AN13291

FLAC7.1 Implementation and Audio Quality Measurement on RT1060 and CS42448 Audio Card

Rev. 0 — 10 June 2021 Application Note

Contents

1 Introduction

The i.MX RT series are the industry's first crossover processor. i.MX RT1060 is a part of the EdgeVerse™ edge computing platform. The i.MX RT1060 device runs on the Arm® Cortex®-M7 core at 600 MHz, which can support high-performance applications.

CS42448 Audio Card is an audio expansion card, which can be directly connected to RT1060EVKB, to support more complex audio applications on RT060EVKB.

i.MX RT1060 provides rich audio interfaces, including SAI-1, SAI-2, SAI-3, SPDIF and MQS. This application note introduces:

- How to implement Flac7.1 decoding on RT1060
- How to implement 8-channel playback in TDM mode with SAI-1 interface on the CS42448 audio card
- How to use APx525 to measure the audio quality of Cs42448 audio card

2 FLAC overview

Free Lossless Audio Codec (FLAC) is an audio format similar to MP3 but lossless, which means that the audio in FLAC is compressed without any quality loss. The features are as follows::

- FLAC source code is available under the open-source license.
- FLAC is the first truly open and free lossless audio format.
- FLAC is compressed without any loss in quality.

After Flac7.1 is decoded, eight channels of audio data are sent to eight speakers for playback. Optimizing the position of these eight speakers will produce an immersive effect. Table 1 lists the channel assignment of the eight channels.

Table 1. FLAC7.1 channel assignment

Front Left	Front Right
Front Center	LFE
Back Left	Back Right
Side Left	Side Right

3 Implementation

As shown in Figure 1 and Figure 2, eight speakers are placed according to the 7.1 surrounding sound standard to obtain a better 7.1 surrounding sound effect. This chapter introduces the specific implementation method.

Figure 1. Actual Demo

Figure 2. Connection between RT1060 EVKB and audio card

3.1 User case system

Figure 3 is the system architecture of the Flac7.1 surrounding user case. As shown in Figure 3, RT1060 retrieves the audio file from the SD card through the SDIO interface. The audio file is FLAC7.1 format. Then RT1060 performs decoding. After the decoding is completed, RT1060 sends 8-channel audio data to the audio card in TDM mode by SAI-1 interface. The audio card integrates the codec cs42448 chip onboard. This codec chip supports up to six channels of input and 8 channels of output at the same time.

SAI-1 sends 8-channel audio digital signals to the cs42448 codec, the cs42448 codec converts 8-channel audio digital signals into 8-channel analog signals, and then it sends them to eight speakers for playback.

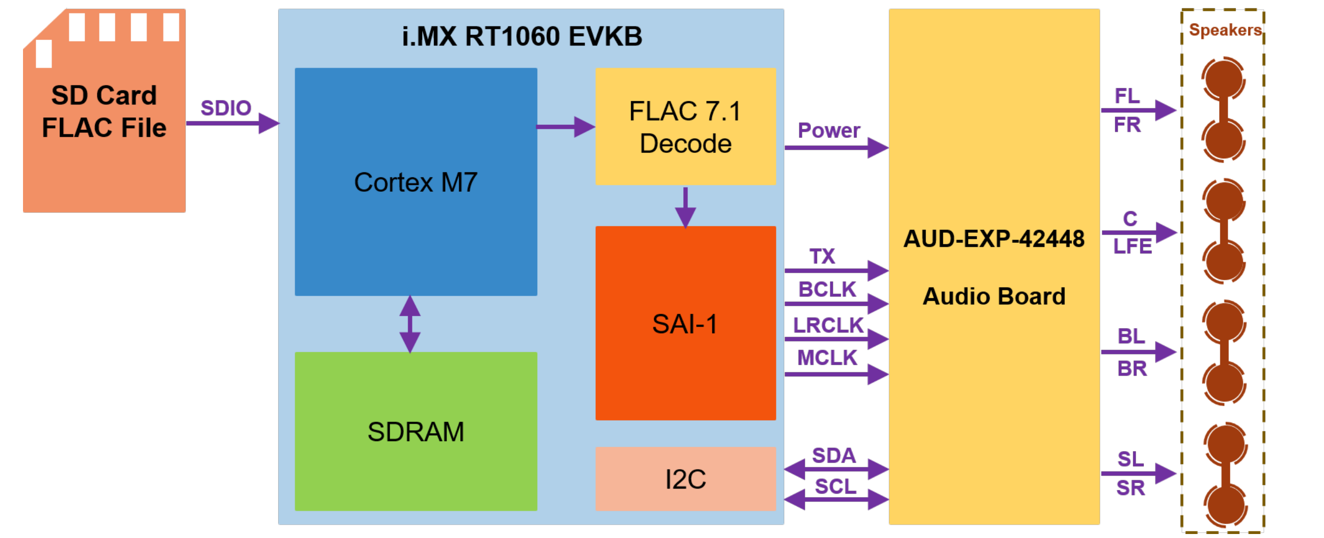

Figure 3. User case system architecture

3.2 CS42448 codec configurations

The cs42448 is a highly integrated mixed signal 24-bit audio codec. It consists of six Analog-to-Digital Converters (ADC) that implemented using multi-bit delta-sigma techniques and eight Digital-to-Analog Converters (DAC) that also implemented using 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 cs42448 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. shows the timing diagram of TDM.

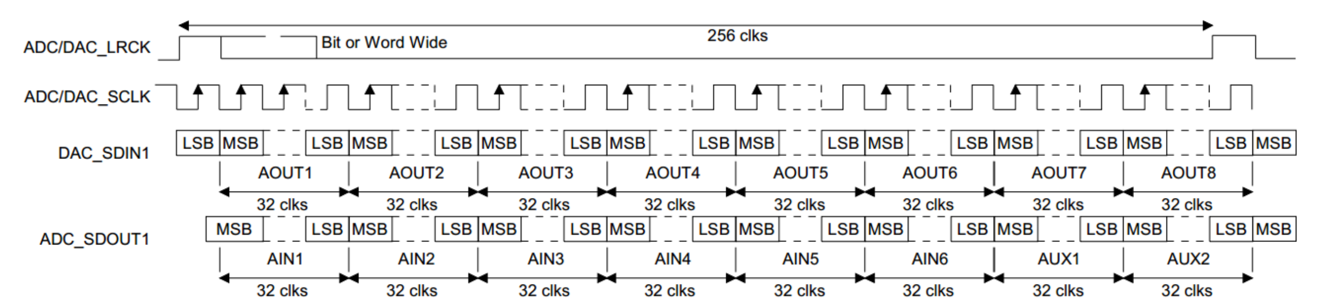

Figure 4. TDM format

TDM data is received 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.

NOTE

`DAC_SCLK` must operate at 256 sample rate.

The code in Example 1 gives the CS42448 codec specific configuration for reference.

Example 1

```
cs42888_config_t cs42448Config = {
    .DACMode      = kCS42888_ModeSlave,
    .ADCMode      = kCS42888_ModeSlave,
    .reset        = BORAD_CodecReset,
    .master       = false,
    .i2cConfig    = {.codecI2CInstance = DEMO_I2C_INSTANCE, .codecI2CSourceClock =
BOARD_CODEC_I2C_CLOCK_FREQ},
    .format       = {.mclk_HZ = 24576000U, .sampleRate = 48000U, .bitWidth = 24U},
    .bus          = kCS42888_BusTDM,
    .slaveAddress = CS42888_I2C_ADDR,
};
```

3.3 FLAC decode process

The FLAC decoding process in this application includes the following steps. The code in Example 2 gives the specific process for reference.

1. Create a new Decoder.
2. MD5 checking.
3. Initialize, retrieve the FLAC file.
4. FLAC starts decoding until the end.
5. Delete Decoder.

Example 2

```
static void DEMO_FlacDecode(void)
{
    static FLAC__bool ok = true;
    static FLAC__StreamDecoderInitStatus init_status;
    /* Creat a new decoder. */
    if((g_decoder = FLAC__stream_decoder_new()) == NULL)
    {
        flac_printf("ERROR: allocating decoder.\r\n");
        assert(false);
     }
     (void)FLAC__stream_decoder_set_md5_checking(g_decoder, true);
     init_status = FLAC__stream_decoder_init_file(g_decoder, fileName, write_callback,
metadata_callback, error_callback, &s_fileObject);
     if(init_status != FLAC__STREAM_DECODER_INIT_STATUS_OK)
     {
         flac_printf("ERROR: initializing decoder: %s\n",
FLAC__StreamDecoderInitStatusString[init_status]);
         ok = false;
      }
      if(ok) {
         ok = FLAC__stream_decoder_process_until_end_of_stream(g_decoder);
         flac_printf("decoding: %s\r\n", ok? "succeeded" : "FAILED");
         flac_printf("   state: %s\r\n",
FLAC__StreamDecoderStateString[FLAC__stream_decoder_get_state(g_decoder)]);
       }
       FLAC__stream_decoder_delete(g_decoder);
}
```

Taking 8-channel FLAC decoding as an example, the `write_callback` of FLAC decoder is triggered in the unit of blocksize. In this application, a blocksize contains eight channels of audio data stream, each channel of audio data stream contains 4608 samples, and each sample contains four bytes of data. In the `write_callback` function, convert the decoded data that SAI can send to the codec for playback. Figure 5 shows the specific software processing.

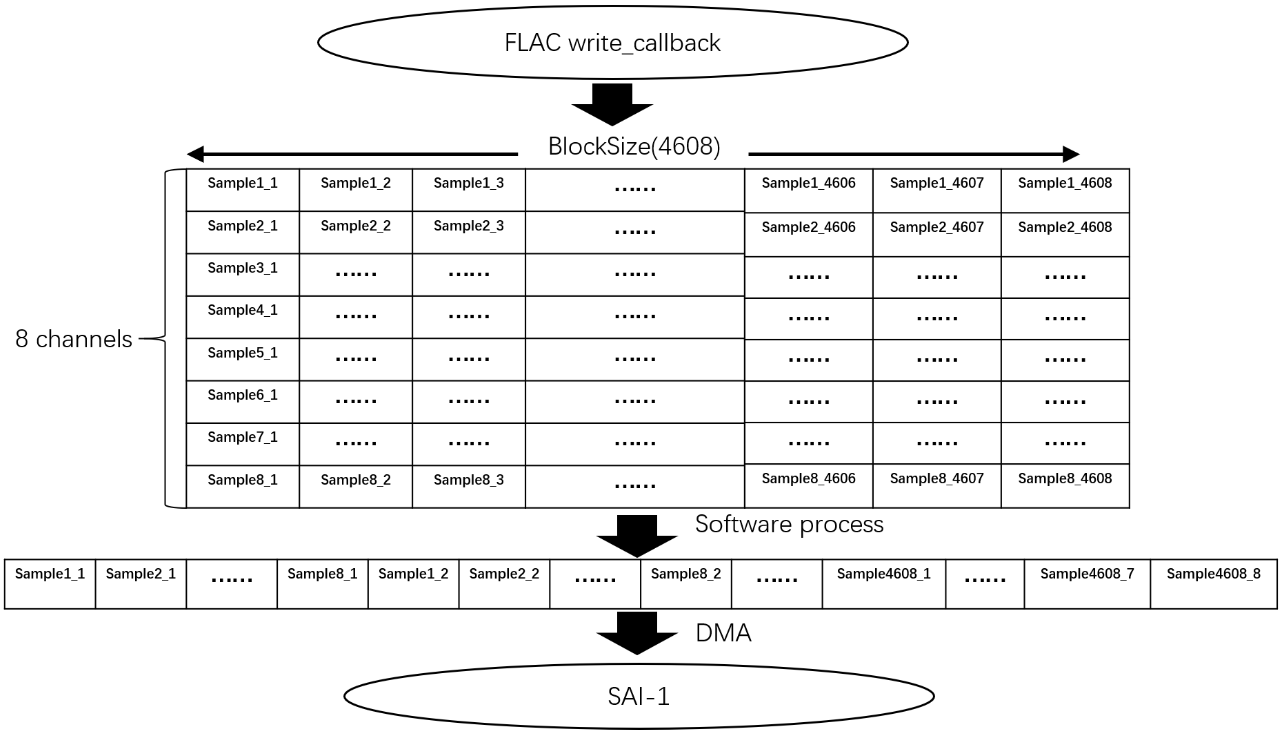

Figure 5. FLAC software process

4 Audio quality measurement

shows the system block diagram of the CS42448 audio card. It can support eight channels of audio data output (DAC) and six channels of audio data input (ADC). To know whether the audio signal quality after Audio Card processing meets the application requirements, the audio card needs to be measured around six major test items: Level&Gain, THD+N, Frequency Response, Signal to Noise, Crosstalk, Inter-channel Phase. In addition, to measure the audio quality of the audio card, a professional audio quality analyzer APx525 is required, which includes the following features:

- Two analog input channels (APx525)
- AES/SPDIF digital I/O
- Typical THD+N < -110 dB
- > 1 MHz bandwidth @ 24 bits on two channels [with BW52 option]
- 1.2 M point FFTs
- Powerful automation and sophisticated reporting
- Support for the complete range of APx digital I/O options

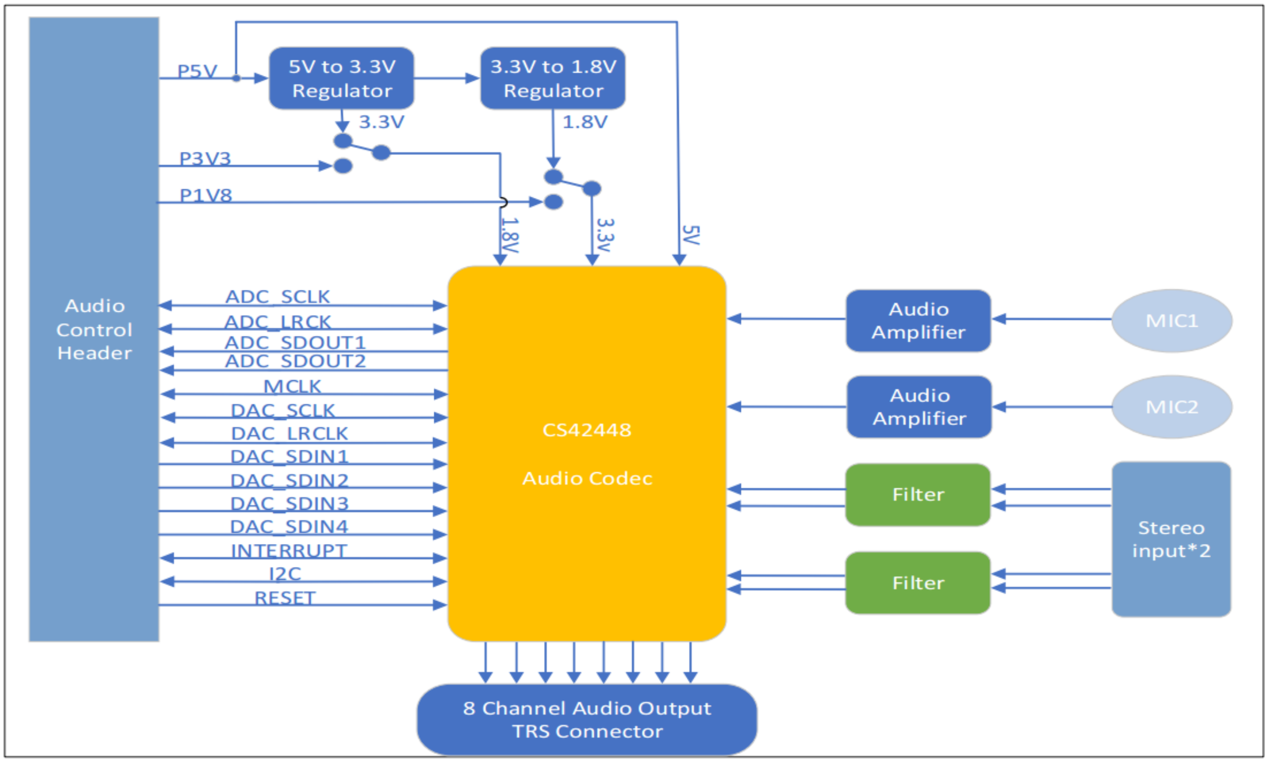

Figure 6. Audio card system block diagram

4.1 Audio card DAC measurement

To measure the audio quality after Audio Card's DAC processing, perform the following steps:

1. APx525's DIGITAL SERIAL I/O (MCLK, LRCLK, BCLK, TXDATA0, TXDATA1) connect to CS42448 Audio Card's SAI interface (MCLK, DAC_LRCLK, DAC_SCLK, DAC_SDIN1, DAC_SDIN2).
2. CS42448 Audio Card's LINE1&LINE2 OUT1 connect to APx525's ANALOG INPUT 1, CS42448 Audio Card's LINE1&LINE2 OUT1 connect to APx525's ANALOG INPUT 2.
3. CS42448 configuration: 48 KHz, 24 bits width, Classic I2S Mode.
4. Figure 7 shows the input and output configuration of APx525.
5. Select six audio test items in the APx525 software operation interface and configure them to automatically generate test reports. For details of the test report, see AN13291SW.

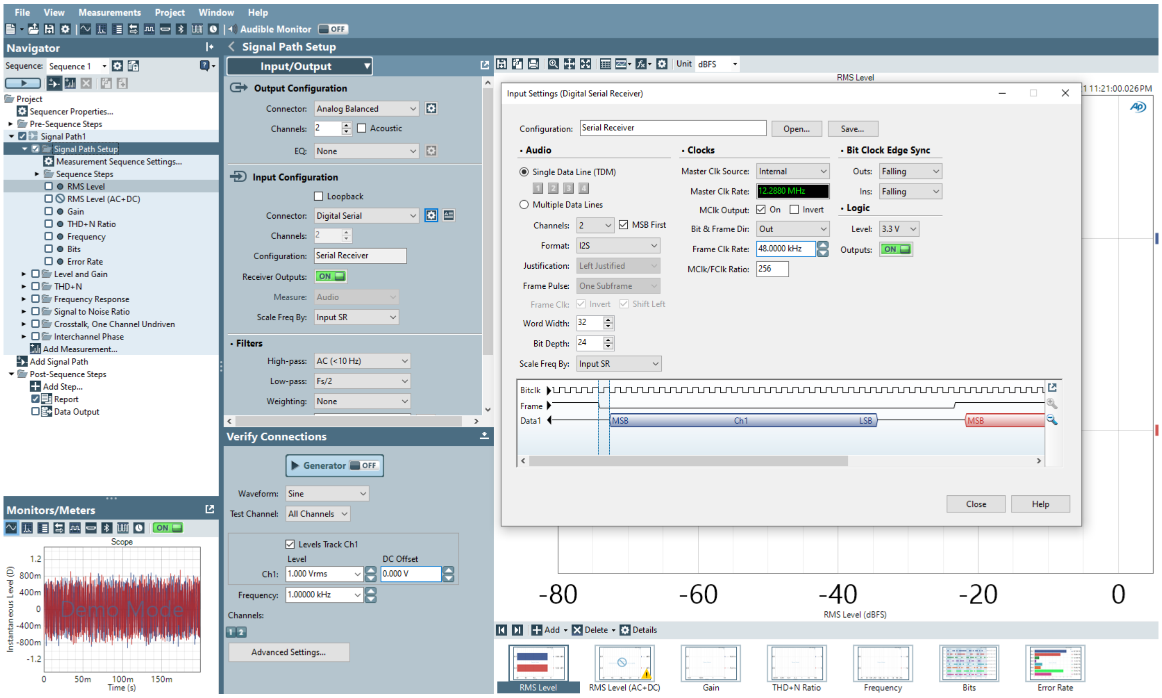

Figure 7. APx525 configuration window

Figure 8 and Figure 9 show the connection diagrams.

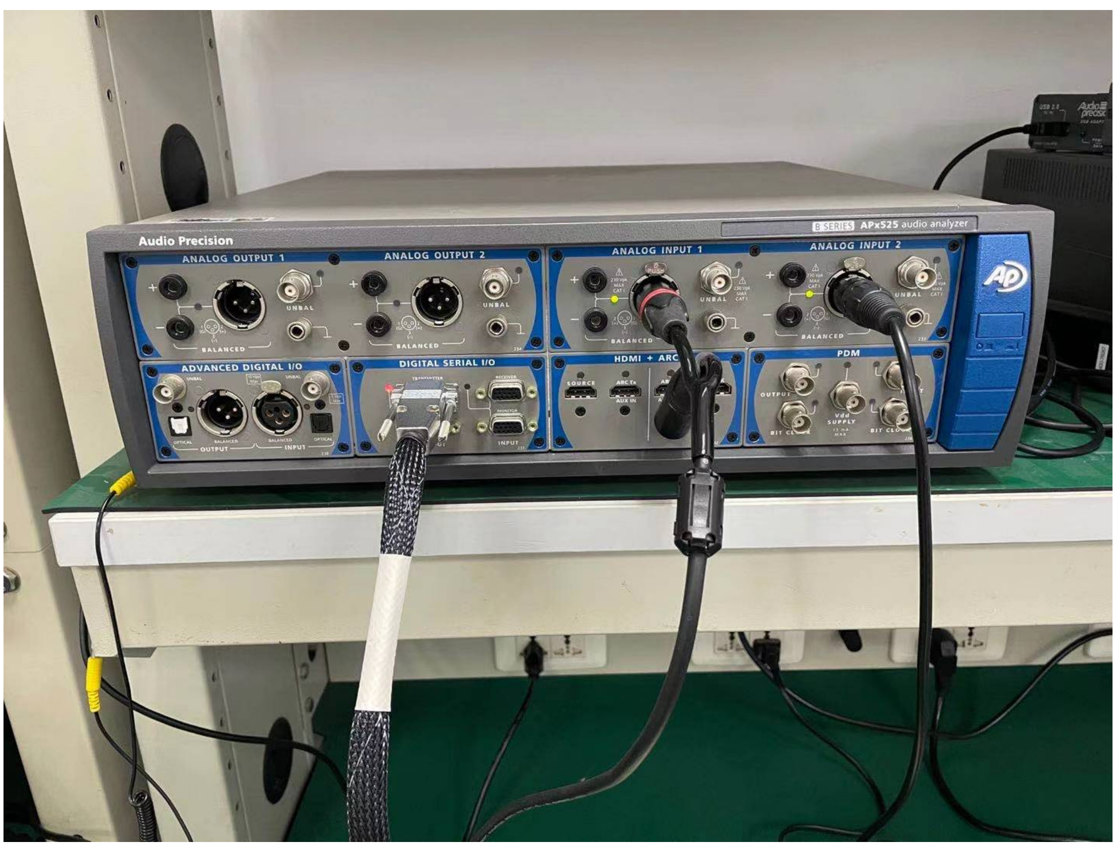

Figure 8. Connection diagram A

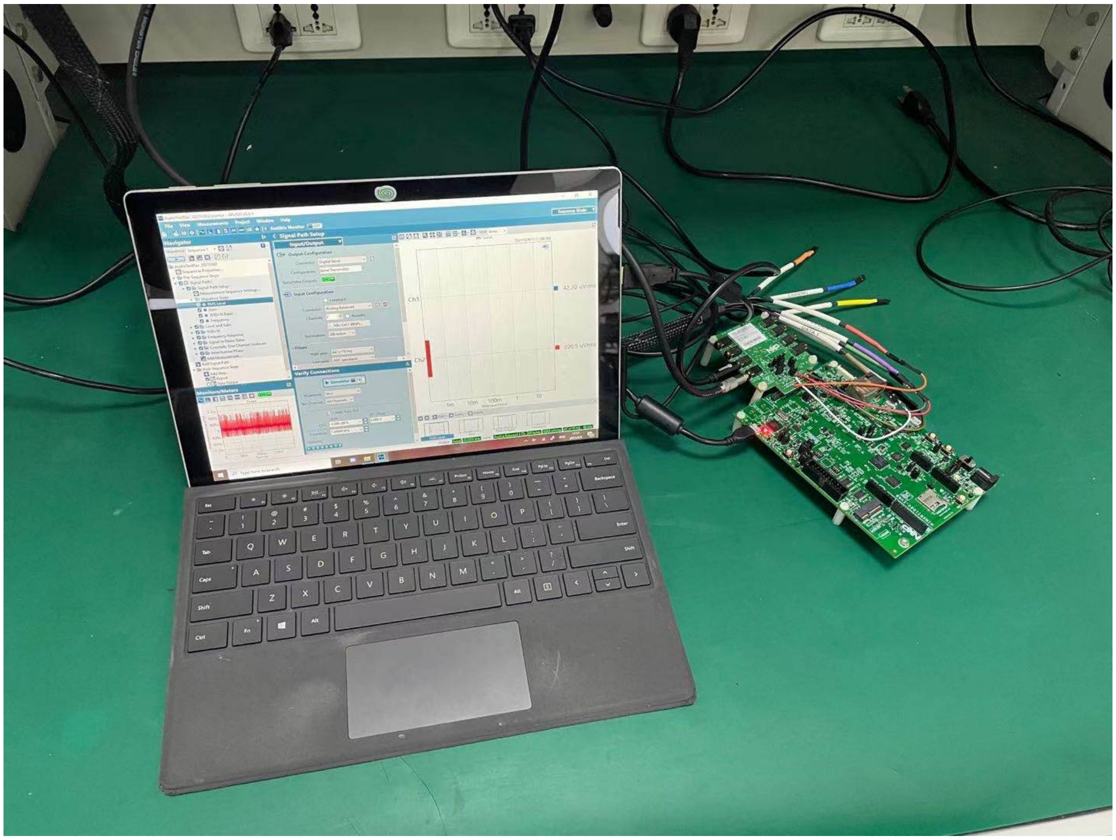

Figure 9. Connection diagram B

4.2 Audio card ADC measurement

To measure the audio quality after Audio Card's ADC processing, perform the following steps:

1. APx525's ANALOG OUTPUT 1 and ANALOG OUTPUT 2 connect to CS42448 Audio Card's Stereo INPUT1 and Stereo INPUT2.
2. CS42448 Audio Card's SAI interface (MCLK, ADC_LRCLK, ADC_SCLK, ADC_SDOUT1, ADC _SDOUT2) connect to APx525's DIGITAL SERIAL I/O (MCLK, LRCLK, BCLK, TXDATA0, TXDATA1).
3. CS42448 configuration: 48KHz, 24bits width, Classic I2S Mode.
4. Figure 10 shows the input and output configuration of APx525.
5. Select six audio test items in the APx525 software operation interface and configure them to automatically generate test reports. For details of the test report, see AN13291SW.

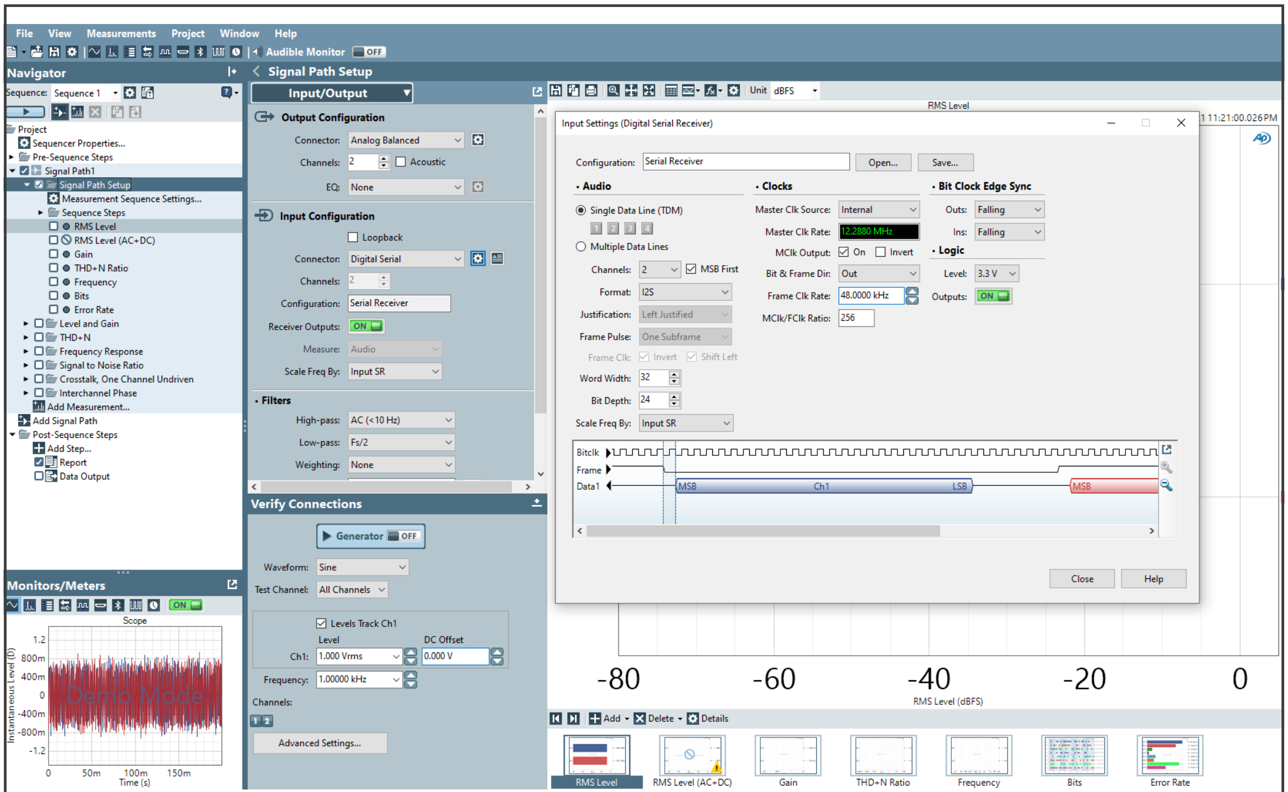

Figure 10. APx525 configuration window

Figure 11 shows the connection diagram.

Figure 11. Connection diagram

5 Conclusion

This application note introduces:

1. How to realize Flac7.1 decoding and playback based on RT1060 EVKB board and audio card.
2. How to use APx525 to do audio quality analysis for the CS42448 audio card.

6 References

1. *i.MX RT1060 Processor Reference Manual* (document IMXRT1060RM)
2. AUD-EXP-42448 Schematic(Rev B)
3. MIMXRT1060-EVKB Schematic(Rev B)
4. CS42448 Data Sheet
5. AUDIO PRECISION-APX525-Datasheet

7 Revision history

Revision number	Date	Substantive changes
0	10 June 2021	Initial release

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 10 June 2021

Document identifier: AN13291