AN14794

Emulating I2S Bus with the FlexIO on MCXA366

Rev. 1.0 — 19 December 2025 Application note

Document information

Information	Content
Keywords	AN14794, FlexIO, MCXA366, I2S Bus, audio, USB audio
Abstract	This application note describes how to use the FlexIO module to emulate the I2S interface on MCXA366.

1 Introduction

This application note describes how to use the FlexIO module to emulate the I2S interface on MCXA366. 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 MCXA366 processor is based on the Arm Cortex-M33 platform with a CPU clock of up to 240 MHz. It does not have an I2S/SAI interface. Therefore, the function of emulating the I2S interface with FlexIO is important. It can enable the MCXA366 to achieve some audio applications, such as a low-cost USB audio speaker. This application note describes how to use FlexIO to emulate an I2S master based on the FRDM-MCXA366 board. The FRDM-MCXA366 board has no audio codec, therefore an external audio codec must be connected to test the audio function. The PmodI2S2 board used in this document is a Pmod interface board, which features a Cirrus CS5343 multi-bit audio A/D converter and a Cirrus CS4344 stereo D/A converter. Each of them are connected to an audio jack.

The FRDM-MCXA366 board also has a Pmod interface. This interface has suitable FlexIO pins used to generate the clock and data lines required by I2S. Figure 1 shows the system block diagram.

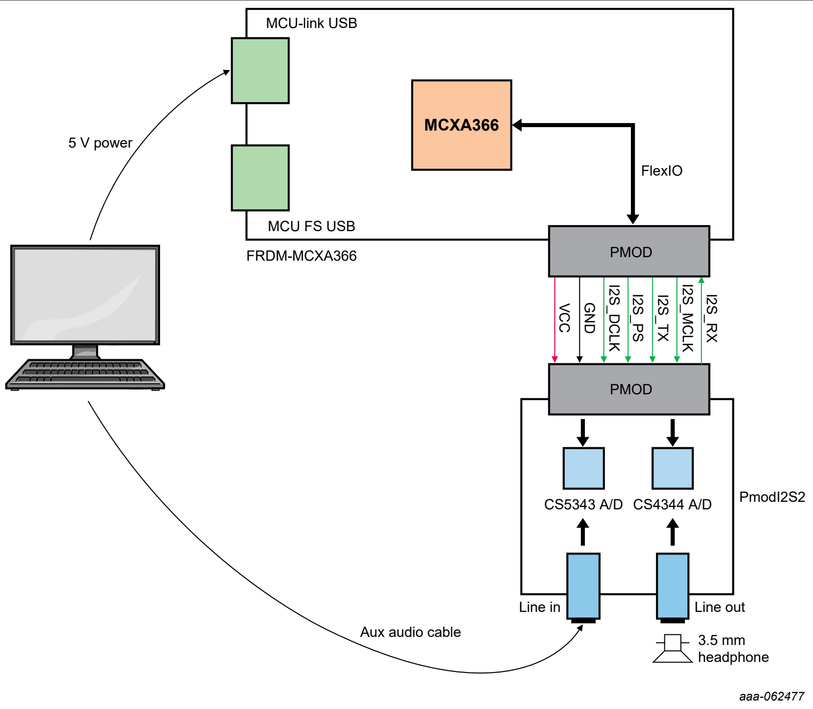

Figure 1. System block diagram

In this application note, the I2S interface emulated by FlexIO is used as an I2S master. The audio format used in this application note is as follows:

- Transmit mode: Classic I2S mode
- Frame word count: 2
- Word length: 16
- BCLK frequency: 1.536 MHz
- FSYNC frequency: 48 kHz
- MCLK frequency: 12.288 MHz

The I2S interface emulated by FlexIO supports not only the classic I2S format but also other audio formats, including left-justified, right-justified, and PCM. Table 1 compares 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 MCXA366	SAI on MCXN947	Comments
I2S	Classic 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. To trigger FSYNC, another timer is required.
	Mode B	Yes	Yes	
Data alignment	First bit shift configuration	No	Yes	
	LSB or MSB first	Yes	Yes	
Word length	8-bit to 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 x 32 bits	8 x 32 bits	
	FIFO packing	No	Yes	
	FIFO combine	No	Yes	
Synchronous mode	Transmitter and receiver sync	Yes	Yes	
Frame sync width	1-bit to 32-bit clock	Yes	Yes	
Clock limitation		Yes	No	In master mode, the maximum baud rate is FlexIO clock frequency/4. For slave mode, the maximum baud rate is FlexIO clock frequency/6.

2 Hardware requirement

The hardware required is listed below:

- FRDM-MCXA366 board
- PmodI2S2 board
- One Type-C USB cable
- 3.5 mm headphone
- 3.5 mm audio cable

Table 2 shows the connection between FRDM-MCXA366 and PmodI2S2 boards.

Table 2. Connection between FRDM-MCXA366 and PmodI2S2

Power and signals	FRDM-MCXA366	PmodI2S2
I2S_TX_MCLK	J7_1/P1_6/FlexIO_D14	J1_1/DA MCLK
I2S_TX_FS	J7_3/P1_0/FlexIO_D8	J1_2/DA LRCK
I2S_TX_BCLK	J7_5/P1_2/FlexIO_D10	J1_3/DA SCLK

Table 2. Connection between FRDM-MCXA366 and PmodI2S2*...continued*

Power and signals	FRDM-MCXA366	PmodI2S2
I2S_TX_DATA	J7_7/P1_1/FlexIO_D9	J1_4/DA SDIN
GND	J7_9/GND	J1_5/GND
VCC	J7_11/VDD_BOARD	J1_6/VCC
I2S_RX_MCLK	J7_2/P1_6	J1_7/AD MCLK
I2S_RX_FS	J7_4/P1_0	J1_8/AD LRCK
I2S_RX_BCLK	J7_6/P1_2	J1_9/AD SCLK
I2S_RX_DATA	J7_8/P1_1/FlexIO_D28	J1_10/AD SDOUT
GND	J7_10/GND	J1_11/GND
VCC	J7_12/VDD_BOARD	J1_12/VCC

In this application note, three FlexIO timers are used to generate I2S_TX_MCLK, I2S_TX_BCLK, and I2S_TX_FS clocks. Due to the FlexIO module on the MCXA366 supporting a maximum of four timers, it is not feasible to generate the I2S_RX_MCLK, I2S_RX_BCLK, and I2S_RX_FS clocks. However, the RX channel can share clock signals with the TX channel, allowing us to connect them on the FRDM-MCXA366 board directly. Table 3 shows the connection.

Table 3. Clock sharing connection on FRDM-MCXA366 board

TX channel		RX channel	
I2S_TX_MCLK	J2_11/P1_6/FLEXIO_D14	I2S_RX_MCLK	J2_7/P1_4/FLEXIO_D12
I2S_TX_FS	J4_10/P1_0/FLEXIO_D8	I2S_RX_FS	J2_19/P1_5/FLEXIO_D13
I2S_TX_BCLK	J6_5/P1_2/FLEXIO_D10	I2S_RX_BCLK	J8_3/P3_27/FLEXIO_D27

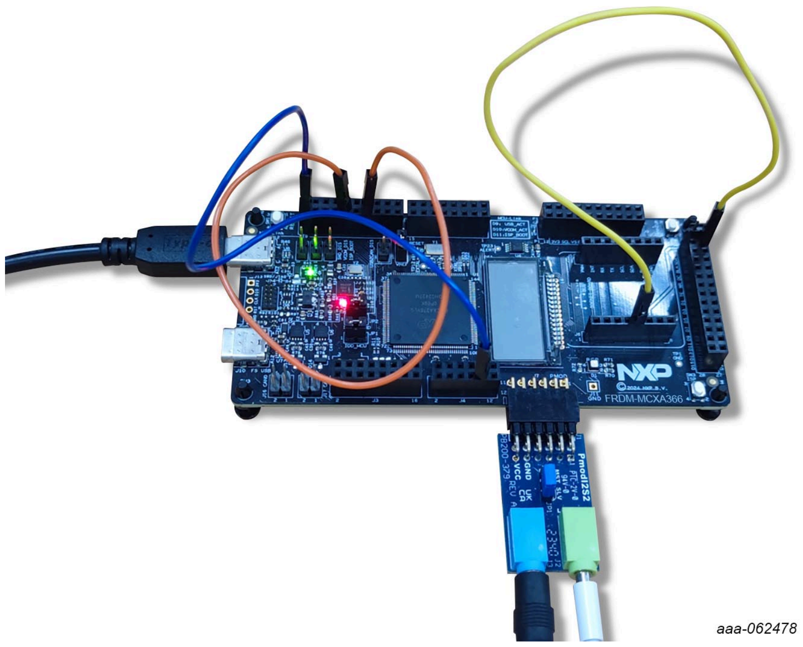

Figure 2. FRDM-MCXA366 board and PmodI2S2 board

3 Implementation of FlexIO emulating I2S interface

This chapter introduces the implementation of emulating I2S interface with FlexIO, including FlexIO introduction, FlexIO configuration, and testing process.

3.1 FlexIO introduction

FlexIO is a highly configurable module with the following features:

- Supports emulation of a wide range of serial/parallel communication protocols, such as UART, I2C, SPI, I2S.
- Supports various triggers, reset, enable, and disable conditions with flexible 16-bit timers.
- Supports programmable logic blocks allowing the implementation of digital logic functions on-chip and configurable interaction of internal and external modules.
- Supports programmable state machine offloading the 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. Figure 3 shows that the 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.

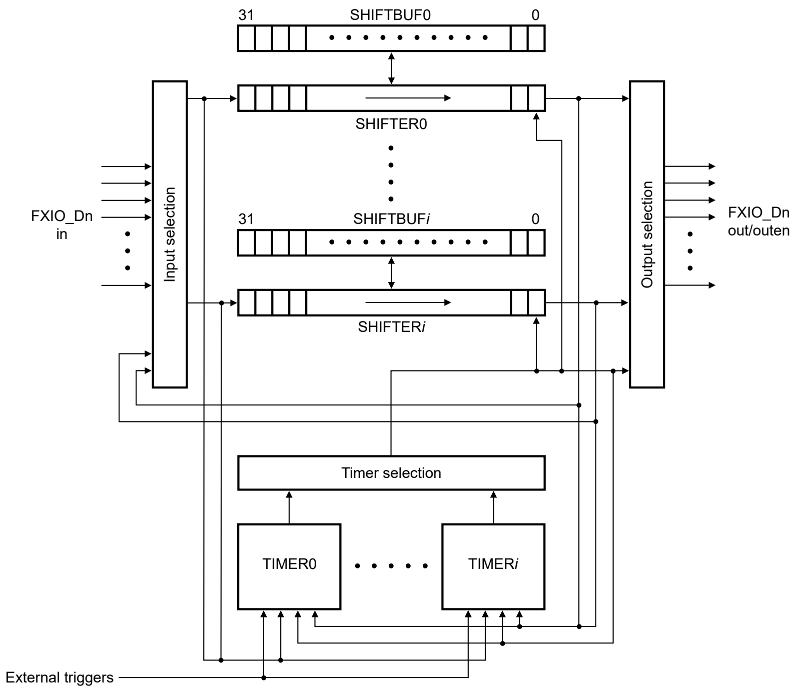

Figure 3. FlexIO block diagram

On MCXA366, FlexIO has 4 shifters and 4 timers, and support a total of 32 FlexIO pins.

In this application note:

- 5 FlexIO pins (FlexIO_D8, FlexIO_D9, FlexIO_D10, FlexIO_D14, FlexIO_D28) are used to emulate the I2S_TX_FS pin, I2S_TX_DATA pin, I2S_TX_BCLK, I2S_TX_MCLK and I2S_RX_DATA pin respectively.
- Shifter0 is used for I2S_TX_DATA pin.
- Shifter1 is used for I2S_RX_DATA pin.
- Timer0, timer1, and timer2 are used for I2S_TX_BCLK and I2S_TX_FS pin and I2S_TX_MCLK pin.

Figure 4 shows the internal connection of the FlexIO emulating I2S interface.

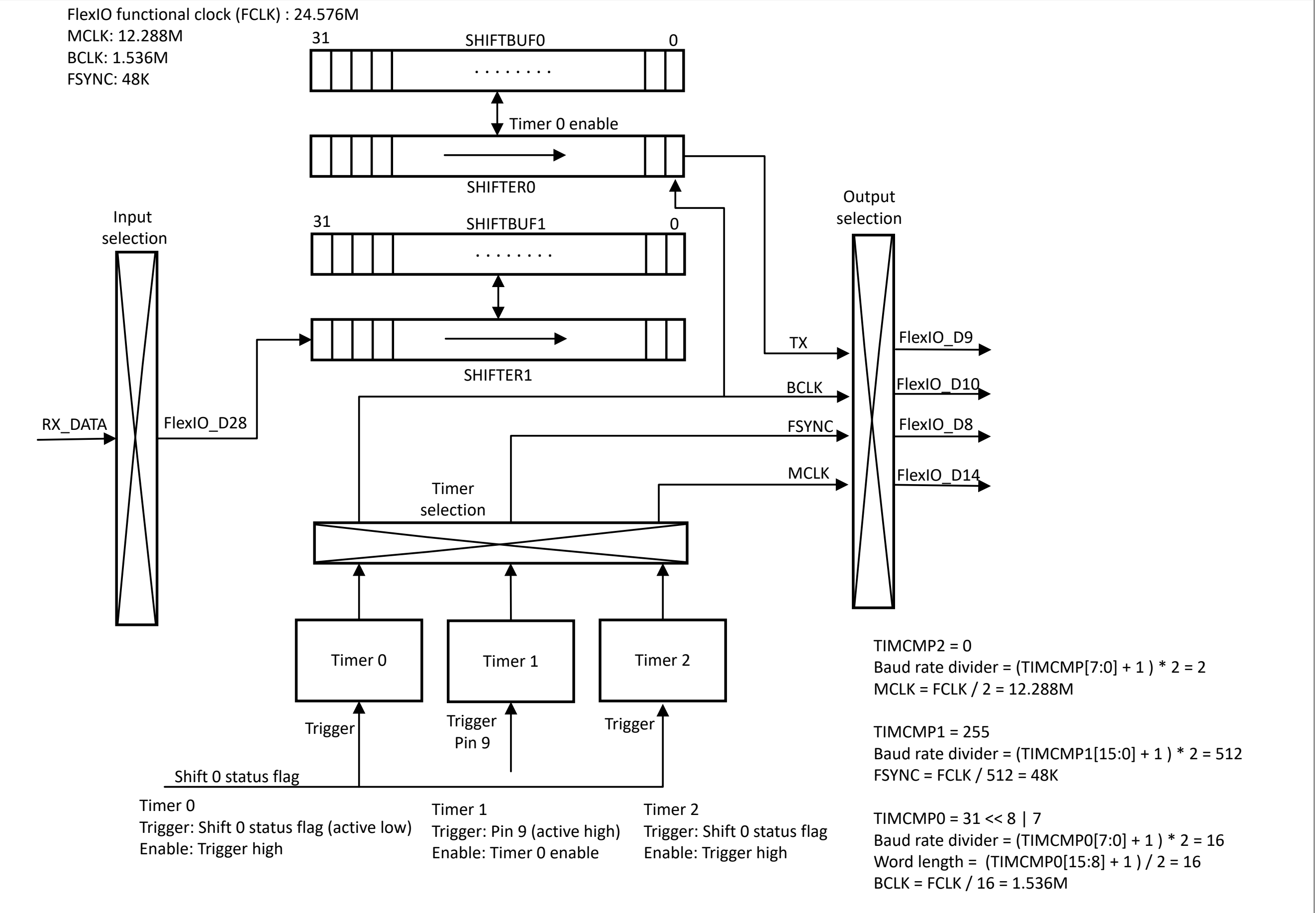

Figure 4. FlexIO internal connection

3.2 FlexIO configuration

This section describes the configuration of the FlexIO shifter and timer.

3.2.1 Shifter configuration

The SHIFTCTL register configure six types of shifter modes. Shifter0 is configured as a Transmit mode and uses timer0 on the rising edge of the shift clock, to output data on 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 the shifter.

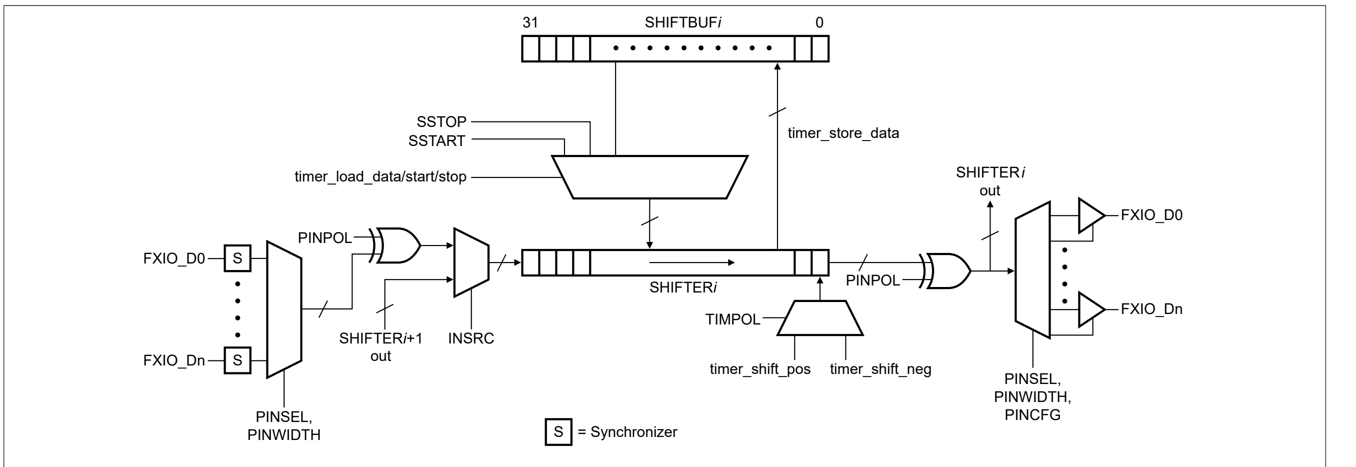

Figure 5. Shifter microarchitecture

The detailed configuration information of shifter0 is shown in Table 4.

Table 4. Shifter0 configuration

Register	Value	Items	Configurations/description
SHIFTCTL0	SMOD = 2	Shifter mode	Transmit
	PINPOL = 0	Shifter pin polarity	Pin is active high
	PINSEL = 9	Shifter pin select	Pin 9 (FlexIO_D9)
	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 = 1**	**Shifter start**	**Transmitter loads data on first shift**
	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.

Shifter1 is used as the receiver and its initial configurations are described in Table 5.

Table 5. Shifter1 configuration

Register	Value	Items	Configurations/description
SHIFTCTL1	SMOD = 1	Shifter mode	Receive
	PINPOL = 0	Shifter pin polarity	Pin is active high
	PINSEL = 28	Shifter pin select	Pin 28 (FlexIO_D28)
	PINCFG = 0	Shifter pin configuration	Shifter pin output disabled
	TIMPOL = 1	Timer polarity	Shift on negedge of Shift clock
	TIMSEL = 0	Timer select	Timer0 is used for controlling the logic/shift register and generating the shift clock.
SHIFTCFG1	SSTART = 0	Shifter start	Transmitter loads data on enabled
	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 bit field values highlighted in bold in Table 4 must be different when the I2S interface emulated by FlexIO is used as the I2S slave.*

3.2.2 Timer configuration

Timer0 is used to generate I2S TX bit clock and it is in 8-bit baud counter mode. In this mode, the 16-bit counter is divided into two 8-bit counters. The lower 8 bits are used to configure the baud rate of the shift clock as (CMP[7:0] + 1) x 2. The upper 8 bits are used to configure the number of bits in each word as (CMP[15:8] + 1)/2. When the lower 8 bits count down to zero, the timer output toggles, and the bits are reloaded from the compare register. The source of timer 0 decrement is the FLEXIO functional clock. In this document, the functional clock of FlexIO is 24.576 MHz, and the audio format is 48 KHz/16 bit stereo. Therefore, the I2S TX bit clock is 1.536 MHz and the baud rate divider = 24.576/1.536 = 16 = (CMP[7:0] +1) x 2, CMP[7:0] = 7.

The number of bits in each word = 16 = (CMP[15:8] + 1)/2, CMP[15:8] = 31. The trigger signal of Timer0 is the shifter0 status flag, and the trigger is active low. As Shifter0 is in Transmit mode, the state flag is set when SHIFTBUF data is transferred to the shifter (SHIFTBUF is empty) or when SHIFTCTL0[SMOD] is initially configured as 010b (Transmit mode). The status flag is cleared when you write the SHIFTBUF register. Timer0 is enabled when the trigger is high, so it is enabled when writing SHIFTBUF register. For detailed configuration of Timer0, see Table 6.

Table 6. Timer0 configuration

Register	Value	Items	Configurations/description
TIMCTL0	**TIMOD = 1**	**Timer mode**	**Dual 8-bit counters baud/bit mode**
	PINPOL = 0	**Timer pin polarity**	**Pin is active low**
	PINSEL = 10	Timer pin select	Pin 10 (FlexIO_D10)
	PINCFG = 3	**Timer pin configuration**	**Pin output**
	TRGSRC = 1	Trigger source	Internal trigger selected
	TRGPOL = 1	**Trigger polarity**	**Trigger active low**
	TRGSEL = 0x01	**Trigger select**	**Shifter0 flag**
TIMCFG0	**TSTART = 1**	**Timer start bit**	**Start bit enabled**
	TSTOP = 0	Timer stop bit	Stop bit disabled
	TIMENA = 2	**Timer enable**	**Timer enabled on trigger high**
	TIMDIS = 0	**Timer disable**	**Timer never disabled**
	TIMRST = 0	Timer reset	Timer never resets
	TIMDEC = 0	**Timer decrement**	**Decrement counter on FlexIO clock** **Shift clock equals timer output**
	TIMOUT =0	Timer output	Timer output is logic 1 when enabled and timer reset does not affect it

Timer1 is used to generate I2S TX FS, and it is in 16-bit counter mode. The source of Timer1 decrement is the FLEXIO functional clock. In this document, the I2S TX FS is 48 kHz. Therefore, the baud rate divider = 24.576 MHz/48 kHz = 512 = (CMP[15:0] +1) x 2, CMP[15:0] = 255. Timer1 is enabled when Timer0 is enabled. For detailed configuration of Timer1, see Table 7.

Table 7. Timer1 configuration

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

Table 7. Timer1 configuration*...continued*

Register	Value	Items	Configurations/description
	PINPOL = 1	Timer pin polarity	Pin is active low
	PINSEL = 0x01	Timer pin select	Pin 8 (FlexIO_D8)
	PINCFG = 3	**Timer pin configuration**	**Timer pin output**
	TRGSRC = 0	**Trigger source**	**External trigger selected**
	TRGPOL = 0	Trigger polarity	Trigger active high
	TRGSEL = 0x00	**Trigger select**	**Select pin 9 as the internal trigger**
TIMCFG1	TSTART = 0	Timer start bit	Start bit disabled
	TSTOP = 0	Timer stop bit	Stop bit disabled
	TIMENA = 1	**Timer enable**	**Timer enabled on Timer N-1 enable**
	TIMDIS = 0	**Timer disable**	**Timer never disabled**
	TIMRST = 0	Timer reset	Timer never resets
	TIMDEC = 0	**Timer decrement**	**Decrement counter on FlexIO clock**
	TIMOUT =0	Timer output	Timer output is logic 1 when enabled and the timer reset does not affect it

Timer2 is used to generate I2S TX MCLK and it is in 16-bit counter mode. The source of Timer2 decrement is the FLEXIO functional clock. In this document, I2S TX MCLK is 12.288 MHz. Therefore, the baud rate divider = 2 = (CMP[15:0] +1) x 2, CMP[15:0] = 0. The trigger signal of Timer2 is also a shifter0 status flag, and it is enabled when the trigger is high. For more detailed configuration of Timer2, see Table 8.

Table 8. Timer2 configuration

Register	Value	Items	Configurations/description
TIMCTL2	TIMOD = 1	Timer mode	Single 16-bit counter mode.
	PINPOL = 0	Timer pin polarity	Pin is active high
	PINSEL = 14	Timer pin select	Pin 14 (FlexIO_D14)
	PINCFG = 3	**Timer pin configuration**	**Pin output**
	TRGSRC = 1	Trigger Source	Internal trigger selected
	TRGPOL = 1	**Trigger Polarity**	**Trigger active low**
	TRGSEL = 0x01	**Trigger Select**	**Shifter0 flag**
TIMCFG2	**TSTART = 1**	**Timer start bit**	**Start bit enabled**
	TSTOP = 0	Timer stop bit	Stop bit disabled
	TIMENA = 2	**Timer Enable**	**Timer enabled on trigger high**
	TIMDIS = 0	**Timer disable**	**Timer never disabled**
	TIMRST = 0	Timer reset	Timer never reset
	TIMDEC = 0	**Timer decrement**	**Decrement counter on FlexIO clock** **Shift clock equals timer output**
	TIMOUT =0	Timer output	Timer output is logic 1 when enabled and the timer reset does not affect it

Note: *The bit field values highlighted in bold in Table 6, Table 7, and Table 8 must be different when the I2S interface emulated by FlexIO is used as the I2S slave. The pin and trigger level and edges specified in Table 6, Table 7, and Table 8, 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 if TIMCTLn[TRGPOL] is 1 (active low).*

The code snippets for configuring shifter and timer in this application note are shown below:

```
/* 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.rxPinIndex     = RX_DATA_PIN;
    s_base.txShifterIndex = 0;
    s_base.rxShifterIndex = 1;
    s_base.bclkTimerIndex = 0;
    s_base.fsTimerIndex   = 1;
    s_base.flexioBase     = DEMO_FLEXIO_BASE;

    dma_request_source_tx = kDma0RequestMuxFlexIO0ShiftRegister0Request;
    dma_request_source_rx = kDma0RequestMuxFlexIO0ShiftRegister1Request;

    /*
     * config.enableI2S = true;
     */
    FLEXIO_I2S_GetDefaultConfig(&config);

    config.masterSlave = kFLEXIO_I2S_Master;
    config.bclkPinPolarity = kFLEXIO_PinActiveLow;
    FLEXIO_I2S_Init(&s_base, &config);
    FLEXIO_I2S_Init_FOR_MCLK(&s_base, &config);

    /* Configure the audio format */
    format.bitWidth      = DEMO_AUDIO_BIT_WIDTH;
    format.sampleRate_Hz = DEMO_AUDIO_SAMPLE_RATE;

    /* Configure EDMA channel for one shot transfer */
    EDMA_GetDefaultConfig(&userConfig);
    EDMA_Init(DEMO_DMA_BASEADDR, &userConfig);
    EDMA_CreateHandle(&txDmaHandle, DEMO_DMA_BASEADDR, DEMO_DMA_CHANNEL_TX);
    EDMA_CreateHandle(&rxDmaHandle, DEMO_DMA_BASEADDR, DEMO_DMA_CHANNEL_RX);

    EDMA_SetChannelMux(DMA0, DEMO_DMA_CHANNEL_TX, dma_request_source_tx);
    EDMA_SetChannelMux(DMA0, DEMO_DMA_CHANNEL_RX, dma_request_source_rx);

    FLEXIO_I2S_TransferTxCreateHandleEDMA(&s_base, &txHandle, tx_callback, NULL,
 &txDmaHandle);
    FLEXIO_I2S_TransferRxCreateHandleEDMA(&s_base, &rxHandle, rx_callback, NULL,
 &rxDmaHandle);

    FLEXIO_I2S_TransferSetFormatEDMA(&s_base, &txHandle, &format,
 CLOCK_GetFlexioClkFreq());
    FLEXIO_I2S_TransferSetFormatEDMA(&s_base, &rxHandle, &format,
 CLOCK_GetFlexioClkFreq());
```

For more detailed code configuration, refer to the code in the attached SW zip folder. FlexIO shifters and timers are configured in the *Audio_FlexioI2Sinit()* function.

3.3 Test

This section describes how to test the function of emulating I2S with FlexIO on FRDM-MCXA366 and PmodI2S2 board.

- Use a Type-C cable to connect J15 on FRDM-MCXA366 board to the PC.
- Compile the MCXA366_flexio_i2s project in the attachment and download it to FRDM-MCXA366 board.
- Use a 3.5 mm audio cable to connect the PC and the LINE IN jack on PmodI2S2 board.
- Connect a 3.5 mm headphone to the LINE OUT jack on the PmodI2S2 board.
- Short the I2S RX MCLK, BCLK, and FS to I2S TX MCLK, BCLK, and FS on FRDM-MCXA366 board.
  - MCLK: Short J2_11/P1_6/FLEXIO_D14 and J2_7/P1_4/FLEXIO_D12
  - BCLK: Short J6_5/P1_2/FLEXIO_D10 - J8_3/P3_27/FLEXIO_D27
  - FS: Short J4_10/P1_0/FLEXIO_D8 - J2_19/P1_5/FLEXIO_D13

You can play any audio file on your PC, which sends the audio data to the PmodI2S2 board via a 3.5 mm audio cable. The data is then sent to the MCXA366 through the I2S RX DATA pin emulated by FlexIO. MCXA366 processes the audio and transmits it back to the PmodI2S2 board via the I2S TX DATA pin emulated by FlexIO. Finally, you can hear the audio through the LINE OUT jack.

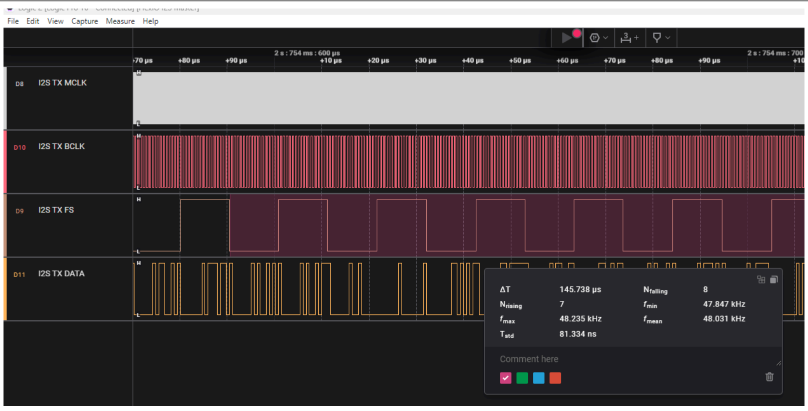

Figure 6. FlexIO emulated I2S waveform

4 Conclusion

This application note describes how to use the FlexIO module on MCXA366 to emulate the classic I2S interface, and test the audio TX/RX function with PmodI2S2 board. The implementation of FlexIO emulating I2S enables the MCXA366 to be applied in low-cost audio scenarios, such as low-cost USB audio speakers, even though the MCXA366 does not have I2S/SAI interfaces.

5 References

Table 9 lists the references used to supplement this document.

Table 9. References

Reference	Link/How to obtain
Emulating I2S Bus with the FlexIO on MCXA156 (document AN14527)	https://www.nxp.com/doc/AN14527
Emulating I2S Bus Master with FlexIO Module (document AN12644)	https://www.nxp.com/doc/AN12644

Table 9. References*...continued*

Reference	Link/How to obtain
Emulating I2S Bus with the FlexIO on RT1010 (document AN12758)	https://www.nxp.com/docs/en/application-note/AN12758.pdf

6 Note about the source code in the document

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

7 Revision history

Table 10 summarizes the revisions to this document.

Table 10. Revision history

Document ID	Release date	Description
AN14794 v.1.0	19 December 2025	Initial public release

Legal information

Definitions

Disclaimers

Trademarks

Contents

1 Introduction 2
2 Hardware requirement 3
3 Implementation of FlexIO emulating I2S interface 5
3.1 FlexIO introduction 5
3.2 FlexIO configuration 6
3.2.1 Shifter configuration 6
3.2.2 Timer configuration 8
3.3 Test 11
4 Conclusion 11
5 References 11
6 Note about the source code in the document 12
7 Revision history 12
Legal information 13

Date of release: 19 December 2025
Document identifier: AN14794