Answer Key

# Chapter 2

### Exercise 2.1

1. Which chip GPIO is used for the I2C SCL and SDA pins?

SCL: WICED\_P26

SDA: WICED\_P28

1. Are the button pins pulled up or down? Where is that specified?

Buttons are pulled up. This can be found in the file wiced\_platform\_pin\_config.c in the platform folder on lines 85 and 96.

### Exercise 2.2

1. What is the name of the first user application function that is executed? What does it do?

APPLICATION\_START. It just initializes the Bluetooth stack and registers the callback.

1. What is the purpose of the function bt\_cback? When does the BTM\_ENABLED\_EVT case occur?

It is the Bluetooth stack management callback function. It is called whenever there is a management event from the stack.

1. What controls the rate of the LED blinking?

The first parameter to the RTOS delay function wiced\_rtos\_delay\_milliseconds specifies the delay which controls the rate of the LED blinking.

# Chapter 3

### Exercise 3.1

1. Do you need *wiced\_rtos\_delay\_millisecon*ds() in the LED thread? Why or why not?

No, because the wiced\_rtos\_get\_semaphore will cause the thread to suspend each time through the infinite loop while it waits for another button press.

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

The LED will blink every 100ms because the semaphore times out. This will happen even without pressing the button.

### Exercise 3.2

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 3.4

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 almost no delay.

# Chapter 4A

### Exercise 4A.1

1. How many bytes is the advertisement packet?

The advertisement packet is19 bytes. They are:

Flags (3)

Length (1)

Type (1)

Data (1)

Local Name (9)

Length (1)

Type (1)

Data (7)

Appearance (4)

Length (1)

Type (1)

Data (2)

Manufacturer Specific Data (3)

Length (1)

Type (1)

Data (1)

### Exercise 4A.2

1. What function is called when there is a Stack event? Where is it registered?

The function is *ex02\_ble\_con\_management\_callback*. It is registered using *wiced\_bt\_stack\_init* in *application\_start*.

1. What function is called when there is a GATT database event? Where is it registered?

The function is *ex02\_ble\_con\_event\_handler*. It is registered using *wiced\_bt\_gatt\_register* in *ex02\_ble\_con\_app\_init*.

1. Which GATT events are implemented? What other GATT events exist? (Hint: right click and select Open Declaration on one of the implemented events)

Implemented:

GATT\_CONNECTION\_STATUS\_EVT

GATT\_ATTRIBUTE\_REQUEST\_EVT

Others:

GATT\_OPERATION\_CPLT\_EVT

GATT\_DISCOVERY\_RESULT\_EVT

GATT\_DISCOVERY\_CPLT\_EVT

GATT\_CONGESTION\_EVT

1. In the GATT "GATT\_ATTRIBUTE\_REQUEST\_EVT", what request types are implemented? What other request types exist?

Implemented:

GATTS\_REQ\_TYPE\_READ

GATTS\_REQ\_TYPE\_WRITE

Others:

GATTS\_REQ\_TYPE\_PREP\_WRITE

GATTS\_REQ\_TYPE\_WRITE\_EXEC

GATTS\_REQ\_TYPE\_MTU

GATTS\_REQ\_TYPE\_CONF

# Chapter 4B

### Exercise 4B.2

1. How long does the device stay in high duty cycle advertising mode? How long does it stay in low duty cycle advertising mode? Where are these values set?

High: 30 seconds

Low: 60 seconds

These are specified in the wiced\_bt\_cfg.c file in *wiced\_bt\_cfg\_settings.ble\_advert\_cfg.high\_duty\_duration* and *wiced\_bt\_cfg\_settings.ble\_advert\_cfg.low\_duty\_duration*

### Exercise 4B.3

1. What items are stored in NVRAM?

*Hostinfo* (Remote BDADDR and Button CCCD state)

*Local Keys*

*Paired Device Keys*

1. Which event stores each piece of information?

*Hostinfo* is stored during BTM\_PAIRING\_COMPLETE\_EVT and in ex03\_ble\_bond\_set\_value if the Button CCCD value was written

*Local Keys* are stored during BTM\_LOCAL\_IDENTITY\_KEYS\_UPDATE\_EVT

*Paired Keys* are stored during BTM\_PAIRED\_DEVICE\_LINK\_KEYS\_UPDATE\_EVT

All three are cleared out (i.e. reset) in the button\_cback function to allow re-pairing.

1. Which event retrieves each piece of information?

*Hostinfo* is retrieved by BTM\_ENCRYPTION\_STATUS\_EVT (if the device was previously bonded

*Local Keys* are retrieved by BTM\_LOCAL\_IDENTITY\_KEYS\_REQUEST\_EVT

*Paired Keys* are retrieved by ex03\_ble\_bond\_app\_init (at startup) and by BTM\_PAIRED\_DEVICE\_LINK\_KEYS\_REQUEST\_EVT

### Exercise 4B.4

1. Other than BTM\_IO\_CAPABILITIES\_NONE and BTM\_IO\_CAPABILITIES\_DISPLAY\_ONLY, what other choices are available? What do they mean?

BTM\_IO\_CAPABILITIES\_DISPLAY\_AND\_YES\_NO\_INPUT

Device can display values (e.g. 6-digit numbers) and can accept a Yes/No input from the user.

BTM\_IO\_CAPABILITIES\_KEYBOARD\_ONLY

Device can accept input (e.g. numbers) but cannot display any values.

BTM\_IO\_CAPABILITIES\_BLE\_DISPLAY\_AND\_KEYBOARD\_INPUT

Device can display values (e.g. 6-digit numbers) and can accept input (e.g. numbers).

1. What additional stack callback event occurs compared to the previous exercise? At what point does it get called?

BTM\_PASSKEY\_NOTIFICATION\_EVT

This event is called between BTM\_PAIRING\_IO\_CAPABILITIES\_BLE\_REQUEST\_EVT and BTM\_ENCRYPTION\_STATUS\_EVT.