AIROC™ CYW9x907 Wi-Fi connectivity processor low-power modes

About this document

Scope and purpose

This application note introduces low-power modes in AIROC™ CYW9x907 Wi-Fi connectivity processors and describes how to adjust the build of an application to use low-power modes.

Intended audience

This application note is for anyone who uses low-power support for an application running on one of the following kits:

1. CYW943907AEVAL1F
2. CYW954907AEVAL1F

Table of contents

1 Introduction

The AIROC™ CYW9x907 Wi-Fi connectivity processor supports all rates specified in the IEEE 802.11 b/g/n specifications and includes an Arm® Cortex®-based applications processor, a single-stream IEEE 802.11n MAC/baseband/radio, a dual-band (2.4 GHz/5 GHz) transmit power amplifier (PA), and a receive low noise amplifier (LNA). Figure 1 shows the interconnections of all the major physical blocks in CYW43907 and their associated external interfaces.

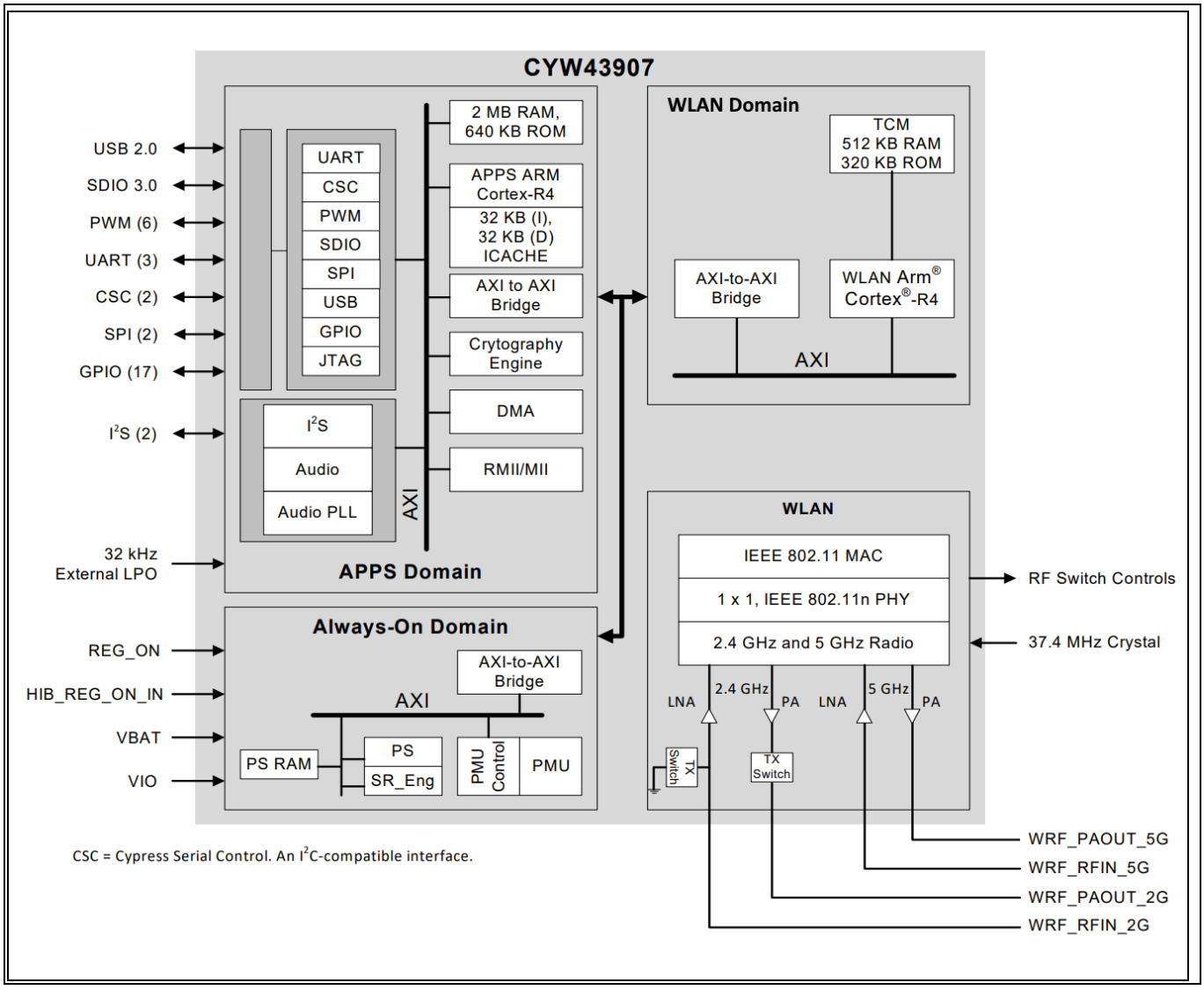

Figure 1 CYW43907 functional block diagram

2 Low-power mode features

This MCU platform has two CPUs:

1. Application processor (APP CPU)
2. Wireless LAN processor (WLAN CPU)

Power modes for both the CPUs are independent and can be used together or separately.

2.1 Application processor (APP CPU)

The APP CPU runs the user application. Two sleep modes (Sleep and DeepSleep) are used to reduce power by halting the processor for short periods of time. This is done automatically in conjunction with the RTOS. Additionally, the CPI has the Hibernation low-power mode, which stops the processor for a specified time and reboots it when the hibernation time is expired.

2.2 Wireless LAN processor (WLAN CPU)

The WLAN CPU is a separate CPU that communicates to the APP CPU through the M2M DMA interface, and controls the Wi-Fi radio. The APP CPU does not directly control the WLAN CPU; it only sends and receives commands and data. The WLAN CPU has its own internal low-power methodology. From the APP CPU, the application can send a command to indicate that the WLAN CPU is allowed to go into a low-power mode when it chooses do so. In addition, the application can send a request to select the low-power mode that the WLAN CPU uses.

3 APP CPU low-power mode settings

The setting for the APP CPU low-power mode is in *<app_root_directory>/bsps/TARGET_APP_CYW943907AEVAL1F/cybsp.h.*

Table 1 APP CPU low-power settings

Define	Notes
CY_CFG_PWR_MODE_ACTIVE	Active mode – do not go into any sleep mode
CY_CFG_PWR_MODE_SLEEP	Sleep mode – reduce power by putting processor to sleep
CY_CFG_PWR_MODE_DEEPSLEEP	DeepSleep – Sleep mode + stop the system clock
CY_CFG_PWR_DEEPSLEEP_LATENCY	Number of ticks required before going into DeepSleep

Code Listing 1 Power mode setting in *cybsp.h*

```
#define CY_CFG_PWR_MODE_ACTIVE          (0x00UL)
#define CY_CFG_PWR_MODE_SLEEP           (0x01UL)
#define CY_CFG_PWR_MODE_DEEPSLEEP       (0x02UL)

#ifndef CY_CFG_PWR_SYS_IDLE_MODE
#define CY_CFG_PWR_SYS_IDLE_MODE        (CY_CFG_PWR_MODE_ACTIVE)
#endif

#define CY_CFG_PWR_DEEPSLEEP_LATENCY    (24UL)  /* Min. 24 for this device */
```

3.1 Sleep

Sleep halts the processor and reduces the power consumption by approximately 5% until there is an interrupt which brings the system back to full power. When enabled, the RTOS determines when the user application would be idle, and puts the APP CPU into sleep mode, reducing the overall power consumption.

3.2 DeepSleep

DeepSleep stops the system clock and reduces the power consumption by 75% until there is an interrupt which brings the system back to full power. When enabled, the RTOS determines when the application would be idle, and puts the APP CPU into DeepSleep mode, reducing the overall power consumption.

3.2.1 DeepSleep latency

The latency is used to determine whether there is a reasonably long enough time to go into DeepSleep, and is measured in milliseconds. This is only used in multi-threaded applications using an RTOS. The RTOS determines how much time there is before the next task runs. If that time exceeds the latency value, the system goes into DeepSleep mode.

3.3 Hibernation

Hibernation uses an external source to wake the system. When hibernating, the device shuts down all peripherals and reduces the power consumption to the lowest possible value while still retaining some control as to when the device wakes up. This means that the current state of the device (RAM, etc.) is lost. Table 2 indicates the hibernation wakeup source. The code needed to enter Hibernation mode is shown in Code Listing 2.

APP CPU low-power mode settings

Table 2 Hibernation wakeup source

Define	Notes
CYHAL_SYSPM_HIBERNATE_WDT	Configures the WDT interrupt as the wakeup source. This is the only wakeup source currently supported.

Code Listing 2 Calling hibernate from main()

```
#include "cyhal_syspm.h"

cy_rslt_t result;
result = cyhal_syspm_hibernate(cyhal_syspm_hibernate_source_t wakeup_source);
```

4 WLAN CPU low-power modes

The WLAN CPU runs a fixed firmware image that is loaded at power-up of the device. The WLAN CPU runs independently of the APP CPU and talks to the APP CPU through a dedicated M2M DMA channel. The application running on the APP CPU can send commands to the WLAN CPU that allows the WLAN CPU to determine when to go into a low-power mode.

Table 3 WLAN CPU power mode settings

Define	Description
CY_WCM_NO_POWERSAVE_MODE	No power savings.
CY_WCM_PM1_POWERSAVE_MODE	Powersave mode on specified interface without regard for throughput reduction.
CY_WCM_PM2_POWERSAVE_MODE	Powersave mode on specified interface with high throughput.

4.1 Default WLAN low-power mode

The default for WLAN low-power mode is CY_WCM_NO_POWERSAVE_MODE. To change modes, the application must call the function to allow the WLAN to change power mode.

Code Listing 3 Allowing WLAN power modes in the user application

```
#include "cy_wcm.h"

Cy_rslt_t result;
result = cy_wcm_allow_low_power_mode(CY_WCM_PM1_POWERSAVE_MODE);
```

4.2 PM1 powersave mode

This mode saves the most power at the expense of some data throughput. The integrity of the Wi-Fi connection is not compromised, but some operations may take slightly longer than in “no powersave” mode.

4.3 PM2 powersave mode

This mode saves less power than PM1, but keeps the data throughput at a higher level. The integrity of the Wi-Fi connection is not compromised

5 HAL functions

The Hardware Abstraction Layer (HAL) API provides functions for applications to affect the power mode of the device. See Hardware Abstraction Layer power management API for more information.

When Sleep or DeepSleep occurs, the code hits an Arm® Cortex®-R4 “Stop and Wait for Interrupt” (SWI) instruction. This instruction halts the processor until an interrupt allows it to continue.

5.1 Single-threaded application

Single-threaded applications do not use an RTOS, and must call the sleep functions directly. The application must also provide a way for the APP CPU to resume.

5.1.1 Single-threaded application using SW1 interrupt

Do the following to use the button marked “SW1” on the CYW943907AEVAL1F board to trigger an interrupt that wakes up the APP CPU from Sleep or DeepSleep:

1. Enable an SW1 event to produce an interrupt when pressed.

Code Listing 4 Enable button press for interrupt to wake from Sleep/DeepSleep

```
#include "cyhal_gpio.h"

    /* Configure GPIO button for an interrupt to wake from sleep / deepsleep. */
    cyhal_gpio_init(CYBSP_SW1, CYHAL_GPIO_DIR_INPUT, CYHAL_GPIO_DRIVE_PULLUP,
                    CYBSP_BTN_OFF);
    cyhal_gpio_enable_event(CYBSP_SW1, CYHAL_GPIO_IRQ_FALL, 7, true);
```

2. Call the Sleep (or DeepSleep) function.

Code Listing 5 Calling the Sleep function

```
#include "cyhal_syspm.h"

    /* Call sleep – must wake through interrupt */
    cyhal_syspm_sleep();
```

Code Listing 6 Calling the DeepSleep function

```
#include "cyhal_syspm.h"

    /* Call deepsleep – must wake through interrupt */
    cyhal_syspm_deepsleep();
```

3. Press SW1 on the board to exit Sleep or DeepSleep.

5.1.2 Single-threaded application using tickless Sleep or DeepSleep

The following code listings show the steps to use the low-power timer (lptimer) and the tickless sleep functions to enable the APP CPU to enter Sleep or DeepSleep mode for a specified time. The APP CPU Sleep mode may end earlier than the specified time due to an interrupt. The value returned in the `actual_sleep_ms` argument reflects the time (in milliseconds) that the APP CPU was in Sleep or DeepSleep mode.

1. Initialize the low-power timer to use tickless Sleep or DeepSleep.

Code Listing 7 Initializing the low-power timer

```
#include "cyhal_lptimer.h"
cyhal_lptimer_t lptimer;

    /* Initialize the low power timer */
    cyhal_lptimer_init(&lptimer);
```

2. Put the APP CPU into Sleep or DeepSleep.

Code Listing 8 Using tickles Sleep

```
uint32_t desired_ms;
uint32_t actual_sleep_ms;

    /* Call tickless sleep – will wake after specified time */
    desired_ms = 1000;
     cyhal_syspm_tickless_sleep(&lptimer, desired_ms, &actual_sleep_ms);
```

Code Listing 9 Using tickless DeepSleep

```
uint32_t desired_ms;
uint32_t actual_sleep_ms;

    /* Call tickless deepsleep – will wake after specified time */
     desired_ms = 1000;
    cyhal_syspm_tickless_deepsleep(&lptimer, desired_ms, &actual_sleep_ms);
```

The device wakes up after the specified `desired_ms` time.

5.2 Multi-threaded applications

In multi-threaded applications, the user application does not call the sleep functions directly. They are called from the RTOS in the idle task. Code Listing 10 and Code Listing 11 show how to enable the RTOS to call Sleep and DeepSleep mode.

Code Listing 10 Enable RTOS to Sleep during APP CPU idle

```
<root of application>/bsps/TARGET_APP_CYW954907AEVAL1F/cybsp.h

#ifndef CY_CFG_PWR_SYS_IDLE_MODE
#define CY_CFG_PWR_SYS_IDLE_MODE        (CY_CFG_PWR_MODE_SLEEP)
#endif
```

Code Listing 11 Enable RTOS to call DeepSleep during APP CPU idle

```
<root of application>/bsps/TARGET_APP_CYW954907AEVAL1F/cybsp.h

#ifndef CY_CFG_PWR_SYS_IDLE_MODE
#define CY_CFG_PWR_SYS_IDLE_MODE        (CY_CFG_PWR_MODE_DEEPSLEEP)
#endif

#define CY_CFG_PWR_DEEPSLEEP_LATENCY    (24UL)  /* Min. 24 for this device */
```

5.3 Lock/Unlock DeepSleep

If the user application needs a thread to delay, but does not want to put the device into DeepSleep, call lock and unlock around the protected code.

Use the following functions to lock or unlock DeepSleep:

- `cyhal_syspm_lock_deepsleep()`: This function disallows the device from entering DeepSleep until the unlock function is called. The function uses a counter so that multiple threads can call this function. Until all threads unlock DeepSleep, the device will not go into DeepSleep.
- `cyhal_syspm_unlock_deepsleep()`: This function decrements the counter. When all threads have called unlock, the RTOS can initiate putting the device into DeepSleep when the correct conditions are met.

Code Listing 12 Locking DeepSleep

```
/* Lock device from entering deepsleep for this section of code */
cyhal_syspm_lock_deepsleep();

    /*
     * Deepsleep will not occur in this section of code
     */

/* Allow deepsleep again */
cyhal_syspm_unlock_deepsleep();
```

References

[1] Hardware Abstraction Layer Power Management API

[2] Infineon AN214828 Power Consumption Measurements Application Note

[3] Low Power Current Measurements Using CYW943907AEVAL1F

Revision history

Document revision	Date	Description of changes
**	2023-05-11	Initial release.

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

Edition 2023-05-11

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email: erratum@infineon.com

Document reference

002-37426 Rev. **

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.