ModusToolbox™

WICED ANCS Library

Associated Part Family: CYW20706, CYW20735, CYW20719, CYW20819

Document Number: 002-16783 Rev. *B

Cypress Semiconductor

198 Champion Court

San Jose, CA 95134-1709

www.cypress.com

Contents

1 Introduction

This document describes the functionality of the WICED ANCS library that can be used in various applications. The library can be used to utilize the functionality of the Apple Notification Center Service [2] running on an iOS device. The document provides information on how the library can be accessed to discover characteristics of the ANCS service, how to register to receive notifications, process notifications and send commands to the iOS device for positive or negative action for control. It is assumed that the reader is familiar with the Bluetooth Core Specification [1].

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™ provides an 'ancs' sample application that utilizes the WICED ANCS library. If your installation of ModusToolbox or included SDKs do not contain the 'ancs' sample application, it can be downloaded from GitHub – see https://github.com/cypresssemiconductorco/Code-Examples-BT-20819A1-1.0-for-ModusToolbox-1.1 or the appropriate Git repo there for your version of ModusToolbox and SDK. While the applications themselves provide generic WICED application functionality, they use the WICED ANCS library to access the ANCS service on the iOS device over the Bluetooth Low Energy (BLE) GATT profile.

4 ANCS Application and ANCS Library

The ModusToolbox 'ancs' sample application together with the WICED ANCS library provides an implementation of the ANCS client protocol to access the ANCS service on an iOS device.

The application typically performs in a GAP peripheral role. At startup, it advertises as a peripheral device and allows an iOS device to connect. The application shall call the `wiced_bt_ancs_client_connection_up` and the `wiced_bt_ancs_client_connection_down` functions to notify the library about the connection state.

When a connection is established, the application performs the GATT discovery of the primary services of the connected device. If the application finds the ANCS service, it instructs the library to perform the discovery of the characteristics and descriptors of the ANCS service by calling the `wiced_bt_ancs_client_discover` function. When the discovery is completed, the library executes the `wiced_bt_ancs_discovery_complete_callback` function, passing the status of the discovery operation. During the discovery process, the application should pass the discovery result and discovery complete events to the library using the `wiced_bt_ancs_client_discovery_result` and the `wiced_bt_ancs_client_discovery_complete` function calls.

The application can instruct the library to start using the ANCS service by calling the `wiced_bt_ancs_client_start` function. At that time, the library registers with the iOS device to receive various notifications. Similarly, the application can deregister from the ANCS service on the iOS device by calling the `wiced_bt_ancs_client_stop` function. The results of the registration and deregistration are passed back to the application using `wiced_bt_ancs_start_complete_callback` and `wiced_bt_ancs_stop_complete_callback` functions.

While the library is performing the discovery or registrations with the iOS device, the application should not send any GATT requests to the iOS device on its own. Similarly, the application should not try to initialize two or more libraries at the same time.

Notifications received by the application on the handles that belong to the ANCS service should be forwarded to the ANCS library for processing. The library talks to the ANCS server on the iOS device to extract information and concludes by calling the application with the full ANCS notification event using `wiced_bt_ancs_notification_callback`.

The ANCS notification event includes the titles of ANCS positive and negative actions. For example, when the iOS device forwards the notification about an incoming call, the positive action will be to answer the call and the negative action will be to decline. The application can call `wiced_bt_ancs_perform_action` to perform selected actions, as demonstrated in the ModusToolbox 'ancs' application.

5 Library Reference

5.1 ANCS Client Initialize

The application should call this function to register application callbacks.

Prototype

```
void wiced_bt_ancs_client_initialize (wiced_bt_ancs_reg_t *p_reg)
```

Parameters

`p_reg`	: Registration control block that includes ANCS application callbacks.

Returns

None

5.2 ANCS Client Discover

The application should call this function when it discovers that a connected central device contains the ANCS service. The function initializes the ANCS library and starts the GATT discovery of ANCS characteristics.

After the application starts the discovery and until it receives the discovery complete callback (see Section 5.3), it shall pass the discovery results (see Section 5.4) and discovery complete events (see Section 5.5) to the library.

Prototype

```
wiced_bool_t wiced_bt_ancs_client_discover (uint16_t conn_id, uint16_t s_handle,
uint16_t e_handle)
```

Parameters

`conn_id`	: GATT connection ID.
`s_handle`	: Start GATT handle of the ANCS service.
`e_handle`	: End GATT handle of the ANCS service.

Returns

`WICED_TRUE` if the library started discovery successfully; `WICED_FALSE` otherwise.

5.3 ANCS Discovery Complete Callback

This callback is executed when the ANCS library completes the discovery of the ANCS service characteristics and descriptors.

Prototype

```
typedef void(* wiced_bt_ancs_discovery_complete_callback_t) (uint16_t conn_id,
wiced_bool_t success)
```

Parameters

`conn_id`	: GATT connection ID.
`status`	: `WICED_TRUE` if initialization completed successfully; `WICED_FALSE` otherwise.

Returns

None

5.4 ANCS Client Discovery Result

While the library performs the GATT discovery, the application shall pass the discovery results received from the stack to the ANCS library. The library must locate all the characteristics that belong to the ANCS service and associated client characteristic configuration descriptors.

Prototype

```
void wiced_bt_ancs_client_discovery_result (wiced_bt_gatt_discovery_result_t *p_data)
```

Parameters

`p_data`	: Discovery result data as passed from the stack.

Returns

None

5.5 ANCS Client Discovery Complete

While the library performs the GATT discovery, the application shall pass the discovery complete callbacks to the ANCS library. As the GATT discovery consists of multiple steps, this function initiates the next discovery request or write request to configure the ANCS service on the iOS device.

Prototype

```
void wiced_bt_ancs_client_discovery_complete(wiced_bt_gatt_discovery_complete_t *p_data)
```

Parameters

`p_data`	: Discovery complete data as passed from the stack.

Returns

None

5.6 ANCS Client Start

The application calls this function to start an ANCS client. The GATT discovery should be completed before this function is called. The start function configures the ANCS server on the iOS device for notification and configuration information that the client wants to monitor.

Prototype

```
wiced_bool_t wiced_bt_ancs_client_start (uint16_t conn_id)
```

Parameters

`conn_id`	: GATT connection ID.

Returns

`WICED_TRUE` if the operation has been initiated successfully; `WICED_FALSE` otherwise.

5.7 ANCS Start Complete Callback

This callback is executed when the ANCS library completes a startup operation of the ANCS.

Prototype

```
typedef void (*wiced_bt_ancs_start_complete_callback_t) (uint16_t conn_id, wiced_bool_t
success)
```

Parameters

`conn_id`	: GATT connection ID.
`status`	: `WICED_TRUE` if initialization completed successfully; `WICED_FALSE` otherwise.

Returns

None

5.8 ANCS Client Stop

The application calls this function to deregister with the ANCS server on the iOS device.

Prototype

```
wiced_bool_t wiced_bt_ancs_client_start (uint16_t conn_id)
```

Parameters

`conn_id`	: GATT connection ID.

Returns

`WICED_TRUE` if the operation has been initiated successfully; `WICED_FALSE` otherwise.

5.9 ANCS Stop Complete Callback

This callback is executed when the ANCS library completes the deregistration with the ANCS server on the iOS device.

Prototype

```
typedef void (*wiced_bt_ancs_stop_complete_callback_t) (uint16_t conn_id, wiced_bool_t
success)
```

Parameters

`conn_id`	: GATT connection ID.
`status`	: `WICED_TRUE` if operation completed successfully; `WICED_FALSE` otherwise.

Returns

None

5.10 ANCS Client Connection Up

The application should call this function when a BLE connection with a peer device has been established.

Prototype

```
void wiced_bt_ancs_client_connection_up (wiced_bt_gatt_connection_status_t
*p_conn_status)
```

Parameters

`p_conn_status`	: pointer to a GATT connection status structure that includes the address and connection ID.

Returns

None

5.11 ANCS Client Connection Down

The application should call this function when a BLE connection with a peer device has been disconnected.

Prototype

```
void wiced_bt_ancs_client_connection_down (wiced_bt_gatt_connection_status_t *p_conn_status)
```

Parameters

`p_conn_status`	: pointer to a GATT connection status structure that includes the address and connection ID.

Returns

None

5.12 ANCS Client Write Response

The application should call this function when it receives a GATT Write Response for an attribute handle that belongs to the ANCS service.

Prototype

```
void wiced_bt_ancs_client_write_rsp (wiced_bt_gatt_operation_complete_t *p_data)
```

Parameters

`p_data`	: Pointer to a GATT operation complete data structure.

Returns

None

5.13 ANCS Notification Callback

The ANCS library executes this callback when it receives a complete notification from the iOS device with a new ANCS event.

Prototype

```
typedef void (*wiced_bt_ancs_notification_callback_t) (uint16_t conn_id, ancs_event_t
event)
```

Parameters

`conn_id`	: GATT connection ID.
`event`	: Received ANCS event.

Returns

None

5.14 ANCS Perform Action

The application calls this function to send the command to the iOS device to perform a specified action. The action command (for example: answer the call, or clear notification), is sent as a response to a notification. The UID of the notification is passed back in this function along with the action ID.

Prototype

```
wiced_bt_gatt_status_t wiced_bt_ancs_perform_action (uint16_t conn_id, uint32_t uid,
uint32_t action_id)
```

Parameters

`conn_id`	: GATT connection ID.
`uid`	: Unique ID of the notification that the action is in response to.
`action_id`	: Positive or negative action ID.

Returns

`WICED_TRUE` if action command is successfully passed to the iOS device; `WICED_FALSE` otherwise.

References

[1] Bluetooth Core Specification, Version 4.2 (see Bluetooth Core Specification 4.2)

[2] Apple Notification Center Service (ANCS) Specification (see Apple Notification Center Service)

Document Revision History

Document Title: WICED ANCS Library

Document Number: 002-16783

Revision	ECN	Issue Date	Description of Change
**	5555181	07/31/2017	Initial release
*A	6336312	10/08/2018	Replaced “WICED Studio” with “ModusToolbox”. Updated Sales page.
*B	6486559	02/15/2019	Updated Design and Architecture Updated template

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 | Code Examples | Projects | Videos | Blogs | Training| Components

Technical Support

cypress.com/support