AN13364

在 KL27 上实现 USB 音频同步模式

版本 0 — 2021 年 8 月 26 日 应用笔记

目录

1 前言

在进行 USB 音频设备开发的时候，我们不得不考虑音频同步的问题。USB 协议中定义了音频传输的三种同步机制，如 表 1 所示。

表 1. USB 音频设备的同步机制

	Source	Sink
Asynchronous	Free running F_s. Provides implicit feedforward (data stream).	Free running F_s. Provides explicit feedback (isochronous pipe).
Synchronous	F_s locked to SOF. Uses implicit feedback (SOF).	F_s locked to SOF. Uses implicit feedback (SOF).
Adaptive	F_s locked to sink. Uses explicit feedback (isochronous pipe).	F_s locked to data flow. Uses implicit feedforward (data stream).

表 1 中的 Source 和 Sink 都是指 USB 设备，Source 是指产生音频数据并发送给主机的设备，如 USB 麦克风。Sink 是指接收来自 USB 主机的音频数据的设备，比如 USB 扬声器（播放器）。本应用笔记主要讲 USB 播放器设备中的音频同步，因此下文中的 USB 设备如无特殊说明都是指 Sink 设备。

三种同步机制的原理如下：

- 异步模式

  当 USB 端点工作在异步模式时，USB 设备端不会和 USB 主机的 SOF（Start of frame）信号保持同步，而是通过一个额外的反馈端点（feedback）告知 USB 主机设备端的真实的发送速率，USB 主机根据反馈值实时的调整自身的发送速率。

- 同步模式

  当 USB 端点工作在同步模式时，USB 设备端的传输速率需要和 USB 主机的 SOF 信号保持同步，通过调整设备端的时钟系统来使设备端的传输速率和 SOF 信号保持同步。

- 自适应模式

  当 USB 端点工作在自适应模式时，它可以工作在操作范围内的任意速率，而不只是 32 KHz，44.1 KHz，48 KHz 这些频率，对于 Sink 设备来说，USB 设备需要调整自身的传输速率来匹配 USB 主机的数据流。

关于同步机制的更多细节，请参考 USB 2.0 协议的 5.12 章节。

在 NXP SDK 中的 USB 音频播放器的例程中，一般使用异步模式来实现 USB 主机和设备之间的音频同步，即设备端通过反馈端点告诉主机设备端的真实传输速率，让主机调整实际的下发速率。有些 USB 主机是不支持异步模式的，即主机不会调整自身的

发送速率的，例如索尼的 Play Station（PS4）和 Nintendo Switch，这时候就需要设备端工作在同步模式，设备端通过调整自身时钟来匹配 USB 主机的传输速率。

根据一些客户的需求，NXP SDK 中有一些设备已经支持了同步模式，比如 LPC54608 和 RT600，他们的同步模式的实现方法如下：

- LPC54608

  在 LPC54608 的 SDK USB 音频播放器的例程中，程序会根据用于存储音频数据的环形缓冲区的余量来实时的调整 Free Running Oscillator（FRO）时钟的 trim 值。由于 FRO 时钟还作为 Phase Locked Loop （PLL）模块的输入时钟，因此由 PLL 分频得到的 I2S_BCLK（Bit Clock）和 I2S_WS（Word Select）也会被同步的更新。

- RT600

  在 RT600 的 SDK USB 音频播放器的例程中，通过 SCTimer 实时的测量 USB 主机的 SOF 帧间隔，根据帧间隔来调整 PLL 的小数分频系数，从而调整 I2S_BCLK 和 I2S_WS。

Kinetis L（KL）系列 MCU 是一款基于 Cortex-M0+的带有 USB 设备控制器的低功耗低成本微控制器，一些客户会选用 KL 系列微控制器作为 USB 音频播放器的控制器，如 KL27，但是如果客户想使 USB 音频播放器支持 PS4 这种不支持异步模式的主机，则需要使 KL27 的 USB 端点工作在同步模式。与 LPC54608, RT600 不同的是，KL27 是一款低成本的微控制器，并没有 PLL 模块和小数分频器，这就使得 KL27 不能像 LPC54608 和 RT600 一样通过调整 PLL 来调整 I2S_BCLK 和 I2S_WS。 本篇应用笔记将介绍一种基于 KL 系列 MCU 的音频同步模式的实现方法，通过调整 I2S_BCLK 的整数分频器和 I2S 字长（Word Length）来调整设备端的传输速率以匹配 USB 主机的发送速率。

2 实现方法

本节将以 KL27 为例，介绍如何在 USB 音频播放器中实现音频同步模式。

2.1 软件和硬件支持

- 硬件

  由于 FRDM-KL27 板子上的 KL27 为 MKL27Z64VLH4，此芯片没有 I2S 外设，且 FRDM 板子上没有 Codec 芯片，因此无法在此板子上运行 USB 音频播放器的程序。 NxH3670 SDK 板是基于 KL27 的无线耳机方案的评估板，板上有 MKL27Z256VMP4 和 Codec WM8904，所以选用此板作为测试的硬件平台，板子的配置如 图 1 所示。关于更多板子的细节，请参考 *NxH3670 SDK board*（文档 UM11150）。

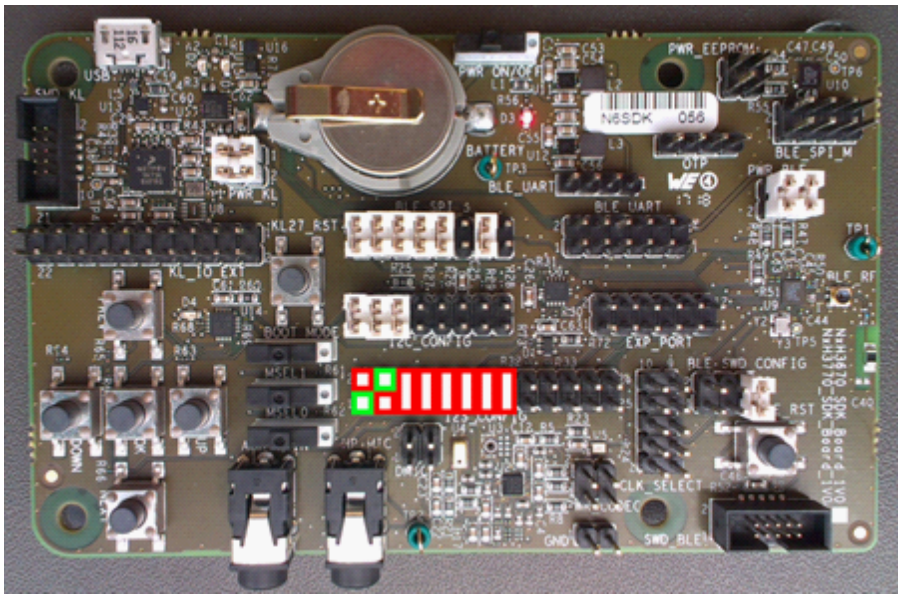

图 1. NxH3670 SDK 板配置

- 软件

  FRDM-KL27 的 SDK 包是针对 MKL27Z64VLH4 这颗芯片的，没有 I2S 的驱动，因此选用 FRDM-KL43 的 SDK 包中的 USB 音频例程作为基础工程。

2.2 系统框图

KL27 音频播放器的系统框图如 图 2 所示。

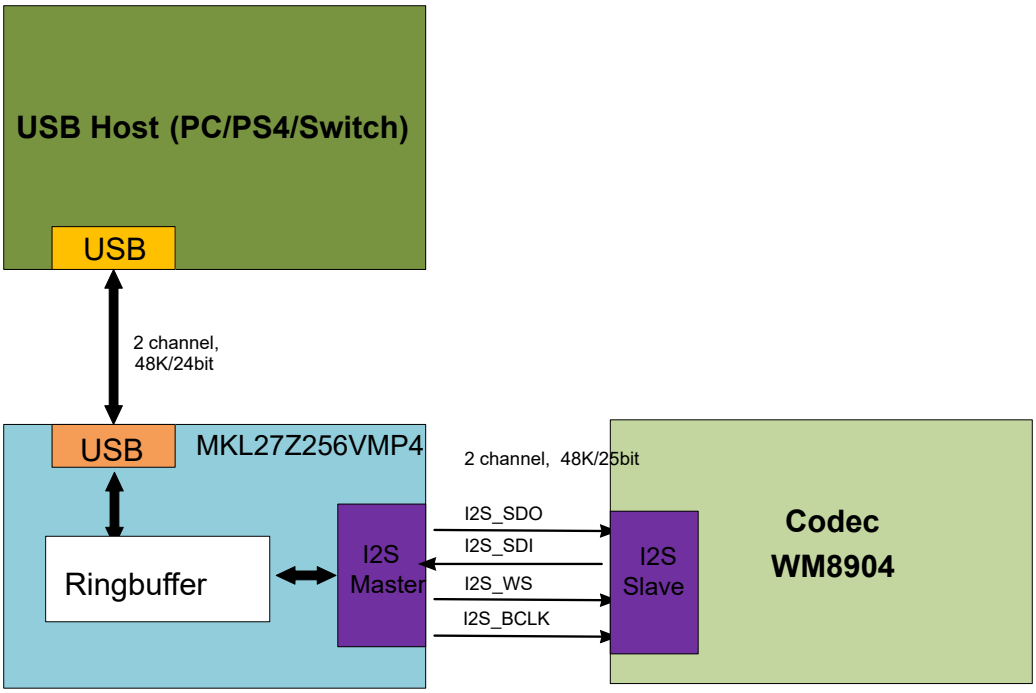

图 2. KL27 音频播放器的系统框图

2.3 USB 音频同步模式的使能

本例程将 KL43 SDK 中的 dev_composite_hid_audio_bm 工程作为基础工程，在此基础上，增加了 Codec 的驱动和音频环形缓冲区的相关代码。

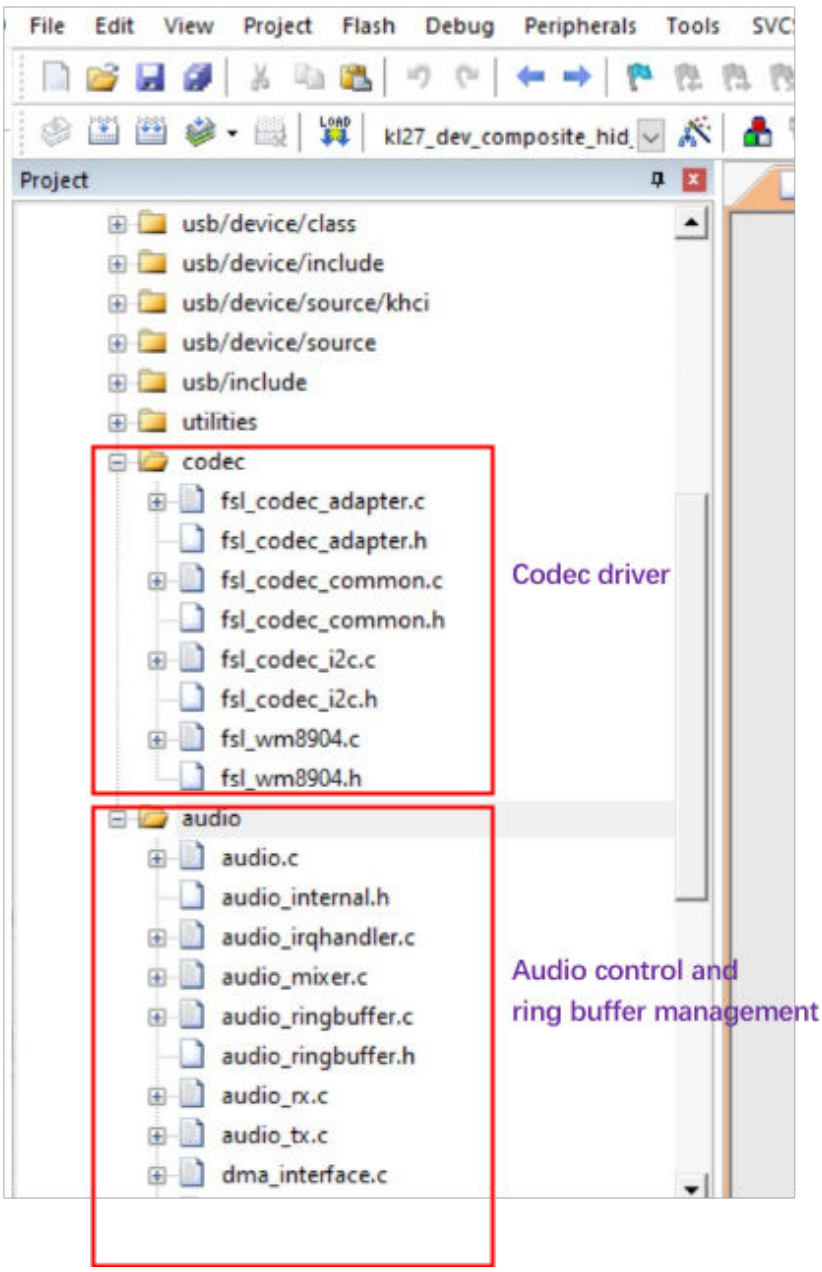

图 3. 增加 Codec 的驱动和环形缓冲区管理的代码

在原始代码中，USB 音频设备是工作在异步模式的，因此需要修改相关的 USB 描述符使其工作在同步模式，本例程中使用 USB_DEVICE_AUDIO_USE_SYNC_MODE 宏来配置 USB 音频设备工作在异步模式还是同步模式，将 USB_DEVICE_AUDIO_USE_SYNC_MODE 宏设置为 1，使 KL27 工作在同步模式，具体的配置细节请参考附件中的代码。

2.4 KL27 的时钟分配

配置 I2S 的 Master clock（I2S_MCLK）为 48 M 的系统时钟，然后将 I2S_MCLK 分频之后产生 I2S_BCLK。本例程中使用的 USB 音频设备的格式为双声道的 48 K/24 bit 数据流。若 I2S 接口也配置为 48 K/24 bit，则对应的 I2S_BCLK = 48K * 24 * 2 = 2.034 M。我们无法从 48 M 的 I2S_MCLK 整数分频得到 2.034 M，所以需要将 I2S 字长调整为 25 位，此时 I2S_BCLK = 48K * 25 * 2 = 2.4 M，可以将 I2S_MCLK 20 分频得到所需要的 I2S_BCLK。

表 2. KL27 时钟分配

Clock	Clock source	Frequency
System clock	HIRC	48 M
Bus clock	HIRC/2	24 M
USB functional clock	HIRC	48 M
I2S_MCLK	System clock	48 M
I2S_BCLK	I2S_MCLK/20	2.4 M

2.5 Codec 的配置

NxH3670 SDK 板上的 Codec 为 WM8904，由于本例程中使用的 USB 音频设备的格式为双声道 48 K/24 bit 数据，因此也需要将 Codec 配置为相同的数据格式，且 Codec 需要工作在 I2S 从机模式，即 I2S_BCLK 和 I2S_WS 信号由 KL27 提供。Codec 的主时钟（master clock）由板子上外接的 12.288 M 的晶振提供。

REGISTER ADDRESS	BIT	LABEL	DEFAULT	DESCRIPTION
	3:2	AIF_WL [1:0]	10	Digital Audio Interface Word Length 00 = 16 bits 01 = 20 bits 10 = 24 bits 11 = 32 bits
	1:0	AIF_FMT [1:0]	10	Digital Audio Interface Format 00 = Right Justified 01 = Left Justified 10 = I2S 11 = DSP

Table 54 Digital Audio Interface Data Control

Note that the WM8904 is a 24-bit device. In 32-bit mode (AIF_WL=11), the 8 LSBs are ignored on the receiving side and not driven on the transmitting side.

图 4. Codec 的接口配置

需要注意的是 WM8904 是一个 24 位的设备，如果将 KL27 的 I2S 字长配置为超过 24 位的长度，那么 WM8904 将会自动忽略超出 24 位长度的数据。即使将 I2S 主机的字长配置为 25 位，也不会影响 Codec 实际的播放效果。

2.6 环形缓冲区的管理

本例程使用环形缓冲区来存储 USB 主机发送过来的音频数据, 同时使用 DMA 将环形缓冲区中的数据搬运至 I2S 的 TX 数据寄存器中。环形缓冲区的管理由两个中断服务函数实现，这两个中断服务函数分别为：

- USB0_IRQHandler
- Audio_DMATxCallback

本例程中使用的环形缓冲区机制来自于 NxH3670 SDK 开发包，使用了三个环形缓冲区，分别为 gs_Interfaces[0].buffer[4096]，gs_Interfaces[1].buffer[4096]和 s_audioService_BufferOut[4096]。NxH3670 SDK 中的环形缓冲区机制支持音频混合的功能，可以同时输出游戏通道和聊天通道两个音频通道中的音频数据，其原理是将游戏和聊天两个音频接口中的数据分别存储在

gs_Interfaces[0].buffer 和 gs_Interfaces[1].buffer 两个环形缓冲区中，然后再按照时间的先后顺序将两个环形缓冲区中的数据拷贝至 s_audioService_BufferOut 数组中，然后使用 DMA 将 s_audioService_BufferOut 数组中的数据搬运至 I2S TX 数据寄存器中，这样人耳就可以同时听到游戏和聊天两个通道混合的声音。音频流示意图如 图 5 所示。

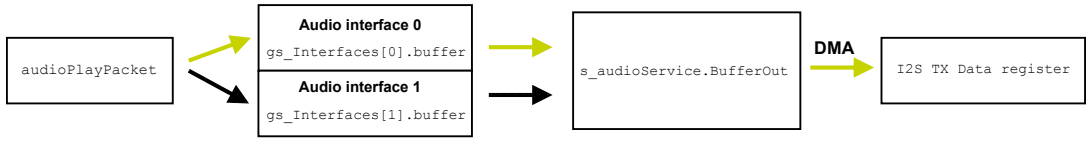

图 5. 音频数据流

在本例程中，只实现了一个音频接口，并没有使用到音频混合的功能，即音频数据流只会按照 图 5 中绿色箭头的指向来移动。

2.6.1 环形缓冲区的阈值设定

本例程中使用的 gs_Interfaces[0].buffer 环形缓冲区的长度为 4096，阈值设定如 图 6 所示。

```
36 /* Desired buffer upper limit (from 0 to 100)*/
37 #define BUFFER_UPPER_LIMIT_PERCENTAGE  (60)
38 /* Desired buffer lower limit (from 0 to 100)*/
39 #define BUFFER_LOWER_LIMIT_PERCENTAGE  (40)
```

图 6. 环形缓冲区的阈值设置

环形缓冲区的余量的正常范围在长度的 40%（1638）和 60%（2457）之间，若超出此范围，则需要调整 I2S 的传输速率。 若余量超过 2457，则需要加快 I2S 的传输速率，若余量小于 1638，则需要降低 I2S 传输速率。

2.6.2 USB 中断服务函数

USB 中断服务函数中对音频流的处理如 图 7 所示。

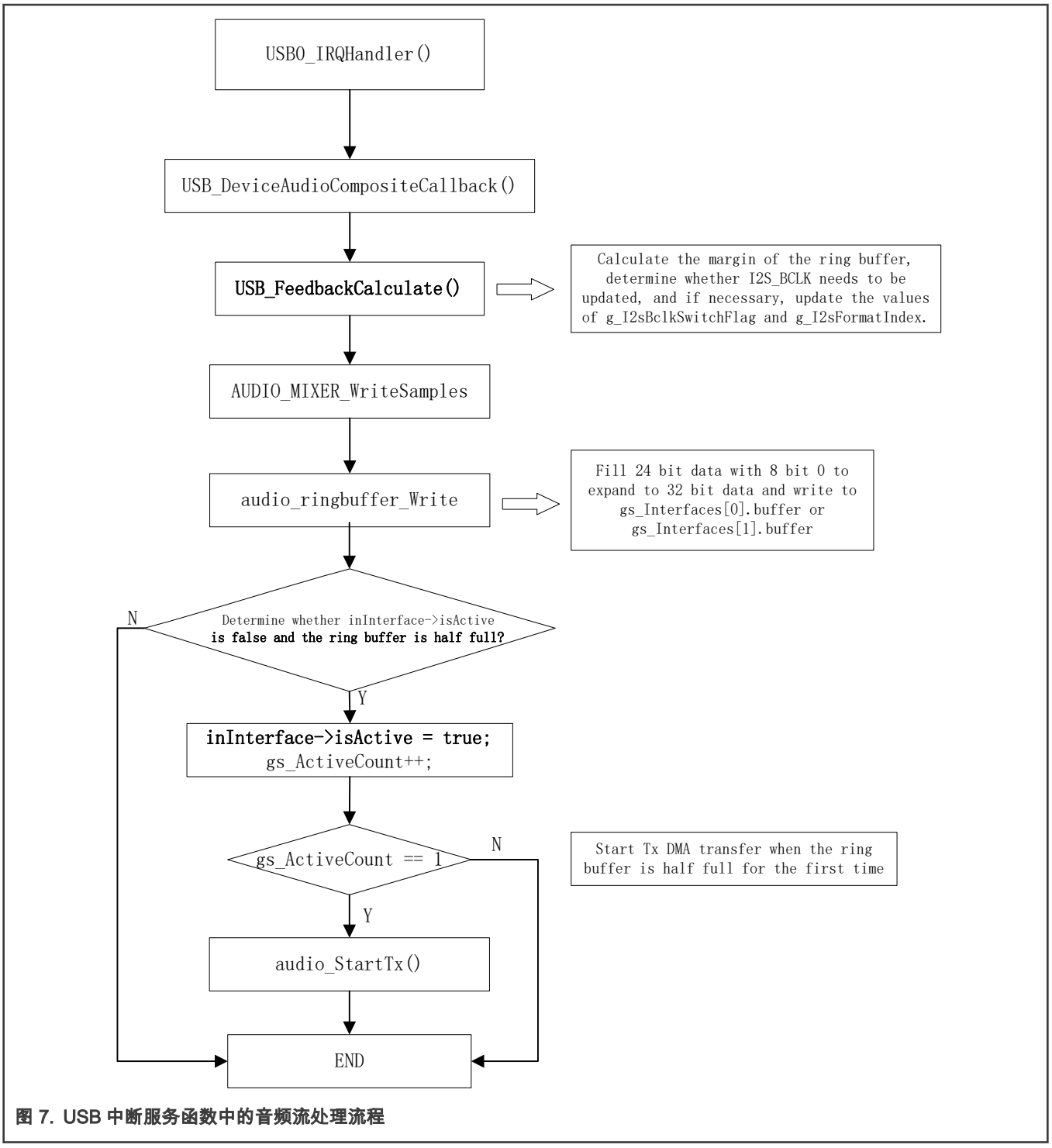

图 7. USB 中断服务函数中的音频流处理流程

对于全速 USB 音频设备, USB 主机每毫秒发送一包音频流数据，数据包大小为 288 个字节（48K*2*3）。由 图 7 可知，在USB 中断服务函数中首先会调用 USB_FeedbackCalculate 函数计算环形缓冲区的余量，并判断是否需要更新 I2S_WS，若需要更新，则更新 g_I2sBclkSwitchFlag 和 g_I2sFormatIndex 变量的值，需要注意的是 I2S_WS 并不会被马上更新，而是会等当前 DMA 传输完成，在 DMA 中断服务函数中被更新。执行完 USB_FeedbackCalculate 函数之后调用 audio_ringbuffer_write 函数将接收到的音频数据写入到 gs_Interfaces[0].buffer，在写入之前，需要先将 24 位数据扩充为 32 位数据，这是因为 DMA 每次搬运的长度为 32 位。然后判断此环形缓冲区是否半满，若半满，则调用 audio_StartTx 函数开启 DMA 传输，所以正常情况下，环形缓冲区的余量应该保持在长度的 50%左右。如果客户对音频数据的延时有比较严格的要求，那么可以减小缓冲缓冲区的长度来缩短音频延时。

2.6.3 DMA 中断服务函数

配置 DMA 的传输长度为 1ms 的音频数据流大小，即 384 个字节（48*2*4），DMA 每毫秒完成一次传输并触发一次 DMA 中断，在 DMA 中断服务函数中调用 audio_DMATxCallback 函数，audio_DMATxCallback 中的音频处理流程如 图 8 所示。

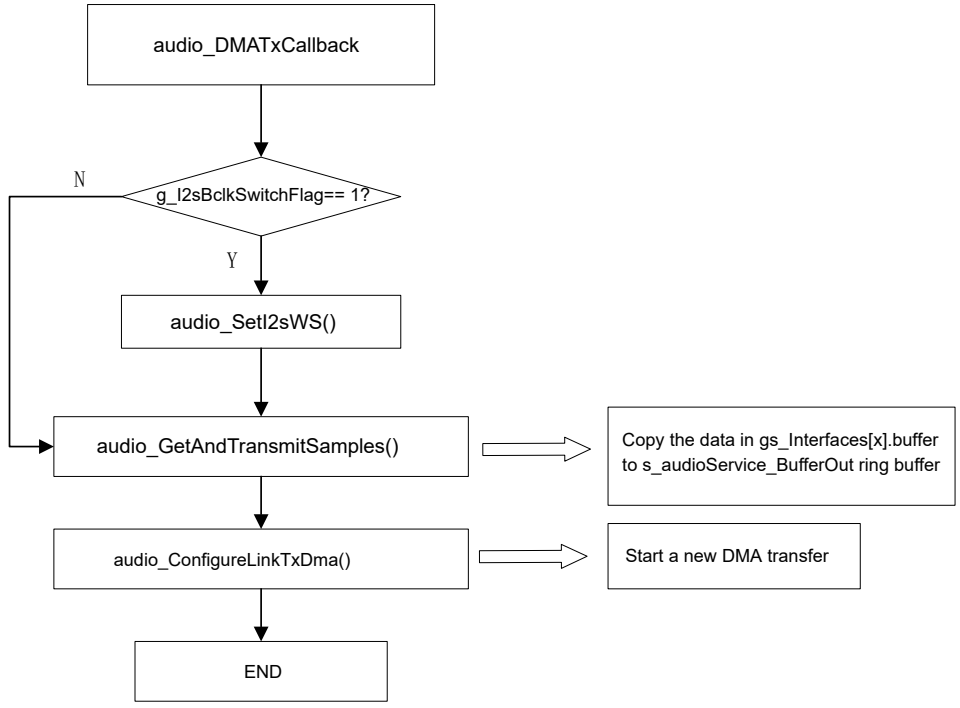

图 8. DMA 中断服务函数中的音频流处理过程

在 audio_DMATxCallback 函数中，首先会判断 g_I2sBclkSwitchFlag 变量是否为 1，若为 1，表明环形缓冲区的余量超出了设定的阈值范围，此时程序会调用 audio_SetI2sWS 函数调整 I2S_BCLK 的分频系数和 I2S 字长来使调整 I2S 的传输速率，使环形缓冲区的余量尽快回归到阈值范围内，避免造成环形缓冲区的上溢或者下溢。关于调整 I2S_WS 的细节，请参考更新 I2S 的传输速率。若不需要调整 I2S_WS，则直接调用 audio_GetAndTransmitSamples 函数将 gs_Interfaces[0].buffer 中的音频数据拷贝 384 个字节到 s_audioService_BufferOut 数组中，然后调用 audio_ConfigureLinkTxDma 函数开始一个新的 DMA 传输，从 s_audioService_BufferOut 数组中搬运 384 个字节到 I2S TX 数据寄存器中。

2.7 更新 I2S 的传输速率

由上一节的内容可知，如果环形缓冲区的余量超出了阈值范围，那么需要在当前 DMA 传输完成触发的中断服务函数中调用 audio_SetI2sWS 函数去更新 I2S_WS。根据 KL27 的时钟分配中的描述，I2S_BCLK 是由 I2S_MCLK 分频得到，且 I2S_BCLK = I2S_MCLK/(DIV+1)/2 = I2S_WS * 2 * WordLength，因此我们可以通过修改 I2S_BCLK 的分频系数和字长来调整 I2S_WS。

DIV	Bit Clock Divide Divides down the audio master clock to generate the bit clock when configured for an internal bit clock. The division value is (DIV + 1) * 2.

图 9. I2S_BCLK 的分频系数

由 图 9 可知，I2S_BCLK 的分频系数只能为偶数，初始的 DIV 的值为 9，此时分频系数为 20，此时 I2S_BCLK = 48M /20 =2.4M。若要增加 I2S_BCLK，需要将 DIV 的值设置为 8，此时分频系数为 18，I2S_BCLK = 48M / 18 = 2.67M, I2S_WS = I2S_BCLK/2/25bit =53.3K。若要减小 I2S_BCLK, 需要将 DIV 的值设置为 10，此时分频系数为 22，I2S_BCLK = 48M / 22 = 2.18M，I2S_WS = I2S_BCLK/2/25bit =43.6K。

如果只修改 I2S_BCLK 的分频系数，得到的 I2S_WS 与 48K 采样率相差较大，此时还可以调整 I2S 字长以获得更接近 48K 的采样率，如当分频系数为 22 时，将字长修改为 24bit，此时 I2S_WS = I2S_BCLK/2/24 = 45.4K。本例程中使用了如 表 3 中列出的三种 I2S_WS 频率。

表 3. 三种 I2S_WS 频率的配置

I2S_WS	Word Length	DIV	g_I2sFormatIndex
47619	28	8	0
48000	25	9	1
48387	31	7	2

```
typedef struct _i2s_tx_format_config_t
{
    uint32_t sampleRate;
    uint32_t  wordLength;
}i2s_tx_format_config_t;

i2s_tx_format_config_t g_I2sTxFormat[3] =
{
    {47619, 28},  // DIV 8
    {48000, 25},  // DIV 9
    {48387, 31},  // DIV 7
};
```

图 10. I2S TX 格式定义

本例程中使用了 47619，48000，和 48387 三种 I2S_WS 频率来匹配 USB 主机的发送速率。在 USB 中断服务函数中周期的检测环形缓冲区的余量并及时调整 I2S_BCLK 和 I2S_WS 来避免环形缓冲区中的音频数据上溢和下溢。关于更多更新 I2S_WS 的细节，请参考 AN13364SW。

3 测试

将修改之后的 SDK 代码下载到 NxH3670 SDK 板中并运行程序，图 11 为使用逻辑分析器抓取的 I2S 信号，测试时使用的USB 主机为 Windows 10。

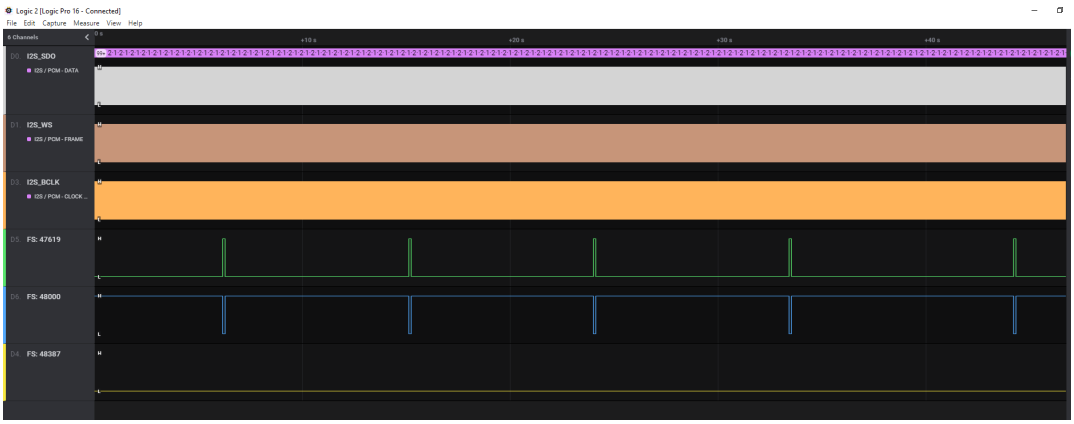

图 11. 逻辑分析仪抓取的 I2S 信号

从 图 11 中可以看出，在 Window 10 主机上测试时，大约九秒调整一次 I2S_WS，实际调整 I2S_WS 的频率与 USB 主机的时钟以及 USB 设备端 I2S 的时钟有关。

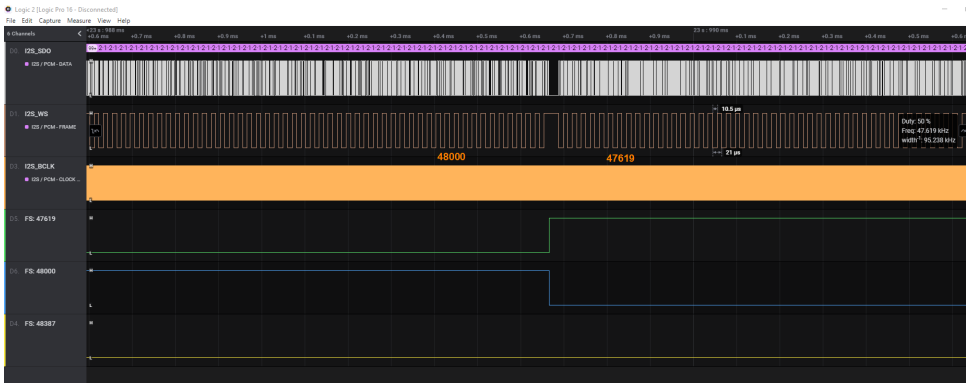

图 12. 更新 I2S_WS 频率

图 12 显示了更新 I2S_WS 的时刻，I2S_WS 的频率从 48000 切换为 47619，且字长从 25 位切换为 28 位，Codec 会自动忽略超出的 4 位数据。

4 结论

本篇应用笔记介绍了一种适用于 Kinetis L 系列 MCU 的 USB 音频播放器的音频同步方式，通过动态调整 I2S BCLK 的整数分频系数和字长来调整 I2S 的采样率，从而使 I2S 的工作频率和 USB 主机的发送速率保持同步。

5 参考

1. *NxH3670 SDK board*（文档 UM11150）
2. USB 2.0 Specification
3. Universal Serial Bus Device Class Definition for Audio Devices

6 修订记录

版本号	日期	说明
0	2021 年 8 月 26 日	初次发布

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Limited warranty and liability — 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.

Right to make changes - NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer’s applications and products. Customer’s responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer’s applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

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

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

Date of release: 2021 年 8 月 26 日

Document identifier: AN13364

arm