ModusToolbox™

WICED HID Device Library

Associated Part Family: CYW20735, CYW20719, CYW20819, and CYW20721
Document Number. 002-16556 Rev. *C

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

Contents

1 Introduction

This document addresses the software design for the WICED HIDD library used in HIDD applications. It provides information on how applications can use the library to define, send, and receive HID reports, and how applications retrieve data from internal HW components and external peripherals. It explains how HID reports are sent in a timely manner and how applications utilize the power management unit (PMU) component to conserve power. It is assumed that the reader is familiar with the Bluetooth Core [1] and the HID over GATT profile from Bluetooth SIG [2].

2 IoT Resources

Cypress provides a wealth of data at http://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 (http://community.cypress.com/).

3 Design and Architecture

ModusToolbox^{TM} provides sample applications for HID devices that include keyboard, mouse, and remote control. While applications themselves mostly concentrate on servicing the device specific hardware, they use common WICED LE HIDD library APIs to provide Bluetooth functionality. Access to the device HW is done using the WICED Hardware Abstraction Layer (HAL).

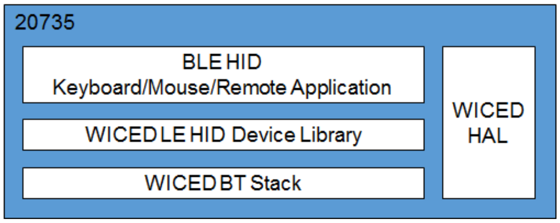
20735
BLE HID
Keyboard/Mouse/Remote Application
WICED LE HID Device Library
WICED BT Stack
WICED
HAL

Figure 1. HID Applications Software Block Diagram

4 WICED BLE HIDD Library

The WICED LE HIDD library provides an implementation of the HID service and the HID over GATT profile. It also provides a set of functions commonly used by HID device applications. This includes battery monitor service, event queues, sleep/wake registration, and BLE link information.

4.1 Low Power Modes

HID devices support two sleep modes: normal sleep and shutdown sleep (SDS). The device enters the normal sleep mode automatically when the controller has sufficient time to sleep before the next timer or the next BT activity and when SDS mode is not activated by the application or cannot be entered.

While in SDS mode, some HW components are still powered on, and the HID device can react to a timer or external GPIO interrupts. The device can also periodically wake up to perform the following BT activities:

- If advertising is turned on, the device wakes up at each advertising interval, sends an advertising packet, and listens for the incoming SCAN_REQ or CONNECT_REQ packets. A SCAN_REQ packet is answered without being awakened from SDS. The device wakes up if it receives a CONNECT_REQ.
- If scanning is turned on, the device wakes up at each scan interval and listens for advertising packets. The device wakes up if it receives an advertising packet that it needs to process.
- In the connection state, the device wakes up and answers polls from the master. Receiving a valid data packet causes the device to wake up.

To support sleep modes, the application must initialize the low power management module by calling `wiced_hal_lpm_register_for_low_power_queries`. The function accepts a low power management callback as a parameter. The stack calls the registered application callback function to request the initiation of the SDS mode. If the callback returns an OK to enter SDS, it will enter SDS. If the application rejects entering the SDS mode, the stack will query for normal sleep.

In the SDS mode, only always on memory (AON) is retained. When the application wakes up, all the variables in the RAM are reset to zero. The application should rely on the WICED HIDD library to maintain the state of the general HID variables. For example, the library saves the connection state, connection handle, and encryption state in the AON. The application should always call the library functions to discover whether the link is connected or encrypted.

4.2 BLE HID Link

The WICED HIDD library manages the discoverable/connectable state of the BLE device as well as the connection state and connection handle. It also manages the queue of the HID events to be sent to the connected HID host. The application shall initialize the BLE HID link functionality by calling the `wiced_ble_hidd_link_init` function.

4.2.1 Connection Management

The WICED BLE HIDD library provides the functions to connect to a previously bonded host (`wiced_ble_hidd_link_connect`), disconnect an existing connection (`wiced_ble_hidd_link_disconnect`), and verify whether a connection is active (`wiced_ble_hidd_link_is_connected`).

The application can call `wiced_blehidd_is_link_encrypted` to determine whether encryption has been set up for the link.

4.2.2 Sending Reports

To send an input or a feature report, the application can call the `wiced_ble_hidd_link_send_report` function. This function verifies that the host is connected, that the connection is encrypted, and that the client characteristic configuration descriptor for this report ID is allowed.

4.2.3 Processing Writes

When the application receives a GATT Write request that corresponds to one of the characteristic or descriptor values in the GATT database, it should call the `wiced_blehidd_write_handler` function to verify that the write is enabled, and execute a handler for its specific attributes.

4.2.4 Virtual Cable Unplug

When the application is required to perform a virtual cable unplug function, it should call the wiced_ble_hidd_link_virtual_cable_unplug function. This is normally done when the user pushes the Connect button. If the host is connected, then the function starts the disconnection process, and the information about the host is removed from the NVRAM, and the device is set as discoverable.

4.2.5 HID Event Queue

The HID event queue is used by the application to temporarily store HW events to be reported to the connected host. The events are de-queued and then scheduled for transmission in a special callback. The application registers this callback with the WICED HIDD library to be called just before the BLE connection event using the wiced_ble_hidd_link_register_poll_callback function. The poll callback functionality can be enabled or disabled using the wiced_ble_hidd_link_enable_poll_callback function.

The event queue is initialized by calling wiced_hidd_event_queue_init. All events in the HID event queue can be deleted by calling wiced_hidd_event_queue_flush. To discover how many elements are currently stored in the HID event queue, the application can call wiced_hidd_event_queue_get_num_elements. The application can read the first element from the queue (wiced_hidd_event_queue_get_current_element) and delete it from the queue (wiced_hidd_event_queue_remove_current_element). To add a new event to the event queue, the application can call the wiced_hidd_event_queue_add_event_with_overflow. If the event queue is full, the function queues an overflow event.

4.3 BLE HID Host Information

HID device applications may require information about HID hosts to be stored in the AON and NVRAM. The WICED HID library provides a set of functions commonly used by HID applications to manipulate this information. For each host, the Bluetooth device address, address type, link keys, bonded flag, and set of notification flags are enabled per each client characteristic configuration descriptor in the GATT database.

After waking from the SDS mode, all local application variables are reset to zero. The application should use the WICED HIDD library functionality to store and retrieve the host information.

To initialize the list and retrieve the data previously saved in the NVRAM, the application should call the wiced_ble_hidd_host_info_init function. The number of hosts currently stored in the NVRAM (and being monitored by the library) can be retrieved using the wiced_ble_hidd_host_info_get_number function.

After a successful pairing, the application should call the wiced_ble_hidd_host_info_add_first function to add the information about the newly paired host.

The application can retrieve the information about the host on the top of the host info list using wiced_ble_hidd_host_info_get_bdaddr, wiced_ble_hidd_host_info_is_bonded, and wiced_ble_hidd_host_info_get_link_keys.

The application should define a bit corresponding to each notifiable characteristic in the GATT database, with each characteristic that has a client characteristic configuration descriptor. The combination of these bits constitutes a host's notification flags value. The application can use wiced_ble_hidd_host_info_get_flags to retrieve the current value of flags corresponding to the first host in the HIDD host information list. To modify the flags value for the first host in the HIDD host information list, the application can use the wiced_ble_hidd_host_info_update_flags function.

The host info can be deleted using wiced_ble_hidd_host_info_delete or wiced_ble_hidd_host_info_delete_all functions.

Note: The application may also require NVRAM storage, for instance to store its own local identity keys. See the ModusToolbox sample application 'ble_remote' usage of VS_LOCAL_IDENTITY_KEYS for an example. As shown there, applications may use NVRAM Volatile Section Identifiers (VSIDs) between WICED_NVRAM_VSID_START and WICED_NVRAM_VSID_STOP, defined in *wiced_hal_nvram.h*.

However, the WICED HIDD library reserves NVRAM VSID (WICED_NVRAM_VSID_START + 1) for its use internally with the wiced_ble_hidd_host_info_* library functions.

5 Library Reference

5.1 HIDD APP Init

Performs the necessary initialization for the HID Device.

Prototype

```
void wiced_hidd_app_init (wiced_bt_device_type_t dev_type)
```

Parameters

`dev_type` : `BT_DEVICE_TYPE_BREDR`, or `BT_DEVICE_TYPE_BLE`, or `BT_DEVICE_TYPE_BREDR_BLE`

Returns

None

5.2 BLE HIDD Allow Slave Latency

Allow applications to enable or disable slave latency. Audio and gestures work best when long interval is disabled.

Prototype

```
void wiced_blehidd_allow_slave_latency (wiced_bool_t allow)
```

Parameters

`allow` : `WICED_TRUE` (allow slave latency), or `WICED_FALSE` (disable slave latency)

Returns

None

5.3 BLE HIDD Get ATT MTU

Get ATT protocol maximum transmission unit (MTU) negotiated during connection establishment.

Prototype

```
uint16_t wiced_blehidd_get_att_mtu_size (BD_ADDR bda)
```

Parameters

`bda` : Peer device Bluetooth Device (BD) address

Returns

Negotiated attribute protocol MTU size

5.4 BLE HIDD Get Connection Handle

Gets the connection handle, which is returned by the controller in the BLE connection complete event. WICED HIDD libraries store the connection handle in the AON. The application should use this method rather than storing the handle in a global variable.

Prototype

```
uint16_t wiced_blehidd_get_connection_handle (void)
```

Parameters

None

Returns

Connection handle

5.5 BLE HIDD Get Connection Interval

Returns the connection interval of the BLE connection.

Prototype

```
uint16_t wiced_blehidd_get_connection_interval (void)
```

Parameters

None

Returns

Current connection interval

5.6 BLE HIDD Get Peer Address

Gets the pointer to the address of the currently connected BLE host.

Prototype

```
uint8_t* wiced_blehidd_get_peer_addr (void)
```

Parameters

None

Returns

BD Address

5.7 BLE HIDD Get Peer Address Type

Gets the device address type of the currently connected BLE host.

Prototype

```
uint8_t wiced_blehidd_get_peer_addr_type (void)
```

Parameters

None

Returns

None

5.8 BLE HIDD Get Slave Latency

Gets connection slave latency.

Prototype

```
uint16_t wiced_blehidd_get_slave_latency (void)
```

Parameters

None

Returns

Current connection slave latency

5.9 BLE HIDD Get Supervision Timeout

Gets the link supervision timeout of the current connection.

Prototype

```
uint16_t wiced_blehidd_get_supervision_timeout (void)
```

Parameters

None

Returns

Current link supervision timeout

5.10 BLE HIDD Is Device Bonded

Checks whether the current BLE connected device is bonded with this device.

Prototype

```
wiced_bool_t wiced_blehidd_is_device_bonded (void)
```

Parameters

None

Returns

WICED_TRUE if device is bonded; WICED_FALSE otherwise

5.11 BLE HIDD Is Link Encrypted

Checks whether the current LE connection is encrypted.

Prototype

```
wiced_bool_t wiced_blehidd_is_link_encrypted (void)
```

Parameters

None

Returns

WICED_TRUE if connection is encrypted; WICED_FALSE otherwise

5.12 BLE HIDD Is Wakeup from Connection Request

Checks whether the wake up from SDS is due to receiving a BLE connect request.

Prototype

```
wiced_bool_t wiced_blehidd_is_wakeup_from_conn_req (void)
```

Parameters

None

Returns

WICED_TRUE if device was woken up due to receiving of the connect request; WICED_FALSE otherwise

5.13 HIDD Register for Periodic Poll

Registers with the controller to be called before the poll event from the master. For BLE links, the callback will be called before every connection event. For BR_EDR links, the callback will be called before every sniff or sniff subrate interval. Applications typically do not call this function directly. The function is called internally by the library in the wiced_ble_hidd_link_enable_poll_callback function after the application registers with the library using wiced_ble_hidd_link_register_poll_callback.

Prototype

```
void wiced_hidd_register_callback_for_poll_event (wiced_bt_transport_t transport, BD_ADDR
peer_bdaddr, wiced_bool_t enabled, void(*)(void *, uint32_t) callback)
```

Parameters

transport	: BT_TRANSPORT_BR_EDR or BT_TRANSPORT_LE
peer_bdaddr	: Peer device BD address
enabled	: WICED_TRUE to register the callback; WICED_FALSE to deregister
callback	: Callback function

Returns

None

5.14 BLE HIDD Set Asymmetric Slave Latency

If the HID host does not accept a requested connection parameter, the application can enable asymmetric slave latency to lower the power consumption.

Prototype

```
void wiced_blehidd_set_asym_slave_latency (uint16_t handle, uint16_t latency)
```

Parameters

`handle`	: Connection handle
`latency`	: Slave latency

Returns

None

5.15 BLE HIDD Set Device Bonded Flag

Sets the bonded flag for the currently connected BLE device. The application calls this function when a successful pairing is completed, and after a connection with a device has been reestablished.

Prototype

```
void wiced_blehidd_set_device_bonded_flag (wiced_bool_t is_bonded)
```

Parameters

`is_bonded`	: `WICED_TRUE` if connected BLE device is bonded; `WICED_FALSE` otherwise

Returns

None

5.16 BLE HIDD Set Link Encrypted Flag

Set the encrypted flag for the current BLE connection.

Prototype

```
void wiced_blehidd_set_link_encrypted_flag (wiced_bool_t is_encrypted)
```

Parameters

`is_encrypted`	: `WICED_TRUE` if link is encrypted; `WICED_FALSE` otherwise

Returns

None

5.17 BLE HIDD Register HID Report Table

The application should call this function to register the HID report table for sending and receiving. Each entry in the table specifies the handler to be invoked when a GATT packet is received or to be sent.

Prototype

```
void wiced_blehidd_register_report_table (wiced_blehidd_report_gatt_characteristic_t*
table, uint32_t num)
```

Parameters

`table`	: Pointer to the HID report table
`num`	: Number of items in the HID report table

Returns

None

5.18 BLE HIDD Send Report

Sends the HID report as a GATT notification.

Prototype

```
wiced_bt_gatt_status_t wiced_blehidd_send_report (uint16_t gatts_conn_id, uint8_t
report_id, wiced_hidd_report_type_t report_type, uint8_t *data, uint8_t length)
```

Parameters

`gatts_conn_id` : GATT connection ID

`report_id` : Report ID

`report_type` : Report type

`data` : Pointer to the report data

`length` : Length of the report data

Returns

'0' if the report was sent successfully; non-zero if send failed

5.19 BLE HIDD Write Handler

The function is called when a peer writes into a characteristic value or a characteristic descriptor of the device's GATT database. The function finds the attribute based on the attribute handle in the data structure passed to the function, and calls the appropriate attribute handle to process.

Prototype

```
wiced_bt_gatt_status_t wiced_blehidd_write_handler (void *data)
```

Parameters

`data` : Pointer to the Attribute Write request/command data

Returns

'0' if write is accepted; others if parsing failed

5.20 HIDD Event Queue Initialization

Initializes the HIDD event queue. The queue is empty upon creation.

Prototype

```
void wiced_hidd_event_queue_init (wiced_hidd_event_queue_t *queue, void *buffer, uint8_t
element_size, uint8_t max_elements)
```

Parameters

`queue` : Pointer to the HIDD event queue

`buffer` : Pointer to the buffer where the queue data will be stored (must have sufficient space to store `element_size * max_elements` bytes)

`element_size` : Maximum size of each element

`max_elements` : Maximum number of elements that can be kept in the queue (must be greater or equal to 2 - one of the elements will be used to provide an overflow slot functionality)

Returns

None

5.21 HIDD Event Queue Flash

Discards all elements in the queue, including any elements in the overflow slot.

Prototype

```
void wiced_hidd_event_queue_flush (wiced_hidd_event_queue_t *queue)
```

Parameters

`queue` : Pointer to the HIDD event queue

Returns

None

5.22 HIDD Event Queue Get Current Element

Returns the pointer to the first element in the queue. If the queue is empty, returns NULL.

Prototype

```
void* wiced_hidd_event_queue_get_current_element (wiced_hidd_event_queue_t *queue)
```

Parameters

`queue` - Pointer to the HIDD event queue

Returns

A pointer to the next element in the queue, or NULL if the queue is empty

5.23 HIDD Event Queue Get Number of Elements

Gets the number of elements currently in the queue.

Prototype

```
uint8_t wiced_hidd_event_queue_get_num_elements (wiced_hidd_event_queue_t *queue)
```

Parameters

`queue` : Pointer to the HIDD event queue

Returns

The number of elements in the queue

5.24 HIDD Event Queue Add Event with Overflow

Queues the given event into the HIDD event queue. If the HIDD event queue is full, then the function queues an overflow event.

Prototype

```
void wiced_hidd_event_queue_add_event_with_overflow (wiced_hidd_event_queue_t *queue,
wiced_hidd_event_t *event, uint8_t len, uint8_t poll_sequence_number)
```

Parameters

`queue` : Pointer to the HIDD event queue

`event` : Pointer to the event to be queued

`len` : Length of the event

`poll_sequence_number` : Poll sequence number

Returns

None

5.25 HIDD Event Queue Remove Current Element

Removes the current element from the queue. Does nothing if the queue is empty.

Prototype

```
void wiced_hidd_event_queue_remove_current_element (wiced_hidd_event_queue_t *queue)
```

Parameters

`queue` : Pointer to the HIDD event queue

Returns

None

5.26 HIDD Is Transport Detection Polling On

On power up or wake from SDS, the chip will poll the available transports (UART, USB, etc.) for some period of time to detect if any transport is present. This function checks whether transport detection polling is still in progress.

Prototype

```
wiced_bool_t wiced_hidd_is_transport_detection_polling_on (void)
```

Parameters

None

Returns

`WICED_TRUE` if transport detection is in progress; `WICED_FALSE` otherwise

5.27 HIDD Is Transport Detected

Checks whether a transport (UART, USB etc.) is detected.

Prototype

```
wiced_bool_t wiced_hidd_is_transport_detected (void)
```

Parameters

None

Returns

`WICED_TRUE` if transport is detected; `WICED_FALSE` otherwise

5.28 HIDD Get Current Native Bluetooth Clock

Gets the current value of the native Bluetooth clock.

Prototype

```
uint32_t wiced_hidd_get_current_native_bt_clock (void)
```

Parameters

None

Returns

The counter value that represents the native Bluetooth clock. The counter is 28 bits, and increases by 1 every 312.5 µs.

5.29 HIDD Get the Time Elapsed in Bluetooth Clock

Computes the time elapsed since "before" in 312.5-µs increments.

Prototype

```
uint32_t wiced_hidd_get_bt_clocks_since (uint32_t before)
```

Parameters

`before` - The previous counter value that was returned by `wiced_hidd_get_current_native_bt_clock`

Returns

The time elapsed in 312.5-µs increments

5.30 HIDD Link Initialization

Initializes the BLE HID link functionality. The function initializes the host info list, registers for the stack notifications, and initializes the timers to support an HID link.

Prototype

```
void wiced_ble_hidd_link_init (void)
```

Parameters

None

Returns

None

5.31 BLE HIDD Link Send Report

Formats and sends an HID report as a GATT notification. The function verifies whether the host is connected and registered for notifications for the characteristic corresponding to the specific report ID. Connection idle and SDS timers are restarted.

Prototype

```
wiced_bool_t wiced_ble_hidd_link_send_report (uint8_t report_id, wiced_hidd_report_type_t
report_type, uint8_t *data, uint8_t length)
```

Parameters

`report_id`	: Report ID
`report_type`	: Report type
`data`	: Pointer to the report data to be included in the report
`length`	: Length of the report data

Returns

`WICED_TRUE` if report is successfully scheduled for transmission; `WICED_FALSE` otherwise

5.32 BLE HIDD Link Is Connected

Checks whether the HID device is currently connected to the HID host

Prototype

```
wiced_bool_t  wiced_ble_hidd_link_is_connected (void)
```

Parameters

None

Returns

`WICED_TRUE` if HID is currently connected to the host; `WICED_FALSE` otherwise

5.33 BLE HIDD Link Is Discoverable

Checks whether the HID device is currently discoverable, i.e., whether the device is sending connectable undirected advertisements.

Prototype

```
wiced_bool_t  wiced_ble_hidd_link_is_discoverable (void)
```

Parameters

None

Returns

`WICED_TRUE` if HID is currently discoverable; `WICED_FALSE` otherwise

5.34 BLE HIDD Link Connect

Establishes a connection to the HID host. If a device is previously bonded, the function enters a reconnection procedure by sending Directed Connected advertisements. If a device is not bonded, the function starts sending undirected connectable advertisements.

Prototype

```
void wiced_ble_hidd_link_connect (void)
```

Parameters

None

Returns

None

5.35 BLE HIDD Link Disconnect

Terminates the current connection.

Prototype

```
void wiced_ble_hidd_link_disconnect (void)
```

Parameters

None

Returns

None

5.36 BLE HIDD Link Enable Application Poll

Enable application polling. If enabled, the callback function registered via **wiced_ble_hidd_link_register_poll_callback** will be called before the poll event from the master. For BLE links, the callback will be called before every connection event. For BR_EDR links, the callback will be called before every sniff or sniff subrate interval.

Prototype

```
void wiced_ble_hidd_link_enable_poll_callback (wiced_bool_t enable)
```

Parameters

`enable` : `WICED_TRUE` if polling shall be enabled; `WICED_FALSE` otherwise

Returns

None

5.37 BLE HIDD Link Register Application Poll Callback

Registers the application callback function to be called when the application is polled.

Prototype

```
void wiced_ble_hidd_link_register_poll_callback (void(*)(void)callback)
```

Parameters

`callback` : Pointer to the application callback function

Returns

None

5.38 BLE HIDD Link Virtual Cable Unplug

Removes the HID host information and sets the device as discoverable, i.e., starts connectable undirected advertising.

Prototype

```
void wiced_ble_hidd_link_virtual_cable_unplug (void)
```

Parameters

None

Returns

None

5.39 BLE HIDD Link AON Action Handler

The application calls this function to save or restore the HIDD Link variables when entering or exiting the SDS. The function saves and restores the HIDD link state and connection-related information.

Prototype

```
wiced_bool_t wiced_ble_hidd_link_aon_action_handler (wiced_bt_aon_driver_action  type,
void *ptr,  uint16_t size)
```

Parameters

`type` : `WICED_BT_AON_DRIVER_SAVE` or `WICED_BT_AON_DRIVER_RESTORE`

`ptr` : Pointer to the contents

`size` : Size of the contents

Returns

`WICED_TRUE` if action is successful; `WICED_FALSE` otherwise

5.40 BLE HIDD Link Update Connection Parameters

Initializes the BLE connection parameters update procedure.

Prototype

```
void wiced_ble_hidd_link_conn_param_update (void)
```

Parameters

None

Returns

None

5.41 BLE HIDD Link Set Slave Latency

Requests asymmetric slave latency. When the HID host does not accept the slave's connection parameter update request, the HIDD can enable asymmetric slave latency to reduce power consumption.

Prototype

```
void wiced_ble_hidd_link_set_slave_latency (uint16_t latency)
```

Parameters

`latency` : Slave latency in milliseconds

Returns

None

5.42 BLE HIDD Link Set Preferred Connection Parameters

Sets the BLE HIDD link preferred connection parameters.

Prototype

```
void wiced_ble_hidd_link_set_preferred_conn_params (uint16_t min_interval, uint16_t
max_interval, uint16_t latency, uint16_t timeout)
```

Parameters

`min_interval`	: Minimum connection interval
`max_interval`	: Maximum connection interval
`latency`	: Slave latency
`timeout`	: Link supervision timeout

Returns

None

5.43 BLE HIDD Link High Duty Cycle Directed Advertising Stopped

The application calls this function to inform the BLE HIDD link that high duty-cycle directed advertising is stopped.

Prototype

```
void wiced_ble_hidd_link_directed_adv_stop (void)
```

Parameters

None

Returns

None

5.44 BLE HIDD Link Advertising Stopped

The application calls this function to inform the BLE HIDD link that LE advertising (except high duty-cycle directed advertising) is stopped.

Prototype

```
void wiced_ble_hidd_link_adv_stop (void)
```

Parameters

None

Returns

None

5.45 BLE HIDD Link BLE Connection Up

The application calls this function to inform the BLE HIDD link that the BLE connection is up.

Prototype

```
void wiced_ble_hidd_link_connected (void)
```

Parameters

None

Returns

None

5.46 BLE HIDD Link BLE Connection Down

The application calls this function to inform the BLE HIDD link that the BLE connection is down.

Prototype

```
void wiced_ble_hidd_link_disconnected (void)
```

Parameters

None

Returns

None

5.47 BLE HIDD Link Add State Observer

The application calls this function to register itself to be notified when the BLE HIDD link state changes.

Prototype

```
void wiced_ble_hidd_link_add_state_observer (wiced_ble_hidd_state_change_callback_t
*callback)
```

Parameters

`callback`	: The function to be called when the BLE HIDD link state changes

Returns

None

5.48 BLE HIDD Host Info Initialization

Reads the BLE HID host information from the reserved NVRAM VSID section and initializes the BLE HIDD host information list.

Prototype

```
void wiced_ble_hidd_host_info_init (void)
```

Parameters

None

Returns

None

5.49 BLE HIDD Add Host Info First

Adds a new BLE HID host as the first host in the BLE HIDD host information list.

Prototype

```
void wiced_ble_hidd_host_info_add_first (uint8_t* bd_addr, uint8_t addr_type,
wiced_bt_device_link_keys_t* link_keys, uint16_t flags)
```

Parameters

`bd_addr`	: Bluetooth Device address of the host to be added
`addr_type`	: Bluetooth Device address type of the host to be added
`link_keys`	: Pointer to the link keys generated during the pairing with the host
`flags`	: Bitmap of the flags associated with the client characteristic configuration descriptor values

Returns

None

5.50 BLE HIDD Host Info Get Number of Hosts

Gets the number of BLE HID hosts stored in the BLE HIDD host information list.

Prototype

```
uint8_t wiced_ble_hidd_host_info_get_number (void)
```

Parameters

None

Returns

The number of HID hosts stored in the HIDD host information list.

5.51 BLE HIDD Host Info Is Bonded

Checks whether the HID device is bonded to the first host in the BLE HIDD host information list

Prototype

```
wiced_bool_t wiced_ble_hidd_host_info_is_bonded (void)
```

Parameters

None

Returns

WICED_TRUE if this device is bonded to the first host in the HIDD host information list; WICED_FALSE otherwise

5.52 BLE HIDD Host Info Get Bluetooth Device Address

Gets the Bluetooth device address of the first host stored in the HIDD host information list.

Prototype

```
uint8_t * wiced_ble_hidd_host_info_get_bdaddr (void)
```

Parameters

None

Returns

Pointer to the BD_ADDR of the first hosts stored in the HIDD host information list

5.53 BLE HIDD Host Info Get Link Keys

Gets the link keys for the first host stored in the BLE HIDD host information list.

Prototype

```
wiced_bt_device_link_keys_t *wiced_ble_hidd_host_info_get_link_keys (void)
```

Parameters

None

Returns

Pointer to the link keys structure for the first host in the BLE HIDD host information list

5.54 BLE HIDD Host Info Get Flags

Gets the flags value associated with the client characteristic configuration descriptor values of the first HID host in the BLE HIDD host information list.

Prototype

```
int32_t wiced_ble_hidd_host_info_get_flags (void)
```

Parameters

None

Returns

The flags value for the host if successful; -1 if no hosts are present in the BLE HIDD host information list

5.55 BLE HIDD Host Info Update Flags

The application calls this function to update a host's flags value in the stored host information when a corresponding client characteristic configuration descriptor is set or cleared by the connected host.

Prototype

```
uint16_t wiced_ble_hidd_host_info_update_flags (wiced_bool_t enable, uint16_t flag_bit)
```

Parameters

enable	: WICED_TRUE if the host enabled notifications for the corresponding characteristic; WICED_FALSE if notifications were disabled
flag_bit	: The bit associated with the characteristic being affected

Returns

The flags value after the operation is completed

5.56 BLE HIDD Host Info Delete

Deletes the information about the specific BLE HID host from the BLE HIDD host information list. The changes are automatically committed to the NVRAM.

Prototype

```
void wiced_ble_hidd_host_info_delete (uint8_t* bd_addr, uint8_t addr_type)
```

Parameters

bd_addr	: Bluetooth Device address of the host to be deleted
addr_type	: Bluetooth Device address type of the host to deleted

Returns

None

5.57 BLE HIDD Host Info Delete All

Cleans up the BLE HIDD host information list. The changes are automatically committed to the NVRAM.

Prototype

```
void wiced_ble_hidd_host_info_delete_all (void)
```

Parameters

None

Returns

None

5.58 BLE HIDD Host Info Get First Host

Gets the Bluetooth device address and address type of the first host stored in the HIDD host information list.

Prototype

```
wiced_bool_t wiced_ble_hidd_host_info_get_first_host (uint8_t** bd_addr, uint8_t*
addr_type)
```

Parameters

bd_addr	: Pointer to the location to return the address of the Bluetooth Device address of the host
addr_type	: Pointer to the variable to return the address type

Returns

WICED_TRUE if address and type were retrieved; WICED_FALSE otherwise

5.59 BLE HIDD Host Info Update First Host

Updates the Bluetooth device address and address type of the first host stored in the BLE HIDD host information list.

Prototype

```
wiced_bool_t wiced_ble_hidd_host_info_update_first_host (uint8_t* bd_addr, uint8_t
addr_type)
```

Parameters

bd_addr	: Bluetooth Device address of the host
addr_type	: Address type of the host

Returns

WICED_TRUE if address and type were updated; WICED_FALSE otherwise

References

1. Bluetooth Core Specification, Version 4.2 (see Bluetooth Core Specification 4.2)
2. HID Over GATT Profile (HOGP) Specification, Version 1.0 (see HID Over GATT Profile (HOGP))

Document Revision History

Document Title: ModusToolbox™ WICED HID Device Library
Document Number: 002-16556

Revision	ECN	Issue Date	Description of Change
**	5555181	07/31/2017	Initial release
*A	5907979	10/03/2017	Added CYW20719 and CYW20739 to Associated Part Family
*B	6302985	09/28/2018	Updated references to “WICED Studio” as “ModusToolbox”. Updated “Associated Part Family”: Replaced CYW20739 with CYW20721. Updated Sales page.
*C	6488541	02/19/2019	Added CYW20819 as an associated part

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 | PSoC 6 MCU

Cypress Developer Community

Community | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

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

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. No computing device can be absolutely secure. Therefore, despite security measures implemented in Cypress hardware or software products, Cypress does not assume any liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. In addition, the products described in these materials may contain design defects or errors known as errata which may cause the product to deviate from published specifications. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.