AN12749
I2S(Inter-IC Sound Bus) Transmit and Receive on RT600 HiFi4

Rev. 0 — 28 February 2020

Application Note

Contents

1 Introduction

This document describes how to use I2S and DMA to record and playback audio with NXP i.MX RT600. It also includes the process of how to use the codec chip to process audio data on the i.MX RT600 Evaluation Kit (EVK) based on the Cadence Xtensa HiFi4 Audio DSP.

The i.MX RT600 is a dual-core microcontroller with 32-bit Cortex®-M33 CPU and Xtensa HiFi4 Audio DSP CPU. In addition, it provides up to 4.5 MB of on-chip SRAM (plus an additional 128 KBytes of tightly coupled HiFi4 ram) and several high-bandwidth interfaces to access off-chip flash.

The Cadence Xtensa HiFi 4 Audio DSP engine is a highly optimized audio processor designed especially for efficient execution of audio and voice codecs and pre- and post-processing modules.

The document mainly introduces the use of I2S interface around the mode and configuration of I2S on RT600, as well as the configuration and usage of DSP. This application uses an I2S interface to receive the audio stream into the buffer, and then uses another I2S interface to send the audio stream to the audio playback device.

2 Development platform

The I2S transmit and receive demo is based on RT600 EVK rev E, the actual demo hardware is shown in Figure 1.

To make example work, the following points need attention:

- JP7.1 connect to JP7.2
- JP8.1 connect to JP8.2
- Source of sound (line input to J3)
- Audio speaker (line output from J4)
- Change the ISP switch (SW5) to FLEXSPI Port B boot (0b010)
- Power on board with USB cable plugged to J6

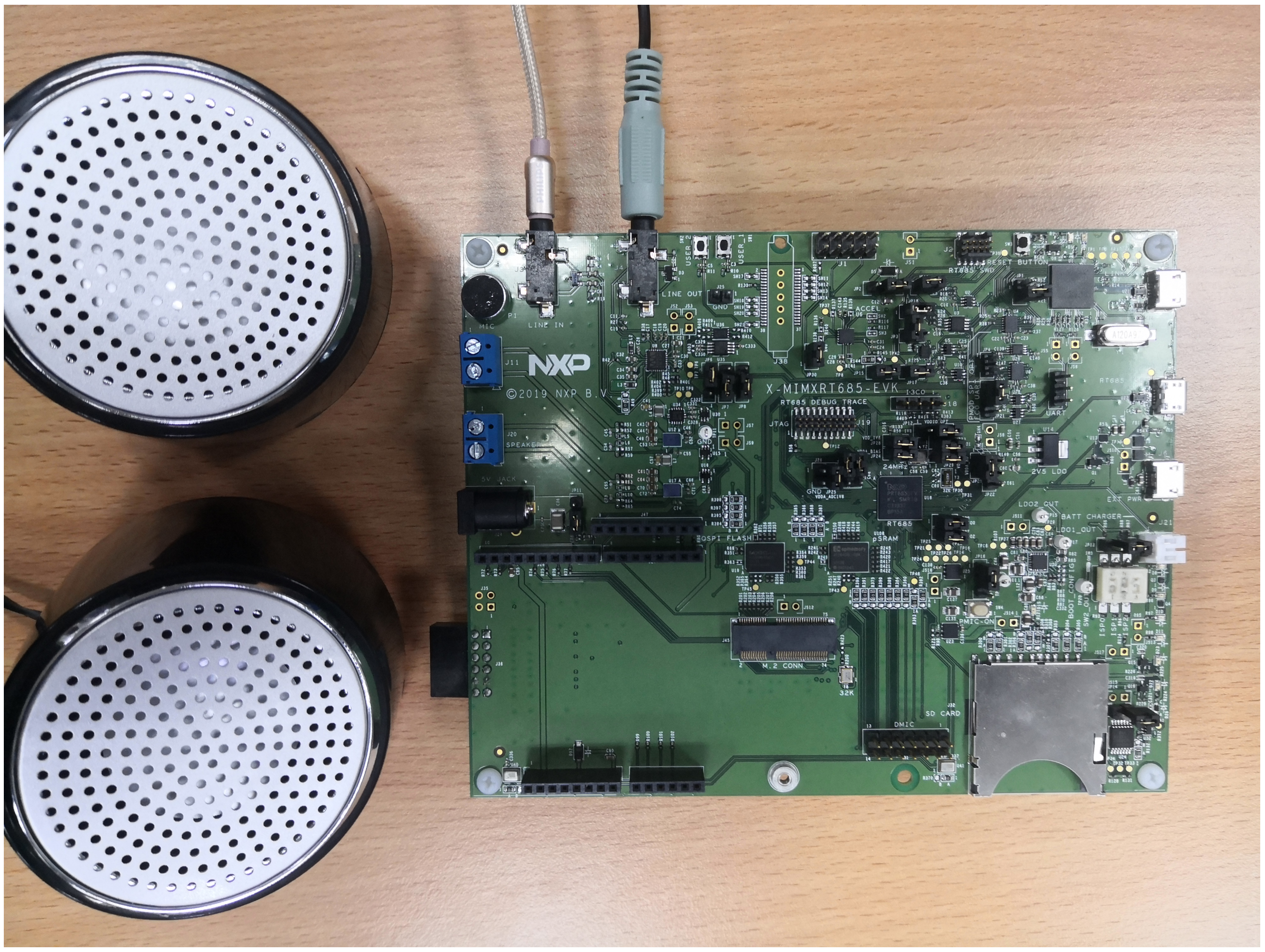

Figure 1. Actual demo hardware

3 I2S overview

3.1 Hardware architecture

The i.MX RT600 includes eight configurable universal serial interface modules (Flexcomm Interfaces), each module contains an integrated FIFO and DMA support. Flexcomms 0 through 5 can be configured as an I2S (Inter-IC Sound) interface for digital audio input or output. Each I2S supports up to four channel-pairs. The overall architecture of an example I2S subsystem is shown in Figure 2.

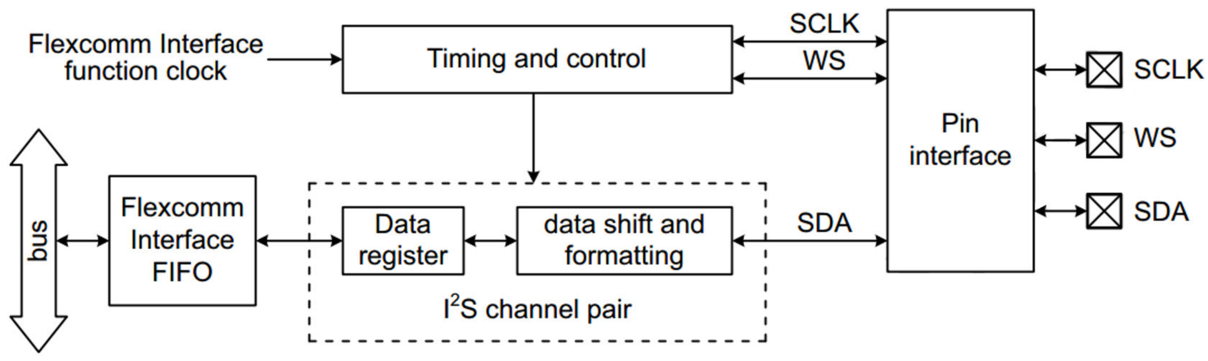

Figure 2. I2S block diagram

3.2 Frame format

The I2S module on i.MX RT600 has the following three basic operating modes: Classic I2S mode, DSP mode, and TDM mode.

The specification of Classic I2S mode defines 2 channel stereo data on SDA, where the WS state identifies the left (low) and right (high) channel, and data is delayed by 1 clock after WS transitions. The schematic diagram of the Classic I2S mode is shown in Figure 3.

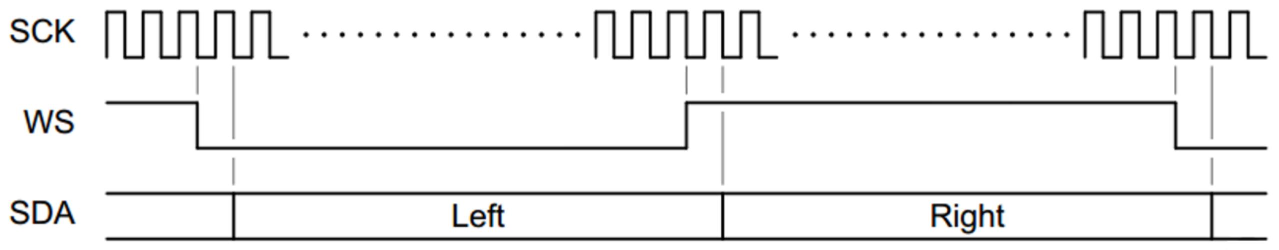

Figure 3. Classic I2S mode

DSP mode packs channel data together in the bit stream (left data followed by right data for each slot) and does not use WS to identify left and right data. In this mode, the data region begins at the leading WS edge for the frame. Through the changes in WS, DSP mode can evolve three modes: DSP mode with 50 % WS, DSP mode with 1 SCK pulsed WS and DSP mode with 1 slot pulsed WS. The schematic diagrams of these three DSP modes are shown in Figure 4 Figure 5 and Figure 6.

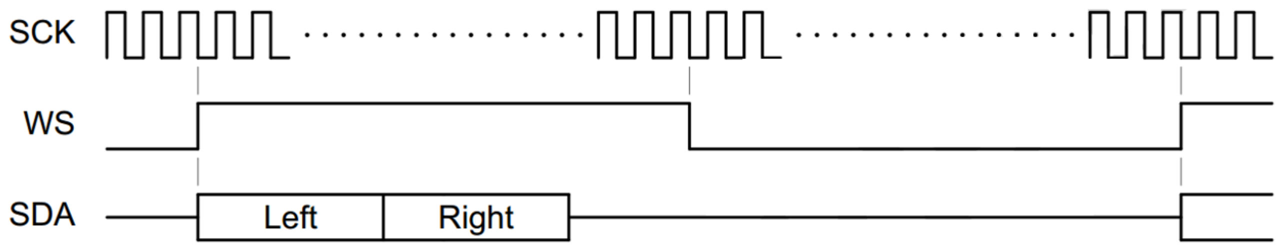

Figure 4. DSP mode with 50 % WS

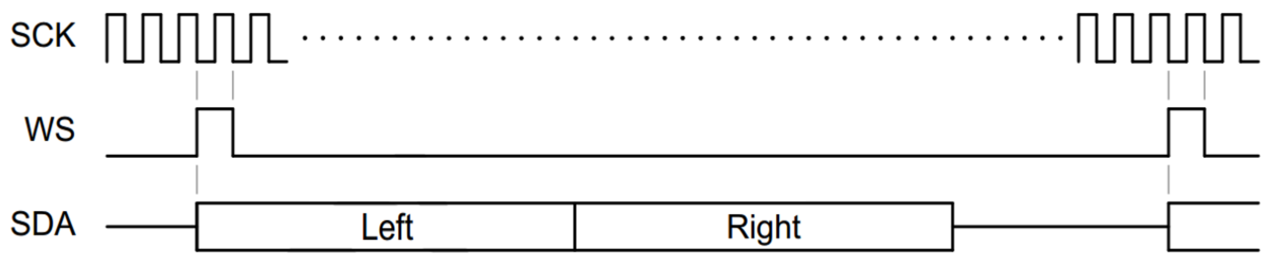

Figure 5. DSP mode with 1 SCK pulsed WS

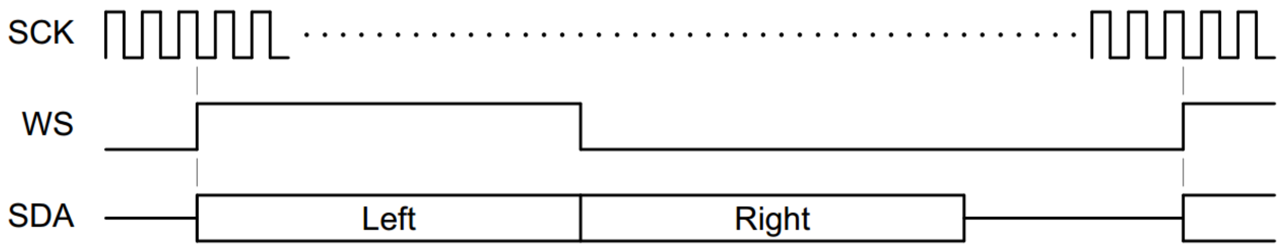

Figure 6. DSP mode with 1 slot pulsed WS

TDM mode uses multiple data slots in order to put more channels of data into a single stream. The schematic diagram of the TDM in Classic I2S mode is shown in Figure 7.

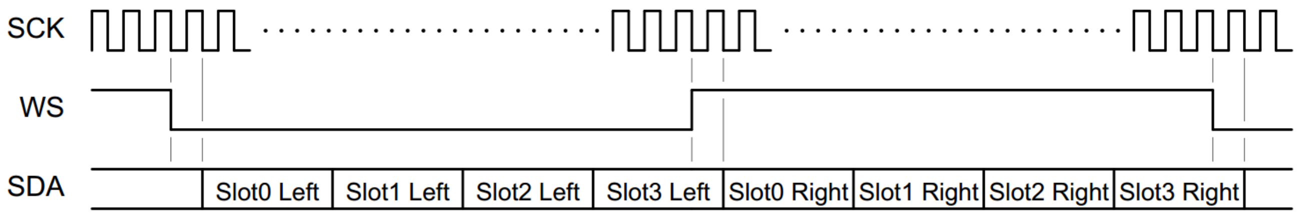

Figure 7. TDM in Classic I2S mode

4 Implementation

This chapter describes some design points of I2S in applications, focusing on the connection with common codec, the selection of I2S clock source, the calculation and configuration of clock frequency.

4.1 User case system

In this application, the architecture of the user case is shown in Figure 8. The WM8904 codec receives the audio signal from the player such as MP3 or PC. HiFi continuously receives the PCM signal of codec through one I2S interface, and uses another I2S interface to send PCM signal to codec and playback in real time.

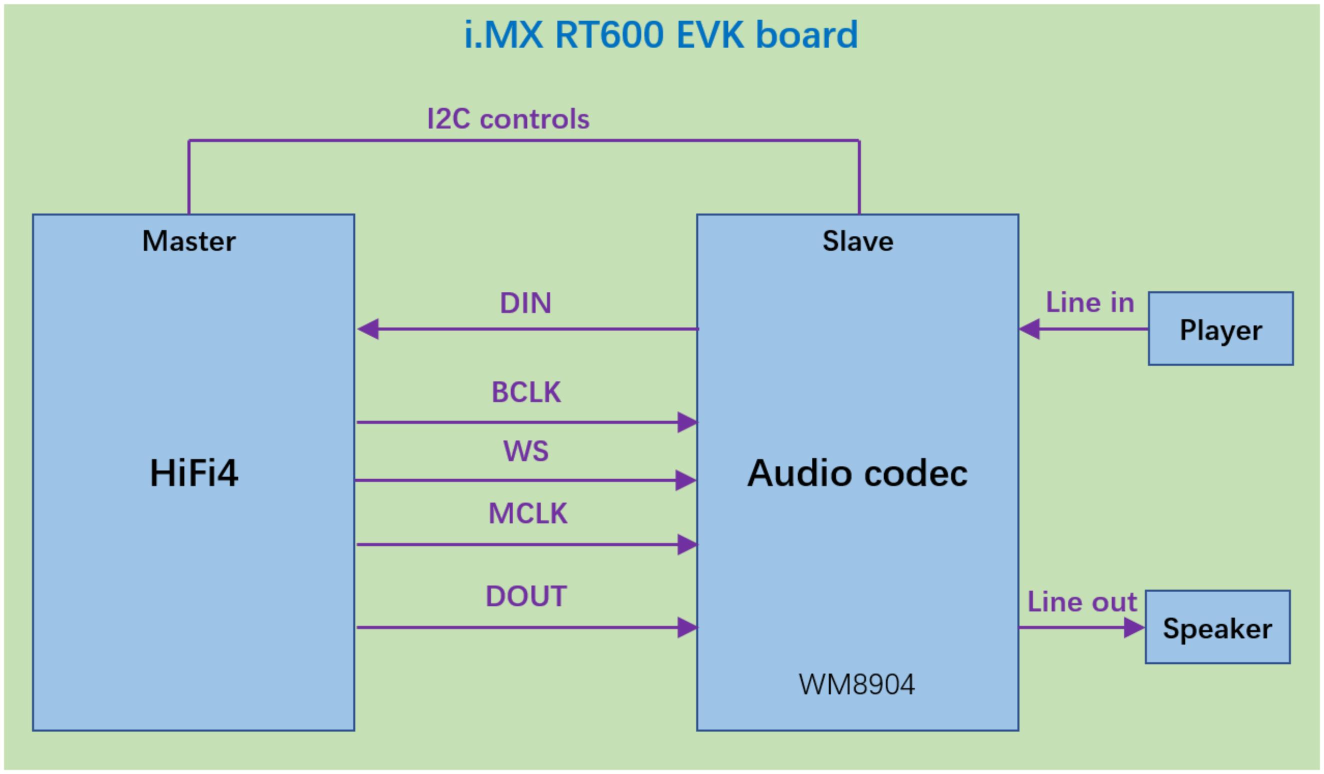

Figure 8. User case system architecture

4.2 Select clock source

A key step in using the I2S interface is to configure the I2S clock and select the clock source. In general, the clock source of MCLK (Master Clock) and I2S needs to be determined first. As shown in Figure 9. There are two clock sources for MCLK, 48/60m_irc and audio_pll_clk. There are at least five clock sources for I2S. In this application, the source of MCLK and I2S are both audio_pll_clk.

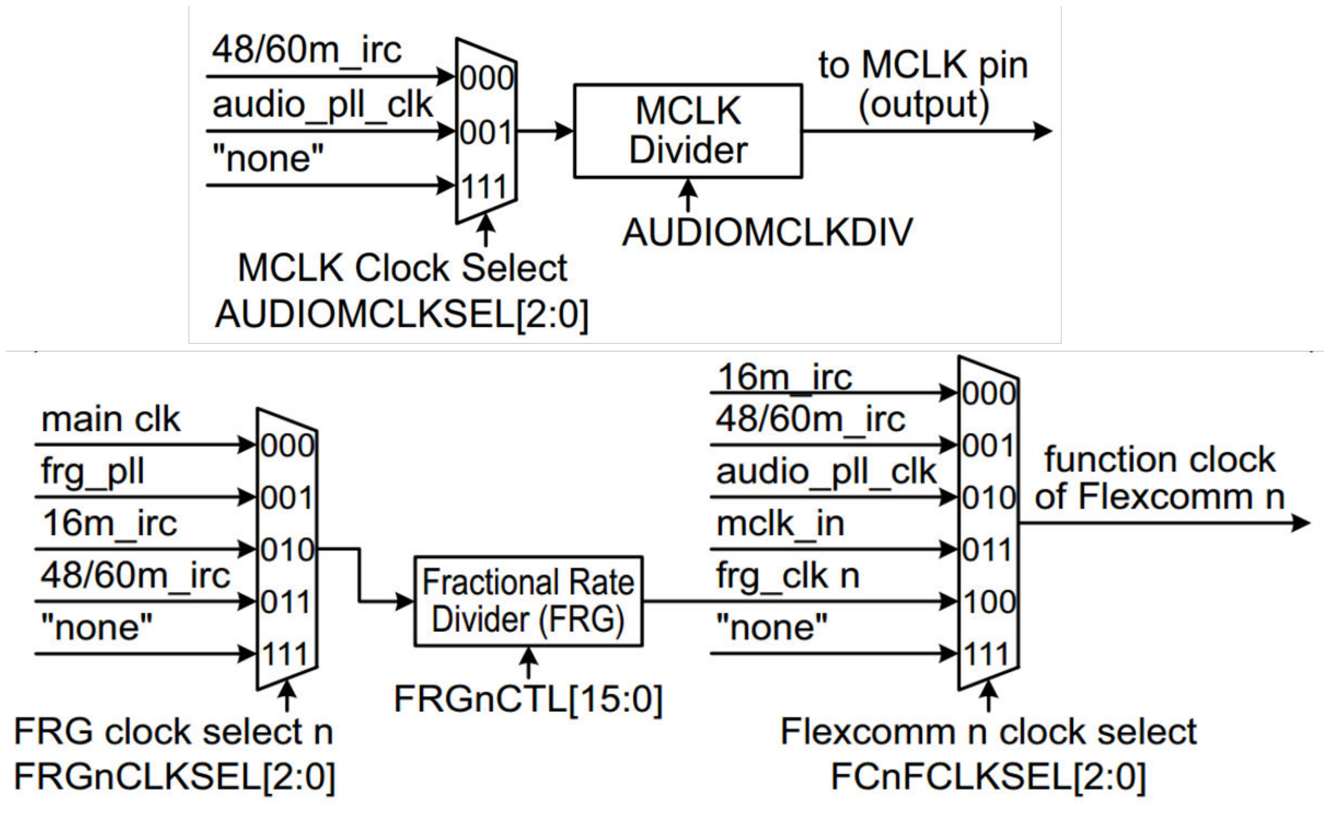

Figure 9. Communication interface function clocks

4.3 Set clock according to application

After configuring the clock source of the I2S, the next step is to configure the clock frequency of the MCLK and the clock frequency of the I2S according to the actual application. There are two kinds of clock frequencies for I2S sample rate. Typical I2S sample frequencies are shown in Table 1.

Table 1. I2S sample frequency

Typical Sample Frequency (Hz)	Typical Sample Frequency (Hz)
11025	8000
22050	16000
44100	24000
--------	32000
--------	48000
--------	96000

In general, user should check the codec data sheet to calculate the clock frequency of MCLK. For WM8904, MCLK is recommended to be 256 times the sample frequency(WS). Once the frequency of WS is determined, the frequency of MCLK can also be determined.

The calculation of the I2S bit clk(BCLK) as follows:

- BCLK = WS * Word count * Word length

In this application, the I2S operating mode is configured as Classic I2S mode, the sample rate, word count, and word length are already known, the calculation process of I2S BLCK is as follows:

- WS frequency is 48 kHz.
- MCLK frequency is configured to 24.576 MHz.
- The number of I2S Word count is set to 2.
- Word length is configured to 16.

So, BCLK = WS * Word count * Word length = 48 kHz * 2 * 16 = 1.536 MHz

Finally, the division coefficient of I2S(I2S_DIV) = audio_pll_clk /BCLK = 24.576 MHz/1.536 MHz = 16

The code in Example 1 shows the configuration of I2S Clock.

Example 1

```
#define DEMO_AUDIO_BIT_WIDTH (16)
#define DEMO_AUDIO_SAMPLE_RATE (48000)
#define DEMO_I2S_CLOCK_DIVIDER 24576000/2/16/48000
/* attach AUDIO PLL clock to FLEXCOMM1 (I2S1) */
CLOCK_AttachClk(kAUDIO_PLL_to_FLEXCOMM1);
/* attach AUDIO PLL clock to FLEXCOMM3 (I2S3) */
CLOCK_AttachClk(kAUDIO_PLL_to_FLEXCOMM3);
/* attach AUDIO PLL clock to MCLK */
CLOCK_AttachClk(kAUDIO_PLL_to_MCLK_CLK);
CLOCK_SetClkDiv(kCLOCK_DivMclkClk, 1);
SYSCTL1->MCLKPINDIR = SYSCTL1_MCLKPINDIR_MCLKPINDIR_MASK;
/* Set shared signal set 0: SCK, WS from Flexcomm1 */
SYSCTL1->SHAREDCTRLSET[0] = SYSCTL1_SHAREDCTRLSET_SHAREDSCKSEL(1) |
SYSCTL1_SHAREDCTRLSET_SHAREDWSSEL(1);
/* Set flexcomm3 SCK, WS from shared signal set 0 */
SYSCTL1->FCCTRLSEL[3] = SYSCTL1_FCCTRLSEL_SCKINSEL(1) |
SYSCTL1_FCCTRLSEL_WSINSEL(1);
s_TxConfig.divider = DEMO_I2S_CLOCK_DIVIDER;
s_RxConfig.divider = DEMO_I2S_CLOCK_DIVIDER;
```

4.4 WM8904 codec configuration

The WM8904 is a high performance ultra-low power stereo codec optimized for portable audio applications. The WM8904 uses a standard 2-wire control interface, providing full software control of all features. The HiFi4 can communicate with the codec by I2C and can use I2C for codec initialization and configuration. The WM8904 is an I2C slave device on the control interface, SCLK is a clock input, while SDA is a bidirectional data pin. To allow arbitration of multiple slaves (and/or multiple masters) on the same interface, the WM8904 transmits logic 1 by tri-stating the SDA pin, rather than pulling it high. An external pull-up resistor is required to pull the SDA line high so that the logic 1 can be recognized by the master. The sequence of signals associated with a single register write and read operation is shown in Figure 10.

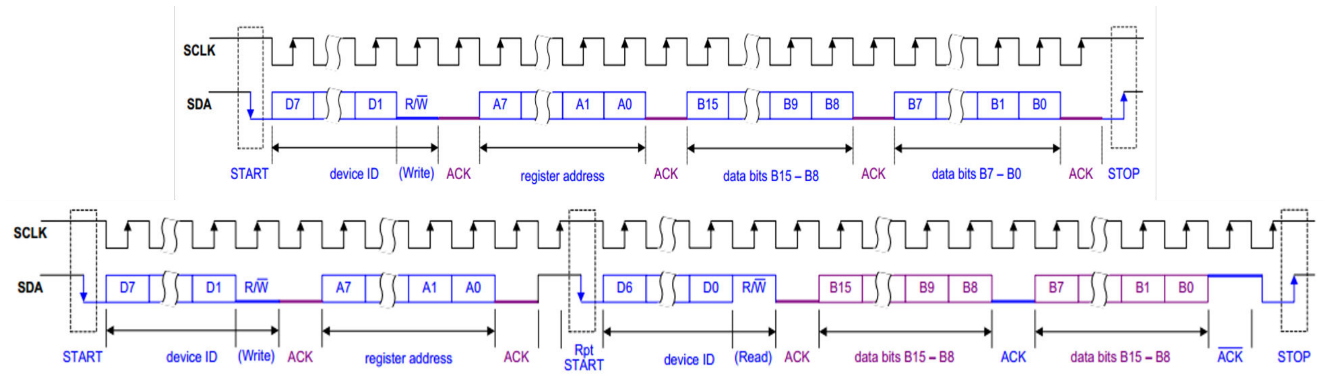

Figure 10. Control interface register write and read

The WM8904 is configured to I2S mode, the schematic diagram of the I2S mode is shown in Figure 11.

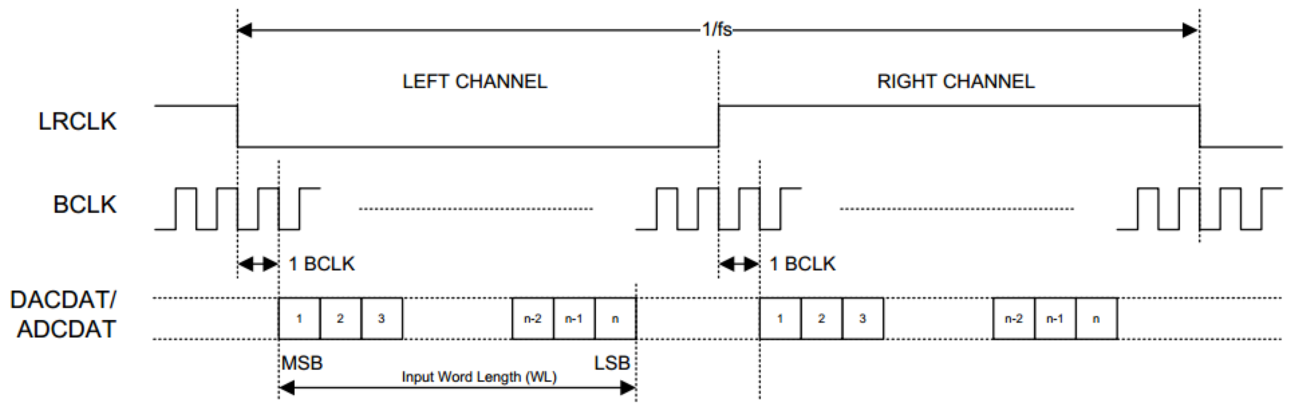

Figure 11. Codec I2S mode schematic diagram

In I2S mode, the MSB is available on the second rising edge of BCLK following an LRCLK transition. The other bits up to the LSB are then transmitted in order. Depending on word length, BCLK frequency and sample rate, there are unused BCLK cycles between the LSB of one sample and the MSB of the next.

The code in Example 2 shows the configuration of I2S Clock.

Example 2

```
wm8904_config_t wm8904Config = {
.i2cConfig = {.codecI2CInstance = BOARD_CODEC_I2C_INSTANCE, .codecI2CSourceClock
= 19000000U},
.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 = 24576000U,
.master = false,
};
static void I2C_Config(void)
```

```
{
PRINTF("Configure WM8904 codec\r\n");
/* protocol: i2s * sampleRate: 48K * bitwidth:16*/
if (CODEC_Init(codecHandle, &boardCodecConfig) != kStatus_Success)
{
PRINTF("WM8904_Init failed!\r\n");
}
/* Initial volume kept low for hearing safety. */
CODEC_SetVolume(codecHandle, kCODEC_PlayChannelHeadphoneLeft |
kCODEC_PlayChannelHeadphoneRight, 0x0020);
}
```

4.5 DMA and interrupt configuration

The DSP input multiplexer allows the user to select which DSP interrupt to be connected. The connection of interrupts to the HiFi4 is shown in Table 2. In addition to providing the required interrupt selection, the selection interrupt can also assign an interrupt priority to suit the application. The L1 interrupt has the lowest priority and the L3 interrupt has the highest priority.

Table 2. HiFi4 interrupt

Interrupt	Description	Priority	Interrupt	Description	Priority	Interrupt	Description	Priority
0	SYS IRQ	NMI	11	Interrupt selected by DSP_INT0_SEL6	L1	22	Interrupt selected by DSP_INT0_SEL17	L2
1	SOFTWARE IRQ0	L2	12	Interrupt selected by DSP_INT0_SEL7	L1	23	Interrupt selected by DSP_INT0_SEL18	L2
2	INTERNAL RTOS TIMER0	L2	13	Interrupt selected by DSP_INT0_SEL8	L1	24	Interrupt selected by DSP_INT0_SEL19	L3
3	INTERNAL RTOS TIMER1	L3	14	Interrupt selected by DSP_INT0_SEL9	L1	25	Interrupt selected by DSP_INT0_SEL20	L3
4	PROFILING IRQ	L3	15	Interrupt selected by DSP_INT0_SEL10	L1	26	Interrupt selected by DSP_INT0_SEL21	L3
5	Interrupt selected by DSP_INT0_SEL0	L1	16	Interrupt selected by DSP_INT0_SEL11	L2	27	Interrupt selected by DSP_INT0_SEL22	L3
6	Interrupt selected by DSP_INT0_SEL1	L1	17	Interrupt selected by DSP_INT0_SEL12	L2	28	Interrupt selected by DSP_INT0_SEL23	L3
7	Interrupt selected by DSP_INT0_SEL2	L1	18	Interrupt selected by DSP_INT0_SEL13	L2	29	Interrupt selected by DSP_INT0_SEL24	L3
8	Interrupt selected by DSP_INT0_SEL3	L1	19	Interrupt selected by DSP_INT0_SEL14	L2	30	Interrupt selected by DSP_INT0_SEL25	L3
9	Interrupt selected by DSP_INT0_SEL4	L1	20	Interrupt selected by DSP_INT0_SEL15	L2	31	Interrupt selected by DSP_INT0_SEL26	L3

Table continues on the next page...

Table 2. HiFi4 interrupt (continued)

10	Interrupt selected by DSP_INT0_SEL5	L1	21	Interrupt selected by DSP_INT0_SEL16	L2	------	------	------

After selecting the interrupt connection to the DSP, it is necessary to use XOS to register an interrupt handler for specified interrupt. In addition, there are two DMA controllers that can be used in a number of ways in RT600. One typical usage of DMA is that CM33 use DMA0 and HiFi4 use DMA1.

The code in Example 3 shows how HiFi4 DMA and interrupt are configured.

Example 3

```
DMA_Init(DMA1);
/* XCHAL_EXTINT19_NUM, intlevel 2 */
INPUTMUX_AttachSignal(INPUTMUX, 18U, kINPUTMUX_Dmac1ToDspInterrupt);
xos_register_interrupt_handler(XCHAL_EXTINT19_NUM,
 (XosIntFunc *) DMA_IRQHandle,
     DMA1);
xos_interrupt_enable(XCHAL_EXTINT19_NUM);
DMA_EnableChannel(DMA1, DEMO_I2S_TX_CHANNEL);
DMA_SetChannelPriority(DMA1, DEMO_I2S_TX_CHANNEL, kDMA_ChannelPriority3);
DMA_CreateHandle(&s_DmaTxHandle, DMA1, DEMO_I2S_TX_CHANNEL);
DMA_EnableChannel(DMA1, DEMO_I2S_RX_CHANNEL);
DMA_SetChannelPriority(DMA1, DEMO_I2S_RX_CHANNEL, kDMA_ChannelPriority2);
DMA_CreateHandle(&s_DmaRxHandle, DMA1, DEMO_I2S_RX_CHANNEL);
```

4.6 Audio stream process

In order to receive and play audio stream data at the same time and avoid stuttering, the method for processing audio streams is shown in Figure 12. A total of three blocks are used for audio transmission and reception buffers, and these three blocks form a cycle buffer. The bit width of a frame of audio data is 16 bits, and each block contains 16 frames of audio data (8 frames for the left and right channels). Whenever the audio frame in one block is full, the next block will start receiving immediately. The audio transmission mechanism is the same as the reception. Of course, the number of blocks depends on the actual needs of the user. When the number of blocks is 2, it forms a typical ping-pang mode.

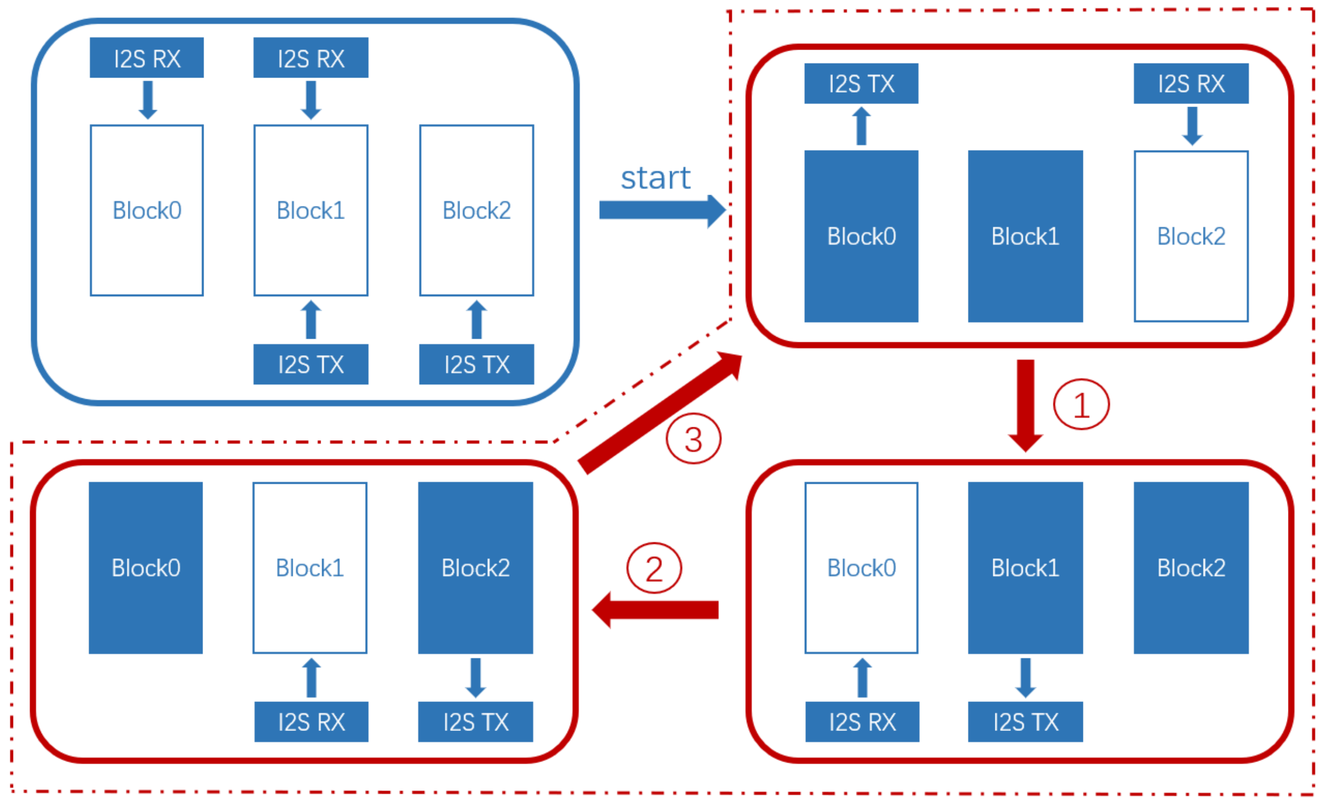

Figure 12. Audio stream process

5 Conclusion

When using I2S, some notes worth paying attention are:

- The transmission mode of I2S and the clock configuration(MCLK, WS, BCLK) needs to refer to the specific codec.
- Processing of audio streams.
- When using DMA interrupt for audio transmission on HiFi4, it is necessary to use XOS to register interrupt handler for specified interrupt.

6 References

1. An I2S (Integrated Interchip Sound Bus) Application on Kinetis (AN4800).
2. Using Multi-Channel Feature of SAI (AN12090).
3. RT6xx User manual (Rev. 0.95 — 11 November 2019).
4. Xtensa XOS Reference.
5. WM8904 Data Sheet.

7 Revision history

Revision number	Date	Substantive changes
0	26/02/2019	Initial release

Date of release: 28 February 2020
Document identifier: AN12749