One Technology Way · P.O. Box 9106 · Norwood, MA 02062-9106 · Tel: 781.329.4700 · Fax: 781.461.3113 · www.analog.com

ADAU1761 Sound CODEC Linux Driver

Supported Devices

- ADAU1361
- ADAU1461
- ADAU1761
- ADAU1961

Reference Circuits

- CN0078
- CN0207
- CN0219
- CN0296

Evaluation Boards

- EVAL-ADAU1361Z
- EVAL-ADAU1761Z

Source Code

Status

Source	Mainlined?
git	In progress

Files

Function	File
driver	sound/soc/codecs/adau1761.c
driver	sound/soc/codecs/adau17x1.c
include	sound/soc/codecs/adau17x1.h
include	include/linux/platform_data/adau17x1.h

Example device initialization

For compile time configuration, it's common Linux practice to keep board- and application-specific configuration out of the main driver file, instead putting it into the board support file.

For devices on custom boards, as typical of embedded and SoC-(system-on-chip) based hardware, Linux uses platform_data to point to board-specific structures describing devices and how they are connected to the SoC. This can include available ports, chip variants, preferred modes, default initialization, additional pin roles, and so on. This shrinks the board-support packages (BSPs) and minimizes board and application specific #ifdefs in drivers.

21 Oct 2010 15:10 · Michael Hennerich

I2C

Declaring I2C devices

Unlike PCI or USB devices, I2C devices are not enumerated at the hardware level. Instead, the software must know which devices are connected on each I2C bus segment, and what address these devices are using. For this reason, the kernel code must instantiate I2C devices explicitly. There are different ways to achieve this, depending on the context and requirements. However the most common method is to declare the I2C devices by bus number.

This method is appropriate when the I2C bus is a system bus, as in many embedded systems, wherein each I2C bus has a number which is known in advance. It is thus possible to pre-declare the I2C devices that inhabit this bus. This is done with an array of struct i2c_board_info, which is registered by calling i2c_register_board_info().

So, to enable such a driver one need only edit the board support file by adding an appropriate entry to i2c_board_info.

For more information see: Documentation/i2c/instantiating-devices

21 Oct 2010 15:10 · Michael Hennerich

The I2C device id depends on the ADDR0 and ADDR1 pin settings and needs to be set according to your board setup.

ADDR1	ADDR0	I2C device id
0	0	0x38
0	1	0x39
1	0	0x3a
1	1	0x3b

In this example we assume ADDR0=0 and ADDR1=0.

```
static struct i2c_board_info __initdata bfin_i2c_board_info[] = {

        [--snip--]
        {
                I2C_BOARD_INFO("adau1761", 0x38),
        },
        [--snip--]
}
```

```
static int __init stamp_init(void)
{
        [--snip--]
        i2c_register_board_info(0, bfin_i2c_board_info,
                                ARRAY_SIZE(bfin_i2c_board_info));
        [--snip--]

        return 0;
}
arch_initcall(board_init);
```

ASoC DAPM Widgets

Name	Description	Configuration
LAUX	Left Channel Single-Ended Auxiliary Input	
RAUX	Right Channel Single-Ended Auxiliary Input	
LINP	Left Channel Noninverting Input or Single-Ended Input 0	
LINN	Left Channel Inverting Input or Single-Ended Input 1	
RINP	Right Channel Noninverting Input or Single-Ended Input 2	
RINN	Right Channel Inverting Input or Single-Ended Input 3	
LOUT	Left Line Output	
ROUT	Right Line Output	
LHP	Left Headphone Output	
RHP	Right Headphone Output	

MONOOUT	Mono Output	Headphone mode not capless
DMIC	Digital Microphone in	JACKDET/MICIN pin configured for DMIC
MICBIAS	Bias Voltage for Electret Microphone	

ALSA Controls

Name	Description	Configuration
Digital Capture Volume	Digital volume attenuation for input from either the ADC or the digital microphone input.	
Digital Playback Volume	Digital volume attenuation for output from the DAC	
ADC High Pass Filter Switch	Enable/Disable ADC high-pass-filter	
Playback De-emphasis	Enable/Disable Playback de-empahsis	
Capture Boost	Mixer amplifier bias boost Valid values: “Normal operation”, “Boost Level 1”, “Boost Level 2”, “Boost Level 3”	
Mic Bias Mode	Microphone bias. Valid values: “Normal operation”, “High performance”	
DAC Mono Stereo	DAC mono mode. Valid values: “Stereo”, “Mono Left Channel (L+R)”, “Mono Right Channel (L+R)”, “Mono (L+R)”	
Capture Mux	Selects the source for the capture I2S audio data stream, can either be the DSP or the ADC/DMIC	ADAU1761 only
DAC Playback Mux	Selects the source for the DACs, can either be from the I2S audio interface or the DSP	ADAU1761 only
Input 1 Capture Volume	Gain for single-ended input from the LINP pin	Single-ended inputs
Input 2 Capture Volume	Gain for single-ended input from the LINN pin	Single-ended inputs
Input 3 Capture Volume	Gain for single-ended input from the RINP pin	Single-ended inputs
Input 4 Capture Volume	Gain for single-ended input from the RINN pin	Single-ended inputs
Capture Volume	Differential PGA input volume	Differential inputs
Capture Switch	Mute/Unmute differential input	Differential inputs
Aux Capture Volume	Single-ended auxiliary input gain in the record path	
PGA Boost Capture Volume	Differential PGA input gain boost	Differential inputs
Headphone Playback Volume	Headphone volume	
Headphone Playback Switch	Mute/Unmute Headphone signal	
Lineout Playback Volume	Lineout volume	
Lineout Playback Switch	Mute/Unmute Lineout signal	
ADC Bias	ADC bias. Valid values: “Normal operation”, “Extreme powersaving”, “Enhanced performance”, “Power saving”	

DAC Bias	DAC bias. Valid values: “Normal operation”, “Extreme powersaving”, “Enhanced performance”, “Power saving”	
Capture Bias	Record path bias. Valid values: “Normal operation”, “Enhanced performance”, “Power saving”	
Playback Bias	Playback path bias. Valid values: “Normal operation”, “Enhanced performance”, “Power saving”	
Headphone Bias	Headphone bias. Valid values: “Normal operation”, “Extreme powersaving”, “Enhanced performance”, “Power saving”	
Right LR Playback Mixer Left Volume	Left Playback Mixer gain to the Right LR Playback Mixer	
Right LR Playback Mixer Right Volume	Right Playback Mixer gain to the Right LR Playback Mixer	
Left LR Playback Mixer Left Volume	Left Playback Mixer gain to the Left LR Playback Mixer	
Left LR Playback Mixer Right Volume	Right Playback Mixer gain to the Left LR Playback Mixer	
Right Playback Mixer Left DAC Switch	Mix Right DAC signal into the Right Playback Mixer	
Right Playback Mixer Right DAC Switch	Mix Left DAC signal into the Right Playback Mixer	
Right Playback Mixer Aux Bypass Volume	Auxiliary input gain to the Right Playback Mixer	
Right Playback Mixer Right Bypass Volume	Right Record Mixer gain to the Right Playback Mixer	
Right Playback Mixer Left Bypass Volume	Left Record Mixer gain to the Right Playback Mixer	
Left Playback Mixer Left DAC Switch	Mix Left DAC signal into the Left Playback Mixer	
Left Playback Mixer Right DAC Switch	Mix Right DAC signal into the Left Playback Mixer	
Left Playback Mixer Aux Bypass Volume	Auxiliary input gain to the Left Playback Mixer	
Left Playback Mixer Right Bypass Volume	Right Record Mixer gain to the Left Playback Mixer	
Left Playback Mixer Left Bypass Volume	Left Record Mixer gain to the Left Playback Mixer	
Mono Playback Switch	Mute/Unmute Mono out signal	Headphone mode not capless
Input Select	Select capture path source. Valid values: “ADC”, “DMIC”	DMIC/JACKDETECT pin configured as DMIC
Jack Detect Switch	Enable/Disable Jack insertion detection. If enable the Lineout signals are muted if a headphone is inserted.	DMIC/JACKDETECT pin configured as JACKDETECT

PLL configuration

The ADAU1761 features one PLL:

```
enum adau17x1_pll {
    ADAU17X1_PLL
};
```

The PLL input signal is the MCLK signal.

```
enum adau17x1_pll_src {
    ADAU17X1_PLL_SRC_MCLK,
};
```

(The input frequency must configured to be between 8000000 and 27000000 Hz (8MHz - 27MHz). The output frequency must be configured to be between 45158000 and 49152000. Configuring the PLL with other input or output frequency will fail.)

The PLL runs at 1024 times the base sample rate. So for a 48000 Hz based sample rate you'd normally choose 49152000 Hz for the PLL output frequncey and for a 44100 Hz based sample rate 45158400 Hz.

DAI configuration

The codec driver registers one DAI: **adau-hifi**

Supported DAI formats

Name	Supported by driver	Description
SND_SOC_DAIFMT_I2S	yes	I2S Justified mode
SND_SOC_DAIFMT_RIGHT_J	yes	Right Justified mode
SND_SOC_DAIFMT_LEFT_J	yes	Left Justified mode
SND_SOC_DAIFMT_DSP_A	yes	data MSB after FRM LRC
SND_SOC_DAIFMT_DSP_B	yes	data MSB during FRM LRC
SND_SOC_DAIFMT_AC97	no	AC97 mode
SND_SOC_DAIFMT_PDM	no	Pulse density modulation
SND_SOC_DAIFMT_NB_NF	yes	Normal bit- and frameclock
SND_SOC_DAIFMT_NB_IF	yes	Normal bitclock, inverted frameclock
SND_SOC_DAIFMT_IB_NF	yes	Inverted frameclock, normal bitclock
SND_SOC_DAIFMT_IB_IF	yes	Inverted bit- and frameclock
SND_SOC_DAIFMT_CBM_CFM	yes	Codec bit- and frameclock master

SND_SOC_DAIFMT_CBS_CFM	no	Codec bitclock slave, frameclock master
SND_SOC_DAIFMT_CBM_CFS	no	Codec bitclock master, frameclock slave
SND_SOC_DAIFMT_CBS_CFS	yes	Codec bit- and frameclock slave

DAI sysclk

The DAIs can either use the PLL or the MCLK signal as source.

When using the PLL the DAIs rate should be set to the rate of the PLL. When using MCLK the rate should be set to frequency of the external MCLK signal.

```
enum adau17x1_clk_src {
    ADAU17X1_CLK_SRC_MCLK,
    ADAU17X1_CLK_SRC_PLL,
};
```

When using the MCLK as the DAI source it is possible to use an internal prescaler to divide the signals frequency. Valid divider values are 1, 2, 3 and 4.

Example clock divider configuration:

```
        ret = snd_soc_dai_set_clkdiv(codec_dai, 0, 4);
```

Example DAI configuration

```
static int bfin_eval_adau1x61_hw_params(struct snd_pcm_substream *substream,
        struct snd_pcm_hw_params *params)
{
        struct snd_soc_pcm_runtime *rtd = substream->private_data;
        struct snd_soc_dai *cpu_dai = rtd->cpu_dai;
        struct snd_soc_dai *codec_dai = rtd->codec_dai;
        int pll_rate;
        int ret;

        ret = snd_soc_dai_set_fmt(cpu_dai, SND_SOC_DAIFMT_I2S |
                        SND_SOC_DAIFMT_NB_NF | SND_SOC_DAIFMT_CBM_CFM);
        if (ret)
                return ret;

        ret = snd_soc_dai_set_fmt(codec_dai, SND_SOC_DAIFMT_I2S |
                        SND_SOC_DAIFMT_NB_NF | SND_SOC_DAIFMT_CBM_CFM);
        if (ret)
                return ret;

        switch (params_rate(params)) {
```

```
        case 48000:
        case 8000:
        case 12000:
        case 16000:
        case 24000:
        case 32000:
                pll_rate = 48000 * 1024;
                break;
        case 44100:
        case 7350:
        case 11025:
        case 14700:
        case 22050:
        case 29400:
                pll_rate = 44100 * 1024;
                break;
        default:
                return -EINVAL;
        }

        ret = snd_soc_dai_set_pll(codec_dai, ADAU17X1_PLL,
                        ADAU17X1_PLL_SRC_MCLK, 12288000, pll_rate);
        if (ret)
                return ret;

        ret = snd_soc_dai_set_sysclk(codec_dai, ADAU17X1_CLK_SRC_PLL,
pll_rate,
                        SND_SOC_CLOCK_IN);

        return ret;
}


static struct snd_soc_ops bfin_eval_adau1x61_ops = {
        .hw_params = bfin_eval_adau1x61_hw_params,
};

static struct snd_soc_dai_link bfin_eval_adau1x61_dai = {
        .name = "adau1x61",
        .stream_name = "ADAU1X61",
        .cpu_dai_name = "bfin-i2s.0",
        .codec_dai_name = "adau-hifi",
        .platform_name = "bfin-i2s-pcm-audio",
        .codec_name = "adau1761.0-0038",
        .ops = &bfin_eval_adau1x61_ops,
};
```

TDM configuration

The ADAU1361 and ADAU1761 chips have basic TDM support.

- The number of slots can be either 2, 4 or 8 (ADAU1761 only).
- The slot width can be 32 (ADAU1361 only), 64, 48, 128 or 256 (ADAU1761 only)
- The slot mask must be either 0x03 (slot 0 and 1), 0x0c (slot 2 and 3), 0x30 (slot 4 and 5), 0xc0 (slot 6 and 7)

Example TDM configuration:

```
        ret = snd_soc_dai_set_tdm_slot(codec_dai, 0x30, 0x0c, 8, 64);
```

SigmaDSP Firmware

In order to use the SigmaDSP core of the ADAU1761 you need to provide a firmware file. Please refer to the SigmaDSP Firmware Utility for Linux page on how to generate a firmware file. The firmware file for ADAU1761 driver has to be named **adau1761.bin**.

ADAU1X61 evaluation board driver

There is no dedicated Blackfin STAMP evaluation board for the ADAU1361 or ADAU1761. During test and driver development we used the EVAL-ADAU1361Z and EVAL-ADAU1761Z boards.

It can be easily wired to the Blackfin STAMP SPORT header.

Source

Status

Source	Mainlined?
In progress	In progress

Files

Function	File

driver	sound/soc/blackfin/bfin-eval-adau1x61.c

Kernel configuration

```
Device Drivers  --->
[*] I2C support  --->
[*]   I2C Hardware Bus support  --->
***     I2C system bus drivers (mostly embedded / system-on-chip) ***
<*>       Blackfin TWI I2C support
(100)     Blackfin TWI I2C clock (kHz)
```

Enable ALSA SoC evaluation board driver:

```
Device Drivers  --->
<M> Sound card support  --->
<M>   Advanced Linux Sound Architecture  --->
<M>     ALSA for SoC audio support  --->
<M>       Support for the EVAL-ADAU1X61 boards on Blackfin eval boards
```

Hardware configuration

Connect the STAMP SPORT 0 port (P6) to the EVAL-ADAU1X61 J1 and J6 headers.

Note that the SPORT has separate signals for the capture and playback clocks, while the ADAU1361 uses the same clock signals for both, so the EVAL-ADAU1X61 clock signal pins need to be connected to two STAMP pins each.

STAMP pin	EVAL-ADAU1X61 pin	Function
P6-26 (SPORT 0 - PJ2_SCL)	J1-1	I2C SCL
P6-24 (SPORT 0 - PJ3_SDA)	J1-3	I2C SDA
P6-6 (SPORT 0 - PJ9_TSCLK0), P6-16 (SPORT 0 - PJ6_RSCLK0)	J6-6	BCLK
P6-11 (SPORT 0 - PJ10_TFS0), P6-7 (SPORT 0 - PJ7_RFS0)	J6-8	LRCLK
P6-14 (SPORT 0 - PJ11_DT0PRI	J6-4	Playback data
P6-8 (SPORT 0 - PJ8_DR0PRI)	J6-2	Captrue data
P6-33	J6-1	GND

Driver testing

Load the driver and make sure the sound card is properly instantiated.

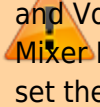
This specifies any shell prompt running on the target

```
root:/> modprobe snd-bf5xx-i2s
root:/> modprobe snd-soc-bf5xx-i2s
root:/> modprobe snd-soc-adau1761
root:/> modprobe snd-soc-bfin-eval-adau1x61
bfin-i2s bfin-i2s.0: dma rx:3 tx:4, err irq:45, regs:ffc00800
dma_alloc_init: dma_page @ 0x02791000 - 256 pages at 0x03f00000
asoc: adau-hifi <-> bfin-i2s.0 mapping ok
ALSA device list:
  #0: bfin-eval-adau1x61
```

This specifies any shell prompt running on the target

```
root:/> modprobe snd-pcm-oss
root:/> tone
TONE: generating sine wave at 1000 Hz...

root:/> arecord -f cd | aplay
Recording WAVE 'stdin' : Signed 16 bit Little Endian, Rate 44100 Hz,
Stereo
Playing WAVE 'stdin' : Signed 16 bit Little Endian, Rate 44100 Hz,
Stereo
```



If you do not hear any sound during testing make sure you have enabled the necessary Switches and Volumes. E.g. for playback on the Headphone output you need to enable the “Left Playback Mixer Left DAC”, “Right Playback Mixer Right DAC” and the “Headphone Playback” switches and set the “DAC Playback Mux” to “AIFIN”.

More information

- SigmaStudio
- SigmaDSP Firmware Utility for Linux

Need Help?

- Analog Devices Linux Device Drivers Help Forum
- Ask a Question

31 Jul 2012 16:53 · Lars-Peter Clausen

www.analog.com