Application note

DA7218 Linux device driver

AN-DA7218_Linux_Mainline_001

Abstract

Descriptions of how to incorporate the source code into the target, build and loading instructions and a list of features provided by this DA7218 driver.

1 Terms and definitions

ALSA	Advanced Linux Sound Architecture
ASoC	ALSA System on Chip
DAI	Digital Audio Interface
DAPM	Dynamic Audio Power Management
DT	Device Tree
IC	Integrated Circuit
I2C	Inter-Integrated Circuit
I2S	Inter-IC Sound
LKML	Linux Kernel Mailing List
PLL	Phase Locked Loop
SRM	Sample Rate Matching
TDM	Time Division Multiplexing

2 Introduction

The purpose of this document is to give an overview of how to incorporate the device driver into the Linux kernel, and a target system. It is expected that anyone reading this document should have a good understanding of the Linux kernel and ALSA framework.

3 Dependencies

This release depends upon a clean, correctly compiled and working Linux kernel mainline tree, tag v4.5-rc1. This external kernel is not provided by Dialog Semiconductor. The kernel is provided by The Linux Kernel Archives (https://www.kernel.org/). The tag v4.5-rc1 can be checked out from the Linux kernel mainline repository.

4 System requirements

- Linux kernel source mainline v4.5-rc1
- Dialog Semiconductor DA7217/8

5 Test status

The device driver was tested successfully with the following setup:

- DA7217/8 Audio IC
- Linux kernel source from mainline v4.5-rc1

6 Features

The following features are supported by the DA7218 Linux audio codec driver:

- I2S/TDM DAI formats
- Master and slave modes for DAI clocking.
- Up to 4 channel microphone audio capture
- Stereo Headphone audio playback
- ALSA Mixer controls for configuration of device

- Jack detection.
- Mic Level Detection.

7 Installation

All DA7217/8 device driver sources come as part of the Linux kernel.

In addition to this, further steps are required to fully integrate the device driver into the kernel tree. This includes the addition of any necessary platform data or device tree files to add the devices into the host platform. Configuration of the Linux build system is also required to include the new components.

8 Compilation

The build instructions for the Linux kernel are provided within the kernel source tree. See the top level Linux kernel README for further instructions.

9 Interfaces

9.1 Device driver interface

This section describes the interfaces supported by the DA7217/8 PMIC driver:

- Control Interface: The driver supports the I2C interface for control of the device.
- Audio Interface: The driver supports the I2S interface for all digital audio data transfer.
- User Interface: The driver conforms to the ALSA SoC Linux framework. It also uses the uevent framework to report Mic Level Detection events.

9.2 Device tree interface

The current support for device tree can be found in the binding documentation. DT bindings for DA7218 are available at the following location. For future DT updates, please refer to the kernel commit logs.

```
./Documentation/devicetree/bindings/sound/da7218.txt
```

10 Driver Configuration Information

10.1 TDM Mode

The driver provides TDM control through the standard ALSA framework. The following are examples of TDM setup of the codec (code usually residing in ALSA ASoC machine code):

Example – 4 slots enabled for codec for 8 slot frame

```
snd_soc_dai_set_tdm_slot(codec_dai,
                         0xf,     /* Enable TDM slots 0-3 */
                         0x0,     /* Data offset into frame (bits) */
                         8,       /* Number of slots in frame */
                         16);     /* Slot width (bits) */
```

The 3rd parameter shown is normally intended as the RX slot configuration, for devices where RX and TX slots are controlled separately. However for DA7217/8 TX and RX slot enabling is tied so this

parameter is instead used to defined the offset (in bits) into the frame for when the data will be transmitted or received for each slot.

10.2 PLL Configuration

10.2.1 MCLK Mode

In this mode the PLL is bypassed and codec internal clocks are divided directly from the master clock provided. This mode can be used for lower power states, when the PLL is not required (i.e. `SND_SOC_BIAS_STANDBY` level of bias in ASoC machine code). This mode can be selected through calls to `snd_soc_dai_set_pll()`, using `DA7218_SYSCLK_MCLK` for the source identifier. For example:

```
snd_soc_dai_set_pll(codec_dai, 0, DA7218_SYSCLK_MCLK, 0, 0);
```

10.2.2 PLL Mode

PLL is enabled in this mode and clocks are derived from MCLK for full sampling rate range. This mode can be selected through calls to `snd_soc_dai_set_pll()`, using `DA7218_SYSCLK_PLL` for the source identifier. `DA7218_PLL_FREQ_OUT_98304` and `DA7218_PLL_FREQ_OUT_90316` define the output frequency parameter for the PLL, and together cover the complete range of sample rates. For example:

```
/* 8|12|16|24|32|48|96KHz sample rate group */
snd_soc_dai_set_pll(codec_dai, 0, DA7218_SYSCLK_PLL, 0,
                    DA7218_PLL_FREQ_OUT_98304);

/* 11.025|22.05|44.1|88.2KHz sample rate group */
snd_soc_dai_set_pll(codec_dai, 0, DA7218_SYSCLK_PLL, 0,
                    DA7218_PLL_FREQ_OUT_90316);
```

10.2.3 PLL SRM Mode

When using the codec driver as the DAI clock slave, it is possible to use the SRM feature which allows the codec's PLL to track the provided clocks from the DAI clock master and synchronise its own internal clocking. This mode can be selected through calls to `snd_soc_dai_set_pll()`, using `DA7218_SYSCLK_PLL_SRM` for the source identifier. `DA7218_PLL_FREQ_OUT_98304` and `DA7218_PLL_FREQ_OUT_90316` define the output frequency parameter for the PLL, and together cover the complete range of sample rates. For example:

```
/* 8|12|16|24|32|48|96KHz sample rate group */
snd_soc_dai_set_pll(codec_dai, 0, DA7218_SYSCLK_PLL_SRM, 0,
                    DA7218_PLL_FREQ_OUT_98304);

/* 11.025|22.05|44.1|88.2KHz sample rate group */
snd_soc_dai_set_pll(codec_dai, 0, DA7218_SYSCLK_PLL_SRM, 0,
                    DA7218_PLL_FREQ_OUT_90316);
```

NOTE: When using the feature, the master provided DAI clocks must be present before the codec driver powers up the relevant audio paths (DAPM enables the required widgets of the codec). Failure to do so can cause audio artefacts to occur.

10.2.4 PLL 32KHz Mode

This mode is reserved for when the master clock provided is 32.768KHz. The PLL tracks this clock to generate the codecs internal clocks. This mode can be selected through calls to `snd_soc_dai_set_pll()`, using `DA7218_SYSCLK_PLL_32KHZ` for the source identifier. `DA7218_PLL_FREQ_OUT_98304` and `DA7218_PLL_FREQ_OUT_90316` define the output frequency parameter for the PLL, and together cover the complete range of sample rates. For example:

```
/* 8|12|16|24|32|48|96KHz sample rate group */
snd_soc_dai_set_pll(codec_dai, 0, DA7218_SYSCLK_PLL_32KHZ, 0,
```

```
                        DA7218_PLL_FREQ_OUT_98304);

/* 11.025|22.05|44.1|88.2KHz sample rate group */
snd_soc_dai_set_pll(codec_dai, 0, DA7218_SYSCLK_ PLL_32KHZ, 0,
                        DA7218_PLL_FREQ_OUT_90316);
```

10.3 Microphone Level Detect

The codec driver provides support for event triggering based on the audio level captured at a chosen microphone, or microphones. This feature could be used for example to provide a voice wake-up mechanism for a system. When the level captured at the microphone is above a chosen threshold, an event is triggered to user-space to be handled accordingly. For further information, please refer to the correct datasheet for the device, which can be obtained through your Dialog Semiconductor contact.

To configure this feature in the codec the following mixer controls are available:

- `Mic Level Detect Level` – configures the threshold at which a level detect event will occur (datasheet describes level settings in further detail)
- `Mic Level Detect Channel1 Switch` – Enable/Disable feature on Channel 1, left and right
- `Mic Level Detect Channel2 Switch` – Enable/Disable feature on Channel 2, left and right

When a level detect channel is enabled using the relevant mixer control, and the associated path is powered for audio capture, the feature will be enabled and events will be reported to user-space, using the Linux uevent framework.

If use of this feature is required during suspend of a target platform, something similar to the following example will also be required in the relevant ALSA ASoC machine code:

```
snd_soc_dapm_ignore_suspend(&codec->dapm, "MIC1");
snd_soc_dapm_ignore_suspend(&codec->dapm, "Capture");
```

This example prevents the DAPM code from powering down the path between “MIC1” and “Capture”, if that path is enabled at the point when system suspend occurs (i.e. audio capture is already taking place). Note that an IRQ line for the codec is required which can wake the system, and the associated DT attribute is provide, as detailed in the bindings, to indicate that the codec should be a wake-up source.

11 Testing

The following sections give examples of how to run basic scenarios with the DA7218 audio codec driver. The examples use the ALSA command line utilities to execute the use-cases described. Note that a full list of mixer controls can be found through the ALSA mixer command line utility (e.g. `amixer controls`).

11.1 Playback

The following is an example of how to execute audio playback through headphones:

```
$ amixer cset name='DMix In DAIL Out FilterL Volume' 20
$ amixer cset name='DMix In DAIR Out FilterR Volume' 20
$ amixer cset name='Out Filter Volume' 120
$ amixer cset name='Out Filter Switch' 1
$ amixer cset name='Mixer Out FilterL DAIL Switch' 1
$ amixer cset name='Mixer Out FilterR DAIR Switch' 1
$ amixer cset name='ST Mixer Out FilterL Out FilterL Switch' 1
$ amixer cset name='ST Mixer Out FilterR Out FilterR Switch' 1
$ amixer cset name='Mixout Volume' 3
$ amixer cset name='Headphone Volume' 30
```

```
$ amixer cset name='Headphone Switch' 1

$ aplay test.wav
```

11.2 Record

The following is an example of how to execute recording of sound from a mono analogue microphone:

```
$ amixer cset name='Mic1 Mux' 'Analog'
$ amixer cset name='Mic1 Volume' 5
$ amixer cset name='Mic1 Switch' 1
$ amixer cset name='In Filter1L Volume' 115
$ amixer cset name='In Filter1L Switch' 1
$ amixer cset name='DMix In Filter1L Out1 DAIL Volume' 25
$ amixer cset name='Mixer DAI1L In Filter1L Switch' 1
$ amixer cset name='Mixin1 Volume' 11
$ amixer cset name='Mixin1 Switch' 1

$ arecord -f S16_LE -c 1 -r 16000 test.wav
```

The following is an example of how to execute recording of sound from a stereo digital microphone:

```
$ amixer cset name='Mic1 Mux' 'Digital'
$ amixer cset name='In Filter1L Volume' 115
$ amixer cset name='In Filter1R Volume' 115
$ amixer cset name='In Filter1L Switch' 1
$ amixer cset name='In Filter1R Switch' 1
$ amixer cset name='DMix In Filter1L Out1 DAIL Volume' 25
$ amixer cset name='DMix In Filter1R Out1 DAIR Volume' 25
$ amixer cset name='Mixer DAI1L In Filter1L Switch' 1
$ amixer cset name='Mixer DAI1R In Filter1R Switch' 1

$ arecord -f S16_LE -c 2 -r 32000 test.wav
```

11.3 Jack Detection

The driver uses the standard ALSA Jack notification interface which can be captured by use of the associated input event device file (e.g. `/dev/input/event1`, number may differ depending on platform). The open-source tools `'evtest'` and `'evemu'` can be used to verify this feature. These are not provided in the release but can be found easily through any web search provider.

11.4 Microphone Level Detection

The following is an example of how to enable the feature and verify events occur as expected, in this case using the generic `'udevadm'` tool to output events (the example makes the assumption that the audio capture path has already been configured, much like the 'Record' example previously described):

```
$ amixer cset name='Mic Level Detect Level' 37
$ amixer cset name='Mic Level Detect Channel1 Switch' 1

$ udevadm monitor --kernel&
$ arecord -f S16_LE -c 2 -r 32000 test.wav&
```

At this point uevents should be displayed when speaking loudly or providing a loud noise source into the mic. If the codec is defined as a wake-up source in the DT schema, then the following step can be performed subsequently to suspend the system, and allows for verification of level detect events triggering wake-up:

```
$ echo mem > /sys/power/state
```

DA7218 Linux device driver

Any sound captured above the set threshold level, after the system is suspended, should trigger wake-up and the associated event to be reported to user-space.

Revision history

Revision	Date	Description
001	16-Jun-2015	Initial version.
002	14-Aug-2015	Added information regarding Mic Level Detection
Mainline_001	2-Feb-2016	Initial LKML release version
Change details: ● Initial version of Application Notes for LKML accepted driver release. ● New document layout		

DA7218 Linux device driver

Status definitions

Status	Definition
DRAFT	The content of this document is under review and subject to formal approval, which may result in modifications or additions.
APPROVED or unmarked	The content of this document has been approved for publication.

Disclaimer

RoHS Compliance

Dialog Semiconductor complies to European Directive 2001/95/EC and from 2 January 2013 onwards to European Directive 2011/65/EU concerning Restriction of Hazardous Substances (RoHS/RoHS2).
Dialog Semiconductor's statement on RoHS can be found on the customer portal https://support.diasemi.com/. RoHS certificates from our suppliers are available on request.

Contacting Dialog Semiconductor

United Kingdom (Headquarters)
Dialog Semiconductor (UK) LTD
Phone: +44 1793 757700

Germany
Dialog Semiconductor GmbH
Phone: +49 7021 805-0

The Netherlands
Dialog Semiconductor B.V.
Phone: +31 73 640 8822

Email:
enquiry@diasemi.com

North America
Dialog Semiconductor Inc.
Phone: +1 408 845 8500

Japan
Dialog Semiconductor K. K.
Phone: +81 3 5425 4567

Taiwan
Dialog Semiconductor Taiwan
Phone: +886 281 786 222

Web site:
www.dialog-semiconductor.com

Singapore
Dialog Semiconductor Singapore
Phone: +65 64 8499 29

Hong Kong
Dialog Semiconductor Hong Kong
Phone: +852 3769 5200

Korea
Dialog Semiconductor Korea
Phone: +82 2 3469 8200

China (Shenzhen)
Dialog Semiconductor China
Phone: +86 755 2981 3669

China (Shanghai)
Dialog Semiconductor China
Phone: +86 21 5424 9058