AN14674

How to Connect Asynchronized Audio Source and Sink with RT685 HiFi4 Cadence ASRC

Rev. 1.0 — 13 June 2025 Application note

Document information

Information	Content
Keywords	AN14674, Cadence HiFi Sample Rate Converter, Cadence ASRC, i.MX RT685 platform
Abstract	In this document, we introduce the Cadence HiFi Sample Rate Converter with the Cadence ASRC and demonstrate how it works on the NXP i.MX RT685 platform.

1 Introduction

How to Connect Asynchronized Audio Source and Sink with RT1170 ASRC (document AN14631) describes a solution for matching an asynchronized audio source and sink by the hardware ASRC block of i.MX RT1170. To handle the same issue on the processors that do not have a hardware ASRC, we must use either a software ASRC or an audio PLL real-time adjusting.

In this document, we introduce the Cadence HiFi Sample Rate Converter with the Cadence ASRC and demonstrate how it works on the NXP i.MX RT685 platform. The details of the audio PLL adjustment to synchronize the audio is not in the scope of this document.

For basic concept and ideas of synchronized or asynchronized audio source and sink, see *How to Connect Asynchronized Audio Source and Sink with RT1170 ASRC* (document AN14631).

2 Cadence HiFi ASRC overview

The HiFi Sample Rate Converter (Cadence ASRC) is designed for high-quality sample rate conversions for audio and speech applications. The HiFi SRC supports the input and output sampled at all the standard audio sample rates from 8 kHz to 192 kHz. It performs the sample-rate conversion on the input signals using one or more sets of linear-phase FIR filters and does not introduce any phase distortion during this conversion. The SRC filters used for integer conversion ratios have 100-dB (or more) stop band attenuation. While the SRC filters used for non-integer conversion ratios have about 80 dB stop band attenuation. The examples of integer conversion ratios are 1/3, 2/3, 3/1, 3/2, 1/4, and so on. The examples of non-integer conversion ratios are 32/44.1, 44.1/48, 48/88.2, and so on.

The Cadence ASRC provides the option to enable the ASRC and the option to use linear interpolation or cubic interpolation.

When the ASRC option is enabled, you must provide a drifting value (between -0.04 ~ +0.04) to let the Cadence ASRC generate a little bit more or less samples than what should be generated by the setting of Fs In and Fs Out to avoid buffer overflow or underflow.

In this document, we test Fs in = 33 kHz, Fs in = 47 kHz, and Fs in = 48 kHz, while Fs Out is fixed at 48 kHz. We enable the ASRC option to match/synchronize the input audio data rate and the output audio data rate. We also enable the cubic interpolation option to achieve better converted audio quality (lower THD+N). When you enable the ASRC, the ASRC drifting value calculation and update are the most critical parts. They are explained later on.

The Cadence ASRC converting performance is analyzed by recording the converted signal and observing the noise in the spectrum of the recorded signal.

3 Connecting asynchronized audio source and sink with Cadence ASRC

This section describes the whole system block and how the asynchronized audio source is connected to the i.MX RT685EVK.

3.1 System structure

To test the Cadence ASRC, you need the following boards and devices:

- A PC with an audio-editing application (such as Ocenaudio, Audacity, or Adobe Audition)
- A LPC55S69EVK board
- An i.MX RT685EVK board

This document comes together with three MCUXpresso projects:

- Lpc55S69_I2sDmaTxRxWithSweepGen
- Rt685_CadenceAsrc_DspPrg
- Rt685_CadenceAsrc_McuPrg

The Whole system is shown in Figure 1:

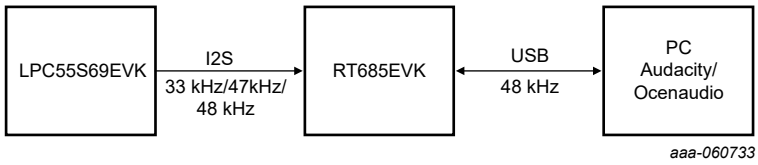

Figure 1. System block

Note:

The IDE for building the "Lpc55S69_I2sDmaTxRxWithSweepGen" project is MCUXpresso IDE v24.12. The IDE for building the "Rt685_CadenceAsrc_McuPrg" project is MCUXpresso IDE v24.12. The IDE for building the "Rt685_CadenceAsrc_DspPrg" project is Xtensa Xplorer Version 10.1.11.3000 and the tool chain is "nxp_rt600_RI23_11_newlib".

For more details on the IDE and toolchain installation, see the *Getting Started with Xplorer for EVK-MIMXRT685.pdf* document in the i.MX RT685 SDK.

When building the projects, due to header file inclusion, place the CM33 and DSP projects in the same folder outside the workspace folder.

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 I2S digital audio (sweeping tone) to the i.MX RT685EVK. When the I2S signal lines from the LPC55S69EVK are connected to the RT685EVK I2S signal lines, the LPC55S69 works as the I2S master and the i.MX RT685 works as the I2S slave. The scenario of the i.MX RT685 receiving an asynchronized audio source is created.

In the LPC55S69 project, there are three macro definitions controlling the external I2S source sampling frequency:

- `#define SetFsTo_33KHz 0`
- `#define SetFsTo_47KHz 1`
- `#define SetFsTo_48KHz 0`

Set only one of them to 1, build and run it, and you have the LPC55S69EVK working at the Fs that you select.

See section 3.2 in *How to Connect Asynchronized Audio Source and Sink with RT1170 ASRC* (document AN14631) for more details about the LPC55S69 project.

3.3 Wires connection between LPC55S69EVK and RT685EVK

By default, the i.MX RT685EVK board has the I2S lines running at 1.8 V. Make sure that the LPC55S69EVK I2S is running at 1.8 V too.

- Short the LPC55S69 P4.1 and P4.2 to select 1.8 V.
- Connect three I2S lines as follows:
  - RT685EVK BICK(P0_14): R397.2 to connect to LPC55S69EVK BICK: P17.14.
  - RT685EVK LRCK(P0_15): J47.6 to connect to LPC55S69EVK LRCK: P17.12.
  - RT685EVK I2SDataIn(P0_16): J47.7 to connect to LPC55S69EVK I2SDataOut: P18.3.

- Connect the GNDs:
  - Connect LPC55S69EVK P17.7, P23.8, and P24.8 with RT685EVK J31.11, J29.6, J28.7. Connect more than one GND line to have better I2S signal quality.

4 RT685 ASRC testing project

This section describes the testing functions of the RT685 ASRC project and how to realize the functions.

The i.MX RT685 HiFi4 project has two configurations, Debug and Release. For building and running on the RT685EVK board, select the Release configuration (with all the optimization options ON). If you have the condition to run the Xtensa Xplorer software simulation, switch to the Debug configuration and launch the soft-simulation, which converts the input WAV files to the output WAV files without connecting to a real i.MX RT685EVK board. This is more straightforward to evaluate the Cadence ASRC.

4.1 Cadence ASRC API functions

The i.MX RT685SDK contains the Cadence ASRC library and an example testing project that can run in the Xtensa Xplorer software simulation. The i.MX RT685 HiFi4 DSP project of this document reuses the Cadence ASRC initialization codes from this simple test example.

Set the following primary parameters in the Cadence ASRC initialization procedure: input Fs, output Fs, number of channels, audio sample width, input chunk size, ASRC enabled or not, cubic interpolation enabled or not, and the ASRC drifting value.

Here is an example of how to set the input Fs:

```
err_code = (*p_xa_process_api)(xa_process_handle,
XA_API_CMD_SET_CONFIG_PARAM,
XA_SRC_PP_CONFIG_PARAM_INPUT_SAMPLE_RATE,
&def_user_config.fs_in);
_XA_HANDLE_ERROR(p_proc_err_info, "Set Input Fs Error", err_code);
```

Let's assume that the input Fs is set to 32000, the output Fs is set to 48000, and the ASRC is not enabled. The Cadence ASRC main processing performs the Synchronized Sampling Rate Converting (SSRC). Every 32 input samples going into the Cadence ASRC generate 48 output samples.

Let's assume that the input Fs is set to 32000, the output Fs is set to 48000, the ASRC is enabled, and the drifting is set to 0.01. The Cadence ASRC main processing performs the Asynchronized Sampling Rate Converting (ASRC). Every 32000 input samples accumulated have 48000*1.01=48480 output samples accumulated if the drifting value is set to 0.01 and not changed during the ASRC processing.

If the cubic interpolation is set to enabled, the Cadence ASRC performs cubic interpolation. Otherwise, it performs linear interpolation. The cubic interpolation provides lower THD+N of the converted signal. The cubic interpolation uses more MIPS (MCPS).

Note: *The drifting value must be between* ***-0.04 ~ +0.04****. The Fs in and the Fs out are limited numbers of standard audio Fs. It is not possible to match the source and sink of the arbitrary Fs.*

In the the i.MX RT685 demo project of this document, the Cadence ASRC is set to ASRC enabled and the cubic interpolation is enabled. After the initialization is done, the i.MX RT685 HiFi4 DSP waits for the audio frame event from the CM33 side. When an audio frame event is received, it means that a certain number audio samples are ready to be fed into the Cadence ASRC. The HiF4 DSP calls the main processing procedure, which includes setting up the input and output buffer address, updating the drifting value (if it is changed), and calling the main ASRC function:

```
err_code = (*p_xa_process_api)(xa_process_handle,
XA_API_CMD_EXECUTE,
```

```
XA_CMD_TYPE_DO_EXECUTE,
NULL);
_XA_HANDLE_ERROR(p_proc_err_info, "Exec Error", err_code);
```

After the main processing function, it calls another function to get the number of generated samples:

```
err_code = (*p_xa_process_api)(xa_process_handle,
XA_API_CMD_GET_CONFIG_PARAM,
XA_SRC_PP_CONFIG_PARAM_OUTPUT_CHUNK_SIZE,
OutputSampleNum);
_XA_HANDLE_ERROR(p_proc_err_info, "Get Output Chunk Size Error", err_code);
```

Each time you call the main processing function, the number of output samples may change. The ratio of the output sample number and the input sample number in one call to the main ASRC processing function may not reflect the FsIn/FsOut ratio. In a long period of time, the ratio of accumulated input sample numbers and accumulated output sample numbers equals the FsIn/FsOut ratio multiplied by (1+drifting) precisely.

4.2 i.MX RT685 system signal flowchart

The RT685 CM33 project configures and enables the I2S audio and the USB audio interface. The i.MX RT685 HiFi4 DSP project performs the ASRC by calling the Cadence ASRC APIs. To test with varieties of signal source and different working modes, the RT685 CM33 project defines and implements a simple single-character USB COM command set. All of this forms the system internal flowchart, shown in Figure 2:

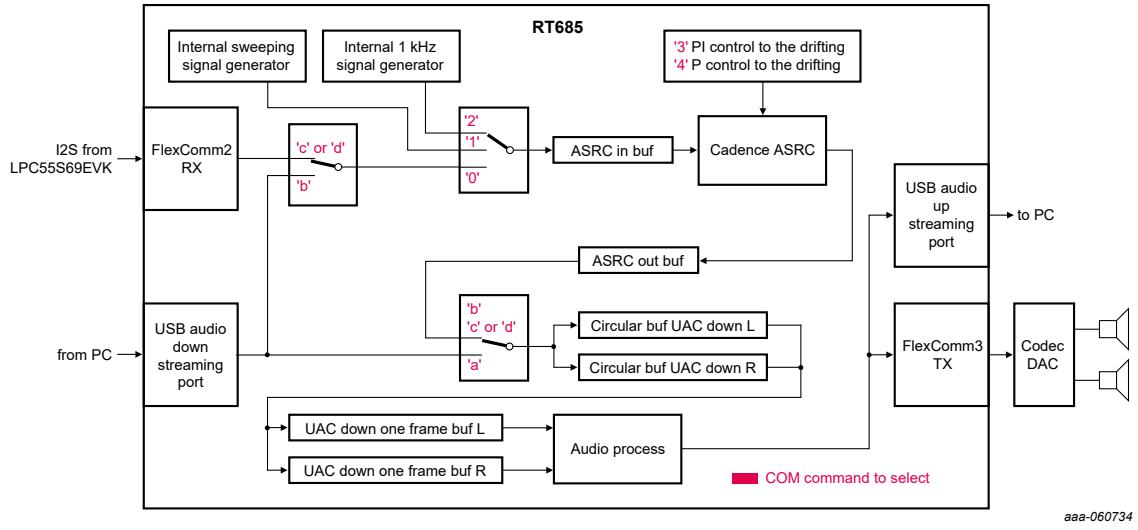

Figure 2. RT685 internal signal flowchart

4.3 Audio DMA and I2S configuration

The i.MX RT685 CM33 project configures the I2S ports in the same way as the other I2S driver examples in the i.MX RT685 SDK:

- I2S2(Flexcomm2): I2S slave, DMA RX frame size = 6 samples.
- I2S2(Flexcomm3): I2S master, DMA TX frame size = 6 samples, Fs = 48 kHz.

To test the Cadence ASRC, we implemented two circular buffers with the same buffer size for the left and right channels. The AOD of these circular buffers is monitored and used for the ASRC drifting calculation. The size of the circular buffer can be 60 or 360 samples.

```
//#define AsrcBufLengthInSamples 60
#define AsrcBufLengthInSamples 360
```

The above macro is defined in the header file that is included by both the i.MX RT685 CM33 project and the i.MX RT685 HiFi4 DSP project. To test with the USB audio down streaming, select 360 and rebuild both the CM33 and the DSP projects. It needs more space to bear the USB audio streaming rate change. If only external I2S is selected as the ASRC source, we can select 60 and rebuild both the CM33 and the DSP projects. This provides much lower latency of the signal IO latency (less than 2 ms).

The reason for the I2S DMA frame size being 6 is to achieve lower DMA buffering size and easier calculation to the AOD of the ASRC input circular buffer.

Since the I2S2(FlexComm2) connects to the external I2S signal, there is the risk that the external I2S clock signal is invalid (corruption in clock timing). This happens when attaching and detaching the external I2S source. The i.MX RT685 CM33 project monitors the clocking status of the I2S2. It closes the I2S2 DMA transferring when it sees an error (detaching) and reenables the DMA transferring when it sees a new correct I2S signal coming. This ensures that the external I2S source can be attached or detached safely at any time.

4.4 USB audio and USB COM interface

The i.MX RT685 CM33 project realizes the USB Audio Class 2.0 and the USB CDC COM port:

- USB audio down streaming: two channels, 32-bit, 48 kHz.
- USB audio upstreaming: four channels, 32-bit, 48 kHz.

The USB audio interface is UAC2.0. So that the USB audio packet is transferred at 8 kHz, each of the six samples (at 48-kHz Fs) is transferred once every 125 µs. To get better AOD checking to the circular buffer when the USB audio down streaming is selected as the source of the Cadence ASRC, we need the input/output chunk size to the circular buffer to be the same. This is another reason why we select to set the I2S DMA frame size to be six samples.

To test with varieties of signal source and different working modes (USB audio down streaming mode, ASRC drifting calculation mode), a simple single-character USB COM command set is defined and implemented in the i.MX RT685 CM33 project. The command list is as follows:

- Type and send a/A to select the USB audio down streaming synchronized by the CTimer.
- Type and send b/B to select the USB audio down streaming synchronized by the Cadence ASRC.
- Type and send c/C to select the external 33-kHz Fs I2S source converted by the Cadence ASRC.
- Type and send d/D to select the external 48-kHz Fs I2S source converted by the Cadence ASRC.
- Type and send 0 to stop the DSP generating signal; the received external I2S audio is not overwritten.
- Type and send 1 to let the DSP generate the sweeping tone and overwrite the received external I2S audio.
- Type and send 2 to let the DSP generate a 1-kHz tone and overwrite the received external I2S audio.
- Type and send 3 to select the PI control for the drift value calculation.
- Type and send 4 to select the P control for the drift value calculation.
- Type and send: PID: 123 1234 12345 to set Kp Ki Kd.

The CM33 project also prints some watch values in the USB COM port RX window. By default, they are the PID Err accumulated value, the drifting value * 10000, the AOD of the circular buffer, and the HiFi4 DSP cycle counts for calling the Cadence ASRC main processing function. These values are copied to the excel table so that we can analyze the PID converge and the Cadence ASRC MIPS (MCPS).

When playing and recording the USB audio, we must ensure that the recording audio enhancement and the playing audio enhancement are turned off. See section 4 of *How to Connect Asynchronized Audio Source and Sink with RT1170 ASRC* (document AN14631) for details.

4.5 ASRC drifting calculation: P control and PI control

In the demo project of this document, we put the Cadence ASRC-generated samples to a circular buffer and monitored the Amount Of Data (AOD) of this circular buffer. If the AOD is becoming bigger and tends to overflow, decrease the ASRC drifting value a little bit. If the AOD is becoming lower and tends to underflow, increase the ASRC drifting value a little bit. We implemented a PID feedback control logic for the drifting value calculation.

PID means stands for Proportional Integral and Derivative. It is a feedback-control logic that calculates the control value from the error value to have the error value as close to zero as possible. In this demo, the control value is the ASRC drifting value. If the error value is "the AOD - half of the circular buffer", it means that the feedback control purpose is to keep the AOD at the half of the circular buffer. We only use the P control when the USB audio down streaming is selected to the source of the Cadence ASRC. We only use the PI control when an external I2S source is selected.

The major P control calculation for the ASRC drifting value is:

```
AsrcDriftingValueTarget=Kp*Err;
```

The simplified PI control calculation for the ASRC drifting value is:

```
AsrcDriftingValueTarget=Kp*Err+Ki*PI_Control_SampleAOD_ErrAccumulated;
```

If the PID control is used, the calculation for the ASRC drifting value is:

```
AsrcDriftingValueTarget=Kp*Err+Ki*PI_Control_SampleAOD_ErrAccumulated+Kd*(Err-
PreErr);
```

Note: *The derivation part of the PID control is not used, because the "Err" value does not change too much. The derivation part is only helpful in a system that has a sudden "Err" change.*

In the above snippets:

- Kp and Ki are the PID coefficients.
- Err is the difference between the current AOD and the target AOD, which is a half of the total circular buffer size.
- PreErr is the previous Err. It is used to get the derivation of Err .
- PI_Control_SampleAOD_ErrAccumulated is the accumulated Err (the integration of Err) and it should be limited to a proper range.
- AsrcDriftingValueTarget is the calculation result that is used to update the Cadence ASRC drifting value.

When the USB audio down streaming is selected to be the Cadence ASRC source, we know that the input audio Fs of 48 kHz is very close to the i.MX RT685 local 48-kHz Fs. Most of the time, the AOD is exactly half of the circular buffer size and Err is 0. In about every 20 ~ 60 seconds, we can see a non-zero Err and need a quick converge (Err is going back to 0). This is the reason why we do the P control for USB audio down streaming ASRC.

When an external I2S audio source is selected to be the Cadence ASRC source, we do not know how far the actual external I2S Fs is away from the specified input Fs when initializing the Cadence ASRC. We compromise the converge time and ensure that a very stable drifting value is achieved. This is the reason why we do the PI control for an external I2S ASRC.

The i.MX RT685 demo of this document has already configured good Kp and Ki coefficient values:

- USB audio down streaming ASRC, P control: Kp = 1 / (3000 * 20).
- External I2S ASRC, PI control: Kp = 1 / 39500.
- External I2S ASRC, PI control: Ki = 1 / 1750000.
- External I2S ASRC, P control: Kp = 1 / 3000 (this is to compare between the P and PI control for an external I2S source).

To evaluate other PID coefficients, type and send "PID: 123 1234 12345 123456" in the USB COM window to set the Kp, Ki, Kd, and ErrAccMax values in the demo project. In this example, new Kp = 1 / 123, new Ki = 1 / 1234, new Kd = 1 / 12345, and new ErrAccMax = 123456.

5 Performance evaluation, ASRC conversion quality

To evaluate the Cadence ASRC conversion quality, we record the output signal from the Cadence ASRC (the USB audio upstreaming) and analyze the spectrum of the recorded signal. In this section, we also evaluate the PID converge time and the Cadence ASRC MIPS (MCPS).

5.1 Software simulation, ASRC with drifting value not changing

In the i.MX RT685 HiFi4 DSP project, the "paramfile_src_pp.txt" file contains commands to convert the existing WAV files in the Xtensa Xplorer software simulation. Disable the "ASRC_ENABLE_RUNTIME_DRIFT_TEST" macro in the HiFi4 DSP project to stop the drifting value change.

Select the Debug mode and build and launch the software simulation, if you have the conditions to run it. It takes several minutes for the WAV file to convert. Figure 3 is the spectrum of the converted WAV files. As the drifting value increases, the noise floor and harmonics increase more.

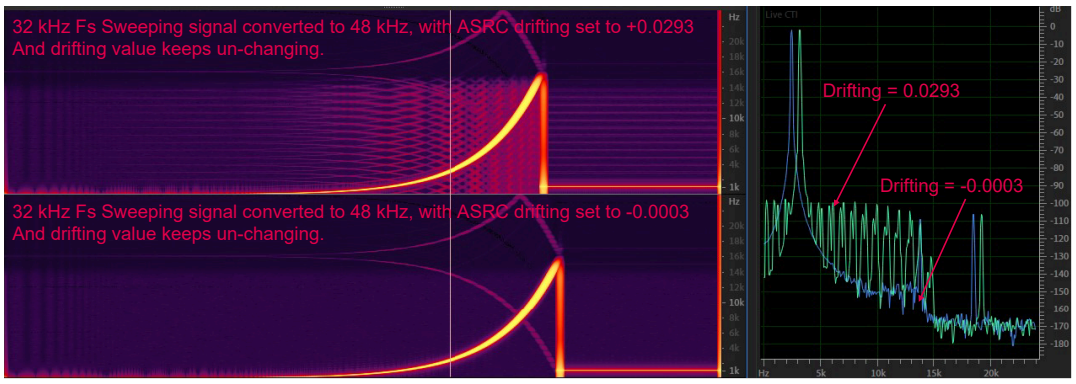

Figure 3. ASRC smaller drifting VS larger drifting

5.2 Software simulation, ASRC with drifting value changing

Enable the "ASRC_ENABLE_RUNTIME_DRIFT_TEST" macro in the HiFi4 DSP project to let the drifting value change. When running the software simulation with a different drifting changing speed, we get the spectrum shown in Figure 4. As the drifting value changes quicker, the noise floor and harmonics increase more.

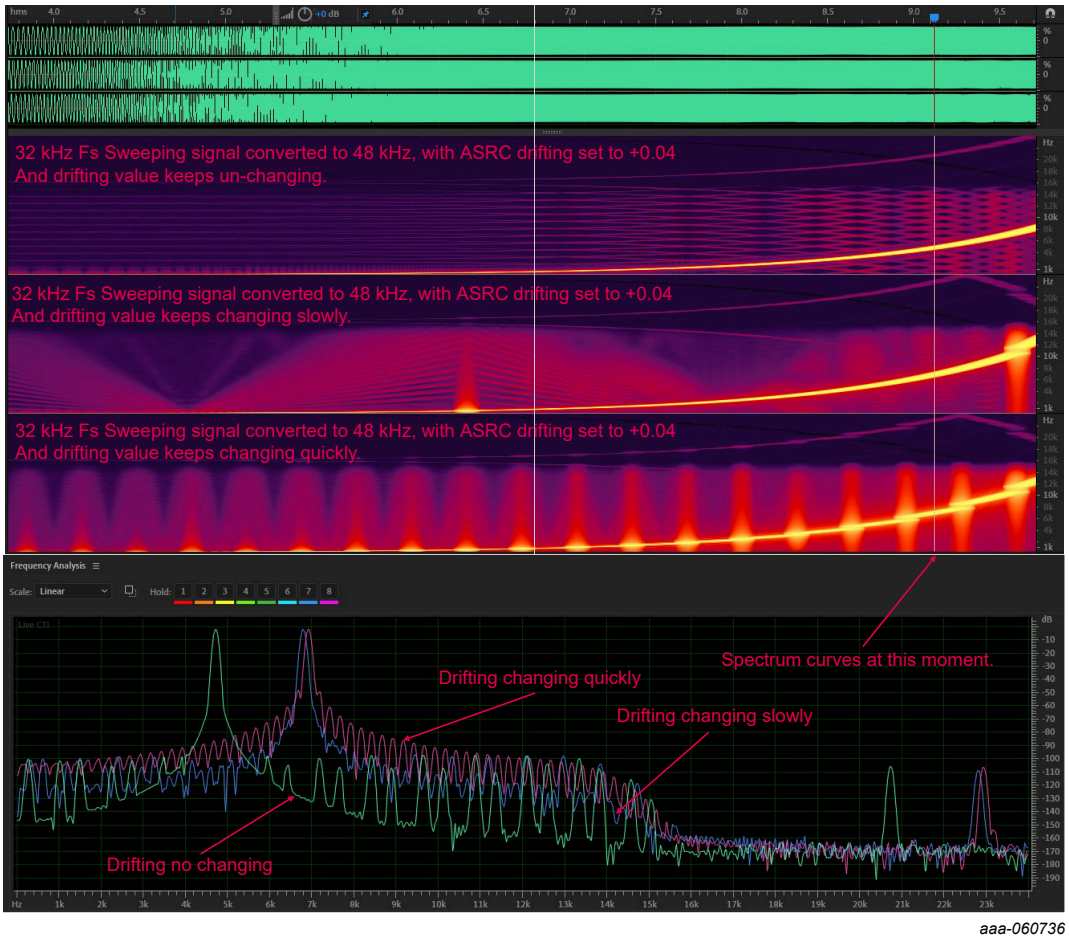

Figure 4. ASRC slower drifting change vs. quicker drifting change

5.3 External 33-kHz Fs to local 48-kHz Fs

In the LPC55S69 project, set **`#define`** `SetFsTo_33KHz 1`, build, and launch so that the i.MX RT55S69EVK starts to stream out I2S audio to the i.MX RT685EVK. In the USB COM port window, type and send "c" to select the external 33-kHz I2S source. In the recorded USB upstreaming, we can see the converted external I2S sweeping signal. Type and send "3" to select the PI control. Type and send "4" to select the P control. Type and send "1" to switch to the i.MX RT685 HiFi4 DSP generated sweeping data overwriting the received signal.

The LPC55S69 sweeping generator is not generating a perfect sweeping tone due to its computation power. We can let the i.MX RT685 HiFi4 DSP generate the sweeping or 1-kHz signal and overwrite what is received from the external LPC55S69EVK. In this way, the audio data is generated by the i.MX RT685 HiFi4 DSP, but still strictly synchronized by the external I2S Fs, so that the signal source to analyze is in a perfect spectrum shape and noise level.

All the following spectrum charts of the recorded signal are from the i.MX RT685 HiFi4 DSP self-generated sweeping, but synchronized by the external I2S, as if the external IS2 host LPC55S69 was streaming out a perfect sweeping tone in 32 bits.

After comparing, we see that the PI control provides a better noise floor.

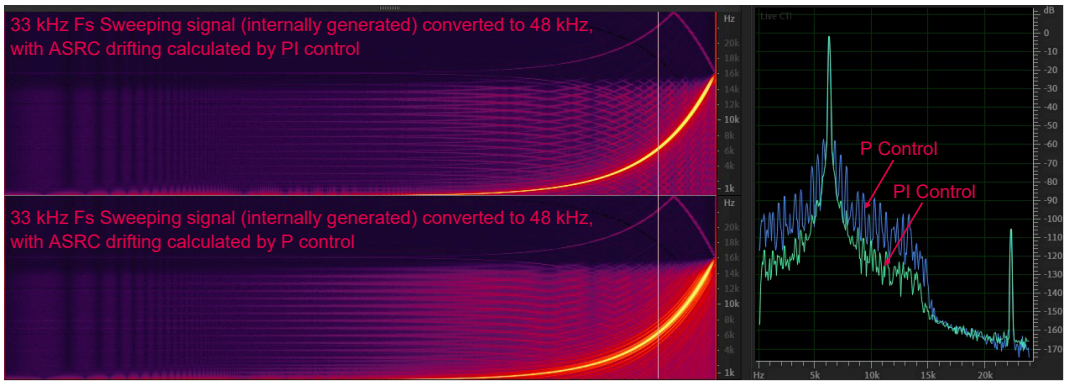

Figure 5. External 33-kHz I2S ASRC with drifting calculated by P or PI control

5.4 External 47-kHz Fs to local 48-kHz Fs

In the LPC55S69 project, set **`#define`** `SetFsTo_47KHz 1`, build, and launch, so that the i.MX RT55S69EVK starts to stream out I2S audio to the i.MX RT685EVK. In the USB COM port window, type and send "d" to select the external 47-kHz or 48-kHz I2S source. Type and send "3" to select the PI control. Type and send "4" to select the P control. Type and send "1" to switch to the i.MX RT685 HiFi4 DSP generated sweeping data overwriting the received signal.

After comparing, we see that the PI control gives a better noise floor.

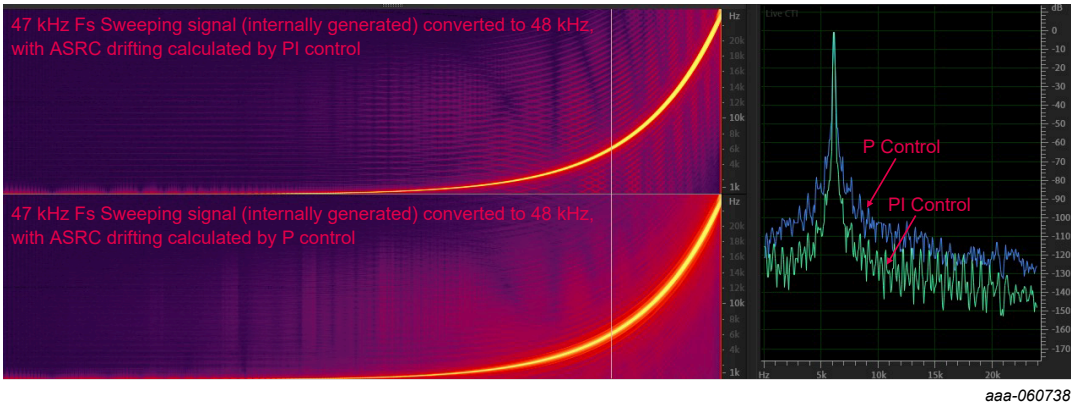

Figure 6. External 47-kHz I2S ASRC with drifting calculated by P or PI control

5.5 External 48-kHz Fs to local 48-kHz Fs

In the LPC55S69 project, set **`#define`** `SetFsTo_48KHz 1`, build, and launch, so that the i.MX RT55S69EVK starts to stream out I2S audio to the i.MX RT685EVK. In the USB COM port window, type and send "d" to select the external 47-kHz or 48-kHz I2S source. Type and send "3" to select the PI control. Type and send "1" to switch to the i.MX RT685 HiFi4 DSP generated sweeping data overwriting the received signal.

Because the input and output Fs are both 48-kHz, they only have a small difference. Most of the time, the drifting value calculated by the PI control is 0. The Cadence ASRC creates almost no noise when the drifting is 0. When the asynchronized external I2S finally makes the AOD not at the half (Err not zero), the PI control generates a none zero drifting value and keeps changing for a short period of time. Figure 7 shows the comparison between the drifting value not changing (being 0) and changing. In this test case, the changing drifting brings a slightly higher noise floor in the spectrum.

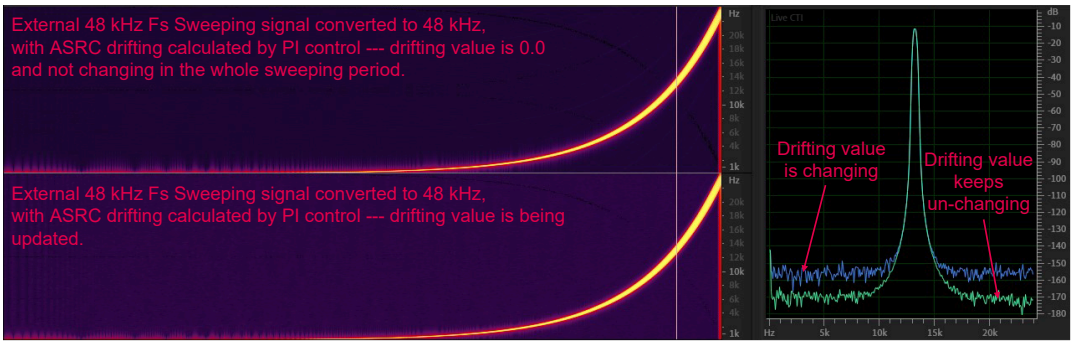

Figure 7. External 48-kHz I2S ASRC with drifting being 0 and changing

5.6 USB audio down streaming 48-kHz Fs to local 48-kHz Fs

In the USB COM port window, type and send "a" to let the 48-kHz USB audio down streaming use the CTimer and the audio PLL adjusting for synchronization. Type and send "b" to let USB audio down streaming 48KHz use the ASRC for synchronization, and in this choice P control is used.

The P control is changing the drifting value and the noise floor in the spectrum is almost the same when using the CTimer for the synchronization.

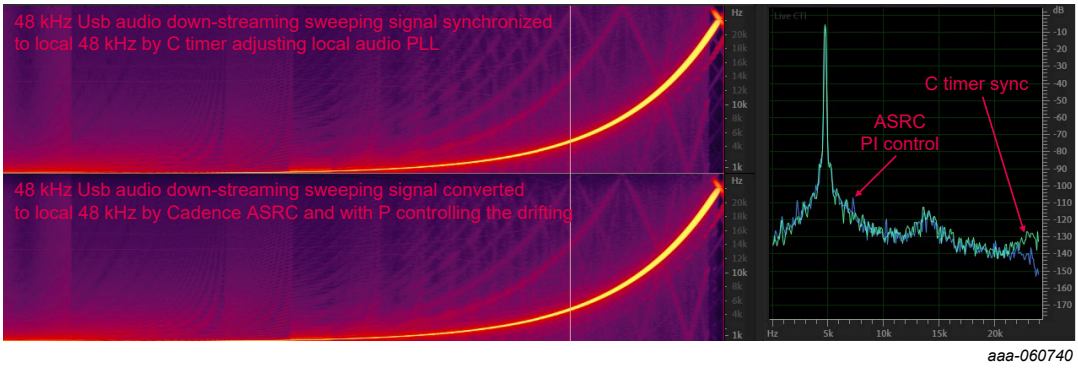

Figure 8. USB audio down streaming 48 kHz synchronized by CTImer and ASRC

Note: *Using the Cadence ASRC to synchronize the USB audio down streaming is only a practice to test the Cadence ASRC. Use the CTimer for synchronization in an ordinary application.*

5.7 ASRC drifting value converge: external 33-kHz Fs to local 48-kHz Fs

In the LPC55S69 project, set **`#define`** `SetFsTo_33KHz 1`, build, and launch, so that the i.MX RT55S69EVK starts to stream out I2S audio to the i.MX RT685EVK. In the USB COM port window, type and send "c" to select the external 33-kHz I2S source. Type and send "3" to select the PI control. Type and send "4" to select the P control. Type and send "2" to switch to the i.MX RT685 HiFi4 DSP generated 1-kHz data overwriting the received signal. Selecting 1 kHz is for easier watching of the converge from the spectrum plot.

Figure 9 shows that the converge time is 10 ~ 15 s when using the PI control. The P control converges immediately in 0.5 s.

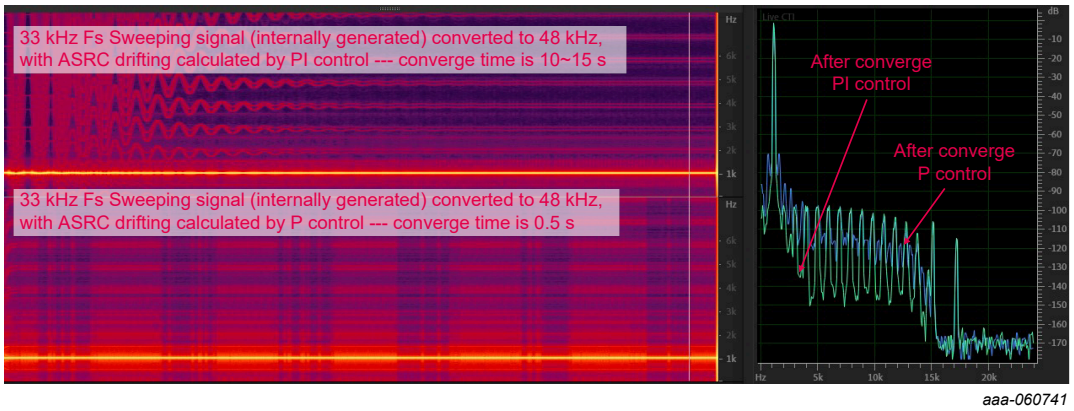

Figure 9. External 33-kHz I2S, PI control converge, and P control converge

The USB COM RX port window keeps receiving the printings from the CM33. The following figures are created from the data copied from the USB COM RX window. It shows the PID-related values changing trend (**`#define`** `AsrcBufLengthInSamples 360` is used).

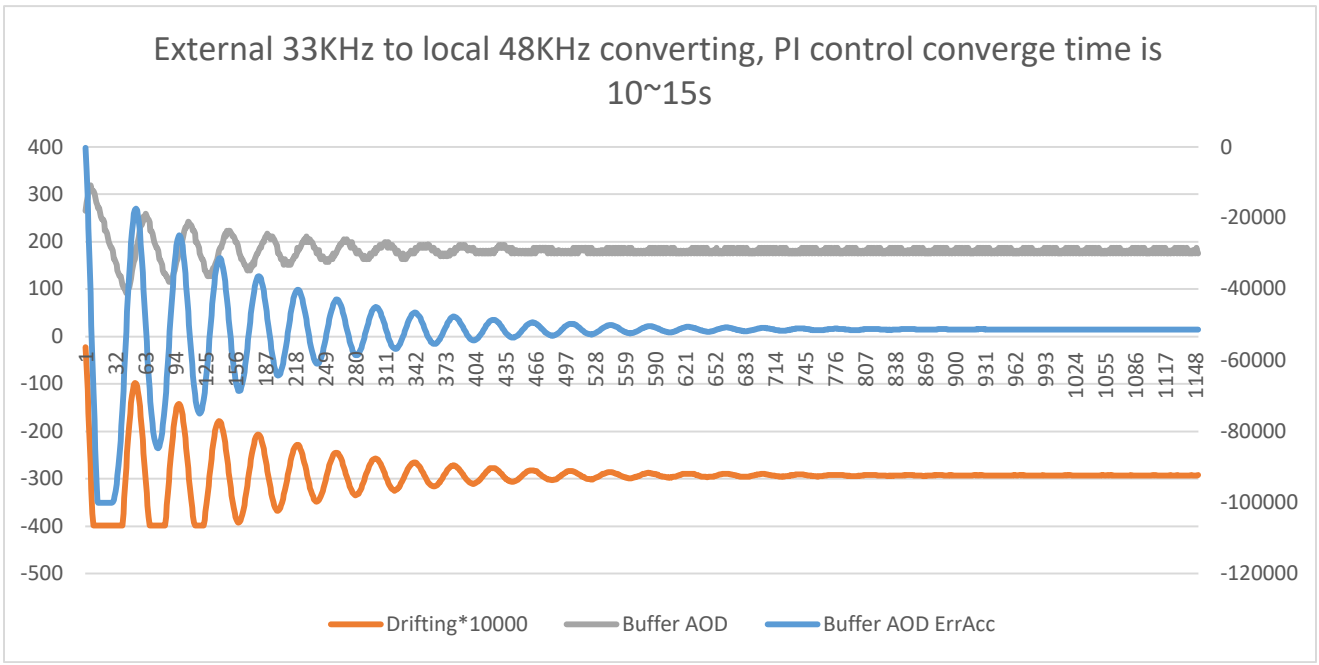

Figure 10. External 33-kHz I2S, PI control converge

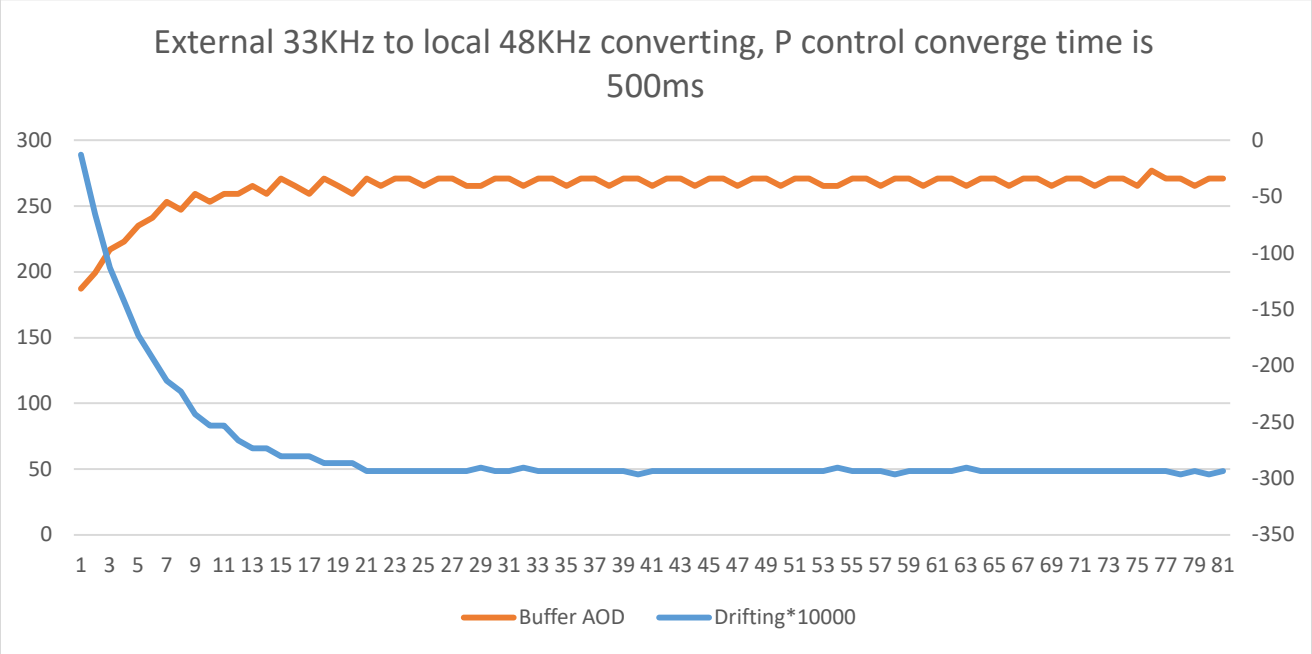

Figure 11. External 33-kHz I2S, P control converge

5.8 ASRC drifting value converge: external 47 kHz Fs to local 48 kHz Fs

In the LPC55S69 project, set **`#define`** `SetFsTo_48KHz 1`, build, and launch, so that the i.MX RT55S69EVK starts to stream out I2S audio to the i.MX RT685EVK. The other steps are the same as in the previous section. Figure 12 shows the converge time and the PID-related value changing trend.

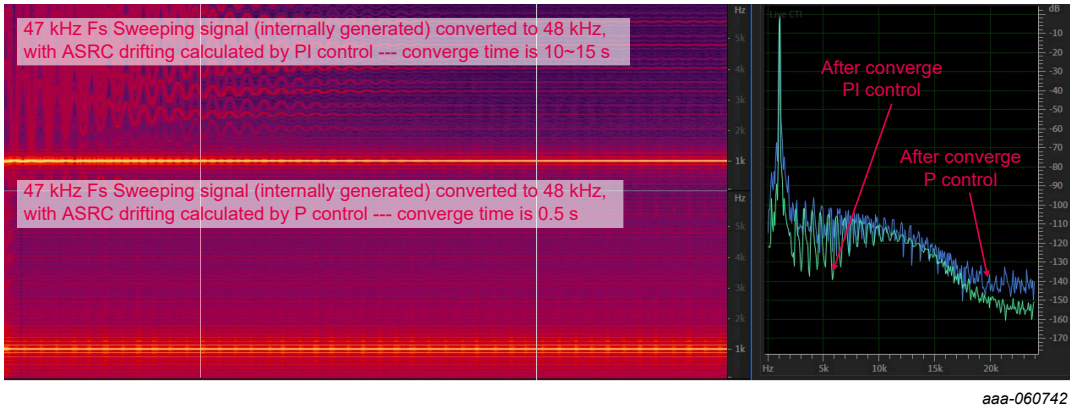

Figure 12. External 48-kHz I2S, PI control converge and P control converge

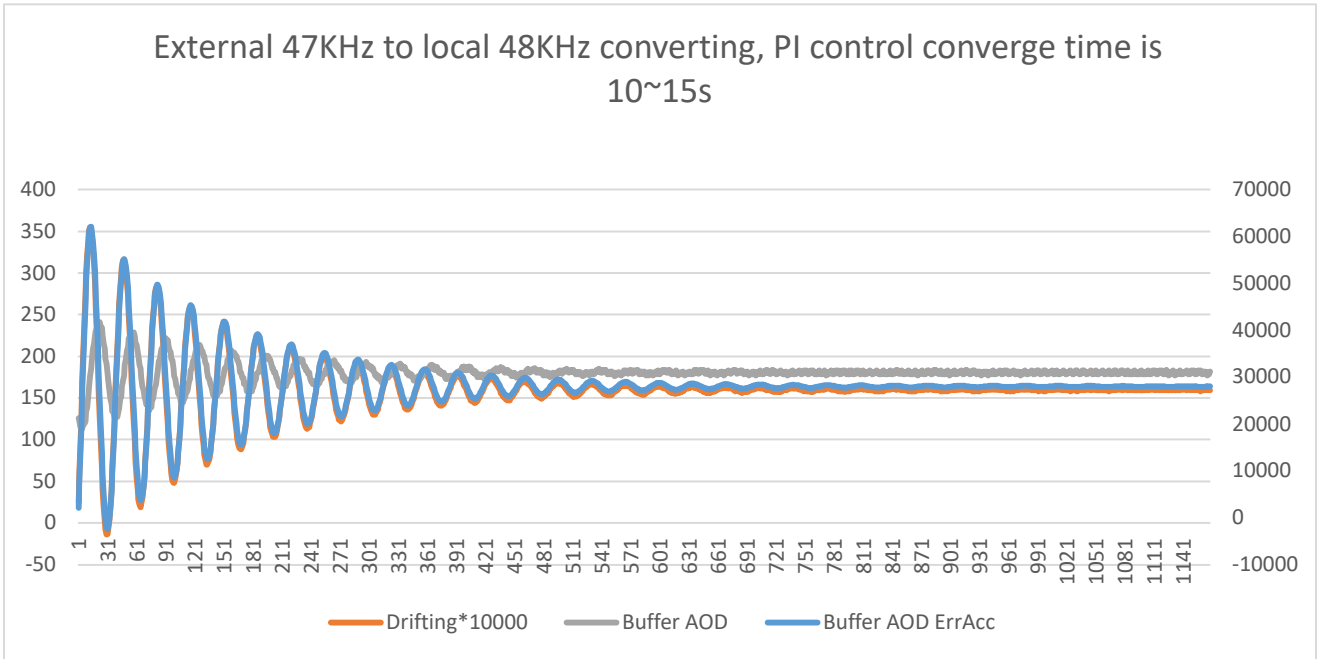

Figure 13. External 47-kHz I2S, PI control converge

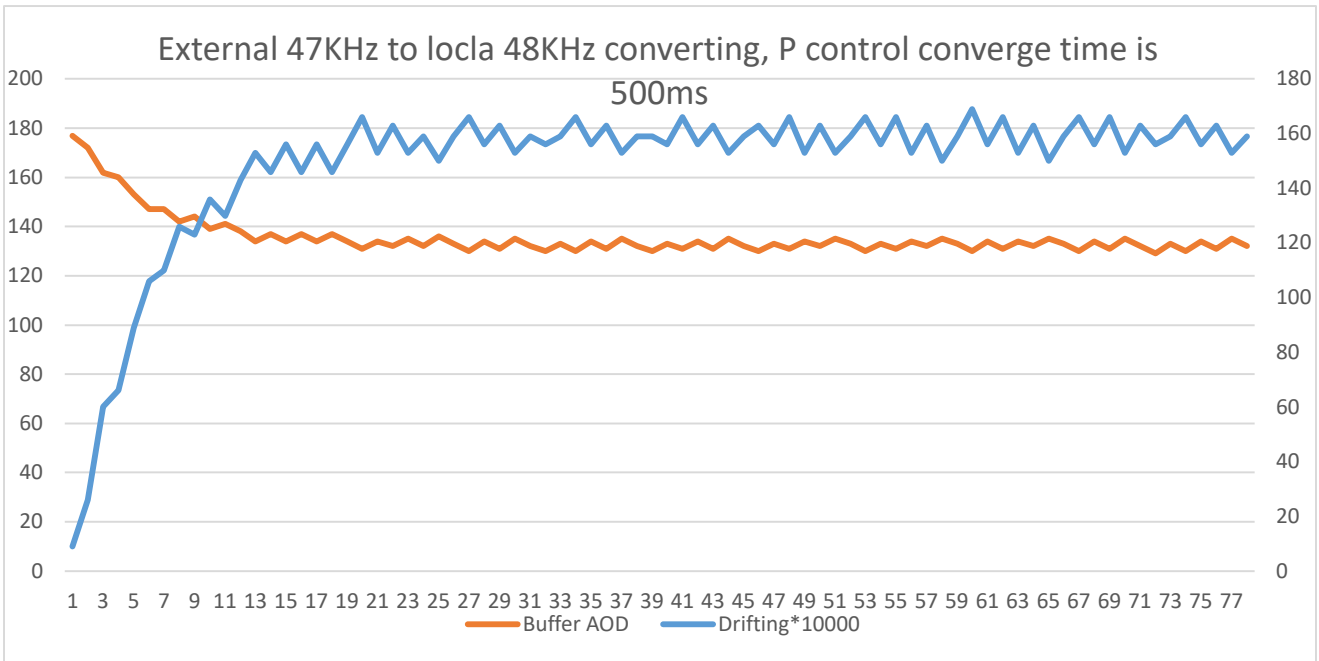

Figure 14. External 47-kHz I2S, P control converge

5.9 Cadence ASRC latency

In the i.MX RT685 HiFi4 DSP project, set `#define EnableLatencyTest` to 1. Build and run the demo. Use the single-character command "b" to select the USB audio down streaming as the input to the Cadence ASRC. Use the single-character command "2" to enable the internal 1-kHz tone generator to overwrite the received audio.

In this scenario, we make a sudden amplitude decrease once every 1 s (or similar) in the right channel in the internal tone-sweeping generator. At the moment when the internal tone-sweeping generator decides to change the amplitude, the USB upstreaming left channel amplitude is also decreased immediately. The USB audio upstreaming left channel reflects the signal amplitude change immediately, while the right channel amplitude change is delayed by the ASRC converting.

Observing the recorded USB upstreaming audio, we can exactly measure the latency in samples. Figure 15 shows the latency of 201 samples. Among this latency period, 180 samples are due to the ASRC input buffer delay effect. The actual latency of the Cadence ASRC is 201 - 180 = 21 samples (0.44 ms at 48 kHz).

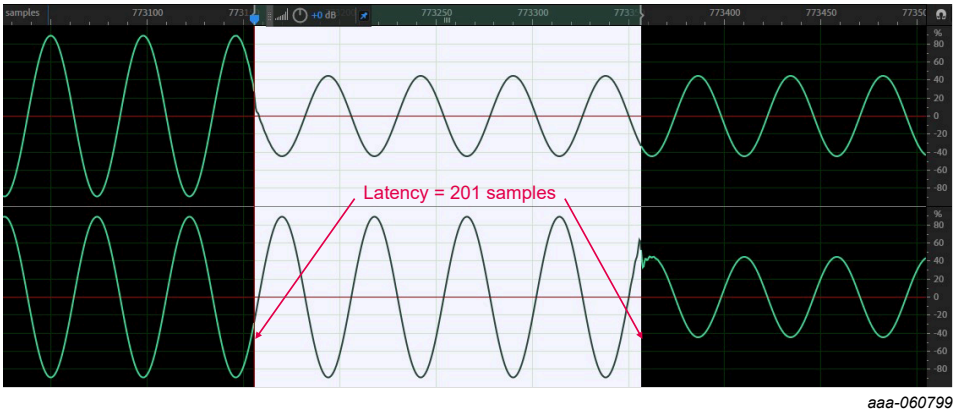

Figure 15. Cadence ASRC latency when Fs In is 48 kHz and Fs Out is 48 kHz

When Fs In is 32 kHz and Fs Out is 48 kHz, the measured Cadence ASRC latency has a similar value.

5.10 Cadence ASRC MIPS

The last column of the data received in the USB COM RX port window is the HiFi4 DSP cycle count of each time calling the Cadence ASRC main processing function. The range is around 2500 ~ 3500 cycles (every six samples). This means that the equivalent MIPS(MCPS) is: (2500 ~ 3500) / 6 * 48000 / 1000000 = 20 ~ 28 MIPS (MCPS, MHz).

According to "Cadence Sample Rate Converter For HiFi DSPs", average CPU load is 14.2 MHz when Fs in is 44.1 kHz, Fs out is 48 kHz, ASRC is ON, cubic interpolation is ON, and audio sample width is 24 bits.

In the demo of this document, the DSP code is running in the external flash space, the input chunk size is six samples, and 20 ~ 28 MHz is a reasonable MIPS (MCPS).

6 Conclusion

- When you want to connect asynchronized audio source and sink and there is no hardware ASRC, use the software ASRC.
- The Cadence ASRC is an highly efficient library that can convert asynchronized audio with drifting change in real time and the converted signal THD+N is less than -80 dB.
- Set a circular buffer with a proper buffer size for as low latency as possible and to support the PID feedback control to the drifting value.
- The ASRC drifting is controlled by the PID feedback control. Tune the PID coefficients to get the PID converged as quick as possible. The Err has as small jittering as possible.

The table of contents updates automatically whenever you open the document. However, there are times when you want to update the TOC when you are working within the document.

7 Abbreviations

Table 1. Abbreviations

Abbreviation	Description
AOD	Amount Of Data
SSRC	Synchronized Sample Rate Converting
ASRC	Asynchronized Sample Rate Converting
P control	Proportional control
PI control	Proportional Integral control
PID control	Proportional Integral Derivative control
MIPS	Million Instructions Per Second
MCPS	Million Cycles Per Second

8 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.

9 Revision history

Table 2. Revision history

Document ID	Release date	Description
AN14674 v.1.0	13 June 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.

Contents

1 Introduction 2
2 Cadence HiFi ASRC overview 2
3 Connecting asynchronized audio source and sink with Cadence ASRC 2
3.1 System structure 2
3.2 LPC55S69EVK as the I2S audio source 3
3.3 Wires connection between LPC55S69EVK and RT685EVK 3
4 RT685 ASRC testing project 4
4.1 Cadence ASRC API functions 4
4.2 i.MX RT685 system signal flowchart 5
4.3 Audio DMA and I2S configuration 5
4.4 USB audio and USB COM interface 6
4.5 ASRC drifting calculation: P control and PI control 7
5 Performance evaluation, ASRC conversion quality 8
5.1 Software simulation, ASRC with drifting value not changing 8
5.2 Software simulation, ASRC with drifting value changing 8
5.3 External 33-kHz Fs to local 48-kHz Fs 9
5.4 External 47-kHz Fs to local 48-kHz Fs 10
5.5 External 48-kHz Fs to local 48-kHz Fs 10
5.6 USB audio down streaming 48-kHz Fs to local 48-kHz Fs 11
5.7 ASRC drifting value converge: external 33-kHz Fs to local 48-kHz Fs 11
5.8 ASRC drifting value converge: external 47 kHz Fs to local 48 kHz Fs 13
5.9 Cadence ASRC latency 14
5.10 Cadence ASRC MIPS 15
6 Conclusion 15
7 Abbreviations 16
8 Note about the source code in the document 16
9 Revision history 16
Legal information 17