Conexant CX2072X Audio codec

Linux Audio driver porting guide

14 October 2016 **Document Revision 0.2**

Administrative Information

Only the forms fields' variables should be changed. Note that some fields are used to fill other parts of the document e.g. the title page and the header fields.

Form Field Variable **(double click to change)**	Notes Bookmark *variable name*	 Description
0.2	MyCurrRev	*Current Revision*
Linux driver porting guide	MyProjectName	*Project or Product Name*
14 October 2016	MyCurrRevDate	*Current Revision Date*
SH	MyCurrRevisor	*Name of Current Revisor(s)*
V001	MyCurrDocNumber	*Present on the bottom of all pages*

Revision History

Revision	Revisor	Date	Comments
0.1	sh	9 September 2016	Initial Draft
0.2	sh	10 October2016	Corrected Device Tree information

TABLE OF CONTENTS

Figures and Tables

No table of figures entries found.

1. Scope

The goal of this document is to give a basic information to people who want to add CX20721/3 Audio Codec device driver in Linux or Android platform.

2. Target Audience

This document assumes that the audience familiar with Linux kernel programming.

3. Prerequisites

- CX2072X driver source code.
- CX2072X EVK board.
- Linux kernel version 3.14. source code.

4. References

Table 1: References

Name of Document	Description	Location
ALSA	Advanced Linux Sound Architecture	

5. Definitions, Acronyms, and Abbreviations

Table 2: Definitions of Abbreviation

Word, Acronym, or Abbreviation	Description
ALSA	Advanced Linux Sound Architecture
ASoC	ALSA System on Chip
I2C	Inter-integrated Circuit
PCM	Pulse Code Modulation
I2S	Inter-IC Sound
TDM	Time Division Multiplexing

6. Overview

CX20721/3 is an ultra-low power hifi codec with 2-watt Class-D amplifier. It also features fully programmable parametric EQ and DRC can be used in playback for getting better sound effect. CX20721/3 provides TDM/I2S interface can be used to transmit and receive audio data. and I2C interface can be used to control device.

This documents gives a step by step guide for porting CX20721/3 audio codec to Linux kernel 3.14. but the concepts can be applied to any kernel after kernel 3.8.

TBD.

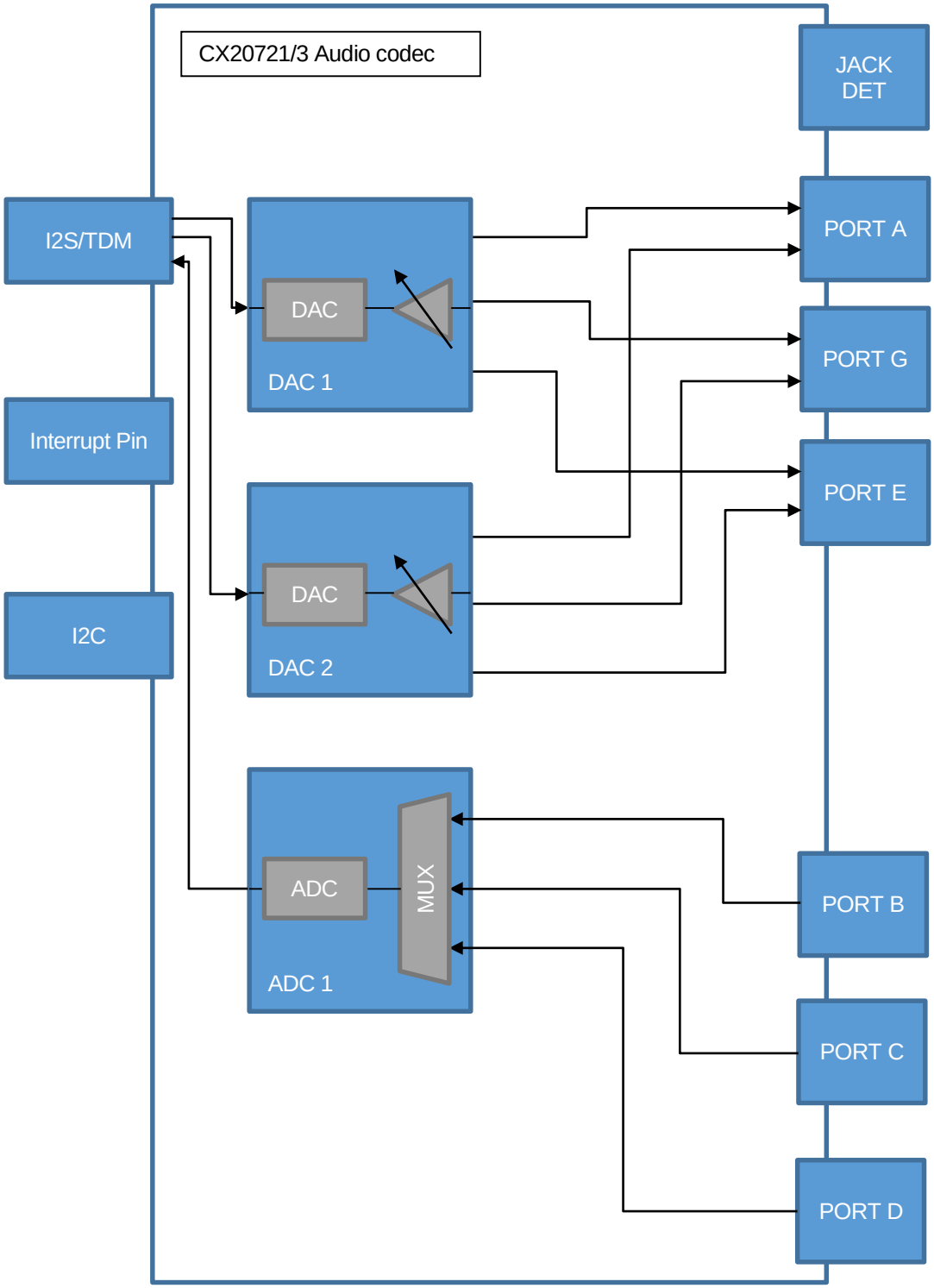
Topic.

- Compile codec driver.
- Modify the board file or Device tree.
- Setting up ASoC machine driver.
- Instantiate I2C device for codec.
- Testing.
- Troubleshooting

.

7. Block Diagram

Note that this block diagram below doesn't contain all components, but only components is used by driver.



8. Driver Features

- Jack sense with headset/headphone deamination.
- Stereo Headphone output {Port A}.
- Stereo analog line output {Port E}.
- 2 W Class-D speaker output with SpeakerShield {Port G}
- Stereo analog line/mic in input {Port B}
- Stereo digital mic input {Port C}
- Headset Mic input {Port D}
- Auto power saving.
- Speaker DRC/EQ.
- Supported sample rate: 48 Khz.
-

9. Step by Step Instructions.

9.1. Integrate codec driver into kernel tree.

1. copy both cx2072x.h and cx2072x.c in the ASoC/Codec folder.

```
sound/soc/codec/cx2072x.h
```

```
sound/soc/codec/cx2072x.c
```

2. Modify the **Kconfig** and **Makefile** to allow cx2072x.c can be built into the kernel.

```
sound/soc/codec/Kconfig
sound/soc/codec/Makefile
```

Add the following statements in RED in file **Kconfig**.

```
Diff –git a/sound/soc/codecs/Kconfig b/sound/soc/codecs/Kconfig

 config SND_SOC_ALL_CODECS
      tristate "Build all ASoC CODEC drivers"

+     select SND_SOC_CX2072X if I2C


 CONFIG SND_SOC_CX20442 if TTY
+CONFIG SND_SOC_CX2072X
+     tristate
```

Add the following statements in file **Makefile**.

```
snd-soc-cx2072x-objs := cx2072x.o
obj-$(CONFIG_SND_SOC_CX2072X) += snd-soc-cx2072x.o
```

3. Enable SND_SOC_ALL_CODECS options.

9.2. Declare the codec via device tree.

Approach A. Declare the codec via Device Tree.

Unlike PCI or USB devices. The audio codec is not enumerated at hardware level. Instead, the software must know which device are connected on each I2C bus, and what address are using. In past, we add platform device through board configuration file. If platform supported Device Tree. We can instantiate codec device through Device Tree.

Adding the following information into device tree to instantiate the codec device.

The device id of codec driver could be one of below.

```
        { .compatible = "cnxt,cx20721", },
        { .compatible = "cnxt,cx20723", },
        { .compatible = "cnxt,cx7601", },
```

The I2C address of codec is 0x33 and max supported speed is 400Khz.

```
i2c1: i2c@400a0000 {
        /* ... master properties skipped ... */
        clock-frequency = <400000>;
        cx2072x@33 {
                compatible = "cxnt,cx2072x";
                reg = <0x33>;
        };
};
```

Approach B. Declare the codec via Board file

9.2.1.1. Modify the boards' file that defines platform devices.

Find and modify current board file (the file under path like /arch/arm/mach-xxxx/) to include audio device driver.

Type	Value
I2C Address	33h
I2C ID	cx2072x
GPIO	For headset plug/unplug interrupt, edge trigger.

Note: Above information might be specified in Device Tree file rather than board file depend on platform design.

9.3. Setting up the Machine driver.

There are two approaches to setting up the Machine driver. one is to use Dummy codec driver, another one is to create a custom machine driver.

The codec dai name is

```
.name = "cx2072x-hifi",
```

Option A. Using Dummy codec driver.

TBD.

Option B. Using custom machine driver.

Create or edit the ASoC Machine driver to glue the Codec and SoC driver.

1. Modify the soc_dai_link

Modify the *.codec_dai_name* and *.dai_fmt* field in *soc_dai_link* structure as below.

```
static struct snd_soc_dai_link dai_cx20723 = {
    .codec_dai_name = "cx2072x",
    .dai_fmt = SND_SOC_DAIFMT_I2S | SND_SOC_DAIFMT_CBS_CFS |
               SND_SOC_DAIFMT_NB_NF,
    …SKIP…
};
```

2. Add the DAPM widgets

```
/* Machine dapm widgets */
static const struct snd_soc_dapm_widget cx20723_dapm_widgets[] = {
    SND_SOC_DAPM_HP("Headphone Jack", NULL),
    SND_SOC_DAPM_SPK("Ext Spk", NULL),
    SND_SOC_DAPM_MIC("Headset Mic", NULL),
    SND_SOC_DAPM_MIC("Internal Mic", NULL),
};
```

3. Setting MCLK frequency to codec.

Using *snd_soc_dai_set_sysclk* to configure the codec MCLK frequency, for example.

```
/* Setting codec MCLK frequency */
snd_soc_dai_set_sysclk(codec_dai, 0, CX2072X_MCLK_EXTERNAL_PLL,
SND_SOC_CLOCK_IN);
```

a. Add DAPM routing.

```
/* machine audio routing to the codec pins */
static const struct snd_soc_dapm_route audio_map[] = {
    {"Headphone", NULL, "PORTA"},
    {"Ext Spk", NULL, "PORTG"},
    {"PORTC", NULL, "Headset Mic"},
    {"PORTD", NULL, "Mic Jack"},
};
```

4. Supported I2S/PCM format

- **Voltage:** 1.8 V, 3.3V
- **Bit clock edge synchronization:** Rising or Falling
- **Bit/Frame Sync clock direction:** In or Out
- **Data order :** LSB or MSB first
- **Formats:** I2S, Sony, Left/Right justified, DSP, PCM short frame sync, PMC long frame snyc, PDM.
- **Sample Rate:**

  48KHZ
- **Sample Size:** 8-Bits, 16-Bits and 24-Bits
- **Frame Size:** From 8 clocks to 256 clocks
- **Multichannel configurations:** 1 to 6 Channels

5. Testing

If driver is porting correctly, you should

```
[    4.006390] ALSA device list:
[    4.006392]   #0: XXXXX
```

Where XXXXX is the audio device name.

You can play the ALSA sample wave file by the following command.

```
root@platfrom:~# aplay /user/share/sounds/alsa/Front_Left.wav
```

10. Amixer commands

Setting up the audio routing on the board.

Playback through Port A

```
amixer sset 'PortA Mux' 'DAC1 Switch'
```

Playback through Port G

```
amixer sset 'PortG Mux' 'DAC1 Switch'
```

Selects recording source.

```
amixer sset 'ADC1 Mux' 'PortB Switch'      #Select Port B port as input.
amixer sset 'ADC1 Mux' 'PortC Switch'      #Select Port C port as input.
amixer sset 'ADC1 Mux' 'PortD Switch'      #Select Port D port as input.
```

Sets playback gain

```
amixer sset 'DAC1 Volume' 74     # To max
amixer sset 'DAC1 Volume' 0      # Mute
```

Sets ADC gain

```
amixer sset 'PortB ADC1 Volume' 74     # Set port B gain to max, range from 0 to 74.
amixer sset 'PortC ADC1 Volume' 74     # Set port C gain to max
amixer sset 'PortD ADC1 Volume' 74     # Set port D gain to max
```

Sets BOOST gain

```
amixer sset 'PortB Boost' 3     # Set port B gain to max, range from 0-3
amixer sset 'PortC Boost' 3     # Set port C gain to max, range from 0-3
amixer sset 'PortD Boost' 3     # Set port D gain to max, range from 0-3
```

Switch EQ/DRC effect.

```
amixer sset 'Playback DSP Switch' 1     # Enables EQ/DRC
amixer sset 'Playback DSP Switch' 0     # Disables EQ/DRC
```

11. Troubleshooting

TBD

12. References:

ALSA Project
ALSA SoC project