Answer Key

Chapter 02

Exercise 01

1. The table at the top of platform.h says that WICED_LED1 connects to WICED_GPIO_12, Arduino header D5, and WICED_PWM_3. Explain how this mapping was determined. You will need to refer to platform.h, platform.c and the schematic for the base board.

   From platform.h line 383, WICED_LED1 is mapped to WICED_GPIO_12

   From platform.c line 53, WICED_GPIO_12 is assigned to PIN_GPIO_16

   From platform.c line 120, PIN_GPIO_16 is assigned to WICED_PWM_3

   From the schematic page 9, GPIO_16 connects to D5

Exercise 02

1. Why can't you read the value of the LED using the *wiced_gpio_input_get* function instead of using a variable to remember the state?

   The wiced_gpio_input_get function is only valid if the pin is configured as an input pin.

2. In what file and on what line does the WICED_LED1 get assigned to the correct pin for this kit?

   platform.h, line 383.

3. In what file and on what line is the pin connected to the LED set as an output?

   platform.c, line 323.

Exercise 08

1. What I2C addresses are found?

   0x42 Analog Co-processor

   0x3C OLED Display

Chapter 03

Exercise 02

1. Do you need *wiced_rtos_delay_millisecon*ds in the LED thread? Why or why not?

   No, because the semaphore causes the thread to suspend until it is set by the button ISR.

2. What happens if you use a value of 100 for the semaphore timeout? Why?

   The LED will blink every 100ms because the semaphore will timeout even when the button is not pressed.

Exercise 03

1. What happens if you forget to unlock the mutex in one of the threads? Why?

   The thread that has the lock will keep running but the other thread will stay suspended because it can never get access to the mutex. Therefore, only one of the buttons will cause the LED to blink (the one that has the lock).

Exercise 05

1. What happens if you don't remove the while(1) loop from the function that blinks the LED? Why?

   The LED will appear to stay on all the time (in fact, it is blinking on/off rapidly) so it appears dim. The reason is that as soon as the timer executes the LED blinking function once, it never exits so it continually blinks the LED with no delay.

2. What happens if the application_start doesn't have a while(1) loop? Why?

   The chip will continuously reset because there are no active threads once application_start exits. Remember that the timer is NOT a thread on its own.

3. Does the while(1) loop in application_start need a delay? Why or why not?

   No, because application_start is the only thread in the project.

Chapter 05

Exercise 02

1. There are three changes required in the wifi_config_dct.h file:

   CLIENT_AP_SSID changes to ***"WW101OPEN"***

   CLIENT_AP_PASSPHRASE changes to ***""***

   CLIENT_AP_SECURITY changes to ***WICED_SECURITY_OPEN***

   Hint: you can find all of the security types available by right clicking on WICED_SECURITY_OPEN (or any other security name) from the DCT file and selecting "Open Declaration".

Chapter 07b

Exercise 03

1. Which server port is used for HTTP (non-secure)?

   Port 80.

2. What function is called each time an HTTP event occurs? Where is that specified?

   The callback function is called event_handler. It is specified as a parameter to http_client_init.

3. What header(s) is/are sent with each request?

   In this case, only a single header is sent:

   Host: httpbin.org

4. What is the purpose of the semaphore "httpWait".

   The semaphore causes the firmware to wait until the first request has completed before sending the second request. Since we are re-using the request structure for the second request this is necessary. Even if we had separate request structures, the semaphore is still useful because it guarantees that the requests won't interfere with each other. If you didn't do this, you could have the streams from multiple requests sending data over the same socket at the same time. Another alternative is to use a separate HTTP client and request structure for each request which means you would have a separate socket for each one.

5. How many response payload packets do we get from the request to /http?

   There are 3 payload packets from the request to /http.

6. Where is the http_request_deinit called? Why?

   The http_request_deinit is called inside the http callback function (event_handler) but only when the response->remaining_length is equal to zero. This must be done because for a large response (like from /html) the payload may be sent in several packets. Therefore, you must make sure that nothing else is coming before you de-init the request.

7. What is the variable "connected" used for? Why is it needed?

   The variable "connected" is used to determine if the connection to the server is still active. It is needed because the server can disconnect at any time. Therefore, before sending another request, we need to see if the connection is still there. If not, we need to restart everything.

8. Uncomment the section of code to wait for the server to disconnect between requests. How long does the server wait before closing the connection?

   The server disconnects after about 60 seconds of inactivity.

Exercise 04

1. Which server port is used for HTTPS (secure)?

   Port 443.

2. What function call and parameter specifies that the connection should use TLS?

   The 4th parameter to http_client_connect is "HTTP_USE_TLS" instead of "HTTP_NO_SECURITY."

3. Where is the certificate stored inside the device?

   The certificate is stored inside the DCT.

4. How is the certificate read into the firmware?

   The certificate is read into the firmware by using the function wiced_dct_read_lock.

Exercise 05

1. What headers sent with the POST request?

   There are 3 headers:

   Host: httpbin.org

   Content-Type: application/json

Content-Length: 15

(back-slashes don't count in the content length)

2. What is the JSON content that is posted?

   The JSON is a key-value pair of {"WICED":"yes"}

Exercise 06

1. Where is the certificate stored inside the device?

   The certificate is stored in Flash after the DCT information.

2. How is the certificate read into the firmware?

   The certificate is read into the firmware by using the function resource_get_readonly_buffer.

Chapter 07c

Exercise 05

1. How do the MQTT library functions (e.g. wiced_aws_publish) get into your project?

   The line "`$(NAME)_COMPONENTS := protocols/AWS`" in the make file and the #include for "wiced_aws.h" and "aws_common.h" in the C file cause the AWS library functions to be included in the project.

2. What function is called when the button is pressed?

   `button_press_callback`

3. How does the button callback unlock the main thread?

   It sets a semaphore using "`wiced_rtos_set_semaphore( &wake_semaphore );`"

4. Are all messages sent to the AWS IOT MQTT Message Broker required to be in JSON format?

   No, but messages that affect the shadow have to be JSON.

5. What are the 6 WICED AWS events? What file are they defined in?

   WICED_AWS_EVENT_ CONNECTED
   WICED_AWS_EVENT_ DISCONNECTED
   WICED_AWS_EVENT_ PUBLISHED
   WICED_AWS_EVENT_ SUBSCRIBED
   WICED_AWS_EVENT_ UNSUBSCRIBED
   WICED_AWS_EVENT_PAYLOAD_RECEIVED

   They are defined in wiced_aws.h.

6. Do you have to name the client certificate client.cer? How would you change the name?

   No, the name can be changed in the make file ($(NAME)_RESOURCES) and in the C source code when the certificate is read in.

7. What is the callback function for an AWS event? How is it registered?

   The callback function is my_publisher_aws_callback. It is registered by the line:

   ret = wiced_aws_init(&my_publisher_aws_config , my_publisher_aws_callback);

8. What steps are required to get an AWS connection established?
   - Connect to the network (wiced_network_up)
   - Read in the 3 credentials using get_aws_credentials_from_resources which calls resource_get_readonly_buffer
   - Initialze the interface using wiced_aws_init
   - Create the endpoint to the Broker using wiced_aws_create_endpoint
   - Open the connection using wiced_aws_connect

9. What function is called to send data to the server?
   wiced_aws_publish

10. What is the name of the flag that prevents the firmware from sending multiple button presses before the publish is finished?

    do_publish