

Getting Started with the Renesas RZ/A2M Evaluation Board Kit

Required Resources

To build and run the RZ/A2M Evaluation Board Kit example, you will need following resources:

Development tools & software

- e²studio IDE v7.6.0 ([e²studio download](#))
- GNU ARM Embedded 6-2016q2-update (bundled in e²studio v7.6.0)

Hardware

- Renesas RZ/A2M Evaluation Board Kit, P/N: RTK7921053S00000BE#WS (<https://www.renesas.com/products/software-tools/boards-and-kits/eval-kits/rz-a2m-evaluation-board-kit.html>)
- PC running Windows 7 or 10; the Tera Term console, or similar application; and an installed web browser (Google Chrome, Internet Explorer, Microsoft Edge, or Mozilla Firefox).
- Ethernet LAN internet access

Before you begin, see [Prerequisites](#).

If you do not have an RZ/A2M Evaluation Board Kit, you can order one from [Renesas](#).

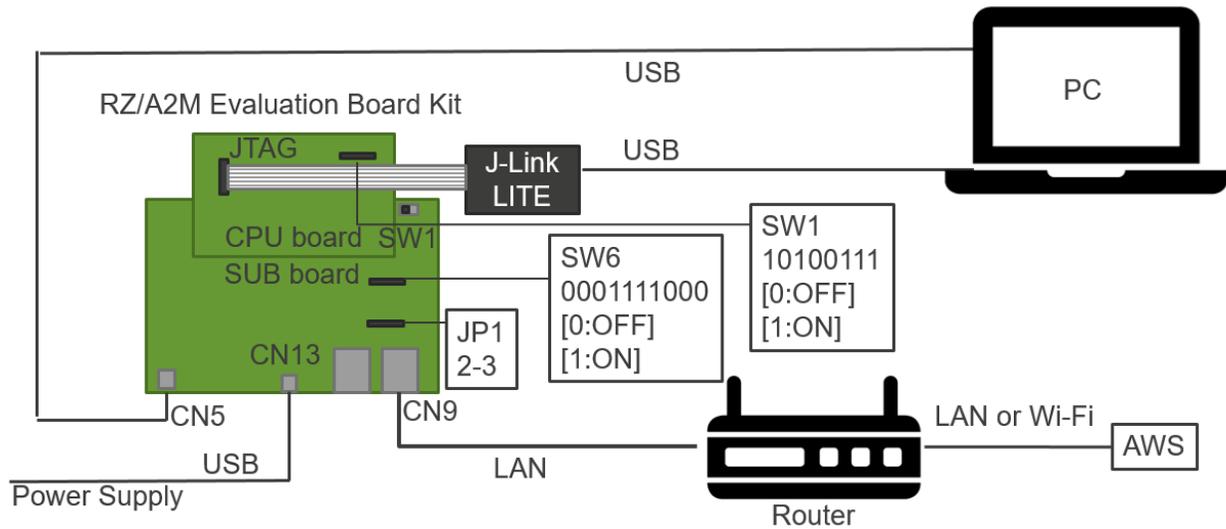
Setting Up Your Environment

FreeRTOS for the RZ/A2M Evaluation Board Kit uses e²studio IDE and GNU ARM Embedded compiler. Before you begin, install the IDE and compiler to your machine:

To install e²studio:

1. Browse to [e²studio](#) and choose **Download Software**. **Make sure to use** e²studio version 7.6.0 or later.
2. Unzip and run the installer. Follow the prompts for the section 2.1 and 2.2 of the [e²studio Getting Started Guide](#).

Connecting a Debugger



1. Confirm Power switch (SW1) on SUB board is ON (left).
2. Connect CPU board and SUB board.
3. Set SW1 on CPU board to 10100111.
4. Set SW6 on SUB board to 0001111000.
5. Connect pin 2 and pin 3 of JP1 on SUB board.
6. Connect J-Link LITE to JTAG connector on CPU board.
7. Connect USB cable from J-Link LITE to a spare USB port on your PC.
8. Connect USB cable from CN5 on SUB board to a spare USB port on your PC.
9. Connect LAN cable from CN9 on SUB board to a router can access AWS.

Download and Build FreeRTOS

After your environment is set up, you can download 'Renesas RZ/A2M Evaluation Board Kit Application Example' and run the demo code.

Download FreeRTOS

1. Browse to the [GitHub Page](#) and download the code.
2. Unzip the downloaded file to a folder and make a note of the folder path. In this tutorial, this folder is referred to as BASE_FOLDER.

Note: The e²studio doesn't support long path names. To accommodate the files in the FreeRTOS projects, make sure the path to the directory is less than 260 characters and does not contain spaces or special characters.

Import the FreeRTOS Demo Code into Your IDE

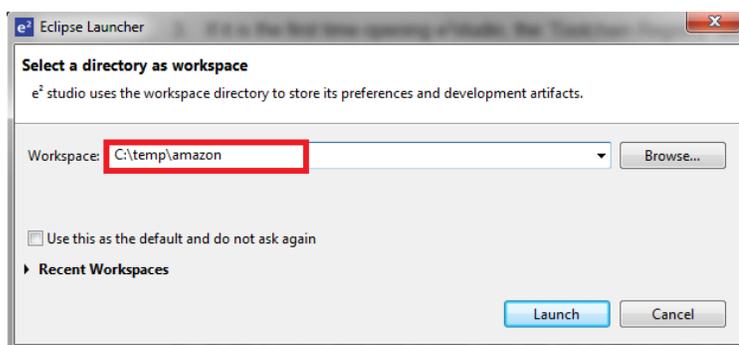
To import the FreeRTOS demo code into e²studio IDE

1. e²studio integrates various tools such as compiler, an assembler, debugger and an editor into a common graphical user interface. Start e²studio:

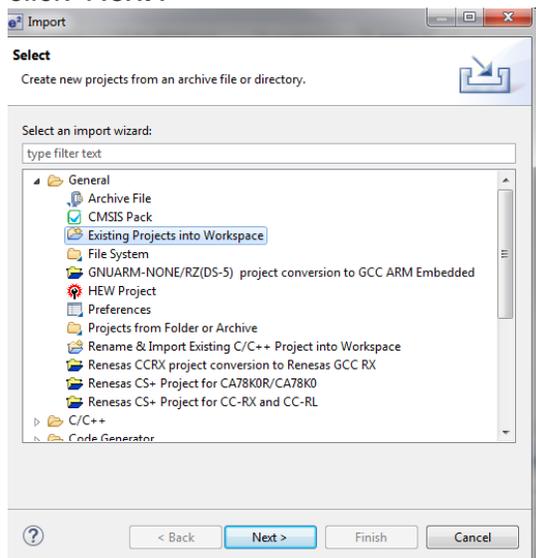
Windows™ 7: Start Menu>All Programs>Renesas Electronics e2studio>e2studio

Windows™ 10: Start Menu>All Apps> Renesas Electronics e2studio>e2studio

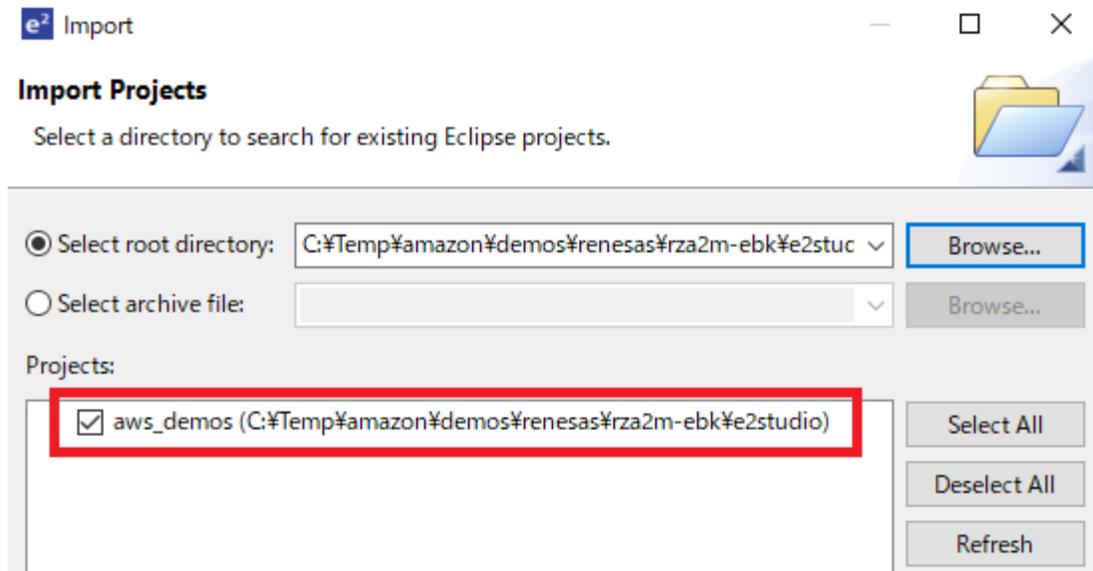
2. In the 'Select a workspace' folder that appears, browse to the folder "...BASE_FOLDER\amazon". Click 'OK' to continue.



3. If it is the first time opening e²studio, the 'Toolchain Registry' window will open. In the 'Toolchain Registry' dialog select GCC ARM Embedded and ensure that '6.3.1.20170620' is selected. Click 'Register'. A dialog will appear "Selected Toolchains were successfully integrated with e²studio ". Click 'OK'.
4. In the 'Code Generator Registration' dialog click 'OK'. This window opens up first time only after installation.
5. A 'Code Generator COM component register' dialog will pop-up with the text "Please restart e²studio to use Code Generator". Click 'OK'.
6. In the 'Restart e²studio dialog, click 'OK'.
7. Once e²studio is restarted, then 'Select a workspace' window appears again with the folder path selected in step 2. Click 'OK'.
8. In the e²studio welcome screen, click 'Go to the e²studio workbench' arrow icon, on the far right.
9. Right click in the Project Explorer window, and select 'Import'.
10. In the import wizard, select General > Existing Projects into Workspace, and click 'Next'.



11. Click the 'Browse' button, and locate the following directory '`<BASE_FOLDER>\amazon\demos\renesas\rza2m-ebk\e2studio'`.



12. Click "Finish".
13. In the **Project** menu, choose **Project->Build All**. The project should build with no errors.

Configure Your Project

To configure your project, you need to know your AWS IoT endpoint and Thing name that represents your board.

Configure AWS IoT endpoint

1. Login to aws account and Click on [IoT Core](#) services.
2. In the left navigation pane, choose **Settings**.
3. Copy your AWS IoT endpoint from the **Endpoint** text box. It should look like `<1234567890123>.iot.<us-east-1>.amazonaws.com`.
4. Open `aws_demos/application_code/common_demos/include/aws_clientcredential.h` and set `clientcredentialMQTT_BROKER_ENDPOINT` to your AWS IoT endpoint.

```
static const char clientcredentialMQTT_BROKER_ENDPOINT[] = "Paste AWS IoT Broker endpoint here.";
```

5. In the left navigation pane, Click on Manage-> Things, and then Click on 'Create' to create a new Thing.
6. In the next window, click on "Create a single thing".

Creating AWS IoT things

An IoT thing is a representation and record of your physical device in the cloud. Any physical device needs a thing record in order to work with AWS IoT. [Learn more](#).

Register a single AWS IoT thing
Create a thing in your registry

Create a single thing

Bulk register many AWS IoT things
Create things in your registry for a large number of devices already using AWS IoT, or register devices so they are ready to connect to AWS IoT.

Create many things

7. Enter thing Name for your IoT board.

CREATE A THING
Add your device to the thing registry

This step creates an entry in the thing registry and a thing shadow for your device.

Name

- Open `aws_demos\application_code\common_demos\include\aws_clientcredential.h`. Specify AWS IoT thing for your board in the following `#define` constants from **Thing** pane in [AWS IoT console](#).

```
#define clientcredentialIOT_THING_NAME "Paste AWS IoT Thing name here."
```

- Click next. In next window click on "Create Certificate"

CREATE A THING

Add a certificate for your thing

STEP 2/3

A certificate is used to authenticate your device's connection to AWS IoT.

One-click certificate creation (recommended)
This will generate a certificate, public key, and private key using AWS IoT's certificate authority.

[Create certificate](#)

- Download the certificate.

Certificate created!

Download these files and save them in a safe place. Certificates can be retrieved at a after you close this page.

In order to connect a device, you need to download the following:

A certificate for this thing	f96334faa1.cert.pem	Download
A public key	f96334faa1.public.key	Download
A private key	f96334faa1.private.key	Download

You also need to download a root CA for AWS IoT:
A root CA for AWS IoT [Download](#)

[Activate](#)

- Activate the certificate.

Certificate created!

Download these files and save them in a safe place. Certificates can be retrieved at a after you close this page.

In order to connect a device, you need to download the following:

A certificate for this thing	f96334faa1.cert.pem	Download
A public key	f96334faa1.public.key	Download
A private key	f96334faa1.private.key	Download

You also need to download a root CA for AWS IoT:
A root CA for AWS IoT [Download](#)

[Activate](#)

Create AWS IoT policy

1. In the left navigation pane, Click on **Secure-> Policies**, and then Click on "**Create a policy**" or "**Create**" to create a new policy.
2. Enter Policy name for your test



Create a policy to define a set of authorized actions. You can learn more about IoT policies go to the [AWS IoT Policies document](#).

Name

3. Enter **iot:Connect** in Action box, replace `replaceWithAClientId` with **MQTTEcho** in Resource ARN box, check **Allow** in Effect, and click on **Add statement**.

Add statements

Policy statements define the types of actions that can be performed by a resource.

Action	<input type="text" value="iot:Connect"/>
Resource ARN	<input type="text" value="arn:aws:iot:ap-northeast-1:123456789012:client/MQTTEcho"/>
Effect	<input checked="" type="checkbox"/> Allow <input type="checkbox"/> Deny

4. Enter **iot:Publish** in Action box, replace `replaceWithATopic` with **freertos/demos/echo** in Resource ARN box, check **Allow** in Effect, and click on **Add statement**.

Action

iot:Publish

Resource ARN

arn:aws:iot:ap-northeast-1: :topic/freertos/demos/echo

Effect

Allow Deny

Add statement

5. Enter **iot:Subscribe** in Action box, replace `replaceWithATopicFilter` with **freertos/demos/echo** in Resource ARN box, check **Allow** in Effect, and click on **Add statement**.

Action

iot:Subscribe

Resource ARN

arn:aws:iot:ap-northeast-1: :topicfilter/freertos/demos/echo

Effect

Allow Deny

Add statement

6. Enter **iot:Receive** in Action box, replace `replaceWithATopic` with **freertos/demos/echo** in Resource ARN box, check **Allow** in Effect, and click on **Create**.

The screenshot shows the configuration of a policy statement in the AWS IAM console. The 'Action' field is set to 'iot:Receive'. The 'Resource ARN' field is set to 'arn:aws:iot:ap-northeast-1:123456789012:topic/freertos/demos/echo'. The 'Effect' field has 'Allow' selected. A 'Create' button is visible at the bottom right.

7. In the left navigation pane, Click on **Secure-> Certificates**, and then Click on certificate created in the sequence 10 of **Configure AWS IoT endpoint** section above.
8. Click on Actions and select **Attach policy**.

The screenshot shows the 'Actions' dropdown menu for a certificate in the AWS IAM console. The 'Attach policy' option is highlighted. The certificate ARN is visible as 'arn:aws:iot:ap-northeast-1:123456789012:cert/91fab...'. The issuer is 'O=Amazon Web Services O=Amazon.com Inc. I=Seattle ST=Washi'.

9. Check the policy you created, then click on **Attach**.

Attach policies to certificate(s)

Policies will be attached to the following certificate(s):
[Blurred text]

Choose one or more policies

Search policies	
<input type="checkbox"/> rz_echo_test	View
<input checked="" type="checkbox"/> test	View

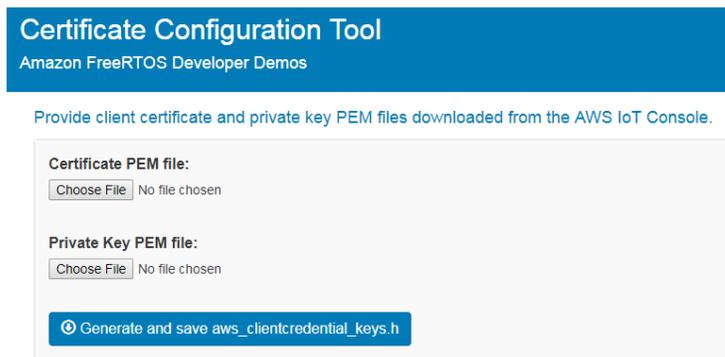
1 policy selected

Configure certificate and private key

The certificate and private key must be hard-coded into the FreeRTOS demo code. This is for demo purposes only. Production level applications should store these files in a secure location. FreeRTOS is a C language project, and the certificate and private key must be specially formatted to be added to the project.

To format your certificate and private key

1. In a browser window, open certificate configuration tool from project `<BASE_FOLDER>\tools\certificate_configuration\CertificateConfigurator.html`.



The screenshot shows the 'Certificate Configuration Tool' interface. At the top, there is a blue header with the text 'Certificate Configuration Tool' and 'Amazon FreeRTOS Developer Demos'. Below the header, a blue instruction bar reads 'Provide client certificate and private key PEM files downloaded from the AWS IoT Console.' The main content area is a light gray box containing two sections: 'Certificate PEM file:' and 'Private Key PEM file:'. Each section has a 'Choose File' button and the text 'No file chosen'. At the bottom of the gray box is a blue button with a circular arrow icon and the text 'Generate and save aws_clientcredential_keys.h'.

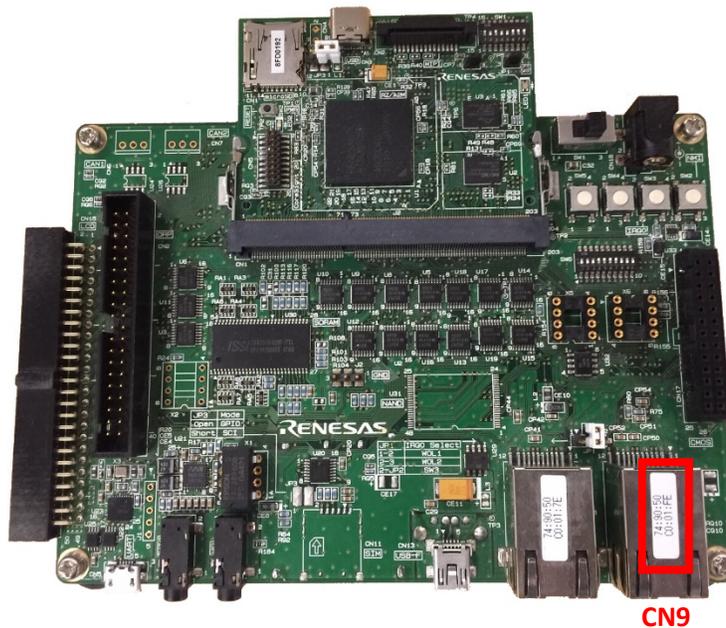
2. Under **Certificate PEM file**, choose `certificate.pem.crt` you downloaded from the AWS IoT console in previous step.
3. Under **Private Key PEM file**, choose `private.pem.key` you downloaded from the AWS IoT console in previous step.
4. Choose **Generate and save aws_clientcredential_keys.h**, and then save the file in `<BASE_FOLDER>\demos\common\include`. This overwrites the file `aws_clientcredential_keys.h` in the directory.

Configure MAC address

MAC address is NOT stored in the storage memory on the board. Therefore, you need to set MAC address to your project.

1. Get your MAC address

You can find your MAC address on CN9 of the board.



2. Set MAC address in your code

Edit `configMAC_ADDRN` (N=0, 1, ... ,5) macros defined in `FreeRTOSConfig.h` to the MAC address printed on CN9. Ethernet driver is configured to use CN9.

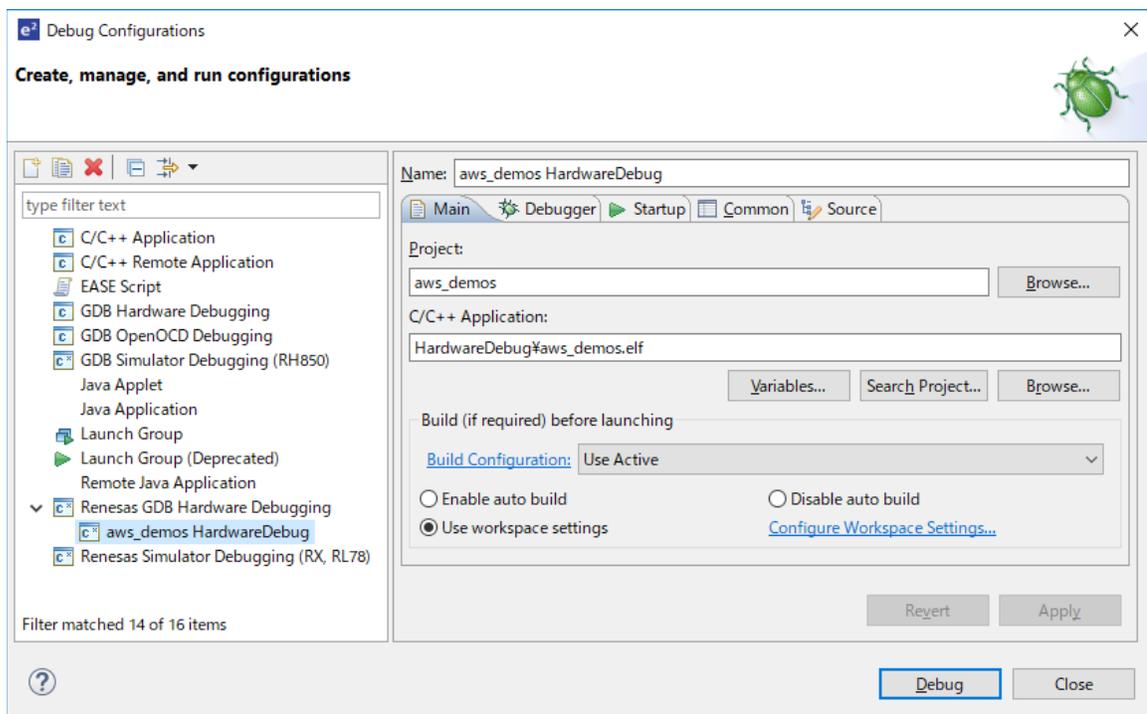
In the case your MAC address is 01:23:45:67:89:AB, set macros as follows:

```
#define configMAC_ADDR0           0x01
#define configMAC_ADDR1           0x23
#define configMAC_ADDR2           0x45
#define configMAC_ADDR3           0x67
#define configMAC_ADDR4           0x89
#define configMAC_ADDR5           0xAB
```

Run the FreeRTOS Demo

To run the FreeRTOS demos on the RZ/A2M Evaluation Board Kit:

1. Sign in to the [AWS IoT console](#).
2. In the left navigation pane, choose **Test** to open the MQTT client.
3. In the **Subscription topic** text box, type **'freertos/demos/echo'**, and then choose **Subscribe to topic**.
4. Rebuild the project, **"Project->Build All"**.
5. Connect USB cable from J-Link LITE to a spare USB port on your PC.
6. Connect USB cable from CN13 on SUB board to a power supply.
7. The debugging can be started by clicking the 'Run-> Debug Configuration'. Click the symbol **"aws_demos HardwareDebug"** under 'Renesas GDB Hardware Debugging' by expanding the list.



8. Click the '**Debug**' button to download the code to the target board to begin debugging. A firewall warning may be displayed for 'e2-server-gdb.exe'. Select the check-box for 'Private networks, such as my home or work network', and click 'Allow access'.
9. e²studio may ask you to change to the 'Renesas Debug Perspective'. Click 'Yes'.
10. Once the code has been downloaded, click the 'Resume' button to run the code up to the first line of the main function. Click 'Resume' button again to run the target through the rest of the code.

In the AWS IOT console MQTT client, you should see the MQTT messages sent by your device.

Note:

Please visit the following GitHub repository to get the latest projects (prototype), but not yet certified for other Renesas devices, compilers, and target boards.

<https://github.com/renesas-rz/amazon-freertos>

Troubleshooting

If no messages appear in the AWS IoT console, try the following:

1. Check that your network credentials are valid.
2. Verify the switch settings on your board.