



# Answer Key

## Chapter 1

### Exercise 1.2

1. Where is the documentation for the PWM API located?

It is in the WICED API Reference. The path in that document is:

- Components
- Hardware Drivers
- PWM

## Chapter 2

### Exercise 2.1

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

The first user function is `application_start`.

It just initializes the Bluetooth stack and registers the callback.

2. What is the purpose of the function `app_bt_management_callback`? 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.

The `BTM_ENABLED_EVT` case occurs once the stack has completed initialization.

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 2.9

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.

### Exercise 3.2

1. Before you added the mutex, how did the LED behave when you pressed the button?

The LED flashes in an irregular pattern.

2. What changed when you added the mutex?

The LED flashes slowly (2Hz) when the button is not pressed and then flashes quickly (5Hz) when the button is pressed.

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

The other thread will never execute. That is:

If you don't unlock the mutex in the fast LED thread, the slow LED thread will not execute. The LED will blink fast when the button is pressed but will not blink when the button is not pressed.

If you don't unlock the mutex in the slow LED thread, the fast LED thread will never execute so the LED will blink slowly no matter what happens with the button.

## Chapter 4A

### Exercise 4A.3

1. How many bytes is the advertisement packet?

The advertisement packet is 17 bytes (assuming your initials are 3 characters). They are:

Flags (3)

Length (1)

Type (1)

Data (1)

Local Name (9)

Length (1)

Type (1)

Data (7)

Manufacturer Specific Data (5)

Length (1)

Type (1)

Data (3)

### Exercise 4A.4

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

The function is `app_bt_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 `app_gatt_callback`. It is registered using `wiced_bt_gatt_register` in `app_bt_management_callback` in the `BTM_ENABLED_EVT` event.

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\_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 (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 `app_gatt_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_PAURED_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 during `BTM_LOCAL_IDENTITY_KEYS_REQUEST_EVT`

Paired Keys are retrieved during `BTM_ENABLED_EVT` and  
`BTM_PAURED_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.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).

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

`BTM_PASKEY_NOTIFICATION_EVT`

This event is called between `BTM_PAIRING_IO_CAPABILITIES_BLE_REQUEST_EVT` and `BTM_ENCRYPTION_STATUS_EVT`.

## Chapter 4C

### Exercise 4C.1

1. Which lines in the code are used to configure and initialize sleep?

In *low\_power\_208xx.c*:

```
/* configure to sleep if sensor is idle */
low_power_sleep_config.sleep_mode           =
WICED_SLEEP_MODE_NO_TRANSPORT;
low_power_sleep_config.device_wake_mode     = WICED_SLEEP_WAKE_ACTIVE_LOW;
low_power_sleep_config.device_wake_source   = WICED_SLEEP_WAKE_SOURCE_GPIO;
low_power_sleep_config.device_wake_gpio_num =
WICED_GET_PIN_FOR_BUTTON(WICED_PLATFORM_BUTTON_1);
low_power_sleep_config.host_wake_mode       = WICED_SLEEP_WAKE_ACTIVE_HIGH;
low_power_sleep_config.sleep_permit_handler = low_power_sleep_handler;
low_power_sleep_config.post_sleep_cback_handler = low_power_post_sleep_cb;

if(WICED_BT_SUCCESS != wiced_sleep_configure(&low_power_sleep_config))
{
    WICED_BT_TRACE("Sleep Configure failed\r\n");
}
```

2. What is used as a wakeup source?

WICED\_PLATFORM\_BUTTON\_1 is used as a wakeup source.

3. What is the name of the sleep permit handler function?

low\_power\_sleep\_handler

4. When are the connection interval min, max, latency, and timeout values updated and what values are used?

The values are updated in the GATT connect callback function when a connection is established. The code is in *low\_power\_208xx\_ble.c*:

```
/* Update connection parameters to 100 ms */
if(TRUE != wiced_bt_l2cap_update_ble_conn_params(p_conn_status->bd_addr,
CONNECTION_INTERVAL, CONNECTION_INTERVAL,
SLAVE_LATENCY, SUPERVISION_TIMEOUT))
{
    WICED_BT_TRACE("Connection parameters update failed\r\n");
}
```

The values are set to:

Min Interval: 100ms	(80 * 1.25ms)
Max Interval: 100ms	(80 * 1.25ms)
Latency: 0	
Timeout: 3000ms	(300 * 10ms)



## Chapter 8

### Exercise 8.4

1. What items are stored in EEPROM?

- Bluetooth Device Address of the bonded device (`remote_bda`)
- CCCD value (`cccd`)
- Local Identity Security Keys (`identity_keys`)
- Security keys for the bonded device (`link_keys`)

2. Which event stores each piece of information?

`remote_bda` is stored during `BTM_PAIRING_COMPLETE_EVT`

`cccd` is stored in `app_gatt_set_value` when the CCCD value is written.

`identity_keys` are stored during `BTM_LOCAL_IDENTITY_KEYS_UPDATE_EVT`

`link_keys` are stored during `BTM_PAIRING_DEVICE_LINK_KEYS_UPDATE_EVT`

Local Keys are stored during `BTM_LOCAL_IDENTITY_KEYS_UPDATE_EVT`

Paired Keys are stored during `BTM_PAIRING_DEVICE_LINK_KEYS_UPDATE_EVT`

All four are cleared out (i.e. reset) in the UART callback function when 'e' is pressed to allow re-pairing.

3. Which event retrieves each piece of information?

`cccd` is retrieved by `BTM_ENCRYPTION_STATUS_EVT` (if the device was previously bonded)

`identity_keys` are retrieved by `BTM_LOCAL_IDENTITY_KEYS_REQUEST_EVT`

`link_keys` are retrieved in `BTM_ENABLED_EVT` and by `BTM_PAIRING_DEVICE_LINK_KEYS_REQUEST_EVT`

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

`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`