Low-power system design with AIROC™ Wi-Fi & Bluetooth® combo chip and PSoC™ 6 MCU

About this document

Scope and purpose

AN227910 describes how to use the AIROC™ Wi-Fi & Bluetooth® combo chip and PSoC™ 6 MCU to design a low-power connectivity solution for Internet of Things (IoT) applications.

The technical description in this application note applies to other AIROC™ Wi-Fi & Bluetooth® combo chip and PSoC™ 6 MCU.

The 28-nm radio combined with a 40-nm PSoC™ 6 MCU enables an ultra-low-power platform for IoT applications. This application note provides an overview of low-power modes and features in the AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip. It also describes various techniques, such as host offload features, to optimize power consumption in the system, and assist with the low-power assistant (LPA) tool.

Intended audience

This document is intended for anyone using the AIROC™ Wi-Fi & Bluetooth® combo chip and PSoC™ 6 MCU to design a low-power connectivity solution for IoT applications.

Table of contents

Table of contents

Table of contents

1 Introduction

Infineon® AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip is a 28-nm, ultra-low-power device that supports single-stream, dual-band IEEE 802.11n-compliant and 802.11ac-friendly Wi-Fi MAC/baseband/radio and Bluetooth® 5.0 BR/EDR/LE. The WLAN section supports an SDIO interface to the host MCU (PSoC™ 6 MCU), and the Bluetooth® section supports a high-speed 4-wire UART interface to the host MCU.

PSoC™ 6 MCU bridges the gap between expensive, power-hungry application processors and low-performance microcontrollers (MCUs). The ultra-low-power PSoC™ 6 MCU architecture offers the processing performance needed by IoT devices, eliminating the tradeoffs between power and performance. The PSoC™ 62 series, built on an ultra-low-power 40-nm platform, complements the AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip, providing an ultra-low-power host interface to the Wi-Fi and Bluetooth® radio available in the AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip.

This application note discusses various low-power design techniques along with low-power features offered by the Infineon® AIROC™ Wi-Fi & Bluetooth® combo chip + PSoC™ 6 MCU platform and how to use them in your application. This application note requires a basic knowledge of the AIROC™ CYW43xxx Wi-Fi & Bluetooth® combo chip and PSoC™ 6 MCU architecture, and the ability to develop applications using ModusToolbox™ software.

If you are new to AIROC™ Wi-Fi & Bluetooth® combo chip, PSoC™ 6 MCU, or ModusToolbox™ software, see the Getting started with PSoC™ 6 MCU on ModusToolbox™ software.

2 Low-power overview

2.1 AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip power modes

AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip has been designed with the stringent power consumption requirements of battery-powered IoT devices in mind. Table 1 lists the power modes supported in CYW43012.

Table 1 Power modes in AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip

Power mode	Description
Active	• All WLAN blocks in the AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip are powered up and fully functional with active carrier sensing and frame transmission and reception. • All required regulators are enabled and put in the most efficient mode based on the load current. • Clock speeds are dynamically adjusted by the PMU sequencer.
Deep Sleep (DS0)	• The radio, analog domains, and most of the linear regulators are powered down. The rest of the AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip remains powered up in an idle/retained state. • All main clocks (PLL, crystal oscillator, or TCXO) are shut down to reduce active power to the minimum level. The 32.768-kHz LPO clock is available only for the PMU sequencer. This is necessary to allow the PMU sequencer to wake up the chip and transition to active mode. • Complete RAM needed for WLAN firmware is retained.
Power down (Off)	• AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip is effectively powered OFF by shutting down all internal regulators. The chip is brought out of this mode by external logic reenabling the internal regulators.

AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip includes an advanced WLAN power management unit (PMU) sequencer, which provides significant power savings. It puts the AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip device into various power management states appropriate to the current environment and activities performed.

The PMU enables and disables internal regulators, switches, and other blocks based on the computation of the required resources and a table that describes the relationship between resources and the time needed to enable and disable them. Configurable, free-running counters (running at 32.768 kHz LPO clock) in the PMU sequencer are used to turn ON/turn OFF individual regulators and power switches. Clock speeds are dynamically changed (or gated altogether) for the current mode. Slower clock speeds are used wherever possible. See the AIROC™ Wi-Fi & Bluetooth® combo chip for details on the PMU.

2.2 PSoC™ 6 MCU power modes

PSoC™ 6 MCU features seven power modes split into system modes that affect the whole device and standard Arm® CPU modes that affect only one CPU. The system power modes are Low-Power (LP), Ultra-Low-Power (ULP), Deep Sleep, and Hibernate. The Arm® CPU power modes are Active, Sleep, and Deep Sleep, and are available in system LP and ULP power modes. See AN219528 for details on PSoC™ 6 MCU low-power modes and power reduction techniques. Table 2 summarizes PSoC™ 6 MCU power modes and provides a simplified version for representing power mode transitions in a PSoC™ 6 MCU and AIROC™ Wi-Fi & Bluetooth® combo chip design (Figure 1).

Table 2 Power modes in PSoC™ 6 MCU

Power mode	Description	Simplified power mode
System LP	• All resources are available with maximum power and speed. • All CPU power modes supported.	Active mode
System ULP	• All blocks are available, but core voltage is lowered resulting in reduced high-frequency clock frequencies. • All CPU power modes supported.	
CPU Active	• Normal CPU code execution • Available in system LP and ULP power modes	
CPU Sleep	• CPU halts code execution. • Available in system LP and ULP power modes	Sleep mode
CPU Deep Sleep	• CPU halts code execution. • Requests system Deep Sleep entry • Available in system LP and ULP power modes	Deep Sleep
System Deep Sleep	• Occurs when all CPUs are in CPU-Deep Sleep • CPUs, most peripherals, and high-frequency clocks are OFF. • Low-frequency clock is ON. • Low-power analog and some digital peripherals are available for operation and as wake-up sources. • SRAM is retained.	
System Hibernate	• CPUs and clocks are OFF. • GPIO output states are frozen. • Low-power comparator, RTC alarm, and dedicated WAKEUP pins are available to wake up the system. • Backup domain is available.	Hibernate

2.3 AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip power-related hardware signals

AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip has six hardware signals that provide the power control interface to the host (PSoC™ 6 MCU).

Table 3 Hardware signals in AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip

Signal	Description
WL_REG_ON	This signal is used by the PMU (with BT_REG_ON) to power up the WLAN section. It is OR-gated with the BT_REG_ON input to control internal AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip regulators. When this pin is HIGH, the regulators are enabled, and the WLAN section is out of reset. When this pin is LOW, the WLAN section is in reset. If BT_REG_ON and WL_REG_ON are both LOW, the regulators are disabled.
BT_REG_ON	This signal is used by the PMU (with WL_REG_ON) to decide whether to power down internal AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip regulators. When this pin is HIGH, the regulators are enabled, and the Bluetooth® section is out of reset. When this pin is LOW, the Bluetooth® section is in reset. If BT_REG_ON and WL_REG_ON are LOW, the regulators will be disabled.
BT_DEV_WAKE	Bluetooth® device wake-up: Signal from the host (PSoC™ 6 MCU) to AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip indicating that the host requires attention like sending data to the Bluetooth® controller. • Asserted: The Bluetooth® device must wake up or remain awake. • Deasserted: The Bluetooth® device may sleep when all other sleep criteria are met. The polarity of this signal is software-configurable and can be asserted as HIGH or LOW. Default is asserted HIGH.
BT_HOST_WAKE	Host wake-up signal from the AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip Bluetooth® controller to the host indicating that the AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip Bluetooth® controller has data for the host. • Asserted: host device must wake up or remain awake. • Deasserted: host device may sleep when all other sleep criteria are met. The polarity of this signal is software-configurable and can be asserted as HIGH or LOW. Default is asserted HIGH.
WL_HOST_WAKE	Host wake-up signal from the AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip WLAN device to the host indicating that the AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip WLAN device has data for the host to process. This signal is an active HIGH signal. • HIGH: host device must wake up or remain awake. • LOW: host device may sleep when sleep criteria are met.
WL_DEV_WAKE	WLAN device wake-up signal. This signal is not used in current designs and SDIO-based wake-up is used to wake up the WLAN device.

2.4 Power mode transition

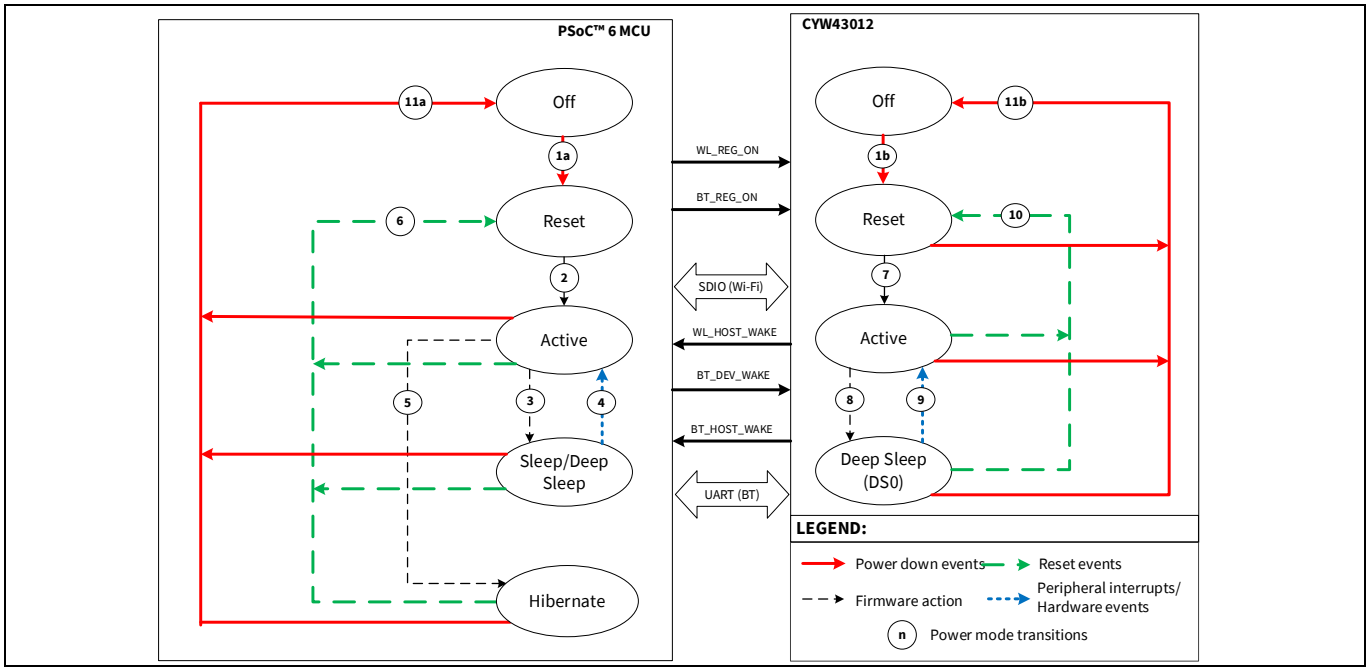

Figure 1 Simplified power mode transitions in "PSoC™ 6 MCU[a] + AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip" design

Transition	Description	Trigger
1a	PSoC™ OFF to reset transition	Power supply applied to PSoC™ 6 MCU.
1b	AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip OFF to reset transition	Power supply applied to AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip.
2	PSoC™ reset to Active transition	Firmware (boot/startup) transitions from reset to Active mode.
3	PSoC™ Active to Sleep or Deep Sleep transition	Firmware (OS idle task) transitions from active to Sleep or Deep Sleep mode.
4	PSoC™ Sleep or Deep Sleep to active transition	Wake-up (interrupt) from Sleep/Deep Sleep; typical sources in an AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip design: • WL_HOST_WAKE (Deep Sleep or Sleep) • BT_HOST_WAKE (Deep Sleep or Sleep) • SDIO and UART interrupt (Sleep) • Low-power timer interrupts
5	PSoC™ active to Hibernate transition	Firmware (enter hibernate) transitions from active to Hibernate mode
6	PSoC™ transitions to reset state	Any reset event including hibernate wake-up, external reset, and so on.

Low-power overview

Transition	Description	Trigger
7	AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip reset to active transition	External events transition AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip out of reset; either BT_REG_ON or WL_REG_ON pulled HIGH by PSoC™ 6 MCU.
8	AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip active to Deep Sleep transition	AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip enters DS0 mode; managed by on-chip PMU and requires enabling power save in the device
9	AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip Deep Sleep to active transition	External events or PMU transitions AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip to active mode; possible PSoC™ 6 MCU actions: • BT_DEV_WAKE • SDIO interrupt
10	AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip transitions to reset state	External events transition AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip to reset state; both WL_REG_ON and BT_REG_ON are pulled LOW by PSoC™ 6 MCU.
11a	PSoC™ transitions to OFF state	Power supply removed from PSoC™ 6 MCU (Power Down)
11b	AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip transitions to OFF state	Power supply removed from AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip (Power Down)

Note: [a] *PSoC™ 6 MCU power modes are represented in a simplified form for ease of depicting the transitions. See Table 2 for details.*

3 WLAN power optimization techniques

3.1 IEEE 802.11 (Wi-Fi) power saving

Topics in this section are partly based on information contained in the IEEE 802.11 specification document. The 802.11 standard includes several parameters that allow stations to save power, although power saving is accomplished at the expense of throughput or latency to the station.

This section only provides an overview of power-saving methods and associated terms from the IEEE specification. If you are already familiar with the topic and interested in the implementation, see

Wi-Fi power save implementation.

3.1.1 Beacon interval

Beacon frames are periodic frames broadcast by the Wi-Fi access points (APs) to communicate throughout the serviced area the characteristics of the connection offered to the Wi-Fi stations (STAs). This information is used by STAs trying to connect to the network as well as STAs already associated with the Basic Service Set (BSS). The period at which beacon frames are transmitted is called “Target Beacon Transmission Time” (TBTT) or simply “Beacon Interval”. Even though the beacon interval is configurable for a given AP, it is typically set to 102.4 ms (100-time units, where the 1-time unit is 1024 µs per 802.11 specifications). The beacon interval information is available as part of the beacon frame itself. See the 802.11 specification for details. In addition, each beacon includes a Traffic Indication Map (TIM) that contains information regarding packets buffered for STAs. It is the responsibility of the STAs to read the TIM and retrieve the packets destined for them.

3.1.2 Listen interval

To save power, STAs sleep by powering down most of the Wi-Fi subsystem. While the STAs are asleep, APs buffer frames for them and indicate the availability through beacon TIMs. The sleeping STAs periodically wake up to listen to traffic announcements (TIMs) and determine whether the AP has any buffered frames. When STAs associate with an AP, part of the data in the Association Request frame is the listen interval. The listen interval indicates to the AP the frequency at which a power save station will wake up to listen to beacon frames. An AP may use the listen interval as a guide to determine the duration it should retain buffered frames for a power save station. Larger listen intervals require more AP memory for frame buffering.

APs generally do not consider the listen interval requested by an STA. If an STA sets the listen interval to a value, there is no guarantee that the AP buffers all the packets received for the station while the station is asleep. In addition, if the AP supports Delivery Traffic Indication Map (DTIM), that setting may override the listen interval requested by STAs.

Most APs enforce an association timeout on stations, that is, if the AP has not received a frame from a station within the association timeout (usually 60 seconds) or the station has not returned an ACK to a keep-alive frame, the station is disassociated. The association timeout cannot be negotiated by the station.

3.1.3 DTIM period

The DTIM period is a parameter associated with an infrastructure network and is advertised in the AP Beacon frame. The polled approach using standard TIMs is not suitable for multicast and broadcast frames, because it takes too much capacity to transmit multicast and broadcast frames multiple times. Instead of the polled approach, broadcast and multicast frames are delivered after every DTIM interval.

Increasing the DTIM allows stations to conserve power more effectively at the cost of buffer space in the AP. It also delays the reception of multicast and broadcast frames by all stations, including stations in active mode.

Note that the listen interval is recommended to be equal to (or less than) the DTIM period for the STA to receive all unicast, multicast, and broadcast packets. This ensures an optimal power and latency tradeoff.

Usually, the DTIM period is set as the number of beacon intervals in the AP. The default DTIM period for most APs is either DTIM=1 (DTIM every beacon), DTIM=3 (DTIM every third beacon), or DTIM=10. In the case of DTIM=3, the station needs to only wake from low-power mode to receive every third beacon and any ensuing queued broadcast or multicast traffic.

Figure 2 explains the beacon interval, listen interval, and DTIM period in an infrastructure network.

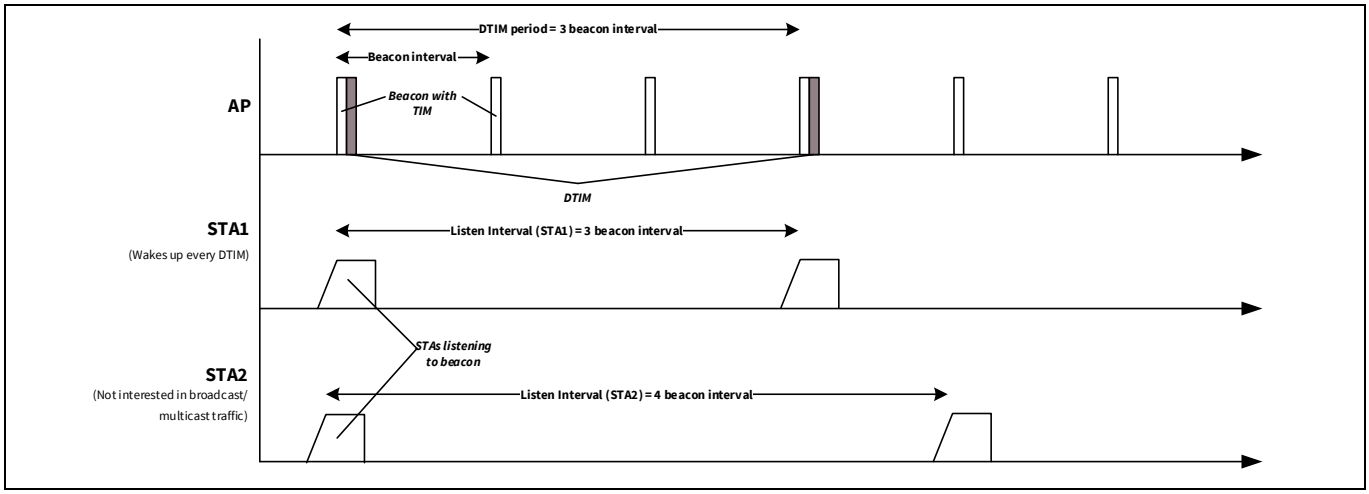

Figure 2 802.11 infrastructure network operation

3.1.4 Power-saving methods

Although the 802.11 transmitted power consumption is at least five times higher than the received power consumption, even for medium transmit-duty-cycle applications, much of the energy in a battery-powered Wi-Fi station is consumed by the receiver. Unless power save techniques are used, the 802.11 receiver may be powered ON for significant periods of time while the station waits for network clients to respond to requests. It does not take very long for a 130-mW receiver power consumption to discharge a battery.

Wi-Fi STAs use different mechanisms to save power. STAs with low duty cycle and long battery life requirements, Wi-Fi sensors, for example, may use the standard 802.11 Power Save Poll (PS-Poll) mechanism. STAs requiring higher throughput, wireless speakers, for example, may consider using a non-standard 802.11 power save mechanism. These methods are described in this section.

Note: STAs requiring a more fine-grained power save mechanism to differentiate power save for various traffic priorities may consider using unscheduled automatic power save delivery (U-APSD) or Wi-Fi multimedia (WMM) power save. AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip currently does not support WMM power save.

3.1.4.1 Power save poll

Power save poll suits for STAs that primarily transmit data to the Wi-Fi network at low-duty cycles. The PS-poll mechanism works as follows:

1. STA sends a frame to the AP with the Power Management bit set and then goes to sleep.
2. AP notes that the STA has gone to sleep, and buffers frames for the STA.
3. STA wakes up periodically to check the AP beacon frame for an indication of buffered frames. The wake period depends on the configured listen interval and whether the STA is configured to receive group-addressed frames from a beacon containing a DTIM.
4. If the AP indicates that it has buffered frames addressed to the STA, the STA sends a PS-Poll frame to the AP.
5. The AP sends a frame to the STA and sets the ‘More Data’ bit if additional frames are buffered.
6. The STA might send another PS-Poll frame to retrieve additional buffered frames (if the More Data bit was set) or return to sleep.

7. When the STA needs to transmit data, it can wake up and transmit data anytime without waiting for the beacon interval as the AP is active all the time.

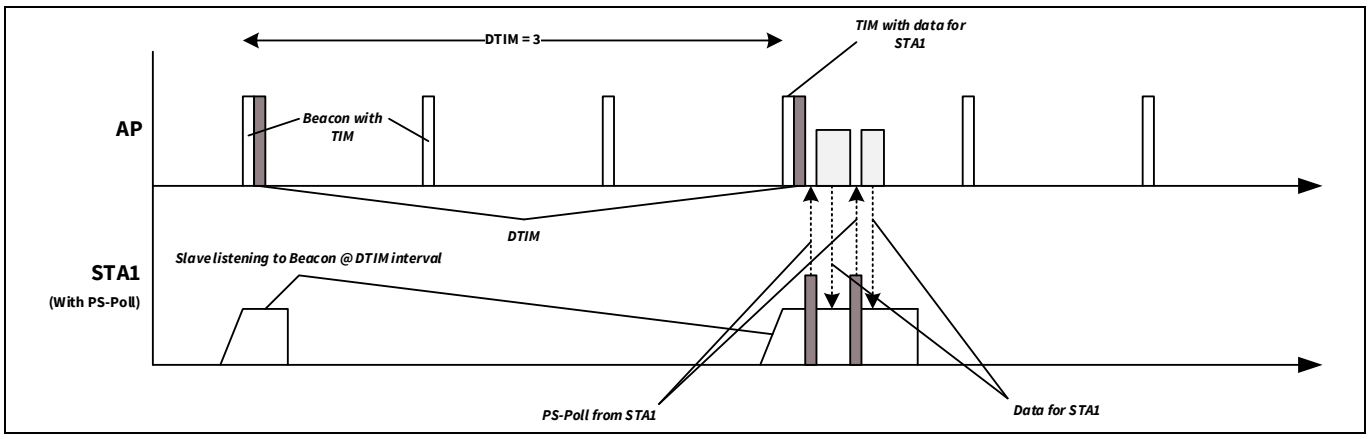

Figure 3 Power save poll

3.1.4.2 802.11 power save without poll

A non-standard mechanism known broadly as 802.11 power save without poll (PS-non-poll) enables STAs to use 802.11 power saving based on a 'trigger' frame as follows:

1. STA sends a frame to the AP with the 'Power Management' bit set and then goes to sleep.
2. AP notes that the STA has gone to sleep and buffers frames for the STA.
3. STA wakes up periodically. The STA may (or may not) first check the AP beacon frame for an indication of buffered frames before sending any frames to the AP.
4. The STA sends a Null Function data frame, or a data frame if available, to the AP with the Power Management bit cleared.
5. The AP notes that the STA is awake and sends buffered frames.
6. STA receives the frames, waits for a timeout period to receive additional frames (if available), and then returns to sleep as described in Step 1.

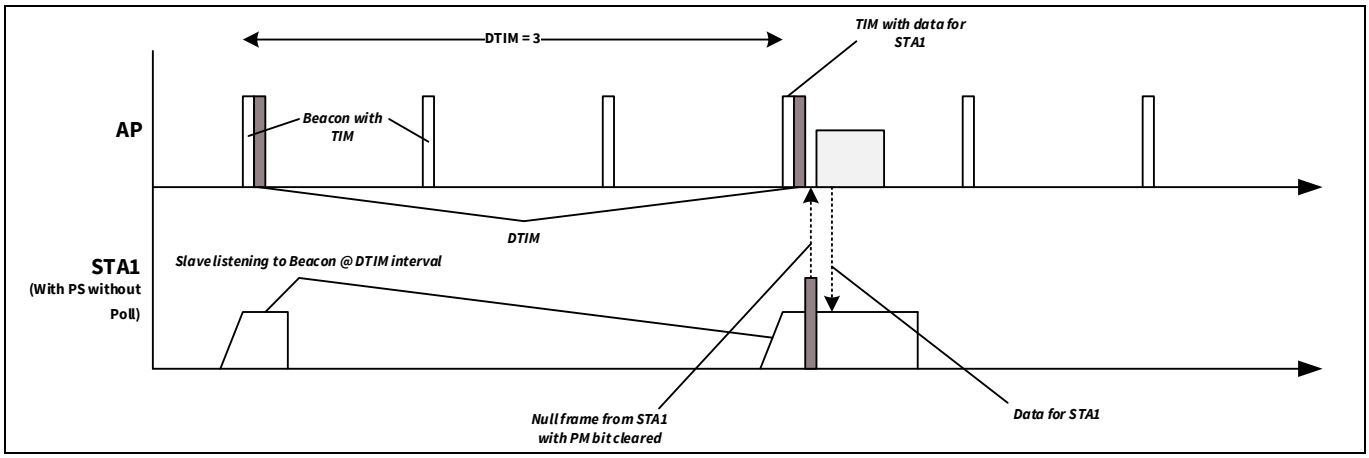

Figure 4 Power save without poll

3.1.4.3 Association timeout limit

If an STA does not expect to receive directed frames from the AP asynchronously, and it is not interested in receiving broadcast or multicast traffic, then further power savings may be achieved.

Because APs generally have an association timeout limit with a default value of 60 seconds, an STA that uses power save must wake up before the association timeout expires and send or receive a directly addressed frame to inform the AP that the STA is still associated. Failure to comply results in the AP de-authenticating the STA. Some APs allow the association timeout limit to be set to a value higher than 60 seconds, but the higher limit is not signaled to the station and must be manually configured.

3.1.5 802.11ac-friendly features

AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip provides 802.11ac friendliness by supporting 256-QAM (for 20-MHz channels in the 5-GHz band) enabling data rates of up to 78 Mbps with 802.11ac APs. This enables superior throughput and reduced power consumption in the device by sending/receiving data and then going back to sleep faster. In addition, it supports 802.11ac's ‘explicit beamforming’ feature to offer significantly higher throughputs than 802.11n chipsets at any given range.

3.2 Wi-Fi power save implementation

3.2.1 Wi-Fi Host Driver (WHD) power save interface

The power save implementation is provided as part of the WHD library. In addition, platforms such as FreeRTOS implement their power save mechanisms on top of the WHD to provide users with seamless low-power integration.

Table 4 describes the low-level functions available as part of WHD providing various power-save features. Refer to the WHD API reference guide for details.

Table 4 WHD power save APIs

WHD API	Description
`whd_wifi_enable_powersave()`	Enables PS-poll mode in the device (see Power save poll)
`whd_wifi_enable_powersave_with_throughput()`	Enables 802.11 PS-non-poll mode in the device (see 802.11 power save without poll). Use this mode when it is important to maintain throughput.
`whd_wifi_disable_powersave()`	Disables power save mode in the device; by default, power save is enabled in the WLAN device.
`whd_wifi_set_listen_interval()`	Sets the number of beacons to be skipped while the Wi-Fi chip is sleeping. Only works when Wi-Fi power save is enabled and the DTIM interval broadcasted by the AP is zero (that is, no DTIM in the network).
`whd_wifi_get_listen_interval()`	Gets the current value of all beacon listen interval variables

Note that power save with throughput is enabled in the WLAN device by default. The enable/disable power save APIs can be called to change or disable the power save option in the WLAN. The WLAN automatically manages the entry and exit of DS0 after the power save is enabled. The mtb-example-wifi-wlan-lowpower code example demonstrates the use of WHD power save APIs.

An example implementation of a power save handler using WHD APIs in ModusToolbox™ is as follows:

Code Listing 1 Example: Power save handler using WHD APIs in ModusToolbox™

```
cy_rslt_t wlan_powersave_handler(struct netif *wifi, enum wlan_powersave_mode_t
mode)
{
    cy_rslt_t result = CY_RSLT_SUCCESS;
    whd_interface_t ifp;
    whd_security_t security_param;
    whd_bss_info_t ap_info;

    if (wifi->flags & NETIF_FLAG_UP)
    {
        /* Get the instance of the WLAN interface.*/
```

WLAN power optimization techniques

Code Listing 1 Example: Power save handler using WHD APIs in ModusToolbox™

```
        ifp = (whd_interface_t)wifi->state;

        /* Obtain network parameters configured in the AP.*/
        result = whd_wifi_get_ap_info(ifp, &ap_info, &security_param);

        if (CY_RSLT_SUCCESS == result)
        {
            APP_INFO(("Beacon period = %d, DTIM period = %d\n",
                      ap_info.beacon_period, ap_info.dtim_period));
        }
        else
        {
            ERR_INFO(("Failed to get AP info.\n"));
        }

        /* Configure power-save mode of the WLAN device.*/
        switch (mode)
        {
        case POWERSAVE_WITHOUT_THROUGHPUT:
            result = whd_wifi_enable_powersave(ifp);

            if (CY_RSLT_SUCCESS != result)
            {
                ERR_INFO(("Failed to enable PM1 mode\n"));
            }

            break;
        case POWERSAVE_WITH_THROUGHPUT:
            result = whd_wifi_enable_powersave_with_throughput(ifp,
RETURN_TO_SLEEP_MS);

            if (CY_RSLT_SUCCESS != result)
            {
                ERR_INFO(("Failed to enable PM2 mode\n"));
            }

            break;
        case POWERSAVE_DISABLED:
            result = whd_wifi_disable_powersave(ifp);
```

Code Listing 1 Example: Power save handler using WHD APIs in ModusToolbox™

```
            if (CY_RSLT_SUCCESS != result)
            {
                ERR_INFO(("Failed to disable powersave\n"));
            }

            break;
        default:
            APP_INFO(("Unknown mode\n"));
            break;
        }
    }
    else
    {
        ERR_INFO(("Wi-Fi interface is not powered on. Failed to configure power-
save mode\n"));
        result = CY_RSLT_TYPE_ERROR;
    }

    return result;
}
```

3.2.2 Cyclic timers in network stack

Platforms that implement network stacks, such as lightweight TCP/IP stack (lwIP), run periodic timers for various network-related activities. These timers are used as ticks for network activity such as address resolution protocol (ARP) cache table expiry and dynamic host configuration protocol (DHCP) lease renewal timeouts. The period of these timers can vary from 100 ms to 60 s. The host waking up periodically to service these timers is required for proper functioning. However, if the application wants to suspend these timers when it does not anticipate any network timeout activity and for a longer stay in low-power modes, there is an option to suspend and resume the network stack.

This feature is provided as part of the low-power assistant middleware helper files. The `wait_net_suspend()` API, available in the *lpa/helpers/net_activity/network_activity_handler.h* file, lets you suspend the network stack for a predefined duration or until an activity is detected, if the network remains inactive within a given time window. This API function lets you safely suspend the network and resume it whenever any network activity gets detected. Figure 5 shows the `wait_net_suspend()` operation in different scenarios with network suspend duration of 5000 ms, network inactivity monitor window of 150 ms, and network inactivity duration of 100 ms.

Scenario 1: `wait_net_suspend()` detects no network activity for more than 100 ms within the 150-ms window and suspends the network. No network activity is detected for the network suspend duration of 5000 ms. Therefore, `wait_net_suspend()` resumes the network stack after 5000 ms.

WLAN power optimization techniques

Scenario 2: `wait_net_suspend()` detects network activity in the 150-ms window and inactivity is less than 100 ms. `wait_net_suspend()` does not suspend the network stack.

Scenario 3: `wait_net_suspend()` detects no network activity for more than 100 ms within the 150-ms window and suspends the network. Network activity is detected before the expiry of 5000 ms duration. Therefore, `wait_net_suspend()` resumes the network stack immediately.

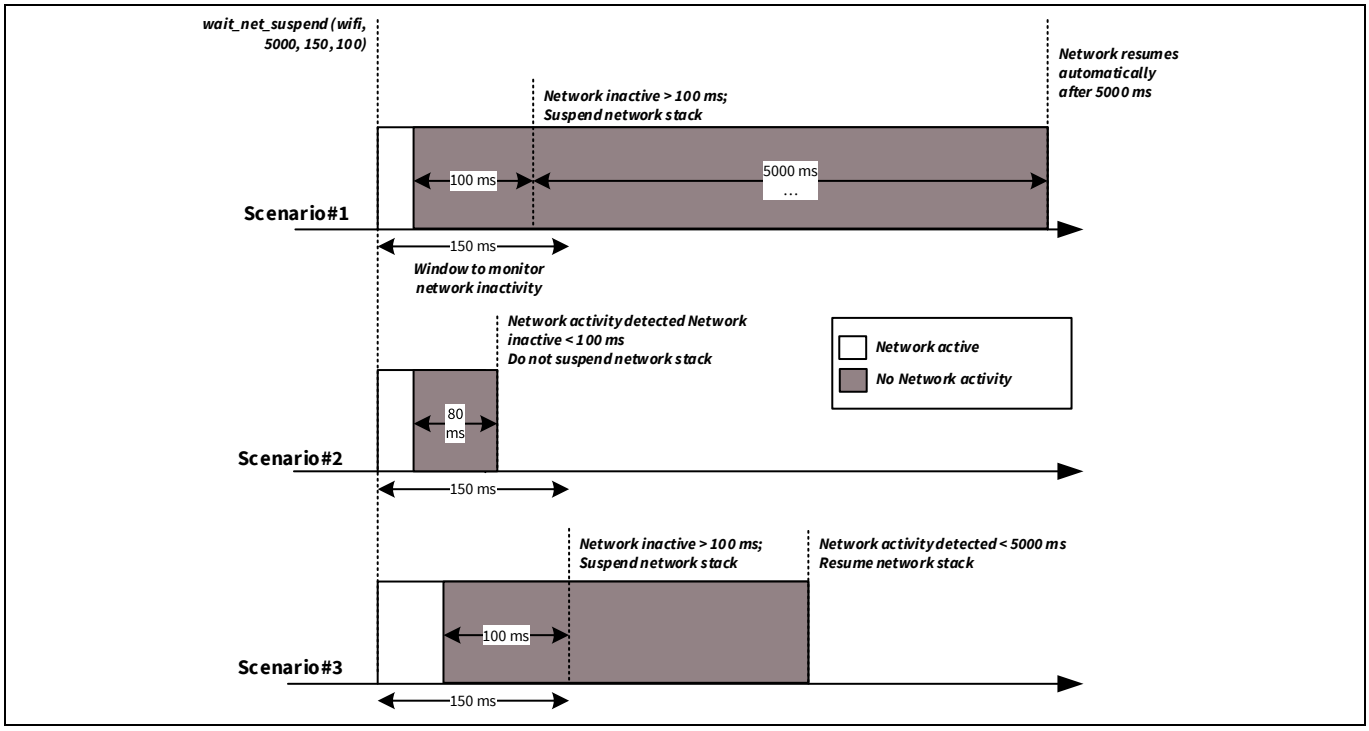

Figure 5 `wait_net_suspend()` operation

If you are using ModusToolbox™ software, see the mtb-example-wifi-wlan-lowpower code example for details on using the network suspend feature safely.

See lpa middleware documentation for additional details on these timers.

3.3 Host offloads to WLAN device

The AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip has an Arm® Cortex®-M3 core that performs all the WLAN device activities. It supports various offloads that execute certain functionalities on behalf of the host without interrupting the host state.

Host offloads play a key role in determining host power consumption because offloads let the host go into Deep Sleep for extended periods of time while the WLAN device handles tasks such as 802.11 roaming, ARP, IPv6 neighbor resolution, key rotation, and TCP keepalives on behalf of the host. In addition, these offloads free the host CPU for other, more powerful tasks such as audio or sensor data processing. This in turn improves the overall system efficiency and power.

The following sections describe offloads supported by AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip. The offloads can be enabled using the Infineon® low-power assistant middleware. Using the LPA middleware in ModusToolbox™ is explained in Low-power assistant (LPA).

3.3.1 ARP offload

ARP is a data link layer protocol for mapping an IP address to a physical MAC address of a device. Such mapping is necessary for any WLAN device to keep updated on [IP:MAC] details of all nearby devices in its network. Using the ARP protocol, a device can detect duplicate IP addresses in a network. This is also useful in notifying peer devices of an IP address change.

Because ARP request and response packets are a common activity in any WLAN network, the host device usually gets flooded by many requests; the host needs to remain active to respond to such requests, especially in a crowded network. ARP requests packets from a peer and usually wakes the host if it is sleeping. The ARP offloading feature in the AIROC™ Wi-Fi & Bluetooth® combo chip handles responding to ARP requests from a peer so that these requests will not disturb the host. The AIROC™ Wi-Fi & Bluetooth® combo chip device holds a cache (of eight entries); the host is woken up only if there is a cache miss as shown in Figure 6.

In addition to auto-replying to peers, AIROC™ Wi-Fi & Bluetooth® combo chip can use the cache to reply to the host (PSoC™ 6 MCU) as well when the host sends an ARP request to the network. This host auto-reply feature works similar to a Peer auto-reply, where a cache miss is forwarded to the network and a cache hit returns an ARP response immediately to the host. This feature is complemented by the ability of the AIROC™ Wi-Fi & Bluetooth® combo chip device to snoop ARP info from host-network communication, that is, whenever a host sends an ARP response packet over to the network, the [IP:MAC] information is cached into the AIROC™ Wi-Fi & Bluetooth® combo chip ARP cache table.

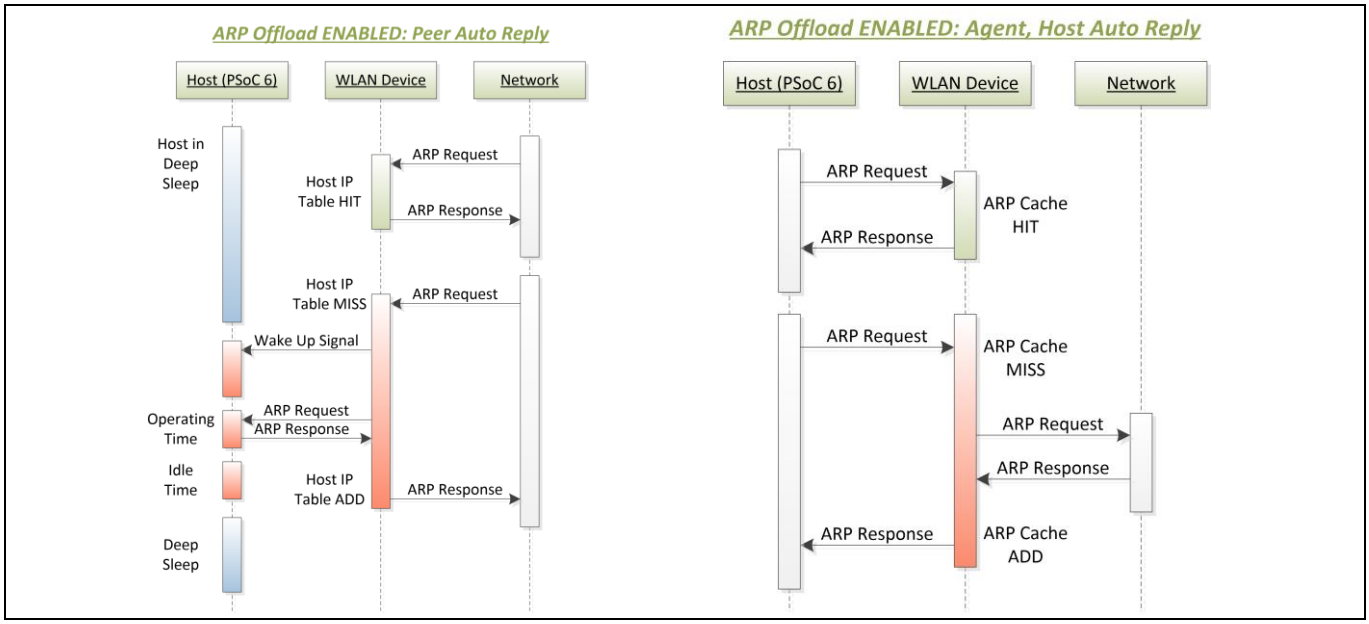

Figure 6 ARP offload feature

3.3.2 Packet filter offload

Packet filters allow the host to limit the types of packets that get passed up to the host processor from the WLAN device. This is useful to keep out unwanted packets from the network that might otherwise wake the host out of a power-saving Deep Sleep mode or prevent it from entering Deep Sleep mode.

In general, whenever a WLAN packet is destined for the host, the WLAN device must awaken the host (if it is asleep) so that the host can retrieve the packet for processing. Often, the host network stack processes the packet only to discover that the packet should be thrown away because it is not needed such as when the

WLAN power optimization techniques

packet is destined for a port or service that is not being used. Packet filters allow these types of packets to be filtered and discarded by the WLAN processor itself so that the host is not interrupted.

In short, packet filters are useful when:

- You want to keep the host processor in Deep Sleep for as long as possible (or enter Deep Sleep as soon as possible) by filtering unwanted traffic.
- You want to keep the host processor involvement to a minimum by filtering and passing only the wanted traffic to it.

Filters can be configured to be active either when the host processor is active or asleep (or both). Multiple filters may be configured to operate simultaneously. One set can be functioning while awake and a separate set can function while sleeping and can be used to wake the host. In addition, filters can be configured to either discard or keep (send to host). When configured to discard, only the specified packets are discarded, while any others not specifically filtered are passed to the host. Conversely, when configured to keep packets, only the specified packets are passed to the host, and the rest are discarded.

A typical model would be to use only 'keep' filters, where the application is aware of all the types of packets it needs to process. In other words, use 'keep' filters to specify the complete list of packet types the host is interested in and discard the rest. This is preferred as it is much simpler to list the packets the host wants to receive versus creating a complete list of packets it does not want. When using keep filters, care must be taken to allow enough packets through for networking protocols to work properly.

For example, while awake, the processor must be able to join a network, get a DHCP address, run ARP, and possibly share network keys. Not creating enough keep filters to allow these types of packets through will prevent the host from even joining the network. A reasonable minimal set of keeping filters should include:

- ARP (allow ARP packets)
- IEEE 802.1X (allow security packets)
- DHCP (UDP port 68)
- DNS (UDP port 53)

There is no minimum set of filters required when enabling discard filters. However, it should be noted that all the filter types in the offload must be either keep or discard and not a mix of both. This is because the complementary nature of the keep and discard filters will result in unexpected behavior.

Three types of packet filters are supported in the AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip based on a standard network stack as seen in Figure 7, which shows the relationships between stack layers, protocols, and packet filters.

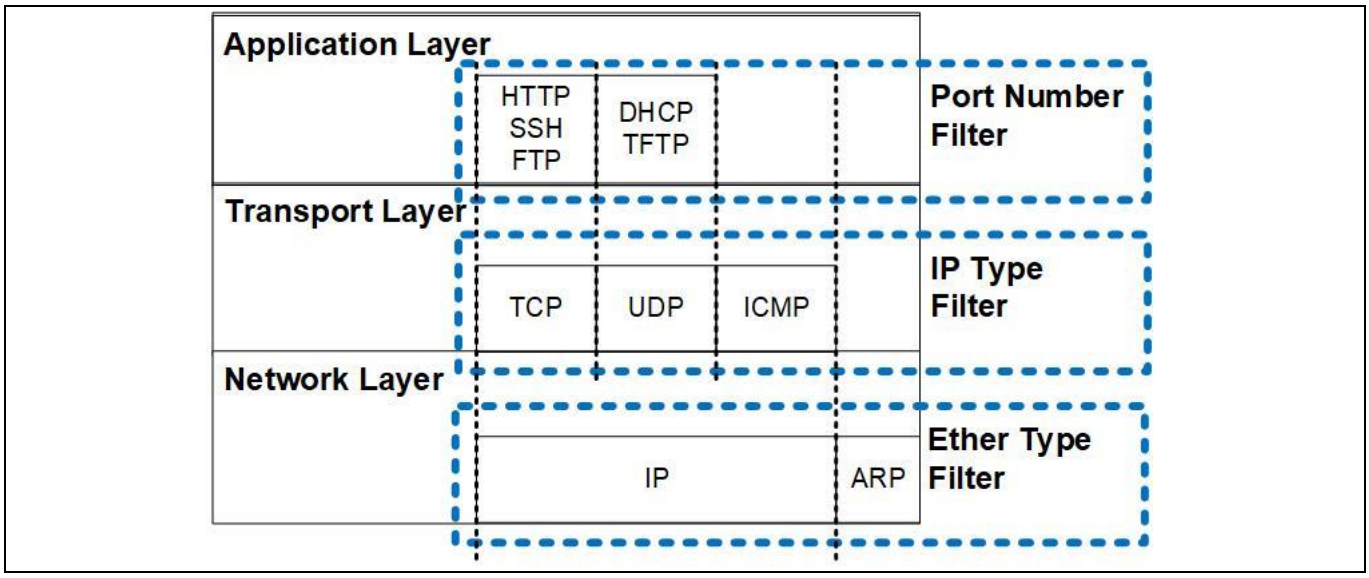

Figure 7 Packet filters

Note: In the AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip, the number of discard packet filter offload is limited to one, that is, only one discard packet filter can be enabled in the design; enabling more than one discard filter lets all packets pass through (no filter).

3.3.2.1 Network layer or EtherType filter

The EtherType filter filters packets based on a 16-bit EtherType field present in Ethernet packets on the network layer (see Internet Assigned Numbers Authority (IANA)). The most commonly used protocols (and most useful filters) are:

- IP (EtherType = 0x800)
- ARP (EtherType = 0x806)
- IEEE 802.1X or EAP over LAN (EtherType = 0x888E)

Filtering on IP EtherType would match all IP packets coming from the network. This is a very coarse filter and will include all ICMP, TCP, and UDP packets as shown in Figure 7. Filtering on ARP or IEEE 802.1X is finer; it will only match the respective packets. Filtering all IP packets will have an enormous impact due to the substantial number of packets it will match and is generally not recommended for typical usage. Valid EtherType filters consist of a 16-bit number greater or equal to 0x800.

3.3.2.2 Transport layer/IP protocols

The next layer up the stack is the Transport layer, which consists of various IP-based protocols such as TCP, UDP, and ICMP. Discussions of the protocols themselves are outside the scope of the application note but are widely available and a list of protocol numbers can be found in numerous sources such as IANA.

IP Protocol filters consist of a single 8-bit number. The filters do not perform any validation check on this 8-bit number and simply filter the packets matching the number. This is because vendors can use proprietary numbers and protocols at this layer. It should be noted that filtering on TCP/UDP protocols is still coarse and will likely include most packets destined for the host processor (depending on the application). Application layer filters that are based on port numbers are the next level of filter refinement, providing finer control on most packets received in the network.

3.3.2.3 Applications layer/TCP and UDP port numbers

Application layer filters or simply port filters perform packet filtering based on the source or destination port of a transport layer packet. These port numbers are well-known numbers used to identify various TCP- and UDP-based protocols (see IANA). These are 16-bit port numbers. For example, with a port filter, a user can filter on only SSH packets (port 22) or only on FTP packets (port 20), or any other of the many applications supported. Due to the large number and constantly changing port definitions, the filters do not validate the port numbers against available applications.

Transport layer packets have both source and destination ports. Destination ports are the well-known port numbers described in the IANA link and are generally the most useful. Source ports describe temporary, ephemeral port numbers used by the host sending the packets and are generated on-the-fly and are not well known. Because source ports are not known ahead of time, creating a source port filter is difficult. Filters can be designed to filter a range of ports (between a start and end port) to match a wide range of source ports to cover this case. Both TCP and UDP use port numbers; therefore, filters can be designed to select TCP or UDP. If both TCP and UDP need to be filtered for a port, two filters can be created.

3.3.3 TCP keepalive offload

TCP keepalive packets, as the name suggests, are used to keep an established TCP/IP connection between two hosts active. These are empty TCP packets that are transmitted at a periodic rate, typically 45 or 60 seconds, by a host. The peer host is expected to ACK the packet; if it does not ACK for a preconfigured number of TCP keepalive packets, the connection is considered dead.

In the idle case, after establishing a TCP connection, the host wakes periodically to send TCP keepalive packets. Whenever any other activity is detected in the TCP connection, such as incoming traffic, the host wakes up to process that as well. Figure 8 shows the TCP transactions that take place when the TCP keepalive offload is disabled. Notice how the host must repeatedly wake up from Deep Sleep to handle the transactions.

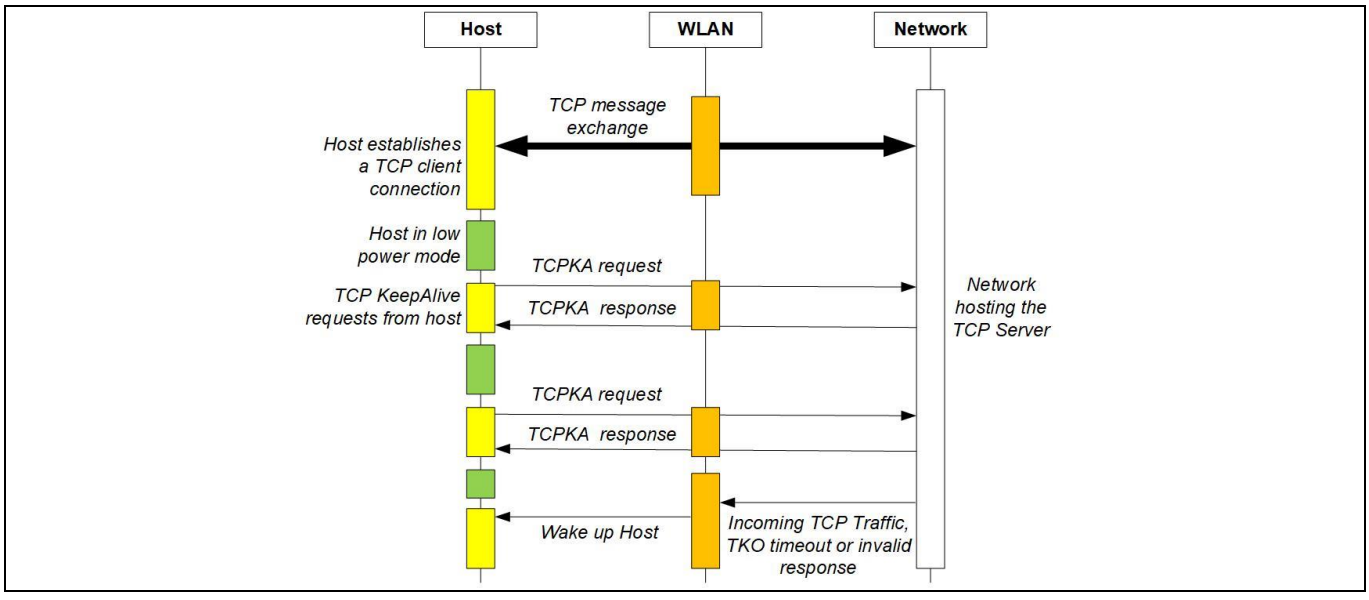

Figure 8 TCP activity when TCP keepalive offload is disabled

When the offload is configured and enabled, after the TCP connection is established, the host MCU configures the WLAN device to send TCP keepalive packets. The host MCU then enters Deep Sleep. When a TCP keepalive

WLAN power optimization techniques

request is received, the WLAN device takes care of responding to it. If there is any incoming TCP traffic or a TCP keepalive timeout, the WLAN device wakes up the host for processing. Figure 9 shows the TCP activity when the TCP keepalive offload is configured. Notice how the host can stay in Deep Sleep longer and wakes up only when responding to incoming TCP traffic.

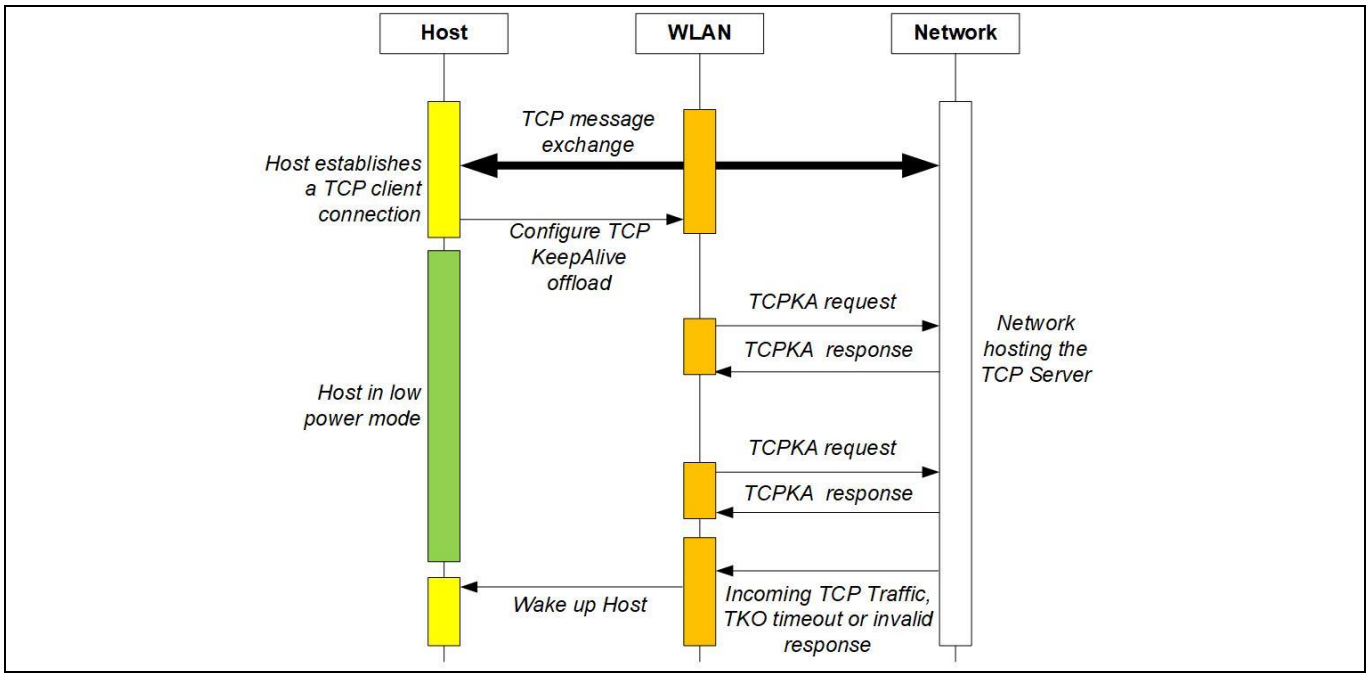

Figure 9 TCP activity when TCP keepalive offload is enabled

4 Bluetooth® power optimization techniques

AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip supports a Bluetooth® 5.0 BR/EDR/Bluetooth® LE radio. This section describes the power optimization techniques.

Common operations related to power consumption in a Bluetooth® LE application include advertisement and connection events. It is discussed how to optimize various parameters related to both advertisement and connection events to reduce power consumption. This section assumes that you are familiar with Bluetooth® LE fundamentals. See Vol 6: “Core System package [Low Energy Controller]” of the latest Bluetooth® Core specification for Bluetooth® LE fundamentals.

4.1 Advertisement events

Advertisement events broadcast the capabilities of a Bluetooth® LE Peripheral device to Bluetooth® LE Central devices listening to the events. These events can constitute the major portion of a Bluetooth® device’s battery consumption, if not carefully chosen. The interval between two successive advertisement events from the Peripheral forms the advertising interval; it can be fixed or a random value between a min and max interval.

To achieve the lowest possible power during advertisement events, you should maximize the advertising interval. However, a larger advertising interval leads to a longer detection time at the Central device and a subsequently longer time for getting connected to the Central. The advertising interval should be between 20 ms and 10.24 seconds. In addition, the advertisement interval between a min and max not only improves the tradeoff between power and latency, but also improves the detection probability in noisy environments by randomizing the time at which an advertisement packet is sent out.

Table 5 Advertising interval selection

Application requirement	Suggested advertisement interval
The application requires shorter detection and connection establishment periods such as audio devices or devices that send real-time data.	20 to 100 ms
The application does not have aggressive connection requirements but still prefers low latency such as fitness trackers or smart watches.	100 to 1000 ms
The application does not care about latency; power consumption is of more importance, such as in the case of beacons.	1000 to 10000 ms

In addition to the advertisement interval, the amount of data transmitted during advertisement impacts the power consumption during advertisement events. The smaller the advertisement payload, the lower the power consumption.

Bluetooth® Configurator

The Bluetooth® peripheral has an additional configurator called the “Bluetooth® Configurator”; used to generate the AIROC™ Bluetooth® LE GATT database and various Bluetooth® settings for the application. These settings are stored in the file named *design.cybt*. Note that, unlike the Device Configurator, the Bluetooth® Configurator settings and files are local to each respective application.

For detailed information on how to use the Bluetooth® Configurator, see the Bluetooth® Configurator guide.

4.2 Connection events

4.2.1 Connection interval

The connection interval for a Bluetooth® LE link is set by the Central device when a connection is created with the Peripheral device. The Peripheral should have a connection interval that matches the rate at which the sensor data from the Peripheral is sent to the Central device. If the initial connection interval set by the Central is lower than necessary, the Peripheral should request the Central to increase the connection interval to reduce current consumption. This can be done after a connection is established by sending a “connection parameter update request” to the Central.

4.2.2 Use slave latency

Slave latency is a Bluetooth® LE feature that allows a Peripheral to listen to the Central device on connection events at a reduced rate. This is useful if the Peripheral device is inactive for some time and has to wait for some activity to occur to send data to the Central device. If no activity is detected for some time, the Peripheral can enter slave latency by sending a connection update request to the Central. The Peripheral can extend the interval between connection events at which it listens to the Central. This allows the Bluetooth® LE device and the host to be put into Deep Sleep mode for longer durations.

The key advantage of using slave latency is that when the peripheral application detects activity, it can switch the Bluetooth® LE device back ON to listen to the Central at the original connection interval until the data transfer is completed. This avoids the latency in sending the data upon the first activity.

4.2.3 Bluetooth® LE parameters in Bluetooth® Configurator

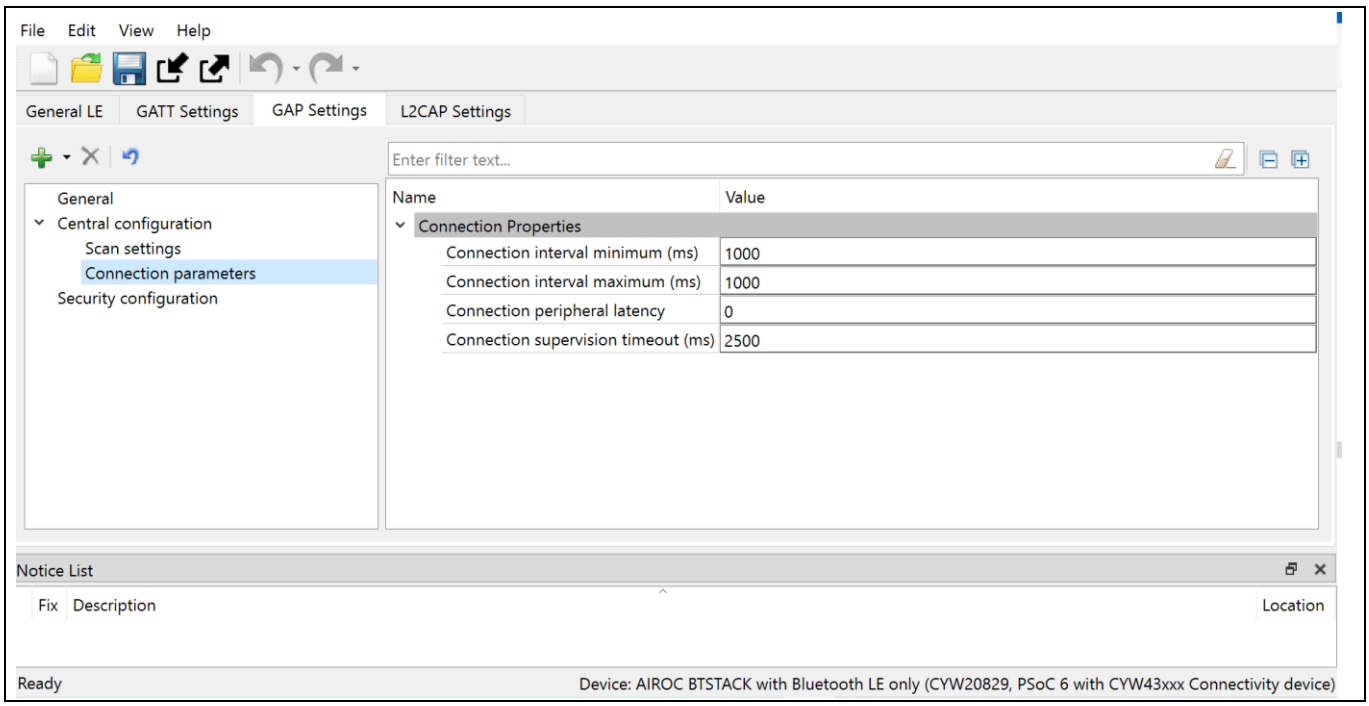

Figure 10 Connection parameters in Bluetooth® Configurator

Bluetooth® power optimization techniques

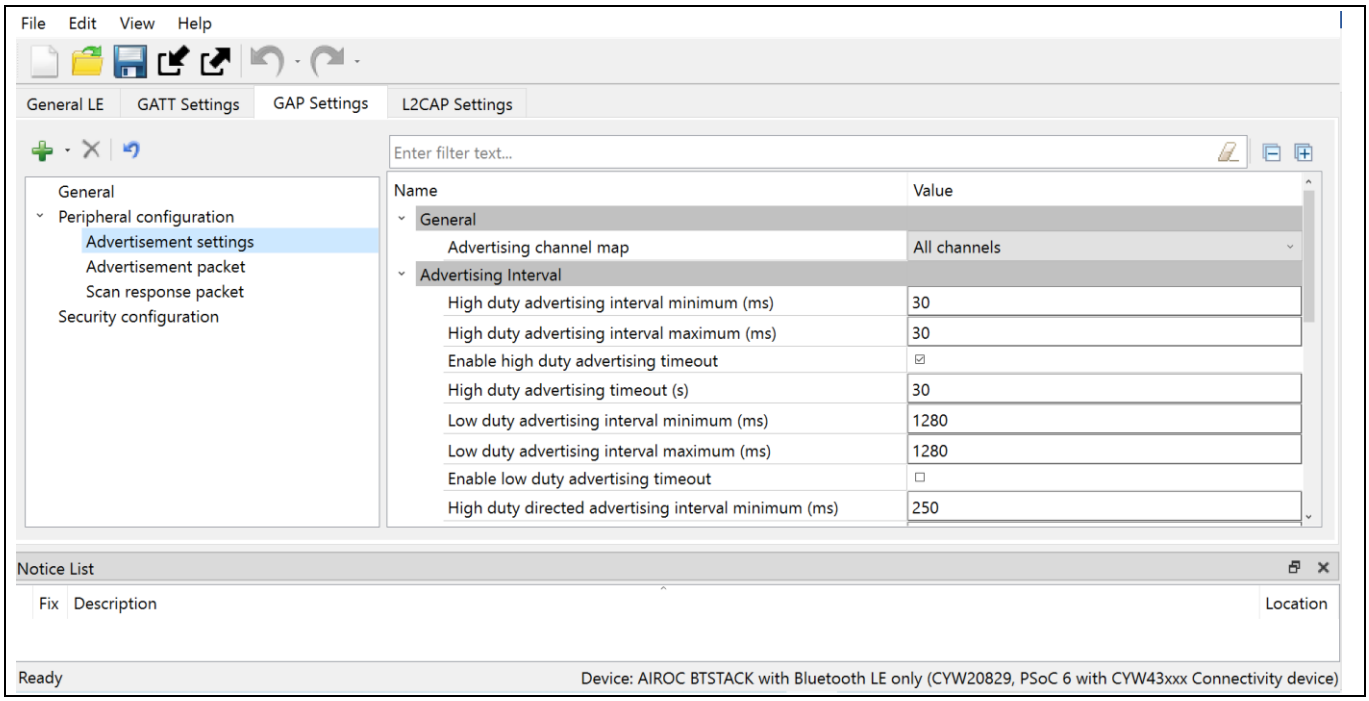

Figure 11 Advertisement settings in Bluetooth® Configurator

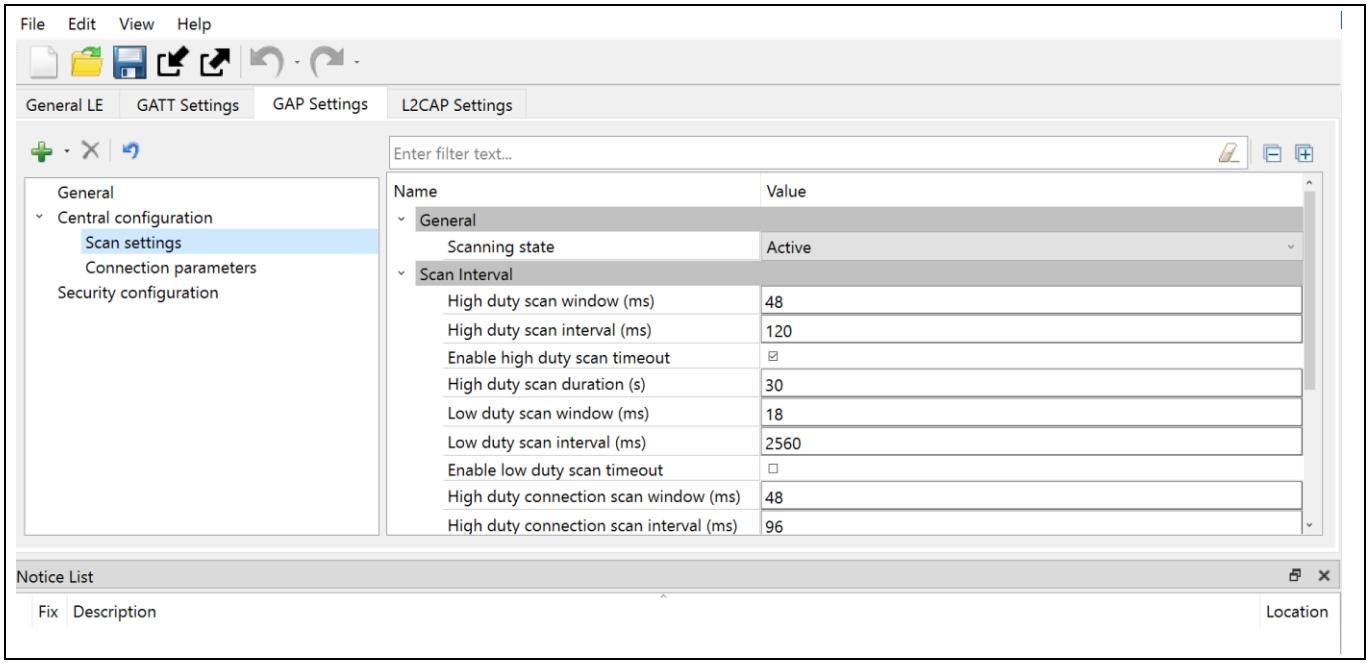

Figure 12 Scan settings in Bluetooth® Configurator

5 PSoC™ 6 MCU power optimization techniques

5.1 Core voltage and operating frequency

PSoC™ 6 MCU supports two core regulators – LDO or Buck – either of which can be used to power the CPU core. In addition, the system supports two active power modes – System LP and System ULP. In the System LP power mode, the CPU frequency can reach up to 150 MHz, whereas in the System ULP mode, the CPU frequency is limited to 50 MHz. The power consumed by the CPU is much lower in System ULP mode.

By default, the LDO powers the core and it is in System LP mode. The LDO regulator consumes a higher power than the buck regulator in active mode (>50%) for the same CPU clock frequency irrespective of the LP or ULP mode. However, in LP mode, the leakage in Deep Sleep is lower (~20%) using the LDO regulator as compared to the buck regulator. Unless the application spends < 0.01% time in active mode (1 ms out of 10 s), it is recommended to use the buck regulator as the core regulator because the added leakage in Deep Sleep using the buck regulator is outweighed by the power saving in active mode. When using ULP mode for active power mode, it is always recommended to use the buck regulator because the leakage in ULP mode is lower than the LDO in System ULP mode.

Table 6 Core regulator and active power mode selection

Application use case	Core regulator	System active power mode
(CPU Clock > 50 MHz OR Peri Clock > 25 MHz) AND CPU active/sleep time < 0.01 %. 0.01 % = 1 ms out of 10 s.	LDO	LP
(CPU Clock > 50 MHz OR Peri Clock > 25 MHz) AND CPU active/sleep time > 0.01 %. 0.01 % = 1 ms out of 10 s.	Buck	LP
(CPU Clock < 50 MHz AND Peri Clock < 25 MHz)	Buck	ULP

See AN219528 for details on the power modes and the PSoC™ 6 MCU datasheet for details on the power parameters.

5.2 Reducing the leakage in Deep Sleep

PSoC™ 6 MCU supports selective retention of SRAM in sizes of 32 KB when the device enters Deep Sleep. The smaller the amount of SRAM retained in Deep Sleep, the lower the Deep Sleep power consumption will be. If the application knows the amount of SRAM it plans to use, the unused SRAM blocks can be disabled to improve the Deep Sleep power consumption. However, some guidelines must be kept in mind while disabling the SRAM blocks:

1. A few platforms, allocate the unused SRAM to the heap. As a result, the amount of SRAM consumed by the heap (or dynamically allocated objects in the code) should be considered and added to the budget.
2. System calls supported in PSoC™ 6 MCU use 2 KB of SRAM available at the end of the SRAM region present in the device. As a result, if the application uses/requires any system calls, such as flash write/erase, it must not disable the last block of the SRAM (block 8 in PSoC™ 6 MCU devices with 288 KB RAM). However, the block can still be turned off before entering Deep Sleep (because system calls are executed in active mode only).
3. All projects should reserve 8 KB (0x2000) of SRAM at the start of the SRAM region for the CM0+ core; this should be accounted into the overall SRAM consumption of the application.

PSoC™ 6 MCU power optimization techniques

Do the following to disable an unused SRAM block in the code:

1. Find out the SRAM consumed by the application. CPU Stack statistics and other memory analyzer tools supported in the platforms can be used to determine memory usage.
2. Update the linker script to reduce the RAM size to the value calculated above. Make it a multiple of 32 KB and fit the size calculated in Step 1. This step is compiler-dependent as well.

   GCC (cy8c6xx*_cm4_dual.ld file):

   ```
   ram                (rwx)    : ORIGIN = 0x08002000, LENGTH = 0xFD800
   ```

   ARM (cy8c6xx*_cm4_dual.sct file):

   ```
   #define RAM_SIZE                  0x000FD800
   ```

   IAR (cy8c6xx*_cm4_dual.icf file):

   ```
   define symbol __ICFEDIT_region_IRAM1_start__ = 0x08002000;
   define symbol __ICFEDIT_region_IRAM1_end__   = 0x080FF7FF;
   ```

   CLANG (cy8c6xx*_dual.mk file):

   ```
   export RAM_SIZE_CM4       := 0x000FD800
   ```

3. Add code to disable SRAM blocks and/or controllers depending on the device and memory consumed. For example, consider the following cases:

a) PSoC™ 6 MCU with 1 MB of SRAM has three controllers – SRAM0 (512 KB), SRAM1 (256 KB), and SRAM2 (256 KB). As an example, if your application requires only 450 KB of SRAM (= 480 KB, multiple of 32 KB), you can completely disable SRAM1 and SRAM2 controllers and the last block of SRAM0 using the following code:

```
CPUSS->RAM0_PWR_MACRO_CTL[15]= 0x05FA0000; //Disable block 15 in SRAM0
CPUSS->RAM1_PWR_CTL= 0x05FA0000;//Disable SRAM1
CPUSS->RAM2_PWR_CTL= 0x05FA0000;//Disable SRAM2
```

b) If you want to add system call support in the same application, you should not disable the last block of the SRAM region, which is the last 32 KB block in SRAM2:

```
CPUSS->RAM0_PWR_MACRO_CTL[15]= 0x05FA0000;
CPUSS->RAM1_PWR_CTL= 0x05FA0000;
for(int32 i = 0; i < 15; i++) //Do not disable block 15 alone.
CPUSS-> RAM0_PWR_MACRO_CTL[i]= 0x05FA0000;
```

5.3 RTOS tickless mode

All RTOS requires a clock or a "tick" for timing and managing tasks. These ticks are usually small RTOS kernel tasks that wake up periodically and update the RTOS tick count used by delays, timeouts, and other timing-related tasks. Often, these ticks are configured in ms. This results in the CPU waking up often just to service the tick task even if it is not running any useful tasks.

Applications that do not use delays or timeouts or use large delays need not wake up this often and can enter a tickless mode provided by the platform. In tickless mode, the RTOS suspends the tick task and lets the CPU enter a low-power state (Deep Sleep) for longer durations. This duration is typically determined by the time the next task that depends on a delay needs to be triggered. If there are no tasks dependent on delays, the device can remain in the low-power state indefinitely until an external event (such as a GPIO interrupt) wakes up the device and triggers the task.

When delays are involved while entering tickless mode, a timer that is active in the System Deep Sleep state is configured to generate an interrupt after the delay expires.

5.4 Additional power optimization techniques

See AN219528 for additional power optimization techniques in PSoC™ 6 MCU.

6 Low-power assistant (LPA)

The Infineon® l assistant (LPA) allows configuring PSoC™ 6 MCU host and WLAN (Wi-Fi/Bluetooth® radio) devices to provide low-power features. Key highlights of the LPA include the following:

- Self-aware firmware that detects configurations automatically and enables appropriate low-power features without any additional API calls from the user
- Supports multiple platforms such as FreeRTOS
- GUI-based configuration for ease of use
- Supports low-power configuration for PSoC™ 6 MCU, Wi-Fi, and Bluetooth®

LPA lets you optimize various parts of your design to be energy-efficient. The LPA configuration is split into three main parts:

- **PSoC™ 6 MCU (host) low-power configuration:** Includes PSoC™ 6 MCU-specific low-power configurations such as core voltage, regulator selection, and RTOS idle power mode
- **Wi-Fi low-power configuration:** Includes Wi-Fi-specific low-power configurations such as Wi-Fi host wake signal selection and host offload configurations (ARP and packet filters)
- **Bluetooth® low-power configuration:** Includes Bluetooth® low-power configurations such as host wake interrupt signal (device to host) and related device integration

These configurations can be generated through the device configurator in ModusToolbox™ software or added manually through code. Given the simplicity of the configurations, adding them through a code is not difficult. However, using the configurator lets you take advantage of a GUI-based configurator in addition to making the generated configuration easily upgradable in the future; it is the recommended way to create low-power configurations.

See LPA documentation for details on the middleware, configurators, and quick start. The following sections provide a brief on the PSoC™ 6 MCU, Wi-Fi, and Bluetooth® low-power configurations available and their usage in different platforms.

6.1 LPA configuration

To use the LPA in your application, you need to input some configurations. Use the *design.modus* device configuration file shipped with the BSP kit for configuring LPA settings. The LPA settings are available under **design.modus** > **[PSoC™ 6 MCU device in target]** > **System** > **Power** for PSoC™ 6 MCU-related power settings and **design.modus** > **[AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip device in target]** > **Power** > **WiFi or BT** for Wi-Fi and Bluetooth® settings. Figure 13 shows the PSoC™ 6 MCU and Wi-Fi LPA (power) settings for the CY8CKIT-062S2-43012 target. You need to enable the “Power” setting to enable LPA configuration generation. For a detailed description of all the configurations, see the LPA documentation.

Low-power assistant (LPA)

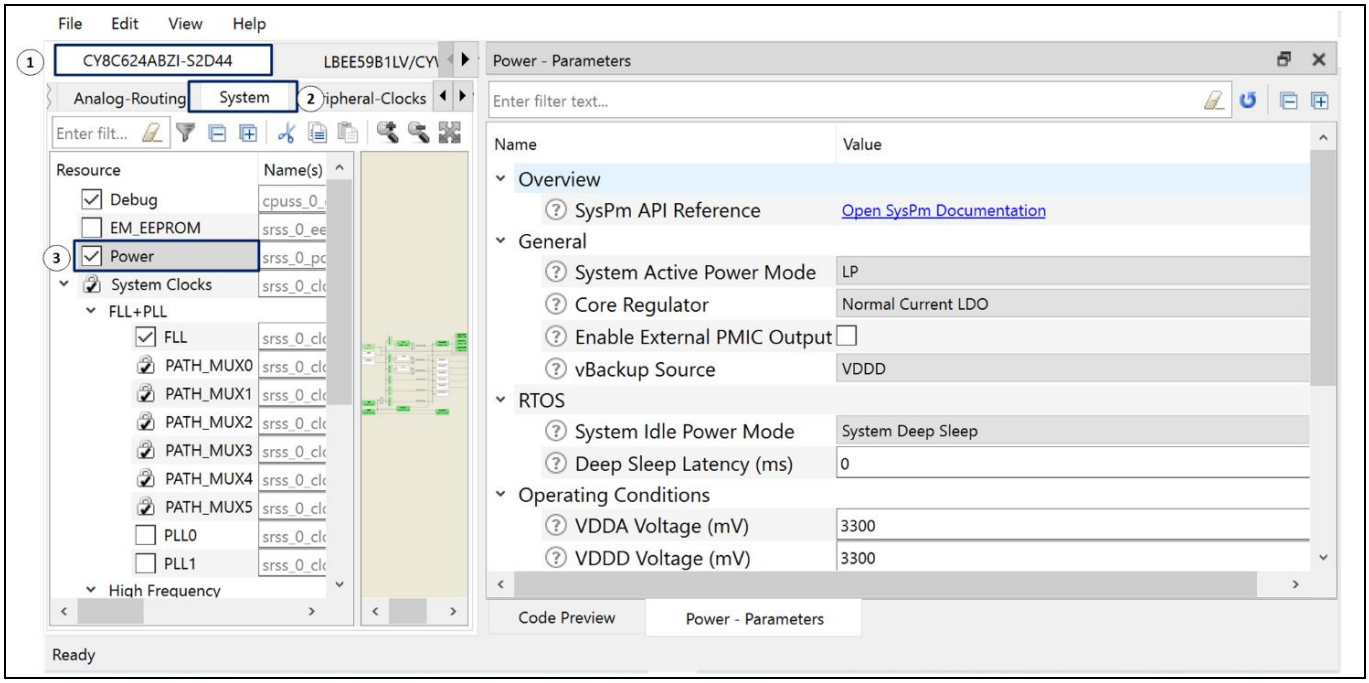

Figure 13 PSoC™ 6 MCU LPA configurations in *design.modus*

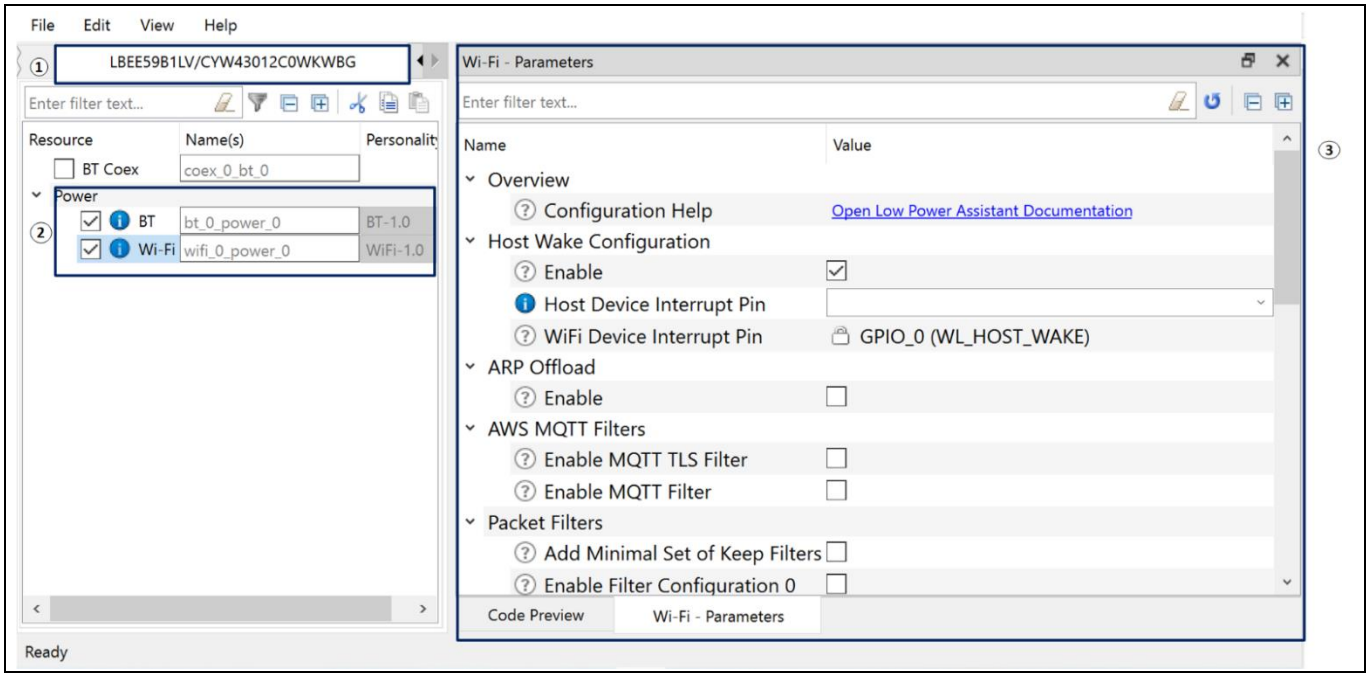

Figure 14 Wi-Fi/Bluetooth® LPA configuration in *design.modus*

Low-power assistant (LPA)

Table 7 Guidelines for LPA configuration

Parameter	Description	Recommended configuration
PSoC™ 6 MCU (Host)		
System active power mode	Selects the power mode the core operates in when active – LP or ULP	See Table 6.
Core regulator	Selects the type of regulator that powers the core	
System idle power mode	Selects the power mode the core operates in when idle – Active, Sleep, or Deep Sleep	Deep sleep (for maximum power saving) Active (for minimum latency)
Deep sleep latency (ms)	Selects the minimum time for which the CPU must be idle before entering Deep Sleep	0 (enters Deep Sleep immediately in idle) because the Deep Sleep wake-up latency is <50 us, it is OK to enter Deep Sleep as soon as the device hits idle. Increase this parameter only if the application involves tasks that need to wake up/run every 2-3 ms at times.
Wi-Fi		
Host wake configuration	Enables the capability of the WLAN device to wake up the host from Deep Sleep	Enable (Check-box ticked)
Host device interrupt pin	Selects the pin from the host to be configured as the host wake signal from the WLAN device	First, configure the pin in the PSoC™ 6 MCU that is connected to WL_HOST_WAKE (see board schematics). The pin configuration includes setting the drive mode to **Digital Hi-Z** and interrupt trigger type to **Falling Edge**. The interrupt trigger type selects the polarity of the WL_HOST_WAKE signal from CYW43012. Falling selects “Active LOW” and Rising selects “Active HIGH” configuration. For example, in CY8CKIT-062S2-43012, P4[1] is connected to WL_HOST_WAKE of the AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip device. So, enable the pin, set the drive mode to **Digital Hi-Z,** interrupt trigger type to **Falling Edge**, and select it from the drop-down.

Low-power assistant (LPA)

Parameter	Description	Recommended configuration
ARP offload	Enables ARP offload feature	Enabled
ARP offload features	Selects the functionality of ARP offload	Peer Auto Reply is sufficient for most use cases.
Snoop host IP from traffic	Selects whether the AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip device snoops ARP information from host ARP responses	Enabled
Packet filters	Configure multiple packet filters	Depends on the application. See Packet filter offload. If using "keep" filters, enable the **Add Minimal Set of Keep Filters** option.
Bluetooth®		
Enable	Enables low-power Bluetooth® operation	Enabled
Host-wake-up configuration	Selects the pin from the host to be configured as the wake-up signal from the Bluetooth® device.	Board-dependent and similar to the "Host Device Interrupt Pin" configuration in Wi-Fi.
Device-wake-up configuration	Selects the pin from the host to be configured as the wake-up signal to the Bluetooth® device	

6.2 Using LPA in ModusToolbox™ software

To include the Low-power assistant in any Wi-Fi or Bluetooth® ModusToolbox™ software code example, open the Library manager and add the low-power assistant middleware. The Library manager will take care of getting dependencies required by the Low-power assistant middleware.

Do the following to open *design.modus* for the target used in the example:

1. Open *[Install directory]/ModusToolbox/tools_version/device-configurator/device-configurator.exe*.
2. In the Device configurator, select File > Open.
3. Navigate to the *design.modus* file for your target available inside the following example folder: *[Example_directory]/bsps /TARGET_[CY_BOARD]/config/design.modus*
4. If an error message as shown in Figure 15 appears, click OK and then follow these instructions:
   a) Pre-ModusToolbox™ 3.0 design:

      Navigate to *[Example_directory]/../mtb_shared/mtb-pdl-cat1/[release_version]/* and select the devicesupport.xml file.

   b) Post-ModusToolbox™ 3.0 design:

      Navigate to *[Example_directory]/../mtb_shared/mtb-pdl-cat1/release-version/props.json*

      Navigate to *[Users_directory]/../.modustoolbox/global/device-db/release-version/props.json*

Low-power assistant (LPA)

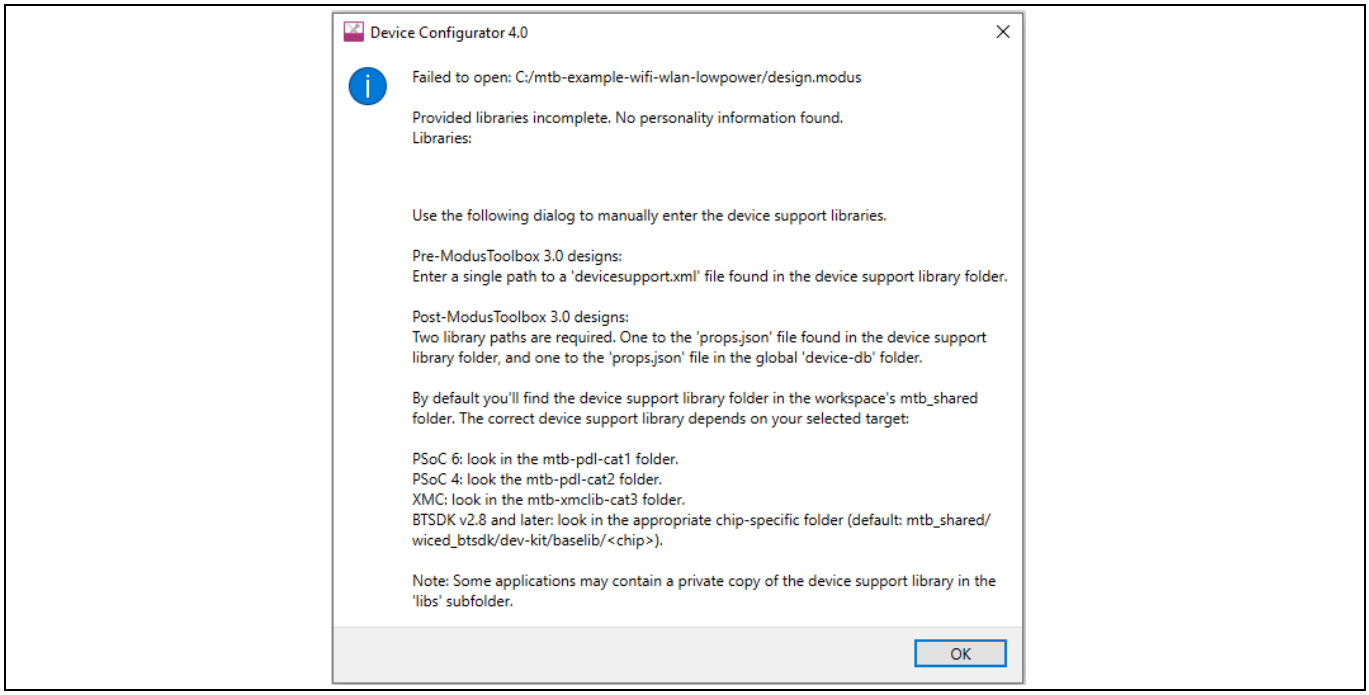

Figure 15 No 'device support library' path found error

5. Back up the existing *design.modus* file if required. Open *design.modus* and configure the LPA as explained in the LPA configuration. For example, to enable low power in the MQTT client example available in GitHub, you can use the configuration provided:

Table 8 LPA configuration

Parameter	Configuration	Comment
PSoC™ 6 MCU (Host)		
System active power mode	LP	You need >50 MHz M0+ for Wi-Fi applications
Core regulator	Buck	CPU will wake up often (>0.1%) to service Wi-Fi activities
System Idle Power mode	Deep sleep	For the lowest power consumption during idle
Deep sleep latency (ms)	0	Enters Deep Sleep immediately in idle
Wi-Fi		
Host wake configuration	Enable	Required for low power
Host device interrupt pin	<P4.1 for CY8CKIT_062S2_43012>	-
ARP offload	Enabled	Reduces the number of ARP requests to host and improves power consumption
ARP offload features	Peer Auto Reply	
Snoop host IP from traffic	Enabled	

Low-power assistant (LPA)

Parameter	Configuration	Comment
Packet filters	Enable the **Add Minimal Set of Keep Filters** option	Ensures that the device can connect to a Wi-Fi AP and obtain an IP address
	Enable **MQTT TLS filter** Enable **MQTT filter** Other configuration - Action: **Keep** Protocol: **TCP** Direction: **Source Port**	Ensures that all MQTT packets reach the host. After connection to the AP, the host receives only the MQTT packets.

6. Disable peripherals such as CAPSENSE™, Serial Communication Block (SCB), and quad serial memory interface in PSoC™ 6 MCU if they are enabled but not used in the example.

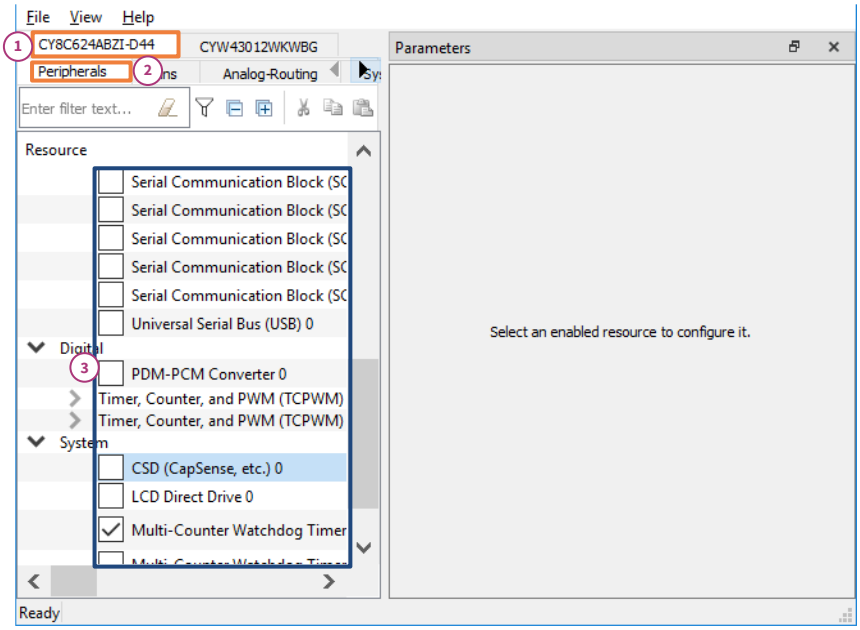

Figure 16 Disable peripherals in *design.modus*

7. Review the system clock settings and disable unused clock paths and PLLs. For example, in the CY8CKIT-062S2-43012 *design.modus*, PLL1 can be disabled if USB is not used. To save further power, PLL0 can be disabled and FLL can be routed to CLK_HF0 (CPU clocks):

Low-power assistant (LPA)

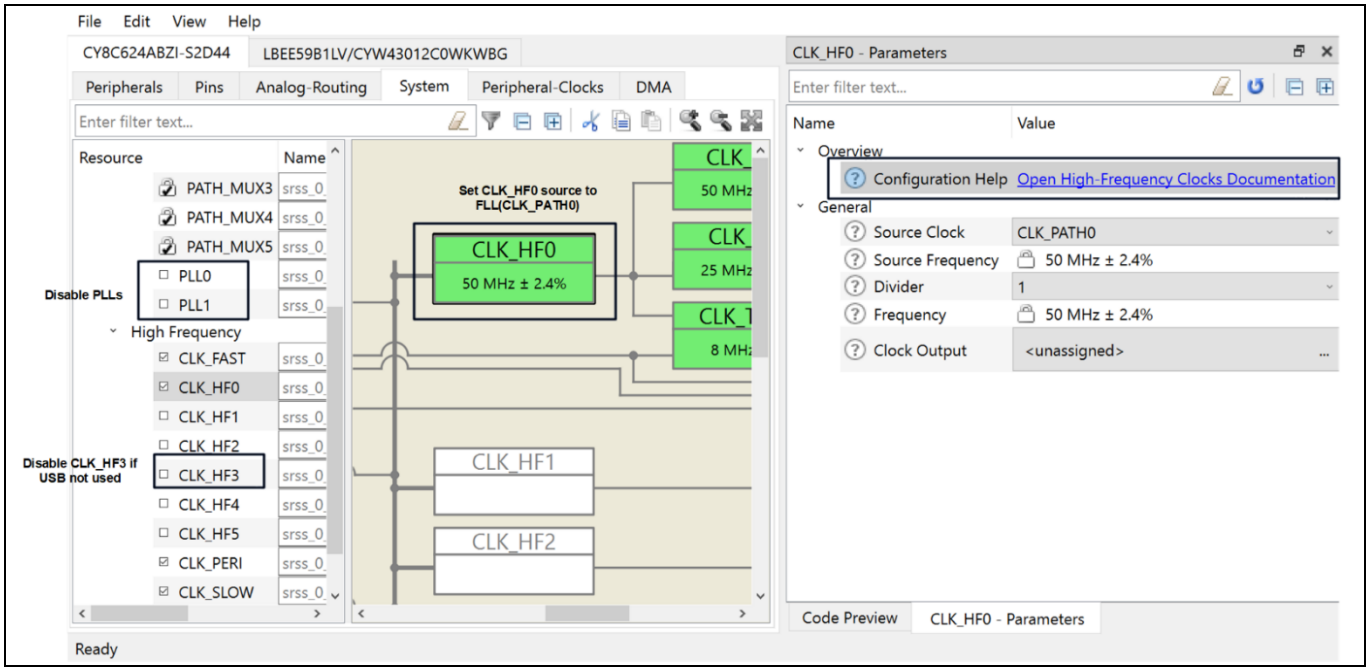

Figure 17 CY8CKIT-062S2-43012 clock setup for low power

8. Select **File** > **Save** to save the configuration.
9. Build and program the example from the Eclipse IDE for ModusToolbox™ software.

A list of LPA examples available in ModusToolbox™ software is provided in Table 9.

Table 9 ModusToolbox™ software examples for LPA

Example	Description	LPA features
mtb-example-wifi-wlan-lowpower	Simple example that demonstrates WLAN low-power modes and host network suspend features	Wi-Fi wake host signal
mtb-example-wlan-offloads	This code example demonstrates various WLAN offloads such as Address Resolution Protocol (ARP) offload, packet filter offload, and the TCP keepalive offload functionality	Wi-Fi wake host signal, TCP keepalive offload, ARP offload, and packet filter

7 Power measurement using CY8CKIT-062S2-43012

7.1 Hardware setup

For power measurement, use a power analyzer such as N6705B from Keysight because it gives insights on power transitions and a better estimate of power saving achieved using offloads. If you do not have access to a power analyzer, use a simple 6 ½-digit multimeter to monitor the PSoC™ 6 MCU and WLAN average power individually. Figure 18 shows various power measurement jumpers available on the CY8CKIT-062S2-43012 kit.

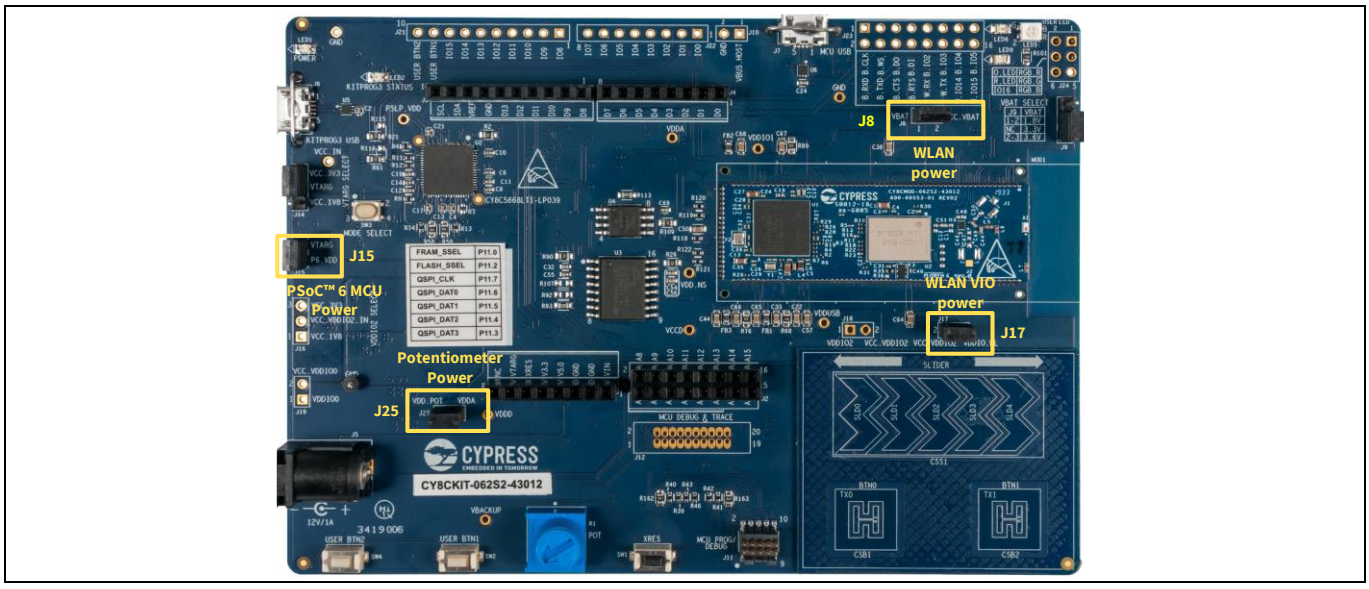

Figure 18 Power measurement jumpers in CY8CKIT-062S2-43012

Do the following for power measurement in CY8CKIT-062S2-43012:

1. Measure current for PSoC™ 6 at J15 across VTARG and P6.VDD.
2. Ensure that J25 is removed. This eliminates the leakage current from VDDA through potentiometer R1.
3. Measure the current for AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip VBAT at J8 across VBAT and VCC.VBAT.
4. Measure the current for AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip VDDIO at J17 across VDDIO.WL and VCC.VDDIO2.
5. See Figure 19 for a sample setup using the N6705B power analyzer from Keysight. Ensure that you select the PSoC™ 6 MCU (VTARG) and AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip (VBAT) voltage as the same as shown in the setup. The setup does not measure AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip VDDIO because it usually varies and depends on the activity on SDIO lines and the impact is minimal during active power measurement.

Power measurement using CY8CKIT-062S2-43012

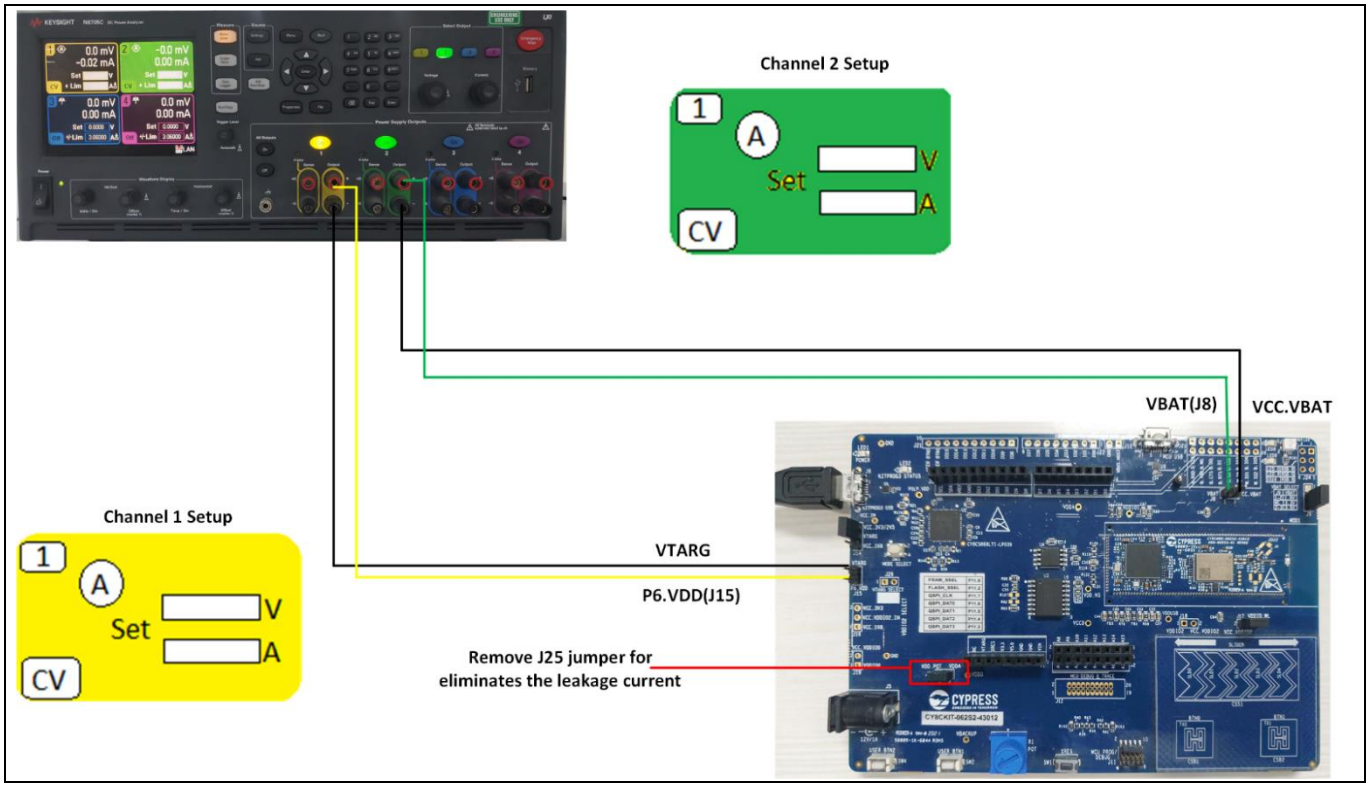

Figure 19 Power measurement setup using N6705B power analyzer from Keysight

7.2 mtb-example-wifi-wlan-lowpower

This section describes power measurement in CY8CKIT-062S2-43012 with the mtb-example-wifi-wlan-lowpower code example. This example can be used to measure AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip power consumption as documented in the datasheet.

By default, the example configures the AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip in Power Save Without Poll (PS-non-Poll) mode for better throughput and suspends the network stack so that PSoC™ 6 MCU remains in Deep Sleep for as long as no activity is detected in the network. Follow the example webpage for details on configuring your access point name/password and for programming the code example into the kit.

1. Set up the CY8CKIT-062S2-43012 kit as explained in the Hardware setup. Note that this section uses a Keysight N6705B as the reference setup to measure power.
2. Turn on the channels to power the PSoC™ 6 MCU and AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip from the N6705B power analyzer from Keysight. You can connect the onboard KitProg for viewing logs in terminal software through the COM port.
3. Ensure that the configured Wi-Fi AP is in range and the device connects to it (Figure 20).

Power measurement using CY8CKIT-062S2-43012

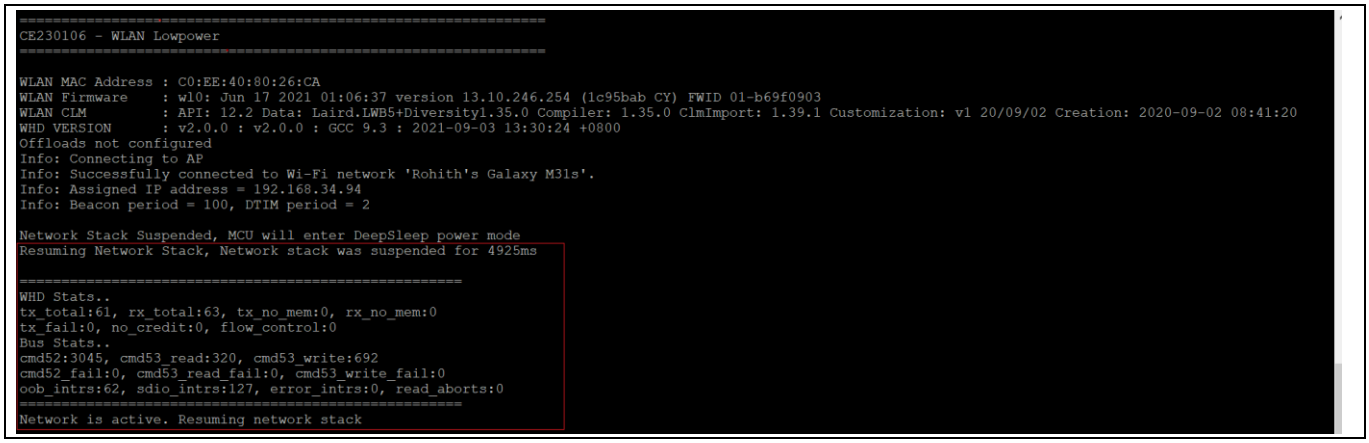

Figure 20 Connect to AP and suspend the network stack

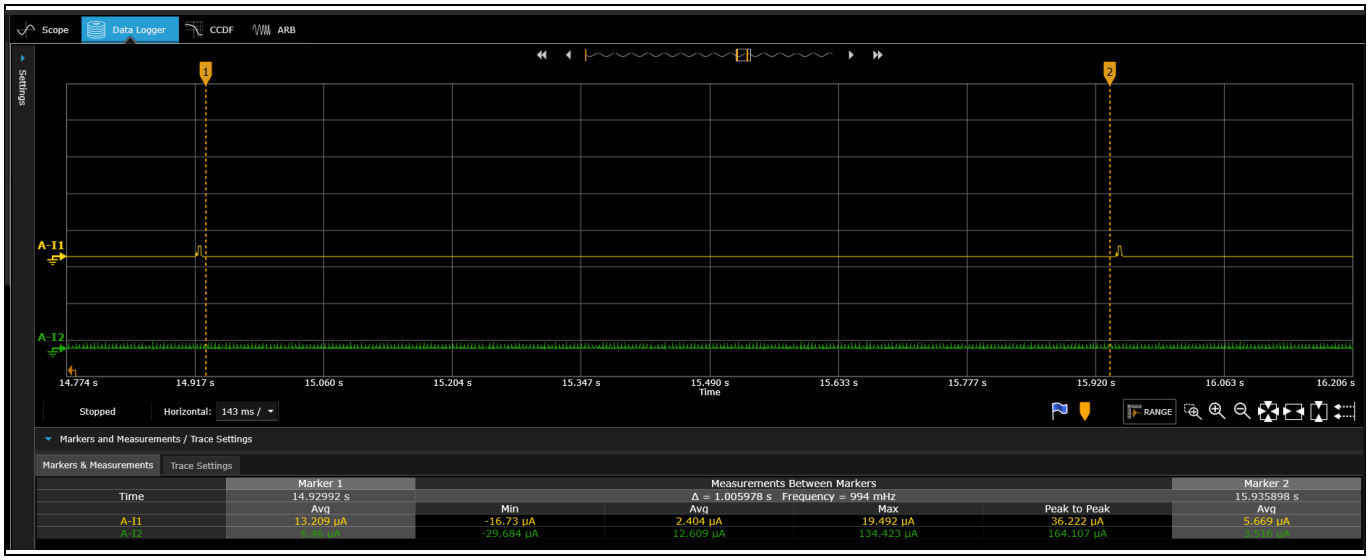

Figure 21 Deep Sleep current measurement values

4. Monitor the power consumption at P6.VDD and VBAT. Figure 22 shows the power consumption of both PSoC™ 6 MCU (P6.VDD) and AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip (VBAT) after board initialization and association with the AP. The yellow pulse represents the AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip current and the green pulse represents the PSoC™ 6 MCU current. The AP is configured for a beacon interval of 100 ms and DTIM = 1 and an interval of 100 ms and DTIM = 2.

   Note that the average power consumption will vary between DTIM intervals. This is because of other activities in the Wi-Fi spectrum (2.4 GHz or 5 GHz) such as other Wi-Fi networks, Bluetooth® (2.4 GHz) or Long-Term Evolution (LTE in 2.4 GHz) interference that increase the duration for which AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip listens for DTIM packet from the AP. This type of activity increases the average power consumption. To obtain the datasheet numbers, measure power in a shielded environment to avoid any interference.

Note: Figure 22 shows a measurement taken in a non-shielded test setup. For power measurements under ideal test conditions (shielded environment), refer to the datasheet.

Power measurement using CY8CKIT-062S2-43012

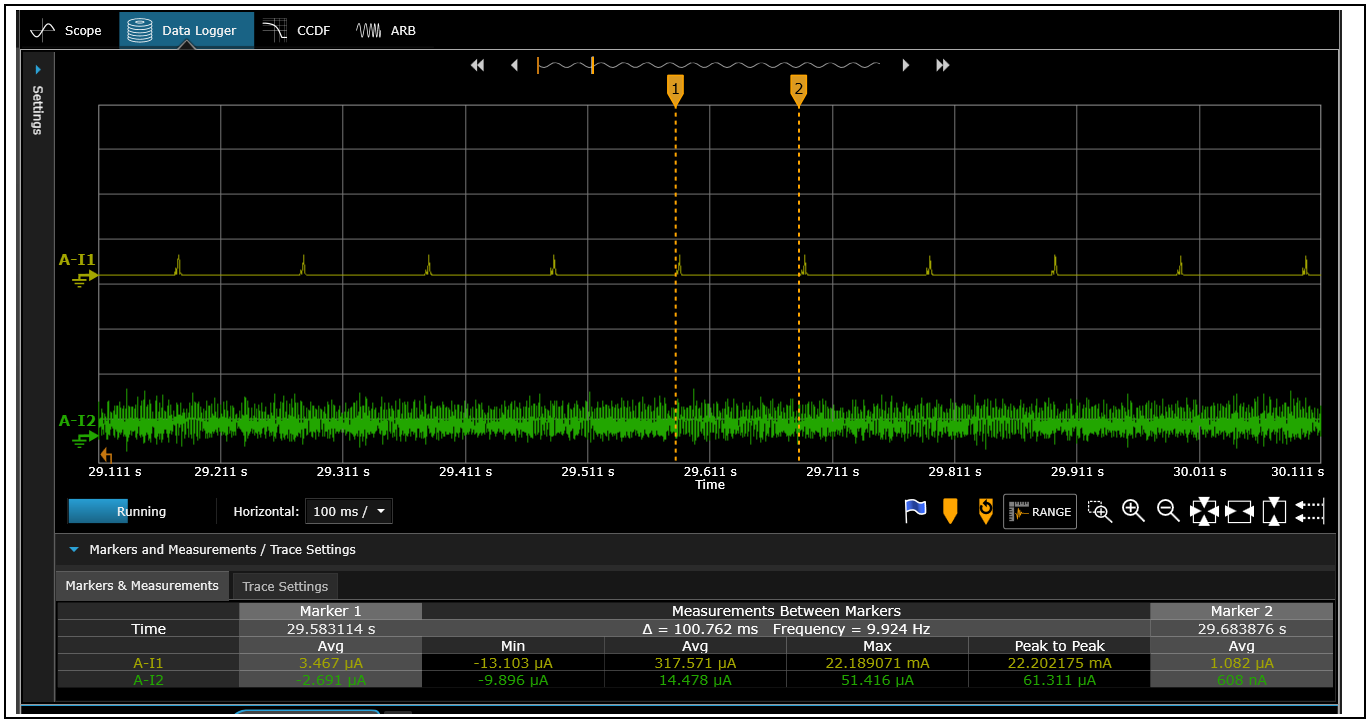

Figure 22 CY8CKIT-062S-43012 power consumption (associated, DTIM = 1, 2.4 GHz)

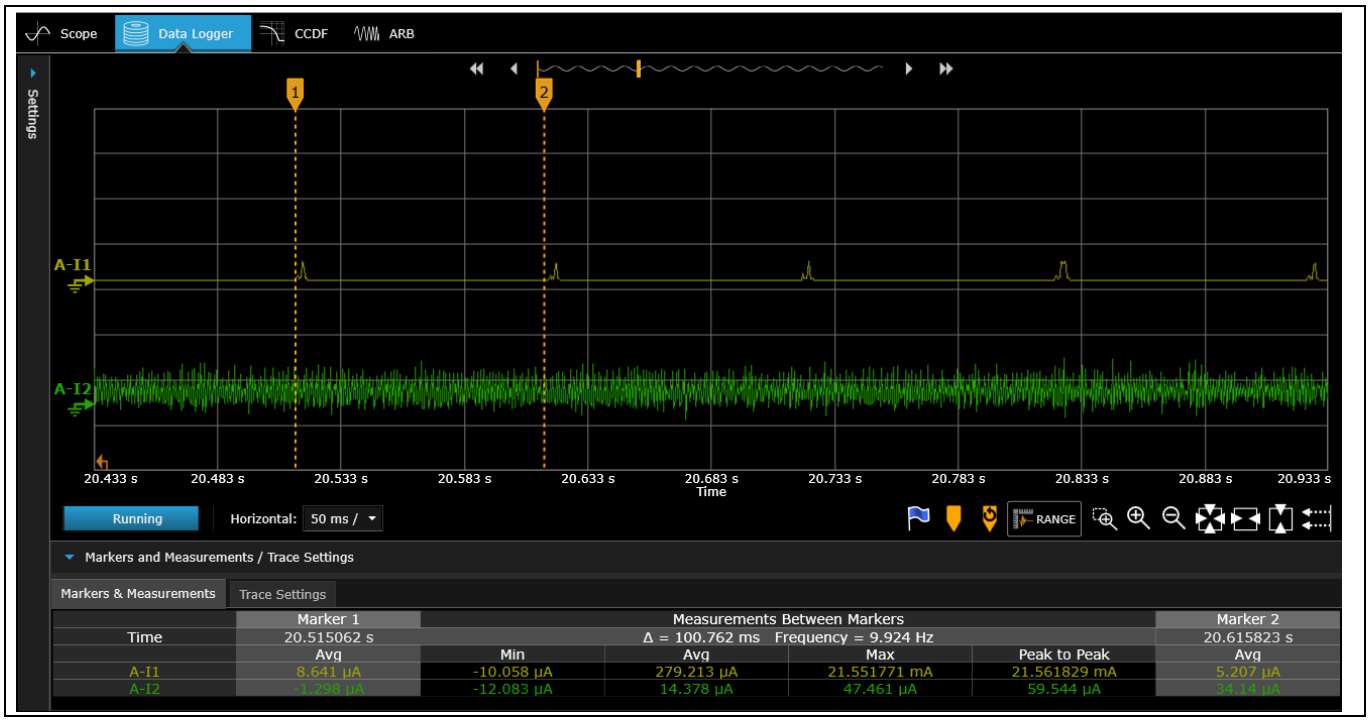

Figure 23 CY8CKIT-062S-43012 power consumption (associated, DTIM = 1, 5 GHz)

Power measurement using CY8CKIT-062S2-43012

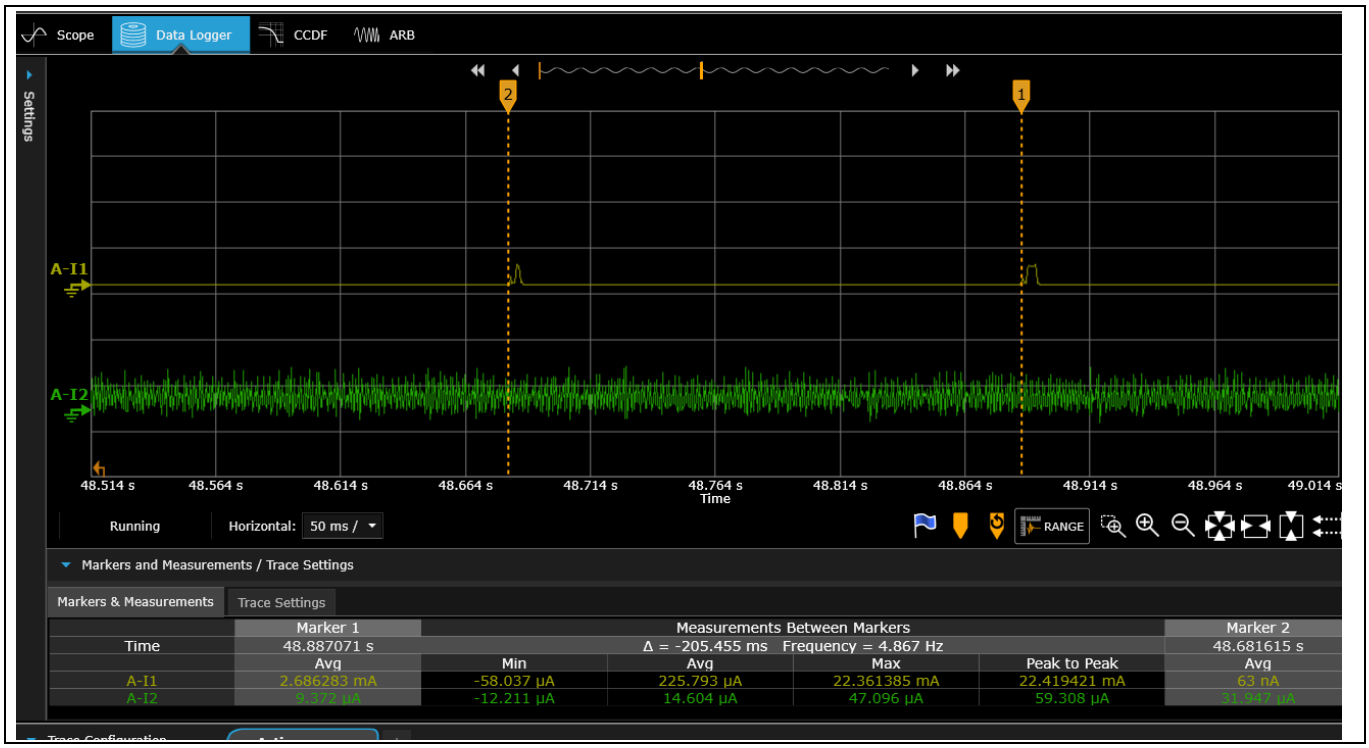

Figure 24 CY8CKIT-062S-43012 power consumption (associated, DTIM = 2, 2.4 GHz)

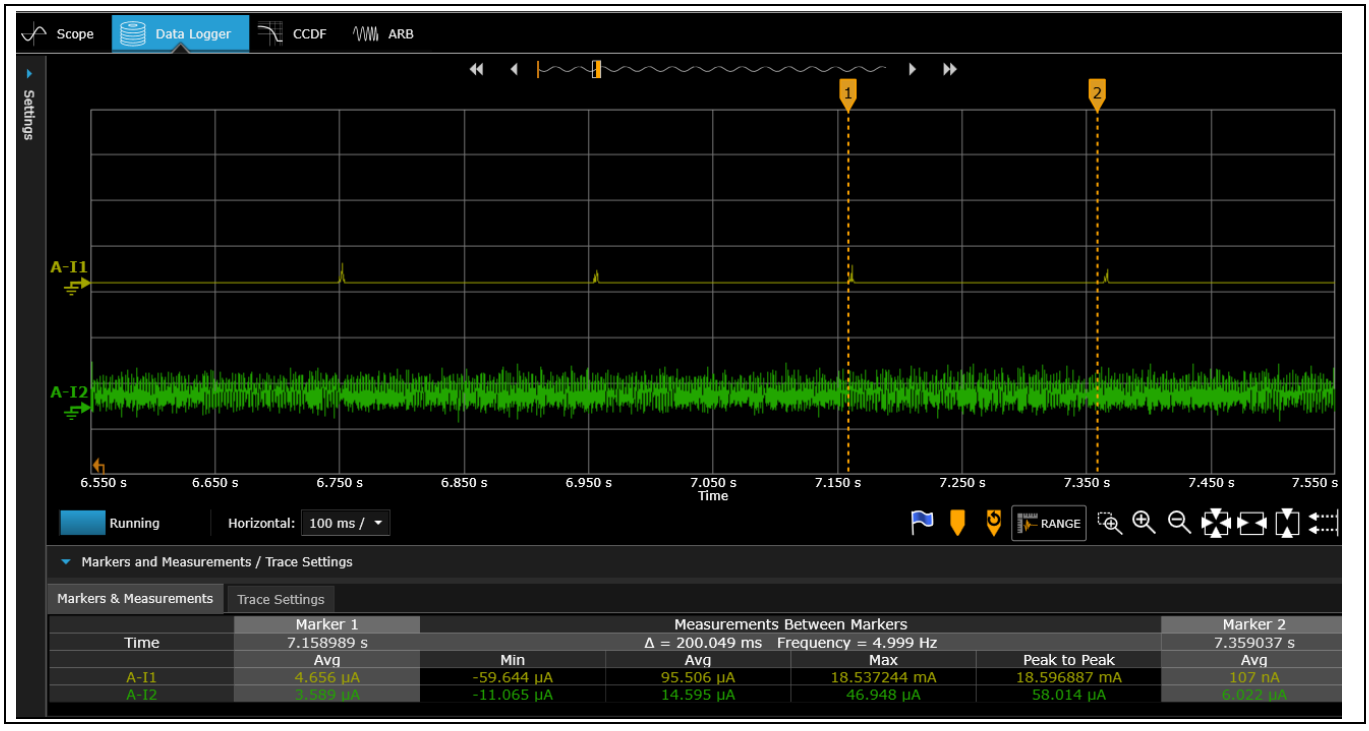

Figure 25 CY8CKIT-062S-43012 power consumption (associated, DTIM = 2, 5 GHz)

Power measurement using CY8CKIT-062S2-43012

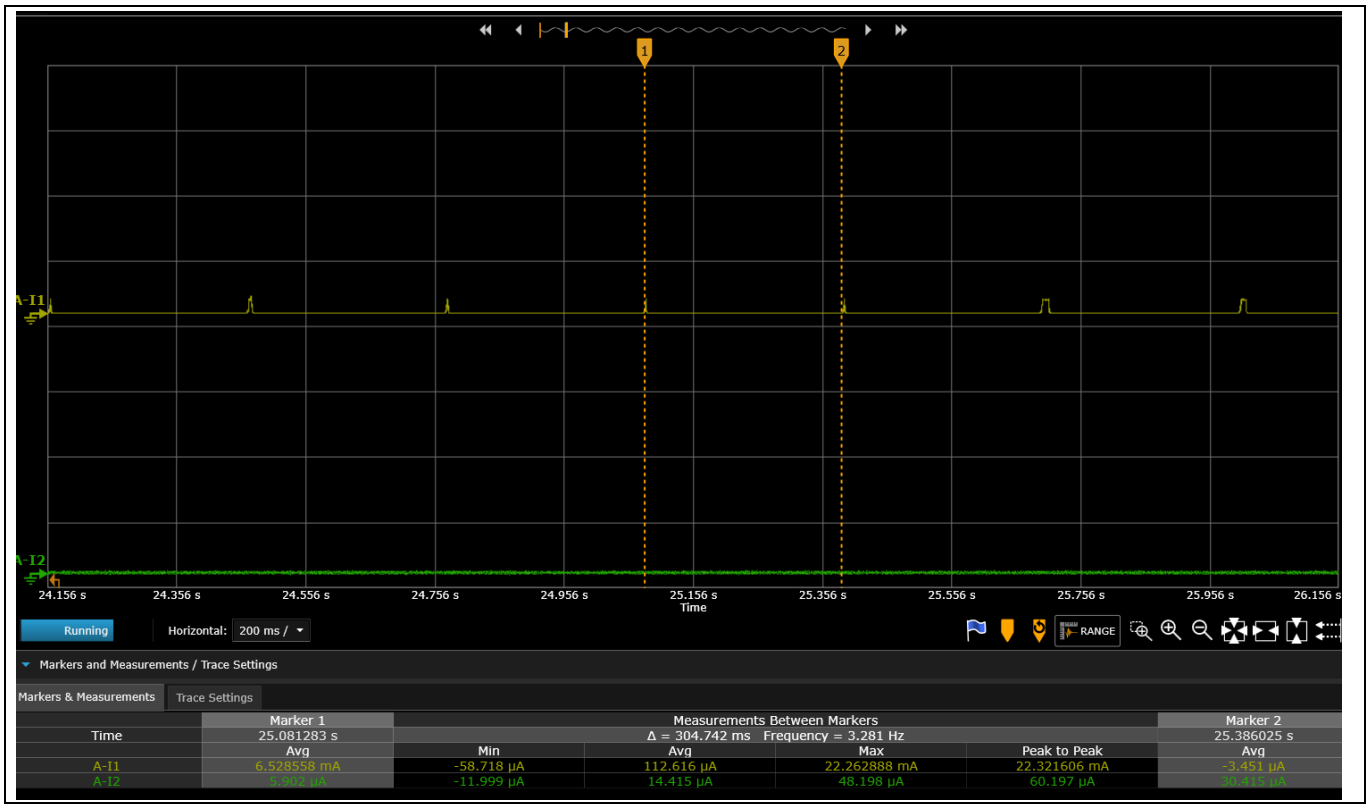

Figure 26 CY8CKIT-062S-43012 power consumption (associated, DTIM = 3, 2.4 GHz)

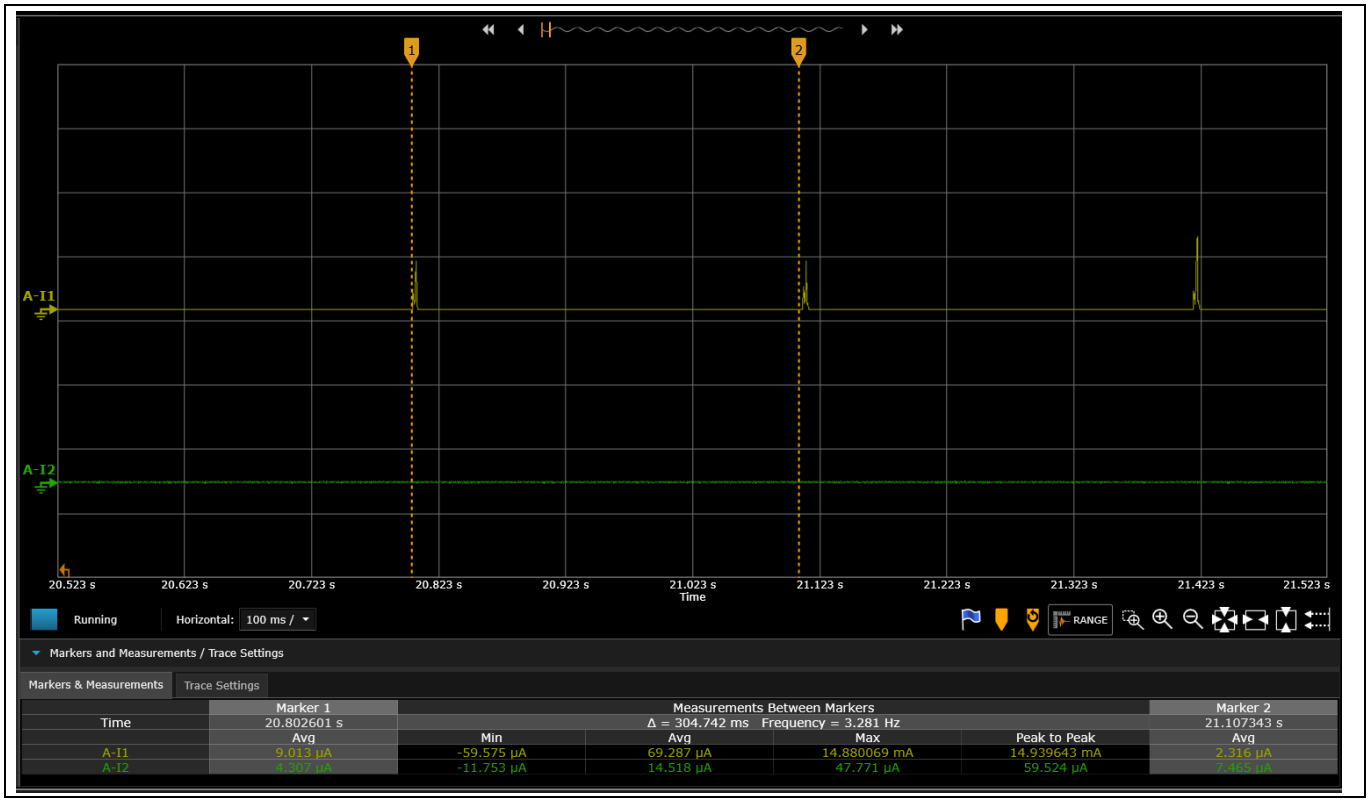

Figure 27 CY8CKIT-062S-43012 power consumption (associated, DTIM = 3, 5 GHz)

Power measurement using CY8CKIT-062S2-43012

5. Open a command prompt and ping the IP address displayed on the serial terminal: **`ping <IP address>`**
   The network stack is resumed. The device displays the Deep Sleep and Wi-Fi SDIO bus statistics on the terminal. The LED toggles after resuming the network stack, after which the network stack is again suspended until network activity is detected.

The host MCU will wake up on any network activity and not necessarily because of the ping from the PC. The reasons for network activity can be because of the broadcast or multicast packets issued by the AP. Further power savings can be made by employing offload features such as packet filtering, which will increase the time the host MCU will be in Deep Sleep.

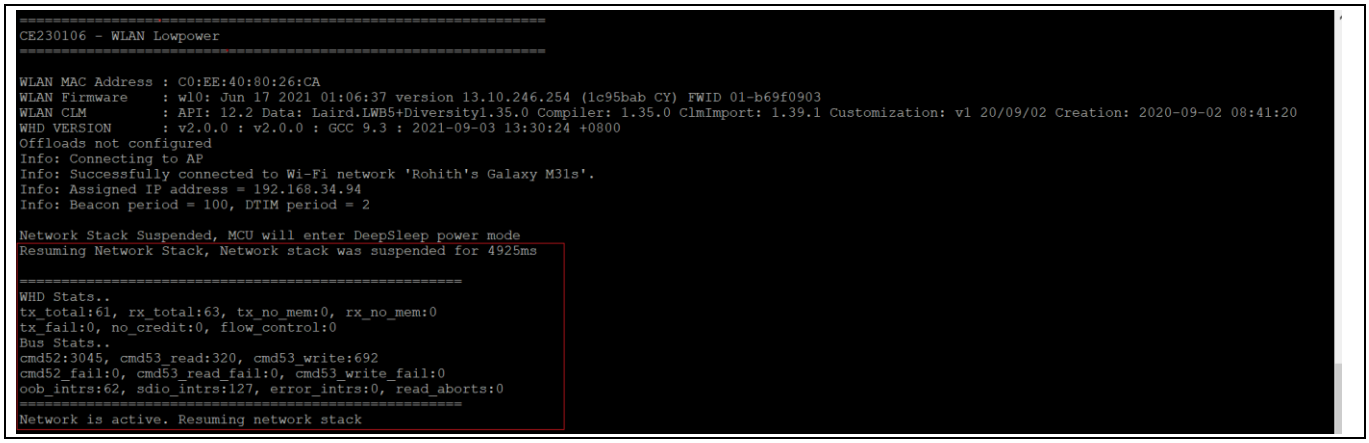

Figure 28 Resuming the network stack

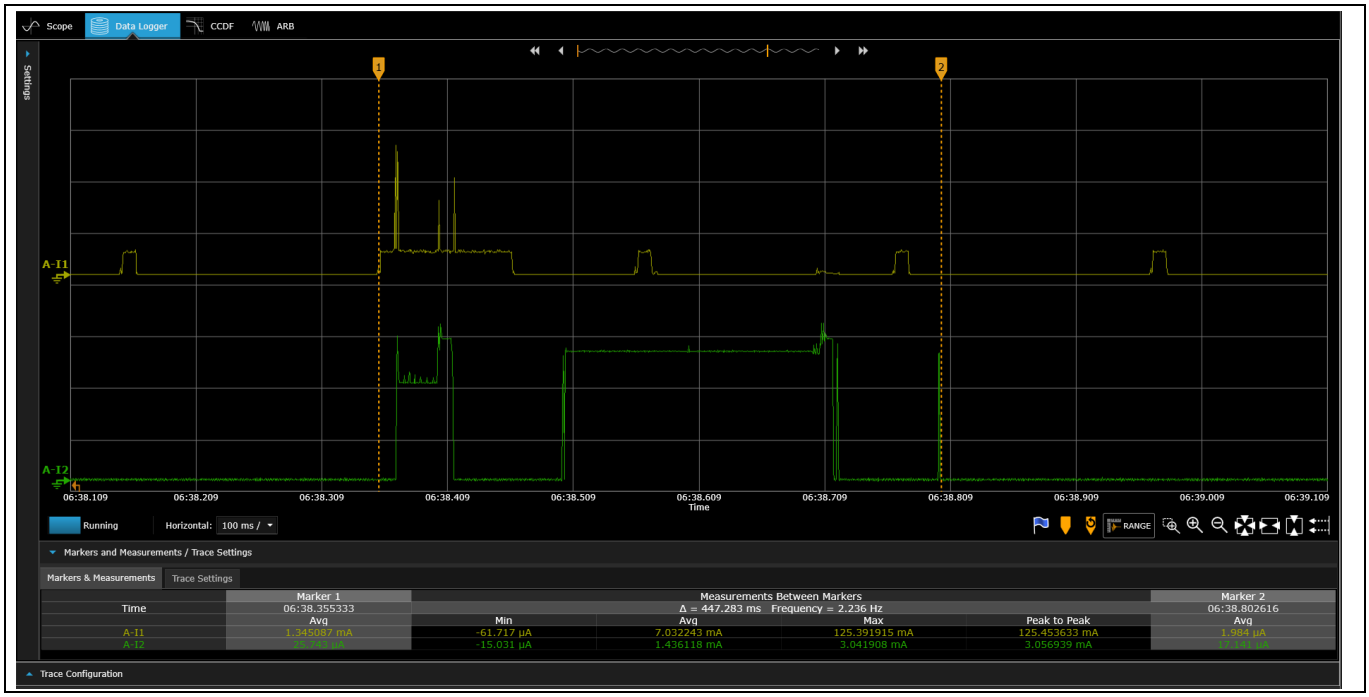

Figure 29 CY8CKIT-062S-43012 power consumption during ping activity

Power measurement using CY8CKIT-062S2-43012

Table 10 Typical current values for mtb-example-wifi-wlan-lowpower

State	Device	Current
Deep Sleep	PSoC™ 6 MCU	12.6 uA
	CYW43012 (VBAT)	2.4 uA
Average current for this AP configuration (2.4 GHz) beacon interval of 100 and AP DTIM of 1	PSoC™ 6 MCU	14 uA
	CYW43012 (VBAT)	317.5 uA
Average current for this AP configuration (2.4 GHz) beacon interval of 100 and AP DTIM of 2	PSoC™ 6 MCU	14 uA
	CYW43012 (VBAT)	225.7 uA
Average current for this AP configuration (2.4 GHz) beacon interval of 100 and AP DTIM of 3	PSoC™ 6 MCU	14 uA
	CYW43012 (VBAT)	112.6 uA
Average current for this AP configuration (5 GHz) beacon interval of 100 and AP DTIM of 1	PSoC™ 6 MCU	14 uA
	CYW43012 (VBAT)	279.2 uA
Average current for this AP configuration (5 GHz) beacon interval of 100 and AP DTIM of 2	PSoC™ 6 MCU	14 uA
	CYW43012 (VBAT)	95.5 uA
Average current for this AP configuration (5 GHz) beacon interval of 100 and AP DTIM of 3	PSoC™ 6 MCU	14 uA
	CYW43012 (VBAT)	69.2 uA

7.3 mtb-example-wlan-offloads

This section describes power measurement in CY8CKIT-062S2-43012 with the mtb-example-wlan-offloads code example. This code example demonstrates various WLAN offloads such as Address Resolution Protocol (ARP) offload, packet filter offload, and the TCP keepalive offload functionality offered by Infineon® AIROC™ Wi-Fi devices using PSoC™ 6 MCU.

The WLAN Offload functionalities allow the WLAN device to handle incoming TCP keepalive, Address Resolution Protocol (ARP), and packet filter packets from the network on its own so that the PSoC™ 6 MCU can remain longer in the low-power mode.

By default, all WLAN offloads are enabled.

1. Set up the CY8CKIT-062S2-43012 kit as explained in the Hardware setup. Note that this section uses the N6705B power analyzer from Keysight as a reference setup to measure power.
2. Turn on the channels to power PSoC™ 6 MCU and AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip from N6705B. You can connect the onboard KitProg to view logs in terminal software through the COM port.
3. Ensure that the configured Wi-Fi AP is in range and that the device connects to it.

Power measurement using CY8CKIT-062S2-43012

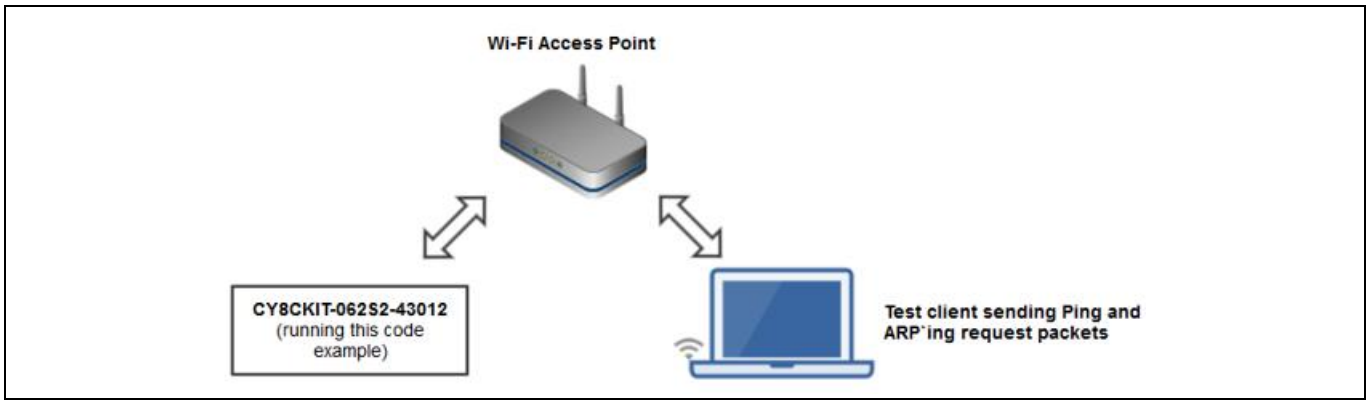

Figure 30 Test network setup

4. To realize the capability of WLAN offloads and their impact on the host MCU in terms of power savings, the current measurement has been done by considering the following cases.
5. To simulate a congested network environment, another Wi-Fi client device has associated to the same network to which the target kit has associated. In all these current measurement cases, the role of the client device is to send ping and ARP request packets periodically in its configured interval to the IP address of the target kit. Based on whether the LPA offloads are enabled, the host MCU stays in Deep Sleep power mode or wake up due to the ping and arp-ping requests from the client device.
6. WLAN offloads can be enabled or disabled in the Wi-Fi parameters using the Device Configurator.

Note: PSoC™ 6 MCU and Wi-Fi device current numbers were measured and averaged over 20 seconds in all the following cases. Current measurements are taken when the associated Wi-Fi client sends a ping request every 5 seconds and an arp-ping request every 10 seconds to the IP address of the target kit.

AP configuration:

- DTIM: 3
- Beacon Interval: 100 ms
- Band: 5 GHz

Case 1: All WLAN offloads enabled (by default).

- PSoC™ 6 MCU stays in Deep Sleep
- The WLAN device responds to ARP request packets
- The WLAN device discards ping request packets

Power measurement using CY8CKIT-062S2-43012

Figure 31 CY8CKIT-062S-43012 power consumption during ping activity

Table 11 Case 1: Power consumption

PSoC™ 6 MCU current	CYW43012 Wi-Fi current	PSoC™ 6 MCU + CYW43012 Wi-Fi current
27 uA	270 uA	297 uA

Case 2: ARP offload is disabled (disabled using Device configurator).

- PSoC™ 6 MCU wakes up to respond to ARP request packets
- The WLAN device discards ping request packets

Power measurement using CY8CKIT-062S2-43012

Figure 32 CY8CKIT-062S-43012 power consumption during ping activity

Table 12 Case 2: Power consumption

PSoC™ 6 MCU current	CYW43012 Wi-Fi current	PSoC™ 6 MCU + CYW43012 Wi-Fi current
66 uA	634 uA	700 uA

Case 3: ARP and packet filter offloads are disabled (disabled using Device configurator).

- PSoC™ 6 MCU wakes up to respond to ARP requests and ping request packets.

Power measurement using CY8CKIT-062S2-43012

Figure 33 CY8CKIT-062S-43012 power consumption during ping activity

Table 13 Case 3: Power consumption

PSoC™ 6 MCU current	CYW43012 Wi-Fi current	PSoC™ 6 MCU + CYW43012 Wi-Fi current
117 uA	1229 uA	1346 uA

7.4 mtb-example-btstack-freertos-cts-client

This section describes power measurement in CY8CKIT-062S2-43012 with the mtb-example-btstack-freertos-cts-client code example. This example allows you to measure the current in different states such as no RF activity, high-duty advertisement, low-duty advertisement, and in the connected state.

1. Set up the CY8CKIT-062S2-43012 kit as explained in the Hardware setup. Note that this section uses the N6705B power analyzer from Keysight as a reference setup to measure power.
2. Turn on the channels to power the PSoC™ 6 MCU and AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip from the power analyzer. You can connect the onboard KitProg to view logs in terminal software through the COM port.
3. On bootup, the AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip will not start any RF activity. You can note the PSoC™ 6 MCU and AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip current in this state. Press USER_BTN1 (SW2) to start a high-duty advertisement. In Figure 34, the yellow pulse represents the AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip current, and the green pulse represents the PSoC™ 6 MCU current when a high-duty advertisement is going on.

Power measurement using CY8CKIT-062S2-43012

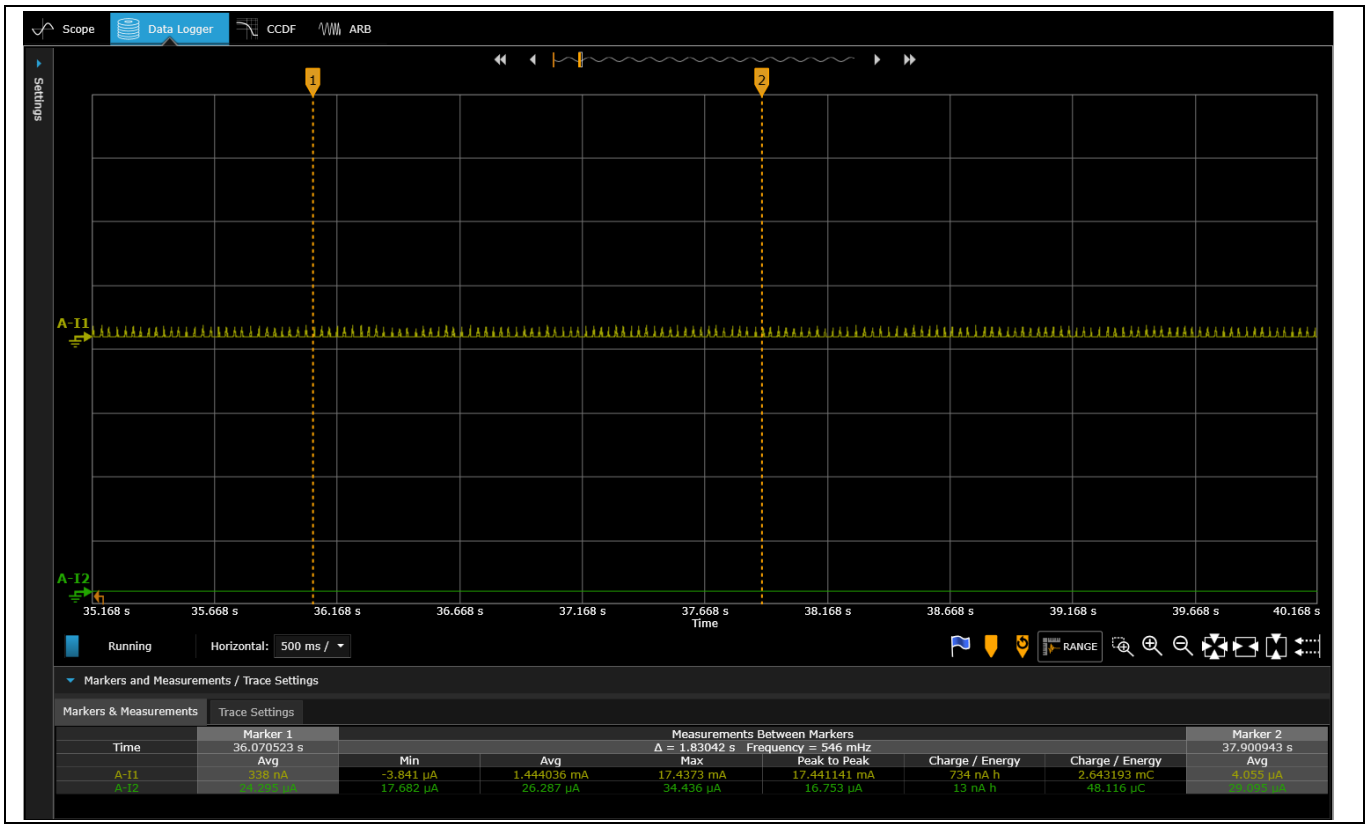

Figure 34 Current consumption during high-duty advertisement

After 60 seconds, the device will switch to the low-duty advertisement with an interval of 1.28 s. You can see the yellow pulses which indicate that the AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip woke up the PSoC™ 6 MCU to notify that low-duty advertisement has started. Figure 35 zooms in on the advertisement peaks during the low-duty advertisement.

Power measurement using CY8CKIT-062S2-43012

Figure 35 Current consumption during low-duty advertisement

From the graph, it may seem that the current in the low-duty advertisement period has a higher peak. This is because the device is in sleep mode and needs to wake up for every advertisement event. However, the average current is much lower compared to the high-duty advertisement period.

Power measurement using CY8CKIT-062S2-43012

Figure 36 Current consumption with connection interval of 10 ms

Figure 37 Current consumption with connection interval of 50 ms

Power measurement using CY8CKIT-062S2-43012

Figure 38 Current consumption with connection interval of 500 ms

Power measurement using CY8CKIT-062S2-43012

Figure 39 Current consumption with connection interval of 1000 ms

Table 14 Typical current values for mtb-example-btstack-freertos-cts-client

Bluetooth® state	Setting	Device	CY8CKIT-062S2-43012
High-duty advertisement	ADV interval: 30 ms	PSoC™ 6 MCU	26 uA
		CYW43012 (VBAT)	1.44 mA
Low-duty advertisement	ADV interval: 1.28 s	PSoC™ 6 MCU	26 uA
		CYW43012 (VBAT)	42.73 uA
Connected	Conn interval: 8.75 ms	PSoC™ 6 MCU	26 uA
		CYW43012 (VBAT)	1.15 mA
Connected	Conn interval: 10 ms	PSoC™ 6 MCU	26 uA
		CYW43012 (VBAT)	1.56 mA
Connected	Conn interval: 50 ms	PSoC™ 6 MCU	26 uA
		CYW43012 (VBAT)	402.5 uA
Connected	Conn interval: 500 ms	PSoC™ 6 MCU	26 uA
		CYW43012 (VBAT)	53.15 uA
Connected	Conn interval: 1 s	PSoC™ 6 MCU	26 uA
		CYW43012 (VBAT)	32.05 uA

Power measurement using CY8CKIT-062S2-43012

7.5 mtb-example-btstack-freertos-cts-server

This section describes power measurement in CY8CKIT-062S2-43012 with the mtb-example-btstack-freertos-cts-server code example. This example allows you to measure the current while the device is scanning in high-duty and in low-duty.

By default, the example implements the CTS service as a GATT server.

1. Set up the CY8CKIT-062S2-43012 kit as explained in the Hardware setup. Note that this section uses the N6705B power analyzer from Keysight as a reference setup to measure power.
2. Turn ON the channels to power the PSoC™ 6 MCU and AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip from the power analyzer. You can connect the onboard KitProg to view logs in terminal software through the COM port.
3. On bootup, the AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip will not have any RF activity. Press USER_BTN1 (SW2) to start high-duty scanning.

7.5.1 Power measurement

This example enables you to measure power in three different AIROC™ Bluetooth® LE states; standby, scanning, and connected states. Follow these steps to enter each state and measure power for different kits.

To enter different states

- **Standby state:** After programming the device, AIROC™ Bluetooth® LE is initialized and stays in the standby state. On the terminal, check for the message 'Bluetooth® stack initialization successful' and now you can measure power for the standby state.
- **Scanning state:** Press the user button(SW2) on your kit to start scanning. Note that the kit with CTS client CE must be advertising when it starts scanning. When it discovers the peer device, it sends a connection request. A high-duty scan will be performed initially for 30 seconds. Then the device switches to low-duty scanning without a timeout. High-duty scanning of 30 seconds is chosen for faster discovery. This configuration can be changed if required using the tool 'bt-configurator' that comes with ModusToolbox™ installation.
- **Connected state:** After the scanner finds the advertiser, the connection is established. The terminal displays the message 'Connected : BDA xx:xx:xx:xx:xx:xx'.

Power measurement using CY8CKIT-062S2-43012

Figure 40 Current consumption during high-duty scanning

Power measurement using CY8CKIT-062S2-43012

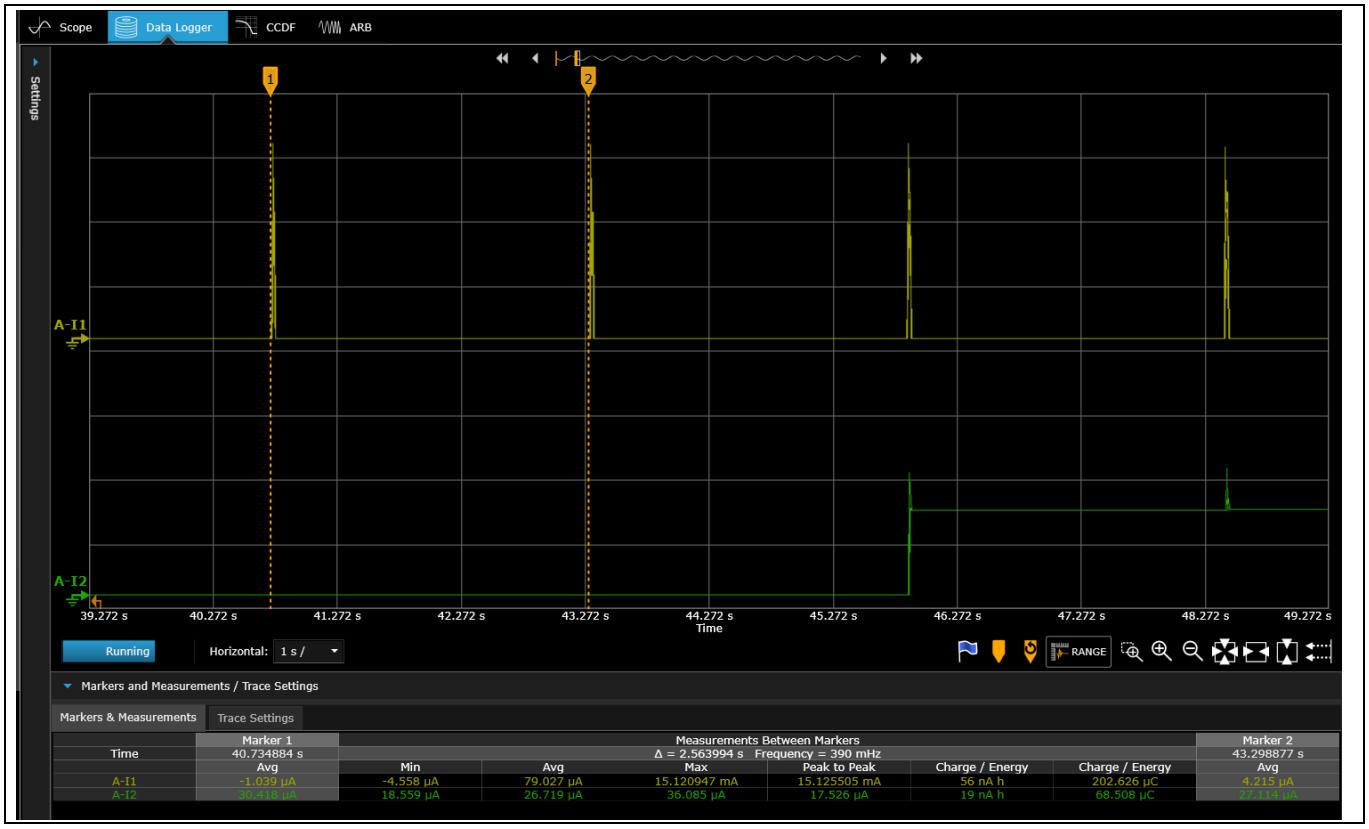

Figure 41 Current consumption during low-duty scanning

Table 15 Typical current values for mtb-example-btstack-freertos-cts-server

Bluetooth® state	Setting	Device	CY8CKIT-062S2-43012
High duty scanning	Scan interval: 120 ms	PSoC™ 6 MCU	26 uA
		CYW43012 (VBAT)	4.14 mA
Low duty scanning	Scan interval: 2560 ms	PSoC™ 6 MCU	26 uA
		CYW43012 (VBAT)	79.02 uA

8 Summary

The PSoC™ 6 MCU and AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip provide an industry-leading low-power IoT platform that provides many power management options. The Infineon® Low-power assistant middleware enables adding the low-power operation to your IoT design with ease. By following proper methods and design techniques, you can optimize your system for the lowest possible power consumption without degrading performance.

References

For a comprehensive list of PSoC™ 6 MCU resources, see KBA223067 in the Infineon community.

Application notes

[1] AN221774 – Getting started with PSoC™ 6 MCU

[2] AN218241 – PSoC™ 6 MCU hardware design considerations

[3] AN219528 – PSoC™ 6 MCU low-power modes and power reduction techniques

Code examples

[4] mtb-example-wifi-wlan-lowpower

[5] mtb-example-wlan-offloads

[6] mtb-example-btstack-freertos-cts-client

[7] mtb-example-btstack-freertos-cts-server

Device documentation

[8] PSoC™ 6 MCU: PSoC™ 62 datasheet

[9] PSoC™ 6 MCU: PSoC™ 62 architecture technical reference manual

[10] AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip datasheet

Development kits and documentation

[11] CY8CKIT-062S2-43012 PSoC™ 6 Wi-Fi Bluetooth® Pioneer Kit (with AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip)

Tools and documentation

[12] ModusToolbox™ software

[13] PSoC™ 6 peripheral driver library (PDL)

[14] Hardware abstraction layer (HAL) library

[15] Low power assistant

Revision history

Document revision	Date	Description of changes
**	2019-12-06	Initial release
*A	2021-02-24	Updated in Infineon template
*B	2021-09-22	Added content on TCP keepalive offload, Bluetooth® LE low power features, and instructions on using LPA in AWS IoT and FreeRTOS, and ModusToolbox™ software
*C	2023-05-29	Removed MBedOS information
*D	2024-03-04	Added note in section 7.2.

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2024-03-04
Published by

Infineon Technologies AG
81726 Munich, Germany

Do you have a question about this document?
Email: erratum@infineon.com

Document reference
002-27910 Rev. *D

Important notice
The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

Warnings
Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.