AN14631

How to Connect Asynchronized Audio Source and Sink with RT1170 ASRC

Rev. 1.1 — 6 May 2025 Application note

Document information

Information	Content
Keywords	AN14631, i.MX RT1170, ASRC, MCUXpresso
Abstract	This application note explains the problem that occurs when connecting asynchronized audio source and sink without any data rate mating processing and the possible solutions.

1 Introduction

In a digital audio system, if the audio source component (for example the ADC) and the audio sink component (for example the MCU / DSP RX port) both have to run in the master clock mode (they are in a different clock domain), they have a different stream data rate. This is a challenge to the developer. Special audio data processing/converting must be performed, or the audio received or played back by the audio sink must be in a poor quality with audible noise or glitches.

The RT1170 series of MCUs have a peripheral called Asynchronous Sample Rate Converter (ASRC). This peripheral can be used to match the digital audio signals of different bit rates or different audio clock domains.

This application note is the first one in the “how to connect asynchronized audio source and sink” series. The other two in this series have the same topic but with using a software ASRC algorithm on RT600 and RT10xx.

This application note explains the problem that occurs when connecting asynchronized audio source and sink without any data rate mating processing and the possible solutions. An RT1170 MCUXpresso ASRC example project and an LPC55S69 MCUXpresso signal generator project are provided together with this application note. By running, testing, and analyzing these accompanied example projects, we can understand how the RT1170 ASRC and the related DMA channels are configured and how to ensure that the signal chain is stable and achieving the best audio performance.

The concept and ideas introduced in this document also apply to the RT1160 and RT1180 series MCUs, because they have the same ASRC peripheral as the RT1170.

2 Overview

This section provides an overview of the application.

2.1 Digital audio source and sink

For a digital audio system, we have the following digital audio sources and components:

- An ADC converter that generates the I2S audio data.
- A PDM microphone that generates the PDM audio stream.
- A USB audio host that downstreams the digital audio.
- A DVD/Blu-Ray disk player that sends out the SPDIF signal.
- An audio player (decoder) in the firmware that generates the audio of a certain sampling frequency, which is specified by the audio file (WAV, OPUS, MP3, or any other media formats).
- A wireless audio receiver or an Ethernet audio receiver that streams out the received audio of a certain sampling frequency, which is determined by the transmitting side.

For a digital audio system, we have the following digital audio sinks and components:

- A DAC converter that receives the I2S audio data.
- A USB audio host that upstreams the digital audio.
- An SPDIF digital transmitter.
- An audio encoder in the firmware that compresses the received audio to a file (WAV, OPUS, MP3, or any other media formats) with a specified sampling frequency.
- A wireless audio transmitter or an Ethernet audio transmitter.

2.2 Audio source and sink synchronization

This section provides examples of the audio source and sink synchronization.

2.2.1 Example of synchronized audio source and audio sink

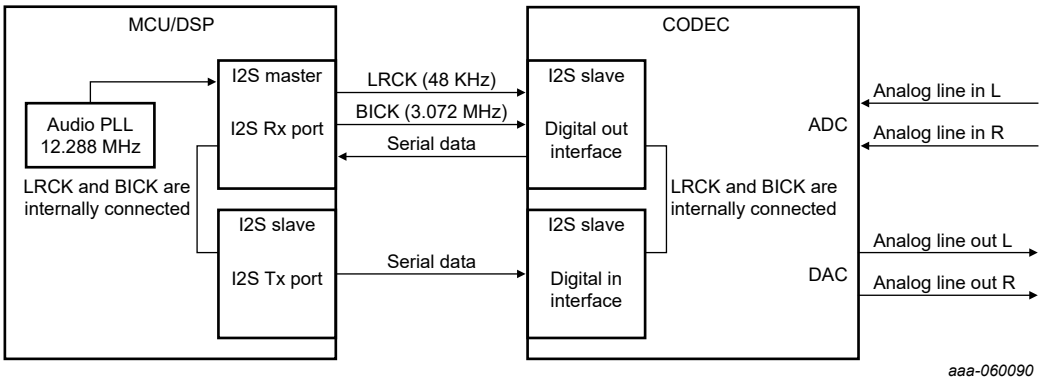


Figure 1. Synchronized audio source and sink

The codec and MCU I2S ports are all working at the audio PLL sample root clock.

In a certain period of time, the codec ADC as the audio source is generating exactly the same amount of audio samples as the codec DAC is expecting. In this period of time, the MCU is moving exactly the same amount of audio samples from the receiving buffer to the transmitting buffer.

2.2.2 Example of asynchronized audio source and audio sink

In this example, the BT chip is generating the I2S clock and slaves the MCU I2S receiving port. The MCU I2S transmitting port is also generating the I2S clock and slaves the DAC. Even though the BT chip is receiving and transmitting 48-kHz audio and the MCU also configures its I2S transmitting port to 48 kHz, the two 48-kHz sample rates are only nominally the same. In fact, they are different. The 48 kHz on the MCU's I2S receiving port side could be 48.001 kHz and the 48 kHz on the MCU's I2S transmitting port side could be 48.002 kHz.

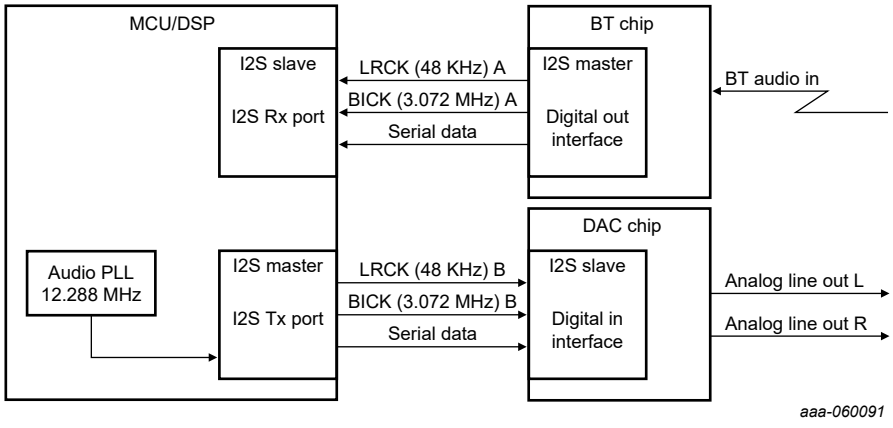


Figure 2. Asynchronized audio source and sink

2.3 Problem of connecting asynchronized audio source and sink

Usually, a digital audio system uses DMA to move audio data and handles the audio stream in framed audio. For example, we can set the frame size to 48 samples for a Fs = 48 kHz system. Each frame (or every 1 ms) after the DMA RX finishes moving 48 samples from the I2S receiving port to the RX buffer, it generates a DMA RX interrupt. When the DMA TX finishes moving 48 samples from the TX buffer to the I2S transmitting port, it generates a DMA TX interrupt.

We can create a circular audio buffer of 480 samples (10 ms) in the firmware, let the DMA RX interrupt write (put) the received 48 samples (1 ms) to this circular buffer, and let the DMA TX interrupt read (take) 48 samples (1 ms) from this circular buffer and put them into the DMA TX buffer.

Consider the synchronized example in Section 2.2.1. The DMA RX interrupt always has the same phase (pace) as the TX interrupt, because the I2S receiving and transmitting ports are both working at the exactly same clock, which is derived from the same root clock (audio PLL3). The 480-sample (10-ms) circular buffer always has a fixed Amount Of Audio (AOD), say 5-ms audio. Because the DMA RX interrupt puts 48 samples (1 ms) to the circular buffer, the DMA TX interrupt must take 48 samples (1 ms) out from the circular buffer.

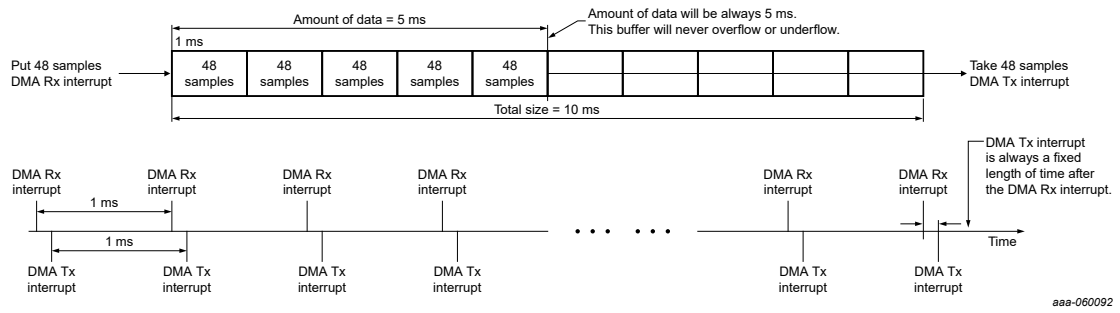


Figure 3. Source and sink are synchronized, AOD in the buffer is stable

Consider the asynchronized example in Section 2.2.2. Because the I2S receiving port and transmitting port are actually working at different I2S clock frequencies, the DMA RX interrupt does not always occur after the DMA TX interrupt. The AOD (Amount Of Audio) in the 480-sample (10-ms) circular buffer must be either dropping down from a certain value (say 5 ms) to 0 ms (this is called underflow), or increasing from a certain value (say 5 ms) to 10 ms (this is called overflow).

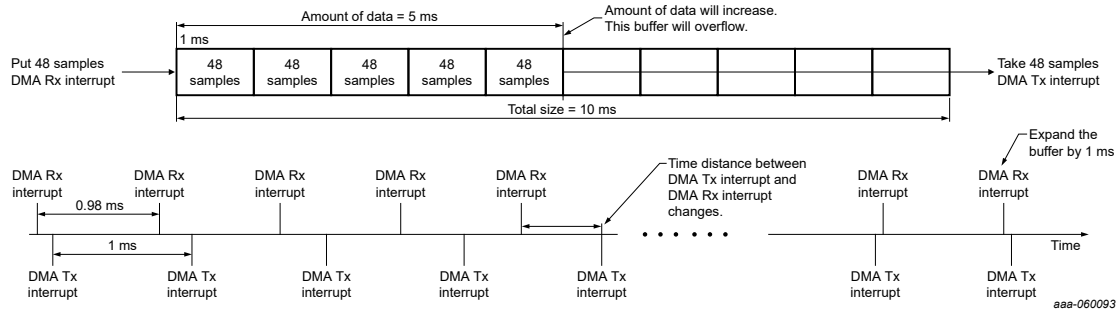


Figure 4. Source and sink are asynchronized, buffer will overflow

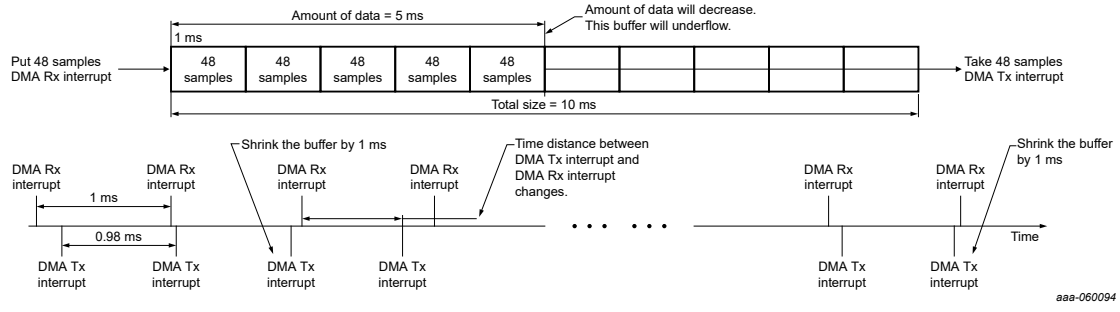


Figure 5. Source and sink are asynchronized, buffer will underflow

When the circular buffer overflows, it means that more audio data is supplied from the I2S receiving port than what the I2S transmitting port needs. This causes the waste of useful audio, some audio wiill be discarded, and the sound (audio signal) is broken.

Figure 6. Useful audio data is thrown

When the circular buffer underflows, it means that less audio data is supplied from the I2S receiving port than what the I2S transmitting port needs. This causes the lack of needed audio, and the sound (audio signal) is broken again.

Figure 7. Lack of audio data and have to put zeros

2.4 Solutions of connecting asynchronized audio source and sink

To avoid the broken signal, the source must generate exactly the same amount of audio data as the sink needs, so that the circular audio buffer never overflows or underflows. We have the following solutions:

- Avoid multiple audio clock masters. Let the MCU / DSP be the master audio clock for all the connected audio components.
- If having multiple audio clock masters is unavoidable, then try to adjust the MCU / DSP local audio clock PLL in real time according to the AOD of the receiving buffer to let the local audio streaming data rate follow the data rate of the external master in real time.
- If real-time audio PLL adjusting is not available, let the MCU / DSP send information to the external master to request more or less data from the master. This is the “feedback” solution used by the USB audio ASYNC mode.
- If all of the above solutions are not possible, the MCU / DSP must resolve the problem using ASRC: either hardware ASRC or a software ASRC algorithm.

3 Connecting asynchronized audio source and sink with RT1170 ASRC

This section describes the connection of asynchronized audio source and sink with RT1170 ASRC.

3.1 System structure

To test the RT1170 ASRC, we need the following boards/devices:

- A DVD Player/BlueRay player
- An LPC55S69 EVK board
- An RT1170EVKB board
- A logic analyzer
- A PC with audio-editing application (such as Ocenaudio or Audacity)

This document comes together with two MCUXpresso projects:

- Lpc55S69_I2sDmaTxRxWithSweepGen
- RT1170EvkB_AudioIoWithAsrc

On the LPC55S69EVK, we run the "Lpc55S69_I2sDmaTxRxWithSweepGen" project to generate the testing audio sweeping signal and monitor the audio sent from the RT1170EVKB. On the RT1170EVKB, we run the ASRC testing project "RT1170EvkB_AudioIoWithAsrc".

A PC with an audio-editing application and a logic analyzer are used to record and capture digital audio, either through USB or I2S.

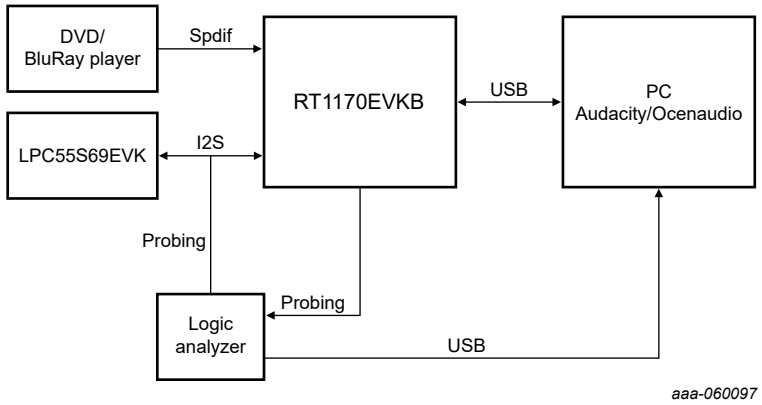


Figure 8. System block

Note: *MCUXpresso IDE v11.10.0 is the IDE for building the RT1170 and LPC55S69 projects. The "Getting Started with MCUXpresso SDK for MIMXRT1170-EVK.pdf" reference is in the RT1170 SDK. See this document for details on building and running the example project. Download the IDE at https://www.nxp.com/design/design-center/software/development-software/mcuxpresso-software-and-tools-/mcuxpresso-integrated-development-environment-ide:MCUXpresso-IDE#downloads.*

3.2 LPC55S69EVK as the I2S audio source

Build and run the "Lpc55S69_I2sDmaTxRxWithSweepGen" project on the LPC55S69 EVK board so that the LPC55S69EVK board streams the testing audio to/from its I2S (Flexcomm) port pins. It is the I2S master external audio source and sink for the 1170EVKB board.

In the system, the LPC55S69EVK board is the I2S clock master, so that it is an asynchronized system.

The "Lpc55S69_I2sDmaTxRxWithSweepGen" project is based on the "lpcxpresso55s69_i2s_dma_record_playback" example in "SDK_2_16_000_LPCXpresso55S69".

It configures and activates three Flexcomm ports:

- FC1: to transmit audio data to the external RT1170EVKB board (SAI4 RxData).
- FC2: to receive audio data from the external RT1170EVKB board (SAI4 TxData).
- FC7: to transmit audio data to a local codec chip, so that you can hear the audio streamed through FC2.

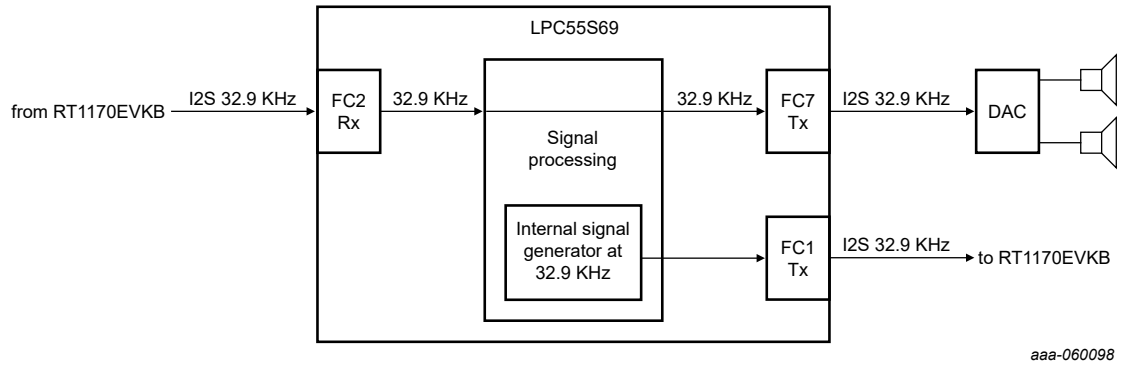


Figure 9. LPC55S69 internal signal flow

These three Flexcomm (I2S) ports work at the same clock source, which is shared from FC7. FC7 is the I2S clock master, FC1 and FC2 are the I2S clock slaves.

By default, the "Lpc55S69_I2sDmaTxRxWithSweepGen" configures its own I2S clock Fs to 32967 Hz (just a frequency value that is different from 48 kHz, no other special meaning). The internal sweeping tone generator generates a 0-dB Fs, 20-second chirp signal, from 20 Hz to 16.4835 kHz (the Nyquist frequency of 32967 Hz) for both the L and R channels of the FC1(I2S1) port. The R channel is the opposite of the L channel. This chirp signal is sent to the 1170EVKB SAI4 RdData. The FC2 (I2S2) receives the audio data sent from the 1170EVKB SAI4 TxData, and the FC7 streams out what the FC2 receives to the onboard codec chip WM8904. You can hear the audio transmitted from the 1170EVKB SAI4 TxData.

The sweeping generator in the "Lpc55S69_I2sDmaTxRxWithSweepGen" project is logarithmic and based on phase accumulation. To reach the highest precision within the available MIPS(MCPS), we use the float-type calculation. The two math functions needed for the calculation are "sinf" and "powf". Directly calling these two functions uses too much time. Replacing the "powf" function by a fast power-calculation algorithm, which is introduced by “Harrison Ainsworth / HXA7241 (2004-2011)” (https://github.com/hxa7241/powfast). The sweeping generator is then fitted to the available MIPS. We balanced the FastPower function lookup table to the maximum possible size, so that FastPower can provide the best accuracy. Figure 26 is the spectrum of the 20-s sweeping. The noise floor in the spectrum is mostly below -108 dB, which is good enough for analyzing.

3.3 RT1170EVKB board rework

To run the RT1170 ASRC testing project "RT1170EvkB_AudioIoWithAsrc", rework the RT1170EVKB board and use the flying wires connecting the LPC55S69 EVK board with the RT1170 EVKB board.

- To connect the SPDIF input signal:
  - Install J35 and J36 (short J35 and J36).
  - Install T1 (inductor and TTWB1010L) and J45.
  - Install R295 (10 kΩ).
  - Remove R209 to disconnect the SPDIF input signal from capacitor C324.
- To connect the external I2S audio source/sink to RT1170 SAI4:
  - Connect a wire to R348 (the side that does not connect to R34). This is for the logic analyzer probing.
  - Connect a wire to R347 (the side that does not connect to R35) to get the external LRCK input and logic analyzer probing.
- To have the four debug GPIO pins for the logic analyzer probing:
  - Add 0-Ω R1809 and R1810 to watch DbgPin3,4.
  - Remove C573 to have a quicker DbgPin4.
- To probe the SAI1 clock and data signal:
  - Connect a wire from J97 for the logic analyzer probing.

- Connect a wire from J98 for the logic analyzer probing.
- Connect a wire from J99 for the logic analyzer probing.
- Connect a wire from J100 for the logic analyzer probing.

- To connect LPC55S69EVK and RT1170EVKB, see Figure 10:
  - SAI4 TX Data and GPIO_LPSR_02 at R348 (the side that does not connect to R34): to connect with LPC55S69EVK P18.15.
  - SAI4 RX Data and GPIO_LPSR_06 at J78.14: to connect with LPC55S69EVK P18.3.
  - SAI4 Tx Sync (LRCK) and GPIO_LPSR_03 at R347 (the side that does not connect to R35): to connect with LPC55S69EVK P17.12.
  - SAI4 Tx BCLK and GPIO_LPSR_12 at J78.10: to connect with LPC55S69EVK P17.14.
  - GND, J78.11, J78.12, and J9.1: to connect with LPC55S69EVK P17.7, P23.8, and P24.8.

  Note: *To start and debug the program, the SAI4 TX Sync (LRCK) GPIO_LPSR_03 is also used as the BOOT_MODE select. If you have this pin connected with the LPC55S69EVK board before powering it on, the RT1170 does not start up. To have the system working properly, connect the SAI4 TX Sync with the LPC55S69EVK board after applying power. In a debugging session, make sure that the SAI4 TX Sync is connected after powering on. If the debugging does not start up correctly, try to remove power and disconnect the SAI4 TX Sync pin and start over again.*

  Note: *The voltage of the I2S signals connected between the two EVK boards is 3.3 V. Make sure that on the LPC55S69EVK board, P4 has 2-3 closed (the 3.3 V side). The RT1170EVKB has the I2S signal voltage fixed at 3.3 V.*

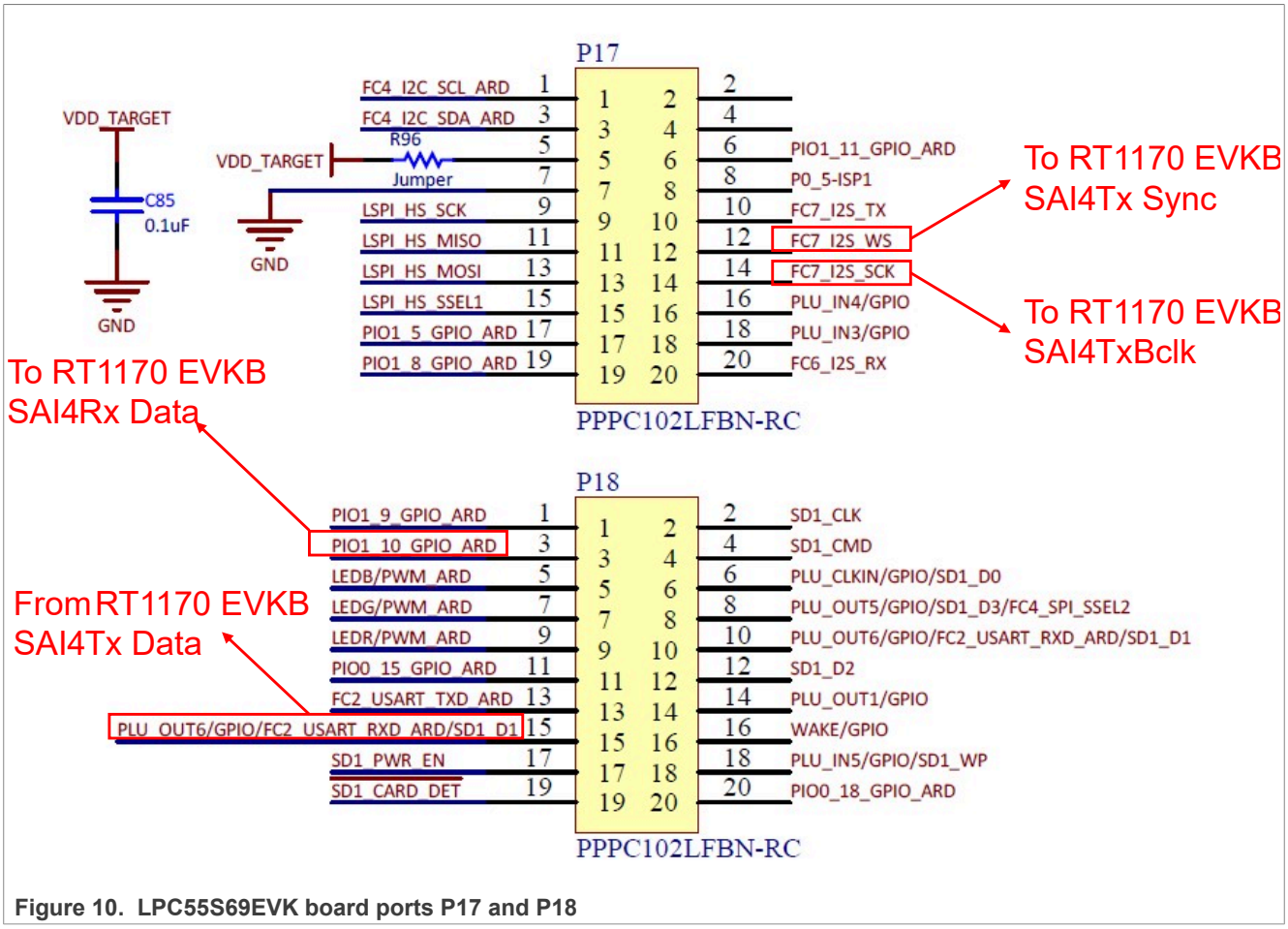


Figure 10. LPC55S69EVK board ports P17 and P18

3.4 RT1170 ASRC MCUXpresso project

The "Lpc55S69_I2sDmaTxRxWithSweepGen" project is an audio DMA interrupt-driven program. It configures all the audio-related peripherals and waits for the audio DMA interrupt. It is bare-metal without RTOS. The most critical and valuable parts of this program are the sophisticated configurations to the audio peripherals. Different from the SDK examples, this program directly manipulates the registers of all the audio peripherals. A more strict and precise start-up timing, non-interleaved DMA audio buffer, and better handling to the external signal dropping down is achieved.

3.4.1 SAI configuration

The testing project uses SAI1 and SAI4. SAI1 is the local I2S master. It drives the external I2S clock lines to the onboard CODEC. SAI4 is configured as the I2S slave. It receives the I2S clock from the connected LPC55S69EVK.

SAI1 is configured with the following key features:

- The SAI1 clock comes from the RT1170 audio PLL (the same clock source as PDM).
- The TX SYNC and BCLK are in the master mode and drive the I2S sampling frequency at 48 kHz.
- The RX SYNC and BCLK follow the TX (it can be considered as the I2S slave).
- Each sample contains two parts of 32-bit data: left 32 bits + right 32 bits.
- The TX FIFO request to the DMA is enabled.
- The RX FIFO request to the DMA is enabled.

SAI4 is configured with the following key features:

- The TX SYNC and BCLK are in the slave mode and driven by the connected LPC55S69EVK.
- The RX SYNC and BCLK follow the TX (it can be considered as the I2S slave).
- Each sample contains two parts of 32-bit data: left 32 bits + right 32 bits.
- The TX FIFO request to the DMA is not enabled.
- The RX FIFO request to the DMA is not enabled.

3.4.2 PDM configuration

Eight PDM microphones are enabled in the testing project. However, only mic0 (U40) and mic1 (U41) are generating the real audio. The testing project calls the FSL driver API function to initialize the PDM interfaces to the following key features:

- Eight PDM microphones (channels) are enabled.
- The PDM sampling frequency is 48 kHz.
- The PDM clock comes from the RT1170 audio PLL (the same clock source as SAI1).

3.4.3 SPDIF configuration

The following key features are configured for SPDIF:

- The SPDIF RX source comes from the SPDIF input pin.
- The SPDIF request to the DMA is disabled.
- The SPDIF clock-locked flag triggers the SPDIF interrupt.
- The SPDIF clock lost flag triggers the SPDIF interrupt.
- The SPDIF receiver automatic resynchronization of FIFOs is enabled.

3.4.4 ASRC configuration

The RT1170 ASRC has three pairs. Each pair can convert one or several subchannels from the input side clock domain to the output side clock domain. This testing project uses all three pairs.

To let the ASRC pair working properly, configure the following major items:

- The input side data rate. It could be a clock source for the input side, or derived from a ratio value together with the clock for the output side.
- The output side data rate. It must be a clock source.
- How to divide the input clock to get the desired input signal sampling frequency.
- How to divide the output clock to get the desired output signal sampling frequency.
- What preprocessing should be used.
- What postprocessing should be used.
- The ideal ratio value of the Fs input side / Fs output side. This value is optional.

In the testing project, pair A is selected to convert the external SPDIF input or to convert the MCU internal signal at a certain Fs that is different from the local MCU I2S Fs.

When pair A is selected for the external SPDIF, it is configured with the following key features:

- The input side clock is set to the SPDIF RX clock and divided by 128 to have the right SPDIF RX Fs for the input side.
- The output side clock is set to SAI1 RX BCLK and divided by 64 to have the right Fs (local I2S Fs) for the output side.
- For the input side, the Fs ratio is used and the ratio comes from the ASRC measuring the input / output clock.
- The automatic selection of preprocessing and postprocessing is enabled.

When pair A is selected as the internal signal source, it is configured with the following key features:

- The input side clock is set to NONE.
- The output side clock is set to SAI1 RX BCLK and divided by 64 to have the right Fs (local I2S Fs) for the output side.
- For the input side, the Fs ratio is used and the ratio comes from the ideal ratio register.
- The ideal ratio register is to be filled with a proper value in case of different testing Fs.
- The automatic selection of preprocessing and postprocessing is disabled.
- A proper preprocessing is selected in case of different testing Fs.
- A proper postprocessing is selected in case of different testing Fs.

Pair B is used to convert the received external I2S signal. Here are the key features:

- The input side clock is set to SAI4 TX BCLK and it divided by 64 to have the right Fs for the input side.
- The output side clock is set to SAI1 RX BCLK and divided by 64 to have the right Fs (local I2S Fs) for the output side.
- For the input side, the Fs ratio is used and the ratio comes from the ASRC measuring the input / output clock.
- The automatic selection of preprocessing and postprocessing is enabled.

Pair C is used to convert the local I2S signal to the transmitted one. Here are the key features:

- The input side clock is set to SAI1 RX BCLK and divided by 64 to have the right Fs (local I2S Fs) for the input side.
- The output side clock is set to SAI4 TX BCLK and divided by 64 to have the right Fs for the output side.
- For the input side, the Fs ratio is used and the ratio comes from the ASRC measuring the input / output clock.
- The automatic selection of preprocessing and postprocessing is enabled.

3.4.5 DMA configuration

The SAI, PDM, ASRC are the peripherals to transmit / convert / receive digital audio data with their RX / TX data registers. How to have these RX/TX data in the registers be moved into / from the MCU memory, form the framed-audio data structure, and generate the interrupt at the end of each audio frame is the job of the DMA.

In this testing project, nine DMA channels are used to move the audio data. The following figures explain the main configuration and running procedure of each DMA channel.

The dash line in the drawing means data moving, solid line means address jumping. Each data moving is triggered by the assigned events configured for the DMA channel. After a data moving, the source and destination addresses change (jump) accordingly. The numbers in the drawing clearly tell the sequence of all the data moving in a major DMA loop. When a major DMA loop is finished, the next TCB is loaded and the next DMA major loop starts. This makes the DMA moving to continue forever and it is called Chained DMA.

3.4.5.1 DMA Ch0: for SAI1 TX - moving audio from memory to SAI1 TX register

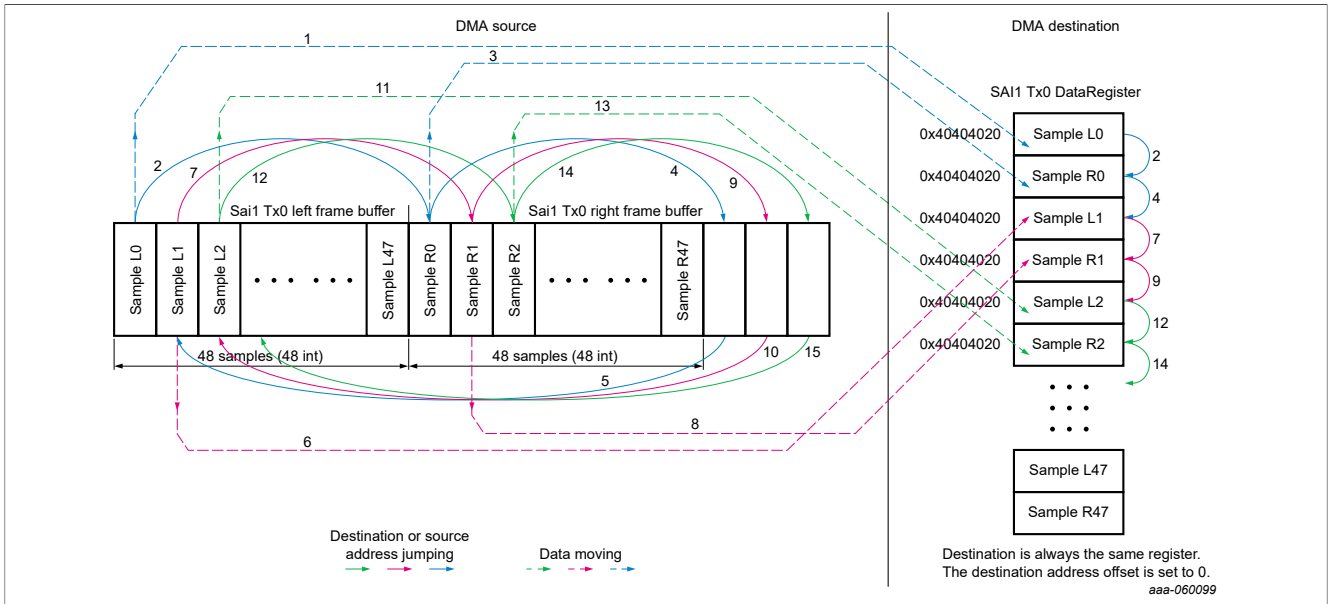


Figure 11. DMA Ch0

3.4.5.2 DMA Ch1: for SAI1 RX - moving audio from SAI1 RX register to memory

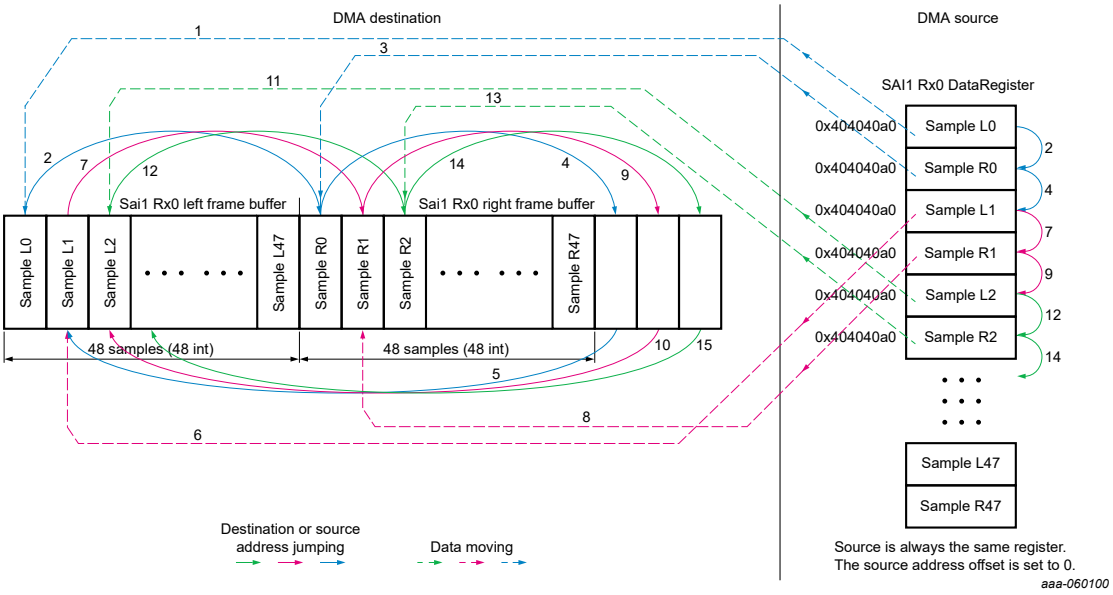


Figure 12. DMA Ch1

3.4.5.3 DMA Ch2: for DMIC Input - moving audio from DMIC RX registers to memory

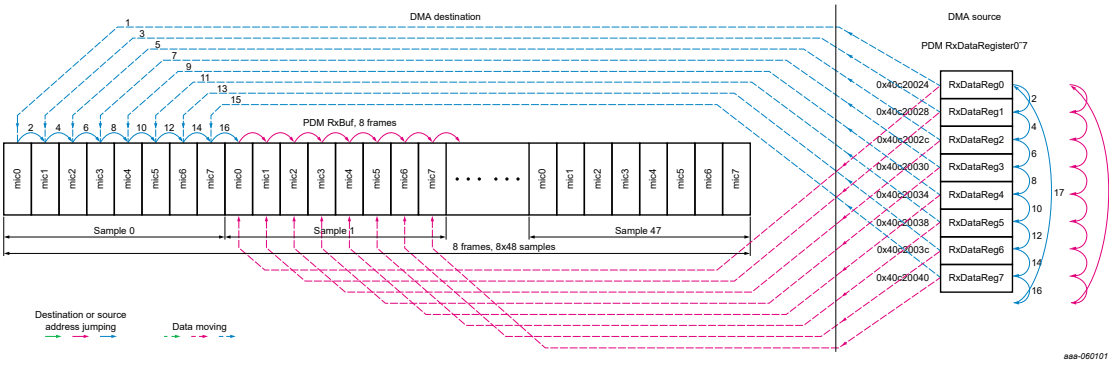


Figure 13. DMA Ch2

3.4.5.4 DMA Ch3: for ASRC pair A input - moving audio from SPDIF RX register (or audio in memory) to ASRC pair A input data register

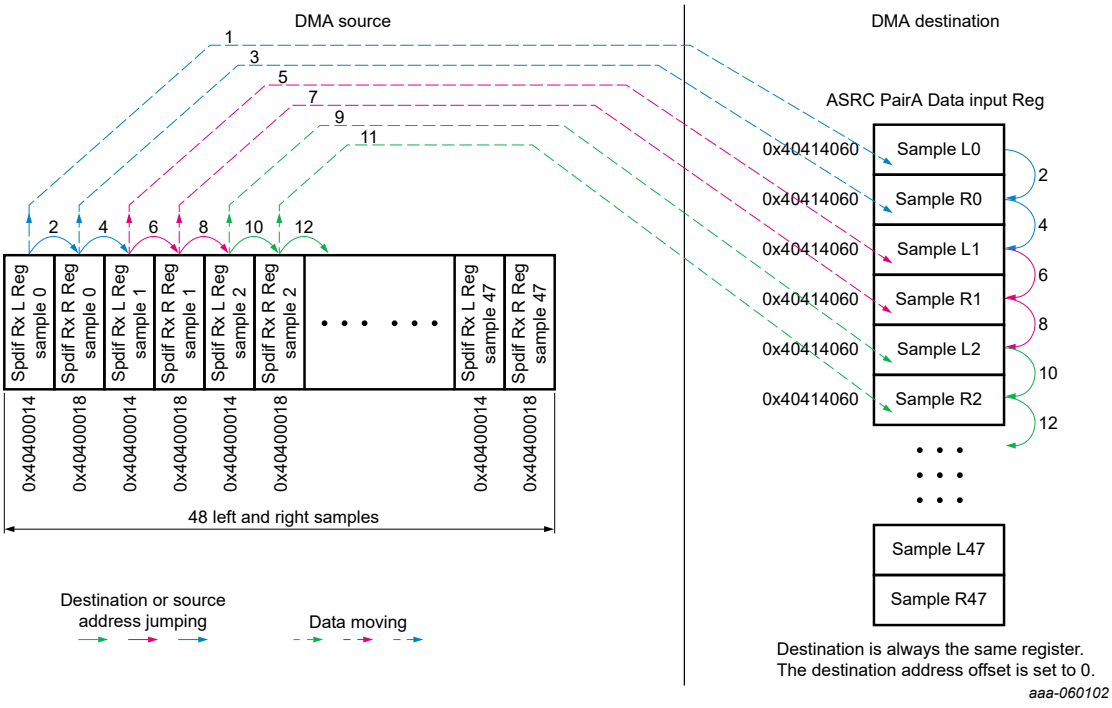


Figure 14. DMA Ch3

3.4.5.5 DMA Ch4: for ASRC pair A output - moving audio from ASRC pair A output data register to memory

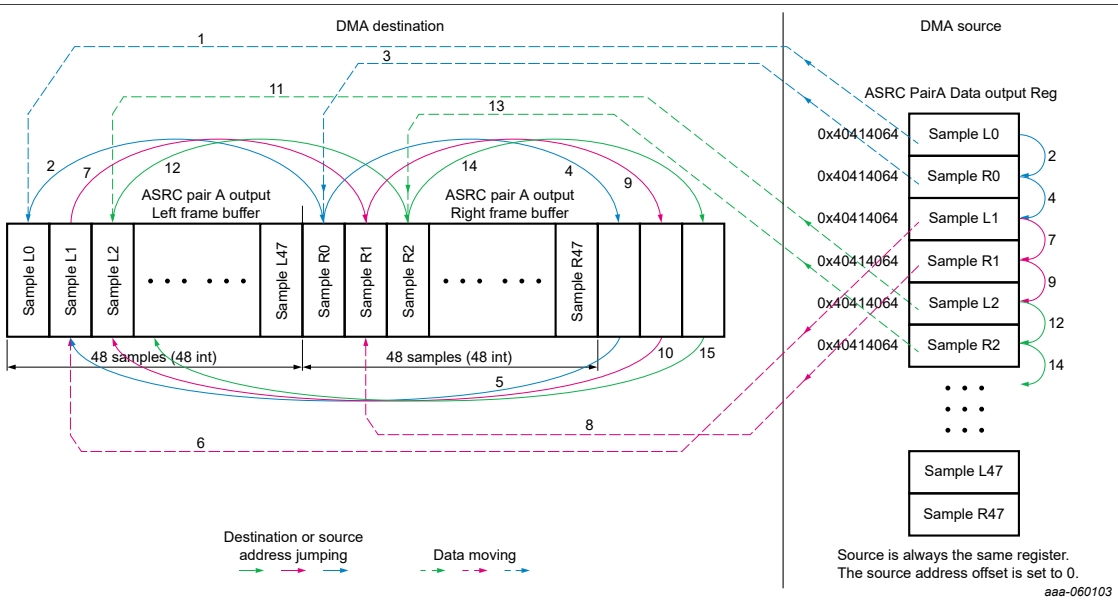


Figure 15. DMA Ch4

3.4.5.6 DMA Ch5: for ASRC pair B input - moving audio from SAI4 RX register to ASRC pair B input data register

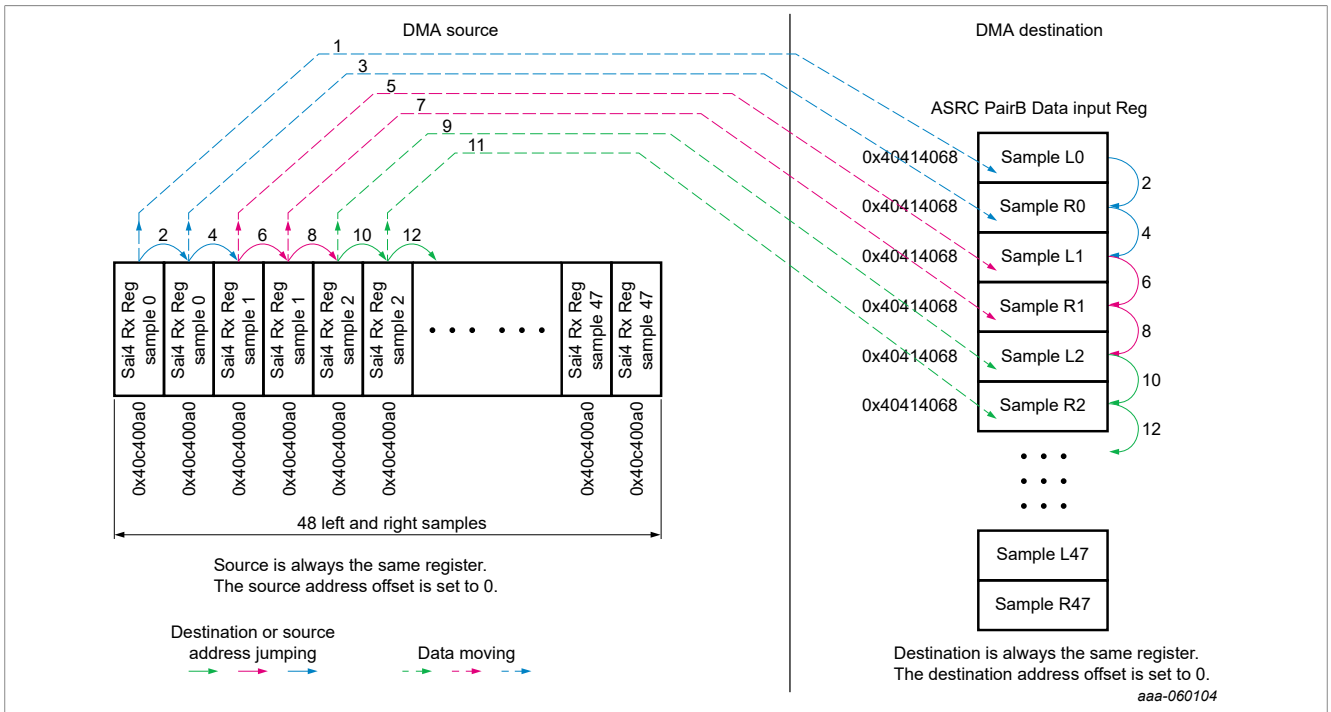


Figure 16. DMA Ch5

3.4.5.7 DMA Ch6: for ASRC pair B output - moving audio from ASRC pair B output data register to memory

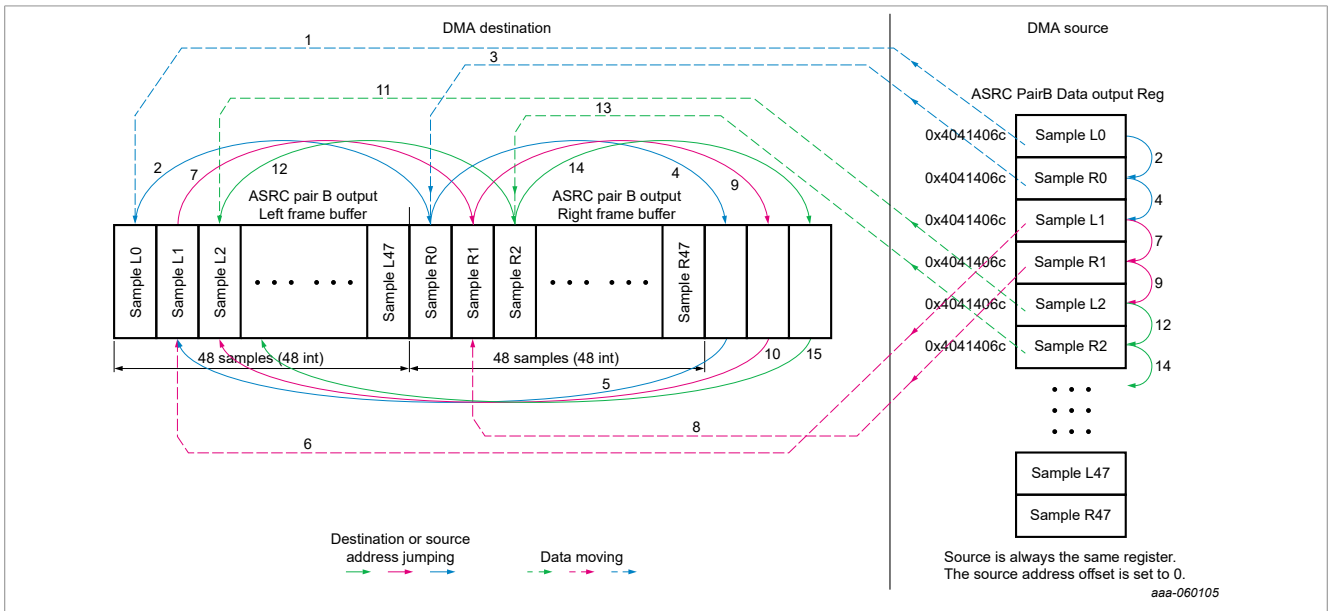


Figure 17. DMA Ch6

3.4.5.8 DMA Ch7: for ASRC pair C input - moving audio in memory to ASRC pair C input data register

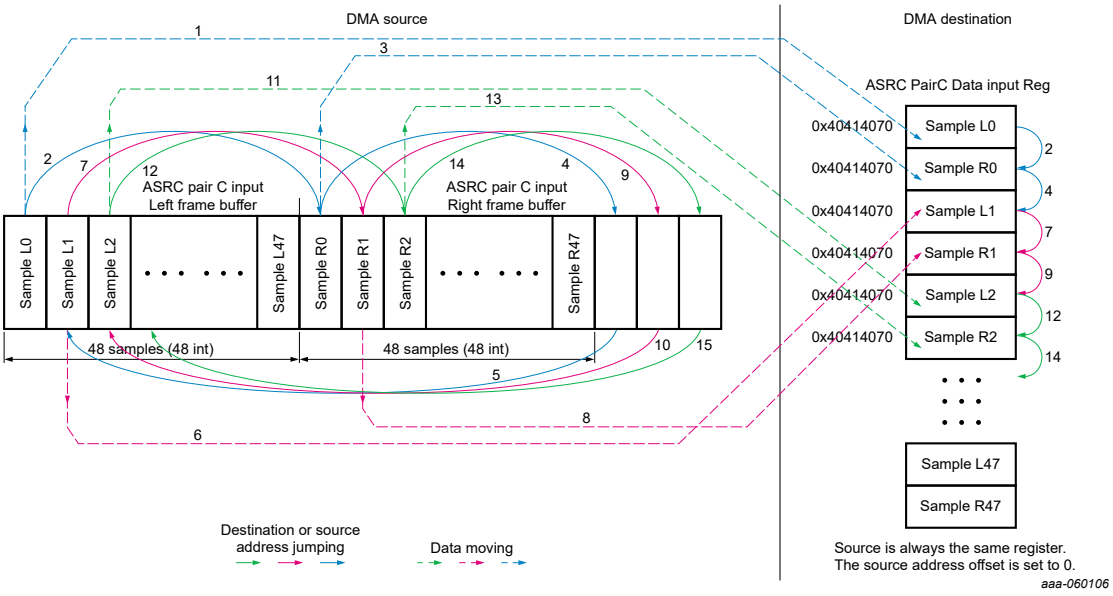


Figure 18. DMA Ch7

3.4.5.9 DMA Ch8: for ASRC pair C output - moving audio from ASRC pair C output data register to SAI4 TX register

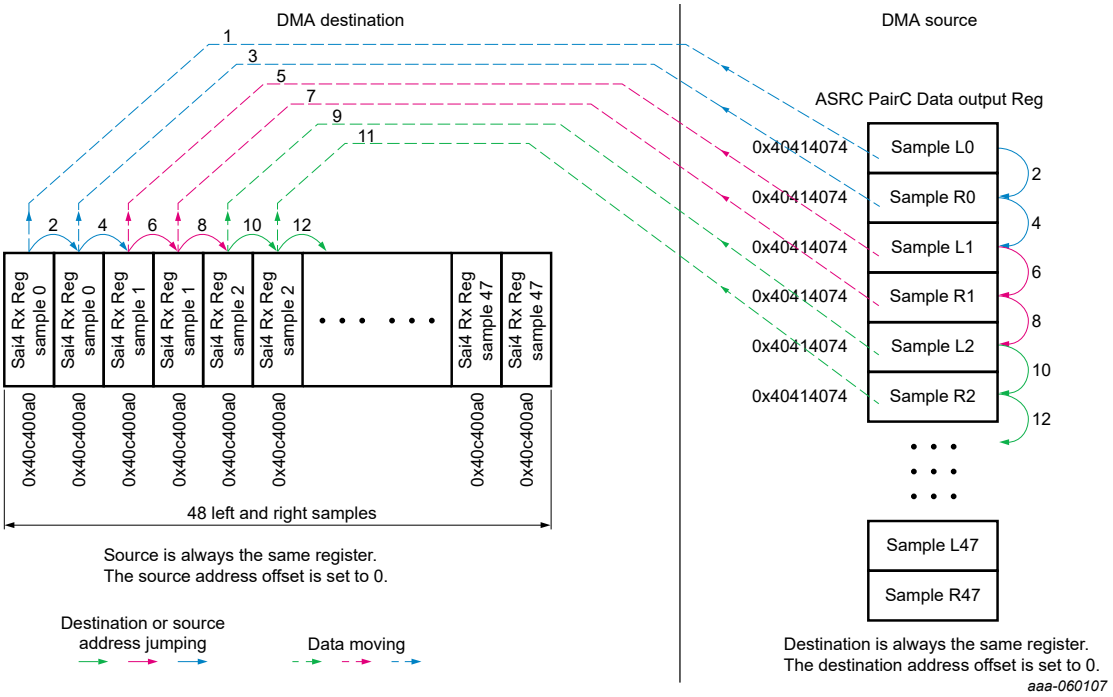


Figure 19. DMA Ch8

3.4.6 Signal flow

The testing project is built to test as many ASRC converting paths as possible. There are five internal sweeping signal generators, five audio RX sources, three audio TX sinks, and three switchers. Simple COM commands (single characters) are used to select the signal routing. The ASRC converted signal is streamed to all three sinks.

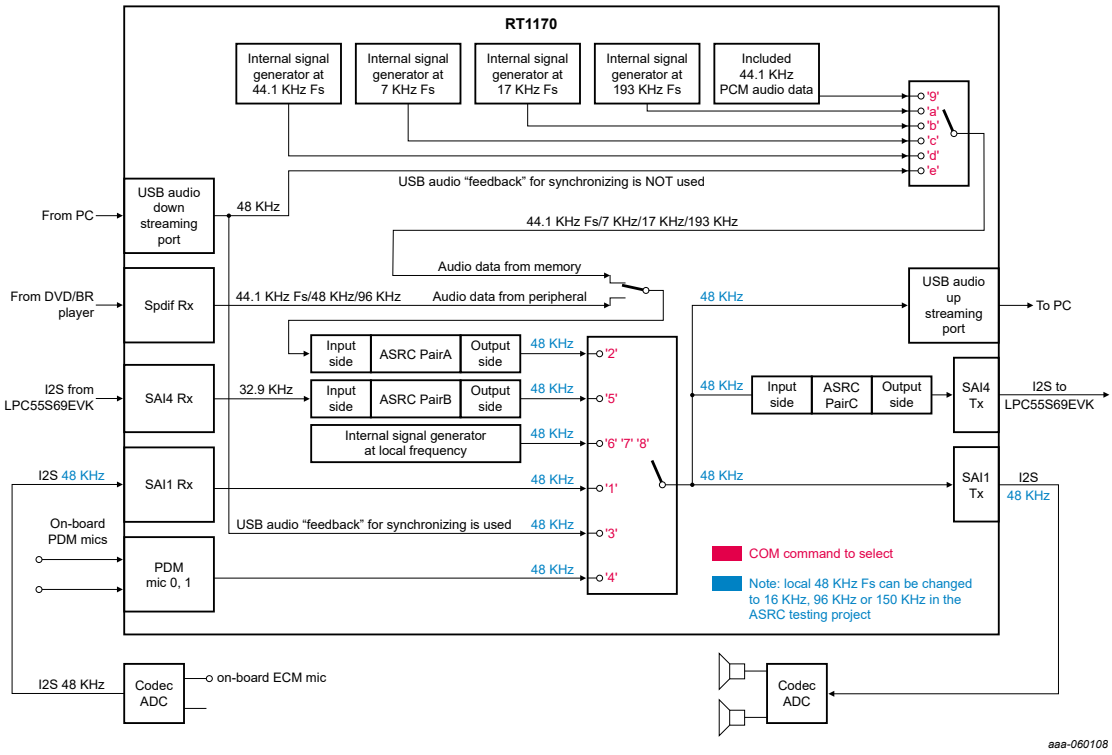


Figure 20. RT1170 internal signal flow

3.4.7 USB audio + COM interface

The testing project realized the USB audio and COM interface. You can connect the RT1170EVKB J20 USB OTG1 port and the PC with a USB cable. The RT1170 is recognized as a composite USB device of audio and COM. The USB audio up/down stream is both 32-bit, 48-kHz. The COM interface is used to send basic commands to the RT1170 and displaying some information printing from the RT1170.

3.5 System reliability - external audio source dropping down

When using the ASRC in real product development, consider the situation of the external signal dropping down and plugged back. The purpose is to avoid audio signal glitches from being received and played back.

In this testing project, we enable the SPDIF RX clock-locked flag interrupt to indicate that the SPDIF cable is plugged in and enable the SPDIF RX clock lost flag interrupt to indicate that the SPDIF cable is unplugged. The ASRC pair A is enabled and disabled according to the plug/unplug action detected. Remove J36 and put it back to test the SPDIF signal being lost and coming back.

For SAI4, we enable the RX FIFO Full (error) flag interrupt to indicate that the external I2S clock is attached and stable. We enable the SAI4 clock sync error flag interrupt to indicate that the external I2S clock is detached. The ASRC pair B and pair C are enabled and disabled according to the attach/detach action detected. Disconnect the LRCK or BICK from the LPC55S69EVK to test the external I2S dropping down and coming back.

We also enable the ASRC task overload error interrupt. If this error interrupt occurs, we reinitialize the ASRC.

With these interrupts enabled and handled, the testing project works very stable and allows for external audio sources to be freely attached or detached (both the SPDIF line and all the external I2S lines).

3.6 RT1170 ASRC MCUXpresso project macro definitions and COM commands

In the testing project, we have several compile macros to let the RT1170EVKB to work at different I2S sampling frequencies and change some other functions.

Set one of the following five macros to 1, to have the desired local I2S sampling frequency.

```
#define Sai1Fs_16KHz 0
#define Sai1Fs_48KHz 1
#define Sai1Fs_44p1KHz 0
#define Sai1Fs_96KHz 0
#define Sai1Fs_150KHz 0
```

Enable this macro to watch the USB audio buffer status. When this macro is enabled, you do not see the single-character COM command printings.

```
#define LetComFeedbackUsbUpDnStreamBufLevel
```

Close this macro to disable some of the DMA interrupts. Having these DMA interrupts enabled is to watch the timing sequence generated by the GPIO in the interrupts.

```
#define CloseUnnecessaryIrq
```

Enable this macro to evaluate the ASRC ideal ratio change in real time.

```
#define EnableAsrcPariAIdealRatioChange
```

Use the COM port of the enumerated composite USB device (not the J-Link COM port) to send single characters to switch the audio source selection:

- 0: mute
- 1: SAI1 Rx0
- 2: ASRC pair A - SPDIF In is used
- 3: USB down streaming
- 4: PDM microphone 0,1
- 5: ASRC pair B --- SAI4 Rx0
- 6: internal sweeping
- 7: internal fixed frequency
- 8: internal fixed frequency as right, and left is raw WAV
- 9: use included 44.1-kHz PCM audio data as ASRC pair A input - SPDIF is disconnected
- a: use 7-kHz Fs sweeping as ASRC pair A input - SPDIF is disconnected
- b: use 17-kHz Fs sweeping as ASRC pair A input - SPDIF is disconnected
- c: use 96-kHz Fs sweeping as ASRC pair A input - SPDIF is disconnected
- d: use 193-kHz Fs sweeping as ASRC pair A input - SPDIF is disconnected
- e: USB downstreaming, and use ASRC pair A for USB downstreaming audio synchronizing

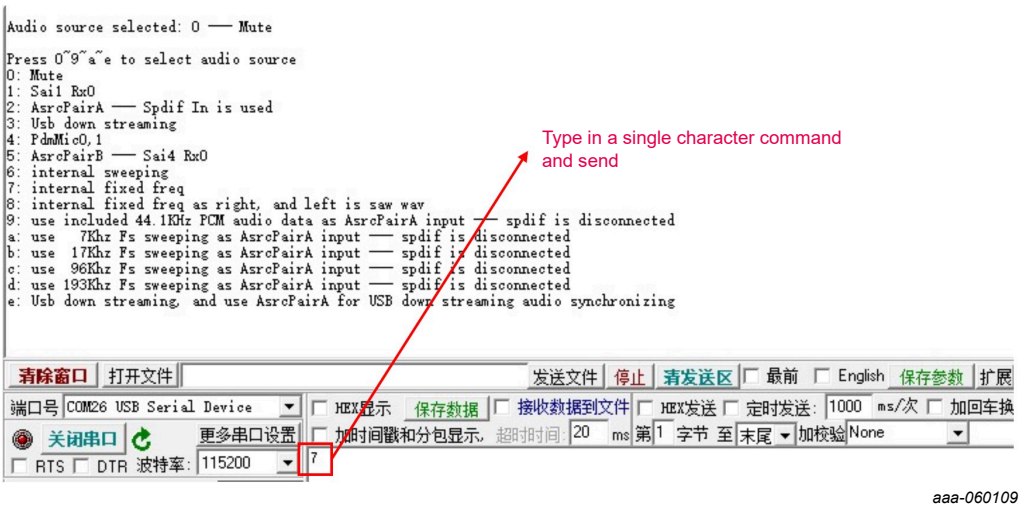


Figure 21. COM commands

3.7 RT1170 USB audio downstreaming in synchronized mode

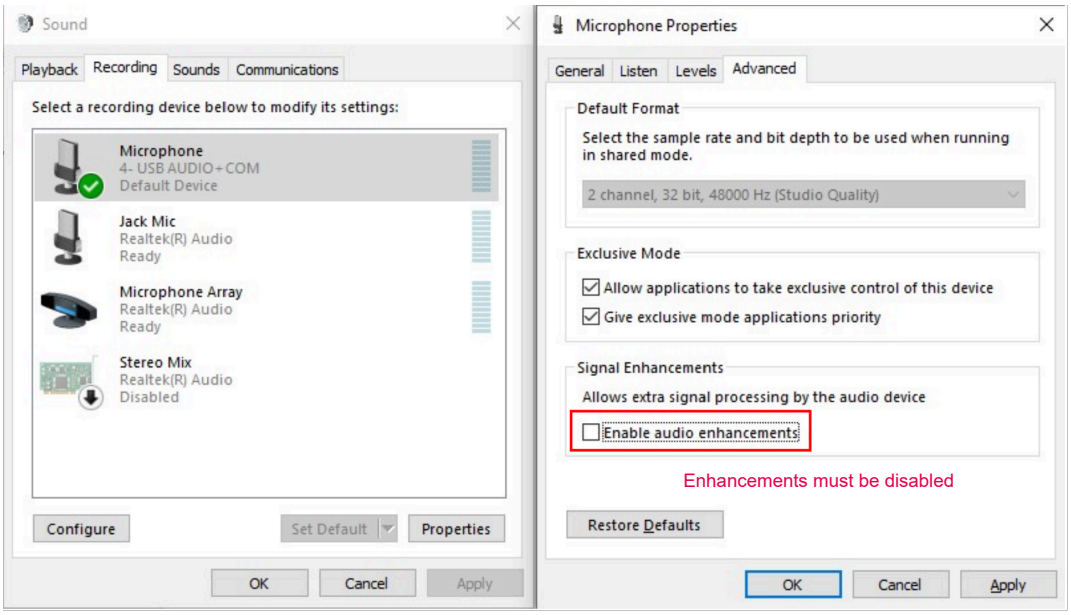
With the help of ASRC, we can implement the Synchronized mode for USB audio downstreaming on RT1170. In this testing project, you can send the "e" COM command to evaluate the USB audio downstreaming in the Synchronized mode. The details will be described in another application note.

4 Performance evaluation, ASRC converting quality

This document does not involve professional audio testing with a dedicated audio tester, such as AP. For professional audio quality testing to the ASRC, see *Vybrid ASRC Performance* (document EB808).

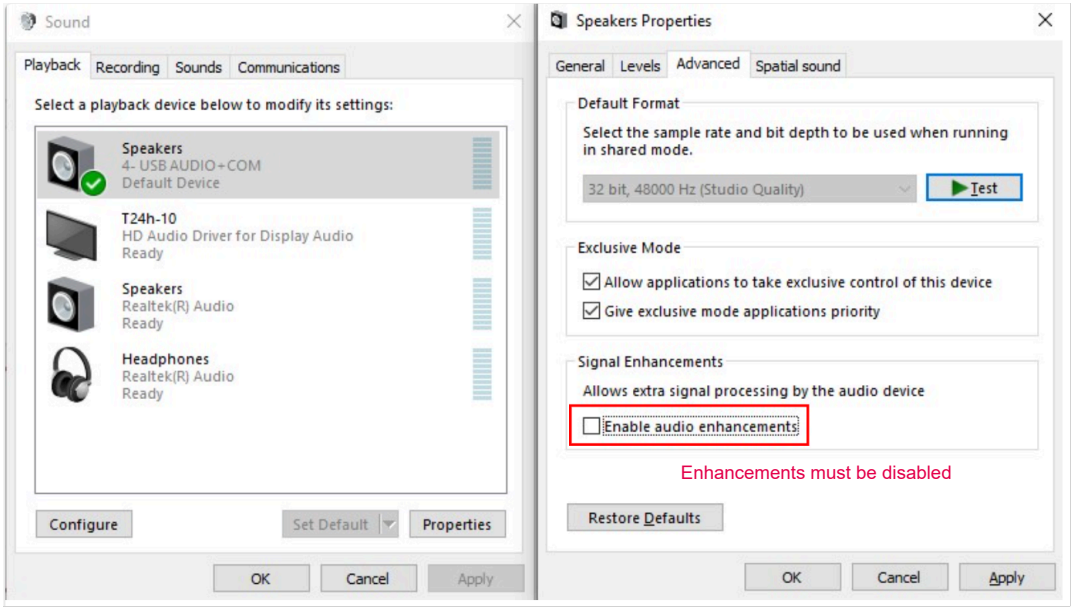
In the scope of this document, we simply use a chirp signal to sweep the ASRC converting. The output from the ASRC is recorded or captured. Then we analyze the spectrum of what we record or capture.

To record the ASRC output, we connect the RT1170EVKB J20 to the PC with a USB cable and ensure that the enhancement option for the recorder and the speaker are both disabled.



aaa-060110

Figure 22. USB audio recording device settings



aaa-060111

Figure 23. USB audio playback device settings

To capture the ASRC output, the I2S signals are the clock and data lines on the RT1170 SAI1 and SAI4. Use a logic analyzer to probe these lines. See the Section 3.3 for the pin information. Set up an I2S analyzer in the logic analyzer application, as shown in Figure 24.

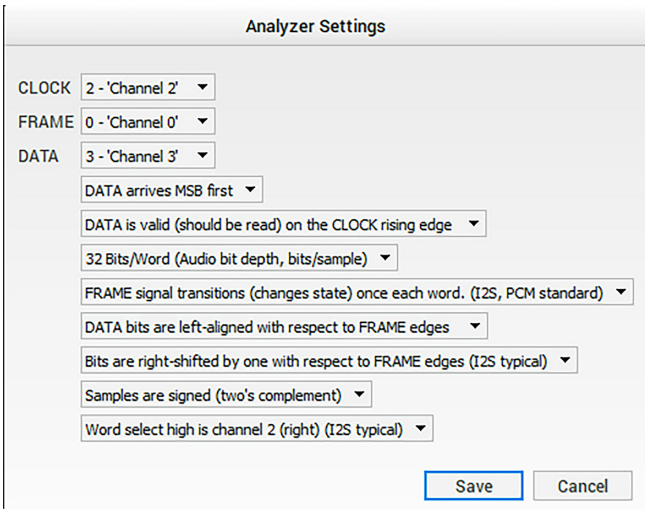


Figure 24. Logic analyzer settings for I2S

Set a proper sample rate for the capturing:

- 24 MHz for I2S Fs = 153 kHz / 96 kHz
- At least 12 MHz for I2S Fs = 48 kHz / 32.9 kHz
- At least 4 MHz for I2S Fs = 16 kHz

When the capturing is finished, dump the translated I2S samples. Because the logic analyzer is not a standard equipment, it is your responsibility to convert the dumped I2S samples to a raw audio file, either WAV (with info header) or PCM (without info header).

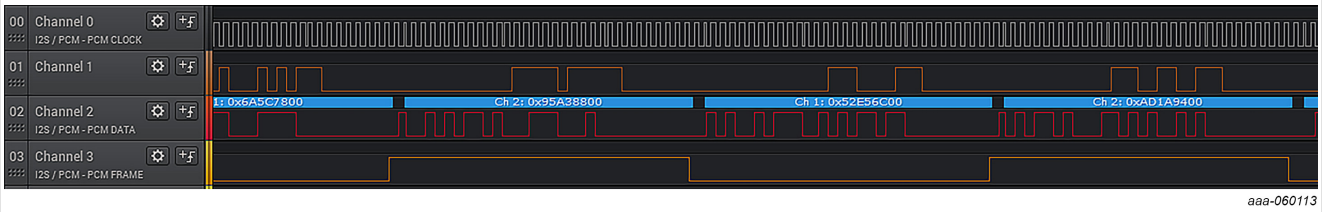


Figure 25. Interpreted I2S samples

4.1 Evaluation 1 - audio quality of the sweeping generator in LPC55S69 and RT1170

LPC55S69 is the external sweeping signal generator, the signal is captured on the RT1170 SAI4 RX. The noise spectrum is below -108 dB.

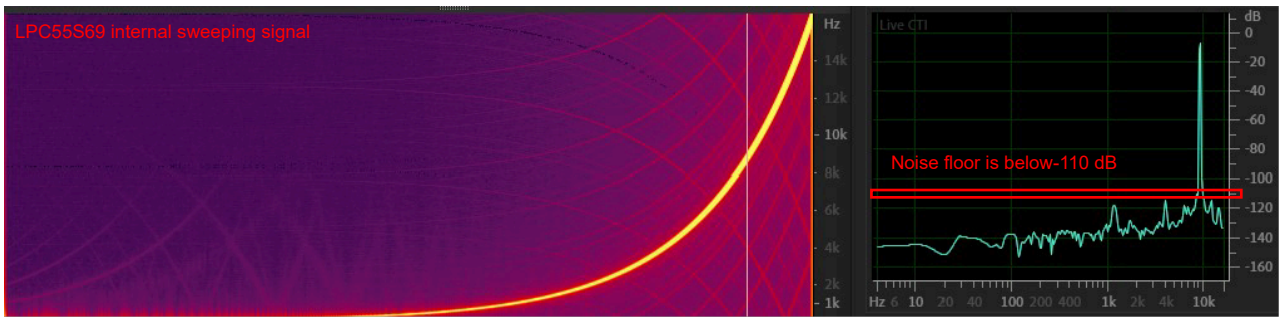


Figure 26. External LPC55S69 sweeping signal

RT1170 has an internal sweeping signal generator, the signal is captured on the RT1170 SAI1 TX. The noise spectrum is below -144 dB.

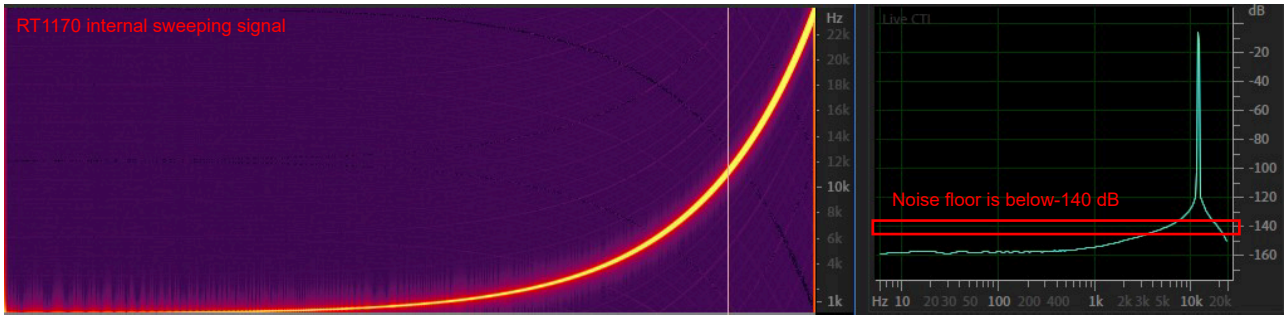


Figure 27. Internal RT1170 sweeping signal

4.2 Evaluation 2 - pair A quality: 44.1 kHz / 48 kHz SPDIF sweeping to RT170 SAI1 TX (ASRC is using measured ratio)

In this group of testing, use a single-character command “2” to activate the SPDIF going into the ASRC pair A as audio source, and set **#define** `Sai1Fs_16KHz` **#define** `Sai1Fs_48KHz` **#define** `Sai1Fs_96KHz` **#define** `Sai1Fs_150KHz` to 1, accordingly.

When converting the SPDIF sweeping to 48 kHz / 96 kHz / 150 kHz, all the effective spectrum remains in the original signal. When converting the SPDIF sweeping to 16 kHz, the spectrum over 8 kHz is removed from the original signal. The best quality is seen when converting the 48-kHz SPDIF to SAI1 local 48 kHz. There is spectrum aliasing/folding close to the Nyquist frequency of either the original signal or the converted signal. The major noise spectrum (excluding the frequency aliasing) is below -110 dB.

Note: *In the following six figures, the noise floor of the spectrum is around -120 dB (not as low as in the other figures). This is because the 44.1-kHz SPDIF source is from a CD disc played in the DVD player. This audio source has 16-bit resolution. The 48-kHz SPDIF source is from a WAV file in a USB drive and played by the DVD player. This audio source has 16-bit resolution too. In the other sweeping tests, the audio sources have 32-bit resolution and the noise floor of the spectrum is as low as -160 dB.*

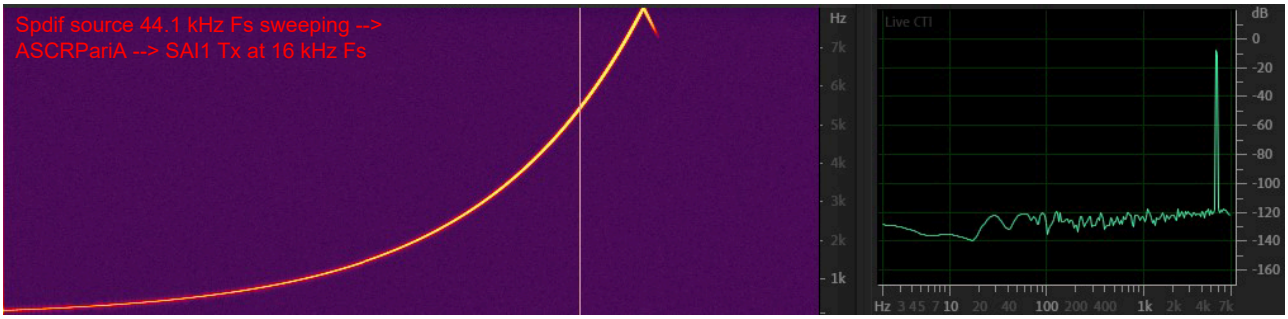


Figure 28. 44.1-kHz Fs SPDIF converted to 16-kHz Fs

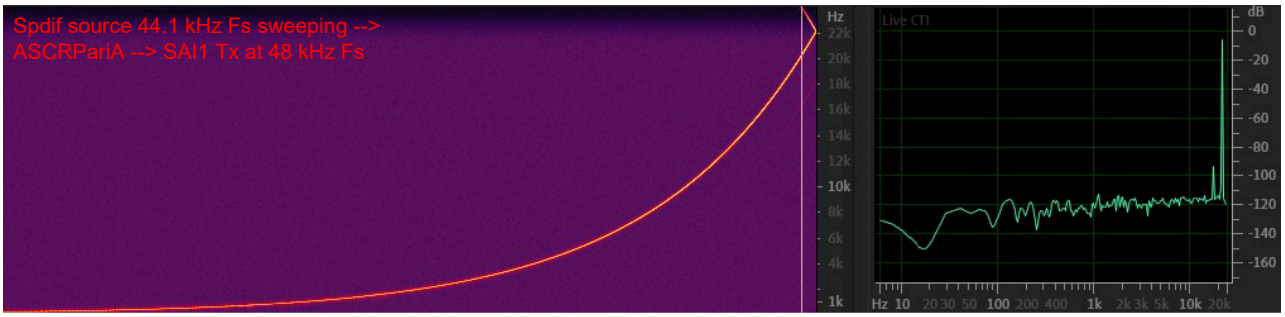


Figure 29. 44.1-kHz Fs SPDIF converted to 48-kHz Fs

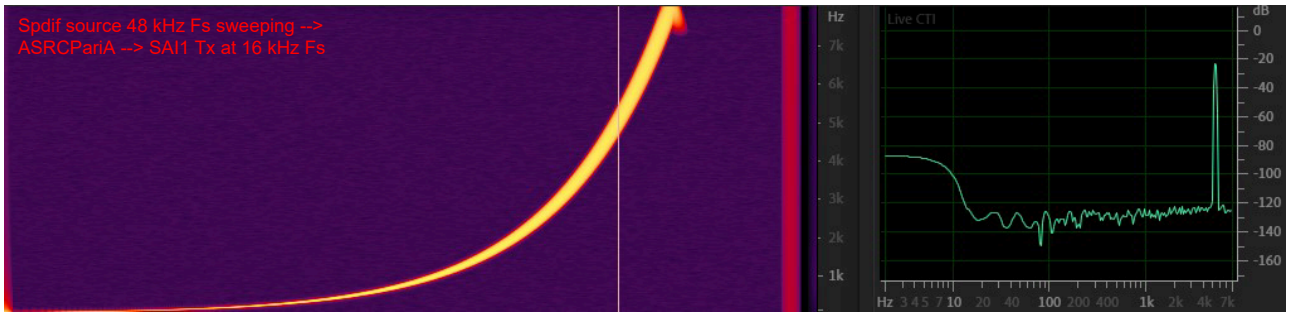


Figure 30. 48-kHz Fs SPDIF converted to 16-kHz Fs

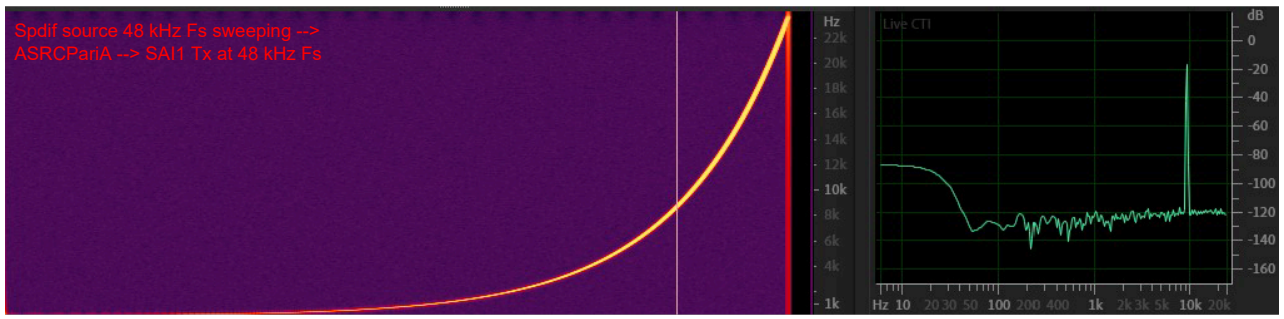


Figure 31. 48-kHz Fs SPDIF converted to 48-kHz Fs

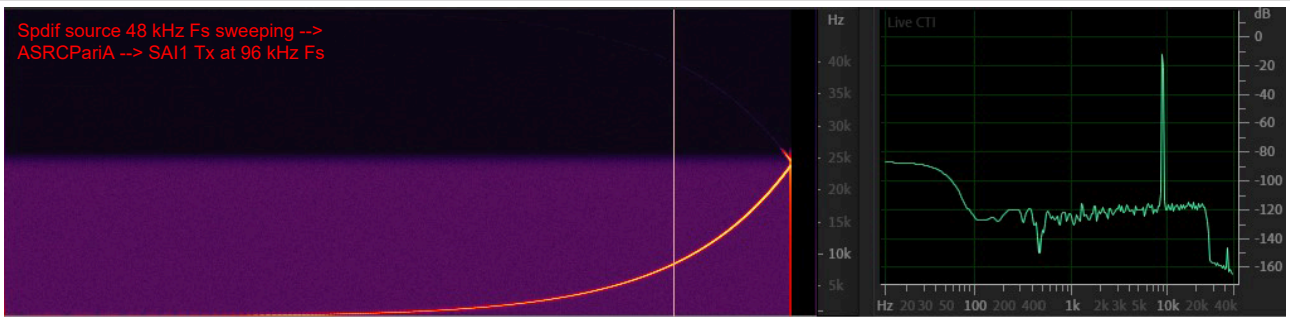


Figure 32. 48-kHz Fs SPDIF converted to 96-kHz Fs

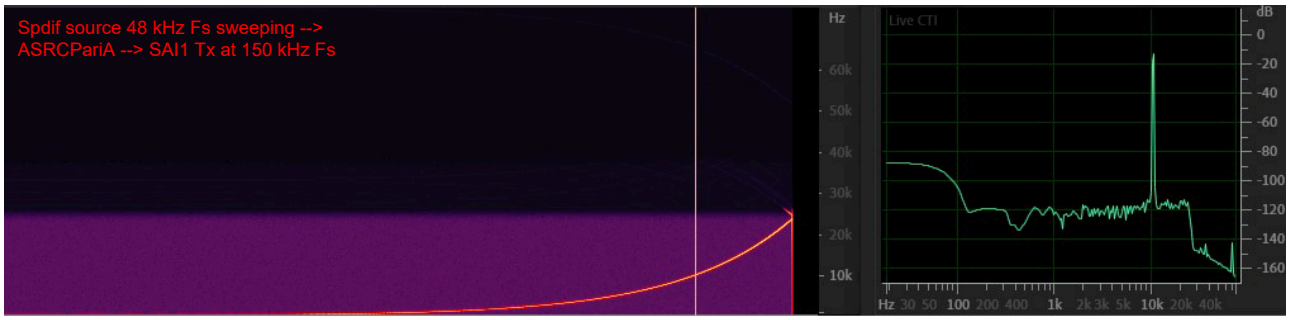


Figure 33. 48-kHz Fs SPDIF converted to 150-kHz Fs

4.3 Evaluation 3 - pair A quality: RT1170 internal sweeping to SAI1 TX 16 kHz Fs (ASRC is using specified ideal ratio)

In this group of testing, use a single-character command “a”, “b”, “c”, “d” to activate the needed audio source, and set **`#define`** `Sai1Fs_16KHz` to 1.

When the RT170 internal sweeping Fs is smaller than the output Fs, all effective spectrum remains in the original signal. Otherwise, the spectrum over the Nyquist frequency of the output side is removed. There is frequency aliasing / folding close to the Nyquist frequency. The major noise spectrum (excluding the frequency aliasing) is below -132 dB.

Note: *For the 193 kHz to 16 kHz conversion, there is the extra spectrum. This means that the RT1170 ASRC cannot convert 193 kHz to 16 kHz, because it is out of the ratio range specified by the RT1170 user manual. The supported ratio (FsIn / FsOut) range is between 1 / 24 to 8.*

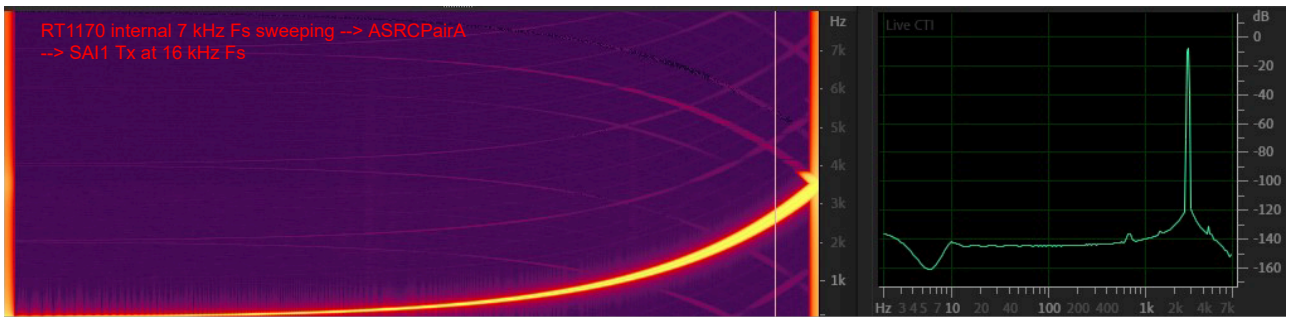


Figure 34. 7-kHz Fs sweeping converted to 16-kHz Fs

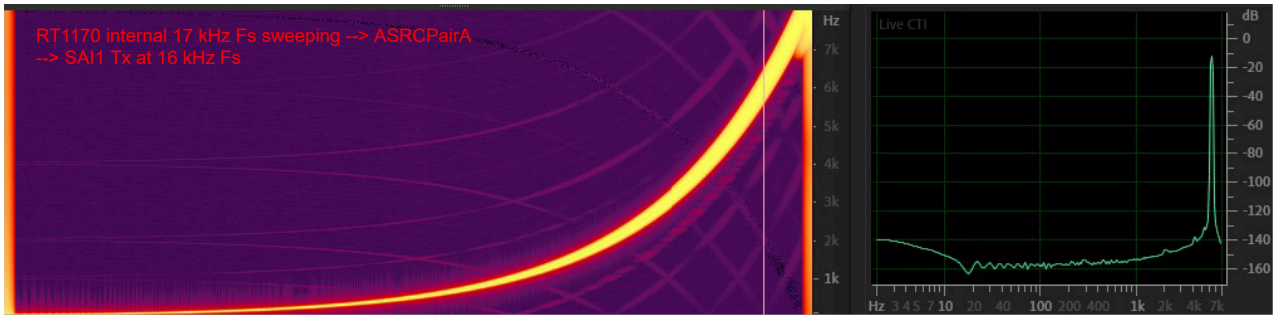


Figure 35. 17-kHz Fs sweeping converted to 16-kHz Fs

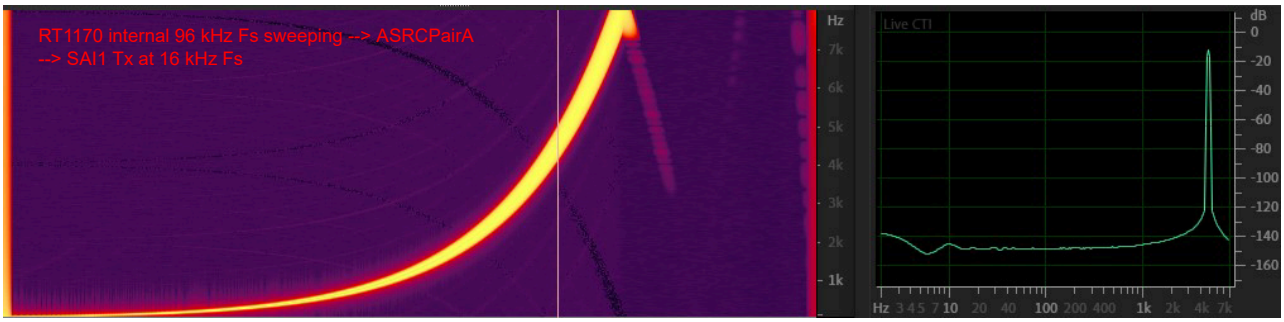


Figure 36. 96-kHz Fs sweeping converted to 16-kHz Fs

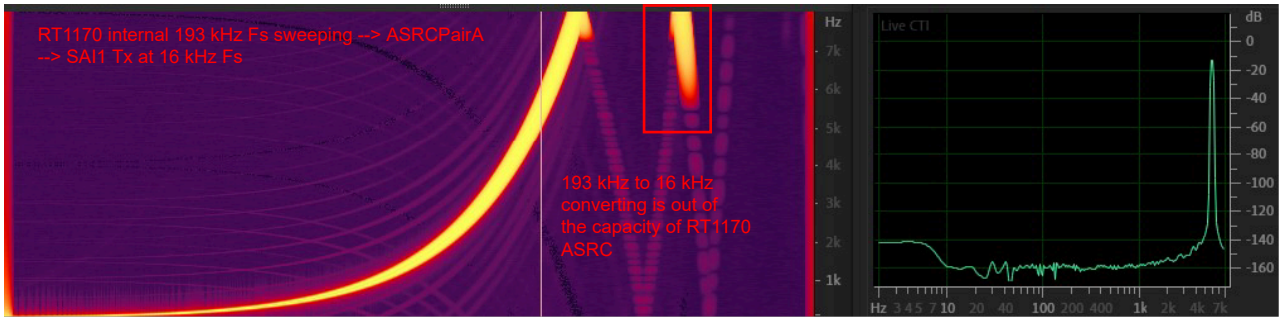


Figure 37. 193-kHz Fs sweeping converted to 16-kHz Fs

4.4 Evaluation 4 - pair A quality: RT1170 internal sweeping to SAI1 TX 48 kHz Fs (ASRC is using specified ideal ratio)

In this group of testing, use a single-character command “a”, “b”, “c”, “d” to activate the needed audio source, and set **`#define`** `Sai1Fs_48KHz` to 1.

When the RT170 internal sweeping Fs is smaller than the output Fs, all effective spectrum remains in the original signal. Otherwise, the spectrum over the Nyquist frequency of the output side is removed. There is frequency aliasing / folding close to the Nyquist frequency. The major noise spectrum (excluding the frequency aliasing) is below -120 dB.

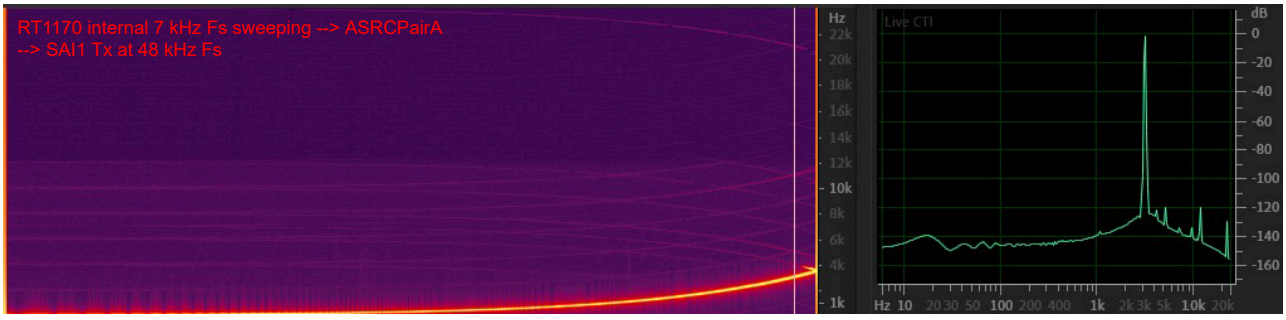


Figure 38. 7-kHz Fs sweeping converted to 48-kHz Fs

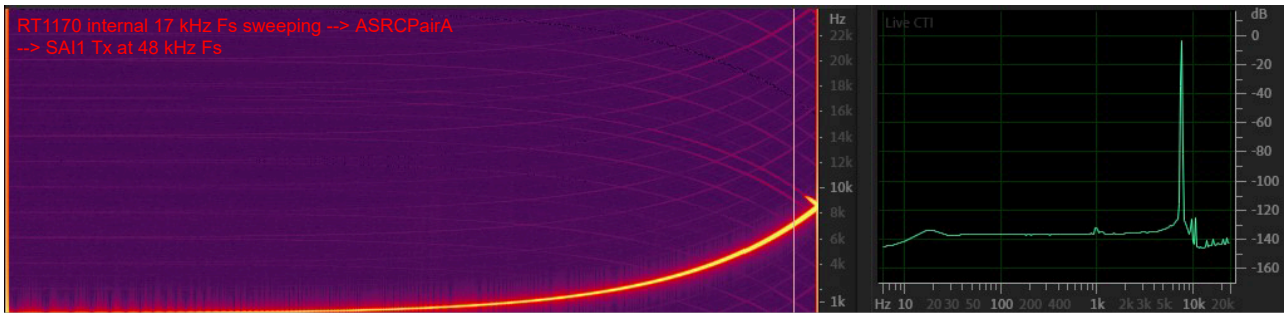


Figure 39. 17-kHz Fs sweeping converted to 48-kHz Fs

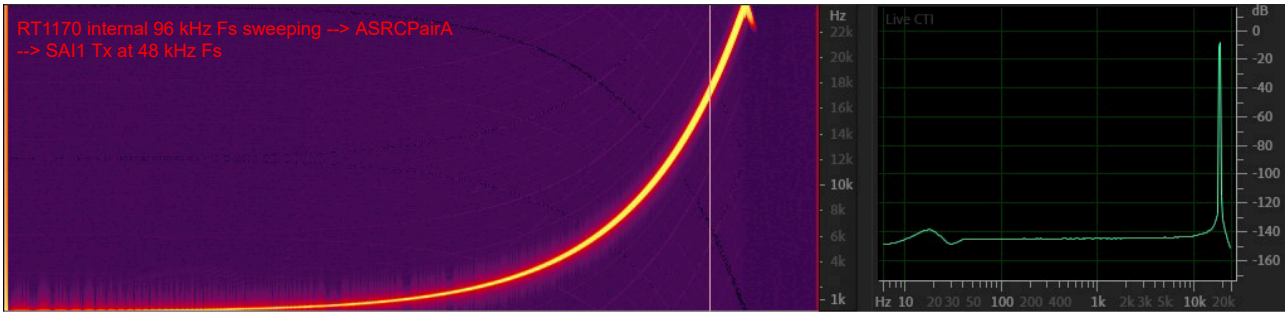


Figure 40. 96-kHz Fs sweeping converted to 48-kHz Fs

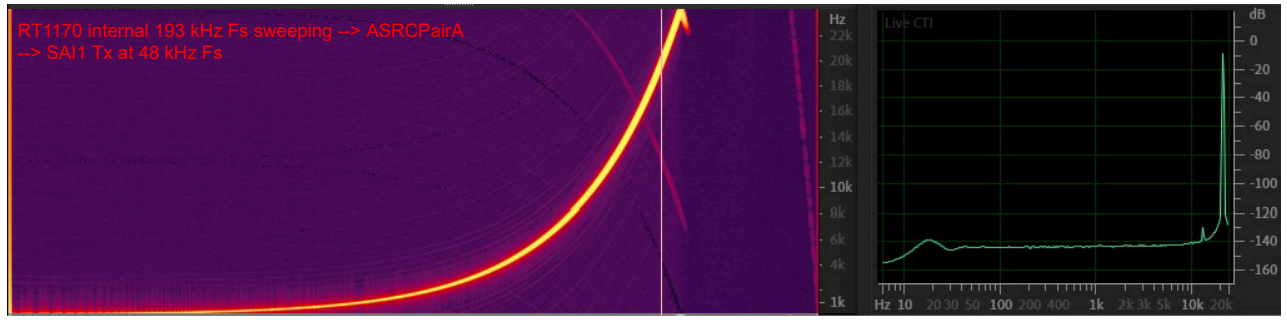


Figure 41. 193-kHz Fs sweeping converted to 48-kHz Fs

4.5 Evaluation 5 - pair A quality: RT1170 internal sweeping to SAI1 TX 150 kHz Fs (ASRC is using specified ideal ratio)

In this group of testing, use a single-character command “a”, “b”, “c”, “d” to activate the needed audio source, and set **`#define`** `Sai1Fs_150KHz` to 1.

When the RT170 SAI1 is running at a 150-kHz Fs, the spectrum up to 75 kHz is available. However, the RT1170 ASRC removes the input signal spectrum above 24 kHz. Even the RT1170 internal sweeping contains spectrum higher than 24 kHz. When the sweeping Fs is 96 kHz or 193 kHz, the output signal still has empty spectrum above 24 kHz (except noise). There is frequency aliasing / folding close to the Nyquist frequency. The major noise spectrum (excluding the frequency aliasing) is below -120 dB.

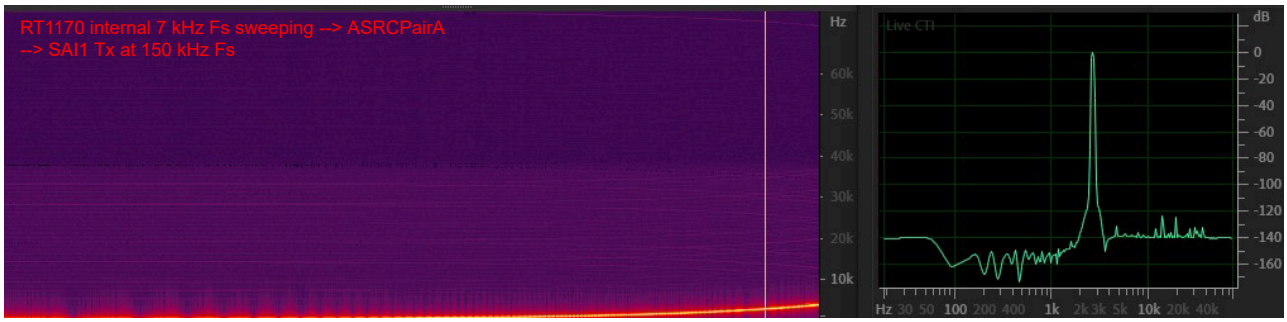


Figure 42. 7-kHz Fs sweeping converted to 150-kHz Fs

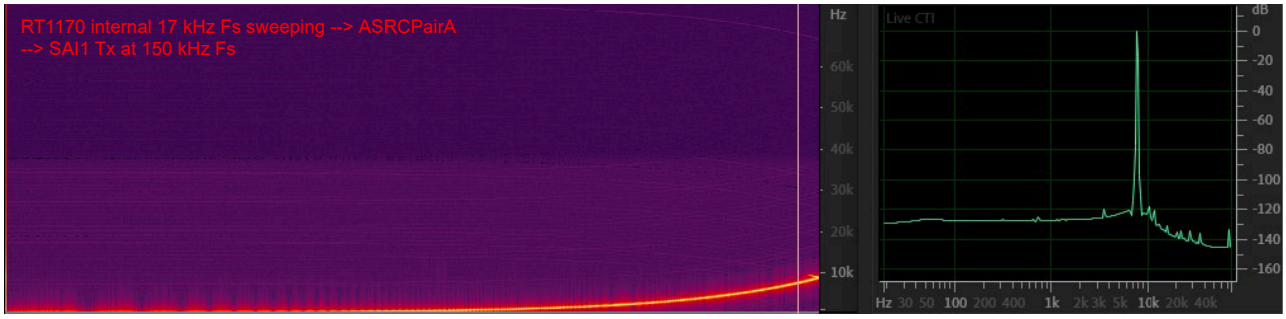


Figure 43. 17-kHz Fs sweeping converted to 150-kHz Fs

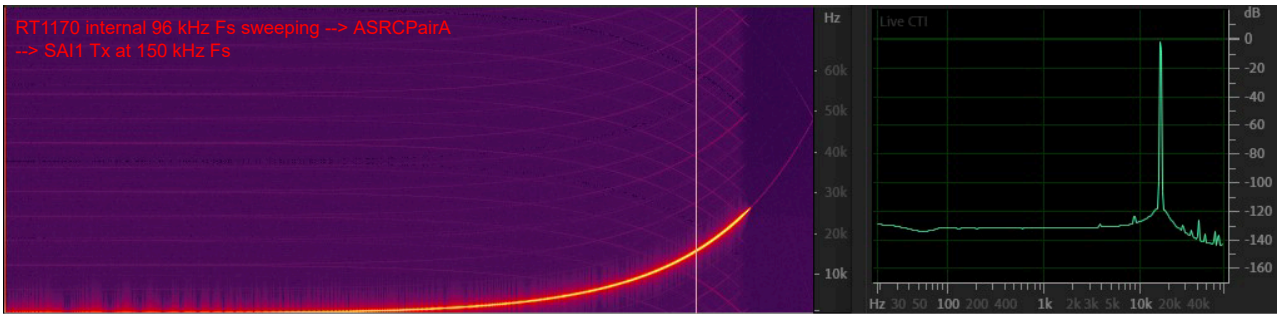


Figure 44. 96-kHz Fs sweeping converted to 150-kHz Fs

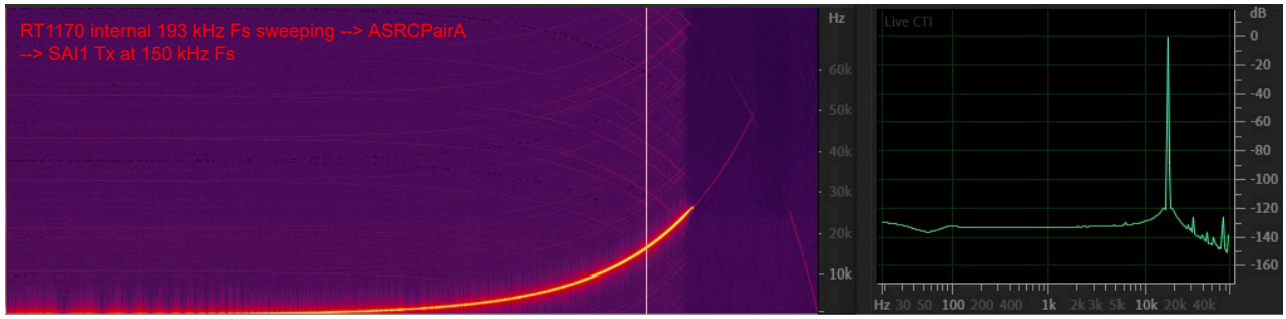


Figure 45. 193-kHz Fs sweeping converted to 150-kHz Fs

4.6 Evaluation 6 - pair B quality: LPC55S69 sweeping at Fs 33 kHz, from RT1170 SAI4 RX to RT1170 SAI1 I2S TX at 48 kHz (ASRC is using detected ratio)

In this testing, use a single-character command “5” to activate the SAI4 Rx going into the ASRC pair B as the audio source, and set **`#define`** `Sai1Fs_48KHz` to 1.

In this case, the whole spectrum in the input signal remains. There is frequency aliasing / folding close to the Nyquist frequency. The major noise spectrum (excluding the frequency aliasing) is below -108 dB. As shown in Figure 46, the original sweeping signal from LPC55S69 does not have as good quality as the ideal sweeping signal. That is why we have the ASRC output noise spectrum close to -108 dB.

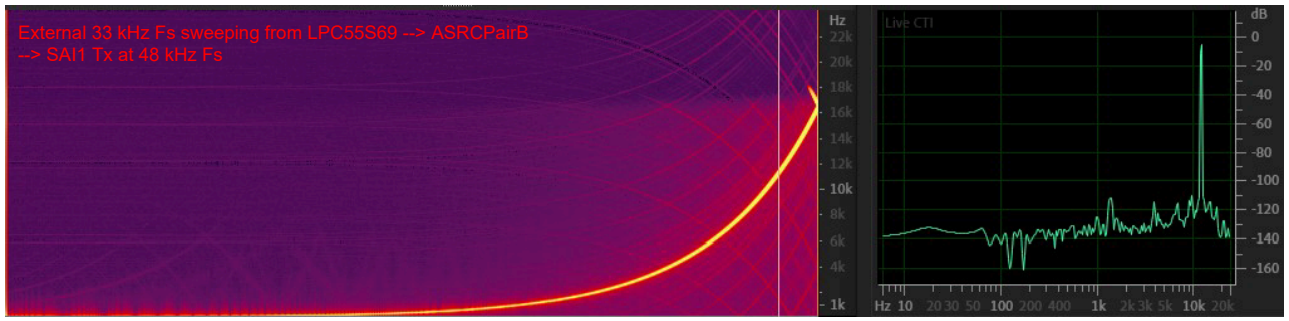


Figure 46. 33-kHz Fs sweeping converted to 48-kHz Fs

4.7 Evaluation 7 - pair C quality - RT1170 internal sweeping at 44.1 kHz to SAI4 TX (to LPC55S69), Fs of 24 kHz, 25 kHz, 32967 Hz, 47244 Hz, 48 kHz, 96774 Hz, 150 kHz (ASRC is using detected ratio)

In this group of testing, use a single-character command “6” to activate the internal tone-sweeping generator as the audio source, and set **`#define`** `Sai1Fs_44p1KHz` to 1. In the LPC55S69 project, set **`#define`** `SetFsTo_24KHz`, **`#define`** `SetFsTo_25KHz`, **`#define`** `SetFsTo_33KHz`, **`#define`** `SetFsTo_47KHz`, **`#define`** `SetFsTo_48KHz`, **`#define`** `SetFsTo_97KHz`, **`#define`** `SetFsTo_150KHz` to 1, accordingly.

This section, Section 4.8, and Section 4.9 show the following. When the RT170 internal sweeping Fs is equal to or smaller than 48-kHz Fs and the output Fs is equal to or greater than 48 kHz, all effective spectrum in the original signal remains. Otherwise, the spectrum over the Nyquist frequency of the output side is removed or the spectrum above 24 kHz in the input signal is removed. There is frequency aliasing/folding close to the Nyquist frequency. The major noise spectrum (excluding the frequency aliasing) is below -120 dB.

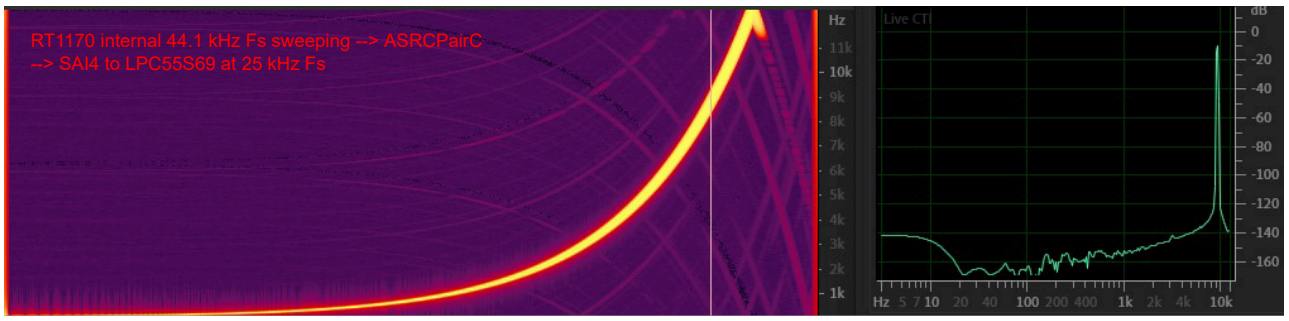


Figure 47. 44.1-kHz Fs sweeping converted to 25-kHz Fs

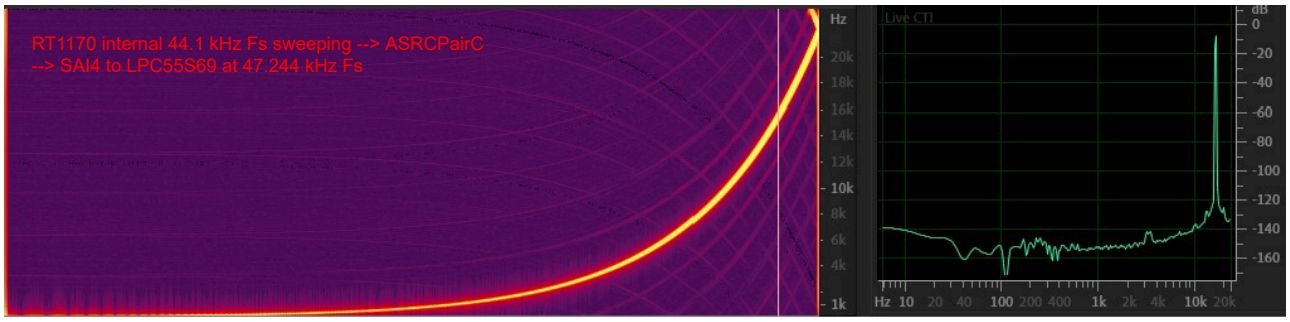


Figure 48. 44.1-kHz Fs sweeping converted to 47.244-kHz Fs

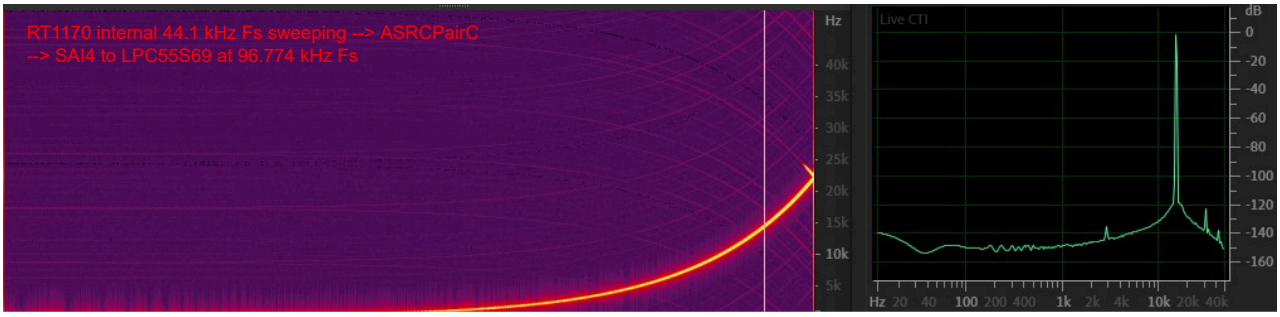


Figure 49. 44.1-kHz Fs sweeping converted to 96.774-kHz Fs

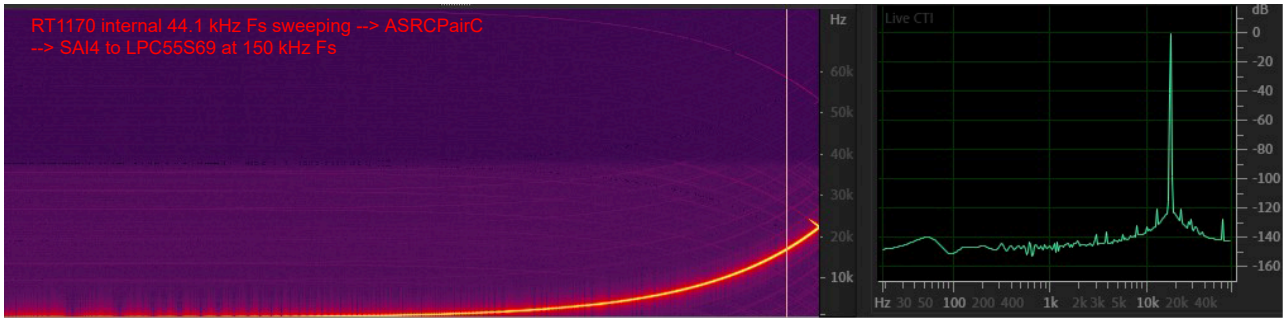


Figure 50. 44.1-kHz Fs sweeping converted to 150-kHz Fs

4.8 Evaluation 8 - pair C quality - RT1170 internal sweeping at 96 kHz to SAI4 TX (to LPC55S69), Fs of 24 kHz, 25 kHz, 32967 Hz, 47244 Hz, 48 kHz, 96774 Hz, 150 kHz (ASRC is using detected ratio)

In this group of testing, use a single-character command “6” to activate the internal tone-sweeping generator as audio source, and set **`#define`** `Sai1Fs_96KHz` to 1. In the LPC55S69 project, set **`#define`** `SetFsTo_24KHz`, **`#define`** `SetFsTo_25KHz`, **`#define`** `SetFsTo_33KHz`, **`#define`** `SetFsTo_47KHz`, **`#define`** `SetFsTo_48KHz`, **`#define`** `SetFsTo_97KHz`, **`#define`** `SetFsTo_150KHz` to 1, accordingly.

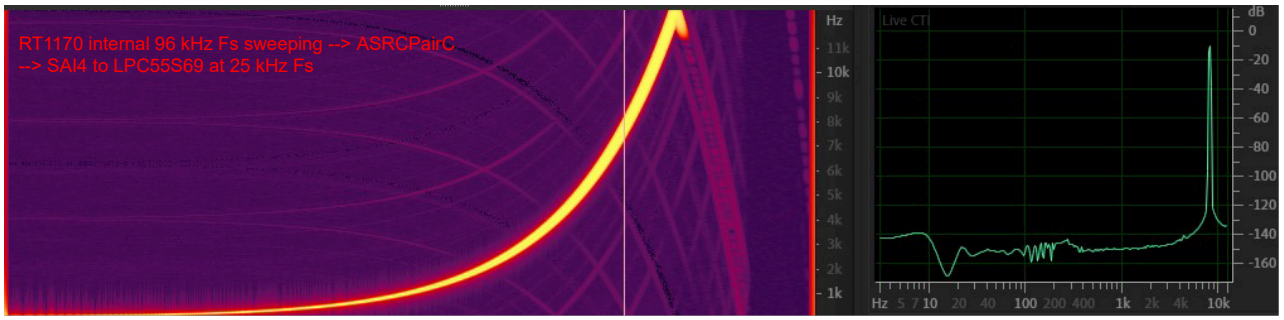


Figure 51. 96-kHz Fs sweeping converted to 25-kHz Fs

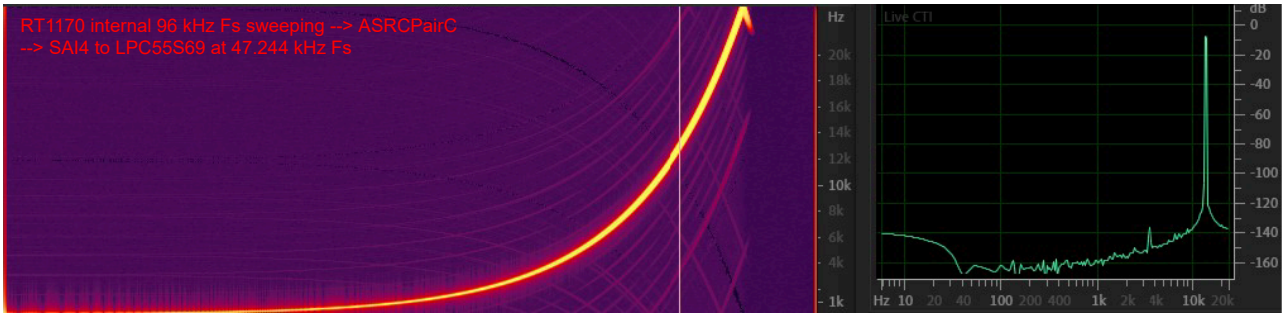


Figure 52. 96-kHz Fs sweeping converted to 47.244-kHz Fs

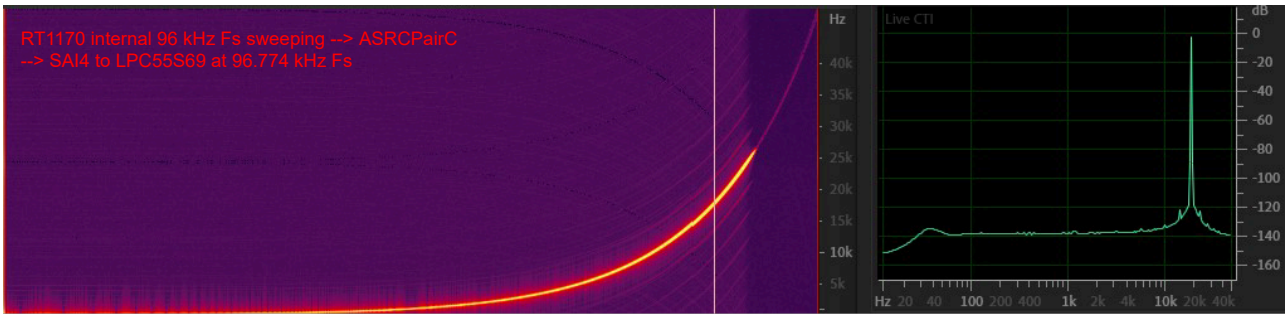


Figure 53. 96-kHz Fs sweeping converted to 96.774-kHz Fs

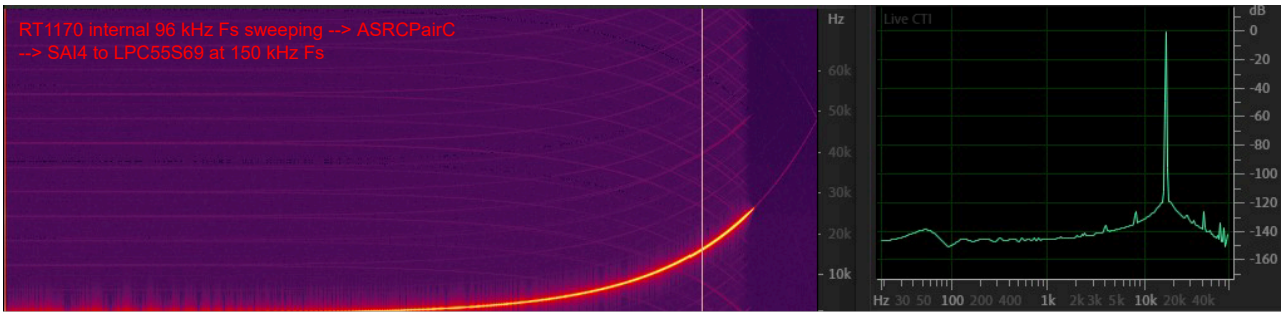


Figure 54. 96-kHz Fs sweeping converted to 150-kHz Fs

4.9 Evaluation 9 - pair C quality - RT1170 internal sweeping at 193 kHz to SAI4 TX (to LPC55S69), Fs of 24 kHz, 25 kHz, 32967 Hz, 47244 Hz, 48 kHz, 96774 Hz, 150 kHz (ASRC is using detected ratio)

In this group of testing, use a single-character command “6” to activate the internal tone-sweeping generator as the audio source, and set `#define Sai1Fs_150KHz` to 1. In the LPC55S69 project,

set **#define** SetFsTo_24KHz, **#define** SetFsTo_25KHz, **#define** SetFsTo_33KHz, **#define** SetFsTo_47KHz, **#define** SetFsTo_48KHz, **#define** SetFsTo_97KHz, **#define** SetFsTo_150KHz to 1, accordingly.

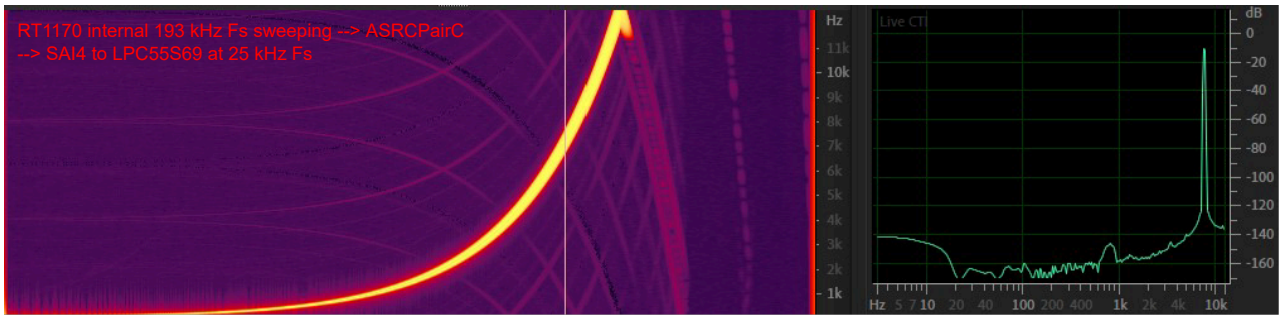


Figure 55. 193-kHz Fs sweeping converted to 25-kHz Fs

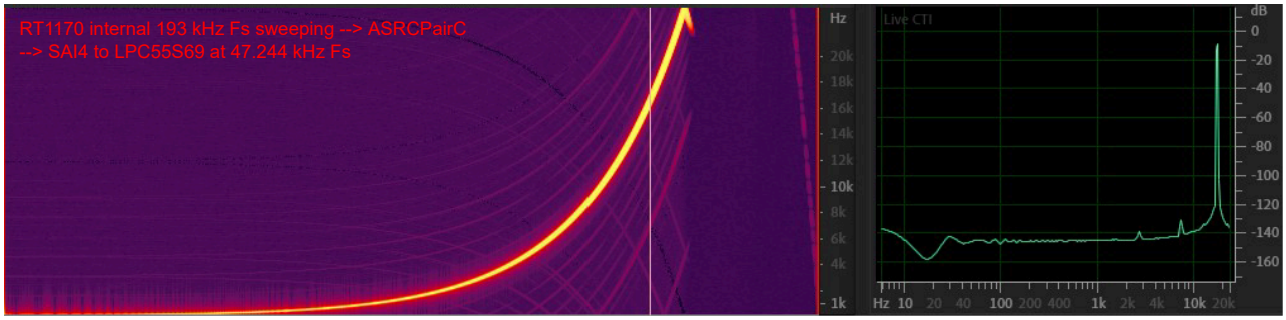


Figure 56. 193-kHz Fs sweeping converted to 47.244-kHz Fs

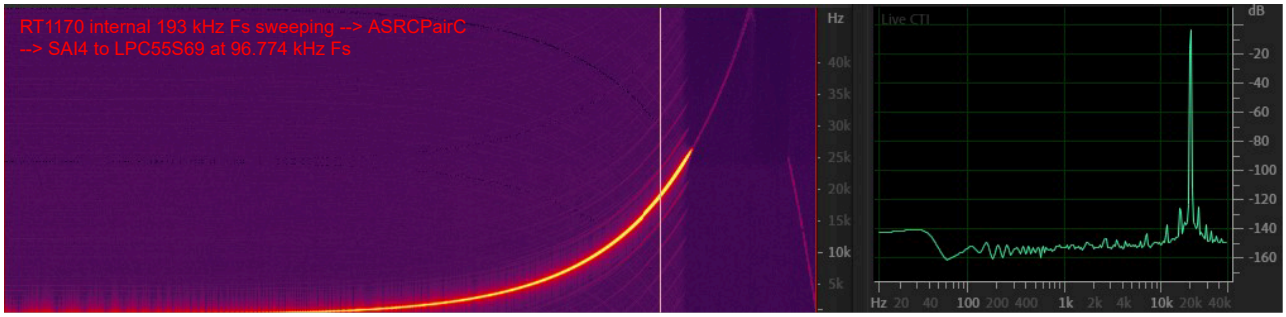


Figure 57. 193-kHz Fs sweeping converted to 96.774-kHz Fs

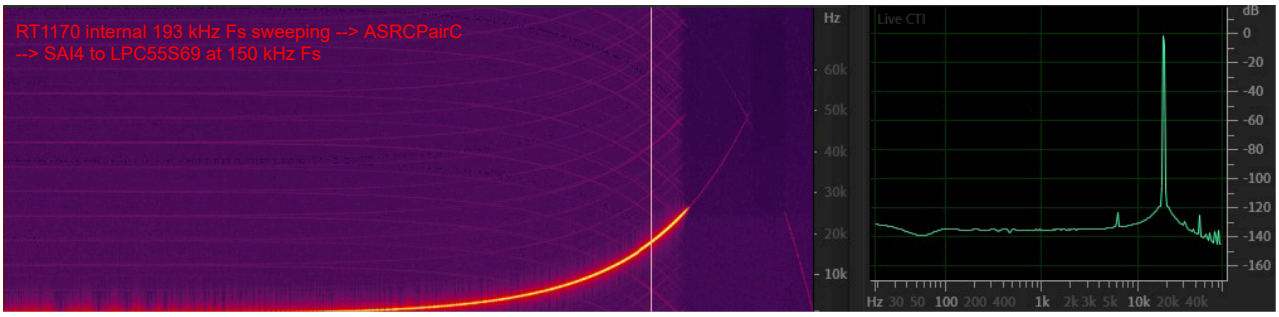


Figure 58. 193-kHz Fs sweeping converted to 150-kHz Fs

4.10 Evaluation 10 - ASRC converting latency

In this group of testing, use a single-character command “a”, “b”, “c”, or “d” to activate the internal tone-sweeping generator as the audio source, and set `#define Sai1Fs_48KHz` to 1 and `#define EnableAsrcPairALatencyTest` to 1.

To evaluate the converting latency, we make a sudden amplitude change every 1 second (or similar) in the right channel in the internal tone-sweeping generator. When the internal tone-sweeping generator decides to change the amplitude, the USB up-streaming left channel amplitude is decreases immediately.

After you type the “c” command, the 96-kHz Fs internally generated signal goes to ASRC pair A and is converted to the local 48-kHz Fs. Then it is up-streamed to the USB audio host. In this scenario, the USB audio up-streaming left channel reflects the signal amplitude change immediately, while the right channel amplitude change is delayed by the ASRC converting.

Observing the recorded USB up-streaming audio, we can exactly measure the latency in samples. Figure 59 shows the latency of 154 samples when you select the 96-kHz Fs internal sweeping.

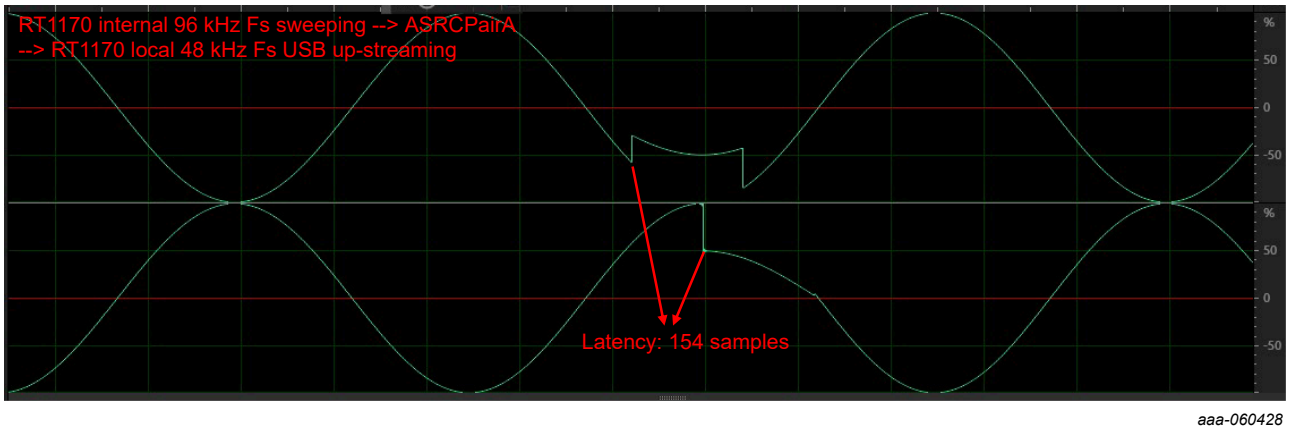


Figure 59. 96-kHz Fs internal sweeping converted to 48-kHz Fs USB up-streamed and with the sudden amplitude change for latency testing

In Figure 59, the latency of 154 samples equals to 154 / 48 = 3.2 ms. It consists of three parts:

- The latency caused by ASRC Pair A input DMA dual buffer is 48 samples of 96-kHz Fs, equal to 0.5 ms.
- The ASRC Pair A converting latency.
- The latency caused by ASRC Pair A output DMA dual buffer is 48 samples of 48 kHz, equal to 1 ms.

The latency caused by ASRC Pair A is 3.2 - 0.5 - 1.0 = 1.7 ms, when the input Fs is 96 kHz and the output Fs is 48 kHz.

Using the same method, we know:

- The ASRC Pair A latency is 15.1 ms when the input Fs is 7 kHz and the output Fs is 48 kHz.
- The ASRC Pair A latency is 4.9 ms when the input Fs is 17 kHz and the output Fs is 48 kHz.
- The ASRC Pair A latency is 1.7 ms when the input Fs is 96 kHz and the output Fs is 48 kHz.
- The ASRC Pair A latency is 2.3 ms when the input Fs is 193 kHz and the output Fs is 48 kHz.

5 Conclusion

When the audio source and the audio sink are asynchronized to each other, use ASRC to match the data rates to avoid audible noise or glitches.

The RT1170 hardware ASRC with proper settings and working together with the DMA provides high-quality signal converting that satisfies consumable and professional audio requirements.

The detailed RT1170 ASRC performance in terms of the pass-band ripple, dynamic distortion, and THD+N is in *Vybrid ASRC Performance* (document EB808).

The following reference testing projects are available for further code reading and analyzing:

- Link to the Rt1170 testing prj
- Link to the LPC55S69 testing prj

6 Note about the source code in the document

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

Copyright 2025 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

7 Revision history

Table 1. Revision history

Document ID	Release date	Description
AN14631 v.1.1	6 May 2025	• Fixed a small typo
AN14631 v.1.0	29 April 2025	• Initial version

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

J-Link — is a trademark of SEGGER Microcontroller GmbH.

Contents