Quick Start Guide

RA FSP Example Project Porting Guide

Quick Start Guide

Renesas RA Family

Renesas Electronics
www.renesas.com

Rev.1.00 Feb.23.26

Notice

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

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

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

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

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

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/

Trademarks

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

Trademarks (continued)

For the “Cortex” notation, it is used as follows;
— Arm® Cortex®-A55
— Arm® Cortex®-M33
Note that after this page, they may be noted as Cortex-A55 and Cortex-M33 respectively.

Examples of trademark or registered trademark used in the RZ/G2L SMARC Module Board RTK9744L23C01000BE User’s Manual: Hardware;
CoreSight™: CoreSight is a trademark of Arm Limited.
MIPI®: MIPI is a registered trademark of MIPI Alliance, Inc.
eMMC™: eMMC is a trademark of MultiMediaCard Association.

Note that in each section of the Manual, trademark notation of ® and TM may be omitted.
All other trademarks and registered trademarks are the property of their respective owners.

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

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

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

Renesas RA Family

RA FSP Example Project Porting Guide

Introduction

Renesas provides evaluation boards and example projects to help customers assess RA MCUs. When two evaluation boards use the same MCU, example projects may be provided for only one of the kits. To run an example project on these evaluation boards, it may be necessary to port an example project from another kit.

This application note outlines the fundamental knowledge needed to port example projects from one kit to another. It then demonstrates these principles by explaining how to port three example projects of varying levels of complexity from the MCK-RA4T1 to the FPB-RA4T1. The ported projects are included with this application note as a bundle so that users can check their work. Although these steps target the RA4T1 specifically, this application note is also applicable to porting between other RA kits.

Target Devices

- EK-RA4E2, FPB-RA4E2
- FPB-RA4T1, MCK-RA4T1
- EK-RA6E2, FPB-RA6E2
- CK-RA6M5, EK-RA6M5
- FPB-RA6T3, MCK-RA6T3

Requirements

Evaluation Hardware

- MCK-RA4T1
- FPB-RA4T1
- Micro USB cable
- USB C cable
- Oscilloscope and probes
- Jumper Wires

Evaluation Hardware

- e² studio 2025-07
- FSP 6.1.0
- SEGGER JLink v8.58
- Keil µVision5, IAR Embedded Workbench for Arm, Renesas RA Smart Configurator for 6.1.0
- GUI Diffing Tool (Beyond Compare, WinMerge, Meld)

Advisory Notes

This application note assumes you have some experience with the Renesas e² studio IDE and RA Family Flexible Software Package (FSP). Before you perform the procedures in this application note, follow the procedure in the FSP User Manual to build and run the Blinky project on the evaluation hardware mentioned above. Doing so enables you to become familiar with e² studio and the FSP, and validates that the debug connection to your board functions properly.

The screenshots provided throughout this document are for reference. The actual content may differ depending on the version of the software and development tools used.

Contents

1. Overview of Porting Process

Porting a project from one target to another introduces unique, context-dependent considerations. There is no one way to port a project, but there are some common steps that are good to follow:

- **Understand the source project.** Choose a project to port, and try to understand its functionality. Read the included text and/or markdown files, and inspect the project's code. Optionally, replicate the project on its intended target to get a better idea of how it operates.
- **Evaluate hardware differences.** When porting a project from one target to another, it is helpful to understand the hardware differences between the targets. The user's manual and board schematic for both hardware targets are helpful for recognizing these differences.
- **Create a new template project for the target.** Be sure to select the correct options during project creation for compatibility with the original project.
- **Move code and configurations.** Copy the src/ folder and other custom code from the source project to the target project. Compile the code and take note of any errors that appear. If applicable, recreate stacks and other hardware-independent configurations from the source project in the target project.
- **Modify configurations to match target hardware.** If the project requires the use of external hardware, ensure that all pin configurations are correct, and if applicable, ensure that stacks are configured to use the corresponding hardware resources. Refer to the MCU and board user's manuals for information about pin assignments and availability of hardware modules.
- **Edit code to match target hardware.** Modify any non-generic code in the project that could cause issues. Look for private FSP structures/macros, BOARD_* macros, magic numbers, and hardcoded memory locations.

2. Hardware Overview

To effectively port projects, it is necessary to understand the hardware capabilities of both the source and target platforms. Each Renesas development board may differ in terms of CPU functionality, pin mappings, and more. These differences can directly impact project portability and runtime behavior.

The FPB-RA4T1 and MCK-RA4T1 belong to two different kit families. The FPB family is intended for general prototyping purposes, while the MCK family specializes for high voltage/motor control use cases. Below is a table with details of the hardware specifications of the FPB-RA4T1 and MCK-RA4T1.

Table 1. Hardware Specifications of FPB-RA4T1 and MCK-RA4T1

Hardware Specifications	Product Name	FPB-RA4T1	MCK-RA4T1
Hardware Specifications	Purpose	Fast Prototyping	Motor Control
	MCU Group	RA4T1	RA4T1
	MCU Product No.	R7FA4T1BB3CFM	R7FA4T1BB3CFM
CPU Information	CPU Max Operating Frequency	100MHz	100MHz
	CPU Bit Count	32 bit	32 bit
	CPU Package / Pin Count	LFQFP / 64 pin	LFQFP / 64 pin

	CPU ROM	256KB	256KB
Available Hardware Features	**MCU input clock**	24 MHz crystal (unpopulated), 32,768 Hz reference clock	10MHz crystal
	User LEDs	x2	x2 on CPU Board, x3 on Inverter Board
	User Switch	x1	x2 on Inverter Board
	Other User Input	N/A	On main board: Through holes for CAN On Inverter Board: Variable Resistor, External connector for Hall sensor, Encoder connector
Compatible Connectors	**PMOD Connectors**	x2	x2
	Other Connectors	Arduino Uno compatible headers, x2 32 pin headers (not populated)	x2 34-pin connector, SCI connector, unpopulated test points

Of note is that the two boards use the same MCU. They use the same pin package and thus have the same number of pins. However, pin access and functionality varies between the MCK-RA4T1 and FPB-RA4T1 evaluation boards. That is, the onboard peripherals may be assigned to different pins on different boards. For more information about hardware specifications and pin connectivity, review the Hardware User's Manuals and schematics for the FPB-RA4T1 and MCK-RA4T1.

3. Template Project and BSP Overview

The Renesas RA Flexible Software Package (FSP) enables users to develop code for RA MCUs in a hardware-agnostic manner. The FSP contains self-contained units of code called modules that may act as peripheral drivers or implement software features. FSP modules may call functions from a Board Support Package (BSP) to implement board-specific behavior. BSPs also define a board-specific startup function that sets up the stacks, heap, clocks, interrupts, C runtime environment, and stack monitor. BSP functions use standardized function signatures, and code in the FSP is guarded by preprocessor macros so that features absent from specific boards don't get compiled.

The next subsections will highlight the aspects of the BSP that influence the porting process.

3.1 Project Layout

When creating a project in e² studio, code from the FSP and BSPs will be placed inside of the ra/, ra_gen/, and ra_cfg/ folders. The image below shows the typical file structure for a minimal e2studio project.

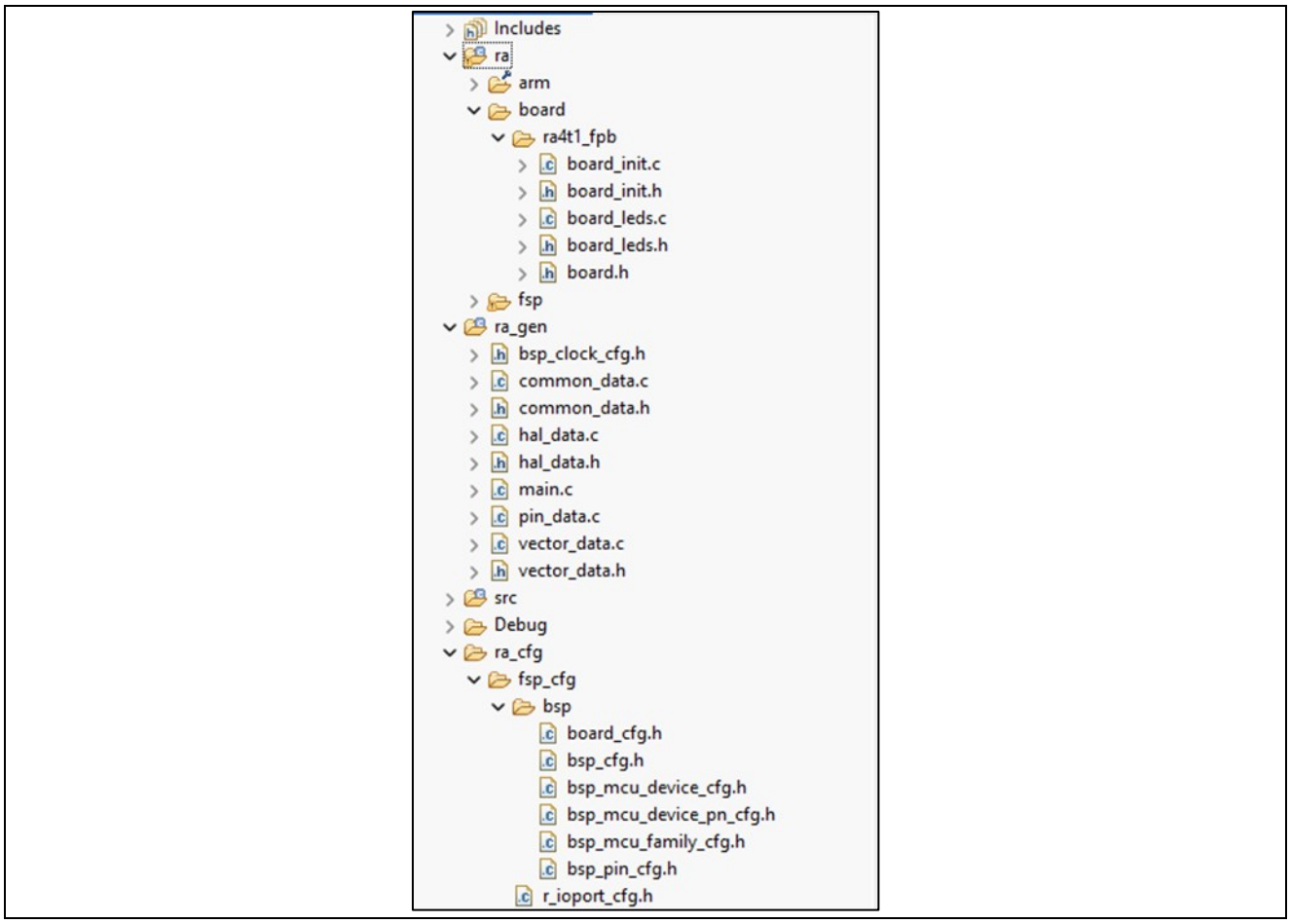


Figure 1. Typical Directory Structure for an Autogenerated e² studio Project

All modules distributed with FSP are packaged as CMSIS components in CMSIS packs. The ra/ folder includes code and headers from these packs. The ra/fsp/ directory includes MCU-specific code, BSP functionality, and stack-level code, while the ra/board/ directory includes code specific to the selected kit. The ra_gen/ and ra_cfg/ folders contain code and header files that are generated based on properties from the project's configuration. The ra_cfg/ folder contains header files with macro definitions that are used by FSP modules and BSP functions for conditional compilation, while the ra_gen/ folder mostly contains data structures that are used when calling module instances.

3.2 Differences between MCK-RA4T1 and FPB-RA4T1 projects

In order to determine what should change when porting a project, it is helpful to use a diffing tool like Beyond Compare or WinMerge. Below is the result of a comparison between two minimal projects created for the MCK-RA4T1 and FPB-RA4T1 respectively. Only files that differ between the two projects are shown below.

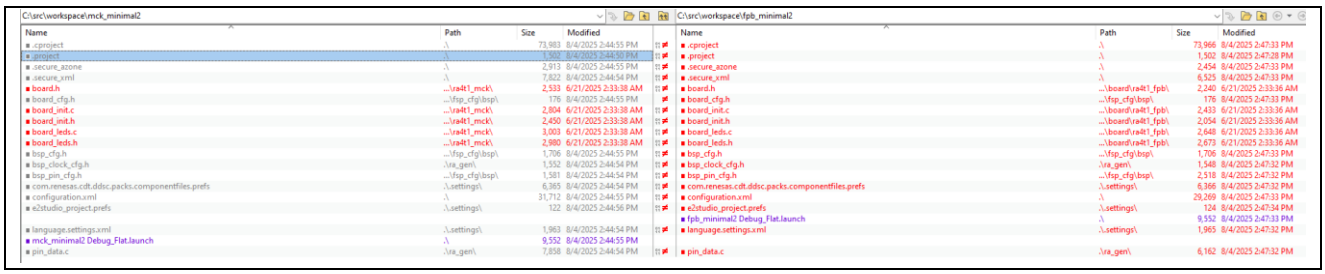


Figure 2. A View of Two Minimal Projects in Beyond Compare

The differences can be categorized as follows:

General:

- The project name is different.
- References to "mck_ra4t1" are replaced with "fpb_ra4t1"

Project Configuration:

- The *.launch debugger configuration has a different filename and configuration based on the project name.
- Some properties in .xml files have randomly generated numbers to prevent conflicts when merging properties.
- Default pin and clock configurations are different. This applies to the .secure_azone and configuration.xml files.

Generated Code:

- Pin symbol names are different. These are specified in the configuration.xml but are included in the auto-generated code under ra_gen/fsp_cfg/bsp/bsp_pin_cfg.h.
- The ra_gen/pin_data.c file is generated from the default pin configuration.
- ra_gen/bsp_clock_cfg.h and ra_cfg/fsp_cfg/bsp/bsp_cfg.h are generated from the clock configurations.
- Different board-level pack files are included in the project. *.pack archives are used by e2studio for code generation. Consequently, the included board-level BSP code in the ra/board/ folder differs based on the kit name.
- The g_bsp_prv_leds[] array in board_leds.c contains different pin numbers.
- board.h defines a macro based on the kit name. Specifically, BOARD_RA4T1_MCK is replaced with BOARD_RA4T1_FPB.

4. Preparations

Before being able to port example projects from the MCK-RA4T1 to the FPB-RA4T1, there are a few preliminary steps that must be followed.

4.1 Configure the Bootloader for Encryption Support

Users can install example projects for the MCK-RA4T1 from the Renesas website by going to https://www.renesas.com/en/design-resources/boards-kits/mck-ra4t1, then clicking on the sample code button under "MCK-RA4T1 Example Project Bundle" in the Documentation section.

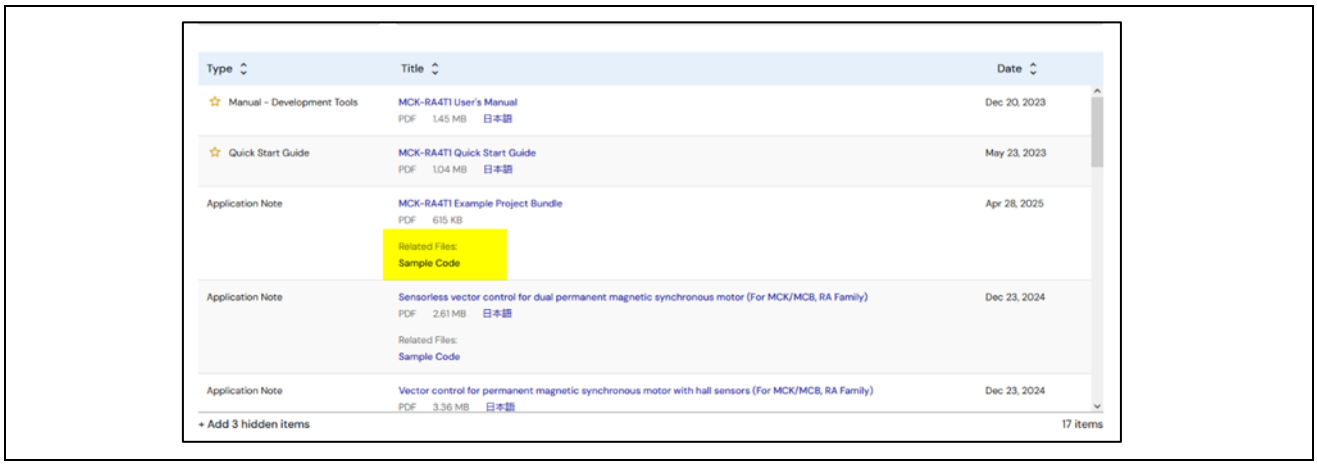


Figure 3. Where to Find Sample Code for the MCK-RA4T1

Alternatively, users can install example projects for Renesas evaluation boards from the renesas/ra-fsp-examples repository on GitHub.

The releases page contains EP bundles for individual kits. Each release description contains a list of evaluation boards and download links in the "Assets" section. It may be necessary to click on the "Show all assets" text to download the appropriate bundle.

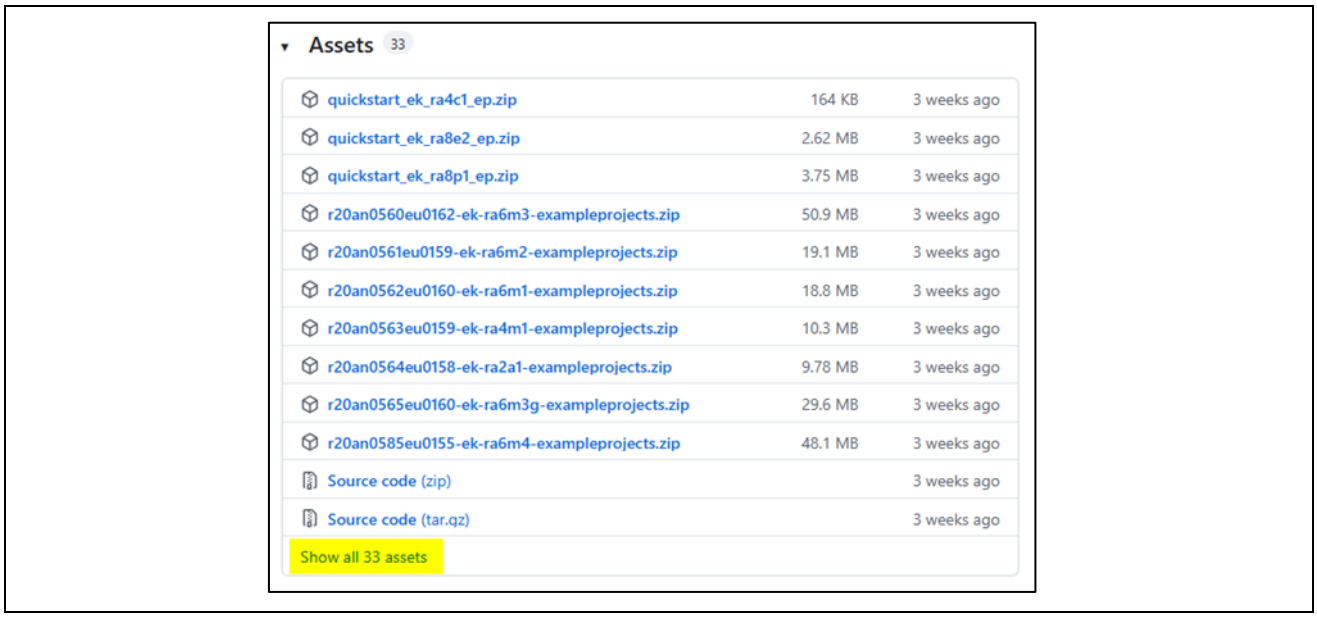


Figure 4. Releases Page for ra-fsp-examples Repository

To download the entire repository, either click the "Download ZIP" button under the "Code" dropdown, or clone the ra-fsp-examples repository using Git.

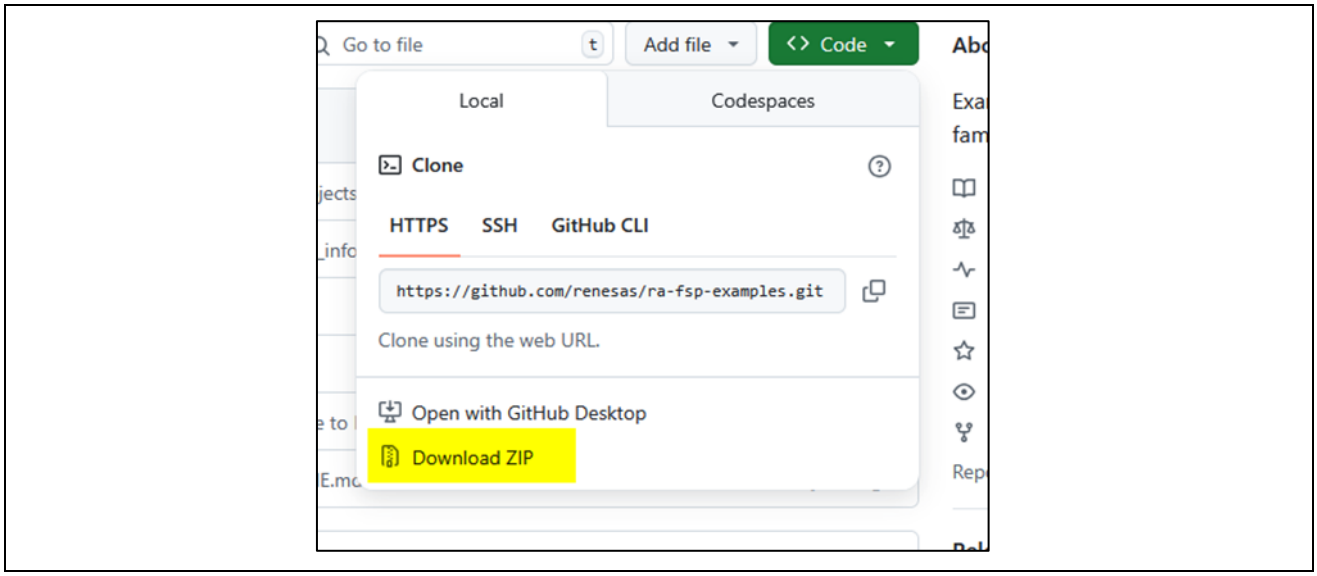


Figure 5. Downloading a GitHub Repository as a ZIP File

4.2 Replicate desired EP on MCK

To access the Import dialog box, click File → Import, or right click on the Project Explorer view and click Import.

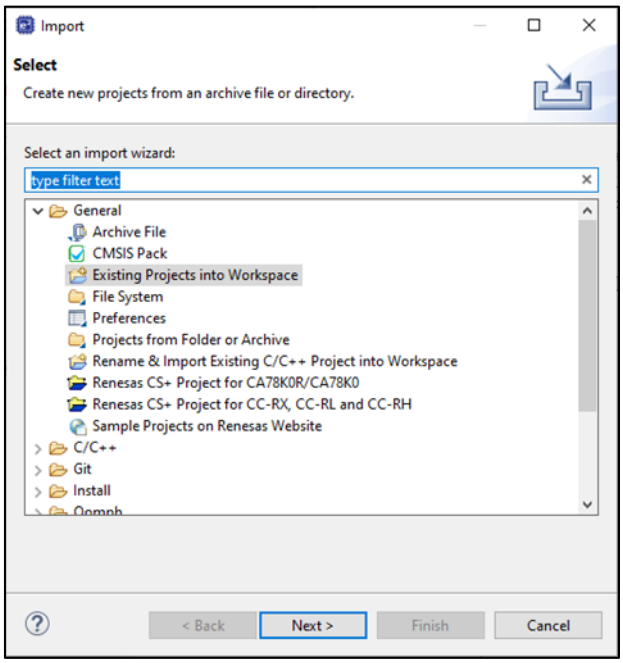


Figure 6. Import Menu in e² studio

Choose the "Existing Projects into Workspace" option. Click "Browse" and select the directory with the MCK-RA4T1 EPs. Under the Projects section, check the projects that should be imported into the workspace. Optionally, select "Copy projects into workspace" to create a new copy of the selected projects in the currently active workspace.

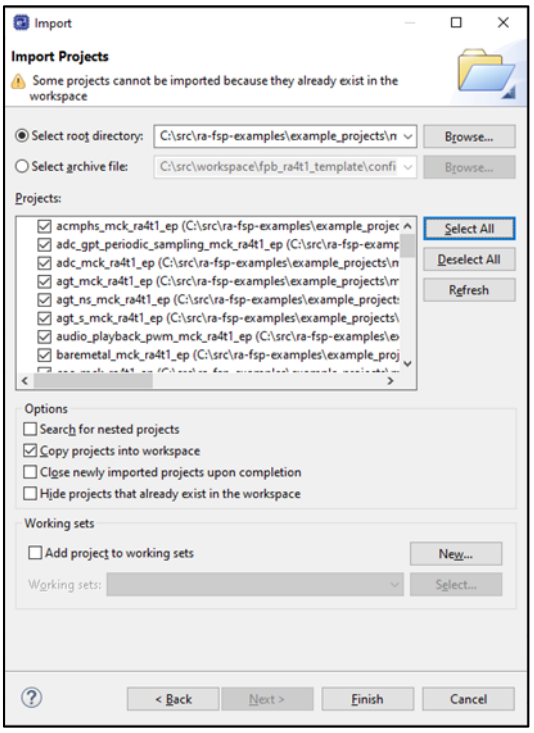


Figure 7. Importing Existing Projects into Workspace

An important part of porting an EP from the MCK-RA4T1 to the FPB-RA4T1 is understanding how the EP works. The example projects may include a readme.txt file to guide users through the evaluation or a Markdown file with additional information on stack setup and screenshots of results. After reading through

the provided instructions, ensure that the example project works by compiling and running the project. Make sure to verify that the output is as expected.

4.3 Create Template Project for FPB-RA4T1

One of the first steps in the porting process is creating a template project for the FPB-RA4T1. e² studio has a native way to create projects. RA Smart Configurator can also be used to set up FSP-based projects for other IDEs like IAR Embedded Workbench and Keil µVision 5.

4.3.1 Create a template project in e² studio

To start, click on File → New → Renesas C/C++ Project → Renesas RA as shown below.

1.

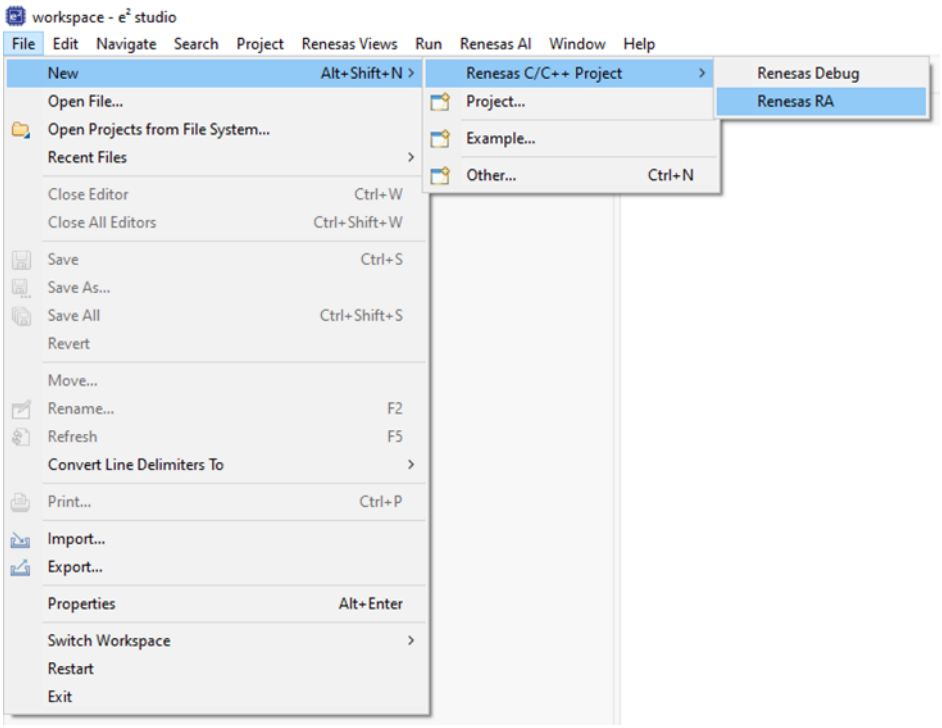


Figure 8. Create a New Renesas RA Project

This should open up the New C/C++ Project window. Choose Renesas RA C/C++ Project and click Next.

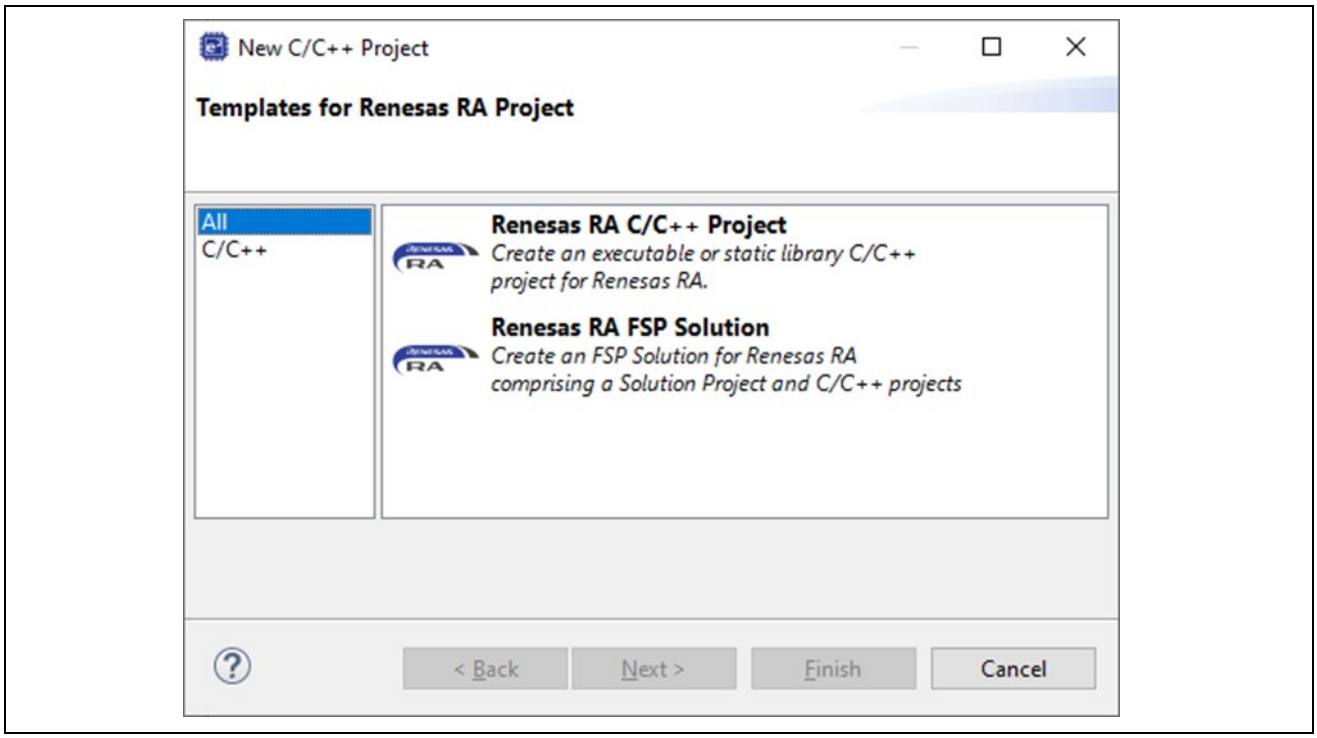


Figure 9. Choosing a Template for a Renesas RA Project

Enter the desired project name in the Project name textbox. Optionally, specify a custom location for the project in the Location textbox. Click Next.

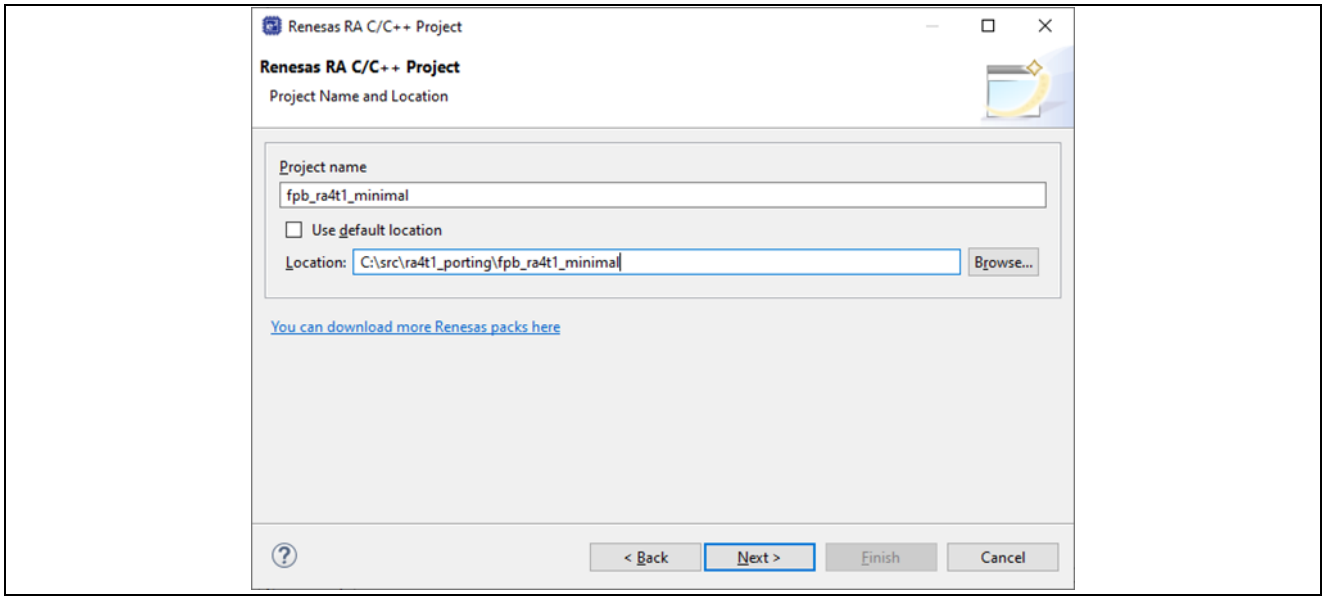


Figure 10. Project Name and Location

Under Device Selection, ensure that the FSP Version of the base project matches the version in the EPs you're trying to port. For example, if the project you're trying to port uses FSP version 5.9.0, choose 5.9.0 here. Click the button with the three dots next to the Board dropdown and choose the FPB-RA4T1. Alternatively, type in the Board dropdown textbox to search for the FPB-RA4T1. Choose the desired toolchain and IDE project type. Click **Next**.

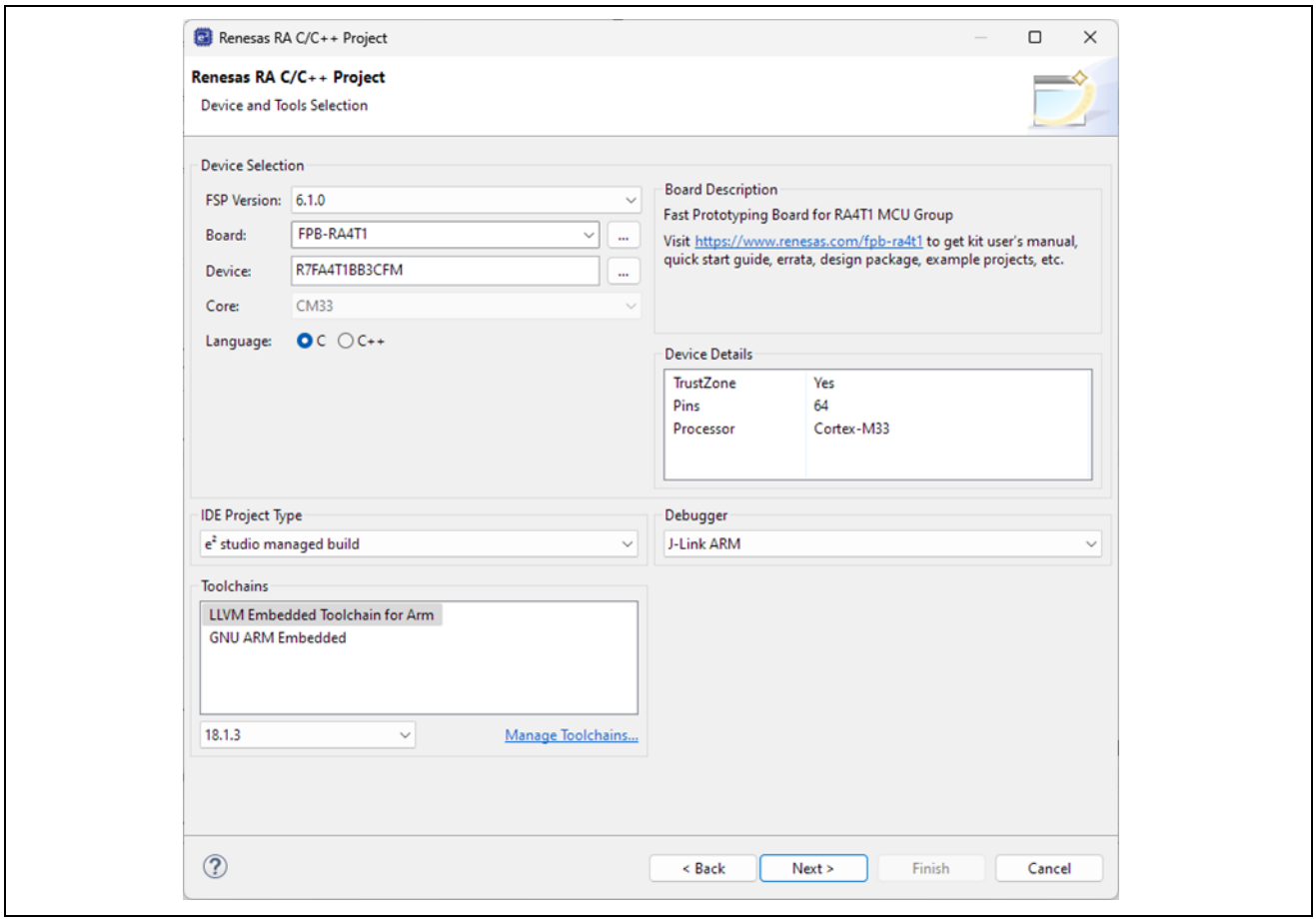


Figure 11. Device and Tools Selection

Choose the desired project type and click **Next**.

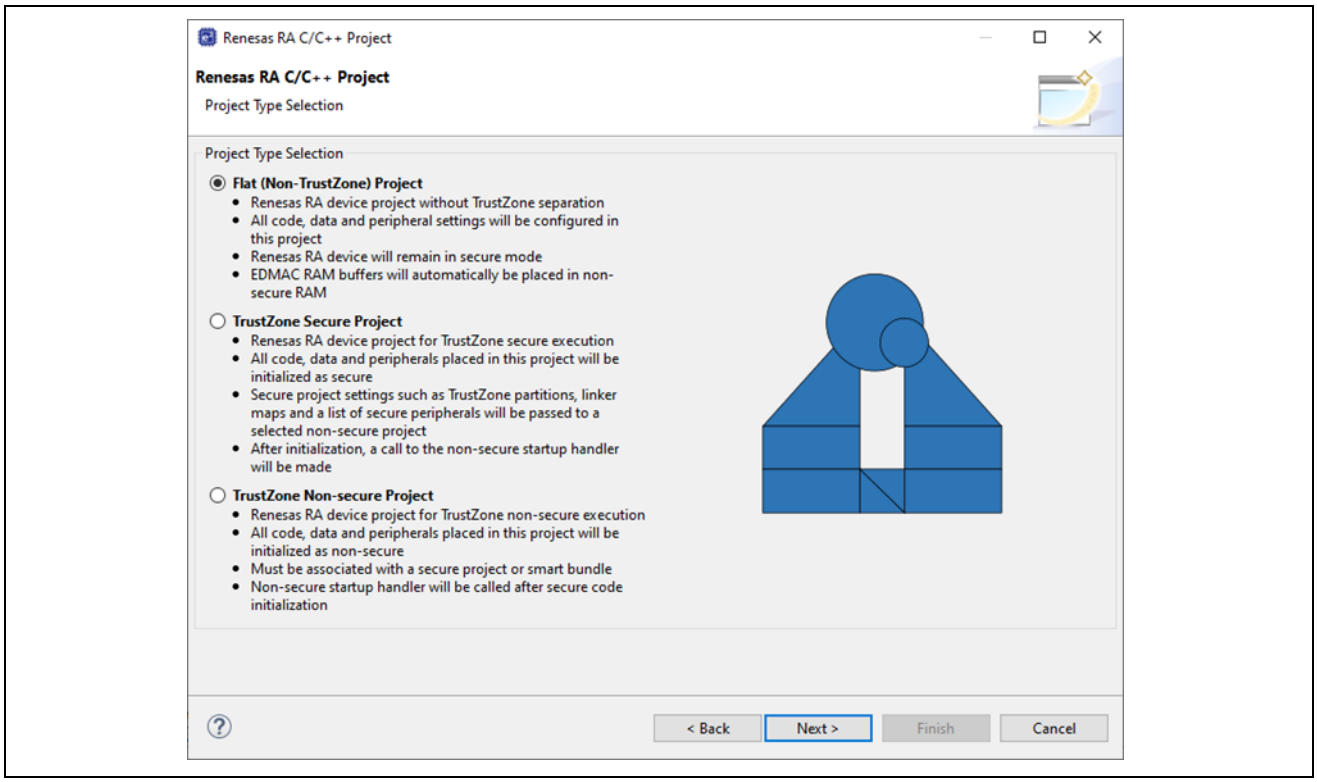


Figure 12. Project Type Selection

If you are setting up a TrustZone solution, choose the desired options. Otherwise, click **Next**.

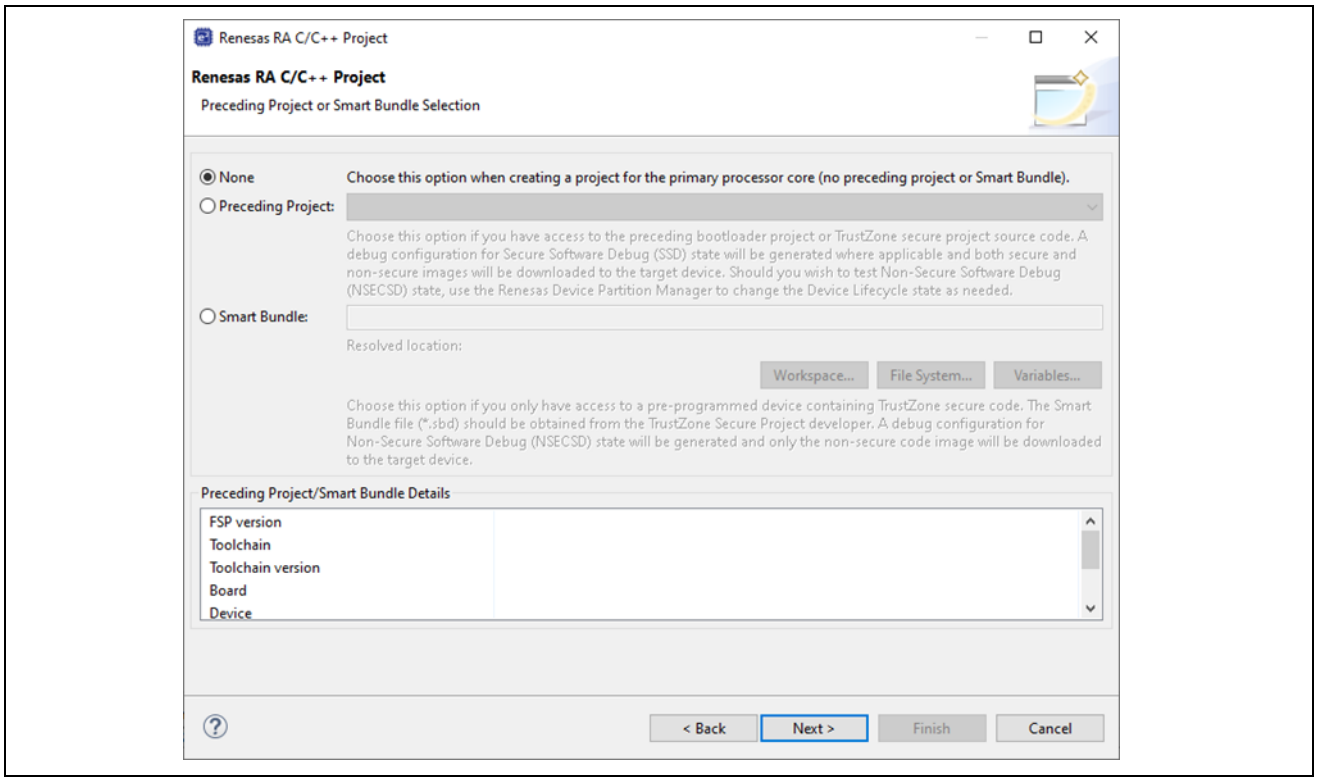


Figure 13. Precedint Project or Smart Bundle Selection

Choose an RTOS under the RTOS Selection dropdown. Make sure this is correct, as this cannot be changed in e² studio after creating the project. Click **Next**.

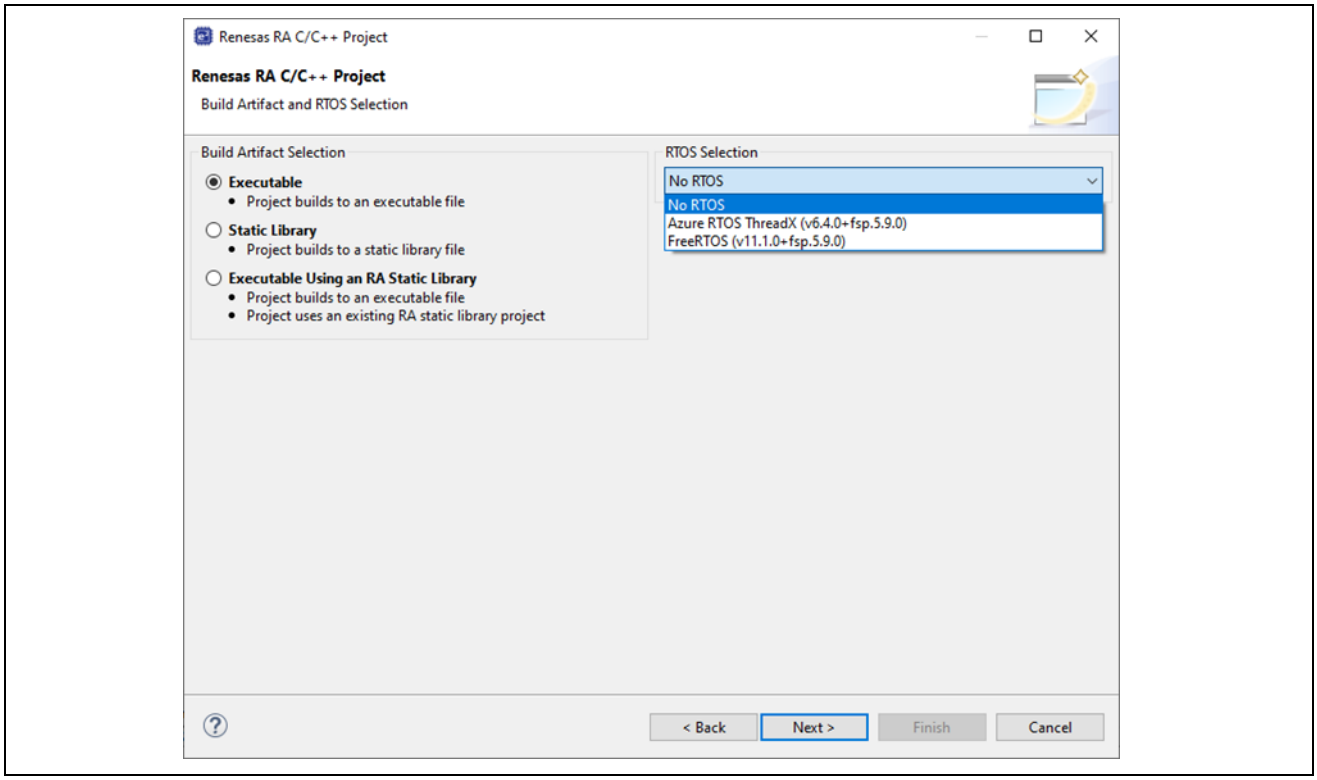


Figure 14. Build Artifact and RTOS Selection

Choose the "Bare Metal - Minimal" option and click **Finish**.

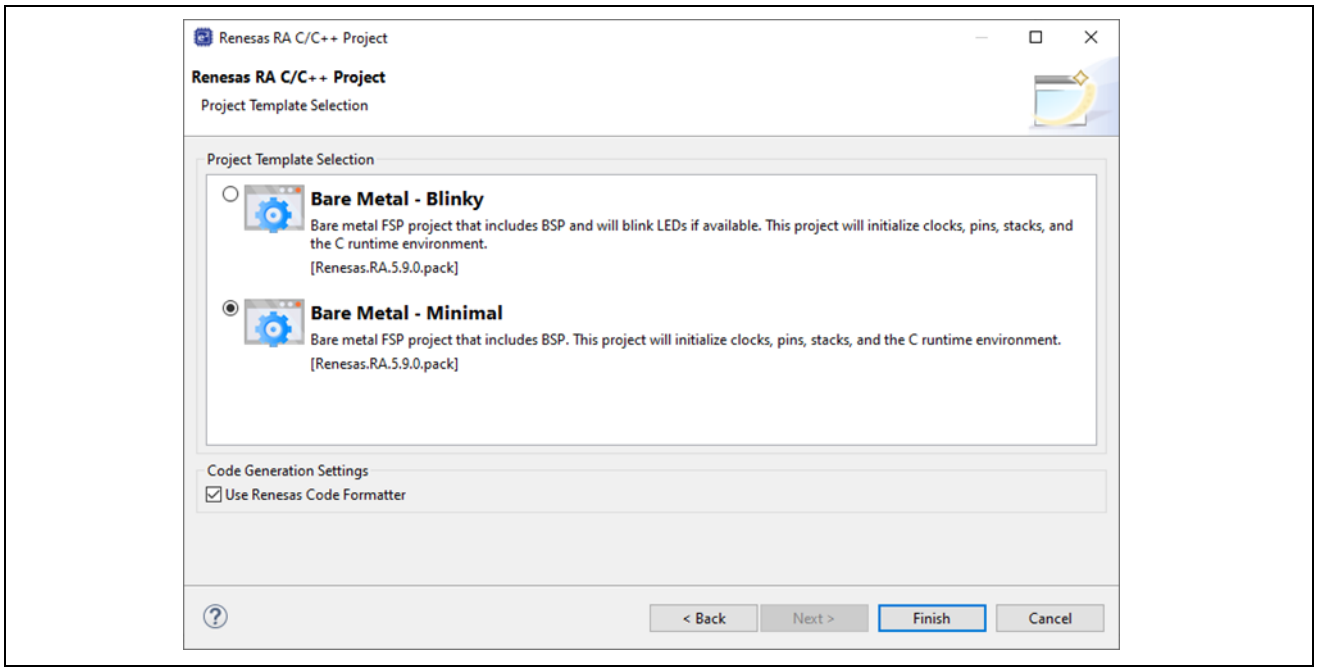


Figure 15. Build Artifact and RTOS Selection

Your new template project should appear in Project Explorer tab. Its folder structure should look like this.

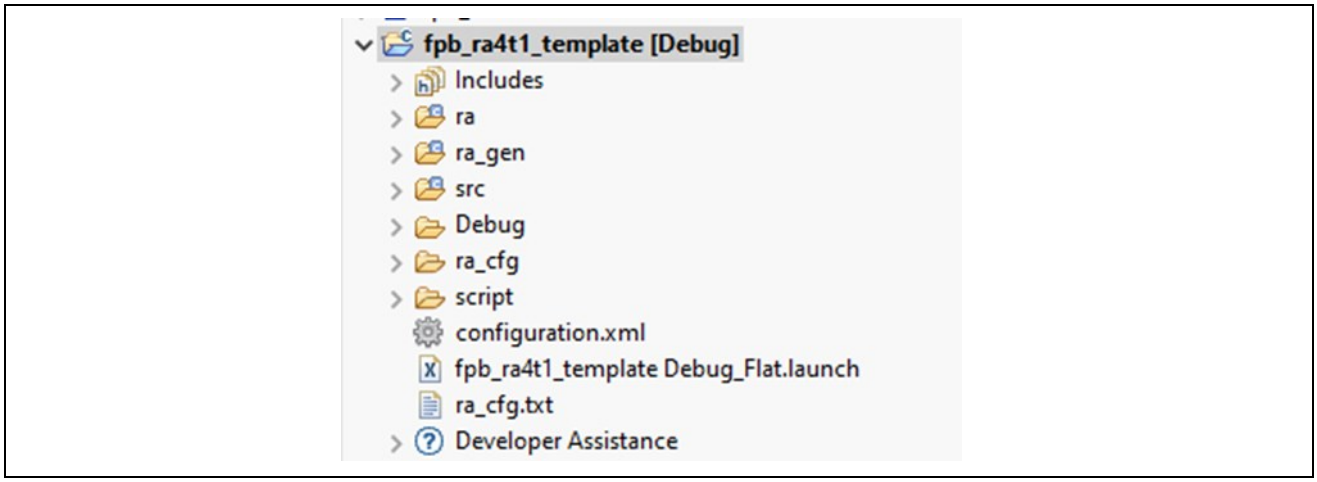


Figure 16. Folder Structure for Template Project

Warning: BSP for FSP v5.9.0 or Earlier has Incorrect Default Clock Setting on FPB-RA4T1

In FSP versions below 6.0.0, the default clock configuration for the FPB-RA4T1 is incorrect. Because the external clock crystal (XTAL) is not populated on the FPB-RA4T1 out of the box, it is necessary to configure the template project to use the High-speed On-chip Oscillator (HOCO) and change the PLL settings to output 200MHz. To perform this fix, first double click on the configuration.xml file. If the Open Associated Perspective dialog box appears, click Open Perspective.

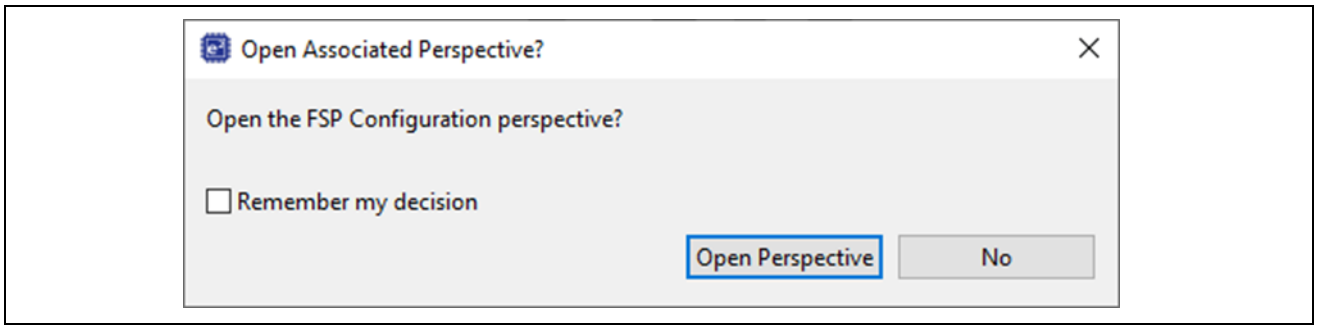


Figure 17. Open Associated Perspective Dialog

A graphical interface should open with multiple menu items on the bottom of the panel, and a "Generate Project Content" button on the top right corner of the panel.

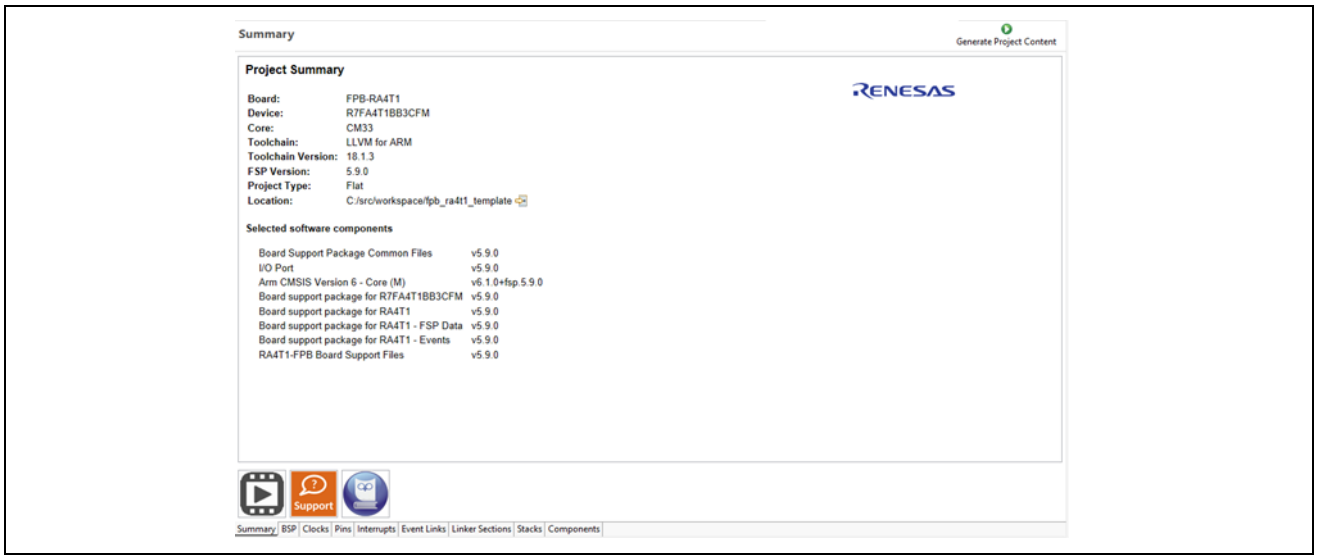


Figure 18. FSP Configuration Summary

Click on the Clocks menu option and modify the highlighted clock settings as shown below.

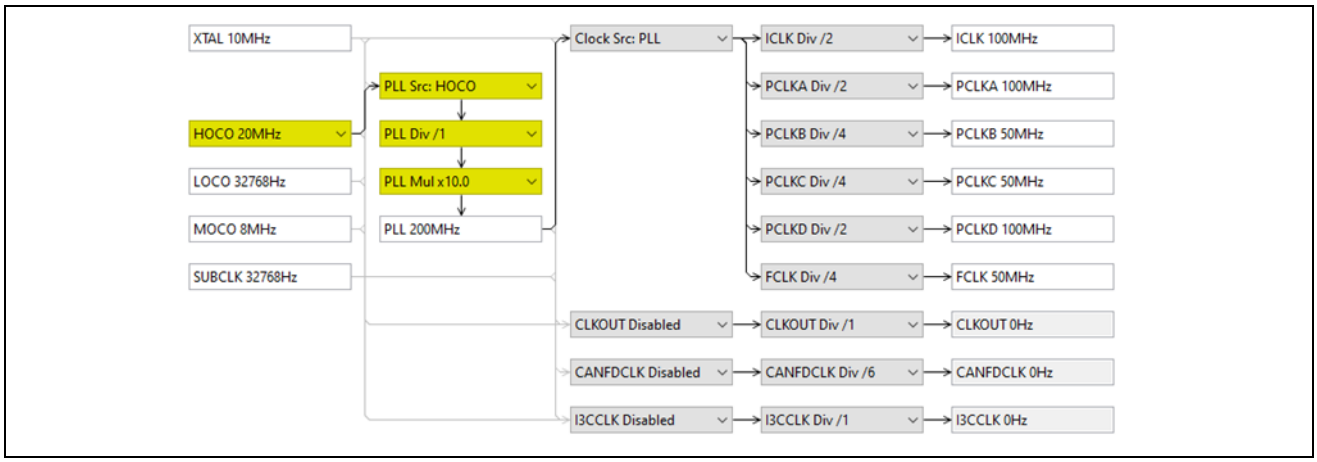


Figure 19. Clocks Menu with Proper Clock Settings Highlighted

After changing these settings, click the Generate Project Content button.

A good step to take after creating a minimal project is to compile and run it. After clicking the resume button twice to advance past the default breakpoints, press the pause button. If the debugger isn't stopped in a while loop as shown below, then the clocks may be configured wrong, or your hardware may be incorrectly configured or faulty.

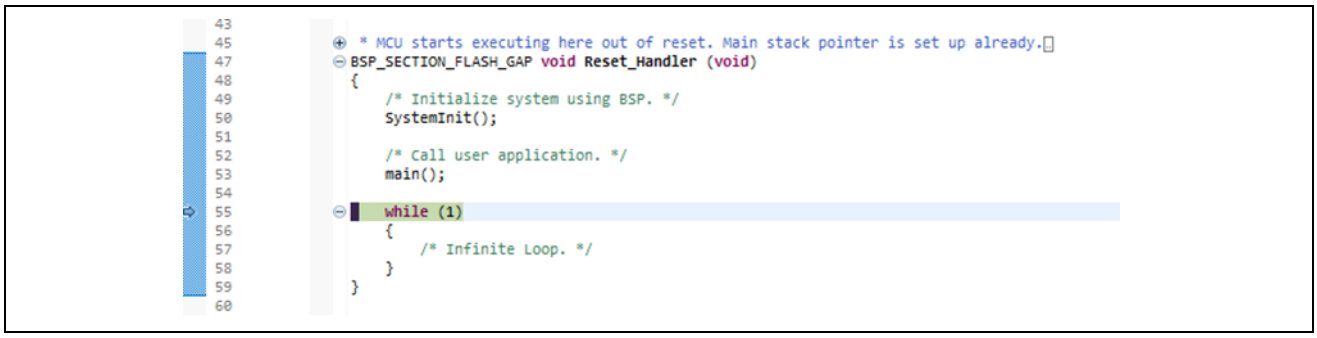


Figure 20. Clocks Menu with Proper Clock Settings Highlighted

4.3.2 Create a template project in RA Smart Configurator

The Renesas RA Smart Configurator is a standalone version of the tools included in e² studio for generating and configuring projects. When opening the RA Smart Configurator, a dialog box similar to the "Renesas RA C/C++ Project" dialog in e² studio appears. This dialog can also be accessed through File → New → FSP Project.

The Device and Tools Selection step has a different list of compilers in IDE Project Type. Choose the desired project and click next. After this point, the workflow should be identical to Section 4.3.1.

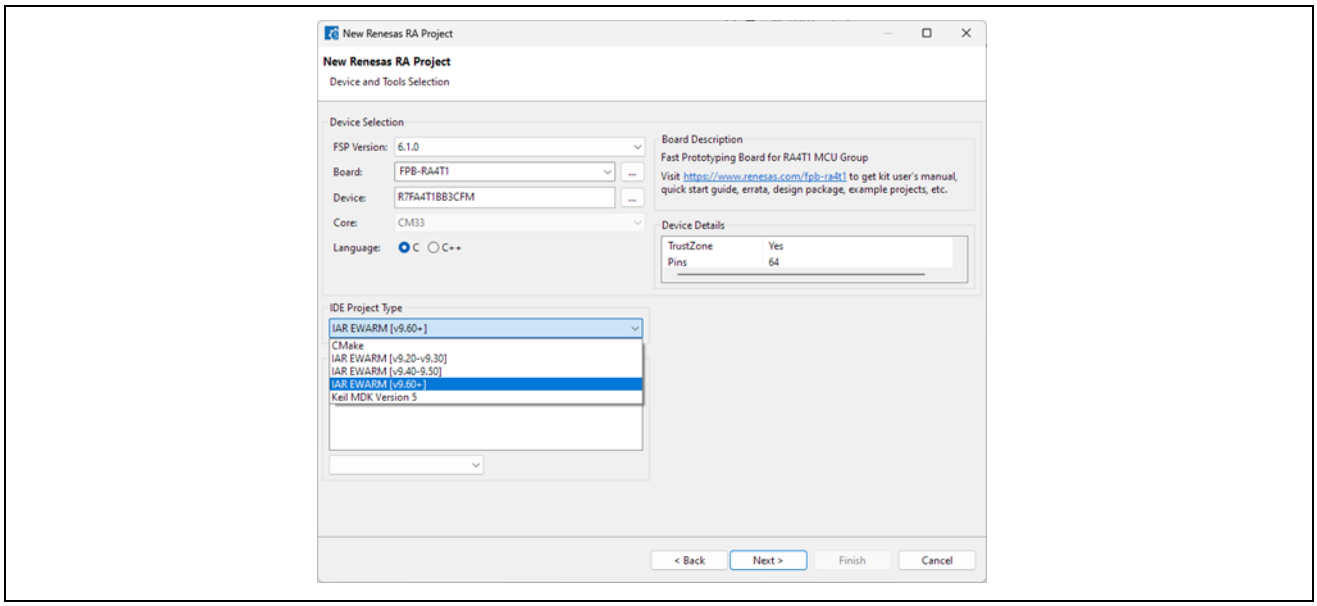


Figure 21. Device and Tools Selection in RA Smart Configurator

After creating a new project in the RA Smart Configurator, a view identical to the FSP Configuration view in e² studio should appear. If applicable, change the clock configuration as outlined in Section 4.3.1.

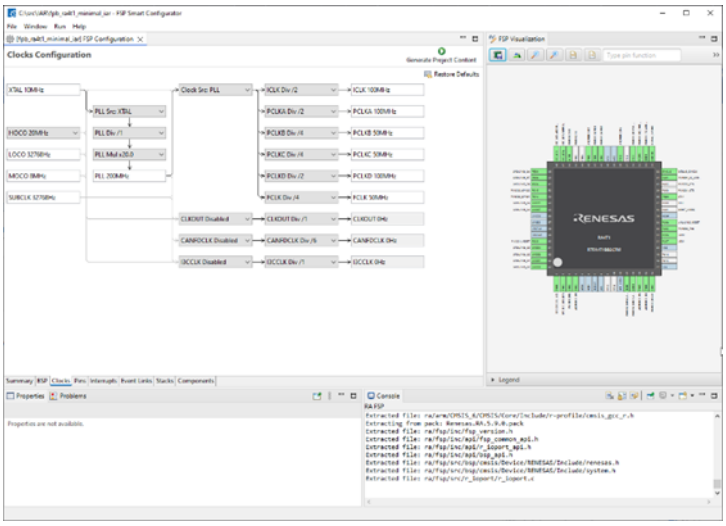


Figure 22. Configuring an FSP Project Through RA Smart Configurator

To learn more about how to use IAR EAW or Keil µVision 5 with RA Smart Configurator, read the RASC User Guide for MDK and IAR in the RA FSP Documentation.

5. Example: Porting TFU EP

This first example project demonstrates the Trigonometric Function Unit on Renesas RA MCUs. Trigonometric functions like sinf(), cosf(), arctan2f(), and hypotf() are used very extensively for tasks like motor control that require processing angular data. By enabling the "TFU Mathlib" option in the BSP configuration, the previously mentioned functions in math.h are accelerated using the on-chip TFU module.

The TFU example project does not rely on any external hardware, making it a good first step for learning the porting process. The Renesas RA FSP enables developers to write code using high-level APIs that remain consistent across different MCUs and kits. This flexibility allows for code and configurations in the TFU EP to remain unchanged between the MCK-RA4T1 and FPB-RA4T1.

5.1 Setting Up a New Project

To create a project, follow the steps outlined in Section 4.3. It may be time-consuming for users to follow the steps in Section 4.3 for each project that they want to port. To alleviate this, users can duplicate the template project by copying and pasting it in the Project Explorer panel.

When copy-pasting a project in the Project Explorer panel, a "Copy Project" dialog box will appear, asking the user to add a new name to the project. Name the new example project as desired, and click "Copy".

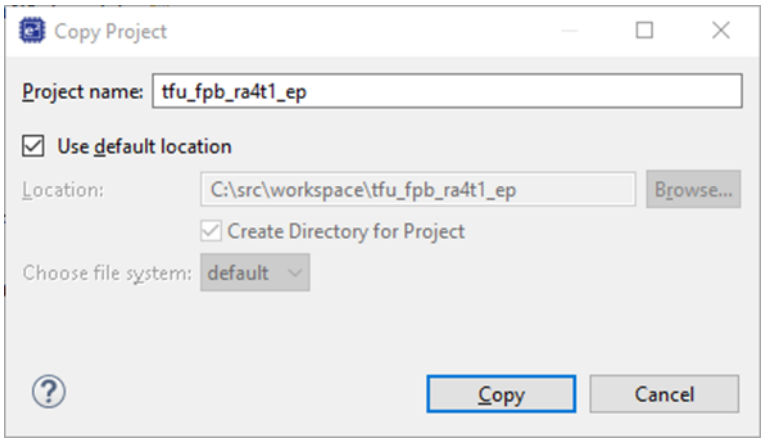


Figure 23. Copying Template Project

A new project with the desired name should appear in the workspace. One thing to note is that in current versions of e² studio, the debugger configuration does not change when copying a project. To create a working debugger configuration, right click on the new project in the Project Explorer window and select "Close Unrelated Projects" to close all other projects.

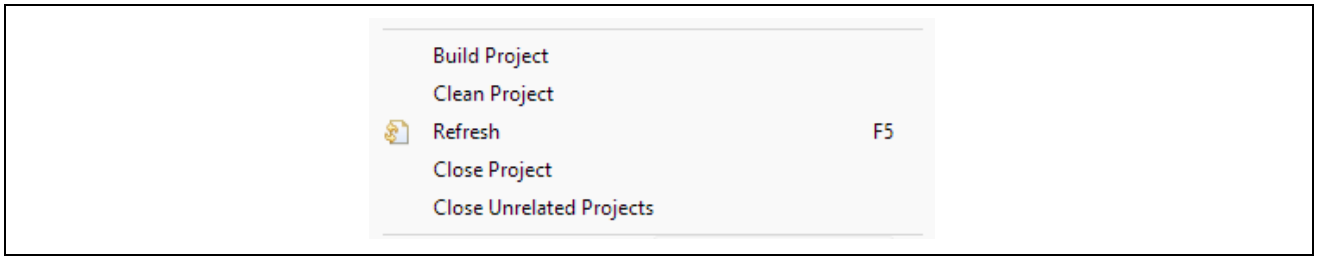


Figure 24. Right Click Menu with Close Unrelated Projects Option

Build the project, then click the "Debug" button. A dialog box should appear asking for a way to debug the project. Choose "Renesas GDB Hardware Debugging" and click OK.

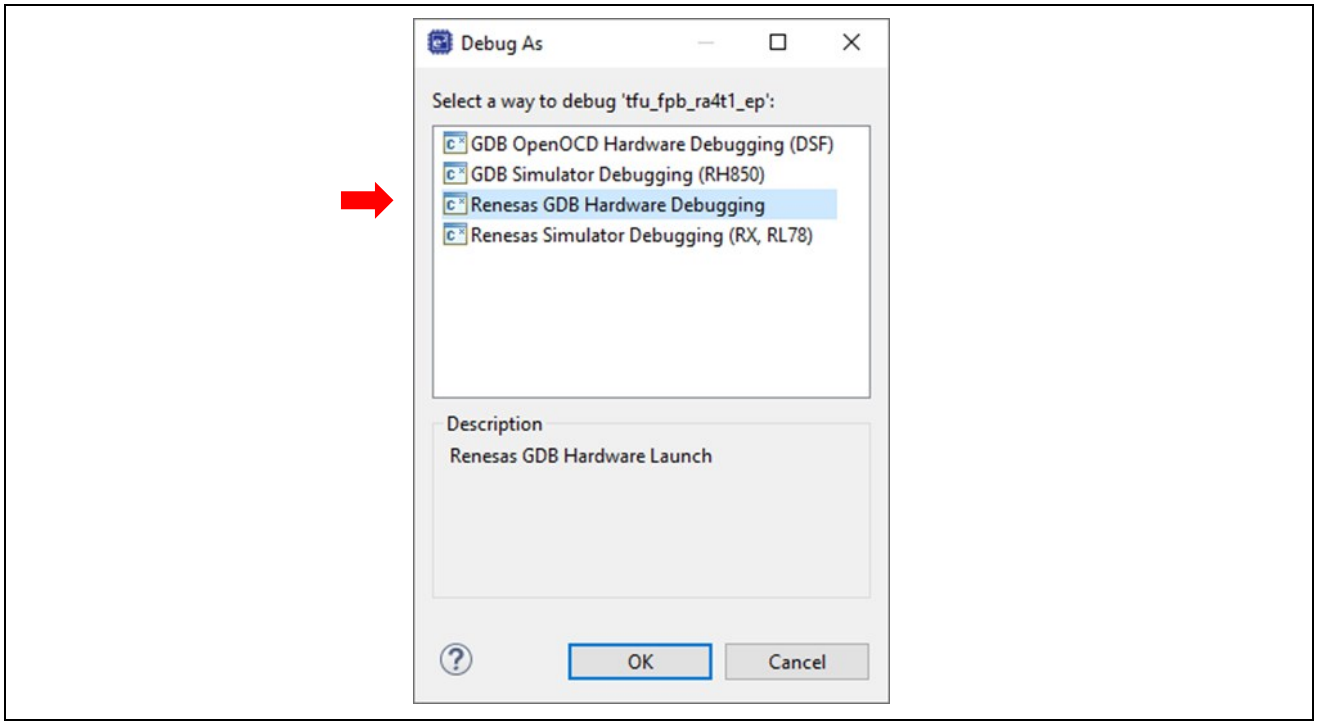


Figure 25. Debug with Renesas GDB Hardware Debugging

In the next menu, choose "J-Link ARM" and click OK.

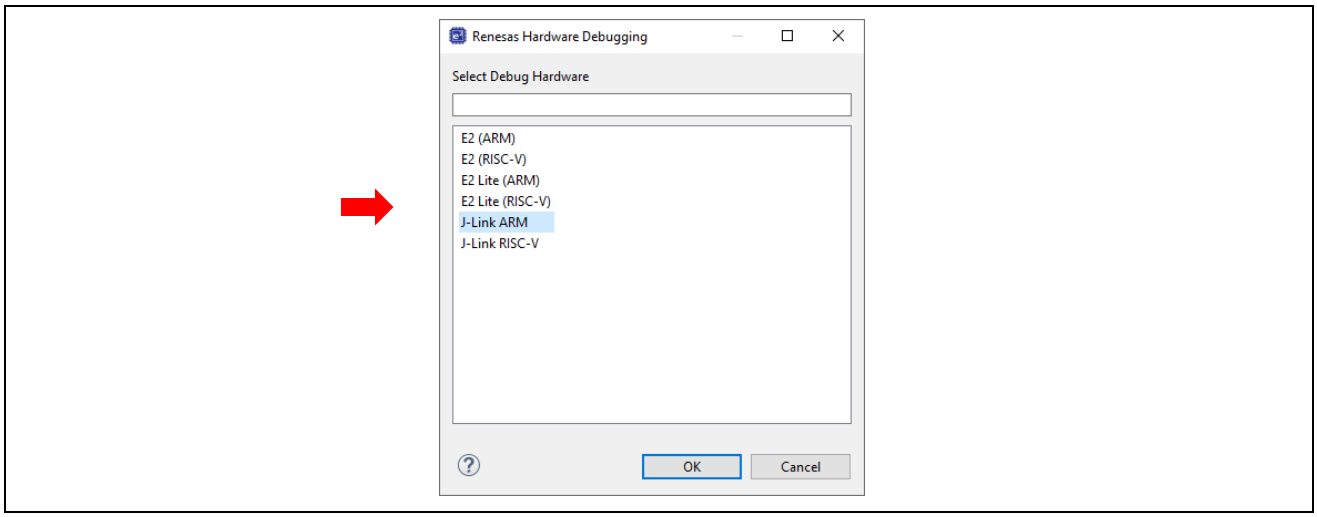


Figure 26. Select Debugging Hardware

Search for "R7FA4T1BB" and click **OK**.

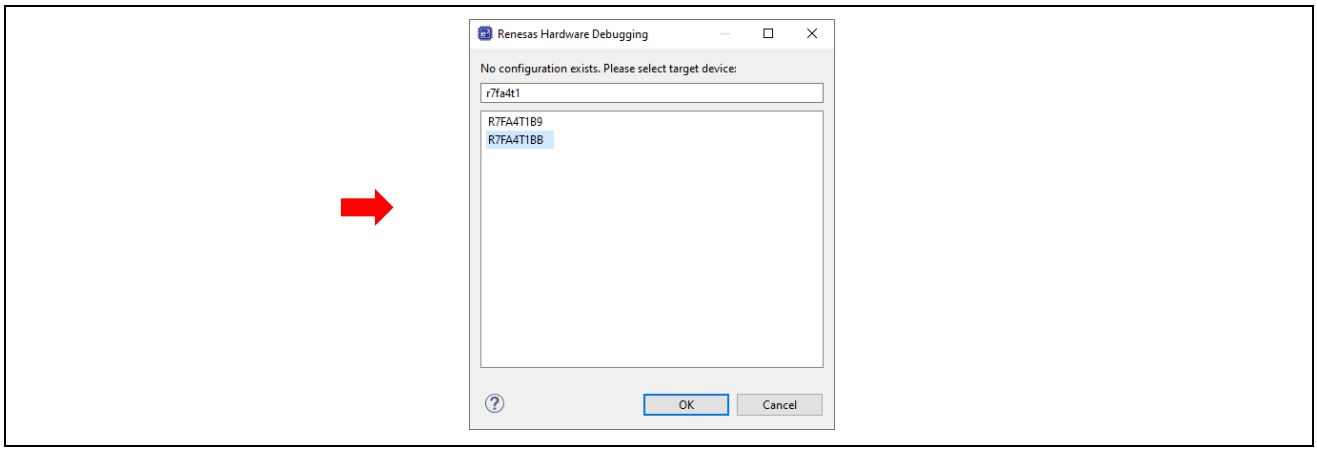


Figure 27. Select Target Device for Debugging

After this point, the new project should be debuggable. You can optionally remove the old ".launch" file, or change debugger configurations through the "Debugger Configurations" option in the debugger dropdown.

5.2 Moving Code Between Projects

2. To transfer code from one project to another, open the "src" folder of the source project in the Project Explorer view, select all of the code files with Shift+Click, and Right Click → Copy (Ctrl+C). Then, click on the "src" folder in the destination project and Right Click → Paste (Ctrl+V).

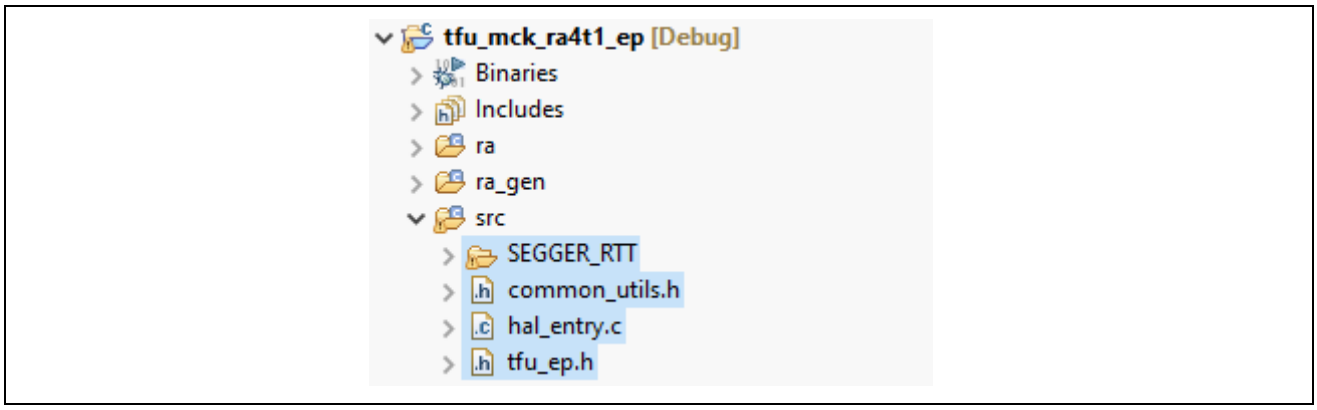


Figure 28. Selecting All Files in "src" Folder

A dialog box should open asking if you want to overwrite "hal_entry.c". Click "Yes To All".

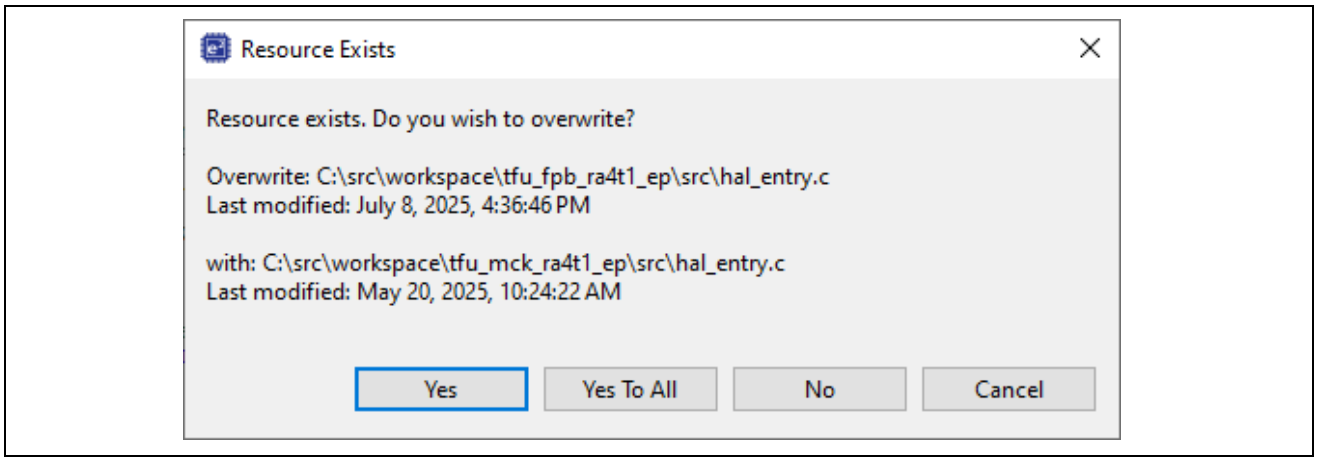


Figure 29. Resource Exists Dialog

In IAR Embedded Workbench, a similar process can be done by moving the source code files. When reopening a project, IAR EW will automatically add all files in the "src" folder to the Program Entry group.

If you are using Keil µVision 5, you can add new source files to the project by right-clicking "Source Group 1" and clicking 'Add Existing Files to "Source Group 1"'.

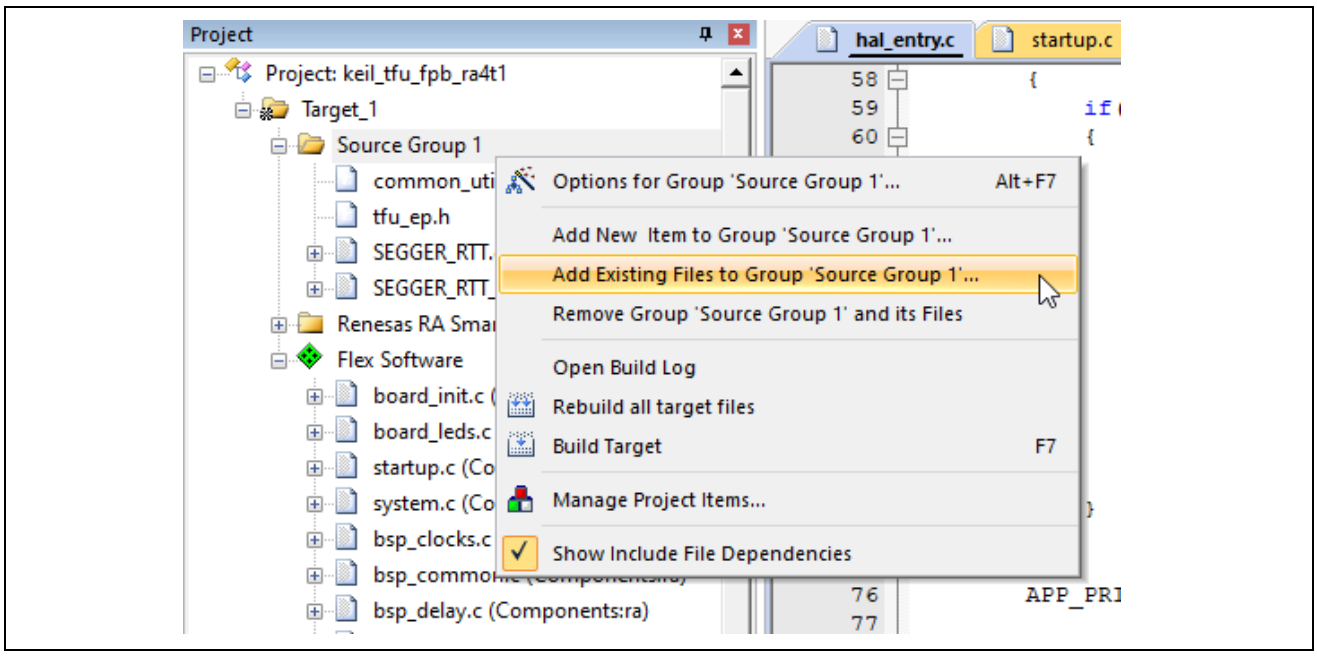


Figure 30. Add Existing Files to Group in Keil µVision 5

5.3 Expected Results

After building the project and starting a debug session, open J-Link RTT Viewer. A window should appear as shown below. Select "Existing Session", and in the RTT Control Block options, select "Search Range" and enter "0x20000000 0x40000" into the textbox.

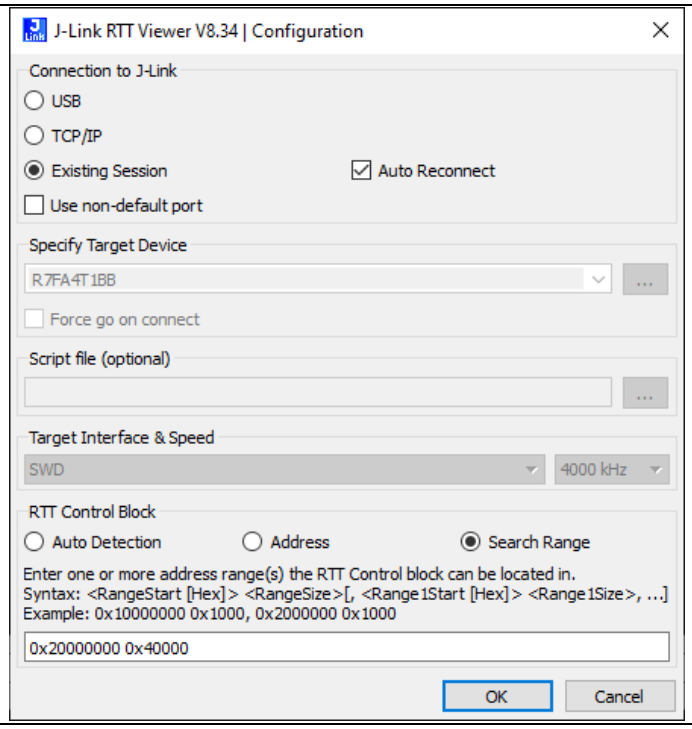


Figure 31. J-Link RTT Viewer Configuration

Press play two times. When prompted, enter a number from 4 to 128 into the textbox below the console view and press "Enter". The EP should output a range of values for each trigonometric function that the TFU supports.

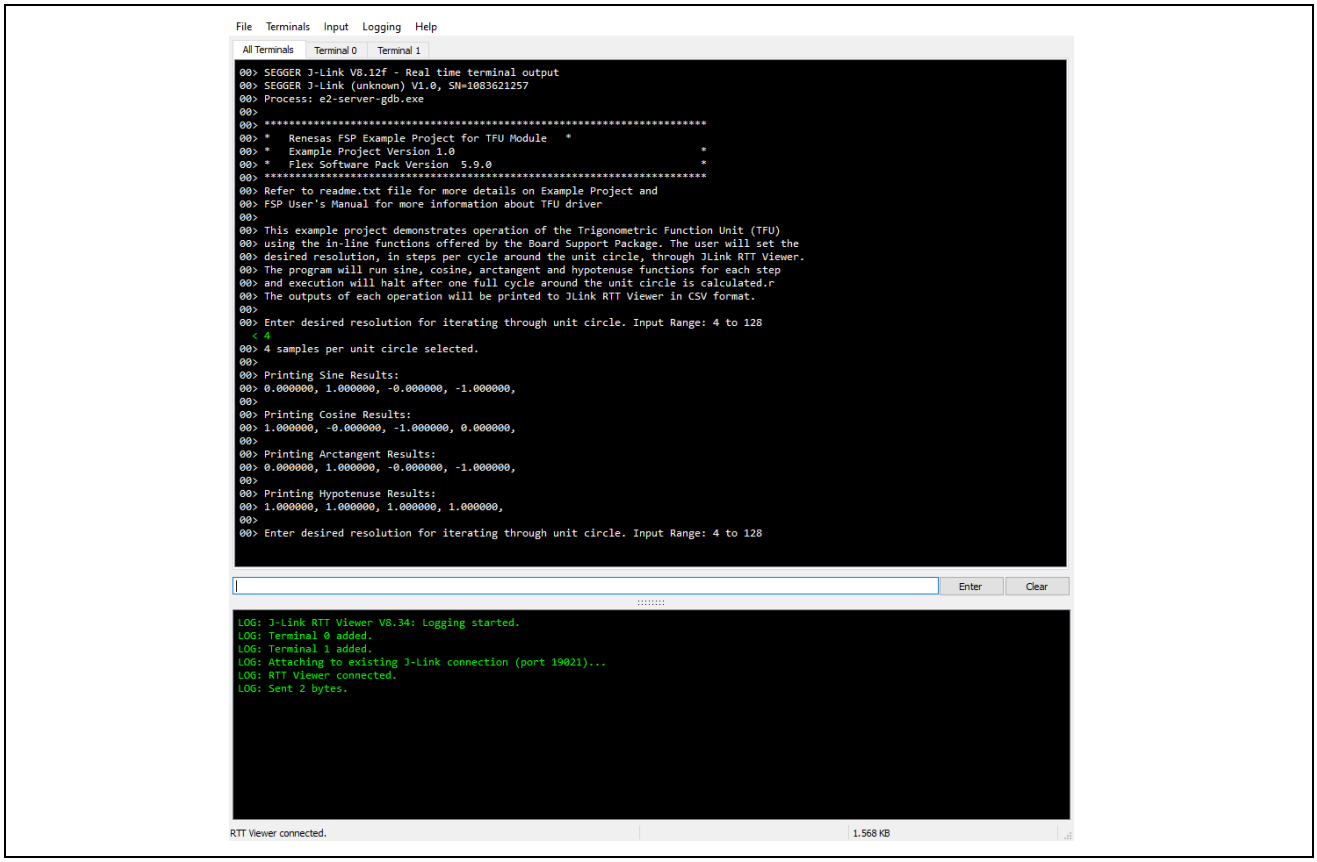


Figure 32. Expected Results for TFU EP

6. Example: Porting DAC_AGT_SIGNAL_GENERATOR EP

This example project sets up multiple hardware modules to output an analog signal independently of the internal ARM core. The Asynchronous General-Purpose Timer (AGT) module is configured to trigger an interrupt periodically, which activates the Device Transfer Controller (DTC). The DTC is then configured to initiate a data transfer after every AGT interrupt, moving data sequentially from a lookup table to the DAC data register. After every transfer, the 12-bit DAC performs a digital to analog conversion and outputs the resulting voltage to a pin.

6.1 Import Stacks

Follow the steps outlined in Section 5 for the DAC AGT Signal Generator EP. However, when building the ported version, the compiler will output an assortment of errors, as shown below.

```
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_delay.c
../src/SEGGER_RTT/SEGGER_RTT_printf.c: In function 'SEGGER_RTT_vprintf':
../src/SEGGER_RTT/SEGGER_RTT_printf.c:384:23: warning: conversion to 'unsigned int' from 'int' may change the sign of the result [-Wsign-conversion]
  384 |           Precision = va_arg(*pParamList, int);
      |                       ^~~~~~
../src/hal_entry.c: In function 'hal_entry':
../src/hal_entry.c:61:11: warning: implicit declaration of function 'R_DAC_Open' [-Wimplicit-function-declaration]
   61 |     err = R_DAC_Open (&g_dac0_ctrl, &g_dac0_cfg);
      |           ^~~~~~~~~~
../src/hal_entry.c:61:24: error: 'g_dac0_ctrl' undeclared (first use in this function)
   61 |     err = R_DAC_Open (&g_dac0_ctrl, &g_dac0_cfg);
      |                       ^~~~~~~~~~~~
../src/hal_entry.c:61:24: note: each undeclared identifier is reported only once for each function it appears in
../src/hal_entry.c:61:38: error: 'g_dac0_cfg' undeclared (first use in this function)
   61 |     err = R_DAC_Open (&g_dac0_ctrl, &g_dac0_cfg);
      |                                      ^~~~~~~~~~~
../src/hal_entry.c:68:11: warning: implicit declaration of function 'R_DAC_Start' [-Wimplicit-function-declaration]
   68 |     err = R_DAC_Start (&g_dac0_ctrl);
      |           ^~~~~~~~~~~
../src/hal_entry.c:76:11: warning: implicit declaration of function 'R_DTC_Open' [-Wimplicit-function-declaration]
   76 |     err = R_DTC_Open(&g_transfer0_ctrl, &g_transfer0_cfg);
      |           ^~~~~~~~~~
../src/hal_entry.c:76:23: error: 'g_transfer0_ctrl' undeclared (first use in this function)
   76 |     err = R_DTC_Open(&g_transfer0_ctrl, &g_transfer0_cfg);
      |                       ^~~~~~~~~~~~~~~~
../src/hal_entry.c:76:42: error: 'g_transfer0_cfg' undeclared (first use in this function)
   76 |     err = R_DTC_Open(&g_transfer0_ctrl, &g_transfer0_cfg);
      |                                          ^~~~~~~~~~~~~~~
../src/hal_entry.c:83:11: warning: implicit declaration of function 'R_DTC_Enable'; did you mean 'ARM_MPU_Enable'? [-Wimplicit-function-declaration]
   83 |     err = R_DTC_Enable(&g_transfer0_ctrl);
      |           ^~~~~~~~~~~~
      |           ARM_MPU_Enable
../src/hal_entry.c:91:11: warning: implicit declaration of function 'R_AGT_Open'; did you mean 'R_IOPORT_Open'? [-Wimplicit-function-declaration]
   91 |     err = R_AGT_Open(&g_timer0_ctrl, &g_timer0_cfg);
      |           ^~~~~~~~~~
      |           R_IOPORT_Open
```

Figure 33. Compiler Warnings

These errors occur because the FSP API functions and structures corresponding to the DAC, AGT, and DTC stacks are absent from the /ra/fsp/ folder. Porting this project from the MCK-RA4T1 to the FPB-RA4T1 requires adding new stacks to the destination project with the same configurations as the source project.

Stack selections and stack-specific properties are stored in a project's configuration.xml file. It is possible to export the properties of a stack by right clicking on the stack in the "Stacks" section of the FSP Configuration window, but this only works for one stack at a time. Fortunately, there is a way to import all of a project's stacks.

First, while in the Stacks view, delete the pre-existing "g_ioport" stack. Then, right click on the "HAL/Common Stacks" section and click "Import". In the "From file" box, choose the source project's configuration.xml. This will populate all of the source project's stacks into a checklist in "Stacks". Click "Select All", check "Include common properties" and click Finish.

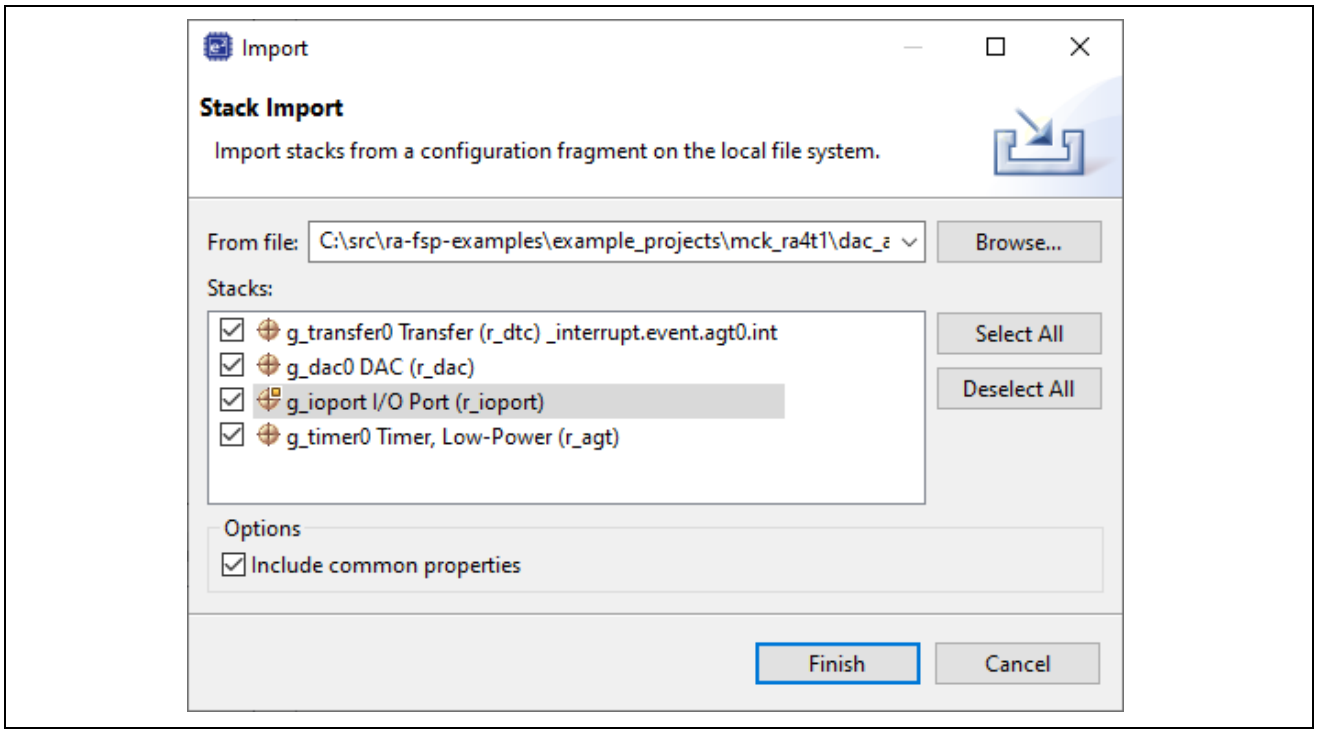


Figure 34. Import Stacks from Another Project

6.1.1 About the "Include common properties

It is important to note that the "Include common properties" option may not be desirable in some cases.

Stack properties are divided into common properties, which apply to every instance of an FSP module, and per-instance modules, which are individual to each module instance. In the Properties view, common properties and per-instance properties are grouped into separate dropdowns. These different sections can be seen in the image below.

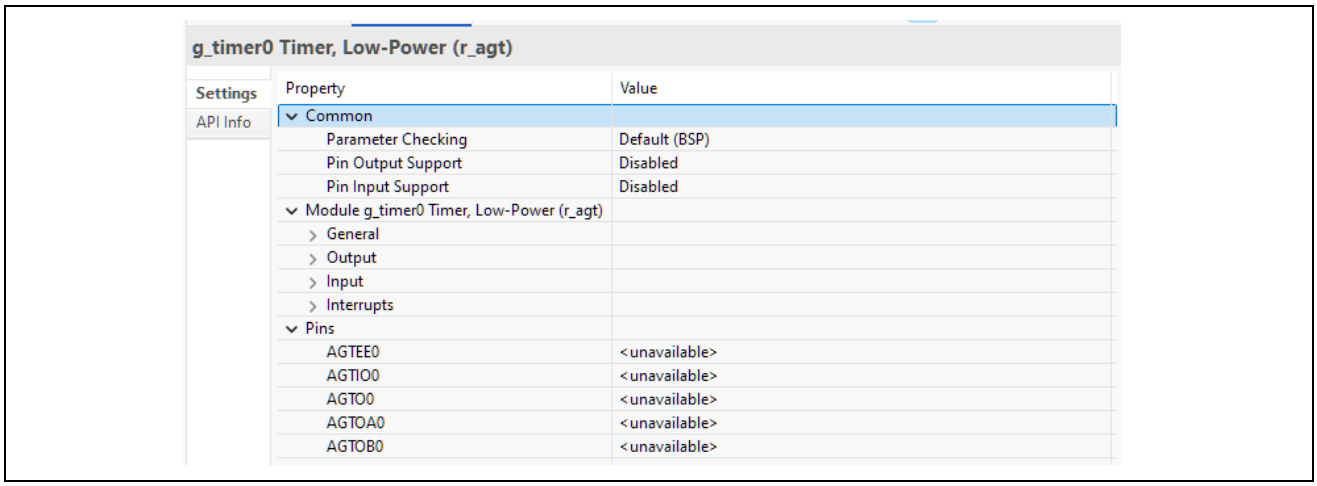


Figure 35. g_timer0 Properties

The "Include common properties" checkbox controls whether these common properties are copied into the current project. When copying stacks to a new project, this option is desirable. However, copying stacks to an existing project with this option selected may overwrite existing common properties.

After this, all four stacks should be present. Press Ctrl+S to save. There may be an error on the DTC stack saying that it requires an activation source.

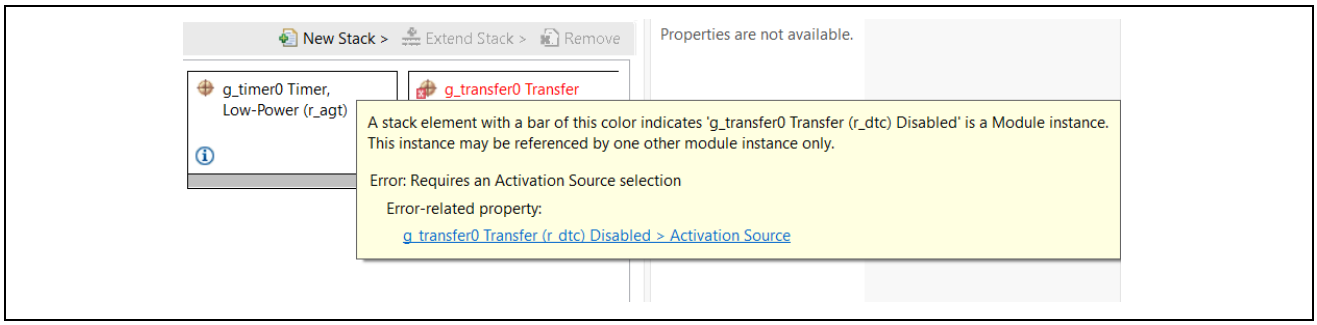


Figure 36. g_transfer0 Stack Error

Closing and reopening the destination project's configuration file will allow the AGT0 Interrupt to be selected under "activation source" in the g_transfer0 configuration. Choose this option and save the configuration.

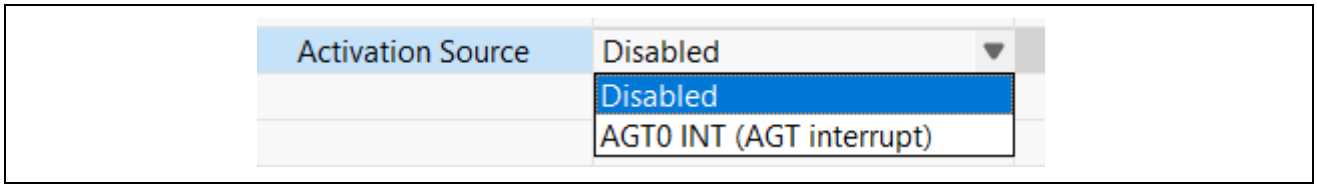


Figure 36. g_transfer0 Stack Error

6.2 Change Pin-Peripheral Assignment

This EP requires additional pin configuration. This project uses channel 0 of the 12-bit DAC, which can be connected to P014 on the R7FA4T1BB. P014 is accessible on the FPB-RA4T1 via the PMOD2 header. However, P014 is configured by default to act as a GPIO pin. To change the default pin configuration, select the "Pins" section of the configuration view. Then, in the Pin Selection menu, select Peripherals > Analog:DAC12 > DAC120. In the Pin Configuration section, change "Operation Mode" to "Enabled" and set "DA0" to "P014". This will lead to a pin conflict as shown below.

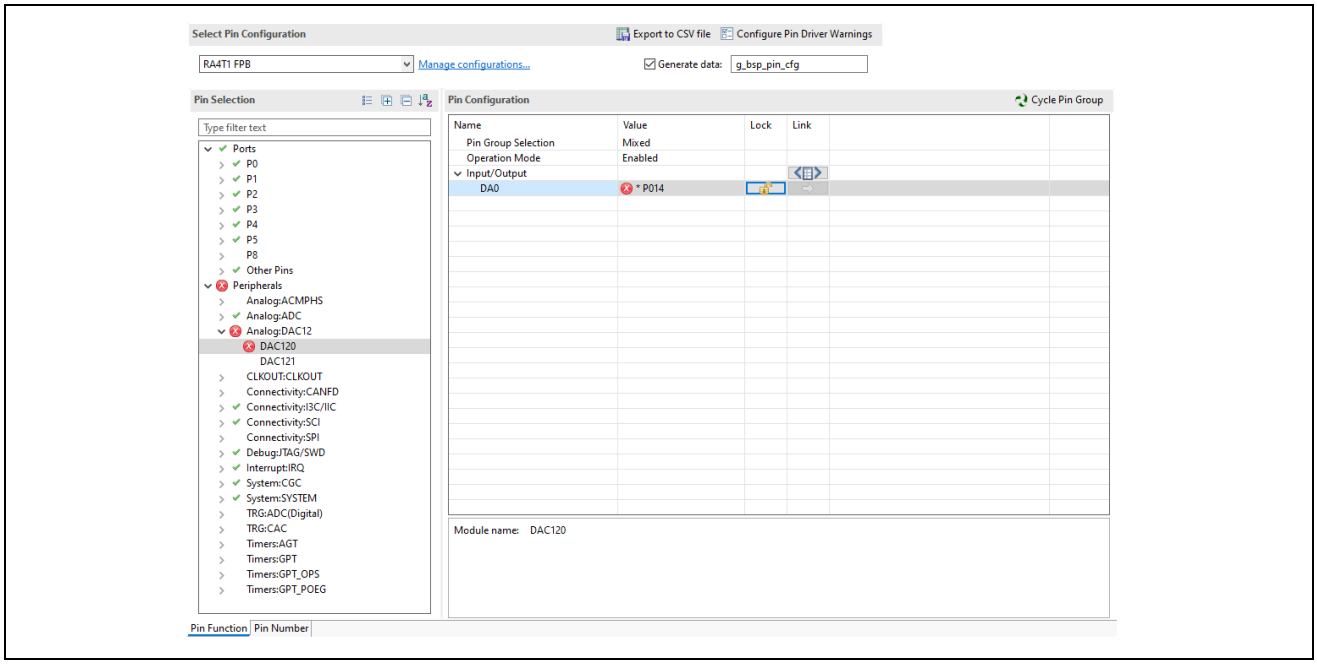


Figure 38. : 12-bit DAC, Channel 0 Configuration

To resolve this conflict, go to the P014 configuration menu under the "Pins" section by typing "P014" into the search box as shown below.

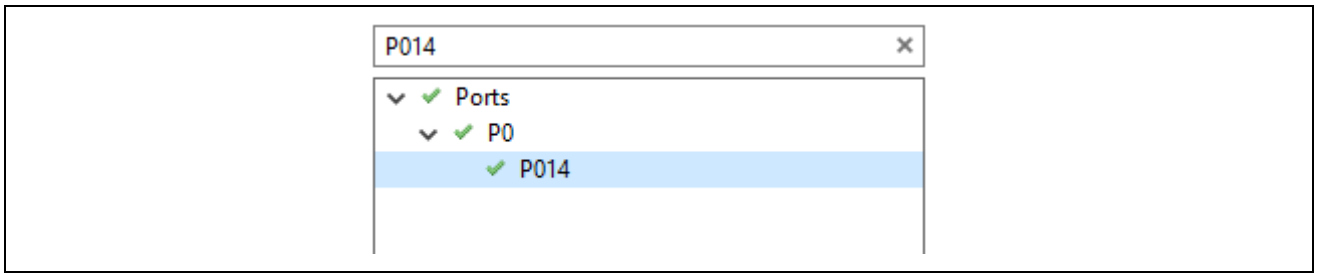


Figure 39. Start Command Prompt under the \MCUboot\scripts Folder

In the P014 Pin Configuration menu, change the "Mode" option to "Disabled". Afterwards, the pin mode will automatically change to "Analog Mode".

6.3 Modify code

Even after changing these settings, the project fails to compile, as shown below.

```
Extracting support files...
14:42:11 **** Incremental Build of configuration Debug for project dac_agt_periodic_signal_generator_fpb_ra4t1_ep ****
make -r --output-sync -j12 all
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_group_irq.c
Building file: ../src/hal_entry.c
../src/hal_entry.c:107:28: error: use of undeclared identifier 'DAC_MAX_CONVERSION_PERIOD'
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_guard.c
  107 |     dac_max_samp_rate = 1/(DAC_MAX_CONVERSION_PERIOD*pow(10,-6));
      |                            ^
../src/hal_entry.c:172:51: error: use of undeclared identifier 'DADR0'
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_io.c
  172 |         g_transfer0_cfg.p_info->p_dest = (void *) DADR0;
      |                                                   ^
2 errors generated.
```

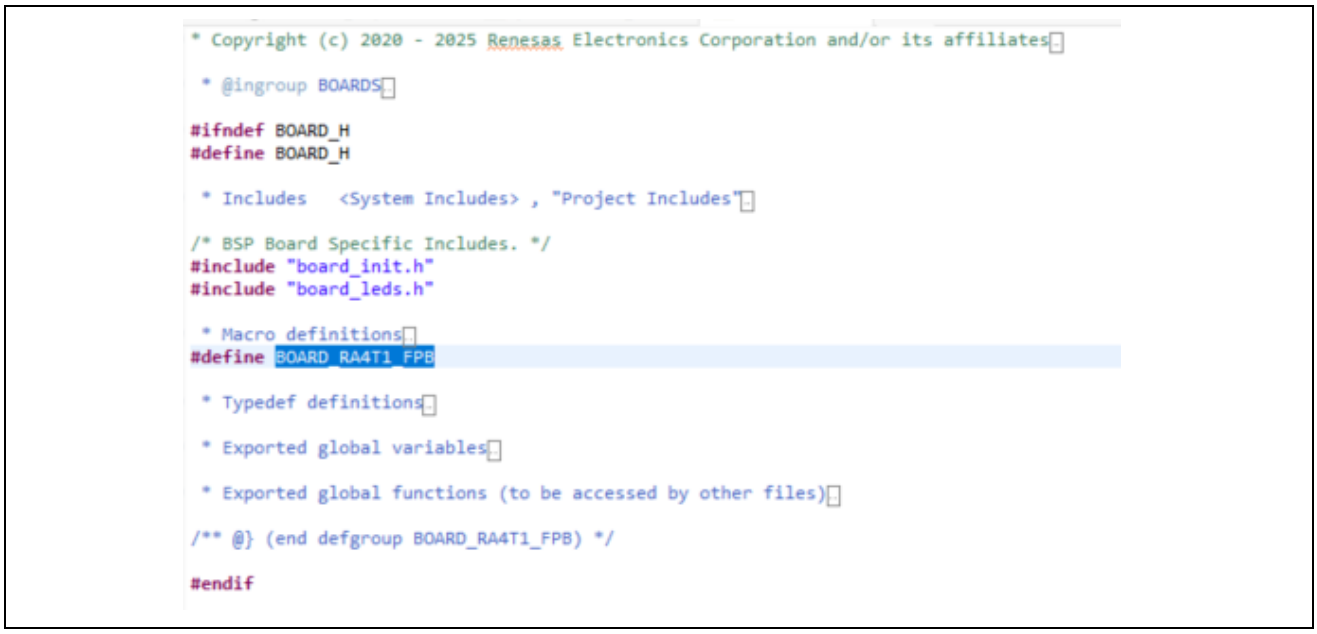
Figure 40. Compilation Errors Caused by Missing Symbols

This error happens because the identifiers "DADR0" and "DAC_MAX_CONVERSION_PERIOD" are not defined. In periodic_signal_generator_ep.h, these two identifiers are defined inside of multiple "#ifdef BOARD_*" blocks.

```
* Description  : Contains declarations of data structures and functions used in hal_entry.c.
* Copyright (c) 2020 - 2024 Renesas Electronics Corporation and/or its affiliates

#ifndef PERIODIC_SIGNAL_GENERATOR_EP_H_
#define PERIODIC_SIGNAL_GENERATOR_EP_H_

#ifndef M_PI
    #define M_PI 3.1415926535897932384626
#endif


#define EP_INFO       "\r\nThis example project demonstrates generation of a sinusoidal signal "\
                      "\r\nusing the DAC, AGT, and DTC modules. The output sinusoid's properties "\
                      "\r\nare set by inputting the desired samples per period and frequency in "\
                      "\r\nJLink RTT Viewer. The application first calculates and stores one period "\
                      "\r\nof the sinusoid to a look up table, and then iterates through the table  "\
                      "\r\nto generate the continuous wave from the DAC. The output wave will "\
                      "\r\noscillate between the minimum and maximum DAC values at the user-specified"\
                      "\r\nsamples per period and frequency.\n\n "


#define SPP_MIN_VAL   (2)
#define SPP_MAX_VAL   (256)

#define DAC_MID_VAL (2047.5)

#ifdef BOARD_RA2L1_EK
    #define DADR0                       0x4005E000
    #define DAC_MAX_CONVERSION_PERIOD   30
#endif

#ifdef BOARD_RA2A1_EK
    #define DADR0                       0x4005E000
    #define DAC_MAX_CONVERSION_PERIOD   1
#endif

#ifdef BOARD_RA4M2_EK
    #define DADR0                       0x40171000
    #define DAC_MAX_CONVERSION_PERIOD   4
#endif

#ifdef BOARD_RA6M4_EK
    #define DADR0                       0x40171000
    #define DAC_MAX_CONVERSION_PERIOD   4
#endif

#ifdef BOARD_RA4E2_EK
    #define DADR0                       0x40171000
    #define DAC_MAX_CONVERSION_PERIOD   4
#endif

#ifdef BOARD_RA4T1_MCK
    #define DADR0                       0x40171000
    #define DAC_MAX_CONVERSION_PERIOD   4
#endif

#endif /*PERIODIC_SIGNAL_GENERATOR_EP_H_ */
```

Figure 41. Contents of periodic_signal_generator_ep.h

These BOARD_* macros are defined in /ra/board/[board-specific-directory]/board.h. A screenshot of this file is shown below.



```
 * Copyright (c) 2020 - 2025 Renesas Electronics Corporation and/or its affiliates
 * @ingroup BOARDS
#ifndef BOARD_H
#define BOARD_H
 * Includes   <System Includes> , "Project Includes"
/* BSP Board Specific Includes. */
#include "board_init.h"
#include "board_leds.h"
 * Macro definitions
#define BOARD_RA4T1_FPB
 * Typedef definitions
 * Exported global variables
 * Exported global functions (to be accessed by other files)
/** @} (end defgroup BOARD_RA4T1_FPB) */
#endif
```

Figure 42. Contents of "board.h"

Since the FPB-RA4T1 and MCK-RA4T1 have identical MCUs, it is safe to add these lines of code to the end of periodic_signal_generator_ep.h, keeping the same constants from the BOARD_RA4T1_MCK block.

```
#ifdef BOARD_RA4T1_FPB
    #define DADR0                          0x40171000
    #define DAC_MAX_CONVERSION_PERIOD    4
#endif
```

DADR0 refers to the address of the two-byte D/A Data Register, as outlined in Section 33.2.1 of RA4T1 Group Hardware User Manual.

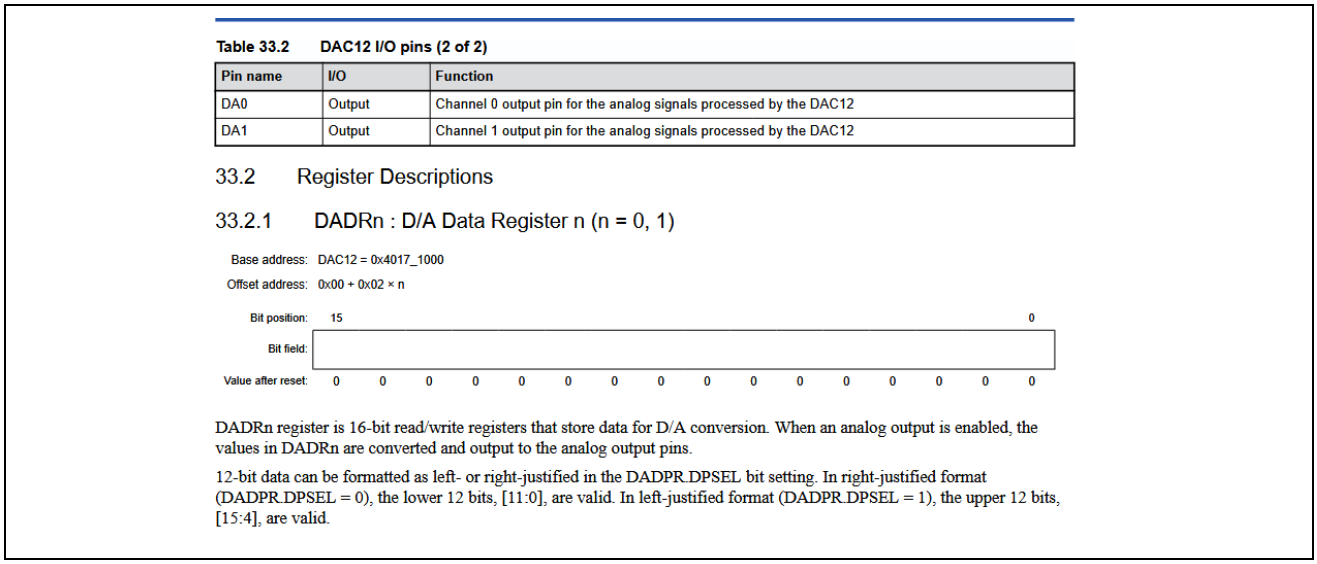


Table 33.2 DAC12 I/O pins (2 of 2)

Pin name	I/O	Function
DA0	Output	Channel 0 output pin for the analog signals processed by the DAC12
DA1	Output	Channel 1 output pin for the analog signals processed by the DAC12

33.2 Register Descriptions

33.2.1 DADRn : D/A Data Register n (n = 0, 1)

Base address: DAC12 = 0x4017_1000

Offset address: 0x00 + 0x02 × n

Bit position:	15															0
Bit field:																
Value after reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DADRn register is 16-bit read/write registers that store data for D/A conversion. When an analog output is enabled, the values in DADRn are converted and output to the analog output pins.

12-bit data can be formatted as left- or right-justified in the DADPR.DPSEL bit setting. In right-justified format (DADPR.DPSEL = 0), the lower 12 bits, [11:0], are valid. In left-justified format (DADPR.DPSEL = 1), the upper 12 bits, [15:4], are valid.

Figure 43. DADRn Data Register Information

DAC_MAX_CONVERSION_PERIOD can also be found in the RA4T1 user's manual. In section 41.5, the conversion time of the 12-bit DAC is stated to be 4 microseconds.

41.5 DAC12 Characteristics

Table 41.37 D/A conversion characteristics

Parameter	Min	Typ	Max	Unit	Test conditions
Resolution	—	—	12	Bits	—
Without output amplifier					
Absolute accuracy	—	—	±24	LSB	Resistive load 2 MΩ
INL	—	±2.0	±8.0	LSB	Resistive load 2 MΩ
DNL	—	±1.0	±2.0	LSB	—
Output impedance	—	8.5	—	kΩ	—
Conversion time	—	—	3	µs	Resistive load 2 MΩ, Capacitive load 20 pF
Output voltage range	0	—	VREFH	V	—
With output amplifier					
INL	—	±2.0	±4.0	LSB	—
DNL	—	±1.0	±2.0	LSB	—
Conversion time	—	—	4.0	µs	—
Resistive load	5	—	—	kΩ	—
Capacitive load	—	—	50	pF	—
Output voltage range	0.2	—	VREFH – 0.2	V	—

Figure 44. 12-Bit DAC Conversion Time

The values for both macros are left unchanged, but it is still important to double-check any registers or magic numbers that may change between hardware targets.

6.4 Results

An oscilloscope is needed to verify the functionality of this project. Before running this project, connect the positive terminal of the oscilloscope probe to pin 10 of the PMOD2 header, and connect the ground terminal to pin 11 of the PMOD2 header,

Pmod 2 Connector				FPB-RA4T1	Pmod 2 Configuration		External Clock Configuration	
PMOD2-6	VCC			+3.3 V				
PMOD2-7	GPIO			P205 (IRQ1-DS)				
PMOD2-8	GPIO / RESET			P407 (RESET)				
PMOD2-9	GPIO			P015				
PMOD2-10	GPIO			P014				
PMOD2-11	GND			GND				
PMOD2-12	VCC			+3.3 V				

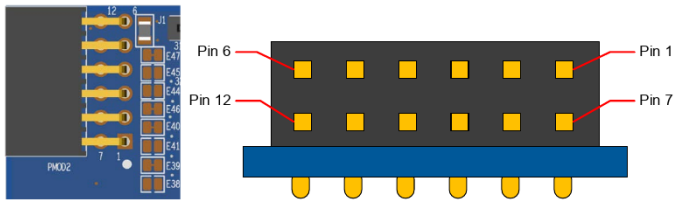


Figure 11. Pmod 2 Connector

Figure 45. PMOD 2 Connector Pinout

Debug the project as per the procedure described in section 5.4, and follow the instructions shown in the RTT Viewer console.

```
00>
00> ********************************************************************
00> *   Renesas FSP Example Project for dac_agt_periodic_signal_generator Module   *
00> *   Example Project Version 1.0                                    *
00> *   Flex Software Pack Version  5.9.0                              *
00> ********************************************************************
00> Refer to readme.txt file for more details on Example Project and
00> FSP User's Manual for more information about dac_agt_periodic_signal_generator driver
00>
00> This example project demonstrates generation of a sinusoidal signal
00> using the DAC, AGT, and DTC modules. The output sinusoid's properties
00> are set by inputting the desired samples per period and frequency in
00> JLink RTT Viewer. The application first calculates and stores one period
00> of the sinusoid to a look up table, and then iterates through the table
00> to generate the continuous wave from the DAC. The output wave will
00> oscillate between the minimum and maximum DAC values at the user-specified
00> samples per period and frequency.
00>
00>
00> Enter the desired samples per period for the output sine wave. Input Range: 2 to 256
  < 256
00> 256 samples per period selected.
00>
00> Enter desired frequency for output sine wave, in Hz. Maximum input frequency: 976 Hz
  < 500
00> 500 Hz selected.
00>
00> Outputting 500 Hz wave with 256 samples per period.
00>
00> Enter the desired samples per period for the output sine wave. Input Range: 2 to 256
```

Figure 46. Expected Results for Periodic Signal Generator EP

After entering the desired samples per period and frequency, a sine wave should appear on the oscilloscope's display.

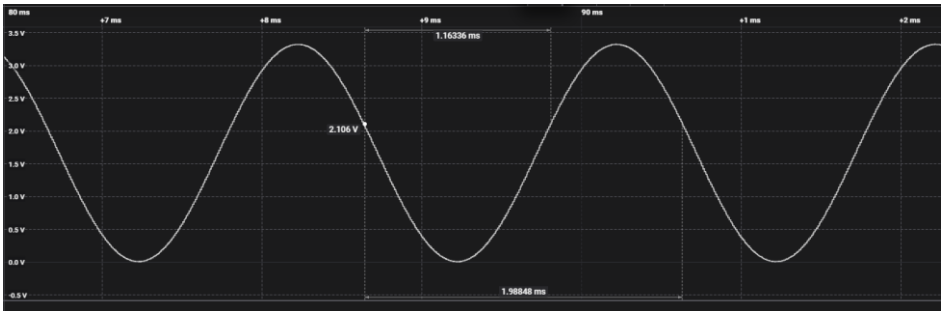


Figure 47. 12-Bit DAC Conversion Time

According to the output shown here, the period of the sine wave is 2ms, which matches with the 500hz frequency that was selected above.

7. Example: Porting SPI EP

The SPI example project demonstrates the capabilities of the SPI peripheral. It uses two SPI channels, one to transmit data and one to receive it. These channels are connected via jumper wires. The DOUT pin of channel 0 is connected to the DIN pin of channel 1, and the DOUT pin of channel 1 is connected to the DIN pin of channel 0. Since this EP uses a hardware-based implementation of SPI, and since the corresponding FSP API functions are non-blocking, it is possible to initiate a read and a write at the same time. After the SPI transactions are done, the EP verifies that the message was successfully received by checking whether the sent and received data is equal.

7.1 Overview of SPI Pin-Peripheral Map

First, perform the project creation and stack importing steps that were explained in sections 5 and 6.1. At this point, the primary concern should be assigning pins for the SPI peripheral channels.

This EP makes use of six pin assignments; it requires three pins for SPI channel 0 and another three for SPI channel 1. Going from one pin assignment (as described in section 6) to six leads to a higher potential for conflicts with target hardware.

While on the Pins menu of the FSP configuration, navigate to **Peripherals > Connectivity:SPI** and select SPI0.

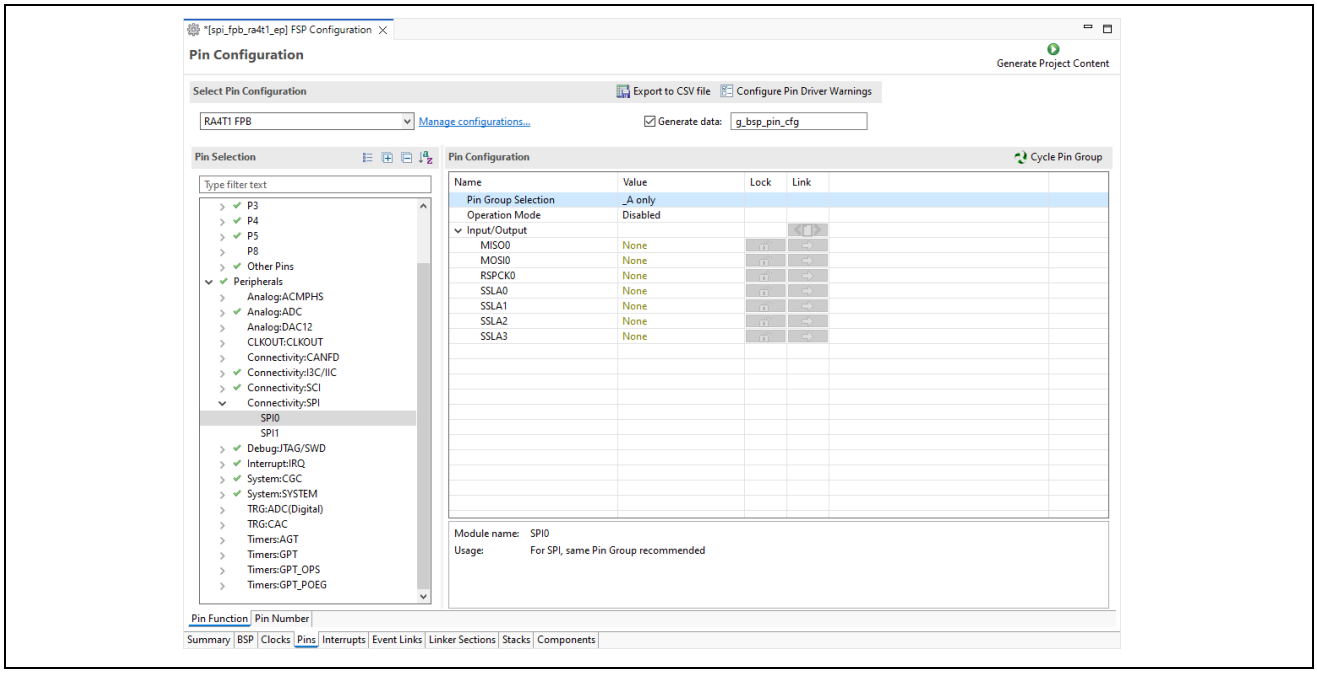


Figure 48. SPI Channel 0 Pin Configuration

Change "Operation Mode" from "Disabled" to "Enabled". This should lead to two pin assignment errors from P206 and P207.

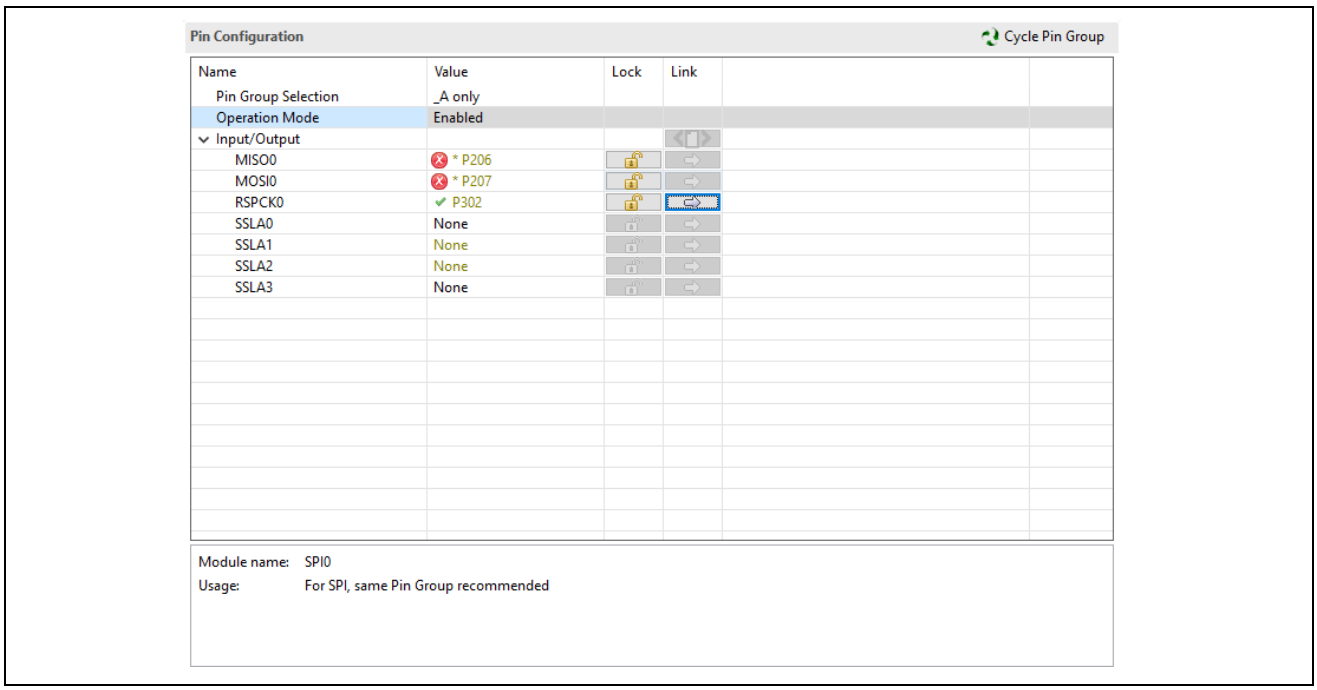


Figure 49. Enabling SPI Channel 0 in Pin Configuration

Although it is possible to resolve these errors, there are two problems with using pin group A here:

Pins 206 and 207 are connected to LED1 and LED2. This is the reason for the original error. As shown in the FPB-RA4T1 user's manual, it is necessary to cut jumpers E7 and E8 in order to disconnect these pins from

the LEDs. Even after soldering on a pin header, it's impossible to use these pins for both SPI and the onboard LEDs.

Table 9. FPB-RA4T1 Board LED Functions

Designator	Color	Function	MCU Control Port
LED1	Green	User LED	P207
LED2	Green	User LED	P206
POWER	Green	Power on indicator	+3.3 V
LED6	Yellow	Debug LED	Renesas RA4M2 Debug MCU

The User LEDs may be isolated from the main MCU so that the associated ports can be used for other purposes. To disconnect LED1 from P207, trace cut jumper E8 must be open. To disconnect LED2 from P206, trace cut jumper E7 must be open.

Figure 50. FPB-RA4T1 Board LED Functions

According to the FPB-RA4T1 user's manual (section 4.3.3 table 3), it is necessary to bridge jumper E38 and cut jumper E39 to connect P302 (SPI0 RSPCK0) to PMOD2-1.

E38	Pmod2 UART	Open	Connects P302 (CTS0) to Pmod 2 pin 1
E39	Pmod2 SPI	Closed	Connects P103 (SS0) to Pmod 2 pin 1

Figure 51. Description of Jumpers E38 and E39

Luckily, by changing the "Pin Group Selection" option for the SPI0 peripheral to "_B only", it is possible to reach a configuration that works without modifying the hardware.

Afterwards, you need to resolve each pin conflict:

P100, P101 - I3C conflicts: Change **Connectivity:I3C/IIC > I3C/IIC > Operation Mode** from Custom to Disabled.

P102 - SCI0 conflict: Change **Connectivity:SCI > SCI0 > Operation Mode** from "Simple SPI" to "Disabled".

P109, P110 - SCI9 conflicts: Change **Connectivity:SCI > SCI9 > Operation Mode** from "Simple I2C" to "Disabled".

P111 - GPIO conflict: Navigate to **Ports > P1 > P111** and change "Mode" from "Input Mode" to "Disabled".

At this point all pins (P100, P101, P102, P109, P110, P111) will be in peripheral mode, as they are configured for the SPI peripheral.

Another thing to keep in mind is to ensure that the pin configurations for the new pin group (P109, P110, P111) meet the system requirements. In the default pin configuration for the MCK-RA4T1, P206 and P207 are connected to the SPI peripheral; they have the internal pull-up resistor enabled to avoid floating voltages and drive capacity set to "M" to maintain proper signal integrity. Although these settings do not impact the functionality of the SPI example project, it may be necessary to change individual pin configurations based on the kit.

7.2 Results

In order to validate this EP, connect a set of jumper wires as follows:

MISO ----> P110 (J1-5) - P100 (J1-10)

MOSI ----> P109 (J1-4) - P101 (J1-9)

RSPCK ----> P111 (J1-6) - P102 (J1-10 or PMOD2-4)

An example of these connections on the FPB is shown below.

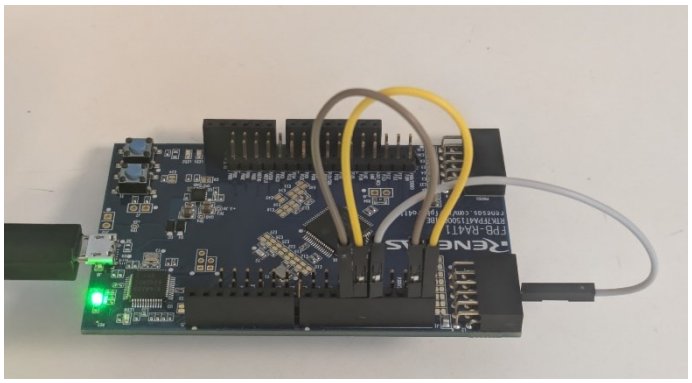


Figure 52. Picture of FPB-RA4T1 with Jumper Connections

After making these connections, compile and run the project while using RTT Viewer. An example of successful output is shown below.

```
00> SEGGER J-Link V8.12f - Real time terminal output
00> SEGGER J-Link (unknown) V1.0, SN=1083621257
00> Process: e2-server-gdb.exe
00>
00> ********************************************************************
00> *   Renesas FSP Example Project for r_spi Module                    *
00> *   Example Project Version 1.0                                     *
00> *   Flex Software Pack Version  5.9.0                               *
00> ********************************************************************
00> Refer to readme.txt file for more details on Example Project and
00> FSP User's Manual for more information about r_spi driver
00>
00> The project initializes SPI driver and configures SPI channels in Master and Slave mode.
00> After initialization, master and slave can transmit and receive data based on the commands from user.
00> Refer to the MCU User Manual for valid bit rates and corresponding clock settings.
00>
00>  ** SPI INIT SUCCESSFUL **
00>
00> Select from the below Menu options
00>
00> Press 1 for Write() and Read()
00> Press 2 for WriteRead()
00> Press 3 to Exit
  < 1
00>
00> Enter text input for Master buffer. Data size should not exceed 64 bytes.
  < Renesas RA4T1
00>
00> Master transmitted user input data to Slave
00>
00> Slave transmitted the data back to Master
00>
00> Master received data: Renesas RA4T1
00>
00>
00> ** SPI WRITE AND READ Demo Successful**
00>
00> Enter any other key to go back to the main menu
```

Figure 53. Expected Results for SPI EP

8. Other Considerations

8.1 FSP Version Incompatibilities

Different versions of the Renesas RA FSP may introduce incompatibilities that remove API methods or change the behavior of projects. Major releases of the Renesas RA FSP are more likely to introduce breaking changes, but minor versions may still include changes that could impact your project. Examples of things that could change between FSP versions are as follows:

- FSP module APIs
- FSP module stack configurations
- Third-party libraries / middleware
- Deprecation status of FSP modules
- Linker scripts
- Build configurations

To minimize errors and simplify the debugging process, it is recommended that FSP versions are kept the same between the two kits/boards. Users should only upgrade to a new FSP version after verifying the functionality of their ported project. It is recommended for users to review FSP release notes and identify potential regressions before upgrading.

8.2 MCU Incompatibilities

The FPB-RA4T1 and MCK-RA4T1 both use RA4T1 group MCUs with identical part numbers (R7FA4T1BB3CFM). This means that register layout, flash memory size, number of pins and IO ports, and number of channels for peripherals like the 12-bit DAC are the same. When porting a project to a target with a different MCU or pin package, code adjustments may need to be made. For example, when moving to a lower pin count version of an MCU, users may need to change peripherals to a different channel number. See sections 1.3 to 1.7 in the RA4T1 Group User's Manual for a full list of differences between different part numbers within the RA4T1 group.

8.3 Compiler Incompatibilities

FSP and BSP functions are verified to be operational on both GCC and LLVM. EPs typically do not use compiler-specific functionality, but moving a project from one compiler/compiler version to another may still introduce performance differences. Additionally, linker scripts differ between compilers, potentially leading to differences in the memory layout of a compiled binary.
FSP v6.0.0 introduces substantial changes to auto-generated linker scripts compared to earlier versions. It is recommended to review the FSP v6.0.0 release notes to understand these updates. However, other versions may also contain modifications, so users should review the release notes for each version.

8.4 8.4 RTOS Stack Porting

As of the publication date of this application note, e² studio does not have a user-friendly way to import RTOS threads, objects, or configurations into another project. After creating an RTOS project with a matching FSP version and performing general configurations, there is a general flow for importing RTOS-specific configurations:

- Import the HAL stacks. This can be done through the "Import stacks" option.
- Generate all the threads.
- Import all stacks for each thread.
- Double check that all properties are configured correctly between the two projects. Look at the BSP and thread properties, IRQ configurations, pin configurations, and RTOS object configurations.

These steps can be accomplished in e² studio, but some users may find it easier to use a diffing tool to move properties between the two projects' configuration.xml files.

8.5 Porting Complex Projects

Sometimes you may need to port a project with many stacks and properties, some of which may hard to move between projects. If you are in this situation, it may be helpful to modify the old project's configuration.xml file and move it to the new project before proceeding with conventional steps. This method involves the following steps:

- Create a new RTOS project for the target board.
- Make a duplicate of the source project's configuration.xml file and give it a different name (e.g. configuration_source.xml). Move this duplicate to the target project for convenience.
- Look at configuration.xml and configuration_source.xml in the target project and take a note of the places where the board name is specified. It is usually written in multiple forms.
- Using this information, change the board names in the configuration_source.xml to match the target kit. When editing configuration files in a text editor, make sure that they are not also open in e² studio.
- Copy configuration.xml to another location for backup, then rename configuration_source.xml to configuration.xml.
- Double-check the clock and pin configurations. Refer to sections 4.3.1 and 6.2-6.4 for more information about possible adjustments.

9. References

1. RA FSP Documentation https://renesas.github.io/fsp/
2. FSP Architecture https://renesas.github.io/fsp/_f_s_p__a_r_c_h_i_t_e_c_t_u_r_e.html
3. BSP API Reference https://renesas.github.io/fsp/group___b_s_p___m_c_u.html
4. Information on CMSIS packs https://developer.arm.com/documentation/109350/v6/CMSIS/Basic-concepts/CMSIS-Pack
5. RA FSP example projects repository https://github.com/renesas/ra-fsp-examples
6. RASC User Guide for MDK and IAR https://renesas.github.io/fsp/_s_t_a_r_t__d_e_v.html#RASC-MDK-IAR-user-guide
7. RA4T1 Group Resources https://www.renesas.com/en/products/ra4t1
8. FPB-RA4T1 Resources https://www.renesas.com/en/design-resources/boards-kits/fpb-ra4t1
9. MCK-RA4T1 Resources https://www.renesas.com/en/design-resources/boards-kits/mck-ra4t1

10. Next steps

Users are encouraged to engage in the following activities after reviewing this application note:

1. Apply the same porting flow to other RA evaluation boards. The introduction section of this application note contains a list of porting candidates.
2. Transition from evaluation boards to custom PCB designs, using the porting techniques described here as a guide.
3. Explore additional example projects involving more advanced concepts like motor control, security, or TrustZone.
4. Optimize example projects for specific considerations like power consumption and system performance.
5. Leverage other Renesas resources, including the RA community forums and GitHub repositories, for further examples and support.
    1. Renesas Engineering Community - https://community.renesas.com/
    2. Renesas Support - https://www.renesas.com/myrenesas

11. Website and References

Visit the following URLs to learn about the RA family of microcontrollers, download tools and documentation, and get support.

RA Product Information	renesas.com/ra
Flexible Software Package (FSP)	renesas.com/ra/fsp
RA Product Support Forum	renesas.com/ra/forum
Renesas Support	renesas.com/support

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Feb.23.26	-	First release document

System Release Package, RA Series – User Manual

Publication Date: Feb.23.26

Published by: Renesas Electronics Corporation

RA Family

Renesas Electronics Corporation

R01AN8249EU0100