

# Answer Key

## Chapter 1

### Exercise 1.3

1. What is the organization of the spec?  
Volumes, Parts, Sections
2. On what page does the Attribute protocol specification start?  
2169

## Chapter 2

### Exercise 2.1

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

SCL: WICED\_P26

SDA: WICED\_P28

2. 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.
2. 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.

3. 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.

### Exercise 2.6

1. How many bytes does the NVRAM read function get before you press the button the first time?

0

2. What is the return status value before you press the button the first time?

The return value is 40 (0x28).

3. What does the return value mean?

The return value of 40 (0x28) means ERROR. This is defined in the file `wiced_result.h`.

## Chapter 3

### Exercise 3.1

1. Do you need `wiced_rtos_delay_milliseconds()` 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.

2. 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.3

1. How many bytes is the advertisement packet?

The advertisement packet is 19 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.4

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

The function is *ex03\_ble\_con\_management\_callback*. It is registered using *wiced\_bt\_stack\_init* in *application\_start*.

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

The function is *ex03\_ble\_con\_event\_handler*. It is registered using *wiced\_bt\_gatt\_register* in *ex03\_ble\_con\_app\_init*.

3. 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

4. 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.3

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.4

1. What items are stored in NVRAM?

*Hostinfo* (Remote BDADDR and Button CCCD state)

*Local Keys* (Privacy Information)

*Paired Device Keys* (Encryption Information)

2. 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_PAIRING_DEVICE_LINK_KEYS_UPDATE_EVT`

All three are cleared out (i.e. reset) in the `button_cback` function to allow re-pairing.

3. 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_PAIRING_DEVICE_LINK_KEYS_REQUEST_EVT`

4. In what event is the privacy info read from NVRAM?

`BTM_LOCAL_IDENTITY_KEYS_REQUEST_EVT`

5. Which event is called if privacy information is not retrieved after new keys have been generated by the stack?

`BTM_LOCAL_IDENTITY_KEYS_UPDATE_EVT`

### Exercise 4B.5

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).

2. 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.

## Chapter 4C

### Exercise 4C.7

1. Which variable is used to control sleep permissions? What are its states?

The variable is `sleep_type`. It can be:

`WICED_SLEEP_NOT_ALLOWED`  
`WICED_SLEEP_ALLOWED_WITHOUT_SHUTDOWN`  
`WICED_SLEEP_ALLOWED_WITH_SHUTDOWN`

2. What is printed to the UART when:
  - a. A sleep request is denied: x
  - b. A sleep request is allowed without shutdown: \$
  - c. A sleep request is allowed with shutdown: .
  - d. A fast boot occurs: f
3. When is sleep not allowed? When is sleep allowed but without shutdown (PDS)? When is sleep allowed with shutdown (SDS)?

Sleep is not allowed from powerup until advertising is started.

PDS is allowed once the GATT connection is made  
(`GATT_CONNECTION_STATUS_EVT` with status of connected) until pairing is complete  
(`BTM_ENCRYPTION_STATUS_EVT` with result of `WICED_SUCCESS`).

SDS is allowed during advertising and during a connection once pairing is complete.

4. When does SDS occur (Hint: you can determine that SDS occurred when you see a fast boot)?

SDS occurs during low duty cycle advertising (because the timeout is set to 0) and during a connection once pairing is complete.

5. What is done differently for a cold boot vs. a fast boot? Why?

For a cold boot, the GPIO for button 1 is configured. That isn't needed for a fast boot since the GPIO configuration is retained. Also, more debug information is printed during a cold boot to get initial startup information without flooding the screen with values during fast boot.

For a fast boot, we check the `connection_id`. If the device is connected, then we read `hostinfo` (pairing information) from the NVRAM, restore the previous CapSense button



value and CCCD value in the GATT database. This allows the firmware to determine if a notification must be sent the next time a CapSense scan is done (i.e. if the CapSense value has changed and notifications are enabled).

6. What is AON RAM? What values is it used for and why?

AON is Always ON RAM. It is RAM that is retained during SDS. It is used for connection\_id, prevVal, and advert\_mode.

connection\_id allows us to tell if we were connected when we went to sleep so that the bonding information can be restored on a fast boot.

prevVal is used to restore the CapSense button value. This is done so that the firmware can tell if the button value changed while the device was asleep.

Advert\_mode is saved so that on a fast boot advertising can start up where it left off. If this wasn't done, the device would start high duty cycle advertisements again as soon as it did a fast boot.

7. Where are the connection parameters (interval and timeout) updated? Why?

In the GATT connection callback. This reduces power consumption and allows power down without losing the connection.