

# Getting started with the Nuvoton NuMaker-IoT-M487 and BG96

## Document Information

This tutorial provides instructions for getting started with the Nuvoton NuMaker-IoT-M487 / BG96 IoT Node. Instructions are only for the Windows platform (Linux and MacOS are not supported).

- Set up your development environment, installing software on the host machine for developing and debugging embedded applications for your microcontroller board.
- Cross compiling a FreeRTOS demo application to a binary image.
- Loading the application binary image to your board, and then running the application.

### Revision History (Version, Date, Description of change)

1.0	01-Dec-2020	Initial version
-----	-------------	-----------------

## Overview

From the basic end nodes to the gateways and the cloud, IoT applications require control, networking, encryption and other related technologies. Nuvoton introduces the NuMaker-IoT-M487 development board for IoT applications. Embedded with the NuMicro M487 series microcontroller, NuMaker-IoT-M487 development board has built-in RJ45 Ethernet and Wi-Fi modules, which allows users to connect to cloud in a wired or wireless manner. NB-IoT and other connectivity capabilities are provided through different wireless daughter boards to meet various application scenarios.

## Hardware Description

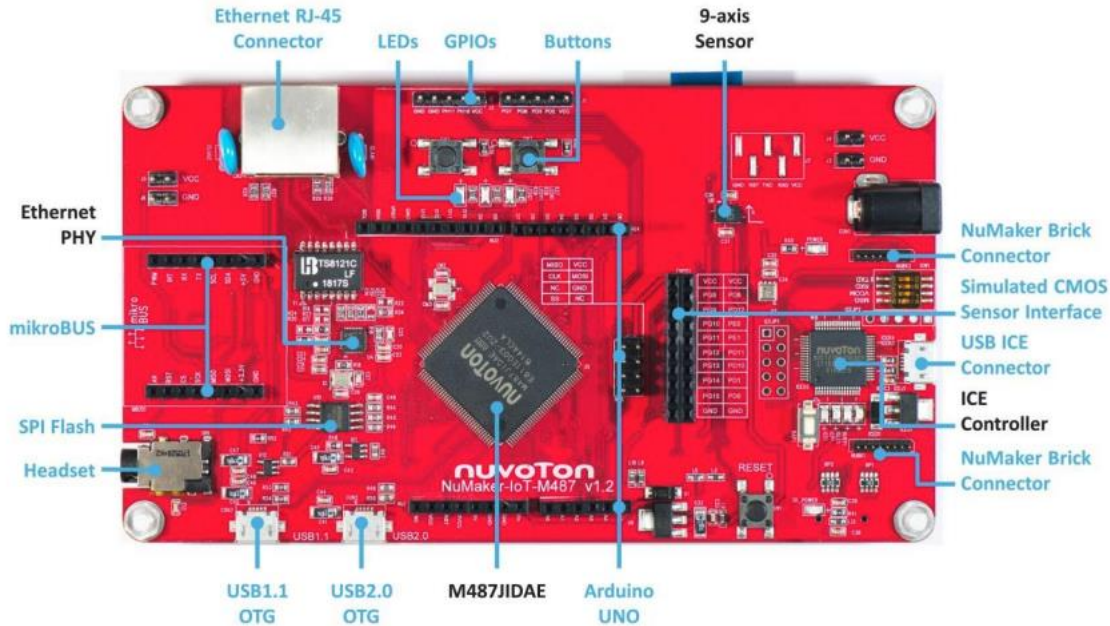
### DataSheet

Refer to the [NuMaker-IoT-M487 User Manual](#) for an introduction to the specifications, features, and uses of the NuMaker-IoT-M487 board.

## Schematic

For the board schematic, see Section 3 of the [User Manual](#).

## Key Components



For a description of key components, jumper settings, LED descriptions and power requirements, see **Section 2** in the [User Manual](#).

## Hardware requirements to run FreeRTOS demo

- [NuMaker-IoT-M487](#)
- [Quectel-BG96A](#)

A SIM card will be required in order to enable communication onto the cellular network. Please make sure that the network supported by your operator is the one supported by the modem embedded on the board.

In most cases, you have a compatible SIM card and an active data plan. If you do not have a compatible SIM card, you can obtain SIMs from the [AWS marketplace](#).

## Additional Hardware References

For more information, refer to the [product information](#) on the company website.

Additional IoT hardware kits can be found on the [Nuvoton Direct](#) microsite.

# Set up your development environment

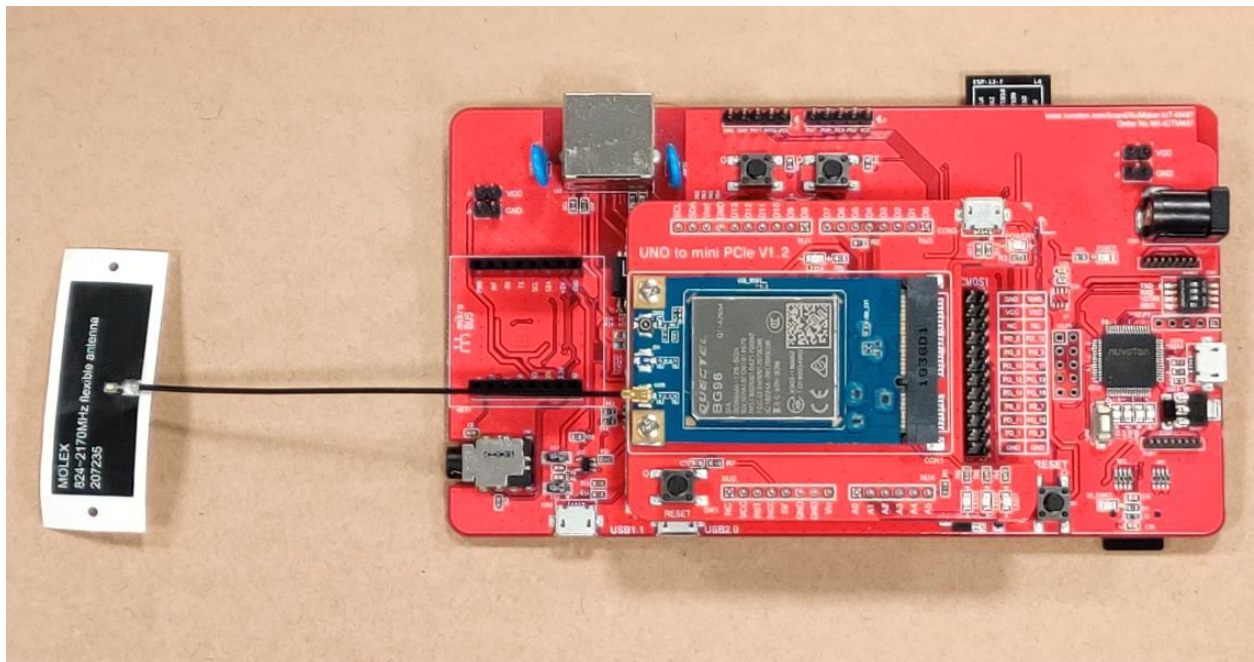
The Keil MDK v5 Essential, Plus, or Pro version should work for the Nuvoton M487 (Cortex-M4 core) MCU. You can also download the Keil MDK Nuvoton edition for Nuvoton Cortex-M4 series MCU at a price discount ([here](#))

## Install Development Tool for NuMaker-IoT-M487

1. Download the Keil MDK Nuvoton Edition on [Keil MDK website](#) or standard edition on [Keil MDK-Arm website](#).
2. Install Keil MDK with your license.

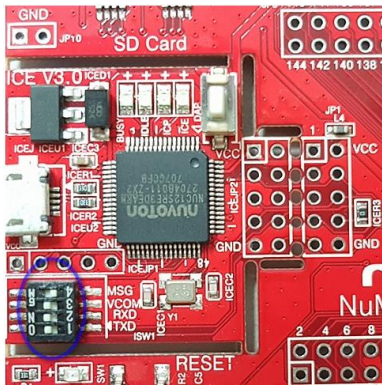
If you experience issues during installation, please contact us ([here](#)).

## Set up your hardware



To set up the NuMaker-IoT-M487 + Quectel-BG96A:

1. Connect your computer to the micro-USB port on NuMaker-IoT-M487.
2. Plug the Quectel-BG96A in the Arduino UNO Pin Header

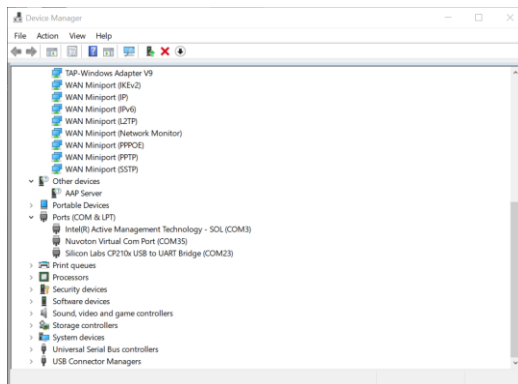


3. Power supply through USB of [Nu-Link](#) and ensure MSG switch (No.4 of ISW1 on ICE) is OFF. For more information, click [here](#) to download the user manual.

## Prerequisites

- Make sure you have installed the latest BG96 firmware. You can download the latest BG96 firmware from [here](#). (You will need to login or sign on with a Quectel account.)

# Establishing a serial connection



1. Check the list of identified COM ports in the Windows Device Manager.
2. Start a serial terminal and open a connection with the following settings:
  - Baud rate: 115200
  - Data: 8 bit
  - Parity: None
  - Stop bits: 1
  - Flow control: None

You will need to install a terminal program for connecting computers to the device.

# Setup your AWS account and Permissions

To create an AWS account, see [Create and Activate an AWS Account](#).

To create an IAM user, follow the instructions at [How to create an IAM user](#). For a deeper understanding of IAM, refer to the [IAM User Guide](#).

To grant your IAM user account access to AWS IoT and FreeRTOS, attach the following IAM policies to your IAM user account:

- AmazonFreeRTOSFullAccess
- AWSIoTFullAccess

## To attach the AmazonFreeRTOSFullAccess policy to your IAM user

1. Browse to the [IAM console](#), and from the navigation pane, choose **\*\* Users\*\***.
2. Enter your user name in the search text box, and then choose it from the list.
3. Choose **Add permissions**.
4. Choose **Attach existing policies directly**.
5. In the search box, enter **AmazonFreeRTOSFullAccess**, choose it from the list, and then choose **Next: Review**.
6. Choose **Add permissions**.

## To attach the AWSIoTFullAccess policy to your IAM user

1. Browse to the [IAM console](#), and from the navigation pane, choose **\*\* Users\*\***.
2. Enter your user name in the search text box, and then choose it from the list.
3. Choose **Add permissions**.
4. Choose **Attach existing policies directly**.
5. In the search box, enter **AWSIoTFullAccess**, choose it from the list, and then choose **Next: Review**.
6. Choose **Add permissions**.

For more information about IAM and user accounts, see [IAM User Guide](#).

For more information about policies, see [IAM Permissions and Policies](#).

# Provision the device with AWS IoT

Your board must be registered with AWS IoT to communicate with the AWS Cloud. To register your board with AWS IoT, you need the following:

## An AWS IoT policy

The AWS IoT policy grants your device permissions to access AWS IoT resources. It is stored on the AWS Cloud.

## An AWS IoT thing

An AWS IoT thing allows you to manage your devices in AWS IoT. It is stored on the AWS Cloud.

## A private key and X.509 certificate

The private key and certificate allow your device to authenticate with AWS IoT.

## To create an AWS IoT policy

1. To create an IAM policy, you need to know your AWS Region and AWS account number.

To find your AWS account number, open the [AWS Management Console](#), locate and expand the menu beneath your account name in the upper-right corner, and choose **My Account**. Your account ID is displayed under **Account Settings**.

To find the AWS region for your AWS account, use the AWS Command Line Interface. To install the AWS CLI, follow the instructions in the [AWS Command Line Interface User Guide](#). After you install the AWS CLI, open a command prompt window and enter the following command:

```
aws iot describe-endpoint
```

The output should look like this:

```
{
  "endpointAddress": "xxxxxxxxxxxxx.iot.us-west-2.amazonaws.com"
}
```

In this example, the region is us-west-2.

2. Browse to the [AWS IoT console](#).
3. In the navigation pane, choose **Secure**, choose **Policies**, and then choose **Create**.
4. Enter a name to identify your policy.

5. In the **Add statements** section, choose **Advanced mode**. Copy and paste the following JSON into the policy editor window. Replace *aws-region* and *aws-account* with your AWS Region and account ID.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "arn:aws:iot:aws-region:aws-account-id:*"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Publish",
      "Resource": "arn:aws:iot:aws-region:aws-account-id:*"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": "arn:aws:iot:aws-region:aws-account-id:*"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Receive",
      "Resource": "arn:aws:iot:aws-region:aws-account-id:*"
    }
  ]
}
```

This policy grants the following permissions:

`iot:Connect`

Grants your device the permission to connect to the AWS IoT message broker with any client ID.

`iot:Publish`



Grants your device the permission to publish an MQTT message on any MQTT topic.

`iot:Subscribe`

Grants your device the permission to subscribe to any MQTT topic filter.

`iot:Receive`

Grants your device the permission to receive messages from the AWS IoT message broker on any MQTT topic.

NOTE – all devices in your fleet must have credentials with privileges that authorize intended actions only, which include (but not limited to) AWS IoT MQTT actions such as publishing messages or subscribing to topics with specific scope and context. The specific permission policies can vary for your use cases. Identify the permission policies that best meet your business and security requirements.

For sample policies, refer to <https://docs.aws.amazon.com/iot/latest/developerguide/example-iot-policies.html>

Also refer to <https://docs.aws.amazon.com/iot/latest/developerguide/security-best-practices.html>

6. Choose **Create**.

#### To create an IoT thing

1. Browse to the [AWS IoT console](#).
2. In the navigation pane, choose **Manage**, and then choose **Things**.
3. If you do not have any IoT things registered in your account, the **You don't have any things yet** page is displayed. If you see this page, choose **Register a thing**. Otherwise, choose **Create**.
4. On the **Creating AWS IoT things** page, choose **Create a single thing**.
5. On the **Add your device to the thing registry** page, enter a name for your thing, and then choose **Next**. The thing name should be unique within your account.
6. On the **Add a certificate for your thing** page, under **One-click certificate creation**, choose **Create certificate**.
7. Download your private key and certificate by choosing the **Download** links for each.
8. Choose **Activate** to activate your certificate. Certificates must be activated prior to use.
9. Choose **Attach a policy** to attach a policy to your certificate that grants your device access to AWS IoT operations.
10. Choose the policy you just created and choose **Register thing**.

After your board is registered with AWS IoT, you can continue to configure FreeRTOS to send those credentials during the TLS connection.



# Download and Build the FreeRTOS demo project

You can download a release from GitHub with the following command:

To clone using HTTPS:

```
git clone --branch nuvoton_m487_cellular https://github.com/OpenNuvoton/aws-freertos.git --recurse-submodules
```

Using SSH:

```
git clone --branch nuvoton_m487_cellular git@github.com:OpenNuvoton/aws-freertos.git --recurse-submodules
```

If you have downloaded the repo without using the `--recurse-submodules` argument, you need to run:

```
git submodule update --init --recursive
```

To checkout the branch to `nuvoton_m487_cellular`, you need to run:

```
git checkout nuvoton_m487_cellular
```

Navigate to the folder `<your FreeRTOS directory>/projects/Nuvoton/numaker_iot_m487_wifi/uvision/aws_demos_bg96/` to open project.

FreeRTOS is a C language project, and the certificate and private key must be specially formatted to be added to the project.

## Include the certificates in your FreeRTOS project

1. In a browser window, open `<your FreeRTOS directory>/tools/certificate_configuration/CertificateConfigurator.html`.
2. Under **Certificate PEM file**, choose the `ID-certificate.pem.crt` that you downloaded from the AWS IoT console.
3. Under **Private Key PEM file**, choose the `ID-private.pem.key` that you downloaded from the AWS IoT console.
4. Choose **Generate and save aws\_clientcredential\_keys.h**, and then save the file in `<your FreeRTOS directory>/demos/include`. This overwrites the existing file in the directory.
5. An APN or Access Point Name is the combination of values that indicates to your device how to connect to the mobile internet servers of your network provider. You can find out your specific APN settings on your network operator website. Define `configCELLULAR_APN` in `aws_demos_cellular.h` with the Cellular network provider.

6. PDN context identifier specifies a particular packet data protocol context definition. The range of permitted values is 1 to 16. The default value is minimum value 1. Define configCELLULAR\_PDN\_CONTEXT\_ID in aws\_demos\_cellular.h with the PDN context id you desire to use.
7. Define configCELLULAR\_DNS\_SERVER in aws\_demos\_cellular.h with the DNS server that will be used.

```
#ifndef __AWS_CELLULAR_DEMO_H__
#define __AWS_CELLULAR_DEMO_H__

/*
 * @brief Access Point Name (APN) for your cellular network.
 * @todo set the corresponding APN according to your network provider.
 */
#define configCELLULAR_APN "MY_APN"

/*
 * @brief PDN context id for cellular network.
 */
#define configCELLULAR_PDN_CONTEXT_ID ( CELLULAR_PDN_CONTEXT_ID_MIN )

/*
 * @brief PDN connect timeout.
 */
#define configCELLULAR_PDN_CONNECT_TIMEOUT ( 20000UL )

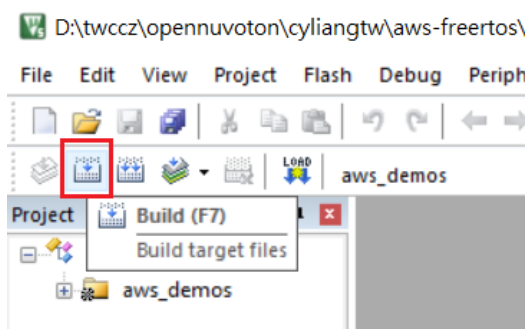
/*
 * @brief DNS server address for cellular network socket service.
 * @todo Set the preferred DNS server address.
 */
#define configCELLULAR_DNS_SERVER "8.8.8.8"
```

### Note

*The certificate and private key are hard-coded for demonstration purposes only. Production-level applications should store these files in a secure location.*

### To build the application

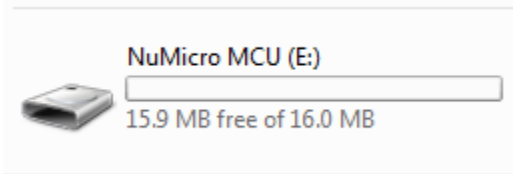
Click the Build button or press F7 to build the project, and the image **aws\_demos.bin** will be generated.



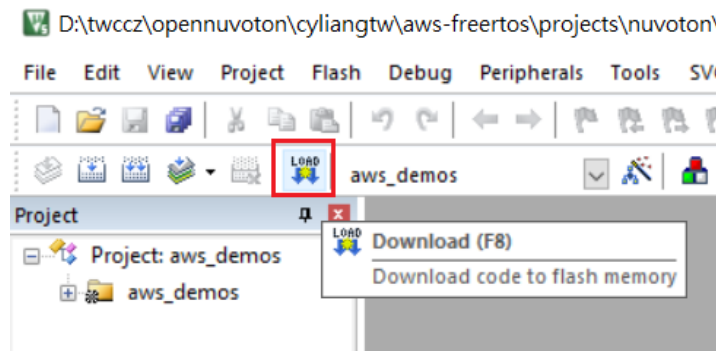
# Run the FreeRTOS demo project

Before downloading the code, make sure you have assembled the NuMaker-IoT-M487 and BG96 together. For more detail, see the section [Set up your hardware](#).

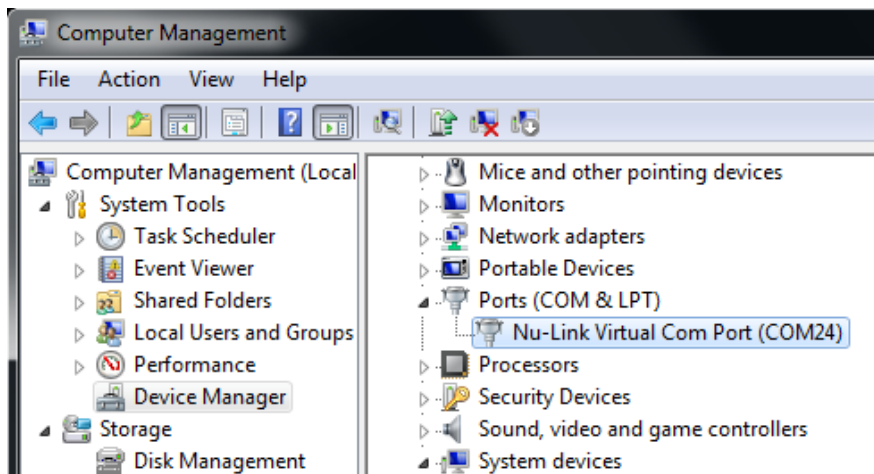
1. Navigate to the firmware directory. In the example working directory:  
**C:/amazon-freertos/projects/Nuvoton/numaker\_iot\_m487\_wifi/uvision/  
aws\_demos\_bg96/obj/**
2. Flash the firmware to NUMAKER-IOT-M487:
  - a. In MSG mode, you can copy the image to the “NuMicro MCU” drive directly.



- b. In ICE mode, you can click the “LOAD” button.



3. Open your favorite tool to monitor the logs with the Port connected to the device. From the console, you can see the demo code run successfully.



```

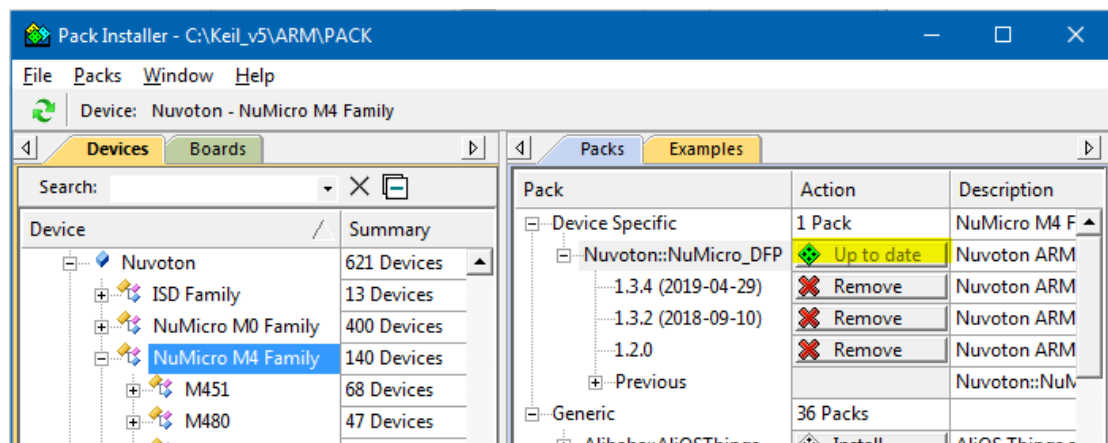
29 14069 [iot_thread] [INFO ][MQTT][14069] Establishing new MQTT connection.
30 14077 [iot_thread] [INFO ][MQTT][14077] Anonymous metrics (SDK language, SDK version) will be provided to AWS IoT. Re
compile31 14079 [iot_thread] [INFO ][MQTT][14079] (MQTT connection 2000f180, CONNECT operation 2000f2a0) Waiting for ope
ration complet32 14751 [iot_thread] [INFO ][MQTT][14751] (MQTT connection 2000f180, CONNECT operation 2000f2a0) Wait com
plete with result SUC33 14752 [iot_thread] [INFO ][MQTT][14752] New MQTT connection 20007b80 established.
34 14755 [iot_thread] [INFO ][MQTT][14755] (MQTT connection 2000f180) SUBSCRIBE operation scheduled.
35 14755 [iot_thread] [INFO ][MQTT][14755] (MQTT connection 2000f180, SUBSCRIBE operation 2000f0d0) Waiting for operatio
n complCheck delay ...15000
36 15518 [iot_thread] [INFO ][MQTT][15518] (MQTT connection 2000f180, SUBSCRIBE operation 2000f0d0) Wait complete with r
esult S37 15518 [iot_thread] [INFO ][DEMO][15518] All demo topic filter subscriptions accepted.
38 15520 [iot_thread] [INFO ][DEMO][15518] Publishing messages 0 to 1.
39 15520 [iot_thread] [INFO ][MQTT][15520] (MQTT connection 2000f180) MQTT PUBLISH operation queued.
40 15527 [iot_thread] [INFO ][MQTT][15527] (MQTT connection 2000f180) MQTT PUBLISH operation queued.
41 15527 [iot_thread] [INFO ][DEMO][15527] Waiting for 2 publishes to be received.
42 15970 [iot_thread] [INFO ][DEMO][15970] MQTT PUBLISH 0 successfully sent.
43 16046 [iot_thread] [INFO ][DEMO][16046] Incoming PUBLISH received:
Subscription topic filter: iotdemo/topic/1
Publish topi44 16046 [iot_thread] [INFO ][MQTT][16046] (MQTT connection 2000f180) MQTT PUBLISH operation queued.
45 16046 [iot_thread] [INFO ][DEMO][16046] Acknowledgment message for PUBLISH 0 will be sent.
46 16115 [iot_thread] [INFO ][DEMO][16115] MQTT PUBLISH 1 successfully sent.
47 16242 [iot_thread] [INFO ][DEMO][16242] Incoming PUBLISH received:
Subscription topic filter: iotdemo/topic/2
Publish topi48 16242 [iot_thread] [INFO ][MQTT][16242] (MQTT connection 2000f180) MQTT PUBLISH operation queued.

```

# Troubleshooting



## Update Nuvoton NuMicro M4 Family of Keil Device Pack

1. If you get “Device Support Check - Missing Device(s)” error in load Keil project, please update M480 Keil driver pack from [standard Kiel IDE Pack installer](#) or [Nuvoton installation pack](#).
2. If you choose to go through [standard Kiel IDE Pack installer](#), please select Nuvoton -> NuMicro M4 Family and then press “Update” button to let the state change as “Up to date” in Keil Pack Installer window.



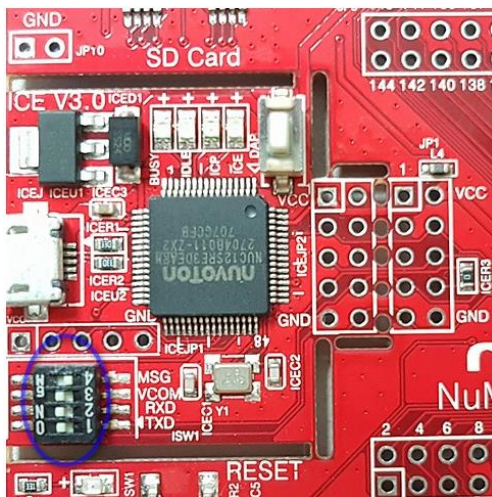
3. If you choose to go through [Nuvoton installation pack](#), please download and install “Nu-Link\_Keil\_Driver xxx.exe” directly from [https://www.nuvoton.com/hq/support/tool-and-software/software/development-tool/?\\_locale=en](https://www.nuvoton.com/hq/support/tool-and-software/software/development-tool/?_locale=en).

#### Nu-Link Driver

File name	Description	Version	Date
 <b>Nu-Link_Keil_Driver_V3.00.6909</b>	This driver is to support Nu-Link to work under Keil RVMDK Development Environment for all NuMicro Family Devices.	V3.00.6909	2019-5-7
 <b>Revision History</b>			

## VCOM driver & switch for Nu-Link

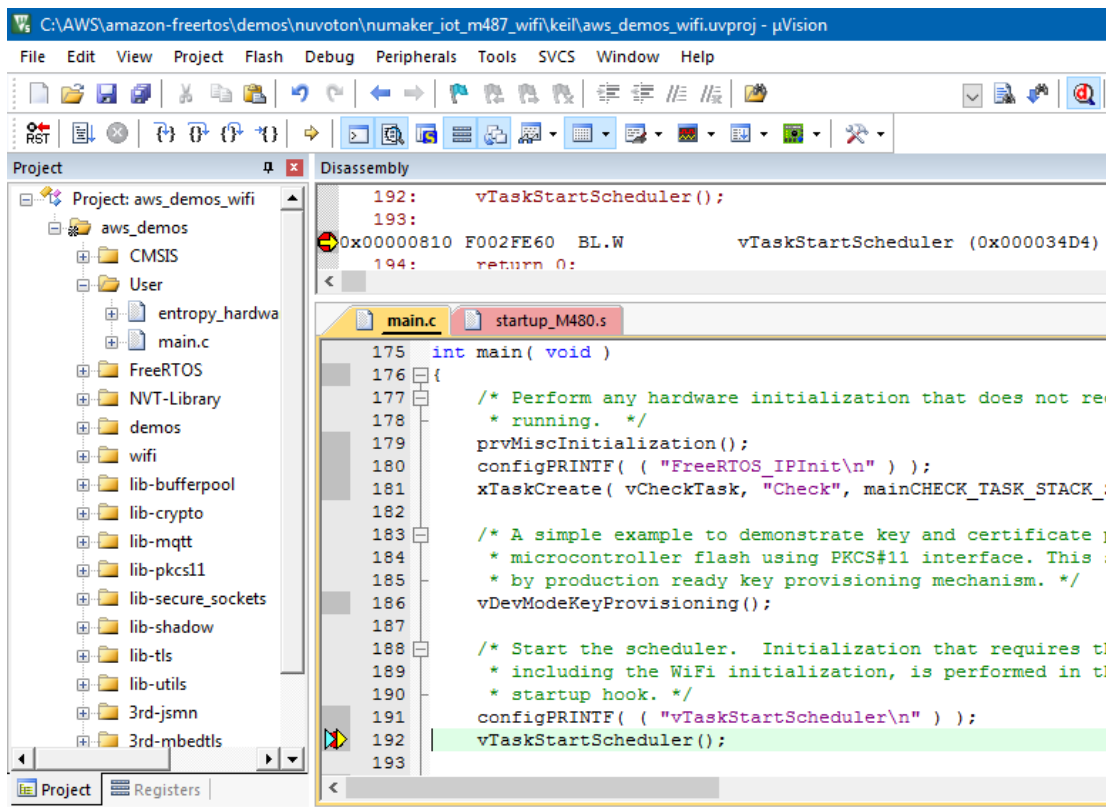
1. If your windows can't recognize device VCOM, please install NuMaker windows serial port driver from <https://goo.gl/4VGca6>.
2. For Keil IDE connection with device through Nu-Link, please ensure MSG switch (No.4 of ISW1 on ICE) is OFF:



## Debugging Amazon FreeRTOS Projects in Keil $\mu$ Vision

### To start debugging the project

1. Open IDE Keil  $\mu$ Vision and choose "File" from the top menu, then click "Open"
2. Open the BG96 demo project, please select target project "aws\_demos.uvproj" from <BASE\_FOLDER>\AmazonFreeRTOS\projects\nuvoton\numaker\_iot\_m487\_wifi\uvision\aws\_demos\_bg96.
3. From the **Project** menu, choose **Build Target**.
4. From the **Debug** menu, choose **Start/Stop Debug Session**. The **Call Stack + Locals** window opens when you start the debug session.  $\mu$ Vision flashes the demo to the board, runs the demo, and stops at the beginning of the `main()` function.
5. You could set breakpoints in your project's source code, and run the code. The program counter stops like as the following:

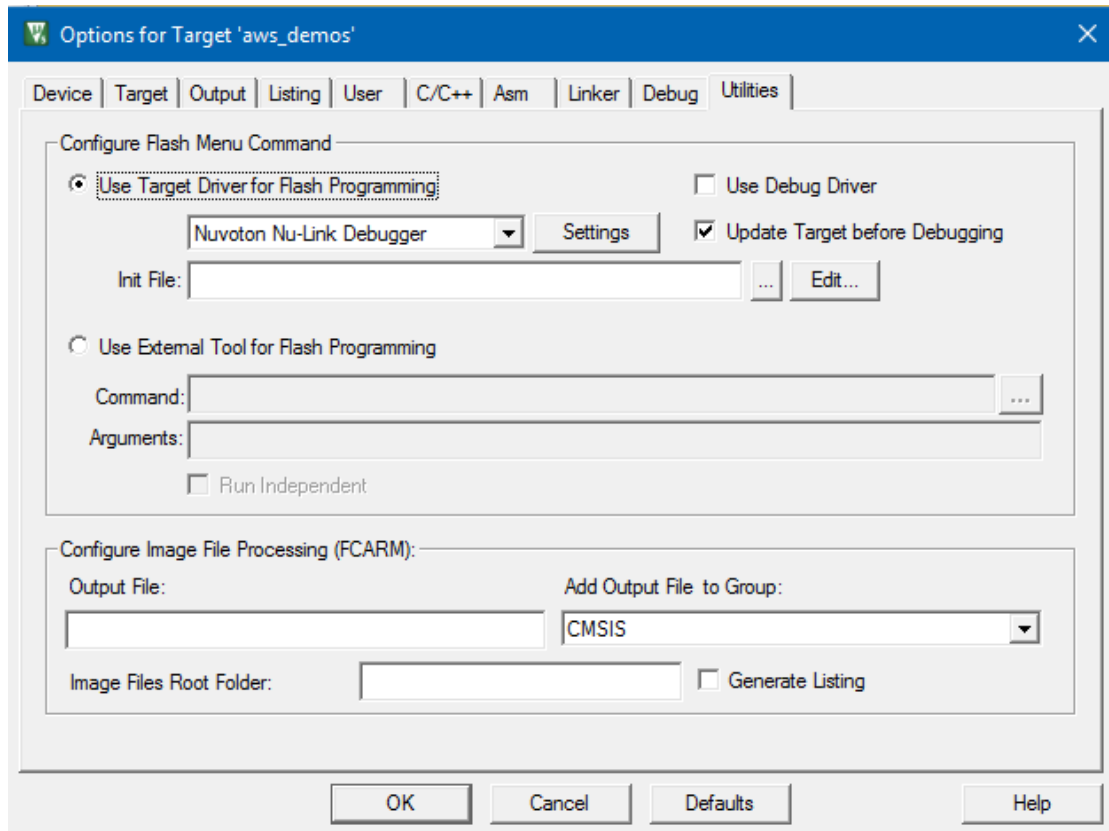


## Troubleshooting the IDE Debugger Settings

If you are having trouble debugging an application, your debugger settings might be incorrect.

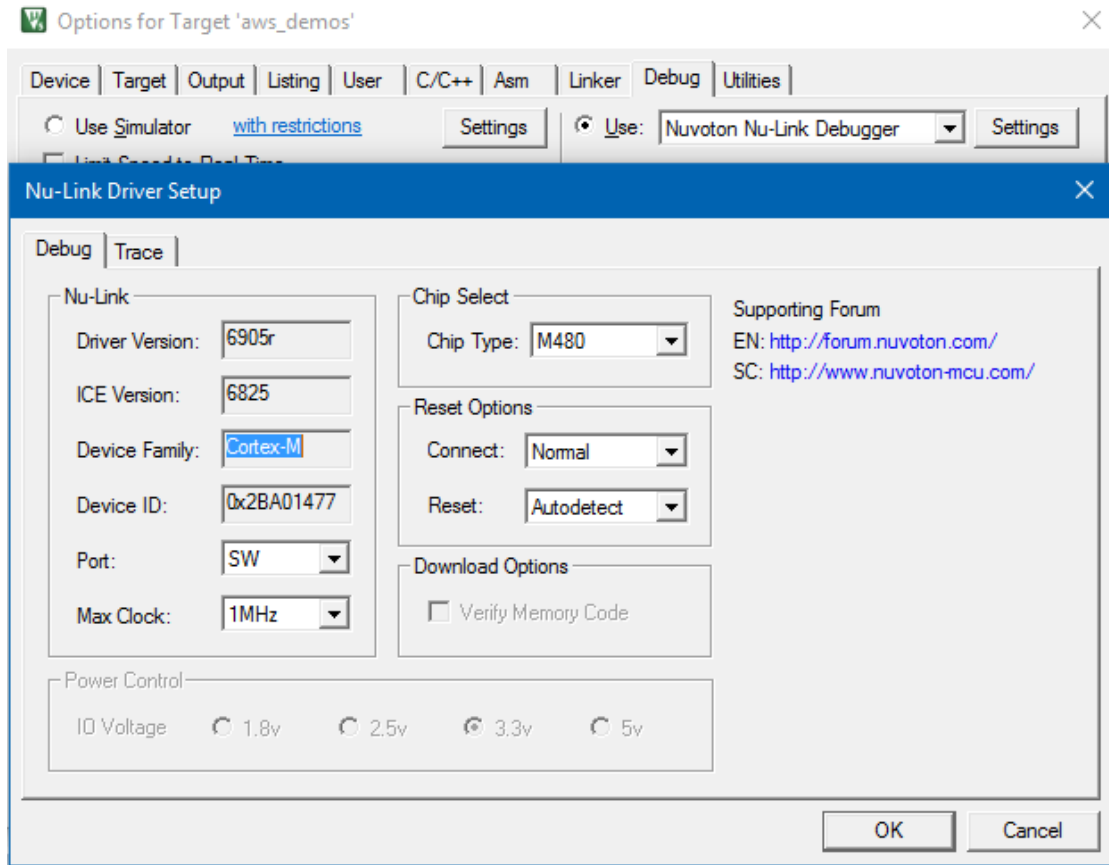
### To verify that your debugger settings are correct

1. Open Keil uVision.
2. Right-click the `aws_demos.uvproj` project, choose **Options**, and under the **Utilities** tab, ensure to set “Nuvoton Nu-Link Debugger” in the dropdown box.



3. choose **Options**, and under the **Debug** tab, choose **Settings**, next to “**Nuvoton Nu-Link Debugger**”. In “**Chip Type**” dropdown box, it should be “**M480**” as below:





For general troubleshooting information about Getting Started with FreeRTOS, see [Troubleshooting getting started](#).