

TLV320AIC310x Linux ASoC Audio Driver

ABSTRACT

The TLV320AIC310x audio driver has been developed with an I2C control interface and I2S audio streaming. The code was tested on a Revision C OMAP3530 Evaluation Module running on Linux PSP release 2.1.1.7. This datasheet discusses the CODEC ASoC driver, including the hardware connections between the AIC310x EVM and the Mistral OMAP3530 EVM, and the respective installation.

Table of contents

1	Introduction.....	3
2	Connections.....	3
3	Device Driver.....	5
4	Features Supported	7
5	Driver Sources	9
6	Driver Installation.....	9
6.1	Pre-Requisites.....	9
6.2	Setup	10
6.3	Playback	10
6.4	Record.....	10
6.5	Mixer Controls.....	11
7	Driver Build	15
8	Driver Configuration.....	17
9	Testing	19
9.1	Pre-Requisites.....	19
9.2	Board Setup	19
9.3	Headset Detection	19
9.4	Jumper Settings for Mic and Line-In Recording	19
10	Testing on other platforms.....	20
11	Support for Variant Chipsets	20
12	Known Issues/Caveats in this release	20

1 Introduction

The TLV320AIC310x (also referred to as the AIC310x) is a flexible, low-power, low-voltage stereo audio codec with Programmable inputs and outputs, integrated PLL, and flexible digital interfaces. This device is ideal for Digital Cameras, Smart Cellular Phones, PDAs, Portable Computing, Communication and Entertainment Applications.

The AIC310x codec supports a stereo DAC and a stereo ADC for portable and entertainment applications. The AIC310x Codec supports the following features:

- Microphone Interfaces
- Audio Codec [Stereo ADC and Stereo DAC]
- AGC
- Stereo Headphone/lineout Amplifier
- I2C Control Interface
- Analog Inputs

This datasheet discusses the Linux Audio Driver for the AIC310x codec that was developed to enable users to quickly set up, run, and use the codec device with the TI Linux PSP release 2.1.1.7 based on the 2.6.29 rc3 Linux Kernel.

The AIC310x driver was coded to support the latest ASoC architecture to make to port into different host processors.

The CODEC driver described in this document was developed and tested using the TLV320AIC310x EVM Rev A and OMAP3530 EVM Rev C.

Please note that this document refers to the term AIC310x which means that the Linux Audio Driver sources are compatible with the following members of the AIC310x Codec Family chipsets.

- TLV320AIC3106 Chipset
- TLV320AIC3104 Chipset
- TLV320AIC3101 Chipset
- TLV310AIC31 Chipset

2 Connections

The AIC310x device must be wired and connected to a host processor, where the device driver code is ported and executed. For interfacing with the host processor, we will require two buses (or ports) namely, the control bus and the audio data bus. The control bus on the AIC310x is the I2C serial bus. For the data bus interface, the McBSP from the OMAP35x is utilized.

In developing the AIC310x drivers for this application, the TI AIC310x EVM board and the OMAP3530 EVM platform with the OMAP3 processor were used.

On the I2C-controlled AIC310x, the following digital signals that are essential for running the audio driver are:

- To reset the Audio Codec: RESET

- The I2C bus, two wires: SCL and SDA
- The main audio codec clock: MCLK
- The data bus, three wires: BCLK, WCLK, DIN, DOUT
- GPIO1 Pin used to serve as the Headset Interrupt Pin (For AIC3106).

The wiring diagram describes the wiring details between the OMAP3530 EVM and AIC310x EVM.

SIGNAL ON OMAP3EVM Main Board	PIN NO ON OMAP3 EVM Main Board	DIR	SIGNAL ON AIC 3104/3106 EVM	PIN NO. ON AIC 3104/3106 EVM
I2C2_SDA	P7.A30	<---->	SDA	J5A.20 / J17A.20
I2C2_SCL	P7.A31	---->	SCL	J5A.16 / J17A.16
SYS_CLKOUT2	P7.B52	---->	MCLK	J5A.17 / J17A.17
MCBSP1_CLKX	P7.A21	<-----	BCLK	J5A.3 / J17A.3
MCBSP1_CLKR	P7.A16	<-----	BCLK	J5A.3 / J17A.3
MCBSP1_DX	P7.A18	---->	DIN	J5A.11 / J17A.11
MCBSP1_FSX	P7.A20	<-----	WCLK	J5A.7 / J17A.7
MCBSP1_FSR	P7.A17	<-----	WCLK	J5A.7 / J17A.7
MCBSP1_DR	P7.A19	<-----	DOUT	J5A.13 / J17A.13
GND	P7.A54	<---->	DGND	J3A.5 / J15A.5
VBAT	P7.A1	<---->	+5 VA	J3A.3 / J15A.3
VIO_1v8	P7.A3	<---->	+1.8 VD	J3A.7 / J15A.7
			+3.3 VD	J3A.9 / J15A.9
GPMC_WAIT3/G PIO_65	P7.A46	---->	/RESET	J4A.8 / J16A.8
SYS_CLKOUT1/G PIO_10	P7.B53	<-----	GPIO1	J4A.2 / J16A.2

Table 1: Connection Diagram between OMAP3530 EVM and AIC310x EVM

NOTE: J13, J14, J15, J16 and J17 on the AIC3106 EVM are called J1, J2, J3, J4 and J5 respectively on the AIC3104 EVM. They have the same pin-out and are both set to interface with the USBMOD-EVM.

3 Device Driver

The codec driver for AIC310x is present under "linux-02.01.01.07/sound/soc/codecs" path in the Linux Kernel.

The OMAP3 ASoC Platform driver is present under "linux-02.01.01.07/sound/soc/omap". This CODEC driver has dependency on I2C driver. Linux 02.01.01.07 Kernel has the adapter driver for OMAP3 I2C controller. The codec driver registers the client driver with the I2C core for communication with AIC310x. Platform Driver has dependency on McBSP driver. Linux 02.01.01.07 has driver support for McBSP.

From a hardware standpoint, the AIC310x audio driver must have both I2C and data buses (for audio control and audio data streaming, respectively). The I2C bus controls the audio codec operation by writing to the AIC310x audio control registers; the McBSP interface transfers audio data between the host and the AIC310x. Additionally, the AIC310x MCLK pin should receive an external clock for ADC and DAC to operate.

On the host side, the McBSP2 pins are connected to the AIC310x's WCLK, BCLK, DIN and DOUT pins. The sys_clkout2 pin is connected to MCLK, which is programmed to generate a 12MHz clock.

I2C Driver

I2C client driver for AIC310x is written for performing write to AIC310x. In the codec driver, "i2c_add_driver (&AIC310x_i2c_driver);" will add i2c client driver. The adapter driver will in turn call "AIC310x_codec_probe". This function will probe for AIC310x and if present will register the i2c client through "i2c_attach_client" function call.

MCBSP DRIVER

The McBSP driver for omap3 is present under "linux/arch/arm/plat-omap/" path in the Linux tree. For testing on omap3, AIC310xEVM is wired with McBSP1 interface. McBSP is configured as Master and is programmed to operate in I2S mode.

DMA DRIVER

McBSP driver utilizes DMA interface for transferring the data from the data buffer to McBSP interface. DMA driver is available under "linux/arch/arm/plat-omap/" path in the Linux tree. McBSP uses two logical channels (which ever are available) and utilizes chaining feature for transferring continuous transfers.

CODEC DRIVER

AIC310x ASoC Codec driver consists of two files tlv320aic310x.c and tlv320aic310x.h available under "linux/sound/soc/codecs" path in the Linux tree. tlv320aic310x.c file contains the driver code for initializing the codec, controls for changing the sampling frequency, playback volume, mute/unmute etc. tlv320AIC310x.h file contains the macros and data structure definitions.

```
struct snd_soc_dai tlv320aic310x_dai = {  
  
    .name = "aic310x",  
  
    .playback = {  
  
        .stream_name = "Playback",  
        .channels_min = 1,  
        .channels_max = 2,  
        .rates = aic310x_RATES,  
        .formats = aic310x_FORMATS, },  
  
    .capture = {  
  
        .stream_name = "Capture",  
        .channels_min = 1,  
        .channels_max = 2,  
        .rates = AIC310x_RATES,  
        .formats = AIC310x_FORMATS, },  
  
    .ops = {  
  
        .hw_params = aic310x_hw_params,  
        .digital_mute = aic310x_mute,  
        .set_sysclk = aic310x_set_dai_sysclk,  
        .set_fmt = aic310x_set_dai_fmt,  
  
    }  
};
```

The above data structure informs the capabilities of AIC310x to ASoC core layer. *channels_min* and *channels_max* specify the number of channels supported. AIC310x_RATES macro indicates the sampling frequencies supported by the driver. AIC310x_FORMATS macro indicates the data format supported by the driver. ops contains callback functions for mute/unmute, setting the sampling frequency and data formats.

4 Features Supported

This section provides the list of the features supported by the AIC310x Linux ASoC Audio Driver.

Feature	Description	Remarks
Sampling Rates	Playback: 8000Hz, 11025Hz, 16000Hz, 22050Hz, 44100Hz, 48000Hz. Recording: 8000Hz, 11025Hz, 16000Hz, 22050Hz, 44100Hz, 48000Hz.	N/A.
Audio Formats	Playback: Mono-8 Bit, Mono-16 Bit, Stereo-8 Bit, Stereo-16 Bit Recording: Mono- 8 Bit, Mono-16 Bit, Stereo- 8 Bit and Stereo-16 Bit	Linux Supports 16-bit only.
File formats supported for Playback	Waveform Audio (.wav)	N/A
File formats supported for Recording	.wav	N/A
Playback Volume Control	It can be controlled between the values -63.5dB to +0dB	amixer command is used to exercise this feature.
Volume Control	Headphone Volume Control: It can be changed between the values 0dB to -78.3dB . The gain for Left Headphone Amplifier and Right Headphone Amplifier can be changed independently. Receive-Out Gain Control: It can be changed between the values 0dB to 6dB . The gain for Left Receiver Out Amplifier and Right Receiver Out Amplifier can be changed independently.	amixer command is used to exercise this feature.
MUTE Control	Headphone Mute Control: User can Mute or UnMute the headphone Output. Receiver Out Mute Control: User can Mute or UnMute the Line-Out Output	amixer command is used to exercise this feature. The same values are applied to Left and Right Headphone / Receiver Out
Input Source Selection	User can select Output Control between Line In and On Board	If all following command are set

	MIC using amixer controller	<ul style="list-style-type: none"> ● <code>amixer cset numid=<Mic-in Func> 1</code> then On Board MIC will be selected, If following command are set to ● <code>amixer cset numid=<Line-in Func> 1</code> then Line In will be selected as input
MIC PGA Control	MIC PGA can be varied from 0dB to +59.5dB in 0.5dB steps.	This feature is exercised with amixer command.
Headset Detection	Platform Driver has been updated to configure the GPIO1 pin from Codec into Processor	
Run-time register access	<p>Register Write: User can write to a register in the current page</p> <p>Register Read: User can read value from a register in the current page.</p>	This feature is exercised using amixer cset numid <'Program Register'> Val command.
Master/Slave Modes	Supports building of the Codec sources in either Master or Slave Mode	
Support for variant Chipsets	This release has support for building the source-code for aic3104, aic3106, aic3101, aic31,	Compile-time flags that needs to be enabled.

5 Driver Sources

This section presents the installation steps for running the AIC310x drivers on Linux.

For testing on OMAP3530 Platform, the Linux kernel 02.01.01.07 along with platform & codec driver for TLV320AIC310x is released. AIC310x ASoC Codec driver consists of two files `tlv320AIC310x.c` and `tlv320AIC310x.h` available under "linux/sound/soc/codecs" path in the Linux tree. `tlv320AIC310x.c` file contains the driver code for initializing the codec, controls for changing the sampling frequency, playback volume, mute/unmute etc. `tlv320AIC310x.h` file contains the macros and data structure definitions.

The ASoC platform driver `omap3evm-aic310x.c` file consists of buffer handling and providing required functionality for ASoC generic layer. This file also configures interfaces for data transfer between data buffers and the hardware.

The user has to build the Linux Image and can start testing on OMAP3530 EVM wired with TLV320AIC310x EVM [AIC3106 or the AIC3104 or the AIC3101 Codec EVM].

6 Driver Installation

This section explains the installation of the AIC310x Driver package on Linux platforms.

1. Copy the **AIC310x_asoc_driver.tar** package and extract the same under the \$(LINUX_02.01.01.07_INSTALLATION_FOLDER) folder.
2. Once the **AIC310x_asoc_driver.tar** package is extracted, manually copy the `tlv320AIC310x.c` and `tlv320AIC310x.h` to the "linux/sound/soc/codecs" path in Linux tree, `omap3evm-AIC310x.c` to linux/sound/soc/omap folder.
3. Copy the Makefile to linux/ directory

This section explains the Driver pre-requisites and testing procedure.

6.1 Pre-Requisites

The driver is tested using `aplay` and `amixer` applications. The driver is tested only CODEC as Slave/Master, with I2S protocol and the audio data type as stereo, 16 bits data format

- Arm Linux toolchain version 4.2.0 available from http://www.codesourcery.com/gnu_toolchains/arm/portal/release306.
- From there choose IA32 GNU/Linux Tar ball
- OMAP3530 Revision C EVM
- TLV320AIC310x Revision A EVM [AIC3106 or the AIC3104 or the AIC3101 Codec EVM]

6.2 Setup

- The connections between OMAP3530EVM and TLV320AIC310xEVM is to be made according to the table described in Section 2
- Use the PSP to build the kernel
- Run command "make omap3_evm_defconfig"
- Build Linux image with **Codec as Slave & McBSP as Master** by giving command **"make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- uImage"**
- Boot the newly build kernel and file system on OMAP3530EVM wired with AIC310xEVM.

6.3 Playback

- Plugin the headphone to HEADSET jack (J9) on AIC310xEVM board
- Run command "aplay xyz.wav"
- Run command "amixer cset numid=<id> 15360"
where, id is numid of 'Program Registers'

Example:

```
aplay /testsamples/song_44100.wav &  
amixer cset numid=7 15360
```

6.4 Record

- Plugin the headphone to HEADSET jack (J9) on AIC310xEVM board
- Connect one end of "audio loop-back" cable to host system and other end to LINE IN (J6), which is on AIC310xEVM
- Play audio on host side (through media player)
- Run the following arecord command

```
[root@OMAP3530EVM ~]# arecord -c2 -t wav -f S16_LE -r 44100 -d 20  
rec.wav
```

Recording WAVE 'rec.wav' : Signed 16 bit Little Endian, Rate 44100 Hz,
Stereo

```
[root@OMAP3530EVM ~]#  
[root@OMAP3530EVM ~]#
```

Usage:

```
arecord -c<number of channels> -t <file_type> -f <format> -r  
<sample_rate> -d <Duration> <file_name>
```

Where

Number of channels	2 stereo
file_type	type of the file format (voc, wav, raw or au)
Format	sample format (case insensitive). The recognized sample formats are S8 U8 S16_LE S16_BE U16_LE U16_BE S24_LE S24_BE U24_LE U24_BE S32_LE S32_BE U32_LE U32_BE FLOAT_LE FLOAT_BE

	FLOAT64_LE FLOAT64_BE IEC958_SUBFRAME_LE IEC958_SUBFRAME_BE MU_LAW A_LAW IMA_ADPCM MPEG GSM SPECIAL S24_3LE S24_3BE U24_3LE U24_3BE S20_3LE S20_3BE U20_3LE U20_3BE S18_3LE S18_3BE U18_3LE
<i>sample_rate</i>	sampling rate (8000 Hz, 11025 Hz, 22050 Hz, 44100 Hz, 48000 Hz)
Duration	Duration in seconds
File Name	File name given to recorded audio

6.5 Mixer Controls

Display Mixer control

Usage: amixer controls

Example:

```
root@arago:~# amixer controls
numid=62,iface=MIXER,name='Headphone Function'

numid=51,iface=MIXER,name='Mic Bias Voltage'

numid=12,iface=MIXER,name=' 2nd Control Register'

numid=10,iface=MIXER,name=' 8th Register'

numid=11,iface=MIXER,name=' 9th Control Register'

numid=15,iface=MIXER,name=' D LSB Value Control Register'

numid=14,iface=MIXER,name=' D MSB Value Control Register'

numid=8,iface=MIXER,name=' DAC_SWTCH Control Register'

numid=13,iface=MIXER,name=' J value Control Register'

numid=6,iface=MIXER,name=' LEFT_LOPM Control Register'
'
numid=19,iface=MIXER,name=' Left ADC Powerup- 0x7C'

numid=17,iface=MIXER,name=' MIC2L/R to Left ADC - 0xF0'

numid=18,iface=MIXER,name=' MIC2L/R to Right ADC - 0xF0'

numid=9,iface=MIXER,name=' PLL Control Register'

numid=16,iface=MIXER,name=' R Value Control Register'

numid=7,iface=MIXER,name=' RIGHT_LOPM Control Register'

numid=20,iface=MIXER,name=' Right ADC Powerup - 0x7C'

numid=2,iface=MIXER,name='Constant VCM Output - '
```

numid=21,iface=MIXER,name='DAC Playback Volume'

numid=1,iface=MIXER,name='DAC Power up/down - '

numid=28,iface=MIXER,name='DAC Volume soft stepping'

numid=27,iface=MIXER,name='DAC volume Extra control'

numid=41,iface=MIXER,name='DAC_L1 to HPCOM Anlog Gain Volume Control(0'

numid=37,iface=MIXER,name='DAC_L1 to HPOUT Anlog Gain Volume Control(0'

numid=45,iface=MIXER,name='DAC_L1 to L/R_LOPM Anlog Gain Vol Control(0'

numid=42,iface=MIXER,name='DAC_R1 to HPCOM Anlog Gain Volume Control(0'

numid=38,iface=MIXER,name='DAC_R1 to HPOUT Anlog Gain Volume Control(0'

numid=46,iface=MIXER,name='DAC_R1 to L/R_LOPM Anlog Gain Vol Control(0'

numid=60,iface=MIXER,name='Driver Power-on Time'

numid=64,iface=MIXER,name='External Mic-in Function'

numid=59,iface=MIXER,name='HP / HighPower Output common-mode voltage c'

numid=33,iface=MIXER,name='HPCOM driver Volume(0=0dB, 9=9dB) '

numid=30,iface=MIXER,name='HPCOM driver mute'

numid=4,iface=MIXER,name='HPLOUT Control Register'

numid=32,iface=MIXER,name='HPOUT driver Volume(0=0dB, 9=9dB) '

numid=29,iface=MIXER,name='HPOUT driver mute'

numid=5,iface=MIXER,name='HPROUT Control Register'

numid=26,iface=MIXER,name='Left & Right ADC PGA Gain'

numid=24,iface=MIXER,name='Left ADC Mute'

numid=22,iface=MIXER,name='Left DAC Mute'

numid=47,iface=MIXER,name='Left DAC input selection'

numid=49,iface=MIXER,name='Left DAC output switching'

numid=55,iface=MIXER,name='Left and Right AGC Attack Time Control'

numid=56,iface=MIXER,name='Left and Right AGC Decay Time Control'

numid=54,iface=MIXER,name='Left and Right AGC Maximum PGA control'

numid=57,iface=MIXER,name='Left and Right AGC Noise bounce control'

```
numid=58,iface=MIXER,name='Left and Right AGC Signal Bounce control'
numid=53,iface=MIXER,name='Left and Right AGC Target level control'
numid=52,iface=MIXER,name='Left and Right Audio Gain Control'
numid=3,iface=MIXER,name='Left and right Data Path - '
numid=31,iface=MIXER,name='Left/Right LOPM Mute'
numid=34,iface=MIXER,name='Left/Right_LOP/M driver Volume(0=0dB, 9=0x0'
numid=65,iface=MIXER,name='On Board Mic-in Function'
numid=39,iface=MIXER,name='PGA_L to HPCOM Anlog Gain Volume Control(0='
numid=35,iface=MIXER,name='PGA_L to HPOUT Analog Gain Volume Control(0='
numid=43,iface=MIXER,name='PGA_L to L/R_LOPM Anlog Gain Vol Control(0='
numid=40,iface=MIXER,name='PGA_R to HPCOM Anlog Gain Volume Control(0='
numid=36,iface=MIXER,name='PGA_R to HPOUT Analog Gain Volume Control(0='
numid=44,iface=MIXER,name='PGA_R to L/R_LOPM Anlog Gain Vol Control(0='
numid=61,iface=MIXER,name='Program Registers'
numid=25,iface=MIXER,name='Right ADC Mute'
numid=23,iface=MIXER,name='Right DAC Mute'
numid=48,iface=MIXER,name='Right DAC input selection'
numid=50,iface=MIXER,name='Right DAC output switching'
numid=63,iface=MIXER,name='Speaker Function'
```

Mute/Unmute Control

Usage:

```
amixer cset numid=<id1> x,x
```

id1 is the numid of control with name "Right DAC Mute"

x=0 mute Right DAC

x=1 unmute Right DAC

```
amixer cset numid=<id1> x,x
```

id1 is the numid of control with name "Left DAC Mute"

x=0 mute Left DAC

x=1 unmute Left DAC

Example:

Mute:

```
amixer cset numid=4 0,0
Unmute:
amixer cset numid=4 1,1
```

Volume control

Usage:
amixer cset numid=<id1> x,x
id1 is the numid of control with name "DAC Playback Volume"
x = 0 to 175

Example:
amixer cset numid=1 150,150

Volume Lite control

Usage:
amixer cset numid=<id1> x

id1 is the numid of control with name "Volume_1 (0=-110dB, 116=+6dB)"
x = 0 to 116

Example:
aplay song_44100.wav &
amixer cset numid=46 100

Program Codec Register

Usage:
amixer cset numid=y x
y is the numid of control with name "Program Registers"
x = 0 to 65535

Example:
[root@OMAP3530EVM ~]# amixer cset numid=7 16392
numid=7, iface=MIXER, name='Program Registers'
; type=INTEGER,access=rw-----,values=1,min=0,max=65535,step=0
: values=8

[root@OMAP3530EVM ~]#
Programs Register 64 with value 0x08
Value to mute left dac is 0x08

7 Driver Build

The following steps are useful for testing AIC310x driver on other platform and/or different kernel version. The steps can be used is useful for testing the TLV320AIC310x on any Hardware platform having ASoC platform, machine support and I2C driver.

Also this procedure can be used on any Linux kernel version having ASoC support.

1. Copy the source files (tlv320AIC310x.c tlv320AIC310x.h) under "linux/sound/soc/codecs".
2. Add the below source to the end of "linux/sound/soc/codecs/Kconfig" file

```
config SND_SOC_TLV320AIC310x  
tristate  
depends on SND_SOC
```

3. Add the below source to the end of "linux/sound/soc/codecs/Makefile" file

```
obj-$(CONFIG_SND_SOC_TLV320AIC310x) += tlv320aic310x.o
```

4. Modify the Machine driver to select TLV320AIC310x codec. In the machine driver initialize "codec_dev" member of "struct snd_soc_device" to &soc_codec_dev_aic310x and "codec_dai" member of "struct snd_soc_dai_link" to & aic310x_dai.
5. Update Kconfig and Makefile for the platform driver to use AIC310x codec.
6. Add the below source to the "linux/Makefile" file to support AIC310x as Master & McBSP as Slave
 - a. ifeq (\$(CONFIG_AIC310x_MASTER_MCBSP_SLAVE), 1)
Update the KBUILD_CFLAGS with the this flag
-DAIC310x_MCBSP_SLAVE

NOTE:

The interface between the platform driver and codec driver is through *struct snd_soc_codec_dai*. The platform driver calls the codec functions thorough function pointers hw_params, digital_mute, set_sysclk, set_fmt are all function pointers

```
struct snd_soc_dai tlv320aic310x_dai = {  
  
    .name = "AIC310x",  
  
    .playback = {  
  
        .stream_name = "Playback",  
        .channels_min = 1,  
        .channels_max = 2,  
        .rates = AIC310x_RATES,  
        .formats = AIC310x_FORMATS, },  
  
}
```

```
.capture = {  
  
    .stream_name = "Capture",  
    .channels_min = 1,  
    .channels_max = 2,  
    .rates = AIC310x_RATES,  
  
    .formats = AIC310x_FORMATS, },  
  
.ops = {  
  
    .hw_params = AIC310x_hw_params,  
    .digital_mute = AIC310x_mute,  
    .set_sysclk = AIC310x_set_dai_sysclk,  
    .set_fmt = AIC310x_set_dai_fmt,  
  
    }  
  
};
```

The Linux Kernel can have many codec drivers and all the codec drivers export object of type *struct snd_soc_codec_dai*

Example: struct snd_soc_codec_dai tlv320aic310x_dai exported by AIC310x codec driver

The programmer should change the machine driver depending on his platform/hardware to choose among many codec drivers

Below is code from machine driver for OMAP3530 using AIC310x codec driver

```
static struct snd_soc_dai_link omap3evm_dai = {  
  
    .name = "TLV320AIC310x",  
    .stream_name = "AIC310x",  
    .codec_dai = &tlv320aic310x_dai,  
    .init = omap3evm_AIC310x_init,  
    .cpu_dai = &omap_mcbasp_dai[0],  
    .ops = &omap3evm_ops,  
  
};  
  
  
static struct snd_soc_card snd_soc_card_omap3evm = {  
  
    .name = "OMAP3EVM",  
    .platform = &omap_soc_platform,  
    .dai_link = &omap3evm_dai,  
    .num_links = 1,  
  
};
```



```
static struct snd_soc_device omap3evm_snd_devdata = {

    .card = &snd_soc_card_omap3evm,
    .codec_dev = &soc_codec_dev_AIC310x,

};
```

8 Driver Configuration

The driver has support for following MCLK frequencies 12MHz. On OMAP35x platform, the testing was conducted with MCLK frequency configured with 12MHz.

For supporting new MCLK frequencies, AIC310x_divs[] array in tlv320AIC310x.c needs to be updated. For that we need to calculate the value of K (ie J.D), P and R using the below formula

$$FsRef = (PLLCLK_IN * R * K) / (P * 2048)$$

Where R=1 which is fixed in this driver.

Choose value of P so that $10MHz \leq PLLCLK_IN / P \leq 20MHz$

As MCLK is 12MHz P is also fixed to 1 for this driver. If it is different from 12MHz value then users are advised to reprogram the Value of P in the register 3 of Page-0.

Once P is fixed calculate K for Different FsRef values given in the following table

FsRef	Required for sampling frequency
90316800	11025Hz, 22050Hz, 44100Hz
98304000	8KHz, 12KHz, 16KHz, 24KHz, 48KHz

Ex: For MCLK=12MHz, FsRef=90316800Hz and P=1

$$K = PLL_CLK / PLL_CLKIN$$

$$K = PLL_CLK / MCLK$$

$$K = (90316800 * 1) / 12000000$$

$$K = 7.5264$$

Entry in aic310x_divs array in tlv320aic310x.c

```
/* mclk   rate   P   J   D   ndac   nadc */
/* 11.025k rate */
{12000000, 11025, 1, 7, 5264, 6, 6},
```

Ex: For MCLK=12MHz, FsRef=98304000 and P=1

$K = (98304000 * 1) / 12000000$
 $K = 8.1920$

Entry in aic310x_divs array in AIC310xAudio.c

```
/* 8k rate */  
{12000000, 8000, 1, 8, 1920, 10, 10},
```

9 Testing

This section provides more information on the testing of the AIC310x Audio Driver.

9.1 Pre-Requisites

- Linux 02.01.01.07 Kernel Image integrated with the AIC310x Audio Driver.
- OMAP35x EVM interfaced with the AIC310x EVM for enabling the Image download and testing.

9.2 Board Setup

This section explains the board setup and instructions to be followed before testing the AIC310x Codec Driver.

- The connections between OMAP3530EVM and TLV320AIC310x EVM are to be made according as per Section 2 of this document.
- Download the newly built Linux uImage into the OMAP3530 EVM wired with AIC310xEVM.
- With this user will be able to use Headphone Output (J9) for playback functionality and on board MIC for recording functionality.

9.3 Headset Detection

For testing the Headset detection, please update the jumpers on the AIC310x Evaluation Module as shown below.

Jumper	Position
JMP2	Should be removed
JMP3	Should be removed

And insert the headset into Line-in Jack.

9.4 Jumper Settings for Mic and Line-In Recording

Following jumper settings are common for Line-in and MIC in recording

Jumper	Position
JMP2	
JMP3	

For recording through MIC place W2 and W3, and for line-in recording remove W2 and W3.

10 Testing on other platforms

User has to make hardware modifications as supported by their platform and has to test Playback and recording functionalities.

11 Support for Variant Chipsets

This release of the Codec Driver supports four different variant chipsets of the AIC310x family. The codec library in the path `sound/soc/codecs` contains code for TLV320AIC3104, TLV320AIC3106, TLV320AIC3101 and TLV320AIC31 chipsets. However the code for a particular codec is controlled by the compilation flag to be enabled manually the developer.

- For building the Linux Audio Driver for AIC3104 chipset, please enable the AIC3104_CODEC_SUPPORT macro in the `soc/codecs/tlv320aic310x.h` header file.
- For building the Linux Audio Driver for AIC3106 chipset, please enable the AIC3106_CODEC_SUPPORT macro in the `soc/codecs/tlv320aic310x.h` header file.
- For building the Linux Audio Driver for AIC3101 chipset, please enable the AIC3101_CODEC_SUPPORT macro in the `soc/codecs/tlv320aic310x.h` header file.
- For building the Linux Audio Driver for AIC31 chipset, please enable the AIC31_CODEC_SUPPORT macro in the `soc/codecs/tlv320aic310x.h` header file.

12 Known Issues/Caveats in this release

- For the ADC section, there are no analog routing related registers and hence the DAPM testing for them have not been conducted.