Airoha IoT SDK Ultra Low Latency V3 Developer's Guide

Version: 1.0

Release date: 14 September 2023

Use of this document and any information contained therein is subject to the terms and conditions set forth in Exhibit 1. This document is subject to change without notice.

Document revision history

Revision	Date	Description
1.0	14 September 2023	Initial release

Table of contents

List of Tables

List of Figures

1. Introduction

Ultra-Low Latency version 3(ULL V3) is an Airoha proprietary technology to support less than 10ms downlink voice/audio latency for headset over Bluetooth LE with a well-matched Bluetooth-Dongle.

There are two roles in the ULL V3 profile:

- **ULL_Server** — a device that usually has a capability with USB-in/line-in/i2S-in Audio Sound and encodes PCM audio data as Airoha ULD format. It can relay the data to remote device via wireless communication. It provides the following functions:
    - 1-Tx (PC→Device) & 1-RX (Device→PC)
    - Firmware update via USB
    - Streaming start/stop state notification
- **ULL_Client** — a device that acts as the remote audio input and output for **ULL_Server**. It provides the following functions:
    - Headset
        - LE Connection with ULL_Server
        - Firmware update via air
        - 3.5mm line-in
        - Latency switch
        - Multi-link (Dongle with ULL_Server Feature + Smartphone's HFP)
        - USB Audio

Figure 1 show that how ULL_Server (dongle) connects to ULL_Client (headset) for transporting the audio streaming. The headset has only one link (link A). Link A is a bidirectional data link includes stereo downlink and mono uplink.

Figure 1. ULL roles and link

Ultra-Low Latency V3 (ULL V3) supports lower latency than Ultra-Low Latency V2 (ULL V2). The downlink speed of ULL V3 supports up to 400 Kbps. The uplink speed of ULL V3 supports 64 Kbps. Table 1 shows more information.

Table 1. Technical parameter support for ULL

Category	ULL V3	ULL V2
Latency	10 ms	< 20 ms
Codec	LC3plus and Airoha in-house	LC3plus and Airoha in-house
Downlink speed	400 kbps	172~304 kbps

Category	ULL V3	ULL V2
Uplink speed	64 kbps	64 kbps

Figure 2 show that the detail codec information of ULL V3.

	Sinks	DL^F	DL^B	DL^{CH}	DL Bit Rate	UL^F	UL^B	UL^{CH}	UL Bit Rate	Urgent(5ms) Share*4	Latency
ULL 1.0 Opus	2	48k	16b	2	256Kbps	16k	16b	1	64Kbps	20B/2.5ms	~23ms
ULL 2.0 Opus*3		48k	16b		320Kbps	16K(156x)*1 32K(158x/157x)			64Kbps	100B	~20ms
LC3plus 96K Hi-Res		96K*2	24b		172.8Kbps				64Kbps	100B	~20ms
48K HD					304Kbps				64Kbps	100B	~20ms
		48k			200Kbps				64Kbps	100B	~20ms
ULL 3.0 ULD*5	1	48K	24b	2	400kbps	32k	16b	1	64Kbps	40B/2ms	DL:~10ms UL:~45ms
Wireless MICs	-	-	-	-	-	48k	24b	2	200Kbps(4D2S)	100B	25ms
	-	-	-	-	-			3	300Kbps(3D2S)	100B	
	-	-	-	-	-			4	400Kbps(3D2S, 10ms)		35ms

*1: 65 support 16K UL only. If want to support 32K need EC and iGO support 32K and that may have RAM insufficient issue.
*2: 65 disable all of WWE (AMA/Gsound) to support LC3plus 96K DL because insufficient IRAM/DRAM.
*3: Only AB156X/AB157X support
*4: Urgent channel data is kind of ISO data, it share BW with audio packet, 100B is maximum throughput w/o audio packet retransmission.
*5: Only AB157X support

Figure 2. ULL v3 Supported Codec

This document guides you through:

- Support for Bluetooth with the library description and supported reference examples.
- Detailed descriptions of the ULL V3 profiles.
- Custom application development and debugging logs.

1.1. Profile Overview

Figure 3 shows the protocols and entities used in this profile.

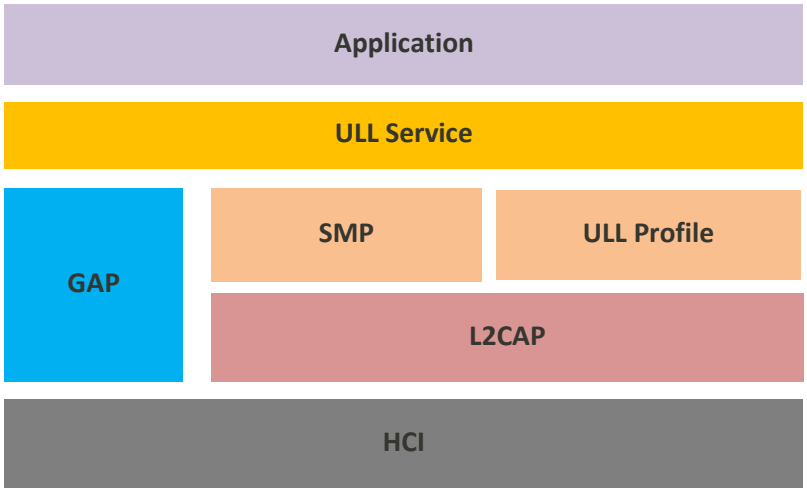

Figure 3. Protocol Model

The HCI, L2CAP, GAP, SMP, and ULL_Profile protocols are described in the

Airoha_IoT_SDK_Bluetooth_Developers_Guide.pdf document under the <SDK_root>/mcu/doc folder. The ULL Service is described in this document.

1.1.1. ULL Service

ULL Service is a service of ULL_Profile to manage the ULL LE connections, and the configuration of audio data and the state machine of streaming transport.

It involves multiple C source files (e.g., bt_ull_le_service.c, bt_ull_le_conn_service.c, bt_ull_le_audio_transmitter.c, bt_ull_le_audio_manager.c and bt_ull_le_utility.c) located in mcu/middleware/airoha/bt_ultra_low_latency folder.

The bt_ull_le_audio_transmitter.c file is only used to manage play/stop audio data for **ULL_Server**, while bt_ull_le_audio_manager.c file is only for **ULL_Client**.

1.2. Usage Scenario

ULL_Server is a device that supports USB-in/line-in/i2S-in Audio Sound capability. It encodes the Airoha ULD format and transmits to **ULL_Client** via Bluetooth LE technology.

ULL_Client can supports multilink connections (Dongle with ULL_Server Feature + Smartphone's HFP).

Headsets support single link only in ULL3.0 mode and multilink in ULL2.0 mode.

The two types of scenarios is shown in Figure 4.

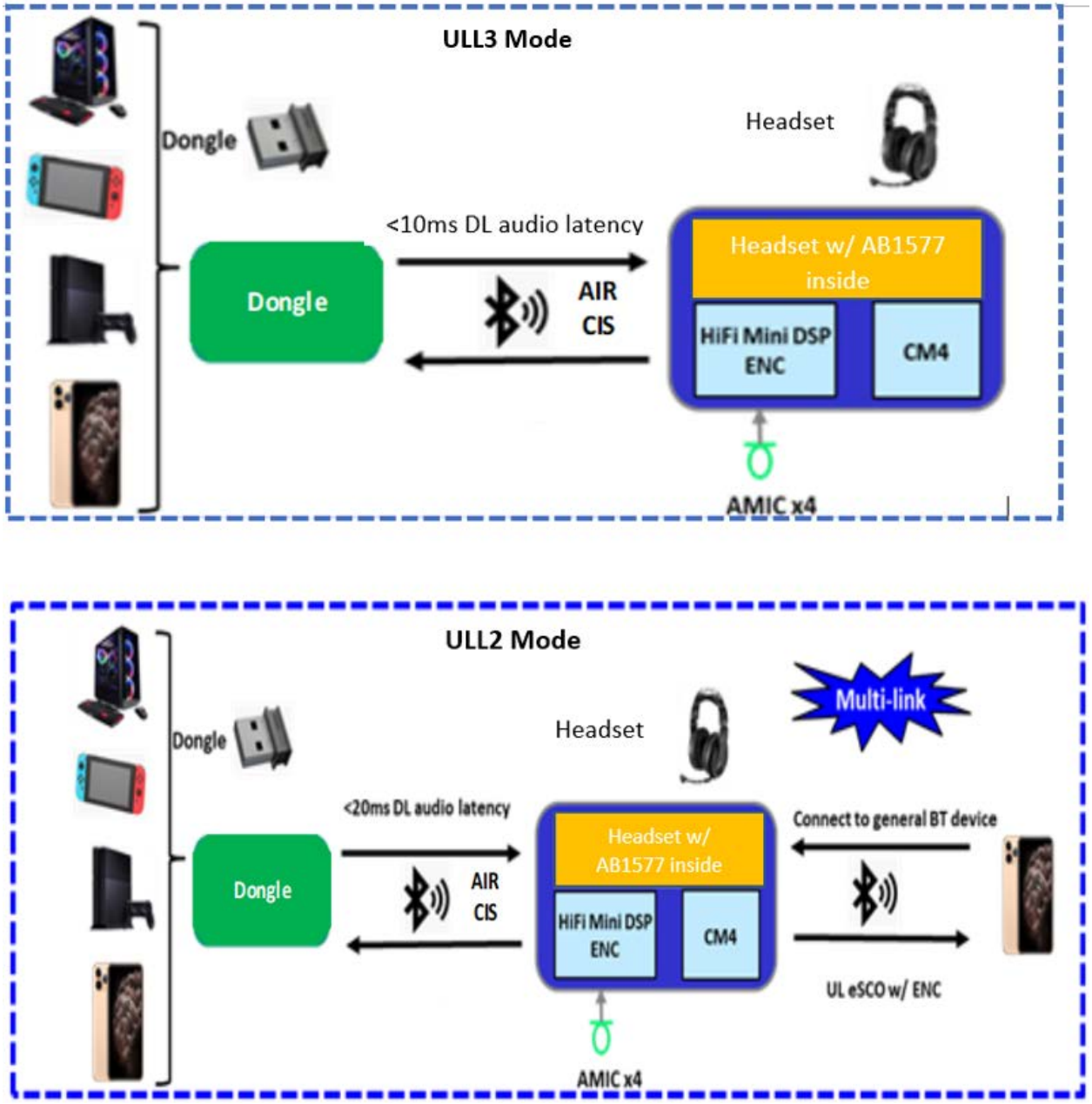

Figure 4. ULL usage scenario

1.3. Related SDK Library Requested

The ULL feature can only be run on Airoha IoT SDK for BT-Audio platform with the requested library files to interface the Bluetooth with C source and header files related to the platform, as shown in Table 2.

Table 2. Airoha IoT SDK library support for ULL

Module	Location	File Name	Function
Bluetooth	mcu/prebuilt/middleware/airoha/bluetooth/lib/	libbt.a	BR/EDR and Bluetooth LE stack library
		libbtdriver_[chip].a	Bluetooth driver library
		libbt_aws_mce.a	MCSync library, including MCSync implementation

Module	Location	File Name	Function
		libbt_ull.a	ULL library
		libpka_ull3_dongle.a	Bluetooth controller library for ULL_Server (with ULL v3 functions)
	mcu/prebuilt/middleware/airoha/le_audio/lib/	libpka_ull3_hs.a	Bluetooth controller library for ULL_Client (with ULL v3 functions)
	mcu/prebuilt/middleware/airoha/bluetooth/inc/	bt_platform.h	Interface for Bluetooth tasks
		bt_type.h	Common data types
		bt_system.h	Interface for the system, such as power on or off, memory initiation, and callback APIs for event handling
		bt_uuid.h	Interface for the UUID
		bt_codec.h	Interface for the codec
		bt_aws_mce.h	Interface for the MCSync
		bt_gap_le.h	Interface for the GAP LE
		bt_os_layer_api.h	Wrapper APIs for RTOS, memory, advanced encryption standard (AES), and rand
		bt_debug.h	Encapsulated debugging interface
		bt_hci_log.h	Encapsulated interface for the HCI logging
		bt_ull_le.h	Interface for the ULL profile
	/mcu/middleware/airoha/bt_ultra_low_latency/inc	bt_ull_service.h	Common API for ULL service
		bt_ull_le_service.h	Interface for ULL service

2. The ULL V3 Service

2.1. The ULL Message Sequence

The ULL V3 procedure can be established using the message sequence. The message sequence for each process is described below:

1) Connection establishment
2) Connection release
3) Set ULL Mode
4) Critical data transmit-receive
5) User data transmit-receive

2.1.1. Connection Establishment

Use the connection establishment operation to establish an LE connection between ULL Server and ULL Client.

Set Identity Resolving Key (SIRK) is associated with the Coordinated Set. All ULL clients that are part of the same Coordinated Set shall use the same SIRK. The SIRK is a 128-bit long random number. For example, earbuds have two devices named “agent” and “partner”. The agent and partner belong to a coordinated set so that they should have the same SIRK. It is also used to generate the RSI information that is included in the data of advertising.

On the ULL Client side, ‘**bt_ull_le_srv_set_device_info ()**’ is used to set SIRK before starting the advertising. User can reset SIRK by AT CMD ‘AT+LEULL=SIRK,SET,xx,xx,xx,xx,xx,xx,xx,xx,xx,xx,xx,xx,xx,xx,xx,xx\0d\0a’.

Also, ‘**bt_ull_le_srv_get_uuid ()**’ and ‘**bt_ull_le_srv_get_rsi ()**’ are used to combine advertising data before starting the advertising.

While, on the ULL Server side, the function ‘**bt_ull_le_srv_verify_rsi ()**’ is used to verify the RSI information using the same SIRK with the ULL Client devices.

For more details, please refer to
<SDK_root>/mcu/middleware/airoha/bt_ultra_low_latency/inc/bt_ull_le_service.h

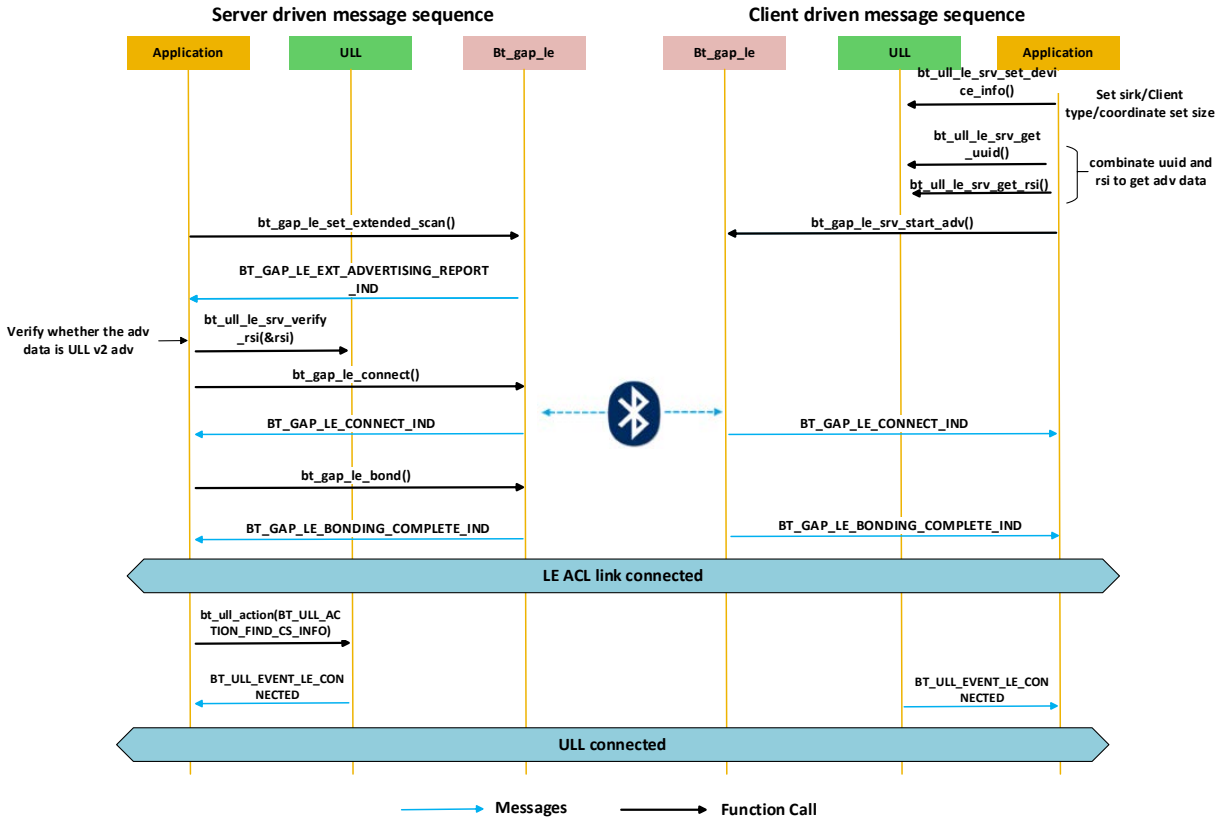

Figure 5. ULL connection establishment message sequence

2.1.2. Connection Release

The connection release procedure is used to disconnect the LE ACL link between the ULL server and ULL client. Both Server and Client can initiate the disconnection procedure.

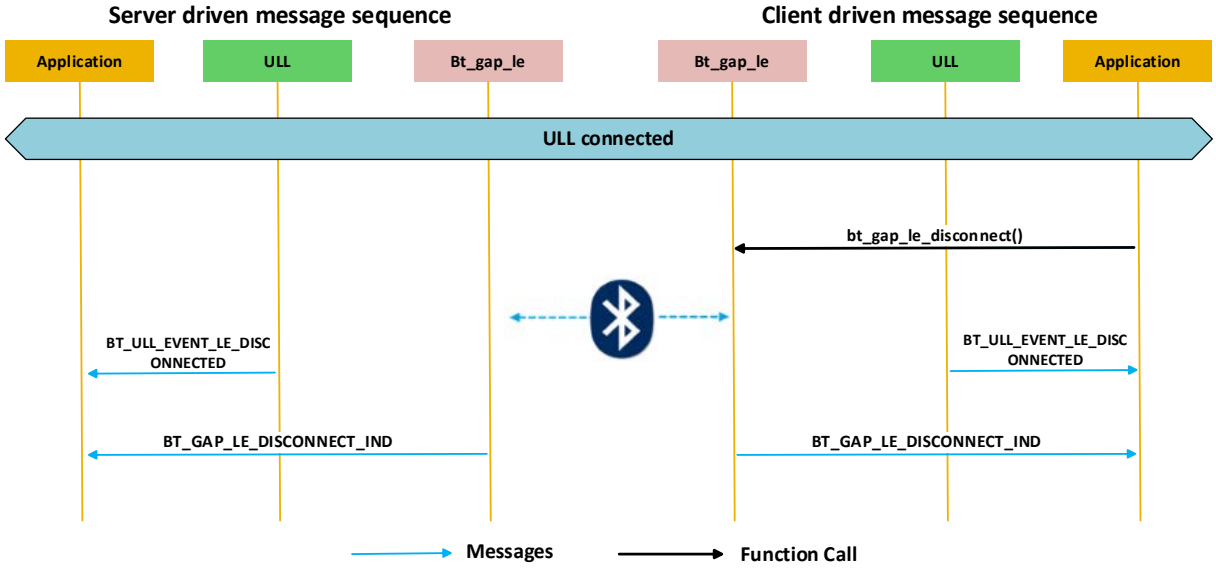

Figure 6. Disconnect ULL profile

2.1.3. Set ULL Mode

Headsets supports ULL mode switching between ULL3.0 and ULL2.0 via related AT commands. It changes ULL mode by updating ULL service information to notify the dongle. Dongle changes ULL mode according to the notification from headset.

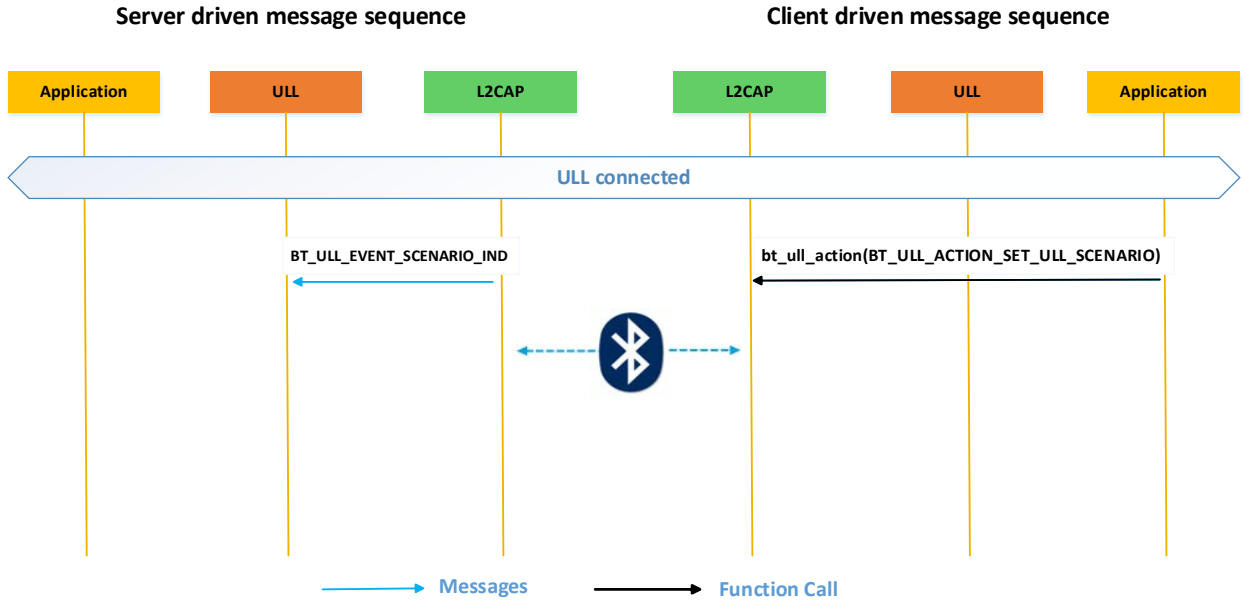

Figure 7. Set ULL Mode

2.1.4. Critical data transmit-receive

Use the critical data transmit-receive to exchange some unreliably continuous data (such as sensor data) with a flush timeout between Server and Client. The maximum length of critical data is **40** bytes. There is currently only support for Client to send critical data to Server when ULL is streaming. For more details, refer to <SDK_root>/mcu/middleware/airoha/bt_ultra_low_latency/inc/bt_ull_service.h.

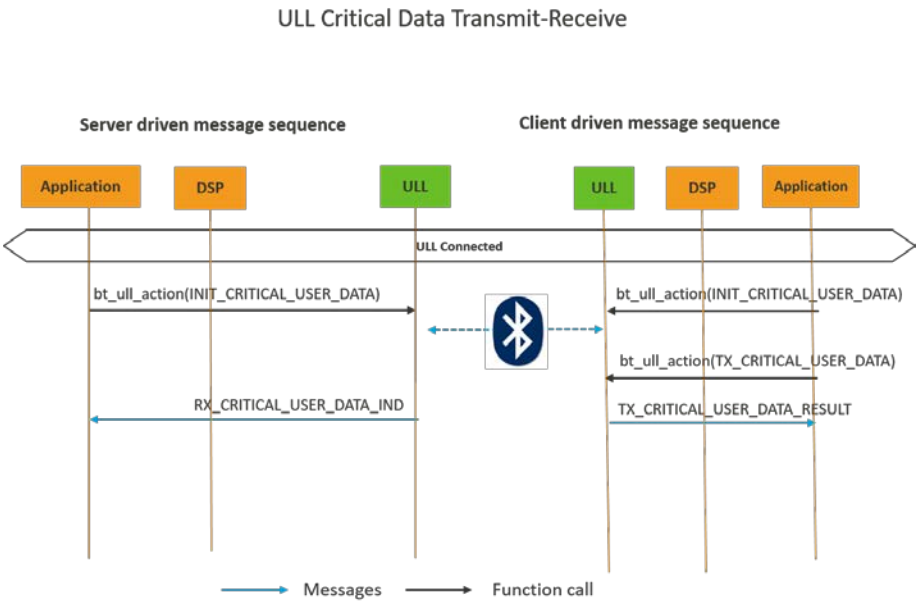

Figure 8. Critical data transmit-receive

2.1.5. User data transmit-receive

Use the user data transmit-receive to exchange user defined data between Server and Client. For more details, refer to <SDK_root>/mcu/middleware/airoha/bt_ultra_low_latency/inc/bt_ull_service.h.

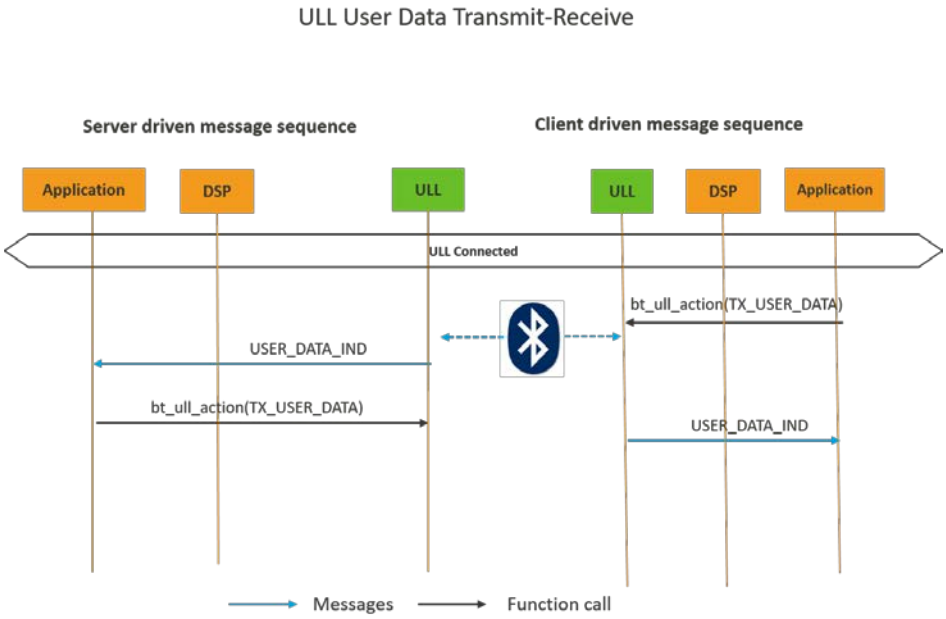

Figure 9. User data transmit-receive

2.2. Using the ULL APIs

This section describes how to use the ULL APIs for application development. The functionality of the ULL APIs is implemented in the module bt_ultra_low_latency, related APIs can be found in <SDK_root>/mcu/middleware/airoha/bt_ultra_low_latency/inc/bt_ull_le_service.h and <SDK_root>/mcu/middleware/airoha/bt_ultra_low_latency/inc/bt_ull_service.h, the other header files are used internally, and applications cannot use them at any time.

1) Call bt_ull_init() to start the ULL role during the initiation process in Dongle as Server or Headset/Earbuds as Client when the system powers on.

```
bt_ull_init(role, callback);
```

2) Call bt_ull_action() to control audio stream, e.g., ULL Server or ULL Client setting the volume.

```
bt_ull_volume_t volume_param;
volume_param.streaming.streaming_interface = BT_ULL_STREAMING_INTERFACE_SPEAKER;
volume_param.streaming.port = 0;
volume_param.action = BT_ULL_VOLUME_ACTION_SET_UP;
volume_param.channel =  BT_ULL_AUDIO_CHANNEL_DUAL;
volume_param.volume = 1;
bt_ull_action(BT_ULL_ACTION_SET_STREAMING_VOLUME,&volume_param,sizeof(volume_param));
```

3) Call bt_ull_action() to switch the ULL mode, e.g., ULL Client switch the ULL mode as ULL3.0.

```
bt_ull_le_scenario_t scenario_type = BT_ULL_LE_SCENARIO_ULLV3_0;
bt_ull_action(BT_ULL_ACTION_SET_ULL_SCENARIO, &scenario_type,sizeof(bt_ull_le_scenario_t));
```

4) Call bt_ull_get_streaming_info() to get the specified streaming information.

```
bt_ull_streaming_t streaming
streaming.streaming_interface = BT_ULL_STREAMING_INTERFACE_SPEAKER;
streaming.port = 0;
bt_ull_streaming_info_t info = {0}
bt_ull_get_streaming_info(streaming,&info);
```

5) Call bt_ull_lock_streaming() to lock or unlock the streaming. For example, the upper user can lock the streaming before the OTA procedure is started.

```
bt_ull_lock_streaming(true);
```

6) Call bt_ull_le_srv_set_device_info() to set the necessary device information of the ULL Client according the ULL client is a Headset or Earbuds device before the Bluetooth powers on. It is only initiated by the ULL Client.

```
bt_ull_le_device_info_t dev_info;
dev_info.client_type = BT_ULL_EARBUDS_CLIENT ;
dev_info.size = 2; /**< The size of ULL LE Coordinated set. */
dev_info.sirk = {0x00,0x01,0x02,....0x0F};
dev_info.group_device_addr = {0xC1,0xC2,....0xC6};//for earbuds, here are the 2 earbuds's device address.
bt_ull_le_srv_set_device_info(&dev_info);
```

7) Call bt_ull_le_srv_get_uuid() to get the UUID of ULL V2. The UUID is included in the advertising data. It is used to verify whether the device supports ULL V2.

```
bt_ull_le_uuid_t *uuid;
uuid  =  bt_ull_le_srv_get_uuid ();
```

8) Call bt_ull_le_srv_get_rsi() to calculate the Resolvable Set Identifier (RSI). The RSI is randomly generated by the SIRK. An RSI can be resolved if the corresponding SIRK is available by using the Resolvable Set Identifier resolution operation.

```
bt_ull_le_rsi_t rsi;
bt_ull_le_srv_get_rsi(&rsi);
```

9) Call bt_ull_le_srv_verify_rsi() to verify the RSI using the correct SIRK.

```
bt_ull_le_rsi_t rsi;
bt_ull_le_srv_verify_rsi(&rsi);
```

10) Call bt_ull_le_srv_get_role () to get the role of ULL service.

```
bt_ull_role_t role;
role =  bt_ull_le_srv_get_role();
```

11) Call bt_ull_le_srv_set_access_address () to set the vendor access address for transmission air interface packets.

```
bt_ull_le_set_adv_scan_access_addr_t access_addr = {0};
access_addr.acess_addr[0] = 0x6D;
access_addr.acess_addr[1] = 0xEB;
access_addr.acess_addr[2] = 0x98;
access_addr.acess_addr[3] = 0xE8;
bt_ull_le_srv_set_access_address(&access_addr);
```

12) Call bt_ull_le_srv_enable_adaptive_bitrate_mode () to enable or disable the adaptive bitrate mode.

```
bt_ull_le_adaptive_bitrate_params_t adaptive_bitrate_param;
adaptive_bitrate_param.enable = true;
adaptive_bitrate_param.crc_threshold = 9;
adaptive_bitrate_param.flush_timeout_threshold = 3;
adaptive_bitrate_param.report_interval = 100;
adaptive_bitrate_param.rx_timeout_threshold = 3;
bt_ull_le_srv_enable_adaptive_bitrate_mode(&adaptive_bitrate_param);
```

3. The UI behavior of ULL V3

3.1. Switch ULL Mode

The headset project on SDK3.9.0 supports the coexistence of the ULL3.0 and ULL2.0 features. However, there can be only one kind of connection at the same time. Therefore, we provide some ways to allow users to switch the headsets to a specific mode, i.e. ULL 3.0 mode or ULL2.0 mode.

The user can switch mode by customization. For example, AT CMD, pressing a button, or via the config tools. The SDK3.9.0 provides a way to select the mode using AT CMD on Headset Side.

1) Send AT CMD: AT+LEULL=VER,GET to get current ULL mode is ULL2.0 mode, i.e. ver=0, it indicates that the ULL mode is ULL2.0.

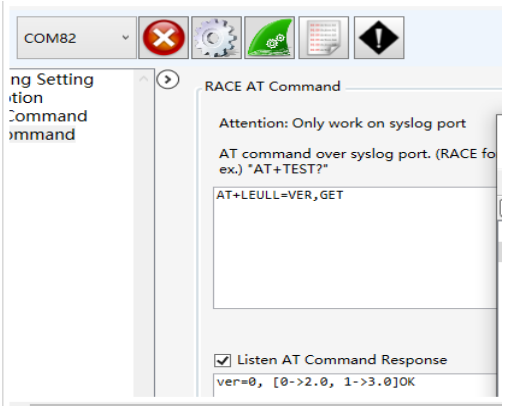

Figure 10. User data transmit-receive

2) Send AT CMD: AT+LEULL=VER,2.0 to switch the Headset to ULL2.0 mode

3) Send AT CMD: AT+LEULL=VER,3.0 to switch the Headset to ULL3.0 mode

3.2. ULL3.0 Mode and ULL2.0 Mode

3.2.1. ULL3.0 Mode

The ULL3.0 mode means the DUT can be connected to only one source device at a time, either the ULL Dongle or smartphone. The latency of the connection between the dongle and the DUT is 10ms. In this mode, the user can use a key to switch the connection between the dongle and the smartphone.

If Bluetooth powers on in this mode, the device reconnects to the last connected device.

3.2.2. ULL2.0 Mode

The behavior in this mode is the same as the design of ULL2.0.

If Bluetooth powers on in this mode, the device reconnects to the last connected device.

3.3. Wired USB Audio and Aux In

This feature is only supported on headset projects.
When the wired USB audio is enabled or Aux in is plugged in, the DUT disconnects the dongle. If the DUT is

currently connected to a smartphone, it tries to reconnect the A2DP profile.
When wired USB audio is disabled or Aux in is not plugged in, the DUT tries to reconnect with the dongle.

3.4. State machine diagram

The state machine diagram includes connection, disconnection, AUX or USB audio in or out, and using the key switch connection.

The multi-link mode has one more state than the single link mode, i.e. connected 2 SRC, which is shown in blue in this diagram.

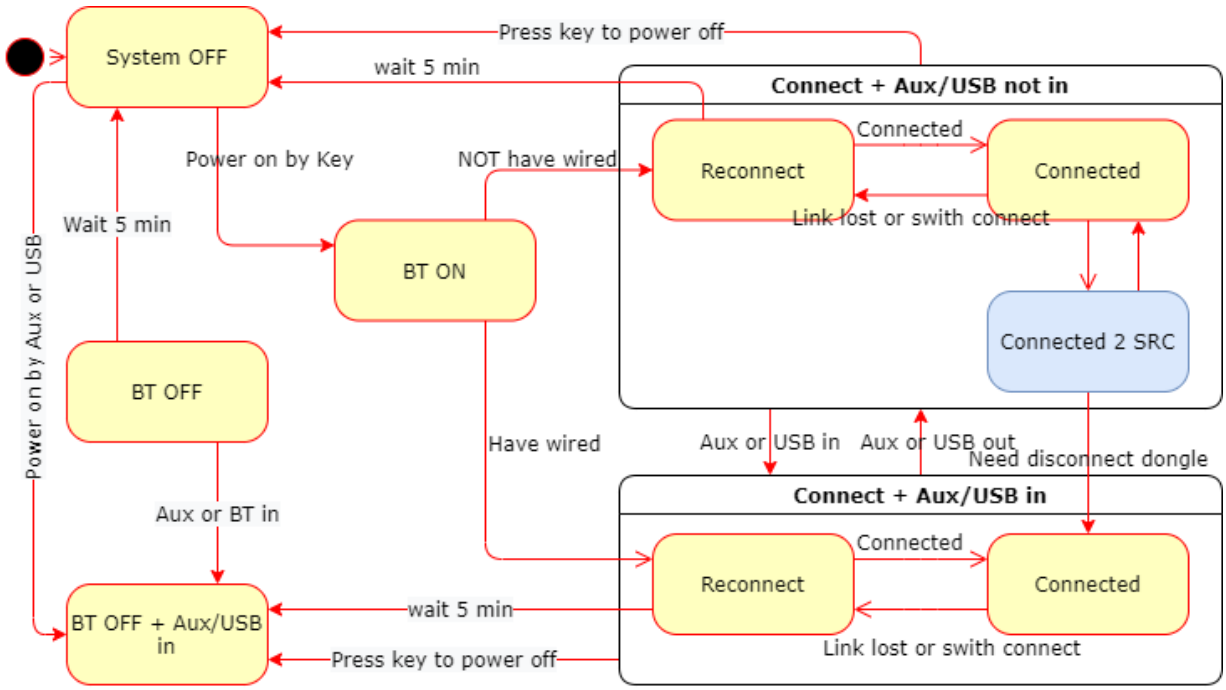

Figure 11. ULL state machine

3.5. ULL Profile Event

3.5.1. Events of ULL

The upper user can register an event callback function to listen to some ULL events from the ULL profile.

e.g., listens BT_ULL_EVENT_LE_CONNECTED and BT_ULL_EVENT_LE_DISCONNECTED to get the result of ULL connection. And, listens BT_ULL_EVENT_LE_STREAMING_START_IND, BT_ULL_EVENT_LE_STREAMING_STOP_IND to get the streaming status.

3.6. Key Actions

3.6.1. ULL Key Actions

There are some key actions that are specifically for the ULL project. They are:

- KEY_DISCOVERABLE, to trigger the headset to start the advertising.

- KEY_ULL_SWITCH_LINK_MODE, on the headset side, to trigger the switch for the link mode between single mode and multi-link mode. The single mode means only connection to either a smartphone or dongle at the same time. The multi-link mode means the DUT can connect to both the smartphone and dongle.
- KEY_ULL_RECONNECT, on the headset or earbuds side, to trigger the switch for the connection between the smartphone and dongle. It is only useful under single mode.

The key mapping table is defined in <project>\src\boards\<Your board>\customerized_key_config.c; Customer can change the table to define the preferred table.

Customer can refer to app_ull_idle_activity.c to review how to process the key events.

3.6.2. Audio Key Actions

Currently, the code uses a rotary key to change the mix ratio and side tone gain. Customer can review the code and implement the feature by the key event.

- Side tone volume:

  Use the KEY_AUDIO_SIDE_TONE_VOLUME_UP and KEY_AUDIO_SIDE_TONE_VOLUME_DOWN to increase or decrease the side tone volume. The minimum value is defined as ULL_SIDE_TOME_VOLUME_MIN_LEVEL; The maximum value is defined as ULL_SIDE_TOME_VOLUME_MAX_LEVEL; The increasing or decreasing value when the user slides the rotary one step is defined as ULL_SIDE_TOME_CHANGE_LEVEL_PRE_STEP.

There is support for pressing a key to mute the microphone. The key action is KEY_MUTE_MIC.

3.6.3. Media Key Actions

Headset or earbuds can control the PC media. The PC media can be ULL connection audio or wired USB audio. The supported actions are KEY_AVRCP_PLAY, KEY_AVRCP_PAUSE, KEY_AVRCP_FORWARD and KEY_AVRCP_BACKWARD. If the headset or earbuds have connected with one smartphone and one PC, and both the smartphone and PC are not playing, the action occurs on the smartphone. If the PC is playing and the smartphone is not playing or is disconnected, the action occurs on the PC. The processing code for controlling the smartphone media or ULL media is in app_music. The processing code for controlling USB audio media is in app_usb_audio which is only supported by the headset.

3.7. FOTA

When doing FOTA with a smartphone, APP calls bt_cm_write_scan_mode() to disable the page scan.

Exhibit 1 Terms and Conditions

Your access to and use of this document and the information contained herein (collectively this "Document") is subject to your (including the corporation or other legal entity you represent, collectively "You") acceptance of the terms and conditions set forth below ("T&C"). By using, accessing or downloading this Document, You are accepting the T&C and agree to be bound by the T&C. If You don't agree to the T&C, You may not use this Document and shall immediately destroy any copy thereof.

This Document contains information that is confidential and proprietary to Airoha Technology Corp. and/or its affiliates (collectively "Airoha") or its licensors and is provided solely for Your internal use with Airoha's chipset(s) described in this Document and shall not be used for any other purposes (including but not limited to identifying or providing evidence to support any potential patent infringement claim against Airoha or any of Airoha's suppliers and/or direct or indirect customers). Unauthorized use or disclosure of the information contained herein is prohibited. You agree to indemnify Airoha for any loss or damages suffered by Airoha for Your unauthorized use or disclosure of this Document, in whole or in part.

Airoha and its licensors retain titles and all ownership rights in and to this Document and no license (express or implied, by estoppels or otherwise) to any intellectual propriety rights is granted hereunder. This Document is subject to change without further notification. Airoha does not assume any responsibility arising out of or in connection with any use of, or reliance on, this Document, and specifically disclaims any and all liability, including, without limitation, consequential or incidental damages.

THIS DOCUMENT AND ANY OTHER MATERIALS OR TECHNICAL SUPPORT PROVIDED BY AIROHA IN CONNECTION WITH THIS DOCUMENT, IF ANY, ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE. AIROHA SPECIFICALLY DISCLAIMS ALL WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, COMPLETENESS OR ACCURACY AND ALL WARRANTIES ARISING OUT OF TRADE USAGE OR OUT OF A COURSE OF DEALING OR COURSE OF PERFORMANCE. AIROHA SHALL NOT BE RESPONSIBLE FOR ANY AIROHA DELIVERABLES MADE TO MEET YOUR SPECIFICATIONS OR TO CONFORM TO A PARTICULAR STANDARD OR OPEN FORUM.

Without limiting the generality of the foregoing, Airoha makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Airoha assume any liability arising out of the application or use of any product, circuit or software. You agree that You are solely responsible for the designing, validating and testing Your product incorporating Airoha's product and ensure such product meets applicable standards and any safety, security or other requirements.

The above T&C and all acts in connection with the T&C or this Document shall be governed, construed and interpreted in accordance with the laws of Taiwan, without giving effect to the principles of conflicts of law.