

Embedded System Design Techniques™

Connecting Edge Devices to the IoT using Amazon FreeRTOS

Class 3: Setting up and Configuring Amazon FreeRTOS

March 21st, 2018
Jacob Beningo

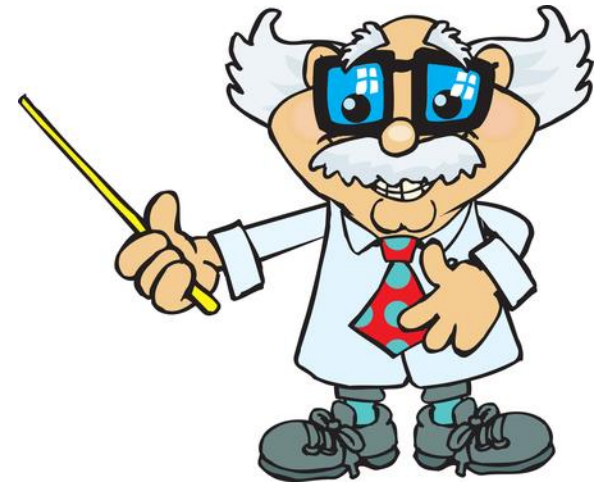
Course Overview

Topics:

- Introduction to Amazon FreeRTOS
- Amazon Web Services Fundamentals
- **Setting up and Configuring Amazon FreeRTOS**
- Amazon FreeRTOS Behind the Scenes
- Creating your own a:FreeRTOS Application

Session Overview

- Setup Procedure
- Toolchain Setup
- Configuring for AWS
- Running a:FreeRTOS



Presented by:

a:FreeRTOS Setup Procedure

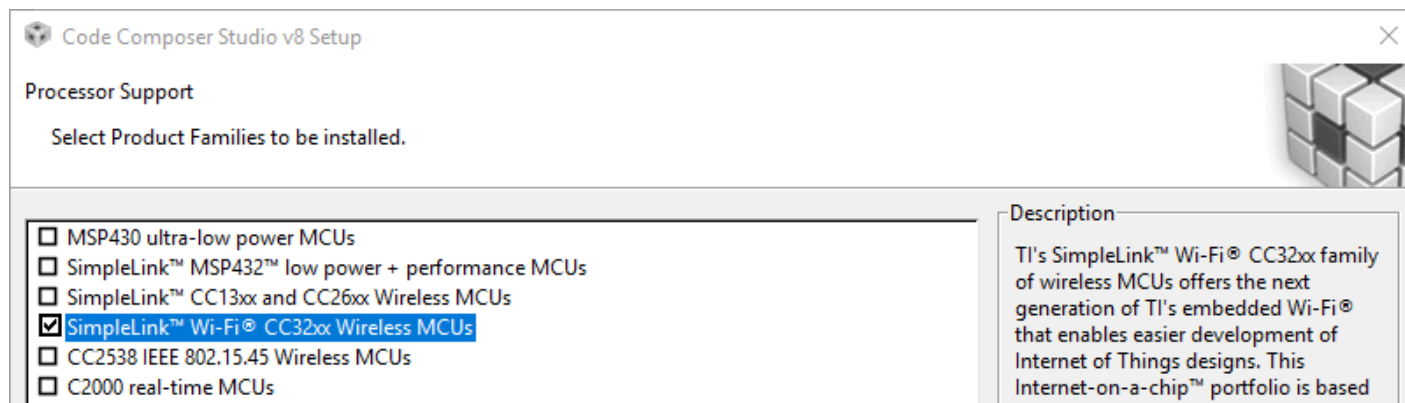
- 1) Install Code Composer
- 2) Download a:FreeRTOS
 - Configured for the CC3220SF-LAUNCHXL
- 3) Import a:FreeRTOS
- 4) Configure for AWS
- 5) Run the example

Step 1 - Code Composer Setup

1) Download the latest Code Composer

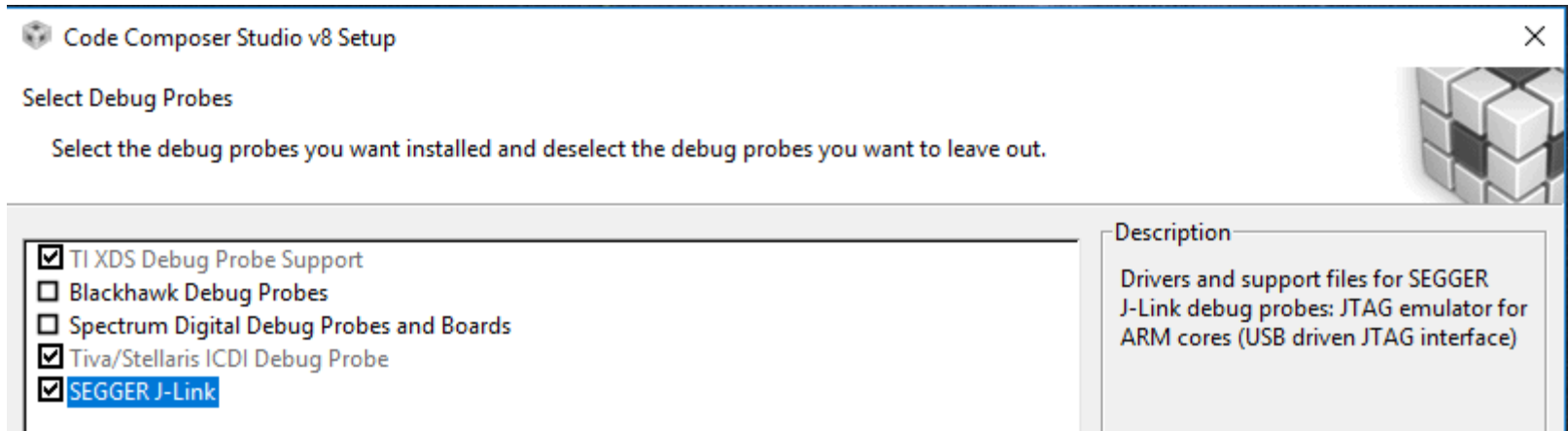
- http://processors.wiki.ti.com/index.php/Download_CCS
- Use the offline installer

2) Select the SimpleLink Wi-Fi CC32xx

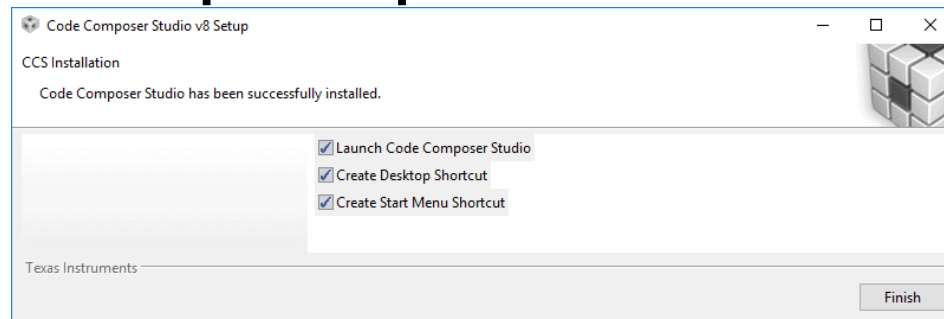


Step 1 - Code Composer Setup

3) Include the J-Link (If you have one)

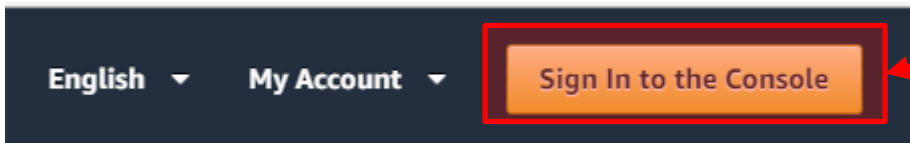


4) Follow the prompts until CC is installed.

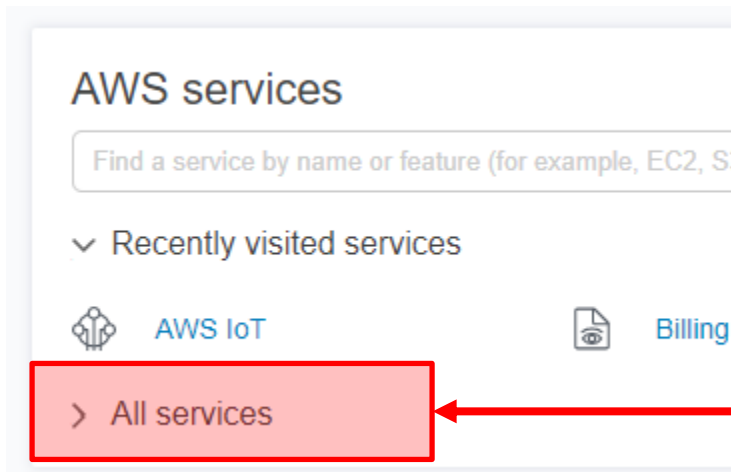


Step 2 – a:FreeRTOS Download

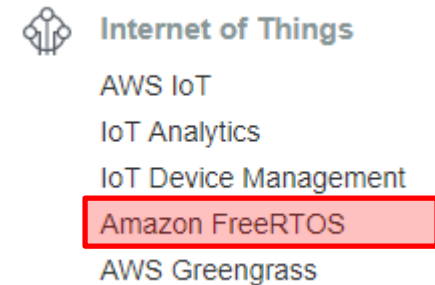
<https://aws.amazon.com>



Sign-in / Create Account



Expand



Step 2 – a: FreeRTOS Download

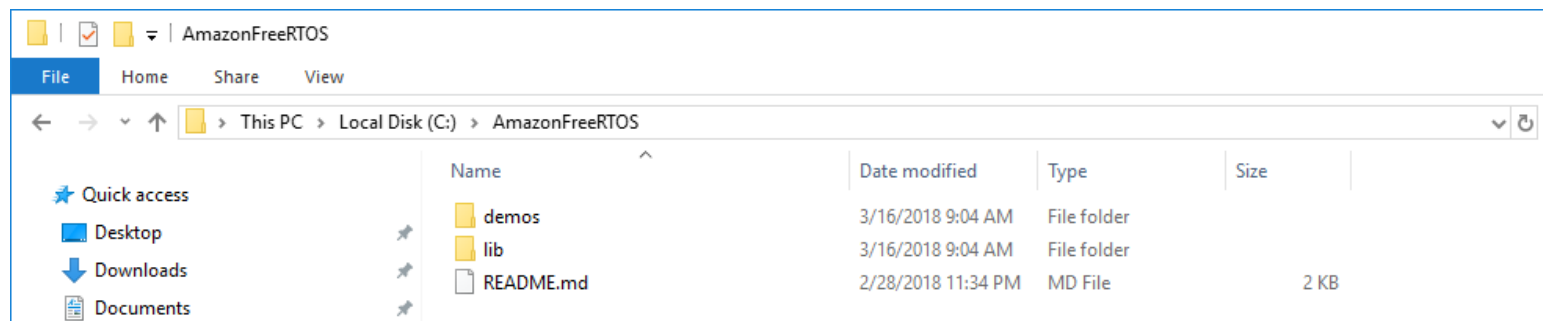
Software Configurations

Show all [Create new](#)

Type	Configuration	Hardware platform		
Predefined	Connect to AWS Greengrass - Microchip	Curiosity PIC32MZEF	Download	...
Predefined	Connect to AWS Greengrass - NXP	LPC54018 IoT Module	Download	...
Predefined	Connect to AWS Greengrass - ST	STM32L4 Discovery kit IoT node	Download	...
Predefined	Connect to AWS Greengrass - TI	CC3220SF-LAUNCHXL	Download	...
Predefined	Connect to AWS Greengrass - Windows	Windows Simulator	Download	...
Predefined	Connect to AWS IoT - Microchip	Curiosity PIC32MZEF	Download	...
Predefined	Connect to AWS IoT - NXP	LPC54018 IoT Module	Download	...
Predefined	Connect to AWS IoT - ST	STM32L4 Discovery kit IoT node	Download	...
Predefined	Connect to AWS IoT - TI	CC3220SF-LAUNCHXL	Download	...
Predefined	Connect to AWS IoT - Windows	Windows Simulator	Download	...

Step 2 – a:FreeRTOS Download

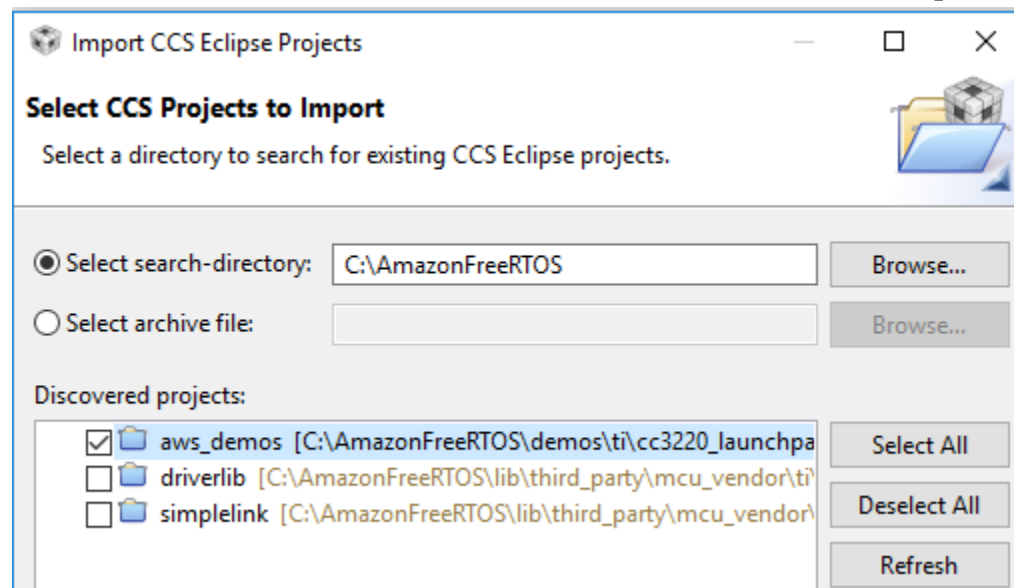
Unzip the sample project and place it in your root C:\



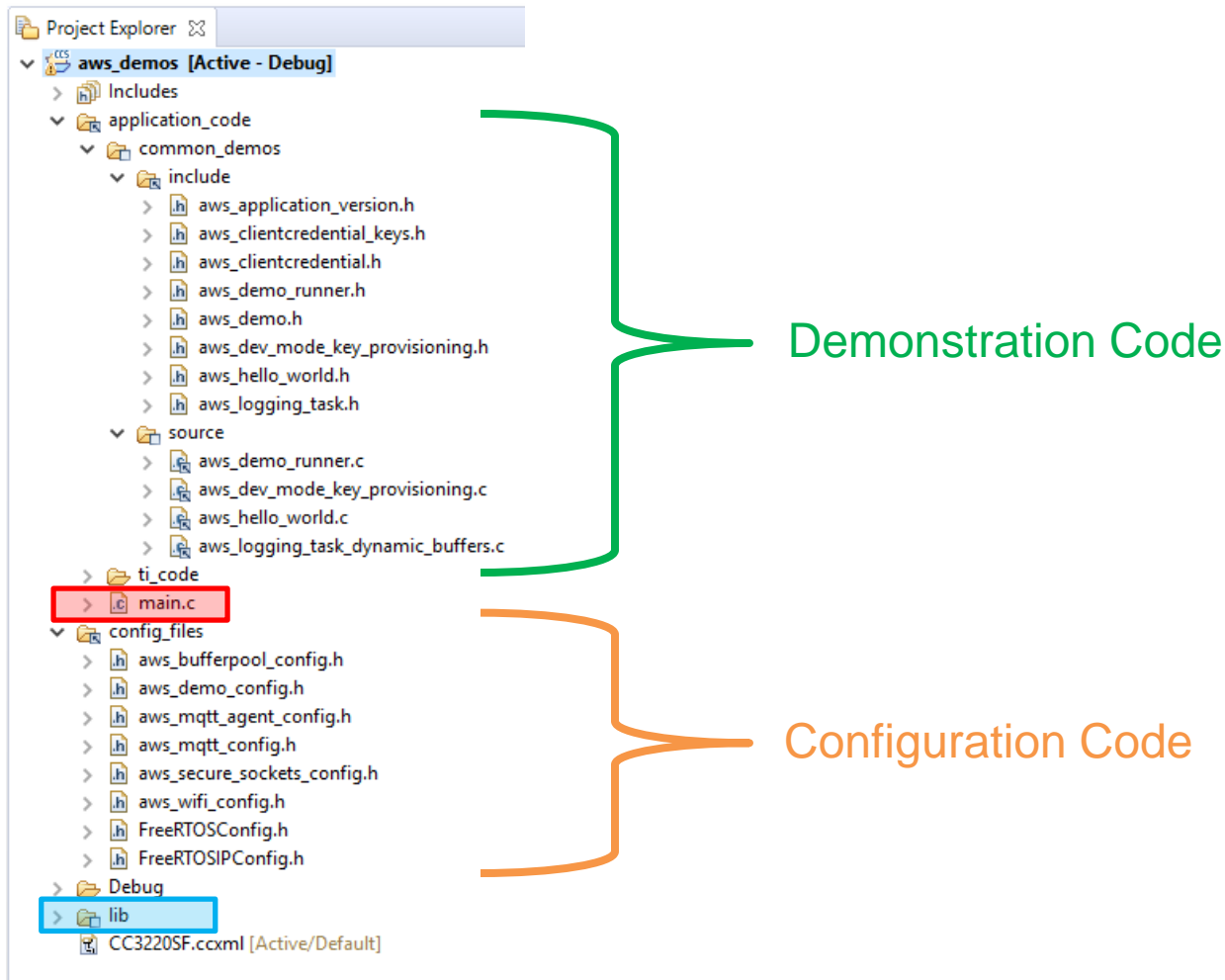
Note: Windows is limited to 260 characters for a path and the longest path in a:FreeRTOS is 122 characters!

Step 3 – Import a:FreeRTOS

- 1) Open Code Composer
- 2) Project -> Import CSS Project
- 3) Select the a:FreeRTOS Directory



Step 3 – Import a:FreeRTOS



Step 4 – Configure for AWS

```
aws_clientcredential.h
23 * http://www.FreeRTOS.org
24 */
25
26
27 #ifndef __AWS_CLIENTCREDENTIAL_H__
28 #define __AWS_CLIENTCREDENTIAL_H__
29
30 /*
31 * Include for device certificate and private key
32 */
33 #include "aws_clientcredential_keys.h"
34
35 /*
36 * MQTT Broker endpoint.
37 */
38 static const char clientcredentialMQTT_BROKER_ENDPOINT[] = "Paste AWS IoT Broker endpoint here.";
39
40
41 /* Use of a "define" and not a "static const" here to be able to
42 * use pre-compile concatenation on the string. */
43 #define clientcredentialIOT_THING_NAME "Paste AWS IoT Thing name here."
44
```

Step 4 – Configure for AWS

Login to the AWS IoT Console



Takes you to
the console

Amazon FreeRTOS Device Software

Amazon FreeRTOS is an operating system for microcontrollers that makes it easy to securely connect IoT devices locally or to the cloud. You can use a predefined configuration or create your own to get started.

Already downloaded your software? [Learn more](#) about next steps.

Software Configurations

Show all ▾

Find a configuration

Create new

Type ▾	Configuration	Hardware platform		
Predefined	Connect to AWS Greengrass - Microchip	Curiosity PIC32MZEF	Download	⋮

Step 4 – Configure for AWS

The screenshot shows the AWS IoT console interface. On the left, a navigation menu includes 'Monitor', 'Onboard', 'Manage', 'Secure', 'Act', and 'Test'. Below this, 'Software' is expanded to show 'Settings' (highlighted with a red box) and 'Learn'. The main content area is titled 'Custom endpoint' and is marked as 'ENABLED'. It contains explanatory text and a redacted 'Endpoint' field. A red arrow points from the 'Settings' menu item to the 'Custom endpoint' section. A green arrow points from the redacted endpoint field to a code snippet where the endpoint value is being assigned to a variable. The code snippet is as follows:

```
36 * MQTT Broker endpoint.  
37 */  
38 static const char clientcredentialMQTT_BROKER_ENDPOINT[] = "Paste AWS IoT Broker endpoint here.";
```

The text "Copy the endpoint" is written in green next to the green arrow.

Step 4 – Configure for AWS

```
aws_clientcredential.h ✕  
55 /*  
56 * WIFI network to join.  
57 */  
58 #define clientcredentialWIFI_SSID "Paste WiFi SSID here."  
59  
60 /*  
61 * Password needed to join WiFi network.  
62 */  
63 #define clientcredentialWIFI_PASSWORD "Paste WiFi password here."  
64  
65 /**  
66 * @brief Security type  
67 * WPA2 Security, @see WiFiSecurity_t  
68 * other values are - eWiFiSecurityOpen, eWiFiSecurityWEP, eWiFiSecurityWPA  
69 */  
70 #define clientcredentialWIFI_SECURITY eWiFiSecurityWPA2
```

Update for your settings!

Step 4 – Configure for AWS

```
aws_clientcredential_keys.h
1 /*
2  * PEM-encoded client certificate
3  *
4  * Must include the PEM header and footer:
5  * "-----BEGIN CERTIFICATE-----"
6  * "...base64 data..."
7  * "-----END CERTIFICATE-----";
8  */
9 static const char clientcredentialCLIENT_CERTIFICATE_PEM[] = "Paste client certificate here.";
10
11
12
13 /*
14  * PEM-encoded client private key.
15  *
16  * Must include the PEM header and footer:
17  * "-----BEGIN RSA PRIVATE KEY-----"
18  * "...base64 data..."
19  * "-----END RSA PRIVATE KEY-----";
20  */
21 static const char clientcredentialCLIENT_PRIVATE_KEY_PEM[] = "Paste client private key here.";
22
23
```


Step 4 – Configure for AWS

Certificate created!

Download these files and save them in a safe place. Certificates can be retrieved at any time, but the private and public keys cannot be retrieved after you close this page.

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

A certificate for this thing	[REDACTED]70f9.cert.pem	Download
A public key	[REDACTED]70f9.public.key	Download
A private key	[REDACTED]70f9.private.key	Download

Download All

You also need to download a root CA for AWS IoT from Symantec:

A root CA for AWS IoT [Download](#)

Activate

Step 4 – Configure for AWS

Navigate to:

aFreeRTOS\demos\common\devmode_key_provisioning\CertificateConfigurationTool\

Open CertificateConfigurator.html in a browser window.

Step 4 – Configure for AWS

Certificate Configuration Tool

Amazon FreeRTOS Developer Demos

Provide client certificate and private key PEM files downloaded from the AWS IoT Console.

Certificate PEM file:
 Browse...

Private Key PEM file:
 Browse...

Generate and save aws_clientcredential_keys.h

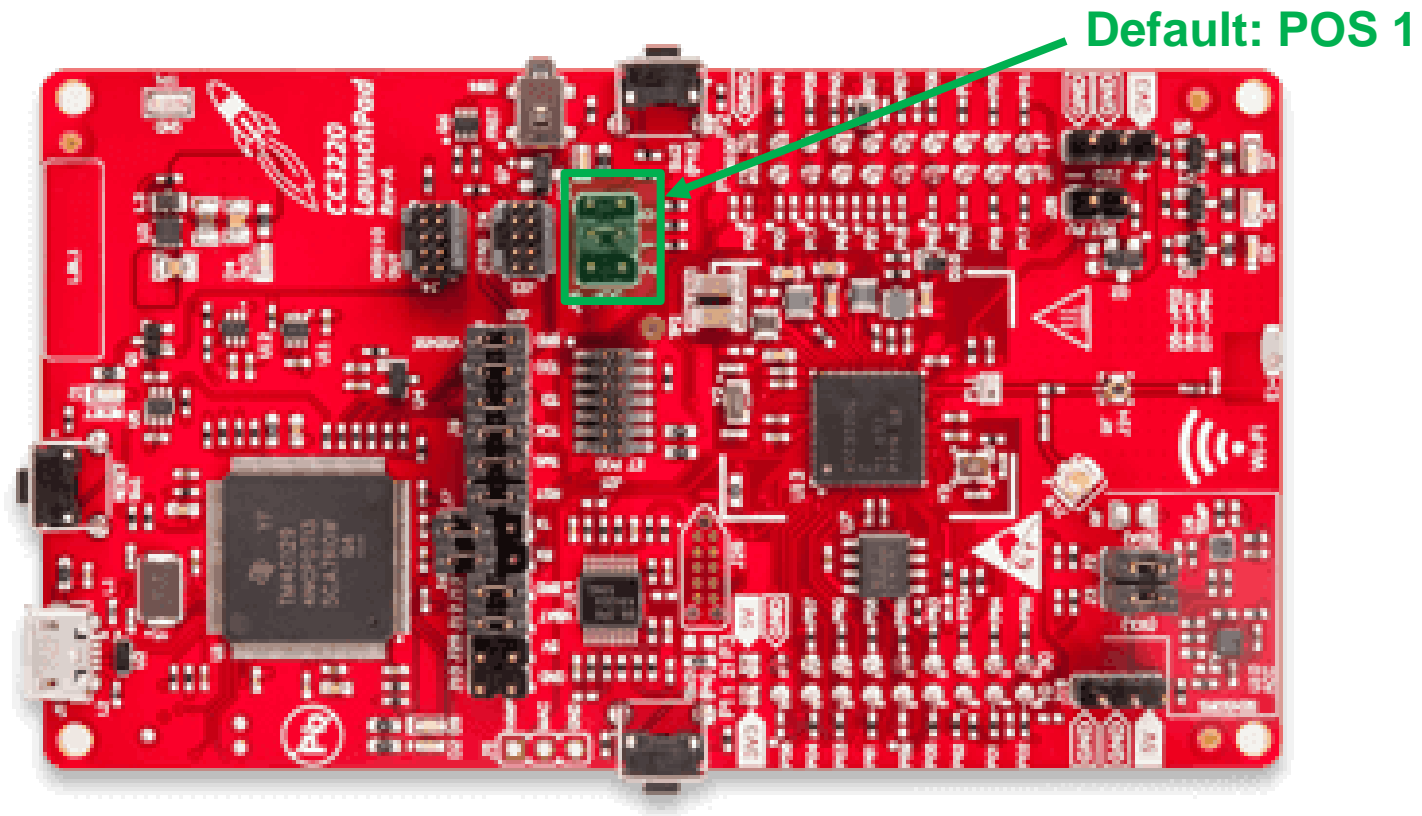
▲ Save the generated header file to the *demos/common/include* folder of the demo project.

Copyright (C) 2017 Amazon Web Services.

Replace existing `aws_clientcredential_key.h` with generated file!

Step 5 – Run the Example

Make sure the SOP is set to position 0



Step 5 – Run the Example

Compile the project

```
Console [aws_demos]
CDT Build Console [aws_demos]
"/lib/third_party/mcu_vendor/ti/SimpleLink_CC32xx/v1_40_01_00/kernel/freertos/posix/sleep_freertos.obj"
"/lib/third_party/mcu_vendor/ti/SimpleLink_CC32xx/v1_40_01_00/kernel/freertos/posix/timer_freertos.obj"
"/lib/third_party/mcu_vendor/ti/SimpleLink_CC32xx/v1_40_01_00/kernel/freertos/startup/startup_cc32xx_ccs.obj"
"C:/AmazonFreeRTOS/demos/ti/cc3220_launchpad/common/application_code/ti_code/CC3220SF_LAUNCHXL_FREERTOS.cmd" -llibc.a -
l"C:/AmazonFreeRTOS/lib/third_party/mcu_vendor/ti/SimpleLink_CC32xx/v1_40_01_00/source/ti/devices/cc32xx/driverlib/ccs/Release/driverlib.a"
l"C:/AmazonFreeRTOS/lib/third_party/mcu_vendor/ti/SimpleLink_CC32xx/v1_40_01_00/source/ti/drivers/lib/drivers_cc32xx.aem4" -
l"C:/AmazonFreeRTOS/lib/third_party/mcu_vendor/ti/SimpleLink_CC32xx/v1_40_01_00/source/ti/drivers/net/wifi/ccs/rtos/simplelink.a"
<Linking>
Finished building target: "aws_demos.out"

C:/ti/ccsv8/utils/tiobj2bin/tiobj2bin aws_demos.out aws_demos.bin C:/ti/ccsv8/tools/compiler/ti-cgt-arm_18.1.1.LTS/bin/armofd
C:/ti/ccsv8/tools/compiler/ti-cgt-arm_18.1.1.LTS/bin/armhex C:/ti/ccsv8/utils/tiobj2bin/mkhex4bin

**** Build Finished ****
```

Step 5 – Run the Example

The screenshot shows the AWS IoT console interface. On the left is a navigation menu with icons and labels for Monitor, Onboard, Manage, Secure, Act, and Test. The 'Test' icon is highlighted with a red box. A red arrow points from this box to the 'Subscription topic' input field. The input field contains the text 'freertos/demos/echo'. Another red arrow points from below the text to this input field. A third red arrow points from below the text to the 'Subscribe to topic' button, which is also highlighted with a red box. The main content area has the heading 'Subscribe' and the text 'Devices publish MQTT messages on topics. You can use this client to subscribe to a topic and receive these messages.'

Subscribe
Devices publish MQTT messages on topics. You can use this client to subscribe to a topic and receive these messages.

Subscription topic

freertos/demos/echo

Subscribe to topic

Test

Monitor
Onboard
Manage
Secure
Act

Subscribe to freertos/demos/echo

Step 5 – Run the Example

```
COM16 - PuTTY
22 3024 [MQTT] Received message 30000 from queue.
23 3125 [MQTT] MQTT Publish was successful.
24 3125 [MQTT] Notifying task.
25 3127 [MQTTEcho] Command sent to MQTT task passed.
26 3127 [MQTTEcho] Echo successfully published 'Hello World 0'
27 3127 [Echoing] Sending command to MQTT task.
28 3127 [MQTT] Received message 40000 from queue.
29 3329 [MQTT] MQTT Publish was successful.
30 3329 [MQTT] Notifying task.
31 3331 [Echoing] Command sent to MQTT task passed.
32 3331 [Echoing] Message returned with ACK: 'Hello World 0 ACK'
33 8127 [MQTTEcho] Sending command to MQTT task.
34 8127 [MQTT] Received message 50000 from queue.
35 8328 [MQTT] MQTT Publish was successful.
36 8328 [MQTT] Notifying task.
37 8330 [MQTTEcho] Command sent to MQTT task passed.
38 8330 [MQTTEcho] Echo successfully published 'Hello World 1'
39 8330 [Echoing] Sending command to MQTT task.
40 8330 [MQTT] Received message 60000 from queue.
41 8432 [MQTT] MQTT Publish was successful.
42 8432 [MQTT] Notifying task.
43 8434 [Echoing] Command sent to MQTT task passed.
44 8434 [Echoing] Message returned with ACK: 'Hello World 1 ACK'
```

Step 5 – Run the Example

Subscriptions

freertos/demos/echo Export Clear Pause

Subscribe to a topic

Publish to a topic

freertos/demos/echo ×

Publish

Specify a topic and a message to publish with a QoS of 0.

freertos/demos/echo Publish to topic

```
1 {
2   "message": "Hello from AWS IoT console"
3 }
```

freertos/demos/echo Mar 17, 2018 8:39:30 AM -0400 Export Hide

We cannot display the message as JSON, and are instead displaying it as UTF-8 String.

Hello World 7 ACK

freertos/demos/echo Mar 17, 2018 8:39:29 AM -0400 Export Hide

We cannot display the message as JSON, and are instead displaying it as UTF-8 String.

Hello World 7

freertos/demos/echo Mar 17, 2018 8:39:24 AM -0400 Export Hide

Additional Resources

- Download Course Material for
 - C/C++ Doxygen Templates
 - Example source code
 - Blog
 - YouTube Videos
- Embedded Bytes Newsletter
 - <http://bit.ly/1BAHYXm>



From www.beningo.com under

- Blog > CEC – Connecting Edge Devices to the IoT using Amazon FreeRTOS