QC-BEKITPOC2Z

QuickConnect Beginners Kit V2.0

Introduction

This manual provides step-by-step guidance using the QuickConnect Beginners Kit V2.0 for three projects: Blinky, Smart Temperature Data Logger, and BLE-based Asset Tracker. Each project includes detailed instructions for hardware setup, software configuration, and result analysis. The document also suggests possible customizations to help users modify and expand the project outcomes.

All applications are developed using the QuickConnect Studio alongside the Beginners Kit V2.0 hardware enabling rapid and flexible prototype development. To ensure proper setup, follow the Prerequisites section.

These labs offer hands-on experience that guides users through both hardware assembly and software development for embedded system prototyping. The QuickConnect platform streamlines the design process allowing users to move from concept to prototype in only minutes. Code generation, compilation, customization, and debugging are all seamlessly managed through a single, unified interface.

Note: To ensure the projects are set up correctly, complete the steps in the order listed in this document.

Contents

1. Kit Information

1.1 Kit Contents

Orderable Part Number – QC-BEKITPOC2Z

Hardware Components – BGK-RA6E2 (R7FA6E2BB3CFM) MCU board

- PMOD Board with Ultra-Low Power Wi-Fi + Bluetooth® Low Energy Combo Module, DA16600MOD
- PMOD Board with Low Power Bluetooth, DA14531MOD
- PMOD Board with Relative Humidity and Temperature Sensor (HS4001)

Note: While the sensor used on the board (HS4001) is marked as NRND (Not Recommended for New Designs), users can still run the examples in this document and fully experience QuickConnect Studio without any issues.

2. Hardware Description

2.1 BGK-RA6E2

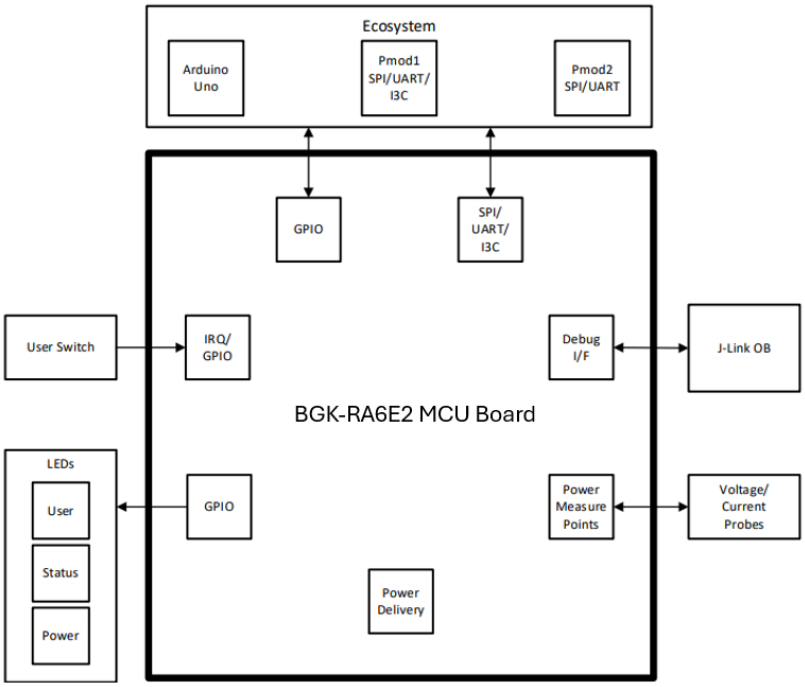
The BGK-RA6E2 Board provides an entry point for evaluation, prototyping, and development with the RA6E2 MCU. Also, because this board incorporates an emulator circuit, it can be used for designing user applications without the requirement of making further investments in tools. This product includes through-holes for pin headers that allow access to all MCU signal pins, which allows easy prototyping with a breadboard.

Table 1. Board Specifications

Item	Specification
Evaluation MCU	Part No: R7FA6E2BB3CFM; package: 64-pin LQFP
	On-chip memory: 256KB code flash, 40KB SRAM, 4KB data flash memory
Board size	Size: 53mm × 85mm; thickness: 1.6mm
Power-supply voltage	Board supply: 5V. VCC: 3.3V. MCU operation voltage range 2.7V to 3.6V
Power-supply circuit	USB connector: VBUS (5V input); VBUS is converted to 3.3V by LDO
	2-pin external power-supply header*1
Push switch	Reset switch x 1, user switch x 1
LED	Power indicator: green x 1, user: green x 2, On-board debugger ACT: yellow x 1
USB connector	Connector: micro-USB type B
PMOD™ connector	Connector: angle type, 12-pin x 2
Arduino™ connector	Connector: 6-pin x 1, 8-pin x 2, 10-pin x 1 The interface is compatible with Arduino™ Uno R3
MCU header[1]	Header: 32 pins x 2
Emulator	J-Link on-board programmer / debugger

1. This part is not mounted.

2.1.1 Block Diagram



2.1.2 Board Specifications

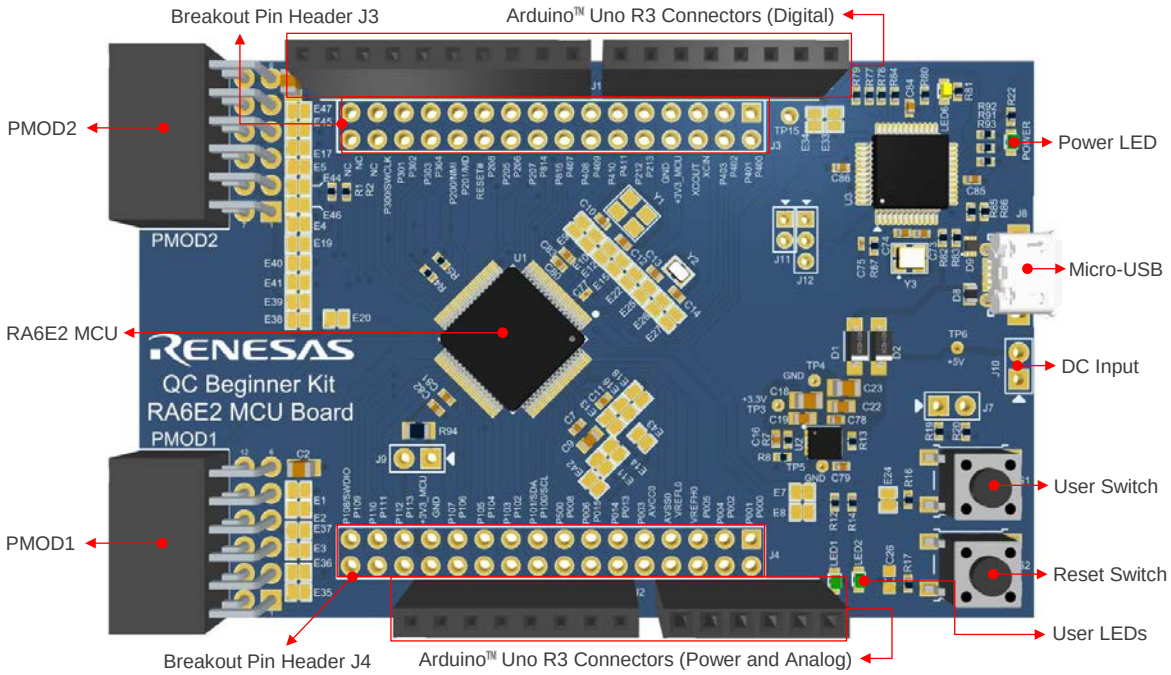


Figure 1. BGK-RA6E2 MCU Board Layout

2.1.2.1 Arduino Interface

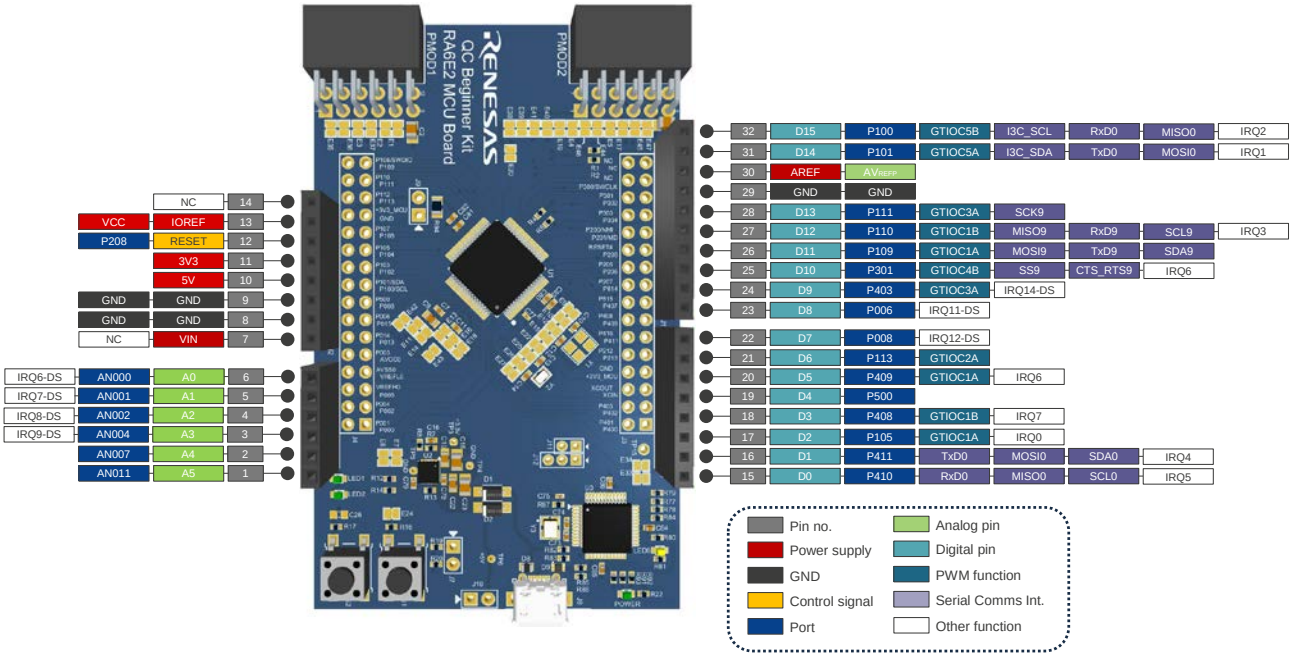
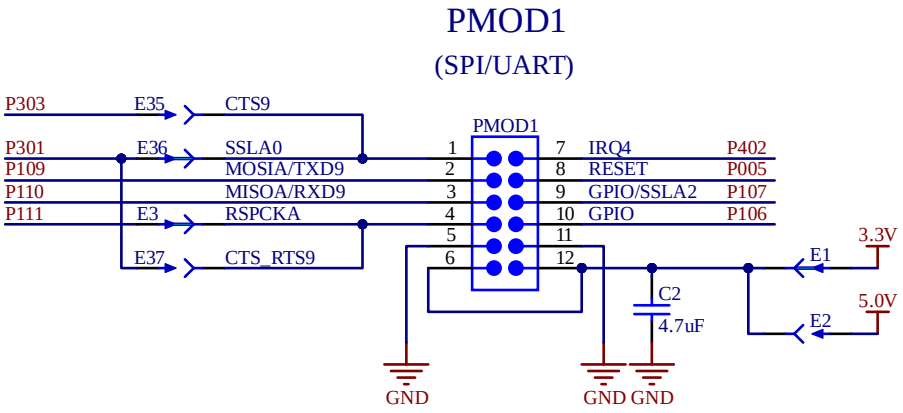


Figure 2. BGK-RA6E2 MCU Board Arduino Interface

2.1.2.2 PMOD Interface

The QC Beginner Kit RA6E2 MCU Board has two PMOD connectors. PMOD 1 supports the Type 2A interfaces and can also be configured for Type 3A. PMOD 2 supports Type 6A interfaces and can also be configured for Type 2A, Type 3A, and Type 7A.

The default setting for PMOD 1 is SPI/UART interfaces.

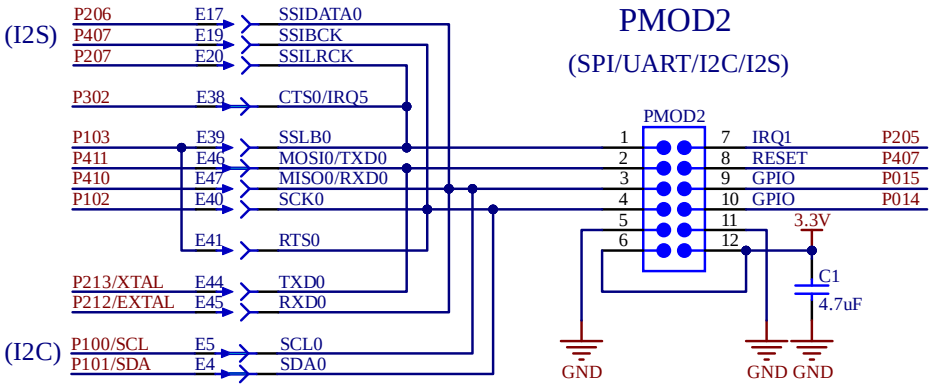


PMOD1 Mode Selection				
Mode	E3	E35	E36	E37
SPI/UART (default)	CLOSED	OPEN	CLOSED	OPEN
UART with flow control	OPEN	CLOSED	OPEN	CLOSED

(Configure the indicated jumpers as shown in the table)

Figure 3. PMOD 1 Interface

The default setting for PMOD 2 is I2C interfaces.



PMOD2 Mode Selection

Mode	E4	E5	E17	E19	E20	E38	E39	E46	E47	E40	E41	E44	E45
I2C (default)	CLOSED	CLOSED	OPEN	OPEN	OPEN	CLOSED	OPEN	CLOSED	OPEN	OPEN	OPEN	OPEN	OPEN
I2S	OPEN	OPEN	CLOSED	CLOSED	CLOSED	OPEN	OPEN	Unused	OPEN	OPEN	OPEN	Unused	OPEN
SPI/UART	OPEN	OPEN	OPEN	OPEN	OPEN	OPEN	CLOSED	CLOSED	CLOSED	CLOSED	OPEN	OPEN	OPEN
(P411 TXD0, P410 RXD0) UART with flow control	OPEN	OPEN	OPEN	OPEN	OPEN	CLOSED	OPEN	CLOSED	CLOSED	OPEN	CLOSED	OPEN	OPEN
(P213 TXD0, P212 RXD0) UART with flow control	OPEN	OPEN	OPEN	OPEN	OPEN	CLOSED	OPEN	OPEN	OPEN	OPEN	CLOSED	CLOSED	CLOSED

(Configure the indicated jumpers as shown in the table)
NOTE1: P213/P212 option can not be selected at the same time as the 24MHz crystal

Figure 4. PMOD 2 Interface

2.1.2.3 Copper Jumpers

Two types of copper jumpers are provided on the RA6E2 MCU board, designated trace-cut and solder-bridge with E.

A trace-cut jumper is provided with a narrow copper trace connecting its pads. The silkscreen overlay printing around a trace-cut jumper is a solid box. To isolate the pads, cut the trace between pads adjacent to each pad, and remove the connecting copper foil either mechanically or with the assistance of heat. When the etched copper trace is removed, the trace-cut jumper is turned into a solder-bridge jumper for any later changes.

A solder-bridge jumper is provided with two isolated pads can be joined together by either of the three methods:

- Solder can be applied to both pads to develop a bulge on each and the bulges joined by touching a soldering iron across the two pads.
- A small wire can be placed across the two pads and soldered in place.
- A SMD resistor, size 0805, 0603, or 0402, can be placed across the two pads and soldered in place. A 0Ω resistor shorts the pads together.

For any copper jumper, the connection is considered closed if there is an electrical connection between the pads (default for trace-cut jumpers.) The connection is considered open if there is no electrical connection between the pads (default for the solder-bridge jumpers).

Solder Bridge  Trace-cut Jumper

Figure 5. Copper Jumpers

2.2 DA16600 PMOD

The US159-DA16600EVZ is a PMOD board for the DA16600MOD. It is a Wi-Fi BLE module (802.11 b/g/n and Bluetooth V5.1). The DA16600 addresses the requirement of battery-use devices that require minimal power consumption and reliable operation. It can be used with a MCU board that supports Type 3A PMOD. In the following projects, this module is used for Wi-Fi communication.

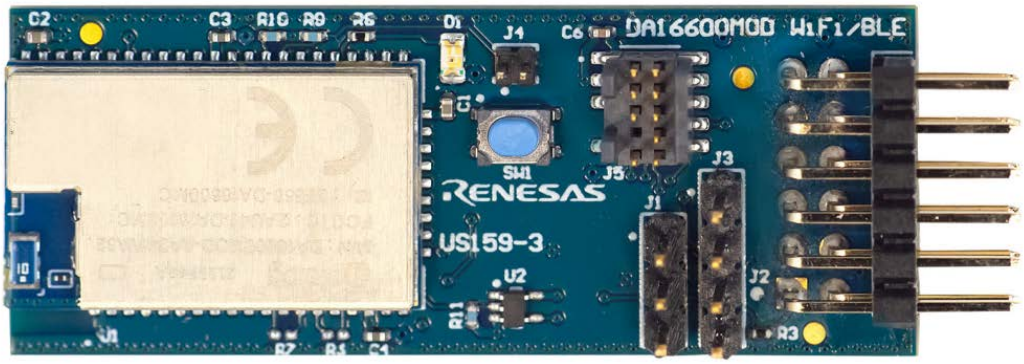


Figure 6. US159-DA16600EVZ PMOD Board

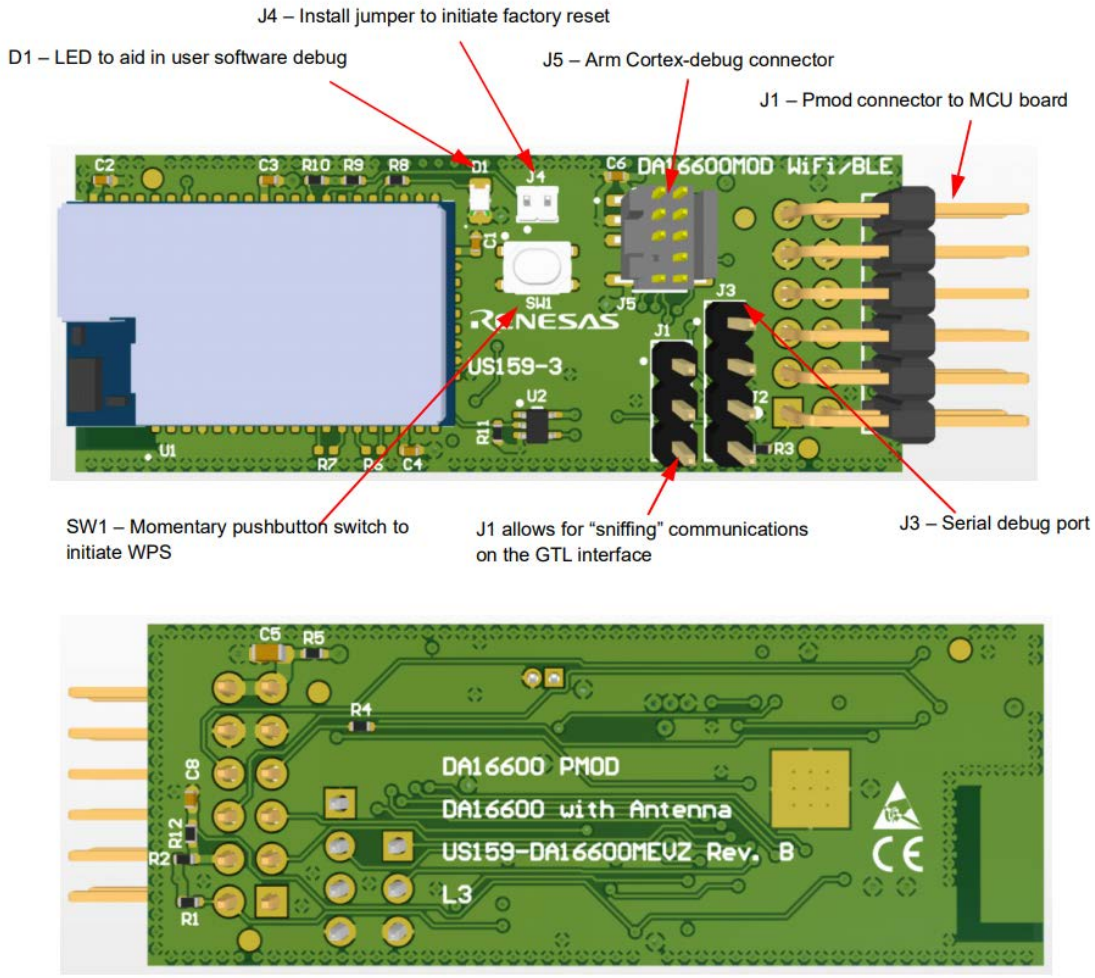


Figure 7. Evaluation Kit Details

2.3 DA14531 PMOD

The US159-DA14531EVZ is a PMOD board with DA14531MOD. It is a low power Bluetooth 5.1 SOC module. In this document, this module is used for BLE based projects.

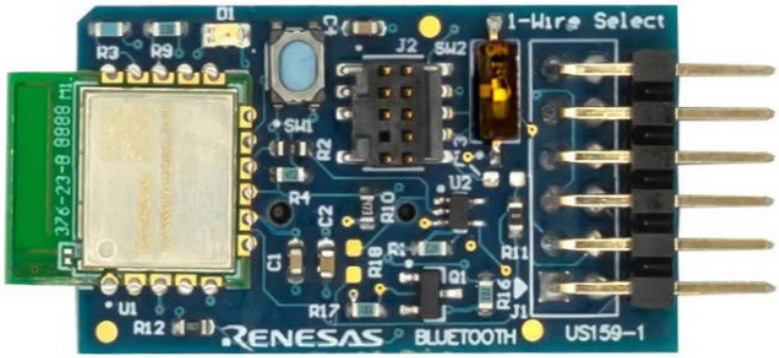


Figure 8. DA14531 Evaluation Board (Top)

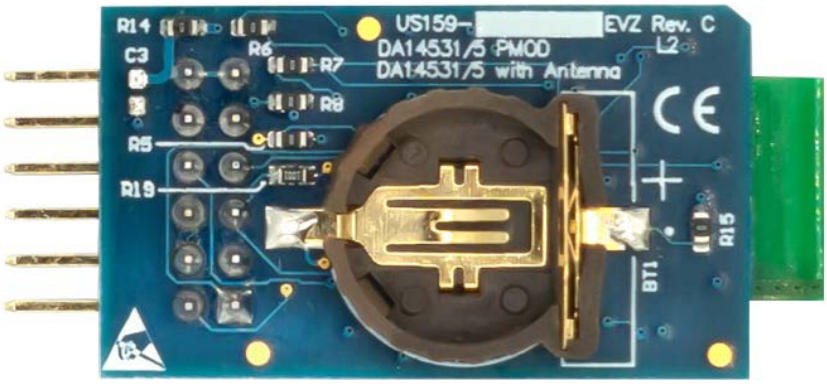


Figure 9. DA14531 Evaluation Board (Bottom)

2.4 Humidity and Temperature Sensor PMOD

The humidity and temperature PMOD board with a digital I^2C humidity and temperature sensor is used in the following labs.

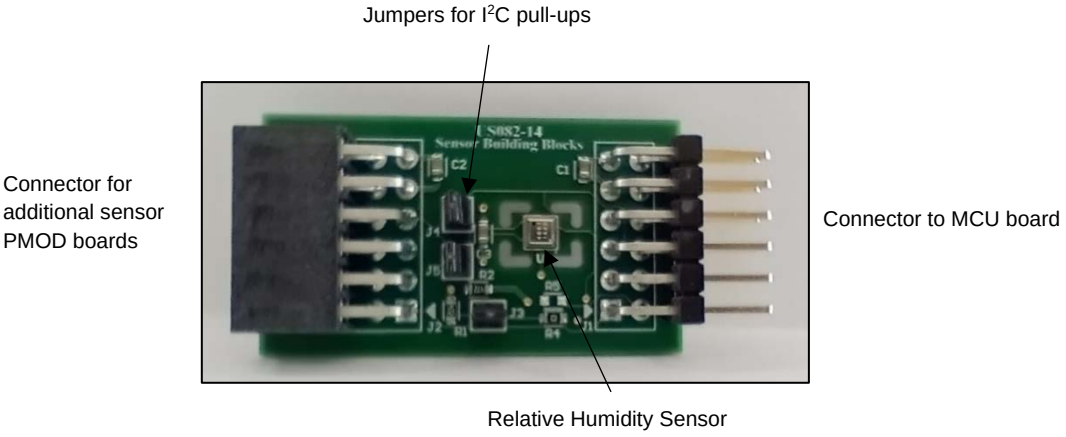


Figure 10. Humidity and Temperature Sensor Board Layout

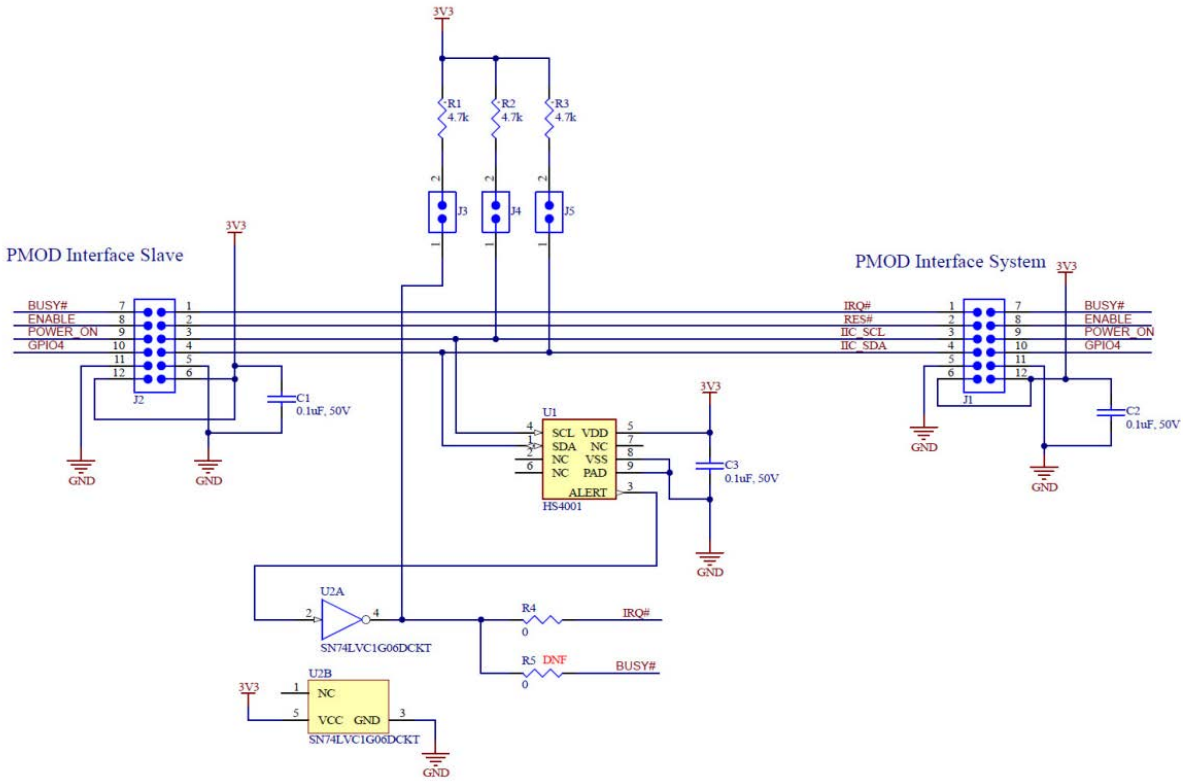


Figure 11. Sensor PMOD Application Schematic

3. QuickConnect Studio

The QuickConnect Studio (QCStudio) is an online, cloud-based embedded system design platform that enables users to graphically drag-and-drop eval boards to build custom prototypes. Users can generate, compile, customize, and debug the base system code on a single browser window without installing multiple applications. This reduces the complexity to build and test embedded system prototypes and increases the time to market.

3.1 Key Features

The following list highlights key features supported on QuickConnect platform.

- QuickConnect supports a broad portfolio of Renesas and Partner boards and devices
- Real-time code customization
- QuickConnect debugging by connecting to remote board farms deployed globally
- Multi-region deployment to increase connection speed from anywhere in the world
- Support for multiple concurrent users globally
- Real-time monitoring for cyber security threats

For more details, refer to the QuickConnect Studio landing page.

4. Prerequisites

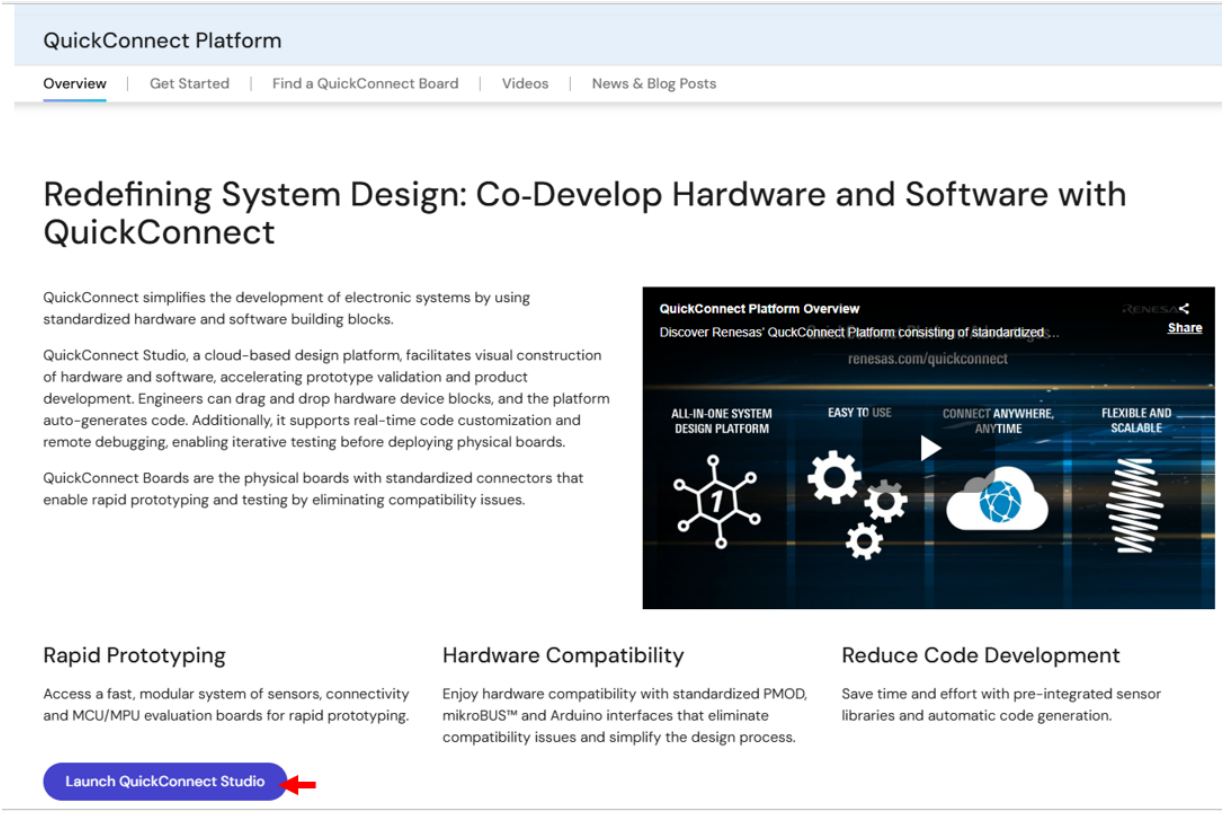
Before you begin this lab, ensure you have the following items:

- QuickConnect Beginners Kit V2.0 – The kit includes all the hardware components needed for this lab.
- Windows PC – A computer running Windows operating system. Renesas recommends using a system running Windows 10 or later versions, equipped with at least one available USB port.
- Web Browser – Ensure a web browser is installed on a Windows PC. For optimal performance, it is advised to use Google Chrome.
- MyRenesas Registration – An active MyRenesas account must be obtained to access QuickConnect. If you do not have one, register on the MyRenesas page.
- Internet Connection: A stable internet connection is required to access online resources and download necessary files.
- Segger J-link flasher – SEGGER - The Embedded Experts - Downloads - J-Link / J-Trace. To flash the boards, Renesas recommends downloading and installing the 64-bit installer, ensuring it is the latest version compatible with the user's device.
- Install latest J-Link Configurator V8.66 for direct debugging.
- QuickConnect Sandbox application – For applications based on Bluetooth connectivity, download and install the QuickConnect Mobile Sandbox application for Android or iOS. It can support Android versions up to 14 and iOS 10.0 or later versions.
- Wi-Fi security requirements – The DA16xxx series Wi-Fi modules support the following WiFi security types: 0 (OPEN), 1 (WEP), 2 (WPA), 3 (WPA2), 4 (WPA+WPA2), 5 (WPA3 OWE), 6 (WPA3 SAE), 7 (WPA2 RSN and WPA3 SAE). The "Sensor Data to AWS IoT" application used in this lab uses WPA2 security type by default. Verify that the security setting on the Wi-Fi access point(AP)/router matches this requirement. If not, update the settings accordingly either in the application or on the Wi-Fi AP/router.
- RTT Viewer – Download and install the RTT Viewer application (version 8.24) which is used to view the debug logs from the Renesas365 Dev Kit. Download link: About the RTT Viewer
- MQTT Explorer – Download and install the MQTT Explorer application, which is used to view the MQTT messages published by the Renesas365 Dev Kit to AWS IoT. Download link: MQTT Explorer | An all-round MQTT client that provides a structured topic overview

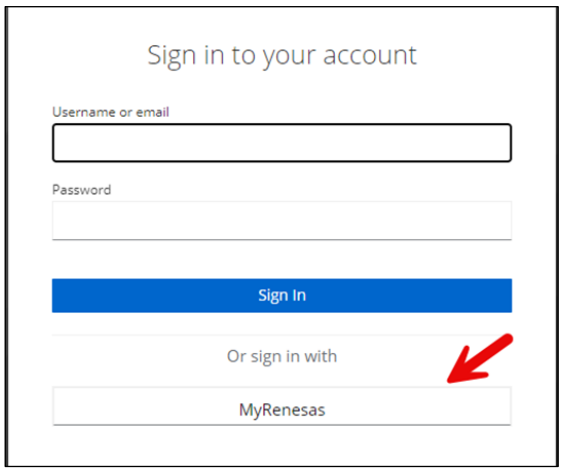
5. Getting Started

5.1 Launch QuickConnect Studio Workspace

1. Launch the QuickConnect Studio platform in a PC browser window.
   a. To launch a QuickConnect Studio user workspace, visit the QuickConnect Studio.
   b. Click on the **Launch QuickConnect Studio** button to launch a unique workspace in a browser window.

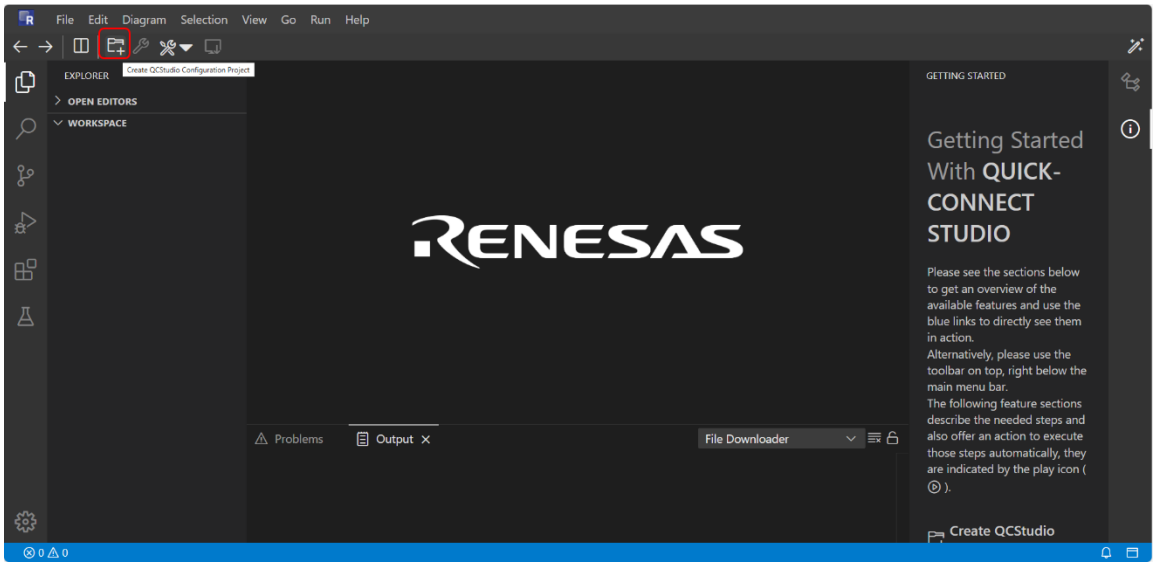


2. At the following screen, click on the **MyRenesas** button to log in using MyRenesas login credentials.

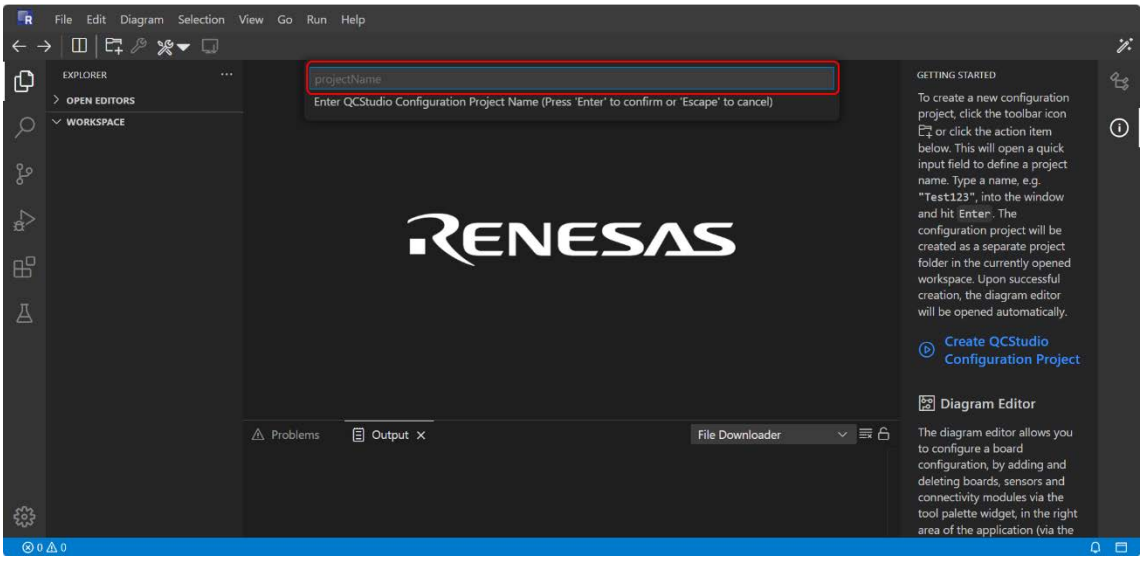


5.2 Create a New Project

1. Click on **Create QuickConnect Studio Configuration Project**.



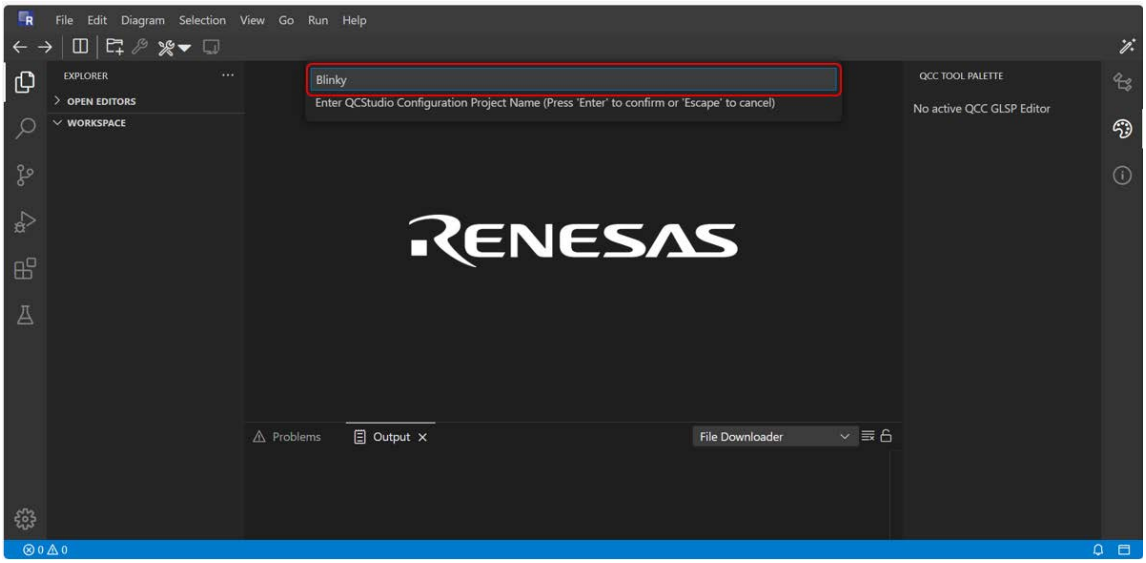
2. Enter the project name in the field that appears and press the **Enter** key.



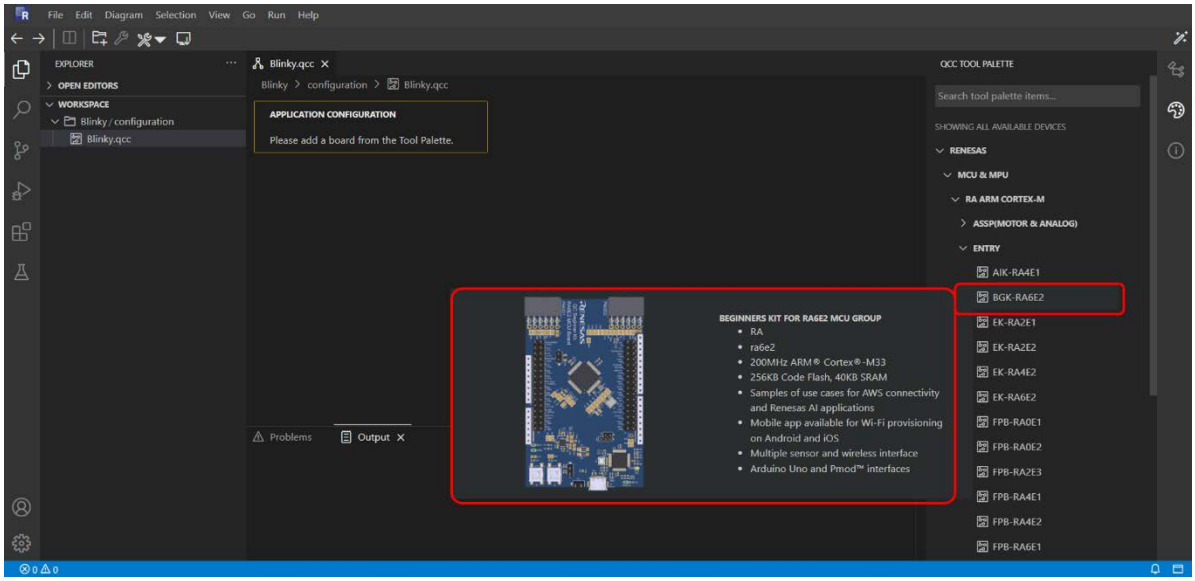
6. Project 1 – Blinky

6.1 Create the Project

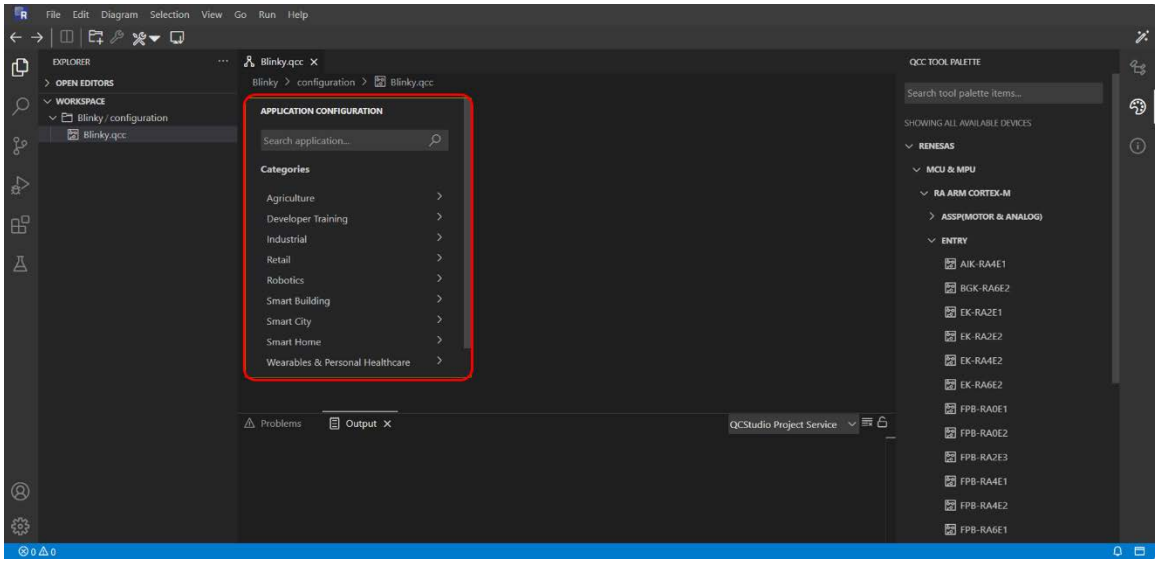
1. Follow the steps in Section 5 to launch and create a new project.



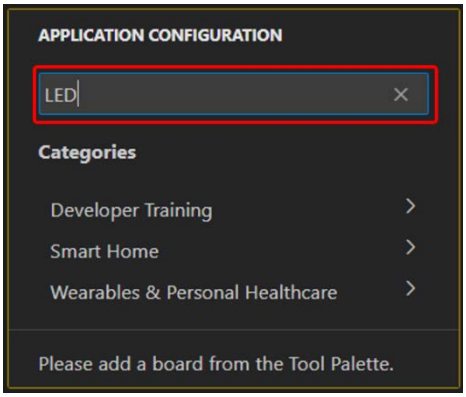
2. Drag and drop the MCU board from the QCC tool palette. Hover the cursor over the MCU to view its features. In this example, BGK-RA6E2 is used.



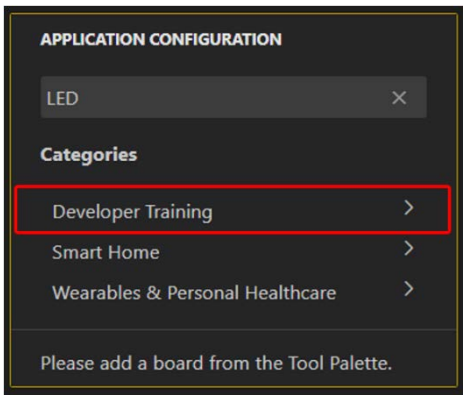
3. Select the MCU and choose **Application Configuration**.

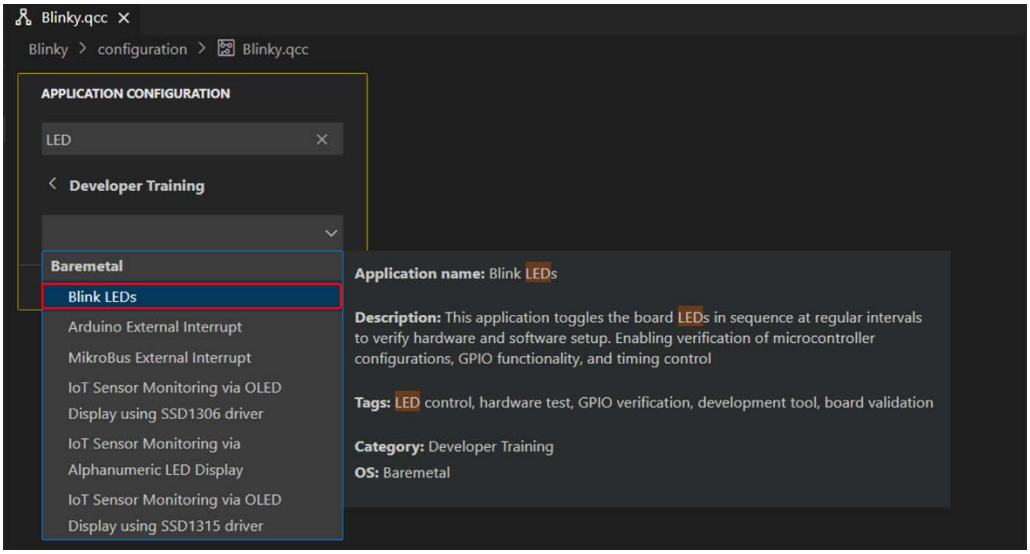


4. In the application configuration search bar, type ‘LED’ to automatically narrow your filtering options.

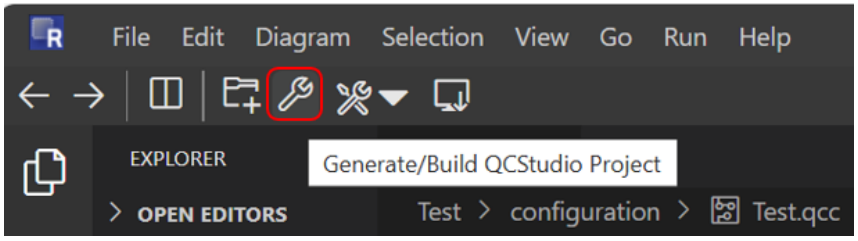


5. Next, in the **Application Configuration** window, select **Developer Training**. Click on **v** (the downward arrow). Select **Blink LEDs**.

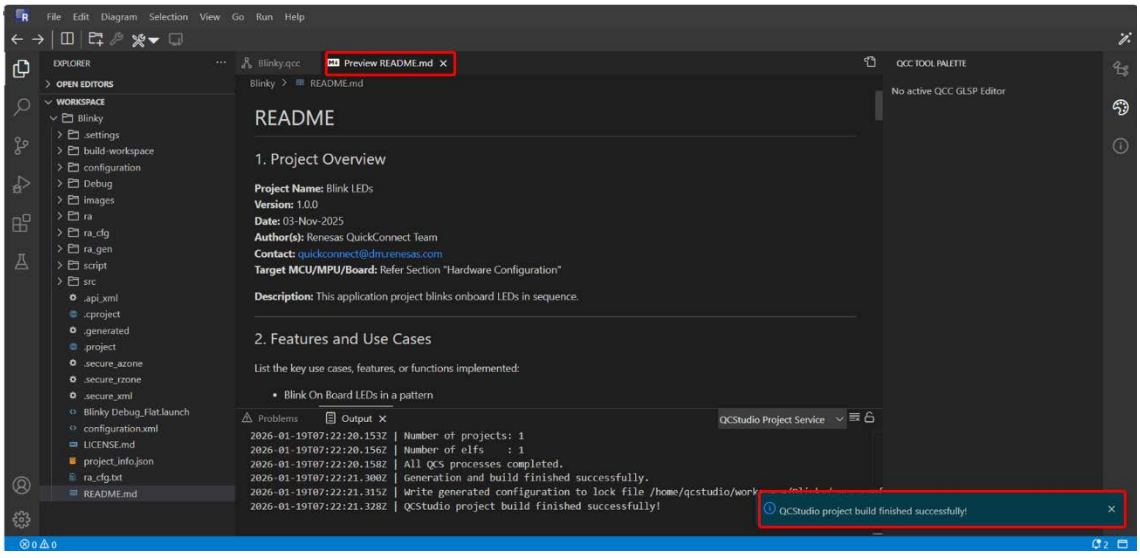




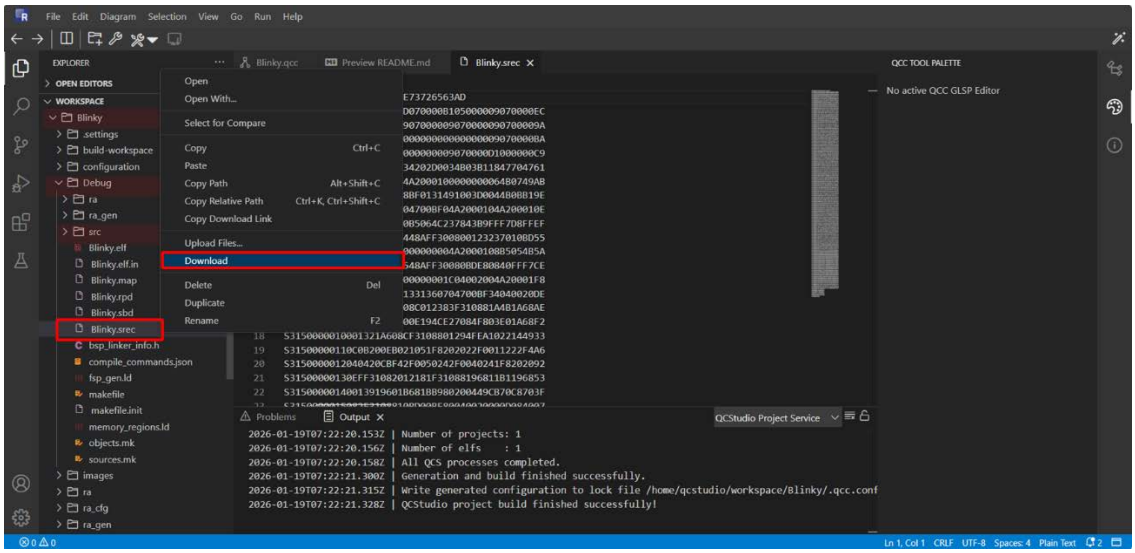
6. Click on **Build Project**. The build progress is displayed in the output log.



7. After the project is built, a notification appears stating “QCStudio project build finished successfully”. The **Readme.md** file opens for further instructions.

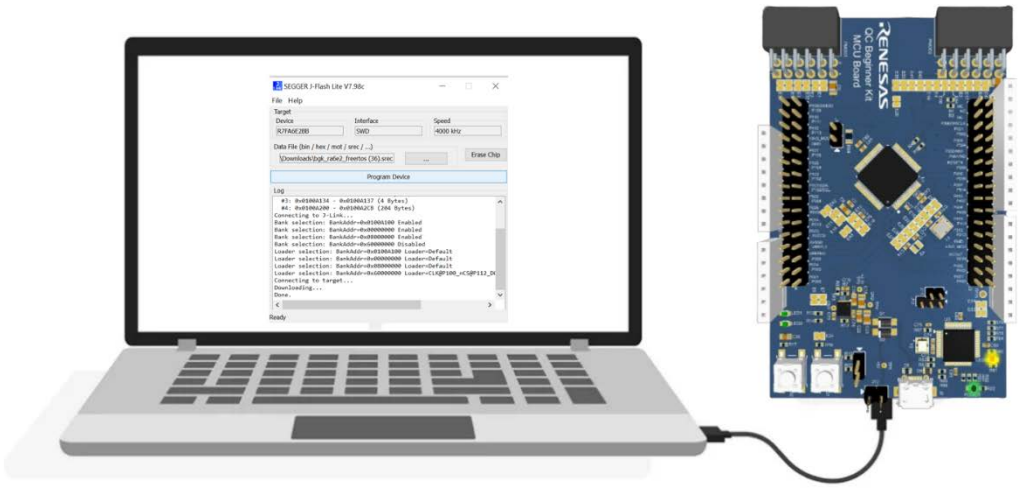


8. Go to Explorer, expand the project, expand **Debug**, select the **.srec** file, right-click, and select **Download**.

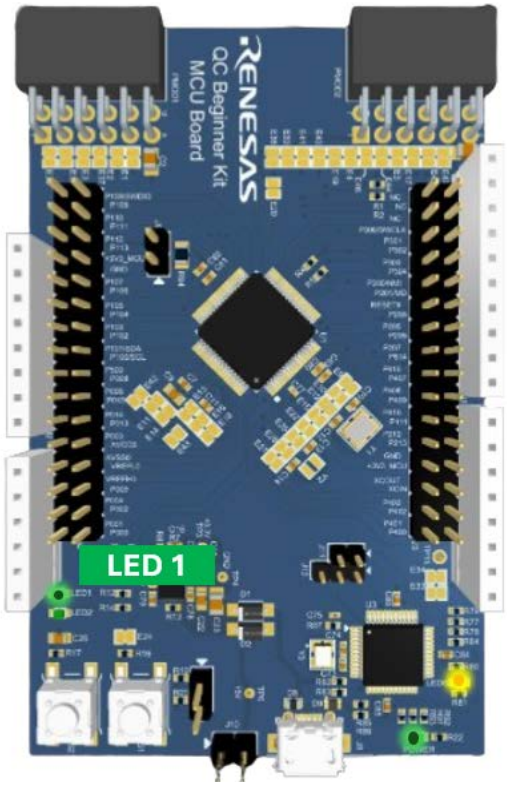


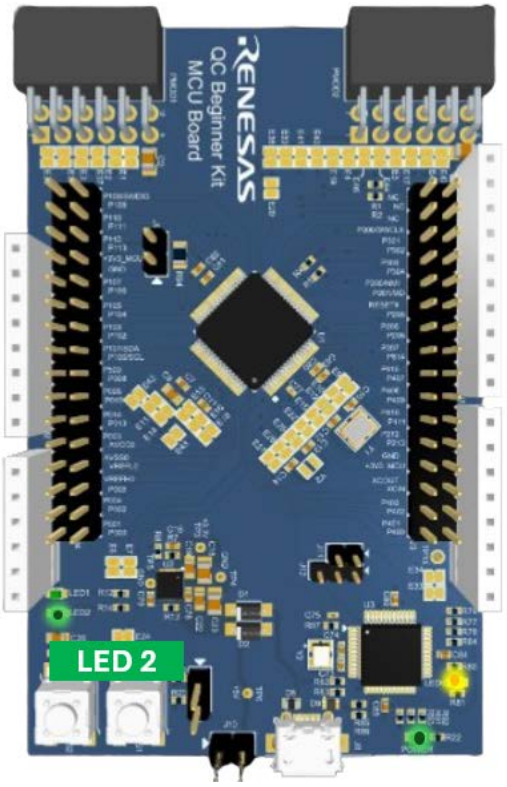
6.2 Programming Hardware and Viewing Results

1. Connect the BGK-RA6E2 to a laptop using a USB cable. Refer to the section, Flashing Code to the Hardware using SEGGER J-Flash Lite, to flash the code.



2. Observe the output - LED 1 and 2 blinking in a pattern.

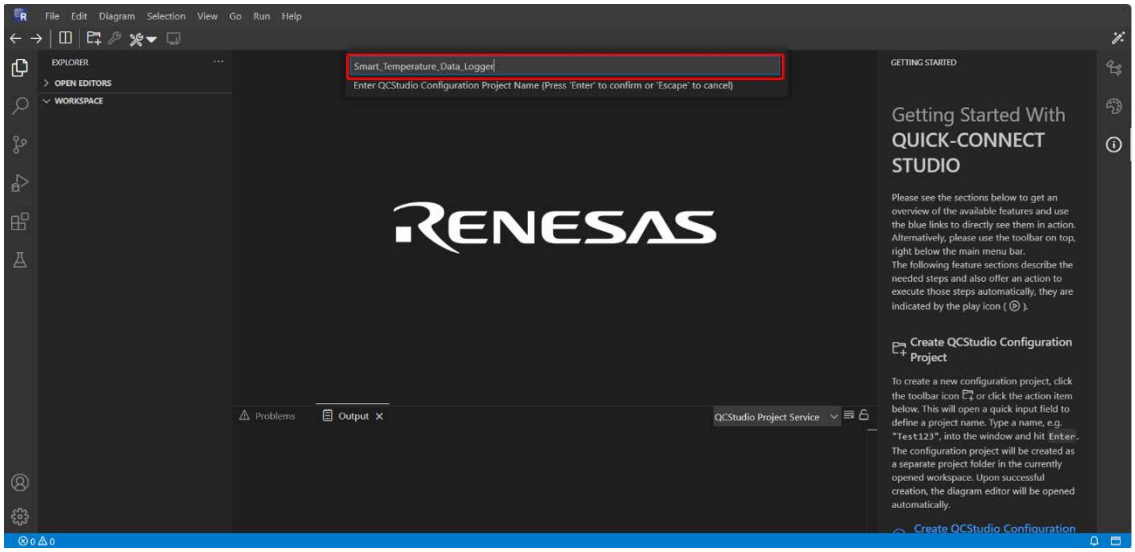




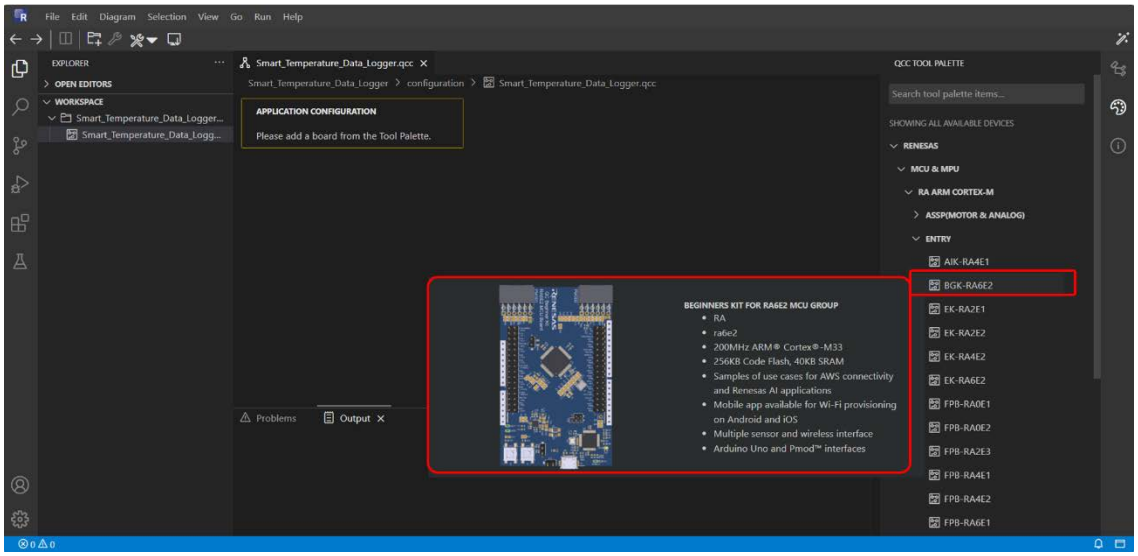
7. Project 2 - Smart Temperature Data Logger

7.1 Steps to Create the Project

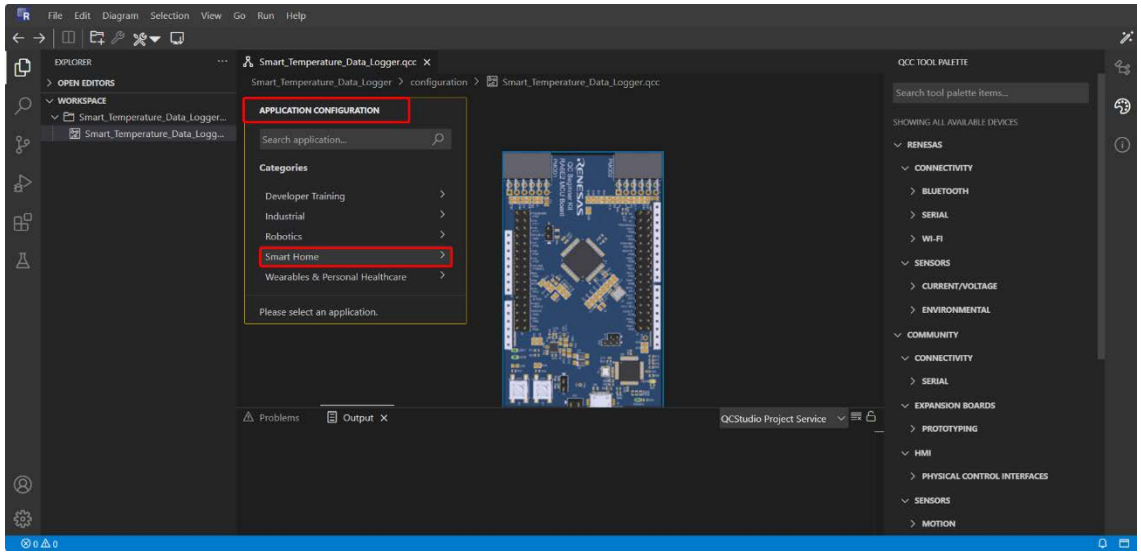
1. Follow the steps in the **Getting Started** section to create a new workspace and a project.

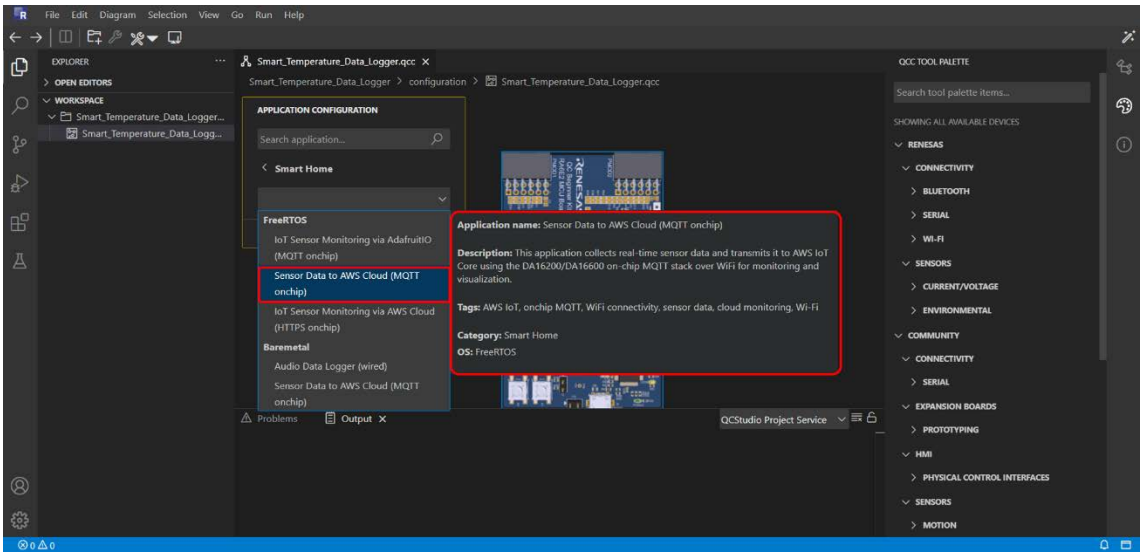


2. Drag and drop the MCU board from the QCC tool palette. In this example, BGK-RA6E2 is used.

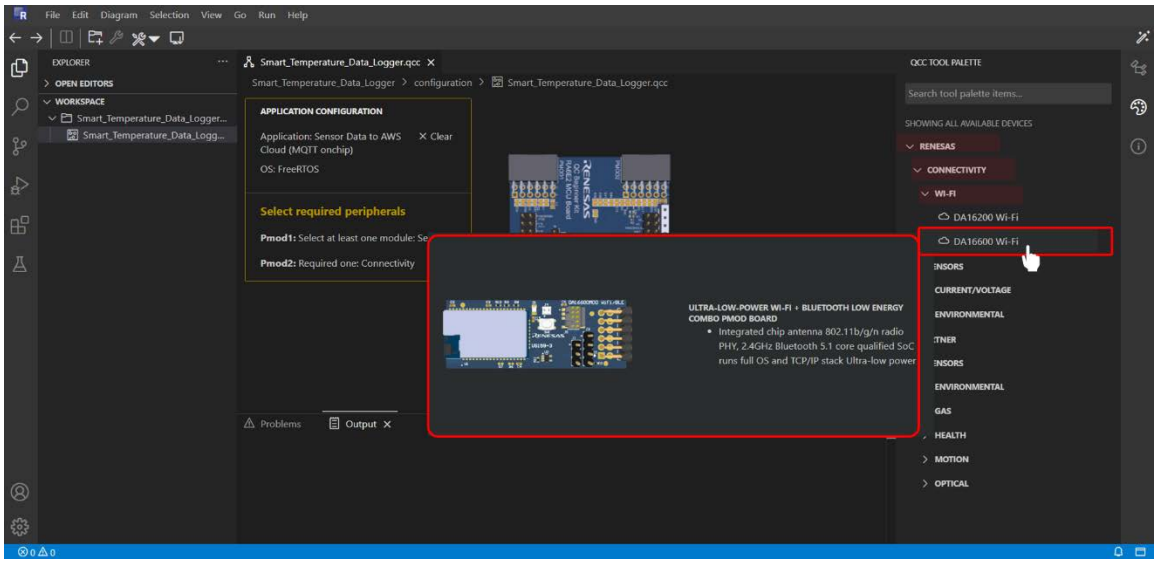


3. In the Application Configuration window, select market segment as **IoT**, then **Other**, and finally **Sensor Data to AWS Cloud (MQTT onchip)**.

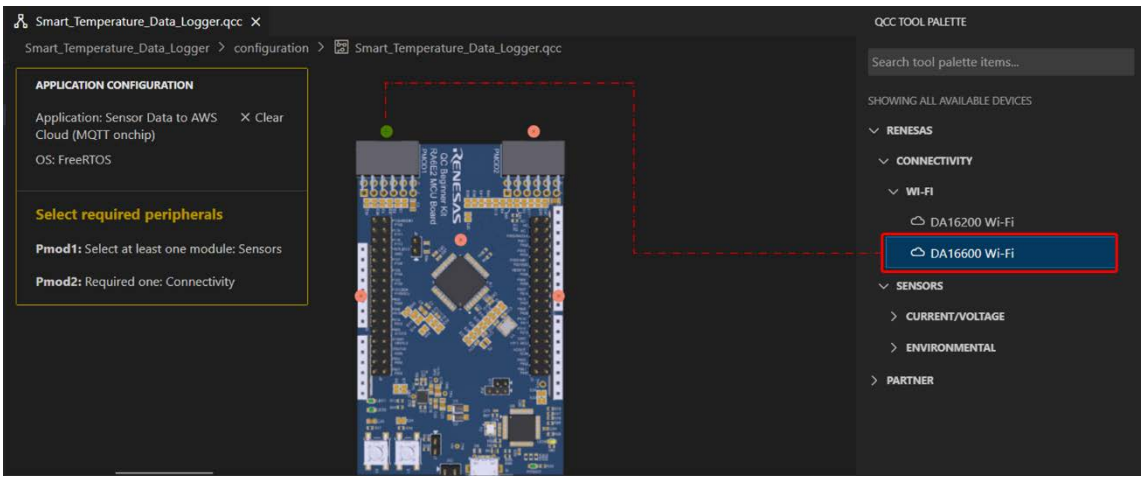




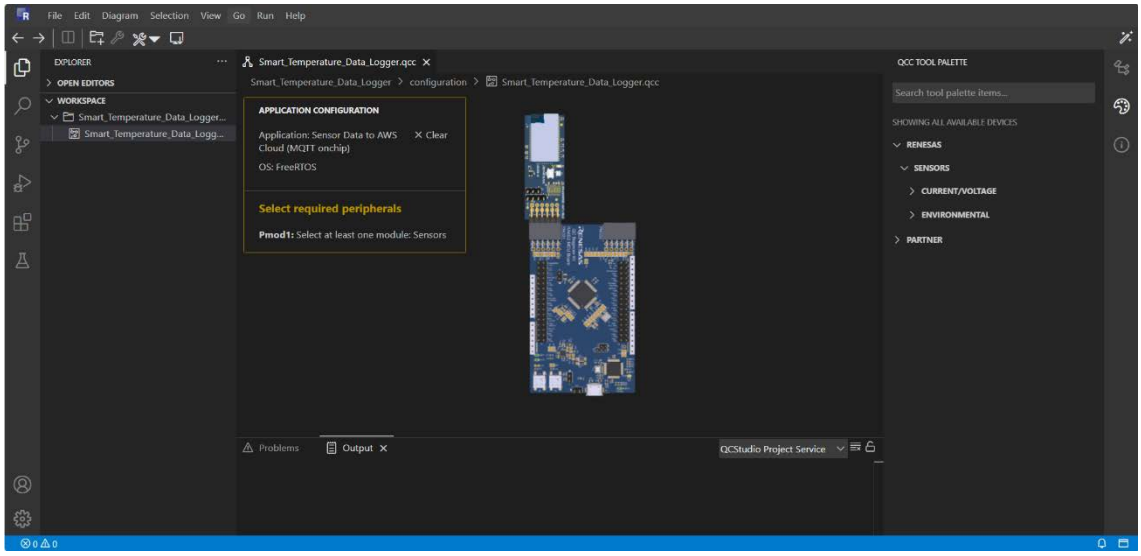
4. After the application configuration is selected, the tool automatically filters and displays the supported connectivity modules. Select **Renesas** > **Connectivity Modules**. Next, drag and drop the connectivity module **DA16600**.



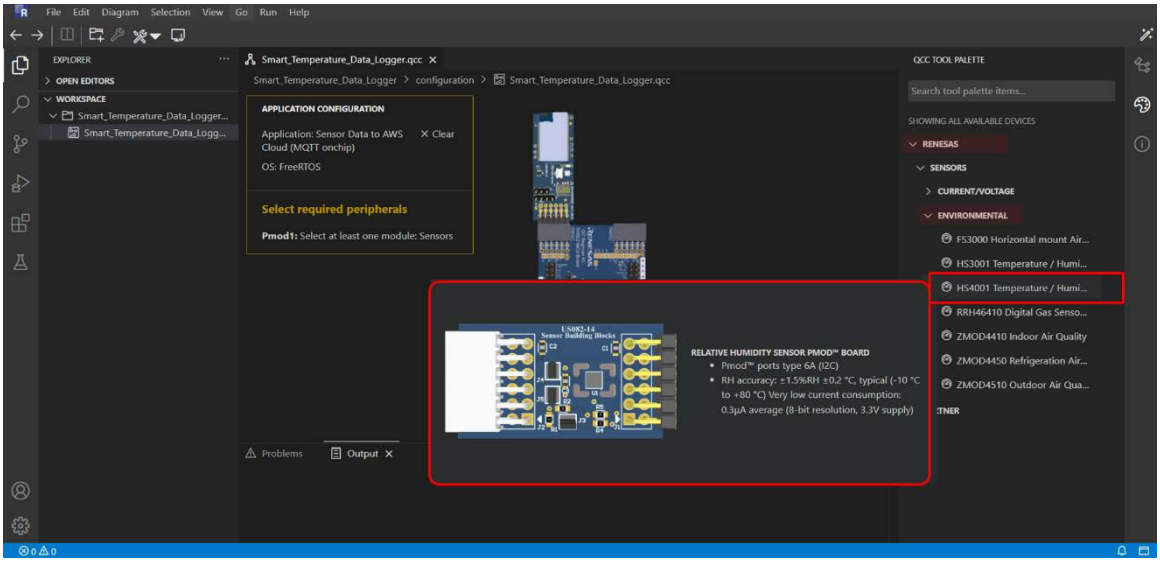
5. Based on the chosen board and application, the QuickConnect tool automatically suggests the PMOD connector compatible with the wi-fi module board.



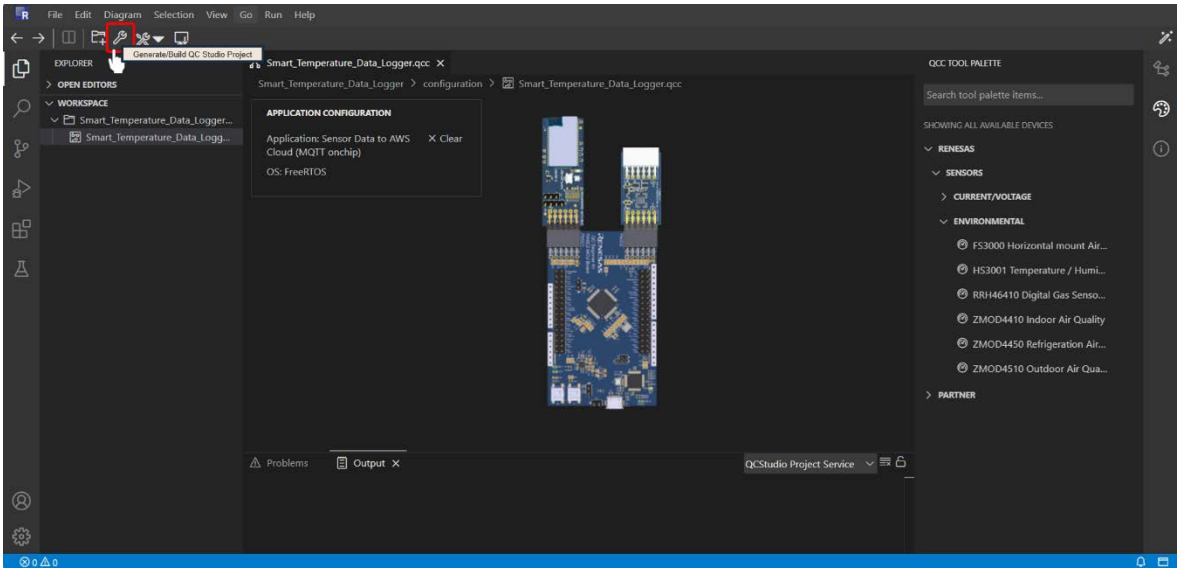
6. Connect the Wi Fi module DA16600 to the MCU board BGK RA6E2.



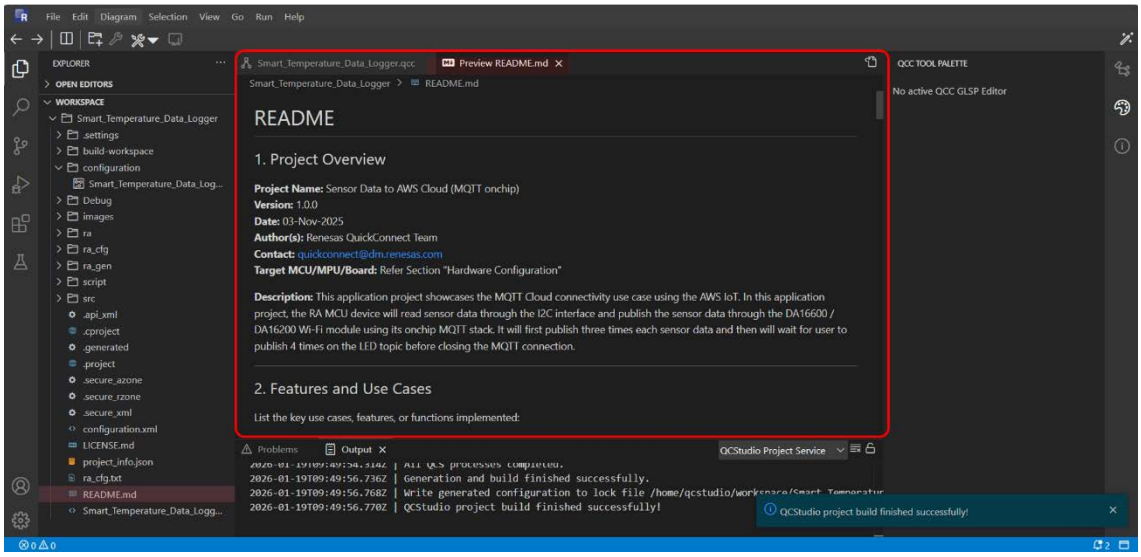
7. Select **Renesas** > **Sensors** > **Environmental**. Next, drag and drop the humidity and temperature sensor board.



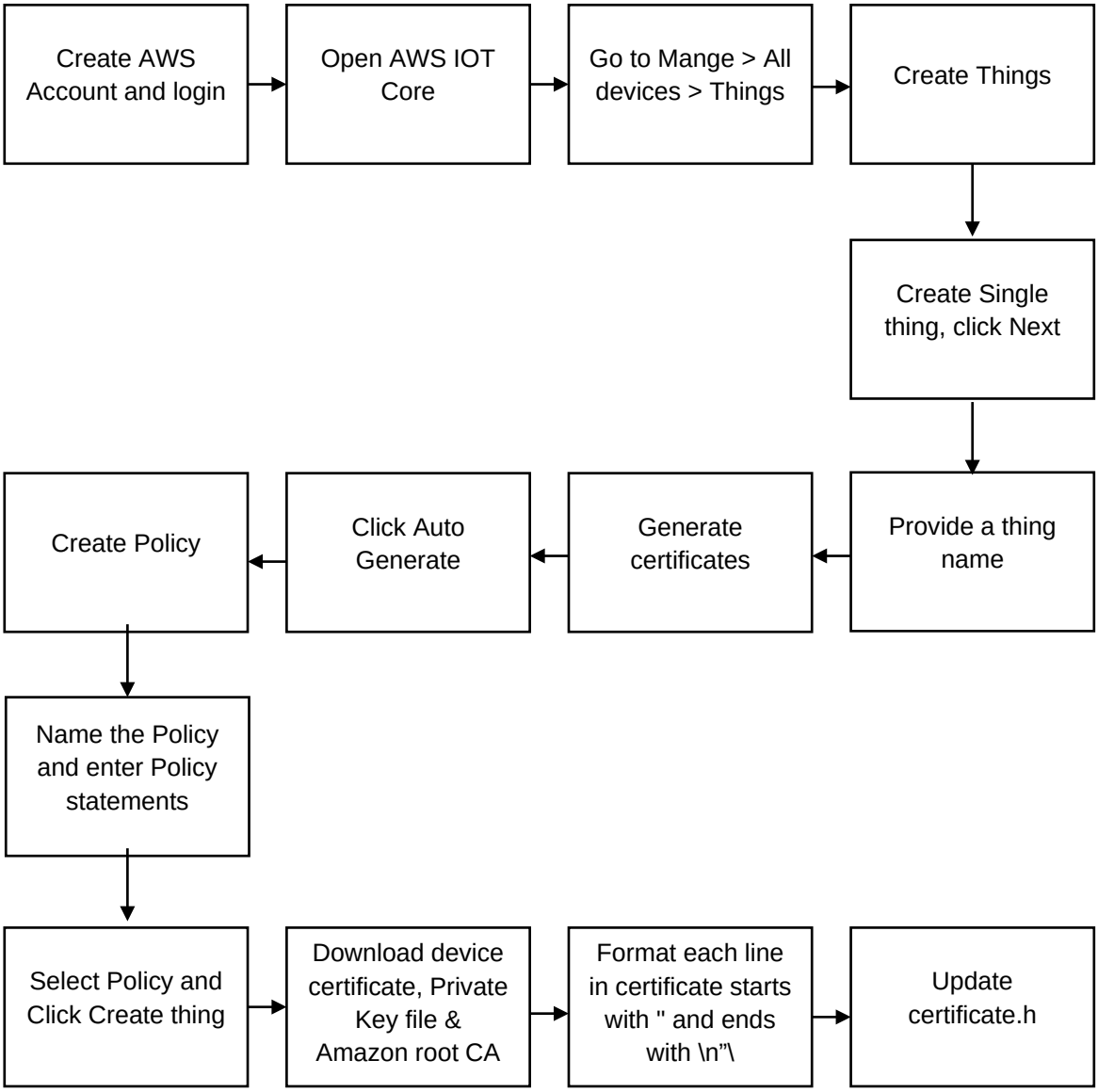
8. To generate and build a project, click on the Generate/Build QCS Project icon on the top left-hand side corner. QuickConnect Studio automatically generates the required software package including drivers, middleware, and network stacks required for the user-created system solution.



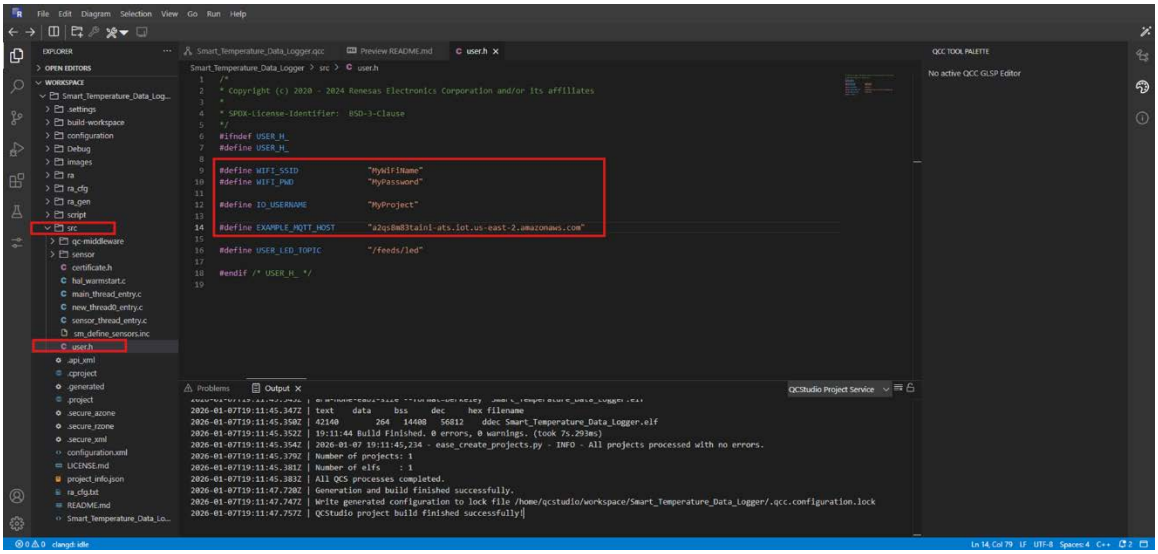
9. To run the application project, refer to the instructions in the **README.md** file located in the project directory. In this example, the Wi-Fi SSID and password must be entered, along with the username and AWS endpoint, should be edited in user.h.



10. Additionally, certificate.h must be updated by adding the device certificate, private key, and Amazon root CA. Refer to the workflow below for setting up an MQTT Thing in AWS IoT Core.

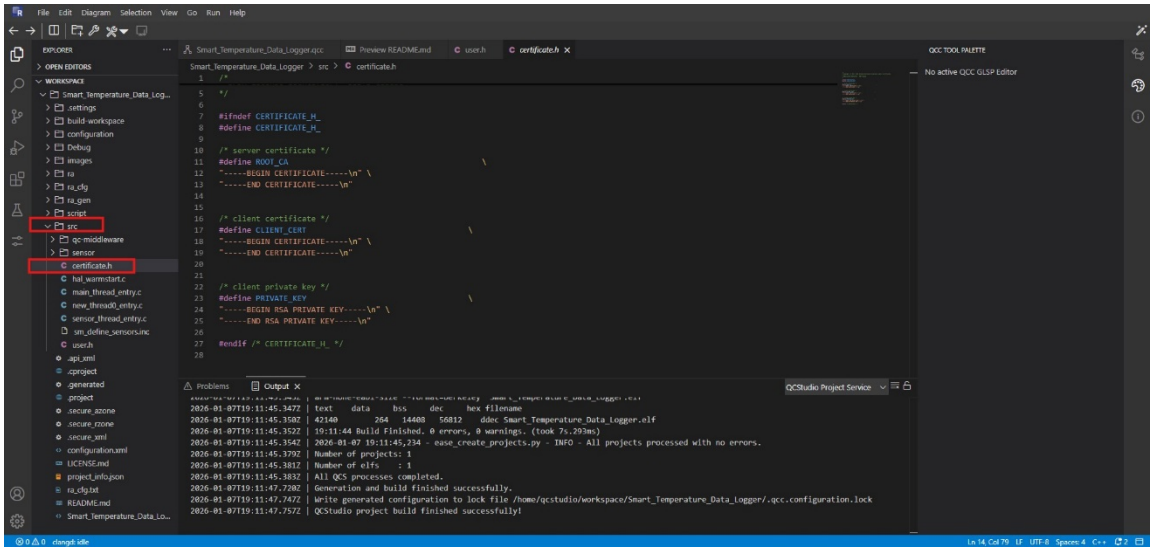


11. The user.h file must be updated with Wi-Fi SSID, Wi-Fi password, IO Username, and MQTT host (AWS data endpoint). For AWS endpoint, Login to the AWS account > select **AWS IO**T > Go to **Settings** > Expand **Domain configuration** > Copy domain **Name**.



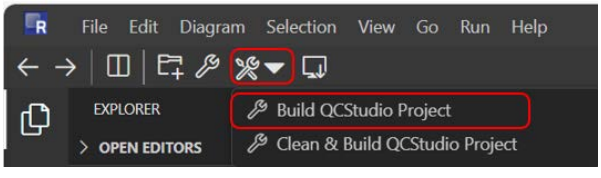
Note: Renesas recommends not to edit the User_LED_Topic or define the sensor topic, as the tool automatically takes care of it.

12. Update the downloaded certificates in the **certificate.h** file. Make sure to download the Root CA, Client Certificate (that is, device certificate), and Private Key file. Additionally, ensure that the macros are edited before uploading.



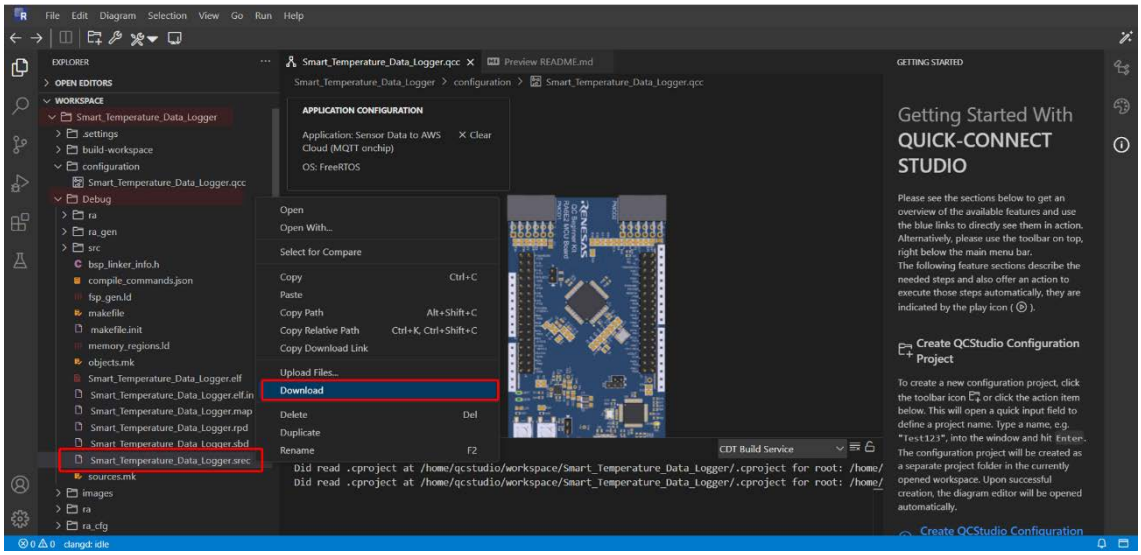
Note: The certificate values or lines provided in above screenshot are just for sample. Refer to the attached screenshot only to ensure the correct format.

13. After making changes in the corresponding .c and .h files, rebuild the application project by selecting **Build QCStudio Project** from the drop-down menu.

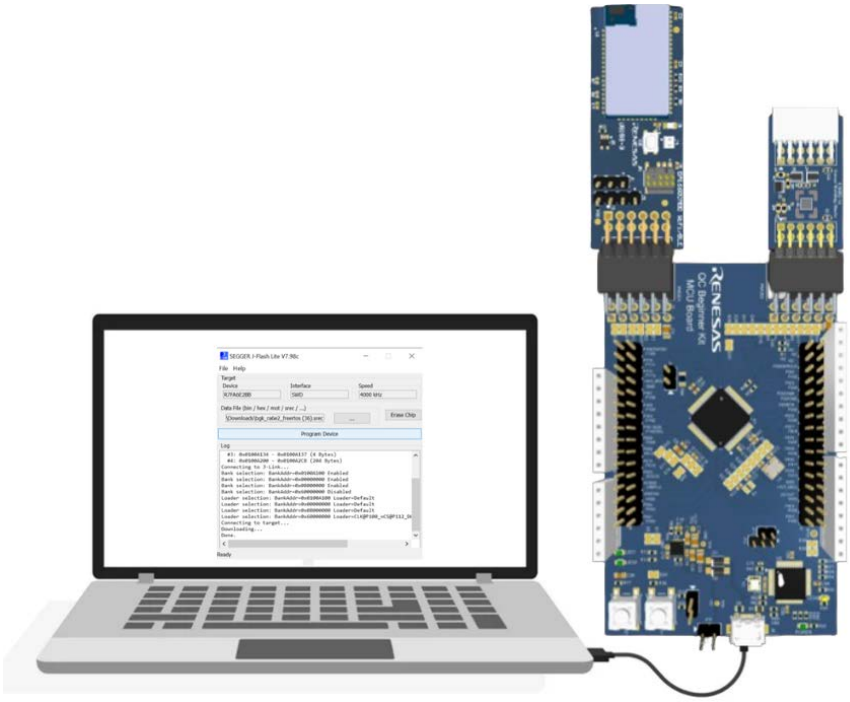


7.2 Programming Hardware and Viewing Results

1. Follow the steps in Steps to Create the Project to create the solution.
2. The application project output files can be found under the **bgk_ra6e2_freertos** > **Debug** folder. Right-click on the **.srec** file and download it to the local PC.



3. Connect the BGK-RA6E2 to the laptop using a USB cable. Refer to the section, Flashing Code to the Hardware using SEGGER J-Flash Lite, to flash the code.

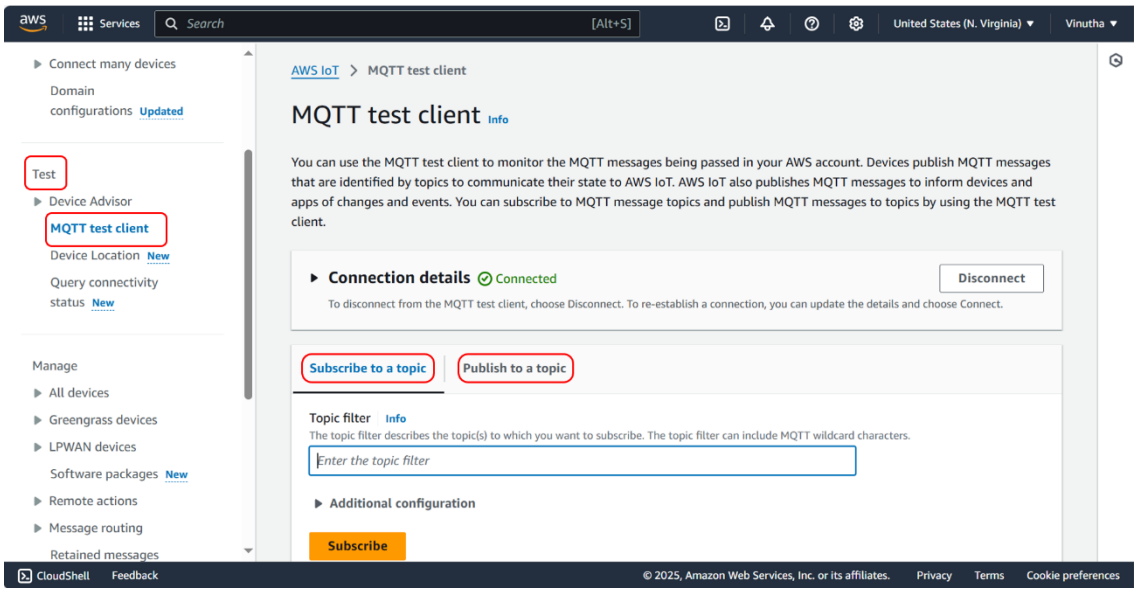


4. Use Jlink Flash programmer to program the **.srec** file into the required MCU kit (QuickConnect Beginner kit). Reference section, Flashing Code to the Hardware using SEGGER J-Flash Lite, to flash the code.
5. View the output on the AWS MQTT test client by subscribing and publishing the data.

Steps to be followed:

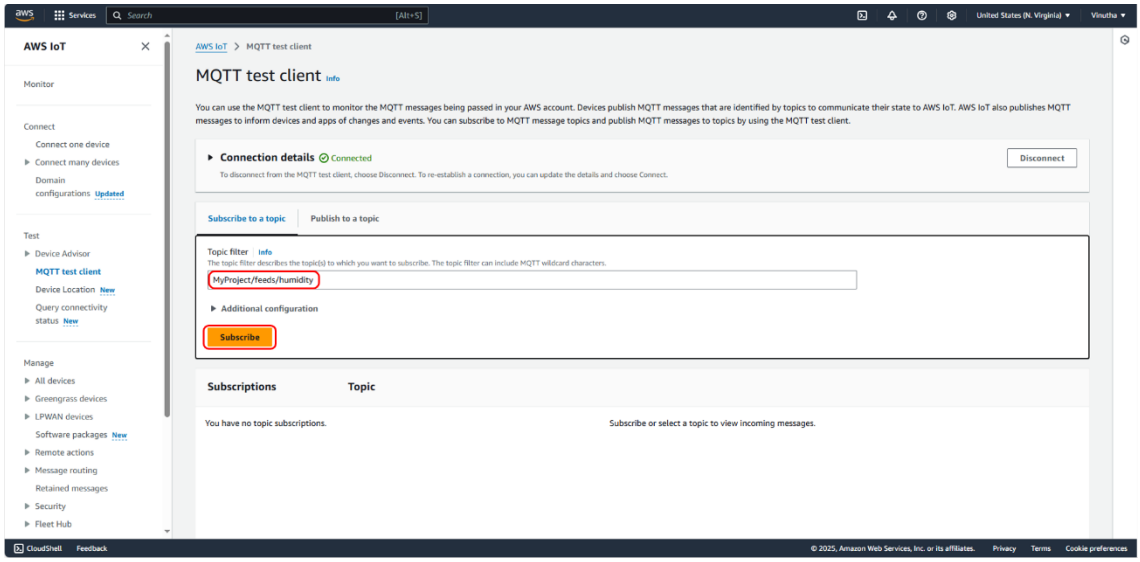
a. Log in to the AWS account and open AWS IoT Core. Follow the instructions according to **README.md** to create a thing.

b. In the left console, under **Test**, click on **MQTT test client**. A window appears for subscribing and publishing a topic.

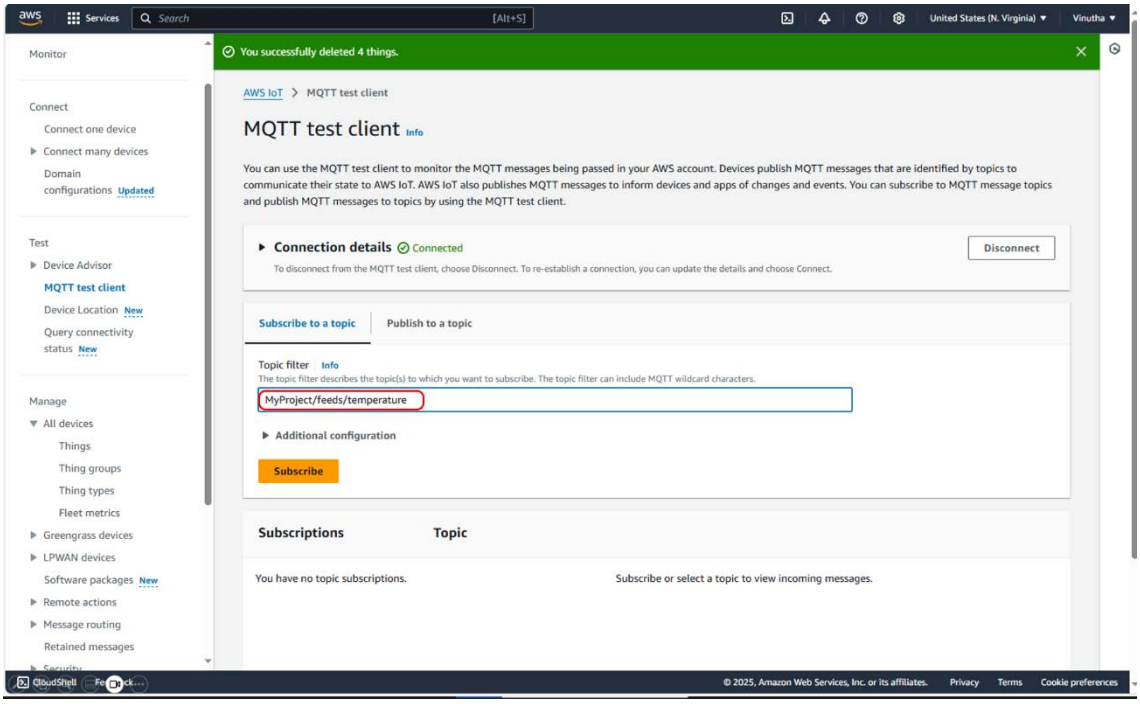


Note: Ensure by subscribing to the topic # and check the connection details.

c. In the topic filter, enter "UserName/feeds/humidity" and click **Subscribe** to see the humidity values.

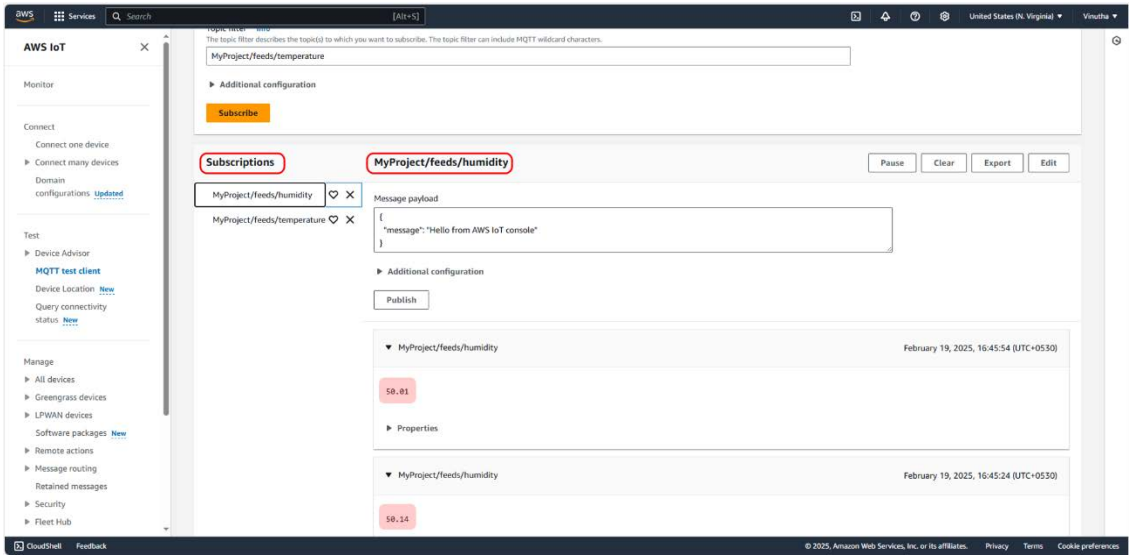


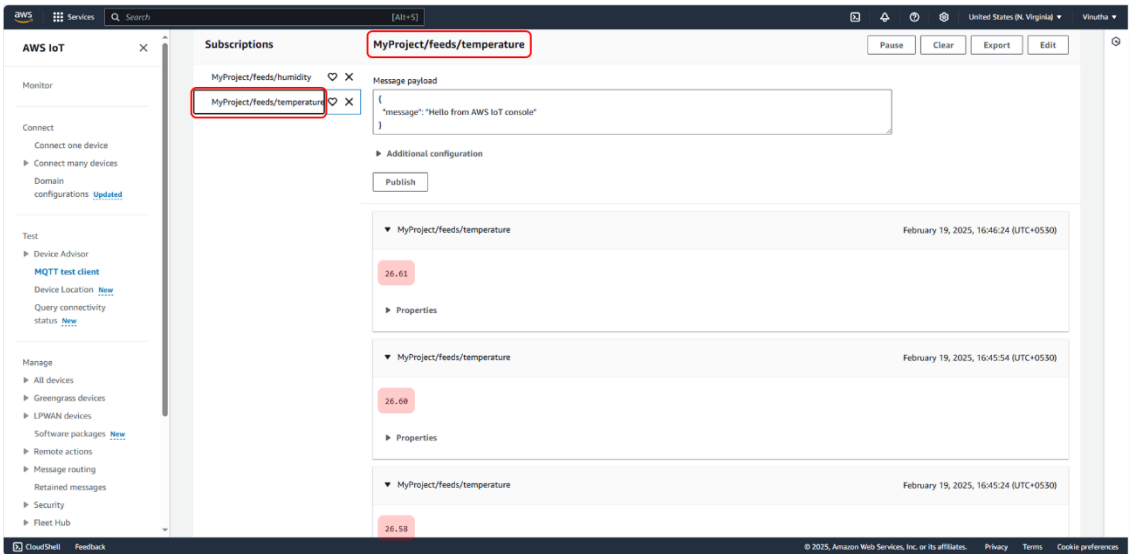
d. Similarly, enter "UserName/feeds/temperature" and click **Subscribe** to see the temperature values.



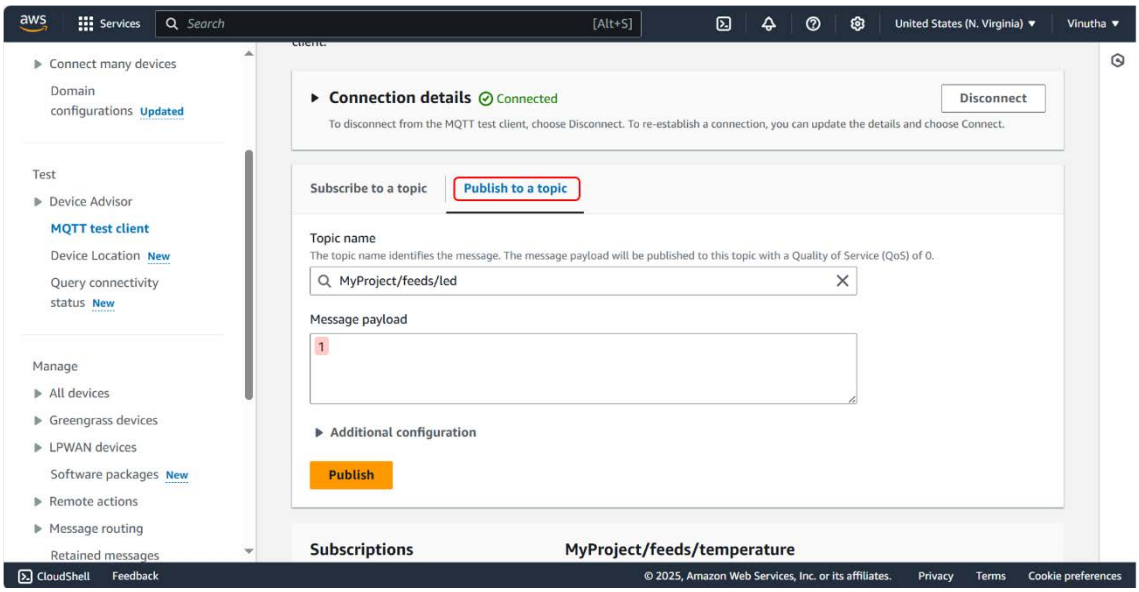
Note: The username should be the same as what was entered in the **user.h** file in the QuickConnect Studio.

e. Observe the output humidity and temperature values.





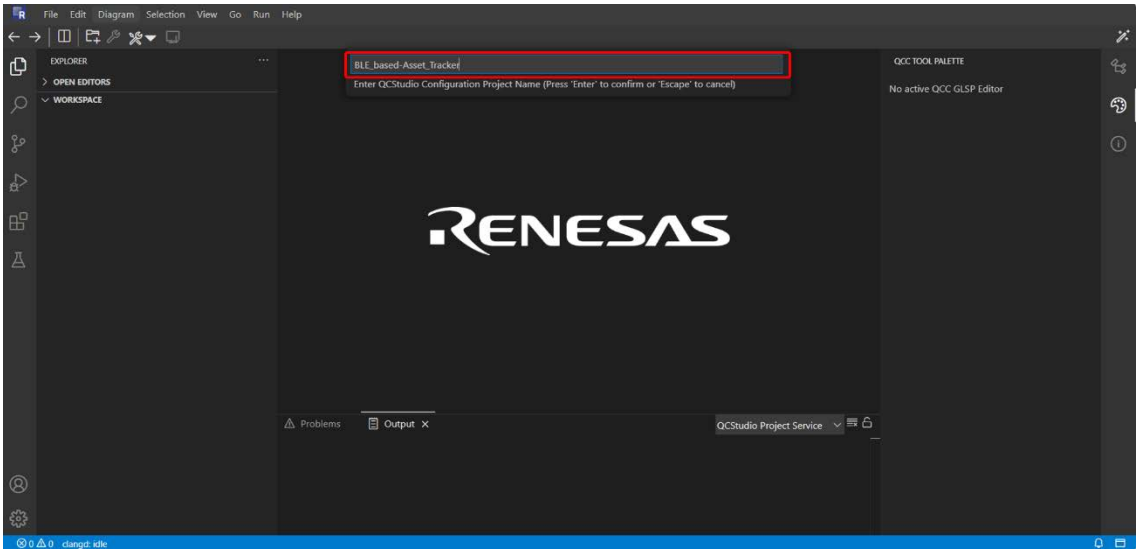
f. The feature to publish to a topic can also be used by giving the topic name as "UserName/feeds/led" with the message payload set to 0 to turn off the LED and 1 to turn on the LED.



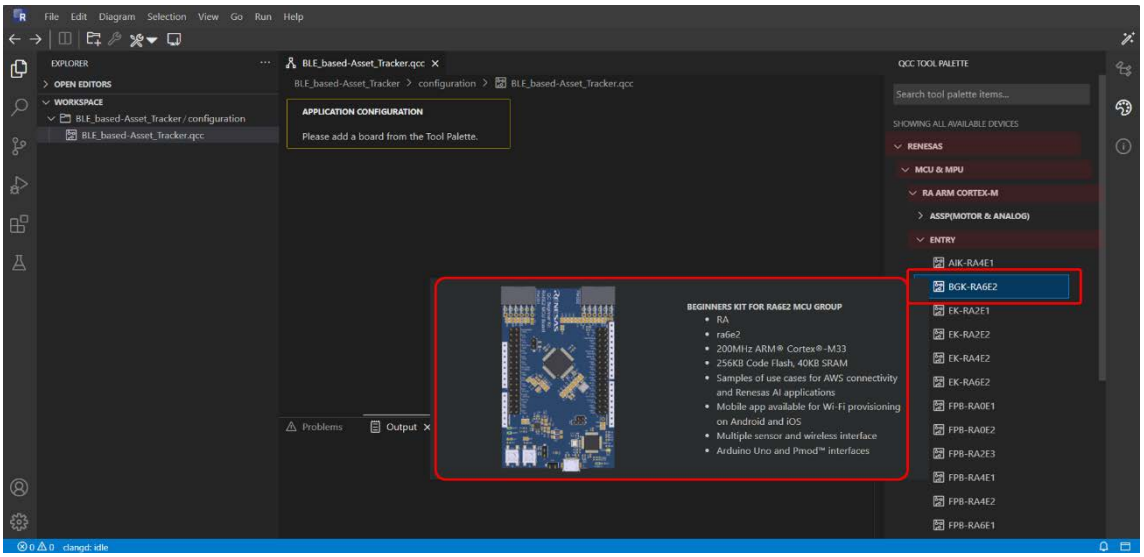
8. Project 3 - BLE-based Asset Tracker

8.1 Steps to Create the BLE-based Project

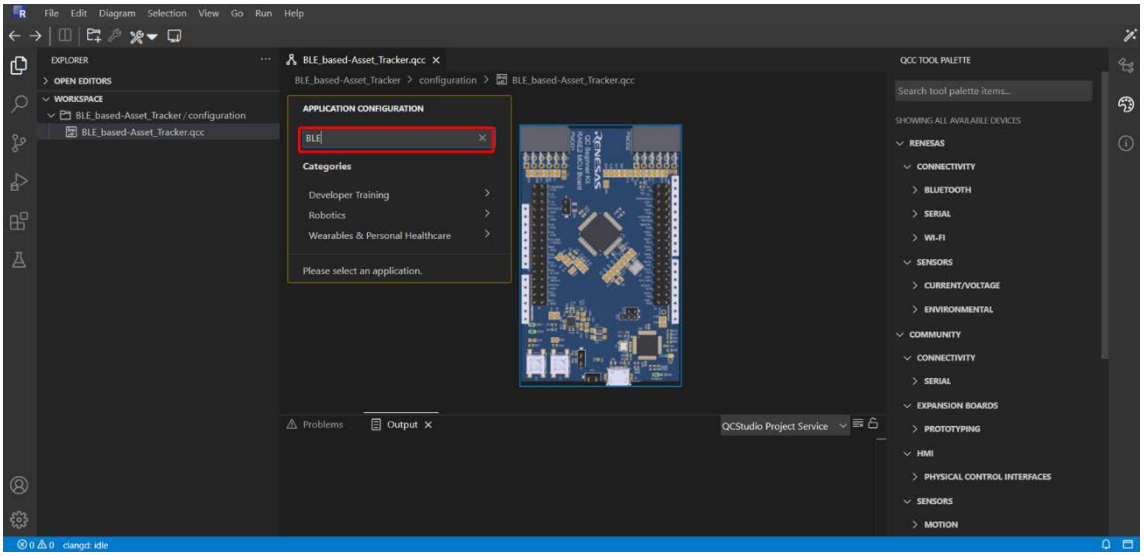
1. Follow the Steps to Create the Project section to create a new project.

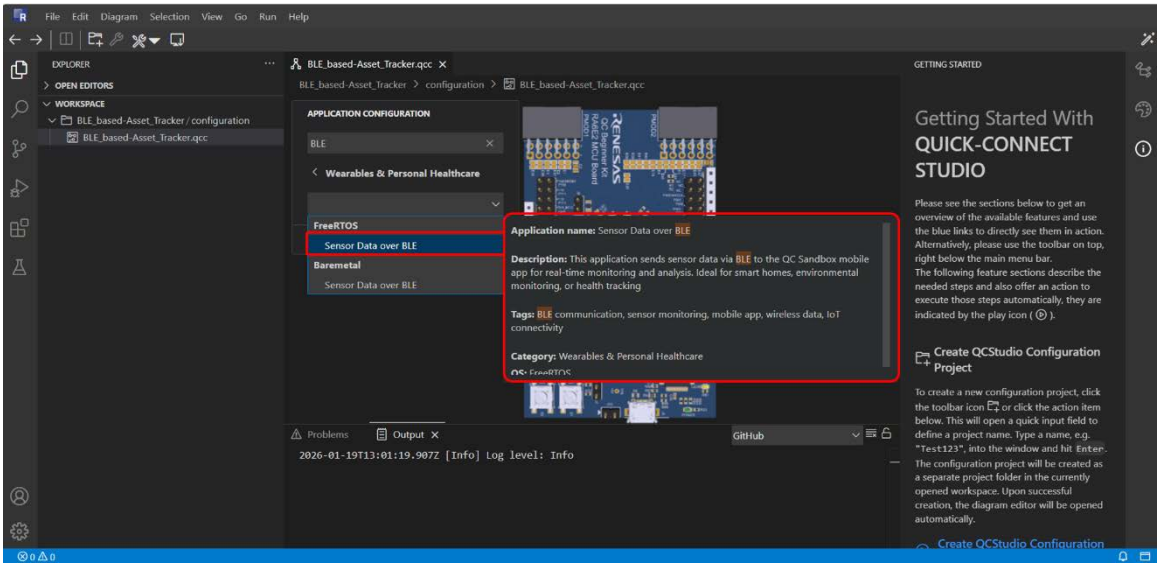


2. Drag and drop the MCU from the QCC tool palette. In this example, BGK-RA6E2 is used.

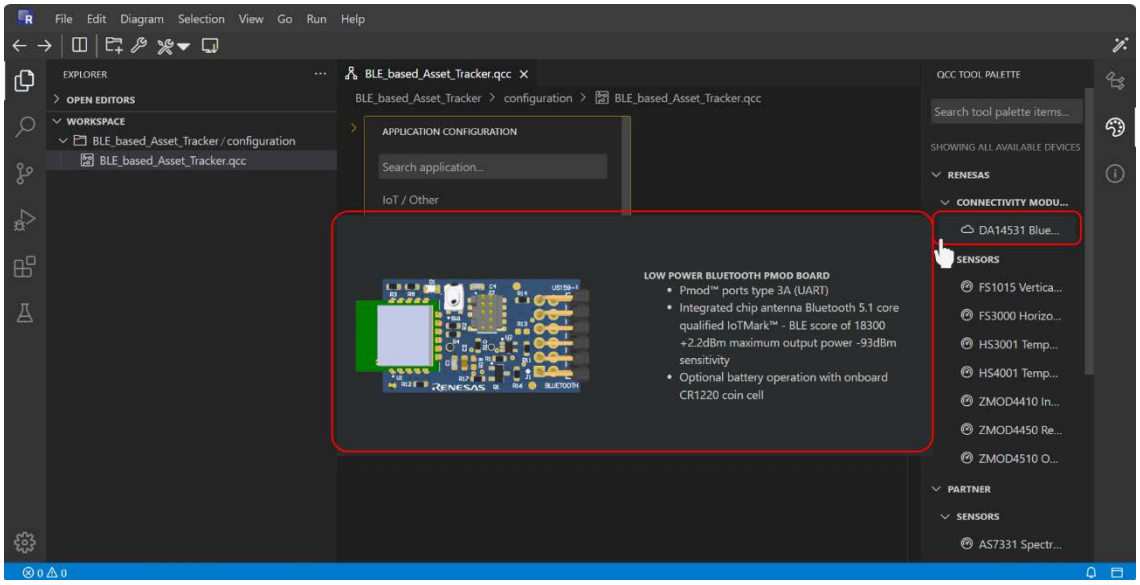


3. Choose the Application Configuration. In the **Application Configuration** window, first select **Wearables & Personal Healthcare**. Next, click on **v** (the downward arrow) and select **Sensor Data over BLE**.

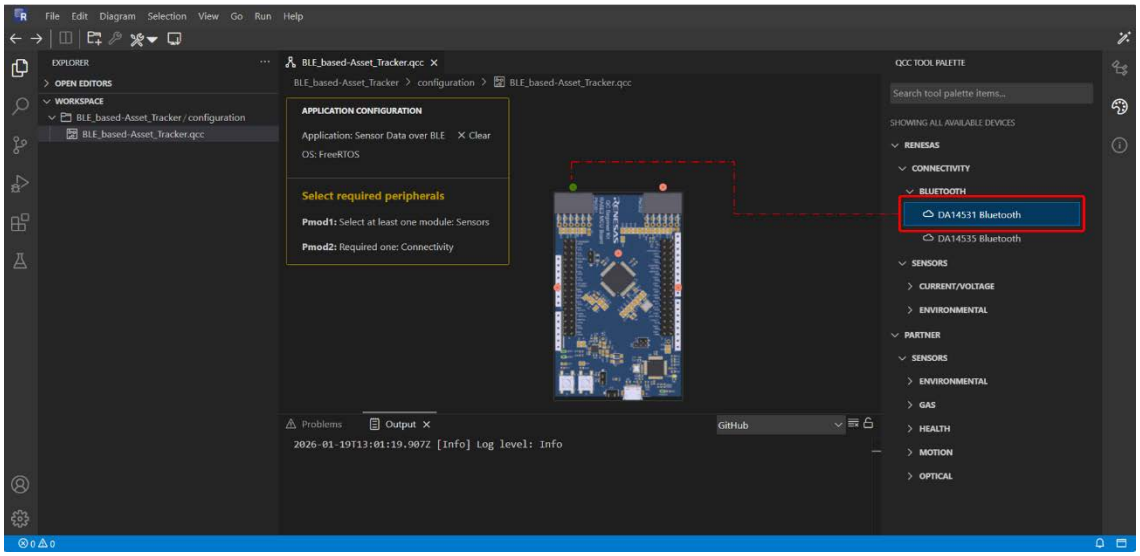


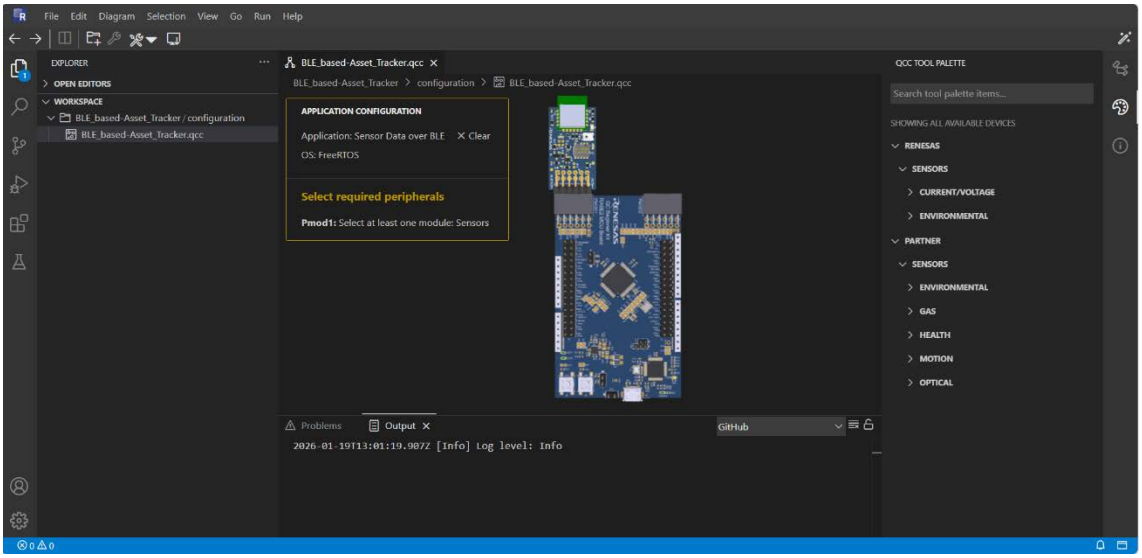


4. Select **Renesas** > **Connectivity Modules**. Next, choose the Bluetooth module **DA14531**.

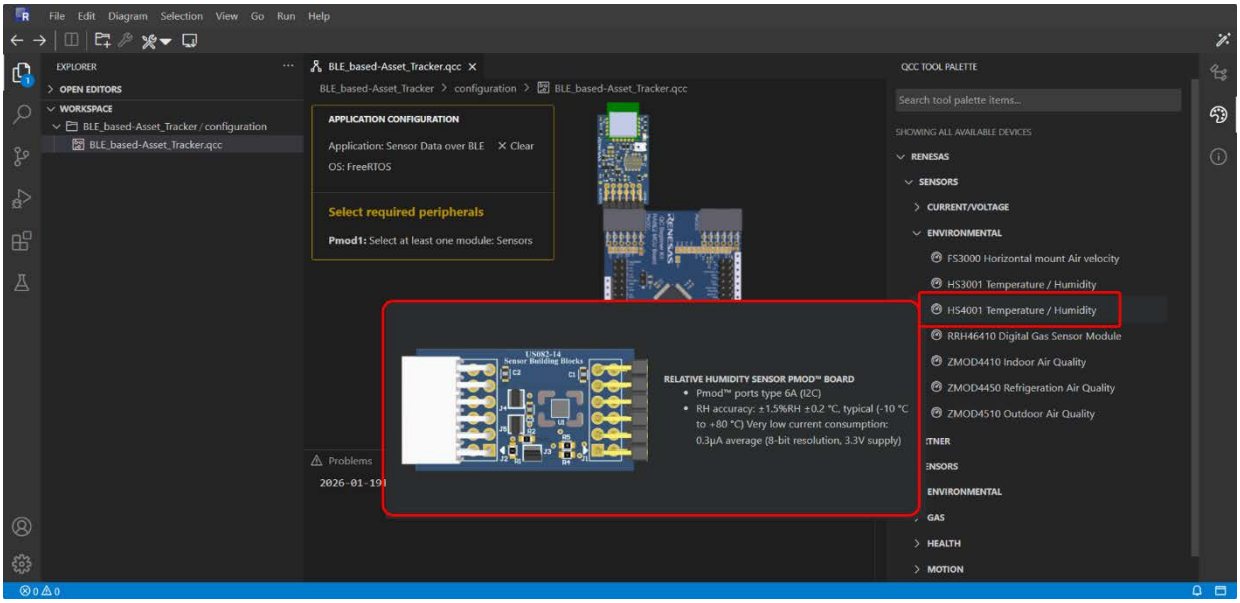


a. Based on the chosen board, the QuickConnect tool automatically suggests the PMOD connection compatibility

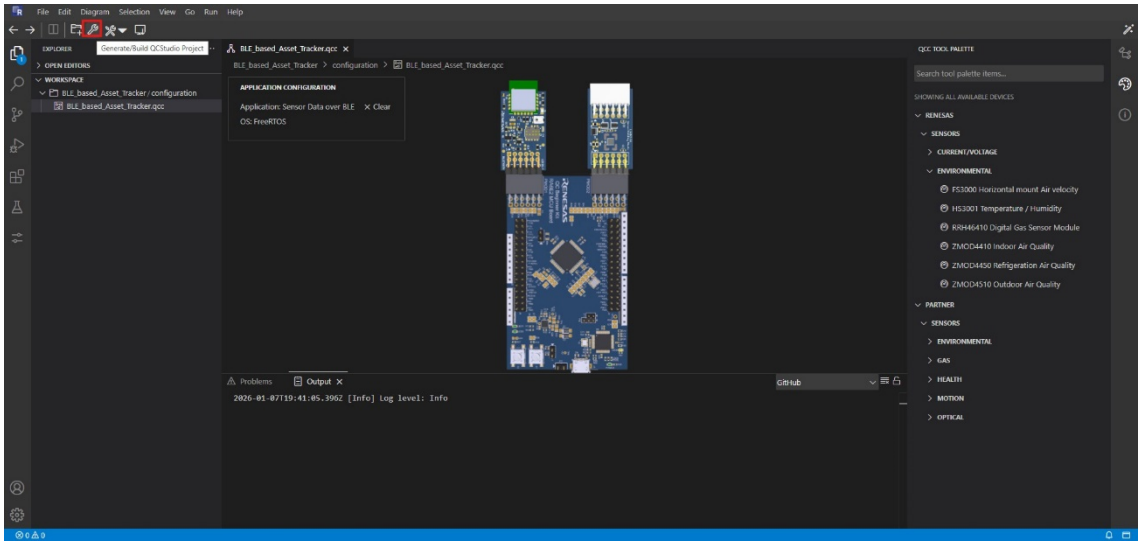




5. Select **Renesas** > **Sensors** > **Environmental**. Next, drag and drop the humidity and temperature sensor in the reference application (HS4001 is used).

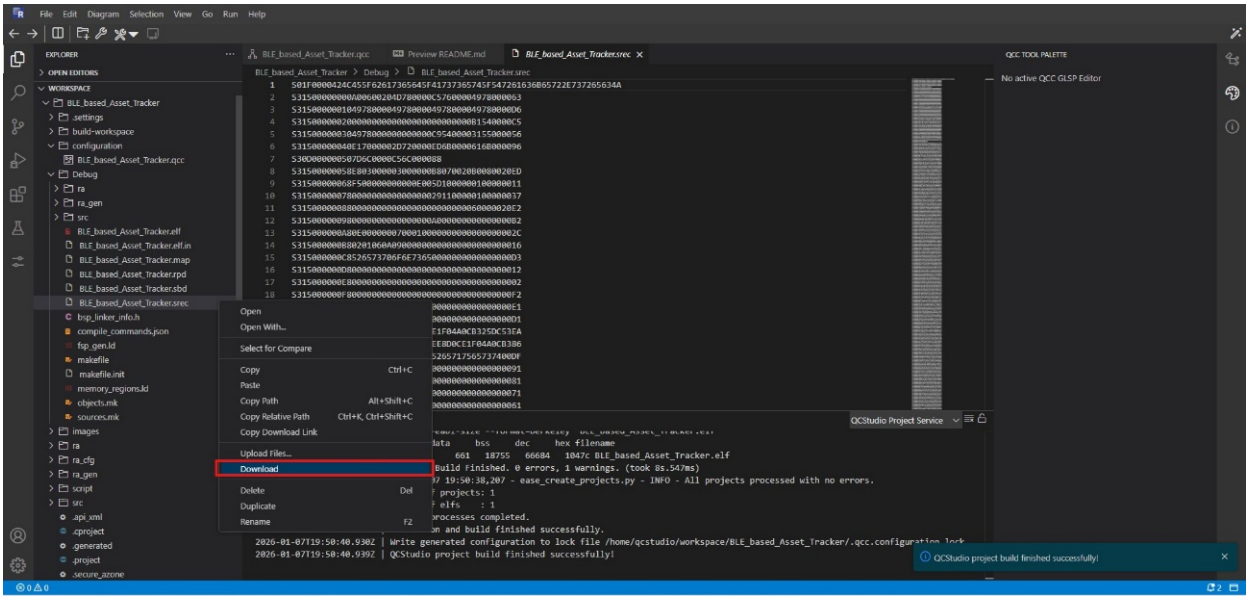


6. To generate and build project, click on the Generate/Build QCS Project icon on the top left-hand side corner. QuickConnect Studio automatically generates the required software package including drivers, middleware, and network stacks required for the user-created system solution.



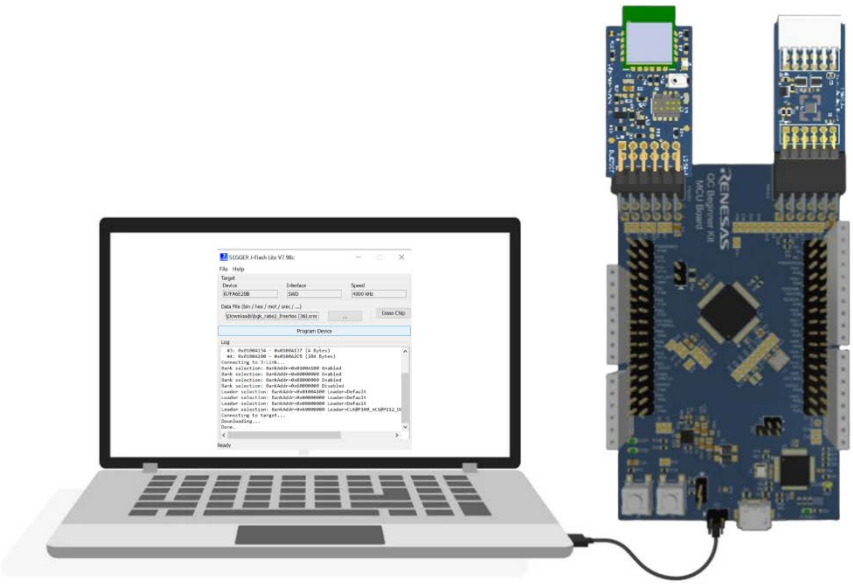
8.2 Programming Hardware and Viewing Results

1. Follow the steps in the Steps to Create the Project section to create the solution.
2. The application project output files is found under the **Debug** folder. Right-click on the **.srec** file and

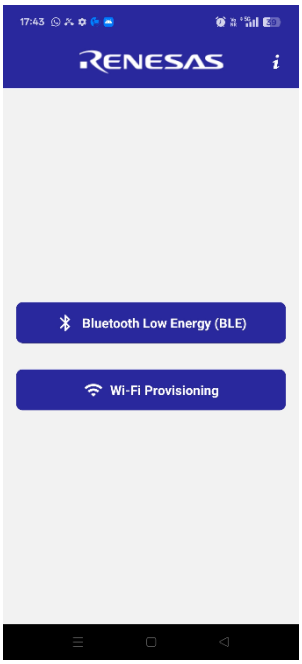


download it to a local PC.

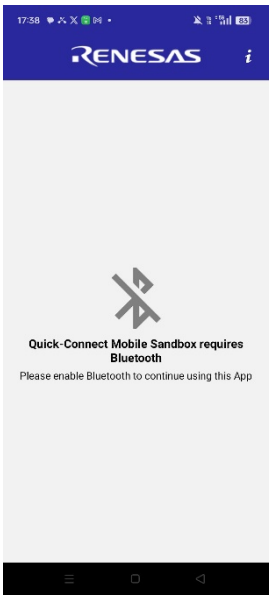
3. Use the J-link Flash programmer to program the **.srec** file into the desired MCU kit. Refer to the Flashing Code to the Hardware using SEGGER J-Flash Lite section.

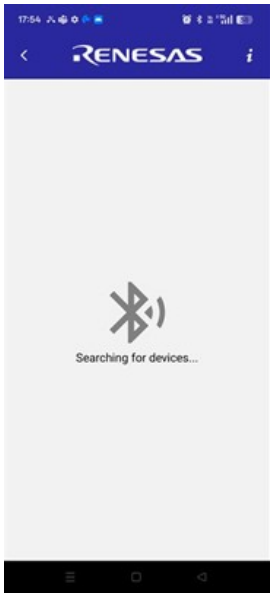


4. View the output on QC Sandbox by downloading the QC Sandbox app.
5. As soon as the application is open, it asks you to select either **Bluetooth Low Energy (BLE)** or **Wi-Fi Provisioning**. For this reference example, select **Bluetooth Low Energy (BLE)**.

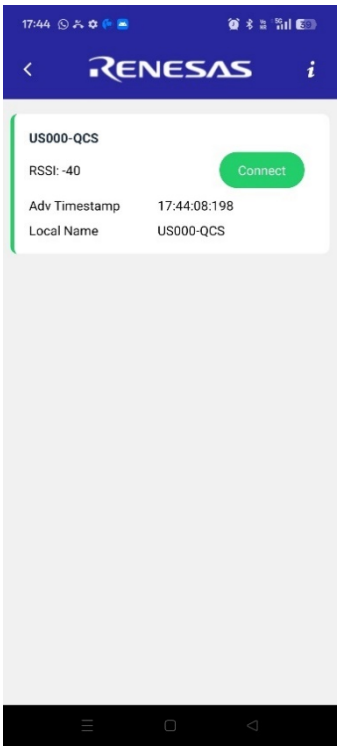


6. To view the output:
   b. Turn on Bluetooth on the device.

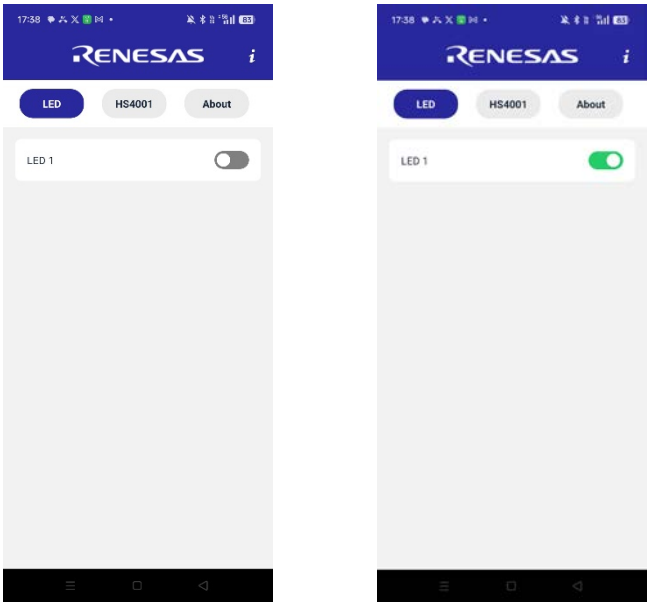




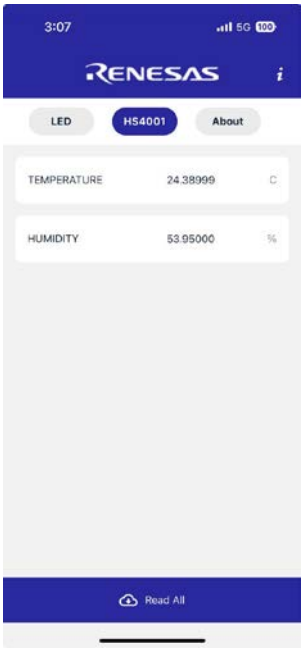
7. Scan to connect and click on the **Connect** button.



8. Select the **LED** option on the QC Sandbox app and toggle the **LED** switch to turn it on and off. Observe the output.



9. Select the sensor **HS4001** on the mobile app to see the humidity and temperature sensor values.



9. Debugging on QuickConnect Studio

9.1 Remote Debugging

Projects can be debugged remotely when it is preferred to validate the generated reference application onto the target hardware before obtaining the boards, and remote board farms deployed globally can be used. The remote debugging feature is supported for the QuickConnect beginners Kit.

For more information, refer to the *QuickConnect Remote Debugging Manual* document available in the Documents section of the Renesas QuickConnect webpage.

9.2 Direct Debugging

The Direct Debug feature in Renesas QuickConnect Platform enables developers to perform Hardware-Assisted Software debugging on the Renesas RA MCU hardware kits directly from a browser window, eliminating the requirement for local IDEs, toolchains, or complex driver setups.

It enables minimal setup debugging so that users can plug in their evaluation kit, open the QuickConnect platform, design an application, and start debugging instantly.

For more information, refer to the *QuickConnect Direct Debugging Manual* document available in the Documents section of the Renesas QuickConnect webpage.

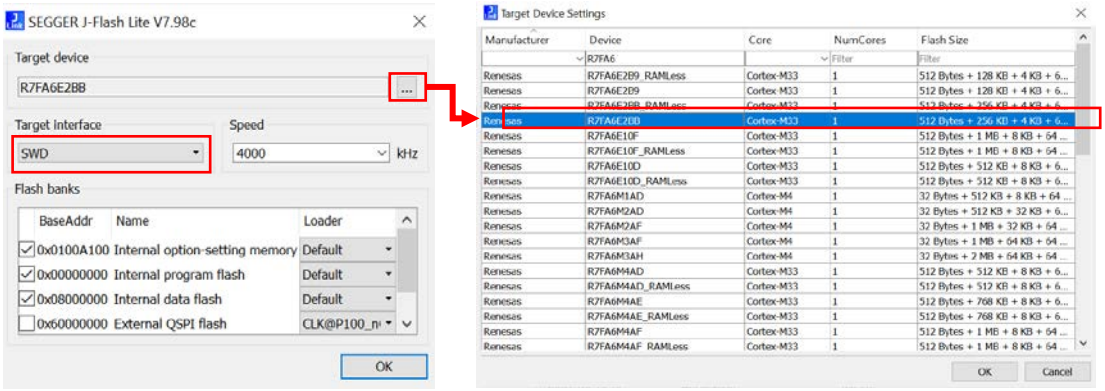
10. Appendix

10.1 Flashing Code to the Hardware using SEGGER J-Flash Lite

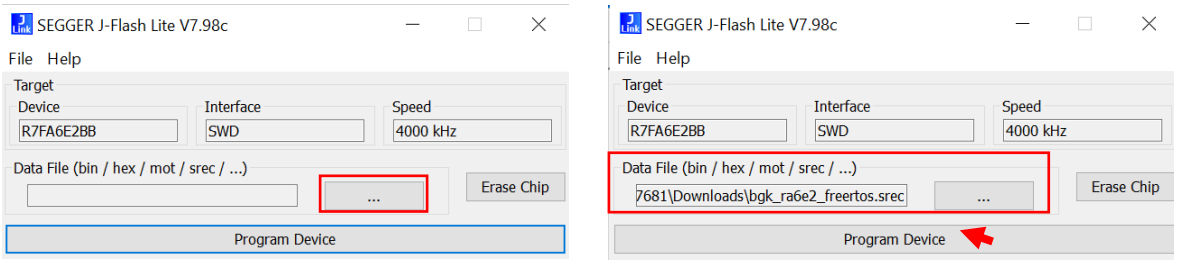
Download the Segger software to the local PC if it is not already installed. According to the device compatibility, choose the installer under J-Link Software and Documentation Pack, Renesas recommends installing the 64-bit installer.

1. Open SEGGER J-Flash Lite:
   a. Navigate to the **Program Files** on the PC.
   b. Open the **SEGGER – Jlink** folder.
   c. Launch **JFlashLite.exe**.

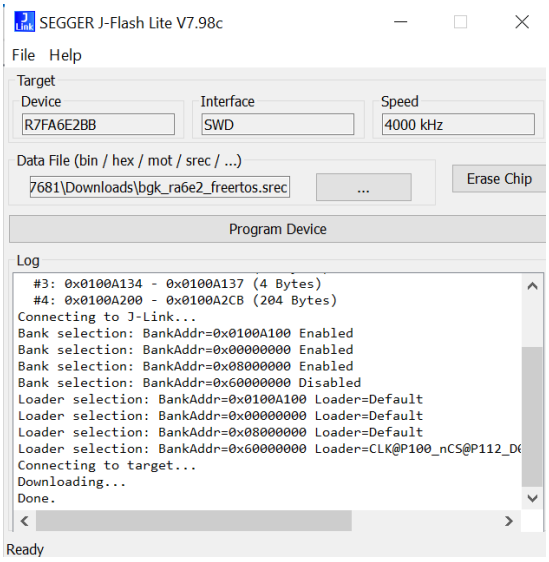
2. Select Target Device:
   a. In the J-Flash Lite window, click on the **(…)** button next to the **Target Device** field.
   b. A new window appears. Here, the user can select the manufacturer and device.
   c. For this project, the **RA6E2 MCU** is used, search for the part number **R7FA6E2BB**.
   d. Select the target device and click **OK**.
   e. Ensure the target interface is set to **SWD**.
   f. Click **OK**.



3. Import the **.srec** File:
   a. In the main J-Flash Lite window, locate the **Data File (bin / Hex / mot / srec / …)** section.
   b. Click on the **(…)** button to import the **.srec** file.
   c. Select the **.srec** file that was downloaded by following the steps in the Quick Start procedure.



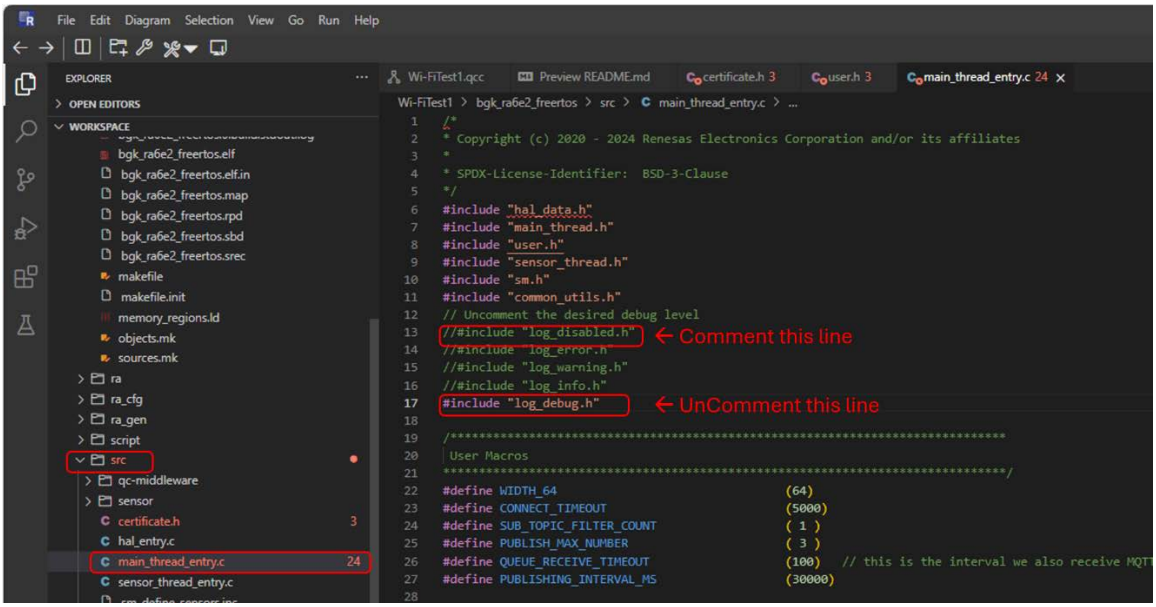
4. Program the Device:
   a. Click on Program Device.
   b. A prompt can appear asking if an update to the latest firmware version is required. Select **No**.
   c. The code now flashes to the MCU.
   d. After the process is complete, the log section of the screen displays **Done**.



10.2 Enable Data Log

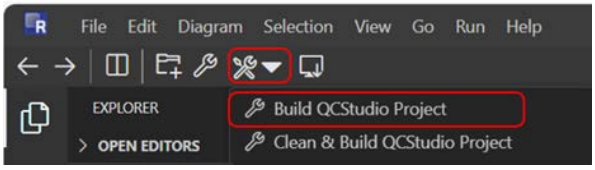
RTT debug logging is a debugging method that is already available in the project generated from the QCS. Enabling this will provide some predefined debugging points through which mistakes in the project can be identified. This is basically some serial messages printed from the application. Complete the following steps to enable this:

1. Go to the **src/main_thread_entry.c** file and make the following changes in the header file inclusion section.

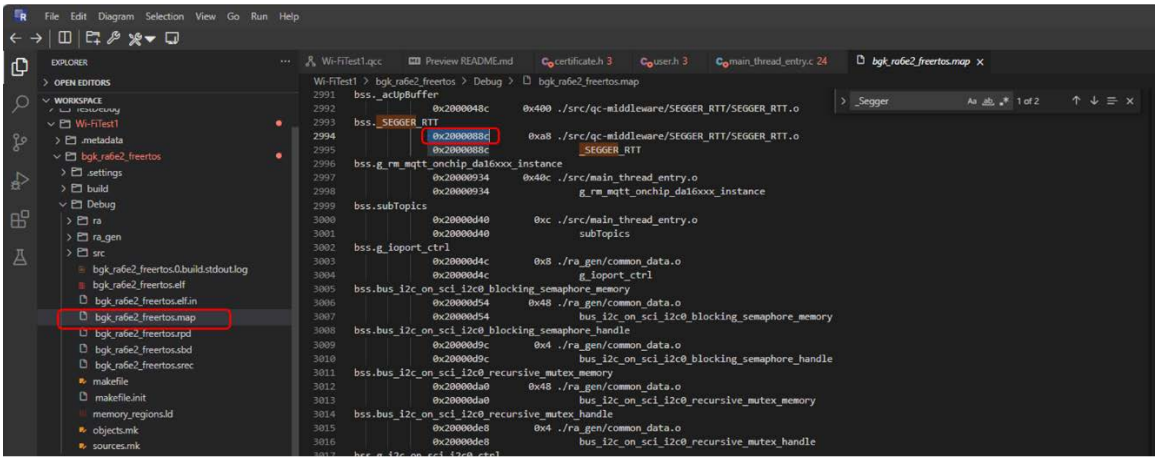


This essentially enables the debug messages on this file. The same can also be followed for sensor files (src/sensor/hs4001_sensor.c).

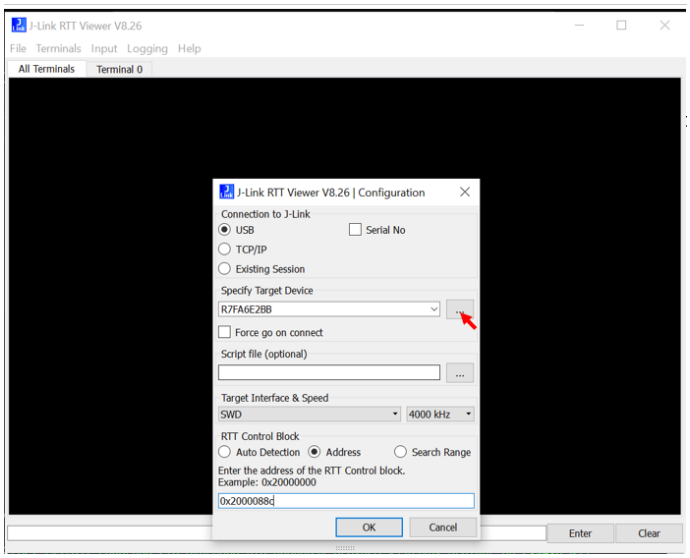
2. Re-compile the project by clicking **Build QCStudio Project**.



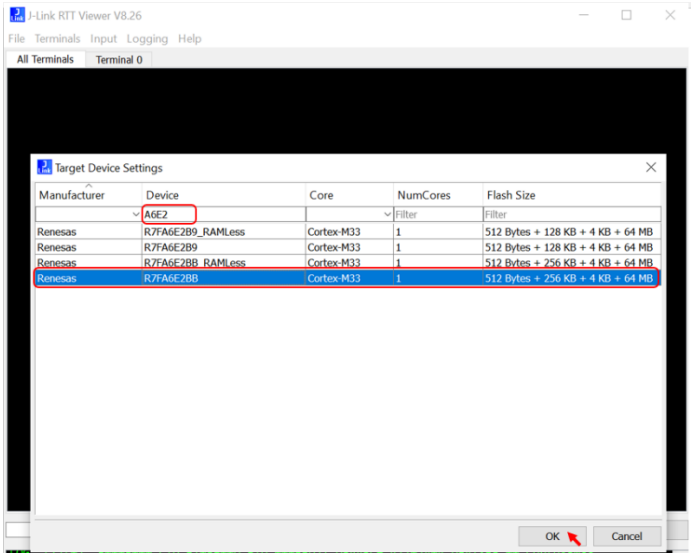
3. Go to **Debug** folder in the repository and open the ***.map** file. This file has the same name as the **srec** file, but the extension is **.map**.
4. Inside the file, search for the word _**segger** and the following appears:



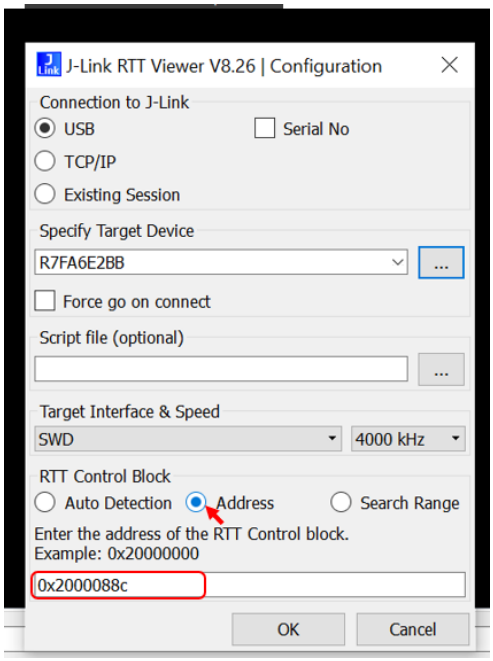
5. A similar address appears in the map file. Copy that address.
6. Flash the **.srec** file to the board.
7. Search the PC for the **JLink RTT Viewer** application and open it.
8. Select the three dots by the Target Device selection.



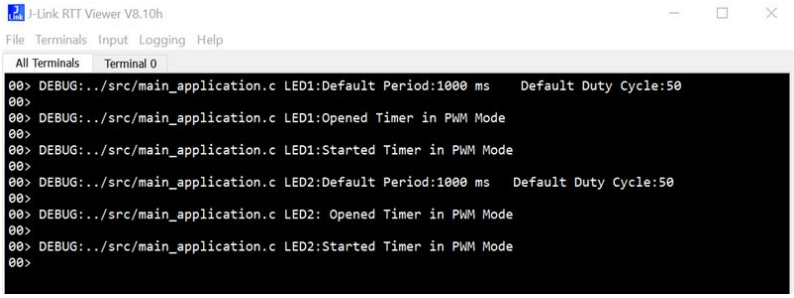
9. On the next window, under Device type **A6E2,** four devices are listed. Select **R7FA6E2BB** and click **OK**.



10. R7FA6E2BB is now selected as the device in first window (see step 8).
11. Under **RTT Control Block**, select **Address** and enter the address copied from step 5.



12. Click **OK** and press No if asked for software update.
13. A window appears:



10.3 Customization

Customization refers to making modifications to the applications. To perform this exercise, it is assumed that the user has built the basic blink, smart temperature data logger, and BLE-based asset tracker.

Note: The sample code for customization is in GitHub.

10.3.1 Foundation for Customization

To update the **QCS project** with modification within the scope of the application, follow these steps:

1. Create the necessary folders in the project to organize and upload the new files.
2. Upload the required files to their corresponding paths within the project structure.
3. Replace any existing files that must be updated with the new versions.

10.3.1.1 Create the Required Folders/files

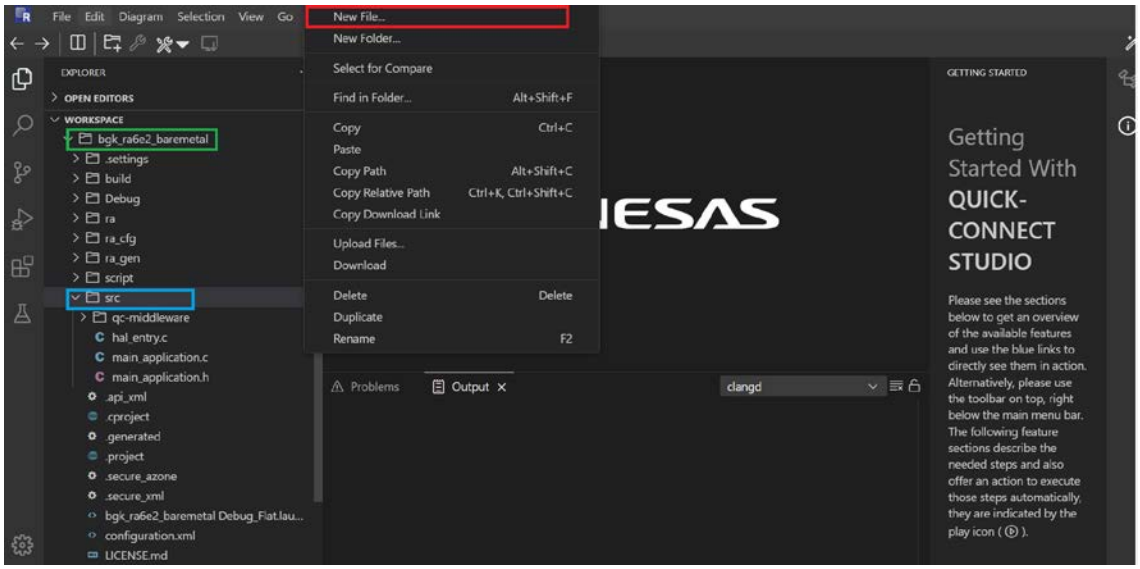
To organize and upload the new files, the user needs to create folders/files as required. For example, the **gpt_timer.c** file should be placed at the path: **bgk_ra6e2_baremetal/src/gpt_timer.c**

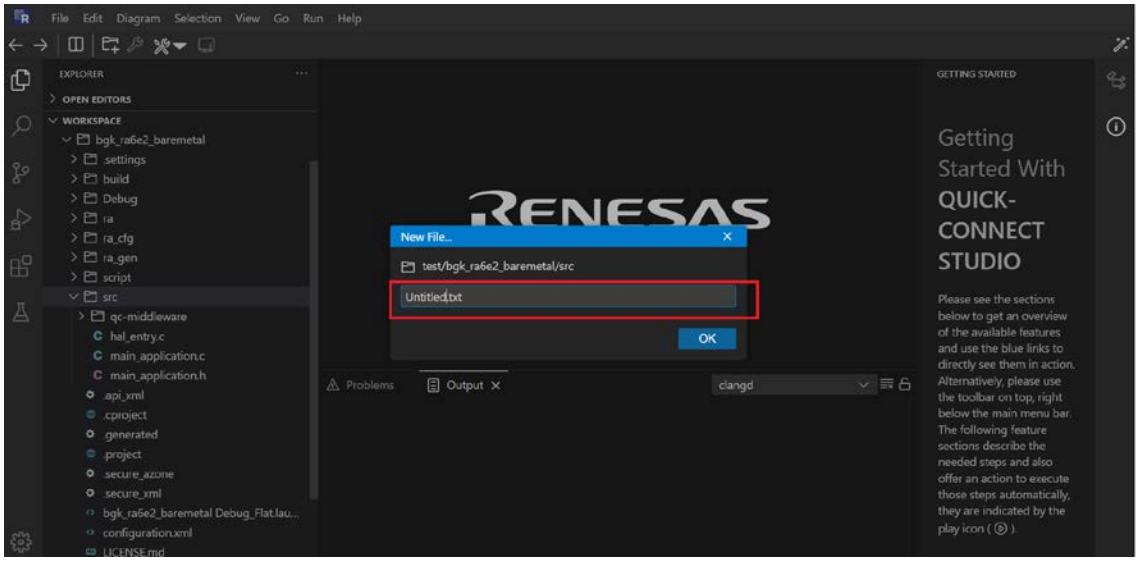
However, in the existing Blinky project, this folder structure does currently exist. Therefore, the user must create the **gpt_timer.c** file under the specified path.

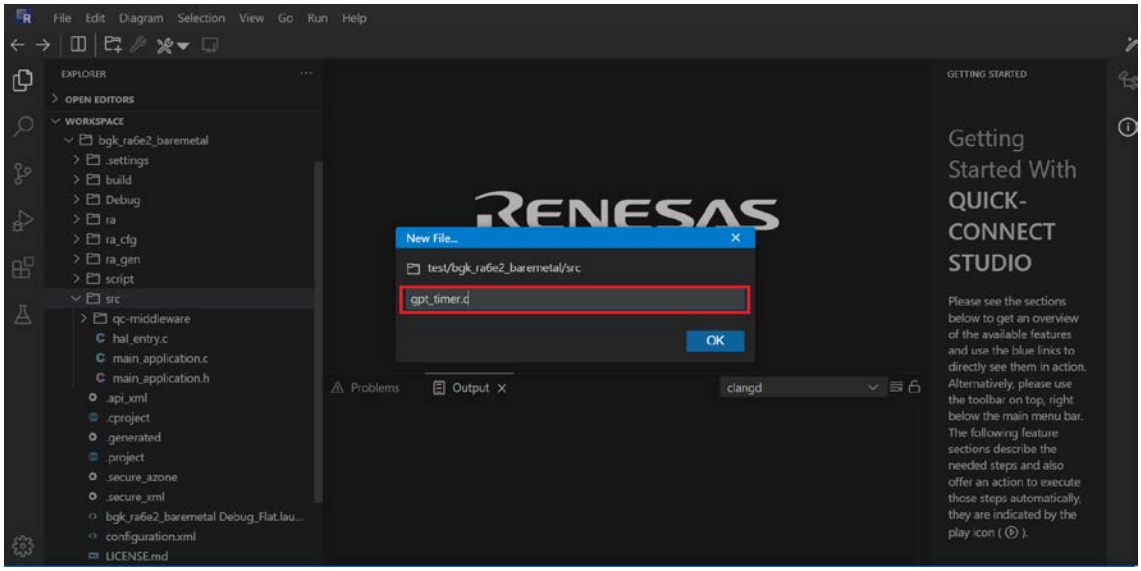
A **new folder/file** can be created by following these steps:

1. Click on the project folder (highlighted in the green box).

2. Navigate to the target path folder (highlighted in the blue box).
3. Right-click on the folder where the new folder needs to be created.
4. Select the **New Folder/New File** option from the context menu (highlighted in the red box in the image).
5. Edit the **folder/File Name** as required.
6. After creating the required file at specified path, a user can paste or edit the content in the file as required.





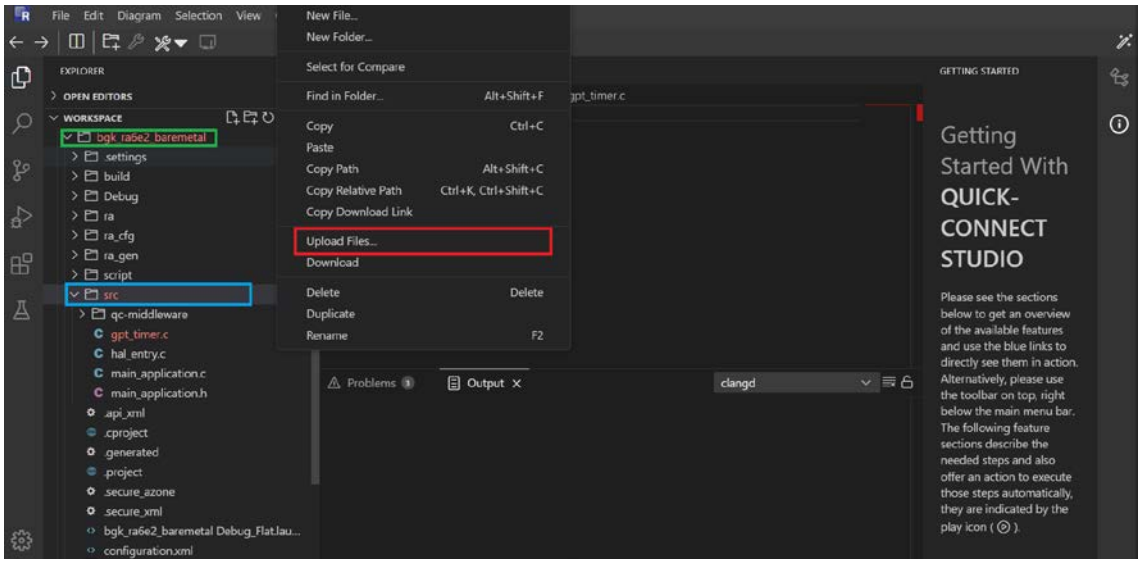


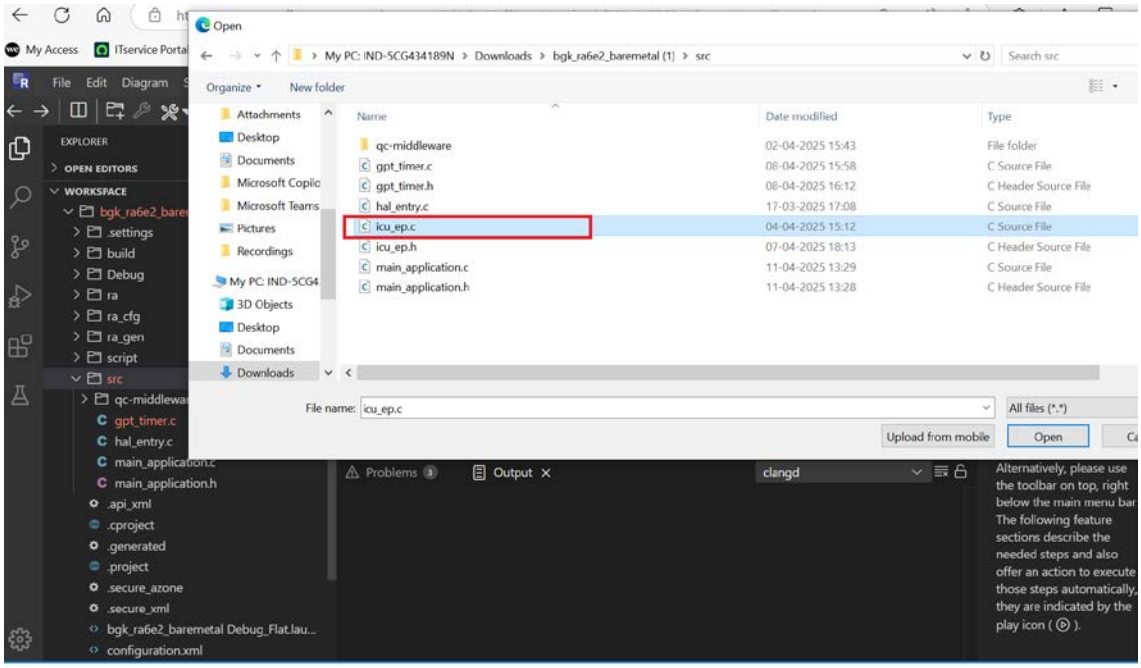
10.3.1.2 Upload the Required Files

To upload the required files to the **QCS project**, complete the following steps:

1. Open the QCS project in the browser.
2. Click on the generated project folder (highlighted with a green box in the image below).
3. Right-click on the folder where the files need to be uploaded (Blue box), select the **Upload Files** option (highlighted in red box in the image).

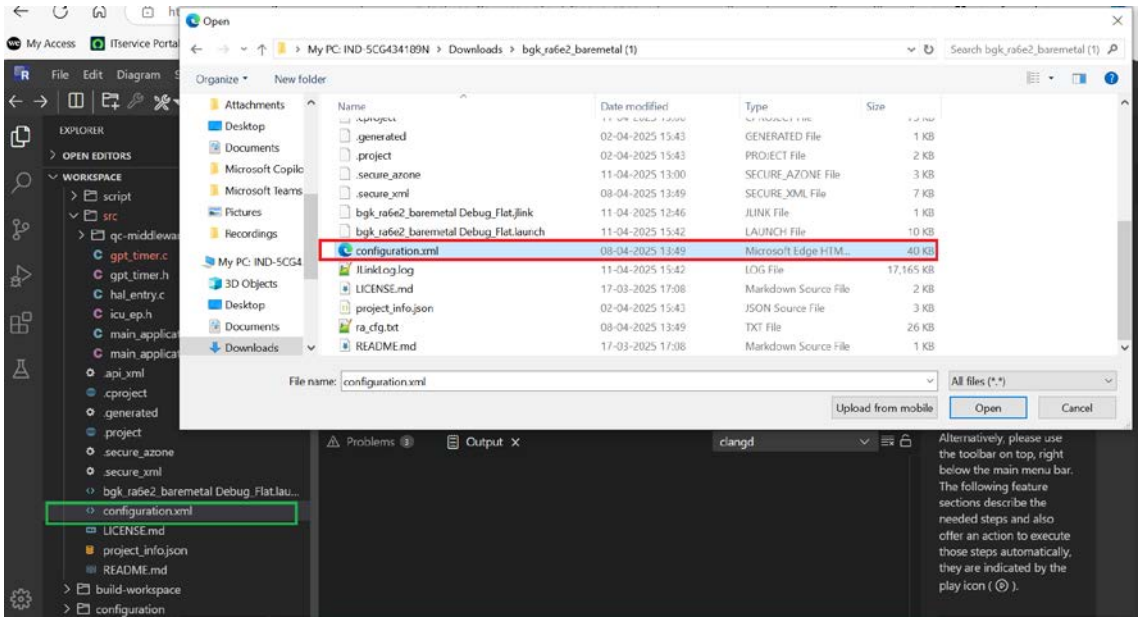
This opens a file browser window on the local PC, allowing for selecting and uploading the required files, as shown in the second image.

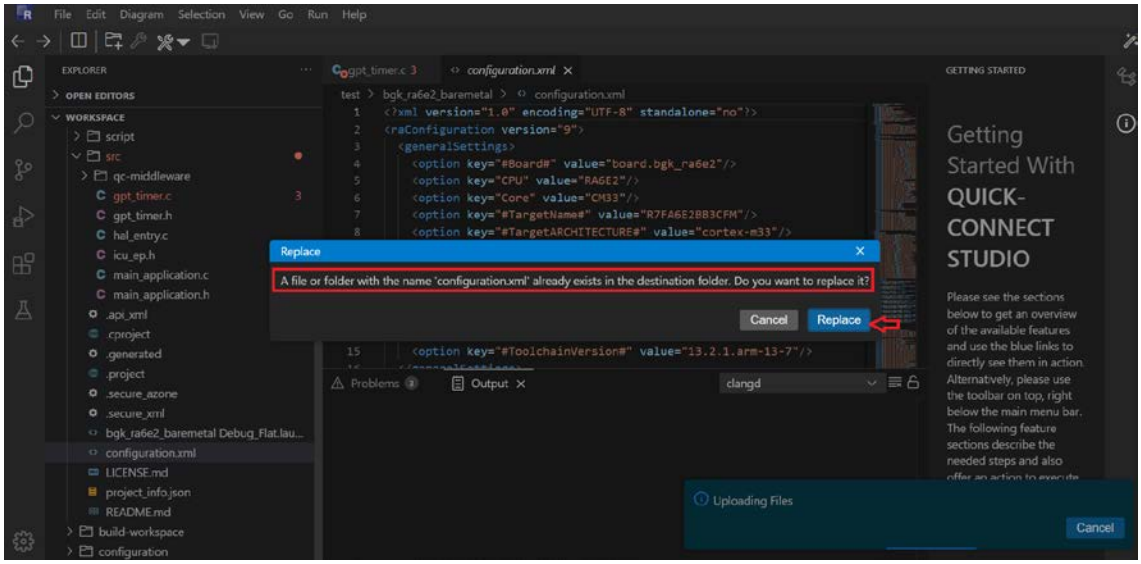




10.3.1.3 Replace Any Existing Files

Similar to how new files are uploaded, existing files can require updates.
While uploading an existing file, a pop-up window appears with the message, "Do you want to replace it?".
In this case, click on **Replace** to overwrite the old version with the updated one.





10.3.2 Customize Blinky Application

This section includes some optional customization users can implement on the Blinky Project.

10.3.2.1 Update the Blinky Application to Code the Desired LED to Blink

If the user wants to control which LED should be enabled in the code, they can do so by defining macros in the **main_application.h** file:

To enable or disable a specific LED, simply comment or uncomment the corresponding macro.

For example, to disable LED1, comment out the **ENABLE_LED1** macro. To enable it again, uncomment the line.

The updated **main_application.h** is available on GitHub. Additionally, the modifications applied to the generated **main_application.h** file for the Blink LEDs code can be identified by the **+** symbols in the following code blocks.

```
#ifndef __MAIN_A#PPLICATION_H
#define __MAIN_APPLICATION_H

#include "common_utils.h"
+ #define ENABLE_LED1   // enable or disable the code for LED1 using this macro

+ #define ENABLE_LED2     // enable or disable the code for LED2 using this macro
/* Function declaration */
void main_application(void);

#endif /* __MAIN_APPLICATION_H */
```

Code Block 1 main_application.h

After defining the macros, the user should update the **main_application.c** file accordingly.

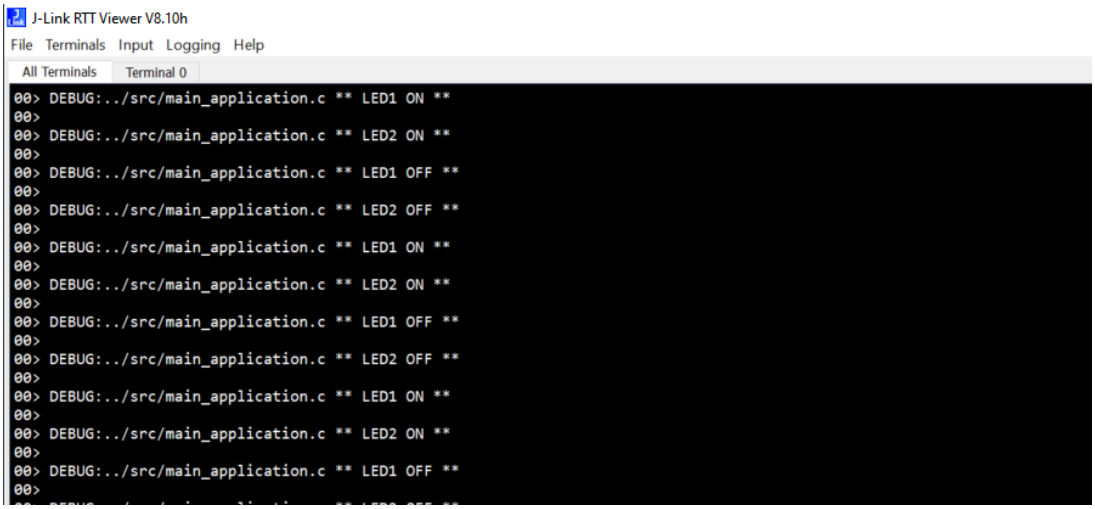
The updated **main_application.c** is available on GitHub. Additionally, the modifications applied to the generated **main_application.c** file for the Blink LEDs code can be identified by the **+** symbols in the following code blocks.

```
void main_application(void) {

// Start of autogenerated code
  while(true)
  {

+   #ifdef ENABLE_LED1
          utils_set_LED(LED1_LED, BSP_IO_LEVEL_HIGH);
          utils_delay_ms(250);
+         log_debug("** LED1 ON ** \r\n");
+   #endif
+   #ifdef ENABLE_LED2
          utils_set_LED(LED2_LED, BSP_IO_LEVEL_HIGH);
          utils_delay_ms(250);
+         log_debug("** LED2 ON ** \r\n");
+   #endif
+   #ifdef ENABLE_LED1
          utils_set_LED(LED1_LED, BSP_IO_LEVEL_LOW);
          utils_delay_ms(250);
+         log_debug("** LED1 OFF ** \r\n");
+   #endif
+   #ifdef ENABLE_LED2
          utils_set_LED(LED2_LED, BSP_IO_LEVEL_LOW);
          utils_delay_ms(250);
+         log_debug("** LED2 OFF ** \r\n");
+   #endif

  }
// End of autogenerated code

}
```

Code Block 2 main_application.h

Expected Output RTT Logs:

Check the following logs for the expected behavior and also observe LED toggle on Board.



10.3.2.2 Toggle LED using GPIO Button

If the user wants to control the on-board LEDs (that is LED1 and LED2) using the GPIO switch S1 (connected to pin **P304** on the **BGK-RA6E2** board), the first step is to enable the **External Interrupt (ICU) stack** in the code by editing the **configuration.xml** file.

Download the configuration.xml from this GitHub link and replace the existing configuration.xml from the project.

The required updated files such as **main_application.c**, **main_application.h**, **icu_ep.c**, and **icu_ep.h**—are available in the same GitHub repository referenced above. Additionally, the modifications applied to the generated Blink LEDs code can be identified by the **"+"** symbols in the code blocks provided below.

Files to be added:

To initialize the external interrupt stack (ICU)**,** middleware API support is required. The following are the files responsible for initializing the external interrupt stack. These functions are called in **main_application.c** to initialize the stack.

```
/src /icu_ep.c
/src /icu_ep.h
```

The following is the content of the file that should be added to the project.

```
/***********************************************************************************************
*******************************
 * File Name    : icu_ep.c
 * Description  : Contains function definition.

***********************************************************************************************
*******************************/
/***********************************************************************************************
*******************************
* Copyright (c) 2020 - 2024 Renesas Electronics Corporation and/or its affiliates
*
* SPDX-License-Identifier: BSD-3-Clause
***********************************************************************************************
*******************************/

#include "common_utils.h"
#include "icu_ep.h"
// Uncomment the desired debug level
#include "log_disabled.h"
//#include "log_error.h"
//#include "log_warning.h"
//#include "log_info.h"
//#include "log_debug.h"
/***********************************************************************************************
*****************************//**
 * @addtogroup icu_ep
 * @{

***********************************************************************************************
*******************************/



/***********************************************************************************************
*****************************//**
 * @addtogroup icu_ep
 * @{

***********************************************************************************************
*******************************/


/***********************************************************************************************
*****************************//**
 * @brief       This functions initializes ICU module.
 * @param[IN]   None
 * @retval      FSP_SUCCESS                 Upon successful open of ICU module
 * @retval      Any Other Error code apart from FSP_SUCCESS  Unsuccessful open

***********************************************************************************************
*******************************/
fsp_err_t icu_init(void)
{
    fsp_err_t err = FSP_SUCCESS;

    /* Open ICU module */
    err = R_ICU_ExternalIrqOpen(&g_external_irq_ctrl, &g_external_irq_cfg);
    /* Handle error */
    if (FSP_SUCCESS != err)
    {
        /* ICU Open failure message */
        log_error ("\r\n**R_ICU_ExternalIrqOpen API FAILED**\r\n");
    }
    return err;
}
```

```
/***********************************************************************************************
****************************//**
 * @brief       This function enables external interrupt for specified channel.
 * @param[IN]   None
 * @retval      FSP_SUCCESS                 Upon successful enable of ICU module
 * @retval      Any Other Error code apart from FSP_SUCCESS  Unsuccessful open

***********************************************************************************************
*****************************/
fsp_err_t icu_enable(void)
{
    fsp_err_t err = FSP_SUCCESS;

    /* Enable ICU module */
    err = R_ICU_ExternalIrqEnable(&g_external_irq_ctrl);
    /* Handle error */
    if (FSP_SUCCESS != err)
    {
        /* ICU Enable failure message */
        log_error ("\r\n**R_ICU_ExternalIrqEnable API FAILED**\r\n");
    }
    return err;
}


/***********************************************************************************************
****************************//**
 * @brief       This function closes opened ICU module before the project ends up in an
Error Trap.
 * @param[IN]   None
 * @retval      None

***********************************************************************************************
*****************************/
void icu_deinit(void)
{
    fsp_err_t err = FSP_SUCCESS;

    /* Close ICU module */
    err = R_ICU_ExternalIrqClose(&g_external_irq_ctrl);
    /* Handle error */
    if (FSP_SUCCESS != err)
    {
        /* ICU Close failure message */
        log_error("\r\n**R_ICU_ExternalIrqClose API FAILED**\r\n");
    }
}
```

Code Block 1 /src /icu_ep.c

```
/***********************************************************************************************
*******************************
 * File Name    : icu_ep.h
 * Description  : Contains Macros and function declarations.

***********************************************************************************************
*******************************/
/***********************************************************************************************
*******************************
* Copyright (c) 2020 - 2024 Renesas Electronics Corporation and/or its affiliates
*
* SPDX-License-Identifier: BSD-3-Clause
***********************************************************************************************
*******************************/
#ifndef ICU_EP_H_
#define ICU_EP_H_

#define USER_SW1_IRQ_NUMBER    (9)



/* Function declaration */
fsp_err_t icu_init(void);
fsp_err_t icu_enable(void);
void icu_deinit(void);

#endif /* ICU_EP_H_ */
```

Code Block 2 /src /icu_ep.h

The user can enable or disable control for each LED based on the macros defined in main_application.h. The process for enabling or disabling these macros has been explained in the previous section.

Files to be updated:

The modifications applied to the generated **main_application.c** file for the Blink LEDs code can be identified by the **+** symbols in the code blocks provided below. Additionally, the updated **main_application.c** can be downloaded from GitHub.

```
/*
* Copyright (c) 2020 - 2024 Renesas Electronics Corporation and/or its affiliates
*
* SPDX-License-Identifier:  BSD-3-Clause
*/
#include <stdio.h>
#include "common_utils.h"
#include "main_application.h"
+ #include "icu_ep.h"
// Uncomment the desired debug level
#include "log_disabled.h"
//#include "log_error.h"
//#include "log_warning.h"
//#include "log_info.h"
//#include "log_debug.h"
+ volatile bool g_sw_press = false;

void main_application(void) {

// Start of autogenerated code
+   fsp_err_t err = FSP_SUCCESS;
+   uint8_t led_state = 1;

+  /* Initialize External IRQ driver*/
+  log_debug("** ICU ICU Stack ** \r\n");
+  err = icu_init();
+  /* Handle error */
+  if(FSP_SUCCESS != err)
+  {
+    log_error("** ICU INIT FAILED ** \r\n");
+  }

+  /* Enable External IRQ driver*/
+  err = icu_enable();
+  /* Handle error */
+  if(FSP_SUCCESS != err)
+  {
+    log_error("** ICU ENABLE FAILED ** \r\n");
+    /* Close External IRQ module.*/
+    icu_deinit();
+  }




  while(true)
  {
+   if(g_sw_press == true)
+    {
+ #ifdef ENABLE_LED1
+      if(led_state)
+      {
+          utils_set_LED(LED1_LED, BSP_IO_LEVEL_HIGH);
+          log_debug("** LED1 ON ** \r\n");
+      }
+      else
+      {
+          utils_set_LED(LED1_LED, BSP_IO_LEVEL_LOW);
+          log_debug("** LED1 OFF ** \r\n");
+      }
+ #endif

+ #ifdef ENABLE_LED2
+      if(led_state)
+      {
+          utils_set_LED(LED2_LED, BSP_IO_LEVEL_HIGH);
```

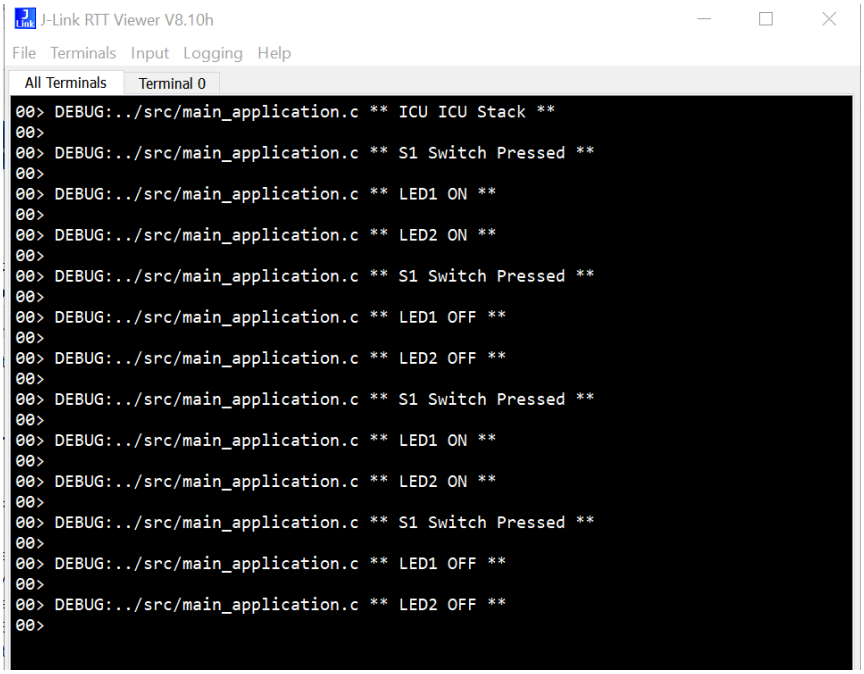
```
+           log_debug("** LED2 ON ** \r\n");
+       }
+       else
+       {
+           utils_set_LED(LED2_LED, BSP_IO_LEVEL_LOW);
+           log_debug("** LED2 OFF ** \r\n");
+       }
+ #endif
+       led_state= !led_state;
+       g_sw_press = false;
+    }
 }
// End of autogenerated code

}

// Start of autogenerated code
+ void user_button_callback( external_irq_callback_args_t * p_args)
+ {
+    if (p_args->channel == USER_SW1_IRQ_NUMBER)
+    {
+      g_sw_press = true;
+      log_debug("** S1 Switch Pressed ** \r\n");
+    }
+ }
// End of autogenerated code
```

Code Block 3 main_application.c

After all the changes are complete, compile and test the code.

Expected Output RTT Logs:

Check the following logs for the expected behavior and also observe LED toggle on board.



10.3.2.3 Control the LED using PWM Timer

If controlling the LED is required, enable the Timer PWM feature on the corresponding LED pins. The first step is to enable the Timer PWM Stack in the code by modifying the **configuration.xml** file.

In the updated file:

- Lines starting with **+** indicate new additions.
- Lines starting with # indicate modifications to the existing default configuration.
- Lines starting with **-** indicate deletion.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<raConfiguration version="9">
  <generalSettings>
    <option key="#Board#" value="board.bgk_ra6e2"/>
    <option key="CPU" value="RA6E2"/>
    <option key="Core" value="CM33"/>
    <option key="#TargetName#" value="R7FA6E2BB3CFM"/>
    <option key="#TargetARCHITECTURE#" value="cortex-m33"/>
    <option key="#DeviceCommand#" value="R7FA6E2BB"/>
    <option key="#RTOS#" value="_none"/>
    <option key="#pinconfiguration#" value="R7FA6E2BB3CFM.pincfg"/>
    <option key="#FSPVersion#" value="5.5.0"/>
    <option key="#ConfigurationFragments#" value="Renesas##BSP##Board##ra6e2_bgk##"/>
    <option key="#SELECTED_TOOLCHAIN#" value="gcc-arm-embedded"/>
    <option key="#ToolchainVersion#" value="13.2.1.arm-13-7"/>
  </generalSettings>
  <raBspConfiguration>
    <config id="config.bsp.ra6e2.R7FA6E2BB3CFM">
      <property id="config.bsp.part_number" value="config.bsp.part_number.value"/>
      <property id="config.bsp.rom_size_bytes" value="config.bsp.rom_size_bytes.value"/>
      <property id="config.bsp.rom_size_bytes_hidden" value="262144"/>
      <property id="config.bsp.ram_size_bytes" value="config.bsp.ram_size_bytes.value"/>
      <property id="config.bsp.data_flash_size_bytes"
value="config.bsp.data_flash_size_bytes.value"/>
      <property id="config.bsp.package_style" value="config.bsp.package_style.value"/>
      <property id="config.bsp.package_pins" value="config.bsp.package_pins.value"/>
      <property id="config.bsp.irq_count_hidden" value="96"/>
    </config>
    <config id="config.bsp.ra6e2">
      <property id="config.bsp.series" value="config.bsp.series.value"/>
    </config>
    <config id="config.bsp.ra6e2.fsp">
      <property id="config.bsp.fsp.inline_irq_functions"
value="config.bsp.common.inline_irq_functions.enabled"/>
      <property id="config.bsp.fsp.tz.exception_response"
value="config.bsp.fsp.tz.exception_response.nmi"/>
      <property id="config.bsp.fsp.tz.cmsis.bfhfnmins"
value="config.bsp.fsp.tz.cmsis.bfhfnmins.secure"/>
      <property id="config.bsp.fsp.tz.cmsis.sysresetreqs"
value="config.bsp.fsp.tz.cmsis.sysresetreqs.secure_only"/>
      <property id="config.bsp.fsp.tz.cmsis.s_priority_boost"
value="config.bsp.fsp.tz.cmsis.s_priority_boost.disabled"/>
      <property id="config.bsp.fsp.tz.csar" value="config.bsp.fsp.tz.csar.both"/>
      <property id="config.bsp.fsp.tz.rstsar" value="config.bsp.fsp.tz.rstsar.both"/>
      <property id="config.bsp.fsp.tz.bbfsar" value="config.bsp.fsp.tz.bbfsar.both"/>
      <property id="config.bsp.fsp.tz.sramsar.sramprcr"
value="config.bsp.fsp.tz.sramsar.sramprcr.both"/>
      <property id="config.bsp.fsp.tz.sramsar.sramecc"
value="config.bsp.fsp.tz.sramsar.sramecc.both"/>
      <property id="config.bsp.fsp.tz.stbramsar"
value="config.bsp.fsp.tz.stbramsar.both"/>
      <property id="config.bsp.fsp.tz.bussara" value="config.bsp.fsp.tz.bussara.both"/>
      <property id="config.bsp.fsp.tz.bussarb" value="config.bsp.fsp.tz.bussarb.both"/>
      <property id="config.bsp.fsp.tz.uninitialized_ns_application_fallback"
value="config.bsp.fsp.tz.uninitialized_ns_application_fallback.enabled"/>
      <property id="config.bsp.fsp.cache_line_size"
value="config.bsp.fsp.cache_line_size.32"/>
      <property id="config.bsp.fsp.OFS0.iwdt_start_mode"
value="config.bsp.fsp.OFS0.iwdt_start_mode.disabled"/>
      <property id="config.bsp.fsp.OFS0.iwdt_timeout"
value="config.bsp.fsp.OFS0.iwdt_timeout.2048"/>
      <property id="config.bsp.fsp.OFS0.iwdt_divisor"
value="config.bsp.fsp.OFS0.iwdt_divisor.128"/>
      <property id="config.bsp.fsp.OFS0.iwdt_window_end"
value="config.bsp.fsp.OFS0.iwdt_window_end.0"/>
```

```
      <property id="config.bsp.fsp.OFS0.iwdt_window_start"
value="config.bsp.fsp.OFS0.iwdt_window_start.100"/>
      <property id="config.bsp.fsp.OFS0.iwdt_reset_interrupt"
value="config.bsp.fsp.OFS0.iwdt_reset_interrupt.Reset"/>
      <property id="config.bsp.fsp.OFS0.iwdt_stop_control"
value="config.bsp.fsp.OFS0.iwdt_stop_control.stops"/>
      <property id="config.bsp.fsp.OFS0.wdt_start_mode"
value="config.bsp.fsp.OFS0.wdt_start_mode.register"/>
      <property id="config.bsp.fsp.OFS0.wdt_timeout"
value="config.bsp.fsp.OFS0.wdt_timeout.16384"/>
      <property id="config.bsp.fsp.OFS0.wdt_divisor"
value="config.bsp.fsp.OFS0.wdt_divisor.128"/>
      <property id="config.bsp.fsp.OFS0.wdt_window_end"
value="config.bsp.fsp.OFS0.wdt_window_end.0"/>
      <property id="config.bsp.fsp.OFS0.wdt_window_start"
value="config.bsp.fsp.OFS0.wdt_window_start.100"/>
      <property id="config.bsp.fsp.OFS0.wdt_reset_interrupt"
value="config.bsp.fsp.OFS0.wdt_reset_interrupt.Reset"/>
      <property id="config.bsp.fsp.OFS0.wdt_stop_control"
value="config.bsp.fsp.OFS0.wdt_stop_control.stops"/>
      <property id="config.bsp.fsp.OFS1.voltage_detection0.start"
value="config.bsp.fsp.OFS1.voltage_detection0.start.disabled"/>
      <property id="config.bsp.fsp.OFS1.voltage_detection0_level"
value="config.bsp.fsp.OFS1.voltage_detection0_level.280"/>
      <property id="config.bsp.fsp.OFS1.hoco_osc"
value="config.bsp.fsp.OFS1.hoco_osc.disabled"/>
      <property id="config.bsp.fsp.BPS.BPS0" value=""/>
      <property id="config.bsp.fsp.PBPS.PBPS0" value=""/>
      <property id="config.bsp.fsp.hoco_fll" value="config.bsp.fsp.hoco_fll.disabled"/>
      <property id="config.bsp.common.main_osc_wait"
value="config.bsp.common.main_osc_wait.wait_8163"/>
      <property id="config.bsp.fsp.mcu.adc.max_freq_hz" value="50000000"/>
      <property id="config.bsp.fsp.mcu.sci_uart.max_baud" value="16666666"/>
      <property id="config.bsp.fsp.mcu.adc.sample_and_hold" value="1"/>
      <property id="config.bsp.fsp.mcu.sci_spi.max_bitrate" value="25000000"/>
      <property id="config.bsp.fsp.mcu.spi.max_bitrate" value="50000000"/>
      <property id="config.bsp.fsp.mcu.iic_master.rate.rate_fastplus" value="1"/>
      <property id="config.bsp.fsp.mcu.canfd.num_channels" value="1"/>
      <property id="config.bsp.fsp.mcu.canfd.rx_fifos" value="2"/>
      <property id="config.bsp.fsp.mcu.canfd.buffer_ram" value="1216"/>
      <property id="config.bsp.fsp.mcu.canfd.afl_rules" value="32"/>
      <property id="config.bsp.fsp.mcu.canfd.afl_rules_each_chnl" value="32"/>
      <property id="config.bsp.fsp.mcu.canfd.max_data_rate_hz" value="5"/>
      <property id="config.bsp.fsp.mcu.sci_uart.cstpen_channels" value="0x0201"/>
      <property id="config.bsp.fsp.mcu.gpt.pin_count_source_channels" value="0xFFFF"/>
      <property id="config.bsp.common.id_mode"
value="config.bsp.common.id_mode.unlocked"/>
      <property id="config.bsp.common.id_code"
value="FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF"/>
      <property id="config.bsp.common.id1" value=""/>
      <property id="config.bsp.common.id2" value=""/>
      <property id="config.bsp.common.id3" value=""/>
      <property id="config.bsp.common.id4" value=""/>
      <property id="config.bsp.common.id_fixed" value=""/>
      <property id="config.bsp.fsp.mcu.adc_dmac.samples_per_channel" value="32767"/>
    </config>
    <config id="config.bsp.ra">
      <property id="config.bsp.common.main" value="1024"/>
      <property id="config.bsp.common.heap" value="1024"/>
      <property id="config.bsp.common.vcc" value="3300"/>
      <property id="config.bsp.common.checking"
value="config.bsp.common.checking.disabled"/>
      <property id="config.bsp.common.assert" value="config.bsp.common.assert.none"/>
      <property id="config.bsp.common.error_log"
value="config.bsp.common.error_log.none"/>
```

```
      <property id="config.bsp.common.soft_reset"
value="config.bsp.common.soft_reset.disabled"/>
      <property id="config.bsp.common.main_osc_populated"
value="config.bsp.common.main_osc_populated.disabled"/>
      <property id="config.bsp.common.pfs_protect"
value="config.bsp.common.pfs_protect.enabled"/>
      <property id="config.bsp.common.c_runtime_init"
value="config.bsp.common.c_runtime_init.enabled"/>
      <property id="config.bsp.common.early_init"
value="config.bsp.common.early_init.disabled"/>
      <property id="config.bsp.common.main_osc_clock_source"
value="config.bsp.common.main_osc_clock_source.crystal"/>
      <property id="config.bsp.common.subclock_populated"
value="config.bsp.common.subclock_populated.enabled"/>
      <property id="config.bsp.common.subclock_drive"
value="config.bsp.common.subclock_drive.standard"/>
      <property id="config.bsp.common.subclock_stabilization_ms" value="1000"/>
    </config>
  </raBspConfiguration>
  <raClockConfiguration>
    <node id="board.clock.xtal.freq" mul="20000000" option="_edit"/>
    <node id="board.clock.hoco.freq" option="board.clock.hoco.freq.20m"/>
    <node id="board.clock.loco.freq" option="board.clock.loco.freq.32768"/>
    <node id="board.clock.moco.freq" option="board.clock.moco.freq.8m"/>
    <node id="board.clock.subclk.freq" option="board.clock.subclk.freq.32768"/>
    <node id="board.clock.pll.source" option="board.clock.pll.source.hoco"/>
    <node id="board.clock.pll.div" option="board.clock.pll.div.1"/>
    <node id="board.clock.pll.mul" option="board.clock.pll.mul.100"/>
    <node id="board.clock.pll.display" option="board.clock.pll.display.value"/>
    <node id="board.clock.clock.source" option="board.clock.clock.source.pll"/>
    <node id="board.clock.clkout.source" option="board.clock.clkout.source.disabled"/>
    <node id="board.clock.uclk.source" option="board.clock.uclk.source.disabled"/>
    <node id="board.clock.canfdclk.source"
option="board.clock.canfdclk.source.disabled"/>
    <node id="board.clock.cecclk.source" option="board.clock.cecclk.source.disabled"/>
    <node id="board.clock.i3cclk.source" option="board.clock.i3cclk.source.disabled"/>
    <node id="board.clock.iclk.div" option="board.clock.iclk.div.1"/>
    <node id="board.clock.pclka.div" option="board.clock.pclka.div.2"/>
    <node id="board.clock.pclkb.div" option="board.clock.pclkb.div.4"/>
    <node id="board.clock.pclkc.div" option="board.clock.pclkc.div.4"/>
    <node id="board.clock.pclkd.div" option="board.clock.pclkd.div.2"/>
    <node id="board.clock.fclk.div" option="board.clock.fclk.div.4"/>
    <node id="board.clock.clkout.div" option="board.clock.clkout.div.1"/>
    <node id="board.clock.uclk.div" option="board.clock.uclk.div.5"/>
    <node id="board.clock.canfdclk.div" option="board.clock.canfdclk.div.6"/>
    <node id="board.clock.cecclk.div" option="board.clock.cecclk.div.1"/>
    <node id="board.clock.i3cclk.div" option="board.clock.i3cclk.div.1"/>
    <node id="board.clock.iclk.display" option="board.clock.iclk.display.value"/>
    <node id="board.clock.pclka.display" option="board.clock.pclka.display.value"/>
    <node id="board.clock.pclkb.display" option="board.clock.pclkb.display.value"/>
    <node id="board.clock.pclkc.display" option="board.clock.pclkc.display.value"/>
    <node id="board.clock.pclkd.display" option="board.clock.pclkd.display.value"/>
    <node id="board.clock.fclk.display" option="board.clock.fclk.display.value"/>
    <node id="board.clock.clkout.display" option="board.clock.clkout.display.value"/>
    <node id="board.clock.uclk.display" option="board.clock.uclk.display.value"/>
    <node id="board.clock.canfdclk.display"
option="board.clock.canfdclk.display.value"/>
    <node id="board.clock.cecclk.display" option="board.clock.cecclk.display.value"/>
    <node id="board.clock.i3cclk.display" option="board.clock.i3cclk.display.value"/>
  </raClockConfiguration>
  <raComponentSelection>
    <component apiversion="" class="Common" condition="" group="all"
subgroup="fsp_common" variant="" vendor="Renesas" version="5.5.0">
      <description>Board Support Package Common Files</description>
      <originalPack>Renesas.RA.5.5.0.pack</originalPack>
    </component>
```

```
+     <component apiversion="" class="HAL Drivers" condition="" group="all"
subgroup="r_gpt" variant="" vendor="Renesas" version="5.5.0">
+      <description>General PWM Timer</description>
+      <originalPack>Renesas.RA.5.5.0.pack</originalPack>
+    </component>
    <component apiversion="" class="HAL Drivers" condition="" group="all"
subgroup="r_ioport" variant="" vendor="Renesas" version="5.5.0">
      <description>I/O Port</description>
      <originalPack>Renesas.RA.5.5.0.pack</originalPack>
    </component>
    <component apiversion="" class="CMSIS" condition="" group="CMSIS5" subgroup="CoreM"
variant="" vendor="Arm" version="6.1.0+fsp.5.5.0">
      <description>Arm CMSIS Version 6 - Core (M)</description>
      <originalPack>Arm.CMSIS6.6.1.0+fsp.5.5.0.pack</originalPack>
    </component>
    <component apiversion="" class="BSP" condition="" group="Board" subgroup="ra6e2_bgk"
variant="" vendor="Renesas" version="5.5.0">
      <description>RA6E2-BGK Board Support Files</description>
      <originalPack>Renesas.RA_board_ra6e2_bgk.5.5.0.pack</originalPack>
    </component>
    <component apiversion="" class="BSP" condition="" group="ra6e2" subgroup="device"
variant="R7FA6E2BB3CFM" vendor="Renesas" version="5.5.0">
      <description>Board support package for R7FA6E2BB3CFM</description>
      <originalPack>Renesas.RA_mcu_ra6e2.5.5.0.pack</originalPack>
    </component>
    <component apiversion="" class="BSP" condition="" group="ra6e2" subgroup="device"
variant="" vendor="Renesas" version="5.5.0">
      <description>Board support package for RA6E2</description>
      <originalPack>Renesas.RA_mcu_ra6e2.5.5.0.pack</originalPack>
    </component>
    <component apiversion="" class="BSP" condition="" group="ra6e2" subgroup="fsp"
variant="" vendor="Renesas" version="5.5.0">
      <description>Board support package for RA6E2 - FSP Data</description>
      <originalPack>Renesas.RA_mcu_ra6e2.5.5.0.pack</originalPack>
    </component>
    <component apiversion="" class="BSP" condition="" group="ra6e2" subgroup="events"
variant="" vendor="Renesas" version="5.5.0">
      <description>Board support package for RA6E2 - Events</description>
      <originalPack>Renesas.RA_mcu_ra6e2.5.5.0.pack</originalPack>
    </component>
  </raComponentSelection>
  <raElcConfiguration/>
  <raIcuConfiguration/>
  <raModuleConfiguration>
    <module id="module.driver.ioport_on_ioport.0">
      <property id="module.driver.ioport.name" value="g_ioport"/>
      <property id="module.driver.ioport.elc_trigger_ioport1" value="_disabled"/>
      <property id="module.driver.ioport.elc_trigger_ioport2" value="_disabled"/>
      <property id="module.driver.ioport.elc_trigger_ioport3" value="_disabled"/>
      <property id="module.driver.ioport.elc_trigger_ioport4" value="_disabled"/>
      <property id="module.driver.ioport.pincfg" value="g_bsp_pin_cfg"/>
    </module>
 <module id="module.driver.timer_on_gpt.1083164079">
      <property id="module.driver.timer.name" value="g_timer_pwm_led1"/>
      <property id="module.driver.timer.channel" value="1"/>
      <property id="module.driver.timer.mode"
value="module.driver.timer.mode.mode_pwm"/>
      <property id="module.driver.timer.period" value="1"/>
      <property id="module.driver.timer.compare_match.a.status"
value="module.driver.timer.compare_match.a.status.disabled"/>
      <property id="module.driver.timer.compare_match.a.value" value="0"/>
      <property id="module.driver.timer.compare_match.b.status"
value="module.driver.timer.compare_match.b.status.disabled"/>
      <property id="module.driver.timer.compare_match.b.value" value="0"/>
      <property id="module.driver.timer.unit"
value="module.driver.timer.unit.unit_period_sec"/>
```

```
      <property id="module.driver.timer.gtior.gtioa.initial_output_level"
value="module.driver.timer.gtior.gtioa.initial_output_level.low"/>
      <property id="module.driver.timer.gtior.gtioa.cycle_end_output_level"
value="module.driver.timer.gtior.gtioa.cycle_end_output_level.retain"/>
      <property id="module.driver.timer.gtior.gtioa.compare_match_output_level"
value="module.driver.timer.gtior.gtioa.compare_match_output_level.retain"/>
      <property id="module.driver.timer.gtior.gtioa.count_stop_retain"
value="module.driver.timer.gtior.gtioa.count_stop_retain.disabled"/>
      <property id="module.driver.timer.gtior.gtiob.initial_output_level"
value="module.driver.timer.gtior.gtiob.initial_output_level.low"/>
      <property id="module.driver.timer.gtior.gtiob.cycle_end_output_level"
value="module.driver.timer.gtior.gtiob.cycle_end_output_level.retain"/>
      <property id="module.driver.timer.gtior.gtiob.compare_match_output_level"
value="module.driver.timer.gtior.gtiob.compare_match_output_level.retain"/>
      <property id="module.driver.timer.gtior.gtiob.count_stop_retain"
value="module.driver.timer.gtior.gtiob.count_stop_retain.disabled"/>
      <property id="module.driver.timer.gtior.custom_waveform_enable"
value="module.driver.timer.gtior.custom_waveform_enable.disabled"/>
      <property id="module.driver.timer.duty_cycle" value="50"/>
      <property id="module.driver.timer.gtioca_output_enabled"
value="module.driver.timer.gtioca_output_enabled.false"/>
      <property id="module.driver.timer.gtioca_stop_level"
value="module.driver.timer.gtioca_stop_level.pin_level_low"/>
      <property id="module.driver.timer.gtiocb_output_enabled"
value="module.driver.timer.gtiocb_output_enabled.true"/>
      <property id="module.driver.timer.gtiocb_stop_level"
value="module.driver.timer.gtiocb_stop_level.pin_level_low"/>
      <property id="module.driver.timer.count_up_source" value=""/>
      <property id="module.driver.timer.count_down_source" value=""/>
      <property id="module.driver.timer.start_source" value=""/>
      <property id="module.driver.timer.stop_source" value=""/>
      <property id="module.driver.timer.clear_source" value=""/>
      <property id="module.driver.timer.capture_a_source" value=""/>
      <property id="module.driver.timer.capture_b_source" value=""/>
      <property id="module.driver.timer.gtioca_filter"
value="module.driver.timer.gtioc_filter.gtioc_filter_none"/>
      <property id="module.driver.timer.gtiocb_filter"
value="module.driver.timer.gtioc_filter.gtioc_filter_none"/>
      <property id="module.driver.timer.p_callback" value="NULL"/>
      <property id="module.driver.timer.ipl" value="_disabled"/>
      <property id="module.driver.timer.capture_a_ipl" value="_disabled"/>
      <property id="module.driver.timer.capture_b_ipl" value="_disabled"/>
      <property id="module.driver.timer.trough_ipl" value="_disabled"/>
      <property id="module.driver.timer.extra"
value="module.driver.timer.extra.disabled"/>
      <property id="module.driver.timer.poeg_link"
value="module.driver.timer.poeg_link.poeg_link_poeg0"/>
      <property id="module.driver.timer.output_disable" value=""/>
      <property id="module.driver.timer.adc_trigger" value=""/>
      <property id="module.driver.timer.adc_a_compare_match" value="0"/>
      <property id="module.driver.timer.adc_b_compare_match" value="0"/>
      <property id="module.driver.timer.dead_time_count_up" value="0"/>
      <property id="module.driver.timer.dead_time_count_down" value="0"/>
      <property id="module.driver.timer.interrupt_skip.source"
value="module.driver.timer.interrupt_skip.source.none"/>
      <property id="module.driver.timer.interrupt_skip.count"
value="module.driver.timer.interrupt_skip.count.count_0"/>
      <property id="module.driver.timer.interrupt_skip.adc"
value="module.driver.timer.interrupt_skip.skip_sources.interrupt_skip.adc.none"/>
      <property id="module.driver.timer.gtioca_disable_setting"
value="module.driver.timer.gtioca_disable_setting.gtioc_disable_prohibited"/>
      <property id="module.driver.timer.gtiocb_disable_setting"
value="module.driver.timer.gtiocb_disable_setting.gtioc_disable_prohibited"/>
    </module>
+    <module id="module.driver.timer_on_gpt.230584519">
+      <property id="module.driver.timer.name" value="g_timer_pwm_led2"/>
```

```
+      <property id="module.driver.timer.channel" value="3"/>
+      <property id="module.driver.timer.mode"
value="module.driver.timer.mode.mode_pwm"/>
+      <property id="module.driver.timer.period" value="1"/>
+      <property id="module.driver.timer.compare_match.a.status"
value="module.driver.timer.compare_match.a.status.disabled"/>
+      <property id="module.driver.timer.compare_match.a.value" value="0"/>
+      <property id="module.driver.timer.compare_match.b.status"
value="module.driver.timer.compare_match.b.status.disabled"/>
+      <property id="module.driver.timer.compare_match.b.value" value="0"/>
+      <property id="module.driver.timer.unit"
value="module.driver.timer.unit.unit_period_sec"/>
+      <property id="module.driver.timer.gtior.gtioa.initial_output_level"
value="module.driver.timer.gtior.gtioa.initial_output_level.low"/>
+      <property id="module.driver.timer.gtior.gtioa.cycle_end_output_level"
value="module.driver.timer.gtior.gtioa.cycle_end_output_level.retain"/>
+      <property id="module.driver.timer.gtior.gtioa.compare_match_output_level"
value="module.driver.timer.gtior.gtioa.compare_match_output_level.retain"/>
+      <property id="module.driver.timer.gtior.gtioa.count_stop_retain"
value="module.driver.timer.gtior.gtioa.count_stop_retain.disabled"/>
+      <property id="module.driver.timer.gtior.gtiob.initial_output_level"
value="module.driver.timer.gtior.gtiob.initial_output_level.low"/>
+      <property id="module.driver.timer.gtior.gtiob.cycle_end_output_level"
value="module.driver.timer.gtior.gtiob.cycle_end_output_level.retain"/>
+      <property id="module.driver.timer.gtior.gtiob.compare_match_output_level"
value="module.driver.timer.gtior.gtiob.compare_match_output_level.retain"/>
+      <property id="module.driver.timer.gtior.gtiob.count_stop_retain"
value="module.driver.timer.gtior.gtiob.count_stop_retain.disabled"/>
+      <property id="module.driver.timer.gtior.custom_waveform_enable"
value="module.driver.timer.gtior.custom_waveform_enable.disabled"/>
+      <property id="module.driver.timer.duty_cycle" value="50"/>
+      <property id="module.driver.timer.gtioca_output_enabled"
value="module.driver.timer.gtioca_output_enabled.false"/>
+      <property id="module.driver.timer.gtioca_stop_level"
value="module.driver.timer.gtioca_stop_level.pin_level_low"/>
+      <property id="module.driver.timer.gtiocb_output_enabled"
value="module.driver.timer.gtiocb_output_enabled.true"/>
+      <property id="module.driver.timer.gtiocb_stop_level"
value="module.driver.timer.gtiocb_stop_level.pin_level_low"/>
+      <property id="module.driver.timer.count_up_source" value=""/>
+      <property id="module.driver.timer.count_down_source" value=""/>
+      <property id="module.driver.timer.start_source" value=""/>
+      <property id="module.driver.timer.stop_source" value=""/>
+      <property id="module.driver.timer.clear_source" value=""/>
+      <property id="module.driver.timer.capture_a_source" value=""/>
+      <property id="module.driver.timer.capture_b_source" value=""/>
+      <property id="module.driver.timer.gtioca_filter"
value="module.driver.timer.gtioc_filter.gtioc_filter_none"/>
+      <property id="module.driver.timer.gtiocb_filter"
value="module.driver.timer.gtioc_filter.gtioc_filter_none"/>
+      <property id="module.driver.timer.p_callback" value="NULL"/>
+      <property id="module.driver.timer.ipl" value="_disabled"/>
+      <property id="module.driver.timer.capture_a_ipl" value="_disabled"/>
+      <property id="module.driver.timer.capture_b_ipl" value="_disabled"/>
+      <property id="module.driver.timer.trough_ipl" value="_disabled"/>
+      <property id="module.driver.timer.extra"
value="module.driver.timer.extra.disabled"/>
+      <property id="module.driver.timer.poeg_link"
value="module.driver.timer.poeg_link.poeg_link_poeg0"/>
+      <property id="module.driver.timer.output_disable" value=""/>
+      <property id="module.driver.timer.adc_trigger" value=""/>
+      <property id="module.driver.timer.adc_a_compare_match" value="0"/>
+      <property id="module.driver.timer.adc_b_compare_match" value="0"/>
+      <property id="module.driver.timer.dead_time_count_up" value="0"/>
+      <property id="module.driver.timer.dead_time_count_down" value="0"/>
```

```
+      <property id="module.driver.timer.interrupt_skip.source"
value="module.driver.timer.interrupt_skip.source.none"/>
+      <property id="module.driver.timer.interrupt_skip.count"
value="module.driver.timer.interrupt_skip.count.count_0"/>
+      <property id="module.driver.timer.interrupt_skip.adc"
value="module.driver.timer.interrupt_skip.skip_sources.interrupt_skip.adc.none"/>
+      <property id="module.driver.timer.gtioca_disable_setting"
value="module.driver.timer.gtioca_disable_setting.gtioc_disable_prohibited"/>
+      <property id="module.driver.timer.gtiocb_disable_setting"
value="module.driver.timer.gtiocb_disable_setting.gtioc_disable_prohibited"/>
+    </module>
    <context id="_hal.0">
      <stack module="module.driver.ioport_on_ioport.0"/>
+      <stack module="module.driver.timer_on_gpt.1083164079"/>
+      <stack module="module.driver.timer_on_gpt.230584519"/>
    </context>
+    <config id="config.driver.gpt">
+      <property id="config.driver.gpt.param_checking_enable"
value="config.driver.gpt.param_checking_enable.bsp"/>
+      <property id="config.driver.gpt.output_support_enable"
value="config.driver.gpt.output_support_enable.enabled"/>
+      <property id="config.driver.gpt.write_protect_enable"
value="config.driver.gpt.write_protect_enable.disabled"/>
+    </config>
    <config id="config.driver.ioport">
      <property id="config.driver.ioport.checking"
value="config.driver.ioport.checking.system"/>
    </config>
  </raModuleConfiguration>
  <raPinConfiguration>
    <symbolicName propertyId="p000.symbolic_name" value="ARDUINO_A0"/>
    <symbolicName propertyId="p001.symbolic_name" value="ARDUINO_A1"/>
    <symbolicName propertyId="p002.symbolic_name" value="ARDUINO_A2"/>
    <symbolicName propertyId="p003.symbolic_name" value="ARDUINO_A4"/>
    <symbolicName propertyId="p004.symbolic_name" value="ARDUINO_A3"/>
    <symbolicName propertyId="p005.symbolic_name" value="PMOD1_RESET"/>
    <symbolicName propertyId="p006.symbolic_name" value="ARDUINO_D8"/>
    <symbolicName propertyId="p008.symbolic_name" value="ARDUINO_D7"/>
    <symbolicName propertyId="p013.symbolic_name" value="ARDUINO_A5"/>
    <symbolicName propertyId="p014.symbolic_name" value="PMOD2_GPIO1"/>
    <symbolicName propertyId="p015.symbolic_name" value="PMOD2_GPIO2"/>
    <symbolicName propertyId="p100.symbolic_name" value="PMOD2_SCL0_ARDUINO_SCL"/>
    <symbolicName propertyId="p101.symbolic_name" value="PMOD2_SDA0_ARDUINO_SDA"/>
    <symbolicName propertyId="p104.symbolic_name" value="LED1"/>
    <symbolicName propertyId="p105.symbolic_name" value="ARDUINO_D2"/>
    <symbolicName propertyId="p106.symbolic_name" value="PMOD1_GPIO1"/>
    <symbolicName propertyId="p107.symbolic_name" value="PMOD1_GPIO2"/>
    <symbolicName propertyId="p108.symbolic_name" value="DEBUG_SWDIO"/>
    <symbolicName propertyId="p109.symbolic_name" value="PMOD1_TX_ARDUINO_D11"/>
    <symbolicName propertyId="p110.symbolic_name" value="PMOD1_RX_ARDUINO_D12"/>
    <symbolicName propertyId="p111.symbolic_name" value="PMOD1_RSPCK_ARDUINO_D13"/>
    <symbolicName propertyId="p112.symbolic_name" value="LED2"/>
    <symbolicName propertyId="p113.symbolic_name" value="ARDUINO_D6"/>
    <symbolicName propertyId="p205.symbolic_name" value="PMOD2_IRQ"/>
    <symbolicName propertyId="p206.symbolic_name" value="PMOD2_SSIDATA"/>
    <symbolicName propertyId="p207.symbolic_name" value="PMOD2_SSICLK"/>
    <symbolicName propertyId="p208.symbolic_name" value="ARDUINO_RESET"/>
    <symbolicName propertyId="p300.symbolic_name" value="DEBUG_SWDCLK"/>
    <symbolicName propertyId="p301.symbolic_name" value="ARDUINO_D10"/>
    <symbolicName propertyId="p402.symbolic_name" value="PMOD1_IRQ"/>
    <symbolicName propertyId="p403.symbolic_name" value="ARDUINO_D9"/>
    <symbolicName propertyId="p407.symbolic_name" value="PMOD2_SSIBCK"/>
    <symbolicName propertyId="p408.symbolic_name" value="ARDUINO_D3"/>
    <symbolicName propertyId="p409.symbolic_name" value="ARDUINO_D5"/>
    <symbolicName propertyId="p410.symbolic_name" value="PMOD2_MISO_ARDUINO_D0"/>
```

```
    <symbolicName propertyId="p411.symbolic_name"
value="PMOD2_SENSOR_RESET_ARDUINO_D1"/>
    <symbolicName propertyId="p500.symbolic_name" value="ARDUINO_D4"/>
    <pincfg active="true" name="BGK_RA6E2.pincfg" selected="true"
symbol="g_bsp_pin_cfg">
      <configSetting altId="adc0.an000.p000" configurationId="adc0.an000"/>
      <configSetting altId="adc0.an001.p001" configurationId="adc0.an001"/>
      <configSetting altId="adc0.an002.p002" configurationId="adc0.an002"/>
      <configSetting altId="adc0.an004.p004" configurationId="adc0.an004"/>
      <configSetting altId="adc0.an007.p003" configurationId="adc0.an007"/>
      <configSetting altId="adc0.an011.p013" configurationId="adc0.an011"/>
      <configSetting altId="adc0.mode.custom.free" configurationId="adc0.mode"/>
+      <configSetting altId="gpt1.gtioc1b.p104" configurationId="gpt1.gtioc1b"/>
+      <configSetting altId="gpt1.mode.gtiocaorgtiocb.free"
configurationId="gpt1.mode"/>
+      <configSetting altId="gpt3.gtioc3b.p112" configurationId="gpt3.gtioc3b"/>
+      <configSetting altId="gpt3.mode.gtiocaorgtiocb.free"
configurationId="gpt3.mode"/>
      <configSetting altId="jtag_fslash_swd.mode.swd.free"
configurationId="jtag_fslash_swd.mode"/>
      <configSetting altId="jtag_fslash_swd.swclk.p300"
configurationId="jtag_fslash_swd.swclk"/>
      <configSetting altId="jtag_fslash_swd.swdio.p108"
configurationId="jtag_fslash_swd.swdio"/>
      <configSetting altId="p000.adc0.an000" configurationId="p000"/>
      <configSetting altId="p000.gpio_mode.gpio_mode_an"
configurationId="p000.gpio_mode"/>
      <configSetting altId="p001.adc0.an001" configurationId="p001"/>
      <configSetting altId="p001.gpio_mode.gpio_mode_an"
configurationId="p001.gpio_mode"/>
      <configSetting altId="p002.adc0.an002" configurationId="p002"/>
      <configSetting altId="p002.gpio_mode.gpio_mode_an"
configurationId="p002.gpio_mode"/>
      <configSetting altId="p003.adc0.an007" configurationId="p003"/>
      <configSetting altId="p003.gpio_mode.gpio_mode_an"
configurationId="p003.gpio_mode"/>
      <configSetting altId="p004.adc0.an004" configurationId="p004"/>
      <configSetting altId="p004.gpio_mode.gpio_mode_an"
configurationId="p004.gpio_mode"/>
      <configSetting altId="p005.output.high" configurationId="p005"/>
      <configSetting altId="p005.gpio_mode.gpio_mode_out.high"
configurationId="p005.gpio_mode"/>
      <configSetting altId="p006.input" configurationId="p006"/>
      <configSetting altId="p006.gpio_mode.gpio_mode_in"
configurationId="p006.gpio_mode"/>
      <configSetting altId="p008.input" configurationId="p008"/>
      <configSetting altId="p008.gpio_mode.gpio_mode_in"
configurationId="p008.gpio_mode"/>
      <configSetting altId="p013.adc0.an011" configurationId="p013"/>
      <configSetting altId="p013.gpio_mode.gpio_mode_an"
configurationId="p013.gpio_mode"/>
      <configSetting altId="p014.input" configurationId="p014"/>
      <configSetting altId="p014.gpio_mode.gpio_mode_in"
configurationId="p014.gpio_mode"/>
      <configSetting altId="p015.input" configurationId="p015"/>
      <configSetting altId="p015.gpio_mode.gpio_mode_in"
configurationId="p015.gpio_mode"/>
      <configSetting altId="p100.sci0.rxd0" configurationId="p100"/>
      <configSetting altId="p100.gpio_mode.gpio_mode_peripheral"
configurationId="p100.gpio_mode"/>
      <configSetting altId="p100.gpio_otype.gpio_otype_n_ch_od"
configurationId="p100.gpio_otype"/>
      <configSetting altId="p101.sci0.txd0" configurationId="p101"/>
      <configSetting altId="p101.gpio_mode.gpio_mode_peripheral"
configurationId="p101.gpio_mode"/>
```

```
      <configSetting altId="p101.gpio_otype.gpio_otype_n_ch_od"
configurationId="p101.gpio_otype"/>
#      <configSetting altId="p104.gpt1.gtioc1b" configurationId="p104"/>
#      <configSetting altId="p104.gpio_mode.gpio_mode_peripheral"
configurationId="p104.gpio_mode"/>
      <configSetting altId="p105.input" configurationId="p105"/>
      <configSetting altId="p105.gpio_mode.gpio_mode_in"
configurationId="p105.gpio_mode"/>
      <configSetting altId="p106.input" configurationId="p106"/>
      <configSetting altId="p106.gpio_mode.gpio_mode_in"
configurationId="p106.gpio_mode"/>
      <configSetting altId="p107.input" configurationId="p107"/>
      <configSetting altId="p107.gpio_mode.gpio_mode_in"
configurationId="p107.gpio_mode"/>
      <configSetting altId="p108.jtag_fslash_swd.swdio" configurationId="p108"/>
      <configSetting altId="p108.gpio_mode.gpio_mode_peripheral"
configurationId="p108.gpio_mode"/>
      <configSetting altId="p109.sci9.txd9" configurationId="p109"/>
      <configSetting altId="p109.gpio_mode.gpio_mode_peripheral"
configurationId="p109.gpio_mode"/>
      <configSetting altId="p110.sci9.rxd9" configurationId="p110"/>
      <configSetting altId="p110.gpio_mode.gpio_mode_peripheral"
configurationId="p110.gpio_mode"/>
      <configSetting altId="p111.input" configurationId="p111"/>
      <configSetting altId="p111.gpio_mode.gpio_mode_in"
configurationId="p111.gpio_mode"/>
#      <configSetting altId="p112.gpt3.gtioc3b" configurationId="p112"/>
#      <configSetting altId="p112.gpio_mode.gpio_mode_peripheral"
configurationId="p112.gpio_mode"/>
      <configSetting altId="p113.input" configurationId="p113"/>
      <configSetting altId="p113.gpio_mode.gpio_mode_in"
configurationId="p113.gpio_mode"/>
      <configSetting altId="p205.input" configurationId="p205"/>
      <configSetting altId="p205.gpio_mode.gpio_mode_in"
configurationId="p205.gpio_mode"/>
      <configSetting altId="p206.ssie0.ssidata0" configurationId="p206"/>
      <configSetting altId="p206.gpio_mode.gpio_mode_peripheral"
configurationId="p206.gpio_mode"/>
      <configSetting altId="p207.ssie0.ssilrck0" configurationId="p207"/>
      <configSetting altId="p207.gpio_mode.gpio_mode_peripheral"
configurationId="p207.gpio_mode"/>
      <configSetting altId="p208.input" configurationId="p208"/>
      <configSetting altId="p208.gpio_mode.gpio_mode_in"
configurationId="p208.gpio_mode"/>
      <configSetting altId="p300.jtag_fslash_swd.swclk" configurationId="p300"/>
      <configSetting altId="p300.gpio_mode.gpio_mode_peripheral"
configurationId="p300.gpio_mode"/>
      <configSetting altId="p301.input" configurationId="p301"/>
      <configSetting altId="p301.gpio_mode.gpio_mode_in"
configurationId="p301.gpio_mode"/>
      <configSetting altId="p402.input" configurationId="p402"/>
      <configSetting altId="p402.gpio_mode.gpio_mode_in"
configurationId="p402.gpio_mode"/>
      <configSetting altId="p403.input" configurationId="p403"/>
      <configSetting altId="p403.gpio_mode.gpio_mode_in"
configurationId="p403.gpio_mode"/>
      <configSetting altId="p407.ssie0.ssibck0" configurationId="p407"/>
      <configSetting altId="p407.gpio_mode.gpio_mode_peripheral"
configurationId="p407.gpio_mode"/>
      <configSetting altId="p408.input" configurationId="p408"/>
      <configSetting altId="p408.gpio_mode.gpio_mode_in"
configurationId="p408.gpio_mode"/>
      <configSetting altId="p409.input" configurationId="p409"/>
      <configSetting altId="p409.gpio_mode.gpio_mode_in"
configurationId="p409.gpio_mode"/>
      <configSetting altId="p410.input" configurationId="p410"/>
```

```
      <configSetting altId="p410.gpio_mode.gpio_mode_in"
configurationId="p410.gpio_mode"/>
      <configSetting altId="p411.output.high" configurationId="p411"/>
      <configSetting altId="p411.gpio_mode.gpio_mode_out.high"
configurationId="p411.gpio_mode"/>
      <configSetting altId="p500.input" configurationId="p500"/>
      <configSetting altId="p500.gpio_mode.gpio_mode_in"
configurationId="p500.gpio_mode"/>
      <configSetting altId="sci0.mode.simplei2c.free" configurationId="sci0.mode"/>
      <configSetting altId="sci0.rxd0.p100" configurationId="sci0.rxd0"/>
      <configSetting altId="sci0.txd0.p101" configurationId="sci0.txd0"/>
      <configSetting altId="sci9.mode.asynchronousuart.free"
configurationId="sci9.mode"/>
      <configSetting altId="sci9.rxd9.p110" configurationId="sci9.rxd9"/>
      <configSetting altId="sci9.txd9.p109" configurationId="sci9.txd9"/>
      <configSetting altId="ssie0.mode.half_dash_duplex.c"
configurationId="ssie0.mode"/>
      <configSetting altId="ssie0.pairing.c" configurationId="ssie0.pairing"/>
      <configSetting altId="ssie0.ssibck0.p407" configurationId="ssie0.ssibck0"/>
      <configSetting altId="ssie0.ssidata0.p206" configurationId="ssie0.ssidata0"/>
      <configSetting altId="ssie0.ssilrck0.p207" configurationId="ssie0.ssilrck0"/>
+     <lockSetting id="gpt1.gtioc1b" lock="true"/>
    </pincfg>
    <pincfg active="false" name="R7FA6E2BB3CFM.pincfg" selected="false" symbol="">
+      <configSetting altId="gpt1.gtioc1b.p104" configurationId="gpt1.gtioc1b"/>
+      <configSetting altId="gpt1.mode.gtiocaorgtiocb.free"
configurationId="gpt1.mode"/>
      <configSetting altId="jtag_fslash_swd.mode.swd.free"
configurationId="jtag_fslash_swd.mode"/>
      <configSetting altId="jtag_fslash_swd.swclk.p300"
configurationId="jtag_fslash_swd.swclk"/>
      <configSetting altId="jtag_fslash_swd.swdio.p108"
configurationId="jtag_fslash_swd.swdio"/>
+      <configSetting altId="p104.gpt1.gtioc1b" configurationId="p104"/>
+      <configSetting altId="p104.gpio_mode.gpio_mode_peripheral"
configurationId="p104.gpio_mode"/>
      <configSetting altId="p108.jtag_fslash_swd.swdio" configurationId="p108"/>
      <configSetting altId="p108.gpio_mode.gpio_mode_peripheral"
configurationId="p108.gpio_mode"/>
      <configSetting altId="p300.jtag_fslash_swd.swclk" configurationId="p300"/>
      <configSetting altId="p300.gpio_mode.gpio_mode_peripheral"
configurationId="p300.gpio_mode"/>
    </pincfg>
  </raPinConfiguration>
</raConfiguration>
```

Code Block 4 configuration.xml

Files to be added:

To initialize the timer PWM stack, middleware API support is required. The following are the files responsible for initializing the external interrupt stack. This function is called in **main_application.c** to initialize the stack.

```
/src/gpt_timer.c
/src/gpt_timer.h
```

The following is the content of the file that should be added to the project.

```
/***********************************************************************************************
 * File Name    : gpt_timer.c
 * Description  : Contains function definition.

***********************************************************************************************/
/***********************************************************************************************
* Copyright (c) 2020 - 2024 Renesas Electronics Corporation and/or its affiliates
*
* SPDX-License-Identifier: BSD-3-Clause
***********************************************************************************************/

#include "common_utils.h"
#include "gpt_timer.h"
#include "log_disabled.h"
//#include "log_error.h"
//#include "log_warning.h"
//#include "log_info.h"
//#include "log_debug.h"
/***********************************************************************************************
 * @addtogroup r_gpt_ep
 * @{

***********************************************************************************************/

/* Boolean flag to determine one-shot mode timer is expired or not.*/

/* Store Timer open state*/

/***********************************************************************************************
 * @brief       Initialize GPT timer.
 * @param[in]   p_timer_ctl     Timer instance control structure
 * @param[in]   p_timer_cfg     Timer instance Configuration structure
 * @param[in]   timer_mode      Mode of GPT Timer
 * @retval      FSP_SUCCESS     Upon successful open of timer.
 * @retval      Any Other Error code apart from FSP_SUCCES on Unsuccessful open .

***********************************************************************************************/
fsp_err_t init_gpt_timer(timer_ctrl_t * const p_timer_ctl, timer_cfg_t const * const
p_timer_cfg)
{
    fsp_err_t err = FSP_SUCCESS;

    /* Initialize GPT Timer */
    err = R_GPT_Open(p_timer_ctl, p_timer_cfg);
    if (FSP_SUCCESS != err)
    {
        log_error ("\r\n ** R_GPT_TimerOpen FAILED ** \r\n");
        return err;
    }

    return err;
}

/***********************************************************************************************
 * @brief       Start GPT timers in periodic, one shot, PWM mode.
 * @param[in]   p_timer_ctl     Timer instance control structure
 * @retval      FSP_SUCCESS     Upon successful start of timer.
```

```
 * @retval      Any Other Error code apart from FSP_SUCCES on Unsuccessful start .

***********************************************************************************************************/
fsp_err_t start_gpt_timer (timer_ctrl_t * const p_timer_ctl)
{
    fsp_err_t err = FSP_SUCCESS;

    /* Starts GPT timer */
    err = R_GPT_Start(p_timer_ctl);
    if (FSP_SUCCESS != err)
    {
        /* In case of GPT_open is successful and start fails, requires a immediate
cleanup.
         * Since, cleanup for GPT open is done in start_gpt_timer,Hence cleanup is not
required */
        log_error ("\r\n ** R_GPT_Start API failed ** \r\n");
    }
    return err;
}



/***********************************************************************************************************
*************************
 *  @brief       set  duty cycle of PWM timer.
 *  @param[in]   duty_cycle_percent.
 *  @retval      FSP_SUCCESS on correct duty cycle set.
 *  @retval      FSP_INVALID_ARGUMENT on invalid info.

***********************************************************************************************************/
fsp_err_t set_timer_duty_cycle(uint8_t duty_cycle_percent ,timer_ctrl_t * const
p_timer_ctl)
{
    fsp_err_t err                           = FSP_SUCCESS;
    uint32_t duty_cycle_counts              = RESET_VALUE;
    uint32_t current_period_counts          = RESET_VALUE;
    timer_info_t info                       = {(timer_direction_t)RESET_VALUE,
RESET_VALUE, RESET_VALUE};

    /* Get the current period setting. */
    err = R_GPT_InfoGet(p_timer_ctl, &info);
    if (FSP_SUCCESS != err)
    {
        /* GPT Timer InfoGet Failure message */
        log_error ("\r\n ** R_GPT_InfoGet API failed ** \r\n");
    }
    else
    {
        /* update period counts locally. */
        current_period_counts = info.period_counts;

        /* Calculate the desired duty cycle based on the current period. Note that if
the period could be larger than
         * UINT32_MAX / 100, this calculation could overflow. A cast to uint64_t is used
to prevent this. The cast is
         * not required for 16-bit timers. */
        duty_cycle_counts =(uint32_t) ((uint64_t) (current_period_counts *
duty_cycle_percent) /
                GPT_MAX_PERCENT);
#if defined(BOARD_RA4W1_EK) || defined (BOARD_RA6T1_RSSK) ||defined (BOARD_RA6T3_MCK) ||
defined (BOARD_RA4T1_MCK)
        duty_cycle_counts = (current_period_counts - duty_cycle_counts);
#endif
```

```
        /* Duty Cycle Set API set the desired intensity on the on-board LED */
        err = R_GPT_DutyCycleSet(p_timer_ctl, duty_cycle_counts, TIMER_PIN);
        if(FSP_SUCCESS != err)
        {
            /* GPT Timer DutyCycleSet Failure message */
            /* In case of GPT_open is successful and DutyCycleSet fails, requires a
immediate cleanup.
             * Since, cleanup for GPT open is done in timer_duty_cycle_set,Hence cleanup
is not required */
            log_error ("\r\n ** R_GPT_DutyCycleSet API failed ** \r\n");
        }
    }
    return err;
}


/***********************************************************************************
************************
 *  @brief       set  duty cycle of PWM timer.
 *  @param[in]   duty_cycle_percent.
 *  @retval      FSP_SUCCESS on correct duty cycle set.
 *  @retval      FSP_INVALID_ARGUMENT on invalid info.

***********************************************************************************
************************/
fsp_err_t set_timer_Period_and_Dutycycle(uint32_t period_counts ,uint8_t
duty_cycle_percent, timer_ctrl_t * const p_timer_ctl)
{
    fsp_err_t err                           = FSP_SUCCESS;
    uint32_t duty_cycle_counts              = RESET_VALUE;
    uint32_t current_period_counts          = RESET_VALUE;
    timer_info_t info                       = {(timer_direction_t)RESET_VALUE,
RESET_VALUE, RESET_VALUE};
    gpt_instance_ctrl_t * p_instance_ctrl = (gpt_instance_ctrl_t *) p_timer_ctl;
    /* PeriodSet Set API set the desired intensity on the on-board LED */
    err = R_GPT_PeriodSet(p_timer_ctl, period_counts);
    if(FSP_SUCCESS != err)
    {
        /* GPT Timer DutyCycleSet Failure message */
        /* In case of GPT_open is successful and DutyCycleSet fails, requires a
immediate cleanup.
         * Since, cleanup for GPT open is done in timer_duty_cycle_set,Hence cleanup is
not required */
        log_error ("\r\n ** R_GPT_PeriodSet API failed ** \r\n");
    }


    current_period_counts = period_counts;

    /* Calculate the desired duty cycle based on the current period. Note that if the
period could be larger than
     * UINT32_MAX / 100, this calculation could overflow. A cast to uint64_t is used to
prevent this. The cast is
     * not required for 16-bit timers. */
    duty_cycle_counts =(uint32_t) ((uint64_t) (current_period_counts *
duty_cycle_percent) /
            GPT_MAX_PERCENT);
#if defined(BOARD_RA4W1_EK) || defined (BOARD_RA6T1_RSSK) ||defined (BOARD_RA6T3_MCK) ||
defined (BOARD_RA4T1_MCK)
    duty_cycle_counts = (current_period_counts - duty_cycle_counts);
#endif

    /* Duty Cycle Set API set the desired intensity on the on-board LED */
    err = R_GPT_DutyCycleSet(p_timer_ctl, duty_cycle_counts, TIMER_PIN);
    if(FSP_SUCCESS != err)
    {
```

```
        /* GPT Timer DutyCycleSet Failure message */
        /* In case of GPT_open is successful and DutyCycleSet fails, requires a
immediate cleanup.
         * Since, cleanup for GPT open is done in timer_duty_cycle_set,Hence cleanup is
not required */
        log_error ("\r\n ** R_GPT_DutyCycleSet API failed ** \r\n");
    }
    return err;
}


/*******************************************************************************************
************************
 * @brief      Close the GPT HAL driver.
 * @param[in]  p_timer_ctl     Timer instance control structure
 * @retval     None

*******************************************************************************************
**********************/
void deinit_gpt_timer(timer_ctrl_t * const p_timer_ctl)
{
    fsp_err_t err = FSP_SUCCESS;

    /* Timer Close API call*/
    err = R_GPT_Close(p_timer_ctl);
    if (FSP_SUCCESS != err)
    {
        /* GPT Close failure message */
        log_error("\r\n ** R_GPT_Close FAILED ** \r\n");
    }

}


/*******************************************************************************************
***************************//**
 * @} (end addtogroup r_gpt_ep)

*******************************************************************************************
*****************************/
```

Code Block 5 /src/gpt_timer.c

```
/***********************************************************************************************
*******************************
 * File Name    : gpt_timer.h
 * Description  : Contains Macros and function declarations.

***********************************************************************************************
*******************************/
/***********************************************************************************************
*******************************
* Copyright (c) 2020 - 2024 Renesas Electronics Corporation and/or its affiliates
*
* SPDX-License-Identifier: BSD-3-Clause
***********************************************************************************************
*******************************/
#ifndef GPT_TIMER_H_
#define GPT_TIMER_H_

/* Macros definitions */
#define GPT_MAX_PERCENT          (100U)          /* Max Duty Cycle percentage */
#define BUF_SIZE                 (16U)           /* Size of buffer for RTT input data */
#define PERIODIC_MODE_TIMER      (1U)            /* To perform GPT Timer in Periodic
mode */
#define PWM_MODE_TIMER           (2U)            /* To perform GPT Timer in PWM mode */
#define ONE_SHOT_MODE_TIMER      (3U)            /* To perform GPT Timer in ONE-SHOT
mode */
#define INITIAL_VALUE            ('\0')
#define TIMER_UNITS_MILLISECONDS  (1000U)        /* timer unit in millisecond */
#define CLOCK_TYPE_SPECIFIER      (1ULL)         /* type specifier */

/* GPT Timer Pin for boards */
#define TIMER_PIN           (GPT_IO_PIN_GTIOCB)

#if defined (BOARD_RA2A1_EK) || defined (BOARD_RA4W1_EK) || defined (BOARD_RA4T1_MCK) ||
defined (BOARD_RA4E2_EK)
#define GPT_MAX_PERIOD_COUNT     (0XFFFF)        /* Max Period Count for 16-bit Timer*/
#else
#define GPT_MAX_PERIOD_COUNT     (0XFFFFFFFF)    /* Max Period Count for 32-bit Timer*/
#endif

#define PERIODIC_MODE      (1U)                  /* To check status of GPT Timer in
Periodic mode */
#define PWM_MODE           (2U)                  /* To check status of GPT Timer in PWM
mode */
#define ONE_SHOT_MODE      (3U)                  /* To check status of GPT Timer in oNE-
SHOT mode */

#define EP_INFO    "\r\nThe project initializes GPT module in Periodic, PWM or One-shot
mode based on user input " \
        "from the displayed menu options." \
        "\r\nIn periodic mode, user can enter the time period within the permitted
ranges to change "\
        "the frequency of the user LED." \
        "\r\nIn PWM mode, user can enter the duty cycle within the specified range to
adjust the "\
        "intensity of the user LED." \
        "\r\nIn ONE SHOT mode, Output will be displayed on JlinkRTTViewer when timer
expires.\r\n "

/* Function declaration */
fsp_err_t init_gpt_timer(timer_ctrl_t * const p_timer_ctl, timer_cfg_t const * const
p_timer_cfg);
fsp_err_t start_gpt_timer (timer_ctrl_t * const p_timer_ctl);
fsp_err_t set_timer_duty_cycle(uint8_t duty_cycle_percent ,timer_ctrl_t * const
p_timer_ctl);
uint32_t  process_input_data(void);
void deinit_gpt_timer(timer_ctrl_t * const p_timer_ctl);
```

```
void print_timer_menu(void);
fsp_err_t set_timer_Period_and_Dutycycle(uint32_t period_counts ,uint8_t
duty_cycle_percent, timer_ctrl_t * const p_timer_ctl);

#endif /* GPT_TIMER_H_ */
```

Code Block 6 /src/gpt_timer.h

Next, edit the **main_application.c** file as the following.

```
#include <stdio.h>
#include "common_utils.h"
#include "main_application.h"
+ #include "gpt_timer.h"
// Uncomment the desired debug level
#include "log_disabled.h"
//#include "log_error.h"
//#include "log_warning.h"
//#include "log_info.h"
//#include "log_debug.h"



+ #ifdef  ENABLE_PWM_LED1
+ void enable_Led1_PWM(void)
+ {
+   fsp_err_t err = FSP_SUCCESS;
+   uint32_t gpt_desired_duty_cycle_percent = RESET_VALUE;
+   uint32_t gpt_desired_period_ms          = RESET_VALUE;
+   uint64_t period_counts                  = RESET_VALUE;
+   uint32_t pclkd_freq_hz                  = RESET_VALUE;
+   gpt_desired_duty_cycle_percent = LED1_DEFAULT_DUTY_CYCLE ; // edit here to change
duty cycle
+   log_debug( "LED1:Default Period:%d ms   Default Duty Cycle:%d \r\n",
((LED1_DEFAULT_PERIOD* 1000) /SET_PERIOD_1_SECS),LED1_DEFAULT_DUTY_CYCLE);

+   /* Validate Duty cycle percentage */
+   if (GPT_MAX_PERCENT < gpt_desired_duty_cycle_percent)
+   {
+       log_error ("\r\n ** LED1:INVALID INPUT, DESIRED DUTY CYCLE IS OUT OF RANGE. **
\r\n");
+   }
+   else
+   {
+           /* we got valid input, Initialize PWM timer */
+           err = init_gpt_timer(&g_timer_pwm_led1_ctrl, &g_timer_pwm_led1_cfg);
+           if(FSP_SUCCESS != err)
+           {
+               log_error("** LED1:GPT TIMER INIT FAILED ** \r\n");
+           }
+           log_debug("LED1:Opened Timer in PWM Mode\r\n");
+
+           /* Start PWM Timer*/
+           err = start_gpt_timer(&g_timer_pwm_led1_ctrl);
+           if(FSP_SUCCESS != err)
+           {
+               log_error("** LED1:GPT TIMER START FAILED ** \r\n");
+               /*Close PWM Timer instance */
+               deinit_gpt_timer(&g_timer_pwm_led1_ctrl);
+           }
+           log_debug("LED1:Started Timer in PWM Mode\r\n");
+
+       /* Set DutyCycle and period of PWM timer */
+           err =
set_timer_Period_and_Dutycycle(LED1_DEFAULT_PERIOD,gpt_desired_duty_cycle_percent,&g_tim
er_pwm_led1_ctrl);
+           if(FSP_SUCCESS != err)
+           {
+               /* GPT Timer DutyCycleSet Failure message */
+               log_error ("\r\n ** LED1:Timer Period and Dutycycle SET FAILED **
\r\n");
+               /*Close PWM Timer instance */
+               deinit_gpt_timer(&set_timer_Period_and_Dutycycle);
+           }
+   }
+}
```

```
+#endif

+ #ifdef  ENABLE_PWM_LED2
+ void enable_Led2_PWM(void)
+ {
+   fsp_err_t err = FSP_SUCCESS;
+   uint32_t gpt_desired_duty_cycle_percent = RESET_VALUE;
+   uint32_t gpt_desired_period_ms          = RESET_VALUE;
+   uint64_t period_counts                  = RESET_VALUE;
+   uint32_t pclkd_freq_hz                  = RESET_VALUE;
+   gpt_desired_duty_cycle_percent = LED2_DEFAULT_DUTY_CYCLE ; // edit here to change
duty cycle
+   log_debug( "LED2:Default Period:%d ms   Default Duty Cycle:%d \r\n",
((LED2_DEFAULT_PERIOD* 1000) /SET_PERIOD_1_SECS),LED2_DEFAULT_DUTY_CYCLE);
+
+   /* Validate Duty cycle percentage */
+   if (GPT_MAX_PERCENT < gpt_desired_duty_cycle_percent)
+   {
+       log_error ("\r\n ** LED2:INVALID INPUT, DESIRED DUTY CYCLE IS OUT OF RANGE. **
\r\n");
+   }
+   else
+   {
+           /* we got valid input, Initialize PWM timer */
+           err = init_gpt_timer(&g_timer_pwm_led2_ctrl, &g_timer_pwm_led2_cfg);
+           if(FSP_SUCCESS != err)
+           {
+               log_error("** LED2:GPT TIMER INIT FAILED ** \r\n");
+           }
+           log_debug("LED2:Opened Timer in PWM Mode\r\n");
+
+           /* Start PWM Timer*/
+           err = start_gpt_timer(&g_timer_pwm_led2_ctrl);
+           if(FSP_SUCCESS != err)
+           {
+               log_error("** LED2:GPT TIMER START FAILED ** \r\n");
+               /*Close PWM Timer instance */
+               deinit_gpt_timer(&g_timer_pwm_led2_ctrl);
+           }
+           log_debug("Started Timer in PWM Mode\r\n");
+
+       err =
set_timer_Period_and_Dutycycle(LED2_DEFAULT_PERIOD,gpt_desired_duty_cycle_percent,&g_tim
er_pwm_led2_ctrl);
+       if(FSP_SUCCESS != err)
+       {
+           /* GPT Timer DutyCycleSet Failure message */
+           log_error ("\r\n ** LED2:Timer Period and Dutycycle SET FAILED ** \r\n");
+           /*Close PWM Timer instance */
+           deinit_gpt_timer(&set_timer_Period_and_Dutycycle);
+       }
+   }
+ }
+ #endif

void main_application(void) {

// Start of autogenerated code

+ #if defined(ENABLE_LED1) && defined(ENABLE_PWM_LED1)
+  enable_Led1_PWM();
+ #endif

+ #if defined(ENABLE_LED2) && defined(ENABLE_PWM_LED2)
+  enable_Led2_PWM();
+ #endif
```

```
  while(true)
  {
    // do nothing
  }
// End of autogenerated code

}
```

Code Block 7 main_application.c

Updated **main_application.h** file.

```
#ifndef __MAIN_APPLICATION_H
#define __MAIN_APPLICATION_H

#include "common_utils.h"
+ #define SET_PERIOD_1_SECS       0x10000  // 1 second period
+ #define SET_PERIOD_500_MSECS    0x8000   // 500 ms period
+ #define SET_PERIOD_250_MSECS    0x4000   // 250 ms period
+ #define SET_PERIOD_125_MSECS    0x2000   // 125 ms period
+ #define SET_PERIOD_75_MSECS     0x1000   // 75 ms period
+ #define SET_PERIOD_37_MSECS     0x800    // 37.5 ms period
+ #define SET_PERIOD_16_MSECS     0x400   // 16 ms period
+ #define SET_PERIOD_8_MSECS      0x200   // 8 ms period
+ #define SET_PERIOD_4_MSECS      0x100   // 4 ms period
+ #define SET_PERIOD_2_MSECS      0x80    // 2 ms period
+ #define SET_PERIOD_1_MSECS      0x40    // 1 ms period

+ #define ENABLE_LED1    // Enable or disable LED1
+ #define ENABLE_LED2    // Enable or disable LED2

+ #ifdef ENABLE_LED1
+ #define ENABLE_PWM_LED1
+ #endif

+ #ifdef ENABLE_LED2
+ #define ENABLE_PWM_LED2
+ #endif

+ #ifdef ENABLE_PWM_LED1
+ #define LED1_DEFAULT_DUTY_CYCLE 50    // 50 %
+ #define LED1_DEFAULT_PERIOD         SET_PERIOD_1_SECS
+ #endif
+ #ifdef ENABLE_PWM_LED2  // 50 %
+ #define LED2_DEFAULT_PERIOD        SET_PERIOD_1_SECS
+ #define LED2_DEFAULT_DUTY_CYCLE 50
+ #endif
/* Function declaration */
void main_application(void);

#endif /* __MAIN_APPLICATION_H */
```

Code Block 8 main_application.h

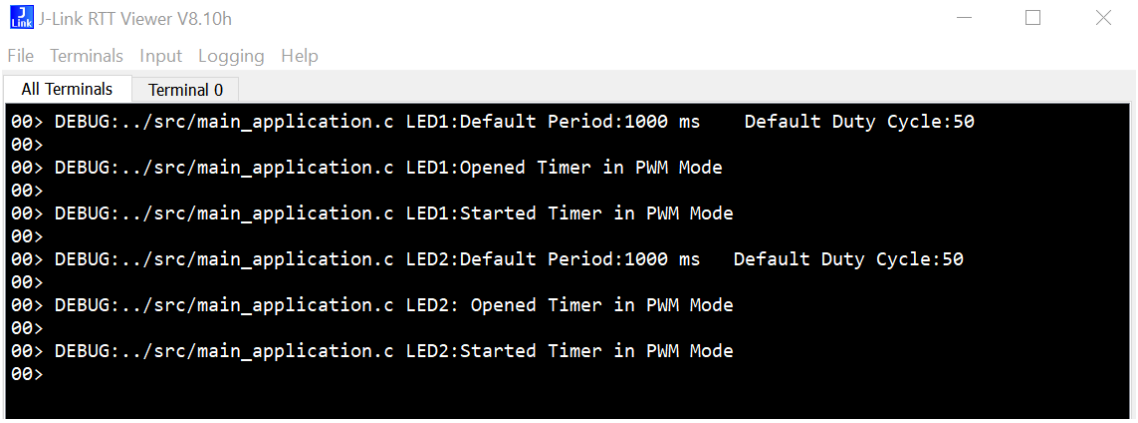
The user can enable or disable the PWM code for individual LEDs using the macros provided below.
To use this feature, the user must make the following changes in **main_application.h**:

Enable the Macro:

```
#define ENABLE_LED1    // Enable or disable LED1
#define ENABLE_LED2    // Enable or disable LED2
```

Expected Output RTT Logs:

Check the following logs for the expected behavior and also observe LEDs on Board to co-relate with the feature.



10.3.2.4 Adjust the LED Period using Macro

This exercise is a continuation from the previous section. It is assumed that the user has replaced configuration.xml, added gpt_timer.c and gpt_timer.h, and made modifications to main_application.c and main_application.h.

The updated files are available with this GitHub link.

By default, both LEDs are configured with a 1s period. However, the blink interval can be modified by changing the macro values in **main_application.h**:

- For LED1: #define LED1_DEFAULT_PERIOD
- For LED2: #define LED2_DEFAULT_PERIOD

```
#define SET_PERIOD_1_SECS       0x10000  // 1 second period
#define SET_PERIOD_500_MSECS    0x8000   // 500 ms period
#define SET_PERIOD_250_MSECS    0x4000   // 250 ms period
#define SET_PERIOD_125_MSECS    0x2000   // 125 ms period
#define SET_PERIOD_75_MSECS     0x1000   // 75 ms period
#define SET_PERIOD_37_MSECS     0x800    // 37.5 ms period
#define SET_PERIOD_16_MSECS     0x400   // 16 ms period
#define SET_PERIOD_8_MSECS      0x200   // 8 ms period
#define SET_PERIOD_4_MSECS      0x100    // 4 ms period
#define SET_PERIOD_2_MSECS      0x80    // 2 ms period
#define SET_PERIOD_1_MSECS      0x40    // 1 ms period
```

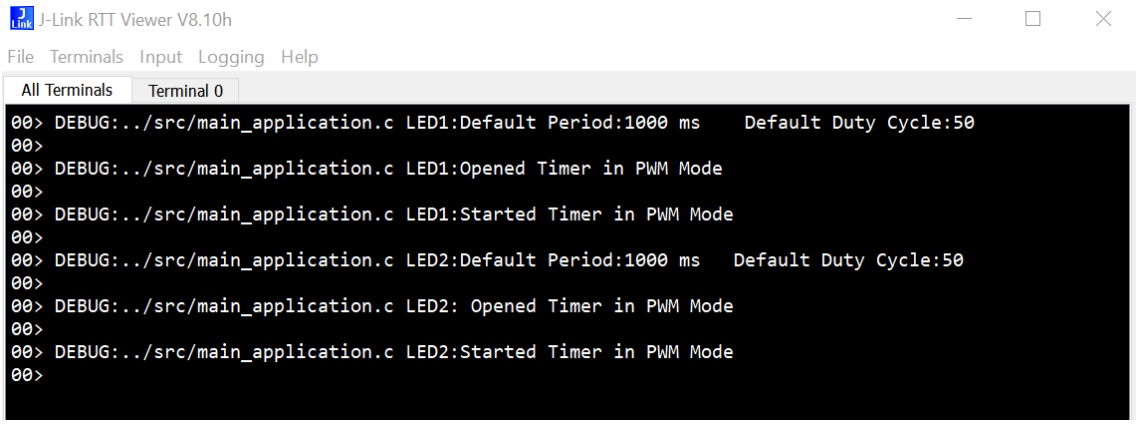
Any of the previous defined values can be selected and set with LED periods like this:

```
#define LED1_DEFAULT_PERIOD     SET_PERIOD_1_SECS
#define LED2_DEFAULT_PERIOD     SET_PERIOD_1_SECS
```

These macros can be adjusted to achieve the required LED blink intervals.

Expected Output RTT Logs:

Check the following logs for the expected behavior and also observe LEDs on board to co-relate with the feature.



10.3.2.5 Adjust the LED ON/OFF duration using macro

Users can adjust the On/OFF duration (that is the duty cycle) using the predefined macro in **main_application.h** file.

These macros as by default set a value of 50% for a duty cycle but they can be set as per required from the range of 1 to 100%.

- #define LED1_DEFAULT_DUTY_CYCLE 50 // duty cycle macro for LED 1
- #define LED2_DEFAULT_DUTY_CYCLE 50 // duty cycle macro for LED 2

The LEDs are configured with a 1-second period, allowing to be set the duty cycle from 1% to 100%.

- A 1% duty cycle means the LED will be ON for 10ms and OFF for 990ms.
- A 100% duty cycle means the LED stays ON continuously.

This way, users can adjust the ON/OFF duration anywhere between 10ms (minimum ON time) and 990ms (maximum OFF-time), depending on the required brightness or behavior.

Note: To control the LED brightness using PWM, the PWM period should be reduced to a level where the LED's on/off toggling is not visually perceptible. This allows for smooth brightness control through the duty cycle.

In this example, set the PWM period to 37ms by using the macro SET_PERIOD_37_MSECS for LED1_DEFAULT_PERIOD, and then vary the duty cycle between 1% to 100% to adjust brightness.

Example:

Maximum brightness

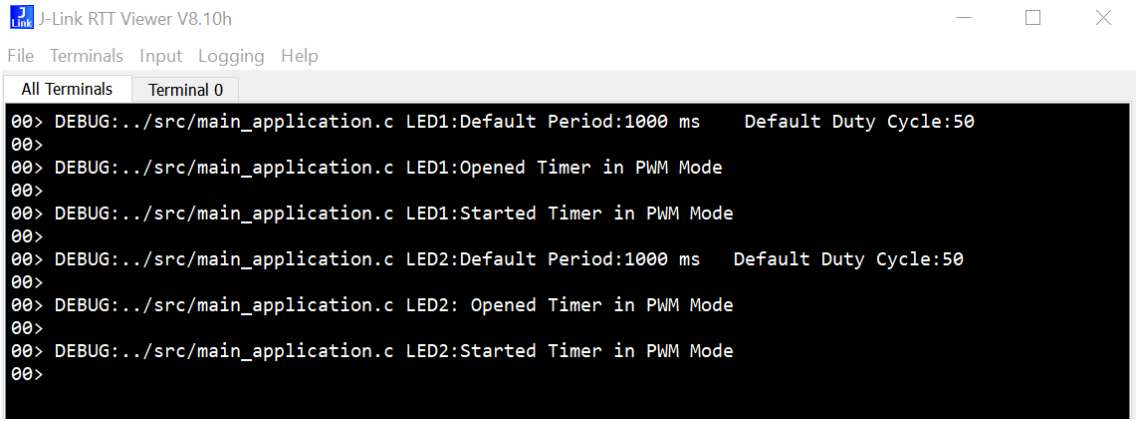
```
#define LED1_DEFAULT_DUTY_CYCLE             100 // 100 %
#define LED1_DEFAULT_PERIOD                 SET_PERIOD_37_MSECS
```

Minimum brightness

```
#define LED1_DEFAULT_DUTY_CYCLE             1 // 1 %
#define LED1_DEFAULT_PERIOD                 SET_PERIOD_37_MSECS
```

Expected Output RTT Logs:

Check the following logs for the expected behavior and also observe LEDs on the board to correlate with the feature.



10.3.3 Customize Sensor Data to AWS Cloud Application

As a first step, the user must create a project in the QuickConnect Studio by selecting the appropriate MCU kit (for example, BGK-RA6E2), sensor (for example, HS4001 Temperature / Humidity), connectivity module (for example, DA16600 Wi-Fi) and Sensor Data to AWS Cloud (MQTT onchip) application

10.3.3.1 Exercise 1: Change Sensor Units (C to F or F to C)

This exercise is to update the temperature units between C and F. The default unit for temperature on QCS is Celsius. This exercise publishes temperature data into AWS in Fahrenheit.

Files to be modified:

- **\src\main_thread_entry.c**

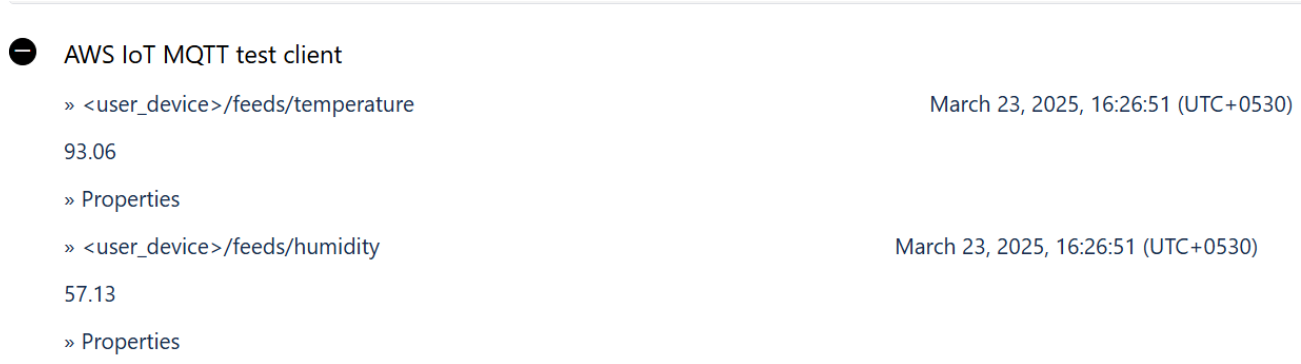
Modifications:

Modify the following code in the main_thread_entry() function to convert the sensor data if the parameter is temperature only.

```
void main_thread_entry(void *pvParameters) {
FSP_PARAMETER_NOT_USED (pvParameters);
uint8_t timeout = 10;
while (1) {
    sm_handle sensor_handle;
    uint8_t sensor_count = 0;
    .
    .
    .
    .
        if (pdTRUE == xQueueReceive(g_sensor_queue,
   &sensor_data, PUBLISHING_INTERVAL_MS)) {
            // Store the received data into the right slot
            for (index = 0; index < NUM_SENSORS; index++) {
                if (0 == sensor_slots[index].handle.value ||
   sensor_slots[index].handle.value ==
   sensor_data.handle.value) {
                    // Found a slot
                    sensor_slots[index].handle.value =
   sensor_data.handle.value;
-                   sensor_slots[index].data =
   sensor_data.data;
+                   sensor_slots[index].data =  (TEMPERATURE
   ==
   sm_get_sensor_type_by_handle(sensor_slots[index].handle))
   ? \
+
   ((sensor_data.data * 9) / 5) + 3200 : \
+
   sensor_data.data;
                    num_sensors = index + 1;
                    break;
                }
            }
        }
    .
    .
    }
}
```

Code Block 9 main_thread_entry.c changes

Expected output:



RTT Viewer

```
00> INFO:../src/sensor/hs4001_sensor.c Sensor channel 0 open success
00> INFO:../src/sensor/hs4001_sensor.c Sensor channel 1 open success
00> DEBUG:../src/sensor/hs4001_sensor.c Sensor read channel 0
00> INFO:../src/main_thread_entry.c Starting Wi-Fi connection...
00> INFO:../src/main_thread_entry.c Wi-Fi connection successful!
00> INFO:../src/main_thread_entry.c MQTT setup successful!
00> INFO:../src/main_thread_entry.c MQTT Subscribe to user_device/feeds/led
00> DEBUG:../src/sensor/hs4001_sensor.c Sensor read channel 0
00> DEBUG:../src/sensor/hs4001_sensor.c Sensor read channel 1
00> DEBUG:../src/main_thread_entry.c Topic: <user_device>/feeds/humidity data: 57.13
00> DEBUG:../src/main_thread_entry.c Topic: <user_device>/feeds/temperature data: 93.06
```

10.3.3.2 Change Frequency of Sensor Update

This exercise is to change the sensor update time from 10 seconds to one minute. The default time for the sensor update on QCS is 30 seconds. This exercise changes the sensor data publish interval the into AWS cloud.

Files to be modified:

- **\src\main_thread_entry.c**

Modifications: Define the following mentioned enum in the User Macros section and assign the variable to PUBLISHING_INTERVAL_M:

```
/**********************************************************************
 ********************
 User Macros
**********************************************************************
 ********************/

+// Define sensor update periods (in milliseconds)
+typedef enum {
+    PERIOD_10SEC = 10000,
+    PERIOD_20SEC = 10000,
+    PERIOD_30SEC = 30000,
+    PERIOD_45SEC = 45000,
+    PERIOD_1MIN  = 60000
+} SensorUpdatePeriod_t;
+SensorUpdatePeriod_t sensor_update_period =  PERIOD_30SEC;
    // Default setting

#define QUEUE_RECEIVE_TIMEOUT                    (100)   //
    this is the interval we also receive MQTT messages
-#define PUBLISHING_INTERVAL_MS                  (30000)
+#define PUBLISHING_INTERVAL_MS
    sensor_update_period
```

Code Block 10 main_thread_entry.c changes

10.3.3.3 Modify the MQTT Topic Name as <sensor name>/<parameter>

This exercise is to modify the **aws mqqt** publish topic name includes sensor name. The default string in the topic name in QCS is feeds. This is modified by the sensor name (for example, hs4001_sensor).

Files to be modified:

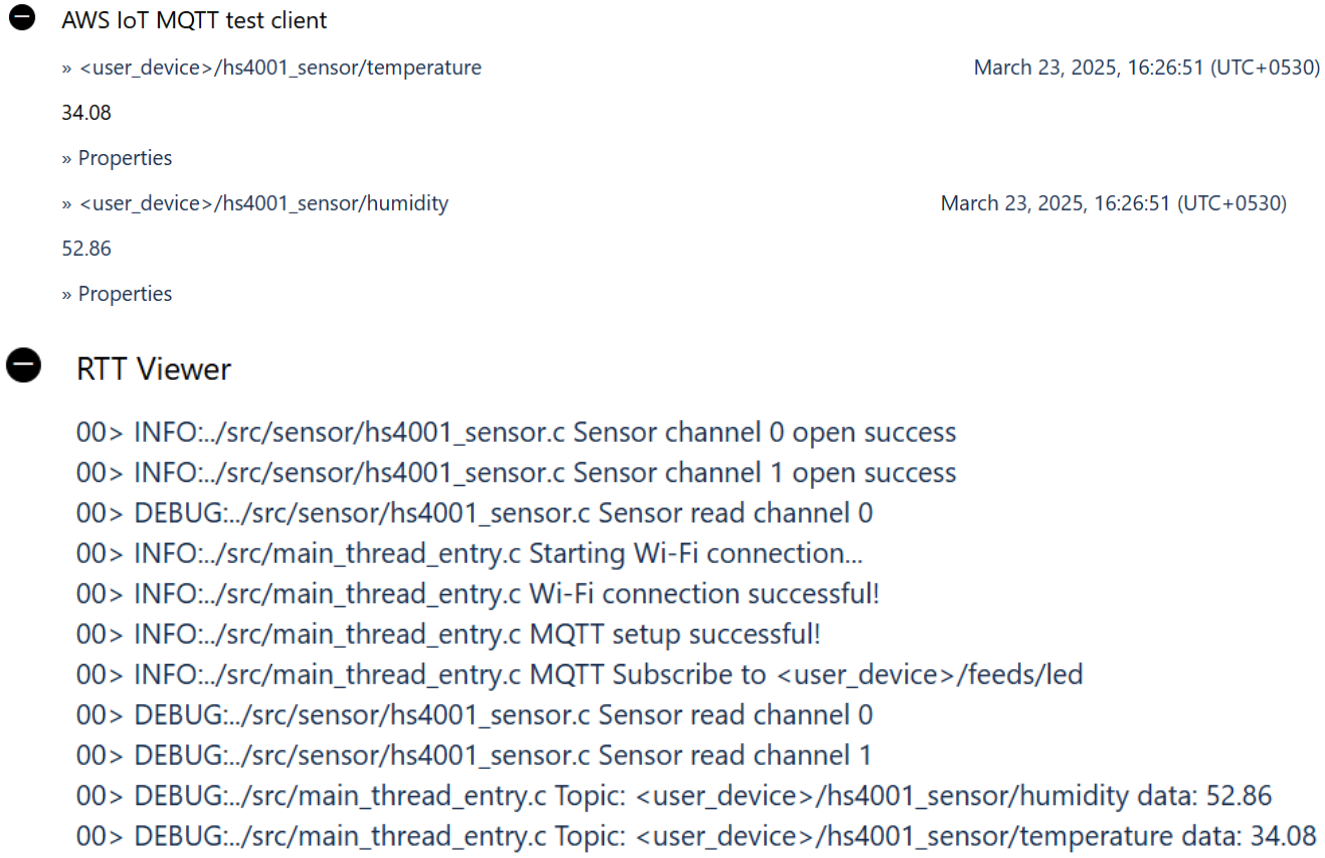
- **\src\main_thread_entry.c**

Modifications: Define the following macro mentioned in User Macros section and change the mentioned code in main_thread_entry() function:

```
/**********************************************************************
 ********************
 User Macros
**********************************************************************
 ********************/

+// create automatic headers for each sensor
+#define DEFINE_SENSOR_DRIVER(DRIVER) char sensor_name[] =
   #DRIVER;
+#include "sm_define_sensors.inc"

/**********************************************************************
 ********************
 User function implementations
**********************************************************************
 ********************/

void main_thread_entry(void *pvParameters) {
FSP_PARAMETER_NOT_USED (pvParameters);
uint8_t timeout = 10;
while (1) {
    sm_handle sensor_handle;
    uint8_t sensor_count = 0;
    .
    .
    .
    .
               char pub_message[WIDTH_64];
               char pub_topic[WIDTH_64];
-              snprintf(pub_topic, WIDTH_64,
   IO_USERNAME"/feeds/%s",\
+              snprintf(pub_topic, WIDTH_64,
   IO_USERNAME"/%s/%s",\
+                       sensor_name,\

   sm_get_sensor_path_by_handle(sensor_slots[index2].handle)
   ,\
                      );
    .
    .
    }
}
```

Code Block 11 main_thread_entry.c changes

Expected output:



10.3.3.4 Identify Min, Max, and Averaging of Sensor Values (Send Only when Absolute Change Observed)

This exercise is to calculate the statistics (min, max average, and absolute change) of a sensor parameter and publish the statistics.

Files to be modified:

\src\main_thread_entry.c

Modifications: Define the following mentioned enum in the User Macro section, add the variables to the User Global Variables section, and assign the variable to PUBLISHING_INTERVAL_MS as mentioned:

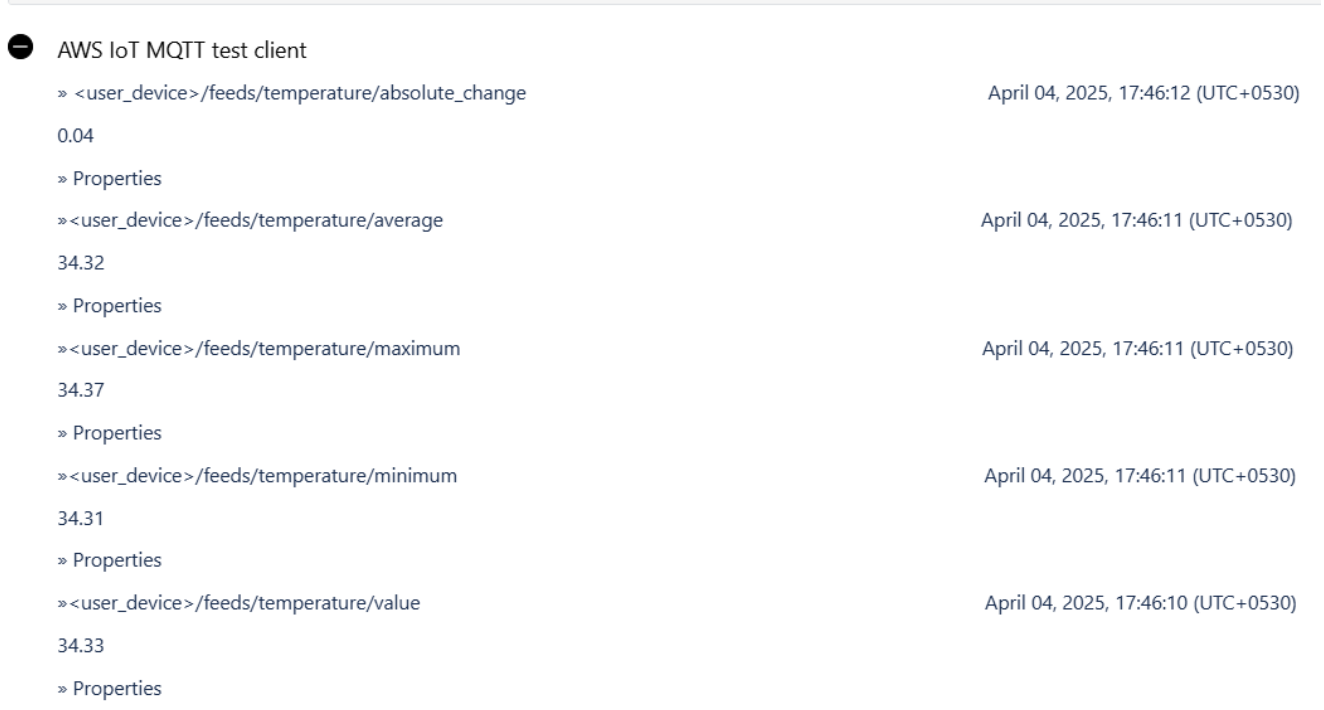
```
/*****************************************************************
   ******************
 User Macros
*****************************************************************
   ******************/
+// Define statistics offsets
+typedef enum {
+   VALUE = 0,
+   MIN,
+   MAX,
+   AVG,
+   ABS,
+   COUNT,
+   STATS_MAX_PARAMS
+}SensorStatsParams_t;

/*****************************************************************
   ******************
 User global variables
*****************************************************************
   ******************/

+static const char* stats_names[] = {"value", "minimum",
   "maximum", "average", "absolute_change"};
+#define STATS_NAMES_COUNT (sizeof(stats_names) /
   sizeof(stats_names[0]))
+static int32_t stats_values[NUM_SENSORS][STATS_MAX_PARAMS];

/*****************************************************************
   ******************
 User function prototype declarations
*****************************************************************
   ******************/

+void update_sensor_stats(int32_t new_value, uint32_t
   senor_index);

/*****************************************************************
   ******************
 User function implementations
*****************************************************************
   ******************/

+// Function to update sensor statistics
+void update_sensor_stats(int32_t new_value, uint32_t
   senor_index) {
+   if(0 == stats_values[senor_index][COUNT])
+   {
+       stats_values[senor_index][MIN]      = new_value;
+       stats_values[senor_index][MAX]      = new_value;
+       stats_values[senor_index][AVG]      = new_value;
+       stats_values[senor_index][VALUE]    = new_value;
+       stats_values[senor_index][ABS]      = 0;
+       stats_values[senor_index][COUNT]++;
+       return;
+   }
+   if (new_value < stats_values[senor_index][MIN]) {
+       stats_values[senor_index][MIN] = new_value;
+    }
+
+    if (new_value > stats_values[senor_index][MAX]) {
+       stats_values[senor_index][MAX] = new_value;
+    }

+    stats_values[senor_index][AVG] =
   (stats_values[senor_index][AVG] *
```

```
    stats_values[senor_index][COUNT] + new_value) /
    (stats_values[senor_index][COUNT] + 1);
+     stats_values[senor_index][COUNT]++;

+     stats_values[senor_index][ABS] =   (new_value >
    stats_values[senor_index][VALUE]) ? \
+                                        (new_value -
    stats_values[senor_index][VALUE]) : \
+
    (stats_values[senor_index][VALUE] - new_value);
+     stats_values[senor_index][VALUE] = new_value;
+}
+

void main_thread_entry(void *pvParameters) {
FSP_PARAMETER_NOT_USED (pvParameters);
uint8_t timeout = 10;
while (1) {
    sm_handle sensor_handle;
    uint8_t sensor_count = 0;
    .
    .
    .
    .
                char pub_message[WIDTH_64];
                char pub_topic[WIDTH_64];
-               snprintf(pub_topic, WIDTH_64,
    IO_USERNAME"/feeds/%s",\
-
    sm_get_sensor_path_by_handle(sensor_slots[index2].handle)
    \
-                       );
-
-                 snprintf((char*)pub_message, WIDTH_64,
    "%ld.%02ld",\
-                       sensor_slots[index2].data/100, \
-                       sensor_slots[index2].data%100);
+
    update_sensor_stats(sensor_slots[index2].data, index2);
+             for(SensorStatsParams_t stats_index = 0;
    stats_index < STATS_NAMES_COUNT; stats_index++){
+                  snprintf(pub_topic, WIDTH_64,
    IO_USERNAME"/feeds/%s/%s",\
+
    sm_get_sensor_path_by_handle(sensor_slots[index2].handle)
    ,\
+                          stats_names[stats_index]\
+                           );
+                  snprintf((char*)pub_message, WIDTH_64,
    "%ld.%02ld",\
+
    stats_values[index2][stats_index]/100, \
+
    stats_values[index2][stats_index]%100);
                  mqtt_onchip_da16xxx_pub_info_t pubData;
                  pubData.qos = MQTT_ONCHIP_DA16XXX_QOS_0,
                  pubData.p_topic_name = pub_topic;
                  pubData.topic_name_Length =
    (uint16_t)strlen(pub_topic);
                  pubData.p_payload = pub_message;
                  pubData.payload_length =
    strlen(pubData.p_payload);
                  log_debug("Topic: %s data:
    %s",pub_topic, pub_message);
                  /* Publish data to the MQTT Broker */
```

```
                    if (FSP_SUCCESS !=
    RM_MQTT_DA16XXX_Publish(&g_rm_mqtt_onchip_da16xxx_instanc
    e, &pubData)) {
                        log_error("MQTT publish failed");
                        utils_halt_func();
                    }
                    // The following delay is very important
    as RM_MQTT_DA16XXX_Publish do not send the data right
    away
                    // and without a delay further publishes
    will silently fail
-                   vTaskDelay(100);
+                   vTaskDelay(200);
+               }
     .
     .
     }
}
```

Code Block 12 main_thread_entry.c changes

Expected output:



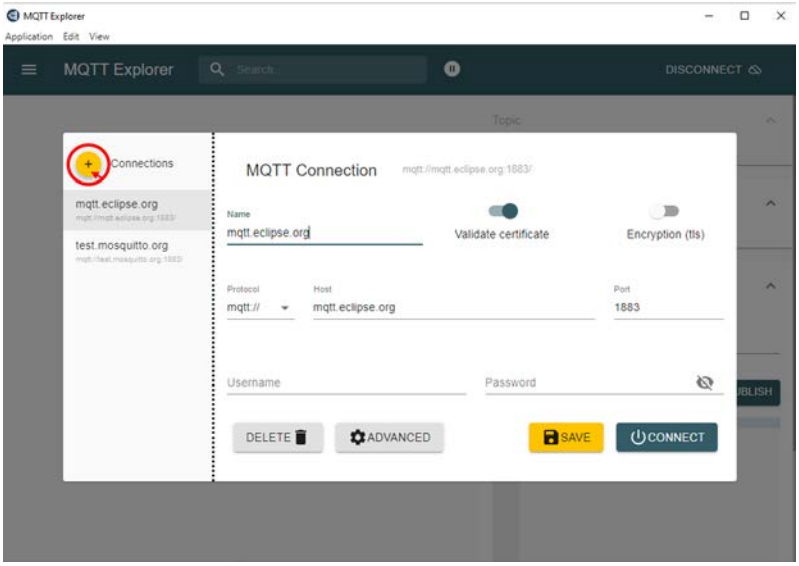
RTT Viewer

```
00> INFO:../src/sensor/hs4001_sensor.c Sensor channel 0 open success
00> INFO:../src/sensor/hs4001_sensor.c Sensor channel 1 open success
00> DEBUG:../src/sensor/hs4001_sensor.c Sensor read channel 0
00> INFO:../src/main_thread_entry.c Starting Wi-Fi connection...
00> INFO:../src/main_thread_entry.c Wi-Fi connection successful!
00> INFO:../src/main_thread_entry.c MQTT setup successful!
00> INFO:../src/main_thread_entry.c MQTT Subscribe to user_device/feeds/led
00> DEBUG:../src/sensor/hs4001_sensor.c Sensor read channel 0
00> DEBUG:../src/sensor/hs4001_sensor.c Sensor read channel 1
00> DEBUG:../src/main_thread_entry.c Topic:<user_device>/feeds/humidity/value data: 56.86
00> DEBUG:../src/main_thread_entry.c Topic:<user_device>/feeds/humidity/minimum data: 56.86
00> DEBUG:../src/main_thread_entry.c Topic:<user_device>/feeds/humidity/maximum data: 57.35
00> DEBUG:../src/main_thread_entry.c Topic:<user_device>/feeds/humidity/average data: 57.17
00> DEBUG:../src/main_thread_entry.c Topic:<user_device>/feeds/humidity/absolute_change data: 0.27
00> DEBUG:../src/main_thread_entry.c Topic:<user_device>/feeds/temperature/value data: 34.33
00> DEBUG:../src/main_thread_entry.c Topic:<user_device>/feeds/temperature/minimum data: 34.31
00> DEBUG:../src/main_thread_entry.c Topic:<user_device>/feeds/temperature/maximum data: 34.33
00> DEBUG:../src/main_thread_entry.c Topic:<user_device>/feeds/temperature/average data: 34.32
00> DEBUG:../src/main_thread_entry.c Topic:<user_device>/feeds/temperature/absolute_change data: 0.02
```

10.3.3.5 Add Widget (Gauge / Box)

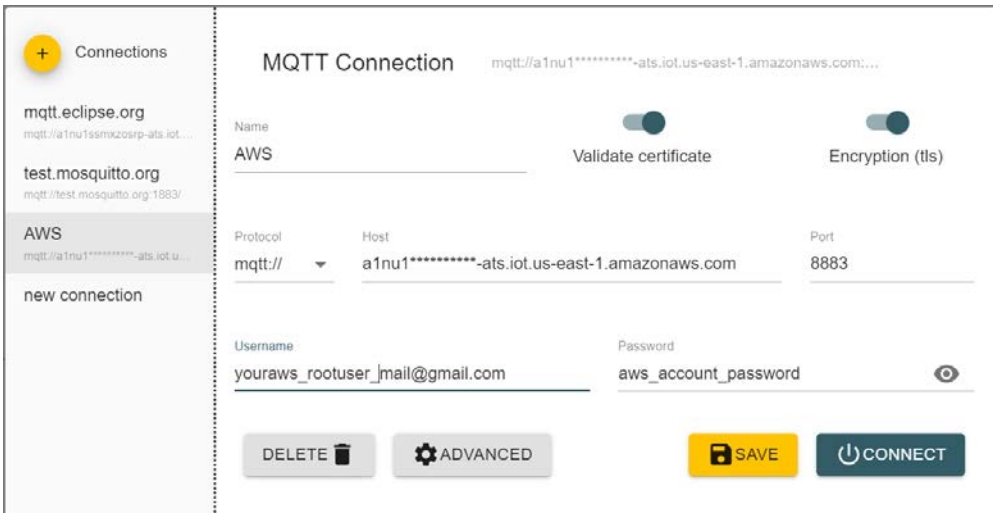
QCS can only publish the sensor data to the AWS cloud. It does not display like a widget. This can be implemented by installing a third-Party dashboard like AWS cloud watch, Grafana, Node-RED, or AWS Amplify with IoT Dashboard (Custom UI). For ease of exercise, use a direct third-party executable MQTT explorer to show the sensor value in widget and graph. Follow these steps to configure MQTT explorer.

1. Open MQTT Explorer
   a. Launch the **.exe** file and click **+** to add a new connection.



2. Set the Connection Settings and input the basic information:
   a. Name: AWS IoT
   b. Protocol: mqtts:// (use secure MQTT over TLS)

c. Host: Enter the AWS IoT endpoint, for example: a3k7example8-ats.iot.us-west-2.amazonaws.com
d. Port: 8883 (TLS secured MQTT)

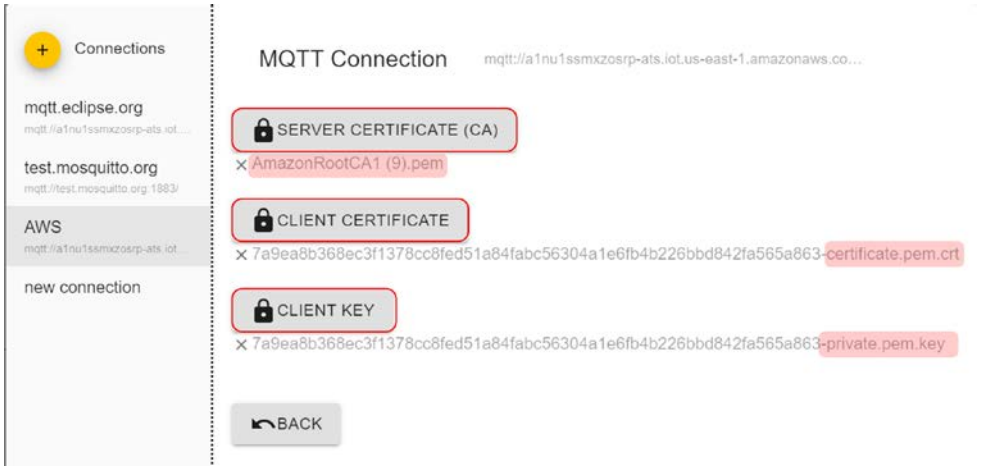


3. Enable TLS/SSL
   a. Toggle Enable TLS

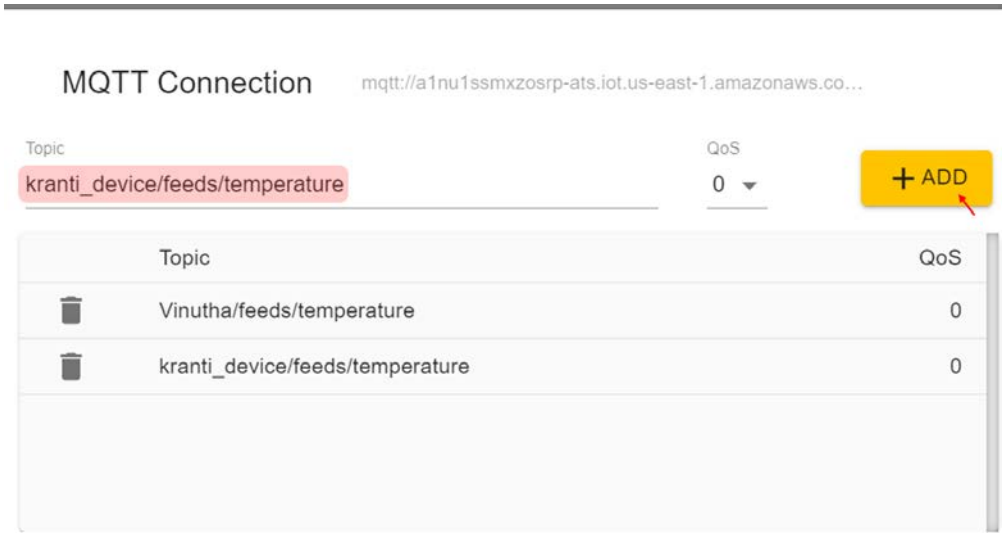
   b. Click on **Advanced**.

   c. Click on **Certificates**.

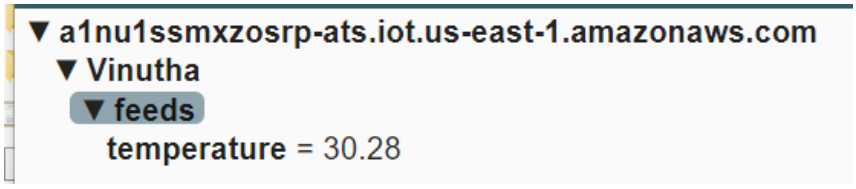
4. Upload the certificate files:
   a. Client Certificate: The device-certificate.pem.crt
   b. Client Key: The private.pem.key
   c. CA Certificate: Amazon root CA (for example, AmazonRootCA1.pem)
   Download CA Cert here



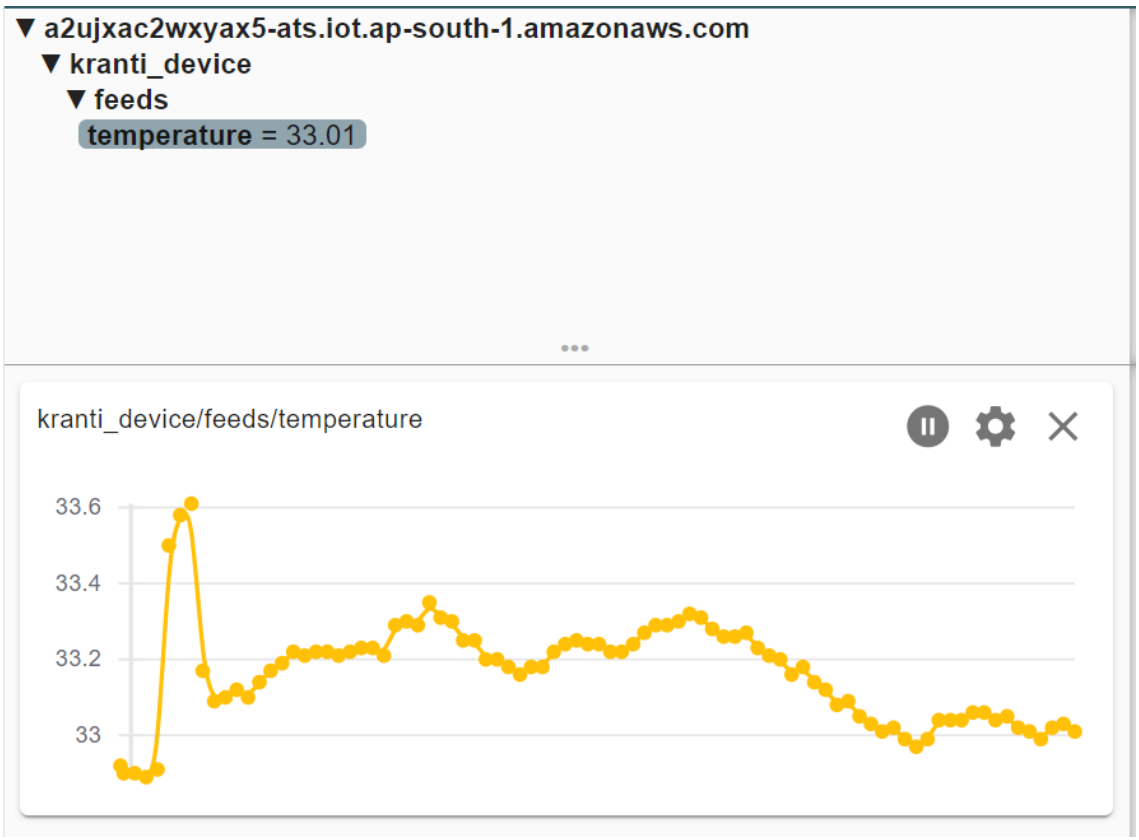
5. Client ID
   a. Optional, but recommended: set a unique client ID, for example, **mqtt-explorer-client**.
6. Click Back and Enter topic
   a. **Retain** or **Clean Session** can be enabled if required.
   b. Add Topic name (for example, <user_device>/feeds/temperature)
   c. Go back and click **Save**.



7. Connect!
   a. Click Connect.
   b. If everything is set up right, the MQTT topics appear and start to populate. (Wait for updated interval period that the user has chosen.)
8. MQTT Explorer
   a. Expand the Username > feeds, to see the real-time value.



   b. Click on **History** to see the history and add charts to observe the real-time values graphically.



Note: <user_device> is replaced by IO_USERNAME from the src/user.h in all output logs.

10.3.4 Customize Sensor Data to BLE Application

As a first step, the user must create a project in the QuickConnect Studio by selecting the appropriate MCU kit (BGK-RA6E2), sensor (HS4001 Temperature / Humidity), connectivity module (DA14531 Bluetooth) and Sensor Data over BLE FreeRTOS application.

10.3.4.1 Change Temperature Sensor Units (C to F and F to C)

This exercise is to update the temperature units between Celsius (C) and Fahrenheit (F). The default unit for temperature on QCS is Celsius. This exercise will convert the value to Fahrenheit and display it on the QC sandbox mobile application. Since the QC sandbox application needs to display the correctly converted data as well as the correct unit, Both changes need to be made to achieve the correct result.

Complete the following steps:

1. Generate the Sensor Data over BLE application on QCS using BGK-RA6E2, HS4001 Sensor, and DA14531 BLE module for FreeRTOS.
2. Modify the **ble_app.c** and **gui_cfg.json** files as mentioned above.
3. Rebuild the project by pressing the **Build QCStudio Project** option in QCS.
4. Download the ***.srec** file from the **Debug/** path and flash the BGK-RA6E2 board.
5. Pair the QC Sandbox mobile application with the board and verify the data.

Files modified:

- **/ src / qe_gen / ble / ble_app.c**
- **/ gui_cfg.json**

Modifications:

ble_app.c: The ble_app_run() function performs the BLE stack execution and obtains the data from FreeRTOS queue whenever it is available. This data is pushed upon receiving the request from mobile application. This function must be modified to convert the temperature value to Fahrenheit. Because this function is generic and is called for both temperature and humidity, the conversion should only take place in the case of temperature. The modification is made to convert only temperature data. (The complete function is not included below, the **+** symbol indicates you must add those lines inside the original function for the **ble_app.c** file).

```
void ble_app_run(void) {
    /* Process BLE Event */
    R_BLE_Execute();
    .
    .
    .
    if (pdTRUE == xQueueReceive(g_sensor_queue, &sensor_data, 0)) {
        uint32_t index = 2;
        while (index < (uint32_t)sm_get_total_sensor_count()+2) {
            if (qc_sv_req_handlers[index].sensor_handler.value ==
sensor_data.handle.value) {
                sm_scaling scaling;
                sm_get_sensor_scaling(sensor_data.handle, &scaling);
                float floatData = ((float)sensor_data.data *
(float)scaling.multiplier)/(float)scaling.divider + (float)scaling.offset;

+               if
(sm_get_sensor_type_by_handle(qc_sv_req_handlers[index].sensor_handler) == TEMPERATURE)
+               {
+                   floatData = ((floatData * 18) / 10) + 32;
+               }

                sensor_data_array[index-2] = floatData;
                break;
            }
            index++;
        }
    }
}
```

Code Block 13 ble_app.c

gui_cfg.json: Parameter units for TEMPERATURE indicates the unit that should be shown against the value in mobile application. Default value for this field is C indicating the value is in Celsius, which should be modified to F because the application is pushing temperature data in Fahrenheit.

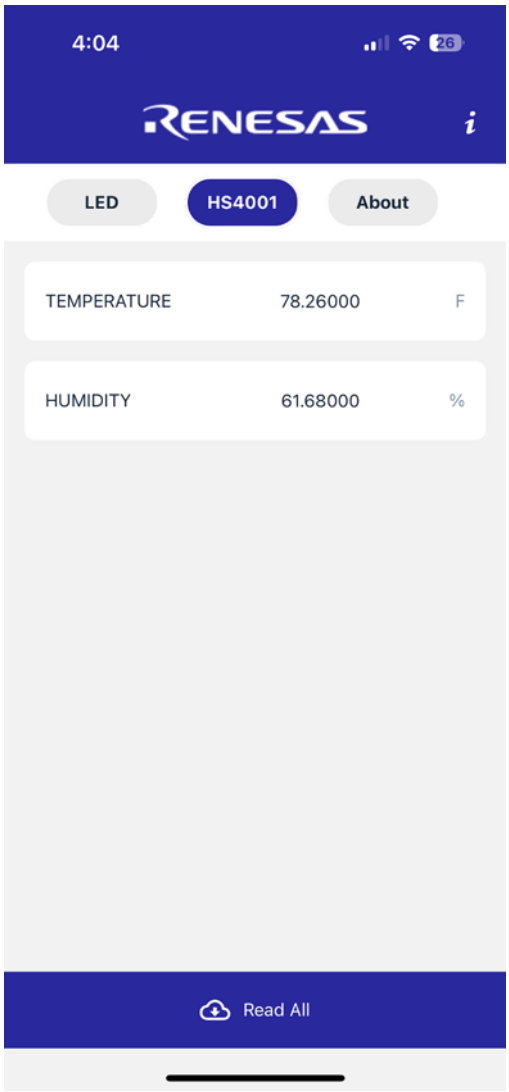
```
{
    "gui":
    {
        "id": "0000",
        "title": "US000 QC Example",
        "version:": "0.0.1",
        "tab":
         [
            {
                "id": "0001",
                "name": "LED",
                "toggle":
                {
                    "id": "0100",
                    "type": "uint8_t",
                    "name": "LED 1",
                    "default": 0
                }
            },
            {
                "id": "0002",
                "name": "HS4001",
                "parameters":
                {
                    "id": "0200",
                    "type": "blob",
                    "access": "r",
                    "list":
                     [
                        {
                            "id": "0201",
                            "name": "TEMPERATURE",
                            "units": "F",
                            "type": "float",
                            "access": "r",
                            "auto_read": 1,
                            "value": null
                        },
                        {
                            "id": "0202",
                            "name": "HUMIDITY",
                            "units": "%",
                            "type": "float",
                            "access": "r",
                            "auto_read": 1,
                            "value": null
                        }
                    ]
                }
            },
            {
                "id": "0003",
                "name": "About",
                "parameters":
                {
                    "id": "0300",
                    "type": "blob",
                    "access": "r",
                    "list":
                    [
                        {
                            "id": "0301",
                            "name": "Test Version",
                            "type": "string",
                            "access": "const",
                            "value": null
```

```
                        }
                    ]
                }
            }
        ]
    }
}
```

Code Block 14 gui_cfg.json

Outcome for this exercise:

After flashing the updated application into BGK-RA6E2 board, open the QC Sandbox mobile application and navigate to HS4001 page. There the user can see the temperature value in Fahrenheit format and the updated unit as well.



10.3.4.2 Change Frequency of Sensor Value Update

This exercise involves updating the time interval at which data is read from the sensor. It adds a custom time interval value for sensor reads, which is reflected on the QC Sandbox mobile application. This update does not modify the interval at which the mobile application requests data from the board.

Complete the following steps:

1. Generate the Sensor Data over BLE application on QCS using BGK-RA6E2, HS4001 Sensor, and DA14531 BLE module for FreeRTOS.
2. Modify the **sensor_thread_entry.c** files as mentioned above.
3. Rebuild the project by pressing the **Build QCStudio Project** option in QCS.
4. Download the ***.srec** file from **Debug/** path and flash the BGK-RA6E2 board.
5. Pair the QC Sandbox mobile application with the board and verify the data.

Files modified:

- **/ src / sensor_thread_entry.c**

Modification:

Sensor thread is responsible for the sensor related activities inside this application. That is initializing the sensor manager and reading and parsing the sensor data. To introduce the custom time interval for a sensor data read, create a static function **static void sensor_configure_interval(uint32_t interval)** that accepts the time interval value as parameter and configures the same as a sensor acquisition interval.

```
static void sensor_configure_interval(uint32_t interval)
{
    sm_handle handle;
    uint16_t index;

    sm_get_sensor_handle(SENSOR_ANY_TYPE, &handle, &index);
    sm_set_sensor_attribute(handle, SM_ACQUISITION_INTERVAL, interval);
}
```

Code Block 15 sensor_thread_entry.c

After this is added, modify the existing **void sensor_thread_entry(void *pvParameters)** by calling the above function with the required time interval value. This way, the sensor manager can be configured with the time interval between sensor reads. Here the function is called with 6000, which means the sensor acquisition interval is 6 seconds. The tested maximum supported value is 8 seconds.

```
/* Sensor entry function */
/* pvParameters contains TaskHandle_t */
void sensor_thread_entry(void *pvParameters) {
    FSP_PARAMETER_NOT_USED (pvParameters);

    sensor_configure_interval(6000);

    /* TODO: add your own code here */
    sm_init();

    while (1) {
        sm_run();
    }
}
```

Code Block 16 sensor_thread_entry.c

Outcome for this exercise:

After flashing the .srec file, connect the board with QC Sandbox mobile application over BLE and navigate to the HS4001 page. There, the data update should be according to the configured time interval only. *Note*: The mobile application has separate update interval. The observation is linked with that parameter and, that is, if the sensor update interval is configured for a lesser value than mobile app update, the user does not see any visible change after this exercise.

10.3.4.3 Add Widget for Sensor Value Visualization (Gauges / Box)

This exercise is on adding extra widget on the mobile application for the sensor data visualization. Default configuration for the mobile application is box widget in which the sensor data will be displayed. Additional configuration must be performed on the application to make the gauge widget available on the mobile application.

Complete the following steps:

1. Generate the **Sensor Data over BLE** application on QCS using the BGK-RA6E2, HS4001 Sensor, and DA14531 BLE module for FreeRTOS.
2. Modify the **gui_cfg.json** file as previously mentioned.
3. Rebuild the project by pressing the **Build QCStudio Project** option in QCS.
4. Download the ***.srec** file from the **Debug/** path and flash the BGK-RA6E2 board.
5. Pair the QC Sandbox mobile application with the board and verify the data.

Files modified:

- **/ gui_cfg.json**

Modifications:

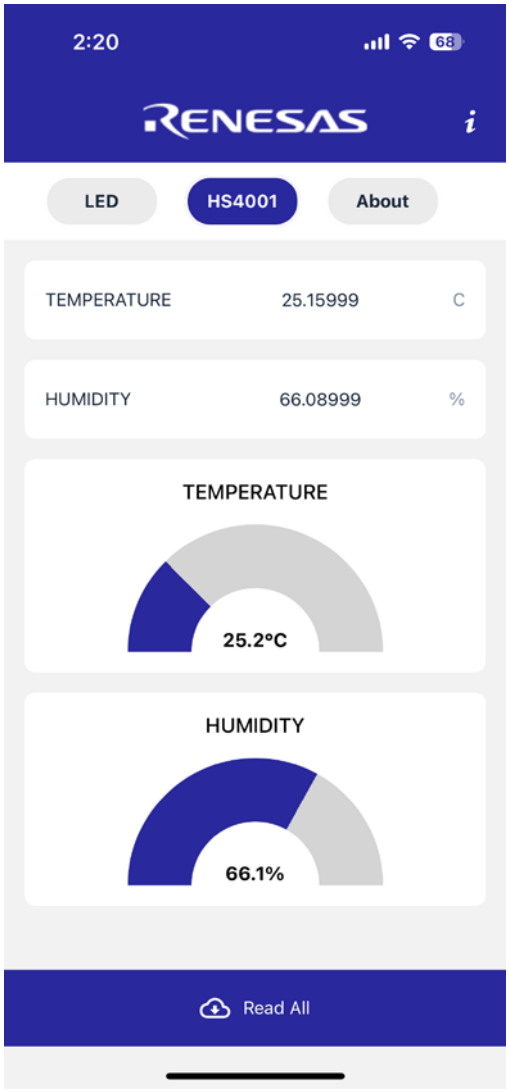
```
{
    "gui": {
    "id": "0000",
    "title": "US000 QC Example",
    "version:": "0.0.1",
    "tab": [
    {
        "id": "0001",
        "name": "LED",
        "toggle": {
            "id": "0100",
            "type": "uint8_t",
            "name": "LED 1",
            "default": 0
        }
    },
    {
    "id": "0002",
    "name": "HS4001",
    "parameters": {
        "id": "0200",
        "type": "blob",
        "access": "r",
        "list": [
        {
            "id": "0201",
            "name": "TEMPERATURE",
            "units": "C",
            "type": "float",
            "access": "r",
            "auto_read": 1,
            "value": null
        },
        {
            "id": "0202",
            "name": "HUMIDITY",
            "units": "%",
            "type": "float",
            "access": "r",
            "auto_read": 1,
            "value": null
        }
        ]
        },
        "gauge": [
        {
            "id": "0201",
            "name": "TEMPERATURE",
            "units": "C",
            "type": "float",
            "minimum": 0,
            "maximum": 100,
            "showinmax": true,
            "default": 0,
            "value": null
        },
        {
            "id": "0202",
            "name": "HUMIDITY",
            "units": "%",
            "type": "float",
            "minimum": 0,
            "maximum": 100,
            "showinmax": true,
            "default": 0,
            "value": null
```

```
            }
            ]
        },
        {
            "id": "0003",
            "name": "About",
            "parameters": {
                "id": "0300",
                "type": "blob",
                "access": "r",
                "list": [
                {
                    "id": "0301",
                    "name": "Test Version",
                    "type": "string",
                    "access": "const",
                    "value": null
                }
                ]
        }
        }
        ]
    }
    }
```

Code Block 17 gui_cfg.json

Outcome for this exercise:

After this change, the QC Sandbox mobile application should show gauge widget as well for the sensor data.



10.3.4.4 Update Frequency of Sensor Data Update on the BLE Mobile Application

This exercise is to update the time interval at which data is read by the QC Sandbox mobile application. This is independent of the time interval at which the sensor data is read by the firmware (discussed on REQ – 2); that is, this exercise defines the rate at which function **handle_read_data()** is invoked inside the application.

Complete the following steps:

1. Generate Sensor Data over BLE application on QCS using BGK-RA6E2, HS4001 Sensor and DA14531 BLE module for FreeRTOS.
2. Modify the **gui_cfg.json** file as mentioned above.
3. Rebuild the project by pressing the **Build QCStudio Project** option in QCS.
4. Download the ***.srec** file from Debug/ path and flash the BGK-RA6E2 board.
5. Pair QC Sandbox mobile application with the board and verify the data.

Files to be modified:

- **/ gui_cfg.json**

Modifications:

The auto_read parameter in the **gui_cfg.json** file mentions the time interval (in seconds) at which the request comes from mobile application. Value of 1 (Default) in this field indicates for every one second, the request is made, and the sensor data is collected.

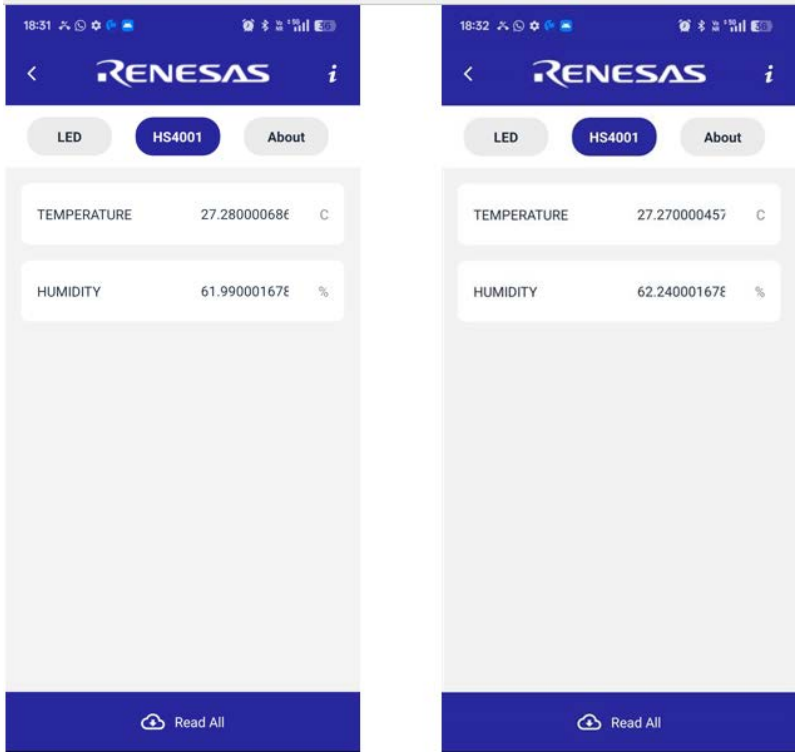
```
{
    "gui": {
        "id": "0000",
        "title": "US000 QC Example",
        "version:": "0.0.1",
        "tab": [
        {
            "id": "0001",
            "name": "LED",
            "toggle": {
                "id": "0100",
                "type": "uint8_t",
                "name": "LED 1",
                "default": 0
            }
        },
        {
            "id": "0002",
            "name": "HS4001",
            "parameters": {
                "id": "0200",
                "type": "blob",
                "access": "r",
                "list": [
                {
                    "id": "0201",
                    "name": "TEMPERATURE",
                    "units": "C",
                    "type": "float",
                    "access": "r",
                    "auto_read": 5,
                    "value": null
                },
                {
                    "id": "0202",
                    "name": "HUMIDITY",
                    "units": "%",
                    "type": "float",
                    "access": "r",
                    "auto_read": 5,
                    "value": null
                }
                ]
            }
        },
        {
            "id": "0003",
            "name": "About",
            "parameters": {
                "id": "0300",
                "type": "blob",
                "access": "r",
                "list": [
                {
                    "id": "0301",
                    "name": "Test Version",
                    "type": "string",
                    "access": "const",
                    "value": null
                }
                ]
            }
        }
        ]
    }
}
```

Code Block 18 gui_cfg.json

Here it is configured for a 5s interval and the mobile application is expected to send the read request for sensor data every 5s. After everything is done, the user should see the data update for every 5s inside the mobile application.

Outcome for this exercise:

After flashing the **.srec** file, connect the board with QC Sandbox mobile application over BLE and navigate to the HS4001 page. The user should see the data update according to the configured time interval only.



10.3.4.5 Send the Sensor Value Only if the Read Button Pressed on the BLE Mobile Application

This requirement is to disable the auto_read functionality of the QC Sandbox mobile application. The user is expected to initiate the read process from mobile application to get the updated data.

Complete the following steps:

1. Generate Sensor Data over BLE application on QCS using the BGK-RA6E2, HS4001 Sensor, and DA14531 BLE module for FreeRTOS.
2. Modify the **gui_cfg.json** file as mentioned above.
3. Rebuild the project by pressing the Build QCStudio Project option in QCS.
4. Download the ***.srec** file from **Debug/** path and flash the BGK-RA6E2 board.
5. Pair the QC Sandbox mobile application with the board and verify the data.

Files to be modified:

- **/ gui_cfg.json**

Modifications:

The auto_read parameter in the **gui_cfg.json** file mentions the time interval (in seconds) at which the request comes from the mobile application. Configuring this value to 0 disables the auto read functionality completely and the user must initiate the data read manually.

```
{
    "gui": {
        "id": "0000",
        "title": "US000 QC Example",
        "version:": "0.0.1",
        "tab": [
        {
            "id": "0001",
            "name": "LED",
            "toggle": {
                "id": "0100",
                "type": "uint8_t",
                "name": "LED 1",
                "default": 0
            }
        },
        {
            "id": "0002",
            "name": "HS4001",
            "parameters": {
                "id": "0200",
                "type": "blob",
                "access": "r",
                "list": [
                {
                    "id": "0201",
                    "name": "TEMPERATURE",
                    "units": "C",
                    "type": "float",
                    "access": "r",
                    "auto_read": 0,
                    "value": null
                },
                {
                    "id": "0202",
                    "name": "HUMIDITY",
                    "units": "%",
                    "type": "float",
                    "access": "r",
                    "auto_read": 0,
                    "value": null
                }
                ]
            }
        },
        {
            "id": "0003",
            "name": "About",
            "parameters": {
                "id": "0300",
                "type": "blob",
                "access": "r",
                "list": [
                {
                    "id": "0301",
                    "name": "Test Version",
                    "type": "string",
                    "access": "const",
                    "value": null
                }
                ]
            }
        }
        ]
    }
}
```

Code Block 19 gui_cfg.json

Outcome for this exercise:

After flashing the **.srec** file, connect the board with QC Sandbox mobile application over BLE and navigate to the HS4001 page. The user should not see the automatic update for the sensor data. The sensor data should get updated only after pressing the **Read All** button.

10.3.4.6 Using the BLE Mobile Application, Toggle the LED which is Currently ON and OFF

This exercise is to update the onboard LED response for the mobile application action. After this exercise, the LED should toggle for each LED button press on the mobile application. Currently, the button is a simple slider button that can be either at OFF position or ON position. This needs to be modified into a simple button so that with each press, the LED toggle can be visible.

Complete the following steps:

1. Generate the **Sensor Data over BLE** application on QCS using BGK-RA6E2, HS4001 Sensor, and DA14531 BLE module for FreeRTOS.
2. Modify the **gui_cfg.json** and **ble_app.c** files as mentioned above.
3. Rebuild the project by pressing the Build QCStudio Project option in QCS.
4. Download the ***.srec** file from **Debug/** path and flash the BGK-RA6E2 board.
5. Pair the QC Sandbox mobile application with the board and verify the data.

Files to be modified:

- **/ src / qe_gen / ble / ble_app.c**
- **/ gui_cfg.json**

Modifications:

ble_app.c: The handle_write_led() function is responsible for defining the LED state based on requests from the mobile application and data received through BLE. The default code turns OFF the LED if the data received over BLE is 0x00; otherwise, it turns the LED on. This function should be modified to toggle the LED state whenever the handler is invoked.

```
static void handle_write_led(uint16_t id, uint8_t const * const data)
{
    FSP_PARAMETER_NOT_USED(id);
    static bool state = false;

    state = !state;

    if (state == false)
    {
        utils_set_LED(BLUE_LED,BSP_IO_LEVEL_LOW);
    }
    else
    {
        utils_set_LED(BLUE_LED,BSP_IO_LEVEL_HIGH);
    }
}
```

Code Block 20 ble_app.c

gui_cfg.json: This file should be modified to render button for the LED instead of ON and OFF switch.

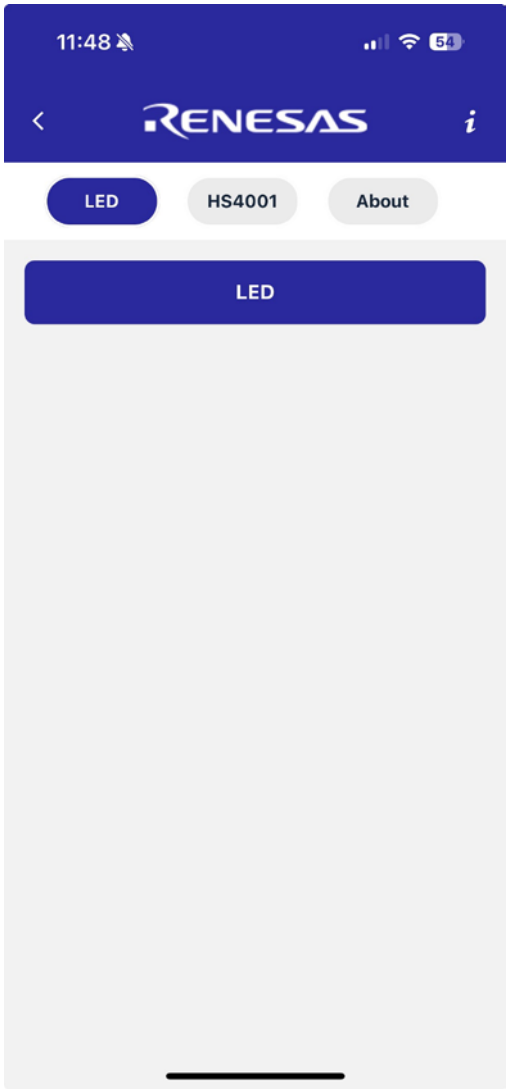
```
{
    "gui":
    {
        "id": "0000",
        "title": "US000 QC Example",
        "version:": "0.0.1",
        "tab":
        [
            {
                "id": "0001",
                "name": "LED",
                "button":
                {
                    "id": "0100",
                    "name": "LED",
                    "ack": false,
                    "timeout": 0,
                    "read": false
                }
            },
            {
                "id": "0002",
                "name": "HS4001",
                "parameters":
                {
                    "id": "0200",
                    "type": "blob",
                    "access": "r",
                    "list":
                    [
                        {
                            "id": "0201",
                            "name": "TEMPERATURE",
                            "units": "C",
                            "type": "float",
                            "access": "r",
                            "auto_read": 1,
                            "value": null
                        },
                        {
                            "id": "0202",
                            "name": "HUMIDITY",
                            "units": "%",
                            "type": "float",
                            "access": "r",
                            "auto_read": 1,
                            "value": null
                        }
                    ]
                }
            },
            {
                "id": "0003",
                "name": "About",
                "parameters":
                {
                    "id": "0300",
                    "type": "blob",
                    "access": "r",
                    "list":
                    [
                        {
                            "id": "0301",
                            "name": "Test Version",
                            "type": "string",
                            "access": "const",
```

```
                        "value": null
                    }
                ]
            }
        }
    ]
}
}
```

Code Block 21 gui_cfg.json

Outcome for this exercise:

The modifications made to **gui_cfg.json** file render buttons on the LED page of QC sandbox application. When ***.srec** is flashed on the BGK-RA6E2 and paired with the QC Sandbox mobile application, on the LED page, button is rendered as follows.



Press this button and the user can see the on-board LED (LED2) toggling for each button press.

11. References

- RA6E2 - Entry-Line 200MHz Arm® Cortex®-M33 General Purpose Microcontroller | Renesas
- DA16600MOD - Ultra-Low Power Wi-Fi + Bluetooth® Low Energy Combo Modules for Battery Powered IoT Devices | Renesas
- HS4001 - Relative Humidity and Temperature Sensor, Digital Output, ±1.5% RH | Renesas
- US159-DA14531EVZ - Low Power Bluetooth PMOD Board (Renesas QuickConnect IoT)

Technical Updates/Technical News

- The latest information can be downloaded from the Renesas Electronics Website.

Website and Support

Renesas Electronics Website - https://www.renesas.com/

Inquiries **-** https://www.renesas.com/contact/

12. Ordering Information

Part Number	Description
QC-BEKITPOC2Z	QuickConnect Beginners Kit V2.0

13. Revision History

Revision	Date	Description
1.01	Feb 20, 2026	Updated the following sections: ▪ Prereequisites ▪ Project 1, Project 2 , and Project 3 ▪ Debugging on QuickConnect Studio ▪ Customize Blinky Application
1.00	May 8, 2025	Initial release.

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.

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD-PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers who are designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01)

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-us/.

Trademarks

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