

1 Introduction

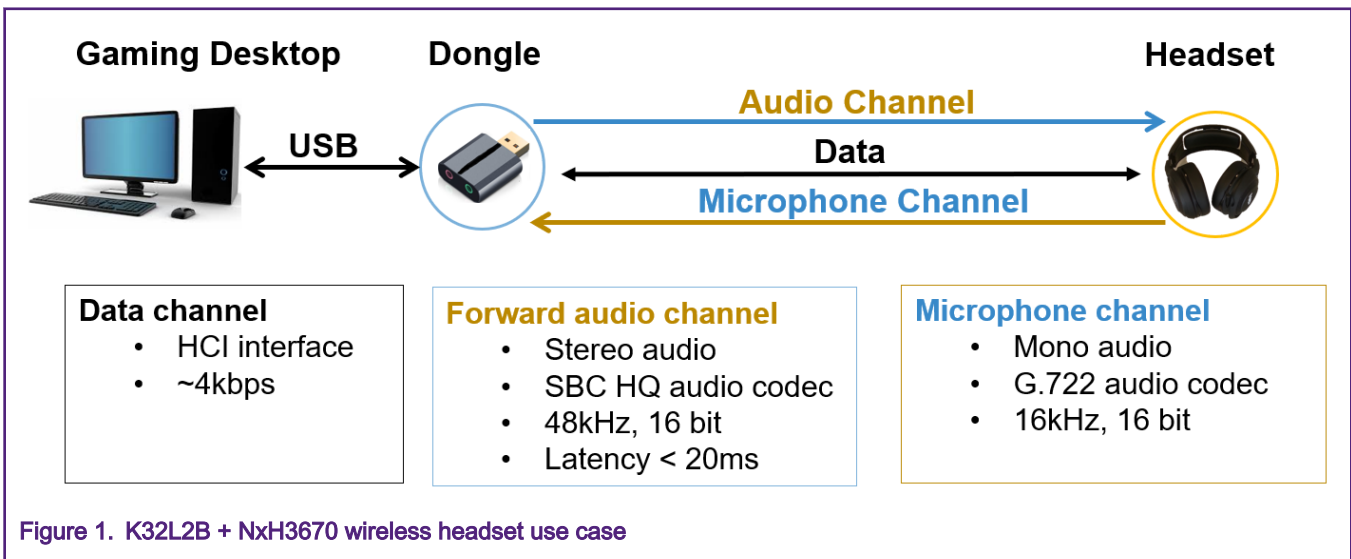
1.1 Overview

This document describes the hardware design of **K32L2B Bluetooth Low Energy (Bluetooth LE) Audio System** and software architecture (top-level design) of Host Controller (K32L2B). This document is provided for those who intend to have a systematic view of **K32L2B Bluetooth LE Audio System**. They can also refer to relevant ANs if need more introduction about Dongle (*K32L2B Dongle*), Headset (*K32L2B Headset*) and OTA (*K32L2B OTA*).

- [Hardware introduction](#) introduces the hardware composition of K32L2B Dongle and Headset, as well as the connection diagram.
- [Software introduction](#) introduces audio data path and software framework.

1.2 Summary

This document provides necessary information on how to get started on the **K32L2B Bluetooth LE Audio System** based on FRDM-K32L2B board and NxH3670 SDK boards.



The system consists of a Dongle and a Headset, using K32L2B as Host Controller.

- **Dongle:** The Dongle has a USB interface that connects to PC. Dongle is responsible for setting up a wireless audio link with Headset.
- **Headset:** The Headset has a speaker, a microphone and some User Interface (UI) components, such as, buttons, sliders, rotary switches and LED. Headset is responsible for receiving audio data sent from Dongle and sending the recorded audio to Dongle.

1.3 Reference documents

Table 1. References

Reference	Definition
<i>K32L2B Dongle</i>	K32L2B USB Dongle with NXH3670
<i>K32L2B Headset</i>	K32L2B Headset with NXH3670
<i>K32L2B OTA</i>	K32L2B Bluetooth LE Audio System OTA operation steps
<i>K32L2B Emulating the I²S Bus</i>	Emulating the I ² S bus master with the FlexIO module

2 Hardware introduction

2.1 System overview

Figure 2 shows the simplified system overview.

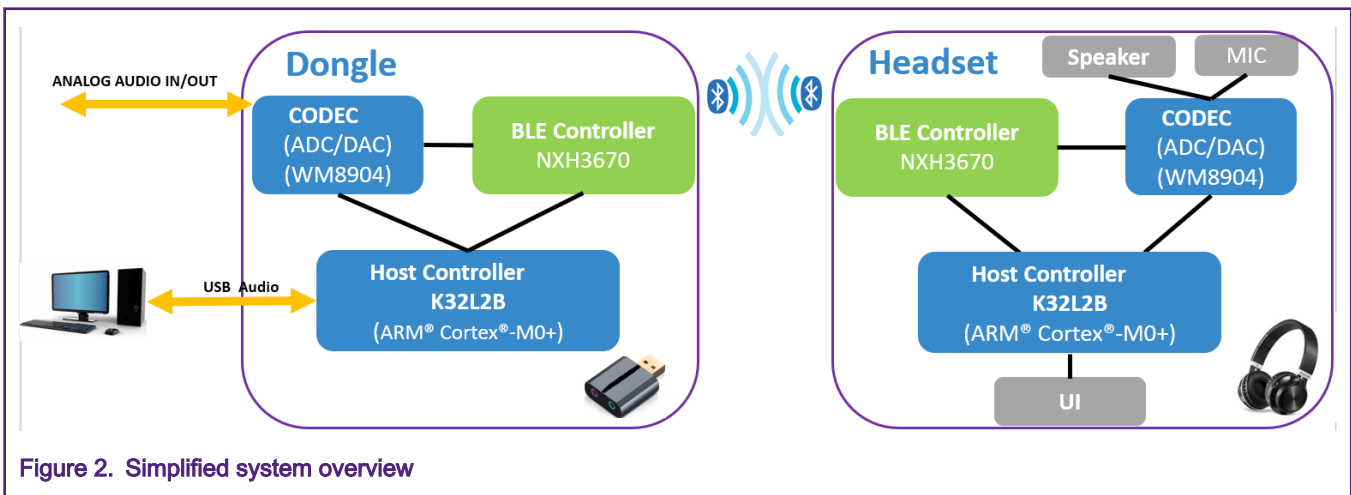


Figure 2. Simplified system overview

As shown in Figure 2, the audio transfer process include the following steps:

1. The NXH3670 boots up, starts and then communicates with K32L2B through the SPI interface.
2. Assuming NXH3670 can work well after Step 1, we use the USB interface to transfer audio stream from PC to host controller. The 48 KHz USB audio is converted to an I²S signal, and then transmitted to NXH3670 of Dongle through the I²S master emulated by FlexIO (*K32L2B Emulating the I²S Bus*).
3. The audio stream can be transmitted to NXH3670 of Headset automatically. Users can hear voices with their headsets.

The current **K32L2B Bluetooth LE Audio System** includes FRDM-K32L2B board and NxH3670 SDK boards (KL27 Dongle and Headset board to provide the basic Audio function respectively). This platform can:

- Send audio stream from PC to Headset.
- Receive control signal and recorded audio from Headset to PC.
- Update new firmware through Over The Air (OTA).

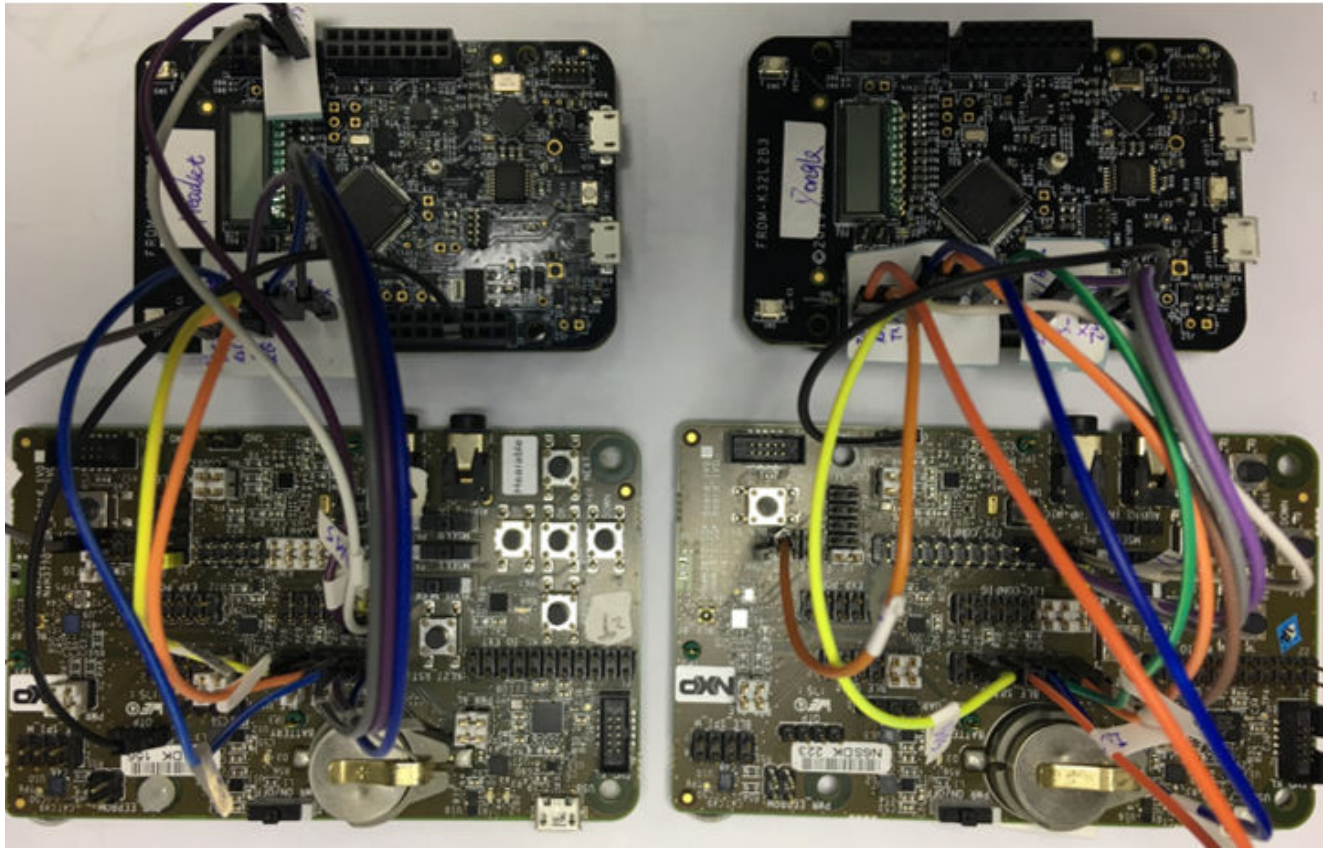


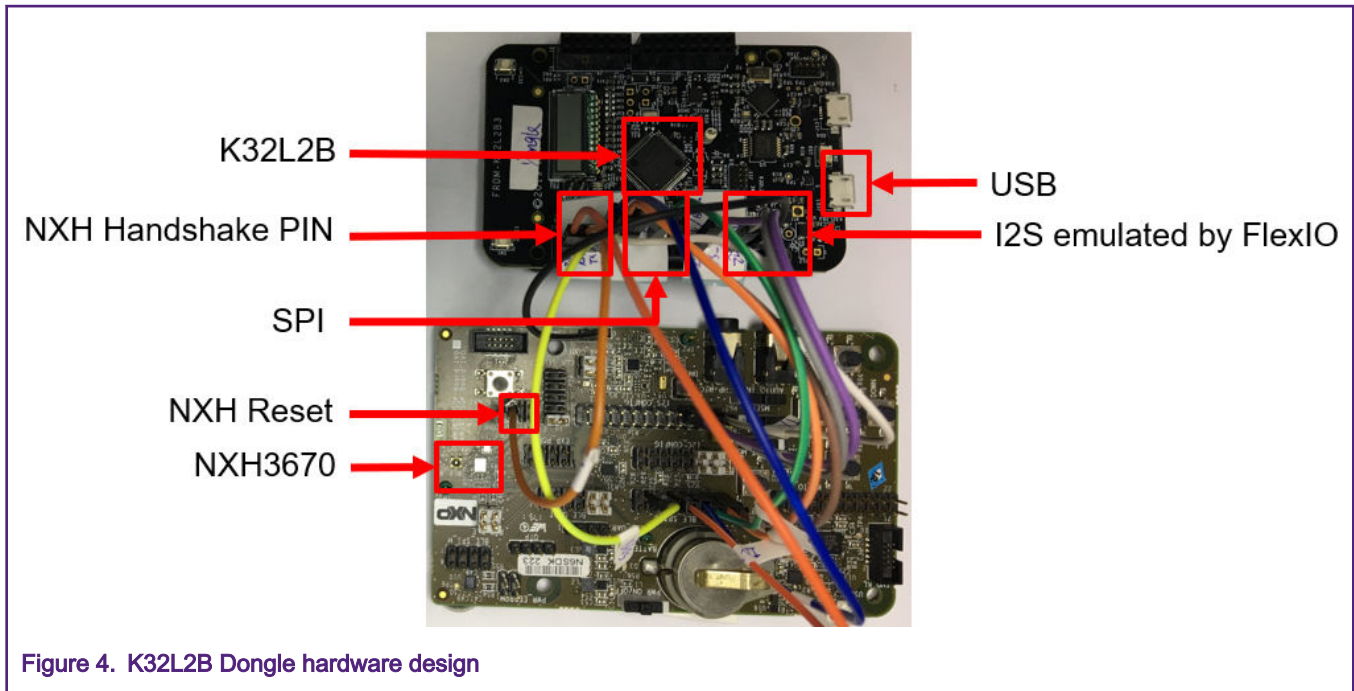
Figure 3. K32L2B Bluetooth LE audio system hardware

As shown in [Figure 3](#),

- In the Dongle part, the host controller is K32L2B and the Bluetooth LE device is NXH3670 on KL27 Dongle board. K32L2B configures and communicates with NXH3670 through the SPI interface. K32L2B transfers audio data to NXH3670 through the I²S bus emulated by FlexIO.
- In the Headset part, the host controller is K32L2B and the Bluetooth LE device is NXH3670 on KL27 Headset board. K32L2B configures and communicates with NXH3670 through the SPI interface and configures CODEC using the I²C interface.

2.2 K32L2B Dongle

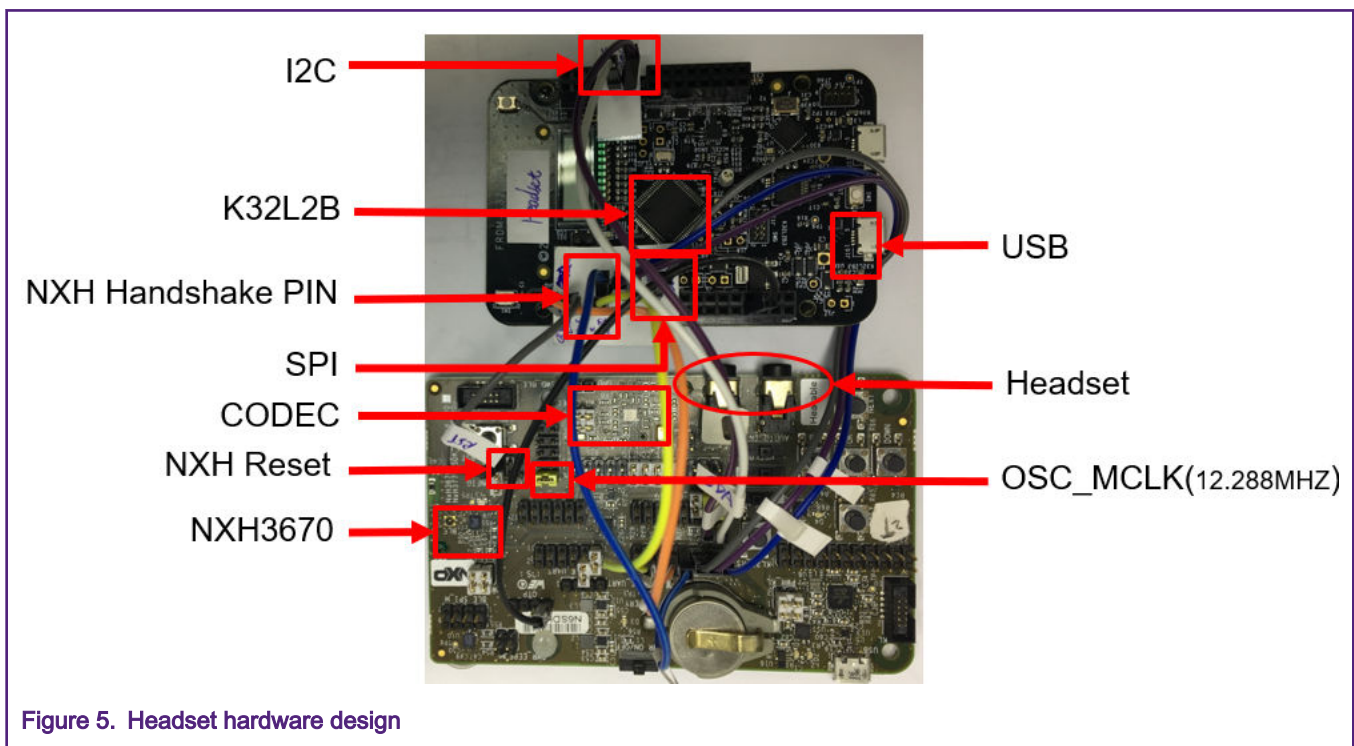
This section describes the current hardware design of K32L2B Dongle based on FRDM-K32L2B and KL27 Dongle board. [Figure 4](#) shows the components and interfaces.



User can use a USB cable to connect J13 (FRDM-K32L2B) with PC to power or download firmware.

2.3 Headset

This section describes the current hardware design of K32L2B Headset based on FRDM-K32L2B and KL27 Headset board. [Figure 5](#) shows the components and interfaces.



NOTE

- In the Headset design, we use I²C interface to configure CODEC.
- The NXH3670 communicates with CODEC using I²S (as shown in Figure 5, the attached jumpers of 9-10, 11-12, and 13-14 indicate that they can transfer data directly without the extra operation of K32L2B).
- User must make sure the existence of jumper 9-10 (J10 CLK_SELECT, the yellow jumper **OSC_MCLK (12.288 MHz)** in Figure 5), as it is used to obtain 12.288 MHz frequency and then can provide 24.576 MHz frequency to I²S_MCLK.

3 Software introduction

3.1 Audio path

The **Bluetooth LE Audio System** consists of two channels:

1. The forward-channel transmits the audio from the PC to the Headset.
2. The backward-channel transmits the microphone signal from the Headset to the PC.

3.1.1 Forward audio channel

- The forward-channel is a stereo-channel.
- The RF transports the forward-channel as 16-bit samples @ 48 KHz sample-rate.
- The I²S bus emulated by FlexIO and USB signals uses 48 KHz sample-rate.

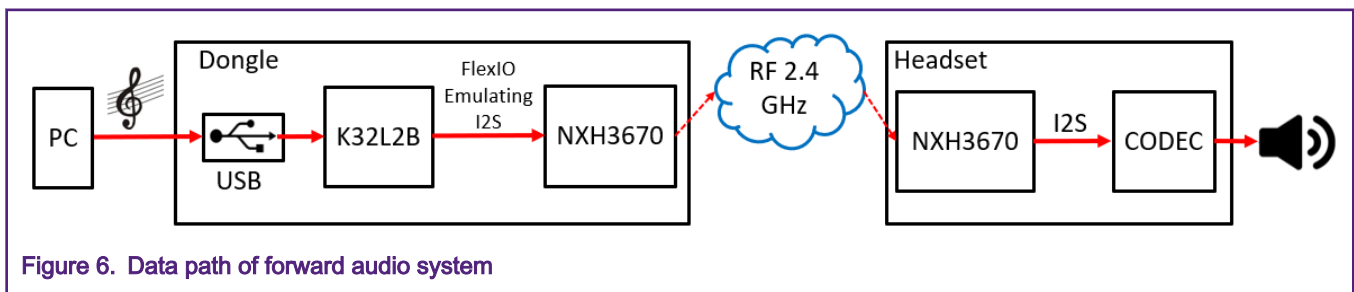


Figure 6. Data path of forward audio system

3.1.2 Backward audio channel

- The backward-channel is a mono-channel, only the left channel used.
- The RF transports the backward channel as 16-bit samples @ 16 KHz sample-rate.
- The I²S bus emulated by FlexIO and USB signals uses 48 KHz sample-rate.

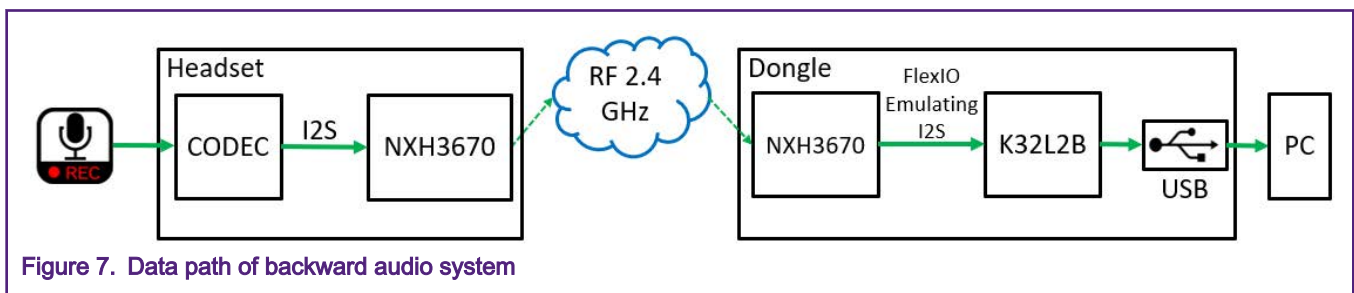


Figure 7. Data path of backward audio system

3.2 Application framework

The Application framework defines the software architecture of the reference application. The focus is modularity, code re-use and software maintainability.

1. The top layer is **Application Layer**, which is strictly application specific.
2. The layer below is called **Services Layer**.
3. Underneath the **Service Layer** is **Driver Layer** which controls the hardware.
4. Board Support Package (BSP) contains the board-specific software, such as, hardware initialization, GPIO-pin configuration, clock settings, etc.

Figure 8 shows the entire application architecture.

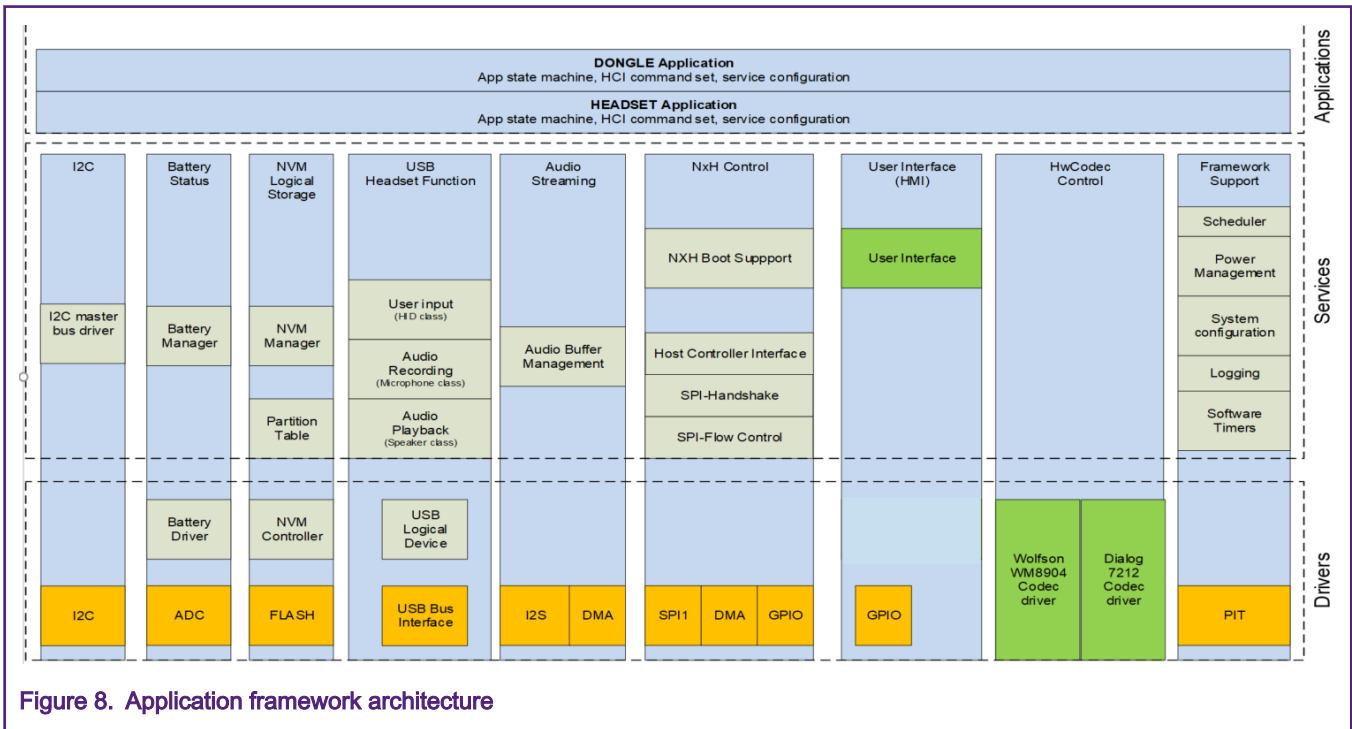


Figure 8. Application framework architecture

Users can design their own application or service on need. This document gives a brief introduction to the state machine that will be used to control other services and the application.

Users can decide what service to perform in a state as they wish. For example,

- The state of an uninitialized USB-service is uninitialized, and then the state will change from **Uninitialized** to **Stopped** by calling API `initialize (&cfg)` and executing successfully.

Figure 9 shows the mandatory API and corresponding state machine.

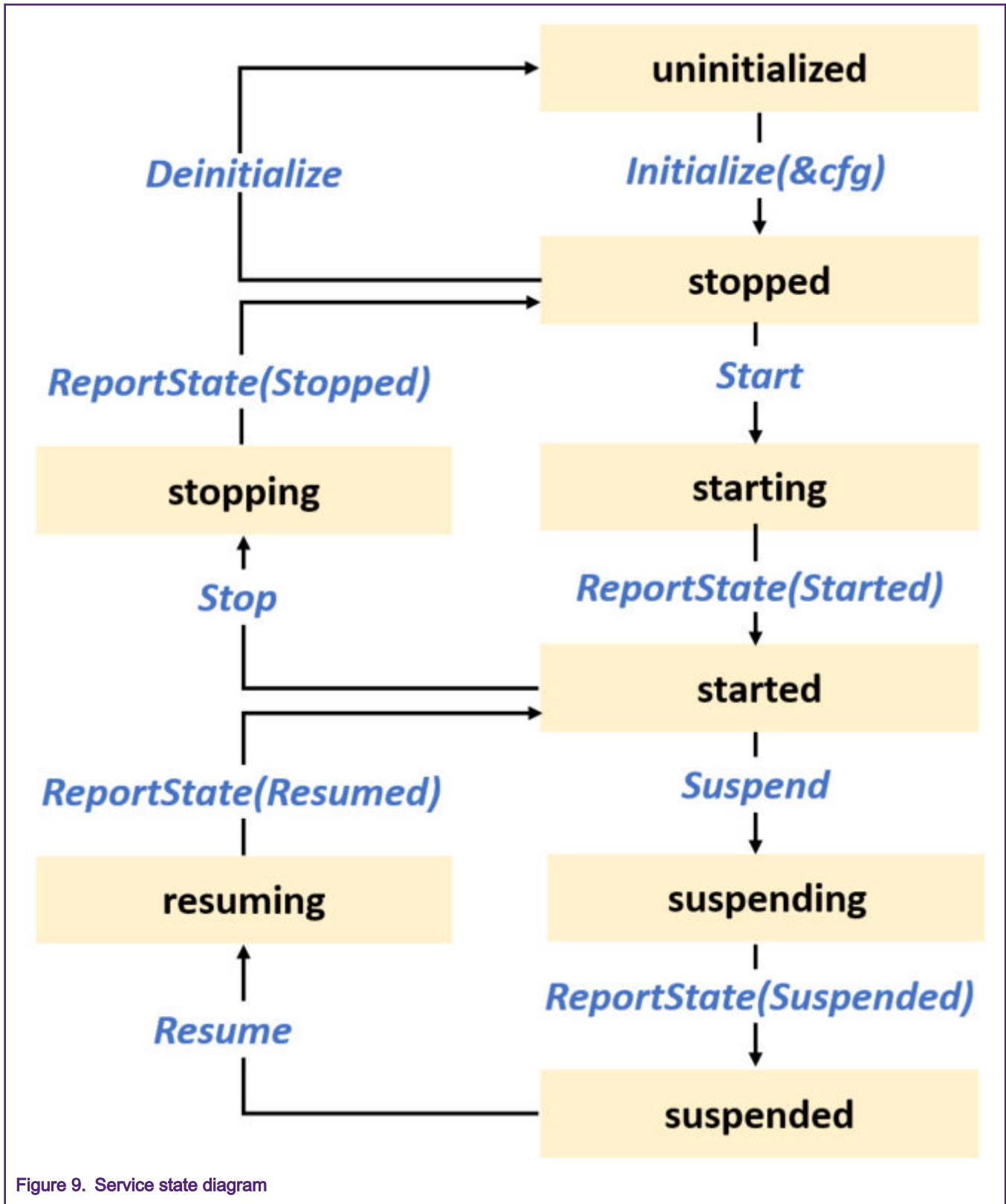


Figure 9. Service state diagram

For example, the state can jump from **Starting** to **Started** by calling `ReportState (Started)`.

```
FRAMEWORK_ServiceReportState (&g_XXXServiceApi, kSTATE_Started);
```

3.3 Firmware development

3.3.1 Setting up the environment

This section introduces how to set up the MDK environment and the materials required in each step are described in [Table 2](#).

Table 2. Materials required in Firmware development

List	Description
PC	Host device connected to the development board
Debugger	<ul style="list-style-type: none"> • Default CMSIS-DAP firmware in the debugger onboard. • Replace the default CMSIS-DAP firmware with J-LINK firmware if user want to use <code>JLink.exe</code> to download Bin file without IDE.
IDE	MDK (V5.26.2.0)
Demos	<p><code>K32L2B+NxH3670.zip</code> for Gaming use case, including:</p> <ul style="list-style-type: none"> • Bin files that can be download through <code>JLink.exe</code>. • Debug version of demo that can be used to re-developed.

3.3.2 Software based on MDK

This document ports five demos:

- Dongle
- K32L2B_Headset
- K32L2B_OTA_Dongle
- K32L2B_OTA_Headset
- K32L2B_SSB

[Figure 10](#) and [Figure 11](#) list two demos as references for users if they want to port the demo to other boards.

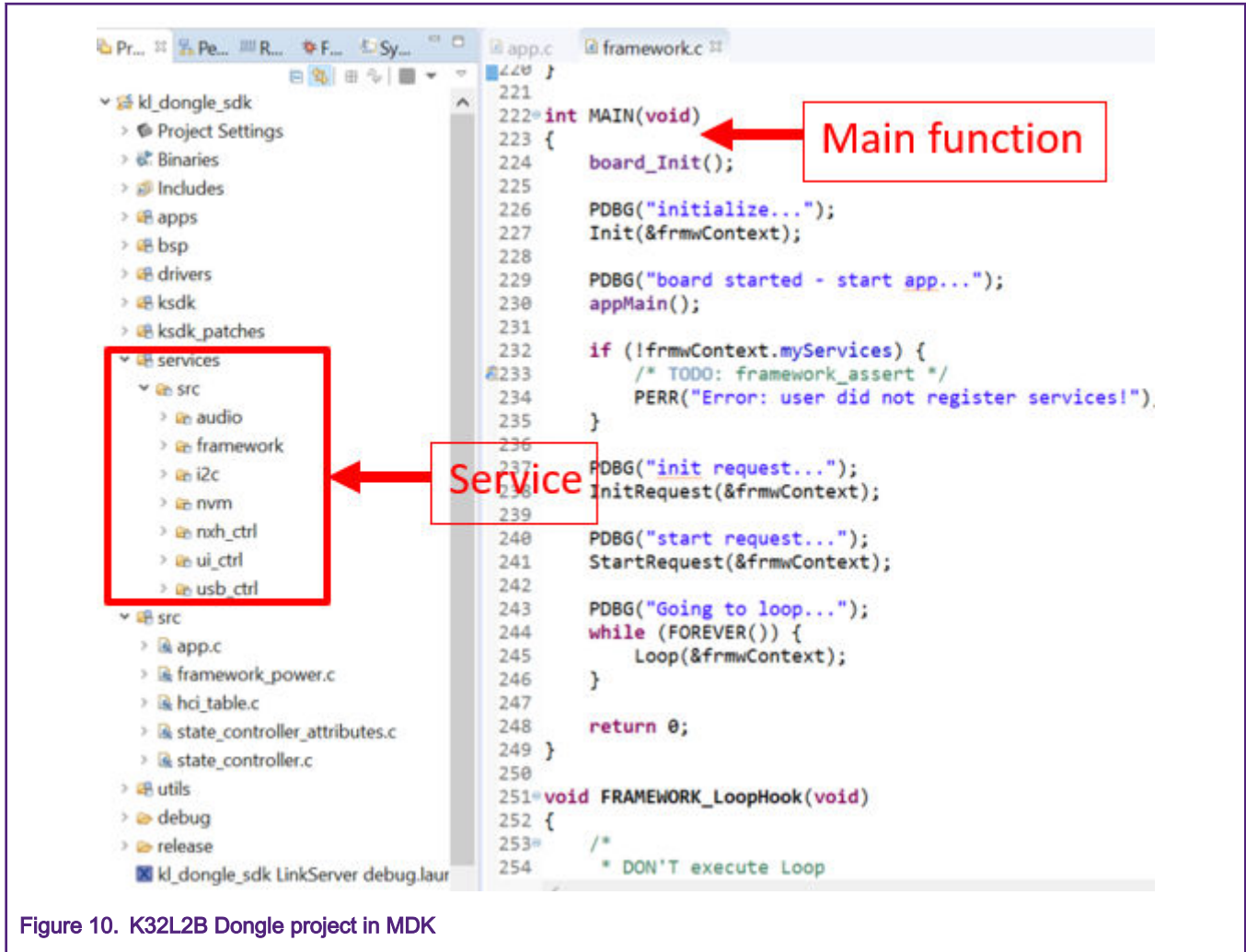


Figure 10. K32L2B Dongle project in MDK

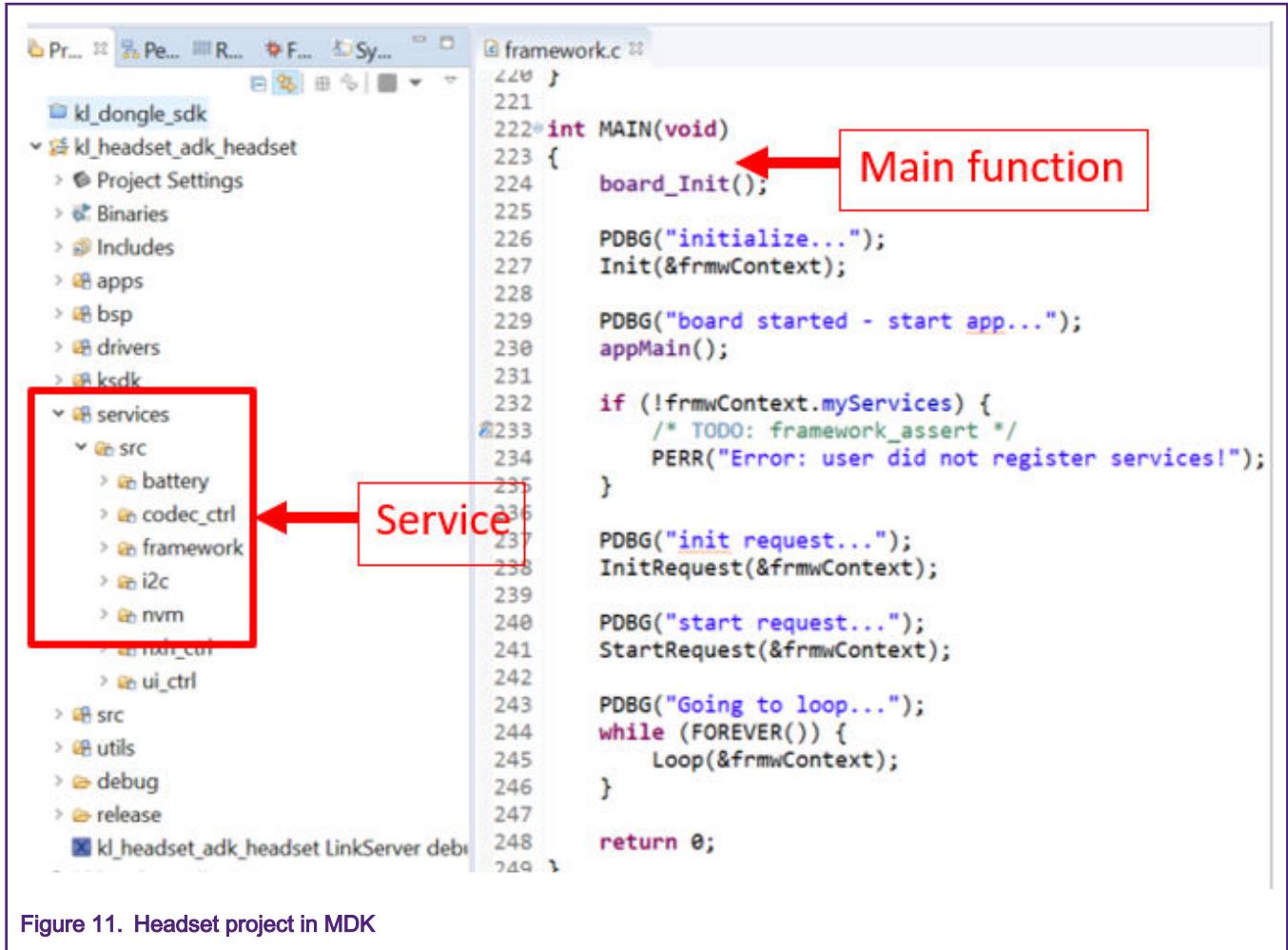


Figure 11. Headset project in MDK

If users want to use JLink.exe to download Bin file, make sure they have J-LINK firmware in Debugger onboard (MDK can be used for checking), as shown in Figure 12.

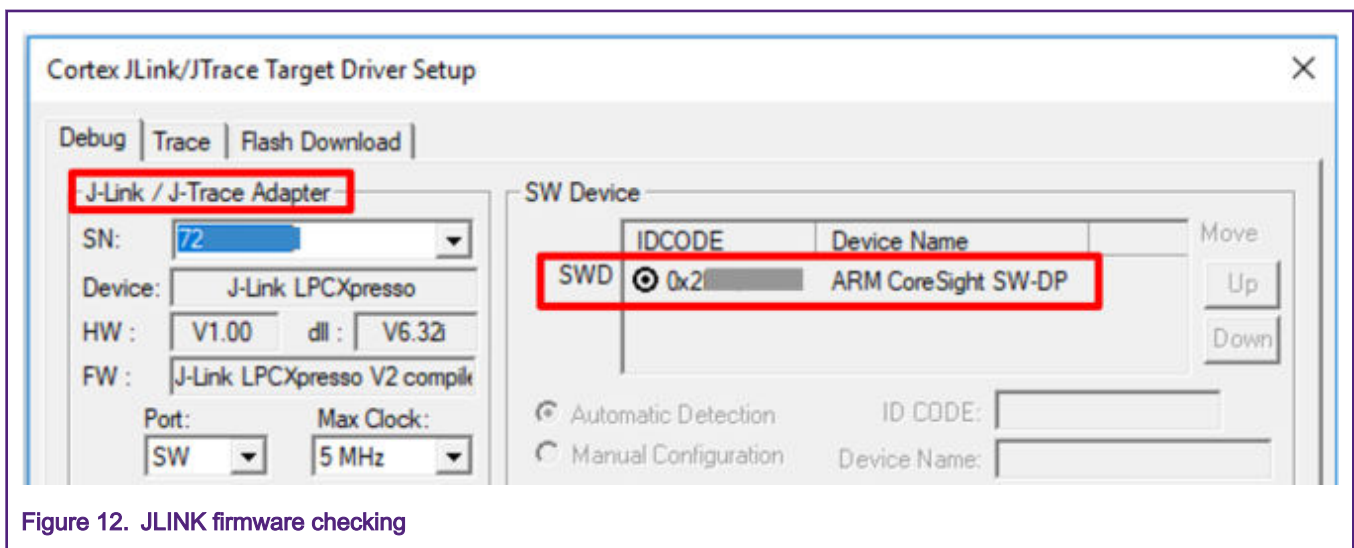


Figure 12. JLINK firmware checking

The following files are required for users if they want to download Bin files from PC to K32L2B:

- J-LINK.exe

- JLinkARM.dll
- XXXX.bat
- XXXX.txt
- Bin

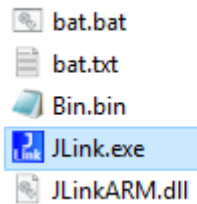


Figure 13. Test zip file list

For example, as shown in [Figure 13](#):

- bat.bat: Its content is `call JLink.exe -CommanderScript bat.txt` and it is responsible for calling JLink.exe.
- bat.txt: It defines where to download the specified Bin files.
- Bin.bin: Users can provide related Bin file according their real needs.

```

C:\WINDOWS\system32\cmd.exe
Downloading file [C:\... bin]...
J-Link: Flash download: Bank 0 @ 0x00000000: 1 range affected (32768 bytes)
J-Link: Flash download: Total time needed: 0.356s (Prepare: 0.084s, Compare: 0.059s, Erase: 0.112s, Program: 0.075s, Verify: 0.004s, Restore: 0.020s)
O.K.
Script processing completed.

Type "connect" to establish a target connection, '?' for help
J-Link>

```

Figure 14. Download process through JLINK

4 Verification

4.1 Logic analyzer results

[Figure 15](#), [Figure 16](#), and [Figure 17](#) help users to make sure that the K32L2B Bluetooth LE Audio System is working well.

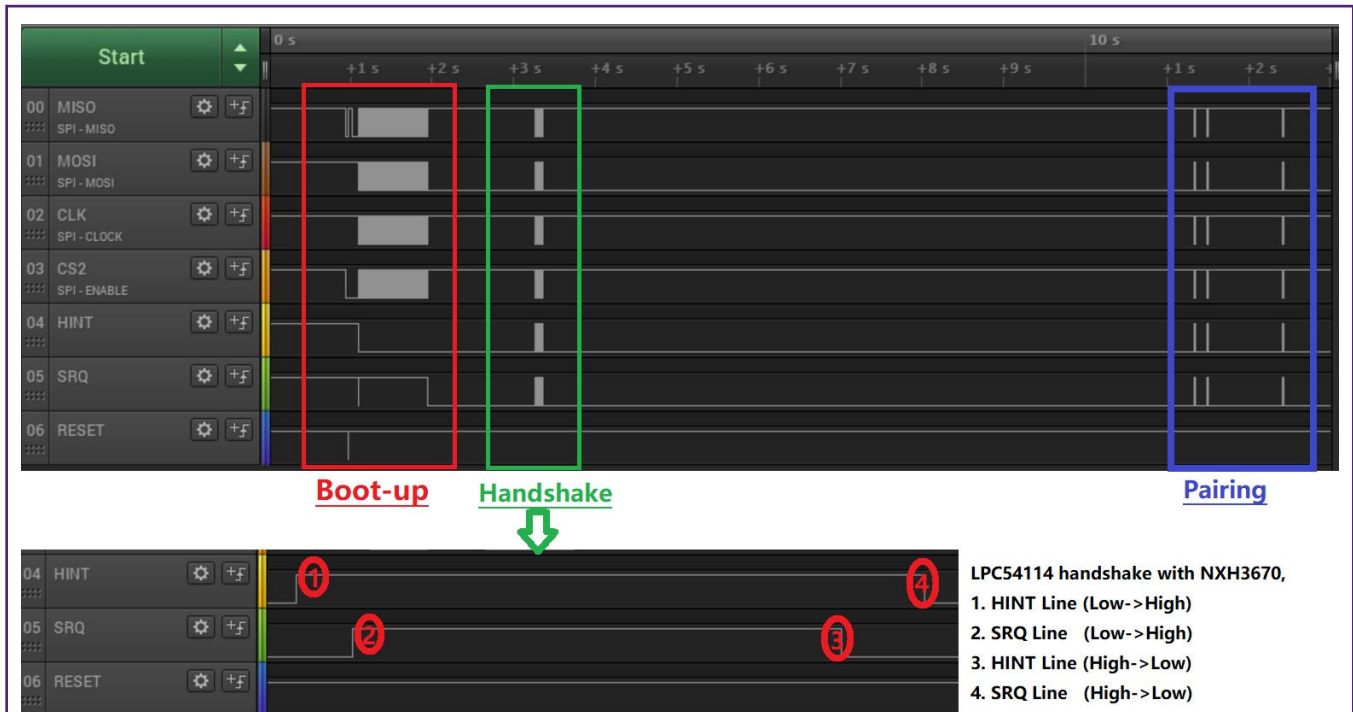


Figure 15. Logic analyzer result of Dongle

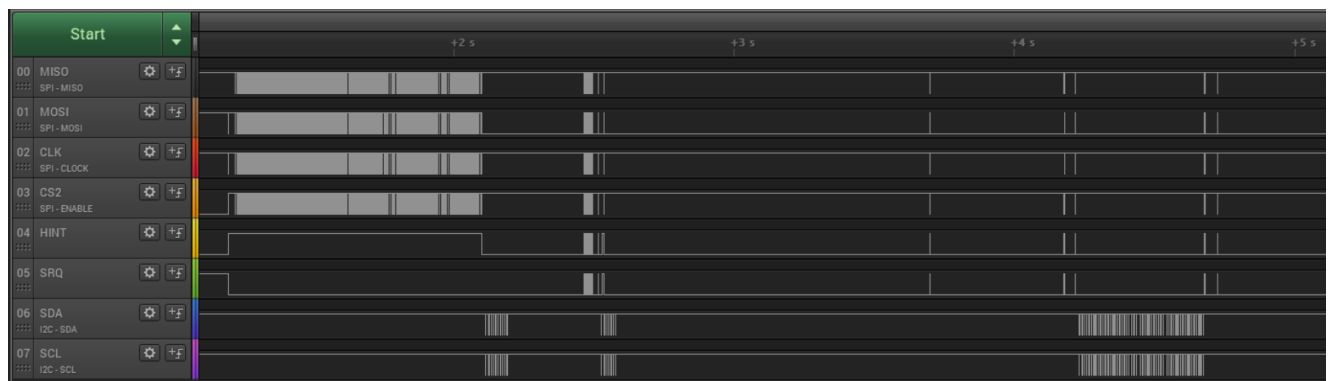


Figure 16. Logic analyzer result of Headset

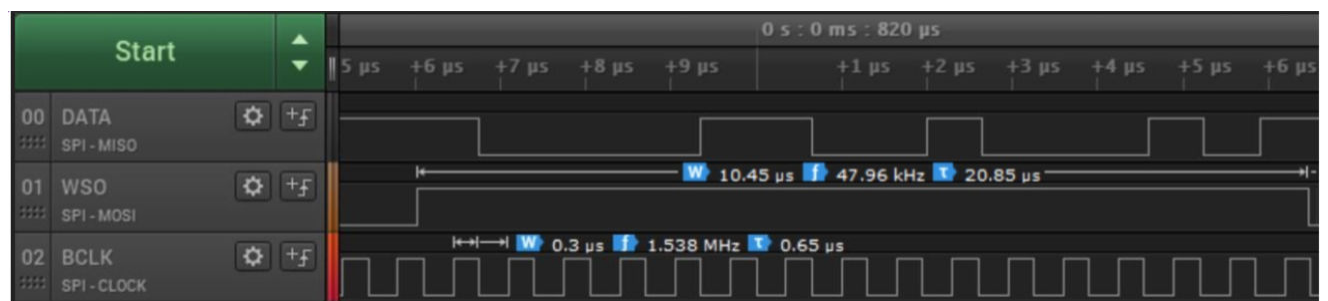


Figure 17. Logic analyzer result of I²S master emulated by FlexIO signal

4.2 Getting started with gaming user case

This application provides two cases for users:

- Play audio
- OTA

4.2.1 Play audio

Users can follow the steps below to verify the audio play function of **K32L2B Bluetooth LE Audio System**.

1. Connect the hardware following the introduction of Hardware design.
2. Make sure that related demos have been downloaded correctly, no matter using `IDE` or `JLINK.exe`.
Power on NXH3670 using the **PWR ON/OFF** button, and then boot-up, start and communicate with it.
3. Wait for Dongle to be paired and connected with Headset successfully.

The two NXH3670 are paired first and then connected.

For SDK Dongle board, the red LED is blinking while pairing, ON when paired, and OFF when connected.

For K32L2B Dongle, the blue LED is ON while pairing and OFF when paired, and the red LED is OFF when connected.

4. Select **Headphones** as the playback device for playing music.

Connect **J10** with PC using USB cable instead of **J13**.

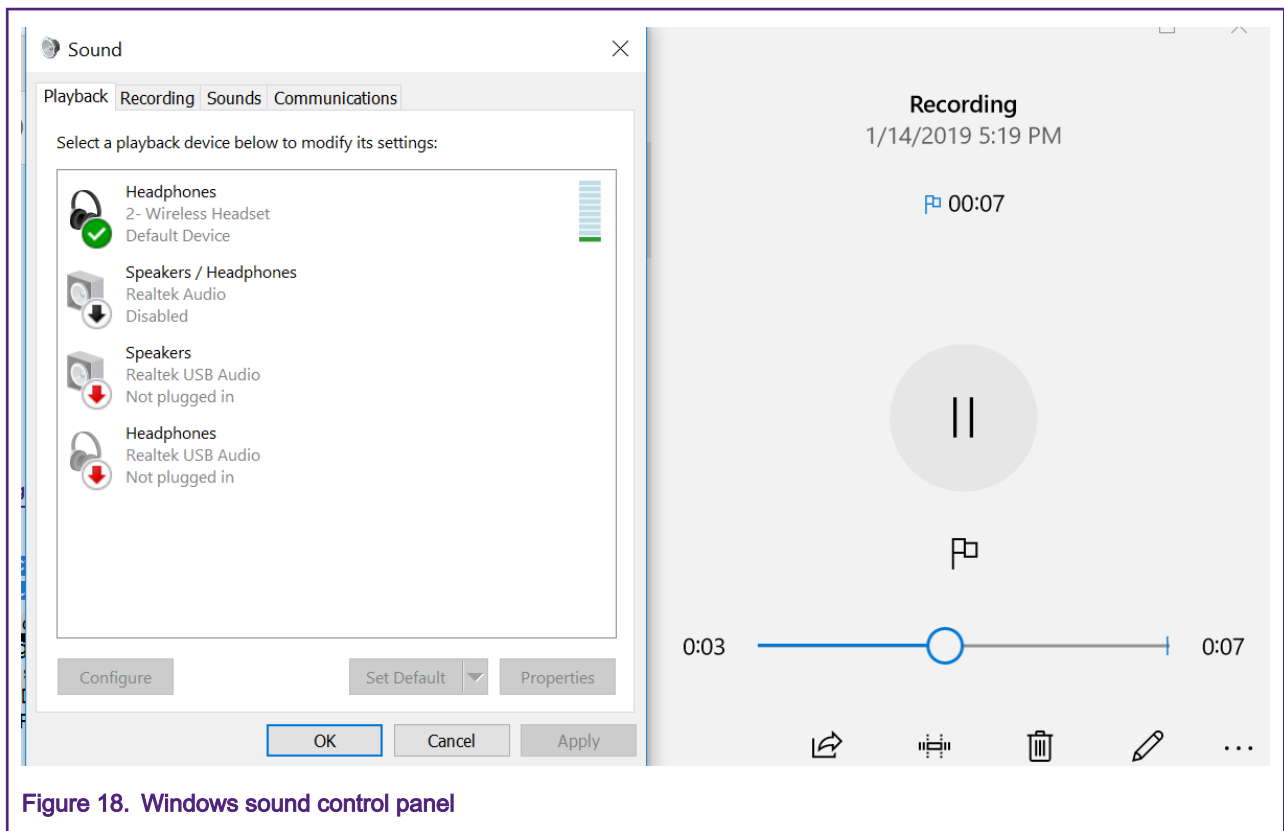


Figure 18. Windows sound control panel

4.2.2 OTA

User can upload new firmware that brings new features through OTA.

```

C:\k127\NXH3670_SDK_Gaming_G3.0\flash_scripts>ota update headset_ADZ96_20190511.bat
Updating an ADK[A] or SDK[S] board for headset? S
Enter USB port name of dongle: COM36
The USB port is COM36
Serial port COM36 opened
+++ reading layout file (C:\k127\NXH3670_SDK_Gaming_G3.0\flash_scripts\..\kinetis_democode\apps\k1_headset\script\layout
release_sdk ADZ96_20190511.yml)...
##### flashing binaries ...
+++ reading flash list file ...
+++ checking flash list file ...
+++ connecting to the remote device ...
+++ getting remote table version ...
version: 48
--- skipping ssb image
+++ flashing k1_app @ index 0 ...
+++ calculating local fingerprint for index 0 ...
local fingerprint: 3517288705
+++ getting remote fingerprint for index 0 ...
remote fingerprint: 1995252804
@ [1|0] 0x0 -> kinetis_democode/apps/k1_headset/sdk/release/4_1_led_blinky_2ABF0_eep.eep
[#####] 100%
--- skipping pairing_data image
--- skipping k1_ota_app image
--- skipping nxh_ota_app image
+++ changing remote active partition (1) ...
+++ rebooting remote to active partition ...

```

ota update headset_ADZ96_20190511.bat
S
COM36
layout
release_sdk ADZ96_20190511.yml)
4_1_led_blinky_2ABF0_eep.eep

- The new changed '.bat', '.yml' and '.eep' files according user's design
- The character that need user input.
- Command send from 'OTA_Dongle' to 'Headset' and event received from 'Headset' to 'OTA_Dongle'.(Flashtool will do this according user's '.bat' file)

Figure 19. CMD of OTA process

To save time for users, the flashtool does not update existing firmware. For more information, users can refer to *K32L2B OTA*.

5 Conclusion

This document provides basic introduction of the **K32L2B Bluetooth LE Audio System**. Users can use Dongle and Headset to verify related functions.

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, Altivec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Converge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, μ Vision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© NXP B.V. 2020.

All rights reserved.

For more information, please visit: <http://www.nxp.com>

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 01/2020

Document identifier: AN12645

