# AN12758

## Emulating I2S Bus with the FlexIO on RT1010

Rev. 0 — March 2020

Application Note

Contents

## 1 Introduction

This application note describes how to use the FlexIO module to simulate I$^2$S interface for audio transmission and reception. Using the FlexIO module can generate all the necessary I$^2$S bus signals, which can replace the traditional I2S/SAI peripherals to transfer audio data and save CPU resources.

The i.MX RT1010 processor is based on the Arm® Cortex®-M7 platform. It provides high CPU performance and best real-time response, and has rich peripheral devices. In order to verify the I2S peripherals emulated via FlexIO, a simple application was implemented on the RT1010 EVK board.

This application uses I$^2$S to receive the audio stream into the `sram` buffer to process, and then uses I$^2$S to transmit the audio stream to the audio playback device. This user case constitutes an audio digital loopback, and all audio data processing is implemented around the codec chip on the board.

## 2 Development platform

The use case of the FlexIO emulating I$^2$S interface is based on the RT1010 EVK board. Figure 1 shows the actual demo hardware.

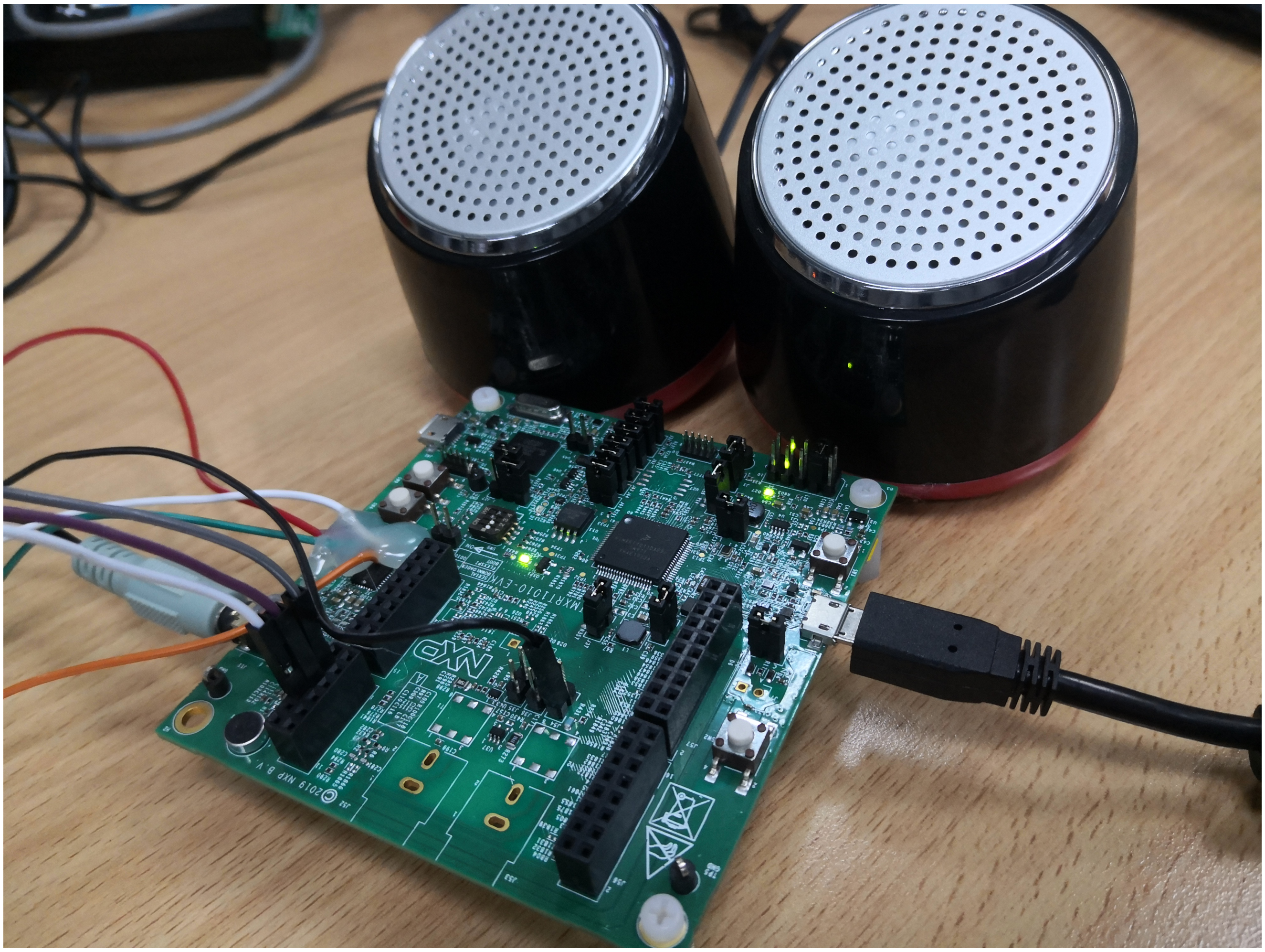

Figure 1. Actual demo hardware

To make the example work, check the following steps:

- Audio speakers plug into J11 port of RT1010 EVK board.
- Change the ISP switch (SW8) to (0b0010).
- Power on board with USB cable plugged to J9.
- J1-3 connect to J1-4.

RT1010 EVK board changes include:

- Remove R85, R87, R88, R20 on RT1010 EVK board.
- Table 1 describes the connections of FLEXIO pins to the pad close to U10.

Table 1. Connections of FLEXIO pins

| Pin name | Board location | Connection | Board location |
|---|---|---|---|
| RX_DATA | J54-2 | <------> | U10-16 |
| TX_DATA | J26-8 | <------> | U10-14 |
| SYNC | J26-6 | <------> | U10-13 |
| BCLK | J26-4 | <------> | U10-12 |

# 3 Implementation

This chapter describes some design points of using FlexIO to emulate I$^2$S applications, focusing on the configuration of FlexIO module.

## 3.1 User case system

The i.MX RT1010 provides FlexIO module to simulate the I$^2$S interface for this application. The WM8960 codec receives the audio signal from the MIC. RT1010 continuously receives the PCM data of codec. Then PCM data is transferred to codec for playback in real time. The architecture of the user case is as shown in Figure 2. The RT1010 initializes and configures WM8904 via the I$^2$C interface. The RT1010 is the slave device of I$^2$S and the WM8960 codec is the master device.

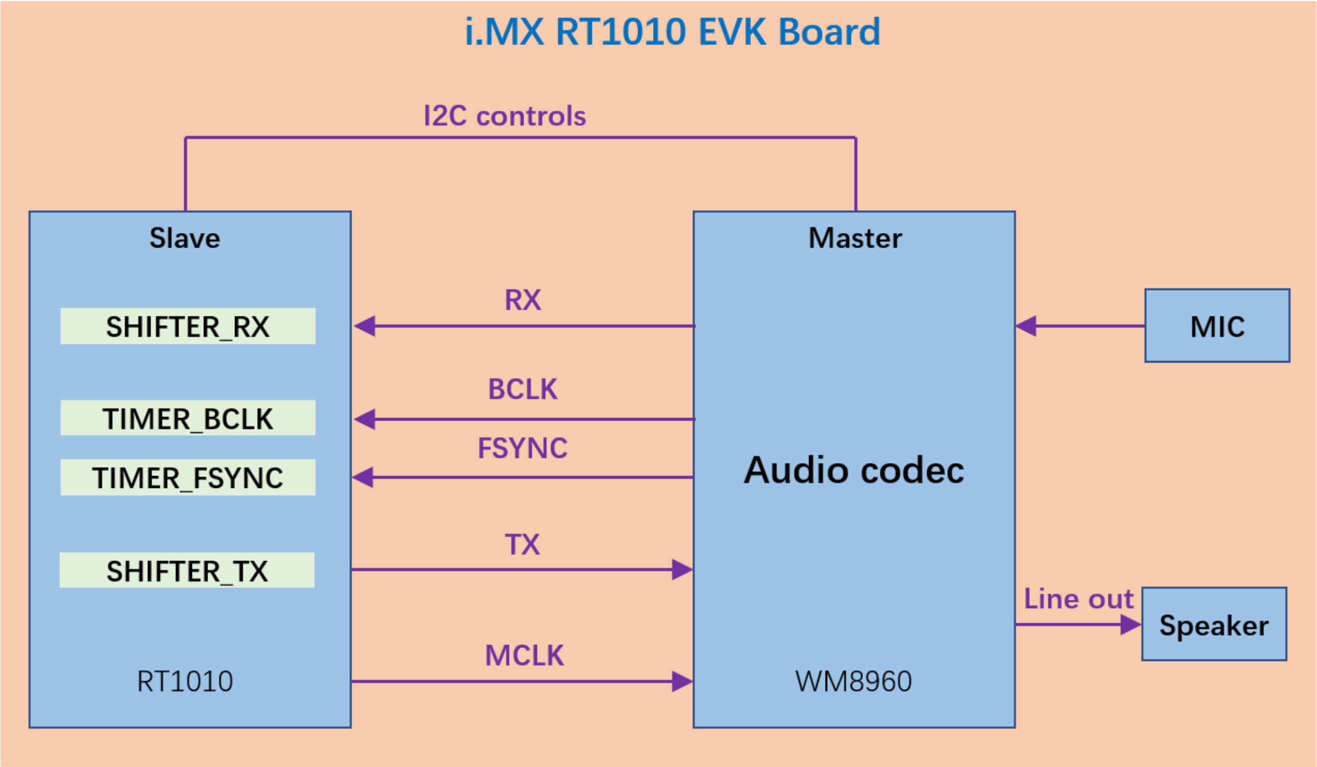

Figure 2. User case system architecture

## 3.2 Clock and codec configuration

In this application, the FlexIO clock is configured to 6.144 MHz. The MCLK and the sample rate of the I$^2$S emulated depends on codec and user requirements. There are two kinds of clock frequencies for I$^2$S sample rate. Table 2 describes typical I$^2$S sample frequencies. In this user case, the sample rate is configured to 16 kHz.

NOTE

Due to FlexIO synchronization delays, the output valid time for the serial output data is 2.5 times the FlexIO clock cycles when FlexIO is used to emulate I$^2$S slave and the maximum baud rate is divide by 6 of the FlexIO clock frequency.

Table 2. I$^2$S sample frequency

| Typical Sample Frequency (Hz) | Typical Sample Frequency (Hz) |
|---|---|
| 11025 | 8000 |
| 22050 | 16000 |
| 44100 | 24000 |
| — | 32000 |
| — | 48000 |

Generally, users should check the codec data sheet to calculate the clock frequency of MCLK. When WM8960 codec is the master device, MCLK can be provided by its own internal clock or external clock. In this application, MCLK is generated by RT1010 and the frequency is 6.144 MHz. In addition, the BCLK is also provided by WM8960 codec, so it needs to be configured according to the sample rate, as shown below.

- Transmit mode: Classic I$^2$S mode
- Frame word count: 2
- Word length: 32
- MCLK frequency: 6.144 MHz
- FSYNC frequency: 16 kHz

Therefore,So

- BCLK = Frame word count × Word length × FSYNC frequency = 2 × 32 × 16 kHz = 1.024 MHz
- The BCLK divider = MCLK frequency/Frame word count/Word length/FSYNC frequency = 6

The initialization and configuration of codec are done by communicating with the I$^2$C interface of RT1010. The code in #GUID-FD70F3BB-DD3F-40D2-A944-BB38DE273A5B/EXAMPLE_EXAMPLE1 shows the configuration of WM8960 codec.

```
wm8960_config_t wm8960Config = {
    .i2cConfig         = {.codecI2CInstance =
BOARD_CODEC_I2C_INSTANCE,.codecI2CSourceClock                           =
BOARD_CODEC_I2C_CLOCK_FREQ},
    .route               = kWM8960_RoutePlaybackandRecord,
    .rightInputSource    = kWM8960_InputDifferentialMicInput2,
    .playSource          = kWM8960_PlaySourceDAC,
    .slaveAddress        = WM8960_I2C_ADDR,
    .bus                 = kWM8960_BusI2S,
    .format              = {.mclk_HZ = 6144000U,
    .sampleRate          = kWM8960_AudioSampleRate16KHz,
    .bitWidth            = kWM8960_AudioBitWidth32bit},
    .master_slave        = true,
};
codec_config_t boardCodecConfig = {
        .codecDevType       = kCODEC_WM8960,
        .codecDevConfig     = &wm8960Config
};
/* Init codec */
CODEC_Init(&codecHandle, &boardCodecConfig);
```

## 3.3 FlexIO emulation

FlexIO is a highly configurable module with the following features:

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

On the i.MX RT1010, FLEXIO has a total of 27 pins. In this case, four FlexIO pins (FlexIO03, FlexIO21, FlexIO22, and FlexIO26) are used to emulate the RX pin, BCLK pin, FSYNC pin and TX pin of the I2S interface respectively. Figure 3 shows the internal connection of the FlexIO emulating I2S interface. The pins of **Timer0** and **Timer2** correspond to BCLK and FSYNC, and the pins of **SHIFTER0** and **SHIFTER2** correspond to TX and RX.

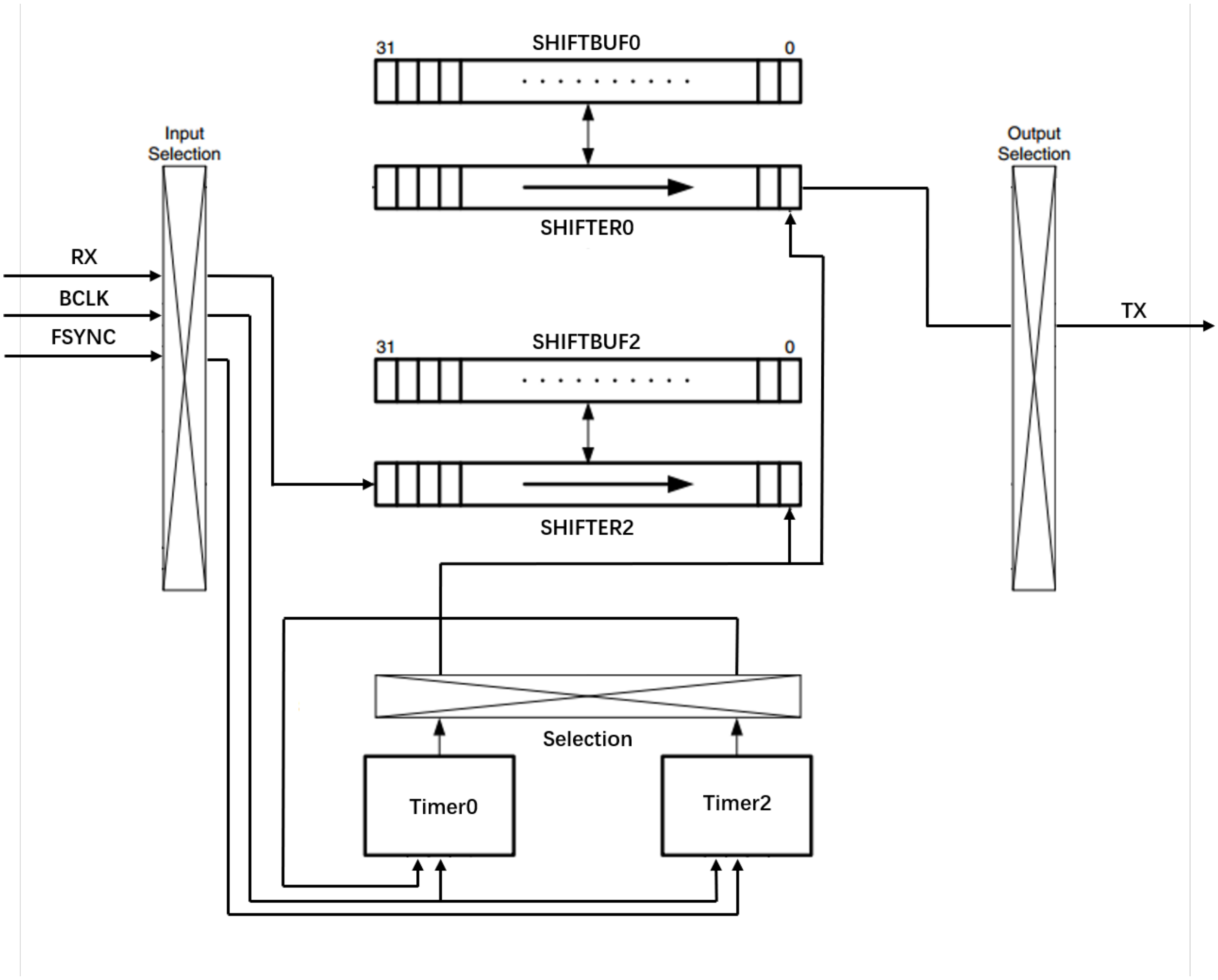

Figure 3. FlexIO internal connection diagram

There are six types of SHIFTER modes configured by the SHIFTCTL register. **SHIFTER0** is configured as transmit mode and uses **Timer0** on rising edge of shift clock to output data on TX pin. **SHIFTER2** is configured as receive mode and uses Timer0 on falling edge of shift clock to input data on RX pin. When data has been stored into the SHIFTBUF register from the SHIFTER or when data has been loaded from the SHIFTBUF register into the SHIFTER, the SHIFTER status flag will be set and generate an enabled DMA request. Figure 4 shows the microarchitecture dragram of SHIFTER.

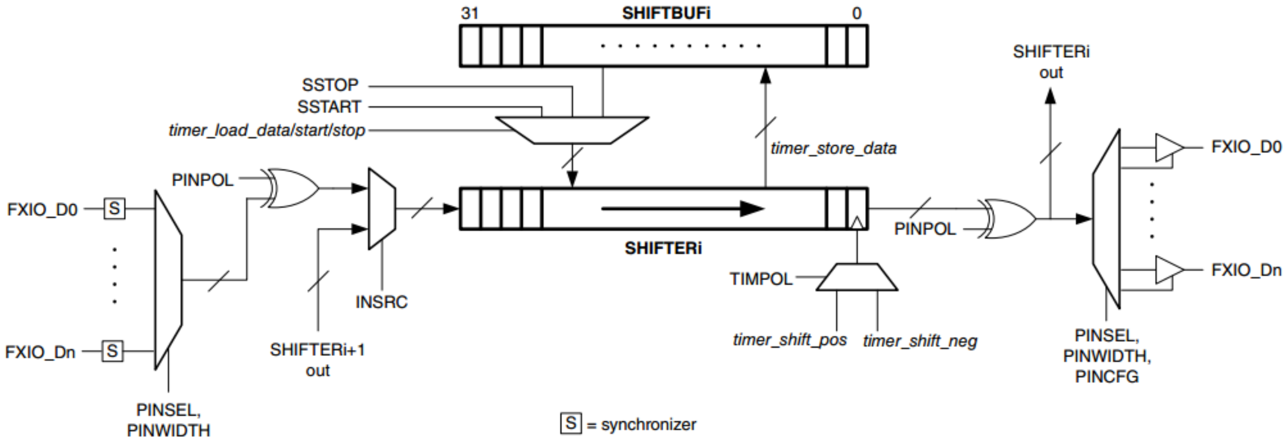

Figure 4. SHIFTER microarchitecture dragram

**Timer2** detects the rising edge of FSYNC to enable the timer and disable on bit clock trigger falling edge. **Timer0** is configured to enable on the rising edge of BLCK with **Timer2** trigger high and disable on compare event. The clock state of **Timer0** and **Timer2** is initialized to be logic 1. In addition, **Timer2** is configured to 16-bit counter and use FSYNC pin input as decrement, BCLK pin input as the trigger. The compare value of **Timer2** is set to 0. **Timer0** is also configured to 16-bit counter and use BCLK pin input as decrement. In this application, the compare value of **Timer2** is set to 127 (32 × 4 - 1) according to the 64 bits frame length.

The detailed register configuration is as follows:

- FIEXIO01.SHIFTCTL[0] = 0x00031A02
- FIEXIO01.SHIFTCTL[2] = 0x00800301
- FIEXIO01.TIMCTL[0] = 0x0B401583
- FIEXIO01.TIMCTL[2] = 0x2A401683
- FIEXIO01.TIMCFG[0] = 0x00202500
- FIEXIO01.TIMCFG[2] = 0x00206400
- FIEXIO01.TIMCMP[0] = 0x0000007F
- FIEXIO01.TIMCMP[2] = 0x00000000

## 3.4 Audio stream process

In order to realize audio digital loopback on RT1010EVK and avoid audio stream stuttering during playback, the design of processing PCM data is very important. When a DMA request is generated, the audio data should be read from or written to SHIFTBUF immediately. Figure 5 shows the specific implementation process. Two blocks are used for audio transmission and reception buffers, and these two blocks form a ping-pong buffer. The bit width of a frame of PCM data is 64 bits (the left channel and right channel), and each block contains four frames of PCM data. Whenever the audio frame in one block is full, the next block will start receiving or transmitting immediately.

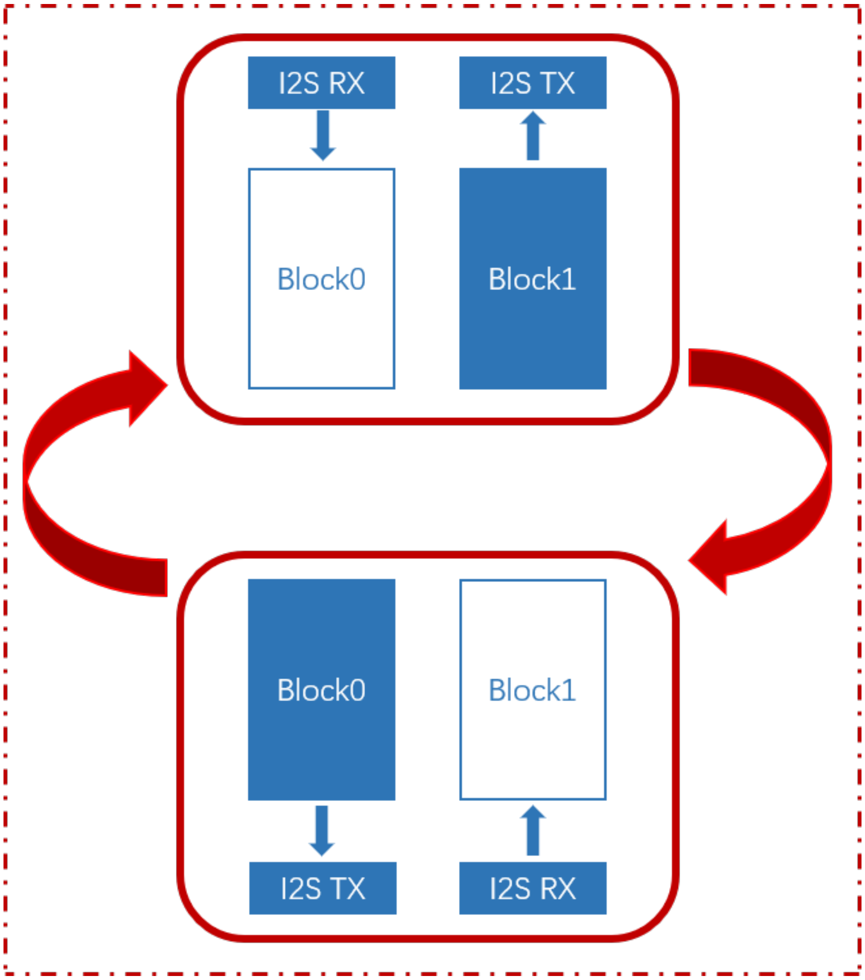

Figure 5. PCM data process

## 4 Conclusion

This note introduces an example of the $I^2S$ interface that can be implemented via the FlexIO peripheral module provided by the RT1010 MCU. If CPU resources are insufficient, it can emulate the $I^2S$ interface to transmit audio data well.

When using FlexIO module to emulate the $I^2S$ interface,

- The output valid time of $I^2S$ slave emulated by FlexIO is max 2.5 cycles because there is a maximum 1.5 cycle delay on the clock synchronization plus 1 cycle to output the data.
- In addition to the configuration of Timer and SHIFTER described in this document, there can be other combinations.

## 5 References

- *i.MX RT1010 Processor Reference Manual (Rev. B, 07/2019)* (document IMXRT1010RM)
- *Emulating the I2S Bus Master with the FlexIO Module* (document AN4955)
- WM8960 Data Sheet

*How To Reach Us*

Home Page:

nxp.com

Web Support:

nxp.com/support

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.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer’s applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and 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: March 2020
Document identifier: AN12758