AIROC™ BTSTACK architecture

About this document

Scope and purpose

This document introduces the software architecture of Infineon's host Bluetooth® stack (BTSTACK) that provides an API to create Bluetooth® applications based on AIROC™ Bluetooth® devices.

Intended audience

This document is intended for anyone creating Bluetooth® applications using Infineon's AIROC™ Bluetooth® devices.

Table of contents

1 Introduction

This document introduces the software architecture of Infineon Host Bluetooth® stack (BTSTACK) that provides an API to create Bluetooth® applications. BTSTACK implements protocols and profiles as required by the Bluetooth® SIG for creating and certifying Bluetooth® applications and components. BTSTACK is supported across multiple platforms and is built and packaged as a BTSTACK Library.

Note: BTSTACK function APIs and typedefs are prefixed with `wiced_bt_`. BTSTACK macros are prefixed with `WICED_BT_`. BTSTACK functional APIs are referred to as "WICED APIs" or "BTSTACK APIs" in this document.

2 Highlights

- Complies with Bluetooth® SIG version 5.3
- Platform-, processor-, and OS-agnostic. The BTSTACK library has been ported across multiple platforms, including various Arm® Cortex®-M class MCUs and A class processors on Linux, FreeRTOS, ThreadX, and others on various platforms.
- Can be built as a Bluetooth® Low Energy (Bluetooth® LE) only or dual-mode (Bluetooth® LE and Bluetooth® Basic Rate/Enhanced Date Rate (BR/EDR)) build variant
- Highly optimized for code size and RAM. The code size is automatically optimized by the application Makefile and is dependent on the features used by the application. The RAM size is optimized by application configuration options.
- Supports all Bluetooth® LE and BR/EDR features and profiles:
  - Bluetooth® LE: GATT, beacon, ISOC, Extended advertising, Bluetooth® LE audio, Mesh, PAwR, LE CoC, LE-LR
  - BR-EDR: SDP, RFCOMM, A2DP, AVRCP, SPP, MAP, PBAP, OPP, PAN, and HID
- Requires minimal OS/platform resources that are provided to it through a platform interface structure typedef as `wiced_bt_stack_platform_t`. The `wiced_bt_stack_platform_t` structure contains a set of function pointers and platform settings for tracing, which are set by the platform porting layer. The following interfaces are supplied by the porting layer:
  - Memory: Functions to allocate and free memory (for example, `malloc` and `free`).
  - Locking mechanism for multi-threaded apps: A lock and unlock function needs to be supplied. This could be a mutex, semaphore, or interrupt locking as required for the platform.
  - Timer: The BTSTACK library manages multiple internal timers with a single acting OS/platform timer. The platform timer is expected to have a resolution of 64 bits. The platform timer should allow setting of timeouts in the range of a few milliseconds to a few seconds.
  - Threads: The BTSTACK library does not create threads. BTSTACK APIs and callbacks are invoked in the context of the application created threads.
- Can work in single or multi-threaded environments
- Entry points into the BTSTACK library:
  - Downstream path: Using the BTSTACK API function calls from the application to the controller
  - Upstream path: Application-registered function callbacks for events and data received from other Bluetooth® devices
- Additional application and platform requirements for BTSTACK include NVRAM support to store the Bluetooth® security key information exchanged with remote devices.

3 Module view

The following figure shows a module view of the application and the constituents of the BTSTACK library.

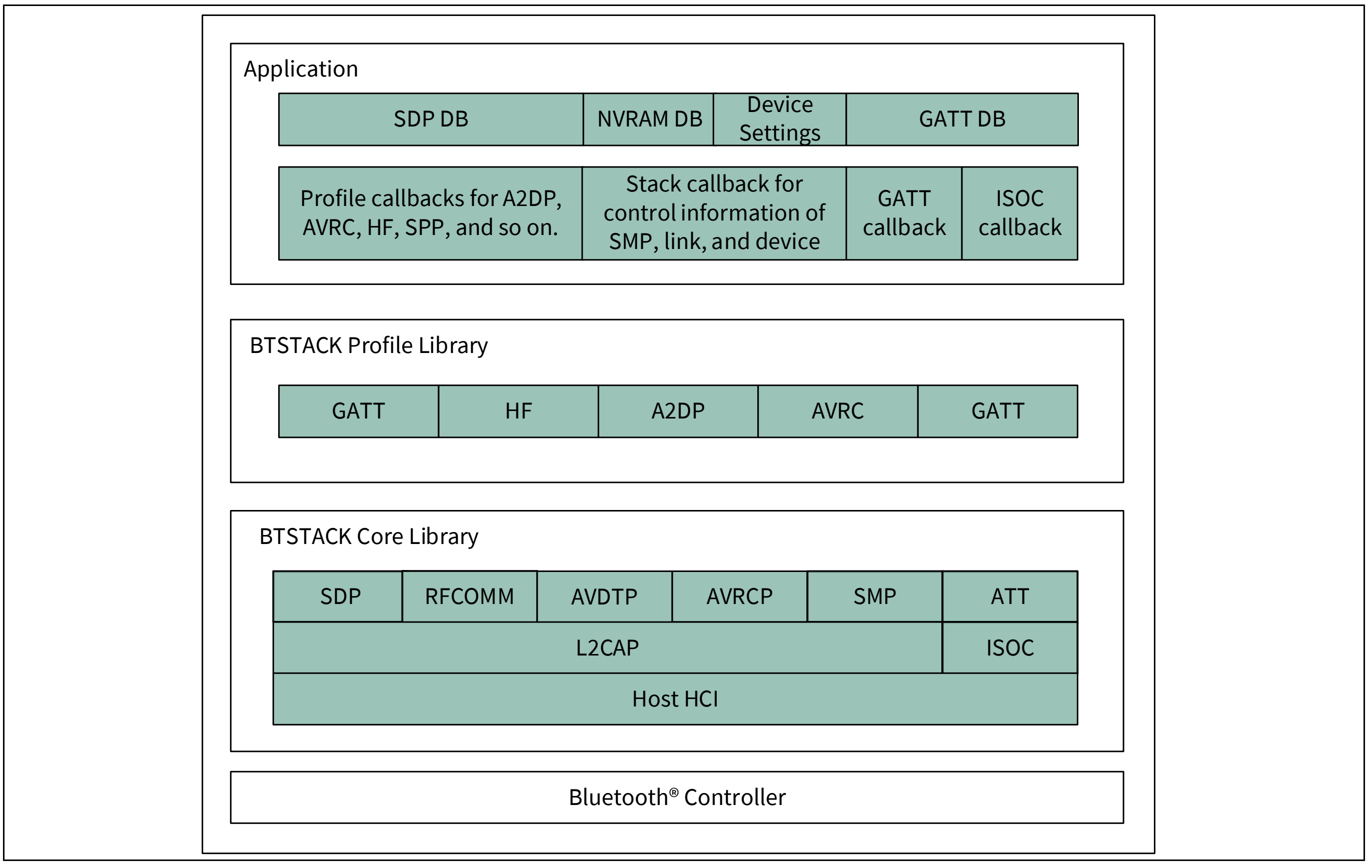

Figure 1 Stack layers

The BTSTACK library consists of the core and profile library layers.

3.1 BTSTACK Core library

The BTSTACK Core library implements the core Bluetooth® protocols, which include:

- L2CAP (Logical Link Control and Adaptation Protocol)
- ATT (Attribute Protocol)
- SMP (Security Manager Protocol)
- SDP (Service Discovery Protocol) (BR/EDR only)
- RFCOMM (RF Communication Protocol) (BR/EDR only)
- AVDTP (Audio/Video Distribution and Transport Protocol) (BR/EDR only)
- AVCTP (Audio/Video Control Transport Protocol) (BR/EDR only)

The BTSTACK Core also implements commands and events required to support audio over Bluetooth® with the following:

- SCO/ESCO (Synchronous Connection Oriented/Extended Synchronous Connection Oriented) data packets
- ISOC (Isochronous Data Packets)

3.2 BTSTACK Profile library

The BTSTACK Profile library includes the following Bluetooth® profiles that operate on top of the protocols implemented in the Core layer.

- GATT (Generic Attribute Profile)
- SPP (Serial Port Profile) (BR/EDR only)
- HF (Hands Free Profile)), HF AG (Hands Free Audio Gateway) (BR/EDR only)
- A2DP (Advanced Audio Distribution Profile) (BR/EDR only)
- AVRCP (Audio Video Remote Control Profile) (BR/EDR only)
- MAP (Message Access Profile) (BR/EDR only)
- PBAP (Phone Book Access Profile) (BR/EDR only)
- PAN (Personal Area Networking Profile)(BR/EDR only)
- OPP (Object Push Profile) (BR/EDR only)
- HID (Human Interface Device Profile) (BR/EDR only)
- HOGP (HID over GATT Profile)
- Mesh (Protocols and Profiles)
- LE Audio (Profiles)

New protocols or profiles may be implemented which could be built on top of the existing protocols/profiles.

3.3 Application modules

A Bluetooth® application written over BTSTACK is expected to set up the application's Bluetooth® device configuration. The application can use the Bluetooth® Configurator to setup the Bluetooth® device configuration. The application Bluetooth® configuration includes the following:

- Device settings: The `wiced_bt_cfg_settings_t` structure that contains the configuration information used to initialize BTSTACK.
- GATT database: A GATT Server application includes a GATT database (GATT DB) that lists supported services, characteristics, and descriptors.
- SDP database (SDP DB): A BR/EDR application includes a Service Discovery Protocol Database (SDP DB) to list the services and attributes offered by the application.

Additionally, if the Bluetooth® application supports Bluetooth® bonding, the application must implement a method to save security material generated during the pairing and delivered by BTSTACK in the SMP callback.

The communication across the Bluetooth® and Bluetooth® LE protocol and profile layers is via API calls from the application to BTSTACK and callbacks from BTSTACK to the application.

4 Deployment options

In each BTSTACK deployment, the BTSTACK Core and Profile components are collocated on the same processor, while the application code and logic may be functionally split across one or more application MCUs. BTSTACK communicates with the controller using the Host Controller Interface (HCI) protocol for sending and receiving commands, events, and data to and from the Bluetooth® controller. When the application is collocated on the same processor as the stack, a simple `wiced_bt_*` API call and callback mechanism are used.

The following are the typical deployments for the BTSTACK and applications:

4.1 Embedded Bluetooth® deployment

In Embedded Bluetooth® deployment, the Bluetooth® application using the BTSTACK Core and Profile libraries coexist with the Bluetooth® controller on the same MCU. For example, the headset BTSTACK library is collocated with the Bluetooth® controller, sharing the RAM and CPU resources. BTSTACK communicates with the controller over message queue API calls to and from the Bluetooth® controller using the standard HCI protocol for sending and receiving commands, events, and data.

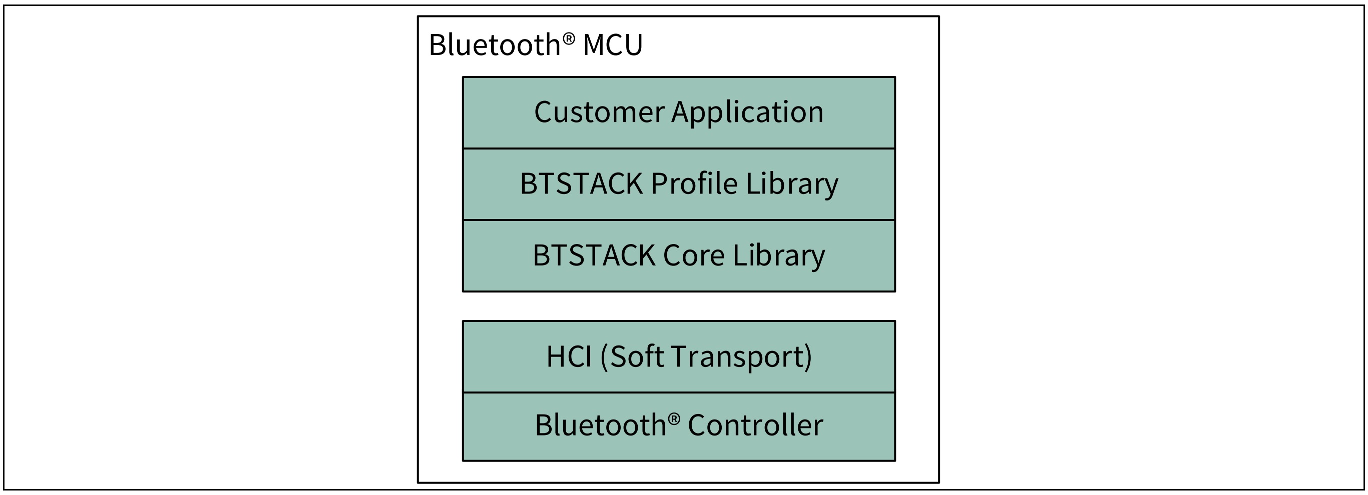

Figure 2 Embedded Bluetooth® deployment

4.1.1 Partial embedded mode interface (WICED HCI)

A Bluetooth® product could have an onboard MCU that uses a Bluetooth® application in either of the deployment options to provide Bluetooth® functionality. For such a product, the MCU software would likely be used to control the device through a UART or SPI interface via a protocol that allows the MCU to send and receive commands, events, and data. A sample protocol, referred to as the "WICED HCI Control Protocol", can be used in the application mode to support the communication between an MCU (host) and a Bluetooth® application. For more information on WICED HCI, see WICED HCI UART Control Protocol user guide.

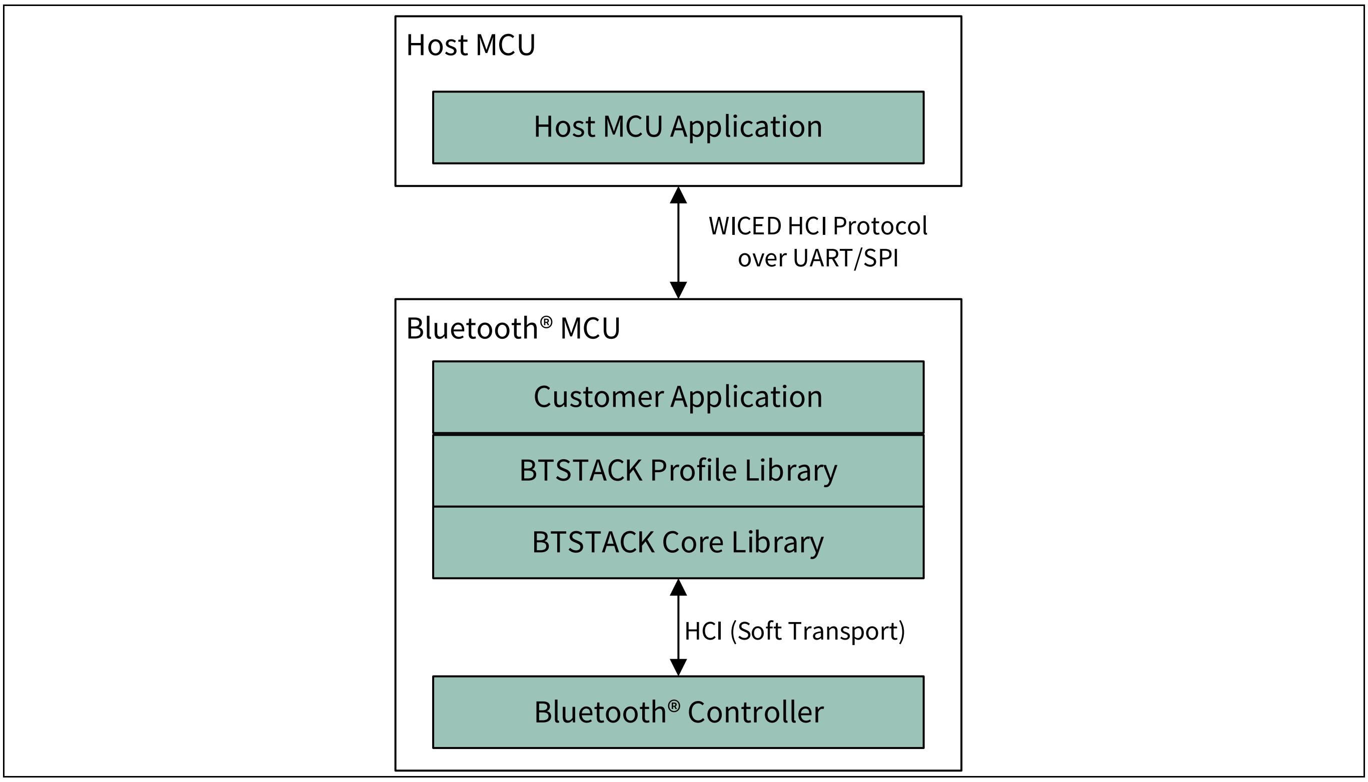

Figure 3 Partial embedded mode interface

4.2 Host Bluetooth® deployment

In Hosted Bluetooth® deployment, the Bluetooth® application using the BTSTACK Core and Profile libraries runs on a separate MCU and communicates with the Bluetooth® controller using HCI commands, events, and data over a transport such as UART, SPI, or IPC. The BTSTACK library is located on a separate MCU and does not share the RAM and CPU resources with the Bluetooth® controller.

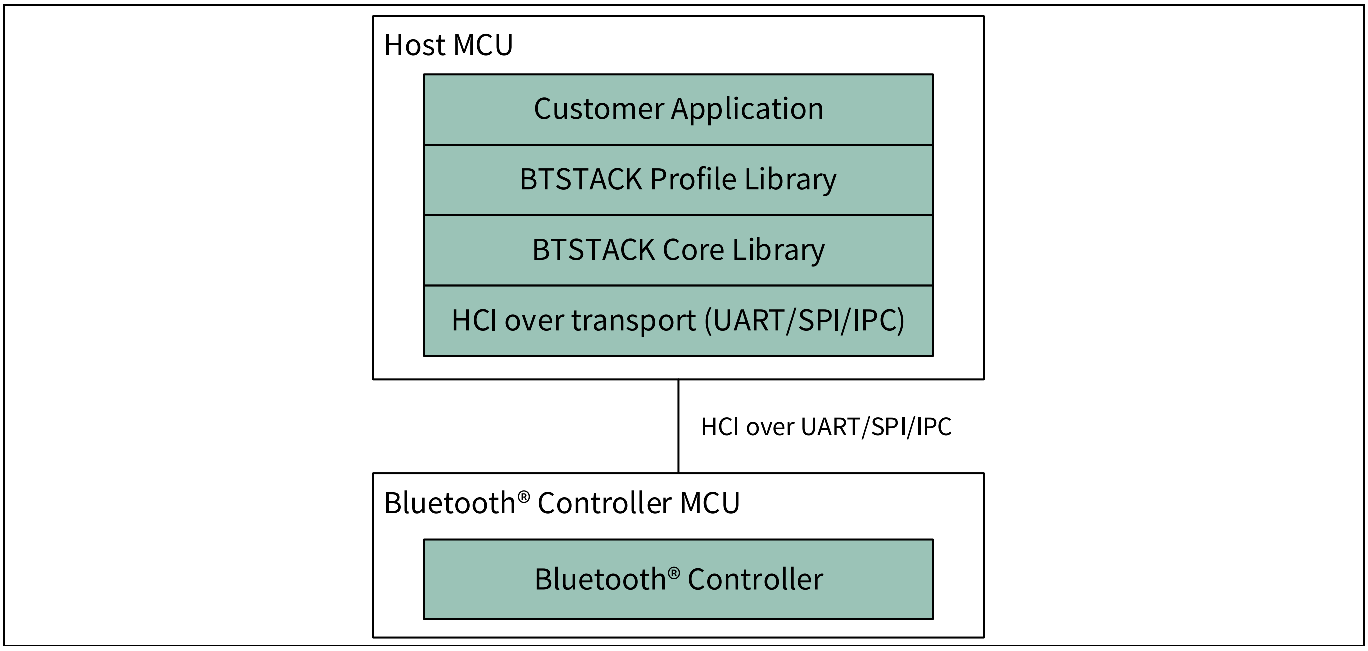

Figure 4 Hosted Bluetooth® deployment

4.2.1 A2DP audio offload

BTSTACK supports the offload of audio data from the host to the controller.

BTSTACK supports the A2DP hardware offload through HCI vendor-specific commands. Audio offload involves offloading the A2DP audio encoding and decoding to an audio processor (LiteHost) attached to the Bluetooth® controller. The encoded audio data stream passes directly from LiteHost to the Bluetooth® controller without the involvement of the Bluetooth® host. The Bluetooth® host is responsible for the configuration and control of the A2DP session.

In the case of an A2DP Source, the host sends PCM data to LiteHost. This data is encoded with the audio codec (for example, SBC) as specified by the application. In the case of an A2DP Sink, LiteHost decodes the incoming A2DP data and can either play it out over the attached DAC or send the decoded data up to the host. Audio offload in BTSTACK is compatible with Android A2DP hardware Offload.

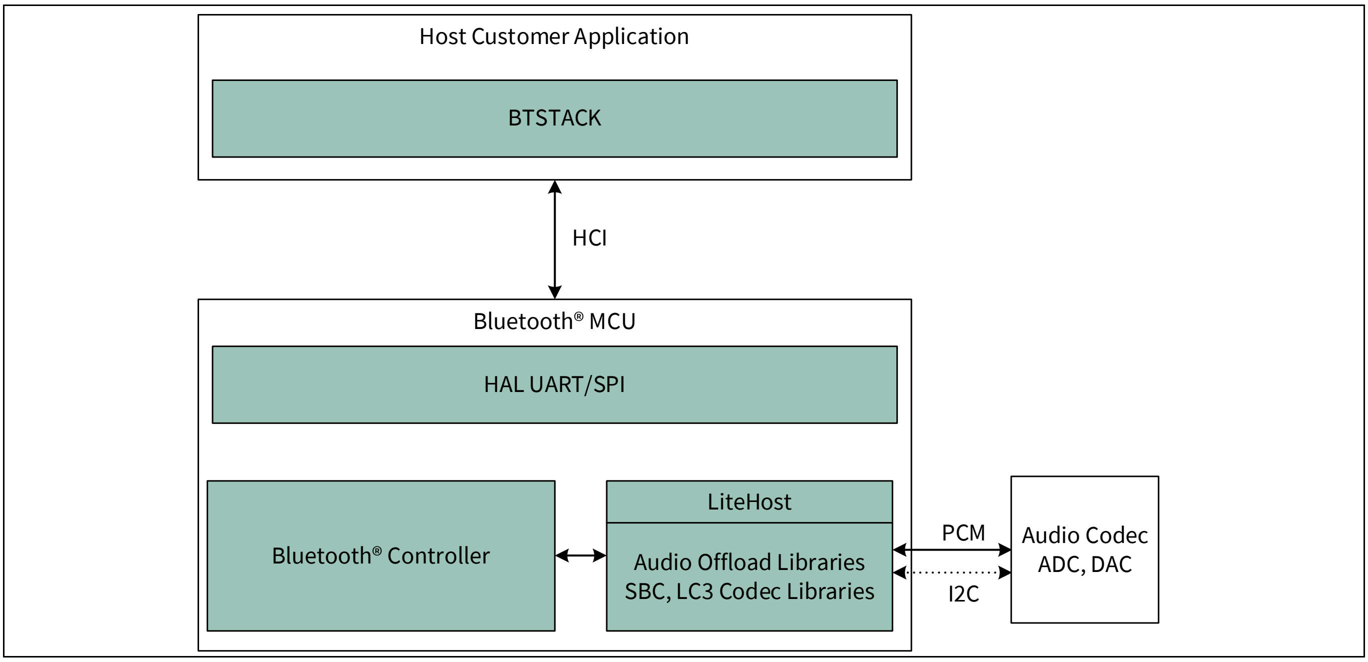

Figure 5 A2DP audio offload over HCI

5 Managing memory allocations for data transfers

5.1 Memory management

A fundamental construct of the BTSTACK architecture is the concept of memory ownership. On startup, BTSTACK allocates the memory it needs for internal use. This memory is not available to be used by applications. The size of the memory allocated depends on the configuration supplied by the application.

Applications need static or dynamically allocated RAM to function. Applications are allowed to use native OS functions to allocate and free the RAM. To assist in portability across multiple operating systems, BTSTACK provides utility functions to create and use memory from dedicated private memory heaps and buffer pools. Buffer pools may be thought of as heaps that support a fixed-size allocation. The use of buffer pools can be much more efficient for certain types of applications that transfer blocks of data of fixed sizes.

Applications are free to create multiple heaps and/or buffer pools. One of the application heaps should be designated as “default”. The default heap is used to allocate the memory using the `wiced_bt_get_buffer` API.

Note: Most BTSTACK libraries require that the application create a heap marked as default. The following figure illustrates the separation of the application and BTSTACK memory.

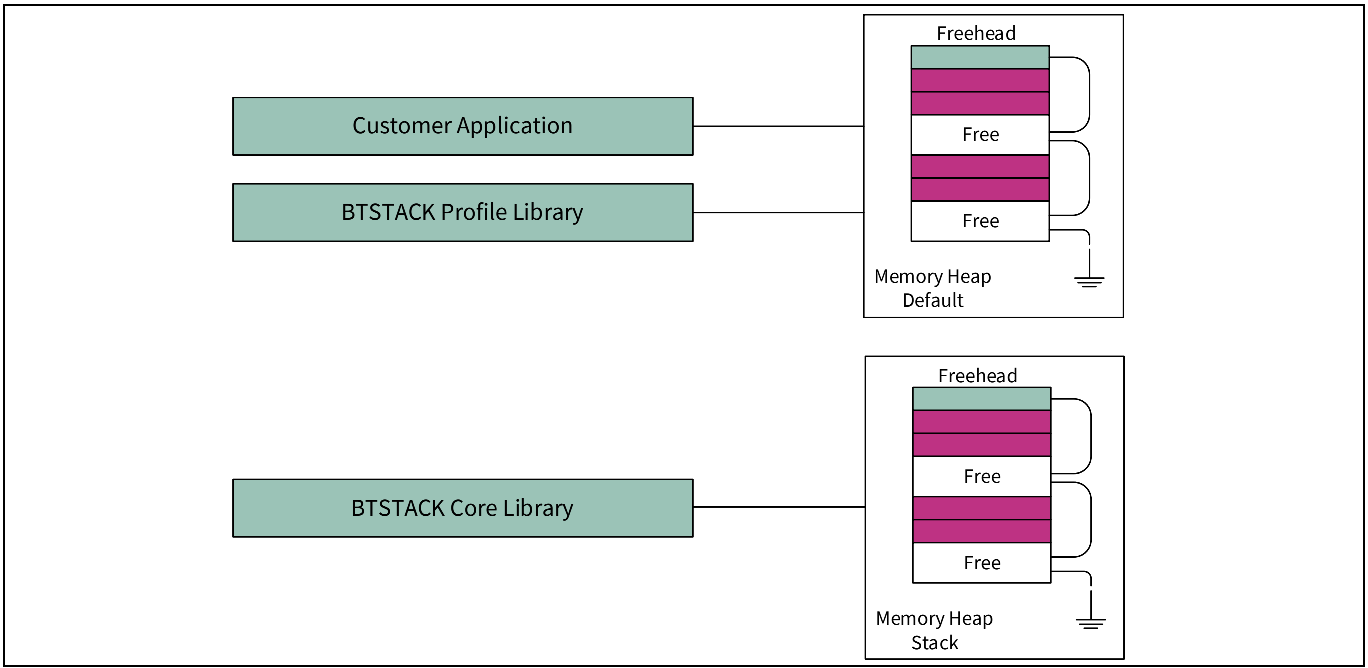

Figure 6 Memory management using heaps

5.2 Downstream memory management

In the downstream path (application to BTSTACK), to transmit data to a peer Bluetooth® device, a pointer to a memory block (allocated or global) containing data to be transmitted is passed down to BTSTACK via an API call. Upon invocation of the call, the contents of the data packet passed must not be changed by the application until BTSTACK indicates the completion of the transmission via a callback.

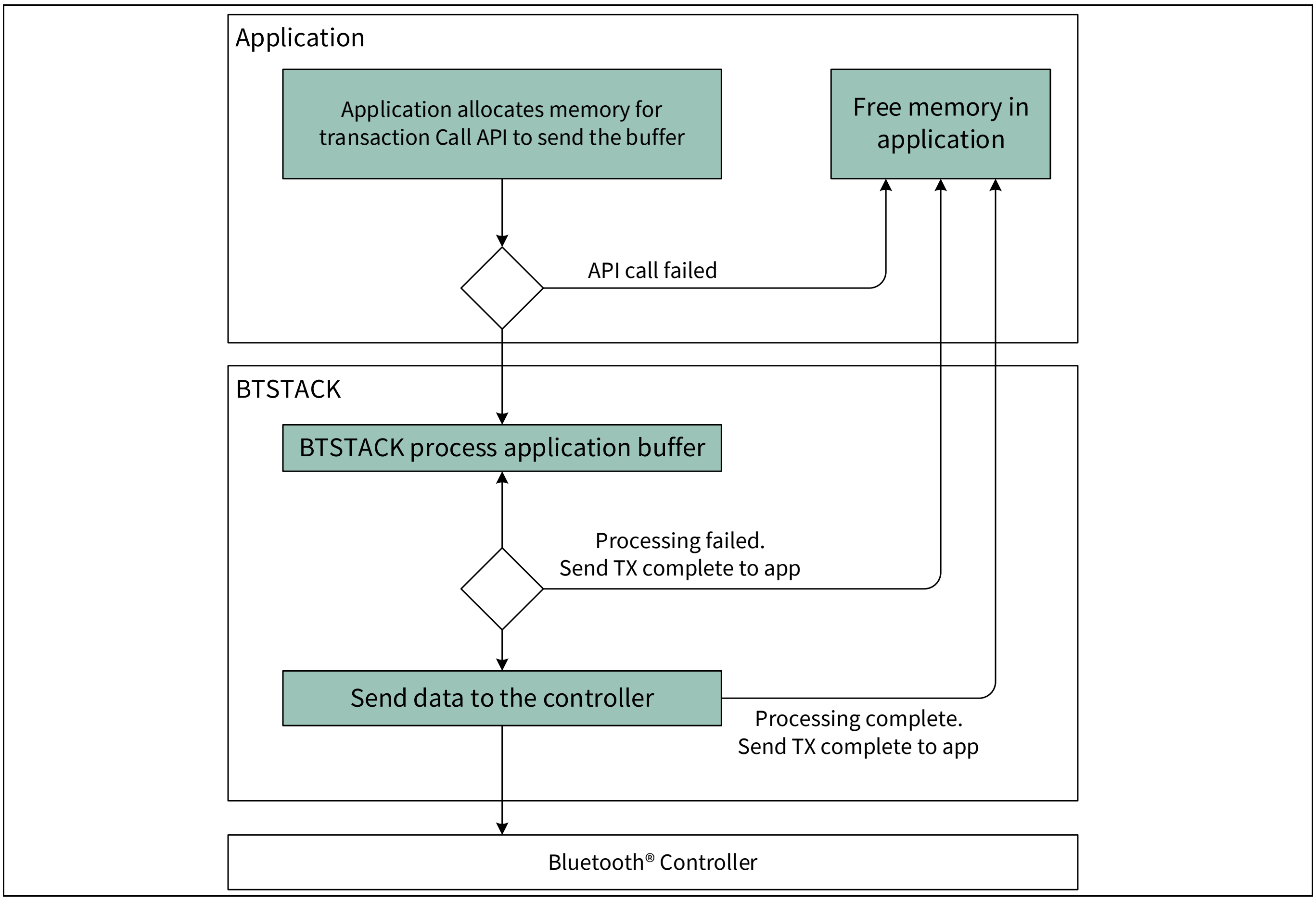

Figure 7 Downstream memory management

5.3 Upstream memory management

In the upstream path (BTSTACK to application):

1. HCI data segments called “Protocol Data Units” (PDUs) received from a peer device are reassembled into the link PDU buffer.
2. On completion of the reassembly, the packets are handed over to the registered protocol or profile.
3. At this stage, the packets could be either complete or considered fragments of a Service Data Unit (SDU), which will be further reassembled into the protocol or profile SDU buffer to get a complete packet.
4. The completed packet is sent up to the application using the registered data callback.
5. The application is expected to process the received data in the context of the callback, or copy it into an application buffer for deferred processing.
6. Upon returning from the callback, BTSTACK immediately reuses the memory buffer for the next incoming packet.

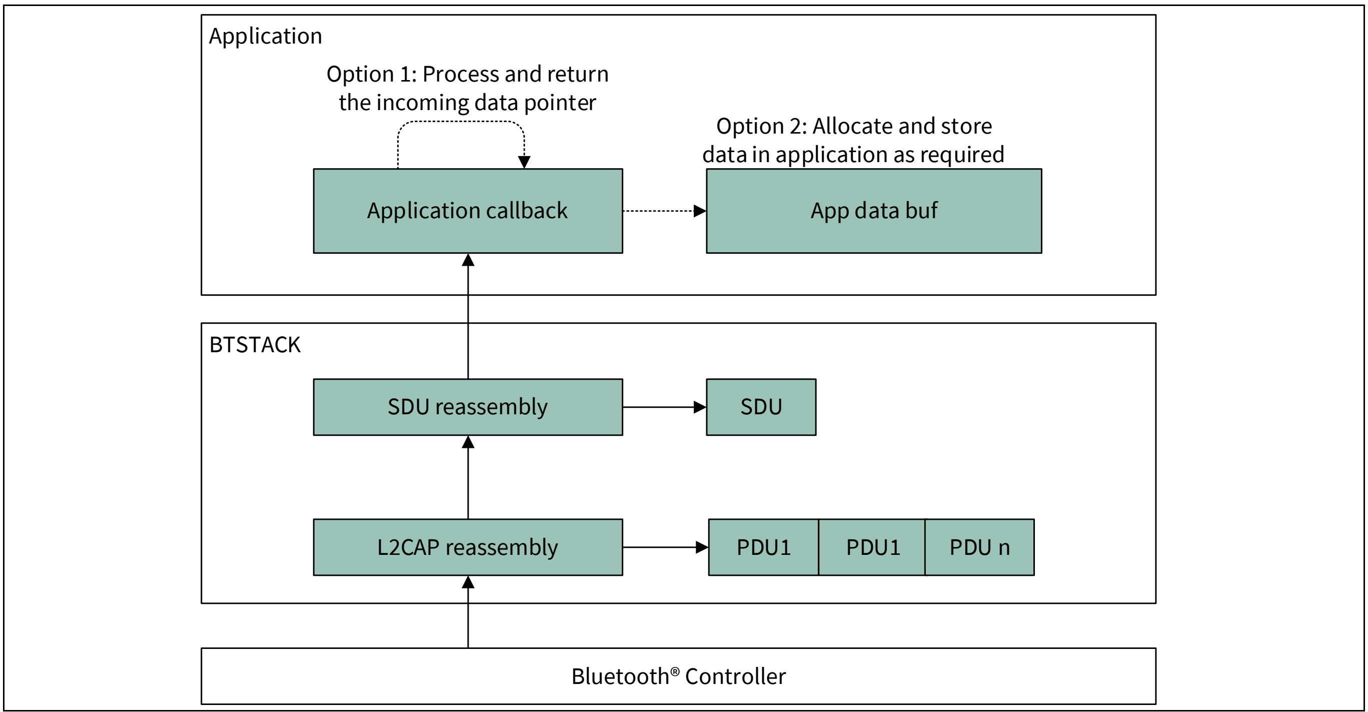

Figure 8 Upstream memory management

6 Tools

6.1 BTSpy

BTSpy is a trace utility that can be used in the AIROC™ Bluetooth® platforms to view protocol and generic trace messages from the Bluetooth® application. BTSpy can also generate Bluetooth® Snoop logs by saving the logs from Menu **Tools** > **File Logging Options** > **Generate snoop log file**. To view the snoop log, use a utility such as FrontLine Viewer.

For more information, see the BTSpy GitHub repository.

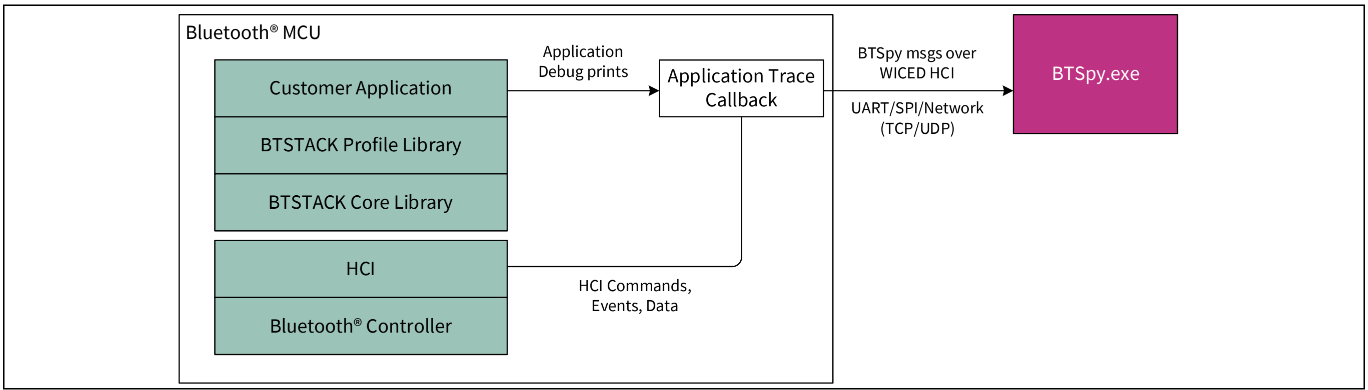

Figure 9 BTSpy

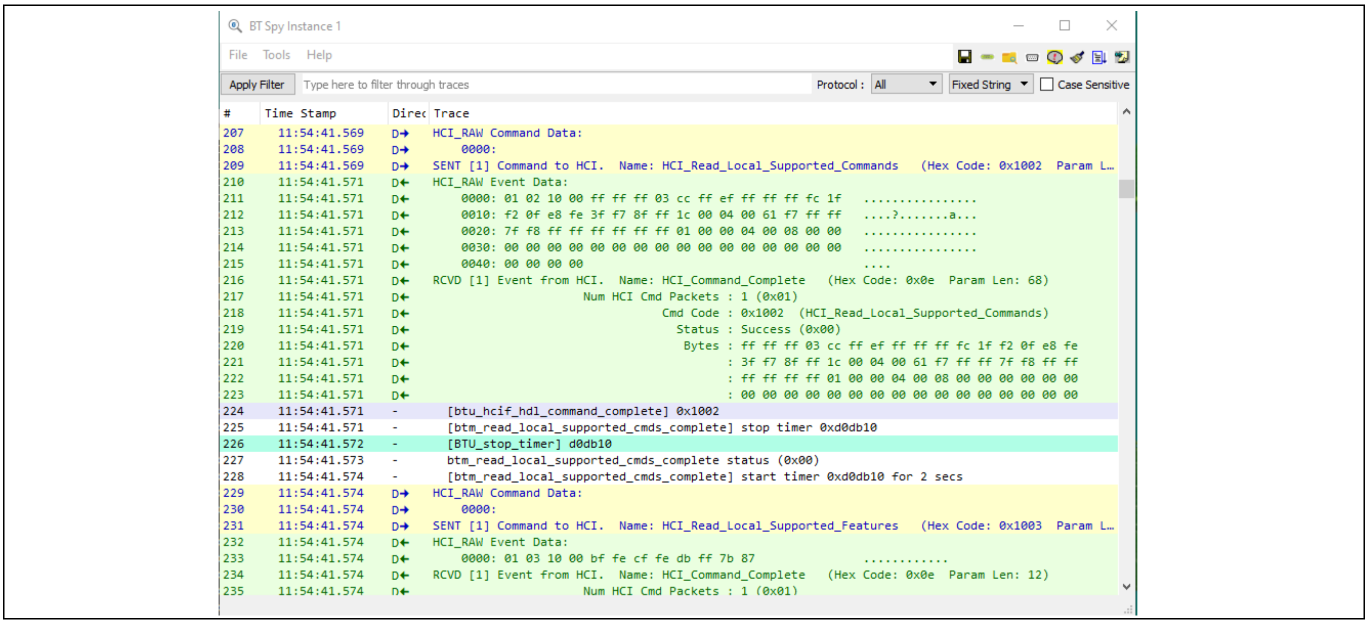

Figure 10 BTSpy sample capture

6.2 Bluetooth® Configurator

Bluetooth® Configurator is a stand-alone graphical tool included with the ModusToolbox™ software. Bluetooth® Configurator helps to generate code for Bluetooth® applications, including the device settings, GATT database, and SDP database. For more information, see the Bluetooth® Configurator guide.

References

[1] Bluetooth® SIG

[2] ModusToolbox™ software

[3] BTSTACK Library

[4] BTSpy GitHub repository

[5] Bluetooth® Configurator

[6] WICED HCI UART Control Protocol

[7] Android A2DP hardware Offload

[8] Bluetooth® Snoop logs

Revision history

Document revision	Date	Description of changes
**	2023-05-22	Initial release.
*A	2023-06-02	Deleted Confidential status.

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2023-06-02

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email: erratum@infineon.com

Document reference

002-37699 Rev. *A