Chapter 5: Low Power for Wi-Fi Connected Devices

After completing this chapter, you will understand various low power concepts and how to use the Low Power Assistant (LPA).

Table of contents

Document conventions

Convention	Usage	Example
`Courier New`	Displays code and text commands	`CY_ISR_PROTO(MyISR);` `make build`
Italics	Displays file names and paths	*sourcefile.hex*
[**bracketed, bold**]	Displays keyboard commands in procedures	[**Enter**] or [**Ctrl**] [**C**]
Menu > Selection	Represents menu paths	**File > New Project > Clone**
Bold	Displays GUI commands, menu paths and selections, and icon names in procedures	Click the **Debugger** icon, and then click **Next**.

5.1 Low Power Assistant for Wi-Fi connected devices

In this section we will learn how the Low Power Assistant library and functionality helps you to minimize the MCU power consumption in a system connected to Wi-Fi.

The PSoC™ 6 communicates with the Wi-Fi device using an SDIO interface. In addition to the SDIO interface, there is a pin dedicated for use in low power. This pin allows the Wi-Fi device to wake the PSoC™ 6 host. In this way, the PSoC™ 6 can go to into a low power mode whenever its application allows, and the Wi-Fi device will wake it up only when it is needed to handle specific network activity.

Refer to the LPA Library Guide Part 2 for additional information.

- LPO_IN (32 kHz): 32 kHz sleep clock used for low-power WLAN operation
- SDIO: Clock, Data
- WL_HOST_WAKE: Interrupt line to wake the Host to service Wi-Fi requests

Once the wake pin is configured and the LPA library is available, you can offload various Wi-Fi operations such as responding to certain packet types so that they can either be handled by the Wi-Fi device or locally.

Exercise 1: Basic low-power for Wi-Fi applications

Rather that start from scratch, we'll start with the WLAN Low Power code example to see how basic low power host wake is configured. In later exercises, we'll add various offloads so that the Wi-Fi device can handle more operations without the host MCU being involved.

1. Create the *AnyCloud_WLAN_Low_Power* example for the CY8CKIT-062S2-43012 board using the Eclipse IDE. Name the application **ch05_ex01_wlan_low_power**.

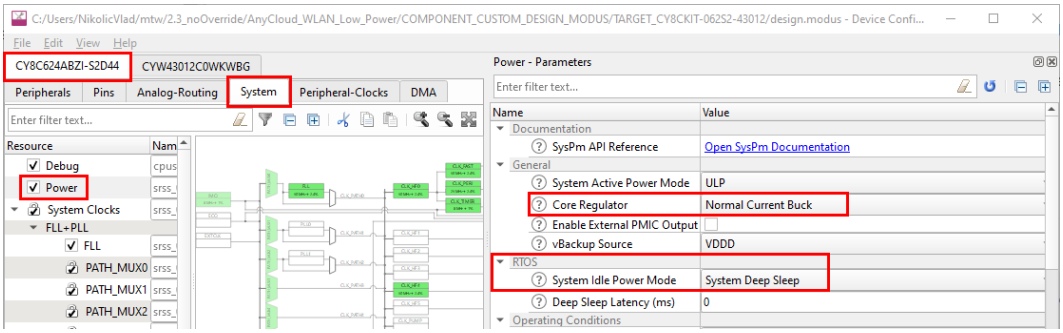
   Once the application has been created, notice that it has a directory called *COMPONENT_CUSTOM_DESIGN_MODUS*. Inside that directory is a directory for each TARGET supported by the CE. The *Makefile* has already been set up to use this custom configuration.

2. Open the custom *design.modus* file for your kit.

Notice that the Power settings for the PSoC™ 6 are configured such that the RTOS will use System Deep Sleep when it is idle. The required changes in the *FreeRTOSConfig.h* file to enable System Deep Sleep have already been done.

Also note that the Normal Current Buck regulator is used for this example.

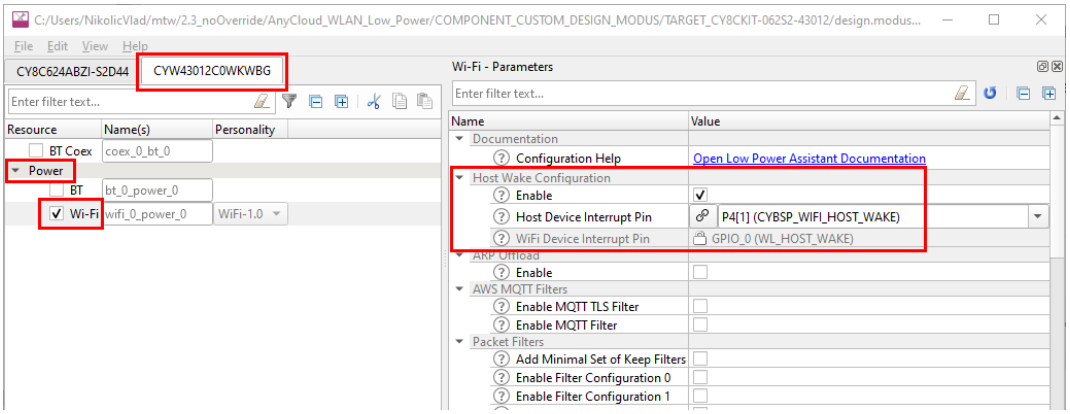
If you don't see "Normal Current Buck" you will need to update the personality. This is also indicated by a blue circle next to the resource name and info messages in the Notice List. To update the personalities, chose **File > Update All Personalities**.



3. Next, switch to the **CYW43012C0WKWBG** tab. This has settings for the WLAN device.

4. Click on **Power > Wi-Fi** to see the settings.

Notice that the Host Wake Configuration is enabled, and the Host Device Interrupt Pin is set to the correct pin for your kit. Also notice that all other Wi-Fi power controls are disabled.



With these settings, the WLAN device will not offload any tasks or filter any packets - it will wake the host CPU for any Wi-Fi activity. However, the PSoC™ 6 does configure the WLAN device for low power operation and suspends the network after periods of inactivity.

5. Open the file *lowpower_task.c* and look at the `lowpower_task` function.

Note that `lowpower_task` calls `wlan_powersave_handler` to configure the WLAN device for power save without throughput, power save with throughput, or power save disabled. It calls the appropriate `whd_wifi_*_powersave` functions to acomplish this. These functions are provided by the WHD library.

Also note that the `lowpower_task` will suspend the network when it has been inactive for a specified period of time by calling `wait_net_suspend`. This function is part of the LPA library.

6. Open the file `lowpower_task.h`. Update the **WIFI_SSID** and **WIFI_PASSWORD** to match your Wi-Fi network. Depending on your Wi-Fi router, you may also need to update **WIFI_SECURITY**.

Note: *See the `cy_wcm_security_t` structure in cy_wcm.h for possible security options - you can find it by right-clicking on `CY_WCM_SECURITY_WPA2_MIXED_PSK` and choosing "Open Declaration".*

```
#define WIFI_SSID                   "SSID"
#define WIFI_PASSWORD               "PASSWORD"

#define WIFI_SECURITY               CY_WCM_SECURITY_WPA2_MIXED_PSK
```

7. Open a UART terminal and then Build and Program the board.

8. Verify in the UART terminal that it successfully connects to Wi-Fi.

   You will see that the Wi-Fi device will wake the host MCU to service many different broadcast and multicast packets issued by the AP such as ARP (Address Resolution Protocol) requests. The frequency and number of events will depend on how heavy network traffic is.

9. Open a command terminal and ping the kit to verify that it responds. Check the UART terminal to see that the host wakes up to service the ping event.

   The IP address for your kit is displayed in the terminal window just after the kit connects to the Wi-Fi AP.

In the next exercises we will configure different offloads to further improve the time for which the host can remain in Deep Sleep.

Exercise 2: Configure packet filter offload

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

Packet filters are useful when:

- Trying to keep the host processor in System Deep Sleep for as long as possible.
- Trying to get the host processor into System Deep Sleep as soon as possible.

Whenever a WLAN packet is destined for the host, the WLAN processor must wake the host (if it is asleep) so it 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. For example, it 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, so the host is not woken.

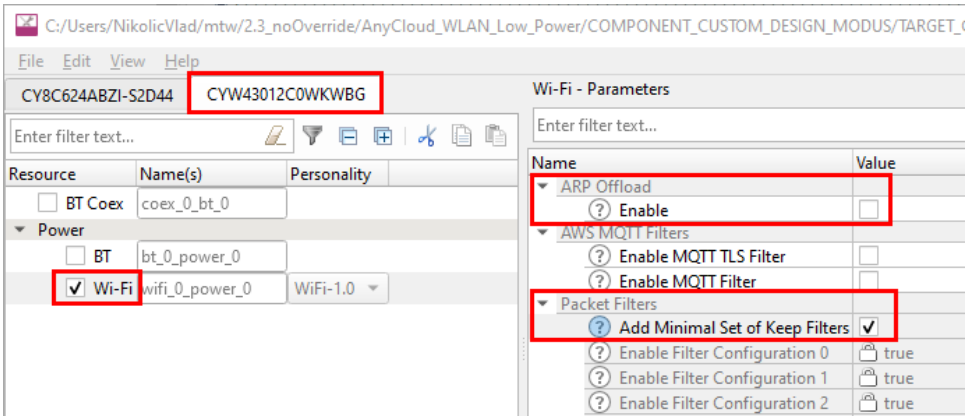
Refer to the Packet Filters Quick Start Guide for additional information.

You can either edit the previous exercise, or if you want separate copies you can create a new application from the previous exercise (use the **Import** button in the Project Creator to select the previous application). If you choose to do that, call the new application **ch05_ex02_packet_filter**.

1. Open the Device Configurator. Verify in the banner that you have the custom file for your application opened.
2. Switch to the connectivity device tab.

   Enable **Add minimal set of keep filters**. Keep **ARP Offload** disabled for now.

   These filters allow ARP, DNS, DHCP and 802.11x security packets to wake up the Host. These are needed to connect to a Wi-Fi Access point. Any other Wi-Fi packets will be dropped by the WLAN chip and not forwarded to the Host MCU (PSoC™ 6).



3. Save the configuration to generate the necessary code.
4. Open a UART terminal to see messages from the board.
5. Build the project and program. Observe the IP address assigned to your kit.
6. Open a command terminal and send a "ping" command to the board.

Observe the UART terminal or observe the power consumption to see that it does not wake up the PSoC™ 6 device since there is no "keep" packet filter for ICMP pings. Also observe that there is no response for the pings.

7. Send an "arping" command and observe that:

   - You get the responses.
   - Observe the UART terminal or observe the power consumption to see that the PSoC™ 6 MCU wakes up to service the ARP ping.

     Linux: arping is built-in

     macOS: you will need to install arping using brew. See:

     https://brewinstall.org/install-arping-on-mac-with-brew/

     Windows: you will need to download a tool to send arping requests. One such tool can be found here:

     https://elifulkerson.com/projects/arp-ping.php

     To use the Windows tool, download it onto your computer. Open a command terminal and go to the directory containing the downloaded file. Run the command:

     ```
     .\arp-ping.exe <ip address>
     ```

     In some cases, the arp-ping may respond with data from a cached ARP table on the PC instead of sending the request to the network. If that is the case, you can delete the table from the PC before each ARP ping by using the -d option, but this requires that you run the command terminal as administrator:

     ```
     .\arp-ping.exe -d <ip address>
     ```

     You can see what is in the cache by running:

     ```
     arp -a
     ```

5.1.1 ARP (Address Resolution Protocol) Offload

Recall that Wi-Fi devices will send broadcast ARP requests to all devices. When a device hears an ARP request for its IP address, it must respond with its MAC address - this is how devices figure out how to send packets through the network to the correct place. Devices often monitor and cache ARP responses for other devices that they hear on the network so that they can build up an ARP table of IP to MAC address that exist on the network. That way, they don't need to send ARP requests as often.

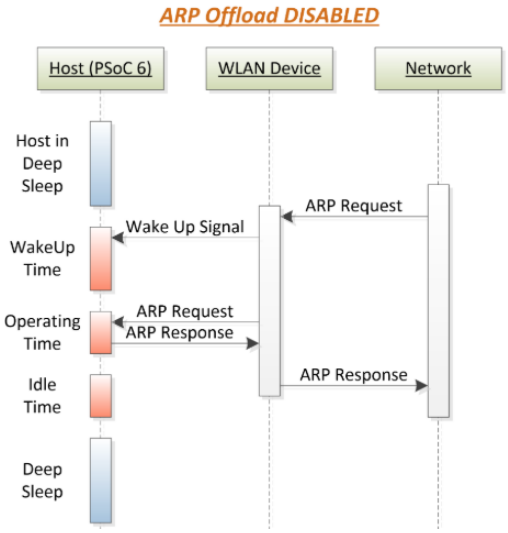
ARP traffic is normally forwarded from the WLAN device to the host MCU. If the Host is sleeping, the Device wakes it up. Having the Device handle some of the ARP traffic will reduce the frequency that the Host sleeps/wakes up, reducing Host power consumption by staying in CPU Sleep and System Deep Sleep states longer. By enabling ARP offload, the WLAN device will not wake the host for all ARP requests. Rather, it will send the response itself whenever an ARP request is addressed to it.

Likewise, when the host wants to send an ARP request, the WLAN device can respond directly without sending the request to the network if the WLAN device has the requested address cached.

The ARP offload features can be found in the LPA library documentation here: ARP Offload features. The main features are summarized below.

5.1.1.1 Disabled

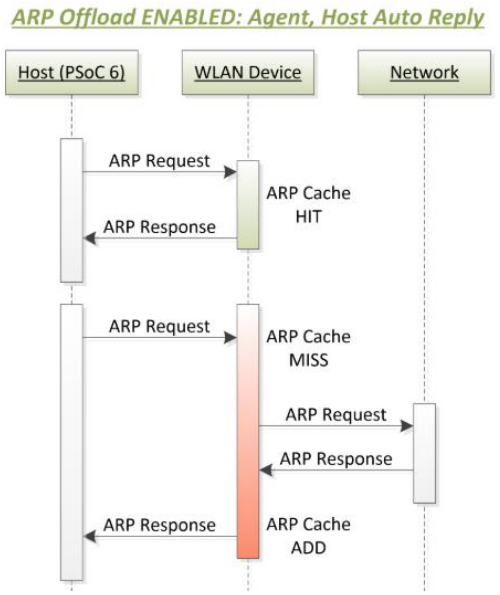
With ARP offload disabled, all ARP requests from the network will wake the host:



In addition, all ARP requests from the host will be sent out to the network by the WLAN device (this case is not shown).

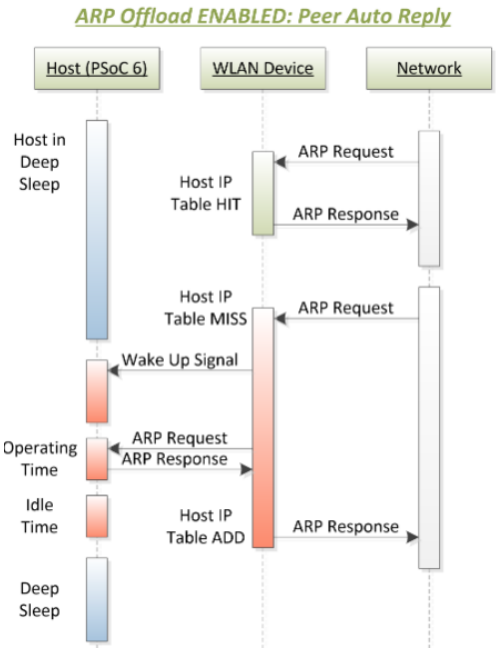
5.1.1.2 Host Auto Reply

Host Auto Reply is a power-saving and network traffic reduction feature. Using the ARP Offload Host Auto Reply feature, the WLAN device will answer ARP requests from the host MCU without broadcasting the request to the Network if it has the requested information in its cache:



5.1.1.3 Peer Auto Reply

Peer Auto Reply targets requests in the other direction. That is, the WLAN device will respond to ARP requests from Peers (i.e. from the network) without waking up the Host Processor:



5.1.1.4 Host IP Snoop

When enabled, the Snoop facility watches for ARP responses from the host to the network, and caches them in the WLAN Device Host IP Table. The size of this table is 8 entries, which allows for the Device to support multiple IP addresses.

5.1.1.5 ARP Offload Cache Entry Expiry

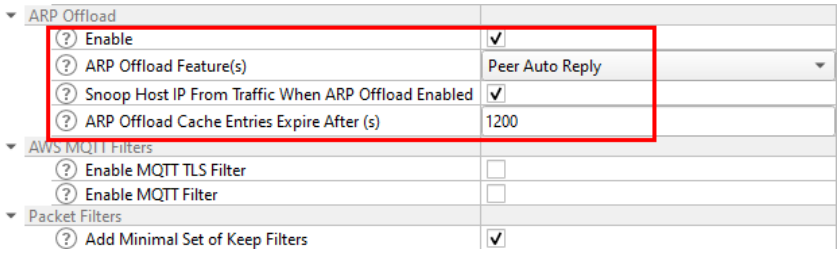
This is an "age out" value that you can set in the ARP Offload configuration to determine the length of time the ARP cache entry is valid. This ensures that the WLAN ARP cache is updated appropriately. Now let's add ARP Offload to our previous exercise.

Exercise 3: ARP Offload

You can either edit the previous exercise, or if you want separate copies you can create a new application from the previous exercise (run the Project Creator, select your kit, and then and use the **Import** button on the application selection screen to select the previous application). If you choose to do that, call the new application **ch05_ex03_ARP_offload.**

1. Open the Device Configurator. Verify in the banner that the correct file is opened.
2. Switch to the connectivity device tab.

   - Enable **ARP offload**.
   - Set **ARP offload Feature(s)** to "Peer Auto Reply".
   - Enable **Snoop Host IP From Traffic When ARP Offload Enabled**.
   - Set **ARP Offload Cache Entries Expire after (s)** to "1200".
   - Keep Packet Filters as configured.
   - Save your changes.



3. Build the project and program.
4. Send a "ping" command to the board and observe no change in behavior.

   In the UART terminal, you will see that it does not wake up the PSoC™ 6 device since there is no "keep" packet filter for ICMP pings. Observe the power consumption to see that the PSoC™ 6 MCU remains in Deep Sleep mode.

5. Send an "arping" command.

   - Observe that you still get the responses.
   - Observe the UART terminal and power consumption to see that the PSoC™ 6 MCU no longer wakes up.

   The PSoC™ 6 (Host) is offloaded from ARP packet handling and can remain in sleep.

Exercise 4: Allow ICMP (ping) packets through the filter

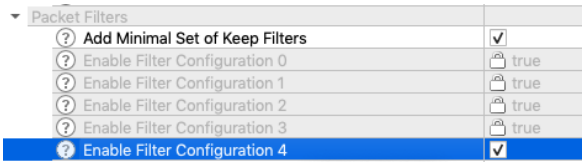
This explains how to modify Packet filters so that ICMP Ping packets are not filtered by the WLAN and are sent to the host.

You can either edit the previous exercise, or if you want separate copies you can create a new application from the previous exercise (use the Import button in the Project Creator to select the previous application). If you choose to do that, call the new application **ch05_ex04_allow_ping.**

1. Open the Device Configurator. Verify in the banner that you have the correct file opened.
2. Switch to the connectivity device tab and select Resource **Power > Wi-Fi**

   Check the box for **Enable Filter Configuration 4**.

   Note: Filter configurations 0-3 are locked because they are used for the minimal set of keep filters that you previously added.



3. Scroll down and set the configuration for filter 4 as follows.

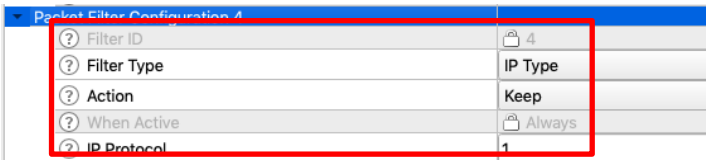
   Refer to the LPA documentation for details:

   Packet Filters Quick Start Guide

   Filter Types

   Filter Type set to “IP Type” (IP Protocol Filter)

   IP Protocol set to “1’ (1 refers to ICMP packet : List_of_IP_protocol_numbers)



4. Save your changes, build the project and program.
5. Send a "ping" command to the board and observe the UART.

   It now wakes the Host up because you have just added a "keep" packet filter for ICMP pings. You can see the response in the terminal and can observe increased power consumption upon receiving the ping packet.

6. Send an "arping" command.

   This behavior did not change; you get the responses to ARP in the PC terminal window while the PSoC™ 6 MCU remains in sleep.

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

Published by

Infineon Technologies AG

81726 Munich, Germany

IMPORTANT NOTICE

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics (“Beschaffenheitsgarantie”) .

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, 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.

In addition, any information given in this document is subject to customer’s compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer’s products and any use of the product of Infineon Technologies in customer’s applications.

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.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

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.