User's Manual

Embedded Target

User's Manual: Operation

Target Device
RX Devices Family
RL78 Devices Family
RA Devices Family

Renesas Electronics
www.renesas.com

Rev.1.00 Mar 2025

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WART OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)
   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on
   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state
   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins
   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals
   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin
   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses
   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products
   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

How to Use This Manual

Target Readers

This manual is intended for users who wish to understand the functions of the MATLAB®/Simulink® and design software and hardware application systems.

Purpose

This manual is intended to give users an understanding of the functions of the Model based Development Tool to use for reference in developing the hardware or software of systems using these devices.

Organization

This manual can be broadly divided into the following units.

Chapter 1	GENERAL
Chapter 2	INSTALLATION
Chapter 3	FUNCTIONS
Chapter 4	ERROR MESSAGES

How to Read This Manual

It is assumed that the readers of this manual have general knowledge of electricity, logic circuits, and microcontrollers.

Conventions

Note:	Footnote for item marked with Note in the text
Caution:	Information requiring particular attention
Remark:	Supplementary information
Numeric representation:	Decimal ... XXXX
	Hexadecimal ... XXXXH or 0xXXXX

Licensing

This product uses the IronPython based on the following license.

List of Abbreviations and Acronyms

Abbreviation	**Full Form**
GUI	Graphical User Interface
GDB	Standard GNU Debugger
IDE	Integrated Development Environment
I/O	Input/Output
LM	Load Module
MCU	Microcontroller Unit
PC	Personal Computer
PIL	Processor in the Loop

Table of Contents

1. GENERAL

This chapter provides an overview of the functions of “Embedded Target (Processor in the Loop Simulation System)”.

1.1 Overview

Embedded Target facilitates the verification of algorithms in embedded models by generating a test environment automatically in Processor in the Loop Simulation System (hereafter referred to as PILS).

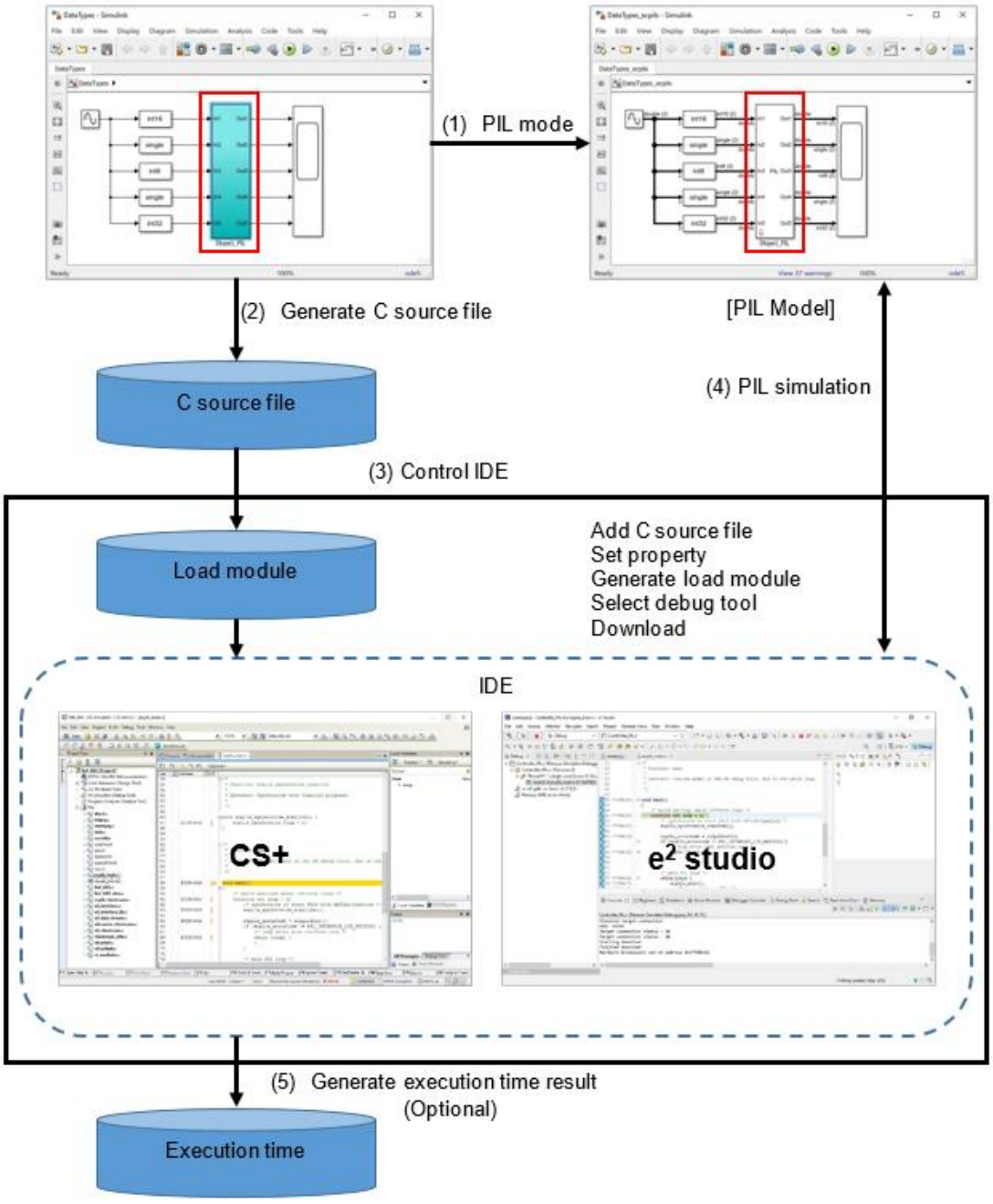

Figure 1-1. Flow of Processing for Generation of Test Environment

Remark Embedded Target executes operations (1) to (5) in the above figure automatically.

1.2 Features

This section lists the features of Embedded Target.

1. Generation of test environment

   The following processing operations, which are necessary to generate a test environment for PIL simulation, can be executed automatically. The processing operations vary depending on the target for code generation: a Subsystem block, a reference model or a top-level model.

   A. Subsystem block
      a. Generation and replacement of a block for PIL linking
      b. Generation of C source files
      c. Start-up of CS+/e^2 studio
         - Registration of C source files
         - Property setting
         - Generation of a load module
         - Connection of a debug tool
         - Download of a load module
      d. PIL simulation is executed manually after generation of a test environment.

   B. Reference model or top-level model
      a. PIL mode specification (Set it beforehand when creating a model)
      b. Generation of C source files
      c. Start-up of CS+/e^2 studio
         - Registration of C source files
         - Property setting
         - Generation of a load module
         - Connection of a debug tool
         - Download of a load module
      d. PIL simulation (Executed automatically after generation of a test environment

2. Algorithm verification

   PIL Simulation, which is sequentially executed in combination with MATLAB®/Simulink® and CS+/e^2 studio, enables the verification of algorithms for the load module generated by embedded models. The load module can be executed on 1 core (hereafter, referred to as Single-Core PIL Simulation) depending on the hardware capability of the target device.

3. Measure the algorithm performance of embedded models

   Embedded Target supplies the execution time measurement (hereafter, referred to as Performance Measurement) of the PIL Simulation by executing the load module, generated by embedded models, on CS+/e^2 studio. The measurement result (automatically saved in file format) provides different execution time information in according to the PIL Simulation mode and the measurement method.

4. Multiple code generation targets (block and model)

   The following code generation targets are supported:
   - Subsystem block
   - Reference model
   - Top-level model

1.3 Operating Environment

Below descriptions are the system requirements for Embedded Target.

1. Hardware environment

Operating system	Microsoft® Windows® 10 (64-bit) and Microsoft Windows® 11 (64-bit) (Windows® 10 is recommended)
Processor	1 GHz or higher (supporting hyper-threading or Multi-Core CPU)
Main memory	4 GB or more is recommended Software environment

2. MATLAB® and Simulink® products (from The MathWorks, Inc.)

MATLAB®	R2018b, R2021b to R2024b (R2024b is recommended)
Simulink®	Same as above
Stateflow®	Same as above
MATLAB® Coder™	Same as above
Simulink® Coder®	Same as above
Embedded Coder®	Same as above

3. MEX-file compiler

MEX-file is the interface that invokes C library from MATLAB®. MEX-file compiler is used to compile MEX files.

Embedded Target has been tested with the following compilers as the MEX file compiler for Windows® 10:

- Microsoft Visual C++ 2015 compiler (from Microsoft Corporation) (for MATLAB® R2018a)
- Microsoft Visual C++ 2019 compiler (from Microsoft Corporation) (for MATLAB® R2021b to R2024b)
- Microsoft Visual C++ 2022 compiler (from Microsoft Corporation) (for MATLAB® R2022a to R2024b)
- MinGW 6.3 C/C++ (distributed by mingw-w64) (only when using MATLAB® R2018b to R2024b)
- MinGW 8.1 C/C++ (distributed by mingw-w64) (only when using MATLAB® R2023a to R2024b)

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

4. IDE (from Renesas Electronics)

CS+	V8.13.00
e² studio	2024-10, 2025-01

5. Building environment (for generating a load module)

CC-RX	Included with CS+ V8.13.00 or later (from Renesas Electronics) Installed along with e² studio (from Renesas Electronics)
CC-RL	Included with CS+ V8.13.00 or later (from Renesas Electronics) Installed along with e² studio (from Renesas Electronics)
GNU ARM Embedded	(version: 13.2.1.arm-13-7 or later)

Remarks

1. For the MATLAB® and Simulink® products, an environment is constructed by using option products corresponding to the versions of MATLAB® and Simulink® being used.
2. When installing MATLAB®, it is recommended that the installation folder is changed to other than the folder for UAC (user account control). Depending on the version of MATLAB® in use, if the installation

folder is the folder for UAC such as “<system drive>: \Program Files”, a problem such that MEX cannot be built or the MATLAB® path cannot be saved may occur.

3. The IDE is CS+ or e² studio.

6. Debug Tools

Emulator E1, E2, E2 Lite, E20, EZ Emulator (For RL78 only), COM Port (For RL78/G23 and RL78/G24 only) (from Renesas Electronics),
J-Link (For RA device family and some device series of RX on e² studio only)

Simulator (Excluded RA devices family) (from Renesas Electronics)

Remark The simulator is included with CS+/e² studio

7. MATLAB®, MATLAB® Coder™, Simulink® Coder™, Embedded Coder ® are trademark or registered trademark of the MathWorks, Inc.
(https://www.mathworks.com/company/aboutus/policies_statements/trademarks.html)
Microsoft®, Windows®, Visual C++® are registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
(https://www.microsoft.com/en-us/legal/intellectualproperty/trademarks/en-us.aspx)

1.4 Feature Use Cases Policy

Embedded Target offers various features verifying algorithm of embedded models. Some features require specific license, which was registered with Renesas Electronics. This chapter describes use cases of these features.

1.4.1 License Policy

The following indicates list of features requiring specific licenses, which were registered with Renesas Electronics:

- Single-Core PIL Simulation on RX devices (Embedded Target for RX)
- Single-Core PIL Simulation on RL78 devices (Embedded Target for RL78)
- Single-Core PIL Simulation on RA devices (Embedded Target for RA)

Renesas Electronics offers flexible license policy in Embedded Target. You can choose some of these above features based on your demand.

When you own above license type, below operation are available:

- Single-Core PIL Simulation on RX devices
- Single-Core PIL Simulation on RL78 devices
- Single-Core PIL Simulation on RA devices
- Generating Load Module by:
    - For RX and RL device family: By Renesas Compiler
    - For RA devices family: By GNU ARM Embedded Compiler

1.4.2 Feature Use Cases

1.4.2.1 Target Devices for PIL Simulation

Embedded Target supports to verify algorithm of embedded models by PIL Simulation on various Renesas MCU families including RX, RL78, RA.

The PIL Simulation on RX, RL78, RA MCU families is required valid licenses, which was registered with Renesas Electronics. The license types are PIL Simulation modes offered by Embedded Target. For details of PIL Simulation modes, refer to Chapter 1.4.2.3 Target MCU

1.4.2.2 Build Tools for Target Devices

The Load Module, which is generated from embedded models, can be generated by Renesas Compilers for RX, RL78 device family and GNU ARM Embedded for RA device family. This feature is free-of-charge in Embedded Target.

1.4.2.3 Target MCU

Depending on a hardware capability of target device for PIL Simulation, Embedded Target offers to Single-Core Target MCU. The Target MCU denotes the number of cores that a load module can execute on during the PIL Simulation:

- Single-Core PIL Simulation: the load module can execute on 1 core of the target device. This feature can be used on all supported MCU families by Embedded Target

All of target MCU require valid licenses.

The Single-Core MCU:

- RX devices require “Embedded Target for RX” license.
- RL78 devices require “Embedded Target for RL78” license.
- RA devices require “Embedded Target for RA” license.

Remark Embedded Target offers methods to measure the performance of algorithms on embedded models. These methods are included in according to Target MCU.
For details, refer to Chapter 3.5 Verifying Algorithms of Code Generation Targets

1.4.3 License Management Model

Use the Renesas License Manager included in CS+ for managing licenses of Embedded Target. A license is made valid by using the Renesas License Manager to add the license key that was provided when purchasing the product. If CS+ is not installed, please install Renesas License Manager standalone.

1.5 Basic Usage

The usage is different depending on the code generation target: a Subsystem Block, a Reference model or a Top-level model. The usage for each of the cases is described below. Refer to the description according to the Simulink® model being used.

1.5.1 When Using a Subsystem Block

When the target for code generation is a Subsystem block, use Embedded Target in accordance with the following procedure.

1. Create a Simulink® model. Convert the code generation target blocks to the Subsystems and group them into a single Subsystem block.

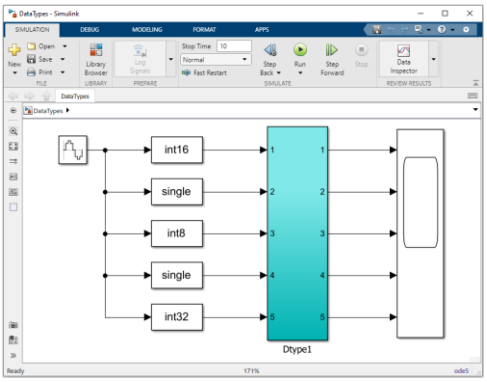

Figure 1-2. Subsystem Model

2. Use the configuration dialog box to set necessary parameters for generation of codes and a test environment.

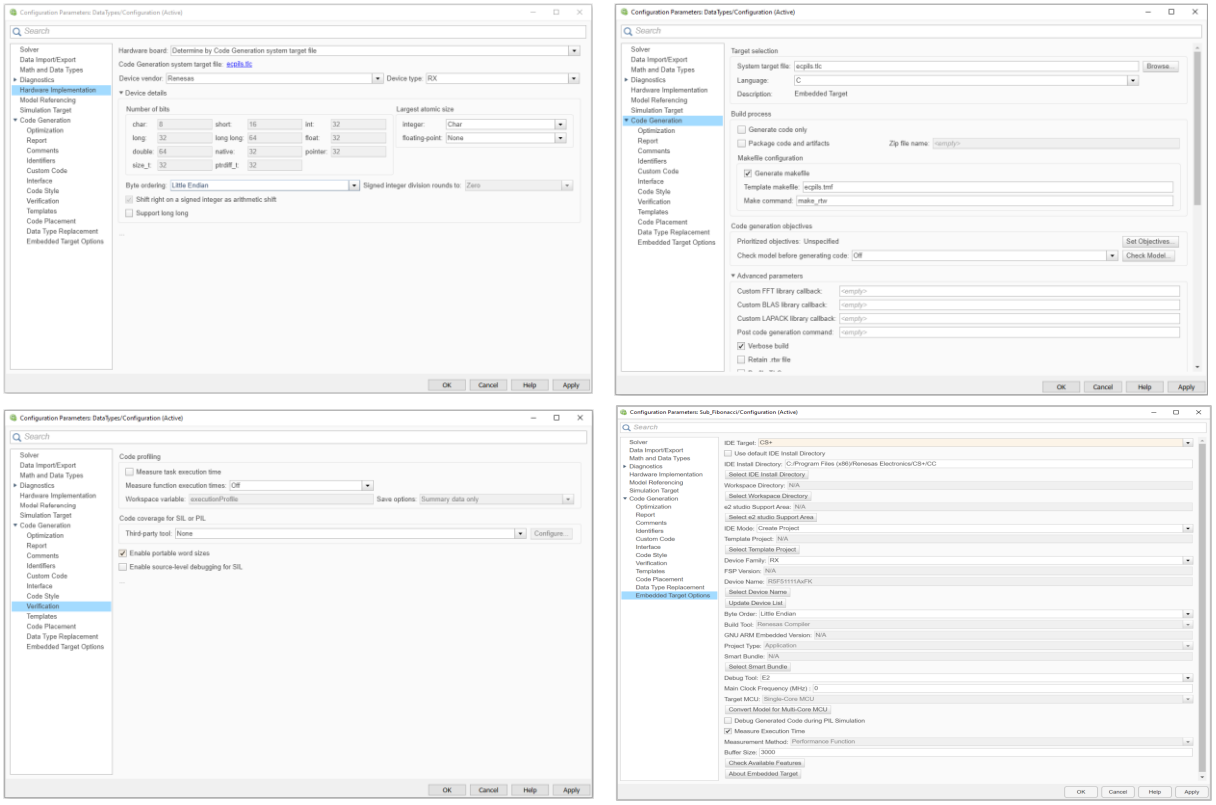

Figure 1-3. Configuration Parameters setting

3. Execute the ecpils_build command in the MATLAB® command window to generate codes and to generate a PIL test environment. Generate a C source code and start CS+/e² studio. The Simulink® model replaces the generated block for PIL sequential execution with the Subsystem block.

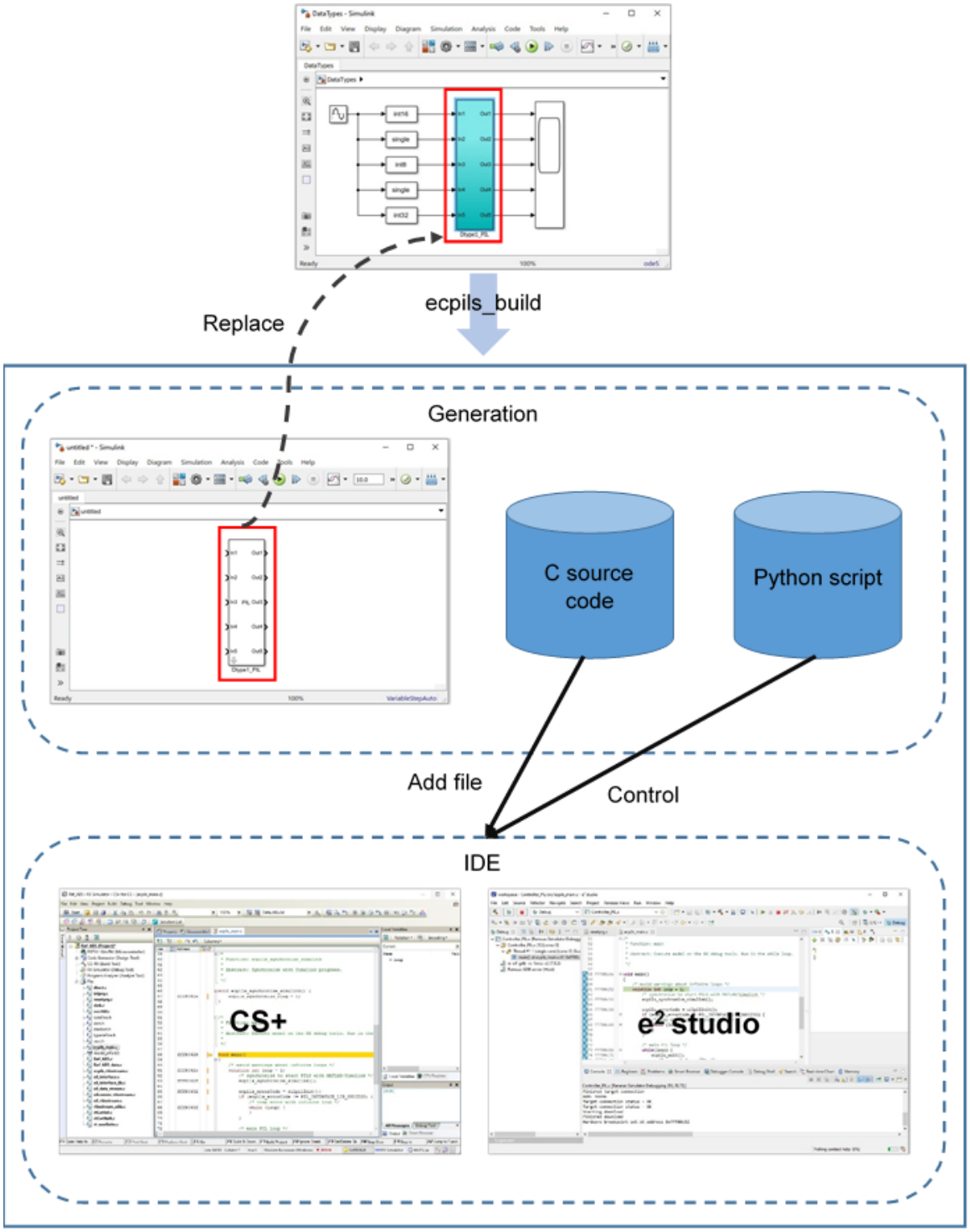

Figure 1-4. Flow of Embedded Target Processing, Case: Subsystem

4. Start simulation using the Simulink® model to execute PIL simulation.

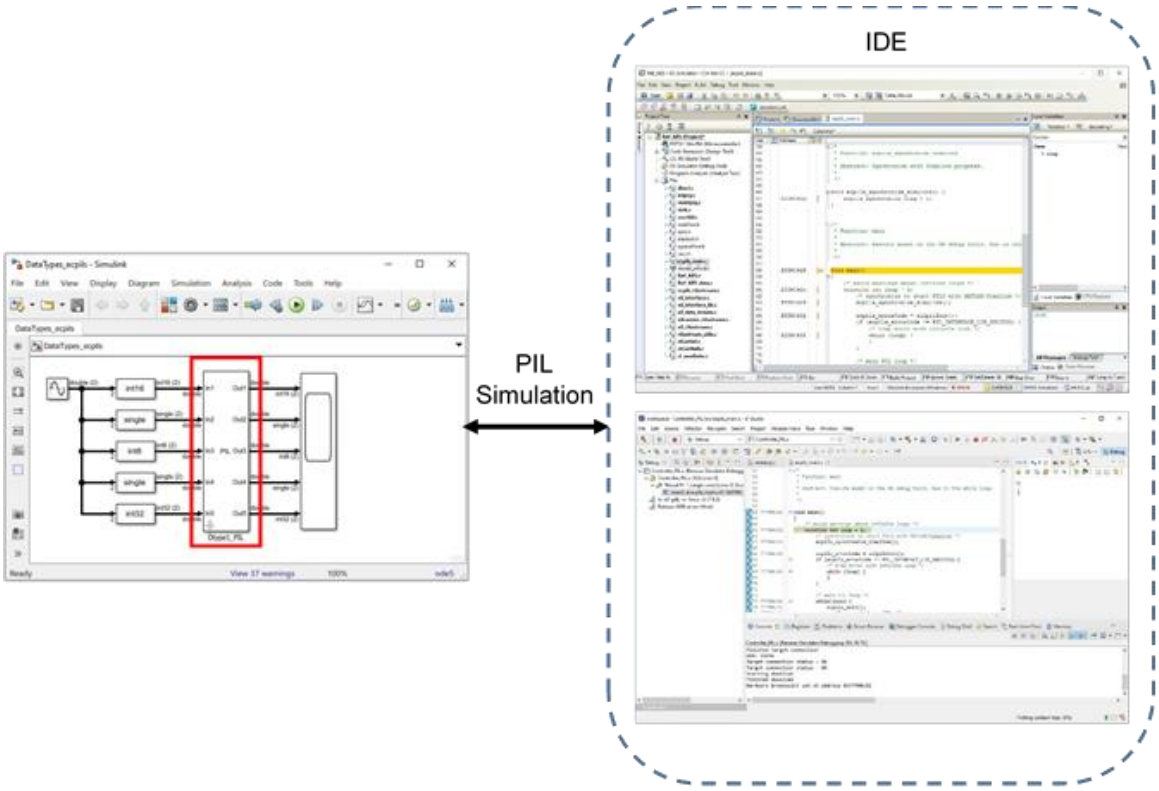

Figure 1-5. Flow of PIL Simulation, Case: Subsystem

1.5.2 When Using a Reference Model

When the target for code generation is a reference model, use Embedded Target in accordance with the following procedure.

1. Create a Simulink® model. The target for code generation is a Model block.

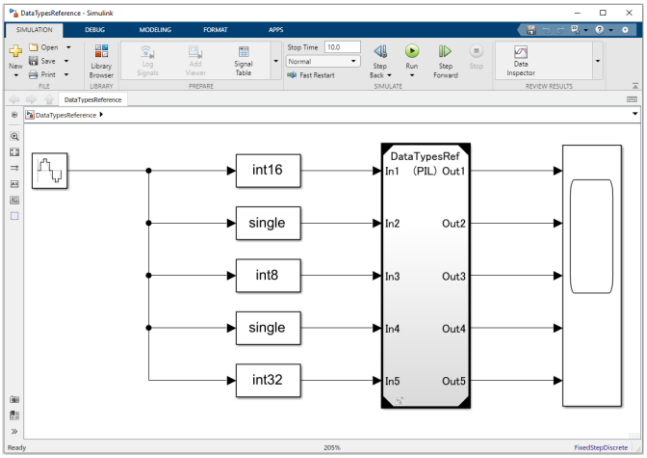

Figure 1-6. Reference Model

2. Use the configuration dialog box to set necessary parameters for generation of codes and a test environment.

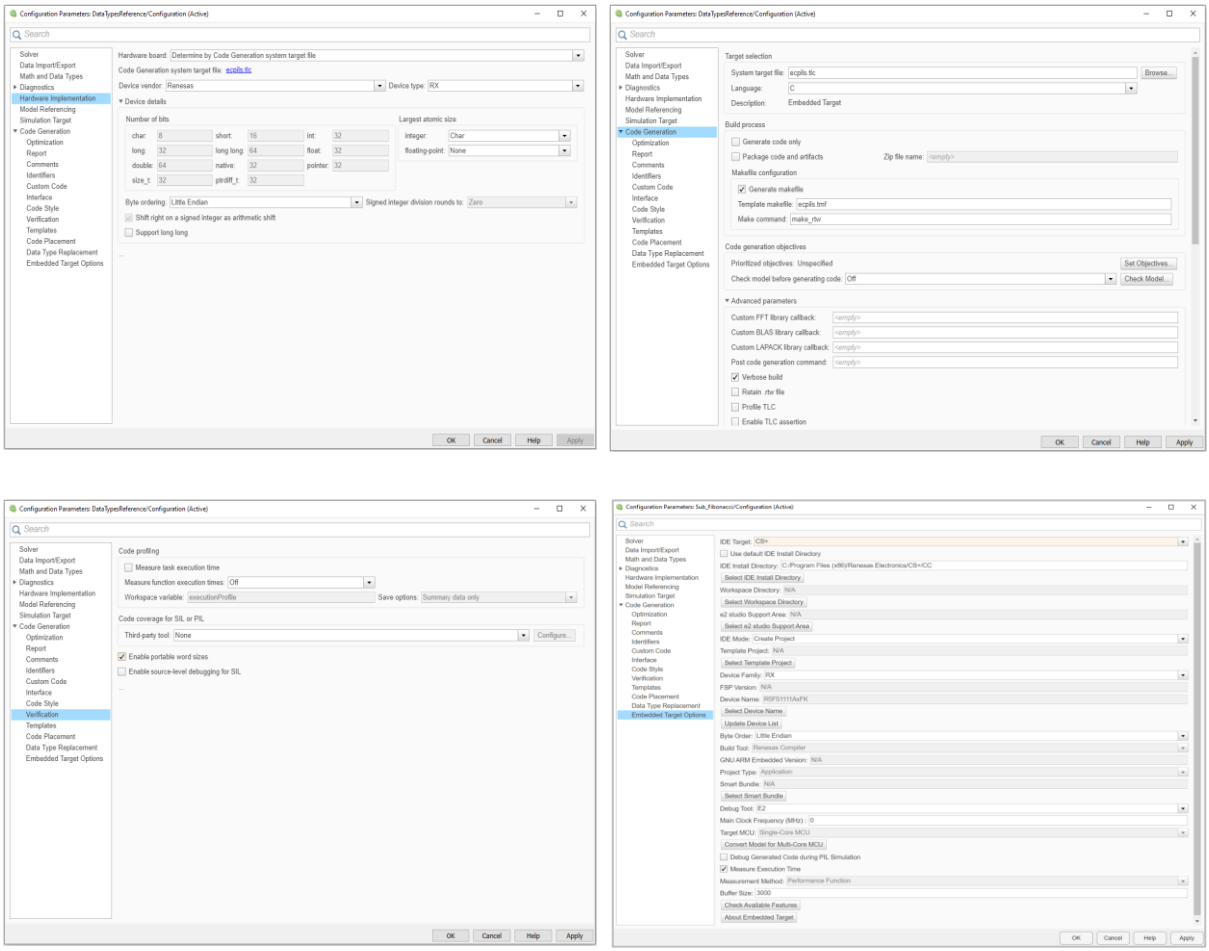

Figure 1-7. Configuration Parameters setting

3. Set PIL mode for the Model block.

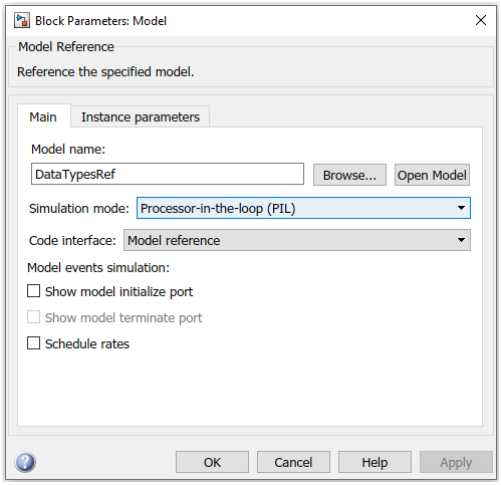

Figure 1-8. Block Parameters setting

4. Start simulation using the Simulink® model. Code generation, test environment generation and PIL simulation are executed automatically.

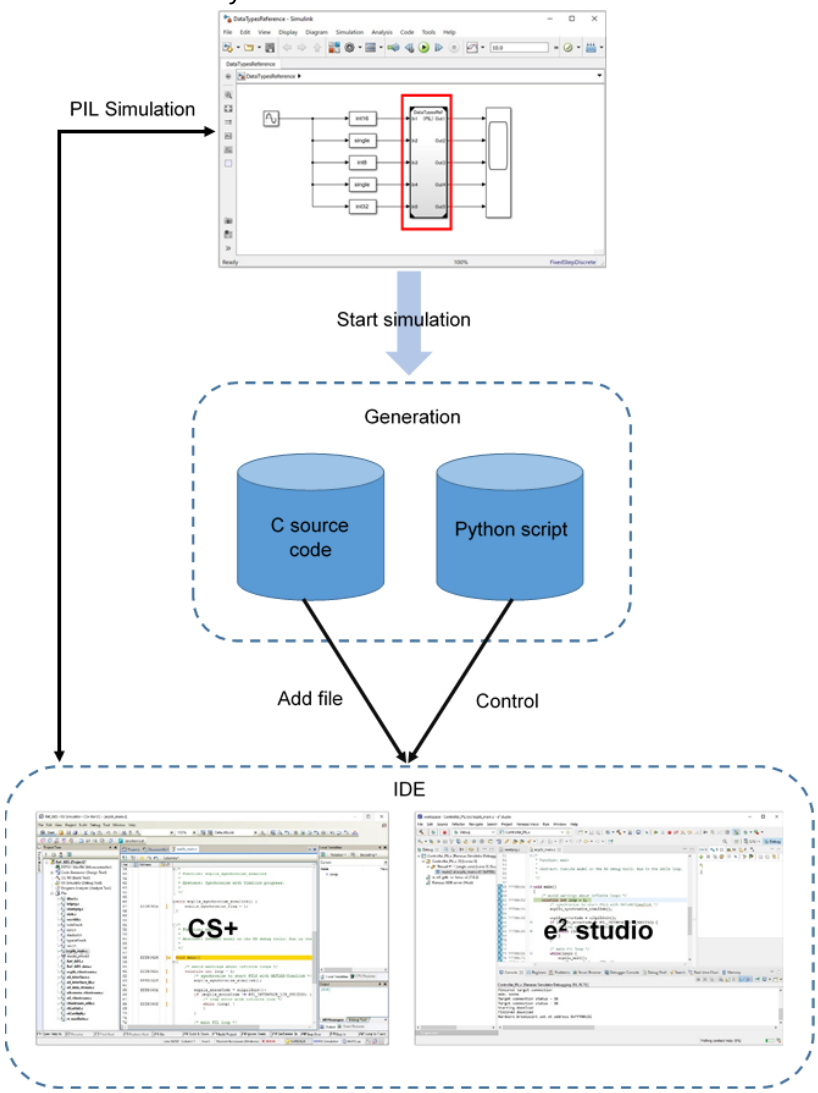

Figure 1-9. Flow of Embedded Target Processing, Case: Reference Model

1.5.3 When Using a Top-level Model

When the target for code generation is a top-level model, use Embedded Target in accordance with the following procedure.

1. Create a Simulink® model. The target for code generation is a top-level model.

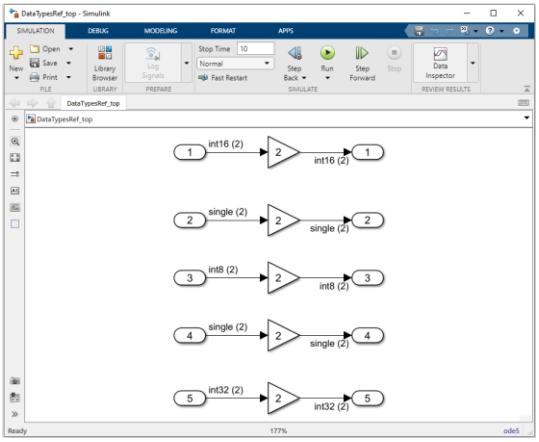

Figure 1-10. Top-level Model

2. Use the configuration dialog box to set necessary parameters for generation of codes and a test environment.

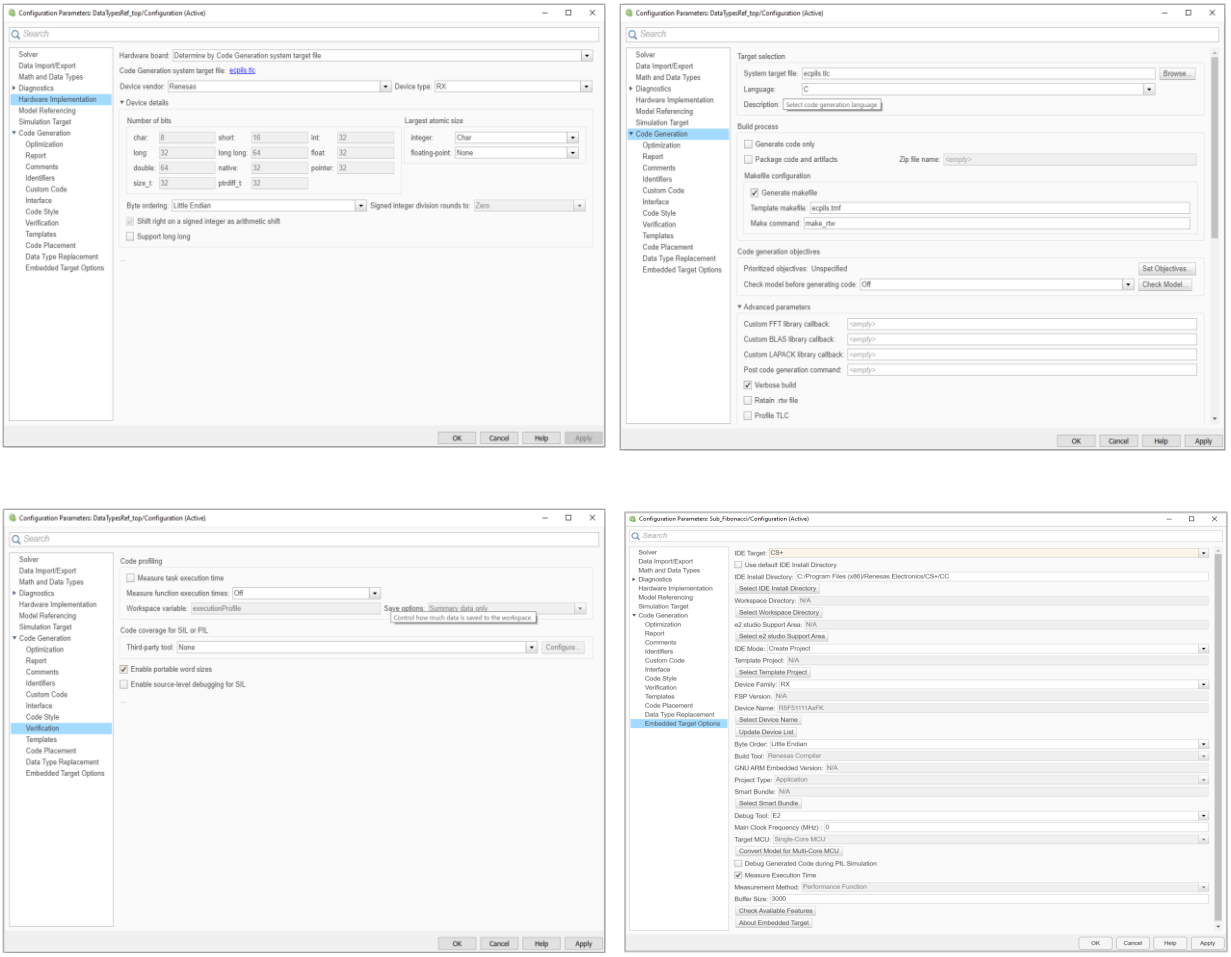

Figure 1-11. Configuration Parameters setting

3. Set PIL mode for the Simulink® model.

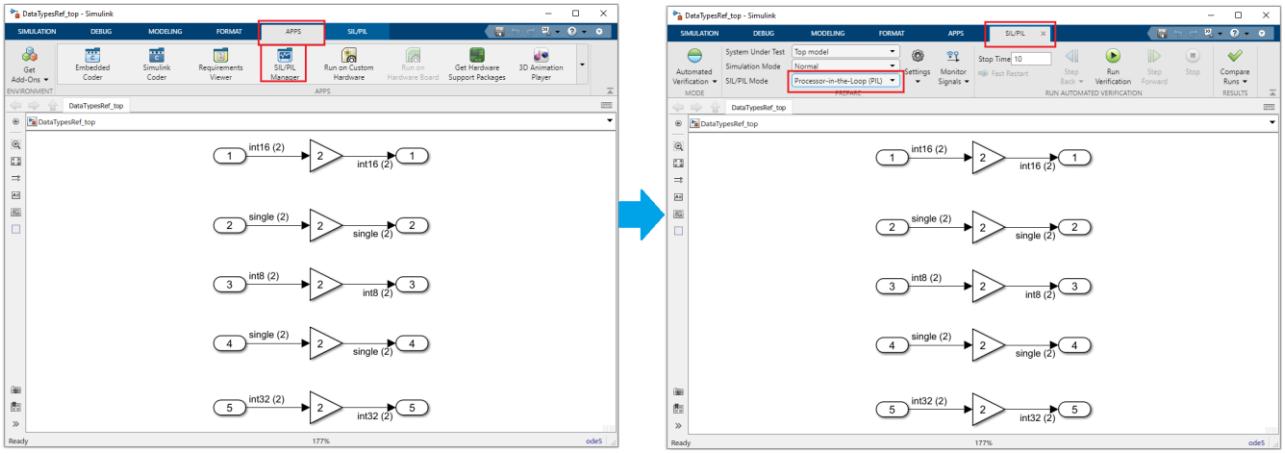

Figure 1-12. PIL Mode Setting

4. Start simulation using the Simulink® model. Code generation, test environment generation and PIL simulation are executed automatically.

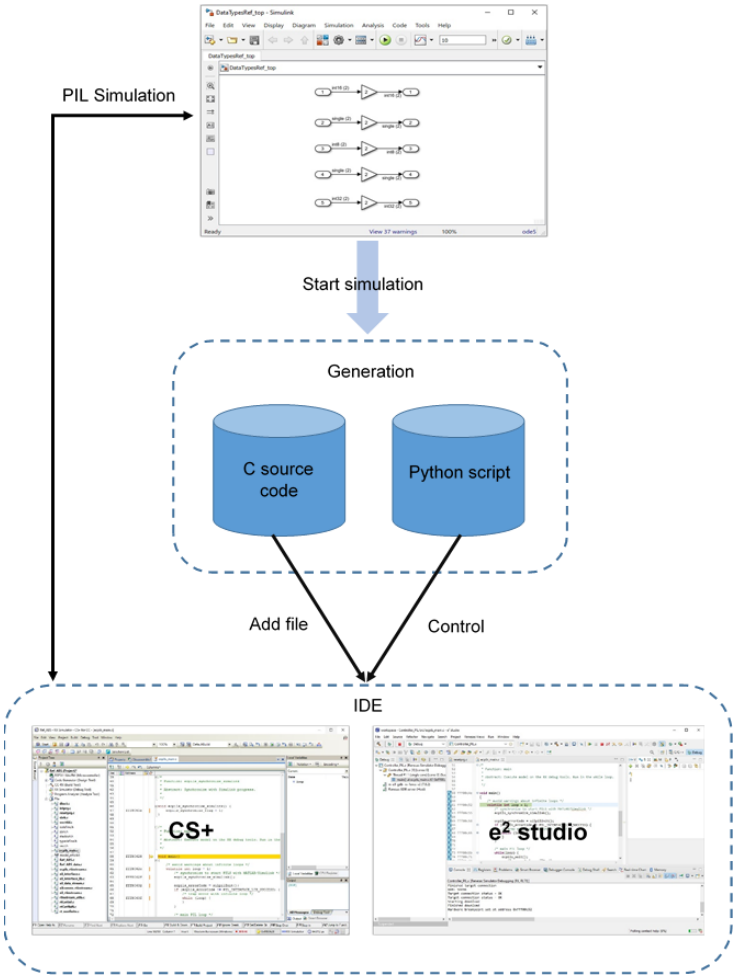

Figure 1-13. Flow of Embedded Target Processing, Case: Top-level Model

2. INSTALLATION

This chapter explains how to install and uninstall Embedded Target.

2.1 Installing Embedded Target

Described below is the information about the Embedded Target package and the installation procedure.

2.1.1 Package

The following installation file is necessary to install Embedded Target.

- Renesas_Embedded_Target_<version information>_Setup.exe

After installed Embedded Target, the programs, documents, and samples are in the following folder structure.

<version information>\	et\	A set of Embedded Target programs
	et\plugins\IronPython	A package of IronPython 2.7.4
	smp\	Sample models

2.1.2 Procedure

See the Quick Started Guide and complete installation, license addition, and making initial settings.

2.2 Uninstalling Embedded Target

Proceed as follows to uninstall Embedded Target.

1. Start MATLAB® and remove the <Embedded Target installation folder>\<version information>\ EmbeddedTarget folder using the Set Path dialog box.

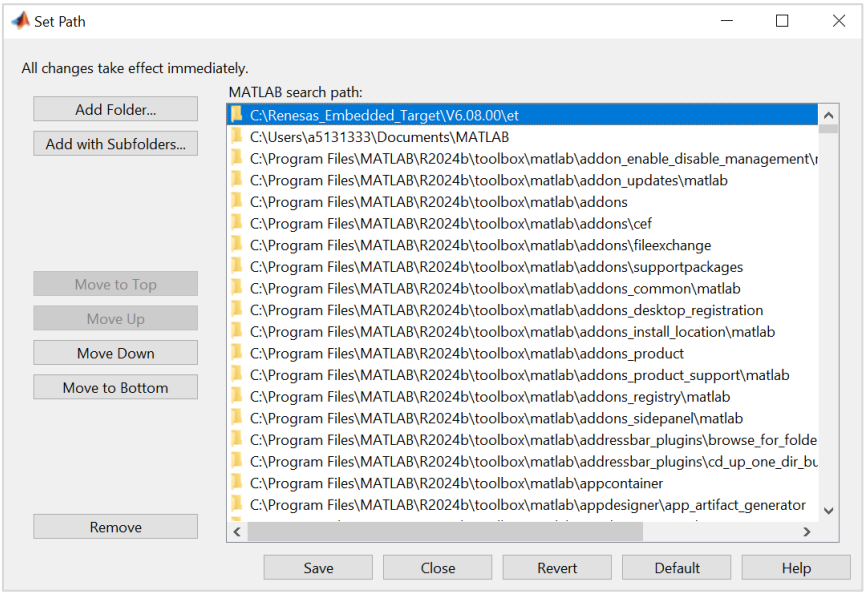

Figure 2-1. Set Path Dialog Box

2. Delete the <Embedded Target installation folder>\<version information> folder and delete the files in <MATLAB® install folder>\bin\win64\ (when using MATLAB® 64-bit versions) which has been copied in installation.

2.3 Deleting a License

A license can be deleted by the Renesas License Manager. When changing the PC in use, delete the license and then register the license in the new PC.

3. FUNCTIONS

This chapter describes the functions provided by Embedded Target.

3.1 Overview

Embedded Target provides the functions to generate a test environment and to verify algorithms.

Embedded Target generates a test environment in cooperation with the Embedded Coder®.

Code can be generated for three kinds of targets: a subsystem block, a reference model, and a top-level model. The procedure to generate a test environment is different in these three kinds.

Table 3-1. Three Kinds of Code Generation Targets

Code generation target	Subsystem block	Reference model	Top-level model
Input/output for generated code	Uses an I/O port of the subsystem block.	Uses an I/O port of the model block.	Uses the MATLAB® workspace variable.
PIL sequential execution	Performs PIL sequential execution by replacing the subsystem block with the block for PIL sequential execution.	Sets the PIL mode for the model block and performs PIL sequential execution.	Sets the PIL mode for the model and performs PIL sequential execution.

1. When the target for code generation is a subsystem block

   A block for PIL sequential execution is generated from the Subsystem block and verification is performed by replacing that block with the Subsystem block.

   a. Generate the block for PIL sequential execution from an embedded model using a C code generation tool (Embedded Coder®)
   b. Replace the block for PIL sequential execution with an embedded model Subsystem block
   c. Generate C source files from an embedded model using a C code generation tool (Embedded Coder®)
   d. Start CS+/e^2 studio
   e. Register the generated C source files in the CS+/e^2 studio project
   f. Select the debug tool to use when running PIL simulation
   g. Generate a load module from the C source files using the build function of the build tool of CS+/e^2 studio
   h. Download the generated load module into the debug tool
   i. Information obtained from PIL simulation allows you to verify the algorithms in an embedded model

2. When the target for code generation is a top-level model or a reference model

   When the target for code generation is a top-level model or a reference model, a test can be performed by setting PIL mode before running simulation, unlike when the target is a Subsystem block. Once a test is started, C source files and a CS+/e^2 studio project file are generated automatically. A test is performed by using them for communication with CS+/e^2 studio.

   a. Prepare a top-level model or a reference model
   b. Set the configuration parameters
   c. Specify PIL mode as simulation mode
   d. Generate C source files from an embedded model using a C code generation tool (Embedded Coder®)
   e. Start CS+/e^2 studio
   f. Register the generated C source files in the CS+/e^2 studio project
   g. Select the debug tool to use when running PIL simulation
   h. Generate a load module from the C source files using the build function of the build tool of CS+/e^2 studio
   i. Download the generated load module into the debug tool
   j. Information obtained from PIL simulation allows you to verify the algorithms in an embedded model

Sections of “Executing PIL Simulation for Subsystem Code Generation Target Block”, "Executing PIL Simulation for Reference Code Generation Target Model", and "Executing PIL Simulation for Top-level Code Generation Target Model" describe the cases when the target of code generation is the subsystem block, the reference model and the top-level model, respectively.

3.2 Executing PIL Simulation for Subsystem Code Generation Target Block

The following describes how to generate a test environment necessary for PIL simulation when the target for code generation is a subsystem block.

3.2.1 Generating a Test Environment

This section explains how to generate a test environment necessary for PIL simulation.

The explanation uses a sample model DataTypes.slx provided with Embedded Target for Single-Core PIL Simulation modes.

3.2.1.1 Prepare debug configuration for Non-secure with Secure Bundle for RA family

To execute PIL Simulation with RA TrustZone® Non-Secure project successfully, which needs to refer to the smart bundle (*.sbd) file of RA TrustZone® Secure project type. So, in this section describes how to generate RA TrustZone® Secure project and necessary settings to do this.

- Step 1: Open e² studio, create project RA TrustZone® Secure project using the following link: **Generating an RA Secure Project for e² studio.**
- Step 2: Connect to the target device. Select [Menu Bar] > [Run] > [Renesas Debug Tools] > [Renesas Device Partition Manager].
  In [Renesas Device Partition Manager] > Select [Device Family] is “Renesas RA” > Check on checkbox [Initialize device back to factory default] > Click “Run” and “Close” as below images:

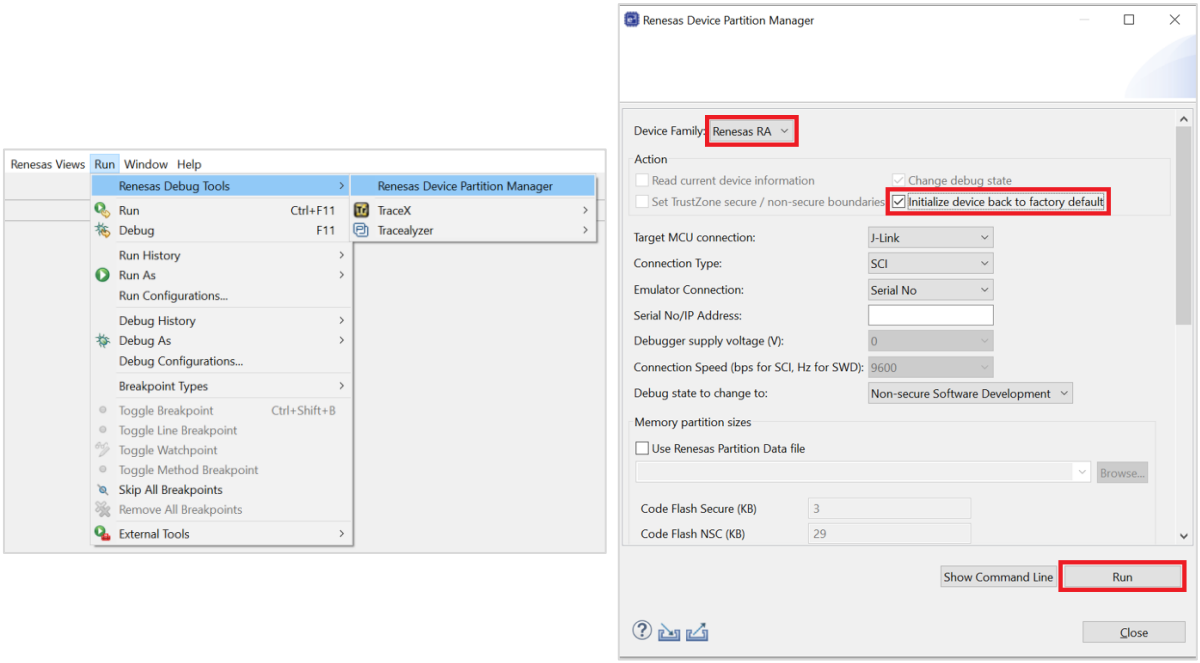

Figure 3-1. Renesas Device Partition Manager for RA TrustZone® Secure

- Step 3: Build & Download RA TrustZone® Secure project on Step 1.
  Smart Bundle file (*.sbd) is generated in the same folder as the build target during this build.
- Step 4: Terminate current debug session.

- Step 5: Disconnect and re-connect to the target device. Select [Menu Bar] > [Run] > [Renesas Debug Tools] > [Renesas Device Partition Manager].
  In [Renesas Device Partition Manager] > Select [Device Family] is “Renesas RA” > Uncheck on checkbox [Initialize device back to factory default] > Check on checkbox [Change debug state] > Select [Debug state to change to] is “Non-secure Software Development” > Click [Run] and [Close] as below images:

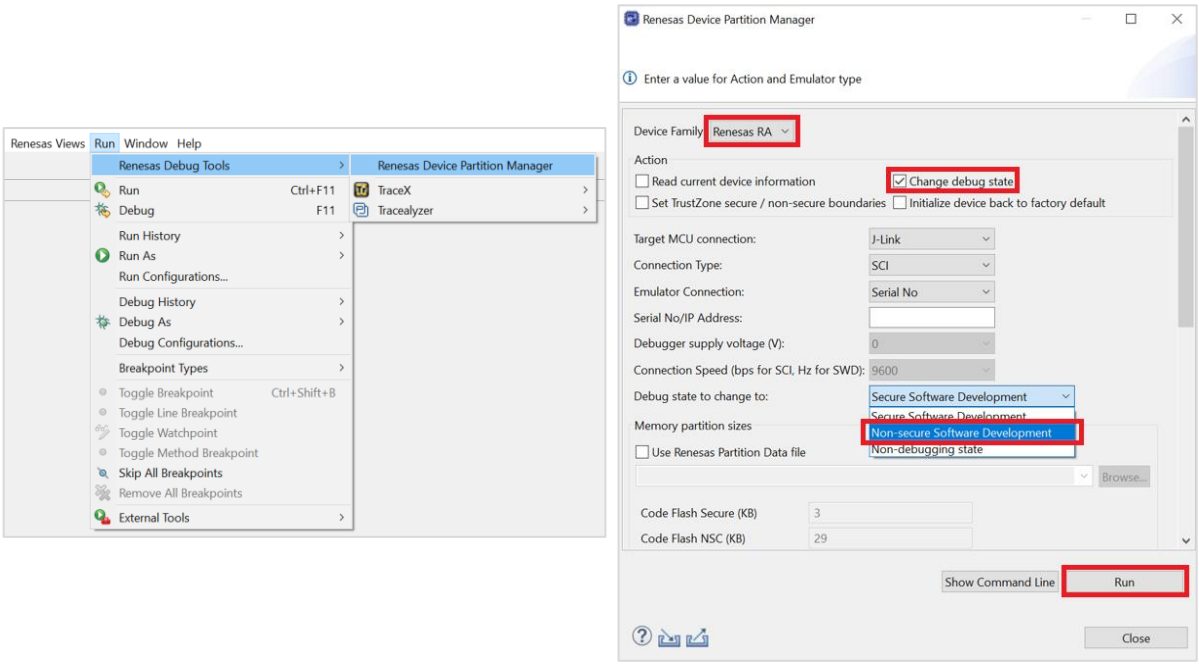

Figure 3-2. Renesas Device Partition Manager for RA TrustZone® Non-Secure

- ·Step 6: Close e2 studio.

3.2.1.2 Embedded Model Subsystem

Convert embedded model blocks from which a C source file will be generated to a subsystem.

Remark In the sample models provided with Embedded Target, blocks have already been converted to subsystems. Therefore, there is no need to convert blocks in the sample models to subsystems.

The following table shows the block name of the subsystem in the sample model.

Table 3-2. Subsystem Block Name

Sample model name	Code generation target	Subsystem block name
DataTypes.slx	Subsystem block	DataTypes_PIL

3.2.1.3 Setting configuration parameters

Embedded Target implements execution of test environment generation by interworking with Embedded Coder®. Therefore, it is necessary to check/set Embedded Coder® options when using the test environment generation functions provided by Embedded Target.

1. Open the Configuration Parameters dialog box

Select [Model Configuration Parameters] from the [Simulation] menu in the DataTypes window to open the Configuration Parameters dialog box.

2. Set [Hardware Implementation] options

   Select [Hardware Implementation] in the [Select] area and select [Renesas] for [Device vendor]. Then make the settings described below and click the [Apply] button.

   [When using RX]

   Select [RX] for [Device type].

   Select [Little Endian] or [Big Endian] for [Byte ordering].

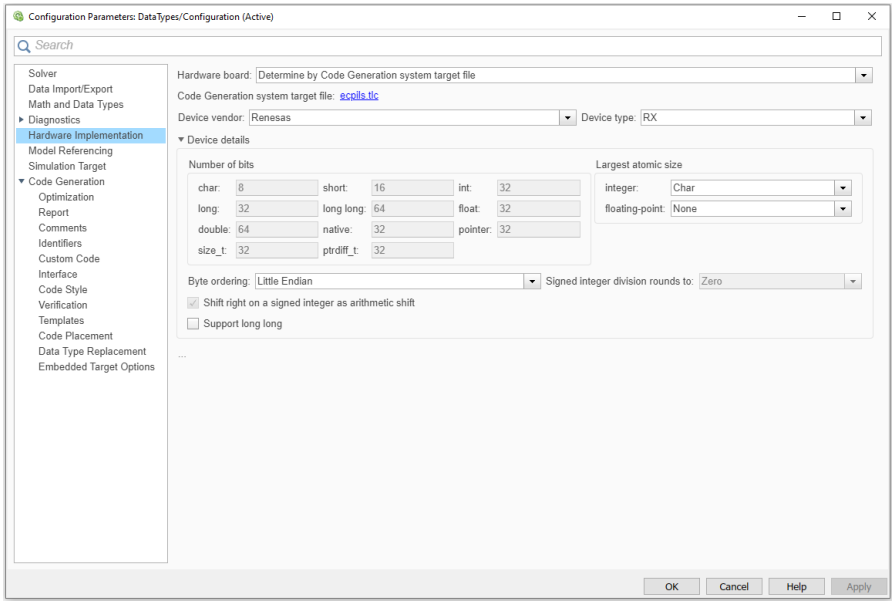

Figure 3-3. [Hardware Implementation] Options for RX

[When using RL78 or RA]

Select [RL78] or [ARM Cortex] for [Device type] in MATLAB R2018b (or [ARM Cortex-M] for R2021b and latest).

[Byte ordering] is set as [Little Endian] and cannot be changed.

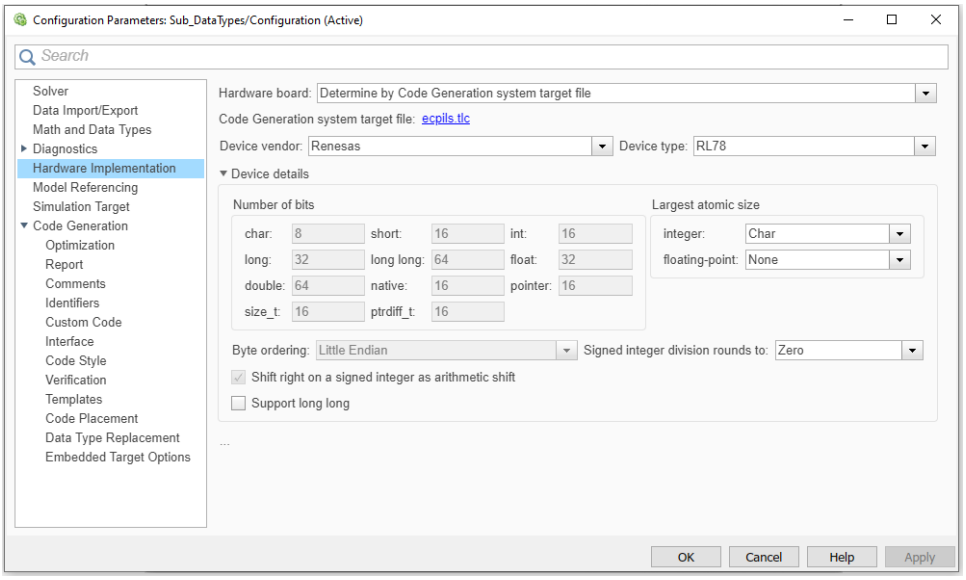

Figure 3-4. [Hardware Implementation] Options for RL78

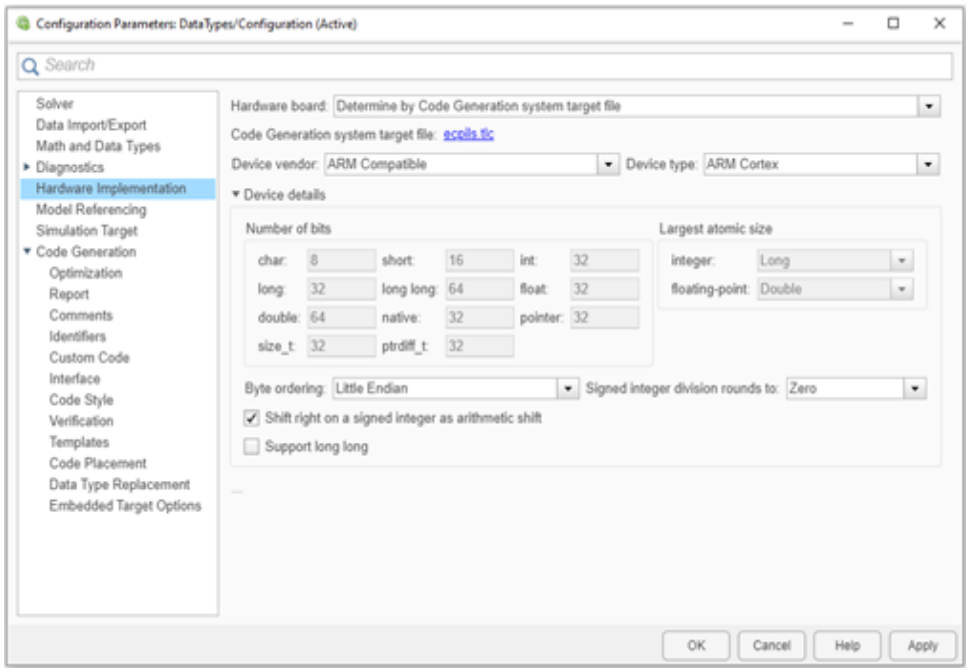

Figure 3-5. [Hardware Implementation] Options for RA

Remark 1. Device type value in [Hardware Implementation] panel can be changed by setting Device Family value in [Embedded Target Options] panel. This result goes into effect after pressing “OK” or “Apply” button on [Embedded Target Options] panel.

2. The setting of Device Family value is enabled when [IDE Mode] value is Create Project on [Embedded Target Options] panel.

3. Set [Code Generation] options

Select [Code Generation] in the [Select] area. Then make the settings shown in the figure below and click the [Apply] button.

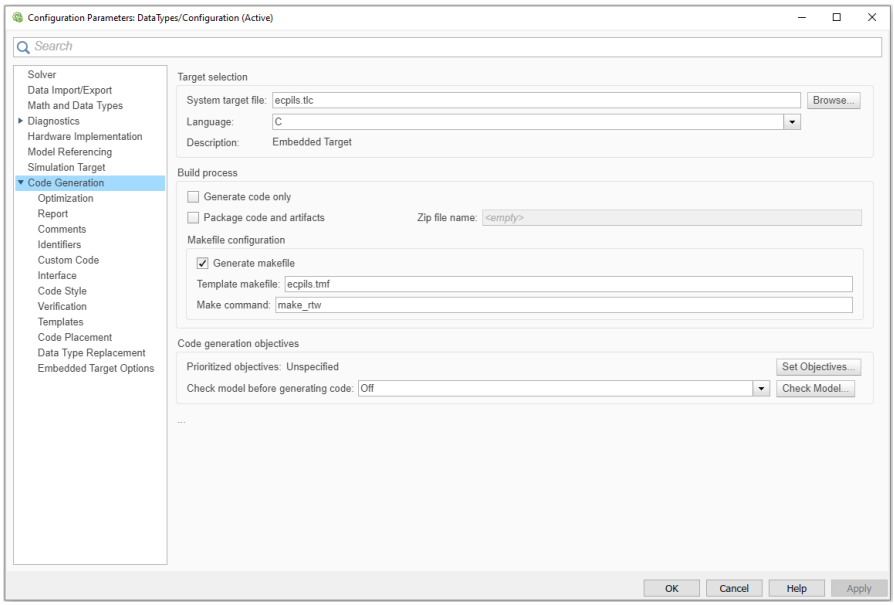

Figure 3-6. [Code Generation] Options

Remark The template make file (ecpils.tmf) will be overwritten according to selected Mex compiler (>>mex -setup).

4. Set [SIL and PIL Verification] or [Verification] options

   Select [Code Generation] - [Verification] in the [Select] area. Then make the settings shown in the figure below and click the [Apply] button.

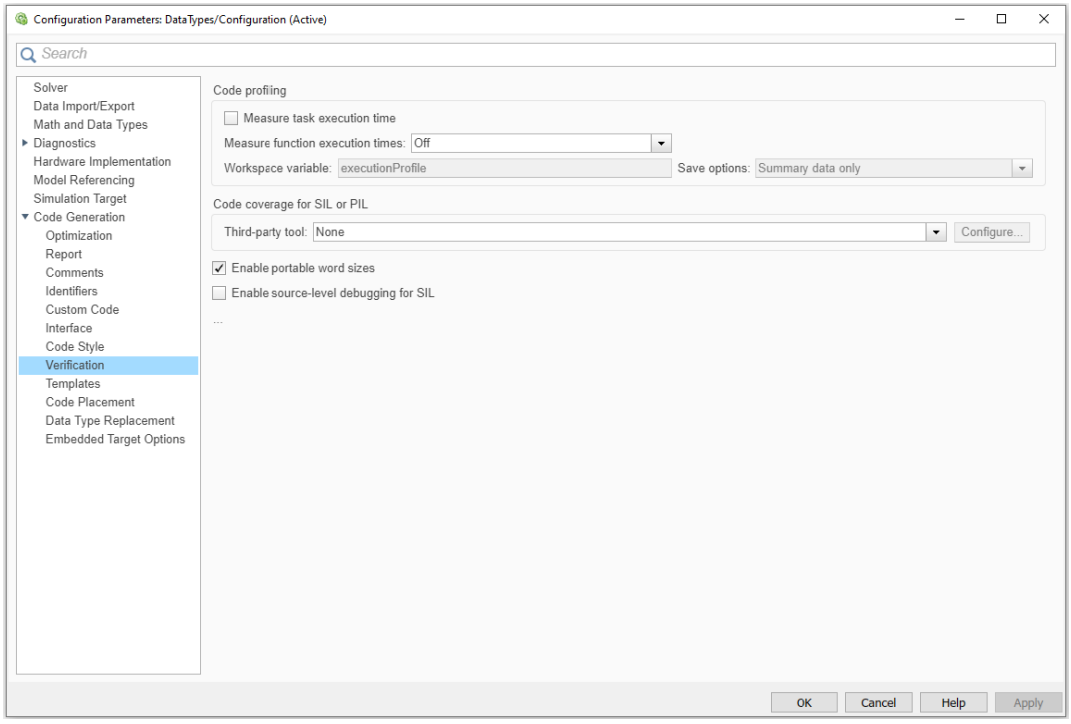

Figure 3-7. [Verification] Options

5. Set [Embedded Target Options]

   Select [Code Generation] - [Embedded Target Options] in the [Select] area. Then make the settings shown in the figure below and click the [Apply] button.

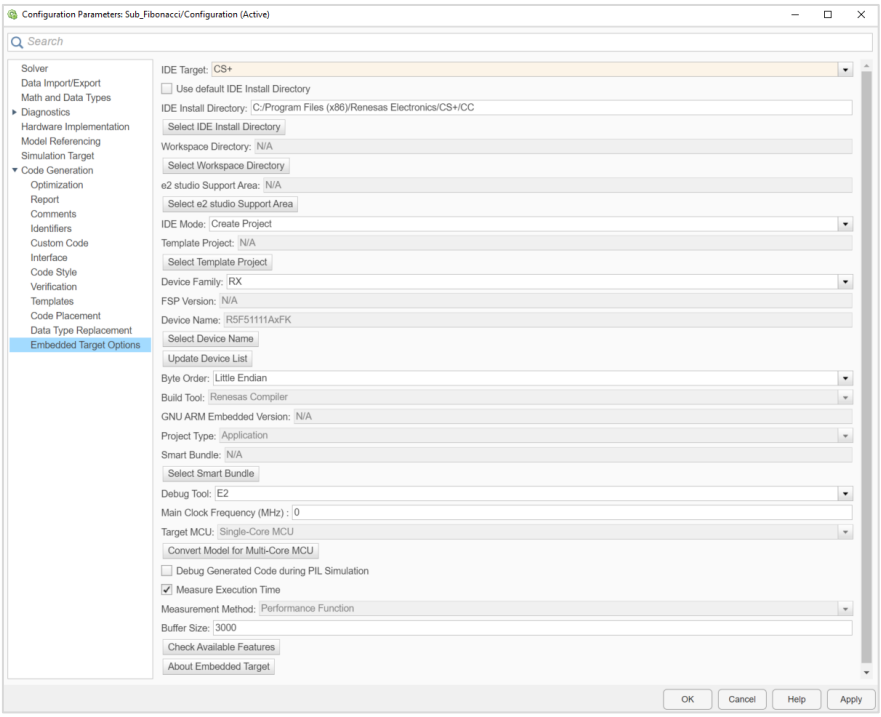

Figure 3-8. [Embedded Target Options] Settings

The following table shows the items in the [Embedded Target Options] pane.

Table 3-3. Embedded Target Options

Item name	Description
IDE Target *1	Select the IDE which you want to use. IDE is CS+ (default) or e^2 studio.
[Use default IDE Install Directory] checkbox	This option sets the default IDE installation directory of selected IDE target to [IDE Install Directory] option. • For CS+: "C:/Program Files (x86)/Renesas Electronics/CS+/CC" • For e^2 studio: "C:/Renesas/e2_studio" If the default IDE installation directory is invalid, this option is unchecked automatically and the value of [IDE Install Directory] option is changed to empty.
IDE Install Directory *2	Specifies the folder where CS+/e^2 studio has been installed (the folder where CubeSuiteW+.exe or e2studio.exe is stored) as an absolute path.
[Select IDE Install Directory] button *2 *3	Clicking this button displays the dialog box for selecting the absolute path of the folder where the CS+/e^2 studio is installed. Folder specifications made in the dialog box that is opened by this button are reflected in the [IDE Install Directory] field.
Workspace Directory	Specifies the workspace folder where e^2 studio project has been stored as an absolute path.
[Select Workspace Directory] button *4	Clicks this button to display the dialog box for selecting the absolute path of the workspace folder of e^2 studio. Folder specifications made in the dialog box that is opened by this button are reflected in the [Workspace Directory] field.
e2 studio Support Area *5	Specifies the e^2 studio support folder where e^2 studio generates Integration Service API to drive it as an absolute path

[Select e2 studio Support Area] button *6	Clicking this button to display the dialog box for selecting the absolute path of the e² studio support folder. Folder specifications made in the dialog box that is opened by this button are reflected in the [e2 studio Support Area] field	
IDE Mode *7	Selects the type of project file that is loaded when the CS+/e² studio starts and whether a series of processing including download of a load module is performed after the CS+/e² studio start-up.	
	Create Project	The default CS+/e² studio project file provided by Embedded Target is loaded (default).
	Open Project	The CS+/e² studio project file saved at the last exit is loaded.
	Open Project & LM Download	The CS+/e² studio project file saved at the last exit is loaded on start-up. Then, a load module is downloaded to the debug tool.
	Template Project & LM Download	The CS+/e² studio project file generated based on the CS+/e² studio project file specified with [Template Project] is loaded on start-up. Then, a load module is downloaded to the debug tool.
Template Project *8	When [Template Project & LM Download] is selected for [IDE Mode], use the [Select Template Project] button and specify the CS+ project file name or the e² studio project folder which is used as a template with the absolute path.	
[Select Template Project] button *9	Displays the dialog box to specify the CS+ project file or the e² studio project folder which is used as a template with the absolute path. The selected result is reflected to [Template Project].	
Device Family *10	Selects the series name of the microcontroller being used.	
	RX	Selects the RX family as the microcontroller series.
	RL78	Selects the RL78 family as the microcontroller series.
	RA *11	Selects the RA family as the microcontroller series.
FSP Version	Specifies version of FSP being used to create RA family project.	
Device Name *12	Uses the [Select Device Name] button to specify the name of the microcontroller being used.	
[Select Device Name] button *13	Displays a list of microcontrollers, from which you can select the microcontroller that you are using. The selection is reflected in the [Device Name] field.	
[Update Device List] button *14	Updates the list of microcontrollers displayed by clicking [Select Device Name] button.	
Byte Order *15	Selects the Byte Order of the microcontroller being used.	
	Little Endian	Select the Little Endian as the Byte Order
	Big Endian	Select the Big Endian as the Byte Order
Build Tool *16	Selects the Build tool for CS+/e² studio project, this indicates the compiler will be used to generate the load module.	
	Renesas Compiler	Selects any of Renesas compilers, which will be determined by CS+/e² studio when creating new project.
	GCC ARM Embedded Compiler	Selects any of GCC ARM Embedded compilers, which will be determined by e² studio when creating new project for RA devices family.
GNU ARM Embedded Version	Specifies version of GNU being used to create RA family project	

Project Type *17 *20	Selects the Project Type.	
	Application	Default value for RX, RL78 device in CS+
	Executable	Default value for RX, RL78 device in e²studio
	Flat (Non-TrustZone®)	Default value for RA device family in e²studio. Select the Flat (Non-TrustZone®) as the Project Type for RA device family
	TrustZone® Secure	Select the TrustZone® Secure as the Project Type for RA device family
	TrustZone® Non-Secure	Select the TrustZone® Non-secure as the Project Type for RA device family
Smart Bundle *18 *20	Specifies the absolute path of the Smart Bundle (*.sbd) file of the RA device family project.	
[Select Smart Bundle] button *19 *20	Clicking this button to display the dialog box for selecting the absolute path of the Smart Bundle (*.sbd) file. Smart Bundle (*.sbd) file specifications made in the dialog box that is opened by this button are reflected in the [Smart Bundle] field.	
Debug Tool *21	Selects a Debug Tool and connection type connecting to the target device	
	E2 (default)	Connects to the target device through E2 Emulator
	E2 Lite	Connects to the target device through E2 Lite Emulator
	COM Port	Connects to the target device through COM Port (For RL78/G23, RL78/G24 only)
	Simulator *22	Uses Simulator of the target device
	E20(Serial)	Connects to the target device through E20 Emulator with Serial connection.
	E20(JTAG)	Connects to the target device through E20 Emulator with JTAG connection.
	EZ_Emulator *23	Connects to the target device through EZ Emulator
	E1(JTAG)	Connects to the target device through E1 Emulator with JTAG connection.
	E1(Serial)	Connects to the target device through E1 Emulator with Serial connection.
	J-Link	Connects to the target device through J-Link Emulator
Main Clock Frequency (MHz) *24	Sets main clock frequency of the target device	
Debug Generated Code during PIL Simulation	When checked	Measure Execution Time feature is disabled. PIL Simulation could be interrupted by User's operation on CS+/e² studio.
	When not checked	Measure Execution Time feature can be enabled by User. PIL Simulation could NOT be interrupted by user's operation on CS+/e² studio.

Measure Execution Time *22	When checked	Execution time is measured in verification of algorithm.
	When not checked	Execution time is not measured in verification of algorithm and PIL simulation is performed at high speed (default).
Measurement Method	Selects the Time Measurement method is being used.	
	Performance Function	The execution time of the load module which executes on CS+/e² studio during the PIL Simulation is measured by using Performance Function of CS+/e² studio debug tools. This option is set automatically when [Device Family] are RL78, RX, RA and [Target MCU] is “Single-Core MCU”.
	Software Trace *26	The execution time of the e² studio load module is measured by the debugging function of software trace during PIL simulation.
Buffer Size	Specifies value of Buffer Size in range 1000 ~ 3000	
[Check Available Features] button *25	Displays list of available requiring features in Embedded Target System.	
[About Embedded Target] button	Displays version information and copyright information of Embedded Target.	

*1… When CS+ is selected for [IDE Target], the value of [IDE Install Directory] is changed to the default folder where CS+ has been installed, [Workspace Directory] and [e2 studio Support Area] are disabled.

Otherwise, when e² studio is selected for [IDE Target], the value of [IDE Install Directory] is changed to the default folder where e² studio has been installed, [Workspace Directory] and [e2 studio Support Area] are enabled.

*2… When CS+/e² studio has not been installed in the folder specified with the dialog box (CubeSuiteW+.exe/e2studio.exe file does not exist in the specified folder), an error is output, and the information of the specified folder is not reflected in [IDE Install Directory].

*3… When the [Use default IDE Install Directory] checkbox is checked, if the [Select IDE Install Directory] button is clicked, an error message displays.

*4… When the [IDE Target] is “CS+”, if the [Select Workspace Directory] button is clicked, an error message displays.

*5… To specify [e2 studio Support Area], do the following steps: invoke e² studio > [Help] > [About e² studio] > [Installation Details] to show [e² studio Installation Details] dialog. In [e² studio Installation Details] dialog, select [Support Folders] tab, you can see the absolute path of “e² studio support area”.

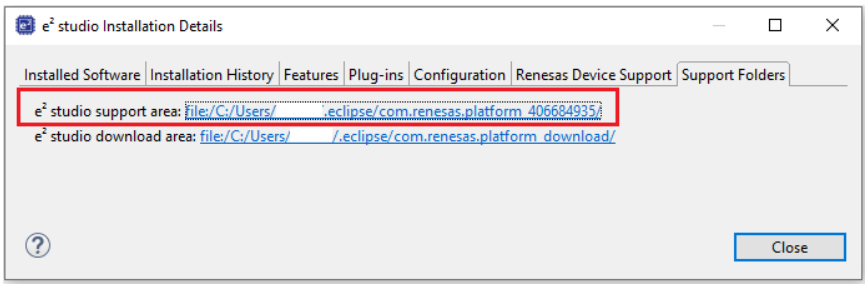

Figure 3-9. The absolute path of "e² studio support area"

*6… When the [IDE Target] is “CS+”, if the [Select e2 studio Support Area] button is clicked, an error message displays.

*7… When “Open Project” or “Open Project & LM Download” is selected and there is no project file saved at the last exit, the default project file is created and then CS+/e^2 studio is started as in the case of selecting “Create Project”.

*8… The setting is only valid if “Template Project & LM Download” is selected for [IDE Mode].

Specify the CS+ project file or e^2 studio project folder created with Embedded Target. When a CS+ project file or e^2 studio project folder which has not been created with Embedded Target is specified, normal operation is not guaranteed.

*9… If this button is clicked while a mode other than “Template Project & LM Download” is selected for [IDE Mode], an error occurs.

*10... The setting is only valid if “Create Project” is selected for [IDE Mode] and the used IDE contains the device family and its compiler.

The “RX” item is valid to choose in [Device Family] list only when “Embedded Target for RX” license is valid.

The “RL78” item is valid to choose in [Device Family] list only when “Embedded Target for RL78” license is valid.

The “RA” item is valid to choose in [Device Family] list only when “Embedded Target for RA” license is valid.

To check validity of a Device Family, press the [Check Available Features] button.

*11… When creating project for RA device family on e^2 studio, there is a case that version of FSP or GCC ARM Embedded is different with the packages included with e^2 studio. If the specified version of FSP and GCC ARM Embedded is not correct, the project creation will be failed. To specify the correct version, modify the value of “FSP Version” or “GNU ARM Embedded Version” in “Embedded Target Options”.

*12... If there is no CS+ project file or e^2 studio project folder that has been generated at the last exit, information that has been specified previously is used to generate the device name.

*13... The button is only displayed if “Create Project” is selected for [IDE Mode].

The list of the microcontrollers of the series selected with [Device Family] is displayed.

If this button is clicked while “Open Project” or “Open Project & LM Download” or “Template Project & LM Download” is selected for [IDE Mode], the list is not displayed, and an error occurs.

*14... CS+/e^2 studio is started to get the latest information from CS+/e^2 studio, but it is automatically terminated. The information is updated at the time of the CS+/e^2 studio termination.

*15... The [Byte Order] list is valid to set when [Device Family] is set to “RX”.

The [Byte Order] is set to [Little Endian] and cannot be changed when [Device Family] list is set to “RL78” or “RA”.

*16... When [Build Tool] is set to “Renesas Compiler” or “GCC ARM Embedded”, the build tool is decided by CS+/e^2 studio at the project creation time.

*17… [Project Type] is enabled only when RA Device Name that supports TrustZone® is selected. In other cases, the [Project Type] value will change to the default value on the table above automatically.

18… [Smart Bundle] is used only for RA TrustZone® Non-Secure project type, which needs to refer to the smart bundle (.sbd) file of RA TrustZone® Secure project type. So, need to specify [Smart Bundle] by smart bundle (*.sbd) file of RA TrustZone® Secure project type to create RA TrustZone® Non-Secure project successfully.

To specify [Smart Bundle], get (*.sbd) file in RA TrustZone® Non-Secure which have created in Section 3.2.1.1Prepare debug configuration for Non-secure with Secure Bundle for RA family.

*19… When the [Device Family] is different from “RA” or [Project Type] is different from “TrustZone® Non-Secure”, if the [Select Smart Bundle] button is clicked, an error message displays.

*20… These options are used to support RA TrustZone® project type, which only available on e^2 studio 2024-07

*21... The connection to real target device (LSI device) by any of E1 connection types through Embedded Target is same as manual connection on CS+/e^2 studio GUI. For an effective usage, please create a dummy project for the same target on CS+/e^2 studio. Then, connect it to the target device through available emulator connection types. Once you get success, apply the same emulator connection type and Main Clock Frequency value to Embedded Target for PIL Simulation.

COM Port only supports for RL78/G23 series.

J-Link supports for RA device family and some devices of RX and it can be used if [IDE Target] is e^2 studio.

This list is available to set when [Device Family] list is set to “RX” or “RL78” or “RA” and [IDE Mode] is set to “Create Project”.

The default E2 Emulator type will be used in corresponding CS+/e^2 studio packages.

*22… [Measure Execution Time] does not support for some cases. Refer to "1.3 Can not support measuring execution time” in RESTRICTIONS.

*23… EZ emulator will not support RX device family on CS+ V8.09.00 / e^2 studio 2023-01 and later.

*24… This textbox is valid for changing when [Debug Tool] list is set to any of “E1(JTAG/Serial)/E2/E2 Lite/E20 (JTAG/Serial)/EZ” connection types.

*25… When clicking on this button, the dialog box displays and shows list of requiring features. Free-of-charge features are not showed in this box. These can be used freely.

*26… Currently, Embedded Target does not support run PIL Simulation with “Software Trace”

Remark When directly entering the installation directory path for each tool:

1. “¥” must not be used. Use “/” instead.
2. Do not let the “/” symbol at the end of installation directory path.

6. Close the Configuration Parameters dialog box.
   Check the settings and then click the [OK] button.

3.2.1.4 Generating a test environment

This section explains how to execute generation of the test environment required for PIL simulation when using a Subsystem block.

Embedded Target provides the following command, which can be used in the MATLAB® command window. When a Subsystem block is used, this command automatically executes a series of operations for generation of a test environment.

Table 3-4. Provided Command

Command name	Description
ecpils_build	Generates a test environment (only when a Subsystem block is used)

There is only way to execute generation of a test environment.

Select a Subsystem block in the DataTypes window and then use the following method to execute generation of a test environment.

Execute generation from the command window.

Execute generation of a test environment by entering the ecpils_build command provided by Embedded Target in the MATLAB® command window, using the following syntax.

Here ">>" denotes the command prompt and "[Enter]" denotes entry of the Enter key.

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

Remarks

When executing generation of a test environment by using this method, the following operations are also carried out. Therefore, it is not necessary to perform the operation described in section “Replacing blocks for PIL sequential execution for PIL simulation”.

The model file, including the selected Subsystem block, is copied (the destination model file has the same name as the original model file but "_ecpils" suffix is added).

The Subsystem block is replaced with the block for PIL sequential execution for the model file to be copied. Save the modified model file before execution of ecpils_build command.

3.2.1.5 Replacing blocks for PIL sequential execution for PIL simulation

When generation of a test environment is carried out from the MATLAB® command window, block replacement is carried out as a series of operations for generating a test environment. Accordingly, explicit operation by the user is not necessary.

The following explains how to replace blocks when generation of a test environment is carried out from the DataTypes window. In this case, this operation must be performed by the user.

- How to replace blocks

  Delete the Subsystem block "DataTypes" in the DataTypes window and then drag the block for PIL sequential execution "DataTypes" from the untitled window and drop it in the corresponding position in the DataTypes window. This generates the PIL simulation model.

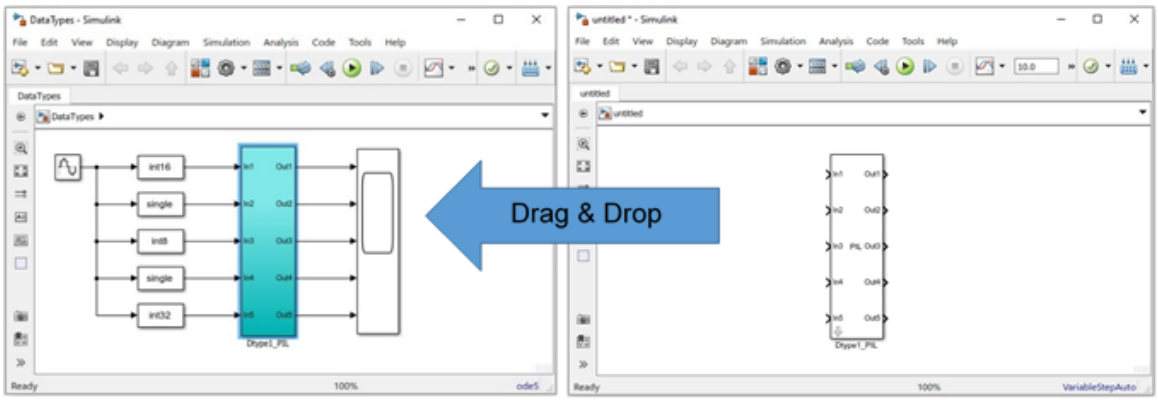

Figure 3-10. Replacing a Block for PIL Sequential Execution

Remarks 1. After blocks for PIL sequential execution have been generated, the block for PIL sequential execution created on the untitled model file is replaced with the Subsystem of the base model. The untitled model can be closed without being saved.

2. Multiple blocks for PIL sequential execution cannot be assigned in the model.

3.2.1.6 Generating a load module

When “Open Project & LM Download” or “Template Project & LM Download” is set for [IDE Mode] in the [Embedded Target Options] of the Configuration Parameters dialog box, generation and downloading of a load module is carried out automatically as a series of the operations for generating a test environment. Accordingly, explicit operation by the user is not necessary.

When “Create Project” is set for [IDE Mode] in the [Embedded Target Options] of the Configuration Parameters dialog box, on the first generate, please wait until the successful project creation dialog appears as shown below.

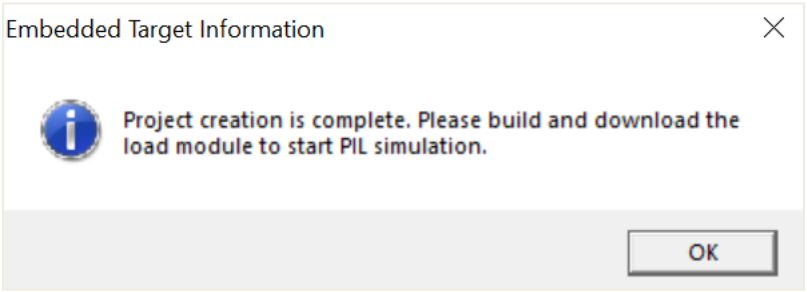

Figure 3-11. Successful project creation dialog

The following explains how to generate a load module when [Create Project] or [Open Project] is set for [IDE Mode].

- How to generate a load module
  [When using CS+ IDE]
  (1) Make option settings for a build tool (such as compiler and assembler) in the Property panel of CS+. Note that the default settings of CS+ should be changed in the following cases.

  [When using big endian in RX]
  Select [Build Tool] in the CS+ project tree and set [Big-endian data (-endian=big)] for [Common Options] - [CPU] - [Endian type for data].
  Here, a message dialog box saying, “Also change the endianness of the debug tool?” appears. Select “Yes”.

  [When using RL78]
  Select [Build Tool] in the CS+ project tree and set value for (*) [Compile Options] – [Output Code] – [Process double type / long double type as float type] to “No(-dbl_size=8)”. And set value for [Link Options] – [Device] – [Option byte values for OCD] and [User option byte value]. And set [Yes (Specify address range) (-DEBUG_MONITOR=<Address range>)] for [Link Options] – [Device] – [Set debug monitor area]. These values are described in user’s hardware manual of each device family.
  If [Debug Generated Code during PIL Simulation] in [Configuration Parameter] – [Embedded Target Options] is checked, set value for [Compile Options] – [Optimization (Details)] - [Perform inline expansion] to “No(-Oinline_level=0)”.

  (*): Only set this option when device is RL78 S3-core and use higher 2-bytes data type.

  (2) Save the CS+ project.

  (3) Select [Build Project] from the [Build] menu of CS+.

[When using e² studio IDE]

(1) Make option settings for a build tool in the Property panel of e² studio.

[When using RL78 in e² studio IDE]

Right click to project choose [Properties], select [C/C++ Build] > [Settings] to open Setting dialog of Build Tool. In [Compiler] – [Output Code], uncheck for (*) [Process double type / long double type as float type (-dbl_size)]. In [Linker] – [Device], check and set value for [Set user option byte (-user_opt_byte)], [Set enable/disable on-chip debug byte link option (-ocdbg)] and [Secure memory area of OCD monitor (-debug_monitor)]. These values are described in user's hardware manual of each device family.

If [Debug Generated Code during PIL Simulation] in [Configuration Parameter] – [Embedded Target Options] is checked, set value for [Compiler] – [Optimization] - [Perform inline expansion (-Oinline_level)] to "No".

[When using RA in e² studio IDE]

To ensure clock matches with device, please follow these steps.

(1) Double click "configuration.xml" in [Project Explorer] panel to open "FSP Configuration"

(2) Click [BSP] tab → [Board] → [Device]. Choose the appropriate board and device.

(3) Press [Generate Project Content].

(*): Only uncheck this option when device is RL78 S3-core and use higher 2-bytes data type.

(2) Build the project by one of the below ways:

- Right click on the project and select [Build Project]
- Click on the project to set focus and select [Project] → [Build Project]
- Click on the project to set focus and click on  icon
- Click on the project to set focus and press [Ctrl] + [B]

Remark For details on generating the load module, refer to "Build" of CS+/e² studio User's Manual.

3.2.1.7 Downloading a load module

When "Open Project & LM Download" or "Template Project & LM Download" is set for [IDE Mode] in the [Embedded Target Options] of the Configuration Parameters dialog box, downloading of a load module is carried out automatically as a part of a series of the operations for generating a test environment. Accordingly, an explicit operation by the user is not necessary.

The following describes how to make settings for a debug tool and how to download a load module when "Create Project" or "Open Project" is set for [IDE Mode].

1. For Single-Core project:

- How to make settings for a debug tool

  [When using E1/E2/E2 Lite/E20 (JTAG/Serial)/EZ Emulator or COM Port with RX, RL78 families on CS+ IDE]

  Select [Debug Tool] in the CS+ project tree and select [Yes] for [Debug Tool Settings] - [Access Memory While Running] - [Access during execution] (or [Access by stopping execution]) in property setting.

  [When using Big Endian in E2/E2 Lite/E20 (JTAG/Serial)/EZ Emulator with RX families on e² studio IDE]

  (1) Click "Tutorial" project in [Project Explorer] panel to set focus

(2) Click [Run] → [Debug Configurations...] or icon (downward arrow) → [Debug Configurations...] to open the “Debug Configurations” window.

(3) In “Debug Configurations” window, go to [Renesas GDB Hardware Debugging] → [Tutorial HardwareDebug]. Switch to the [Debugger] tab.

Under the [Debugger] tab, go to the [Debug Tool Settings] sub tab, select [Big Endian] for [Memory] – [Endian].

- How to download a load module

  [When using CS+ IDE]

  Make option settings for a debug tool (E1/E2/E2 Lite/E20 (JTAG/Serial)/EZ/J-Link Emulator or COM Port or a simulator) with the property panel of CS+ and then download a load module by selecting [Download] from the [Debug] menu of CS+.

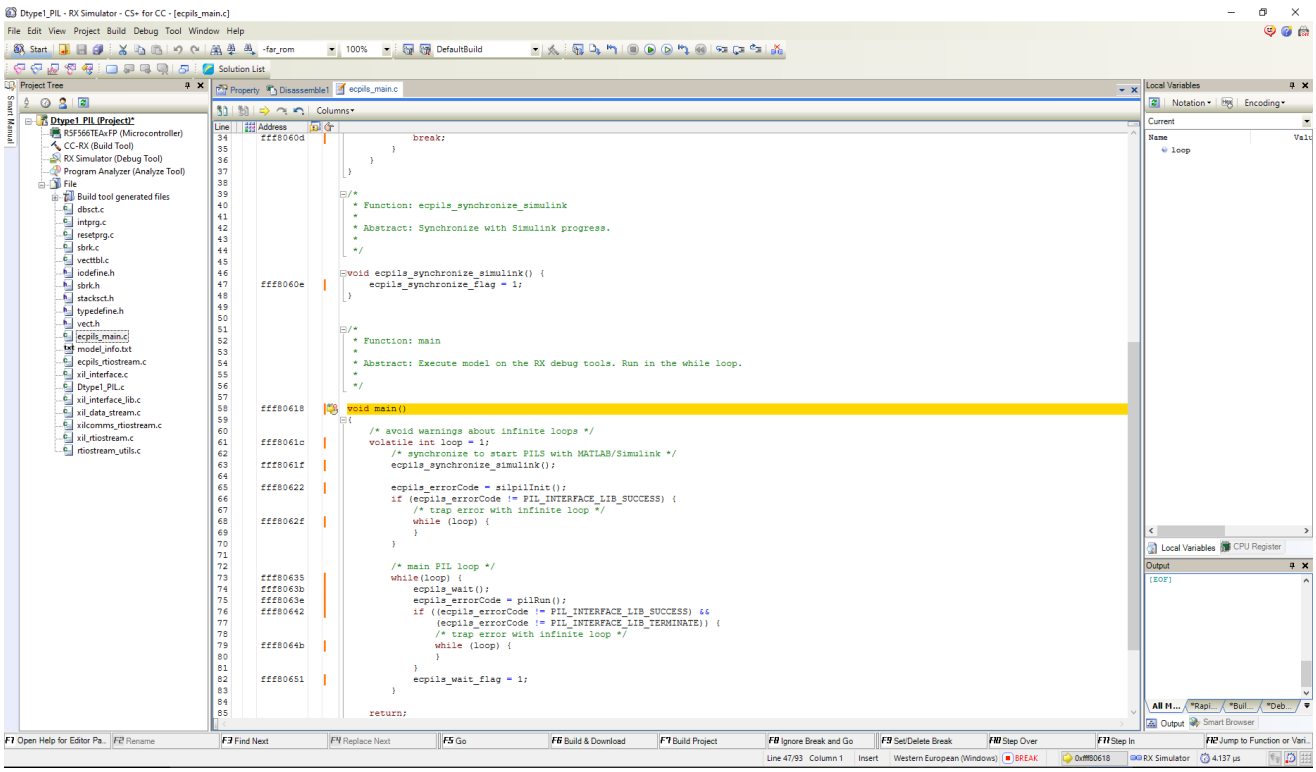

Figure 3-12. CS+ on Completion of Downloading to the Debugger

[When using e^2 studio IDE]

Click the target project in [Project Explorer] panel to set focus and click  icon to launch a debugger session and then download a load module.

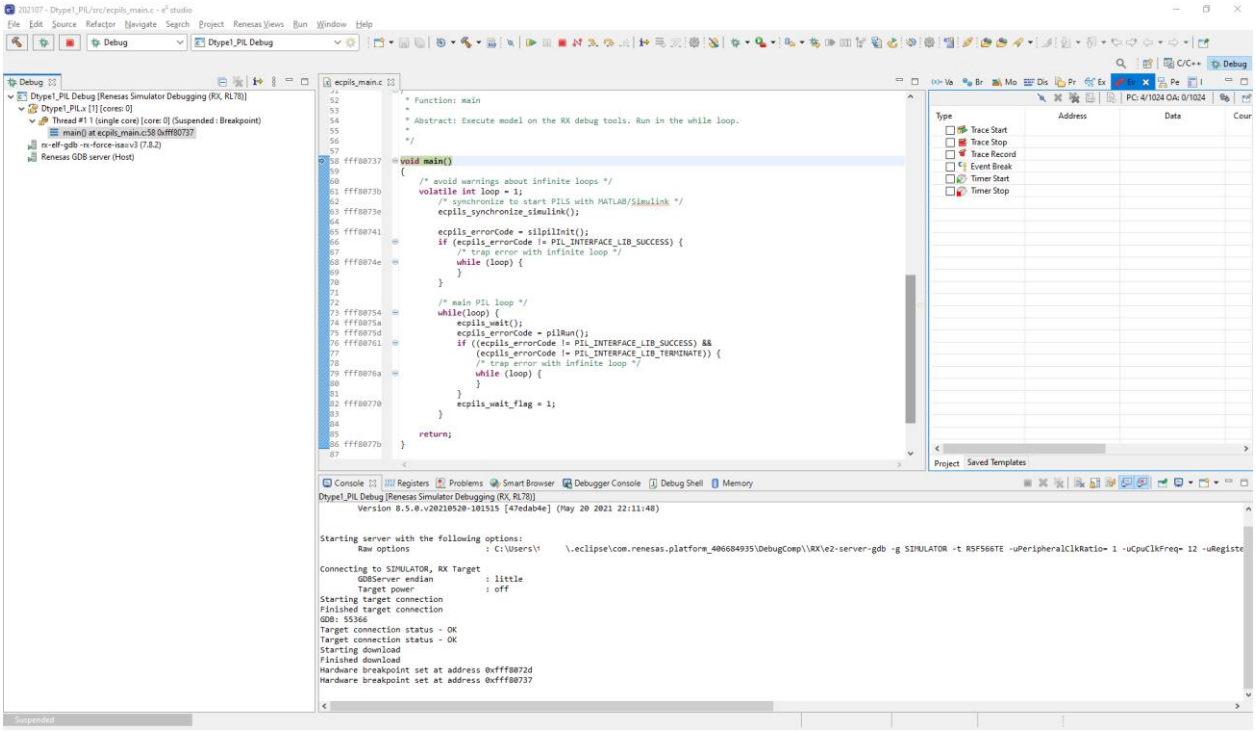

Figure 3-13. e^2 studio on Completion of Downloading to the Debugger

3.2.2 Executing PIL Simulation

The following explains how to run PIL simulation when generation of a test environment is carried out from the DataTypes window.

- How to execute PIL simulation

  Verify that the information in the DataTypes window has changed to that of the PIL simulation model. Then select [Run] from the [Simulation] menu to start PIL simulation.

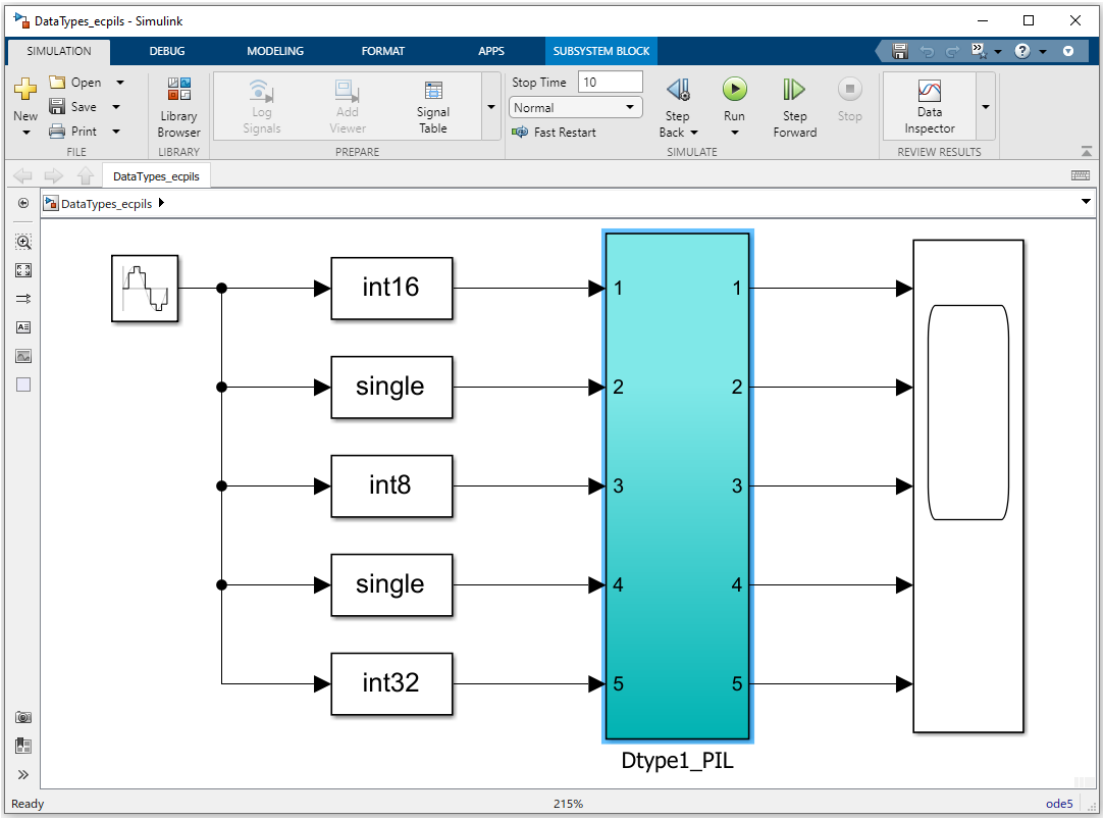

Figure 3-14. PIL Simulation Executions

Remark If you save the model file after executing this operation, the original model file is overwritten.

The following confirmation dialog box is displayed until Embedded Target confirms that downloading is completed. The dialog box is closed automatically after completion of download. To stop downloading and suspend a series of operations, click the OK button in the dialog box.

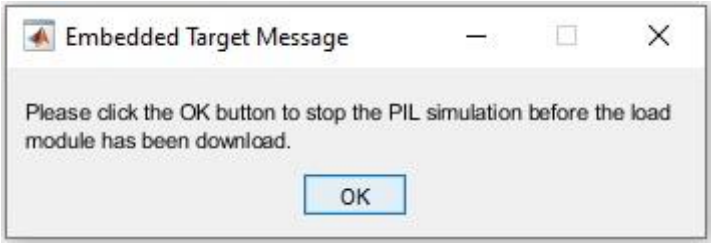

Figure 3-15. Confirmation Dialog Box

3.2.3 Debugging Generated Code during PIL Simulation

This section describes how to debug generated code from embedded models during PIL Simulation. In this, you can use all debugging features offered by CS+/e^2 studio's Debug Tools such as:

- Step 1: Open model, select Code Generation Target Subsystem and open Model Configuration Parameters window.
- Step 2: Setting all necessary conditions and check the [Debug Generated Code during PIL Simulation] checkbox.
- Step 3: Save model and execute MATLAB® command: ">> ecpils_build [Enter]"
- Step 4: Build and Download CS+/e^2 studio project. There are two breakpoints that set at main() and *ecpils_synchronize_Simulink()* functions of "ecpils_main.c" file automatically.
- Step 5: Hit [Run] button in Simulink® Model and start debugging:
  - Step-by-step go through each instruction
  - Step into a code block, function
  - Stop CPU
  - Etc.

To enable this mode, please check the [Debug Generated Code during PIL Simulation] during the Setting configuration parameters procedure. Bellow figure shows sample of setting on Embedded Target Options GUI:

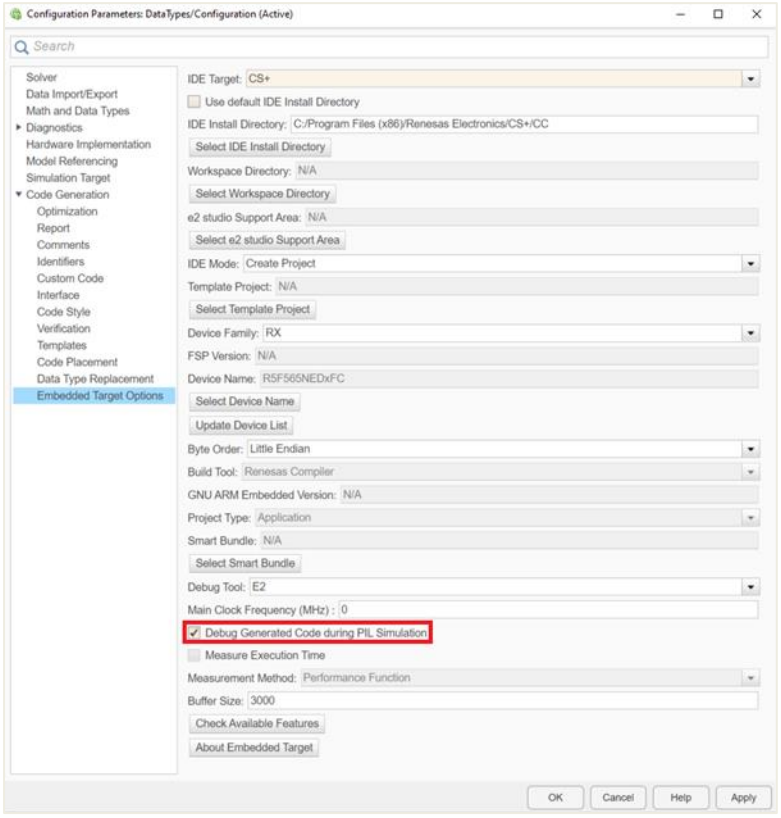

Figure 3-16. Enable Debug Generated Code during PIL Simulation

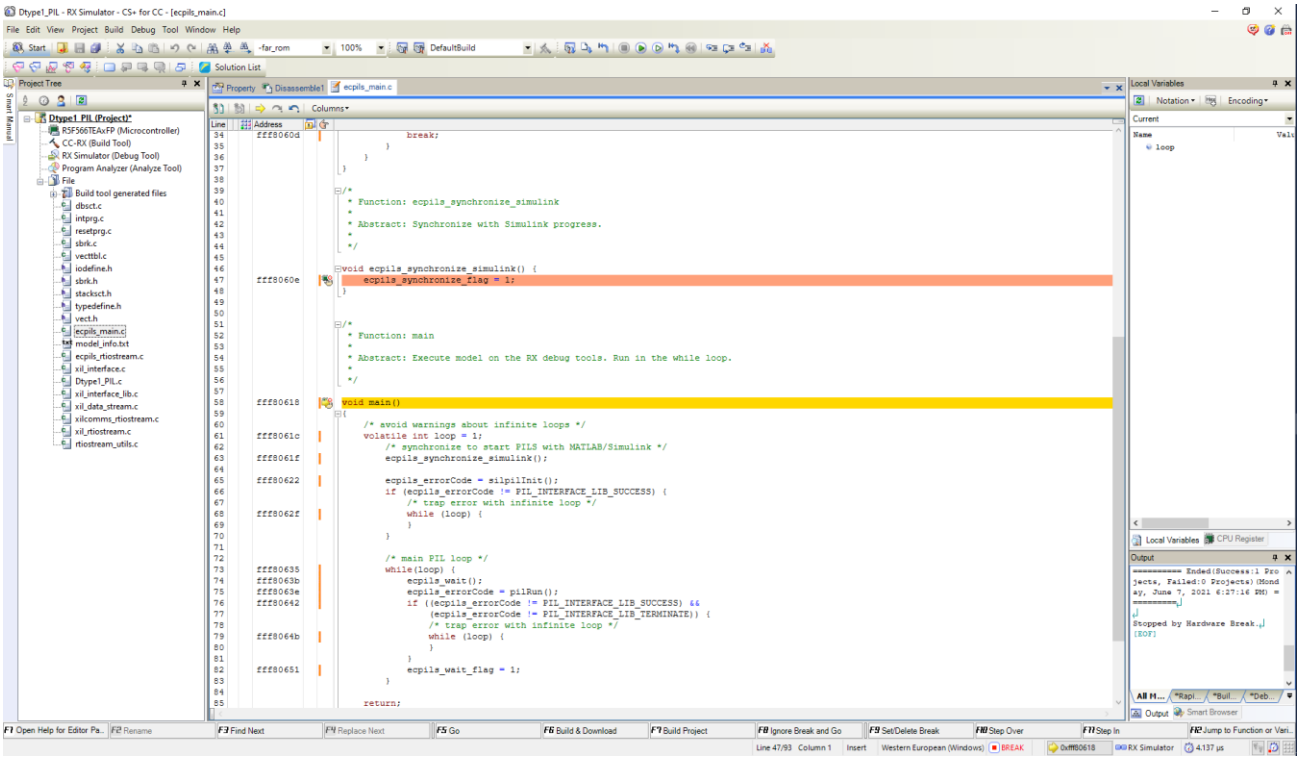

Figure 3-17. Two automatic breakpoints in “ecpils_main.c” file in CS+

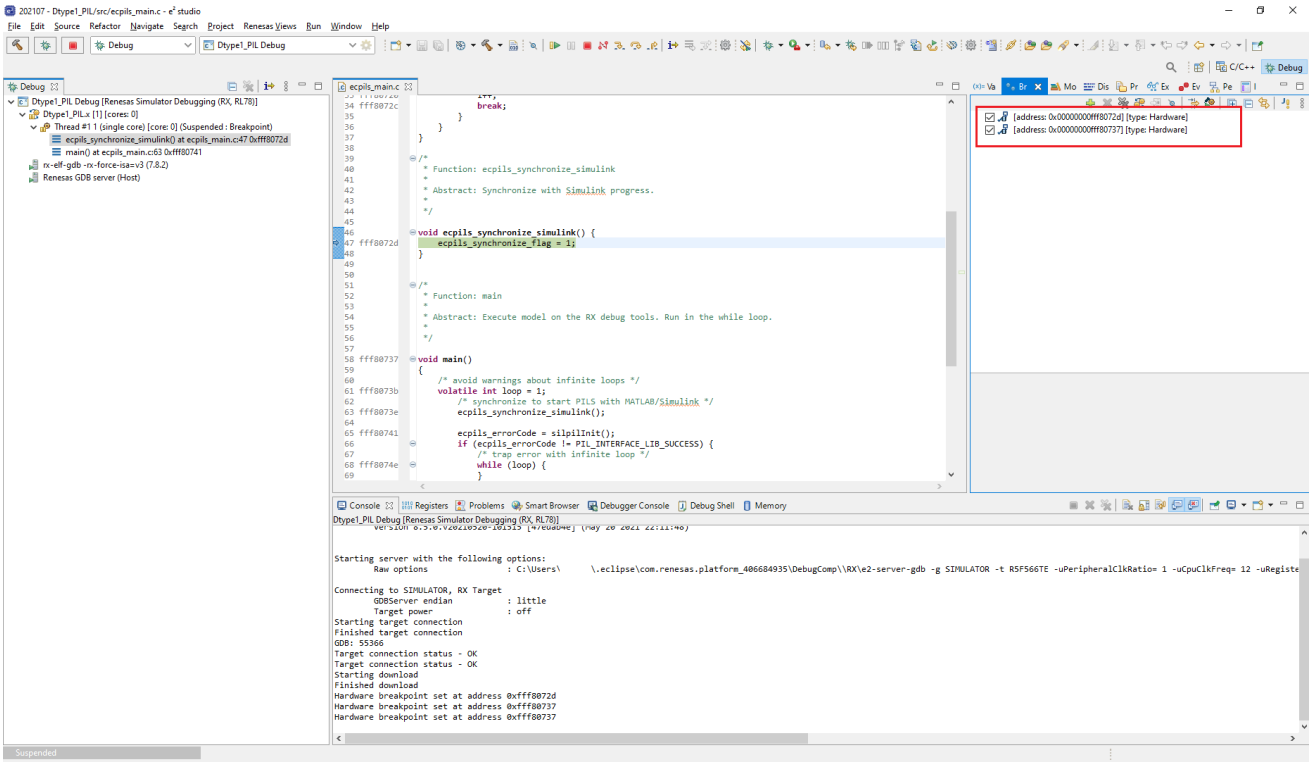

Figure 3-18. Two automatic breakpoints in “ecpils_main.c” file in e² studio

Remarks 1. When using this function, you cannot measure the execution time. As a result, the [Measure Execution Time] checkbox is disabled.

2. The timeout for debugging 1 step is 24 hours as maximum. When the timeout exceeds, PIL Simulation process is stopped.

3. When the program on target side (CS+/e^2 studio) is stopped. The Simulink® GUI is also frozen, it means cannot pause or stop the Simulation on the Simulink® GUI. This is a limitation of MATLAB®/Simulink®. To stop the PIL Simulation, remove breakpoints (if added) in the debugging process, then let the code run through and finish the Simulation.

4. Do not disconnect the Debug Tool or end the CS+/e^2 studio process during debugging. This will cause unexpected behaviors.

5. Do not change the workflow of PIL Simulation on the CS+/e^2 studio side such as: modify Embedded Target generated code, change value of Program Counter, etc., such actions make PIL Simulation process operates abnormally and some error may display.

6. When using this function with RL78 device family, setting for [Perform inline expansion] follow 3.2.1.6 Generating a load module.

3.2.4 Re-executing Embedded Target

Re-execute Embedded Target with either of the following steps.

- Using the ecpils_build command (when the Simulink® model is updated or code generation is re-executed)

  (1) Terminate the <model name>_ecpils.slx window, of which model is for Simulink® and has been previously executed and CS+/e^2 studio.

  (2) Remove the slprj folder that has been generated at last execution.

  (3) Run the ecpils_build command in the MATLAB® command window.

  (4) Run simulation from the Simulink® window.

- PIL simulation only (when the Simulink® model is not updated and code generation is omitted)

  (1) Exit the <model name>_ecpils.slx window, of which model is for Simulink® and has been last executed and CS+/e^2 studio.

  (2) Start the <model name>_ecpils.slx window and run simulation.

  (3) CS+/e^2 studio is started. When [Create Project] or [Open Project] is set for [IDE Mode] of [Embedded Target Options] in the Configuration dialog box, start downloading.

  (4) When CS+/e^2 studio finished downloading, PIL simulation is automatically started.

3.2.5 Cleanup Embedded Target workspace after PIL Simulation

Cleanup Embedded Target workspace with either of the following steps:

- Manually delete the following folders and files or using "ecpils_cleanup" command for automatic clean up (if necessary):
  - Folders: +ecpils, < ModelName >_ecpils, slprj
  - Model: < ModelName >_ecpils.slx

3.3 Executing PIL Simulation for Reference Code Generation Target Model

The following describes how to generate a test environment necessary for PIL simulation when the target for code generation is a reference model.

3.3.1 Generating a Test Environment

This section explains how to generate a test environment necessary for PIL simulation.

The explanation uses sample models DataTypesReference.slx and DataTypesRef.slx provided with Embedded Target.

3.3.1.1 Prepare debug configuration for Non-secure with Secure Bundle for RA family

Refers to Section 3.2.1.1 Prepare debug configuration for Non-secure with Secure Bundle for RA family.

3.3.1.2 Preparing a model using a reference model

Use sample models DataTypesReference.slx and DataTypesRef.slx.

DataTypesReference.slx references DataTypesRef.slx. DataTypesRef.slx is not directly used.

3.3.1.3 Setting configuration parameters

Embedded Target checks or sets options for code generation. The same settings must be made by a set of DataTypesReference.slx and DataTypesRef.slx.

1. Open the Configuration Parameters dialog box

   Select [Model Configuration Parameters] from the [Simulation] menu in the DataTypesReference window to open the Configuration Parameters dialog box.

2. Set [Hardware Implementation] options

   Select [Hardware Implementation] in the [Select] area and make the settings described below. Then click the [Apply] button.

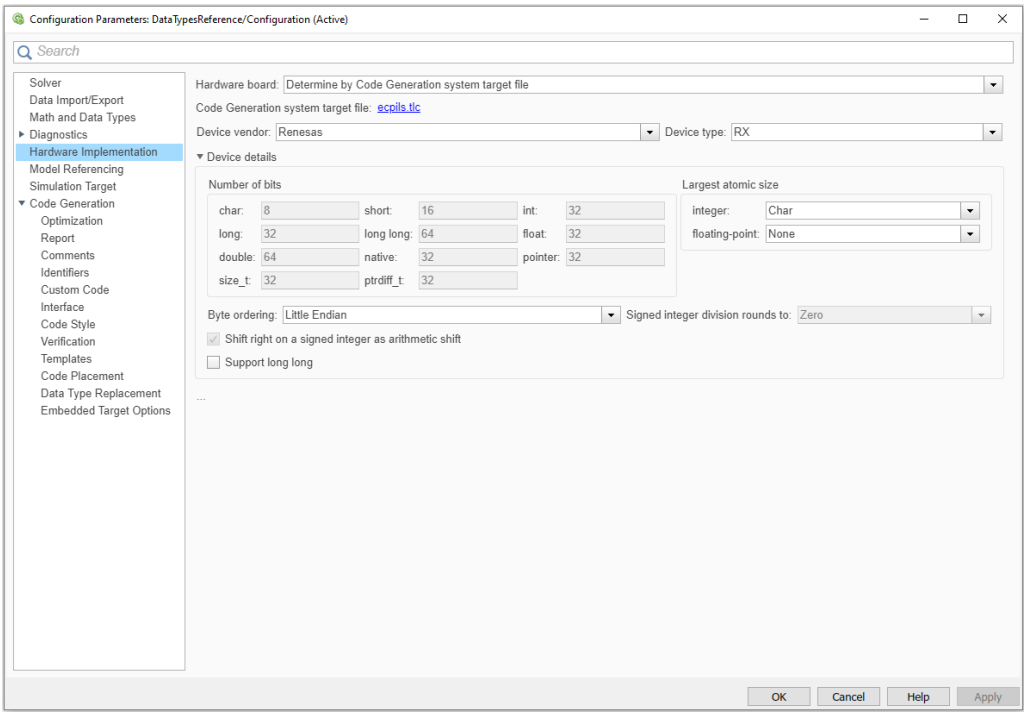

Figure 3-19. [Hardware Implementation] Options

Remark 1. Device type value in [Hardware Implementation] panel can be changed by setting Device Family value in [Embedded Target Options] panel. The change goes into effect when pressing “OK” or “Apply” button on [Embedded Target Options] panel.

2. The setting of Device Family value is enabled when IDE Mode value is [Create Project] value in [Embedded Target Options] panel.

3. Set [Model Referencing] options

Select [Model Referencing] in the [Select] area. Then make the setting of [Rebuild] and click the [Apply] button.

When [Always] is selected, code generation is performed regardless of the change of the Simulink® model. However, when [If any changes detected] or [If any changes in known dependencies detected] is selected and the change of the Simulink® model is not detected, code generation is omitted. Note that [Never] must not be set.

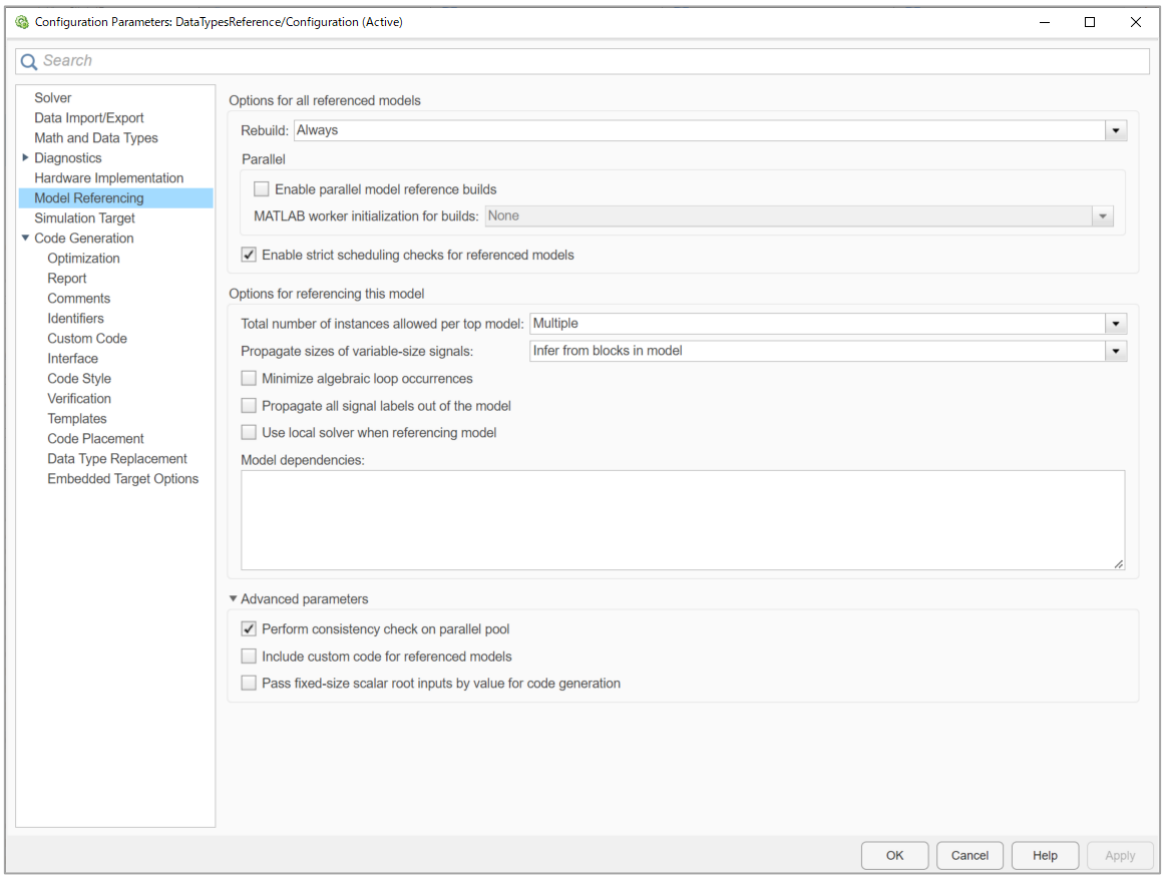

Figure 3-20. [Model Referencing] Options

4. Set [Code Generation] options

   Select [Code Generation] in the [Select] area. Then make the settings shown in the figure below and click the [Apply] button.

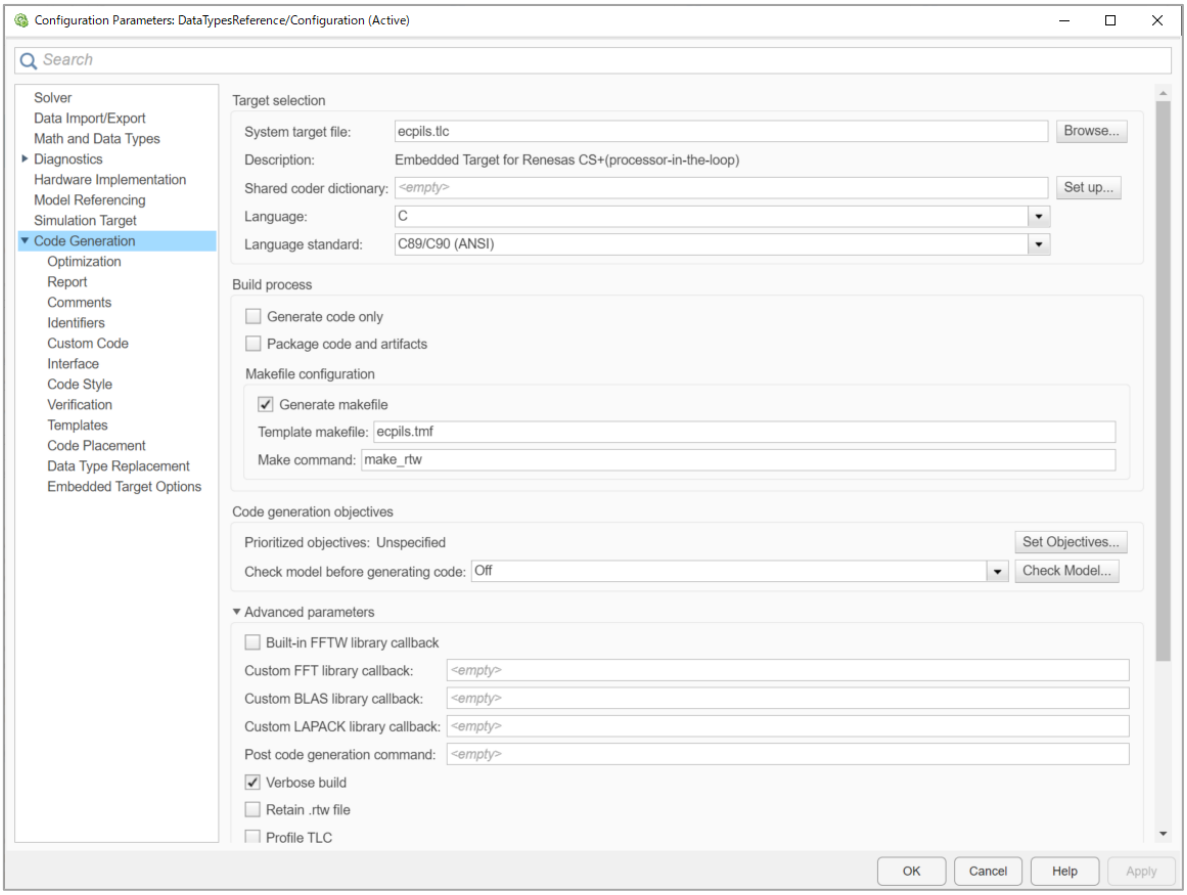

Figure 3-21. [Code Generation] Options

Remark The template make file (ecpils.tmf) will be overwritten according to selected Mex compiler (>>mex -setup).

5. Set [SIL and PIL Verification] or [Verification] options

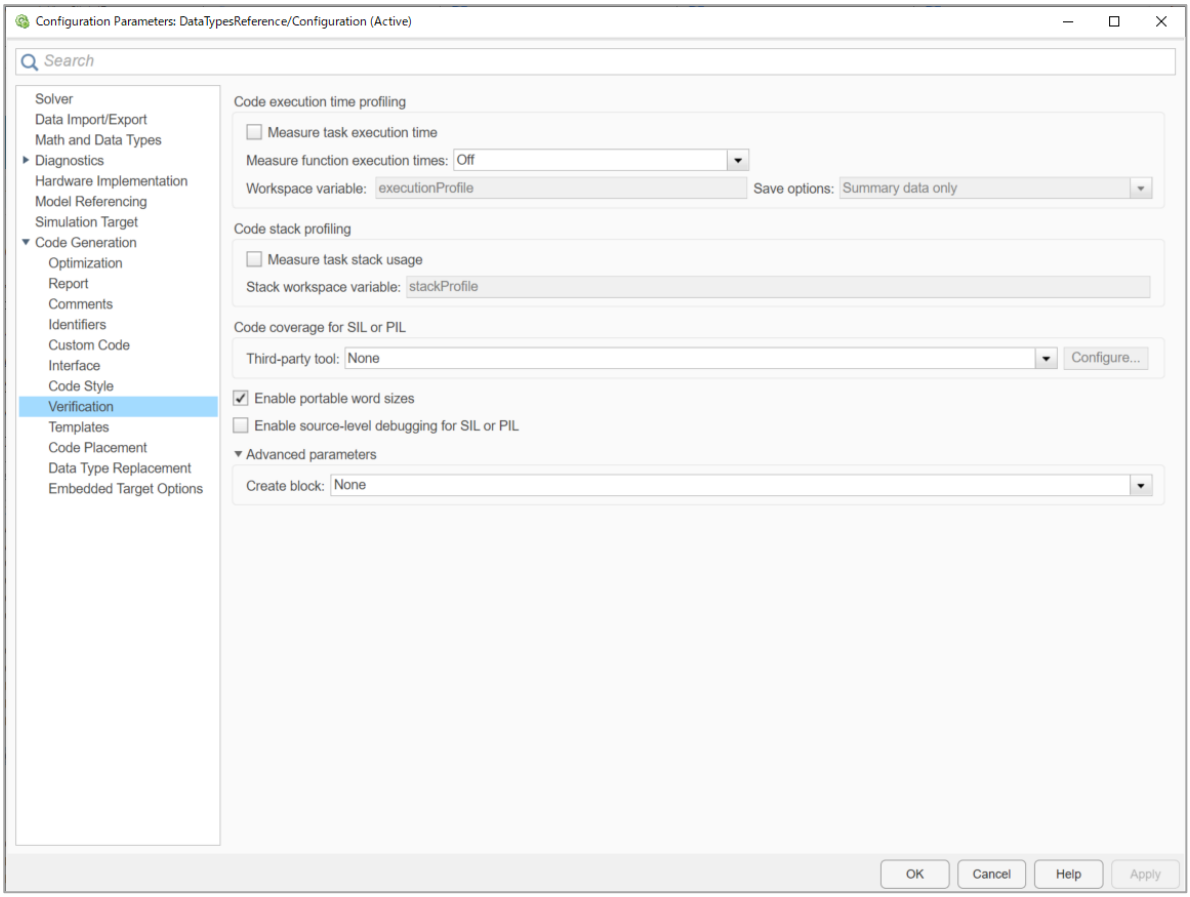

Figure 3-22. [Verification] Options

6. Set [Embedded Target Options]

   Use the same setting as when a Subsystem block is used. Refer to section “Setting configuration parameters ” /” (6) Set [Embedded Target Options]”.

7. Close the Configuration Parameters dialog box

   Check the settings and then click the [OK] button.

8. Set the configuration parameters for the DataTypesRef model

   Double-click the target Model block for code generation and open the DataTypesRef window.
   Select [Code Generation] - [Options] from the [Tools] menu in the DataTypesRef window and then make the same settings described in procedures (2) to (6).

3.3.1.4 Specifying PIL mode

Specify the PIL mode for the target model for code generation.

1. Open the Function Block Parameters dialog box

   Select the target Model block for code generation and right-click it to select [Block Parameters (ModelReference)]. Then the Function Block Parameters dialog box will open.

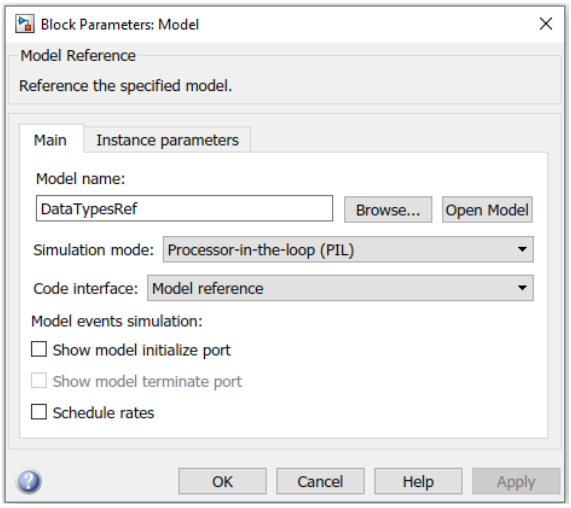

Figure 3-23. Function Block Parameters Dialog Box

2. Select PIL mode

   Select Processor-in-the-loop (PIL) as a Simulation mode.

3.3.1.5 Executing PIL Simulation

Verify that the information in the DataTypesReference window has changed to that of the PIL simulation model. Then select [Run] from the [Simulation] menu to start PIL simulation. Code generation and start-up of CS+/e^2 studio are performed in preparation for PIL simulation.

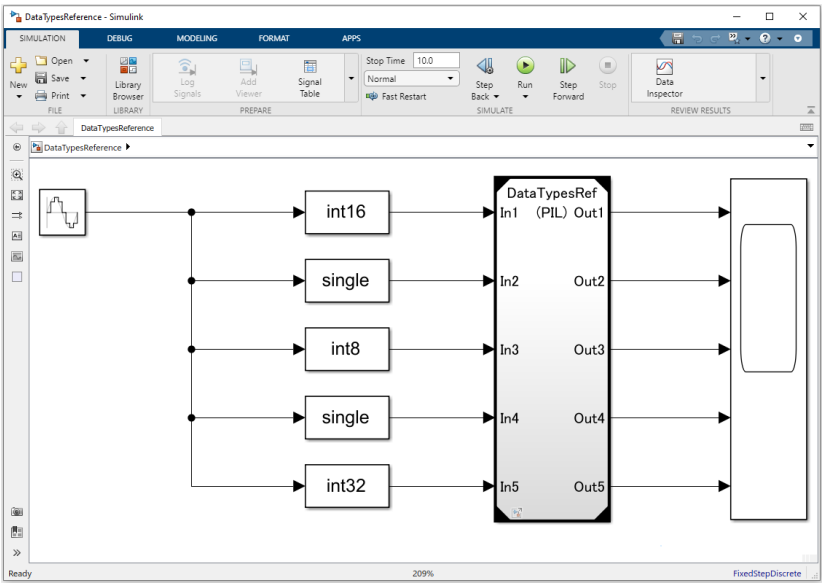

Figure 3-24. PIL Simulation Executions

Remark If code generation is performed after this operation, CS+/e^2 studio is started. The storage file for execution time measurement result will be removed.

3.3.1.6 Generating a load module

The steps of Generating a load module for Reference model are the same as the steps of generating for Subsystem block. Refer to section “Generating a load module”.

3.3.1.7 Downloading a load module

CS+/e^2 studio is started up. When “Create Project” or “Open Project” is set for [IDE Mode] in the [Embedded Target Options] of the Configuration dialog box, build and download a load module by following the procedure.

When “Open Project & LM Download” or “Template Project & LM Download” is set, a load module is built and downloaded to the debugger automatically. PIL simulation starts on completion of downloading.

- How to make settings for a debug tool

  [When using E1/E2/E2 Lite/E20 (JTAG/Serial)/EZ Emulator or COM Port with RX, RL78 families on CS+ IDE]

  Select [Debug Tool] in the CS+ project tree and select [Yes] for [Debug Tool Settings] - [Access Memory While Running] - [Access during execution] (or [Access by stopping execution]) in property setting.

  [When using Big Endian in E2/E2 Lite/E20 (JTAG/Serial)/EZ Emulator with RX families on e^2 studio IDE]

  (1) Click “Tutorial” project in [Project Explorer] panel to set focus

  (2) Click [Run] → [Debug Configurations...] or  icon (downward arrow) → [Debug Configurations...] to open the “Debug Configurations” window.

  (3) In “Debug Configurations” Windows®, go to [Renesas GDB Hardware Debugging] → [Tutorial HardwareDebug]. Switch to the [Debugger] tab.

  Under the [Debugger] tab, go to the [Debug Tool Settings] sub tab, select [Big Endian] for [Memory] – [Endian].

- How to download a load module

  [When using CS+ IDE]

  Make option settings for a debug tool (E1/E2/E2 Lite/E20 (JTAG/Serial)/EZ Emulator or COM Port or a simulator) with the property panel of CS+ and then download a load module by selecting [Download] from the [Debug] menu of CS+.

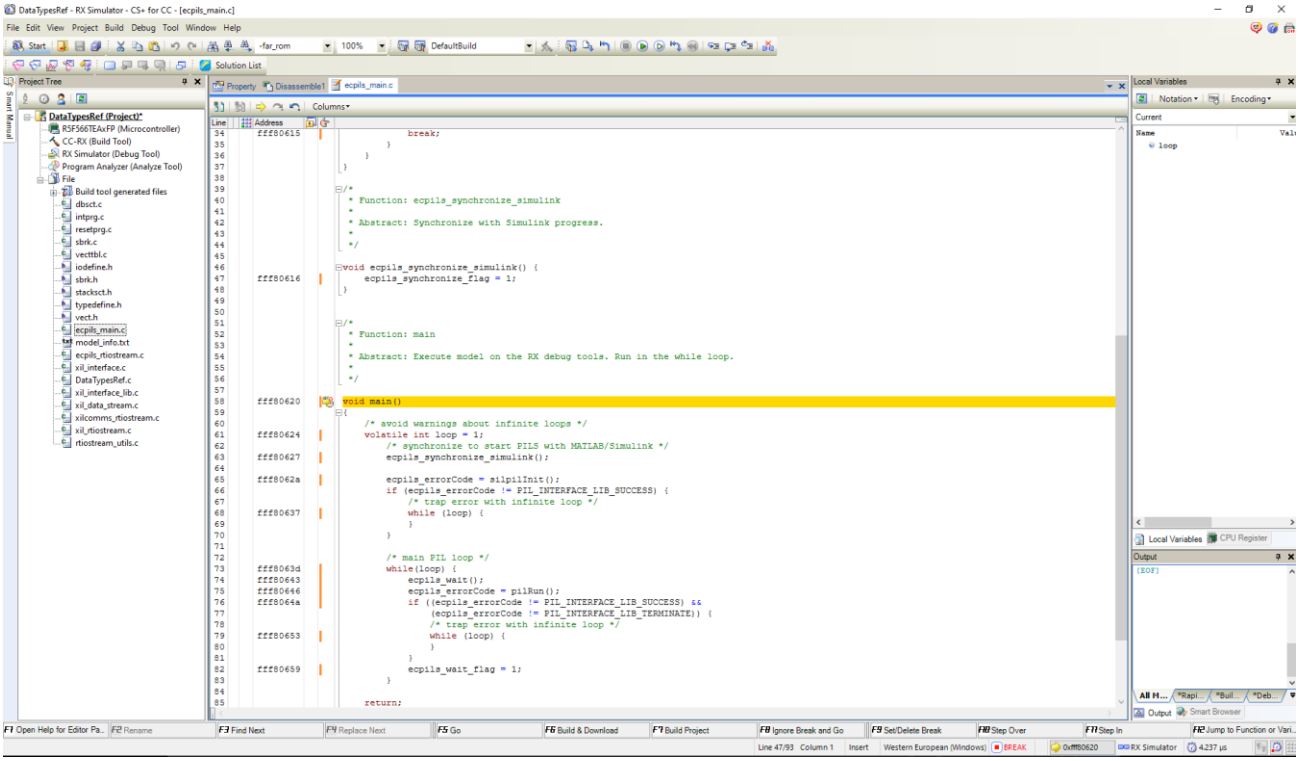

Figure 3-25. CS+ on Completion of Downloading to the Debugger

[When using e^2 studio IDE]

Click the target project in [Project Explorer] panel to set focus and click  icon to launch a debugger session and then download a load module.

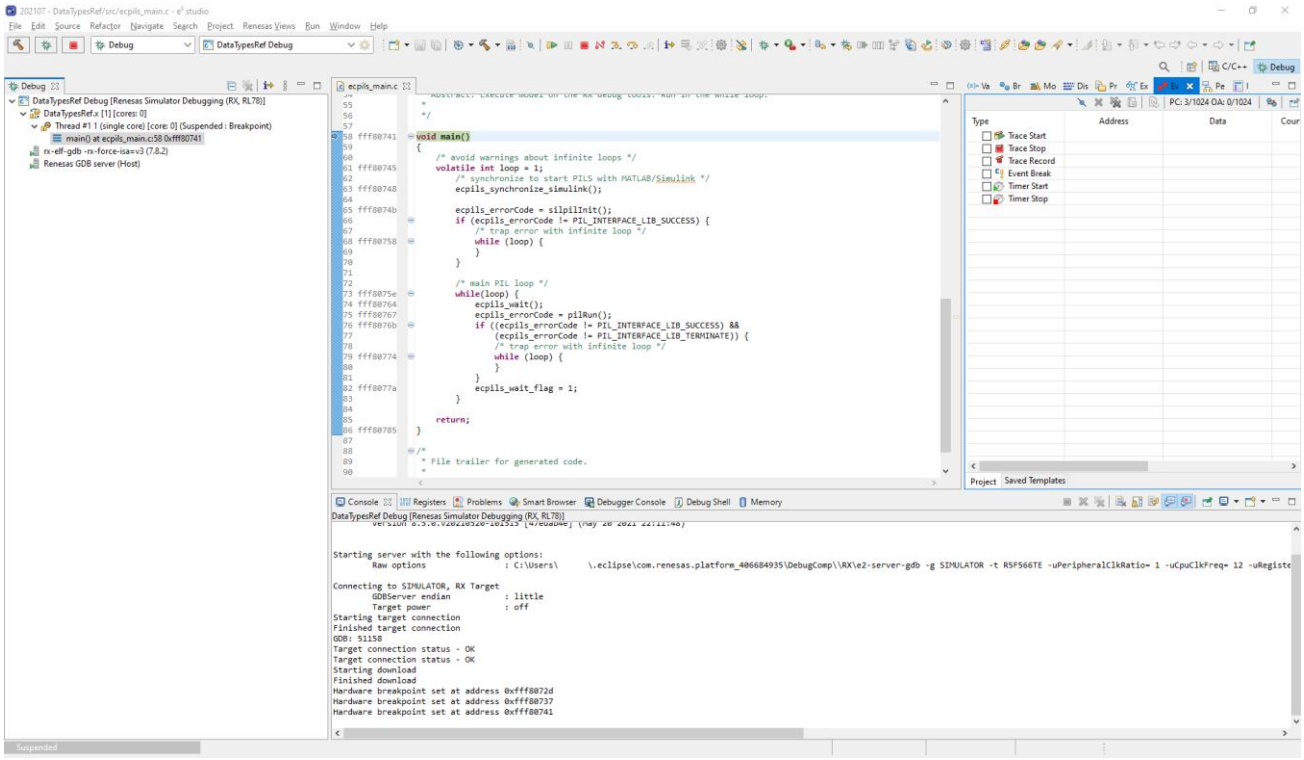

Figure 3-26. e^2 studio on Completion of Downloading to the Debugger

3.3.2 Debugging Generated Code during PIL Simulation

This section describes how to debug generated code from embedded models during PIL Simulation. In this, you can use all debugging features offered by CS+/e^2 studio's Debug Tools such as:

- Step 1: Open model, select Code Generation Target Subsystem and open Model Configuration Parameters Window.
- Step 2: Setting all necessary conditions and check the [Debug Generated Code during PIL Simulation] checkbox (setting for both Reference and Ref models).
- Step 3: Save model and hit [Run] button in Simulink® Model.
- Step 4: Build and Download CS+/e^2 studio project. There are two breakpoints that set at *main()* and *ecpils_synchronize_Simulink()* functions of "ecpils_main.c" file automatically.
- Step 5: Start debugging:
  - Step-by-step go through each instruction
  - Step into a code block, function
  - Stop CPU
  - Etc.

To enable this mode, please check the [Debug Generated Code during PIL Simulation] during the Setting configuration parameters procedure. Bellow figure shows sample of setting on Embedded Target Options GUI:

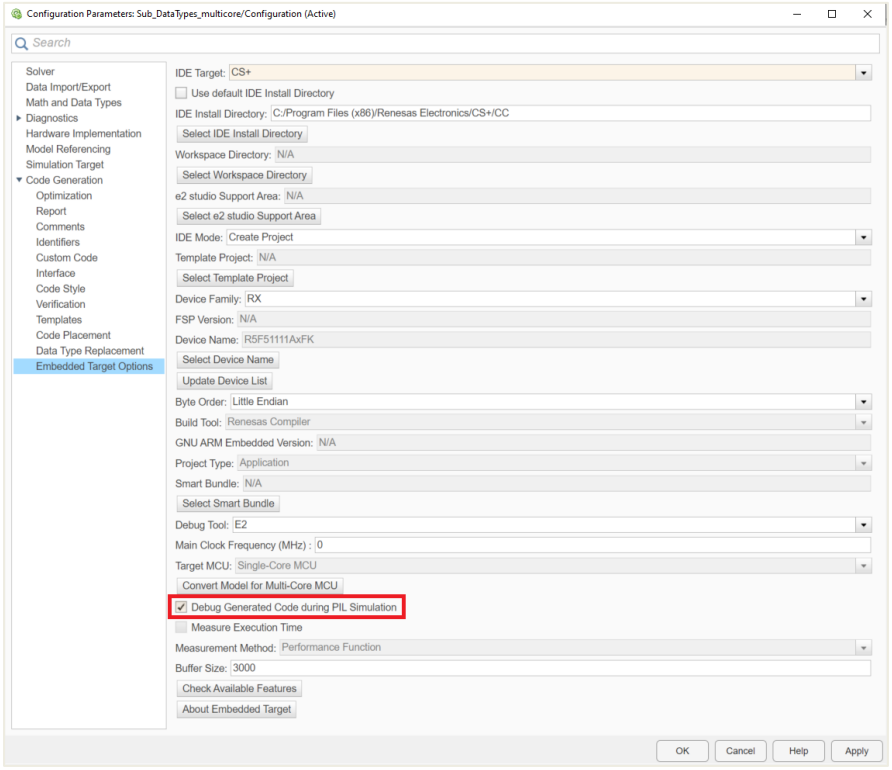

Figure 3-27. Enable Debug Generated Code during PIL Simulation

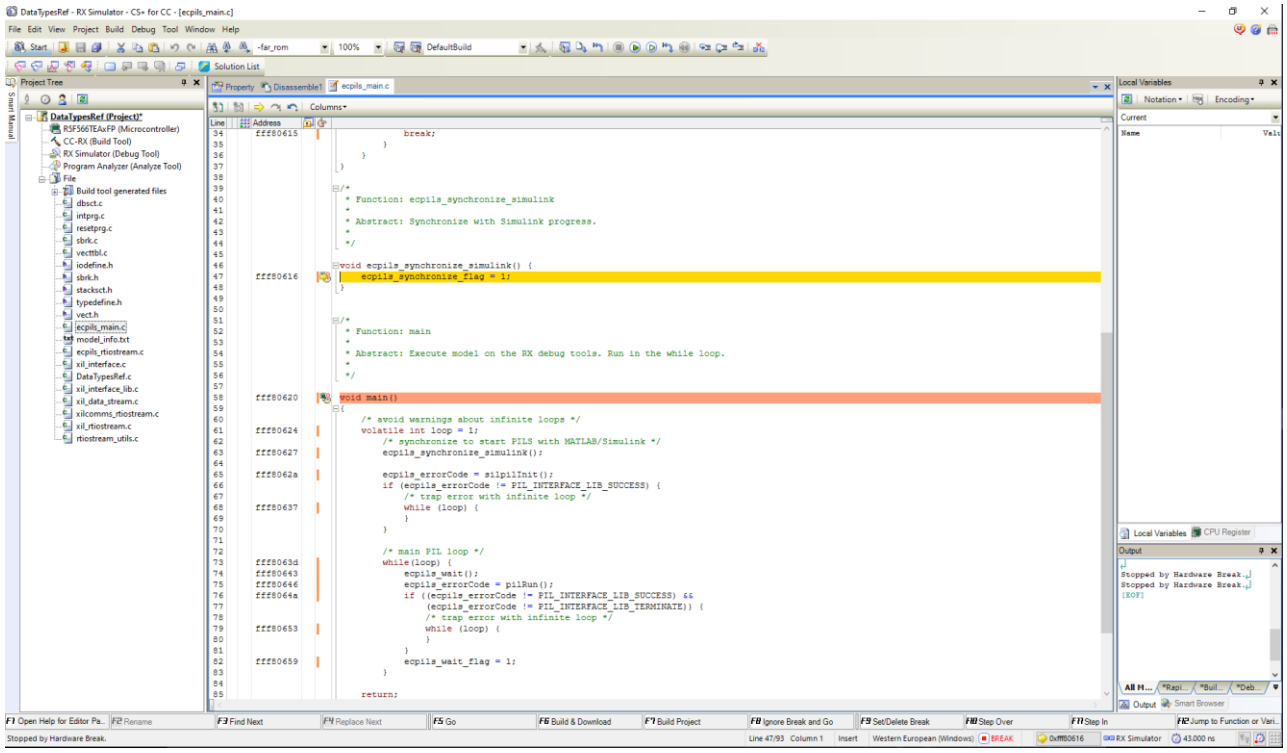

Figure 3-28. Two automatic breakpoints in “ecpils_main.c” file

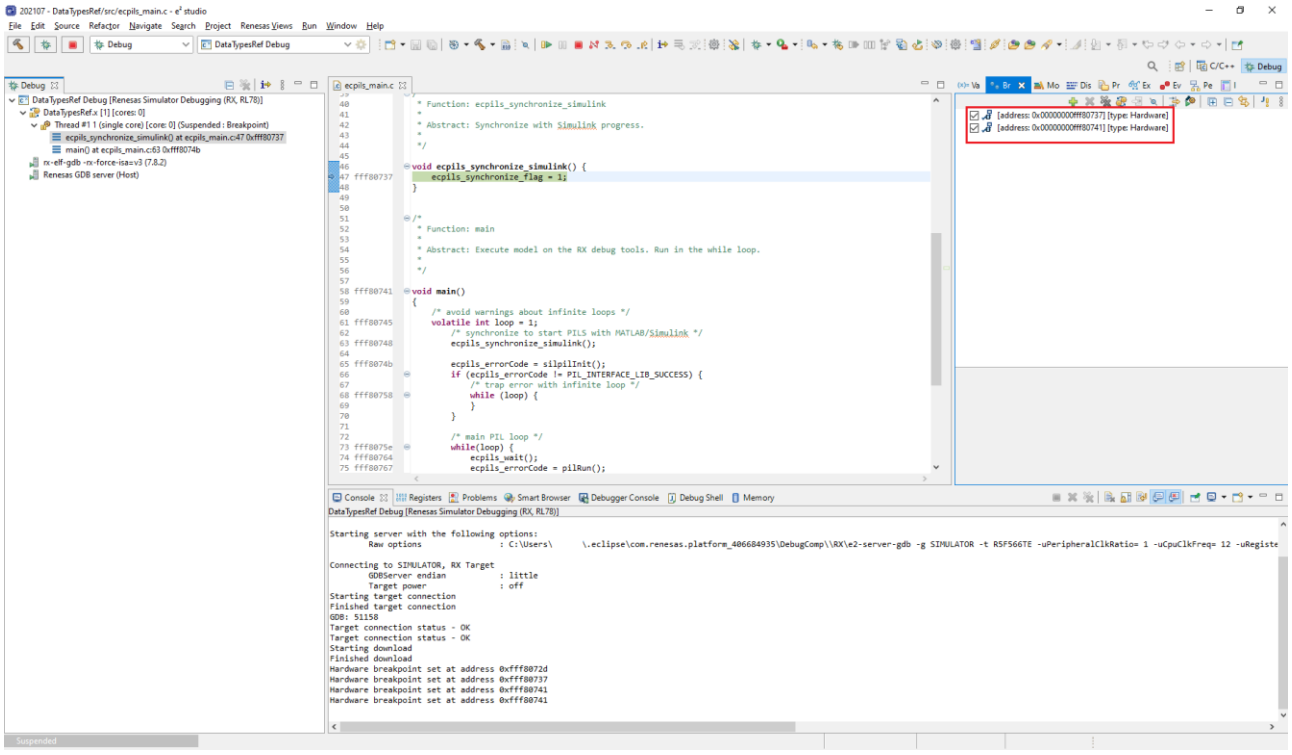

Figure 3-29. Two automatic breakpoints in “ecpils_main.c” file in e² studio

Remarks 1. When using this function, you cannot measure the execution time. As a result, the [Measure Execution Time] checkbox is disabled.

2. The timeout for debugging 1 step is 24 hours as maximum. When the timeout exceeds, PIL Simulation process is stopped.

3. When the program on target side (CS+/e^2 studio) is stopped. The Simulink® GUI is also frozen, it means cannot pause or stop the Simulation on the Simulink® GUI. This is a limitation of MATLAB®/Simulink®. To stop the PIL Simulation, remove breakpoints (if added) in the debugging process, then let the code run through and finish the Simulation.

4. Do not disconnect the Debug Tool or end the CS+/e^2 studio process during debugging. This will cause unexpected behaviors.

5. Do not change the workflow of PIL Simulation on the CS+/e^2 studio side such as: modify Embedded Target generated code, change value of Program Counter, etc., such actions make PIL Simulation process operates abnormally and some error may display.

6. When using this function with RL78 device family, setting for [Perform inline expansion] follow 3.2.1.6 Generating a load module.

3.3.3 Re-executing Embedded Target

Terminate the Simulink® model previously executed and CS+/e^2 studio. Then, start the Simulink® model and run simulation.

Operation differs according to the setting of [Rebuild] for [Model Referencing] in the Configuration dialog box. When [Always] is set, code generation is performed. When [If any changes detected] or [If any changes in known dependencies detected] is set, code generation is omitted if the Simulink® model is not updated.

3.3.4 Cleanup Embedded Target workspace after PIL Simulation

Cleanup Embedded Target workspace with either of the following steps:

- Manually delete the following folders and files or using "ecpils_cleanup" command for automatic clean up (if necessary):
  - Folders: +ecpils, slprj

3.4 Executing PIL Simulation for Top-level Code Generation Target Model

The following describes how to generate a test environment necessary for PIL simulation when the target for code generation is a top-level model.

3.4.1 Generating a Test Environment

This section explains how to generate a test environment necessary for PIL simulation.

The explanation uses a sample model DataTypesRef_Top.slx provided with Embedded Target.

3.4.1.1 Prepare debug configuration for Non-secure with Secure Bundle for RA family

Refers to Section 3.2.1.1 Prepare debug configuration for Non-secure with Secure Bundle for RA family.

3.4.1.2 Preparing a model using a top-level model

Use a sample model DataTypesRef_Top.slx.

3.4.1.3 Setting configuration parameters

Use the same setting as when a Subsystem block is used. Refer to section “Setting configuration parameters”.

3.4.1.4 Specifying PIL mode

Specify the PIL mode for the target model for code generation.

1. Select Processor-in-the-loop (PIL) mode
   Select [App] – [SIL/PIL Manager] and select [Processor-in-the-Loop (PIL)] from the [SIL/PIL] menu window.

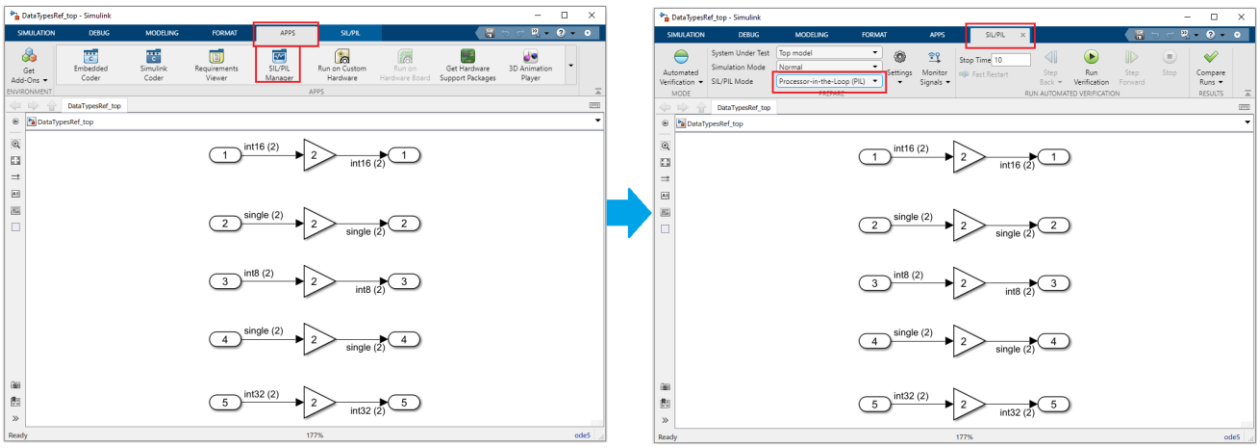

Figure 3-30. Simulation Menus for MATLAB® R2021b and above

2. Set a parameter
   Set a parameter of the model and a workspace variable. The .m file that is used to set a workspace variable is prepared in a sample model.

3.4.1.5 Executing PIL Simulation

Verify that the information in the DataTypesRef_Top window has changed to that of the PIL simulation model and that a workspace variable has been set. Then select [Run] from the [SIL/PIL] menu to start PIL simulation. Code generation and start-up of CS+/e^2 studio are performed in preparation for PIL simulation.

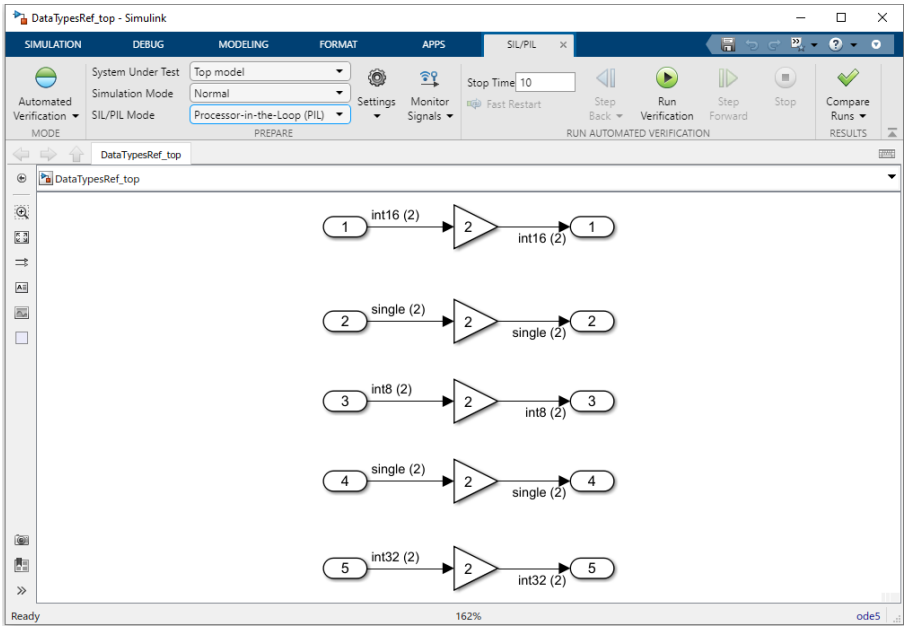

Figure 3-31. PIL Simulation Executions for MATLAB® R2021b and above

Remark If code generation is performed after this operation, CS+/e^2 studio is started. The storage file for execution time measurement result will be removed.

3.4.1.6 Generating a load module

The steps of Generating a load module for Top-level model are the same as the steps of Generating a load module for Subsystem block. Refer to section “3.2.1.6 Generating a load module”.

3.4.1.7 Downloading a load module

The steps of Downloading a load module during PIL Simulation used for Top-level model are the same as the steps of downloading used for Reference model. Refer to section 3.3.1.7 Downloading a load module.

3.4.2 Debugging Generated Code during PIL Simulation

The steps of Debugging Generated Code during PIL Simulation used for Top-level model are the same as the steps of debugging used for Reference model. Refer to section 3.3.2 Debugging Generated Code during PIL Simulation.

3.4.3 Re-executing Embedded Target

Terminate the Simulink® model previously executed and CS+/e^2 studio. Then, start the Simulink® model and run simulation.

When the Simulink® model is not updated and Embedded Target is re-executed, code generation is omitted. When the Simulink® model is updated, code generation is not omitted.

3.4.4 Cleanup Embedded Target workspace after PIL Simulation

Cleanup Embedded Target workspace with either of the following steps:

- Manually delete the following folders and files or using "ecpils_cleanup" command for automatic clean up(if necessary):
  - Folders: +ecpils, < ModelName >_ecpils, slprj

3.5 Verifying Algorithms of Code Generation Targets

You can use information (execution time of a load module generated from the embedded model and result of execution) obtained by running a PIL Simulation to verify algorithms.

The execution time measurement exported only when the [Measure Execution Time] checkbox is selected in [Embedded Target Options] of the Configuration Parameters dialog box.

Depending on the PIL Simulation mode and Time Measurement method, the procedures to get the execution time might be different. Refer to the blow sections in accordance with the PIL Simulation modes and Time Measurement methods for details.

Remark 1. Time Measurement by Performance Function can be used only on Single-Core PIL Simulation with all supported MCU families.

2. Time Measurement using Performance Function supplies the execution time which is measured by using the timer function provided by CS+/e^2 studio. Therefore, before measuring execution time with a debug tool, the timer function must be turned on. You can turn the timer function on and off by clicking the [Timer] button on the right side of the CS+/e^2 studio status bar.

3. The execution time measurement result is in nanosecond (ns) unit.

The following sections describe the Time Measurement with Performance Function method. This supplies the total execution time of whole PIL Simulation process and can be used on Single-Core PIL Simulation only.

To enable this feature, check the [Measure Execution Time] checkbox, set [Measurement Method] selection list to “Performance Function”.

The measurement result is stored in the following files in according to code generation target:

1. When Code Generation Target is a Subsystem Block

Storage file for execution time measurement result of Subsystem Code Generation Target Block
<*directory containing the model*> \<Subsystem *name*>_ecpils\ <Subsystem *name*>.txt

2. When Code Generation Target is a Reference Model

Storage file for execution time measurement result of Reference Code Generation Target Model
<directory containing model> \slprj\ecpils\<reference model name>\<reference model name>.txt

3. When Code Generation Target is a Top-level Model

Storage file for execution time measurement result of Top-level Code Generation Target Model
<directory containing model>\<top-level model name>_ecpils\<top-level model name>.txt

4. The measurement result is stored in the following format

Normal case	Total: <total execution time>ns, Pass Count: <pass count>, Average: <average execution time>ns, Max: <maximum execution time>ns, Min: <minimum execution time>ns
An overflow has occurred when measuring the execution time of each step	Total: Overflow, Pass Count: <pass count>, Average: N/A, Max: N/A, Min: <minimum execution time>ns
An overflow has occurred when calculating the total execution time	Total: Overflow, Pass Count: <pass count>, Average: N/A, Max: <maximum execution time>ns, Min: <minimum execution time>ns

4. ERROR MESSAGES

This chapter explains the error messages output by Embedded Target.

4.1 Overview

Error messages are output to notify you of information that you should know about events that occur while you are setting [Embedded Target Options] in the Configuration Parameters dialog box or while a PIL simulation is running.

Remark Error messages output by Embedded Target are not linked to CS+/e^2 studio. Therefore, no help is displayed even if you press the F1 key after Embedded Target displays an error message.

4.2 Errors Detected in Configuration Parameters Dialog Box

The following table lists the messages that are output when an error is detected while settings are being made in the Configuration Parameters dialog box.

These error messages are output to the Embedded Target Error dialog box.

Table 4-1. Error Messages for Configuration Parameters Dialog Box

[Message]	E0101 The <Embedded Target option> is invalid. Please check the entered value.
[Explanation]	This error message is displayed when value of an option is entered incorrectly.
[Action by User]	Specify the correct value for options that have type are edit box.
[Message]	E0102 This button cannot currently be pressed. Please change the value of <Embedded Target option>.
[Explanation]	This error message is displayed when a button cannot be pressed because the value of the option indicated in the error message does not match.
[Action by User]	Change the value of the option indicated in the error message.
[Message]	E0104 An incorrect automation server is registered.
[Explanation]	This error message is displayed if a different version of MATLAB is registered as the automation server.
[Action by User]	Register the MATLAB in use as the automation server using the “regmatlabserver” command.
[Message]	E0105 <Embedded Target license> is not registered.
[Explanation]	This error message displayed when the license corresponding to the selected device family is not registered.
[Action by User]	If you don’t have license for PIL Simulation on supported devices, register with Renesas Electronics. If you have license file for PIL Simulation on supported devices already, check if it is put in the Embedded Target installation. To confirm the availability of the feature, please “Check Available Features” on [Embedded Target Options] panel.

[Message]	E0106 The selected debug tool is not supported. Please choose another debug tool.
[Explanation]	This error message displayed when selected Debug Tool is not supported by device family or device name.
[Action by User]	Select another debug tool corresponding to the value of [Device Family] option and [Device Name] option.
[Message]	E0108 No devices are available. Please press [Update Device List] button to update the list of devices corresponding to the selected device family.
[Explanation]	This error message displayed when there are no devices in the device file of the selected device family.
[Action by User]	Check the IDE used and press the [Update Device List] button to update the device file
[Message]	E0109 The model conversion mode does not match <Target MCU>.
[Explanation]	This error message is displayed when the MCU type set in [Target MCU] does not match the model conversion mode.
[Action by User]	When [Multi-Core MCU] has been set, click on the [Convert Model for Multicore MCU] button.
[Message]	E0110 No license is registered. Please register a valid license.
[Explanation]	This error message is displayed when no license is added or license was expired on your system.
[Action by User]	Register Embedded Target License with Renesas Electronics Corporation.
[Message]	E0111 e^2 studio does not support the specified device name.
[Explanation]	This error message is displayed when CS+ does not support this device name
[Action by User]	• Check if [Device Name] is correct. Check if e^2 studio supports the specified device name.
[Message]	E0112 Software Trace is not selected for Measurement Method.
[Explanation]	This error message is displayed when [Performance Function] is set in [Measurement Method].
[Action by User]	Set [Software Trace] in [Measurement Method].

[Message]	E0114 No "Embedded Target for RA" license. Please register license with Renesas Electronics.
[Explanation]	This error message is displayed when you attempt to convert a Multi-rate model of a Multicore MCU using the “Embedded Target for RH850 Multicore” license.
[Action by User]	(1) If you do not have the “Embedded Target for RA” license, please contact Renesas Electronics sales office. (2) If you have the “Embedded Target for RA” license, add the license with reference to the Installation Guide. (3) To confirm the license, click on the [Check Available Features] button on the [Embedded Target Options] panel.
[Message]	E0115 File <target file> has been changed.
[Explanation]	The ecpils files in package has been changed or any file which same name in the current workspace.
[Action by User]	• Re-install Embedded Target • Check whether the files in current workspace have the same name as describe in the error message by use “which –all <target file> “? If so, rename the files in the workspace.

4.3 Errors at Build

The following table lists the messages that are detected at build.

These error messages are output in the Embedded Target error dialog box.

Table 4-2. Error Messages at Build

[Message]	E0201 Please exit <IDE Target>, which has started.
[Explanation]	CS+ or e^2 studio has already been started.
[Action by User]	• Exit the active CS+/e^2 studio and execute rebuilding. • Terminate the CS+/e^2 studio process on the Windows® task manager. • Check that the rapid start function of CS+/e^2 studio is not used.
[Message]	E0202 The current mex compiler configuration is not supported.
[Explanation]	Current compiler tool chain in MATLAB® doesn't contain: mingw64, MSVC (Microsoft Visual Compiler)
[Action by User]	Install MinGW, MSVC compatible with MATLAB® version
[Message]	E0203 The selected template project file does not exist.
[Explanation]	Address of template project which specified on Embedded Target options is not correct.
[Action by User]	Correct address of the template project
[Message]	E0204 The current working directory does not contain model <ModelName>.
[Explanation]	Code generation during the build caused error if current directory is different from project directory. Therefore, Embedded Target will throw warning before generating the code.
[Action by User]	Change the current directory to project directory.
[Message]	E0205 Opening the PIL simulation communications channel was not possible.
[Explanation]	This error message displayed when user want to stop the PIL simulation by click [OK] button in Figure 3-15. Confirmation Dialog Box
[Action by User]	While Embedded Target is building and show message box to stop PIL simulation don’t click [OK].
[Message]	E0206 The GenCodeOnly option is not supported.
[Explanation]	This error message displayed when the [Generate code only] checkbox is checked.
[Action by User]	Uncheck [Generate code only] checkbox.
[Message]	E0207 The Create Block option is not set to "PIL".
[Explanation]	[PIL] is not set to [Create Block] in the Configuration Dialog.
[Action by User]	Open the Configuration Dialog for target model. -> Select All Parameter -> Search for the [Create Block] option -> Specify [PIL] to [Create Block].

4.4 Errors during Starting CS+/e^2 studio and Downloading

The following table lists the messages that are detected in Embedded Target processing from starting CS+/e^2 studio to downloading.

Table 4-3. Error Messages in CS+/ e^2 studio

[Message]	E0300 Creating the <IDE Target> project was not possible (project.Create error). [Direct Cause] <The direct error cause message>
[Explanation]	The CS+ or e^2 studio project file could not be generated.
[Action by User]	• Check that the CS+ or e^2 studio version is supported by Embedded Target. • Check that the CS+ Python plug-in is enabled. • Check that the e^2 studio support area is specified correctly.
[Message]	E0302 Adding the source file was not possible (project.File.Add error).
[Explanation]	The source file could not be registered in the CS+ project file.
[Action by User]	Check that the MATLAB® version is supported by Embedded Target.
[Message]	E0303 Removing the source file was not possible (project.File.Remove error).
[Explanation]	The source file could not be removed from the IDE project file.
[Action by User]	When "Template Project & LM Download" is selected for IDE Mode, check that the IDE project created by Embedded Target has been specified.
[Message]	E0304 Setting the debug tool was not possible (debugger.DebugTool.Change error).
[Explanation]	Cannot change to target Debug Tool, which was set on [Embedded Target options] panel.
[Action by User]	Confirm available connection types (when using E1/E2 emulator) on CS+ project. Re-generate test environment again.
[Message]	E0312 Building was not possible (build.All error).
[Explanation]	An error occurred at build.
[Action by User]	• Check the following and regenerate the test environment. • Review the property setting of CS+/e^2 studio. • Check the error message displayed in the CS+/e^2 studio output panel. • When the memory size of the device is small, consider the use of a device of large memory size.
[Message]	E0320 Connecting the debug tool was not possible (debugger.Connect error).
[Explanation]	An error occurred at connecting the debug tool.
[Action by User]	• Check the property setting of CS+/e^2 studio. • Check that E1/E2/E2 Lite/E20 (JTAG/Serial)/EZ/COM Port/J-Link have been correctly connected.
[Message]	E0321 Downloading the load module was not possible (debugger.Download.LoadModule error).

[Explanation]	An error occurred at downloading a load module.
[Action by User]	• Check the property setting of CS+/e^2 studio. • Check that no error occurred at build of CS+/e^2 studio.
[Message]	E0322 Setting the timer event was not possible (debugger.Timer.Set error).
[Explanation]	An error occurred when cannot set time events.
[Action by User]	Re-allocate core assignment on Simulink® model to reduce the number of timer events. Re-generate and re-execute load module from embedded model.
[Message]	E0323 Opening the e^2 studio project was not possible (project.Open error).
[Explanation]	The e^2 studio project file could not be imported to workspace in e^2 studio IDE.
[Action by User]	• Check that the e^2 studio version is supported by Embedded Target. • Check that the e^2 studio support area is specified correctly.

4.5 Errors during PIL Simulation

The following describes error messages detected during PIL simulation. Error dialog boxes during PIL simulation are output from MATLAB®/Simulink®.

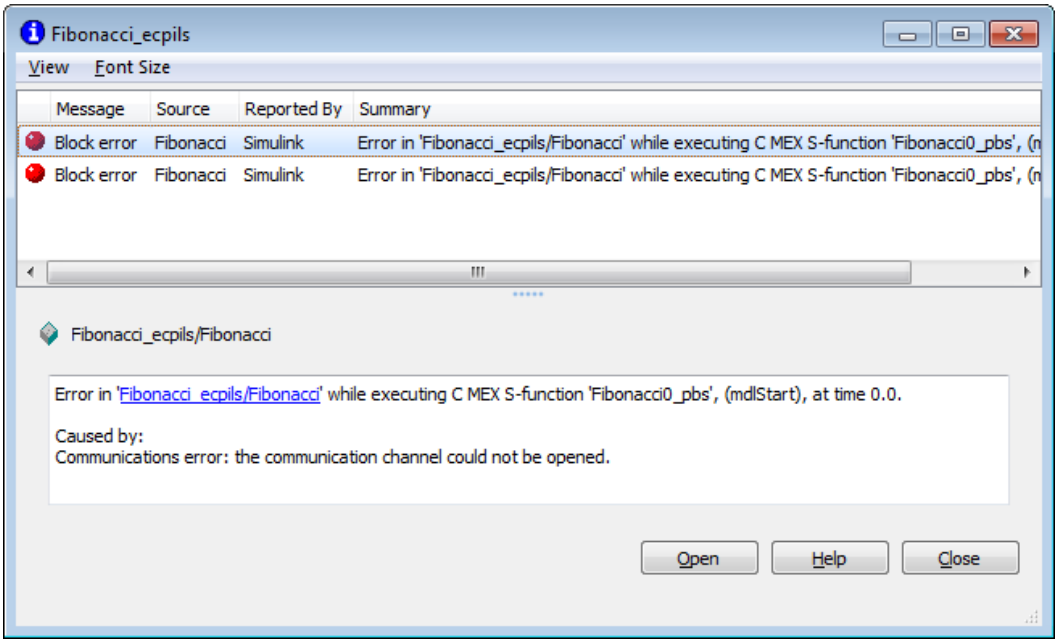

Figure 4-1. Messages in the Error Dialog Box during PIL Simulation

Table 4-4. Actions for Errors during PIL Simulation

[Action by User]	(1) Check that CS+/e^2 studio has been started (2) Check that the debug tool of CS+/e^2 studio is connectable (3) Check that the program has been downloaded to CS+/e^2 studio (4) Check that multiple CS+/e^2 studio has not been started (5) Check that the rapid start function of CS+/e^2 studio has not been used (6) Terminate all processes of MATLAB® and CS+/e^2 studio (7) Use Windows® Task Manager to terminate process regarding CS+/e^2 studio (7-1) Right click on the Task Bar of Windows®, click on “Start Task Manager” (7-2) In the Windows® Task Manager window, choose Processes tab (7-3) Check whether the CubeSuiteW+.exe process has existed. If no, right click on that item and choose End Process item (8) Modify the value of “3000” of Buffer Size option that has been defined in Embedded Target Options (9) Start MATLAB® Re-execute PIL simulation

4.6 Errors during Model Conversion

The following table lists the error messages that are displayed when errors are detected during model conversion.

Table 4-5. Error Messages during Model Conversion

[Message]	E0412 Execution of the model conversion script failed.
[Explanation]	This error message is displayed when execution of the script file for model conversion failed.
[Action by User]	Check information of the input file and make sure the settings of the model are correct before converting it.
[Message]	E0413 Creating core allocation pattern files for model conversion was not possible.
[Explanation]	This error message is displayed when the intermediate files to be used in model conversion cannot be generated.
[Action by User]	If M2PinterfaceCore.csv and M2PInterfaceConn.csv already exist, make sure they are not being opened by another program.
[Message]	E0414 ECPILS_Core.csv and ECPILS_Conn.csv do not exist.
[Explanation]	The "ECPILS_Core.csv" and "ECPILS_Conn.csv" files do not exist.
[Action by User]	The core allocation information is missing or incorrect. Re-generate the information.
[Message]	E0415 Getting information on conversion failed.
[Explanation]	The model parameters cannot access model information for some reason.
[Action by User]	Check whether the model has been opened before using the ecpils_convert_model command.
[Message]	E0416 Converting the model was not possible.
[Explanation]	The conditions required for model conversion are not satisfied.
[Action by User]	Check if the following conditions are satisfied. • The automation server must be set up correctly. • The button for model conversion appropriate for the selected [Target MCU] must be clicked on. • [Software Trace] must be set in [Measurement Method]. • The device set in [Device Name] must be supported by e^2 studio.
[Message]	E0417 The model has already been converted.
[Explanation]	An attempt was made to covert a model that has already been converted.
[Action by User]	Use a model that has not been converted.
[Message]	E0418 Multiple model conversion parameters are specified.
[Explanation]	The number of input arguments of the ecpils_convert_model command is not one.
[Action by User]	Specify either [Single-Core MCU] or [Multi-Core MCU].

[Message]	E0419 Failed by the search processing at list of candidate for measurement block.
[Explanation]	An error has occurred due to one of the following reasons: • The format of the signal name of the Multi-rate model and scheduler in use is incorrect. The format must be “u1_XXtic_out”. • The list of measurement target blocks cannot be written to the measurement target information file due to the lack of access permission.
[Action by User]	• Check the signal name of the scheduler. • If the measurement target information file already exists, make sure it is not being opened by another program.
[Message]	E0420 Failed by the connection search processing.
[Explanation]	This error message is displayed when the intermediate files to be used in model conversion cannot be accessed
[Action by User]	If M2PinterfaceCore.csv and M2PInterfaceConn.csv already exist, perform model conversion after confirming that you have permission to access the files.
[Message]	E0421 There is no candidate for measurement target. Please put only “usable blocks" under the code generation target.
[Explanation]	The list of measurement target blocks in the measurement target information file is empty.
[Action by User]	Confirm the following conditions and then re-execute model conversion. • There must be at least one Subsystem block under the code generation target block. • Only usable blocks must be placed under the code generation target block.
[Message]	E0601 Changing rate information was not possible because the scheduler does not meet the required specification.
[Explanation]	This error message is displayed when the output port name of the scheduler is in an incorrect format (e.g. not u1_XXtic_out).
[Action by User]	Check if the output port name of the scheduler is correct. Another solution is to delete the current Scheduler block and execute the ecpils_create_sample_multi_rate_scheduler command to create a new Scheduler block
[Message]	E0603 The number of existing Scheduler outports does not match the number of new Scheduler outports.
[Explanation]	The number of arguments of the ecpils_create_sample_multi_rate_scheduler command is incorrect.
[Action by User]	Set the number of rate values of the ecpils_create_sample_multi_rate_scheduler command to be equal to the number of output ports of the Scheduler block.
[Message]	E0700 No Subsystem or duplicate Subsystems are selected as the target of measurement.
[Explanation]	This error message is displayed when there is a mistake in checkboxes when selecting the form for specifying measurement targets.
[Action by User]	This is an internal error. Please contact a Renesas Electronics sales office.

[Message]	E0701 Output of the Core.csv file was not possible.
[Explanation]	This error message is displayed when the intermediate files to be used in model conversion cannot be generated.
[Action by User]	Check if there is an M2PinterfaceCore.csv file that you have permission to access. If M2PinterfaceCore.csv that is generated from ECPILS_Core.csv already exists, make sure it is not being opened by another program.
[Message]	E0702 Output of an export file was not possible.
[Explanation]	This error message is displayed when the measurement target information file cannot be generated.
[Action by User]	If the measurement target information file already exists, make sure it is not being opened by another program.
[Message]	E0703 Import of the selected model failed. The target of code generation in the imported file differs from that of the actual model.
[Explanation]	Import of the measurement target information file failed due to one of the following reasons: • The measurement target information file may have been modified manually. • The measurement target information file may be for a different model or a different code generation target block.
[Action by User]	Re-confirm with the measurement target information file that the code generation target and target model for mode conversion are correct. If they differ, the measurement target information file cannot be used. Re-create the measurement target information file.
[Message]	E0704 Import of the selected model failed. The structure of the model in the imported file differs from that of the actual model.
[Explanation]	Import of the measurement target information file failed due to the following reason. Though the model and code generation target are correct, the internal structure of the code generation target has been changed (e.g., usable blocks were added, block name was changed).
[Action by User]	Re-confirm with the measurement target information file that the code generation target has the same model structure as the target model for mode conversion. If they differ, the measurement target information file cannot be used. Re-create the measurement target information file.
[Message]	E0705 Input of an import file was not possible.
[Explanation]	This error message is displayed when importing of the measurement target information file failed for some reason (e.g., not having access permission).
[Action by User]	Re-import the measurement target information file after confirming that you have permission to access it.
[Message]	E0706 Cannot get model information or wrong information.
[Explanation]	This error occurs at startup of the form for specifying measurement targets.
[Action by User]	This is an internal error. Please contact a Renesas Electronics sales office

[Message]	E0801 Output of the script file was not possible.
[Explanation]	An error has occurred due to one of the following reasons: • The format of M2PinterfaceConn.csv and M2PinterfaceCore.csv is incorrect. • Does not have permission to access TestModelScript.m.
[Action by User]	• Re-create the measurement target information file. • If TestModelScript.m already exists, make sure it is not being opened by another program.

Revision History	Embedded Target User's Manual: Operation

Rev.	Date	Description	
		Page	Summary
1.00	Mar.01.25	—	First Edition issued

Embedded Target User's Manual: Operation

Publication Date: Rev.1.00 Mar.01.2025

Published by: Renesas Electronics Corporation

.

Embedded Target

Renesas Electronics Corporation

R20UT5640EJ0100