Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

CYW954907AEVAL1F

Evaluation Kit User Guide

Document Number: 002-22338 Rev. *A

Cypress Semiconductor
An Infineon Technologies Company
198 Champion Court
San Jose, CA 95134-1709
www.cypress.com
www.infineon.com

Copyrights

Contents

Safety Information

The CYW954907AEVAL1F EVK is intended for use as a development platform for hardware or software in a laboratory environment. The board is an open-system design, which does not include a shielded enclosure. Due to this reason, the board may cause interference with other electrical or electronic devices in close proximity. In a domestic environment, this product may cause radio interference. In such cases, take adequate preventive measures. Also, do not use this board near any medical equipment or RF devices.

Attaching additional wiring to this product or modifying the product operation from the factory default may affect its performance and cause interference with other apparatus in the immediate vicinity. If such interference is detected, suitable mitigating measures must be taken.

	The CYW954907AEVAL1F contains electrostatic discharge (ESD)-sensitive devices. Electrostatic charges readily accumulate on the human body and any equipment, and can discharge without detection. Permanent damage may occur on devices subjected to high-energy discharges. Proper ESD precautions are recommended to avoid performance degradation or loss of functionality. Store unused kits in the protective shipping package.

	End-of-Life/Product Recycling This kit has an end-of-life cycle of five years from the year of manufacturing mentioned on the back of the box. Contact your nearest recycler for discarding the kit.

General Safety Instructions

ESD Protection

ESD can damage boards and associated components. Cypress recommends that the user perform procedures only at an ESD workstation. If an ESD workstation is not available, use appropriate ESD protection by wearing an antistatic wrist strap attached to the chassis ground (any unpainted metal surface) on the board when handling parts.

Handling Boards

CYW954907AEVAL1F boards are sensitive to ESD. Hold the board only by its edges. After removing the board from its box, place it on a grounded, static-free surface. Use a conductive foam pad if available. Do not slide the board over any surface. Any physical action on the kit such as changing wires, jumper settings, or measuring voltages can cause stress on the kit printed circuit board assembly (PCBA). You must ensure that the PCBA has proper support on the bottom side to avoid stress on the PCBA when the EVK is in operation.

1. Introduction

Thank you for your interest in the CYW954907AEVAL1F Evaluation Kit (EVK). The EVK enables customers to evaluate and develop single-chip Wi-Fi applications using CYW54907 devices.

The EVK uses WICED Studio 6.0 (or later) to develop and debug your CYW54907 project. It offers footprint-compatibility with Arduino shields. In addition, the kit features an RJ-45 Ethernet connector, and an onboard programmer/debugger and serial bridge chip. The EVK supports only 3.3 V as the operating voltage.

WICED Studio 6.0 (or later) supports application development using a WICED development board (CYW954907AEVAL1F). The development system is compatible with Windows, macOS, and Linux. This document provides instructions for utilizing peripherals, such as I2C or SPI, in WICED sample applications using the WICED Studio IDE.

Note: This document applies to WICED Studio 6.0 (or later).

The CYW954907AEVAL1F EVK is available through the Cypress Online Store or through our distributors.

1.1 CYW954907AEVAL1F EVK Contents

The CYW954907AEVAL1F EVK includes the following:

- One CYW954907AEVAL1F Evaluation Board with assembled Arduino headers
- One USB 2.0 Type-A to Micro-B cable

Figure 1-1. CYW954907AEVAL1F Kit Contents

Inspect the contents of the kit. If you find any part missing, contact your nearest Cypress sales office for assistance: www.cypress.com/support.

Hardware Not Included with the Kit

The EVK does not come with all the hardware needed to perform the demonstrations documented in this guide.

The following hardware is not included with this kit:

- RJ-45 Ethernet cable
- External power supply
- Dual external antenna
- Potentiometer
- Jumper Wires
- SD card

1.2 Board Details

The board consists of the blocks shown in Figure 1-2 and Figure 1-3.

1. Reset Switch (SW2)
2. RJ45 Connector (J14)
3. Micro USB (Programming and Debugging) (J5)
4. 5-12V Power Input (J8)
5. WICED Header (J6)
6. Arduino Header (J13)
7. User Switch 1 (SW3)
8. User Switch 2 (SW1)
9. Arduino Header (J9)
10. PCB Antenna-Main (ANT1)
11. Connector for External Antenna 1 (J1)
12. CYW54907 Type 1PS Module (Murata) (U14)
13. PCB Antenna-Diversity (ANT0)
14. Connector for External Antenna 0 (J2)
15. On-board /External JTAG Switch (SW4)
16. External JTAG Header (J3)
17. Arduino Header (J10)
18. Arduino Header (J12)
19. External PHY chip (U12) - BCM5241
20. External ADC Chip (U3)
21. Micro SD Connector/slot (J7)

Figure 1-2. CYW954907AEVAL1F Evaluation Board

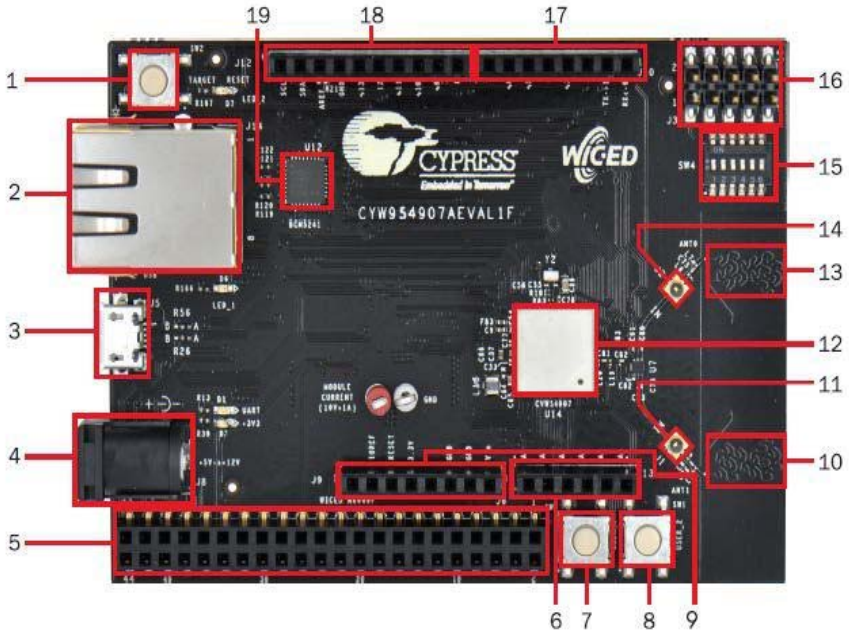
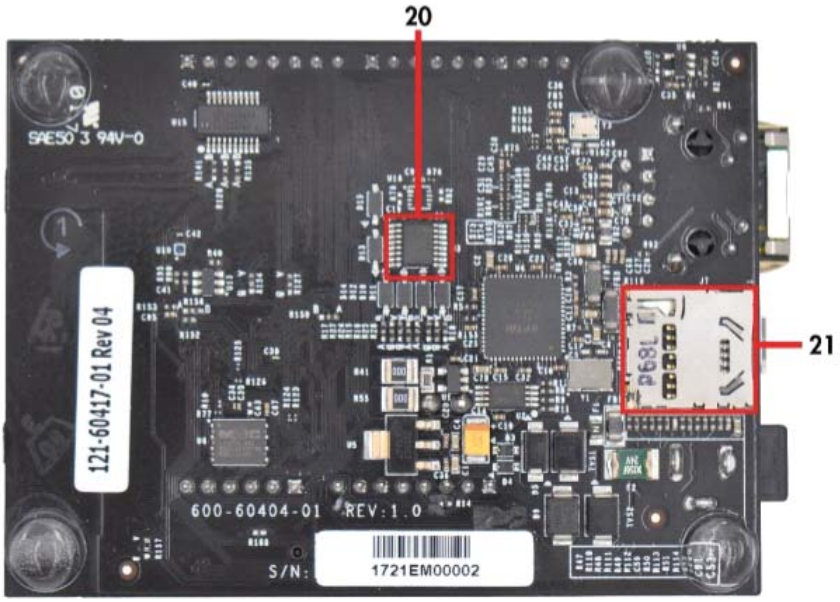


Figure 1-3. CYW954907AEVAL1F Evaluation Board (Back View)



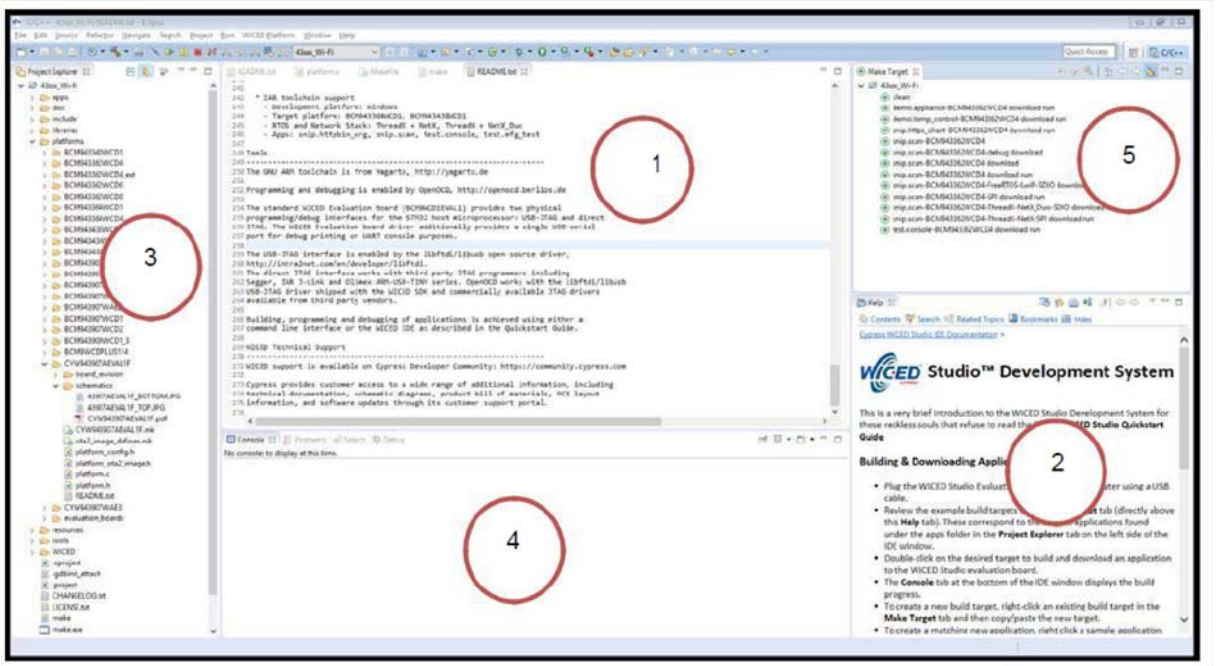
1.3 WICED Studio Development System Overview

WICED Studio 6.0 (or later) supports application development using the WICED Evaluation Board (CYW954907AEVAL1F EVK). Tabs and their location in the WICED IDE are as shown in Figure 1-4.

Figure 1-4 illustrates the following:

1. Edit your application firmware.
2. Help Window that contains instructions on building and downloading applications.
3. Explore existing applications/firmware and library of the Software Development Kit (SDK).
4. View Build messages in the Console window.
5. Create and edit Make Targets for the platform to build your Application/Project.

Figure 1-4. WICED IDE



1.4 WICED Studio Code Examples

WICED Studio includes libraries and code examples supporting both Bluetooth and Wi-Fi platforms. Selecting the 43xxx_Wi-Fi Filter will show only Wi-Fi platform related files in the project explorer as shown in Figure 1-5.

Application examples can speed up the design process by serving as templates for development. Code examples are located under the apps category (in the Project explorer window), as shown in Figure 1-6. Code examples under *apps* are further grouped into demo, snip, test, waf (WICED Application Framework), and wwd (WICED Wi-Fi Driver Application) directories.

The *demo* directory contains applications that combine various WICED features into a single application. The snip directory contains application snippets that demonstrate how to use various WICED libraries and API functions. The test directory contains applications that are used for simple test and utility. The waf directory contains applications that are part of WICED Application framework, for instance, the bootloader. The wwd directory contains applications that are developed using the low level wwd API calls and do not rely on higher level WICED APIs. Located within each subdirectory in the apps folder is a README.txt that lists and summarizes the applications located within the folder. It should also be noted that not all applications are supported in all platforms. The snip directory contains a README.txt with a matrix on what applications are supported in what platforms. For more details on the WICED software stack and APIs, review the Application notes and documents available in the doc folder <WICED SDK installation folder>/WICED-Studio-6.0/43xxx_Wi-Fi/doc. WICED-QSG204 available in the same path is a good document to start with.

Figure 1-5. Filter for Wi-Fi Code Example in WICED Studio

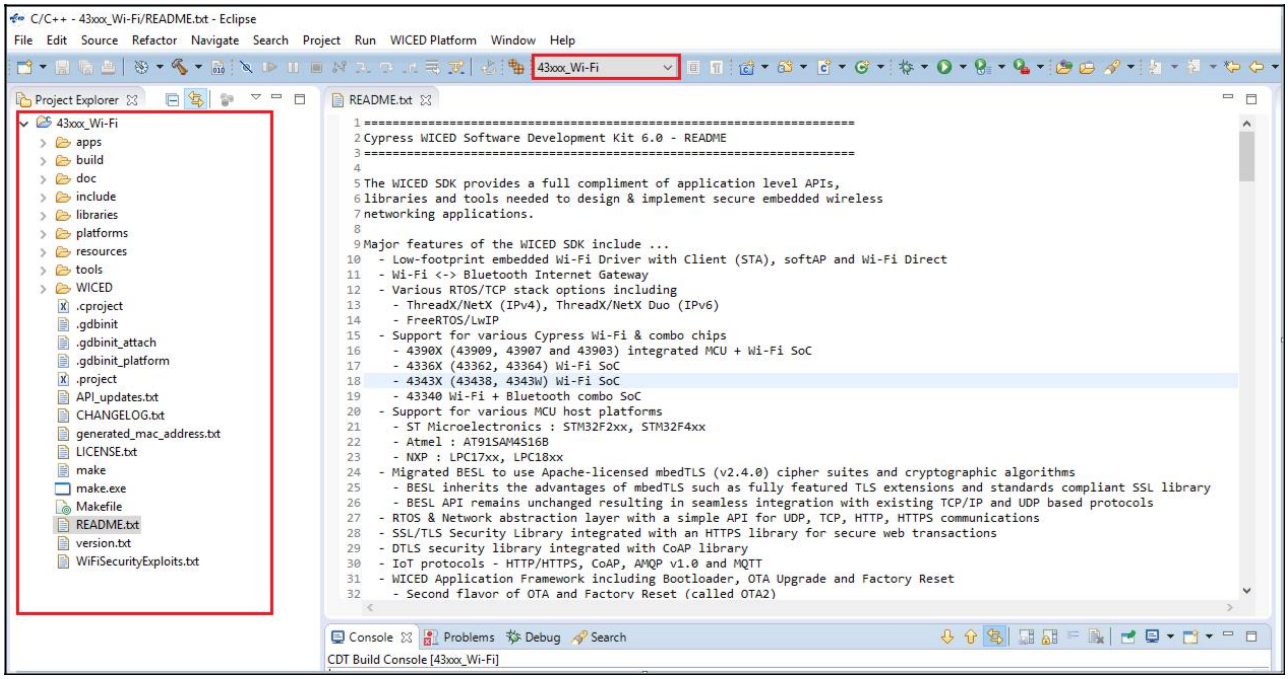
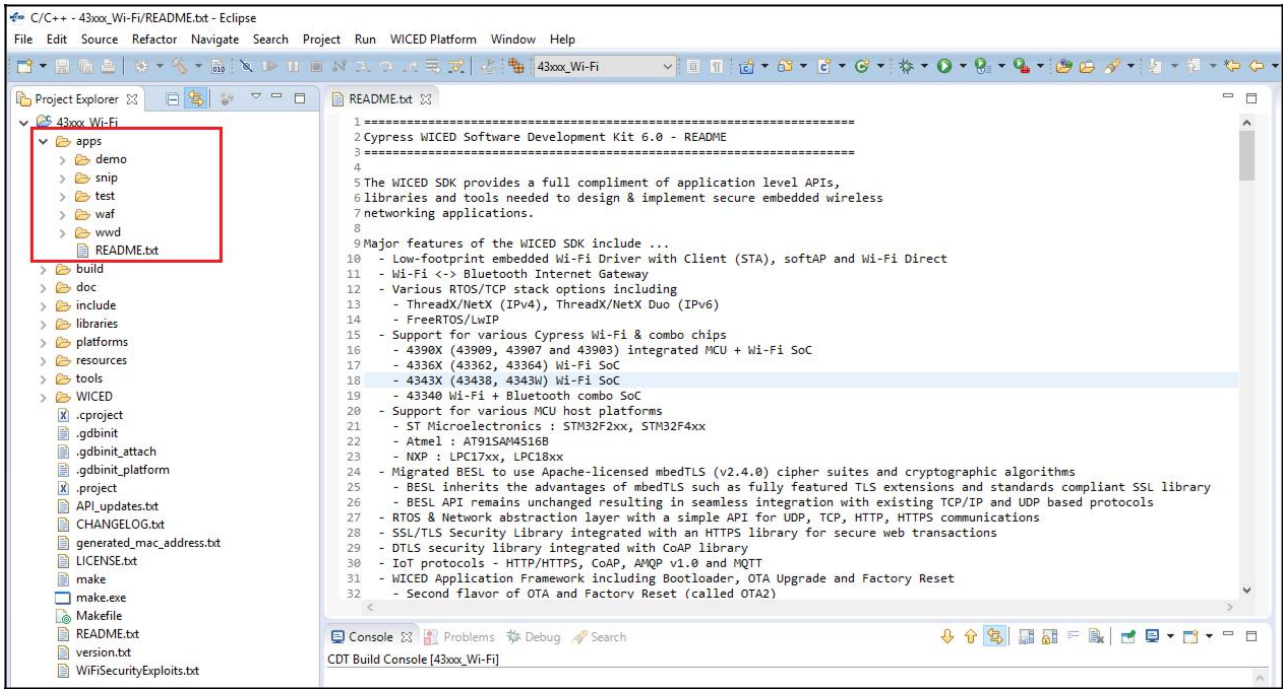


Figure 1-6. Code Examples under *apps* Category



1.5 Kit Code Examples

In addition to the examples available in WICED Studio, this EVK includes a few additional code examples, which can be used to quickly evaluate CYW54907 using this kit. These examples are described in the Code Examples chapter.

1.6 Getting Started

To learn quickly about CYW954907AEVAL1F EVK, refer to the CYW954907AEVAL1F Quick Start Guide inside the kit box.

This user guide will help you get acquainted with CYW954907AEVAL1F EVK:

- The Software Installation chapter describes the installation of the kit software. This includes extracting the required files for WICED Studio 6.0 (or later).
- The Kit Operation chapter describes the major sections of the kit such as the on-board programmer/debugger chip, reset control, headers, programming and debugging of the kit, and Ethernet interface.
- The Hardware chapter describes the CYW954907AEVAL1F EVK hardware and its different blocks.
- The Code Examples chapter describes code examples that will help you understand how to get started with WLAN basic examples.

1.7 IoT Resources and Technical Support

Cypress provides a wealth of data at www.cypress.com/internet-things-iot to help you to select the right IoT device for your design, and quickly and effectively integrate the device into your design. Cypress provides customer access to a wide range of information, including technical documentation, schematic diagrams, product bill of materials, PCB layout information, and software updates. Customers can acquire technical documentation and software from the Cypress Support Community website (https://community.cypress.com). For assistance, go to: www.cypress.com/support.

1.8 Additional Learning Resources

Visit CYW954907AEVAL1F EVK and CYW54907 for additional learning resources including datasheets and application notes.

1.9 Document Conventions

Table 1-1. Document Conventions for Guides

Convention	Usage
Courier New	Displays file locations, user entered text, and source code: C:\ ...cd\icc\
Italics	Displays file names and reference documentation.
Bold	Displays keyboard commands in procedures: **Enter or Ctrl+C**
File > Open	Represents menu paths: File > Open > New Project
Bold	Displays commands, menu paths and icon names in procedures: Click the File icon and then click Open.
Times New Roman	Displays an equation: 2 + 2 = 4
Text in gray boxes	Describes Cautions or unique functionality of the product.

1.10 Acronyms

Table 1-2. List of Acronyms used in this Document

Acronym	Definition
SPI	Serial Peripheral Interface
EVK	Evaluation Kit
SDK	Software Development Kit
WICED	Wireless Internet Connectivity for Embedded Devices
JTAG	Joint Test Action Group
I^2C	Inter-Integrated Circuit
MQTT	Message Queue Telemetry Transport
POR	Power-on-Reset
PMU	Power Management Unit
VTRIM	Voltage Trimming
LPO	Low Power Oscillator
GPIO	General Purpose Input Output
UART	Universal Asynchronous Receiver/Transmitter
AWS	Amazon Web Services
IDE	Integrated Development Environment
WLAN	Wireless Local Area Network

2. Software Installation

This chapter describes the steps to install the software tools and packages on a PC for using the CYW954907AEVAL1F EVK. This includes the WICED IDE in which the projects will be built and used for programming.

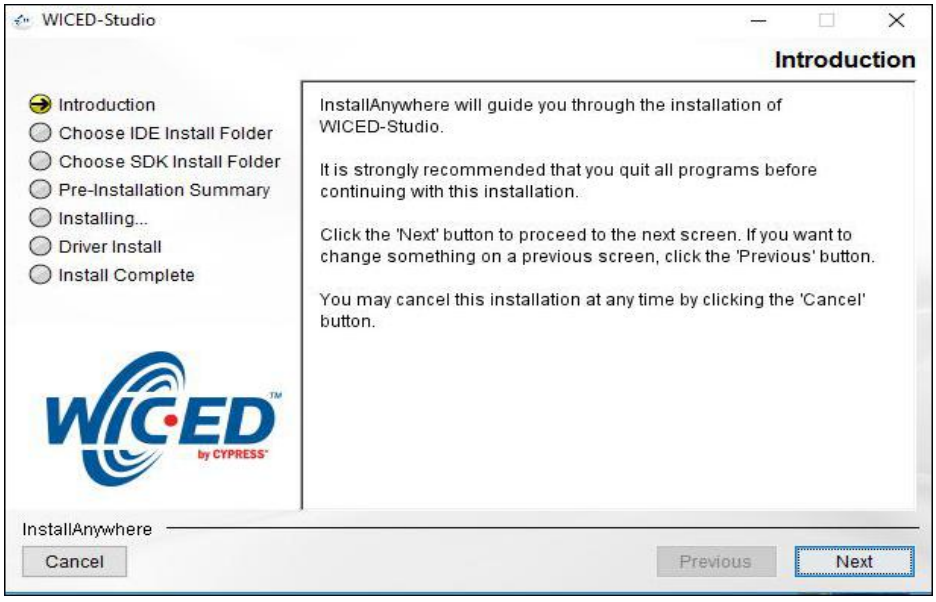
2.1 Before You Begin

All Cypress software installations require administrator privileges. Ensure that you have the required privileges on the system for successful installation. Before you install the kit software, close any other Cypress software that is currently running. Ensure you have installed WICED Studio 6.0 (or later).

2.2 Install Software

Follow these steps to install the CYW954907AEVAL1F Evaluation Kit software:

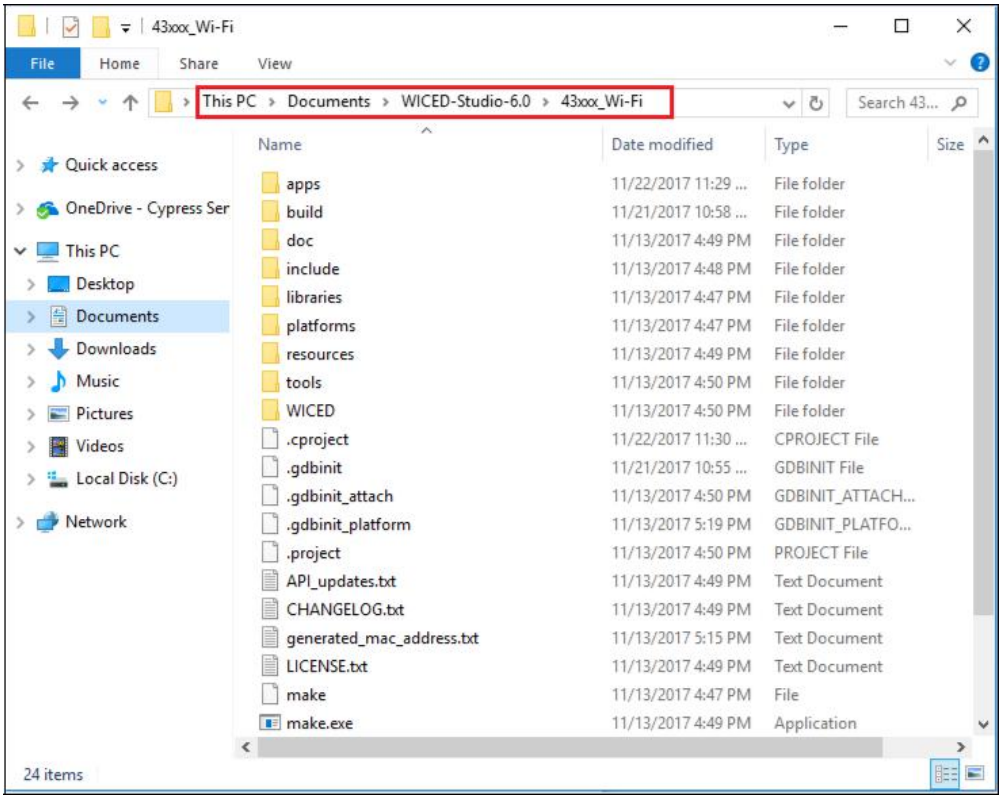
1. Download and install WICED Studio 6.0 (or later) from this web page. The following is a screenshot of the Installer Window when opened.



2. Select two folders, one for the IDE and the other for the SDK. The folder for the SDK contains the framework for developing Wi-Fi applications.
3. As a last step in installation, installer will ask to select between Wi-Fi and Bluetooth platform. Select **43xxx_Wi-Fi** as default.

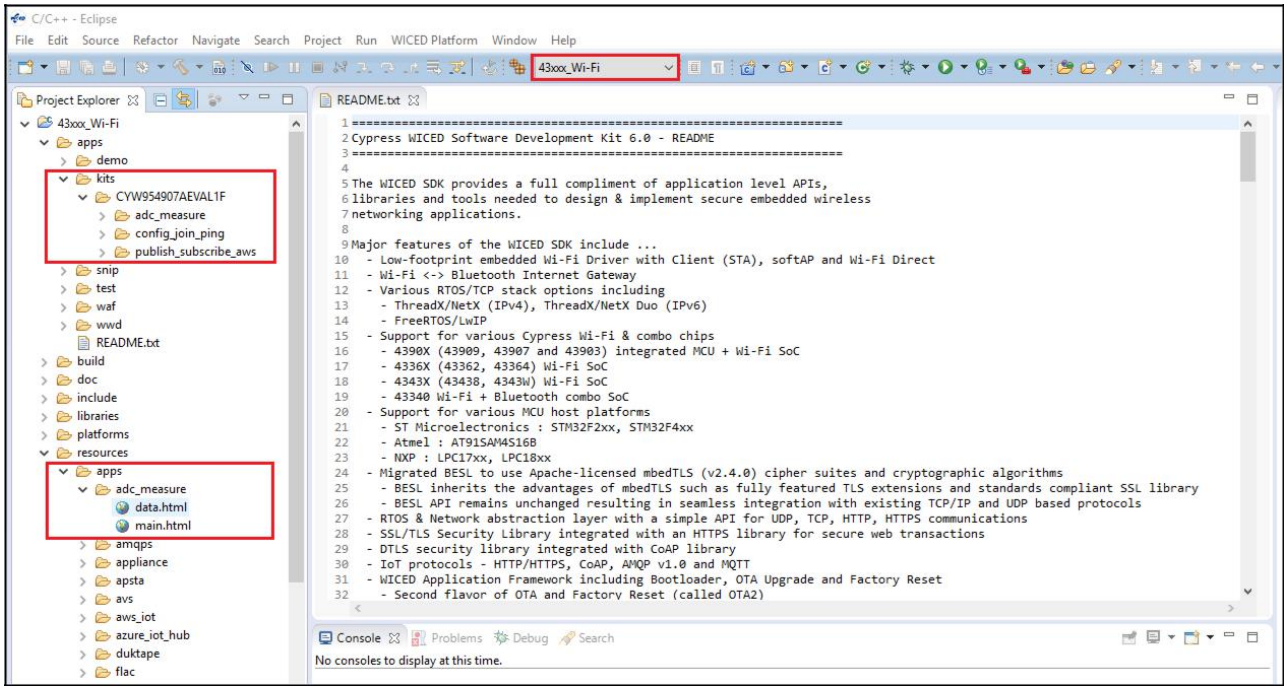
4. Download the CY954907AEVAL1F_KitPackage.zip software from here. The software is available as a zip file.
5. Locate the WICED Wi-Fi-SDK directory in your PC. The default location is *C:\Users\<user name>\Documents\WICED-Studio-6.0\43xxx_Wi-Fi*, as shown in Figure 2-1. However, it may be in a different location depending on the path you choose when installing WICED Studio.
USB to serial UART with a 3.3V TTL Adapter cable allows to connect between the host or computer and CYW89072EVAL. This connection refers as a regular serial communication

Figure 2-1. WICED SDK Directory



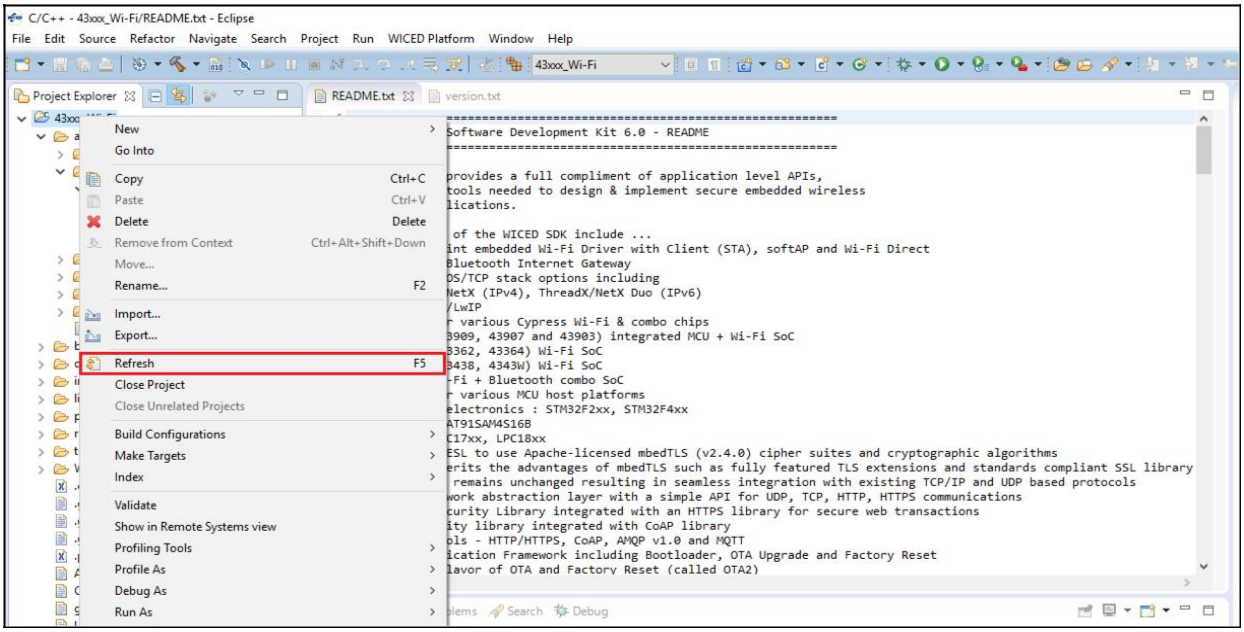
6. Copy the *CY954907AEVAL1F_KitPackage.zip* file and extract to a temporary location such as "temp". The zip file will extract two directories called "apps" and "resources" inside the *temp/CYW954907AEVAL1F_KitPackage/* directory. Select both of them, Copy (Ctrl+C) and paste (Ctrl+V) into *C:\Users\<user name>\Documents\WICED-Studio-6.0\43xxx_Wi-Fi.* Choose the option to merge with existing folders.
Alternately, copy the *CY954907AEVAL1F_KitPackage.zip* to the location specified above and use the **Extract Here** option if you have 7-Zip or another unzip utility. The zip file should be merged to the existing folders. If WICED Studio 6.0 (or later) is opened with *43xxx_Wi-Fi* as the WICED Filter (Figure 1-5), then the new folders appear as shown in Figure 2-2.

Figure 2-2. Setup Package in WICED Studio 6.0 (or later)



7. The *CY954907AEVAL1F_KitPackage.zip* package contains three code examples which add to the existing set of examples available in WICED Studio 6.0 or later. Unzipping creates the kits directory under apps, and a*dc_measure* in the *resources\apps* directory.
   After unzipping, if the projects are not visible in WICED Studio 6.0 (or later), then right-click the top most folder (*43xxx_Wi-Fi)* and click **Refresh**, as shown in Figure 2-3.

Figure 2-3. **Refresh** Top Folder



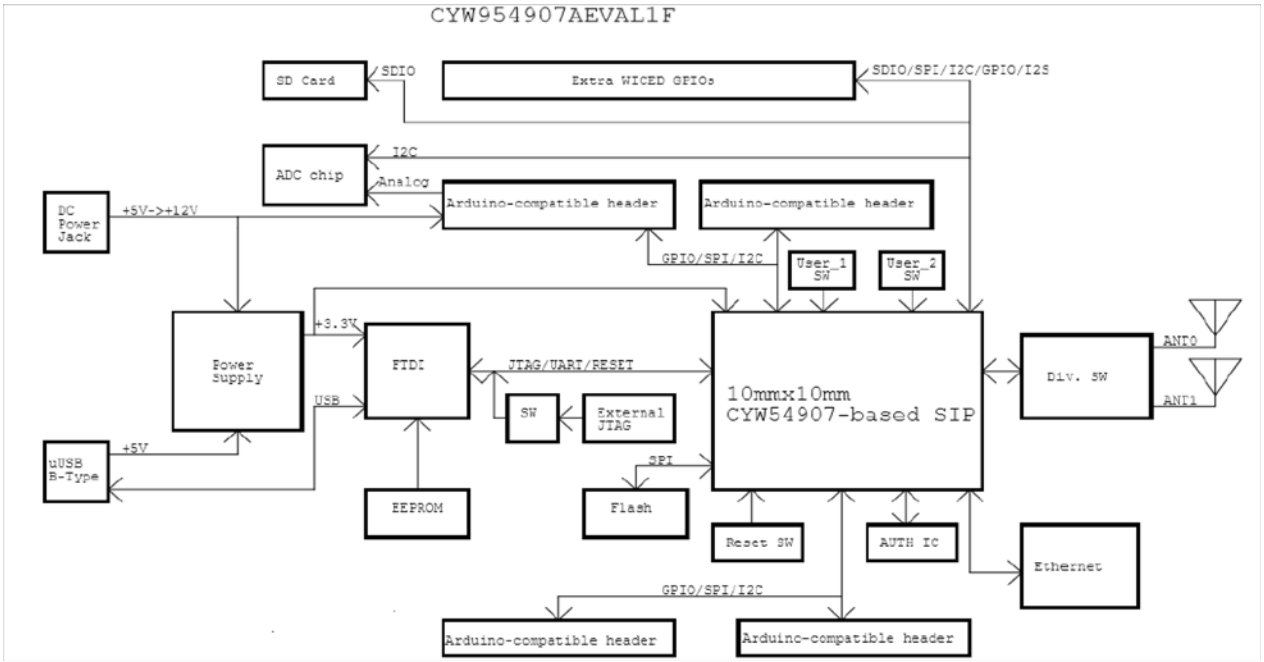
3. Kit Operation

This chapter introduces you to the CYW954907AEVAL1F EVK and the features that will be used as part of the kit operation. Features such as Wi-Fi connection and programming/debugging are discussed in this chapter. The chapter also describes the USB-UART that can be used to communicate with the CYW54907 device on this EVK.

3.1 Theory of Operation

Figure 3-1 illustrates the block diagram of the CYW954907AEVAL1F EVK. This board contains CYW54907-based SiP, which is a Type 1PS Wireless module. This module is an embedded network controller solution from Murata. This board also contains a USB-Serial interface / JTAG programmer / debugger. This board features Arduino form-factor-compatible headers, which enables Arduino shields to be plugged on top, extending its capabilities. This board also features two user switches, two user LEDs, an RJ-45 connector for Ethernet, and a reset switch for the wireless module.

Figure 3-1. Block Diagram of CYW954907AEVAL1F EVK



3.2 On-board Programmer/Debugger and Serial Interface Chip

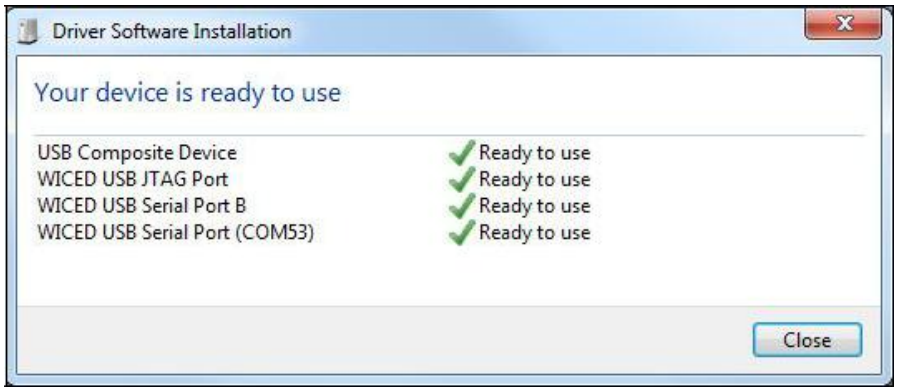
An FT-2232-HQ chip is used for onboard programming, debugging and USB-Serial functionality. It connects to the computer over a USB interface and connects to the CYW54907-based SiP module over JTAG and UART pins. Alternatively, you can use the External JTAG connector (J3) along with switch **SW4** (in all closed positions) to use JTAG from connectors such as Olimex.

3.3 CYW954907AEVAL1F Kit Connection

The CYW954907AEVAL1F EVK can be powered by the following options: External power supply and USB.

When using an external power supply, use a 5 V - 12 V, 2A power supply with 2.1-mm DC Jack (center pin positive). When powered from USB, there are two logical USB devices: a USB-JTAG device and a USB-UART device. Drivers for the CYW954907AEVAL1F EVK are automatically installed during the WICED SDK installation process. When you connect the kit for first time to your PC, it will initiate the driver search as shown in Figure 3-2.

Figure 3-2. Driver Software Installation



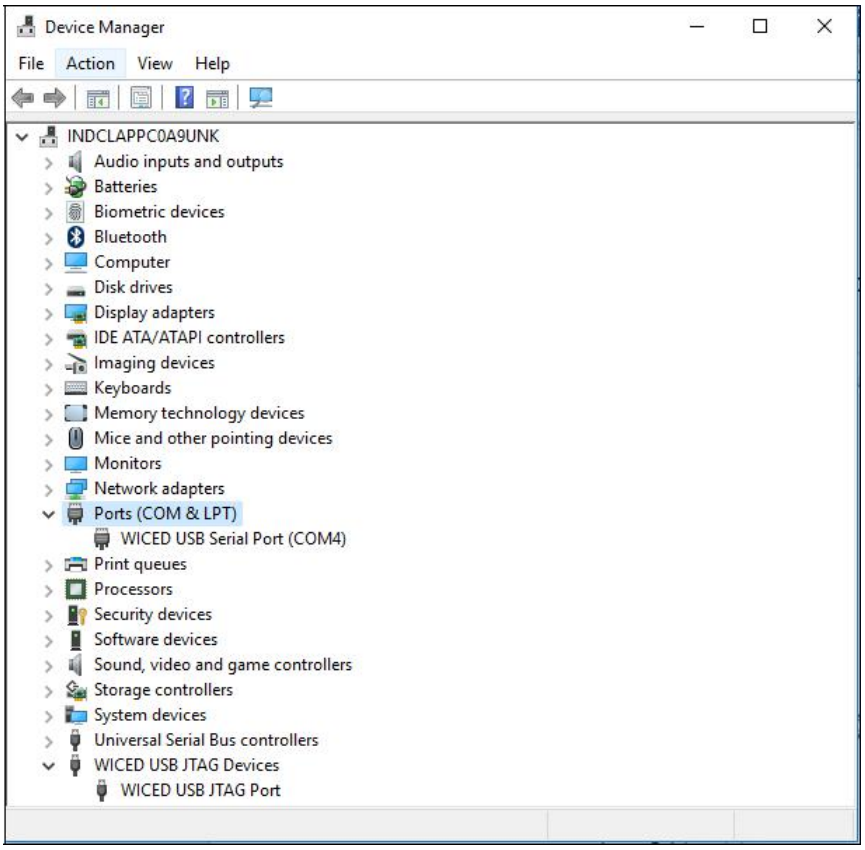
3.3.1 Verifying Driver Installation

Do the following to verify the successful completion of driver installation:

1. Right-click **My Computer > Properties**.
2. In the System Properties window, select **Device Manager**.
   a. The WICED USB Serial Port is listed under Ports (COM & LPT) as shown in Figure 3-3.
   b. The WICED USB JTAG Port is listed under **WICED USB JTAG Devices** as shown in Figure 3-3.

The Device Manager window identifies the WICED USB Serial COM port as COMXX. The assigned port number varies between systems. If the device displays two WICED USB Serial Ports (WICED USB Serial port and WICED USB JTAG Port) instead of one, then follow the link mentioned in this post.

Figure 3-3. Verifying Device Driver Installation



3.3.2 Troubleshooting

If an error occurred during the automatic driver installation process, the driver may be manually installed from the following directory: <*WICED-SDK*>*Drivers**Windows**wiced_uart.*

If the CYW954907AEVAL1F EVK does not appear in the Device Manager, verify that the +3V3 LED is turned ON and check the USB cable.

3.3.3 External Power Supply

The CYW954907AEVAL1F EVK can be supplied using an external power supply (5V-12V, 2A), using a 2.5 mm DC Jack with center pin positive. When using an external power supply and also connecting a USB cable (for programming/debugging or USB-UART), the voltage on the external power supply should be greater than that of the USB supply; if not, the kit will be actually sourcing its power from USB rather than the external power supply.

3.4 Building, Programming, and Debugging

3.4.1 Building and Programming a Project in WICED Studio IDE

Do the following to build and program a project for CYW954907AEVAL1F EVK:

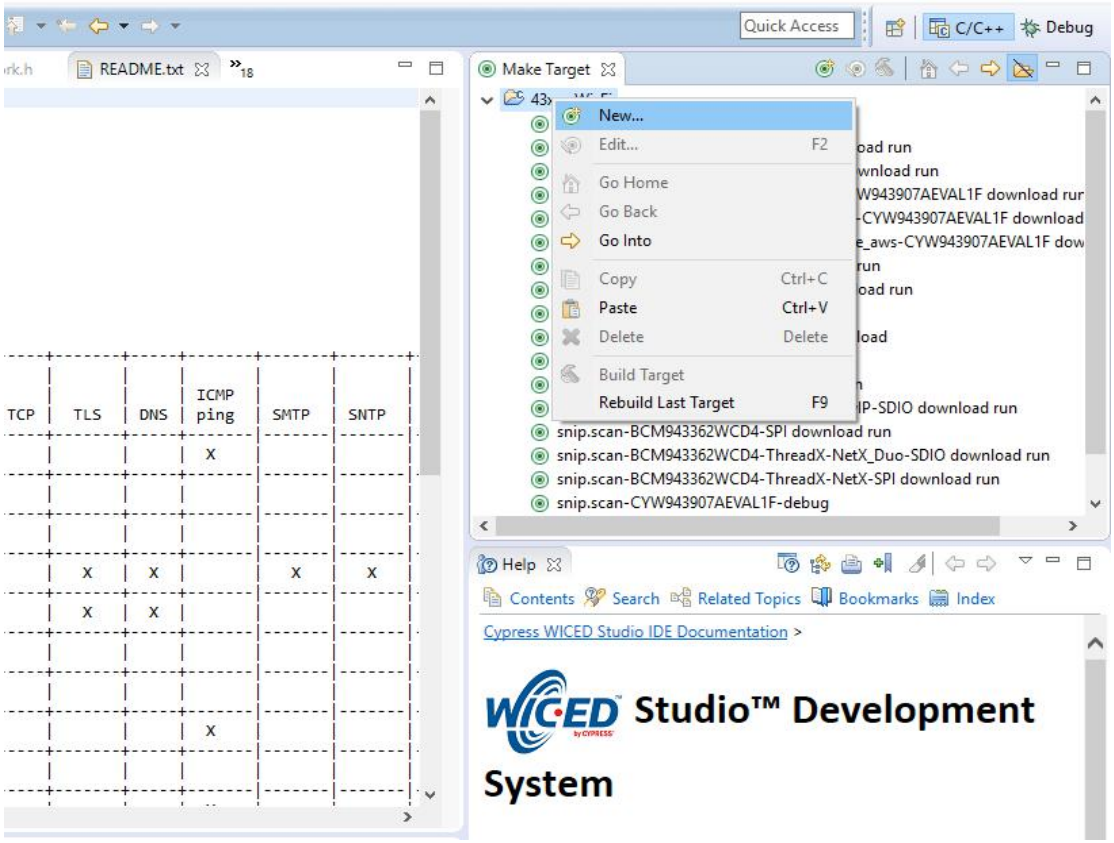
1. Open the WICED IDE on Windows PC: go to **Start > All Programs > Cypress > WICED-Studio**.
2. Select **43xxx_Wi-Fi** in the WICED Target selector drop-down box as shown in Figure 3-4. Building a project requires a corresponding make target, located in the Make Target window. All applications should go under the *apps* directory. The make target path will contain the directory hierarchy starting from *apps* with directory names separated by a period. The project name is followed by a hyphen and then the platform name. Finally, the actions to be performed after the build are specified such as download and run. For example, to build, download, and run the application scan which exists in *apps**snip**scan*, create the following make target:
   `snip.scan-CYW954907AEVAL1F download run`

   This project will periodically scan for Wi-Fi access points and will list them using the serial to USB connection on the kit.

   Note: By default, the kit comes pre-programmed with the same *snip.scan* example.

   Do the following to create the make target, build, program, and test application scan:
3. Right-click **43xxx_Wi-Fi** in the Make Target window as shown in Figure 3-4, and click **New**.

Figure 3-4. Creating New Make Target



4. Enter `snip.scan-CYW954907AEVAL1F download run` in the **Target name** field and click **OK**.

   Note: The list of all commands that can be provided in the Make target is listed in <*WICED-SDK installation directory>/ 43xxx_Wi-Fi/Makefile*.

   `snip.scan-CYW954907AEVAL1F download run` indicates the following:

   `snip` = directory inside **apps** folder

   `scan` = Sub-directory and name of the application to be built. For example, to build the console application under *test* directory in *apps*, then use `test.console` instead of `snip.scan`.

   `CYW954907AEVAL1F` = Board/platform name

   `download` = Indicates download to target

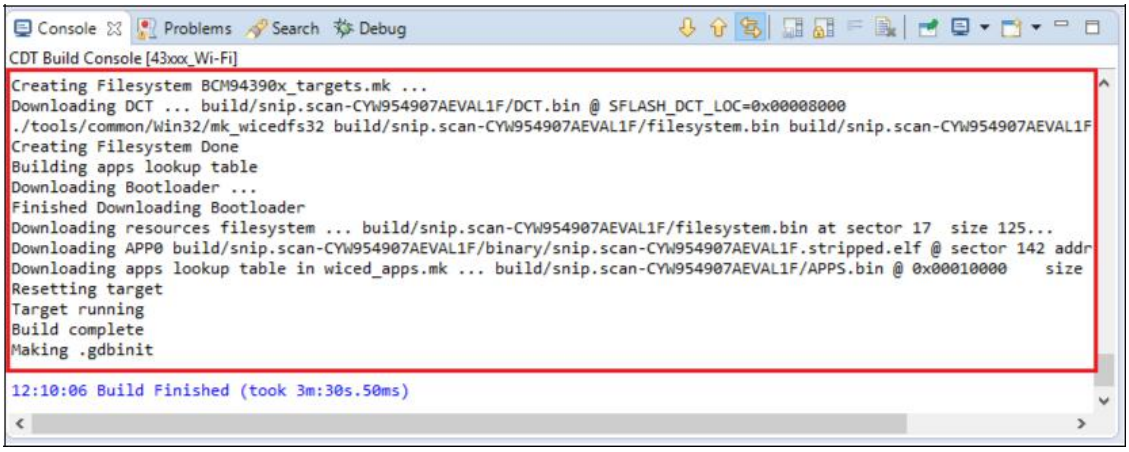
   `run` = Resets the target and starts execution

5. Double-click (alternately, right-click and select **Build Target**) the Clean Make Target to remove any output from the previous build. You should do a Make clean when any new files are added or removed to the corresponding target.

   Note: Ensure that you have connected CYW954907AEVAL1F EVK to the same PC via USB prior to executing the build target.

6. Double-click (alternatively right-click and select **Build Target**) the `snip.scan-CYW954907AEVAL1F download run` make target to build and download it to the CYW954907AEVAL1F EVK.
   The project is built and programmed into the CYW954907AEVAL1F EVK, as shown in Figure 3-5.

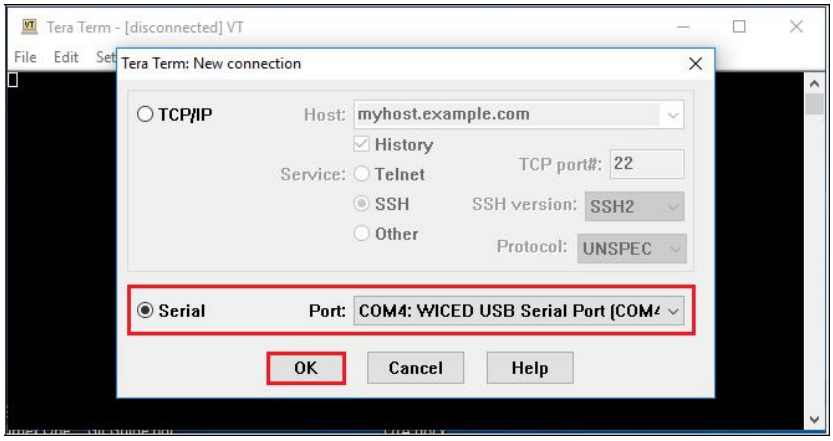
Figure 3-5. Successful Build and Program



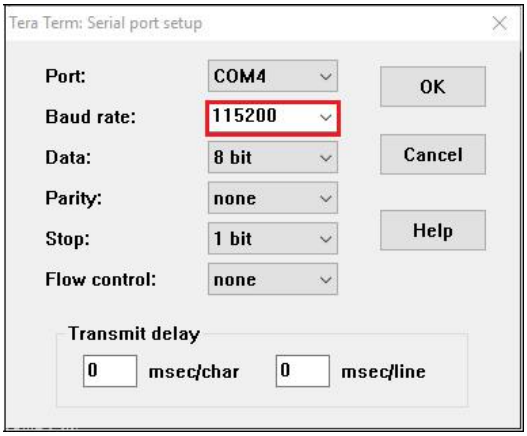
7. To view output messages with a terminal emulation program (such as Tera Term), follow these steps:

   a. Start the terminal emulation program.

b. You will see the following window. Click on **Serial** and select corresponding COM Port for your WICED device. Then click on **OK**



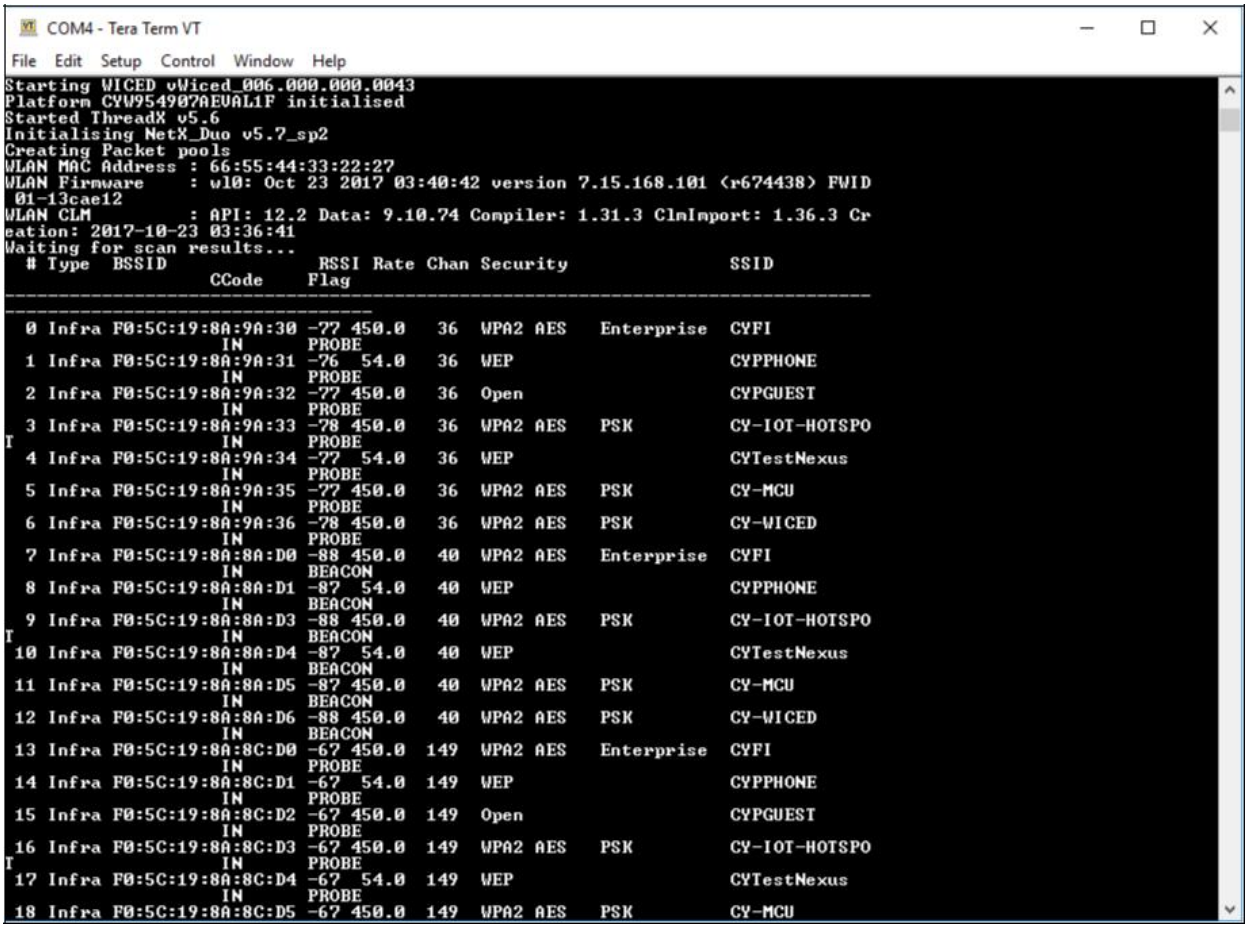
c. In the Terminal Emulator, go to **Setup > Serial port**…. Select the correct COM port and baud rate as follows.
Note: Exact Port number will vary with the corresponding PC port



d. Press the Reset button (see Figure 1-1) on the CYW954907AEVAL1F EVK to view the application startup messages.

8. The output of the Terminal Emulation program should be similar to what is shown in Figure 3-6.

Figure 3-6. Console Output



3.4.2 Troubleshooting

If a “download_dct” error message is displayed despite connecting the board, follow the steps outlined in this post.

3.4.3 Debugging a Project Using Breakpoints

After programming a project, it is possible to debug it in CYW954907AEVAL1F EVK using the built-in debugger.

Note that the scan example used in 3.4.1 Building and Programming a Project in WICED Studio IDE section is also used here. Steps outlined there should be first followed with a slight change (adding -debug to the Make Target command and removing run).

Instead of

```
snip.scan-CYW954907AEVAL1F download run
```

Use the following make command:

```
snip.scan-CYW954907AEVAL1F-debug download
```

If `-debug` is not added, it will be built for release. The important difference between the debug and release configurations is the optimization. Debug is built with no optimization and release is built with optimization. It is possible to debug without using `debug` as well, but with many variables and lines optimized away, many breakpoints might not get hit.

Note that breakpoints must be placed after starting a debug session in WICED Studio 5.0 or later. If there are any breakpoints that were created prior to the start of debug session, their properties must be changed to be enabled for all threads.

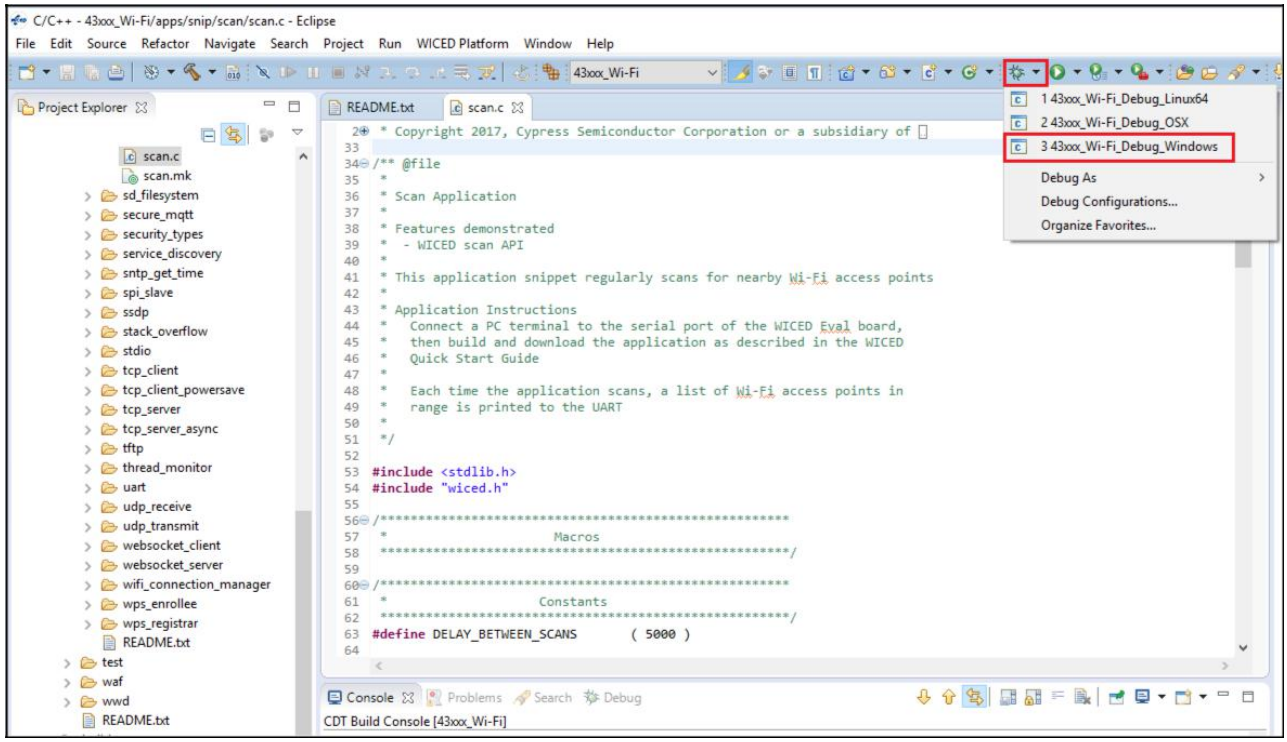
Do the following to debug the project:

1. Execute the make target described above to download the project to the device.
2. Click the arrow next to the **Debug** icon as shown in Figure 3-7 and select *43xxx-Wi-Fi_Debug_Windows*. On the **Confirm Perspective Switch** dialog, click **Yes**. The debug session starts and halts in the *start_GCC.s* file.

Note:

- The Confirm Perspective Switch dialog is not displayed if you have previously selected the **Remember my decision checkbox** in the Confirm Perspective Switch dialog.
- If any MakeFile/Build error occurs, then clean (using the *Clean* make target), rebuild, and download to the CYW954907AEVAL1F EVK again. The Debug session starts and halts in the *start_GCC.s* file.
- In the Debug Perspective, the Project explorer window goes away by default. To view the source files, switch back to the "C/C++" perspective.
- To switch between perspectives, use the "C/C++" or "Debug" icon at the top right corner of screen.

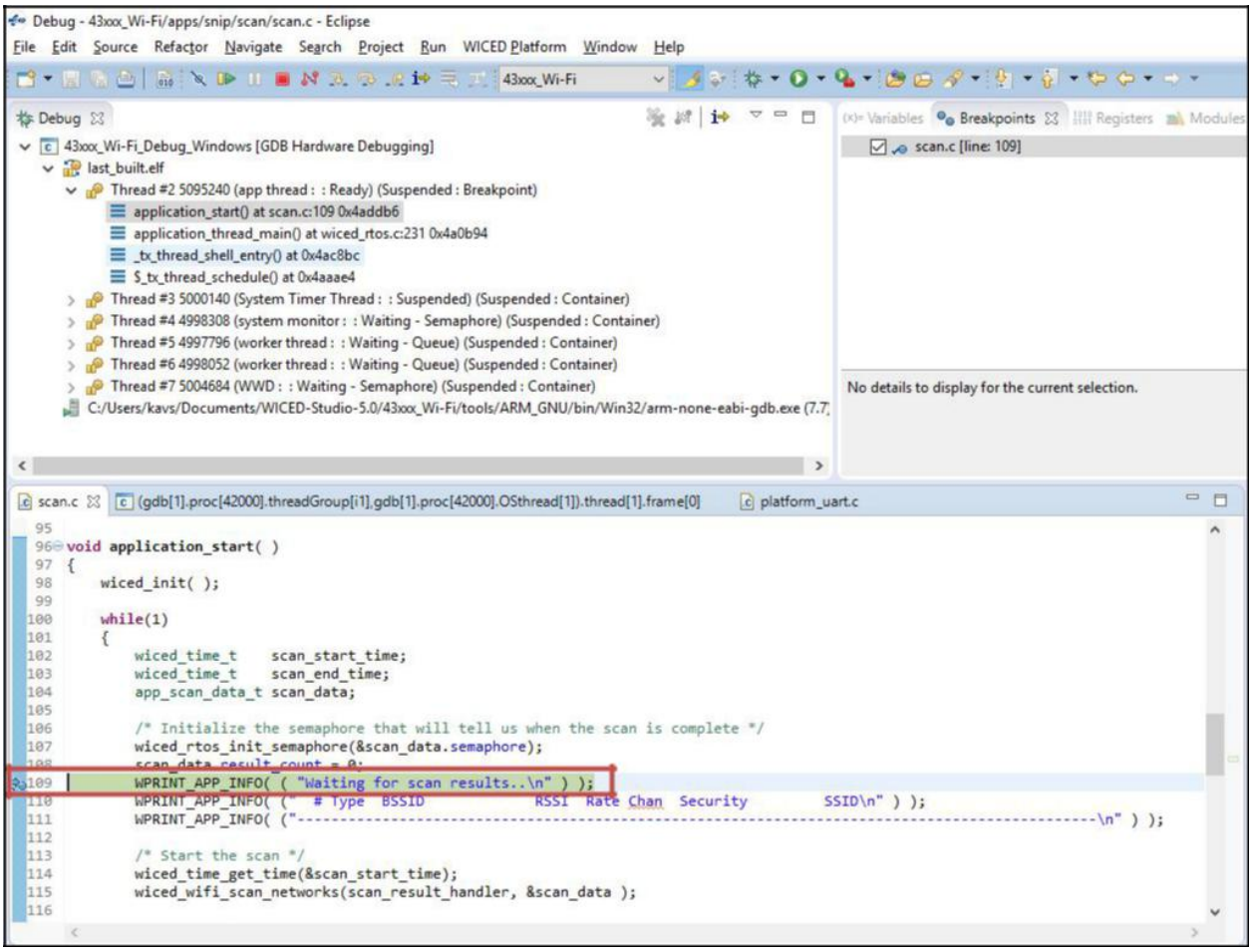
Figure 3-7. Debugging Project



3. Open the *scan.c* file from the Project Explorer window. Click on the line with `WPRINT_APP_INFO( ( "Waiting for scan results...\n" ) );` and press **Ctrl+Shift+B**. A blue hollow circle along with a check mark appears next to the line number as shown in Figure 3-8.
4. From the main menu, click **Run > Resume**. Execution will stop at the breakpoint that you added. To continue after hitting the breakpoint, click **Resume** again.
5. To disable the breakpoint, press **Ctrl+Shift+B** again on the same line, or deselect the corresponding checkbox in the Breakpoints window.

   Note: If the Breakpoint window does not appear, choose **Window > Show View > Breakpoints**.
6. To terminate the debugging session, click **Run > Terminate**, or click on the red square icon. Once you terminate the session, click on "**C/C++**" in the upper-right corner to return to the **C/C++** perspective.

Figure 3-8. Placing Breakpoint in Code



7. If Breakpoints are created prior to starting the current debug session, they will not be associated with the current thread and will be indicated with a blue circle without a check mark. To enable breakpoints in the current thread, associate the properties from the Breakpoints window with the current thread.

   Note: If you do not see any breakpoints in the Breakpoints window, click the **Show Breakpoints Supported by Selected Target** icon as shown in Figure 3-9. The breakpoints are displayed.

8. Right-click the desired breakpoint checkbox and click **Breakpoint Properties**…. Click the *last_built.elf* check box, as shown in Figure 3-10. The check mark appears before the actual breakpoint indicating its association with the current execution.

Figure 3-9. Show Breakpoints Icon

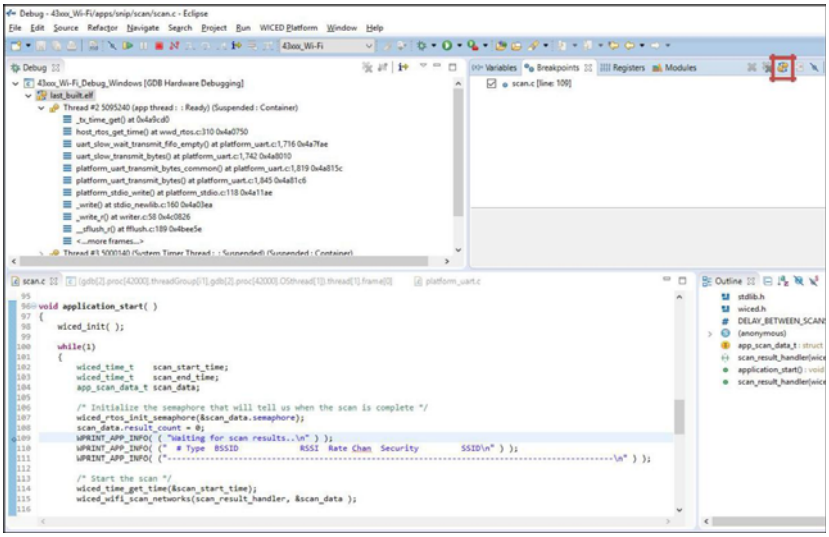
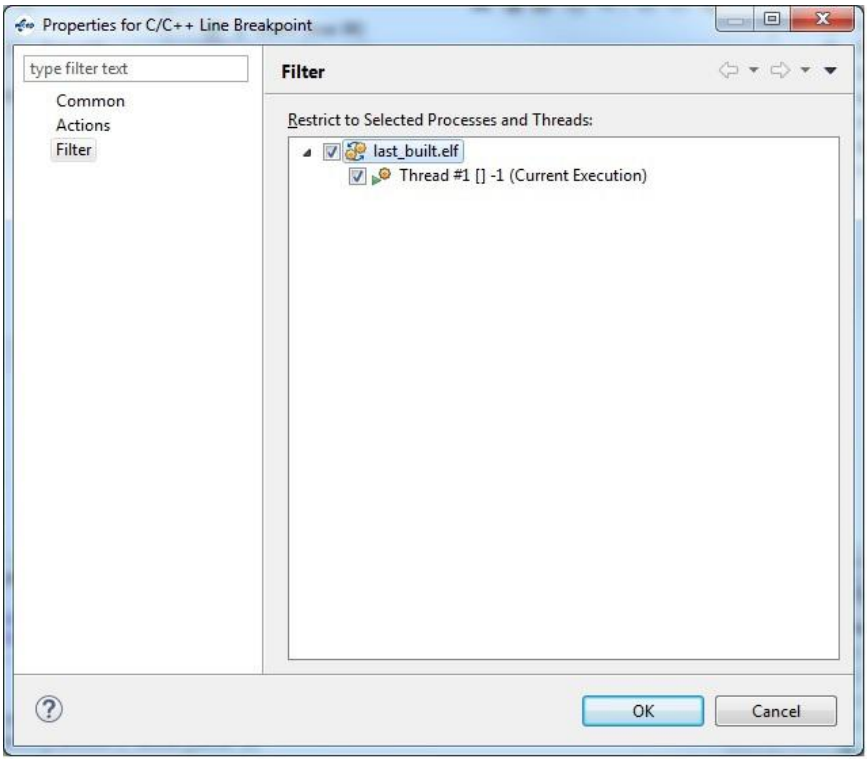


Figure 3-10. Enabling Breakpoint for Current Execution



4. Hardware

This chapter describes the CYW954907AEVAL1F EVK hardware and its different blocks, such as Bootstrap, reset control, Arduino-compatible headers, and module connectors.

The schematic is available at the following location after installing the software from Software Installation:
`<WICED_SDK_Directory>\43xx_Wifi\platforms\CYW954907AEVAL1F\schematics`

4.1 Bootstrap and Control Pins

Bootstrap options available in the CYW954907AEVAL1F EVK are shown in Table 4-1. The pins are sampled at power-on reset (POR) to determine various operating modes. Sampling occurs a few milliseconds after an internal POR or deassertion of the external POR. After the POR, each pin assumes the GPIO or alternative function specified in the CYW54907 Alternate GPIO function table in the CYW54907 datasheet (002-19312).

You must ensure that SPI mode and SDIO Host are not turned on at the same time because they share the same set of lines. For more information regarding bootstrap options, see the CYW54907 datasheet (002-19312).

Bootstrap options other than GPIO_7 and GPIO_13 are not available to modify in this board.

To change bootstrap options for GPIO_7 and GPIO_13, see the "Bootstraps, Flash" page of the schematics.

Table 4-1. Bootstrap Options Available in CYW954907AEVAL1F EVK

Pin	Strp Function	Strap Pull	
		Chip default	Board Default
GPIO_1	gSPI Mode 0 = Enable gSPI Mode 1 = Disable gSPI Mode	0	0
GPIO_7	WCPU Boot Mode: 0 = TCROM Boot 1 = TCMSRAM Boot	0	1 R135=10K to WLAN_VDDIO
GPIO_11	ACPU Boot Mode: 0 = SOCROM Boot 1 = SOCSRAM Boot	0	0

Table 4-1. Bootstrap Options Available in CYW954907AEVAL1F EVK *(continued)*

Pin	Strp Function	Strap Pull	
		Chip default	Board Default
GPIO_13	SDIO Mode: 0 = SDIO Device 1 = SDIO Host	0	1 R141=10K to WLAN_VDDIO
RF_SW_CTRL_5	Host DAP Clock Sel 1 = Enable XTAL clock for DAP sub system 0 = Disable Use Test clock TCK for DAP sub system	0	0
RF_SW_CTRL_7	PMU resource initialization mode selection 1 = Mode 1 0 = Mode 2	0	0
RF_SW_CTRL_9	LPO(Low Power oscillator) Selection: 0 = LPO from HIB (Hibernation Block) 1 = Internal 32KHz LPO	0	0

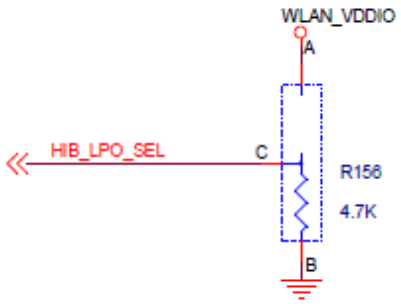
Note: There is no bootstrapping provision for GPIO_1, GPIO_11, GPIO_15, RF_SW_CTRL_5, RF_SW_CTRL_7, and RF_SW_CTRL_9 on the EVK board.

1. HIB_LPO_SEL and RF_SW_CTRL_9_HIB_LPO_SEL_UART2_TX

HIB_LPO_SEL

By default, this is automatically selected between the external crystal and internal oscillator. If only an internal oscillator is used, strapping of HIB_LPO_SEL is irrelevant.

Figure 4-1. HIB_LPO_SEL strapping Option



RF_SW_CTRL_9 can also be used for LPO Selection. However, it should be pulled LOW to use the recommended external 32.768-kHz HIB crystal.

0 = LPO from HIB

1 = Internal 32-kHz LPO

Note:

You should use the external 32.768-kHz crystal for a more accurate clock. Accuracy of the clock is critical when using power save mode. If an internal LPO is used, the board may become unresponsive at higher temperatures (>80°C).

2. HIB_REG_ON_IN

Used by the Hibernation (HIB) block to power up internal CYW54907 regulators. If the HIB_REG_ON_IN pin is LOW, regulators are disabled. For the HIB_REG_ON_IN pin to work as designed, HIB_REG_ON_OUT must be connected to REG_ON.

The CYW54907/BCM54907 datasheet states that HIB_REG_ON_IN needs to be delayed by at least two cycles of the 32.768-kHz clock after VBAT and VDDIO have reached 90% of their final values. To ensure a proper bootup, the RC delay circuit for HIB_REG_ON_IN is essential as shown in the following figure:

Figure 4-2. HIB_REG_ON_IN Delay Circuit

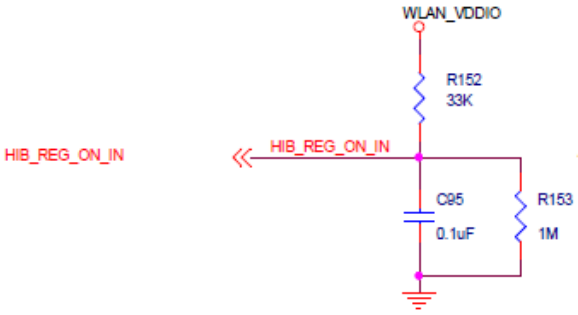


Figure 4-3. An Excerpt from CYW54907 Datasheet (Section 18)

Notes:

- The CYW54907 has an internal POR circuit. The device will be held in reset for a maximum of 110 ms after VDDC and VDDIO have both passed the POR threshold.
- The 10%–90% V_{BAT} rise time should not be faster than 40 microseconds. V_{BAT} should be up before or at the same time as VDDIO. VDDIO should not be present first or be held high before V_{BAT} is high.

18.1.2 Control Signal Timing Diagrams

Figure 37. REG_ON = High, No HIB_REG_ON_OUT Connection to REG_ON

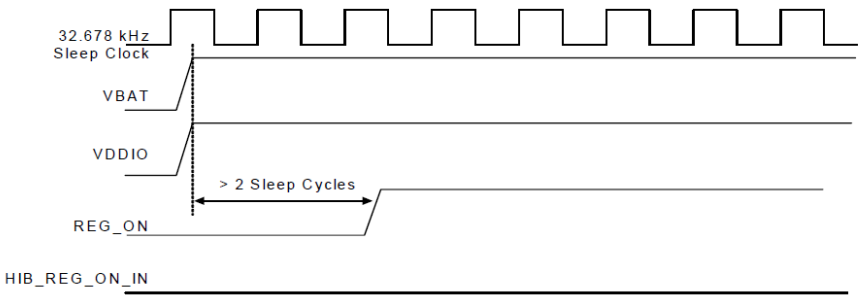
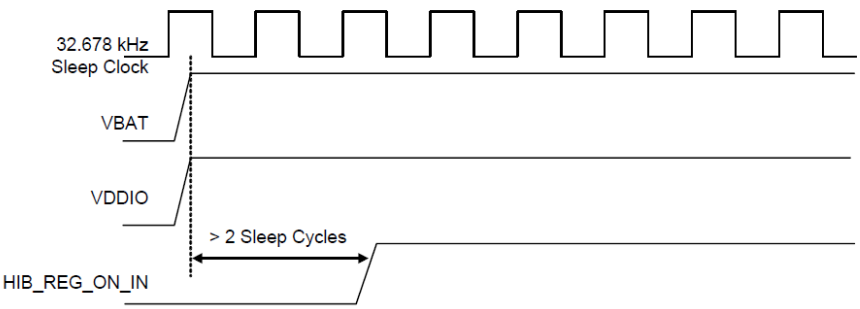


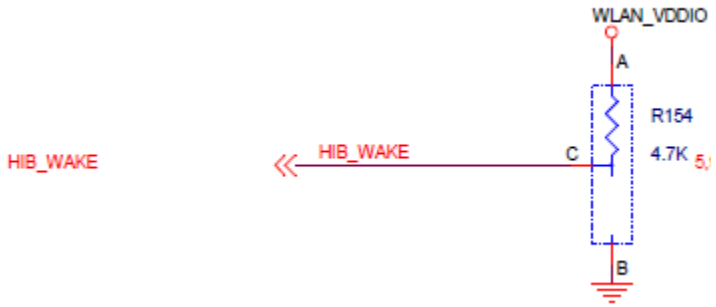
Figure 38. HIB_REG_ON_IN = High, HIB_REG_ON_OUT Connected to REG_ON



3. HIB_WAKE

Used to wakeup chip from Hibernation mode. This pin should be pulled HIGH.

Figure 4-4. HIB_WAKE Strapping Option



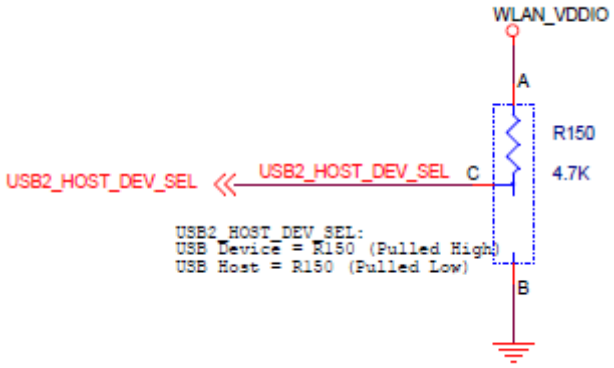
4. USB2_HOST_DEV_SEL

Used to select the USB mode; it is set in USB DEVICE mode by default.

0 = USB HOST mode

1 = USB DEVICE mode

Figure 4-5. USB2_HOST_DEV_SEL Strapping Option



5. JTAG_SEL and GPIO_8_TAP_SEL States for Test and Debug Function Selection

JTAG_SEL

Set JTAG_SEL:

0 = JTAG interface disabled

1 = JTAG debug mode enabled

GPIO_8_TAP_SEL

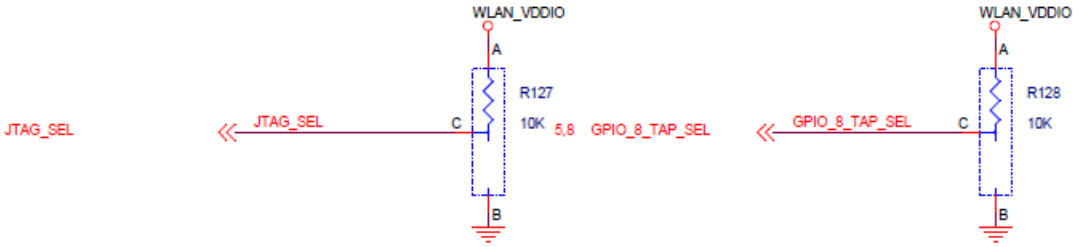
Set GPIO_8 (TAP_SEL):

0 = WLAN JTAG

1 = APPS JTAG

Note: Default setup is JTAG_SEL = HIGH / TAP_SEL = HIGH.

Figure 4-6. JTAG_SEL and GPIO_8_TAP_SEL Strapping Option



6. GPIO_1_GSPI_MODE

GPIO_1 is used for gSPI mode. By default, CYW54907 enables gSPI.

0 = gSPI engine enabled

1 = gSPI engine disabled

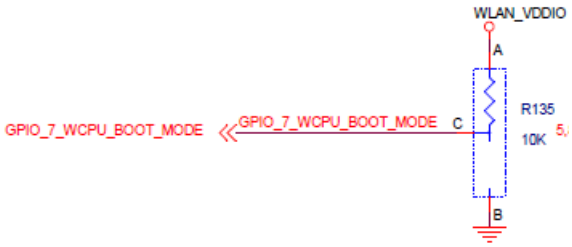
7. GPIO_7_WCPU_BOOT_MODE

GPIO_7 is used for WCPU Boot mode, and pulled HIGH for recommended TCMSRAM Boot mode.

0 = TCROM Boot

1 = TCMSRAM Boot

Figure 4-7. GPIO_7_WCPU_BOOT_MODE Strapping Option



8. GPIO_9_USB_SEL

GPIO_9 is used for USB selection. CYW54907 uses USB mode only; therefore, pull this pin HIGH.

0 = HSIC Sel

1 = USB PHY

9. GPIO_11_ACPU_Boot Mode

GPIO_11 is used for ACPU Boot mode; by default, CYW54907 sets it to the recommended SOCROM Boot.

0 = SOCROM Boot

1 = SOCSRAM Boot

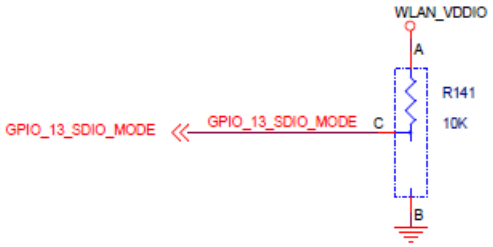
10. GPIO_13_SDIO_MODE

GPIO_13 is used to select the SDIO mode; it is set in SDIO HOST mode by default.

0 = SDIO Device

1 = SDIO Host

Figure 4-8. GPIO_13_SDIO_MODE Strapping Option



11. RF_SW_CTRL_7_RSRC_INIT_MODE_UART1_TX_OUT

This pin should be pulled HIGH.

Highly Recommended to pull up RF_SW_CTRL_7 via a 10K resistor to WLAN_VDDIO during bootup. Do not leave it floating. If left floating, the first-time programming of the SFlash fails, while second-time programming passes.

12. CYW54907 in Deep Sleep Mode

To wake up CYW54907 from deep sleep mode, any GPIOs from GPIO_0 to GPIO_15 (except GPIO_13) can be used. The selected GPIO should not be pin-muxed; instead it should be a dedicated one for deep sleep wakeup of CYW54907.

13. CYW54907 Power Signals for Unused Interfaces

Connect the following power signals to GND when the associated interface is not used.

VDDIO_SD

VDDIO_AUDIO

USB_VDD_3V3

VDDIO_RMII

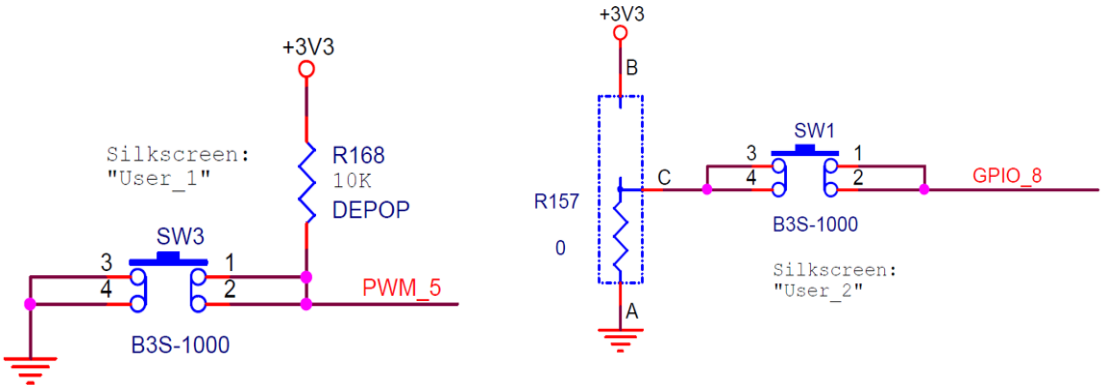
4.2 User Switches

There are two user switches available on the board named USER_1 and USER_2. Table 4-2 shows the pin names and enumeration used in WICED for the switches.

Table 4-2. User Switch available on the board

Switch	CYW54907 Pin Name	WICED_ENUM_ID	Alternate Enumeration in WICED
USER_1 (SW3)	PWM_5	WICED_GPIO_18	WICED_BUTTON1
USER_2 (SW1)	GPIO_8	WICED_GPIO_4	WICED_BUTTON2

Figure 4-9. User Switch Circuit Diagram



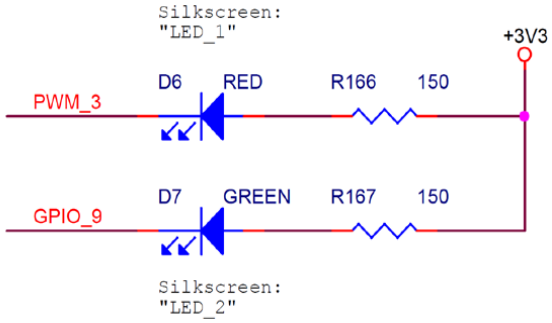
4.3 LED

There are two user LEDs available named LED_1 and LED_2. Table 4-3 shows the pin name and enumeration used in WICED for these LEDs.

Table 4-3. User LED Available on the Board

Switch	CYW54907 Pin Name	WICED_ENUM_ID	Alternate Enumeration in WICED
LED_1	PWM_3	WICED_GPIO_16	WICED_LED1
LED_2	GPIO_9	WICED_GPIO_5	WICED_LED2

Figure 4-10. User LED Circuit Diagram



4.4 Reset Control

CYW54907 device can be reset using the "Target Reset" switch **SW2** or a reset command from the on-board programmer/debugger and serial interface chip, as shown in Figure 4-11. The CYW54907/BCM54907 datasheet states that HIB_REG_ON_IN needs to be delayed by at least 2 cycles of the 32.768-kHz clock after VBAT and VDDIO have reached 90% of their final values. To ensure proper bootup, the RC delay circuit for HIB_REG_ON_IN is essential as shown in Figure 4-12. See Bootstrap and Control Pins on page 28 for details on RC Delay Circuit.

Figure 4-11. Reset Circuit Diagram

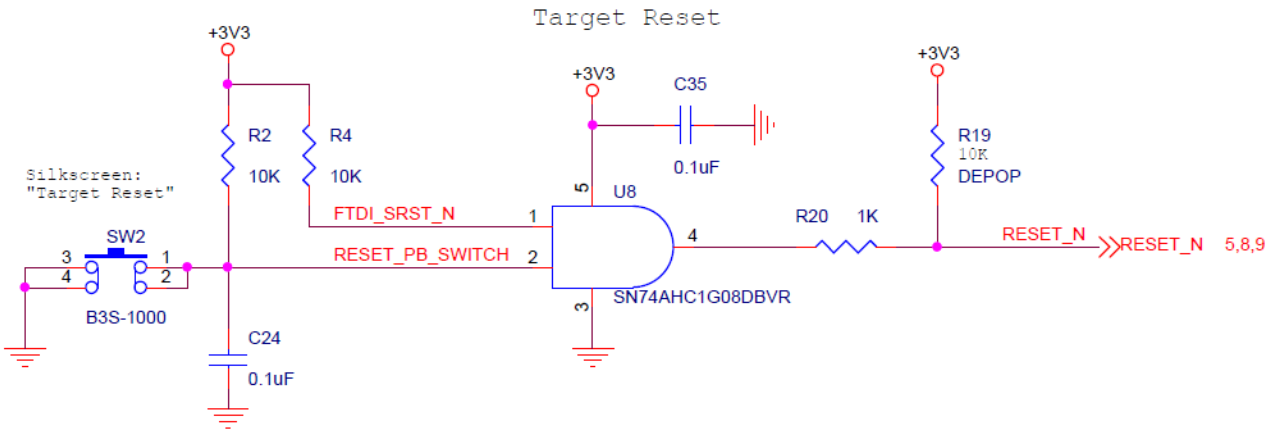
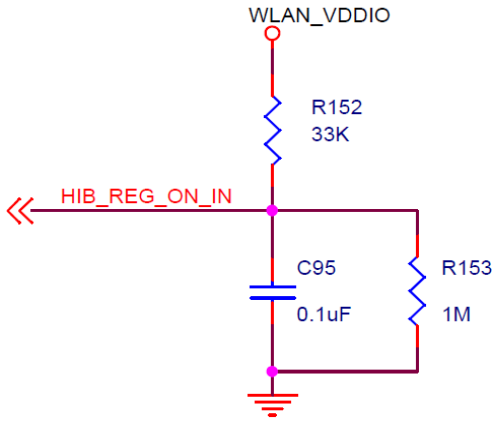


Figure 4-12. HIB_REG_ON_IN RC Delay Circuit



4.5 Ethernet

The Ethernet MAC Controller in the CYW54907 interfaces to an external PHY chip BCM5241 using the Media Independent Interface (MII) as shown in Figure 4-13. The same signals are also listed in Table 4-4. CYW54907 also supports Reduced Media Independent Interface (RMII). The controller can transmit and receive data at 10 Mbps and 100 Mbps.

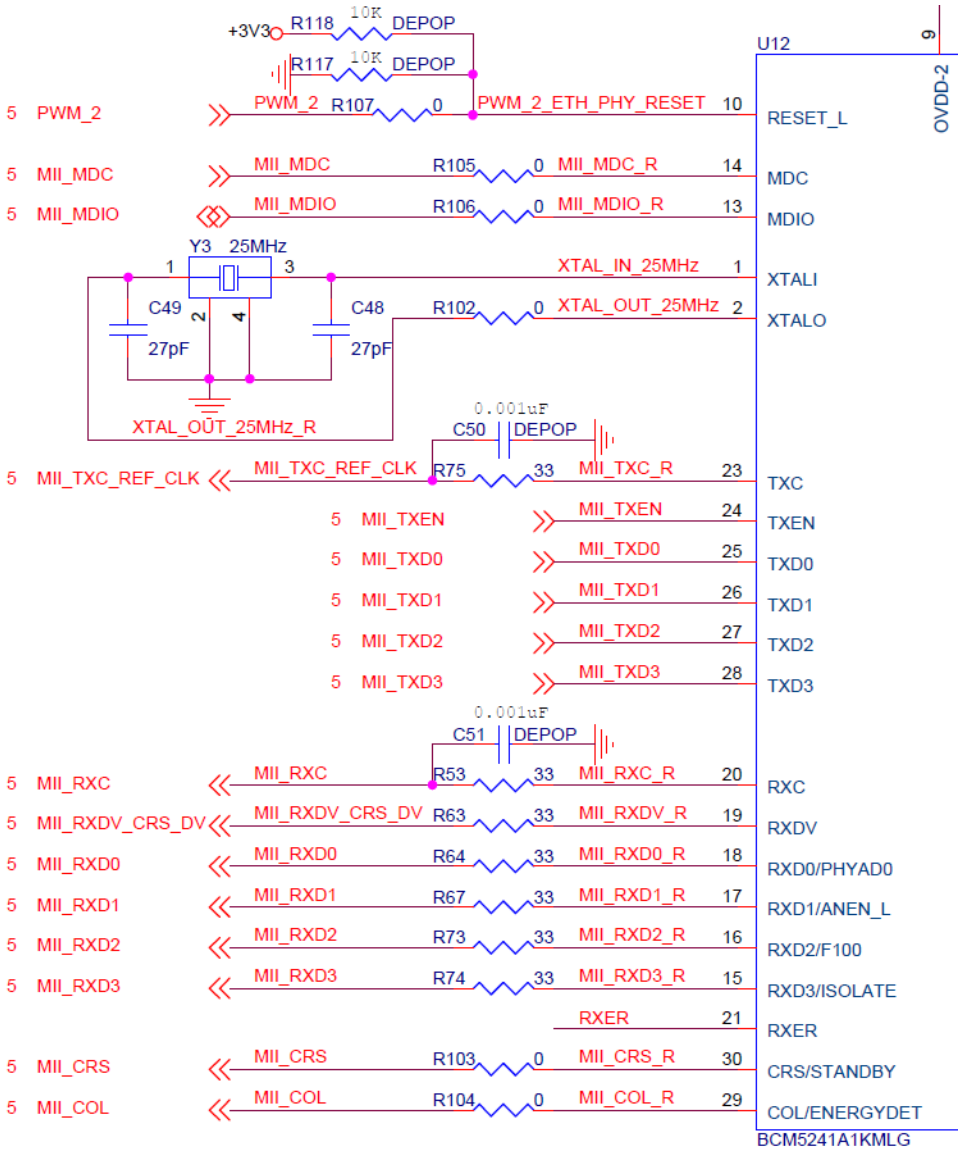
Table 4-4. CYW54907 EMAC to PHY Chip Connection

SL. NO	CYW54907 Pin Name	Net Name in Schematic	BCM5241 Pin Name
1	RMII_G_RXC	MII_RXC	RXC
2	RMII_G_COL	MII_COL	COL/ENERGYDET
3	RMII_G_CRS	MII_CRS	CRS/STANDBY
4	RMII_G_TXC	MII_TXC_RMII_REF_CLK	TXC
5	RMII_G_TXD0	MII_TXD0	TXD0
6	RMII_G_TXD1	MII_TXD1	TXD1
7	RMII_G_TXD2	MII_TXD2	TXD2
8	RMII_G_TXD3	MII_TXD3	TXD3
9	RMII_G_RXD0	MII_RXD0	RXD0/PHYAD0

Table 4-4. CYW54907 EMAC to PHY Chip Connection *(continued)*

SL. NO	CYW54907 Pin Name	Net Name in Schematic	BCM5241 Pin Name
10	RMII_G_RXD1	MII_RXD1	RXD1/ANEN_L
11	RMII_G_RXD2	MII_RXD2	RXD2/F100
12	RMII_G_RXD3	MII_RXD3	RXD3/ISOLATE
13	RMII_MDIO	MII_MDIO	MDIO
14	RMII_MDC	MII_MDC	MDC
15	RMII_G_TXEN	MII_TXEN	TXEN
16	RMII_G_RXDV	MII_RXDV_CRS_DV	RXDV
17	PWM_2	PWM_2	RESET_L

Figure 4-13. Ethernet MAC Controller to External PHY Connection



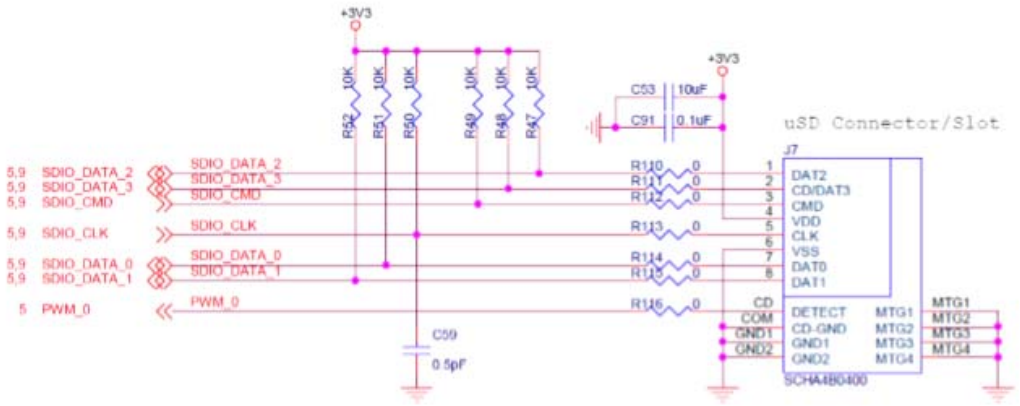
4.6 Micro SD Connector/Slot

Micro SD connector is connected to the SDIO Interface of CYW54907. CYW54907 supports both SDIO 3.0 Host and device modes. Figure 4-14 shows the interface between Micro SD connector and CYW54907. These signals are listed in Table 4-5.

Table 4-5. Micro SD Connector signals

SL.No	CYW54907 Based SIP Pin Name	Micro SD Connector/Slot Name
1	SDIO_DATA_0	DAT0
2	SDIO_DATA_1	DAT1
3	SDIO_DATA_2	DAT2
4	SDIO_DATA_3	CD/DAT3
5	SDIO_CMD	CMD
6	SDIO_CLK	CLK
7	PWM_0	DETECT

Figure 4-14. Micro SD Connector Circuit Diagram



4.7 JTAG Connector

4.7.1 On-board Programmer/Debugger and Serial Interface Chip

The on-board programmer/debugger chip uses JTAG to program/debug CYW54907-based SiP module.

Table 4-6 shows the connection between CYW54907 and On-board Programmer/Debugger chip. In addition to the connections listed in the table, JTAG_SEL and GPIO_8_TAP_SEL lines have been pulled high to make sure programming/debugging is enabled through JTAG in CYW54907.

Table 4-6. Connection between CYW54907 and On-board Programmer/Debugger

SL. No	CYW54907 Based SIP Pin Name	On-board Programmer/Debugger Connection
1	GPIO_2_JTAG_TCK	FTDI_JTAG_TCK
2	GPIO_3_JTAG_TMS	FTDI_JTAG_TMS
3	GPIO_4_JTAG_TDI	FTDI_JTAG_TDI
4	GPIO_5_JTAG_TDO	FTDI_JTAG_TDO
5	GPIO_6_JTAG_TRST_L	FTDI_JTAG_TRST

4.7.2 External JTAG

To use the External JTAG connector (J3), set all positions in switch **SW4** to closed and connect your external JTAG debugger. Ensure that the drivers for the debugger hardware are installed in the same PC where WICED Studio is installed. When using Olimex connectors, for example Olimex_ARM-USB-TINY-H, add "JTAG=Olimex_ARM-USB-TINY- H" in your make target to debug. Figure 4-15 shows the relevant part of the schematic for connecting an External JTAG device. Figure 4-16 shows the connection between Olimex and the CYW954907AEVAL1F EVK.

Figure 4-15. External JTAG Connector Circuit Diagram

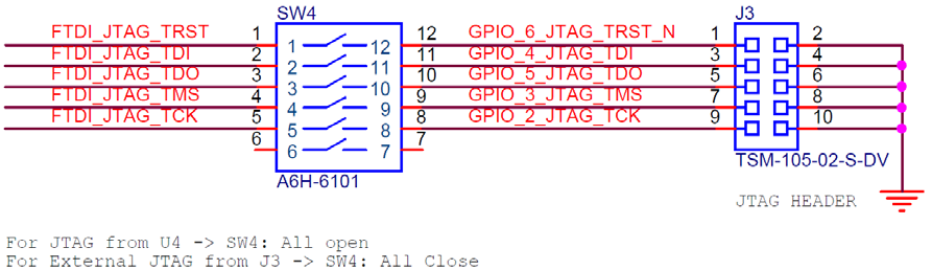


Figure 4-16. JTAG to Olimex Connection

4.8 Connectors

4.8.1 WICED Header

J6 is the WICED header available on CYW954907AEVAL1F EVK. This is a 44-pin header containing I2C, SDIO, UART, SPI, PWM lines, and I/Os. Note that some signals are shared with Arduino header (UART0 Tx/Rx) and On-board Programmer/debugger chip (UART1). Table 4-7 illustrates the J6 pinout.

Table 4-7. WICED Header Pinout

Eval Board Header	CYW54907 Pin Name	WICED Enumeration	Alternate Enumeration
J6.1	PWM_4	WICED_GPIO_17	WICED_PWM_5
J6.2	PWM_5	WICED_GPIO_18	WICED_BUTTON1
J6.3	I2S0_MCK	WICED_GPIO_28	WICED_I2S_1
J6.4	I2S0_SD_OUT	WICED_GPIO_32	WICED_I2S_1
J6.5	I2S0_SCK_BCLK	WICED_GPIO_29	WICED_I2S_1
J6.6	I2S0_WS_LRCLK	WICED_GPIO_30	WICED_I2S_1
J6.7	PWM_3	WICED_GPIO_16	WICED_LED1
J6.8	GND	N/A	N/A
J6.9	SPI_1_CLK	WICED_GPIO_38	WICED_SPI_2
J6.10	I2S1_SD_OUT	WICED_GPIO_37	WICED_I2S_3
J6.11	SPI_1_MISO	WICED_GPIO_39	WICED_SPI_2
J6.12	SPI_0_CLK	WICED_GPIO_20	WICED_SPI_1
J6.13	SPI_1_MOSI	WICED_GPIO_40	WICED_SPI_2
J6.14	SPI_0_MOSI	WICED_GPIO_21	WICED_SPI_1
J6.15	SPI_1_CS	WICED_GPIO_41	WICED_SPI_2
J6.16	SPI_0_CS	WICED_GPIO_22	WICED_SPI_1
J6.17	SPI_0_MISO	WICED_GPIO_19	WICED_SPI_1
J6.18	UART0_RXD_IN	WICED_PERIPHERAL_PIN_3	WICED_UART_2
J6.19	GND	N/A	N/A
J6.20	UART0_TXD_OUT	WICED_PERIPHERAL_PIN_4	WICED_UART_2
J6.21	USB2_HOST_DEV_SEL	N/A	N/A
J6.22	UART0_CTS_IN	WICED_PERIPHERAL_PIN_5	WICED_UART_2
J6.23	I2C_0_SCL	WICED_GPIO_49	WICED_I2C_1
J6.24	UART0_RTS_OUT	WICED_PERIPHERAL_PIN_6	WICED_UART_2
J6.25	I2C_0_SDA	WICED_GPIO_48	WICED_I2C_1
J6.26	I2S1_MCK	WICED_GPIO_33	WICED_I2S_3
J6.27	I2S1_WS_LRCLK	WICED_GPIO_35	WICED_I2S_3
J6.28	GND	N/A	N/A
J6.29	I2S1_SCK_BCLK	WICED_GPIO_34	WICED_I2S_3
J6.30	SDIO_DATA_1	WICED_GPIO_45	N/A
J6.31	SDIO_DATA_0	WICED_GPIO_44	N/A
J6.32	SDIO_CLK	WICED_GPIO_42	N/A

Table 4-7. WICED Header Pinout *(continued)*

Eval Board Header	CYW54907 Pin Name	WICED Enumeration	Alternate Enumeration
J6.33	SDIO_CMD	WICED_GPIO_43	N/A
J6.34	SDIO_DATA_3	WICED_GPIO_47	N/A
J6.35	SDIO_DATA_2	WICED_GPIO_46	N/A
J6.36	RF_SW_CTRL_6_UART1_RXD	WICED_PERIPHERAL_PIN_1	WICED_UART_1
J6.37	UART1_TXD	WICED_PERIPHERAL_PIN_2	WICED_UART_1
J6.38	RF_SW_CTRL_8_UART2_RXD	WICED_PERIPHERAL_PIN_7	WICED_UART_3
J6.39	UART2_TXD	WICED_PERIPHERAL_PIN_8	WICED_UART_3
J6.40	HIB_WAKE	N/A	N/A
J6.41	HIB_LPO_SEL	N/A	N/A
J6.42	HIB_REG_ON_IN	N/A	N/A
J6.43	USB2_DN	N/A	N/A
J6.44	USB2_DP	N/A	N/A

4.8.2 Arduino-Compatible Headers

J9, J13, J12 and J10 are the Arduino headers available in the CYW954907AEVAL1F EVK. Table 4-8 shows the pinout of the Arduino Header. Note the following while connecting an Arduino shield to the board:

- 5V pin of Header (J9) is not connected to the board.
- The maximum current that an Arduino shield can sink from the board depends on the application that is running. In general, 100 mA is the worst-case scenario.
- The Arduino Analog reference is connected to the 3V3 (3.3V) power supply through R21, which is not populated by default. In other words, the analog reference is not driven by default.
- An external ADC attached to CYW54907 helps to achieve analog functionality on the Arduino headers.

Table 4-8. Arduino Header Pinout

Eval Board Header	CYW54907 Pin Name/ Kit Signal Name	ARDUINO Header Name	WICED Enumeration	Alternate Enumeration
J10.1	GPIO_0		D0	N/A
J10.2	GPIO_1		D1	N/A
J10.3	GPIO_13		D2	N/A
J10.4	GPIO_7		D3	WICED_PWM_6
J10.5	GPIO_14		D4	N/A
J10.6	GPIO_16		D5	WICED_PWM_3
J10.7	GPIO_15		D6	WICED_PWM_4
J10.8	I2S0_SD_IN		D7	WICED_I2S_1
J12.1	I2S1_SD_IN		D8	WICED_I2S_3
J12.2	PWM_4		D9	WICED_PWM_5
J12.3	GPIO_11		D10	WICED_PWM_2
J12.4	GPIO_10		D11	WICED_PWM_1
J12.5	GPIO_12		D12	N/A

Table 4-8. Arduino Header Pinout *(continued)*

Eval Board Header	CYW54907 Pin Name/ Kit Signal Name	ARDUINO Header Name	WICED Enumeration	Alternate Enumeration
J12.6	GPIO_9		D13	WICED_LED2
J12.7	GND		GND	N/A
J12.8	ARD_AREF		AREF	N/A
J12.9	I2C_1_SDA		SDA	WICED_I2C_2
J12.10	I2C_1_SCL		SCL	WICED_I2C_2
J13.1	ARD_AD0		A0	N/A
J13.2	ARD_AD1		A1	N/A
J13.3	ARD_AD2		A2	N/A
J13.4	ARD_AD3		A3	N/A
J13.5	ARD_AD4_SDA		A4	N/A
J13.6	ARD_AD5_SCL		A5	N/A
J9.1	NC		NC	N/A
J9.2	ARD_IOREF		IOREF	N/A
J9.3	ARD_RESET		RESET	N/A
J9.4	3V3		3.3V	N/A
J9.5	NC		5V	N/A
J9.6	GND		GND	N/A
J9.7	GND		GND	N/A
J9.8	VIN_EXT		VIN	N/A

4.9 UART Port Configuration on CYW954907AEVAL1F Kit

CYW54907 has three UART ports: slow UART, fast UART, and GCI UART. Slow UART and GCI UART are 2-wire interfaces, while fast UART is a 4-wire interface that can support up to a 3-Mbps baud rate. Slow UART is routed to the on-board programmer/debugger chip for UART-to-USB communication. The UART peripherals are defined in *platforms/CYW954907AEVAL1F/platform.c*. Following table (also available in *platforms/CYW954907AEVAL1F/platform.h*) shows the UART pins available on the kit.

WICED Peripheral Enumeration ID	Pin Name on CYW54907	MURATA Module Pin Name	Header Pin Number	WICED Enumeration
WICED_PERIPHERAL_PIN_2	RF_SW_CTRL_7	RF_SW_CTRL_7_UART1_TXD	J6:37	WICED_UART_1
WICED_PERIPHERAL_PIN_3	UART0_RXD	UART0_RXD_IN	J6:18	WICED_UART_2
WICED_PERIPHERAL_PIN_4	UART0_TXD	UART0_TXD_OUT	J6:20	WICED_UART_2
WICED_PERIPHERAL_PIN_5	UART0_CTS	UART0_CTS_IN	J6:22	WICED_UART_2
WICED_PERIPHERAL_PIN_6	UART0_RTS	UART0_RTS_OUT	J6:24	WICED_UART_2
WICED_PERIPHERAL_PIN_7	RF_SW_CTRL_8	RF_SW_CTRL_8_UART2_RXD	J6:38	WICED_UART_3
WICED_PERIPHERAL_PIN_8	RF_SW_CTRL_9	RF_SW_CTRL_9_UART2_TXD	J6:39	WICED_UART_3

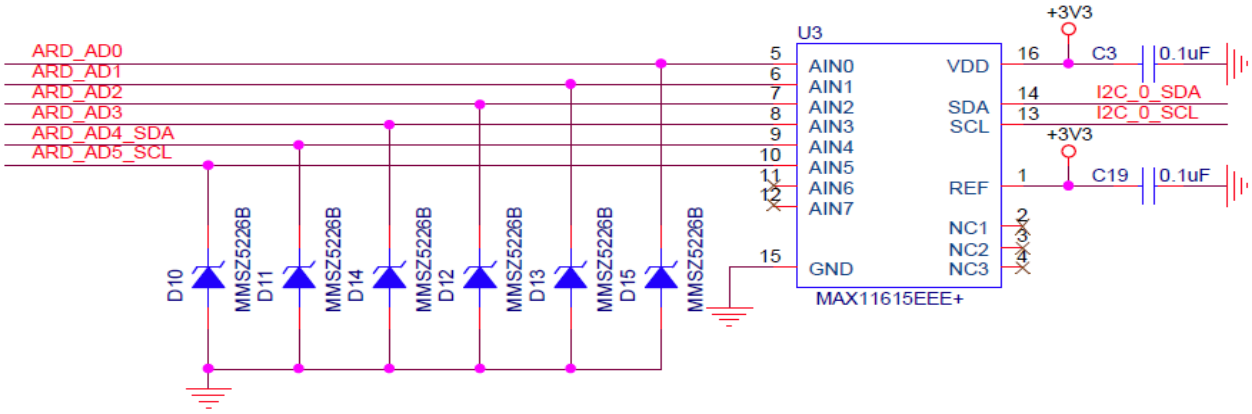
4.10 External ADC

CYW54907 does not have an in-built ADC block. Analog measurements from the Arduino header analog pins is achieved using an external ADC chip (MAX11615) connected to CYW54907 through an I2C interface (I2C_0 module-Slave Address 0x33). Table 4-9 lists the connections between CYW54907 and the external ADC Circuit diagram is shown in Figure 4-17.

Table 4-9. External ADC Connection

I2C Line	CYW54907 Pin Name	WICED Enumeration	Alternate Enumeration
SDA	I2C_0_SDA	WICED_GPIO_48	WICED_I2C_1
SCL	I2C_0_SCL	WICED_GPIO_49	WICED_I2C_1

Figure 4-17. External ADC Circuit Diagram



4.11 PWM

There are six dedicated PWM outputs available on CYW54907. These PWMs can be multiplexed onto different pins. You can find their definitions in *platforms/CYW954907AEVAL1F/platform.c* inside WICED Studio.

The PWMs can be reassigned to other pins by changing the first argument of the `platform_pwm_t platform_pwm_peripherals` structure in *platform.c*. Table 4-10 through Table 4-15 show the possible combinations and their Arduino header locations.

Table 4-10. WICED_PWM_1 Combinations

Pin MUX Selection	Header Pin	Header Name
PIN_GPIO_10 (DEFAULT)	J12.4	Arduino D11 (MOSI)
PIN_GPIO_0	J10.1	Arduino D0
PIN_GPIO_8	–	–
PIN_GPIO_12	J12.5	Arduino D12 (MISO)
PIN_GPIO_14	–	–
PIN_GPIO_16	J10.6	Arduino D5
PIN_PWM_0	–	–

Table 4-11. WICED_PWM_2 Combinations

Pin MUX Selection	Header Pin	Header Name
PIN_GPIO_11 (DEFAULT)	J12.3	Arduino D10
PIN_GPIO_1	J10.1	Arduino D0
PIN_GPIO_7	J10.4	Arduino D3
PIN_GPIO_9	J12.9	Arduino SCK
PIN_GPIO_13	J10.3	Arduino D2
PIN_GPIO_15	J10.7	Arduino D6
PIN_PWM_1	–	–

Table 4-12. WICED_PWM_3 Combinations

Pin MUX Selection	Header Pin	Header Name
PIN_GPIO_16 (DEFAULT)	J10.6	Arduino D5
PIN_GPIO_8	–	–
PIN_GPIO_0	J10.1	Arduino D0
PIN_GPIO_10	J12.4	Arduino D11 (MOSI)
PIN_GPIO_12	J12.5	Arduino D12 (MISO)
PIN_GPIO_14	–	–
PIN_PWM_2	–	–

Table 4-13. WICED_PWM_4 Combinations

Pin MUX Selection	Header Pin	Header Name
PIN_GPIO_15 (DEFAULT)	J10.7	Arduino D6
PIN_GPIO_1	J10.1	Arduino D0
PIN_GPIO_7	J10.4	Arduino D3
PIN_GPIO_9	J12.9	Arduino SCK
PIN_GPIO_11	J12.3	Arduino D10
PIN_GPIO_13	J10.3	Arduino D2
PIN_PWM_3	–	–

Table 4-14. WICED_PWM_5 Combinations

Pin MUX Selection	Header Pin	Header Name
PIN_PWM_4 (DEFAULT)	J6.1	Arduino A1
PIN_GPIO_0	J10.1	Arduino D0
PIN_GPIO_8	–	–
PIN_GPIO_10	J12.4	Arduino D11 (MOSI)
PIN_GPIO_12	J12.5	Arduino D12 (MISO)
PIN_GPIO_14	–	–
PIN_GPIO_16	J10.6	Arduino D5

Table 4-15. WICED_PWM_6 Combinations

Pin MUX Selection	Header Pin	Header Name
PIN_GPIO_7 (DEFAULT)	J10.4.4	Arduino D3
PIN_GPIO_1	J10.1	Arduino D0
PIN_GPIO_9	J12.9	Arduino SCK
PIN_GPIO_11	J12.3	Arduino D10
PIN_GPIO_13	J10.3	Arduino D2
PIN_GPIO_15	J10.7	Arduino D6
PIN_PWM_5	–	–

5. Code Examples

This chapter demonstrates the functionality of CYW54907 devices using the CYW954907AEVAL1F EVK code examples. Download and extract the zip file from the CYW954907AEVAL1F EVK web page as specified in the Software Installation section. The code examples once unzipped can be viewed in WICED Studio 6.0 (or later). In addition to the added examples, there are already many apps (snip.gpio, test.console, and so on) that are available in WICED Studio 6.0.

5.1 Using Code Examples

Code examples already added can be compiled after creating Make Targets. See Building and Programming a Project in WICED Studio IDE for the process of creating targets.

Create the following three make targets in WICED Studio 6.0 (or later):

- "`snip.gpio-CYW954907AEVAL1F download run`" for the gpio example which is already present in WICED Studio.
- "`kits.CYW954907AEVAL1F.config_join_ping-CYW954907AEVAL1F download run`" for the config_join_ping project.
- "`kits.CYW954907AEVAL1F.publish_subscribe_aws-CYW954907AEVAL1F download run`" for the aws publish and subscribe project.
- "`kits.CYW954907AEVAL1F.adc_measure-CYW954907AEVAL1F download run`" for the adc_measure project.

5.2 GPIO

5.2.1 Project Description

The `gpio` project demonstrates toggling of LEDs and turning them off when one of the user switches is pressed.

The `gpio` project consists of the following files:

- *gpio.c:* Contains the main application function **application_start()**, which is the entry point and execution of the firmware application.
- *gpio.mk:* This is the makefile that adds the source of the application.

5.2.2 Hardware Connections

No specific hardware connections are required for this project because all connections are hardwired on the CYW954907AEVAL1F EVK.

5.2.3 Verify Output

Do the following to verify the output:

1. Create and run a make target for the `gpio` project using the description specified in Building and Programming a Project in WICED Studio IDE.
   After initialization of the platform, LEDs will keep flashing (toggling). When a user switch is pressed, the corresponding LED turns off. The example also prints a message to the debug UART at startup.
2. Open a terminal emulation program and connect to the WICED serial port as detailed in Step 8 in the section UART Port Configuration on CYW954907AEVAL1F Kit.

5.3 Config_join_ping

5.3.1 Project Description

The `config_join_ping` project demonstrates connectivity between CYW954907AEVAL1F EVK and a Wi-Fi access point. This example is based on existing examples available in the WICED Studio 6.0 (or later) SDK namely, *apps/snip/scan, apps/snip/dct_read_write* and *test/console*. On startup, this application shows a console through which you can enter commands to scan, configure, join, and ping Wi-Fi access points.

The `config_join_ping` project consists of the following files:

- *config_join_ping.c:* This file contains the main application function **application_start()** which is the entry point and execution of the firmware application. It also contains the function definitions for joining, pinging, printing Wi-Fi configuration, scanning Wi-Fi and the scan result handler.
- *config_join_ping.mk:* This is the makefile which adds the sources, components (in this application, console and ping are used), and the name of the application. Note that the name of the makefile must match the name of the project folder for the make process to work properly. Also, the "NAME" string in the makefile must be unique among all projects in the *apps* folder.

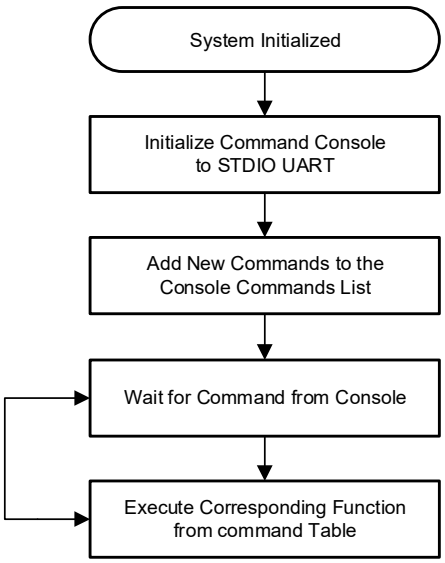
5.3.2 Hardware Connections

No specific hardware connections are required for this project because all connections are hardwired on the CYW954907AEVAL1F EVK.

5.3.3 Flowchart

Figure 5-1 illustrates the config_join_ping flowchart.

Figure 5-1. `config_join_ping` Flowchart

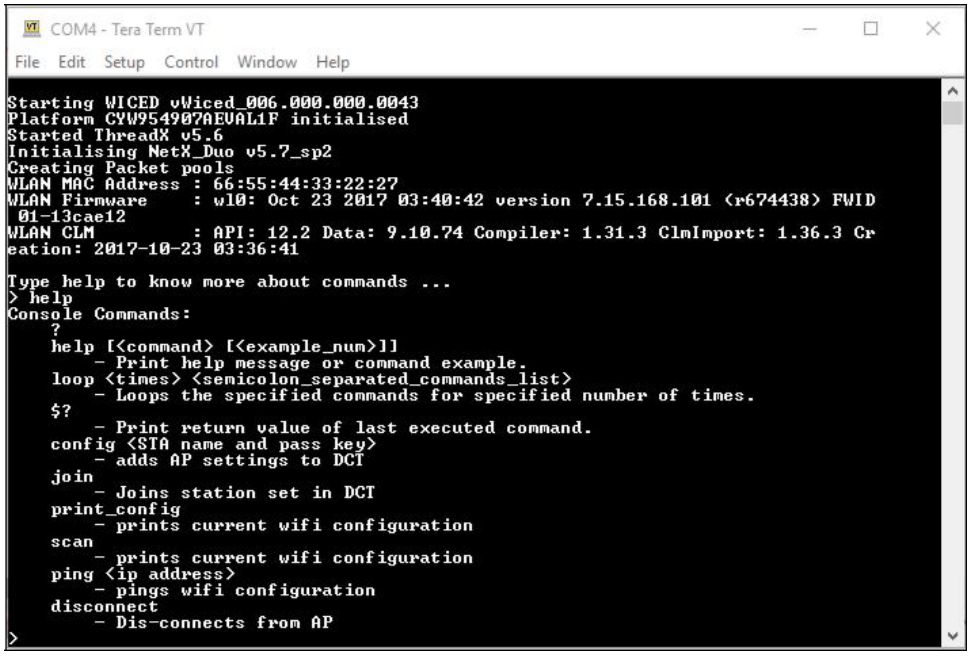


5.3.4 Verify Output

Do the following to verify the output:

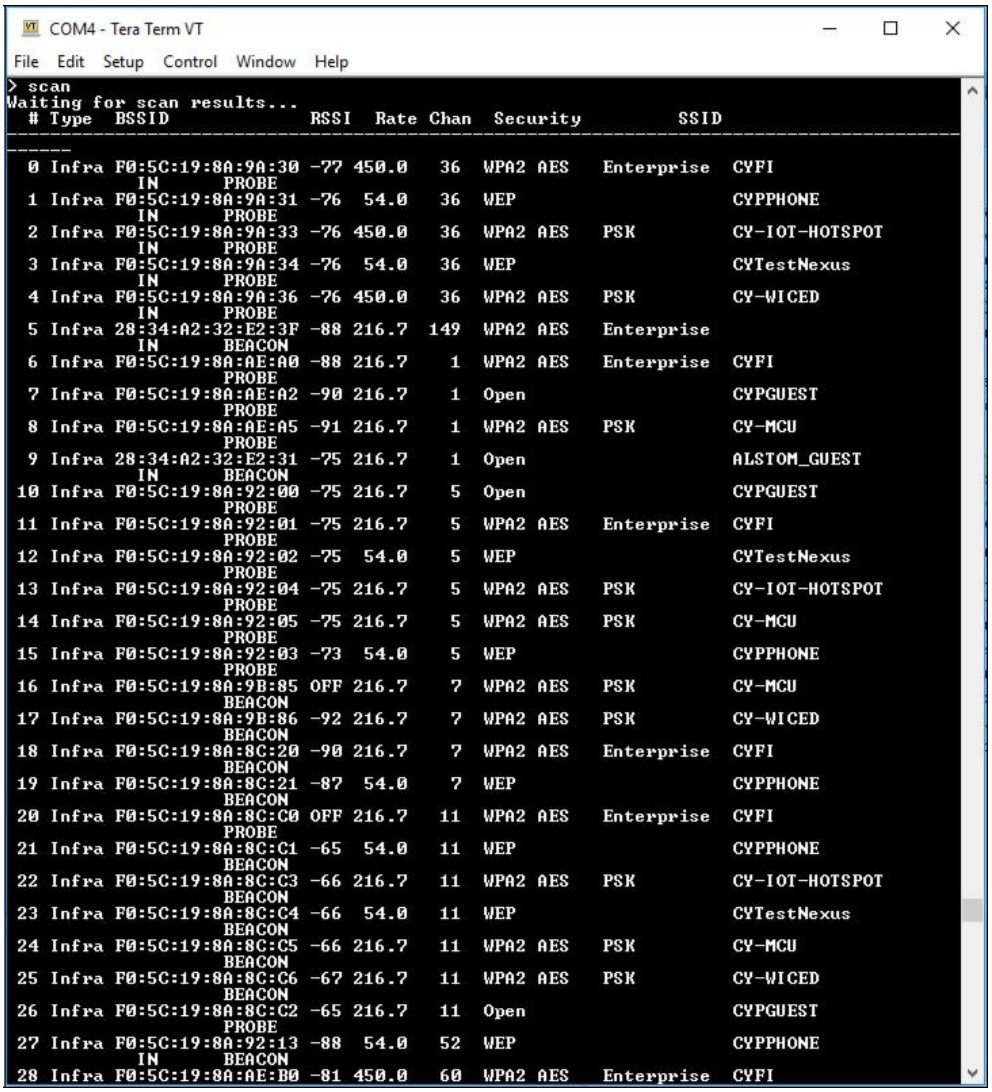
1. Create and run a make target for the `config_join_ping` project using the description specified in Building and Programming a Project in WICED Studio IDE.
2. Open a Terminal Emulation program and connect to the WICED serial port as detailed in Atep 8 in the section UART Port Configuration on CYW954907AEVAL1F Kit.
   After initialization of the platform, Wi-Fi and other components, the cursor will stop and wait for you to enter commands.
3. Type the `help` command to see the list of available commands as shown in Figure 5-2.

Figure 5-2. Initial Console Output



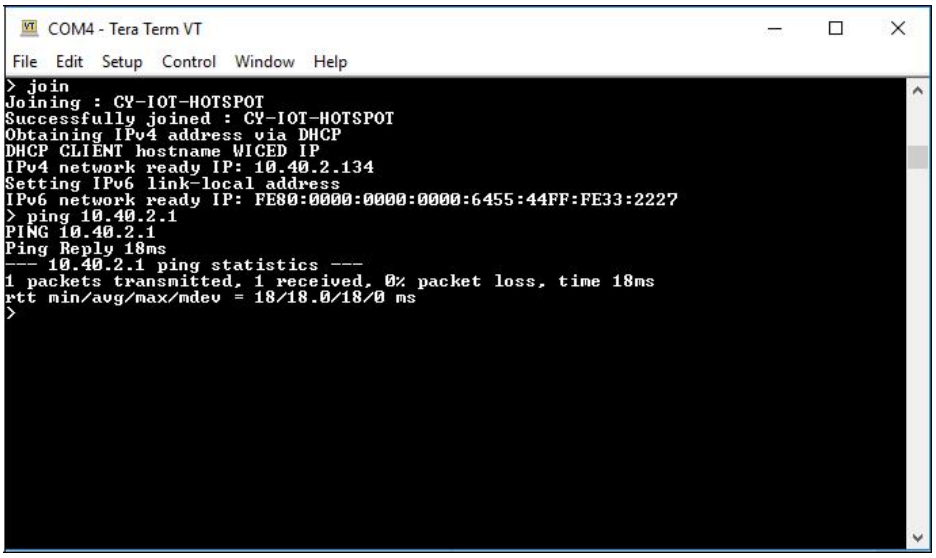
4. Type the command `scan` to find the list of available Wi-Fi access points as shown in Figure 5-3.

Figure 5-3. Scan Output



5. Type the command `config <SSID> <password>` . This command writes the given configuration in the Device Configuration Table (DCT). These values are stored in flash memory on the board.
6. Type the command `print_config` to validate if the SSID and password match and are appropriately written in the DCT.
7. Type the command `join`. The join command joins the network specified by the SSID and password from the DCT. Ping the Access point (usually 192.168.1.1) or 8.8.8.8 (IP address of Google, if your AP is connected to internet) and check if the network is up and responding. The message `"Ping Reply 11ms"` is displayed as shown in Figure 5-4.

Figure 5-4. Join and Ping



8. To disconnect from the access point, use the `disconnect` command.

The console component maintains a history of commands typed, which can be accessed using the **Up/Down** arrow keys.

5.4 ADC_measure

5.4.1 Project Description

This project demonstrates measuring values from the external ADC chip on the board and posting the values to a web page accessible from the WLAN network. This code example is based on existing code example (*apps/demo/temp_control*) available in the WICED Studio 6.0 (or later). On startup, the adc_measure code example joins the Wi-Fi Access Point specified in the *wifi_config_dct.h* file and starts a web page where the ADC count is reported.

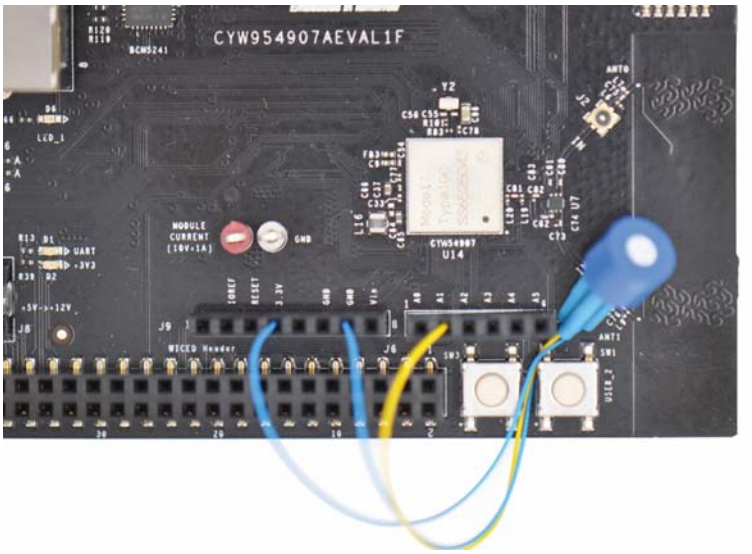
The project consists of the following files:

- *adc_measure.c:* This file contains the main application function **application_start()** which is the entry point and execution of the firmware application. It also contains the function definitions for initializing, conducting ADC measurement, starting the web page, and processing an ADC update.
- *adc_measure.mk:* This is the makefile which adds the sources, components (in this application, component HTTP_server, device_configuration, Xively, SNTP, and Gedday are used) and the name of the application. It also adds the required resources for the web page which is available in the *resources/apps* directory.
- *i2c.c:* This file contains the required function definitions for initializing and taking ADC samples from the external ADC (MAX 11615) available in the CY9W54907AEVAL1F EVK.
- *wifi_config_dct.h:* This file contains the Wi-Fi Access Point credentials (SSID and pass phrase key) and soft AP credentials. Enter the client access point name and password credentials prior to building the application. These are specified as CLIENT_AP_SSID and CLIENT_AP_PASSPHRASE. Note that the security type may also have to be changed if the access point does not use WPA2 security. The Wi-Fi access point must be connected to the internet to get the current time using Network Time protocol (NTP). If the Wi-Fi access point is not connected to the internet, then it will assume 00:00:00 UTC time and will start the web page.

5.4.2 Hardware Connections

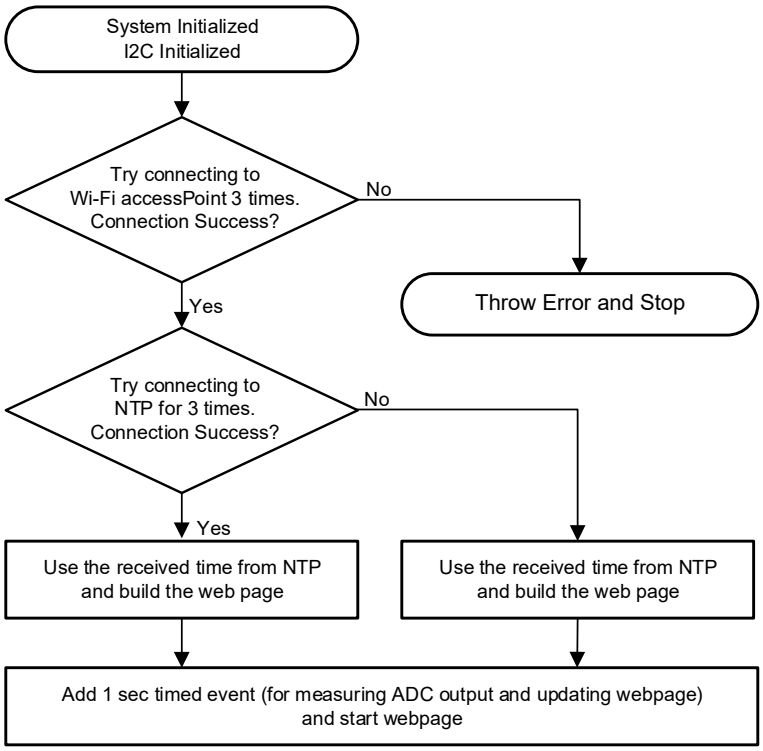
Connect a potentiometer (10 kΩ) between VCC and GND with the center terminal connected to channel 1 of the ADC (pin A1 in the Arduino header) as shown in Figure 5-5. If you do not have a potentiometer to test, you can connect a wire between VCC and ADC channel 2 (to simulate full scale) or a wire between GND and ADC channel 1 (to simulate zero scale). Alternately, you can connect an adjustable DC power supply to ADC channel 1.

Figure 5-5. Potentiometer Connection



5.4.3 Flowchart

Figure 5-6. adc_measure Flowchart



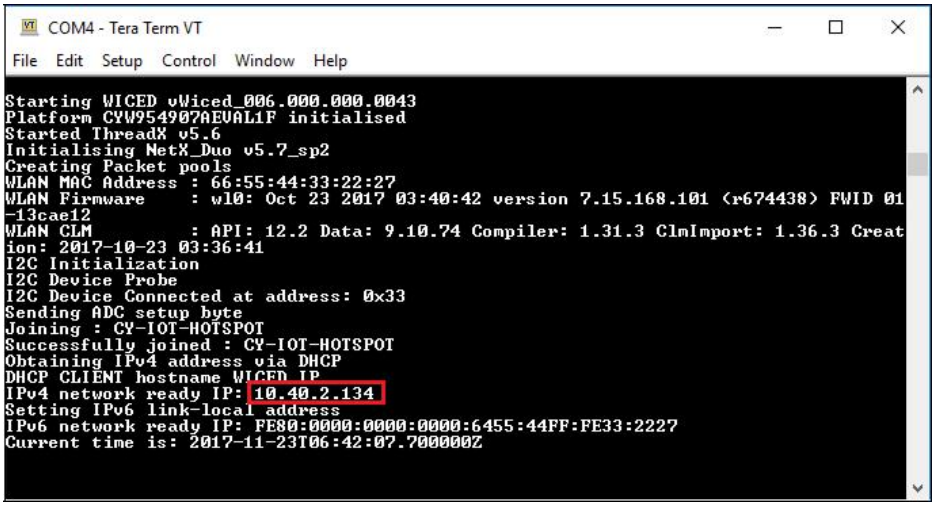
5.4.4 Access Point Credentials

1. Enter your credentials (SSID and pass phrase key) in the *wifi_config_dct.h* file.
2. Update the following macros:
   - `CLIENT_AP_SSID`: update with your access point's SSID
   - `CLIENT_AP_PASSPHRASE`: update with your access point's pass phrase key
   - `CLIENT_AP_SECURITY`: update with the security type of your access point. This is "`WICED_SECURITY_WPA2 _MIXED _PSK`" if the access point uses WPA2-PSK. If the AP uses a different security mechanism, choose the correct one defined in `enum wiced_security_t` from 43xxx_Wi-Fi\WICED\WWD\include\wwd_constants.h.

5.4.5 Verify Output

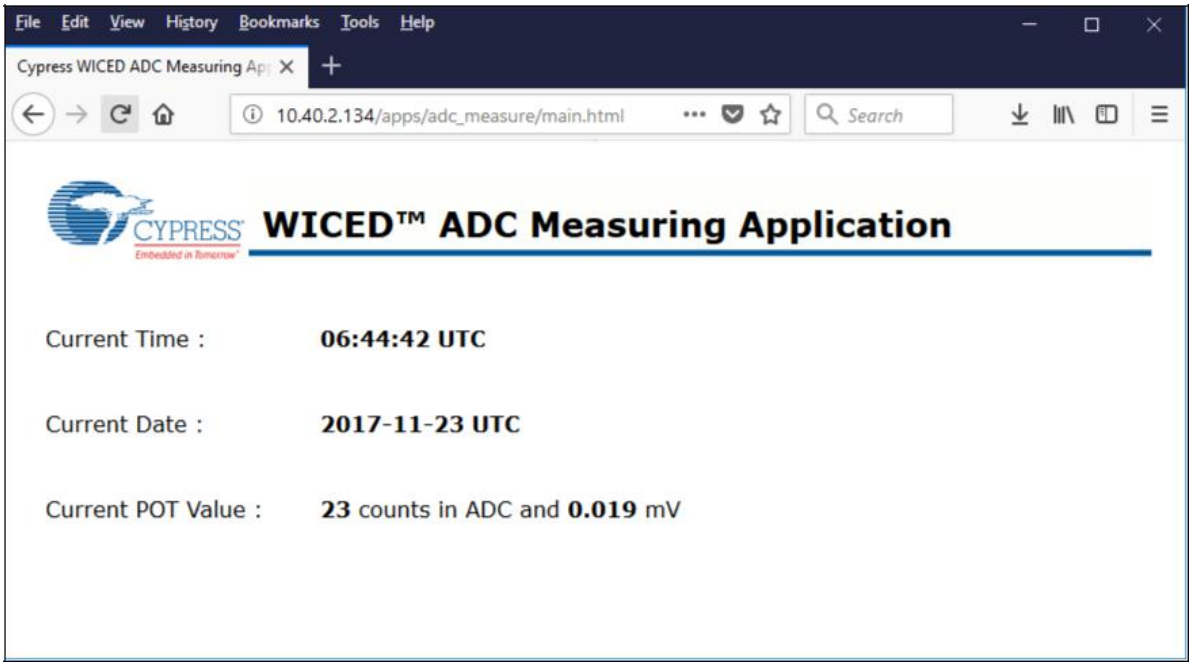
1. Create and run a make target for the adc_measure project similar to the procedure provided in Building and Programming a Project in WICED Studio IDE.
2. If connection to the Wi-Fi access point is successful, wait for NTP time request to complete. The output of the terminal program should be similar to the screenshot in Figure 5-7.

Figure 5-7. NTP Success



3. Enter the IP address as the URL in your web browser, as shown in the terminal output in Figure 5-7, such as 10.40.2.134.
   The browser will show the output as shown in Figure 5-8. Note that the PC and CYW954907AEVAL1F EVK should be connected to the same access point.
4. Rotate the potentiometer and verify that the value shown on web page changes accordingly. One easy way to validate the correct functioning is to rotate the potentiometer to one of the extremes and observe if the full-scale value appears. If you do not have access to a potentiometer, you can use an adjustable power supply or wires to connect 3.3V and GND to the ADC input alternatively.

Figure 5-8. Webpage



5.5 Publish_subscribe_aws

5.5.1 Project Description

This project demonstrates publishing a message to a *Thing* in the Amazon Web Services (AWS) cloud and subscribing to the same messages. A *Thing* is a representation of a specific device or logical entity. For more information, refer to the AWS Documentation.

This example is based on existing code example (*apps/aws_iot/publish and apps/aws_ iot/subscribe*.) available in WICED Studio 6.0 (or later). On startup, the publish_subscribe_aws code example joins a Wi-Fi access point specified in the *wifi_config_dct.h file*, connects to AWS, subscribes to the specified topic and then alternately tries to publish LIGHT ON and LIGHT OFF messages.

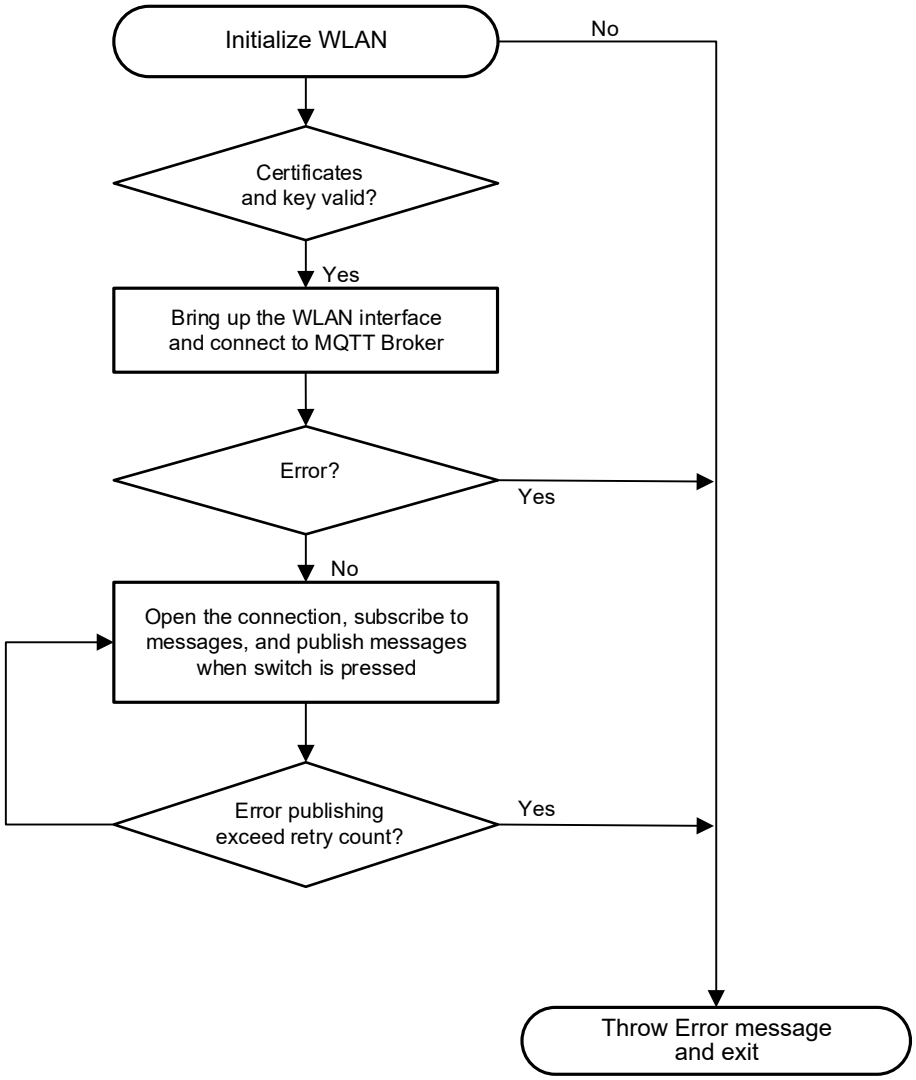
The project consists of the following files:

- *publish_subscribe.c:* This file contains the main application function **application_start()**, which is the entry point and execution of the firmware application. It also contains the function definitions for initializing, publishing and subscribing to AWS.
- *publish_subscribe_aws.mk:* This is the makefile which adds the sources, protocols, components (in this application, the MQTT component is used) and the name of the application. It also adds the required resources for the web page which are available in the *resources/apps* directory. Note that this project uses certificates from *apps/aws_iot* directory.
- *wifi_config_dct.h:* This file contains the Wi-Fi access point credentials (SSID and pass phrase key) and soft AP credentials. You should enter the client access point name and password credentials prior to building the application. These are specified as `CLIENT_AP_SSID` and `CLIENT_AP_PASSPHRASE`. Note that the security type may also have to be changed if the access point does not use WPA2 security. The Wi-Fi access point must have access to the internet to connect with AWS.

5.5.2 Hardware Connections

No specific hardware connections are required for this project because all connections are hardwired on the CYW954907AEVAL1F EVK.

5.5.3 Flowchart



5.5.4 Verify Output

5.5.4.1 Set up an AWS Account and Create a Thing, Policy, and Certificate

An AWS account allows you to view AWS account activity, view usage reports, and manage AWS Security Credentials. When you sign up for AWS, your AWS account is automatically signed up for all services in AWS, including AWS IoT. You are charged only for the services that you use.

For more information about AWS IoT, see the help pages of AWS here.

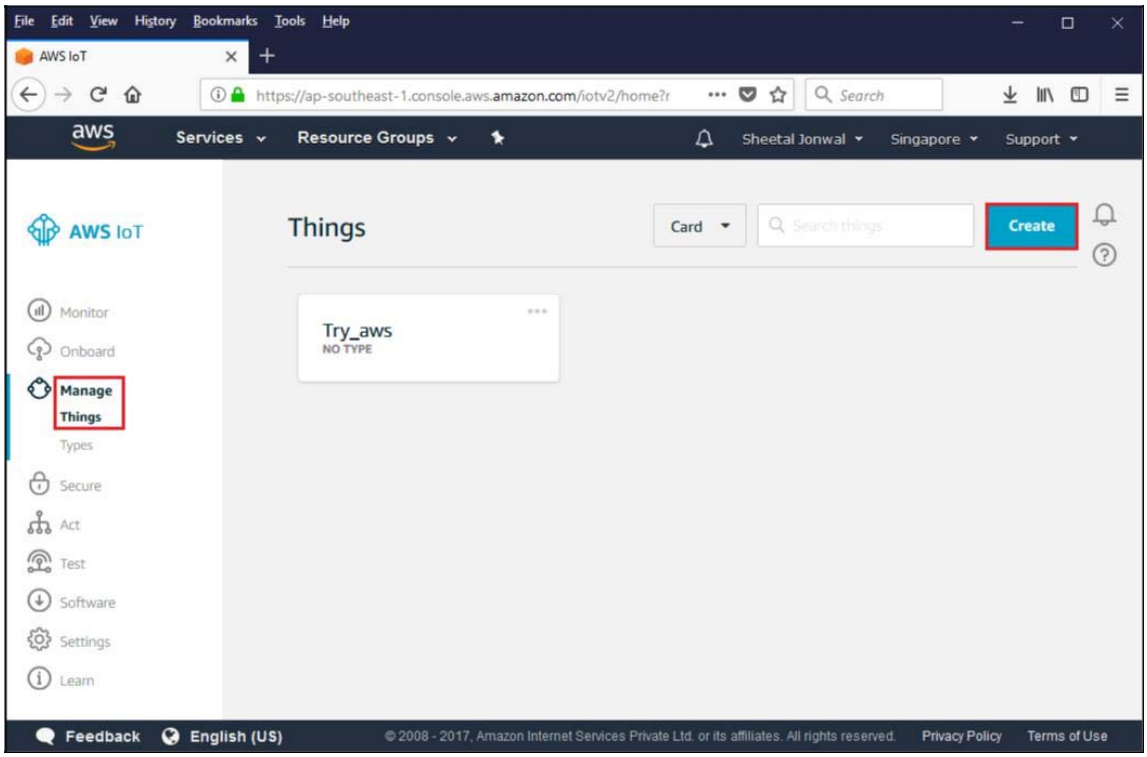
Do the following to set up a new account:

1. Open https://aws.amazon.com and choose **Create an AWS Account**.
2. Follow the online instructions. Part of the sign-up procedure involves receiving a phone call and entering a PIN using the phone keypad.
3. In the Console Home page, select your AWS Region (in this example Asia pacific (Singapore) is used), and choose the **AWS IoT service**. The AWS IoT Console window appears.

Create a Thing

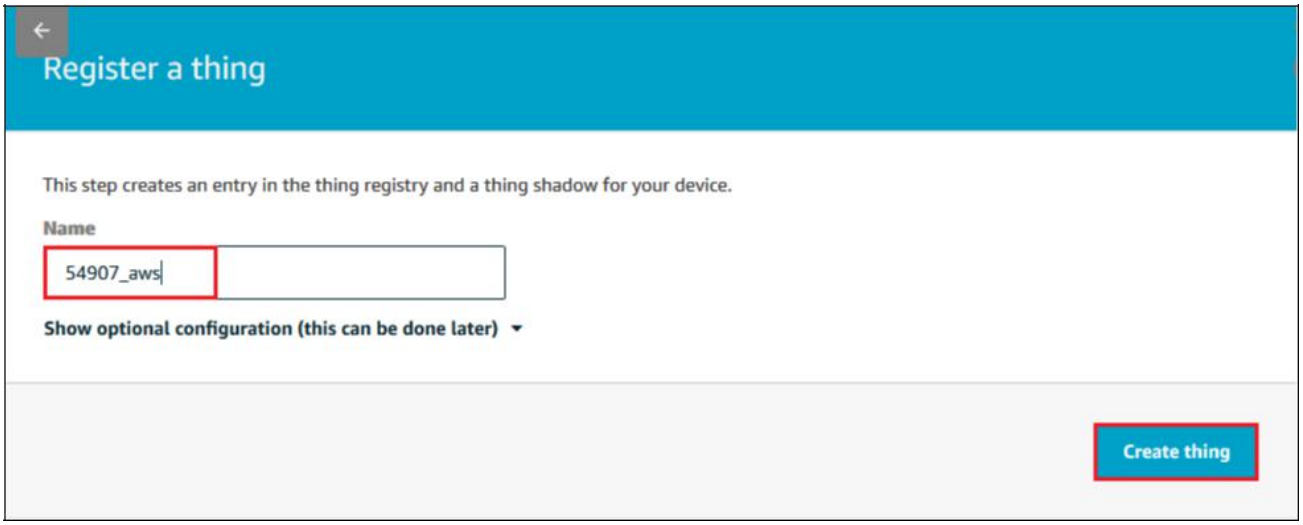
1. In the AWS IoT Console window, choose **Manage > Things** on the left-hand panel, and then click **Create** as shown in Figure 5-9.

Figure 5-9. Create Thing



2. Each Thing is uniquely identified by its name. Assign a name in the **Name** field, and click **Create thing**. For example, “54907_aws”.

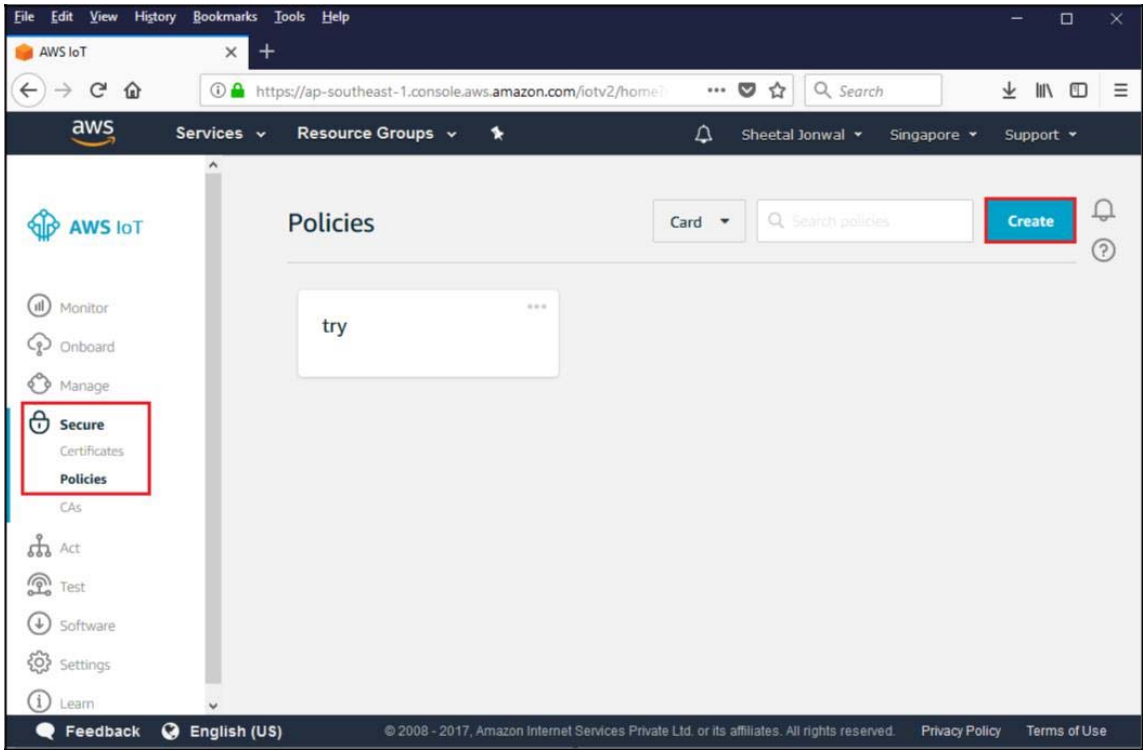
   Note: It is possible to exchange messages without a need to create a thing (by having a certificate with an attached policy), but it is recommended by AWS to create it.



3. In the created Thing window, click the left arrow to navigate back to the AWS IoT Console window.

Create a Policy

1. In the AWS IoT Console window, go to **Secure > Policies**, and then click the **Create** button. The Create a policy window appears.



2. Assign a policy name in the **Name** field. For example, “54907_policy”.

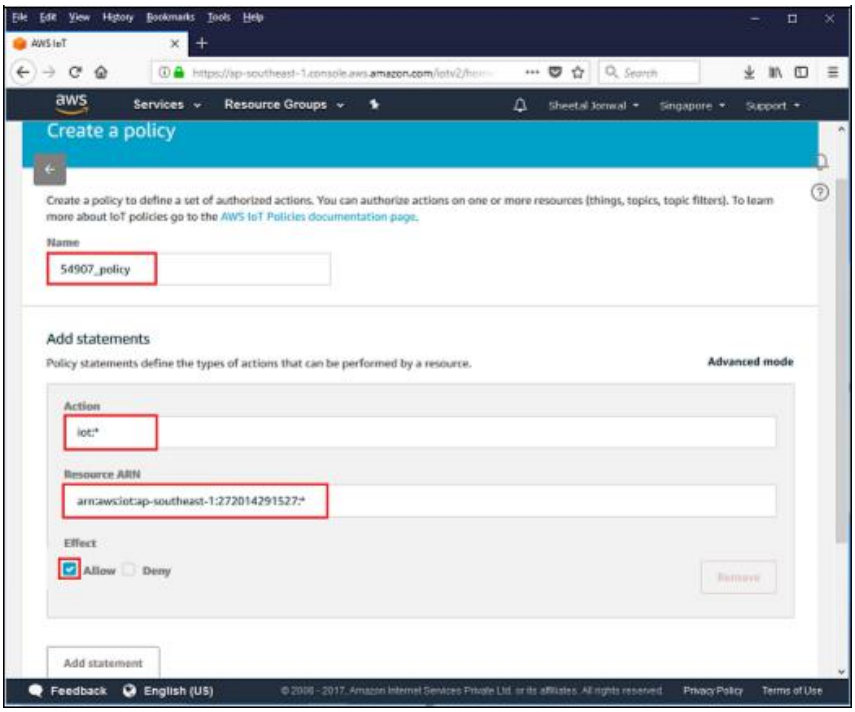
3. In Add statement, specify the Action as **iot:***.
4. Assign an Amazon Resource Name (ARN) in the **Resource ARN** field. To use a wild card, change the last part of Resource ARN as follows:
   `"arn:aws:iot:us-east-1:xxxxxxxxxxxx:topic/replaceWithATopic"`
   to
   `"arn:aws:iot:us-east-1:xxxxxxxxxxxx:*`

Notes:

- Use the region that you selected when you set up your account.
- Replace `xxxxxxxxxxxx` with the appropriate value for your ARN.
- In the ARN name, ensure that you change `"topic/replaceWithATopic"` to "*", where "*" indicates all topics. If you want to use the certificates only for a specific topic (in this case, `"54907_led_onoff"` is the one defined as `WICED_TOPIC` macro in *publish_subscribe.c*), use the following Resource ARN:
  `"arn:aws:iot:us-east-1:xxxxxxxxxxxx:54907_led_onoff"`.

5. Select the **Allow Effect** check box and then click the **Create** button as shown in Figure 5-10.
6. In the created policy window, navigate back to the AWS IoT Console window.

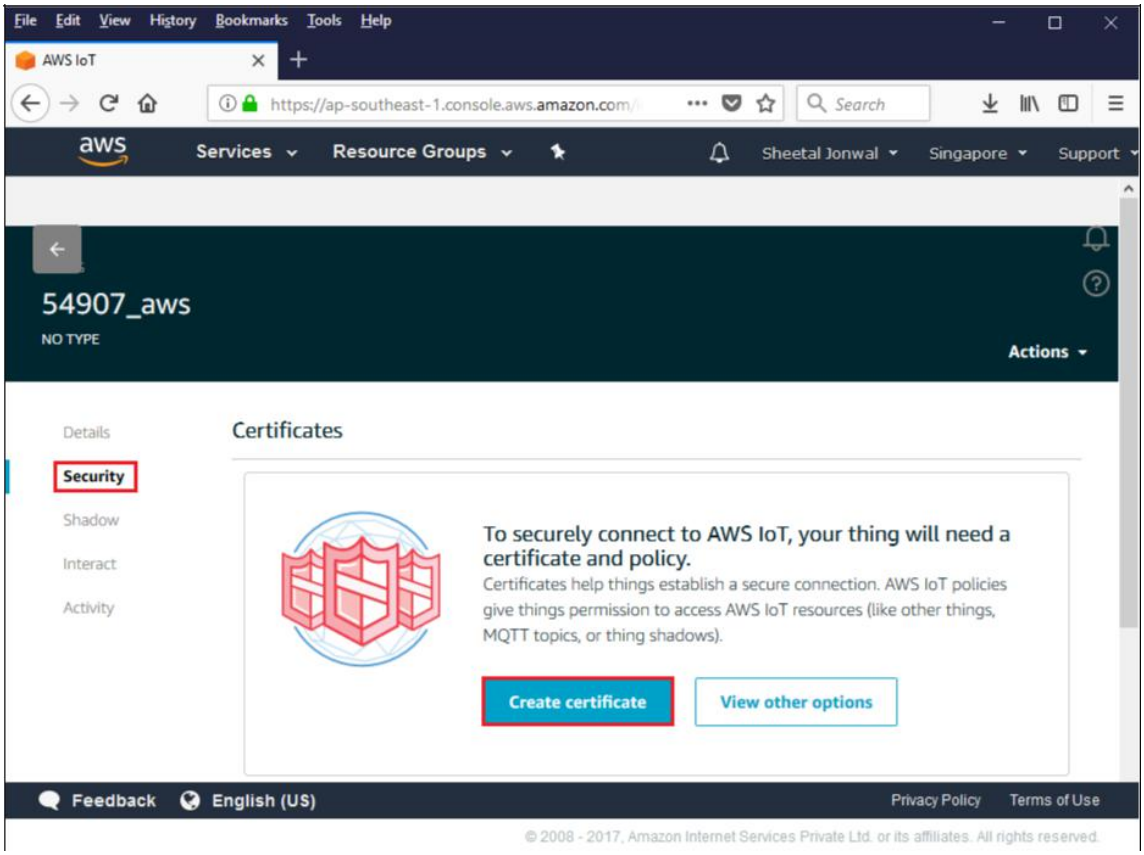
Figure 5-10. Create Policy



Create a Certificate for a Thing

1. In the AWS IoT Console window, go to **Manage > Things**, and then click the created Thing (for example: 54907_aws).
   The created Thing window appears.
2. In the left navigation pane, click **Security** and then click **Create certificate**.

Figure 5-11. Create Certificate



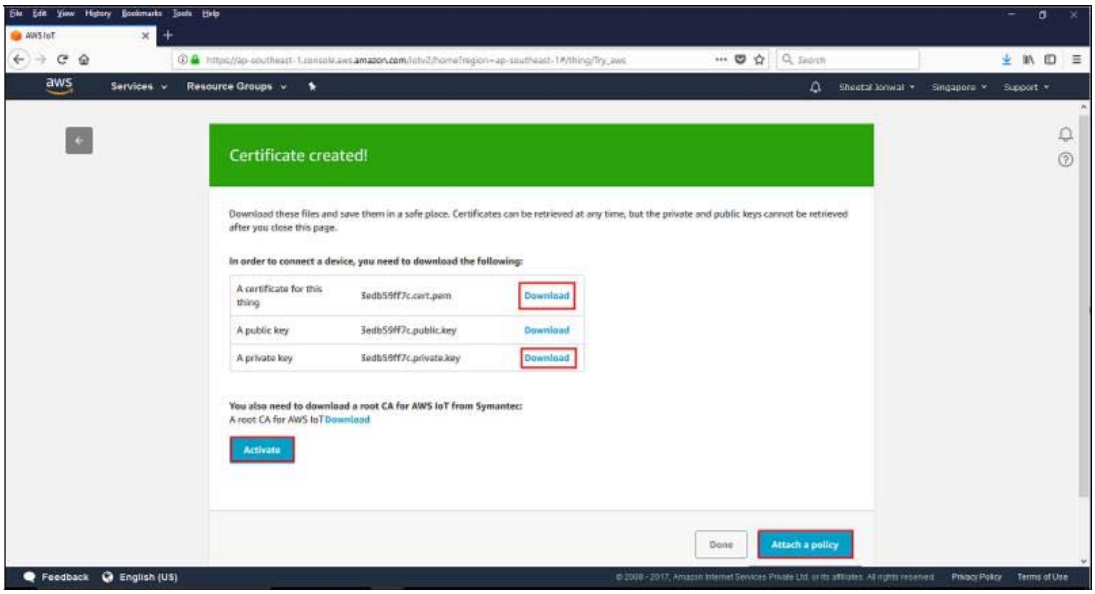
3. On the Certificate created page, click the **Download** button for the certificate and private key to save each of them to your PC.

Notes:

- The certificate and private key cannot be revisited later for download and must be saved while creating the Certificate.
- Back up the existing <*WICED-SDK*>*43xxx_Wi-Fi**resources**apps**aws_iot**client.cer* and rename the downloaded certificate as *client.cer* in <*WICED-SDK*>*43xxx_Wi-Fi**resources**apps**aws_iot*\.
- Back up the existing <*WICED-SDK*>*43xxx_Wi-Fi**resources**apps**aws_iot**privkey.cer* and rename the downloaded private key as privkey.cer in <*WICED-SDK*>*43xxx_Wi-Fi**resources**apps**aws_iot*\.

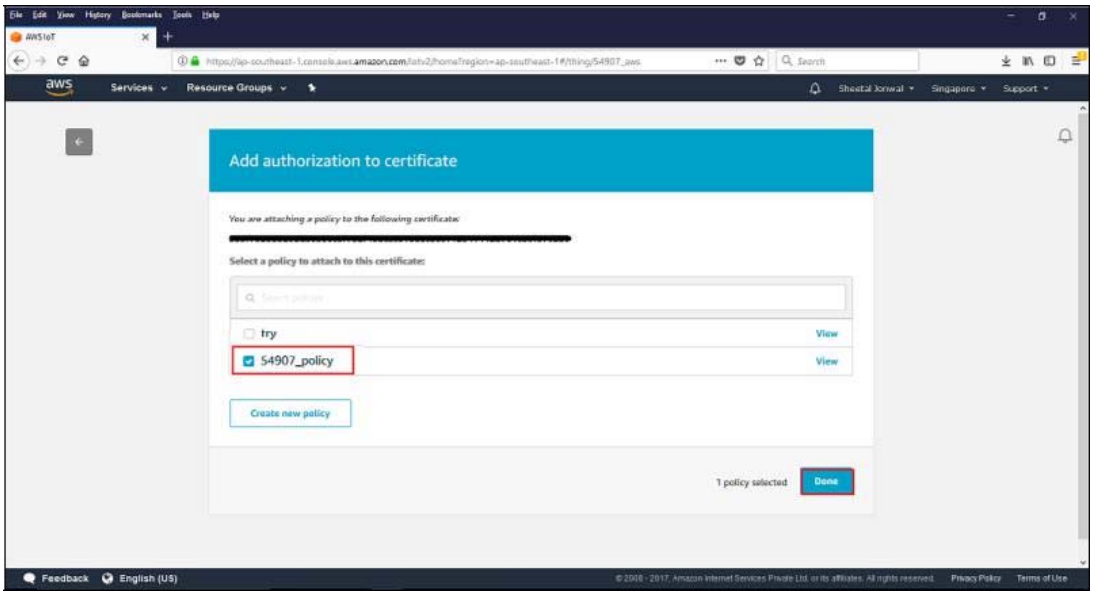
4. Click the **Activate** button and then click the **Attach a policy** button as shown in Figure 5-12. The **Add authorization to certificate** window appears (see Figure 5-13).

Figure 5-12. Activate and Attach `policy`



5. Select the check box next to the policy you want to choose and then click **Done**.

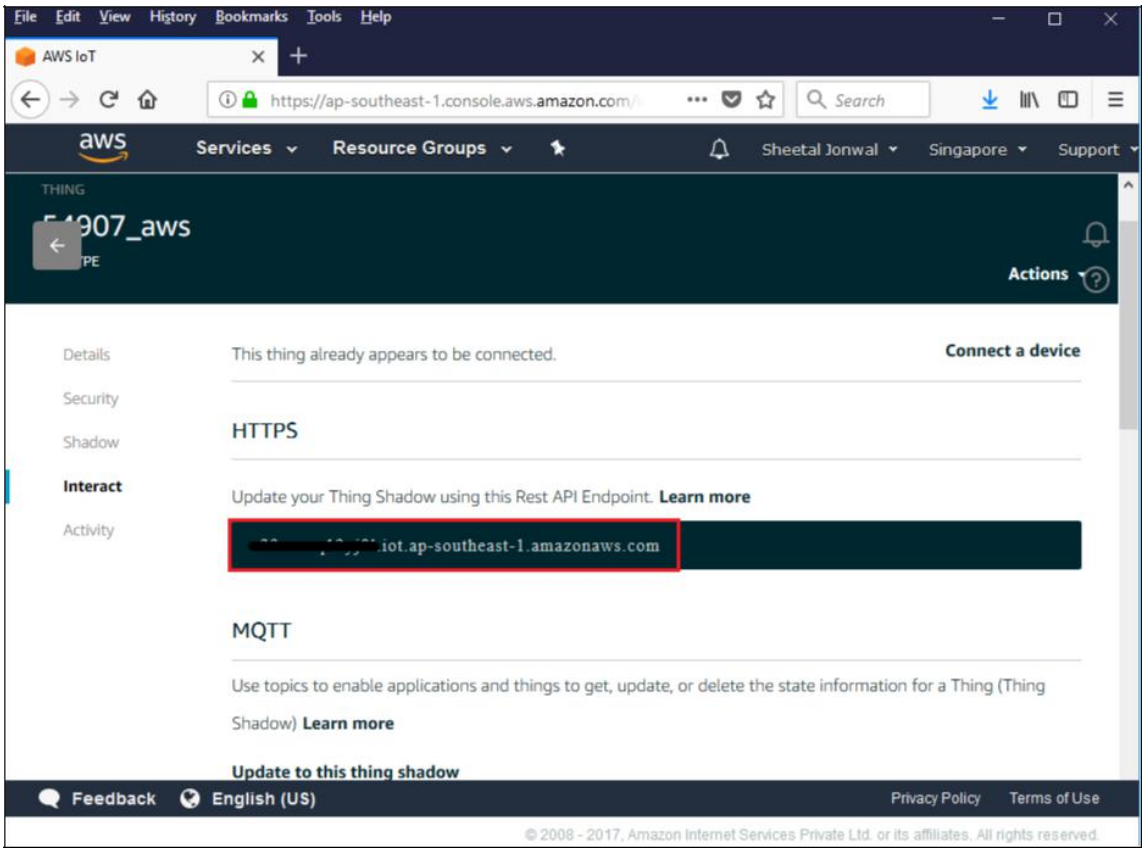
Figure 5-13. Select Policy



6. In the created policy window, click to navigate back to the AWS IoT Console window.
7. Go to **Security > Certificates**. The Certificates window appears.
8. Click the created certificate. The Certificate ARN window appears.
9. Click **Policies** in the left-hand panel to validate if the correct policy is linked.
10. Click **Things** in the left-hand panel to validate if the correct Thing is linked.
11. Click the specific Thing. The Thing ARN window appears.

12. In left navigation pane, choose **Interact**.
13. Copy the Endpoint from the HTTPS tab as shown in Figure 5-14

Figure 5-14. Endpoint.



14. Navigate to the *publish_subscribe.c* file to update the `MQTT_BROKER_ADDRESS` macro with the endpoint address copied from **HTTPS** tab. Remove the first string before "." in endpoint and replace it with * and copy it to the `REGION` macro.
    In this case, it is `"*.iot.ap-southeast-1.amazonaws.com"`.

The created Thing, policy and certificate are used to interact with the AWS IoT.

5.5.4.2 *Access Point Credentials*

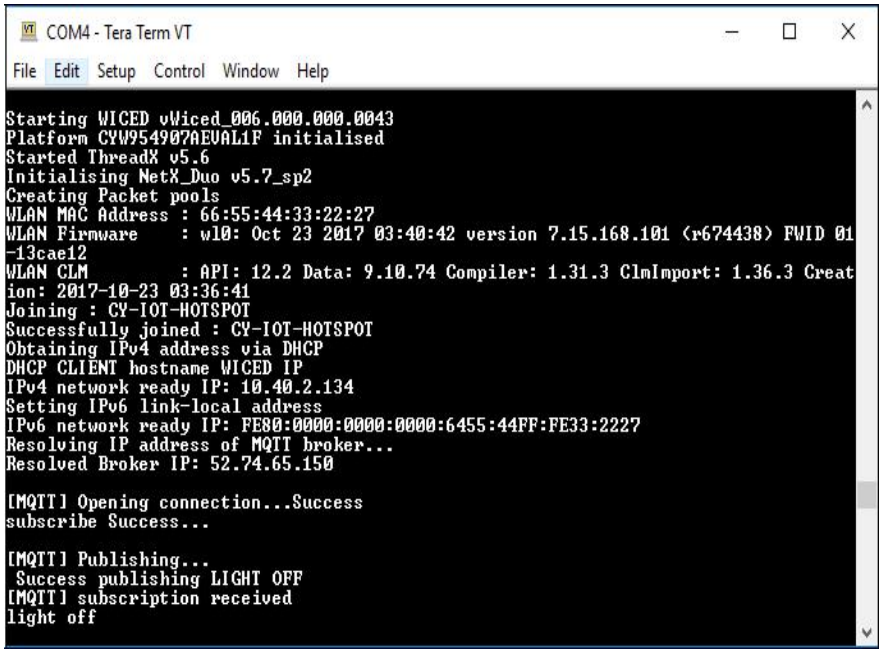
1. Enter your credentials (SSID and pass phrase key) in the *wifi_config_dct.h* file.
2. Update the following macros:
   - `CLIENT_AP_SSID`: update with your access point's SSID
   - `CLIENT_AP_PASSPHRASE`: update with your access point's pass phrase key
   - `CLIENT_AP_SECURITY`: update with the security type of your access point. This is `WICED_SECURITY _WPA2_MIXED_PSK` if your access point uses WPA2-PSK. If your AP uses different security then choose correct one defined in `enum wiced_security_t` from *43xxx_Wi- Fi\WICED\WWD\include\wwd_constants.h*.

5.5.4.3 Build, Program, and Verify

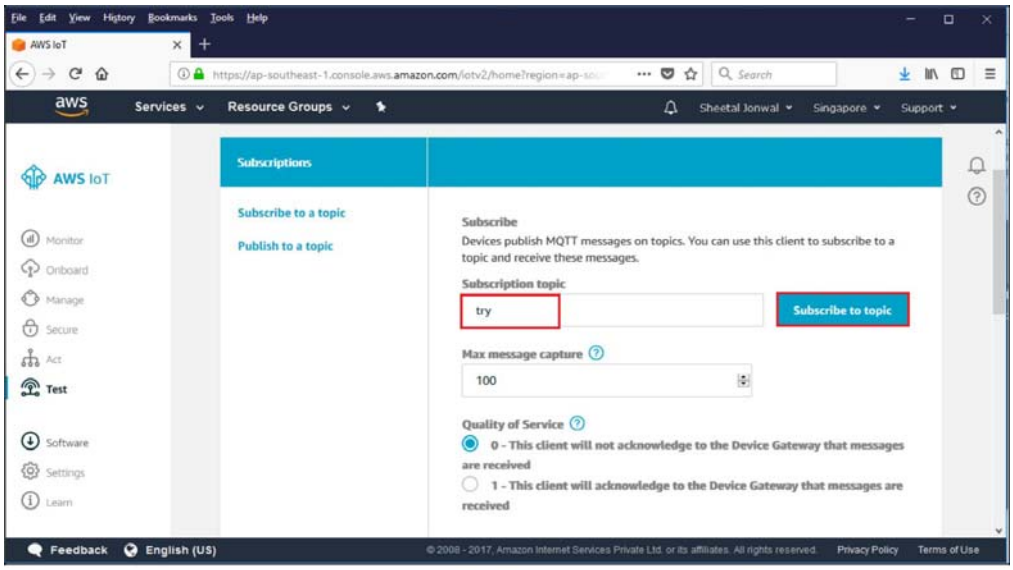
Your Wi-Fi access point must be connected to the internet to verify the example.

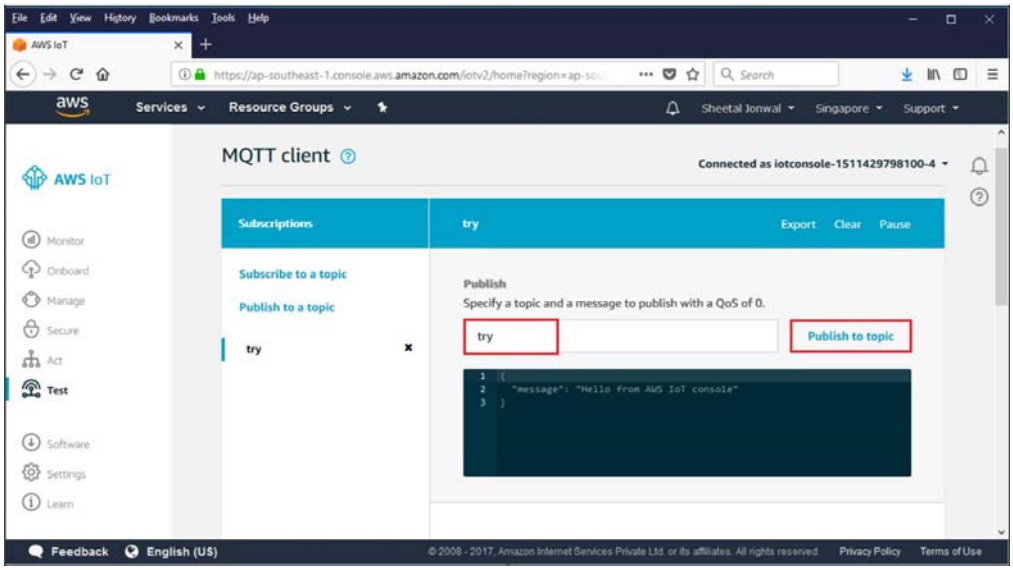
1. Build and program the `publish_subscribe_aws` example using a similar procedure to the one provided in Building, Programming, and Debugging.
   Once programmed, the CYW954907AEVAL1F EVK will try to connect to AWS IoT and subscribe to the specified topic.
2. Press the switch USER _1 to turn LED_1 ON and OFF alternately as shown in Figure 5-15. Note that this is being done over the cloud. That is, pushing the switch publishes a message to the cloud. The LED turns on in response to a notification from the cloud. You can also see observe the messages inside the AWS console itself.
3. In the AWS IoT Console window, go to **Dashboard > Messages published** to observe the number of messages exchanged.

Figure 5-15. `Publish_Subscribe` Output



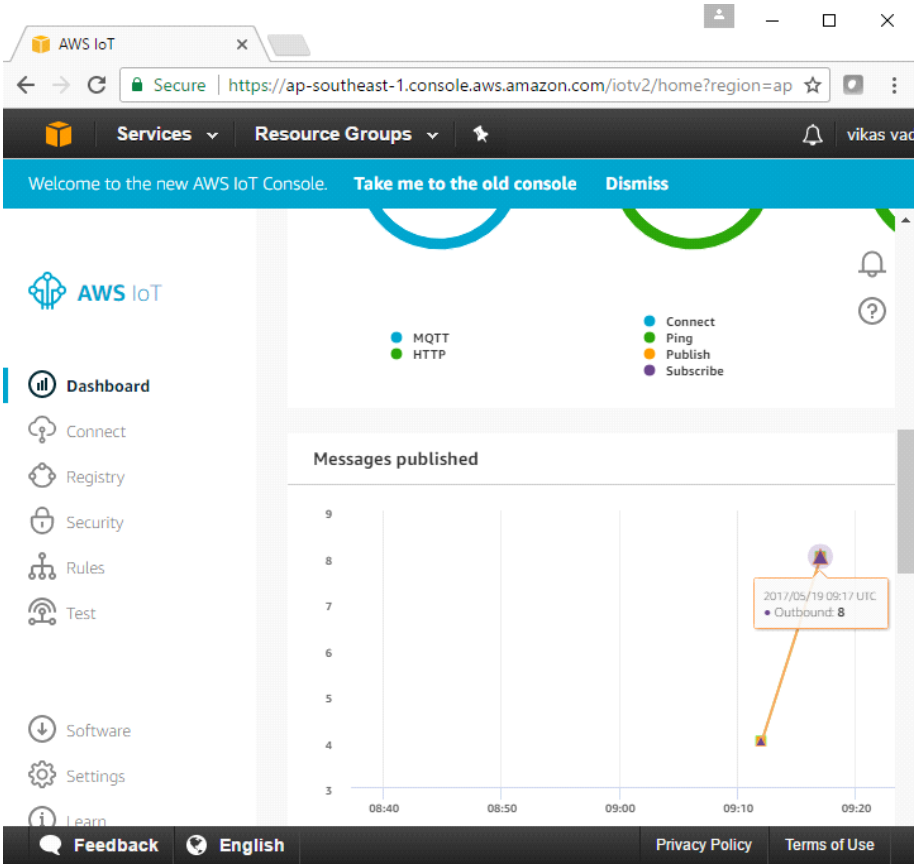
4. Go to **Test**. Type a topic name and click on **Subscribe to topic**. Publish to the same topic.





You will see the message published as follows in **Monitor** section.

Figure 5-16. Messages Published



Revision History

Document Title: CYW954907AEVAL1F Evaluation Kit User Guide Document Number: 002-22338			
Revision	**ECN#**	**Issue Date**	**Description of Change**
**	6010685	01/02/2018	Initial release
*A	6894645	06/08/2020	Revised and added contents in Section 4.1 Bootstrap and Control Pins