RV1106 Linux Audio Codec Developer Guide

ID: RK-KF-YF-934

Release Version: V1.3.0

Release Date: 2023-12-18

Security Level: □Top-Secret □Secret □Internal ■Public

DISCLAIMER

Trademark Statement

Rockchip Electronics Co., Ltd.

No.18 Building, A District, No.89, software Boulevard Fuzhou, Fujian,PRC

Website: www.rock-chips.com

Customer service Tel: +86-4007-700-590

Customer service Fax: +86-591-83951833

Customer service e-Mail: fae@rock-chips.com

Preface

Overview

This document primarily introduces the features and functions of RV1106/RV1103 Audio Codec, as well as common attribute configurations of ACodec.

Product Version

Chipset	Kernel Version
RV1106/RV1103	5.10

Intended Audience

This document (this guide) is mainly intended for:

Technical support engineers

Software development engineers

Revision History

Version	Author	Date	Change Description
V1.0.0	Xing Zheng	2022-05-12	Added the initial version of RV1106 Linux Audio Codec Developer Guide
V1.1.0	Xing Zheng	2022-08-31	Added commonly used playback and recording test commands of RV1106 Audio Codec
V1.2.0	Xing Zheng	2023-10-20	Fixed the `range` description error of ADC ALC node, added "DAC Control Manually" description
V1.3.0	Xing Zheng	2023-12-18	Fixed the `range` description error of ADC ALC node, and fiexed the reference of the invalid document.

Contents

1. Overview

1.1 Function Introduction

RV1106/RV1103 ACodec is an integrated audio IP module within RV1106/RV1103 SoC. It allows for the connection of an external analog microphone, through analog-to-digital conversion, captures external signals for recording and transmission to CPU. The CPU can also play back locally stored PCM audio digital signals after digital-to-analog conversion. The ACodec IP included in RV1106 and RV1103 is identical, but there are some limitations based on their usage scenarios (which will be discussed next).

Unless noted otherwise, the descriptions of ACodec apply to both RV1106 and RV1103.

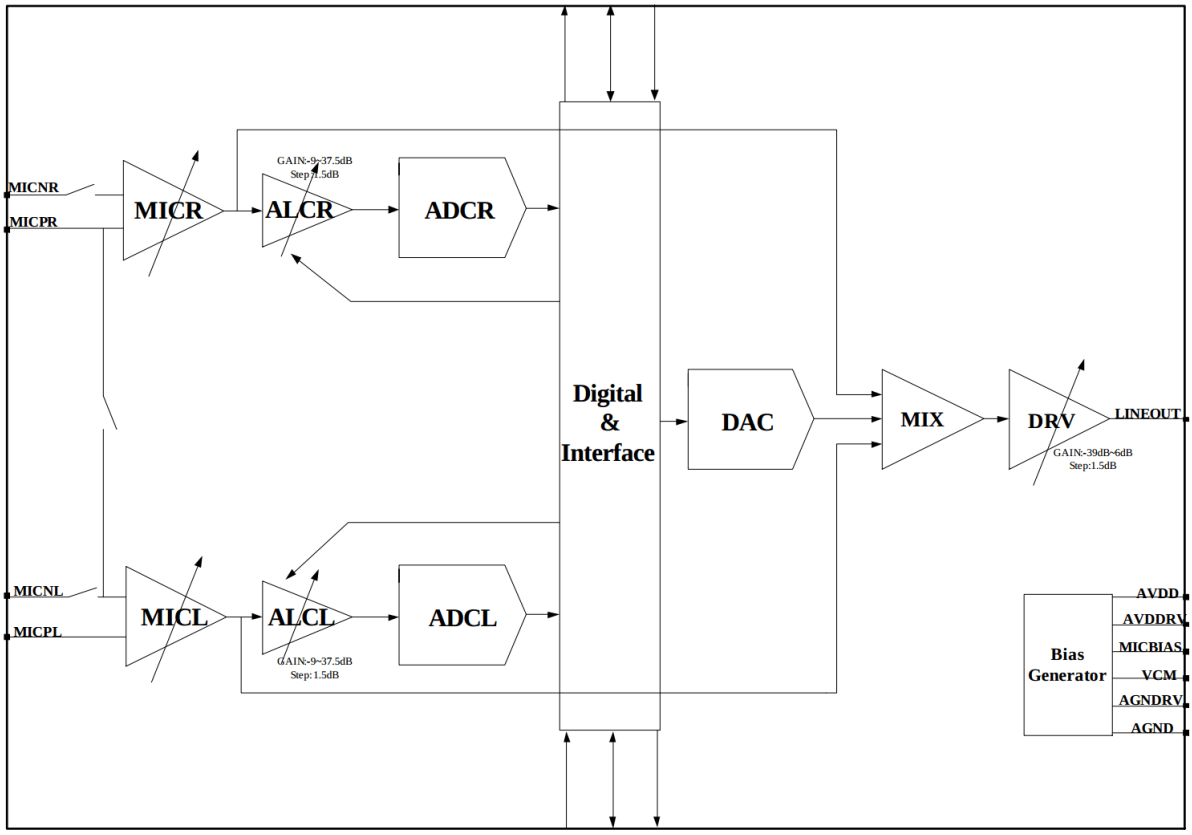
2. RV1106 ACodec Introduction

2.1 RV1106 ACodec Features

The basic features of RV1106/RV1103 ACodec are as follows:

- 24 bits DAC with 93dB(A-weighted) SNR
- Support 600Ω line output
- Low power: 2.5mA for playback
- 24 bits ADC with 92dB(A-weighted) SNR
- Support differential and single-ended microphone or line input
- Low power: 5mA for stereo recording
- Automatic Level Control (ALC) for smooth audio recording
- Low power: less than 0.05mA for standby
- Programmable input and output analog gains
- Digital interpolation and decimation filter integrated
- Sampling rate of 8kHz/12kHz/16kHz/24kHz/32kHz/44.1KHz/48KHz/96KHz
- 1.8V supply for analog and 0.9V supply for digital

2.2 RV1106 ACodec Block Diagram



From the block diagram, it can be found that both RV1106 and RV1103 support 2 ADC inputs (for recording) and 1 DAC output (for playback). It's important to note that in differential mode, RV1106 can support a maximum of 2 microphones, while RV1103, in differential mode, only supports the connection of 1 microphone. In single-ended mode, both RV1106 and RV1103 can support a maximum of 2 microphones each.

SoC ID	Maximum number of MICs in single-ended mode	Maximum number of MICs in differential mode
RV1106	2	2
RV1103	2	1

For typical MIC connection methods, please refer to RK EVB drawings and hardware design reference guide.

3. RV1106 ACodec Software Configuration

3.1 RV1106 ACodec Kernel Configuration

Taking RK kernel-5.10 SDK code as an example, you can refer to the following dtsi file:

```
arch/arm/boot/dts/rv1106-evb-v10.dtsi
```

Firstly, define an acodec_sound node, which describes some basic information of the sound card and references the DAI (Digital Audio Interface) i2s0_8ch on the controller side and the DAI acodec on the peripheral side:

```
    acodec_sound: acodec-sound {
        compatible = "simple-audio-card";
        simple-audio-card,name = "rv1106-acodec";
        simple-audio-card,format = "i2s";
        simple-audio-card,mclk-fs = <256>;
        simple-audio-card,cpu {
            sound-dai = <&i2s0_8ch>;
        };
        simple-audio-card,codec {
            sound-dai = <&acodec>;
        };
    };
```

Enable I2S0:

```
&i2s0_8ch {
    #sound-dai-cells = <0>;
    status = "okay";
};
```

Enable ACodec and define the GPIO pin that controls the external power amplifier:

```
&acodec {
    #sound-dai-cells = <0>;
    pa-ctl-gpios = <&gpio1 RK_PA1 GPIO_ACTIVE_HIGH>;
    status = "okay";
};
```

In the above example, the I2S0 inside RV1106/RV1103 can be bound to ACodec.

It should be noted that there is only one I2S controller inside RV1106/RV1103, which is I2S0. If the internal ACodec is enabled, I2S0 will be occupied at this time, and it is connected to ACodec through the SoC internal lead wires. Therefore, in this scenario, the pins connected to external IO by I2S0 will be unavailable.

3.2 RV1106 ACodec User Level Configuration

Usually, we can use the common and authoritative tinymix tool to obtain and set the control node information that the codec exposes to user layer. However, since tinymix tool was cropped in the SDK, rockit integrates the rk_mpi_amix_test tool that is equivalent to tinymix and can be used to list all contents information:

```
# rk_mpi_amix_test --list_contents
cmd parse result:
sound control id      : 0
control name          : (null)
control value         : (null)
list controls         : 0
list contents         : 1
Number of controls: 26
ctl     type    num     name                                    value
0       ENUM    1       I2STDM Digital Loopback Mode            ,
DisabledMode1Mode2Mode2 Swap
1       INT     1       ADC MIC Left Gain                       2 (range 0->3)
```

```
2       INT     1       ADC MIC Right Gain                      2 (range 0->3)
3       INT     1       ADC ALC Left Volume                     6 (range 0->31)
4       INT     1       ADC ALC Right Volume                    6 (range 0->31)
5       INT     1       ADC Digital Left Volume                 195 (range 0-
>255)
6       INT     1       ADC Digital Right Volume                195 (range 0-
>255)
7       ENUM    1       ADC HPF Cut-off                         , OffOn
8       INT     1       ALC AGC Left Volume                     12 (range 0->31)
9       INT     1       ALC AGC Right Volume                    12 (range 0->31)
10      INT     1       ALC AGC Left Max Volume                 7 (range 0->7)
11      INT     1       ALC AGC Right Max Volume                7 (range 0->7)
12      INT     1       ALC AGC Left Min Volume                 0 (range 0->7)
13      INT     1       ALC AGC Right Min Volume                0 (range 0->7)
14      ENUM    1       ALC AGC Left Switch                     , OffOn
15      ENUM    1       ALC AGC Right Switch                    , OffOn
16      ENUM    1       AGC Left Approximate Sample Rate        ,
96KHz48KHz44.1KHz32KHz24KHz16KHz12KHz8KHz
17      ENUM    1       AGC Right Approximate Sample Rate       ,
96KHz48KHz44.1KHz32KHz24KHz16KHz12KHz8KHz
18      ENUM    1       ADC Mode                                ,
DiffadcLSingadcLDiffadcRSingadcRSingadcLRDiffadcLR
19      ENUM    1       ADC MICBIAS Voltage
VREFx0_8VREFx0_825VREFx0_85VREFx0_875, VREFx0_9VREFx0_925VREFx0_95V
REFx0_975
20      ENUM    1       ADC Main MICBIAS                        Off, On
21      ENUM    1       ADC MIC Left Switch                     , WorkMute
22      ENUM    1       ADC MIC Right Switch                    , WorkMute
23      INT     1       DAC LINEOUT Volume                      26 (range 0->30)
24      INT     1       DAC HPMIX Volume                        1 (range 0->2)
25      ENUM    1       DAC Control Manually                    , NoneOffOn
```

There are lots of nodes, and it is going to introduce some commonly used node information here:

```
0       ENUM    1       I2STDM Digital Loopback Mode            ,
DisabledMode1Mode2Mode2 Swap
```

This node is actually the control node exposed to users by I2S0, but it is closely related to ACodec. It indicates whether the I2STDM controller is working in recovery mode. In the node information:

- Disabled: Default state, recovery mode is disabled
- Mode1: Suitable for 4ch usage scenarios. Channels 1-2 are data of MIC pickup, and channels 3-4 are recovery data.
- Mode2: Suitable for 2ch usage scenarios. The left channel is data of MIC pickup, and the right channel is the playback data of the right channel.
- Mode2 Swap: suitable for 2ch usage scenarios. The left channel is the recovery data of the left channel played, and the right channel is the MIC pickup data. Mode2 Swap and Mode2 channel order are opposite

Therefore, if Mode2 and Mode2 Swap are playing 2ch data, it is best to perform software MIX before sending data to ACodec DAC, so that the recovery reference signal is closer to the signal of the real echo path.

Remarks: the selected item displayed in tinymix style is added with a comma "," on the left. For example, the displayed ", Disabled" means that the "Disabled" state is currently selected, and so on.

```
1       INT     1       ADC MIC Left Gain                       2 (range 0->3)
2       INT     1       ADC MIC Right Gain                      2 (range 0->3)
```

ACodec ADC Boost Gain is the codec analog volume. The actual valid value range is 1 ~ 3:

- vol 0: Disabled and not recommended
- vol 1: 0dB
- vol 2: 20dB
- vol 3: 12dB

```
3       INT     1       ADC ALC Left Volume                     6 (range 0->31)
4       INT     1       ADC ALC Right Volume                    6 (range 0->31)
```

ACodec ADC ALC PGA Gain is the codec analog volume. The value range is 0~31:

- min: -9dB (vol: 0)
- max: +37.5dB (vol: 31)
- step: +1.5dB
- location: 0dB (vol: 6)

```
5       INT     1       ADC Digital Left Volume                 195 (range 0-
>255)
6       INT     1       ADC Digital Right Volume                195 (range 0-
>255)
```

ACodec ADC Digital Gain is the codec digital volume. The value range is 0~255:

- min: -97.5dB (vol: 0)
- max: +30dB (vol: 255)
- step: +0.5dB
- location: 0dB (vol: 195)

```
18      ENUM    1       ADC Mode                                ,
DiffadcLSingadcLDiffadcRSingadcRSingadcLRDiffadcLR
```

"Diff" is the abbreviation of "Differential"; "Sing" is the abbreviation of "Single-end". Used to configure the ACodec ADC to work in differential or single-ended mode. The default is "Diff" differential mode. And in order to minimize power consumption to the greatest extent, the ADC only enables the L left channel. Therefore, "DiffadcL" is the default preference.

It should be noted that due to different chip orientation, RV1103 does not support two-channel pickup in differential mode, that is, it does not support the "DiffadcLR" option.

```
23      INT     1       DAC LINEOUT Volume                      26 (range 0->30)
```

ACodec DAC Lineout Gain, the volume value range is 0~30:

- min: -39dB (vol: 0)
- max: +6dB (vol: 30)
- step: +1.5dB
- location: 0dB (vol: 26)

```
24      INT     1       DAC HPMIX Volume                        1 (range 0->2)
```

ACodec DAC HPMIX Gain, which is the preamp Gain of Lineout, is usually not set. The value range of the volume value is actually 1 and 2:

- vol 0: Disabled and not recommended
- vol 1: 0dB
- vol 2: 6dB

```
25      ENUM    1       DAC Control Manually                    , NoneOffOn
```

Developers usually use the media framework with the AO Stream is always on (always enabled the sound card). When stopping playback, they only need to quickly turn off and on the peripheral PA. Therefore, a "DAC Control Manually" node is added here:

- None: Default status, the media framework should be actually closed and AO Stream be enabled to control the sound card's off and on.
- Off: When the AO Stream is always on, you can just manually turn off the peripheral PA to make the speaker playback silent.
- On: When the AO Stream is always on and the previous state is Off, you can manually turn on the peripheral PA again to resume playback of the speaker.

The above are `codec control` node that are commonly used by developers during the development process. For other options, you can refer to the `control name` and options and configure and adjust them according to the specific project.

4. RV1106 ACodec Commonly Used Test Commands

Since RV1106/RV1103 operating environment is usually based on the rockit framework, the following are combination of basic examples and rockit related commands.

4.1 Commonly Used Playback Test Commands

The following uses a PCM with a sampling rate of 16kHz/2ch/16bit as the playback sound source. The rk_mpi_ao_test command handles 16bit depth by default. in which, the parameters "device_rate=16000" and "input_rate=16000" are the same, indicating that the re-sampling function does not need to be enabled; the parameter "input_ch=2" needs to correspond to the original sound source 2ch:

```
rk_mpi_ao_test -i /tmp/sine_16000_500_1000_2ch.pcm --sound_card_name=hw:0,0 --
device_ch=2 --device_rate=16000 --input_rate=16000 --input_ch=2
```

For more command parameter usage, please refer to the "MPI" related documents in the SDK Rockit project directory and the relevant test cases in the mpi example directory.

4.2 Commonly Used Recording Test Commands

The following uses PCM format recording with a sampling rate of 16kHz/2ch/16bit. The rk_mpi_ai_test command also processes 16 bit depth by default. in which, the parameters "device_rate=16000" and "input_rate=16000" are the same, indicating that the re-sampling function does not need to be enabled; the parameters "device_ch=2" and "input_ch=2" are the same, indicating that channel processing is not required. The data generated by recording is the cap_out.pcm file in the /tmp directory.

```
rk_mpi_ai_test --sound_card_name=hw:0,0 --device_rate=16000 --device_ch=2 --
out_rate=16000 --out_ch=2 --output=/tmp
```

For more command parameter usage, please refer to the "AO/AI debugging steps" chapter in "Rockchip_Developer_Guide_MPI_FAQ.pdf".