

Embedded Target

User's Manual: Operation

Target Device RX Devices Family RL78 Devices Family RA Devices Family

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corp. without notice. Please review the latest information published by Renesas Electronics Corp. through various means, including the Renesas Electronics Corp. website (http://www.renesas.com).

Renesas Electronics

Rev.1.00 Jan 2024

Notice

- 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.
- 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 computers: o
 - "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 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.
- This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
 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 is supplied until the power reaches the level at which reseting 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

- T a r g e t This manual is intended for users who wish to understand the functions of the
- R e a d e r s MATLAB[®]/Simulink[®] and design software and hardware application systems.
- P u r p o s e 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.

This manual can be broadly divided into the following units.

Organization

	Chapter 1	GENERAL	
	Chapter 2	INSTALLATION	
	Chapter 3	FUNCTIONS	
	Chapter 4	ERROR MESSAGES	
How to Read This Manual	It is assumed and microcol	that the readers of this manual have general knowledge of electricity, logic circuits, htrollers.	
Conventions	Note:	Footnote for item marked with Note in the text	
	Caution:	Information requiring particular attention	
Remark:		Supplementary information	
	Numeric	Decimal XXXX	
	representatio		

Hexadecimal ... XXXXH or 0xXXXX

Licensing

This product uses the IronPython based on the following license.

Copyright 2024 Renesas Electronics Corporation

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the 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. GE	NERAL		9			
1.1	Overv	iew	9			
1.2	Features10					
1.3	Opera	ting Environment	11			
1.4	Featu	re Use Cases Policy	13			
1.4	4.1	License Policy	13			
1.4	4.2	Feature Use Cases	13			
1.4	4.3	License Management Model	14			
1.5	Basic	Usage	15			
1.5	5.1	When Using a Subsystem Block	15			
1.5	5.2	When Using a Reference Model	18			
1.5	5.3	When Using a Top-level Model	20			
2.	INSTA	LLATION	22			
2.1	Installi	ing Embedded Target	22			
2.′	1.1	Package	22			
2.1	1.2	Procedure	22			
2.2	Uninst	alling Embedded Target	22			
2.3	Deletii	ng a License	23			
3. FUI		NS	24			
3.1	Overv	iew	24			
3.2	Execu	ting PIL Simulation for Subsystem Code Generation Target Block	26			
3.2	2.1	Generating a Test Environment	26			
3.2	2.2	Executing PIL Simulation	41			
3.2	2.3	Debugging Generated Code during PIL Simulation	42			
3.2	2.4	Re-executing Embedded Target	44			
3.2	2.5	Cleanup Embedded Target workspace after PIL Simulation	44			
3.3	Execu	ting PIL Simulation for Reference Code Generation Target Model	45			
3.3	3.1	Generating a Test Environment	45			
3.3	3.2	Debugging Generated Code during PIL Simulation	52			

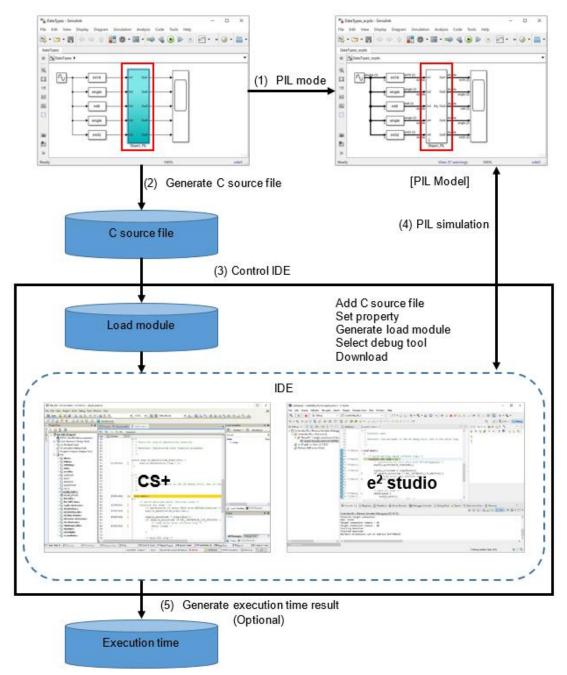
3.3	3.3 Re-executing Embedded Target	55
3.3	3.4 Cleanup Embedded Target workspace after PIL Simulation	55
3.4	Executing PIL Simulation for Top-level Code Generation Target Model	56
3.4	4.1 Generating a Test Environment	56
3.4	4.2 Debugging Generated Code during PIL Simulation	57
3.4	4.3 Re-executing Embedded Target	57
3.4	4.4 Cleanup Embedded Target workspace after PIL Simulation	58
3.5	Verifying Algorithms of Code Generation Targets	50
	Verifying Algonitinis of Code Generation Targets	59
4. ERF	ROR MESSAGES	
4. ERF 4.1		60
	ROR MESSAGES	60
4.1	ROR MESSAGES	60 60
4.1 4.2	ROR MESSAGES	60 60 60 64

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).





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+/e² 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+/e² 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+/e² 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+/e² 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 recommended)	Microsoft [®] Windows [®] 10 (64-bit) and Microsoft Windows [®] 11 (64-bit) (Windows [®] 10 is
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, R2021a to R2023b (R2023b 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 and R2021a)
- Microsoft Visual C++ 2019 compiler (from Microsoft Corporation) (for MATLAB® R2021b to R2023b)
- MinGW 6.3 C/C++ (distributed by mingw-w64) (only when using MATLAB® R2018b to R2023b)
- MinGW 8.1 C/C++ (distributed by mingw-w64) (only when using MATLAB® R2023a to R2023b)

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

4. IDE (from Renesas Electronics)

CS+	V8.10.00, V8.11.00
e ² studio	2023-07, 2023-10, 2024-01

5. Building environment (for generating a load module)

CC-RX	Included with CS+ V8.10.00 or later (from Renesas Electronics) Installed along with e ² studio
CC-RL	Included with CS+ V8.10.00 or later (from Renesas Electronics) Installed along with e ² studio
GNU ARM Embedded	(version: 10.3-2021.10 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 e^2 studio.
- 6. Debug Tools

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

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 to verify 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



Both of target MCU require valid licenses. The Single core MCU:

- RX devices requires "Embedded Target for RX" license.
- RL78 devices requires "Embedded Target for RL78" license.
- RA devices requires "Embedded Target for RA" license. *
 - Remark Embedded Target offers methods to measure the performance of algorithm 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
 - * RA family devices can only be used in e² studio 2023-10

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 according 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.

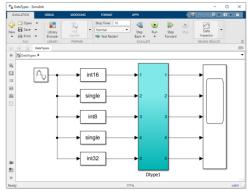


Figure 1-2. Subsystem Model

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

Configuration Parameters: Data1	ypes/Configuration (Active) – 🗆 🗙	Configuration Parameters: DataTypes/	Configuration (Active)	- 0
Q Search		Q Search		
Solver	Hardware board: Determine by Code Generation system target file	Solver		
Data Import/Export	Code Generation system target file: ecols lis: Code Generation system target file: ecols lis:	Data Import/Export	arget selection	
Math and Data Types			System target file: ecpils thc	Brawse
 Diagnostics 	Device vendor: Renesas	 Diagnostics Hardware Implementation 	Language: C	•
Hardware Implementation Model Referencing	* Device details	Model Referencing	Description: Embedded Target	
Simulation Target	Number of bits Largest atomic size	Simulation Target	uild process	
 Code Generation 	char: 8 short: 16 int: 32 integer: Char •	Code Generation		
Optimization Report	long 32 long long 64 float: 32 floating-point None •		Generate code only	
Comments	double 64 native: 32 pointe: 32	Comments	Package code and artifacts Zip file name: ">semply>	
Identifiers	size_t: 32 ptdH_t: 32	Identifiers	Makefile configuration	
Custom Code	NDE_L 32 prom_L 32	Custom Code	✓ Generate makefile	
Interface Code Style	Byte ordering: Little Endian Signed Integer division rounds to: Zero	Interface	Template makefile: ecpils.tmf	
Code Style Verification	Shift right on a signed integer as arithmetic shift	Code Style Verification	Make command: make_rtw	
Templates	Support long long	Templates	mana samanan (mana) m	
Code Placement		Code Placement	ode generation objectives	
Data Type Replacement		Data Type Replacement		0.001.0
Embedded Target Options			Prioritized objectives: Unspecified	Set Objectives
			Check model before generating code: Off	Check Model
			Advanced parameters	
			Custom FFT library caliback: <empty></empty>	
			Custom BLAS library caliback: ">	
			Custom LAPACK library caliback: <emply></emply>	
			Post code generation command: <amply></amply>	
			Verbose build	
			Retain .rtw file	
	OK Cancel Help Apply			
Search		Q Search Solver	IDE Taroat: CS+	
Solver	Code profiling	Data Import/Export	IDE Install Directory: C:/Program Files (x86)/Renesas Electronics/CS+/CC	
Data Import/Export	Measure task execution time	Math and Data Types Plagnostics	Use default IDE Install Directory	
Math and Data Types Diagnostics	Measure function execution times: Off	Hardware Implementation	Select IDE Install Directory	
Hardware Implementation	Workspace variable: executionProfile Save options: Summary data only *	Model Referencing Simulation Target	Workspace Directory: N/A	
Model Referencing	violappee venues. Commany said only	Code Generation	Select Workspace Directory	
Simulation Target Code Generation	Code coverage for SIL or PIL	Optimization	e2 studio Support Area: N/A	
Optimization	Third-party tool: None Configure	Report Comments	Select e2 studio Support Area	
Report		Identifiers	IDE Mode: Create Project	
Comments	C Enable portable word sizes	Custom Code Interface	Template Project: C:/user/template.mtpj	
Identifiers Custom Code	Enable source-level debugging for SIL	Code Style	Select Template Project	
Interface	-	Verification	Device Family: RX	
Code Style		Templates Code Placement	Byte Order: LittleEndian	
Verification		Data Type Replacement	Build Tool: Renesas Compiler	
Templates Code Placement		Embedded Target Options		
Data Type Replacement			Select Device Name	
Embedded Target Options			Update Device List	
			FSP Version: N/A GNU ARM Embedded Version: N/A	
			GNU ARM Embedded Version: N/A Debug Tool: E2	
			Main Clock Frequency (MHz) : 0	10
			Debug Generated Code during PIL Simulation	
			Measure Execution Time	
			Measurement Method: Performance Function	
			Buffer Size: 3000	
			Check Available Features	
			About Embedded Target	
	OK Cancel Help Apply			OK Cancel Help App

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.

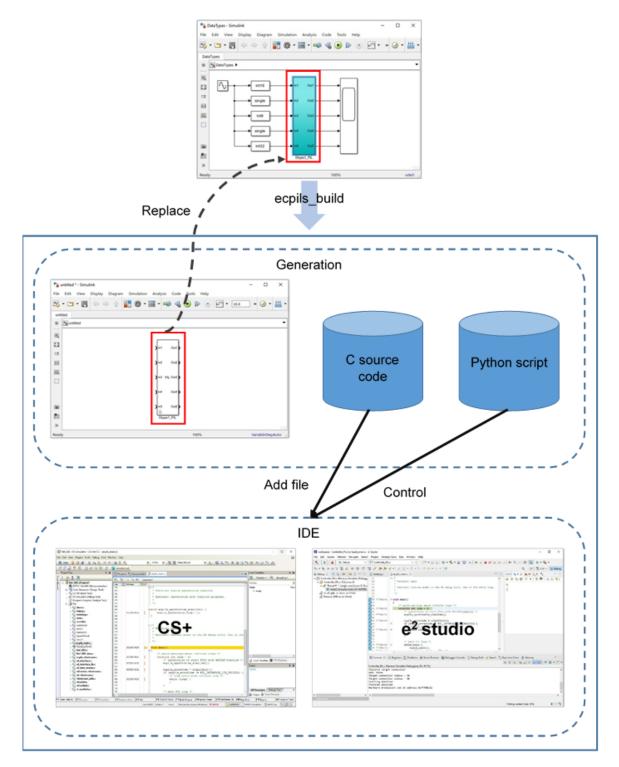


Figure 1-4. Flow of Embedded Target Processing, Case: Subsystem



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

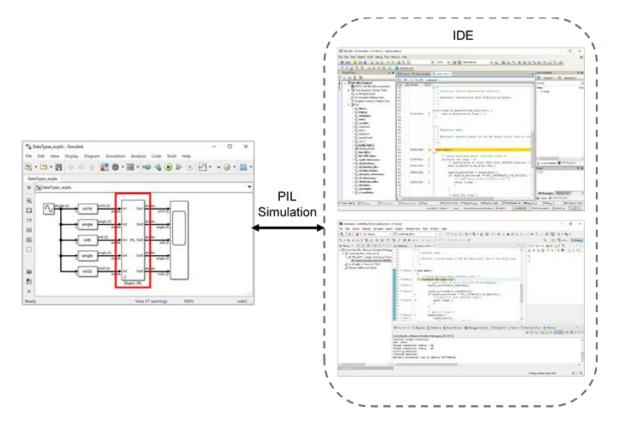


Figure 1-5. Flow of PIL Simulation, Case: Subsystem



1.5.2 When Using a Reference Model

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

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

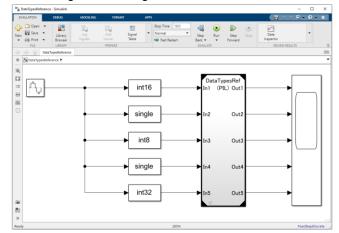


Figure 1-6. Reference Model

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

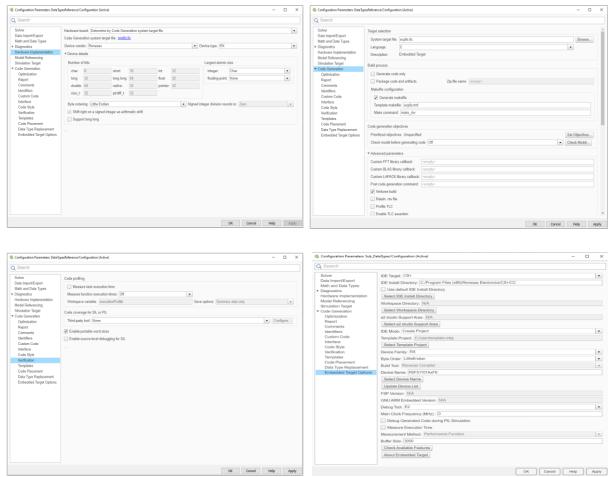


Figure 1-7. Configuration Parameters setting



3. Set PIL mode for the Model block.

Dia ala	Parameters	Madal					×
		Model					
- Model Re	eference						
Reference	e the spec	ified model.					
Main	Instance	e parameters					
Model	name:						
DataT	ypesRef				Browse	Open Model	
Simula	tion mode	Processor-in	-the-loop) (PIL)		•	
Code in	nterface:	Model referen	ce			•	
Model	events sim	ulation:					
Sho	w model ir	itialize port					
Sho	w model te	erminate port					
Sch	edule rates						
0		C	Ж	Cancel	Help	Apply	/

Figure 1-8. Block Parameters setting

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

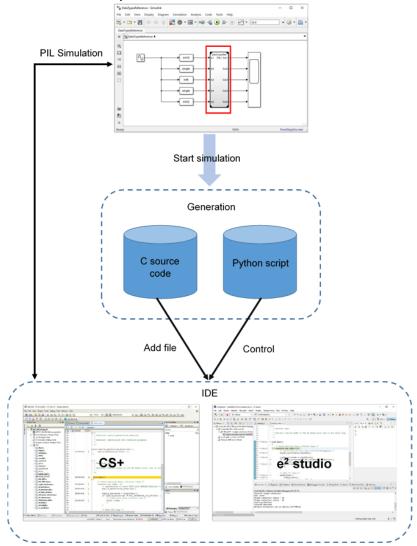


Figure 1-9. Flow of Embedded Target Processing, Case: Reference Model



1.5.3 When Using a Top-level Model

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

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

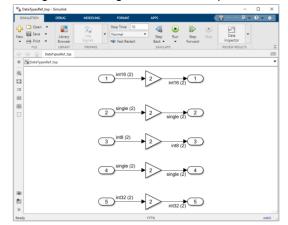


Figure 1-10. Top-level Model

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

Configuration Parameters: Data F	Systelikef_top/Configuration (Activa) - 🗆 🗙	Configuration Parameters: DataTypes	esRef_top/Configuration (Active)	- 🗆 X
Q Search		Q Search		
Solar Data Isyah Tigat Data Isyah Tigat Nakat data Tiga Harakan Ingeneratian Mada International Sandalan Ingen Casta Data Casta Harakan Casta Ca	tectano kol: (Sterrinko fyr. Ch. Generation system target 18	Data Ingest/Egent Marh and Usi Speen Degensitio Media Thermonity Simulation Target Code Generation Report Code Speets Code Speets Code Sple Verification Code Sple Verification Target/Sple Data Target/Splennet Enthedded Target/Splenn	Advanced parameters Coston FFT Binary collevat: Gengelyo Coston LAAS Bony collevat: Gengelyo Coston LAASC Brain collevation Genglyo Post coston connext: Genglyo Vederes load Genation refi Genation ref Genation	Errore
	OK Cancel Help Apply	1	OK Cancel	Help Apply

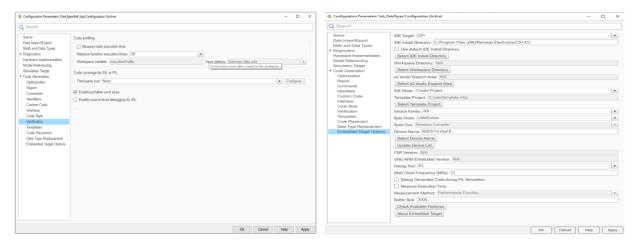


Figure 1-11. Configuration Parameters setting



3. Set PIL mode for the Simulink[®] model.

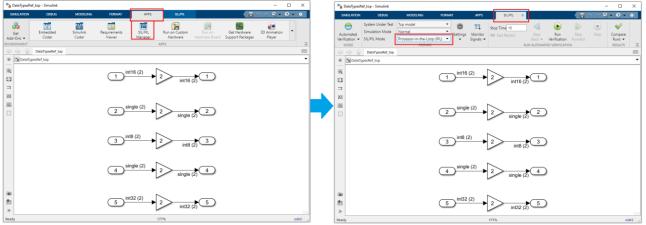


Figure 1-12. PIL Mode Setting

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

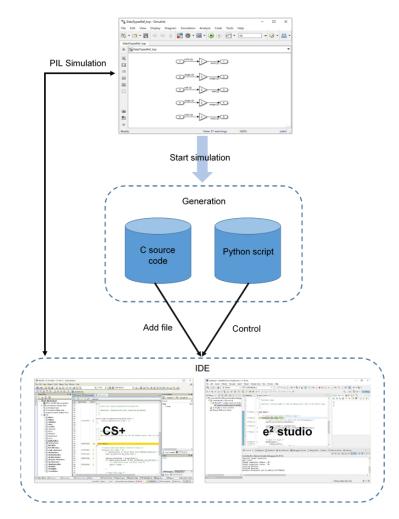


Figure 1-13. Flow of Embedded Target Processing, Case: Top-level Model



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="">\</version>	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.

Add Folder					
Aug roldel	MATLAB search p	nbedded_Target\V6.06.0	0\et		
Add with Subfolders	C:\Program Fil C:\Program Fil C:\Program Fil C:\Program Fil	les\MATLAB\R2023b\too les\MATLAB\R2023b\too les\MATLAB\R2023b\too les\MATLAB\R2023b\too les\MATLAB\R2023b\too	olbox\matlab\addoi olbox\matlab\addoi olbox\matlab\addoi olbox\matlab\addoi	n_updates\matlab ns ns\cef	anagement
	C:\Program Fil	les\MATLAB\R2023b\too les\MATLAB\R2023b\too	olbox\matlab\addor	ns\supportpackage	
Move to Top	C:\Program Fil	les\MATLAB\R2023b\too	blbox\matlab\addor	ns_desktop_registra	tion
Move Up		les\MATLAB\R2023b\too les\MATLAB\R2023b\too			natlab
Move Down	C:\Program Fi	les\MATLAB\R2023b\too	blox\matlab\addor	ns_product_support	(matlab
Move to Bottom	C:\Program Fil	les\MATLAB\R2023b\toc les\MATLAB\R2023b\toc les\MATLAB\R2023b\toc les\MATLAB\R2023b\toc	olbox\matlab\addor olbox\matlab\addre	ns_sidepanel\matlab ssbar_plugins\brow	se_for_fold
Remove	C:\Program Fil	les\MATLAB\R2023b\too les\MATLAB\R2023b\too les\MATLAB\R2023b\too	olbox\matlab\appd	esigner\appdesigne	

Figure 2-1. Set Path Dialog Box



 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	Uses an I/O port of the	Uses an I/O port of the	Uses the MATLAB [®]
generated code	subsystem block.	model block.	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+/e² studio
- e. Register the generated C source files in the CS+/e² 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+/e² 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+/e² studio project file are generated automatically. A test is performed by using them for communication with CS+/e² 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+/e² studio
- f. Register the generated C source files in the CS+/e² 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 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

3.2.1.2 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.

 Open the Configuration Parameters dialog box Select [Model Configuration Parameters] from the [Simulation] menu in the D

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].

Q Search								
Solver Data Import/Export Math and Data Types > Diagnostics Hardware Implementation Model Referencing	Code Genera Device vend Device det	ation system or: Renesas tails	nine by Code Gene target file: <u>ecpils:</u> a		arget file	Device type: RX		•
Simulation Target Code Generation Optimization Report Comments Identifiers Custom Code Interface Code Style Verification Templates Code Placement Data Type Replacement Embedded Target Options	double: size_t: Byte order	8 32 64 32 ing: Little E		32 32	int: float: pointer:	Largest atomic integer: floating-point	Char None	•

Figure 3-1. [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 R2021a and latest).

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

Configuration Parameters: Sub_Da	taTypes/Configu	ration (Activ	e)							-		×
Q Search												
Solver Data Import/Export Math and Data Types Diagnostics Hardware Implementation Model Referencing		ation system or: Renesa	nine by Code n target file: e s		n system tarı	get fil		e type: RL78				•
Simulation Target	Number o	f bits						Largest atomic s	ize			
Code Generation Optimization Report Comments Identifiers Custom Code Interface Code Style Verification Templates Code Placement Det Placement	Byte order	16 ing: Little E	short: long long: native: ptrdiff_t: indian ned integer as	16 64 16 16 arithmetic	float: pointer:		Signed integ	integer: floating-point: ger division round			•	•
Data Type Replacement Embedded Target Options								OK	Cancel	Help	A	pply

Figure 3-2. [Hardware Implementation] Options for RL78



Data Import/Export Math and Data Types Diagnostics Hardware Implementation Model Referencing Stimulation Target Code Generation Optimization	de Gener	ation system for: ARM C	mine by Cod n target file: compatible		on system	target file			
Simulation Target N Code Generation Optimization	Number o					•	Device type: ARM C	ortex	
Code Generation Optimization		f bits					Largest atomic	size	
Report Comments Identifiers Custom Code	char: long: double: size_t:	8 32 64 32	short: long long: native: ptrdiff_t	16 64 32 32	int: float: pointer:	32 32 32	integer: floating-point	Long Double	v
Verification	Shift ri	ring: Little ght on a sig rt long long	ned integer	as arithmet		• Signer	d integer division roun	ds to: Zero	

Figure 3-3. [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.

© Configuration Parameters: DataTy	pes/Configuration (Active	e)					- 0	×
Q Search								
Solver Data Import/Export Math and Data Types > Diagnostics Hardware Implementation Model Referencing Simulation Target • Code Generation Optimization Report Comments Identifiers Custom Code Interface Code Style Verification Templates Code Placement Data Type Replacement Embedded Target Options	Target selection System target file: Language: Description: Build process Generate code Package code Makefile configura (I Generate m Template makef Make command Code generation obj Prioritized objectiv Check model befo	c only and artifacts tion keefle ke copis truf ke (make_thw ectives	Zip file name:	cemply>			Set Objective Check Model	s
					ОК	Cancel	Help	Apply

Figure 3-4. [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.

Configuration Parameters: Data Ty	pes/Configuration (Active)		- C	X I
Q Search				
Solver Data Import/Export Math and Data Types Diagnostics Hardware Implementation Model Referencing Simulation Target Code Generation Optimization Report Comments Identifiers Custom Code Interface Code Style Verification Templates Code Placement Data Type Replacement Embedded Target Options	Code profiing Measure task execution time Measure function execution times: Off Workspace variable: executionProfile Code coverage for SIL or PIL Third-party tool: None C Enable portable word sizes Enable source-level debugging for SIL	Summary data only	▼ Configu	v
		OK Cancel	Help	Apply

Figure 3-5. [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.

Search Solver Data Import/Export Math and Data Types > Diagnostics Hardware Implementation Model Referencing Simulation Target Code Generation Optimization Report Comments Identifiers Custom Code Interface Code Style Verification Project: C: Use Orden: Interface Code Style Verification Template Project: Device Family: RX Device Family: RX Device Family: Device Family: RX Device Family:
Data Import/Export IDE Install Directory: C:/Program Files (x86)/Renesas Electronics/CS+/CC Math and Data Types Use default IDE Install Directory IDE Install Directory: Use default IDE Install Directory Hardware Implementation Select IDE Install Directory Model Referencing Workspace Directory Vorkspace Directory: N/A Select Workspace Directory e2 studio Support Area: Optimization e2 studio Support Area Identifiers IDE Mode: Create Project Custom Code Template Project: Interface Select Template Project Code Style Select Template Project
Liemplates Byte Order: LittleEndian Code Placement Build Tool: Renesas Compiler Data Type Replacement Device Name: R551101AxFK Embedded Target Options Select Device Name: Update Device List FSP Version: N/A GNU ARM Embedded Version: N/A Debug Tool: E2 Main Clock Frequency (MHz) : 0 Main Clock Frequency (MHz) : 0 Debug Generated Code during PIL Simulation Measure Execution Time Measurement Method: Performance Function Buffer Size: 3000 Check Available Features Check Available Features

Figure 3-6. [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	Selects the IDE which you want to use. IDE is CS+ (default) or e ² studio.
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.
[Use default IDE Install Directory] checkbox	Specifies the default folder where CS+/e ² studio has been installed. On Windows [®] 64-bit, it is "C:/Program Files (x86)/Renesas Electronics/CS+/CC" for CS+ or <%userprofile%>/AppData/Local/Programs/Renesas/e2_studio/eclipse for e ² studio.
[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	Clicking 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 e ² 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					
	Selects the type of project file that is loaded when the $CS+/e^2$ studio starts and whether a series of processing including download of a load module is performed after the $CS+/e^2$ studio start-up.					
	Create Project	The default CS+/e ² studio project file provided by Embedded Target is loaded (default).				
	Open Project	The CS+/e ² studio project file saved at the last exit is loaded.				
IDE Mode *7	Open Project & LM Download	The CS+/e ² 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+/e ² studio project file generated based on the CS+/e ² 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], the [Select Template Project] button and specify the CS+ project file name or the e ² studio project folder which is used as a template with t absolute path.					
[Select Template Project] button *9	Displays the dialog box to specify the CS+ project file or the e ² studio project folder which is used as a template with the absolute path. The selected result is reflected to [Template Project].					
	Selects the series name	elects the series name of the microcontroller being used.				
	RX	Selects the RX family as the microcontroller series.				
Device Family *10	RL78	Selects the RL78 family as the microcontroller series.				
	RA *20	Selects the RA family as the microcontroller series.				
	Selects the Byte Order of	of the microcontroller being used.				
Byte Order *11	LittleEndian	Select the Little Endian as the Byte Order				
	BigEndian	Select the Big Endian as the Byte Order				
		r CS+/e ² studio project, this indicates the generate the load module.				
Build Tool *12	Renesas Compiler	Selects any of Renesas compilers, which will be determined by CS+/e ² studio when creating new project.				
	GCC ARM Embedded Compiler	Selects any of GCC ARM Embedded compilers, which will be determined by e ² studio when creating new project for RA devices family.				
Device Name *13	microcontroller being us					
[Select Device Name] button *14	Displays a list of microcontrollers, from which you can select the					



[Update Device List] button *15	Updates the list of microcontrollers displayed by clicking [Select Device Name] button.					
FSP Version	Specifies version of FSF	P being used to create RA family project				
GNU ARM Embedded Version	Specifies version of GN	U being used to create RA family project				
	Selects a Debug Tool and connection type connecting to the target device					
	E2 (default)	Connects to the target device through E2 Emulator				
	E2Lite	Connects to the target device through E2 Lite Emulator				
	COM Port	Connects to the target device through COM Port (For RL78/G23 only)				
	Simulator *18	Uses Simulator of the target device				
Debug Tool *16	E20Serial	Connects to the target device through E20 Emulator with Serial connection.				
	E20Jtag	Connects to the target device through E20 Emulator with JTAG connection.				
	EZ_Emulator *21	Connects to the target device through EZ Emulator				
	E1Jtag	Connects to the target device through E1 Emulator with JTAG connection.				
	E1Serial	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) *17	Sets main clock frequency of the target device					
		Measure Execution Time feature is disabled.				
Debug Generated Code	When checked	PIL Simulation could be interrupted by User's operation on CS+/e ² studio.				
during PIL Simulation	When not checked	Measure Execution Time feature can be enabled by User.				
	When not checked	PIL Simulation could NOT be interrupted by user's operation on CS+/e ² studio.				
	When checked	Execution time is measured in verification of algorithm.				
Measure Execution Time *18	When not checked	Execution time is not measured in verification of algorithm and PIL simulation is performed at high speed (default).				
	Selects the Time Measu	irement method is being used.				
Measurement Method	Performance Function	The execution time of the load module which executes on CS+/e ² studio during the PIL Simulation is measured by using Performance Function of CS+/e ² studio debug tools.				
Buffer Size	Specifies value of Buffer	r Size in range 1000 ~ 3000				
[Check Available Features] button *19	Displays list of available	requiring features in Embedded Target System.				
[About Embedded Target] button	Displays version informa Target.	ation and copyright information of Embedded				



*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".

e ² studio Installation Details	_		×
Installed Software Installation History Features Plug-ins Configuration Renesas Device Support	Support F	olders	
e² studio support area: file:/C:/Users/eclipse/com.renesas.platform_406684935			
e ² studio download area: <u>file:/C:/Users//.eclipse/com.renesas.platform_download/</u>			
	_		
?	L	Close	

Figure 3-7. 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+/e² 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 e² studio project folder created with Embedded Target. When a CS+ project file or e² 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].

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... The [Byte Order] list is valid to set when [Device Family] list is set to "RX". The [Byte Order] is set to [Little Endian] and can not be changed when [Device Family] list is set to "RL78" or "RA".



- *12... When [Build Tool] is set to "Renesas Compiler" or "GCC ARM Embedded", the build tool is decided by CS+/e² studio at the project creation time.
- *13... The setting is only valid if "Create Project" is selected for [IDE Mode]. When "Open Project" or "Open Project & LM Download" is selected, an error occurs. However, if there is no CS+ project file or e² studio project folder that has been generated at the last exit, information that has been specified is used to generate the device name.
- *14... 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" is selected for [IDE Mode], the list is not displayed, and an error occurs.
- *15... CS+/e² studio is started to get the latest information from CS+/e² studio, but it is automatically terminated. The information is updated at the time of the CS+/e² studio termination.
- *16... The connection to real target device (LSI device) by any of E1 connection types through Embedded Target is same with manual connection on CS+/e² studio GUI. For an effective usage, please create a dummy project for the same target on CS+/e² 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.

EZ Emulator only supports for some of RX100, RX200, RX63x series.

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+/e² studio packages.

- *17... 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/J-Link Emulator or COM Port" connection types.
- *18... [Measure Execution Time] does not support for some cases. Refer to "4.3 Can not support measuring execution time" in RESTRICTIONS.
- *19... When clicking on this button, the dialog box displays and shows list of requiring features. Free-ofcharge features are not showed in this box. These can be used freely.
- *20... When creating project for RA device family on e² studio, there is a case that version of FSP or GCC ARM Embedded is difference with the packages included with e² studio. If the default version of FSP and GCC ARM Embedded do 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".
- *21... EZ emulator will not support RX device family on CS+ V8.09.00 / e² studio 2023-01 and later

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



3.2.1.3 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.



3.2.1.4 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 the operations for generating a test environment. Accordingly, an 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.

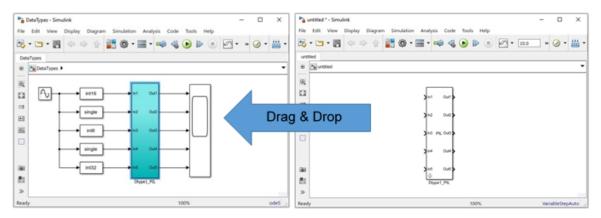


Figure 3-8. Replacing a Block for PIL Sequential Execution

Remarks 1. After blocks for PIL sequential execution has 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.



3.2.1.5 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, an explicit operation by the user is not necessary.

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 RX] Select [Build Tool] in the CS+ project tree and set [Handles in double precision (-dbl_size=8)] for [Common Options] - [CPU] - [Precision of the double type and long double type].

[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 e^2 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 RA8T1 device series on 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 \rightarrow [Board] \rightarrow [Device]. Choose the appropriate board and device.

(*): 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] \rightarrow [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.

3.2.1.6 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].

• 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+.

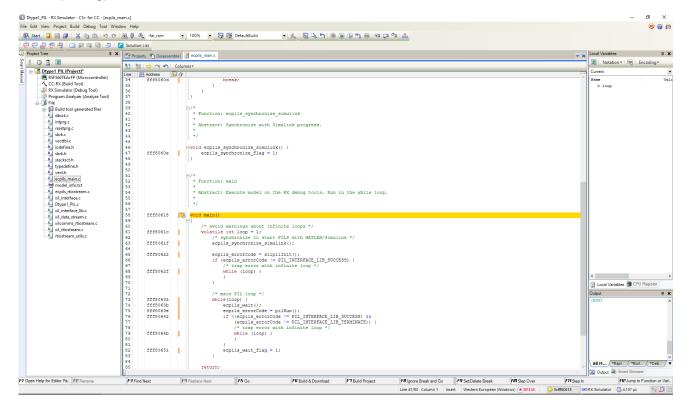


Figure 3-9. CS+ on Completion of Downloading to the Debugger



[When using e² studio IDE] 称 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. ġ. ८ ୭ ୭ ୦ - ୦ - ۲ - ۲ 2 - 8 - 5 - 0 - 0 - 1 -Q. 2 C/C++ D Debug Debug 33
 Debug 34
 Va 💁 Br 📸 Mo 📰 Dis 🏠 Pr 🛠 Ex 🛹 Ex X 🛃 Pe 📰 I 👘 🗰 * Function: main * * Abstract: Execut Address Data fype Frace Start Frace Stop Frace Record Frace Record Frace Record Frace Record Frace Record Frace Record Frace Start Frace S void main() (/* avoid warnings about infinite loops */
volatile int loop = 1;
 /* synchronize to start PILS with MATLAB/Simulink */
ecplis_synchronize_simulink(); fff8073e fff80741 /* main Pll loop */ while(loop) { explip_wit(); explip_wit(); explip_errorCode = pll_plrtEMAKE_itS_SACCES) && (f(cplip_errorCode is Pll_plrtEMAKE_itS_SACCES) && (f traps_error with infinite loop */ while (loop) { } } }
ecpils_wait_flag = 1; } return; ff8077b Project Saved Ter Console [1] ||III Registers (E) Problems @ Smart Browser @ Debugger Console [] Debug Shell [] Men J.P.N.Debug Remeans Smulator Debugging (RC, R178) Version 8.5.0-v28218528-185515 [47cda364] (fray 28 2821.22111.48) EC = x % | % § # # # # # • • • • • arting server with the following options: Raw options : C:\Users\1 \.eclipse\com.renesas.platform_406684935\DebugComp\\RX\02-server-gdb -g SIMULATOR -t R5F565TE -uPeripheralClkRatio- 1 -uCpuClkF c: user
 c: user
 construction
 construction ished target uneman... is 55366 rget connection status - OK arting download rdware breakpoint set at address 0xfff00737 rdware breakpoint set at address 0xfff00737

Figure 3-10. e² 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.

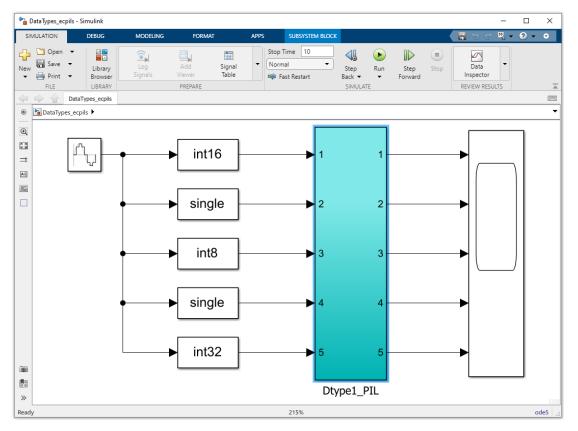


Figure 3-11. PIL Simulation Executions

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 3-12. 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+/e² 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+/e² 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:

Solver Data Import/Export Data Import/Export IDE Target: CS+ Diagnostics IDE Install Directory: [C:/Program Files (x86)/Renesas Electronics/CS+/CC Hardware Implementation Model Referencing Simulation Target Use default IDE Install Directory Code Generation Select IDE Install Directory Optimization Report Comments Identifiers Identifiers Custom Code Interface Code Style Verification Templates Code Placement Device Family: RX Byte Order: LittleEndian Build Tool: Report Code Placement Device Name: Build Tool: Renesas Compiler Device Name: R5F51101AxFK Select Device List FSP Version: N/A GNU ARM Embedded Version: N/A GNU ARM Embedded Version: N/A Debug Tool: E2 Main Clock Frequency (MHz): 0	O Search		
Debug Generated Code during PIL Simulation Measure Execution Time Measurement Method: Performance Function Buffer Size; 3000 Check Available Features	Data Import/Export Math and Data Types Diagnostics Hardware Implementation Model Referencing Simulation Target Code Generation Optimization Report Comments Identifiers Custom Code Interface Code Style Verification Templates Code Placement Data Type Replacement	IDE Target: CS+ IDE Install Directory: C:/Program Files (x86)/Renesas Electronics/CS+/CC Use default IDE Install Directory Select IDE Install Directory Workspace Directory: N/A Select Workspace Directory e2 studio Support Area IDE Mode: Create Project Template Project: C:/user/template.mtpj Select Template Project Device Family: RX Byte Order: LittleEndian Build Tool: Renesas Compiler Device Name: RSF51101AxFK Select Device Name Update Device List FSP Version: N/A GNU ARM Embedded Version: N/A Debug Generated Code during PIL Simulation Measure Execution Time Measurement Method; Performance Function Buffer Size; 3000	

Figure 3-13. Enable Debug Generated Code during PIL Simulation



I Openhole Kamele. I Openhole Kamele. I Openhole Kamele Kamele. I Openhole	Dtype1_PIL - RX Simulator - CS+ for CC - [ecpils_	main.c]		– a ×
Comparing Comparing <thcomparing< th=""> <thcomparing< th=""> <thc< td=""><td>File Edit View Project Build Debug Tool Wir</td><td>ndow Help</td><td></td><td>🧐 🔞 🧐</td></thc<></thcomparing<></thcomparing<>	File Edit View Project Build Debug Tool Wir	ndow Help		🧐 🔞 🧐
Company Company <t< td=""><td>💐 Start 🔒 🖶 🍠 💥 🐚 🖄 ભ</td><td>a a a -far_rom</td><td>💌 100% 👻 🔄 DefaultBuild 🔍 🔨 🖏 🖧 🐂 📵 🕞 🕼 🐂 🛞 😒 🖓 🖄</td><td></td></t<>	💐 Start 🔒 🖶 🍠 💥 🐚 🖄 ભ	a a a -far_rom	💌 100% 👻 🔄 DefaultBuild 🔍 🔨 🖏 🖧 🐂 📵 🕞 🕼 🐂 🛞 😒 🖓 🖄	
A Decision 100 100 100 100 100 100 100 100 100 10		Solution List		
Image: Control of the control of t			Net Secola mains	Local Variables 4 X
B Control Ling Ling Action Control Contro Control Control Con	° 2 Ø 2 ⊠			📳 Notation • 😁 Encoding •
<pre>1 / Openholds Edure A: Phone // Ph</pre>				Current -
<pre>P Open the for the first first</pre>	CC-RX (Build Tool)	34 fff8060d 35 36		
<pre>Pi Judits_trans.</pre>	Bre Bre Bre Bre Bud tool generated files Bud tool generated B	38 39 40 41 42 43 44 45 46 47 7 ff8060e 48 50 50 51 52 53 53 55 56	<pre>* Abstract: Synchronize with Simulink progress. */ */ */ */ */ ** ** ** ** ** ** ** **</pre>	
Pi Deen Mep for Edator Ph. PF Register Pf Section Pix Material Sectin Pix Material Section Pix Material Section Pix Material Section P	xil_data_stream.c xilcomms_ttiostream.c xil_tiostream.c	58 fff80618 59 60 61 fff8061c	<pre>0f /* avoid warnings about infinite loops */ volatile int loop = 1;</pre>	
i ff200422 i explig_error0xde = silplifin(); if (explig_error0xde = silplifin(); i	tiostream_utils.c	62 63 fff8061f 64		
 		65 fff80622 66	if (ecpils errorCode != PIL INTERFACE LIB SUCCESS) (
73 fff0063 while(loop) (eppil_wait(); eppi		68 fff8062f 69 70 71	while (loop) ())	
P3 fff004bb j while (loop) (j 13 j		73 fff80635 74 fff8063b 75 fff8063e 76 fff80642 77	<pre>while(loop) (ecplia_wait(); ecplia_errorCode = pilRun(); if (leoplia_errorCode = FIL_INTERFACE_LIB_SUCCESS) && (ecplia_errorCode = FIL_INTERFACE_LIB_TEMENATE)) (</pre>	<pre>inded(Success:1 Pro A jects, Failed:0 Projects)(Mond ay, June 7, 2021 6:27:16 BM) = intermed itstopped by Hardware Break.j</pre>
B5 returns/ PT Capet Help for Editor Pis. PT Stand Project		79 fff8064b 80 81 82 fff80651 83	<pre>while (loop) (</pre>	~
17 Open Help for Editor Ph. //R Rename // 2 Find Next // Replace Next // Replace Next // Sop 0 // F5 Build & Download // 1 Build Project // Bu		85	return; v	
	FI Open Help for Editor Pa. F2 Rename	F3 Find Next	PH Replace Next #5 Go #6 Build & Download F7 Build Project F8 Ignore Break and Go #59 Set/Delete Break #80 Step Over F71 Step In	FR2 Jump to Function or Vari

Figure 3-14. Two automatic breakpoints in "ecpils_main.c" file in CS+

🐞 🔳 🎄 Debug 🗸 💽 Dtype1_PIL Debug		- [씨 - 卷 마 田 남 왕 값 왕 御 [僧] 🌶 💩 🏷 - [기 위 - 집 - 역 각 수 - 수 - [㎡ -]
ebug 🛛 📄 🐩 🕯 🖻	□ Recplis_main.c ⊠	🖳 🔅 🕫 😯 📴 📴 👘 🖓 👘 🖓 👘 👘 🖓 👘 🖓 👘 👘 🖓 👘 👘 👘
Dtype1_PIL Debug [Renesas Simulator Debugging (RX, RL78)]	34 fff8072c break;	
Provide a straight of the s	35 } 36 }	✓ A ² [address: 0x00000000fff8072d] [type: Hardware] ✓ A ² [address: 0x00000000fff80737] [type: Hardware]
ecpils_synchronize_simulink() at ecpils_main.c:47 0xfff8072d	37 }	✓ A [address: 0x00000000000/3/1[type: Hardware]
main() at ecpils_main.c:63 0xfff80741	36 · · · · · · · · · · · · · · · · · · ·	
rx-elf-gdb -rx-force-isa=v3 (7.8.2)	40 * Function: ecpils_synchronize_simulink	
📕 Renesas GDB server (Host)	41 *	
	42 * Abstract: Synchronize with <u>Simulink</u> progress. 43 *	
	44 */	
	45 46 Svoid ecpils_synchronize_simulink() {	
	e 47 fff8072d ecpils synchronize flag = 1;	
	48 }	
	49 50	
	51 @/*	
	52 * Function: main 53 *	
	54 * Abstract: Execute model on the RX debug tools. Run in the while loop.	
	55 *	
	56 */ 57	
	58 fff80737 @void main()	
	59 { 60 /* avoid warnings about infinite loops */	
	61 fff8073b volatile int loop = 1;	
	62 /* synchronize to start PILS with MATLAB/Simulink */ 63 fff8073e ecpils synchronize simulink();	
	64 ecplis_synchronize_simulink();	
	<pre>65 fff80741 ecpils_errorCode = silpilInit();</pre>	
	66 ⊖ if (ecpils_errorCode != PIL_INTERFACE_LIB_SUCCESS) { 67 /* trap error with infinite loop */	
	68 fff8074e ⊖ while (loop) {	
	69 }	×
		· · · · · · · · · · · · · · · · · · ·
	📮 Console 🛛 🔠 Registers 🖹 Problems 🏶 Smart Browser 🖳 Debugger Console 🗍 Debug Shell 📋 Memory	= X ½ k k Ø 🖓 🖓 ଟ 😁 ▾ 📬 ▾
	Dtype1_PIL Debug (Renesas Simulator Debugging (RX, RL78)) version 0.5.0.v.20210520-101515 (#7204046) (may 20 2021 22:11:46)	
	Starting server with the following options:	
		<pre>bugComp\\RX\e2-server-gdb -g SIMULATOR -t R5F566TE -uPeripheralClkRatio= 1 -uCpuClkFreq= 12 -uReg</pre>
	Connecting to SIMULATOR, RX Target	
	GDBServer endian : little	
	Target power : off	
	Starting target connection Finished target connection	
	GDB: 55366	
	Target connection status - OK Target connection status - OK	
	Starting download	
	Finished download	
	Hardware breakpoint set at address 0xfff8072d Hardware breakpoint set at address 0xfff80737	
	Hardware breakpoint set at address 0xfff80737	
	6	

Figure 3-15. 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+/e² 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+/e² studio process during debugging. This will cause unexpected behaviors.

5. Do not change the workflow of PIL Simulation on the CS+/e² 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 3.2.1.5 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+/e² 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+/e² 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+/e² 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 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.

3.3.1.2 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.

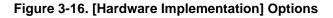
1. Open the Configuration Parameters dialog box

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.

Configuration Parameters: DataTyp	esReference/Cor	nfiguration (Active)									-		×
Q Search													
Solver Data Import/Export Math and Data Types Diagnostics Hardware Implementation Model Referencing		tion system target or: Renesas		ation system target f	ile		•	Device type: RX					•
Simulation Target	Number of	bits						Largest atomic	size				
 Code Generation Optimization 		8	short:	16	int:	32		integer:	Char			•	
Report Comments	long: double:	32	long long: native:	64 32	float: pointer:	32		floating-point	None			•	
Identifiers		32	ptrdiff_t:	32	pointer:	32							
Identifiers Custom Code Interface Code Style Verification Templates Code Placement Data Type Replacement Embedded Target Options		ng: <u>Little Endian</u> ht on a signed intr	eger as arithn	netic shift		• Si	gned	i integer division rou	nds to: Z	ero			Ÿ
									ОК	Cancel	Help	A	pply





- 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.

Configuration Parameters: DataTyp	esReference/Configuration (Active)		-	
Q Search				
Solver Data Import/Export Math and Data Types Diagnostics Hardware Implementation Model Referencing Simulation Target Code Generation Optimization Report Comments Identifiers Custom Code Interface Code Style Verification Templates Code Placement Data Type Replacement Embedded Target Options	Options for all referenced models Rebuild: Always Parallel Enable parallel model reference builds MATLAB worker initialization for builds: None ✓ Enable strict scheduling checks for referenced models Options for referencing this model Total number of instances allowed per top model: Multiple Propagate sizes of variable-size signals: Infer for Minimize algebraic loop occurrences Propagate all signal labels out of the model Use local solver when referencing model	n blocks in model		•
	Ose local sover when referencing model Model dependencies: Advanced parameters Perform consistency check on parallel pool Include custom code for referenced models Pass fixed-size scalar root inputs by value for code gen	ration		*
			OK Cancel Help	Apply

Figure 3-17. [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.

Search							
Solver	Target selection						
Data Import/Export Math and Data Types	System target file:	ecpils.tlc			Browse		
Diagnostics	Description:	Embedde	d Target for Renesas CS+(processor-in-the-loop)			-	
Hardware Implementation	Shared coder dictionary:	<empty></empty>			Set up		
Nodel Referencing Simulation Target	Language:	С			•		
ode Generation	Language standard:	C89/C90	(ANSI)		-		
Optimization							
Report Comments	Build process						
Identifiers	Generate code only						
Custom Code Interface	Package code and ar	rtifacts					
Code Style	Makefile configuration						
Verification	Generate makefile						
Templates Code Placement	Template makefile: ecpils.tmf						
Data Type Replacement	Make command: make	æ_rtw					
Embedded Target Options							
	Code generation objectives	S					
	Prioritized objectives: Un	nspecified			Set Objectives		
	Check model before gene	erating co	e: Off	-	Check Model		
	 Advanced parameters 						
	Built-in FFTW library	callback					
	Custom FFT library callba		<empty></empty>				
	Custom BLAS library call		<empty></empty>				
	Custom LAPACK library		<empty></empty>				
	Post code generation cor		<empty></empty>				
	Verbose build						
	Retain .rtw file						
	Profile TLC						

Figure 3-18. [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

Configuration Parameters: DataTy	pesReference/Configuration (Active)		- 🗆 ×
Q Search			
Solver Data Import/Export Math and Data Types Diagnostics Hardware Implementation Model Referencing Simulation Target Code Generation Optimization Report Comments Identifiers Custom Code Interface Code Style Verification Templates Code Placement Data Type Replacement Embedded Target Options	Code execution time profiling Measure task execution times: Off Workspace variable: executionProfile Code stack profiling Measure task stack usage Stack workspace variable: stackProfile Code coverage for SIL or PIL Third-party tool: None Catable portable word sizes Enable source-level debugging for SIL or PIL Advanced parameters Create block: None	Save options: Summary data only	Configure
		OK Cancel	Help Apply

Figure 3-19. [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]".

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



3.3.1.3 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.

Block Parameters: Model	×
Model Reference	
Reference the specified model.	
Main Instance parameters	
Model name:	
DataTypesRef Browse Open Model	
Simulation mode: Processor-in-the-loop (PIL)	
Code interface: Model reference	
Model events simulation:	
Show model initialize port	
Show model terminate port	
Schedule rates	
OK Cancel Help Apply	

Figure 3-20. Function Block Parameters Dialog Box

2. Select PIL mode

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

3.3.1.4 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+/e² studio are performed in preparation for PIL simulation.

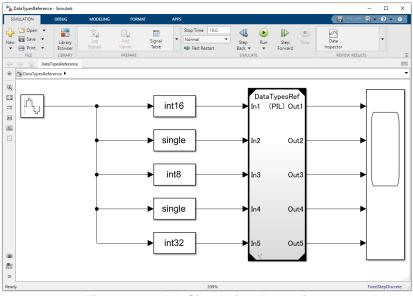


Figure 3-21. PIL Simulation Executions



Remark If code generation is performed after this operation, CS+/e² studio is started. The storage file for execution time measurement result will be removed.

3.3.1.5 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".

3.3.1.6 Downloading a load module

CS+/e² 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 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" 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+.



E Rename	F3 Find	u ivext	 Replace 	next Jara Go	Dulid & Download	· · cuira Project		nsert Western European (RX Simulator (2) 4237 µs	Function of
en Help for Editor Pa_ F2 Rename	79 80 81 82 83 84 85 64 85	fff80653 fff80659	I I F9 Replace	<pre>while (loop) } copils_wait_flag veturn;</pre>	(F7 Build Project	FB Ignore Break and Go	F9 Set/Delete Break	FHD Step Over	FIIStee	All H (*Rapi (*Bail Output @ Smart Brows In FR2Jump to	er
	75 76 77 78	fff80646 fff8064a		ecpils_errorCode if ((ecpils_erro (ecpils_erro	= pilRun(); rCode != PIL_INTERFACE_LI rCode != PIL_INTERFACE_LI r with infinite loop */							
	74	fff80643		ecpils_wait();							(EOF)	
	72	fff8063d		<pre>/* main PIL loop */ while(loop) {</pre>							Output	
	71										-	riegis
	69			3							CPL	Deele
	68	fff80637		while (loop) {								
	66 67				<pre>!= PIL_INTERFACE_LIB_SUC th infinite loop */</pre>	CESS) {						
	65	fff8062a	1	ecpils_errorCode = s	ilpilInit();							
	63 64	fff80627		ecpils_synchronize_s	imulink();							
rtiostream_utils.c	62		· · ·	/* synchronize to st	art PILS with MATLAB/Simu	link */						
	60 61	fff80624		* avoid warnings about volatile int loop = 1;	infinite loops */							
xil_data_stream.c	59	11130620										
	57	fff80620	C% void	main()								
	56		*/									
	54 55		* Ab	stract: Execute model o	n the RX debug tools. Run	in the while loop.						
	53		*									
ecpils_main.c	52			inction: main								
	50 51		E/*									
	49		L*									
	47	11180616	, e	cpils_synchronize_flag	- 1;							
	46			ecpils_synchronize_simu								
vecttbl.c	44		L */									
e resetprg.c	43											
	42		* Ab	stract: Synchronize wit	h Simulink progress.							
	40		* Fu	inction: ecpils_synchron	ize_simulink							
- J File	39											
- Program Analyzer (Analyze Tool)	37 38		}									
🔊 RX Simulator (Debug Tool)	36		}								* 100p	
CC-RX (Build Tool)	34 35	fff80615		break;							Name loop	
DataTypesRef (Project)* R5F566TEAxFP (Microcontroller)	Line	Address									Current	
Ø 🙎 🔳	\$1 50	- 1 -> -	Columns								🔁 Notation 🕶 👼 🛛 E	ncodin
	× Prope	erty 🐴 Disasser	ible1 🥑 ecpils_	main.c						-	× Local Variables	
P 🖉 🐬 🍕 🗀 🗭 🖷 🔍 🗗	🛛 Solution	n List										
	(* 198 * ** *	-far_rom	• 100%	💌 🔄 🕎 DefaultBuild	- 🔨 🖓 🗅 ") 🔘	ک کې کې (ی) کې 🕲 🔘	*= 👗					
t 📕 🖩 🗿 🖁 🖓 🖪 🖪 🕐 (Di 1.55 (8) (8)											

Figure 3-22. CS+ on Completion of Downloading to the Debugger

[When using e² studio IDE] Click the target project in [Project Explorer] panel to set focus and click session and then download a load module.

icon to launch a debugger

*

- a ×

					λ 😰 🖫 c/c++	
Debug 🖂 📄 🙀 🖬 👘	ADSCIACC, EXECUTE HOUSE OF THE RA GEOUN LOOIS, BUT IN THE HITLE LOOP.		(x)-Va 🗣 Br 🛋 Mo 🗄			- 0
C DataTypesRef Debug [Renesas Simulator Debugging (RX, RL78)] 2 DataTypesRef.x [1] [cores: 0]	55 56 */	^			PC: 3/1024 OA: 0/1024	
Phread #1 1 (single core) [core: 0] (Suspended : Breakpoint)	57		Туре	Address	Data	Cou
main() at ecpils_main.c:58 0xfff80741	9 58 fff80741 ⊖void main()		Trace Start			
nx-elf-gdb -nx-force-isa=v3 (7.8.2) Renesas GDB server (Host)	60 /* avoid warnings about infinite loops */		Trace Record			
All Hereas dou server (Host)	61 fff80745 volatile int loop = 1; 62 /* synchronize to start PILS with MATLAB/Simulink */		C Event Break			
	63 fff80748 ecpils_synchronize_simulink();		Timer Start O Timer Stop			
	<pre>p4 65 fff8074b ecpils_errorCode = silpilInit();</pre>		L P timer stop			
	66 e if (ecpils_errorCode != PIL_INTERFACE_LIB_SUCCESS) { /* trap error with infinite loop */					
	68 fff80758 e while (loop) {		-			
	69 } 70 }					
	71					
	73 fff5075e ⊖ while(loop) {					
	74 fff80764 ecpils_wait(); 75 fff80767 ecpils errorCode = pilRun();					
	76 fff8076b = if ((ecpils errorCode != PIL INTERFACE LIB SUCCESS) &&					
	77 (ecpils_errorCode != PIL_INTERFACE_LIB_TERMINATE)) { 78 /* trap_error with infinite loop */		_			
	79 fff80774 🖷 while (loop) {					
	82 fff8877a ecpils_wait_flag = 1; 83 }					
	84					
	85 return; 86 fff80785 }					
	87					
	88		4			
	90	~	Project Saved Templates			
	🖸 Console 🔀 IIII Registers 🙎 Problems 🌒 Smart Browser 🙀 Debugger Console 🗍 Debug Shell 🚺 Memory					
	DataTypesRef Debug [Renesas Simulator Debugging (RX, RL78)] version 6.5.6.v20210520-101515 [44/eudo4e] (ray 20 2021 22:11:46)					
	Starting server with the following options:					
	Raw options : C:\Users\ \.eclipse\com.renesas.platform_406684935\DebugComp	\\RX\e2-server-gdb -g SIM	JLATOR -t R5F566TE -uPe	ripheralClkRatio= 1	-uCpuClkFreq= 12 -	uRegiste
	Connecting to SIMULATOR, RX Target GDBServer endian : little					
	GDBServer endian : little Target power : off					
	Starting target connection Finished target connection					
	GD8: 51158					
	Target connection status - OK Target connection status - OK					
	Starting download					
	Finished download Hardware breakpoint set at address 0xfff8072d					
	Hardware breakpoint set at address 0xfff80737					
	Hardware breakpoint set at address 0xfff80741					

Figure 3-23. e² studio on Completion of Downloading to the Debugger



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



Figure 3-24. Confirmation Dialog Box

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+/e² 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+/e² 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:



Configuration Parameters: Sub_D	ataTypes/Configuration (Active) -	\times
Q Search		
Solver Data Import/Export Math and Data Types Diagnostics Hardware Implementation Model Referencing Simulation Target Code Generation Optimization Report Comments	IDE Target: CS+ IDE Install Directory: C:/Program Files (x86)/Renesas Electronics/CS+/CC Use default IDE Install Directory Select IDE Install Directory Workspace Directory: N/A Select Workspace Directory e2 studio Support Area: N/A Select e2 studio Support Area	•
Identifiers	IDE Mode: Create Project	-
Custom Code Interface Code Style	Template Project: C:/user/template.mtpj Select Template Project	
Verification Templates	Device Family: RX	-
Code Placement	Byte Order: BigEndian	•
Data Type Replacement Embedded Target Options	Build Tool: Renesas Compiler Device Name: R5F566TEAxFP	*
	Select Device Name Update Device List	
	Debug Tool: E2	-
	Main Clock Frequency (MHz) : 8	
	Debug Generated Code during PIL Simulation Measure Execution Time	
	Measurement Method: Performance Function	-
	Check Available Features	
	About Embedded Target	
	OK Cancel Help	Apply

Figure 3-25. Enable Debug Generated Code during PIL Simulation



DataTypesRef - RX Simulator - CS+ for CC - [ecpils	main.c]	-
File Edit View Project Build Debug Tool Wine	ow Help	🥹 🥝 🎰
Start 🔒 🗐 🗿 🐰 🖄 🖄 🕫 🕫	🏨 🌺 -far_rom 🔹 100% 🔹 💀 💀 DefaultBuild 🔹 🤸 💱 🖓 DefaultBuild	
	Solution List	
🗘 Project Tree 4 🗙	Property T Disassemble1 decpils_main.c	- x Local Variables # X
S 2 🗿 🙎 📓	3) 5) → <	Notation • 🔤 Encoding •
DataTypesRef (Project)* Aspect Ref (Project)* Aspect Ref (Microcontroller)	Line ### Address D @	Current
CC-RX (Build Tool)	34 fff80615 break; 35	Name Valu
RX Simulator (Debug Tool)	36 }	
Program Analyzer (Analyze Tool)	37 38	
- J File	39 🖂 /*	
	40 * Function: ecpils synchronize simulink	
🛀 intprg.c	41 42 * Abstract: Synchronize with Simulink progress.	
	43 *	
vecttbl.c	44 45	
iodefine.h	15 46 Broid ecpils synchronize simulink() (
🔛 sbrk.h	47 fff80616 ecpils synchronize_flag = 1;	
	48	
	50	
ecpils_main.c	51 ⊟/* 52 * Function: main	
	53 *	
	54 * Abstract: Execute model on the RX debug tools. Run in the while loop.	
Sil_interface.c	55 * 56 */	
xil_interface_lib.c	57	
	58 fff80620 void main() 59	
	60 /* avoid warnings about infinite loops */	
	61 fff80624 volatile int loop = 1;	
The second contraction of the second contrac	62 /* synchronize to start FILS with MATLAB/Simulink */ 63 fff80627 ecpils_synchronize_simulink();	
	64	
	65 fff8062a ecpils_errorCode = silpilInit(); 66 if (ecpils errorCode != FIL INTERFACE LIB SUCCESS) {	
	67 /* trap error with infinite loop */	
	68 fff80637 while (loop) { 69 }	
	70)	Local Variables 👼 CPU Register
	71	
	72 /* main PIL loop */ 73 fff8063d while(loop) {	Output 🧛 🗙
	74 fff80643 ecpils wait();	Stopped by Hardware Break.
	75 fff80646 ecpils errorCode = pilRun(); 76 fff8064a if ((ecpils errorCode != PIL INTERFACE LIB SUCCESS) 55	Stopped by Hardware Break.
	76 fff8064a if ((ccpils_errorCode != PIL_INTERFACE_LIB_SUCCESS) 66 77 (ccpils_errorCode != PIL_INTERFACE_LIB_TERMINATE)) ((KOF)
	78 /* trap error with infinite loop */	
	79 fff80653 while (loop) (80 }	
	81 }	
	82 fff80659 ecpils_wait_flag = 1; 83 }	
	53) 54	All H / "Rapi / "Buil / "Deb / "
	85 return;	All Pr (Kapi Buit Buit Buit
Close like to Stim Dr. 1720		
F? Open Help for Editor Pa. FE Rename	F3 Find Next F4 Replace Next F5 Go F6 Build & Download F7 Build Project F8 Ignore Break and G	
Stopped by Hardware Break.	Line 47/93 Column 1	Insert Western European (Windows) 🔳 BREAK 😜 0xff80616 🚥 RX Simulator 👸 43.000 ns 🛐 💭 🔛

Figure 3-26. Two automatic breakpoints in "ecpils_main.c" file

🔯 202107 - DataTypesRef/src/ecpils_main.c - e² studio			– 8 ×
Elle Edit Source Refactor Navigate Search Project Renesas Views Bun		to De la fonda	
🐔 🎄 🔳 🎋 Debug 🗸 💽 Data Types Ref Debug	· · · · · · · · · · · · · · · · · · ·	18° 🖬 🝊 🕸 11	
			Q 📴 🛱 C/C++ 🏘 Debug
	(e) ecpls_mainc [3] (2) (4) * Function: ecpls synchronize simulink	•	00 Wa
PataTypesRef.x [1] [cores: 0] PataTypesRef.x [1] [cores: 0] (Suspended : Breakpoint)	41 * 42 * Abstract: Synchronize with Simulink progress.		✓ Jad [address: 0x00000000fff80737] [type: Hardware]
ecpils_synchronize_simulink() at ecpils_main.c:47 0xfff80737	43 * 44 */		Jaddress: 0x0000000fff80741] [type: Hardware]
■ main() at ecpils_main.c:63 0xfff8074b ↓ nx-elf-gdb -nx-force-isa=v3 (7.8.2)	45 46 @void ecpils_synchronize_simulink() {		
Renesas GDB server (Host)	47 fff80737 ecpils_synchronize_flag = 1;		
	49 50		
	51 ⊕/* 52 * Function: main		
	53 * 54 * Abstract: Execute model on the RX debug tools. Run in the while loop.		
	55 * 56 */		
	57 58 fff80741 @void main()		
	59 { 60 /* avoid warnings about infinite loops */		
	61 fff80745 volatile int loop = 1; 62 /* synchronize to start PILS with MATLAB/Simulink */		
	63 fff80748 ecpils_synchronize_simulink();		
	64 65 fff8074b ecpils_errorCode = silpilInit();		
	66 ⊕ if (ecpils_errorCode != PIL_INTERFACE_LIB_SUCCESS) { 67 /* trap error with infinite loop [#] /		
	68 fff80758 while (loop) { 69 }		
	70 }		
	72 /* main PIL loop */ 73 fff8075e ⊕ while(loop) {		
	74 fff80764 ecpils_wait(); 75 fff80767 ecpils_errorCode = pilRun();	~	
		>	
	🔁 Console 🕴 👭 Registers 🖹 Problems 🌸 Smart Browser 🙀 Debugger Console 👔 Debug Shell 🔋 Memory		= X 🔆 🔒 🖬 🖗 💭 🛃 🗢 🖘 😁 🗆
	Data TypesRef Debug (Renesas Simulator Debugging (RX, RL78))		~
	Starting server with the following options: Raw options : C:\Users\ \.eclipse\com.renesas.platform_406684935\DebugComp\\RX\e2-serv	/er-gdb -g SIMU	LATOR -t R5F566TE -uPeripheralClkRatio= 1 -uCpuClkFreq= 12 -uRegiste
	Connecting to SIMULATOR, RX Target		
	GDBServer endian : little Target power : off		
	Starting target connection Finished target connection		
	GD8: 51158 Target connection status - OK		
	Target connection status - OK Starting download		
	Finished download Hardware breakpoint set at address 0xfff8072d		
	Hardware breakpoint set at address 0xfff80737 Hardware breakpoint set at address 0xfff80737		
	Hardware breakpoint set at address 0x11700/44 Hardware breakpoint set at address 0xfff80741		
	¢		×
Surgeorded			

Figure 3-27. 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+/e² 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+/e² studio process during debugging. This will cause unexpected behaviors.

5. Do not change the workflow of PIL Simulation on the CS+/e² 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 3.2.1.5 Generating a load module.

3.3.3 Re-executing Embedded Target

Terminate the Simulink[®] model previously executed and CS+/e² 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 Preparing a model using a top-level model

Use a sample model DataTypesRef_Top.slx.

3.4.1.2 Setting configuration parameters

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

3.4.1.3 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.

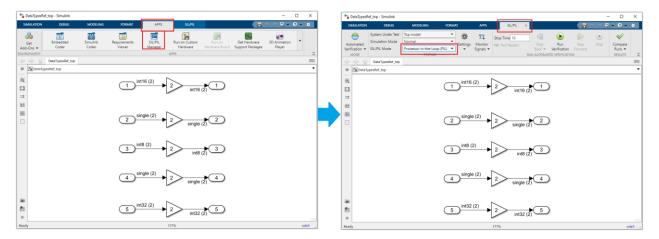


Figure 3-28. Simulation Menus for MATLAB[®] R2021a and above

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.



3.4.1.4 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+/e² studio are performed in preparation for PIL simulation.

鞜 DataTypes	Ref_top - Simulink								-		×
SIMULATION	DEBUG	MODELING	FORMAT	APPS	SIL/PIL	×		6 8	c 🖳	• ? •	•
Automated Verification 🕶 MODE	System Under Test Simulation Mode SIL/PIL Mode	Top model Normal Processor-in-the-Loop PREPARE	▼ @ ▼ Settings	Monitor Signals 👻	Stop Time 10	Step Back V	Run Verification ATED VERIFICATI	Step Forward ON	Stop	Compare Runs • RESULTS	
	DataTypesRef_top										10000
🛞 🎦 DataT	ypesRef_top										•
© ≅ ⇒			1	(2)	2 int16	(2) 1					
Ai											
			2 singl	e (2)	2 single	2					
			3 int8 ((2)	2 int8	→ 3					
			4 singl	e (2)	2 single	→ (2)					
• • •			5 int32	(2)	2 int32	► 5 (2)					
Ready					162%					o	le5 🔡



Remark If code generation is performed after this operation, CS+/e² studio is started. The storage file for execution time measurement result will be removed.

3.4.1.5 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 "3.2.1.5 Generating a load module".

3.4.1.6 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 3.3.1.6 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 3.3.3 Debugging Generated Code during PIL Simulation.

3.4.3 Re-executing Embedded Target

Terminate the Simulink[®] model previously executed and CS+/e² 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 according 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

Total: <total execution time>ns, Pass Count: <pass count>ns, Average: <average execution time>ns, 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+/e² 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.



[Message]	E0101				
	The IDE directory setting is incorrect.				
[Explanation]	This error message is displayed when information on the correct installation destination has not been set in [IDE Install Directory].				
[Action by User]	 Set the correct path of the installation destination of CS+/e² studio to [IDE Install Directory]. The folder containing CubeSuiteW+.exe must be specified. 				
	 Check that CS+/e² studio has normally been installed. 				
[Message]	E0102				
	A template project file is not currently selectable. Change the IDE mode.				
[Explanation]	This error message is displayed when other than [Template Project & LM Download] is specified for [IDE Mode] and the [Select Template Project] button is clicked.				
[Action by User]	Specify [Template Project & LM Download] for [IDE Mode] and click on the [Select Template Project] button.				
[Message]	E0103				
	A device name is not currently selectable. Change the IDE mode.				
[Explanation]	This error message is displayed when other than [Create Project] is specified for [IDE Mode] and the [Select Device Name] button is clicked.				
[Action by User]	Specify [Create Project] for [IDE Mode] and click on the [Select Device Name] button.				
[Message]	E0104				
	Cannot execute Multi-Core Model Conversion action. Please change the PIL Simulation Mode.				
[Explanation]	This error message is displayed when other than [Multi-Core] is specified for [PIL Simulation Mode] and the [Generate] button is clicked.				
[Action by User]	Specify [Multi-Core] for [PIL Simulation Mode] and click on the [Generate] button.				
[Message]	E0105				
	PIL simulation of <device family=""> is not available. Register a valid license.</device>				
[Explanation]	This error message displayed when the license for any PIL Simulation mode for RX/RL78 devices is invalid.				
[Action by User]	If you have got license file for PIL Simulation on RX/RL78 devices, 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.				

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



[Message]	E0106
[The selected debug tool is not supported.
[Explanation]	This error message displayed when selected Debug Tool is not supported.
[Action by User]	Check Debug Tool and related settings such as:
. , .	 Check if IDE Install Directory is correct
	 Check if Device Family and Device Name is correct
[Message]	E0107
[Message]	The default directory is selected as the IDE installation directory.
	Deselect the use of the default directory.
[Explanation]	This error message displayed when [Select IDE Install Directory] button is clicked, while the [Use default IDE Install Directory] checkbox is checked.
[Action by User]	Uncheck [Use default IDE Install Directory] checkbox.
[Message]	E0108
[meesage]	No devices are available.
[Explanation]	This error message displayed when Device List of selected Device Family is empty.
[Action by User]	Try to update Device List by pushing on [Update Device List] button.
	 Check if currently used IDE supports selected Device Family or not.
[Message]	E0110
	A license is not registered.
[Explanation]	This message displays when no license or license was expired on your system.
[Action by User]	Register Embedded Target License with Renesas Electronics Corporation.
[Message]	E0115
	File <target file=""> has been changed.</target>
[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.</target>
[Message]	E0116
	The workspace directory is not currently selectable. Change the IDE target.
[Explanation]	This error message is displayed when other than [e ² studio] is specified for [IDE Target] and the [Select Workspace Directory] button is clicked.
[Action by User]	Specify [e ² studio] for [IDE Target] and click on the [Select Workspace Directory] button.
[Message]	E0117
	The e ² studio support area is not currently selectable. Change the IDE target.
[Explanation]	This error message is displayed when other than [e ² studio] is specified for [IDE Target] and the [Select e2 studio Support Area] button is clicked.
[Action by User]	Specify [e ² studio] for [IDE Target] and click on the [Select e2 studio Support Area] button.
[Message]	E0118 The specified <fsp buffer="" device="" gnu="" name="" size="" version=""> is not valid.</fsp>
[Explanation]	This error message is displayed when the specified FSP/GNU ARM Embedded version does not exist in the selected e ² studio or the specified device name does not exist in the device file or the specified FSP version does not match the selected device name or Buffer Size is not in the range 1000~3000.



		 Specify [e² studio] for [IDE Target], [RA] for [Device Family] and edit on the [FSP Version]/ [GNU ARM Embedded Version]. Specify value on the [Buffer Size]
		• Opeony value on the [Duner Oize].
		• Update the device file by pressing [Update Device List] button and specifying the value again.
		Specify another FSP version, which matches the selected device name.



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.</ide>			
[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>.</modelname>			
[Explanation] Code generation during the build caused error if current directory is different from project directory. Therefore, Embedded Target will throw warning before generative 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 simulatio [OK] button in Figure 3-12. 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 -> Sear for the [Create Block] option -> Specify [PIL] to [Create Block].				



4.4 Errors during Starting CS+/e² 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+/ e² studio

[Message]	E0300					
	Creating the <ide target=""> project was not possible (project.Create error).</ide>					
	[Direct Cause]					
	<the cause="" direct="" error="" message=""></the>					
[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+/e² studio. 					
	 Check the error message displayed in the CS+/e² 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]	n by User] • Check the property setting of CS+/e ² studio.				
	 Check that no error occurred at build of CS+/e² 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 stu					
[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[®].

1	1 Fibonacci_ecpils							
⊻i	<u>V</u> iew <u>F</u> ont Size							
	Message	Source	Reported By	Summary				
0	Block error	Fibonacci	Simulink	Error in 'Fibonacci_ecpils/Fibonacci' while executing C MEX S-function 'Fibonacci0_pbs', (n				
0	Block error	Fibonacci	Simulink	Error in 'Fibonacci_ecpils/Fibonacci' while executing C MEX S-function 'Fibonacci0_pbs', (n				
•				4				
(Fibonacci_ecpils/Fibonacci Error in 'Fibonacci_ecpils/Fibonacci' while executing C MEX S-function 'Fibonacci0_pbs', (mdlStart), at time 0.0. Caused by: Communications error: the communication channel could not be opened. 							
	Open Help Close							

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 stud				
	(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			



Revision History		У	Embedded Target User's Manual: Operation		
Rev.	Date	Description			
		Page Summary			
1.00	Jan.23.24		First Edition issued		

Embedded Target User's Manual: Operation

Publication Date: Rev.1.00 Jan.23.2024

.

Published by: Renesas Electronics Corporation

Embedded Target



•

R20UT5448EJ0100