Hi, I'm Alan Hawse. Welcome back to Amazon FreeRTOS 101 with the Cypress 43907 and 54907 Wi-Fi chips. In this video, I'm going to show you how to setup the Cypress Amazon FreeRTOS environment and how to run your first application.

One of the reasons that I really like Amazon is they're committed to good documentation. That must be because of their lineage with books. Seriously, if you can't figure out how to use their Cloud how good can it be? So, let me show you, they've got excellent documentation. First go to this link (https://docs.aws.amazon.com/freertos). Here you will find a bunch of documentation for AFR with boards and specific user guides.

Click the plus mark on the left next to the "Board-specific Getting Started Guides". Click the link for the applicable kit based on the board that you have – either they CYW943907AEVAL1F or the CYW954907AEVAL1F. For these videos, we're using the 43907 version so I'll click on that one.

Let's go through the list of high-level tasks that we're going to do. I will do this by walking your through each step in the Getting Started Guide. Once I've gone through all of the getting started guide I'll loop back, and I'll actually demonstrate each step live.

#1 (walk through from getting started guide)

First, you need to setup an AWS account and configure the permissions. Although you need a credit card to register, they give you really what I think is an amazing amount of functionality for free. Maybe it should be obvious... but you can build big time IoT and Cloud applications using Amazon. You can also do a bunch of cool stuff for free. I actively use AWS for my hobby project, www.iotexpert.com, where I get access to a MYSQL database using Amazon RDS. I use Lamba functions, and I use EC2, and it's really awesome and it's also free or essentially free.

#2

Anyway... once you have an account, you will need to register your MCU Board with AWS IoT. I know it will drive Greg, the guy who helps me with all of these videos crazy, but I'm going to digress a little bit... At this point in the world security matters... actually security matters a LOT. The world is full of people who want to turn your doorbell or your microwave oven or all the other crazy things that people put in IoT -they want to turn those things into a zombie internet army. So, AWS - they "force" you to do good security when connecting to their Cloud. Quite frankly I don't blame them, because really it's just an intelligence test using good security. So what I would say to you is just say no to non-secure devices. There's no excuse implement any Cloud connection without TLS. The good news is it's easy to do with the Cypress devices and Amazon FreeRTOS, and I'm going to explain all of that in these videos.

So, specifically, what does "secure" mean? Simple, to be secure you need to create a two-sided encrypted TLS connection before you transfer data. This means that both sides verify uniquely the identity of the other side using strong certificates. AWS has a console where you can create all of the pieces you need to make this work, specifically things, certificates, and policies.

#3

Now, with that registration stuff out of the way, you will download Amazon FreeRTOS from GitHub.

#4

Next, you will configure the Amazon FreeRTOS Demos. This includes things like the MQTT broker address, the Wi-Fi and network credentials, and the device Certificates. This will be easier to do with WICED Studio IDE installed so when I get to the demo, I going to do this step out of order from the getting started guide.

#5

Step 5 is to install WICED Studio.

#6

Step 6 is to setup an environment variable which hooks it all together. As you can see, there's instructions here for Windows, Linux, and macOS – all three that we support.

#7

Next, you will establish a serial connection with your board so that you can see messages from the Cloud as well as debug information. I know that there's a bunch of you out there who like to use printf debugging, and that console is exactly what we put that there for.

#8

In step 8 we will get down to brass tacks and actually import, build and run the demo project from WICED Studio.

#9

I will also show you how to monitor messages using the MQTT test client on the Cloud side.

#10

Finally, there's a section on troubleshooting in case you run into an issue. Of course, I don't need this section since I'm perfect … right Greg? Oh no... alright, see he's shaking his head. Aright, actually I make a lot of mistakes but as I'm standing here in front of a camera and I have a really excellent editor, we can fix all of these things and you don't get to see the mistakes live in action.

#1 (demo)

Alright, so let's go through each step. I'll assume that you have already created an AWS login and you have configured the permissions, so I'll start with registering the Cypress 43907 MCU board with AWS IoT.

#2

Let's create a Policy and AWS Thing with a Certificate to use with our kit. Amazon likes to “improve” their website on a regular basis so the exact steps and the exact screenshots I'm showing you – they may change a little bit when you try this out for yourself, but really it's not too hard to figure out, but yes, the screens do move around.

Go to your AWS Amazon Console (console.aws.amazon.com) and open "IoT Core". Now go to "Secure", click "Policies" and then click "Create".

Give a name to your Policy. I'll use “all”.

In the Action section you need to add 4 statements with 4 actions: iot:Connect, iot:Publish, iot:Subscribe, and iot:Receive. Add the actions one at a time and check "Allow" for each action. You should put a "*" for the Resource ARN since this is meant to be a generic Policy for all of the things in your Cloud.

If you click on the Advanced mode, you can see the JSON created by your choices. You can modify the JSON directly in the Advanced mode if you really want to do something more complex.

Once you are satisfied with the Policy settings, click “Create”. You will see the Policy ”all” in the Policy window now.

Next let's create a new Thing. Click "Manage" and "Things". Click "Create" and then “Create a single thing”. Give it a name of your choice. I use “afr_43907”.

In a real scenario with a lot of things going on in your Cloud, you'll probably use one of the APIs that they provide as part of their command line interface when you deploy, hopefully, your army of doorbells or your army of cameras or your army of whatever you feel like making an army of just as long as you've got 43907s on the inside!

Leave other fields to default and click "Next". Click on "One-click certificate creation > Create certificate". Save the certificate, public key and private key to a location on your machine – you're going to need these in a minute to embed into your firmware.

It's important that you download these now since you can't get them once you leave this page. Let me say this again... if you leave this page without downloading these documents you WILL LOSE THEM, and you will need to recreate them... and let me tell you, this is the voice of experience. They do this for security reasons – you don’t want anyone else to be able to hack their way in later and get access to your keys. Afterall, they call them private keys for a reason – you need to keep your keys private.

Click Activate to activate the Certificate. Now click on “Attach a Policy”. You will see the policy “all”. Select it and click “Register thing”. You will now see the created Thing in your list of Things. See, I told you this thing word can get really awkward to say when you're building these things.

As a final step in the AWS console, click Settings and save the Broker Endpoint name somewhere. You will need this in a step later. The Broker name is just a fully qualified DNS name that gives you the IP address of the message broker – the MQTT message broker – which is where you're going to send and receive the messages. Actually, with MQTT they don't call it send and receive, they call it publish and subscribe, but you get the idea.

#3

Now we are going to download Amazon FreeRTOS from GitHub. You need to have a Git client installed to clone the repository. There are lots of Git clients out there so if you don't have one just Google it, or use a Mac or Linux where it's already built in. There's also lots of training available on Git and GitHub so I won't cover these topics in these videos.

First, clone the GitHub repository for Amazon FreeRTOS (github.com/aws/amazon-freertos/) on your computer. You can checkout the master branch if you want the latest of everything, but changes done in the master may conflict with existing files and cause things to act strangely so I would be a little bit careful there.

#5

If you haven't installed WICED Studio yet, now's a good time to do that. You should install WICED Studio 6.2.1 or later. You will need to create a community account with Cypress if you don't have one already, but don't worry it's also free.

#6

We need to create an environment variable for the WICED Studio SDK so that AFR projects can find the tools that they need. Make sure to close the WICED Studio IDE before creating this environment variable.

When the WICED Studio SDK got installed on your machine there were two separate folders. One for the WICED Studio SDK and another for the WICED Studio IDE – we need a variable that points to the SDK location, not the IDE location.

By default, on a Windows machine the SDK is created inside the of user's Documents folder in a sub-folder called WICED-Studio-<x.y> where x and y are the major and minor numbers of the WICED Studio release. Here's how the environment variable is setup for WICED Studio 6.2 on a Windows machine…

#7

Before we go back to WICED Studio, let's connect our kit and open a serial terminal window. This will allow us to see debug messages and messages from the Cloud once we program the kit.

I'm using putty, but you can use any terminal emulator that you like. The baud rate is 115200.

#8

Now we will re-launch the WICED Studio IDE. Make sure to choose "WICED Filters off" in the WICED Platform menu.

From the File menu, choose Import. Expand the "General" folder, choose "Existing projects into Workspace", and then choose Next. In "Select root directory", click Browse and follow the path to where you cloned the Git repository. The path inside the repository is projects/cypress/CYW943907AEVAL1F/wicedstudio. Once you have the path set, click OK. Under Projects, select only the aws_demos. Choose Finish to import the project. See how aws_demos is imported as a separate project into your workspace?

#4

Although we can enter the Certificate and Private key by hand, Amazon has created a simple utility that will automatically generate a header file containing both. Navigate to the location where you cloned the Git repository and go to amazon-freertos/tools/certificate_configuration.

Open the CertificateConfigurator.html in a browser. Point to the Certificate and Private key files that you saved a while ago while you were creating the AWS Thing. Click on “Generate and save aws_clientcredential_keys.h”. Save and replace that file in the “amazon-freertos/demos/include” directory.

Now let’s get back to WICED Studio. I'm going to do a "Refresh" first to make sure the changes to the credential and key file are loaded.

Now, open “demos/include/aws_clientcredential.h”. Set the Broker endpoint name - the one that we copied from AWS IoT Core Settings - into clientcredentialMQTT_BROKER_ENDPOINT[].

Then, enter the “clientcredentialIOT_THING_NAME” variable with the Thing name that you used in the AWS setup.

You should also enter the WIFI_SSID and password into this file.

#8 (part 2)

Finally, we are ready to build the project and program the board. Open the make target and double-click on “demo.aws_demo-CYW943907AEVAL1F-FreeRTOS-LwIP HOST_OS=Win32 download run”. This will build the demo, then download and run it on your board.

Once programming is done, you will see that it connects to the AWS broker and publishes 20 messages and then disconnects. It will do this each time you reset the kit.

#9

On the Cloud side, if you click on "Test" you can use the test client to subscribe to a Topic to see the messages received by the broker on that topic. Enter the topic iotdemo/# and click on "Subscribe to topic".

The name iotdemo is used as a prefix for all the messages from the demo firmware – I'll show you where that's set in the firmware in the next video. The pound sign - or the hashtag for all of you social media people out there - is a wildcard so any messages to topics starting with iotdemo/ will be displayed.

The demo sends strings instead of JSON, so we'll just display things as strings.

Now reset the kit and you will see all of the messages that arrive at the broker. The MQTT broker test client is really super important for debugging messages because you can both Publish to Topics as well as Subscribe to Topics so when you're debugging your system, having access to this client is really the best way to see what's actually happening on the AWS side.

Awesome – we have a fully functional AFR IoT device communicating with the Cloud and it really didn't take too long to get set up and running considering how much is really going on behind the scenes like making a Wi-Fi connection to an access point, establishing secure MQTT connections to the Cloud, sending and receiving data…also known as publishing and subscribing.

In the next video we'll talk about the firmware architecture and how to use the hardware debugger.

You can post your comments and questions in our developer community and one of the Cypress people or one of our many customers will respond. Or, as always you're welcome to email me a personal comment or question to alan_hawse@cypress.com or tweet me @askioexpert. Thank You!