AN14531

在 **NXP LPC55S69** 上实现 **USB** 音频混合器

Rev. 1.0 — **2025**年**1**月**13**日 应用笔记

文档信息

信息	内容
关键词	AN14531_ZH, USB音频，音频混合器，USB音频1.0， USB音频2.0，音频延时
摘要	本篇应用笔记介绍了如何在 NXP LPC55S69 上实现 USB 音频混合器，介绍了基于 NXP SDK 例程来实现 USB 音频混合器的相关步骤，音频环形数组的管理，以及 USB 音频延时的测量。

1 简介

在有线游戏耳机的或者带有 USB Dongle 的无线游戏耳机的应用中，通常需要使用到 USB 音频混合器（USB audio mixer），例如在玩需要联机的网络游戏时，我们既要可以听到游戏中的声音，也要可以听到游戏里队友的声音，这就要求一个 USB 接口可以枚举出两个或者多个 USB 音频播放器（USB audio speaker）, 而且还要能够同步听到多个 USB 音频播放器的声音，同时也要支持 USB 麦克风的功能。 以有线游戏耳机为例，它通常支持聊天（Chat）和游戏（Game）两个 USB 音频播放器。 本篇应用笔记中实现的 USB 音频混合器简单来说就是将两个 USB 音频播放器接收到的USB音频数据进行融合，然后再通过 I^2S 接口将融合的音频数据传输给音频 Codec 播放，使客户可以同时听到这两个声音。 本篇应用笔记将指导客户如果在 LPC55S69 上实现 USB 音频混合器的功能，此 USB 音频混合器可以支持 USB 音频 1.0（USB audio class 1.0/UAC1.0）和 USB 音频 2.0 (USB audio class 2.0/UAC2.0)。

NXP 基于 KL27 和 NXH3670 无线游戏耳机的方案中（NXH3670 SDK Gaming Package）已经实现了 USB 音频混合器的功能，但是这个方案只支持 UAC1.0, 且使用的 KL27 SDK 版本为 SDK 2.7, 本篇应用笔记将介绍如何在 LPC55S69 上实现 USB 音频混合器的功能，此 USB 音频混合器可以支持以下的功能：

- 支持 UAC1.0 和 UAC2.0
- 支持 USB 音频同步模式
- 支持 LPC55S69 的高速 USB 和全速 USB 接口
- 支持 LPC55S69 SDK 2.15 版本
- 支持双声道 48 K/16 bit 音频格式

本方案中的 USB 音频混合器是基于 LPC55S69-EVK 和 LPC55S69 SDK 2.15 中的`usb_composite_audio_unified_bm` 例程实现, 将 NXH3670 SDK G9.2 中的 USB dongle mixer 的相关代码移植到 LPC55S69 SDK 2.15 的 USB 音频的例程中，并增加了 UAC2.0 的 USB 描述符。下面的章节将介绍如何在 LPC55S69 上实现 USB 音频混合器。

2 USB 音频混合器的实现

2.1 LPC55S69 SDK 例程

LPC55S69 SDK 2.15 中的 `usb_composite_audio_unified_bm` 例程中已经实现了一个 USB 复合类设备，且可以支持一个 USB 音频播放器, 一个 USB 录音器和一个 USB HID 类，我们可以把 SDK 中原有的 USB 音频播放器作为 USB 音频混合器中的 Chat 播放器, 因此我们只需要在这个基础上增加一个 Game 播放器就能实现 USB 音频混合器的功能了。

2.2 增加新的 USB 音频播放器接口

本章节将描述如何修改 USB 描述以及相关的变量和函数来增加一个新的 USB 音频播放器。

2.2.1 增加相应的 USB 描述符

增加如 表 1 中 USB 描述符来支持一个新的 USB 音频播放器（Game speaker）接口。

表 1. USB 音频播放器的 USB 描述符

USB descriptor type	USB descriptor name	Variable in USB descriptor structure	Descriptor length in UAC1.0	Descriptor length in UAC2.0	Comments
Interface Association Descriptor	Standard Interface Association Descriptor	`iadAudio`	8	8	
Audio Control (AC) Interface Descriptor	Standard AC Interface Descriptor	`control`	9	9	**UAC1.0** byte 7: `bInterface Protocol`, not used, must be set to 0. **UAC2.0** byte 7: `IP_VERSION_02_00`
Audio Class-Specific AC Interface Descriptor	Class-Specific AC Interface Header Descriptor	`controlSub`	9	9	**UAC1.0** byte 7: `bInCollection` byte 8: `baInterfaceNr` UAC2.0: byte 5: `bCategory` byte 8: `bmControls`
	Clock Source Descriptor	`controlSpkr.clock Source`	0	8	UAC1.0 has no CLOCK Source Descriptor.
	Input Terminal Descriptor	`controlSpkr.input Terminal`	12	17	**UAC2.0** byte 7: `bCSourceID` byte 14-15: `bmControls`
	Feature Unit Descriptor		0	18	UAC1.0 has no Feature Unit.
	Output Terminal Descriptor	`controlSpkr.output Terminal`	9	12	**UAC2.0** byte 8 `bCSourceID` byte 9 `bmControls`
Endpoint Descriptor	Endpoint Descriptor	`controlInterrupt Endpoint`	9	0	UAC2.0 does not require an interrupt In endpoint descriptor.
Audio Streaming Interface Descriptor	Standard AS Interface Descriptor (alt 0)	`streamSpkr.altSet0`	9	9	**UAC1.0** byte 7: `bInterface Protocol`, not used, must be set to 0. **UAC2.0** byte 7: `IP_VERSION_02_00`
	Standard AS Interface Descriptor (alt 1)	`streamSpkr.altSet1`	9	9	**UAC1.0** byte 7: `bInterface Protocol`, not used, must be set to 0. **UAC2.0** byte 7: `IP_VERSION_02_00`

表 1. USB 音频播放器的 USB 描述符...续上页

USB descriptor type	USB descriptor name	Variable in USB descriptor structure	Descriptor length in UAC1.0	Descriptor length in UAC2.0	Comments
	Class-Specific AS Interface Descriptor	`streamSpkr.as Interface`	7	16	**UAC1.0** byte 3: `bTerminalLink` byte 4: `bDelay` byte 5: `wFormatTag` **UAC2.0** byte 4: `bmControls` byte 5: `bFormatType` byte 6-9: `bmFormats` byte 10: `bNrChannels` byte 11-14: `bmChannel Config` byte 15: `iChannelNames`
	Class-Specific AS Format Type Descriptor	`streamSpkr.audio Format`	11	6	**UAC1.0** byte 4: `bNrChannels` byte 7 `bSamFreqType` byte 8-11 `tSamFreq`
Audio Streaming Endpoint Descriptors	Standard AS Isochronous Audio Data Endpoint Descriptor	`streamSpkr.iso Endpoint`	9	7	**UAC1.0** byte 7: `bRefresh` byte 8: `bSynchAddress`
	Class-Specific AS Isochronous Audio Data Endpoint Descriptor	`streamSpkr. specificIso Endpoint`	7	8	**UAC1.0** byte 4: `bLockDelayUnits` byte 5-6: `wLockDelay` **UAC2.0** byte 3: `bEndpoint Address` byte 5: `wMaxPacketSize`

本篇应用笔记中用 `usb_class_audio_headphones_device_descriptor_t` 结构体来表示 Game 接口的 USB 描述符，此结构体的包含的成员变量如 图 1 所示。

```
typedef struct _usb_class_audio_headphones_device_descriptor {
    usb_descriptor_interface_association_t iadAudio;
    usb_descriptor_interface_t control;
    usb_descriptor_class_specific_ac_interface_headphones_t controlSub;

    usb_class_audio_control_io_descriptor_group_t controlSpkr;
#if USB_DEVICE_CONFIG_AUDIO_CLASS_2_0
#else
    usb_descriptor_ac_interrupt_endpoint_t controlInterruptEndpoint;
#endif

    usb_class_audio_stream_descriptor_group_t streamSpkr;
#if USB_DEVICE_AUDIO_USE_SYNC_MODE
#else
    usb_descriptor_as_iso_sync_endpoint_t feedbackEndpointSpkr;
#endif
} usb_class_audio_headphones_device_descriptor_t;
```

图 1. `usb_class_audio_headphones_device_descriptor_t` 结构体

从 表 1 中可以看出 USB audio game speaker 接口描述符主要包括了接口关联描述符，音频控制接口描述符，音频流接口描述符，音频流端点描述符。另外，比较 UAC1.0 和 UAC2.0，相同的描述符的内容可能有所不

同，表 1 中 Comments 列也简单列出了 UAC1.0 和 UAC2.0 的描述符的不同之处，关于更加具体的对比，请参考 USB spec （USB Audio class 1.0 和 USB Audio class 2.0）和 AN14531SW。

2.2.2 修改其他配置

除了要增加 表 1 中 USB 描述符之外，也需要修改接口数量和端点的配置, 以及修改其他的变量和回调函数。

2.2.2.1 修改接口和端点的配置

实现 USB 音频 Game 播放器需要增加两个接口, 一个是音频控制接口，另一个是音频流接口。

- 新的接口配置如下
  - #define USB_AUDIO_CHAT_CONTROL_INTERFACE_INDEX (0)
  - #define USB_AUDIO_RECORDER_STREAM_INTERFACE_INDEX (1)
  - #define USB_AUDIO_CHAT_SPEAKER_STREAM_INTERFACE_INDEX (2)
  - **#define USB_AUDIO_GAME_CONTROL_INTERFACE_INDEX (3)**
  - **#define USB_AUDIO_GAME_SPEAKER_STREAM_INTERFACE_INDEX (4)**
  - #define USB_HID_CONSUMER_CONTROL_INTERFACE_INDEX (5)

需要为 Game 播放器增加两个 USB 端点，一个是音频控制端点，另一个是音频流端点。

- 新的端点配置如下：
  - #define USB_AUDIO_CHAT_CONTROL_ENDPOINT (6)
  - #define USB_AUDIO_CHAT_SPEAKER_STREAM_ENDPOINT (1)
  - #define USB_AUDIO_RECORDER_STREAM_ENDPOINT (3)
  - **#define USB_AUDIO_GAME_CONTROL_ENDPOINT (7)**
  - **#define USB_AUDIO_GAME_SPEAKER_STREAM_ENDPOINT (2)**
  - #define USB_HID_CONSUMER_CONTROL_ENDPOINT (4)

2.2.2.2 修改相关的变量和函数

除了要修改接口和端点数量之外，还需要增加 USB 枚举过程中使用到的 USB Game 播放器相关的一些变量，如 表 2 所示。

表 2. USB 音频播放器中添加的变量

USB Game speaker related variables
g_UsbDeviceAudioGameSpeakerEntity
g_UsbDeviceAudioGameSpeakerEntities
g_UsbDeviceAudioGameSpeakerControInterface
g_UsbDeviceAudioGameSpeakerInterfaces
g_UsbDeviceAudioInterfaceListGameSpeaker
g_UsbDeviceAudioClassGameSpeaker
g_CompositeClassConfig

关于更多的相关变量的修改，请参考 AN14531SW。

除了要修改 表 2 中的变量之外，也要修改相关的函数来处理 USB 音频接口相关的请求，这些函数如 表 3 所示。

表 3. 需要修改的函数

函数名称	说明
USB_DeviceCallback()	Add the processing of Set Interface request of game interface
APPInit()	Add audioGameSpeakerHandle related configuration
USB_DeviceAudioRequest()	Update audio specific request handling
USB_DeviceAudioSpeakerSetInterface()	Add processing for set Game interface request

注意 表 2 和 表 3 中的只是提到了需要修改或者增加的部分变量和函数，关于更多代码的修改请参考 AN14531SW。

2.3 环形数组的管理

本篇应用笔记沿用了 NXH3670 SDK 中 USB dongle mixer 的环形数组管理机制，每个音频播放器都使用了一个环形数组来管理音频的数据，其中音频数据的流向如 图 2 所示。

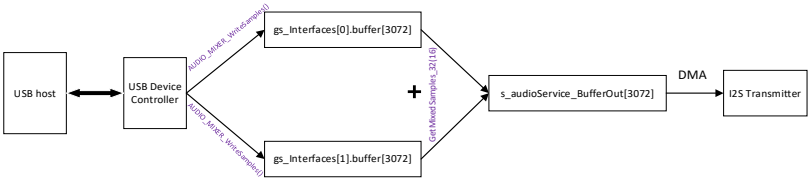

图 2. 音频数据的流向

环形数组的管理会使用到下面的几个文件。

- audio_ringbuffer.h
- audio_ringbuffer.c
- audio_mixer.h
- audio_mixer.c
- audio_tx.h
- audio_tx.c

其中 audio_mixer.c 文件中定义了 gs_Interfaces[2] 结构体数组，包括了两个结构体，对应了 Chat 和 Game 两个音频播放器。每个结构体中有个 buffer 成员变量，这两个 buffer 都为环形数组, 大小为 3072个 字节, 用来存储 Chat 和 Game 接口接收到的 USB 音频数据。在 USB 中断服务函数中，调用 AUDIO_MIXER_WriteSamples() 函数将接收的 USB 音频包拷贝至 gs_Interfaces[x].buffer[3072] 中。当环形数组的填充值达到设定的阈值时，开始 DMA 的传输，每次开始新的 DMA 传输之前都会调用 audio_GetAndTransmitSamples（audio_GetAndTransmitSamples 函数会调用 GetMixedSamples_32(16) 函数）函数将 gs_Interfaces[0].buffer 和 gs_Interfaces[1].buffer 中的音频数据进行合并，并将合并之后的数据拷贝至 s_audioService_BufferOut[3072] 数组中，然后开始 DMA 传输将混合的音频数据搬运至 I^2S TX FIFO 中进行播放。

2.4 USB 音频同步

本篇应用笔记中的 USB 音频设备是工作在同步模式的，这意味着 USB 设备需要以 USB 主机的 Start Of Frame (SOF) 信号为准，使 USB 设备端的音频数据的播放速度和 USB SOF 信号匹配。本篇应用笔记中通过 Ctimer 定

时器来捕获 USB SOF 信号，并在 Ctimer 的中断服务函数中调整音频时钟（Audio PLL）的小数分频系数来使音频时钟和 USB SOF 信号匹配。具体的实现方法请参考代码中 `CTIMER_SOF_TOGGLE_HANDLER_PLL()` 函数。

2.5 音频延时的测量

本篇应用笔记中的音频延时主要和开始 DMA 传输时的环形数组的填充水平有关，在本方案中，每一个 DMA 传输长度为一个 USB 音频包的长度，对于音频格式 2 声道的 48 K/16 bit 的 UAC1.0 设备，每一个USB音频包长度为 192 个字节，每一个 DMA 传输长度也为 192 字节，从 图 3 中可以看出，当环形数组的填充水平大于等于 4 个 DMA 传输长度时，才会开始 DMA 传输，也就是说至少要收到 4 个 USB 音频包才会开始 DMA 传输。对于 UAC1.0 设备，USB 主机每 1 ms 发送一个 USB 音频包，也就是从 USB 主机开始发送音频数据到 USB 设备端开始传输音频数据给 Codec 中间的音频延时约为 3 - 4 ms。而对于高速的 UAC2.0 设备，USB 音频包的间隔最小为 125 µs，也就是每一个微帧发送一个 USB 音频包，此音频包大小为 24 个字节。在 LPC55S69 SDK 中，默认将高速 UAC2.0 设备的音频发送间隔设置为了 4 个微帧，即 0.5 ms 发送一个音频包，每个音频包大小为 96 个字节，那么音频延时应该为 1.5 - 2 ms。 对于高速 UAC2.0 设备，如果将音频包间隔调整为 125 µs并且将初始的填充水平调整为 2 - 3 个 USB 音频包长度，那么理论上音频延时可以小于 1 ms。

```
if (!inInterface->isActive && (inInterface->fillLevel >= (AUDIO_GetTxDMATransferSize() * 4) )) {
    inInterface->isActive = true;
    gs_ActiveCount++;

    /* if this was the first interface to become active, then the mixer is ready to start outputting samples */
    if ((gs_ActiveCount == 1) && (gs_ReadyStateChangedCb != NULL)) {
        gs_ReadyStateChangedCb(true);
    }
}
```

Start I2S TX DMA transfer

图 3. 开始 DMA 传输的时机

可以使用逻辑分析仪测量 USB_DP/DM 信号以及 I^2S 信号来计算 USB 到 I^2S 的音频延时，如 图 4 所示。

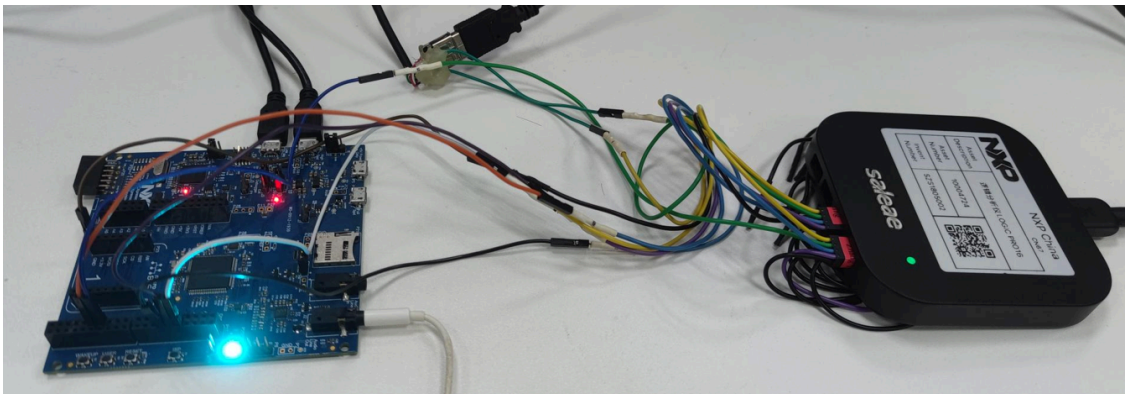

图 4. 使用逻辑分析仪测量 USB 到 I^2S 之间的音频延时

当初始填充值的阈值为 AUDIO_GetTxDMATransferSize() * 4 时，即收到 4 个 USB 音频包之后开始 DMA 传输，UAC1.0 的音频延时测量结果如 图 5所示，USB 到 I^2S 的音频延时为 3.4 ms。

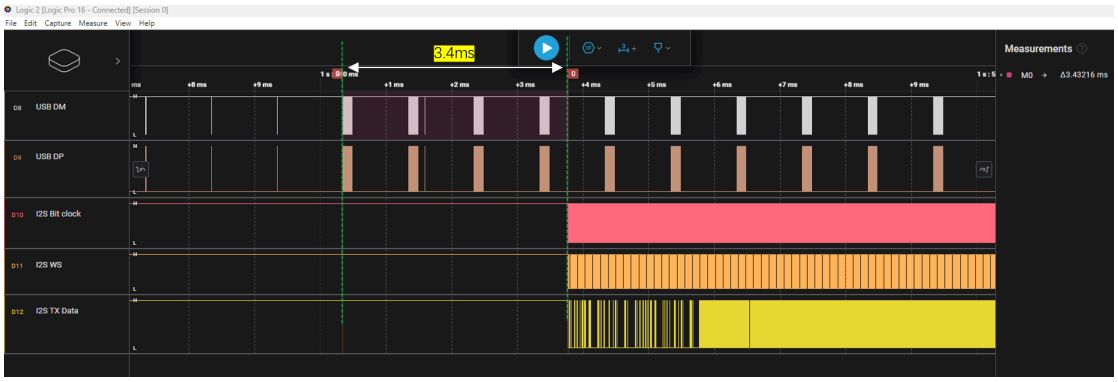

图 5. 测量 UAC1.0 设备的音频延时

当使用了高速 USB 接口且使能了 UAC2.0 模式时，高速 USB 的时钟为 480 M, 由于使用的逻辑分析仪的采样率最高为 500 M，因此是无法准确采集 USB DP/DM 信号的，此时我们可以用翻转 GPIO 的方式来测量 USB 到 I^2S 的延时，在每次接收到 USB 音频包的时候翻转一个 GPIO (P1_7).

```
case kUSB_DeviceAudioEventStreamRecvResponse:
    /* endpoint callback length is USB_CANCELLED_TRANSFER_LENGTH (0xFFFFFFFFU) when transfer is canceled */
    if ((g_deviceAudioComposite->audioUnified.attach) &&
        (ep_cb_param->length != (USB_CANCELLED_TRANSFER_LENGTH)))
    {
        USB_LatencyDebug_ReceivingAudioFromHost(true);  toggle P1_7
```

图 6. 在收到 USB 音频包时翻转 GPIO P1_7

然后在开始 DMA 传输以及 DMA 传输完成时翻转另一个 GPIO (P1_6)。

```
s_audio_GetSamplesCb = getMixedSamplesCb;
/* get and write samples into the output ring buffer */
audio_GetAndTransmitSamples(nbSamples);


HAL_AudioTransferSendNonBlocking((hal_audio_handle_t)&audioTxHandle[0], &s_TxTransfer);  Start TX DMA transfer
AUDIO_LatencyDebug_SendingAudioToI2s(true);
AUDIO_LatencyDebug_SendingAudioToI2s(false);   toggle P1_6
```

图 7. 在开始 DMA 传输以及 DMA 传输完成时翻转 GPIO P1_6

测量接收到第一个 USB 音频包到开始 DMA 传输之间的时间, 即为 USB 到 I^2S 的音频延时，如 图 8 所示，USB 音频的包的间隔为 500 µs, 即每 4 个微帧发送一个 USB 音频包，每个 USB 音频包长度为 96 字节，收到 4 个 USB 音频包之后开始 DMA 传输，此时 USB 到 I^2S 的音频延时为 1.65 ms。

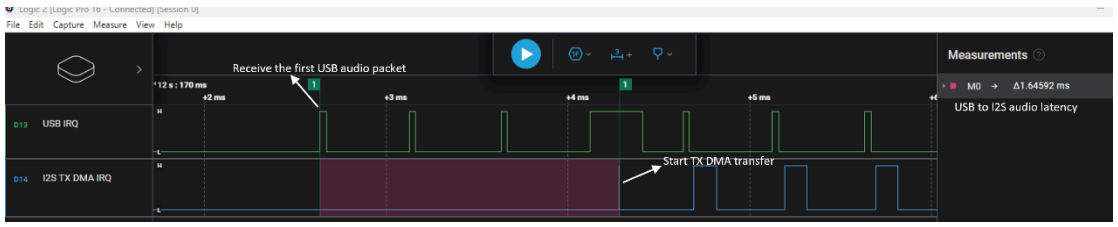

图 8. 测量高速 UAC2.0 设备的音频延时

如果想得到更小的音频延时，也可以将 USB 音频包的间隔设置为 125 µs, 即每一个微帧发送一个 USB 音频包，同时将环形数组的初始填充阈值调小，但阈值最小也要为两个 USB 音频包的长度，因为在开始 DMA 传输之前，我们至少需要准备两个 DMA 传输的数据来形成链式传输，第一个 DMA 传输完成之后立即开始第二个 DMA 传输来保证音频数据的连续播放。

3 USB 音频混合器的测试

本篇应用笔记实现的 USB 音频混合器支持 UAC1.0 和 UAC2.0 模式，且支持全速 USB 和高速 USB 接口，客户可以设置下面的几个宏定义来选择不同的工作模式，这几个宏是在 usb_device_config.h 文件中定义的。

- `#define USB_DEVICE_CONFIG_LPCIP3511FS (1U)`
- `#define USB_DEVICE_CONFIG_LPCIP3511HS (0U)`
- `#define USB_DEVICE_CONFIG_AUDIO_CLASS_2_0 (0U)`

当设置了想要的模式之后，就可以开始编译工程，并通过板载的调试器接口(P6)或者外部调试器接口 (P7) 将编译好的程序下载到 LPC55S69-EVK 中。按下板子上的 RESET 按键 (S4) 开始运行程序，请注意连接正确的 USB 口到 PC 或者其他 USB 主机上，全速 USB 接口是 P10, 高速 USB 接口是 P9，USB 主机会识别出一个复合的 USB 音频设备，如 图 9 所示。

图 9. USB 主机枚举出的 USB 音频混合器

在 USB 主机上打开两个的音频播放器，并分别选择 NXP Dongle mix (Chat) 和 NXP Dongle mix (Game) 播放音频，然后你可以连接 3.5 mm 接口的耳机到 LPC55S69 的耳机接口（J2）上，来听到一个混合的音频。

4 结论

本篇应用笔记介绍了如何在 LPC55S69-EVK 上实现一个USB音频混合器的功能，以 LPC55S69 SDK 中的 `usb_composite_audio_unified_bm example` 为基础，将 NXH3670 SDK 中的 USB 音频混合器的描述符移到这个基础工程中，并增加了 UAC2.0 的功能，另外还介绍了环形数组管理机制，USB 音频同步模式的实现，以及音频延时的测量，客户可以基于附件中的代码在 LPC55S69 以及其他的 NXP MCU 平台上实现 USB 音频混合器的功能。

5 参考

1. NXH3670 SDK Gaming Package
2. Getting started with NxH3670 gaming use case (document AN12360)
3. LPC55S6x/LPC55S2x/LPC552x User manual (document UM11126)
4. USB spec, Audio Device Document 1.0
5. USB spec, Audio Device Rev. 2.0

6 Note about the source code in the document

The example code shown in this document has the following copyright and BSD-3-Clause license:

7 修订记录

表 4 汇总了自初始版以来对本文档所做的更改。

表 **4.** 修订记录

文档号	日期	说明
AN14531_ZH v.1.0	2025 年 1 月 13 日	初次发布

内容