AN14177

Headset with Touch Function on MCX Nx4x

Rev. 1.0 — 20 January 2024 **Application note**

Document information

Information	Content
Keywords	AN14177, MCX Nx4x, gaming headset, USB audio, MICFIL, touch
Abstract	This application note describes how to use the MCX-N5XX-EVK to implement USB audio with touch control.

1 Introduction

Adding touch controls to the headset brings more operational diversity and convenience to users. This application note describes how to use the MCX-N5XX-EVK to implement USB audio with touch control.

The MCX Nx4x series microcontrollers combine the Arm Cortex-M33 TrustZone core with a CoolFlux BSP32, a PowerQuad DSP co-processor, and multiple high-speed connectivity options running at 150 MHz. MCX N54x and MCX N947 (VFBGA184) have high speed (HS) USB, SAI, DMIC, and TSI. Therefore, MCX Nx4x devices are suitable for the gaming headset solution.

Note: *The MCX N94x includes MCX N947 and MCX N946, among which MCX N946 and MCX N947 (100HLQFP) are not supporting DMIC.*

2 Implementation

The system block diagram of this application note is shown in Figure 1, which provides a brief overview of how the MCXN implements the headset with touch function.

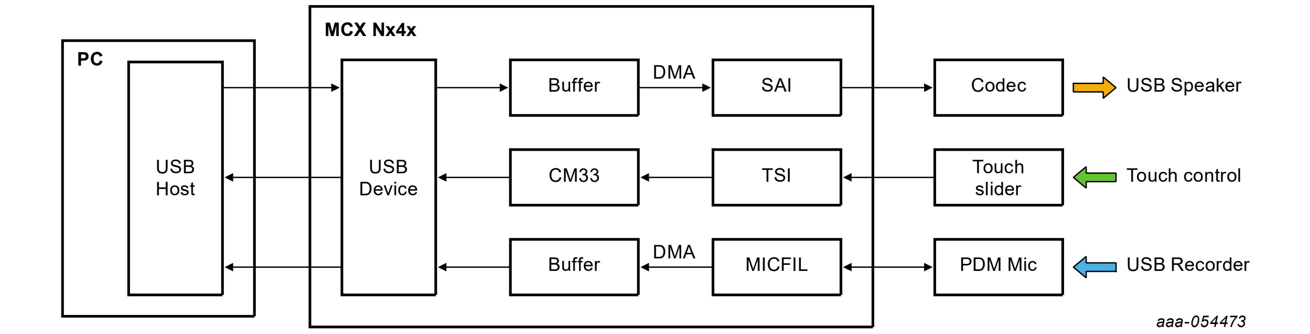


Figure 1. System block diagram

The code for this document is developed based on the `usb_device_composite_hid_audio_unified` example of MCX-N5XX-EVK SDK 2.13, and the IDE is IAR 9.40.1. To download the MCX-N5XX-EVK SDK 2.13, refer Build SDK for MCX-N5XX-EVK, and ensure to tick USB in the MCUXpresso SDK Builder.

The SDK example implements the USB speaker and recorder function. Based on this implementation, the document introduces how to enable the MICFIL module and touch control.

2.1 MICFIL introduction and use

The block diagram of the USB recorder is shown in Figure 1:

1. MICFIL module provides clock to PDM microphone and transforms the PDM data generated by the microphone into PCM data.
2. Then, the DMA transfers PCM data to the ring buffer.
3. Finally, the MCU sends the PCM data to the USB host via a high-speed USB interface.

MICFIL module transforms a pulse density modulated (PDM) microphone bitstream into a 24-bit PCM signal in the audio band, at a configurable output sample rate. Figure 2 shows the block diagram of MICFIL:

1. The internal time generator is used to generate a programmable clock for the PDM microphone. PDM data is converted into PCM data after the decimation filter.
2. Then, the PCM data is loaded into FIFO.
3. Finally, the PCM data in the FIFO can be read through interrupt or DMA.

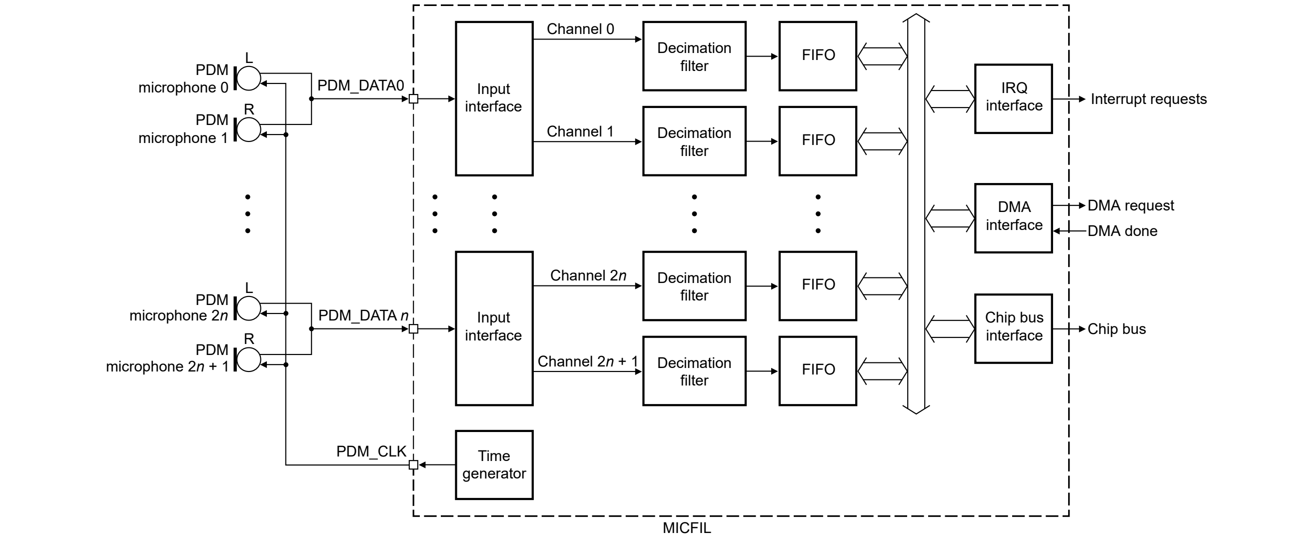


Figure 2. MICFIL block diagram

Figure 3 shows the timing diagram of the input interface signals when the clock divider is enabled. The bitstream incoming from the microphone data input "n" (PDM_DATAn) in the first half (right microphone) of the PDM_CLK is directed to channel "2n+1". The data generated during the last half (left microphone) is directed to channel "2n".

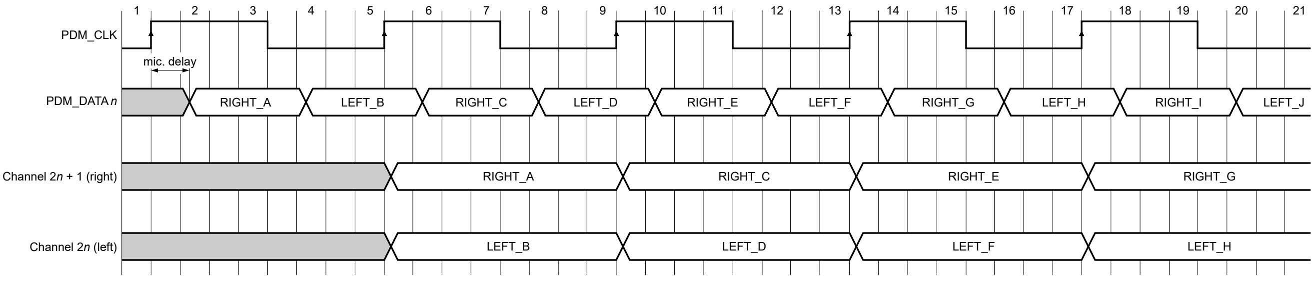


Figure 3. Input interface signals

Figure 4 shows the block diagram of the decimation filter:

- The cascaded integrator comb (CIC) filter converts PDM data from a digital microphone to PCM data at a given oversampling rate.
- Two half-band filters for each channel, which implement a low-pass digital filter with decimation by 2, can be used to compensate for the high CIC drop in passband.
- The DC remover is a high-pass filter that is used to remove the DC component of the processed signal with a configurable cut-off frequency.

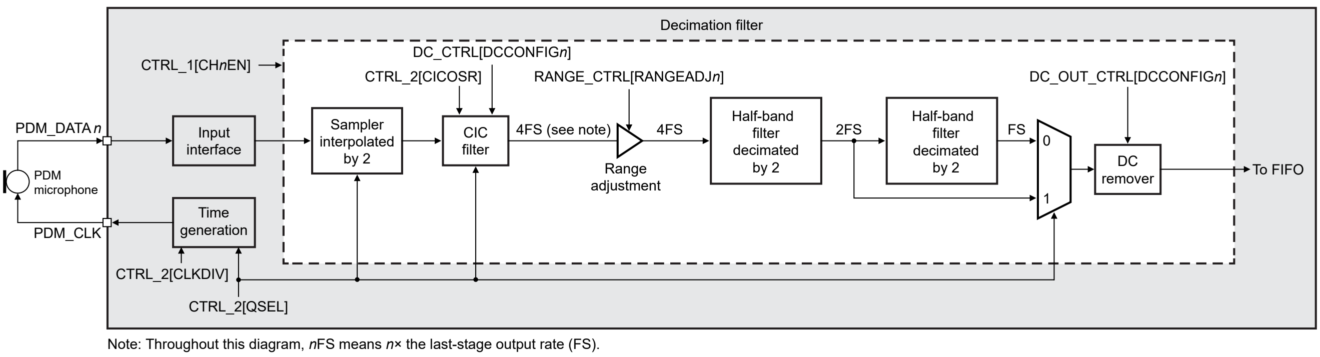


Figure 4. Decimation filter block diagram

OSR is the abbreviation of over sample rate. Equation (1) and Equation (2) show that CTRL_2[CICOSR] and quality mode selected define the CIC decimation rate:

$$OSR = 16 - \mathrm{CICOSR} \tag{1}$$

$$CIC\ \ decimation\ \ rate = \begin{cases} 2\ x\ \ OSR\ ; If\ \ HQ,\ \ VLQ0 \\ \quad OSR\ ; others \end{cases} \tag{2}$$

Table 1 shows the relationship between quality mode and coefficients in the decimation filter.

Table 1. Quality modes

Quality mode	CTRL_2[QSEL]	Sampler interpolation	CIC filter decimation	First-half band filter decimation	Second-half band filter decimation	PDM_CLK rate	Passband
High quality	001	-	: (2OSR)	: 2	: 2	Output rate x 8 x OSR	To ~ 0.5 x output rate
Medium quality	000	-	: OSR	: 2	: 2	Output rate x 4 x OSR	To ~ 0.5 x output rate
Low quality	111	x 2	: OSR	: 2	: 2	Output rate x 2 x OSR	To ~ 0.5 x output rate
Very-low quality 0	110	-	: (2OSR)	: 2	-	Output rate x 4 x OSR	To ~ 0.25 x output rate
Very-low quality 1	101	-	: OSR	: 2	-	Output rate x 2 x OSR	To ~ 0.25 x output rate
Very-low quality 2	100	x 2	: OSR	: 2	-	Output rate x OSR	To ~ 0.25 x output rate

As shown in Equation (3), the overall filter gain depends on quality mode, CIC decimation rate, and dynamic range adjustment of the CIC filter.

$$Overall\ \ filter\ \ gain\left(dB\right) = \begin{cases} 100\ x\ \log_{10}\left(32\ -\ 2\ x\ CICOSR\right)\ +\ 6.02\ x\ RANGE\_CTRL\left[RANGEADJ_n\right]\ -\ 150.50\ ; if\ \ QSEL \in \left[HQ,\ \ VLQ0\right] \\ 100\ x\ \log_{10}\left(16\ -\ CICOSR\right)\ +\ 6.02\ x\ RANGE\_CTRL\left[RANGEADJ_n\right]\ -\ 150.50\ ; others \end{cases} \tag{3}$$

Table 2 shows the dynamic range adjustment in different quality modes.

Table 2. Channel range adjustment

QSEL	$RANGEADJ_n$
HQ, VLQ0	<=25-ceil(5log2(2OSR))

Table 2. Channel range adjustment *...continued*

QSEL	$RANGEADJ_n$
MQ, VLQ1	<=25-ceil(5log2(OSR))
LQ, VLQ2	<=24-ceil(5log2(OSR))

Equation (4) and Equation (5) show how to calculate the CLKDIV value and PDM_CLK value:

$$CLKDIV = \frac{MICFIL\_CLK\_ROOT\ \ rate}{8\ x\ \ OSR\ \ x\ \ (output\ \ rate)} \tag{4}$$

$$PDM\_CLK\ \ rate = \frac{MICFIL\_CLK\_ROOT}{2\ x\ \ floor\ (K\ \ x\ \ CLKDIV)} \tag{5}$$

Note: *The output rate represents the sample rate.*

For K factor value, refer Table 3.

Table 3. K factor value

Quality mode	K factor
High quality	1/2
Medium quality, very-low quality 0	1
Low quality, very-low quality 1	2
Very-low quality 2	4

The following is the main configuration code of MICFIL for reference. For more related code, refer to the `pdm_sai_edma` example in the SDK.

```
/* Watermark value for FIFO: half of PDM FIFO depth */
pdmConfig.fifoWatermark = DEMO_PDM_FIFO_WATERMARK;
/* Quality mode: high quality */
pdmConfig.qualityMode = DEMO_PDM_QUALITY_MODE;
/* CIC filter over sampling rate: 0 */
pdmConfig.cicOverSampleRate = DEMO_PDM_CIC_OVERSAMPLE_RATE;
/* output DC remover cut off frequency: Bypass */
channelConfig.outputCutOffFreq = DEMO_PDM_OUTPUTCUTOFFFREQ;
/* Configure filter dynamic range */
channelConfig.gain = DEMO_PDM_CHANNEL_GAIN;
/* Initializes the MICFIL peripheral */
PDM_Init(DEMO_PDM, &pdmConfig);
/* Configures the MICFIL channel */
PDM_TransferSetChannelConfigEDMA(DEMO_PDM, &s_pdmRxHandle, DEMO_PDM_ENABLE_CHANNEL_LEFT,
 &channelConfig);
PDM_TransferSetChannelConfigEDMA(DEMO_PDM, &s_pdmRxHandle, DEMO_PDM_ENABLE_CHANNEL_RIGHT,
 &channelConfig);
/* MICFIL set sample rate */
PDM_SetSampleRateConfig(DEMO_PDM, DEMO_PDM_CLK_FREQ, DEMO_AUDIO_SAMPLE_RATE);
/* Performs a non-blocking PDM receive using eDMA */
PDM_TransferReceiveEDMA(DEMO_PDM, &s_pdmRxHandle, pdmXfer);
```

In addition, the user has to change the USB description for the recorder interface in the project from 16 bit to 32 bit. The specific implementation can refer to the code in this application note attachment.

To reduce latency, DMA is used to transfer PCM data in the MICFIL FIFO. The following is the main configuration code of DMA for reference. For more related code, refer to the `pdm_sai_edma` example in the SDK.

```
#define DEMO_DMA                  DMA0
#define DEMO_PDM_EDMA_CHANNEL     0
#define DEMO_PDM_EDMA_SOURCE      kDmaRequestMuxMicfil0FifoRequest
/* Initializes the eDMA peripheral */
EDMA_Init(DEMO_DMA, &dmaConfig);
```

```
/* Creates the eDMA handle */
EDMA_CreateHandle(&s_pdmDmaHandle, DEMO_DMA, DEMO_PDM_EDMA_CHANNEL);
/* Set channel request source */
EDMA_SetChannelMux(DEMO_DMA, DEMO_PDM_EDMA_CHANNEL, DEMO_PDM_EDMA_SOURCE);
/* Initializes the PDM Rx eDMA handle */
PDM_TransferCreateHandleEDMA(DEMO_PDM, &s_pdmRxHandle, pdmCallback, NULL, &s_pdmDmaHandle);
/* Install EDMA descriptor memory */
PDM_TransferInstallEDMATCDMemory(&s_pdmRxHandle, s_edmaTcd, 2);
```

Table 4 shows the specific configuration of the TCD, considering two-channel MICFIL as an example.

Table 4. TCD configuration

Register	Field	Description	Value
TCDn_SADDR	SADDR	Source address	DATACH[channel]
TCDn_SOFF	SOFF	Source address offset	FIFO_Width
TCDn_DADDR	DADDR	Destination address	Buffer
TCDn_DOFF	DOFF	Destination address offset	FIFO_Width
TCDn_ATTR	SSIZE	Source data transfer size	FIFO_Width
	DSIZE	Destination data transfer size	FIFO_Width
TCDn_NBYTES	NBYTES	Number of bytes to be transferred for each service request of the channel	channelNums* FIFO_Width
TCDn_BITER	BITER	Starting major iteration count	Data_Size/NBYTES

2.2 Touch introduction and usage

Touch sensing input (TSI) module provides touch sensing detection on capacitive touch sensors. The TSI module supports Self-capacitance mode and Mutual-capacitance mode. TSI fully supports the NXP touch library based on the SDK, which provides a solid capacitive measurement module for the implementation of touch keyboard, rotaries, and sliders. The TSI module provides 25 input channels.

This application note introduces the use of the touch slider on MCX-N5XX-EVK to implement USB audio control.

2.2.1 USB host control codec volume

Based on the SDK code, we are going to introduce how to implement the USB host control codec volume.

1. First, set the volume control range to the range corresponding to the codec:

```
g_deviceAudioComposite->audioUnified.volumeControlRange.wMIN         =
 USB_VolumeConversion_ConvertVolumeToUsb(AUDIOCODEC_MIN_OUTPUT_VOLUME_DB);
g_deviceAudioComposite->audioUnified.volumeControlRange.wMAX         =
 USB_VolumeConversion_ConvertVolumeToUsb(AUDIOCODEC_MAX_OUTPUT_VOLUME_DB);
```

2. Then, add the following code to change the volume of the device codec:

```
int16_t range_volume_db = 0;
int16_t range_volume_USB = 0;
int16_t codec_min_USB = 0;
int16_t codec_max_db = (int16_t)AUDIOCODEC_MAX_OUTPUT_VOLUME_DB;
int16_t codec_min_db = (int16_t)AUDIOCODEC_MIN_OUTPUT_VOLUME_DB;
range_volume_db = codec_max_db - codec_min_db;
range_volume_USB = USB_VolumeConversion_ConvertVolumeToUsb(range_volume_db);
codec_min_USB = USB_VolumeConversion_ConvertVolumeToUsb(codec_min_db);
uint8_t limit_value = 80;
/*
    0xb300: covert Db-value(-77db) to USB-value
    0x4d00: covert Db-value(77db) to USB-value
*/
```

```
uint8_t volAdj = ((int16_t)ATOS(g_deviceAudioComposite->audioUnified.curSpeakerVolume20) -
 (int16_t)codec_min_USB) * limit_value / range_volume_USB;
usb_echo("vol_codec_val = %d\r\n", volAdj);
BOARD_SetCodecVol(volAdj);
```

2.2.2 Touch control USB host audio

To get the SDK touch demo, tick the NXP touch library in the MCUXpresso SDK builder. Then, refer to the `touch_sensing` demo code in the `demo_apps` folder.

Note: *MCX-N5XX-EVK SDK 2.13 does not support ticking the NXP touch library, and it is going to be supported in SDK 2.14. You can also tick NXP Touch Library in MCX-N9XX-EVK SDK 2.13 and refer to the corresponding demo code.*

The `aslider_callback` function in the `touch_sensing` demo code returns an analog slider event and position. By comparing the positions of the release event and initial touch event, determine the left/right sliding distance or signal touch event.

Update the USB device hid keyboard report descriptor for USB host audio control:

```
uint8_t g_UsbDeviceHidKeyboardReportDescriptor[] = {
    0x05, 0x0c,                   /* USAGE_PAGE (Consumer Devices) */
    0x09, 0x01,                   /* USAGE (Consumer Control) */
    0xa1, 0x01,                   /* COLLECTION (Application) */
    0x15, 0x00,                   /*   LOGICAL_MINIMUM (0) */
    0x25, 0x01,                   /*   LOGICAL_MAXIMUM (1) */
    0x05, 0x0c,                   /*   USAGE_PAGE (Consumer Devices) */
    0x09, 0xcd,                   /*   USAGE (Play/Pause) */
    0x09, 0xe9,                   /*   USAGE (Volume Up) */
    0x09, 0xea,                   /*   USAGE (Volume Down) */
    0x09, 0xb5,                   /*   USAGE (Scan Next Track) */
    0x09, 0xb6,                   /*   USAGE (Scan Previous Track) */
    0x09, 0xb7,                   /*   USAGE (Stop) */
    0x95, 0x06,                   /*   REPORT_COUNT (6) */
    0x75, 0x01,                   /*   REPORT_SIZE (1) */
    0x81, 0x02,                   /*   INPUT (Data,Var,Abs) */
    0x75, 0x02,                   /*   REPORT_SIZE (2) */
    0x95, 0x01,                   /*   REPORT_COUNT (1) */
    0x81, 0x03,                   /*   INPUT (Cnst,Var,Abs) */
    0xc0                          /* END_COLLECTION */
};
```

In `hid_keyboard.c`, the `USB_DeviceHidKeyboardAction` and `USB_DeviceHidKeyboardCallback` functions send corresponding device requests based on the sliding direction and distance.

3 Test

To test the basic USB audio playback function of the SDK demo code, perform the following steps:

1. Connect MCU Link (J5) and HS USB (J27) to the computer via USB cable.
2. After compiling and downloading the demo to the MCX-N5XX-EVK board, reset the MCU.
3. Plug the 3.5 mm headphones into J7 of the EVK.
4. Set a USB device as the default playback device and the default recording device as shown in Figure 5.
5. Right-click on the speakers icon, then click **Sounds**.
6. Select **Playback** or **Recording**, then select "USB AUDIO+HID DEMO" and click **Set Default**.
7. Now, the headphones on the USB device side (EVK) is going to hear the sound played on the USB host side (PC).

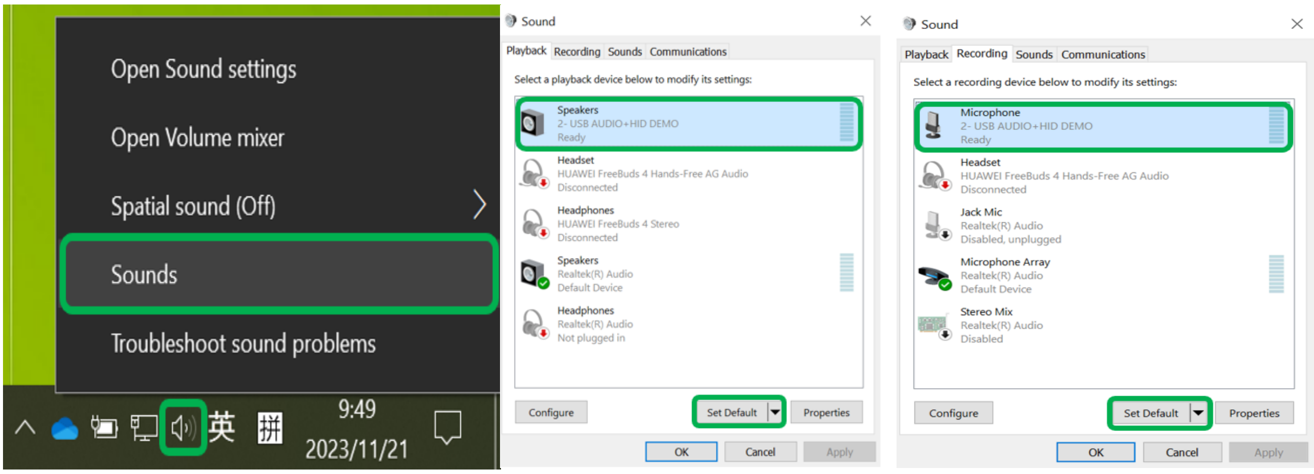


Figure 5. Set a USB device as the default device

3.1 Test USB device recording function

To test the USB device recording function, perform the following steps:

1. Change the jumper of JP7, JP8, JP10, and JP11 from the default to 2-3.
2. Click the Start button on Windows and search for "Voice Recorder", as shown in Figure 6.
3. Play the music closer to U30 and U32 on MCX-N5XX-EVK.
4. Play the recording and headphones on the USB device side (EVK) hear the sound acquired by DMIC (U30 and U32 on EVK).

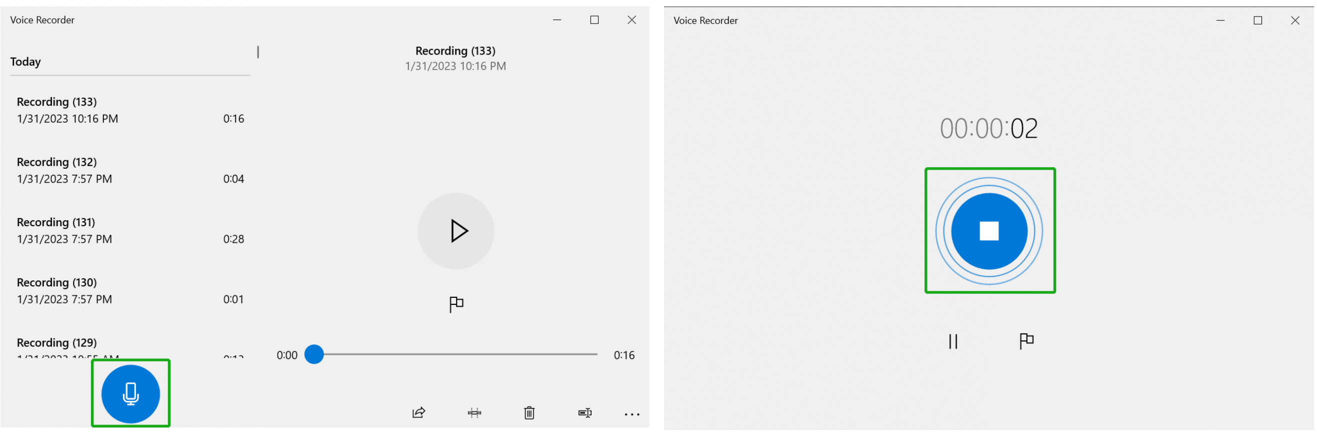


Figure 6. Voice recorder

If only one DMIC acquires the sound (U30 or U32 on EVK), disable audio enhancements as shown in Figure 7. After resetting the MCU, the user can hear the sound acquired by the left and right channels.

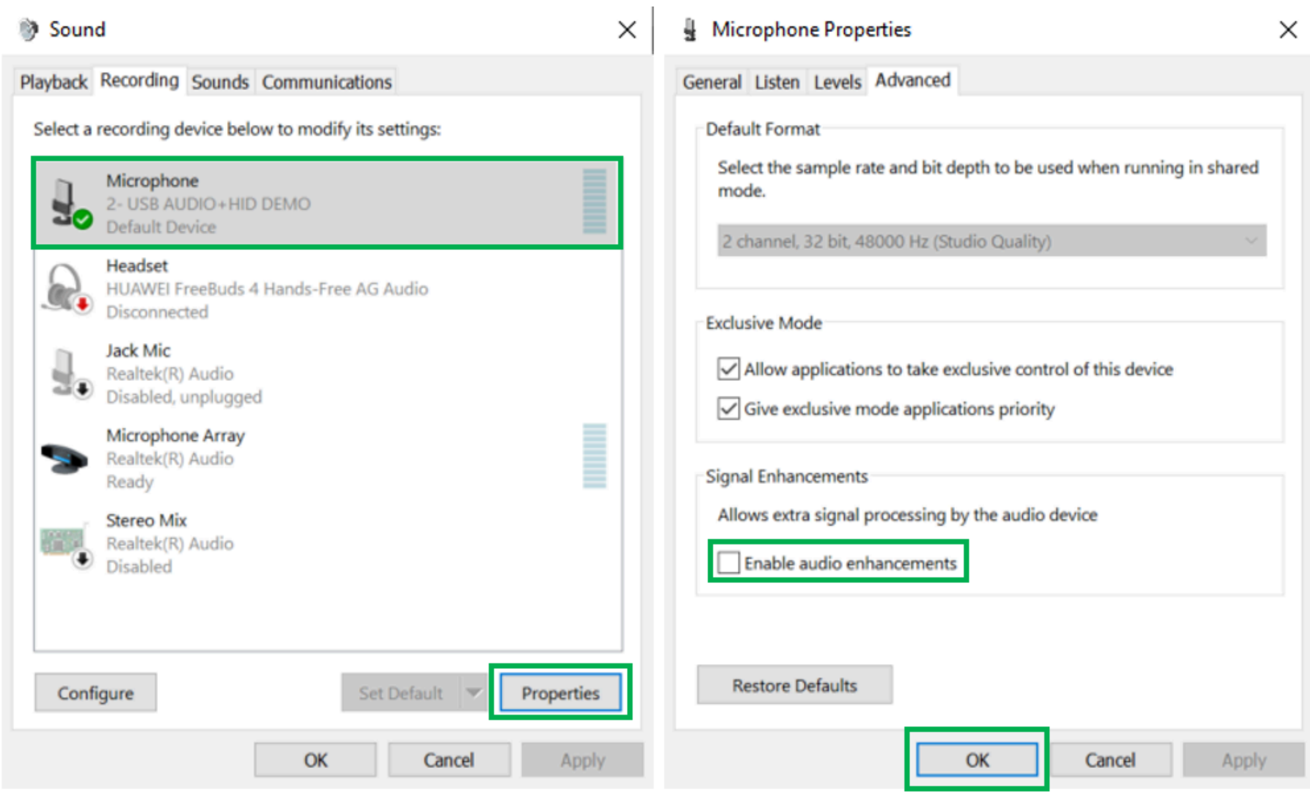
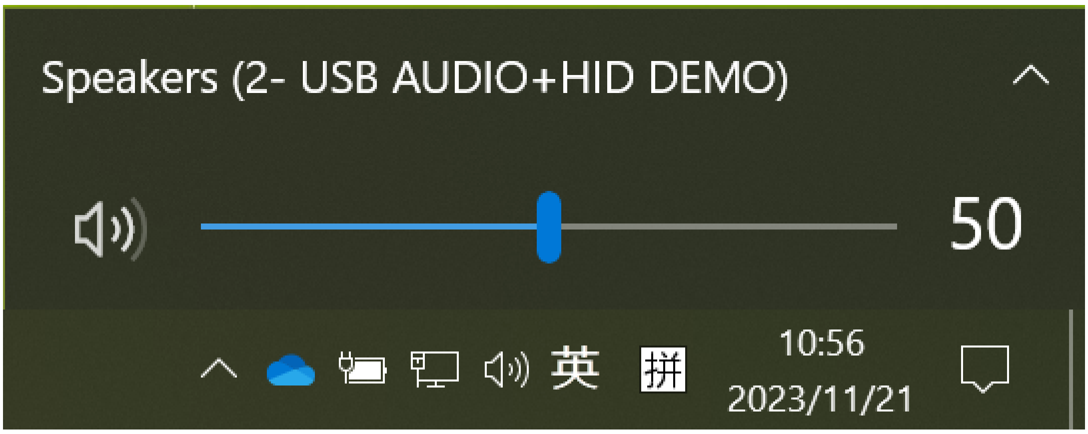


Figure 7. Disable audio enhancements

3.2 Test USB host change volume

To test the USB host change volume, perform the following steps:

1. Change the volume on the USB host side (PC). The volume adjustment information is printed on the terminal, as shown in Figure 8.
2. At the same time, the volume changes are heard on the headset side (headphones on EVK).



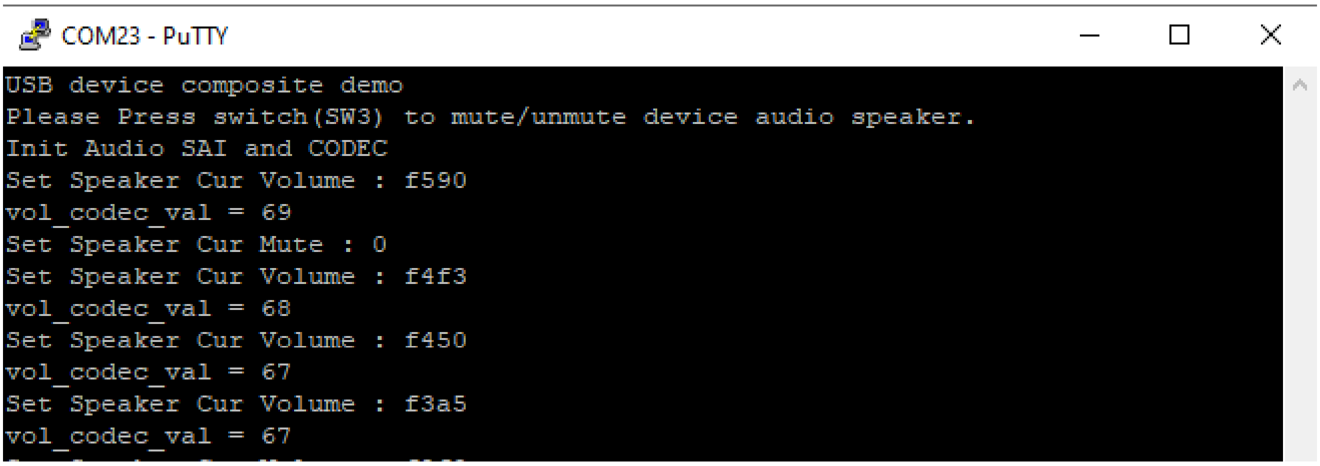


Figure 8. USB host change volume

3.3 Test touch control USB host audio

To test touch control USB host audio, perform the following steps and the result is shown in Figure 9:

1. To adjust the volume, swipe the touch slider (E1) left or right on the EVK.
2. To play/pause the music, click the touch slider (E1) on the EVK.
   Note: *When sliding and clicking, make the contact area between the finger and the slider as large as possible, and don't slide too fast.*

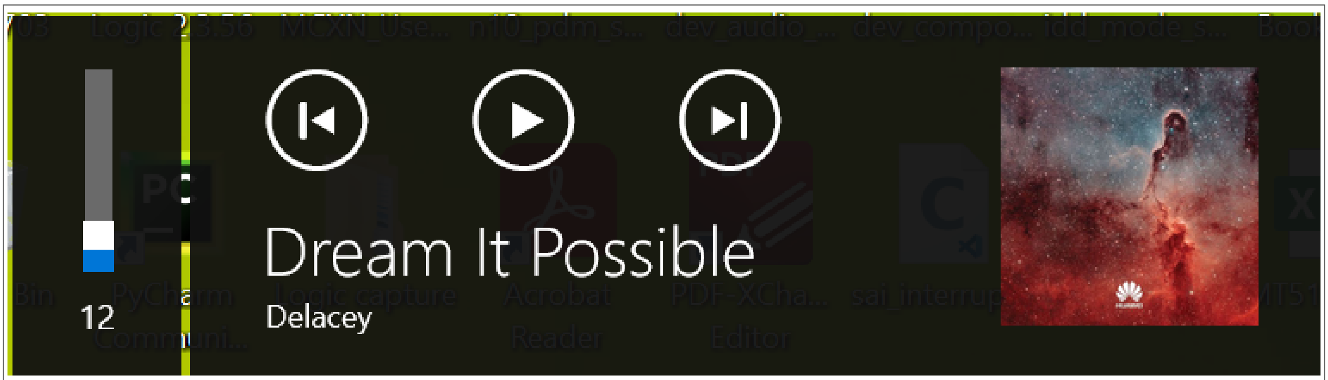


Figure 9. Touch control USB host audio

4 Summary

This application note based on the SDK example introduces how to enable the MICFIL module and use the TSI module to control the volume. It provides a reference for the gaming headset and brings more operational diversity and convenience to users.

5 Note about the source code in the document

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

Copyright 2024 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

6 Revision history

Table 5 summarizes the revisions to this document.

Table 5. Revision history

Document ID	Release date	Description
AN14177 v.1.0	20 January 2024	Initial public release

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

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.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. 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.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

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, μVision, Versatile — are trademarks and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) 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.

CoolFlux — is a trademark of NXP B.V.

IAR — is a trademark of IAR Systems AB.

MCX — is a trademark of NXP B.V.

Contents

Date of release: 20 January 2024
Document identifier: AN14177