AN12764

8-channel DMIC Audio Acquisition on RT600 HiFi4

Rev. 0 — March 7 2020

Application Note

Contents

1 Introduction

The i.MX RT600 is a dual-core MCU with 32-bit Cortex®-M33 CPU and Xtensa HiFi4 Audio DSP CPU. The Cadence Xtensa HiFi4 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.

This application note describes how to use 8-channel dual/stereo Digital Microphone Interface (DMIC) to acquire audio data, and then send the 8-channel PCM data to the codec in TDM mode via I^2S for real-time playback. The demo is based on the NXP i.MX RT600 EValuation Kit (EVK) board.

2 Development platform

The 8-channel DMIC acquisition demo is based on RT600 EVK Rev E. The actual demo hardware consists of the following four parts, as shown in Figure 1.

- QSPI NOR flash board
- DMIC audio board
- CS42888 audio board
- RT600 EVK board

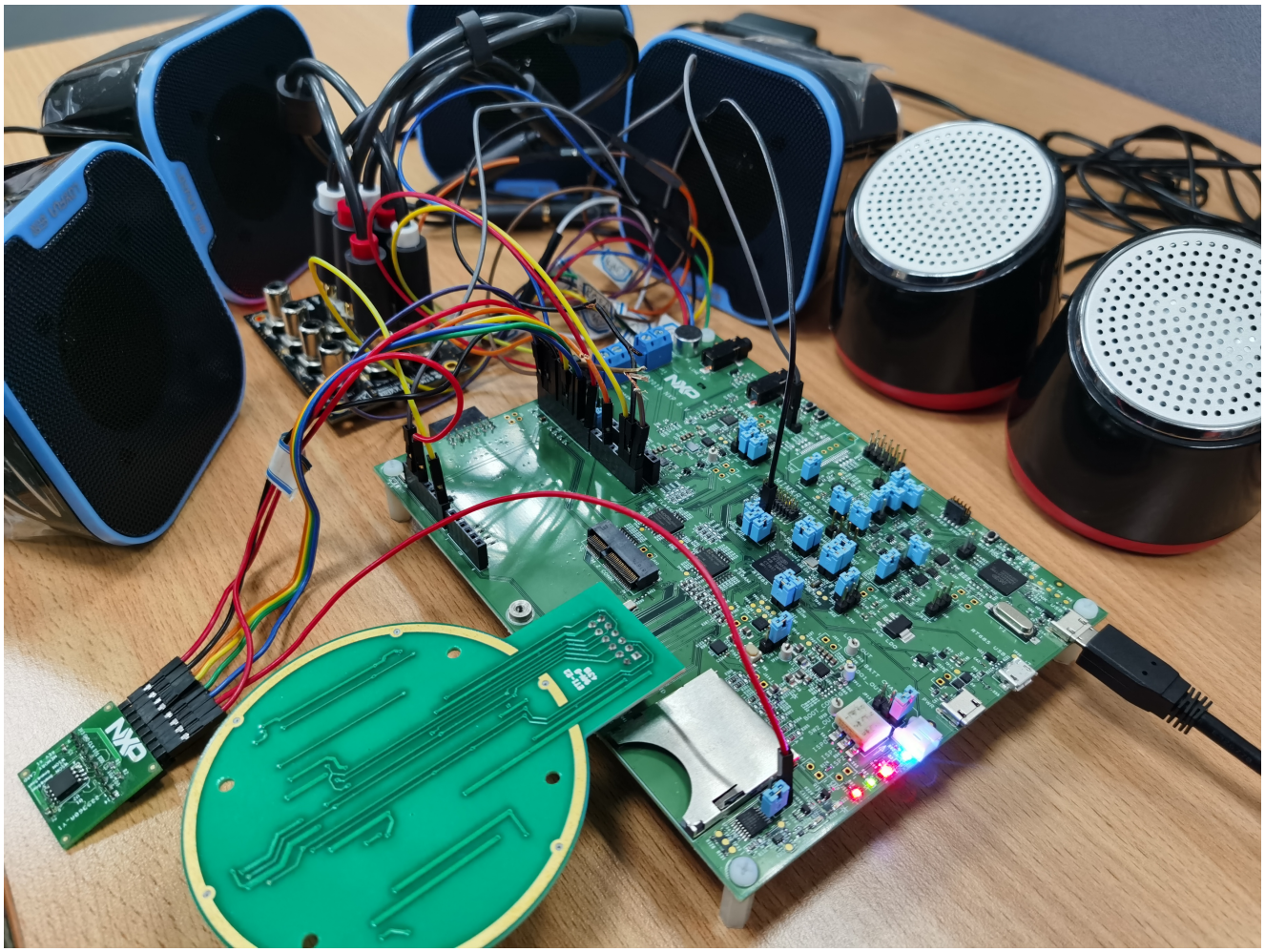

Figure 1. Actual demo hardware

To keep the example work, make sure that:

- DMIC audio board is connected to RT600 EVK board J31.
- The connection of CS42888 audio board and RT600 EVK board is as described in Table 1.

Table 1. Connections between boards

CS42888 audio board	RT600 EVK board
A4 PIN (GND)	J25 PIN2
A5 PIN (BCLK)	J27 PIN3
A6 PIN (WS)	J27 PIN2
A8 PIN (TX)	J27 PIN1
A9 PIN (3V3)	J29 PIN2
A11 PIN (RESET)	J28 PIN1
A13 PIN (5V)	J29 PIN8

Table continues on the next page...

Table 1. Connections between boards (continued)

CS42888 audio board	RT600 EVK board
A16 PIN (1V8)	JP10 PIN1
B5 PIN (SCL)	J28 PIN10
B6 PIN (SDA)	J28 PIN9
B12 PIN (MCLK)	J27 PIN6

- The connection of QSPI NOR flash board and RT600 EVK board is as described in Table 2. The hardware connection between Flexspi Port B and DMIC of the RT600 EVK board conflicts. If DMIC is used, the Flexspi Port B cannot be used. It is necessary to use another flash.

Table 2. Connections between boards

QSPI NOR flash board	RT600 EVK board
PIN_3V3	J28 PIN8 (3V3)
PIN_GND	J28 PIN7 (GND)
PIN_CLK	J28 PIN6 (CLK)
PIN_IO1	J28 PIN5 (MISO)
PIN_IO0	J28 PIN4 (MOSI)
PIN_IO2	JS15 PIN1 (3V3)
PIN_IO3	J29 PIN4 (3V3)
PIN_CS	J28 PIN3 (SS0)

- Audio speakers plug into J11, J12, J13, J14, J15, and J16 ports of CS42888 audio board.
- RT600 EVK board changes include:
  - In order to use DMIC interface, these changes need to be made: remove R379-A, R380-A, R384-A, R389-A, R390-A, R391-A, R392-A, R393-A, and weld R379-B, R380-B, R384-B, R389-B, R390-B, R391-B, R392-B, R393-B with 0 ohm resistor.
  - Change the ISP switch (SW5) to (ON OFF OFF) and download the demo image into the QSPI NOR flash. Then change the ISP switch (SW5) to (ON OFF ON) and make RT600 boot from QSPI NOR flash. For specific implementation methods, see *How to Enable Recovery Boot from QSPI Flash* (document AN12751).
  - Power on board with USB cable plugged to J6.

3 DMIC overview

This chapter introduces the modules that need to be configured in the DMIC interface of i.MX RT600 and how to use them.

3.1 Feature

The DMIC subsystem on RT600 has the following features:

- Pulse-Density Modulation (PDM) data input for up to eight total channels on four data lines.
- Flexible decimation.
- 16-entry FIFO for each channel.
- DC blocking or unaltered DC bias can be selected.

- DMA supported on a channel-by-channel basis.

Each PDM interface provides the option of supporting two single-channel microphones or a single stereo microphone. Figure 2 shows an 8-channel DMIC multiplexing connection diagram.

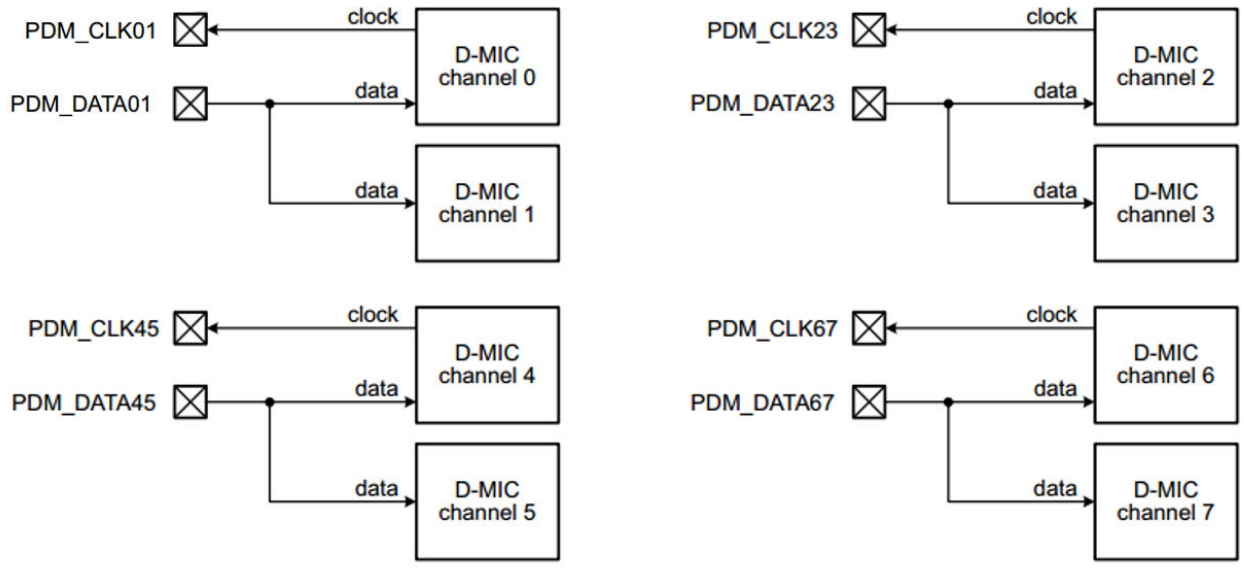

Figure 2. DMIC multiplexing connection

3.2 PDM-to-PCM conversion

The DMIC interface receives PDM data from multiple digital microphones and processes it to produce 16-bits or 24-bit PCM data. PCM data can be read by the CPU or DMA, and can be processed according to actual needs. In this application, the PCM data is sent to the codec chip via the I^2S interface. RT600 contains eight DMIC channels in total. Each DMIC channel can be configured in many aspects. Figure 3 shows a block diagram of a DMIC channel.

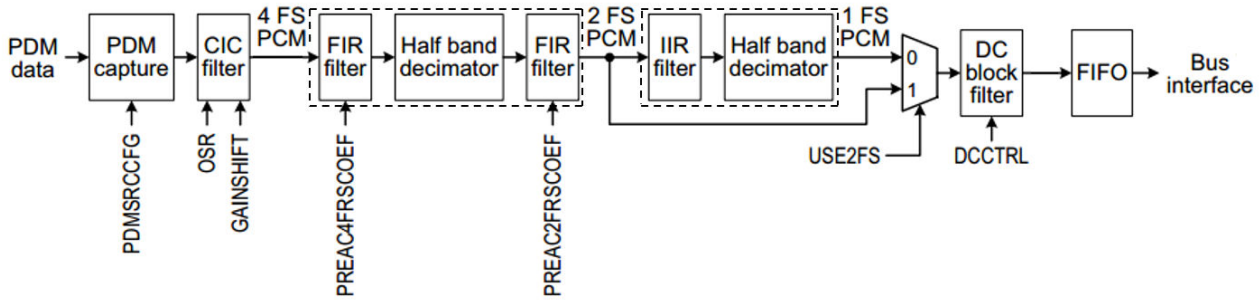

Figure 3. DMIC channel block diagram

According to the block diagram of each DMIC channel, the PDM needs to be converted to PCM through a filter when the PDM data is captured. The filter module for PDM to PCM conversion includes four stages. It starts with a Cascaded-Integrator Comb (CIC) filter, which is an optimized Finite Impulse Response (FIR) filter combined with a decimator. The CIC filter converts the PDM stream from the digital microphone into PCM data at a given oversampling rate, which is set for each channel in the OSR register. Then the PCM data is processed by a half-band decimator, while compensating for the roll-off of the upper limit of the

audio band. The next block again decimates the signal to produce a PCM signal with the required sampling rate. Finally, DC block filter eliminates any unwanted DC components in the audio signal.

In addition, in order to achieve lower power consumption, setting the USE2FS bit in the USE2FS register can bypass the second half-band decimation filter and provide 2FS instead of 1FS signals for the DC filter. When the PCM data is read from the FIFO via CPU or DMA, its bit width can be 16-bit and 24-bit, which can be controlled by configuring the `SATURATEAT16BIT` and `SIGNEXTEND` bits of the `DC_CTRL` register.

3.3 Clock

DMIC sample rate depends on three aspects:

1. DMIC interface base clock.

   DMIC interface base clock supplies the peripheral block. It can be configured by register `CLKCTL1 DMIC0CLKSEL` and `DMIC0CLKDIV`. The source of DMIC interface peripheral includes SFRO, FFRO, Audio PLL, MCLK input, LPOSC and 32k_wake_clk, as shown in Figure 4.

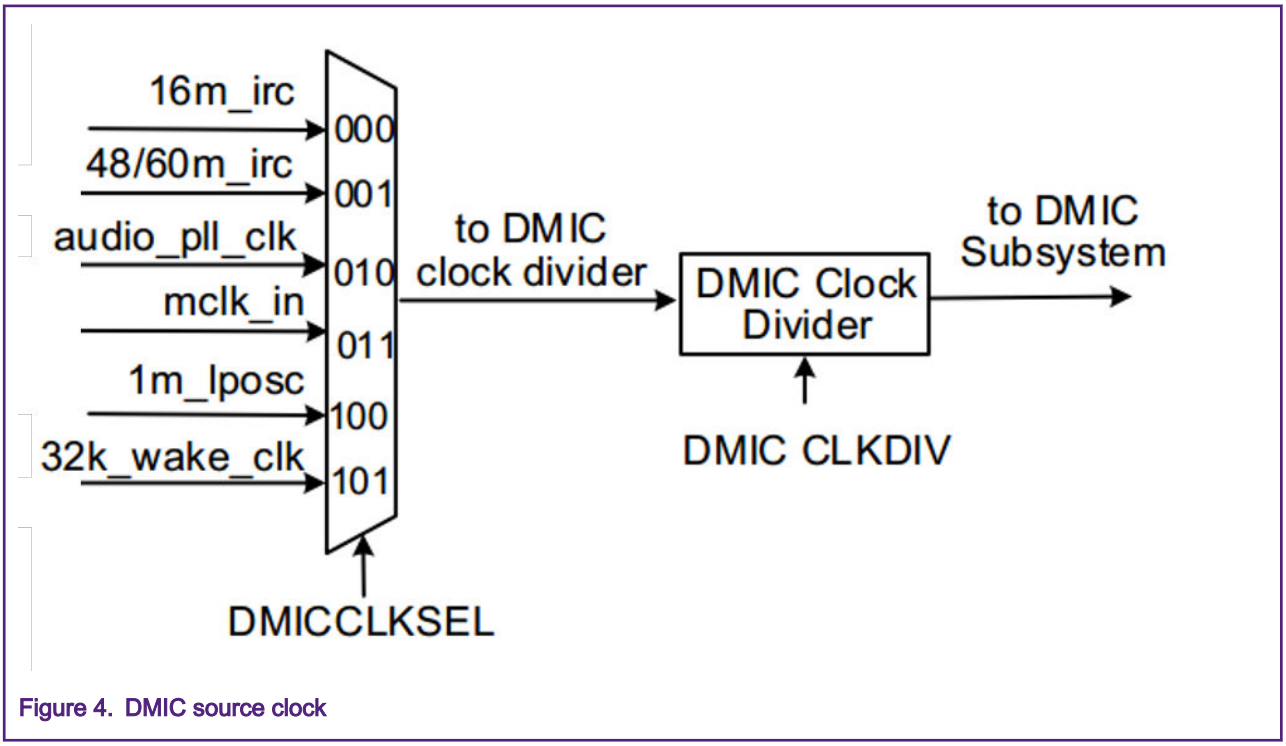

Figure 4. DMIC source clock

2. DMIC sample clock for the digital microphone.

   NOTE

   As the DMIC peripheral block is designed to run at a DMIC sample clock speed no faster than 6.144 MHz, the frequency of the input source clock should not exceed 24.576 MHz.

3. PCM sample rate.

   The relation between the DMIC clock rate and the PCM sample rate is as follows:

   PCM sample rate = DMIC clock rate/(N * OSR) (2 FS mode, N = 2; 1 FS mode, N = 4)

3.4 FIFO and DMA

The effective width of the PCM data in the FIFO can be either 16-bit or 24-bit. There are two types of access to the FIFO data. First, configuring the trigger level of the FIFO, and then select the data trigger level for interrupt or DMA operation. Both methods PCM data can be copied from FIFO to SRAM. Figure 5 shows a block diagram of a process for copying DMIC FIFO data via DMA.

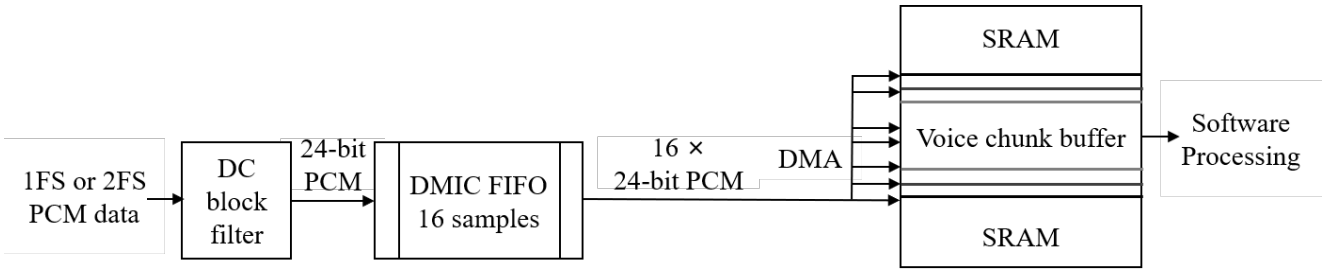

Figure 5. DMIC FIFO and DMA diagram

There are two DMA controllers in i.MX RT600. Generally, the main recommended usage method is to allocate DMA0 to CM33 and DMA1 to HiFi4. There are several points to note when using DMA in HiFi4:

- HiFi4 interrupt should be registered and enabled in XOS or XTOS. The XOS is an embedded kernel from Cadence is designed for efficient operation on embedded systems built and the XTOS is a single-threaded runtime for Xtensa processors.
- The SRAM address of DMA operation needs to be non-cacheable.
- HiFi4 interrupt needs to be configured by INPUTMUX.

4 Implementation

This chapter describes the design points of the eight-channel dmic acquisition use case.

4.1 User case system

In this application, the architecture of the user case is as shown in Figure 6. The HiFi4 receives the PDM data from the DMIC audio board via 8-channel DMIC interfaces. Then, the cs42888 codec receives the PCM data processed by HIFI4 via the I^2S interface and transmits it to the speakers for real-time playback.

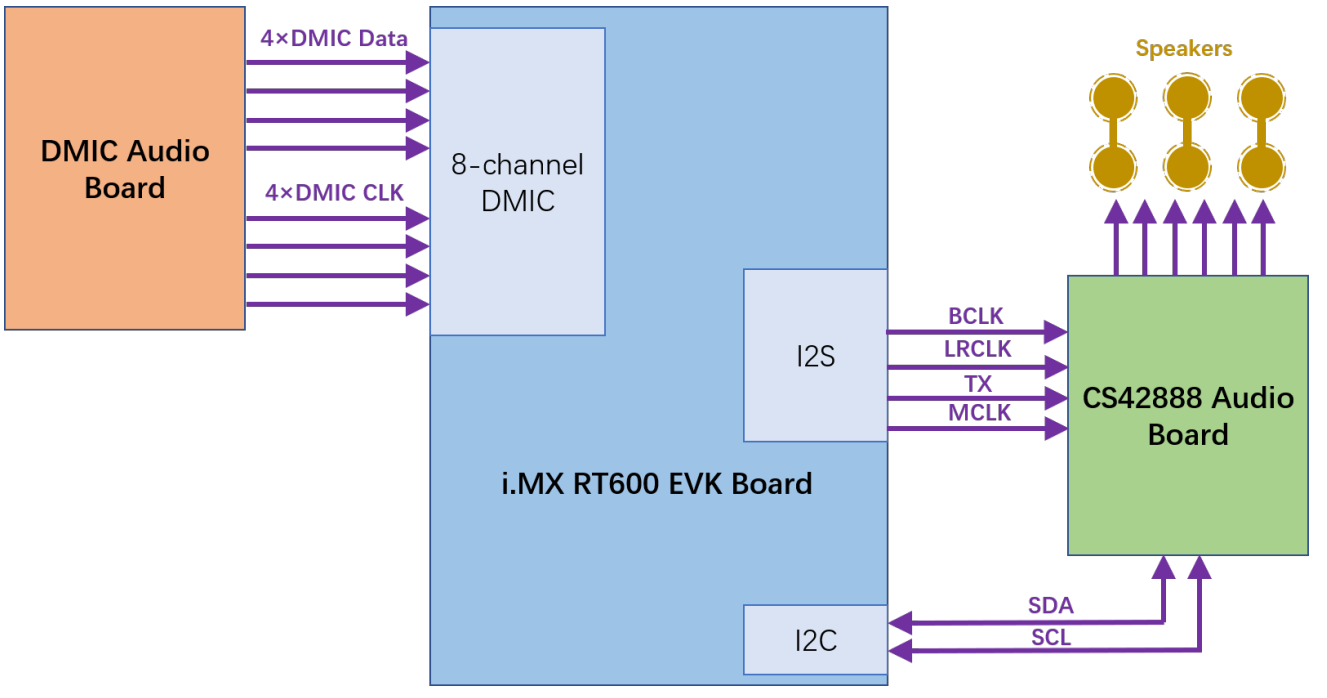

Figure 6. User case system architecture

The design process of the system includes the following steps:

1. Board, pin, and clock initialization, such as system clock, audio clock, DSP clock.
2. Initialization and configuration of the cs42888 codec by I^2C.
3. DMIC channel, clock and sample rate configuration.
4. I^2S clock and format configuration.
5. Interrupt and DMA configuration.
6. Audio stream process design.

4.2 CS42888 codec configuration

The cs42888 is a highly-integrated mixed signal 24-bit audio codec. It comprises of four Analog-to-Digital Converters (ADC) implemented with multi-bit delta-sigma techniques and eight Digital-to-Analog Converters (DAC) also with multi-bit delta-sigma techniques. All eight DAC channels provide digital volume control and can operate with differential or single-ended outputs. The DAC serial ports of cs42888 support I2S TDM digital interface formats with varying bit depths from 16 to 24 and allow up to eight DAC channels in a Time-Division Multiplexed (TDM) interface format. The timing diagram of TDM is as shown in Figure 7.

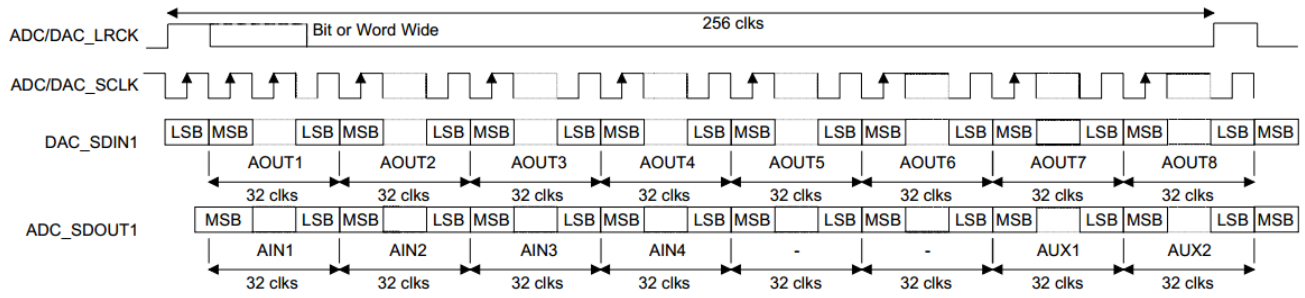

Figure 7. TDM format

The TDM data receives Most Significant Bit (MSB) first, on the second rising edge of the `DAC_SCLK` occurring after a `DAC_LRCK` rising edge. All data is valid on the rising edge of DAC_SCLK. The AIN1 MSB is transmitted early, but is guaranteed valid for a specified time after SCLK rises. All other bits are transmitted on the falling edge of `DAC_SCLK`. Each time slot is 32-bit wide, with the valid data sample left justified within the time slot. `DAC_SCLK` must operate at 256 sample rate.

4.3 DMIC configuration

In this use case, perform the following setting:

- Set the PCM sample rate to 48 kHz.
- Set the effective bit width of each frame to 16 bit.
- Configure the DMIC around the sample rate as below.
  - DMIC clock source: `audio_pll_clk`
  - DMIC peripheral input clock: 24.576 MHz
  - DMIC clock divider: 8
  - DMIC clock rate: 3.072 MHz
  - OSR register: 32
  - FS Mode: 2FS

Therefore,

DMIC PCM sample rate = DMIC clock rate/(N × OSR) = 3.072 MHz/2/32 = 48 kHz

Each DMIC channel has a corresponding Decimator Gain Shift registers and DC Control register that can be set. The code below gives the specific configuration for reference.

```
/* dmic channel configurations */
static dmic_channel_config_t s_dmicChannelConfig = {
    .divhfclk            = kDMIC_PdmDiv1,
    .osr                 = 32U,
    .gainshft            = 3U,
    .preac2coef          = kDMIC_CompValueZero,
    .preac4coef          = kDMIC_CompValueZero,
    .dc_cut_level        = kDMIC_DcCut155,
    .post_dc_gain_reduce = 1U,
    .saturate16bit       = 1U,
    .sample_rate         = kDMIC_PhyFullSpeed,
    .enableSignExtend    = false,
};
```

4.4 Flexcomm configuration

There are up to eight multi-function flexcomm interfaces available on i.MXRT600. In this application, the flexcomm2 is configured as I²C interface.

The cs42888 uses a standard 2-wire control interface and provides full software control of all features. The HiFi4 can communicate with the codec via the I²C interface and can use I²C for codec initialization and configuration. Cs42888 is configured as an I²C slave device. Figure 8 shows the signal timings for a read-and-write cycle.

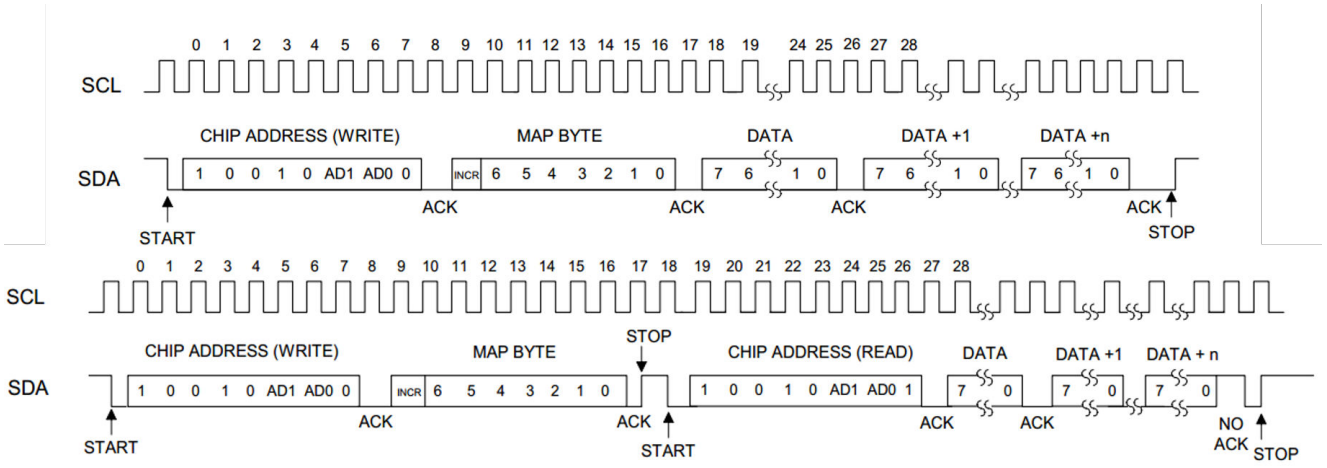

Figure 8. Codec I²C mode schematic diagram

The flexcomm4 interface is configured as I²S and used to transmit 8-channel PCM data. The sample rate of each frame of DMIC PCM data is 48 kHz, so the configuration of I²S based on this sample rate is as follows:

- I²S transmit mode: TDM
- I²S word count: 8
- I²S word length: 32
- MCLK frequency: 12.288 MHz
- I²S clock source: `audio_pll_clk`
- I²S peripheral input clock: 24.576 MHz
- I²S clock divider: 2

Therefore,

I2S BCLK = I2S peripheral input clock/I2S clock divider = 24.576 MHz/2 = 12.288 MHz

I2S sample rate = I2S BCLK/I2S word count/I2S word length = 12.288 MHz/8/32 = 48 KHz.

4.5 DMA and interrupt configuration

The INPUTMUX allows the user to select which DSP interrupt to be connected. The connection of interrupts to the HiFi4 is as described in Table 3. In this application, interrupt 23 (Interrupt selected by `DSP_INT0_SEL18`) is selected.

Table 3. 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
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. The code below shows the how HiFi4 DMA and interrupt are configured.

```
#define XCHAL_EXTINT19_NUM        23    /* (intlevel 2) */
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);
```

4.6 PCM stream process

According to the codec cs42888 data sheet, if cs42888 works in the TDM mode, the PCM data bit width received by cs42888 needs to be processed into 32-bit, and the effective data can be 16-bit or 24-bit. In this application, the PCM data received by the 8-channel DMIC are all configured as 16-bit, so the PCM data needs to be padded from 16-bit to 32-bit wide. After that, the PCM data with a frame length of 256-bit can be transmitted to the codec via the I^2S interface. In order to realize the data padding mentioned above, the function of DMA memory to memory is adopted. The schematic diagram of the entire data processing is as shown in Figure 9.

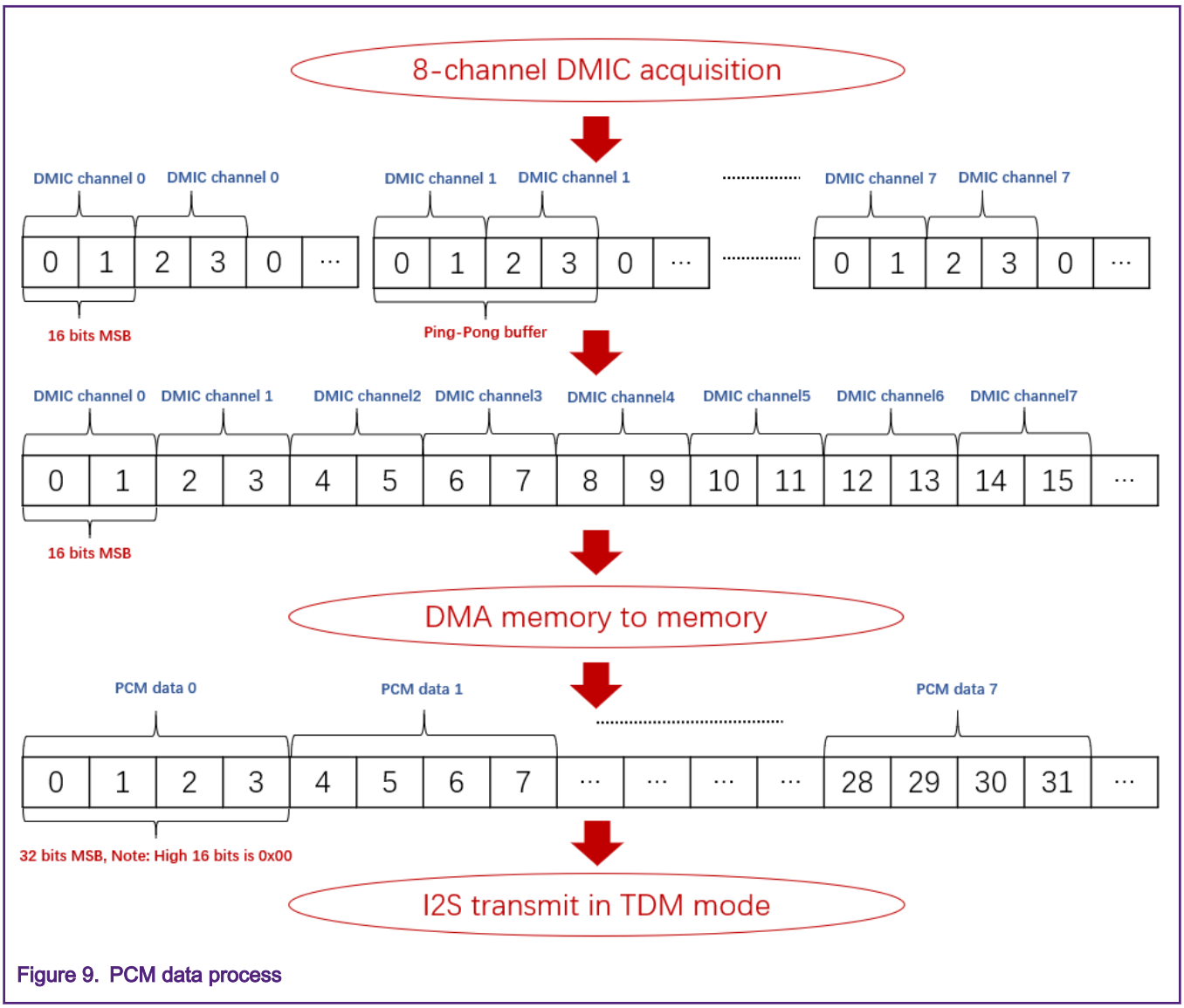

Figure 9. PCM data process

After processing eight channels of 16-bit PCM into 8 channels of 32-bit PCM data, HiFi4 uses I^2S to transmit PCM data to cs42888 codec. In order to play audio stream data and avoid stuttering, the method for processing PCM streams is as shown in Figure 10. A total of three blocks are used for PCM data buffers, and these three blocks form a cycle buffer. Each block is 1024 bytes and contains 32 frames each including eight channels of 32-bit PCM data. Whenever the DMA-processed PCM data in one block is full, the next block will start receiving immediately. In addition, the I^2S TX and MToM operations in Figure 10 have been operated twice in the beginning, so that all subsequent DMA operations are seamlessly connected, which can ensure the continuity of audio transmission.

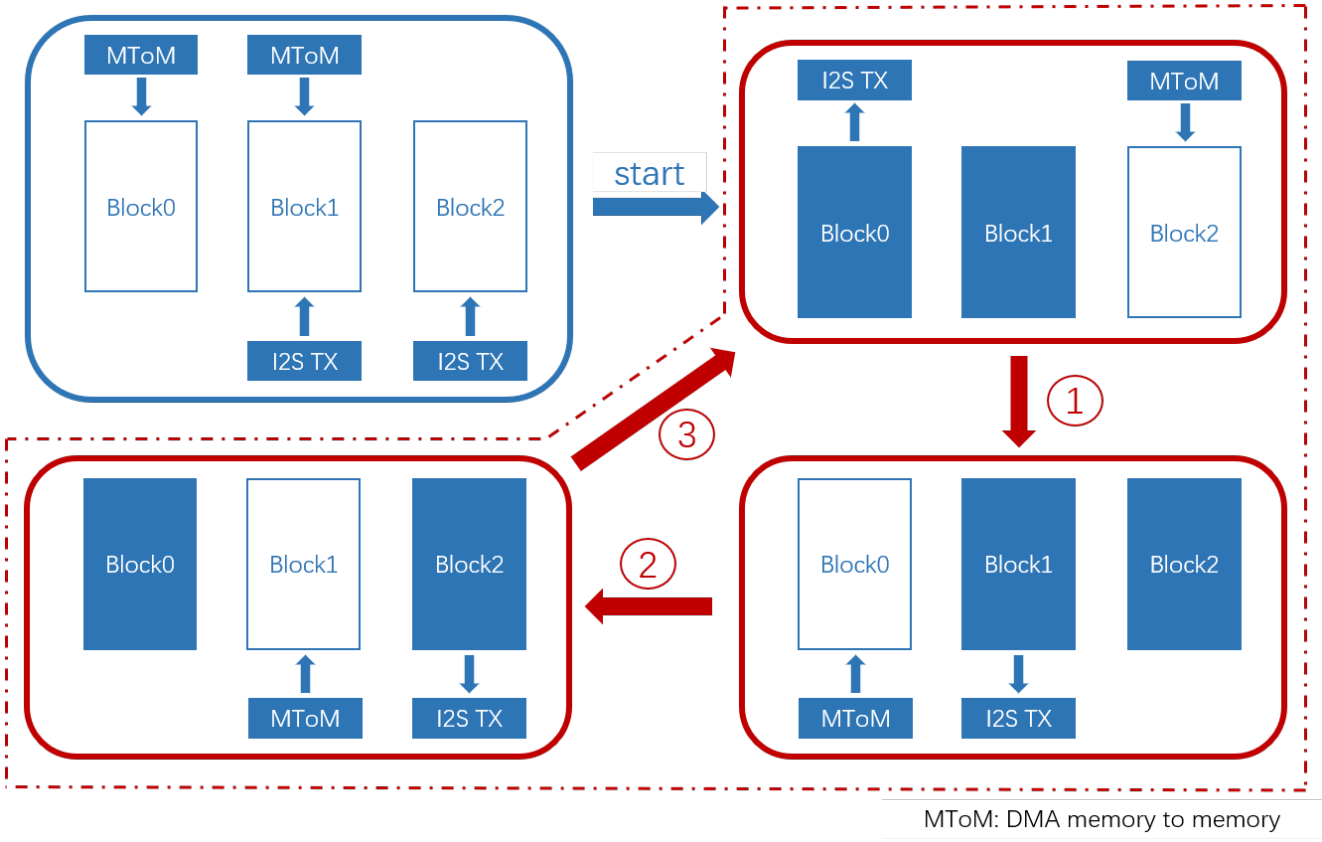

Figure 10. PCM stream process

5 Conclusion

When using DMIC, some notes worth paying attention are:

- When the 8-channel DMIC acquires audio data at the same time, the PDM data needs to be stitched and processed. The audio stream process logic must be designed carefully.
- The configuration of the DMIC channel needs to consider factors such as the sample rate and the internal filter of the channel.
- When using DMA on HiFi4, HiFi4 interrupts should be configured by INPUTMUX and registered in XOS or XTOS.

6 References

- *RT6xx User Manual* (Document UM11147)
- X-MIMXRT686-EVK Board Rev E.
- Cadence Xtensa XOS Reference Manual
- CS42888 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 7 2020

Document identifier: AN12764

arm