SILICON LABS

works with
BY SILICON LABS

WF-301

Designing Low Power Applications with Wi-Fi 6

Alfredo Pérez Grovas | August 2023

Agenda

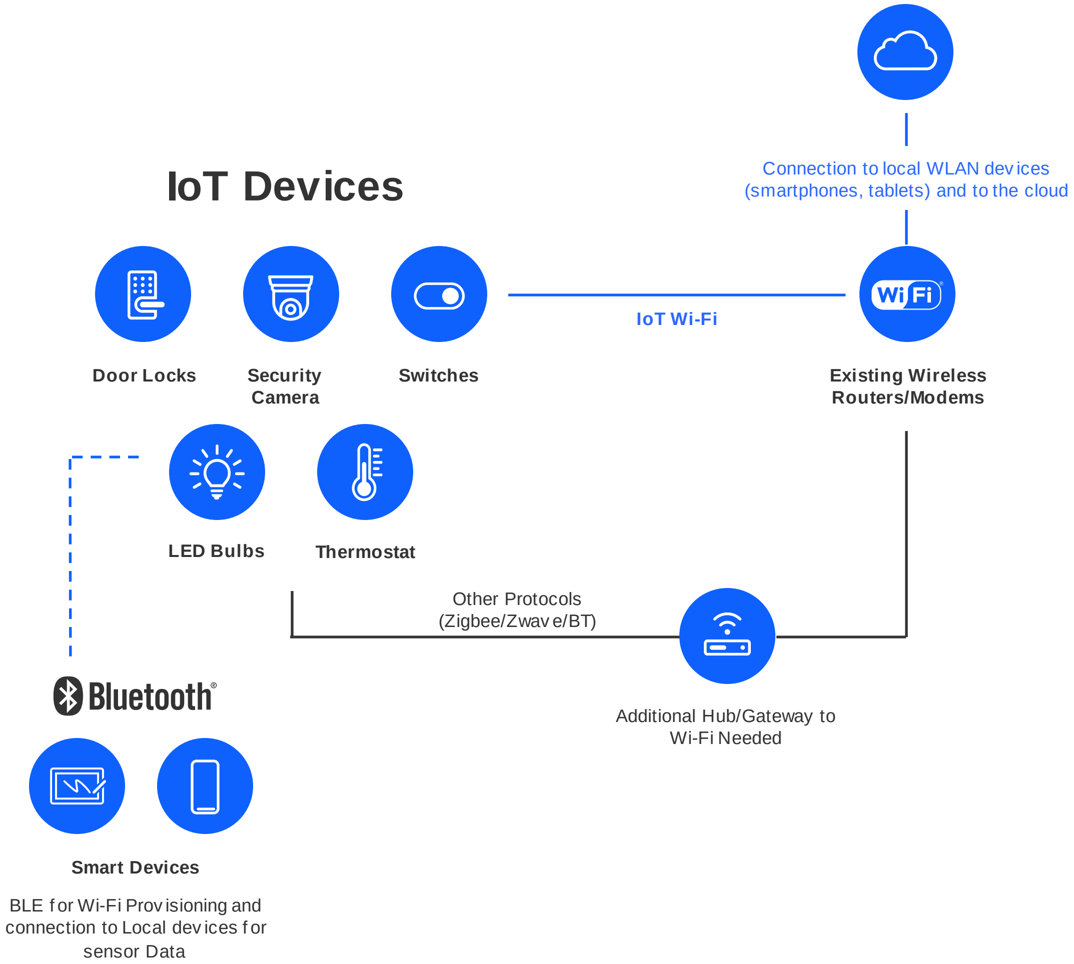
- Introduction
- Wi-Fi 6 Power Saving Features
- SiWx917 Development Kit Overview
- SiWx917 Code Examples Available in SDK
- Going Deep into a Code Example: Wi-Fi-only Standby Associate (DTIM/Listen Interval)
- Going Deep into a Code Example: Wi-Fi-only Standby Associate (TWT)
- Low Power Measurement of SiWx917 using AEM
- Silicon Labs’ Wi-Fi Portfolio

SILICON LABS ©2023 Silicon Laboratories Inc. All rights reserved.

works with
BY SILICON LABS

Introduction

Wi-Fi Usage in IoT Applications


IoT Devices
Connection to local WLAN devices (smartphones, tablets) and to the cloud
Door Locks
Security Camera
Switches
IoT Wi-Fi
Wi Fi
Existing Wireless Routers/Modems
LED Bulbs
Thermostat
Other Protocols (Zigbee/Zwave/BT)
Additional Hub/Gateway to Wi-Fi Needed
Bluetooth
Smart Devices
BLE for Wi-Fi Provisioning and connection to Local devices for sensor Data

- **Simplified installations and cost reductions:**
  - Use existing Wi-Fi router/modem
  - Native IP protocol for internet communication
  - No additional Hub/Gateway required
- **Extended range, battery life, throughput**
  - Energy efficient and longer range 2.4GHz single-band
  - Power saving capabilities
  - Higher data rate support
- **Improve user experience and interoperability with**
  - The new Matter protocol
  - Ecosystem cloud integration and connectivity
  - Local area network connectivity
- **Bluetooth Low Energy usage with Wi-Fi**
  - Simplified provisioning
  - Proximity detection
  - Sensor connectivity

©2023 Silicon Laboratories Inc. All rights reserved.

Low Power Requirements for IoT Wi-Fi Devices

- **Why Low Power?**
  - IoT devices are different from traditional Wi-Fi devices such as laptops, tablets and cell phones
    - Limited resources (MCU, memory, etc.)
    - Lower requirements (lower throughput)
  - Like laptops, tablets and cell phones, they tend to be battery powered
  - Their batteries are expected to last long periods of time (months or years) before being replaced.
- **What are the main requirements?**
  - Low power consumption to ensure long battery lifetime
  - Wireless and networking stack integration
  - Cloud connectivity
  - Cost and size constraints
  - (Newer / future) AI/ML integration

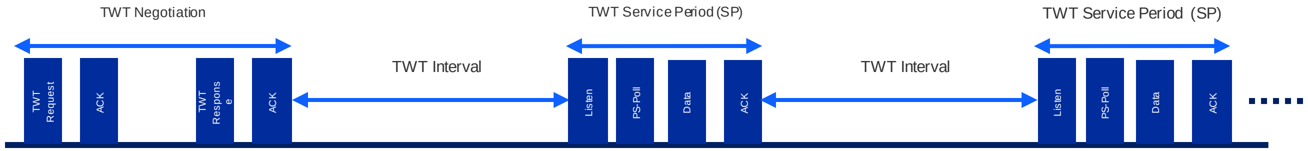
©2023 Silicon Laboratories Inc. All rights reserved.

works with
BY SILICON LABS

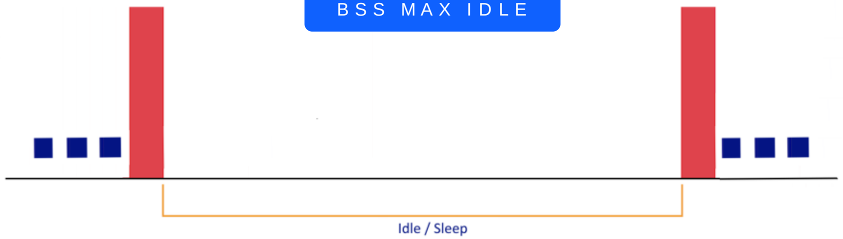
Wi-Fi 6 Power Saving Features

Wi-Fi 6 Power Saving Features

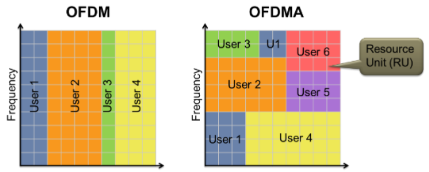
TARGET WAKE TIME

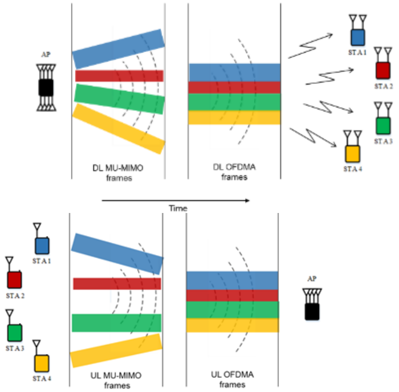


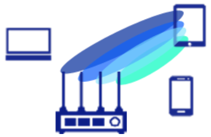
BSS MAX IDLE

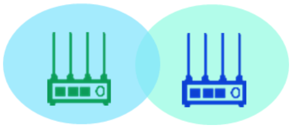


OTHER WI-FI 6 FEATURES







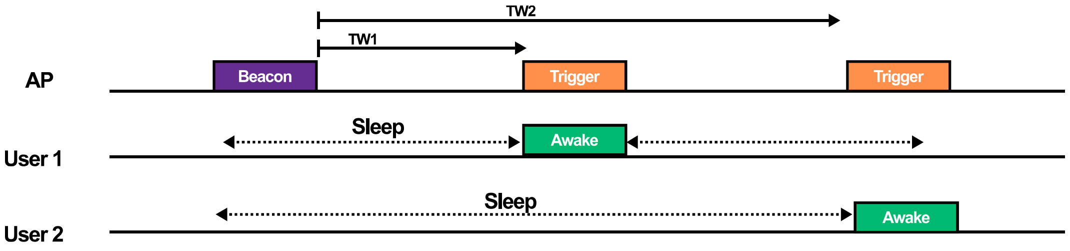


- **Wi-Fi 6 is meant to support battery-powered devices**
  - Wi-Fi 4 provided power saving mechanisms sufficient to support traditional mobile devices (cell phones, tablets)
  - The battery lifetimes of those devices are in hours or days in the best case
  - Wi-Fi 6 was designed to support IoT and other low power devices with battery lifetimes of months or years
- **Wi-Fi 6 Power saving features**
  - Target Wake Time (TWT)
  - BSS Max Idle
- **Other Wi-Fi 6 features that help current consumption**
  - OFDMA (Orthogonal Frequency Division Multiple Access)
  - Beamforming
  - Multi User MIMO (MU-MIMO)
  - BSS Coloring

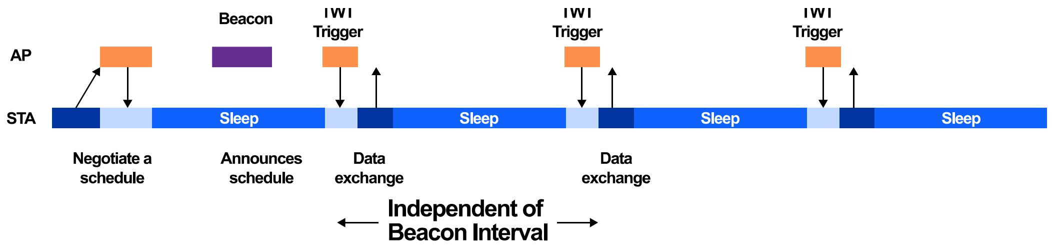
BSS: Basic Service Set

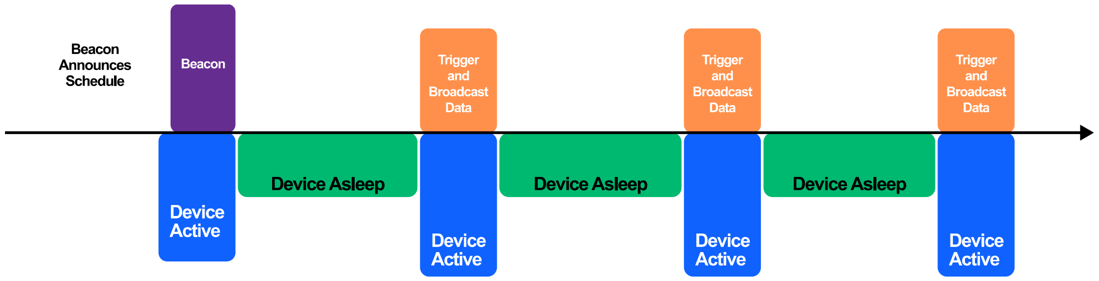
©2023 Silicon Laboratories Inc. All rights reserved.

Target Wake Time (TWT)



INDIVIDUAL TWT




Beacon Announces Schedule
Beacon
Trigger and Broadcast Data
Trigger and Broadcast Data
Trigger and Broadcast Data
Device Active
Device Asleep
Device Active
Device Asleep
Device Active
Device Asleep
Device Active

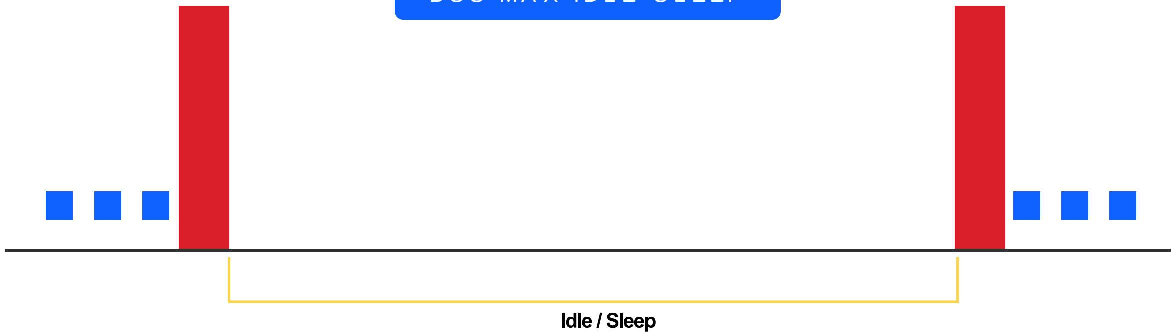
Broadcast TWT

- **TWT enables wireless AP and devices to negotiate and define specific times to access the medium.**
  - Enables devices to determine when and how frequently they will wake up to send or receive data (independent of Beacon)
- **TWT has two methods available**
  - Individual TWT: each device can negotiate sleep period with AP
  - Broadcast TWT: AP provides sleep period for a group of devices
- **Individual TWT is ideal for battery operated IoT devices**
  - Further reduces power consumption for devices on battery
  - Eliminates interop issues due to client long sleep durations
  - Optimize spectral efficiency by reducing contention
  - Combined with other Wi-Fi 6 features helps significantly reduce power consumption in congested environments compared to previous generation Wi-Fi
- **TWT provides three major benefits**
  - Allows Wi-Fi stations to increase their sleep times
  - Reduces contention between stations by scheduling air usage times.
  - Helps collect information from devices on the network through channel sounding

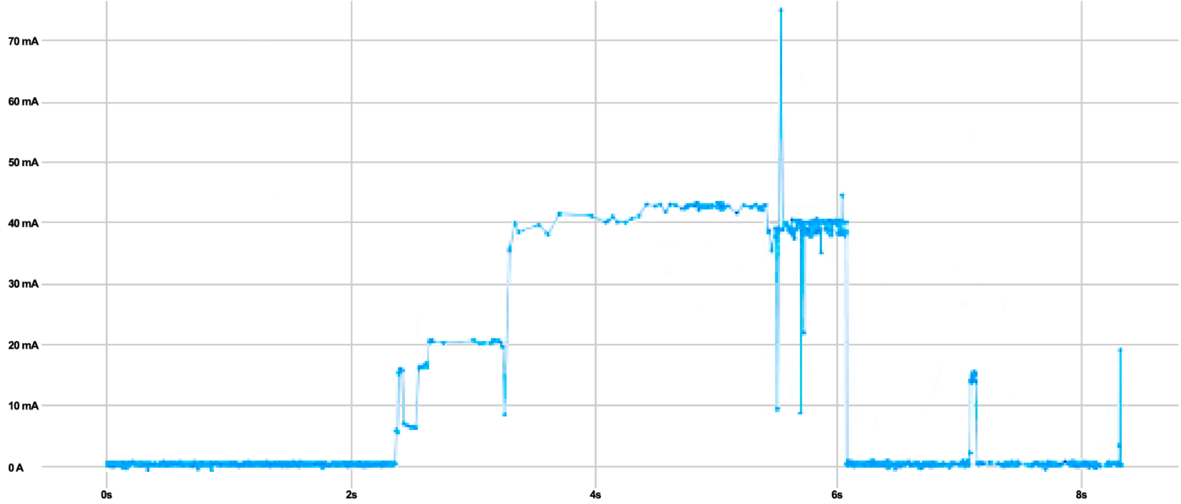
Wi-Fi 6 TWT further reduces power consumption for devices on battery, enabling longer battery life

BSS Max Idle

BSS MAX IDLE SLEEP



WPA2 ASSOCIATION CURRENT CONSUMPTION

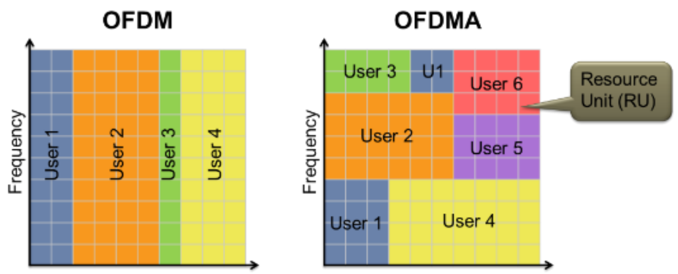


- **BSS Max Idle**
  - In typical Wi-Fi networks, stations associated to an AP must transmit frames within timeouts defined by the AP to avoid being disassociated
  - Typically APs set those timeouts to be one or a couple of minutes long, thus limiting how long clients can sleep
  - BSS Max Idle feature allows clients to request a longer sleep period from AP
  - Allows clients to remain associated for up to 18 hours
  - Avoids the need for reassociation, which is highly costly energy-wise
  - Enables higher energy savings

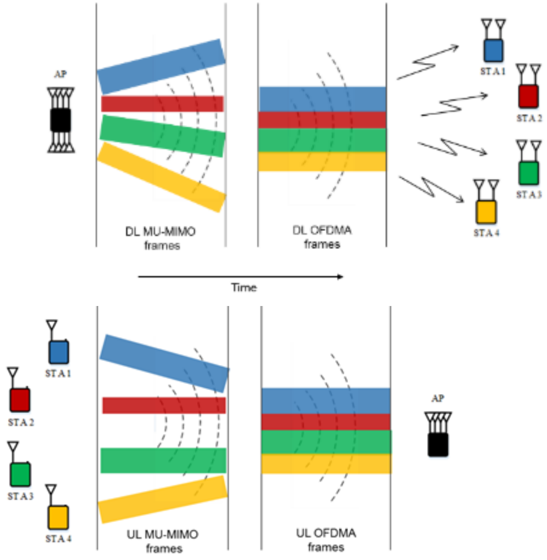
©2023 Silicon Laboratories Inc. All rights reserved.

Other Wi-Fi 6 features that help current consumption

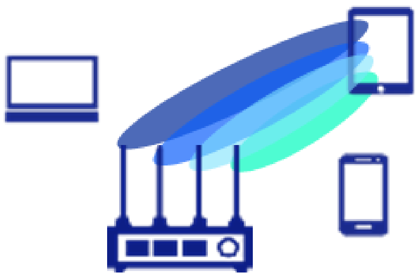
OFDMA



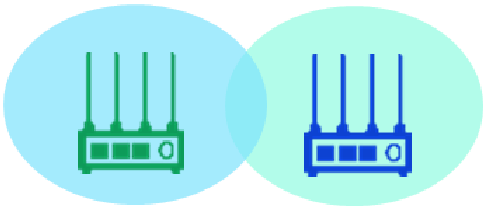
MU-MIMO



BEAMFORMING



BSS COLORING



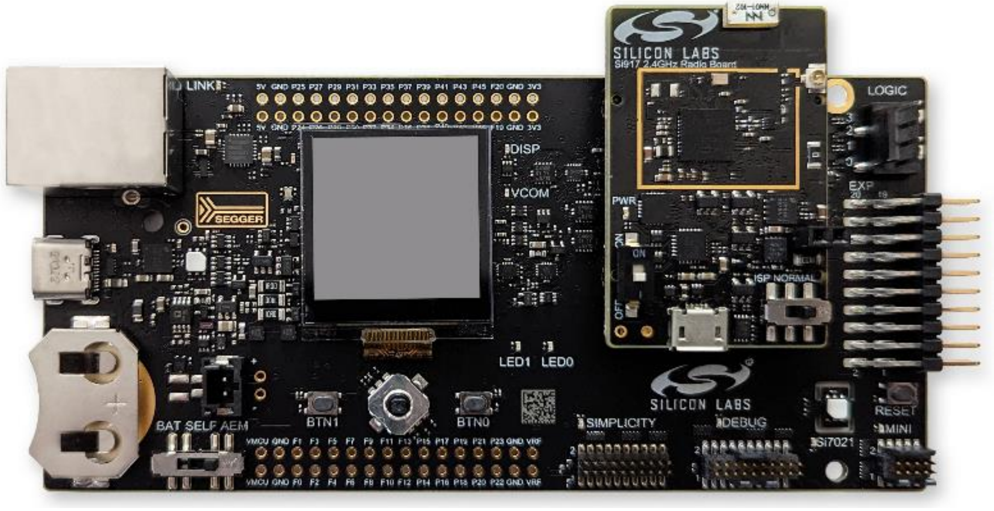
- **Wi-Fi 6 has multiple features that help alleviate congestion**
  - OFDMA
  - Beamforming and MU-MIMO
  - BSS Coloring
- **OFDMA**
  - Allows for spectral reuse through frequency multiplexing
- **Beamforming and MU-MIMO**
  - Allow for spectral reuse through spatial multiplexing
- **BSS Coloring**
  - Allow devices (APs and stations) to differentiate packets transmitted by its network from packets transmitted by other networks in the same channel
- **By alleviating congestion these features allow devices to stay on the air smaller amounts of time and thus, reduce current consumption**

©2023 Silicon Laboratories Inc. All rights reserved.

works with
BY SILICON LABS

SiWx917 Development Kit Overview

SiWx917 SoC Development Kit



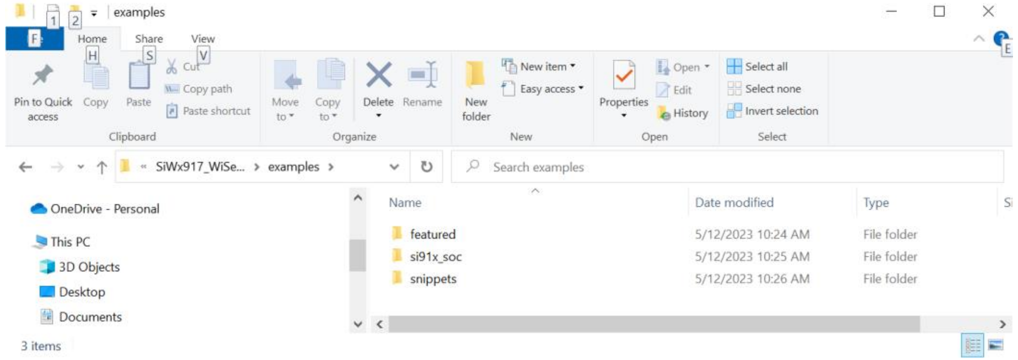
SiWx917 SoC Pro Kit

- **SiWx917 Pro kit for use in SoC mode**
  - SiWx917 Radio Board
  - Pro Kit Main Board

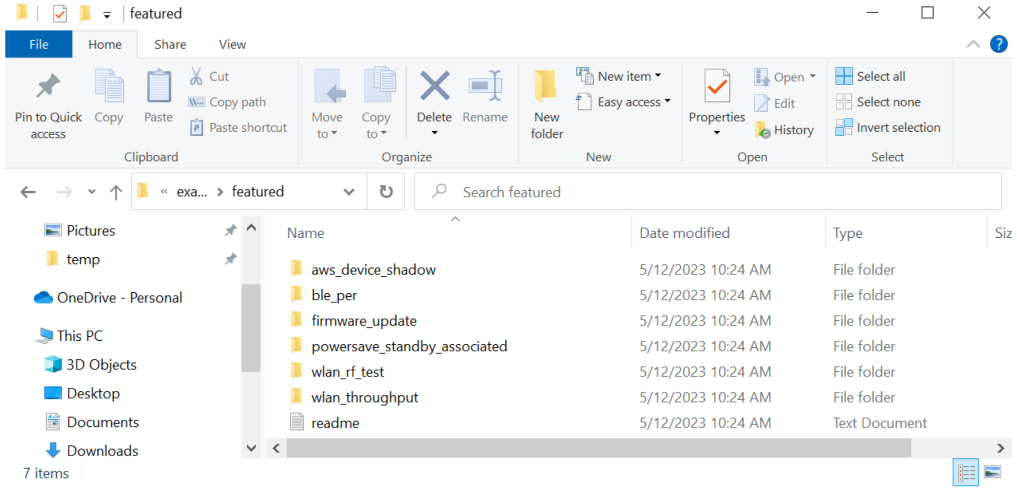
©2023 Silicon Laboratories Inc. All rights reserved.

SiWx917 Code Examples Available in SDK

SiWx917 Code Examples Available in SDK



Name	Date modified	Type
featured	5/12/2023 10:24 AM	File folder
si91x_soc	5/12/2023 10:25 AM	File folder
snippets	5/12/2023 10:26 AM	File folder

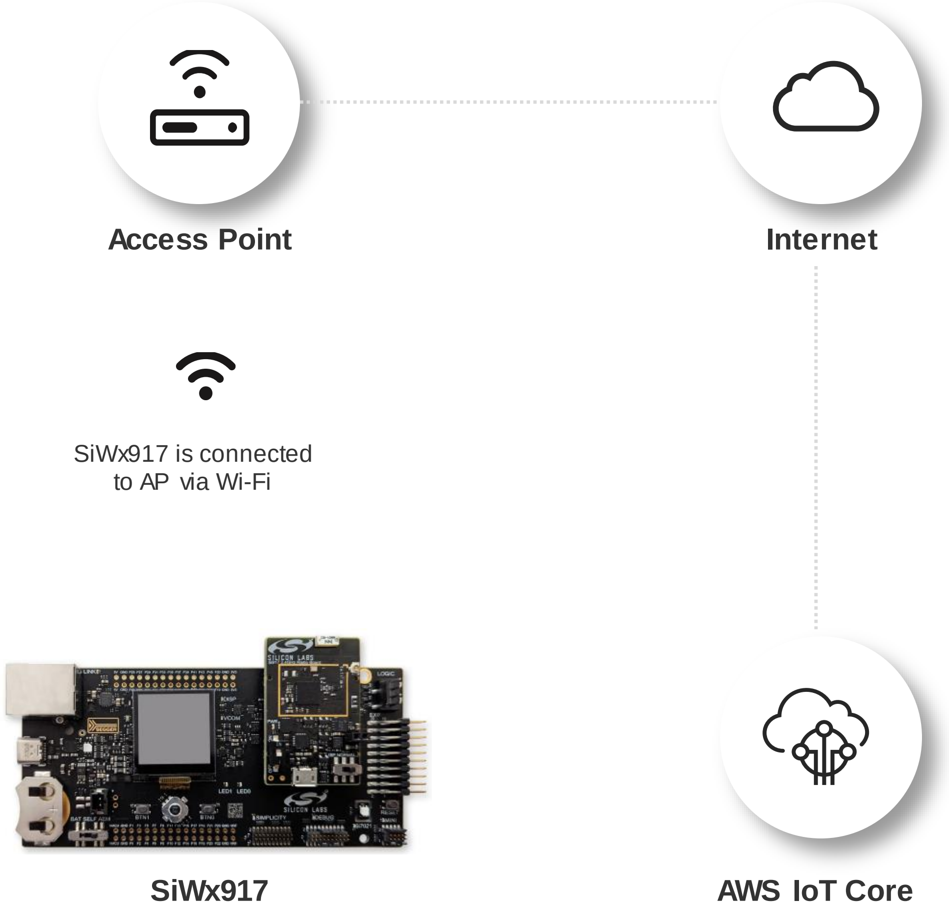


Name	Date modified	Type
aws_device_shadow	5/12/2023 10:24 AM	File folder
ble_per	5/12/2023 10:24 AM	File folder
firmware_update	5/12/2023 10:24 AM	File folder
powersave_standby_associated	5/12/2023 10:24 AM	File folder
wlan_rf_test	5/12/2023 10:24 AM	File folder
wlan_throughput	5/12/2023 10:24 AM	File folder
readme	5/12/2023 10:24 AM	Text Document

- **In order to see the code examples included in the release, open the examples subdirectory included in it to see the directory structure shown here.**
- **Within this directory, the “featured” subdirectory should be your main stop, as this subdirectory includes fully fleshed out code examples that can be used as references for your code development. Its contents look as shown here**
- **We’ll describe the included examples in the next slides.**

©2023 Silicon Laboratories Inc. All rights reserved.

AWS Device Shadow Code Example



Access Point

Internet

SiWx917 is connected to AP via Wi-Fi

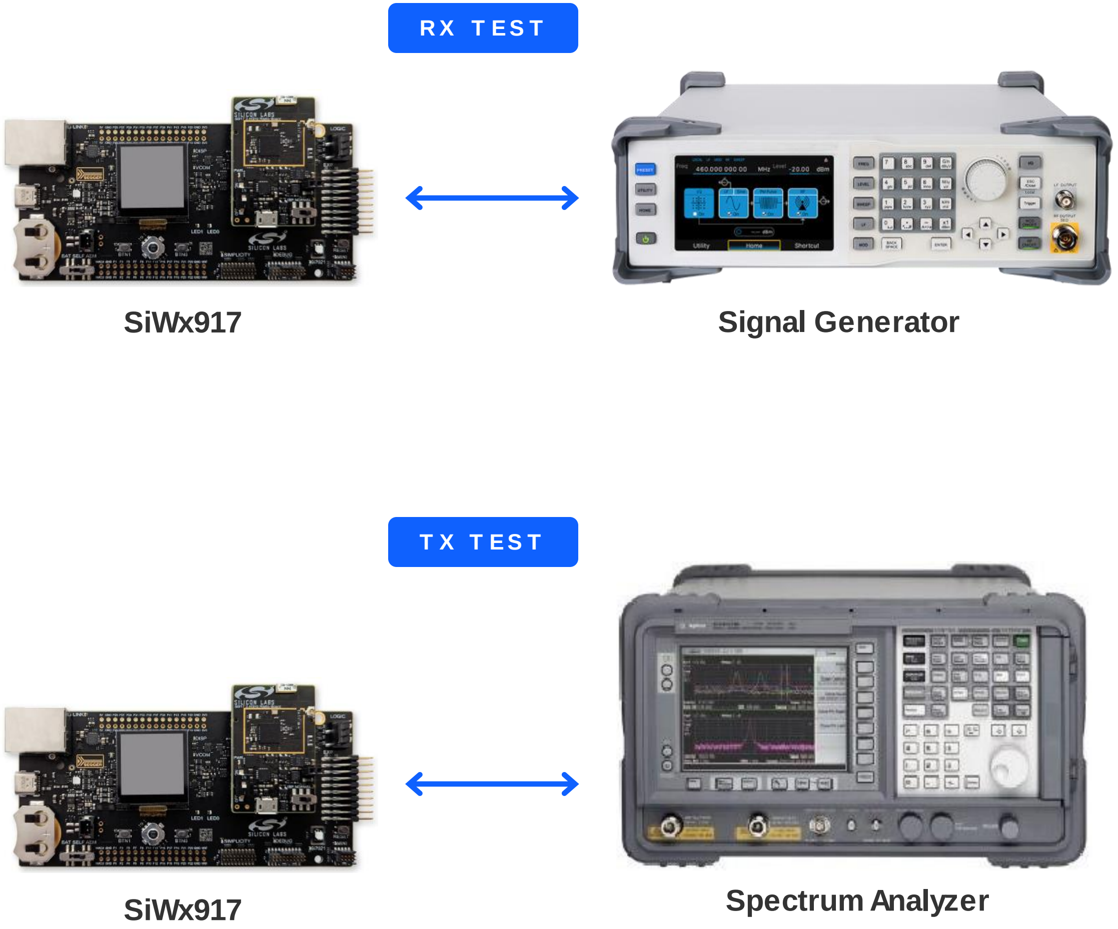
SiWx917

AWS IoT Core

- This example demonstrates how to connect an SiWx917 device securely to the AWS IoT Core to send and receive data
- This example creates a AWS Device Shadow on the SiWx917
- This provides a persistent virtual representation of the device that can be accessed even when the SiWx917 is offline
- The created AWS Device Shadow provides the following information:
  - Room temperature
  - Window open/close status
- This code example can be used as a reference to create product code to report different sets of information to AWS
- In order to use this application successfully, we recommend that you familiarize yourself with the following:
  - The basics of AWS IoT Core operation:
  - The following are good references for this purpose:
    - https://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html
    - https://docs.aws.amazon.com/iot/latest/developerguide/iot-tutorials.html

©2023 Silicon Laboratories Inc. All rights reserved.

BLE PER Code Example



RX TEST

SiWx917

Signal Generator

TX TEST

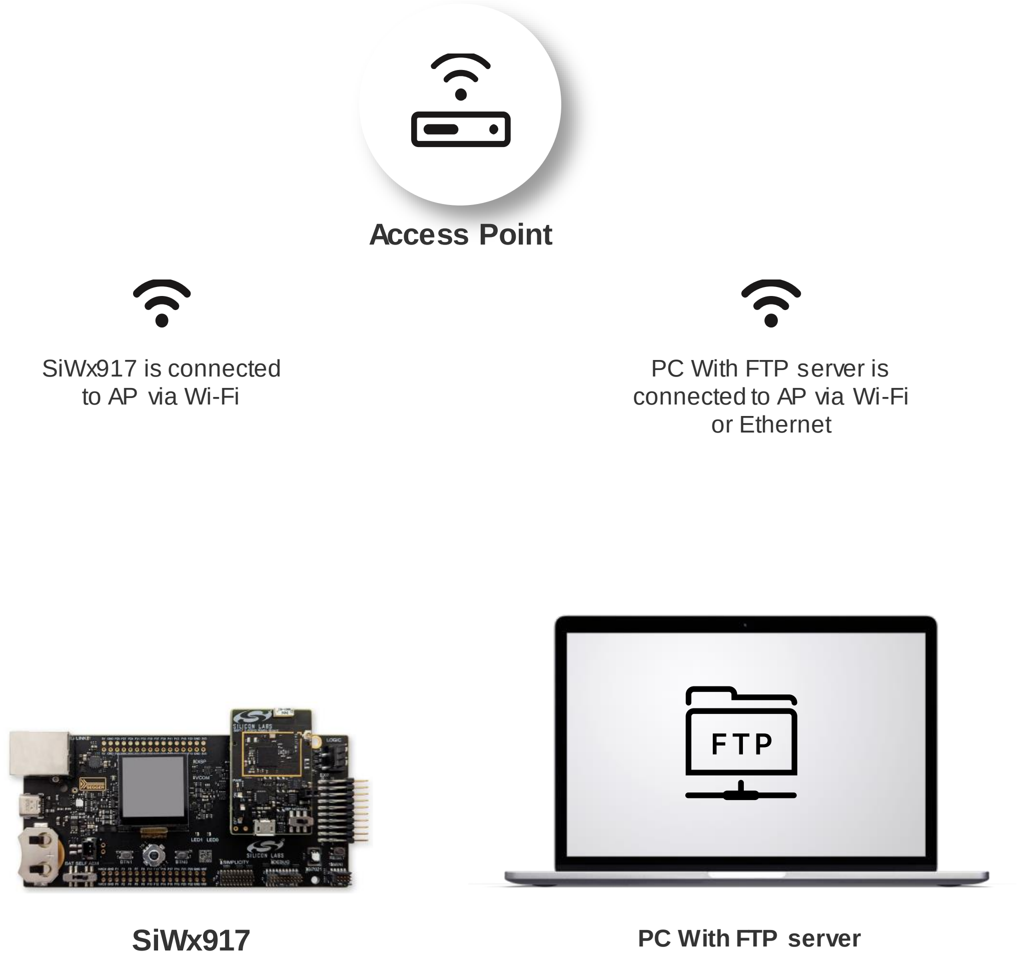
SiWx917

Spectrum Analyzer

- **This example demonstrates how to configure an SiWx917 to perform the transmission or reception of BLE Packets to be used for the following purposes:**
  - PER measurement
  - RF measurement
- **Code allows to configure SiWx917 in the following modes:**
  - BLE PER Transmit mode
  - BLE PER Receive mode
- **It allows for the payload to be configured as any of the following:**
  - PRBS9
  - PRBS15
  - Four ones + four zeros alternating data
  - Four zeros+ four ones alternating data
  - Alternating ones and zeros data
  - Alternating zeros and ones data
  - All ones data
  - All zeros data
- **The code also allows for the configuration of the following**
  - Packet length
  - Data rate (2 mbps, 1 mbps, 500 kbps or 125 kbps)
  - Continuous mode or burst mode operation
  - Hopping configuration (No hopping, fixed hopping or random hopping)
  - Antenna selection (Onboard or external)
  - External or internal BLE RF
  - Loopback or non-loopback operation

©2023 Silicon Laboratories Inc. All rights reserved.

Firmware Update Code Example



Access Point

SiWx917 is connected to AP via Wi-Fi

PC With FTP server is connected to AP via Wi-Fi or Ethernet

SiWx917

PC With FTP server

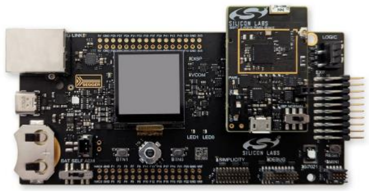
- **This code example shows how to update SiWx917 firmware over the air**
- **The code instructs the SiWx917 to retrieve a firmware image from an FTP server**
- **The FTP server should be located on a PC on the local WLAN/LAN**
- **It is also possible to update from a remote cloud server (AWS, Microsoft Azure for example), but this is not shown by this code example**
- **The demo code will perform the following steps:**
  - SiWx917 connects to FTP server
  - SiWx917's OTA application sends firmware file request to FTP server
  - Server replies to SiWx917 with firmware file
  - SIX917's OTA application programs firmware into flash memory and reboots SiWx917

©2023 Silicon Laboratories Inc. All rights reserved.

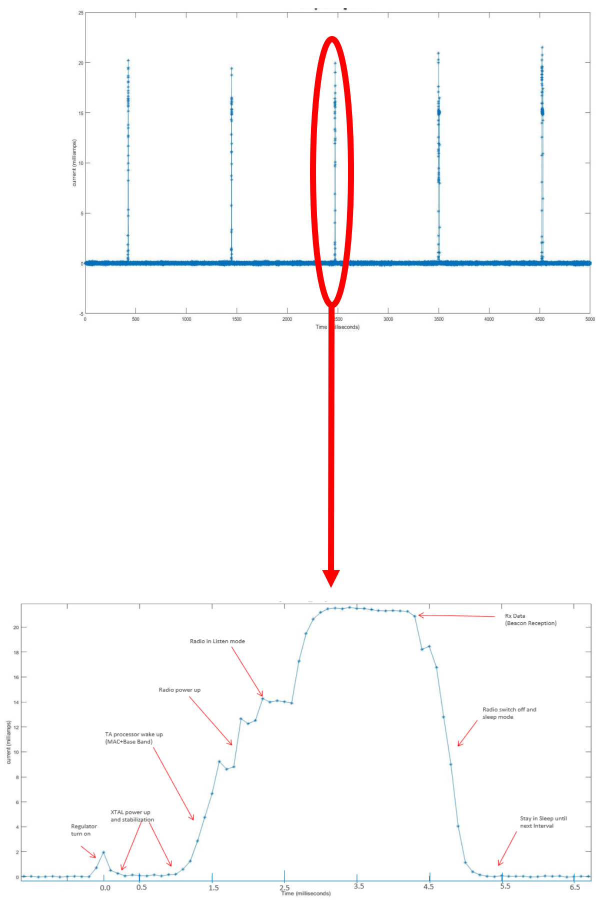
Powersave Standby Associated Code Example

Access Point

SiWx917 is connected
to AP via Wi-Fi



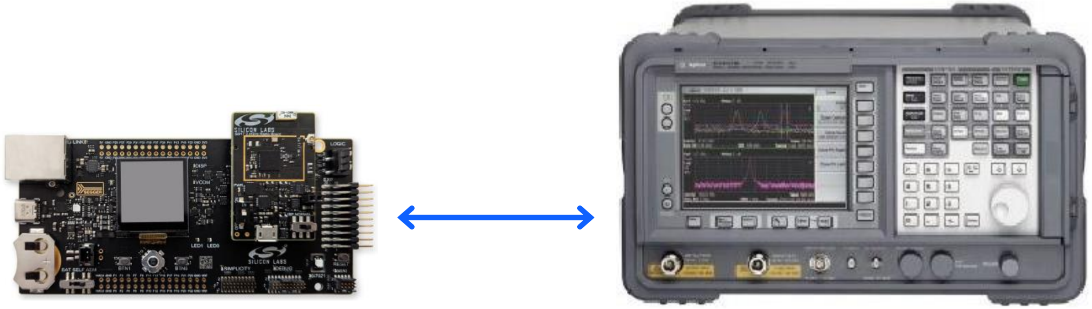
SiWx917



- **This code example shows how to configure SiWx917 to do the following:**
  - Configure the SiWx917 into station mode
  - Associate to an Access Point
  - Obtain an IP address from that access point
  - Set the SiWx917 to standby associated mode
  - Wake up the SiWx917 from standby mode to wake up to listen to beacons with either of the following:
    - DTIM wakeups
    - Listen interval wakeups
- **After executing the above, the code has the option to enable UDP transfer**
- **If this is enabled, SiWx917 will transmit UDP data to a server specified in the code**

©2023 Silicon Laboratories Inc. All rights reserved.

WLAN RF Test Code Example



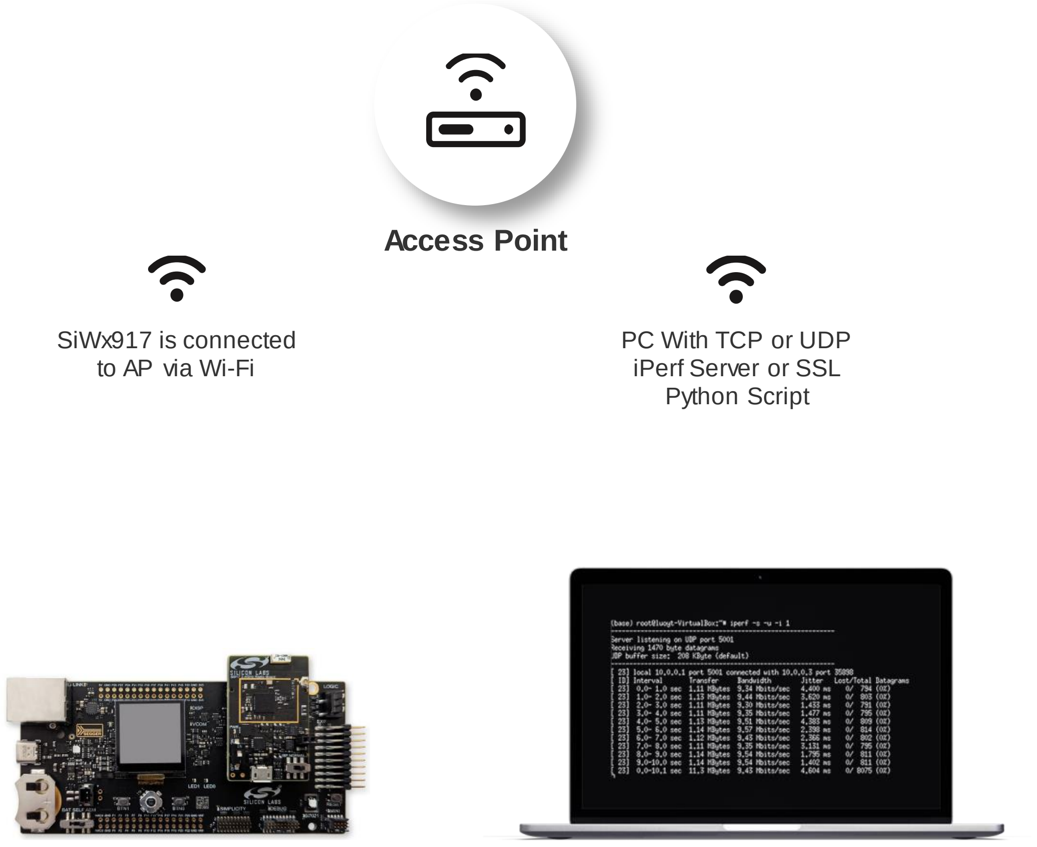
SiWx917

Spectrum Analyzer

- **This code example demonstrates how to configure an SiWx917 to transmit Wi-Fi data packets**
- **Its purpose is to be used for regulatory certification testing**
- **It allows for the configuration of the following:**
  - Transmit power
  - Transmit data rate
  - Burst or continuous transmit mode
  - Transmit channel
  - Internal or external antenna
  - Antenna gain
  - Number of packets to be transmitted

©2023 Silicon Laboratories Inc. All rights reserved.

WLAN Throughput Code Example



Access Point

SiWx917 is connected
to AP via Wi-Fi

PC With TCP or UDP
iPerf Server or SSL
Python Script

```
(base) root@luoyt-VirtualBox:~# iperf -s -u -i 1
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[ 23] local 10.0.0.1 port 5001 connected with 10.0.0.3 port 35898
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[ 23]  0.0- 1.0 sec  1.11 MBytes  9.34 Mbits/sec   4.400 ms    0/  794 (0%)
[ 23]  1.0- 2.0 sec  1.13 MBytes  9.44 Mbits/sec   3.620 ms    0/  803 (0%)
[ 23]  2.0- 3.0 sec  1.11 MBytes  9.30 Mbits/sec   1.433 ms    0/  791 (0%)
[ 23]  3.0- 4.0 sec  1.11 MBytes  9.35 Mbits/sec   1.477 ms    0/  795 (0%)
[ 23]  4.0- 5.0 sec  1.13 MBytes  9.51 Mbits/sec   4.383 ms    0/  809 (0%)
[ 23]  5.0- 6.0 sec  1.14 MBytes  9.57 Mbits/sec   2.398 ms    0/  814 (0%)
[ 23]  6.0- 7.0 sec  1.12 MBytes  9.43 Mbits/sec   2.366 ms    0/  802 (0%)
[ 23]  7.0- 8.0 sec  1.11 MBytes  9.35 Mbits/sec   3.131 ms    0/  795 (0%)
[ 23]  8.0- 9.0 sec  1.14 MBytes  9.54 Mbits/sec   1.795 ms    0/  811 (0%)
[ 23]  9.0-10.0 sec  1.14 MBytes  9.54 Mbits/sec   1.402 ms    0/  811 (0%)
[ 23]  0.0-10.1 sec  11.3 MBytes  9.43 Mbits/sec   4.604 ms    0/ 8075 (0%)
```

SiWx917

PC With TCP or UDP
iPerf Server or
SSL Python Script

- **This code example allows for testing the throughput of an SiWx917 system for the following:**
  - TCP uplink and downlink
  - UDP uplink and downlink
  - SSL uplink and downlink
- **The code configures SiWx917 to perform the following**
  - Associate to access point
  - Obtain IP address from access point
  - Connect to relevant server on PC running either of the following:
    - iPerf server or client for TCP or UDP tests,
    - Python-based SSL scripts for SSL tests
- **Provides measurement of obtained throughput for specified test**
- **Code allows for the configuration of the following**
  - Selection of desired test (TCP, UDP or SSL)
  - Local port number
  - Server port number
  - Server IP address

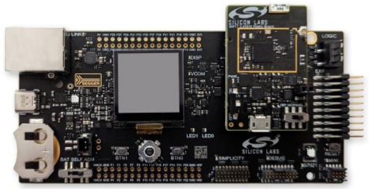
©2023 Silicon Laboratories Inc. All rights reserved.

Getting Deep into a Code Example: Wi-Fi-Only Standby Associate (DTIM/Listen Interval)

What does the Wi-Fi-Only Standby Associate Code Example do?

Access Point

SiWx917 is connected
to AP via Wi-Fi

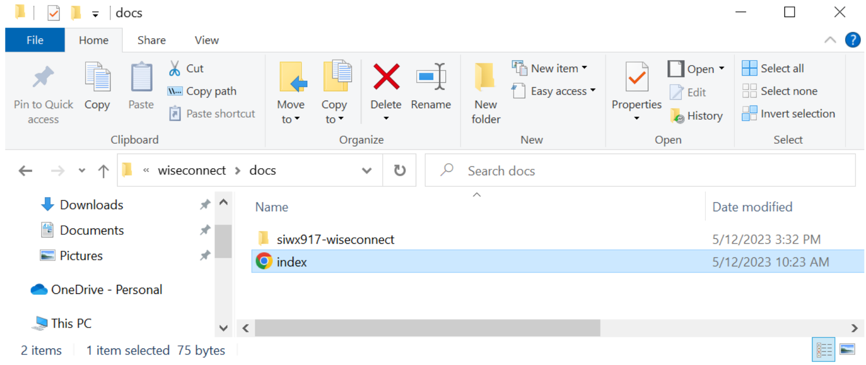


SiWx917

- **As stated before, this code example shows how to configure SiWx917 to do the following:**
  - Configure the SiWx917 into station mode
  - Associate to an Access Point
  - Obtain an IP address from that access point
  - Set the SiWx917 to standby associated mode
  - Wake up the SiWx917 from standby mode to wake up to listen to beacons with either of the following:
    - DTIM wakeups
    - Listen interval wakeups
- **The following slides will show us how this is done through SAPIs function calls**

©2023 Silicon Laboratories Inc. All rights reserved.

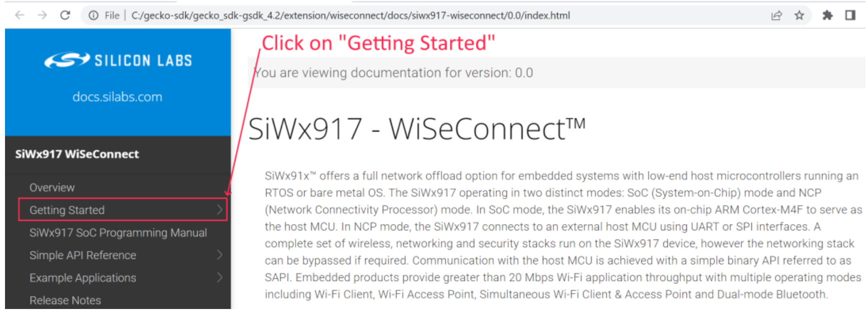
Let’s open this application up



« wiseconnect › docs

Name	Date modified
siwx917-wiseconnect	5/12/2023 3:32 PM
index	5/12/2023 10:23 AM

2 items   1 item selected 75 bytes


File | C:/gecko-sdk/gecko_sdk-gsdk_4.2/extension/wiseconnect/docs/siwx917-wiseconnect/0.0/index.html

Click on "Getting Started"

You are viewing documentation for version: 0.0

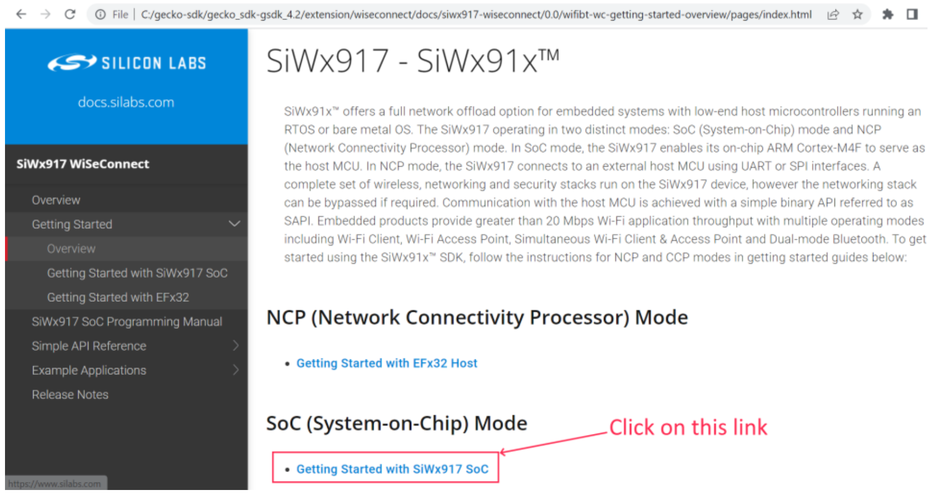
SiWx917 - WiSeConnect™

SiWx91x™ offers a full network offload option for embedded systems with low-end host microcontrollers running an RTOS or bare metal OS. The SiWx917 operating in two distinct modes: SoC (System-on-Chip) mode and NCP (Network Connectivity Processor) mode. In SoC mode, the SiWx917 enables its on-chip ARM Cortex-M4F to serve as the host MCU. In NCP mode, the SiWx917 connects to an external host MCU using UART or SPI interfaces. A complete set of wireless, networking and security stacks run on the SiWx917 device, however the networking stack can be bypassed if required. Communication with the host MCU is achieved with a simple binary API referred to as SAPI. Embedded products provide greater than 20 Mbps Wi-Fi application throughput with multiple operating modes including Wi-Fi Client, Wi-Fi Access Point, Simultaneous Wi-Fi Client & Access Point and Dual-mode Bluetooth.

- **In order to see this code example, you should import its project into Simplicity Studio.**
- **To do so, follow the instructions given in the SiWx917 WiseConnect guide document included with the release**
- **First open the “index.html” document included in the docs folder of the release as shown here**
- **Then click on the “Getting Started” link on the left side of the webpage**

©2023 Silicon Laboratories Inc. All rights reserved.

Let’s open this application up



File | C:/gecko-sdk/gecko_sdk-gsdk_4.2/extension/wiseconnect/docs/siwx917-wiseconnect/0.0/wifibt-wc-getting-started-overview/pages/index.html

SILICON LABS

docs.silabs.com

SiWx917 WiSeConnect

- Overview
- Getting Started
  - Overview
  - Getting Started with SiWx917 SoC
  - Getting Started with EFx32
- SiWx917 SoC Programming Manual
- Simple API Reference
- Example Applications
- Release Notes

https://www.silabs.com

SiWx917 - SiWx91x™

SiWx91x™ offers a full network offload option for embedded systems with low-end host microcontrollers running an RTOS or bare metal OS. The SiWx917 operating in two distinct modes: SoC (System-on-Chip) mode and NCP (Network Connectivity Processor) mode. In SoC mode, the SiWx917 enables its on-chip ARM Cortex-M4F to serve as the host MCU. In NCP mode, the SiWx917 connects to an external host MCU using UART or SPI interfaces. A complete set of wireless, networking and security stacks run on the SiWx917 device, however the networking stack can be bypassed if required. Communication with the host MCU is achieved with a simple binary API referred to as SAPI. Embedded products provide greater than 20 Mbps Wi-Fi application throughput with multiple operating modes including Wi-Fi Client, Wi-Fi Access Point, Simultaneous Wi-Fi Client & Access Point and Dual-mode Bluetooth. To get started using the SiWx91x™ SDK, follow the instructions for NCP and CCP modes in getting started guides below:

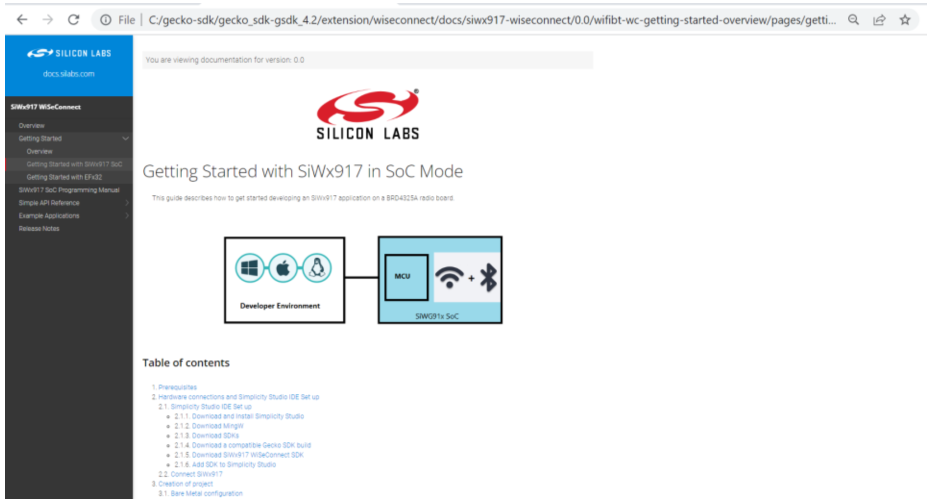
NCP (Network Connectivity Processor) Mode

- Getting Started with EFx32 Host

SoC (System-on-Chip) Mode

- Getting Started with SiWx917 SoC

Click on this link



File | C:/gecko-sdk/gecko_sdk-gsdk_4.2/extension/wiseconnect/docs/siwx917-wiseconnect/0.0/wifibt-wc-getting-started-overview/pages/getti...

SILICON LABS

docs.silabs.com

SiWx917 WiSeConnect

- Overview
- Getting Started
  - Overview
  - Getting Started with SiWx917 SoC
  - Getting Started with EFx32
- SiWx917 SoC Programming Manual
- Simple API Reference
- Example Applications
- Release Notes

You are viewing documentation for version: 0.0

SILICON LABS

Getting Started with SiWx917 in SoC Mode

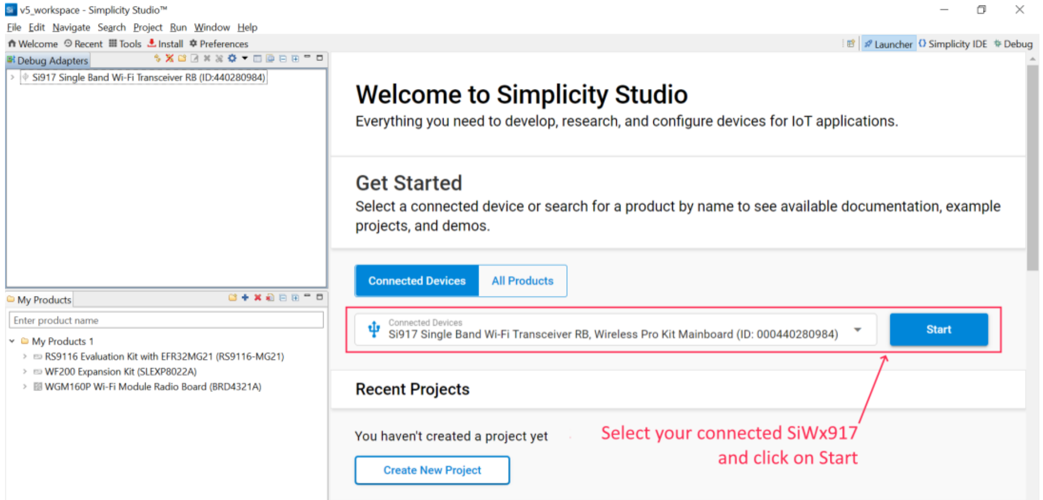
This guide describes how to get started developing an SiWx917 application on a BRD4325A radio board.

Table of contents

- **Then click on the “Getting Started with SiWx917 SoC” link**
- **Follow the instructions on the following sections of the document on this page:**
  - Make sure that you meet the prerequisites in section 1 (Prerequisites)
  - Set up Simplicity Studio as stated in section 2.1 (Simplicity Studio IDE Set Up)
  - Connect your SiWx917 development board to your PC as stated in section 2.2 (Connect SiWx917)

©2023 Silicon Laboratories Inc. All rights reserved.

Let’s open this application up


v5_workspace - Simplicity Studio™
File Edit Navigate Search Project Run Window Help
Welcome Recent Tools Install Preferences
Launcher Simplicity IDE Debug
Debug Adapters
My Products
Connected Devices
All Products
Start
Create New Project

Si917 Single Band Wi-Fi Transceiver RB (ID:440280984)

Enter product name

My Products 1
- RS9116 Evaluation Kit with EFR32MG21 (RS9116-MG21)
- WF200 Expansion Kit (SLEXP8022A)
- WGM160P Wi-Fi Module Radio Board (BRD4321A)

Welcome to Simplicity Studio

Everything you need to develop, research, and configure devices for IoT applications.

Get Started

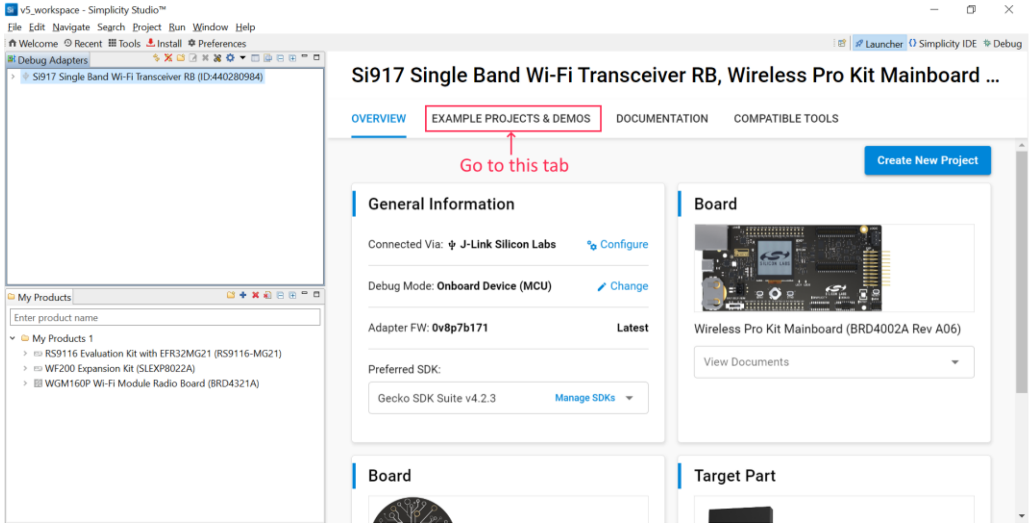
Select a connected device or search for a product by name to see available documentation, example projects, and demos.

Connected Devices: Si917 Single Band Wi-Fi Transceiver RB, Wireless Pro Kit Mainboard (ID: 000440280984)

Recent Projects

You haven't created a project yet

Select your connected SiWx917 and click on Start


v5_workspace - Simplicity Studio™
File Edit Navigate Search Project Run Window Help
Welcome Recent Tools Install Preferences
Launcher Simplicity IDE Debug
Debug Adapters
My Products
OVERVIEW
EXAMPLE PROJECTS & DEMOS
DOCUMENTATION
COMPATIBLE TOOLS
Create New Project
Configure
Change
Manage SDKs
View Documents

Si917 Single Band Wi-Fi Transceiver RB (ID:440280984)

Enter product name

My Products 1
- RS9116 Evaluation Kit with EFR32MG21 (RS9116-MG21)
- WF200 Expansion Kit (SLEXP8022A)
- WGM160P Wi-Fi Module Radio Board (BRD4321A)

Si917 Single Band Wi-Fi Transceiver RB, Wireless Pro Kit Mainboard ...

Go to this tab

General Information

Connected Via: J-Link Silicon Labs

Debug Mode: Onboard Device (MCU)

Adapter FW: 0v8p7b171 Latest

Preferred SDK: Gecko SDK Suite v4.2.3

Board

Wireless Pro Kit Mainboard (BRD4002A Rev A06)

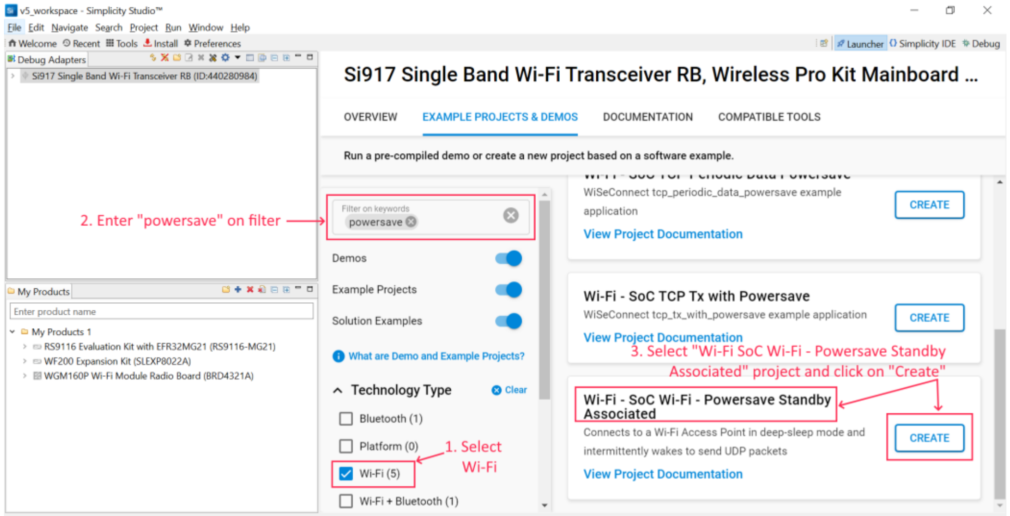
Board

Target Part

- Follow the instructions on section 3 of the document (Creation of Project) to import the project onto Simplicity Studio. To do so, do the following:
- Select your SiWx917 and click on Start.
- Go to the “Example Projects & Demos” tab

©2023 Silicon Laboratories Inc. All rights reserved.

Let’s open this application up



v5_workspace - Simplicity Studio™

File Edit Navigate Search Project Run Window Help
Welcome Recent Tools Install Preferences
Launcher Simplicity IDE Debug
Debug Adapters
Si917 Single Band Wi-Fi Transceiver RB (ID:440280984)
My Products
Enter product name
My Products 1
RS9116 Evaluation Kit with EFR32MG21 (RS9116-MG21)
WF200 Expansion Kit (SLEXP8022A)
WGM160P Wi-Fi Module Radio Board (BRD4321A)
OVERVIEW
EXAMPLE PROJECTS & DEMOS
DOCUMENTATION
COMPATIBLE TOOLS
Filter on keywords
Demos
Example Projects
Solution Examples
What are Demo and Example Projects?
Technology Type
Clear
CREATE
View Project Documentation

Si917 Single Band Wi-Fi Transceiver RB, Wireless Pro Kit Mainboard ...

Run a pre-compiled demo or create a new project based on a software example.

2. Enter "powersave" on filter

powersave

- ☐ Bluetooth (1)
- ☐ Platform (0)
- ☑ Wi-Fi (5)
- ☐ Wi-Fi + Bluetooth (1)

1. Select Wi-Fi

WiSeConnect tcp_periodic_data_powersave example application

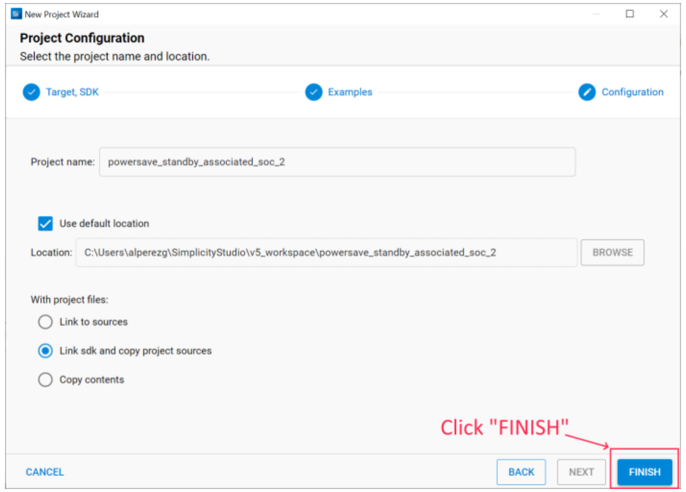
Wi-Fi - SoC TCP Tx with Powersave

WiSeConnect tcp_tx_with_powersave example application

3. Select "Wi-Fi SoC Wi-Fi - Powersave Standby Associated" project and click on "Create"

Wi-Fi - SoC Wi-Fi - Powersave Standby Associated

Connects to a Wi-Fi Access Point in deep-sleep mode and intermittently wakes to send UDP packets



New Project Wizard

Project Configuration

Select the project name and location.

Project name: powersave_standby_associated_soc_2

☑ Use default location

Location: C:\Users\alperezg\SimplicityStudio\v5_workspace\powersave_standby_associated_soc_2

With project files:

○ Link to sources

● Link sdk and copy project sources

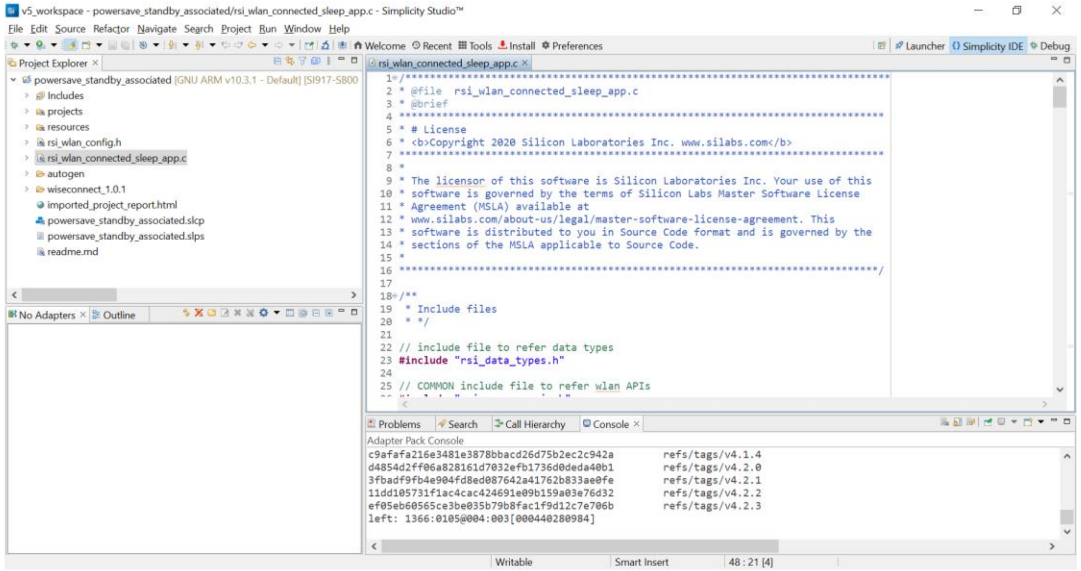
○ Copy contents

Click "FINISH"

- Select “Wi-Fi” under technology type
- Enter “powersave” in “Filter on keywords”
- Scroll down to find “Wi-Fi – SoC Wi-Fi – Powersave Standby Associated” project
- Click on “Create”
- Select your SiWx917 and click on Start.
- On the screen that will pop-up click “Finish”. This will now import the project

©2023 Silicon Laboratories Inc. All rights reserved.

Let’s open this application up

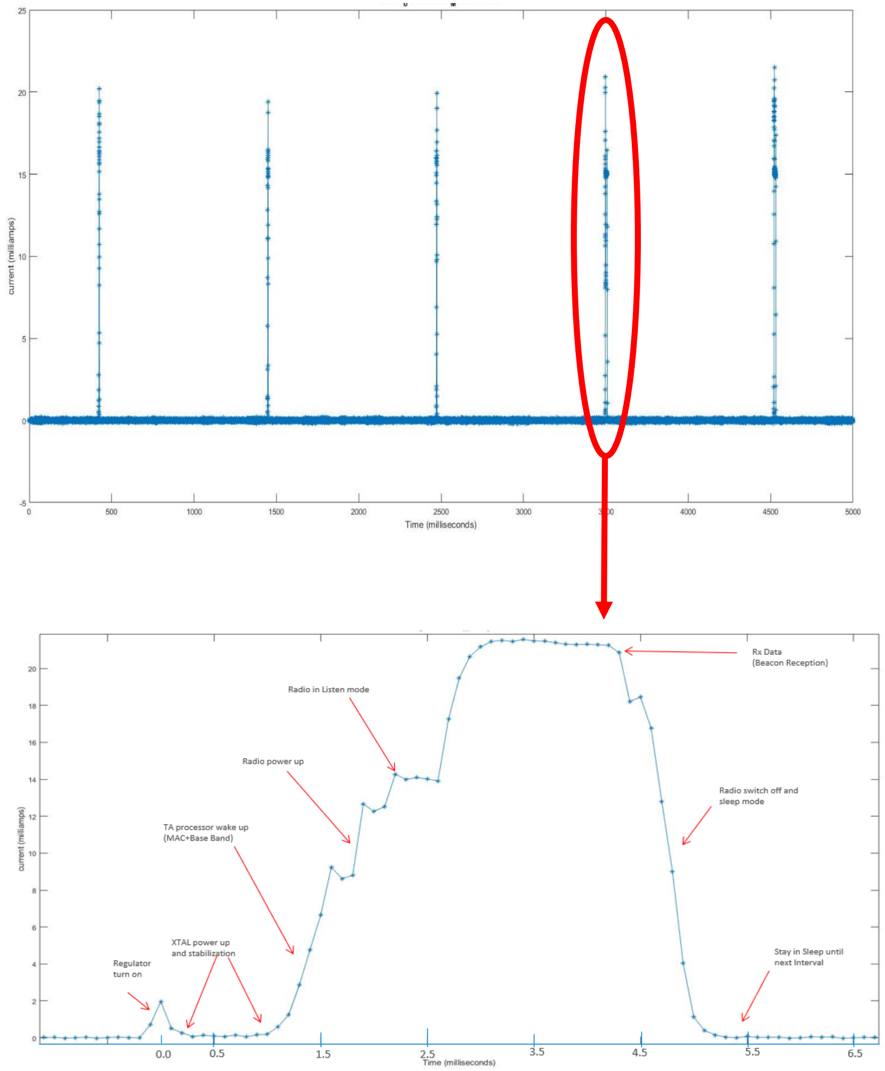

v5_workspace - powersave_standby_associated/rsi_wlan_connected_sleep_app.c - Simplicity Studio™
File Edit Source Refactor Navigate Search Project Run Window Help
Welcome Recent Tools Install Preferences
Launcher Simplicity IDE Debug
Project Explorer
powersave_standby_associated [GNU ARM v10.3.1 - Default] [SI917-SB00
Includes
projects
resources
rsi_wlan_config.h
rsi_wlan_connected_sleep_app.c
autogen
wiseconnect_1.0.1
imported_project_report.html
powersave_standby_associated.slcp
powersave_standby_associated.slps
readme.md
No Adapters
Outline
rsi_wlan_connected_sleep_app.c
Problems
Search
Call Hierarchy
Console
Adapter Pack Console
Writable
Smart Insert
48 : 21 [4]

```
/*******************************************************************************
* @file  rsi_wlan_connected_sleep_app.c
* @brief
*******************************************************************************
* # License
* <b>Copyright 2020 Silicon Laboratories Inc. www.silabs.com</b>
*******************************************************************************
*
* The licensor of this software is Silicon Laboratories Inc. Your use of this
* software is governed by the terms of Silicon Labs Master Software License
* Agreement (MSLA) available at
* www.silabs.com/about-us/legal/master-software-license-agreement. This
* software is distributed to you in Source Code format and is governed by the
* sections of the MSLA applicable to Source Code.
*
******************************************************************************/

/**
 * Include files
 * */

// include file to refer data types
#include "rsi_data_types.h"

// COMMON include file to refer wlan APIs
```

```
c9afafa216e3481e3878bbacd26d75b2ec2c942a        refs/tags/v4.1.4
d4854d2ff06a828161d7032efb1736d0deda40b1        refs/tags/v4.2.0
3fbadf9fb4e904fd8ed087642a41762b833ae0fe        refs/tags/v4.2.1
11dd105731f1ac4cac424691e09b159a03e76d32        refs/tags/v4.2.2
ef05eb60565ce3be035b79b8fac1f9d12c7e706b        refs/tags/v4.2.3
left: 1366:0105@004:003[000440280984]
```

- **After the project has been imported, open the following file as shown here.**
    - rsi_wlan_connected_sleep_app.c
- **This is the main code file for this application, we’ll talk now about what it does.**

Let’s explain what the application does

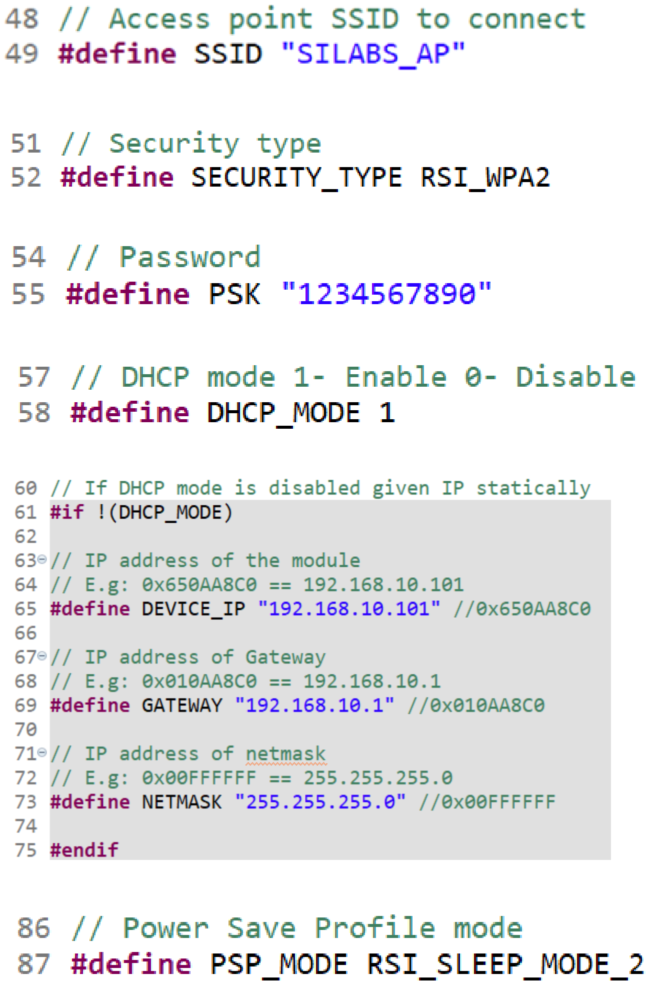


- **Let’s talk about what is it that this application does. It is the following:**
  - Application first initializes SiWx917 drivers
  - After this, SiWx917 device is initialized, valid firmware is loaded and module is powered up to run application.
  - SiWx917 operating mode is set to client mode and parameters are given to module including:
    - Security type (Open / WPA2-PSK, etc.)
    - TCP/IP feature bitmap
  - SiWx917 is provided SSID for network to associate (and, optionally, Wi-Fi channel to be used) and instructed to scan for this network.
  - If network is found, SiWx917 is provided WPA2 password (if needed) and is instructed to associate to access point.
  - After association is successful, SiWx917 is instructed to obtain IP address from AP through DHCP.
  - After obtaining IP address, SiWx917 is instructed to go to sleep

- **Sleep interval used by application to have SiW917 go to sleep is configured to use either listen interval or DTIM in the following way:**
  - If listen interval is used, this application configures SiWx917 to wake up every 1 second, listen for outstanding data from AP and go back to sleep
  - If DTIM is used, the application will have SiWx917 use DTIM period defined at AP to wake up

©2023 Silicon Laboratories Inc. All rights reserved.

What can we configure in this application?

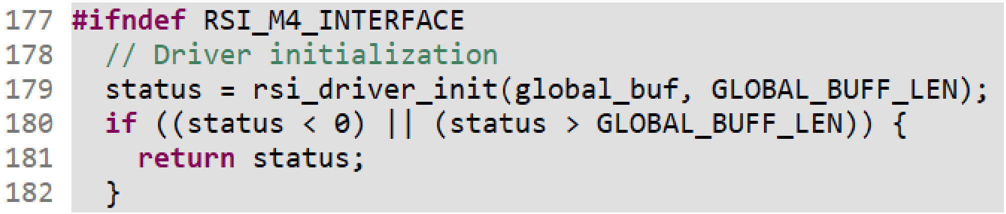


```
// Access point SSID to connect
#define SSID "SILABS_AP"

// Security type
#define SECURITY_TYPE RSI_WPA2

// Password
#define PSK "1234567890"

// DHCP mode 1- Enable 0- Disable
#define DHCP_MODE 1

// If DHCP mode is disabled given IP statically
#if !(DHCP_MODE)

// IP address of the module
// E.g: 0x650AA8C0 == 192.168.10.101
#define DEVICE_IP "192.168.10.101" //0x650AA8C0

// IP address of Gateway
// E.g: 0x010AA8C0 == 192.168.10.1
#define GATEWAY "192.168.10.1" //0x010AA8C0

// IP address of netmask
// E.g: 0x00FFFFFF == 255.255.255.0
#define NETMASK "255.255.255.0" //0x00FFFFFF

#endif

// Power Save Profile mode
#define PSP_MODE RSI_SLEEP_MODE_2
```

- **Let’s go over the variables that are configurable in this application.**
  - SSID for network to connect (#define SSID)
  - Security type (#define SECURITY_TYPE). Some valid options are:
    - RSI_OPEN = open security
    - RSI_WPA = WPA
    - RSI_WPA2 = WPA2
    - RSI_WPA3 = WPA3
    - RSI_WEP = WEP
    - RSI_WPA_EAP = Enterprise WPA EAP
    - RSI_WPA2_EAP = Enterprise WPA2 EAP
    - RSI_WPS_PI = WPS with PIN
  - Security Password (#define PSK)
  - Enabling or disabling DHCP (#define DHCP_MODE)
  - IP addressing for SiWx917 (If DHCP is disabled)
    - #define DEVICE_IP for SiWx917 IP address
    - #define GATEWAT for Gateway IP address
    - #define NETMASK for Netmask
  - Sleep mode to be used (#define PSP_MODE)

©2023 Silicon Laboratories Inc. All rights reserved.

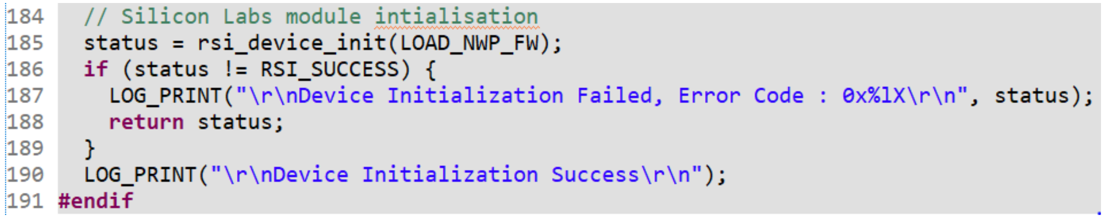
What is the code flow of this application?



```
#ifndef RSI_M4_INTERFACE
  // Driver initialization
  status = rsi_driver_init(global_buf, GLOBAL_BUFF_LEN);
  if ((status < 0) || (status > GLOBAL_BUFF_LEN)) {
    return status;
  }
```

- **Let’s now go over how the code of this application flows to do what it does**
- **Driver Initialization**
  - Is done by rsi_driver_init function
  - Initializes SiWx917 driver
  - Designates memory for all driver components
  - Initializes scheduler, events and queues needed by driver
  - Format:
    - int32_t rsi_driver_init(uint8_t * buffer, unit32_t length)

What is the code flow of this application?



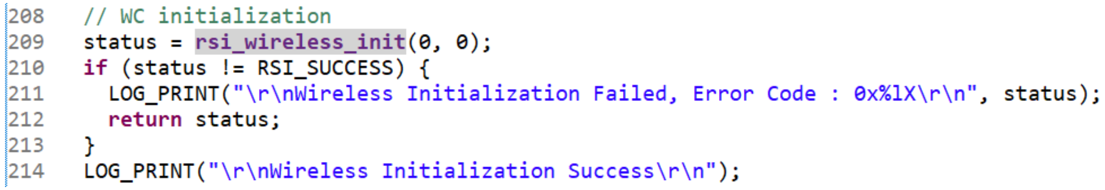
```
  // Silicon Labs module intialisation
  status = rsi_device_init(LOAD_NWP_FW);
  if (status != RSI_SUCCESS) {
    LOG_PRINT("\r\nDevice Initialization Failed, Error Code : 0x%lX\r\n", status);
    return status;
  }
  LOG_PRINT("\r\nDevice Initialization Success\r\n");
#endif
```

- **SiWx917 device initialization**
  - Is done by rsi_device_init
  - It initializes SiWx917 interface, AHB and bootloader
  - It sets the firmware image type to be loaded for the needed SiWx917 features
  - This is a blocking API
  - Format:
    - Int32_t rsi_device_init(unit8_t select option)

- **Where possible options are as follows:**
  - LOAD_NWP_FW: Will load firmware image
  - LOAD_DEFAULT_NWP_FW_ACTIVE_LOW: Will load active low firmware image
  - Active low firmware image will generate active low interrupts to indicate that packets are pending on module, instead of default active high

©2023 Silicon Laboratories Inc. All rights reserved.

What is the code flow of this application (Continued)?

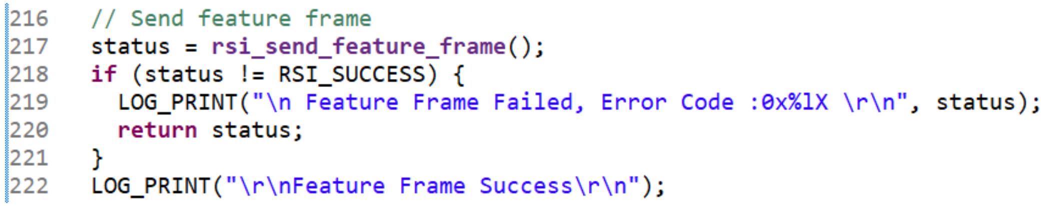


```
// WC initialization
status = rsi_wireless_init(0, 0);
if (status != RSI_SUCCESS) {
  LOG_PRINT("\r\nWireless Initialization Failed, Error Code : 0x%lX\r\n", status);
  return status;
}
LOG_PRINT("\r\nWireless Initialization Success\r\n");
```

- **Wireless Initialization**
  - Is done by rsi_wireless_init function
  - Sets WLAN/Coex operating mode
  - Initializes SiWx917 module features
  - Format:
    - Int32_t rsi_wireless_init(uint8_t opermode, uint16_t coex_mode)
  - So function is called with opermode = 0 coex_mode = 0
  - This initializes SiWx917 as a Wi-Fi client
  - This is a blocking API
  - opermode is defined as follows:
    - 0 – Client mode
    - 2 – Enterprise security client mode
    - 6 – Access point mode
    - 8 – Transmit test mode
    - 9 – Concurrent mode
  - coex_mode is defined as follows:
    - 0 – WLAN-only
    - 12 – BLE Mode
    - 13 – WLAN + BLE

©2023 Silicon Laboratories Inc. All rights reserved.

What is the code flow of this application (Continued)?

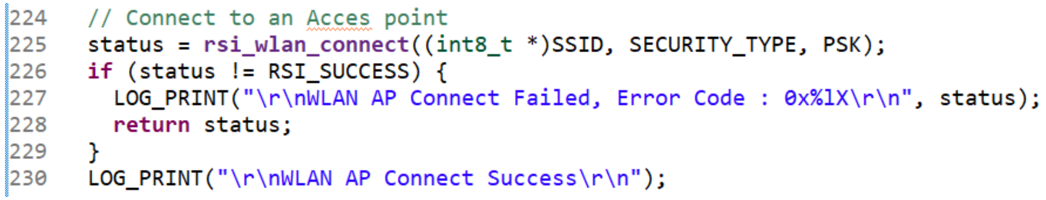


```
// Send feature frame
status = rsi_send_feature_frame();
if (status != RSI_SUCCESS) {
  LOG_PRINT("\n Feature Frame Failed, Error Code :0x%lX \r\n", status);
  return status;
}
LOG_PRINT("\r\nFeature Frame Success\r\n");
```

- **Send Feature Frame**
  - Is done by rsi_send_feature_frame
  - This API is used in power save mechanism
  - It is used for enabling/disabling LP chain, PPP and preamble duty cycle
  - It is a blocking API
  - Format:
    - int32_t rsi_send_feature_frame (void)

©2023 Silicon Laboratories Inc. All rights reserved.

What is the code flow of this application (Continued)?



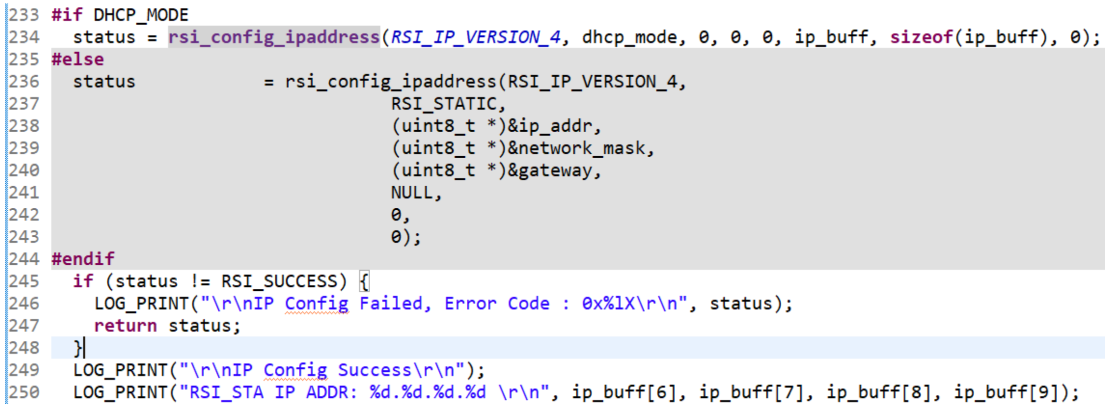
```
// Connect to an Acces point
status = rsi_wlan_connect((int8_t *)SSID, SECURITY_TYPE, PSK);
if (status != RSI_SUCCESS) {
  LOG_PRINT("\r\nWLAN AP Connect Failed, Error Code : 0x%lX\r\n", status);
  return status;
}
LOG_PRINT("\r\nWLAN AP Connect Success\r\n");
```

sec_type	Setting
0 (RSI_Open)	Open security
1 (RSI_WPA)	WPA
2 (RSI_WPA2)	WPA 2
3 (RSI_WEP)	WEP
4 (RSI_WPA_EAP)	WPA EAP enterprise
5 (RSI_WPA2_EAP)	WPA 2 EAP enterprise
6 (RSI_WPA_WPA2_MIXED)	WPA + WPA 2 mixed
7 (RSI_WPA_PMK)	WPA PMK
8 (RSI_WPA2_PMK)	WPA 2 PMK
9 (RSI_WPS_PIN)	WPS with PIN
10 (RSI_USE_GENERATED_WPSPIN)	Use generated WPS pin
11 (RSI_WPS_PUSH_BUTTON)	WPS with Pushbutton
12 (RSI_WPA_WPA2_MIXED_PMK)	WPA + WPA 2 mixed w/PMK
13 (RSI_WPA3)	WPA 3

- **Connect to Access Point**
  - Is done by rsi_wlan_connect
  - This function is used to scan for selected access point and associate to its network if scan finds it
  - This is a blocking API
  - Format:
    - int32_t rsi_wlan_connect(int8_t *ssid, rsi_security_mode_t sec_type, void *secret_key)
  - Where:
  - SSID = SSID of access point to connect
  - sec_type = Security type of access point to connect, options are as shown to the left:
  - secret_ley = Pointer to buffer containing security information based on sec_type

©2023 Silicon Laboratories Inc. All rights reserved.

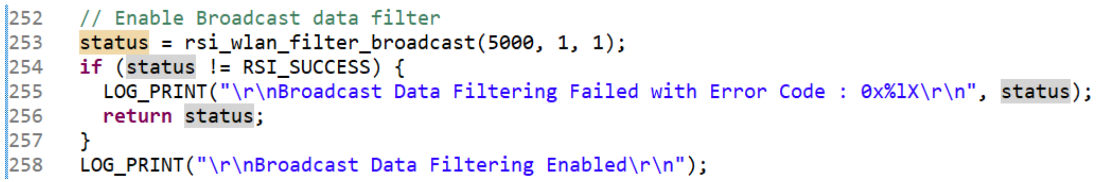
What is the code flow of this application (Continued)?



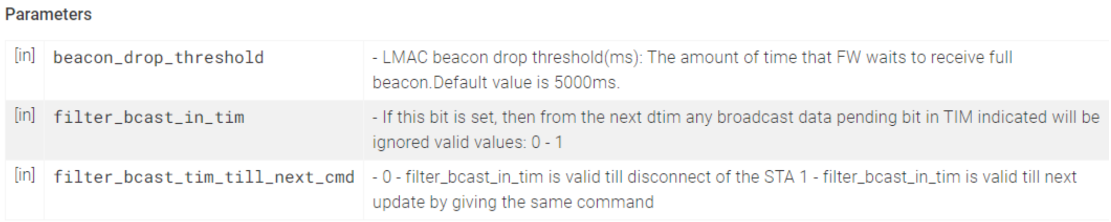
```
#if DHCP_MODE
  status = rsi_config_ipaddress(RSI_IP_VERSION_4, dhcp_mode, 0, 0, 0, ip_buff, sizeof(ip_buff), 0);
#else
  status            = rsi_config_ipaddress(RSI_IP_VERSION_4,
                                 RSI_STATIC,
                                 (uint8_t *)&ip_addr,
                                 (uint8_t *)&network_mask,
                                 (uint8_t *)&gateway,
                                 NULL,
                                 0,
                                 0);
#endif
  if (status != RSI_SUCCESS) {
    LOG_PRINT("\r\nIP Config Failed, Error Code : 0x%lX\r\n", status);
    return status;
  }
  LOG_PRINT("\r\nIP Config Success\r\n");
  LOG_PRINT("RSI_STA IP ADDR: %d.%d.%d.%d \r\n", ip_buff[6], ip_buff[7], ip_buff[8], ip_buff[9]);
```

- **Configure SiWx917’s IP address**
  - Is done by rsi_config_ipaddress
  - Performs IP address configuration of SiWx917
  - Uses provided parameters including:
    - IP Version
    - Static or Dynamic DHCP mode
  - Format:
    - Int32_t rsi_config_ipaddress(rsi_ip_version_t version, unit8_t * ip_addr, uint8_t * mask, uint8_t *gw, uint8_t *ipconfig_rsp, uint16_t length, uint8_t vap_id)
  - Version can be:
    - 4 (RSI_IP_VERSION_4): IP version 4
    - 6 (RSI_IP_VERSION_6): IP version 6
  - Mode can be:
    - 0: Static IP addressing
    - 1: DHCP
  - IP_addr is a pointer to the desired IP address (if using static addressing)
  - mask is a pointer to the desired network mask (if using static addressing)
  - gw is a pointer to the desired gateway IP address (if using static addressing)
  - ipconfig_rsp: holds the IP address obtained through DHCP
  - length: Length of the ipconfig_rsp_buffer
  - vap_id is a VAP ID used to differentiate between AP and station when SiWx917 is used in concurrent (AP + station mode). It is configurable as follows:
    - 0: For station
    - 1: For AP

What is the code flow of this application (Continued)?



```
// Enable Broadcast data filter
status = rsi_wlan_filter_broadcast(5000, 1, 1);
if (status != RSI_SUCCESS) {
  LOG_PRINT("\r\nBroadcast Data Filtering Failed with Error Code : 0x%lX\r\n", status);
  return status;
}
LOG_PRINT("\r\nBroadcast Data Filtering Enabled\r\n");
```



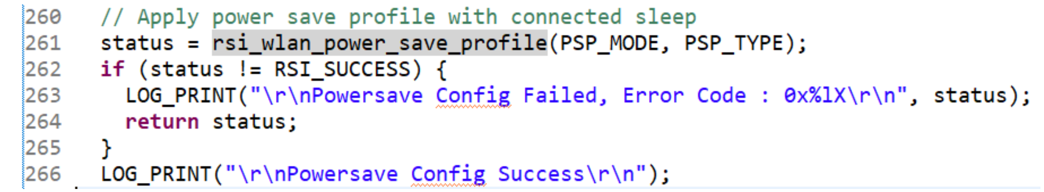
Parameters

[in]	beacon_drop_threshold	- LMAC beacon drop threshold(ms): The amount of time that FW waits to receive full beacon.Default value is 5000ms.
[in]	filter_bcast_in_tim	- If this bit is set, then from the next dtim any broadcast data pending bit in TIM indicated will be ignored valid values: 0 - 1
[in]	filter_bcast_tim_till_next_cmd	- 0 - filter_bcast_in_tim is valid till disconnect of the STA 1 - filter_bcast_in_tim is valid till next update by giving the same command

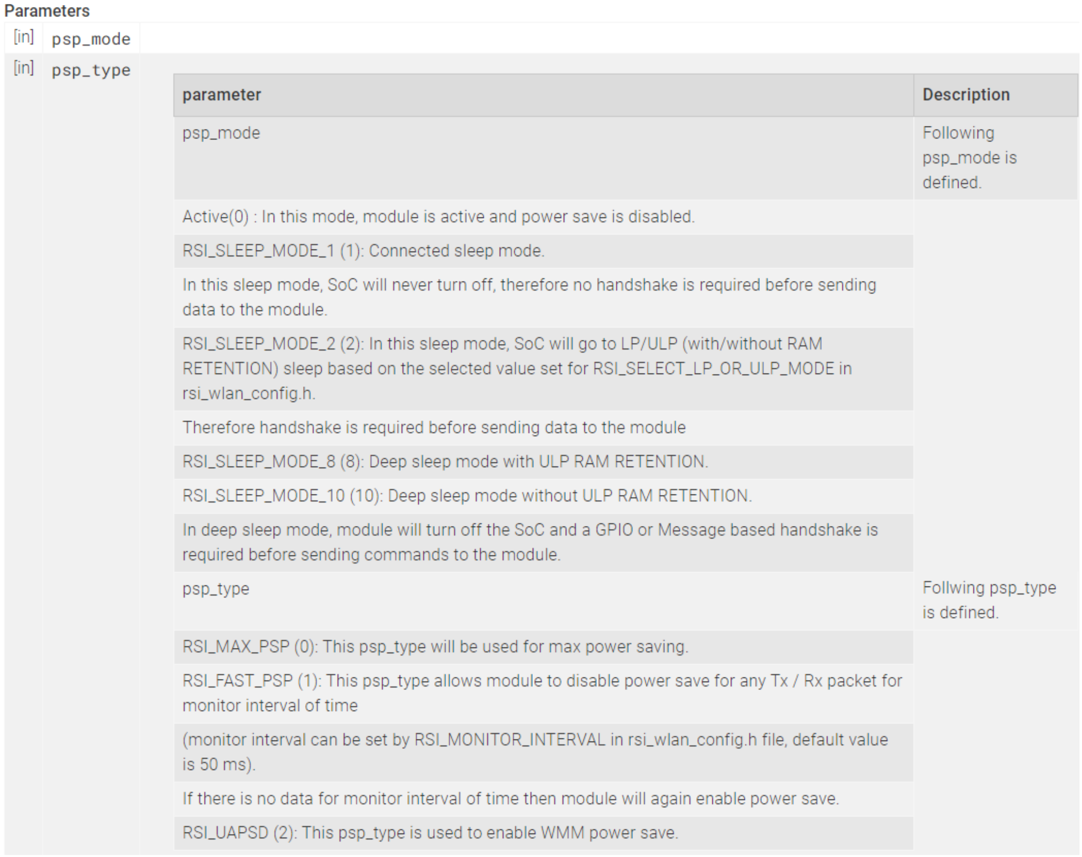
- **Enable Broadcast Data Filter**
  - Is done by rsi_wlan_filter_broadcast
  - This function is used to program the ignoring of broadcast packets as per defined threshold levels when SiWa917 is in power save mode
  - It is used to achieve low current consumption in standby associated mode
  - This is a blocking API
  - Format:
    - int32_t rsi_wlan_filter_broadcast (unit16_t beacon_drop_threshold, unit8_t filter_bcast_in_tim, unit8_t_filter_bcast_tim_till_next_cmd)
  - Description of parameters is as shown to the left

©2023 Silicon Laboratories Inc. All rights reserved.

What is the code flow of this application (Continued)?



```
// Apply power save profile with connected sleep
status = rsi_wlan_power_save_profile(PSP_MODE, PSP_TYPE);
if (status != RSI_SUCCESS) {
  LOG_PRINT("\r\nPowersave Config Failed, Error Code : 0x%lX\r\n", status);
  return status;
}
LOG_PRINT("\r\nPowersave Config Success\r\n");
```



Parameters

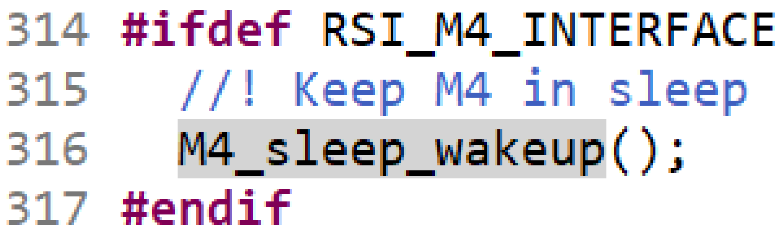
[in] psp_mode

[in] psp_type

parameter	Description
psp_mode	Following psp_mode is defined.
Active(0) : In this mode, module is active and power save is disabled.	
RSI_SLEEP_MODE_1 (1): Connected sleep mode.	
In this sleep mode, SoC will never turn off, therefore no handshake is required before sending data to the module.	
RSI_SLEEP_MODE_2 (2): In this sleep mode, SoC will go to LP/ULP (with/without RAM RETENTION) sleep based on the selected value set for RSI_SELECT_LP_OR_ULP_MODE in rsi_wlan_config.h.	
Therefore handshake is required before sending data to the module	
RSI_SLEEP_MODE_8 (8): Deep sleep mode with ULP RAM RETENTION.	
RSI_SLEEP_MODE_10 (10): Deep sleep mode without ULP RAM RETENTION.	
In deep sleep mode, module will turn off the SoC and a GPIO or Message based handshake is required before sending commands to the module.	
psp_type	Follwing psp_type is defined.
RSI_MAX_PSP (0): This psp_type will be used for max power saving.	
RSI_FAST_PSP (1): This psp_type allows module to disable power save for any Tx / Rx packet for monitor interval of time	
(monitor interval can be set by RSI_MONITOR_INTERVAL in rsi_wlan_config.h file, default value is 50 ms).	
If there is no data for monitor interval of time then module will again enable power save.	
RSI_UAPSD (2): This psp_type is used to enable WMM power save.	

- **Configure Power Save Profile with Connected Sleep**
  - Is done by rsi_wlan_power_save_profile
  - It sets the SiWx917 into power save mode in WLAN mode
  - This is a blocking API
  - Format:
    - Int32_t rsi_wlan_power_save_profile (uint8_t psp_mode, uint8_t psp_type)
  - psp_mode and psp_type can be set as shown by the table to the left

What is the code flow of this application (Continued)?



```
#ifdef RSI_M4_INTERFACE
  //! Keep M4 in sleep
  M4_sleep_wakeup();
#endif
```

- **Set SiWx917’s M4 CPU into sleep mode**
  - Is done by M4_sleep_wakeup
  - This function is used to make the SiWx917’s M4 CPU to go into sleep

Getting Deep into a Code Example: Wi-Fi-Only Standby Associate (TWT)

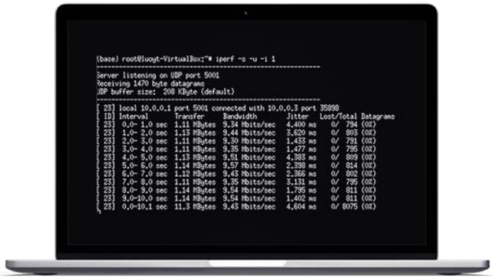
What does the TWT Wi-Fi-Only Standby Associate Code Example do?

Access Point

SiWx917 is connected
to AP via Wi-Fi

PC With TCP
iPerf Server

SiWx917



```
(base) root@luoyt-VirtualBox:~# iperf -s -u -i 1
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[ 23] local 10.0.0.1 port 5001 connected with 10.0.0.3 port 35898
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[ 23]  0.0- 1.0 sec  1.11 MBytes  9.34 Mbits/sec   4.400 ms    0/  794 (0%)
[ 23]  1.0- 2.0 sec  1.13 MBytes  9.44 Mbits/sec   3.620 ms    0/  803 (0%)
[ 23]  2.0- 3.0 sec  1.11 MBytes  9.30 Mbits/sec   1.433 ms    0/  791 (0%)
[ 23]  3.0- 4.0 sec  1.11 MBytes  9.35 Mbits/sec   1.477 ms    0/  795 (0%)
[ 23]  4.0- 5.0 sec  1.13 MBytes  9.51 Mbits/sec   4.383 ms    0/  809 (0%)
[ 23]  5.0- 6.0 sec  1.14 MBytes  9.57 Mbits/sec   2.398 ms    0/  814 (0%)
[ 23]  6.0- 7.0 sec  1.12 MBytes  9.43 Mbits/sec   2.366 ms    0/  802 (0%)
[ 23]  7.0- 8.0 sec  1.11 MBytes  9.35 Mbits/sec   3.131 ms    0/  795 (0%)
[ 23]  8.0- 9.0 sec  1.14 MBytes  9.54 Mbits/sec   1.795 ms    0/  811 (0%)
[ 23]  9.0-10.0 sec  1.14 MBytes  9.54 Mbits/sec   1.402 ms    0/  811 (0%)
[ 23]  0.0-10.1 sec  11.3 MBytes  9.43 Mbits/sec   4.604 ms    0/ 8075 (0%)
```

PC With TCP
iPerf Server

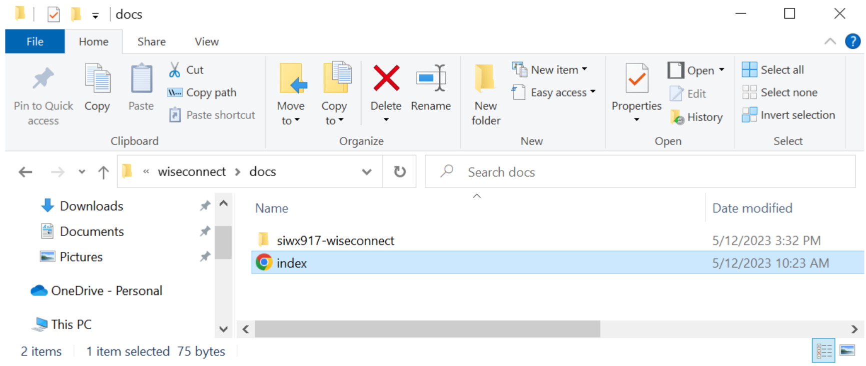
- **Just like the Wi-Fi only standby associate code using DTIM/listen interval, the TWT WI-FI-only Standby Associate code does the following**
  - Configure the SiWx917 into station mode
  - Associate to an Access Point
  - Obtain an IP address from that access point
  - Set the SiWx917 to standby associated mode

  What it does differently to that code is that it wakes up the SiWx917 from standby mode to listen to beacons based on TWT instead of DTIM or listen intervals

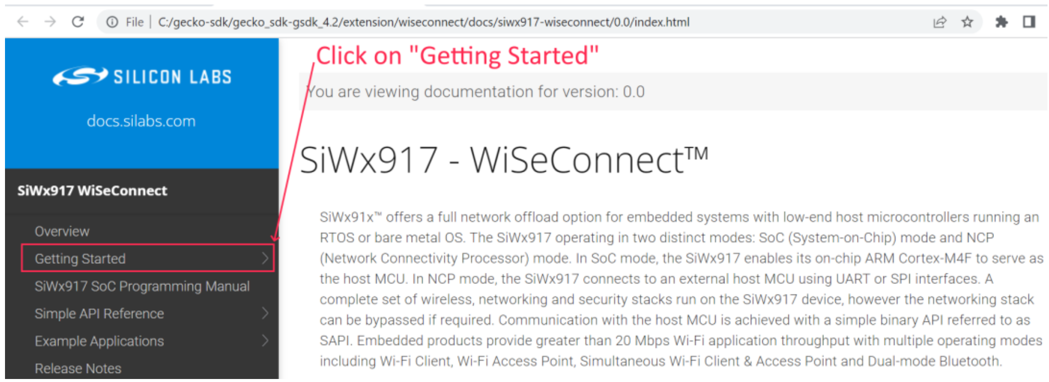
- **The following slides will show us how this is done through SAPIs function calls**

©2023 Silicon Laboratories Inc. All rights reserved.

Let’s open this application up



Name	Date modified
siwx917-wiseconnect	5/12/2023 3:32 PM
index	5/12/2023 10:23 AM


File | C:/gecko-sdk/gecko_sdk-gsdk_4.2/extension/wiseconnect/docs/siwx917-wiseconnect/0.0/index.html
SILICON LABS
docs.silabs.com
SiWx917 WiSeConnect
Overview
Getting Started
SiWx917 SoC Programming Manual
Simple API Reference
Example Applications
Release Notes

Click on "Getting Started"

You are viewing documentation for version: 0.0

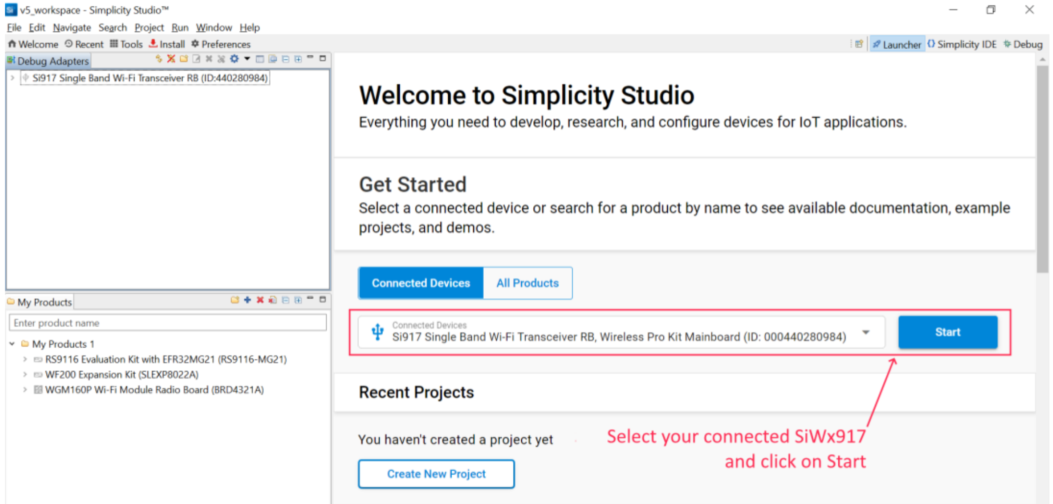
SiWx917 - WiSeConnect™

SiWx91x™ offers a full network offload option for embedded systems with low-end host microcontrollers running an RTOS or bare metal OS. The SiWx917 operating in two distinct modes: SoC (System-on-Chip) mode and NCP (Network Connectivity Processor) mode. In SoC mode, the SiWx917 enables its on-chip ARM Cortex-M4F to serve as the host MCU. In NCP mode, the SiWx917 connects to an external host MCU using UART or SPI interfaces. A complete set of wireless, networking and security stacks run on the SiWx917 device, however the networking stack can be bypassed if required. Communication with the host MCU is achieved with a simple binary API referred to as SAPI. Embedded products provide greater than 20 Mbps Wi-Fi application throughput with multiple operating modes including Wi-Fi Client, Wi-Fi Access Point, Simultaneous Wi-Fi Client & Access Point and Dual-mode Bluetooth.

- **In order to see this code example, you should import its project into Simplicity Studio.**

- **Just like for the DTIM/Listen Interval Wi-Fi-Only Standby Associate code example, follow the instructions on the “index,html” document to get your setup ready to import this project.**

Let’s open this application up



v5_workspace - Simplicity Studio™

File Edit Navigate Search Project Run Window Help

Welcome · Recent · Tools · Install · Preferences | Launcher · Simplicity IDE · Debug

Debug Adapters
- Si917 Single Band Wi-Fi Transceiver RB (ID:440280984)

My Products
- Enter product name
- My Products 1
  - RS9116 Evaluation Kit with EFR32MG21 (RS9116-MG21)
  - WF200 Expansion Kit (SLEXP8022A)
  - WGM160P Wi-Fi Module Radio Board (BRD4321A)

Welcome to Simplicity Studio

Everything you need to develop, research, and configure devices for IoT applications.

Get Started

Select a connected device or search for a product by name to see available documentation, example projects, and demos.

Connected Devices | All Products

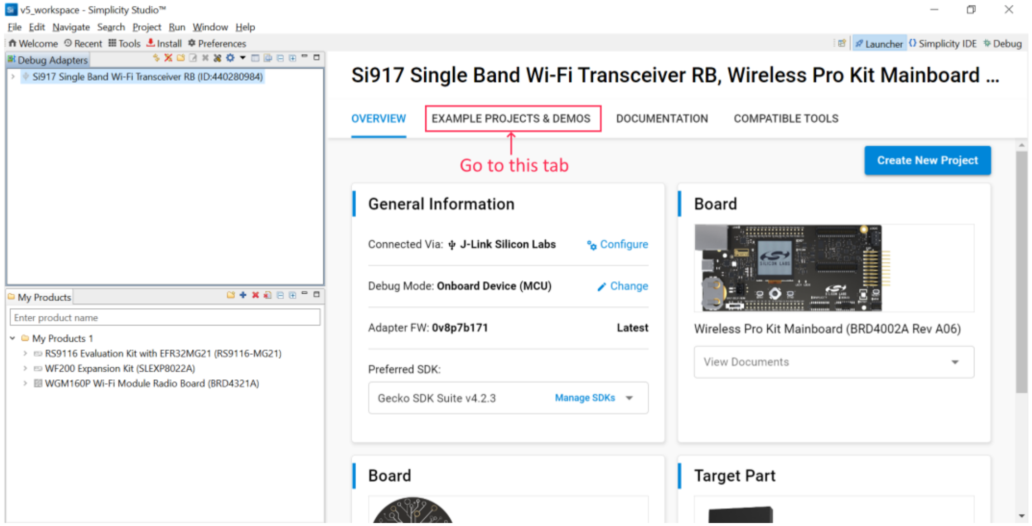
Connected Devices: Si917 Single Band Wi-Fi Transceiver RB, Wireless Pro Kit Mainboard (ID: 000440280984) | Start

Recent Projects

You haven't created a project yet

Create New Project

Select your connected SiWx917 and click on Start



v5_workspace - Simplicity Studio™

File Edit Navigate Search Project Run Window Help

Welcome · Recent · Tools · Install · Preferences | Launcher · Simplicity IDE · Debug

Debug Adapters
- Si917 Single Band Wi-Fi Transceiver RB (ID:440280984)

My Products
- Enter product name
- My Products 1
  - RS9116 Evaluation Kit with EFR32MG21 (RS9116-MG21)
  - WF200 Expansion Kit (SLEXP8022A)
  - WGM160P Wi-Fi Module Radio Board (BRD4321A)

Si917 Single Band Wi-Fi Transceiver RB, Wireless Pro Kit Mainboard ...

OVERVIEW | EXAMPLE PROJECTS & DEMOS | DOCUMENTATION | COMPATIBLE TOOLS

Go to this tab

Create New Project

General Information

Connected Via: J-Link Silicon Labs — Configure

Debug Mode: Onboard Device (MCU) — Change

Adapter FW: 0v8p7b171 — Latest

Preferred SDK: Gecko SDK Suite v4.2.3 — Manage SDKs

Board

Wireless Pro Kit Mainboard (BRD4002A Rev A06)

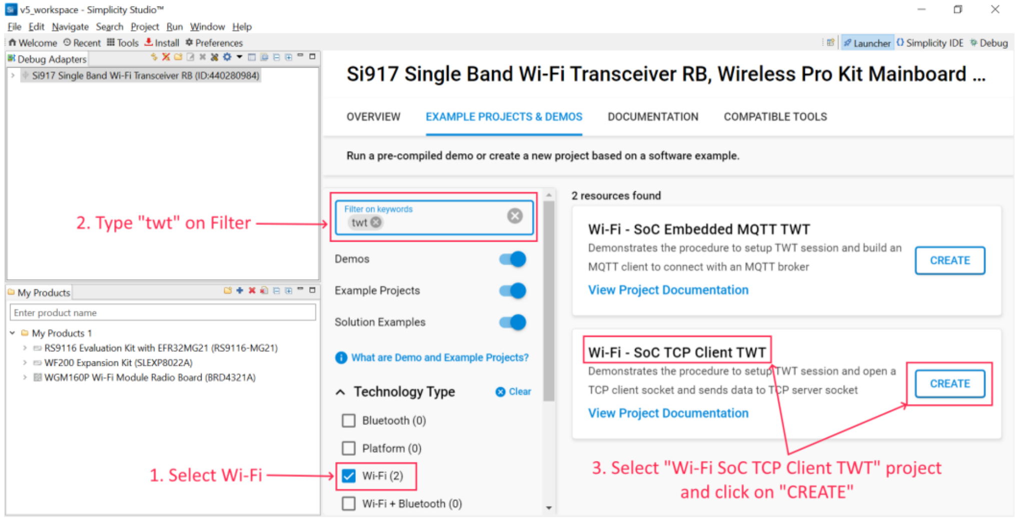
View Documents

Board

Target Part

- Once you have done so, the project that you will now import will be
- Select your SiWx917 and click on Start.
- Go to the “Example Projects & Demos” tab

Let’s open this application up


v5_workspace - Simplicity Studio™
File Edit Navigate Search Project Run Window Help
Welcome Recent Tools Install Preferences
Launcher Simplicity IDE Debug
Debug Adapters
My Products
OVERVIEW
EXAMPLE PROJECTS & DEMOS
DOCUMENTATION
COMPATIBLE TOOLS
Filter on keywords
Clear
CREATE
CREATE

Si917 Single Band Wi-Fi Transceiver RB (ID:440280984)

Enter product name

My Products 1
- RS9116 Evaluation Kit with EFR32MG21 (RS9116-MG21)
- WF200 Expansion Kit (SLEXP8022A)
- WGM160P Wi-Fi Module Radio Board (BRD4321A)

Si917 Single Band Wi-Fi Transceiver RB, Wireless Pro Kit Mainboard ...

Run a pre-compiled demo or create a new project based on a software example.

Filter on keywords: twt

Demos ●
Example Projects ●
Solution Examples ●

What are Demo and Example Projects?

Technology Type
- □ Bluetooth (0)
- □ Platform (0)
- ☑ Wi-Fi (2)
- □ Wi-Fi + Bluetooth (0)

2 resources found

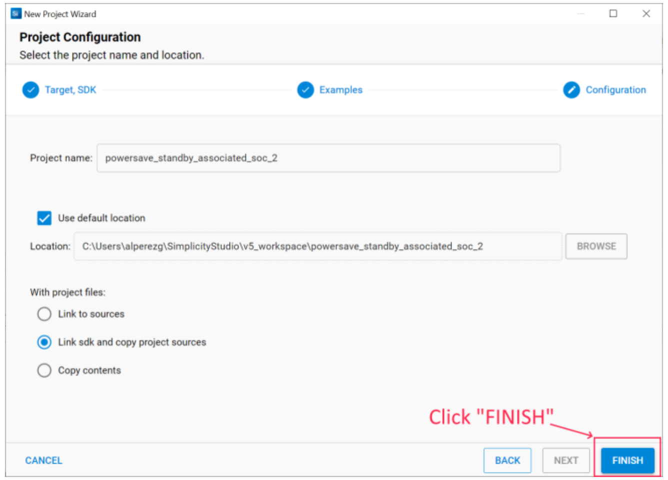
Wi-Fi - SoC Embedded MQTT TWT
Demonstrates the procedure to setup TWT session and build an MQTT client to connect with an MQTT broker
View Project Documentation

Wi-Fi - SoC TCP Client TWT
Demonstrates the procedure to setup TWT session and open a TCP client socket and sends data to TCP server socket
View Project Documentation

2. Type "twt" on Filter

1. Select Wi-Fi

3. Select "Wi-Fi SoC TCP Client TWT" project and click on "CREATE"



Project Configuration
Select the project name and location.

Project name: powersave_standby_associated_soc_2

☑ Use default location

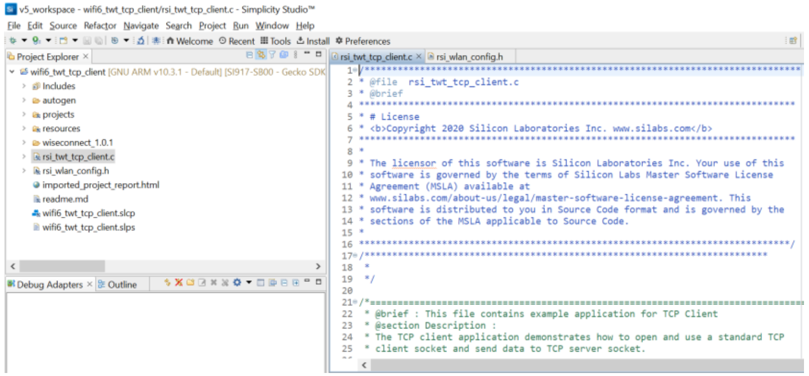
Location: C:\Users\alperezg\SimplicityStudio\v5_workspace\powersave_standby_associated_soc_2

With project files:
- ○ Link to sources
- ● Link sdk and copy project sources
- ○ Copy contents

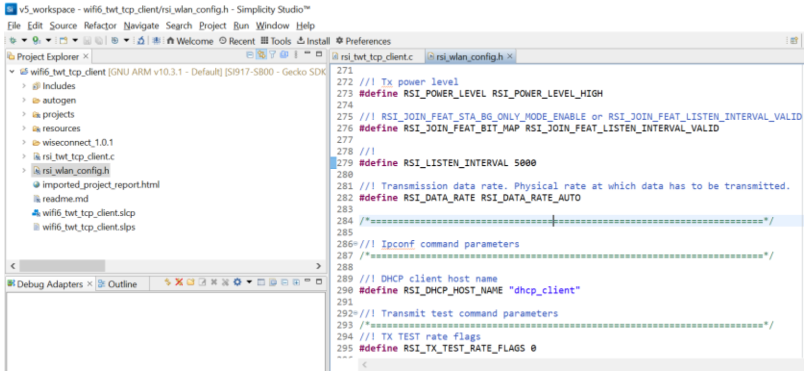
Click "FINISH"

- Once you have done so, to select the project to import do the following:
- Select “Wi-Fi” under technology type
- Enter “twt” in “Filter on keywords”
- Scroll down to find “Wi-Fi – SoC TCP Client TWT” project
- Click on “Create”
- Select your SiWx917 and click on Start.
- On the screen that will pop-up click “Finish”. This will now import the project

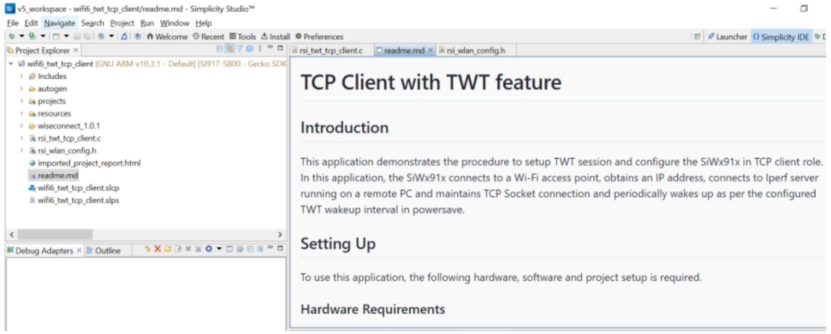
Let’s open this application up


v5_workspace - wifi6_twt_tcp_client/rsi_twt_tcp_client.c - Simplicity Studio™
File Edit Source Refactor Navigate Search Project Run Window Help
Welcome Recent Tools Install Preferences
Project Explorer
wifi6_twt_tcp_client [GNU ARM v10.3.1 - Default] [SI917-SB00 - Gecko SDK
Includes
autogen
projects
resources
wiseconnect_1.0.1
rsi_twt_tcp_client.c
rsi_wlan_config.h
imported_project_report.html
readme.md
wifi6_twt_tcp_client.slcp
wifi6_twt_tcp_client.slps
Debug Adapters
Outline
rsi_twt_tcp_client.c
rsi_wlan_config.h

```
/***************************************************************************
 * @file  rsi_twt_tcp_client.c
 * @brief
 ***************************************************************************
 * # License
 * <b>Copyright 2020 Silicon Laboratories Inc. www.silabs.com</b>
 ***************************************************************************
 *
 * The licensor of this software is Silicon Laboratories Inc. Your use of this
 * software is governed by the terms of Silicon Labs Master Software License
 * Agreement (MSLA) available at
 * www.silabs.com/about-us/legal/master-software-license-agreement. This
 * software is distributed to you in Source Code format and is governed by the
 * sections of the MSLA applicable to Source Code.
 *
 **************************************************************************/
/*************************************************************************
  *
  */

/*=========================================================================
  * @brief : This file contains example application for TCP Client
  * @section Description :
  * The TCP client application demonstrates how to open and use a standard TCP
  * client socket and send data to TCP server socket.
```


v5_workspace - wifi6_twt_tcp_client/rsi_wlan_config.h - Simplicity Studio™
File Edit Source Refactor Navigate Search Project Run Window Help
Welcome Recent Tools Install Preferences
Project Explorer
wifi6_twt_tcp_client [GNU ARM v10.3.1 - Default] [SI917-SB00 - Gecko SDK
Includes
autogen
projects
resources
wiseconnect_1.0.1
rsi_twt_tcp_client.c
rsi_wlan_config.h
imported_project_report.html
readme.md
wifi6_twt_tcp_client.slcp
wifi6_twt_tcp_client.slps
Debug Adapters
Outline
rsi_twt_tcp_client.c
rsi_wlan_config.h

```

//! Tx power level
#define RSI_POWER_LEVEL RSI_POWER_LEVEL_HIGH

//! RSI_JOIN_FEAT_STA_BG_ONLY_MODE_ENABLE or RSI_JOIN_FEAT_LISTEN_INTERVAL_VALID
#define RSI_JOIN_FEAT_BIT_MAP RSI_JOIN_FEAT_LISTEN_INTERVAL_VALID

//!
#define RSI_LISTEN_INTERVAL 5000

//! Transmission data rate. Physical rate at which data has to be transmitted.
#define RSI_DATA_RATE RSI_DATA_RATE_AUTO

/*====================================================================*/

//! Ipconf command parameters
/*====================================================================*/

//! DHCP client host name
#define RSI_DHCP_HOST_NAME "dhcp_client"

//! Transmit test command parameters
/*====================================================================*/
//! TX TEST rate flags
#define RSI_TX_TEST_RATE_FLAGS 0
```


v5_workspace - wifi6_twt_tcp_client/readme.md - Simplicity Studio™
File Edit Navigate Search Project Run Window Help
Welcome Recent Tools Install Preferences
Launcher
Simplicity IDE
Project Explorer
wifi6_twt_tcp_client [GNU ARM v10.3.1 - Default] [SI917-SB00 - Gecko SDK
Includes
autogen
projects
resources
wiseconnect_1.0.1
rsi_twt_tcp_client.c
rsi_wlan_config.h
imported_project_report.html
readme.md
wifi6_twt_tcp_client.slcp
wifi6_twt_tcp_client.slps
Debug Adapters
Outline
rsi_twt_tcp_client.c
readme.md
rsi_wlan_config.h

TCP Client with TWT feature

Introduction

This application demonstrates the procedure to setup TWT session and configure the SiWx91x in TCP client role. In this application, the SiWx91x connects to a Wi-Fi access point, obtains an IP address, connects to Iperf server running on a remote PC and maintains TCP Socket connection and periodically wakes up as per the configured TWT wakeup interval in powersave.

Setting Up

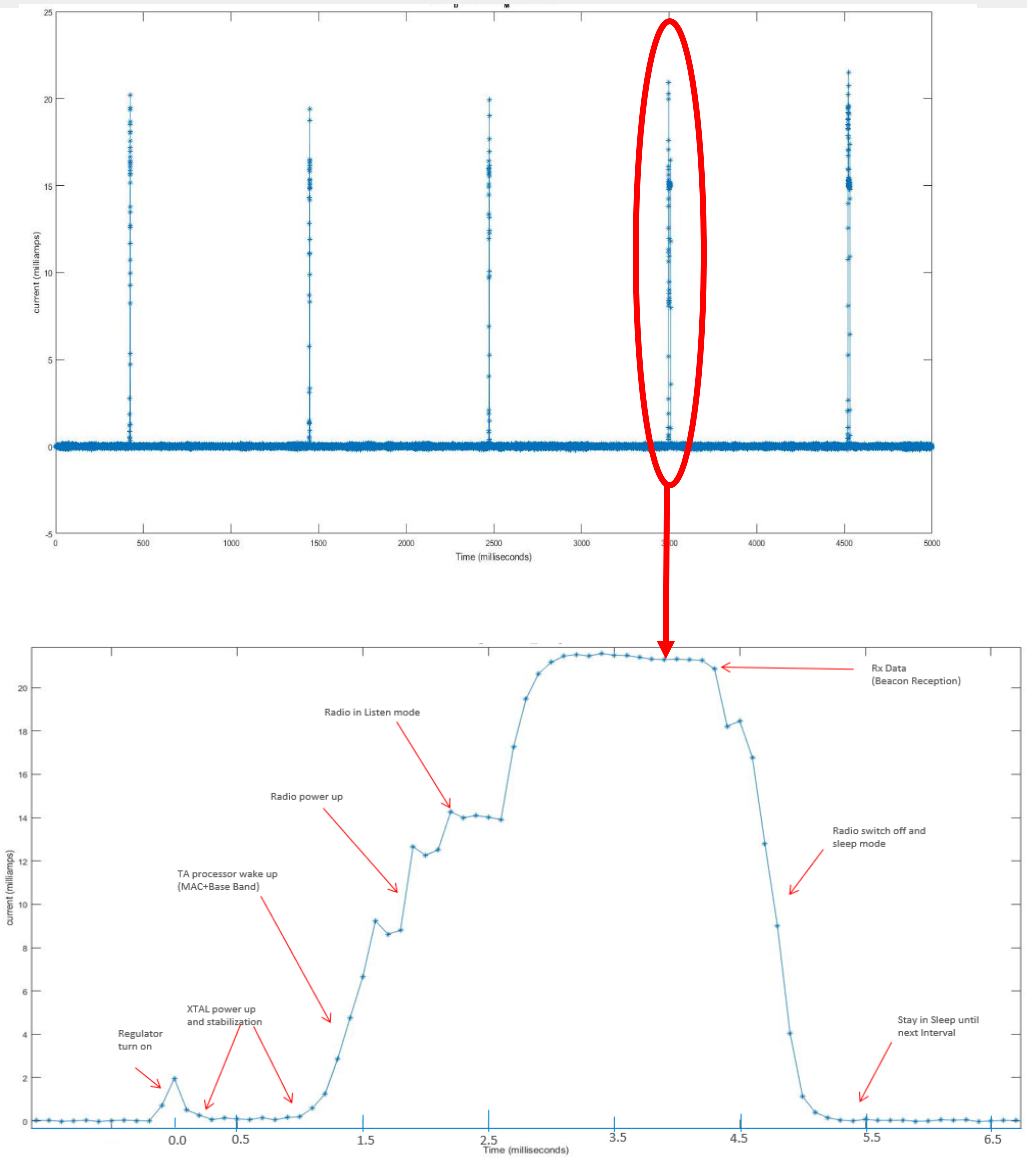
To use this application, the following hardware, software and project setup is required.

Hardware Requirements

- **After importing this project onto Simplicity Studio, open the following two files as shown here.**
  - rsi_twt_tcp_client.c
  - rsi_wlan_config.h
  - readme.md
- **The rsi_twt_tcp_client.c file is the main code file for this application, we’ll talk now about what it does.**

©2023 Silicon Laboratories Inc. All rights reserved.

Let’s explain what the application does

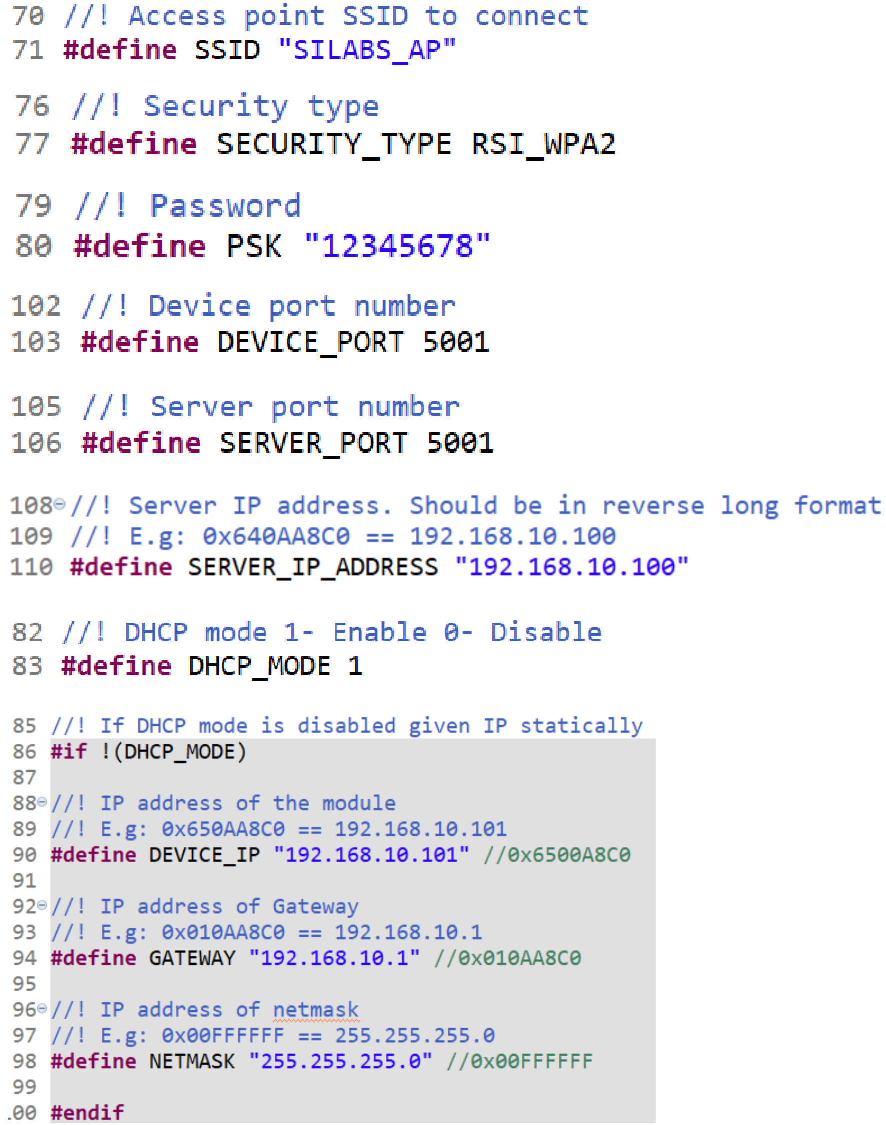


- **Let’s talk about what is it that this application does. It is the following:**
  - Application first initializes SiWx917 drivers. Buffer allocation is done for all required buffers
  - After this, SiWx917 device is initialized, valid firmware is loaded and module is powered up to run application.
  - As SiWx917 is initialized, its required features are enabled using wireless initialization.
  - Application gives SSID to SiWx917 and it’s instructed to scan for this SSID in all Wi-Fi channels
  - Application gives security type and password to SiWx917. Device checks if it matches that of the AP and, if so associates to it
  - SiWx917 obtains IP address either through configured info or through DHCP
  - Before setting TWT parameters, a socket is created and a connection to a remote PC running a TCP iPerf server is established with SiWx917 in client mode
  - Application enables TWT support as specified in rsi_wlan_common_config.h file by enabling the following parameter macros::
    - HE_PARAMS_SUPPORT
    - TWT_SUPPORT

- **TWT parameters used for sleep should be configured as per user’s requirements. How to do so is described in the application readme and in the next slides**

©2023 Silicon Laboratories Inc. All rights reserved.

What can we configure in this application? (In rsi_twt_tcp_client.c)

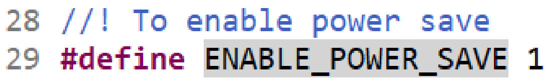


```
//! Access point SSID to connect
#define SSID "SILABS_AP"

//! Security type
#define SECURITY_TYPE RSI_WPA2

//! Password
#define PSK "12345678"

//! Device port number
#define DEVICE_PORT 5001

//! Server port number
#define SERVER_PORT 5001

//! Server IP address. Should be in reverse long format
//! E.g: 0x640AA8C0 == 192.168.10.100
#define SERVER_IP_ADDRESS "192.168.10.100"

//! DHCP mode 1- Enable 0- Disable
#define DHCP_MODE 1

//! If DHCP mode is disabled given IP statically
#if !(DHCP_MODE)

//! IP address of the module
//! E.g: 0x650AA8C0 == 192.168.10.101
#define DEVICE_IP "192.168.10.101" //0x6500A8C0

//! IP address of Gateway
//! E.g: 0x010AA8C0 == 192.168.10.1
#define GATEWAY "192.168.10.1" //0x010AA8C0

//! IP address of netmask
//! E.g: 0x00FFFFFF == 255.255.255.0
#define NETMASK "255.255.255.0" //0x00FFFFFF

#endif
```

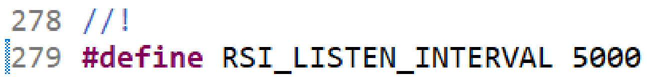
- **Let's go over the variables that are configurable in this application.**
  - SSID for network to connect (#define SSID)
  - Security type (#define SECURITY_TYPE). Some valid options are:
    - RSI_OPEN = open security
    - RSI_WPA = WPA
    - RSI_WPA2 = WPA2
    - RSI_WPA3 = WPA3
    - RSI_WEP = WEP
    - RSI_WPA_EAP = Enterprise WPA EAP
    - RSI_WPA2_EAP = Enterprise WPA2 EAP
    - RSI_WPS_PI = WPS with PIN
  - Security Password (#define PSK)
  - TCP source port for TCP connection (#define DEVICE_PORT)
  - TCP destination port at TCP server on PC (#define SERVER_PORT)
  - IP address of TCP server (#define SERVER_IP_ADDRESS)
  - Enabling or disabling DHCP (#define DHCP_MODE)
  - IP addressing for SiWx917 (If DHCP is disabled)
    - #define DEVICE_IP for SiWx917 IP address
    - #define GATEWAY for Gateway IP address
    - #define NETMASK for Netmask

©2023 Silicon Laboratories Inc. All rights reserved.

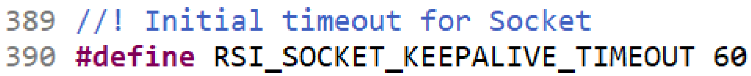
What can we configure in this application? (In rsi_wlan_config.h)



```
//! To enable power save
#define ENABLE_POWER_SAVE 1
```



```
//!
#define RSI_LISTEN_INTERVAL 5000
```

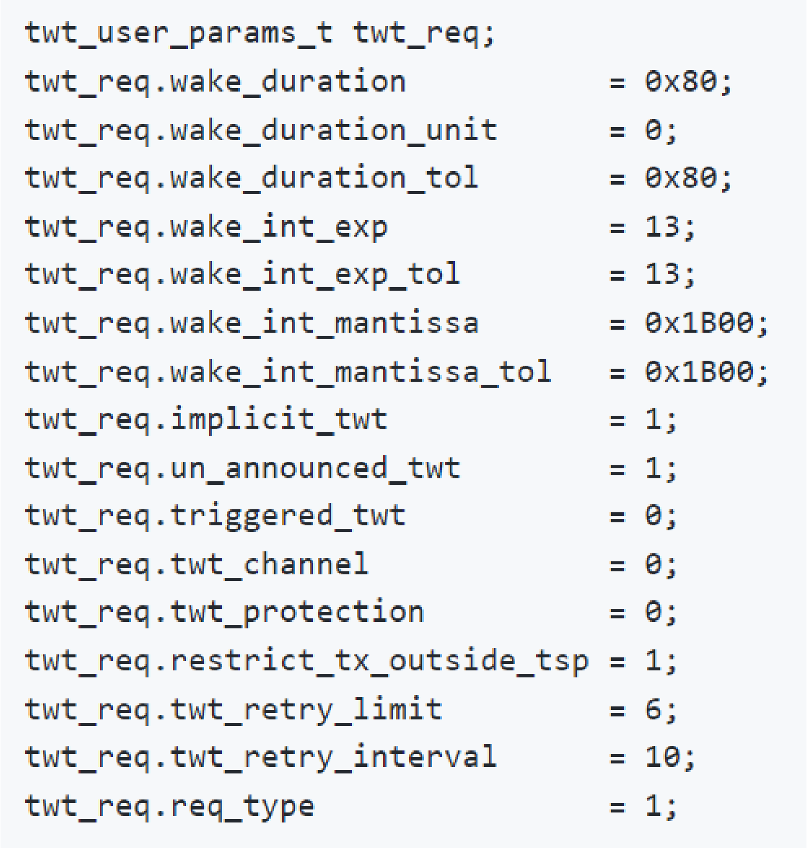


```
//! Initial timeout for Socket
#define RSI_SOCKET_KEEPALIVE_TIMEOUT 60
```

- **Let’s go over the variables that are configurable in this application in its .**
  - Enabling power save (#define ENABLE_POWER_SAVE)
    - Note that by default, power save is disabled (#define ENABLE_POWER_SAVE 0)
    - It must be enabled to use TWT (#define ENABLE_POWER_SAVE 1)
  - Listen interval (#define RSI_LISTEN_INTERVAL)
    - Is set in mSec. Default it is set to 5 seconds (5,000 mSec)
  - Socket keepalive timeout
    - It is set In Seconds, default is set to 60 seconds
  - TWT parameters should be set as described in the readme included with this application (readme.md). The following slide will give a quick description of these parameters

©2023 Silicon Laboratories Inc. All rights reserved.

What can we configure in this application (TWT Parameters)?

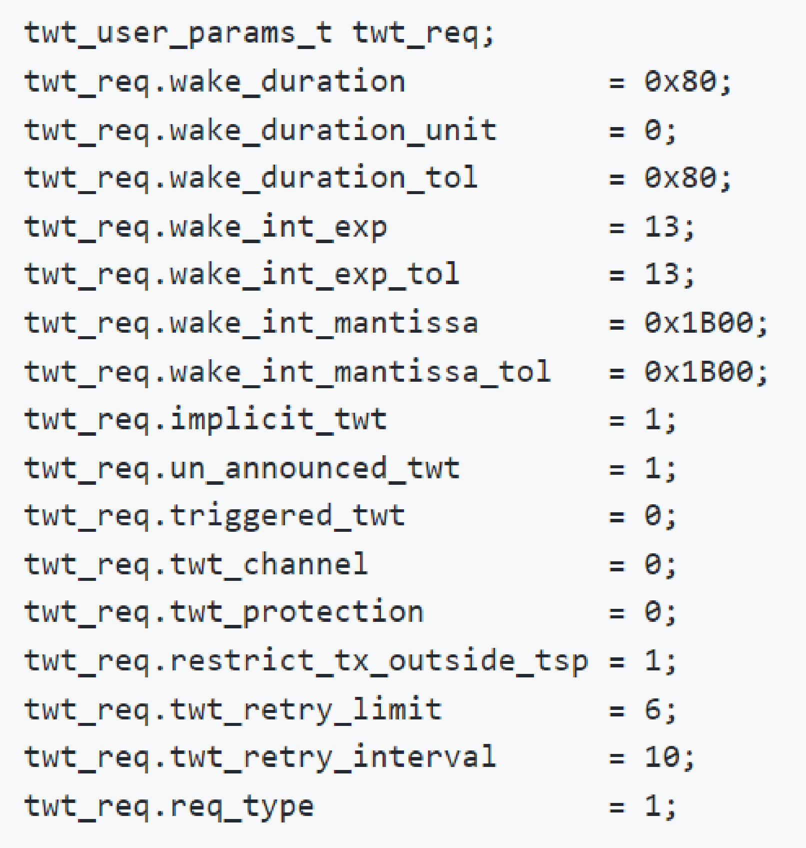


```
twt_user_params_t twt_req;
twt_req.wake_duration             = 0x80;
twt_req.wake_duration_unit        = 0;
twt_req.wake_duration_tol         = 0x80;
twt_req.wake_int_exp              = 13;
twt_req.wake_int_exp_tol          = 13;
twt_req.wake_int_mantissa         = 0x1B00;
twt_req.wake_int_mantissa_tol     = 0x1B00;
twt_req.implicit_twt              = 1;
twt_req.un_announced_twt          = 1;
twt_req.triggered_twt             = 0;
twt_req.twt_channel               = 0;
twt_req.twt_protection            = 0;
twt_req.restrict_tx_outside_tsp   = 1;
twt_req.twt_retry_limit           = 6;
twt_req.twt_retry_interval        = 10;
twt_req.req_type                  = 1;
```

- **Let's now give a quick description of the TWT parameters**
- **These parameters are set in the rsi_wlan_common_config.h file**
- **This file can be located as described on the readme.md file in the project**
- **The configurable parameters are as follows:**
  - HE Parameters support
    - Is set through #define HE_PARAMS_SUPPORT macro. By default it is enabled.
  - TWT Support
    - Is set through #define TWT_SUPPORT. By default it is enabled
  - iTWT Setup configuration
    - Is configured and filled into the structure type twt_user_params_t in rsi_twt_tcp_client.c and passed as a parameter to rsi_wlan_config() API
    - The following page will talk more about this and show a sample configuration
    - The following is a sample setup API call with twt_enable = 1 and flow_id = 1
      - status = rsi_wlan_twt_config(1,1,&twt_req)

©2023 Silicon Laboratories Inc. All rights reserved.

What can we configure in this application? (TWT Parameters continued)

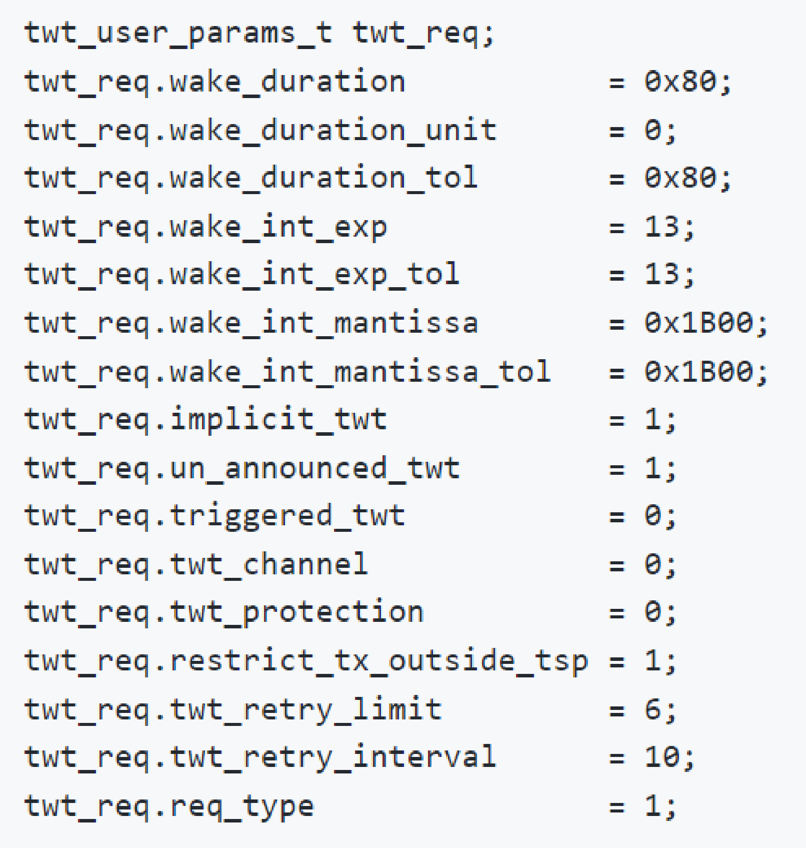


```
twt_user_params_t twt_req;
twt_req.wake_duration             = 0x80;
twt_req.wake_duration_unit        = 0;
twt_req.wake_duration_tol         = 0x80;
twt_req.wake_int_exp              = 13;
twt_req.wake_int_exp_tol          = 13;
twt_req.wake_int_mantissa         = 0x1B00;
twt_req.wake_int_mantissa_tol     = 0x1B00;
twt_req.implicit_twt              = 1;
twt_req.un_announced_twt          = 1;
twt_req.triggered_twt             = 0;
twt_req.twt_channel               = 0;
twt_req.twt_protection            = 0;
twt_req.restrict_tx_outside_tsp = 1;
twt_req.twt_retry_limit           = 6;
twt_req.twt_retry_interval        = 10;
twt_req.req_type                  = 1;
```

- **Let’s go over the twt_req parameters:**
- **Wake Duration:**
  - Wake duration (wake_duration):
    - nominal minimum TWT wake duration of TWT.
    - Time for which SiWx917 will be in wake state for data Tx or Rx.
    - Allowed values range is 0-255
  - Wake duration unit (wake_duration_unit):
    - Specifies unit used for wake duration
    - Allowed values are 0 (256 uSec) and 1 (1,024 uSec)
  - Wake_duration_tol (wake_duration_tol):
    - Wake duration tolerance allowed for wake duration in case of suggested TWT
    - If AP suggests wake duration outside of tolerance, TWT suggestion will be rejected
    - Allowed range is 0-255

©2023 Silicon Laboratories Inc. All rights reserv ed.

What can we configure in this application? (TWT Parameters continued)



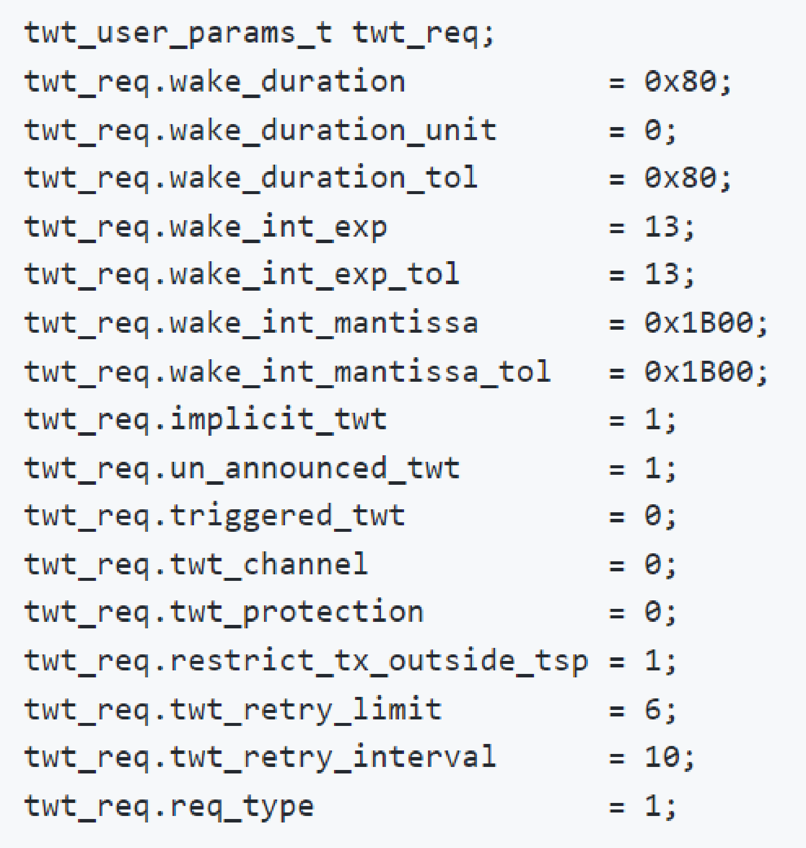
```
twt_user_params_t twt_req;
twt_req.wake_duration              = 0x80;
twt_req.wake_duration_unit         = 0;
twt_req.wake_duration_tol          = 0x80;
twt_req.wake_int_exp               = 13;
twt_req.wake_int_exp_tol           = 13;
twt_req.wake_int_mantissa          = 0x1B00;
twt_req.wake_int_mantissa_tol      = 0x1B00;
twt_req.implicit_twt               = 1;
twt_req.un_announced_twt           = 1;
twt_req.triggered_twt              = 0;
twt_req.twt_channel                = 0;
twt_req.twt_protection             = 0;
twt_req.restrict_tx_outside_tsp = 1;
twt_req.twt_retry_limit            = 6;
twt_req.twt_retry_interval         = 10;
twt_req.req_type                   = 1;
```

- **Wake Interval:**

$$\text{TWT wake interval} = (\text{TWT Wake Interval Mantissa}) \times 2^{(\text{TWT Wake Interval Exponent})} \text{ (in microseconds)}$$

- Wake interval exponent (wake_int_exp):
  - Specifies the TWT wake interval exponent in base 2
  - Allowed values go from 0 to 31
- Wake Interval Exponent Tolerance (wake_int_exp_tol):
  - Wake interval exponent tolerance allowed for wake duration in case of suggested TWT
  - If AP suggests wake interval exponent outside of tolerance, TWT suggestion will be rejected
  - Allowed range is 0 to 31
- Wake Interval Mantissa (wake_int_mantissa):
  - This is the TWT Wake interval mantissa
  - Allowed range is 0-65535

What can we configure in this application? (TWT Parameters continued)

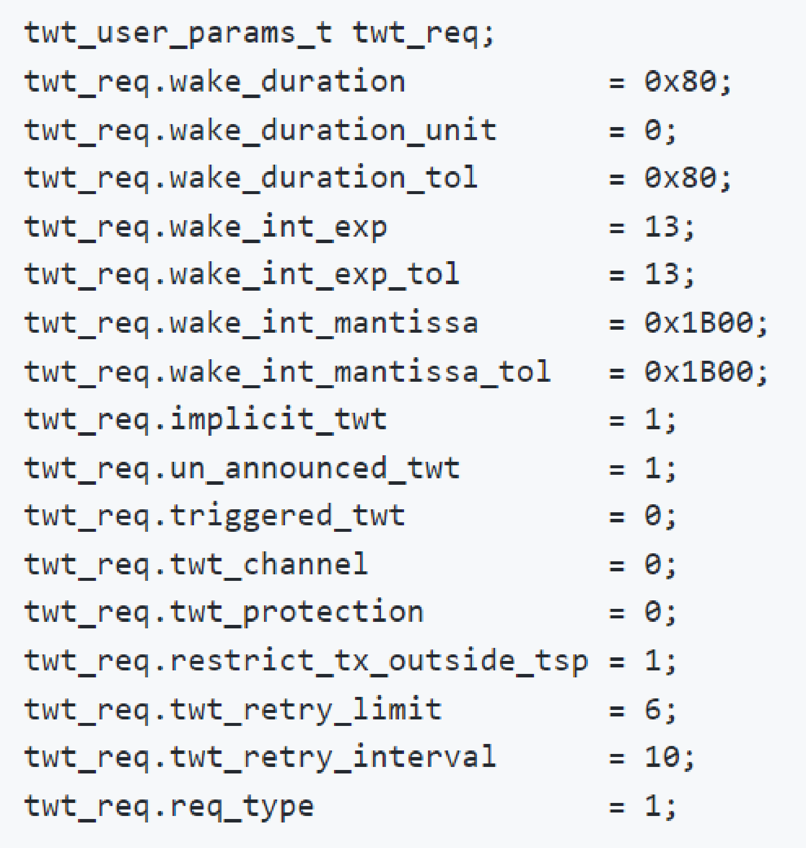


```
twt_user_params_t twt_req;
twt_req.wake_duration             = 0x80;
twt_req.wake_duration_unit        = 0;
twt_req.wake_duration_tol         = 0x80;
twt_req.wake_int_exp              = 13;
twt_req.wake_int_exp_tol          = 13;
twt_req.wake_int_mantissa         = 0x1B00;
twt_req.wake_int_mantissa_tol     = 0x1B00;
twt_req.implicit_twt              = 1;
twt_req.un_announced_twt          = 1;
twt_req.triggered_twt             = 0;
twt_req.twt_channel               = 0;
twt_req.twt_protection            = 0;
twt_req.restrict_tx_outside_tsp = 1;
twt_req.twt_retry_limit           = 6;
twt_req.twt_retry_interval        = 10;
twt_req.req_type                  = 1;
```

- **General TWT Configuration:**
  - Implicit TWT (implicit_twt):
    - If enabled (1), TWT requesting station calculates next TWT by adding fixed value to current TWT value
    - Explicit TWT is currently not allowed
  - Unannounced TWT (un_announced_twt)
    - If enabled (1) TWT requesting STA doesn't announce its wake up to AP through PS-Poll or UAPSD trigger frames
  - Triggered TWT (triggered_twt)
    - If enabled(1), at least one trigger frame is included in TWT Service Period (TSP)
  - TWT channel (twt_channel)
    - Currently this configuration is not allowed
  - TWT protection (twt_protection)
    - If enabled (1), TSP is protected. This is negotiable with AP.
    - Currently not supported, thus only 0 is allowed

©2023 Silicon Laboratories Inc. All rights reserved.

What can we configure in this application? (TWT Parameters continued)

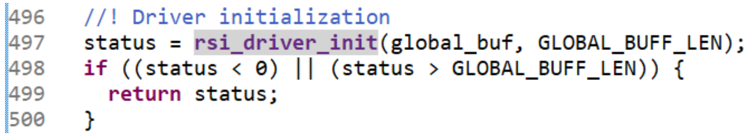


```
twt_user_params_t twt_req;
twt_req.wake_duration              = 0x80;
twt_req.wake_duration_unit         = 0;
twt_req.wake_duration_tol          = 0x80;
twt_req.wake_int_exp               = 13;
twt_req.wake_int_exp_tol           = 13;
twt_req.wake_int_mantissa          = 0x1B00;
twt_req.wake_int_mantissa_tol      = 0x1B00;
twt_req.implicit_twt               = 1;
twt_req.un_announced_twt           = 1;
twt_req.triggered_twt              = 0;
twt_req.twt_channel                = 0;
twt_req.twt_protection             = 0;
twt_req.restrict_tx_outside_tsp = 1;
twt_req.twt_retry_limit            = 6;
twt_req.twt_retry_interval         = 10;
twt_req.req_type                   = 1;
```

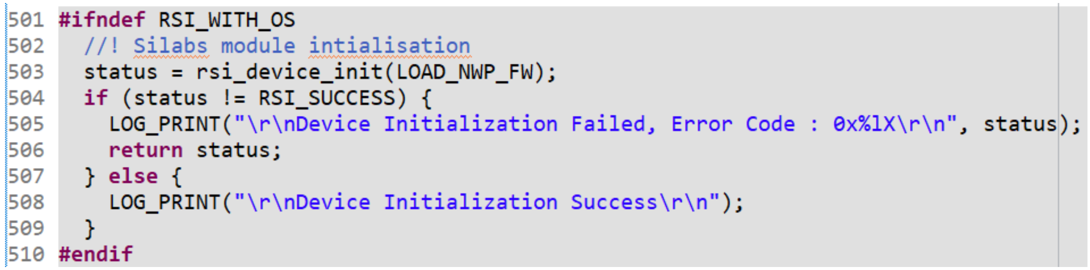
- **General TWT Configuration (Continued):**
  - Restrict Transmission Outside TSP (testrict_tx_outside_tsp):
    - If enabled (1), any Tx outside the TSP is restricted.
    - Else, TX can also happen outside the TSP
  - TWT Retry Limit (twt_retry_limit)
    - The interval between two TWT request retries
    - Specified in seconds
    - Allowed values are 5 - 255
  - TWT Request Type (req_type)
    - This is the TWT request type
    - Options are as follows:
      - 0 – Request TWT
      - 1 – Suggest TWT
      - 2 – Demand TWT

©2023 Silicon Laboratories Inc. All rights reserved.

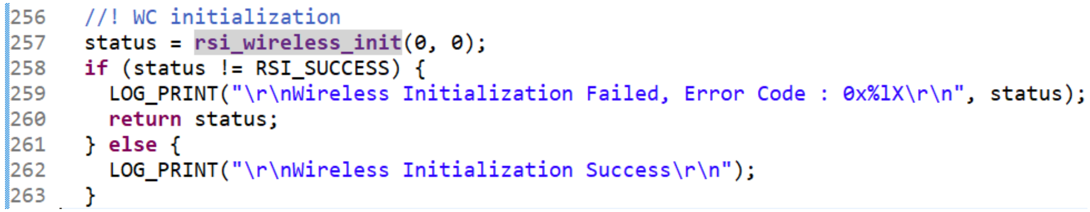
What is the code flow of this application?



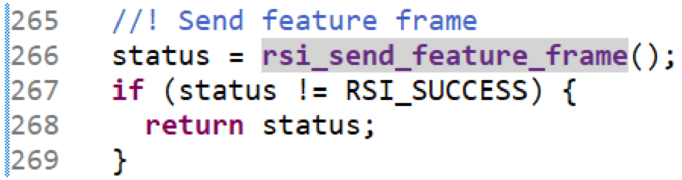
```
  //! Driver initialization
  status = rsi_driver_init(global_buf, GLOBAL_BUFF_LEN);
  if ((status < 0) || (status > GLOBAL_BUFF_LEN)) {
    return status;
  }
```



```
#ifndef RSI_WITH_OS
  //! Silabs module intialisation
  status = rsi_device_init(LOAD_NWP_FW);
  if (status != RSI_SUCCESS) {
    LOG_PRINT("\r\nDevice Initialization Failed, Error Code : 0x%lX\r\n", status);
    return status;
  } else {
    LOG_PRINT("\r\nDevice Initialization Success\r\n");
  }
#endif
```



```
  //! WC initialization
  status = rsi_wireless_init(0, 0);
  if (status != RSI_SUCCESS) {
    LOG_PRINT("\r\nWireless Initialization Failed, Error Code : 0x%lX\r\n", status);
    return status;
  } else {
    LOG_PRINT("\r\nWireless Initialization Success\r\n");
  }
```

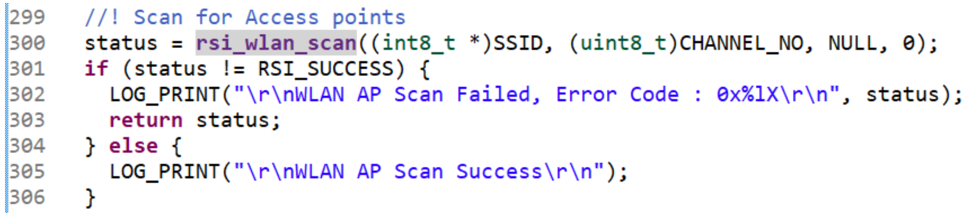


```
  //! Send feature frame
  status = rsi_send_feature_frame();
  if (status != RSI_SUCCESS) {
    return status;
  }
```

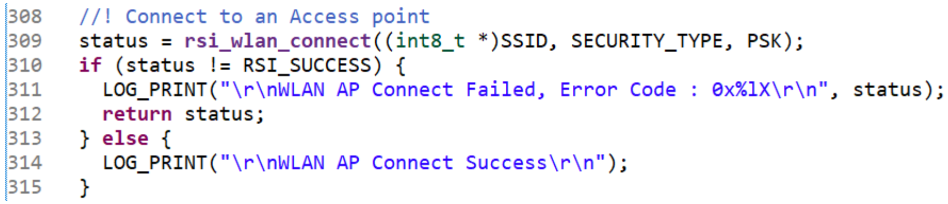
- **Let's now go over how the code of this application flows to do what it does**
- **Driver Initialization**
  - Is done by rsi_driver_init function as in DTIM/Listen Interval code
- **Device Initialization**
  - Is done by rsi_device_init function as in DTIM/Listen Interval code
- **Wireless Initialization**
  - Is done by rsi_wireless_init as in DTIM/Listen Interval code
- **Send Feature Frame**
  - Is done by rsi_send_feature_frame as in DTIM/Listen Interval code

©2023 Silicon Laboratories Inc. All rights reserved.

What is the code flow of this application? (Continued)



```c
//! Scan for Access points
status = rsi_wlan_scan((int8_t *)SSID, (uint8_t)CHANNEL_NO, NULL, 0);
if (status != RSI_SUCCESS) {
  LOG_PRINT("\r\nWLAN AP Scan Failed, Error Code : 0x%lX\r\n", status);
  return status;
} else {
  LOG_PRINT("\r\nWLAN AP Scan Success\r\n");
}
```

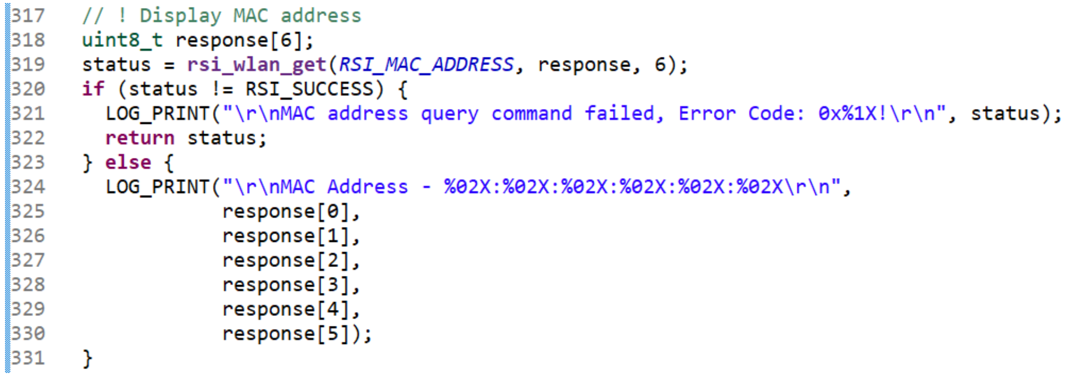


```c
//! Connect to an Access point
status = rsi_wlan_connect((int8_t *)SSID, SECURITY_TYPE, PSK);
if (status != RSI_SUCCESS) {
  LOG_PRINT("\r\nWLAN AP Connect Failed, Error Code : 0x%lX\r\n", status);
  return status;
} else {
  LOG_PRINT("\r\nWLAN AP Connect Success\r\n");
}
```

- **Scan for Access Points**
  - Is done by rsi_wlan_scan function as in DTIM/Listen Interval code
- **Connect to Access Point**
  - Is done by rsi_wlan_connect function as in DTIM/Listen Interval code

©2023 Silicon Laboratories Inc. All rights reserved.

What is the code flow of this application? (Continued)

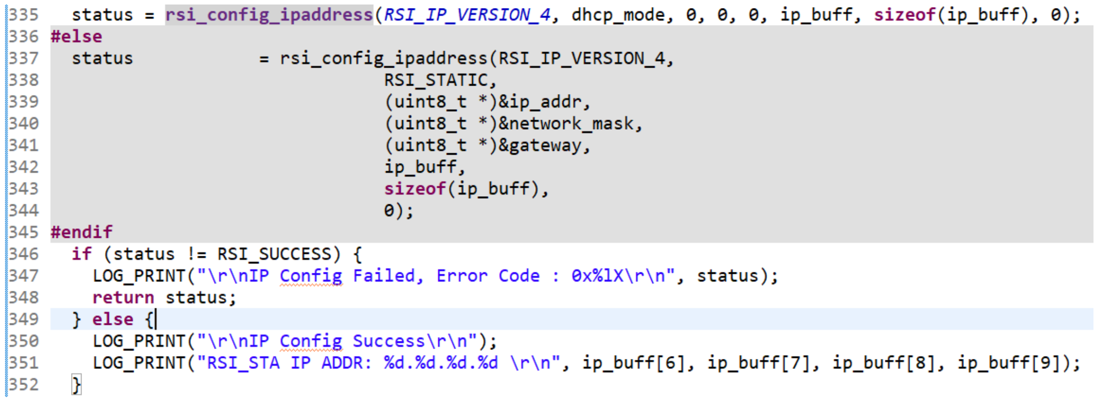


```c
// ! Display MAC address
uint8_t response[6];
status = rsi_wlan_get(RSI_MAC_ADDRESS, response, 6);
if (status != RSI_SUCCESS) {
  LOG_PRINT("\r\nMAC address query command failed, Error Code: 0x%1X!\r\n", status);
  return status;
} else {
  LOG_PRINT("\r\nMAC Address - %02X:%02X:%02X:%02X:%02X:%02X\r\n",
            response[0],
            response[1],
            response[2],
            response[3],
            response[4],
            response[5]);
}
```

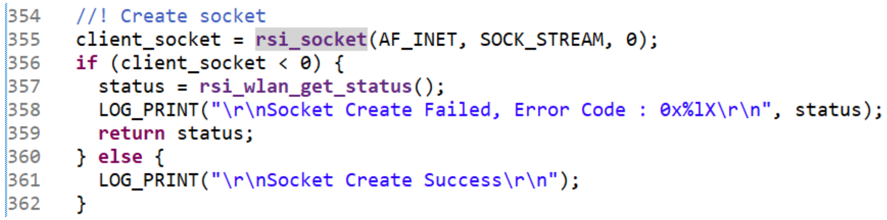
- **Display MAC Address**
  - Is done using rsi_wlan_get function call
  - This is a blocking API
  - Format:
    - rsi_wlan_get(rsi_wlan_query_cmd_t cmd_type, uint8_t *response, uint16_t length)
    - cmd_type can be:
      - 1 (RSI_FW_VERSION): Firmware version
      - 2 (RSI_MAC_ADDRESS): MAC Address
      - 3 (RSI_RSSI): RSSI
      - 4 (RSI_WLAN_INFO): WLAN Information
      - 5 (RSI_CONNECTION_STATUS): Wi-Fi connection status
      - 6 (RSI_STATIONS_INFO): Wi-Fi station information
      - 7 (RSI_SOCKETS_INFO): Socket information
      - 8 (RSI_CFG_GET): Configuration get
      - 9 (RSI_GET_WLAN_STATS): Query for WLAN statistics
      - 10 (RSI_WLAN_EXT_STATS): Query for WLAN EXT statistics
    - Response is output parameter where response is provided
    - Length is length of response buffer in bytes

©2023 Silicon Laboratories Inc. All rights reserved.

What is the code flow of this application?



```
  status = rsi_config_ipaddress(RSI_IP_VERSION_4, dhcp_mode, 0, 0, 0, ip_buff, sizeof(ip_buff), 0);
#else
  status             = rsi_config_ipaddress(RSI_IP_VERSION_4,
                                RSI_STATIC,
                                (uint8_t *)&ip_addr,
                                (uint8_t *)&network_mask,
                                (uint8_t *)&gateway,
                                ip_buff,
                                sizeof(ip_buff),
                                0);
#endif
  if (status != RSI_SUCCESS) {
    LOG_PRINT("\r\nIP Config Failed, Error Code : 0x%lX\r\n", status);
    return status;
  } else {
    LOG_PRINT("\r\nIP Config Success\r\n");
    LOG_PRINT("RSI_STA IP ADDR: %d.%d.%d.%d \r\n", ip_buff[6], ip_buff[7], ip_buff[8], ip_buff[9]);
  }
```

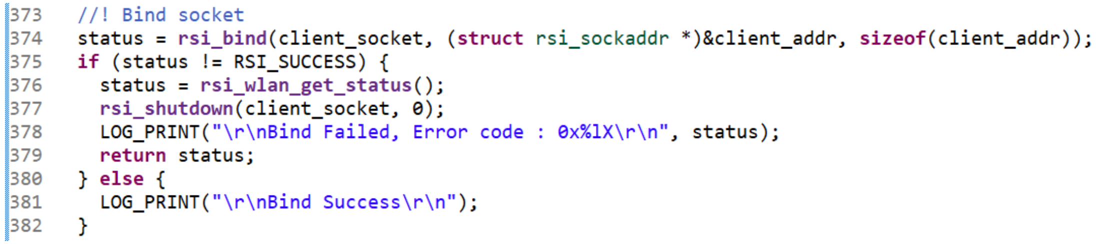


```
  //! Create socket
  client_socket = rsi_socket(AF_INET, SOCK_STREAM, 0);
  if (client_socket < 0) {
    status = rsi_wlan_get_status();
    LOG_PRINT("\r\nSocket Create Failed, Error Code : 0x%lX\r\n", status);
    return status;
  } else {
    LOG_PRINT("\r\nSocket Create Success\r\n");
  }
```

- **Configure IP address of SiWx917**
  - Is done by rsi_config_ipaddress function as in DTIM/Listen Interval code
- **Create Socket**
  - Is done by rsi_socket function
  - This is a non-blocking API
  - Format:
    - Int32_t rsi_socket(int32_t protocolFamily, int32_t type, int32_t protocol)
  - protocolFamily
    - Use 2 (AF_INET) for IPv4 socket
    - Use 3 (AF_INET6) for IPv6 socket
  - type
    - Use 1 (SOCK_STREAM) for TCP socket
    - Use 2 (SOCK_DGRAM) for UDP socket
    - Use 3 (SOCK_RAW) for Raw socket
  - Protocol
    - Use 0 for Non-SSL sockets
    - Use 1 for SSL sockets

©2023 Silicon Laboratories Inc. All rights reserved.

What is the code flow of this application?

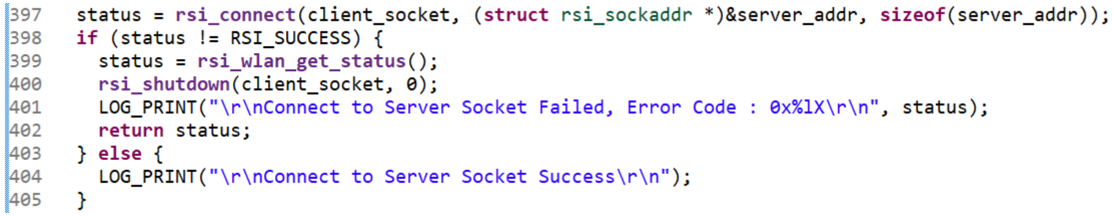


```c
//! Bind socket
status = rsi_bind(client_socket, (struct rsi_sockaddr *)&client_addr, sizeof(client_addr));
if (status != RSI_SUCCESS) {
  status = rsi_wlan_get_status();
  rsi_shutdown(client_socket, 0);
  LOG_PRINT("\r\nBind Failed, Error code : 0x%lX\r\n", status);
  return status;
} else {
  LOG_PRINT("\r\nBind Success\r\n");
}
```

- **Bind Socket**
  - Is done by rsi_bind function
  - This is a non-blocking API
  - Format:
    - int32_t rsi_bind (int32_t sockID, struct rsi_sockaddr *localAddress, int32_t addressLength)
  - sockID
    - This is the socket descriptor ID
  - localAddress
    - This is the address assigned to the socket
    - It uses BSD socket compatible format
  - addressLength
    - This is the length of the address in bytes

©2023 Silicon Laboratories Inc. All rights reserved.

What is the code flow of this application?

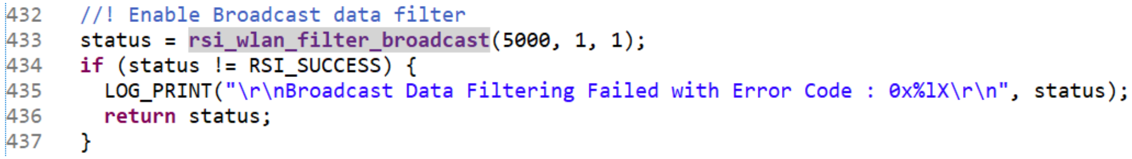


```
status = rsi_connect(client_socket, (struct rsi_sockaddr *)&server_addr, sizeof(server_addr));
if (status != RSI_SUCCESS) {
  status = rsi_wlan_get_status();
  rsi_shutdown(client_socket, 0);
  LOG_PRINT("\r\nConnect to Server Socket Failed, Error Code : 0x%lX\r\n", status);
  return status;
} else {
  LOG_PRINT("\r\nConnect to Server Socket Success\r\n");
}
```

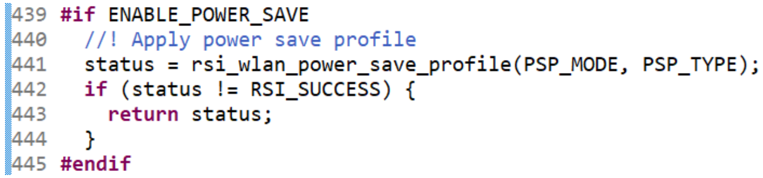
- **Connect to Server Socket**
  - Is done by rsi_connect function
  - This is a blocking API
  - Format:
    - int32_t rsi_connect (int32_t sockID, struct rsi_sockaddr *remoteAddress, int32_t addressLength)
  - sockID
    - This is the socket descriptor ID
  - remoteAddress
    - This is the remote peer address.
    - Its format is compatible with BSD sockets
  - addressLength
    - This is the length of the address in bytes

©2023 Silicon Laboratories Inc. All rights reserved.

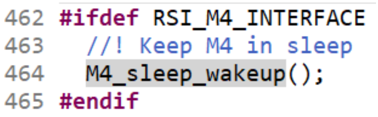
What is the code flow of this application?



```c
//! Enable Broadcast data filter
status = rsi_wlan_filter_broadcast(5000, 1, 1);
if (status != RSI_SUCCESS) {
  LOG_PRINT("\r\nBroadcast Data Filtering Failed with Error Code : 0x%lX\r\n", status);
  return status;
}
```



```c
#if ENABLE_POWER_SAVE
  //! Apply power save profile
  status = rsi_wlan_power_save_profile(PSP_MODE, PSP_TYPE);
  if (status != RSI_SUCCESS) {
    return status;
  }
#endif
```



```c
#ifdef RSI_M4_INTERFACE
  //! Keep M4 in sleep
  M4_sleep_wakeup();
#endif
```

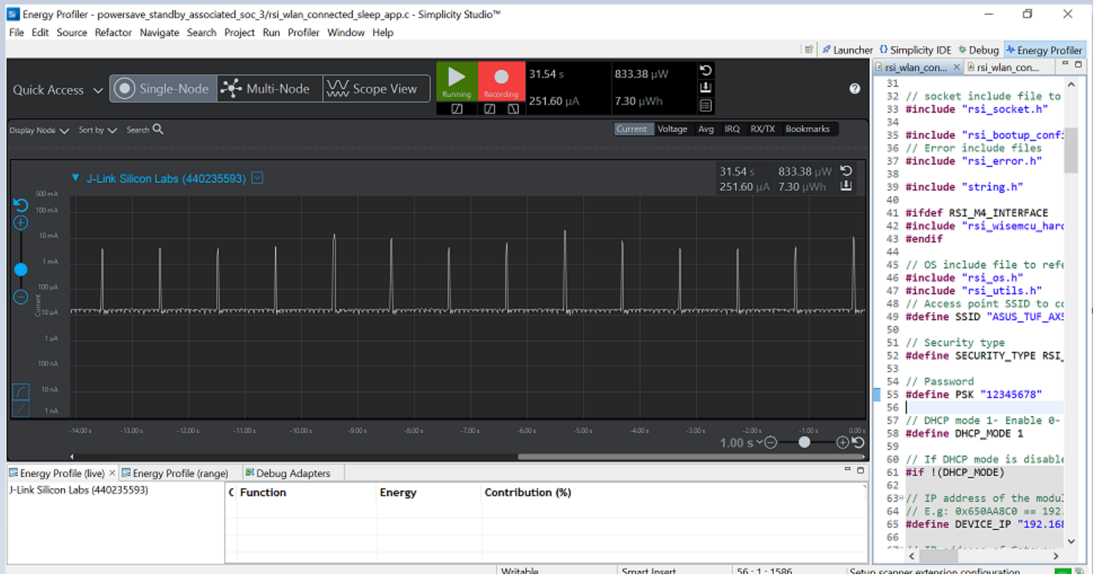
- **Enable Broadcast Data Filter**
  - Is done by rsi_wlan_filter_broadcast as in DTIM/Listen Interval code
- **Power Save with Connected Sleep**
  - Is done by rsi_wlan_power_save_profile API as in DTIM/Listen Interval code
- **M4 Sleep Wakeup**
  - Is done by M4_sleep_wakeup API as in DTIM/Listen Interval code

©2023 Silicon Laboratories Inc. All rights reserved.

w/ works with
BY SILICON LABS

Low Power Measurement of SiWx917 using AEM

Low Power Measurement with AEM – What is it?

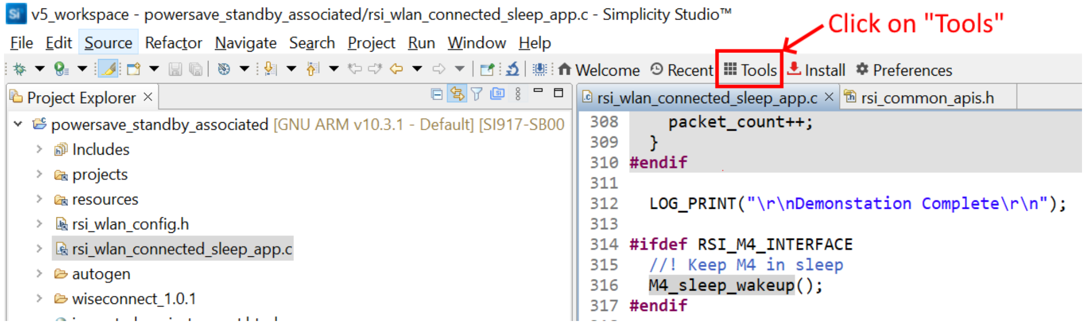


```
// socket include file to
#include "rsi_socket.h"

#include "rsi_bootup_confi
// Error include files
#include "rsi_error.h"

#include "string.h"

#ifdef RSI_M4_INTERFACE
#include "rsi_wisemcu_hard
#endif

// OS include file to refe
#include "rsi_os.h"
#include "rsi_utils.h"
// Access point SSID to co
#define SSID "ASUS_TUF_AX5
// Security type
#define SECURITY_TYPE RSI_

// Password
#define PSK "12345678"

// DHCP mode 1- Enable 0-
#define DHCP_MODE 1

// If DHCP mode is disable
#if !(DHCP_MODE)

// IP address of the modul
// E.g: 0x650AA8C0 == 192.
#define DEVICE_IP "192.168

```

- **Using AEM (Advanced Energy Monitoring) we will be able to measure the current consumption of an SiWx917 system**
- **AEM is a tool built into Simplicity Studio that allows for the real-time monitoring of the current consumed by a Silicon Labs system (SiWx917, EFM32, etc.)**
- **AEM Allows the user to do the following:**
  - Plot the system's current consumption
  - Obtain an average current consumption measurement
  - Zoom in and out to observe current consumption peaks
- **In order to perform an SiWx917 AEM current consumption measurement, a user will need the following:**
  - An Si917-PK6030A Pro Kit including Radio Board and 4002A WPK
  - A PC with Simplicity Studio release 5.6.4 or later

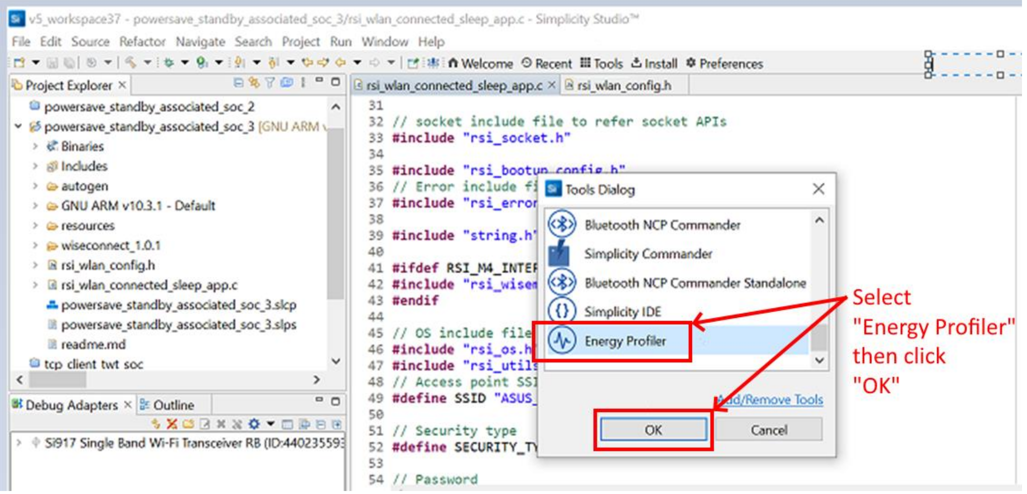
©2023 Silicon Laboratories Inc. All rights reserved.

Low Power Measurement with AEM – How to do it?


v5_workspace - powersave_standby_associated/rsi_wlan_connected_sleep_app.c - Simplicity Studio™
File Edit Source Refactor Navigate Search Project Run Window Help
Welcome
Recent
Tools
Install
Preferences
Project Explorer
powersave_standby_associated [GNU ARM v10.3.1 - Default] [SI917-SB00
Includes
projects
resources
rsi_wlan_config.h
rsi_wlan_connected_sleep_app.c
autogen
wiseconnect_1.0.1
rsi_wlan_connected_sleep_app.c
rsi_common_apis.h

```
    packet_count++;
  }
#endif

  LOG_PRINT("\r\nDemonstation Complete\r\n");

#ifdef RSI_M4_INTERFACE
  //! Keep M4 in sleep
  M4_sleep_wakeup();
#endif
```

Click on "Tools"

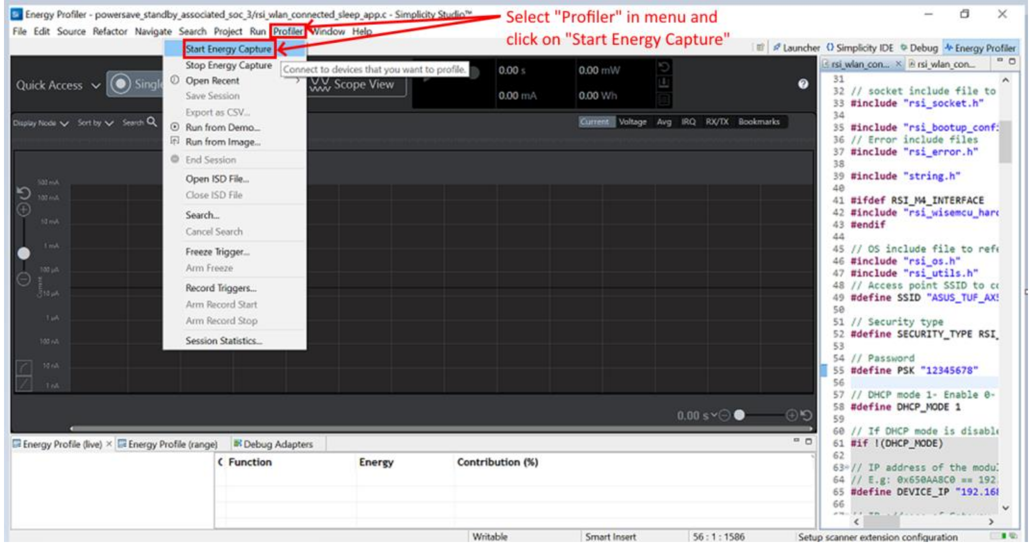

v5_workspace37 - powersave_standby_associated_soc_3/rsi_wlan_connected_sleep_app.c - Simplicity Studio™
File Edit Source Refactor Navigate Search Project Run Window Help
Welcome
Recent
Tools
Install
Preferences
Project Explorer
powersave_standby_associated_soc_2
powersave_standby_associated_soc_3 [GNU ARM v
Binaries
Includes
autogen
GNU ARM v10.3.1 - Default
resources
wiseconnect_1.0.1
rsi_wlan_config.h
rsi_wlan_connected_sleep_app.c
powersave_standby_associated_soc_3.slcp
powersave_standby_associated_soc_3.slps
readme.md
tcp client twt soc
Debug Adapters
Outline
Si917 Single Band Wi-Fi Transceiver RB (ID:440235593
rsi_wlan_connected_sleep_app.c
rsi_wlan_config.h
Tools Dialog
Bluetooth NCP Commander
Simplicity Commander
Bluetooth NCP Commander Standalone
Simplicity IDE
Energy Profiler
Add/Remove Tools
OK
Cancel

```

// socket include file to refer socket APIs
#include "rsi_socket.h"

#include "rsi_bootup_config.h"
// Error include fi
#include "rsi_error

#include "string.h"

#ifdef RSI_M4_INTER
#include "rsi_wisem
#endif

// OS include file
#include "rsi_os.h
#include "rsi_utils
// Access point SSI
#define SSID "ASUS_

// Security type
#define SECURITY_TY

// Password
```

Select "Energy Profiler" then click "OK"

- In simplicity Studio click on the tools icon and a dialog box will appear
- Click on the “Energy Profiler” icon on the dialog box

Low Power Measurement with AEM – How to do it?

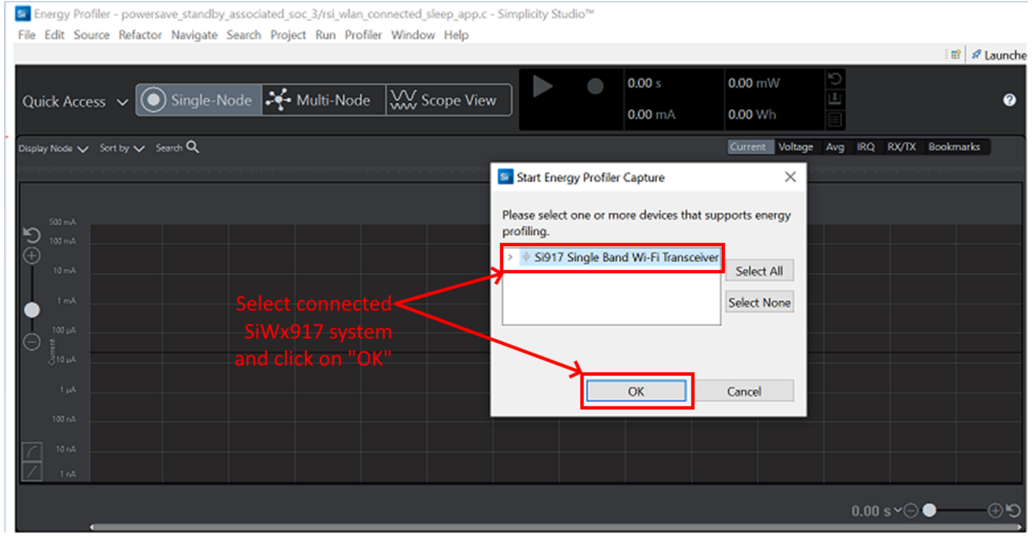


Energy Profiler - powersave_standby_associated_soc_3/rsi_wlan_connected_sleep_app.c - Simplicity Studio™

Select "Profiler" in menu and click on "Start Energy Capture"

Connect to devices that you want to profile.

```
// socket include file to
#include "rsi_socket.h"

#include "rsi_bootup_conf:
// Error include files
#include "rsi_error.h"

#include "string.h"

#ifdef RSI_M4_INTERFACE
#include "rsi_wisemcu_harc
#endif

// OS include file to refe
#include "rsi_os.h"
#include "rsi_utils.h"
// Access point SSID to cc
#define SSID "ASUS_TUF_AX!
 
// Security type
#define SECURITY_TYPE RSI_

// Password
#define PSK "12345678"

// DHCP mode 1- Enable 0-
#define DHCP_MODE 1

// If DHCP mode is disabl
#if !(DHCP_MODE)

// IP address of the modu
// E.g: 0x650AA8C0 == 192
#define DEVICE_IP "192.16
```



Energy Profiler - powersave_standby_associated_soc_3/rsi_wlan_connected_sleep_app.c - Simplicity Studio™

Start Energy Profiler Capture

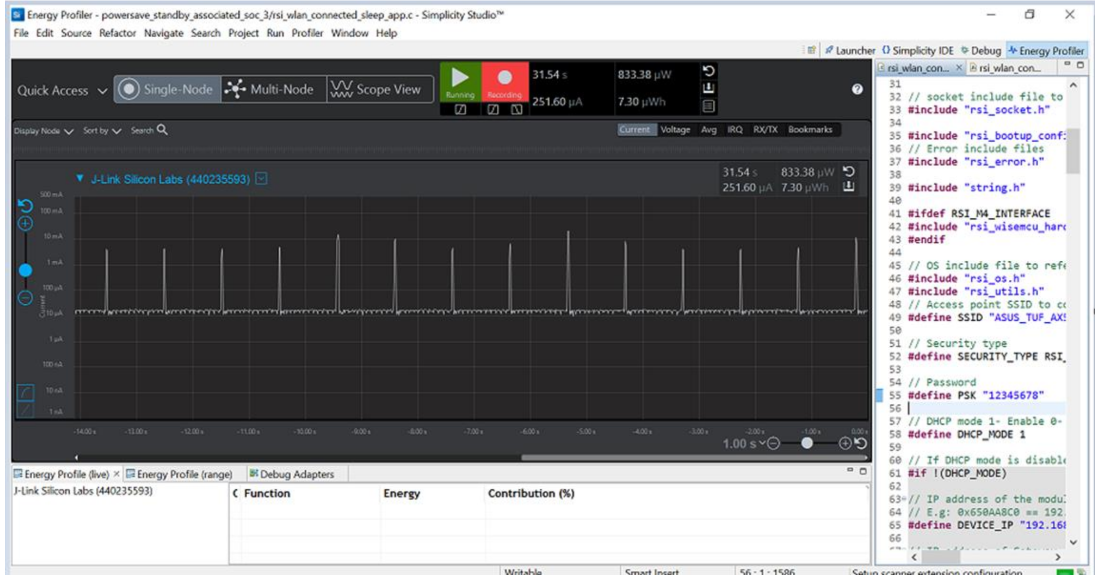
Please select one or more devices that supports energy profiling.

Si917 Single Band Wi-Fi Transceiver

Select connected SiWx917 system and click on "OK"

- The Energy Profiler tool will now open. Once on it, select the “Profiler” option on its menu and click “Start energy capture” as shown here.
- A pop-up window will now open, in it, select the connected SiWx917 device and click on OK.
- This will make the energy profiler to start a current consumption capture

Low Power Measurement with AEM – How to do it?



```
// socket include file to
#include "rsi_socket.h"

#include "rsi_bootup_conf:
// Error include files
#include "rsi_error.h"

#include "string.h"

#ifdef RSI_M4_INTERFACE
#include "rsi_wisemcu_harc
#endif

// OS include file to refe
#include "rsi_os.h"
#include "rsi_utils.h"
// Access point SSID to cc
#define SSID "ASUS_TUF_AX!

// Security type
#define SECURITY_TYPE RSI_

// Password
#define PSK "12345678"

// DHCP mode 1- Enable 0-
#define DHCP_MODE 1

// If DHCP mode is disable
#if !(DHCP_MODE)

// IP address of the modul
// E.g: 0x650AA8C0 == 192.
#define DEVICE_IP "192.16
```

- **After doing this, the Energy Profiler will display a real time current consumption plot on which you will be able to do the following:**
  - Obtain an average current consumption measurement
  - Zoom in and out to observe current consumption peaks

- **You can see an example of such a plot for an SiWx917 system in standby associated mode in this screen**

- **The next slide will show you a brief video demonstrating an AEM measurement on an SiWx917**

©2023 Silicon Laboratories Inc. All rights reserved.

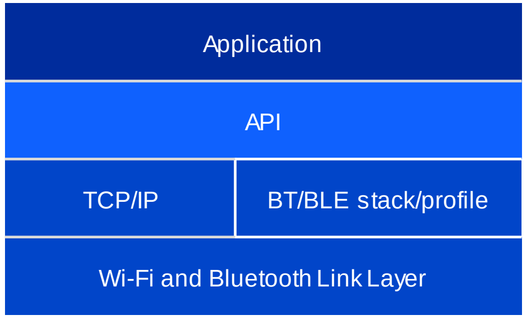
Silicon Labs' Wi-Fi Portfolio

works with
BY SILICON LABS

Silicon Labs - Complete Solution for Enabling Wi-Fi Products

SoCS AND MODULES

Industry leading Ultra Low Power Wi-Fi 4 and Wi-Fi 6 SoCs and pre-certified modules

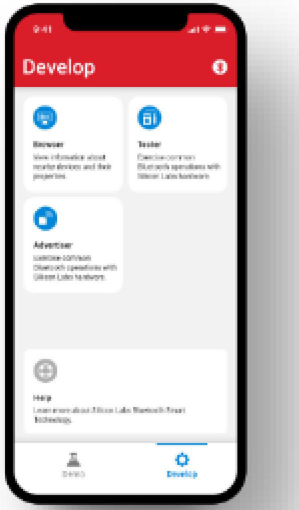


EMBEDDED SOFTWARE

Wi-Fi SDK with Integrated Wi-Fi, BT/BLE and IP networking stacks

DEVELOPMENT TOOLS

Evaluation Kit hardware and Studio software simplify development and speed time to market



MOBILE APPLICATIONS

EFR Connect for Wi-Fi Provisioning using BLE

©2023 Silicon Laboratories Inc. All rights reserved.

Q&A

works with
BY SILICON LABS

works with
BY SILICON LABS

Thank you!