AN14527

Emulating I2S Bus with the FlexIO on MCXA156

Rev. 1.0 — 9 January 2025 **Application note**

Document information

Information	Content
Keywords	AN14527, FlexIO, Audio, MCXA156, USB audio
Abstract	This application note describes how to emulate I2S interface with FlexIO on MCXA156 and implement a USB speaker device on MCXA156. The audio function is tested using the codec on LPCXpresso55S69 board.

1 Introduction

This application note describes how to use the FlexIO module to emulate the I2S interface on MCXA156. Using the FlexIO module can generate all the necessary I2S bus signals, which can replace the traditional I2S/SAI peripherals to transfer audio data.

The MCXA156 processor is based on the Arm Cortex-M33 platform. It is a low-power and low-cost MCU with a CPU clock of up to 96 MHz. It does not have the I2S/SAI interface, so the function of emulating the I2S interface with FlexIO is particularly important. It can enable the MCXA156 to achieve low-cost audio applications, such as a low-cost USB audio speaker. To verify the I2S interface emulated by FlexIO, a USB audio speaker application was implemented, based on the FRDM-MCXA156 and LPCXpresso55S69 boards. The system block diagram is shown in Figure 1.

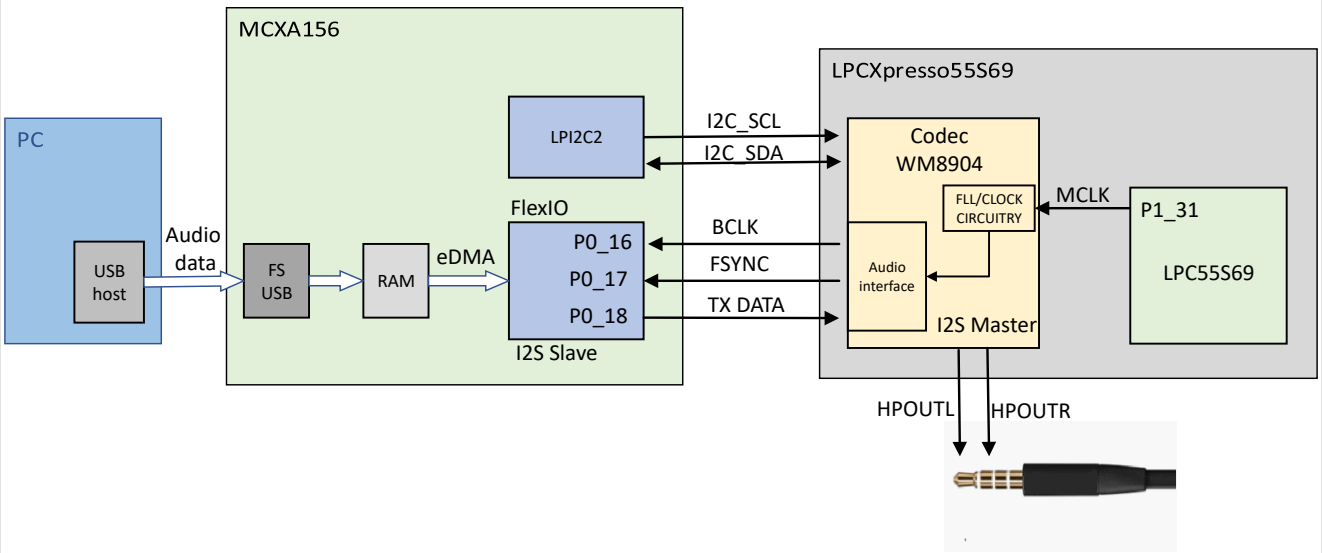

Figure 1. System block diagram of USB audio on MCXA156

In this application note, the I2S interface, emulated by FlexIO, is used as an I2S slave because MCXA156 does not have a PLL and there is no external audio crystal oscillator on the FRDM-MCXA156 board, such as 12.288M or 24.576M. Therefore, BCLK (Bit Clock) and FSYNC (Frame Sync/Word Select) must be provided by the WM8904 audio codec, which integrates the FLL internally. It can generate accurate BCLK and FSYNC based on the MCLK. The audio format used in this application note is as follows:

- Transmit mode: standard I2S mode
- Frame word count: 2
- Word length: 16
- BCLK frequency: 1.536 MHz
- FSYNC frequency: 48 kHz
- MCLK frequency: 24.576 MHz (provided by LPC55S69)

In a customer application, the MCLK can be provided by an external crystal oscillator. If the external crystal Y2 (8M) on FRDM-MXA156 is replaced with an audio crystal (12.288M/24.576M), the clock of this crystal can be used as the functional clock of the FlexIO module. The MCLK, BCLK, and FSYNC can be generated by the FlexIO. In this case, the I2S interface, emulated by FlexIO, can be used as the I2S master, and the codec can be used as the I2S slave.

The I2S interface, emulated by FlexIO, supports not only the standard I2S format, but also other audio formats, such as the left/right justified PCM format. Table 1 lists the features supported by the I2S interface, emulated by FlexIO, and compares them with the SAI/I2S interface on MCXN947.

Table 1. Features supported by the I2S interface emulated by FlexIO

SAI/I2S feature		I2S emulated with FlexIO on MCXA156	SAI on MCXN947	Comments
I2S	Standard I2S	Yes	Yes	-
	Left Justified	Yes	Yes	-
	Right Justified	Yes	Yes	-
PCM/TDM	Mode A	Yes	Yes	The I2S, emulated by FlexIO, supports up to 8 channels when the word length is 32. Another timer is needed to trigger FSYNC.
	Mode B	Yes	Yes	
Data alignment	First bit shift configuration	No	Yes	-
	LSB or MSB first	Yes	Yes	-
Word length	8 - 32-bit word length	Yes	Yes	-
Frame length	Maximum frame size of 32 words	No	Yes	FlexIO supports a maximum frame length of 256 bits.
FIFO	FIFO depth	1 * 32-bit	8 * 32-bit	-
	FIFO packing	No	Yes	-
	FIFO combine	No	Yes	-
Synchronous mode	Transmitter and receiver synchronization	Yes	Yes	-
Frame synchronization width	1 - 32-bit clock	Yes	Yes	-
Clock limitation	-	Yes	No	In the master mode, the maximum baud rate is the FlexIO clock frequency/4. For the slave mode, the maximum baud rate is the FlexIO clock frenquency/6.

2 Development platform

The hardware required for this application note is as follows:

- FRDM-MCXA156 A2
- LPCXpresso55S69 A2
- 2 Type-C USB cables
- 1 micro-USB cable
- 7 DuPont lines
- 3.5-mm headphone

Due to the lack of an audio codec on the FRDM-MXA156 board, use other boards with a codec to test the I2S interface emulated by FlexIO. In this application note, we use the codec on the LPCXpresso55S69 board to test the audio function, as shown in Figure 2.

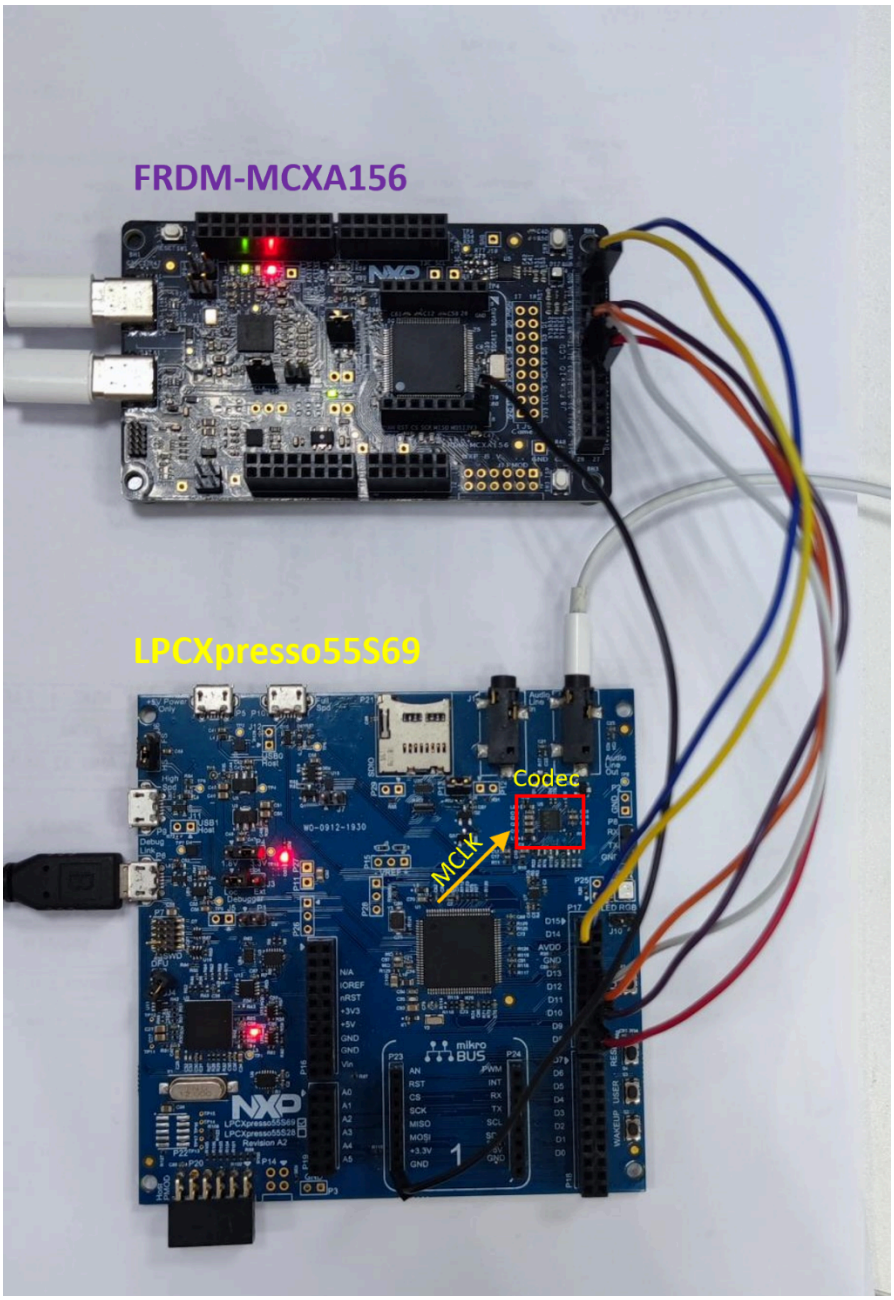

Figure 2. FRDM-MCXA156 and LPCXpresso55S69 board

The connection between the FRDM-MCXA156 and LPCXpresso55S69 boards is shown in Table 2.

Table 2. Connection between FRDM-MCXA156 and LPCXpresso55S69

I2S signals	FRDM-MCXA156	LPCXpresso55S69
BCLK	J8_13/P0_16	P17_14
FSYNC	J8_4/P0_17	P17_12
TX_DATA	J8_15/P0_18	P17_10
I2C_SCL	J8_4/P1_8	P17_3

Table 2. Connection between FRDM-MCXA156 and LPCXpresso55S69*...continued*

I2S signals	FRDM-MCXA156	LPCXpresso55S69
I2C_SDA	J8_3/P1_9	P17_1
GPIO	J8_7/P0_20	P17_18/P1_28
GND	J6_8	P23_8

On the LPCXpresso55S69 board, the I2S and I2C pins of the LPC55S69 connect to the codec are routed out to the P17 connector. Therefore, we can connect the FlexIO and I2C pins of the MCXA156 to the corresponding pins of the P17 connector to achieve the connection between the MCXA156 and the codec.

Note: *In this application note, MCLK is generated by the LPC55S69 and provided to the WM8904 audio codec.*

3 Implementation of FlexIO emulating I2S interface

This section describes the implementation of emulating the I2S interface with FlexIO, including FlexIO introduction, FlexIO configuration, codec configuration, LPC55S69 firmware development, and testing process.

3.1 FlexIO introduction

FlexIO is a highly configurable module with the following features:

- It supports the emulation of a wide range of serial/parallel communication protocols, such as UART, I2C, SPI, I2S, and so on.
- With flexible 16-bit timers, it supports a variety of trigger, reset, enable, and disable conditions.
- Its programmable logic blocks allow the implementation of on-chip digital logic functions and configurable interaction of internal and external modules.
- It contains the programmable state machine for offloading basic system control functions from the CPU.

Figure 3 provides a high-level overview of the FlexIO timer and shifter configuration.

FlexIO uses shifters, timers, and external triggers to shift data into or out of FlexIO. As shown in Figure 3, timers control the timing of this data shift. You can configure the timers to use generic timer functions, external triggers, or various other conditions to determine the control.

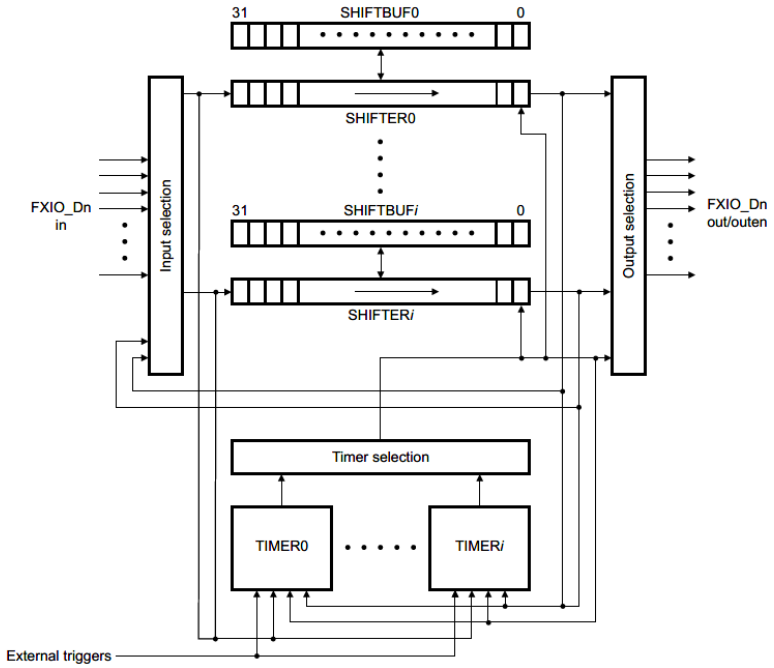

Figure 3. FlexIO block diagram

On MCXA156, FlexIO has 4 shifters and 4 timers, and supports a total of 32 FlexIO pins. In this application note, 3 FlexIO pins (FlexIO_D0, FlexIO_D1, FlexIO_D2) are used to emulate the BCLK pin, FSYNC pin, and TX pin of the I2S interface, respectively. Shifter0 is used for the TX pin. Timer0 and Timer1 are used for the BCLK and FSYNC pins. Figure 4 shows the internal connection of the FlexIO emulating I2S interface. The pins of Timer0 and Timer1 correspond to BCLK and FSYNC, and the SHIFTER0 pin corresponds to the TX.

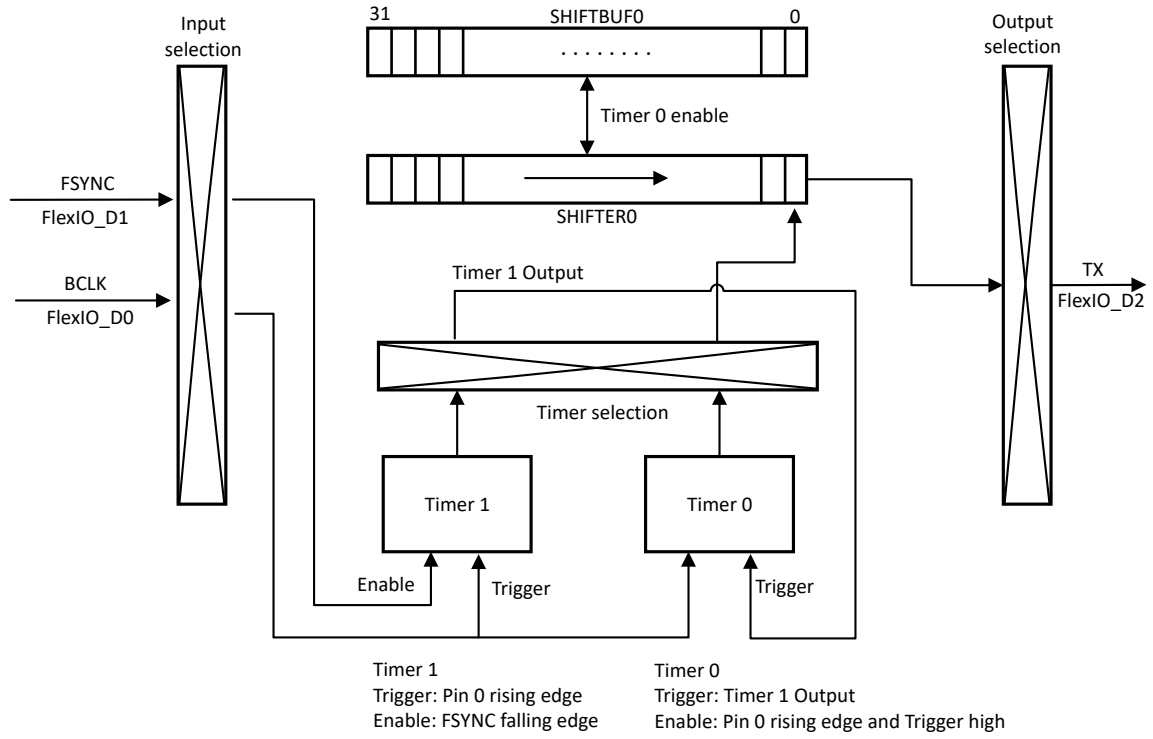

Figure 4. FlexIO internal connection

3.2 FlexIO configuration

This section describes the configuration of the FlexIO shifter and timer used in this application note.

3.2.1 Shifter configuration

There are 6 types of shifter modes configured by the SHIFTCTL register. Shifter0 is configured as a transmit mode and uses Timer0 on the rising edge of the shift clock to output data on the TX pin. When data has been loaded from the SHIFTBUF register into the SHIFTER, the shifter status flag is set and generates an enabled DMA request. Figure 5 shows the microarchitecture diagram of a shifter.

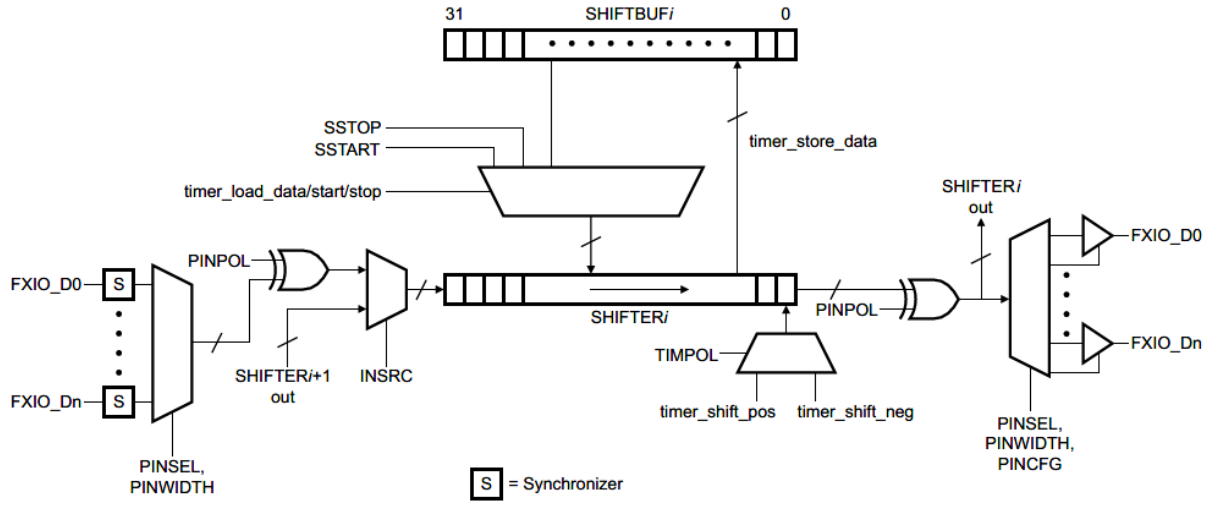

Figure 5. Shifter microarchitecture

The detailed configuration information of Shifter0 is shown in Table 3.

Table 3. Shifter0 configuration

Register	Value	Items	Configurations/description
SHIFTCTL0	SMOD = 2	Shifter mode	Transmit
	PINPOL = 0	Shifter pin polarity	Pin is active high
	PINSEL = 2	Shifter pin select	Pin 2 (FlexIO_D2)
	PINCFG = 3	Shifter pin configuration	Shifter pin output
	TIMPOL = 0	Timer polarity	Shift on posedge of Shift clock
	TIMSEL = 0	Timer select	Timer0 is used for controlling the logic/shift register and generating the Shift clock.
SHIFTCFG0	**SSTART = 0**	**Shifter start**	**Transmitter loads data on enable**
	SSTOP = 0	Shifter stop	Stop bit disabled for transmitter/receiver/match store
	INSRC = 0	Input source	Selects the PIN as the input source for the shifter.

Note: *The bolded bit field values in Table 3 should be different when the I2S interface, emulated by FlexIO, is used as the I2S master.*

3.2.2 Timer configuration

Timer1 detects the falling edge of FSYNC (the start of a new frame) and asserts the output until the rising edge of BCLK and disables it on the timer compare event. Timer0 is configured to enable on the rising edge of BLCK with the Timer1 trigger high (Timer1 output high) and disable on a compare event. The clock state of Timer0 and Timer1 is initialized to be logic 1. In addition, Timer1 is configured to a 16-bit counter and uses the trigger input as a decrement. The BCLK pin input is the trigger. The compare value of Timer1 is used to control the length of the frame. Timer0 is also configured to a 16-bit counter and uses the BCLK pin input as a decrement. The compare value of Timer0 is used to control the word length.

Table 4. Timer1 configuration

Register	Value	Items	Configurations/description
TIMCTL1	TIMOD = 3	Timer mode	Single 16-bit counter mode

Table 4. Timer1 configuration*...continued*

Register	Value	Items	Configurations/description
	PINPOL = 1	Timer pin polarity	Pin is active low
	PINSEL = 0x01	Timer pin select	Pin 1 (FlexIO_D1)
	PINCFG = 0	**Timer pin configuration**	**Timer pin output disabled**
	TRGSRC = 1	**Trigger source**	**Internal trigger selected**
	TRGPOL = 0	Trigger polarity	Trigger active high
	TRGSEL = 0x00	**Trigger select**	**Select pin 0 as internal trigger**
TIMCFG1	TSTART = 0	Timer start bit	Start bit disabled
	TSTOP = 0	Timer stop bit	Stop bit disabled
	TIMENA = 4	**Timer enable**	**Timer enabled on pin rising edge**
	TIMDIS = 2	**Timer disable**	**Timer disabled on timer compare**
	TIMRST = 0	Timer reset	Timer never reset
	TIMDEC = 3	**Timer decrement**	**Decrement counter on trigger input (both edges), shift clock equals trigger clock**
	TIMOUT =0	Timer output	Timer output is logic 1 when enabled and not affected by timer reset

Table 5. Timer0 configuration

Register	Value	Items	Configurations/description
TIMCTL0	**TIMOD = 3**	**Timer mode**	**Single 16-bit counter mode**
	PINPOL = 0	Timer pin polarity	Pin is active high
	PINSEL = 0x00	Timer pin select	Pin 0 (FlexIO_D0)
	PINCFG = 0	**Timer pin configuration**	**Timer pin output disabled**
	TRGSRC = 1	Trigger source	Internal trigger selected
	TRGPOL = 0	**Trigger polarity**	**Trigger active high**
	TRGSEL = 0x07	**Trigger select**	**Timer 1 trigger output**
TIMCFG0	**TSTART = 0**	**Timer start bit**	**Start bit disabled**
	TSTOP = 0	Timer stop bit	Stop bit disabled
	TIMENA = 5	**Timer enable**	**Timer enabled on pin rising edge and trigger high**
	TIMDIS = 3	**Timer disable**	**Timer disabled on timer compare**
	TIMRST = 0	Timer reset	Timer never reset
	TIMDEC = 2	**Timer decrement**	**Decrement counter on pin input (both edges), Shift clock equals pin input**
	TIMOUT =0	Timer output	Timer output is logic 1 when enabled and not affected by timer reset

Note: *The bolded bit field values in Table 4 and Table 5 should be different when the I2S interface, emulated by FlexIO, is used as the I2S master. The pin and trigger level and edges specified in tables 3, 4, and 5 refer to the*

signal state after being modified by the settings of TIMCTLn[PINPOL] and TIMCTLn[TRGPOL]. For example, "trigger low" means that a trigger is actually at logic level 1 when TIMCTLn[TRGPOL] is 1 (active low). The "timer 1 enabled on pin rising edge" setting means that it is enabled on the FSYNC (pin1) falling edge, because FSYNC is active low.

The code snippets for configuring the shifter and timer in this application note are as follows:

```
/* Set flexio i2s pin, shifter and timer */
 s_base.bclkPinIndex = BCLK_PIN;
 s_base.fsPinIndex = FRAME_SYNC_PIN;
 s_base.txPinIndex = TX_DATA_PIN;
 s_base.txShifterIndex = 0;
 s_base.bclkTimerIndex = 0;
 s_base.fsTimerIndex = 1;
 s_base.flexioBase = DEMO_FLEXIO_BASE;
 /*
 * config.enableI2S = true;
 */
 FLEXIO_I2S_GetDefaultConfig(&config);
 config.masterSlave = kFLEXIO_I2S_Slave;
 FLEXIO_I2S_Init(&s_base, &config);
 /* Configure the audio format */
 format.bitWidth = DEMO_AUDIO_BIT_WIDTH;
 format.sampleRate_Hz = DEMO_AUDIO_SAMPLE_RATE;
 FLEXIO_I2S_TransferTxCreateHandleEDMA(&s_base, &txHandle, TxCallback, NULL,
 &txDmaHandle);
 FLEXIO_I2S_TransferSetFormatEDMA(&s_base, &txHandle, &format, 0);
```

For more detailed code configuration, see the code in the attachment. FlexIO shifters and timers are configured in the *FLEXIO_I2S_Init()* function.

3.3 Codec configuration

In this application note, the audio codec on the LPCXpresso55S69 board is the I2S master and provides the BCLK and FSYNC. The LPI2C2 of MCXA156 is used to configure the codec as the I2S master. The configuration code is as follows:

```
wm8904_config_t wm8904Config = {
 .i2cConfig = {.codecI2CInstance =
 BOARD_CODEC_I2C_INSTANCE, .codecI2CSourceClock = BOARD_CODEC_I2C_CLOCK_FREQ},
 .recordSource = kWM8904_RecordSourceLineInput,
 .recordChannelLeft = kWM8904_RecordChannelLeft2,
 .recordChannelRight = kWM8904_RecordChannelRight2,
 .playSource = kWM8904_PlaySourceDAC,
 .slaveAddress = WM8904_I2C_ADDRESS,
 .protocol = kWM8904_ProtocolI2S,
 .format = {.sampleRate = kWM8904_SampleRate48kHz, .bitWidth =
 kWM8904_BitWidth16},
 .mclk_HZ = DEMO_I2S_MASTER_CLOCK_FREQUENCY,
 .master = true,
};
codec_config_t boardCodecConfig = {.codecDevType =
 kCODEC_WM8904, .codecDevConfig = &wm8904Config};
 if (CODEC_Init(&codecHandle, &boardCodecConfig) != kStatus_Success)
 {
   assert(false);
 }
```

```
    if (CODEC_SetVolume(&codecHandle, kCODEC_PlayChannelHeadphoneLeft |
  kCODEC_PlayChannelHeadphoneRight, 0x0020) != kStatus_Success)
  {
    assert(false);
  }
```

Note: *See the attachment for the complete code.*

3.4 LPC55S69 firmware development

In addition to configuring the codec as the I2S master, configure the LPC55S69 to generate MCLK. In the attached `lpc55s69_firmware` folder, there is firmware (`lpc55s69_i2s_mclk.bin`) that can be used to configure the LPC55S69 to generate MCLK. You can use the ISP UART mode to download this firmware to the LPC55S69. This firmware is developed based on the `i2s_edma_transfer` example in LPCXpresso55S9 SDK v2.16. In the original `i2s_dma_transfer` example, `LPC55S69 P1_31` is configured as the MCLK pin.

```
const uint32_t port1_pin31_config = (/* Pin is configured as MCLK */
                                      IOCON_PIO_FUNC1 |
                                      /* No addition pin function */
                                      IOCON_PIO_MODE_INACT |
                                      /* Standard mode, output slew rate
                                      control is enabled */
                                      IOCON_PIO_SLEW_STANDARD |
                                      /* Input function is not inverted */
                                      IOCON_PIO_INV_DI |
                                      /* Enables digital function */
                                      IOCON_PIO_DIGITAL_EN |
                                      /* Open drain is disabled */
                                      IOCON_PIO_OPENDRAIN_DI);
/* PORT1 PIN31 (coords: 91) is configured as MCLK */
IOCON_PinMuxSet(IOCON, 1U, 31U, port1_pin31_config);
```

In addition, disable the I2S function of the LPC55S69 and remove the configuration code of the LPC55S69 I2S and codec. When running this firmware, the LPC55S69 only provides the MCLK to the codec.

3.5 Test

This section describes how to test the function of emulating I2S with FlexIO on the FRDM-MCXA156 and LPCXpresso55S69 boards.

3.5.1 Downloading firmware to LPC55S69

The `lpc55s69_i2s_mclk.bin` and `blhost.exe` files are in the `lpc55s69_firmware` folder of the attachment, as shown in Figure 6. Use the ISP command to download firmware to the LPC55S69.

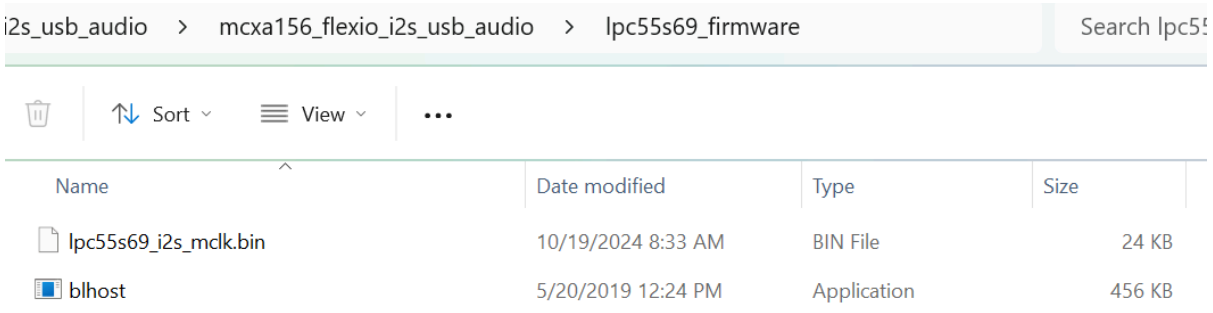

Figure 6. lpc55s69_firmware folder

Download the firmware using the ISP UART interface. Press the ISP button S1 on the LPCXpresso55S69 board and then connect P6 to the PC using a micro-USB cable to power on the LPC55S69 and put it into the ISP mode. Open the "CMD" window on the PC and change the path to the `lpc55s69_firmware` folder. Enter the following ISP command to download the firmware to the LPC55S69:

```
blhost.exe -p COM25 flash-erase-all
blhost.exe -p COM25 write-memory 0x00 lpc55s69_i2s_mclk.bin
```

Note: *The serial port number (COMxx) is the LPC-Link2 serial port number recognized by the PC. It may be different on a different PC.*

Run the LPC55S69 firmware. Press the reset button S4 on the LPCXpresso55S69 board to run the firmware. The LPC55S69 provides the MCLK to the codec. There is a handshake before the LPC55S69 generates the MCLK.

The P1_28 pin of the LPC55S69 is used to implement the handshake with the MCXA156, which is configured as an input pin and connected to the P0_20 pin of the MCXA156. After configuring the codec, the MCXA156 outputs a high level on P0_20 to put P1_28 to a high level. The LPC55S69 starts outputting the MCLK after detecting that P1_28 is at a high level.

3.5.2 Emulating I2S on MCXA156 with FlexIO

Compile the "mcxa156_flexio_i2s_usb_audio" project in the attachment and download it to FRDM-MCXA156. In this project, the MCXA156 is enumerated by the PC as a USB speaker device, as shown in Figure 7.

Figure 7. USB audio speaker

You can play any audio file on your PC and select "USB AUDIO DEMO" as the playback device. The PC sends the audio data to the MCXA156 through the USB interface. The FlexIO emulated I2S interface then sends the audio data to the codec for playback. After receiving the audio data, the codec plays it through the J2 audio output jack on the LPCXpresso55S69 board. You can connect 3.5-mm headphones to the J2 jack to hear the audio.

Note: *Run LPC55S69 firmware before running the MCXA156 program to complete the handshake.*

The I2S signals emulated by FlexIO are shown in Figure 8.

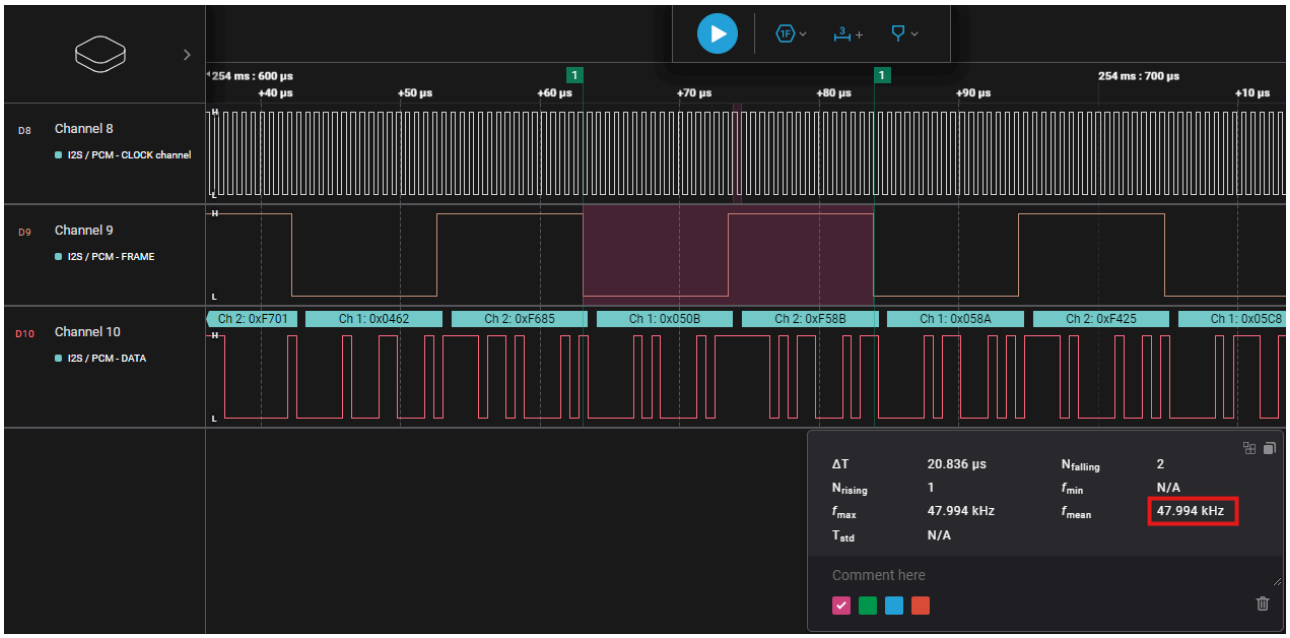

Figure 8. I2S signal emulated by FlexIO

4 Conclusion

This application note describes how to use the FlexIO module on MCXA156 to emulate the standard I2S interface and perform a functional test using the codec on the LPCXpresso55S69 board. The implementation of a FlexIO emulating I2S enables the MCXA156 to be applied in low-cost audio scenarios, such as low-cost USB audio speakers, even though the MCXA156 does not have any I2S/SAI interfaces.

5 References

1. *Emulating I2S Bus Master with FlexIO Module* (document AN12644)
2. *Emulating I2S Bus with the FlexIO on RT1010* (document AN12758)
3. MCXA156 RM
4. LPC55S69 UM

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:

Copyright 2025 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES

OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN

ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Example code shown in this document has the following copyright and Apache-2.0 license:

Copyright 2024 NXP

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License.

You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

7 Revision history

Table 6. Revision history

Document ID	Release date	Description
AN14527 v.1.0	09 January 2025	• Initial version

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.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

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.

Contents