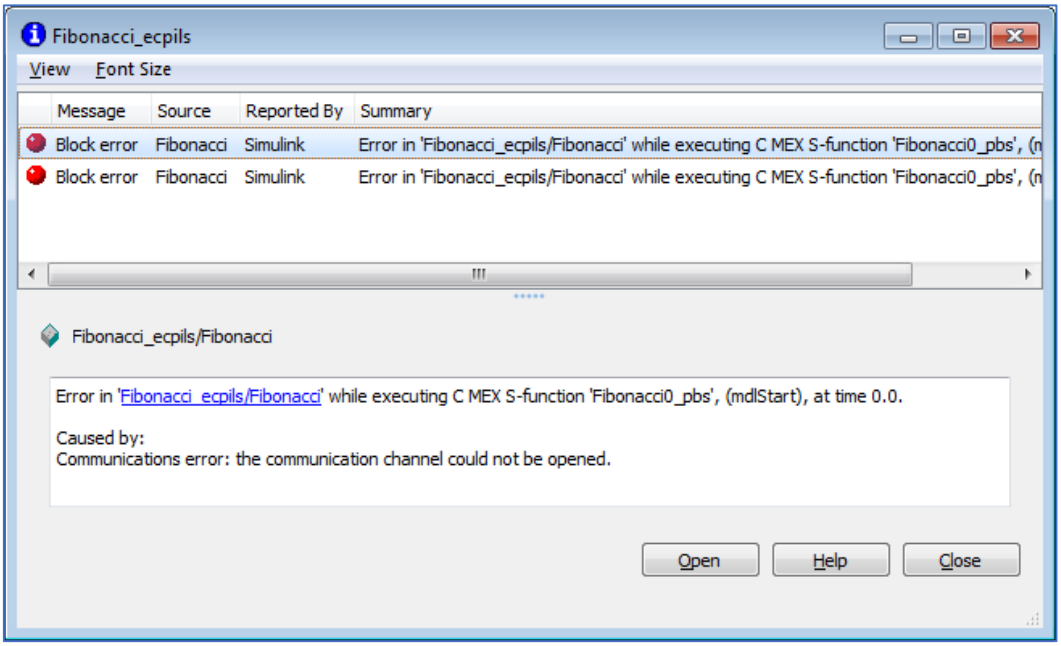

図 4-1 PILシミュレーション時のエラー・ダイアログ中のメッセージ

表 4-4 PILシミュレーション時のエラーの対処

[対処方法]	(1) CS+/e² studio が起動していることを確認します。 (2) CS+/e² studio のデバッグツールが接続できることを確認します。 (3) CS+/e² studio にプログラムがダウンロードされていることを確認します。 (4) 複数の CS+/e² studio が起動していないことを確認します。 (5) CS+/e² studio のラピッド・スタート機能が使用されていないことを確認します。 (6) すべての MATLAB®および CS+/e² studio のプロセスを閉じます。 (7) Windows®タスクマネージャーを使用して CS+/e² studio 関連するプロセスを閉じます。 (7.1) Windows®のタスクバーを右クリックして、タスクマネージャーを起動します。 (7.2) タスクマネージャーで[プロセス]タブを選択します。 (7.3) CubeSuiteW+.exe プロセスや e2studio.exe プロセスが終了していることを確認し、もしプロセスが終了していない場合は選択しプロセスを終了させます。 (8) Embedded Target Options の Buffer size オプションの値を 3000 から別の値に変更する (9) MATLAB®を起動する PILシミュレーションを再実行する

改訂記録	Embedded Target ユーザーズマニュアル　操作編

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Aug.01.24	ー	初版発行

Embedded Target ユーザーズマニュアル
操作編

発行年月日　2024年8月1日　　Rev.1.00

発行　　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）

Embedded Target

RENESAS

ルネサス エレクトロニクス株式会社

R20UT5542JJ0100