AIROC™ Application Buffer Pools

ModusToolbox™

About this document

Intended audience

AIROC™ Bluetooth® SDK (BTSDK) embedded developers using any of the following device families (all use BTSTACK versions lower than 3.0):
CYW20706, CYW20719, CYW20721, CYW20735, CYW20819, CYW20820, CYW20835, CYW30739, CYW43012

Table of contents

1 Introduction

This document provides a description of buffers used by the application and the upper layer stack of the AIROC™ Bluetooth® Stack (BTSTACK). This only applies to devices that contain BTSTACK versions lower than 3.0.

In BTSTACK versions 3.0 and greater, buffer management is handled internally by the stack, but applications have access to heap and buffer pools via BTSTACK APIs, see document number 002-34884 “AIROC™ BTSTACK-v3.X Application Memory Management”.

Currently, only CYW55572 and CYW20829 based devices use BTSTACK version 3.0.

2 IoT resources

Infineon provides a wealth of data at **https://www.infineon.com/cms/en/about-infineon/make-iot-work/iot-solutions/** to help you to select the right IoT device for your design, quickly and effectively integrate the device into your design. Infineon 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 **Infineon support community** website.

3 Stack buffer pools

To avoid fragmentation and promote efficient data transfer, dynamic memory is organized in buffers. A buffer can be allocated either by an application or the stack. A pointer to the buffer is passed down or up the stack without additional copying. In the forward direction, the buffer is released by the HCI transport layer when it is delivered to the controller component of the chip. In the reverse direction, a buffer is allocated by the HCI transport layer, moved through the stack by passing the pointer, and released after the data has been consumed by the appropriate layer of the stack or by the application.

The buffers are organized in pools. Four buffer pools (Pool IDs 0 – 3) are automatically allocated by the stack using the configuration provided by the application in the `wiced_bt_cfg_buf_pool_t` structure passed to the `wiced_bt_stack_init` function. The buffers are used by the stack for building and parsing packets for various profiles and protocols, such as AVCTP, AVDTP and RFCOMM, for GATT and SDP databases, EIR data, and so forth. In the `wiced_bt_cfg_buf_pool_t` configuration structure, the four stack pools must be ordered by increasing buffer sizes. When the stack requests a buffer of a specified size (and if the corresponding pool is exhausted), the next pool is used, although this situation should be avoided if possible. The size and the number of buffers in each pool are determined by the application use case and Bluetooth® profiles used. The following table specifies major use cases and buffer utilization.

Table 1 Pool ID used by different stack layers

Stack layer	Pool ID	Description
Bluetooth® Management, HCI Commands	1	Internal to the stack, and used in all applications. The HCI layer may queue up to 5 commands during the device startup. Typically, 1-2 buffers can be used at the connection establishment time and during the connection.
L2CAP Control	1	Used by all data connections. Typically requires up to 2 buffers during connection establishment.
SDP	2	Used by all BR/EDR connections. Typically, 1 buffer is used during the service search, but up to 2 buffers may be used to construct SDP responses. Used during connection establishment, but not during operation.
RFCOMM Control	1	Used by handsfree, handsfree audio gateway, and serial port profile. Typically requires 1-2 buffers during connection establishment in both originating and terminating connections.
RFCOMM Data	2	Used by handsfree, handsfree audio gateway, serial port profile. Handsfree and handsfree audio gateways use a single buffer for AT command construction and parsing. The serial port profile application may use this pool, or allocate an application pool if more control is required. See section **4**.
AVDT Control	1	Used by the AV source and AV sink profiles. Uses 1 buffer for the connection establishment and for AV control (suspend/resume/start) operations.
AVDT Data	Proprietary	Used by the AV source profile for audio raw data. A special memory chunk is allocated using the `wiced_audio_buffer_initialize` function.
AVRC Commands	2	Used by the AV remote control profile. Uses 1-2 buffers to construct and parse remote control messages, including simple commands and AVRC metadata.

Stack layer	Pool ID	Description
GATT data	2	Used by all LE GATT applications.
Received data	2	All received data packets are copied to a buffer from pool ID 2 and processed by the stack in the same thread, or are consumed by the application.

The size and number of buffers in each pool are determined by the application use case and profiles used. The following table specifies major use cases and buffer utilization.

Table 2 Typical buffer pools configuration

Pool ID	Description	Size	Count
0	Small buffer pool	64	12
1	Medium buffer pool	360	8
2	Large buffer pool	720	5
3	Extra-large buffer pool	1024	0

The small buffer pool is used for all transfers that require a minimal number of bytes (such as GATT discovery request, AVDTP set configuration/get capabilities, and so forth). The number of buffers in the pool should be set to at least 12 and the size of the buffer is recommended to be set to 64.

Medium sized buffers are used where data does not fit into a small buffer (such as EIR data, RFCOMM control packets or AVRC commands).

The buffer size in the large buffer pool is dependent on the MTU value supported by the application. This should be set to an application-defined MTU value, plus an additional 12 bytes to accommodate the AIROC™ internal header.

Extra-large buffers are required if still larger packet sizes are expected. Extra-large buffers are not required for the current ModusToolbox™ version. The buffer count of the extra-large pool can be set to zero.

4 Application buffer pools

Buffer pools are shared in different use cases. The buffers from the same pool can be used for HCI commands when a connection is being established, and subsequently for GATT discovery or data transfer. In some cases, the application may require complete control of a pool. In this case, a separate application buffer pool can be allocated and managed by the application. For example, when data is being sent over the RFCOMM channel, the application can create a separate pool. This will ensure that a high-speed connection will not use all buffers in the system. The number of buffers currently in use may be used by the flow control functionality.

To create a pool, the application can call the `wiced_bt_create_pool` function by passing the number of buffers and the size of the buffer. Functions `wiced_bt_get_buffer_from_pool` and `wiced_bt_free_buffer` can be used to allocate a buffer from the pool and free the buffer back to the pool. Note that the current version of ModusToolbox™ does not allow release of memory allocated by the `wiced_bt_create_pool` function.

See documentation in *wiced_memory.h* for details.

In addition to `wiced_bt_create_pool`, there are other AIROC™ APIs that will result in additional buffer pools being created by those APIs. For example, when creating worker threads or queues, the stack will create additional buffer pools. Some AIROC™ APIs also create their own queues internally. An application must increment the `max_number_of_buffer_pools` member of the `wiced_bt_cfg_settings_t` configuration structure passed to the `wiced_bt_stack_init` function for each invocation of these APIs in the application. The APIs that require incrementing this value are listed here:

```
wiced_bt_create_pool()
wiced_rtos_init_queue ()
wiced_rtos_init_worker_thread ()
wiced_bt_rfcomm_set_buffer_pool()
wiced_bt_avrc_set_buffer_pool()
```

By default, the `max_number_of_buffer_pools` value is 4. So, if an application creates two application buffer pools and a queue, `max_number_of_buffer_pools` would need to be changed to 7.

Note: *Applications do not need to modify anything in the* `wiced_bt_cfg_buf_pool_t` *structure for these buffer pools created with the above APIs; that structure is only for the default stack pools.*

5 Special pools

If the application requires transport connection to exchange information with an MCU, then it typically uses the AIROC™ transport process which supports data exchange over the HCI UART or SPI. The transport connection uses special Transport Rx and Transport Tx buffer pools.

The transport Rx buffer is allocated when the application calls the `wiced_transport_init` function. The size of a buffer and number of buffers in the pool are declared in the transport configuration which is passed as a parameter. Typically, the application needs two buffers of up to 1024 byte each. However, the size depends on the protocol used between the MCU and the application.

If the application does not transmit data chunks to the MCU of more than 264 bytes, the transport speed is sufficiently fast and the MCU never flow controls the application, then there is no need to allocate a Transport Tx pool. Otherwise, the application can call the `wiced_transport_create_buffer_pool` function passing the buffer size and number of buffers as parameters.

Note: *Application uses the same Transport Tx pool if it is configured to send traces over the HCI UART.*

6 Buffer usage statistics

To interpret the stack buffer pool usage, and to identify if there is a possibility of exceeding buffer capacity for a given use case, the application can call the `wiced_bt_get_buffer_usage` function. For all pools currently used by the application, the function returns the total number of buffers in the pool and size of buffers, the number of buffers currently in use, and the maximum number of buffers that have been in use since the start of the system. To fine tune the number of buffers required for a given scenario, check the `max_allocated_count` after the scenario of interest is run and compare it to the `total_count`. If necessary (such as when `max_allocated_count` equals `total_count`), the buffer count for that pool should be increased.

When calling the `wiced_bt_get_buffer_usage` function, sufficient memory should be allocated to ensure it returns statistics for all pools created by the application, along with the stack buffers configured by the application. See the following code snippet example.

Code Listing 1 Buffer use statistics

```
wiced_bt_buffer_statistics_t buff_stats[ wiced_bt_get_number_of_buffer_pools() ];
uint8_t i;

if( wiced_bt_get_buffer_usage( buff_stats, sizeof(buff_stats) ) == WICED_BT_SUCCESS )
{
    WICED_BT_TRACE ("Buffer usage statistics:\n");

    for( i = 0; i < wiced_bt_get_number_of_buffer_pools(); i++)
        WICED_BT_TRACE ("pool_id:%d size:%d curr_cnt:%d max_cnt:%d total:%d\n",
                         buff_stats[i].pool_id, buff_stats[i].pool_size,
                         buff_stats[i].current_allocated_count,
                         buff_stats[i].max_allocated_count,
                         buff_stats[i].total_count);
}
```

Revision history

Document version	Date of release	Description of changes
**	2016-09-17	Initial release
*A	2017-08-23	Updated template
*B	2018-09-19	Updated for ModusToolbox™
*C	2019-02-15	Added as CYW20819 as an associated part
*D	2019-04-23	Removed Associated Part Family
*E	2019-12-11	Updated Application buffer pools
*F	2021-11-18	Branding updates
*G	2022-02-28	Updated Introduction Updated reference link in IoT resources

Edition 2022-02-28

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Go to www.infineon.com/support

Document reference

002-16403 Rev. *G