WICED Studio

WICED™ Development Powersave System

Associated Part Family: BT CYW2070x

Doc. No.: 002-19004 Rev. *A

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
www.cypress.com

Contents

About This Document **3**

Purpose and Scope 3
Audience 3
Acronyms and Abbreviations 3
IoT Resources and Technical Support 3
Document Conventions 3

1 802.11 (Wi-Fi) Power Saving Overview **4**

1.1 Listen Interval 4
1.2 DTIM Period 4
1.3 Practical Power Saving 4
1.3.1 Power Save Poll 5
1.3.2 802.11 Power Save without Poll 5
1.3.3 Association Timeout Limit 5
1.4 Final Note 5

2 WICED Power save Implementation **6**

2.1 Wi-Fi Power Saving 6
2.2 Microprocessor Power Saving 6
2.2.1 How to control MCU Powersave 7
2.2.2 RTOS Operation with MCU Powersave Mode 7

3 Example Power Measurements **9**

3.1 The Big Picture 9
3.2 DTIM Wake-up 10
3.3 Ping Transaction 11

4 Low Power Measurement Techniques **12**

4.1 Introduction 12
4.2 Measuring Low-Range Sleep Current 12
4.2.1 Custom Low-Power Application 12
4.2.2 WICED Module Modifications 13
4.2.3 WICED Evaluation Board Modifications 13
4.2.4 Measurement Technique 13
References 14

Document Revision History **15**

Worldwide Sales and Design Support **16**

Products 16
PSoC® Solutions 16
Cypress Developer Community 16
Technical Support 16

About This Document

Purpose and Scope

This document describes the power saving implementation and features of the Cypress Wireless Internet Connectivity for Embedded Devices (WICED™; pronounced “wicked”) Development System for Wi-Fi stations.

Note: This document applies to **WICED-SDK-2.4.0**

Audience

This document is for software engineers who are using the WICED Development System to create low power applications for embedded wireless devices.

Acronyms and Abbreviations

In most cases, acronyms and abbreviations are defined on first use.

For a comprehensive list of acronyms and other terms used in Cypress documents, go to www.cypress.com/glossary.

IoT Resources and Technical Support

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

Document Conventions

The following conventions may be used in this document:

Convention	Description
Bold	Buttons, tabs, lists and other GUI items: click Next, select the Startup tab
`Monospace`	Command lines and application outputs: `snip.scan-BCM943362WCD4 download run`
< >	Placeholders for required elements: <WICED-SDK>
‘ ’	Application Names, Configuration Parameters: ‘YOUR_AP_SSID’

1 802.11 (Wi-Fi) Power Saving Overview

The subsections in this chapter are partly based on information contained in *802.11 Wireless Networks, The Definitive Guide [1]*. Developers planning to integrate products using 802.11 Wi-Fi are highly encouraged to purchase a copy of this book.

The 802.11 standard includes a number of parameters that allow stations to save power, although power saving is accomplished at the expense of throughput or latency to the station.

1.1 Listen Interval

To save power, Wi-Fi stations can sleep by powering down most of the Wi-Fi subsystem. While stations are asleep, access points (APs) must buffer frames for them. Sleeping stations periodically wakeup to listen to traffic announcements to determine whether the access point has any buffered frames. When stations associate with an AP, part of the data in the Association Request frame is the Listen Interval. The Listen Interval is used to indicate to the AP how often a power save station will wake to listen to Beacon frames.

An AP may use the Listen Interval as a guide to how long it should retain buffered frames for a power save station. Larger listen intervals require more AP memory for frame buffering.

In reality, APs generally do not pay much attention to the listen interval requested by a station. If a station sets the listen interval to a particular value, there is no guarantee that the AP will buffer all of the packets received for the station while the station is asleep. Wi-Fi Certification test plans do not currently test behavior related to the Listen Interval, and implementation of Listen Interval functionality is not enforced.

Most APs enforce an association timeout on stations, i.e. 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 will be disassociated. The association timeout cannot be negotiated by the station.

1.2 DTIM Period

The DTIM period is a parameter associated with an infrastructure network, and is advertised in an access point Beacon frame. All Beacon frames include a Traffic Indication Map (TIM) which indicate to stations that buffered frames are available. Unicast frames buffered for individual stations are delivered in response to a query from the station. This polled approach is not suitable for multicast and broadcast frames though, 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 Delivery TIM (DTIM).

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

The default DTIM beacon interval for most APs is either DTIM=1 or DTIM=3. In the case of DTIM=3, the station need only wake from low power mode to receive every third beacon and any ensuing queued broadcast or multicast traffic.

1.3 Practical Power Saving

A number of mechanisms may be used by Wi-Fi stations to save power.

Stations 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. Stations requiring higher throughput, wireless speakers for example, may consider using a non-standard 802.11 powersave mechanism. These methods are described in this section.

Stations requiring a more fine-grained powersave mechanism to differentiate power save for various traffic priorities may consider using WMM-Powersave. WMM Powersave is not currently supported by the WICED SDK however and is not described further in this document.

1.3.1 Power Save Poll

Power Save Poll is suited to stations that primarily transmit data to the Wi-Fi network at low duty cycle. The PS-Poll mechanim 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 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 frame(s). The wake period depends on the configured listen interval and whether or not the STA is configured to receive group addressed frames from a beacon containing a DTIM.
4. If the AP indicates it has buffered frame(s) 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 may decide to send another PS-Poll frame to retrieve additional buffered frames (if the More Data bit was set) or return to sleep.

1.3.2 802.11 Power Save without Poll

A non-standard mechanism known broadly as 802.11 Power Save without Poll (PS-non-Poll) enables Wi-Fi stations 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 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 frame(s) 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 the STA is awake and sends buffered frame(s).
6. STA receives the frame(s), waits for a timeout period to receive additional frame(s) (if available), then returns to sleep as described in Step 1.

1.3.3 Association Timeout Limit

If a station 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.

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

The Wi-Fi Alliance is currently working to adopt various new power save features that are in the 802.11v standard.

1.4 Final Note

Despite the fact that 802.11 transmit power consumption is at least five times higher than receive power consumption, for even medium transmit duty cycle applications, much of the energy in a battery powered Wi-Fi station is consumed by the receiver. Unless powersave 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 130mW of receiver power consumption to flatten a battery.

The key to minimizing power consumption is to minimize the ON duty cycle of the system. Minimizing the integrated area under the current consumption curve is critical to obtaining maximum life from a battery.

2 WICED Power save Implementation

The power saving implementation provided in the WICED Development System attempts to minimize the combined ON time of the Wi-Fi chip and the microprocessor.

2.1 Wi-Fi Power Saving

There are eight WICED API functions available to control power saving on the Wi-Fi chip. These functions are described in Table 2-1.

Table 2-1. WICED Wi-Fi API Functions to control Power Saving

`wiced_init()`	Powers up the Wi-Fi chip and initializes the SPI/SDIO bus interface, RTOS and networking interface.
`wiced_wifi_enable_powersave()`	Enable PS-Poll mode on the Wi-Fi chip (see Section 1.3.1)
`wiced_wifi_enable_powersave_with_throughput()`	Enable 802.11 PS-non-Poll mode on the Wi-Fi chip (see Section 1.3.2). Use this mode when it is important to maintain throughput.
`wiced_wifi_disable_powersave()`	Disable powersave mode on the Wi-Fi chip
`wiced_wifi_set_listen_interval()`	Set the number of beacons to skip while the Wi-Fi chip is sleeping (only works when Wi-Fi powersave is enabled)
`wiced_wifi_set_listen_interval_assoc()`	Tells the AP how many beacons will be skipped while the Wi-Fi chip is sleeping. The number of beacons actually skipped is set by the `wiced_wifi_set_listen_interval()` function (only works when Wi-Fi powersave is enabled)
`wiced_wifi_get_listen_interval()`	Returns information about the number of beacons to skip while the Wi-Fi chip is sleeping
`wiced_deinit()`	Cuts power to the Wi-Fi chip entirely. Among other tasks, this function de-initialiases network interfaces and threads associated with the wireless network interface.

When the Wi-Fi chip is enabled for powersave, it sleeps between DTIMs in a manner similar to the description at the beginning of Section 1.3. If the beacon listen interval is subsequently configured to a value other than zero, the Wi-Fi chip only wakes up according to the value configured. It no longer wakes at DTIM intervals.

2.2 Microprocessor Power Saving

The description in this section applies to both STM32 and AT91SAM4S microprocessors. The powersave mode used by WICED for STM32 is Stop Mode, and for AT91SAM4S is Wait Mode. In the following text, Stop Mode and Wait Mode are generally referred to as Powersave Mode. In Powersave mode, the MCU retains the entire contents of RAM.

There should be no noticeable difference in the operation of an application when MCU powersave is enabled, other than a significant reduction in current consumption when the processor idles, and a possible reduction in maximum network data throughput.

2.2.1 How to control MCU Powersave

There are four WICED API functions available to control power saving on the MCU. These functions are described in Table 2-2.

Table 2-2. WICED API Functions to control MCU Power Saving

`wiced_platform_mcu_enable_powersave()`	Enables powersave mode on the MCU
`wiced_platform_mcu_disable_powersave()`	Disables powersave mode on the MCU
`wiced_network_suspend()`	Suspends network services and disables all network related timers. This API call prevents the MCU from unnecessarily waking to service high frequency network timers
`wiced_network_resume()`	Resumes network services

Note:

- If an application uses interrupts, other than external GPIO interrupts or the RTC interrupt, these interrupts will not be detected or serviced while the MCU is in Powersave Mode.
- If the global `#define WICED_DISABLE_MCU_POWERSAVE` is enabled in `<WICED-SDK>/include/wiced_defaults.h`, the API functions in Table 2-2 have no effect.
- When `wiced_platform_mcu_enable_powersave()` is called, the MCU might not immediately enter powersave mode. The behaviour of powersave mode is described in Section 2.2.2 for FreeRTOS and ThreadX separately.

2.2.2 RTOS Operation with MCU Powersave Mode

In MCU Powersave mode, all clocks except the 32kHz Real Time Clock (RTC), are powered down. All memory and registers retain state. There are only two ways to exit Powersave Mode, an external interrupt (i.e. an edge on a GPIO pin) or an RTC interrupt.

With Powersave Mode enabled, the WICED Wi-Fi Driver enables an Out-of-Band (OOB) interrupt on an external GPIO from the Wi-Fi chip. The OOB interrupt is required because the SDIO bus interface clock and interrupt handling on the MCU are disabled when Powersave Mode is active. There is no additional requirement if the Wi-Fi chip uses gSPI to communicate with the MCU, since a GPIO interrupt is required for normal communication.

FreeRTOS Implementation

A platform specific function `platform_power_down_hook()` is integrated into the FreeRTOS idle thread to make applications work almost independently of Powersave Mode.

The FreeRTOS scheduler normally runs the idle thread every system tick when no other thread is ready to run. Rather than running the idle thread when powersave is enabled, the processor is placed in Powersave Mode. The following steps describe the process.

1. The Idle thread determines the next pending timeout for delayed threads and calls the platform specific idle handler `platform_power_down_hook()` function.
2. The idle handler checks whether any threads have disabled Powersave Mode and if this is the case, power down is disabled. Otherwise …
3. The idle handler programs the RTC with the next pending timeout value.
4. The idle handler directs the MCU to enter Powersave Mode until the next interrupt.
5. Some time later, an interrupt occurs either from an external GPIO pin or from the RTC timeout.
6. The processor wakes and the idle handler disables the RTC interrupt.
7. The idle handler reads the time elapsed from the RTC and returns it to the idle thread.
8. The idle thread uses the elapsed time to instigate any processing that was missed (eg. update the tick count).
9. The scheduler runs to start any threads that are now ready to run.

ThreadX Implementation

Unlike FreeRTOS, ThreadX does not provide an idle thread mechanism. ThreadX waits in the SVC software interrupt handler if there is no other task ready to run. The power save mechanism for ThreadX is described in the following text.

1. If no task is ready to run, the wait routine inside the SVC handler calls a ThreadX function `tx_low_power_enter()`.
2. The `tx_low_power_enter()` function checks how much time remains until the next ThreadX timer is due to expire. This timer expiry value is passed to the platform specific idle handler `platform_power_down_hook()`
3. The idle handler checks whether all threads mutually agree to enter Powersave Mode. If there is no agreement, a WFI instruction is called which makes the processor enter idle mode, leaving MCU peripherals powered up. The processor returns from idle mode as soon as one of the available interrupts is triggered. If all threads agree to enter Powersave Mode …
4. The idle handler programs the RTC with the next pending timeout value.
5. The idle handler directs the MCU to enter Powersave Mode until the next interrupt.
6. Some time later, an interrupt occurs either from an external pin or from the RTC timeout.
7. The processor wakes and the idle handler disables the RTC interrupt.
8. The idle handler reads the time elapsed from the RTC.
9. The `tx_low_power_enter()` function passes the time elapsed to another ThreadX function `tx_time_increment()`. This function updates the timer list and determines whether any tasks are ready to run.
10. The SVC handler restores the context of the task (if any) and the processor starts execution.

3 Example Power Measurements

The plots provided in this section show various aspects of the power consumption using the WICED Ping Powersave snippet application running on a BCM943362WCD4 WICED module acting as a Wi-Fi station. The application sends a Ping to the gateway at one second intervals with Wi-Fi powersave enabled. The RTOS/TCP stack used to take the measurements is FreeRTOS/LwIP, however ThreadX/NetX and ThreadX/NetXDuo yield virtually identical results. All plots show current consumption separated out for the BCM43362 Wi-Fi chip and STM32F205 microprocessor. The power supply to the module is 3.3V.

3.1 The Big Picture

Figure 3-1 is a 5 second capture of current consumed when the application sends an ICMP Ping to the AP at intervals of approximately 1 second. The large blue spikes are Wi-Fi packet transmissions related to the Ping transmission. The smaller blue spikes at intervals of approximately 300ms occur when the Wi-Fi wakes to listen to the AP for a DTIM beacon.

Red spikes indicate when the MCU is awake. Nearly all MCU wakeups coincide with the transmission of a ping packet. The red spike coinciding with the small blue spike just after 4000ms is a result of the Wi-Fi chip waking the MCU to pass a received packet to the network stack, in this case possibly an ARP packet.

The code used to take the following measurement is provided in Figure 3-2.

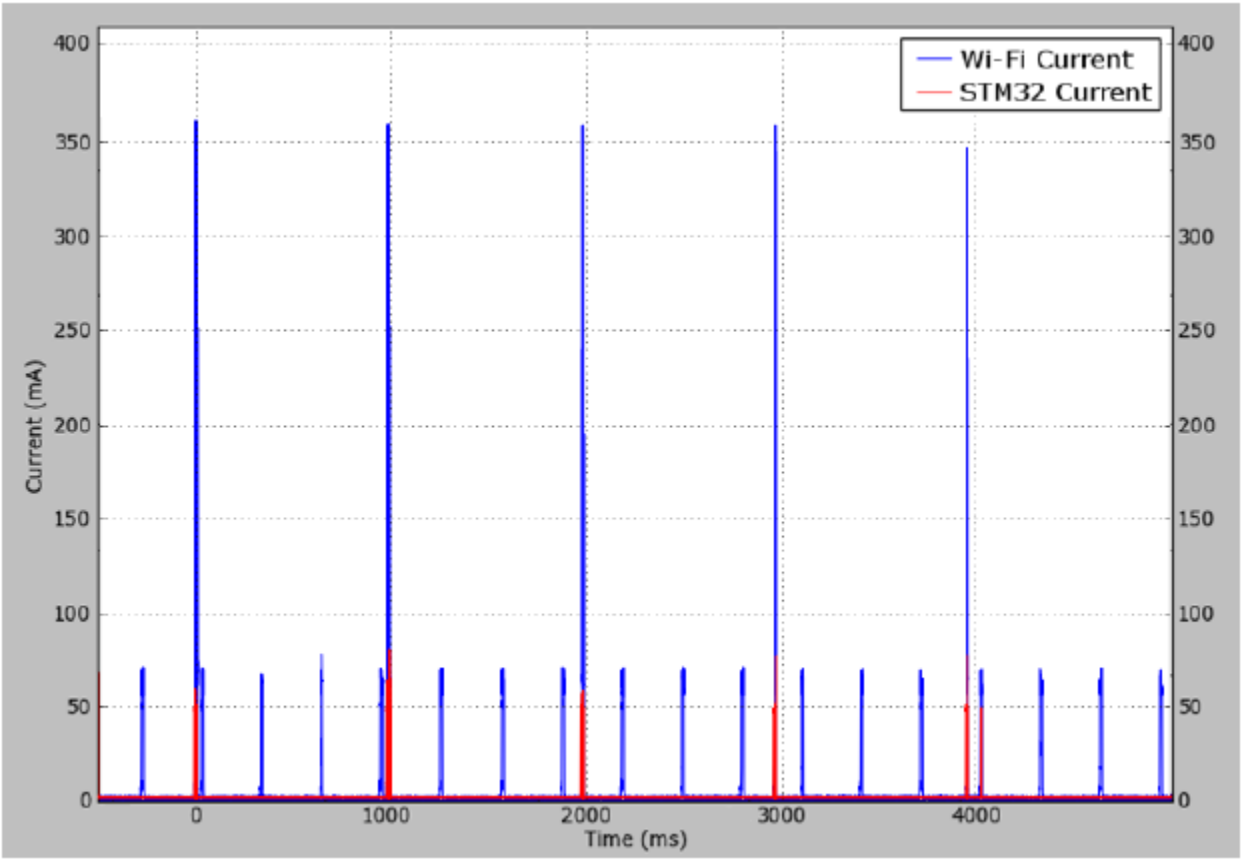

Figure 3-1. Current Consumption with the Ping Powersave application, AP DTIM=3

```
void application_start(void)
{
    wiced_init();  /* Initialise the WICED device */
    wiced_network_up( WICED_STA_INTERFACE, WICED_USE_EXTERNAL_DHCP_SERVER, NULL ); /*Connect*/
    wiced_platform_mcu_enable_powersave();            /* Enable MCU powersave */
    wiced_wifi_enable_powersave_with_throughput(10);  /* Enable Wi-Fi powersave */

    while (1)
    {
        send_ping();                    /* Send an ICMP ping to the gateway */
        wiced_network_suspend();        /* Suspend all network activity, including timers */
        wiced_rtos_delay_milliseconds( WIFI_SLEEP_TIME );  /* Sleep for a while */
        wiced_network_resume();         /* Resume network activity and restart timers */
    }
}
```

Figure 3-2. Representative Ping Powersave Application Source

3.2 DTIM Wake-up

Figure 3-3 provides a close-up showing the Wi-Fi chip (blue trace) waking at an interval of 300ms to receive the AP DTIM beacon. The calls to network suspend/resume have been removed for this plot, and a 100ms IGMP timer that causes the MCU to wake (red trace) is now present.

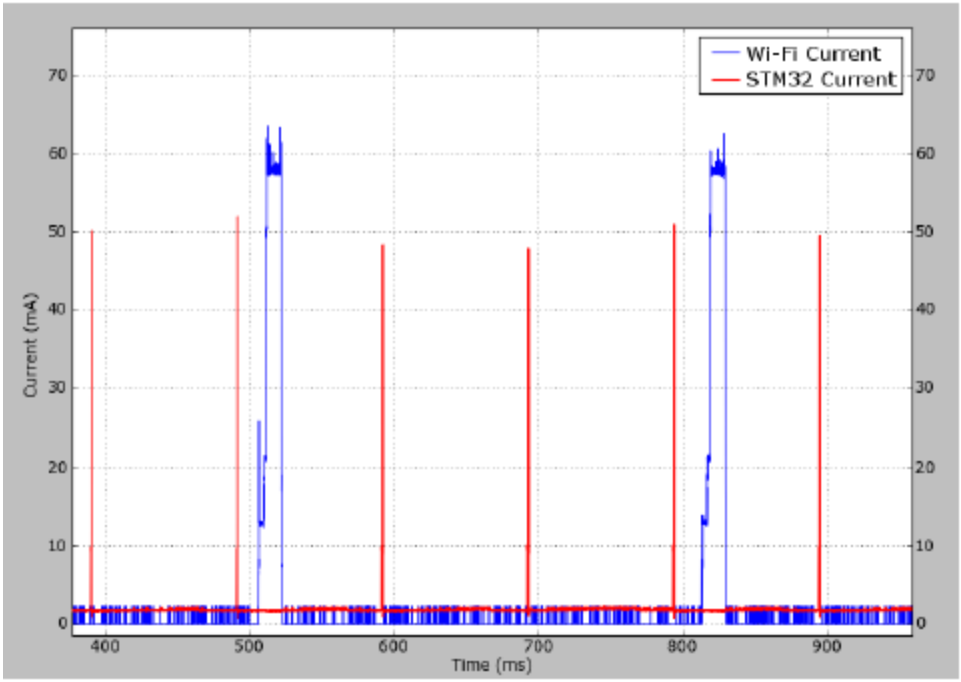

Figure 3-3. Current Consumption for DTIM Beacon Reception

3.3 Ping Transaction

Figure 3-4 shows the detailed current consumption of a Ping transaction. The MCU wakes to send a ping packet. Once awake, it attempts to wake the Wi-Fi chip by polling the SDIO bus. With both the MCU and Wi-Fi chip awake, the transaction begins.

There are four packets transmitted by the Wi-Fi chip, the first is a Null-function Data frame which indicates to the AP that the Wi-Fi station is awake. The second is the ICMP Ping packet. The third is an 802.11 acknowledgement packet for the Ping reply, and the final packet is another Null-Function Data frame to indicate to the AP that the station is about to return to sleep.

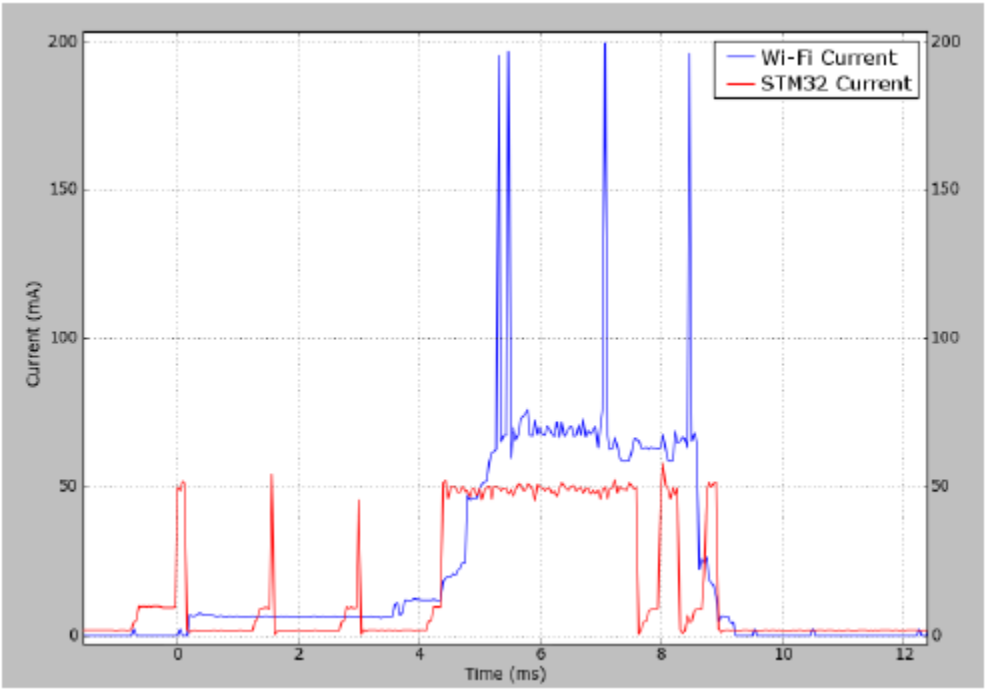

Figure 3-4. Current Consumption for a Ping Transaction

4 Low Power Measurement Techniques

4.1 Introduction

Measuring current consumption over a large dynamic range is difficult without the use of expensive instrumentation. Current consumed during the transmission of a Wi-Fi packet may be 300mA or more, and current consumed during sleep mode may be 10's of microamperes or less. The dynamic range of the measurement covers nearly six orders of magnitude with microsecond switching times!

Most WICED evaluation boards include a current measurement circuit that is accurate for high range (>5mA) measurements. Much can be learned about the operation of an application by observing the current profile, and the circuit on the evaluation board provides an effective means to do this.

Measuring low-range sleep current is more challenging since the current measurement circuit requires a larger resistor in series with the power supply to the device under test. While the current draw is low, there is no issue, but when the board draws a large current (to transmit a WLAN packet, for instance), a large voltage drop occurs across the series resistor and the power supply browns out or fails entirely.

To complicate the measurement further, bulk capacitance provides short-term power to the board during high current loads. In very low-power sleep modes, it may take seconds or even minutes for the energy in the bulk capacitance to dissipate. Energy supplied from the bulk capacitance is typically not measured by the low range current measurement circuit, since bulk capacitance is necessarily placed very close to the Wi-Fi subsystem. The energy provided by the bulk capacitance does not pass through the current measurement circuit.

4.2 Measuring Low-Range Sleep Current

To measure low-range sleep current using a WICED BCM943362WCD4_EVB evaluation board, the following items are needed:

- A custom low power application
- Minor hardware modifications to the WICED evaluation board and WICED module
- Configuration of the BCM943362WCD4 platform
- A multimeter capable of measuring current in the milliamp and microamp range
- The correct measurement technique

4.2.1 Custom Low-Power Application

The goal of the custom application is to make the period between large current spikes as long as possible. This provides time for the energy stored in any bulk capacitance to discharge so that the current draw from the DUT can reach a low power steady state.

The WICED SDK ping powersave snippet application is a good example to start with, the application should be modified as follows to minimise the MCU and Wi-Fi power consumption.

- Ensure the MCU and Wi-Fi power save API calls have not been commented out or removed, and check that MCU powersave has not been disabled in the `wiced_defaults.h` header file.
- Before the main `while(1)` loop, add a call to the listen interval API function to force the Wi-Fi chip to stay in sleep mode for periods of at least 25 seconds (assumes AP beacons are a typical 100ms apart) :

  ```
  wiced_wifi_set_listen_interval(250, WICED_LISTEN_INTERVAL_TIME_UNIT_BEACON);
  ```

- Near the top of `ping_powersave.c`, change the following directive to set the ping interval to 25 seconds: `#define WIFI_SLEEP_TIME` to `(25000 * MILLISECONDS)`

The typical client association timeout limit for most APs is 60 seconds, so it is important to check-in with the AP well within this timeframe to avoid being disassociated.

- Open the file <WICED-SDK>/include/platforms/BCM943362WCD4/platform.h and scroll to the bottom. Ensure the following directive is correctly set to use the STM32 MCO oscillator: #define-WICED_WLAN_POWERSAVE_CLOCK_SOURCE--WICED_WLAN_POWERSAVE_CLOCK_IS_MCO. In powersave mode, the Wi-Fi chip requires a 32kHz clock input to stay in sync with beacons from the AP. The MCO oscillator is used to provide the necessary 32kHz clock when the STM32 is in low power STOP MODE.

  An STM32 timer may alternately be used to provide a 32kHz signal using a pulse-width modulated (PWM) output. However, the STM32 can NOT use MCU powersave mode if a timer is used, since the timer powers down and the 32kHz clock stops.

4.2.2 WICED Module Modifications

The following modification is ONLY required for P100 and P200 revisions of the BCM943362WCD4 module :

Locate the STM32F205 microprocessor and short Pin 41 (PA8) to Pin 44 (PA11). A very limited number of STM32 pins are connected to an internal 32kHz low power MCO oscillator that remains operational in sleep mode. Pin 41 may be configured to connect to the internal oscillator, Pin 44 may not. Pin 44 is however connected to the BCM43362 external sleep clock input pin. Bridging Pin 41 and Pin 44 connects the BCM43362 sleep clock input to the STM32 32kHz low power clock output.

Before proceeding, download and test the application works correctly by looking at the application current profile using an oscilloscope connected to the current measurement circuit on the evaluation board. After the WICED module connects to the AP, the current should be almost always near zero, with a spike once every 25 seconds when the ICMP ping is sent.

If the application fails to stay connected and continually transmits 802.11 probes, the WLAN may not be able to correctly keep in sync with beacons due to a poor timing reference. In this situation, it is possible the 32kHz sleep clock is not connected to the BCM43362 or is not working properly.

Debug this issue before proceeding if necessary.

4.2.3 WICED Evaluation Board Modifications

Locate and desolder the 0805-sized SMT resistor R14. R14 is a 0.2 Ohm resistor in series with the WICED module power supply.

Connect one lead of the multimeter to the VDD_3V3 pad of R14, and connect the other lead of the multimeter to the VDD_3V3_WIFI pad of R14. The multimeter is now connected in series with the WICED module power supply.

4.2.4 Measurement Technique

Low power current measurements are made by following these steps:

1. On the multimeter, select the low-range current mode (microamps to milliamps).
2. Connect the multimeter leads across the location of R14, one lead on each pad of R14.
3. Apply a short across the location of R14. One technique to achieve this is to solder a 2-pin header across the location of R14, and place a jumper on the header.
4. While watching the UART output of the evaluation board, press the reset button and wait for a print indicating a successful ping reply. An ICMP ping transaction occurs approximately every 25 seconds.
5. Wait 1-2 seconds after a print indicating a successful ping reply, then remove the short (jumper) across the location of R14.
6. The multimeter current settles to the minimum sleep current of approximately 500 microamps. Note that an intermittent current blip may also be observed every 4 or 5 seconds, the current blip corresponds to an MCU timer firing which may briefly wake the MCU.
7. After making the measurement (within ~20 seconds), replace the short across the location of R14. If the WICED module attempts to transmit or transition into a high power mode while the multimeter is in the low-range current mode, the module will almost certainly brown out. Additionally, there is some chance of blowing a fuse in the multimeter.

References

The references in this section may be used with this document.

Note: Cypress provides customer access to technical documentation and software through the WICED website (community.cypress.com/). Additional restricted material may be provided through the Customer Support Portal (CSP) and Downloads.

For Cypress documents, replace the 'xx' in the document number with the largest number available to ensure you have the most current version of this document.

Document (or Item) Name	Number	Source
[1] 802.11 Wireless Networks "The Definitive Guide"	2nd Edition, April 2005 Matthew S. Gast	O'REILLY
[2] WICED Quick Start Guide	WICED-QSG2xx-R	WICED SDK
[3] STM32F20x Reference Manual	CD00225773	ST website
[4] AT91SAM4S Datasheet		Atmel website

Document Revision History

Document Title: WICED™ Development Powersave System

Document Number:002-19004

Revision	ECN	Issue Date	Description of Change
**		03/02/2012	**WICED-AN100-D1 :** Initial release
		12/14/2012	**WICED-AN101-R :** Updated for WICED-SDK-2.2.0
		02/11/2013	**WICED-AN102-R :** Updated for WICED-SDK-2.2.1
		04/19/2013	**WICED-AN103-R :** Updated for WICED-SDK-2.3.0
		07/07/2013	**WICED-AN104-R :** Updated for WICED-SDK-2.4.0. Added Section 5, Low Power Measurement Techniques.
*A		03/27/2017	Converted to Cypress template format.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709